Startup Status Prediction

Caitlin Whitter
cwhitter@purdue.edu
Purdue University
West Lafayette, IN, USA

Nikhil Gupta
gupta883@purdue.edu
Purdue University
West Lafayette, IN, USA

Shafkat Islam
islam59@purdue.edu
Purdue University
West Lafayette, IN, USA

Sachit Kothari
kothar13@purdue.edu
Purdue University
West Lafayette, IN, USA

ABSTRACT

With the advent of Machine Learning, retail investors and venture capital firms are using data and past trends to understand which startups will succeed and where they should invest their money. We aim to explore machine-learning methods to understand these questions based on the Kaggle 'Startup Investments' dataset [1]. This data has been taken from Crunchbase 2013 Snapshot up to December 2013.

CCS CONCEPTS

• **Data** → **Data mining**; *Data models*; Machine Learning; • **Startup** → Startup investment; Startup success prediction.

KEYWORDS

Data mining, Machine learning, Startup investment, Prediction, Supervised learning, Multi-layer Perceptron, Logistic Regression, Extreme Gradient Boosting, Random Forest

ACM Reference Format:
Caitlin Whitter, Nikhil Gupta, Shafkat Islam, and Sachit Kothari. 2018. Startup Status Prediction. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX).* ACM, New York, NY, USA, 9 pages. https://doi.org/XXXXXXX.XXXXXXX

1 PROJECT TOPIC

Several of today's biggest companies like Apple, Facebook, Tesla were once startups and had humble beginnings. Successful startups make big bucks. They make for great stories of hard work and persistence that often positively impact society while contributing to the economy and creating jobs. Shows like 'Shark Tank' show the lengths people go to pitch their products to investors. But startups fail. In fact, Startups fail too often. Forbes reports that nine out of ten startups will fail [9]. While this may seem like a pessimistic statistic, it becomes pertinent to understand which factors influence a startup's success. Is it the product-market fit, funding amount, the founding year/decade, the prestige of investors in the industry, or the qualifications of the founders? Which factors influence whether a startup succeeds and IPOs or is acquired or fails? This data-driven decision-making is becoming an integral part in funding decisions and Gartner reports that 75% of venture capital funds will use AI to make investment decisions by 2025. [11]

The end goal of our project is to predict whether a startup will IPO, will be acquired, closed, or will remain operational based on its features. We plan to use the Kaggle 'Startup Investments' dataset [1] consists of 11 tables containing data related to acquisitions, funding rounds, investments, IPOs, milestones, and people data, among other information, that can be joined with unique IDs.

2 MACHINE LEARNING PIPELINE

2.1 Literature Survey

In [3], the authors developed a deep learning based time series analysis method for investigating startup evaluations, which provides detailed characteristics of companies since the dataset contains data of different times. The authors in [8] developed a DNN model to predict the generated profit of a startup based on its R&D, administration, marketing, and state. The authors in [6] developed data mining models which can help startup founders to look at particular factors that require attention to hit the success mark. The authors in [12] advised VC/PE firms to utilize machine learning techniques on publicly available data to decide on funding. In [7] and [4], the authors provide a summary of recent academic studies and companies applying machine learning models to venture capital data.

2.2 Dataset

We collected the startup investment data from [1]. The tabular data consists of 11 different csv files along with different attributes. Based on the chosen features, we predict the status of startups. Categories for status include IPO, acquired, closed, and operating. The dataset has high class imbalance and contains 196,553 companies in total, of which 9394 are acquired, 2584 are closed, 1134 are IPO and 183441 are operating. Note that this distribution does not reflect real world data, where startups are more likely to close than remain operating. There is information up to December 2013 and is taken from the Crunchbase 2013 snapshot. The data consists of a number of things that could affect a business, such as funding rounds, the industry the business is in, information on its founders, headquarters location, important milestones and news articles about the business and several others.

2.3 Initial Data Preprocessing

As an initial pre-processing step, we parsed and converted the data from a tabular csv format to a pandas dataframe format. In doing so, we formatted our data in such a way as to be more amenable to our Exploratory Data Analysis and Feature Selection and Engineering steps.

2.4 Exploratory Data Analysis

The biggest challenge in the dataset [1] is the huge amount of data spread over 11 tables, each depicting a distinct aspect of the startup. We had these tables in comma separated values (csv) format:

- **acquisitions**: Contains information about startups that have been bought.
- **degrees**: Contains the education backgrounds of individuals involved in the startup world.
- **funding_rounds**: Contains information about startup funding rounds.
- **funds**: Contains data on the venture capital funds that make investments.
- **investments**: Contains data on the various different investments made by venture capitalists.
- **ipos**: Contains data on initial public offerings.
- **milestones**: Contains significant events within the startup ecosystem.
- **objects**: Main file containing base information regarding Person, Financial Organization, Company, and Product.
- **offices**: Contains information about startup company offices.
- **people**: Contains information about individuals in the startup world.
- **relationships**: Contains relationship data that links companies to individuals and their positions.

The dataset contains 196,553 companies in the objects table. We engineer the most important features from these tables and include them in the final feature matrix using table joins in pandas. The objective is to predict 'status' column.

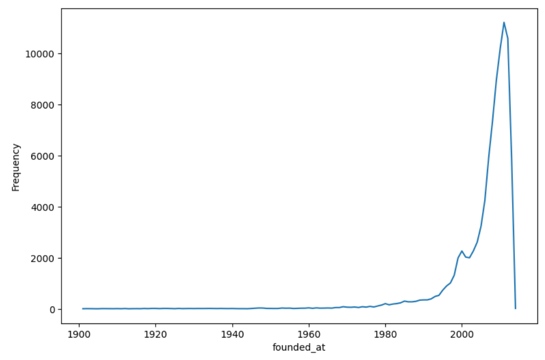

Figure 1: Time Series of founding year

The dip in frequency of founding year is data quality issue since we have data points in 2014 even though the data should be until 2013. We deal with this during data pre-processing.

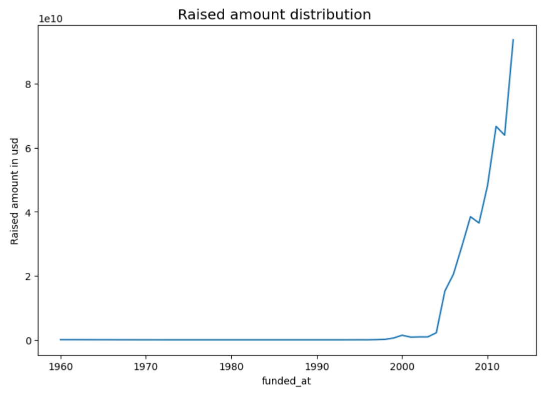

Figure 2: Time Series of funding round year

Both the time series plots show a significant jump in the recent years which aligns with the startup boom we have seen in the recent decades.

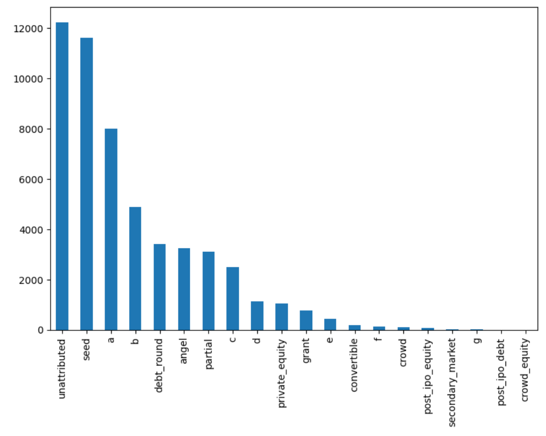

Figure 3: Frequency of funding rounds

The majority class is 'unattributed' which suggests missing values. Apart from that, most startups are funded only in the initial rounds which matches reality.

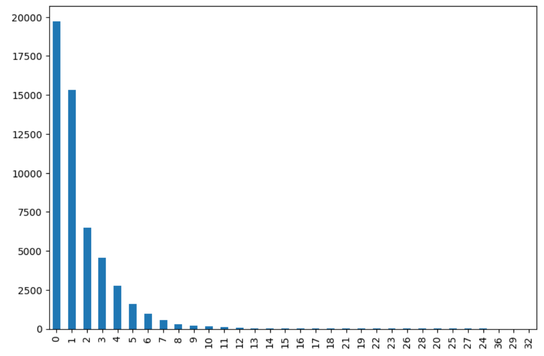
Figure 4: Frequency of participants in funding rounds

Participants = 0 implies missing values. Apart from that, we have a descending order of number of participants which makes sense.

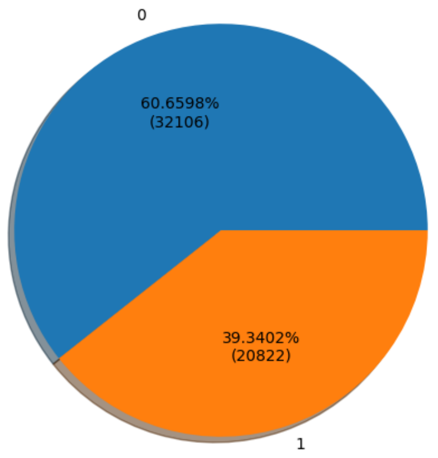

Figure 5: Pie chart of the boolean value of only one round

The pie chart shows that out of all the startups that are funded at least once, 39.3% startups are funded only once.

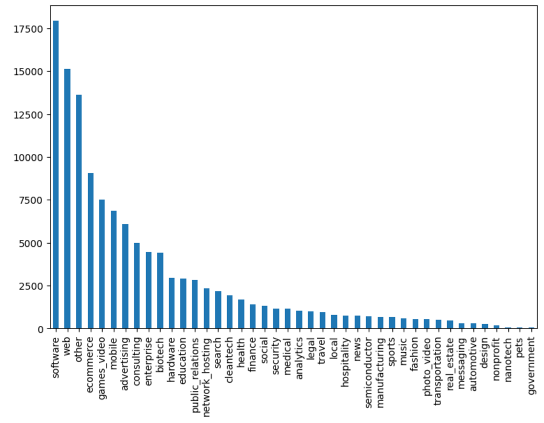
Figure 6: Industry distribution

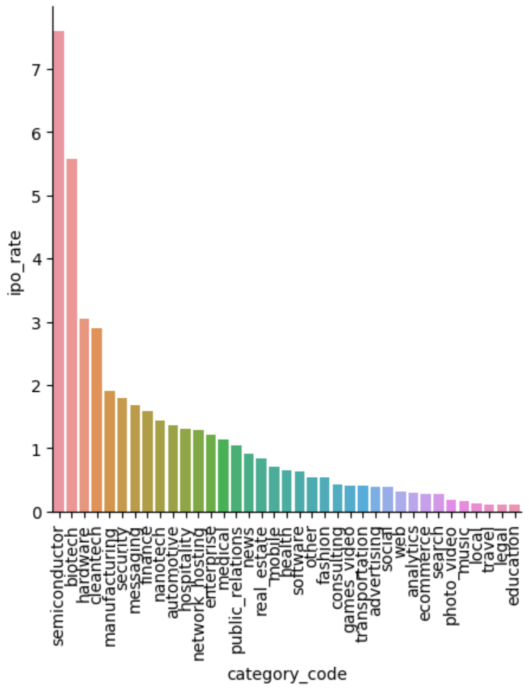

Figure 7: Industry-wise IPO rate

The IPO rate differs based on the industry which suggests industry affects the status.

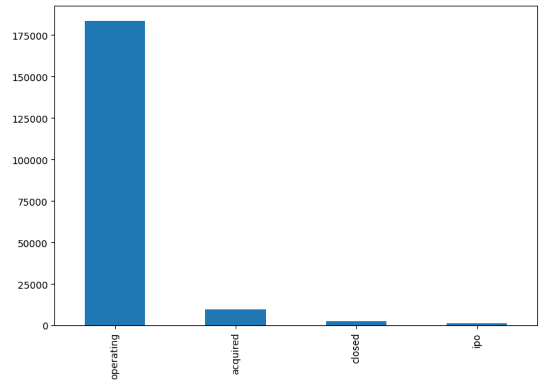

Figure 8: Distribution of status

We see that most of the startups are in the operating category. IPO has the least number of startups.

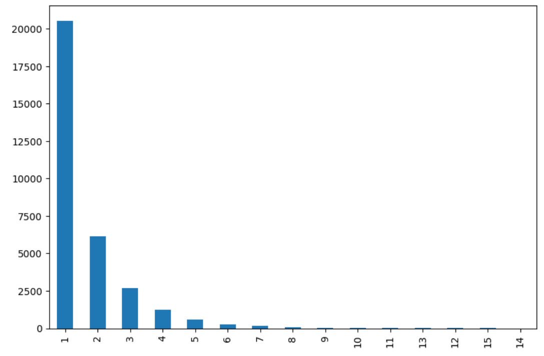

Figure 9: Frequency of funding rounds

The following boxplots show how different status categories have different distributions of the corresponding features. This suggests that these features are important in predicting the status.

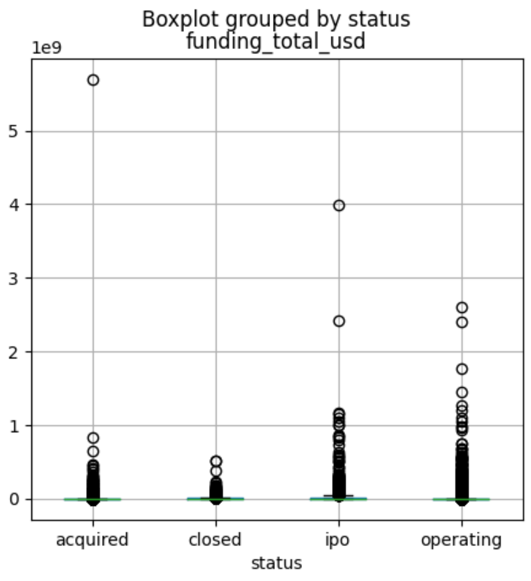

Figure 10: Boxplot of total funding

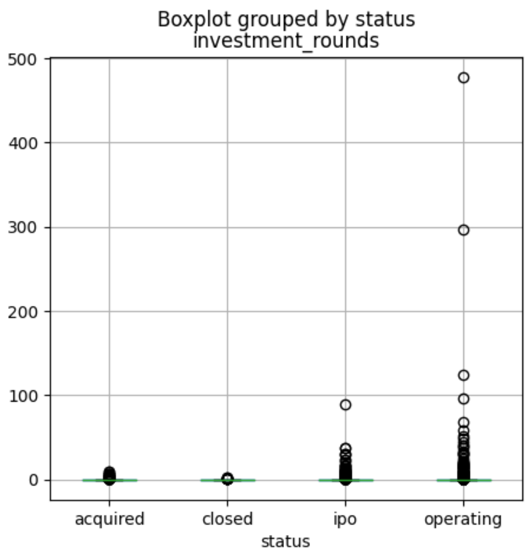

Figure 11: Boxplot of investment rounds

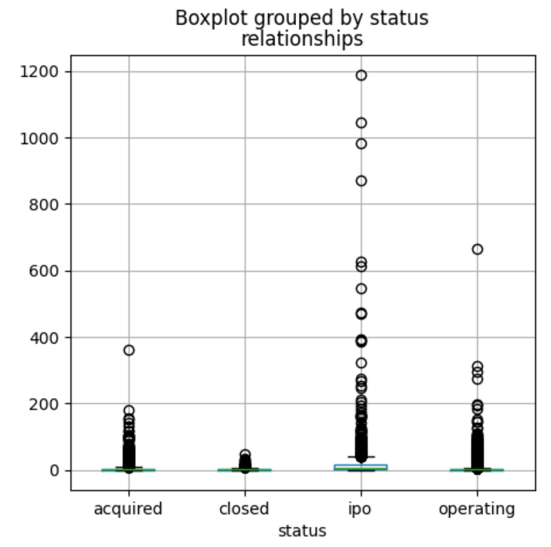

Figure 12: Boxplot of relationships

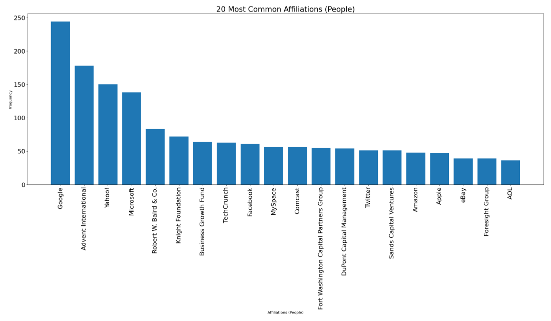

Figure 13: Distribution of affiliations

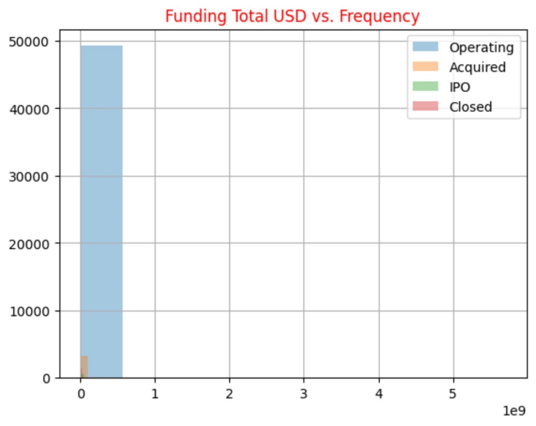

Figure 14: Funding total usd vs. data frequency for different classes

Figure 14 illustrates the distribution of funding total usd feature for different classes. From the figure we can infer that for different classes the feature value overlaps which makes it difficult for the classifiers to distinct between different data classes. We observe such behavior for many other features as well.

2.5 Feature Selection

The original dataset consisted of 11 tables, each containing up to 40 columns. As a result, there we many potential features to choose from, and we needed a way to narrow the field. We chose the following features due to their relevance and data abundance.

Final set of selected features:

- **Feature 1**: Industry
- **Feature 2**: Number of relationships
- **Feature 3**: Number of funding rounds
- **Feature 4**: Number of investment rounds
- **Feature 5**: Number of invested companies

2.6 Feature Engineering

In addition to the features we selected directly from the Kaggle dataset, we engineered new features based on the features that Kaggle provided. Our goal was to design new features that would assist in explaining the success/failure of a startup. Once again, we assessed these features based on relevance, data abundance.

We also chose features based on a lack of exact mathematical relation between them. For example, we noticed a positive correlation between two monetary features due to one feature being a linear combination of the other. As a result, we removed one of those features and kept the other. We based one of our engineered features, Feature 12, on this kept feature.

Final set of engineered features:

- **Feature 6**: Founding year
- **Feature 7**: Country code
- **Feature 8**: Number of products
- **Feature 9**: Number of positive news article descriptions
- **Feature 10**: Number of negative news article descriptions
- **Feature 11**: Number of neutral news article descriptions
- **Feature 12**: Log of total funding raised across all rounds
- **Feature 13**: Number of acquired companies
- **Feature 14**: Number of financial organizations invested in this company
- **Feature 15**: Number of companies invested in this company
- **Feature 16**: Number of people invested in this company

We computed three of our engineered features, Features 9-11, through performing sentiment analysis on the news article descriptions associated with each company, as given to us in the Kaggle dataset. For this purpose, we used *Financial-RoBERTa* [13], a pre-trained NLP model for performing sentiment analysis on financial text.

Example Financial-RoBERTa text classifications:

- **Positive**: Wetpaint named in Lead411's Hottest Seattle Companies list.
- **Negative**: GameLayers shuts down.

- **Neutral**: Yammer relocates headquarters from Los Angeles to San Francisco.

2.7 Data Preprocessing

Many of the potential features we chose had missing data for several companies within the dataset. (In the case of the features we engineered, many of the Kaggle columns our engineered features were based on had missing data.) To resolve this issue, we decided to remove companies with missing feature information for any of our chosen features. As a result, when choosing features, we had to balance including more features with reducing the number of companies in our dataset. In the future, we could try interpolating the missing values instead of removing the corresponding companies, which may allow us to include more features while keeping more companies in the dataset.

Due to the high dimensionality of our chosen categorical features, industry and country code, we decided to encode each unique value as a single number, instead of performing a one-hot encoding. Following some preliminary experimentation, we saw no worsening in performance when encoding the categorical features as a single number, but in the future we could perform more comprehensive studies.

2.8 Final Dataset

We built our final dataset through combining our selected and engineered features. We formed the train/test split on which our machine learning models would train and predict using stratified random sampling. We randomly sampled such that 85% of the data belonged to the training set and 15% to the test set. We performed stratified sampling due to our high class imbalance, so that the training and test sets would have matching class label distributions.

Training set size: 54,484 (85%)
Label count:

- **Operating**: 49,264 (90%)
- **Acquired**: 3,315 (6%)
- **IPO**: 1,385 (3%)
- **Closed**: 520 (1%)

Test set size: 9,615 (15%)
Label count:

- **Operating**: 8,694 (90%)
- **Acquired**: 585 (6%)
- **IPO**: 244 (3%)
- **Closed**: 92 (1%)

As a final data transformation step, we normalized all features so that they have mean zero and standard deviation one, in order to make the values more amenable to our machine learning models.

2.9 Methods: Machine Learning Models

The four models we chose were multi-layer perception, logistic regression, extreme gradient boosting, and random forests. We trained all of these models in a supervised setting. We also experimented with using an unsupervised method, but found its predictive power lacking for this task and dataset. For each model, prior to training and testing our final model, we performed hyperparameter tuning. Additionally, we experimented with different techniques to help resolve our class imbalances.

2.9.1 Multi-Layer Perceptron.

Multi-layer perceptron (MLP) is a fully connected feed-forward neural network [2]. Multi-layer perceptron is used for learning non-linear functions which is difficult to learn for single perceptron. In multi-layer perceptron there can be one or multiple hidden layers with arbitrary number of neurons in each layers. In this project we used single hidden layer with 100 neurons, and used Adam [5] based gradient descent technique as an optimizer for backpropagation. We used regularized linear unit (ReLu) as activation function, constant learning rate of 0.001 and maximum iteration size is 300. We used grid search technique to select the values of these hyperparameters.

2.9.2 Logistic Regression.

In logistic regression, a linear combination of input features is transformed into a probability value between 0 and 1 using a logistic function. The value indicates the chance that an event took place. Based on this chance and a certain threshold, the model decides if the event takes place. In this project, we use logistic regression to build a multi class classification model. The hyperparameters of interest are the regularization strength (C) and the penalty type (L1 or L2). The procedure uses the solver 'lbfgs' and multi-class is set to 'auto' to let the model pick the best approach for the classification. The, a grid search is performed with 10-fold cross-validation to identify the best hyperparameters, using the GridSearchCV function from sklearn.model_selection, with the model, hyperparameters, and cross-validation scheme specified. C is the regularization strength and is used to avoid over-fitting. Penalty represents the regularization type and can be set to either L1 or L2. L1 is used for feature selection by setting the weights of irrelevant features to zero. L2 is used to reduce the coefficients of irrelevant features. After the grid search, the best hyperparameters are retrieved using the best_params_ attribute of the grid_search object.
The final model is trained using the best hyperparameters and class weights. The weights are specified in a dictionary format where each class is assigned a weight that determines the contribution of the class to the loss function. The weights are 'operating': 1, 'acquired': 2, 'ipo': 5, 'closed': 10.

2.9.3 Extreme Gradient Boosting.

Extreme Gradient Boosting (or, XGBoost) is an optimized distributed implementation of Decision Trees using gradient boosting that is designed for speed and performance. The library supports various regularization techniques, Parallelization, Distributed Computing for training large datasets, Out-of-Core Computing for when dataset is too large to fit into memory, and Cache Optimization to optimize for the hardware. Furthermore, the algorithm handles missing values, sparse datasets, outliers to some extent, has in-built cross-validation, and supports continued training after the initial round of training.

In this project, we used the XGBoost implementation from the xgboost library. To choose the best hyperparameters for our random forest model, we performed a grid search on the training set. The tuned hyperparameters were as follows: gamma = 2, max_depth = 8, n_estimators = 10, reg_alpha = 40, and reg_lambda = 1.

To help resolve class imbalances, we undersampled the majority class "operating." There where 5220 data points in three non-majority classes: acquired, IPO, and closed. Grid search determined 8000 to be the best value for the number of data points in the operating category.

2.9.4 Random Forest.
A Random Forest Classifier is an ensemble method for classification tasks. It functions by constructing multiple decision trees during training and then taking an agglomeration of their predictions as the final result. A decision tree is a supervised, non-linear learning algorithm, which maps a path from a decision on a series of attributes to the final class label prediction.

In this project, we used the Random Forest Classifier implementation from the imbalanced learning library (v. 0.4). To choose the best hyperparameters for our random forest model, we performed a grid search on the training set with 10-fold cross-validation (scikit-learn v. 1.2.2). The first hyperparameter we tuned was n_estimators, the number of trees in the forest. The options were [50, 100, 150], and the selected best value was 100.

To help resolve class imbalances, we decided to undersample the majority class "operating." We also tried oversampling and assigning different weights to the classes, but undersampling performed best. Our second hyperparameter was the number of undersampled values to select randomly for the operating class. The options were [5220, 2*5220, 3*5220], where 5220 is the sum of the three non-majority classes: acquired, IPO, and closed. Grid search determined 2*5220 to be the best value.

We used the default settings for all other hyperparameters.

2.10 Results: Machine Learning Models

2.10.1 Multi-Layer Perceptron.
To analyze the performance of multi-layer perceptron (MLP) classifier, we present the performance of MLP on test dataset. The fine tuning of hyperparameters are conducted only on training dataset to avoid any kind of bias in the test set predictions. Figure 15 presents the result while under-sampling the majority class (during training), Figure 16 presents the result while over sampling the minority class (during training), and Figure 17 presents results without imposing any sampling technique on the training set. From the results it is evident that while under sampling the majority class makes the classifier more accurate for minority classes, over sampling the minority class is not very effective in comparison. However, we can achieve the best accuracy, F1-score and precision value for training set without using any sampling method. Though the accuracy on minority classes was very less in that case. We use scikit-learn 0.22 library and macro F1-score for the MLP classifier.

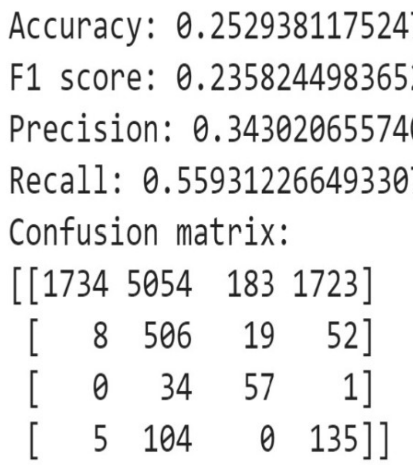
```
Accuracy: 0.25293811752470097
F1 score: 0.23582449836525665
Precision: 0.34302065574042023
Recall: 0.5593122664933072
Confusion matrix:
[[1734 5054  183 1723]
 [   8  506   19   52]
 [   0   34   57    1]
 [   5  104    0  135]]
```

Figure 15: Result using undersampling technique for MLP classifier

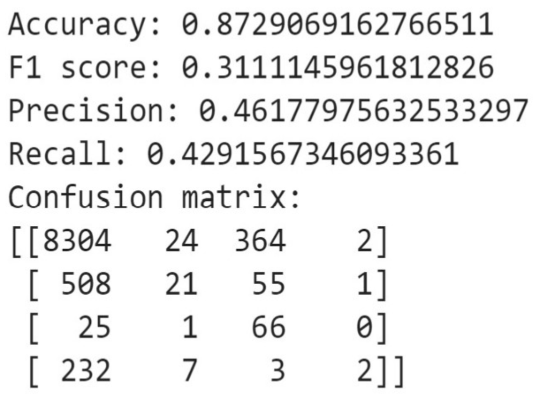
```
Accuracy: 0.8729069162766511
F1 score: 0.3111145961812826
Precision: 0.46177975632533297
Recall: 0.4291567346093361
Confusion matrix:
[[8304   24  364    2]
 [ 508   21   55    1]
 [  25    1   66    0]
 [ 232    7    3    2]]
```

Figure 16: Result using oversampling technique for MLP classifier

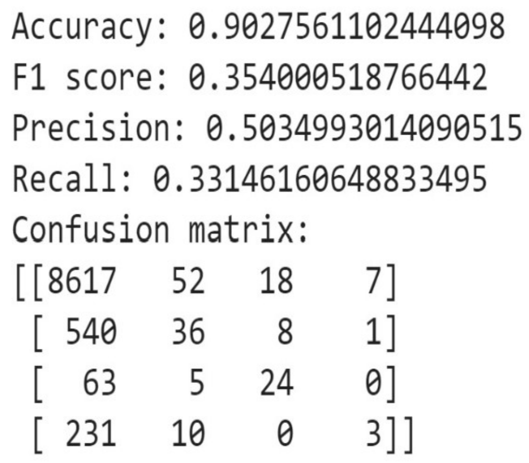
```
Accuracy: 0.9027561102444098
F1 score: 0.354000518766442
Precision: 0.5034993014090515
Recall: 0.33146160648833495
Confusion matrix:
[[8617   52   18    7]
 [ 540   36    8    1]
 [  63    5   24    0]
 [ 231   10    0    3]]
```

Figure 17: Result without using any sampling technique for MLP classifier

2.10.2 Logistic Regression.
The performance of the model is evaluated on the test set using accuracy, F1 score, precision, recall, and the confusion matrix. The average parameter is set to 'macro', which means that the metrics are computed for each class and then averaged. Confusion matrix is needed due to large class imbalance. The final results are obtained using class weights, as under sampling gave terrible results and without under sampling or weights the results were not as good as when using class weights. The final results we obtained are displayed in figure 18. Compared to the other Kaggle notebook that has performed a similar study and used multinomial logistic regression [10] , our accuracy is significantly higher than their accuracy when they use all features and when they use only significant features. However our F1 score is about the same as the notebook when all features are used, and when they use only significant features we get slightly worse f1 score.

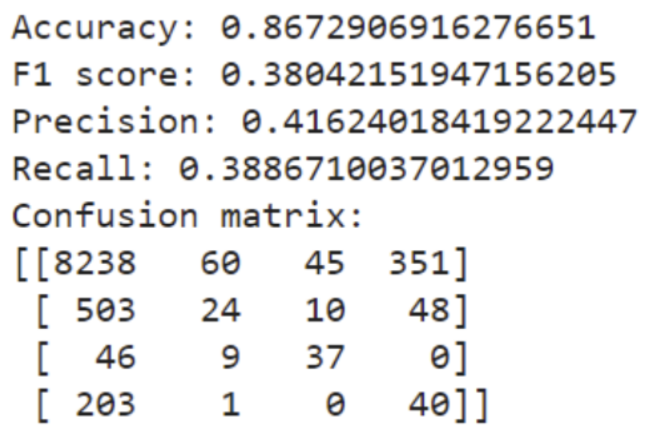
```
Accuracy: 0.8672906916276651
F1 score: 0.38042151947156205
Precision: 0.41624018419222447
Recall: 0.3886710037012959
Confusion matrix:
[[8238   60   45  351]
 [ 503   24   10   48]
 [  46    9   37    0]
 [ 203    1    0   40]]
```

Figure 18: Results for logistic regression with class weights

2.10.3 Extreme Gradient Boosting.
Following hyperparameter selection, we fit ten XGBoost models with undersampling on the training set, taking the mode of the predictions on the test set as the final prediction. Figure 19 displays the final results on the test set. The F1 score, Precision, and Recall were computing using the 'macro' average parameter from scikit-learn.

2.10.4 Random Forest.
Following hyperparameter selection, we fit ten Random Forest models with undersampling on the training set, taking the mode of the predictions on the test set as the final prediction. Figure 20 displays the final results on the test set. The F1 score, Precision, and Recall were computing using the 'macro' average parameter from scikit-learn.

Another Kaggle notebook [10] that performed a similar study to ours reported better results for Random Forest. In addition to future work discussed in the Conclusions section, modifying our sampling and averaging technique to what is described in the other Kaggle notebook might further improve our results.

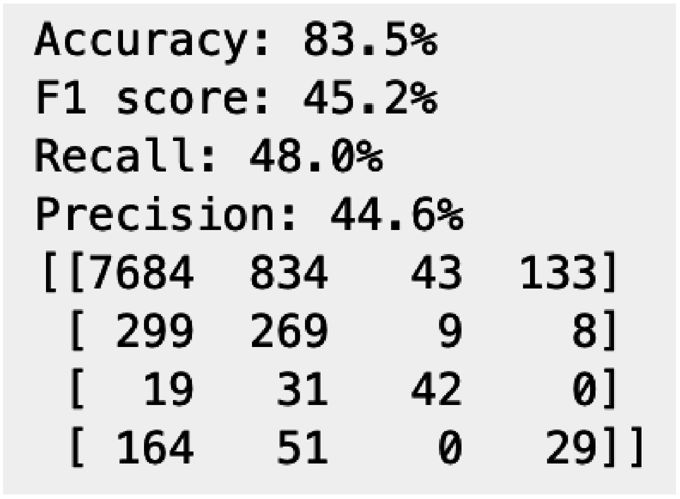
```
Accuracy: 83.5%
F1 score: 45.2%
Recall: 48.0%
Precision: 44.6%
[[7684  834   43  133]
 [ 299  269    9    8]
 [  19   31   42    0]
 [ 164   51    0   29]]
```

Figure 19: Results using undersampling for XGBoost

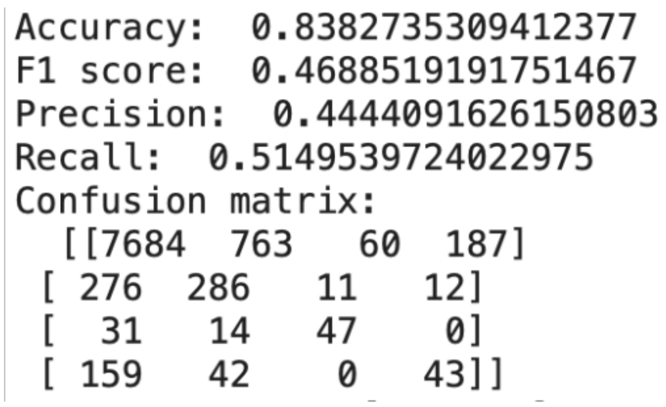
```
Accuracy:  0.8382735309412377
F1 score:  0.4688519191751467
Precision:  0.4444091626150803
Recall:  0.5149539724022975
Confusion matrix:
  [[7684  763   60  187]
 [ 276  286   11   12]
 [  31   14   47    0]
 [ 159   42    0   43]]
```

Figure 20: Results using undersampling for random forest

3 CONCLUSIONS

We found our best performing model to be random forest with an F1 score of 0.468. F1 score is used as the measure for the best model as there is heavy class imbalance and all the models have similar accuracy. XGBoost is close behind with 0.452 F1 score. Logistic regression and multi layer perceptron did not perform as good as random forest. This is likely because random forest and XGBoost are ensemble models. Another possible cause is that the random forest and XGBoost were run 10 times and the mode of the findings was taken. The 5 best features were picked based on ANOVA F-value to be acquired companies (1482.77639764), relationships (1338.94758499), founding year (801.40120767), milestones 678.59251676) and log_funding_total_usd (500.20285472) in decreasing order of importance. The importance of these features can also be understood intuitively, as they have obvious effect on the performance of a business.
At the end of this project we have created and tested most of the

models outlined in the plan and successfully evaluated their performance with appropriate metrics. The project work also largely stayed on the planned timeline.
In the future, we will collect data samples which are more accurate representative of the true underlying distribution. Moreover, we observe that many of the features have overlapping values for different classes. This situation may cause issue for any learning technique. Hence we will try to find a solution to avoid this scenario. Finally, we will try other machine learning models to find the best performance for this dataset.

4 TEAM MEMBER CONTRIBUTIONS

We divided the work evenly, and every team member was a positive contributor to the group.

Task Specification: Nikhil (mostly), Shafkat, Caitlin, Sachit
Data Pre-processing: Caitlin
Literature Survey: Shafkat
Exploratory Data Analysis: Nikhil (mostly), Shafkat, Caitlin, Sachit
Feature Selection: Caitlin, Sachit
Feature Engineering: Nikhil, Caitlin
Model Selection, Tuning, and Testing:

(1) Shafkat (Multi-Layer Perceptron)
(2) Sachit (Logistic Regression)
(3) Nikhil (Extreme Gradient Boosting)
(4) Caitlin (Random Forest)

Project Proposal Writing: Caitlin, Nikhil, Shafkat, Sachit
Project Final Report Writing: Caitlin, Nikhil, Shafkat, Sachit

REFERENCES

[1] Justinas Cirtautas. 2019. Startup Investment. https://www.kaggle.com/datasets/justinas/startup-investments?select=objects.csv
[2] Meha Desai and Manan Shah. 2021. An anatomization on breast cancer detection and diagnosis employing multi-layer perceptron neural network (MLP) and Convolutional neural network (CNN). *Clinical eHealth* 4 (2021), 1–11.
[3] Francesco Ferrati, Haiquan Chen, and Moreno Muffatto. 2021. A deep learning model for startups evaluation using time series analysis. In *European Conference on Innovation and Entrepreneurship*. Academic Conferences International Limited, 311.
[4] Peter Foy. 2022. Applications of AI and Machine Learning in Venture Capital. https://www.mlq.ai/ai-machine-learning-venture-capital/
[5] Imran Khan Mohd Jais, Amelia Ritahani Ismail, and Syed Qamrun Nisa. 2019. Adam optimization algorithm for wide and deep neural network. *Knowledge Engineering and Data Science* 2, 1 (2019), 41–46.
[6] Amar Krishna, Ankit Agrawal, and Alok Choudhary. 2016. Predicting the outcome of startups: less failure, more success. In *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*. IEEE, 798–805.
[7] João Nunes. 2022. Technical perspective-machine learning models. https://medium.com/@JFVNunes/technical-perspective-%EF%B8%8F-machine-learning-models-422f9b8db810
[8] Kanhaiya Pandey, Rahul Sharma, Swapnali Shinde, Samiksha Ghuge, and S.R Patil. 2022. Start-up profit prediction. *International Journal of Advanced Research in Computer and Communication Engineering* 11, 5 (May 2022), 320–325. https://doi.org/wp-content/uploads/2022/05/IJARCCE.2022.11561.pdf
[9] Neil Patel. 2015. 90% Of Startups Fail: Here's What You Need To Know About The 10%. https://www.forbes.com/sites/neilpatel/2015/01/16/90-of-startups-will-fail-heres-what-you-need-to-know-about-the-10/
[10] Fabio Polcari. 2021. Startup success prediction. https://www.kaggle.com/code/fpolcari/startup-success-prediction
[11] Meghan Rimol and Katie Costello. 2021. VCs decision-making. https://www.gartner.com/en/newsroom/press-releases/2021-03-10-gartner-says-tech-investors-will-prioritize-data-science-and-artificial-intelligence-above-gut-feel-for-investment-decisions-by-20250
[12] Greg Ross, Sanjiv Das, Daniel Sciro, and Hussain Raza. 2021. CapitalVX: a machine learning model for startup selection and exit prediction. *The Journal of Finance and Data Science* 7 (2021), 94–114.
[13] Mohammad Soleimanian. 2022. Financial-RoBERTa. https://huggingface.co/soleimanian/financial-roberta-large-sentiment