Machine Learning Engineer Nanodegree
Capstone Project Report

Startup's acquisition forecasting

Fernando Leandro dos Santos

July 4, 2017

1 Definition

Project Overview

Startup companies are recently emerging and evolving more than ever. But still, many of them fail, in fact, 9 out of 10 startups fail [1]. So, what are the characteristics of startups that succeed? There are many factors that might play a role in this question: founders and team experience, fundings, location, product, market and so on. Why some startups are acquired very early and others are not? Are there maybe some factors that attract bigger companies into some startup acquisition? This study aims to dig into these questions by investigating a dataset of startup companies and acquisitions provided by Crunchbase[2], a company that provides business information with data about companies and investments all over the world.

Previous researchers have tried to predict startups success from different types of data. Some have tried to predict startup acquisitions based on similar data (Crunchbase) together with TechCrunch[3] news about the companies[3]. Others have tried to predict startups survival time based on a venture screening questionnaire[2] as well as predicting success from business model documents[1]. The final product of this project will be a classification model that tries to predict, based on investments data, whether a startup company would be acquired or not, which we consider as a sign of success.

Problem Statement

The startup world requires a lot of decision making from the people involved, specially when dealing with situations of funding or acquisitions that will impact in ownership changes of the company. In these situations, the decision maker should have on hand all information that can be provided and with the amount of data available today about companies, funding rounds and acquisitions, it could be very helpful to have statistical models helping and optimizing this decision. This study proposes a classification model that tries to predict if any operating company has more chances to be acquired (indicating success) or to be closed (indicating failure) in a near future.

This study proposes a supervised learning approach to create a model that might be able to forecast a startup success as well as a data exploration approach to better understand the variables that most impact on the acquisition or failure of a startup company. Different supervised learning algorithms will be tested in order to find the most appropriate and the one with better performance according to the evaluation metrics defined below.

At first, the data will be explored and better understood. A PCA algorithm will be also used to better understand the variance expressed in the features. This analysis give us a good intuition to realize which

[1] http://fortune.com/2014/09/25/why-startups-fail-according-to-their-founders/
[2] https://www.crunchbase.com/
[3] https://techcrunch.com/

features are more expressive. Once the data is well explored and the best features are selected, Support Vector Machine and K-Nearest Neighbor algorithms are gonna be modelled. Since we'll have a relatively large amount of samples (more than 30.000), applying SVM will not be a problem. These algorithms were chosen due to their extra flexibility by testing different Kernels (for SVM) as well as different distance functions (for k-NN). Also, those are from two different family of algorithms, kernel methods and instance based, so they are covering a good range of approaches to deal with our classification problem. As a final attempt for a better performance, we will also model two ensemble learning algorithms: Random Forest and Gradient Boosting, implemented with XGBoost, which is a framework that has been recently increasing popularity among applied machine learning and data science competitions.

Metrics

For measuring the performance of our classification models, we'll use three simple metrics: True Positive rate, False Positive rate and the area under the ROC curve (AUC). True Positive and False Positive rates are fairly simple metrics, directly derived from a classification confusion matrix of a binary classifier:

- **True Positive (TP)**: Rate of instances that were classified with a positive label and are positive
- **False Positive (FP)**: Rate of instances that were classified with a positive label and are **not** positive
- **True Negative (TN)**: Rate of instances that were classified with a negative label and are negative
- **False Negative (FN)**: Rate of instances that were classified with a negative label and are **not** negative

These two metrics (TP and FP) are specially useful in this problem because we are dealing with an imbalanced dataset. The amount of companies that are acquired is, in general, much smaller than the amount of companies currently in operation or closed. Therefore, we need metrics that directly express the precision and the recall of our model. If we were using, for example, a simple accuracy, our models would probably have a high score even though when performing very poorly on the task of retrieving the few companies that are more likely to be acquired.

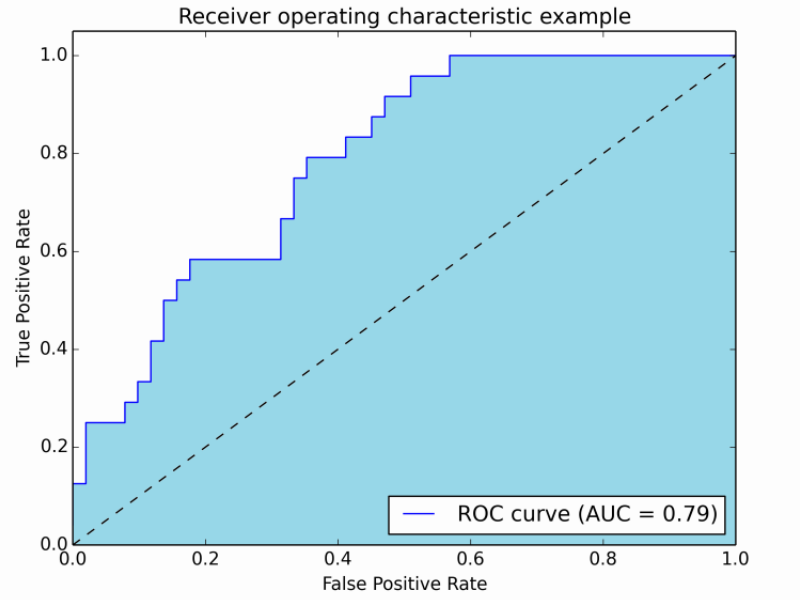

Figure 1: Area under the ROC curve in blue

From these concepts, we can also explain the intuition behind the AUC metric. The ROC curve is basically computed by combining TP rate with TN rate in such a way that by plotting the TPR (in the Y axis) and FPR (in the X axis), we'll have a curve as stated in Figure 1. The blue area corresponds to the area under the ROC curve while the dashed line represents the ROC curve of a random predictor, which serves as a baseline to

see whether the model is useful. The AUC metric is a specially good metric for binary classification problems, which is the case of this project.

2 Analysis

Data Exploration and Visualization

The data for this project was gathered from Peter Tripp's GitHub repository[4]. The dataset in this repository was already extracted from Crunchbase dump file into CSV files and dates from December 4, 2015. The dataset contains information of 66.368 companies from many countries, but only USA based companies will be considered for this study, which will decrease the amount of companies to 37.598 (this number might change after pre-processing and cleansing of the data).

The initial dataset obtained for this project consists of 4 tabular files:

- **companies.csv**: Contains general information about each company, like name, line of business, total amount of funding received, location of the company, founding date, etc..
- **acquisitions.csv**: Contains information about acquisitions in a format of: which company acquired which other company. Includes the companies information as well the amount paid for the acquired company
- **investments.csv**: Contains information of investments made by companies in a format of: which company invested in which other company. Includes information about each investment like which was the round of investment, the investment type, the date of the investment as well as the amount invested.
- **rounds.csv**: Contains similar information from *investments.csv*, but in this file the investments information is grouped by rounds. While *investments.csv* provides information about where the money came from and which investors paid each round, *rounds.csv* is grouping and totalizing all this information by rounds of investment.

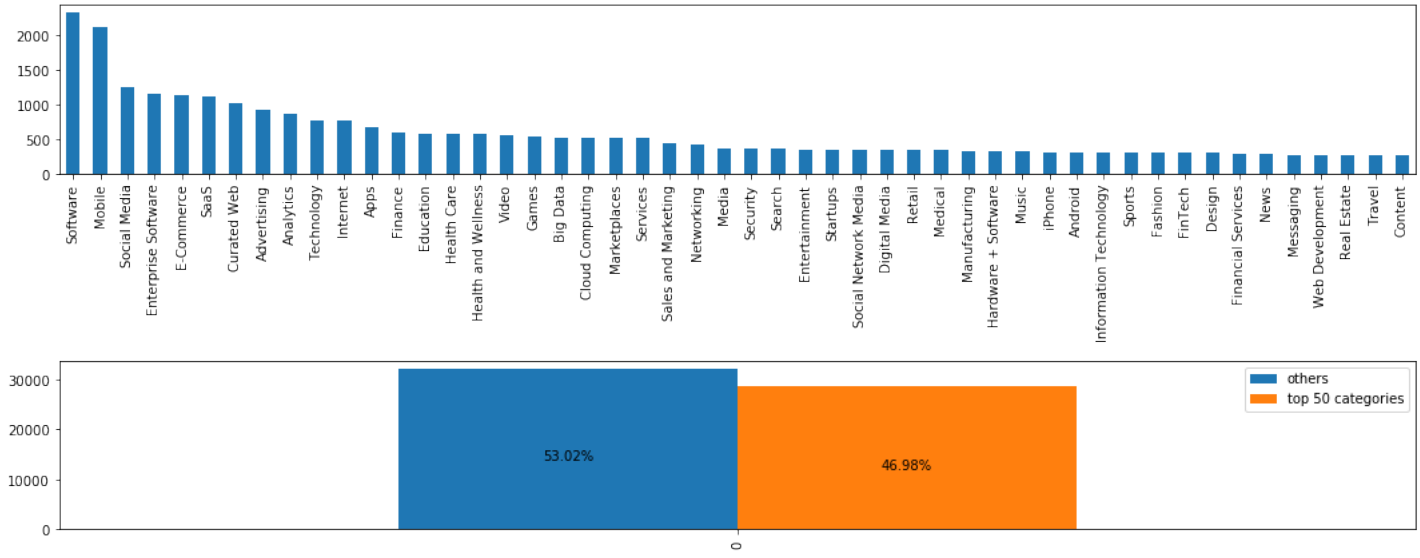

Figure 2: Top 50 categories of companies selected

The dataset used in this analysis will be a composition of the information present in all these 4 files. It is worth mentioning that the dataset is composed of companies all around the world, but for consistency, we are selecting only companies based in USA since companies from other countries have a higher frequency of missing data. The main file is *companies.csv*, where most of the information lies. From there, we'll ignore some columns like name, website url and even very detailed location like city and region (we'll use only the information about

[4] `https://github.com/notpeter/crunchbase-data`

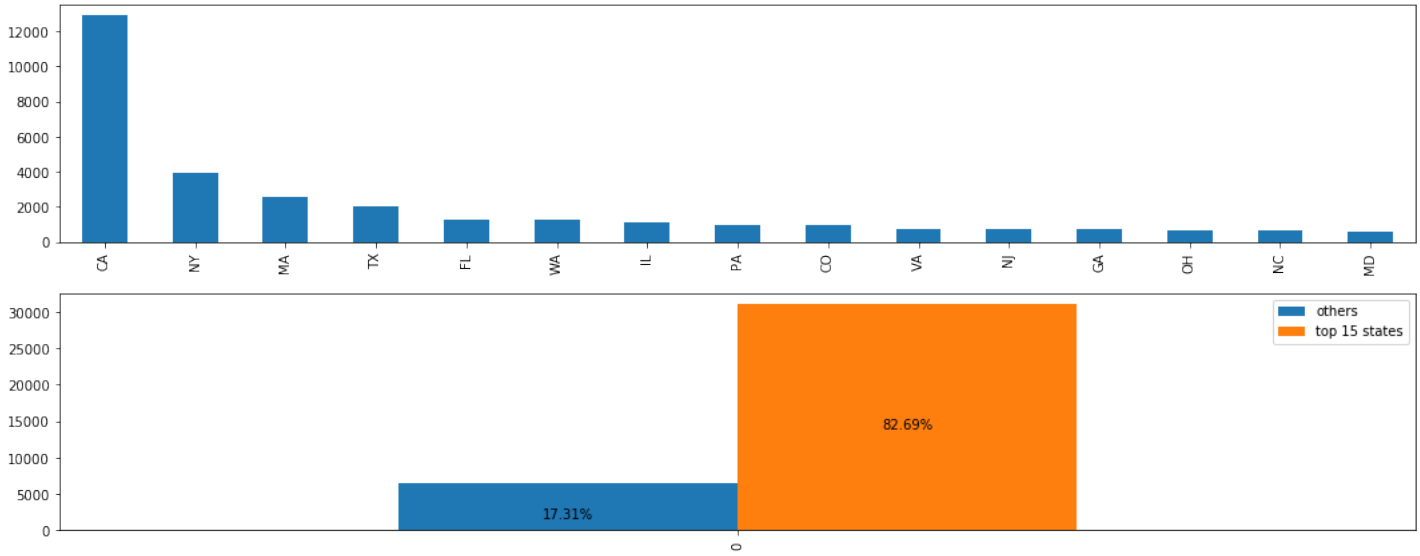

Figure 3: Top 15 states selected

the USA state of the company). This is necessary because these are categorical values with many possible values that don't carry much useful information for the prediction. Other categorical variables like the state and the line of business of the company will later be pre-processed to be transformed into single dummy features. In the same file we also have some date columns, about the date of first and last funding as well as founding date. These features will need to be converted into age. Finally, we have the amount of investment rounds and the total amount of investment received by the company, which are already numerical and very important features. Our target variable, status, is also present in this file and it contains 4 possible values: *operating*, *acquired*, *closed* and *IPO*, which will later be converted to *acquired* and *not acquired*.

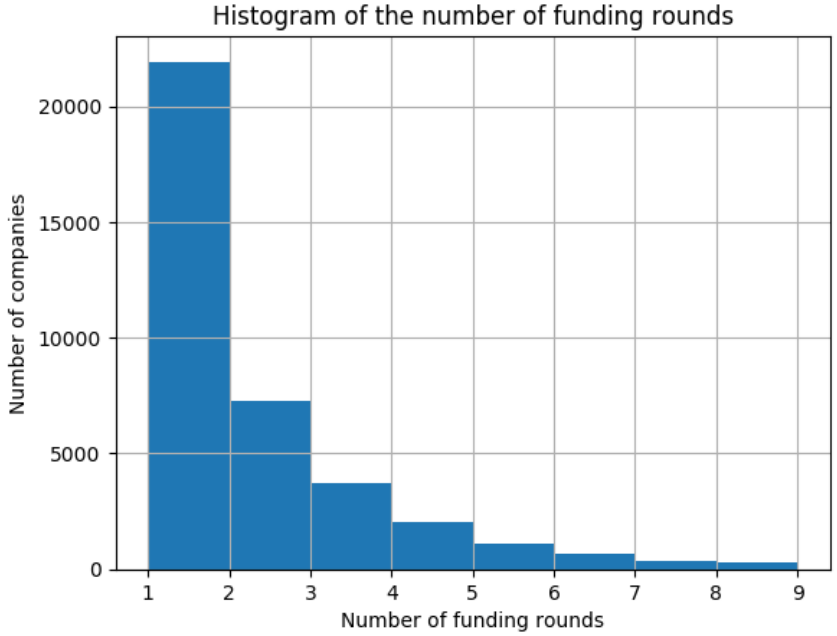

Figure 4: Number of companies by the number of funding rounds

From the other files, some aggregation of values will be necessary to extract more features. From *acquisi-*

tions.csv we'll extract the number of acquisitions that were made by each company. From *investments.csv* we'll extract the number of investments made by each company, the average number of investors per round and the average amount invested in each round. Finally, from *rounds.csv* we'll extract the number of funding rounds and the total amount invested by each type of investment (seed, venture, angel, private equity, crowdfunding, etc..).

The final aggregated dataset contains a total 37601 companies with 104 features each. The number of features is quite high because of the two categorical variables. For both of them, we applied a rule to select the minimum number of categories that would cover enough amount of companies in the dataset. Therefore, we selected the most frequent 50 (out of 60813) lines of business (which covered 46% of the companies) and the most frequent 15 states (which covered 82% of the companies), as presented in figures 2 and 3.

From figure 5 we see a ranking of the companies that had the highest amount of funding in the period of our data, where we can see many of the top companies of the moment like Uber, Facebook, Airbnb, Netflix, etc. Also, from figure 4 we see another interesting chart describing the frequency of companies by the amount of funding rounds. We see that most of the companies had 1 or 2 funding rounds only.

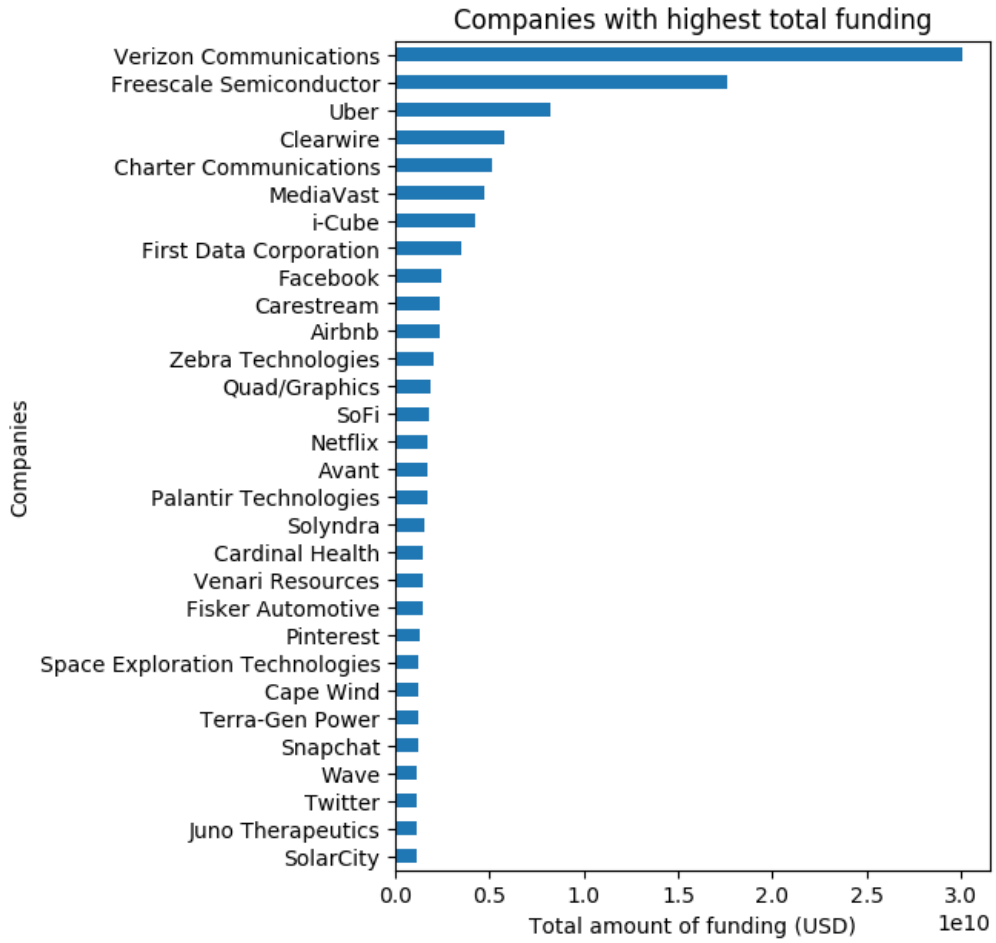

Figure 5: Companies with highest total funding

This dataset has a particular characteristic that needs special attention, which is the imbalance of the class distributions. This is a common problem where the classes are not represented equally. It can happen for multiclasses as well as for a binary classification problem, which is the case in this project.

In figure 6, we see the difference in our dataset between the number of acquired startups and not acquired startups. There are a few ways to tackle this problem, like subsampling the majority class, oversampling the minority class with synthetic instances, trying different algorithms, penalized models and so on. In this project, we tried to use a subsample of the dataset and also tried to adjust our models with extra penalties for the

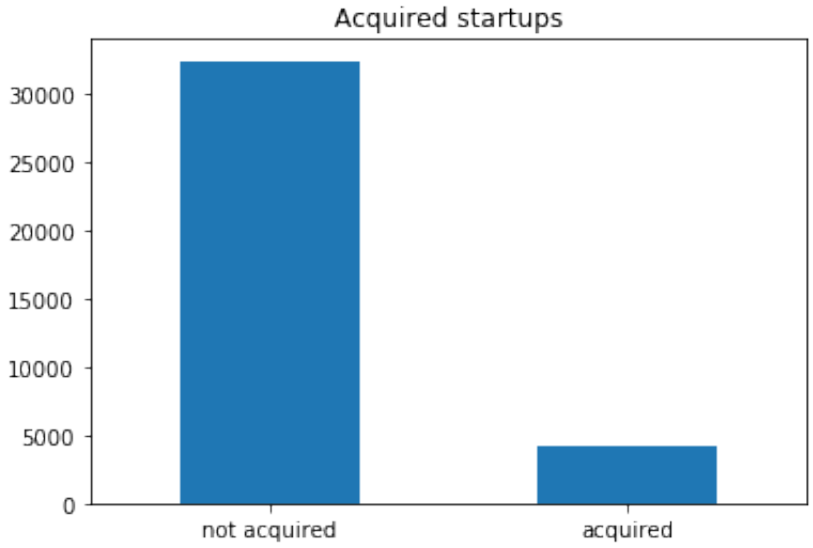

Figure 6: Class distribution over the startups dataset

minority class.

Algorithms and Techniques

As described in the Problem Statement section, we explore a few different algorithms during this project. Besides the use of the classification algorithms Random Forest, SVM and k-NN, we also use PCA, which is a technique for dimensionality reduction. Basically, in PCA, we want to create new (and less) features from the original features. To do that, it makes use of a covariance matrix and Eigen decomposition.

The Random Forest is an ensemble algorithm applied to Decision Trees. Basically, it constructs many separate decision trees and combines them to produce an aggregate model that is more powerful than any of its decision trees alone. Just a few parameters are usually required for this model, such as the max depth of the decision trees and the size of the forest (number of trees). These parameters are very important in this algorithm, specially to avoid overfitting.

Another algorithm used is the SVM. This algorithm works by defining a hyperplane that divides the data correctly in its two classes and at the same time, this hyperplane has the maximum distance from both sides. In other words, the algorithm tries to maximize the margin (the distance between the main decision boundary and the closest points of that side) while still classifying correctly all the input data. Because of the algorithm is only interested in the data points close to the hyperplane, those are the data points that matter, called the support vectors. Figure 7 illustrates the concepts of margin and support vectors.

For this algorithm to work, it is important that the data can be linearly separable in two planes. So, for cases where the data is initially not separable, a kernel trick can be applied, which is basically a mathematical transformation conveniently applied to every input data, so that the data becomes linearly separable in this higher dimensional plane. Besides the kernel, which is a very important parameter, we have the parameter C, the penalty for errors, which is responsible for avoid overfitting.

We also applied the k-NN algorithm, a non-parametric method for classification. This algorithm is self explained by its name. In short, it classifies an instance according to its k nearest neighbors in the feature space. As illustrated in 8, an instance is classified by majority of classes from the k nearest instances. This is quite a simple model and its only parameter is the k, the number of neighbors to consider.

As an extra implementation effort, we also applied another model with XGBoost, which is a recent and very popular machine learning library [5]. Basically, it implements prediction models in the form of an ensenble of weak classifiers, typically decision trees. In some way, it is similar to a random forest algorithm, but it generalizes the model with an optimization of an arbitrary differentiable loss function. It uses an iterative process to find

[5] `https://github.com/dmlc/xgboost`

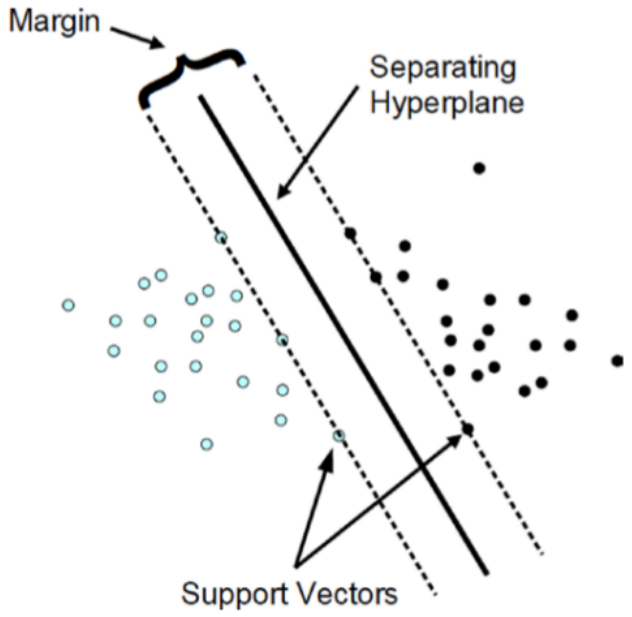

Figure 7: Margins and support vectors of a SVM

the local minimum of the specified loss function and thus, find an ensenble trees model that best minimizes the validation error.

Finally, due to the imbalance of the dataset used in the project, a technique of subsample was applied as well. This is one simple solution for imbalanced datasets where samples of the majority class are removed until an equal distribution of classes is reached. We tested this subsampled dataset with a k-NN model, specially because it is an algorithm very sensible to imbalanced datasets.

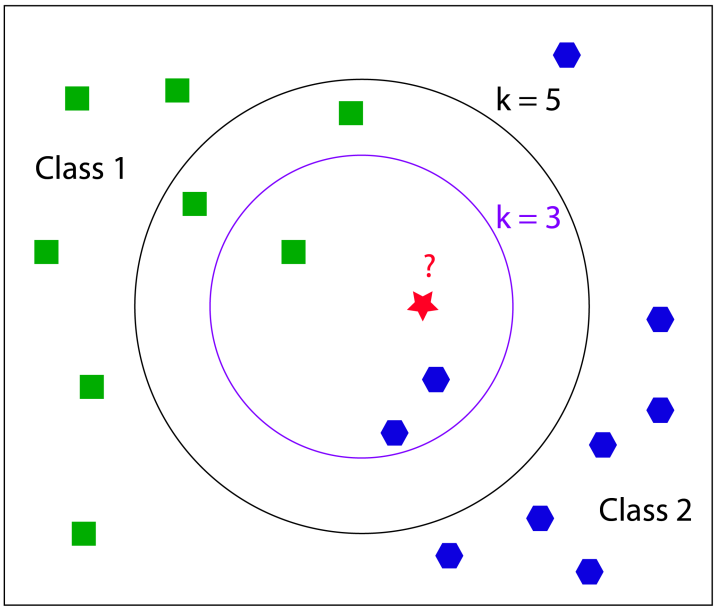

Figure 8: K-Nearest Neighbors illustration

Benchmark

The model proposed by Xiang et al.[3] will be used as a comparison base. They created a model using Crunchbase data as well, but from a different period of time. They measured their model performance using a True Positive rate, False Positive rate and the area under the ROC curve (AUC). It is worth noting that they included in their model data from TechCrunch news articles about the companies, which they concluded to have improved the overall performance of the model. They also used some basic information about the companies that we did not have access in this project, such as number of employees, number of offices and some more basic company features. In this project, the same metrics will be used to evaluate the model.

3 Methodology

Data Pre-processing

After all the new features extraction, described in the Data Exploration section, a few other pre-processing tasks were applied. Firstly, numeric variables were normalized by the distance between maximum and minimum values of the feature. Some useless features were removed, such as company name or company homepage url. Date variables were converted to an age in months. During this phase, for simplicity and for avoiding bias in the model, companies that went though an IPO were removed. Finally, the categorical variables *state* and *categories of companies* were extracted into dummy variables, establishing a final normalized dataset of 103 features and a target variable consisting of *acquired* and *not acquired* labels.

During this phase, a different dataset was generated by applying a PCA transformation to the final pre-processed dataset and selecting the first 6 main components, which are, 6 features that represents most of the variance present in the original 103 features.

Implementation

The implementation of the models were quite simple thanks to scikit-learn, a python library for machine learning. Since all the algorithms used in this project follow the same classification interface, which is fitting the train set and then predicting in a validation set, a simple helper function was defined to generate models for different algorithms.

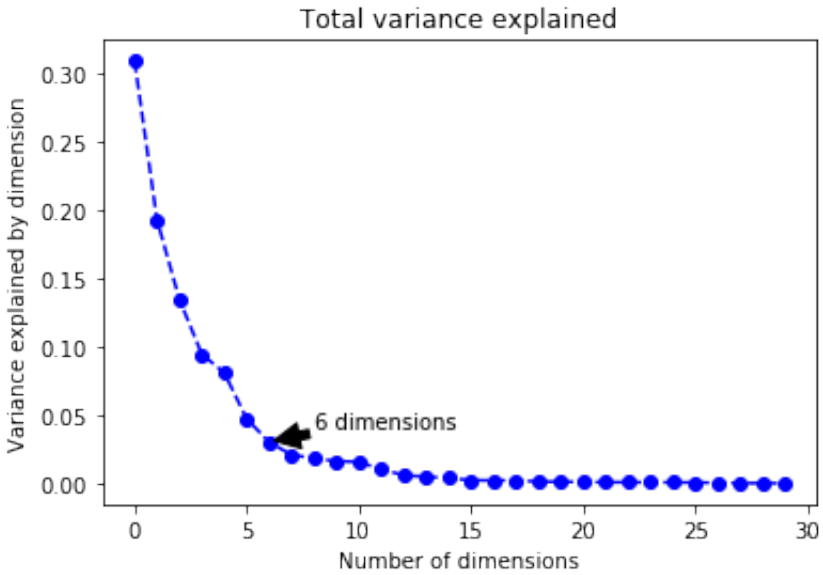

Figure 9: Variance in the data explained by each principal component

First of all, the dataset was split into train (80%) and test set (20%). The latter was never used for training the data, it was used only in the end of the process for measuring the final performance of the models. Meanwhile,

the train set was split again, leaving 20% of it as a validation set for parameter tuning. Then, every model was trained with the remaining 60% of the train data, using a GridSearch approach, for testing different parameters with a 5-fold cross validation. This same training process was repeated for Random Forest, SVM and k-NN.

The dataset of 6 features, prepared with PCA transformation was used for generating a SVM model. The choice of 6 features was made from the analysis of the amount of variance explained by each dimension. Figure 9 shows that the variance explained is quite low after the 6th dimension. For each dimension generated, we could also visualize the ratio variance explained by each feature. To better visualize this idea, figure 10 shows a PCA transformation using only the numerical variables of our dataset. We see that the first dimension explains around 30 of the variance of the data. Most part of this percentage is explained by funding_rounds and venture_funding_rounds variables. For the second dimension, which explains 19 of the variance in the data, the most expressing features are last_funding_at and seed_funding_rounds.

Also, a subsampled dataset was used to train another model of k-NN. This was necessary because k-NN is an algorithm very sensible to imbalanced datasets, which is the case for this project. It is worth mentioning that an extra parameter was applied for Random Forest and SVM, which is the *class_weight* set to *balanced*. This is, again, a necessary parameter for imbalanced data. It forces the model to apply different weights for each class based on the frequency of the classes in the training set.

Finally, once all the models were trained with the best parametrization possible, they were tested against the final test set and had their metrics saved and compared. The metrics used in this phase and also for tuning the models were TP rate, FP rate and AUC score.

It is worth noting that not all of these processes were directly thought and applied at first. It was an iterative process aiming to reach the best performance for the models. For instance, some of the investment features extracted from other files were, in fact, extracted in a second moment, since the features available so far weren't good enough. Also, the refinement for hyperparameters were developed after exhaustingly testing different sets of parameters, with the use of GridSearchCV.

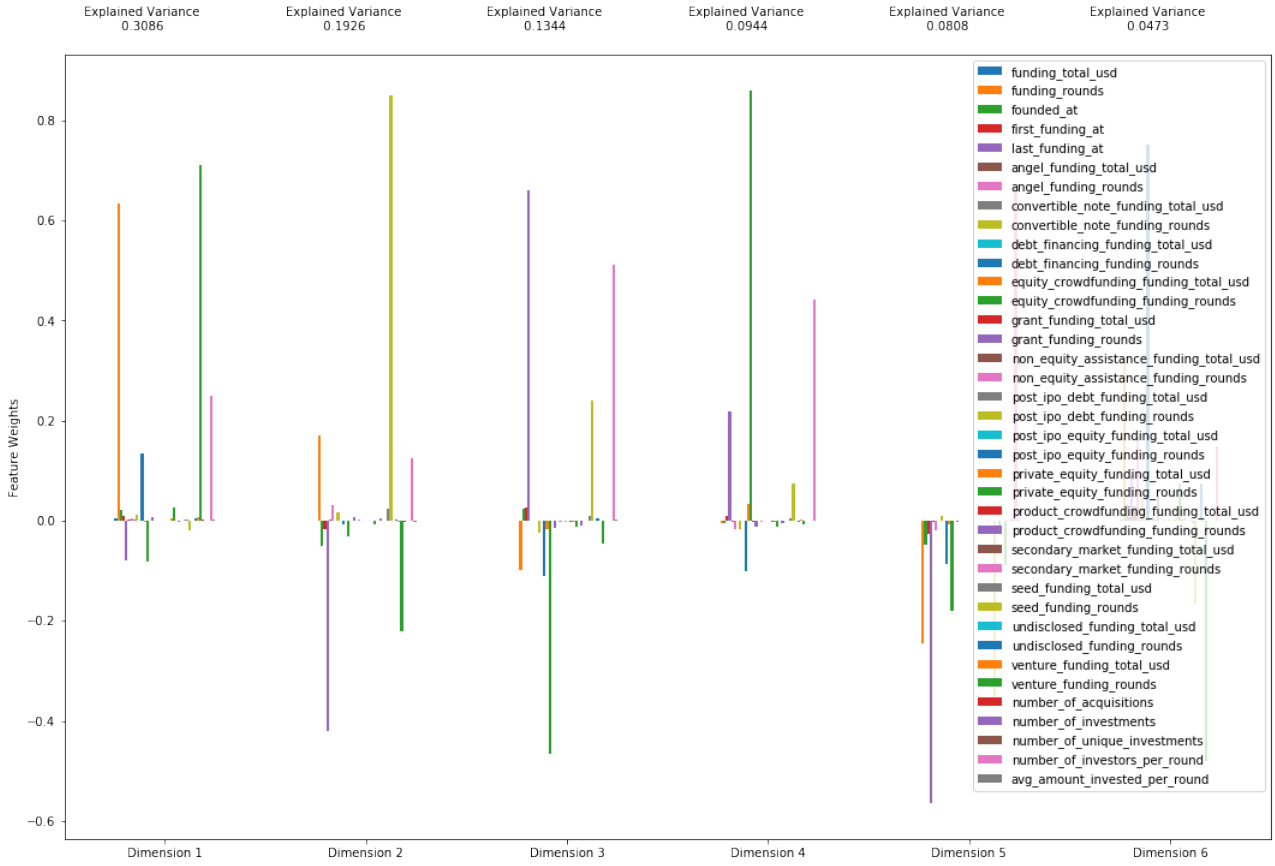

Figure 10: Variance ratio in the features for each dimension

4 Results

Model Evaluation and Justification

Once all the models were trained and tested with train and validation set, we selected the best model of each algorithm and applied to the final unseen test data. By doing this, we guarantee that the model has never seen that data before and thus, can generalize and still express reasonable results for any new data. From table 1 we see a slightly better performance for the Random Forest model.

Our best model turned out to be very simple. A Random Forest implementation with 50 ensemble trees with max depth of 10 degrees. Despite its simplicity, it was good enough to understand the data behavior and generalize relatively well to unseen data. It achieves a high TP rate, but ends up with a relatively high FP rate. The model obtained is robust enough for the given dataset, but more data would be necessary for a better prediction. A FP rate of 18.8% is too high for a real world scenario, where we need very good recall, in order to avoid suggestions of company purchases that are not worth.

Although the final models obtained can not yet be trusted for a real application, it reached our initial expectations, as it could definitely understand and identify value in a company just by looking at its investments information.

Model	**TP rate**	**FP rate**	**AUC**
Random Forest	**73.4%**	**18.8%**	**0.773**
SVM	74.8%	23.5%	0.757
k-NN	15.6%	2.9%	0.564
k-NN Subsampled	65.8%	32.8%	0.665
XGBoost	67.4%	15.1%	0.761

Table 1: Performance scores obtained in the final unseen test set

Our benchmark model, proposed by Xiang et al.[3] has a good performance, achieving a high TP (from 60% to 80%) with acceptable FP. However, these results were achieved with the use of important data about the company that we did not have access to. Moreover, they divided their data, creating separate classification models for each category of company while in this project, categories of companies were the features themselves. Nevertheless, we can still use their results as a benchmark and compare their results with ours. In table 2, we show their best classifier performance scores for a few categories. We see that they achieved a good TP rate while maintaining the FP rate very low, which is important in this scenario, since we do not want to predict that a company will be acquired if it will not.

Model	**TP rate**	**FP rate**	**AUC**
Bayesian Networks (Computer related)	59.9%	2.2%	0.882
Bayesian Networks (Hardware related)	56.5%	0.0%	0.857
Bayesian Networks (Other)	79.8%	0.0%	0.942

Table 2: Our best model with some models obtained from the benchmark comparison

In short, we can confirm that our model did not outperformed the benchmark model and the reasons for that lies basically in the data used for the analysis. The benchmark model used more data about companies and applied natural language processing techniques over articles about that companies. However, from the AUC score achieved in this project (around 77%), we can definitely say that our model were able to identify patterns that helps to classify whether companies will be acquired or not. The proposed solution is not yet significant enough to completely solve the problem, but it indicates a good direction on where to head next.

5 Conclusion

Free Form Visualization

As explained in the data exploration section, this project had a particular characteristic, which is the imbalance of the class distributions. Figure 11 illustrates this behavior and demonstrates the results of the Random Forest classifier in the test dataset trhough a confusion matrix. When classifying 7299 companies, the model misclassified 1219 companies as they would be acquired but they were not. Similarly, it classified 610 companies as they would be acquired and they were, in fact, acquired. The results obtained in this project varied around this trade-off. Some models increased the accuracy of acquisitions suggestions, but at the same time, increased the missclassification of companies that should not be tagged positively.

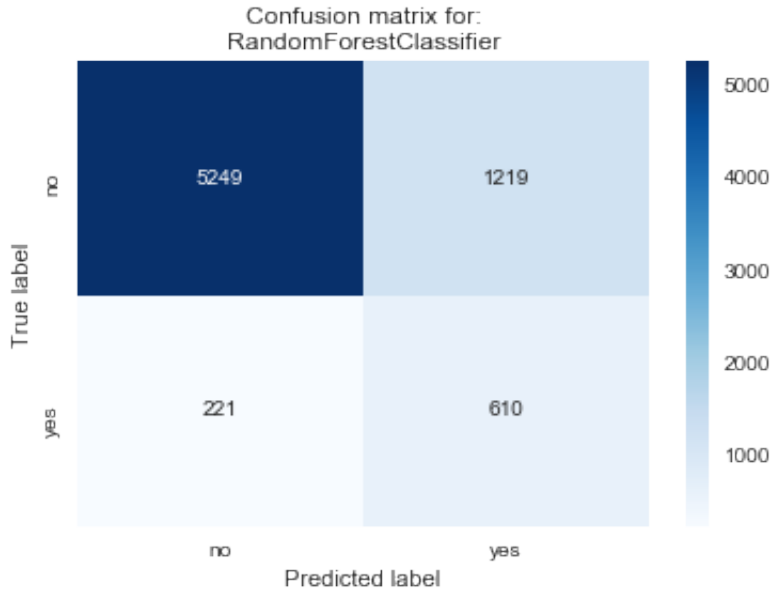

Figure 11: Random Forest classification in the final test dataset

In a real world scenario, we would want to minimize as much as possible the false-positive quadrant of this matrix. This error is more sensible than false-negatives, since whenever the model suggests the acquisition of a company, we would like this company to be acquired.

Reflection

In this project, a dataset of startups and investments was gathered from an online repository and transformed into a dataset of features focused in the prediction of startup acquisition. Features from the initial dataset were transformed and then normalized. New features were created and normalized by aggregating different files from the initial dataset. This dataset was cleaned and had some instances removed in order to simplify the analysis. Once the dataset was ready, some techniques for exploring this features were applied, like PCA, and finally different classification models were created upon this data. Models of SVM, Random Forest and k-NN were repeatedly created and tested with different parameters in order to find the model that best generalizes to an unseen validation set. Finally, the best model found was tested against a final unseen test set and that was the final and best performance score obtained in the project. True Positive rate, False Positive rate and Area Under the ROC curve were the metrics used during the project to estimate the classifiers performance.

As described in the previous section, the imbalance of the dataset brought some issues that needed to be addressed. Feature extraction was a tricky part of the project as well, since data needed to be aggregated and merged from different files, which required some knowledge of data wrangling in Python.

As a conclusion, the final model obtained does not yet solve the problem of forecasting startups acquisition and could not yet been used in a real world scenario to help decision makers when to acquire companies. Nevertheless, with the limited data available, which was very basic information about the company and detailed

information about the investments around the company, the model managed to predict much better than random. So, very probably, making use of more data, we would be able to get closer to solve the problem for a real world scenario.

Improvement

The algorithms used in this project were, in my opinion, well prepared and tuned. However, improvements could have been done regarding the imbalance of the data. A few more techniques such as oversampling, or even searching for more data, could have been used to get around this characteristic of the data. Also, there are many penalized classification algorithms that could be useful for the problem and they could have been explored in more depth.

References

[1] Markus Böhm et al. "The Business Model DNA: Towards an Approach for Predicting Business Model Success". In: (2017).

[2] WilliamB Gartner, JenniferA Starr, and Subodh Bhat. "Predicting new venture survival: an analysis of "anatomy of a start-up." cases from Inc. Magazine". In: *Journal of Business Venturing* 14.2 (1999), pp. 215–232.

[3] Guang Xiang et al. "A Supervised Approach to Predict Company Acquisition with Factual and Topic Features Using Profiles and News Articles on TechCrunch." In: *ICWSM*. 2012.