Startup Success Predictor

(CS 5785) Applied Machine Learning, Cornell Tech, Fall 2022, Volodymyr Kuleshov

Hassan Kheireddine
hk682@cornell.edu

Lillian Chen
lc966@cornell.edu

ABSTRACT

Living in this world made up of rapid growth and innovation, more and more businesses and individuals aim to develop unique products and services, bring them to the market and change the world. They build over extremely high risks, hoping to fill the market gap they noticed. These kinds of businesses and companies are considered as startups. Some most familiar startups in BigTech include Netflix, Apple,... and Amazon etc., and there are also countless startups in all sizes and industries. Nevertheless, it is worth mentioning that 90% of the startups fail, and there are indeed some complicated factors behind these failures and successes. In this project, we used machine learning classification models to predict the success/failure of startups, namely Logistic Regression, k-Nearest Neighbors, SVM (Support Vector Machine), Naive Bayes, and Decision Trees. We then introduced Random Forest, ADABoost, and Gradient Boost into the mix, in an effort to improve upon the results we obtained from the previously mentioned classification models.

1. MOTIVATION

In the recent decade, there have been increasingly more entrepreneurs who believe that their ideas could create quite a lot of value to the world and go on the path of creating startups. However, according to the Bureau of Labor, 2 out of 10 new businesses fail in the first year of operations, and according to Startup Genome, 9 out of 10 startups fail. There are many different outcomes that could happen to a startup, and it is best if we could utilize real world data to better understand the success and failure. In the project, our preliminary motivation is to tackle the problem of predicting success and failure for startup companies. Additional objectives include finding out what classifier works best, and the effect of correlation between our two classifiers to a startup's success. This project will be an application to the real world. The data we found includes 9 different quantitative features and 10 different categorical features. We will determine the success/failure of each startup by setting the two paths of successes. The first is to get into the S&P 500, and the second is to be acquired by another company.

2. CONTEXT

Countless huge companies, individuals and investors are interested in the prediction of a startups' success or failure, but how can we determine it? Historically, researchers have used survey studies, longitudinal studies, and case-based inductive studies to approach this question. Nowadays, the availability of

data has tremendously grown. Hence, researchers started using machine learning and data analysis techniques to obtain a more precise prediction. For example, the 2021 study representing McKinsey's Technology, Media & Telecommunications Practice by Baroudy et al studied the top 1,000 most successful European tech startups. They analyzed them along 15 critical growth dimensions and variables, using techniques including clusters and data visualization, and reached the generalization of three key insights: there are four distinct roads to success, different strategic plays require different success factors, and the importance of huge funding. Furthermore, another study in 2021 by Zbikowski and Antosiuk utilizes machine learning and compared three algorithms, including regression, support vector machine, and the gradient boosting classifier, reaching F1 scores of 57%, 34%, and 43% respectively. From here, we hope to build upon the prior experiments by testing more models that has the potential to create a more accurate predictive model, yet to gain more insights of the key factors that affect a startup's success or failure.

3. PREPROCESSING

The data we used for this paper was collected from Kaggle, shared by Ramikishan Panthena, a machine learning engineer at GMO. The data contains industry trends, investment insights and individual company information, summing to a total of 923 entries and 49 columns. We dropped 15 columns and added 9 columns in total, coming to a final of 43 columns by the end of our preprocessing.

3.1 Initial Visualization of Data

Looking at the data (specifically, the columns within), there were a few values of interest that we thought would be key to making "educated" predictions whether a company has potential to either be acquired or enter the S&P 500.

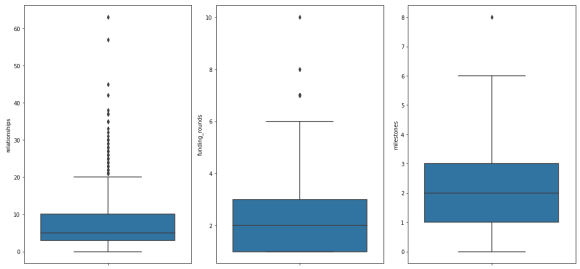

Fig 1: distribution of startups according to # of relationships (i.e.: investors), # of funding rounds, and # of milestones (which include funding rounds, VCs, etc.).

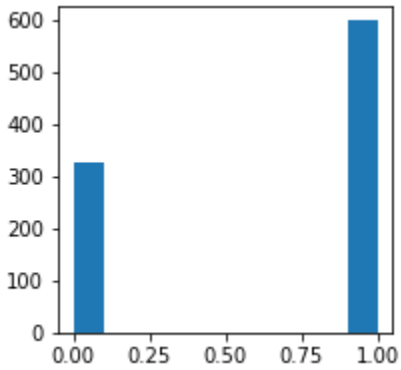

Fig 2: status of startups, where 1 means acquired, and 0 means not acquired.

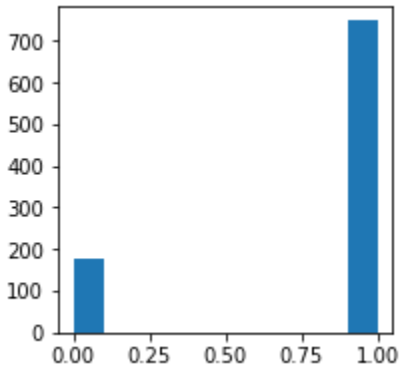

Fig 3: number of startups that made it to the S&P 500 at any point during their existence, where 1 means it has been on there, and 0 means it hasn't.

From these initial observations, we see that the majority of startups in the dataset have both either been acquired or made it to the S&P 500. Although this can be problematic as it is not what we expected (as mentioned before, according to Startup Genome, 9 out of 10 startups fail), bias could come into play here. However, despite the least likely outcome(s) being the majority in this dataset, it makes sense that they are, since the average startup in the database has over 5 relationships (i.e.: investors, VCs, etc.), successfully had 2 funding rounds, and reached 2 crucial milestones that surely contributed to their success.

3.2 Details of Preprocessing + Feature Engineering

To discuss further in detail, the first step of preprocessing we took is to convert the "status" column into binary classifiers as "0" and "1". Later on, we dropped 13 irrelevant columns as far as the model is concerned. Next, we provided values of the means of the columns to the empty rows of data for "age_first_milestone_year". We also did the same for "age_last_milestone_year", to avoid there being any empty values. We then study the "state_code" and "category_code" to see if there are more columns worth one-hot encoding separately than including them in the one-hot encoded "is_otherstate" and "is_othercategory" columns respectively. We noticed that the four states, WA, CO, IL, and PA, have considerable counts of startups so we then set a threshold of 15+ startups and separate them from "is_otherstate", which contains all other startup hubs that are not as well-known. By this step we added 4 columns. Similarly, by analyzing "category_code" we added columns for semiconductor, network_hosting, hardware, public_relations, and cleantech with a threshold of 20+ startups. After these two steps, we dropped both "state_code" and "category_code" since we no longer need them. Finally, we normalized all features, with the hopes of getting better results than we did in preliminary experiments. This comes to the end of our data preprocessing, which left us with 43 columns with essential information, useful classifiers, and non-empty rows.

3.3 Study Correlation Between Labels

Before we start to build our model, we decide to study the correlation of our labels, which are "is_top500" and "status" respectively. We noticed that none of their correlations with other columns exceed 0.4, which they probably do not have any correlation that is significantly strong. In particular, "is_top500" and "status" have a correlation coefficient of 0.310652, and they are among the three strongest correlations of each other. Specifically, "is_top500" has a slightly stronger correlation of 0.331581 with "avg_participants", and "status" has two stronger correlations of 0.360434 and 0.328260, respectively with "relationships" and "milestones". By analyzing the correlation, we get a better idea of how we will look deeper into the data while we start building our actual models. This also set us a stage to discuss the relationship between the correlation and the classification of

success by the end of the project when we gain more insights from the intricacies of machine learning.

is_top500	1.000000
avg_participants	0.331581
status	0.310652
has_roundB	0.305183
funding_rounds	0.259345
has_roundC	0.234862
relationships	0.222207
age_last_funding_year	0.200727
age_last_milestone_year	0.181288
milestones	0.171412
has_roundA	0.162354
has_roundD	0.152300
age_first_milestone_year	0.136128
is_CA	0.098677
has_VC	0.096456
is_MA	0.085095
is_semiconductor	0.053048
is_hardware	0.051528
age_first_funding_year	0.050638
funding_total_usd	0.048978
is_mobile	0.040063
is_othercategory	0.039686
is_software	0.038376
is_biotech	0.036359
latitude	0.032675
is_CO	0.031523
is_consulting	0.027718
is_WA	0.013348
is_enterprise	0.009400
is_IL	0.008622
is_cleantech	0.006824
is_gamesvideo	-0.001011
is_network_hosting	-0.007567
is_NY	-0.015463
is_TX	-0.026352
is_PA	-0.036067
is_public_relations	-0.037936
is_advertising	-0.046028
is_ecommerce	-0.054925
longitude	-0.091913
is_web	-0.118123
is_otherstate	-0.209409
has_angel	-0.222777

status	1.000000
relationships	0.360434
milestones	0.328260
is_top500	0.310652
age_last_milestone_year	0.229893
has_roundB	0.208257
funding_rounds	0.206049
avg_participants	0.185992
has_roundA	0.184307
has_roundC	0.165902
age_first_milestone_year	0.140320
has_roundD	0.139940
is_MA	0.081735
is_CA	0.077217
is_enterprise	0.073772
age_last_funding_year	0.073731
is_NY	0.059996
latitude	0.046560
is_advertising	0.044355
funding_total_usd	0.040176
is_othercategory	0.032783
is_CO	0.027310
is_network_hosting	0.024173
is_semiconductor	0.016162
is_software	0.012429
is_mobile	0.007312
is_consulting	0.002373
is_biotech	0.000104
is_web	-0.000873
is_gamesvideo	-0.025893
is_WA	-0.034433
longitude	-0.036092
is_IL	-0.043317
is_TX	-0.045309
has_VC	-0.056515
is_cleantech	-0.070913
is_ecommerce	-0.072193
has_angel	-0.072840
age_first_funding_year	-0.075637
is_PA	-0.084219
is_public_relations	-0.086157
is_hardware	-0.086946
is_otherstate	-0.154121

Fig 4.1 (left): correlation between all features and "is_top500" label.
Fig 4.2 (right): correlation between all features and "status" label.

4. METHOD

In this project, five machine learning classification models are used, namely Logistic Regression, K-Nearest Neighbors, Support Vector Machine, Naive Bayes Classifier, and Decision Trees.

4.1 Logistic Regression

Logistic Regression is a supervised learning binary classification algorithm which uses a model of the form:

$$f_\theta(x) = \sigma(\theta^\top x) = \frac{1}{1+\exp(-\theta^\top x)}$$

where $\sigma(z) = \frac{1}{1+\exp(-z)}$.

Logistic regression is commonly used to calculate the probability of two-outcomes, for example yes and no, and to due with classification of binary events.

4.2 K-Nearest Neighbors

K- Nearest Neighbors is a non-parametric supervised learning data algorithm that given a query datapoint x, will find K training examples that are closest to x and determine its class. While it can be used as both a regression and classification problem, it is typically used as classification.

4.3 Support Vector Machine

The objective of the support vector machine is to find a maximum margin hyperplane in a N-dimensional space that distinctly classifies the data points, where N is the number of features. Maximizing the margin can be done by the following form:

$$\min_{\theta,\theta_0,\xi} \sum_{i=1}^{n} \underbrace{\left(1 - y^{(i)}\left((x^{(i)})^\top\theta + \theta_0\right)\right)^+}_{\text{hinge loss}} + \underbrace{\frac{\lambda}{2}||\theta||^2}_{\text{regularizer}}$$

4.4 Naive Bayes Classifier

Naive Bayes is a supervised learning multiclass classification. It is based on applying Bayes' theorem and the "naive" assumption of conditional independence between every pair of features given the value of the class variable. The *Bernoulli Naive Bayes* model $P_\theta(x,y)$ is defined for *binary data* $x \in \{0,1\}^d$ (e.g., bag-of-words documents). The θ contains prior parameters $\phi = (\phi 1,...,\phi K)$ and K sets of per-class parameters $\psi k = (\psi 1k,...,\phi dk)$. The probability of the data x for each class equals:

$$P_\theta(x|y=k) = \prod_{j=1}^{d} P(x_j \mid y=k),$$

where each $P_\theta(xj \mid y=k)$ is a Bernoullli(ψjk).

4.5 Decision Trees

Decision tree is a supervised model of the form:

$$f(x) = \sum_{R \in \mathcal{R}} y_R \mathbb{I}\{x \in R\}.$$

where $\mathbb{I}\{\cdot\}$ is an indicator function (one if true, else zero) and values $yR \in \in Y$ are the outputs for that region. The set R is a collection of decision regions obtained by recursive binary splitting. The rules defining the regions R can be organized into a tree, with one rule per internal node and regions being the leaves. The main benefit of decision trees is its simplicity to visualize and understand when in the decision making process.

4.6 Random Forest

The Random Forest Algorithm is a supervised learning algorithm that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. Namely, it performs bagging on decision trees in order to mitigate the negative effects that high depth levels, overfitting, and variance have on accuracy levels (all of which are well-known problems of standalone decision trees).

4.7 ADABoost

ADABoost is a supervised learning algorithm that follows the idea of boosting: correcting underfitting by combining models (usually what we call "weak learners") that correct each others' errors. Here's how ADABoost works:

1) Fit a weak learner g_0 on dataset

$$\mathcal{D} = \{(x^{(i)}, y^{(i)})\}. \text{ Let } f = g_0.$$

2) Compute weights $w^{(i)}$ for each i based on model predictions $f(x^{(i)})$ and targets $y^{(i)}$. Give more weight to points with errors.
3) Fit another weak learner g_1 on $D=\{(x^{(i)},y^{(i)})\}$ with weights $w^{(i)}$.
4) Set $f_1=g_0+\alpha_1 g$ for some weight α_1. Go to Step 2 and repeat.

4.8 Gradient Boost

Gradient Boosting is a supervised learning algorithm that also follows the idea of boosting: correcting underfitting by combining models (usually what we call "weak learners") that correct each others' errors. However, Gradient Boosting works differently: it performs functional gradient descent with approximate gradients. Here's how it works:

Start with f(x)=0. Then, at each step t>1:

1. Create a training dataset D_g and fit $g_t(x^{(i)})$ using loss L_g:

$$g_t(x) \approx \frac{\partial L(\mathrm{y}, \mathrm{f})}{\partial \mathrm{f}}\bigg|_{\mathrm{f}=f_0(x)}.$$

2. Take a step of grad descent using approximate grads with step α_t:

$$f_t(x) = f_{t-1}(x) - \alpha_t \cdot g_t(x).$$

5. SETUP

5.1. Initial Experiment ("First Try")

After preprocessing, engineering some features, and analyzing the data in front of

us, we combined the two labels under one y as a tuple (to make the train-test split easier), and then we split our data into 80% training and 20% testing data. We then used K-folds cross-validation to introduce randomness into the training and testing of the data. With all that said, here's what we obtained the first time around

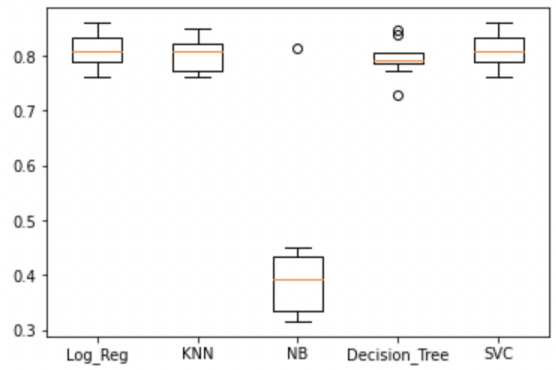

Fig 5.1: Testing accuracy results for label 1 ("is-top500")

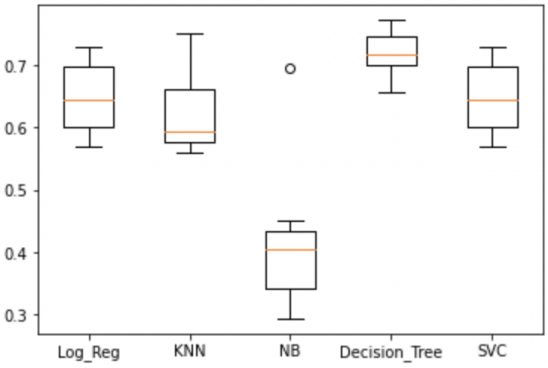

Fig 5.2: Testing accuracy results for label 2 ("status")

Instantly, looking at these results, Naive Bayes results were much lower than we expected for both labels, and we don't quite know why this was the case. Overall, we got better results for label 1 compared to label 2. We did not foresee this, and we attempted to investigate ways to improve these numbers.

5.2 1st Optimizing: Normalization of *All* Features

In our first effort to improve our results in one way or another, the first adjustment we made to our initial experiment was normalizing the data. We thought this would be helpful because of the varying ranges of all values in the dataset: We have some features that are only defined by zero or one values (One-Hot Encoded), some are relatively small float values (i.e.: latitude and longitude), and some are absolutely huge (i.e.: "funding_total_usd"). Thankfully, our results improved substantially across the board, especially with Naive Bayes.

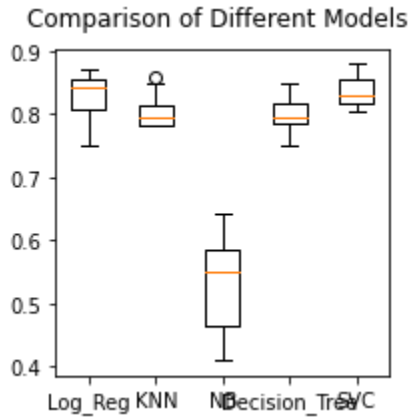

Fig 6.1: Testing accuracy results for label 1 ("is-top500") after normalization

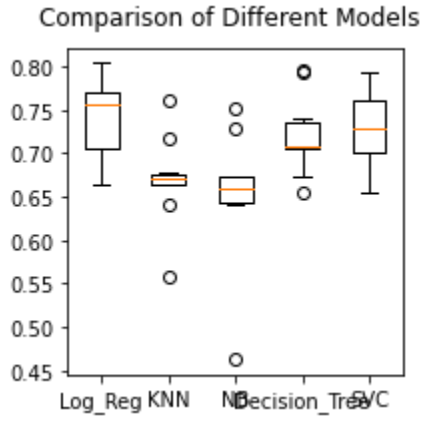

Fig 6.2: Testing accuracy results for label 2 ("status") after normalization

Even with these improvements, we still wanted to investigate how we can improve these test accuracy numbers even further. At

this point, we thought we'd mitigate underfitting in some form.

5.3 2nd Optimizing: Introducing Random Forest, ADABoost, and Gradient Boost

With Decision Trees being our second-best performing algorithm overall, we thought we could test the waters with Random Forest, which is a constitution of multiple decision trees at training time. We also had fears regarding the possibility of underfitting, and so we introduced two algorithms of boosting: ADABoost and Gradient Boost. As we predicted, we observe the 3 best test accuracy results yet, especially with label 2:

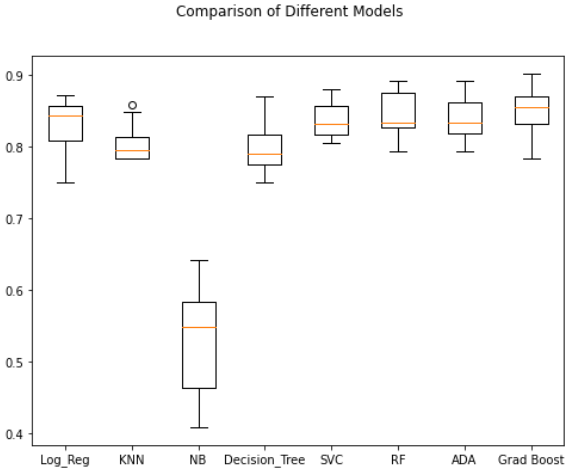

Fig 7.1: Testing accuracy results for label 1 ("is-top500") after adding the 3 new algorithms to the right of the figure (Random Forest, ADABoost, and Gradient Boost)

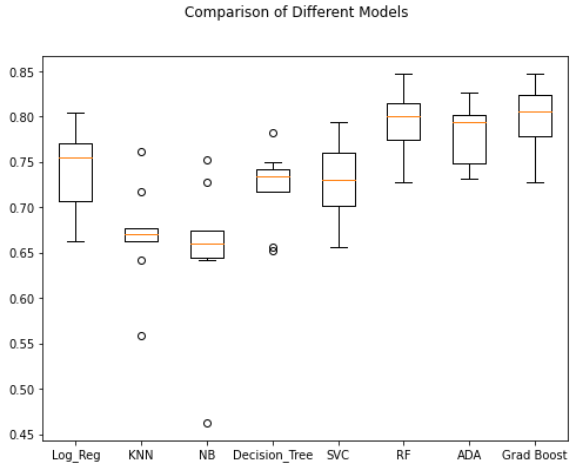

Fig 7.2: Testing accuracy results for label 2 ("status") after adding the 3 new algorithms to the right of the figure (Random Forest, ADABoost, and Gradient Boost)

6. OUTCOMES AND RESULTS

With the purpose of this project being to understand what determines a startup to become successful (or a failure), we constructed several models based on given crucial funding-related parameters that really contribute to building a startup from the ground up. Upon preprocessing our data and engineering some crucial features, we managed to first use the likes of Logistic Regression, k-Nearest Neighbors, SVM (Support Vector Machine), Naive Bayes, and Decision Trees, and obtained rather underwhelming results. From there, we fine-tuned our features by normalizing them, and then introduced Random Forest, ADABoost, and Gradient Boost. Thankfully, with these changes, our best performing models (Random Forest & Gradient Boost) had 84% accuracy for label 1 ("is_top500") and just under 80% for label 2 ("status"), proving that funding and investment, as

expected, play a huge role in the success of a startup.

In the future, we'd want to use a dataset that actually reflects the reality of launching a startup, since most startups in this dataset have either been in the top 500 at some point or have been acquired, and that's obviously not the reality of the space. So, to mitigate the risk of this model being biased, and in order to include more decisive features that determine a startup's success, we'd test our proposed model on the suggested dataset. The main takeaway from this study is indeed that the more money your startup receives, the more likely it is to succeed, because, logically speaking, you will, in the process, obtain all the resources and firepower to turn your idea into a successful business.