"To predict whether a team/company will reach its fundraising goal successfully through the ICO, using various machine learning models."

1 Introduction

Crowdfunding has the potential to democratize opportunities and resources, enabling people to join together and support projects they believe in. It has transformed startup ecosystems providing entrepreneurs with unprecedented access to capital. As opposed to more conventional approaches, crowdfunding uses the internet's ability to link fundraising teams with a large number of individual investors. It has opened a pathway to equal opportunities in a form that no matter what your background is if your idea has potential, you might be able to find someone somewhere in the world who can appreciate its worth. Entrepreneurs no longer must rely on traditional venture capitalists or angel investors for their capital demand and can now pursue their ideas with greater autonomy and flexibility. In addition to giving entrepreneurs the money they need; successful crowdfunding campaigns also demonstrate the value of the concepts. The capacity to muster backing from a network of supporters indicates investment confidence and market demand.
Financing for projects has been revolutionised because of Initial Coin Offerings (ICOs), which brought forth the idea of digital coins built on blockchain technology as a financing mechanism. This makes investing possibilities available to a far wider range of people, maybe drawing money from sources that previously might not have been available to them. To raise money, ICOs introduce the idea of digital tokens or coins. These tokens represent a project's ownership or access to its goods or services and Startups can use tokenization to produce an intrinsically valuable digital asset that can be sold on cryptocurrency exchanges, possibly boosting investor liquidity.

This report aims to leverage machine learning techniques to predict the success of ICOs using information about the project and analysing factors that make an ICO successful or unsuccessful and accurately forecast whether a crowdfunding campaign will achieve its objectives or not.

2 Data Understanding and Preparation

The dataset contains **2767** entries of various fundraising projects containing information in **16** columns about their " Unique ID, whether the project achieved their funding goal or not, the start and end date of their campaign, the number of blockchains that can be issued, the price of each blockchain coin, the team size of the fundraising project, the country where the company is located in, the brand slogan of the company, experts rating on the quality of projects, whether the project has any minimum investment threshold or not, The percentage of blockchain coin distributed to investors relative to all the coin created, information about which blockchain platform is used, if there is a video about fundraising project or not and if they have an internet presence on Github and Reddit or not".

For proper analysis and forecasting, the study of each of these parameters is important.

First and foremost, 180 rows consisted of **nan** values, hence they were removed to have a complete dataset. The start and End date columns had **wrong data types** and were amended**. Label encoding** was done on the success column to make it '0/1' instead of 'Y/N'. So, now the dataset looks like Figure 1.

	ID	hasVideo	rating	priceUSD	startDate	endDate	teamSize	hasGithub	hasReddit	coinNum	minInvestment	distributedPercentage	success_encoded
count	2392.000000	2392.000000	2392.000000	2392.000000	2392	2392	2392.000000	2392.000000	2392.000000	2.392000e+03	2392.000000	2392.000000	2392.000000
mean	1284.065635	0.754181	3.211329	19.908223	2018-08-04 03:09:37.926421248	2018-10-13 07:36:55.384615424	13.314799	0.592809	0.658445	9.459402e+12	0.466973	1.105443	0.394649
min	1.000000	0.000000	1.200000	0.000000	2010-01-10 00:00:00	2016-05-31 00:00:00	1.000000	0.000000	0.000000	1.200000e+01	0.000000	0.000000	0.000000
25%	622.750000	1.000000	2.700000	0.040000	2018-03-09 00:00:00	2018-04-30 18:00:00	8.000000	0.000000	0.000000	5.000000e+07	0.000000	0.400000	0.000000
50%	1253.500000	1.000000	3.200000	0.120000	2018-07-01 00:00:00	2018-09-06 00:00:00	12.000000	1.000000	1.000000	1.981258e+08	0.000000	0.560000	0.000000
75%	1933.250000	1.000000	3.800000	0.500000	2018-11-26 00:00:00	2019-03-01 00:00:00	17.000000	1.000000	1.000000	5.835950e+08	1.000000	0.700000	1.000000
max	2753.000000	1.000000	4.800000	39384.000000	2020-06-06 00:00:00	2020-08-10 00:00:00	75.000000	1.000000	1.000000	2.261908e+16	1.000000	869.750000	1.000000
std	767.050833	0.430662	0.671086	806.114315	NaN	NaN	8.139056	0.491414	0.474330	4.624814e+14	0.499012	18.696118	0.488877

```
df.platform.unique()
```

```
array(['Ethereum', 'XAYA', 'Stellar', 'Separate blockchain',
       'IOV Blockchain', 'Separate Blockchain  ', 'DECOIN Blockchain',
       'EOS', 'Native', 'VeChain', 'Stellar Protocol', 'Stratis', 'X11',
       'Ethereum   ', 'NEO', 'NEM', 'X11 blockchain', 'Ethereum ',
       'CryptoNight', 'PivX', 'TEZOS', 'ERC20', 'Waves', 'BitForex',
       'Tomochain', 'ENTRY', 'Graphene', 'Acclaim', 'Zilliqa',
       'Hyperledger', 'GoChain', 'Slatechain', 'BitShares',
       'Ethereum, Waves', 'RSK', 'IronGeekChain', 'ENLTE PLATFORM',
       'Scrypt', 'Apollo Blockchain', 'Coincart', 'Achain', 'Tron',
       'Neurochain', 'Separate Blockchain', 'Blockchain', 'Ardor',
       'Ethereum     ', 'WizeBit', 'Lisk', 'ICON', 'Wanchain',
       'Ethereum  ', 'ETH', 'Hard-Fork of Litecoin', 'Fiber', 'QRC',
       'TRON', 'Ventureon', 'StartEngine', 'Ethereum    ', 'Bitcoin',
       'Steem', 'Coffe', 'Nem', ' NEO', 'Komodo', 'Monero', 'MultiChain',
       'SmartX', 'Newton', 'Ripemd160', 'New Blockchain', 'Multichain',
       ' Ethereum', 'X13', 'NXT', 'iOlite Blockchain', 'Litecoin',
       'Keccak', 'Etherum', 'Pivx', 'Neblio ', 'TON', 'xDAC', 'Neo',
       'DAG', 'Separate Blockchain ', 'Bitshares', 'Tron    ', 'ST20',
       'STRATIS', 'UNIVERSA', 'AION', 'QTUM', 'ChainRepublik Blockchain',
       'Counterparty', 'Filecoin network', 'POS,POW',
       'Infinity Blockchain', 'Akroma', 'DPoS', ' ', 'Eos', 'Ethererum',
       'YouToken', 'POS', 'pow/pos', 'CryptoNote-based Blockchain',
       'Zuum', 'ISL-Blockchain', '\u200bKomodo', 'MAHRA platform ',
       'WAVES'], dtype=object)
```

```
df.countryRegion.unique()
```

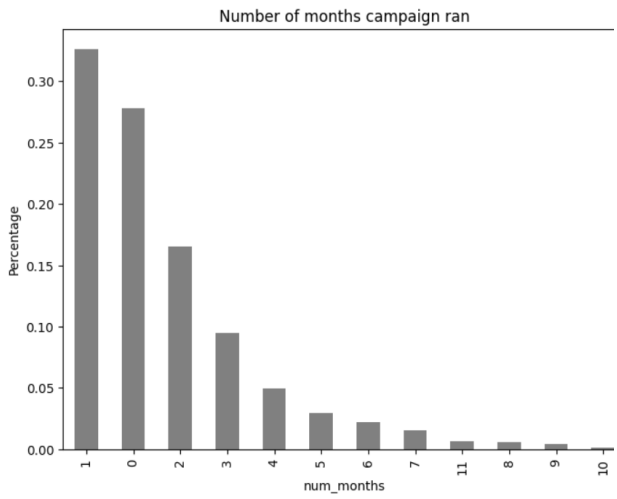
```
array(['Singapore', 'Malta', 'UK', 'Mauritius', 'Poland', 'Germany',
       'Samoa', 'Thailand', 'Estonia', 'Netherlands', 'Cayman Islands',
       'Gibraltar', 'France', 'Saint Kitts and Nevis',
       'British Virgin Islands', 'USA', 'Belize', 'Panama', 'Slovenia',
       'Liechtenstein', 'Australia', 'Venezuela', 'Switzerland',
       'Seychelles', 'Portugal', 'Nigeria', 'Bulgaria', 'Cyprus',
       'Brazil', 'India', 'Ukraine', 'United Arab Emirates',
       'Sierra Leone', 'Canada', 'Lithuania', 'Austria', 'Georgia',
       'Russia', 'Argentina', 'Montenegro', 'Denmark', 'Romania',
       'Latvia', 'Serbia', 'Malaysia', 'South Africa', 'Kyrgyzstan',
       'Saint Vincent and the Grenadines', 'Costa Rica', 'Ireland',
       'Egypt', 'Japan', 'Israel', 'Bahamas', 'Belarus',
       'Marshall Islands', 'Zimbabwe', 'Northern Mariana Islands',
       'Norway', 'South Korea', 'Spain', 'Isle of Man', 'Italy',
       'Bermuda', 'México', 'Czech Republic', 'Sweden', 'Indonesia',
       'Barbados', 'New Zealand', 'Bosnia and Herzegovina', 'Croatia',
       'China', 'Hungary', 'Mongolia', 'Turkey', 'Mexico', 'Anguilla',
       'Belgium', 'Andorra', 'Ecuador', 'Macedonia', 'Dominican Republic',
       'Greece', 'Luxembourg', 'SINGAPORE', 'Philippines', 'Peru',
       'Chile', 'Vietnam', 'Finland', 'Syria', 'New Caledonia',
       'Kazakhstan', 'Colombia', 'Curaçao', 'French Polynesia',
       'Tanzania', 'Guinea-Bissau', 'Ghana', 'Afghanistan',
       'Saudi Arabia', 'Puerto Rico', 'Tunisia', 'Cambodia', 'Slovakia',
       'Pakistan', 'Vanuatu', 'Timor-Leste', 'Congo', 'Kuwait', 'Curacao',
       'india', 'Iceland'], dtype=object)
```

Figure 1: Data Description

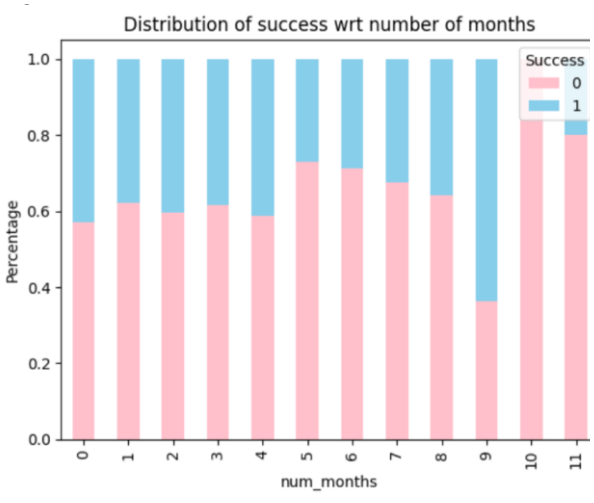
There were a lot of data quality issues within the dataset which needed to be resolved.

There were 11 fundraising projects where the **end date of the campaign was before the start date** which is impractical, hence those data rows were removed from the dataset and **a new column "num_months"** was defined to calculate the period of the entire campaign and columns "startDate" and "endDate" were removed.

The distribution of the number of months campaign ran for is presented in Figure 2.



(a)



(b)

Figure 2: (a) Percentage distribution of data for campaign duration (b) success distribution for campaign duration

In the "platform" column, there were a lot of entries with **misspelt words and case sensitivity and spacing issues** which were rectified, Ethereum was written as "Ethereum " or "Etherum" or the same values were misclassified as new unique values.

"Ethereum" is the most widely used platform constituting 87%+ of all platforms. Hence platform column was divided into "Ethereum", "waves" (second most popular platform) and "others".

The histogram for the column 'platform' was plotted as shown in Figure 3.

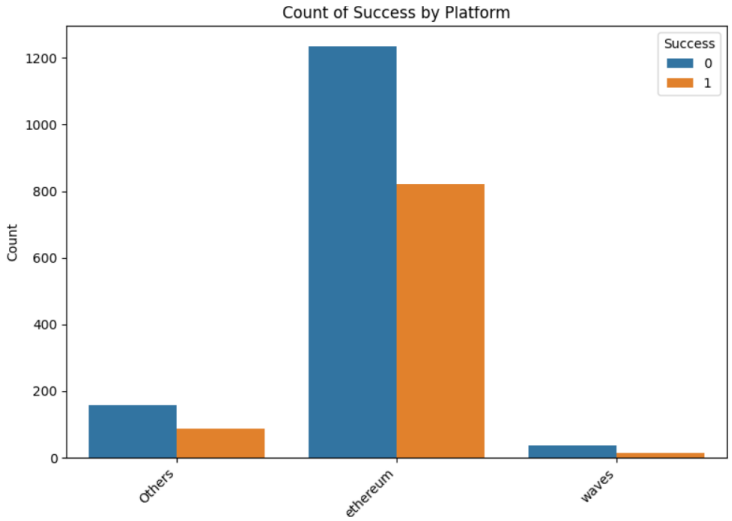


Figure 3: Count of success and failure per blockchain platform.

The number of blockchains that can be issued i.e. "coinNum" **value ranges from 12 to $2.2e^{16}$** . Hence log scaling was appropriate before feeding it to any machine-learning model since it can affect model performance where values with larger magnitude would dominate and degrade interpretability of model results and affect decision making.

The transformed coinNum distribution now looks like Figure 4.

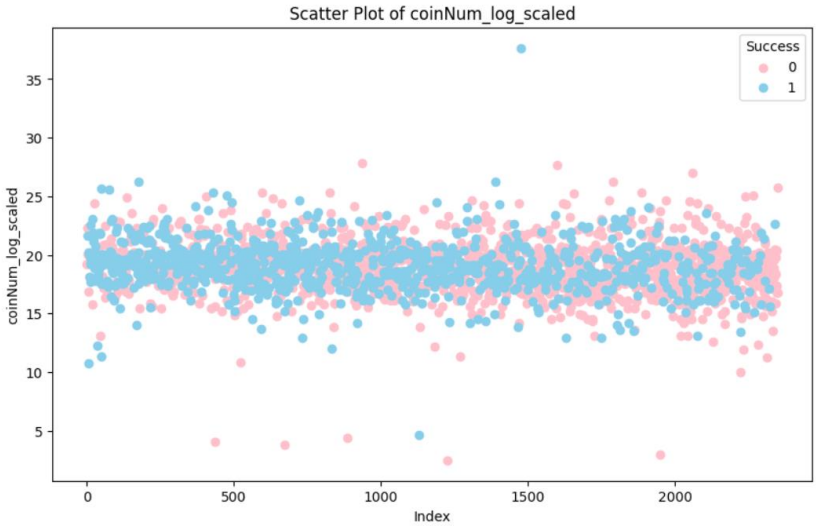


Figure 4: Scatter plot of Log of the number of blockchains that can be issued,

Many Outliers were present in the "priceUSD" column which were removed from the dataset since the analysis aims to train the model for generalized data.

The rest of the data was normalized using min-max normalization to improve the performance, convergence, and robustness of the machine learning models. So now the distribution of the price of each blockchain coin is presented in Figure 5. Even though it might appear most of the data is near 0, all values are within acceptable range and removing more points will lead to loss of valuable data.

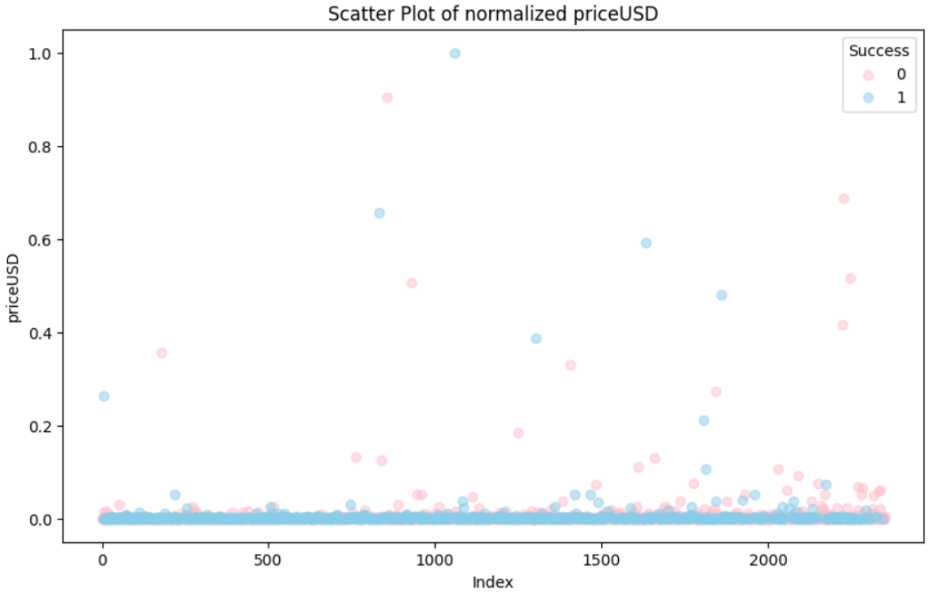


Figure 5: Normalized distribution of the price of each blockchain coin

In the percentage column, a few **rows had values greater than 1** which is logically flawed and hence were removed.

The spread of the percentage of blockchain coins distributed to investors relative to all the coins created is shown in Figure 6.

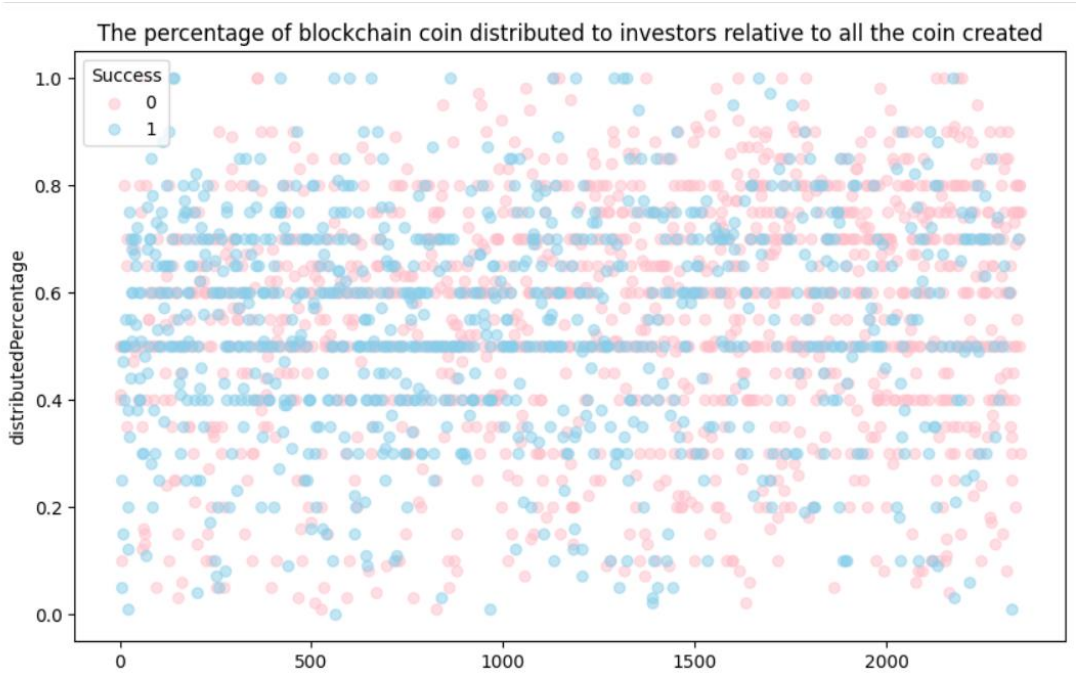


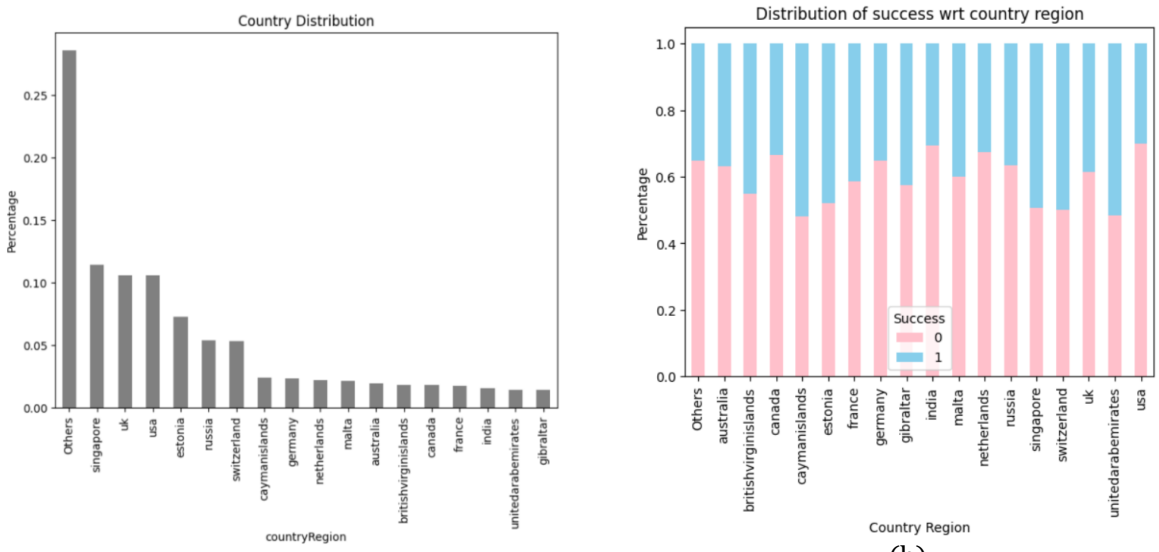
Figure 6: The percentage of blockchain coins distributed to investors relative to all the coins created

The dataset is very diverse consisting of funding campaigns from around the world.

In the "countryRegion" column, there were a lot of entries with **misspelt words and case sensitivity and spacing issues** which were rectified. Few countries had special characters in their names which were **converted to ASCII characters**.

The dataset has crowdfunding projects from 109 countries, but the maximum campaigns were from "Singapore, UK, USA, Estonia, Russia and Switzerland". Most of the other countries have one or two campaigns so all the countries with a count of less than 30 were **grouped** for convenience and efficiency.

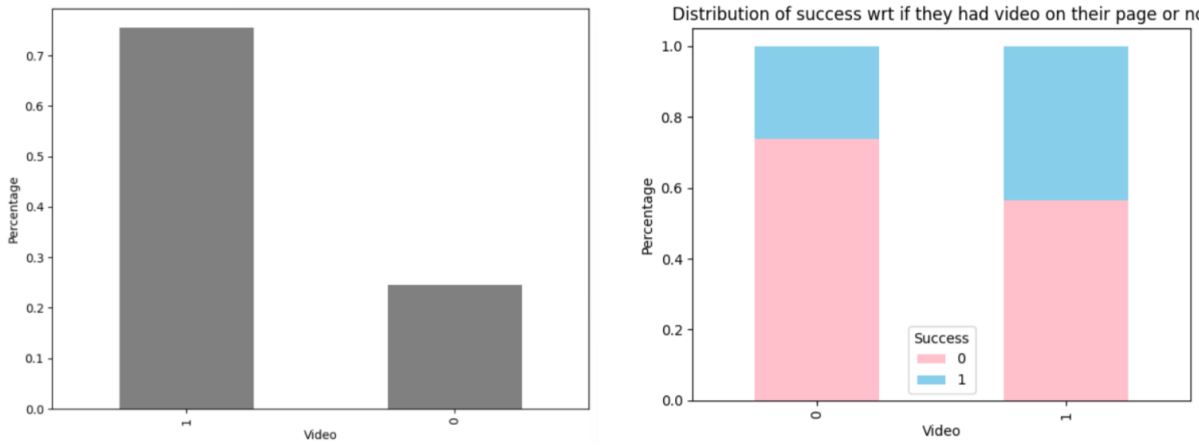
Country-wise count of campaign distribution and success distribution is presented in Figure 7.



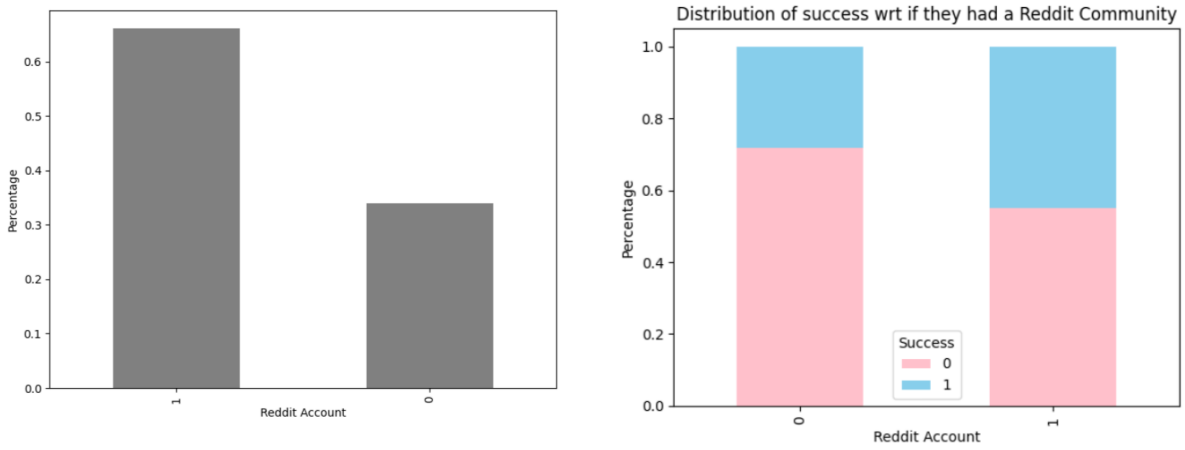
(a) (b)

Figure 7: Country-wise (a) count of campaign distribution (b)success distribution.

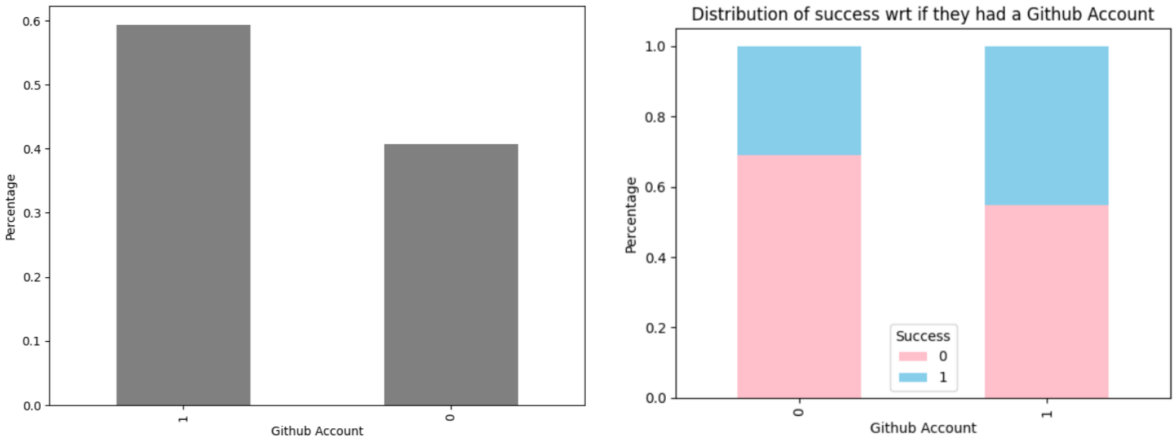
Figure 8 shows the percentage distribution and success distribution for the “hasVideo”, “hasReddit” and “hasGithub” columns of the dataset.



(a)



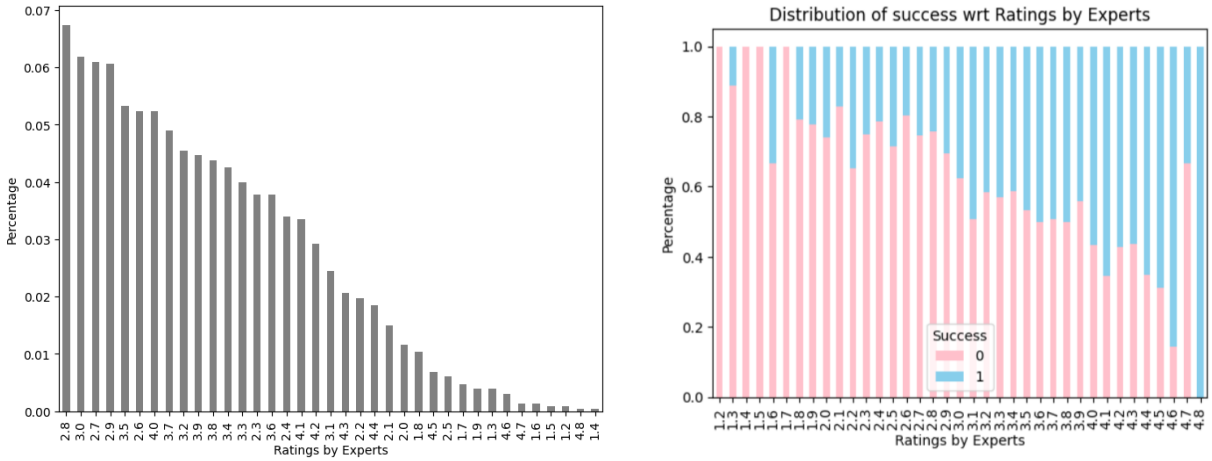
(b)



(c)

Figure 8: The percentage distribution and success distribution for (a) whether there was a video about the fundraising project or not (b) whether they had an internet presence on Github (c) Reddit or not.

Figure 9 displays how projects with **higher ratings by experts have been more successful** and this makes sense since a better-presented campaign will attract more people to invest in their projects.



(a) (b)

Figure 9: (a) Percentage distribution of Expert rating on the quality of projects (b) Success distribution based on expert ratings

A well-balanced team is very important for the success of any endeavour, too many chefs ruin the dish, too little and nothing gets done. So, balance is important. Team size distribution for different projects is showcased in Figure 10.

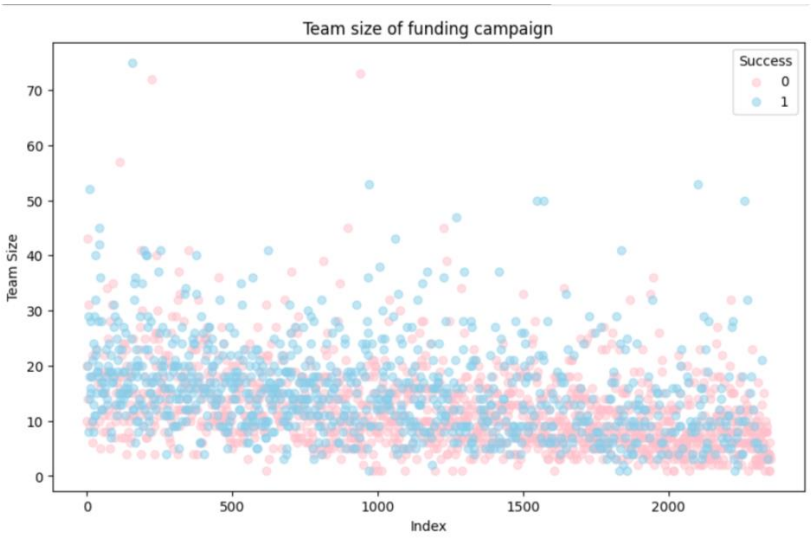


Figure 10: Distribution of the team size of the fundraising project.

"ID" and "Brand slogan" variables were not relevant for analysis, since they are **unique features to any project** and cannot help with generalizing and finding what makes any fundraising campaign successful hence cannot be used for training a machine learning model.

Even though a good brand slogan can elevate the project's value, still it's not prominent for this analysis.

After cleaning the data and resolving data quality issues, the cleaned dataset looks like Figure 11.

	hasVideo	rating	priceUSD	teamSize	hasGithub	hasReddit	minInvestment	distributedPercentage	success_encoded	num_months	coinNum_log_scaled
count	2329.000000	2329.000000	2329.000000	2329.000000	2329.000000	2329.000000	2329.000000	2329.000000	2329.000000	2329.000000	2329.000000
mean	0.755260	3.210133	0.006334	13.272649	0.593388	0.661228	0.471018	0.547793	0.393731	1.699442	19.005828
std	0.430025	0.671294	0.045088	8.084069	0.491307	0.473394	0.499267	0.209353	0.488681	1.915483	2.251722
min	0.000000	1.200000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	2.484907
25%	1.000000	2.700000	0.000212	8.000000	0.000000	0.000000	0.000000	0.400000	0.000000	0.000000	17.727534
50%	1.000000	3.200000	0.000635	12.000000	1.000000	1.000000	0.000000	0.570000	0.000000	1.000000	19.113828
75%	1.000000	3.800000	0.002646	17.000000	1.000000	1.000000	1.000000	0.700000	1.000000	2.000000	20.143447
max	1.000000	4.800000	1.000000	75.000000	1.000000	1.000000	1.000000	1.000000	1.000000	11.000000	37.657570

```
df.countryRegion.unique()
```

```
array(['malta', 'uk', 'Others', 'singapore', 'germany', 'estonia',
       'netherlands', 'caymanislands', 'gibraltar', 'france',
       'britishvirginislands', 'usa', 'switzerland', 'india',
       'unitedarabemirates', 'canada', 'russia', 'australia'],
      dtype=object)
```

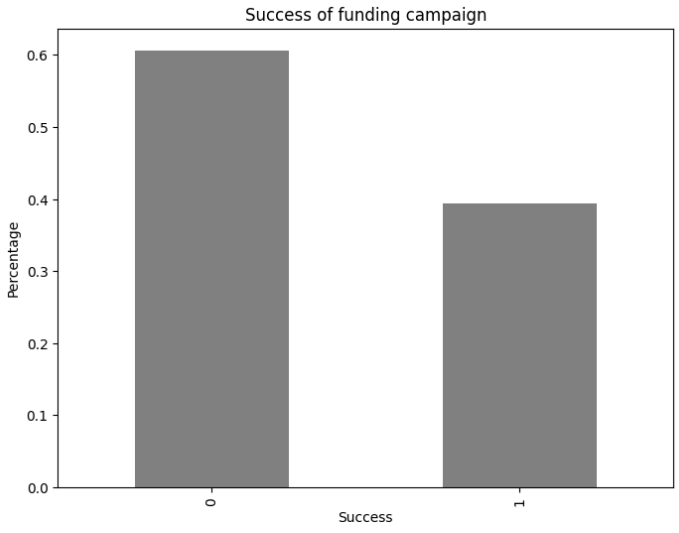
```
df.platform.unique()
```

```
array(['Others', 'ethereum', 'waves'], dtype=object)
```

Figure 11: Summary statistics of cleaned data.

Since machine learning techniques used in this analysis cannot handle categorical data, variables "countryRegion" and "Platform" were converted to numerical data using **one-hot encoding** and now the dataset was ready for modelling machine learning techniques.

Still, there was an issue with the dataset. The prediction class 'success_encoded' was unbalanced as shown in Figure 12(a). There were more unsuccessful fundraising campaigns than successful, and taking consideration of this was crucial while modelling. However, by calculating the correlation of 'Success_encoding' with all other columns as shown in Figure 12(b), we can observe how the success of the fundraising campaign was impacted by all these variables. There is a lot of ambiguity in what features make a campaign successful.



(a)

```
success_encoded                      1.000000
rating                               0.266477
teamSize                             0.236166
hasReddit                            0.162757
hasVideo                             0.152134
hasGithub                            0.141096
countryRegion_singapore              0.074130
countryRegion_switzerland            0.051580
minInvestment                        0.049432
countryRegion_estonia                0.048983
countryRegion_caymanislands          0.039877
coinNum_log_scaled                   0.039828
platform_ethereum                    0.034419
countryRegion_unitedarabemirates     0.029794
countryRegion_britishvirginislands   0.016268
countryRegion_gibraltar              0.007487
countryRegion_france                 0.005727
countryRegion_malta                  0.001900
priceUSD                             0.000034
countryRegion_uk                    -0.006426
countryRegion_australia             -0.007021
countryRegion_germany               -0.013206
countryRegion_russia                -0.014025
countryRegion_canada                -0.016753
countryRegion_netherlands           -0.020664
countryRegion_india                 -0.022613
platform_waves                      -0.028416
distributedPercentage               -0.041845
num_months                          -0.054787
countryRegion_usa                   -0.065357
Name: success_encoded, dtype: float64
```

(b)

Figure 12: (a)Distribution of successful campaigns (b) Correlation of success and all variables

The dataset was now divided into features containing all variables except 'success_encoded' and the target variable containing 'success_encoded'. This was further split in training and test set randomly in an 80-20 ratio. Using this method, it is possible to identify any overfitting symptoms shown by the model while training on a train set and evaluate how well any model generalizes to new data.

3 Modelling

Now that we have a complete dataset and train test split, we can feed this data to various machine learning algorithms and train it to explore patterns in the dataset.

Modelling Techniques:

1) Random Forest Classifier

A Random Forest Classifier is an ensemble learning algorithm used for classification tasks. It produces predictions by combining several decision trees, which improves robustness and accuracy. It performs exceptionally well with a variety of data kinds, resists overfitting, and offers insights into the significance of features.

For classification tasks, the Random Forest Classifier is highly regarded for its high accuracy, scalability, and interpretability.

For classification, RandomForestClassifier with the following parameters for analysis was used:

max_depth=3 and n_estimators=100

We used a combination of 100 decision trees of a maximum depth of 3 for training and decision-making.

2) XGBOOST Classifier

Extreme Gradient Boosting Classifier, or XGBoost, is a powerful machine learning technique that is well-known for its quickness and precision when applied to classification problems.
It makes use of a group of decision trees that have been trained successively, fixing the mistakes made by earlier trees to enhance prediction accuracy.

3) Support Vector Machine (SVM) Classifier

SVM is a supervised machine learning algorithm used for classification tasks. It determines the ideal hyperplane that maximizes the margin between classes in the feature space. It is capable of handling both linear and nonlinear classification tasks with versatility, robustness against overfitting, and effectiveness in high-dimensional domains.

For classification Support Vector Classifier (SVC) with the following parameters for classification was used:

C=0.013 and kernel='linear'

With a 'linear' kernel, SVC divides classes using a linear decision boundary and 'C' is the regularization parameter.

4) K-nearest neighbours (KNN) classifier:

KNN is a straightforward classification technique that gives a data point a class label based on the majority class of its closest neighbours.

The following parameters were used for the KNeighborsClassifier during this analysis:

p: 2, n_neighbors: 90, weights: 'uniform', algorithm: 'auto'

"90" neighbours were considered, "uniform" weights mean all points in the neighbourhood are considered equal, "auto" selects the most appropriate algorithm to compute nearest neighbours automatically based on input data, p="2" means Euclidean distance is used for calculating the distance to neighbours.

5) Logistic regression:

Logistic Regression is a simple and efficient algorithm used for binary classification tasks. It uses a logistic function to model the relationship between features and the likelihood of a binary outcome.

The following parameters were used for the Logistic Regression during this analysis:

Penalty: 'l1', Solver: 'lbfgs', Maximum Iterations: 10

Lasso regularization, or "l1" penalty, encourages sparsity in the model by adding the absolute values of the coefficients as a penalty term to the loss function. 'lbfgs'(Limited-memory Broyden–Fletcher–Goldfarb–Shanno) is a popular optimization algorithm for logistic regression. Maximum iteration restricts the number of iterations the optimization algorithm can iterate up to before reaching convergence i.e. "10" here.

6) Neural Networks:

Neural networks are composed of interconnected nodes arranged into layers, modelling the organization of the brain. To convert input data and generate an output, each node employs an activation function. To minimize the differences between expected and actual outputs, the network optimizes its learning process by changing the weights of connections between nodes during training.
These networks are trained using training data to identify patterns and features that allow them to classify input data into distinct groups.

- **Neural Network model 1:**

  Architecture:

  model = Sequential([Dense(128, activation='relu', input_shape=(X_train.shape[1],)), Dense(128, activation='relu'), Dense(64, activation='relu'), Dense(64, activation='relu'), Dense(32, activation='relu'), Dense(32, activation='relu'), Dense(16, activation='relu'), Dense(16, activation='relu'), Dense(2, activation='softmax')])

  Optimization algorithm: 'adam', **Metrics:** 'Accuracy', **Epochs:**6, **Batch Size:**4

- **Neural Network Model 2:**

  Architecture:

  model = Sequential([Dense(64, activation='relu', input_shape=(X_train.shape[1],)), Dense(64, activation='relu'), Dense(32, activation='relu'), Dense(32, activation='tanh'), Dense(16, activation='relu'), Dense(16, activation='relu'), Dense(2, activation='softmax'])

  Optimization algorithm: 'adam', **Metrics:** 'Accuracy', **Epochs:**6, **Batch Size**:4

- **Neural Network Model 3:**

  Architecture:

  model = Sequential([Dense(32, activation='relu', input_shape=(X_train.shape[1],)), Dense(32, activation='relu'), Dense(32, activation='relu'), Dense(32, activation='relu'), Dense(32, activation='relu'), Dense(16, activation='relu'), Dense(2, activation='softmax')])

  Optimization algorithm: 'adam'**, Metrics**: 'Accuracy'**, Epochs:**10**, Batch Size:**8

Dense ("Number of neurons", activation='Activation Function'): The number of neurons and type of activation function can be changed depending on the model and requirement.

'Adam', which stands for Adaptive Moment Estimation, is an optimization algorithm used to adjust network weights during training. It enables quicker convergence and improved performance by modifying the learning rate for each parameter per the past gradients.

'Metrics' are used to evaluate the model's performance during training and validation.
One full iteration of the training dataset is referred to as an 'epoch'. The model's weights are updated during training to minimize the loss function throughout epochs.

The number of samples processed before the model's weights are updated is determined by the 'batch size'. For example if batch size=4, The model will process four samples at a time computing gradients and updating weights in the process.

Evaluation:

Following training, each model was put to the test on the test set to assess its performance using the following success metrics:
This includes:

1) **Confusion Matrix**
   A confusion matrix is a table that summarizes the performance of a classification model.
   It displays the number of predictions that are true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN).
   This matrix offers insight into how well the model can classify instances in various classes.

2) **Accuracy**
   The ratio of correctly predicted cases both true positives and true negatives, to the total number of samples is known as accuracy.

It is a broad indicator of how accurate the model is at predicting the correct class of new data.

3) **Precision:**
Precision is the percentage of actual positive predictions among all of the model's positive predictions.
It is helpful when the cost of false positives is high and concentrates on the accuracy of positive predictions.

4) **F1 score**
The harmonic mean of recall and precision is the F1 score.
It provides a statistic to assess the model's performance that strikes a compromise between precision and recall.
When there is an imbalance in the dataset between the amount of positive and negative examples, the F1 score is especially helpful.

For this analysis:

- **True Positive (TP):** The model correctly predicts a successful fundraising campaign (positive class).
- **True Negative (TN):** The model correctly predicts an unsuccessful fundraising campaign (negative class).
- **False Positive (FP): The** model incorrectly predicts a successful fundraising campaign when it's unsuccessful.
- **False Negative (FN):** The model incorrectly predicts an unsuccessful fundraising campaign when it's successful.

Model evaluation results are presented in Table 1.

S.NO	MODEL	CONFUSION MATRIX	F1 SCORE	ACCURACY	PRECISION
1)	Random Forest Classifier	[258 33 126 51]	0.3908	0.66025	0.6071
2)	XGBoost Classifier	[200 91 97 80]	0.4597	0.5982	0.4678
3)	SVM	[255 36 116 61]	0.4452	0.6752	0.6288
4)	KNN	[232 59 102 75]	0.4823	0.6559	0.5597
5)	Logistic Regression	[222 69 96 81]	0.4954	0.6474	0.54
6)	Neural Network Model 1	[219 72 90 87]	0.6497	0.6538	0.6476
7)	Neural Network Model 2	[258 33 119 58]	0.6440	0.6752	0.6665
8)	Neural Network Model 3	[263 28 126 51]	0.6316	0.6709	0.6645

Table 1: Success Metrics score for all models trained

Neural network models outperformed the traditional machine learning models in classifying successful and unsuccessful crowdfunding campaigns in all evaluation metrics.
The baseline accuracy score is around 0.6 since the dataset is unbalanced, so even if the model predicts '0' as an output every time, even then it would be accurate 60% of the time. Hence accuracy is not an appropriate measure for evaluating models for this project.

For assessing models on unbalanced datasets**, the F1 score** is a more suitable metric. Because the F1 score accounts for both recall and precision, it can be used to assess models on datasets that are unbalanced.
Through the computation of the harmonic mean of these two measures, it strikes a compromise between recall (the classifier's capacity to identify all positive samples) and precision and therefore it offers a more thorough evaluation of a model's performance, especially in situations when one class is significantly more common than the other. Because it gives a more realistic picture of a model's capacity to accurately classify instances across all classes.

Hence **Neural Network Model 1** is best for correctly classifying a campaign to be successful or unsuccessful because of the high F1 score of 0.6497. **Neural network model 2** is overall the most balanced classifier with not only a high F1 score of 0.6440 but an accuracy score of 0.6752 and a precision score of 0.6665 too. So, these two models should be used.

4 Conclusion

Since there are no set requirements for success, forecasting the result of a crowdfunding campaign is extremely difficult. This report showcased the effectiveness of different machine learning models in identifying fundraising campaigns as successful or unsuccessful, despite considerable complexity.

Among the features present in dataset **ratings given to the project and the team size of the campaign project team** are the most influential in determining the campaign's success, Among the models examined, **Neural Network Model 1** emerged as the top performer, with a high F1 score of 0.6497. However, **Neural Network Model 2** demonstrated exceptional balance across multiple metrics, including an impressive F1 score of 0.6440, accuracy score of 0.6752, and precision score of 0.6665. Therefore, these two models, with their respective strengths, stand out as recommended choices for predicting the fate of crowdfunding campaigns.

However, it's crucial to recognize that there remains space for improvement, even in the models that performed the best according to our analysis. More variables or larger datasets may improve model performance and offer a more in-depth understanding of the variables that affect campaign success.
Maybe a dataset with more variables about campaign promotion strategy, market information of the time of the fundraising campaign, background or track record of the fundraising team or past successes could provide more insights into what makes a campaign successful.

In conclusion, even if it's still difficult to forecast the results of crowdfunding campaigns, our research shows how machine learning models can help with decision-making. To improve comprehension and forecasting capability in this dynamic field, further model investigation and improvement is essential.