1. <u>Introduction</u>

Crowdfunding is the process of generating required funds for a venture by collecting money from unknown individuals who are attracted towards the idea of the venture. This is usually done through campaigning their project through digital platforms which establishes the connection between the investors and the venturers. (Belleflamme et al., 2015).

With the evolution of technology, crowdfunding concept also revolutionized its method of funds through **ICO**s (**Initial Coin Offerings**) which is based on the concept of **Decentralized digital currencies called cryptocurrencies**. This became widely popular with the usage of **blockchain technology**. (Vujicic et al., 2018).

Blockchain is a chain of blocks that are linked together in a chronological order where each block contains a set of data or transactions. Blockchain became widely accepted due to certain aspects such as **Decentralized System, Security** and **Transparency.** Many venturers started using this tech to distribute their own Digital coins in exchange for people's investments.

Thus, ICO s gave birth to a **new way** of **generating entrepreneurial finance** where the venturers can determine the price and the number of coins generated(Fisch, 2019). Usually, ICO s set targets for the funds and is declared a success if it achieves its target within the timeframe.

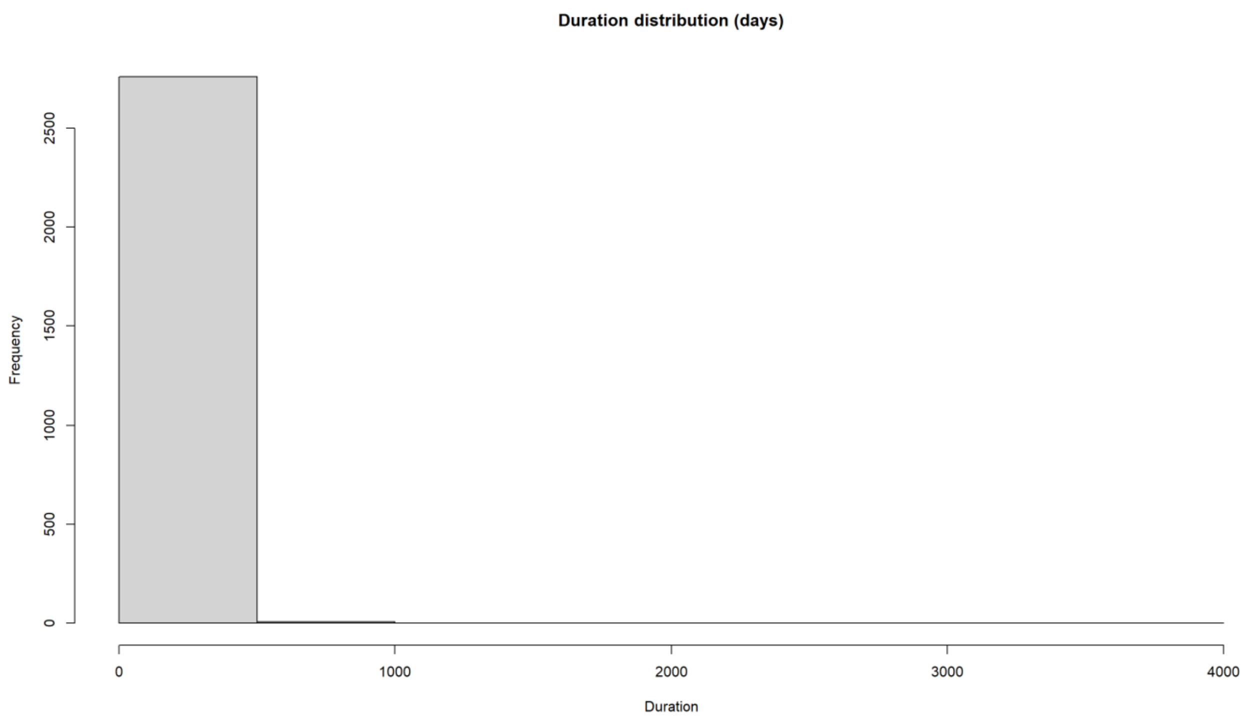
The main **objective** of this coursework is to predict the **success of ICOs**, to achieve this multiple machine learning models will be built using the given dataset and the most accurate and reliable model will be suggested for prediction.

2. <u>Data Exploration</u>

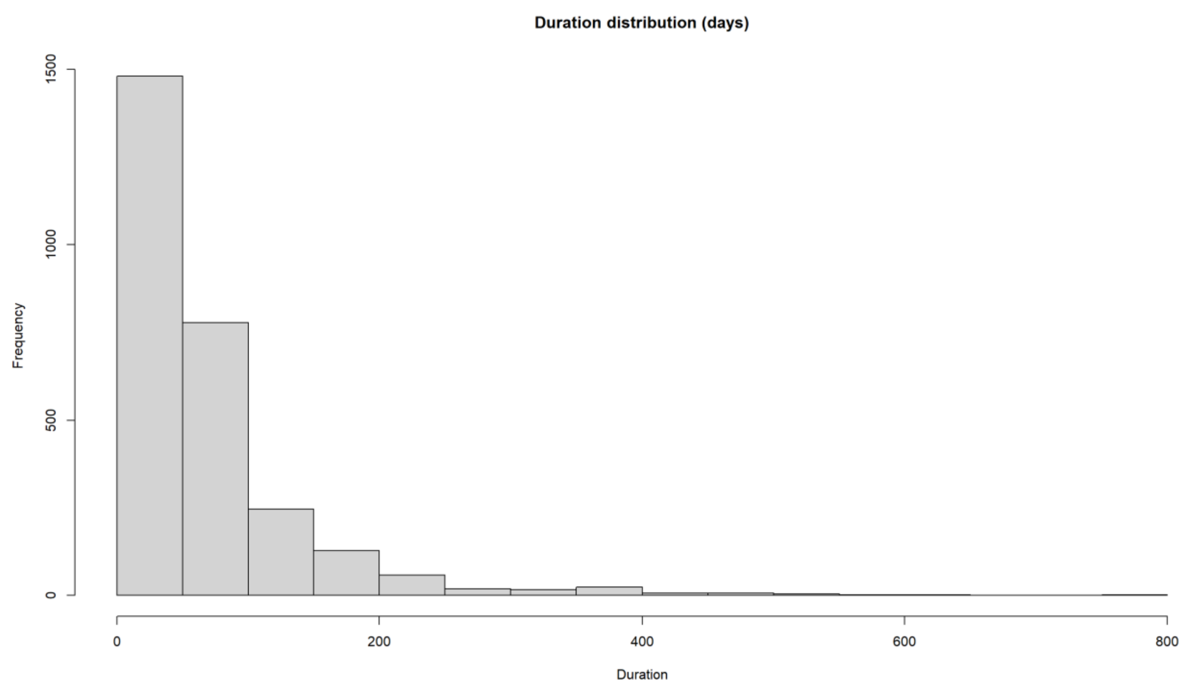
- The given dataset contains **2767** rows with **14** different **features** influencing the class label '**Success'** containing two **categorical factors**: **Y** → *successful*, **N** → *Not successful*.
- The data is made up of **63%** of **unsuccessful** ICOs and **37%** of **successful** ICOs.
- Out of the 14 features **9** are **categorical**, while the remining are numerical.

2(a). startDate & endDate

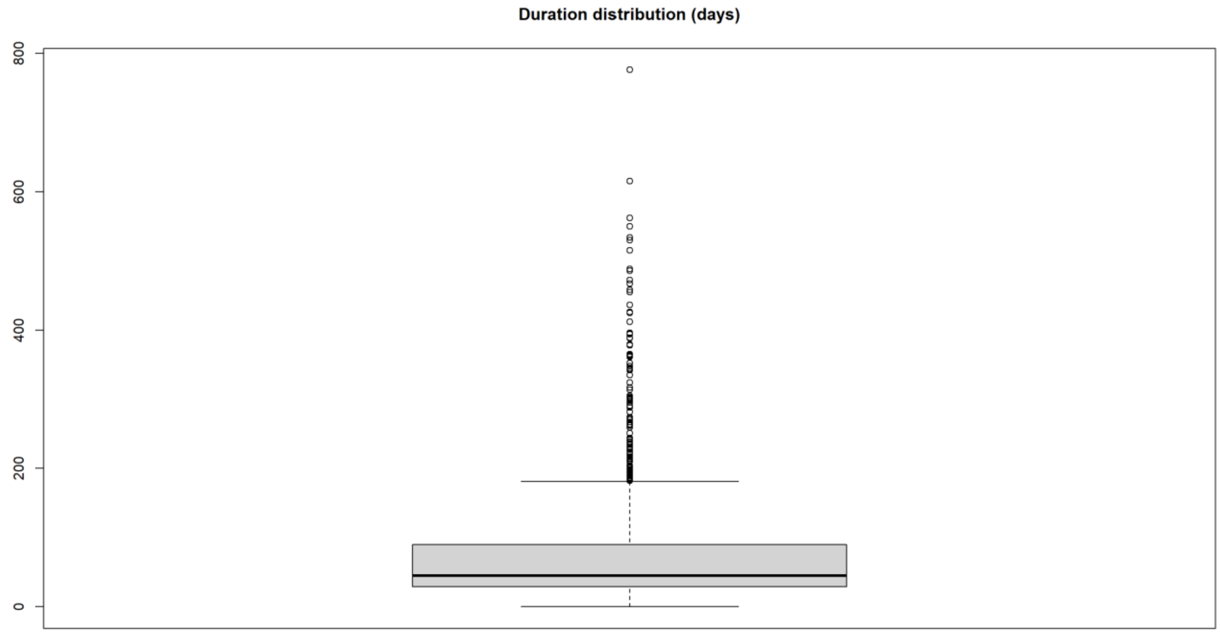
Logically speaking, dates don't have a direct influence on the success or failure of an ICO, but the duration of each ICO may have, so these two columns were used to compute the duration, after which these were removed.



The data is right skewed because of one extreme value, making the distribution unrealistic, so that record was removed.



Now, the distribution is realistic, but still appears to be right skewed hinting on the presence of outliers.

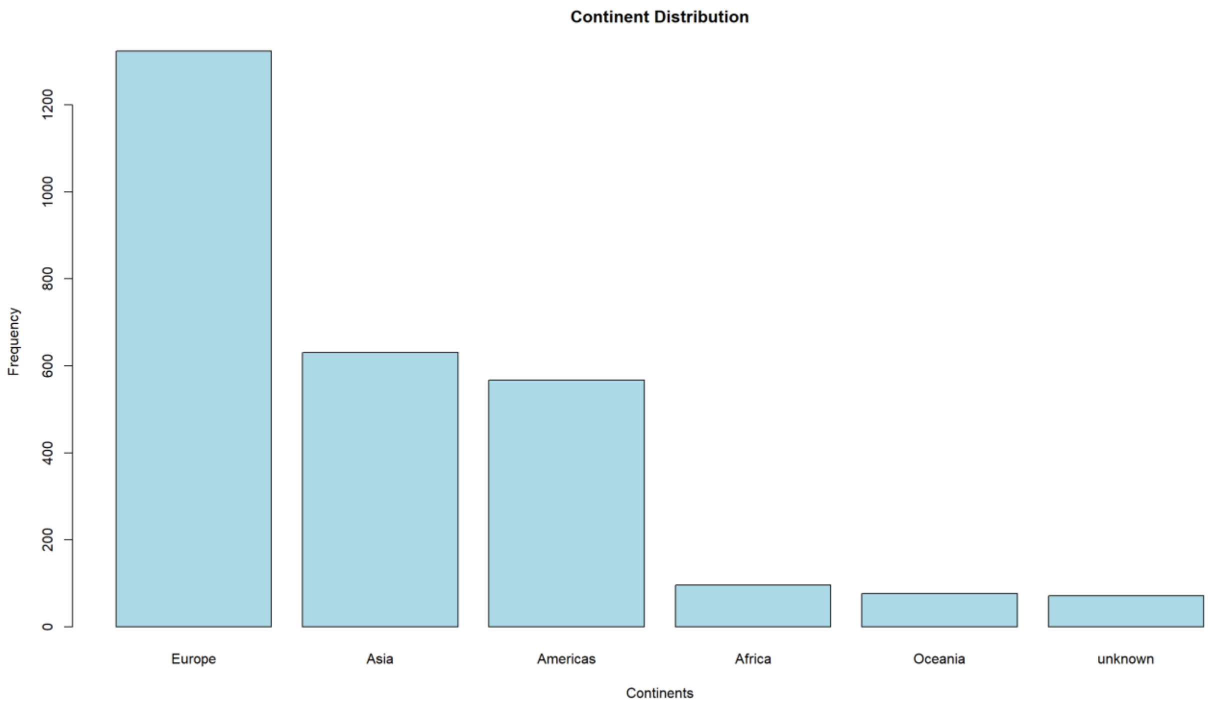


According to the boxplot, anything above the duration of 200 days is an outlier, but there isn't any universal timeframe for ICOs, so logically these outliers cannot be removed.

2(b). CountryRegion

The dataset contains ***121*** distinct **countries**, majority of them were **European countries(*1325*)** followed by **Asian(*631*) & American(*568*)**, while the remaining were **African & Oceania** and ***71* records missed** the Country which were filled in as '***Unknown'***.

On analyzing deeply, irrespective of the country ***the proportion distribution of class labels remained constant for all the countries, so it doesn't make sense to have 121 individual categories for just 2767 records, this will result in weaker learning curve of models***. Thus, the countries were **converted** to respective **continents** using '***countrycode'*** library in R.



Continents	Y(%)	N(%)
Europe	38	62
Asia	40	60
America	34	66
Africa	38	62
Oceanica	34	66
Unknown	10	90

The proportion distribution of the class label for each continent except the 'unknown' is very similar to the overall proportion distribution of the dataset, thus this grouping has not made the data biased or unstable.

Certain models do not work with categorical data, **so seven dummy variables** were created for **seven distinct continents**.

2(C). Platforms

Every ICO uses different blockchain technology available in the market. This feature's recording contained a lot of errors and required pre-processing as explained.

Step 1 *→ Names with Latin words were transformed to English words.*

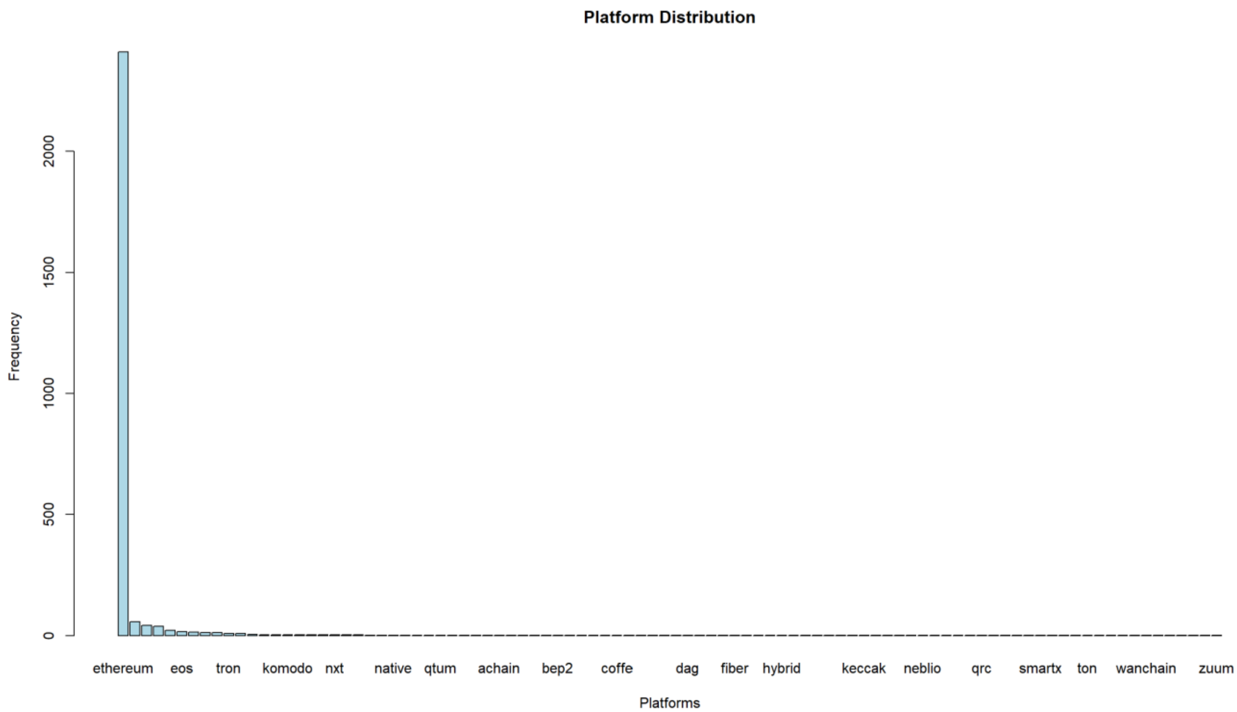
Step 2 *→ 7 records filled with blanks were replaced with 'Others'.*

Step 3 *→ Whitespaces were trimmed down.*

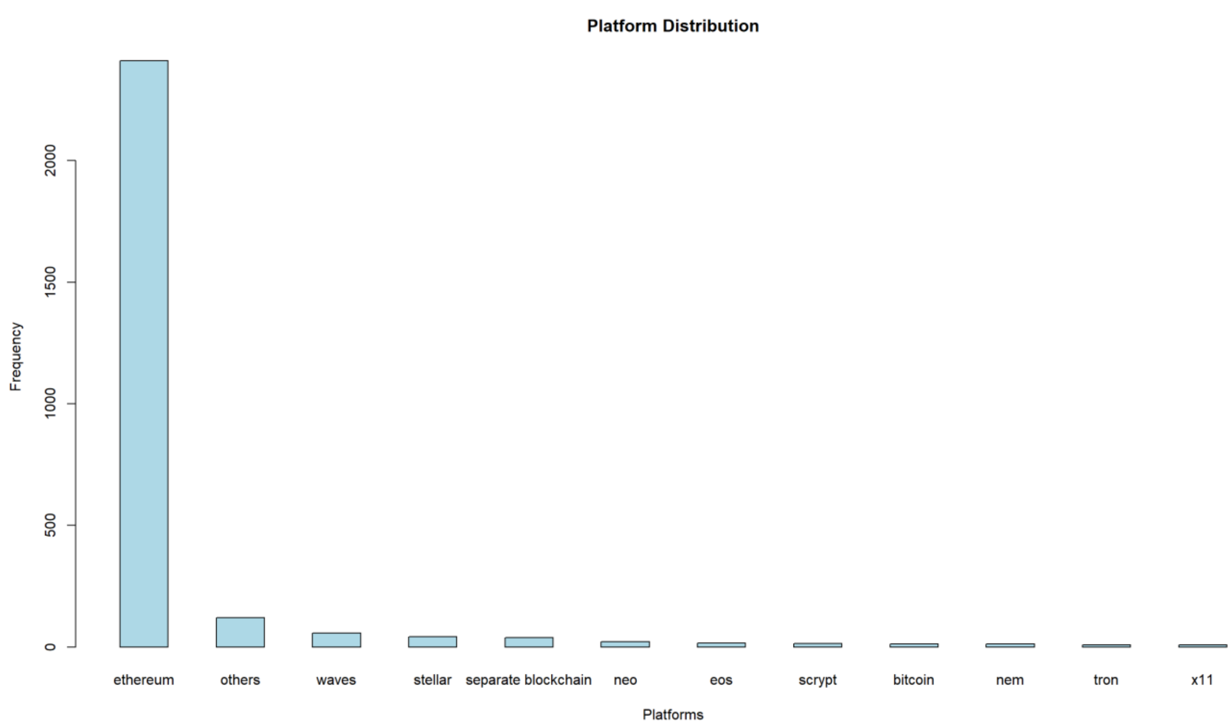
Step 4 *→ words with mixed casings were transformed to lower case.*

Step 5 *→ spelling mistakes were recoded to original name using 'recode' function. For ex: records spelled 'ethererum' or 'etherum', were transformed to 'ethereum'.*

The distribution of the data was as follows.



*About **87**% of records belonged to '**Ethereum'** platforms while the remaining belonged to various other platforms **where more than 90% of the platforms had just one record in the dataset**. This **kind of distribution weakens the learning curve of the model**, so **platforms with less than 5 records were grouped together and were replaced as 'Others'.***



Platforms	Frequency	Y	N
bitcoin	13	7%	93%
eos	17	30%	70%
ethereum	**2411**	**37%**	**63%**
nem	13	23%	77%
neo	22	40%	60%
others	120	40%	60%
scrypt	14	35%	65%
separate blockchain	39	51%	49%
stellar	42	23%	77%
tron	9	33%	66%
waves	57	30%	70%
x11	9	10%	90%

The proportion distribution of class labels for Ethereum is similar to the overall proportion distribution of the dataset thus ***it is evident that this feature might not be distinctive enough in splitting the data.***

As explained above, certain models like SVM or KNN can't process categorical data, so these **platforms were converted as dummy variables**.

2(d). hasVideo, hasReddit, hasGithub & minInvsetment

hasVideo → *Indicator variable set to 1 if the venturer* ***provided a video*** *on campaign page, else 0.*

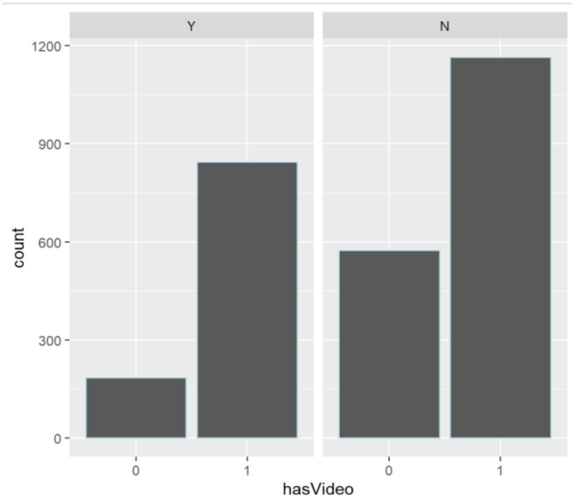
hasReddit → *Indicator variable set to 1 if the venturer* ***provided their official reddit page****, else 0.*

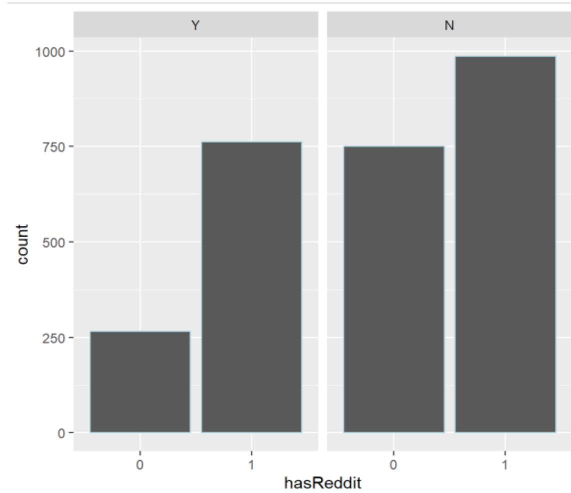
hasGithub → *Indicator variable set to 1 if the venturer* ***provided their official GitHub****, else 0.*

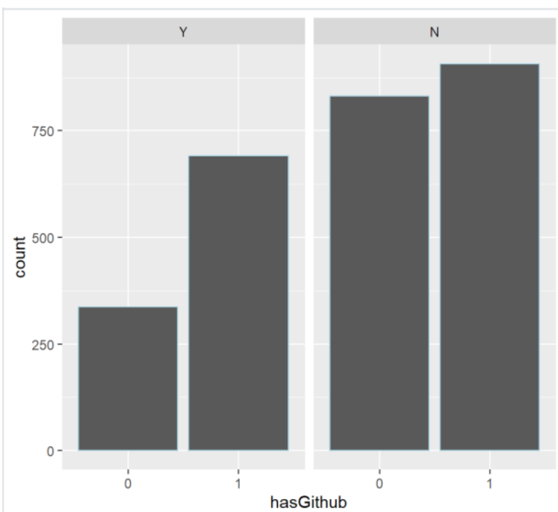
minInvestment → *Indicator variable set to 1 if the venturer* ***has set a minimum investment*** *amount, else 0.*

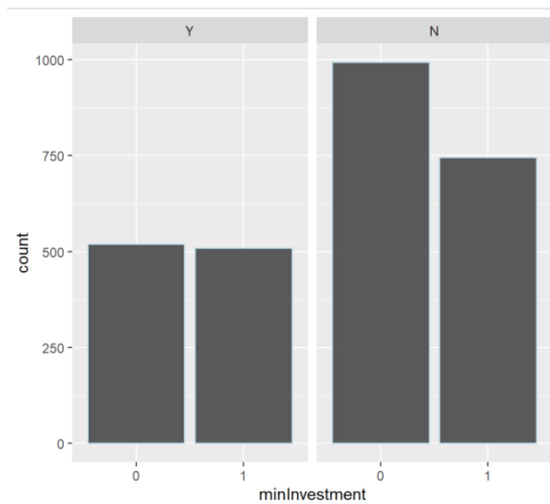
All these four features **did not have any missing values**.

The distribution of these features are as follows:





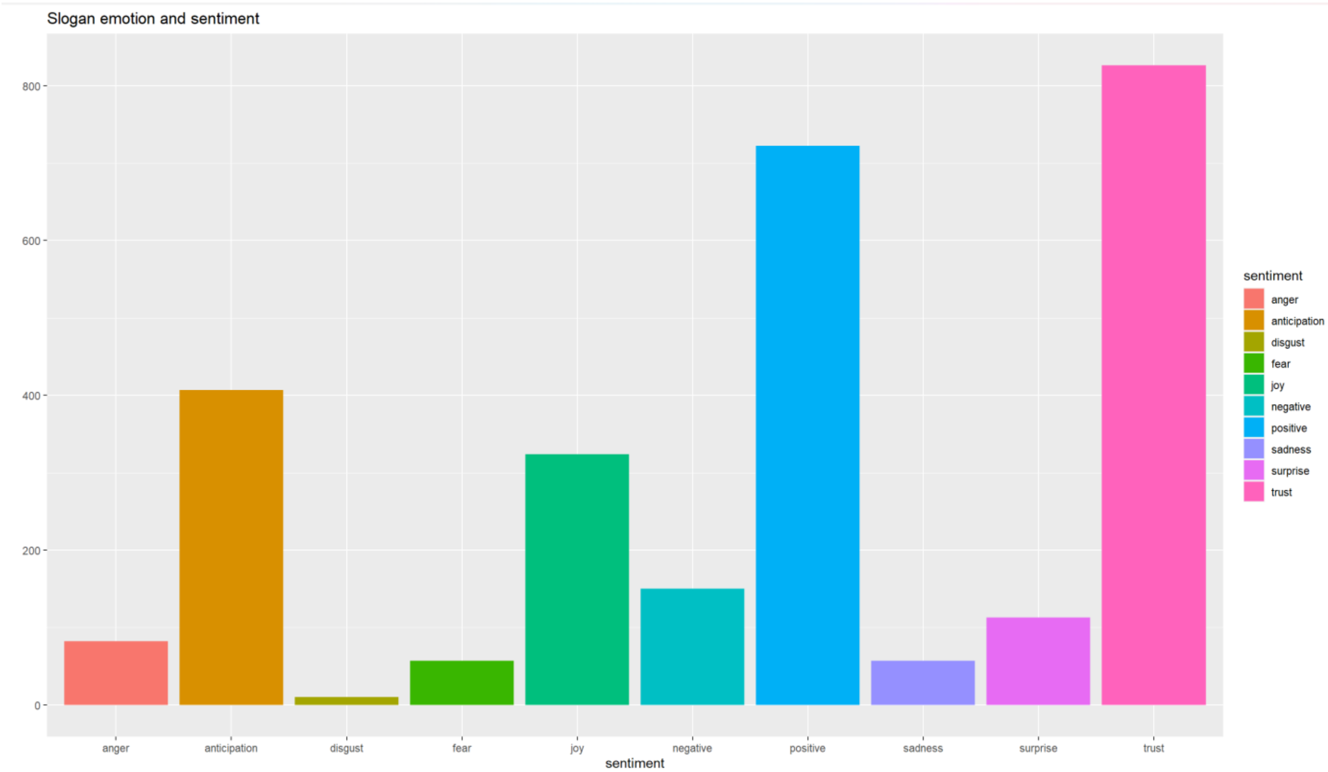




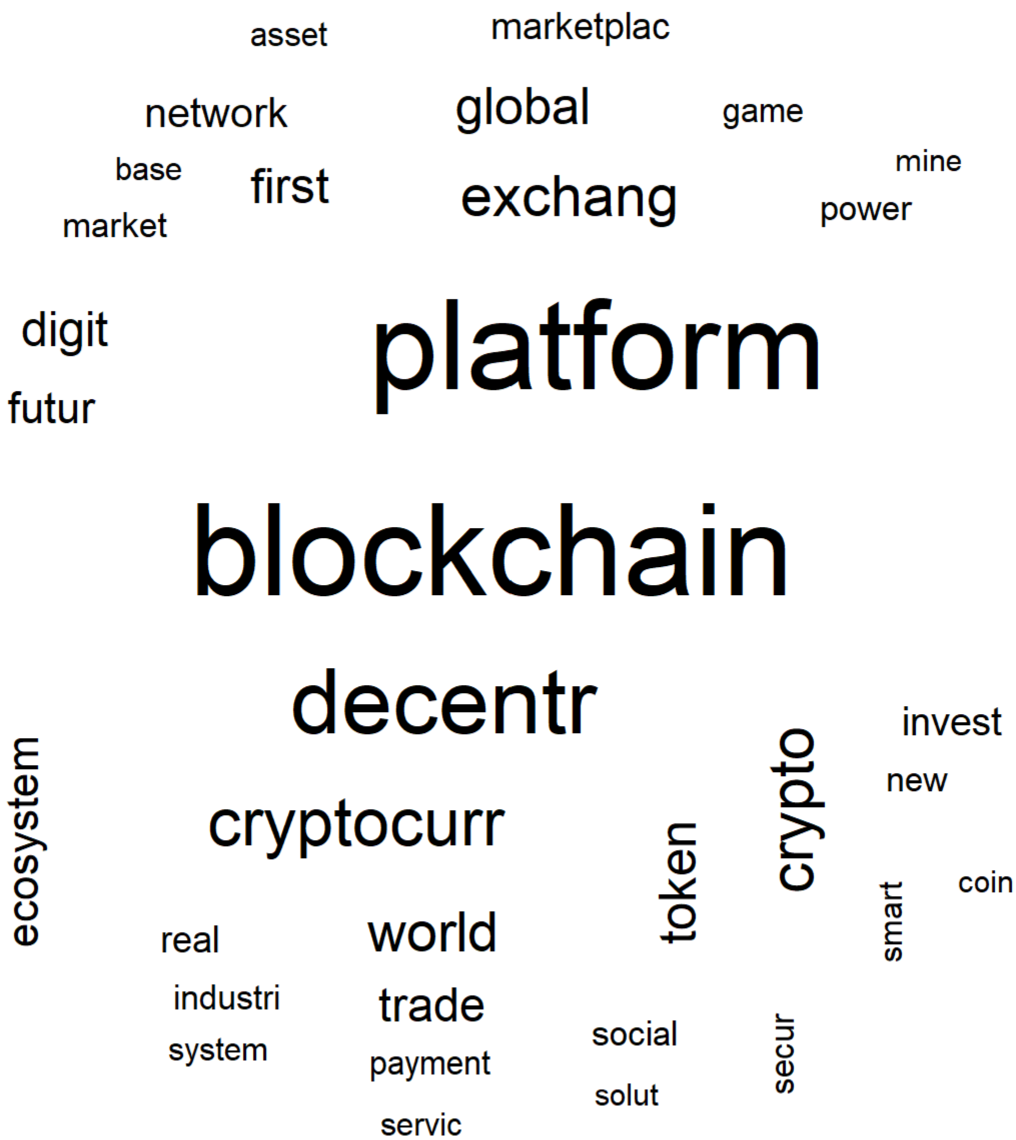
The distribution of these features displays **a similar trend for both the class labels 'Y' & 'N'** except minInvestment.

2(e). **brandSlogan**
This column indicates the slogan of each venture, logically thinking this feature will not be influential on the venture's success or failure. So, this column will be removed, but sentiment analysis in R was done to get an overview of the venturer's motives and background.

Below visuals represent the emotion conveyed by the people through their slogans.

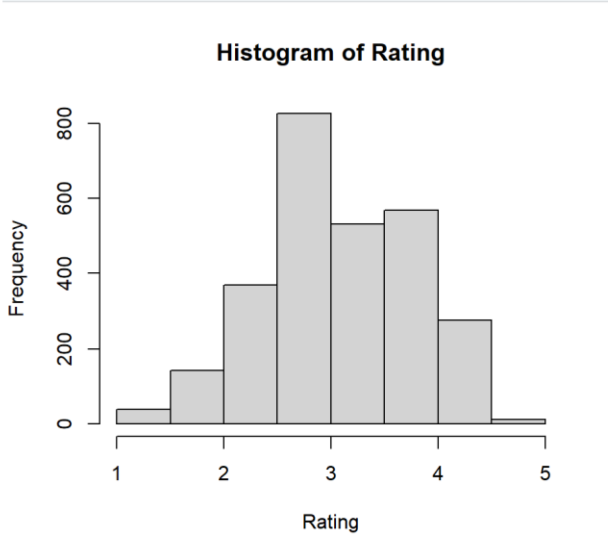


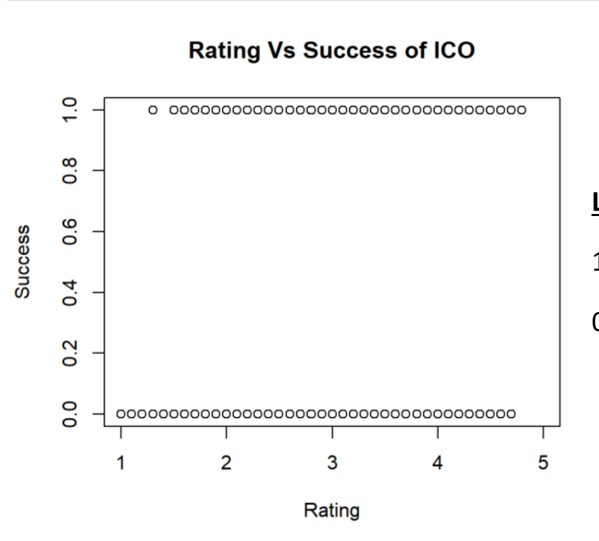
Most of the people try to convey ***'Trust', 'Positivity' & 'Anticipation'*** through their slogans which makes sense as **cryptocurrencies** are **very uncertain**, and it becomes important to make sure that the **investors feel safe and confident** about their project. Below visual gives a glimpse of most used words.


asset
marketplac
network
global
game
base
first
exchang
mine
power
market
digit
platform
futur
blockchain
decentr
invest
new
ecosystem
cryptocurr
token
crypto
smart
coin
real
world
industri
trade
social
secur
system
payment
solut
servic

2(f). Rating

This column gives the rating of each ICO based on the quality of the venture which is determined by investments experts.





Legend

1.0 → 'Y'

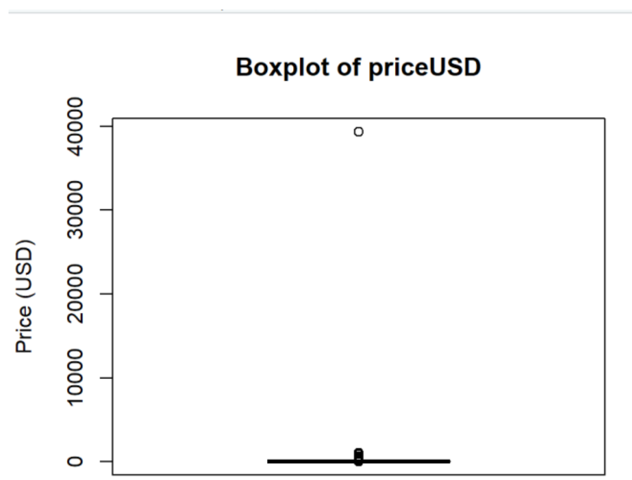
0.0 → 'N'

There were no missing values in this column and this column follows a normal distribution without outliers.

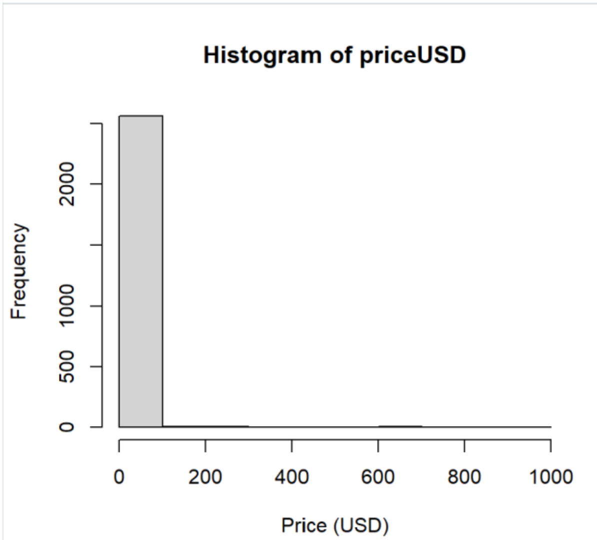
Scatter plot displays the fact that rating is very slightly biased towards class label 'Y'. The average rating for **class** label '**Y'** sums up to **3.4** while for 'N' it is **2.96**.

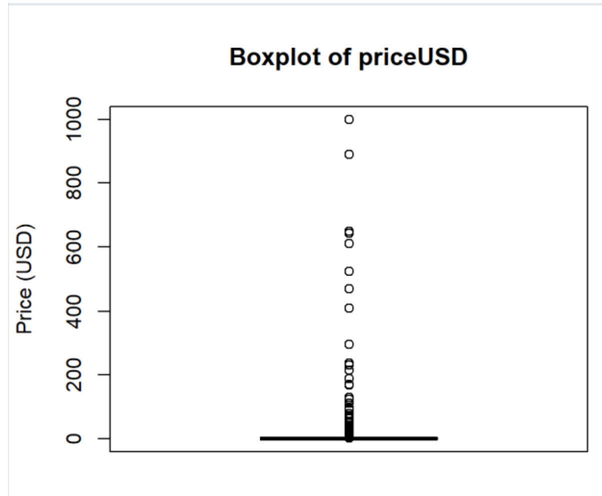
2(g). priceUSD

This is the price of each bitcoin issued by the venturers and distribution is as follows.



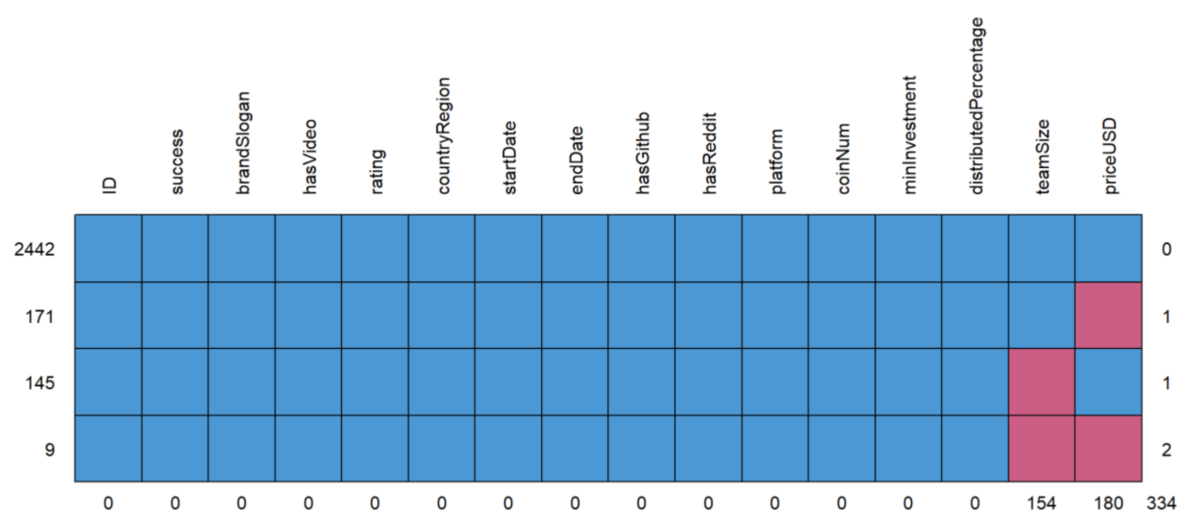
The data is right skewed because of one extreme value as shown in the boxplot making the data statistically weaker, so this record was removed.



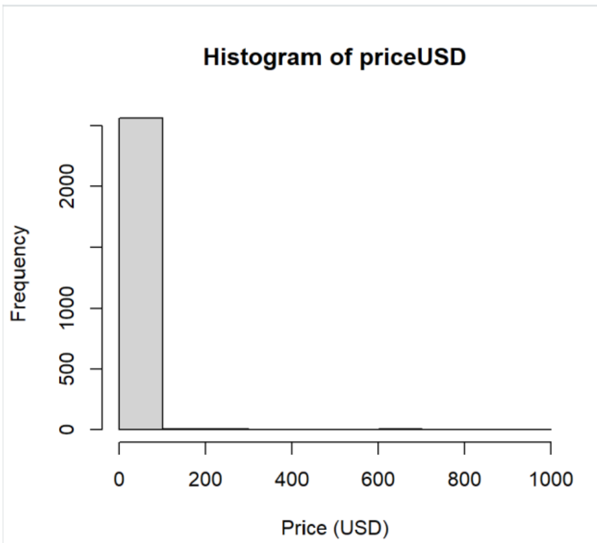



```
> summary(ico_data$priceUSD)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max. 
   0.000    0.040    0.120    3.793    0.500 1000.000 
```

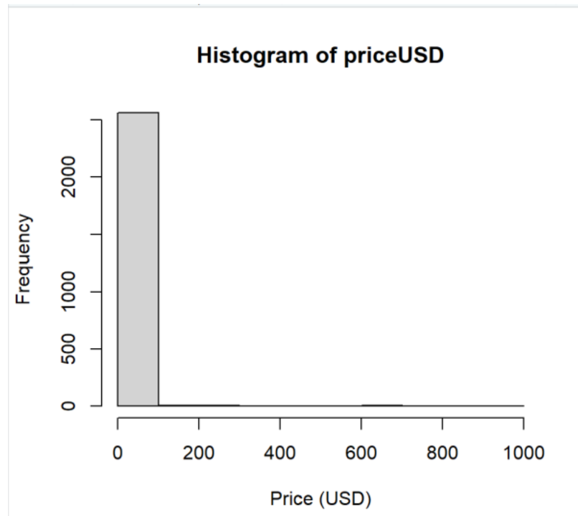
The data is still right skewed, but these **prices** are **practically possible** in the uncertain cryptocurrency world, hence the **outliers were not removed**.



This column missed **180 values**, hence **simple imputation** was used to fill in the missing values, **median was used instead of mean as mean value lies outside the third quadrant** as shown in the previous figure. Distribution of the dataset before and after imputation remained same as shown below.



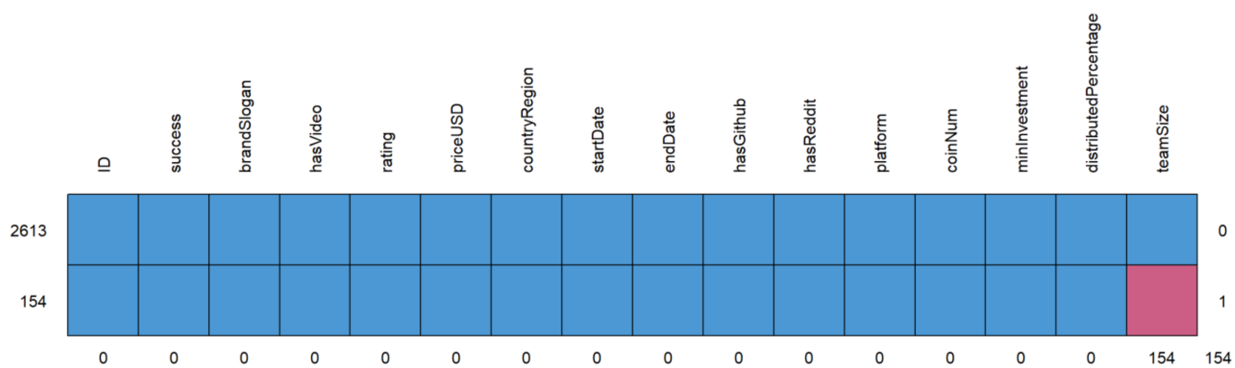
Before Imputation



After Imputation

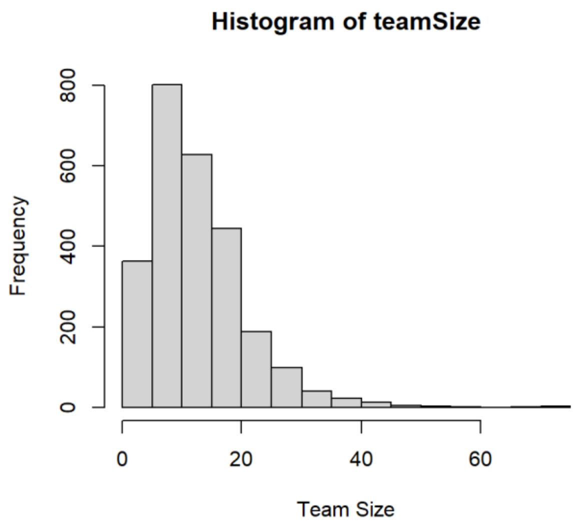
2(h). teamSize

This column represents the number of team members in each venture. This column had **154 missing** values.

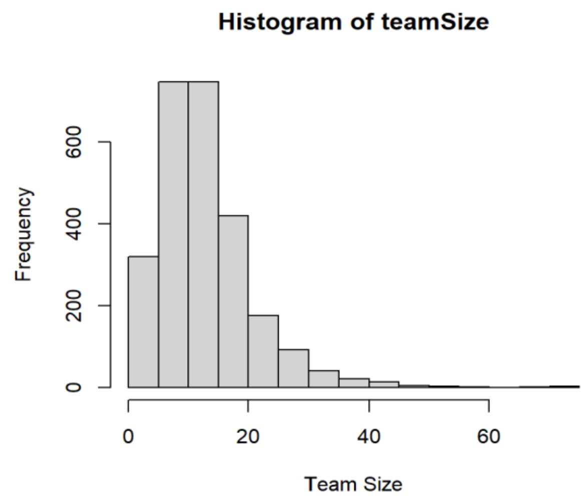



```
> summary(ico_data$teamSize)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
   1.00    7.75   12.00   13.24   17.00   75.00     145 
```

Simple imputation using the mean of the data was used to fill in the missing values. The distribution before and after imputation remained the same as shown below.

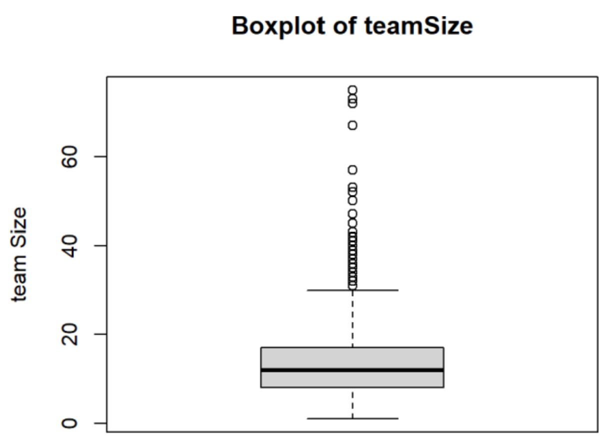


Before Imputation



After Imputation

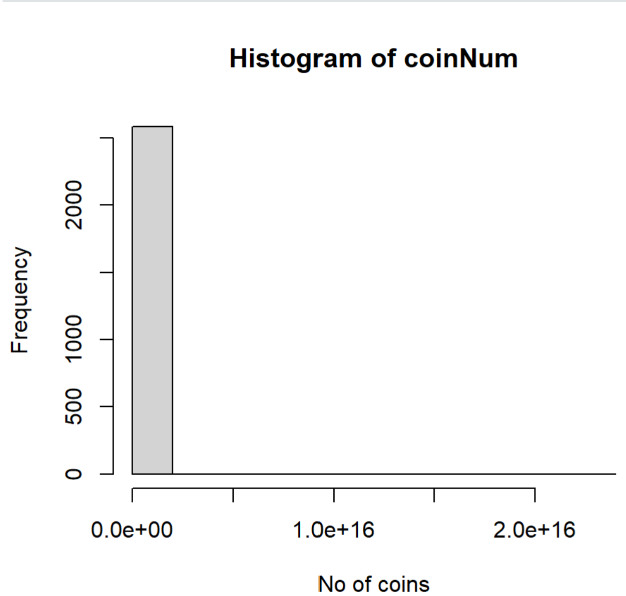
The data appears to be slightly right skewed, as explained by the boxplot.

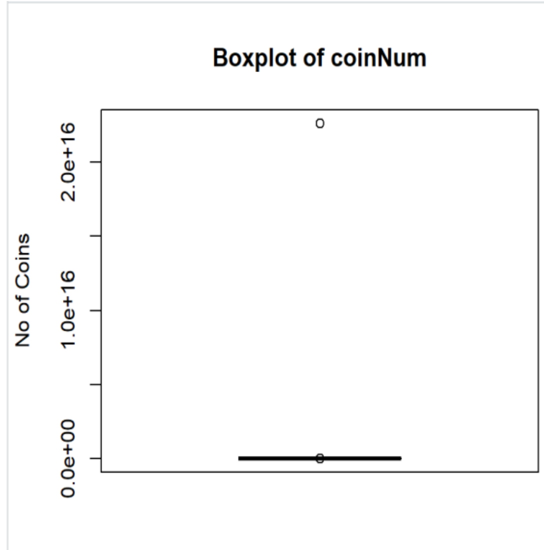


There are statistical outliers, but there isn't any global rule on the team size, so the outliers were not removed.

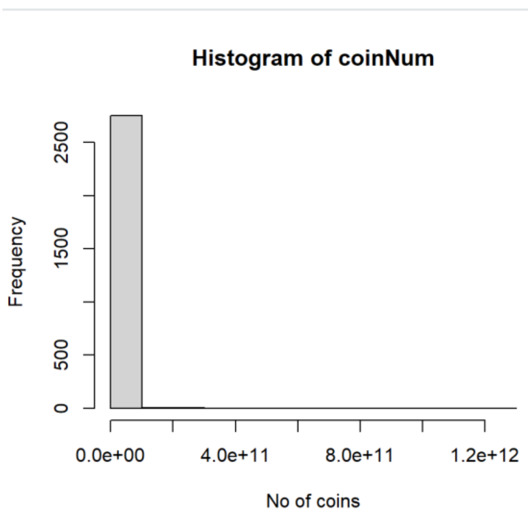
2(I). coinNum

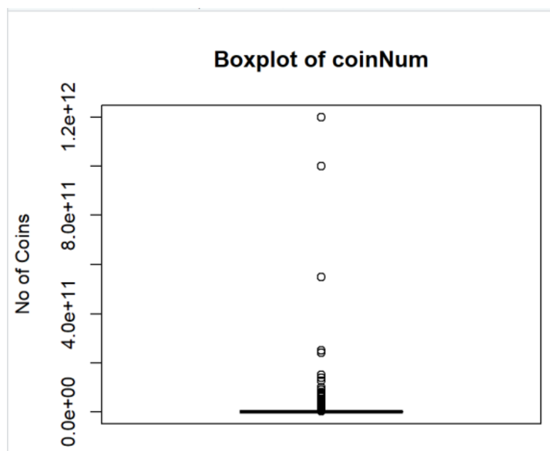
It is the number of digital coins issued by the fund-raising team; the distribution is as follows:





The data is **right-skewed** because of one extreme value making the distribution very unrealistic, hence this record removed.

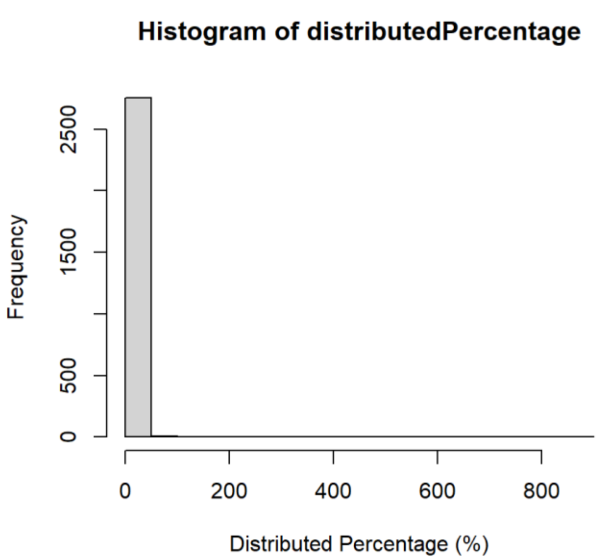


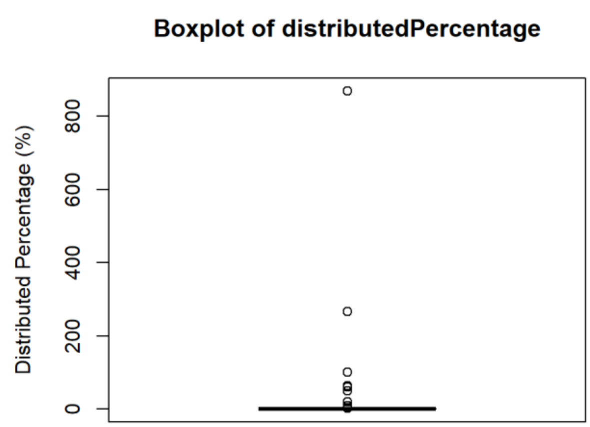


The data is still right-skewed but there is practically **no limit on the number of coins issued** by the fund-raising team and **to account in all extreme characteristics** these **outliers were not removed.**

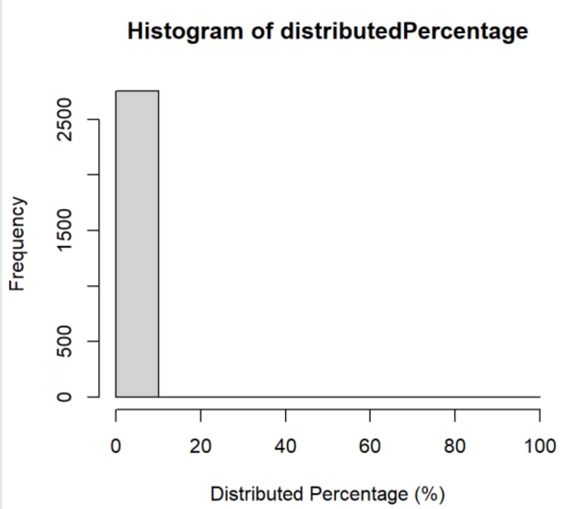
2(J). distributedPercentage

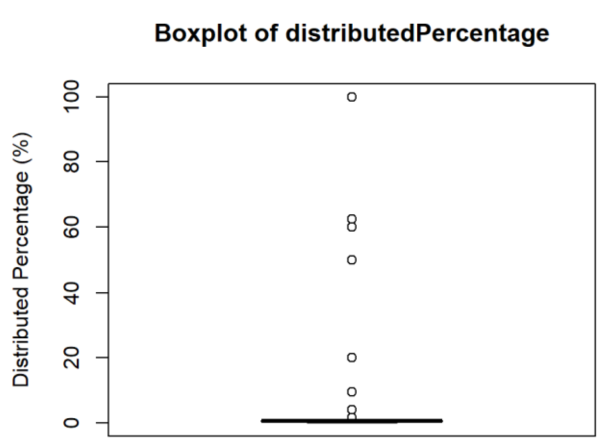
It represents the percentage of digital coins distributed to the investors.





The data is **right-skewed**, logically speaking percentage can only range from 0-100% so here records with more than 100% were removed.





The data is still right-skewed indicating the presence of outliers, but practically speaking a fund-raising team can distribute 100% of their digital coins to the investors, hence it becomes logically incorrect to remove the outliers here as well.

2(L). Relationships, Significance & Feature selection of Predictor variables

Chi-square & two-sample T test was done for categorical and numerical variables respectively keeping the **confidence interval** at **95**%. The results are as shown:

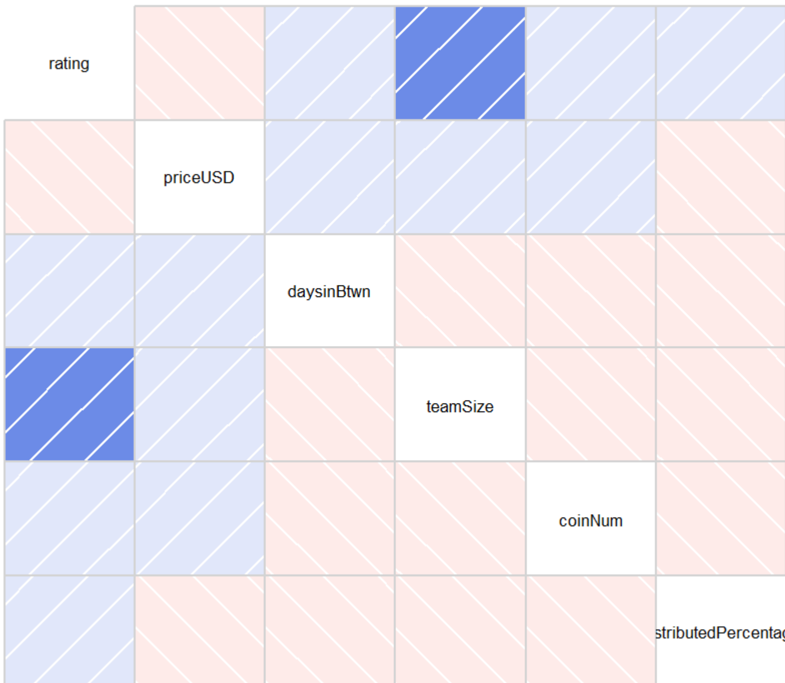
Chi-Square Test				
Dependent Variable	**Predictor Variable**	**Chi-sqr Value**	**P- value**	**Significant(Yes/No)**
Success	***Continent***	29.18202	2.14E-05	**Yes**
	Platform	16.93471	1.10E-01	**No**

Two-Sample t-test				
Dependent Variable	**Predictor Variable**	**t Value**	**P-value**	**Significant(Yes/No)**
Success	***Rating***	-167.43	**2.20E-16**	**Yes**
	priceUSD	-4.3072	1.71E-05	
	daysinBtwn	-49.698	**2.20E-16**	
	teamSize	-85.085	**2.20E-16**	
	coinNum	-4.9313	8.65E-07	
	minInvestment	-6.2238	5.21E-10	
	hasGithub	-15.608	**2.20E-16**	
	hasVideo	-28.256	**2.20E-16**	
	hasReddit	-20.035	**2.20E-16**	
	distributedPercentage	-5.3376	1.01E-07	

All the predictor variables were included in the model as all of them were significantly related to the dependent variable. ‘Platforms’ despite being insignificant was also included in the model as the data has a lot of outliers and having an extra dimension might come in handy while handling such outliers.

2(M). Collinearity

High degree of collinearity within the predictor variables will **reduce their significance** on the **dependent variables**.




```
> cor(ico_data$rating, ico_data$teamSize)
[1] 0.3823078
```

Only **teamsize** and **rating** columns have **significant correlation** between each other, but the **correlation coefficient is not significant enough to cause collinearity issues** in our model.

Thus, we can start building our model.

3. <u>Modelling</u>

3(a). Decision Tree

Decision tree (**DT**) works using '**Divide & Conquer'** concept where it identifies the most **predictive feature** and **splits** the data into **subsets** until **homogeneity is reached**. For deciding on split it initially **calculates** the **entropy** of the class variable and calculates the **entropy change** while splitting, the **feature** which **produces the highest change in entropy creates the split in that step**. This step is repeated till it reaches a level of homogeneity.

The built model is as follows:

```
Decision tree:

rating <= 3:
:...daysinBtwn > 4: no (988/231)
:   daysinBtwn <= 4:
:   :...hasReddit <= 0: no (30/12)
:       hasReddit > 0: yes (13/2)
rating > 3:
:...teamSize <= 11: no (339/122)
    teamSize > 11:
    :...rating > 3.9:
        :...daysinBtwn <= 171: yes (233/72)
        :   daysinBtwn > 171: no (21/8)
        rating <= 3.9:
```

Rating has been the **most predictive variable** in this data followed by **duration** and **team size**. *It is worth noting that all these three variables have the most significant relationship with the dependent variable as explained in the previous section.*

→ *Appendix(1)*

The diagram highlights the disadvantage of this model called '**axis-parallel split'** i.e., **it can only handle one feature at a time, making it vulnerable to special or extreme characteristics**.

Adaboosting technique was used to improve the performance of the model which is based on **training the weak learners**. **Caret** package was used to identify the right parameter as shown below:

```
Kappa was used to select the optimal model using  the one SE rule.
The final values used for the model were trials = 1, model = tree and winnow = FALSE.
```

3(b). Support Vector Machine (SVM)

SVM partitions the data by **plotting a hyperplane at a multi-dimensional feature space**. A hyperplane typically is made up of planes which split the data eventually creating a **boundary** between different class labels. A **slack variable** is introduced to deal with **non-linear data** which creates a **soft- boundary** allowing some wrong classification and a **cost** is assigned to **minimize this occurrence.**

Different kernel functions are used in SVM, **linear kernel** is used for **linearly separable data** and **Polynomial & Gaussian RBF kernel** is used for dealing with **non-linear spaces**.

In our case the data was tested with all three kernel functions and the results did not vary significantly.

Stratified Random sampling `caret::createDataPartition` was used to ensure equal proportion of class labels in both the training as well as test data, this ensures that the obtained accuracy is consistent and not biased.

Caret Package was used to obtain the most optimum value for the 'cost' parameter.

```
Accuracy was used to select the optimal model using  the one SE rule.
The final value used for the model was C = 1.
```

3(c). KNN - Classifier

The **KNN** algorithm is used when the relationships between the predictor and the class variables are difficult to understand which is the case with our data, thus using KNN makes a lot of sense. **This algorithm forms groups of data by measuring the Euclidean distance between datapoints, grouping the closer ones together**. Since the algorithm uses Euclidean distance, **it loses the capability to capture the relationships among the predictors, thus making this model vulnerable to special or extreme characteristics of the data** (Lantz, 2023).

K value determines the number of votes based on which test data will be assigned to a group by comparing the datapoint's Euclidean distance with training datapoints. K value determines the nature of the prediction, **larger K value leads to under-fitting of data, while smaller K value may lead to over-fitting of data,** finding an optimum value of K- value is the answer to avoid both unfavorable scenarios. This is called '***bias-variance tradeoff'***. (Lantz, 2023)

The thumb rule for determining the right K value is to take the ***sqrt(training data size),*** for our data it was **sqrt(2071) = 45.**

Caret package `caret::train` was used to obtain the optimum value of K.

```
Accuracy was used to select the optimal model using  the one SE rule.
The final value used for the model was k = 99.
```

Stratified Random sampling `caret::createDataPartition` was used to ensure that the proportion of class labels remained constant in both the training as well as test data.

4. Evaluation of the Models

Performance can be evaluated by statistically **analyzing the positive & negative prediction** of the test data and comparing it with actual class labels, 'Y' is positive while 'N' is negative in our case, the parameters used are as shown below: (Lantz, 2023)

- ***Accuracy*** *→ The proportion of prediction that matches with the actual class labels.*
- ***Sensitivity*** *→ The proportion of positive examples that were correctly classified.*
- ***Specificity*** *→ The proportion of negative examples that were correctly classified.*

Sensitivity & specificity might be contradicting to each other, For ex: if a model has higher sensitivity, it might produce a lot of false positives reducing the specificity, while if a model has higher specificity, it might produce a lot of false negatives reducing the sensitivity. Both the cases are an indication of biased model, having a higher and a balanced value of both these parameters is an indication of perfect model.

- **Kappa** → It measures the **agreement between the predicted and actual values** by **excluding out prediction by chance**. It is a representative of both sensitivity and specificity, higher and a balanced the value of both these, higher is the Kappa.

  (All these parameters range from 0-1)

The evaluation parameters for each body are listed below in the table, **10-fold CV** was done for every model, for **tuned model** the **tuning process has conducted CV** as reported in the table.

Data Split ***:75%*** *--> Training set,* ***25%*** *--> Test Set*					
Model	**Evaluation**	**Accuracy**	**Kappa**	**Sensitivity**	**Specificity**
Default DT	*Testing Set*	0.6705	0.2392	0.3789	0.843
	10-fold CV	0.657	0.219		
Tuned DT: Trials = 1, model = tree, winnow = FALSE (AdaBoost with 1 trial)	***selectionFunction = 'oneSE', CV = 10 folds***	***0.6705***	***0.2392***	***0.3789***	***0.843***
KSVM: C =1, Kernel = VanillaDot	*Testing Set*	0.6923	0.2661	0.3359	0.903
	10-fold CV	0.6766	0.2444		
KSVM: C =1, Kernel = polyDot	*Testing Set*	0.6909	0.262	0.332	0.903
	10-fold CV	0.6766	0.2444		
Tuned KSVM: Kernel = VanillaDot, C = 0.1	***selectionFunction = 'oneSE', CV = 10 folds***	**0.6788**	**0.249**	**0.363**	**0.865**
KNN, k = 45	*Testing Set*	0.6705	0.2232	0.332	0.8707
	10-fold CV	0.657	0.198		
Tuned KNN, k= 99	***selectionFunction = 'oneSE', CV = 10 folds***	***0.658***	***0.21***	***0.34***	***0.86***

Results from **10-fold CV** were used to compare the results as it gives more **consistent figures** than the testing set ones, tuned models have produced higher performance figures which follows a rank of Tuned **SVM> Tuned DT > Tuned KNN**.

4(a). DT vs SVM vs KNN

The data is very noisy and the **successful and unsuccessful ICO s are evenly spread among the features** making it hard for the algorithms to build a solid classification model, to deal with such data the algorithm should be able to **identify the significant features**. Logistic Regression model was used to identify such features:

```
Coefficients:
                        Estimate Std. Error z value Pr(>|z|)
(Intercept)           -4.222e+00  4.855e-01  -8.697  < 2e-16 ***

rating                 6.308e-01  8.155e-02   7.735 1.04e-14 ***

daysinBtwn            -3.111e-03  6.617e-04  -4.702 2.57e-06 ***
teamSize               3.707e-02  6.012e-03   6.167 6.96e-10 ***

continenetAfrica       1.274e+00  4.660e-01   2.735  0.00624 **
continenetAsia         1.334e+00  4.178e-01   3.193  0.00141 **
continenetEurope       1.154e+00  4.143e-01   2.786  0.00534 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

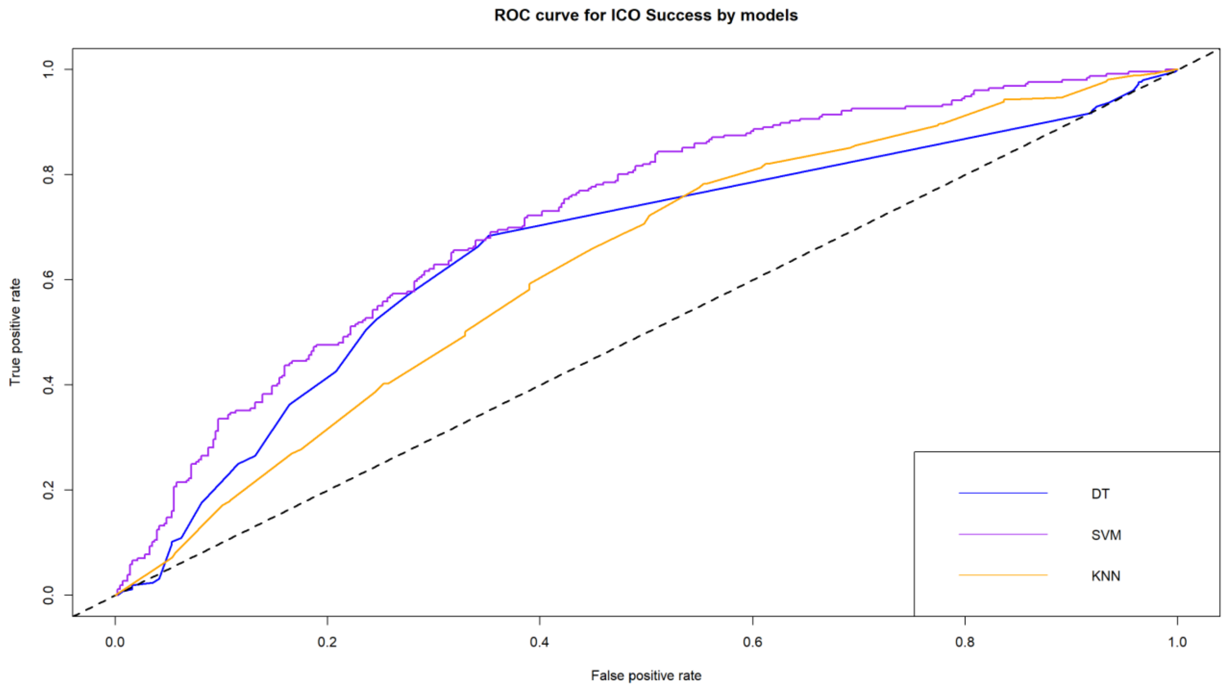
→ Appendix(4)

It can be noted that only **7-8 features are significant** out of available **27 variables**, it is important for models to identify these features to build an accurate model.

KNN cannot identify **significant features** or the insights behind each feature’s relationship making it **very vulnerable to noisy data** such as ours.

DT works by identifying the most predictive features *before splitting, thus avoiding insignificant features. This makes it* ***more efficient in dealing with noisier data****, thus producing higher performance figure than KNN, on the other hand this* ***strength*** *is also its* ***weakness****, since it considers* ***only one feature at a time*** *it lacks the* ***ability to identify the relationships among the features*** *which is a very important phenomenon in* ***handling outliers or extreme characteristics.***

SVM is the only model that overcomes the **weaknesses of both the DT and KNN** model, because SVM creates a multi-dimensional space by creating dimensions for each feature, processing the data in such high-dimensional space makes **this model less susceptible to noisier data while making it sensitive in identifying extreme characteristics or smaller pattern**. This is the reason for the performance figures following a rank order of **SVM > DT > KNN**. This ranking is clearly represented in ROC curve below:



The only **weakness** of **SVM is it is very hard to interpret and understand** as it cannot be visualized, but **DT can be easily visualized** and is the **easiest to interpret**, ***so we can use SVM to produce more accurate predictions while DT can be used to identify the most significant predictors and to understand the split of data.***

5. Conclusion & Findings

The main objective was to develop a model that correctly predicts the success of ICO. For all the models, **Accuracy** ranged between **60%-70%** with **SVM** being the **highest**, but model should also be able to distinguish between successful and unsuccessful ICOs in every scenario possible which is measured by the '**Kappa'** which ranged between **0.2-0.25** accounting for a **fair agreement** with **SVM** producing the **highest** figures again. *For models like this kappa value > 0.6 is preferred as this low kappa makes the prediction very unreliable and useless at some point.*

The reason for low performance is because of every model's ***low sensitivity but high specificity, i.e., the model can predict True Negatives accurately, while misses out on True Positives by classifying the actual positives as negatives.***

This is because, as discussed earlier, **63%** of the records belong to **unsuccessful ICOs** making the **data biased on negatives** and at the same time the **features are not distinctive enough** i.e., ***the proportion of the successful and unsuccessful ICO s are evenly spread over the range of every feature making the data noisier and more irrelevant to split effectively***.

To add on this drawback, many numeric features like **priceUSD, coinNum and distributedPercentage** have a wide range of **outliers** that logically cannot be removed, **reducing their significance on the predictor variables.**

With above analysis it is safe to say that for the given data, **SVM model comparatively produces reliable and accurate predictions**, which mostly depends upon **features like rating, location, and duration of the ICO,** while **DT** can be used to obtain the **interpretation** about the split of the data. *But to sum up, the performance figures are not sufficient to completely rely on the predictions given out by these models.*

One valuable insight that can be drawn from this analysis is that, ***given this nature of data there is a high possibility that a model like SVM which works on a multi-dimensional feature space*** *will* ***work comparatively well*** *in* ***serving this objective in the future if tried with a fresher dataset using the same features.***

REFERENCES

- Belleflamme, P., Omrani, N. and Peitz, M. 2015. The economics of crowdfunding platforms. Information economics and policy. 33, pp.11–28.

- Vujicic, D., Jagodic, D. and Randic, S. 2018. Blockchain technology, bitcoin, and Ethereum: A brief overview In: 2018 17th International Symposium INFOTEH-JAHORINA (INFOTEH). IEEE.

- Lantz, B. 2023. *Machine Learning with R* - 2nd ed. Birmingham: Packt Publishing.

Appendix

1. Decision Tree

```
Decision tree:

rating <= 3:
:...daysinBtwn > 4: no (988/231)
:   daysinBtwn <= 4:
:   :...hasReddit <= 0: no (30/12)
:       hasReddit > 0: yes (13/2)
rating > 3:
:...teamSize <= 11: no (339/122)
    teamSize > 11:
    :...rating > 3.9:
        :...daysinBtwn <= 171: yes (233/72)
        :   daysinBtwn > 171: no (21/8)
        rating <= 3.9:
        :...platform in {stellar,eos,nem,scrypt}: no (11/2)
            platform in {separate blockchain,x11,neo,waves,tron,
            :             bitcoin}: yes (20/6)
            platform = others:
            :...hasGithub <= 0: yes (3)
            :   hasGithub > 0: no (14/5)
            platform = ethereum:
            :...hasReddit <= 0: no (86/36)
                hasReddit > 0:
                :...hasGithub <= 0: yes (70/26)
                    hasGithub > 0:
                    :...continentRegion in {Africa,Oceania,
                        :                   unknown}: no (9/2)
                        continentRegion = Asia:
                        :...hasVideo <= 0: no (2)
                        :   hasVideo > 0:
                        :   :...priceUSD <= 0.11: no (30/12)
                        :       priceUSD > 0.11: yes (13/3)
                        continentRegion = Americas:
                        :...priceUSD > 0.18: no (17/2)
                        :   priceUSD <= 0.18:
                        :   :...distributedPercentage <= 0.67: yes (23/7)
                        :       distributedPercentage > 0.67: no (6)
                        continentRegion = Europe:
                        :...minInvestment > 0:
                            :...hasVideo <= 0: no (7/2)
                            :   hasVideo > 0: yes (68/25)
```


```
                            :   hasVideo > 0: yes (68/25)
```


```
                            minInvestment <= 0:
                            :...priceUSD > 1.11: no (4)
                                priceUSD <= 1.11:
                                :...priceUSD > 0.65: yes (5)
                                    priceUSD <= 0.65:
                                    :...coinNum <= 2.0482e+08: no (24/6)
                                        coinNum > 2.0482e+08: yes (37/13)
```

1.1 Decision Tree data set

```
'data.frame':   2762 obs. of  13 variables:
 $ success              : Factor w/ 2 levels "yes","no": 2 2 2 1 2 2 1 2 1 1 ...
 $ hasVideo             : int  1 1 1 1 1 1 1 1 1 1 ...
 $ rating               : num  4 4.3 4.4 4.3 4.3 4.7 4.1 4.5 4.8 4.2 ...
 $ priceUSD             : num  30 0.13 0.01 0.12 0.03 0.1 0.02 2.8 50 0.1 ...
 $ continentRegion      : chr  "Asia" "Europe" "Europe" "Europe" ...
 $ daysinBtwn           : num  0 35 365 75 125 126 58 364 42 29 ...
 $ teamSize             : num  31 20 10 27 14 43 20 31 8 29 ...
 $ hasGithub            : int  1 1 1 1 1 1 1 1 1 1 ...
 $ hasReddit            : int  1 1 1 1 1 1 1 1 1 1 ...
 $ platform             : chr  "ethereum" "others" "stellar" "separate blockchain" ...
 $ coinNum              : num  5.10e+05 2.25e+08 5.00e+09 1.25e+08 5.00e+09 ...
 $ minInvestment        : int  0 1 1 1 1 1 1 1 1 1 ...
 $ distributedPercentage: num  0.49 0.41 0.4 0.13 0.5 0.5 0.25 0.1 0.05 0.15 ...
```

1.2 Decision Tree Confusion Matrix

```
Total Observations in Table:  689

 
               | predicted success 
actual success |       yes |        no | Row Total | 
---------------|-----------|-----------|-----------|
           yes |        97 |       159 |       256 | 
               |    20.782 |     6.544 |           | 
               |     0.379 |     0.621 |     0.372 | 
               |     0.588 |     0.303 |           | 
               |     0.141 |     0.231 |           | 
---------------|-----------|-----------|-----------|
            no |        68 |       365 |       433 | 
               |    12.287 |     3.869 |           | 
               |     0.157 |     0.843 |     0.628 | 
               |     0.412 |     0.697 |           | 
               |     0.099 |     0.530 |           | 
---------------|-----------|-----------|-----------|
  Column Total |       165 |       524 |       689 | 
               |     0.239 |     0.761 |           | 
---------------|-----------|-----------|-----------|
```

```
Confusion Matrix and Statistics

          Reference
Prediction yes  no
       yes  97  68
       no  159 365

               Accuracy : 0.6705
                 95% CI : (0.634, 0.7056)
    No Information Rate : 0.6284
    P-Value [Acc > NIR] : 0.01185

                  Kappa : 0.2392

 Mcnemar's Test P-Value : 2.322e-09

            Sensitivity : 0.3789
            Specificity : 0.8430
         Pos Pred Value : 0.5879
         Neg Pred Value : 0.6966
             Prevalence : 0.3716
         Detection Rate : 0.1408
   Detection Prevalence : 0.2395
      Balanced Accuracy : 0.6109

       'Positive' Class : yes
```

1.3 Tuned DT

```
C5.0

2762 samples
  12 predictor
   2 classes: 'yes', 'no'

No pre-processing
Resampling: Cross-Validated (50 fold)
Summary of sample sizes: 2707, 2706, 2707, 2707, 2706, 2706, ...
Resampling results across tuning parameters:

  trials  Accuracy   Kappa
   1      0.6670584  0.2433104
   2      0.6713172  0.2304484
   3      0.6757273  0.2648864
   4      0.6703050  0.2387488
   5      0.6695707  0.2469623
   6      0.6718057  0.2473830
   7      0.6670839  0.2461930
   8      0.6677982  0.2435848
   9      0.6677982  0.2467378
  10      0.6677982  0.2467378
  11      0.6677982  0.2467378
  12      0.6677982  0.2467378
  13      0.6677982  0.2467378
  14      0.6677982  0.2467378
  15      0.6677982  0.2467378
  16      0.6677982  0.2467378
  17      0.6677982  0.2467378
  18      0.6677982  0.2467378
  19      0.6677982  0.2467378
  20      0.6677982  0.2467378
  21      0.6677982  0.2467378
  22      0.6677982  0.2467378
  23      0.6677982  0.2467378
  24      0.6677982  0.2467378
  25      0.6677982  0.2467378
  26      0.6677982  0.2467378
  27      0.6677982  0.2467378
```

```
  28      0.6677982  0.2467378
  29      0.6677982  0.2467378
  30      0.6677982  0.2467378
  31      0.6677982  0.2467378
  32      0.6677982  0.2467378
  33      0.6677982  0.2467378
  34      0.6677982  0.2467378
  35      0.6677982  0.2467378
  36      0.6677982  0.2467378
  37      0.6677982  0.2467378
  38      0.6677982  0.2467378
  39      0.6677982  0.2467378
  40      0.6677982  0.2467378
  41      0.6677982  0.2467378
  42      0.6677982  0.2467378
  43      0.6677982  0.2467378
  44      0.6677982  0.2467378
  45      0.6677982  0.2467378
  46      0.6677982  0.2467378
  47      0.6677982  0.2467378
  48      0.6677982  0.2467378
  49      0.6677982  0.2467378
  50      0.6677982  0.2467378

Tuning parameter 'model' was held constant at a value of tree
Tuning parameter 'winnow' was held
 constant at a value of FALSE

Kappa was used to select the optimal model using  the one SE rule.
The final values used for the model were trials = 1, model = tree and winnow = FALSE.
```

2.1 SVM Dataset

```
str(ico_data_svm)
ata.frame':   2762 obs. of  28 variables:
 success             : Factor w/ 2 levels "Y","N": 2 2 2 1 2 2 1 2 1 1 ...
 hasVideo            : int  1 1 1 1 1 1 1 1 1 1 ...
 rating              : num  4 4.3 4.4 4.3 4.3 4.7 4.1 4.5 4.8 4.2 ...
 priceUSD            : num  30 0.13 0.01 0.12 0.03 0.1 0.02 2.8 50 0.1 ...
 continenetAmericas  : num  0 0 0 0 0 0 0 0 0 0 ...
 continenetAfrica    : num  0 0 0 0 1 0 0 0 0 0 ...
 continenetAsia      : num  1 0 0 0 0 0 1 0 0 0 ...
 continenetEurope    : num  0 1 1 1 0 1 0 1 1 0 ...
 continenetOceania   : num  0 0 0 0 0 0 0 0 0 1 ...
 daysinBtwn          : num  0 35 365 75 125 126 58 364 42 29 ...
 teamSize            : num  31 20 10 27 14 43 20 31 8 29 ...
 hasGithub           : int  1 1 1 1 1 1 1 1 1 1 ...
 hasReddit           : int  1 1 1 1 1 1 1 1 1 1 ...
 pltfrmEthereum      : num  1 0 0 0 1 1 1 0 0 1 ...
 pltfrmSprtblkchn    : num  0 0 0 1 0 0 0 1 1 0 ...
 pltfrmStellar       : num  0 0 1 0 0 0 0 0 0 0 ...
 pltfrmWaves         : num  0 0 0 0 0 0 0 0 0 0 ...
 pltfrmBtcn          : num  0 0 0 0 0 0 0 0 0 0 ...
 pltfrmEos           : num  0 0 0 0 0 0 0 0 0 0 ...
 pltfrmNem           : num  0 0 0 0 0 0 0 0 0 0 ...
 pltfrmNeo           : num  0 0 0 0 0 0 0 0 0 0 ...
 pltfrmScrpt         : num  0 0 0 0 0 0 0 0 0 0 ...
 pltfrmTrn           : num  0 0 0 0 0 0 0 0 0 0 ...
 pltfrmX11           : num  0 0 0 0 0 0 0 0 0 0 ...
 pltfrmOthers        : num  0 1 0 0 0 0 0 0 0 0 ...
 coinNum             : num  5.10e+05 2.25e+08 5.00e+09 1.25e+08 5.00e+09 ...
 minInvestment       : int  0 1 1 1 1 1 1 1 1 1 ...
 distributedPercentage: num  0.49 0.41 0.4 0.13 0.5 0.5 0.25 0.1 0.05 0.15 ...
```

2.2 Tuned SVM

```
Support Vector Machines with Linear Kernel

2762 samples
  27 predictor
   2 classes: 'Y', 'N'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 2486, 2486, 2485, 2487, 2486, 2487, ...
Resampling results across tuning parameters:

  C   Accuracy   Kappa
   1  0.6774405  0.2468900
   2  0.6770769  0.2462032
   3  0.6774405  0.2468900
   4  0.6770769  0.2462032
   5  0.6770782  0.2462299
   6  0.6774405  0.2468900
   7  0.6774405  0.2468900
   8  0.6774405  0.2468900
   9  0.6774405  0.2468900
  10  0.6770769  0.2462032

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was C = 1.
```

2.3 SVM confusion Matrix

```
Total Observations in Table:  689


               | predicted success
actual success |          Y |          N | Row Total |
---------------|------------|------------|-----------|
             Y |         86 |        170 |       256 |
               |     31.072 |      7.089 |           |
               |      0.336 |      0.664 |     0.372 |
               |      0.672 |      0.303 |           |
               |      0.125 |      0.247 |           |
---------------|------------|------------|-----------|
             N |         42 |        391 |       433 |
               |     18.370 |      4.191 |           |
               |      0.097 |      0.903 |     0.628 |
               |      0.328 |      0.697 |           |
               |      0.061 |      0.567 |           |
---------------|------------|------------|-----------|
  Column Total |        128 |        561 |       689 |
               |      0.186 |      0.814 |           |
---------------|------------|------------|-----------|
```

```
Confusion Matrix and Statistics

          Reference
Prediction   Y   N
         Y  86  42
         N 170 391

               Accuracy : 0.6923
                 95% CI : (0.6563, 0.7266)
    No Information Rate : 0.6284
    P-Value [Acc > NIR] : 0.0002584

                  Kappa : 0.2661

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.3359
            Specificity : 0.9030
         Pos Pred Value : 0.6719
         Neg Pred Value : 0.6970
             Prevalence : 0.3716
         Detection Rate : 0.1248
   Detection Prevalence : 0.1858
      Balanced Accuracy : 0.6195

       'Positive' Class : Y
```

3.1 KNN Data set

```
> str(ico_data_knn)
'data.frame':	2762 obs. of  28 variables:
 $ success              : Factor w/ 2 levels "Y","N": 2 2 2 1 2 2 1 2 1 1 ...
 $ hasVideo             : int  1 1 1 1 1 1 1 1 1 1 ...
 $ rating               : num  4 4.3 4.4 4.3 4.3 4.7 4.1 4.5 4.8 4.2 ...
 $ priceUSD             : num  30 0.13 0.01 0.12 0.03 0.1 0.02 2.8 50 0.1 ...
 $ continenetAmericas   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ continenetAfrica     : num  0 0 0 0 1 0 0 0 0 0 ...
 $ continenetAsia       : num  1 0 0 0 0 0 1 0 0 0 ...
 $ continenetEurope     : num  0 1 1 1 0 1 0 1 1 0 ...
 $ continenetOceania    : num  0 0 0 0 0 0 0 0 0 1 ...
 $ daysinBtwn           : num  0 35 365 75 125 126 58 364 42 29 ...
 $ teamSize             : num  31 20 10 27 14 43 20 31 8 29 ...
 $ hasGithub            : int  1 1 1 1 1 1 1 1 1 1 ...
 $ hasReddit            : int  1 1 1 1 1 1 1 1 1 1 ...
 $ pltfrmEthereum       : num  1 0 0 0 1 1 1 0 0 1 ...
 $ pltfrmSprtblkchn     : num  0 0 0 1 0 0 0 1 1 0 ...
 $ pltfrmStellar        : num  0 0 1 0 0 0 0 0 0 0 ...
 $ pltfrmWaves          : num  0 0 0 0 0 0 0 0 0 0 ...
 $ pltfrmBtcn           : num  0 0 0 0 0 0 0 0 0 0 ...
 $ pltfrmEos            : num  0 0 0 0 0 0 0 0 0 0 ...
 $ pltfrmNem            : num  0 0 0 0 0 0 0 0 0 0 ...
 $ pltfrmNeo            : num  0 0 0 0 0 0 0 0 0 0 ...
 $ pltfrmScrpt          : num  0 0 0 0 0 0 0 0 0 0 ...
 $ pltfrmTrn            : num  0 0 0 0 0 0 0 0 0 0 ...
 $ pltfrmX11            : num  0 0 0 0 0 0 0 0 0 0 ...
 $ pltfrmOthers         : num  0 1 0 0 0 0 0 0 0 0 ...
 $ coinNum              : num  5.10e+05 2.25e+08 5.00e+09 1.25e+08 5.00e+09 ...
 $ minInvestment        : int  0 1 1 1 1 1 1 1 1 1 ...
 $ distributedPercentage: num  0.49 0.41 0.4 0.13 0.5 0.5 0.25 0.1 0.05 0.15 ...
>
```

3.2 KNN Confusion Matrix

```
Total Observations in Table:  691


               | predicted success
actual success |         Y |         N | Row Total |
---------------|-----------|-----------|-----------|
             Y |        81 |       182 |       263 |
               |    15.970 |     3.949 |           |
               |     0.308 |     0.692 |     0.381 |
               |     0.591 |     0.329 |           |
               |     0.117 |     0.263 |           |
---------------|-----------|-----------|-----------|
             N |        56 |       372 |       428 |
               |     9.813 |     2.427 |           |
               |     0.131 |     0.869 |     0.619 |
               |     0.409 |     0.671 |           |
               |     0.081 |     0.538 |           |
---------------|-----------|-----------|-----------|
  Column Total |       137 |       554 |       691 |
               |     0.198 |     0.802 |           |
---------------|-----------|-----------|-----------|
```

```
Confusion Matrix and Statistics

          Reference
Prediction   Y   N
         Y  81  56
         N 182 372

               Accuracy : 0.6556
                 95% CI : (0.6188, 0.691)
    No Information Rate : 0.6194
    P-Value [Acc > NIR] : 0.02688

                  Kappa : 0.1952

 Mcnemar's Test P-Value : 5.382e-16

            Sensitivity : 0.3080
            Specificity : 0.8692
         Pos Pred Value : 0.5912
         Neg Pred Value : 0.6715
             Prevalence : 0.3806
         Detection Rate : 0.1172
   Detection Prevalence : 0.1983
      Balanced Accuracy : 0.5886

       'Positive' Class : Y
```

4. Logistic Regression Model

```
glm(formula = success ~ ., family = binomial, data = ico_data1)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2823  -0.9290  -0.6463   1.1077   2.5257

Coefficients:
                        Estimate Std. Error z value Pr(>|z|)
(Intercept)           -4.222e+00  4.855e-01  -8.697  < 2e-16 ***
hasVideo               2.077e-01  1.109e-01   1.874  0.06094 .
rating                 6.308e-01  8.155e-02   7.735 1.04e-14 ***
priceUSD               2.485e-04  1.109e-03   0.224  0.82278
continenetAmericas     1.024e+00  4.204e-01   2.437  0.01482 *
continenetAfrica       1.274e+00  4.660e-01   2.735  0.00624 **
continenetAsia         1.334e+00  4.178e-01   3.193  0.00141 **
continenetEurope       1.154e+00  4.143e-01   2.786  0.00534 **
continenetOceania      8.910e-01  4.830e-01   1.845  0.06508 .
daysinBtwn            -3.111e-03  6.617e-04  -4.702 2.57e-06 ***
teamSize               3.707e-02  6.012e-03   6.167 6.96e-10 ***
hasGithub              4.281e-02  9.754e-02   0.439  0.66075
hasReddit              1.971e-01  1.027e-01   1.919  0.05494 .
pltfrmEthereum        -7.214e-02  2.058e-01  -0.351  0.72591
pltfrmSprtblkchn       4.487e-01  4.038e-01   1.111  0.26648
pltfrmStellar         -9.902e-01  4.316e-01  -2.294  0.02177 *
pltfrmWaves            5.898e-03  3.676e-01   0.016  0.98720
pltfrmBtcn            -1.397e+00  1.070e+00  -1.305  0.19187
pltfrmEos             -8.317e-01  5.931e-01  -1.402  0.16085
pltfrmNem             -1.161e+00  7.033e-01  -1.651  0.09881 .
pltfrmNeo             -5.282e-02  4.951e-01  -0.107  0.91505
pltfrmScrpt            4.581e-01  6.327e-01   0.724  0.46903
pltfrmTrn             -2.088e-01  7.514e-01  -0.278  0.78111
pltfrmX11             -1.434e+00  1.111e+00  -1.291  0.19668
coinNum               -2.555e-12  2.389e-12  -1.070  0.28482
minInvestment          7.680e-02  8.680e-02   0.885  0.37628
distributedPercentage  3.369e-04  1.511e-02   0.022  0.98221
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```