AMERICAN UNIVERSITY OF BEIRUT

DISCOVERING THE MOST PROMISING IDEAS IN A CROWDSOURCING PLATFORM FOR PRODUCT DEVELOPMENT

by
NOUR JAMAL ABSI HALABI

A thesis
submitted in partial fulfillment of the requirements
for the degree of Master of Engineering Management
to the Department of Industrial Engineering and Management
of the Maroun Semaan Faculty of Engineering and Architecture
at the American University of Beirut

Beirut, Lebanon
February 2021

AMERICAN UNIVERSITY OF BEIRUT

DISCOVERING THE MOST PROMISING IDEAS IN A CROWDSOURCING PLATFORM FOR PRODUCT DEVELOPMENT

by

NOUR JAMAL ABSI HALABI

Approved by:

[illegible]

Dr. Ali Yassine, Professor — Advisor
Department of Industrial Engineering and Management

Hazem

Dr. Hazem Hajj, Associate Professor — Member of Committee
Department of Electrical and Computer Engineering

J Azar

Dr. Jimmy Azar, Assistant Professor — Member of Committee
Department of Industrial Engineering and Management

Date of thesis defense: January 27, 2021

AMERICAN UNIVERSITY OF BEIRUT

THESIS RELEASE FORM

Student Name: Absi Halabi Nour Jamal

I authorize the American University of Beirut, to: (a) reproduce hard or electronic copies of my thesis; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes:

☒ As of the date of submission

☐ One year from the date of submission of my thesis.

☐ Two years from the date of submission of my thesis.

☐ Three years from the date of submission of my thesis.

Signature [illegible] Date: Feb 8th 2021

ACKNOWLEDGEMENTS

I would like to express my deepest appreciation and gratitude to my advisor, Prof. Ali Yassine, for his continuous support and advice throughout this journey. His guidance and encouragement are the reason for my research accomplishment today.

I would like to thank the committee members as well, Prof. Hazem Hajj and Prof. Jimmy Azar, for their feedback and insightful comments.

Finally, I would like to thank my family and friends for being my consistent source of motivation.

ABSTRACT
OF THE THESIS OF

Nour Jamal Absi Halabi for Master of Engineering Management

Title: Discovering the Most Promising Ideas in a Crowdsourcing Platform for Product Development

Obtaining and analyzing key customer and product information from various sources has become a top priority for major competitive companies who are striving to keep up with the digital and technological progress. From this point, the need for creating an idea crowdsourcing platform to collect ideas from different stakeholders has become a major component of a company's digital transformation strategy. Today, companies resort to idea crowdsourcing platforms to discover novel ideas from the public, employees, and vendors that they can use in their product development processes. However, these platforms suffer from problems that are related to the voluminous and vast amount of data. Different large sets of data are being spurred in these platforms as time goes by that render them unbeneficial or useless.

The aim of this thesis is to propose a solution on how to discover the most promising ideas to match them to the strategic decisions of a business regarding resource allocation and product development roadmap. The thesis introduces a 2-stage filtering process that includes a prediction model using a Random Forest Classifier that predicts ideas most likely to be implemented and a resource allocation optimization model based on Integer Linear Programming that produces an optimal release plan for the predicted ideas. The model was tested using real data on an idea crowdsourcing platform that remains unnamed in the thesis due to confidentiality. Our prediction model has proved to be 93% accurate in predicting promising ideas and our release planning optimization problem results were found out to be 85% accurate in producing an optimal release plan for ideas.

TABLE OF CONTENTS

ILLUSTRATIONS

Figure

TABLES

CHAPTER I

INTRODUCTION

The ongoing digital transformation has intensified the competition among rival companies causing the innovation process to be a collective process rather than individual *(Westerski, Dalamagas & Iglesias, 2013).* Today, companies from different industries (e.g. Spigit, Imaginitik, Nosco, BrightIdea, Salesforce, and Ideascale) strive for getting ideas from external sources and encourage the engagement of external contributors on their crowdsourcing platforms for innovation and ideation. For instance, “Proctor and Gamble” developed “Connect and Develop” to discover the most innovative ideas, and the platform actually helped increase their productivity by 60% where 45% of their adopted ideas were sought externally. Also, Walkers Crisps launched a “Do us a flavor” campaign to engage customers to send suggestions of a new flavor and promised the winner to be granted a million euros for the idea (Forbes & Schaefer, 2017). The point behind having a crowdsourcing platform is that while many problems and ideas can be solved and provided internally within the company, sometimes knowledge and creativity can be sought externally from different stakeholders like customers, vendors, or even employees of different cultures, age, and demographic locations. Also, Poetz and Schreier (2012) found that compared to in-house idea innovators, crowdsourcing participants provide ideas that are more novel and customer oriented. In addition, Crowdsourcing is certainly cheaper than outsourcing or hiring conventional employees (Howe 2006).

A. Definition of Crowdsourcing

There are interchangeable synonyms for idea crowdsourcing as there is no agreement on the clear definition associated with crowdsourcing. The terms cloud-based, co-creation, collective intelligence, and open innovation are all used interchangeably as idea crowdsourcing. According to Estellés-Arolas and González-Ladrón-de-Guevara (2012), there are at least 40 definitions of crowdsourcing in the literature. For instance, Alonso and Lease (2011) state that "crowdsourcing is the outsourcing of tasks to a large group of people instead of assigning such tasks to an in-house employee or contractor", while Brabham *(2008)* defines crowdsourcing as "a strategic model to attract an interested, motivated crowd of individuals capable of providing solutions superior in quality and quantity to those that even traditional forms of business can". However, the term crowdsourcing was first coined by Jeff Howe and Mark Robinson (2006) in a Wired Magazine article and was defined as "the act of a company or institution taking a function once performed by employees and outsourcing it to an undefined (and generally large) network of people in the form of an open call." It should be noted that the idea of crowdsourcing is not completely new, as it was applied before in many historical events but not as part of the digital web revolution. Back in 1714, the British Parliament offered rewards to anyone who could be able to develop a method to examine the longitude at sea. Another example dates back when Napoleon wanted someone to discover an idea to preserve food and offered 12,000 francs for the person who invented canned food in 1810. However, what is different today, is that the new information and communication technologies along with Web 2.0 have provided an environment that allows the submission, discussion, and evaluation of ideas across a single platform all around the world. It is important to note also that the concept of idea crowdsourcing stems from a more general concept referred to as social product development. Social Product Development is a new term in engineering design which includes several tenants like crowdsourcing, open innovation, IoT, crowdfunding, and mass collaboration coexisting altogether (Forbes & Schaefer, 2017). Abhari et al. (2016) refer to social product development as "the social product development model extends open innovation beyond customer involvement models to socially engaged individual actors fully involved in ideation and development of new products".

B. Implementation Types and organizational purpose

Several types of crowdsourcing initiatives can be identified based on two dimensions: tasks to solve and incentive structure. Idea Crowdsourcing on web platforms can be applied in short-lived contests, open calls with direct rewards, open calls with indirect benefits, and micro tasks as shown in Figure 1 (Shergadwala et al., 2019). Contests are applied in crowdsourcing when a well-bounded problem is made available for the crowd to solve in a short time and with a winning prize. An example of a crowdsourcing contest is the Gold Corp Global Search Challenge where participants submitted ideas of the next potential gold targets, and the top 25 finalists were offered $500,000 for their contribution. On the other hand, in open calls with direct rewards, the quality metrics are not clearly defined, the time frame is not clearly communicated, and the reward may be in the form of cash or royalties. For instance, Procter & Gamble's platform "Connect & Develop" is used to solicit advertising ideas in return of a financial award that ranges from $10,000 to $100,000. However, in an open call with indirect benefit, participants indirectly benefit from submitting their ideas by gaining satisfaction when the company implements their ideas. An example of this is Dell's Idea

Storm platform that seeks new product ideas from the crowd without rewarding them financially. Finally, participants usually complete micro-tasks in return of monetary rewards as part of a formalized process or platform. For instance, Amazon Mechanical Turk is a website that gives businesses the capability of hiring participants to perform certain tasks. Thus, the purpose of the crowdsourcing platform also can either be to increase the productivity, quality, innovation, product knowledge of a company, or emphasize and raise awareness about a company's brand (Forbes and Schaefer, 2018).

Implementation Type	Organisational Purpose
• Crowdsourcing Contests	• Increase productivity
• Open Calls with direct rewards	• Solicit Innovation
• Open Calls with indirect benefits	• Improve product knowledge
• Microtasks	• Raise Brand Awareness
	• Improve/Build Product

Figure 1: Implementation Type and Purpose

C. Crowdsourcing and Product Development Process

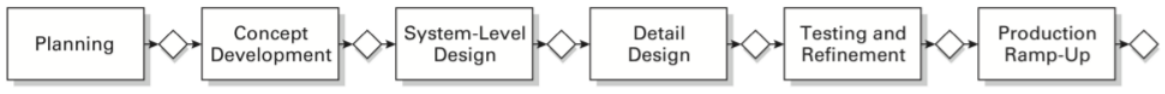


Figure 2: Product Development Process

The traditional product development process involves planning, concept development, system level design, detail design, testing and refinement, and production as shown in Figure 2 (Karl Ulrich & Steven Eppinger, 2015). During the planning phase, a market research is conducted, and an opportunity is identified. In the concept development phase, market needs are identified, and various concepts are generated where only once concept is chosen. In the system level and detail design phases, the product architecture is defined including the decomposition of the system and the subsystems and the determination of complete specifications. Finally, during testing and production, the construction of preproduction versions of the product and final version takes place. Crowdsourcing can be applied at all these different phases of the product development process through an online platform collecting efforts from the crowd. However, idea crowdsourcing could be particularly applied at the planning and concept development phases where the crowd brainstorms solutions to problems and is a natural need specifications producer.

D. Problem Statement

Companies face today big problems that are related to the huge and voluminous amount of data in crowdsourcing platforms, and their incapability of dealing and benefiting from this data to get the needed results. This caused skepticism toward the importance of crowdsourcing. Previous literature (Poetz & Schreier, 2012) shows the negative attitude toward the effectiveness of crowdsourcing. Logically, an idea in crowdsourcing platform is chosen according to different factors like votes, comments, points earned, feasibility, and alignment of an idea with the business strategy which might seem like a simple task. However, the sheer volume of data in crowdsourcing platforms make selecting the "right" idea a tedious process. This data that is submitted in crowdsourcing platforms is divided into three main categories: (1) contributions such as ideas, prototypes or suggestions submitted by participants, (2) metadata including evaluations, comments, or tags, (3) and finally data about contributors like personal characteristics, activity, social network, and preferences (Zhang et al., 2016).

This volume and variety of data pose a difficulty in the evaluation process because it is time consuming and because crowdsourcers may not have the full capability to assess all types of contributions. Thus, many ideas and contributions tend to get unheard and do not progress further in the ideation selection process. To filter ideas and contributions and simplify the evaluation process with the presence of the high volume of data that is being generated at a very fast rate in crowdsourcing platforms, companies currently undergo the following strategies of evaluations:

1. *Experts Versus Crowd*

Some companies hire experts to filter the best ideas which is a costly and a lengthy evaluation process that is sometimes even infeasible. Also, evaluating ideas by experts usually cause the company to miss good solutions due to the huge cognitive load on these experts in reviewing all these submissions in a very short time. For instance, Google hired more than 3,000 employees to evaluate over 150,000 ideas for its Google 10th to 100th project which put them 9 months behind their schedule. Also, IBM employed 50 senior executives to evaluate all 50,000 ideas that its employees submitted in one of its innovation contests (Blohm et al., 2013). Nonetheless, the change.gov website shut down because the huge volume of submissions were overwhelming to the staff that they found a big difficulty to analyze these contributions (Klein & Garcia, 2015). In addition, it has been estimated that it takes about $500 and four hours to evaluate one idea in a Fortune 100 company (Robinson and Schroeder, 2009).
Other companies resort to other crowdsourcing evaluation techniques based on involving the crowd in the evaluation process. One strategy in doing this is to use majority voting similar to Facebook "likes" where every participant can simply vote for

the ideas they find as the best, and ideas are then listed in a ranking order. Many companies like OpenIDEO and GrabCAD use this strategy in their crowdsourcing projects. This strategy is evidently faster and more cost effective since it involves the opinions of a diverse group of different backgrounds. However, this strategy also faces many struggles. One of them is due to the phenomena of “rich get richer” where people tend to vote positively for ideas that are already on the top of the list and “seem” better and tend to disregard other ideas that they cannot see or have to do the extra effort of scrolling to see.

2. *Author Vs Content*

Some companies evaluate ideas according to content while others do that according to the author. Some crowdsourcing platforms prefer choosing ideas of users who have a higher reputation than others or eliminate certain contributors based on their previous “bad behavior”. However, this poses the problem of actually missing ideas from “infamous” authors. Others base their evaluation process on the content rather than the submitter and thus disregard ideas that are not well- structured or well-written. This also poses a problem when a very promising idea is not taken into consideration because it was not well-written.

3. *Machine Vs. Human*

Some companies use a machine-based approach, while others use a human-made approach in their evaluation process. For machine-based approaches, companies rely on the semantics of text mining to evaluate ideas and contributions similar to the idea of ETS grading that processes papers to grade them. These methods study variables like length specificity, writing style, readability, spelling mistakes, context, and also data related to crowd activity. However, machine-based approaches miss judging all aspects of an idea because this requires a human intervention which constitutes knowledge from different fields. When basing their evaluation on a human-made approach, companies either select ideas after evaluating each idea on a predefined scale or select ideas who are all in accordance with specific evaluation guidelines they put and then choose according to their preference.

However, filtering ideas is not the only problem related to crowdsourcing platforms. Companies also should possess the ability to assimilate and then dissimilate data into actual decisions after studying the resource allocation, economic feasibility, and estimates of potential revenues (Blohm et. al, 2013). The ability of effectively matching the generated crowdsourcing ideas with the company’s strategy for product development and placing them in the pipeline is also a struggle for companies today who are constantly striving to stay ahead in the competition field. These companies need to map ideas to actual decisions regarding what to select and place on the road map for implementation and what to neglect based on a comprehensive study of internal factors and resources.

E. Definition of data science, machine learning, and predictive analytics and their relation to idea crowdsourcing in product development:

Data science is another term for discovering knowledge from data to perform analysis and predictions, and it exists with other terms like machine learning and predictive analytics in the data discovery ecosystem (Wimmer & Powell, 2016). Machine learning represents a critical aspect of data science and is the process of programming computers to learn from existing data to create predictions and classifications or make decisions. In addition, a term that is coined with machine learning is predictive analytics which is the ability to create predictions based on statistical methods such as regression. These significant terms are being utilized recently to analyze and improve systems and strategies of businesses and companies. Therefore, data science, machine learning, and predictive analytics are major keys to understanding and analyzing idea crowdsourcing platforms today. With the tools offered by data science, we will be able to discover the most wanted and needed ideas that align with business strategy and internal resources of a company.

The thesis discusses the problems that companies face today regarding the management of crowdsourcing platforms with the sheer amount of data and variety, analyzes a use case platform with all its data, and finally maps the results obtained to the resource allocation decisions a company must take with this analysis. The thesis thus filters the most promising ideas in a crowdsourcing platform to include in the product roadmap by considering not only the feedback of customers but also a company's strategic decisions.

The rest of the thesis proceeds as follows. In the next chapter, we present the literature review regarding this subject. Then, we propose our methodology of how we to predict winning ideas in crowdsourcing platforms and how to assign them to the product development releases. After that, we apply our methodology in a use case crowdsourcing platform and analyze the results. Finally, we conclude our thesis with suggestions and recommendations for future research.

CHAPTER II

LITERATURE REVIEW

The idea of effectively exploiting crowdsourcing platforms is not entirely a new concept. In this thesis, we will explore further utilizing crowdsourcing platforms to assist management in resource allocation decisions during the ideation or release planning phase of the product development process. The following literature review discusses the algorithms, methods, and tools that were already used to address the problems that crowdsourcing platforms or in general online communities with huge amount of data suffer from. The literature review is divided into different sections as per Figure 3.

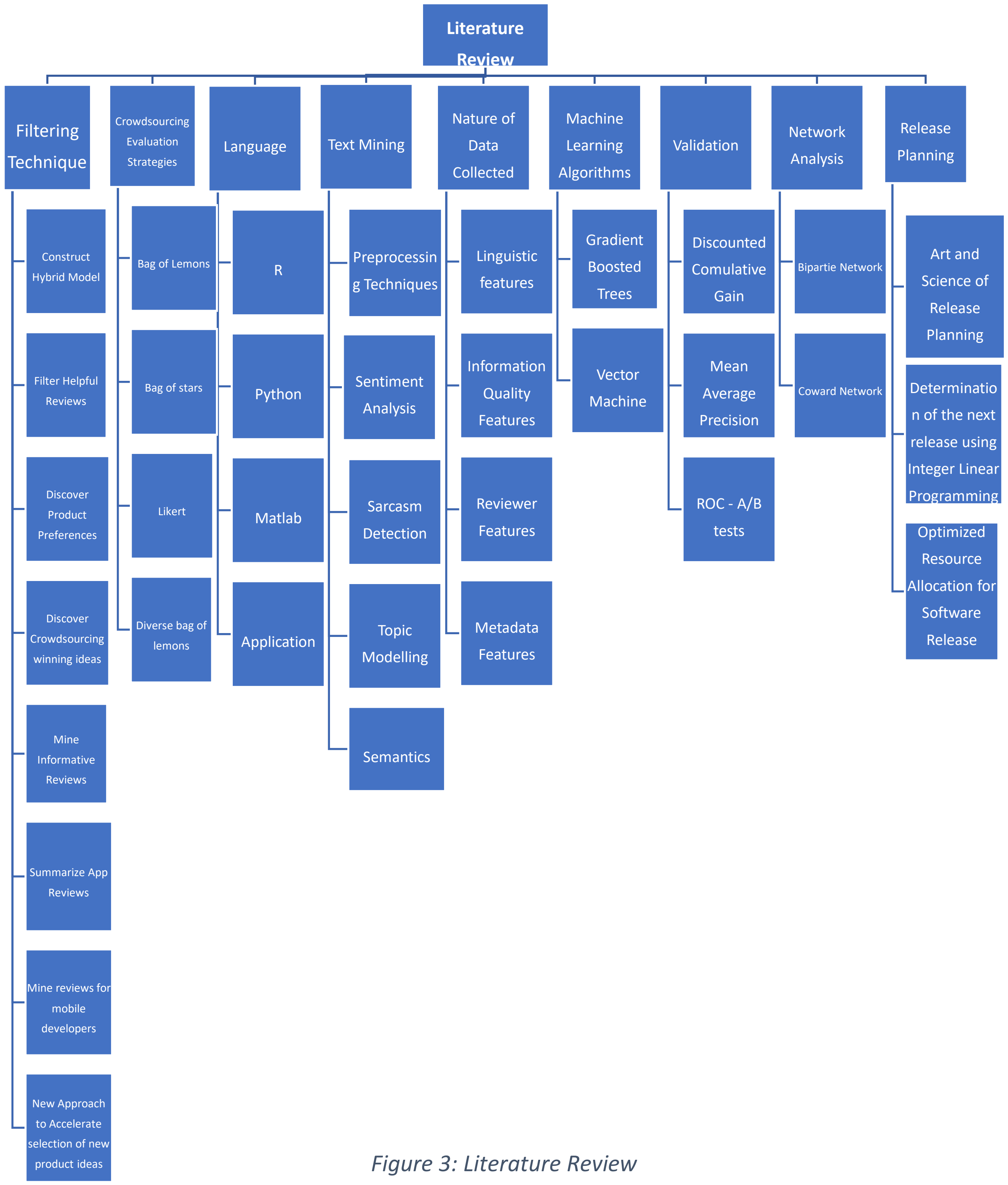


Figure 3: Literature Review

A. Filtering Techniques

Filtering techniques that narrow down a collection of items into a smaller subset of items have been widely implemented. Different filtering algorithms and techniques were used to filter winning and useful ideas, requirements, and reviews from online communities, platforms, and systems in the literature review.

1. *Construct Hybrid model that combines humans and machine learning to evaluate crowdsourcing contributions in Idea Contests*

Dellermann et al. (2018) addresses the problem related to the challenges that come with evaluating ideas generated from crowdsourcing platforms based on machine learning techniques alone or based on crowd evaluation methods alone. They propose a hybrid model that combines machine learning techniques with crowd evaluation ones to provide the benefit of both sides. Machine learning provides valid accurate results when processing information, and on the other hand the human decision makers are better at assessing a context according to their intuition or gut feeling. Thus, the paper showed how essential is to propose a model that combines machine learning with crowd-based evaluation. It is important to note that machine learning techniques include those that evaluate (1) textual contributions like length, specificity, completeness, and writing style or (2) representational submissions like readability and spelling mistakes or (3) crowd activity like page views, votes, and comments. Crowd based evaluations include (1) crowd voting mechanisms and (2) rating mechanisms from the crowd. The paper proposed a design science research project to provide prescriptive knowledge on how to construct the hybrid model that combines machine learning techniques with crowd evaluation ones. This iterative design research cycle methodology interpretation is illustrated in Figure 4.

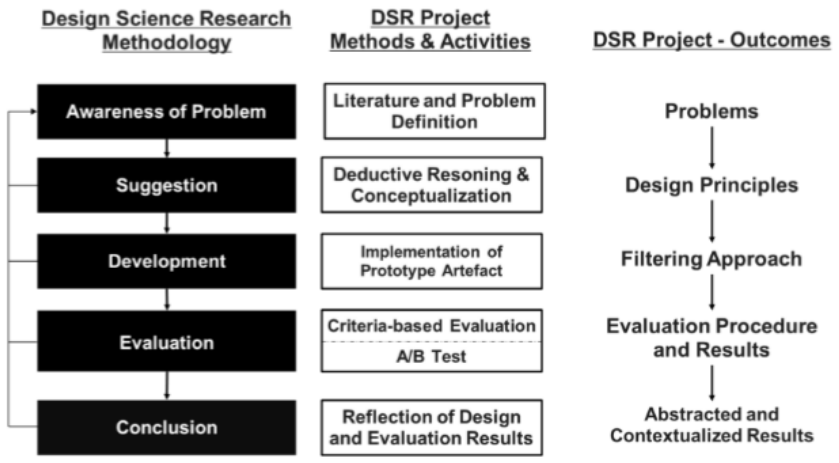

Design Science Research Methodology
DSR Project Methods & Activities
DSR Project - Outcomes
Awareness of Problem
Suggestion
Development
Evaluation
Conclusion
Literature and Problem Definition
Deductive Resoning & Conceptualization
Implementation of Prototype Artefact
Criteria-based Evaluation
A/B Test
Reflection of Design and Evaluation Results
Problems
Design Principles
Filtering Approach
Evaluation Procedure and Results
Abstracted and Contextualized Results

Figure 4: Design Research (Dellermann et al., 2018)

Based on Figure 4, the problems related to the current evaluation techniques were first identified, then based on deductive reasoning, five design principles were concluded to form a novel prototype version of evaluating ideas. Finally, this novel evaluation technique was put as a future milestone to be compared with the other two already mentioned evaluation techniques alone by conducting an A/B test which is a split testing method that compares different methods to understand which performs better.

Four problems were identified in idea crowdsourcing platforms (Dellermann et al., 2018):

(1) Because of computational machine learning algorithms that are based on idea shortlisting, sometimes ideas are marked as innovative when in fact they are not truly innovative.

(2) The huge number of contributions poses a cognitive load on crowd-based evaluation techniques rendering them problematic.

(3) Ideas represented at the top of a page always receive the greatest attention (and perhaps votes) which may give potentially fake positive feedback.

(4) Not every participant using the crowdsourcing platform is equally appropriate to evaluate an idea.

Depending on these four problems, five propositions were discussed in the paper to solve these problems. The propositions are summarized in the Figure 5. As a summary, the idea behind the propositions is to determine the topics of ideas submitted on a crowdsourcing platform and match them with participants based on their profiles and previous submissions for evaluation. In this way, appropriate experienced participants can selectively evaluate ideas based on this match.

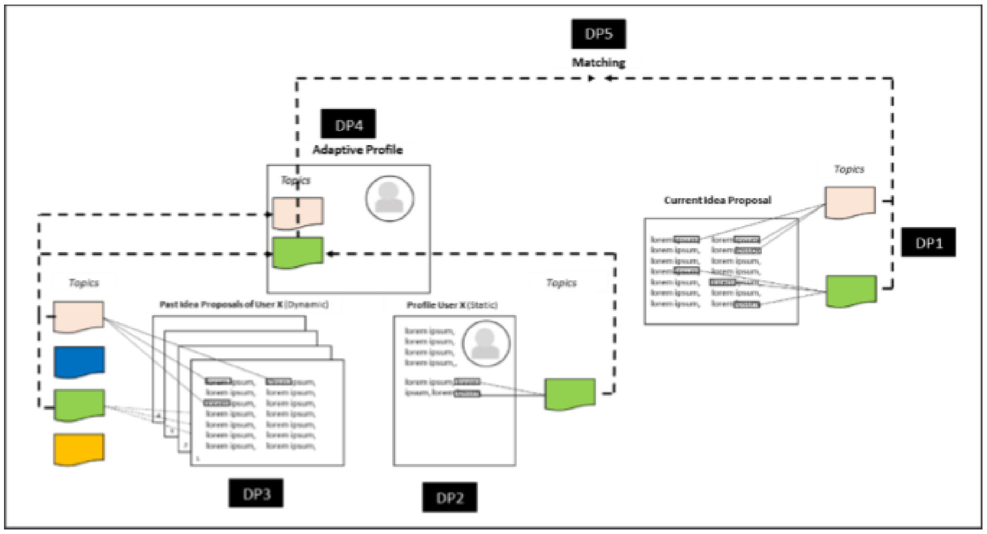


Figure 5: Proposed Hybrid Filtering Approach (*Dellermann et al., 2018)*

These propositions were then given to a group of experts in community and system engineering to evaluate them based on completeness, understandability, fidelity, applicability, and clarity, and the p-values of their opinions then showed that the design principles are clear for the group of experts and could be beneficial. This filtering strategy defined by the design principles will then be compared with the two other existing ones (crowd based or computational alone) (A/B tests) and the concept of ROC-curve will be used to determine if the evaluation technique is appropriate or not.

Although the paper only relied on qualitative design science research (DSR) method to conclude findings on crowdsourcing platforms, the paper addressed the importance of having both evaluation techniques; the computational and the crowd voting; which makes it necessary for our model to adopt them both.

2. *Determine requirements from helpful online reviews to be used in the product development cycle*

Compared to traditional offline or paper and pencil market surveys, online reviews provide a way for product designers to understand the requirements of their customers in less amount of time and at a much lower cost. Zhang et. al (2016) proposed a model to automatically filter helpful online reviews in the product design process and analyze them using the KANO model (usually used for standard questionnaires) to develop appropriate design strategies. The model proposed thus combined the notion of big data and classical management models to solve big data problem in ecommerce. Figure 6 shows the model that was developed to determine the helpfulness of reviews and create product improvement strategies. The model is divided into 3 parts: Data collection and preprocessing though lexicon building, determining helpfulness of an online review through model training and prediction, and creating product improvement strategies. A Chinese commerce JD.com was used as a use case to apply the model on where phone reviews were extracted. Data about each review was collected, metadata (date of the review posted, replies, and votes), and reviewer data (name and grade). Phone product attributes were then identified by Dirichlet Allocation and Page Ranking which groups data into topics. Some of these attributes were determined to be related to edition, CPU, music, battery, and screen. Sentiment analysis was also applied on reviews, and helpfulness score was then determined based on linguistic, metadata, and reviewer features. The function that was used to determine helpfulness was: y = a (Text features like Number of words/sentences) + b (reviewer features like grade of reviewer) + C (metadata features like number of replies and stars). Two product designers trained the model by identifying 20% of reviews as helpful or not. Through the KANO model, the different attributes were extracted and were identified as one dimensional, reverse, attractive, or must be attributes. Must be attributes were identified to be battery life, camera, signal, and appearance. QFD was then used to map the product attributes to engineering features, and the ranking of the features easily helped in the product design process to assign utility factors to features.

Thus, the three parts model inspires our approach in creating more than one filtering stage to determine winning ideas to be implemented in the product development process.

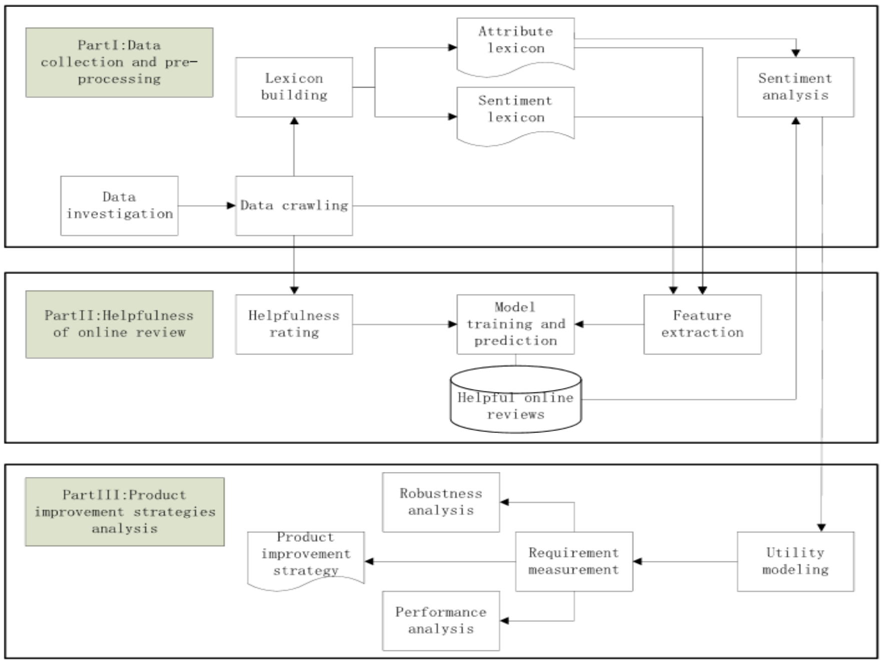


Figure 6: Process to Product Improvement Zhang et. al (2016)

3. Discover Product Preferences in Ubiquitous Social Media

Tuarob & Tucker (2016) discuss a methodology based on POS tokenization, sentiment analysis (people's opinion and emotional strength toward a product feature), Dirichlet Allocation (which divides the social media messages into attributes), and consumer reports (which validates the preferences) to extract customer preferences on automobile products from social media. Instead of resorting to traditional methods like group studies in which random users are selected to give their opinions which is costly and only considers a small sample, companies today resort to social media platforms like Facebook, Instagram and Google+ to solicit product features because they demonstrate high degree of creativity and openness and heterogeneity.

The paper shows the importance of sentiment analysis algorithms in predicting not only the most important features but also product demand. Thus, sentiment analysis should take part of our methodology to find best ideas in crowdsourcing platforms.

4. Capturing Winning Ideas in Online Design Communities

"In a sea of thousands, how can someone sift through ideas to find high quality submissions they can build upon or be inspired by?" Ahmed & Fuge (2017) propose a model to managers on how to effectively manage large collaborative online communities and enable them to filter high quality ideas submitted. The model combines the strengths of human and automated techniques (discounted cumulative gain- DCG) to find quality ideas in the submission haystack. The model thus uses important factors like community feedback, idea uniqueness, and text features to effectively rank submissions by quality. The model was demonstrated on OpenIDEO which is a collaborative online community where designers are awarded for winning design challenges. A training set of winning ideas was partitioned according to a set of features like readability, coherence, and semantics. For each idea, the data describing the idea, the number and timestamp of any comments left on the idea, and whether the idea got through to the evaluation or winner stage was captured. Data concerning the text readability and complexity of ideas was analyzed using the python readability package. Text Cohesion was measured using the online Coh-metrix. Linguistic Inquiry and Word Count (LIWC) was used to determine the psychological associations. Also, idea uniqueness was determined through network models like TextRank. The results of this set were then used to predict ideas on a test set using

Gradient Boosted Trees. The results showed that winning ideas had more comments on them. Also, winning ideas have unique topics and are long with larger vocabulary. They also showed how location of the author had no effect on advancing an idea to later stages.

Thus, the model proposed helps solve problems related to human bias and machine discrepancy since it combines human and automated techniques to find quality ideas. Thus, the model tackles problems like the rich get richer “members only visit and vote on already highly voted ideas thus biasing vote counts” which is caused because of members’ votes or the sparsity problem “members may still never assess some ideas due to fatigue or lack of bandwidth, leaving a large number of ideas without any ratings.” It also shows how machines alone including text semantics are insufficient for determining quality ideas.

The paper emphasized our research on the importance of determining the several key factors included in filtering quality proposed ideas like votes and text in ideas.

5. *Mining Informative Reviews for Developers from Mobile App Marketplace*

According to Chen et. Al (2014), the ability to collect and digest constructive reviews from app marketplace is challenging for two reasons:

(1) Popular apps like Facebook receive around 2000 reviews per day, so digesting these reviews is challenging on the cognitive level.

(2) Only around 35.1% of the reviews in the marketplace contain helpful information to improve reviews.

Thus, the paper discusses an innovative app review mining technique “AR Miner- application review” that successfully (1) extracts informative reviews and filters irrelevant ones, (2) groups reviews into categories, (3) prioritizes reviews by a ranking scheme, and (4) presents groups of informative reviews in an intuitive visualization manner. This technique is finally evaluated on 4 apps on the app store to test its efficiency and effectiveness.

Information about reviews (text review, timestamp, and rating) were collected and classified as informative and non-informative reviews. Informative reviews were reviews that show the need for adding or changing a new feature and were important to developers, and non-informative reviews were reviews that were general descriptions with pure emotional expression. Preprocessing was applied first to divide reviews into sentences and treat reviews in sentence level granularity. Filtering was then applied to classify reviews into informative and non-informative ones using an Expectation Maximization for Naive Bayes (EMNB) semi supervised machine learning algorithm since it outperforms other machine learning algorithms in text classification. To group different reviews, topic modelling was used including Latent Dirichlet Allocation (LDA) and Aspect and Sentiment Unification Model (ASUM). Prioritization of these reviews was then done using a ranking technique. This technique was built based on group ranking and instance ranking both of which relied on the rating provided (groups with lower rating were prioritized) and time submitted (fresh reviews about new bugs were given the priority). The model was then tested on 4 apps of different backgrounds including Facebook and Temple run. In testing the validity, precision and recall rates were calculated. Also, NDCG Normalized discounted gain for measuring the quality of top k ranking results was used.

The methodology thus obtained in this paper highlights the importance of preprocessing our ideas before evaluating them which should be adopted in our model.

6. *Summarizing App Reviews for Recommending Software Changes*

Di Sorbo et. al (2016) discuss SURF (Summarizer of Reviews Feedback) which provides a novel approach to filter and summarize the enormous number of reviews that developers of popular apps need to manage. These reviews are usually used by the app developers to perform maintenance and evolution tasks. The algorithm was tested on 17 mobiles apps and showed high accuracy in summarizing reviews and reducing the time required by developers to go manually over the reviews. Since developers of popular apps receive hundreds of reviews every day, it hard on them to filter and read every feedback request and aggregate them together as well on Apple Store and Google Play. These reviews may include bugs, summaries of user experience with a certain feature, request for enhancements, and ideas for new features. The approach determines for a large number of reviews the specific topic discussed in the review (UI improvement, security...), identifies the maintenance task to perform to address the request (fix bugs, enhance features), and presents this information to the developers in a structured agenda form with actionable items. Therefore, the collected reviews were identified according to intention (information giving, seeking, feature request, problem discovery, or other) and topics (app, GUI, contents, pricing, feature, improvement, security, download, model, company) to help developers in taking the necessary actions. The model consisted of the following steps:

Data collection:

The review information text, date of the review, the star rating given by the reviewer, the title, the app version, and the handler who posted the review were collected in XML format.

a. Intent Classification:

Intent classification was then applied. This combines NLP, sentiment analysis, and text analysis techniques through a ML algorithm. The reviews were then categorized according to intention (information giving, seeking, feature request, problem discovery, or other).

b. Topic Classification:

Concept dictionaries were identified first to classify reviews into topics. For example, reviews that have the words “orientation” or button showed that they are dealing with GUI. An NLP classifier was then built to specify topics based on keywords they consists. To make the dictionaries exhaustive, wordNet was used. To stem sentences to the original sentence, Snowball stemmer algorithm.

c. Sentence Scoring:

Sentences were scored then depending on the topic and the intent to provide recommendations for developers on what to do next.

7. *Mining Informative Reviews for Developers from Mobile App Marketplace*

It is very important to stay ahead of competition by constantly analyzing the feedback of customers for mobile product development (Ciurumelea et al., 2017). To help developers benefit from the big number of reviews about their applications, a User Request Referencer (URR prototype) was built to classify reviews into categories according to a predefined taxonomy that was built manually and also recommend for a particular review what are the source code files that need to be modified. Reviews were extracted of different applications and text preprocessing was applied including stop words removal, punctuation removal, and reducing words to their stemmed words, and the model was trained using Gradient Boosted Regression Trees. To be able to localize source code to be suggested for reviews in order to be modified, information retrieval methods were used were source code files were indexed to compute the textual similarity between user reviews and source code. To validate the results, the accuracy and precision of the classifier were determined, and two external evaluators took a post experiment survey to illustrate the importance of the tool and confirm that it would save time for developers.

This paper inspired our approach to include qualitative assessment to validate our proposed model.

8. *A new approach based on Soft Computing to Accelerate the Selection of New Product Ideas*

To survive in the competitive market today, it is important to produce high quality products in short development cycle times which makes the need for an efficient product development process a vital one. Buyukozkan and Feyzioglu (2003) have proposed a methodology to improve the decision making for the development of new products while minimizing the uncertainty involved. Uncertainties include technical, management, and commercial sources that are both external and internal to a company. To help identify the products needed for implementation and their order, they constructed their model based on fuzzy logic, neural networks, multi-criteria decision making, and artificial intelligence techniques. Their model consisted of 2 stages where the first stage allows practitioners to roughly identify good ideas from bad ones based on previous experience using Artificial neural networks and fuzzy logic, and the second stage allows practitioners to analyze in detail a more shortened list and is based on multi-criteria decision making.

B. Crowdsourcing Evaluation Strategies

To determine if an idea is a winning idea or not, different evaluation strategies can be adopted. For instance, participants can like, vote, dislike, downvote, or rate an idea. The efficiency of these evaluation strategies in a pool of thousand ideas is a challenge, however. Thus, many crowdsourcing evaluation strategies were proposed in the literature review.

1. High speed filtering using a bag of lemons

Identifying the best ideas in a pool of contributions is time consuming and inefficient. Klein and Garcia (2015) proposed the “bag of lemons” approach that provides an accurate measure of the best ideas in a fraction of time. This approach is based on the idea that the crowd is better at actually eliminating the worst ideas than actually picking the best ideas. A research was conducted on R&D lab members of Fluminense Federal

Univeristy in Brazil that was facing high competition worldwide and wanted to find new innovative productivity enhancement suggestions.

The study conducted by Klein and Garcia (2015) was divided into two phases. In phase 1, an experiment was conducted on an open innovation platform called Deliberatoriumin that allows users to submit ideas in a hierarchy. A contest was constructed with three financial reward prizes. The contest's duration was for one month and collected around 48 ideas from 23 authors all of which were lab members themselves familiar with the lab's challenges and problems. Also, two cons and two pros of each idea were collected.

Then, these ideas were evaluated by experienced research managers who had different backgrounds, gender, and age without knowing the authors of these ideas. The experts reviewed and rated these ideas as good, bad, or average according to three criteria:

a. Cost for implementing the idea
b. Productivity benefit resulting from the idea
c. Time needed to start benefiting from the idea

Nineteen ideas were rated as good by the experts, so the experts did a four round Delphi process to choose the three best ideas out of those 19.

In the second phase, three groups of 20 members each were assembled to evaluate the ideas according to the same criteria that was used by the committee members and without seeing the authors 'ideas. The three groups were balanced according to gender and educational level, and each group used a certain technique to find the best three ideas:

a. Likert

In this technique, the participants rated each idea on a Likert scale from 1 (representing highly not to be selected) to 5 (representing highly to be selected as the best idea)

b. Bag of stars

Each user was given 10 stars in total to distribute on the present ideas. The idea with the greatest number of stars is the idea with the best quality.

c. Bag of lemons

Each user was given a budget of 10 lemons to distribute on the ideas. Thus, an idea with the big number of lemons is actually the worst idea.

All the collected ideas were then recorded and time stamped.

To compare how each idea technique was close to the ideal (experts 'decisions), the standard ROC curve was used which is a useful tool for evaluating and comparing predictive models.

The results showed that the larger the group, the more accurate the best ideas result was.

The results also showed that the raters' demographic criteria had no effect on improving the accuracy. More importantly, the results showed that the BOL (bag of lemons) had the highest accuracy in determining the best ideas, followed by Likert and then BOS (bag of stars).

BOS and BOL required roughly one third the time needed for the Likert approach ($p<0.05$). The difference in time between BOS and BOL was not statistically significant which made sense since people clicked fewer on “more” link to view the pros and cons when they used the BOS/BOL evaluation techniques in contrary to Likert technique.

Thus, in conclusion, the crowd was much (about 60%) more accurate at eliminating bad ideas (BOL) than selecting good ones (BOS) and spent less amount of time in doing so. The challenge to these findings is that they can be applied on a small-scale crowdsourcing platform. What if there were thousands of ideas, how can users use the bag of lemons to identify the worst ideas from these thousand ideas? One suggestion might be to group similar ideas, and then rate the worst cluster of ideas. Another suggestion is to use the group of lemons as the first step in the filtering process instead of having it as the complete filtering process. However, one important limitation in this strategy is knowing the suitable number of tokens to be associated to each user in the evaluation process taking into consideration that this number should adapt to the past contribution of the participant.

2. *Diverse Bag of Lemons (DBLemons):*

Can we overcome the problems posed by the crowd-based evaluation strategies (like crowd voting) which are separating the mediocre from the excellent, and directing attention to certain ideas rather than others? Sadien et al. (2018) discuss a new crowdsourcing evaluation strategy that is based on the concept of a DBLemons (diverse bag of lemons) that solves the problems that some crowdsourcing strategies suffer from. The idea of the DBlemons stems from the notion that the crowd is actually better at identifying the bad ideas from the good ones. This study is a continuation of the previous paper discussed earlier.

In the paper, a dataset of a real-world open innovation problems was created. The dataset was collected from the platform OpenIdeo around women's safety challenges ideas: "How might we make low-income urban areas safer and more empowering for women and girls?" 52 ideas in total regarding the topic were summarized into 150 words each and evaluated by 3 viewers. This dataset was then evaluated according to three techniques: bag of stars (majority voting), bag of lemons and dB lemons of idea diversification.
520 workers were hired to evaluate 3 ideas each according to the Likert scale.

The ideas were evaluated by following these criteria: i) Investment potential, ii) Novelty, iii) Impact potential, iv) Feasibility, v) Scalability, vi) Understandability and vii) Overall feeling. Each idea was evaluated by 3 evaluators. The quality score then for each idea was collected, and the top 30% ideas (16 ideas) were then deduced and was chosen as a golden set to compare the different evaluation strategies.

a. Bag of stars

Each rater got 52 votes to distribute on the different ideas where they cannot use more than one vote on an idea. When the rater sees the ideas to evaluate, he sees them in descending order where he sees the most voted on ideas at the top. After putting his votes, dynamic voting is assured, and the information becomes updated to put back the ones with the greater number of votes back to the top.

b. Bag of lemons

Each user gets a budget of 10 down votes (lemons) and distributes them on the ideas. The ideas then are re-ordered according to the number of lemons received. The ideas with the least number of lemons are displayed first followed by the ones with the lowest number. Similar to bag of stars, according to the number of lemons each vote receives, the list is updated.

c. Diverse Bag of lemons

In this technique, two concepts are combined: diversity and bag of lemons.

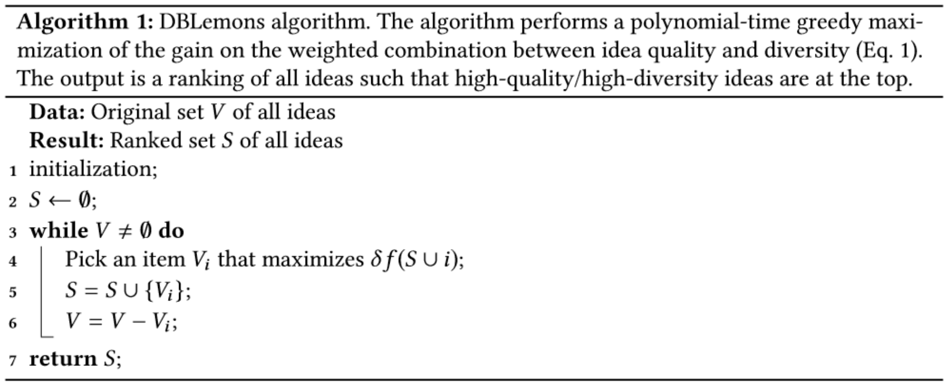
Each participant is given a budget of 10 lemons and are asked to distribute them. From the user's perspective, this technique is similar to the bag of lemons. What is different is what happens behind the scenes. After each participant distributes the bag of lemons, a greedy algorithm then displays the top ideas according to the least number of lemons and most diverse criteria.

Idea ranking is displayed according to a submodular diversity function:

$$f(S) = \sum_{j \in S} W_j + \lambda \times \sum_{c=1}^{K} \sqrt{|S \cap P_c|}$$

The first part of the equation is related to the quality. The higher the quality, the higher the function (number of votes in this case). The second part denotes the diversity. This part is greater when the idea is from a cluster that has not yet contributed to set S. λ represents the value to signify if quality is preferred over diversity.

This function is based on the following greedy algorithm that is used to order the different ideas:



Algorithm 1: DBLemons algorithm. The algorithm performs a polynomial-time greedy maximization of the gain on the weighted combination between idea quality and diversity (Eq. 1). The output is a ranking of all ideas such that high-quality/high-diversity ideas are at the top.

```
Data: Original set $V$ of all ideas
Result: Ranked set $S$ of all ideas
initialization;
$S \leftarrow \emptyset$;
while $V \neq \emptyset$ do
    Pick an item $V_i$ that maximizes $\delta f(S \cup i)$;
    $S = S \cup \{V_i\}$;
    $V = V - V_i$;
return $S$;
```

Figure 7: DBLemons Greedy Algorithm

Λ favors diversity and in the experiment's case, a higher lambda was preferred (In other words, a more diverse idea had more value than an idea with a higher quality).

Results:

The DBLemons technique was then shown in the paper to be the best strategy to evaluate ideas of the three because it considers both diversity and quality in its algorithm and puts more weight on diversity which ensures that all ideas get evaluated.

C. Languages Used in data science

To be able to extract, manipulate, and analyze data, choosing the right Language is crucial in the process.

1. Comparison between Python Vs. Matlab Vs. R:

In today's data driven environment, there are different data mining and analytics programming languages that can perform different functions including visualization, data manipulation, classifications, and analysis. Three of them are MatLab, Python, and R which are used to collect and analyze data for decision making purposes (Ozgur et. al, 2017). Between the three programming languages, Python is easier to learn because the syntax is read like the normal human language and is one of the top coding languages as of 2014. However, visualization in Matlab is user friendly, but the language is used by data analysts mainly for numerical computations. On the other hand, R has an unmatched analytical and statistical power but is much harder to learn as a language than others. Also, Muenchen (2017) explained that R is rapidly gaining a share of the data analytics market.

Because of its relative simplicity and popularity, Python will be used as a starting point in this paper to collect, analyze, and understand data collected from a crowdsourcing platform use case.

2. Application of Python in crowdsourcing platforms

Python and R were both used in previous literature to analyze data in crowdsourcing platforms. Sha et al. (2019) used Python web crawler which is Scrapy to pull data information of contests between 2010 and 2016 from the platform GrabCad which is a website that consisted of contests that offered prizes to designers with the best ideas. They also used statnet package in R to analyze the collected data using exponential random graph models (ERGMs). Also, data concerning the text readability and complexity of ideas was analyzed using the python readability package (Ahmed & Fuge, 2017).

D. Text Mining

To analyze collected data and draw meaningful conclusions, text mining which involves a series of steps including data preprocessing, sarcasm detection, and topic modeling are necessary for data analysis.

1. Text Preprocessing

It is important prior to processing and analyzing the data in a textual context to apply preprocessing techniques. These techniques eliminate unnecessary noise that impacts the smooth analysis of data extracted from crowdsourcing platforms (Kannan & Gurusamy, 2014). These techniques can be summarized in the three categories:

a. Tokenization:

This is the process of dividing a stream of sentences and words into tokens. These tokens then are used for further text mining and parsing. The goal behind this technique is to remove punctuation marks, and other characters like hyphens, brackets, etc…

b. Stop Word Removal:

Stop words are commonly used words like "and", "are", and "this" that occur frequently in sentences. Because of their nature, these words are not useful when documents are classified and thus are always preferred to be removed.

c. Stemming:

Stemming is a widely used method when retrieving information which groups the variant forms of a word into a common representation which is referred to as a stem. For example, the words “presentation”, “presenting”, and “presented” could all be grouped under the same word which is present. This method suffers from two problems which are over stemming and under stemming. Over stemming occurs when there are two words are grouped under the same root but instead should not. Under stemming occurs when two words that should be stemmed under the same root are stemmed under two different roots.

d. <u>Application of preprocessing techniques in evaluating and analyzing tweets (online communities similar to crowdsourcing platforms)</u>

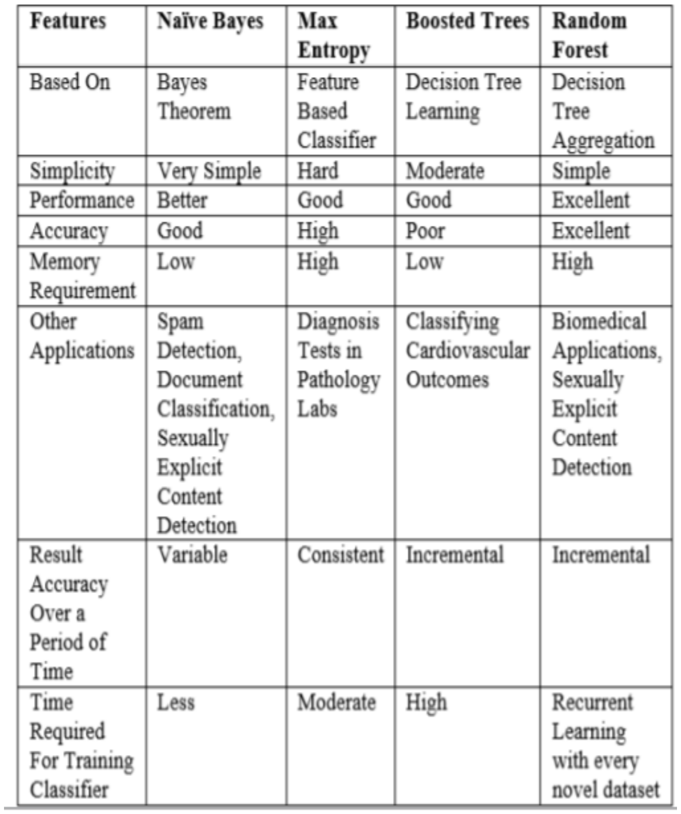
In Tuarob et al. (2018), lowercasing was applied and hashtags and stop words were removed. Misspelled words were intentionally preserved in addition to emoticons like [-_-]. These methods for text preprocessing were utilized before processing the data in tweets and determining sarcasm in them. Also, to remove the noise found in tweets collected from twitter, preprocessing techniques were applied in (Mehndiratta et al., 2017). Only tweets which were created in specific locations like USA or UK were extracted because these locations allowed the collection of English only related tweets rather than any other language. In addition, short tweets with 3 words or less and retweets were filtered out.

2. Sentiment Analysis

a. <u>Algorithms:</u>

Gupte et. al (2014) discussed and compared different machine learning techniques for sentiment analysis. These techniques include Naïve Bayes, max entropy, boosted trees, and random forest algorithms. These classifier techniques are different according to their complexity, performance, accuracy, and memory requirement. It is shown in Figure 8 that

Random Forest is with the highest accuracy and best fits for sentiment analysis although it has a high learning time. However, Bayes classifier is preferred if power and memory are issues that need to be taken into consideration. In addition, boosted trees is preferred if an average classifier is needed.



Features	Naïve Bayes	Max Entropy	Boosted Trees	Random Forest
Based On	Bayes Theorem	Feature Based Classifier	Decision Tree Learning	Decision Tree Aggregation
Simplicity	Very Simple	Hard	Moderate	Simple
Performance	Better	Good	Good	Excellent
Accuracy	Good	High	Poor	Excellent
Memory Requirement	Low	High	Low	High
Other Applications	Spam Detection, Document Classification, Sexually Explicit Content Detection	Diagnosis Tests in Pathology Labs	Classifying Cardiovascular Outcomes	Biomedical Applications, Sexually Explicit Content Detection
Result Accuracy Over a Period of Time	Variable	Consistent	Incremental	Incremental
Time Required For Training Classifier	Less	Moderate	High	Recurrent Learning with every novel dataset

Figure 8: Sentiment Analysis Classifiers (Gupte et al. , 2014)

b. Application of sentiment analysis in online communities similar to crowdsourcing platforms

Sentiment analysis was applied in different articles tackling online communities. For instance, Taurob & Tucker (2016) used sentiment analysis to understand customer demand. According to the paper, the collected sentiment from social media can help determine the demand and thus the sales of a product. 800 million tweets about automobile products created in USA from march 2011 to Sep 2012 were collected, and sentiment analysis was then applied to these tweets to determine their polarity by an algorithm proposed by (Thelwall et al., 2010). The polarity of each tweet message was expressed as [number1, number2] where number1 represented the score of positive sentiment and number2 represented the score of negative sentiment, and the two numbers ranged from 0 to 5. The purpose behind placing two numbers in the sentiment polarity was to determine if sarcasm existed in the tweet message and to understand also if two sentiments co-existed at the same time. Emotional strength was then calculated by using the following formula: negative score – positive score. At the end, the demand for a product was calculated as the number of positive messages (positive(s)) about a certain product S. This demand was compared to the actual sales as informed on Goodcarbadcar.net of these products, and results were found to be compatible with the calculated demand. The paper then concluded that temporal demand of product could be quantitatively captured from social media and then be used by designers to manage their production levels. The same sentiment analysis method was also used in (Taurub et al., 2018) as a building step in creating an algorithm that determines sarcasm in tweets related to products about phones. Around 20,000 sarcastic tweets and 100,000 non-sarcastic tweets about phones were collected over a period between June-July 14, and sentiment analysis was applied to these tweets.

On the other hand, in (Zhang et al., 2016), reviews of phones from one of the largest electronic marketplaces in China; JD.com, were collected, and sentiment analysis was then carried out on these reviews but in a different approach. Instead of using the overall rating which is 1 to 5 stars present on the ecommerce that could depend on how customers viewed their perception of these ratings, NLP was used to infer the sentiment of text on a three-point scale: 1 representing positive sentiment, -1 representing negative sentiment, and 0 representing neutrality.

Sentiment Orientation was also utilized in Zhang et al. (2011). Reviews about SLR cameras and TVs with price range of 500$-700$ from Amazon.com were split into sentences using MXTERMINATOR, and the sentences were checked if they were positive or negative by marking them using POS tagger to label each word in the sentence. A set of words of sentiment orientation were added from Mpa cite corpus and WordNet to get a total of 1974 positive words and 4605 negative words. To identify the sentiment of the sentence, the sentence which was composed of words was compared to the formed set of positive/negative words. In addition, the possibility of having a negation in a review was considered (this is not a good camera) which reverses the orientation of a sentence.

3. *Sarcasm Detection:*

a. <u>Coward Network</u>

Ivanke & Pexman (2003) identified sarcastic sentences in linguistics to consist of 6 main tuples: s = speaker, h = listener, c= content, u= utterance, L = Literal proposition, p'= Intended Proposition. This can be translated into "Speaker S generates an utterance u in Context C meaning proposition p but intending that hearer H understands p'." In social networks, sarcasm has become a social norm where roughly 22.75 % of social media is sarcastic. Information collected from social media that contain sarcasm is considered implicit information in which the meaning is intended but not directly stated. Algorithms involving natural language processing usually just consider data that is explicit and consider implicit data as irrelevant or misinterpret it (noise). An example of an implicit message is “I love when my blackberry screen freezes, I definitely do not like to get an iPhone”. However, traditional natural language processing algorithm fail on social media data because social media contains a lot of noise like grammatical mistakes whether intentional or unintentional, typos, and symbolic words like “lol” that are not well-formed and grammatically correct. Taurub et al. (2018) described a model based on heterogeneous coward network patterns in order to translate implicit messages (sarcastic) in social media to explicit ones with direct meanings. The model is described in the Figure 9.

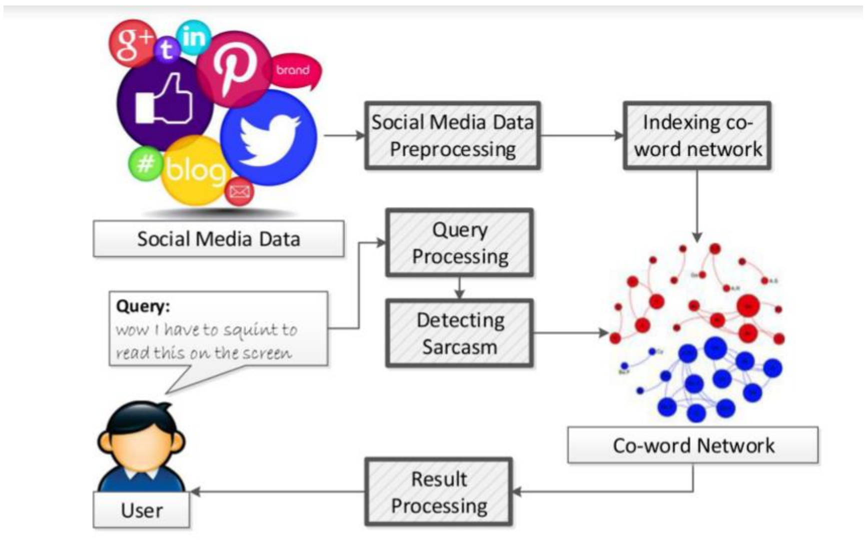


Figure 9: Detecting sarcasm using coward networks (Taurub et al. 2018)

b. Winnow Classifier

Liebrecht et al. (2013) have developed a predictor to detect sarcasm in Dutch tweets based on the winnow classifier algorithm. **The winnow classifier algorithm** is a machine learning technique that is usually given a sequence of positive and negative examples to train it to predict and classify novel examples if they were in the target class (positive) or not (negative). In the field of machine learning, it is a type of a linear classifier that uses the characteristics of an object to determine to which class it belongs. They trained their model by collecting tweets with the hashtag #sarcasm and then tested their model on tweets they collected on a specific date. They concluded that mostly people are sarcastic about topics that include the weather, school and related subjects, social media itself, sports, and celebrities. They also showed that the use of markers (hashtags) such as #LOL, #jk, #sarcasm, #humor, #NOT were also strong indicators for sarcasm. In addition, they revealed that around 94 % of sarcastic tweets mostly included positive exclamations like wow, yes in addition to strong intensifiers like amazing, soooo, and veery words.

c. <u>Deep Convolutional Network</u>

Mehndiratta et al., (2017) has proposed a technique to detect sarcasm using **deep convolutional neural** networks which focused on skip gram technique to convert words into vectors. This approach produced results of overall accuracy of 89.9%. Convolutional neural network is a deep learning algorithm that takes an input image and differentiates the different aspects in this image.

d. <u>Hashtag Tokenizer</u>

Maynard et al. (2014) designed a hashtag tokenizer for their solution and considered that sarcasm found within hashtags can be detected more easily. They also showed that sarcasm cannot be only considered as negation. Some tweets are straightforward like “this project is great”, and when the negation is applied, the sentence becomes “this project is bad” which shows the intended meaning, but other sentences are not as simple. For the tweet, “I am not happy that I woke up at 5:15 this morning. #greatstart #sarcasm”, the negation is not the meaning that is intended, “we are happy to wake up”. Also, if the sentence “You are really mature #lying #sarcasm” is negated, the intended meaning will be correct but negating the hashtag will not be.

4. Topic Modelling

a. Algorithms

Topic modelling is used to classify text in a document to a topic that appears abstractly in a document. There are different algorithms in topic modelling including Latent Semantic Analysis (LSA), probabilistic Latent Semantic Analysis (pLSA), and Latent Dirichlet Allocation (LDA) (Rajasundari et al., 2017). LSA’s goal is to find the meaning behind the words about the different documents. pLSA is used in information retrieval and is an automated document indexing. Also, LDA is an example of topic modeling that builds a topic per document and words per topic model.

b. Application in online communities

Zhang et al. (2016) used Latent Dirichlet Allocation to mine customer requirements from online reviews under different categories. First, reviews from one of the largest electronic marketplaces in China; JD.com, were collected about 757 phones. Then, attributes were identified in these reviews where POS tagging was applied, and a Latent Dirichlet Allocation and page rank were then utilized to rank the different terms in the reviews based on the frequency and the relationship between the terms. This enabled filtering the terms into 15 categories of features for phones including the screen, music, battery, and CPU.

5. Readability, Coherence, Semantics

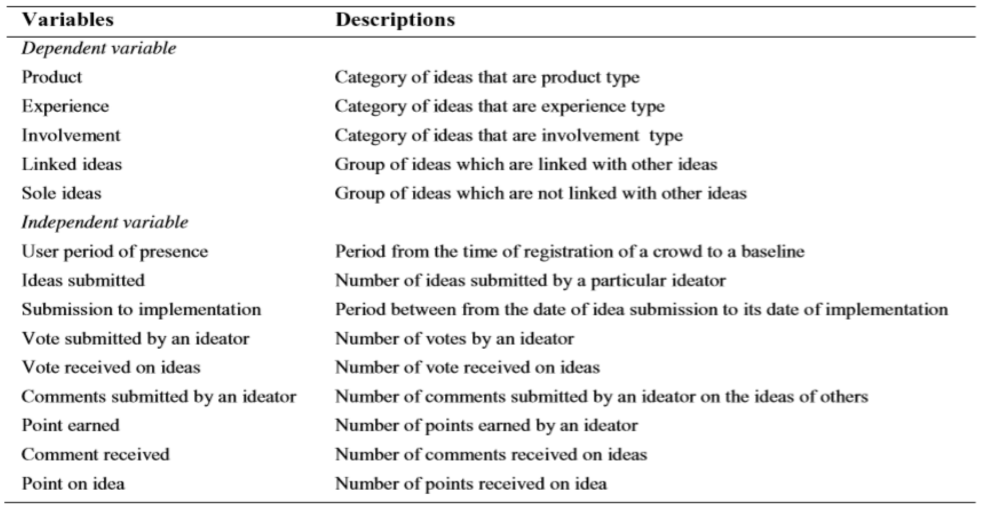
"In a sea of thousand ideas, how can someone sift through ideas to find high quality submissions they can build upon or be inspired by?" Ahmed & Fuge (2017) proposed a model to managers on how to effectively manage large collaborative online communities and enable them to filter high quality ideas submitted using features like readability, coherence, idea uniqueness, community feedback, and semantics. The model was demonstrated on OpenIDEO which is a collaborative online community where designers are awarded for winning design challenges and for each idea, the data describing the idea, the number and timestamp of any comments left on the idea, and whether the idea got through to the evaluation or winner stage were captured. To analyze the different features of data, data concerning the **text readability and complexity** of ideas was analyzed using the Python readability package. Text Cohesion was measured using the online **Coh-metrix**. **Linguistic Inquiry and Word Count (LIWC)** was used to determine the psychological associations. Also, idea uniqueness was determined through network models like **TextRank**. These text analytics showed that winning ideas had more comments on them. Also, the analysis showed that winning ideas have unique topics and are long with larger vocabulary.

E. Nature of Data Collected

Different categories of information can be collected in idea crowdsourcing platforms that can help in the analysis and understanding of these platforms. The categories range from data submitted by participants, to metadata about the submitted ideas, and finally to data about the participants themselves. For instance, in Zhang et al. (2016), different features were collected from reviews of one of the largest electronic marketplaces in China; JD.com. These features can be divided into the following:

a. linguistic features (# of words, # of sentences, average length of sentence, # of adjectives, # of adverbs)
b. Features based on information quality: # of product features, # of subjective sentences, # of objective sentences
c. Reviewer Features: the grade of reviewer, # of reviews
d. Metadata features: # of replies, helpful votes, # of replies, # stars

Similarly, Hossain & Islam (2015) collected information from My Starbucks Idea platform that included details like the number of votes received, points earned by submitter, points earned on idea, number of comments received, category of the idea, and nature of the idea if it is a sole idea or is related with other ideas. Dates of registration of submitter, date of implementation of the idea, and the user time of involvement with the platform itself were also collected. This information was summarized in the Figure 10.



Variables	Descriptions
Dependent variable	
Product	Category of ideas that are product type
Experience	Category of ideas that are experience type
Involvement	Category of ideas that are involvement type
Linked ideas	Group of ideas which are linked with other ideas
Sole ideas	Group of ideas which are not linked with other ideas
Independent variable	
User period of presence	Period from the time of registration of a crowd to a baseline
Ideas submitted	Number of ideas submitted by a particular ideator
Submission to implementation	Period between from the date of idea submission to its date of implementation
Vote submitted by an ideator	Number of votes by an ideator
Vote received on ideas	Number of vote received on ideas
Comments submitted by an ideator	Number of comments submitted by an ideator on the ideas of others
Point earned	Number of points earned by an ideator
Comment received	Number of comments received on ideas
Point on idea	Number of points received on idea

Figure 10: Nature of Collected Data (Hossain & Islam, 2015)

F. Network Analysis

Networks provide a complex representation of interdependence between different points of interaction.

1. Modeling a bipartite network using ERMG

ERGM (Exponential Random graph model) is usually utilized to model social interactions and understand interactions in different types of networks. It is a suitable tool for modeling crowdsourcing platforms and the interactions between the different elements of the platform because it is a method of social network analysis for building complex network structures. Sha et al. (2019) modeled the interactions between participants and that of design contests as a bipartite network using ERMGs to understand their effect on improving participation rates in design contests. The model was applied on a use case which is GrabCAD; a crowdsourcing online platform that connects designers who submit CAD files with sponsors who offer prizes to solving their design problems. The methodology applied showed positive correlation between providing incentives in design contests and participation rates. It also showed how the fraction of total prize allocated to the first prize negatively influences participation, the positive relationship between the contest popularity and participation rates, and how the associations between participants had no effect on improving the participation rate. The goal of the conducted research was to understand the factors that increase participation rates in design contests and more importantly help crowdsourcing implementers design the most effective crowdsourcing platforms that boost productivity and ERMG has helped in this.

2. Coward Networks

As already mentioned in the saracasm detection section of this paper, Taurub et al. (2018) described a model based on heterogeneous **coward network patterns** in order to translate implicit messages (sarcastic) in social media to explicit ones with direct meanings.

G. Machine Learning Algorithms:

There are mainly two types of Machine Learning algorithms (Ayodele, 2010) that help analyze data and draw conclusions:

1. ***Supervised learning***

Supervised learning generates a function that maps inputs into the desired outputs. It includes different classification problems like Regression, Random Forest Classification, and Logistic Regression.

a. <u>*Random Forest Classification*</u>

The basic building block of a Random Forest Classifier is the decision tree. The goal of decision tree learning is to predict target variables based on several input variables using a tree-like model. Tree models that consist of target variables that take discrete values are actually classification trees. However, tree models that consist of target variables that take continuous values are regression trees.

A Random forest builds multiple decision trees, gets prediction from each tree, and averages their results to form a forest. It is a supervised learning method that can be used either for regression or classification (in our case). Since random forest is a highly accurate method that does not suffer from overfitting because it involves taking into consideration several decision trees in the process which cancels any biases, it was adopted for our prediction model. However, it is important to note that this technique is considerably slow in generating predictions because it involves the creation of a number of decision trees to average their values.

b. <u>*Logistic Regression*</u>

Logistic regression is a simple machine learning technique that allows you to classify data into categories. There are different types of logistic regression including binary logistic regression and multinomial logistic regression. Binary logistic regression is when the target variable can only be of two categories (true or false) while multinomial regression is when the target variable takes more than 3 categories.

Figure 11 shows how binary logistic regression takes two discrete values 0 and 1. Unlike linear regression which is estimated using ordinary least squares, logistic regression is estimated using maximum likelihood approach to determine the specific parametric values for a given prediction model.

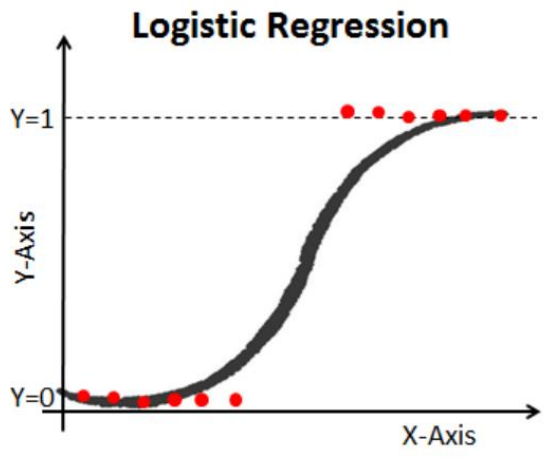


Figure 11: Logistic Regression

Logistic regression is represented by the Sigmoid function "S" shaped curved that can take any real valued number between 0 and 1. If the output is more than 0.5, the outcome is identified as "Yes", and if it is less than 0.5, it is identified as a "No". If the output of an idea in our model is for example 0.8, this means that 80 % that this specific idea will be won. However, the disadvantage of this machine learning technique is that it cannot handle too many features in the model and can suffer from overfitting.

c. *Support Vector Machine algorithm*

Support Vector machine algorithm is a classification model where each data item is plotted in a nth dimensional space. Zhang et al. (2011) constructed a model to rank products (quality) according to customer reviews to help potential customers make more informed purchasing decisions since reading and comprehending all reviews submitted is infeasible by customers. The model described in the paper used Vector Machine algorithm (old machine learning algorithm) to measure the quality of products from reviews by filtering unrelated comments in reviews such as those related customer service (has nothing to do with product quality).

A feature vector X was fed into the following equation: $h(X) = \beta^T X + \beta$ to determine the probability of it being relevant or not. A training set of 1000 sentences collected manually was used to determine the linear regression model described in the equation.

d. *Gradient Boosted Trees*

Gradient Boosted trees is a machine learning technique that is used in classification problems. This technique produces a prediction model in a form of decision trees. The technique was used in (Ahmad & Fuge, 2017) where challenges and ideas were classified using RUSBoost algorithm at first, and then, the results of this set were used to predict ideas on a test set using Gradient Boosted Trees.

2. UnSupervised learning

Unsupervised learning does not have any target or desired outcome to predict. It is used for clustering population in different groups and includes algorithms like Apriori algorithm and K-means.

H. Results Validation

Validating results and ensuring that the target is achieved can be done using different tools including using Mean Average Precision, Discounted Cumulative gain, and ROC Curve.

1. Discounted Cumulative Gain

Discounted Cumulative gain is a measure of ranking quality. It measures the usefulness of a document based on its position in the result haystack. This is mostly used in information retrieval to measure effectiveness of search algorithms. This method was used to measure the effectiveness of the ranking quality algorithm that was proposed in Ahmed & Fuge (2017).

2. Mean Average Precision

AP is used to score document retrieval and it reflects how relevant results are upon searching. If we type something in Google for example and it shows us 10 results, it is best if all these results were relevant and if some of them are relevant, it is better if the relevant ones are displayed first.

To check if the model adopted in (Zhang et. al, 2011) was ranking the products correctly, the results were compared to the sales rank adopted by amazon (If the product is sold more, it means it is better – of higher quality). Also, a method which is called Mean Average Precision (MAP); a popular measure used in information retrieval for evaluating ranking accuracy (Turpin & Schole, 2006) was used in the same paper for results validation.

3. ROC Curve

An ROC Curve measures the accuracy of a classification model. Accuracy is determined by the proportion of observations that are correctly predicted to be positive and the proportion of observations that are incorrectly predicted to be positive.

An important factor in an ROC Curve is measuring the AUC ROC score which is the area under the curve and is usually between 0 and 1 where a meaningful classifier usually has an AUC greater than 0.5.

I. Release Planning

Release planning is the ability to select and assign features to consecutive releases taking into consideration technical, resource, budget, and risk constraints (Ruhe & Saliu, 2005). It is a crucial step in improving any product because it makes sure customers' expectations are met and quality, cost, and effort constraints are not violated. According to Pfleeger (2002), the best release plan is the plan that involves feasible features that bring the greatest business value in the right sequence of releases satisfying all the stakeholders involved and taking into consideration resource capacities and feature dependencies.

1. The Art and Science of Software Release Planning

An inclusive release planning process is hybrid and is divided into two important parts: the human intuition (expert knowledge) and the science of problem formulation (computational intelligence) (Ruhe & Saliu, 2005). All stakeholders like project managers, senior developers, and project sponsors meet and decide which features should be developed informally or through spreadsheets by balancing manually resources with their interests. This is complex to be applied alone when multiple stakeholders are involved, and there are conflicting demands between resources, interests, and constraints (Ruhe & Saliu, 2005) This is mainly due to the big number of features involved and different conflicting objectives between stakeholders. For example, out of a set of 100 features, a project manager could prefer to launch a feature in the first release to enhance user experience where as a senior developer might believe it is not as important as another feature that optimizes a code script. This leads to a complex optimization problem that makes sure all stake holders are satisfied with the release plan produced.

Ruhe & Saliu (2005) formulated an optimization problem and decomposed it into the following:

a. *<u>Decision Variables</u>*

Suppose there are n features, these features can be described as decision variables $x_{(1)}$, $x_{(2)}$….$x_{(n)}$ with $x_{(i)} = k$ if feature "i" ($1 \leq i \leq n$) is assigned to release option k where $1 \leq k \leq K$, and $x_{(i)} = 0$ if a feature is not implemented in any release.

b. *<u>Dependencies:</u>*

Features can be related to each other. They can either have a coupling relation C (they should be implemented in the same release because they depend on each other), a precedence relation P (they should be implemented consecutively), or no relation at all.

c. *<u>Resource constraints:</u>*

Resource constraints are related to budget and effort consumption. The sum of resources of type t needed by features in a certain release should be less than the available capacity of resources of type t of this release.

$$\sum\nolimits_{\mathbf{x(i)=k}} \mathbf{r(i,t)} \leq \mathbf{Cap(k,t)}$$

where t represents type of resource, k represents the release number, i represents the feature number, r represents the amount of resources of type t, and Cap(k,t) represents the capacity of a resource of a certain release.

These are estimates determined by experts and prone to uncertainties.

d. *<u>Stakeholders:</u>*

In choosing what idea to implement, stakeholders are usually involved. These stakeholders have different relative importance assigned to them depending on their position in the organization (Ruhe & Saliu, 2005). For a set of stakeholders, each stakeholder can have a different importance level based on a 9-point scale ordinal [1,3,5,7,9] where 1 represents very low importance and 9 represents extremely high importance. Stakeholders can be product managers or developers.

$$\boldsymbol{P(s) = p}$$

Where $1 \leq s \leq S$ *stakeholders and* $p \in [1,3,5,7,9]$.

e. *Prioritization of ideas:*

Prioritization of ideas is determined based on 2 factors: value (how valuable is this feature) and urgency (how urgent is this feature). These values are set by the stakeholders. They are also determined based on a 9-scale ordinal. A feature with Value= 1 has a lower priority than a feature with value= 9.

$$\boldsymbol{V(s, i) = v}$$

Where $v \in [1,3,5,7,9]$ *,* $1 \leq s \leq S$ *stakeholders,* $1 \leq i \leq n$.

Urgency is determined by giving a set of votes to a stakeholder. For example, suppose there are 2 releases. Each stakeholder is given 9 votes on a particular feature to distribute among the releases to allow sufficient differentiation in the degree of importance (Number of votes = 9). If a stakeholder voted as (9,0,0), then this feature is very urgent to be produced in the first release as per the stakeholder. If a stakeholder voted (0,0,9), then the stakeholder thinks this feature is not urgent and should be postponed after the 2 releases. If a stakeholder voted (3,3,3), then this stakeholder thinks that this feature can be implemented in any release or even postponed.

$$\boldsymbol{U(s,i) = [u_1, u_2 \ldots u_k]}$$

$$\boldsymbol{Where\ \sum_{y=1\to k} u_y \le Number\ of\ Votes_s}$$

$$\boldsymbol{1 \le s \le S\ stakeholders}$$

$$\mathbf{1 \le \boldsymbol{i} \le n}$$

f. *Objective Function:*

The objective is to increase the weighted average satisfaction of all stake holders with the release plan of features:

$$\mathrm{F(x)} = \sum_{\mathrm{k=1\ldots K}} \sum_{\mathrm{x(i)=k}} \mathrm{WAS_{(i,k)}}$$

$$\text{where } \mathrm{WAS_{(i,k)}} = \xi(\mathrm{k})[\textstyle\sum_{\mathrm{s=1..k}} \mathrm{P(s).V(s,i).U(s,i,k)}]$$

$$\mathrm{P(s)} = \text{importance of stakeholder}$$

$$\xi(\mathrm{k}) = \text{importance of the release}$$

$$\mathrm{V(s,i)} = \text{Value of feature i from stake holder s}$$

$$\mathrm{U(s,i,k)} = Urgency\ of\ feature\ i\ from\ stake\ holder\ s\ for\ release\ k$$

$$\textit{Subject to Constraints: } \textstyle\sum_{x(i)=k} r(i,t) \le Cap(k,t)$$

2. *Determination of the Next Release of a Software Product: An Approach using Integer Linear Programming*

Selecting the needed requirements for the next release is a complex task due to many reasons including the different interests of stakeholders involved and the high number of requirements at hand (Akker et al., 2005). To aid managers in requirements management, an Integer Linear Programming technique was proposed that takes as input the different number of requirements, estimated revenue per requirement, and availability of resources and produces a release's best set of requirements that leads to maximum projected revenue against available resources in a given period of time. In addition, managerial steering mechanisms were introduced like transferring developers across teams.

To model the Integer Linear Programming model, they let {R1, R2…Rn} be the set of requirements, and they assumed it was possible to estimate the revenue for each requirement as v_j. They also assumed that there was a fixed development period denoted by T and denoted d(T) as the number of working days in a development period, and Q as the number of persons working in the development teams of the company.

The problem was thus modeled at first as the following Linear Integer programming problem:

$$\text{Maximize } \sum_{j=1}^{n} v_i x_j$$

$$\text{Subject to } \sum_{j=1}^{n} a_j x_j \leq d(T)Q$$

$$x_j \in \{0,1\} \text{ for } j=1,\cdots,n$$

However, the problem was also extended to include other important scenarios. If a certain company decides for sure to have a requirement i in the release for example, then x_j will be equal to 1 for this requirement "i" and the equation would be added as one of the constraints above. In addition, if the developers working have different number of working days, this will be denoted by $\sum_{p=1}^{P} d_p(T)$ where $d_p(T)$ is the number of working days per person.

If there is more than one development team each with their own specialization in the organization, and m is the total number of teams available, then G_i (i =1,...,m) consists of Q_i persons, so the implementation of requirement R_j would need a_{ij} of man days from team $G_{i.}$

Thus, the constraint can be modified into the following:

$$\sum_{j=1}^{n} a_{ij}x_j \leq d(T)\, Q_i \text{ for I =1,…,m}$$

To increase revenue, one team member from team G_i can work in team G_k to provide more flexibility so the constraint can finally be modified into:

$$\sum_{j=1}^{n} a_{ij}x_j \leq y_{ii} + \sum_{k:k\neq i} b_{ki}x_{ki}$$

$$\sum_{k=1}^{m} y_{ik} = m_i \; x_j \in \{0,1\} \text{ for j=1,}\cdots\text{,n , i=1,}\cdots\text{,m}$$

It was shown that the assumption that considered no different teams of developers involved produced the highest revenue, followed by the scenario which included transfers between teams, and followed lastly by the scenario that treated teams as independent.

3. Optimized Resource Allocation for Software Release Planning

Ruhe (2009) considers the planning of software releases and allocation of resources as one entity: defining releases without digging into the resources leads to unfeasible plans. The paper uses Integer Linear programming and genetic programming to generate operation resource allocation plans. According to Ruhe (2009), without optimal release planning, companies will miss implementing the right features at the right time causing dissatisfied customers and exceeding budget.

The paper defines a release plan as the assignment of features to releases where x(n,k) = 1 if feature f(n) is offered at release k (k =1…K) and x(n,k)= o otherwise. The paper also considers that features can have a coupling or precedence relationship. For example, if feature f(i) precedes feature f(j), and f(i) is to be implemented in release k, then feature f(j) cannot be implemented at any release 1…k-1 earlier than k.

F(x) represents the total value of individual values v(n,k) of each feature f(n) assigned to release k. The model also considers that each feature may require different tasks that include technical, quality assurance and managerial tasks. It also considers that different human resources may be needed to complete the different tasks of features (analysts, developers for example). Also, features might require the consumption of non-human resources like capital. The model considers as well that different developers might have different productivity or expertise levels and thus this affects their assignment to features and release planning. Finally, the objective function is defined to increase the value of different features subject to the resource allocation u.

Maximize $\sum_{n=1}^{N} \sum_{k=1}^{K} v(n,k).\, x(n,k)$

Subject to $(x,u) \in \{X,U\}$

CHAPTER III

METHODOLOGY

The objective of this thesis is to determine the winning ideas from an idea crowdsourcing platform to implement in the next release for the product in an innovative 2-stage amalgamated process. To achieve this objective, the methodology is divided into two sections:

1. Predicting the winning ideas needed to be implemented based on the crowd profiles and feedback.
2. Putting these winning ideas optimally on the product roadmap by the management team taking into consideration the resource, budget, technical, and risk constraints.

Thus, as shown in Figure 12, our model combines the ability to determine winning ideas in crowdsourcing platforms through a classifier which is based on the crowd’s feedback with the capability to determine which of these wining ideas should be put in the current release product roadmap through resource allocation.

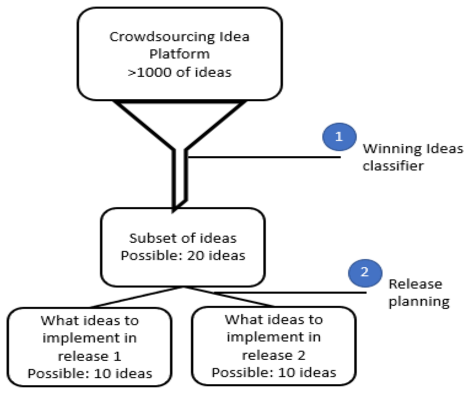
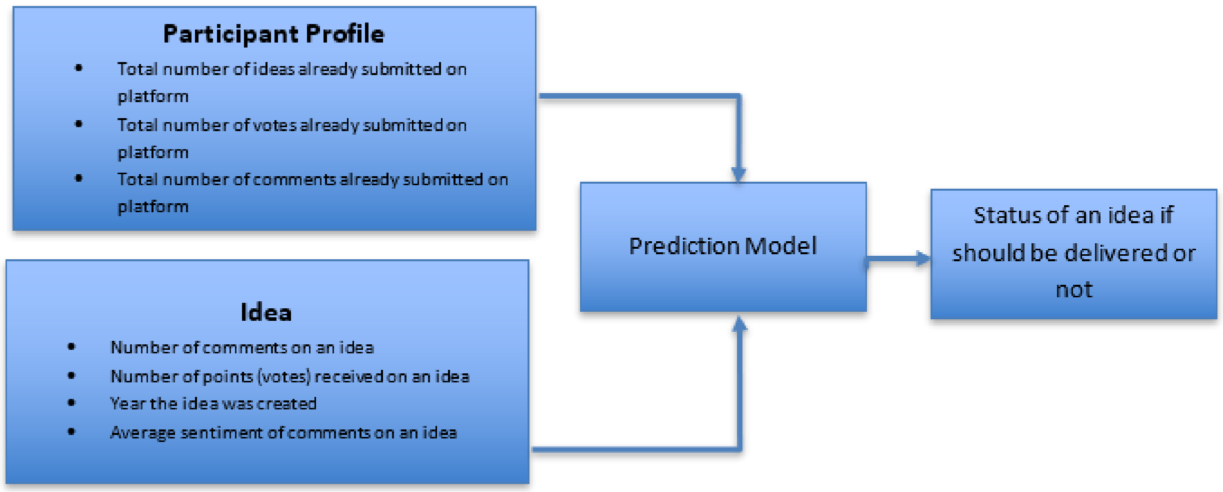

Crowdsourcing Idea Platform >1000 of ideas
1
Winning Ideas classifier
Subset of ideas Possible: 20 ideas
2
Release planning
What ideas to implement in release 1 Possible: 10 ideas
What ideas to implement in release 2 Possible: 10 ideas

Figure 12: Methodology General Overview

A. Prediction Model based on Crowd's feedback and Profiles

Idea Crowdsourcing platforms are rich in data. They do not only include information about the ideas submitted, but also details about the profiles of the participants who submitted them. To construct our prediction model that predicts winning ideas in crowdsourcing platforms, it is very important that we extract this data, analyze it, and then produce our predictions. As shown in Figure 13, the information we collect about an idea includes not only the text of the idea but also the number of comments it received, votes it acquired, year it was created in, and the average sentiment it portrayed. Similarly, the information collected about a participant's profile is not only directly related to the idea itself but also connected to his previous participation through votes, comments, and ideas. After extracting this valuable information from the crowdsourcing platform, we can use this collected information to discover a trend and create a prediction that lets us know if this idea has the potential of being executed by the implementation team or not.



Participant Profile
Total number of ideas already submitted on platform
Total number of votes already submitted on platform
Total number of comments already submitted on platform
Idea
Number of comments on an idea
Number of points (votes) received on an idea
Year the idea was created
Average sentiment of comments on an idea
Prediction Model
Status of an idea if should be delivered or not

Figure 13: Prediction Model

To build and test the prediction model using different tools and mechanisms, the following methodology as described in Figure 14 was followed. Data is first extracted from the crowdsourcing sourcing platform using the Python software and collected into excel sheets. These excel sheets are then processed to remove any noise that could affect our results. To understand our data better, we used Tableau to draw conclusions about the data we collected and form preliminary hypotheses. Finally, a random Forest Classifier was created and fed with the data to produce the necessary predictions.

Each part of the methodology that we followed is further illustrated in the sections that follow as summarized in Figure 14.

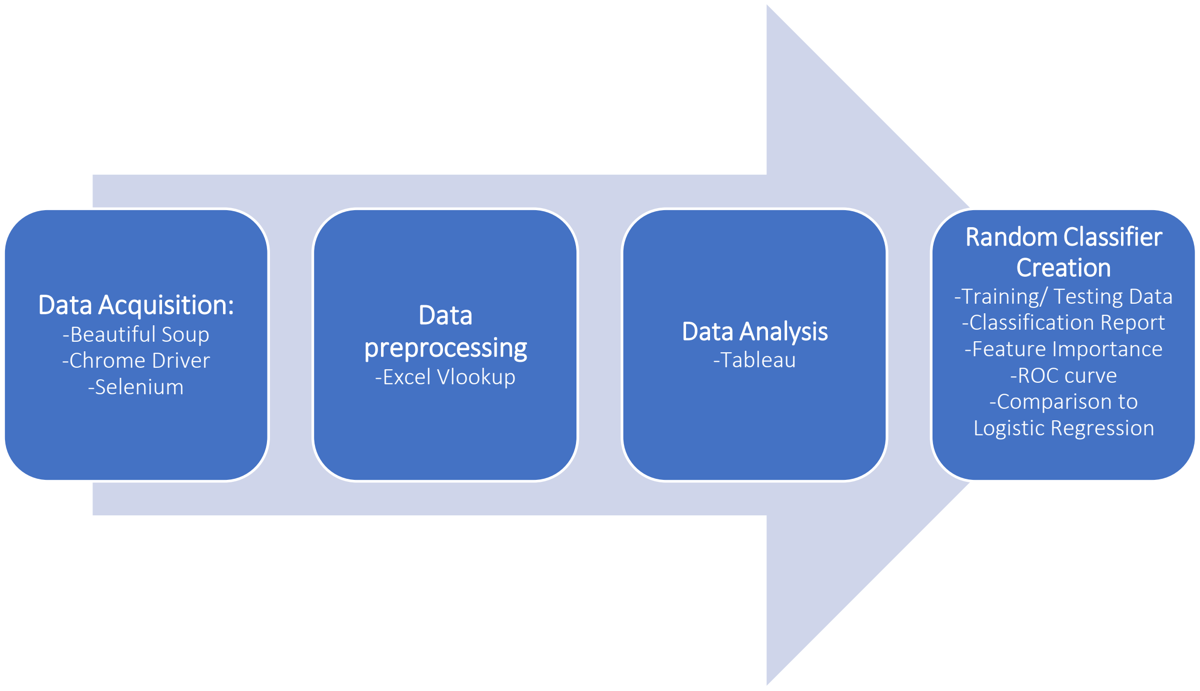


Figure 14: Prediction Classifier Creation Process

1. *Data Acquisition*

Data was collected from an online idea crowdsourcing platform using "Beautiful soup" library in Python. The Beautiful Soup Library allows you to collect data from a web browser by parsing the HTML code to fill data in excel sheets. A spider code was created to scrape the ideas with their statuses, number of votes earned, number of comments received, and year created. Also, profiles of the idea submitters were scraped using Selenium and chrome driver in Python. The data collected about the participants included their total number of ideas, comments, and votes resulting from all their participation previously with the platform.

The aim of gathering data about the participants was to understand the influence of the submitters and how their profiles can have a strong effect on getting their ideas to be put on the product roadmap of the company.

2. Data processing

As was just discussed, two different codes were used in Python to scrape the data from the online idea crowdsourcing platform: one related to the ideas, and the other related to the profiles of the submitters. Thus, the data was scraped and collected into two different sheets: (1) ideas data and (2) idea submitter data corresponding to the different ideas each. To be able to analyze this data and input it into the prediction model, it was necessary to filter any null values first and then combine the two different excel sheets into one table sheet. This is because null values have devastating effects on prediction models. Thus, the two sheets were then combined into one sheet using VLOOKUP depending on the idea ID as a key. The final ideas sheet consisted of non-null values of ideas, number of votes received per idea, number of comments received per idea, number of total votes submitted by a participant, and total number of comments submitted by a participant. The average sentiment of comments of an idea was also calculated as the average of total comments for an idea by using the Sentiment package in Python. Ideas that were submitted months, days, or even hours ago were assumed to be created a year ago. For ideas in which we were not able to access the profile participant data due to privacy or expiry, we decided to remove them.

3. Data Analysis

To analyze and investigate the data collected before building the prediction model, Tableau was used to construct visualizations of the data. By building these visualizations, we were able to understand our data better before creating our prediction model. We were able to understand the percentage of the ideas collected that were delivered relative to the total number, the relationship between the average number of votes received on the idea and the year it was created, and the effect of the participants profile on the status of the idea.

4. Random Forest Classifier Creation

To predict and thus filter ideas that are to be delivered or not, a Random Forest Classifier was adopted and fed with the collected processed data.

Our random Forest Classifier was created using Python. It involved reading the data and then splitting it into training and testing data sets. In machine learning, training data sets are used to train a prediction model on making accurate predictions whereas a testing data set is used to validate the accuracy of the predictions.

After creating our Random Forest Classifier, it was very important to determine the accuracy of it because accuracy plays an important role in understanding how effective a prediction model is. To determine the accuracy of the prediction model, a classification report and an ROC Curve can be used in Python. Accuracy is the fraction of predictions that the model got correct and is defined by: $\frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ predictions}$.

Therefore, both a classification report and an ROC Curve were constructed to show the accuracy of determining winning ideas. Finally, the importance of every feature was determined to understand what predictors have the greatest effect on the prediction model, and what features were better to be removed to improve our model.

5. Logistic Regression Creation

To validate that Random Forest Classifier is the most accurate to use in predicting ideas to be delivered, Logistic Regression was created to compare it with Random Forest Classifier's results.

Logistic Regression was implemented using Python as well and compared to Random Forest Classifier in terms of Accuracy using an ROC Curve to illustrate how Random Forest Classifier is the most accurate.

B. Release Planning

After determining the most promising ideas through our prediction model based on the Random Forest Classifier, we will be presented with a subset of ideas based on the crowd's input. This subset can then be filtered as a second stage based on different stakeholders' decisions. To filter these ideas, we have formalized an optimization problem that maximizes the value of features given by different stakeholders. This optimization problem is built based on the Knapsack formulation problem. The objective of a Knapsack is to maximize the value of items that can fit into a knapsack without exceeding a maximum weight constraint. In our case, it is to maximize the value of implementing ideas without exceeding the budget allocated for every release.

The general formulation of a multiple knapsack problem will be our starting point for our optimization model:

$$\text{Maximize } \sum_{k=1}^{K} \sum_{i=1}^{F} v_i x_{ik}$$

$$\text{subject to } \sum_{i=1}^{F} w_i x_{ik} \leq c_k \text{ where k} \in \text{K} = \{1,\ldots,\text{K}\}$$

$$\sum_{k=1}^{K} x_{ik} \leq 1 \text{ where i} \in \text{F} = \{1,\ldots,\text{F}\}.$$

$$x_{ik} = 0 \; or \; 1 \; depending \; on \; if \; item \; i \; is \; assigned \; to \; knapsack \; k \text{ where i} \in \text{F , k} \in \text{K.}$$

1. Decision variables

Starting from the Knapsack optimization problem illustrated before, suppose there are F predicted winning ideas produced (1≤ *i* ≤ F) from the first filtering stage through the Random Forest Classifier, these promising ideas can be described as decision variables $\mathbf{x_{ik}}$ **= 1** if idea **"i" (1 ≤ *i* ≤ F)** is assigned to release option **k** where **1≤ *k* ≤ K**, and $\mathbf{x_{ik}}$ **= 0** if an idea "i" is not implemented in release k.

2. Constraints

An idea is only implemented in one release, and thus cannot be assigned to more than one release which can be denoted by Equation (1):

$$\sum_{k=1}^{K} x_{ik} \leq 1 \text{ where } i \in F = \{1,\ldots,F\} \qquad (1)$$

In addition, resource constraints are related to budget and effort consumption. The total number of resources consumed for all ideas in a release should be less than the budget allocated for a certain release. To implement an idea, it is important to ensure not only to cover development cost, but also quality assurance and project management cost. Thus, the Cost of implementing one idea is usually composed of the sum of management cost, development cost, and quality cost needed to implement an idea.

Management cost is usually equal to 20% of (development and quality assurance cost) and quality assurance cost is usually 25% of development cost.

Let B_k be the available development budget of release k.

Define c_i as the development cost for feature i, where:

$$c_i = \text{Labor (Man-days)} * \text{Labor rate (\$/man day)} \ (2)$$

We assume that the management and quality assurance cost combined is 50% of development cost. This leads to our new constraint as depicted in equation (3).

Constraints:

$$\sum_{i=1}^{F} 1.5 c_i x_{ik} \leq b_k \qquad (3)$$

Where k ∈ K = {1,…,K}

3. Stakeholders

In choosing what idea to implement, stakeholders are usually involved. These stakeholders can have different relative importance assigned to them depending on their positions and power in an organization. The higher the importance, the more control they have on implementing the idea in a certain release. Importance priority of a stakeholder can be defined by Equation (4):

$$\boldsymbol{P(j) = p}$$

Where

$$\boldsymbol{1 \leq j \leq S \text{ stakeholders}} \qquad \textbf{(4)}$$

$$\boldsymbol{p \in [1, 2, 3, 4, 5, 6, 7, 8, 9]}$$

4. Release Importance:

Driven by market demands, each release can have an importance factor denoted by ξ_k which represents the weighted average of importance factors given by the different stakeholders for a given release. Each stakeholder can distribute a sum of 100 votes on the different releases to indicate the importance of each.

$$\boldsymbol{\xi_k = \frac{\sum_{j=1}^{S} \xi(k,j)}{100S}} \qquad \textbf{(5)}$$

$$\textbf{Where } \boldsymbol{\sum_{k=1}^{K} \xi_k = 1}$$

S = Number of Stakeholders

5. Prioritization of ideas:

Each idea can be given a value by a stakeholder for a given release to indicate how valuable it is and how urgent. An idea “1” with Value V(1,2,3)= 1 assigned by stakeholder

2 for release 3 has a lower value than the same idea “1” with V(1,1,3) = 4 assigned by stakeholder 1 for the same release 3.

$$\mathbf{V}\ (\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}) = \boldsymbol{v} \qquad \textbf{(6)}$$

Where

$$\boldsymbol{v} \in [\mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{4}, \mathbf{5}, \mathbf{6}, \mathbf{7}, \mathbf{8}, \mathbf{9}]$$

$$\mathit{1} \leq \boldsymbol{i} \leq \boldsymbol{F}$$

$$\mathit{1} \leq \boldsymbol{j} \leq \boldsymbol{S}$$

$$\mathit{1} \leq \boldsymbol{k} \leq \boldsymbol{K}$$

6. Objective function:

It is important to maximize the weighted average value (WAV) of all features given by different stakeholders for different releases to ensure the maximum satisfaction of stakeholders. Thus, we denote our objective function by:

$$\boldsymbol{Max\ F}(\boldsymbol{x}) = \sum_{\boldsymbol{k}=\mathbf{1}}^{\boldsymbol{k}} \sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{F}} \boldsymbol{WAV}(\boldsymbol{i}, \boldsymbol{k}) \boldsymbol{x}_{\boldsymbol{ik}}$$

$$\boldsymbol{WAV}(\boldsymbol{i}, \boldsymbol{k}) = \boldsymbol{\xi}_{\boldsymbol{k}} \sum_{\boldsymbol{j}=\mathbf{1}}^{\boldsymbol{S}} \boldsymbol{P}(\boldsymbol{j}).\boldsymbol{V}(\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}).$$

P(j) =importance priority of stakeholder

$$\boldsymbol{V}(\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k})$$
$$= \boldsymbol{Value\ of\ feature\ i\ given\ by\ stake\ holder\ j\ for\ release\ k}$$

$$\boldsymbol{\xi}_{\boldsymbol{k}} = \boldsymbol{importance\ of\ release\ k}$$

$$\boldsymbol{\xi}_{\boldsymbol{k}} = \frac{\sum_{\boldsymbol{j}=\mathbf{1}}^{\boldsymbol{S}} \boldsymbol{\xi}(\boldsymbol{k}, \boldsymbol{j})}{\mathbf{100}\boldsymbol{S}}\ \boldsymbol{where}\ \sum_{\boldsymbol{k}=\mathbf{1}}^{\boldsymbol{K}} \boldsymbol{\xi}_{\boldsymbol{k}} = \mathbf{1}\ \textbf{and}\ \mathbf{j} \in\ \mathbf{S} = \{\mathbf{1},\ldots,\mathbf{S}\}.$$

Subject to Constraints:

$\sum_{k=1}^{K} x_{ik} \leq 1$ **where i ∈ F = {1,…,F}.**

$\sum_{i=1}^{F} 1.5 c_i x_{ik} \leq b_k$ **where k ∈ K = {1,…,K}**

Where c_i***= Number of Man days * Price/man day.***

CHAPTER IV

CASE STUDY

Our model was constructed and applied to an idea crowdsourcing platform of a Multinational Technology Provider Company that remains confidential. The crowdsourcing platform of our company collects ideas from the public about desirable features to be included in the next scheduled Technology releases of the company. The platform is suffering today from the voluminous amount of data that is rendering it ineffective in responding to the public's needs and requirements. The company is not being able to effectively prioritize the needed ideas that are necessary to implement in the next releases. The data was extracted from the platform and analyzed to predict winning ideas using Random Forest Classifier. They were then filtered by a second stage filtering optimization solution based on Knapsack algorithm with multiple knapsacks.

A. Prediction Model based on Crowd's feedback and Profile

1. *Data Acquisition, Processing, and Analysis:*

Data was crawled using Selenium, ChromeDriver, and BeautifulSoup in Python from the crowdsourcing platform that remains anonymous in the thesis due to confidentiality. To analyze and investigate the data collected, Tableau was used to construct visualizations of the data. However, it is important to note that any other data analysis tool like Excel can also be used. The result of our collected data consisted of around 1152 records of delivered and non-delivered ideas. Figure 15 shows the number of records per status. More than 85% of the collected data were ideas that were not yet delivered.

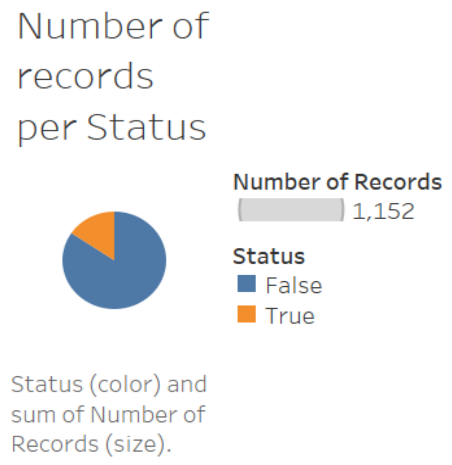


Figure 15: Number of records per status

To understand the effect of points (votes) received on an idea and their relationship to the year the idea was created in, Figure 16 was constructed to display the average points per created year that delivered and non-delivered ideas have received. The results show that the average points (votes) are higher for ideas that were created long time ago (13 years ago) than ideas that were recently created. This coincides with the notion that older ideas should have a greater number of votes than newer ideas because they were exposed to the public for longer periods. What is significant and bizarre however, is that delivered ideas received less average number of points than non-delivered ideas.

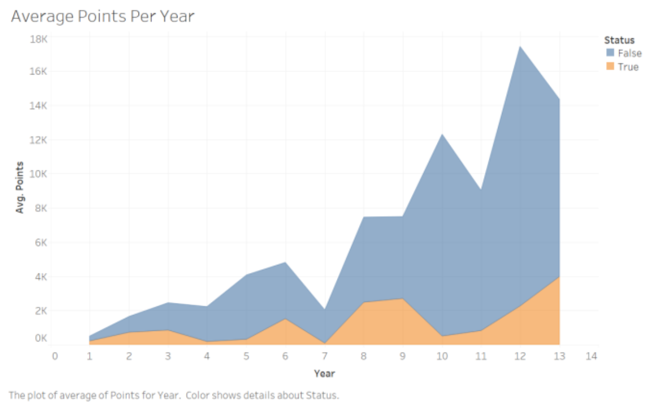


Figure 16: Average Points / Year

To illustrate the relationships between ideas submitters and the deliverability of ideas, Figure 17 compared the profiles of submitters of winning and non-delivered ideas. It shows that the average number of total votes that participants of winning ideas have given is much higher than the average number of total votes that participants of non-delivered ideas have given. This might give a notion that ideas of participants with more active profiles are preferred over those of less active participants.

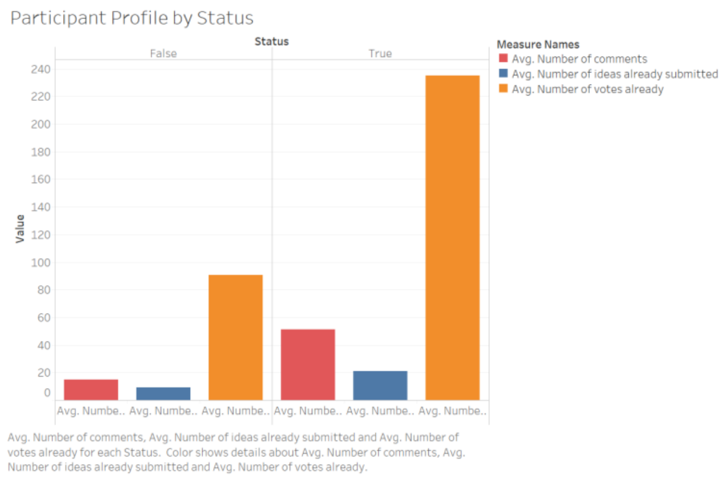


Figure 17: Participant Profile by Status

After investigating the data collected and understanding which attributes or factors that might have the greatest effect on determining if an idea will be a winning idea or not, a random Forest Classifier was created and fed with 1152 entries of data with the following columns:

- **Points** which are the earned votes an idea received from the different participants and public
- **Comments** which is the number of comments an idea received from the public
- **Status** of the idea which determines if an idea was delivered and implemented or not
- **Year** which represents the year the idea was posted on the platform
- **Sentiment** of average of comments which determines the average sentiment of the comments posted on an idea

- **Number of ideas** already submitted by the idea submitter which signifies how advanced his or her profile is
- **Number of votes** already submitted by the idea submitter which signifies how interactive the submitter is with other existing posts
- **Number of comments** already submitted by the idea submitter which shows how much experience the submitter has

Data was then split into training and testing data sets. 20% of the data was allocated for testing which are 231 records out of 1152 records and 80% was used in training.

Before taking a single prediction run and investigating it further, we ran our model 10 times with randomness for our training and testing data sets which means on each run, our training and testing data sets were chosen randomly. It is important to note that unless we pick different datasets for our training and testing data sets or change the percentage of data allocated to testing and training, we will always get the same results when we run our prediction model multiple times. Our prediction model was run 10 times to understand what ideas were always predicted to be delivered between the different runs, and if there were any pattern between them. Our Python code produced predicted ideas on each run, and each set of predictions was outputted to an excel sheet. We used COUNTIF between the excel sheets to consolidate which ideas were predicted in common between all the sheets and investigate the pattern between them.

Some ideas were common in 2, 3, 5, or 5 runs. Ideas that were commonly predicted to be delivered across 5 out of the different 10 runs are shown in Table 1. We can see from Table 1 that these ideas were created many years ago, and they had received a high number of votes which means that years created and number of votes received plays an important role in our prediction model.

Idea	Number of Votes	Year
1139	424	12
1040th	104	11

Table 1: Ideas Produced from 10 Runs

A confusion matrix was built to understand how close our predictions and our actual values are. The number of ideas that were correctly predicted to be delivered and they were indeed delivered were 9. The ideas that were correctly predicted to be non-delivered and were in fact non-delivered are 201 ideas. There were 5 ideas that were falsely predicted to be delivered when they are not in fact delivered, and 16 ideas that were falsely predicted to be non-delivered when they are in fact delivered.

201 (TN)	5 (FP)
16 (FN)	9 (TP)

Table 2:Confusion Matrix

To understand how effective our Random Forest Classifier is, we created our classification report. Our classification report shows the Precision, Recall, F1-Score, and values for our different ideas.

- Accuracy represents the proportion of the predictions that the model classified correctly: Accuracy =$\frac{TP+TN}{TP+TN+FP+FN}$
- Precision shows the proportion of predictions that were correctly predicted to be true: Precision = $\frac{TP}{TP+FP}$
- Recall (True Positive Rate) shows the proportion of actual positives that was identified correctly without missing any hits: Recall = $\frac{TP}{TP+FN}$
- F1-Score represents a harmonic mean between precision and recall and it measures the preciseness and robustness of a model: F1-Score = $\frac{2TP}{2TP+FP+FN}$

The accuracy of our prediction model was found out to be 91 % which means 91% of our all predictions were correctly predicted. However, our model is 64% precise in predicting correct ideas which means that 64% of the ideas were predicted to be delivered and they were in fact delivered. In addition, our model has a recall of 36% which means that 36% of our ideas were predicted to be delivered and 64% of ideas that were supposed to be predicted to be delivered were missed. The closer our percentages to 100, the better.

	Precision	Recall	F1-Score	Support
Non-Delivered	0.93	0.98	0.95	206
Delivered	0.64	0.36	0.46	25
Accuracy			0.91	231
Macro Avg	0.78	067	0.71	231

Table 3: Classification Report

We constructed an ROC to find AUC in our prediction model and to assess our classifier further. The ROC Curve shows the TPR or Sensitivity OR recall ($\frac{TP}{TP+FN}$) vs the FPR ($\frac{FP}{FP+TN}$) which is also 1-specificity (1- $\frac{TN}{TN+FP}$).As shown in figure 18, AUC in our prediction model is 0.88 which shows that our classifier is 88 % accurate in predicting the next winning idea. This means that our classifier is relatively a good classifier since AUC > 50%. Also, since our TPR is 36% (0.36), we can conclude from our graph that our FPR is around 2.5% (0.025) which is close to zero. This is a good indication since this means that our false positives are low which means the number of ideas that are predicted to be winning and are not in fact not winning is low.

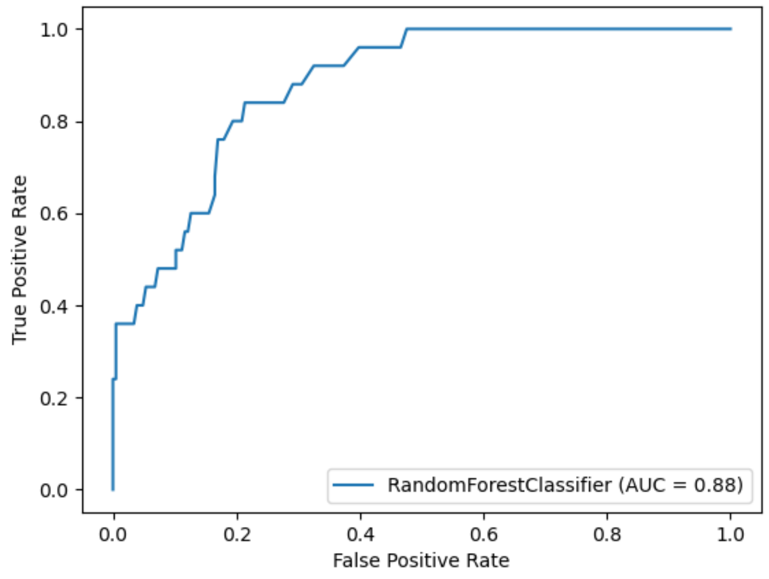


Figure 18: Prediction Model ROC Curve

To interpret and explain machine learning models that are not intuitive and too complicated for a human to understand, we can use specific techniques to understand explainable models such as feature importance and LIME (Local Interpretable Model-Agonistic Explanation). The importance of every feature was determined to understand what predictors had the greatest effect on the prediction model, and what features were better to be removed.

As shown in figure 19, year created was found to have the biggest effect (30%) on predicting ideas to be delivered followed by points received (20%) whereas the number of total comments and total ideas that the submitter has submitted previously on the platform had the lowest effect on the prediction model (less than 10%).

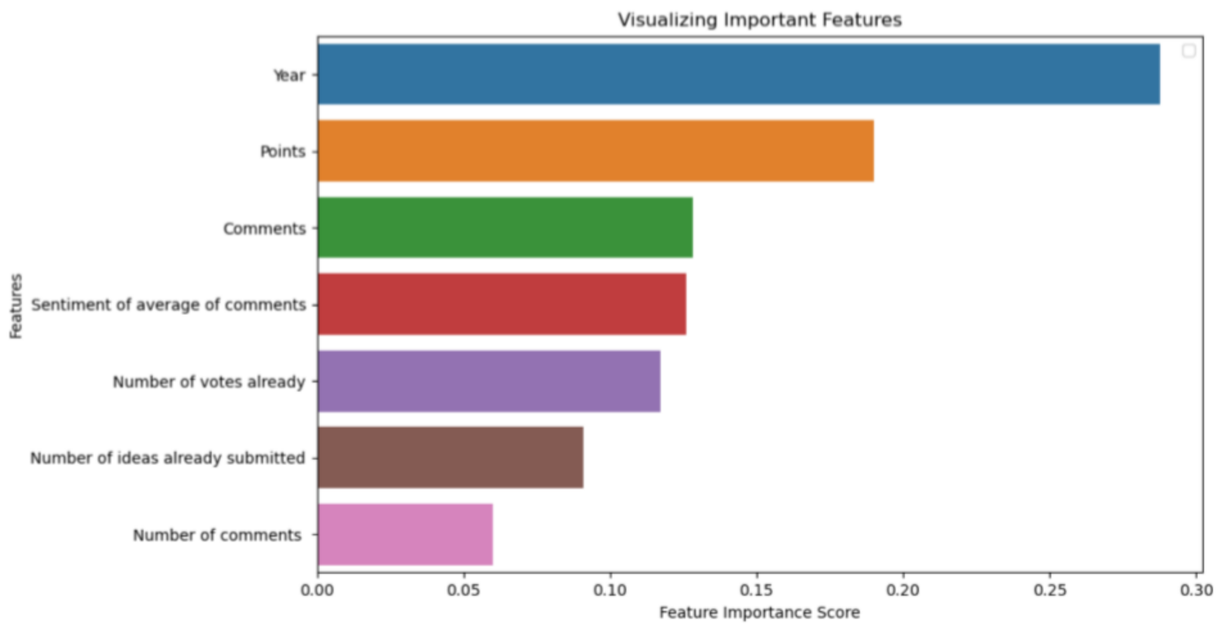


Figure 19: Visualizing Important Features

To increase accuracy in our prediction model, the total number of ideas already submitted by the idea submitter on the platform and total number of comments already submitted by the idea submitter on the platform were dropped. The model was run again to check for improvements. As shown in Table 4, Precision increased from 64% to 79-80% for delivered ideas, and recall improved from 36% to 44%. In addition, accuracy increased by 2% to reach 93%

	Precision	Recall	F1-Score	Support
False	0.94	0.99	0.96	206
True	0.79	0.44	0.56	25
Accuracy			0.93	231
Macro Avg	0.86	0.71	0.76	231

Table 4: Classification Report

2. *Logistic Regression*

To compare the results of Random Forest Classifier with another machine learning algorithm, Logistic regression model was adopted. Accuracy and precision of the model were determined to be 89% and 50% respectively which were lower than the Random Forest classifier.

Similar to Random Forest Classifier, we constructed an ROC to find AUC in our prediction model. As shown in figure 10, AUC in our prediction model is 0.83 which shows that Logistic Regression Classifier is 83 % accurate in predicting the next winning idea. This means that our Random Forest classifier is relatively better than the Logistic Regression Classifier.

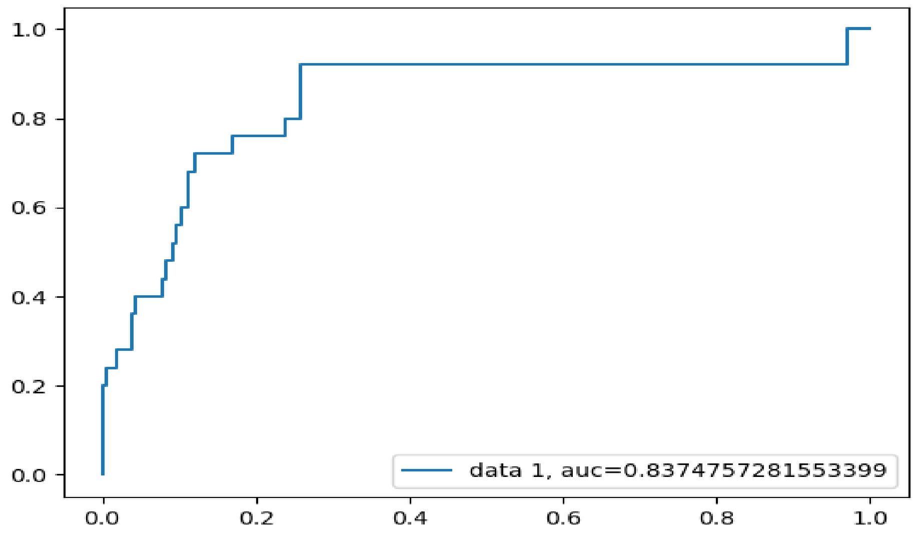


Figure 20: Logistic Regression ROC

B. Release Planning

Our prediction model through the Random Forest Classifier has given us 14 ideas that were predicted to be winning ideas. We compared predicted values with actual values and found out that only 4 ideas out of 14 were wrongly predicted to be delivered when they were not actually winning ideas. However, we kept the 14 ideas to use in our second stage filtering process to check if they will be filtered out by our second stage filtering process.

After predicting our winning ideas through a Random Forest Classifier, we have used our formulized optimization problem to narrow down what ideas out of these 14 should be implemented and in which release.

1. ***Number of releases:***

After interviewing 3 stakeholders from our company, we have discovered that they were considering which features to include in their next release, which features to postpone in the release after, and which features not to implement at all. According to the stakeholders, it is only important to plan for the year ahead which consists of 2 releases since planning for more than 2 releases is prone to market and demand changes as new ideas are produced and collected. Therefore, we took into consideration that there only 2 releases to include in the release plan for the confidential platform $1 \leq k \leq 2$.

2. ***Decision Variables:***

After running our Random Forest Classification model, we have concluded a subset of 14 predicted ideas to be winning with $x_{ik} = 1$ if idea i ($1 \leq i \leq 10$) is assigned to release option k where $1 \leq k \leq 2$, and $x_{ik} = 0$ if an idea is not implemented in release k.

3. ***Stakeholders:***

We met with three stakeholders who are the Project Sponsor, Project Manager, and Developer denoted by j = 1,2,3. The first stakeholder is the Project Sponsor, the second is the Project Manager. and the third is the Developer. We denote by P(j) the importance level of each stakeholder which illustrates the power involved in determining which ideas to implement in which release and this power corresponds to the positions in the company hierarchy.

$$P(1) = 9$$

$$P(2) = 6$$

$$P(3) = 3$$

Where

1≤ j ≤3 stakeholders

4. *Release Importance:*

Each stakeholder has voted on which release was the most important. The average importance of each release was then calculated using table 5 and equation (5). It was found that the $\boldsymbol{\xi_1}$= 26.7% of release 1 and $\boldsymbol{\xi_2}$= 73.3%. of release 2.

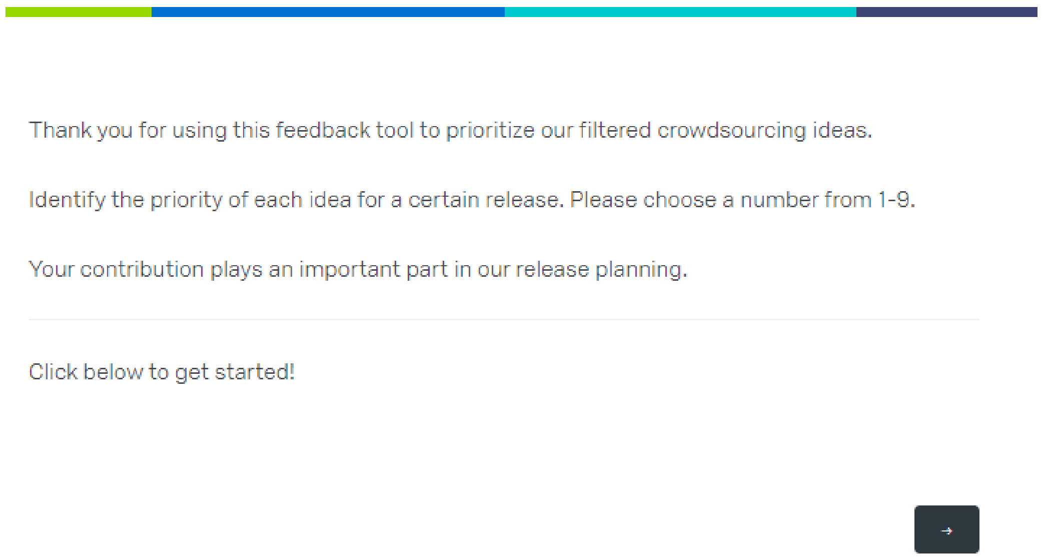
Release	$\boldsymbol{\xi(k,1)}$	$\boldsymbol{\xi(k,2)}$	$\boldsymbol{\xi(k,3)}$	$\boldsymbol{\xi_k}$
1	40	30	10	$\frac{40+30+10}{100*3}$ = 26.7% (5)
2	60	70	90	$\frac{60+70+90}{100*3}$ = 73.3% (5)

Table 5: Release Importance

5. *Survey Results:*

We have used Qualtrics (www.qualtrics.com) which is feedback tool to be able to collect feedback from the different stakeholders on how they value features related to releases. A survey was constructed and distributed to the different stakeholders. Each of the stakeholders received the survey, checked the different ideas, and filled the corresponding value of each idea corresponding to which release it should be. Figure

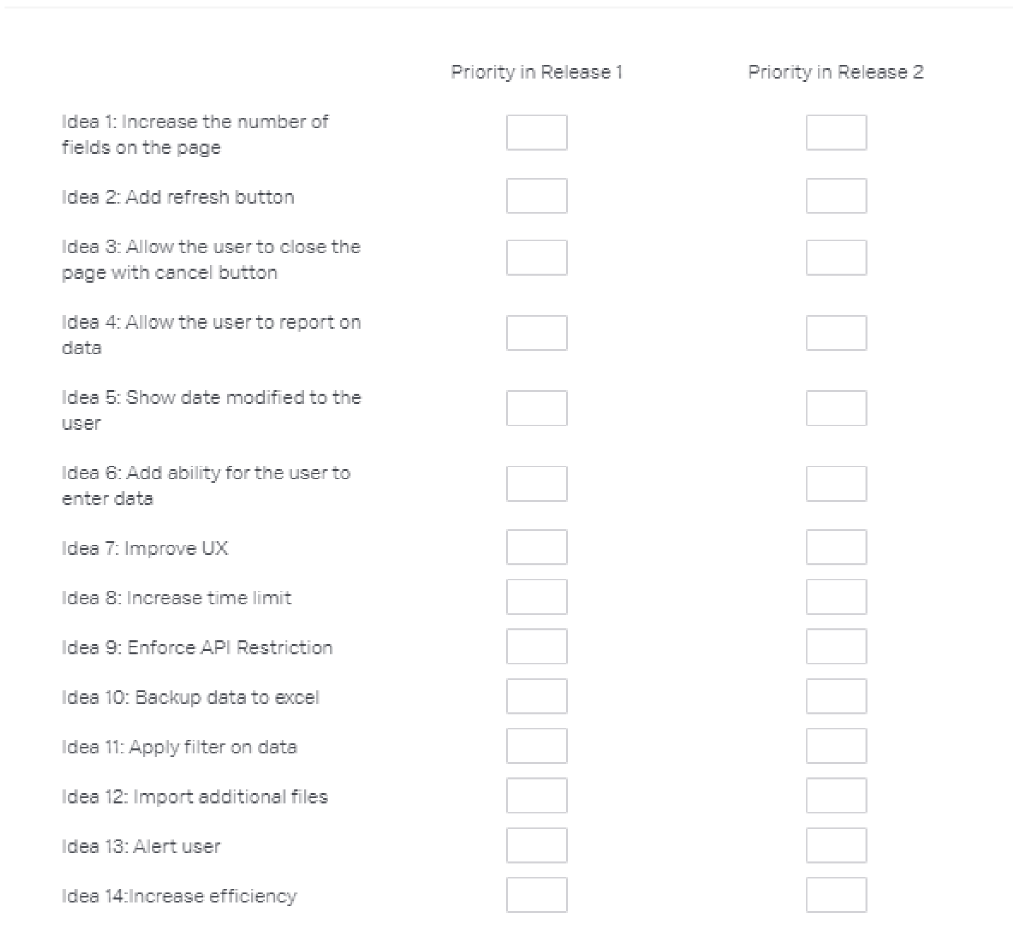
21 shows how the survey allows our different stakeholders to enter the priority of the ideas to be implemented in the next releases. The ideas described in the figure are transformed into generic labels to protect the confidentiality of the platform.


Thank you for using this feedback tool to prioritize our filtered crowdsourcing ideas.

Identify the priority of each idea for a certain release. Please choose a number from 1-9.

Your contribution plays an important part in our release planning.

Click below to get started!



	Priority in Release 1	Priority in Release 2
Idea 1: Increase the number of fields on the page		
Idea 2: Add refresh button		
Idea 3: Allow the user to close the page with cancel button		
Idea 4: Allow the user to report on data		
Idea 5: Show date modified to the user		
Idea 6: Add ability for the user to enter data		
Idea 7: Improve UX		
Idea 8: Increase time limit		
Idea 9: Enforce API Restriction		
Idea 10: Backup data to excel		
Idea 11: Apply filter on data		
Idea 12: Import additional files		
Idea 13: Alert user		
Idea 14:Increase efficiency		

Figure 21: Survey

6. Budget:

The project sponsor has then informed us that the approval for **the Budget of release 1** is $50,000 whereas the **budget allocated for release 2** is $65,000. He also informed us that the Cost per man day that is taken into consideration by the company usually in any projects delivered is equal to $500/day.

This can be illustrated for release 1 by the following:

$$\sum_{i=1}^{F} \mathbf{1.5} c_i x_{i1} \leq \$\mathbf{50,000}$$

This can be illustrated for release 2 by the following:

$$\sum_{i=1}^{F} \mathbf{1.5} c_i x_{i2} \leq \$\mathbf{65,000}$$

c_i= Number of Man days * 500$.

After analyzing the different ideas sent through the survey, each stakeholder inputted the value of a given idea for a certain release. The results of their inputs are shown in Table 6.

Idea	**Man Days**	**C**$_i$	$V(i,1,1)$	$V(i,1,2)$	$V(i,2,1)$	$V(i,2,2)$	$V(i,3,1)$	$V(i,3,2)$
1	20	10000	3	6	2	1	2	7
2	10	5000	5	9	8	3	4	9
3	24	12000	7	9	4	6	8	5
4	15	7500	8	2	9	0	9	3
5	35	17500	7	2	7	5	7	0
6	50	25000	6	7	8	4	6	2
7	38	19000	2	3	3	6	8	9
8	30	15000	5	4	9	7	5	1
9	35	17500	2	8	6	8	7	2
10	42	21000	7	5	7	9	8	5
11	20	10000	3	9	5	6	7	7

12	24	12000	3	5	3	2	0	9
13	56	28000	2	2	4	5	4	4
14	35	17500	3	4	5	6	4	4

Table 6:Stakeholders Inputs

A python code (Appendix 8.3) was run using Table 6 to deduce using a source script what ideas should be implemented in which releases and find the optimal release plan.

According to Table 6 and based on our code, we were able to deduce the following results in Table 7 related to the ideas and which releases to implement them in. Table 7 thus presents the optimal solution that illustrates which ideas should be implemented and in what releases, and Table 8 and Table 9 show the different optimal release plans for release 1 and 2.

Idea	**Release**
1	Release 1
2	Release 2
3	Release 2
4	Release 1
5	-
6	-
7	-

8	Release 1
9	-
10	-
11	Release 2
12	Release 2
13	-
14	-

Table 7: Multiple Knapsack results

Release 1	**Cost**	**WAV**
Idea 1	15,000	12.015
Idea 4	11,250	40.851
Idea 8	22,500	30.43

Table 8:Release 1 Plan

Release 2	**Cost**	**WAV**
Idea 2	7,500	92.358
Idea 3	18000	97.756
Idea 11	15,000	101.154
Idea 12	18000	61.571

Table 9:Release 2 Plan

7. ***Sensitivity Analysis:***

To understand how sensitive our data is and how prone it is to changes depending on different factors like change of release budget, we have increased and decreased the budgets of Release 1 and Release 2 as shown in Table 10&11. Our aim was to discover the cut-off points at which our release plans will change, so we increased and decreased our release 1 and release 2 budgets by 1% gradually to notice the change in the release plans.

Upon increasing or decreasing the budget of release 1 by less than 4%, our release plan remained the same. However, as shown in Table 10, our release plan changed upon increasing or decreasing release 1's budget to more than 4% and not only release 1's plan was affected but also release 2 was affected as well by this change. Only ideas 4 (release 1), 2 (release 2), and 11 (release 2) remained intact in the release plan.

Similarly, we decreased and increased release 2's budget gradually by 1 % to detect our cut-off points at which our release plan changed. Our release plan remained the same up until we decreased release 2's budget by 10% and increased it by 3%. At that stage, only ideas 4 (release 1), 2 (release 2), and 11 (release 2) remained intact in the release plan.

This information gives important information to project sponsors to acquire budget approvals knowing how sensitive the set of requirements chosen is.

Release 1 Budget	**$50,000**	**$52,000 (+ 4%)**	**$48,000 (- 4%)**
Release 1			
	1	**3**	**3**
	4	**4**	**4**
	8	**8**	**12**
Release 2			
	2	**2**	**2**

	3	**9**	**9**
	11	**11**	**11**
	12	**1**	**1**

Table 10: Budget Sensitivity Analysis for Release 1

Release 2 Budget	**$65,000**	**$58,000 (-10%)**	**67,000 (+3%)**
Release 1			
	1		**1**
	4	**4**	**4**
	8	**10**	**8**
Release 2			
	2	**2**	**2**
	3	**3**	**3**
	11	**11**	**11**
	12	**1**	**9**

Table 11: Budget Sensitivity Analysis for Release 2

8. *Validation*

Python Code Results Validation

To validate the results obtained in Table 4 & 5, we have calculated the WAV for idea 1 using a sample calculation and indeed we got WAV (1,1) = 12 for idea 1 in release 1 and CDev = 1.5 * 10,000 = $15,000

Value of idea 1 implemented in Release 1:

$$\boldsymbol{V(1,1) = P(1)V(1,1,1) + P(2)V(1,2,1) + P(3)V(1,3,1) =}$$

$$(9*3) + (6*2) + (3*2) = 45$$

$$\boldsymbol{WAV(1,1) = 0.2677 * 45 = 12.015}$$

Value of idea 1 implemented in Release 2:

$$\boldsymbol{V(1,2) = P(1)V(1,1,2) + P(2)V(1,2,2) + P(3)V(1,3,2) =}$$

$$(9*6) + (6*1) + (3*7) = 81$$

$$\boldsymbol{WAV(1,2) = 0.733 * 81 = 59.3}$$

9. Stakeholders Consensus

After running the results and validating our mathematical model using a sample, we have presented the 2 release plans shown in Table 12 on the different stakeholders to discuss them. None of the stakeholders were satisfied with the results for idea 1 as they saw it not necessary to be included in the first release and preferred to have it in release 2. However, the different stakeholders strongly agreed to have idea 4 implemented in release 1, and they were okay with idea 8 in release 1 especially the project manager who strongly wanted to implement idea 8 in Release 1. This coincides with the sensitivity analysis conducted earlier where idea 4 remained intact in the release plan no matter how much we changed the budget since it gives us a great value.

Concerning release 2's plan, the stakeholders strongly agreed on having ideas 2,3, and 11 in release 2 but disagreed to implement idea 12. Only the developer was in favor for implementing idea 12 in release 2 due to its technical but non-urgent importance and tried to convince the project sponsor and project manager to reach a consensus.

Release 1	**Release 2**
Idea 1	Idea 2
Idea 4	Idea 3
Idea 8	Idea 11
	Idea 12

Table 12: Release Plans

10. Comparison of Predicted Results with Actually Delivered Ideas

It was important to compare the actual data with our findings and predictions to validate the accuracy of our release plans with what has actually happened in the past. The ideas which were predicted to be winning ideas using our first stage filtering Random Forest Classifier are presented in Table 8 below. Table 12 compares the results of our predictions and assignment of ideas to release plans with what has happened in the past. As shown in Table 13, ideas 2,3,4,8,11 and 12 were indeed delivered before and thus were correctly predicted to be winning ideas to be assigned to release plans. In addition, ideas 4 & 8 were implemented before ideas 2,3,11, and 12 which coincides with our notion of release planning illustrated in table 7. However, idea 1 was wrongly assigned to a release plan which makes our release planning Accuracy for this run = $\frac{6}{7}$ x 100 = 85.7%. In addition, ideas 1,6, 9, and 14 were wrongly predicted to be winning ideas by our Random Forest Classifier to be winning ideas.

Idea	**Actually Delivered**	**Predicted to be Delivered using Random Forest Classifier**	**Delivered Years Ago**	**Assigned to a release using our Optimization Algorithm**
1	False	True	-	True
2	True	True	11	True
3	True	True	5	True
4	True	True	12	True
5	True	True	3	
6	False	True	-	
7	True	True	5	
8	True	True	12	True
9	False	True	-	
10	True	True	11	
11	True	True	5	True
12	True	True	2	True
13	True	True	13	
14	False	True	-	

Table 13: Actual Ideas Delivered Vs Ideas Predicted to be Delivered.

CHAPTER V

CONCLUSION

Voluminous number of ideas are being submitted to crowdsourcing platforms which make them difficult to process and filter in order to allocate them to the product roadmap and integrate them into the company's strategy. In other words, the sheer volume of data in crowdsourcing platforms make selecting the "right" idea a tedious process and not only poses a huge cognitive load on experts but also renders filtering processes that are only based on computational algorithms as lacking to the importance of human intuition and experience factor.

Dellermann et al. (2018) recommended that a combined human and computational method builds a strong foundation for filtering the most promising ideas, which was followed in this research. This thesis aimed to identify a way to filter ideas submitted in a seamless 2-stage filtering process combining crowd results (first filter) with management allocation decisions (second filter). The first stage created a Random Forest Classifier that predicts the most promising ideas in a crowdsourcing platform by scraping and analyzing inputted data from the crowd while the second stage tackled the resource allocation management optimization problem by introducing management's decisions.

The methodology was built and tested on a two-stage filtering process that includes a Random Forest Classifier and an optimization problem based on multiple Knapsack. It has proved to be 89% accurate in predicting promising ideas and 85% accurate in release planning.

However, our current research was only applied to one use case, but more accurate results can be achieved if it is applied to more than one platform and performance results are compared between platforms.

In addition, our approach did not consider the precedence and coupling nature between ideas. It assumes that all ideas are treated independently which is not usually the case. In some cases, an idea can only be implemented if another idea related to it is implemented in an iteration before it. However, this scenario can be incorporated by adding the following constraint to our model: If idea x_n precedes idea x_m and idea x_n is planned to be implemented at release k, then idea x_m cannot be implemented at any release 1…k-1 earlier than k.

Also, it is important to note that values given by our stakeholders in release planning for the different ideas are perceived estimates and thus prone to inaccuracy in determining the real effectivity or importance behind ideas.

In addition, our methodology does not take into consideration that a set of ideas if implemented together can create or cause a higher value than when implemented individually. This is true because sometimes ideas in synergies can produce a higher value than separate ideas and thus should be considered in bulk.

Our model also assumes that the data inputted in our idea crowdsourcing platform is from legitimate users and is not polluted by malicious automations. It assumes that automation restrictions are handled on the platform side to prevent any robotic malicious actions. Malicious activities can include intentional placement of false information, hacking attempts, botnet attacks, privacy violation attempts, and these can be prevented by different control measures (Onuchowska & de Vreede, 2018). These measures include constant monitoring of the platform and its recurring data input to protect the integrity of the crowdsourcing platform.

In addition, our model assumes that there are no duplicate ideas in the idea crowdsourcing platform from which we are scraping the ideas which is not usually the case. Sometimes similar ideas are submitted by different participants and thus a mechanism can be used to detect the similarity between ideas in the first filtering stage.

To train our Random Forest Classifier, our model only considers the dataset of ideas that were delivered. However, this dataset does not take into consideration that ideas delivered might in fact not be successful or generate revenue at the end which means that ideas delivered are not necessarily the “right” or “winning” to train the Random Forest Classifier with.

It is important to note that our Optimization model can be extended to include the commercial team and their inputs on the ROI generated from ideas and not just the priority perceived by the different stakeholders on how valuable implementing an idea is.

Our methodology also opens the possibility of building a seamless complete one user interface portal using our code that scrapes the ideas from an idea crowdsourcing platform to produce the most promising ideas to be inputted into a resource allocation optimization solution with a few clicks in a user experience fashion.

Future research can apply our proposed methodology on different implementation types of crowdsourcing platforms like crowdsourcing contests, open calls with direct rewards, or microtasks to understand the effectivity of the model on different implementation types. For instance, our model can help contest creators evaluate, predict, and plan for winning ideas for contests in a faster and more efficient manner especially that contests have a time frame.

APPENDIX

A. Web Scraping Code

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
import time
import csv
from textblob import TextBlob
import nltk
import requests
import selenium
from selenium import webdriver
from selenium.webdriver.common.by import By

#define empty lists
listComments = [None]
listVotes = [None]
listStatuses = [None]
listYears = [None]
listProfileLinks = [None]
ListIdeaLinks = [None]
listTitles = [None]
ideaProfileLinkMap = {}
ideaLinkMap = {}

#for 5 different web pages of ideas
for x in range(1, 30):
    print('Scraping webpage ')
    print(x)
    time.sleep(5)
    # scrap the web/insert website link
    html=
urlopen("https://confidentialplatform?pageNo="+x.__str__()+"&filter=Deliv
ered")
        bsObj = BeautifulSoup(html.read())

#get all ideas on the web page ( 10 ideas)
    for x in range(0, 10):
        numberOfComments = ''
    # get number of comments
        if(bsObj.findAll('a', {"id":" confidential" + x.__str__() + "
confidential"})):
            numberOfComments = bsObj.findAll('a', {"id":"confidential:" +
x.__str__() + " confidential"})[0].text.strip()
        listComments.insert(x,numberOfComments)
```

```
#export to file

    with open('ideas.csv', 'a' , newline='' , encoding="utf-8") as file:
        for x in range(0, 10):
            writer = csv.writer(file)
            writer.writerow([listVotes[x], listComments[x],
listStatuses[x],listYears[x],ListIdeaLinks[x],listProfileLinks[x]]
```

B. Random Forest Classifier

```
import matplotlib
# Load library
#import nltk
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn import svm
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split


import matplotlib.pyplot as plt
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import plot_roc_curve
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split

ideas = pd.read_csv('ideastoBeInserted2.csv',sep=',')
print('General information of ideas including head, info describe, how many
delivered or not ideas')
print(ideas.head())
print(ideas.info())
print(ideas.describe())
print(ideas['Status'].value_counts())
print(sns.countplot(ideas['Status']))
plt.show()
#get features of the model
X = ideas.drop('Status', axis = 1)
# get names of the features
X_list = list(X.columns)
#get target of the model
Y = ideas['Status']
# get target column name
Y_list = ['Status']
```

```python
# Split dataset into training set and test set
X_train, X_test , Y_train , Y_test = train_test_split(X, Y , test_size = 0.2 ,
random_state = 42)
sc = StandardScaler()
X_train =sc.fit_transform(X_train)
X_test = sc.transform(X_test)
print('Printing x train 10 records')
print(X_train[:10])

#Create a Gaussian Classifier
rfc = RandomForestClassifier(n_estimators=200)

#Train the model using the training sets y_pred=clf.predict(X_test)
rfc.fit(X_train,Y_train)
pred_rfc = rfc.predict(X_test)

# Model Accuracy, how often is the classifier correct?
from sklearn import metrics
print("Accuracy:", metrics.accuracy_score(Y_test, pred_rfc))

#predict a value
#rfc.predict([[3, 5, 4, 2]])

print(classification_report(Y_test, pred_rfc))
print(rfc.score(X_train,Y_train))

import pandas as pd
feature_imp =
pd.Series(rfc.feature_importances_,index=X_list).sort_values(ascending=False)
print(feature_imp)
import matplotlib.pyplot as plt
import seaborn as sns
# Creating a bar plot
sns.barplot(x=feature_imp, y=feature_imp.index)
# Add labels to your graph
plt.xlabel('Feature Importance Score')
plt.ylabel('Features')
plt.title("Visualizing Important Features")
plt.legend()
plt.show()

#ROC Curve
svc_disp = plot_roc_curve(rfc, X_test, Y_test)

# show the plots
plt.show()
```

C. Optimization Problem Code

```
import ortools
from ortools.linear_solver import pywraplp

def create_data_model():
    """Create the data for the example."""
    data = {}
    weights = []
    values = []
    # Defining cost dev of every feature
    costDev = [10000,5000,12000,7500,17500,25000,19000,15000,17500,
21000,10000,12000,28000,17500]
    i=0
    j=0

    # Defining the total cost of every feature - used in Constraint
    while i <= 13:
        weights.append( 1.5 * costDev[i] )
        i += 1
    print(weights)

    # Defining the values for every feature -
    # Value1 is the value given by stakeholder 1
    value1 = [[3,6], [5, 9],[7, 9], [8, 2] , [7, 2], [6, 7], [2, 3], [5, 4], [2,
8], [7, 5] , [3,9], [3,5],[2,2] ,[3,4] ]
    value2 = [[2,1], [8, 3],[4, 6], [9, 0] , [7, 5], [8, 4], [3, 6], [9, 7], [6,
8], [7, 9] , [5,6],[3,2],[4,5],[5,6] ]
    value3 = [[2,7], [4, 9],[8, 5], [9, 3] , [7, 0], [6, 2], [8, 9], [5, 1], [7,
2], [8, 5] ,[7,7],[0,9],[4,4],[4,4]]
    totalValues = []
    i=0
    j=0
    while j <= 13:
        while i<=1:
            priority = ( 9 * value1[j][i] ) + ( 6 * value2[j][i] ) + ( 3 *
value3[j][i] )
            values.append(priority)
            i +=1
        totalValues.append(values)
        values=[]
        i=0
        j += 1

    print(totalValues)

    # Defining the objective function variables
    data['releaseImportance'] =[0.267,0.733]
    data['weights'] = weights
    data['values'] = totalValues
    data['items'] = list(range(len(weights)))
    data['num_items'] = len(weights)
    num_bins = 2
    data['bins'] = list(range(num_bins))
    data['bin_capacities'] = [48000, 65000]
    return data

    # Create the mip solver with the SCIP backend.
solver = pywraplp.Solver.CreateSolver('SCIP')
data = create_data_model()
# Variables
# x[i, j] = 1 if item i is packed in bin j.
x = {}
for i in data['items']:
    for j in data['bins']:
        x[(i, j)] = solver.IntVar(0, 1, 'x_%i_%i' % (i, j))

# Constraints
# Each item can be in at most one bin.
for i in data['items']:
    solver.Add(sum(x[i, j] for j in data['bins']) <= 1)
# The amount packed in each bin cannot exceed its capacity.
for j in data['bins']:
    solver.Add(
        sum(x[(i, j)] * data['weights'][i]
            for i in data['items']) <= data['bin_capacities'][j])

# Objective
objective = solver.Objective()

for i in data['items']:
    for j in data['bins']:
        objective.SetCoefficient(x[(i, j)], data['releaseImportance'][j] *
data['values'][i][j])
objective.SetMaximization()
status = solver.Solve()
if status == pywraplp.Solver.OPTIMAL:
    print('Total packed value:', objective.Value())
    total_weight = 0
    for j in data['bins']:
        bin_weight = 0
        bin_value = 0
        print('Release ', j, '\n')
        for i in data['items']:
            if x[i, j].solution_value() > 0:
                print('Idea', i+1, '- weight:', data['weights'][i], ' value:',
                      data['releaseImportance'][j] * data['values'][i][j])
                bin_weight += data['weights'][i]
                bin_value += data['releaseImportance'][j] *
data['values'][i][j]
        print('Packed bin weight:', bin_weight)
        print('Packed bin value:', bin_value)
        print()
        total_weight += bin_weight
    print('Total packed weight:', total_weight)
else:
    print('The problem does not have an optimal solution.')
```

REFERENCES

Abhari, K., Davidson, E. J., & Xiao, B. (2016, January). Measuring the perceived functional affordances of collaborative innovation networks in social product development. In *2016 49th Hawaii International Conference on System Sciences (HICSS)* (pp. 929-938). IEEE.

Alonso, O., & Lease, M. (2011, February). Crowdsourcing 101: putting the WSDM of crowds to work for you. In *WSDM* (pp. 1-2).

Akker, J.M. & Brinkkemper, Sjaak & Diepen, Guido. (2005). Determination of the Next Release of a Software Product: an Approach using Integer Linear Programming. Proceeding of the 11th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ 2005). 161.

Ahmed, F., & Fuge, M. (2017, February). Capturing winning ideas in online design communities. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing* (pp. 1675-1687). ACM.

Brabham, D. C. (2008). Crowdsourcing as a model for problem solving: An introduction and cases. *Convergence*, *14*(1), 75-90.

Blohm, I., Leimeister, J. M., & Krcmar, H. (2013). Crowdsourcing: how to benefit from (too) many great ideas. *MIS Quarterly Executive*, *12*(4), 199-211.

Ciurumelea, Adelina & Schaufelbühl, Andreas & Panichella, Sebastiano & Gall, Harald. (2016). Analyzing Reviews and Code of Mobile Apps for Better Release Planning. 10.1109/SANER.2017.7884612.

Dellermann, D., Lipusch, N., & Li, M. (2018). Combining Humans and Machine Learning: A Novel Approach for Evaluating Crowdsourcing Contributions in Idea Contests.

Estellés-Arolas, E., & González-Ladrón-De-Guevara, F. (2012). Towards an integrated crowdsourcing definition. *Journal of Information science*, *38*(2), 189-200.

Forbes, H., & Schaefer, D. (2017). Social product development: the democratization of design, manufacture and innovation. *Procedia CIRP*, *60*, 404-409.

Forbes, H., & Schaefer, D. (2018). Crowdsourcing in Product Development: Current State and Future Research Directions. In *DS92: Proceedings of the DESIGN 2018 15th International Design Conference*, *Dubrovnik*.

Gupte, A., Joshi, S., Gadgul, P., Kadam, A., & Gupte, A. (2014). Comparative study of classification algorithms used in sentiment analysis. *International Journal of Computer Science and Information Technologies*, *5*(5), 6261-6264.

Gülçin Büyüközkan, Orhan Feyzioğlu. (2003). A new approach based on soft computing to accelerate the selection of new product ideas, Computers in Industry, Volume 54, Issue 2.

Hossain, M., & Islam, K. Z. (2015). Generating ideas on online platforms: A case study of "My Starbucks Idea". *Arab Economic and Business Journal*, *10*(2), 102-111.

Ivanko, S. L., & Pexman, P. M. (2003). Context incongruity and irony processing. *Discourse Processes*, *35*(3), 241-279.

Klein, M., & Garcia, A. C. B. (2015). High-speed idea filtering with the bag of lemons. *Decision Support Systems*, *78*, 39-50.

Karl T. Ulrich & Steven D. Eppinger. (2015). Product Design and Development, 5th Edition

Kannan, S., & Gurusamy, V. (2014, October). Preprocessing techniques for text mining. In *Conference Paper*. *India*.

Klein, M., & Garcia, A. C. B. (2015). High-speed idea filtering with the bag of lemons. *Decision Support Systems*, *78*, 39-50.

Liebrecht, C. C., Kunneman, F. A., & van Den Bosch, A. P. J. (2013). The perfect solution for detecting sarcasm in tweets# not.

Lykourentzou, I., Ahmed, F., Papastathis, C., Sadien, I., & Papangelis, K. (2018). When Crowds Give You Lemons: Filtering Innovative Ideas using a Diverse-Bag-of-Lemons Strategy. *Proceedings of the ACM on Human-Computer Interaction*, *2*(CSCW), 115.

Muenchen, B. (2017). Data Science Job Report 2017: R Passes SAS, But Python Leaves them Both Behind.

Mehndiratta, P., Sachdeva, S., & Soni, D. (2017). Detection of sarcasm in text data using deep convolutional neural networks. *Scalable Computing: Practice and Experience, 18*(3), 219-228.

Maynard, D. G., & Greenwood, M. A. (2014, March). Who cares about sarcastic tweets? investigating the impact of sarcasm on sentiment analysis. In *LREC 2014 Proceedings*. ELRA.

Ngo-The and G. Ruhe .(2009).Optimized Resource Allocation for Software Release Planning in *IEEE Transactions on Software Engineering*, vol. 35, no. 1, pp. 109-123, Jan.-Feb. 2009, doi: 10.1109/TSE.2008.80.

Ozgur, C., Colliau, T., Rogers, G., Hughes, Z., & Myer-Tyson, B. (2017). MatLab vs. Python vs. R. *Journal of Data Science*, *15*(3), 355-372.

Onuchowska, Agnieszka & de Vreede, Gert-Jan. (2018). Disruption and Deception in Crowdsourcing: Towards a Crowdsourcing Risk Framework. 10.24251/HICSS.2018.498.

Poetz, M. K., & Schreier, M. (2012). The value of crowdsourcing: can users really compete with professionals in generating new product ideas?. *Journal of product innovation management*, *29*(2), 245-256.

Qi, J., Zhang, Z., Jeon, S., & Zhou, Y. (2016). Mining customer requirements from online reviews: A product improvement perspective. *Information & Management*, *53*(8), 951-963.

Robinson, A. G., & Schroeder, D. M. (2009). The role of front-line ideas in lean performance improvement. *Quality Management Journal*, *16*(4), 27-40.

Rajasundari, T., Subathra, P., & Kumar, P. (2017). Performance analysis of topic modeling algorithms for news articles. *Journal of Advanced Research in Dynamical and Control Systems (11)*.

Shergadwala, M., Forbes, H., Schaefer, D., & Panchal, J. H. (2019). Challenges and Research Directions in Crowdsourcing for Engineering Design: An Interview Study with Industry Professionals.

Tuarob, S., Lim, S., & Tucker, C. S. (2018). Automated Discovery of Product Feature Inferences Within Large-Scale Implicit Social Media Data. *Journal of Computing and Information Science in Engineering*, *18*(2), 021017.

Tuarob, S., & Tucker, C. S. (2016, December). Automated discovery of product preferences in ubiquitous social media data: A case study of automobile market. In *2016 International Computer Science and Engineering Conference (ICSEC)* (pp. 1-6). IEEE.

Thelwall, M., Buckley, K., Paltoglou, G., Cai, D., & Kappas, A. (2010). Sentiment strength detection in short informal text. *Journal of the American Society for Information Science and Technology*, *61*(12), 2544-2558.

Westerski, A., Dalamagas, T., & Iglesias, C. A. (2013). Classifying and comparing community innovation in Idea Management Systems. *Decision Support Systems*, *54*(3), 1316-1326.

Wimmer, H., & Powell, L. M. (2016). A comparison of open source tools for data science. *Journal of Information Systems Applied Research*, *9*(2), 4.

Zhang, K., Cheng, Y., Liao, W. K., & Choudhary, A. (2011, August). Mining millions of reviews: a technique to rank products based on importance of reviews. In *Proceedings of the 13th international conference on electronic commerce* (p. 12). ACM.