UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Sofiya Demchuk

Predicting hackathon outcomes using Machine Learning (Data Analytics)

Master's Thesis (30 ECTS)

Supervisor: Irene-Angelica Chounta
Supervisor: Alexander Nolte

Tartu 2019

Predicting hackathon outcomes using Machine Learning (Data Analytics)

Abstract:
Over the past two decades, hackathons continue to increase in importance and frequency. Winning hackathon competitions can increase the visibility for winning teams and benefit participants in terms of future job opportunities, personal development and finding potential investors for a project. Based on an existing dataset that covers around 2000 hackathons and more than 60000 projects over the period of 5 years gathered from Devpost hackathon platform, in this study Data Analysis and Machine Learning techniques were used to identify aspects of hackathon teams that improve their chances of winning. This thesis is an attempt to address the gap in hackathon outcome prediction and to demonstrate the importance of different project features by presenting findings from large scope dataset. Applied techniques outline a framework for approaching the Machine Learning process on a brand-new classification problem addressing the particular difficulties and needs of the desired outcome. Naive Bayes, Logistic Regression and Random Forest were selected because they are widely in use in similar classification tasks, while XGBoost was chosen since in recent years it has given a state-of-the-art performance for different Data Science problems. Besides that, the main focus was made on project feature extraction and feature selection for a better prediction. The developed classifiers are shown to outperform the common-sense rule-based baseline.

Keywords:
Hackathon, outcome prediction, Machine Learning, Data Analytics, technologies, project, winner

CERCS: P170 - Computer science, numerical analysis, systems, control

Häkatonide tulemuste ennustamine masinõppe abil (Andmeanalüüs)

Lühikokkuvõte:
Häkatonide tähtsus ja toimumise sagedus on viimase kahe aastakümne jooksul jätkuvalt kasvanud. Häkatonide võitmine võib suurendada võitnud meeskondade tuntust ja tulla osavõtjatele kasuks töökohtade leidmisel, isikliku arengu jaoks ja projektidele investorite leidmisel. Antud uurimus tugineb olemasoleval andmestikul, mis koguti 5 aasta jooksul Devposti häkatoni platvormilt ja mis sisaldab umbes 5000 häkatoni ja enam kui 60000 projekti andmeid. Uurimuses kasutati andmeanalüüsi ja masinõppe tehnikaid tuvastamaks häkatoni meeskondade neid aspekte, mis parandavad meeskondade võiduvõimalusi. Antud töö on katse tegeleda lüngaga häkatonide tulemuste ennustamisel ja demonstreerida erinevate projekti tunnuste tähtsust suure ulatusega andmestiku uurimise tulemuste põhjal. Rakendatud tehnikad visandavad raamistiku masinõppe protsessile lähenemiseks täiesti uue klassifikatsiooni probleemi jaoks. Raamistik adresseerib antud probleemile iseäraseid raskusi ja soovitud tulemuse vajadusi. Valitud meetoditeks olid naiivne Bayes, logistiline regressioon ja juhuslik mets, kuna neid meetodeid kasutatakse laialdaselt sarnaste klassifitseerimisülesannete jaoks. Lisaks valiti XGBoost, kuna viimastel aastatel on see meetod andnud tipptasemel tulemusi erinevate andmeteaduse probleemide lahendamisel. Samuti oli fookuses projektide tunnuste leidmine ja tunnuste valik klassifikatsioonimudelite suutlikkuse parandamiseks. Töös näidatakse, et arendatud algoritmid töötavad paremini kui tavamõistusel tuginev reeglipõhine lähtetase.

Võtmesõnad:
häkaton, tulemuste ennustamine, masinõpe, andmeanalüüs, tehnoloogiad, projekt, võitja

CERCS: P170 - arvutiteadus, arvuline analüüs, süsteemid, kontroll

Contents

1 Introduction

Hackathons and hack-style events (*hack days*, *hackfests* and *codefests*) are developing rapidly in recent years and have been evolved to an intense competitions that cannot be ignored [PKI+19]. According to statistics report[1] based on data from the largest online hackathon community worldwide[2], there has been a rapid growth in a number of hackathons, almost 26% more in 2018 than in 2017 and nearly twice as many as in 2016. The data of this domain is large in size and extensive in its scope, has the characteristics of easily accessible, comprehensive and feature scalable data. Together with the number of hackathons and participants in the world the data about these events is exponentially growing and at a very fast rate. Thus, *Data Analytics* and *Machine Learning* methods can be applied to analyse the aspects of hackathons, to predict the outcome of the competition, to improve participants' skills and to help teams make strategies.

The word *hackathon* is a linguistic blend of two words: *hack* and *marathon*[3].The word *hack* refers to the activity, mainly exploratory programming or other media, where design thinking and prototyping are involved to develop usually a software project, while the word *marathon* indicates the intensity of the event and the dedication of the team to complete a goal at a given time (usually 24-48 hours) and place [KPR+15]. Usually, at the end of the event, the project prototypes are presented followed by prize-giving ceremony [KPR+15]. At the stage of demo presentation some of the projects can be ready for deploying it on an ongoing basis[4], that is why such events are interesting not only for developers but also attract sponsors [NM16]. Among top intentions why to attend hack-style competitions participants named *learning*, *networking* and *prizes winning* [GB14, KPR+15]. Companies also find it beneficial to organize a hackathon to overcome actual challenges and obstacles [KPR+15], to drive innovation across the company and to come up with new business solutions quicker avoiding traditional investing in R&D [5]. A winning hackathon can influence the project and team visibility and can help participants in terms of future employment, personal development [Hen15] and investor search.

Based on an existing data that covers around 2000 hackathons and more than 60000 projects over the period of 5 years gathered from Devpost hackathon platform, the goal of this study is to use Data Analysis and Machine Learning techniques mainly focusing on projects dataset in order to identify factors that can influence the victory of the team in hackathon. This thesis is an attempt to demonstrate the importance of different project

[1] https://corporate.hackathon.com/infographic-2018

[2] https://www.hackathon.com/

[3] https://en.wikipedia.org/wiki/Hackathon

[4] https://medium.com/@gordienok/why-does-your-company-need-a-hackathon-99e944618ff4

[5] https://gordienok.com/blog/why_does_your_company_need_a_hackathon

features on hackathon winner prediction. Applied techniques cover traditional Supervised Machine Learning methods, such as Naive Bayes, Logistic Regression, Random Forest and XGBoost classifiers. In general, this work demonstrates the application of commonly used step in Machine Learning on a brand-new classification problem addressing the particular difficulties and needs of the desired outcome.

1.1 Related work

Over the past two decades, hackathons continue to rise in importance and frequency [TC18, PKI+19] followed by an increase in the number of paper publications about these events over the past 10 years [PKI+19]. There are several papers describing the origins and the aspects of hackathons [DEV15, TKCH16, KPR+15], its phenomenon [GB14], its role in education [Kie16, SLL+15], in digital innovations [GB14, RKS14, CGJ+17] and in social ties [CGJ+17]. Multiple studies [TKCH16, TC18, Gam19] show that competition events like hackathons and codefests with their competitive environment advance hard and soft skills of participants, at the same time helping companies to achieve strategic innovation goals [FGM+18]. As a result, more and more companies and organizations are jumping on the hackathon bandwagon every year offering a new competition format and a problem to solve [BLM+17]. As a reward for the best ideas, the host organization or event sponsors might offer business collaboration to further work on the project and/or prizes [NM16, NPPPTF+18, CGJ+17]. As a year before the main objectives for organizations to run a hackathon in 2018 was to recruit top talents, collaboration with startups and to launch new products and services [6], while participants are usually looking for new ways to learn something new, to network, to win a prize and to find a sponsorship [GB14, KPR+15]. Even though, there has been a rise in the popularity of this topic lately, based on a systematic literature review [PKI+19, TC18] the related works so far were mainly focused on the structure, organizational aspects, and motivation for attending the hackathon, including some studies that focused on emphasized skills in the hack-style events and learning outcomes of the competition [NM16]. Only a few papers researched what leads to the continuation of hackathon projects[NPPPTF+18]. After literature review, it can be concluded that the studies so far were done with the little amount of projects and hackathons which does not allow us to gain strong insights about the topic of this thesis work. To this end, the contribution of this thesis is to address the gap in hackathon outcome prediction by presenting findings from large scope dataset.

[6] https://www.bemyapp.com/insights/infographics-hackathon-figures-in-2018.html

1.2 Structure of the thesis

The thesis background is given in Section 2 outlining a framework for approaching the Machine Learning process for a new classification problem. The next section is Methodology (Section 3) and it is an extension of Section 2 focusing particularly on hackathon outcome prediction problem. Thus, features, methods, and models developed for the study are explained in this section. In Section 4, details of the experiments are the results are demonstrated and discussed. Finally, the conclusion of this work together with the possible future works is addressed in Section 5.

2 Background

In this section, a summary of Supervised Machine Learning algorithms, followed by Performance Metrics and methods to deal with Class Imbalance is presented. Also, a brief description of baseline and Feature Selection techniques is reviewed in the second part of the background. Cross-Validation overview encloses the section.

2.1 Supervised Machine Learning methods

Usually, for classification problems when data is labeled, Supervised Machine Learning is applied. The supervised learning algorithm is done in two steps: first, it develops a model during a training stage based on the labeled data input and output, then once the mapping function has been learned, it should be able to predict a tag for a new unseen data.

Naive Bayes, Logistic Regression, Random Forest and XGBoost algorithms were selected among other supervised machine learning methods. Naive Bayes, Logistic Regression and Random Forest were selected because they are widely in use in similar classification tasks [BB18, Kot07] while XGBoost in recent years is gaining more and more popularity by giving a state-of-the-art performance for different Machine Learning problems [Li10, CG16].

Naive Bayes classifier is one of the simplest Machine Learning algorithm [Lew98] that belongs to the family of probabilistic classifiers. It infers the class probability by applying Bayes Theorem and the simple (naive) assumption that every pair of prediction features are conditionally independent, which it usually not the case. First the *posterior probability* [Fla12] for each class is calculated, and then *Maximum a Posteriori decision* [Fla12] rule is applied to decide what class to assign to a new unseen data point meaning that in the end the algorithm selects the tag that has higher probability.

Logistic regression [Kot07] is a technique that can be used for traditional statistics as well as Machine Learning classification problem when target variable is categorical. Logistic regression takes the probabilities output calculated by linear model and uses an *'S' shaped logistic (sigmoid) function* [Fla12] to fit into it.Then, depending on the threshold the class is assigned.

Both Random Forest and Gradient Boosting are ensemble learning algorithms based on decision trees that use *Bagging (Bootstrap aggregating)* and *Boosting* as their fundamental operations. With *Bagging* individual models are built separately and then are combined by averaging, given equal weights to each model. In contrast, with *Boosting* each next individual model is built in the sequence emphasizing the samples that were miss-classified by the previous predictor. **Random Forest** [Ho95] uses bootstrapped

samples and considers only a randomly selected subset of the variables at each step. Thus, it combines a wide variety of decision trees and that is what makes it more effective than individual decision trees [KS18]. Finally, a bunch of different settings is tested and the most accurate Random Forest is chosen.

Gradient Boosting builds new trees with fixed size trying to correct previous predictor's errors. **XGboost** [CG16] stands for 'Extreme Gradient Boosting' and is an optimized Machine Learning system under the Gradient Boosting framework. It aims to provide scalability and portability in different scenarios by getting to the extreme of machine computation limits [CG16].

2.2 Quality assessment metrics

Not only prediction models but evaluation metrics must be adapted to the specific classification problem to carry out its goals not to end up getting the most out of meaningless metric in the context of specific use cases. A properly calibrated method may achieve a lower classification accuracy which alone is not enough information to select a well-performing model, but would have a substantially higher *Recall* or *Precision* which are highly recommended additional measures to evaluate a classifier[7]. In order to define these metrics a confusion matrix will be used (Fig. 1),

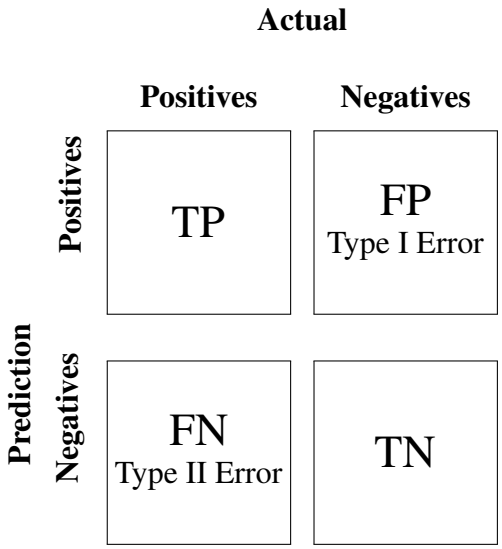

Figure 1. Confusion Matrix

[7] https://machinelearningmastery.com/classification-accuracy-is-not-enough-more-performance-measures-you-can-use/

where *True Positives (TP)* - positives that were correctly classified,

False Positives (FP) - positives that were falsely classified,

True Negatives (TN) - negatives that were correctly classified,

False Negatives (FN) - negatives that were falsely classified as positives.

One of the most universal metrics is Accuracy which might be misleading in case with imbalanced data, thus putting more emphasis on Cohen's Kappa instead would be more useful. *Kappa* is similar to Accuracy, but it is normalized at the baseline of random chance for a given dataset. It can range from `-1` to `1` and the higher model's Kappa score is, the better model is comparing to random guess classifier. According to Fig. 1 all the Performance metrics can be explained now by equations:

$$\texttt{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}$$

- the overall performance of the model,

$$\textit{Recall (Sensitivity or True Positive Rate)} = \frac{TP}{TP + FN}$$

- the coverage of actual positives that were correctly classified,

$$\textit{Specificity (True Negative Rate)} = \frac{TN}{TN + FP}$$

- the coverage of actual negatives that were correctly classified,

$$\textit{Precision} = \frac{TP}{TP + FP}$$

- defines how accurate the positives prediction are,

$$\textit{F1 score} = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

- harmonic mean of Precision and Recall.

ROC - curve (Receiver Operating Characteristic) curve displays the effectiveness of a binary classifier. It is represented by plot of the True Positive Rate against the False Positive Rate with a probability threshold.

AUC or Area under the ROC Curve is a value calculated as an area under the ROC and is in a range from `0` to `1`. AUC for ideal classifier is equal to `1`.

Precision-Recall Curve , similar to ROC, summarize the trade-off between the True Positive Rate and the Positive Predictive Value (Precision) for a predictive model with a probability threshold. The area under the curves can be calculated as well to compare the models.

2.3 Class imbalance

Most real-world classification problems entail a certain level of class imbalance issues. The class imbalance may cause different problems in building a prediction model, specifically it might fail to accurately classify the minority class. There are different popularly-used ways how to deal with imbalanced data [THM18].

Random Oversampling simply replicates the observations from minority class and is reasonable in case dataset is relatively small, but might lead to overfitting. In contrast, **Random Undersampling** for the majority class leads to information loss and is not recommended to use when there is not enough data.

Alternatively, there are hybrid class rebalancing methods - **Synthetic Minority Oversampling Technique (SMOTE)** based on feature space, and **Bootstrap Random Oversampling Examples Technique (ROSE)** that uses oversampling and undersampling to generate a new synthetic data in its neighborhood. A visual illustration by Tantithamthavorn et al. [THM18] of sampling methods is shown in Fig. 2 below.

Figure 2. Sampling methods illustration [THM18]

2.4 Baselines

Besides clean and accurate data when you set out to solve a new data science problem, it is very important to create a common-sense baseline first before diving in and starting to build models. A common-sense baseline solution intends to solve a problem without applying any Data Science[8]. When there is no solution provided to the current problem yet, this approach gives a benchmark to target. In addition, for a new machine learning

[8] https://towardsdatascience.com/first-create-a-common-sense-baseline-e66dbf8a8a47

problem *Base Rate* (also called *Zero R*)[9] classifier can be selected as a comparison model of lower-bound instead of Random Guess. *Zero R* algorithm in classification type of problem simply predicts the majority class. However, in this project, we aim to predict the minority class so our baseline model should also be able to do that. Since hackathon outcome prediction topic is relatively new and has not been solved by anyone yet or at least results and methods were not published yet, we have to develop a baseline ourselves[10]. Later, all other models are supposed to beat the current approach.

In the paper on baseline comparison in software engineering, *Whigham et al.*[WOM15] suggest guidelines for creating a baseline model that includes the simplicity in describing and implementation, and applicability to a range of models. Also, in the recent article by *Krishna et al.* [KM18], it is proposed that the baseline should provide comparable performance to other methods:

> *While we do not expect a baseline method to out-perform all state-of-the-art methods, for a baseline to be insightful, it needs to offer a level of performance that often approaches the state-of-the-art.*

A common-sense baseline solution is presented in the Methodology section.

2.5 Feature selection techniques

Features, referring to book [Fla12] about Machine Learning by Peter Flach, are the *'workhorses'* of Machine Learning. Thus, selecting the right features for the classification model is crucial. There are varies Feature Selection techniques that were used for different stages of this study: [11]:

- **Univariate Selection.** To select features that have the strongest correlation with the target variable different statistical tests can be used. For instance, with Pearson's Chi-Squared [Fla12] statistical test dependency between feature and target variable based on Chi-square score is determined.
- **Correlation Matrix with the Pearson correlation coefficient.** The matrix is an easy way to show the pairwise correlation between features and the dependent variable. The correlation in the matrix can be calculated with the Pearson's correlation coefficient that is measured as the covariance of two variables divided by the product of their standard deviations. With this approach, one of two highly

[9] https://gerardnico.com/data_mining/baseline

[10] https://towardsdatascience.com/how-to-construct-valuable-data-science-projects-in-the-real-world-203a4f520d54

[11] https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e

correlated features can be dropped to reduce the feature dimension since both hold the same effect on target variable as well as it can show how strong dependency they have with the target variable.

- **Backward Elimination with p-value**. Probability value shows the probability of finding an observation under an assumption that some statistical hypothesis is true. And by using it, we accept or reject that hypothesis. P-value of 5% (.05) or less is a widely used cut-off point. Together with Backward Elimination, features with the highest p-value (but > 5% threshold) are removed one by one until the final set of features (all less than a cut-off point) is significant to predict the outcome.

- **Feature Importance based on model results.** Also, feature usefulness can be checked with Random Forest model property [Bre01]. By this approach for each feature *Mean Decrease Accuracy* and *Mean Decrease Gini* are calculated. *Mean Decrease Gini* is based on *Gini impurity* metric used for measuring the splits in trees and shows how much each variable decreases weighted impurity in it. *Mean Decrease Accuracy* simply determines how much model accuracy decreases if we drop that variable. The higher scores, the higher Random Forest ranks the importance of the feature.

2.6 Cross-Validation

As illustrated by empirically [AKB18], it is important to do cross-validation to validate Machine Learning model stability and to check how it generalizes to an unseen dataset. Cross-validation is necessary to estimate how accurately will be our prediction in practice, or other words, is our predictive model low on bias and variance. Since we are experimenting with several machine learning models on a brand new classification problem, validation helps us to first choose a model, and then to determine the hyper-parameters of the model for a new independent data. Cross-validation is a technique that randomly divides the original dataset into K subsamples, where K-*1* subsamples are used to train model, and the remaining sample is used to validate it. This process is then repeated K times, where each subsample is used as the validation set. This way, K fold cross-validation may significantly reduce the overfitting [Sch93, Koh95].

3 Methodology

In this section, the hackathon data and the methods used for hackathon outcome prediction are described. First, a detailed data pre-processing and analysis will be provided, followed by Feature Extraction and Feature Engineering. Then baselines for comparison are outlined. And the last subsection introduces the metrics important for winner classification evaluation.

3.1 Data

3.1.1 Data Collection

The data was extracted from Devpost site[12], the hackathon platform that helps to participate in and organize online and in-person competitions, also it provides an opportunity for software engineers to find a job. Each website page was web scrapped and represented in JSON format comprising many attributes. All the data about projects and team members is curated by the participants themselves meaning that the content does not get any formal review. The example of JSON file structure can be found in Appendix (Fig. 15).

Two datasets extracted from merged JSON files were used for this research: hackathon dataset and project dataset. The last one includes information about the project and a piece of brief information about the team. Two datasets were merged into one dataframe and were pre-processed.

3.1.2 Data Preprocessing

Devpost platform allows anyone to organize a hack competition, to add a project and to manage the content by oneself. This caused a lot of data cleaning that included the following steps:

- Removing observations from the dataset where data was missing for most of the important attributes where the value should be present (e.g. no team members).
- Handling missing values: some missing values were replaced with zeroes, for instance, team may not fulfill all the details about the project on Devpost page where fields are optional.
- Removing hackathons and their submitted projects where there are no winners marked, or zero non-winners.
- Keeping hackathons only with more than 5 submissions (projects) in order to be considered as valid ones.

[12] `https://devpost.com/`

- Removing hackathons and their submitted projects where the majority of submissions were marked as winners. Often, those are competitions organised by small untrusted organisations with the little amount of submissions and many comforting prizes.
- Dropping projects that did not participate in any hackathon.
- Handling names mismatch of technologies (eg. *node-js* and *node.js*) that were used to build a project.
- Converting time to the same format.

Additionally, to make data consistent and comparable it is important to recreate the snapshot of the past. Taking into consideration this and the updates that could be done after the hackathon was held (e.g. modyfying project information and adding more likes), only the competitions that took place from March to May 2018, right before data was gathered, were taken to the final dataset. Thus, the dataset was reduced to **3288 projects** making **130 hackathons** in total. As expected, data is imbalanced and holds:

- **870** winner projects - 26.5% of all projects
- **2418** non-winners - 73.5% of all projects.

3.2 Features

Before going into the modeling stage a detailed description of projects data and its variables is given in this subsection including Feature Extraction, Engineering, and Selection that are discussed here.

3.2.1 Features Extraction

Project profile information is very likely to be influential on the success of the project in the competition. The data has been extracted and provided in the structure of a project object and some of the attributes can be easily used without any particular pre-processing. The detailed feature overview is provided in Table 1 for all extracted features.

As can be seen from the Table 1 the target variable is *winner*.

All other binary features were extracted from Devpost text fields from the project web-page, where `1` was assigned if a field was fulfilled, and `0` - it was left empty. *Subtitle* is also a text field but it is mandatory on Devpost site so the number of characters in this field was extracted instead. By using these Devpost text fields we can assume that a project profile on the web-site is a 'calling card' and has an impact on project success. The order of binary features in the Table 1 follows the same order in which the fields are listed in Devpost Project create/edit page.

Table 1. Feature Description

Feature name	Information	JSON variables	Devpost field name/description
winner	Binary: 1(winner) or 0(non-winner)	hackathon-winner	*Winner* tag
subtitle	Subtitle number of characters	project-subtitle	*What's your idea? This will be a short tagline for the project* (mandatory field)
technologies	Text: e.g. *python#node.js#css*	project-technologies	*Built With*: What languages, APIs, hardware, hosts, libraries, UI Kits or frameworks are you using?
recogn_tech	Number of technologies with URL	project-technologies	Extracted from *Built with* field
number.of.technologies	Number of technologies provided	project-technologies-used	Extracted from *Built with* field
number.of.participants	Number of team members	team-size	Extracted from *Created by* field
inspiration	Binary: 1 (provided) or 0 (not provided)	project-inspiration	*Inspiration*
purpose	Binary: 1 (provided) or 0 (not provided)	project-purpose	*What it does*
basis	Binary: 1 (provided) or 0 (not provided)	project-basis	*How we builts it*
challenges	Binary: 1 (provided) or 0 (not provided)	project-challenges	*Challenges we ran into*
accomplishments	Binary: 1 (provided) or 0 (not provided)	project-accomplishments	*Accomplishments that we're proud of*
lessonslearned	Binary: 1 (provided) or 0 (not provided)	project-lessons-learned	*What we learned*
futureplans	Binary: 1 (provided) or 0 (not provided)	project-future-plans	*What's next for <project name>*
github	Binary: 1 (provided) or 0 (not provided)	project-github	*Try it out* (URL for demo site, GitHub repo, etc.)
video	Binary: 1 (provided) or 0 (not provided)	project-video	*Video demo* (URL)
likes	Number of likes	project-likes	*Likes*
comments	Number of comments	project-number-of-comments	*Comments*

A new feature *recogn_tech* was created based on *project-technologies* object in JSON file and defines all recognizable technologies used for project development. As can be seen in JSON example (Fig.) two (*node.js* and *xero*) out of four technologies have Devpost URL that refers to the list of projects that also used those technologies, and usually, those are well-recognized ones.

Projects profile completion statistic by field is shown in Fig. 3 and additional visualizations of hackathon data distribution over the class with density curves can be found in the Appendix.

As can be seen from Fig. 3, the order of the most 'filling' fields repeats the same order of those fields on Devpost page meaning that the higher position of the field is in profile, the lower chance that team fills in this field. Taking a look at the heat-map matrix in Fig.4, a very high correlation between all profile features except *video* and *GitHub* was detected.

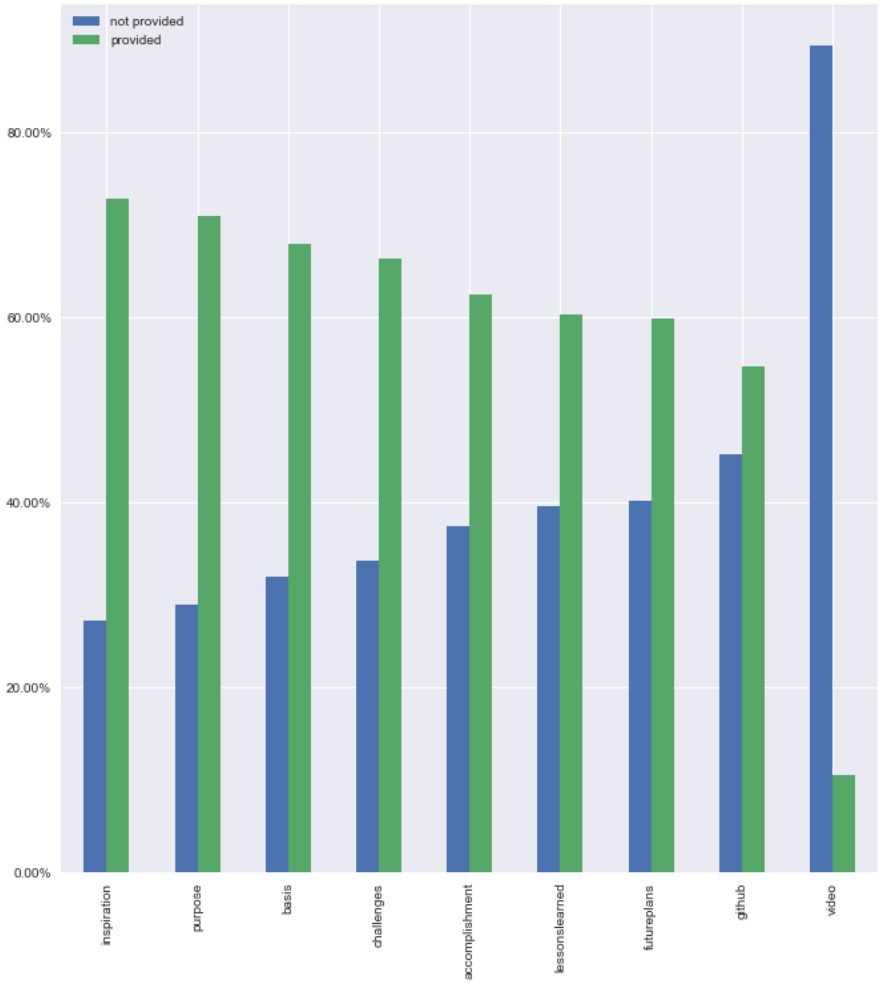

Figure 3. Project profile fields completion

3.2.2 Features Engineering

The core of any software project is the technologies used for idea development. As can be seen from data distribution (Appendix, Fig. 16) more than 50% of the project use four and more recognizable technologies to build a project. Efficient technology, adequate team knowledge, and experience are commonly specified as critical success factors across IT projects [IAF19]. Additionally, as it will be shown in Univariate selection results (Table 4, Section 3) and initial correlation matrix (Fig. 4), *number.of.technologies* feature shows high Chi-squared score and dependency concerning hackathon winner prediction. Thus, as a part of hackathon Data Analysis, some simple text data mining was used to

derive more information from this text field.

Text mining

Most frequently used tools, programming languages, libraries, etc. based on *technologies* text field were identified for all projects and winners separately. Word Cloud of Technologies for winner projects can be seen in Fig. 5.

A strong correlation with the Stackoverflow Developer Survey Result 2018[13] was found. The same as our results based on hackathon dataset, the survey's top 7 most popular technologies among professional developers include JavaScript, HTML, CSS, Python and Java. Both hackathon data and survey's statistic show that Node.js is the most commonly used platform in its category (*Frameworks, Libraries, Tools*). Based on the analysis given above we can assume that technologies used in projects are influential on project success and should be addressed as good predictive features.

New Features - Tech Score and Dummy variables

Based on *technologies* variable, where all used technologies are listed, new features can be constructed. Instead of having multiple binary features (Dummy variables) we can have one vector feature that represents the presence of each technology. To calculate a tech score the text mining results from previous subsection were used. For each technology a winning ratio was calculated:

$$tech\_win\_ratio = \frac{\#\ \textit{of wins for projects using this technology}}{\#\ \textit{of all project using this technology}}$$

Then the tech score for each project is calculated as the sum of winning ratios of all technologies used in that project:

$$scores\_tech = \sum_{i=0}^{n} tech\_win\_ratio_i,$$

where *n* is *number.of.technologies*.

For example, we can have a look at two hackathon projects and scores calculated for them in Table 2.

The *scores_tech* was calculated based on hackathon results happened from December 2017 to February 2018, the period of 3 months right before March - May 2018 (the period we use for model training). The data of December 2017 - February 2018 period

[13] https://insights.stackoverflow.com/survey/2018#most-popular-technologies

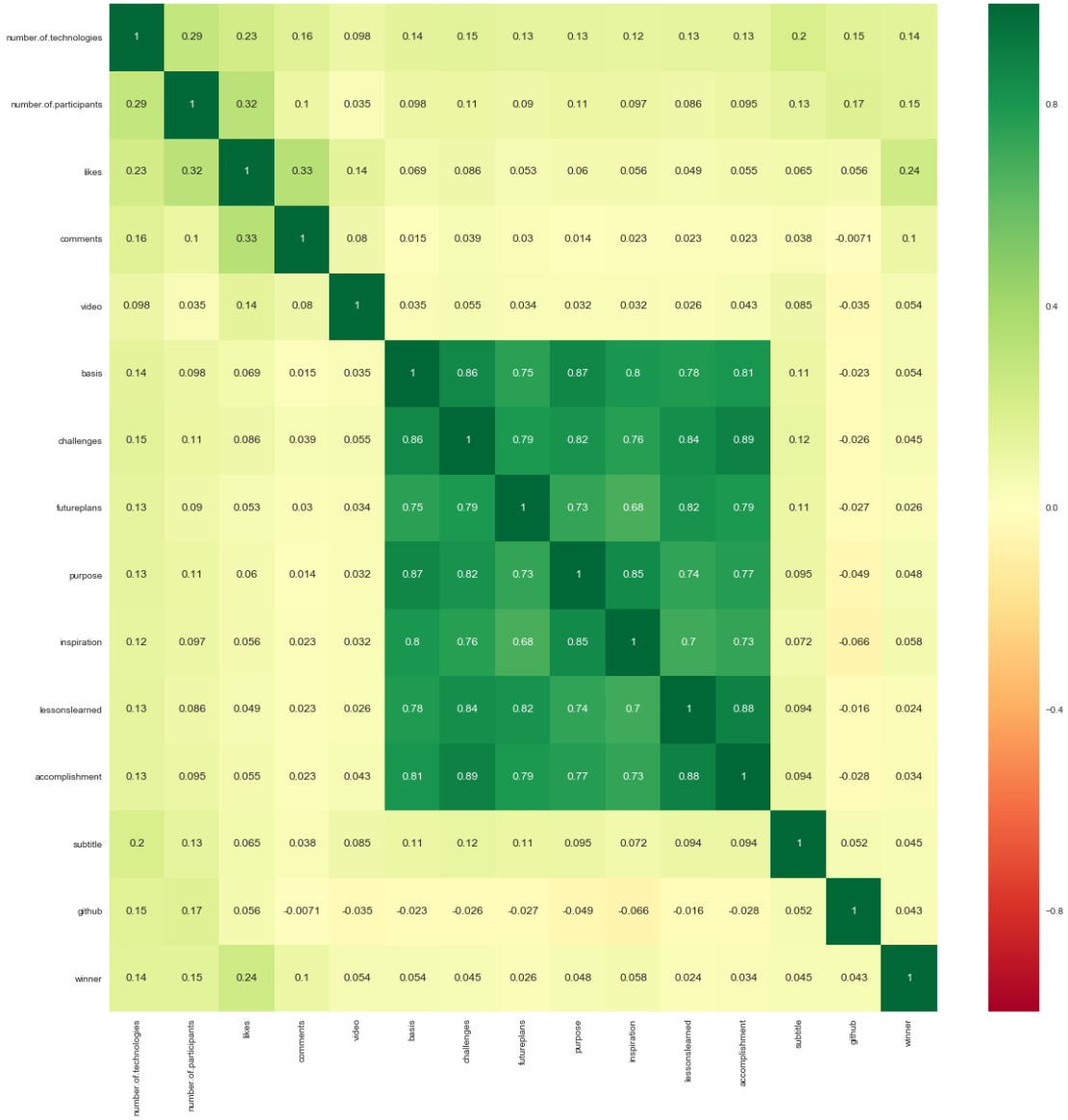

Figure 4. Initial Feature Correlation matrix

Figure 5. Technologies Word Cloud

Table 2. Project Tech Score example

project-name	technologies	number.of.technologies	scores_tech
foodcam	python#html5#google-cloud#clarifai#flask#google-web-speech-api#numpy#css#bootstrap#javascript	10	12.44
rally-ios-app	xcode#swift	2	2.15

was pre-processed the same way as stated in subsection 3.1.2, preserving the overall class distribution of the data that we use for modeling in this thesis. Since tech trends are changing rapidly, only the period of three months was chosen.

This score represents the technological impact on the project and the hypothesis would be that the existence of certain technologies can impact winning chances, for example Python tends to be a very fast-growing programming language[14] and is often chosen over other languages, this might have an impact on visibility and success of the project that hypothetically leads to the victory.

For the sake of argument, technology dummy variables were also added to the feature set in order to compare those features and *scores_tech* importance for predicting the target variable. The choice was made using technology frequency in this study and StackOverflow statistics 2018. Thus, the top six popular technologies were used as

[14] https://stackoverflow.blog/2017/09/06/incredible-growth-python/?_ga=2.85418201.1960755406.1565355798-1442620076.1549204565

binary variables for hackathon dataset: Javascript, Python, HTML, CSS, Node.js, Java, where 1 means that technology was used in the project.

3.3 Common-sense baseline

As described in subsection 2.4, a common-sense baseline has to be set for further comparison with Machine Learning models. Based on density curve and data bar plots (Fig. 16, Appendix) showing the distribution of hackathon data over the class, it was decided that simple rule baseline classifier works as follows:

Algorithm 1: Common-sense rulebased baseline

Data: Hackathon data
Result: Hackathon winner prediction
if *number.of.participants >= 4 and number.of.technologies >= 6* **then**
    winner == 1;
else
    winner == 0;

Based on this algorithm the quality assessment metrics results for hackathons dataset are presented in Table 3.

Table 3. Baseline performance results

Baseline	Accuracy	F1 score	Recall	Precision	Specificity
ZeroR (majority class prediction)	0.735	0.847	0.735	1	0
Common-sense baseline	0.713	0.82	0.9	0.745	0.17

3.4 Performance metrics

Generally, in the context of our classification problem we have to deal with the trade-off between Recall and Precision. In our situation where we want to detect instances of a minority class (the 'negatives' in the models), we are concerned more so with Precision than Recall. It is important to mention that Precision might be more useful than the False Positive Rate since it does not include the number of True Negatives in its calculation and is not affected by the imbalance. Also, in our case the minority class (winners) are 'negatives' so Specificity (True Negative Rate) should be taken into account. In addition, when comparing approaches for imbalanced classification problems, we use metrics beyond Accuracy such as F1 score, which is a harmonic mean of Recall and Precision, therefore Precision-Recall Curve and Area Under Precision-Recall Curve (AUPRC) was used instead of ROC curve.

4 Experiments and Results

For this study the data is labelled - each hackathon project is marked as *winner* or *non-winner*, so traditional Supervised Machine Learning approaches were applied. The choice for classifier was made by trying commonly used Machine Learning algorithms solving classification problems as specified is subsection 2.1 together with XGBoost. First, features that do not bring any value were removed, then Machine Learning methods were applied to original unbalanced data, and finally re-sampling techniques described in subsection 2.3 were applied to best performing models.

4.1 Feature Selection results

As a part of Feature Selection process with the help of `Python scikit-learn` library top 11 features based on *Chi-squared* Univariate statistical test are shown in Table 4 below.

Table 4. Feature selection with chi-squared

Feature	**Chi-square score**
likes	600.584972
number.of.technologies	154.159590
subtitle	102.822331
scores_tech	71.081079
comments	61.994103
recogn_tech	52.585030
number.of.participants	46.922457
video	8.449297
NODEjs	6.416052
JavaScript	6.398501
basis	3.038143

From Table 4 we can see that the only binary profile feature *basis* was included in the top list, however, it still has a very low Chi-squared score. Even though, all the binary project profile features that are text fields (from *inspiration* to *futureplans*) on Devpost have a strong correlation between each other, they have a very weak dependency with a target variable (Fig. 4). Thus all of them including *video* and *github* can be removed from the list of features to reduce the feature dimensionality and increase computational speed. Based on heat map correlation matrix (Fig. 6) one can notice that *scores_tech* and *recogn_tech* are highly dependent having Pearson's correlation coefficient equal to `0.98` which can be explained by the way how *scores_tech* was calculated. Since *scores_tech* has higher Chi-squared and is a top important feature on initial classifier modeling with Random Forest, *recogn_tech* feature was marked as redundant one and was eliminated. A new updated correlation matrix is provided in Fig. 6.

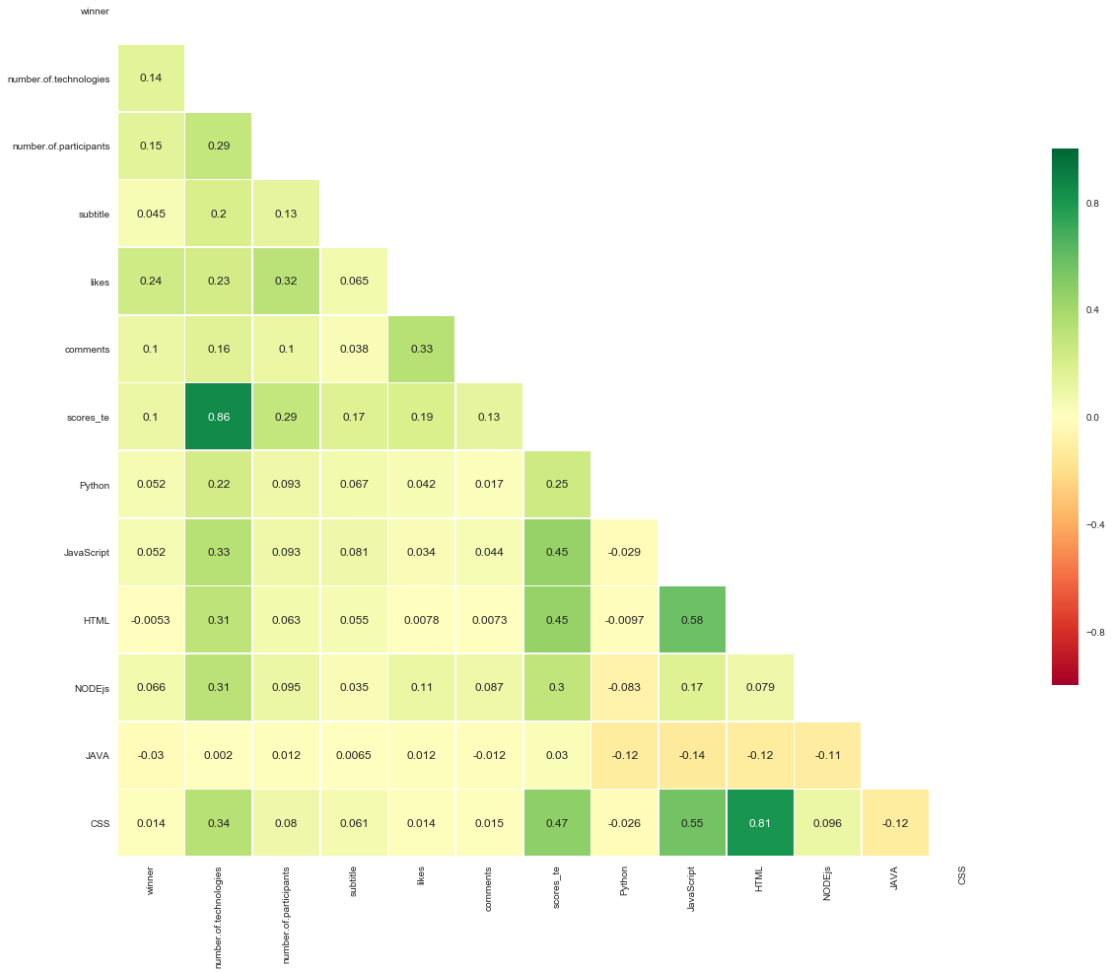

Figure 6. Feature correlation matrix

As mentioned in subsection 3.2.2, technologies dummy features such as *Python* and *JavaScript* were added in order to compare it with *scores_tech* variable. As expected, those have no significant impact on *winner* tag and did not come up in any other output list of Feature Selection techniques. As result, those were dropped. Moreover, most of them have a positive medium correlation with *scores_tech* which strengthens our conviction that a new engineered feature vector introduces the presence of all useful technologies used for hackathon project. In addition, it worth to mention that *HTML* and *CSS* binary variables have very large strength of association (Fig. 6) which makes sense since both are core technologies for Web page creation, where HTML takes care of

structure and CSS of style. Both at the same time have a high dependency with *JavaScript* which is also used in Web development. Additionally, to analyse the variables and their impact on winner tag more, **the final set of features** enough to have a significant impact in final model fit is presented in Table 5 based on Backward Elimination with p-value and 5% (.05) cut-off point.

Table 5. Backward Elimination with p-value

Feature	**p-value**
likes	< 2e-16
number.of.technologies	4.83e-06
scores_tech	0.006897
number.of.participants	0.000519
subtitle	0.007302

4.2 Supervised Machine Learning methods results

At this stage of the study, the goal of predictive modeling was to build and evaluate an actual classifier with selected features. The whole dataset was apportion into training and test sets randomly with an 80-20 split maintaining the overall class distribution.

On the initial stage of experiments we have experienced some models to perform well on the training data while showing very poor results for test dataset which means that the models captured patterns that do not generalize well to test data. Thus, for this classification task, repeated *5-fold cross-validation* with *five repeats* was used to tune parameters and to preserve the balance between bias and variance. For all the models the same seed was used to ensure the same cross-validation splits. Due to the class imbalance, Kappa metric was used instead of Accuracy, with the help of `caret` library[15] in `R` in order to choose the optimal model parameters. Additionally, `caret` library provides the opportunity to *center* and *scale* data before training as a part of parameter tuning process. *Automatic grid search* was used to find optimal parameters for each of the classification model. Naive Bayes, Logistic Regression, Random Forest and XGBoost were built and trained this way, and then tested on unseen data. It is also worth to mention again that in our case the 'negatives' of the classifier are *winners* (minority class).

First, modeling with original unbalanced data was done. Performance results of all four models can be found in Table 6. And to make it easier to compare Precision-Recall curve is presented in Fig. 7, where grey horizontal line represents a random classifier baseline.

As can be seen from the Precision-Recall curve plot, XGBoost outperfoms other models with the largest AUPRC of `0.56` and comparable F1 score of `0.84`. As expected

[15] `https://topepo.github.io/caret/model-training-and-tuning.html`

Table 6. Performance results of classifiers on original unbalanced data

	AUPRC	F1 score	Kappa	Recall	Precision	Specificity
Naive Bayes	0.44	0.84	0.08	0.99	0.73	0.05
Logistic Regression	**0.47**	**0.85**	0.1	0.99	0.74	0.08
Random Forest	0.43	0.83	0.16	0.91	0.75	**0.2**
XGBoost	**0.56**	0.84	**0.2**	0.95	0.74	**0.21**

XGBoost showed the best results due to sequent forming of individual trees taking into account previous predictor‘s miss-classification error. XGBoost classifier is followed by Logistic Regression and Naive Bayes, and only then by Random Forest. However, Kappa results give the complete picture of the performance since Cohen‘s Kappa basically shows how much better the model performs over random guess classifier in regard to class imbalance. As can be seen, XGBoost is leading in this metric as well while Naive Bayes and Logistic Regression show very poor Kappa scores. Indeed, the performance of Random Forest can be also explained by taking a look at other performance metrics comparison of the models (Fig. 8) below: XGBoost and Random Forest have the highest True Negative Rate (Specificity) which indicates winner projects that were correctly classified, but in exchange Random Forest has the lowest True Positive Rate.

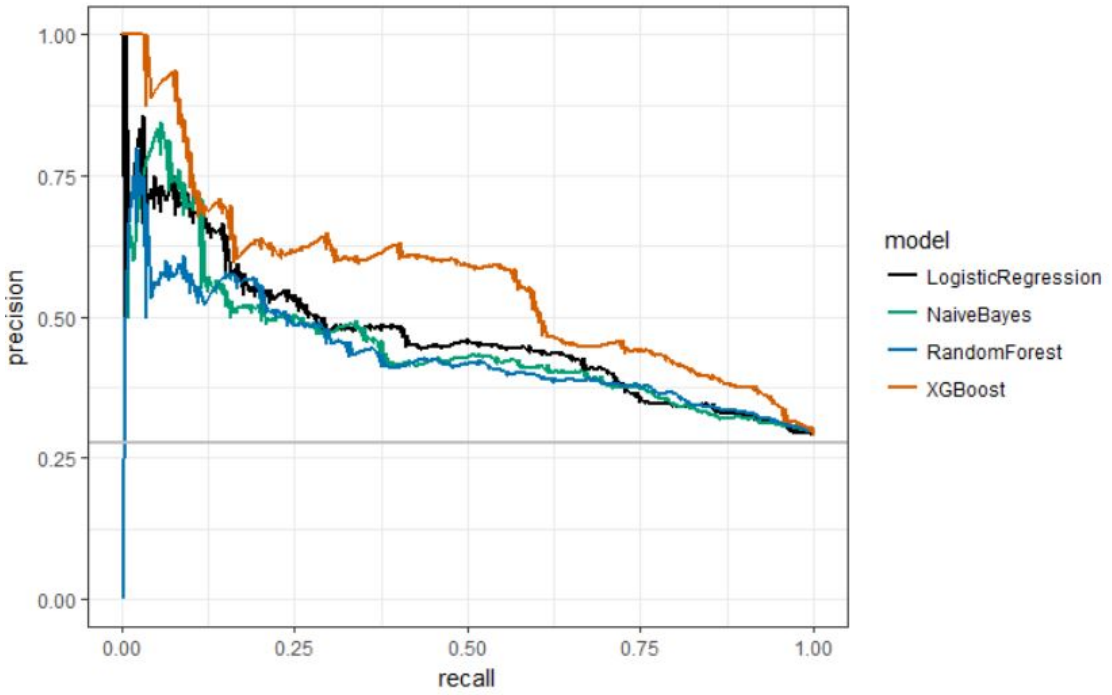

Figure 7. Precision-Recall curves for all classifiers built with unbalanced data

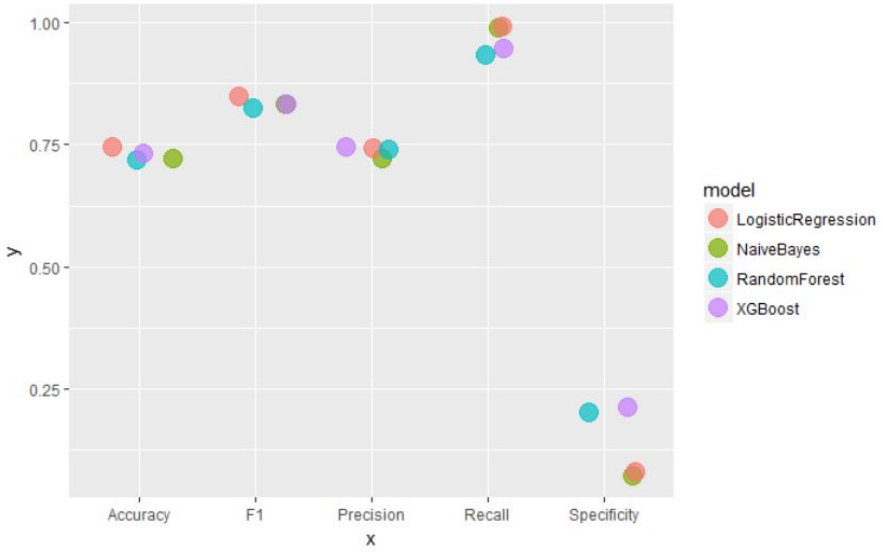

Figure 8. Performance metrics results of classifiers on original unbalanced data

To ensure the importance of Feature Selection step that was done before modeling, we also tested XGBoost with all the features and compared it to the model with final set of features (Fig. 9), where XGBoost with all features achieved only AUPRC of `0.51` and Specificity of `0.18`.

Addressing the class imbalance problem, re-sampling techniques were applied to two best performed models based on AUPRC - XGBoost and Logistic Regression, and the Precision-Recall curve results are shown in Fig. 10 and Fig. 11 accordingly.

Based on Precision-Recall curve plot and AUPRC it can be concluded that XGBoost performs the best on original unbalanced data, followed by re-samling with ROSE technique that has much lower AUPRC of `0.44`.

On the contrary, all re-sampling techniques showed a slight improvement with Logistic Regression. Random Undersampling performed almost the same as with unbalanced data, while sampling with SMOTE and ROSE achieved AUPRC of `0.48` and `0.485` respectively.

Feature importance for Machine Learning classifiers is presented in Fig. 12, Fig. 13 and Fig. 14.

Based on these results one might notice that the list of top five features is the same across the models and conforms to the list of top features selected with all the methods that were done in subsection 4.1.

As a summary, all four classifiers outperform the common-sense baseline by F1 score and Sensitivity, while only XGBoost and Random Forest beats rule-based algorithm by the True Negative Rate meaning that both predictors better classify winners. Among re-sampling techniques ROSE demonstrated the best results, although none of the re-

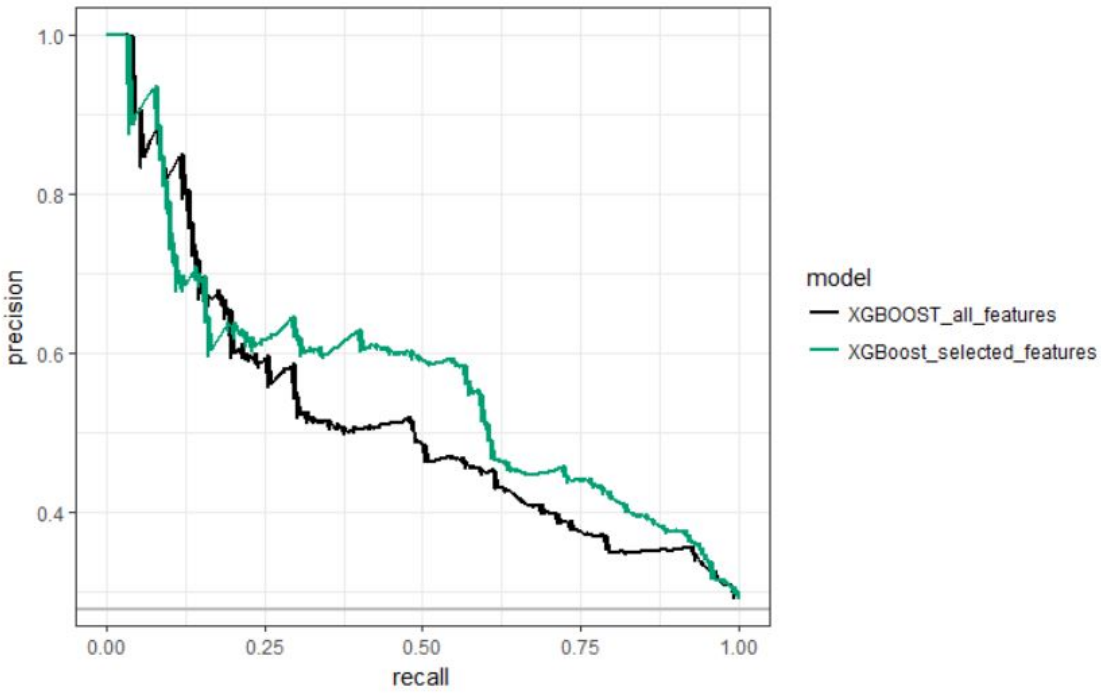

Figure 9. XGBoost with initial feature set and with selected features

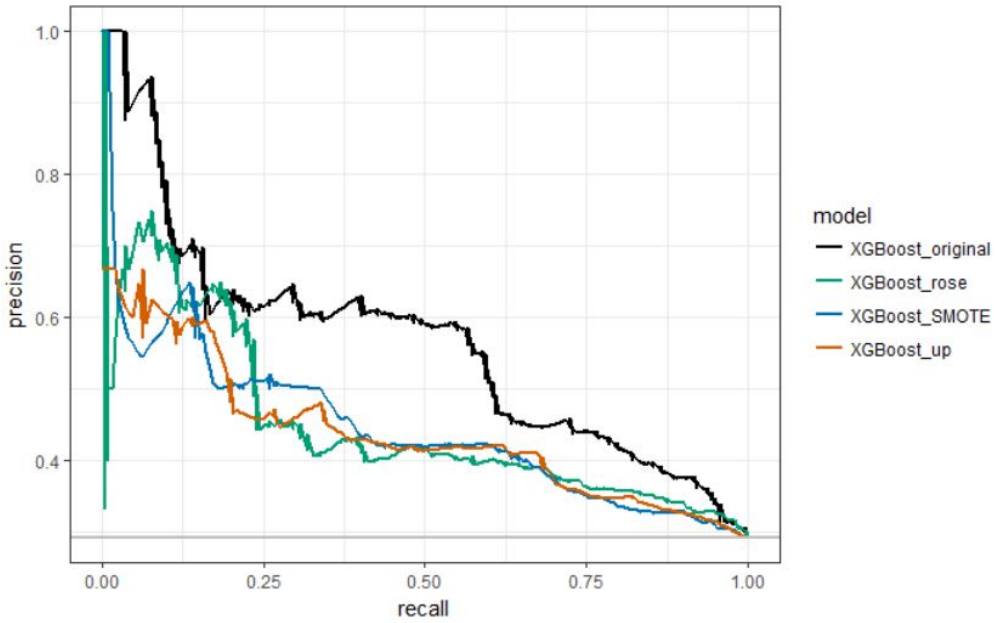

Figure 10. XGBoost P-R curves with re-sampling techniques

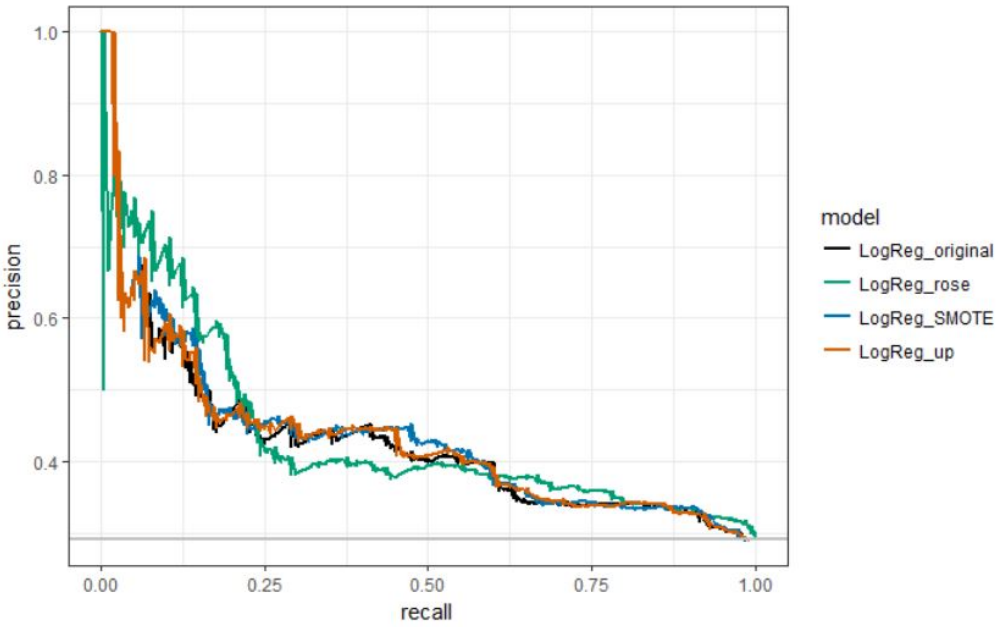

Figure 11. Logistic Regression P-R curves with re-sampling techniques

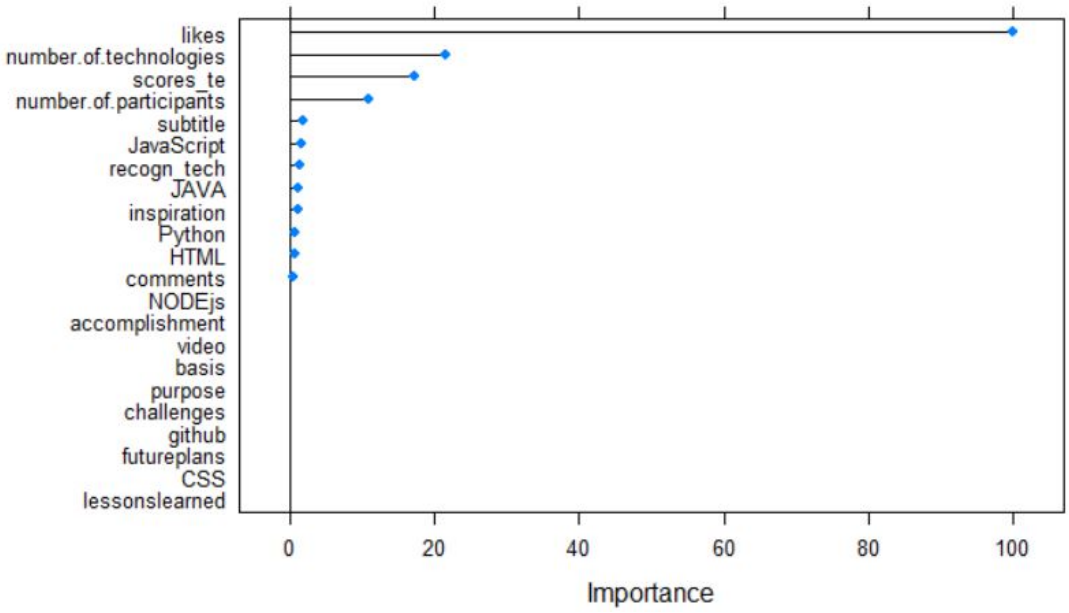

Figure 12. Feature importance of XGBoost classifier trained with initial set of features

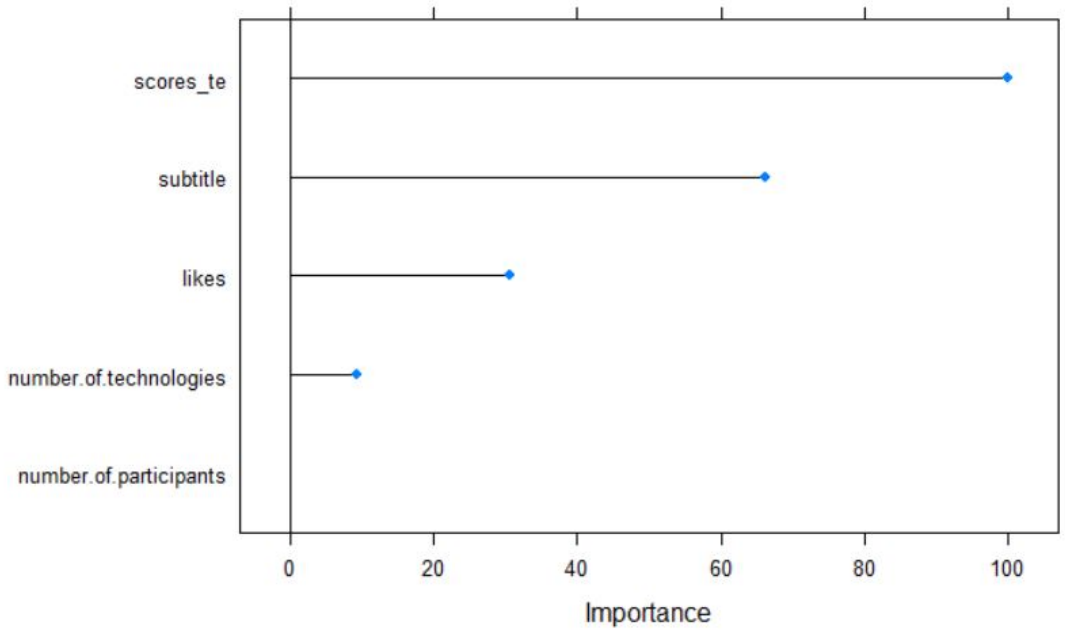

Figure 13. Feature importance of XGBoost classifier trained with final set of features (Table 5)

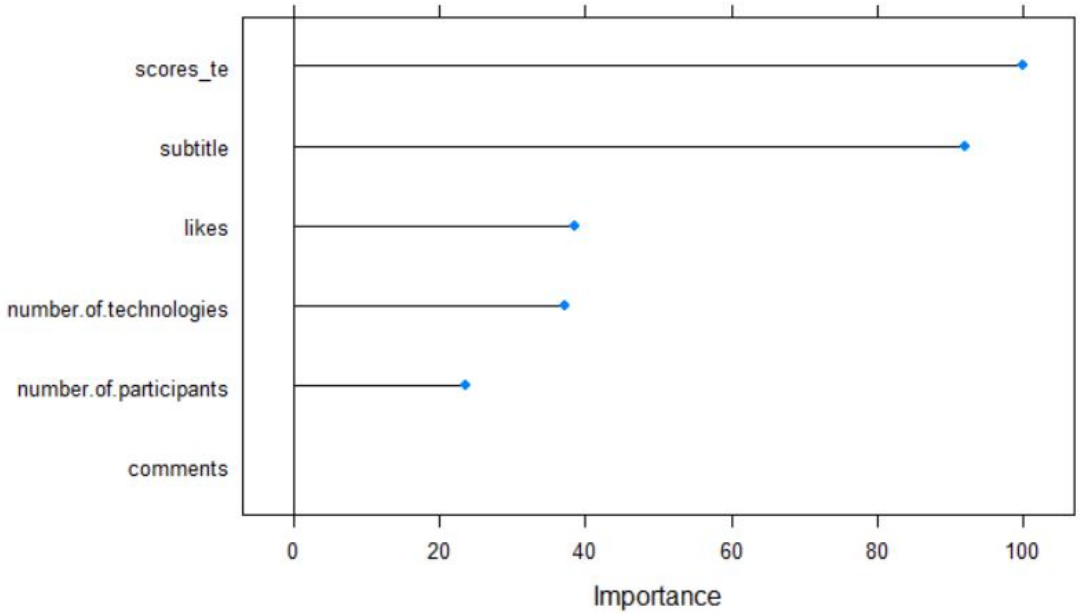

Figure 14. Feature importance of Random Forest classifier trained with final set of features (Table 4 and 5)

samplings showed a significant impact on the predictive model.

5 Conclusion and Future Work

In conclusion, the main goal of this study was fulfilled which was first to apply Data Analysis to understand and prepare hackathon data, accompanied by Machine Learning approaches for hackathon outcome prediction. The main focus in this thesis was on project features and their impact on the *winner* tag. It may be concluded that the presence of certain technologies can influence the outcome of hackathon competition for a team. A new engineered feature that represents the score of all technologies used in the project has the highest importance across most predictive models and was selected for the final set of variables by all Feature Selection approaches. In addition, *likes*, *team size* and *number of technologies* were proven to be the most useful variables for winner detection. The main motivation of this thesis was to address the gap in hackathon outcome prediction as a brand-new classification problem by passing through all stages of a Machine Learning approach. Moreover, it turned out that initial Data Analysis gave us a lot of insights about hackathon projects and those can potentially inspire the future works. For instance, Natural Language Processing can be used in order to extract more features from optional text fields on Devpost such as *Inspiration* and *Lessons Learned*. Additionally, the high positive correlation between some technologies was detected. Thus, after further analysis those can be grouped into categories, for example Web development, IOS Development or Data Science. As a part of modeling other Gradient Boosting algorithms should be used considering the fact that XGBoost performed the best in this thesis.

We hope that this study will contribute to the general understanding of hackathon topic as well as to the specific classification problem of hackathon winner prediction.

Acknowledgement

I would like to thank everyone, who have contributed to this Master's Thesis. I would like to express my sincere gratitude to my supervisor Irene-Angelica Chounta for providing her consistent support, help and valuable guidance for this study. Also, I would like to thank Alexander Nolte for giving me useful comments and suggestions for this thesis work. Additionally, I am very grateful to my friend Novin Shahroudi for methodological review of the paper and helpful comments.

Sofiya Demchuk
August, 2019

References

[AKB18] Kohei Arai, Supriya Kapoor, and Rahul Bhatia. *Intelligent Systems and Applications: Proceedings of the 2018 Intelligent Systems Conference (IntelliSys) Volume 2*. Springer Publishing Company, Incorporated, 1st edition, 2018.

[BB18] Pablo Bosch and Sandjai Bhulai. Predicting the winner of nfl-games using machine and deep learning. 2018.

[BLM+17] Nataly Birbeck, Shaun Lawson, Kellie Morrissey, Tim Rapley, and Patrick Olivier. Self harmony: Rethinking hackathons to design and critique digital technologies for those affected by self-harm. pages 146–157, 05 2017.

[Bre01] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001.

[CG16] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM.

[CGJ+17] David Cobham, Carl Gowen, Kevin Jacques, Jack Laurel, and Scott Ringham. From appfest to entrepreneurs: Using a hackathon event to seed a university student-led enterprise. pages 522–529, 03 2017.

[DEV15] A. Decker, K. Eiselt, and K. Voll. Understanding and improving the culture of hackathons: Think global hack local. In *2015 IEEE Frontiers in Education Conference (FIE)*, pages 1–8, Oct 2015.

[FGM+18] Myrna Flores, Matic Golob, Doroteja Maklin, Martin Herrera, Christopher Tucci, Ahmed Al-Ashaab, Leon Williams, Adriana Encinas, Veronica Martinez, Mohamed Zaki, Lourdes Sosa, and Karina Flores Pineda. How can hackathons accelerate corporate innovation? In Ilkyeong Moon, Gyu M. Lee, Jinwoo Park, Dimitris Kiritsis, and Gregor von Cieminski, editors, *Advances in Production Management Systems. Production Management for Data-Driven, Intelligent, Collaborative, and Sustainable Manufacturing*, pages 167–175, Cham, 2018. Springer International Publishing.

[Fla12] Peter Flach. *Machine Learning: The Art and Science of Algorithms That Make Sense of Data*. 01 2012.

[Gam19] Kiev Gama. Developing course projects in a hack day: An experience report. pages 388–394, 07 2019.

[GB14] Catherine Mulligan Gerard Briscoe. Digital innovation: The hackathon phenomenon. 2, 05 2014.

[Hen15] Scott Henderson. *Getting the Most Out of Hackathons for Social Good*, pages 182–194. 05 2015.

[Ho95] Tin Kam Ho. Random decision forests. In *Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 1) - Volume 1*, ICDAR '95, pages 278–, Washington, DC, USA, 1995. IEEE Computer Society.

[IAF19] Mohammad Ibraigheeth and Syed Abdullah Fadzli. Core factors for software projects success. *JOIV : International Journal on Informatics Visualization*, 3, 01 2019.

[Kie16] Hanna Kienzler. Learning through inquiry: a global health hackathon. *Teaching in Higher Education*, 22, 09 2016.

[KM18] R. Krishna and T. Menzies. Bellwethers: A baseline method for transfer learning. *IEEE Transactions on Software Engineering*, pages 1–1, 2018.

[Koh95] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, pages 1137–1143, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[Kot07] S. B. Kotsiantis. Supervised machine learning: A review of classification techniques. In *Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*, pages 3–24, Amsterdam, The Netherlands, The Netherlands, 2007. IOS Press.

[KPR+15] M. Komssi, D. Pichlis, M. Raatikainen, K. Kindstrom, and J. Jarvinen. What are hackathons for? *IEEE Software*, 32(05):60–67, sep 2015.

[KS18] Trace; Kirasich, Kaitlin; Smith and Bivin Sadler. 2018.

[Lew98] David D. Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In Claire Nédellec and Céline Rouveirol, editors,

Machine Learning: ECML-98, pages 4–15, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[Li10] Ping Li. Robust logitboost and adaptive base class (abc) logitboost. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, UAI'10, pages 302–311, Arlington, Virginia, United States, 2010. AUAI Press.

[NM16] Arnab Nandi and Meris Mandernach. Hackathons as an informal learning platform. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, SIGCSE '16, pages 346–351, New York, NY, USA, 2016. ACM.

[NPPPTF+18] Alexander Nolte, Ei Pa Pa Pe-Than, Anna Filippova, Christian Bird, Steve Scallen, and James Herbsleb. You hacked and now what? – exploring outcomes of a corporate hackathon. 2:129:1–129:23, 11 2018.

[PKI+19] Jari Porras, Antti Knutas, Jouni Ikonen, Ari Happonen, Jayden Khakurel, and Antti Herala. Code camps and hackathons in education-literature review and lessons learned. 01 2019.

[RKS14] Bard Rosell, Shiven Kumar, and John Shepherd. Unleashing innovation through internal hackathons. pages 1–8, 05 2014.

[Sch93] Cullen Schaffer. Selecting a classification method by cross-validation. In *Machine Learning*, pages 135–143, 1993.

[SLL+15] Bara Safarova, Edna Ledesma, Gregory Luhan, Stephen Caffey, and Cecilia Giusti. Learning from collaborative integration:the hackathon as design charrette. 01 2015.

[TC18] Nick Taylor and Loraine Clarke. Everybody's hacking: Participation and the mainstreaming of hackathons. In *CHI 2018*, Conference on Human Factors in Computing Systems - Proceedings, pages 1–2. Association for Computing Machinery, 4 2018. Funding: EPSRC (EP/N005619/1).

[THM18] Chakkrit Tantithamthavorn, Ahmed E. Hassan, and Kenichi Matsumoto. The impact of class rebalancing techniques on the performance and interpretation of defect prediction models. *IEEE Transactions on Software Engineering*, PP, 01 2018.

[TKCH16] Erik H. Trainer, Arun Kalyanasundaram, Chalalai Chaihirunkarn, and James D. Herbsleb. How to hackathon: Socio-technical tradeoffs in brief, intensive collocation. In *Proceedings of the 19th ACM Conference on*

Computer-Supported Cooperative Work & Social Computing, CSCW '16, pages 1118–1130, New York, NY, USA, 2016. ACM.

[WOM15] Peter A. Whigham, Caitlin A. Owen, and Stephen G. Macdonell. A baseline model for software effort estimation. *ACM Trans. Softw. Eng. Methodol.*, 24(3):20:1–20:11, May 2015.

Appendix

I. Additional figures and plots

project_zero-chat-bot.json


```
{"hackathon-id": "xdhax",
 "hackathon-winner": "False",
  "project-technologies-used": "5",
 "project-video": "https://www.youtube.com/embed/paE3Ytbiwb4?enablejsapi=1&hl=en_US&rel=0&start=&version=3&wmode=transparent",
 "project-basis": "We built it using Xero API, AWS lambda (node.js), AWS Lex and webhook to FB Messenger.",
 "project-technologies": [
  {"url": "https://devpost.com/software/built-with/node-js", "name": "node.js"},
  {"url": "", "name": "facebook-messenger"},
  {"url": "", "name": "aws-lambda"},
  {"url": "https://devpost.com/software/built-with/xero", "name": "xero"}],
 "project-challenges": "We did not run into any big challenges.",
 "project-future-plans": "More information will be provided using additional Xero API, such as various reports,bank transactions, purchase orders, etc.",
 "project-purpose": "The current version is a private Xero app integrated with AWS Lex and FB Messenger. You can ask about the company name you are connected to, query about the invoice status and bill status reports.",
 "hackathon-url": "https://xdhax.devpost.com/",
  "project-title": "Zero Chat Bot",
 "project-inspiration": "This is a chat bot called Zero (Xero) Chat Bot.I think it will be helpful to build a chat bot which can navigate Xero API and provide user with information thru chating.",
 "project-lessons-learned": "We learned with the Xero API which will be helpful to become a Xero partner.",
 "project-accomplishments": "Integration of Xero API, AWS Lex and FB Messenger.",
 "project-likes": "0",
 "hackathon-name": "XDHax",
 "project-subtitle": "Chating with your Accounting App",
 "project-github-url": "",
 "project-number-of-comments": "0",
 "team-size": "2",
 "project-creation-timestamp": "2017-10-12T22:40:41-04:00",
 "team": [{"participant-bubble": "I work on the Xero API and AWS Lambda",
  "participant-name": "Vincent Wong",
  "participant-desc": "",
  "participant-url": "https://devpost.com/wesee",
  "participant-id": "wesee"},
 {"participant-bubble": "I work on the lex.",
  "participant-name": "munfong",
  "participant-desc": "",
  "participant-url": "https://devpost.com/munfong",
  "participant-id": "munfong"}],
 "project-url": "https://devpost.com/software/zero-chat-bot",
 "project-id": "zero-chat-bot"}
```


Figure 15. JSON file example

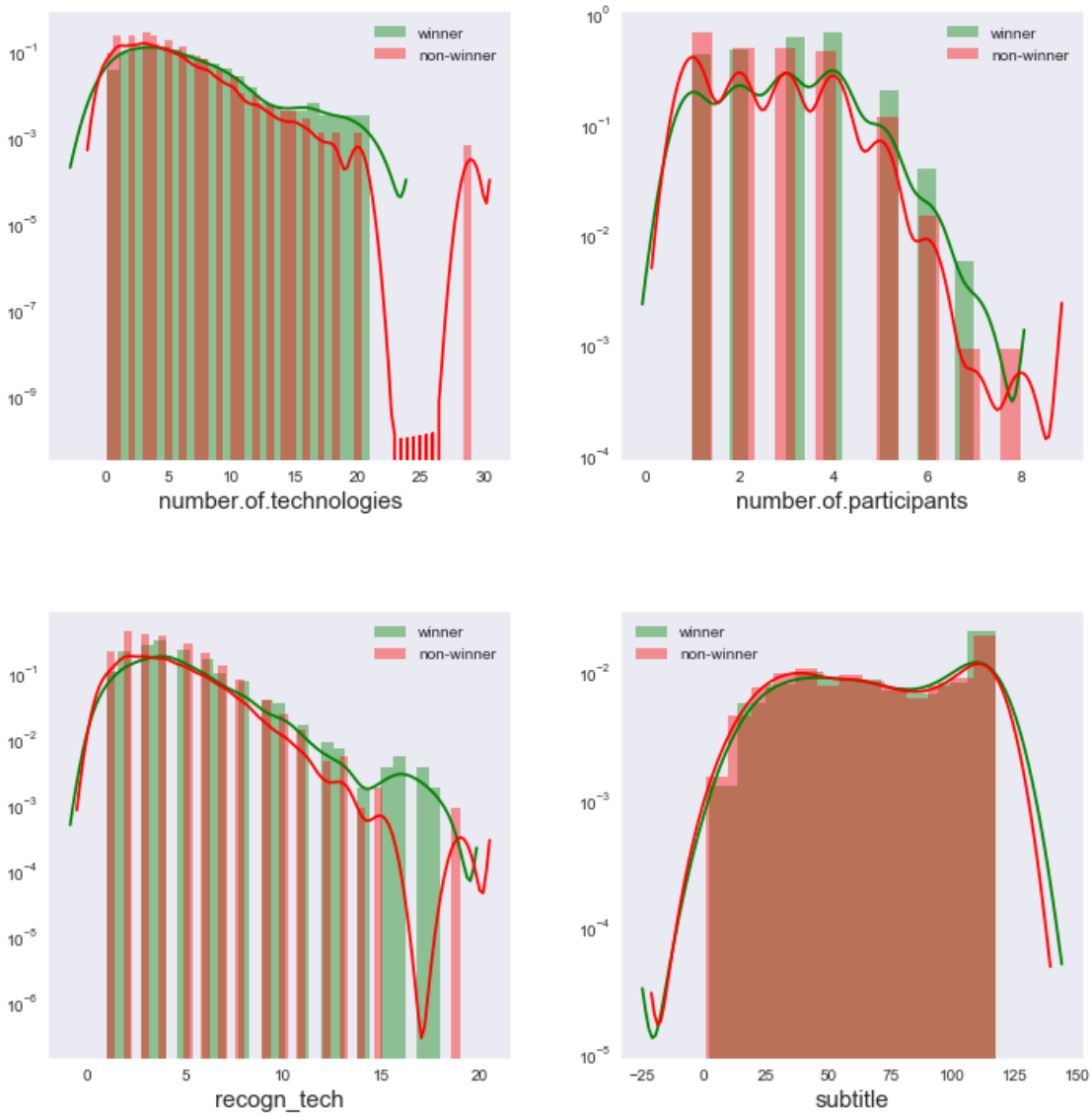

Figure 16. Hackathon data distribution by class with density curve (1)

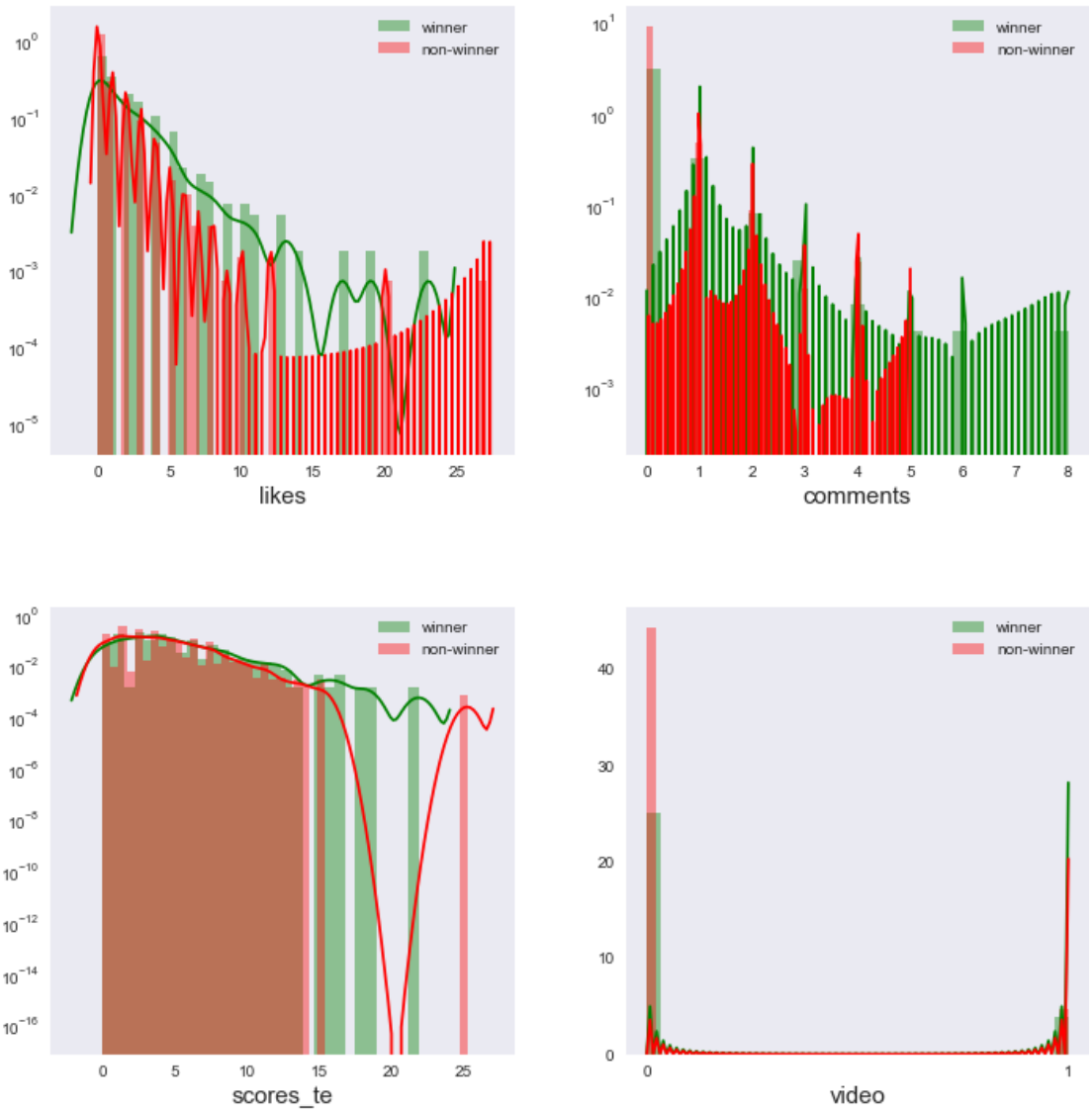

Figure 17. Hackathon data distribution by class with density curve (2)

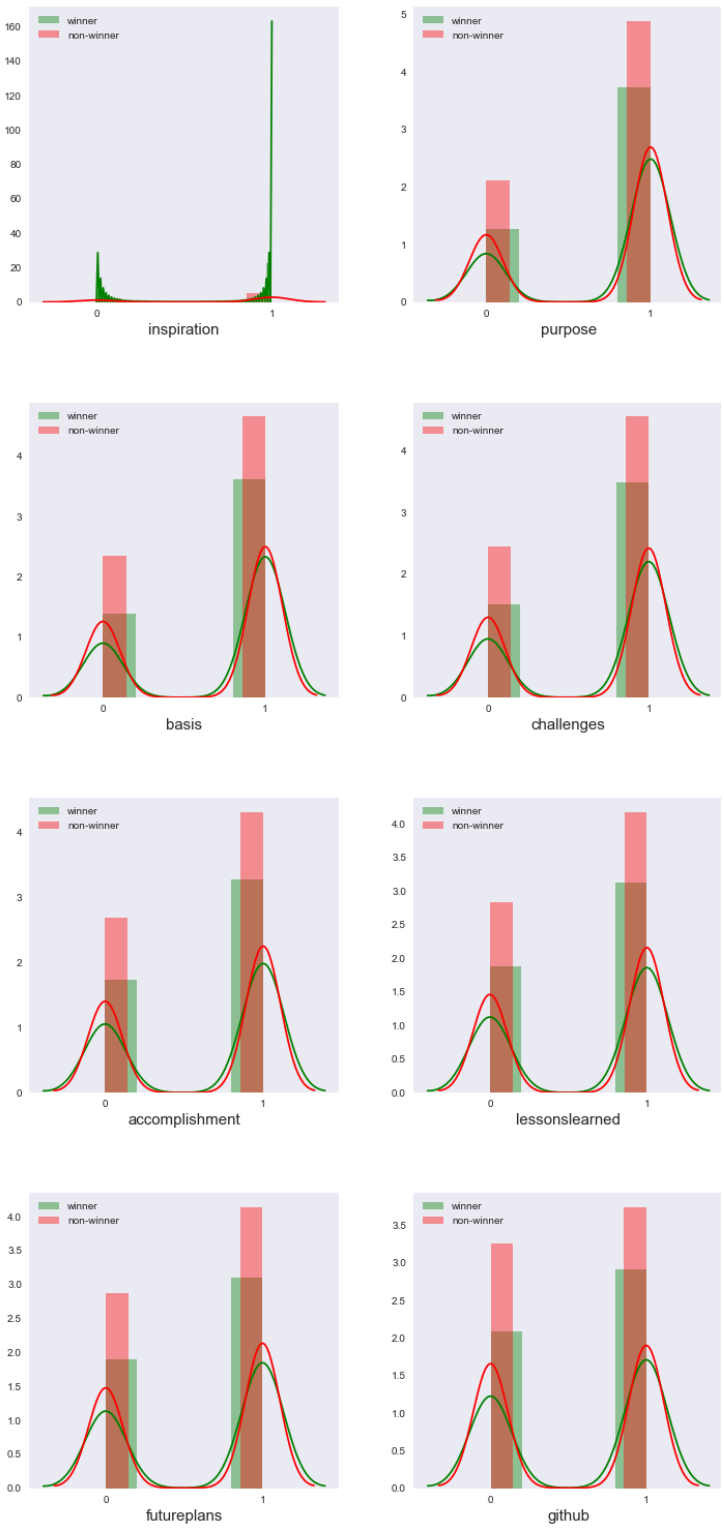

Figure 18. Hackathon data distribution by class with density curve (3)

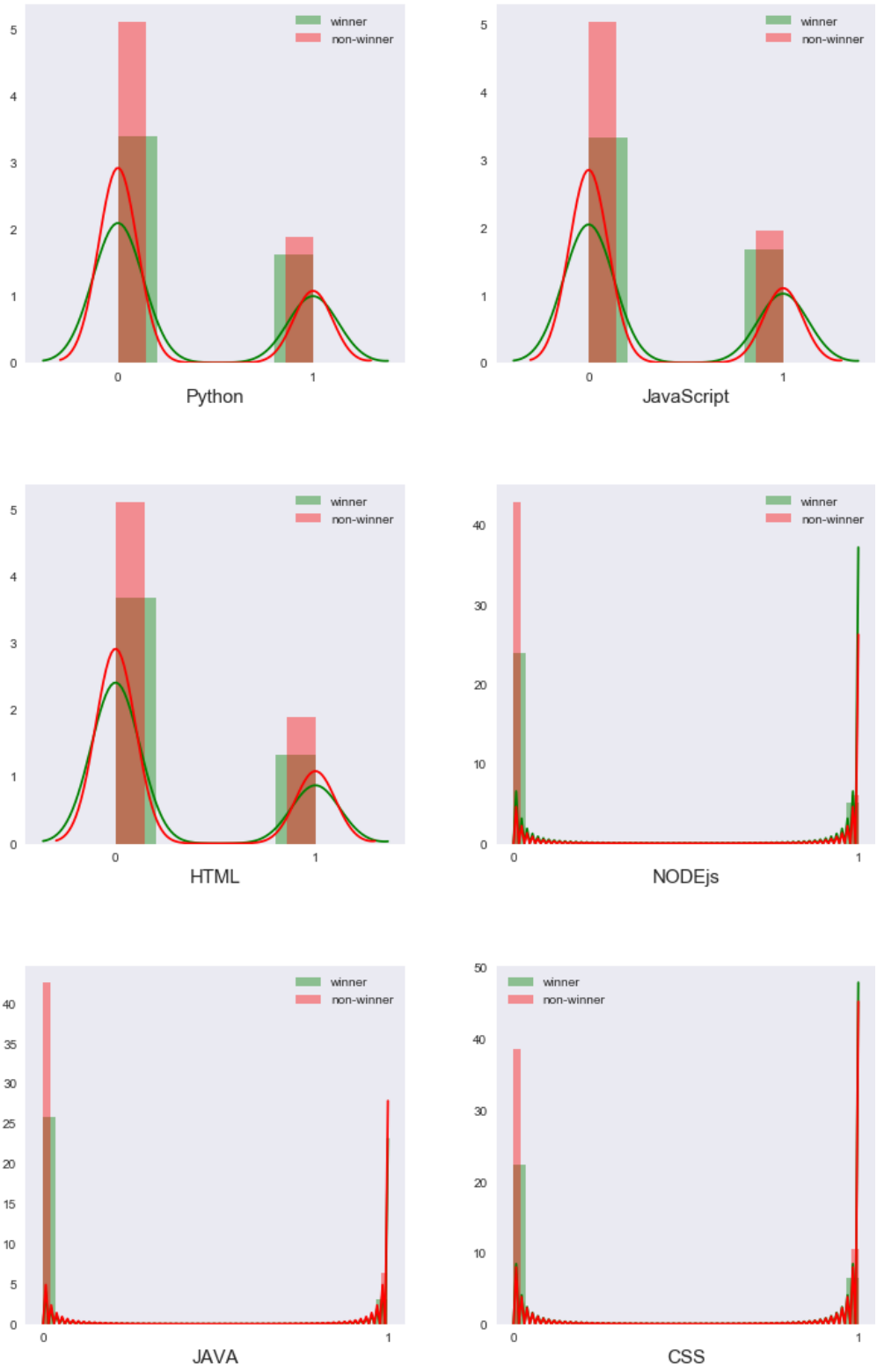

Figure 19. Hackathon data distribution by class with density curve (4)

II. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Sofiya Demchuk**,
(author's name)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Predicting hackathon outcomes using Machine Learning (Data Analytics),
   (title of thesis)

   supervised by Irene-Angelica Chounta and Alexander Nolte.
   (supervisor's name)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Sofiya Demchuk
14/08/2019