Startup Success Prediction

1st Chinmay Patankar
Department of Mathematics
Stevens Institute of Technology
Hoboken, United States
cpatanka@stevens.edu

2nd Mandar Parab
Department of Mathematics
Stevens Institute of Technology
Hoboken, United States
mparab1@stevens.edu

3rd Kush Jani
Department of Mathematics
Stevens Institute of Technology
Hoboken, United States
kjani1@stevens.edu

***Abstract*—Startups are companies or ventures that are focused on a single product or service that the founders want to bring to market. Startups play a major role in economic growth and a few are valued at over US 1 billion dollars. So how do we know whether a startup is successful or not? The success of a startup is defined based on the amount of money they acquired through the process of Merger and Acquisition or an IPO (Initial Public Offering). To solve this we are using a data set that contains detailed information of a startup including its various stages of funding to identify characteristics of a successful startup. Identifying these characteristics is one of the main aims of the project. Creating a visual representation of the features helps us identify which are significant. For this we are using three machine learning algorithms which are Logistic Regression, K Nearest Neighbour (KNN), Random Forest, Adaptive Boosting and Gradient Boosting .**

I. Introduction

Our main purpose is to find out whether a presently operational startup will succeed or fail. We use a startup success prediction data set for this, which includes both quantitative and categorical factors that aid in making a solid prediction. We seek to achieve this goal by combining Machine Learning techniques with the Python programming language. We can use machine learning algorithms to predict if a startup will succeed or not. The goal of this project is to provide a variety of insights based on classification. We will utilize three algorithms for this project: Logistic Regression, K Nearest Neighbour (KNN), Random Forest, Adaptive Boosting and Gradient Boosting .

II. Related Work

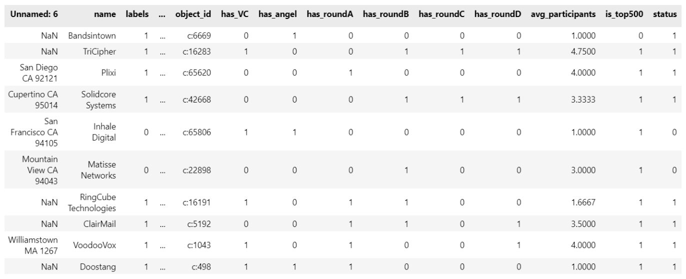
A group of scientists(Phys.Org) was able to develop an AI that could accurately predict whether a startup will end up being the next big thing, or it will be another failed attempt to disrupt the market. The machine learning models that were used for this AI predicting tool look into over a million companies to further say the scalability of a startup. That said, if the tool turns out to be absolutely accurate, more investors could help the capitalization of these budding startups. On top of that, it would also be beneficial to investors as they will no longer be risking their money for startups that would end up being a waste of their time.

III. Our Solution

To achieve the objective is to analyze the startup behaviour based on several variables and determine what variables affects startups success the most, then build the model that can predict the success of a startup.
We identify best possible solution to predict the success rate of startup by creating and comparing multiple machine learning models. We identified five machine learning algorithms for classification as our output is binary and optimized these algorithms to achieve a higher accuracy.
After comparing the results of all the machine learning algorithms we identify the algorithm with the highest accuracy as the best algorithm suitable for our problem dataset.
After comparing result of the algorithms, Boosting algorithms have achieved a higher accuracy. The boosting algorithms are, Adaptive Boosting and Gradient Boosting gives a better prediction in predicting the success of startup.

A. Description of Dataset

The dataset contains 48 columns/features. The data included industry trends, investment insights and individual company information about a startup like name, state(which state the startup is located in), city(city in which the startup has its main office), categorycode (which industry does the startup belong to), it also includes individual industry features such as issoftware, isweb, ismobile, isenterprise, isadvertising, isgamesvideo, isecommerce, isbiotech, isconsulting, zipcode, latitude (Latitude coordinate for Global Coordinate System,), longitude (Longitude coordinate for Global Coordinate System), fundingtotalusd (total amount of vending received), foundedat (year the startup was found in), closedate (year the startup was shutdown), agefirstfundingyear (age of the startup when it received its first funding), agelastfundingyear (age of the startup when it received its last funding), firstfundingat (year the startup received its first funding), lastfundingat (year the startup received its last funding), relationships, fundingrounds (number of funding rounds the startup went through), milestones (number of milestones in the startup's lifespan), agefirstmilestoneyear (age of the startup at its first milestone), agelastmilestoneyear (age of the startup at its last milestone), hasVC (whether startup received funding from Venture capital or not), hasangel (whether startup received angel funding or not), hasroundA (whether startup received round A funding or not), hasroundB (whether startup received round B funding or not), hasroundC (whether startup received round C funding or not), hasroundD (whether startup received round D funding or not), avgparticipants (average number of funding received), istop500 (startup is top 500 or not), status(acquired/closed - the target variable, if a startup is 'acquired' by some other organization, means the startup succeed).



Unnamed: 6	name	labels	...	object_id	has_VC	has_angel	has_roundA	has_roundB	has_roundC	has_roundD	avg_participants	is_top500	status
NaN	Bandsintown	1	...	c:6669	0	1	0	0	0	0	1.0000	0	1
NaN	TriCipher	1	...	c:16283	1	0	0	1	1	1	4.7500	1	1
San Diego CA 92121	Plixi	1	...	c:65620	0	0	1	0	0	0	4.0000	1	1
Cupertino CA 95014	Solidcore Systems	1	...	c:42668	0	0	0	1	1	1	3.3333	1	1
San Francisco CA 94105	Inhale Digital	0	...	c:65806	1	1	0	0	0	0	1.0000	1	0
Mountain View CA 94043	Matisse Networks	0	...	c:22898	0	0	0	1	0	0	3.0000	1	0
NaN	RingCube Technologies	1	...	c:16191	1	0	1	1	0	0	1.6667	1	1
NaN	ClairMail	1	...	c:5192	0	0	1	1	0	1	3.5000	1	1
Williamstown MA 1267	VoodooVox	1	...	c:1043	1	0	1	0	0	1	4.0000	1	1
NaN	Doostang	1	...	c:498	1	1	1	0	0	0	1.0000	1	1

Fig 1. Sample of the Dataset

As this dataset has a lot of features we have therefore dropped unnecessary and redundant features. There were some null value entries within the dataset which we removed and cleaned from the main dataset.

B. *Machine Learning Algorithms*

Our dataset has two target classes and hence a binary output. We are using following five ML algorithms:

1. Logistic Regression

2. K Nearest Neighbor (KNN)

3. Random Forest

4. Adaptive Boosting

5. Gradient Boosting

Firstly, we have implemented Logistic Regression Algorithm. The Logistic Regression algorithm was chosen as it works best when there is a binary output. Our dataset has two target classes 'acquired' and 'not acquired' which we have encoded with 1 and 0 in the status column.

The k-nearest neighbors (KNN) algorithm is a data classification method for estimating the likelihood that a data point will become a member of one group or another based on what group the data points nearest to it belong to. The k-nearest neighbor algorithm is a type of supervised machine learning algorithm used to solve classification and regression problems. However, it's mainly used for classification problems. In short, KNN involves classifying a data point by looking at the nearest annotated data point, also known as the nearest neighbor.

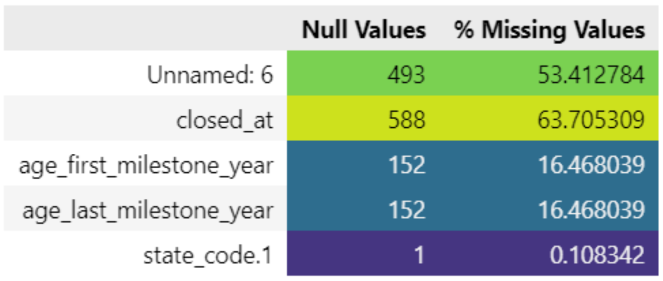
Random Forest classification algorithm will be used. It utilizes ensemble learning, which is a technique that combines many classifiers to provide solutions to complex problems. By developing multiple decision trees and considering the majority of the output values from these decision trees we will be able to correctly predict whether a startup is successful or not.

Boosting is a general ensemble method that creates a strong classifier from a number of weak classifiers. This is done by building a model from the training data, then creating a second model that attempts to correct the errors from the first model. Models are added until the training set is predicted perfectly or a maximum number of models are added. AdaBoost was the first really successful boosting algorithm developed for binary classification. It is the best starting point for understanding boosting. AdaBoost is best used to boost the performance of decision trees on binary classification problems.

Gradient boosting is a generalisation of AdaBoost algorithm. Gradient Boosting trains many models in a gradual, additive and sequential manner. The major difference between AdaBoost and Gradient Boosting Algorithm is how the two algorithms identify the shortcomings of weak learners (e.g. decision trees). While the AdaBoost model identifies the shortcomings by using high weight data points, gradient boosting performs the same by using gradients in the loss function. The loss function is a measure indicating how good are model's coefficients are at fitting the underlying data.

C. *Implementation Details*

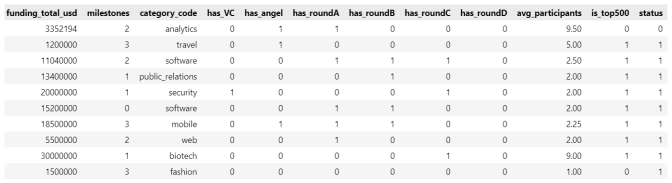
Project includes, applying algorithms such as Logistic Regression, KNN, Random Forest, Adaptive Boosting and Gradient Boosting to obtain insights and make predictions. First comes the data analysis before which we did removed null values in dataset. Below is the initial null value report of the dataset.



	Null Values	% Missing Values
Unnamed: 6	493	53.412784
closed_at	588	63.705309
age_first_milestone_year	152	16.468039
age_last_milestone_year	152	16.468039
state_code.1	1	0.108342

Fig 2. Null value statistics

Columns "Unnamed:6", "closedat", "agefirstmilestoneyear", "agelastmilestoneyear" and "statecode.1" are the columns which have null values. We first try to eliminate the null values in "Unnamed: 6" columns. After close examination we find out the column is a combination of city, zipcode and statecode so we removed the contents of the column and replaced it by combination of city, zipcode and statecode. The "closedat" column represent when the startup closed and null values indicate the startup has bee acquired when comparing the null values with status column. So we replaced the null values with newest date from the founded column. The 'agefirstmilestoneyear' and 'agelastmilestoneyear' represent age of the startup on its first and last milestone. So, the null values were replaced with zero. We dropped the "statecode.1" column because its a duplicate of the "statecode" column and similarly we dropped the ”label” column because its a duplicate of the status column. Below are some graphs that show relation between different attributes.



funding_total_usd	milestones	category_code	has_VC	has_angel	has_roundA	has_roundB	has_roundC	has_roundD	avg_participants	is_top500	status
3352194	2	analytics	0	1	1	0	0	0	9.50	0	0
1200000	3	travel	0	1	0	0	0	0	5.00	1	1
11040000	2	software	0	0	1	1	1	0	2.50	1	1
13400000	1	public_relations	0	0	0	1	0	0	2.00	1	1
20000000	1	security	1	0	0	0	1	0	2.00	1	1
15200000	0	software	0	0	1	1	0	0	2.00	1	1
18500000	3	mobile	0	1	1	1	0	0	2.25	1	1
5500000	2	web	0	0	1	0	0	0	2.00	1	1
30000000	1	biotech	0	0	0	0	1	0	9.00	1	1
1500000	3	fashion	0	0	0	0	0	0	1.00	0	1

Fig 3. Sample of the Dataset after Data cleaning and Data pre-processing

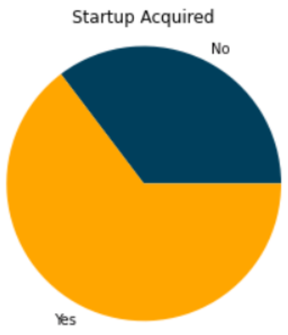


Fig 4. Pie chart showcasing acquired(successful) and not acquired(unsuccessful) startups

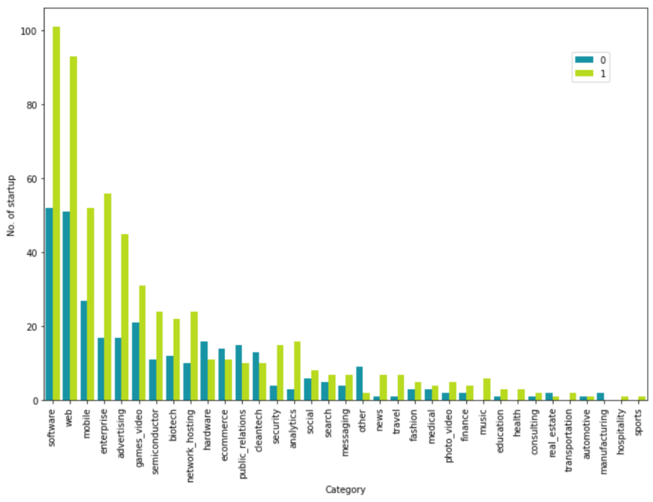


Fig 5. Which industry has most number of startup.

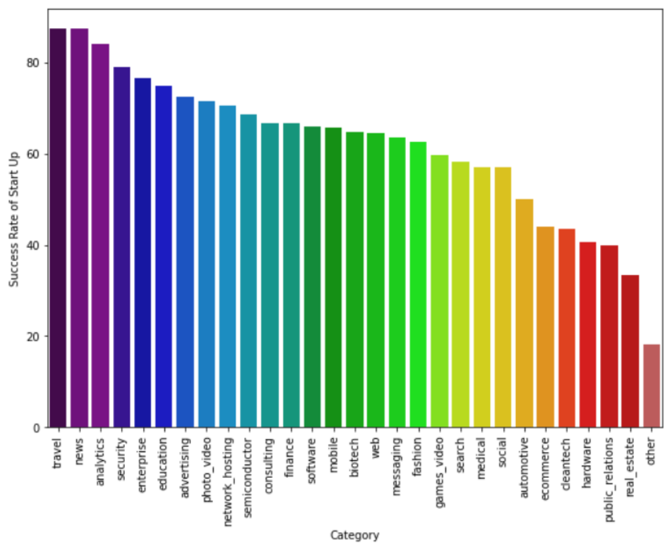


Fig 6. Which industry has the highest success rate for a startup.

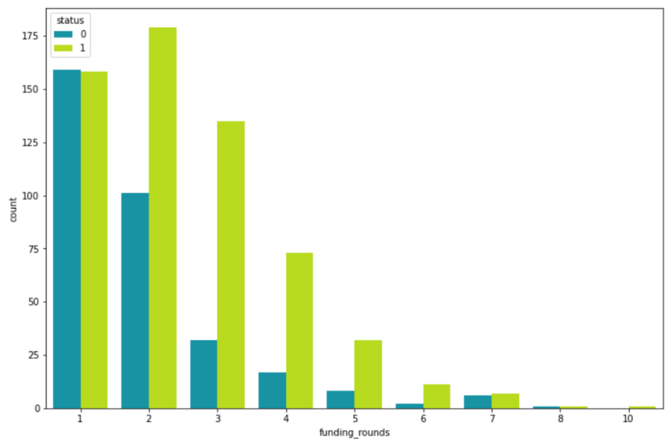


Fig 7. Funding received per round for a successful(1) and unsuccessful(0) startup.

We have made different visualizations for better understanding of the data. In figure 4 we show how many startups have been acquired and not acquired. In figure 5 a bar graph visualizes how many successful and unsuccessful startups are there in each industry. In figure 6, the graph highlights the success rate of a startup in each industry and figure 7, the graph shows the amount of funding received in each stage of the funding round for an acquired and not acquired startups. Before we move forward with modeling, we did some preprocessing and feature engineering by first checking for duplicate values, then negative values in attributes like ”agefirstfundingyear”, ”agelastfundingyear”, ”agefirstmilestoneyear” and ”agelastmilestoneyear” then removing it. Then we logged the value and checked for any outliers in the above attribute including ”fundingtotalusd”. We combined multiple columns into one and dropped the unnecessary columns such as reduced the relationship column and assigned tiers of relationship such as 1,2, 3 and 4 based on definite intervals. Below is the graph showcasing feature engineering of relationship attribute and checking any outliers.

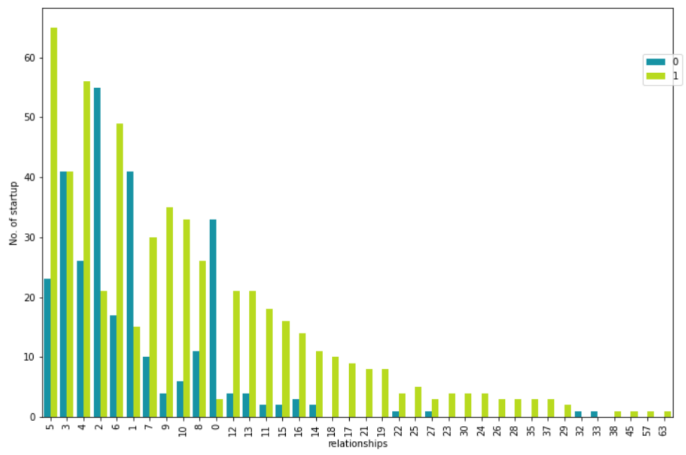


Fig 8. Number of relations of a startup

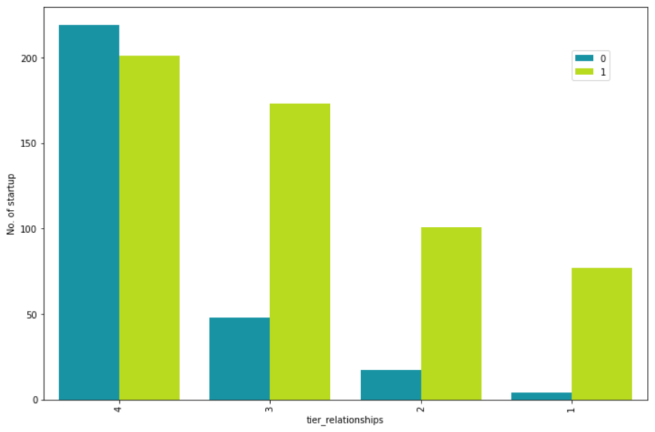


Fig 8. Converted relations into tiers

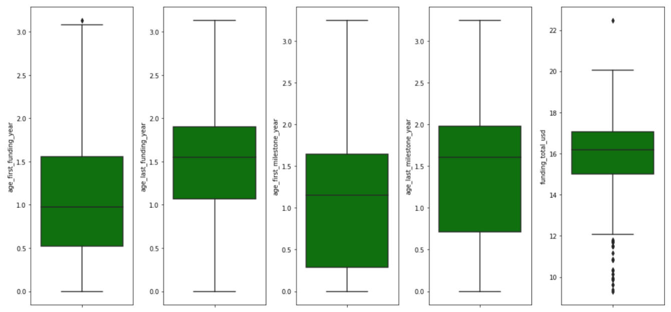


Fig 8.Box plot graph for checking outliers

Logistic Regression (also called Logit Regression) is commonly used to estimate the probability that an instance belongs to a particular class.If the estimated probability is greater than 50 percent, then the model predicts that the instance belongs to that class (called the positive class, labeled "1"), or else it predicts that it does not (i.e., it belongs to the negative class, labeled "0"). This makes it a binary classifier. It uses a logistic function also called the sigmoid function:

$$f(value) = 1/(1 + e^{-value})$$

that outputs a number between 0 and 1. Since the dataset has 900 instances we optimized our logistic regression algorithm by using LIBLINEAR solver that is a linear classifier for data. We saw an increase in the accuracy after using liblinear compare to the default solver LBFGSB which stands for Large scale Bound constrained Optimization. Below is a comparison of the accuracy when using both the solver.

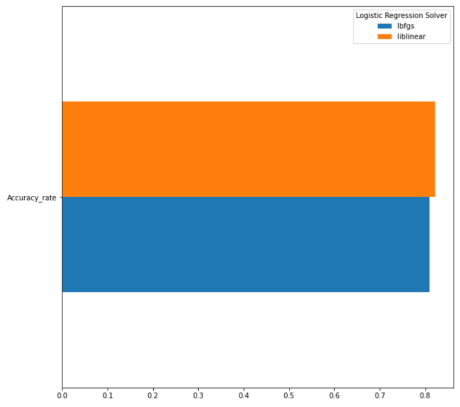


Fig 9. Comparison of two optimizing algorithms

The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems. KNN performs better with a lower number of features than a large number of features. KNN algorithm is predominantly used as a classifier. KNN performs a voting mechanism to determine the class of an unseen observation. This means that the class with the majority vote will become the class of the data point in question. If the value of K is equal to one, then we'll use only the nearest neighbor to determine the class of a data point. If the value of K is equal to ten, then we'll use the ten nearest neighbors, and so on. Therefore it is important to choose the correct value of K which we decide by comparing the error curves of test and train data.

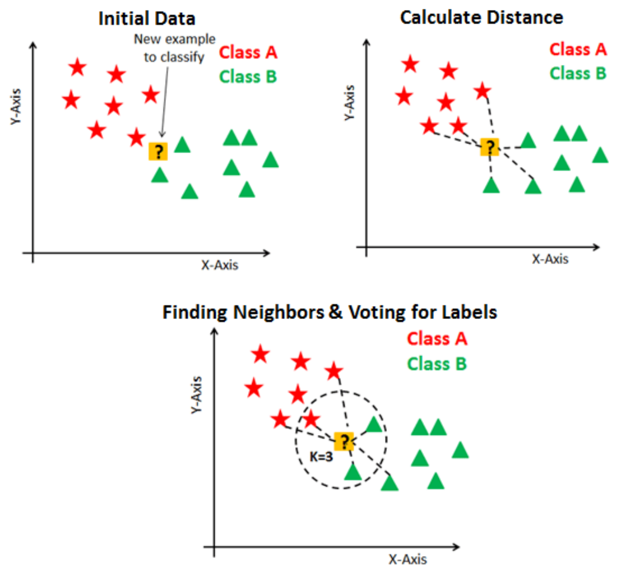


Fig 10. Working of KNN algorithm

K is a crucial parameter in the KNN algorithm. Using error curves the figure below shows error curves for different values of K for training and test data.At low K values, there is overfitting of data/high variance. Therefore test error is high and train error is low. At K=1 in train data, the error is always zero, because the nearest neighbor to that point is that point itself. Therefore though training error is low test error is high at lower K values. As we increase the value for K, the test error is reduced. But after a certain K value, bias/underfitting is introduced and test error goes high. So we can say initially test data error is high(due to variance) then it goes low and stabilizes and with further increase in K value, it again increases(due to bias). The K value when test error stabilizes and is low is considered as optimal value for K. From the above error curve we can choose K=6 for our KNN algorithm implementation.

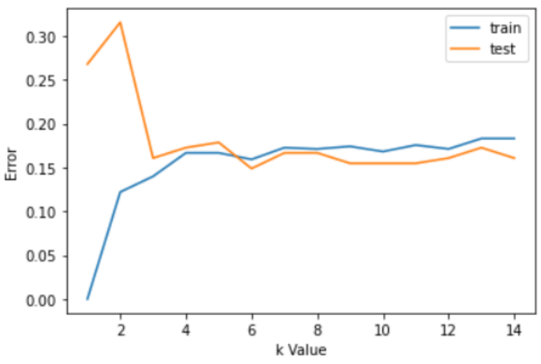


Fig 10. Choosing a value for K

Classification in random forests employs an ensemble methodology to attain the outcome. The training data is fed to train various decision trees. This dataset consists of observations and features that will be selected randomly during the splitting of nodes. A rain forest system relies on various decision trees. Every decision tree consists of decision nodes, leaf nodes, and a root node. The leaf node of each tree is the final output produced by that specific decision tree. The selection of the final output follows the majority-voting system. In this case, the output chosen by the majority of the decision trees becomes the final output of the rain forest system. The diagram below shows a simple random forest classifier.

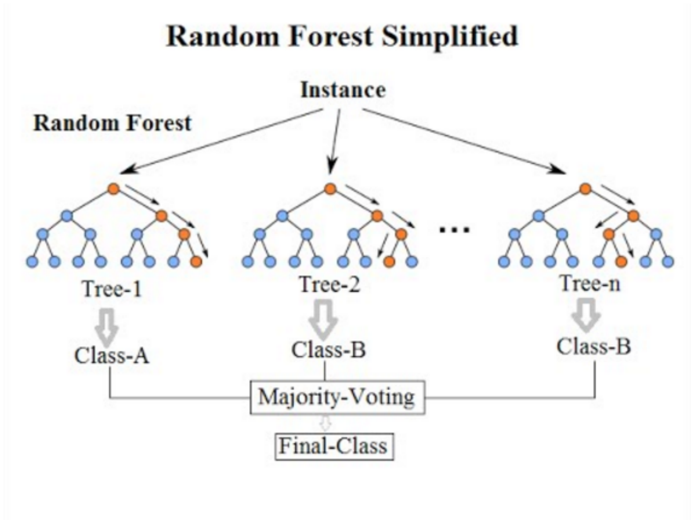


Fig 11. Working of Random Forest

For Adaptive boosting, predictions are made by calculating the weighted average of the weak classifiers. The AdaBoost algorithm is an iterative procedure that combines many weak classifiers to approximate the Bayes classifier. Starting with the unweighted training sample, the AdaBoost builds a classifier, for example a classification tree, that produces class labels. If a training data point is misclassified, the weight of that training data point is increased (boosted). A second classifier is built using the new weights, which are no longer equal. Again, misclassified training data have their weights boosted and the procedure is repeated. Typically, one may build 500 or 1000 classifiers this way. A score is assigned to each classifier, and the final classifier is defined as the linear combination of the classifiers from each stage. We used SAMME — Stagewise Additive Modeling using a Multi-class Exponential loss function as it works best for our problem.

```
1. Initialize the observation weights $w_i = 1/n,\ i = 1, 2, \ldots, n$.
2. For $m = 1$ to $M$:
    (a) Fit a classifier $T^{(m)}(\boldsymbol{x})$ to the training data using weights $w_i$.
    (b) Compute
        $$err^{(m)} = \sum_{i=1}^{n} w_i \mathbb{I}\left(c_i \neq T^{(m)}(\boldsymbol{x}_i)\right) / \sum_{i=1}^{n} w_i.$$
    (c) Compute
        $$\alpha^{(m)} = \log \frac{1 - err^{(m)}}{err^{(m)}} + \log(K-1).$$
    (d) Set
        $$w_i \leftarrow w_i \cdot \exp\left(\alpha^{(m)} \cdot \mathbb{I}\left(c_i \neq T^{(m)}(\boldsymbol{x}_i)\right)\right),\ i = 1, \ldots, n.$$
    (e) Re-normalize $w_i$.
3. Output
        $$C(\boldsymbol{x}) = \arg\max_k \sum_{m=1}^{M} \alpha^{(m)} \cdot \mathbb{I}(T^{(m)}(\boldsymbol{x}) = k).$$
```

Fig 11. Algorithm of SAMME

SAMME is very similar to AdaBoost with a major difference. Now in order for alpha(m) to be positive, we only need

$$(1(err^m)) > 1/K$$

, or the accuracy of each weak classifier be better than random guessing rather than 1/2. As a consequence, the new algorithm puts more weight on the misclassified data points in (2d) than AdaBoost, and the new algorithm also combines weak classifiers a little differently from AdaBoost, i.e. differ by

$$log(K-1) \sum_{m=1}^{M} I(T^m(x) = k)$$

It is worth noting that when K = 2, SAMME reduces to AdaBoost. the extra term

$$log(K-1)$$

is not artificial; it makes the new algorithm equivalent to fitting a forward stage wise additive model using a multi-class exponential loss function.

Gradient boosting involves three elements:

1. A loss function to be optimized.
2. A weak learner to make predictions.
3. An additive model to add weak learners to minimize the loss function.

1. Loss Function:
The loss function used depends on the type of problem being solved. It must be differentiable, but many standard loss functions are supported and you can define your own. For classification we used logarithmic loss.

2. Weak Learner:
Decision trees are used as the weak learner in gradient boosting. Specifically regression trees are used that output real values for splits and whose output can be added together, allowing subsequent models outputs to be added and "correct" the residuals in the predictions. Trees are constructed in a greedy manner, choosing the best split points based on purity scores like Gini or to minimize the loss.

3. Additive Model:
Trees are added one at a time, and existing trees in the model are not changed. A gradient descent procedure is used to minimize the loss when adding trees. Traditionally, gradient descent is used to minimize a set of parameters, such as the coefficients in a regression equation or weights in a neural network. After calculating error or loss, the weights are updated to minimize that error.
Instead of parameters, we have weak learner sub-models or more specifically decision trees. After calculating the loss, to perform the gradient descent procedure, we must add a tree to the model that reduces the loss (i.e. follow the gradient). We do this by parameterizing the tree, then modify the parameters of the tree and move in the right direction by (reducing the residual loss. Generally this approach is called functional gradient descent or gradient descent with functions. The output for the new tree is then added to the output of the existing sequence of trees in an effort to correct or improve the final output of the model.

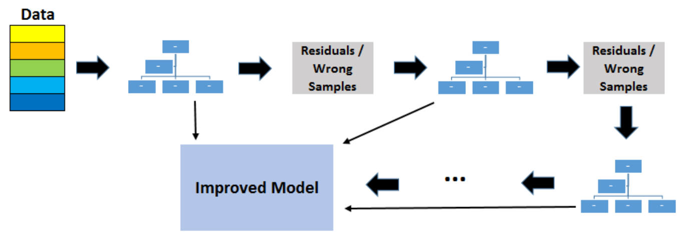
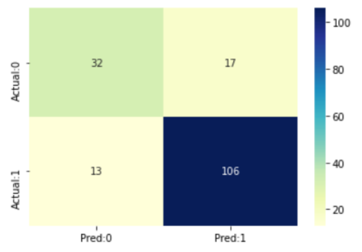


Fig 9. Algorithm of Gradient Boosting

IV. Comparison

After implementing all 5 algorithms it has been found that boosting algorithms like Adaptive and Gradient Boosting have the highest accuracy rate of 86 percent followed by K Nearest Neighbour with an accuracy of 85 percent and with the lowest accuracy of Random Forest and Logistic Regression with an accuracy of 82 percent.

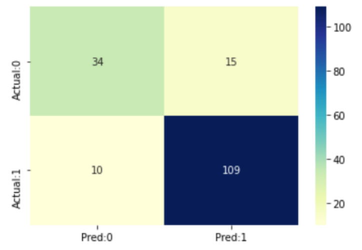
The logistic regression has a accuracy rate of 82 percent with accuracy of predicting successful startup as 89 percent and unsuccessful startup as 65 percent.



```
         Pred:0  Pred:1
Actual:0      32      17
Actual:1      13     106
Percent Acquired correctly predicted:  89.07563025210085
Percent Not Acquired correctly predicted:  65.3061224489796
Accuracy of the test set:  0.8214285714285714
```

Fig 10. Accuracy rate of Logistic Regression

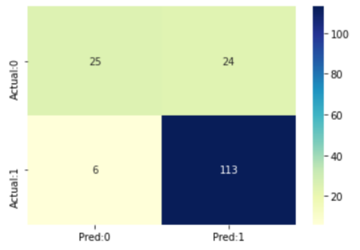
The K nearest neighbour has a accuracy rate of 85 percent with accuracy of predicting successful startup as 91 percent and unsuccessful startup as 69 percent.



```
         Pred:0  Pred:1
Actual:0      34      15
Actual:1      10     109
Percent Acquired correctly predicted:  91.59663865546219
Percent Not Acquired correctly predicted:  69.38775510204081
Accuracy of the test set:  0.8511904761904762
```

Fig 11. Accuracy rate of K Nearest Neighbour

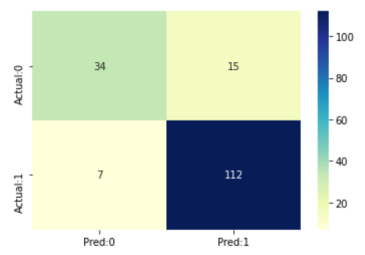
The random forest has a accuracy rate of 82 percent with accuracy of predicting successful startup as 94 percent and unsuccessful startup as 51 percent.



```
         Pred:0  Pred:1
Actual:0      25      24
Actual:1       6     113
Percent Acquired correctly predicted:  94.9579831932773
Percent Not Acquired correctly predicted:  51.02040816326531
Accuracy of the test set:  0.8214285714285714
```

Fig 12. Accuracy rate of Random Forest

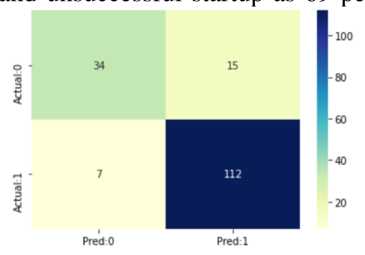
The adaptive boosting has a accuracy rate of 86 percent with accuracy of predicting successful startup as 94 percent and unsuccessful startup as 69 percent.



```
         Pred:0  Pred:1
Actual:0      34      15
Actual:1       7     112
Percent Acquired correctly predicted:  94.11764705882352
Percent Not Acquired correctly predicted:  69.38775510204081
Accuracy of the test set:  0.8690476190476191
```

Fig 13. Accuracy rate of Adaptive Boosting algorithm

The gradient boosting has a accuracy rate of 86 percent with accuracy of predicting successful startup as 94 percent and unsuccessful startup as 69 percent.



```
         Pred:0  Pred:1
Actual:0      34      15
Actual:1       7     112
Percent Acquired correctly predicted:  94.11764705882352
Percent Not Acquired correctly predicted:  69.38775510204081
Accuracy of the test set:  0.8690476190476191
```

Fig 14. Accuracy rate of Gradient Boosting Algorithm

The below graph shows a comparison of accuracy rate of five machine learning algorithms

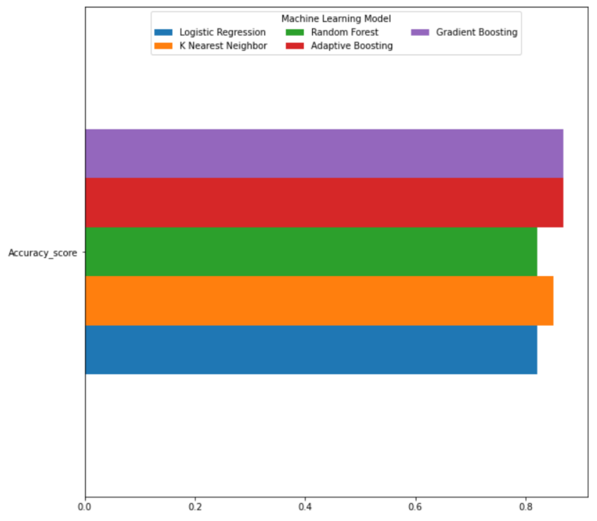


Fig 15. Comparison of Accuracy rate

V. Future Directions

In the future we would like to identify common patterns or traits between different successful startups and failed startups and predict whether a startup would be successful or not. We would also find the more accurate machine learning algorithm for prediction. In further stages we would recommend add more data and features to train the model. With more data we could achieve an even higher rate of accuracy which clearly identifying attributes contributing to the success and failure of a startup.

VI. Conclusion

After implementation all five machine learning algorithms, we noticed that boosting algorithms, adaptive boosting and gradient boosting gives us better prediction on our problem statement. The accuracy achieved with both the algorithms are 86 percent.

References

https://www.techtimes.com/articles/265116/20210908/this-ai-could-predict-startup-success-research-claims-it-has-90-accuracy.htm

https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2019/3878-2019.pdf

https://www.csie.ntu.edu.tw/ cjlin/papers/liblinear.pdf

https://hastie.su.domains/Papers/samme.pdf