Machine Learning & Startups: Predicting The Next Unicorn?

Student details

Name: S.J.J. Krijger
Student number: U479274

THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE IN DATA SCIENCE & SOCIETY
DEPARTMENT OF COGNITIVE SCIENCE & ARTIFICIAL INTELLIGENCE
SCHOOL OF HUMANITIES AND DIGITAL SCIENCES
TILBURG UNIVERSITY

Thesis committee

Supervisor: dr. A.T. Hendrickson
Second reader: dr. W.A. Powell

Tilburg University
School of Humanities & Digital Sciences
Department of Cognitive Science & Artificial Intelligence
Tilburg, The Netherlands
25-06-21

Preface

You are now going to read the thesis "*Machine Learning & Startups: Predicting The Next Unicorn?"* I thoroughly enjoyed the process of writing this thesis and I hope this shows itself in the writings you are now going to read.

I would like to thank the following people that have helped me during this process of writing my thesis. First of all, Dr. A.T. Hendrickson for his continued guidance and support throughout the period, as well as my fellow thesis colleagues who were always there if I had a question. I would also like to thank my mother, father, brother and girlfriend for supporting me during this time. And lastly, I want to thank authors Tim Ferris & M.J. DeMarco for sparking my interest in entrepreneurship, which has eventually led me to writing this thesis.

Enjoy reading.

Stephan Krijger

Abstract

This thesis has focused on predicting startup status for different categories of startups based on publicly available data. A startup can have four types of status: operating, acquired, initial public offering (IPO), or closed. The research question was "*To what extent can startup status be predicted for different categories of startups, based on publicly available data?*" This study featured two clustering algorithms to decide the optimal number of categories of startups, seven models for predicting startup status, and six methods for overcoming class imbalance as the target class features a heavy class imbalance. The data comes from the startup database website Crunchbase.com and houses information about startups from 2000-2015.

Features from current literature, where possible, were put into the model. Four startup categories were studied: Information Technology, Health Care, Consumer Discretionary & Finance. The best F1-scores per startup category were respectively: 40.7%, 39.2%, 45.4% and 38.5%. The best machine learning models per startup category were all tree-based classifiers, but they differed per category. They all outperformed the baseline that did not make use of oversampling techniques. The best methods for oversampling the minority class were SMOTE-ENN for one startup category and ProWSyn for three startup categories.

To assess how well the models are at future prediction, the best models and methods for oversampling were then tested on a newer dataset of 2016-2019. Results for Information Technology, Health Care, Consumer Discretionary & Finance were: 35.6%, 40.8%, 31.5% and 33.3% on the F1-score respectively.

This thesis highlights the differences between startup categories and the performance of tree-based classifiers on this problem. It also shows how difficult it still is for Machine Learning models to predict on multi-class imbalanced datasets. Although results were better than the baseline, more work should be done in order to give venture capitalists or startup founders an accurate prediction of their future startup status.

Contents

1 Introduction

1.1 Context

This thesis focuses on the prediction task of startup status for specific startup categories. Startup status can be divided into four classes: operating, Initial Public Offering (IPO), acquired and closed. Operating means that the startup is still operating, IPO that the startup has done an IPO at the stock market, acquired that the startup is acquired by another company and lastly, closed indicates that the startup has failed.

1.1.1 Relevance of prediction for startups

Startups might be a small part of the economy, their effect on it is large. In the Netherlands startups already account for 109,000 jobs, with an annual increase of 8% (S. Sharma, 2020). On a global level, the startup economy creates nearly $3 trillion in value, similar to that of an entire G7 economy. In 2019 there was close to $300 billion invested in startups by venture capital (Startup Genome, 2020).

Looking at the failure rates of startups is frightening. For every ten startups created, around nine will fail (Investopedia, 2020a). Founding a successful startup or making a profitable investment is therefore not an easy thing to do. The best strategy a venture capitalist has is to bet that one of their investments turns out to be a unicorn, a term that is used in the startup landscape to denote a startup that has reached over $1 billion in valuation (Investopedia, 2021a). Finding the right startup is the hardest task for venture capitalists (Republic, 2020). Therefore, it is of vital importance to create machine learning models that can predict startup success or failure. Specifically, machine learning models should be made to predict the specific status of a startup, as what accounts for startup success is still not clear.

Knowing the future status of a startup is essential in supporting the startup climate. It can give startup founders a model to see where they can improve their company, and it can give venture capitalists a framework for quantitatively seeing what status a startup will have in the future. This can aid them in their decision-making process and will make sure they make more valuable investments.

1.1.2 Scientific Relevance

Most research in this domain has been done towards startup success prediction. Startup success is characterized differently by various scholars, as can be seen in the Related Work section of this study. Ross et al. (2020) only focused on three classes: closed, IPO, or acquired. Ünal (2019) focused on two classes: failure, which contains only the closed class, and success, containing the operating, IPO, and acquired class. This study puts its focus on the prediction of the four individual classes. Ignoring operating startups is questionable (Ross et al., 2020), as operating startups can still be successful, and this class is the biggest class of all startups in the dataset. However, putting operating, acquired and startups that have done an IPO all into one category (Ünal, 2019) is also not the best method, as not every startup that is operating will be successful. By focusing on all four classes, a venture capitalist or startup owner can predict in what class their startup will belong to in the future. Arroyo et al. (2019) have done a similar task at predicting these individual classes. The goal of this study is to improve the performance of their model as this was still low when looking at the F1-score.

Furthermore, no current literature has studied the performance of models on different categories of startups. This will be beneficial for startups and venture capitalists, as there could be differences between categories, making the models proposed by earlier research less compatible with their specific category. As this data involves a heavy class imbalance, several methods for overcoming class imbalance will be used to test which is the best one for this specific problem. Previous research has used only one class imbalance method for their study or none. A wider range of overcoming class imbalance methods has not been tested yet.

1.2 Research Questions

This thesis is focused on answering the research question: "*To what extent can startup status be predicted for different categories of startups, based on publicly available data?*" Which is answered by analyzing the following five sub-questions.

SQ1: What are the different categories of startups?

Before predicting startup status for different categories of startups, the startup categories have to be defined. This question will be answered by training two clustering algorithms on the category list variable of the dataset.

SQ2: Per category of startup, how does the model of Arroyo et al. (2019) perform on this dataset with a longer timeframe and different features, when evaluated by the F1-score?

The study of Arroyo et al. (2019) is most similar to the study proposed here. That is why as a baseline the best performing model of their study will be fitted to this dataset to see how it performs per startup category.

SQ3: Per category of startup, which machine learning model and method for overcoming class imbalance are best suited when evaluated by the F1-score?

In the literature, several models have performed well. No wide-ranging methods for overcoming class imbalance have been tested. That is why seven different models, and six different methods will be tested per startup category. In this way, it can be seen which model and oversampling method is best for which startup category.

SQ4: What are the differences in types of errors between startup categories?

In order to find out the differences between startup categories, the types of errors will be analyzed by comparing the confusion matrices of the startup categories.

SQ5: How do the best models and imbalance methods per startup category perform on more recent data of 2016-2019?

Lastly, the best performing models and methods for overcoming class imbalance from sub-question 3 will be used to predict on a newer dataset of 2016-2019. This is done in order to see how well these models predict on future data.

1.3 Findings

The findings show that tree-based classifiers work best on this problem (Random Forest, Adaptive Boosting (AdaBoost), Category Boosting (CatBoost) & eXtreme Gradient Boosting (XGBoost)) with the combination of either Synthetic Minority Oversampling Technique – Edited Neirest Neighbors (SMOTE-ENN) or Proximity Weighted Synthetic Oversampling Technique (ProWSyn) as a class imbalance method. The four startup categories studied are Information Technology, Health Care, Consumer Discretionary and Finance. Their respective best average F1-score (being the average F1-score across all startup status classes) are: 39.2%, 45.4%, 40.7% and 38.5%. These results translate to an additional test set of 2016-2019 with a slightly lower average F1-score: 35.6%, 40.8%, 31.5% and 33.3%.

2 Related Work

In the last few years, the body of literature on machine learning models to the startup landscape has grown significantly. This section is divided into five sections. First, the research for startup status prediction is examined and will serve as a baseline for this research. Then, the larger body of startup success prediction will be analyzed. The best models found in current literature will be used in this study to see how those models translate to this different task of specific startup status prediction. Then, the best models for imbalanced dataset prediction will studied. The literature surrounding overcoming class imbalance is further examined to find out the best methods for doing so. Lastly, a paragraph is devoted to the algorithms used in current literature, what this study uses from these and where it differs.

2.1 Startup Status Prediction

Arroyo et al. (2019) simulated three years from 2015-2018 based on Crunchbase data and tried to predict startup status, which is most similar to the study that is proposed here. The difference lies in splitting up the operating category into two more categories: if the startup had a funding event, or not. For machine learning models, the focus was put on tree-based classifiers and the Support Vector Machine. The highest accuracy reported was 82%. The closed category was the hardest to predict, with the highest F1-score being 2%. The IPO category was not much better, with the highest F1-score being 5%. Random Forest has deemed the best model for predicting startup status when looking at the F1-score. The variables that contributed most to successful prediction were the age of the startup company, if there was a LinkedIn profile, and the raised amount in U.S. Dollar (USD). No oversampling techniques were used. Recommendations were given to try out different methods for oversampling techniques to improve performance as well as train classifiers on specific business sectors (i.e., categories of startups).

Something similar to what Arroyo et al. (2019) have done could have been done in this study as well. By simulating a timeframe and adding the class of funding event and non-funding event, it can be seen where the startup moves to. However, the choice was made to keep the original four classes, as this is more in line with current literature about startup success prediction that can be found in Section 2.2.

2.2 Startup Success Prediction

Ross et al. (2020) have made a machine learning model for startup selection and exit prediction called ‘CapitalVX’. Publicly available data from Crunchbase, data from the Securities & Exchange Commission (SEC) and data from the US Patent Office (USPTO) were used. No focus is put on a specific period, and Synthetic Minority Oversampling TEchnique (SMOTE) is used as an oversampling method. A startup that was still operating was not considered a successful startup. Only the categories IPO, acquired, and closed, were put in the model. Several models were used (Multilayer Perceptron, Random Forest, XGBoost & Ensemble) and the best algorithm, XGBoost, got an accuracy of 89.4%. Interesting to note is that the model got a 96.3% on precision and 92.9% on recall on the 'Failed' category, meaning that their model could very accurately predict which companies were going to fail.

Ang et al. (2020) tried to use machine learning to predict startup funding, post-money valuation, and success. Its data was collected from Crunchbase and spans the timeframe of 2009 - 2018. It added to the current literature by using cluster analysis to determine 16 categories for startups from the dataset. The machine learning model used for success prediction was a Neural Network. Two definitions of success were used. The first definition was only counting startups that have been acquired or have done an IPO as a successful startup, which got an accuracy of 92.48%. The second definition also included startups that were still in the growing phase (i.e., operating startups were also included), and that model only got 81.21% accuracy. There is no mention of using an oversampling method, which is unexpected, as all data downloaded from Crunchbase features this heavy class imbalance. If no oversampling method was used, then using accuracy as a way of measuring performance raises questions. The operating class has over 80% in most samples. Predicting every startup to be operating would already lead to an accuracy of around 80%. No further study was done regarding the differences between startup categories.

Krishna et al. (2016) tried to predict startup success based on Crunchbase data from 1999 - 2014 and used several classification algorithms to do that (NaiveBayes, ADTrees, BayesNet, Lazylb1, Random Forest, SimpleLogistic). Leave-One-Out Cross Validation was used. In total, nine models were made, where the biggest difference lies in adding the funding rounds. The model with series G funding performed best with an accuracy score of 96.3%. However, when series G funding is dropped, the accuracy score of the model drops to around 87%. The split of successful companies vs. failed companies was 73% - 27%, but it was not mentioned how this percentage was reached. The top performing models contain Random Forest, SimpleLogistic and Naive Bayes and no definition of startup success was given. Startup status must have been categorized as the IPO and acquired category, as adding the operating class would feature a heavier class imbalance.

Lastly, there have been a few master theses that focus on these problems. The one that is most similar to the research that is proposed is the thesis written by Ünal (2019). It focused on taking a machine learning approach to startup success prediction. In this case, startup success was characterized by a startup being either in an operating, acquired, or IPO class. The data was collected from Crunchbase and spans the range of 2009-2018. The data features a heavy class imbalance (96% vs. 4%) which is mitigated by using the Adaptive Synthetic Sampling Approach (ADASYN). Furthermore, Logistic Regression, Decision Tree, Random Forest, and XGBoost algorithms were used to make predictions. The Random Forest and XGBoost model both performed well, scoring over 94% accuracy, 97%+

sensitivity, and 86%+ specificity. The model finds that last funding to date, first funding lag, and company age are the best predictors for startup success.

Other research has been done that does not only focus on quantitative data from Crunchbase. Sharchilev et al. (2018, p. 2287) for instance try to predict startup success and use the Crunchbase data, as well as data from monitoring the web presence of a startup. Startup success in this paper is categorized differently than others have done. It focuses on predicting funding events of startups that have already got seed or angel funding and then predicts if they will get further funding. The model that performs best scores 85.4% on the ROC-AUC curve.

Overall, it can be seen that tree-based classifiers work best across the literature (Krishna et al, 2016; Ross et al, 2020; Ünal, 2019). No scholar examines the differences between startup categories or uses several oversampling methods for overcoming class imbalance. The definitions of startup success differ per scholar as well.

2.4 Algorithms for Multi-Class Imbalanced datasets

There is a growing body of literature concerning the best algorithms for multi-class imbalanced dataset prediction. Tanha et al (2020) did an experimental review of boosting methods in imbalanced data classification. A wide range of boosting algorithms was tested, ranging from the AdaBoost algorithm to the XGBoost algorithm. In the end, CatBoost & LogitBoost were deemed the best performing boosting algorithms for multi-class imbalanced datasets. Rajesh & Duli (2018) focused on classifying the imbalanced dataset containing ECG beats. Several oversampling techniques were tried and AdaBoost performed well on this dataset containing imbalanced data.

2.5 Methods for overcoming class imbalance

Research towards finding the best method for oversampling the minority class is numerous. Amin et al. (2016) used six methods for predicting customer churn and found that Mega-trend Diffusion Function (MTDF) was best. Le et al. (2018) focused on bankruptcy prediction and used five variations of SMOTE, where the SMOTE with the Edited Nearest Neighbor (SMOTE-ENN) was deemed to be the best performer. Kong et al. (2020) also studied six different techniques and found that the Rapidly Converging Gibbs algorithm (RaCOG) performed well, and in general, oversampling methods that consider the distribution of the minority class performed better. Bajer et al. (2019) looked at all of the different variants of SMOTE and found that weighted-SMOTE and random-SMOTE performed highest in accuracy. Kovács (2019) used 85 methods to overcome class imbalance and found that ProWSyn, SMOTE-Iterative Partitioning Filter and polynom-fit-SMOTE are the top methods. Furthermore, Dudjak & Martinović (2020) looked at binary classification and found that random-SMOTE was best. Lastly, Bej et al. (2020) proposed a new method for overcoming class imbalance: Localized

Randomized Affine Shadowsampling (LoRAS), which performed better than standard oversampling techniques such as SMOTE. From this literature, it can be seen that SMOTE and its variants perform best for overcoming class imbalance, though there are some newcomers such as the LoRAS sampler.

2.6 Algorithms used in the literature

Almost all of the related works make use of tree-based classifiers with good performance. These classifiers are chosen because they handle data with many features well and can be trained quickly, as was seen in Ross et al (2020). However, in the one study that also focuses on the specific startup status (Arroyo et al., 2019) the performance is still weak. This begs the question of how these tree-based classifiers work for this prediction task, and how they can be improved. One of the things that is proposed and added in this study is the oversampling methods. Most of the current literature surrounding startup status either ignores the majority class in the dataset (operating) or adds it to the success class. By doing this, no clear answer can be given in what class the startup will come, and that is the importance of this study. This study also works with tree-based classifiers in order to compare the results to current literature. Other Machine Learning algorithms are used as well, such as the Neural Network of Ang et al. (2020) and the Naive Bayes classifier of Krishna et al (2016). This was done in order to compare the best performing models of the current literature on this dataset with a longer timeframe and different features. Other models that performed well in the literature surrounding imbalanced dataset prediction were also added to see how these models perform on this dataset. Finally, the Elastic Net classifier of Ang et al. (2020) was also included to see how a linear classifier performs on this dataset.

3 Methods

In this chapter, the theoretical background of the models and methods for overcoming class imbalance will be given. The first section discusses the two clustering algorithms used for finding out the startup categories. The second section the models for predicting startup status. Finally, the methods for overcoming class imbalance will be discussed.

3.1 Clustering Algorithms

Two types of clustering algorithms are used in this thesis. The first one is the K-means algorithm, the second one the Latent Dirichlet Allocation algorithm (LDA). The K-means algorithm is chosen as it is one of the most used clustering algorithms (Brownlee, 2020). The LDA algorithm is chosen as it is also used by Ang et al. (2020) doing a similar task of clustering startup categories.

3.1.1. K-Means Clustering

K-means clustering is an unsupervised machine learning algorithm that is used to find clusters of data in a dataset (MacQueen, 1967). First, the number of clusters *k* should be specified so that the data points can be assigned to those clusters. Then, the mean of all the points for each cluster of *k* will be computed and new centroids will be placed at the mean of those points. This will be done until the centroid positions do not change anymore.

Before using a clustering algorithm, the optimal number of clusters might not be known. In that case, it is best to try out different values for K and see which is the best one (P. Sharma, 2020). The best value of K can be found by trying out different values of K, plotting the silhouette graphs and taking the average silhouette scores.

Silhouette analysis is a way to find out how well clusters are separated from each other. This ranges from [-1, 1]. A silhouette scores near 1 indicates that the cluster is well separated from other clusters. A value of 0 means that the cluster is close to another cluster, and a value of -1 means that the cluster samples might have been assigned to the wrong cluster.

3.1.2. Latent Dirichlet Allocation

Latent Dirichlet Allocation is an unsupervised learning algorithm that is most used in Natural Language Processing (Blei et al., 2003). It is an example of a topic model. It maps the different words in the target variable to different topics.

The result is a list of topics with the most prevailing keywords in that topic. In this way, it can be seen what kind of cluster it is, and which words are most representable for that cluster. A range of clusters *k* can be implemented and evaluated by the coherence score. The coherence score looks at the distance between words within a topic. The number of topics with the highest coherence score is the best one. If several of the same keywords are shown in different topics, this is a sign that the number of K is too high (Kumar, 2018).

The difference between the K-means algorithm and the LDA is that the K-means partitions the data into disjoint clusters whilst LDA assigns the data to a mixture of topics. In K-means a startup would be entirely Finance category, whilst in the LDA a startup could belong to 60% Finance, 40% Information Technology for instance.

3.2 Prediction Algorithms

Seven different prediction algorithms are used: Random Forest, CatBoost, AdaBoost, XGBoost, Elastic Net, Naive Bayes and Neural Network. These algorithms have been shown to perform best on the prediction task of either startup success or status. The reason for using these specific algorithms is given in Section 2.6.

3.2.1. Random Forest

A random forest is a collection of decision trees. Decision trees are built from leaf nodes. These nodes perform a logical test on the predicting variables, which give a probability of a data point belong to one class or another. Together, all of these nodes decide the classification of a startup (Dietterich, 1995)

One of the major issues with decision trees is overfitting (Dietterich, 1995). Overfitting means that the model fails to generalize the prediction of a training set to a testing set. In other words, a model cannot predict well on new data and overfits the training data. That can be mitigated by using random forests, which are a collection of decision trees (Breiman, 2001). This collection of decision trees (e.g., 100), is thus called a random forest and these decision trees are trained on the data. The data used is different for every decision tree, as every decision tree randomly samples from the dataset with replacement. Furthermore, these decision trees can only pick features from a random subset of features, increasing the variations between decision trees. When all these decision trees are trained, they take a vote and the label that has the most votes will be used for prediction. All in all, this leads to less overfitting.

3.2.2. CatBoost

Another algorithm that makes use of decision trees is the CatBoost algorithm. It can be seen as an extension of normal gradient boosting algorithms, and it is relatively new, being developed in 2018 (Prokhorenkova et al., 2018). It works well for categorical variables, and it does not make use of one-hot-encoding (Seggura, 2020).

First, it creates permutations by shuffling the data. Then it assigns a default value to the first few examples of the categorical variables. After that, the values of other rows will be calculated by counting the number of positive labels of examples with the same class and performing a calculation:

$$(countInClass\ +\ prior) \div (totalCount + 1).$$

When the calculations are done, symmetric binary trees are made for each shuffle of the data. The final step is the building of new models. This is done by creating permutations of the rows and then looking at the squared number step of the previous examples.

3.2.3. AdaBoost

AdaBoost, also known as adaptive boosting, is another boosting algorithm. The basic premise is that it assigns higher weights on each run instance to incorrectly classified instances, thus hoping to classify the more difficult instances (Freund & Schapire, 1997). Contrary to the CatBoost algorithm, AdaBoost can work on any classifier, but usually make use of a collection of decision trees. Boosting algorithms work in the way that first a model is made, and then subsequent models are made that learn from the mistakes of the previous models.

Furthermore, AdaBoost works with *Decision Stumps.* These are similar to a decision tree, but only have one node and two leaves. AdaBoost then uses a large number of these stumps and combines these into one large model.

3.2.4. XGBoost

Similarly to the CatBoost and Random Forest algorithms, the XGBoost algorithm makes use of multiple decision trees. It differs from CatBoost as it cannot handle categorical features on its own. The difference between a boosting algorithm such as AdaBoost and XGBoost is the gradient boosting involved. This means that it minimizes its loss by making use of the gradient descent algorithm. XGBoost goes even further than normal gradient boosting algorithms, using software and hardware optimizations to get the best results (Morde, 2019). The sequential tree models that learn from each other, do so in the way that they correct errors made by the previous models (Chen & Guestrin, 2016).

3.2.5. Elastic Net

The elastic net classifier is a multinomial logistic regression algorithm with a penalty set to Elastic Net. Multinomial logistic regression can be seen as an extension of the normal binary logistic regression, in order to predict a dependent variable with more than two classes (Böhning, 1992). Elastic Net is a regularization method that combines the L1 and L2 penalties of Least Absolute Shrinkage and Selection

Operator (LASSO) and RIDGE methods (Algamal & Lee, 2015). One of its parameters is *alpha,* where *alpha* is the mix between RIDGE (*alpha* = 0) and LASSO (*alpha* = 1).

The Elastic Net classifier is different from the other algorithms as it is focused on linear classification instead of non-linear classification in Random Forest, XGBoost and CatBoost.

3.2.6. Naive Bayes

Naive Bayes is a classification algorithm that is widely used and quite simple. The basis of the algorithm is Bayes Theorem, which is a method for calculating probabilities and conditional probabilities (Bayes, 1763). The algorithm has a conditional independent assumption meaning that it handles features independently. Its advantage is that it is simple to build and understand, but it can also make mistakes due to its independence assumption that might not always be true. The goal of the algorithm is to find the most likely prediction, and that is done by solving the bayes theorem for every option. The option with the highest likelihood wins and is chosen (Zhang, 2020).

3.2.7. Neural Network

A neural network tries to replicate the way the brain works. This makes it excellent at finding underlying relationships. It makes use of a lot of neurons to learn about a problem (McCollough & Pitts, 1943). Three types of units that can be found in neural networks: input, hidden and output units. Input units get the information onto which the training is to be done. The hidden units form the majority of the units and reside in between the input and output units. Lastly, the output units tell something about how well the model has learned (Krogh, 2008). Both neural networks and tree-based classifiers find non-linear relationships, but neural networks tend to require more computation power (Nitze et al., 2012).

3.3 Methods for overcoming class imbalance

Six different methods for overcoming class imbalance are used: SMOTE, ProWSyn, Random SMOTE, ADASYN, SMOTE-ENN and LoRAS. In current literature, these methods are either a baseline or perform best on similar tasks. Most of these methods are either the SMOTE method or slight changes in the SMOTE method. All of them focus on oversampling the minority class, where the SMOTE-ENN algorithm also adds under-sampling with the ENN technique. A focus was put on the oversampling techniques over under sampling techniques. The use of under sampling techniques would delete information in the training dataset that could be vital for the prediction task at hand. Also, the dataset is split up into different categories of startups, which reduces the available training data. As the minority classes are already quite small, it is deemed better to oversample those, than to delete instances of the majority class with under sampling.

3.3.1. SMOTE

The Synthetic Minority Over-sampling TEchnique adds new synthetic samples to the minority class that are linearly correlated with the parent instance. These are created by getting the k-nearest-neighbours of the minority class (Chawla et al., 2002). The new samples are then chosen across the line of the k-nearest-neighbours. The success of the method in comparison to other methods lies in its oversampling of the minority samples, it does not remove instances of the majority class. Contrary to replication, this leads to larger decision regions for a classifier, which in turn makes the model better.

3.3.2. ADASYN

ADASYN is a variation of SMOTE. It is in essence the same as SMOTE, but it adds a random value to the new synthetic points. This means that the synthetic samples are not linearly correlated to the parent anymore and are more scattered (Haibo et al., 2008). The algorithms adds more data in samples that are harder to learn, than samples that are easier to learn.

3.3.3. Random SMOTE

Random SMOTE is another variation of SMOTE that puts its main focus on the failure of SMOTE to change the sparsity of the minority class samples. Instead of the range used in SMOTE, Random-SMOTE can generate new minorities in a wider range and thus change the sparsity of the minority class samples (Dong & Wang, 2011).

3.3.4. ProWSyn

Another variation of SMOTE is the ProWSyn. It builds further on the Corpus Based Statistics-Oriented method (Barua et al., 2011), which tries to mitigate the failures of the k-nearest-neighbours algorithm in SMOTE. It differs from it by using a different weight generation technique (Barua et al., 2013). This weight generation technique weights values for the minority sample based on the distance it has to the boundary. This makes sure that the synthetic samples are evenly created across the minority dataset.

3.3.5. SMOTE-ENN

The last variation of SMOTE used is SMOTE-ENN. This is a variation that uses over-sampling from SMOTE and cleaning with the Edited Nearest Neighbors (an under-sampling technique). The Edited Nearest Neighbors works in the following way. It uses k=3 to locate the instances in the dataset that are misclassified which are then removed (Broder et al., 1985).

3.3.6. LoRAS

LoRAS is a new method proposed to overcome the limitations of SMOTE (Bej et al., 2020). It works by first identifying the k-nearest-neighbours of a point *p.* Then it takes shadow points from those k-nearest-neighbour, which are points close to the original point that is taken from a normal distribution. Subsequently, the algorithm generates random new points from the area in between the newly made shadow points.

4 Experimental Setup

4.1 Data

4.1.1. Dataset Description

The data is taken from the research application programming interface (API) of Crunchbase. Crunchbase.com is a website that houses information about companies and startups all over the world. Most research in this sphere uses Crunchbase as either the main or supplementary data source (Ang et al, 2020; Arroyo et al, 2019; Krishna et al., 2016; Ross et al., 2020; Sharchilev et al., 2018). It contains data about companies and startups going back from 1902. Four comma-separated values (CSV) files were taken from the total 18 CSV files available. These were the only ones that were relevant for the features created and explained in subsequent sections. After doing feature transformations, these datasets will be merged into one final dataset. A summary of the contents of each file will now be given.

Organizations

The organizations file contains information about startups that have made an account on Crunchbase.com. In total, the data spans 1,301,141 startups with 41 features, ranging from companies founded in 1902 to companies founded in 2021. The data is divided into two main sections: general features and investment features. General features include the name, country code, founded on date, email, phone and so forth. The investment features have information about the investments made into the company and tell something about its performance. For example, the total amount of funding a startup has received, the number of funding rounds there were and when the last funding was received.

Funding rounds

The funding rounds file contains information about all of the various funding rounds done for all of the startups. Funding rounds can range from seed funding (the first capital a company raises) to series E funding (the sixth round of funding for a startup) to an initial coin offering, which is the typical way for a cryptocurrency startup to go public (Investopedia, 2020b). It houses information about 388,348 funding rounds and has 24 features. The data ranges from general information such as country code, state code, city, to specific information about the funding rounds such as the raised amount in USD, the investment type and the investor count.

People

The people file contains information about the various employees working at the startup companies. It has information about where the person comes from, what their job title is and their various social media URLs. In total, the dataset houses information about 1,136,209 persons with 22 features.

Degrees

The degrees file contains information about the type of degrees that the employees of a startup have pursued. It contains information about failed and successful degrees. The data houses information about 374,464 degrees with 17 features. Examples of features are the person that pursued the degree, if it was completed and at what institution it was pursued.

4.1.2 Data Preprocessing & EDA

After collecting the data from Crunchbase, all the subsequent processing and analysis was done using Python 3.8.3. in Anaconda's JupyterLab (Van Rossum & Drake, 2009). The libraries used were Pandas (McKinney, 2010), NumPy (Harris et al., 2020), and Matplotlib (Hunter, 2007). The main file that was pre-processed was the organizations.csv file which housed the bulk of the data about startups. The other files are merged on the startup name with this file. In this way, only the relevant startups are taken into account. For example, if a startup is not in the pre-processed organizations file, but can be found in the fundings file, it will not be taken into account as this means it misses key values in the organizations file.

Organizations Cleaning

Four variables in this study are essential to have values in. First, category_list, because this is the variable that is used for the creation of startup categories (see Section 4.2.1). Second, founded_on, because only startups from 2000 - 2015 are targeted. Third and fourth, total_funding_usd & num_funding rounds as these are important predictors in current literature (Arroyo et al., 2019; Ross et al., 2020). The first thing that was done was delete the rows with missing values in these variables. This led to a reduction from 1,301,141 startups available to 142,920. Then, startups that had a founding date below 2000 or above 2015 were also deleted. This resulted in a final organizations file of 91,801 companies.

Funding rounds, person & degrees cleaning.

For the funding rounds file, data was deleted that had an announcement date of the funding rounds before 2000 or after 2015. This led to a reduction from 388,348 funding rounds to 179,710 funding rounds available.

The people file contains information about all employees, housing info about 1,136,209 persons. First, the rows that did not contain a value for the featured_job_title variable were deleted. This led to a reduction to 677,664 people available. This research focuses on the characteristics of founders as a predictor for startup status, as these features performed well in current literature. The rows where the featured_job_title did not include "Founder" were also deleted. This led to a further reduction to 168,179 people available.

Lastly, in the degrees file, there were 374,464 rows available. Only the completed degrees were taken into account as this is a feature that is also used by other scholars (Arroyo et al., 2019; Ross et al., 2020). This led to a reduction of 203,200 rows available.

4.1.3 Feature Engineering

The basic features of the model are the same as in other literature (Arroyo et al., 2019; Ross et al., 2020; Ünal, 2019), as this is taken from the Crunchbase dataset. There are some notable differences. Ross et al. (2020) use data from the SEC and the USPTO which will not be in this dataset. Not all features of that model are described in the paper, therefore some feature transformations that have been done on the Crunchbase data might also not be included. The same goes for the features of Arroyo et al. (2019). In general, the features and the feature transformations that are known are added to the model. Lastly, all features that were used by Ünal (2019) are used in this model.

In the organizations file, several feature transformations were done which was the same for the other files. After transforming the features in the other files, they were merged with the main organizations file. Below an explanation of the transformations per file is given.

Organizations file transformations

The creation date of a startup was in a date format, this was transformed to a year_founded_on variable that only contained the creation year and was stored as an integer. This was done in order to make the other variables. A company age variable was made by subtracting the year that a startup was founded on from 2015.

The dataset houses information about whether a startup has a Facebook URL, LinkedIn URL etc. These features were transformed into a binary version where 0 was assigned if there was no URL and 1 if there was an URL. In this way, it could be seen which startups have Facebook, LinkedIn, Twitter etc. The country_code variable, which contains the country of origin of a startup, is transformed into a continent variable that contains the continent of a startup. The full list of these transformed variables can be seen in Table 9.1. in the Appendix. Lastly, a description_length variable was made by counting the length of the short_description variable.

Fundings file transformations

From the fundings file, three variables were made. The first one being the year_last_funding_on, which was a transformation of the last_funding_on, which contained the date of the last funding, to a variable that only contained the year. This was stored as an integer. A similar transformation was done by transforming the announced_on variable. This is a variable containing the date of the first funding event. This was transformed into the year_first_funding_on.

Several variables were created that contained information about the last funding round. First, the last funding round per startup was found. After doing that, the variables investment_type, raised_amount_usd, post_money_valuation_usd & investor_count of the last funding round were transformed to last_round_investment_type, last_round_raised_amount_usd, last_round_post_money_valuation_usd & last_round_investor_count.

After adding these variables to the organizations dataset, the first funding lag variable was made by subtracting the year_founded_on variable from the year_first_funding_on variable. This was done in order to see how long it took a startup to get its first funding. The last funding lag variable was created by subtracting the year_first_funding_on variable from the year_last_funding_on variable, to capture the time between the first funding of a startup and the last.

People & degree file transformations

From the people and degrees file, four variables were made: founders_count, founders_female_count, founders_male_count and founder_degree_count. The founders_degree_count variable was made by grouping the degrees file by name and then counting the number of degrees. This was then merged with the people file. Grouping the persons per startup created the final founders_degree_count variable. Then, the founders_count variable was made by counting the number of founders per startup, the founders_female count by which of these were female, and founders_male for which of these were male.

4.1.4 Further Processing

After merging all of the above datasets into one large dataset, the variables that did not contain information were dropped. Final feature transformations were done. The last_round_investment_type contained 27 different investment types. As the categorical variables are dummy encoded, this would lead to an extra 27 variables and thus increasing the dimensionality of the dataset. Therefore, it was chosen to put the investment types that had lower than 1000 instances into the “Other” category.

Furthermore, two of the employee_count categories (5001-10000 & 10000+) were transformed into the 5000+ category to reduce dimensionality. Lastly, the continent, employee_count and last_round_investment_type variables were dummy encoded.

Not every startup has data about every feature in the model. An imputation method had to be chosen to combat the missing data in the dataset. The *IterativeImputer* from Scikit-Learn (Pedregosa et al., 2011) is chosen. It is a way of imputing that estimates a feature based on all of the other features.

The final dataset houses 90815 companies and 26 features. The distribution of the target variable can be seen in Table 4.1. All the features used for the final model can be found in the Appendix Table 9.1.

Table 4.1. Distribution of the target variable startup status

Startup Status	Amount of startups	Percentage
Operating	68619	75.6%
Acquired	11051	12.2%
Closed	7453	8.2%
IPO	3692	4.1%

4.1.5 Setup Additional Test Set

This study uses an additional test set as a robustness check and to see how well the best combinations of models and oversampling methods perform on newer data. This is data from startups that were founded in 2016-2019, and thus do not contain any startups of the 2000-2015 dataset. The dataset is made by following the same steps outlined in sections 4.1 - 4.1.4.

4.2 Modeling

The setup of the clustering and prediction algorithms, as well as the methods for overcoming class imbalance, will now be explained. 5-fold-cross validation is used and the average of the five test scores is reported. This is done to get less biased results (Fushiki, 2009). A choice could also have been made for 10-fold-cross validation as this reduces bias even more. However, this was not done due to computational reasons.

4.2.1 Clustering Algorithms Setup

Two clustering algorithms were fitted to the category_list variable of the dataset. The category_list variable of the dataset contains one or a multiple of the 46 categories that can be chosen via the website. Examples are (Publishing, Social Media, Social Media Management) or (3D Technology, Developer Tools). The goal of these algorithms is to get a set of startup categories. For both algorithms, a range of clusters from 5-20 will be implemented.

K-Means Clustering

For implementing the K-means algorithm, Scikit-learn (Pedregosa et al., 2011), Pandas (McKinney, 2010), NumPy (Harris et al., 2020), and Matplotlib (Hunter, 2007) are used. The *Kmeans* function of *sklearn.cluster* is used, and English stop words were removed. The Term Frequency - Inverse Document Frequency (TF-IDF) is calculated by vectorizing the category_list variable which resulted in a final variable. As noted before the range of K 5-20 was tried.

Per value of k, the silhouette plot was plotted as well as the average silhouette score. Lastly, the words of the clusters were plotted to see what kind of words belonged to which cluster. In this way, next to the silhouette score, the results can be evaluated on the words that belong to the clusters. If certain clusters contain the same words, then this means the clusters are not well-separated.

Latent Dirichlet Allocation

For implementing the Latent Dirichlet Allocation algorithm, Pandas (McKinney, 2010), Gensim (Rehurek & Sojka, 2011) and Natural Language Toolkit (Bird et al, 2009) libraries are used. English stop words are removed. A dictionary of words will then be made, and from that dictionary, a corpus of words is made. After fitting the model, the keywords of the topics are printed, as well as a computation of the coherence score.

4.2.2. Set-up Prediction Algorithms

The target variable "status" is label encoded before training the algorithms. All models are implemented using the base values. If something different was done, it is noted below:

- The Random Forest is implemented via Scikit-learn (Pedregosa et al., 2011) with *RandomForestClassifier.*
- The CatBoost algorithm is implemented via the *CatboostClassifier* in Python. (Banerjee, 2020).
- The XGBoost algorithm was trained and implemented using the *XGBRegressor* from xgboost (Chen & Guestrin, 2016).
- The AdaBoost classifier is implemented using Scikit-learn (Pedregosa et al., 2011) with *AdaBoostClassifier.*

- The Elastic Net classifier is a Multinomial Logistic Regression with penalty ‘elasticnet’. Scikit-learn is used for the implementation (Pedregosa et al., 2011). A large number of max_iter was chosen as at first a convergence warning was given.
- Naive Bayes is implemented using the naive_bayes model of Scikit-learn (Pedregosa et al., 2011) *GaussianNB()*.
- The Neural Network is implemented via Keras (Chollet et al., 2015) with *Sequential().* The simple neural network with one layer which was also used by Ang et al. (2020) was taken.

Figures 5.4. and 5.6. are made using the stacked barchart tool from at https://graphmaker.imageonline.co/

4.2.3. Set-up Class Imbalance Methods

The data is first split using the 5-fold train-test split. After doing that, the train version of the set is oversampled. The test set is not oversampled, as this would lead to issues in generalizability (Ertekin, 2013).

SMOTE, ADASYN & SMOTE-ENN are implemented using imbalanced-learn (Lemaître et al, 2017). The ProWSyn & Random Smote are implemented using smote_variants (Kovacs, 2019). Lastly, LoRAS is implemented using pyloras (Bej et al., 2020).

4.2.4. Evaluation Metrics

A baseline is taken by modelling the Random Forest without oversampling on the dataset, which is the best performing model of Arroyo et al. (2019). The models and methods for oversampling are then evaluated by using 5-fold cross-validation and taking the average test score of the five test scores. The F1-score is the main evaluation metric, and it is chosen over the accuracy score because of the highly imbalanced dataset.

$$F1score = 2 * \frac{precision * recall}{precision + recall} = \frac{tp}{tp + \frac{1}{2}(fp + fn)}$$

Using accuracy for this study would give a very biased view, as the majority class already houses 75% of the data. The F1-score gives a less biased view because it is the harmonic mean between precision and recall. The precision, recall and accuracy scores will also be given. The average of these scores across the four startup classes will be given.

5 Results

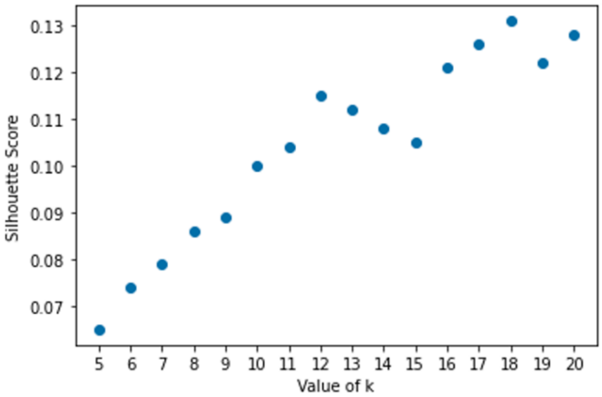
In this section, the results of the clustering algorithms and the combinations of models and oversampling methods will be given. First, the clustering results will be discussed, as this is the steppingstone for the further modelling results. Then, the modelling results on the 2000-2015 dataset will be analyzed and finally, the future predictions on the 2016-2019 dataset are discussed.

5.1. Clustering Results

5.1.1. K-Means Clustering Results

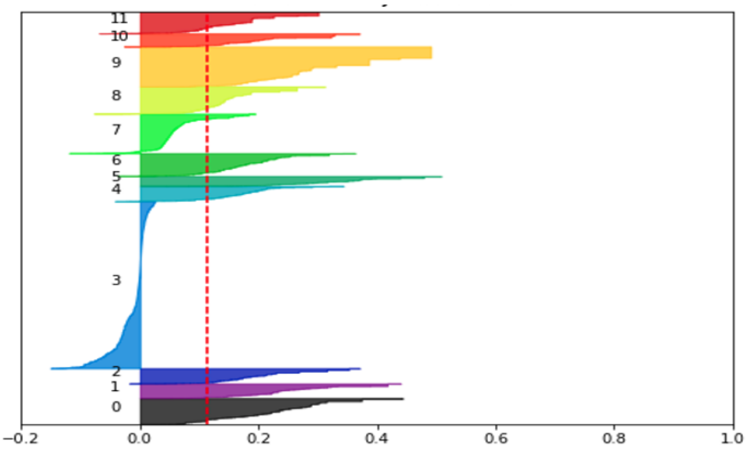
In Figure 5.1. the results of the K-means clustering algorithm per value of k can be seen. Most values of k average around a silhouette score of 0.10.

Figure 5.1. K-means Clustering Results



The clusters all had at least one cluster that had words that could not be grouped well. This lowered the silhouette score significantly. When purely looking at the silhouette score, K = 20 is deemed best. However, when looking at the clusters that were made as a result of K = 20, this does not seem the best way to separate the categories. Two clusters contained matching keywords, as well as one cluster mentioned above that contained words that could not be grouped well. That is why in the end K=12 was deemed best. Even though its silhouette score is lower (0.11 vs 0.13), the resulting clusters intuitively make more sense. To add to that, most clusters had a silhouette score ranging between 0.2 and 0.5, as can be seen in figure 5.2. One cluster, which is not used in the final categories, has a terrible score, dropping the average silhouette score of K = 12.

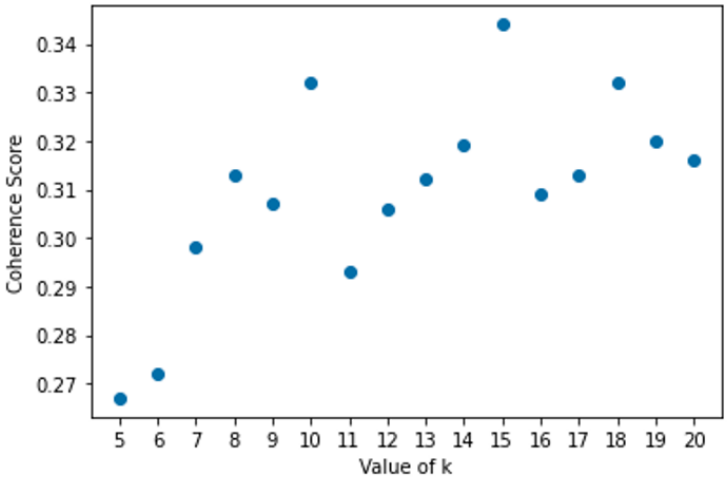
Figure 5.2. Silhouette plot of K = 12



5.1.2. Latent Dirichlet Allocation Results

An overview of the coherence scores per value of k can be found in Figure 5.3. K = 15 is the best when looking at the coherence score, which is 0.344. Similarly to the K-means algorithm, the clusters formed were inspected to see if some clusters intuitively made more sense. In this case, all values of K had around two to three clusters that were not clear cut or captured a large number of startups into one cluster that did not make sense (e.g., in K = 5).

Figure 5.3. Latent Dirichlet Allocation Results

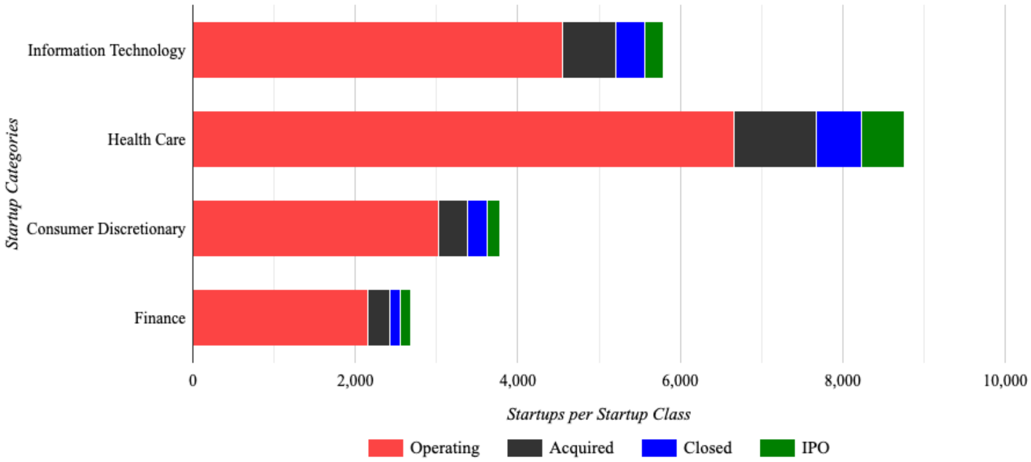


5.1.3. Optimal amount of clusters

Before moving further into the results of the prediction algorithms and class imbalance methods, a choice has to be made for the number of clusters. From the LDA results, three clusters were deemed insufficient. After looking at the words of the clusters, no clear category could be found. The same was true for one category of the K-means algorithm. The algorithms both ended up with a lot of the same clusters, but there are also some notable differences. These can be seen in Table 9.2. in the Appendix.

In the end, the K-means clusters were chosen as this included consumer discretionary, a category that can also be found in the 11 categories of the S&P 500 (Investopedia, 2021b). As twelve categories are still a lot to do a good comparison on, it was chosen to look at the differences and results of four categories instead of the 12 mentioned above. These are Information Technology, Health Care, Consumer Discretionary & Finance. Adding Consumer Discretionary from the K-means algorithm gives the ability to state differences about four very different clusters, which are also the four largest categories of companies in the S&P 500 eleven sectors (Investopedia, 2021b). The number of startups per startup status of the four categories can be found in Figure 5.4. There are a lot more startups in the Information Technology & Health Care category than in the others, and that the operating class is by far the majority in every startup category.

Figure 5.4. The number of startups per startup class and category.



5.2 Modelling Results

5.2.1. Baseline Results

As a baseline, the Random Forest model that was also used by Arroyo et al. (2019) is used, without a method for overcoming class imbalance. This model is fitted on the individual startup categories. The results can be seen in Table 5.1. The full table that includes the average precision and recall scores can be found in the Appendix Table 9.3. The average F1-score is highest in the Health Care category, and lowest in the Finance category.

Table 5.1. Baseline results of the Random Forest model.

Startup Category	Average F1-Score	Accuracy
Information Technology	0.305	0.785
Health Care	0.404	0.786
Consumer Discretionary	0.299	0.805
Finance	0.265	0.801

An interesting comparison can be made between this study and that of Arroyo et al. (2019) by looking at the F1-scores per startup status which can be seen in Table 5.2. The operating class in their study is split into two: funding-event or non-funding event. For the comparison the non-funding event was chosen. The dataset studied had different features, a shorter time-range than this study (2015-2018) and no oversampling method was used. 5-fold cross validation was also used. These average F1-scores per startup status are all higher than that of Arroyo et al. (2019).

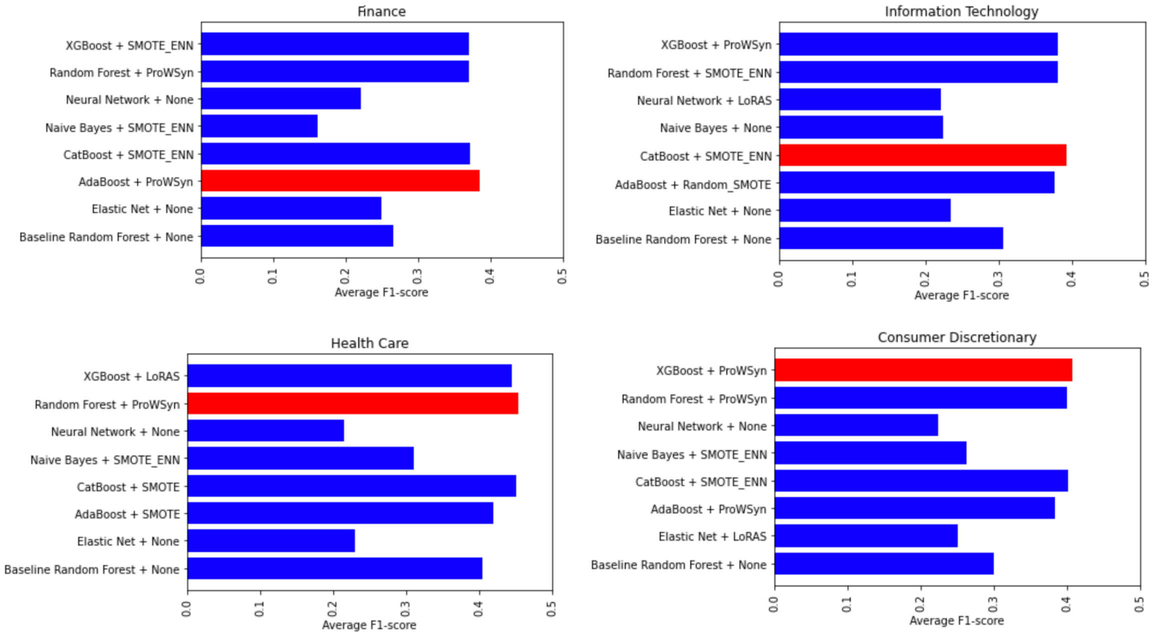
Table 5.2. Baseline F1-score per startup status and startup category compared to Arroyo et al. (2019). The dataset used in their study has different features and a shorter timeframe, with comparable outcomes of startup status prediction.

Startup Category	Acquired	Operating	IPO	Closed
Information Technology	0.093	0.873	0.184	0.069
Health Care	0.130	0.872	0.384	0.229
Consumer Discretionary	0.068	0.887	0.092	0.149
Finance	0.037	0.886	0.129	0.010
Average	0.082	0.880	0.198	0.114
Arroyo et al. (2019)	0.050	0.890	0.050	0.000

5.2.2. Overall Modelling Results

The average F1-score per startup category and machine learning model, combined with the best method for oversampling, can be found in Figure 5.5. In general, the tree-based algorithms performed best. Looking at the best methods for oversampling the minority class, the SMOTE-ENN method and the ProWSyn method come up a lot. The best performing combinations of models & oversampling methods (explained in Section 5.2.3.) all make use of either SMOTE-ENN or ProWSyn. In general, the models that performed worse than the baseline had their best performance with either the LoRAS method or no method for oversampling at all.

Figure 5.5. The average F1-score per model and best method for overcoming class imbalance.



The best models and methods for overcoming class imbalance differ per startup category. For the Consumer Discretionary category, this is the XGBoost & ProWSyn combination. Though it must be noted that the Random Forest & ProWSyn and CatBoost & SMOTE-ENN have similar F1-scores. In the Information Technology category, the CatBoost & SMOTE-ENN combination was best. In the Health Care category, this differs again with the Random Forest & ProwSyn being the best combination. In this category, the CatBoost & SMOTE combination got a similar performance. Lastly, in the Finance category of startups, AdaBoost & ProWSyn are best. All of the best combinations of model and oversampling method per startup category performed better than the baseline, albeit some better than others. In general, the ProWSyn method for overcoming class imbalance is best, being in the best combination of three of the four categories of startups examined.

5.2.3. Best performing models and oversampling methods

The full overview of the average F1-score and accuracy scores per startup category and best performing model and method combination can be seen in Table 5.3. The best model and method for the Consumer Discretionary category had an average F1-score of 40.7%, the Information Technology category 39.2%, the Health Care category 45.4% and finally, the Finance category 38.5%. The accuracy scores are notably lower than the baseline scores. The differences in these results per startup category are further confirmed by the average precision & recall scores that can be found in Table 9.4. of the Appendix.

Table 5.3. Average F1-score and accuracy per startup category

Startup Category	Best model & method combination	Average F1-Score	Accuracy
Information Technology	CatBoost + SMOTE-ENN	0.392	0.721
Health Care	Random Forest + ProWSyn	0.454	0.779
Consumer Discretionary	XGBoost + ProWSyn	0.407	0.790
Finance	AdaBoost + ProWSyn	0.385	0.664

In Tables 9.5.1.-9.5.4 in the Appendix the confusion matrices per startup category can be seen. The startup categories are all the same in that the minority classes get predicted to be the majority class (operating) most of the time. The acquired class gets predicted a lot better in the Information Technology category of startups. The closed class is the best in the Information Technology & Finance categories, and finally, the IPO class is a lot better predicted in the Health Care category when compared to the other categories.

Now, a deeper dive is made into the prediction of specific startup status per category. The F1-score per class of startup status and startup category can be seen in Table 5.4.

Table 5.4. The F1-score per startup status and startup category

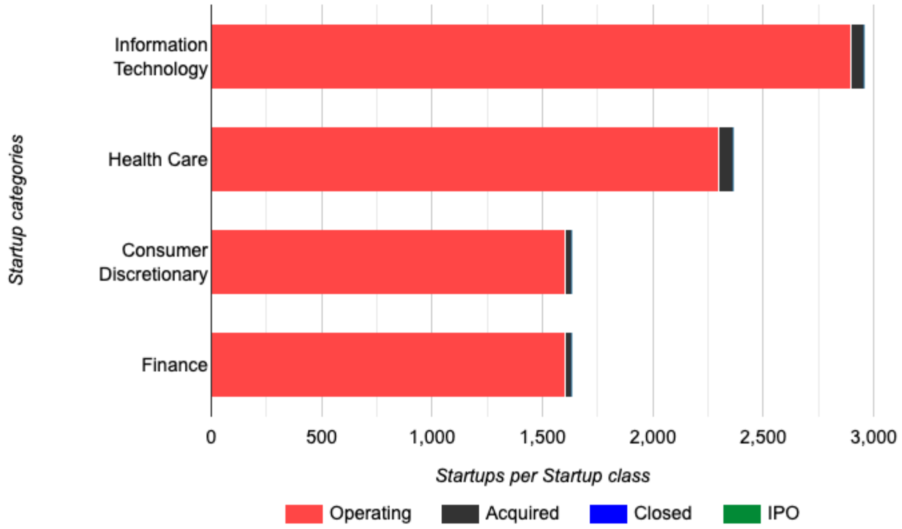
Startup Category	Acquired	Operating	IPO	Closed
Information Technology	0.242	0.818	0.300	0.208
Health Care	0.218	0.865	0.460	0.273
Consumer Discretionary	0.159	0.876	0.313	0.280
Finance	0.226	0.791	0.305	0.219
Average	0.211	0.838	0.345	0.245
Average Baseline	0.082	0.880	0.198	0.114

As expected, the operating class, which is also the majority class, performs best across all metrics. Compared to the baseline, the operating class performs a bit worse with a decrease of 4.2%. The other classes all have a noticeable improvement on the baseline. The average F1-score for the acquired class increased by 12.9%, the IPO class by 14.7% and the closed class by 13.1%. All in all, it shows that adding oversampling methods increases performance on the F1-score. The accuracy scores are all lower than the baseline. This is because more weight is given to the minority classes. A trade-off is made between accuracy and F1-score.

5.2.4. Future prediction performance on dataset of 2016-2019

These best models and methods for oversampling per startup category were then used to make a future prediction. The models were trained on a dataset from 2000-2015, and the interesting question is how this performs on future data of 2016-2019. The number of companies per status class and startup category can be seen in Figure 5.6. Most startups for all startup categories fall into the operating category. The IPO category has a low number of startups in the Information Technology, Consumer Discretionary & Finance sectors.

Figure 5.6. The number of startups per startup class and category in the test set



The average F1-score and accuracy per startup category on the test set, can be seen in Table 5.5. The results are a bit worse than on the dataset of 2000-2015 but hold up relatively well. The biggest difference can be seen in the Consumer Discretionary category with a difference of 8.7% on the average F1-score. The Information Technology category performs the best with a difference of 4% on the average F1-score.

Table 5.5. Average F1-score and accuracy per startup category on the test set

Startup Category	Average F1-Score	Accuracy
Health Care	0.408	0.909
Information Technology	0.356	0.955
Consumer Discretionary	0.315	0.965
Finance	0.333	0.830

The F1-score per startup status and startup category can be seen in Table 5.6. When looking at the results per specific startup status, the explanation for this performance can be found. In general, the scores on the operating class (majority) class are very high. The scores on the Acquired and IPO class are quite low. The results of the closed class hold up relatively well. This shows that the model predicts the majority class for almost every instance. This can also be because the class imbalance is even heavier in the test set, due to the startups only between zero and four years old. The chance of being acquired or having done an IPO is a lot lower than companies that have been operating for longer.

Table 5.6. The F1-score per startup status and startup category on the test set

Startup Category	Acquired	Operating	IPO	Closed	Average
Information Technology	0	0.977	0	0.499	0.356
Health Care	0.132	0.954	0.308	0.238	0.408
Consumer Discretionary	0.077	0.982	0	0.200	0.315
Finance	0.067	0.909	0.231	0.128	0.333
Average	0.069	0.956	0.135	0.266	0.353

6 Discussion

The goal of this study was to answer the research question "*To what extent can startup status be predicted for different categories of startups, based on publicly available data?*" The findings will now be discussed per sub-question.

6.1. Sub-question 1

SQ1: What are the different categories of startups? This question will be answered by training two clustering algorithms on the category_list variable dataset.

K-means clustering and the Latent Dirichlet Allocation algorithm were trained on the category_list variable of the dataset. The category_list variable of the dataset contains a list of categories that a startup belongs to that can range from 1 to 10 categories.

Both clustering results were not perfect. The best performing clusters (as measured by the silhouette and coherence score), still had clusters that, when looking at the words associated with those clusters, could not be categorized. In the end, the K-means clusters were chosen with K = 12. This was not in line with Ang et al (2020), that found 16 clusters of startups. However, it does come closer to the 11 categories of companies that are used by the S&P 500 (Investopedia, 2021b). Using clustering algorithms might not be the best way to find the different categories of startups in this case. Other clustering algorithms could be tried to increase the performance, or the categories of the category_list variable could be hand mapped to the S&P 500 categories. In the end, from the 12 clusters defined by the K-means algorithm, four were chosen to further examine the research question. These were: Information Technology, Health Care, Consumer Discretionary & Finance.

6.2. Sub-question 2

SQ2: Per category of startup, how does the model of Arroyo et al. (2019) perform on this dataset with a longer timeframe and different features, when evaluated by the F1-score?

When comparing the scores of precision, recall and F1-score to the study of Arroyo et al. (2019), these all performed better than in their study in every startup category. This might be due to the addition of variables that were not used in their research but did perform well in current literature. However, these baseline scores are still too low if they are to be used by startup founders & venture capitalists. The model thus performs better on this dataset that but can still be improved.

6.3. Sub-question 3

SQ3: Per category of startup, which machine learning model and method for overcoming class imbalance are best suited when evaluated by the F1-score? Seven different models and six different methods will be tested.

The best performing models and methods for overcoming class imbalance can be seen in Figure 5.5. Tree-based algorithms all performed best. This is in line with Arroyo et al. (2019) that found the Random Forest algorithm to be the best and tree-based algorithms in general. Other literature also supports this point (Ross et al., 2020; Ünal (2019). On the other hand, the other models tested (Neural Network, ElasticNet & Naive Bayes) performed worse than the baseline in every combination, which is not in line with current research (Ang et al, 2020; Krishna et al., 2016). However, it has to be noted that their prediction task differed from the one that is done here. The reason that Elastic Net performed so poorly was probably due to the non-linear separability of the data. This is also further confirmed by the tree-based algorithms performing well, which are known to be able to classify data well that is not linearly separable (Polikar, 2006). Naive Bayes also performed poor, which might be due to the conditional independence assumption of Naive Bayes. Naive Bayes assumes that the features are conditionally independent. For a lot of features in this dataset that is probably not the case. For instance, it can be assumed that the total funding raised is correlated with the total amount of investors a startup has. There could be several reasons that the Neural Network performs badly on this dataset. The basic model of Ang et al. (2020) was taken but different variations of the Neural Network might work a lot better. The model used was still quite simple. More complex Neural Networks might perform better (e.g., adding multiple layers and dropout layers). All in all, differences could be seen between categories. All categories had a tree-based classifier that was deemed best, though the classifier differed per startup category. The best oversampling method was the same in three of the four categories.

These best performing models and methods for overcoming class imbalance are better than the baseline and highlight the importance of using oversampling techniques in multi-class imbalanced datasets. The F1-scores reached are still too low if these models want to be used by startup founders and venture capitalists. Recommendations for future research will be given in section 6.7.

6.4. Sub-question 4

SQ4: What are the differences in types of errors between startup categories?

There were notable differences to be seen between startup categories. A startup that is acquired by another company is best predicted in the Information Technology category of startups. A failed startup is best classified in the Information Technology & Finance startups. Lastly, a startup that goes to the stock market through an IPO is best predicted in the Health Care category when it is compared to the other categories. It is difficult to say what the underlying reasons for these are, but these results highlight the differences between startup categories and support the argument of training models on specific categories of startups.

6.5. Sub-question 5

SQ5: How do the best models and imbalance methods per startup category perform on more recent data of 2016-2019?

To simulate how the best models and imbalance methods predict future data, an additional dataset spanning the next 4 years after the original dataset was taken. Results can be seen in Table 5.5. Performance was a bit worse than on the 2000-2015 set but held up relatively well. The Consumer Discretionary sector is the hardest to give a future prediction of, and Information Technology the best. In general, the majority class of startups received the highest scores on recall, precision and F1-score. This means that the model is not able to distinguish the majority class from the minority classes, even with the use of oversampling methods. It might be because a significant difference between the classes cannot be found in this dataset. Scores on the IPO and acquired class were especially bad, which might be due to the low sample of IPO class in the test set. The startups that have done an IPO are thus hard to classify correctly, and the reason might be that the differences between the operating and IPO startups are not that big as previously thought or can't be seen in the quantitative data.

6.6. Research Question

RQ: To what extent can startup status be predicted for different categories of startups, based on publicly available data?

The first question was to find out the different categories of startups. The second to see how a baseline model would perform on these prediction tasks. The last three sub-questions were targeted at the performance of the best models and methods for overcoming class imbalance. To answer the research question: to some extent. The best average F1-score was reached in the Health Care sector with a score of 45.4%. Four of the seven models performed better than the baseline, showing that using oversampling methods increases performance. In general, tree-based algorithms and the SMOTE-ENN or ProWSYN method for overcoming class imbalance are deemed best for this prediction task.

6.7. Implications, Limitations and Future Research

This study showed a significant improvement on the model of Arroyo et al. (2019). It shows that startup status can be predicted, but that there is also still a way to go in order to use a model for startup founders and venture capitalists. Further developing these models could lead to aiding startup founders and venture capitalists so that less startups fail, and more investments go to the right startups. This is turn has a positive effect on the economy and society at large. Furthermore, the importance of using oversampling methods is illustrated by the increase in performance over the baseline without using oversampling methods. Differences between startup categories were also seen which highlight the importance of training models on specific categories of startups. All in all, this study adds to the current literature by showing that adding oversampling techniques increases performance, that there are notable differences between startup categories, and confirming that tree-based classifiers are best for this prediction problem.

However, this study also has some limitations. As noted in the first sub-question, the clustering algorithms did not perform well. Future work could make use of other clustering algorithms or, hand map the categories to the S&P 500's classification of industries. This study also did not make use of all features available in the literature that are deemed indicative of predicting startup status or startup success. The best performing model in the literature (Ross et al, 2020) makes use of USPTO data as well as data from the SEC. It is recommended for future studies to include data from the USPTO and the SEC, as well as adding more feature transformations from the Crunchbase dataset.

Another limitation of this study was the lack of hyperparameter tuning. The goal of this study was to find the best model and method for overcoming class imbalance, and no emphasis was put on the tuning of hyperparameters. A recommendation for future research is to pick the best models and methods for overcoming class imbalance and use hyperparameter tuning to increase performance. The weak performance of this model could also show that purely using quantitative data does not predict startup status well. Other more qualitative factors (e.g., startup culture) could lead to a company going for an IPO, being acquired or closing down that cannot be included in quantitative models. Lastly, a broader study could be done to compare the differences of all startup categories instead of the four done in this study.

7 Conclusion

Different machine learning models and methods for oversampling the minority class were compared in this study. A dataset was created by combining several files, preprocessing and feature transformations, and ranges startups from 2000-2015. The categories of startups studied can be found in Figure 5.4., which answers the first question: *"What are the different categories of startups?"*. A baseline was then taken by using the best performing model of Arroyo et al. (2019), these results can be found in Table 5.1., which answers the second research question: *"Per category of startup, how does the model of Arroyo et al. (2019) perform on this dataset with a longer timeframe and different features, when evaluated by the F1-score?"*. The best models and methods for overcoming class imbalance can be found in Table 5.3. This answers the third research question: *"Per category of startup, which machine learning model and method for overcoming class imbalance are best suited when evaluated by the F1-score?"*. The differences in errors between startup categories were then looked at and can be found by comparing the confusion matrices in Table 9.5.1 to 9.5.4. in the Appendix. This answers the fourth research question: *"What are the differences in types of errors between startup categories?"*. Lastly, the best performing models and oversampling methods per startup category were tested on a dataset of startup companies in 2016-2019. The results can be seen in Table 5.5 and provide the answer to the fifth research question: *"How do the best models and imbalance methods per startup category perform on more recent data of 2016-2019?"*. The main question of this research: *"To what extent can startup status be predicted for different categories of startups, based on publicly available data"* and can be answered by looking at the average F1-scores per startup category. The highest achieved F1-score was in the Health Care sector getting 45.4%. These results are better than the baseline and that of similar research by Arroyo et al. (2019). It shows the difficulty there still is for machine learning models to handle highly imbalanced multiclass datasets. Future studies could look at incorporating more features. A recommendation is given to work with tree-based classifiers for this problem as they perform best in this study and current literature. Further hyperparameter tuning on the best results in this study could also increase performance. All in all, it shows that there is still a way to go if startup founders and venture capitalists want to make use of machine learning models to predict the future status of their startup.

References

Algamal, Z. Y., & Lee, M. H. (2015). High Dimensional Logistic Regression Model using Adjusted Elastic Net Penalty. *Pakistan Journal of Statistics and Operation Research*, *11*(4), 667. https://doi.org/10.18187/pjsor.v11i4.990

Amin, A., Anwar, S., Adnan, A., Nawaz, M., Howard, N., Qadir, J., Hawalah, A., & Hussain, A. (2016). Comparing Oversampling Techniques to Handle the Class Imbalance Problem: A Customer Churn Prediction Case Study. *IEEE Access*, *4*, 7940–7957. https://doi.org/10.1109/access.2016.2619719

Ang, Y.Q., Chia, A & Saghafian, S. (2020) *Using Machine Learning to Demystify Startups Funding, Post-Money Valuation, and Success* (HKS Working Paper No. RWP20-028). Retrieved from Harvard University website:https://scholar.harvard.edu/saghafian/publications/using-machine-learning-demystify-startups -fun ding-post-money-valuation-and

Arroyo, J., Corea, F., Jimenez-Diaz, G., & Recio-Garcia, J. A. (2019). Assessment of Machine Learning Performance for Decision Support in Venture Capital Investments. *IEEE Access*, *7*, 124233–124243. https://doi.org/10.1109/access.2019.2938659

Bajer, D., Zonc, B., Dudjak, M., & Martinovic, G. (2019). Performance Analysis of SMOTE-based Oversampling Techniques When Dealing with Data Imbalance. *2019 International Conference on Systems, Signals and Image Processing (IWSSIP)*, 265–271. https://doi.org/10.1109/iwssip.2019.8787306

Banerjee, P. (2020, August 20). *CatBoost Classifier in Python*. Kaggle. https://www.kaggle.com/prashant111/catboost-classifier-in-python

Barua, S., Islam, M. M., & Murase, K. (2011). A Novel Synthetic Minority Oversampling Technique for Imbalanced Data Set Learning. *Neural Information Processing*, 735–744. https://doi.org/10.1007/978-3-642-24958-7_85

Barua, S., Islam, M. M., & Murase, K. (2013). ProWSyn: Proximity Weighted Synthetic Oversampling Technique for Imbalanced Data Set Learning. *Advances in Knowledge Discovery and Data Mining*, 317–328. https://doi.org/10.1007/978-3-642-37456-2_27

Bayes, F. R. S. (1763). LII. An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London*, *53*, 370–418. https://doi.org/10.1098/rstl.1763.0053

Bej, S., Davtyan, N., Wolfien, M., Nassar, M., & Wolkenhauer, O. (2020). LoRAS: an oversampling approach for imbalanced datasets. *Machine Learning*, *110*(2), 279–301. https://doi.org/10.1007/s10994-020-05913-4

Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Processing with Python*. Van Duuren Media.

Blei DM, Ng AY, Jordan MI (2003) Latent Dirichlet Allocation. Journal of Machine Learning Research 3(Jan):993–1022

Böhning, D. (1992). Multinomial logistic regression algorithm. *Annals of the Institute of Statistical Mathematics*, *44*(1), 197–200. https://doi.org/10.1007/bf00048682

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/a:1010933404324

Broder, A. Z., Bruckstein, A. M., & Koplowitz, J. (1985). On the performance of edited nearest neighbor rules in high dimensions. *IEEE Transactions on Systems, Man, and Cybernetics*, *SMC-15*(1), 136–139. https://doi.org/10.1109/tsmc.1985.6313401

Brownlee, J. (2020, August 20). *10 Clustering Algorithms With Python*. Machine Learning Mastery.

https://machinelearningmastery.com/clustering-algorithms-with-python/

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, *16*, 321–357. https://doi.org/10.1613/jair.953

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794). New York, NY, USA: ACM. https://doi.org/10.1145/2939672.2939785

Chollet, F., & others. (2015). Keras. GitHub. Retrieved from https://github.com/fchollet/keras

Dietterich, T. (1995). Overfitting and undercomputing in machine learning. *ACM Computing Surveys*, *27*(3), 326–327. https://doi.org/10.1145/212094.212114

Dong, Y., & Wang, X. (2011). A New Over-Sampling Approach: Random-SMOTE for Learning from Imbalanced Data Sets. *Knowledge Science, Engineering and Management*, 343–352. https://doi.org/10.1007/978-3-642-25975-3_30

Dudjak, M., & Martinović, G. (2020). In-Depth Performance Analysis of SMOTE-Based Oversampling Algorithms in Binary Classification. *International Journal of Electrical and Computer Engineering Systems*, *11*(1), 13–23. https://doi.org/10.32985/ijeces.11.1.2

Ertekin, E. (2013). Adaptive Oversampling for Imbalanced Data Classification. *Information Sciences and Systems 2013*, 261–269. https://doi.org/10.1007/978-3-319-01604-7_26

Freund, Y., & Schapire, R. E. (1997). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, *55*(1), 119–139. https://doi.org/10.1006/jcss.1997.1504

Fushiki, T. (2009). Estimation of prediction error by using K-fold cross-validation. *Statistics and Computing*, *21*(2), 137–146. https://doi.org/10.1007/s11222-009-9153-8

Haibo He, Yang Bai, Garcia, E. A., & Shutao Li. (2008). ADASYN: Adaptive synthetic sampling approach for imbalanced learning. *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. Published. https://doi.org/10.1109/ijcnn.2008.4633969

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., . . . Oliphant, T. E. (2020, September). Array programming with NumPy. Nature, 585 (7825), 357–362. Retrieved from https://doi.org/10.1038/s41586-020-2649-2 doi: 10.1038/s41586-020-2649-2

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. Computing in Science & Engineering, 9 (3), 90–95. doi: 10.1109/MCSE.2007.55

Investopedia. (2020a, November 9). *How Many Startups Fail and Why?* https://www.investopedia.com/articles/personal-finance/040915/how-many-startups-fail-and-why.asp

Investopedia. (2020b, March 5). *Series A, B, C Funding: How It Works*. https://www.investopedia.com/articles/personal-finance/102015/series-b-c-funding-what-it-all-means-and-how-it-works.asp

Investopedia. (2021a, March 30). *What Is a Unicorn in Business?* https://www.investopedia.com/terms/u/unicorn.asp

Investopedia. (2021b, March 26). *Sector Breakdown*. https://www.investopedia.com/terms/s/sector-breakdown.asp

Kong, J., Rios, T., Kowalczyk, W., Menzel, S., & Bäck, T. (2020). On the Performance of Oversampling Techniques for Class Imbalance Problems. *Advances in Knowledge Discovery and Data Mining*, 84–96. https://doi.org/10.1007/978-3-030-47436-2_7

Kovács, G. (2019). An empirical comparison and evaluation of minority oversampling techniques on a large number of imbalanced datasets. *Applied Soft Computing*, *83*, 105662. https://doi.org/10.1016/j.asoc.2019.105662

Krishna, A., Agrawal, A., & Choudhary, A. (2016, December). Predicting the Outcome of Startups: Less Failure, More Success. *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*. https://doi.org/10.1109/icdmw.2016.0118

Krogh, A. (2008). What are artificial neural networks? *Nature Biotechnology*, *26*(2), 195–197. https://doi.org/10.1038/nbt1386

Kumar, K. (2018, May 3). *Evaluation of Topic Modeling: Topic Coherence*. DataScience+. https://datascienceplus.com/evaluation-of-topic-modeling-topic-coherence/

Le, T., Lee, M., Park, J., & Baik, S. (2018). Oversampling Techniques for Bankruptcy Prediction: Novel Features from a Transaction Dataset. *Symmetry*, *10*(4), 79. https://doi.org/10.3390/sym10040079

Lemaître, G., Nogueira, F., & Aridas, C. K. (2017). Imbalanced-learn: a python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, *18*(1).

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, *1*(14), 281–297.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, *5*(4), 115–133. https://doi.org/10.1007/bf02478259

McKinney, W. (2010). Data Structures for Statistical Computing in Python. In Stefan van der Walt & Jarrod Millman (Eds.), Proceedings of the 9th Python in Science Conference (p. 56 - 61). doi: 10.25080/Majora-92bf1922-00a

Morde, V. (2019, April 15). XGBoost Algorithm: Long May She Reign! - Towards Data Science. Medium. https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d

Nitze, I., Schulthess, U., & Asche, H. (2012). Comparison of machine learning algorithms random forest, artificial neural network and support vector machine to maximum likelihood for supervised crop type classification. Proc. of the 4th GEOBIA, 35.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12 , 2825–2830.

Polikar, R. (2006). Ensemble based systems in decision making. IEEE Circuits and Systems Magazine, 6(3), 21–45. https://doi.org/10.1109/mcas.2006.1688199

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). Catboost: unbiased boosting with categorical features. *Advances in Neural Information Processing Systems 31*, 6638–6648.

Rajesh, K. N., & Dhuli, R. (2018). Classification of imbalanced ECG beats using re-sampling techniques and AdaBoost ensemble classifier. *Biomedical Signal Processing and Control*, *41*, 242–254. https://doi.org/10.1016/j.bspc.2017.12.004

Rehurek, R., & Sojka, P. (2011). Gensim–python framework for vector space modelling. *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic*, *3*(2).

Republic. (2020, September 30). *How do I pick startups to invest in? Tips from 10 top VC and angel investors*. https://republic.co/blog/investor-education/how-do-i-pick-startups-to-invest-in

Ross, G., Sciro, D., Ranjan, S.D., Raza, H. (2020). *CapitalVX: A Machine Learning Model for Startup Selection and Exit Prediction.* (Working Paper). Retrieved from SSRN website: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3684185

Scikit-learn. (n.d.). *sklearn.impute.IterativeImputer — scikit-learn 0.24.2 documentation*. https://scikit-learn.org/stable/modules/generated/sklearn.impute.IterativeImputer.html

Scikit-learn. (n.d.). *Selecting the number of clusters with silhouette analysis on KMeans clustering — scikit-learn 0.24.2 documentation*. https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html

Sechidis, K., Tsoumakas, G., & Vlahavas, I. (2011). On the Stratification of Multi-label Data. *Machine Learning and Knowledge Discovery in Databases*, 145–158. https://doi.org/10.1007/978-3-642-23808-6_10

Seggura, T. (2020, October 9). *"CatBoost Algorithm" explained in 200 words.* Data Science. https://thaddeus-segura.com/catboost/

Sharchilev, B., Roizner, M., Rumyantsev, A., Ozornin, D., Serdyukov, P., & de Rijke, M. (2018). Web-based Startup Success Prediction. *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2283–2291. https://doi.org/10.1145/3269206.3272011

Sharma, S. (2020, October 11). *Dutch startups are the best job growth engine in the Netherlands; stayed strong during COVID-19 | Silicon*. Silicon Canals. https://siliconcanals.com/news/dutch-startups-are-the-best-job-growth-engine/

Sharma, P. (2020, October 18). *The Most Comprehensive Guide to K-Means Clustering You'll Ever Need*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/

Startup Genome. (2020). *State of the Global Startup Economy 2020*. https://startupgenome.com/article/state-of-the-global-startup-economy

Tanha, J., Abdi, Y., Samadi, N., Razzaghi, N., & Asadpour, M. (2020). Boosting methods for multi-class imbalanced data classification: an experimental review. *Journal of Big Data*, *7*(1). https://doi.org/10.1186/s40537-020-00349-y

Ünal, C. (2019). *Searching for a Unicorn: A Machine Learning Approach Towards Startup Success Prediction* (Thesis). https://doi.org/10.18452/20347

Van Rossum, G., & Drake, F. L. (2009). *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace.

Zhang, Z. (2020, February 6). Naive Bayes Explained - Towards Data Science. Medium. https://towardsdatascience.com/naive-bayes-explained-9d2b96f4a9c0

Appendices and Supplementary Materials

Table 9.1. All the features in the dataset after preprocessing and cleaning.

Feature	Data type	Description
status	categorical	The status of a startup (operating, IPO, closed or acquired)
num_funding_rounds	integer	The number of funding rounds
total_funding_usd	float	The total funding in USD
year_founded_on	integer	The year the startup was founded on
company_age	integer	Difference between 2016 and year_founded_on
last_funding_in_years	integer	Difference between 2016 and year_last_funding_on
has_facebook	integer	Binary value indicating if a startup has Facebook or not
has_linkedin	integer	Binary value indicating if a startup has LinkedIn or not
has_twitter	integer	Binary value indicating if a startup has Twitter or not
has_email	integer	Binary value indicating if a startup has email or not
has_phone	integer	Binary value indicating if a startup has a phone number or not
has_domain	integer	Binary value indicating if a startup has a domain or not
description_length	integer	The number of characters in the short_description
last_round_raised_amount_usd	float	The amount of USD raised in the last round

last_round_post_money_valuatino_USD	float	The valuation of a startup after the last funding round in USD
last_round_investor_count	integer	The total amount of investors in the last round
investor_count	integer	The total amount of investors
year_announced_on	integer	Year of first funding
first_funding_lag	integer	Difference between year_founded_on and year_announced_on
year_last_funding_on	integer	Year of last funding
last_funding_lag	integer	Difference between year_announced_on and year_last_funding_on
founders_count	integer	The amount of founders
founder_degree_count	integer	The total amount of degrees earned by founders
founder_male_count	integer	The total amount of male founders
founder_female_count	integer	The total amount of female founders
continent	categorical	This variable is dummy encoded. Features are Asia, Australia, Europe, North-America and South-America
employee_count	categorical	This variable is dummy encoded. Features are: 1-10, 11-50, 51-100, 101-250, 251-500, 5-1-1000, 1—1-5000, 5000+ and unknown.
last_round_investment_type	categorical	This variable is dummy encoded. Features are angel, debt, grant, other, private equity, seed funding, series A, series B, series C, series D, series E, series F, series G and series unknown.

Table 9.2. Similarities and differences of the resulting clusters.

Cluster	K-Means	Latent Dirichlet Allocation
1	Educational	Educational
2	Manufacturing	Video
3	Information technology	Information Technology
4	Health Care	Health Care
5	Consumer Discretionary	Gaming and Sports
6	Mobile Apps	Mobile Apps
7	Finance	Finance
8	Cloud Computing	Digital Media
9	E-commerce	E-commerce
10	Software	Software
11	Artificial Intelligence	Artificial Intelligence
12	No clear category	Security
13		No clear category
14		No clear category
15		No clear category

Table 9.3. Baseline results of the Random Forest model including average precision & recall scores.

Startup Category	Average Precision	Average F1-Score	Average Recall	Accuracy
Information Technology	0.423	0.305	0.302	0.785
Health Care	0.637	0.404	0.379	0.786
Consumer Discretionary	0.520	0.299	0.302	0.805
Finance	0.474	0.265	0.272	0.801

Table 9.4. Average precision, F1-score, recall and accuracy per startup category

Startup Category	Model	Method	Average Precision	Average F1-Score	Average Recall	Accuracy
Information Technology	CatBoost	SMOTE-ENN	0.400	0.392	0.412	0.721
Health Care	Random Forest	ProWSyn	0.445	0.454	0.444	0.779
Consumer Discretionary	XGBoost	ProWSyn	0.487	0.407	0.393	0.790
Finance	AdaBoost	ProWSyn	0.373	0.385	0.438	0.664

Table 9.5.1. Confusion Matrix Information Technology Sector

		Predicted			
		Acquired	Closed	IPO	Operating
Actual	Acquired	**50**	10	4	67
	Closed	11	**18**	0	40
	IPO	4	0	**17**	25

Operating	89	52	22	**746**

Table 9.5.2. Confusion Matrix Health Care Sector

		Predicted			
		Acquired	Closed	IPO	Operating
Actual	Acquired	**58**	4	17	121
	Closed	13	**21**	1	76
	IPO	6	0	**51**	45
	Operating	47	29	24	**1232**

Table 9.5.3. Confusion Matrix Consumer Discretionary Sector

		Predicted			
		Acquired	Closed	IPO	Operating
Actual	Acquired	**13**	1	1	53
	Closed	4	**10**	0	35
	IPO	1	0	**6**	21
	Operating	23	9	5	**566**

Table 9.5.4. Confusion Matrix Finance Sector

		Predicted			
		Acquired	Closed	IPO	Operating
Actual	Acquired	**16**	4	5	27
	Closed	3	**10**	1	10
	IPO	5	0	**10**	10
	Operating	51	35	26	**319**