Predicting a startup's Acquisition Status

Team Members:

Shridhar

Doleshor

Damilola

Melroy

Aman

Pragati

Table of Contents

Introduction

The goal of this project is to predict startup acquistion status based on a comapnys financial statistics. While the area of using machine learning to predict IPO under pricing has neen well researched,the topic has beeen surprisingly understudied.

The resulting algorithm takes in a startups financial statistics such as funding dollars, active days number of funding rounds and location.The main challenge for this problem is dealing with an imbalance dataset where one class is overrepresented, but under/oversampling cannot be used as a technique to balance the data. In order to address this, an ensemble-based technique that can combine the results of a high precision anomaly detection algorithm with a random forest classifier

Abstract

The goal of this study is to determine whether a startup is active or not. This information system was developed using the SDLC (Systems Development Life Cycle), which has four stages: planning, analysis, design, and implementation. It is primarily intended to forecast results. The major goal is to create a model that can anticipate the startup's financial health. However, a large section of the research focuses on project success or failure probabilities over time, taking into account temporally dynamic data like the amount raised, funding rounds, amount funded and active days.

Data collection

The data is scraped from Crunchbase-2013 which contain 196,553 rows and 44 columns. Each row contains company's status (Operating, IPO, Acquired, Closed). Dataset provides company name, category, country, founding rounds, founding total usd, milestone etc.

Data preprocessing

Dataset needed to be preprocessed before modelling was done.

- Features of the Data: -

  'id', 'Unnamed: 0.1', 'entity_type', 'entity_id', 'parent_id', 'name', 'normalized name', 'permalink', 'category code', 'status', 'founded at', 'closed at', 'domain', 'homepage URL', 'twitter username', 'logo URL', 'logo width', 'logo height', 'short description', 'description', 'overview', 'tag list', 'country code', 'state code', 'city', 'region', 'first_investment_at', 'last_investment_at', 'investment rounds', 'invested companies', 'first_funding_at', 'last_funding_at', 'funding_rounds', 'funding_total_usd', 'first_milestone_at', 'last_milestone_at', 'milestones', 'relationships', 'created by', 'created at', 'updated at', 'lat', 'lng', 'ROI'

- Removal of all the Unwanted Columns.
  As we can clearly see there were many irrelevant and reductant columns in the dataset which needed to be removed like -'domain', 'homepage URL', 'twitter_username''logo URL', 'logo width', 'logo height', 'short description', 'description', 'overview', 'tag list', 'name', 'normalized name', 'permalink', 'invested companies'. These are all the irrelevant columns

  There were columns with N/A values more than 96% which were also dropped. This was Done due to the fact that those columns will not help regarding the modelling and would only hinder the overall process.

- Outliers.
  Removal of all the outliers in ‘funding rounds’ and ‘funding_total_usd’ as they can

Be clearly seen in the boxplot. IQR Method was used in order to drop them.

- Generalizing the categorical data.
  Categorical dataset was present in the category code and country code columns. Category code feature contained 42 categories, but only the first 15 were kept and others were named as other. Similarly for country code column, first 10 was kept with names and others as other.
  Now One Hot Encoder was used in these columns to convert them into categorical Dataset.
- Creating new features 'is close' and 'active days'.
  New feature 'Is Close' was created from 'closed at' and 'status'.
  - If the value in 'status' is 'operating' or 'ipo', 1 was given to the 'Is Close' column.
  - Where as if the value is 'acquired' or 'closed', 0 was given to the 'Is Close' column.

  New feature 'active days' was also created from status and 'closed at' columns
  - If the value in status is 'operating' or 'ipo', put 2021 in 'closed at'.
  - Where as if the value is 'acquired' or 'closed, put 0.
  - Now getting the 'active days' column from 'founded date' and 'closed at'.

- Filling Null values with Mean Value in Numerical data.
  In all the columns with numerical dataset, Null values were filled with the mean value of that feature.

Modeling

We define our task as a binary classification problem where we predict whether a startup project will succeed (1) or fail(0)

Logistic Regression

Logistic Regression is commonly used to estimate the probability that an instance belongs to a particular class (e.g., what is the probability that this startup company will fail?). If the estimated probability is greater than 50%, 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.

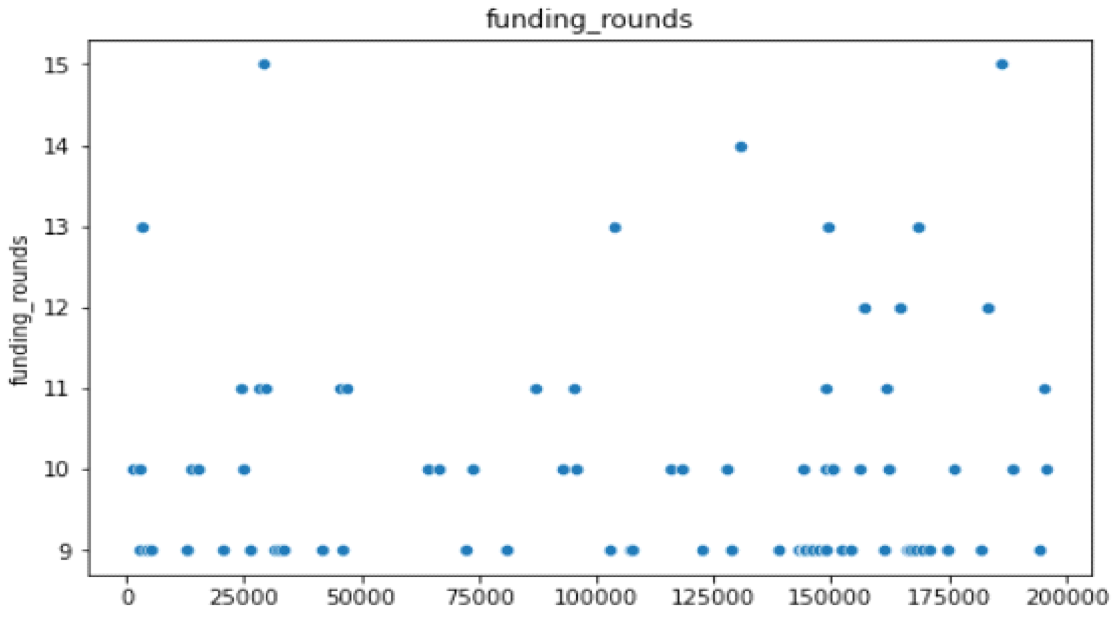
XGBoost

XGBoost is an efficient implementation of gradient boosting that can be used for regression predictive modeling.In this tutorial, you will learn how to build and optimize models with gradient boosting. This method dominates many Kaggle competitions and achieves state-of-the-art results on a variety of datasets.

Gradient boosting is a method that goes through cycles to iteratively add models into an ensemble.It begins by initializing the ensemble with a single model, whose predictions can be pretty naive. (Even if its predictions are wildly inaccurate, subsequent additions to the ensemble will address those errors.)
Then, we start the cycle:
First, we use the current ensemble to generate predictions for each observation in the dataset. To make a prediction, we add the predictions from all models in the ensemble.
These predictions are used to calculate a loss function (like mean squared error, for instance).Then, we use the loss function to fit a new model that will be added to the ensemble. Specifically, we determine model parameters so that adding this new model to the ensemble will reduce the loss. (Side note: The "gradient" in "gradient boosting" refers to the fact that we'll use gradient descent on the loss function to determine the parameters in this new model.)
Finally, we add the new model to ensemble, and ...... repeat!

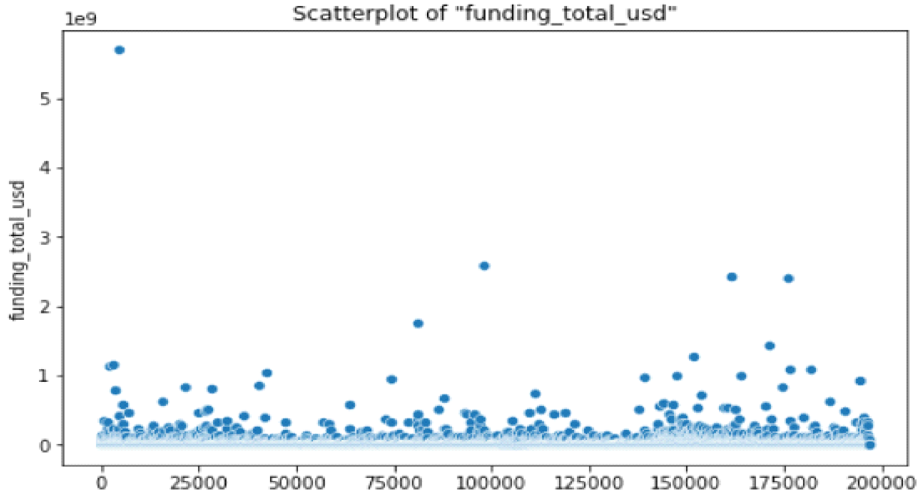
Data Visualizations

```
# ploting datas with values of 'funding_rounds'>8 as it will be easy to spot outl
funding_round = data_df[data_df['funding_rounds']>8]['funding_rounds']
plt.title('funding_rounds')
sns.scatterplot(x = funding_round.index, y =funding_round)
plt.show()
```



- funding_total_usd

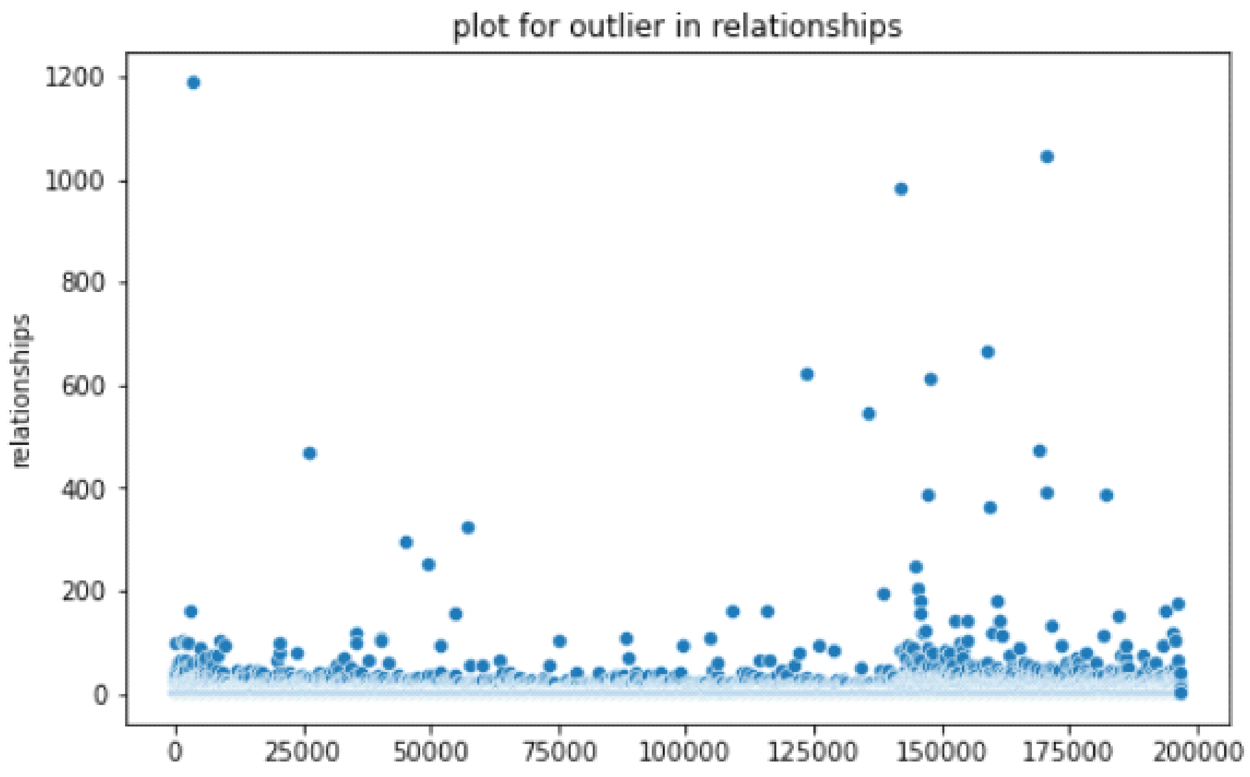
```
]: # plotting funding_total_usd column
sns.scatterplot(x=data_df.index,y=data_df.funding_total_usd)
plt.title('Scatterplot of "funding_total_usd"')
plt.show()
```



We can clearly see outlier data above 2e+9 in the plot. So now removing outliers from both columns as,

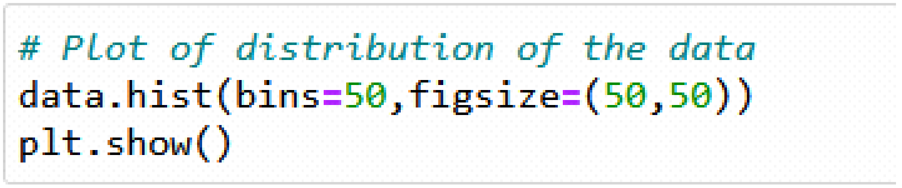
- relationships

```
# Scatterplot of relationship column
sns.scatterplot(x=data_df.index,y=data_df.relationships)
plt.title('plot for outlier in relationships')
plt.show()
```

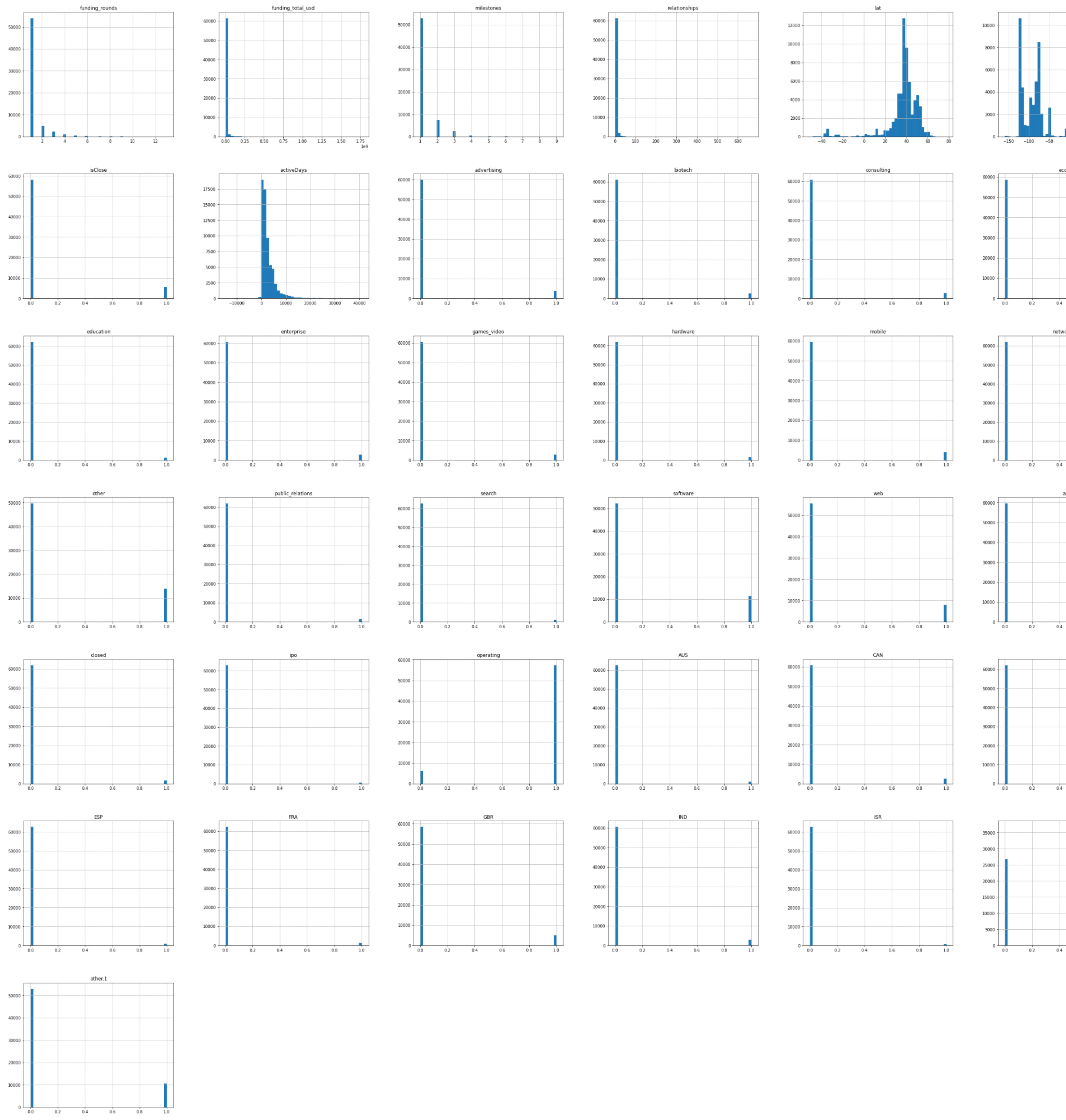


Here there are three outliers which are above 800 values. So now removing outliers as,

#Checking the normalization of the data



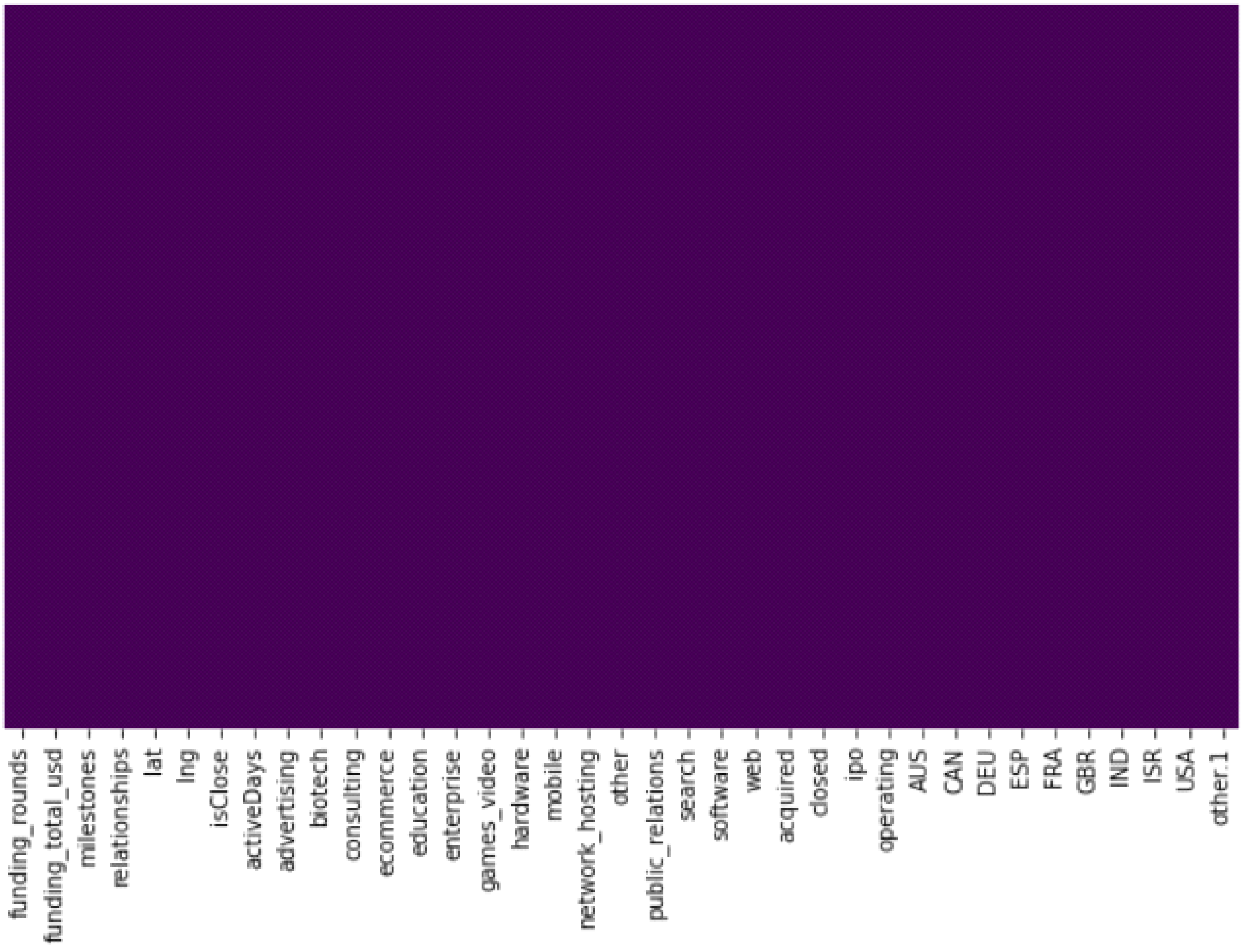
```
# Plot of distribution of the data
data.hist(bins=50,figsize=(50,50))
plt.show()
```


funding_rounds
funding_total_usd
milestones
relationships
lat
isClose
activeDays
advertising
biotech
consulting
eco
education
enterprise
games_video
hardware
mobile
netwo
other
public_relations
search
software
web
a
closed
ipo
operating
AUS
CAN
ESP
FRA
GBR
IND
ISR
other.1

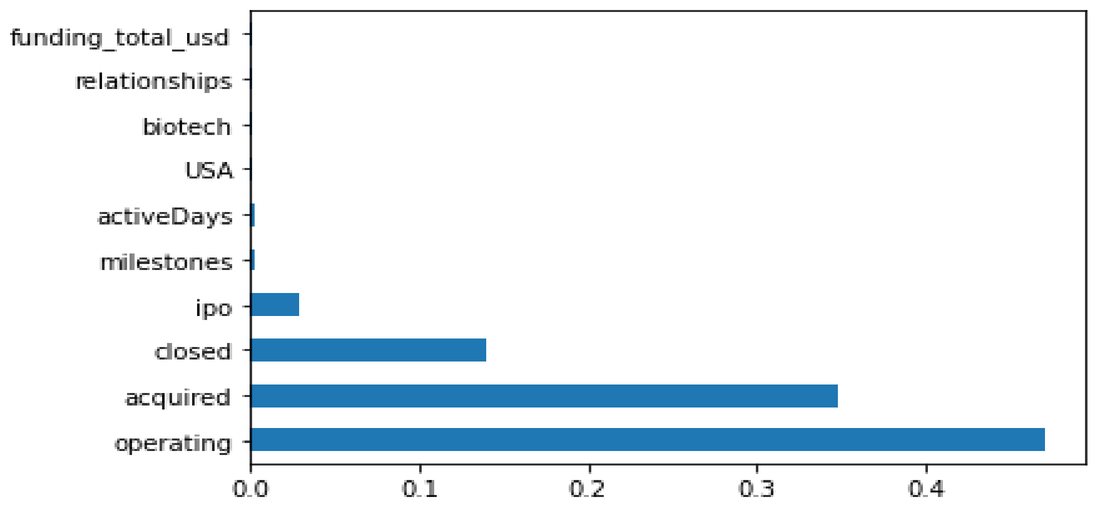
Checking the Missing values

```
plt.figure(figsize=(10,6))
sns.heatmap(data.isna(), cbar=False, cmap='viridis', yticklabels=False)
```

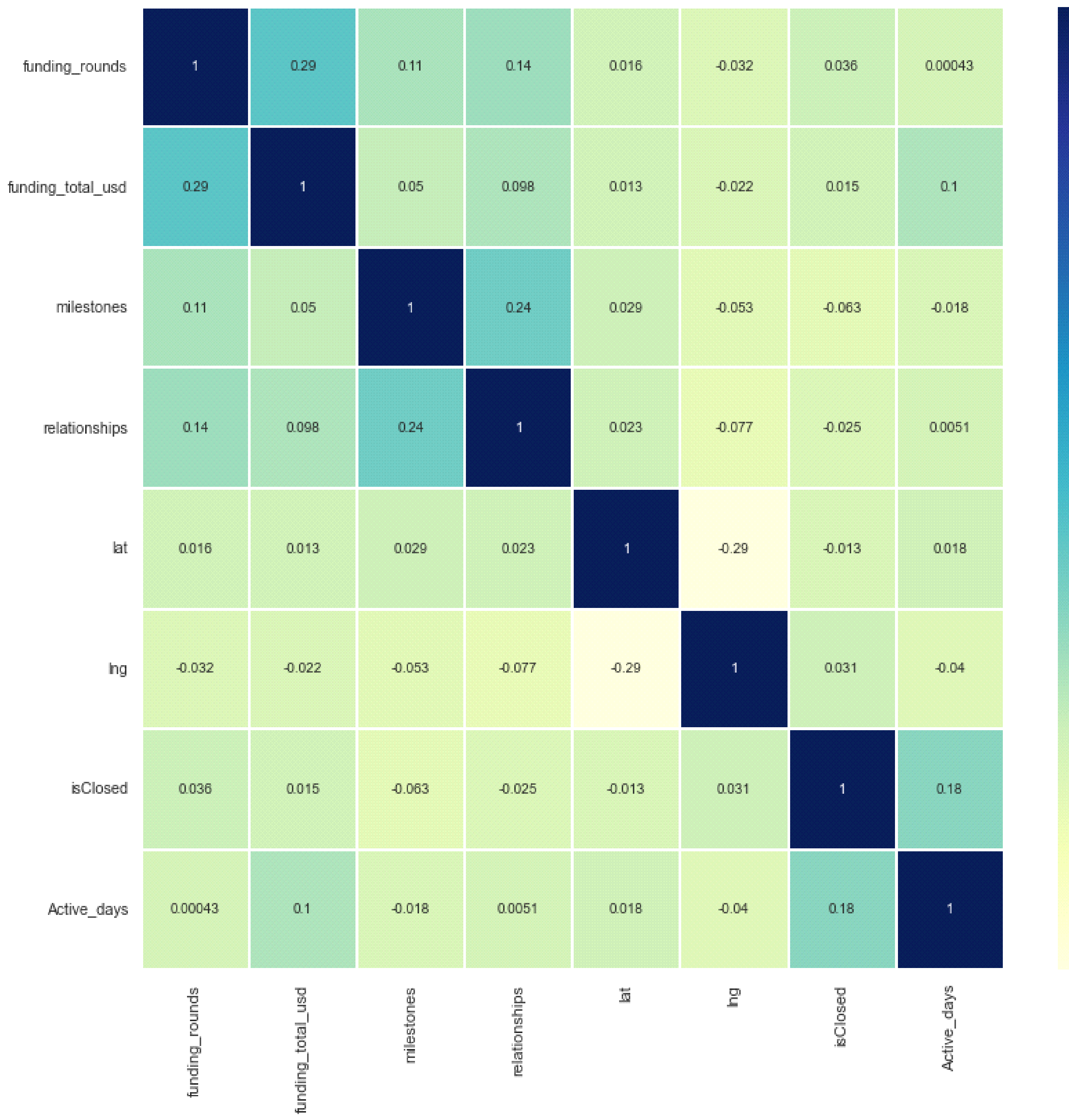
```
<AxesSubplot:>
```

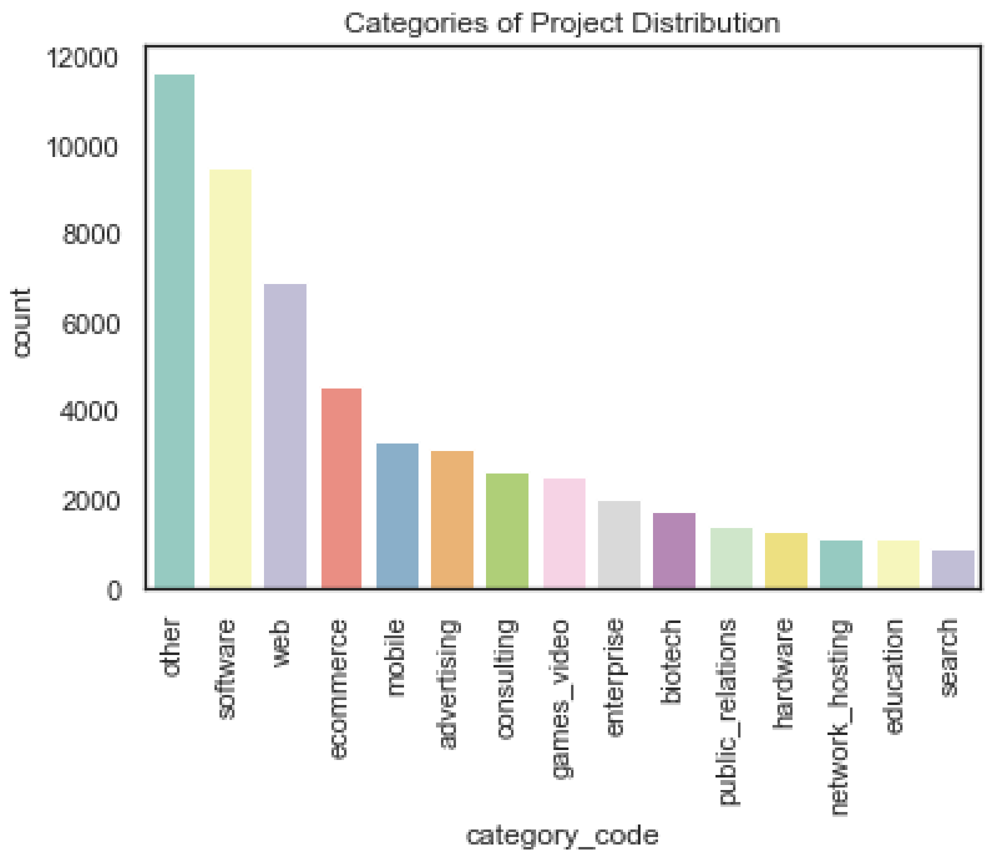


Feature Importance:



Correlation:

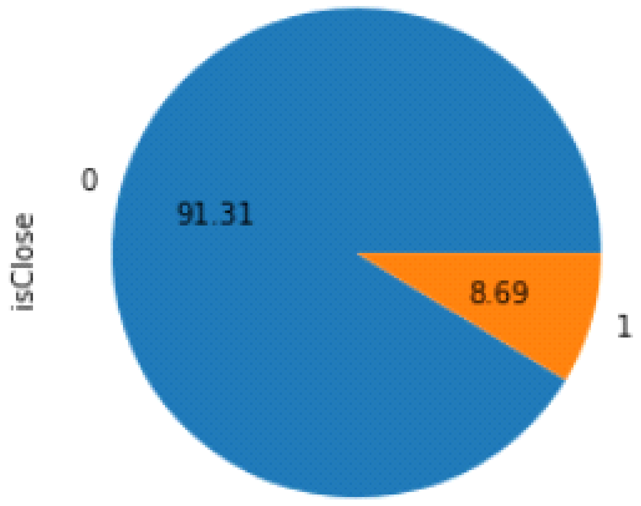

funding_rounds
funding_total_usd
milestones
relationships
lat
lng
isClosed
Active_days
1 0.29 0.11 0.14 0.016 -0.032 0.036 0.00043
0.29 1 0.05 0.098 0.013 -0.022 0.015 0.1
0.11 0.05 1 0.24 0.029 -0.053 -0.063 -0.018
0.14 0.098 0.24 1 0.023 -0.077 -0.025 0.0051
0.016 0.013 0.029 0.023 1 -0.29 -0.013 0.018
-0.032 -0.022 -0.053 -0.077 -0.29 1 0.031 -0.04
0.036 0.015 -0.063 -0.025 -0.013 0.031 1 0.18
0.00043 0.1 -0.018 0.0051 0.018 -0.04 0.18 1
funding_rounds
funding_total_usd
milestones
relationships
lat
lng
isClosed
Active_days



Data Imbalance

```
# checking the data imbalance

    ax= data.isClose.value_counts().plot.pie(autopct='%.2f')
```



What is the issue with imbalanced dataset?

Most models trained on imbalanced data will have a bias towards predicting the larger class(es) and, in many cases, may ignore the smaller class(es) altogether.

As a result, the instances belonging to the smaller class(es) are typically misclassified mo

often than those belonging to the larger class(es)

How to deal with imbalance dataset?

Resample the training set

Oversampling — Duplicating samples from the minority class

Undersampling — Deleting samples from the majority class

Combining Both Random Sampling Techniques

Combining both random sampling methods can occasionally result in overall improved performance in compa
to the methods being performed in isolation.

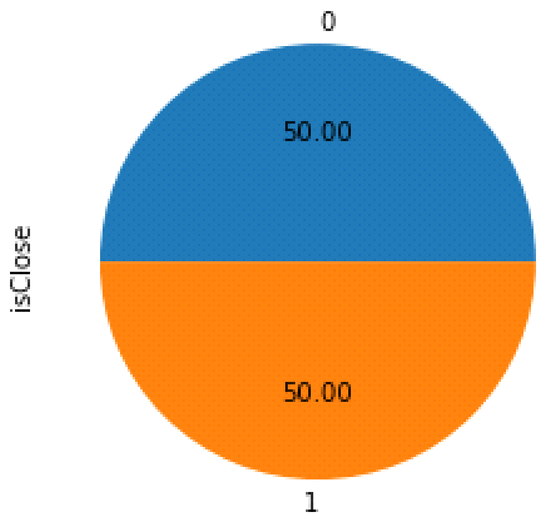
:
```
#Sampling technique to balance the data
#Over sampling
```

:
```
from imblearn.under_sampling import RandomUnderSampler
from imblearn.over_sampling import RandomOverSampler
from collections import Counter
from imblearn.over_sampling import SMOTE
from sklearn.metrics import confusion_matrix
```

:
```
oversample = RandomOverSampler(sampling_strategy='minority')
Undersample = RandomUnderSampler(sampling_strategy='majority')
```

:
```
# fit and apply the transform
X_train_over, y_train_over = oversample.fit_resample(X,y)
print("After oversampling: ",Counter(y_train_over))
# summarize class distribution
print("After oversampling: ",Counter(y_train_over))
from collections import Counter
print(sorted(Counter(y_train_over).items()))
```

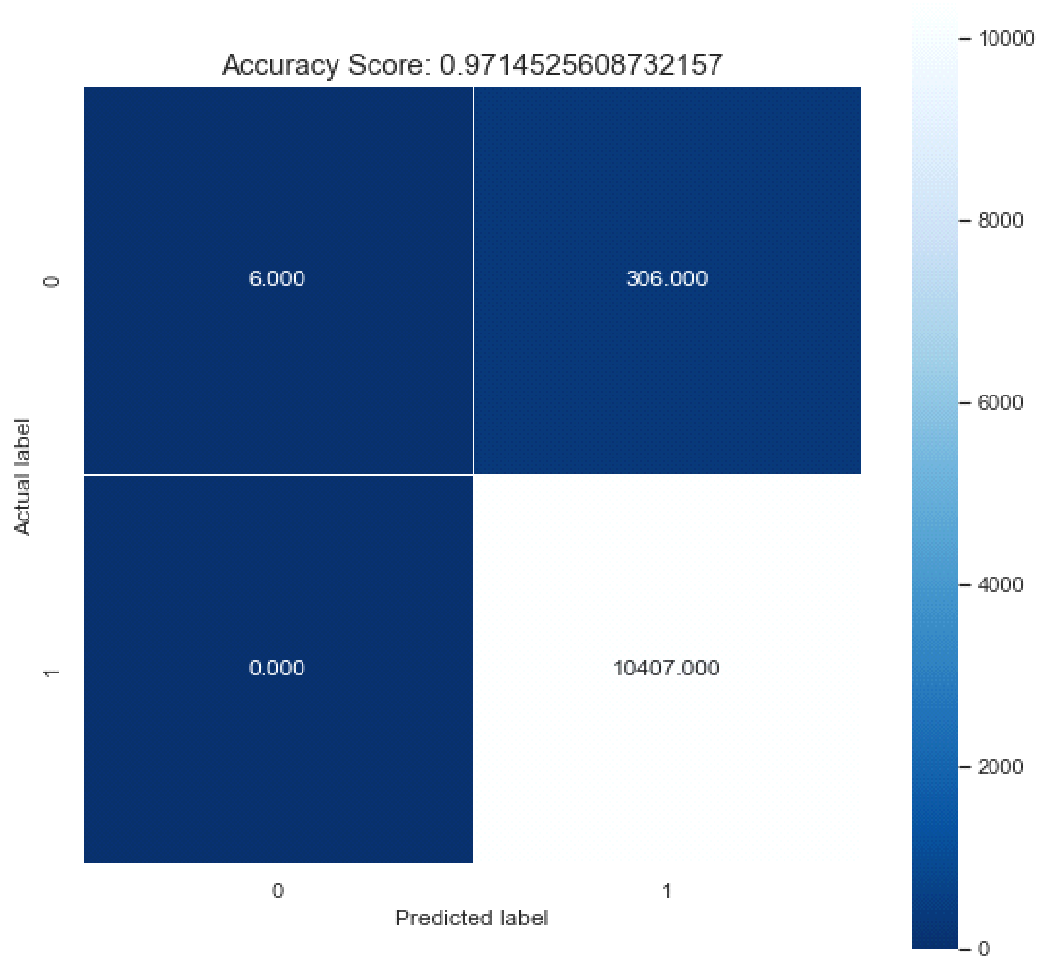
```
After oversampling:  Counter({0: 58050, 1: 58050})
After oversampling:  Counter({0: 58050, 1: 58050})
[(0, 58050), (1, 58050)]
```



Results and Discussions:

Model 1: Logistic Regression

Logistic regression was used in the biological sciences in the early twentieth century. It was then used in many social science applications. Logistic Regression is used when the dependent variable(target) is categorical. LR increases precision, But overfits the training data, sometimes to the detriment of accuracy. Combining a high precision model allows us to increase precision without decreasing accuracy. Precision increase is statistically significant, but not very large. A more effective approach for this problem would be to focus on how LR can be tuned to generalise better through more effective over/under-sampling techniques. LR models are high variance and dependent on the output of the classifier. We examine how tuning one model's parameters and features affect the others. The technique is not limited to LR. We can explore how other models can be combined using this technique. The practical aspect of this approach made it a very suitable one for at least smaller datasets. **Accuracy 97%**



Model 2: XGBoost

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solves many data science problems in a fast and accurate way. The same code runs on major distributed environments (Hadoop, SGE, MPI) and can solve problems beyond billions of examples. The second approach is validated based on CV scores. In this approach, we addressed the problem through a classical binary classification model. We ran Logistic Regression as well to classify the startup operating or not. We divided the dataset into train/test and did up to 10-fold cross-validations on each of the models. We generated the R2 Error values and compared them. The cross-validated scores were used to validate the errors generated. Also, we used cross-validation to get the optimal model. Overall after the comparisons, we found that the XGBoost has an **accuracy of ~0.91** on the test set which makes it a stand out model. Since, we did the cross-validations, the risk of overfitting the model is also reduced. All the scores and measures are documented well in the jupyter notebooks.

Conclusion:

We find that the logistic regression provides us with our best results, consistently across samples. XGBoost also performed very well, though were computationally significantly more expensive. The size of the dataset is likely a regression model performing up to par with some of the other, more robust models.

Our best performing features were the categorical mean encodings, underscoring the importance of feature engineering. While we experimented with more aggressive feature selection, we did not exhibit any serious signs of overfitting, as they generalized extremely well to the test set. One potential explanation for this is that our training dataset is large.

Reference:

1.https://flask.palletsprojects.com/en/2.1.x/tutorial/

2.https://www.askpython.com/python/examples/k-fold-cross-validation

3.https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6

4.https://machinelearningmastery.com/repeated-k-fold-cross-validation-with-python/

5.https://machinelearningknowledge.ai/python-sklearn-logistic-regression-tutorial-with-example

6.https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76

7.https://www.analyticsvidhya.com/blog/2021/03/step-by-step-process-of-feature-engineering-for-machine-learning-algorithms-in-data-science/

8. https://www.projectpro.io/article/8-feature-engineering-techniques-for-machine-learning/423

9. https://www.kaggle.com/code/alexisbcook/xgboost/tutorial