Contents

1 Introduction **1**

2 Data Understanding & Preparation **2**

2.1 Numerical Data . . . 2

2.1.1 Scatter Plot Matrix Of Normalised Numerical Variables . . . 2

2.1.2 Correlation Matrix Of Normalised Numerical Variables . . . 3

2.1.3 Box Plots Of Normalised Numerical Variables . . . 4

2.1.4 Histograms Of Normalised Numerical Variables . . . 4

2.2 Categorical Data . . . 5

2.2.1 Bar Plots Of Binary Variables . . . 5

2.2.2 Bar Plots Of Platform's Success Rate . . . 5

2.2.3 Bar Plots Of Country's Success Rate . . . 6

2.2.4 Bar Plots Of Category's Success Rate . . . 6

3 Modelling **7**

3.1 Decision Classification Trees . . . 7

3.1.1 Prediction Accuracy For Each Model Selection With 10-Fold CV . . . 7

3.1.2 Visualisation Of Highest Performing Decision Tree Model . . . 8

3.2 Random Forest . . . 8

3.2.1 Prediction Accuracy For Each Model Selection With 5-Fold CV . . . 8

3.2.2 Variable Importance Of Highest Performing Random Forest Model . . . 9

3.3 K-Nearest Neighbour . . . 9

3.3.1 Scatter Plot Of Prediction Accuracy For $k \in \{1,..,100\}$ . . . 9

3.4 Support Vector Machine . . . 10

3.4.1 Heat-Map Scatter Plot Of Predictive Performance For Each Model Combination . . . 10

3.5 Modeling Summary Tabular . . . 10

4 Evaluation **11**

4.0.1 Confusion Matrices . . . 11

4.0.2 ROC Curves & AUC Values . . . 12

4.0.3 Model Stability . . . 12

5 Deployment **13**

1 Introduction

In an initial coin offering (ICO), new ventures raise capital by selling tokens to a crowd of investors. Often, this token is a cryptocurrency, a digital medium of value exchange based on the distributed ledger technology [1]. If the money raised in an ICO is less than the minimum amount required by the ICO's criteria, then all of the money is returned to the project's investors. The ICO would then be deemed unsuccessful. Given a dataset containing 19 variables about 1181 ICO projects from different startups, **our aim is to apply machine learning models to predict whether a startup will reach its fundraising goal successfully through the ICO**. We shall focus on applying decision trees, random forest, k-nearest neighbour and support vector machine algorithms, determine advantages and disadvantages of each model, predictive performances with evaluation techniques applied.

2 Data Understanding & Preparation

Before we apply such algorithms, we must wrangle and visualise our dataset in order to find a subset of the 19 variables that are most optimum to train our machine learning models. The required output is **whether a startup will reach its fundraising goal successfully through the ICO**. We will focus on partitioning our dataset to visualize relationships with the **goal** output, colour coding "Y" as green and "N" as red in our visualizations.

2.1 Numerical Data

There are 11 numerical variables in the ICO dataset. We combine **end_date** and **start_date** to show project duration in days with **date_diff**. Negative project duration values are imputed as the median duration, 31 days (3 observations contained negative days). The variables with interesting relations can be seen in the scatter plot matrix below.

2.1.1 Scatter Plot Matrix Of Normalised Numerical Variables

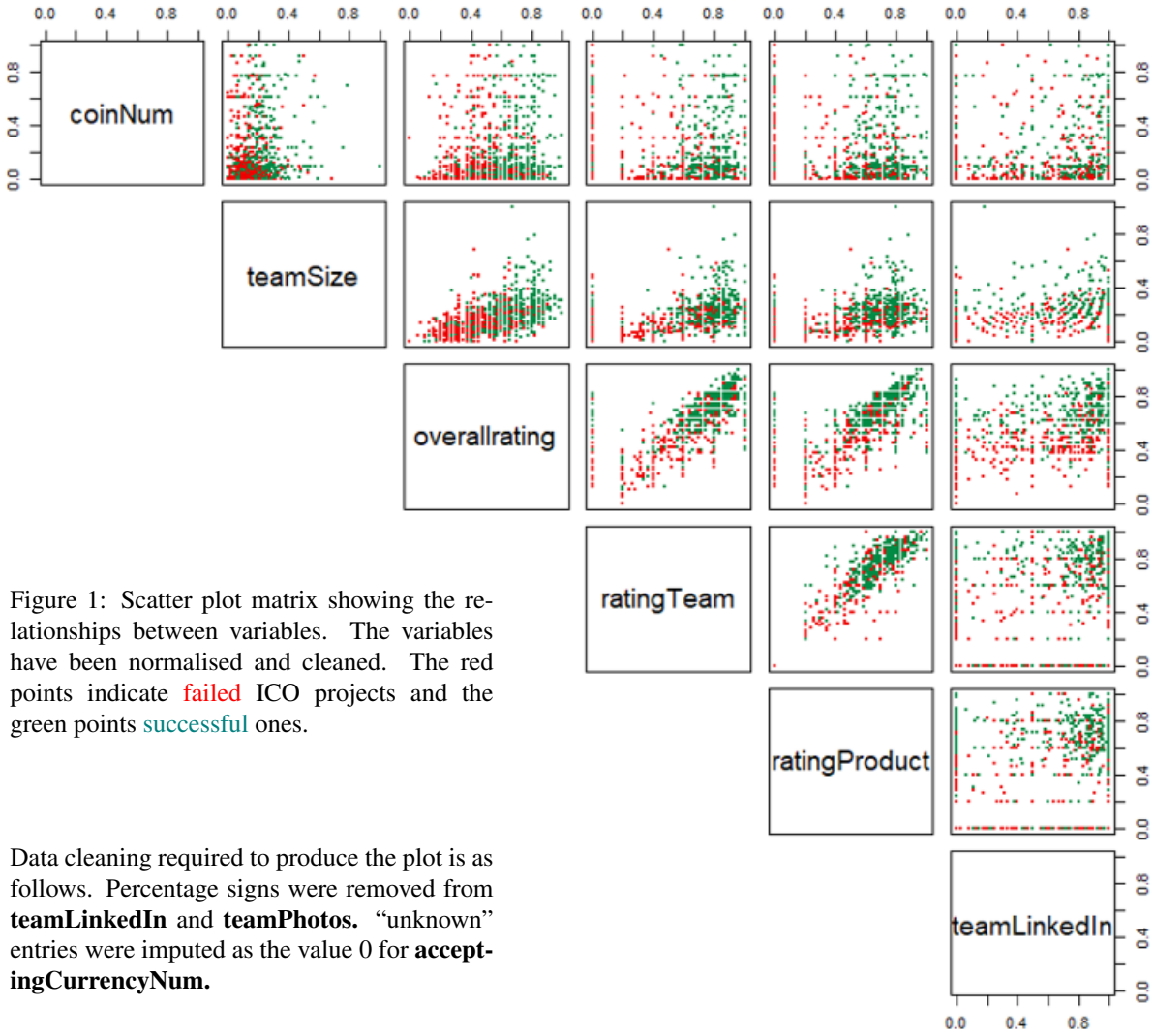


Figure 1: Scatter plot matrix showing the relationships between variables. The variables have been normalised and cleaned. The red points indicate failed ICO projects and the green points successful ones.

Data cleaning required to produce the plot is as follows. Percentage signs were removed from **teamLinkedIn** and **teamPhotos.** "unknown" entries were imputed as the value 0 for **acceptingCurrencyNum.**

Outliers for the **coinNum** variable were imputed as the median value. We defined outliers to be values outside the interval I such that

$$I = [q_{0.25} - 1.5 \cdot IQR, q_{0.75} + 1.5 \cdot IQR],$$

where $q_{0.25}$ and $q_{0.75}$ denote the upper and lower quartiles and $IQR = q_{0.75} - q_{0.25}$ the interquartile range. 144 ICO projects took **coinNum** values outside I. The justifications of these cleaning procedures are so that we can usefully format the data whilst we maintain possible relational trends to be inferred from plots.

2.1.2 Correlation Matrix Of Normalised Numerical Variables

Clusters of successful projects can be seen in **figure (1)** by the green points. This suggests training machine learning models on this numeric data may yield fruitful results. The variables have been normalised to take values between 0 and 1. We can already make some interesting inferences. For instance, the greater the team size the more likely a project will succeed, this is shown by the vast amount of green points for large **teamSize** input values. The equivalent statement holds for **overallrating**, **ratingTeam** and **ratingProduct**.
Relational trends between variables are present. For example, as the team rating of a project increases we find the product rating also increases. The inputs are correlated. It is useful to quantify correlation coefficients of all variable combinations for future use. A correlation coefficient matrix can be seen in **figure (2)**.

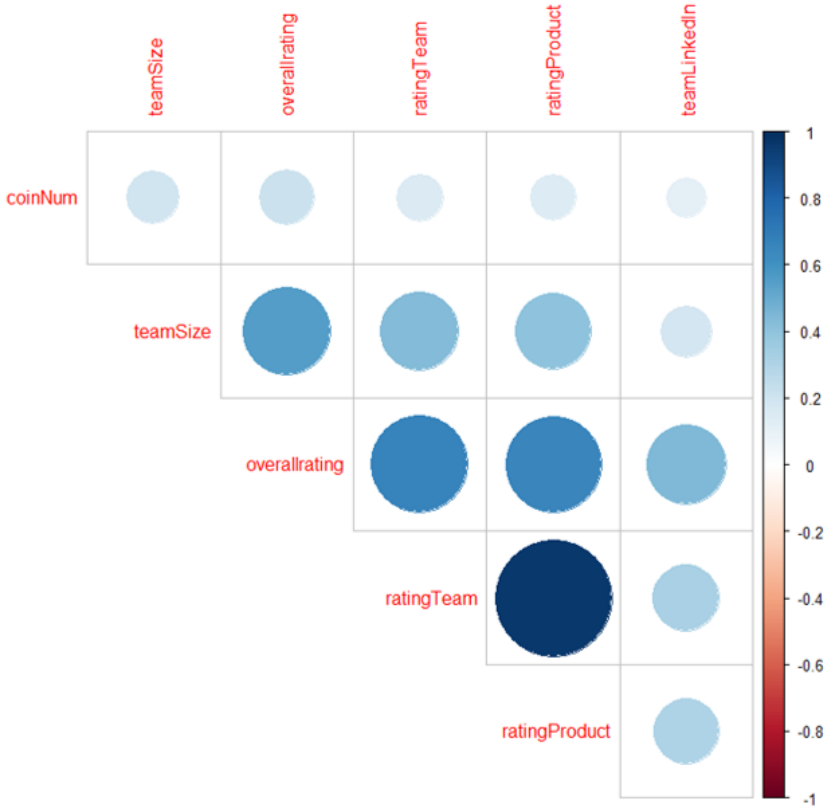


Figure 2: Correlation matrix for the equivalent variables used in **figure (1)**. The greater the size of the circle the greater the correlation between inputs. A blue colour denotes positive correlation.

2.1.3 Box Plots Of Normalised Numerical Variables

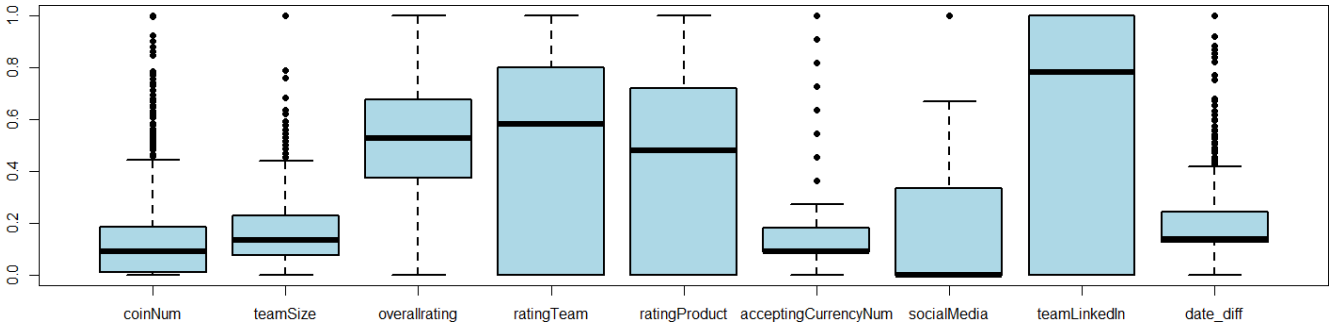


Figure 3: Box plots of all numeric input variables. Values are normalised in the interval 0 to 1. The spread of the data can be seen, black dots indicate potential outliers, these are values outside the interval I defined prior. 50% of data values span the blue boxes. Thus coin numbers and team size tend to span lower values and overall rating and team photos higher values. Number of accepted currency is skewed due to the "unknown" values being imputed to take the value 0.

2.1.4 Histograms Of Normalised Numerical Variables

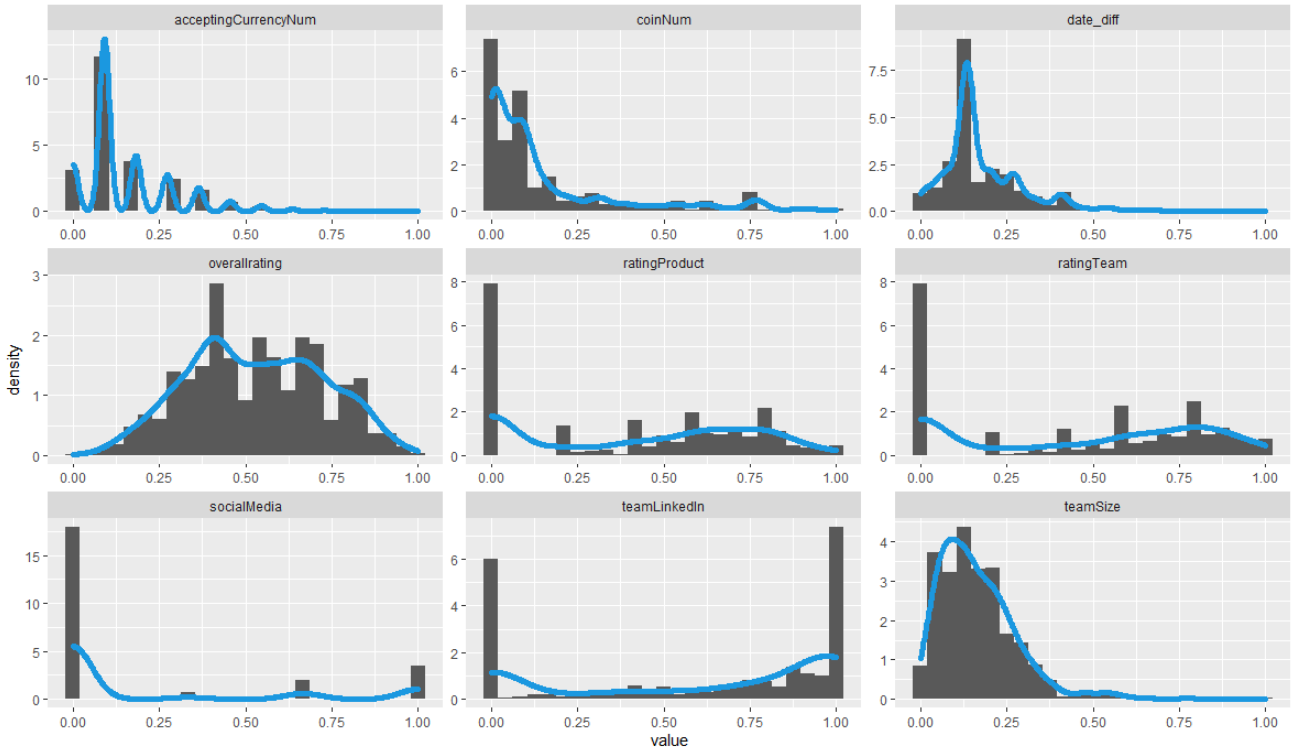


Figure 4: Histograms of normalised numerical variables. The density of the variables is shown by the grey bins. The distribution can be seen by the blue curves. **overallrating** takes a Gaussian distribution, **coinNum** , **teamSize** and **date_diff** a poisson distribution with high densities at lower relative values. The remaining variables are more difficult to interpret.

2.2 Categorical Data

2.2.1 Bar Plots Of Binary Variables

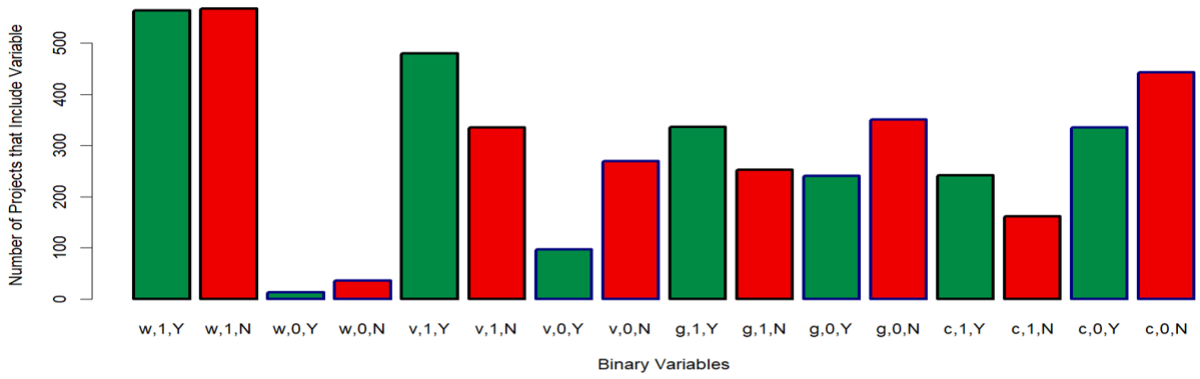


Figure 5: Bar plots of the binary variables **whitepaper**, **video**, **GitHub** and **CEOPhoto**. We denote the variables by w,v,g and c respectively and partition their outputs into bars with values 1 or 0 and projects success "Y" or "N". Green bars denote successful projects and red unsuccessful.

Many inferences can be made from **figure (5)**. The vast amount of 1181 projects contain a whitepaper and a video of their campaign page as shown by the large bar plots in the output "1" charts of **whitepaper** and **video**. An interesting observation is that projects with video campaigns and a GitHub page succeed more than projects without a video or GitHub page. This is shown by the larger green bars compared to red in each of the variables **video** and **GitHub** with outputs "1". They are reasonable indicators that could be used to train our machine learning models.

2.2.2 Bar Plots Of Platform's Success Rate

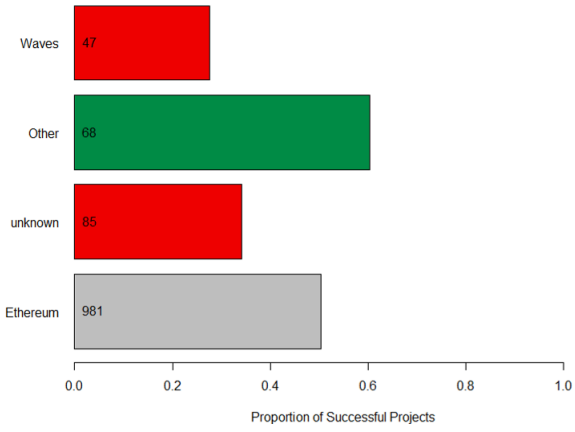


Figure 6: Bar plots showing the proportion of successful projects for each platform. Green bars indicate countries that have at minimum a 60% success rate, red bars are poor performing coins, and grey neutral coins. Bar labels denote the frequency of the coin in the dataset.

Cleaning was required on the variable **platform**. Abbreviations, spelling mistakes and capitalization of coin platforms were fixed. For instance, "ETH" and "Ehtereum" was imputed as "Ethereum", "WAVES" to "Waves" and "Stellar Consensus Protocol (SCP)" to "Stellar".
Platforms with a frequency of 10 or less were grouped together into the category "Other". In **figure (6)**, we find the vast majority of ICO projects use Ethereum as their platform, 981 projects with a neutral success rate of 50.4% seen by the grey bar.
Projects with an unknown platform or "Waves" platform tend to perform poorly with success rates around 30% as shown by the red bars. The grouped platforms tend to perform well, with a success rate more than 60% as shown by the green bar.

2.2.3 Bar Plots Of Country's Success Rate

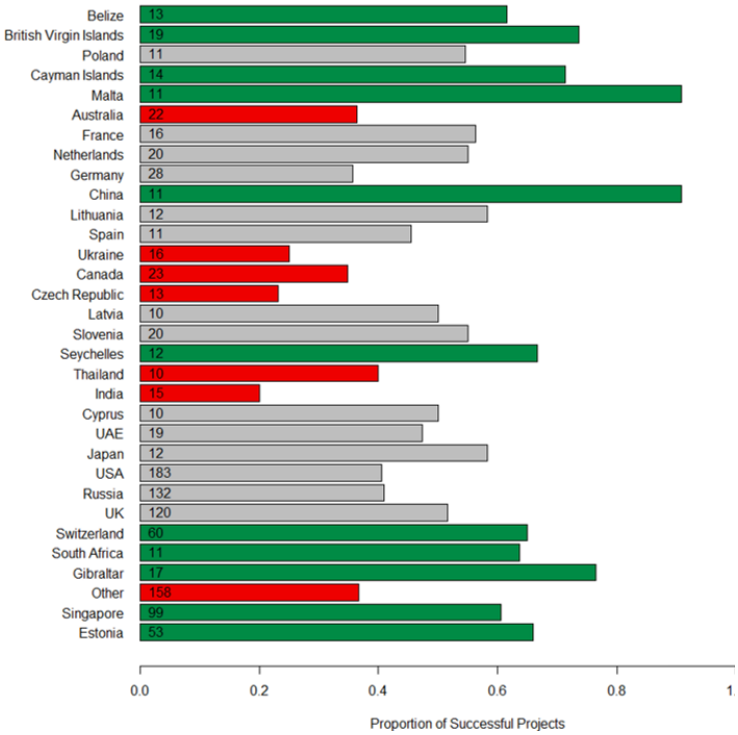


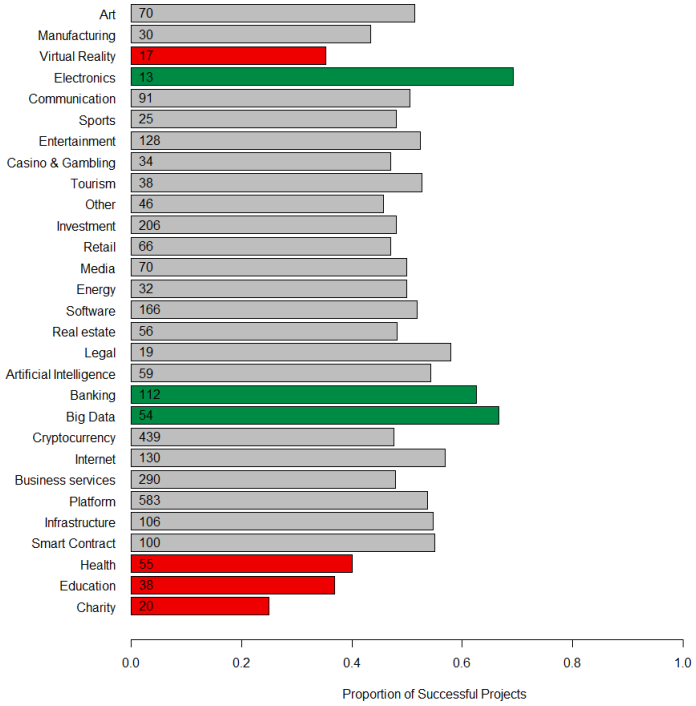
Figure 7: Bar plots showing the proportion of successful projects for each country. See **figure (6)** for equivalent plot explanation. Here the success rate thresholds are 60% and 40%.

The cleaning for **country_region** is as follows. Equivalent countries were clustered with the same value. For instance values "United Kingdom" , "Ireland" and "England" were imputed to take the value "UK". Similar imputations were applied to other countries such as the USA and the UAE. Spelling mistakes and incorrect values were fixed. For example, the value "PerÃº" and the city "Ras al-Khaimah" were converted to take values "Peru" and "UAE" respectively. Countries with a frequency in our dataset of less than 10 were grouped into the category "Other". This left 32 countries to explore and our shown in **figure (7)**. The highest success rates are found in China and Malta at 91%. Inferences could be questioned due to small country sample sizes.

2.2.4 Bar Plots Of Category's Success Rate

Figure 8: Bar plots showing the proportion of successful projects for each category. Chart explanation is equivalent to that found in **figure (6)**.

Projects may take multiple categories seen in **figure (8)**. The best performing categories are electronics, banking and big data with project success rates greater than 60% seen in green. The worst performing is virtual reality, health, education and charitable categories, with success rate less than 40%. The remaining lie in-between these success thresholds.



3 Modelling

3.1 Decision Classification Trees

Decision Classification Trees are a type of supervised machine learning that iteratively partitions categorical data with the goal of classifying the output variable. Partitions are subject to maximizing homogeneity within the partitioned subgroups. [2] Justification for model selection stems from its unique strength flexibility and simplicity for partitioning data sets as a function of the input feature space. Common metrics used to quantify the homogeneity of a subgroup are entropy and gini impurity, with popular algorithms applying this metric being **C5.0** and **rpart** respectively. We apply the rpart algorithm to predict whether a startup will reach its fundraising goal successfully through the ICO.

The input variables considered are **whitepaper**, **video**, **GitHub**, **CEOPhoto**, **USA**, **Banking** and **Unknown**. The later three variables are manually constructed variables, taking input "1" - the project is situated in the USA, inclusive of a banking category or uses an unknown platform, and input "0" - the complement. Reasoning behind such variable constructions are as follows. USA projects tend to perform below average with almost a 40% success rate as seen in **figure (7)**, banking above 60% seen in **figure (8)** and unknown platforms extremely low at 30%, see **figure (6)** for reference. All variable frequencies are sufficiently large to warrant investigation.

We investigate the seven input variables by performing model selection. The idea is to compute decision classification trees for every combination of the seven inputs and compare predictive performances, choosing the input combination with greatest predictive performance as the most desirable model to predict whether a startup will reach its fundraising goal successfully through the ICO. Thus, excluding the empty input set, there are

$$\binom{7}{1} + \ldots + \binom{7}{7} = 127 \text{ input variable combinations.}$$

Cross-validation is a strategy used to increase prediction accuracy, giving more insight into decision classification tree performance. 10-fold cross-validation is applied and so the total number of decision tree models produced is $10 \times 127 = 1270$ models. Training and testing sets are partitioned randomly with an 80%:20% split for all folds. This will be the underlying split for all models considered in this report.

3.1.1 Prediction Accuracy For Each Model Selection With 10-Fold CV

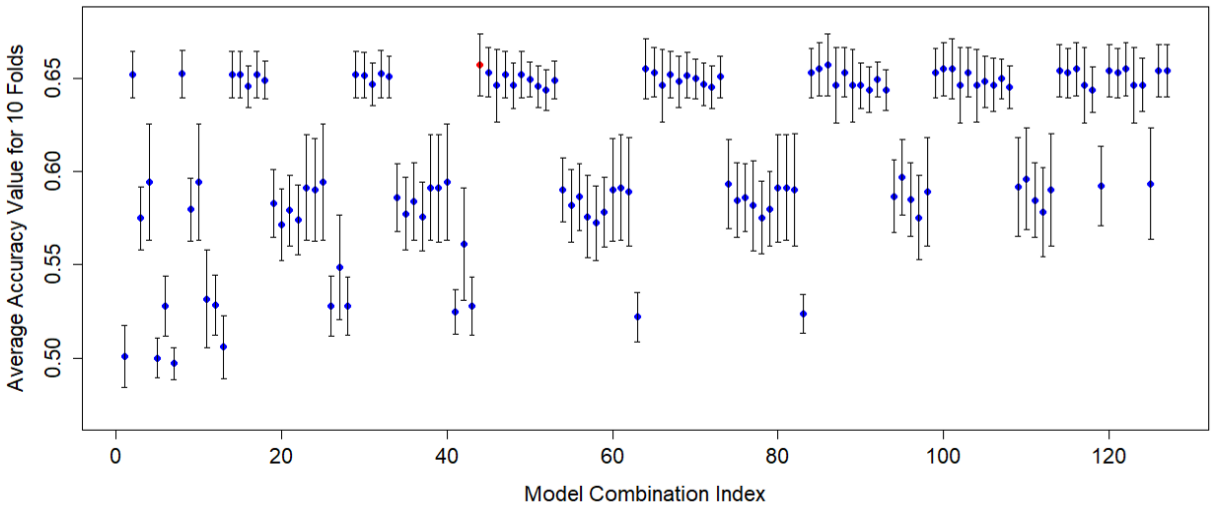


Figure 9: Scatter plot showing predictive accuracy of all 127 input model combinations with 10-fold cross-validation applied. The blue dots are the average predictive performance values, the whiskers the standard deviation throughout the 10 folds. The red dot is the highest performing model goal ~ video + GitHub + CEOPhoto with an accuracy of 65.7%.

3.1.2 Visualisation Of Highest Performing Decision Tree Model

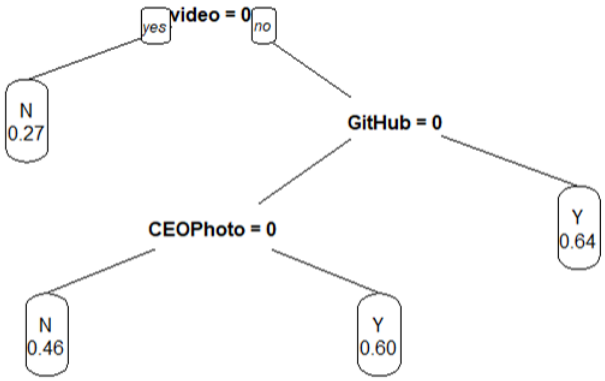


Figure 10: Visualisation of highest performing decision classification tree model that predicts whether a startup will reach its fundraising goal successfully through the ICO at a 65.7% success rate.

The model seen in **figure (10)** can be interpreted as follows. Given a new, ICO project with data on 3 feature input variables seen in the tree, the decision tree model may classify such a project, determining whether it is successful or may fail with approximately a 65.7% success rate. The importance of each feature variable is determined by the hierarchical structure of the decision tree.
For instance, being the root node, **video** dominates the model determining whether a project is deemed successful with a 73% partition score - this is shown by the numeric of each leaf in the plot. These are the class probabilities at each end point i.e., 73% of projects in the training set that have a video of their project are deemed successful.

3.2 Random Forest

Since decision classification trees are well known to be inherently unstable, applying a random forest strategy to the equivalent data used in **section (3.1)** may yield greater predictive performances. This concept is a form of ensemble learning and involves bootstrap aggregating (BAGGING) many individually poor decision tree models to produce one accurate predictive model. [4] Key advantages of random forest include: their non-parametric nature; high classification accuracy; and capability to determine variable importance. The **randomForest** algorithm is used with 500 bootstrapped decision classification trees producing 1 forest. 5-fold cross-validation is used thus the total number of models produced is $5 \times 127 \times 500 = 317500$.

3.2.1 Prediction Accuracy For Each Model Selection With 5-Fold CV

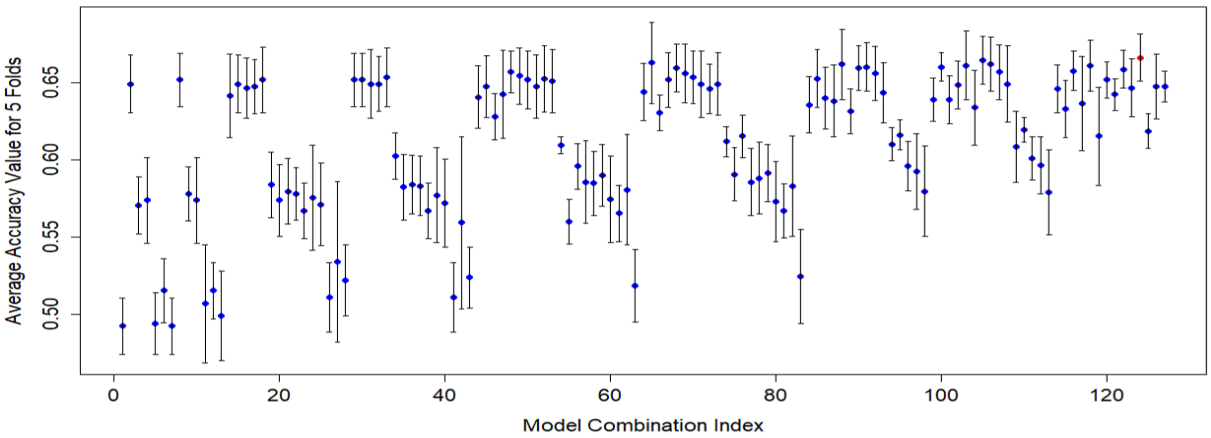


Figure 11: Scatter plot showing predictive accuracy of all 127 input model combinations with 5-fold cross-validation applied. See **figure (9)** for plot explanation. Highest performing model is goal ~ video + CEOPhoto + Banking + USA + Unknown + whitepaper with an accuracy of 66.6%.

3.2.2 Variable Importance Of Highest Performing Random Forest Model

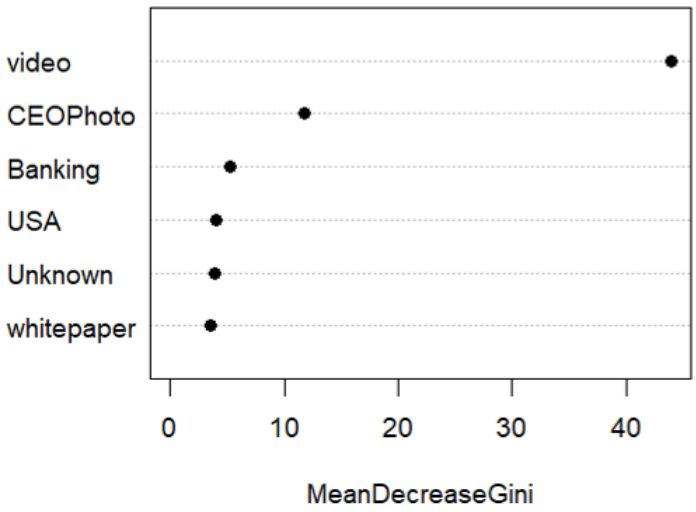


Figure 12: Dot chart showing the input variable hierarchy of the highest performing model goal ~ video + CEOPhoto + Banking + USA + Unknown + whitepaper.

Variable importance can be seen in descending order in **figure (12)**. Analogous to decision classification trees, **video** is the dominant variable with a mean decrease in gini impurity index of 43 throughout the 500 bootstrap models. Interestingly, **GitHub** is determined to be insignificant by random forest, contrasting the decision classification model output observed in **section (3.1)**. Applying random forest has increased predictive accuracy from 65.7% to 66.6%.

3.3 K-Nearest Neighbour

K-nearest neighbour (KNN) is a supervised machine learning technique used to classify the ICO projects. The algorithm learns by considering distances between input features and their relation to output, where k denotes the number of nearest observations to consider in the feature space. [3] Justification of model selection arises from its simplicity and easily implementation. Building the model is cheap. It is extremely flexible classification scheme and well suited for Multi-modal classes. **knn** is the algorithm used with a Euclidean distance metric chosen. Input features used are **teamSize**, **overallrating**, **ratingTeam** and **ratingProduct**, cleaned and normalised as discussed in **section (2)**. Reasoning for this model selection stems from clear evidence of clusters formed between successful and unsuccessful projects seen in **figure (1)**. The value $k \in \{1,..,100\}$ is chosen such that predictive performance is maximized,. Due to knn's efficient computation, 50-fold cross-validation is used resulting in a total model output of $4 \times 100 \times 50 = 20000$ models.

3.3.1 Scatter Plot Of Prediction Accuracy For $k \in \{1,..,100\}$

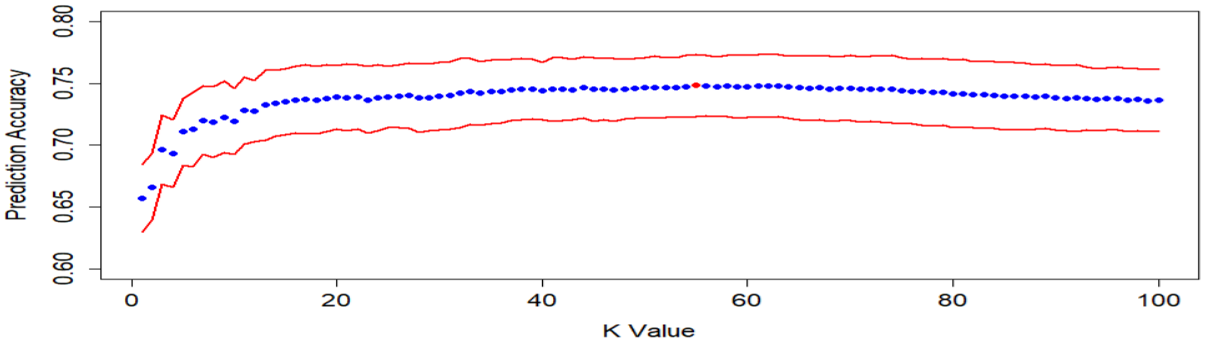


Figure 13: Scatter plot showing the prediction accuracy of knn for all values $k \in \{1,..,100\}$ shown by the blue points, for model discussed above. The standard deviation throughout the 50-folds can be seen by the red lines. The highest performing value is k=55 with a 74.8% predictive performance rate.

3.4 Support Vector Machine

Support Vector Machine (SVM) is a tool to classify ICO projects that constructs hyperplanes in our feature space subject to maximizing homogeneity on either side of the hyperplane. [5] SVM has excellent performance in generalization so it can produce high accuracy when there is a clear margin of separation between classes, resulting in highly effective high dimensional task performance. **ksvm** algorithm is used with **vanilladot** kernel criterion. If multiple hyperplanes exist that provide complete classification then the hyperplane with greatest margin is chosen, where the margin of a hyperplane is the orthogonal distance from the plane to the nearest feature input. Model selection strategy is used whereby our feature space under investigation span all numeric variables that is **coinNum**, **teamSize**, **overallrating**, **ratingTeam**, **ratingProduct**, **acceptingCurrencyNum**, **socialMedia**, **teamLinkedIn**, **teamPhotos** and **date_diff**. Thus the total number of model combinations considered are

$$\binom{10}{1} + \ldots + \binom{10}{10} = 1023 \text{ input variable combinations.}$$

3.4.1 Heat-Map Scatter Plot Of Predictive Performance For Each Model Combination

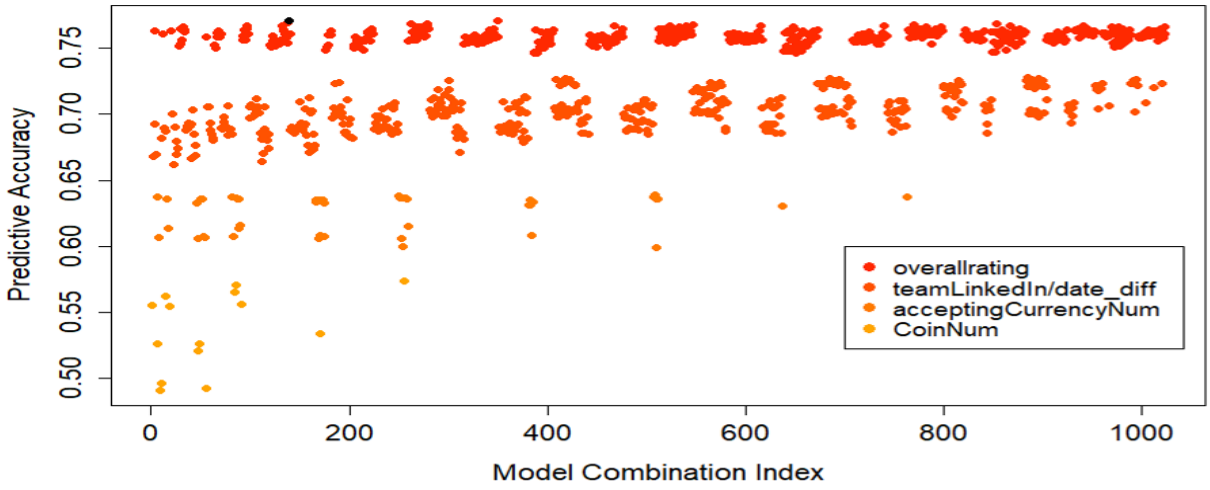


Figure 14: Scatter plot of predictive performance of 1023 model combinations for SVM with 5-fold cross-validation. Feature variable inclusions form clusters labeled and colour-coded by the legend. The highest performing model is **goal ~ overallrating + teamLinkedIn + date_diff** with an accuracy of 77.1% - see black point on plot.

3.5 Modeling Summary Tabular

Model	Feature Type	Highest Performing Feature Combination	Predictive Performance %
Decision Classification Tree	Categorical	video + GitHub + CEOPhoto	65.7
Random Forest	Categorical	video + CEOPhoto + Banking + USA + Unknown + whitepaper	66.6
KNN	Numerical	teamSize + overallrating + ratingTeam + rating Product, k=55	74.8
SVM	Numerical	overallrating + teamLinkedIn + date_diff	77.1

4 Evaluation

4.0.1 Confusion Matrices

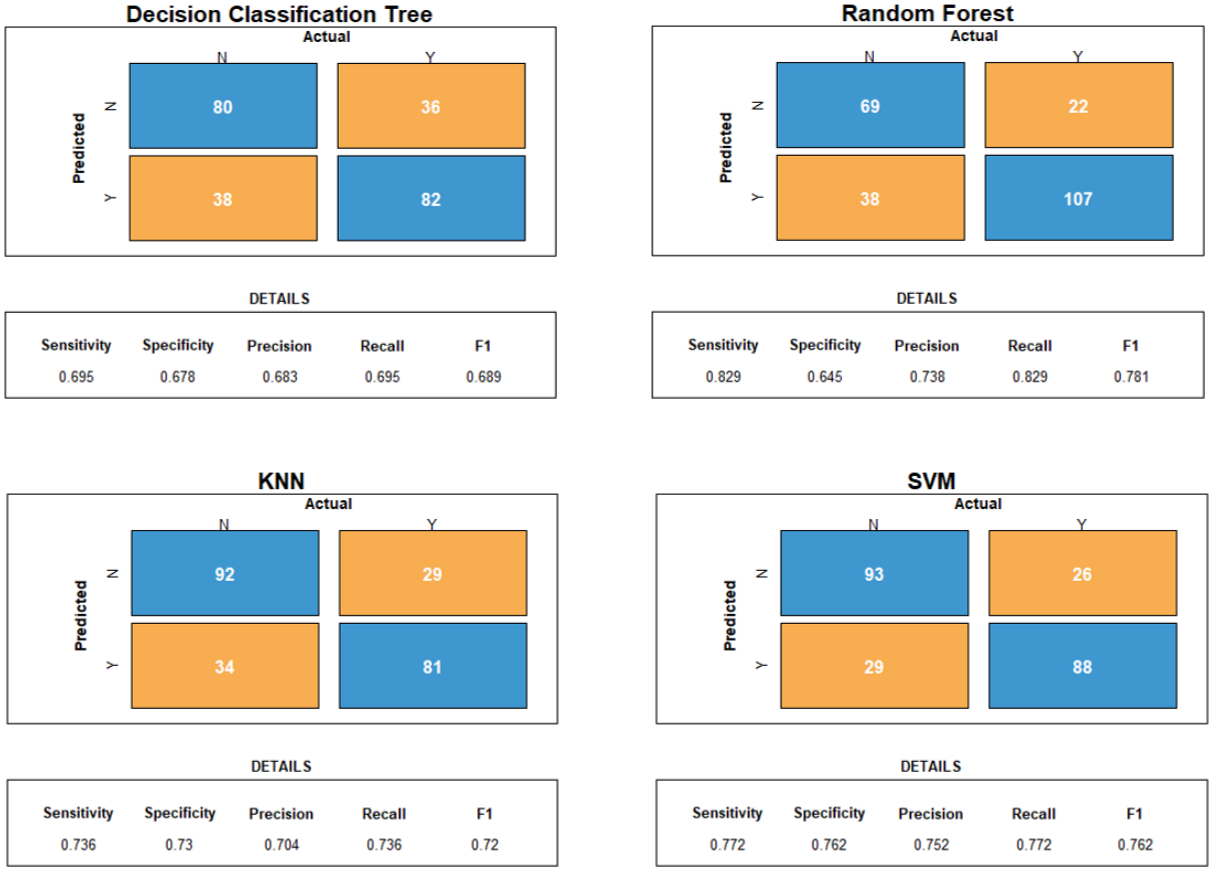


Figure 15: Confusion matrices for models seen in **section (3.5)**. The number of true positive/negative outcomes can be seen by values in the blue boxes, false positive/negative outcomes can be seen by values in the orange boxes.

Evaluation metrics are seen in the **Details** section below each confusion matrix. Description of such metrics are as follows. All metrics are proportion based. Sensitivity measures real positives correctly predicted, specificity real negatives. Precision measures the proportion of positive predictions that are truly positive and recall the number of true positives over the total positive count - F1 is a combined measure to assess model in terms of both precision and recall together. Desired models span evaluation metrics with high values. To quantify the significance of certain evaluation metrics in relation to each other, we must consider and weight the detrimental effects of false negatives that is, predicting an ICO project will reach it's fundraising goal when in actuality the project fails and is deemed unsuccessful. [1] If the money raised in an ICO is less than the minimum amount required by the ICO's criteria, then all of the money is returned to the project's investors. Thus we deem specificity metrics more important than sensitivity metrics - a model with a greater proportion of failed ICOs that were predicted correctly has the most use to **predict whether a startup will reach its fundraising goal successfully through the ICO**. Not only does SVM have the greatest predictive performance seen in **section (3.5)**, SVM produces predictions with largest specificity value too - 0.762, in comparison to values of 0.730, 0.645 and 0.683 produced by KNN, random forest and decision classification tree models respectively.

4.0.2 ROC Curves & AUC Values

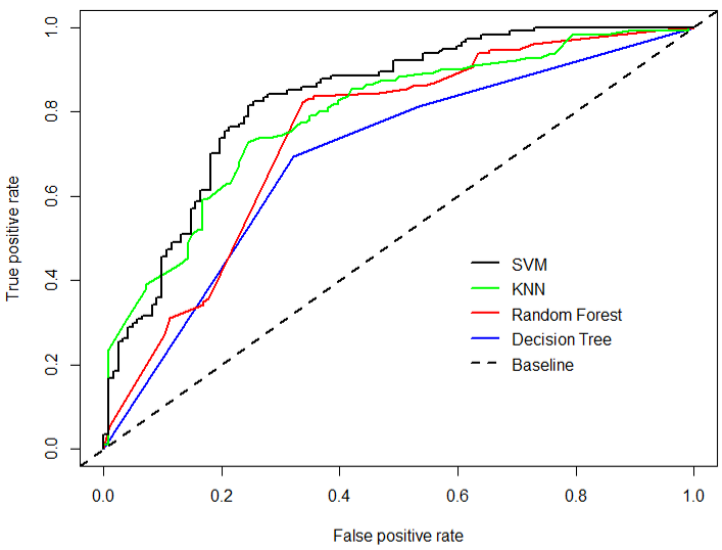


Figure 16: **R**eceiver-**O**perating **C**haracteristic (**ROC**) curve showing trade-offs between sensitivity (true positive rate) and specificity (false positive rate) for all models with a baseline random classifier shown by the black dotted line - see legend for labels.

The ROC curves seen in **figure (16)** are formed subject to variations in the threshold rate for the respective model that is, the raw prediction probability that partitions our class label **goal**. The curves show sensitivity and specificity metrics at given thresholds, with optimum models containing threshold values inclusive of low false positive rates and high true positive rates. Empirically, being the model with the greatest prevalence of these attributes, SVM is again our highest performing model **to predict whether a startup will reach its fundraising goal successfully through the ICO**. To determine the remaining model hierarchy, a useful metric to compute are the **A**rea **U**nder **C**urve (**AUC**) values, with the greater the AUC value determining a greater performing model.

Model		**AUC**	
DT	RF	KNN	SVM
0.694	0.744	0.789	0.830

4.0.3 Model Stability

Cross-validation implemented throughout modeling in **section (3)** may be cross-examined to determine instabilities within variations of model training and testing data. The table below shows stability metrics across the four models for the highest performing input features determined prior.

Models with smallest standard deviations between folds, **M**ean **S**quared **E**rror (**MSE**) against the average prediction value, **R**oot **M**ean **S**quared **E**rror (**RMSE**) and **M**ean **A**bsolute **E**rror (**MAE**) are most stable - their outputs are consistent with variations in training and testing datasets. Mean squared error emphasizes large margins of error present in models relative to mean absolute error, this considers absolute distance as its error metric.

From the table we find SVM to be most stable in output, with the lowest deviation value of 1.16% in predictive performance throughout the cross-validation folds. This is more than half the value for its numerical counterpart - KNN with a deviation of 2.5%. Furthermore, SVM performs greatest when we consider error values too with an MSE, RMSE and MAE of 0.1%, 1.04% and 0.85% respectively, compare this to KNN with an MSE of 0.6%, six times greater than the SVM error.

Model/ Stability Metric	**Decision Classification Tree**	**Random Forest**	**KNN**	**SVM**
Standard Deviation	0.0165	0.0151	0.0250	0.0116
Mean Squared Error	0.0002	0.0002	0.0006	0.0001
Root Mean Squared Error	0.0157	0.0135	0.0247	0.0104
Mean Absolute Error	0.0132	0.0129	0.0206	0.0085

5 Deployment

Given a dataset containing 19 variables about 1181 ICO projects from different startups, this report has shown applications of machine learning models predicting whether a startup will reach its fundraising goal successfully through the ICO. Model justifications in relation to the context of the data were given, with descriptions of optimal predictors and their respective output present. Our findings our as follows. Not only is support vector machine highly effective relative to other models at predicting project outcomes, but this model outputs the greatest proportion of failed ICOs that were predicted correctly in addition to producing the most stable outputs throughout variations in training and testing data seen via cross-validation. Support vector machine produced the highest project success prediction rate of 77.1%, specificity score of 0.762 and AUC value of 0.830. Undergoing further research into determining whether a startup will reach its fundraising goal successfully through the ICO, possible avenues to consider are increasing the quantity of the equivalent data used, applying new input features such as Bitcoin prices and considering performances of alternative machine learning model such as naive bayes, neural networks and cluster algorithms.

References

[1] Christian Fisch. Initial coin offerings (icos) to finance new ventures. *Journal of Business Venturing*, 2019.

[2] M. A. Friedl. Decision tree classification of land coverfrom remotely sensed data. *Remote Sensing of Environment*, 1997.

[3] Susmita Ray. A quick review of machine learning algorithms. 2019.

[4] V.F. Rodriguez-Galiano. An assessment of the effectiveness of a random forest classifierfor land-cover classification. *ISPRS Journal of Photogrammetry and Remote Sensing*, 2012.

[5] Achmad Widodo. Support vector machine in machine condition monitoring andfault diagnosis. *Mechanical Systems and Signal Processing*, 2007.

Appendix

```
rm(list=ls())
setwd("/Users/marcu/OneDrive/Desktop/Uni/LUBS5990M/Coursework")
ICO <- read.csv("data_LUBS5990M_202122.csv")
# ------------------------------------------
# ---- Data Understanding & Preparation ----
# ------------------------------------------
# ---- Numeric Data ----

# normalisation function
normalisation_funct <- function(x) {
  (x - min(x)) / (max(x) - min(x))
}

# Finds difference in days
ICO$date_diff <- as.numeric(as.Date(as.character(ICO$enddate),
    format="%d/%m/%Y")-as.Date(as.character(ICO$startdate), format="%d/%m/%Y"))

# imputes negative days with median value
ICO$date_diff[c(264,398,790)] <- median(ICO$date_diff)

# numeric data
ICO_numeric <- data.frame(ICO[,c("coinNum","teamSize","overallrating",
                        "ratingTeam","ratingProduct","acceptingCurrencyNum",
                        "socialMedia","teamLinkedIn","teamPhotos","date_diff")])

# removes percentage sign
ICO_numeric$teamLinkedIn<-as.numeric(gsub("%$","",ICO_numeric$teamLinkedIn))
ICO_numeric$teamPhotos<-as.numeric(gsub("%$","",ICO_numeric$teamPhotos))

# changes "unknown" entry to 0
ICO_numeric["acceptingCurrencyNum"][ICO_numeric["acceptingCurrencyNum"] == "unknown"] <- 0
ICO_numeric$acceptingCurrencyNum <- as.numeric(ICO_numeric$acceptingCurrencyNum)

# replace outliers with median value
outliers <- boxplot(ICO_numeric$coinNum, plot=FALSE)$out
outliers_indx <- which(ICO_numeric$coinNum %in% outliers)

for (i in 1:length(outliers_indx)){
  indx <- outliers_indx[i]
  ICO_numeric$coinNum[indx] <- median(ICO_numeric$coinNum)
}

# normalisation of numeric values
ICO_normalised <- data.frame(apply(ICO_numeric[,],2,normalisation_funct))

# append goal
```

```
ICO_normalised$goal <- as.factor(ICO$goal)

# scatter plot matrix
par(mar=c(0,0,0,0))
colours <- c("red2","springgreen4")
pairs(ICO_normalised[,c(-6,-7,-9,-10,-11)],col=colours[ICO_normalised$goal],
      pch=16,cex=0.1,lower.panel=NULL)

# correlation coefficient matrix
library(corrplot)
c <- cor(ICO_normalised[,c(-6,-7,-9,-10,-11)])
corrplot(c,type="upper",diag=F,cex.main=0.8)

# box plots
par(mar=c(2,2,0,0))
boxplot(ICO_normalised[c(-9,-11)],data=ICO_normalised, col="lightblue",
        lwd=2,pch=16,cex.main=2.2)

# Histogram matrix
library("ggplot2")
library("tidyr")
#     makes data frame long (2 columns)
data_long <- ICO_normalised[,c(-9,-11)] %>%
  pivot_longer(colnames(ICO_normalised[,c(-9,-11)])) %>%
  as.data.frame()
#     Draw histogram & density
ggp3 <- ggplot(data_long, aes(x = value)) +
  geom_histogram(aes(y = ..density..),bins=25) +
  geom_density(col = "#1b98e0", size = 2) +
  facet_wrap(~ name, scales = "free")
ggp3

# ---- Categorical Data ----
# categorical variables
ICO_categorical <- data.frame(ICO[,c("country_region","categories","platform",
                    "whitepaper", "video","GitHub","CEOPhoto","goal")])

# Binary variables
ICO_categorical_binary <- ICO_categorical[,c(4,5,6,7,8)]
ICO_categorical_binary$goal <- as.factor(ICO_categorical_binary$goal)

# Bar Plot
Bars <- numeric(16)

Bars_names <- c("w,1,Y","w,1,N","w,0,Y","w,0,N",
                "v,1,Y","v,1,N","v,0,Y","v,0,N",
                "g,1,Y","g,1,N","g,0,Y","g,0,N",
                "c,1,Y","c,1,N","c,0,Y","c,0,N")

for (i in 1:dim(ICO_categorical_binary)[1]){

    if (ICO_categorical_binary$whitepaper[i]==1 & ICO_categorical_binary$goal[i]=="Y"){
        Bars[1]= Bars[1]+1
```

```
      }
      else if (ICO_categorical_binary$whitepaper[i]==1 & ICO_categorical_binary$goal[i]=="N"){
        Bars[2]= Bars[2]+1
      }
      else if (ICO_categorical_binary$whitepaper[i]==0 & ICO_categorical_binary$goal[i]=="Y"){
        Bars[3]= Bars[3]+1
      }
      else if (ICO_categorical_binary$whitepaper[i]==0 & ICO_categorical_binary$goal[i]=="N"){
        Bars[4]= Bars[4]+1
      }
}

for (i in 1:dim(ICO_categorical_binary)[1]){

      if (ICO_categorical_binary$video[i]==1 & ICO_categorical_binary$goal[i]=="Y"){
        Bars[5]= Bars[5]+1
      }
      else if (ICO_categorical_binary$video[i]==1 & ICO_categorical_binary$goal[i]=="N"){
        Bars[6]= Bars[6]+1
      }
      else if (ICO_categorical_binary$video[i]==0 & ICO_categorical_binary$goal[i]=="Y"){
        Bars[7]= Bars[7]+1
      }
      else if (ICO_categorical_binary$video[i]==0 & ICO_categorical_binary$goal[i]=="N"){
        Bars[8]= Bars[8]+1
      }
}


for (i in 1:dim(ICO_categorical_binary)[1]){

      if (ICO_categorical_binary$GitHub[i]==1 & ICO_categorical_binary$goal[i]=="Y"){
        Bars[9]= Bars[9]+1
      }
      else if (ICO_categorical_binary$GitHub[i]==1 & ICO_categorical_binary$goal[i]=="N"){
        Bars[10]= Bars[10]+1
      }
      else if (ICO_categorical_binary$GitHub[i]==0 & ICO_categorical_binary$goal[i]=="Y"){
        Bars[11]= Bars[11]+1
      }
      else if (ICO_categorical_binary$GitHub[i]==0 & ICO_categorical_binary$goal[i]=="N"){
        Bars[12]= Bars[12]+1
      }
}

for (i in 1:dim(ICO_categorical_binary)[1]){

      if (ICO_categorical_binary$CEOPhoto[i]==1 & ICO_categorical_binary$goal[i]=="Y"){
        Bars[13]= Bars[13]+1
      }
      else if (ICO_categorical_binary$CEOPhoto[i]==1 & ICO_categorical_binary$goal[i]=="N"){
        Bars[14]= Bars[14]+1
      }
```

```
      else if (ICO_categorical_binary$CEOPhoto[i]==0 & ICO_categorical_binary$goal[i]=="Y"){
        Bars[15]= Bars[15]+1
      }
      else if (ICO_categorical_binary$CEOPhoto[i]==0 & ICO_categorical_binary$goal[i]=="N"){
        Bars[16]= Bars[16]+1
      }
}

par(lwd = 4,cex=1.5)
barplot(Bars, axisnames=T,names.arg=Bars_names, xlab="Binary Variables"
        ,ylab="Number of Projects that Include Variable",
        col=c("springgreen4","red2","springgreen4","red2",
              "springgreen4","red2","springgreen4","red2",
              "springgreen4","red2","springgreen4","red2",
              "springgreen4","red2","springgreen4","red2"),
        border=c("black","black","navy","navy",
                 "black","black","navy","navy",
                 "black","black","navy","navy",
                 "black","black","navy","navy"))

# Countries
# Cleaning
ICO_categorical["country_region"][ICO_categorical["country_region"] == "United Kingdom"] <- "UK"
ICO_categorical["country_region"][ICO_categorical["country_region"] == "England"] <- "UK"
ICO_categorical["country_region"][ICO_categorical["country_region"] == "PerÃº"] <- "Peru"
ICO_categorical["country_region"][ICO_categorical["country_region"] == "United States of America"] <- "U
ICO_categorical["country_region"][ICO_categorical["country_region"] == "United States"] <- "USA"
ICO_categorical["country_region"][ICO_categorical["country_region"] == "Russian"] <- "Russia"
ICO_categorical["country_region"][ICO_categorical["country_region"] == "india"] <- "India"
ICO_categorical["country_region"][ICO_categorical["country_region"] == "Ras al-Khaimah"] <- "UAE"
ICO_categorical["country_region"][ICO_categorical["country_region"] == "Dubai"] <- "UAE"
ICO_categorical["country_region"][ICO_categorical["country_region"] == "United Arab Emirates"] <- "UAE"
ICO_categorical["country_region"][ICO_categorical["country_region"] == "Cayman Island"] <- "Cayman Isla
ICO_categorical["country_region"][ICO_categorical["country_region"] == "Virgin Islands"] <- "British Vi
ICO_categorical["country_region"][ICO_categorical["country_region"] == "Netherland"] <- "Netherlands"
ICO_categorical["country_region"][ICO_categorical["country_region"] == "Czechia"] <- "Czech Republic"
ICO_categorical["country_region"][ICO_categorical["country_region"] == "Ireland"] <- "UK"

# Grouping
library("tidyverse")
country_freq <- data.frame(sapply(ICO_categorical[1], function(x) {
          sapply(unique(ICO_categorical$country_region), function(y) {
            sum(grepl(y, x))
        })
        }))
group_countries <- data.frame(filter(country_freq,country_region<10))
group_countries <- cbind(Country = rownames(group_countries), group_countries)
rownames(group_countries) <- 1:nrow(group_countries)

for (i in 1:dim(ICO_categorical)[1]){
  if(ICO_categorical$country_region[i] %in% group_countries$Country == TRUE){
    ICO_categorical$country_region[i] <- "Other"
  }
```

```
}
# Success ratio
country_freq <- data.frame(sapply(ICO_categorical[1], function(x) {
  sapply(unique(ICO_categorical$country_region), function(y) {
    sum(grepl(y, x))
  })
}))
country_freq <- cbind(Country = rownames(country_freq), country_freq)
rownames(country_freq) <- 1:nrow(country_freq)
colnames(country_freq)[2] <- "frequency"

no.of.Y <- numeric(dim(country_freq)[1])

for (i in 1:dim(country_freq)[1]){
  country <- country_freq$Country[i]
  rowsofcountry <- data.frame(filter(ICO_categorical,country_region==country))
  no.of.Y[i] <- sum(rowsofcountry$goal=="Y")

}
country_freq$no.of.Y <- no.of.Y
country_freq$Yratio <- country_freq$no.of.Y/country_freq$frequency


#Bar plot of Country Success ratio
par(mar=c(4,9,0,0))
bp <- barplot(country_freq$Yratio, axisnames=T,names.arg=country_freq$Country,
              xlab="Proportion of Successful Projects"
        , horiz=T,las=1, xlim=c(0,1.1),
        col=c("springgreen4","springgreen4","red2","springgreen4",
              "springgreen4","springgreen4","grey","grey",
              "grey","grey","grey","grey",
              "red2","red2","springgreen4","grey",
              "grey","red2","red2","red2",
              "grey","grey","springgreen4","grey",
              "grey","grey","red2","springgreen4",
              "springgreen4","grey","springgreen4","springgreen4"))
text(0,bp, labels=country_freq$frequency,cex=1,pos=4)

# Platform
# Cleaning
ICO_categorical["platform"][ICO_categorical["platform"] == "WAVES"] <- "Waves"
ICO_categorical["platform"][ICO_categorical["platform"] == "ETH"] <- "Ethereum"
ICO_categorical["platform"][ICO_categorical["platform"] == "Stellar Consensus Protocol (SCP)"] <- "Stel
ICO_categorical["platform"][ICO_categorical["platform"] == "Ehtereum"] <- "Ethereum"

# Grouping
library("tidyverse")
platform_freq <- data.frame(sapply(ICO_categorical[3], function(x) {
  sapply(unique(ICO_categorical$platform), function(y) {
    sum(grepl(y, x))
  })
}))
group_platform <- data.frame(filter(platform_freq,platform<10))
```

```r
group_platform <- cbind(Plat = rownames(group_platform), group_platform)
rownames(group_platform) <- 1:nrow(group_platform)

for (i in 1:dim(ICO_categorical)[1]){
  if(ICO_categorical$platform[i] %in% group_platform$Plat == TRUE){
    ICO_categorical$platform[i] <- "Other"
  }
}

#Success ratio
platform_freq <- data.frame(sapply(ICO_categorical[3], function(x) {
  sapply(unique(ICO_categorical$platform), function(y) {
    sum(grepl(y, x))
  })
}))

platform_freq <- cbind(Plat = rownames(platform_freq), platform_freq)
rownames(platform_freq) <- 1:nrow(platform_freq)
colnames(platform_freq)[2] <- "frequency"

no.of.Y <- numeric(dim(platform_freq)[1])

for (i in 1:dim(platform_freq)[1]){
  Plat <- platform_freq$Plat[i]
  rowsofplatform <- data.frame(filter(ICO_categorical,platform==Plat))
  no.of.Y[i] <- sum(rowsofplatform$goal=="Y")

}
platform_freq$no.of.Y <- no.of.Y
platform_freq$Yratio <- platform_freq$no.of.Y/platform_freq$frequency

#Bar plot of Platform Success ratio
par(mar=c(4,9,0,0))
bp <- barplot(platform_freq$Yratio, axisnames=T,names.arg=platform_freq$Plat,
              xlab="Proportion of Successful Projects"
              , horiz=T,las=1, xlim=c(0,1.1),
              col=c("grey","red2","springgreen4","red2"))
text(0,bp, labels=platform_freq$frequency,cex=1,pos=4)

# categories
# unique categories
cat_unique <- unique(unlist(strsplit(ICO_categorical$categories, ",")))

# frequency
cat_freq <- data.frame(sapply(ICO_categorical[2], function(x) {
  sapply(cat_unique, function(y) {
    sum(grepl(y, x))
  })
}))
cat_freq <- cbind(Category = rownames(cat_freq), cat_freq)
rownames(cat_freq) <- 1:nrow(cat_freq)
colnames(cat_freq)[2] <- "frequency"
```

```r
# Success ratio

no.of.Y <- numeric(dim(cat_freq)[1])

library("tidyverse")
for (i in 1:dim(cat_freq)[1]){
  cat <- cat_freq$Category[i]
  rowsofcat <- data.frame(filter(ICO_categorical,
                                 grepl(cat,ICO_categorical$categories)))
  no.of.Y[i] <- sum(rowsofcat$goal=="Y")
}
cat_freq$no.of.Y <- no.of.Y
cat_freq$Yratio <- cat_freq$no.of.Y/cat_freq$frequency

#Bar plot of Country Success ratio
par(mar=c(4,9,0,0))
bp <- barplot(cat_freq$Yratio, axisnames=T,names.arg=cat_freq$Category,
              xlab="Proportion of Successful Projects"
              , horiz=T,las=1, xlim=c(0,1.1),
              col=c("red2","red2","red2","grey",
                    "grey","grey","grey","grey",
                    "grey","springgreen4","springgreen4","grey",
                    "grey","grey","grey","grey",
                    "grey","grey","grey","grey",
                    "grey","grey","grey","grey",
                    "grey","springgreen4","red2","grey",
                    "grey"))
text(0,bp, labels=cat_freq$frequency,cex=1,pos=4)




# ----------------------------------------
# ----------------------------------------
# -------------- Modelling ---------------
# ----------------------------------------
# clean environment
dev.off()
rm(bp,c,data_long,ggp3,country_freq,cat_freq,group_countries,group_platform,
   platform_freq,rowsofcat,rowsofcountry,rowsofplatform,Bars,Bars_names,cat,
   cat_unique,colours,country,i,indx,no.of.Y,outliers,outliers_indx,Plat,
   normalisation_funct,ICO_categorical_binary,ICO_numeric)
# ----Decision tree----
# The goal here is to determine from categorical data, which model selection
# results in the greatest prediction accuracy - and evaluate after.

# convert output to factor type
ICO_categorical$goal <- as.factor(ICO_categorical$goal)

library(rpart)
library(rpart.plot)
```

```r
# ICO in USA variable
ICO_categorical$USA <- ifelse(ICO_categorical$country_region=="USA",1,0)

# ICO banking category variable
ICO_categorical$Banking <- ifelse(grepl("Banking", ICO_categorical$categories,
                                        fixed = TRUE),1,0)

# ICO unknown platform variable
ICO_categorical$Unknown <- ifelse(grepl("unknown", ICO_categorical$platform,
                                        fixed = TRUE),1,0)

# Removes not needed variables
ICO_categorical <- ICO_categorical[,-c(1,2,3)]

# Reorder dataframe
ICO_categorical <- ICO_categorical[,c(5,1,2,3,4,6,7,8)]

library(dagR)
#--Produces vector of input combinations--
#Matrix of all combinations of 7 input elements using allCombs function
x <- seq(1,7)
mat <- (allCombs(x))
mat <- replace(mat,mat==1,"whitepaper")
mat <- replace(mat,mat==2,"video")
mat <- replace(mat,mat==3,"GitHub")
mat <- replace(mat,mat==4,"CEOPhoto")
mat <- replace(mat,mat==5,"USA")
mat <- replace(mat,mat==6,"Banking")
mat <- replace(mat,mat==7,"Unknown")
mat <- replace(mat,is.na(mat),"")

formulas <- rep(NA,length.out=128)
#formats matrix, adding plus signs and removing plus signs where necessary
for (i in 1:length(formulas)){
  string <- paste(mat[i,], sep = "", collapse = "+")
  formulas[i] <- (gsub("\\+\\+.*","",string))
  if (endsWith(formulas[i],"+")==TRUE){
    formulas[i]<- substr(formulas[i],1,nchar(formulas[i])-1)
  }
  formulas[i] <- paste("goal~",formulas[i],sep = "", collapse = "")
}
#removes quotation marks and zero entry
formulas <- noquote(formulas)
formulas <- formulas[-1]

#--Cross Validation Strategy Used--
set.seed(121)

#folds
k = 10

#matrix of all models with value - accuracy to be filled
Single_accuracy<-matrix(nrow=length(formulas), ncol=k)
```

```
#consider fold 1 to k
for (j in 1:k){

  #creates testing and training sets randomly, test=20% of dataset
  idx <- sample(1:1181,236)
  train_data <- ICO_categorical[-idx, ]
  test_data <- ICO_categorical[idx, ]

  #applies decision tree model for current formula , predicts using test
  #data and outputs accuracy
  for (i in 1:length(formulas)){
    current_model <- rpart(as.formula(formulas[i]), data=train_data,
                           method='class',control=rpart.control(0.01))
    #predicts using test data, outputting raw probabilities
    raw_probabilities= predict(current_model, newdata=test_data)
    raw_probabilities = raw_probabilities[, 2]
    #converts raw prediction probabilities to most likely output
    predicted_output <- rep("N", dim(test_data)[1])
    predicted_output[raw_probabilities > .5] = "Y"
    #calculates prediction accuracy
    table <- table(predicted_output, test_data$goal)
    Single_accuracy[i,j] <- sum(diag(table))/sum(table)
  }

}
Average_accuracy <- rowSums(Single_accuracy)/k
Average_accuracy

#--Stability Plot---
#creates whiskers for points on plot showing deviation
Single_accuracy_df <- data.frame(Single_accuracy)
accuracy_sd <- rep(NA,dim(Single_accuracy_df)[1])
for (i in 1:dim(Single_accuracy_df)[1]){
  accuracy_sd[i] <- sd(Single_accuracy_df[i,])
}
Indx <- seq(1,length(formulas))
plot(Indx,Average_accuracy,pch=19,col=ifelse(Indx==which.max(Average_accuracy),
    "red", "blue"),ylim=c(0.47,0.68),xlab="Model Combination Index",
    ylab="Average Accuracy Value for 10 Folds",cex.lab=1.4,cex.axis=1.4)
arrows(x0=Indx, y0=Average_accuracy-accuracy_sd, x1=Indx,
       y1=Average_accuracy+accuracy_sd,code=3, angle=90, length=0.025)

#--Plot Winning Tree--
mytree <- rpart(formulas[which.max(Average_accuracy)], data=ICO_categorical,
                method='class',control=rpart.control(0.01))
prp(mytree, extra=6)

# ----Random Forest----
# Bagging of DT models
library(randomForest)

#--Cross Validation Strategy Used--
set.seed(121)
```

```
#folds
k = 5

#matrix of all models with value - accuracy to be filled
Single_accuracy<-matrix(nrow=length(formulas), ncol=k)

#consider fold 1 to k
for (j in 1:k){

  #creates testing and training sets randomly, test=20% of dataset
  idx <- sample(1:1181,236)
  train_data <- ICO_categorical[-idx, ]
  test_data <- ICO_categorical[idx, ]

  #applies rand forest model for current formula , predicts using test data and
  #outputs accuracy
  for (i in 1:length(formulas)){
    current_model <- randomForest(as.formula(formulas[i]), data=train_data,
                                  type="classification")
    #predicts using test data, outputting raw probabilities
    raw_probabilities= predict(current_model, newdata=test_data, type="prob")
    raw_probabilities = raw_probabilities[,2]
    #remove 100% and 0% outcomes !TRICK!
    number_of_trees = current_model$ntree
    raw_probabilities = (number_of_trees*raw_probabilities + 1)/(number_of_trees + 2)
    #converts raw prediction probabilities to most likely output
    predicted_output <- rep("N", dim(test_data)[1])
    predicted_output[raw_probabilities > .5] = "Y"
    #calculates prediction accuracy
    table <- table(predicted_output, test_data$goal)
    Single_accuracy[i,j] <- sum(diag(table))/sum(table)
  }

}
Average_accuracy <- rowSums(Single_accuracy)/k
Average_accuracy

#--Stability Plot---
#creates whiskers for points on plot showing deviation
Single_accuracy_df <- data.frame(Single_accuracy)
accuracy_sd <- rep(NA,dim(Single_accuracy_df)[1])
for (i in 1:dim(Single_accuracy_df)[1]){
  accuracy_sd[i] <- sd(Single_accuracy_df[i,])
}
Indx <- seq(1,length(formulas))
plot(Indx,Average_accuracy,pch=19,col=ifelse(Indx==which.max(Average_accuracy),
                                             "red", "blue"),ylim=c(0.47,0.69),
     xlab="Model Combination Index", ylab="Average Accuracy Value for 5 Folds",
     cex.lab=1.4,cex.axis=1.4)
arrows(x0=Indx, y0=Average_accuracy-accuracy_sd, x1=Indx,
       y1=Average_accuracy+accuracy_sd, code=3, angle=90, length=0.025)

myforest <- randomForest(as.formula(formulas[which.max(Average_accuracy)]),
```

```
                          data=ICO_categorical)
varImpPlot(myforest,pch=19,cex=1.2, main=NULL)

# ----KNN----
library(class)
library(gmodels)

# data used
knn_data<-ICO_normalised[,-c(1,6,7,8,9,10)]

# Possible k values
kvalues <- seq(1,100)

#--Cross Validation Strategy Used--
set.seed(121)

#folds
k = 50

#matrix of all models with value - accuracy to be filled
Single_accuracy<-matrix(nrow=length(kvalues), ncol=k)

#consider fold 1 to k
for (j in 1:k){

  #creates testing and training sets randomly, test=20% of dataset
  idx <- sample(1:1181,236)
  train_data <- knn_data[-idx, ]
  test_data <- knn_data[idx, ]

  #applies knn model for current k value , predicts using test data and
  #outputs accuracy
  for (i in 1:length(kvalues)){
    current_model <- knn(train = train_data[,-5], test = test_data[,-5],
                         cl = train_data[,5], k=kvalues[i])
    #calculates prediction accuracy
    table <- table(current_model, test_data[,5])
    Single_accuracy[i,j] <- sum(diag(table))/sum(table)
  }

}
Average_accuracy <- rowSums(Single_accuracy)/k
Average_accuracy

#--Stability Plot---
#creates bounds for points on plot showing deviation
Single_accuracy_df <- data.frame(Single_accuracy)
accuracy_sd <- rep(NA,dim(Single_accuracy_df)[1])
for (i in 1:dim(Single_accuracy_df)[1]){
  accuracy_sd[i] <- sd(Single_accuracy_df[i,])
}
Indx <- seq(1,length(kvalues))
plot(Indx,Average_accuracy,pch=19,
```

```
     col=ifelse(Indx==which.max(Average_accuracy), "red", "blue"),
     xlab="K Value", ylab="Prediction Accuracy",cex.lab=1.4,
     cex.axis=1.4,ylim=c(0.6,0.8))
lines(Indx,Average_accuracy-accuracy_sd, col="red", lwd =2)
lines(Indx,Average_accuracy+accuracy_sd, col="red", lwd =2)

# ----SVM----
library(kernlab)

#--Produces vector of input combinations--
#Matrix of all combinations of 7 input elements using allCombs function
library(dagR)
x <- seq(1,10)
mat <- (allCombs(x))
mat <- replace(mat,mat==1,"coinNum")
mat <- replace(mat,mat==2,"teamSize")
mat <- replace(mat,mat==3,"overallrating")
mat <- replace(mat,mat==4,"ratingTeam")
mat <- replace(mat,mat==5,"ratingProduct")
mat <- replace(mat,mat==6,"acceptingCurrencyNum")
mat <- replace(mat,mat==7,"socialMedia")
mat <- replace(mat,mat==8,"teamLinkedIn")
mat <- replace(mat,mat==9,"teamPhotos")
mat <- replace(mat,mat==10,"date_diff")
mat <- replace(mat,is.na(mat),"")

formulas <- rep(NA,length.out=1024)
#formats matrix, adding plus signs and removing plus signs where necessary
for (i in 1:length(formulas)){
  string <- paste(mat[i,], sep = "", collapse = "+")
  formulas[i] <- (gsub("\\+\\+.*","",string))
  if (endsWith(formulas[i],"+")==TRUE){
    formulas[i]<- substr(formulas[i],1,nchar(formulas[i])-1)
  }
  formulas[i] <- paste("goal~",formulas[i],sep = "", collapse = "")
}
#removes quotation marks and zero entry
formulas <- noquote(formulas)
formulas <- formulas[-1]


#--Cross Validation Strategy Used--
set.seed(121)

#folds
k = 5

#matrix of all models with value - accuracy to be filled
Single_accuracy<-matrix(nrow=length(formulas), ncol=k)

#consider fold 1 to k
for (j in 1:k){
```

```
  #creates testing and training sets randomly, test=20% of dataset
  idx <- sample(1:1181,236)
  train_data <- ICO_normalised[-idx, ]
  test_data <- ICO_normalised[idx, ]

  #applies ksvm model for current formula , predicts using test data and
  #outputs accuracy
  for (i in 1:length(formulas)){
    current_model <- ksvm(as.formula(formulas[i]), data = train_data,
                          kernel = "vanilladot")
    #predicts using test data, outputting raw probabilities
    predictions= predict(current_model, newdata=test_data)
    #calculates prediction accuracy
    table <- table(predictions, test_data$goal)
    Single_accuracy[i,j] <- sum(diag(table))/sum(table)
  }

}
Average_accuracy <- rowSums(Single_accuracy)/k

#--Prediction Plot---
#creates heat map of predictions
Single_accuracy_df <- data.frame(Single_accuracy)
accuracy_sd <- rep(NA,dim(Single_accuracy_df)[1])
for (i in 1:dim(Single_accuracy_df)[1]){
  accuracy_sd[i] <- sd(Single_accuracy_df[i,])
}
Indx <- seq(1,length(formulas))
plot(Indx,Average_accuracy,pch=19,col=ifelse(Indx==which.max(Average_accuracy),
     "black", ifelse(Average_accuracy>0.73,
    "#FF2900",ifelse(Average_accuracy>0.65,
    "#FF5200",ifelse(Average_accuracy>0.58,"#FF7B00","#FFA500")))),
     xlab="Model Combination Index", ylab="Predictive Accuracy",cex.lab=1.4,
     cex.axis=1.4)
legend(700, .6, legend=c("overallrating", "teamLinkedIn/date_diff",
                         "acceptingCurrencyNum", "CoinNum"),
       col=c("#FF2900", "#FF5200","#FF7B00","#FFA500"), pch=19, cex=1.2)

# ----------------------------------------
# ----------------------------------------
# -------------- Evaluation --------------
# ----------------------------------------
# remember to run formatting code in decision tree
library(ROCR)
library(caret)
library(randomForest)
library(class)
library(rpart)
library(gmodels)
library(kernlab)
set.seed(100)
# ----Decision Tree----
# Generate Model
```

```
idx <- sample(1:1181,236)
train_data <- ICO_categorical[-idx, ]
test_data <- ICO_categorical[idx, ]
mytree <- rpart(goal~video+GitHub+CEOPhoto, data=ICO_categorical,
                method='class',control=rpart.control(0.01))
raw_probabilities= predict(mytree, newdata=test_data)
raw_probabilities = raw_probabilities[, 2]

#converts raw prediction probabilities to most likely output
predicted_output <- rep("N", dim(test_data)[1])
predicted_output[raw_probabilities > .5] = "Y"

# confusion matrix
cm_DT <- confusionMatrix(as.factor(predicted_output),
                         test_data$goal, positive = "Y")


# create the prediction object (particularly used by the ROCR package)
pred_object <- prediction(raw_probabilities, test_data$goal)
# calculate the ROC curve
roc_DT <- performance(pred_object, measure = "tpr", x.measure = "fpr")
auc_DT <- performance(pred_object, measure = "auc")

# ----Random Forest----
# Generate Model
idx <- sample(1:1181,236)
train_data <- ICO_categorical[-idx, ]
test_data <- ICO_categorical[idx, ]
myforest <- randomForest(goal ~ video + CEOPhoto + Banking + USA + Unknown +
                           whitepaper, data=train_data,type="classification")
raw_probabilities= predict(myforest, newdata=test_data,type="prob")
raw_probabilities = raw_probabilities[, 2]

#converts raw prediction probabilities to most likely output
predicted_output <- rep("N", dim(test_data)[1])
predicted_output[raw_probabilities > .5] = "Y"

# confusion matrix
cm_RF <- confusionMatrix(as.factor(predicted_output),
                         test_data$goal, positive = "Y")

# create the prediction object (particularly used by the ROCR package)
pred_object <- prediction(raw_probabilities, test_data$goal)
# calculate the ROC curve
roc_RF <- performance(pred_object, measure = "tpr", x.measure = "fpr")
auc_RF <- performance(pred_object, measure = "auc")

# ----KNN----
# data used
knn_data<-ICO_normalised[,-c(1,6,7,8,9,10)]
# Generate Model
idx <- sample(1:1181,236)
train_data <- knn_data[-idx, ]
```

```
test_data <- knn_data[idx, ]
myknn <- knn(train = train_data[,-5], test = test_data[,-5],
             cl = train_data[,5], k=55, prob=T)
# confusion matrix
cm_knn <- confusionMatrix(as.factor(myknn), test_data$goal, positive = "Y")


messyprobs <- attr(myknn,"prob")
for (i in 1:length(myknn)){
  if (myknn[i]=="N"){
    messyprobs[i] <- 1-messyprobs[i]
  }
}

# create the prediction object (particularly used by the ROCR package)
pred_object <- prediction( messyprobs, test_data$goal)
# calculate the ROC curve
roc_knn <- performance(pred_object, measure = "tpr", x.measure = "fpr")
auc_knn <- performance(pred_object, measure = "auc")
# ----SVM----
# data used
# Generate Model
idx <- sample(1:1181,236)
train_data <- ICO_normalised[-idx, ]
test_data <- ICO_normalised[idx, ]
mysvm <- ksvm(goal~overallrating + teamLinkedIn + date_diff, data = train_data,
              kernel = "vanilladot",prob.model=T)

predictions= predict(mysvm, newdata=test_data,type="prob")
predictions <- predictions[,2]

#converts raw prediction probabilities to most likely output
predicted_output <- rep("N", dim(test_data)[1])
predicted_output[predictions > .5] = "Y"

# confusion matrix
cm_svm <- confusionMatrix(as.factor(predicted_output),
                          test_data$goal, positive = "Y")

# create the prediction object (particularly used by the ROCR package)
pred_object <- prediction( predictions, test_data$goal)
# calculate the ROC curve
roc_svm <- performance(pred_object, measure = "tpr", x.measure = "fpr")
auc_svm <- performance(pred_object, measure = "auc")
# ----ROC Curve----
plot(roc_DT, col = "blue", lwd = 2)
plot(roc_RF, col = "red", lwd = 2, add = TRUE)
plot(roc_knn, col = "green", lwd = 2, add = TRUE)
plot(roc_svm, col = "black", lwd = 2, add = TRUE)
# this line indicates the performance of random guess (50:50)
abline(a = 0, b = 1, lwd = 2, lty = 2)
legend(0.6,0.5,legend=c("SVM","KNN","Random Forest","Decision Tree","Baseline"),
       col=c("black","Green","Red","Blue","Black"),lty=c(1,1,1,1,2),lwd=2,cex=1
```

```
       ,bty="n")
# ----AUC Values----
auc_DT@y.values[[1]]
auc_RF@y.values[[1]]
auc_knn@y.values[[1]]
auc_svm@y.values[[1]]
# ----Confusion matrices: Precision vs Recall----
draw_confusion_matrix <- function(cm) {

  layout(matrix(c(1,1,2)))
  par(mar=c(2,2,2,2))
  plot(c(100, 345), c(300, 450), type = "n", xlab="", ylab=""
       , xaxt='n', yaxt='n')
  title('Decision Classification Tree', cex.main=2)

  # create the matrix
  rect(150, 430, 240, 370, col='#3F97D0')
  text(195, 435, 'N', cex=1.2)
  rect(250, 430, 340, 370, col='#F7AD50')
  text(295, 435, 'Y', cex=1.2)
  text(125, 370, 'Predicted', cex=1.3, srt=90, font=2)
  text(245, 450, 'Actual', cex=1.3, font=2)
  rect(150, 305, 240, 365, col='#F7AD50')
  rect(250, 305, 340, 365, col='#3F97D0')
  text(140, 400, 'N', cex=1.2, srt=90)
  text(140, 335, 'Y', cex=1.2, srt=90)

  # add in the cm results
  res <- as.numeric(cm$table)
  text(195, 400, res[1], cex=1.6, font=2, col='white')
  text(195, 335, res[2], cex=1.6, font=2, col='white')
  text(295, 400, res[3], cex=1.6, font=2, col='white')
  text(295, 335, res[4], cex=1.6, font=2, col='white')

  # add in the specifics
  plot(c(100, 0), c(100, 60), type = "n", xlab="", ylab="",
       main = "DETAILS", xaxt='n', yaxt='n')
  text(10, 85, names(cm$byClass[1]), cex=1.2, font=2)
  text(10, 70, round(as.numeric(cm$byClass[1]), 3), cex=1.2)
  text(30, 85, names(cm$byClass[2]), cex=1.2, font=2)
  text(30, 70, round(as.numeric(cm$byClass[2]), 3), cex=1.2)
  text(50, 85, names(cm$byClass[5]), cex=1.2, font=2)
  text(50, 70, round(as.numeric(cm$byClass[5]), 3), cex=1.2)
  text(70, 85, names(cm$byClass[6]), cex=1.2, font=2)
  text(70, 70, round(as.numeric(cm$byClass[6]), 3), cex=1.2)
  text(90, 85, names(cm$byClass[7]), cex=1.2, font=2)
  text(90, 70, round(as.numeric(cm$byClass[7]), 3), cex=1.2)

}
draw_confusion_matrix(cm_DT)

# ----CV metrics----
# run respective model first
```

```r
x <- Single_accuracy[which.max(Average_accuracy),]
# standard deviation
sd(x)
# MSE
mean((x - mean(x))^2)
# RMSE
sqrt(mean((x - mean(x))^2))
# MAE
mean((abs(x - mean(x))))
```