Title: Predictive Insights: Machine Learning and FOSS Project Sustainability

Project period: February 2024 - June 2024

AALBORG UNIVERSITET

Project group:
cs-24-dad-10-09

Participants:

Jonas Højen Tarp

jtarp19@student.aau.dk

Mathias Kudahl Laursen

mkla19@student.aau.dk

Steffan Klockmann

sklock18@student.aau.dk

Abstract:

This thesis explores the use of machine learning models to predict the graduation and retirement of Free and Open Source Software (FOSS) projects within the Apache Software Foundation Incubator (ASFI) using sustainability metrics. By training models with established sustainability indicators, we demonstrated predictive potential for ASFI projects. Key metrics impacting predictions included community size, development activity, growth, communication frequency, and turnover. Our findings highlight a link between sustainability and ASFI graduation and retirement, advocating for a combined qualitative and quantitative approach to improve our understanding and its prediction accuracy.

Supervisor: Adam Alami

Department of Computer Science

Date: 14/06/2024

Number of pages: 58

Contents

Summary

Motivation

Many software projects today use open source software in their development. This dependency makes it essential to understand what makes open source projects fail or succeed. The Apache Software Foundation Incubator (ASFI) provides support for a broad range of open source software projects wishing to enter the Apache Software Foundation. These project repositories are publicly available and it is possible to track their development progress from their beginnings to their outcomes of either retirement or graduation. Machine learning makes it possible to use this data to predict the success or failure of projects based health indicators from the literature.

Aim

Established indicators of FOSS sustainability are inherently present and relevant when investigating Apache projects, providing opportunities for deeper insights into the internal process of retiring and graduating these projects. The objective of this thesis is to utilize a set of FOSS sustainability indicators to train and test predictive machine learning models on the question of whether a given project would retire or fail based on its health indicators. We propose the following research question:

How can machine learning models that incorporate FOSS sustainability metrics be used to predict the likelihood of graduation and retirement for ASF incubator projects?

Methods

In the pursuit of answering our research question we utilized a variety of methods. We used Perceval to expand an existing dataset, which contained data from ASFI projects. The dataset was expanded from 217 to 224 ASFI projects, to train a variety of machine learning models to predict project graduation and retirement. While training the models, we used the stratified *k* fold technique using 10 splits. In evaluating these models, we used ROC-AUC scores, F1 scores and confusion matrices. We also investigated the feature importances from the best performing models, in order to uncover which sustainability metric had the largest impact on the models.

Results

We found that the models with the highest ROC-AUC and F1 scores were random forest, gradient boosting and logistic regression. When we evaluated the confusion matrices we found only random forest and gradient boosting showed confidence in their predictions of retired samples. We extracted the feature importances from these two models. Evaluation of these feature importances revealed that the sustainability metrics that had the largest impact on the prediction of our models were STA-8 (Size), TEC-4 (Overall dev. activity), STA-4 (Growth), COM-2 (Frequency of communication) and STA-9 (Turnover).

Conclusion

We conclude that it is possible to use machine learning models that incorporate FOSS sustainability metrics to predict the likelihood of graduation and retirement for ASFI projects. This can be done using random forest and gradient boosting algorithms, but we also recommend future studies use a dataset that contains more retired samples than the one used in this thesis, as the data imbalance proved to be a detriment to the performance of the models.

Chapter 1

Introduction

Free and Open Source Software (FOSS) has become an integral part of digital infrastructure. It is used in every part of our society, from large corporations, to small startups, to government institutions. Recent studies have shown that 90% percent of all software developed uses FOSS in some regard, which further emphasizes the critical role FOSS plays in software development[1].

FOSS development is often characterized by teams that are distributed over a wide range of geographical and organizational boundaries, and under a license that allows use, modification and redistribution of the software's source code [2]. The developers of these projects often work for free, and perform the core tasks such as developing, debugging and improving the software itself. They are often motivated to work on FOSS because they either have a personal need for what is being developed, enjoy the work in itself or want to improve their skills and reputation [3].

As a significant amount of software development projects use FOSS to some degree, it is important to understand what makes FOSS projects sustainable [4]. FOSS projects are communities that often consist of several contributors with different roles. Furthermore, the people participating in FOSS project communities often do so for different reasons[2]. Another characteristic of FOSS projects is that people can participate as they see fit, and are also able to easily leave or join the projects[2].

This alludes to a large number of different factors that determine the sustainability of a given project. This is further supported by Linåker et al. who have defined 107 different health indicators of FOSS projects[4]. This suggests that if a FOSS project wants to maintain steady progress, these health indicators should be considered.

To narrow the scope of this thesis, we will focus on FOSS projects from the Apache Software Foundation Incubator (ASFI). The ASFI provides services for projects looking to enter the Apache Software foundation. More specifically, it helps projects adapt to the Apache style of governance and operations, while also providing support to help them progress [5]. While in the incubator, a project can either graduate to become a part of the Apache Software Foundation, or retire to the Apache Attic. Project graduation occurs through a democratic process involving the voting of board members [6]. When a project retires it is often a process begun by the developers themselves, and it is rare for the board of the Apache Software Foundation to shut down a project against their will. The final retirement decision, however, always comes down to board member voting [7].

Studies such as Linåker et al. have extensively documented different indicators that reflect the sustainability of FOSS communities both on a operational and a developmental level [4]. Looking at the sustainability of FOSS projects in the context of the ASFI, these indicators may in some ways be linked to the principles of ”retirement” and ”graduation”. By making this assumption, the indicators have the potential to be used as predictors of whether a project is going to graduate and become an official Apache project or retire to the Apache Attic, where it may not be further developed[8].

By predicting if a FOSS project is going to graduate or retire, stakeholders are able to make informed decisions about the future of the project and the potential support it should receive. By further understanding which indicators have the biggest impact on the prediction of the FOSS project, we can provide valuable insight into the prediction of FOSS sustainability.

A way to establish a deeper understanding of the underlying relation between these indicators and ASFI graduation and retirement is to use machine learning to train predictive models. These models have the potential to recognize patterns and analyze health indicators, in order to create a solid framework for predicting whether a project in the ASFI should be graduated or retired. The models are able to process a large amount of different indicators with complex relations and are able to identify patterns that might not be apparent with the use of other analytical practices.

The insights gained from these predictive machine learning models can inform the Apache Foundation mentors about potential sustainability issues, and would allow them to make informed decisions on how to alleviate potential sustainability issues. By further understanding the importance of different indicators, a more targeted and informed effort could be utilized to provide more effective support, thereby enhancing the overall health of the projects. To investigate our premise, we propose the following research question:

How can machine learning models that incorporate FOSS sustainability metrics be used to predict the likelihood of graduation and retirement for ASF incubator projects?

This thesis is structured as follows: firstly, an in-depth analysis of related research is made to outline the current knowledge on the subject of sustainability of FOSS projects. Secondly, a methods chapter that describes and discusses the different methods used in this thesis and the implications they bring. Thirdly, a chapter which describes the findings and what they mean in the context of the research question. This is followed by a discussion of the results, reflections on how future studies can benefit from our findings, a chapter describing the possible threats to the validity of this thesis, and a conclusion summarizing the contributions made.

Chapter 2

Related Works

Related work includes research focusing on attempts to measure the sustainability of FOSS communities. This also includes studies that investigate health and sustainability as tangible metrics. It also lists previous studies utilizing machine learning in order to investigate sustainability. Related work also includes research that raises questions about the comparability of sustainability and retirement from the ASFI. This chapter also establishes how this thesis differs from existing literature.

2.1 Health and sustainability of FOSS communities

Attempts to document and assess the viability of a FOSS community have previously described said viability using terms such as health and sustainability, often referring to the same thing. Linåker et al. specifically refers to health as "a project's capability to stay viable and maintained over time without interruption or weakening [4]." For the sake of continuity, this thesis will refer to this capability moving forward as sustainability, whereas 'project' and 'community' are used interchangeably.

Assessing the sustainability of a community is inherently a complex task, given the range of factors one could imagine potentially affect this evaluation. For this reason, it is necessary to establish a basis for characterizing what exactly defines sustainability as a metric for FOSS communities.

In the literature, there is a tendency of retrieving project data from GitHub when assessing sustainability. The implication hereof is that while the term might be broad with varying definitions, it can usually be observed at least partially in characteristics available through data mining. Han et al. specifically focuses on the popularity of projects on their GitHub pages, using data available on the site to train machine learning models to predict popularity [9]. In practice, the work of Han et al. treats the overall popularity of a project as the success criteria, thus making it significant in predicting the sustainability of projects by their definition of the term. It is however far from the only definition in the literature.

Linåker et al. provides a framework composed of 107 sustainability characteristics categorized into 15 themes that could be considered important when evaluating sustainability [4].

While this is a considerably large number of characteristics to be aware of, it is also important to note that the study does not aim to provide guidance in terms of which characteristics to consider, or how. This leaves ample space for further exploration into how these sustainability characteristics contribute to other aspects of software development. In this thesis we refer to these characteristics as indicators and metrics interchangeably.

Linåker et al. notes that the method of measuring the sustainability of FOSS projects is still emergent, and many researchers have also previously explored the field of FOSS communities, sustainability indicators and the science of trying to understand what has positive and negative effects on FOSS project development, sometimes with different framework-based approaches.

Crowston et al. argues that it is crucial to understand a FOSS community's life cycle and the motivations of its participants in order to get a deeper understanding of why the project being worked on by its participants is successful or not [10]. The implication is that there is something inherently complex about why FOSS projects develop they way they do. The work of Crowston et al. finds that a healthy FOSS community can be visualised with an onion consisting of layers of passive users, active users, core developers, co-developers and at the centre, the founders. The underlining belief is that a sustainable FOSS project is closely tied to a community-driven backbone of active developers, testers, and founders with strong leadership skills and defined roles.

The importance of an active base of developers and testers is further amplified by the work of Wahyudin et al., in which the management and monitoring of the sustainability of FOSS projects is found to be multi-faceted and complex. Wahyudin et al. also calls for further investigation into future work focusing on enhancing the ability of FOSS stakeholders to monitor and receive early warnings about the state of the project [11]. In this regard, it might be relevant to consider an established sustainability framework to be the catalyst of such an investigation.

An example of utilizing the sustainability framework proposed by Linåker et al. can be found in Alami et al., in which they empirically explore how different sustainability indicators impact the software quality of FOSS communities, defining and using a set of sustainability indicators applied on a set of data mined from ASFI projects [12]. The same study acknowledges that the broad nature of the sustainability framework, on which it is based, means that a very large amount of indicators could be considered factors to take into account. This is why Alami et al. opted to narrow their focus from the original 107 sustainability indicators, down to a more feasible 16 sustainability metrics, across four different categories based on relevancy in the literature.

The sustainability metrics chosen by Alami et al. also exclude more qualitative indicators of sustainability that are complex to investigate, such as culture or finance, which are not well-acknowledged in the literature. Data for this is also not readily available through a simple data mining process. Likewise, this project also uses incubator projects, as our training data mainly consists of the same set of Apache projects and metrics used by Alami et al. Our study seeks to establish a narrow focus in order to reduce the ambiguity of a trained predictive model, and the set of sustainability metrics presented by Alami et al. provides ample opportunity for a focused approach to sustainability prediction with few concise metrics. This also means that we do not need to re-introduce a definition to sustainability, or evaluate the most meaningful indicators from the sustainability framework proposed by Linåker et al., as the sustainability metrics used by Alami et al. are already proven to work.

2.1.1 Predicting FOSS sustainability

The development activities of FOSS communities as a field of study is highly contested and constantly evolving, and previous efforts at predicting these activities are well documented. The literature presents numerous papers exploring the prediction of FOSS development activities, popularity, sustainability and effort estimation.

Robbles et al. provides guidelines and a tool to estimate the overall effort (months of work) needed to put into a FOSS project for it to reach completion, based on historic data of developer effort. This was previously done as a way to attempt to produce a tangible way of forecasting resources and costs associated with development [13]. Xia et al. attempted to perform a series of estimations on Apache and Linux development teams inspired by these guidelines. What they found was that it was difficult to estimate 'efforts needed to finish' among the FOSS developers, as the open source projects and the developers did not necessarily have a goal of a 'finished' state where development would no longer take place [14].

Due to the difficulties faced by Xia et al., we believe that working with, and applying sustainability indicators to ongoing projects while expecting them to be in perpetual development is beneficial, as it only emphasizes the importance of establishing the sustainability of any FOSS project at any given time. For the same reason, we believe it also makes sense for us to approach the topic of sustainability with a broad definition, and indicators that transcend 'months needed to finish' or 'monthly activity'.

In a later study, Xia et al. demonstrates that the development activity of individuals can be too random to accurately predict, but development behavior of larger groups of developers working together can be predicted with good accuracy [15]. In their approach of applying machine learning prediction to a large selection of GitHub projects, they were able to predict the value of their chosen indicators of sustainability on a project by project basis. Their definition of sustainability, however, is strictly tied to measuring monthly project activity, and thus differs from our approach to defining sustainability; our set of metrics is extensive and covers many community aspects beyond technical activities.

The data collection method used by Xia et al. centers around performing data scraping across several months in order to establish patterns of activity, which is a comparatively different approach than our thesis, as they instead focus on how developer activity is affected over time. Our study differs from that approach by instead training predictive models using data mined by capturing a snapshot of a selection of FOSS projects without taking into account how they have developed over time.

This also means that despite utilizing machine learning prediction, the findings in this thesis will inherently differ as both the definitions of sustainability and the nature of the training data are different. While the metrics defined by Xia et al. are not related to the framework by Linåker et al., it is important to note that there is no unique and consolidated definition of sustainability or project health, and we therefore expect to have widely different results than Xia et al., as our approach uses indicators of sustainability, rather than monthly activity. We also do not claim that the findings of Xia et al. are inherently inaccurate in results or methods, but instead a different and valid approach to measuring and predicting sustainability.

2.2 Apache retirement and sustainability

The work of Yehudi et al. was designed using the Community Health Analytics Open Source Software (CHAOSS) framework, in an attempt to assess the sustainability of thirty-eight open source projects by analyzing data collected over the course of a year [16]. The purpose was to examine project culture and governance, in-person events as well as online interactions, number of commits and number of pull requests. The study was unable to draw any strong conclusions however, as none of the observed projects failed or suffered significant drawbacks. The findings showed that several of the attributes stemming from the CHAOSS framework did not seem to be significant to project longevity.

It is also relevant to note that Yehudi et al. establishes an expectation of project failure in order to gain any insight into indicators that cause a project to become unsustainable. This means that gaining this insight would require coincidental failure in the sense that if none of the projects actually fail, the outcome is somewhat predictably unfulfilling. With this in mind, this thesis differs by using data from Apache incubator projects, which are inherently either retired or graduated, meaning that our findings are not dependent on waiting for an ongoing project to ’fail’.

The research by Stănciulescu et al. draws direct parallels between project graduation from the Apache Incubator program, and project sustainability. Our approach differs, as we consider FOSS sustainability to be a complex set of factors, while graduation and retirement could be considered more of a subjective evaluation [17]. While we do not consider the subjectiveness of that evaluation to be something negative, we still believe it is an important consideration when investigating FOSS sustainability. We also do not postulate that the approach of Stănciulescu et al. is wrong. Our approach differs by having a bigger emphasis on the complexities of what defines sustainability. This also means that we do not consider the ability to predict graduation or retirement necessarily equal to predicting sustainability. While this dichotomy has its merits, graduation from the incubator program is also influenced by factors unrelated to sustainability. For example, the ASFI board may decide to retire a project due to its inability to attract contributors, or due to the project idea itself being considered subjectively unappealing to the board members.

This approach to sustainability as a broader term aligns our research with Alami et al., in the sense that our investigation into sustainability prediction is also linked to the same metrics of sustainability found in the sustainability framework proposed by Linåker et al. In order to empirically explore the different sustainability metrics among 217 FOSS projects, Alami et al. examines 16 sustainability metrics across four categories to be analyzed in order to explore the impact of sustainability metrics on software quality. While we do not focus on software quality, we adopted the same set of 16 sustainability metrics. We believe that the proven ability of sustainability metrics to examine code quality impact proves their potential for the prediction of project graduation and retirement, as we are utilizing sustainability metrics as independent variables in the same manner, merely with a different research focus.

The framework of Linåker et al. specifically describes how the degree of sustainability of a project at a given time is a complex thing to measure, affected by many factors. Alami et al. specifically employs the framework to attempt to better convey sustainability as something complex, where several different indicators must be considered. For the same reason, our research takes the same holistic approach of using a collection of sustainability indicators when training machine learning models to measure sustainability.

Methods

Our thesis seeks to enhance the understanding and predictions of project outcomes within the ASFI. Specifically, we focus on determining the likelihood of whether projects will graduate to become fully-fledged ASF projects or retire. The study centers around the application of machine learning models that are trained using a set of sustainability metrics to predict the likelihood of retirement or graduation.

3.1 FOSS Sustainability Measurement

This thesis uses an adopted version of the FOSS sustainability framework used by Alami et al., which is based upon the framework developed by Linåker et al [12]. The framework of Linåker et al. is divided into a set of themes with each containing a range of indicators of sustainability, where each indicator has a distinct purpose. An example of this is the metric ”size” in the theme ”popularity”, describes how the amount of users and developers affects the resilience of a FOSS community and its ecosystem [4]. The framework of Linåker et al. is extensive with 107 sustainability metrics, divided into 15 themes, and is based on 146 related publications [4].

Alami et al. has rationalized this extensive framework by reducing the themes and their sustainability metrics, based on their prevalence in the literature [12]. The framework was further reduced by Alami et al. due to data being inaccessible because it could not be sourced with the methods used in their research [12]. This was caused by the use of Mining Software Repositories (MSR) techniques, constraining data mining of certain indicators such as finance and culture. Alami et al. acknowledges that this reduction potentially discriminates against some indicators, but still argue that the reduced framework remains extensive, with its 4 themes and 16 indicators [12]. The framework used in this thesis is almost identical to the framework used by Alami et al. The only difference being the removal of the indicator STA-5 (knowledge concentration). When we used this metric as computed by Alami et al., we observed that the scripts to re-compute the raw files were erroneous. Hence, we de-scoped the metric for practicality reasons. We opted not to re-write or debug the scripts for time constraints. As this is only a slight reduction, we argue that the framework still remains extensive with four themes and 15 indicators.

We chose to work with the indicators used by Alami et al. as they allow for a broad assessment of FOSS sustainability. The framework has been developed based on an extensive amount of literature on FOSS sustainability, and allows us to make comprehensive assessments. The reduced framework of Alami et al. includes enough indicators to remain comprehensive while fitting the scope of this thesis. We are also able to gather and use data from the Apache Software Foundation, as the framework was made with the foresight to do this [12].

3.2 Data Collection

We utilized much of the same project data as Alami et al. However, this dataset had a severe data imbalance between projects which were retired and graduated. Therefore, we opted to data mine more retired projects. The data we mined originates from ASFI projects, as this ensures continuity with existing research. The diversity of projects in the ASFI gives a broad insight into different types of FOSS projects, spanning from data processing to web development and artificial intelligence, which in turn makes the findings applicable to a wide range of FOSS projects.

The projects included in this thesis are based on the list of projects used by Alami et al. [12]. Alami et al. used the list made by Stănciulescu et al. and includes 236 projects from the ASFI[12]. We used this list as it is recent, facilitates data retrieval, and ensures transparency in project selection. Alami et al. excluded projects without project repositories in GitHub or Jira, as well as projects lacking issue trackers. Projects with unavailable defect labels were excluded as some metrics could not be computed on these. This left a total of 217 projects. We added several projects to this dataset, as it did not contain a sufficient amount retired samples, and would create a large data imbalance when training the models. This modification of the dataset resulted in an increase from 217 to 224 projects, with 187 graduated projects and 37 retired projects.

3.3 Data processing

In processing the dataset, we followed much of the methodology laid out by Alami et al. The data was mined using Python scripts as well as Perceval, with the latter being used to aggregate data components such as commits, issues and project repositories. Python scripts were used to clone repositories from GitHub, as well as extract commits from these. The syntax used can be found in appendix A.1. The scripts also served to download the issues, project repositories and repository information from GitHub and Jira. Certain software quality metrics were also extracted from the dataset using Sokrates. Finally, the data was stored in a compressed tarball file. [18] The definitions Alami et al. used to compute data for the various indicators can be viewed in tables 3.1 and 3.2 below

Metric Code	Metric Name	Description
COM-1	Response Time	Average time it takes for the first comment to appear for an issue.
COM-2	Frequency of Communication	Total number of comments in all issues plus the total number of issues.
POP-1	Project Popularity	Combined total of forks, stars, and watchers.
STA-1	Age	Age in years, calculated from the date of download since the project inception.
STA-2	Attrition	Cumulative decrease in the number of commits in periods of twelve weeks during a specified time span.
STA-3	Forks	Number of forks.
STA-4	Growth	Cumulative increase in the number of PRs in periods of twelve weeks during a specified time span.
STA-6	Life-Cycle Stage	Average number of commits per month in the last twelve months; if below zero, the project is considered dormant.
STA-7	Retention	Cumulative total of annual increases in the number of active contributors, defined by activities within consecutive 3-month periods.
STA-8	Size	Number of contributors who have engaged in at least one commit, PR, issue, or issue comment, counting each contributor only once.

Table 3.1: Summary of Metrics part 1 [18]

Metric Code	Metric Name	Description
STA-9	Turnover	Count of contributors who authored commits and have been inactive in the preceding six-month period.
TEC-1	Contributors' Development Activity	Total count of commits made by non-maintainers (contributors who have not yet merged any PR).
TEC-2	Efficiency	Time elapsed from PR creation until it is merged or closed.
TEC-3	Non-Code Contributions	Count of commits of files that are not related to programming code (file formats considered: "txt" and "md").
TEC-4	Overall Development Activity	Count of commits of coding files (excluding "txt" and "md").

Table 3.2: Summary of Metrics part 2 [18]

Our methodology kept true to the methods of Alami et al. Perceval and Git were used to mine and aggregate the various data components from both GitHub and Jira. This study deviated mostly in the case of the techniques used to mine large scale data. This means that new data was mined by manually inputting the commands in Perceval and Git, which can be seen on appendix A.1. The reason for this deviation were twofold. Firstly, this study did not have access to the precise data mining scripts used by Alami et al., so writing a new script would also be deviating from Alami et al. as it would be difficult to know whether the method was similar or not. Secondly, the scale of data mined was significantly smaller, as only 8 projects were added by this thesis, meaning that writing a script for this purpose was deemed to be excessive. In addition to the data collection and processing methods outlined previously, this thesis also introduced a new column labeled *status*. This column was specifically added to reflect the final outcome of each project within the ASFI, categorizing projects as either 'retired' or 'graduated'. This column serves as the target variable for our machine learning models, enabling them to learn and predict this binary outcome.

3.4 Models

This study investigated the use of six machine learning algorithms. These algorithms were chosen because they are supervised learning algorithms. This means there is a target variable which must be predicted[19]. Supervised learning algorithms were chosen as this study aims to be able to predict if a project from the ASFI is going to retire or graduate. These outcomes are binary, and are located in the ”status” column. The algorithms used in this thesis were chosen as they are regarded as the most commonly used [19]. The different algorithms chosen can be seen in table 3.3 below.

Decision Trees	**Random Forest**	**Logistic Regression**	**Gradient Boosting**	**Support Vector Machines**	**K-nearest Neighbours**

Table 3.3: Chosen machine learning algorithms

The complexity of sustainability indicators creates a large chance of non-linear relationships and feature interactions among the indicators. If this is the case, the gradient boosting and random forest algorithms should prove to be the most successful, as these models excel at capturing it. Decision trees are also capable of handling non-linear relationships and feature interactions, but is prone to overfitting, which could prove to be a challenge [19]. Overfitting means that a model has trained too much on a dataset, and thus learnt to predict all the ”noise” in the dataset, such as irrelevant information. This results in a model that is able to predict the information from the training data flawlessly, but will struggle severely when facing new data [20]. If the complexity of the sustainability indicators creates non-linear relationships and feature interactions, the rest of the models could possibly struggle.

3.4.1 Description of the models

A decision tree is a supervised learning algorithm that can be used for both classification and regression. It is structured like a tree, consisting of a root node, which expands into other internal nodes, which then in turn expand into branch nodes until the algorithm reaches a leaf node, which represents a possible outcome [21]. Looking at the decision tree algorithm, there is a high chance of it being affected by data imbalance. This is the case as it is a fairly simple type of algorithm, and its efficiency is mostly proven on relatively balanced data sets [22]. The simplicity of decision trees could potentially prove detrimental when paired with the complexity of sustainability indicators.

The random forest algorithm is an ensemble learning algorithm, which means that it works by running several models and synthesizing them. The random forest algorithm is essentially multiple decision trees strung together. This algorithm functions by splitting the dataset into multiple training sets as well as a testing set, which is known as an out-of-bag sample [23]. The final prediction depends on the type of task, as this algorithm can be used both for classification and regression tasks. For a classification task, the algorithm considers the most frequent variable to make the prediction. Also, for both types of tasks the out-of-bag sample is used to cross-validate, which provides the final prediction. A random forest algorithm is in many cases superior to a decision tree, as it reduces the risk of overfitting due to the out-of-bag sample and other methods that increase the diversity of the training samples. A downside of this is the increased processing time required by random forest models. [23]

Logistic regression is a supervised learning algorithm used for classification tasks. The algorithm attempts to estimate the probability of an instance belonging to a class. While there are different types of logistic regression, the one used in this study is called binomial logistic regression, as this only deals with two possible classes for a given instance. This study only considers the status of FOSS projects to be either retired or graduated, and therefore the logistic regression is binomial [24].

Gradient boosting is an ensemble learning method, which means it works by combining multiple models. Conceptually, this method of learning is similar to that of random forest.

There are two main types of ensemble methods, bagging and boosting. Whereas random forest falls under the bagging category, gradient boosting falls under the boosting category. In practice, gradient boosting sequentially builds a model by combining the predictions of multiple simpler models across several iterations, minimizing the error rate of the previous iteration. This process is repeated until reaching a predefined stopping criteria [25].

A support vector machine is a form of supervised machine learning algorithm used for classification and regression tasks. It attempts to find an optimal line that maximizes the distance between each type of class in an N-dimensional space [26].

The K-nearest neighbours algorithm is a supervised machine learning classifier. It uses the proximity of data points to make predictions about the groupings of the testing data. *K* represents the number of neighbours that has to be considered for each prediction. It is usually best to use an odd number, as this avoids ties in classification [27]. In this thesis the value of *K* was set to 3. This is a relatively simple algorithm, and is therefore easy to implement. It is however considered a ”lazy algorithm”, meaning that it takes significant computing power and is thus very time and resource-consuming, making the algorithm unable to scale effectively. Furthermore, the algorithm is prone to the ”Curse of Dimensionality” which means it struggles with classifying data points when the number of dimensions increases. In the context of this study there are 16 dimensions as these are the features of the dataset. This problem usually results in overfitting [27].

3.5 Training and testing of the models

The training and testing of the machine learning models was done using the Scikit-learn library for Python. This library contains the *train_test_split* method, which was used for both training and testing the models. The method works by splitting the dataset into two subsets, one for training and one for testing. The size of the two subsets depends on the input provided by either the *test_size* or *train_size* parameters. It is only necessary to provide a value (0.0-1.0) for one of the parameters, as they are complementary, which means if the test size is set to 0.2 the train size is automatically set to the remaining 0.8. If no input is provided, the method defaults to a test size of 0.25. [28] This study uses a train size of 0.8 and test size of 0.2. The reason for this distribution is the relatively low amount of retired project samples in the dataset. This made it necessary to dedicate a larger size of the dataset to the training size, so as to be certain the models would have enough retired samples to train on. This is also a common distribution of training and testing data [19]. The trade off meant a lower amount of retired samples in the testing set, which put an increased significance upon the retired samples used for testing, however this drawback was judged to be acceptable, as the parameters chosen to evaluate the models work well with data imbalance. By default, the dataset is shuffled before being split into subsets, but this can be turned off. For this study, the shuffle was left on. The reason for this is simply that this removes the problem of arguing for one particular order of the data during the training. It is possible to provide the *random_state* parameter with an *int* which works like a seed, in that it makes it possible to reproduce the same shuffle order by inputting the same number. If no random_state is provided, the shuffle is done at random. For the training of the models used in this study, no random seed was provided, as while inputting a random seed would make comparison between different models easier, it would be difficult to argue for one specific number. While this study is interested in investigating which models perform best, the chance of picking a number which would result in a model which was an outlier in terms of accuracy was too great. For this reason, the method of using the built-in cross-validation function from Scikit-learn and then examining the averages was superior in that it allows for a more clear image of how these models will tend to perform. The *train_test_split* method requires the user to input the features and the target variable. As this study is interested in predicting the graduation status, this was set as the target variable, and the sustainability indicators were set as the features. [28]

3.5.1 Parameters

Three parameters were used to sort through the models as well as to find the most useful algorithms: The ROC-AUC (Receiver Operating Characteristic - Area Under Curve) score and the F1 score as well as the weighted F1 score. The ROC-AUC score is a metric used for evaluating the relationship between a machine learning model's true positives and false positives, ranging from 0 to 1. A score of 0.5 is equivalent to random guessing, and a score of 1 indicates perfect performance. Generally speaking, most models do not reach a score of 1. In fact, a perfect score may be a sign of an overfitted model. A rule of thumb is that a score above 0.7 is considered acceptable, while a score above 0.8 is considered good and above 0.9 is considered great. [29] The F1 score is a combination of the precision (How many of the positive predictions were true positives?) and recall (How many of the positive samples in the dataset were correctly identified by the model) metrics. It is a good metric for measuring the performance of a model built upon an imbalanced dataset. This is because it is the *harmonic mean* of recall and precision. [30] Taking both of these into account is crucial, as an imbalanced dataset often leads to misleading precision and recall scores. For instance, consider a dataset consisting of 990 negative samples and 10 positive samples. If a model were to predict that all 1000 samples were negative it would have an accuracy score of 99%. If one does not take into account that the recall score would be 0%, the accuracy score would be able to trick an observer into thinking that the model performs well. The recall score can be similarly tricky. A model that correctly identifies 8 out of 10 positive samples but also falsely identifies 20 negative samples as positives, would result in a recall score of 0.8 and an accuracy score of 0.28. The advantage of the F1 score is that it avoids falling into these traps caused by data imbalance, as both metrics are taken into account. Like ROC-AUC, the F1 score ranges from 0 to 1. The interpretation of the score is typically as follows: A score of below 0.5 is considered not good, while a score between 0.5 and 0.8 is considered okay. A score between 0.8 and 0.9 is good, and anything above 0.9 is considered very good.[30] The weighted F1 score (F1_w) is used instead of the F1 score, as the F1 score treats each class equally, which is an issue in datasets such as this one, where one class has significantly less samples than the other classes. The weighted F1 score takes into account how many samples of each class are present in the dataset, and weighs the score of the classes with more samples higher than those with fewer samples. The difference between the two parameters is seen in their formulas, where w_i is the number of samples in class i divided by the total number of samples. [31]

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$F1_{\text{w}} = \frac{\sum_{i=1}^{N} w_i \cdot F1_i}{\sum_{i=1}^{N} w_i}$$

This is useful in imbalanced datasets as it gives an indication of how the model performs overall, rather than being dragged down by poor performance resulting from the class with fewer samples. [32]

3.5.2 Stratified cross-validation

The six machine learning algorithms were used to make models using the dataset and the input described in section 3.5.1 In order to reduce the variance in the results and thus give more accurate F1 and AUC-ROC scores, the method of using cross-validation was used. The Scikit-learn library has a wide variety of built-in cross validation functions. This thesis uses the *StratifiedKFold* function. It works similar to the more generic *K-fold cross-validation* function which splits the training data into *K* amount of *folds*. Then, the function trains and tests the model *K* times, using a different fold for testing and training on the remaining folds. The difference between this basic function and the *StratifiedKFold* function is that the latter ensures that the distributions of classes in each fold is identical to the distribution in the original dataset [33]. This is the reason for choosing this particular function, as the dataset used in this thesis is imbalanced. For this thesis, the amount of folds was set to 10. This value is supported in literature, and the amount of folds are typically set to either 5 or 10, as these have been empirically shown to result in outcomes that are neither high in bias nor suffer from high variance [34]. The models that are trained in this thesis therefore produce ROC-AUC and F1 scores that are based on the averages generated by stratified cross-validation, thus reducing how much these numbers suffer from imbalance, which in turn improves their value as parameters.

On the basis that ROC-AUC scores of at least 0.7 are considered acceptable, and only F1 scores above 0.5 were considered okay, only the models performing above these scores were used to investigate feature importances, with the rest being discarded. As only some algorithms have feature importances, models meeting the minimum score success criteria that did not have feature importances would have to be omitted. For each cross-validation iteration, the feature importances were stored in a list, from which the average feature importances were calculated. In order to present the most significant features, a method of determining which features are significant had to be chosen. This study considers a feature to be of significant impact to the model, if the importance is at least 20 % more than would be expected if all features were equally important. This means that in order for a feature to be considered significant, the importance score has to be at least 0.08. This number was calculated using the following method, where N = number of features.

$$\frac{100}{N} \cdot 1.2 = \frac{100}{15} \cdot 1.2 = 0.08$$

3.5.3 Confusion Matrix

This thesis also uses confusion matrices to evaluate the performance of the models. The reason for this is that it gives a clear overview of which areas a model performs good or bad in. A confusion matrix shows the number of true positives, false positives, true negatives and false negatives made by a model. This gives a clear overview of which areas a model may be underperforming in, which is crucial for identifying areas for future improvement. This is especially useful in this thesis, as an imbalanced dataset may lead to unsatisfactory results for the underrepresented class, which may impact the performance of other evaluation parameters, without these parameters being able to easily show why the performance of the model may be hindered [19].

3.5.4 Feature importances

Some algorithms have a built in method of showing how much each feature in the dataset impacted the predictions of models that use these algorithms. This is shown as a number between 0 and 1, with the total sum of the feature importances being 1. Feature importances are therefore often used to remove features that have a low impact on predictions, and may therefore be harmful to the overall performance of the model, as the feature may be contributing more noise than it contributes to making accurate predictions [35]. For the purpose of this thesis however, the feature importances will simply be used to investigate which sustainability metrics impact the prediction of retirement and graduation the most. As is argued in chapter 5, the data for the features used in this thesis is clearly labeled, cleaned and distinct from each other, and therefore it is unlikely that enough noise is generated from the low performing features to have a noticeable impact on the overall performance of our models.

Findings

This chapter contains the findings that resulted from the process outlined in chapter 3. Firstly, the confusion matrices as well as the accuracy of the algorithms will be presented, which is then followed by some brief reflections. Secondly, the algorithms that pass the success criteria of sufficiently high ROC-AUC and $\mathrm{F1}_w$ scores will also have the importance of their features presented and reflected upon. As only decision tree, random forest and gradient boosting algorithms contain feature importances, these are the only ones that will be considered for the feature importance section, provided their scores are high enough.

The code can be accessed in full in the GitHub repository found in appendix A.

4.1 Algorithm accuracy

The following section presents metrics of accuracy relating to the testing of the trained models. The accuracy of each model algorithm is examined in this regard.

4.1.1 Confusion matrices

The confusion matrices that resulted from the machine learning models can be found in table 4.1 below.

Algorithm	True Positive	False Positive	False Negative	True Negative
Decision Tree	155	32	19	18
Random Forest	178	9	23	14
Gradient Boosting	173	14	22	15
Support Vector Machine	187	0	37	0
Logistic Regression	184	3	33	4
k-Nearest Neighbors	173	14	33	4

Table 4.1: Confusion matrices for different algorithms

As can be seen in the table, all models can confidently identify the true positives.

The model that performs worst regarding the positives is the decision tree model, which mistakenly identified 32 negative samples as positives, while it correctly identified 155 positive samples. It is in identifying the negatives we find the largest issue. None of the models have more true negatives than false negatives, which means that they are not confident in predicting the retired samples. In this area, the models can be grouped into two categories. Decision tree, random forest and gradient boosting all have false negative and true negative scores that are relatively close to each other. This means that while the models are not exactly making random guesses, they are unable to effectively identify the negative class. Still, they are able to correctly identify around half of the retired samples. The second grouping of models contains the support vector machine, logistic regression and k-nearest neighbours. These models all show a significant inability to correctly identify negative samples. Worst is the support vector machine which did not identify a single negative sample. While both groupings of models struggle with identifying the negative samples correctly, the first group does see limited success. The reason for this inability to identify retired samples likely stems from the low amount of retired projects in the dataset. A similar dataset with more retired samples may see more success in this regard.

4.1.2 ROC-AUC & F1 scores

After training and testing the models, the average ROC-AUC and F1_w scores of each model has been calculated and presented in table 4.2 below. The table also contains the outcome of the algorithm, which refers to whether or not the algorithm passes the minimum scores of 0.70 ROC-AUC and 0.5 F1_w to be considered at least ”acceptable” and ”okay”

Algorithm	**Avg. ROC-AUC Score**	**Avg. F1 Score**	**Avg. F1_w Score**	**Outcome**
Decision tree	0.66404	0.63690	0.78306	Failed
Random forest	0.81919	0.69430	0.84258	Passed
Gradient boosting	0.79554	0.67845	0.82916	Passed
Support vector machines	0.38596	0.45498	0.76001	Failed
Logistic regression	0.84313	0.52884	0.78374	Passed
K-nearest neighbors	0.61824	0.50251	0.75424	Failed

Table 4.2: Algorithm performance summary

Table 4.2 contains three algorithms that pass the success criteria, and three that do not pass.

The decision tree algorithm had a score of 0.66404 (ROC-AUC) and 0.78306 (F1_w), which means it did not meet the success criteria, but it was close to doing so. It had lower scores than the random forest algorithm, which did pass with good scores of 0.81919 (ROC-AUC) and 0.84258 (F1_w). This is as expected, as a random forest algorithm uses multiple decision trees as well as containing multiple methods to reduce overfitting and other margins of error. This could be caused by the large amount of features, as this increases the complexity of the model, which may increase the problem caused by having a small dataset.

Gradient boosting passed with scores of 0.79554 (ROC-AUC) and 0.82916 (F1_w), which means it has an acceptable ROC-AUC score and a good F1_w score. These scores may be due to the nature of gradient boosting, as the model will iterate several times and improve in ways that specifically reduce the error rate of the previous iteration.

The support vector machine algorithm failed with scores of 0.38596 (ROC-AUC) and 0.76001 (F1$_w$). There can be different causes for this, but an immediate cause can stem from the nature of the algorithm. The support vector machine algorithm is most effective when there are more features than there are samples. As this thesis has 15 features and 225 samples, it does not put this algorithm in the optimal conditions for it to be precise [36] . However, the algorithm fails only on account of the ROC-AUC score. The F1$_w$ score is almost high enough for to be considered a ”good” score. The fact that the ROC-AUC score is below 0.5 means it is performing worse than randomly guessing would. However, as the models are trained on an imbalanced dataset, it would be expected that the F1$_w$ score performs better than the ROC-AUC score does. The low ROC-AUC score is therefore likely a result of this, as well as the issue of samples exceeding the amount of features.

Logistic regression passed with a score of 0.84313 (ROC-AUC) and 0.78374 (F1$_w$), which is not surprising as logistic regression is targeted towards classification tasks where the goal is to identify the probability of outcomes.

The K-nearest neighbours algorithm failed, but was still close to passing, as it had scores of 0.61824 (ROC-AUC) and 0.75424 (F1$_w$). The failure of this algorithm is likely due to overfitting, as this algorithm is particularly vulnerable to this issue. The performance of K-nearest neighbours is also negatively impacted by datasets with a high amount of features. While the model documentation does not explicitly dictate what constitutes a high amount of features, our findings indicate that 16 features might be problematic for this particular algorithm.

4.2 Feature importance

The investigation of feature importances was done on the basis of the algorithms that met the success criteria and can be seen to have passed in table 4.2. These are gradient boosting and random forest. Logistic regression is omitted here despite meeting the success criteria, as it does not contain feature importances. As outlined in the methods chapter, this part of the study also uses cross-validation. After each fold is trained and tested, the feature importances are appended to a list corresponding to the feature. Finally, the average importance of all features is calculated. These can be seen below in table 4.3

Feature	Sustainability indicator	Random Forest	Gradient Boosting
COM-1	Response time	0.07430	0.06910
COM-2	Frequency of communication	0.09298	0.09375
POP-1	Project popularity	0.06190	0.01906
STA-1	Age	0.04800	0.02900
STA-2	Attrition	0.05180	0.06040
STA-3	Forks	0.06300	0.03327
STA-4	Growth	0.07780	0.09275
STA-6	Life-cycle stage	0.01210	0.00851
STA-7	Retention	0.04580	0.04452
STA-8	Size	0.12900	0.27023
STA-9	Turnover	0.08918	0.04131
TEC-1	Contributors' dev. activity	0.02150	0.00195
TEC-2	Efficiency	0.05230	0.04327
TEC-3	Non-code contributions	0.06250	0.05418
TEC-4	Overall development activity	0.12467	0.13798

Table 4.3: Feature importance table (rounded to 5 significant decimals)

Table 4.1a and 4.1b below show the features that are considered significant for each algorithm, with the definition of significant being at least an importance score of 0.08, as outlined in chapter 3

Feature	Importance score
STA-8	0.12900
TEC-4	0.12467
COM-2	0.09298
STA-9	0.08918

(a) Significant features for random forest (rounded to 5 significant decimals)

Feature	Importance score
STA-8	0.27023
TEC-4	0.13798
COM-2	0.09375
STA-4	0.09275

(b) Significant features for gradient boosting (rounded to 5 significant decimals)

Figure 4.1: Significant features for random forest and gradient boosting

Features STA-8, TEC-4 and COM-2 are present in both tables, while the random forest table also contains STA-9 and the gradient boosting table contains STA-4. In both tables, STA-8 and TEC-4 have the highest scores, so these can be considered to be the most significant features. The models do not place them in the same order of importance however, as the gradient boosting table gives STA-8 more than twice the score than the random forest table does. A score of 0.27023 means that the gradient boosting algorithm considers STA-8 (size) To account for more than a fourth of the weight in the model's decisions to predict either graduation or retirement. This is significant considering there are a total of 16 features. Random forest gives a greater importance to COM-2 than gradient boosting does, however this may be due to the significance of STA-8, as it monopolizes a fourth of the available feature importance score, which drags the score of the other features down. By comparing the tables and only noting the unique features, this study has found the most significant features for predicting retirement or graduation. These are listed below in table 4.4

Feature	Sustainability indicator
STA-8	Size
TEC-4	Overall dev. activity
STA-4	Growth
COM-2	Frequency of communication
STA-9	Turnover

Table 4.4: Unique features with importance score above 0.8

4.3 Summary

Here we combine findings from the confusion matrices as well as the ROC-AUC and $\mathrm{F1}_w$ scores. The models that did not satisfy our accuracy criteria were discarded. The ROC and $\mathrm{F1}_w$ scores indicate that random forest, gradient boosting and logistic regression perform well. However, if this is compared with the confusion matrices, logistic regression should not be considered viable due to the almost complete inability to correctly identify a retired sample. Random forest and gradient boosting also struggle in this area, but to a much lesser extent, as they are able to correctly predict almost half the retired samples. This indicates these models have potential, and that they are reliable candidates for predictive models for retirement or graduation. If they were to be trained on a similar dataset with the addition of more retired project samples, it is likely that these models would improve significantly. In response to the research question of this study, we find that it is possible to predict graduation and retirement for ASFI projects, based on a dataset that incorporates sustainability metrics. This study finds that the most accurate machine learning models are based on the random forest and gradient boosting algorithms. Furthermore, random forest and gradient boosting models put a high degree of importance on STA-8 (Size), TEC-4 (Overall dev. activity), STA-4 (Growth), COM-2 (Frequency of communication) and STA-9 (Turnover).

Discussion

The purpose of this section is to discuss our findings, as well as the relevance and meaning of the observed results in regard to literature relating to FOSS sustainability. We will also discuss and suggest perspectives on FOSS sustainability prediction as a whole, as well as the implications of integrating machine learning prediction into sustainability prediction research and Apache project decision making.

5.1 Predictive capabilities

In this section, we will discuss the predictive capabilities of the models used to predict graduation and retirement of ASFI projects.

Firstly, analysis of the models shows that it is possible to predict graduation and retirement. This is shown by the random forest and gradient boosting models, as these had sufficiently high ROC-AUC scores as well as weighted F1 scores, and they showed promise in their ability to identify the retired and graduated samples, as shown in their confusion matrices. The results suggest that while the models demonstrate promising potential, caution is advised in interpreting their predictive accuracy due to the challenges posed by data imbalance in the training and testing sets. Still, the results suggest that if the models were to be trained on data consisting of more retired projects, the models would have a better performance. This argument rests on the high performance of the weighted F1 score, as this score gives insight into how the performance would be if the influence of the data imbalance were to be removed. However, it is also important to keep in mind, that the thresholds for acceptable scores for $\mathrm{F1}_w$ and ROC-AUC all come from convention and ”rules of thumb”, as explained in chapter 3. The validity of these thresholds can always be argued against on the basis that algorithms may have wildly different scores depending on dataset size. However, the scores of the models tested in this thesis generally rank as would be expected, which supports the chosen thresholds.

Secondly, it is also important to discuss the size of the dataset used in this thesis. Much effort has been put into trying to combat the effects of data imbalance, but this does very little to address the fact that the dataset itself is relatively small. Datasets are typically much larger than the dataset used in this thesis. For instance, Google Translate is trained on a dataset that uses trillions of samples [37]. While this is an extreme example, it illustrates that a dataset consisting of 224 samples such as this one, can be considered a small sized dataset. This is addressed in two ways. Firstly, while the dataset is indeed small, the samples are of good quality, as there is a lot of information in each sample. Also, the data has been cleaned and processed, which eliminates the risk of duplicate values, bad labels and omitted values and so forth. Furthermore, they are clearly labelled, with each label being distinct from the others. While this does not increase the size of the dataset, ensuring a high data quality improves the usefulness of model predictions [37]. Secondly, a general rule of thumb for the size of datasets is that the number of samples should be an order of magnitude larger than the amount of trainable features [37]. As this dataset has 16 trainable features, it must therefore have a minimum of 160 samples, which this dataset does. Therefore we argue that the size of the dataset does not significantly impact the outcome of this thesis.

One point of critique for the predictive capabilities of the models comes from the nature of retirement and graduation. The models are trained on data which labels a given project as retired or graduated from the Apache Incubator. However, as the final decision to retire or graduate a project is taken by the ASFI board, this means the decision is to a certain extent influenced by human factors. This impacts what the models actually predict. The models do not predict whether or not a project should be retired or graduated, they predict whether or not the board would retire or graduate them. They are effectively learning to replicate the decision making patterns of the board. While this does not make the models wrong, it is important to keep this issue in mind when using these models as tools for predicting graduation and retirement.

5.2 Graduation, retirement and sustainability

Graduation and retirement are, in this thesis, defined as binary outcomes for whether or not a a given ASFI project is able to come to fruition. As opposed to FOSS sustainability literature, this is a simplified and more reductionist approach. In that sense, the predictive capabilities discovered in our findings might imply that machine learning can contribute to gaining valuable insights within the sphere of ASFI projects, but it is not the same as sustainability outright. As established in chapter 2, the field of FOSS sustainability research is extensive and the methods used in this thesis relies to a great extent on a framework that assumes the measurement of sustainability to be a complex matter. With this in mind, it is important to establish that we do not make the assumption that a retired project was necessarily unsustainable. It is, however, relevant to explore the implication that an unsustainable project should, in theory, be retired. This line of thinking arguably establishes a connection between ASFI retirement and sustainability, but not one that is as reductionist as our approach to predicting retirement in the first place. The implication hereof is that predicting the outcome of ASFI projects is inherently different from predicting the sustainability of said projects.

There is, however, potential for valuable insight somewhere in between. Parallels can be drawn to the findings of Yehudi et al., in which the study of FOSS sustainability indicators proved to be difficult if only quantitative comparisons were taken into account [16]. This can be said to be the case in our training data, in which the models trained, even though they use different algorithms, all use cross-project quantitative comparisons to attempt classification and prediction. In order to gain a deeper understanding of the sustainability of FOSS projects after predicting their graduation or retirement, it may prove beneficial to follow the recommendations of Yehudi et al., and further analyze the model data with a combination of quantitative and qualitative data.

While this is beyond the scope of this thesis, we argue that the methodology used to train and test our models, along with our findings, provide valuable contributions towards deeper insight into sustainability prediction of FOSS projects in the ASFI. Specifically, the models in this thesis are trained using data processed through the sustainability framework of Linåker et al. Thus, even though our models produce predictions relating to graduation and retirement, the predictions of those outcomes are based on a sustainability framework. As established, some of the models were capable of achieving a predictive capability that can be considered reasonable, despite their difficulties. In practice, the high model accuracy in predicting graduation outcomes makes an interesting contribution in and of itself, considering that the implication hereof is that sustainability metrics can be used to predict ASFI graduation and retirement. While the argument of data imbalance being an important factor still stands, the model imperfections observed could also stem from sustainability metrics inherently being difficult to use to make quantitative comparisons, as concluded by Yehudi et al.

5.3 The implications of important features

Looking at the features with the highest importance scores, which can be found in table 4.1a and 4.1b, and the guidelines the ASFI use to determine if a project should graduate [6], we see several differences. Firstly, neither STA-8 (size) STA-9 (turnover) or STA-4 (growth) are apparent in the ASFI guide to successful graduation, aside from the fact that a project should not rely on a single contributor[6]. Secondly the features TEC-4(overall development activity) and COM-2 (Frequency of communication) do share some resemblance to the guide. This can be seen by the prioritization of a community's ability to be able to disagree and discuss technical matters without destroying relations between contributors [6]. While this is not the same as COM-2, both relate to the communication in a community, but the ASFI guide takes a different approach, focusing on how the community should communicate and not how much. The same comparison can be made with TEC-4, due to the fact that the ASFI guide does not have requirements regarding the overall development activity of the project, but has a significant focus on the meritocratic structure of a project[6]. This means that a member of the community should be given responsibility based on the value they create for the project. Looking at how the ASFI guidelines for graduation compares to the literature of FOSS sustainability metrics, several similarities can be seen. For instance, it is a priority for the ASFI that project communities are open and diverse[6], which also can be seen in Linåker et al's. framework. While many of the priorities of the ASFI concerns the community of a project, there are a few guidelines regarding the technical requirements of projects in the Apache Project Maturity Model [38]. This model provides a framework for evaluating the maturity of a project and its codebase, there is still no great correlation between this suggested framework and the features used in this thesis [38]. These are mainly about broader topics such as availability of libraries, security and requirements for releases. These can also be found in Linåker et al's. framework[4].

This clarifies that the data of the indicators used in this thesis does not represent the current criteria the ASFI officially employ to determine if a project should graduate or not. While this is the case, there could be several other criteria, used unconsciously, that are related to the indicators used in this thesis, and affect the decision to graduate a project. If the board were to graduate a particularly large project, they may not explicitly state that ”size” was taken into account, despite the fact that it could make a significant difference in their subjective evaluation. This is interesting as the models are trained on data from the ASFI, whose official criteria for graduation relies on other measures, but are still able to predict the graduation or retirement of a project based on more quantitative data from the projects. This raises two considerations. Firstly, if the ASFI should have a larger focus on some of the quantitative measures of their incubator projects. Secondly, if future studies should include indicators or methods capable of representing the ASFI's current criteria for graduation.

As all the data used in this thesis is from ASFI projects that have either retired or graduated, it would make sense to include data on the criteria the ASFI use to determine if a project should graduate or retire. While the random forest and gradient boosting models are able to predict if a project is graduated comfortably and to some degree if a project is retired, there is room for improvement based on the confusion matrices. Models that are trained on the indicators that the ASFI prioritizes for project graduation could lead to further findings.

It is unclear if the ASFI should incorporate more of the indicators used in this thesis in their decision to graduate a project. There are several reasons for this. Firstly, we can only look at the guidelines the ASFI has published, and it is unknown what discussions take place when deciding if a project should graduate. Therefore, some of the indicators used in this thesis could influence decision making when graduating a project. Secondly, it is important to note that the importance score of each feature used to develop the models does not necessarily reflect the actual impact on sustainability, but rather the impact a feature has on the predictions. Therefore it is difficult to recommend if the ASFI should include the indicators used in this thesis.

5.4 Future Work

5.4.1 Applications for non-Apache projects

The models trained in this thesis have only been trained on data from ASFI projects, and therefore the findings are not necessarily applicable to FOSS projects outside of the ASFI. It could, however, prove valuable to further study how our findings would apply to FOSS projects outside of the ASFI. If further research were to take place, it would require focusing on the broader context of FOSS sustainability. Furthermore, a specific definition of when a project is sustainable and unsustainable would have to be defined, if the same approach used in this thesis were to be replicated. If it is proven possible to predict whether a given FOSS project was sustainable or unsustainable, this capability could be valuable for FOSS projects in general.

5.4.2 Applications for Apache decision making

An important distinction that can also be inferred from the models regards the very nature of their training. While it is true that the focus of this thesis is on the prediction of ASFI projects, the data used to train the models are still based on board decisions. Thus, it can be argued that the internal process of discussion and voting exists as a black box where subjective factors also play a part. In practice, this means that our models do not predict project outcomes in the literal sense, but rather board decisions on retirement. With this in mind, it may be beneficial to explore these qualitative avenues that are unavailable through strictly quantitative comparison. In theory, a direct research partnership with the Apache Foundation may allow for more project-oriented ML-prediction research, rather than looking in at board decisions from the outside. More specifically, if the exact qualitative requirements for board decisions were implemented into model training, it may be possible to enhance the decision making of the board in order to mitigate the impact of the subjective human factor. In this regard, it may also be possible to explore whether or not retirement and graduation can be done objectively at all, as well as the implications of this perceived lack of objective evaluation in the question of retirement.

5.4.3 Addressing data imbalance

One of the largest issues faced by this thesis is the data imbalance of the dataset. The impact of this is further addressed in chapter 6.

There are several ways to avoid the issues faced by data imbalance. The most obvious solution would be to simply avoid using ASFI retirement and graduation, and instead focus on investigating sustainability metrics. This thesis has already gathered all the retired projects that are listed in the Apache Incubator, and this was not enough to train models that are confident in identifying retired projects. A future study that investigates projects that have been retired from the Apache Incubator will have to either wait for a sufficient amount of projects to retire, or reduce the amount of graduated samples (undersampling) and find a more qualitative way to train and evaluate machine learning models [39].

It is also possible to manipulate the data in such a way that the dataset is artificially augmented with additional retired samples (oversampling) [40]. There are different ways to do this. Some techniques simply clone a retired sample resulting in two identical retired samples. While this is not problematic on a small scale, this results in a less useful and overfitted model if done on a larger scale, as this would likely result in a model being very good at identifying these specific samples, but would likely struggle when faced with new data. A more sophisticated approach to oversample a dataset, is to create ”synthetic” retired samples which are not completely identical to the original samples. Rather, these synthetic samples are generated from the relationships between the existing samples, with a degree of randomness introduced. This method is called Synthetic Minority Oversamplig Technique (SMOTE), and works by taking each minority class sample and introducing synthetic examples which are based on a selected instance of a minority class, and a k

nearest neighbour. A random point along the line segment between these two datapoints is selected, and this point is then added to the dataset as a new instance of the minority class [41] .

A future study might benefit from a combination of these solutions. Waiting for more projects to retire and then oversampling using SMOTE could be a good balance between not having too wait too long for new retired projects and not being forced to use too much oversampling.

5.4.4 Using sequential data

While this thesis investigates a broad range of sustainability metrics as well as ASFI retirement and graduation, it does so using data that is only collected for a single point in time. This limits the models in what type of predictions they are able to make, as the current models are only able to predict whether or not a project as it currently stands would be retired or graduated from the Apache Incubator. If a future study were to dedicate a longer period of time such as a year to gathering data monthly for a predetermined list of Incubator podlings, it could in many ways be an improvement over the capabilities of the models in this thesis. Firstly, this could improve the overall performance of the models, as the dataset would be larger, and this would result in more data for the models to train on. Secondly, it would teach the models to identify trends in Apache Incubator podlings. For instance, it could be that a significant drop in a metric such as COM-2 (frequency of communication) is usually accompanied by a drop in TEC-4 (overall dev. activity), which is a metric our findings have shown to have significant impact on whether or not a project is graduated or retired from the Apache Incubator. Of course, expanding the dataset to include sequential data would warrant a change in the choice of machine learning algorithms, as the algorithms used in this thesis are meant for classification tasks, and are therefore not necessarily useful for handling datasets that are sequential rather than static. A more appropriate machine learning algorithm for a future study attempting this is recurrent neural networks. These are specifically designed to handle sequential data, and are ideal for tasks where the relationship between different data-points in time is of particular importance [42]. Gathering sequential data for this dataset would dramatically increase the size of the new dataset, which would increase computation time as well. Depending on the computation power available to this hypothetical study, it might be beneficial to exclude some of the sustainability indicators that consistently had low feature importance scores, or simply focus on only the indicators which had high feature importance scores. This could also further reduce the amount of noise present in the dataset.

5.4.5 Using indicators from the ASFI

Our thesis includes indicators that do not reflect the current ASFI decision making process regarding the graduation of projects. While we had some success predicting project graduation and retirement with these indicators, replicating our research with indicators that reflect the ASFI's graduation requirements could prove interesting. Given that the current graduation criteria largely align with Linåker et al.'s sustainability indicator framework, the implications regarding graduation and retirement remain applicable to sustainability. It can be argued that a model trained specifically on data representing the current graduation criteria should be highly effective. This is because there should be a clear distinction between graduated and retired projects. If this is not the case, it could be because the ASFI is not consistent in their graduation and retirement decisions. Furthermore, it could also show that there might be other factors used to determine if a project should retire or graduate. A potential challenge for this study would be to acquire the necessary data. The criteria used to determine if a project should graduate is mostly qualitative, and would therefore require a different method for data collection. Furthermore the data would potentially have to be converted from qualitative data to values that the models can be trained on.

5.5 Summary

To summarize, this discussion considers the critical aspects of the dataset and the models used in this thesis. It highlights the challenges faced by a small and imbalanced dataset, while weighing the merits of data quality and data quantity. The discussion critiques the models predictive capabilities, and points out that they predict the ASFI board's subjective decision making, rather than predicting if a project objectively merits graduation or retirement. The chapter differentiates between predicting project graduation or retirement and evaluating project sustainability, suggesting a combination of quantitative and qualitative data for deeper insights. Further analysis has revealed discrepancies between the most important features, and the ASFI's graduation guidelines, and questions whether the ASFI should incorporate more quantitative measure or if the thesis should have included more indicators that align with the ASFI guidelines. Finally, this chapter suggests several directions that might be taken by future studies, such as incorporating non-Apache projects, exploring qualitative decision making aspects, addressing data imbalance and using sequential data to predict trends in FOSS projects.

Chapter 6

Threats to Validity

The following chapter is composed of two sections, internal validity and external validity. These exist to discuss the validity of our findings and to establish the credibility and trustworthiness of this study. This chapter is also a discussion of research design issues, biases, limitations and weaknesses. Understanding and addressing these concerns is important to enhance the reliability of our findings.

6.1 Internal validity

6.1.1 The human factor

The data used to train the models has been mined from projects that were either retired or graduated based on the ASFI board's decision [7]. The voting process to graduate or retire a project differs if the project intends to become a top level individual project or a sub-project to an already existing top level project. Both have in common that the community itself needs to first agree in a vote to graduate a project, to start the process. If the majority agrees, the final decision is made by the board of the Apache Software Foundation if the project aims to become a top level project, or the Incubator Project Management Committee (IPMC) if the project aims to be a sub-project. [6]. Because the final decision is made by the board and the Apache Incubator, it is not possible to know precisely what they base their decisions on. While this creates some uncertainty about the similarities between a project graduating and it being sustainable, the Apache Incubator states that it highly values open and diverse communities and argues that Apache projects should be self-sustainable, with the ability to recruit new members, take responsible collective action and be able to disagree on technical matters without being destructive[6]. This shows that

The Apache Software Foundation and Incubator to some degree includes sustainability in their decision to graduate projects, albeit with an uncertain definition.

The board or the IPMC are also able to force a project to retire, creating the same threat to validity of not knowing why they choose to retire a project [7]. This threat is to some extent alleviated by the fact that it is mostly the project itself that decides to retire[7].

6.1.2 Singular data points

A concern about the data used to train the models is that our data is taken from a single point in time for each project. Furthermore, the data used to train the models is mined from projects that have already either retired or graduated. Since our models have not been validated with data from projects currently active in the incubator, the outcome remains uncertain. While the effectiveness of the models on data from currently active projects are uncertain, there are several ways to improve this. Firstly data from current projects could be included, allowing re-training of the models on this data. Secondly, we could test the models on data from podlings that are not yet retired or graduated, and then observe if the predictions match the eventual outcomes. These additions would help prove that the models are capable of predicting ongoing projects.

6.1.3 Unknown indicator factors

Another potential threat to the validity of this thesis comes from the indicators themselves. Despite the chosen indicators being proven to have the capacity to provide valuable insight, it should also be taken into account what they individually imply. In particular, the indicator POP-1 (popularity) is of note. This indicator is defined by how many stars a given project has on its respective GitHub page, representing amount of times a user has marked the project as a favourite. While this gives a picture of how many people have shown interest in the project, it can be argued that it might not be expected of every user to remove stars from projects they have lost interest in. This has the potential effect of giving a false picture of the actual popularity of a project, since there is no guarantee of users removing their stars from a project they no longer have an interest in.

Furthermore it is possible that the computation methods of the sustainability indicators have some inherent biases. An example of this could be seen looking at the computation of the indicator Growth (STA-4). This is an indicator with a complex definition, but was simplified during the computation of Alami et al. due to the constraints of MSR. While this could impact the validity of the thesis, it is only one of 15 indicators, and the feature importance search showed that this indicator was not critical when making a prediction. Furthermore, we acknowledge that there may be unknown, complex and qualitative factors effecting one or more sustainability indicators, but the investigation of online user behaviors is beyond the scope of this thesis.

6.1.4 Unused indicators of sustainability

The sustainability of FOSS projects is a complex metric to predict. This thesis uses 15 different indicators, which raises the question of whether 15 indicators is enough to fully understand the sustainability of a project. While using all 107 indicators found in the framework by Linåker et al. could potentially give a broader picture of the current state of the sustainability of a given project, it would be excessive and unfeasible for the scale of the podling projects in the Apache Incubator. These projects are all in their early development, and are adapting to the format and processes Apache projects work by. Furthermore, some of the models used in this thesis show promise in their ability to accurately make predictions, suggesting that the indicators used are sufficient to make accurate predictions on Apache incubator projects.

6.1.5 Using stratified cross validation

Stratified cross validation was used when training the models to decrease the potential problem of class imbalance. However, using this method has downsides. When using stratified cross validation, an equal number of each sample is placed on each partition at random in order to maintain an even distribution. This creates another problem as it is done at random, and therefore could create a data shift. A data shift occurs when the data is distributed in ways that do not accurately represent the actual distribution of the data [43]. In our case, this could occur if a fold is created that exclusively has projects that are highly influenced by a single indicator. This would not be representative of the distribution of the entire dataset, which leads to a flawed fold. While this could have a negative effect on the results of the models, we argue that the negative effects are sufficiently negated. This is because our data shares the same context of the ASFI, and the same intention of reaching graduation. This means that the dataset is unlikely to have differences significant enough to cause a data shift.

6.1.6 Data imbalance

A significant challenge to the validity of this thesis comes from the imbalanced nature of the dataset used. While the data imbalance is acknowledged throughout this thesis, it is equally important to elaborate on, to what degree the dataset is imbalanced. Table 6.1 below illustrates what constitutes a mild, moderate or extreme data imbalance.

Degree of Imbalance	**Proportion of Minority Class**
Mild	20-40% of the data set
Moderate	1-20% of the data set
Extreme	<1% of the data set

Table 6.1: Degree of Imbalance and Proportion of Minority Class [44]

As the dataset used in this thesis consists of 224 projects with 37 of these being retired projects, this results in the minority class being represented in only 16.52% of the total amount of samples. This classifies the models as suffering from a moderate degree of data imbalance. This imbalance is mitigated to a certain extent due to the methods used in this thesis. Firstly, using the weighted F1 parameter as a measurement of model performance reduces the impact of data imbalance on the performance scores. Secondly, the use of stratified cross validation ensures that no model will be trained or tested on subsets containing no samples from the minority retired class. While an imbalanced dataset can be the cause of a multitude of issues, the core problem is that the models will spend most of their training on the majority class and thus neglect the minority class. This seems to have been the case for the models trained in this thesis, as they are unable to confidently identify the minority class. It can be argued that this is inherently tied to the nature of the data from the ASFI, as projects currently graduate more often than retire, thus creating the circumstances for data imbalance. [45]

6.2 External validity

6.2.1 Transferability

Given the focus of this thesis being ASFI projects, it is important to consider the transferability of our findings to other contexts. While the findings may hold relevance within the broader study of sustainability prediction in open source, it is also important to note that our methods are focused specifically on ASFI projects, and the findings from our models do not necessarily provide insights that can be generalized across every open source project. Gaining such insights outside the domain of incubator projects could require more extensive ML training, or a stronger emphasis on the more qualitative indicators omitted from the methods used in this thesis, such as the impact of culture or finance on FOSS projects. However, certain aspects of this thesis such as the impact of sustainability indicators on incubator project outcomes, or the lessons learned through data processing and model selection, may have broader applicability. It should be noted, however, that the training and testing data used in this thesis are specifically tied to the ASFI, with the models being trained specifically to predict a binary outcome of retirement and graduation. Outside the context of the ASFI, there may be various other outcomes for FOSS projects such as abandonment, in-operation, failure, etc.

Chapter 7

Conclusion

The purpose of this thesis was to investigate how machine learning models can be used to predict FOSS graduation and retirement. Based on our methodology, we sought to answer the following research question:

How can machine learning models that incorporate FOSS sustainability metrics be used to predict the likelihood of graduation and retirement for ASF incubator projects?

Based on ML-model training using established indicators of FOSS sustainability, we were able to demonstrate the predictive potential within the sphere of ASFI projects when utilizing metrics of sustainability proposed in the literature. The findings show that it is possible to predict ASFI project graduation and retirement. Through the analysis of confusion matrices, ROC-AUC and F1 scores, we find that random forest and gradient boosting algorithms result in models that perform best overall. While they do not achieve flawless performance, they show considerable promise in their ability to accurately predict retired and graduated projects. Furthermore, the findings show that the sustainability metrics that have the largest impact on the prediction of project graduation and retirement are STA-8 (Size), TEC-4 (Overall dev. activity), STA-4 (Growth), COM-2 (Frequency of communication) and STA-9 (Turnover).

Our findings highlight the challenges faced by performing model training using a small and imbalanced dataset, which proved difficult to avoid due to our focus on ASFI projects, of which most are graduated and not retired. We find that a close relationship between graduation, retirement and sustainability exists, but while a project that is not sustainable may become a candidate for project retirement, a retired project does not necessarily lack sustainability. A combined approach of using qualitative and quantitative data in model training may be required in order to gain a deeper understanding of this relation, while mitigating the implications of subjective evaluation.

Bibliography

[1] Solutionshub, *The state of open source software: Current trends, future outlook, benefits, and challenges*, Accessed - 09-05-2024. [Online]. Available: https://solutionshub.epam.com/blog/post/the-state-of-open-source.

[2] Kevin Crowston, Kangning Wei, James Howison, Andrea Wiggins, *Free/libre open-source software development - what we know and what we do not know*, Accessed - 25/04/2024, 2006.

[3] Karim R. Lakhani, Eric von Hippel, "How open source software works: "free" user-to-user assistance," *journal Research Policy*, 2005, Accessed - 25/04/2024.

[4] J. Linåker, E. Papatheocharous, and T. Olsson, "How to characterize the health of an open source software project? a snowball literature review of an emerging practice," in *Proceedings of the 18th International Symposium on Open Collaboration*, ser. OpenSym '22, , Madrid, Spain, Association for Computing Machinery, 2022, ISBN: 9781450398459. DOI: 10.1145/3555051.3555067. [Online]. Available: https://doi.org/10.1145/3555051.3555067.

[5] Apache Incubator, *The apache incubator*, Accessed - 09-05-2024. [Online]. Available: https://incubator.apache.org/.

[6] Apache Incubator, *Guide :: Guide to successful graduation*, Accessed - 10-05-2024. [Online]. Available: https://incubator.apache.org/guides/graduation.html#what_is_graduation.

[7] Apache Incubator, *Guide :: Guide to retirement*, Accessed - 09-05-2024. [Online]. Available: https://incubator.apache.org/guides/retirement.html.

[8] Apache Attic, *The apache attic*, Accessed - 09-05-2024. [Online]. Available: https://attic.apache.org/.

[9] J. Han, S. Deng, X. Xia, D. Wang, and J. Yin, "Characterization and prediction of popular projects on github," in *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 1, 2019, pp. 21–26. DOI: 10.1109/COMPSAC.2019.00013.

[10] K. Crowston and J. Howison, "Assessing the health of open source communities," *IEEE Computer*, vol. 39, no. 5, pp. 89–91, May 2006. DOI: 10.1109/MC.2006.152.

[11] D. Wahyudin, K. Mustofa, A. Schatten, S. Biffl, and A. M. Tjoa, "Monitoring the "health" status of open source web-engineering projects," *Int. J. Web Inf. Syst.*, vol. 3, pp. 116–139, 2007. [Online]. Available: https://api.semanticscholar.org/CorpusID:15898385.

[12] Alami, A., Pardo, R., Linåker, J., "Free open source communities sustainability: Does it make a difference in software quality?" *Zenodo*, 2023.

[13] G. Robles, A. Capiluppi, J. Gonzalez-Barahona, B. Lundell, and J. Gamalielsson, *Development effort estimation in free/open source software from activity in version control systems*, Mar. 2022.

[14] T. Xia, W. Fu, R. Shu, and T. Menzies, *Predicting project health for open source projects (using the decart hyperparameter optimizer)*, Jun. 2020.

[15] T. Xia, W. Fu, R. Shu, R. Agrawal, and T. Menzies, "Predicting health indicators for open source projects (using hyperparameter optimization)," *Empirical Software Engineering*, vol. 27, Jun. 2022. DOI: 10.1007/s10664-022-10171-0.

[16] Y. Yehudi, C. Goble, and C. Jay, *Individual context-free online community health indicators fail to identify open source software sustainability*, 2024. arXiv: 2309.12120 [cs.SE].

[17] u. Stănciulescu, L. Yin, and V. Filkov, "Code, quality, and process metrics in graduated and retired asfi projects," in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2022, , Singapore, Singapore, Association for Computing

Machinery, 2022, pp. 495–506, ISBN: 9781450394130. DOI: 10.1145/3540250.3549132. [Online]. Available: https://doi.org/10.1145/3540250.3549132.

[18] J. L. Adam Alami Raúl Pardo, *Free open source communities sustainability: Does it make a difference in software quality?* Accessed - 09/06/2024, 2024.

[19] Aurélien Géron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 2017.

[20] IBM, *What is overfitting?* Accessed - 09-06-2024. [Online]. Available: https://www.ibm.com/topics/overfitting.

[21] IBM, *What is a decision tree?* Accessed - 01/05/2024. [Online]. Available: https://www.ibm.com/topics/decision-trees.

[22] M. H. Ikram Chaabane Radhouane Guermazi, *Enhancing techniques for learning decision trees from imbalanced data*, Accessed - 29/05/2024, 2019.

[23] IBM, *What is random forest?* Accessed - 01/05/2024. [Online]. Available: https://www.ibm.com/topics/random-forest.

[24] Geeks for Geeks, *Understanding logistic regression*, Accessed - 01/05/2024. [Online]. Available: https://www.geeksforgeeks.org/understanding-logistic-regression/.

[25] Geeks for Geeks, *Gradient boosting in ml*, Accessed - 01/05/2024. [Online]. Available: https://www.geeksforgeeks.org/ml-gradient-boosting.

[26] Geeks for Geeks, *Support vector machine (svm) algorithm*, Accessed - 01/05/2024. [Online]. Available: https://www.geeksforgeeks.org/support-vector-machine-algorithm.

[27] Geeks for Geeks, *K-nearest neighbor(knn) algorithm*, Accessed - 01/05/2024. [Online]. Available: https://www.geeksforgeeks.org/k-nearest-neighbours/.

[28] Scikit-learn, *Train_test_split*, Accessed - 24/05/2024. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html#sklearn.model_selection.train_test_split.

[29] EvidentlyAI, *How to explain the roc curve and roc auc score?* Accessed - 01/05/2024. [Online]. Available: https://www.evidentlyai.com/classification-metrics/explain-roc-curve.

[30] Nikolaj Buhl, *F1 score in machine learning*, Accessed - 27/05/2024. [Online]. Available: https://encord.com/blog/f1-score-in-machine-learning/.

[31] Geeks For Geeks, *F1 score in machine learning*, Accessed - 01-06-2024. [Online]. Available: https://www.geeksforgeeks.org/f1-score-in-machine-learning/.

[32] Scikit-learn, *F1_score*, Accessed - 30/05/2024. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html.

[33] Scikit-learn, *Stratifiedkfold*, Accessed - 27/05/2024. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html.

[34] James et al., *An Introduction to Statistical Learning: with Applications in R*. Springer-verlag New York Inc, 2013.

[35] Codecademy Team, *Feature importance*, Accessed - 24/05/2024. [Online]. Available: https://www.codecademy.com/article/fe-feature-importance-final.

[36] Geeks For Geeks, *Support vector machine in machine learning*, Accessed - 29/05/2024. [Online]. Available: https://www.geeksforgeeks.org/support-vector-machine-in-machine-learning/.

[37] Google for Developers, *The size and quality of a data set*, Accessed - 24/05/2024. [Online]. Available: https://developers.google.com/machine-learning/data-prep/construct/collect/data-size-quality.

[38] Apache Software Foundation, *Apache project maturity model*, Accessed - 09-06-2024. [Online]. Available: https://community.apache.org/apache-way/apache-project-maturity-model.html#communit.

[39] The imbalanced-learn developers, *Under-sampling*, Accessed - 24/05/2024. [Online]. Available: https://imbalanced-learn.org/stable/under_sampling.html.

[40] The imbalanced-learn developers, *Over-sampling*, Accessed - 24/05/2024. [Online]. Available: `https://imbalanced-learn.org/stable/over_sampling.html`.

[41] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, W. Philip Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, 2002.

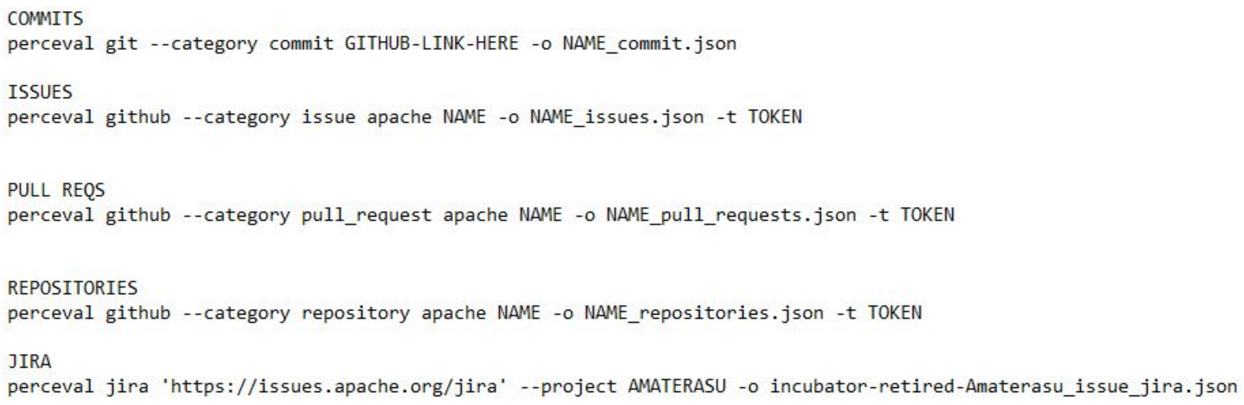
[42] IBM, *What are recurrent neural networks?* Accessed - 01/06/2024. [Online]. Available: `https://www.ibm.com/topics/recurrent-neural-networks`.

[43] F. H. Victoria López Alberto Fernández, *On the importance of the validation technique for classification with imbalanced datasets: Addressing covariate shift when data is skewed*, Accessed - 29/05/2024, 2012.

[44] Google for Developers, *Imbalanced data*, Accessed - 24/05/2024. [Online]. Available: `https://developers.google.com/machine-learning/data-prep/construct/sampling-splitting/imbalanced-data`.

[45] Apache Incubator, *All incubator projects by status*, Accessed - 12-06-2024. [Online]. Available: `https://incubator.apache.org/projects`.

Appendix



```
COMMITS
perceval git --category commit GITHUB-LINK-HERE -o NAME_commit.json

ISSUES
perceval github --category issue apache NAME -o NAME_issues.json -t TOKEN


PULL REQS
perceval github --category pull_request apache NAME -o NAME_pull_requests.json -t TOKEN


REPOSITORIES
perceval github --category repository apache NAME -o NAME_repositories.json -t TOKEN

JIRA
perceval jira 'https://issues.apache.org/jira' --project AMATERASU -o incubator-retired-Amaterasu_issue_jira.json
```

Figure A.1: Perceval syntax

GitHub repository: https://github.com/m-kudahl/fosspred