

AMERICAN UNIVERSITY OF BEIRUT

MACHINE LEARNING OUTPERFORMS
EEFLUX METRIC MODEL ON POINT
ESTIMATES

by

YASMINE HISHAM HAMDAR

A thesis

submitted in partial fulfillment of the requirements
for the degree of Master of Science
to the Department of Computer Science
of the Faculty of Arts and Sciences
at the American University of Beirut

Beirut, Lebanon
January 2021

AMERICAN UNIVERSITY OF BEIRUT

MACHINE LEARNING OUTPERFORMS
EEFLUX METRIC MODEL ON POINT
ESTIMATES

by
YASMINE HISHAM HAMDAR

Approved by:

Dr. Fatima Abu Salem, Associate Professor
Department of Computer Science

Advisor



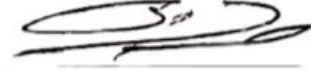
Dr. Hadi Jaafar, Associate Professor
Department of Agriculture

Co-Advisor



Dr. Samer Kharroubi, Associate Professor
Department of Nutrition and Food Science

Committee Member



Dr. Mohamed El Baker Nassar, Assistant Professor
Department of Computer Science

Committee Member



Date of Thesis Defense: January 25, 2021

Acknowledgements

I would first like to point out how proud I am that this work was done in a year which had a revolution, the worst economic inflation, the worst bombing in the history of this country, and a pandemic. Hence, I would like to say that I was very lucky I chose Dr. Fatima Abu Salem as an advisor, because she was not only a mentor, but also a close friend who was very understanding, supportive, motivating, and patient through it all.

I would like to say a special thank you to Dr. Hadi Jaafar for sharing his immense knowledge as well as giving his guidance and assistance as the co-advisor on this project. Also a great thank you goes to the Google AI Impact Challenge project and all the committee members for their great support.

I am very grateful for my family, my friends, and my fiancée whom without I could have never had the love, encouragement, and enthusiasm needed to finish this thesis.

An Abstract of the Thesis of

Yasmine Hisham Hamdar for Master of Science
Major: Computer Science

Title: Machine Learning Outperforms EEFlux METRIC Model on Point Estimates

For a vast number of countries, especially in the Mediterranean area, water shortage is among the most important agricultural and environmental hazards. As water consumption rises, the situation worsens due to a deficient supply of resources because of unnecessary exploitation, deforestation, unfair distribution, water wars, and ill management. More than half of the water supply is used for agricultural purposes and the goal was always to grow more crops to satisfy the demands of a growing population. Not only does agriculture demand the highest water supply, but it also has the highest potential for improving quality. Hence, efficient and smart irrigation would be a necessary step for water preservation.

In this thesis, we developed a full fledged utility based regression module using point-wise, probabilistic, conformal, and quantile regression trained on data gathered from flux towers across America and Europe. Our module aided in predicting evapotranspiration, a metric which allows farmers to know how much water to use for crop irrigation. Adding to that, we implemented a data over-sampling module - SMOGN - which helped in up-sampling data in our regression problem based on rare versus none rare values. Using our probabilistic models, we were able to quantify the uncertainty in our predictions. We also performed several feature selection techniques and studied model interpretability using SHAP and LIME. We developed our research in a way to allow farmers and agricultural experts to choose between a point-wise prediction, a probabilistic and certain prediction, and a prediction interval. We were able to achieve best results using our conformal model, yielding 81% coverage rate, meaning that 81% of the actual data lie in its corresponding prediction interval.

Contents

Acknowledgements	v
Abstract	vi
1 Introduction	1
1.1 Background	1
1.2 Overview	4
2 Literature Review	5
3 Systems and Hardware	9
3.1 Libraries and Frameworks	9
3.2 Environments	9
3.2.1 Google Cloud Platform	10
3.2.2 HPC Octopus Cluster	10
3.2.3 Docker	10
3.3 Code and Data	10
4 Study Area and Surveyed Sites	11
4.1 AmeriFlux	12
4.1.1 Sites	12
4.2 EuroFlux	12
4.2.1 Sites	13
4.3 EEFlux	13
5 Quality Control	15
5.1 Data Cleaning	15
5.1.1 AmeriFlux and EuroFlux	15
5.1.2 EEFlux	15
5.2 Data Correction	16
5.2.1 Bowen’s Ratio	16
5.2.2 Bowen’s Ratio - FluxQAQC	16
5.2.3 Energy Balanced Ratio (EBR)	17
5.3 Data Generation	17

5.3.1	Datasets	18
5.3.2	Data Legend	18
5.3.3	Daily-Joint Mapping	20
5.3.4	Choice of Dataset	21
5.3.5	Final Dataset	21
5.4	Feature Engineering	22
5.4.1	Time-Series Analysis	22
5.4.2	Exploratory Data Analysis	24
5.4.3	Data Transformations	32
5.4.4	Unsupervised Clustering Analysis	36
5.4.5	Seasonality Study	37
6	Utility-Based Regression and Minority Up-sampling	39
6.1	Relevance Function	39
6.2	Defining Rarity	40
6.3	Relevance Matrix and Relevance Threshold	42
6.4	SMOBN	42
6.4.1	SMOTER and Gaussian Noise	43
6.5	Utility-Based Regression and SMOBN Hyper-parameters	45
6.6	Repeated Stratified K-Fold Cross-Validation	45
7	Feature Selection	46
7.1	Methods	46
7.1.1	Feature Importance	47
7.1.2	Mutual Information Feature Selection	47
7.2	Scenarios	48
7.2.1	Scenario A	48
7.2.2	Scenario B	48
7.2.3	Scenario C	49
7.2.4	Scenario D	49
8	Model Assessment	50
8.1	Evaluation Metrics	50
8.1.1	Regression Metrics	50
8.1.2	Correlation Metrics and Agreement Metrics	51
8.1.3	Probabilistic Model Selection Metrics	52
8.1.4	Utility-Based Regression Metrics	53
8.2	Negative Accuracy and R^2 Score	54
8.2.1	R^2 Score	54
8.2.2	MAPE	55

9	Uncertainty Quantification	56
9.1	Uncertainty Types	56
9.1.1	Aleotoric Uncertainty	57
9.1.2	Epistemic Uncertainty	59
10	Experimental Variations	62
10.1	Experiment A	62
10.2	Experiment B	62
10.3	Experiment C	63
10.4	Experiment D	63
11	Point-wise Modeling	64
11.1	Models	64
11.2	Choosing the Best Model	65
11.3	Utility-Based Learning and SMOGN up-sampling	68
11.4	Base Model	69
11.4.1	Architecture	69
11.4.2	Hyper-parameters	70
11.5	Best Model	70
11.5.1	Architecture	70
11.5.2	Hyper-parameters	71
11.6	Implementation	72
11.6.1	Linear SVR Model	72
11.6.2	Gradient Boost	73
11.7	Results	74
11.7.1	Across Weather Clusters and Union of Clusters	74
11.7.2	Across Different Climates	80
11.7.3	Across Different Seasons	85
11.8	Models' Stability and Performance	89
11.8.1	The Model with the Least Training Time	90
11.8.2	Learning Experience	90
12	Probabilistic Modeling	91
12.1	NGBoost	91
12.1.1	Architecture	92
12.1.2	Hyper-parameters	93
12.1.3	Implementation	94
12.2	MC Dropout	96
12.2.1	Architecture	96
12.2.2	Hyper-parameters	96
12.2.3	Implementation	96
12.3	Deep Ensembles	98
12.3.1	Architecture	98

12.3.2	Hyper-parameters	98
12.3.3	Implementation	98
12.4	Choosing the Best Model	99
12.5	Utility-Based Learning and SMOGN Up-sampling	99
12.6	Quantifying Uncertainty	101
12.7	Results	105
12.7.1	Across Weather Clusters and Union of Clusters	105
12.7.2	Across Different Climates	111
12.7.3	Across Different Seasons	115
12.8	Models' Stability and Performance	120
12.8.1	The Model with the Least Training Time	121
12.8.2	Learning Experience	121
13	Conformal Quantile Modeling	122
13.1	Overview	122
13.1.1	Quantile Regression	122
13.1.2	Conformal Prediction	123
13.1.3	Conformalized Quantile Regression	124
13.2	Models	125
13.3	Base Model	125
13.3.1	Architecture	126
13.3.2	Hyper-parameters	126
13.4	Best Model	126
13.4.1	Architecture	127
13.4.2	Hyper-parameters	128
13.5	Implementation	129
13.5.1	Split Conformal Random Forests	129
13.5.2	CQR Neural Networks	130
13.6	Results	136
13.6.1	Optimized vs Non-Optimized CQR Neural Network Model	137
13.6.2	Verifying the Choice of α	137
13.6.3	Effect of SMOGN	139
13.6.4	Feature Selection and Clustering	141
13.6.5	Varying Climates	144
13.6.6	Varying Seasons	144
14	SHAP	146
14.0.1	Dealing with Categorical Data	147
14.1	Point-wise SHAP	147
14.1.1	SHAP Summary Bar Plot	147
14.1.2	SHAP Summary Plot	149
14.1.3	SHAP Decision Plot for All Predictions	150
14.2	Probabilistic SHAP	151

14.2.1	SHAP Summary Bar Plot	151
14.2.2	SHAP Summary Plot	151
14.2.3	SHAP Decision Plot for All Predictions	153
14.3	Comparison between Point-wise and Probabilistic SHAP	153
14.4	Common Observations	154
15	LIME	155
15.1	Observatiions	156
15.1.1	LIME on Two Accurate Data Point	156
15.1.2	Lime on an Inaccurate Data Point	158
15.1.3	Comparison between LIME on Accurate and Inaccurate Data Points	158
15.2	Comparison between SHAP and LIME	159
15.3	Comparison between SHAP, LIME, and Feature Selection	159
16	Deployment	161
17	Conclusion and Future Work	163
18	Appendix	165
18.1	Point-wise Models	166
18.1.1	Gradient Boost	166
18.1.2	Linear SVR	172
18.2	Probabilistic Models	178
18.2.1	NGBoost	179
18.2.2	MC Dropout	187
19	Abbreviations	190

List of Figures

1.1	Overview	4
4.1	Area of Study	12
4.2	Sample Sites Ameriflux	14
5.1	ADF Test Results	23
5.2	White Noise Test Results	23
5.3	Site Distribution	25
5.4	Number of Rows per Site	26
5.5	Distribution of Climates	27
5.6	Seasons Density Plot	28
5.7	Distribution of TA, RH, and WS	29
5.8	Distribution of EEflux Albedo, EEflux NDVI, and EEflux LST	30
5.9	ET	30
5.10	Ameriflux Site US-Goo	31
5.11	Euroflux Site DE-Kli	32
5.12	Auto-correlation Scatter Plot for WS, RH, TA, amd LST	33
5.13	Auto-correlation Scatter Plot for NDVI and Albedo	34
5.14	TA Auto-correlation Plot	35
5.15	Distribution of TA across Clusters	37
6.1	Real ET Value Distribution	41
6.2	Real ET Values Box Plot	41
6.3	Snippet of Rare Study per Site	41
6.4	Relevance per ET Value	42
6.5	SMOTE (Branco et al., 2013)	44
7.1	Feature Importance	47
7.2	Feature Selection based on NMI	48
9.1	Uncertainty Types	57
9.2	Aleotoric Uncertainty - 1 (Why Uncertainty Matters)	58
9.3	Aleotoric Uncertainty - 2 (Why Uncertainty Matters)	58
9.4	Epistemic Uncertainty (Why Uncertainty Matters)	59
9.5	Bayesian Inference (Gustafson, 2014)	61

11.1 Gradient Boosting	71
11.2 Scatter plot for Point-wise Models	77
11.3 Residual Analysis Scatter Plot for Gradient Boost	78
11.4 Density Plot for ET across all Climates for Point-wise Models	83
11.5 Density Plot for ET across all Seasons for Point-wise Models	88
11.6 Gradient Boost Learning Curve	90
12.1 Natural Gradient Boosting (Duan et al., 2020)	93
12.2 ET Distribution	94
12.3 Epistemic Uncertainty for all Scenarios	103
12.4 Epistemic Uncertainty for NGBoost3	104
12.5 Scatter plot for Probabilistic Models	107
12.6 Line plot for a set of sites for NGBoost	108
12.7 Line plot for US-Kon for Probabilistic Models	109
12.8 Residual Analysis Scatter Plot for NGBoost	110
12.9 Density Plot for ET across all Climates for Probabilistic Models	114
12.10 Density Plot for ET across all Seasons for Probabilistic Models	119
13.1 Neural Network Architecture (Musiol, 2016)	128
13.2 Dropout (Wang and Manning, 2016)	135
14.1 SHAP summary bar plot	148
14.2 SHAP summary plot	149
14.3 SHAP decision plot	151
14.4 SHAP summary bar plot	152
14.5 SHAP summary plot	152
14.6 SHAP decision plot	153
15.1 LIME plot	156
15.2 LIME plot	157
15.3 LIME plot	158

List of Tables

4.1	Climate Types	11
4.2	Vegetation Types	11
5.1	Number of Rows in Library Dataset	24
5.2	TA Clusters Distribution	38
7.1	Feature Selection Scenarios	49
11.1	Point-wise Models Results part 1	66
11.2	Point-wise Models Results part 2	67
11.3	SMOGN Comparison	68
11.4	Experimental Results for Point-wise Models	76
11.5	Residual Analysis for Point-wise Models	78
11.6	Experimental Variations Point-wise Models part 1	81
11.7	Experimental Variations for Point-wise Models part 2	82
11.8	Residual Analysis for Point-wise Models	84
11.9	Experimental Variations for Point-wise Models part 1	86
11.10	Experimental Variations for Point-wise Models part 2	87
11.11	Residual Analysis for Point-wise Models	89
12.1	Best Probabilistic Model	99
12.2	SMOGN Comparison	100
12.3	Feature Selection Scenarios	101
12.4	Experimental Results for Probabilistic Models	102
12.5	Experimental Variations for Probabilistic Models	106
12.6	Residual Analysis for Probabilistic Models	110
12.7	Experimental Variations for Probabilistic Models part 1	112
12.8	Experimental Variations for Probabilistic Models part 2	113
12.9	Residual Analysis for Probabilistic Models	115
12.10	Experimental Variations for Probabilistic Models part 1	117
12.11	Experimental Variations for Probabilistic Models part 2	118
12.12	Residual Analysis for Probabilistic Models	120
13.1	Experimental Results	137
13.2	Experimental Results on Miscoverage Rate of 17%	138

13.3	Experimental Results on Miscoverage Rate of 22%	138
13.4	Experimental Results on Miscoverage Rate of 25%	138
13.5	Experimental Results without SMOGN	139
13.6	Experimental Results with SMOGN	139
13.7	Feature Selection Scenarios for CQR and CRF	141
13.8	Experimental Results for $\alpha = 0.17$	142
13.9	Experimental Results on Various Climates for $\alpha = 0.17$	144
13.10	Experimental Results on Various Seasons for $\alpha = 0.17$	145
18.1	Experimental Results on all the Data	166
18.2	Experimental Results on WS Cluster 0	167
18.3	Experimental Results on WS Cluster 1	167
18.4	Experimental Results on All the Data	168
18.5	Experimental Results on WS Cluster 0	168
18.6	Experimental Results on WS Cluster 1	169
18.7	Experimental Results on All the Data	169
18.8	Experimental Results on WS Cluster 0	170
18.9	Experimental Results on WS Cluster 1	170
18.10	Experimental Results on All the Data	171
18.11	Experimental Results on WS Cluster 0	171
18.12	Experimental Results on WS Cluster 1	172
18.13	Experimental Results on All the Data	172
18.14	Experimental Results on WS Cluster 0	173
18.15	Experimental Results on WS Cluster 1	173
18.16	Experimental Results on All the Data	174
18.17	Experimental Results on WS Cluster 0	174
18.18	Experimental Results on WS Cluster 1	175
18.19	Experimental Results on All the Data	175
18.20	Experimental Results on WS Cluster 0	176
18.21	Experimental Results on WS Cluster 1	176
18.22	Experimental Results on All the Data	177
18.23	Experimental Results on WS Cluster 0	177
18.24	Experimental Results on WS Cluster 1	178
18.25	Experimental Results on All the Data	179
18.26	Experimental Results on WS Cluster 0	180
18.27	Experimental Results on WS Cluster 1	180
18.28	Experimental Results on All the Data	181
18.29	Experimental Results on WS Cluster 0	182
18.30	Experimental Results on WS Cluster 1	182
18.31	Experimental Results on All the Data	183
18.32	Experimental Results on WS Cluster 0	184
18.33	Experimental Results on WS Cluster 1	184
18.34	Experimental Results on All the Data	185

18.35	Experimental Results on WS Cluster 0	186
18.36	Experimental Results on WS Cluster 1	186
18.37	Experimental Results on All Data	187
18.38	Experimental Results on WS Cluster 0	188
18.39	Experimental Results on WS Cluster 1	188

Chapter 1

Introduction

1.1 Background

Evapotranspiration (ET) is a metric that measures the degree of water lost from the soil either by evaporation from the soil surface or by transpiration from the plant's leaves. Reliable estimation of ET is essential for water consumption and irrigation water management to combat excessive water loss. Researchers have developed several remote sensing techniques to accurately estimate ET, but its calculation is considered a complex process in which several meteorological variables are associated. This has motivated researchers to use machine learning for predicting ET due to their capability of tracking complex relationships between dependent and independent variables. Traditionally, ET was estimated using models such as those integrated with EEflux (the Landsat-based evapotranspiration tool - Earth Engine Evapotranspiration Flux (EEflux)), an automated calibration process based initially on METRIC, which uses an archive of Landsat imagery maintained on the Google Earth Engine ([Allen, 2005](#)) to estimate EEFlux ET.

In this work, we explore the following research questions:

1. Are machine learning based predictive models more accurate than EEflux models for predicting ET?
2. What machine learning frameworks can outperform classical statistical methods for predicting ET?
3. Can we minimize the bias between the real ET and the EEflux (METRIC) ET?

To this end, we frame our problem as a regression problem capturing temporal and spatial variations on ET across the United States and Europe and explore a variety of predictive frameworks incorporating machine learning models at their

base, including pointwise prediction, Auto Sklearn, probabilistic prediction, and conformal quantile prediction. Our datasets comprise a collection of publicly available remotely sensed data spanning 26 sites from 2000 to 2018 such as real ET values (the response variable) obtained from the Ameriflux and Euroflux towers, as well as weather and remotely-sensed data (LST, NDVI, and ALBEDO) obtained from EEflux. Our data processing incorporates elements for imbalanced learning for regression through the utility-based regression technique, and in our design, we target to improve the recall of our models to capture the utility of identifying extreme values of ET on excessively hot weather and that are most relevant to farmers. Our clustering of the data across various climates and seasons allows for transfer learning from sites with higher air temperature to those with lower air temperature. Our feature selection analysis allows us to further reduce the dimensionality of our problem in such a way to require minimal input whilst maintaining reasonable accuracy measures.

We have trained several point-wise prediction models on our dataset, proving that Gradient Boost is the best model of them all. Gradient Boost yielded an R2 score of 0.637, an RMSE of 1.359, and a recall of 0.917. An improvement in scores is observed when the model is trained on wind speed clusters, yielding an R2 score of 0.654, an RMSE of 1.355. Gradient Boost beats the base learner Linear SVR in R2 by 32% - 33%, and in RMSE by 24% - 26%. Gradient Boost also beats Auto-sklearn in R2 by 6%, and in RMSE by 4% - 45%. Gradient Boost outperforms EEflux (METRIC) ET in R2 by a 100%, in accuracy by 17.144% - 24.408%, and in RMSE by 82%-100%. Nonetheless, Gradient Boost performs best in minimizing the proportional bias, beating Linear SVR by a 100% in R2, and a 28%-60% in RMSE. Hypothesis testing was applied to validate these percentages.

We have also tested a range of probabilistic models. The best probabilistic performing model is NGBoost. NGBoost is a more favorable learner since it has a multitude of advantages including its ability to produce probability distribution predictions over an input data point rather than fixed point predictions. Probabilistic prediction algorithms are more commendatory in real-life business models since they allow room for predictive uncertainty estimations, unlike point-wise models. NGBoost yielded an R2 score of 0.641, an RMSE of 1.368, and a recall of 0.88 when trained on the whole dataset. The NGBoost model witnesses a boost in scores when trained on wind speed clusters, achieving an R2 of 0.686, an RMSE of 1.303, and a recall of 0.91. NGBoost beats MC Dropout in R2 by 2% to 12%, in accuracy by 1% to 2%, and in RMSE by 24% to 26%. NGBoost also beats Auto-sklearn in R2 by 6%, and in RMSE by 29% to 41%. NGBoost outperforms EEflux ET METRIC in R2 by a 100%, in accuracy by 20% - 23.5%, and in RMSE by 82%-100%. Nonetheless, NGBoost performs best in minimizing the proportional bias on the union of clusters, beating MC Dropout by a 16% - in R2, an 11% in RMSE. Hypothesis testing was applied to validate these per-

centages.

We also experimented with models producing prediction intervals (an interval with a lower and upper bound for the predicted variable) rather than point or probabilistic predictions. The best model with the highest prediction interval coverage was conformalized quantile neural networks, yielding a coverage rate (percentage of real ET falling in prediction interval) of almost 81%, and an average prediction interval length of 0.75 mm across the whole data set. Conformal Quantile Neural Networks beat Conformal Random Forests by yielding a lower average prediction interval length by 19%. Prediction intervals are very useful in this business project since they yield a range for ET (mm) rather than an exact quantified measure.

To interpret our best performing probabilistic and pointwise machine learning models (NGBoost and Gradient Boost), we conclude with a SHAP analysis, where it is seen that the top contributing features are air temperature and NDVI. SHAP also showed that high values of air temperature, NDVI, and wind speed push the models to predict high ET values across the pointwise and probabilistic best performing models. SHAP showed, on the other hand, that low LST (land surface temperature) and relative humidity values indicate high predicted ET values.

1.2 Overview

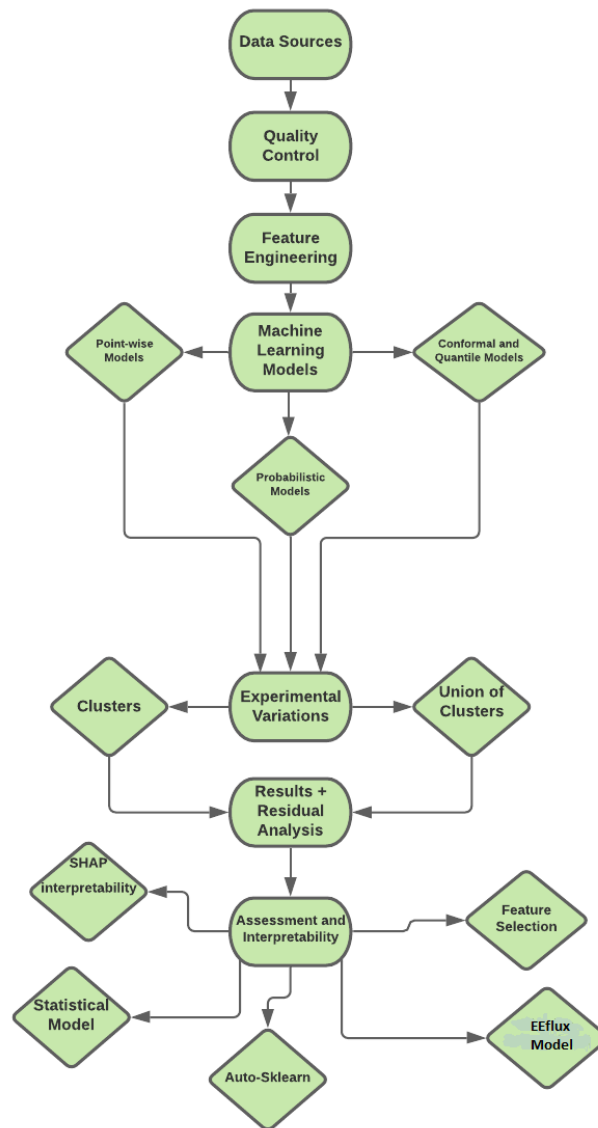


Figure 1.1: Overview

Chapter 2

Literature Review

([Baier and Robertson, 1965](#)) quantified Daily Latent Evaporation from basic meteorological measurements across six research areas in Canada. The data from this research was collected over the years of 1953 to 1957 and meteorological measurements were taken annually from May to October. To discover the effect of several meteorological and astronomical variables, such as maximum temperature, temperature range, wind speed, vapor pressure deficit, solar energy, day length, total sky, solar energy on a horizontal surface, and the period of bright sunlight on latent evaporation, simple and multiple linear correlation and regression were used. The simple correlation indicates that the latent evaporation was more closely associated with solar energy and sunlight than with temperature terms such as maximum and range. In addition, multiple regression analysis was also used to study the significance of the factors involved in forming equations for latent evapotranspiration computation. Results showed that with only the minimum and maximum temperature and extraterrestrial radiation, the correlation coefficient was very significant with an R^2 of 0.68. The correlation coefficient varies from an R^2 of 0.75 to an R^2 of 0.81 upon the addition of one or more solar energy variables, vapor pressure deficit, and wind speed. Results boosted to an R^2 of 0.84 for all six variables.

Furthermore, another research was performed by the authors ([Granata, 2019](#)) to predict the actual ET on a Central Florida location. The data collected happened to be between 28 September 2000 and 28 September 2004. On three different input combinations and on various models, multiple machine learning algorithms (M5P Regression Tree, Bagging Random Forest, and Support Vector Regression) were used. Sensitive-heat flux, net solar radiation, soil moisture content, wind speed, mean relative humidity, and mean temperature were included in Model 1. Net solar radiation, wind speed, mean relative humidity and mean temperature were included in Model 2. Net solar radiation, mean temperature and mean relative humidity were present in Model 3. A 10-fold cross-validation technique was employed where the data is randomly split into 10 subsets with

one set allocated as the validation data. These statistical metrics were used to test the performance of the studied models: Nash-Sutcliffe Model Efficiency Coefficient (NSE), Root Mean Squared Error (RMSE), and Relative Absolute Error (RAE). This study showed that Model 1 provided the best forecast for all error metrics in which an NSE of 0.987, 0.14 mm/day for MAE, 0.179 for RMSE, and 15.4 percent for RAE was obtained for the best-performing algorithm, Random Forests. The results of Models 2 and 3 were very similar but less favorable than Model 1. The proposed development would be to design a more efficient machine learning model to predict real ET using mean temperature, relative humidity, and net solar radiation with more sophisticated algorithms such as the Artificial Neural Network (ANN) or the Extreme Learning Machine (ELM).

An analysis by (Huang et al., 2019) was also done to estimate the actual ET. This paper estimates the daily ET with minimal meteorological data using the CatBoost algorithm and the gradient-boosting decision tree. The product of this algorithm is compared to Support Vector Machine (SVM) and Random Forests (RF). The data used is a mixture of meteorological data that comprises both complete and incomplete combinations of solar radiation, relative humidity, maximum and minimum temperature, and wind speed at multiple weather stations in South China throughout the years 2001 and 2015. The assessment measures used are Root Mean Squared Error (RMSE), Mean Bias Error (MBE), Mean Absolute Percentage Deviation (MAPD), and R2 score. The study showed that CatBoost reported the best accuracy when the full combination of inputs occurs, contrary to the seven other stations with incomplete input variables. The SVM reported the best accurate rating for RMSE (from 4.8 percent when using a portion of the input variable to 37.4 percent when using all the input variables) and MAPE (from 3.3 percent to 33.3 percent). In addition, regarding CatBoost, the memory use and computational burden for data processing are far less, which enables it to be a promising ET0 estimation algorithm.

Nevertheless, a report on estimating the daily ET with minimal meteorological data from 1961-2010 from eight representative weather stations in different climates in China was published by the authors (Fan and Yue, 2019). To test their effectiveness on ET estimation, four different input combinations were tested. These combinations of inputs included minimum and maximum daily temperatures, relative humidity, wind speed, global and extraterrestrial solar radiation, respectively. With four tree-based machine learning models including Random Forests (RF), M5 model tree (M5Tree), Gradient Boosting Decision Tree (GBDT) and Extreme Gradient Boosting (XGBoost), Support Vector Machine (SVM), and Extreme Learning Machine (ELM), a K-fold cross-validation approach was used in which data was split into five folds. Using three statistical measures, the efficiency of the studied models was evaluated: Coefficient of Determination (R2), Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE).

This research revealed that when all the input variables were present, SVM and ELM were the most effective and stable models with the least error. In addition, XGBoost and GBDT showed similar results to Support Vector Machines (SVM) and Extreme Learning Model (ELM) in which they exhibited reduced computational losses that made them an eligible alternative technique for predicting ET. The results showed that global solar radiation was more significant than the other variables in the tropical and subtropical zones of China. Yet more research would entail model evaluation in areas other than China, and would also be based on various geological timescales (hourly or monthly).

When forecasting real ET, (Kaneko et al., 2019) introduced the idea of transfer learning from several regions, in which they researched maize fields in various African countries by referring to remote sensing data accessible to the public. A deep learning model, based on long short-term memory (LSTM), was used to predict crop fields with a Gaussian Process Layer. The data was divided into various segments: a random split and a sequential split. Both models achieved high levels of accuracy when utilizing the random split, unlike a chronological one based on data quality. In addition, the combined model trained on all countries showed comparable results with in-country models scoring an R2 of 0.63. For countries with sparse data, this supports the concept of learning from out-of-country features. Different algorithms such as a DNN model may be used in further studies.

Since no machine learning models were used (Baier and Robertson, 1965), but rather the association between many meteorological variables and latent evaporation was studied, we were encouraged to use Machine Learning to predict real ET. The work carried out by (Granata, 2019) has inspired us to experiment with multiple deep learning models coupled with probabilistic and conformal quantile models. We were also inspired to utilize data of different climatic conditions, hence different air temperature, relative humidity, incoming shortwave radiation, etc. . . Interestingly, (Granata, 2019) experimented with only one location with a humid subtropical climate, ranging from June to September (a warm and wet season) and from October to May (a moderate dry season). We assume that this limitation would not help in model generalization for unseen climatic conditions, and therefore it is important to consider multiple sites and climatic regions. We were inspired by the studies of (Huang et al., 2019) to predict real ET from data throughout a multitude of regions and not to be restricted to particular areas. Hence, we conducted research on the prediction of daily ET for 26 stations accumulated from Ameriflux and Euroflux sites in various climate and vegetation regions throughout the years 2000 and 2019. Our model is trained on various input variables such as wind speed, relative humidity, air temperature, land surface temperature, Albedo, vegetation index of normalized difference (NDVI), site id, month, and vegetation. By including all stations, only Ameriflux stations, and

only Euroflux stations, we vary the aforementioned input features and research the influence of each. A set of point-wise, probabilistic, and conformal quantile models were used. The data is split into training (60%), validation (20%), and testing (20%) using a custom split by stations to guarantee the presence of each station in all data splits. Several error metrics, correlation metrics, and utility-based metrics have been used to measure the efficiency of our models including Coefficient of Determination (R2), Root Mean Squared Error (RMSE), Mean Absolute Error (MSE), Recall, Precision, and Accuracy. Our research focuses on multiple regions as opposed to (Fan and Yue, 2019). We are also using more complex regression models with fewer and different meteorological variables as a result of filter-based feature selection techniques. Our findings show an 81% coverage rate and a 0.75 mm prediction interval length when using Conformalized Quantile regression models. We also note an R2 of 0.64, an accuracy of 68%, and an F2-measure of 0.9 when using the probabilistic NGBoost model. We were indeed influenced by the work (Kaneko et al., 2019) that led us to apply various methods of clustering. Our data clustering enables learning to be transferred from sites with particular climates and seasons to sites with different ones. We were also very encouraged to compare the performance of our machine learning model to that of a EEflux(METRIC) model. The EEflux(METRIC) model compared horribly to real ET, with an R2 of -0.5, an accuracy of 48%, and a negligible recall of 0.88. We succeeded in minimizing the bias between the EEflux(METRIC) model and the predicted ET in which NGBoost scored an R2 of 0.74, and accuracy of 70%.

Chapter 3

Systems and Hardware

3.1 Libraries and Frameworks

All the code for our modules is written in python and R. We have used the following libraries and frameworks:

- [TensorFlow](#): TensorFlow is an end-to-end open-source machine learning framework. It has a robust, scalable ecosystem of tools, libraries, and community resources.
- [Scikit-learn](#): Scikit-learn is an ML library containing various regression and classification shallow models.
- [UBL](#): Utility-based Learning package
- [pyTorch](#): PyTorch is a machine learning library containing various deep and shallow learning models.
- [rpy2](#): rpy2 is a python interface to the R language.
- [Numpy](#): Numpy is a python library that allows better control over big-sized arrays.
- [Pandas](#): Pandas is a python library that allows better control over datasets.
- [Matplotlib](#): Matplotlib is a python plotting library.
- [Seaborn](#): Seaborn is a python library for statistic data visualization.

3.2 Environments

We ran all our code modules on three different environments: Google Cloud Platform, HPC Octopus Cluster, and Docker.

3.2.1 Google Cloud Platform

We utilized the Google cloud platform since it offers great services like Auto-ML and Auto-sklearn. It is also a very flexible and useful computing service that enabled us to train machine learning models at a fast pace. It also offers the option of specifying the OS type, and the number of master and slave nodes. Furthermore, the cores needed, the RAM specifications, the memory for the GPU, and the GPU model type can also be tweaked. The Google cloud platform also offers interesting tools for machine learning model interpretation like What If and Facets. The node we used has 8GB RAM along with a master and two slave nodes.

3.2.2 HPC Octopus Cluster

AUB's Octopus high-performance computing cluster helped us plenty since it offers a strong computing power which enabled us to run our scripts in a shorter amount of time. The HPC Octopus cluster also has a GPU, which we utilized when using any TensorFlow model to shorten model training time. A node on the Octopus cluster mimics a CentOS machine. Before running our scripts, we can specify the cores needed, the RAM specifications, the memory for the GPU, and the GPU model type. We then submit our job script and it will be added to the queue of executable jobs. A matching node with the required specifications will handle the submitted job. The machine we used is a CentOS with 16 GB RAM.

3.2.3 Docker

Docker is a Google product that is referred to as the Next Big Thing in Cloud Computing. Docker offers the ability to run an application in a completely isolated environment which is referred to as a container. Containers are lightweight since they don't need the load of a hypervisor, thus they run directly within the host machine's kernel. Hence all the needed libraries are provided in the Docker container we created. Docker invokes as many cores as the PC it is running on. In our case, it is 6 cores. Now, any person could just install the docker container that we developed and run our code with no prior library and setup configuration needed. The machine we used is a Windows 10 which has 16 GB RAM and an intel core i-7 processor. Please check the repository on [Github](#).

3.3 Code and Data

All the code and data used for this thesis is found on [Gitlab](#).

Chapter 4

Study Area and Surveyed Sites

The data was accumulated from Ameriflux and Euroflux (Flux Towers) around various climates and vegetation between the years 2000 and 2018. The corresponding Landsat data was gathered from the Earth Engine Evapotranspiration Flux (EEflux) website. The data is obtained from 26 different sites present on Ameriflux and Euroflux in North America, South America, and Europe as shown in **Figure 4.1**. Our goal was to collect data from sites with various types of climates and terrestrial vegetation portrayed in **Table 4.1** and **Table 4.2** below.

Climate	Type
Cfa	Humid Subtropical - a mild climate with no dry seasons and a hot summer
Dsa	Dry Continental - a climate with a hot summer
Csa	Mediterranean - a mild climate with a dry and hot summer
Csb	Mediterranean - a mild climate with a dry and warm summer
Cwa	Humid Subtropical - a climate with a dry winter and a hot summer

Table 4.1: Climate Types

Vegetation	Type
Grasslands (GRA)	Lands with herbaceous covers
Crop Lands (CRO)	Land covered by temporary crops followed by harvesting, then bare soil
Closed Shrublands	Lands with woody vegetation and less than 2 meters high with shrub canopy cover.
Evergreen Broadleaf Forests (EBF)	Woody vegetation dominated land
Evergreen Needle Leaf Forests (ENF)	Woody vegetation dominated land

Table 4.2: Vegetation Types

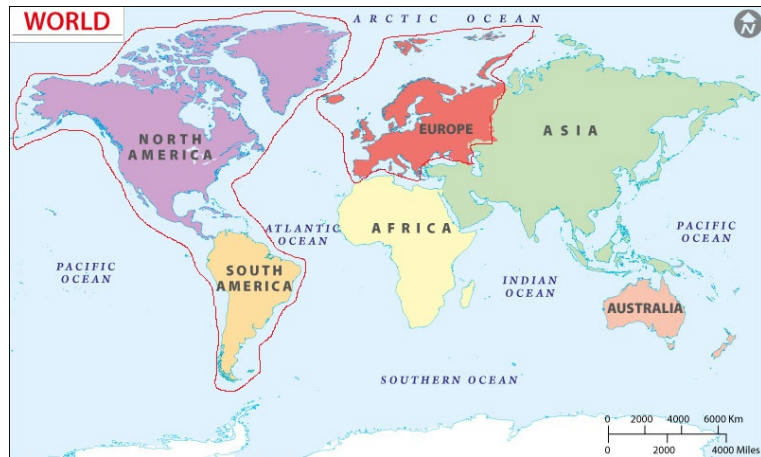


Figure 4.1: Area of Study

4.1 AmeriFlux

AmeriFlux is a network of PI-managed sites monitoring CO₂ habitats, water and energy flow in North, Central, and South America. AmeriFlux is now one of the most well-known and highly valued brands in climate and environmental science at the DOE Office of Biological and Environmental Research (BER). The AmeriFlux datasets and the understanding extracted from them provide critical ties between terrestrial ecosystem processes and climate-relevant responses at a landscape, regional and continental levels. AmeriFlux, which offers evapotranspiration fluxes for more than 50 sites, tracks the time frame, geographic location, and other variables for all its sites ([AmeriFlux](#)). As per the AmeriFlux data format, data is accessible at half-hourly intervals. A sample is shown in **Figure 4.2**.

4.1.1 Sites

The sites pertaining to AmeriFlux are 'US-A32', 'US-Wlr', 'US-Snd', 'US-Kon', 'US-Goo', 'US-SP2', 'US-AR1', 'US-Twt', 'US-Ced', 'US-A74', 'US-AR2', 'US-Shd', 'US-SO2', 'US-Skr', 'US-Tw2', and 'US-Bi2'.

4.2 EuroFlux

The European Fluxes Database Cluster ([EuroFlux](#)) is an initiative aimed at improving standardization, convergence, and cooperation between the European research projects' databases. It was developed with the goal of hosting measurements between ecosystems and the environment in a single infrastructure and providing standard and high-quality resources for data processing and data

sharing. The Euroflux Database Cluster maintains a variety of flux measurements, including meteorological variables, ancillary data, and meta-information from various locations in and around Europe.

4.2.1 Sites

The sites pertaining to EuroFlux are 'FI-Jok', 'DE-Kli', 'DE-Seh', 'DK-Ris', 'FR-Gri', 'IT-CA2', 'DE-RuS', and 'DK-Fou'.

4.3 EEFlux

[EEflux](#), Earth Engine Evapotranspiration Flux, is based on the METRIC model ([Allen, 2015](#)) in the Google Earth Engine System. EEflux makes on-demand ET forecasts for Landsat scenes (Landsat 5,7 or 8) in addition to processing Landsat images in almost every land area of the globe at a resolution of 30 m. From EEflux, we were able to retrieve actual ET, Albedo, normalized difference vegetation index (NDVI), and land surface temperature (LST) data for the 26 sites obtained above across the years 2000-2018.

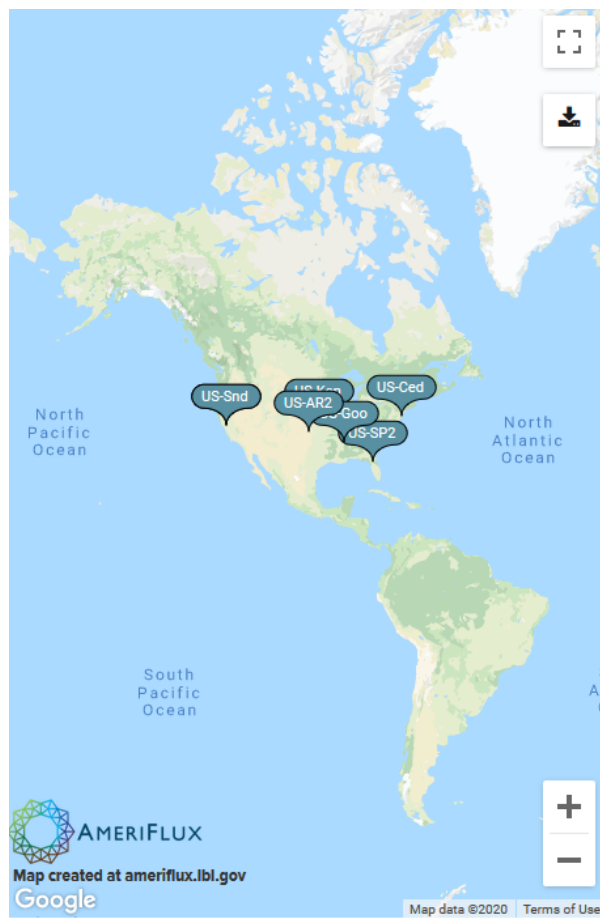


Figure 4.2: Sample Sites Ameriflux

Chapter 5

Quality Control

5.1 Data Cleaning

A set of data cleaning techniques were applied to the collected data from Ameriflux, Euroflux, and EEflux.

5.1.1 AmeriFlux and EuroFlux

First, We converted all “-9999” values to NaNs. We then removed all data instances with missing Real ET values and instances with malformed dates. To exclude outliers, we narrowed the ranges of real ET between 1 and 15 mm.

5.1.2 EEFlux

We imputed missing EEflux ET instances and replaced them with the mean of EEflux ET. We have also carried out a series of experiments with a number of cut-offs ranging from 0.1,0.2,0.3,0.4,0.5 to 1. We reported the percentage of EEflux ET values below the specific cut-off. We found that less than 5% of the EEflux data is less than 0.3, so we chose it as a lower bound cutoff. The reason we did this study was to make sure there was no data tampering.

Duplicate rows that have the same values for all features but differ in EEflux ET and cloud cover percentage are omitted. Only one row is corrected and retained. We corrected EEflux ET values as follows:

- For rows with the same data (Site Id, Date, and Weather parameters) with variations in EEflux ET values less than 0.3, the EEflux ET average of these rows will be computed and the EEflux ET value will be replaced.
- For rows with the same data (Site Id, Date, and Weather parameters) with variations in EEflux ET values greater than 0.3, the difference between the

EEflux ET and the actual ET rows will be calculated, and the one with the least variance is the row we hold.

5.2 Data Correction

Verifying the accuracy of hydro-meteorological measurements along with the turbulent fluxes has always been a vital issue. This verification is important to quantify any systematic errors or uncertainties which play a major role in irrigation management. The turbulent heat fluxes are somehow smaller than the available energy. Thus, ET values should be corrected. We have applied two different correction methods, one where we implemented Bowen’s Ratio (Mauder et al., 2017) and one where we used the library fluxQAQC which applied Energy Balanced Ratio (EBR) and a variant of Bowen’s Ratio.

5.2.1 Bowen’s Ratio

The correction method we utilized is Bowens Ratio(Mauder et al., 2017). The Bowen ratio (H/LE) is utilized to estimate heat loss or gain in a substance. Hence, it is the ratio of energy fluxes moving from one state to another: sensible heat by latent heating.

$$C = \frac{\sum_{i=1}^k H_i + LE_i}{\sum_{i=1}^k R_{n,i} + G_i} \quad (5.1)$$

$$H_c = \frac{H_m}{C} \quad (5.2)$$

$$LE_c = \frac{LE_m}{C} \quad (5.3)$$

We refer to this method as the manual Bowen’s Ratio method.

5.2.2 Bowen’s Ratio - FluxQAQC

The FluxQAQC library computes the Bowen’s ratio in a different manner than the Bowen’s Ratio in (Mauder et al., 2017). The Bowen Ratio Energy Balance closing approach is conducted on a regular basis by maintaining the Bowens Ratio (H/LE). This adjustment closes the energy balance exactly when the balance is measured on a daily time scale (i.e. daily amounts), but not when assessed on flux data points (e.g., 30-minutes).

1. Compute the Bowens Ratio β

$$\beta = \frac{H}{LE} \quad (5.4)$$

2. Compute the corrected LE

$$LE_{corr} = \frac{(Rn - G)}{(1 + \beta)} \quad (5.5)$$

3. Compute the corrected H

$$H_{corr} = LE_{corr} \times \beta \quad (5.6)$$

We refer to this method as the library Bowen's Ratio method.

5.2.3 Energy Balanced Ratio (EBR)

This method entails removing and gap-filling the extreme values of the daily time series data. Then the inverse of this data is used for the initial latent energy LE and sensible heat H time series flux data as a series of correction factors.

- Filter out poor quality data if quality control flags are provided with the data set, for example a QC field for a gap filled LE .
- Calculate the energy balance ratio daily from the raw data.

$$EBR = \frac{H + LE}{Rn - G} \quad (5.7)$$

- Filter out EBR values that are outside 1.5 times the inter-quartile range
- Use the filtered EBR time-series data to correct LE and H

$$EBC_{CF} = \frac{1}{EBR} \quad (5.8)$$

$$LE_{corr} = LE \times EBC_{CF} \quad (5.9)$$

$$H_{corr} = H \times EBC_{CF} \quad (5.10)$$

We refer to this method as the library EBR method.

5.3 Data Generation

After applying necessary data cleaning techniques to AmeriFlux and EuroFlux tower data, we corrected the ET in the three different methods: manual Bowen's Ratio, library Bowen's Ratio, and library EBR. We then resampled our AmeriFlux and EuroFlux data from half-hourly to daily, followed by mapping each instance to the corresponding Landsat EEflux data.

5.3.1 Datasets

After collecting all the needed metrics and applying the correction, we end up with the following datasets:

- Manual Bowen Daily Data set: A daily data set acquired by implementing the manual Bowen correction method on the real ET.
- Library Bowen Daily Data set: A daily data set acquired by implementing the library Bowen correction method on the real ET.
- Library EBR Daily Data set: A daily data set acquired by implementing the library EBR correction method on the real ET.
- Manual Bowen Joint Data set: A weekly data set merged with the EEflux data set acquired by implementing the manual Bowen correction method on our daily data set and merging it with EEflux fields.
- Library Bowen Joint Data set: A weekly data set merged with EEflux data set acquired by implementing the library Bowen correction method on our daily data set and merging it with EEflux fields.
- Library EBR Joint Data set: A weekly data set merged with EEflux data set acquired by implementing the library EBR correction method on our daily data set and merging it with EEflux fields.

5.3.2 Data Legend

Each data set consists of all or some of the below fields:

- *Date*: The daily date at which the data was recorded at with the format “MM/dd/YY
- *SiteId*: A unique identifier for each site
- *Year*: The year at which the data was recorded
- *Month*: The month at which the data was recorded
- *Day*: The day at which the data was recorded
- *Elevation*: The elevation in meters
- *Latitude*: The latitude
- *Longitude*: The longitude
- *Climate*: If available one of 5 values (Cfa, Csa, Csb, Dsa, Cwa) or Other for sites coming from Euroflux

- *Vegetation(IGBP)*: Values are one of which (CRO, GRA, ENF, CSH, EBF, DBF, WET, WSA)
- *WS*: Wind speed
- *RH*: Relative humidity
- *TA*: Air temperature
- *NETRAD*: Net radiation
- *SW_IN*: Incoming Shortwave Solar Radiation
- *G*: Ground/Soil flux
- *H*: Sensible heat flux
- *H_ebr_corr*: The corrected sensible heat flux using EBR method
- *H_bowen_corr*: The corrected sensible heat flux using Bowen method
- *LE*: Latent energy flux
- *LE_ebr_corr*: The corrected Latent energy flux using EBR method
- *RealET*) : *ThecalibratedLE_ebr_corr*
- *LE_bowen_corr*: The corrected Latent energy flux using Bowen method
- *LE_bowen_corr(mm)*: The calibrated *LE_bowen_corr*
- *ET_ebr*: The evapotranspiration calculated from LE and TA using EBR method
- *ET_ebr_corr*: The corrected evapotranspiration calculated from LE and TA using EBR method
- *ET_ebr_corr(mm)*: The calibrated *ET_ebr_corr*
- *ET_bowen*: The evapotranspiration calculated from LE and TA using Bowen method
- *ET_bowen_corr*: The corrected evapotranspiration calculated from LE and TA using Bowen method
- *ET_bowen_corr(mm)*: The calibrated *ET_bowen_corr*
- *gridMET_ETr*: The gridMET alfalfa reference *ET* (nearest cell)
- *ET_r*: The alfalfa reference evapotranspiration

- ET_{rF_bowen} : The fraction reference evapotranspiration for ET_bowen_corr , i.e. ET_bowen_corr / ET_r
- ET_{rF_ebr} : The fraction reference evapotranspiration for ET_ebr_corr , i.e. ET_ebr_corr / ET_r
- ET_o : The grass reference evapotranspiration
- ET_{oF_bowen} : The fraction reference evapotranspiration for ET_bowen_corr , i.e. ET_bowen_corr / ET_o
- ET_{oF_ebr} : The fraction reference evapotranspiration for ET_ebr_corr , i.e. ET_ebr_corr / ET_o
- $EEfluxET$: ET from EEflux which initially has the value of ET, if ET is empty it is populated from the mean of the 8 neighboring pixels
- $EEfluxET_o$: The grass reference from the Climate Engine
- $EEfluxET_r$: The grass reference from the Climate Engine
- $EEfluxLST$: Land Surface temperature from EEflux
- $EEfluxNDVI$: NDVI from EEflux
- $EEfluxAlbedo$: Albedo from EEflux
- $column_number$: The site id, month, and vegetation columns are encoded using binary encoding with a format of the column name followed by the number of the encoding column, i.e. $SiteId.1$
- $column - number$: This is a lag of the column name i.e. $RH - 1$ is the first lag of RH, $RH - 2$ is the second lag of RH

5.3.3 Daily-Joint Mapping

We have two types of datasets: Joint and Daily. Instances of the Daily dataset are collected on a daily basis, meanwhile, instances of the Joint dataset are collected on a weekly basis. The Joint dataset contains EEflux ET, a measurement of ET collected on a weekly basis as well. The Daily dataset does not contain EEflux ET, it only contains real ET (collected on a daily basis). We seek to create a mapping between the Joint and Daily datasets on the following criteria:

- Site Id
- Date
- Real ET value

Once we do that, we would have a subset of the Daily dataset but with extra EEFlux ET values mapped from the Joint dataset.

5.3.4 Choice of Dataset

In order to decide which of the datasets we should proceed with (i.e train our machine learning models on), we did preliminary experiments with all of the available datasets and assessed the performance of each accordingly.

- Models trained on Library Bowen daily dataset gave better results than models trained on Manual Bowen daily dataset shown by a 6.3% increase in accuracy.
- Models trained on Library Bowen daily dataset gave better results than models trained on Library EBR daily dataset shown by a 8.9% increase in accuracy.

Thus, our chosen dataset is the Library Bowen Daily data set.

5.3.5 Final Dataset

The final data set is divided into input features and an output variable as shown below. The Library data set consists of 5,123 rows.

- Input:
 - Wind speed (WS)
 - Air temperature (TA)
 - Relative humidity (RH)
 - Normalized Difference Vegetation Index (EEflux NDVI)
 - EEflux Albedo
 - Land Surface Temperature (EEflux LST)
 - Vegetation (Categorical Variable Binary Encoded)
 - Site Id (Categorical Variable Binary Encoded)
 - Month (Categorical Variable Binary Encoded)
 - EEflux ET: Modeled ET coming from EEflux
- Output:
 - Real ET: The calibrated and corrected real ET

5.4 Feature Engineering

The feature engineering process comprises exploratory data analysis coupled with data transformation techniques, data encoding techniques, and a seasonality study.

5.4.1 Time-Series Analysis

Our problem is a time series problem because it shows the date component. To ensure our data is stationary and can be modeled further, we will perform time-series analysis.

Stationary Study

A time series is stationary if it has no trend or seasonal effect. Summary statistics must be compatible between observations to be predictable. There are many approaches for checking whether or not a time-series is stationary, including some that are as follows:

1. Summary Statistics: For example mean and variance, which should be consistent over time
2. Statistical Tests: For example, the Augmented Dicky-Fuller test confirms if the data is stationary according to several hypothesis tests.

The Augmented Dickey-Fuller (ADF) test is a statistical test. The main idea behind this unit root test would be that it dictates how heavily a time series is defined by a certain trend. It is utilized to recognize whether or not a time series is stationary (Mushtaq, 2011). This test yields the following hypotheses:

1. H0 or Null Hypothesis: If it is not rejected, it means that the time series has a root unit, so it is not stationary. It has a time-dependent structure.
2. H1 or Alternate Hypothesis: The null hypothesis is rejected, which means that the time series does not have a root unit, so it is stationary. It doesn't have a time-dependent structure.

We have tried the ADF test on our data and got the results portrayed in **Figure 5.2**.

Figure 5.2 shows the value of the test statistics being -10.24, which means that it is lower than all the critical values and the p-value ≤ 0.05 , that also means that we reject the null hypothesis (H0) and therefore the data is stationary.

```

ADF Statistic: -10.244663
p-value: 0.000000
Lags used: 7
Critical Values:
1%: -3.961
5%: -3.411
10%: -3.128
-----

```

Figure 5.1: ADF Test Results

White Noise Study

White noise is believed to be a the crucial notion in time series forecasting. A time series is a white noise if it's dependent features are spread equally and independently with a zero mean. If a series is white noise, it imitates a sequence of random numbers and therefore cannot be predicted.

`ljung_box_test` is used to test whether or not there is white noise. This test yields the following hypotheses:

1. H0: The data is distributed independently, which means that a zero correlation between dependent variables is observed.
2. H1: The data is not independently distributed, which means that dependent variables exhibit a non-zero correlation.

There are two conditions upon which we accept or reject the null hypothesis:

1. The p-value is ≤ 0.05 : We reject the null hypothesis
2. The p-value is < 0.05 : We reject the null hypothesis

We apply the white noise test on our data in **Figure** [7.2]. It is noted that all p-values are less than 0.05, which implies that the time series doesn't exhibit white noise.

```

all p-values for ljung box <= 0.05
all p-values for box pierce <= 0.05

```

Figure 5.2: White Noise Test Results

Random Walk Study

A time series is identified as a Random Walk if one of the following is met:

1. A heavy temporal dependency that decays linearly or in a similar pattern is seen in the time series.

2. The time series is non-stationary and it reveals no learnable structure in the data to make it stationary.

To research the relation between lags, we generate auto-correlation plots for each input and output function in **Section 5.4.3**. The auto-correlation plots for the weather variables show a substantial association between lagged observations, implying that the time series does not show a random walk.

5.4.2 Exploratory Data Analysis

Exploratory data analysis refers to the vital method of conducting initial data study to find trends, spot irregularities, and test theories. In this section, we portray summary statistics, variations in sites, climates, and seasons, in addition to the distribution of the input and the output variable. We have utilized the tools [pandas-profiling](#), [facets](#), [fitter](#), and [matplotlib](#).

Statistics

In this section, we have used [pandas-profiling](#) and [facets](#) to extract the summary statistics. We are using the Library dataset, which has the following characteristics shown in **Table 5.1**.

Number of observations	5128
Total Missing (%)	0.0%
Total size in memory	4.8 MiB
Average record size in memory	984.0 B

Table 5.1: Number of Rows in Library Dataset

Our dataset is made up of 5,128 rows.

Site Variation

In this section, we have used matplotlib to produce the visualizations.

In our dataset, each row pertains to a certain site with a specific site Id. We aim to visualize how many sites in our data belong to either Euroflux or Ameriflux.

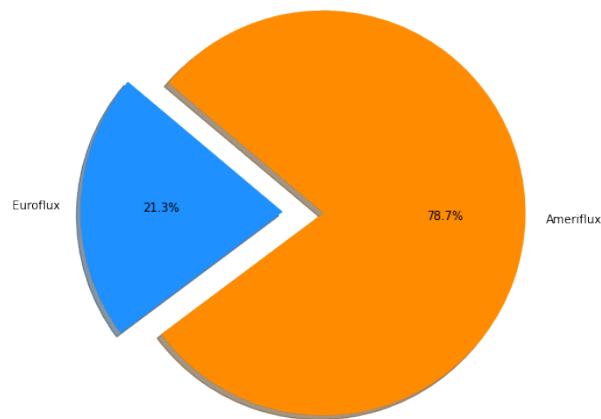


Figure 5.3: Site Distribution

As portrayed in **Figure 5.3**, it is noted that 21.3% of the sites are Euroflux sites, and 78.7% of the sites are Ameriflux sites.

We now aim to see how many rows per site exist in our data.

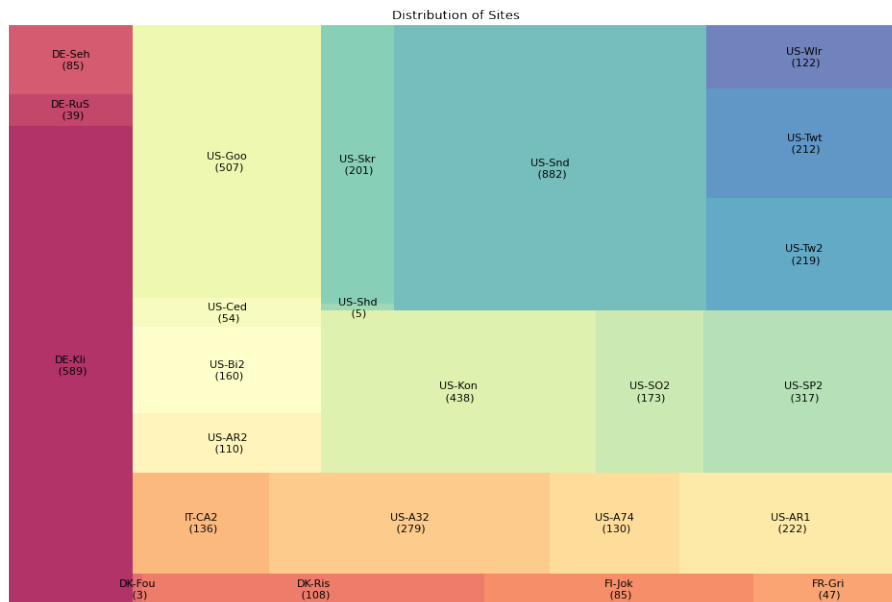


Figure 5.4: Number of Rows per Site

According to **Figure 5.4**, it is noted that the sites with the highest frequency in our data are:

1. US-Snd, which pertains to Ameriflux, has 882 rows.
2. DE-Kli, which pertains to Euroflux, has 589 rows.
3. US-Goo, which pertains to Ameriflux, has 507 rows.
4. US-Kon, which pertains to Ameriflux, has 438 rows.
5. US-SP2, which pertains to Ameriflux, has 317 rows.

Climate Variation

In this section, we have used matplotlib to produce the visualizations.

In our dataset, each instance pertains to a specific climate, as shown in **Table 4.1**. We aim to visualize the frequency of each climate and its ET distribution.

According to **Figure 5.5a**, which represents the bar plot of each climate and its frequency in the dataset, we note that the top three climates with the highest frequencies are Cfa (1,852 rows), Csa (1,646 rows), and other (1,092 rows).

According to **Figure 5.5b**, which is a density plot for ET (mm) for each of

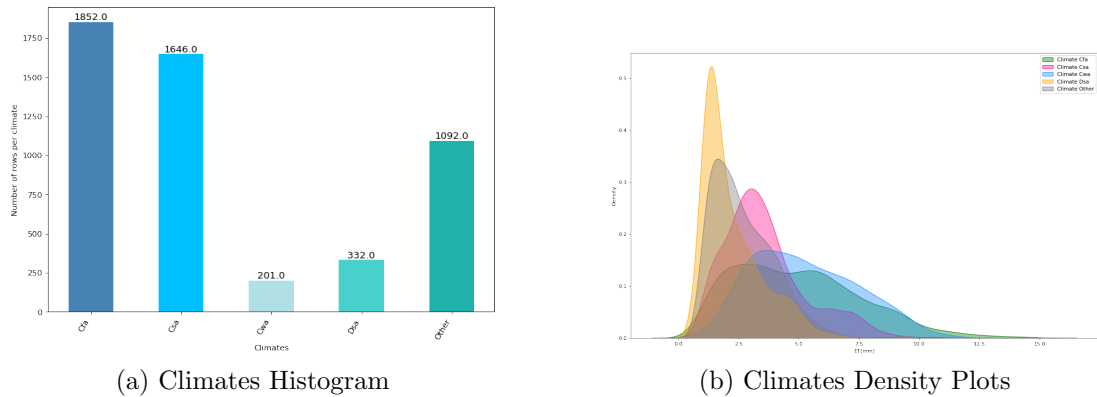


Figure 5.5: Distribution of Climates

our climates, we note that climates Cfa and Cwa have a similar ET distribution ranging from 1 to 15mm including all rare but high values of ET. We also note that climates Csa and Other have a very close ET distribution, with values ranging from 1 to 10mm. Climate Dsa is however unique, and has an ET distribution between 1 and 7.5mm, not including most high and rare values of ET.

We note from **Figure 5.4** that the top sites with the highest frequency are:

1. US-Snd which has a Csa climate (Mediterranean: mild with dry, hot summer).
2. DE-Kli which has the climate Other.
3. US-Goo, US-Kon, and US-SP2 which have a Cfa climate (Humid subtropical: mild with no dry season, hot summer).

Thus, we conclude that the top sites are represented by the top two climates (Cfa and Csa).

Season Variation

In this section, we have used matplotlib to produce the visualizations.

In our dataset, each instance pertains to a specific season, as shown in **Section 5.4.5**. We aim to visualize the ET distribution per season.

According to **Figure 5.6**, which is a density plot of ET (mm) per season, it is prevalent that the summer season is unique because it covers all ET values from 1 to 15mm (including rare and high ET values). The spring season also offers good coverage of ET, where the range stretches between 1 and 10mm. The winter season, however, has the lowest ET values, ranging from 1 to 7.5 mm, not including rare and high ET values.

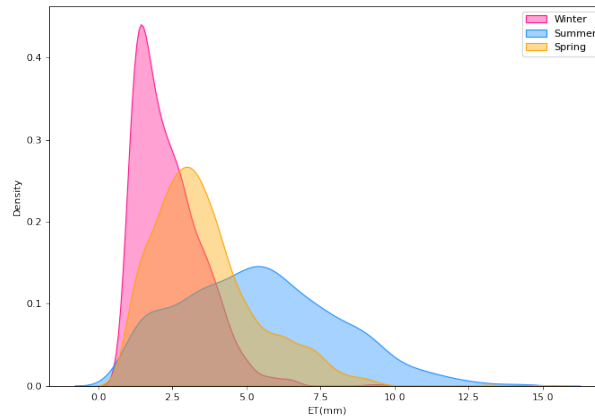


Figure 5.6: Seasons Density Plot

Distribution of the Input Features

We have used the fitter tool to produce all the plots in this section. We now aim to study the distribution of each input feature on US-Goo, one of the most representative sites in our dataset. The fitter package aids in identifying the distribution of each feature by comparing its histogram to probability density functions of famous distributions like the normal distribution, exponential, Cauchy, etc... This package returns a set of distributions ranked from best to worst fit according to the distribution having the least sum of squares error.

According to **Figure 5.7**:

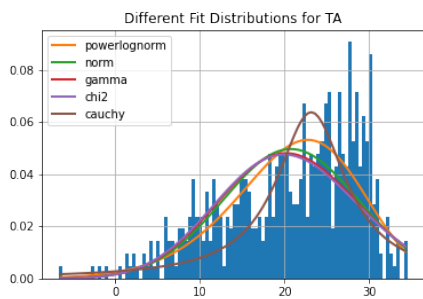
- TA has a distribution of powerlognorm
- RH has a distribution of norm
- WS has a distribution of lognorm

According to **Figure 5.8**:

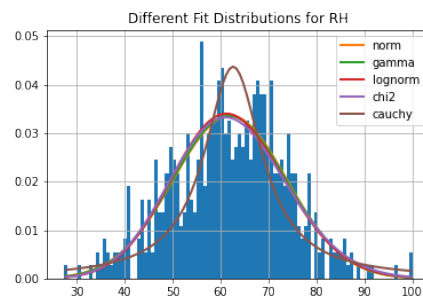
- EEflux Albedo has a distribution of lognorm
- RH has a distribution of exponpow (exponential power)
- WS has a distribution of cauchy

Distribution of the Output Feature

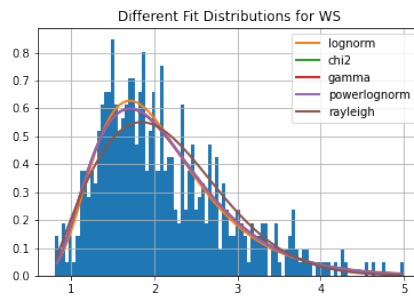
In this section we have also used the fitter tool to produce this plot. According to **Figure 5.9**, we note the distribution of our output variable ET is exponential power, which is also referred to as the generalized normal distribution.



(a) TA

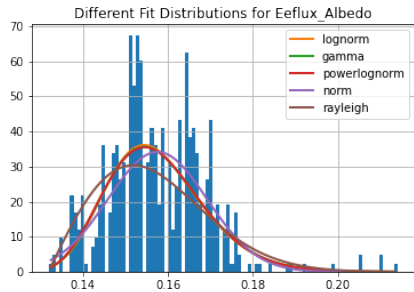


(b) RH

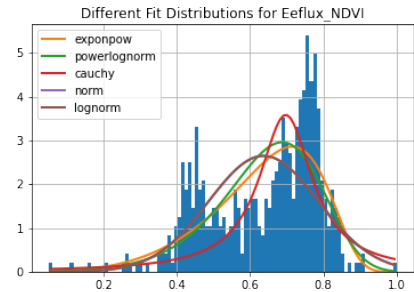


(c) WS

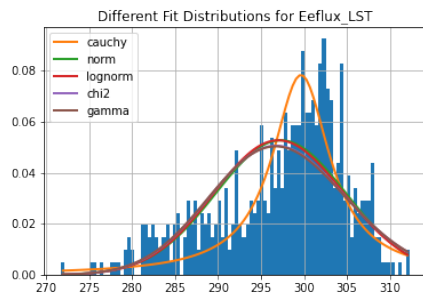
Figure 5.7: Distribution of TA, RH, and WS



(a) Eefflux Albedo



(b) Eefflux NDVI



(c) Eefflux LST

Figure 5.8: Distribution of Eefflux Albedo, Eefflux NDVI, and Eefflux LST

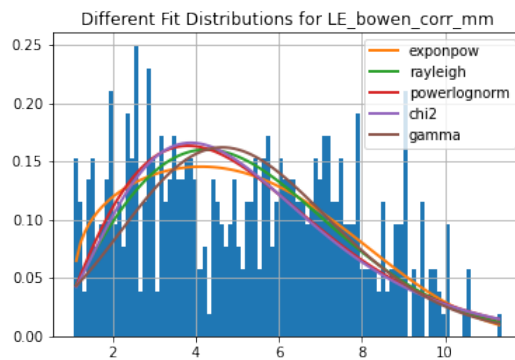


Figure 5.9: ET

Seasonal Trends of TA and RH alongside ET

We have used the tool matplotlib to generate the plots in this section.

We aim to study how values of TA and RH relate to values of ET as a function of seasons across one Ameriflux site US-Goo and one Euroflux site DE-Kli. In **Figure 5.10**, in the first row, the x -axis represents TA in degrees Celsius versus ET in mm, and the y -axis represents the number of data points. In the second row, the x -axis represents RH in (%) versus ET in mm and the y -axis represents the number of data points. The first column shows the data for the winter season and the second column shows the data for the summer season.

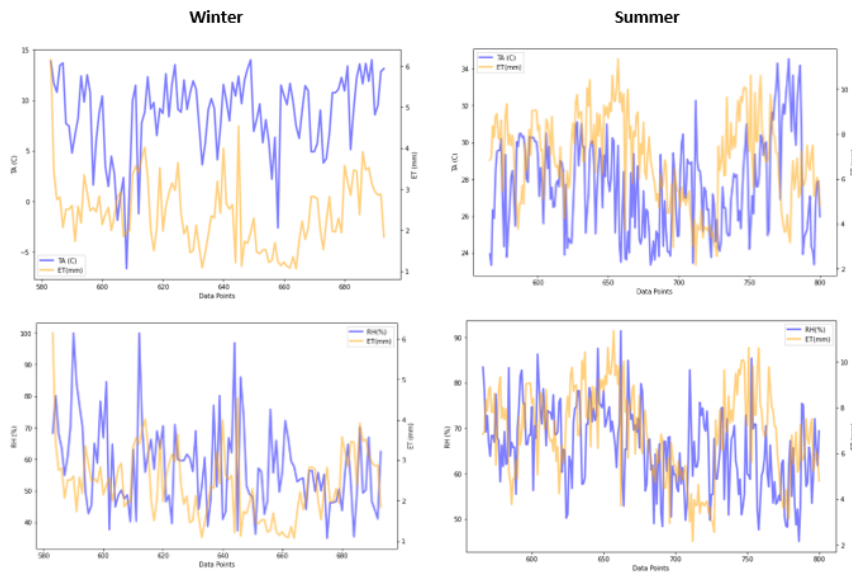


Figure 5.10: Ameriflux Site US-Goo

According to **Figure 5.10**, TA and ET are directly proportional in the summer season, in which high TA implies high ET, and in the winter season in which low TA implies low ET. This observation coordinates with the observations in Chapter 7 and Chapter 14. RH and ET are inversely proportional in the summer season, in which low RH implies high ET, and in the winter season in which high RH implies low ET. This observation also coordinates with the observations in Chapter 7 and Chapter 14. We now study one of the top representative sites from Euroflux, DE-Kli. In **Figure 5.11**, the same observations are made as in **Figure 5.10**.

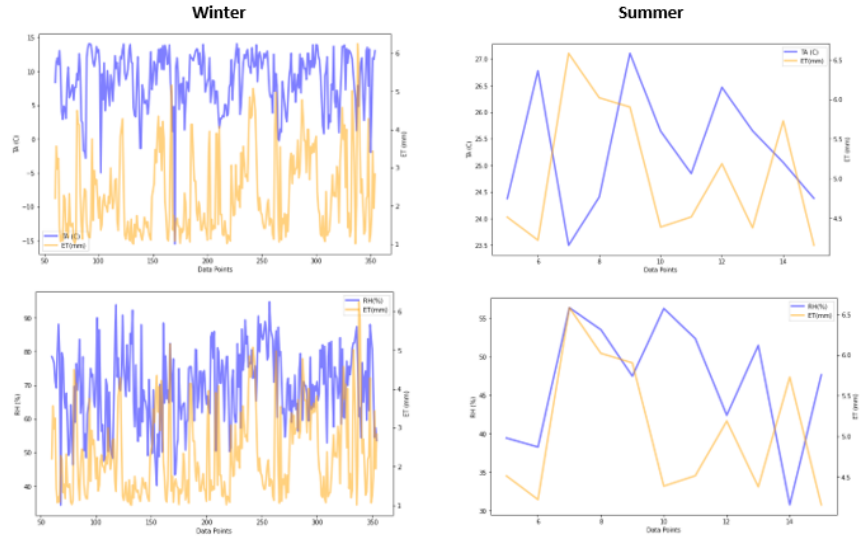


Figure 5.11: Euroflux Site DE-Kli

5.4.3 Data Transformations

A variety of data transformation techniques are applied to our data set. Since our data set contains a date column that reflects a time structure, we have replaced it by creating lags for several input columns. We conducted a series of auto-correlation plots for each input variable to produce an adequate number of lags.

Auto-correlation

Correlation is a technique used to measure the intensity and form of a relationship between an observation and its lag(s). This is often called auto-correlation (Huitema and Laraway, 2006). The value of the correlation varies between -1 and 1. A value close to zero shows a weak correlation, while a value similar to -1 or 1 shows a strong negative or a strong positive correlation. Using autocorrelation graphs, we visually reflect this association in order to better understand the correlation between an observation and its lag. The stronger the correlation between the output variable and the particular lagged variable, the greater the weight that the regression model will place on that variable while modeling. The highest and most positive correlations with their lags are TA, EEflux LST, and RH.

Deciding Number of Lags

Time series analysis implies the association between a current measurement and its previous observation. Previous observations in time series are labeled as lags.

An observation in the previous time step is marked as lag number 1, an observation in the two-step is marked as lag number 2, etc... The scatter plot is a useful form of plot to examine the relationship between each observation and its lag. For this study, we utilized Pandas' lag plot. It plots the studied measurement at time t on the x-axis and the measurement lag observation (at time $t-1$) on the y-axis. If the data points pile from the bottom-left to the top-right of the plot along the bisector, a positive correlation relationship is reflected. If the data points pile from the top-left to the bottom-right along the bisector, a negative correlation is reflected. We plot the scatter plots for our studied features:

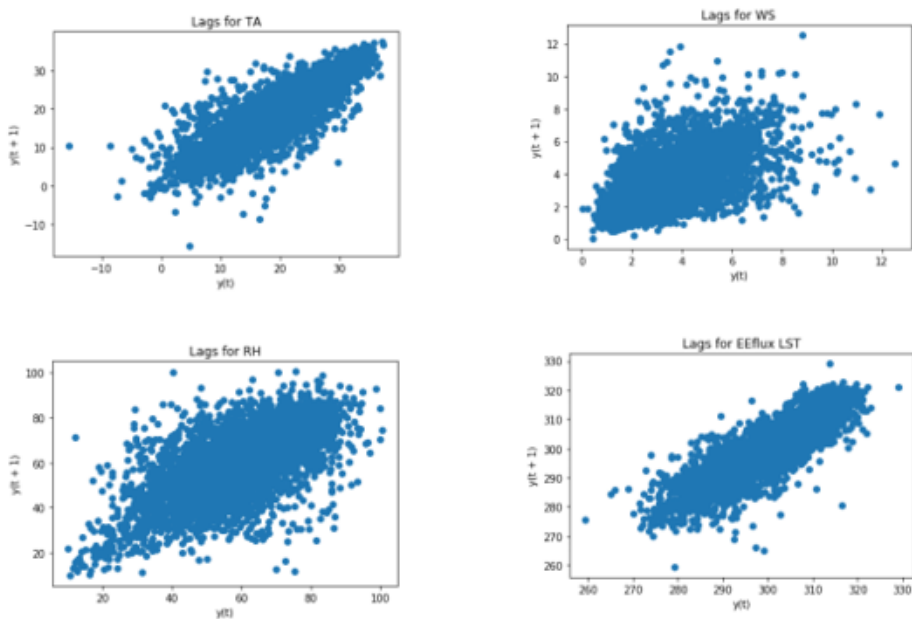


Figure 5.12: Auto-correlation Scatter Plot for WS, RH, TA, and LST

Figure 5.12 demonstrates a positive correlation between the element studied and its first lag as the scatter plot begins to form a diagonal shape from the bottom left to the top right, which is an indicator of a strong positive correlation.

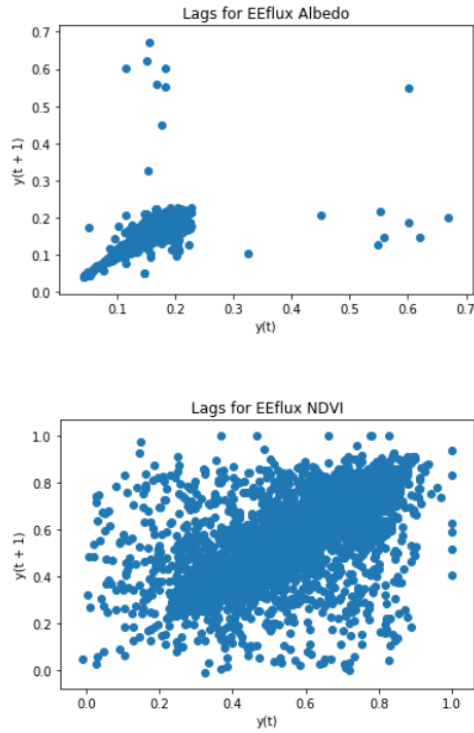


Figure 5.13: Auto-correlation Scatter Plot for NDVI and Albedo

We also note that NDVI and Albedo show no obvious correlation in **Figure 5.13**.

Auto-correlation Plots

Using autocorrelation plots, we also visually show this connection in order to further explain the association between an observation and its lag. The stronger the correlation between the output variable and a particular lagged variable, the greater the weight that the regression model will place upon this variable when modeling. TA, EEflux LST, and RH are the most significant characteristics that display a high and positive correlation with their lag.

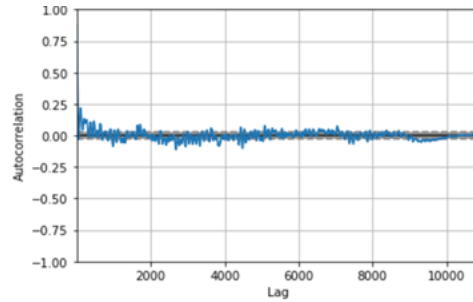


Figure 5.14: TA Auto-correlation Plot

In **Figure 5.14** , the x-axis shows the number of lags and the y-axis shows the auto-correlation value ranging from -1 to 1. This figure portrays that upon increasing the number of lags to a certain threshold, the correlation value will decrease. The highest correlation value obtained was 0.25, hence choosing five lags for TA.

Lag Generation

We have generated lags for the following columns in accordance to the auto-correlation plots:

- “WS”: 2 lags
- “RH”: 3 lags
- “TA”: 5 lags
- “EEflux Albedo”: 2 lags
- “EEflux NDVI”: 2 lags
- “EEFlux LST”: 5 lags

Data Encoding

We have converted our categorical columns, which are Site Id, Month, Vegetation, and Environment, to a numeric binary representation using a binary encoder. We used binary encoding instead of one-hot encoding since it yields in fewer dimensions. The number of encoded binary columns created for each categorical function will vary depending on the specific number of available categories per column. This binary encoder works in the following manner:

- An ordinal representation is given for each input feature
- The ordinal representation is changed to a binary representation

- The resulting set of binary digits is sectioned into individual columns with the value 0 or 1
- Each categorical column is renamed according to the binary encoding: “Site Id”, “Site Id_0”, “Site Id_1, ...

5.4.4 Unsupervised Clustering Analysis

A multitude of clustering algorithms is utilized to analyze various trends of model performance. Two key clustering methods are used: clustering by K means and Dendrograms or clustering by subsetting centered on a selection of a category pertaining to a specific input feature.

K Means

K Means is an unsupervised machine learning clustering algorithm that initially sets random cluster centers (centroids) and then maps each data point to its nearest cluster by calculating its distance from each centroid ([Bano and Khan, 2018](#)). The centroid will be updated and the process will be repeated until no further cluster update is carried out. We specify the number of clusters using the elbow method.

Dendrograms

Dendrograms is a clustering algorithm that works by assigning an individual cluster to each point, merging the nearest cluster pair, and then repeating the process until there is only one cluster left ([Bano and Khan, 2018](#)). This merge approach is based on the calculation of the similarity between the data points by measuring their distance to the centroids of those clusters. We perform several clustering experiments by varying the input features we fed to our algorithm.

- Clustering on one weather column only at a time (WS, RH, TA) and we apply that for $k = 2$ and $k = 3$
- Clustering on two columns (WS RH, WS TA, RH TA) and we apply that for $k = 2$ and $k = 3$
- Clustering on all three columns (WS, RH, TA) and we apply that for $k = 2$ and $k = 3$
- Clustering on different climates encoded columns for $k = 4$

The choice of k was based on running the elbow method on K Means, finding the best cut, and trying with different k values. The aim of clustering by Kmeans or Dendrograms is to identify whether certain WS, TA, and RH values would yield

better real ET predictions, rather than modeling on the whole dataset with all ranges of WS, TA, and RH values.

Clustering by Subsetting on Climates

We split our data into five different data sets according to the five types of climates available (Cfa, Csa, Cwa, Dsa, Csb). We then model on each climate to analyze model trends. The aim of this study is to analyze model performance across individual climates.

5.4.5 Seasonality Study

After performing several clustering techniques on several criteria, we were interested in studying data separability by the air temperature. In order to observe our model performance across the three seasons (summer, winter, and spring), we performed a seasonality study. We observed which clustering technique showed clear separability in air temperature allowing us to generalize the climate it belongs to. We have noticed that clustering by air temperature (TA) using K means $k=3$ showed a clear separability in TA, portraying clear seasonal representations for each cluster. Cluster 0 represented winter, cluster 1 represented summer, and cluster 2 represented spring. Hence, we performed modeling on these three clusters to observe model performance. We observe the air temperature distribution in the three clusters in **Figure 5.15** and in **Table 5.2**.

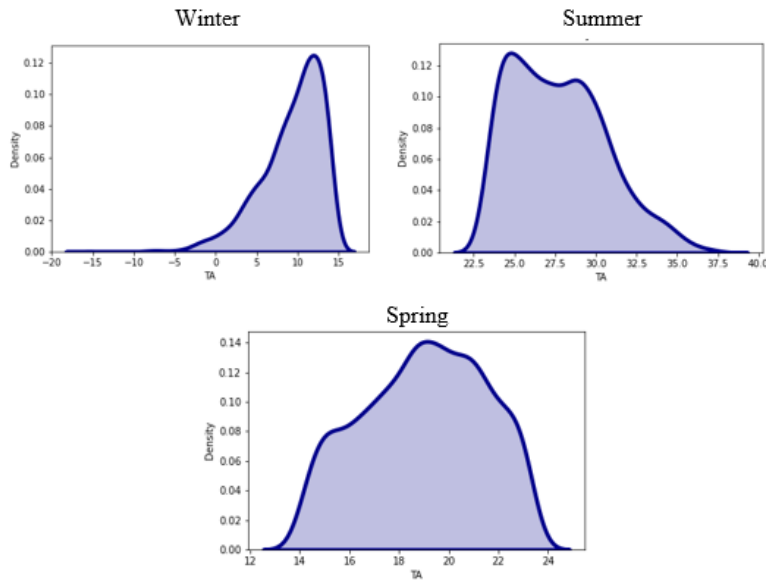


Figure 5.15: Distribution of TA across Clusters

	cluster 0: Winter	cluster 1: Summer	cluster 2: Spring
minimum TA	-15.4 C	23.33 C	14.07 C
maximum TA	14.07 C	37.31 C	23.31 C
dataset size	1130	1538	2455

Table 5.2: TA Clusters Distribution

Chapter 6

Utility-Based Regression and Minority Up-sampling

Our work is heavily inspired by (Ribeiro and Torgo, 2008). Cost-sensitive learning has become a vital approach when dealing with current machine learning problems. Prior existing research has been very focused on classification problems rather than regression problems in matters of cost-sensitive analysis. The majority of literature works tackling regression presume uniform costs and utilize average error statistics, which is very erroneous in real-world problems. However, (Ribeiro and Torgo, 2008) have developed the UBR framework which allows evaluating regression models in a cost-sensitive manner. With utility-based regression, we can transform our regression problem to a classification-like problem by classifying our output variables as rare vs not rare. (Ribeiro and Torgo, 2008) emphasize that models should be evaluated by their benefits and rewards rather than cost. The UBR module estimates the utility of any regression model by attaining a balance between cost and benefits from the obtained predictions.

6.1 Relevance Function

(Ribeiro and Torgo, 2008) assume that the benefits of the target variable values are not uniform across the target variable domain. Hence, this information which depends on the domain will be obtained by the relevance function. This relevance function will map each target variable value to a spectrum of 0 to 1, 0 being not relevant and 1 being very relevant. (Ribeiro and Torgo, 2008) also emphasize that relevance is associated with rarity, hence the most relevant values are the rares (which do not occur multiple times in the dataset). Thus, a function that is inversely proportional to the probability density function of the target variable values can be defined as a relevance function. The idea of the relevance of the prediction for a certain test instance is portrayed by defining a bi-variate relevance

function:

$$\Phi(\hat{y}, y) = (1 - m)\phi(\hat{y}) + m\phi(y) \quad (6.1)$$

where:

- \hat{y} is $y_{predicted}$
- y is y_{actual}
- m is a parameter ($0 \leq m \leq 1$) which differentiates between situations 1 (false alarms) and 2 (opportunity costs). m is set to be 0.5 to have a balance between the two scenarios.

The relevance function is simply the weighted average of the relevance for \hat{y} and y . The relevance is high when both y and \hat{y} are highly relevant.

6.2 Defining Rarity

In our problem, the target variable is Real ET which ranges between 1 and 15mm. In order to determine the range of rare vs not rare, we conducted the following experiment:

- We separated the data by site Id.
- For each site Id, we calculated the percentage of data lying between the following ranges: [1,15], [2,15], [3,15]...[14,15]. The distribution of the data is shown in figure 6.1. The box plot and density plot for Real ET are also shown in **Figures** 6.2 and 6.1 respectively , where it is clear that ET values above 10 are outliers.
- We checked the ranges in which the majority of sites have only less than 10% data falling into them. A sample of this study is shown in **Figure** 6.3.
- We experimented with each range on a base model in order to choose which gives the highest precision and recall. The best range with the best results was [5,15].
- We defined all values of ET falling in the range [5,15] are considered rare, with 15 being the rarest.

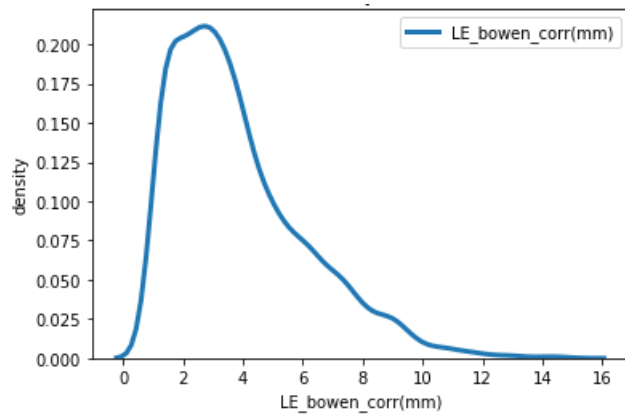


Figure 6.1: Real ET Value Distribution

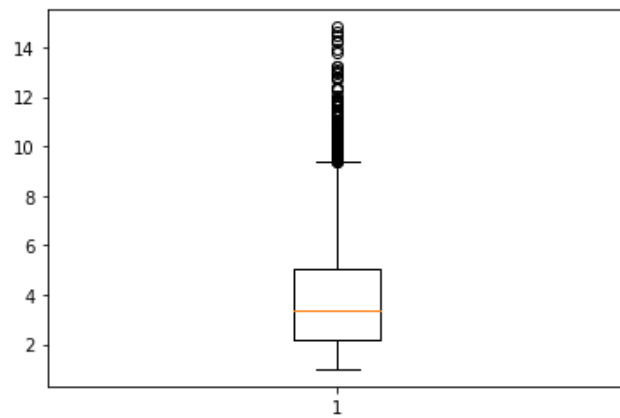


Figure 6.2: Real ET Values Box Plot

Site ID	% ET in [1-15]	% ET in [2-15]	% ET in [3-15]	% ET in [4-15]	% ET in [5-15]	% ET in [6-15]
FI-Jok	100%	66.67%	41.67%	19.87%	6.41%	4.49%
DE-Kli	100%	77.81%	54.50%	38.59%	21.58%	8.63%
US-WBW	100%	79.80%	66.73%	50.50%	35.05%	17.43%
US-A32	100%	79.76%	67.21%	55.87%	40.08%	25.10%
US-Me2	100%	66.54%	31.89%	6.69%	1.18%	0.00%

Figure 6.3: Snippet of Rare Study per Site

6.3 Relevance Matrix and Relevance Threshold

(Ribeiro and Torgo, 2008) allowed the choice between two methods when defining rare values: range and extremes. In extremes, an automated process decides the range of rare values depending on the box plot of the target variable (real ET in our case). It then allocates bigger importance to the least represented target variable values. Here, the user does not have to supply any external data. In range, the user has to define a set of reference points, called a relevance matrix, in order to assess relevance. In this study, it is defined as follows:

$$\text{Relevance Matrix} = \begin{pmatrix} 1 & 0 & 0 \\ 4 & 0 & 0 \\ 15 & 1 & 0 \end{pmatrix} \quad (6.2)$$

With 1-4 being the most represented, and 5-15 is the least represented. The user also has to define a relevance threshold. The threshold ranges between 0-1 (similar to the relevance). The algorithm works in a way that it assigns a rare label to the target variable value if the relevance of the target variable value is above the relevance threshold. The relevance threshold is decided per experiment, in which it is part of the cross-validation. The threshold giving the highest precision and recall will be chosen. We plot the real ET values and their corresponding relevance in **Figure 6.4**.

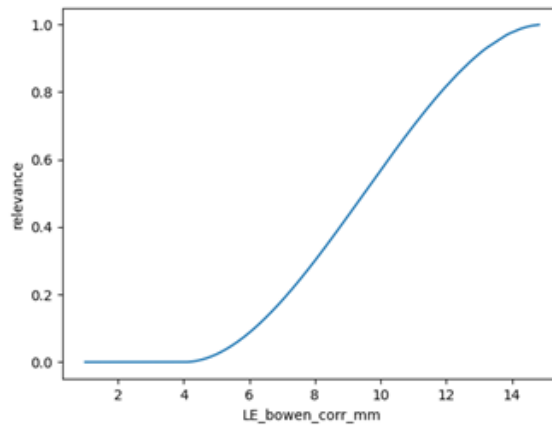


Figure 6.4: Relevance per ET Value

6.4 SMOGN

Precision and recall which are obtained from the UBR transformation serve more than just measuring model accuracy. Precision implies the percentage of relevant results. On the other hand, recall implies the percentage of relevant results

correctly classified by the specified algorithm, i.e the percentage of rare but relevant values accurately predicted by our machine learning model. In order to get higher recall values, we applied the works of (Branco et al., 2017). SMOGN (a combination of Gaussian noise and SmoteR) is a new method that tackles imbalanced data in any regression problem. The most important variables are rare cases that are poorly represented in our data. For example, high values of real ET are poorly represented in our data, however, they are very relevant and the farmer would be highly affected if these high ET values were wrongly predicted. Hence, SMOGN generates more instances of these rare but relevant ET values in order for the ML model to learn better, and thus predict better. This algorithm consists of a random under-sampling technique and two over-sampling ones i.e SmoteR and Gaussian Noise.

6.4.1 SMOTER and Gaussian Noise

SmoteR (Synthetic Minority Oversampling Technique for Regression) is developed by (Branco et al., 2013). The regular SMOTE algorithm targeted classification problems when there exists imbalanced class distribution. SMOTER, a variant of SMOTE, aims at regression problems, where the concern is to accurately predict rare extreme values. The original SMOTE algorithm creates synthetic samples by randomly choosing a k-nearest neighbor for a minority data point, and creates a new synthetic data point by interpolating the values of the original and its neighbor. SMOTE is visually explained in **Figure 6.4** below.

Since SMOTE is targeted at classification problems, the output variable intersects. Thus, in SMOTER, three main points should be identified:

1. Defining rare versus non-rare ET values
 - This point is already covered above, in which we decided the range of ET between [5-15] to be rare, and the range of ET between [1-4] to be non-rare.
2. Creating synthetic samples
 - The same method for SMOTE is applied.
3. Generating logical output variables (real ET) for the synthetic test points
 - This is a simple matter in the existing algorithm since all rare cases have the same class (target minority class). The solution is not so obvious in the case of regression. Although they have high relevance, the cases that need to be over-sampled do not have the same target variable value. This implies that they will not have the same target

variable value when a pair of data points is used to create a new synthetic case. (Branco et al., 2013) suggested using a weighted average of the two examples' target variable values. The weights for each of the two seed examples are calculated as an inverse function of the distance of the case generated.

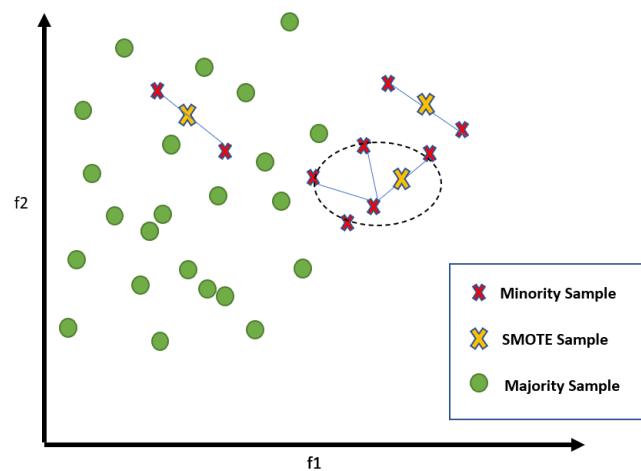


Figure 6.5: SMOTE (Branco et al., 2013)

SMOIGN will generate more synthetic data using SmoteR when the selected seed and the K-nearest neighbors are close in distance and it introduces Gaussian Noise when they are not. The main idea of the SMOIGN algorithm is to incorporate both resources to gain synthetic examples with the objective of potentially restricting the hazards that SmoteR will inflict by introducing the Gaussian Noise strategy and facilitating the diversity of the data.

In the SMOIGN module, we have the choice to down-sample the non-rare data and oversample the rare data. In this paper, we decided not to do any down-sampling and only apply the oversampling of rare values. The SMOIGN module also allows users to choose the oversampling percentage, the k for the k-nearest neighbor, and the distance type (Manhattan, Euclidean, etc...). We also note that we are applying the SMOIGN up-sampling technique on the training data only, not on the validation and testing. SMOIGN is also followed by repeated K-Fold stratified cross-validation, to avoid data leakage.

6.5 Utility-Based Regression and SMOGN Hyper-parameters

There are a set of hyper-parameters that need tuning when implementing UBR and SMOGN:

1. `rel_method`: relevance method - range or extremes.
2. `extr_type`: extreme type - high or low (meaning if rare values are high or low)
3. `rell`: relevance matrix (specified in the study)
4. `thr_rel`: threshold for relevance
5. `k`: K for K-nearest neighbor
6. `epl`: boolean value controlling the possibility of having a repetition of examples when performing under-sampling by selecting among the normal examples.
7. `dist`: distance function used (Manhattan, Euclidean, etc..)
8. `cperc`: A list containing the percentage(s) of under-or/and over-sampling to apply to each class.
9. `p`: A number indicating the value of p if the p-norm distance is chosen.

6.6 Repeated Stratified K-Fold Cross-Validation

The data set will be split first into 70% train and 30% test. The train data set is then further split into 70% train and 30% validation. We implemented a special split in which each site is equally represented in each split. We perform repeated Stratified K-Fold cross-validation. K-Fold cross-validation on its own works by splitting the data into K folds, where each fold represents training and validation data. The model will be evaluated at each fold respectively, and the performance of the model is assessed by averaging the scores across all folds. Stratified K-fold ensures that the data is split into stratified folds, the folds are made by preserving the percentage of samples for each class (rare vs not rare in the case of applying UBR). The repeated stratified K-fold Cross-validation works on repeating the stratified K-fold cross-validation by n repeats, in which each repeat, the data is shuffled to produce various fold combinations. The model is thus assessed by averaging the scores over all the repeats.

Chapter 7

Feature Selection

Feature selection is a process of choosing an adequate number of input features, which yields satisfactory model performance compared to all sets of features. Since there are plenty of challenging input features to obtain, feature selection must be done. In the feature selection process, we care to:

1. Reduce overfitting: Less redundant data means less opportunity for redundant data/noise-based decision making.
2. Improve accuracy: Less inaccurate data implies that the accuracy of the models improves.
3. Reduce training time: Less knowledge ensures that algorithms learn faster.

7.1 Methods

A supervised feature selection method is divided into three groups ([Miao and Niu, 2016](#)):

- Intrinsic: It is a method that achieves automated selection of features during the training phase.
- Filter: It uses statistical techniques to assess how a subset of input features relates to the target variable.
- Wrapper: It is a method that picks a subset of input features based on trying out different subsets of input features and monitoring all others which perform best in predictive modeling.

We have tried two methods for feature selection, one is a filter-based method based on mutual information and the other is a wrapper method based on feature importance.

7.1.1 Feature Importance

A wrapper method is a method which depends on the machine learning model used. The assessment of importance for each feature is based on model performance when that feature is included in training. The Feature Importance method searches for well-performing subsets of features. Using our best models, we obtain the analysis in **Figure 7.1**.

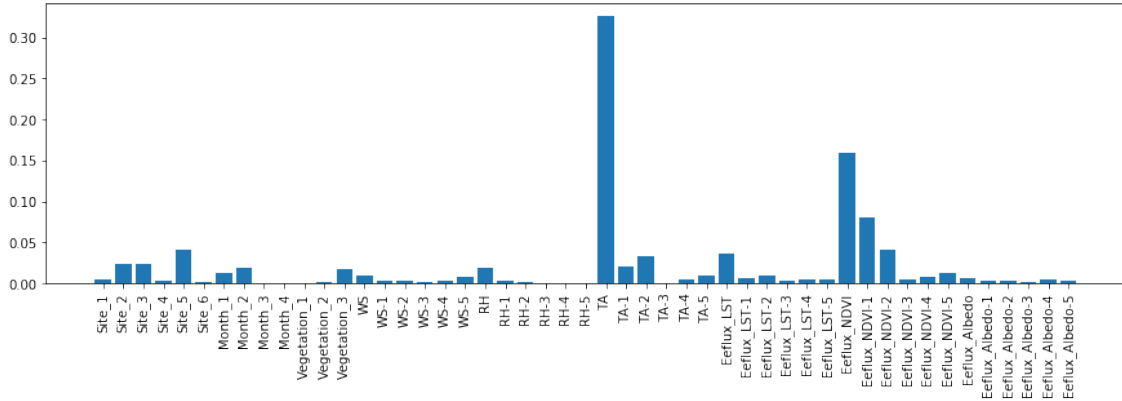


Figure 7.1: Feature Importance

Figure 7.1 shows the input features on the x -axis and the scores on the y -axis. The score is always positive and has a bounded value between 0 and 1. The higher the score, the better contribution an input feature has on its target. As it is noticed, the top contributing features are TA, NDVI, and LST.

7.1.2 Mutual Information Feature Selection

A filter-based feature selection method is independent of the model used. It chooses the top contributing features based on a correlation metric (here we chose mutual information) between the input features and the target feature. We chose this method as our final feature selection method since it is model agnostic. We obtained the results in **Figure 9.1**.

We note that the top contributing features are TA, NDVI, LST, Albedo, WS, and RH.

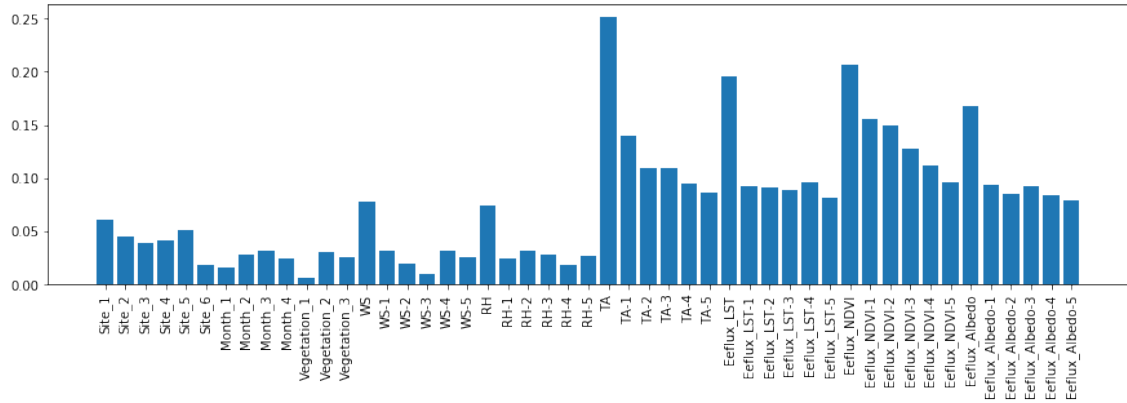


Figure 7.2: Feature Selection based on NMI

7.2 Scenarios

The feature selection process starts by ranking all input variables by Mutual Information Score using a filter-based FS algorithm (SelectKBest according to Mutual Information). Then, our model will be fed the input features one by one in order of highest Mutual Information Score. The feature selection process is quite important for finding which scenario best fits the business requirements. We also use feature selection techniques to reduce the complexity of our problem to make a compromise between the number of input features and accuracy, and training speed resulting in the following scenarios:

7.2.1 Scenario A

No feature selection methods are applied. The model is fed the total number of input features. The columns for this scenario are Site Encoded, Month Encoded, Vegetation Encoded, TA + 5 Lags, NDVI +5 lags, LST + 5 lags, Albedo + 5 lags, WS+ 5 lags, and RH + 5 lags.

7.2.2 Scenario B

The model with satisfactory performance but the least training time and economically inexpensive. What is meant by economically inexpensive is that it should require input features not hard to collect, i.e input features that are frequently available. The columns for this scenario are Site Encoded, Month Encoded, Vegetation Encoded, TA and 5 of its lags.

7.2.3 Scenario C

The model performs in a comparable manner when fed a number of features less than the total, as opposed to a model that is fed all the number of input features. The columns for this scenario are Month Encoded, Site Encoded, TA + 5 Lags, NDVI + 2 lags, LST + 5 lags, Albedo + 2 lags, WS+ 2 lags, and RH + 3 lags.

7.2.4 Scenario D

The model is fed top contributing input features extracted from SHAP, which are RH, TA, EEflux LST, and their corresponding lags.

The final the scenarios are illustrated in **Table 7.1**.

Name	Input Combinations
Scenario A	All Columns
Scenario B	TA(5 lags)
Scenario C	TA(5 lags) WS(2 lags) RH(3 lags) EEflux LST(5 lags) EEflux NDVI(2 lags) EEflux Albedo(2 lags)
Scenario D	TA(5 lags) RH(3 lags) EEflux LST(5 lags)

Table 7.1: Feature Selection Scenarios

Chapter 8

Model Assessment

8.1 Evaluation Metrics

In order to evaluate our models, we have used regression, correlation and agreement, utility-based regression, and probabilistic model selection metrics. .

8.1.1 Regression Metrics

All error metrics definition are derived from [scikit-learn](#).

- Mean Squared Error(MSE): A metric that is mostly used for regression problems. It measures the average of the squared difference between the target variable and its predicted value. Having the squared behavior plays a role in over-estimating how bad the model behaves and is often a preferable metric given the fact that it is differentiable which helps in optimizing it more.

$$MSE = \frac{1}{n} \sum (y - \hat{y})^2 \quad (8.1)$$

where:

n = number of observations

y = array of the target variable values

- Root-Mean-Squared-Error(RMSE): It is the square root of the MSE metric. It is preferable in cases where large errors are undesired since it first squares the error before applying the average.

$$RMSE = \sqrt{\frac{1}{n} \sum (y - \hat{y})^2} \quad (8.2)$$

- Mean-Absolute-Error(MAE): A linear score metric that measures the absolute difference between the target variable and its predicted value. It is

often more robust to outliers and doesn't penalize errors as MSE.

$$MAE = \frac{1}{n} \sum |y - \hat{y}| \quad (8.3)$$

- Mean-Absolute-Percentage-Error(MAPE): A metric that measures the percentage of how bad the model performs. It is the percentage of MAE. The least desirable value is 100% and the most desirable is 0%.

$$MAPE = \frac{100}{n} \sum \frac{|y - \hat{y}|}{|y|} \quad (8.4)$$

- Accuracy: A metric that measures the percentage of how well a model performs. It is quite the opposite of MAPE. A high value close to 100 indicates a well-performing model. The least desirable value is 0% and the most desirable is 100%.

$$Accuracy = 1 - MAPE \quad (8.5)$$

- Coefficient of Determination(R^2): A metric that helps in comparing our model's performance to a base-line model, which is the mean of the data, to indicate how well our model behaves in a scale-free matter scaling up to 1. A value close to 1 indicates that the model is performing well.

$$R^2 = 1 - \frac{MSE(model)}{MSE(baseline)} \quad (8.6)$$

- Adjusted R^2 : A metric that is similar to R^2 , it is often always less than R^2 since it only detects improvements when there is a real increasing predictor rather than improving on increasing terms while the model is not really improving.

$$R_a^2 = 1 - \left[\frac{n-1}{n-k-1} \right] \times (1 - R^2) \quad (8.7)$$

where:

n = number of observations

k = number of independent variables

R_a^2 = Adjusted R^2

8.1.2 Correlation Metrics and Agreement Metrics

All error metrics definition are derived from [scikit-learn](#).

- **Pearson Correlation Coefficient:** It is used to quantify the strength of the linear correlation between two entities x and y , where the value correlation = 1 means a positive correlation and the value correlation = -1 means a negative correlation. The least desirable value is a 0 and the most desirable is a 1.

$$Pearson = \frac{\sum (x - |\bar{x}|) \times (y - |\bar{y}|)}{\sqrt{\sum (x - |\bar{x}|)^2} \sqrt{\sum (y - |\bar{y}|)^2}} \quad (8.8)$$

- **Spearman Correlation Coefficient:** Similar to Pearson but does not measure the linear correlation, it measures the monotonic correlation between two entities. Monotonicity is “less restrictive” than that of a linear relationship. The least desirable value is a 0 and the most desirable is a 1.

$$Spearman = 1 - \frac{6 \sum d^2}{n^3 - n} \quad (8.9)$$

where:

n = data size

d = difference between ranks

- **Spatial Correlation Distance:** It is the correlation distance between two entities u and v . The least desirable value is a 0 and the most desirable is a 1.

$$Distance = 1 - \frac{(u - \bar{u}) \times (v - \bar{v})}{\|u - \bar{u}\|_2 \|v - \bar{v}\|_2} \quad (8.10)$$

- **Normalized Mutual Information (NMI):** A metric used to study the agreement between two independent labels when the real ground truth is unknown. It ranges between 0 (no mutual information) and 1 (perfect correlation).

$$NMI(Y, C) = \frac{2 \times I(Y, C)}{[H(Y) + H(C)]} \quad (8.11)$$

where:

Y = class labels

C = cluster labels

$H()$ = entropy

$I(Y, C)$ = mutual information between Y and C

8.1.3 Probabilistic Model Selection Metrics

All error metrics definition are derived from (Kuha, 2004).

- Akaike Information Criterion (AIC): A model selection metric that quantifies the quality of a model with respect to other models. A low AIC is an indication of a better model. It is often prone to over-fitting when adding more parameters which increase the model fitness this is why a penalty term is added for the number of parameters in the model.

$$AIC = n \times \log MSE + 2 \times num_params \quad (8.12)$$

where:

n = number of observations in the training data set

num_params = number of trainable models which is model dependent

- Bayesian Information Criterion (BIC): It is a metric for model selection, similar to AIC, and can be applied on a finite number of models. Its basis is on the likelihood function but adds a higher penalty.

$$BIC = n \times \log MSE + num_params \times \log(n) \quad (8.13)$$

8.1.4 Utility-Based Regression Metrics

After applying the UBR module to our data, we are now able to obtain cost-sensitive error metrics from our regression problem. Our regression problem is governed by rare but relevant events (extreme values not abundantly present in our dataset but are however relevant). Usually, in classification problems, predictions driven by rare/not rare values are measured by precision and recall. The general primacy of precision and recall is that they capture model performance upon encountering rare values and not on the general accuracy of not rare predictions (Torgo and Ribeiro, 2009).

- Precision: In regression, Precision is a calculated metric that appraises the rare/not rare predictions. The least desirable value is a 0 and the most desirable is a 1 (Torgo and Ribeiro, 2009).

$$Precision = \frac{\sum_{\phi(\hat{y}_i) \geq t_E} \alpha(\hat{y}_i, y_i) \phi(\hat{y}_i)}{\sum_{\phi(\hat{y}_i) \geq t_E} \phi(\hat{y}_i)} \quad (8.14)$$

where:

- \hat{y}_i is $y_{predicted}$
- y_i is y_{actual}
- ϕ is the relevance function
- t_E is the relevance threshold

- α is a function which defines the accuracy of the prediction, where $\alpha(\hat{y}_i, y_i) = I(L_{0/1}(\hat{y}_i, y_i))$
 - $I()$ is the indicator function given 1 if its argument is true and 0 otherwise
 - $L_{0/1}$ is a standard 0/1 loss function
- Recall: In regression, Recall computes the portion of events happening in the domain that are caught by the regression model. Recall is a calculated metric that appraises the number of true rare predictions yielded from all rare predictions that should have been made. The least desirable value is a 0 and the most desirable is a 1 (Torgo and Ribeiro, 2009).

$$Recall = \frac{\sum_{\phi(y_i) \geq t_E} \alpha(\hat{y}_i, y_i) \phi(\hat{y}_i)}{\sum_{\phi(y_i) \geq t_E} \phi(\hat{y}_i)} \quad (8.15)$$

- F1: It captures the balance between the precision and the recall. The least desirable value is a 0 and the most desirable is a 1.

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} \quad (8.16)$$

- F2: The F2 score is used as well, in which twice the weight is given to recall as opposed to the weight given to precision. The least desirable value is a 0 and the most desirable is a 1.

$$F2 = 5 \times \frac{precision \times recall}{4 \times precision + recall} \quad (8.17)$$

- F0.5: F 0.5 score is used also in which we give twice as much weight to precision than recall. The least desirable value is a 0 and the most desirable is a 1.

$$F0.5 = 1.25 \times \frac{precision \times recall}{0.25 \times precision + recall} \quad (8.18)$$

8.2 Negative Accuracy and R^2 Score

There exist special cases in which negative R^2 scores and accuracy measures are exhibited.

8.2.1 R^2 Score

The equation of R^2 score as defined by scikit-learn is :

$$R^2 \text{ score} = 1 - \frac{\text{residual sum of squares}}{\text{total sum of squares (proportional to the variance of the data)}} \quad (8.19)$$

Where:

$$\text{Residual Sum of Squares (SSres)} = \sum_i (y_i - f_i)^2 \quad (8.20)$$

$$\text{Total Sum of Squares (SStot)} = \sum_i (y_i - \bar{y})^2 \quad (8.21)$$

y_i is the actual target variable

f_i is the predicted target variable by the model

\bar{y} is the mean target variable

Since the R^2 score is defined as the proportion of variance explained by the fit, R^2 is negative if the fit is actually worse than just fitting a horizontal line. The R^2 score is calculated on the premise that the target variable's average line has to be the worst fit a model can have. The square difference between this average line and original data points is $SStot$ (total number of squares). Likewise, the square difference between the expected data points (by the model plane) and initial data points is $SSres$ (residual sum of squares). $SSres/SStot$ includes a ratio of how much worse $SSres$ is than $SStot$. If our model is capable of building a plane that is reasonably good than the worst, then $SSres \leq SStot$ in most cases. Ultimately, it makes the R^2 score positive. And what if $SSres \geq SStot$? This implies that our plane of regression is even worse than the mean line ($SStot$). The R^2 score in this case is going to be negative.

N.B: The $SSres$ is the regression plane because in its calculation the predicted y from the regression model is utilized. The $SStot$ represents the mean line because in its calculation the mean y is subtracted from the y actual.

8.2.2 MAPE

The equation of MAPE (Mean Absolute Percentage Error) is as follows:

$$\frac{1}{n} \sum_{i=1}^n \left| \frac{y_i - \hat{y}}{y_i} \right| \times 100 \quad (8.22)$$

A MAPE 100% means that the errors are much greater than the actual values. For example, if the actual target variable is a 1, and we predict a 3, the MAPE is 200%. Hence, the accuracy, which is 100-MAPE will be negative.

Chapter 9

Uncertainty Quantification

Machine learning models demand uncertainty measurement. In any machine learning task, there exist several sources of uncertainty, namely variations in particular data points, data samples obtained, and the flawed design of any model built from this type of data. The measurement of uncertainty prevalent in machine learning is accomplished by probability methods and techniques which are explicitly developed to deal with uncertainty. It is ultimately necessary and technically more significant to measure and quantify uncertainty in a transductive manner, customizing it for particular cases, rather than measuring generic error metrics or accuracy. In particular, uncertainty is defined as what is not clearly known but can be quantified differently, by taking into account its causes or effects as well. Therefore, there are different methodologies of uncertainty definitions that include various thoughts and implementations (aleatoric and epistemic, irreducible, and reducible) in addition to their forms of inference (systematic or statistical). In this section, we measure uncertainty using our best probabilistic model, NGBoost (which has its architecture and mode of action explained in [Section 12.1](#)).

9.1 Uncertainty Types

The concept of uncertainty is of great significance and is a core element in machine learning. Uncertainty has traditionally been viewed as nearly interchangeable with generic probability and probabilistic forecasts. However, because of the growing importance of machine learning for real-world applications, machine learning researchers have recently identified fresh difficulties, yet these issues may require new technical innovations ([Hüllermeier and Waegeman, 2019](#)). This includes the importance of quantifying uncertainty and differentiating between two forms of uncertainty: the aleatoric (irreducible) and epistemic (reducible). **Figure 9.1** illustrates an overview of the types of uncertainties to be discussed in [Sections 9.1.1](#) and [9.1.2](#)

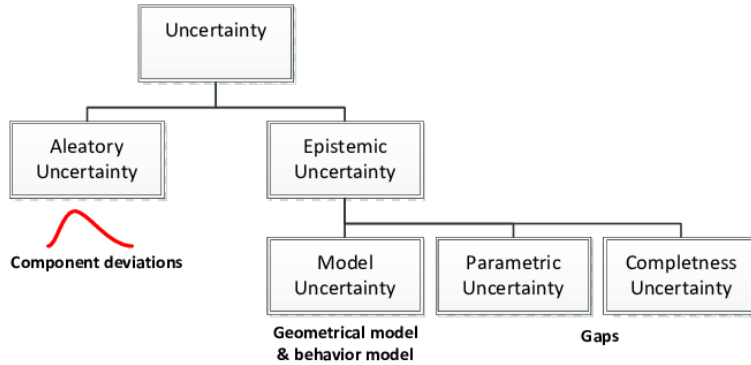


Figure 9.1: Uncertainty Types

9.1.1 Aleotoric Uncertainty

As (Hüllermeier and Waegeman, 2019) explain, aleatoric uncertainty - which is based on a statistical inference - supports the concept of randomness, i.e. the variance due to necessarily random effects of the results of an experiment. For example, rolling dice is a basic example of aleatoric uncertainty in which the outcome data cannot be changed given any additional outside information. Aleatoric uncertainty signifies the irreducible uncertainty because it is based on the stochastic connection between input variables x and output variable y , which is expressed as the conditional probability. This uncertainty is irreducible (cannot be decreased) even when additional data or model information is supplied to the machine learning model. In modern regression models, aleatoric uncertainty is mainly calculated as follows: (Hüllermeier and Waegeman, 2019)

$$aleoteric_uncertainty = \sigma_{RMSE}^2 \quad (9.1)$$

where σ_{RMSE}^2 is the variance of an error term (here we chose RMSE - Root Mean Squared Error). When quantifying uncertainty, each test data point should have an uncertainty measure. To measure epistemic uncertainty, we should get the variance of the probability distribution per point. This is achievable in NGBoost because NGBoost outputs the mean and standard deviation of the distribution per test data point, so we calculate the variance from the obtained standard deviation. However, in order to calculate the aleatoric uncertainty, we need the variance of the RMSE per point. Hence, each test data point should have a list of RMSEs in order to calculate the variance per data point. This is achievable in neural networks because of their layered property, where we can play around and achieve multiple RMSEs per point. For example, the deep ensemble neural network architecture consists of an ensemble of neural networks. Each neural network of this ensemble is tested on the testing data. Therefore, per data point, we have an array of RMSEs, each RMSE pertaining to a neural network model

in this ensemble. Hence, we can calculate the variance of the RMSE per point. However, this is not achievable in NGBoost because it is a tree structure, where each test data point has 1 RMSE, not a list of RMSEs, not allowing us to calculate the variance of the RMSE like in deep ensembles.

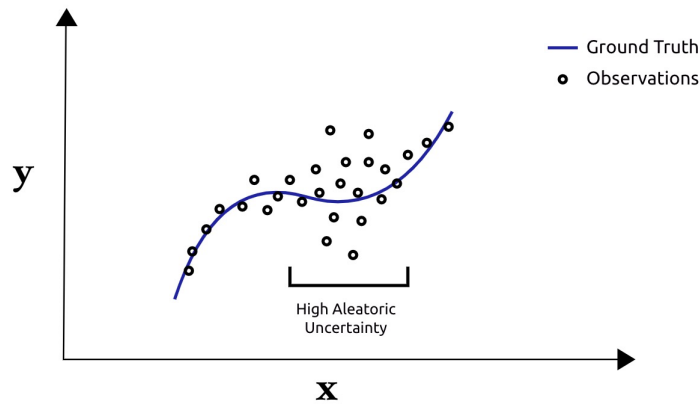


Figure 9.2: Aleatoric Uncertainty - 1 ([Why Uncertainty Matters](#))

Figure 9.2 represents a sample depiction of a set of their observations coupled with their ground truth. We note that noisy measurements of an underlying process can be the reason behind high Aleatoric uncertainty.

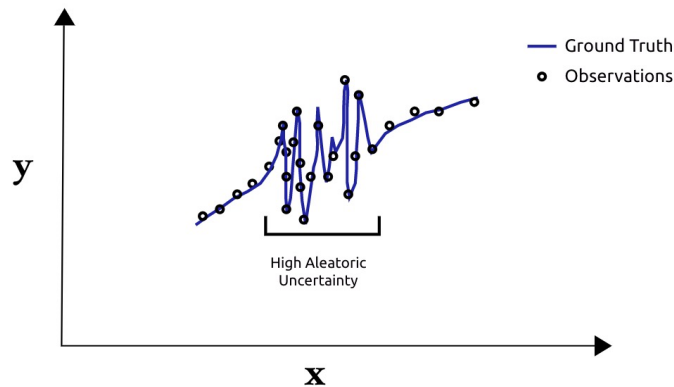


Figure 9.3: Aleatoric Uncertainty - 2 ([Why Uncertainty Matters](#))

Figure 9.4 represents a sample depiction of a set of their observations but coupled with error-less ground truth. We also observe a noisy underlying process that lead to high Aleatoric uncertainty even with accurate observations, hence

comes the term irreducible to describe this uncertainty (unaffected or minimized by additional data information).

The unit for Aleatoric uncertainty is mm^2 .

9.1.2 Epistemic Uncertainty

Epistemic uncertainty - which is based on a systematic inference - corresponds to an uncertainty created by insufficient information. Epistemic uncertainty relates to the confusion of the machine learning model. Hence, unlike the aleatoric uncertainty which was based on the occurrence of any random event, the epistemic uncertainty is based on the status of the machine learning model and its knowledge (Hüllermeier and Waegeman, 2019). This type of epistemic uncertainty can be minimized upon introducing additional knowledge to the machine learning model, unlike aleatoric uncertainty. Epistemic uncertainty is the reducible uncertainty, which can be reduced upon introducing extra model/data information. Epistemic uncertainty is calculated as follows: (Hüllermeier and Waegeman, 2019)

$$epistemic_uncertainty = \sigma_{PDF}^2 \tag{9.2}$$

Where σ_{PDF}^2 is the variance of the probability density function pertaining to the probability distribution predicted by the probabilistic model. In other words, σ_{PDF}^2 is the variance of the posterior distribution or probability density function pertaining to the probability distribution predicted by the probabilistic model.

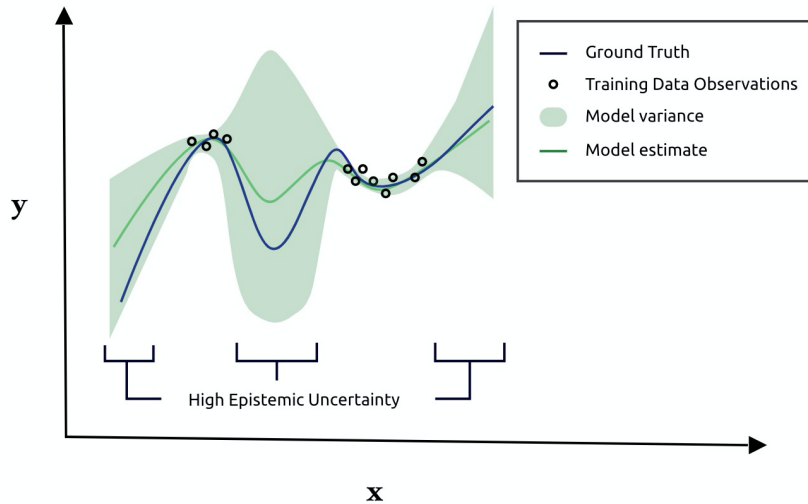


Figure 9.4: Epistemic Uncertainty (Why Uncertainty Matters)

Epistemic uncertainty is high when there is not enough training data, and hence the σ_{PDF}^2 is high, as portrayed in **Figure 9.4**. In fact, the concept of epistemic uncertainty is actually built upon Bayesian inference which is based on Bayes's theorem ([Gustafson, 2014](#)). Baye's theorem computes the probability of a certain event from prior knowledge or data. Let us consider a random unknown quantity θ . Baye's theorem places an initial guess about θ 's distribution, naming it the prior distribution. This prior distribution is hence updated upon gaining more knowledge on θ (same as how the epistemic uncertainty is reduced upon introducing additional data or model information). Baye's theorem is defined as follows:

$$P(\theta|D) = \frac{P(D|\theta) \times P(\theta)}{P(D)} \quad (9.3)$$

where:

1. D is the prior knowledge or data
2. θ is a given event
3. $P(\theta)$ is the prior distribution representing our initial guess for the distribution of θ
4. $P(D|\theta)$ is the likelihood distribution. The likelihood distribution is dependent on the $P(D)$ given that the likelihood is true, and then multiplied by a constant.
5. $P(D)$ is the marginal distribution - sometimes referred to as the evidence. Its calculation is a complex process. It is usually a normalizing constant.
6. $P(\theta|D)$ is the famous posterior distribution. It represents the probability distribution of θ after receiving and observing all the additional data/model information.

The Bayesian inference is further explained in **Figure 12.3** below, where it is shown that the prior distribution lies between the posterior and likelihood distributions.

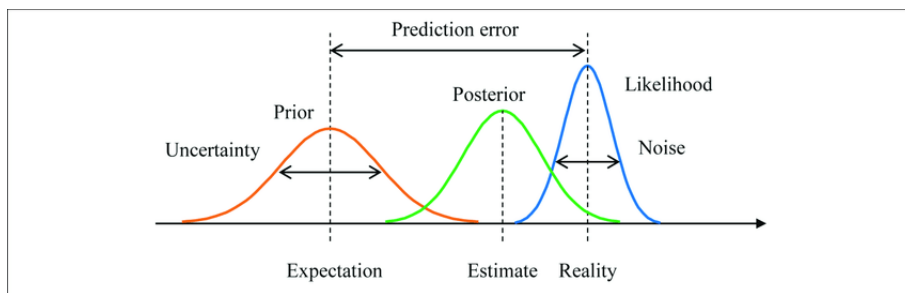


Figure 9.5: Bayesian Inference ([Gustafson, 2014](#))

Chapter 10

Experimental Variations

10.1 Experiment A

Real ET versus ET (predicted or obtained by EEflux METRIC). We denote this experiment by Experiment A. Error metrics are calculated between Variable 1 and Variable 2:

- Variable 1: Real ET
- Variable 2: Predicted ET or EEflux METRIC ET

High R2 (close to 1) and low error metrics indicate that our model is able to predict real ET well.

Research Question: Which model best estimates real ET?

10.2 Experiment B

The proportional residual between real ET and predicted ET is computed by getting the error metrics corresponding between two entities:

- Variable 1: EEflux ET / Real ET
- Variable 2: EEflux ET / Predicted ET

We denote this experiment by Experiment C.

Research Question: Which model best minimizes the proportional bias between the real ET and the EEFlux ET?

10.3 Experiment C

The absolute residual between real ET and predicted ET is computed by getting error metrics and other accuracy measures corresponding between two variables:

- Variable 1 = EEflux ET - Real ET
- Variable 2 = EEflux ET - Predicted ET

We denote this experiment by Experiment B.

Research Question: Which model best minimizes the absolute bias between the real ET and the EEFlux ET?

10.4 Experiment D

A combination between the multiplicative and the absolute residual. The combination residual between real ET and predicted ET is computed by getting the error metrics corresponding between two entities:

- Variable 1: $(\text{EEflux ET} - \text{Real ET})/\text{Real ET}$
- Variable 2: $(\text{EEflux ET} - \text{Predicted ET})/\text{Predicted ET}$

We denote this experiment by Experiment D.

Research Question: Which model best minimizes the combined bias between the real ET and the EEFlux ET?

Chapter 11

Point-wise Modeling

Point-wise models generate point predictions - i.e one predicted value per one input test point. We first started with a basic support vector regressor as our base model, which was coupled with 10 folds 10 repeated stratified cross-validation. SMOGN up-sampling was also applied. Upon observing unsatisfactory results for our base model which is Linear SVR, we explored the performance of Gradient Boost, Extra Trees, Random Forests, Ensemble Bagging, Ada Boost, CatBoost, and XGBoost.

11.1 Models

The description of all the models is adapted from [scikit-learn](#).

1. **Linear Support Vector Regression:** The objective function of Linear SVR is to minimize the coefficients, specifically the coefficient vector l_2 norm. Thus in the option of penalties and loss functions, it has more versatility and can scale to large numbers of samples better.
2. **Gradient Boost:** Gradient Boosting is a machine learning technique that produces a prediction model in the form of an ensemble of weak prediction models. Like other boosting techniques, it constructs the model in a phase-wise fashion and generalizes them by allowing an arbitrary differentiable loss function to be optimized.
3. **Extra Trees:** Extra Trees is an algorithm for ensemble machine learning that incorporates the predictions from multiple decision trees.
4. **Random Forests:** A Random Forest is a meta estimator that suits a variety of decision-making trees on different data subsets and utilizes averaging to enhance accuracy and reduce over-fitting.

5. **Ensemble Bagging:** Ensemble learning is a form of machine learning where several models are trained to solve the same problem and are later combined to produce better results. Ensemble Bagging often regards homogeneous poor learners, learns them in parallel independently from each other, and combines them according to a form of deterministic averaging method.
6. **AdaBoost:** An AdaBoost regressor is a meta-estimator that starts by fitting a regressor on the whole dataset and later fits copies of this regressor on the same dataset but with altered weights of instances in accordance to the current prediction error. Subsequent regressors will hence concentrate more on challenging test points.
7. **XGBoost:** XGBoost is an optimized library for distributed gradient boosting, engineered to be extremely powerful, scalable, and compact. Under the Gradient Boosting paradigm, XGBoost offers a parallel tree boost that easily and reliably addresses several regression problems.
8. **Auto-Sklearn:** Auto-sklearn is an automated machine learning tool that replaces any scikit-learn estimator. It gives users an opportunity for supervised machine learning. Auto-sklearn automatically looks for the best learning algorithm for the given dataset and optimizes the hyperparameters. Thus, it frees the machine learning practitioner from these tedious tasks and allows him/her to focus on the real problem. Auto-sklearn does not include deep learning models.

11.2 Choosing the Best Model

We have performed Experiment A on all point-wise models which were trained on all the input features. We would also like to note that the percentage of improvement in the following sections is calculated as:

$$\text{percentage of improvement} = \frac{\text{metric}_{\text{variable1}} - \text{metric}_{\text{variable2}}}{\text{metric}_{\text{variable1}}} \quad (11.1)$$

Where the metric is any error metric we are comparing between the two variables (Accuracy, RMSE, MAE, etc...).

The results are observed in **Tables** 11.1 and 11.2. We note that the top two performing models were Gradient Boost and Extra trees, with the first scoring an R2 of 0.633, an accuracy of 65.7%, an MAE of 1.056, and an F2 measure of 0.929, whereas the second scoring an R2 of 0.62, an accuracy of 65.9%, an MAE

	Metrics	Experiment A	
Models	Average ET	3.87926	
Linear SVR	Data sets	Validation	Testing
	F2	0.725781 +- 0.363565	0.848155
	R2	0.196681 +- 0.27087	-0.15421
	RMSE	1.973392 +- 0.34798	2.486121
	MAE	1.56825 +- 0.330283	1.991371
	Accuracy	45.31988 +- 13.34844	45.51837
	NMI	0.999571 +- 0.000211	0.997618
	Training Time (s)	NA	0.992755
	Testing Time (s)	NA	0.006835
Gradient Boost	F2	0.914566 +- 0.008088	0.929839
	R2	0.595749 +- 0.04218	0.63341
	RMSE	1.456111 +- 0.073924	1.311459
	MAE	1.056378 +- 0.036625	0.990982
	Accuracy	64.75955 +- 2.217643	65.68292
	NMI	0.999175 +- 0.000614	0.997899
	Training Time (s)	NA	0.177992
	Testing Time (s)	NA	0.002279
	Extra Trees	F2	0.428902 +- 0.354878
R2		0.476546 +- 0.114703	0.620088
RMSE		1.535501 +- 0.34275	1.406279
MAE		1.148595 +- 0.230346	1.029289
Accuracy		61.4985 +- 9.157579	65.86307
NMI		0.270833 +- 0.078712	0.212276
Training Time (s)		NA	3.083522
Testing Time (s)		NA	0.02574

Table 11.1: Point-wise Models Results part 1

	Metrics	Experiment A	
Models	Average ET	3.87926	
Random Forests	Data sets	Validation	Testing
	F2	0.486061113 +- 0.323221702	0.786444689
	R2	0.450883926 +- 0.111755125	0.593321667
	RMSE	1.574282052 +- 0.338207369	1.454974596
	MAE	1.187299606 +- 0.211431136	1.091651016
	Accuracy	61.23807793+- 7.868750036	62.55916559
	NMI	0.993161631 +- 0.001871528	0.992597673
	Training Time (s)	NA	5.749958992
	Testing Time (s)	NA	0.018198729
Ensemble Bagging	F2	0.488634 +- 0.324995	0.782509
	R2	0.443691 +- 0.11528	0.58665
	RMSE	1.585172 +- 0.11528	1.46686
	MAE	1.196087 +- 0.218719	1.103313
	Accuracy	60.79691 +- 7.92898	61.8822
	NMI	0.993162 +- 0.001872	0.992598
	Training Time (s)	NA	7.811507
	Testing Time (s)	NA	0.030581
Ada Boost	F2	0.276052 +- 0.339753	5.00E-05
	R2	0.296989 +- 0.148171	0.432491
	RMSE	1.793102 +- 0.429293	1.718764
	MAE	1.354429 +- 0.273296	1.310485
	Accuracy	54.96582 +- 11.57557	53.19274
	NMI	0.603475 +- 0.037421	0.608152
	Training Time (s)	NA	0.154175
	Testing Time (s)	NA	0.00277
XGBoost	F2	0.583904393 +- 0.339753	0.79519714
	R2	0.437403573 +- 0.148171	0.580627208
	RMSE	1.589694449 +- 0.429293	1.477508603
	MAE	1.196082017 +- 0.273296	1.082356466
	Accuracy	60.74221564 +- 11.57557	65.61014119
	NMI	0.993161631 +- 0.037421	0.992597673
	Training Time (s)	NA	0.174965382
	Testing Time (s)	NA	0.003976583

Table 11.2: Point-wise Models Results part 2

of 1.148, and an F2 measure of 0.79. However, it is quite noticeable that Gradient Boost beat Extra trees by 15% in F2 measure and by 7.3% in RMSE. We would also like to highlight that Gradient Boost is not over-fitting, unlike Extra Trees which has validation scores much lower than the testing scores. We also like to add that Gradient Boost is one of the fastest models, with a training time of fewer than 0.2 seconds.

11.3 Utility-Based Learning and SMOGN up-sampling

We have applied SMOGN up-sampling to all our pointwise machine learning models. However, we would like to shed light on the results of our best (Gradient Boost) and base (Linear SVR) models. The details for the SMOGN hyperparameters for both models are found in **Sections** 11.6.1 and 11.6.2. We portray the results of both models when trained on the full dataset before and after SMOGN in **Table** 11.3 (before and after SMOGN scores for all other models are found in the appendix):

	Metrics	Before SMOGN		After SMOGN	
Models	Average ET	3.879262		3.879262	
Gradient Boost	Train Size	3576		3576	
	Test Size	1547		1547	
	Data sets	Validation	Testing	Validation	Testing
	Recall	0.9 +-0.0101	0.892	0.915 +- 0.008	0.93
	F2	0.918 +- 0.008	0.910	0.915 +- 0.008	0.930
	R2	0.6 +-0.018	0.586	0.596 +- 0.042	0.633
	RMSE	1.417 +-0.037	1.470	1.456 +- 0.074	1.311
	MAE	1.04 +-0.023	1.076	1.056 +- 0.037	0.991
	Accuracy	65.049 +-1.856	63.824	64.760 +- 2.218	65.683
	NMI	0.999 +-0.0003	0.998	0.999 +- 0.001	0.998
	Training (s)	NA	0.199	NA	0.178
Testing (s)	NA	0.002524	NA	0.002	
Linear SVR	Recall	0.86 +- 0.0465	0.825	0.868 +- 0.066	0.912
	F2	0.8 +-0.268	0.866	0.554 +- 0.452	0.920
	R2	0.046 +- 0.600	0.130	-0.06 +- 0.507	0.253
	RMSE	2.125 +- 0.599	2.115	2.252 +- 0.548	1.986
	MAE	1.697 +- 0.623	1.629	1.85 +- 0.541	1.591
	Accuracy	43 +- 31.073	55.031	36.898 +- 17.934	35.916
	NMI	0.999 +- 0.0003	0.998	0.999 +- 0.0003	0.998
	Training (s)	NA	1.339	NA	1.025
	Testing (s)	NA	0.007	NA	0.002

Table 11.3: SMOGN Comparison

We note that after applying SMOGN, all error metrics changed for the better. In Gradient Boost, the Recall boosted from 0.892 to 0.93, the F2 measure boosted from 0.910 to 0.930, the R2 from 0.586 to 0.633, the RMSE from 1.470 to 1.311, the MAE from 1.076 to 0.991, the accuracy from a 64% to a 66%. The same case applies for Linear SVR, where the Recall increased from a 0.25 to a 0.912, the F2 measure increased from a 0.866 to a 0.920, the R2 from a 0.130 to a 0.253, the RMSE decreased from a 2.115 to a 1.986, the MAE from a 1.629 to a 1.591.

Takeaway Message

SMOGN has left a positive impact on our best model Gradient boost, increasing the Recall by 4.2%, the F2 measure by a 2.1%, the R2 score by 8%, and the accuracy by a 2%.

11.4 Base Model

Our base model was Linear Support Vector Regressor. Linear SVR was the first model we experimented on, which was coupled with 10 folds 10 repeated stratified cross-validation, and SMOGN up-sampling.

11.4.1 Architecture

Linear SVR gives us the ability to define how much error is acceptable in our model (Awad and Khann, 2015). It will find a suitable higher dimensional hyperplane to fit the data. The objective function of SVR minimizes the coefficients, more specifically the coefficient vector l_2 -norm, not the squared error. The error term is handled in the constraints, where the absolute error is set to less than or equal to a given margin, which is referred to as the maximum error, ϵ (epsilon). The objective function and the constraints are as follows:

$$\text{Minimize : } \quad \text{MIN } \frac{1}{2} \|w\|^2 \quad (11.2)$$

$$\text{Constraints : } \quad \|y_i - w_i x_i\| \leq \epsilon \quad (11.3)$$

where:

- w is the coefficient vector
- x_i and y_i are our input and output test data points respectively at a given point i .

11.4.2 Hyper-parameters

The hyper-parameters we tuned for Linear SVR are as follows:

1. C: regularization parameter
2. loss: type of loss - either epsilon insensitive or squared epsilon insensitive
3. max_iter: the maximum number of iterations needed before converging
4. tol: the tolerance for the stopping criteria

11.5 Best Model

Boosting is a way for weak learners to be transformed into good learners. Each new tree is fit on a different version of the original data set during boosting. Gradient Boosting is a machine learning technique that produces a prediction model in the form of an ensemble of weak prediction models, usually decision trees, for regression and classification problems. As other boosting techniques do, it constructs the model in a phase-wise fashion and generalizes them by allowing an arbitrary differentiable loss function to be optimized. Gradient Boost gave the best results across all point-wise models, which was coupled with 10 folds 10 repeated stratified cross-validation, and SMOGN up-sampling.

11.5.1 Architecture

Gradient Boosting is a form of boosting for machine learning. It is based upon the belief that when paired with previous models, the next best possible model minimizes the cumulative prediction error. In order to alleviate the error, the main concept is to set the main goal for this next model. The logic in which the subsequent predictors learn from the errors of the previous predictors are used in this method. (Natekin and Knoll, 2013) Therefore in subsequent models, the results have an unequal likelihood of occurring and those with the highest error appear more. Decision Trees are the predictor we are using here. Because new predictors learn from mistakes made by previous predictors, it takes less time and error to achieve high accuracy. The Gradient Boost algorithm is shown in **Figure 11.1** below.

The idea, therefore, is to repeatedly exploit the residual patterns to reinforce and enhance a model with poor predictions. We can stop modeling residuals until we reach a point where residuals do not have any pattern that can be modeled (or the model would hence over-fit). Algorithmically, our loss function is reduced, so that the test loss reaches its minimum. A model's fit to a data point is determined

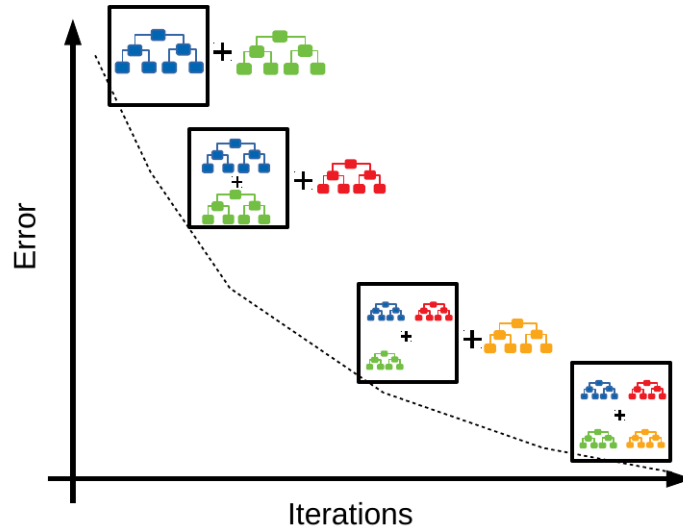


Figure 11.1: Gradient Boosting

by its residual value, identified as the difference between the dependent variable's actual value and the model's expected value.

$$residual = y_{actual} - y_{predicted} \quad (11.4)$$

In Gradient Boosting, we aim to minimize the least squares error loss function (the sum S of the squared residuals):

$$S = \sum_{i=1}^n residual^2 \quad (11.5)$$

11.5.2 Hyper-parameters

The hyper-parameters we tuned for Gradient Boost is as follows:

1. Number of estimators: Represents the number of trees used in the model
2. Min Sample Split: Represents the minimum number of samples required in a node before it is split
3. Min Sample Leaf: Represents the minimum samples required in a leaf.
4. Max Features: Represents the number of features considered when searching for the best split.
5. Max depth: Represents the maximum depth of a tree.
6. Learning Rate: Represents the impact of each tree on the final outcome.

7. Subsample: Represents the fraction of observations to be selected for each tree. Selection is done by random sampling

11.6 Implementation

For both Gradient Boost and Linear SVR, we have used the generic Scikit-learn library's implementation. However, the generic Scikit-learn library outputs a predicted and true output variable only. We have developed a point-wise models code module that implements stratified repeated K-Fold cross-validation, utility-based learning, SMOGN up-sampling, dynamic variable scaling, and dynamic model selection. This module outputs validation scores, standard deviation validation scores, test scores, in addition to all necessary plots, figures, and datasets needed.

11.6.1 Linear SVR Model

We have utilized Scikit-learn's implementation of Linear SVR which is embedded in our point-wise prediction module. We tuned the Linear SVR parameters by utilizing the stratified 10 repeated 10 folds cross-validation and SMOGN up-sampling. Each range of values for each hyper-parameter is chosen based on what is accepted in the literature:

- C: [0.1, 0.001, 0.0001, 0.01, 0.2, 0.002, 0.0002, 0.02]
- loss: epsilon_insensitive, squared_epsilon_insensitive
- max_iter: [100, 200, 300, 500, 1000, 1200, 1100, 1500]
- tol: [1e-4, 1e-3, 1e-2, 1e-5]

The best hyper-parameter combination giving the highest accuracy and lowest error metrics is:

- C: [0.1]
- loss: epsilon_insensitive
- max_iter: [1000]
- tol: [1e-4]

The best UBR and SMOGN parameters are:

- rel_method=range
- extr_type=high

- coef=1.5
- relevance_pts = np.array([[1, 0 , 0],[4, 0 , 0],[15, 1 , 0],])
- thr_rel=0.2
- Cperc=np.array([1,1.2])
- m=5
- repl=False
- dist=Manhattan
- p=2
- pert=0.1

11.6.2 Gradient Boost

We have utilized Scikit-learn's implementation of Gradient Boost which is embedded in our point-wise prediction module. We tuned the hyper-parameters of Gradient Boost where we tried a set of over 300 hyper-parameters using stratified 10 repeated 10 folds cross validation and SMOGN up-sampling. The range of values for each hyper-parameter is chosen according to what is deemed acceptable in the literature:

- n_estimators: [30,35,40,45, 50, 60, 70, 80]
- min_samples_split: [50,60,70,80,90]
- min_samples_leaf: [50,60,70,80,90]
- max_features: [0.5,0.6,0.7,0.8]
- max_depth: [4, 5, 6, 7, 8, 9, 10, 11 , 12, 14]
- learning_rate: [0.1, 0.001, 0.002, 0.01, 0.02]

The best hyper-parameter combination giving the highest accuracy and lowest error metrics is:

- n_estimators: [35]
- min_samples_split: [90]
- min_samples_leaf: [60]
- max_features: [0.5]

- max_depth: [4]
- learning_rate: [0.1]

The best UBR and SMOGN parameters are:

- rel_method=range
- extr_type=high
- coef=1.5
- relevance_pts = np.array([[1, 0 , 0],[4, 0 , 0],[15, 1 , 0],])
- thr_rel=0.2
- Cperc=np.array([1,1.2])
- m=5
- repl=False
- dist=Manhattan
- p=2
- pert=0.1

11.7 Results

11.7.1 Across Weather Clusters and Union of Clusters

We have conducted Experiment A across Gradient Boost, Linear SVR, EEflux METRIC, Auto-Sklearn, and Stat on all the data (union of clusters). We have also conducted this experiment on the latter mentioned models but on all available cluster combinations mentioned in **Section 5.4.4** on all weather parameters using k-means and dendrograms. The best performing cluster giving the best accuracy, recall, and the least error metrics was clustering by Wind Speed (WS) using k-means, with k=2. WS cluster one has wind speed values ranging from 3.647 to 12.518 meters per second. WS cluster two has wind speed values ranging from 0.0516 to 4.129 meters per second.

We have also conducted Experiment A on the latter mentioned models and on the best clusters across all feature selection scenarios (scenario B, scenario C, and scenario D). The best feature selection scenario giving the best accuracy, recall,

and the least error metrics was scenario C.

We portray the experimental results for scenario C in **Table 11.4**. All other experiments for scenarios (A, B, and D) are present in the appendix. The columns in our table represent:

- WS Cluster 0: The model is being trained on data clustered by WS using k-means, $k = 2$ on the first cluster.
- Union: The model is being trained on the union of clusters.
- WS Cluster 1: The model is being trained on data clustered by WS using k-means, $k = 2$ on the second cluster.

	Metrics	WS Cluster 0		Union of Clusters		WS Cluster 1	
Models	Average ET	3.745098		3.879262		3.951248	
EEflux	Train Size	1202		3576		2361	
	Test Size	530		1547		1025	
	Data sets	Validation	Testing	Validation	Testing	Validation	Testing
	Recall	NA	0.839	NA	0.880	NA	0.787
	F2	NA	0.850	NA	0.894	NA	0.000
	R2	NA	-0.489	NA	-0.540	NA	-0.577
	RMSE	NA	3.519	NA	2.479	NA	2.907
	MAE	NA	2.558	NA	1.939	NA	2.237
	Accuracy	NA	29.346	NA	47.612	NA	42.426
	NMI	NA	0.813	NA	0.892	NA	0.865
	Training (s)	NA	NA	NA	NA	NA	NA
	Testing (s)	NA	NA	NA	NA	NA	NA
Gradient Boost	Recall	0.872 +- 0.039	0.875	0.895 +- 0.016	0.917	0.911 +- 0.015	0.909
	F2	0.895 +- 0.035	0.898	0.912 +- 0.014	0.930	0.925 +- 0.012	0.924
	R2	0.444 +- 0.094	0.519	0.591 +- 0.043	0.637	0.639 +- 0.046	0.654
	RMSE	1.621 +- 0.227	1.527	1.44 +- 0.112	1.359	1.354 +- 0.090	1.355
	MAE	1.173 +- 0.110	1.101	1.053 +- 0.061	1.019	1.013 +- 0.051	0.996
	Accuracy	59.413 +- 6.336	59.377	64.85 +- 2.638	64.756	66.765 +- 1.817	66.834
	NMI	0.999 +- 0.001	0.998	0.999 +- 0.000	0.998	0.999 +- 0.000	0.999
	Training (s)	NA	0.083	NA	0.151	NA	0.095
	Testing (s)	NA	0.001	NA	0.002	NA	0.002
	Linear SVR	Recall	0.799 +- 0.076	0.858	0.862 +- 0.063	0.886	0.857 +- 0.051
F2		0.183 +- 0.366	0.894	0.637 +- 0.417	0.907	0.722 +- 0.362	0.908
R2		-0.121 +- 0.376	0.236	-0.101 +- 0.557	0.429	0.167 +- 0.403	0.445
RMSE		2.398 +- 0.520	1.761	2.309 +- 0.530	1.681	2.007 +- 0.395	1.711
MAE		1.869 +- 0.477	1.353	1.893 +- 0.548	1.302	1.583 +- 0.369	1.298
Accuracy		44.352 +- 18.018	58.945	34.123 +- 23.415	52.778	49.048 +- 11.210	56.926
NMI		0.999 +- 0.001	0.997	0.999 +- 0.000	0.998	1 +- 0.000	0.998
Training (s)		NA	0.194	NA	0.663	NA	0.421
Testing (s)		NA	0.001	NA	0.010	NA	0.001
Stat		Recall					
	F2	-	-	-	-	-	-
	R2	-	-	-	-	-	-
	RMSE	-	-	-	-	-	-
	MAE	-	-	-	-	-	-
	Accuracy	-	-	-	-	-	-
	NMI	-	-	-	-	-	-
	Testing (s)	-	-	-	-	-	-
Auto Sklearn	Recall						
	F2	NA	0.879	NA	0.906	NA	0.888
	R2	NA	-0.047	NA	0.601	NA	0.613
	RMSE	NA	1.917	NA	1.972	NA	1.401
	MAE	NA	1.373	NA	1.101	NA	0.863
	Accuracy	NA	58.152	NA	64.981	NA	71.72
	NMI	NA	0.372	NA	0.993	NA	0.794
	Testing (s)	NA	1200	NA	1200	NA	1200
Testing (s)	NA	80	NA	80	NA	80	

Table 11.4: Experimental Results for Point-wise Models

Table 11.4 shows the experimental results across different wind speed clusters (by Kmeans) and across the union of clusters. We compare the performance of our four models: EEflux (METRIC) model, Linear SVR (base model), Gradient Boost (best model), and AutoSklearn. We note that Gradient Boost outperforms all others in predicting real ET across the union of clusters, producing an R2 score of 0.637, an MAE of 1.019, an accuracy of 64.756 %, a recall of 0.917, and an F2 measure of 0.930. Transfer learning is also highlighted when training the model on the union of clusters. All the five models poorly performed on WS cluster 0, but excellently on WS cluster 1. This implies that the models have learned from WS cluster 1 yielding more accurate predictions for the union of clusters. The clusters are ordered from worst to best in terms of their results.

N.B: Auto Sklearn validation scores are not available for us to extract.

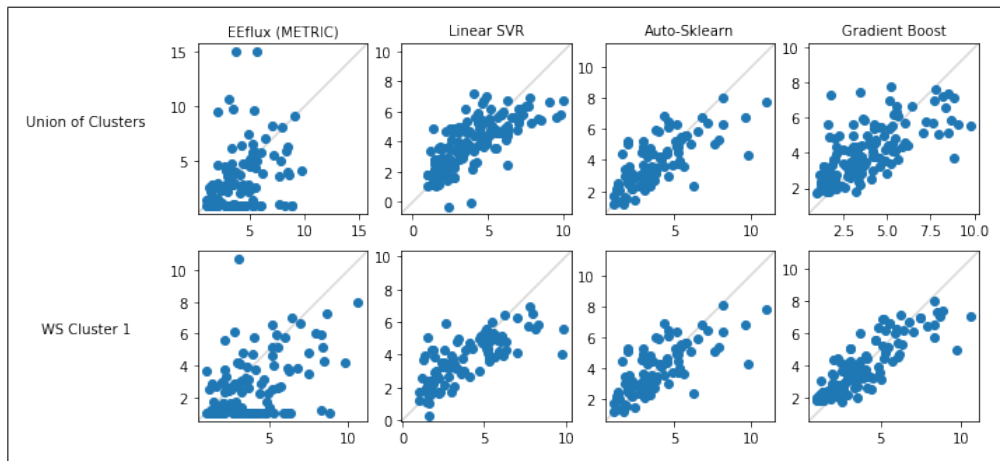


Figure 11.2: Scatter plot for Point-wise Models

Figure 11.2 illustrates the real ET (x -axis) versus the predicted ET (y -axis) across the four models: EEFlux (METRIC), Linear SVR, AutoSklearn, and Gradient Boost. It is noted that the Gradient Boost model shows a better diagonal fit than the EEFlux (METRIC) model across the union of clusters and WS cluster 1. The points in the EEflux model were scattered and not centered around the bisector. Gradient Boost trained on WS Cluster 1 yielded a better concentration around the bisector in comparison to the union of clusters. Gradient Boost and Auto-Sklearn showed a comparable performance. **Figure 11.2** also confirms our quantitative observations present in **Table 11.4**.

Table 11.5 shows the residual analysis experiments (B,C, and D) across the union of clusters and across the best performing cluster WS cluster 1 on both models Gradient Boost and Linear SVR. It is quite clear that Gradient Boost beats Linear SVR in all residual analysis experiments. However, we note that the residual analysis experiment yielding the best result is experiment B, the

proportional residual. Gradient Boost scored on Experiment B an R2 of 0.722 - 0.82, an RMSE of 0.27-0.4, and an accuracy of 71%-76% across the union of clusters and WS cluster 1.

Models	Clusters	Union of Clusters			WS Cluster 1			
	Average outcome	0.749859357	-1.08945795	-0.2501406	0.678275	-1.466979	-0.3217	
Gradient Boost	Residuals	Experiment B	Experiment C	Experiment D	Experiment B	Experiment C	Experiment D	
	Recall	0.000	0.941	0.000	0.00	0.95	0.00	
	F2	0.000	0.953	0.000	0.00	0.97	0.00	
	R2	0.722	0.715	0.722	0.82	0.73	0.82	
	RMSE	0.400	1.535	0.400	0.27	1.15	0.27	
	MAE	0.220	1.159	0.220	0.16	0.89	0.16	
	Accuracy	70.980	-149.421	-112.499	76.00	-46.15	-4.94	
	NMI	1.000	1.000	1.000	1	1	1	
	Training Time (s)	NA	NA	NA	NA	NA	NA	
	Testing Time (s)	NA	NA	NA	NA	NA	NA	
	Linear SVR	Recall	0.000	0.000	0.000	0.000	0.924	0.000
F2		0.000	0.000	0.000	0.000	0.000	0.000	
R2		-1.893	0.517	-1.893	-0.118	0.647	-0.118	
RMSE		1.002	1.424	1.002	0.679	1.430	0.679	
MAE		0.289	1.034	0.289	0.245	1.105	0.245	
Accuracy		67.368	-53.905	-56.470	15.447	-141.222	-76.301	
NMI		1.000	1.000	1.000	1.000	1.000	1.000	
Training Time (s)		NA	NA	NA	NA	NA	NA	
Testing Time (s)		NA	NA	NA	NA	NA	NA	
Stat		Recall	-	-	-	-	-	-
		F2	-	-	-	-	-	-
	R2	-	-	-	-	-	-	
	RMSE	-	-	-	-	-	-	
	MAE	-	-	-	-	-	-	
	Accuracy	-	-	-	-	-	-	
	NMI	-	-	-	-	-	-	
	Training Time (s)	-	-	-	-	-	-	
	Testing Time (s)	-	-	-	-	-	-	

Table 11.5: Residual Analysis for Point-wise Models

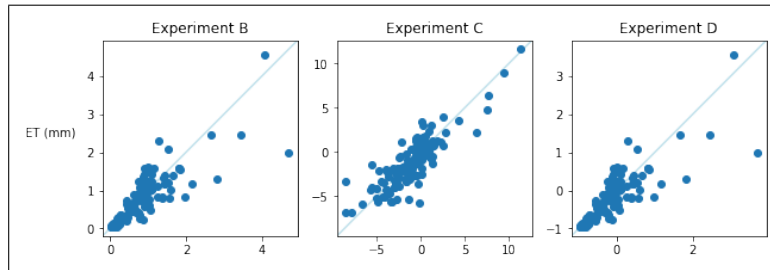


Figure 11.3: Residual Analysis Scatter Plot for Gradient Boost

Figure 11.3 illustrates the actual residual (x -axis) versus the predicted residual (y -axis) across the three residual analysis experiments (B,C,and D) upon using the Gradient Boost model. We note that there is a high resemblance between experiment B and D showing high concentration around the bottom left part of the bisector. Experiment C, however, show a more scattered distribution around the middle of the bisector. Hence, experiment B and D are the best performing according to this scatter plot. However, **Table 11.5** confirms that indeed the best residual analysis experiment is B.

Takeaway Message: Gradient Boost was the best point-wise model to predict ET. Gradient Boost beats Linear SVR in R2 by 32% to 33%, in accuracy by 10% to 12%, and in RMSE by 24% to 26%. Gradient Boost also beats Auto-sklearn in R2 by 6%, and in RMSE by 4% to 45%. Gradient Boost outperforms EEflux ET METRIC in R2 by a 100%, in accuracy by 17.144% - 24.408%, and in RMSE by 82%-100%. Nonetheless, Gradient Boost performs best in minimizing the proportional bias, beating Linear SVR by a 100% in R2, a 4%-61% in accuracy, and a 28%-60% in RMSE. Hypothesis testing was applied to validate these percentages.

We also like to highlight the effect of SMOGN on our best model Gradient Boost in Scenario C on the union of clusters. Before SMOGN, the F2 measure was a 0.92 in comparison to a 0.930, the R2 score was a 0.613 in comparison to a 0.637. The full table is found in the appendix.

11.7.2 Across Different Climates

We have previously done Experiment A on our point-wise models across the whole dataset. We now aim at conducting this experiment on our models but on each climate separately with scenario C. The types of climates and their meaning are mentioned in **Chapter 4, Table 4.1** . We portray the experimental results of scenario C in **Tables 11.6 and 11.7** where the columns in our table represent:

- Cwa: The model is being trained on data of climate Cwa.
- Dsa: The model is being trained on data of climate Dsa.
- Cfa: The model is being trained on data of climate Cfa.
- Csa: The model is being trained on data of climate Csa.
- Union: The model is being trained on the union of climates.

Tables 11.6 and 11.7 show the experimental results across different climates and across the union of climates (we note that the tables are split for visual reasons). We compare the performance of our four models: EEflux (METRIC) model , Linear SVR (base model), and Gradient Boost (best model). We note that Gradient Boost outperforms all others in predicting real ET across the union of climates and on each climate separately. We do realize that the Gradient Boost model does not perform well on the climates Cwa, Dsa, and Other which is due to the little data available for these two climates (training data of 137, 228, and 761 rows respectively). However, Gradient Boost performs in a comparable manner to the union of climates on climates Cfa and Csa, with Csa being the best in terms of giving the best error metrics and accuracy. Gradient Boost scores an R2 of 0.624, an MAE of 0.806, an accuracy of 72%, and a recall of 0.935 on climate Csa. Transfer learning is highlighted when training the model on the union of climates, where the model learned from each climate to perform better on their union.

	Metrics	Cwa		Dsa		Other	
Models	Average ET	5.199416872		2.190195719		2.717362	
EEflux	Train Size	137		228		761	
	Test Size	59		99		331	
	Data sets	Validation	Testing	Validation	Testing	Validation	Testing
	Recall	NA	0.898	NA	0.000	NA	0.000
	F2	NA	0.000	NA	0.000	NA	0.000
	R2	NA	-4016.791	NA	-3.611	NA	0.767
	RMSE	NA	4.685	NA	1.620	NA	0.314
	MAE	NA	3.848	NA	1.443	NA	0.213
	Accuracy	NA	43.771	NA	16.103	NA	68.730
	NMI	NA	1.000	NA	0.637	NA	1.000
	Training (s)	NA	NA	NA	NA	NA	NA
Testing (s)	NA	NA	NA	NA	NA	NA	
Gradient Boost	Recall	0.866 +- 0.041	0.897	0 +- 0.000	0.892	0.387 +- 0.54	0.859
	F2	0 +- 0.000	5.00E-05	0 +-0.000	0.000	0.0 +- 0.0	0.000
	R2	-0.006 +- 0.007	-0.012	0.28 +-0.072	0.250	0.146 +- 0.389	0.351
	RMSE	2.166 +- 0.293	1.983	0.917 +-0.159	1.100	0.271 +- 1.053	1.080
	MAE	1.826 +-0.310	1.687	0.7303 +-0.119	0.811	0.093 +- 0.763	0.785
	Accuracy	55.5 +-8.596	57.459	62.122 +-4.408	62.552	3.525 +- 67.251	65.975
	NMI	0 +-0.000	0.000	0.996 +-0.007	0.985	0.001 +- 0.999	0.998
	Training (s)	NA	0.042	NA	0.013	0.013	2.079
	Testing (s)	NA	0.001	NA	0.001	0.001	2.342
Linear SVR	Recall	0.931 +- 0.041	0.954	0.178 +- 0.357	0.000	0.397 +- 0.586	0.725
	F2	0.94 +- 0.028	0.962	0 +- 0.000	0.000	0.0 +- 0.0	0.000
	R2	0.335 +- 0.540	0.676	-0.368 +- 0.982	-0.682	0.393 +- -0.408	-0.009
	RMSE	1.528 +- 0.523	1.248	1.274 +- 0.320	1.464	0.271 +- 1.572	1.368
	MAE	1.238 +- 0.503	0.992	1.012 +- 0.322	1.164	0.198 +- 1.248	1.121
	Accuracy	71.29 +- 13.980	75.535	47.162 +- 24.342	50.355	18.678 +- 42.089	42.102
	NMI	1 +- 0.000	0.997	0.998 +- 0.004	0.997	0.0 +- 1.0	0.999
	Training (s)	NA	0.022	NA	0.044	-	2.079
	Testing (s)	NA	0.001	NA	0.001	-	2.342
Stat	Recall	-	-	-	-	-	-
	F2	-	-	-	-	-	-
	R2	-	-	-	-	-	-
	RMSE	-	-	-	-	-	-
	MAE	-	-	-	-	-	-
	Accuracy	-	-	-	-	-	-
	NMI	-	-	-	-	-	-
	Training (s)	-	-	-	-	-	-
Testing (s)	-	-	-	-	-	-	

Table 11.6: Experimental Variations Point-wise Models part 1

	Metrics	Cfa		Csa		Union	
Models	Average ET	4.979814		3.588537		3.879262	
EEflux	Train Size	1288		1147		3576	
	Test Size	559		494		1547	
	Data sets	Validation	Testing	Validation	Testing	Validation	Testing
	Recall	NA	0.830	NA	0.856	NA	0.880
	F2	NA	0.848	NA	0.864	NA	0.894
	R2	NA	-1.140	NA	-1.054	NA	-0.540
	RMSE	NA	3.184	NA	2.922	NA	2.479
	MAE	NA	2.571	NA	2.047	NA	1.939
	Accuracy	NA	39.694	NA	47.683	NA	47.612
	NMI	NA	0.893	NA	0.888	NA	0.892
	Training (s)	NA	NA	NA	NA	NA	NA
Testing (s)	NA	NA	NA	NA	NA	NA	
Gradient Boost	Recall	0.901 +-0.012	0.913	0.928 +-0.022	0.935	0.895 +- 0.016	0.917
	F2	0.918 +-0.011	0.926	0.944 +- 0.016	0.951	0.912 +- 0.014	0.930
	R2	0.546 +-0.081	0.578	0.586 +- 0.082	0.624	0.591 +- 0.043	0.637
	RMSE	1.762 +-0.169	1.683	1.098 +- 0.113	1.041	1.44 +- 0.112	1.359
	MAE	1.301 +-0.100	1.247	0.821 +- 0.068	0.806	1.053 +- 0.061	1.019
	Accuracy	64.616 +-4.045	64.845	71.39 +- 2.635	72.413	64.85 +- 2.638	64.756
	NMI	0.999 +-0.001	0.998	0.999 +- 0.001	0.996	0.999 +- 0.000	0.998
	Training (s)	NA	0.050	NA	0.042	NA	0.151
	Testing (s)	NA	0.002	NA	0.001	NA	0.002
	Linear SVR	Recall	0.841 +- 0.048	0.951	0.885 +- 0.059	0.846	0.862 +- 0.063
F2		0.702 +- 0.352	0.930	0.28 +- 0.427	0.000	0.637 +- 0.417	0.907
R2		0.0946 +- 0.390	-0.142	-1.083 +- 2.261	0.006	-0.101 +- 0.557	0.429
RMSE		2.393 +- 0.498	2.884	2.24 +- 1.155	1.743	2.309 +- 0.530	1.681
MAE		1.906 +- 0.434	2.417	1.924 +- 1.164	1.290	1.893 +- 0.548	1.302
Accuracy		56.345 +- 8.650	14.523	30.638 +- 2.635	66.860	34.123 +- 23.415	52.778
NMI		0.999 +- 0.001	0.998	0.998 +- 0.001	0.997	0.999 +- 0.000	0.998
Training (s)		NA	0.221	NA	0.177	NA	0.663
Testing (s)		NA	0.001	NA	0.001	NA	0.010
Stat	Recall	-	-	-	-	-	-
	F2	-	-	-	-	-	-
	R2	-	-	-	-	-	-
	RMSE	-	-	-	-	-	-
	MAE	-	-	-	-	-	-
	Accuracy	-	-	-	-	-	-
	NMI	-	-	-	-	-	-
	Training (s)	-	-	-	-	-	-
	Testing (s)	-	-	-	-	-	-

Table 11.7: Experimental Variations for Point-wise Models part 2

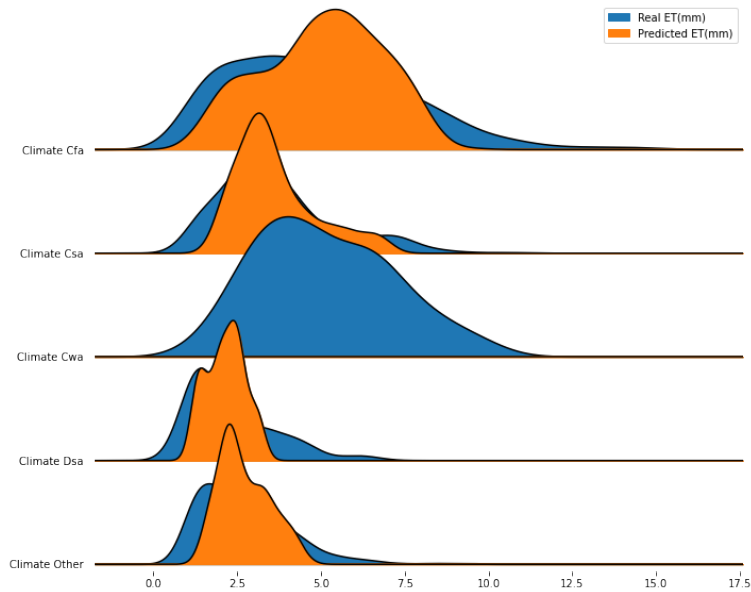


Figure 11.4: Density Plot for ET across all Climates for Point-wise Models

In **Figure** 11.4, the x -axis represents ET(mm) and the y -axis represents the density. The color blue represents the real ET(mm) and the orange color represents the predicted ET(mm). We plot the real vs the predicted ET(mm) across all the available climates (Cfa, Csa, Cwa, Dsa, and other) upon using the Gradient Boost model. We note that climate Csa, which is proven to be the best performing in **Table** 11.7, offers the best trace for real versus predicted ET (mm). Climate Cfa also show a comparable performance, unlike Dsa, Cwa, and Other.

Table 11.8 shows the residual analysis experiments across the union of climates and across the best performing climate Csa on both models Gradient Boost and Linear SVR. It is obvious that Gradient Boost beats Linear SVR in all residual analysis experiments as seen previously. We note that the residual analysis experiment yielding the best result is also experiment B in this case, the proportional residual. Gradient Boost scored an R2 of 0.722 - 0.74, an RMSE of 0.29-0.4, and an accuracy of 71%-72% across the union of climates and climate Csa.

Models	Clusters	Union of Climates			Csa		
	Average outcome	0.749859357	-1.08945795	-0.2501406	0.74643	-1.14597705	-0.253569671
Gradient Boost	Residuals	Experiment B	Experiment C	Experiment D	Experiment B	Experiment C	Experiment D
	F2	0.000	0.941	0.000	0.00	0.00	0.00
	Recall	0.000	0.953	0.000	0.00	0.00	0.00
	R2	0.722	0.715	0.722	0.74	0.78	0.74
	RMSE	0.400	1.535	0.400	0.29	1.26	0.29
	MAE	0.220	1.159	0.220	0.21	1.09	0.21
	Accuracy	70.980	-149.421	-112.499	71.64	-128.43	-78.78
	NMI	1.000	1.000	1.000	1	1	1
	Training Time (s)	NA	NA	NA	NA	NA	NA
	Testing Time (s)	NA	NA	NA	NA	NA	NA
Linear SVR	F2	0.000	0.000	0.000	0.000	0.000	0.000
	Recall	0.000	0.000	0.000	0.000	0.000	0.000
	R2	-1.893	0.517	-1.893	-107.309	-0.131	-107.309
	RMSE	1.002	1.424	1.002	4.456	2.247	4.456
	MAE	0.289	1.034	0.289	1.456	1.906	1.456
	Accuracy	67.368	-53.905	-56.470	-72.522	-373.025	-3189.497
	NMI	1.000	1.000	1.000	1.000	1.000	1.000
	Training Time (s)	NA	NA	NA	NA	NA	NA
	Testing Time (s)	NA	NA	NA	NA	NA	NA
	Stat	F2	-	-	-	-	-
Recall		-	-	-	-	-	-
R2		-	-	-	-	-	-
RMSE		-	-	-	-	-	-
MAE		-	-	-	-	-	-
Accuracy		-	-	-	-	-	-
NMI		-	-	-	-	-	-
Training Time (s)		-	-	-	-	-	-
Testing Time (s)		-	-	-	-	-	-

Table 11.8: Residual Analysis for Point-wise Models

Takeaway message: Gradient Boost performs fairly well on separate climates if the training data on each climate is sufficient. Nevertheless, Gradient Boost performed the best on climate Csa (Mediterranean - a mild climate with a dry and hot summer), which is most suitable since farmers would need to know how much to irrigate (through predicted ET values) the most in this dry and hot season, rather than the cold and moist seasons. In fact, the Gradient Boost model trained on climate Csa beat the one trained on the union of climates in accuracy by 7.648%, in MAE by 27.3%, and in recall by 2%. Furthermore, Gradient Boost on climate Csa performs better than on the union of climates in minimizing the proportional bias, scoring a 2.5% higher R2, and a 37% lower RMSE.

11.7.3 Across Different Seasons

After observing model performance on weather clusters and climates, we now aim to study model performance across different seasons. We identify three seasons: Summer, Spring, and Winter. We note that clustering by TA using kmeans, where $k = 3$ was the most successful in showing distinct seasons per clusters (more details are explained in **Section 5.4.5**). We compare the performance of our four models: EEflux (METRIC) model, Linear SVR (base model), and Gradient Boost (best model) on three clusters by TA on scenario C. We portray the experimental results of scenario C in **Table 11.9** and **Table 11.10** where the columns in our table represent:

- TA cluster 0: The model is being trained on data clustered by TA using k-means, $k = 3$ on the first cluster.
- TA cluster 1 : The model is being trained on data clustered by TA using k-means, $k = 3$ on the second cluster.
- TA cluster 2 : The model is being trained on data clustered by TA using k-means, $k = 3$ on the third cluster.
- Union : The model is being trained on the union of clusters.

Table 11.9 and **Table 11.10** show the experimental results across different seasons and across the union of seasons (the tables are split for better visuals). We compare the performance of our four models: EEflux (METRIC) model, Linear SVR (base model), and Gradient Boost (best model). We note that Gradient Boost outperforms all others in predicting real ET across the union of seasons and on each season separately. We do realize that the Gradient Boost model does not perform well in the winter season (TA cluster 0), which is due to the little data available for this season (training data of 774 row). However, Gradient Boost performs in a comparable manner to the union of seasons, with summer being the best in terms of giving the best error metrics. Gradient Boost scores an R2 of 0.578, an MAE of 1.267, an accuracy of 65%, and a recall of 0.923 in the summer season.

	Metrics	TA Cluster 0		TA Cluster 1	
Models	Average ET	2.401		5.415	
EEflux	Train Size	774.000		1063.000	
	Test Size	351.000		470.000	
	Data sets	Validation	Testing	Validation	Testing
	Recall	NA	0.000	NA	0.831
	F2	NA	0.000	NA	0.854
	R2	NA	-0.771	NA	-1.344
	RMSE	NA	1.491	NA	3.636
	MAE	NA	1.162	NA	2.807
	Accuracy	NA	49.566	NA	46.462
	NMI	NA	0.872	NA	0.872
	Training (s)	NA	NA	NA	NA
	Testing (s)	NA	NA	NA	NA
Gradient Boost	Recall	0.311 +-0.388	0.910	0.904 +- 0.013	0.923
	F2	0 +-0.000	0.000	0.92 +-0.011	0.934
	R2	0.234 +-0.064	0.274	0.509 +-0.076	0.578
	RMSE	1.048 +-0.303	0.894	1.824 +-0.149	1.677
	MAE	0.767 +-0.092	0.693	1.381 +-0.093	1.267
	Accuracy	63.504 +-2.839	66.799	65.565 +-3.399	65.016
	NMI	1 +-0.000	0.998	1 +-0.001	0.999
	Training (s)	NA	0.028	NA	0.042
	Testing (s)	NA	0.001	NA	0.001
Linear SVR	Recall	0.457 +-0.404	0.838	0.917 +- 0.039	0.785
	F2	0 +-0.000	0.000	0.899 +-0.023	0.833
	R2	-1.627 +-3.757	-2.034	-0.914 +-1.103	-0.591
	RMSE	1.66 +-0.814	1.943	3.48 +-1.048	3.245
	MAE	1.305 +-0.842	1.743	2.972 +-1.058	2.578
	Accuracy	43.662 +-47.419	-1.668	5.859 +-35.502	54.017
	NMI	1 +-0.001	1.000	0.999 +-0.001	0.998
	Training (s)	NA	0.114	NA	0.192
	Testing (s)	NA	0.003	NA	0.001
Stat	Recall	-	-	-	-
	F2	-	-	-	-
	R2	-	-	-	-
	RMSE	-	-	-	-
	MAE	-	-	-	-
	Accuracy	-	-	-	-
	NMI	-	-	-	-
	Training (s)	-	-	-	-
	Testing (s)	-	-	-	-

Table 11.9: Experimental Variations for Point-wise Models part 1

	Metrics	TA Cluster 2		Union of Clusters	
Models	Average ET	3.592		3.879262	
12*EEflux	Train Size	1705.000		3576	
	Test Size	745.000		1547	
	Data sets	Validation	Testing	Validation	Testing
	Recall	NA	0.939	NA	0.880
	F2	NA	0.925	NA	0.894
	R2	NA	-1.179	NA	-0.540
	RMSE	NA	2.175	NA	2.479
	MAE	NA	1.750	NA	1.939
	Accuracy	NA	50.331	NA	47.612
	NMI	NA	0.905	NA	0.892
	Training (s)	NA	NA	NA	NA
	Testing (s)	NA	NA	NA	NA
Gradient Boost	Recall	0.889 +-0.028	0.911	0.895 +- 0.016	0.917
	F2	0.915 +-0.022	0.925	0.912 +- 0.014	0.930
	R2	0.459 +-0.068	0.502	0.591 +- 0.043	0.637
	RMSE	1.3 +-0.073	1.224	1.44 +- 0.112	1.359
	MAE	0.949 +-0.044	0.922	1.053 +- 0.061	1.019
	Accuracy	66.925 +-2.853	67.827	64.85 +- 2.638	64.756
	NMI	0.999 +-0.001	0.997	0.999 +- 0.000	0.998
	Training (s)	NA	0.078	NA	0.151
	Testing (s)	NA	0.002	NA	0.002
Linear SVR	Recall	0.849 +- 0.023	0.803	0.862 +- 0.063	0.886
	F2	0 +-0.000	0.000	0.637 +- 0.417	0.907
	R2	-0.101 +- 0.246	-1.431	-0.101 +- 0.557	0.429
	RMSE	1.852 +- 0.211	2.720	2.309 +- 0.530	1.681
	MAE	1.447 +- 0.233	2.275	1.893 +- 0.548	1.302
	Accuracy	50.572 +- 15.901	40.364	34.123 +- 23.415	52.778
	NMI	0.999 +- 0.001	0.997	0.999 +- 0.000	0.998
	Training (s)	NA	0.289	NA	0.663
	Testing (s)	NA	0.002	NA	0.010
Stat	Recall	-	-	-	-
	F2	-	-	-	-
	R2	-	-	-	-
	RMSE	-	-	-	-
	MAE	-	-	-	-
	Accuracy	-	-	-	-
	NMI	-	-	-	-
	Training (s)	-	-	-	-
	Testing (s)	-	-	-	-

Table 11.10: Experimental Variations for Point-wise Models part 2

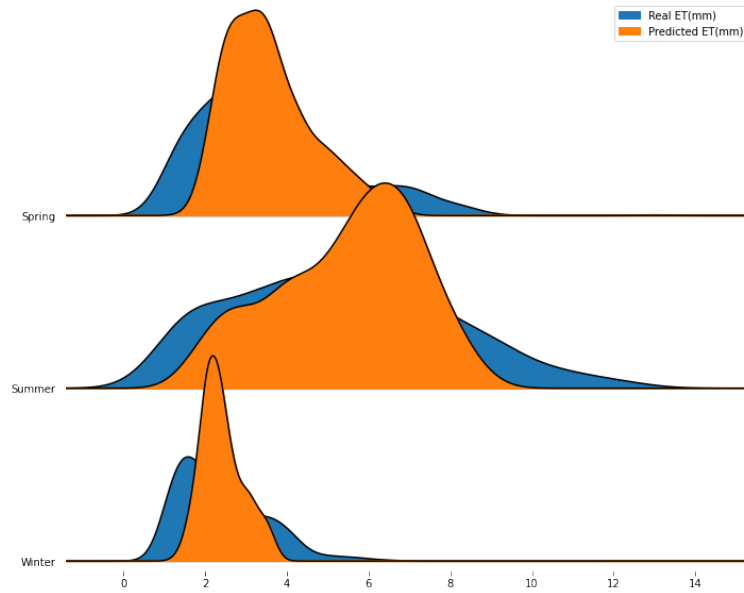


Figure 11.5: Density Plot for ET across all Seasons for Point-wise Models

In **Figure 11.5**, the x -axis represents ET(mm) and the y - axis represents the density. The color blue represents the real ET(mm) and the orange color represents the predicted ET(mm). We plot the real vs the predicted ET(mm) across all the seasons (Spring, Summer, Winter) upon using the Gradient Boost model. We note that the summer season, which is proven to be the best performing in **Table 11.10**, offers the best trace for real versus predicted ET (mm). The spring season also show a comparable performance. The model trained on the winter season, however, produces predicted ET far from the actual ET, shown in the figure through different peaks.

Table 11.11 shows the residual analysis experiments across the union of seasons and the best performing season (summer - TA cluster 1) on both models Gradient Boost and Linear SVR. Gradient Boost beats Linear SVR in all residual analysis experiments as seen previously. We note that the residual analysis experiment yielding the best result is also experiment B in this case, the proportional residual. Gradient Boost scored an R2 of 0.722 - 0.75, an RMSE of 0.32-0.4, and an accuracy of 71%-76% across the union of seasons and the summer season.

Models	Clusters	Union of Clusters			TA cluster 1		
	Average outcome	0.749859357	-1.08945795	-0.2501406	0.632584	-2.44181237	-0.367415
Gradient Boost	Residuals	Experiment B	Experiment C	Experiment D	Experiment B	Experiment C	Experiment D
	F2	0.000	0.941	0.000	0.00	0.93	0.00
	Recall	0.000	0.953	0.000	0.00	0.95	0.00
	R2	0.722	0.715	0.722	0.75	0.70	0.75
	RMSE	0.400	1.535	0.400	0.32	1.87	0.32
	MAE	0.220	1.159	0.220	0.18	1.44	0.18
	Accuracy	70.980	-149.421	-112.499	75.68	-80.83	-71.97
	NMI	1.000	1.000	1.000	1	1	1
	Training Time (s)	NA	NA	NA	NA	NA	NA
	Testing Time (s)	NA	NA	NA	NA	NA	NA
Linear SVR	F2	0.000	0.000	0.000	0.000	0.000	0.000
	Recall	0.000	0.000	0.000	0.000	0.000	0.000
	R2	-1.893	0.517	-1.893	-4.047	-0.599	-4.047
	RMSE	1.002	1.424	1.002	1.054	3.345	1.054
	MAE	0.289	1.034	0.289	0.552	2.586	0.552
	Accuracy	67.368	-53.905	-56.470	-7.279	-455.554	-1919.182
	NMI	1.000	1.000	1.000	1.000	1.000	1.000
	Training Time (s)	NA	NA	NA	NA	NA	NA
	Testing Time (s)	NA	NA	NA	NA	NA	NA
	Stat	F2	-	-	-	-	-
	Recall	-	-	-	-	-	-
	R2	-	-	-	-	-	-
	RMSE	-	-	-	-	-	-
	MAE	-	-	-	-	-	-
	Accuracy	-	-	-	-	-	-
	NMI	-	-	-	-	-	-
	Training Time (s)	-	-	-	-	-	-
	Testing Time (s)	-	-	-	-	-	-

Table 11.11: Residual Analysis for Point-wise Models

Takeaway message: Gradient Boost performs well in the spring and summer season, which is what is desirable by farmers since in these seasons ET values are a necessity to predict to know how much to irrigate and to preserve water. Unlike the winter season, in which there is rain hence the necessity of accurately predicting ET is lessened. In the summer season, Gradient Boost beat Linear SVR by a 100% in R2 score, by 23% in RMSE, and by 11.978% in accuracy. Furthermore, Gradient Boost performs best in minimizing the proportional bias on the summer season, yielding an R2 higher by 4% and accuracy higher by 5% than the union of seasons.

11.8 Models' Stability and Performance

For each of the conducted experiments, we have performed 10-folds, 10-repeats stratified cross-validation to validate the stability of our results. We will compare our models according to different criteria: the most accurate, the most precise, and the one with the best training time.

Most Accurate Model

In terms of the highest accuracy, we note that Gradient Boost is the best point-wise model with an accuracy of 65% as opposed to the Linear SVR which gave an accuracy of 53%.

Most Precise Model

In terms of the highest utility-based scores, we note that Gradient Boost is the best point-wise model with a recall of 0.917, and an F2 score of 0.930. Linear SVR, however, yielded in a recall of 0.886 and an F2 score of 0.907, which is lower than Gradient Boost.

11.8.1 The Model with the Least Training Time

When comparing training times, we do note that Gradient Boost has the least training time of 0.151 seconds as opposed to Linear SVR which has a training time of 0.663 seconds.

11.8.2 Learning Experience

We have computed validation scores, validation standard deviation scores, and evaluated our model on a shuffled testing data set for each fold. In addition to that, we have produced a learning curve which represents the mean squared error versus the number of training samples across the validation and test sets. The learning curve for our best model Gradient Boost is observed in **Figure 11.6**. We note that as the number of training samples increases, the validation and test MSE decrease until they track each other when trained on all the samples. This implies that the model is learning and not over-fitting, yielding in low bias and low variance.

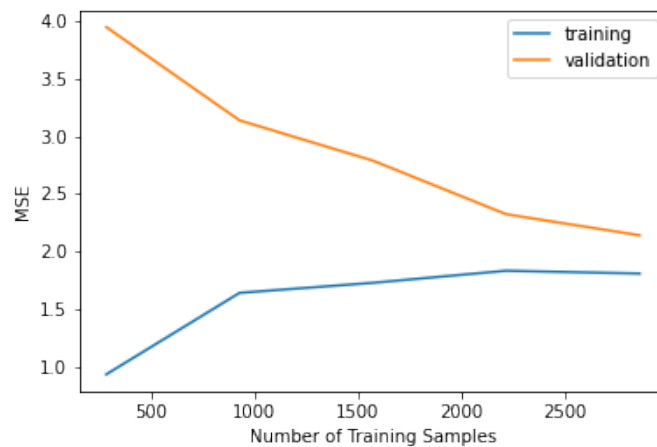


Figure 11.6: Gradient Boost Learning Curve

Chapter 12

Probabilistic Modeling

Basic regression point-wise models are designed to output point predictions on every set of input variables. However, in some business models, point estimates are not a good choice since they are prone to be erroneous. Hence, probability estimations were introduced. Probabilistic regression models output instead of a point prediction a probability distribution over the output space, which is conditional on covariates (Duan et al., 2020). Probabilistic modeling has recently become a pillar in machine learning modeling due to its ability to quantify uncertainty in our predictions, as mentioned in **Chapter 9**. Nevertheless, a machine learning model with accurate predictions and high error metrics is of no use if the model is uncertain. In this chapter, we first experimented with a well known probabilistic model - MC dropout - as our base model, which was coupled with 10 folds 10 repeated stratified cross-validation. SMOGN up-sampling was also applied, and uncertainty calculation was added. Seeking better results, we also tested on other models such as NGBoost and deep ensembles to reveal that NGBoost was the best probabilistic model.

12.1 NGBoost

NGBoost (Duan et al., 2020) offers Gradient Boosting with statistical uncertainty estimation. In matters of predictive power over standardized or tabulated inputs, Gradient Boosting techniques have traditionally been one of the top performers. NGBoost allows estimation of predictive uncertainty by probabilistic predictions with Gradient Boosting, by solving the practical challenges with the use of natural gradients of the probabilistic prediction. NGBoost is a supervised learning technique for probabilistic predictions that utilizes boosting to pick the parameters pertaining to the conditional probability distribution. This model is quite flexible and can be used with any sklearn regressor as the base learner, specified with the Base argument.

12.1.1 Architecture

NGBoost aims to predict a probabilistic distribution $P_\theta(y|x)$ of a certain parametric form and having parameters θ over each input point x . To do such predictions, we need a learning objective. In standard point prediction problems, the loss is calculated by contrasting the actual and predicted output data points. However, in the case of NGBoost, we would need a scoring rule which compares the output probability distribution to the actual test data point. The scoring rule we would be using would take as input the predicted probability distribution $P_\theta(y|x)$ and the real observation y and calculates the score (Gneiting et al., 2007). The most popular scoring rule suitable is negative log-likelihood, which is defined as:

$$L(\theta, y) = -\log P_\theta(y) \tag{12.1}$$

Where:

- θ is the parameters of the distribution
- P_θ is the predicted probability distribution
- y is the real outcome

The use of natural gradients rather than gradients in the boosting algorithm is the primary breakthrough in NGBoost. This method models a complete distribution of probability over the output space, dependent on the covariates, by following this probabilistic path. Natural gradients do not limit the movement of the parameters in the parameter space, but rather detain the output probability distribution at each input point.

The recipe for the NGBoost model is the following:

1. A base learner f
2. A probability distribution P_θ
3. A scoring rule S

A prediction of $y|x$ on a certain input value x is obtained as a form of a probability distribution of parameters θ . The latter parameters are obtained by adding the outputs of M base learners f which belong to the M stages of the generic Gradient Boosting algorithm. We also note the the parameters θ are completely dependent on the chosen probability distribution P_θ , and they are key for determining the output prediction. In order to output the predicted distribution parameter $\theta = (\mu, \log \sigma)$ for input value x , two base learners f_μ and f_σ are fit on input x and

the output of these learners are scaled with a scaling factor p and a learning rate n , such as (Duan et al., 2020):

$$P_\theta(x) = \theta_{initial} - n \sum_{m=1}^M p_m \cdot f_m(x) \quad (12.2)$$

where:

- $\theta_{initial}$ is the initial predicted parameter
- n is the learning rate
- p is a scaling factor at iteration m
- f_m is the base learner at iteration m

The mechanism of NGBoost is further explained in the flow diagram of **Figure 12.1**.

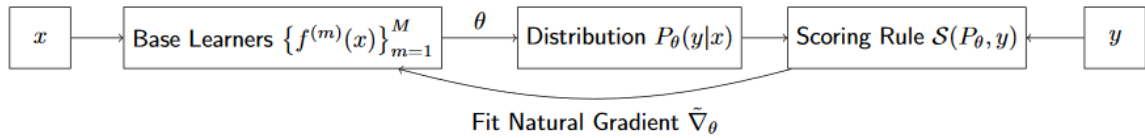


Figure 12.1: Natural Gradient Boosting (Duan et al., 2020)

12.1.2 Hyper-parameters

Our selection of hyper-parameters was initially based on the paper’s implementation in addition to our research for each hyper-parameter (what does it mean, how does it affect the model, what are the acceptable ranges). We have used a grid search to select the best hyperparameters and by referring to what is deemed acceptable in the literature. We tuned the following hyper-parameters for NGBoost:

1. Base learner: Choice of any sklearn regressor to be the base learner (decision trees, SVR)
2. Distribution: The distribution of the output variable, either exponential, normal, or log normal.
3. Score: Either NLL or CRPS
4. Number of estimators: The number of estimators you want to build before taking the maximum voting or averages of predictions.

5. Learning Rate: A tuning parameter that determines the step size at each iteration while moving toward a minimum of a loss function
6. Mini batch fraction: The fraction of training data that is split into small batches.

12.1.3 Implementation

We have utilized the implementation of NGBoost used in the paper of (Duan et al., 2020). However, we have embedded this model in a probabilistic forecasting module that implements stratified repeated K-Fold cross-validation, utility-based learning, SMOGN up-sampling, dynamic variable scaling, and dynamic model selection. This module outputs validation scores, standard deviation validation scores, test scores, uncertainty, in addition to all necessary plots, figures, and datasets needed.

To choose the parametric distribution we desire, we plotted the distribution of the output variable ET in **Figure 12.2**:

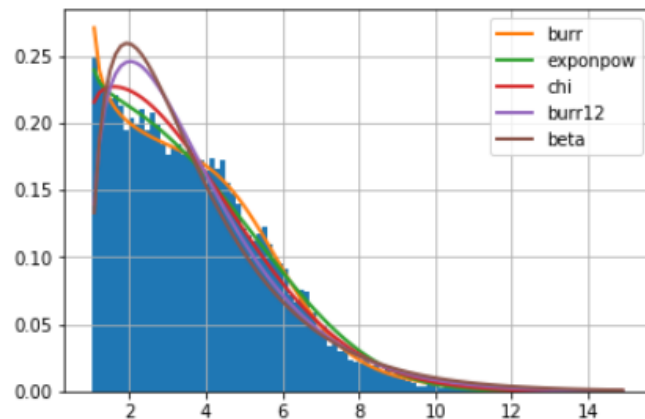


Figure 12.2: ET Distribution

We clarify that the only supported distributions in NGBoost are normal, log normal, and exponential. We do note that the distribution of ET is closest to the burr distribution, and to the exponential power distribution (which is also known as the generalized normal distribution). Hence, we chose to tune the distribution parameter according to this plot, choosing between exponential and normal distribution.

We tuned the hyper-parameters of NGBoost where we tried a set of over 300 hyper-parameters using stratified 10 repeated 10 folds cross validation and SMOGN up-sampling:

- Distribution: [Exponential, Normal]
- Number of estimators: [500, 600, 700, 800, 1000, 1200, 1300, 1500],
- Base: [DecisionTreeRegressor(criterion='friedman_mse', max_depth=2), DecisionTreeRegressor(criterion='friedman_mse', max_depth=3), DecisionTreeRegressor(criterion='friedman_mse', max_depth=4)]
- Learning rate: [0.0001, 0.001, 0.002, 0.01, 0.1, 0.2,]
- Minibatch fraction: [0.1, 0.2, 0.3, 0.4, 0.5, 0.7, 1.0]
- Scoring: NLL

The best set of hyper-parameters turned out to be:

- Distribution: [Normal]
- Number of estimators: [500]
- Base: [DecisionTreeRegressor(criterion='friedman_mse', max_depth=2)]
- Learning rate: [0.1]
- Minibatch fraction: [1]

The best UBR and SMOGN parameters are:

- rel_method=range
- extr_type=high
- coef=1.5
- rell = np.array([[1, 0 , 0],[4, 0 , 0],[15, 1 , 0]])
- relevance_pts=rell
- thr_rel=0.2
- Cperc=np.array([1,1.2])
- m=5
- repl=False
- dist=Manhattan
- p=2
- pert=0.1

12.2 MC Dropout

Monte Carlo Dropout ([Gal and Ghahramani, 2016](#)), is a smart idea that emphasizes that it is possible to view the use of normal dropout as a Bayesian approximation of a popular probabilistic model such as the Gaussian method. The MC dropout model is based on ([Gal and Ghahramani, 2016](#)). Modeling uncertainty with Monte Carlo dropout works by running multiple forward passes through the model with a different dropout mask every time.

12.2.1 Architecture

We have discussed the notion of dropout in detail in **Section 13.5.2**. Dropout corresponds to the randomly selected ignorance of a neural network’s neurons during the training process. By disregarding some neurons, hence during a specific forward or backward pass, Monte-Carlo Dropout (MC Dropout) applies dropout during predictions (not just during training). It is obvious that the latter technique lowers overall model performance, but it is supposed to make the model more uncertain when it should. For example, this is useful when the input data is far away from the data it was trained on.

12.2.2 Hyper-parameters

We tuned the following hyper-parameters for MC Dropout:

1. Number of hidden layers in the neural network
2. Number of epochs: The number of epochs is a hyper-parameter that defines the number of times that the learning algorithm will work through the entire training dataset.
3. Tau: tau value used for regularization
4. Dropout rate: ”The term “dropout” refers to dropping out units (both hidden and visible) in a neural network. Simply put, dropout refers to ignoring units (i.e. neurons) during the training phase of a certain set of neurons which is chosen at random.”
5. T: The number of predictions made for each observation

12.2.3 Implementation

We used the MC dropout model implementation by ([Gal and Ghahramani, 2016](#)). We have embedded this model in a probabilistic forecasting module that implements stratified repeated K-Fold cross-validation, utility-based learning, SMOGN

up-sampling, dynamic variable scaling, and dynamic model selection. This module outputs validation scores, standard deviation validation scores, test scores, uncertainty, in addition to all necessary plots, figures, and datasets needed.

We tuned the following set of hyperparameters according to what is deemed acceptable in the literature:

- Number of epochs: [4, 5, 10, 15, 20, 30, 50, 80, 100]
- Number of hidden layers : [4,5,6,7,8,9,10]
- Normalize: [False, True]
- Tau : [0.1, 0.15, 0.2]
- Dropout rate: [0.005, 0.01, 0.05, 0.1]
- T: [100, 1000, 1500]

The best hyper-parameter set turned out to be:

- Number of epochs: [4]
- Number of hidden layers : [4]
- Normalize: [True]
- Tau : [0.1]
- Dropout rate: [0.001]
- T: [100]

The best UBR and SMOGN parameters are:

- rel_method=range
- extr_type=high
- coef=1.5
- rell = np.array([[1, 0 , 0],[4, 0 , 0],[15, 1 , 0]])
- relevance_pts=rell
- thr_rel=0.2
- Cperc=np.array([1,1.2])
- m=5

- repl=False
- dist=Manhattan
- p=2
- pert=0.1

12.3 Deep Ensembles

12.3.1 Architecture

Deep Ensemble is a group of neural network models used to predict uncertainty. This algorithm is an implementation of ([Lakshminarayanan et al., 2017](#)). The deep learning ensemble model utilizes the entire training dataset to train each network. A random subsample of data can be used instead, however, deep neural networks typically perform better with more data. The authors found that random initialization of the parameters of the neural network, along with random shuffling of the data points, was sufficient to obtain good performance in practice.

12.3.2 Hyper-parameters

We tuned the following hyper-parameters for deep ensembles:

1. Learning rate: learning rate is a tuning parameter that determines the step size at each iteration while moving toward a minimum of a loss function
2. Number of iterations: similar to epochs
3. Batch size: The size of training data that is split into small batches

12.3.3 Implementation

We utilized the implementation of deep ensembles by ([Lakshminarayanan et al., 2017](#)). We have tuned the parameters of the model as follows:

- Learning rate: [0.0001, 0.0002, 0.002, 0.02, 0.001, 0.01, 0.1, 0.2]
- Batch size : [32, 64, 256, 512]
- Number of iterations: [100, 300, 400, 500, 600]

The best hyper-parameter combination turned out to be:

- Learning rate: [0.0001]
- Batch size : [64]
- Number of iterations: [500]

12.4 Choosing the Best Model

We have performed Experiment A on all probabilistic models which were trained on all the input features. In order to evaluate which model is the best, it is not fair to compare the error metrics (R2, Accuracy, etc..) which are calculated on the mean of the probability distribution. It is only fair to compare the probabilistic models in accordance with the negative log-likelihood, which is the loss function each of our probabilistic models utilize and found in **Equation 12.1**. The results are observed in **Table 12.1**. We would also like to note that the percentage of improvement in the following sections is calculated as:

$$\text{percentage of improvement} = \frac{\text{metric}_{\text{variable1}} - \text{metric}_{\text{variable2}}}{\text{metric}_{\text{variable1}}} \quad (12.3)$$

Where the metric is any error metric we are comparing.

	NGBoost	MC Dropout	Deep Ensemble
Probabilistic NLL	1.861220308	2.016	6.4983

Table 12.1: Best Probabilistic Model

We note that the best performing model here is NGBoost, with an NLL of 1.86, in comparison to MC Dropout and Deep Ensemble with NLL of 2.016 and 6.4983 respectively. Hence, we choose NGBoost to be our best model, where NGBoost beat MC dropout by 8.3% in NLL and Deep Ensemble by 249% in NLL.

12.5 Utility-Based Learning and SMOGN Up-sampling

We have implemented SMOGN up-sampling on the top two performing probabilistic machine learning models. We portray the results of our best NGBoost model and our base MC Dropout models. The SMOGN hyper-parameters for both models are found in **Sections 12.1.3** and **12.2.3** respectively. We show the results of both models when trained on the full dataset before and after SMOGN in **Table 12.2**.

	Metrics	Before SmoGn		After SmoGn	
Models	Average ET	3.879262		3.879262	
NGBoost	Train Size	3576		3576	
	Test Size	1547		1547	
	Data sets	Validation	Testing	Validation	Testing
	Recall	0.905 +- 0.064	0.893	0.904 +- 0.048	0.895
	F2	0.92 +- 0.052	0.906	0.708 +- 0.041	0.908
	R2	0.649 +- 0.22	0.629	0.668 +- 0.084	0.667
	RMSE	1.245 +- 0.555	1.405	1.066 +- 0.142	1.318
	MAE	0.926 +- 0.329	1.013	0.84 +- 0.109	0.940
	Accuracy	70.56 +- 5.594	65.737	71.83 +- 4.799	68.906
	NMI	0.995 +- 0.001	0.998	0.996 +- 0.001	0.998
	Training (s)	NA	31.559	NA	25.306
	Testing (s)	NA	2.189	NA	1.928
MC Dropout	Recall	0.776 +- 0.076	0.889	0.848 +- 0.015	0.891
	F2	0.629 +- 0.364	0.902	0.862 +- 0.011	0.901
	R2	0.101 +- 0.262	0.544	0.371 +- 0.079	0.597
	RMSE	1.991 +- 0.365	1.496	1.705 +- 0.176	1.430
	MAE	1.524 +- 0.273	1.133	1.301 +- 0.166	1.071
	Accuracy	49.545 +- 9.406	59.346	56.977 +- 8.646	63.552
	NMI	0.976 +- 0.022	0.978	0.984 +- 0.016	0.987
	Training (s)	-	2.374	NA	3.260
	Testing (s)	-	6.282	NA	3.785

Table 12.2: SMOGN Comparison

We observe that the majority of the error metrics showed an improvement after applying SMOGN up-sampling. In NGBoost, the Recall boosted from 0.892 to 0.93, the F2 measure boosted from 0.906 to 0.908, the R2 from 0.629 to 0.667, the RMSE decreased from 1.405 to 1.318, the MAE from 1.013 to 0.940, the accuracy increased from a 66% to a 69%. The same case applies for MC Dropout, where the Recall increased from a 0.889 to a 0.891, the R2 from a 0.544 to a 0.597, the RMSE decreased from a 1.496 to a 1.430, the MAE from a 1.133 to a 1.071, and the accuracy increased from a 59% to 64%.

Takeaway Message

SMOGN has left a good impact on our best model NGBoost, decreasing the RMSE by 6.6% and increasing the accuracy by 3%. Hence, SMOGN will be applied to both NGBoost and MC Dropout when training on any data subset or clustering because it aims at boosting the results.

12.6 Quantifying Uncertainty

We have performed four sets of experiments, each pertaining to a feature selection method, utilizing our best probabilistic model NGBoost. We have experimented with different feature selection methods as mentioned in **Table 12.3**. We demonstrate the experimental results in **Table 12.4**.

Combinations	Model	Name	Input Combinations
1	NGBoost1	Scenario A	
2	NGBoost2	Scenario B	TA(5 lags)
3	NGBoost3	Scenario C	TA(5 lags) WS(2 lags) RH(3 lags) EEflux LST(5 lags) EEflux NDVI(2 lags) EEflux Albedo(2 lags)
4	NGBoost4	Scenario D	TA(5 lags) RH(3 lags) EEflux LST(5 lags)

Table 12.3: Feature Selection Scenarios

Models	Metrics	Validation Scores	Validation Scores Std	Testing Scores
NGBoost1	Recall	0.904	0.048	0.895
	F2	0.708	0.041	0.908
	R2	0.668	0.084	0.667
	RMSE	1.066	0.142	1.318
	Accuracy	71.830	4.799	68.906
	Training Time (seconds)	-	-	25.306
	Testing Time (seconds)	-	-	1.928
NGBoost2	Recall	0.847	0.073	0.862
	F2	0.680	0.039	0.878
	R2	0.557	0.036	0.542
	RMSE	1.643	0.522	1.546
	Accuracy	60.688	8.135	61.614
	Training Time (seconds)	-	-	10.239
	Testing Time (seconds)	-	-	1.225
NGBoost3	Recall	0.893	0.062	0.880
	F2	0.909	0.057	0.897
	R2	0.646	0.083	0.641
	RMSE	1.237	0.234	1.368
	Accuracy	69.274	4.624	67.776
	Training Time (seconds)	-	-	20.336
	Testing Time (seconds)	-	-	1.663
NGBoost4	Recall	0.879	0.040	0.842
	F2	0.894	0.042	0.858
	R2	0.508	0.169	0.417
	RMSE	1.482	0.359	1.743
	Accuracy	62.415	5.718	56.110
	Training Time (seconds)	-	-	14.196
	Testing Time (seconds)	-	-	12.497

Table 12.4: Experimental Results for Probabilistic Models

When comparing different error and accuracy measures, it is noted that NGBoost1 was the best performing model across all, scoring an R2 of 0.67, Accuracy of 69% and a Recall of 0.91. Our best feature selection giving comparable scores with lower runtime is NGBoost3, scoring an R2 of 0.641, Accuracy of 67.7% and a Recall of 0.88 . We aim to quantify the epistemic uncertainty for all the feature selection methods used, and decide which model is the most certain.

Figure 12.3 represents the epistemic uncertainty for 100 data points of the testing data set. The x-axis represents the number of data points and the y-axis represents the epistemic uncertainty in mm which is of the same unit as our output variable ET. As highlighted in **Figure 12.3**, we note the following:

1. NGBoost3 is the least uncertain.
2. NGBoost1 is the second least uncertain.
3. NGBoost2 is the most uncertain.
4. NGBoost4 is the second most uncertain.

According to **Table 12.4**, NGBoost2 and NGBoost4 are the worst performing. They are also the least certain as proven in **Figure 12.3**) by having high epistemic uncertainty for both. However, NGBoost1 and NGBoost3 are the most certain because they have the lowest epistemic uncertainty (as shown by the green and blue line plots of **Figure 12.3**). We do note that NGBoost3 is the least uncertain compared to NGBoost1. This is explained by the fact that the complexity of NGBoost1 (having more input features than NGBoost3) could boost its uncertainty. Hence, NGBoost3, which offers comparable experimental results to NGBoost1, is the least uncertain.

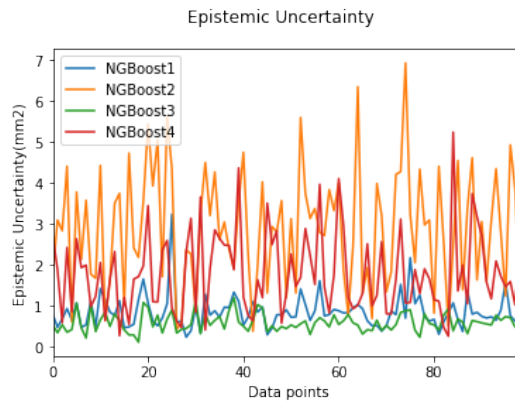


Figure 12.3: Epistemic Uncertainty for all Scenarios

Let us zoom in further to NGBoost3 in **Figure 12.4**. We do note that the epistemic uncertainty ranges between 0 and 1.2 mm^2 . We note that as the number

of data points increases, the uncertainty decreases. This is a property of the epistemic uncertainty, which is reducible and hence decreases upon the introduction of more data points.

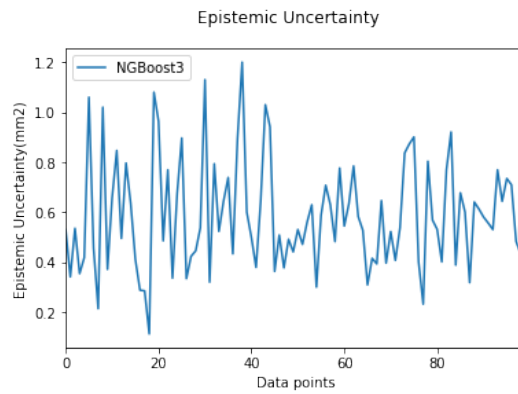


Figure 12.4: Epistemic Uncertainty for NGBBoost3

12.7 Results

12.7.1 Across Weather Clusters and Union of Clusters

Experiment A was performed on all the data (union of clusters) for NGBoost, MC Dropout, EEflux METRIC, Auto-Sklearn, and Stat. We have indeed performed this experiment on the above listed models but on all available cluster combinations identified in **Section 5.4.4** on all weather parameters utilizing k-means and dendrograms. Clustering by Wind Speed (WS) using k-means, with $k=2$, was the best performing cluster offering the best precision, recall, and least error metrics. WS cluster one has wind speed values ranging from 3.647 to 12.518 meters per second. WS cluster two has wind speed values ranging from 0.0516 to 4.129 meters per seconds.

We also carried out Experiment A on the above models and on the best clusters in all feature selection scenarios (scenario B, scenario C, and scenario D). Scenario C was the best feature selection scenario that offered the least uncertain model as specified in **Chapter 9**.

In **Table 12.5**, we present the experimental results for scenario C. In the appendix, all other experiments for scenarios (A, B, and D) are present. In our table, the columns represent:

- WS Cluster 0: The model is being trained on data clustered by WS using k-means, $k = 2$ on the first cluster.
- Union: The model is being trained on the union of clusters.
- WS Cluster 1: The model is being trained on data clustered by WS using k-means, $k = 2$ on the second cluster.

	Metrics	WS Cluster 0		Union of Clusters		WS Cluster 1	
Models	Average ET	3.745098		3.879262		3.951248	
EEflux	Train Size	1202		3576		2361	
	Test Size	530		1547		1025	
	Data sets	Validation	Testing	Validation	Testing	Validation	Testing
	Recall	NA	0.839	NA	0.880	NA	0.787
	F2	NA	0.850	NA	0.894	NA	0.000
	R2	NA	-0.489	NA	-0.540	NA	-0.577
	RMSE	NA	3.519	NA	2.479	NA	2.907
	MAE	NA	2.558	NA	1.939	NA	2.237
	Accuracy	NA	29.346	NA	47.612	NA	42.426
	NMI	NA	0.813	NA	0.892	NA	0.865
	Training (s)	NA	NA	NA	NA	NA	NA
Testing (s)	NA	NA	NA	NA	NA	NA	
NGBoost	Recall	0.765 +- 0.383	0.851	0.893 +- 0.062	0.880	0.937 +- 0.014	0.910
	F2	0.772 +- 0.386	0.876	0.909 +- 0.057	0.897	0.942 +- 0.025	0.919
	R2	0.683 +- 0.31	0.539	0.646 +- 0.083	0.641	0.624 +- 0.174	0.686
	RMSE	0.76 +- 0.148	1.586	1.237 +- 0.234	1.368	1.12 +- 0.202	1.303
	MAE	0.595 +- 0.122	1.056	0.92 +- 0.028	0.988	0.719 +- 0.156	0.981
	Accuracy	79.198 +- 5.95	63.833	69.274 +- 4.624	67.776	66.466 +- 2.122	65.965
	NMI	0.994 +- 0.002	0.996	0.996 +- 0.001	0.998	0.997 +- 0.0	0.998
	Training (s)	NA	7.982	NA	20.336	NA	14.121
	Testing (s)	NA	1.162	NA	1.663	NA	1.441
	MC Dropout	Recall	0.638 +- 0.33	0.854	0.811 +- 0.074	0.888	0.806 +- 0.065
F2		0.493 +- 0.407	0.874	0.521 +- 0.426	0.903	0.684 +- 0.343	0.913
R2		0.009 +- 0.113	0.420	0.214 +- 0.035	0.627	0.192 +- 0.324	0.612
RMSE		2.008 +- 0.652	1.681	1.83 +- 0.456	1.408	1.873 +- 0.134	1.415
MAE		1.558 +- 0.504	1.248	1.376 +- 0.314	1.018	1.438 +- 0.077	1.072
Accuracy		47.053 +- 19.199	53.448	56.617 +- 5.676	66.528	53.463 +- 11.458	63.999
NMI		0.932 +- 0.063	0.880	0.906 +- 0.082	0.944	0.982 +- 0.025	0.982
Training (s)		-	1.850	-	2.995	-	2.962
Testing (s)		-	1.946	-	3.581	-	3.518
Stat		Recall	-	-	-	-	-
	F2	-	-	-	-	-	-
	R2	-	-	-	-	-	-
	RMSE	-	-	-	-	-	-
	MAE	-	-	-	-	-	-
	Accuracy	-	-	-	-	-	-
	NMI	-	-	-	-	-	-
	Training (s)	-	-	-	-	-	-
Auto Sklearn	Recall						
	F2	NA	0.879	NA	0.906	NA	0.888
	R2	NA	-0.047	NA	0.601	NA	0.613
	RMSE	NA	1.917	NA	1.972	NA	1.401
	MAE	NA	1.373	NA	1.101	NA	0.863
	Accuracy	NA	58.152	NA	64.981	NA	71.72
	NMI	NA	0.372	NA	0.993	NA	0.794
	Training (s)	NA	1200	NA	1200	NA	1200
Testing (s)	NA	80	NA	80	NA	80	

Table 12.5: Experimental Variations for Probabilistic Models

Table 12.5 portrays the experimental results across different wind speed clusters (by Kmeans) and across the union of clusters. We contrast the performance of our four models: EEflux (METRIC) model, MC Dropout (base model), NGBoost (best model), and AutoSklearn. We note that NGBoost outperforms all others in predicting real ET across the union of clusters, producing an R2 score of 0.641, an MAE of 0.988, an accuracy of 68 %, a recall of 0.880, and an F2 measure of 0.897. When training the model on the union of clusters, transfer learning is also emphasized. All five models on WS cluster 0 performed badly, but on WS cluster 1 performed well. This means that the models trained on WS cluster 1 have learned to make more precise predictions on the union of clusters. The clusters are ordered from worst to best in terms of their results.

N.B: Auto Sklearn validation scores are not available for us to extract.

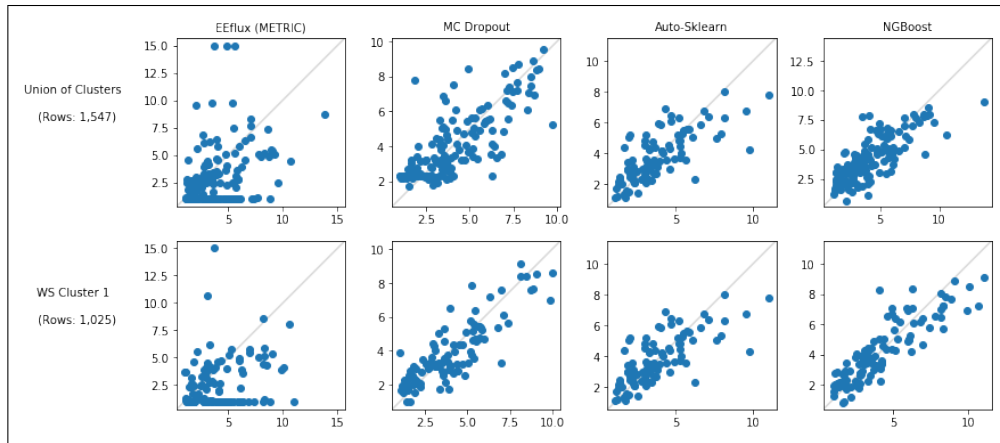


Figure 12.5: Scatter plot for Probabilistic Models

Figure 12.5 illustrates the real ET (x -axis) versus the predicted ET (y -axis) across the four models: EEFlux (METRIC), MC Dropout, AutoSklearn, and NGBoost. It is noted that the NGBoost model shows a better diagonal fit than the EEFlux (METRIC) model, the Auto-Sklearn model, and the MC Dropout model across the union of clusters and WS cluster 1. The points in the EEflux model, the MC Dropout model, and the Auto-Sklearn model were scattered and not centered around the bisector. NGBoost trained on the union of clusters yielded a better concentration around the bisector in comparison to WS cluster 1. **Figure 12.5** also confirms our quantitative observations present in **Table 12.5**.

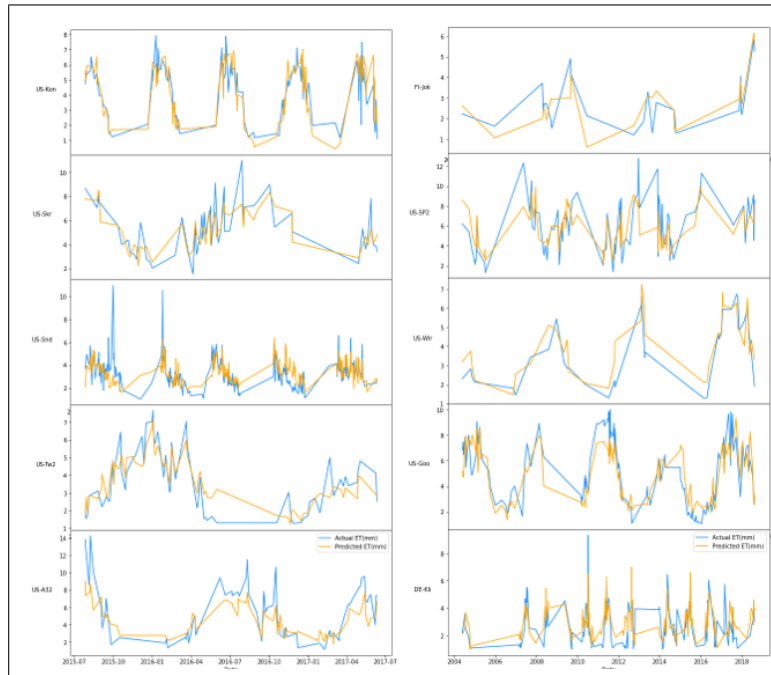


Figure 12.6: Line plot for a set of sites for NGBBoost

Figure 12.6 shows Real ET versus Predicted ET for a set of sites pertaining to our testing data upon using the NGBBoost model. The x -axis represents the date in years and the y -axis represents the Real ET versus the Predicted ET in (mm) for each site. As shown in **Figure 12.6**, Predicted ET by the NGBBoost model tracks the Real ET in an excellent manner in almost all of the years for all the chosen sites. We now zoom further into one of the sites US-Kon.

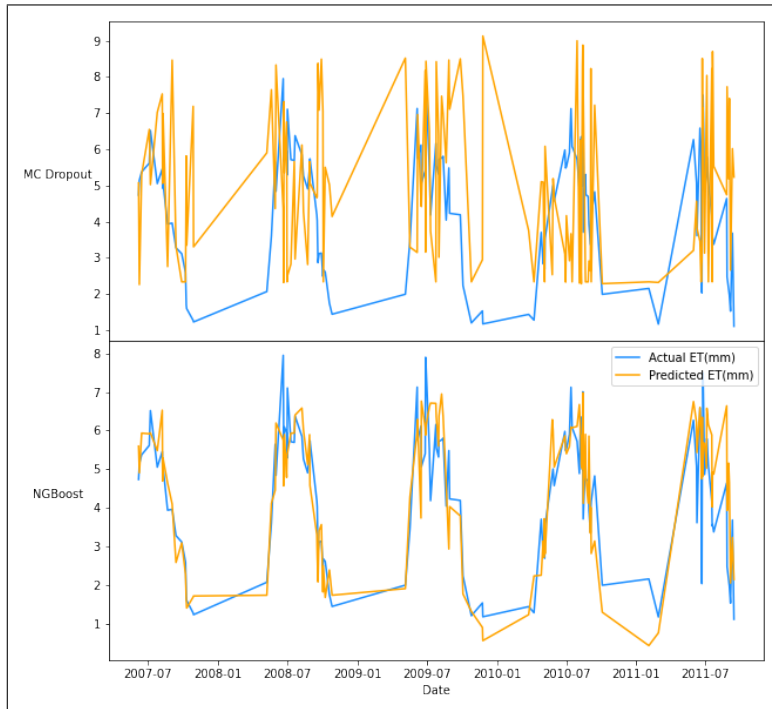


Figure 12.7: Line plot for US-Kon for Probabilistic Models

In **Figure 12.7**, we plot the real versus the predicted ET of NGBoost and MC Dropout on a sample site (US-KON) across the years. We note how the predicted ET by MC Dropout does not seem to track the real ET at all, however the predicted ET of NGBoost tracks the real ET in a good manner, successfully capturing rare values (ET values above 5mm).

Table 12.6 shows the residual analysis experiments (B,C, and D) across the union of clusters and across the best performing cluster WS cluster 1 on both models NGBoost and MC Dropout. It is noticed that NGBoost beats MC Dropout in all residual analysis experiments. However, we note that the residual analysis experiment yielding the best result is experiment B, the proportional residual. NGBoost scored on Experiment B an R2 of 0.726 - 0.738, an RMSE of 0.412-0.406, and accuracy of 70%-75% across the union of clusters and WS cluster 1.

Figure 12.8 illustrates the actual residual (x -axis) versus the predicted residual (y -axis) across the three residual analysis experiments (B,C,and D) upon using the NGBoost model. We note that there is a high resemblance between experiment B and D showing high concentration around the bottom left part of the bisector. Experiment C, however, show a more scattered distribution around the middle of the bisector. Hence, experiment B and D are the best performing

Models	Clusters	Union of Clusters			WS Cluster 1		
	Average outcome	0.8063705957	-1.160801815	-0.1936294043	0.7400011073	-1.689967037	-0.2599988927
NGBoost	Residuals	Experiment B	Experiment C	Experiment D	Experiment B	Experiment C	Experiment D
	Recall	0.000	0.937	0.000	0.000	0.964	0.000
	F2	0.000	0.948	0.000	0.000	0.967	0.000
	R2	0.738	0.794	0.738	0.726	0.855	0.726
	RMSE	0.406	1.390	0.406	0.412	1.162	0.412
	MAE	0.222	1.044	0.222	0.215	0.922	0.215
	Accuracy	70.083	-25.215	1.500	74.837	-68.813	-26.778
	NMI	1.000	1.000	1.000	1.000	1.000	1.000
	Training Time (s)	NA	NA	NA	NA	NA	NA
	Testing Time (s)	NA	NA	NA	NA	NA	NA
MC Dropout	Recall	0.000	0.927	0.000	0.000	0.996	0.000
	F2	0.000	0.929	0.000	0.000	0.993	0.000
	R2	0.633	0.751	0.633	0.778	0.808	0.778
	RMSE	0.453	1.341	0.453	0.309	1.112	0.309
	MAE	0.226	0.998	0.226	0.175	0.865	0.175
	Accuracy	71.696	-74.944	-55.068	73.816	-85.428	-60.108
	NMI	0.986	0.986	0.986	1.000	1.000	1.000
	Training Time (s)	NA	NA	NA	NA	NA	NA
	Testing Time (s)	NA	NA	NA	NA	NA	NA
	Stat	Recall	-	-	-	-	-
F2		-	-	-	-	-	-
R2		-	-	-	-	-	-
RMSE		-	-	-	-	-	-
MAE		-	-	-	-	-	-
Accuracy		-	-	-	-	-	-
NMI		-	-	-	-	-	-
Training Time (s)		-	-	-	-	-	-
Testing Time (s)		-	-	-	-	-	-

Table 12.6: Residual Analysis for Probabilistic Models

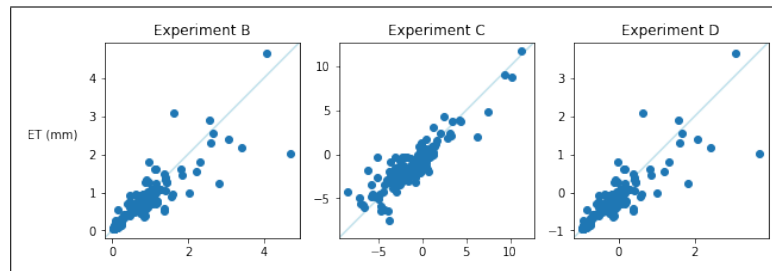


Figure 12.8: Residual Analysis Scatter Plot for NGBoost

according to this scatter plot. However, **Table 12.6** confirms that indeed the best residual analysis experiment is B.

Takeaway Message: NGBoost was the best probabilistic model to predict ET. NGBoost beats MC Dropout in R2 by 2% to 12%, in accuracy by 1% to 2%, and in RMSE by 24% to 26%. NGBoost also beats Auto-sklearn in R2 by 6%, and in RMSE by 29% to 41%. NGBoost outperforms EEflux ET METRIC in R2 by a 100%, in accuracy by 20% - 23.5%, and in RMSE by 82%-100%. Nonetheless, NGBoost performs best in minimizing the proportional bias on the union of clusters, beating MC Dropout by a 16% - in R2, an 11% in RMSE. Hypothesis testing was applied to validate these percentages.

To affirm that SMOGN was of benefit to NGBoost across Scenario C, as discussed in **Section 12.5**, we have trained NGBoost on two datasets: the original dataset and the dataset up-sampled by SMOGN. The experiments confirm our affirmation, since before SMOGN, the R2 score was 0.625 and after SMOGN the R2 score boosted to a 0.641 and the RMSE decreased from 1.395 to 1.368. Hence, SMOGN on Scenario C will be used when studying climates and seasonality. The full table is found in the appendix.

12.7.2 Across Different Climates

We have previously conducted Experiment A through the entire dataset on our point-wise models. We are now aiming at performing this experiment with scenario C on our models but on each climate separately. **Chapter 4, Table 4.1**, lists the types of climates and their significance. We show the experimental results of scenario C in **Table 12.7** and **12.8** where the columns in our table represent:

- Cwa: The model is being trained on data of climate Cwa.
- Dsa: The model is being trained on data of climate Dsa.
- Cfa: The model is being trained on data of climate Cfa.
- Csa: The model is being trained on data of climate Csa.
- Other: The model is being trained on data of undefined climates labeled as Other.
- Union: The model is being trained on the union of climates.

	Metrics	Other		Dsa		Cwa	
Models	Average ET	2.717362		2.190195719		5.199416872	
EEflux	Train Size	761		228		137	
	Test Size	331		99		59	
	Data sets	Validation	Testing	Validation	Testing	Validation	Testing
	Recall	NA	0.000	NA	0.000	NA	0.898
	F2	NA	0.000	NA	0.000	NA	0.000
	R2	NA	0.767	NA	-3.611	NA	-4016.791
	RMSE	NA	0.314	NA	1.620	NA	4.685
	MAE	NA	0.213	NA	1.443	NA	3.848
	Accuracy	NA	68.730	NA	16.103	NA	43.771
	NMI	NA	1.000	NA	0.637	NA	1.000
	Training (s)	NA	NA	NA	NA	NA	NA
Testing (s)	NA	NA	NA	NA	NA	NA	
NGBoost	Recall	0.74 +- 0.37	0.724	0.0 +- 0.0	0.000	0.979 +- 0.033	0.925
	F2	0.567 +- 0.463	0.71	0.0 +- 0.0	0.00E+00	0.978 +- 0.036	0.927
	R2	0.824 +- 0.155	0.345	0.867 +- 0.202	0.376	0.619 +- 0.032	0.606
	RMSE	0.522 +- 0.201	1.061	0.341 +- 0.299	0.968	1.287 +- 0.385	1.351
	MAE	0.401 +- 0.161	0.774	0.253 +- 0.228	0.702	1.034 +- 0.431	1.044
	Accuracy	82.614 +- 7.003	66.285	86.875 +- 11.214	67.049	92.111 +- 15.127	74.11
	NMI	0.998 +- 0.001	0.999	0.998 +- 0.002	0.995	0.999 +- 0.003	1
	Training (s)	-	8.358	NA	5.502	NA	5.623
	Testing (s)	-	1.317	NA	0.796	NA	0.72
MC Dropout	Recall	0.139 +- 0.278	0.633	0.0 +- 0.0	0.000	0.855 +- 0.033	0.874
	F2	0.0 +- 0.0	0	0.0 +- 0.0	0.000	0.357 +- 0.437	0.866
	R2	0.162 +- 0.079	0.096	-0.249 +- 0.673	-1.248	0.2 +- 0.164	0.027
	RMSE	1.112 +- 0.047	1.514	1.14 +- 0.109	1.864	1.927 +- 0.08	1.772
	MAE	0.898 +- 0.039	0.987	0.918 +- 0.061	1.161	1.54 +- 0.097	1.563
	Accuracy	59.22 +- 2.822	62.877	51.152 +- 10.633	41.098	63.528 +- 10.751	64.613
	NMI	0.981 +- 0.021	0.989	0.977 +- 0.021	0.971	0.966 +- 0.057	0.984
	Training (s)	NA	2.079	NA	0.809	NA	0.825
	Testing (s)	NA	2.342	NA	0.722	NA	0.704
Stat	Recall	-	-	-	-	-	-
	F2	-	-	-	-	-	-
	R2	-	-	-	-	-	-
	RMSE	-	-	-	-	-	-
	MAE	-	-	-	-	-	-
	Accuracy	-	-	-	-	-	-
	NMI	-	-	-	-	-	-
	Training (s)	-	-	-	-	-	-
Testing (s)	-	-	-	-	-	-	

Table 12.7: Experimental Variations for Probabilistic Models part 1

	Metrics	Cfa		Union		Csa	
Models	Average ET	4.979814		3.879262		3.588537	
EEflux	Train Size	1288		3576		1147	
	Test Size	559		1547		494	
	Data sets	Validation	Testing	Validation	Testing	Validation	Testing
	Recall	NA	0.830	NA	0.880	NA	0.856
	F2	NA	0.848	NA	0.894	NA	0.864
	R2	NA	-1.140	NA	-0.540	NA	-1.054
	RMSE	NA	3.184	NA	2.479	NA	2.922
	MAE	NA	2.571	NA	1.939	NA	2.047
	Accuracy	NA	39.694	NA	47.612	NA	47.683
	NMI	NA	0.893	NA	0.892	NA	0.888
Training (s)	NA	NA	NA	NA	NA	NA	
Testing (s)	NA	NA	NA	NA	NA	NA	
NGBoost	Recall	0.948 +- 0.013	0.905	0.893 +- 0.062	0.880	0.704 +- 0.37	0.962
	F2	0.954 +- 0.012	0.916	0.909 +- 0.057	0.897	0.716 +- 0.373	0.965
	R2	0.637 +- 0.091	0.628	0.646 +- 0.083	0.641	0.656 +- 0.103	0.681
	RMSE	0.944 +- 0.256	1.608	1.237 +- 0.234	1.368	0.723 +- 0.193	0.940
	MAE	0.948 +- 0.2	1.182	0.92 +- 0.028	0.988	0.665 +- 0.12	0.719
	Accuracy	69.782 +- 4.771	68.600	69.274 +- 4.624	67.776	76.302 +- 4.405	73.308
	NMI	0.995 +- 0.002	0.998	0.996 +- 0.001	0.998	0.995 +- 0.002	0.997
	Training (s)	NA	8.541	NA	20.336	NA	8.052
Testing (s)	NA	1.133	NA	1.663	NA	1.086	
MC Dropout	Recall	0.837 +- 0.075	0.858	0.811 +- 0.074	0.888	0.771 +- 0.122	0.804
	F2	0.849 +- 0.063	0.874	0.521 +- 0.426	0.903	0.0 +- 0.0	0.000
	R2	0.306 +- 0.116	0.285	0.214 +- 0.035	0.627	0.29 +- 0.076	0.131
	RMSE	2.136 +- 0.293	2.243	1.83 +- 0.456	1.408	1.291 +- 0.288	1.693
	MAE	1.632 +- 0.159	1.808	1.376 +- 0.314	1.018	1.021 +- 0.27	1.279
	Accuracy	55.039 +- 4.226	41.324	56.617 +- 5.676	66.528	63.258 +- 11.016	59.481
	NMI	0.974 +- 0.025	0.887	0.906 +- 0.082	0.944	0.962 +- 0.042	0.893
	Training (s)	NA	1.945	NA	2.995	NA	1.364
Testing (s)	NA	2.009	NA	3.581	NA	1.162	
Stat	Recall	-	-	-	-	-	-
	F2	-	-	-	-	-	-
	R2	-	-	-	-	-	-
	RMSE	-	-	-	-	-	-
	MAE	-	-	-	-	-	-
	Accuracy	-	-	-	-	-	-
	NMI	-	-	-	-	-	-
	Training (s)	-	-	-	-	-	-
Testing (s)	-	-	-	-	-	-	

Table 12.8: Experimental Variations for Probabilistic Models part 2

Table 12.7 and **12.8** portray the experimental results across different climates and across the union of climates (we note that the tables are split for visual reasons). We compare the performance of our four models: EEflux (METRIC) model, MC Dropout (base model), and NGBoost (best model). We note that NGBoost outperforms all others in predicting real ET across the union of climates and on each climate separately. We also note that MC Dropout performs horribly in almost all the climates. This is due to the fact that when we subset our data by climate, the dataset size decreases drastically. MC Dropout is a complex neural network model, which does not behave well on small portions of the data. We do realize that the NGBoost model does not perform as well on the climates Other and Dsa. However, NGBoost performs very well on climates Cwa, Cfa, and Csa, with climate Csa giving better results in terms of giving the best error metrics and accuracy than the union of climates. NGBoost scores an R2 of 0.641, an MAE of 0.988, an accuracy of 68%, and a recall of 0.880 on the union of climates, however, it scores an R2 of 0.681, an MAE of 0.719, an accuracy of 73.3%, and a recall of 0.962 on climate Csa. Transfer learning is also highlighted when training the model on the union of climates, where the model learned from each climate to perform better on their union.

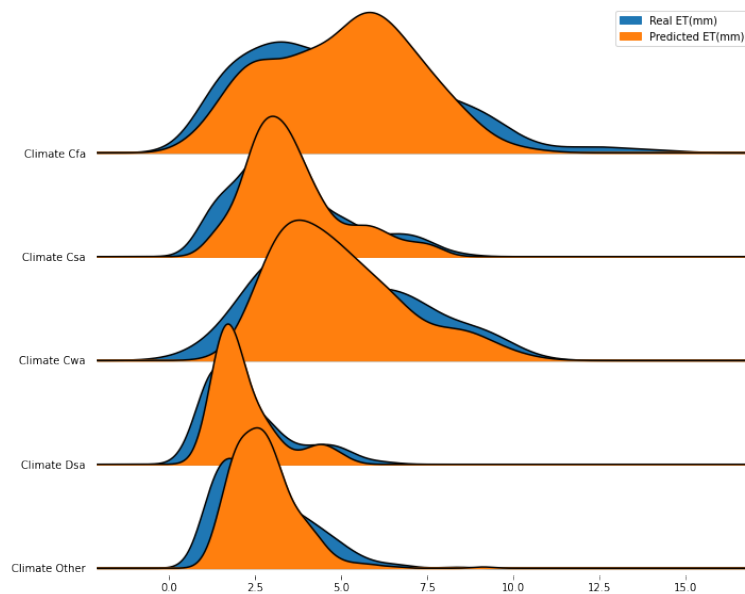


Figure 12.9: Density Plot for ET across all Climates for Probabilistic Models

In **Figure 12.9**, the x -axis represents ET(mm) and the y -axis represents the density. The color blue represents the real ET(mm) and the orange color represents the predicted ET(mm). We plot the real vs the predicted ET(mm) across all the available climates (Cfa, Csa, Cwa, Dsa, and other) upon using the NGBoost model. We note that climate Csa, which is proven to be the best performing in

Table 12.8, offers the best trace for real versus predicted ET (mm). Climate Cfa also show a comparable performance, unlike Dsa, Cwa, and Other.

	Clusters	Union of Clusters			Csa			
Models	Average outcome	0.8063705957	-1.160801815	-0.1936294043	0.646644	-1.3881749	-0.3533554409	
NGBoost	Residuals	Experiment B	Experiment C	Experiment D	Experiment B	Experiment C	Experiment D	
	F2	0.000	0.937	0.000	0.00	0.00	0.00	
	Recall	0.000	0.948	0.000	0.00	0.00	0.00	
	R2	0.738	0.794	0.738	0.73	0.74	0.73	
	RMSE	0.406	1.390	0.406	0.26	1.08	0.26	
	MAE	0.222	1.044	0.222	0.17	0.79	0.17	
	Accuracy	70.083	-25.215	1.500	76.69	-38.08	-29.50	
	NMI	1.000	1.000	1.000	1	1	1	
	Training Time (s)	NA	NA	NA	NA	NA	NA	
	Testing Time (s)	NA	NA	NA	NA	NA	NA	
	MC Dropout	F2	0.00	0.93	0.00	0.000	0.000	0.000
Recall		0.00	0.93	0.00	0.000	0.000	0.000	
R2		0.63	0.75	0.63	0.581	0.449	0.581	
RMSE		0.45	1.34	0.45	0.324	1.857	0.324	
MAE		0.23	1.00	0.23	0.227	1.458	0.227	
Accuracy		71.70	-74.94	-55.07	60.830	-94.910	-117.613	
NMI		0.985932284	0.985932284	0.985932284	0.963	0.963	0.963	
Training Time (s)		NA	NA	NA	NA	NA	NA	
Testing Time (s)		NA	NA	NA	NA	NA	NA	
Stat		F2	-	-	-	-	-	-
		Recall	-	-	-	-	-	-
	R2	-	-	-	-	-	-	
	RMSE	-	-	-	-	-	-	
	MAE	-	-	-	-	-	-	
	Accuracy	-	-	-	-	-	-	
	NMI	-	-	-	-	-	-	
	Training Time (s)	-	-	-	-	-	-	
	Testing Time (s)	-	-	-	-	-	-	

Table 12.9: Residual Analysis for Probabilistic Models

Table 12.9 shows the residual analysis experiments across the union of climates and across the best performing climate Csa on both models NGBoost and MC Dropout. It is clear that NGBoost beats MC Dropout in all residual analysis experiments as seen in the previous section. We highlight that the residual analysis experiment giving the best result is also experiment B in this case, the proportional residual. NGBoost scored an R2 of 0.73 - 0.74, an RMSE of 0.26-0.4, and an accuracy of 70%-77% across the union of climates and climate Csa.

Takeaway message: NGBoost performs very well on all climates except Dsa and Other. Nevertheless, NGBoost performed the best on Csa (Mediterranean - a moderate climate with a dry and hot summer) climate, which is most appropriate because farmers will need to know how much to irrigate the most in this dry and hot season (through expected ET values), rather than the cold and humid seasons. The NGBoost model trained on climate Csa beat the one trained on the union of climates in accuracy by 5.532%, in MAE by 19.3%, and in recall by 9.3%. Furthermore, NGBoost on climate Csa performs better than on the union of climates in minimizing the proportional bias, scoring a 7% higher accuracy, and a 56% lower RMSE.

12.7.3 Across Different Seasons

We now plan to examine model performance across various seasons after studying model performance on weather clusters and climates. Three seasons are identified:

summer, spring, and winter. We note that TA clustering using kmeans, where $k = 3$ was the most efficient way to display different seasons per cluster (more details are explained in **Section 5.4.5**). We show the experimental results of scenario C in **Table 12.10** and **12.11** where the columns in our table represent:

- TA cluster 0: The model is being trained on data clustered by TA using k-means, $k = 3$ on the first cluster.
- TA cluster 1: The model is being trained on data clustered by TA using k-means, $k = 3$ on the second cluster.
- TA cluster 2: The model is being trained on data clustered by TA using k-means, $k = 3$ on the third cluster.
- Union: The model is being trained on the union of clusters.

Table 12.10 and **12.11** show the experimental results across different seasons and across the union of seasons (the tables are split for better visuals). We compare the performance of our four models: EEflux (METRIC) model, MC Dropout (base model), and NGBoost (best model). We note that NGBoost outperforms all others in predicting real ET across the union of seasons and on each season separately. We note that the Gradient Boost model and the MC dropout model does not perform well in the winter season (TA cluster 0), which is due to the little data available for this season (training data of 774 row). However, NGBoost performs in a comparable manner to the union of seasons, with summer being the best in terms of giving the best error metrics. NGBoost scores an R2 of 0.516, an MAE of 1.355, an accuracy of 65.5%, and a recall of 0.9 in the summer season.

	Metrics	TA Cluster 0		TA Cluster 1	
Models	Average ET	2.401		5.415	
EEflux	Train Size	774.000		1063.000	
	Test Size	351.000		470.000	
	Data sets	Validation	Testing	Validation	Testing
	Recall	NA	0.000		0.831
	F2	NA	0.000	NA	0.854
	R2	NA	-0.771	NA	-1.344
	RMSE	NA	1.491	NA	3.636
	MAE	NA	1.162	NA	2.807
	Accuracy	NA	49.566	NA	46.462
	NMI	NA	0.872	NA	0.872
	Training (s)	NA	NA	NA	NA
	Testing (s)	NA	NA	NA	NA
NGBoost	Recall	0.185 +- 0.369	0.000	0.956 +- 0.019	0.900
	F2	0.188 +- 0.375	0.000	0.955 +- 0.029	0.907
	R2	0.56 +- 0.534	0.248	0.527 +- 0.072	0.516
	RMSE	0.591 +- 0.265	0.977	1.145 +- 0.676	1.793
	MAE	0.455 +- 0.19	0.743	1.418 +- 0.056	1.355
	Accuracy	78.523 +- 7.587	66.107	68.381 +- 2.378	65.551
	NMI	0.997 +- 0.002	0.999	0.996 +- 0.002	0.999
	Training (s)	NA	6.752	NA	7.756
	Testing (s)	NA	0.996	NA	1.014
MC Dropout	Recall	0.23 +- 0.285	0.000	0.81 +- 0.052	0.867
	F2	0.0 +- 0.0	0.000	0.656 +- 0.332	0.883
	R2	0.016 +- 0.031	0.041	-0.037 +- 0.235	0.468
	RMSE	1.2 +- 0.138	1.045	2.492 +- 0.407	1.964
	MAE	0.895 +- 0.05	0.837	1.981 +- 0.358	1.502
	Accuracy	56.443 +- 3.375	59.448	48.287 +- 16.166	61.171
	NMI	0.937 +- 0.059	0.821	0.915 +- 0.042	0.991
	Training (s)	NA	1.231	NA	1.319
	Testing (s)	NA	0.911	NA	1.095
Stat	Recall	-	-	-	-
	F2	-	-	-	-
	R2	-	-	-	-
	RMSE	-	-	-	-
	MAE	-	-	-	-
	Accuracy	-	-	-	-
	NMI	-	-	-	-
	Training (s)	-	-	-	-
	Testing (s)	-	-	-	-

Table 12.10: Experimental Variations for Probabilistic Models part 1

	Metrics	TA Cluster 2		Union of Clusters	
Models	Average ET	3.592		3.879262	
EEflux	Train Size	1705.000		3576	
	Test Size	745.000		1547	
	Data sets	Validation	Testing	Validation	Testing
	Recall	NA	0.939		0.880
	F2	NA	0.925	NA	0.894
	R2	NA	-1.179	NA	-0.540
	RMSE	NA	2.175	NA	2.479
	MAE	NA	1.750	NA	1.939
	Accuracy	NA	50.331	NA	47.612
	NMI	NA	0.905	NA	0.892
	Training (s)	NA	NA	NA	NA
	Testing (s)	NA	NA	NA	NA
NGBoost	Recall	0.862 +- 0.113	0.877	0.893 +- 0.062	0.880
	F2	0.575 +- 0.47	0.885	0.909 +- 0.057	0.897
	R2	0.642 +- 0.022	0.502	0.646 +- 0.083	0.641
	RMSE	1.434 +- 0.032	1.233	1.237 +- 0.234	1.368
	MAE	0.711 +- 0.252	0.918	0.92 +- 0.028	0.988
	Accuracy	66.819 +- 4.208	67.176	69.274 +- 4.624	67.776
	NMI	0.994 +- 0.001	0.997	0.996 +- 0.001	0.998
	Training (s)	NA	10.360	NA	20.336
	Testing (s)	NA	1.192	NA	1.663
MC Dropout	Recall	0.808 +- 0.077	0.896	0.811 +- 0.074	0.888
	F2	0.471 +- 0.388	0.904	0.521 +- 0.426	0.903
	R2	0.062 +- 0.198	0.450	0.214 +- 0.035	0.627
	RMSE	1.581 +- 0.414	1.309	1.83 +- 0.456	1.408
	MAE	1.214 +- 0.295	0.996	1.376 +- 0.314	1.018
	Accuracy	59.983 +- 7.569	62.405	56.617 +- 5.676	66.528
	NMI	0.957 +- 0.026	0.995	0.906 +- 0.082	0.944
	Training (s)	NA	1.655	NA	2.995
	Testing (s)	NA	2.029	NA	3.581
Stat	Recall	-	-	-	-
	F2	-	-	-	-
	R2	-	-	-	-
	RMSE	-	-	-	-
	MAE	-	-	-	-
	Accuracy	-	-	-	-
	NMI	-	-	-	-
	Training (s)	-	-	-	-
	Testing (s)	-	-	-	-

Table 12.11: Experimental Variations for Probabilistic Models part 2

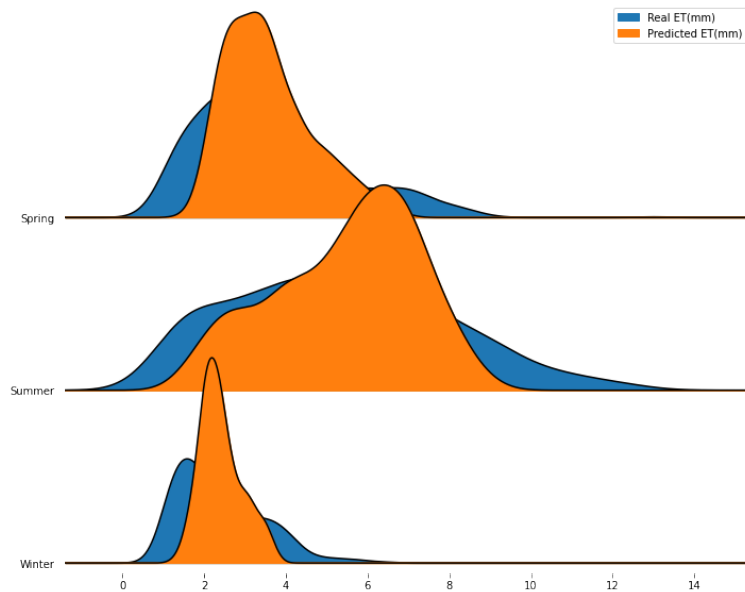


Figure 12.10: Density Plot for ET across all Seasons for Probabilistic Models

In **Figure 12.10**, the x -axis represents ET(mm) and the y - axis represents the density. The color blue represents the real ET(mm) and the orange color represents the predicted ET(mm). We plot the real vs the predicted ET(mm) across all the seasons (Spring, Summer, Winter) upon using the NGBoost model. We note that the summer season, which is proven to be the best performing in **Table 12.11**, offers the best trace for real versus predicted ET (mm). The spring season also show a comparable performance. The model trained on the winter season, however, produces predicted ET fat from the actual ET, shown in the figure through varying peaks.

Models	Clusters	Union of Clusters			TA cluster 1			
	Average outcome	0.806	-1.161	-0.194	0.598	-1.853	-0.402	
NGBoost	Residuals	Experiment B	Experiment C	Experiment D	Experiment B	Experiment C	Experiment D	
	F2	0.000	0.937	0.000	0.00	0.94	0.00	
	Recall	0.000	0.948	0.000	0.00	0.96	0.00	
	R2	0.738	0.794	0.738	0.79	0.56	0.79	
	RMSE	0.406	1.390	0.406	0.28	1.98	0.28	
	MAE	0.222	1.044	0.222	0.17	1.44	0.17	
	Accuracy	70.083	-25.215	1.500	71.64	-104.18	-78.69	
	NMI	1.000	1.000	1.000	1	1	1	
	Training Time (s)	NA	NA	NA	NA	NA	NA	
	Testing Time (s)	NA	NA	NA	NA	NA	NA	
	MC Dropout	F2	0.00	0.93	0.00	0.000	0.977	0.000
Recall		0.00	0.93	0.00	0.000	0.000	0.000	
R2		0.63	0.75	0.63	0.788	0.589	0.788	
RMSE		0.45	1.34	0.45	0.288	2.060	0.288	
MAE		0.23	1.00	0.23	0.191	1.527	0.191	
Accuracy		71.70	-74.94	-55.07	71.818	-166.096	-70.723	
NMI		0.986	0.986	0.986	1.000	1.000	1.000	
Training Time (s)		NA	NA	NA	NA	NA	NA	
Testing Time (s)		NA	NA	NA	NA	NA	NA	
Stat		F2	-	-	-	-	-	-
		Recall	-	-	-	-	-	-
	R2	-	-	-	-	-	-	
	RMSE	-	-	-	-	-	-	
	MAE	-	-	-	-	-	-	
	Accuracy	-	-	-	-	-	-	
	NMI	-	-	-	-	-	-	
	Training Time (s)	-	-	-	-	-	-	
	Testing Time (s)	-	-	-	-	-	-	

Table 12.12: Residual Analysis for Probabilistic Models

Table 12.12 shows the residual analysis experiments across the union of seasons and the best performing season (summer - TA cluster 1) on both models NGBoost and MC Dropout. NGBoost beats MC Dropout in all residual analysis experiments as observed before. We note that the residual analysis experiment yielding the best result is the proportional residual (experiment B) as well. NGBoost scored an R2 of 0.738 - 0.79, an RMSE of 0.28-0.406, and an accuracy of 70%-72% across the union of seasons and the summer season.

Takeaway message: In the spring and summer seasons, NGBoost performs reasonably well, which is what farmers need, because ET values are a requirement to predict how much to irrigate and conserve water in these seasons. In comparison to the winter season, in which rain is present, the need to accurately forecast ET is therefore reduced. In the summer season, NGBoost beat MC Dropout by 10% in R2 score, 13% in RMSE, and by 11% in MAE. Furthermore, Gradient Boost performs best in minimizing the proportional bias on the summer season, yielding a lower RMSE by 3%, and by 12% in MAE.

12.8 Models' Stability and Performance

For each of the conducted experiments, we have performed 10-folds, 10-repeats stratified cross-validation to validate the stability of our results. We will compare our models according to different criteria: the most accurate, the most precise, the most certain, and the one with the best training time.

Most Accurate Model

In terms of the highest accuracy, we note that NGBoost is the best probabilistic model with an accuracy of 68% as opposed to the MC Dropout which gave an accuracy of 66%.

Most Precise Model

In terms of the highest utility-based scores, we note that NGBoost and MC Dropout show a tie, in which both models produce a recall of 0.88. Hence, both can accurately predict rare ET values.

12.8.1 The Model with the Least Training Time

When comparing training times, we do note that NGBoost has a training time of 20 seconds, which is higher than MC Dropout (a training time of 3 seconds). This is explained by the fact that MC Dropout is TensorFlow-based, which is proven to be faster than other libraries used in NGBoost like scikit-learn.

12.8.2 Learning Experience

We have computed validation scores, validation standard deviation scores, and evaluated our model on a shuffled testing data set for each fold. We compare the validation and testing scores of NGBoost:

- NGBoost yields a validation accuracy of 71% and a test accuracy of 68%.
- NGBoost yields a validation RMSE of 1.23 and a test MSE of 1.36

We now do the same for MC Dropout:

- MC Dropout yields a validation accuracy of 57% and a test accuracy of 66%.
- MC Dropout yields a validation RMSE of 1.8 and a test MSE of 1.4

We observe that the testing and validation scores of both models track each other, hence they do not overfit, showing low bias and low variance.

Chapter 13

Conformal Quantile Modeling

13.1 Overview

There exist several models for producing prediction intervals, some of which are quantile regression, conformal regression, and conformal quantile regression which will be discussed in deep detail in the following sections.

13.1.1 Quantile Regression

When studying a regression problem, a regression model is designed to output a numerical prediction to a problem, and the confidence of this numerical prediction should be portrayed. (Meinshausen, 2006) further explained that quantile regression quantifies the conditional median of the target rather than using the least-squares method to quantify the conditional mean. We note that the conditional distribution function is expressed as follows:

$$F(y|X = x) := P(Y \leq y|X = x) \quad (13.1)$$

However, in quantile regression, the quantile - which is the percentage - can be calculated as well. Quantile regression works in a different way than other regression models. Any chosen algorithm for quantile regression will be fit on the specified quantiles (for example 5% and 95%) to estimate upper and lower bounds for the interval. For the instance in which the data is heteroscedastic (heteroscedasticity occurs when the standard deviation of a specific predicted variable, tracked over specific values of input variables, is not constant), quantile regression showed to perform very well. We also note that the quantile function dependent on α is :

$$q_\alpha(x) := \inf(y \in \mathfrak{R} : F(y|X = x) \geq \alpha) \quad (13.2)$$

The upper and lower quantiles are fixed at $\alpha_{low} (\frac{\alpha}{2})$ and $\alpha_{high} (\frac{1-\alpha}{2})$. We then fit the quantile function q on both of the latter α giving the lower and upper bound

$q_{\alpha_{low}}$ and $q_{\alpha_{high}}$ respectively.

We obtain a conditional prediction interval on Y given $X = x$ with a miscoverage rate α :

$$C(x) = [q_{\alpha_{low}}, q_{\alpha_{high}}] \quad (13.3)$$

A miscoverage rate α is specified by the user, it signifies the margin of error or miscoverage the model is allowed to have when generating the prediction interval. For example, a miscoverage rate of 10% signifies that across the whole data, 90% of the target variable values fall in their corresponding prediction interval. Hence, if I have 10 real target variable instances, the CQR model will produce 10 prediction intervals each for a single target variable. If I have 90% coverage, this will imply that there is a guarantee that 9 of these target variable values will be included in their corresponding prediction interval. Note that, depending on the value of X , the length of the interval $C(X)$ will vary greatly. In the length of the interval, the uncertainty in the estimation of Y is naturally expressed. We can not recognize this ideal prediction interval in practice, but we can try to approximate it from the given data.

13.1.2 Conformal Prediction

(Romano et al., 2019) explains that to obtain a coverage guarantee that is distribution-free and nonasymptotic, one must use conformal predictions. The key concept is to fit a regression model on the training samples, and after that utilize the residuals on a validation set which was extracted before the training, to quantify the uncertainty in certain future predictions. Present and applied conformal methods produce conformal intervals of mostly fixed lengths, this is due to the fact of using the residuals when calculating the prediction interval, rather than fitting two quantile functions like quantile regression (Vovk et al., 2019). The recipe for any conformal method is to have a predictive model, a calibration set (on which no training occurs), and a nonconformity measure.

The algorithm behind the split conformal method first splits the data frame into a training (I_1) and a calibration (I_2) set. After that, the regression model of choice (Z) will fit the training data

$$\mu(x) = Z(X_i, Y_i) \quad i \in I_1 \quad (13.4)$$

Then, the residual are calculated on the calibration set (I_2):

$$R_i = |Y_i - \mu(X_i)| \quad i \in I_2 \quad (13.5)$$

The quantile of the empirical distribution of the absolute residuals for a given level α is then calculated:

$$Q := (1 - \alpha)(1 + 1/|I_2|)th \text{ empirical quantile of } R_i \quad i \in I_2 \quad (13.6)$$

Hence, the final prediction at a new point X is obtained as follows:

$$C(X) = [\mu(X) - Q, \mu(X) + Q] \quad (13.7)$$

13.1.3 Conformalized Quantile Regression

Conformalized Quantile Regression (CQR) is a method for obtaining prediction intervals that yield valid coverage in finite samples. As mentioned in (Romano et al., 2019), a prediction interval, which consists of a lower and upper bound for the predicted variable, attains a higher probability of inclusion. (Romano et al., 2019), combine conformal prediction with classical quantile regression, benefiting from the advantages of both, yielding a model fully adaptable to heteroscedasticity. Prediction intervals should provide coverage without distributional assumptions and be short in length. CQR works by first splitting the data into train (I_1) and calibration (I_2) sets. We then use the quantile regressor Z to fit on our training data.

$$q_{\alpha_{low}}, q_{\alpha_{high}} = Z(X_i, Y_i) \quad i \in I_1 \quad (13.8)$$

CQR next computes the conformity scores which represents the error made by the prediction interval $q_{\alpha_{low}}, q_{\alpha_{high}}$. The error is quantified as follows:

$$Error_i := \max(q(x_i) - Y_i, Y_i - q(x_i)) \quad i \in I_2 \quad (13.9)$$

The prediction interval at a point X is then computed as follows:

$$C(X) = [q_{\alpha_{low}} - Q, q_{\alpha_{high}} + Q] \quad (13.10)$$

where Q is:

$$Q := (1 - \alpha)(1 + 1/|I_2|)^{th} \text{ empirical quantile of } Error_i \quad i \in I_2 \quad (13.11)$$

The CQR algorithm is summarized as follows:

Algorithm 1 Conformalized Quantile Regression

Input:Data (X_i, Y_i) Miscoverage rate α Quantile Regressor Z **Process:**Split data into training (I_1) and calibration (I_2) setsFit two quantile functions $q_{\alpha low}, q_{\alpha high}$ Compute $Error_i$ Compute Q **Output:**Compute prediction interval $C(x)$

13.2 Models

We have tested the following models:

1. **Split Conformal Random Forests:** The conventional Split conformal method described in **Section 13.1.2** using the Random Forest Regressor as the base learner.
2. **Local Conformal Random Forests:** A variant of the locally weighted split conformal technique built to create adaptive intervals. As in the typical conformal split, this approach fits the random forest regressor to the conditional mean regression function to the correct training set. After that, the locally weighted model fits another random forest regressor to the residuals of the training set using a MAD estimator.
3. **CQR Random Forests:** The Conformal Quantile Regression technique explained in **Section 13.1.3** using Random Forests as the Quantile Regressor.
4. **CQR Neural Networks:** The Conformal Quantile Regression technique explained in **Section 13.1.3** using a deep Neural Network as the Quantile Regressor.

13.3 Base Model

Our base model is the conventional Split Conformal Regression using the Random Forest Regressor.

13.3.1 Architecture

We implemented the split conformal method. Specifically, we used the proper training and calibration subsets. We train a random forest regressor on the first set, and calibrated the intervals on the second set by computing the absolute residual error, as mentioned in **Section 13.1.2** on Conformal Prediction. We note that the random forests regression estimates the conditional mean of Y_i given $X_i = x$. The Random Forest Regressor used has 2,000 estimators, a minimum sample leaf of 1, and a maximum number of features being the square root of the total feature. We split the data into training and testing according to our special split, with 70% training and 20% testing.

13.3.2 Hyper-parameters

We tuned a set of hyper-parameters, either of the base learner itself or to the conformal prediction.

Conformal Regression Hyper-parameters

- α : Miscoverage rate which ranges between 0 and 1
- `test_ratio`: Ratio of testing data
- `base_learner`: Random Forests (no other because it is the only supported shallow model in the quantile regression module).

Base Regressor Hyper-parameters

- `N_estimators`: Represents the number of trees used in the model
- `Min Sample Leaf`: Represents the minimum samples required in a leaf.
- `Max features`: Represents the number of features considered when searching for the best split.
- `Max depth`: Represents the maximum depth of a tree.

13.4 Best Model

Our best scoring model giving the lowest average prediction interval is the conformalized quantile regression using deep neural networks as a base quantile learner.

13.4.1 Architecture

The CQR Neural Network model follows the CQR architecture in **Section 13.1.3**. We used a neural network as the underlying quantile regression method instead of random forests to explore both available architectures in the CQR code module. The neural network used is a PyTorch deep sequential neural network model. In PyTorch, building a model in a sequential manner is the most convenient since it allows the user to build the model layer by layer, including several options for activation functions or dropout. There is an option that allows using the default neural network model in PyTorch, which is not sequential and has a simple architecture (resembling a perceptron), however, it is a base model and is weak, not fit to our problem. Hence, we chose to customize the model using a sequential architecture. A deep neural network is capable of solving complex problems along with producing accurate predictions, identifying as more successful than the traditional artificial neural network. Deep neural networks are more powerful because of their deep architecture consisting of several layers. Deep neural networks have witnessed many alterations across the literature like including dropout methods and performing different gradient calculations. However, the base idea behind the deep neural network is prominent, which consists of utilizing back-propagation and performing stochastic gradient descent to calculate the gradients in a recursive manner minimizing the loss function by tuning the network's weights. The CQR deep neural network of choice consists of the following layers:

- The input layer
- $\text{CELU}()$ = the activation function
- $\text{Dropout}()$ = dropout layer
- A hidden layer
- $\text{CELU}()$ =the activation function
- $\text{Dropout}()$ = dropout layer
- A hidden layer
- $\text{CELU}()$ = the activation function
- $\text{Dropout}()$ = dropout layer
- The output layer

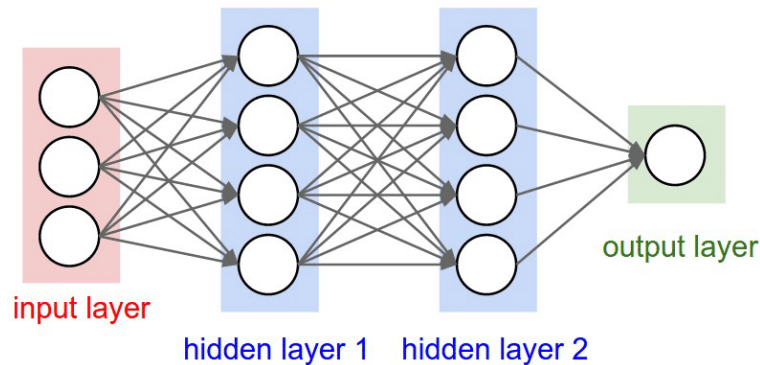


Figure 13.1: Neural Network Architecture (Musiol, 2016)

The architecture is illustrated in figure 13.1:

All layers, dropouts, and activation functions are explained thoroughly in **Section 13.5**.

13.4.2 Hyper-parameters

We tuned a set of hyper-parameters, either of the deep neural network itself or to the CQR model. The CQR hyperparameters differ from the Conformal Random Forests by having the `cv_qforest` parameter, and the `quantiles` parameter.

CQR Hyper-parameters

- `cv_qforest`: True or False - whether to use cross-validation to tune the quantile levels
- α : Miscoverage rate which ranges between 0 and 1
- `quantiles`: The desired quantile levels. A range of 2 values between 0 and 1.
- `test_ratio`: Ratio of testing data
- `base_learner`: Sequential Neural Network

Neural Network Hyper-parameters

- `epochs`: The number of iterations the training data set is being shown to the neural network model while training
- `Lr`: Learning rate, which is the step size at each iteration.
- `batch_size`: A mini-batch size is the number of samples we give to our model after the parameter update occurs.

- `hidden_size`: The size of the hidden layer
- `dropout`: A regularization technique that randomly drops neurons so that their weights will not be updated. This improves generalization and decreases overfitting.
- `weight_decay`: after each update, the weights are multiplied by a weight decay which is slightly less than 1
- `optimizer`: Type of optimizer used
- `hidden_layer_number`: number of hidden layers

13.5 Implementation

We implemented both algorithms (CQR Neural Network and Split Conformal Random Forest) using the Github code of ([Romano et al., 2019](#)) but with slight alterations for each model, discussed in details in **Sections** 13.5.1 and 13.5.2.

13.5.1 Split Conformal Random Forests

This module was implemented by ([Romano et al., 2019](#)) for handling single column inputs. However, our input data is multi-columned, hence, we altered the code to fit our problem. We also applied Grid Search to find the optimal hyper-parameters for our random forest regressor:

- Number of estimators: [600, 700,800, 900, 1000,1500, 2000],
- Max depth: [5, 10, 15, 20, 25, 30, 35, 40, 45, 50,None],
- Max features: [x_train.shape[1], auto, sqrt, log2],
- Min samples leaf: [1, 2, 3,4,5]

The best hyper-parameter combination giving the least average prediction interval length and the most coverage is:

- Number of estimators: [20]
- Max depth: [10]
- Max features: [sqrt]
- Min samples leaf: [1]

In addition, we applied grid search to find the optimal Split Conformal hyper-parameters. The best hyper-parameter combination giving the least average prediction interval length and the most coverage is:

- α : [0.17, 0.22, 0.25]
- test_ratio: 0.3
- base_learner: Random Forests

13.5.2 CQR Neural Networks

This module was implemented by (Romano et al., 2019) with a vanilla Neural Network model. We altered the code in which we applied several better deep learning techniques from the book of Jason Brown Lee [Better Deep Learning](#):

Enhanced learning through an understanding of optimization

The neural network model utilizes examples to understand how to assign particular sets of input variables to output variables. This should be achieved in a way that this mapping fits very well for the training dataset, but it works well enough on training instances not seen by the model throughout the training. This ability to perform well on particular data and new data is referred to as model generalization.

To minimize losses, optimizers are algorithms or methods used to adjust the properties of your neural network, such as weights and learning rate. We tune a set of five optimizers and chose the best performing:

1. Adagrad: Adagrad is a gradient-based optimization technique that adjusts the learning rate to the parameters, executing minor updates for frequently occurring parameters, and major updates for infrequently occurring feature-related parameters.
2. Adam: Another approach that computes adaptive learning rates for each parameter is Adaptive Moment Estimation (Adam). An exponentially decaying average of past square gradients is processed.
3. AdamW: Similar to Adam but fixes the weight decay.
4. RMSProp: RMSProp has been established arising from the need to address Adagrad's radically declining learning rates. The learning rate is also split by an exponentially decaying squared gradient average by RMSprop.

5. SGD: For each training example x_i and label y_i , Stochastic gradient descent performs an update of parameters. SGD conducts regular high-variance changes that trigger the objective function to change significantly.

The best performing optimizer turned out to be Adam.

Configured capability with various layers and nodes

The ability of a deep learning neural network model determines the variety of mapping functions that can be trained. A model with less capacity cannot recognize the training data set, which implies under-fitting, whereas a model with far too much power will remember the training data set, which implies over-fitting. The capacity of the neural network model is illustrated by setting the count of nodes within the network in addition to the number of layers.

A layer's number of nodes is referred to as the width. It was fairly straightforward to build large networks with one layer and several nodes. A network with sufficient nodes in the single hidden layer can in theory, learn to estimate any mapping function, even though we do not know in practice how many nodes are necessary. In a model, the number of layers is known as its depth. Improving the depth enhances the model's ability. We have tuned the number of neurons as follows:

- `hidden_size = [16,32,64,128]`

The ideal number of nodes turned out to be 64.

We have also tuned the number of hidden layers:

- `hidden_layer_number = [2,3,4,5,6,7,8,9,10]`

The best number of hidden layers turned out to be 3.

Configured gradient precision with batch size

Using gradient descent, neural networks are trained in which the approximation of the loss used for updating the weights is determined according to a subset of the training dataset. The batch size is simply the number of training data points used in the approximation of the error gradient and is a significant hyperparameter that affects the learning algorithm dynamics. It determines the number of observations to train the neural network before the internal parameters of the model are updated. Over one or multiple data samples, a batch size iterates and the model makes predictions accordingly. The predictions are then contrasted after the batch to the real output variables and an error is measured. The update

algorithm is often utilized to boost the model performance, alleviating this error.

There are multiple types of batch sizes:

- The learning algorithm is defined as batch gradient descent, when all training samples are utilized to build one batch.
- The learning algorithm is defined as stochastic gradient descent, when the batch size is of one sample.
- The learning algorithm is called mini-batch gradient descent, when the batch size is even more than one sample but less than the size of the training dataset.

In our problem, we have tuned multiple options for mini batch size:

- batch_size: [16, 32, 64, 128]

The best batch size turned out to be 32.

Configured optimization techniques in loss functions

When developing and customizing a neural network, the training process is done using stochastic gradient descent and demands the selection of a loss function. There exist several loss functions out there but understanding which to choose can be difficult. The method used to assess a candidate solution is pointed to as the objective function in any optimization problem. It is hence sought to maximize or minimize the objective function, which implies looking for a solution relative to the candidate with the highest or lowest score.

Generally, we try to mitigate the error with neural networks. Therefore, the objective function is alluded to as a cost or loss function. We have tuned the loss function types as follows:

- Mean Squared Error Loss
- Mean Squared Logarithmic Error Loss
- Mean Absolute Error Loss

The best loss function giving the best results is Mean Squared Error Loss.

Configured learning speed with learning rate

It is difficult to determine the weight of a neural network using an empirical tool. Alternatively, via an analytical optimization technique called stochastic gradient descent, the weights must be determined. Stochastic gradient descent used for neural networks is actually difficult to optimize, and the solution space could constitute multiple good solutions referred to as global optima in contrast to not-so-good solutions referred to as local optima. During each stage of this search process, which is referred to as step size, the rate of change occurring to the model is labeled as the learning rate. The learning rate is the most significant hyperparameter to be tuned in any neural network in order to achieve optimal results.

A high learning rate usually helps the model to learn quicker, at the expense of settling at an almost optimized final weight collection. The model can learn a more optimized or even globally optimized collection of weights with a reduced learning rate, however, it can require a substantially longer time to practice. A learning rate that is very high will manifest in weight updates that would be too big and the model's failure on the training dataset will prevail, hence overfitting would occur. Choosing the correct learning rate is a sensitive process, hence we tuned it the most. We chose the following values of learning rate to tune based on existing literature:

- lr: [0.0005, 0.0025, 0.0001, 0.001, 0.025, 0.01, 0.005, 0.05, 0.25, 0.1, 0.5, 1]

The best learning rate for our problem is 0.005.

Resolved vanishing gradients utilizing various activation functions

The activation function in a neural network is capable of translating the aggregate weighted input from the neuron into neuron activation or output for specific input. We tuned several activation functions:

- RELU: The rectified linear unit activation function is indeed a linear function that outputs the input if the value is positive but outputs zero if the input is negative. For several kinds of neural networks, RELU is the standard activation function since it achieves better results.
- CELU: Continuously differentiable exponential linear units are special since its derivative is restricted with respect to x , includes both the linear transfer function and the ReLU, and is identical in size with respect to the input shape. CELU was recently developed by (Barron, 2017) and is exclusively ready to use in PyTorch.

- GELU: Gaussian error linear units is a new powerful activation function (Hendrycks and Gimpel, 2016). The nonlinearity of GELU allows it to weigh the neural network's inputs by their value, instead of their sign as in conventional ReLUs.
- Leaky RELU: The Leaky ReLU changes the RELU function so that small negative values are output instead of zero when the input is less than zero.
- Tanh: The Tanh function stretched through a range of (-1,1). It maps negative inputs to negative values, zero inputs to zero values, and positive inputs to positive values. Tanh is also differentiable.

The best performing activation function in our problem is CELU.

Resolved overfitting with regularization

A challenging issue is training a deep neural network that has good generalization power when faced with new data. A deep learning model can perform horribly on new data, while another model with far too much capacity can learn very well and over-fit the training dataset. A recent trend to mitigating generalization error is by using a bigger capacity model that might need to use regularization, which holds the model weights when training.

Weight decay is a technique used for regularization by applying a little penalty to the loss function, generally the L2 norm of the weights.

$$loss = loss + weight\ decay\ parameter \times L2\ norm\ of\ the\ weights \quad (13.12)$$

where:

- weight decay parameter is a constant slightly less than 1, which is multiplied by the weights after each update.
- L2 norm of the weights is defined as:

$$|w| = \sqrt{\sum_{i=1}^N |w_i|^2} \quad (13.13)$$

where:

w is the weights

N is the size of the train data

Hence, in our problem, we have tuned the weight decay parameter as follows:

- weight_decay: [1e-4, 1e-3, 1e-2, 1e-6, 1e-5, 1e-7]

The best weight decay parameter turned out to be 1e-6.

Separate layers with Dropout

Dropout corresponds to the randomly selected ignorance of a neural network's neurons during the training process. By disregarding some neurons, hence during a specific forward or backward pass, these units are not considered as shown in **Figure 13.2**.

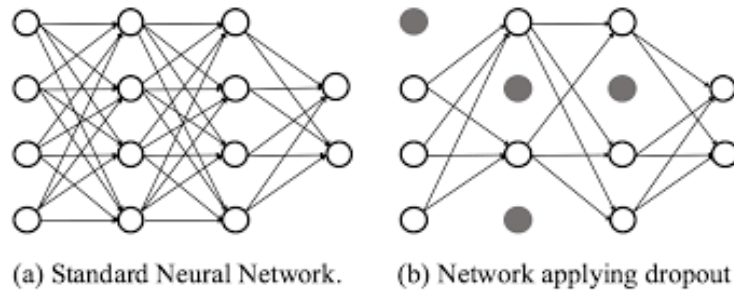


Figure 13.2: Dropout ([Wang and Manning, 2016](#))

Neurons are dropped (with a probability p) from the neural network at each training stage or are kept (with a probability $1 - p$). Hence, a reduced neural network is obtained. Also, inbound and outbound edges are also extracted from the dropped nodes. Dropout is actually used to prevent over-fitting. Most of the neural network models are occupied by a completely connected layer, so neurons establish co-dependence with each other during training, which lowers the power of each neuron, contributing to over-fitting training data. Hence, in our CQR neural network model, we decoupled all our layers with dropout as shown in **Section 13.5.2**. We have tuned several dropout rates:

- dropout_rate: [0.1, 0.01, 0.001, 0.0001, 0.002, 0.0002, 0.2, 0.002]

The best dropout rate giving the best results is 0.1.

13.6 Results

We care to compare the results of the CQR Neural Network default model, the optimized CQR Neural Network model after applying better deep learning techniques, and the Conformal Random Forests model. We chose the Conformal Random Forests model to be our base model because it lacks the quantile aspect, and hence follows a generic split conformal algorithm. We care to compare these two models to observe the effects of quantile regression on conformal methods. The following results are done on an alpha of 0.17 (choice of alpha is explained in **Section 13.6.2**). We note here that the coverage rate is outputted by the model according to the miscoverage rate α . The coverage rate should be around $1 - \alpha$. The results portrayed in the following tables show the α (miscoverage rate), the coverage rate (around $1 - \alpha$), and the average prediction interval length. We also added a new metric (which was not included in the CQR module): the standard deviation of the prediction interval lengths. We care to know if each prediction interval length varies greatly in comparison to the other prediction interval lengths to confirm if our model produces valid and consistent prediction interval lengths. It is unfavorable to obtain very low prediction interval lengths at some input data points and very high prediction interval lengths at others, making the model inconsistent and inaccurate. In this section we measure the percentage of improvement in the measured metrics between variable 1 and variable 2 by doing the following:

$$\text{percentage of improvement} = \frac{\text{metric}_{\text{variable1}} - \text{metric}_{\text{variable2}}}{\text{metric}_{\text{variable1}}} \quad (13.14)$$

where the metric is anything we measure (average prediction interval length or standard deviation prediction interval length).

13.6.1 Optimized vs Non-Optimized CQR Neural Network Model

Models	α	Coverage Rate	Average Length	Std Length
CQR NN	0.17	80.866193	0.872869	0.110452
Optimized CQR NN	0.17	80.672269	0.749394	0.213181
Conformal RF	0.17	81.965093	0.891318	0

Table 13.1: Experimental Results

We first train the CQR neural network model on all the data and all the columns, with no feature selection applied. We perform a 70/30 train and test split plus shuffle. We note in **Table 13.1** that the average length of the prediction interval before applying optimization techniques is 0.872mm when using the CQR neural network model. This average prediction interval length decreased upon applying optimization techniques to the CQR neural network model to reach 0.749mm. The standard deviation in the Conformal Random Forests cannot be compared to that of the CQR NN model, because conformal models which do not have the quantile aspect integrated in them by nature do not produce varying prediction interval lengths. The standard deviation of the prediction interval lengths of the CQR NN model increased slightly from 0.11mm when not applying optimization to 0.213mm when applying optimization techniques. We also note that the conformal random forest model is the worst performing, giving the highest prediction interval length of 0.891mm. The coverage rate is comparable across the three models.

Takeaway Message

Applying optimization techniques caused a decrease by 14% in average prediction interval length and an increase of 47% in the standard deviation of the prediction interval length. However, this trade-off between standard deviation and average prediction interval length is acceptable, since the standard deviation with the optimized model is still acceptable (0.21 mm compared to an average of 3.87mm of ET). Hence, we opt for the CQR model which we applied better deep learning techniques since it offered a lower average prediction interval length.

13.6.2 Verifying the Choice of α

We now aim to verify our choice of α . We compare the base model (conformal RF) and the best model (CQR NN) across higher miscoverage rate α to validate our choice. We do note that the coverage rate and the average prediction interval length are directly proportional - i.e if the coverage rate increased, the length of the prediction interval increases. This is due to the fact that when choosing a

lower miscoverage rate, we would be fitting the two quantiles on a lower α_{high} and a higher α_{low} , which would yield a larger prediction interval $[q_{\alpha_{high}}$ and $q_{\alpha_{low}}]$. We note that in **Tables** 13.3 and 13.4, the average prediction interval length decreased to 0.734mm when the coverage rate decreased to 74.66%, and to 0.679mm when the coverage rate decreased to 72%. Across the three tables (13.2, 13.3, and 13.4), the average prediction interval length upon using CQR NN is less than upon using Conformal RF. We also note that the optimized CQR NN performed better than the non-optimized CQR NN, yielding a lower average prediction interval length in both cases where $\alpha = 0.17$ (0.75 mm vs 0.872mm) and $\alpha = 0.25$ (0.68mm vs 0.69mm). We would also like to highlight that for $\alpha = 0.22$, the optimized CQR NN and the non-optimized gave comparable results with regards to the average prediction interval length, however, the non-optimized yielded a much higher standard deviation prediction interval length compared to the optimized version (0.351mm vs 0.11mm).

Models	α	Coverage Rate	Average Length	Std Length
Conformal RF	0.17	81.965093	0.891318	0
CQR NN	0.17	80.866193	0.872869	0.110452
Optimized CQR NN	0.17	80.672269	0.749394	0.213181

Table 13.2: Experimental Results on Miscoverage Rate of 17%

Models	α	Coverage Rate	Average Length	Std Length
Conformal RF	0.22	75.9534	0.76766	0
CQR NN	0.22	76.664512	0.723591	0.351379
Optimized CQR NN	0.22	74.6606	0.73424	0.110452

Table 13.3: Experimental Results on Miscoverage Rate of 22%

Models	α	Coverage Rate	Average Length	Std Length
Conformal RF	0.25	73.10924	0.718653	0
CQR NN	0.25	73.497091	0.692032	0.352847
Optimized CQR NN	0.25	71.687136	0.679308	0.103954

Table 13.4: Experimental Results on Miscoverage Rate of 25%

Takeaway message

An increase in the miscoverage rate is not worth the trade-off of a decreased average prediction interval length, since the best scenario caused a decrease of 9.3% in average prediction interval length but a decrease in 8.99% of coverage

rate. Hence, we choose $\alpha = 0.17$ as the optimal for giving a high coverage rate and an acceptable average prediction interval length.

13.6.3 Effect of SMOGN

We have tried to up-sample our data using the SMOGN method we described prior. We trained our two models, the optimized CQR NN and the Conformal RF, on the new up-sampled data by SMOGN. The results are described in **Tables 13.5** and **13.6**

Models	α	Coverage Rate	Average Length	Std Length
Conformal RF	0.17	81.965093	0.891318	0
CQR NN	0.17	80.866193	0.872869	0.110452
Optimized CQR NN	0.17	80.672269	0.749394	0.213181

Table 13.5: Experimental Results without SMOGN

Models	α	Coverage Rate	Average Length	Std Length
Conformal RF	0.17	83.219512	0.89566	0
CQR NN	0.17	82.146341	0.854948	0.371933
Optimized CQR NN	0.17	81.268293	0.95891	0.429621

Table 13.6: Experimental Results with SMOGN

We note that the SMOGN up-sampling was not of benefit for both the optimized CQR NN and the Conformal RF. For the case of the optimized CQR NN, the average prediction interval length increased from 0.749394mm to 0.95891mm. As for Conformal RF, the average prediction interval length increased from 0.891318mm to 0.89566mm. The coverage rate for both models remains comparable before and after SMOGN. The non-optimized version of CQR NN however, benefited from SMOGN up-sampling where the average prediction interval length decreased from 0.872mm to 0.855mm, however, the standard deviation prediction interval length increased from 0.11mm to 0.372mm. Nonetheless, the results for the optimized CQR NN without SMOGN remain the best, with an average prediction interval length of 0.749 and a low standard deviation prediction interval length of 0.213mm.

Takeaway Message

Applying SMOGN for CQR NN and Conformal RF was not efficient, for it increased the average prediction interval length instead of decreasing it. The average prediction interval length increased by 28% for CQR NN, and by a minor 0.45% for Conformal RF. SMOGN affected CQR NN negatively because the

SMOEN up-sampling created noisy data, causing the model to lose some of its performance power.

13.6.4 Feature Selection and Clustering

We perform clustering on all various clustering combinations by weather parameters and by climate, using clustering by kmeans or dendrograms. The best clustering combination is clustering by WS using kmeans (k=2), which yielded the best coverage rate and the lowest prediction interval length. We perform a 70/30 train and test split plus shuffle, we also perform the experiments on the best $\alpha = 0.17$. The feature selection scenarios for CQR neural networks and Conformal Random Forests (CRF) are all present in **Table 13.7**. The experimental results across all models (best model CQR and base model CRF) on the full dataset and dataset clustered by WS are found in **Table 13.9**.

Combinations	Model	Name	Input Combinations
1	CQR/CRF 1	Scenario A	All Columns
2	CQR/CRF 2	Scenario B	TA(5 lags)
3	CQR/CRF 3	Scenario C	TA(5 lags) WS(2 lags) RH(3 lags) EEflux LST(5 lags) EEflux NDVI(2 lags) EEflux Albedo(2 lags)
4	CQR/CRF4	Scenario D	TA(5 lags) RH(3 lags) EEflux LST(5 lags)

Table 13.7: Feature Selection Scenarios for CQR and CRF

Models	Dataset	Coverage Rate	Average Length	Standard Deviation Length
CQR1	All	80.672269	0.749394	0.213181
	WS0	80.907372	0.887651	0.179532
	WS1	82.731707	0.853867	0.061886
CRF1	All	81.96509373	0.8913186162	1.22E-16
	WS0	81.09640832	0.9560275584	2.38E-16
	WS1	82.63414634	0.8926340163	2.63E-16
CQR2	All	81.318681	0.866168	0.211656
	WS0	81.285444	0.870713	0.199262
	WS1	82.243902	0.922748	0.228682
CRF2	All	82.93471235	0.9741039634	2.39E-16
	WS0	80.52930057	0.9833938926	7.01E-17
	WS1	80.97560976	0.9710434943	2.36E-16
CQR3	All	81.060116	0.817072	0.069729
	WS0	85.471698	0.853635	0.106818
	WS1	82.926829	0.875833	0.113304
CRF3	All	81.70652877	0.8633496523	2.40E-16
	WS0	83.77358491	0.9803198099	1.15E-16
	WS1	81.26829268	0.8587318957	6.55E-17
CQR4	All	82.611506	0.938787	0.244776
	WS0	83.773585	1.005186	0.09444
	WS1	83.707317	1.001759	0.123977
CRF4	All	82.15901745	1.036273715	1.88E-16
	WS0	82.45283019	1.055340642	6.18E-17
	WS1	79.70731707	1.007670705	7.14E-17

Table 13.8: Experimental Results for $\alpha = 0.17$

We care to decide which FS scenario performed best on the CQR neural network model on the full dataset. According to **Table** 13.9 we note that CQR1 was the best performing in terms of the lowest average prediction interval length of 0.75mm and a standard deviation prediction interval length of 0.21 mm. The second best performing was CQR3, offering a prediction interval length of 0.82 mm and a standard deviation prediction interval length of 0.07mm. We note that applying FS scenario 2 increased the average prediction interval length by 8.5%, decreased the standard deviation prediction interval length by 66.6%, where the coverage rate remained comparable between CQR1 and CQR3.

We also note that CQR1 performed better than CRF1, yielding in a decrease of 15.7% in terms of average prediction interval length. We cannot compare the standard deviation of the prediction interval length between the two models because CRF models have a property of non-varying prediction interval lengths.

We also point out that in CQR1, experiments on the WS clusters did not cause a decrease in the average prediction interval length, however, it caused an increase of 2.06% in the coverage rate for WS1, and a 0.23% in WS0 (in which both are minimal), and a decrease of 70.4% in standard deviation prediction interval length for WS1, and 16.2% for WS0. Hence, we conclude that clustering did not aid in decreasing average prediction interval length, but however decreased the standard deviation prediction interval length.

Takeaway Message

We conclude that CQR1 is the best model if we are seeking the lowest average prediction interval length, and CQR2 is the best model if we are seeking comparable average prediction interval length to that of CQR1 but with a much lower standard deviation prediction interval length.

13.6.5 Varying Climates

We now aim to study model performance across all available climates. We demonstrate the results in **Table 13.9**. We note that the CQR1 model performed best on the climate Csa, giving the lowest average prediction interval length of 0.82mm, a coverage rate of 83.23%, and a standard deviation prediction interval length of 0.159 mm. However, we do note that CRF1 performed better than CQR1, which is quite justified. CQR is neural network-based, a complex model of three layers, dropout, and activation functions. This CQR model requires a big amount of data to generalize and perform well, which is not provided when subsetting the data by climate. Unlike CRF, which is a shallow model based on random forests, not requiring big amounts of data. CRF beat CQR by 2% in average prediction interval length, and by 2.62% coverage rate.

Takeaway message

In terms of prediction interval lengths, the CQR model performed best on Csa, followed by Cfa, Cwa, and Dsa. Similar to CRF, which performed best on Csa, followed by Cwa, Cfa, and Dsa. The best model to use when predicting on data climate subsets is CRF, due to its non-complex architecture.

Models	Climate	Coverage Rate	Average Length	Standard Deviation Length
CQR1	Cfa	87.678571	0.957683	0.140732
	Csa	83.232323	0.822785	0.158969
	Cwa	95.081967	1.120691	0.047453
	Dsa	84	1.160634	0.17885
CRF1	Cfa	86.78571429	0.9877247612	6.62E-17
	Csa	85.85858586	0.8033096313	1.30E-16
	Cwa	85.24590164	0.8239228278	6.36E-17
	Dsa	84	1.108557326	2.30E-16

Table 13.9: Experimental Results on Various Climates for $\alpha = 0.17$

13.6.6 Varying Seasons

We now study model performance across all seasons based on the seasonality study we did prior in **Section 5.4.5**. We portray the results in **Table 13.10**. We note that the CQR1 model performed best in the spring season, giving the lowest average prediction interval length of 0.90mm, a coverage rate of 84%, and a standard deviation prediction interval length of 0.196 mm. However, we do note that CRF1 performed better than CQR1 on the summer season, which is quite justified as said prior in the climate study. CQR is a complex model that requires a big amount of data to generalize and perform well, which is not given

Models	Dataset	Coverage Rate	Average Length	Standard Deviation Length
CQR1	TA Cluster 0	83.381089	1.11627	0.283659
	TA Cluster 1	83.974359	0.94573	0.316395
	TA Cluster 2	84.005376	0.89829	0.19615
CRF1	TA Cluster 0	83.95415473	1.058267593	2.66E-17
	TA Cluster 1	82.05128205	0.877425158	4.45E-16
	TA Cluster 2	83.60215054	0.9536486477	2.37E-16

Table 13.10: Experimental Results on Various Seasons for $\alpha = 0.17$

when clustering the data by seasons. Unlike CRF, which is a shallow model based on random forests, not requiring big amounts of data. CRF beat CQR by 7.7% in average prediction interval length.

Takeaway message

In terms of prediction interval lengths, the CQR model performed best in the spring season, followed by the summer season, and finally the winter season. CRF, however, performed best in the summer season, followed by the spring season, then the winter season. The best model to use when predicting data from the summer season is CRF, due to its non-complex architecture.

Chapter 14

SHAP

Interpretability is the extent to which the origin of a decision can be comprehended by a person. The greater a machine learning model's interpretability, the easier it is for anyone to understand why certain predictions have been done. Usually, models are thought of as black boxes, in which their mode of action is mysterious. Knowing the reason behind why a certain machine learning model took a certain decision is quite vital, to analyze its inner workings. A multitude of interpretability techniques are present, two of which we will be using are SHAP and LIME.

The aim of using SHAP lies in interpreting a predicted value by calculating the contribution of every input feature used in the prediction (Lundberg, 2017). Each input variable is given a Shapely value to point to its significance. SHAPely value is explained to be the average of the marginal contributions across all permutations (Parsa and Movahedi, 2020). The SHAPely value has three benefits:

- Global interpretability: the cumulative SHAPely values will explain how much each input variable contributes to the target variable, either positively or negatively. SHAP can display a positive or negative relationship with the output target variable. In the case of regression, SHAP shows if a low or high value of an input variable contributes to increasing or decreasing the value of an output variable. In the case of classification, SHAP shows if a low or high value of an input variable contributes to increasing or decreasing the probability of the output variable belonging to a certain class.
- Local interpretability: Each observation gets its own set of values from SHAP. This improves accountability greatly. Traditional input variable importance algorithms only display the results for the whole population, but not for each variable. Local interpretability helps us define and contrast the effects of the input variables on the target variable. This is important because SHAP allows us to study each instance of any input variable, and analyze its effects on our model prediction.

- Flexibility: SHAPely values can be calculated for any tree-based model in SHAP, while other methods used for model interpretation (like ELI5, Skater) use linear regression or logistic regression models.

14.0.1 Dealing with Categorical Data

In our data, we have three encoded columns: Site, Month, and Vegetation. The importance of these encoded columns cannot be represented by one column - say Month_1 because this column may contain 0/1 and it showing in the feature importance will be questionable or unreasonable. Our way of dealing with categorical data is adapted from the creator of the SHAP package, [Lundberg](#). Hence, to represent the encoded columns as 1 column, with its contribution to the prediction rather than the contribution of its encoded parts, we did the following:

- We fit our best probabilistic and point-wise models, which are the NGBoost and Gradient Boost respectively.
- We calculate the SHAPely value array for each data point in our data frame.
- In the array of SHAPely values, we sum up the SHAPely values corresponding to the encoded parts of each column (Sum up SHAPely values for Month_1, Month_2, Month_3). Thus, each encoded column, for example, Month, has 1 column of SHAPely values rather than 3 columns for Month_1, Month_2, Month_3. This will depict the importance of the column as a whole not as its encoded parts.
- We OR the encoded column parts to produce 1 column for each encoded column. Now we have 1 column which has the SUM of Shapley values to depict the power of all encoded columns rather than its parts (Lundberg et al, 2017).

14.1 Point-wise SHAP

We perform SHAP analysis on our best point-wise model Gradient Boost.

14.1.1 SHAP Summary Bar Plot

In **Figure 14.1**, the y -axis represents the input variables and the x -axis represents the mean SHAPely value for each variable. This plot only shows us how much a certain input variable contributed to shifting the output variable (ET) by a certain value. It does not show us if this shift is a positive or a negative, or if this shift caused the predicted ET to be closer to the actual. As noticed, TA is the largest contributor which shifted the output variable by an average of 0.7

mm. Following TA is NDVI, which contributed to shifting the output variable by an average of 0.4 mm. Month comes in third, shifting the output variable by an average of 0.25 mm.

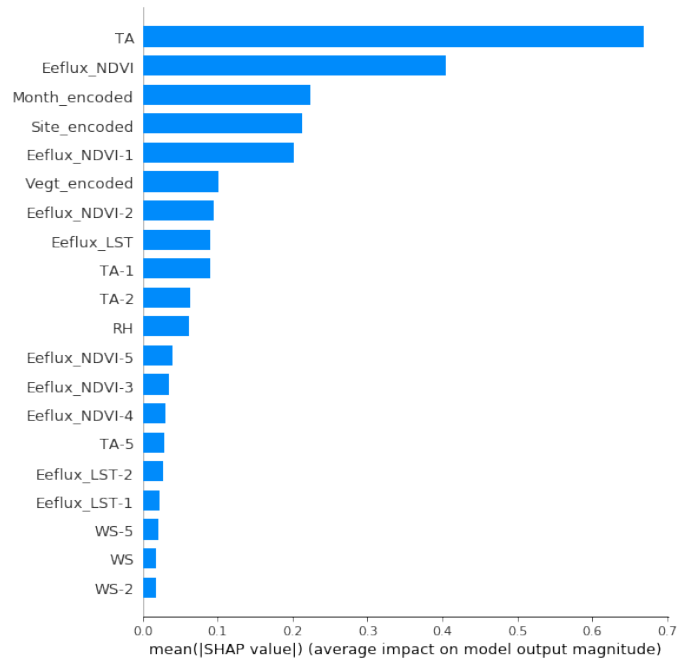


Figure 14.1: SHAP summary bar plot

14.1.2 SHAP Summary Plot

The biggest difference of this plot from the regular variable importance plot **Figure 14.1** is that it shows the positive and negative relationships of the predictors with the target variable. It looks dotted because it is made of all the dots in the training data. The plot's main features are:

- Feature importance: The variables are ranked in descending order.
- Impact: The horizontal location shows whether the effect of that value is associated with a higher or lower value for prediction.
- Original value: The color shows whether the input variable value is high (in red) or low (in blue) for that observation. A mix of blue and red shows that the input variable value is around the average.

From **Figure 14.2**, we analyze that TA is the highest contributing feature for our model. When TA has high values, the SHAPely values are high (hence the model output value increased). NDVI values, for example, yield higher SHAPely values when they are high and lower SHAPely values when they are low. Month and site, since they are encoded variables, cannot be measured by their value. It can only be mentioned that they rank second and third in terms of contribution respectively.

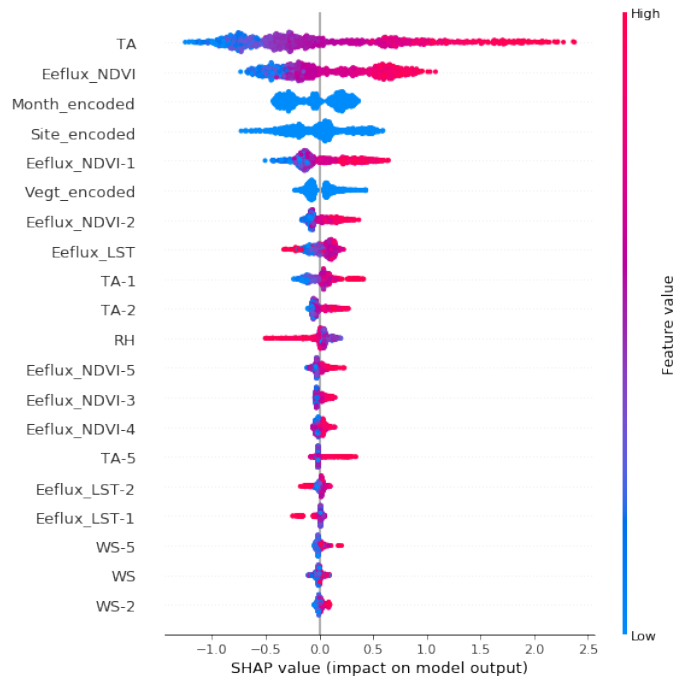


Figure 14.2: SHAP summary plot

14.1.3 SHAP Decision Plot for All Predictions

- The x -axis represents the model's output.
- The y -axis lists the model's features. By default, the features are ordered by descending importance.
- The importance is calculated over the observations plotted. For local decision plots (made on single instances of the dataset), the importance is different from the importance ordering for the entire dataset.
- Each observation's prediction is represented by a colored line.
- The model base value is the value that would be predicted if the model was not exposed to the top contributing features. At the top of the plot, each line strikes the x -axis at its corresponding observation's predicted value. This value determines the color of the line on a spectrum. Moving from the bottom of the plot to the top, SHAPely values for each feature are added to the model's base value. This shows how each feature contributes to the overall prediction.

Figure 14.3 represents the decision plot on a global level (for all predictions). The top and bottom vertical bands represent the model output value (real ET). The y -axis represents the input variables. As noticed, the model output value before introducing the major contributing input variables was restricted between 3.5 and 4.5 mm. This range witnessed a big shift upon introducing Month, Site, EEflux NDVI, and TA. The red color represents a high-value influence (introduction of a variable yielded in a higher value real ET), and the blue color represents a low-value influence (introduction of a variable yielded in a lower value real ET). For example, TA has a negative effect on ET values from 0 to 4, causing a decrease in the predicted ET value. However, TA has a positive effect on ET values which are greater than 4, causing a decrease in the predicted ET value.

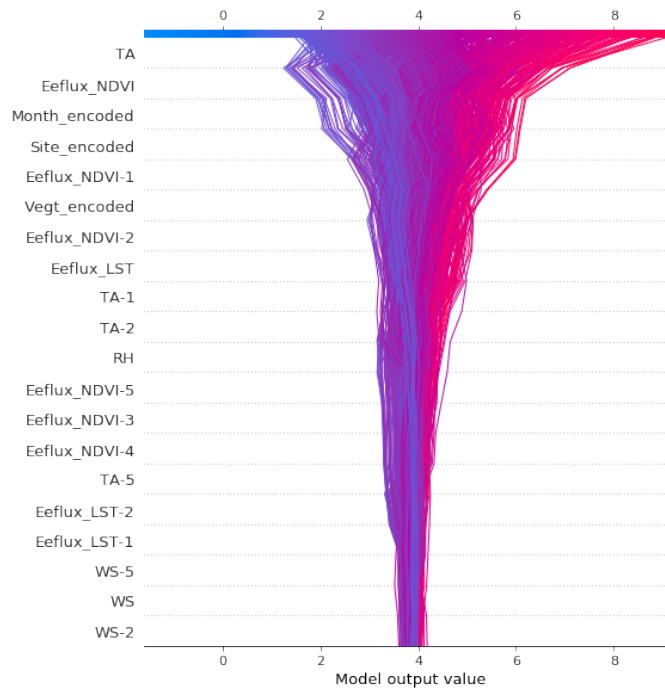


Figure 14.3: SHAP decision plot

14.2 Probabilistic SHAP

We perform SHAP analysis on our best probabilistic model NGBoost.

14.2.1 SHAP Summary Bar Plot

In **Figure 14.4**, the y -axis represents the input variables and the x -axis represents the mean SHAPely value for each variable. The SHAPely value will determine the impact of each variable on our predictions (how much does the input variable contribute to increasing or decreasing the value of the output variable). As noticed, TA is the largest contributor which shifted the output variable by an average of 0.2 mm. Following TA is Month, which contributed to shifting the output variable by an average of 0.1 mm. Site comes in third, shifting the output variable by an average of 0.08 mm.

14.2.2 SHAP Summary Plot

From **Figure 14.5**, we analyze that TA is the highest contributing feature for our model. When TA has high values, the SHAPely values are high (hence the model output value increased). NDVI values, for example, yield higher SHAPely values when they are high and lower SHAPely values when they are low. Month and site, since they are encoded variables, cannot be measured by their value, it

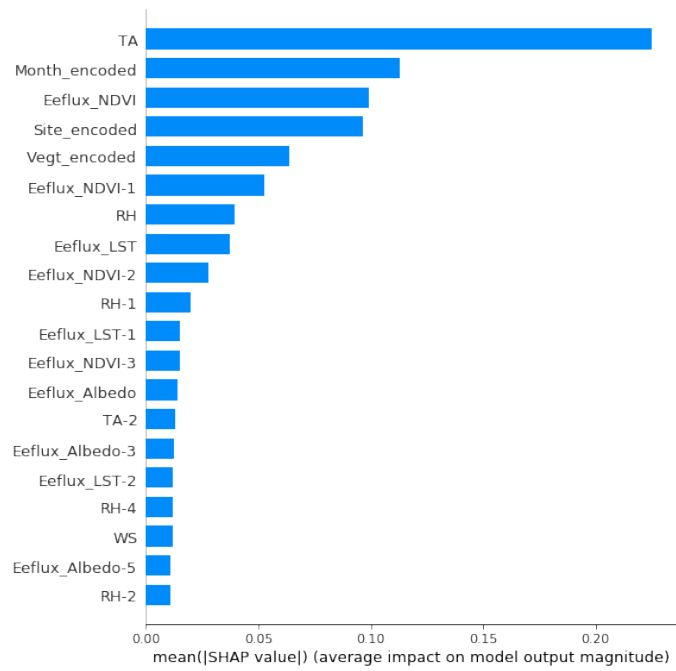


Figure 14.4: SHAP summary bar plot

can only be mentioned that they rank second and third in terms of contribution respectively.

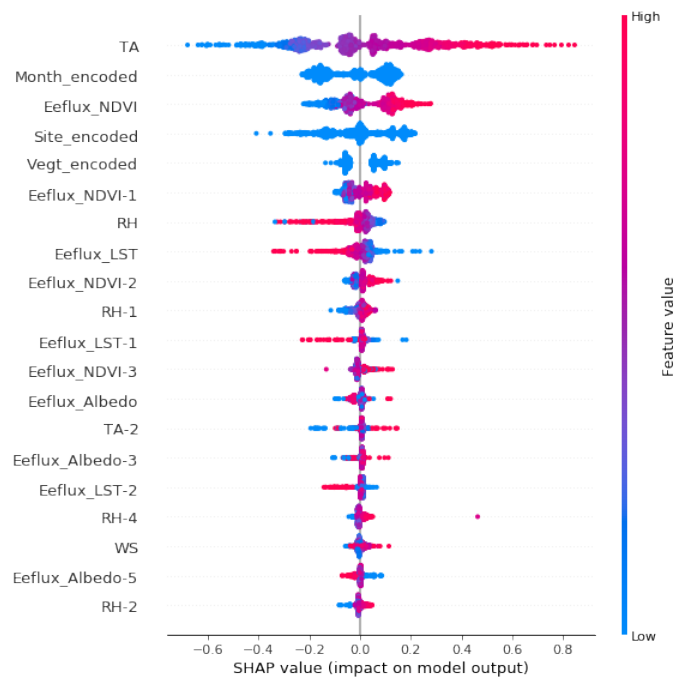


Figure 14.5: SHAP summary plot

14.2.3 SHAP Decision Plot for All Predictions

Figure 14.6 represents the decision plot on a global level (for all predictions). The top and bottom vertical bands represent the model output value (real ET). The y-axis represents the input variables. As it is noticed, the model output value before introducing the major contributing input variables was restricted between 0.75 and 1.5. This range witnessed a big shift upon introducing RH, EEflux NDVI, Site, Month, Vegetation, and TA. The red color represents a high-value influence (introduction of a variable yielded in a higher value real ET), and the blue color represents a low-value influence (introduction of a variable yielded in a lower value real ET). For example, TA has a negative effect on ET values from 0 to 1.5, causing a decrease in the predicted ET value. However, TA has a positive effect on ET values which are greater than 1.5, causing a decrease in the predicted ET value.

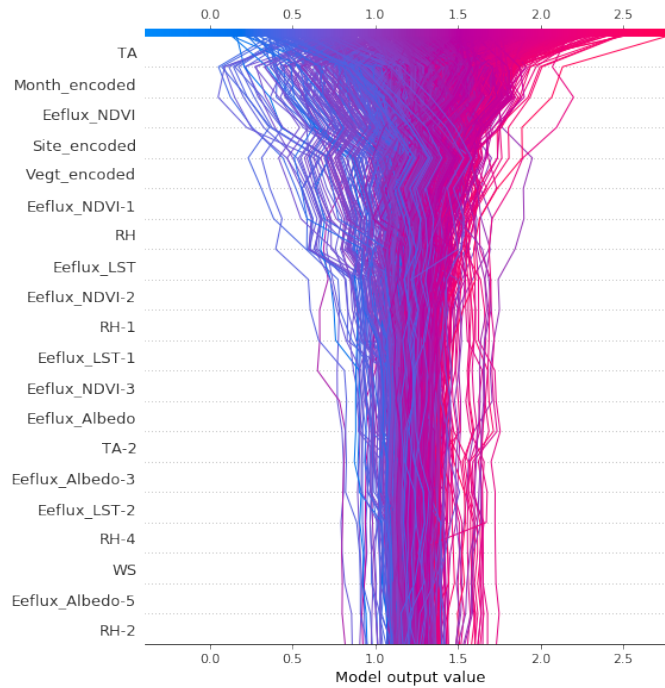


Figure 14.6: SHAP decision plot

14.3 Comparison between Point-wise and Probabilistic SHAP

The results of SHAP probabilistic and SHAP point-wise are quite identical. We notice that both exhibit the same trends, in which high values of EEflux NDVI implies high values of predicted ET, low values of RH implies high values of

predicted ET, high values of WS implies high values of predicted ET. Also, they both show that the most contributing features are TA, NDVI, Site, and Month.

14.4 Common Observations

The following summarizes our findings from the SHAP plots we implemented across probabilistic and point-wise models:

- High values of TA imply high values of predicted ET
- High values of EEflux NDVI imply high values of predicted ET
- Low values of EEflux LST imply high values of predicted ET
- Low values of RH imply high values of predicted ET
- Albedo does not impact the predicted ET
- High values of WS imply high values of predicted ET
- TA, NDVI, and LST are the top contributing features

All these observations seem to be in agreement with what irrigation experts consider to be true. We would also like to highlight that for both models, the interpretability did not change when we tested SHAP on the most accurate or rare predictions.

Chapter 15

LIME

LIME stands for local interpretable model-agnostic explanations. This interpretability tool helps us understand why a certain black-box machine learning model outputs a specific prediction (Ribeiro et al., 2017). LIME is based on implementing a local surrogate model, which is trained on individual data points and tries to approximate predictions of the black-box model. LIME does not offer a global interpretation, but rather a local one. Local interpretability is important since it shows us how each instance of our data is affected by the set of input features. The creators of LIME argued that locality is what makes LIME special, for no two input data points have the same interpretability, thus it is important to study each input test point rather than make a generalization. When using LIME, the machine learning model is thought of as an unknown entity that cannot be analyzed, hence it is the job of LIME to check how the model will react upon being exposed to new data. LIME creates a new dataset made of permutations of the original data and produces the corresponding predictions. LIME trains a linear regression model on the permuted (newly generated) data set. LIME then calculates the distance (using euclidean distance) between the generated samples of the permuted dataset and the chosen test point the user wants to analyze (this point is referred to as the point of interest). LIME aims to create a good local learner, not a global one, which would be able to approximate the prediction.

The steps taken to train a local surrogate model are:

1. We choose a test point we want to interpret.
2. LIME creates a dataset of permutations belonging to the instance of interest.
3. LIME weighs the created samples according to their distance to the instance of interest.
4. A weighted base model is trained on the permuted dataset, and thus we interpret the instance of interest according to our local surrogate model.

We use surrogate models if a model is too complex to test. Hence, the surrogate model is a basic model that imitates complex model mechanisms. Surrogate models are usually a linear regression or decision tree trained on a complex model’s original inputs and predictions.

15.1 Observations

We applied LIME after we split our data by 70/30 ratio and performed shuffling. LIME uses a base linear regression model. We further define a LIME tabular explainer, which takes in as input:

1. The training data
2. One random testing data point
3. The categorical columns: Site, Month, and Vegetation
4. The numerical columns: Wind Speed, Relative Humidity, Air Temperature, Eefflux LST, Eefflux Albedo, and Eefflux NDVI
5. The number of top contributing features to portray: 10

15.1.1 LIME on Two Accurate Data Point

Now we aim to portray how LIME’s locality is useful by observing the interpretation across two accurate predictions.

We analyze the results for the first accurate test point ($RMSE \leq 1$) at a local level in **Figure 15.1**. We note that the predicted ET value for this specific input data point is 1.74 mm.

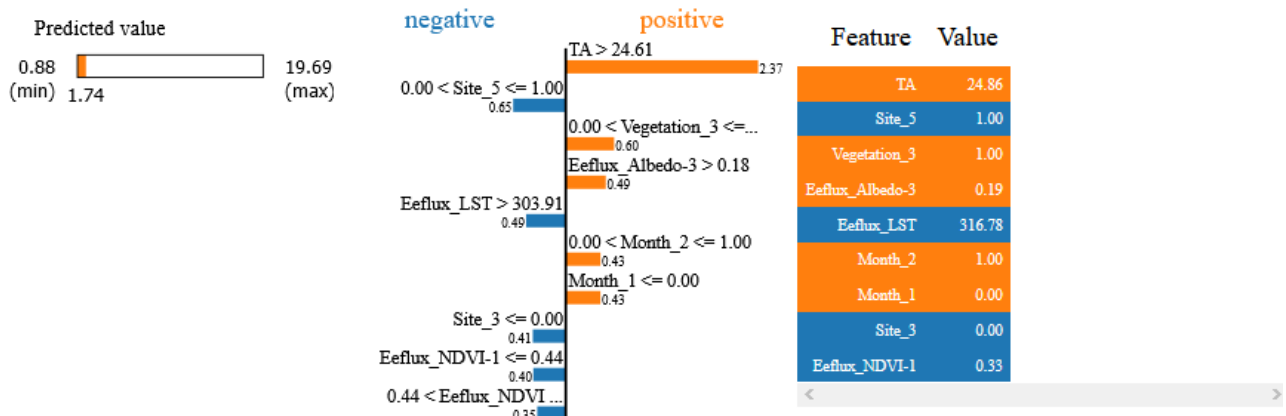


Figure 15.1: LIME plot

We note the following observations:

- TA values greater than 24.61 C are highly positively correlated with the output variable ET.
- Eefflux NDVI values less than 0.44 are highly negatively correlated with the output variable ET.
- Eefflux LST values greater than 303.91 are highly negatively correlated with the output variable ET.

We analyze the results for the second accurate test point ($RMSE \leq 1$) at a local level as well in **Figure 15.2**. We note that the predicted ET value for this specific input data point is 1.83 mm.

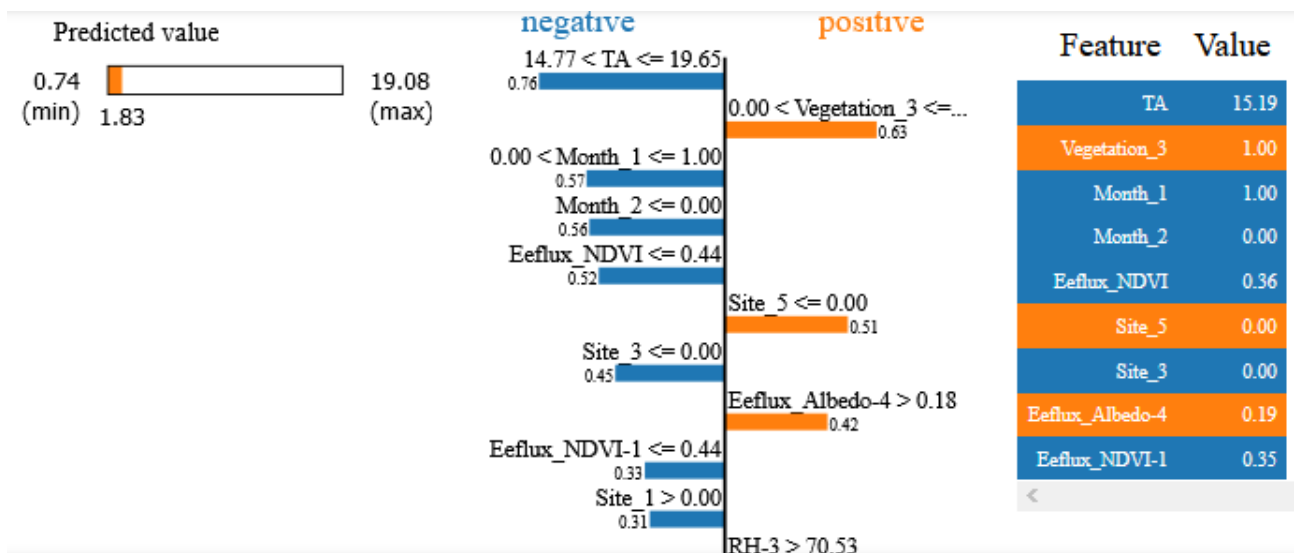


Figure 15.2: LIME plot

We note the following observations:

- TA values between 14.77 and 19.65 C are highly negatively correlated with the output variable ET.
- Eefflux NDVI values less than 0.44 are highly negatively correlated with the output variable ET.
- Eefflux Albedo values greater than 0.18 are highly positively correlated with the output variable ET.

We note that even these two local points are accurate and fall within the same range of ET values, they do not have the same exact interpretation for all feature values. The two points match in interpretation for only EEFlux NDVI, where values above 0.44 are highly negatively correlated with ET. This highlights the importance of locality in LIME where we can zoom in into each test point rather than issue a generalization across all the data.

15.1.2 Lime on an Inaccurate Data Point

We analyze the results for an inaccurate test point ($RMSE \geq 2$) at a local level in **Figure 15.3**. We note that the predicted ET value for this specific input data point is 3.56 mm.

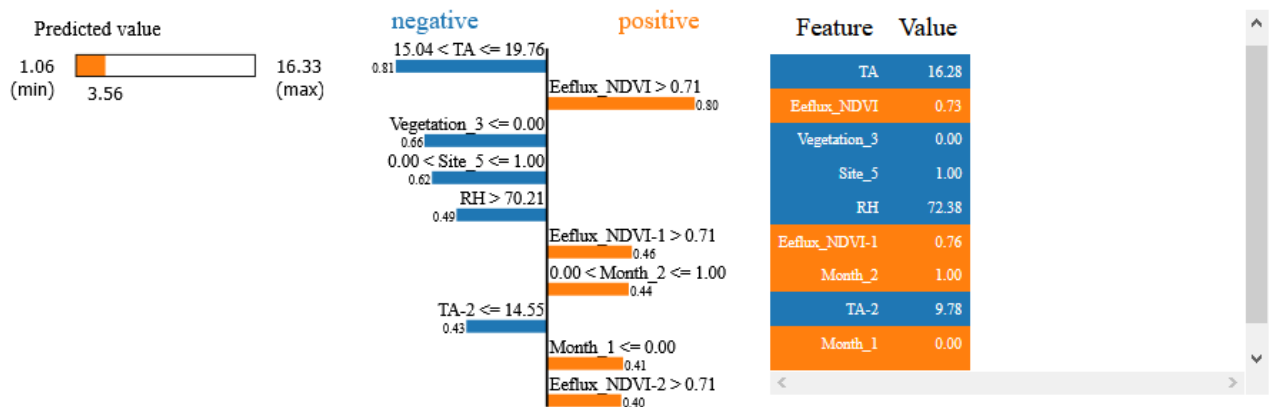


Figure 15.3: LIME plot

We note the following observations:

- TA values between 15.04 C and 19.76 C are highly negatively correlated with the output variable ET.
- EEflux NDVI values greater than 0.71 are highly positively correlated with the output variable ET.
- RH values greater than 70.21% are highly negatively correlated with the output variable ET.

15.1.3 Comparison between LIME on Accurate and Inaccurate Data Points

We note that for the first two observations, the numerical results of course differ but the trend is somehow similar:

- For accurate and inaccurate predictions, TA values greater than 24.61 C have a positive correlation with output ET. This implies that high values of TA imply high values of ET, aligning with the claims of agricultural experts.
- For accurate and inaccurate predictions, NDVI values greater than 0.71 have a positive correlation with output ET.

However, both these examples are local interpretations, the values may change if we change the point we are studying. We cannot add a generalization based on two points.

15.2 Comparison between SHAP and LIME

- Both are model agnostic and are surrogate models in the sense that they use the black-box machine learning models and tweak the input slightly by creating permutations of the data set to test the effect of this change on the prediction. For instance, if we have a sentence as an input, we create a new sentence by removing or adding a word. This becomes a new permuted sentence.
- LIME is faster than SHAP since it perturbs the data around a single prediction to build a model whereas, SHAP has to compute all the permutations globally.
- SHAP does not have an optimized module to support all types of algorithms, however, LIME allows so.
- SHAP offers global and local interpretability, unlike LIME which allows only local interpretability.

15.3 Comparison between SHAP, LIME, and Feature Selection

It is important to differentiate between feature selection methods and interpretability methods. Feature selection is a process that uses statistical techniques to assess how a subset of input features relates to the target variable, hence increases the accuracy of the predicted variable values. However, interpretability methods, identify which values contributed to changing the value of the target variable (be it a good change towards higher accuracy or a bad change towards lower accuracy). Interpretability methods allow the user to unravel the mystery behind black-box model predictions, whereas feature selection simply highlights which set of input variables reveal more accurate results. However, both feature

selection and interpretability techniques shed light on the most contributing i.e the most important input variables to be included when training a machine learning model. In our experiments, both interpretability methods (SHAP and LIME) and feature selection show that the top contributing features are TA, LST, RH, and NDVI. Thus, SHAP, LIME, and feature selection highlight the same notions.

Chapter 16

Deployment

The purpose of creating a machine learning model is to address a problem, and this model only does so when it is in development and regularly used by customers. Therefore, the deployment of models is as critical as model construction.

Before undertaking any machine learning projects, there are three main areas that we need to consider:

- **Storing and retrieving data:** It is possible to store data on-site, in cloud storage, or in a combination of the two. It makes sense to store the data where the model training will take place and the outcomes will be delivered. Hence, on-site model training and maintenance will be better suited for on-site data, particularly if the data is significant, whereas cloud-based data stored in cloud storage systems such as GCS, AWS S3, or Azure storage should be accompanied with cloud model training.
- **Frameworks and tools:** Our model won't train, operate, and launch on its own. We need systems and tools coupled with necessary software and hardware that would help us deploy ML models securely. For training our models, the frameworks we used are well known such as PyTorch, TensorFlow, and scikit-learn. These tools offer the three important aspects in deployment: efficiency, popularity, and support.
- **Feedback and iteration:** It's critical to get input from a model in development. In cases of model output depreciation/decay, bias creep, or even data skew and drift, actively tracking and controlling model status will inform you of any issue. This would guarantee that before the end-user notices, such concerns are easily resolved. A new model to be deployed should be checked properly. Continuous integration is the process where continuous testing and deploying new models without interrupting the current model processes is done.

There are many factors affecting deployment. We however chose to mention the three most important ones, and it is our goal to find a trade-off between these three.

Chapter 17

Conclusion and Future Work

We have constructed a flexible and robust probabilistic NGBoost model that successfully predicts ET (mm) with an R2 of 0.64-0.69 and an accuracy of 66% - 68% across the whole dataset and across the wind speed clusters. This model was also successful in minimizing the proportional bias between real ET and EEFlux ET, yielding an R2 of 0.73-0.74 across the wind speed cluster and the union of clusters. NGBoost proved to be an adequate model for this problem, for it beat the EEflux model and the base MC Dropout model. We have also experimented with conformal quantile regression, in which the CQR Neural Network produced competitive results, giving a coverage rate of 81% and an average prediction interval of 0.75 mm for ET (mm). Both CQR Neural Network and NGBoost proved to be favorable algorithms for this business model, since they produce prediction intervals and probability distributions respectively, providing more inclusion for predicted ET (mm), unlike point-wise models. To explore the inner workings of our probabilistic and point-wise models, we resorted to interpretability techniques using SHAP, where it was shown that TA and NDVI were the top contributing features. To identify the most important input features for obtaining the fastest training and the least economically expensive model, along with the best-performing model, we applied different filter-based feature selection techniques based on normalized mutual information. Feature selection techniques also matched the analysis of SHAP, for both NGBoost and Gradient Boost. Hence, our study offers a solution for each client. If the client wants accurate ET point predictions, Gradient Boost models have shown to be effective in predicting real ET with 67% accuracy and minimizing the bias between real and EEflux (METRIC) ET with an accuracy of 76%. If the client wishes to obtain ET predictions that are certain, then NGBoost is the answer, giving the option for a probabilistic ET prediction, with an accuracy of 68%. NGBoost is also successful in minimizing the proportional bias between EEFlux ET and real ET with an accuracy of 75%. However, the client has an option to also receive a range of ET predictions, not just point predictions. This can be achieved using the CQR model, which guarantees an 81% coverage rate, i.e. guaranteeing that 81% of the

given data points will fit in its corresponding prediction interval. Our future work would be to continue to develop our models, include more recent technologies that would improve our predictions. We would also want to find more resources to scrape more data, to expose our models to new data, especially data in which ET is higher than 5 (mm), since it is considered to be rare. We would also want to integrate our models in an easy-to-use tool, which agricultural engineers and farmers can use on the fly to obtain ET predictions, especially across the summer and spring seasons.

Chapter 18

Appendix

This chapter includes all of the experimental results that were done but not mentioned in the original report. We have performed our experiments on all of our best and base probabilistic and point-wise models across all feature selection Scenarios (A,B,C, and D) mentioned in **Table 7.1** . This chapter includes the following:

1. Experiments done using Gradient Boost on all feature selection scenarios, on all the dataset and the clusters in **Section 18.1.1**
2. Experiments done using Linear SVR model on all feature selection scenarios, on all the dataset and the clusters in **Section 18.1.2**
3. Experiments done using NGBoost model on all feature selection scenarios, on all the dataset and the clusters in **Section 18.2.1**
4. Experiments done using MC Dropout model on all the dataset and the clusters in **Section 18.2.2**

18.1 Point-wise Models

18.1.1 Gradient Boost

Scenario A

All				
Error Metrics	Validation Scores	standard deviation	Validation Scores	Testing Scores
mean target		3.879262	3.879262	3.879262
F1		0.009579	0.943483	0.953806
F2		0.008088	0.914566	0.929839
F05		0.012953	0.974334	0.979041
Precision		0.015996	0.996077	0.996619
Recall		0.008241	0.896273	0.91452
R2		0.04218	0.595749	0.63341
Adjusted R2		0.049125	0.529818	0.621157
RMSE		0.073924	1.456111	1.311459
MSE		0.219019	2.125724	1.719925
MAE		0.036625	1.056378	0.990982
MAPE		2.217643	35.24045	34.31708
Accuracy		2.217643	64.75955	65.68292
Pearson C.C.		0.03113	0.77911	0.799876
Spearman C.C.		0.018561	0.776672	0.770248
Spatial Distance		0.03113	0.22089	0.200124
NMI		0.000614	0.999175	0.997899
AIC		35.82274	267.8505	838.9086
BIC		35.82274	267.8505	838.9086
Data Size train	-		-	3576
Data Size test	-		-	1547
Training Time (seconds)	-		-	0.177992
Testing Time (seconds)	-		-	0.002279

Table 18.1: Experimental Results on all the Data

WS C0			
Error Metrics	Validation Scores standard deviation	Validation Scores	Testing Scores
mean target	3.745098	3.745098	3.745098
F1	0.027374	0.92685	0.926386
F2	0.029049	0.892478	0.883619
F05	0.02957	0.964239	0.973504
Precision	0.033772	0.991058	1.007672
Recall	0.031392	0.871052	0.857235
R2	0.073106	0.453037	0.490665
Adjusted R2	0.12456	0.056828	0.437387
RMSE	0.165422	1.593956	1.646461
MSE	0.531677	2.568059	2.710833
MAE	0.086563	1.152431	1.152606
MAPE	4.736033	40.24127	39.25267
Accuracy	4.736033	59.75873	60.74733
Pearson C.C.	0.063836	0.687078	0.711975
Spearman C.C.	0.052277	0.651669	0.660099
Spatial Distance	0.063836	0.312922	0.288025
NMI	0.001366	0.998913	0.99686
AIC	24.66698	110.5425	527.5483
BIC	24.66698	110.5425	527.5483
Data Size train	-	-	1200
Data Size test	-	-	529
Training Time (seconds)	-	-	0.06011
Testing Time (seconds)	-	-	0.001536

Table 18.2: Experimental Results on WS Cluster 0

WS C1			
Error Metrics	Validation Scores standard deviation	Validation Scores	Testing Scores
mean target	3.951248052	3.951248052	3.951248052
F1	0.008925529	0.948749306	0.949443914
F2	0.010073752	0.924248653	0.929717553
F05	0.00976581	0.97462682	0.970025512
Precision	0.011559441	0.992703714	0.984249577
Recall	0.011462156	0.908625057	0.917015813
R2	0.043204055	0.642880051	0.650203029
Adjusted R2	0.055221609	0.546372499	0.632246306
RMSE	0.083376642	1.35169146	1.350347051
MSE	0.226121578	1.834021469	1.823437159
MAE	0.050981173	1.01344096	0.994269837
MAPE	2.267045228	33.01500184	33.46165559
Accuracy	2.267045228	66.98499816	66.53834441
Pearson C.C.	0.029266572	0.805897195	0.808527856
Spearman C.C.	0.022133	0.802358182	0.808881446
Spatial Distance	0.029266572	0.194102805	0.191472144
NMI	0.000358038	0.999676541	0.998046176
AIC	28.85423953	141.3310263	615.7413504
BIC	28.85423953	141.3310263	615.7413504
Data Size train	-	-	2361
Data Size test	-	-	1025
Training Time (seconds)	-	-	0.118891001
Testing Time (seconds)	-	-	0.002454996

Table 18.3: Experimental Results on WS Cluster 1

Scenario B

All				
Error Metrics	Validation Scores standard deviation	Validation Scores	Testing Scores	
mean target	3.879262	3.879262	3.879262	
F1	0.009983	0.937118	0.93851	
F2	0.009047	0.903913	0.909813	
F05	0.018899	0.973101	0.969077	
Precision	0.026778	0.998822	0.990586	
Recall	0.012745	0.883156	0.891636	
R2	0.045591	0.485414	0.469934	
Adjusted R2	0.048334	0.454835	0.462986	
RMSE	0.078028	1.62882	1.603379	
MSE	0.257189	2.659143	2.570825	
MAE	0.059527	1.222788	1.220784	
MAPE	2.181658	42.66504	43.68547	
Accuracy	2.181658	57.33496	56.31453	
Pearson C.C.	0.039972	0.706166	0.68779	
Spearman C.C.	0.030284	0.66237	0.654584	
Spatial Distance	0.039972	0.293834	0.31221	
NMI	0.000999	0.998654	0.995385	
AIC	33.9182	348.1034	1460.719	
BIC	33.9182	348.1034	1460.719	
Data Size train	-	-	3576	
Data Size test	-	-	1547	
Training Time (seconds)	-	-	0.071038	
Testing Time (seconds)	-	-	0.001942	

Table 18.4: Experimental Results on All the Data

WS C0				
Error Metrics	Validation Scores standard deviation	Validation Scores	Testing Scores	
mean target	3.745098	3.745098	3.745098	
F1	0.015986	0.926492	0.935222	
F2	0.028138	0.893956	0.906321	
F05	0.0171	0.962215	0.966026	
Precision	0.027981	0.988068	0.987715	
Recall	0.036912	0.873812	0.888027	
R2	0.121612	0.433398	0.394698	
Adjusted R2	0.146847	0.318816	0.370867	
RMSE	0.121885	1.665946	1.629318	
MSE	0.405218	2.790233	2.654676	
MAE	0.064715	1.207201	1.184502	
MAPE	7.23526	42.83931	46.68585	
Accuracy	7.23526	57.16069	53.31415	
Pearson C.C.	0.085993	0.67971	0.634766	
Spearman C.C.	0.083903	0.609475	0.533012	
Spatial Distance	0.085993	0.32029	0.365234	
NMI	0.002023	0.997043	0.990598	
AIC	18.15833	121.9175	516.4747	
BIC	18.15833	121.9175	516.4747	
Data Size train	-	-	1200	
Data Size test	-	-	529	
Training Time (seconds)	-	-	0.028335	
Testing Time (seconds)	-	-	0.001271	

Table 18.5: Experimental Results on WS Cluster 0

WS C1			
Error Metrics	Validation Scores standard deviation	Validation Scores	Testing Scores
mean target	3.951248	3.951248	3.951248
F1	0.011806	0.94046	0.937408
F2	0.014612	0.910867	0.907567
F05	0.011821	0.972137	0.969278
Precision	0.014126	0.994528	0.991757
Recall	0.017051	0.892192	0.888706
R2	0.041836	0.514393	0.486552
Adjusted R2	0.045738	0.469238	0.476324
RMSE	0.069092	1.569785	1.651558
MSE	0.212712	2.468998	2.727642
MAE	0.050245	1.214891	1.247809
MAPE	3.153278	41.71088	44.38431
Accuracy	3.153278	58.28912	55.61569
Pearson C.C.	0.032779	0.724482	0.701336
Spearman C.C.	0.04299	0.699381	0.687139
Spatial Distance	0.032779	0.275518	0.298664
NMI	0.001014	0.998742	0.996242
AIC	21.09805	212.4425	1028.524
BIC	21.09805	212.4425	1028.524
Data Size train	-	-	2361
Data Size test	-	-	1025
Training Time (seconds)	-	-	0.051571
Testing Time (seconds)	-	-	0.00172

Table 18.6: Experimental Results on WS Cluster 1

Scenario C

All			
Error Metrics	Validation Scores standard deviation	Validation Scores	Testing Scores
mean target	3.879	3.879262	3.879262
F1	0.011	0.93982	0.951174
F2	0.014	0.912242	0.930333
F05	0.012	0.969217	0.972971
Precision	0.015	0.989921	0.988066
Recall	0.016	0.894782	0.916938
R2	0.043	0.591334	0.636632
Adjusted R2	0.048	0.541156	0.627229
RMSE	0.112	1.439562	1.359309
MSE	0.323	2.084929	1.847722
MAE	0.061	1.053202	1.018594
MAPE	2.638	35.1504	35.24429
Accuracy	2.638	64.8496	64.75571
Pearson C.C.	0.030	0.776217	0.803192
Spearman C.C.	0.029	0.768927	0.794576
Spatial Distance	0.030	0.223783	0.196808
NMI	0.000	0.999439	0.997715
AIC	56.299	258.4658	949.786
BIC	56.299	258.4658	949.786
Data Size train	-	-	3576
Data Size test	-	-	1547
Training Time (seconds)	-	-	0.151055
Testing Time (seconds)	-	-	0.002236

Table 18.7: Experimental Results on All the Data

WS C0			
Error Metrics	Validation Scores standard deviation	Validation Scores	Testing Scores
mean target	3.743	3.742533	3.742533
F1	0.029	0.931506	0.9346
F2	0.035	0.894514	0.897781
F05	0.029	0.972184	0.974569
Precision	0.034	1.001659	1.00317
Recall	0.039	0.871645	0.874805
R2	0.094	0.444383	0.518532
Adjusted R2	0.141	0.173885	0.480211
RMSE	0.227	1.621497	1.526884
MSE	0.758	2.680561	2.331375
MAE	0.110	1.172807	1.101012
MAPE	6.336	40.58652	40.6233
Accuracy	6.336	59.41348	59.3767
Pearson C.C.	0.074	0.681918	0.731576
Spearman C.C.	0.073	0.639615	0.685008
Spatial Distance	0.074	0.318082	0.268424
NMI	0.001	0.999158	0.997913
AIC	33.483	113.9953	448.6229
BIC	33.483	113.9953	448.6229
Data Size train	-	-	1202
Data Size test	-	-	530
Training Time (seconds)	-	-	0.044682
Testing Time (seconds)	-	-	0.001285

Table 18.8: Experimental Results on WS Cluster 0

WS C1			
Error Metrics	Validation Scores standard deviation	Validation Scores	Testing Scores
mean target	3.951	3.951248	3.951248
F1	0.009	0.948772	0.947193
F2	0.012	0.925458	0.924108
F05	0.009	0.973388	0.971461
Precision	0.012	0.990582	0.988342
Recall	0.015	0.910584	0.909334
R2	0.046	0.639139	0.653936
Adjusted R2	0.055	0.567357	0.640234
RMSE	0.090	1.353668	1.355415
MSE	0.248	1.84059	1.83715
MAE	0.051	1.013431	0.9963
MAPE	1.817	33.23469	33.16562
Accuracy	1.817	66.76531	66.83438
Pearson C.C.	0.031	0.803365	0.812382
Spearman C.C.	0.032	0.803988	0.815532
Spatial Distance	0.031	0.196635	0.187618
NMI	0.000	0.999409	0.998829
AIC	31.184	141.8995	623.4207
BIC	31.184	141.8995	623.4207
Data Size train	-	-	2361
Data Size test	-	-	1025
Training Time (seconds)	-	-	0.095191
Testing Time (seconds)	-	-	0.001811

Table 18.9: Experimental Results on WS Cluster 1

Scenario D

All				
Error Metrics	Validation Scores	standard deviation	Validation Scores	Testing Scores
mean target		3.879262	3.879262	3.879262
F1		0.012778	0.92437	0.919354
F2		0.013115	0.894404	0.888078
F05		0.015137	0.956496	0.952913
Precision		0.018142	0.979235	0.976681
Recall		0.01442	0.87552	0.868383
R2		0.048915	0.418474	0.415533
Adjusted R2		0.051349	0.389365	0.409035
RMSE		0.085611	1.715746	1.726605
MSE		0.299801	2.951114	2.981166
MAE		0.043926	1.311657	1.301229
MAPE		1.754164	44.5794	44.85336
Accuracy		1.754164	55.4206	55.14664
Pearson C.C.		0.038922	0.650062	0.645619
Spearman C.C.		0.03992	0.625122	0.633939
Spatial Distance		0.038922	0.349938	0.354381
NMI		0.000506	0.999407	0.998144
AIC		35.67534	385.2812	1689.81
BIC		35.67534	385.2812	1689.81
Data Size train	-		-	3576
Data Size test	-		-	1547
Training Time (seconds)	-		-	0.090217
Testing Time (seconds)	-		-	0.001952

Table 18.10: Experimental Results on All the Data

WS C0				
Error Metrics	Validation Scores	standard deviation	Validation Scores	Testing Scores
mean target		3.742533	3.742533	3.742533
F1		0.041911	0.910124	0.888677
F2		0.040164	0.861927	0.855941
F05		0.051389	0.964801	0.924017
Precision		0.062935	1.00561	0.94918
Recall		0.041715	0.832803	0.835425
R2		0.085376	0.243521	0.307434
Adjusted R2		0.099709	0.117661	0.284438
RMSE		0.220222	1.874767	1.914317
MSE		0.851076	3.563249	3.664609
MAE		0.126429	1.381494	1.419603
MAPE		5.893981	48.30195	50.7289
Accuracy		5.893981	51.69805	49.2711
Pearson C.C.		0.08399	0.507843	0.566834
Spearman C.C.		0.058549	0.481305	0.509288
Spatial Distance		0.08399	0.492157	0.433166
NMI		0.00154	0.99843	0.996029
AIC		28.34436	149.6035	688.3224
BIC		28.34436	149.6035	688.3224
Data Size train	-		-	1202
Data Size test	-		-	530
Training Time (seconds)	-		-	0.030599
Testing Time (seconds)	-		-	0.001153

Table 18.11: Experimental Results on WS Cluster 0

WS C1				
Error Metrics	Validation Scores	standard deviation	Validation Scores	Testing Scores
mean target		3.951248	3.951248	3.951248
F1		0.007859	0.937015	0.935843
F2		0.009543	0.91118	0.909044
F05		0.012401	0.964489	0.964271
Precision		0.017417	0.9838	0.984202
Recall		0.012704	0.894793	0.892014
R2		0.03445	0.498096	0.504465
Adjusted R2		0.037111	0.458975	0.4961
RMSE		0.103809	1.604592	1.602702
MSE		0.337029	2.585492	2.568655
MAE		0.081706	1.25124	1.241833
MAPE		1.565223	42.41261	42.25334
Accuracy		1.565223	57.58739	57.74666
Pearson C.C.		0.023652	0.709745	0.711744
Spearman C.C.		0.024637	0.678111	0.70639
Spatial Distance		0.023652	0.290255	0.288256
NMI		0.000526	0.99957	0.998144
AIC		30.89788	222.3827	966.967
BIC		30.89788	222.3827	966.967
Data Size train	-		-	2361
Data Size test	-		-	1025
Training Time (seconds)	-		-	0.059912
Testing Time (seconds)	-		-	0.001719

Table 18.12: Experimental Results on WS Cluster 1

18.1.2 Linear SVR

Scenario A

All				
Error Metrics	Validation Scores	standard deviation	Validation Scores	Testing Scores
mean target		3.879262	3.879262	3.879262
F1		0.376086	0.752046	0.929881
F2		0.363565	0.725781	0.848155
F05		0.391585	0.781488	1.029037
Precision		0.404004	0.803207	1.107787
Recall		0.052771	0.867746	0.80121
R2		0.27087	0.196681	-0.15421
Adjusted R2		0.315254	0.06561	-0.19278
RMSE		0.34798	1.973392	2.486121
MSE		1.474799	4.015365	6.180795
MAE		0.330283	1.56825	1.991371
MAPE		13.34844	54.68012	54.48163
Accuracy		13.34844	45.31988	45.51837
Pearson C.C.		0.024177	0.672447	0.688844
Spearman C.C.		0.027197	0.674354	0.70171
Spatial Distance		0.024177	0.327553	0.311156
NMI		0.000211	0.999571	0.997618
AIC		119.6544	475.6014	2817.778
BIC		119.6544	475.6014	2817.778
Data Size train	-		-	3576
Data Size test	-		-	1547
Training Time (seconds)	-		-	0.992755
Testing Time (seconds)	-		-	0.006835

Table 18.13: Experimental Results on All the Data

WS C0			
Error Metrics	Validation Scores standard deviation	Validation Scores	Testing Scores
mean target	3.745098	3.745098	3.745098
F1	0.450751	0.450476	0.924358
F2	0.435117	0.434516	0.889356
F05	0.469426	0.468614	0.962228
Precision	0.484068	0.482186	0.989247
Recall	0.043862	0.832222	0.867458
R2	1.131227	-0.13919	0.334503
Adjusted R2	1.936735	-0.96088	0.26489
RMSE	0.77314	2.146732	1.802069
MSE	4.632016	5.206204	3.247454
MAE	0.795949	1.660204	1.347861
MAPE	23.22621	53.40591	49.03033
Accuracy	23.22621	46.59409	50.96967
Pearson C.C.	0.06202	0.574373	0.581932
Spearman C.C.	0.082147	0.529495	0.543894
Spatial Distance	0.06202	0.425627	0.418068
NMI	0.000804	0.9994	0.995812
AIC	69.32889	172.1702	623.0939
BIC	69.32889	172.1702	623.0939
Data Size train	-	-	1200
Data Size test	-	-	529
Training Time (seconds)	-	-	0.280065
Testing Time (seconds)	-	-	0.010685

Table 18.14: Experimental Results on WS Cluster 0

WS C1			
Error Metrics	Validation Scores standard deviation	Validation Scores	Testing Scores
mean target	3.951248	3.951248	3.951248
F1	0.279933	0.833756	0.913042
F2	0.275003	0.823533	0.942047
F05	0.28979	0.846945	0.885771
Precision	0.298993	0.857444	0.868477
Recall	0.043831	0.899431	0.962429
R2	1.004383	-0.11725	-0.56287
Adjusted R2	1.277514	-0.41946	-0.6431
RMSE	0.870662	2.265968	2.823497
MSE	5.344671	5.892666	7.972133
MAE	0.894505	1.898259	2.466694
MAPE	40.23148	70.54216	102.9951
Accuracy	40.23148	29.45784	-2.99507
Pearson C.C.	0.026368	0.696863	0.686307
Spearman C.C.	0.040037	0.691162	0.682317
Spatial Distance	0.026368	0.303137	0.313693
NMI	0.000542	0.999623	0.99834
AIC	148.1202	359.5049	2127.851
BIC	148.1202	359.5049	2127.851
Data Size train	-	-	2361
Data Size test	-	-	1025
Training Time (seconds)	-	-	0.612407
Testing Time (seconds)	-	-	0.001326

Table 18.15: Experimental Results on WS Cluster 1

Scenario B

All				
Error Metrics	Validation Scores	standard deviation	Validation Scores	Testing Scores
mean target		3.879262	3.879262	3.879262
F1		0.013428	0.92848	0.917308
F2		0.008049	0.887247	0.883889
F05		0.026028	0.974086	0.953354
Precision		0.036586	1.007305	0.979
Recall		0.010961	0.861871	0.86293
R2		0.047011	0.361246	0.353204
Adjusted R2		0.049817	0.323292	0.344727
RMSE		0.047607	1.792615	1.827637
MSE		0.170187	3.215735	3.340256
MAE		0.051404	1.354387	1.358255
MAPE		5.001448	45.74227	45.68759
Accuracy		5.001448	54.25773	54.31241
Pearson C.C.		0.040821	0.613159	0.598858
Spearman C.C.		0.039761	0.593332	0.597106
Spatial Distance		0.040821	0.386841	0.401142
NMI		0.000559	0.999374	0.998168
AIC		19.25119	417.1992	1865.755
BIC		19.25119	417.1992	1865.755
Data Size train	-		-	3576
Data Size test	-		-	1547
Training Time (seconds)	-		-	0.220173
Testing Time (seconds)	-		-	0.004298

Table 18.16: Experimental Results on All the Data

WS C0				
Error Metrics	Validation Scores	standard deviation	Validation Scores	Testing Scores
mean target		3.745098	3.745098	3.745098
F1		0.033781	0.92238	0.917312
F2		0.03951	0.885272	0.879682
F05		0.037485	0.963562	0.958305
Precision		0.046688	0.993674	0.987732
Recall		0.045552	0.862478	0.856265
R2		0.078437	0.354144	0.354556
Adjusted R2		0.094796	0.223599	0.329145
RMSE		0.244279	1.751876	1.803064
MSE		0.84672	3.128743	3.25104
MAE		0.14843	1.294705	1.289544
MAPE		4.811758	44.85172	42.39868
Accuracy		4.811758	55.14828	57.60132
Pearson C.C.		0.066796	0.60928	0.603655
Spearman C.C.		0.071444	0.539931	0.573292
Spatial Distance		0.066796	0.39072	0.396345
NMI		0.000721	0.999034	0.996022
AIC		34.38011	132.1092	623.6777
BIC		34.38011	132.1092	623.6777
Data Size train	-		-	1200
Data Size test	-		-	529
Training Time (seconds)	-		-	0.055442
Testing Time (seconds)	-		-	0.001145

Table 18.17: Experimental Results on WS Cluster 0

WS C1				
Error Metrics	Validation Scores standard deviation	Validation Scores	Testing Scores	
mean target	3.951248	3.951248	3.951248	
F1	0.010073	0.937629	0.927885	
F2	0.011911	0.89911	0.892656	
F05	0.013621	0.979735	0.96601	
Precision	0.018623	1.010064	0.993216	
Recall	0.014579	0.875197	0.870619	
R2	0.042391	0.403103	0.420972	
Adjusted R2	0.046232	0.347609	0.409437	
RMSE	0.093375	1.725456	1.784378	
MSE	0.32157	2.985916	3.184005	
MAE	0.075631	1.319576	1.328748	
MAPE	2.079728	44.14905	45.90649	
Accuracy	2.079728	55.85095	54.09351	
Pearson C.C.	0.028431	0.645159	0.648882	
Spearman C.C.	0.020204	0.63524	0.653431	
Spatial Distance	0.028431	0.354841	0.351118	
NMI	0.000686	0.999249	0.998829	
AIC	26.00364	256.9193	1187.093	
BIC	26.00364	256.9193	1187.093	
Data Size train	-	-	2361	
Data Size test	-	-	1025	
Training Time (seconds)	-	-	0.114308	
Testing Time (seconds)	-	-	0.001235	

Table 18.18: Experimental Results on WS Cluster 1

Scenario C

All				
Error Metrics	Validation Scores standard deviation	Validation Scores	Testing Scores	
mean target	3.879	3.879262	3.879262	
F1	0.427	0.651983	0.939456	
F2	0.417	0.636832	0.906565	
F05	0.439	0.6692	0.974824	
Precision	0.449	0.681982	0.999921	
Recall	0.063	0.861566	0.885887	
R2	0.557	-0.10144	0.429446	
Adjusted R2	0.625	-0.2368	0.41468	
RMSE	0.530	2.308936	1.681168	
MSE	2.570	5.611792	2.826325	
MAE	0.548	1.893484	1.302145	
MAPE	23.415	65.87749	47.22217	
Accuracy	23.415	34.12251	52.77783	
Pearson C.C.	0.022	0.660125	0.663295	
Spearman C.C.	0.035	0.658716	0.657714	
Spatial Distance	0.022	0.339875	0.336705	
NMI	0.000	0.999341	0.998412	
AIC	160.700	579.7561	1607.298	
BIC	160.700	579.7561	1607.298	
Data Size train	-	-	3576	
Data Size test	-	-	1547	
Training Time (seconds)	-	-	0.663098	
Testing Time (seconds)	-	-	0.009784	

Table 18.19: Experimental Results on All the Data

WS C0				
Error Metrics	Validation Scores standard deviation	Validation Scores	Testing Scores	
mean target	3.743	3.742533	3.742533	
F1	0.372	0.186123	0.953058	
F2	0.366	0.183107	0.893904	
F05	0.379	0.18945	1.020596	
Precision	0.384	0.191849	1.071202	
Recall	0.076	0.798603	0.858386	
R2	0.376	-0.12115	0.235607	
Adjusted R2	0.557	-0.66582	0.174768	
RMSE	0.520	2.398265	1.760862	
MSE	2.763	6.022501	3.100636	
MAE	0.477	1.869095	1.353061	
MAPE	18.018	55.64788	41.0553	
Accuracy	18.018	44.35212	58.9447	
Pearson C.C.	0.051	0.591143	0.608675	
Spearman C.C.	0.066	0.529432	0.566081	
Spatial Distance	0.051	0.408857	0.391325	
NMI	0.001	0.999158	0.996659	
AIC	50.057	205.061	599.7519	
BIC	50.057	205.061	599.7519	
Data Size train	-	-	1202	
Data Size test	-	-	530	
Training Time (seconds)	-	-	0.193547	
Testing Time (seconds)	-	-	0.001209	

Table 18.20: Experimental Results on WS Cluster 0

WS C1				
Error Metrics	Validation Scores standard deviation	Validation Scores	Testing Scores	
mean target	3.951	3.951248	3.951248	
F1	0.378	0.756506	0.946722	
F2	0.362	0.721801	0.907853	
F05	0.399	0.796168	0.989068	
Precision	0.416	0.825974	1.019468	
Recall	0.051	0.857352	0.883667	
R2	0.403	0.166838	0.44526	
Adjusted R2	0.484	0.001131	0.423296	
RMSE	0.395	2.007105	1.711336	
MSE	1.821	4.184121	2.928671	
MAE	0.369	1.583015	1.297803	
MAPE	11.210	50.95168	43.0744	
Accuracy	11.210	49.04832	56.9256	
Pearson C.C.	0.038	0.687688	0.672179	
Spearman C.C.	0.041	0.672073	0.673846	
Spatial Distance	0.038	0.312312	0.327821	
NMI	0.000	0.999678	0.998047	
AIC	83.795	321.0541	1101.412	
BIC	83.795	321.0541	1101.412	
Data Size train	-	-	2361	
Data Size test	-	-	1025	
Training Time (seconds)	-	-	0.421322	
Testing Time (seconds)	-	-	0.001234	

Table 18.21: Experimental Results on WS Cluster 1

Scenario D

All				
Error Metrics	Validation Scores	standard deviation	Validation Scores	Testing Scores
mean target		3.879262	3.879262	3.879262
F1		0.455363	0.455365	0.924219
F2		0.449877	0.449776	0.90779
F05		0.461765	0.461503	0.941254
Precision		0.466608	0.465924	0.952963
Recall		0.057675	0.843621	0.897158
R2		0.372364	-0.1429	0.132478
Adjusted R2		0.391083	-0.20014	0.122833
RMSE		0.339344	2.384827	2.060176
MSE		1.66952	5.802555	4.244326
MAE		0.335897	1.905829	1.686862
MAPE		22.45371	66.31816	67.88931
Accuracy		22.45371	33.68184	32.11069
Pearson C.C.		0.042703	0.515201	0.544858
Spearman C.C.		0.045639	0.516537	0.528606
Spatial Distance		0.042703	0.484799	0.455142
NMI		0.000331	0.999572	0.998023
AIC		99.4637	614.4546	2236.317
BIC		99.4637	614.4546	2236.317
Data Size train	-		-	3576
Data Size test	-		-	1547
Training Time (seconds)	-		-	0.43862
Testing Time (seconds)	-		-	0.001134

Table 18.22: Experimental Results on All the Data

WS C0				
Error Metrics	Validation Scores	standard deviation	Validation Scores	Testing Scores
mean target		3.742533	3.742533	3.742533
F1		0.44669	0.364697	0.895085
F2		0.4367	0.356343	0.871428
F05		0.457767	0.37375	0.920062
Precision		0.465859	0.380219	0.937502
Recall		0.04984	0.834837	0.85634
R2		0.197306	-0.05384	-0.00967
Adjusted R2		0.230264	-0.22914	-0.04319
RMSE		0.24042	2.196971	2.299013
MSE		1.067503	4.884482	5.285463
MAE		0.218867	1.695446	1.862656
MAPE		15.53585	56.54511	78.62848
Accuracy		15.53585	43.45489	21.37152
Pearson C.C.		0.073617	0.451065	0.46255
Spearman C.C.		0.085267	0.41704	0.438631
Spatial Distance		0.073617	0.548935	0.53745
NMI		0.001084	0.999157	0.996868
AIC		25.46671	187.6405	882.4289
BIC		25.46671	187.6405	882.4289
Data Size train	-		-	1202
Data Size test	-		-	530
Training Time (seconds)	-		-	0.12864
Testing Time (seconds)	-		-	0.000865

Table 18.23: Experimental Results on WS Cluster 0

WS C1			
Error Metrics	Validation Scores standard deviation	Validation Scores	Testing Scores
mean target	3.951248	3.951248	3.951248
F1	0.275823	0.824989	0.924301
F2	0.271839	0.813431	0.887687
F05	0.285019	0.839564	0.964065
Precision	0.294013	0.851054	0.992532
Recall	0.057725	0.88347	0.864848
R2	0.919232	-0.31472	0.304173
Adjusted R2	0.991165	-0.41728	0.292426
RMSE	0.756409	2.43851	1.952354
MSE	4.2412	6.518483	3.811687
MAE	0.754087	2.040769	1.521631
MAPE	34.74502	75.5865	54.2853
Accuracy	34.74502	24.4135	45.7147
Pearson C.C.	0.030796	0.563952	0.556454
Spearman C.C.	0.035725	0.564253	0.548866
Spatial Distance	0.030796	0.436048	0.443546
NMI	0.000322	0.99957	0.999122
AIC	132.8759	400.7501	1371.524
BIC	132.8759	400.7501	1371.524
Data Size train	-	-	2361
Data Size test	-	-	1025
Training Time (seconds)	-	-	0.288966
Testing Time (seconds)	-	-	0.001152

Table 18.24: Experimental Results on WS Cluster 1

18.2 Probabilistic Models

18.2.1 NGBoost

Scenario A

All			
Error Metrics	Validation Scores standard deviation	Validation Scores	Testing Scores
mean target	3.879262	3.879262	3.879262
F1	0.724166	0.0418159	0.929449
F2	0.708234	0.040897	0.908376
F05	0.724166	0.0418159	0.951522
Precision	0.752533	0.0434667	0.96683
Recall	0.903851	0.047759	0.89485
R2	0.668199	0.084339	0.666639
Adjusted R2	0.648947	0.12908	0.655728
RMSE	1.066462	0.1416	1.318048
MSE	1.157392	0.299238	1.737252
MAE	0.839573	0.109329	0.940261
MAPE	28.16987	4.798842	31.09375
Accuracy	71.83013	4.798842	68.90625
Pearson C.C.	0.825286	0.138659	0.816791
Spearman C.C.	0.796484	0.126366	0.79908
Spatial Distance	0.174714	0.138659	0.183209
NMI	0.996052	0.000961	0.998107
AIC	99.10395	242.3847	854.4149
BIC	99.10395	242.3847	854.4149
Probabilistic RMSE	1.066462	0.1416	1.318048
Probabilistic NLL	2.143754	0.883526	1.86122
Data Size train	-	-	(3576)
Data Size test	-	-	(1547)
Training Time (seconds)	-	-	25.30618
Testing Time (seconds)	-	-	1.928314

Table 18.25: Experimental Results on All the Data

WS C0				
Error Metrics	Validation Scores standard deviation	Validation Scores	Testing Scores	
mean target	3.745098	3.745098	3.745098	
F1	0.730998	0.422054	0.904888	
F2	0.722283	0.417039	0.875386	
F05	0.730998	0.422054	0.936447	
Precision	0.746417	0.431405	0.958738	
Recall	0.71668	0.413923	0.856765	
R2	0.651276	0.396647	0.557297	
Adjusted R2	0.582926	0.47439	0.51201	
RMSE	0.817753	0.122582	1.48576	
MSE	0.683747	0.198014	2.207483	
MAE	0.637171	0.093584	1.042673	
MAPE	21.48958	2.263248	35.35601	
Accuracy	78.51042	2.263248	64.64399	
Pearson C.C.	0.809548	0.239649	0.746751	
Spearman C.C.	0.790719	0.221651	0.710765	
Spatial Distance	0.190452	0.239649	0.253249	
NMI	0.993315	0.002649	0.997489	
AIC	-127.685	92.18078	418.8903	
BIC	-127.685	92.18078	418.8903	
Probabilistic RMSE	0.817753	0.122582	1.48576	
Probabilistic NLL	3.005719	1.543737	3.269164	
Data Size train	-	-	(1200)	
Data Size test	-	-	(529)	
Training Time (seconds)	-	-	9.521585	
Testing Time (seconds)	-	-	1.070178	

Table 18.26: Experimental Results on WS Cluster 0

WS C1				
Error Metrics	Validation Scores standard deviation	Validation Scores	Testing Scores	
mean target	3.951248	3.951248	3.951248	
F1	0.966278	0.015427	0.933627	
F2	0.947994	0.017635	0.911369	
F05	0.966278	0.015427	0.956999	
Precision	0.998569	0.017222	0.973242	
Recall	0.936225	0.019854	0.89711	
R2	0.612488	0.135856	0.677077	
Adjusted R2	0.600677	0.148152	0.660848	
RMSE	0.935304	0.258666	1.319989	
MSE	1.141702	0.543863	1.74237	
MAE	0.728649	0.203623	0.932915	
MAPE	26.26328	4.08352	29.99295	
Accuracy	73.73672	4.08352	70.00705	
Pearson C.C.	0.891604	0.078624	0.822881	
Spearman C.C.	0.871276	0.071272	0.82477	
Spatial Distance	0.108396	0.078624	0.177119	
NMI	0.996726	0.001068	0.997778	
AIC	-119.259	301.3352	569.1273	
BIC	-119.259	301.3352	569.1273	
Probabilistic RMSE	0.935304	0.258666	1.319989	
Probabilistic NLL	3.014142	2.701514	2.75348	
Data Size train	-	-	(2361)	
Data Size test	-	-	(1025)	
Training Time (seconds)	-	-	19.05093	
Testing Time (seconds)	-	-	1.649082	

Table 18.27: Experimental Results on WS Cluster 1

Scenario B

All				
Error Metrics	Validation Scores	standard deviation	Validation Scores	Testing Scores
mean target		3.877716	3.877716	3.877716
F1		0.699679	0.404022	0.90521
F2		0.679639	0.0392425	0.87847
F05		0.699679	0.0404022	0.933628
Precision		0.736007	0.0425196	0.953587
Recall		0.847271	0.073136	0.861504
R2		0.556864	0.0357662	0.541646
Adjusted R2		0.142873	0.0365432	0.535939
RMSE		1.642611	0.521867	1.546019
MSE		2.970516	1.849325	2.390175
MAE		1.254276	0.403901	1.145323
MAPE		39.31158	8.134683	38.3856
Accuracy		60.68842	8.134683	61.6144
Pearson C.C.		0.632961	0.204151	0.736801
Spearman C.C.		0.59377	0.160571	0.724057
Spatial Distance		0.367039	0.204151	0.263199
NMI		0.995914	0.001255	0.998167
AIC		799.3161	554.4	1347.133
BIC		799.3161	554.4	1347.133
Probabilistic RMSE		1.642611	0.521867	1.546019
Probabilistic NLL		2.578338	0.853732	1.913148
Data Size train	-	-	-	(3572)
Data Size test	-	-	-	(1546)
Training Time (seconds)	-	-	-	10.23882
Testing Time (seconds)	-	-	-	1.22531

Table 18.28: Experimental Results on All the Data

WS C0				
Error Metrics	Validation Scores standard deviation	Validation Scores	Testing Scores	
mean target	3.745098	3.745098	3.745098	
F1	0.726043	0.041933	0.908864	
F2	0.714957	0.041299	0.892829	
F05	0.726043	0.041933	0.925486	
Precision	0.745529	0.043072	0.936909	
Recall	0.7078	0.040896	0.882449	
R2	0.415772	0.068857	0.489946	
Adjusted R2	0.376226	0.07351	0.470906	
RMSE	1.033579	0.258146	1.527509	
MSE	1.183493	0.558112	2.333285	
MAE	0.984798	0.170842	1.123633	
MAPE	39.98961	2.535843	41.09596	
Accuracy	60.01039	2.535843	58.90404	
Pearson C.C.	0.688641	0.379931	0.702704	
Spearman C.C.	0.681177	0.318357	0.610496	
Spatial Distance	0.311359	0.379931	0.297296	
NMI	0.993558	0.001674	0.995092	
AIC	1.686969	148.1499	448.2096	
BIC	1.686969	148.1499	448.2096	
Probabilistic RMSE	1.033579	0.258146	1.527509	
Probabilistic NLL	2.311552	0.732691	2.376173	
Data Size train	-	-	(1200)	
Data Size test	-	-	(529)	
Training Time (seconds)	-	-	5.911447	
Testing Time (seconds)	-	-	0.952274	

Table 18.29: Experimental Results on WS Cluster 0

WS C1				
Error Metrics	Validation Scores standard deviation	Validation Scores	Testing Scores	
mean target	3.951248	3.951248	3.951248	
F1	0.931389	0.023406	0.901511	
F2	0.901918	0.022751	0.883925	
F05	0.931389	0.023406	0.919812	
Precision	0.986344	0.043709	0.932431	
Recall	0.883529	0.026876	0.872577	
R2	0.580989	0.177524	0.53082	
Adjusted R2	0.567034	0.183423	0.52195	
RMSE	1.347557	0.185174	1.559247	
MSE	1.850199	0.506727	2.431252	
MAE	1.054399	0.157958	1.179834	
MAPE	35.30507	7.028695	40.43974	
Accuracy	64.69493	7.028695	59.56026	
Pearson C.C.	0.788068	0.114323	0.729129	
Spearman C.C.	0.781062	0.120158	0.732186	
Spatial Distance	0.211932	0.114323	0.270871	
NMI	0.996432	0.000537	0.998633	
AIC	341.2263	162.3952	910.6163	
BIC	341.2263	162.3952	910.6163	
Probabilistic RMSE	1.347557	0.185174	1.559247	
Probabilistic NLL	1.882414	0.196875	1.943804	
Data Size train	-	-	(2361)	
Data Size test	-	-	(1025)	
Training Time (seconds)	-	-	6.437168	
Testing Time (seconds)	-	-	1.043871	

Table 18.30: Experimental Results on WS Cluster 1

Scenario C

All				
Error Metrics	Validation Scores	standard deviation	Validation Scores	Testing Scores
mean target		3.877716	3.877716	3.877716
F1		0.932506	0.050719	0.923358
F2		0.908507	0.057097	0.89668
F05		0.932506	0.050719	0.951671
Precision		0.976311	0.044086	0.971532
Recall		0.893353	0.06175	0.879736
R2		0.646066	0.0825319	0.641321
Adjusted R2		0.626159	0.097979	0.632277
RMSE		1.236892	0.233575	1.367626
MSE		1.717889	0.839411	1.870402
MAE		0.920493	0.0275959	0.988437
MAPE		28.72644	4.624189	32.224
Accuracy		69.27356	4.624189	67.776
Pearson C.C.		0.805882	0.142698	0.801498
Spearman C.C.		0.783191	0.099193	0.796945
Spatial Distance		0.194118	0.142698	0.198502
NMI		0.99603	0.001125	0.998167
AIC		222.7236	474.9225	968.033
BIC		222.7236	474.9225	968.033
Probabilistic RMSE		1.236892	0.433575	1.367626
Probabilistic NLL		2.761647	2.191618	1.939267
Data Size train	-	-	-	(3572)
Data Size test	-	-	-	(1546)
Training Time (seconds)	-	-	-	20.33592
Testing Time (seconds)	-	-	-	1.663427

Table 18.31: Experimental Results on All the Data

WS C0				
Error Metrics	Validation Scores standard deviation	Validation Scores	Testing Scores	
mean target	3.745098	3.745097566	3.745097566	
F1	0.781827	0.391150461	0.916023277	
F2	0.771666	0.385828729	0.875750012	
F05	0.781827	0.391150461	0.960179201	
Precision	0.800256	0.401850207	0.992059997	
Recall	0.765241	0.382713405	0.850812511	
R2	0.6833	0.309982533	0.53876824	
Adjusted R2	0.623588	0.368264968	0.502999246	
RMSE	0.760319	0.147650955	1.58580334	
MSE	0.599885	0.208144214	2.514772232	
MAE	0.594864	0.122147627	1.056406012	
MAPE	20.8016	5.950201602	36.1668165	
Accuracy	79.1984	5.950201602	63.8331835	
Pearson C.C.	0.821617	0.202296887	0.73413987	
Spearman C.C.	0.794355	0.219453241	0.704202928	
Spatial Distance	0.178383	0.202296887	0.26586013	
NMI	0.993999	0.001658974	0.996441589	
AIC	-141.707	102.9747948	487.8344026	
BIC	-141.707	102.9747948	487.8344026	
Probabilistic RMSE	0.760319	0.147650955	1.58580334	
Probabilistic NLL	2.39659	1.603067515	4.887243338	
Data Size train	-	-	(1200)	
Data Size test	-	-	(529)	
Training Time (seconds)	-	-	7.98205924	
Testing Time (seconds)	-	-	1.162084103	

Table 18.32: Experimental Results on WS Cluster 0

WS C1				
Error Metrics	Validation Scores standard deviation	Validation Scores	Testing Scores	
mean target	3.951248	3.951248	3.951248	
F1	0.950331	0.041371	0.933013	
F2	0.942027	0.024559	0.919277	
F05	0.950331	0.041371	0.947164	
Precision	0.965985	0.06985	0.95684	
Recall	0.936935	0.014124	0.910343	
R2	0.623965	0.173807	0.685718	
Adjusted R2	0.704152	0.189015	0.673606	
RMSE	1.119916	0.201656	1.303171	
MSE	1.486911	0.373672	1.698255	
MAE	0.718722	0.156307	0.980868	
MAPE	33.53428	2.121963	34.03512	
Accuracy	66.46572	2.121963	65.96488	
Pearson C.C.	0.903294	0.059215	0.828698	
Spearman C.C.	0.887376	0.049892	0.821375	
Spatial Distance	0.096706	0.059215	0.171302	
NMI	0.997021	0.000492	0.99834	
AIC	-102.164	212.2345	542.8416	
BIC	-102.164	212.2345	542.8416	
Probabilistic RMSE	0.919916	0.201656	1.303171	
Probabilistic NLL	1.900941	0.865156	1.885645	
Data Size train	-	-	(2361)	
Data Size test	-	-	(1025)	
Training Time (seconds)	-	-	14.12122	
Testing Time (seconds)	-	-	1.44062	

Table 18.33: Experimental Results on WS Cluster 1

Scenario D

All				
Error Metrics	Validation Scores	standard deviation	Validation Scores	Testing Scores
mean target		3.877716	3.877716	3.877716
F1		0.917845	0.048449	0.882991
F2		0.89385	0.041673	0.85807
F05		0.917845	0.048449	0.909402
Precision		0.962264	0.069755	0.927905
Recall		0.878818	0.040099	0.842224
R2		0.507581	0.169156	0.417306
Adjusted R2		0.495562	0.17328	0.410824
RMSE		1.482023	0.358699	1.743149
MSE		2.325058	1.07122	3.038569
MAE		1.149096	0.26715	1.305928
MAPE		37.58484	5.717834	43.89047
Accuracy		62.41516	5.717834	56.10953
Pearson C.C.		0.745924	0.114074	0.646549
Spearman C.C.		0.698192	0.070662	0.632445
Spatial Distance		0.254076	0.114074	0.353451
NMI		0.99603	0.001125	0.998167
AIC		518.76	356.1511	1718.204
BIC		518.76	356.1511	1718.204
Probabilistic RMSE		1.482023	0.358699	1.743149
Probabilistic NLL		1.874287	0.394493	2.126849
Data Size train	-	-	-	(3572)
Data Size test	-	-	-	(1546)
Training Time (seconds)	-	-	-	14.19564
Testing Time (seconds)	-	-	-	12.49658

Table 18.34: Experimental Results on All the Data

WS C0				
Error Metrics	Validation Scores standard deviation	Validation Scores	Testing Scores	
mean target	3.745098	3.745098	3.745098	
F1	0.719146	0.367424	0.857525	
F2	0.715869	0.363987	0.841837	
F05	0.719146	0.367424	0.873808	
Precision	0.725961	0.375287	0.885012	
Recall	0.714008	0.362204	0.831693	
R2	0.374969	0.348967	0.272161	
Adjusted R2	0.327179	0.37551	0.247947	
RMSE	1.549303	0.911256	1.806996	
MSE	3.230727	3.876021	3.265236	
MAE	1.207339	0.668016	1.328028	
MAPE	41.70573	21.25074	47.14865	
Accuracy	58.29427	21.25074	52.85135	
Pearson C.C.	0.647848	0.262856	0.536214	
Spearman C.C.	0.624136	0.211173	0.512434	
Spatial Distance	0.352152	0.262856	0.463786	
NMI	0.993972	0.001562	0.996232	
AIC	143.6706	240.2746	625.9826	
BIC	143.6706	240.2746	625.9826	
Probabilistic RMSE	1.549303	0.911256	1.806996	
Probabilistic NLL	9.734956	13.8211	3.328815	
Data Size train	-	-	(1200)	
Data Size test	-	-	(529)	
Training Time (seconds)	-	-	8.093079	
Testing Time (seconds)	-	-	9.339847	

Table 18.35: Experimental Results on WS Cluster 0

WS C1				
Error Metrics	Validation Scores standard deviation	Validation Scores	Testing Scores	
mean target	3.951248	3.951248	3.951248	
F1	0.959114	0.021326	0.903909	
F2	0.93647	0.019055	0.884025	
F05	0.959114	0.021326	0.924708	
Precision	0.999995	0.03508	0.939114	
Recall	0.922078	0.020437	0.871248	
R2	0.593986	0.157713	0.489503	
Adjusted R2	0.578793	0.163609	0.480885	
RMSE	1.226962	0.293718	1.666607	
MSE	1.591707	0.751723	2.777578	
MAE	0.969458	0.231547	1.26592	
MAPE	31.87408	6.553955	42.57768	
Accuracy	68.12592	6.553955	57.42232	
Pearson C.C.	0.783874	0.092699	0.70124	
Spearman C.C.	0.759636	0.099854	0.686534	
Spatial Distance	0.216126	0.092699	0.29876	
NMI	0.996446	0.001169	0.998731	
AIC	167.0556	221.769	1047.119	
BIC	167.0556	221.769	1047.119	
Probabilistic RMSE	1.226962	0.293718	1.666607	
Probabilistic NLL	17.65674	31.78998	2.592213	
Data Size train	-	-	(2361)	
Data Size test	-	-	(1025)	
Training Time (seconds)	-	-	11.63386	
Testing Time (seconds)	-	-	10.7755	

Table 18.36: Experimental Results on WS Cluster 1

18.2.2 MC Dropout

Scenario C

All				
Error Metrics	Validation Scores standard deviation	Validation Scores	Testing Scores	
mean target	3.879262	3.879262	3.879262	
F1	0.530466	0.433227	0.926988	
F2	0.520666	0.425613	0.903416	
F05	0.530466	0.433227	0.951824	
Precision	0.552903	0.456169	0.969134	
Recall	0.811006	0.07384	0.888356	
R2	0.213774	0.034763	0.626824	
Adjusted R2	0.169592	0.036695	0.61742	
RMSE	1.829911	0.455682	1.407828	
MSE	3.556222	1.648407	1.98198	
MAE	1.37579	0.31433	1.018445	
MAPE	43.38254	5.676344	33.47211	
Accuracy	56.61746	5.676344	66.52789	
Pearson C.C.	0.562839	0.080806	0.792495	
Spearman C.C.	0.560932	0.09248	0.756005	
Spatial Distance	0.437161	0.080806	0.207505	
NMI	0.906134	0.08233	0.944391	
AIC	815.5856	382.9518	1058.297	
BIC	815.5856	382.9518	1058.297	
Probabilistic RMSE	1.796417	0	1.382826	
Probabilistic NLL	2.12759	0	2.015097	
Data Size train	-	-	(3576)	
Data Size test	-	-	(1547)	
Training Time (seconds)	-	-	2.995007	
Testing Time (seconds)	-	-	3.58109	

Table 18.37: Experimental Results on All Data

WS C0				
Error Metrics	Validation Scores standard deviation	Validation Scores	Testing Scores	
mean target	3.745098	3.745098	3.745098	
F1	0.499931	0.408827	0.906327	
F2	0.492841	0.40663	0.874174	
F05	0.499931	0.408827	0.940935	
Precision	0.521084	0.428289	0.965514	
Recall	0.637813	0.330231	0.853977	
R2	0.008672	0.112621	0.419566	
Adjusted R2	-0.18464	0.134582	0.373273	
RMSE	2.007751	0.651581	1.681331	
MSE	4.455623	2.334936	2.826874	
MAE	1.558244	0.504041	1.247949	
MAPE	52.94683	19.19914	46.55204	
Accuracy	47.05317	19.19914	53.44796	
Pearson C.C.	0.403251	0.15349	0.651439	
Spearman C.C.	0.357276	0.158237	0.464943	
Spatial Distance	0.596749	0.15349	0.348561	
NMI	0.932252	0.063249	0.879956	
AIC	301.0325	192.7612	549.7218	
BIC	301.0325	192.7612	549.7218	
Probabilistic RMSE	2.000466	0	1.685034	
Probabilistic NLL	2.201245	0	2.081765	
Data Size train	-	-	(1200)	
Data Size test	-	-	(529)	
Training Time (seconds)	-	-	1.850291	
Testing Time (seconds)	-	-	1.945739	

Table 18.38: Experimental Results on WS Cluster 0

WS C1				
Error Metrics	Validation Scores standard deviation	Validation Scores	Testing Scores	
mean target	3.951248	3.951248	3.951248	
F1	0.709167	0.357154	0.928957	
F2	0.683581	0.343389	0.912914	
F05	0.709167	0.357154	0.945573	
Precision	0.757161	0.384214	0.956985	
Recall	0.805535	0.064747	0.902523	
R2	0.192074	0.324008	0.611519	
Adjusted R2	0.119112	0.35341	0.596138	
RMSE	1.873377	0.13361	1.41453	
MSE	3.527392	0.51376	2.000896	
MAE	1.438444	0.076933	1.071778	
MAPE	46.53662	11.45806	36.00088	
Accuracy	53.46338	11.45806	63.99912	
Pearson C.C.	0.580384	0.158671	0.783206	
Spearman C.C.	0.627603	0.121583	0.74978	
Spatial Distance	0.419616	0.158671	0.216794	
NMI	0.981691	0.024931	0.982266	
AIC	590.4033	64.91045	710.9349	
BIC	590.4033	64.91045	710.9349	
Probabilistic RMSE	1.84109	0	1.385292	
Probabilistic NLL	2.126309	0	2.015869	
Data Size train	-	-	(2361)	
Data Size test	-	-	(1025)	
Training Time (seconds)	-	-	2.962275	
Testing Time (seconds)	-	-	3.517832	

Table 18.39: Experimental Results on WS Cluster 1

Chapter 19

Abbreviations

ET	Evapotranspiration
TA	Air Temperature
NDVI	Normalized Difference Vegetation Index
LST	Land Surface Temperature
WS	Wind Speed
RH	Relative Humidity
SMOTE	Synthetic Minority Oversampling Technique
SMOEN	SMOTE and Gaussian Noise
UBR	Utility Based Regression
CQR	Conformal Quantile Regression
CRF	Conformal Random Forests
SVR	Support Vector Regressor
MC	Monte Carlo

Bibliography

- Allen, R. (2005). The asce standardized reference evapotranspiration equation. *American Society of Civil Engineers*.
- Allen, R. (2015). Eefflux : A landsat-based evapotranspiration mapping tool on the google earth engine. *Scinapse*.
- Awad, M. and Khann, R. (2015). Support vector regression.
- Baier, W. and Robertson, G. (1965). Estimation of latent evaporation from simple weather observations.
- Bano, S. and Khan, N. (2018). A survey of data clustering methods.
- Barron, J. (2017). Continuously differentiable exponential linear units.
- Branco, P., Torgo, L., and Ribeiro, R. (2013). Smote for regression. *LIAAD - INESC TEC*.
- Branco, P., Torgo, L., and Ribeiro, R. (2017). Smogn: a pre-processing approach for imbalanced regression. *LIDTA*.
- Duan, T., Avanti, A., and Ding, D. (2020). Natural gradient boosting for probabilistic prediction.
- Fan, J. and Yue, W. (2019). Evaluation of svm, elm and four tree-based ensemble models for predicting daily reference evapotranspiration using limited meteorological data in different climates of china. *Agricultural and Forest Meteorology*.
- Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning.
- Gneiting, T., Balabdaoui, F., and Raftery, A. (2007). Probabilistic forecasts, calibration and sharpness.
- Granata, F. (2019). Evapotranspiration evaluation models based on machine learning algorithms—a comparative study. *Elsevier*.

- Gustafson, P. (2014). Bayesian inference in partially identified models: Is the shape of the posterior distribution useful?
- Hendrycks, D. and Gimpel, K. (2016). Gaussian error linear units (gelus).
- Huang, G., Wu, L., Ma, X., Zhang, W., Fan, J., Yu, X., Zeng, W., and Zhou, H. (2019). Evaluation of catboost method for prediction of reference evapotranspiration in humid regions.
- Huitema, B. and Laraway, S. (2006). Autocorrelation.
- Hüllermeier, E. and Waegeman, W. (2019). Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods.
- Kaneko, A., Kennedy, T., and Mei, L. (2019). Deep learning for crop yield prediction in africa.
- Kuha, J. (2004). Aic and bic: Comparisons of assumptions and performance.
- Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. *NIPS*.
- Lundberg, S. (2017). A unified approach to interpreting model predictions.
- Mauder, M., Foken, T., and Cuxart, J. (2017). Surface-energy-balance closure over land: A review.
- Meinshausen, N. (2006). Quantile regression forests.
- Miao, J. and Niu, L. (2016). A survey on feature selection.
- Mushtaq, R. (2011). Augmented dickey fuller test. *LIDTA*.
- Natekin, A. and Knoll, A. (2013). Gradient boosting machines, a tutorial.
- Parsa, A. and Movahedi, A. (2020). Toward safer highways, application of xgboost and shap for real-time accident detection and feature analysis.
- Ribeiro, M. T., Singh, S., and Guestrin, C. (2017). "why should i trust you?": Explaining the predictions of any classifier.
- Ribeiro, M. T. and Torgo, L. (2008). Utility-based performance measures for regression. *ICML*.
- Romano, Y., Patterson, E., and Candès, E. (2019). Conformalized quantile regression.
- Torgo, L. and Ribeiro, R. P. (2009). Precision and recall for regression.

Vovk, V., Shen, J., Manokhin, V., and ge Xie, M. (2019). Conformalized quantile regression.

Wang, S. and Manning, C. (2016). Fast dropout training.

