Leeds University Business School

UNIVERSITY OF LEEDS

Assessed Coursework Coversheet

For use with *individual* assessed work

Student ID Number:	2	0	1	4	8	8	7	4	5
Module Code:	LUBS5990M								
Module Title:	Machine Learning in practice								
Module Leader:	Dr Xingie Wei								
Declared Word Count:	2979								

Please Note:

Your declared word count must be accurate, and should not mislead. Making a fraudulent statement concerning the work submitted for assessment could be considered academic malpractice and investigated as such. If the amount of work submitted is higher than that specified by the word limit or that declared on your word count, this may be reflected in the mark awarded and noted through individual feedback given to you.

It is not acceptable to present matters of substance, which should be included in the main body of the text, in the appendices ("appendix abuse"). It is not acceptable to attempt to hide words in graphs and diagrams; only text which is strictly necessary should be included in graphs and diagrams.

By submitting an assignment you confirm you have read and understood the University of Leeds **Declaration of Academic Integrity** (http://www.leeds.ac.uk/secretariat/documents/academic_integrity.pdf).

Introduction –

The cryptocurrency industry's equivalent of an initial public offering (IPO) is an initial coin offering (ICO) (IPO). An ICO is a method of raising funds for a company aiming to establish a new coin, app, or service.

If the funds raised fall short of the firm's minimal requirements, the money may be returned to the investors; at this point, the ICO is considered a failure. The money received is utilized to pursue the project's aims if the fundraising criteria are reached within the given timeframe (Frankenfield, 2020).

This report details, solving a binary classification problem of success or failure using the dataset provided in order to predict the success or failure of any ICO given that data for the same features are provided.

Data Understanding-

The dataset provided for this analysis consists of 20 features and 1606 records. The success of an ICO is determined by its class attribute '**success**'. A detailed understanding of the features of this data set is mentioned below.

1. **Id –** This is nothing but an identity column for each record in the dataset. The id column does not hold any functional importance in the data.

2. **Success –**It is an indicator column with values '0' and '1' where a value of '1' means the ICO project has been a success and a value of '0' means that it was unsuccessful.

   Below pie chart, tells the proportion of successful ICOs and unsuccessful ICOs in the dataset.

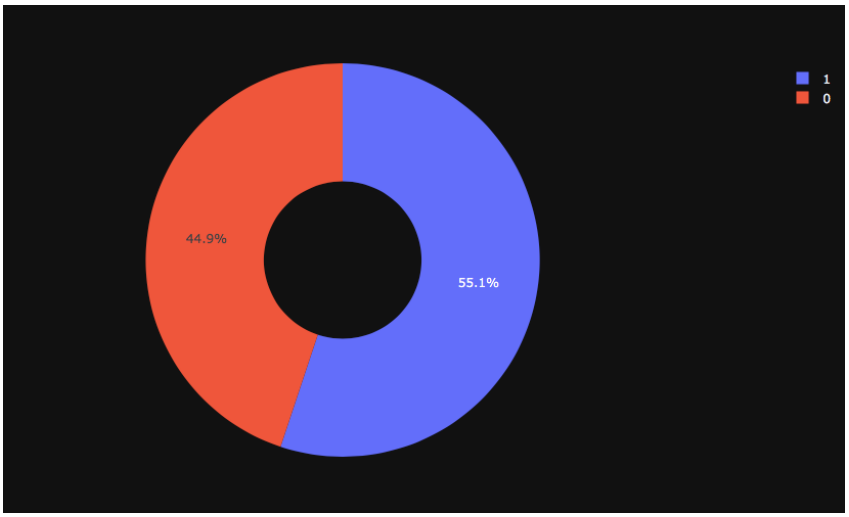


Fig – 1.1

Of the 1606 records present in the dataset, 885 have been successful and 721 unsuccessful.

3. **tokenNum –** This feature represents the token numbers of a project i.e. that is the number of tokens to be issued. Higher number of tokens issued usually signify lower price for each token. This feature in the dataset has missing values for 246 records. Mentioned below are the numerical summaries of this feature.

Mean	Standard Deviation	Min	25% quantile	Median	75% quantile	Max
2.015268e +13	6.264666e +14	0.000000e +00	7.000000e +06	5.500000e +07	2.514804e +08	2.260000e +16

4. **teamSize-** Team size gives the number of members working in a particular ICO campaign. The below histogram depicts the distribution of this feature.

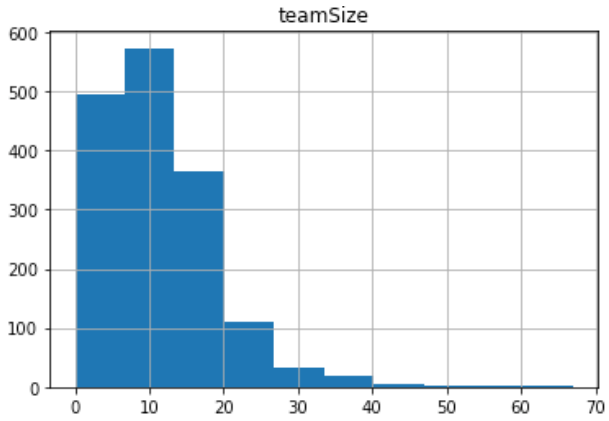


Fig 1.2

It can be observed from this figure that the distribution centers around the mean value of 10 with a long right tail.

5. **Country-** Country name where the ICO project is located. The feature contains data for 177 countries. Below are the top 10 countries in terms of number of ICOs.

Country	Number of ICOs
USA	262
Russia	155
UK	135
Singapore	133
Switzerland	79
Estonia	65
Hong Kong	43
Canada	38
Germany	32
Australia	31

The below graph depicts the count of ICO projects located in different countries.

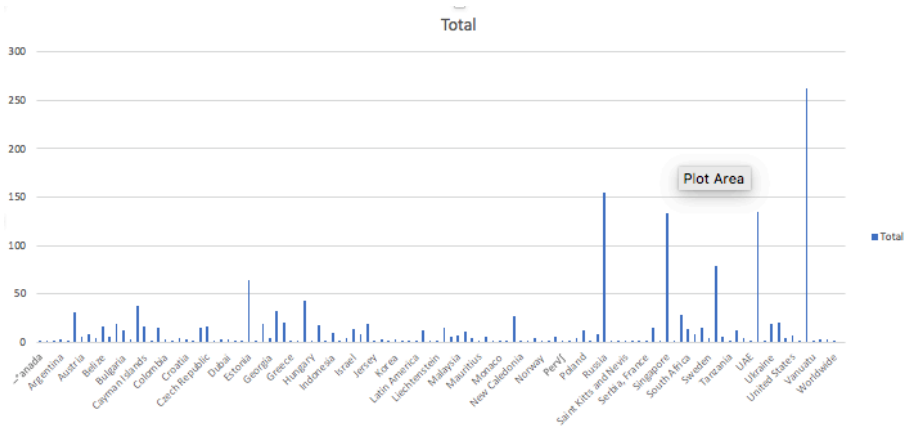


Fig 1.3

6. **Categories-** Category of the ICO project i.e. the industry or the sector that the ICO project belongs to. An ICO project can fall in several categories. Mentioned below are the top 10 categories in terms of number of ICO projects belonging to them.

Category	Count
Cryptocurrency	97
Platform	86
Entertainment	43
Investment	40
Cryptocurrency,Platform	29
Business services	28
Business services,Platform	27
Real estate	26
Platform,Business services	24
Platform,Cryptocurrency	21

7. **Overall rating –** Rating given by investment experts to the quality of an ICO project. The ratings range between 1 to 5 with 1 being the lowest. The rating takes into account several factors such as the team size, the platform, white papers, platform price etc. Below diagram depicts the histogram of this feature.

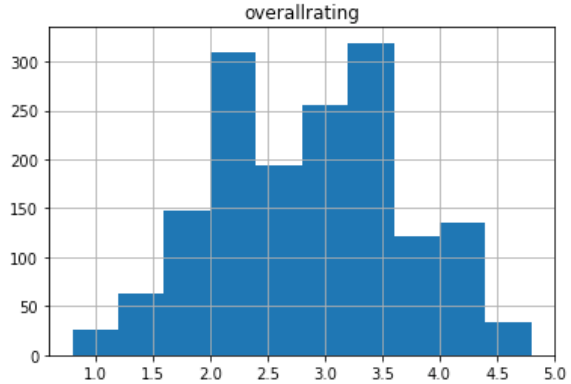


Fig 1.4

The numerical summaries for this feature are mentioned below –

Mean	Standard Deviation	Min	25% quantile	Median	75% quantile	Max
2.850125	0.817257	0.800000	2.300000	2.800000	3.500000	4.800000

8. **Offered_ownership-** Is the percentage of tokens given to investors from the total number of tokens created. Retaining more number of tokens by the ICO company themselves signifies the commitment of the company towards the success of the project. The mean values of the offered_ownership is 1.143. A value of 601.250 has also been spotted which is an outlier.

9. **Start date –** Signifies the start date of the fundraising campaign for a particular ICO. The minimum date value for this column is 04/08/2015 and maximum is 01/12/2018. The average team size is around 11 members.

10. **End date -** Signifies the end date of the fundraising campaign for a particular ICO. The minimum date value for this column is 03/09/2015 and maximum is 01/11/2018.

11. **tokenName-** This is a categorical feature representing the name of the token for each ICO project.

12. **tokenPrice-** This feature tells us the price of the token issued by the ICO project. The feature contains data in uneven formats. Some of the conversions are represented in USD or other popular cryptocurrencies like ETH, BTC with a '=' sign in between such as - **1 TNB = 0.00005 ETH** whereas some have a range of values separated by a '-' such as - **1 ETH = 600-800 AIM** as well as some containing description in words such as – **Determined after pre-sale**. Since this feature is an important metric of measurement, it has been taken well care of in data preparation stage.

13. **Token type-** This feature represents the token type with 43 unique values in them and the most popular value being – **ERC20**.

14. **Platform –** Blockchain technologies are used for crypto currencies. The feature tells us the block chain platform for a given token in the dataset. Some tokens use a combination of platforms and those platform names are separated by a comma. Mentioned below are the top 10 platforms in terms of the number of tokens based on them.

Platform	Count
Ethereum	1307
WAVES	58
NEO	8
Scrypt	8
NEM	7
Separate blockchain	7
Stellar	7
Bitcoin	5
BitShares	4
Ethereum and Waves	4

15. **Accepting currency -** To purchase the tokens or cryptocurrencies, some other form of payment should be accepted, and the token should be returned for that amount. This feature represents the currencies that are accepted for a certain token transaction. The majority of the projects accept Ethereum, Bitcoin, Waves, USD, and EUR as payment methods.

16. **Soft Cap -** If the ICO project states a minimum fundraising target on their campaign website, this binary variable is set to 1, else it is set to 0. Setting a limit means the team has calculated the amount of money needed to complete the project. The below pie chart depicts the proportion of these values in the dataset.

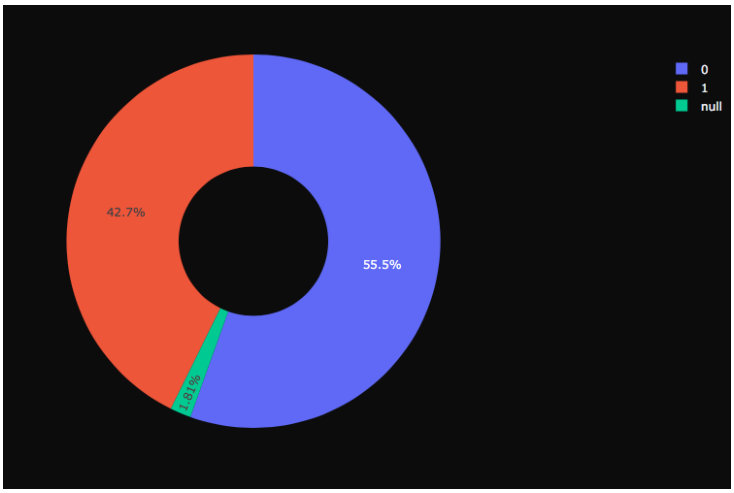


Fig 1.5

17. **Hard Cap -** If the ICO project stated a maximum fundraising target on their website, this indicator variable is set to 1, else it is set to 0. Setting a limit means the team has calculated the amount of money needed to complete the project. The below pie chart depicts the proportion of these values in the dataset.

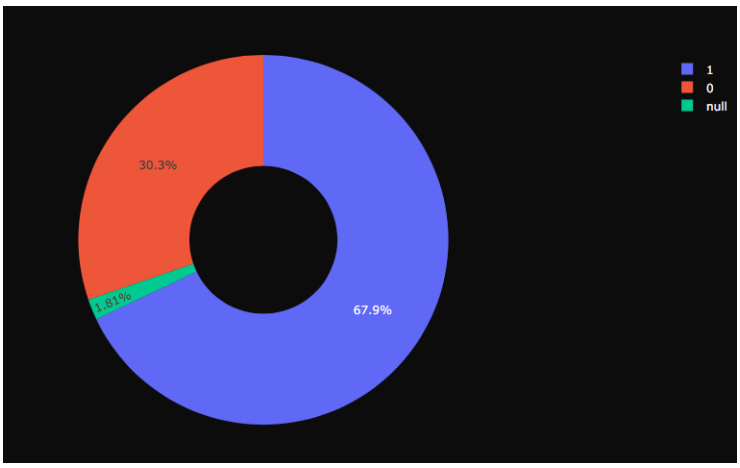


Fig 1.6

18. **Whitepaper-** This is an indicator variable to determine whether a particular project has published any whitepaper. 1 indicates a whitepaper has been provided and 0 otherwise. The below pie chart depicts the proportion of these values in the dataset.

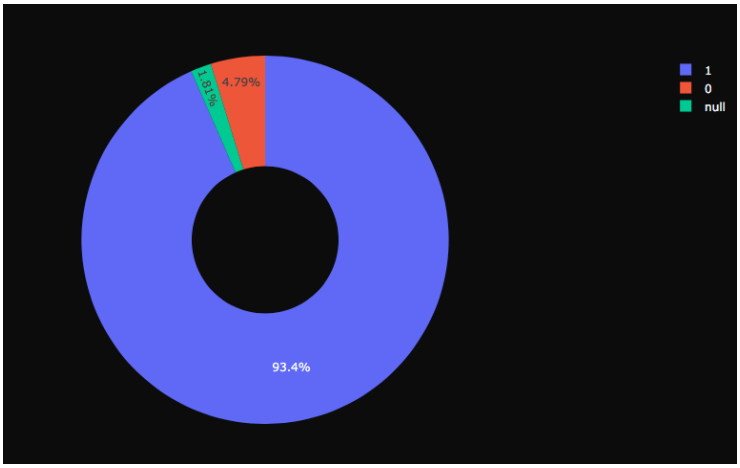
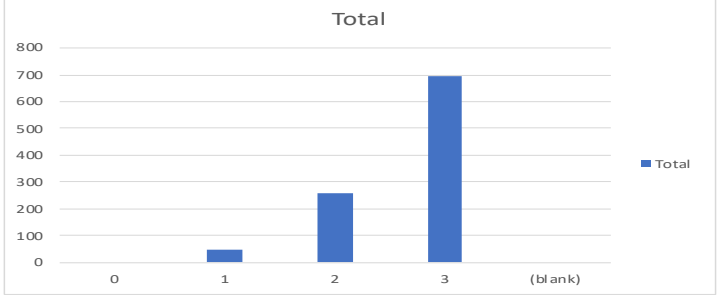


Fig 1.7

19. **Video –** This is an indicator variable to determine whether, a video has been provided by the ICO project. 1 indicates video provided and 0 indicates otherwise.

20. **SocialMedia –** This feature helps determine the activity level of an ICO team in the different social media platforms. The rating is given in a scale of 1 to 3 with 1 meaning least active and 3 being the most active. From the below bar graph, it can be seen that most teams are highly active with 696 records having the value 3.



In order to determine relationships among the features of the dataset, a correlation matrix has been built with selected feature which are deemed influential. Those are success, tokenNum, teamSize, overallrating, offered_ownership, softcap, hardcap, whitepaper, video, social media.

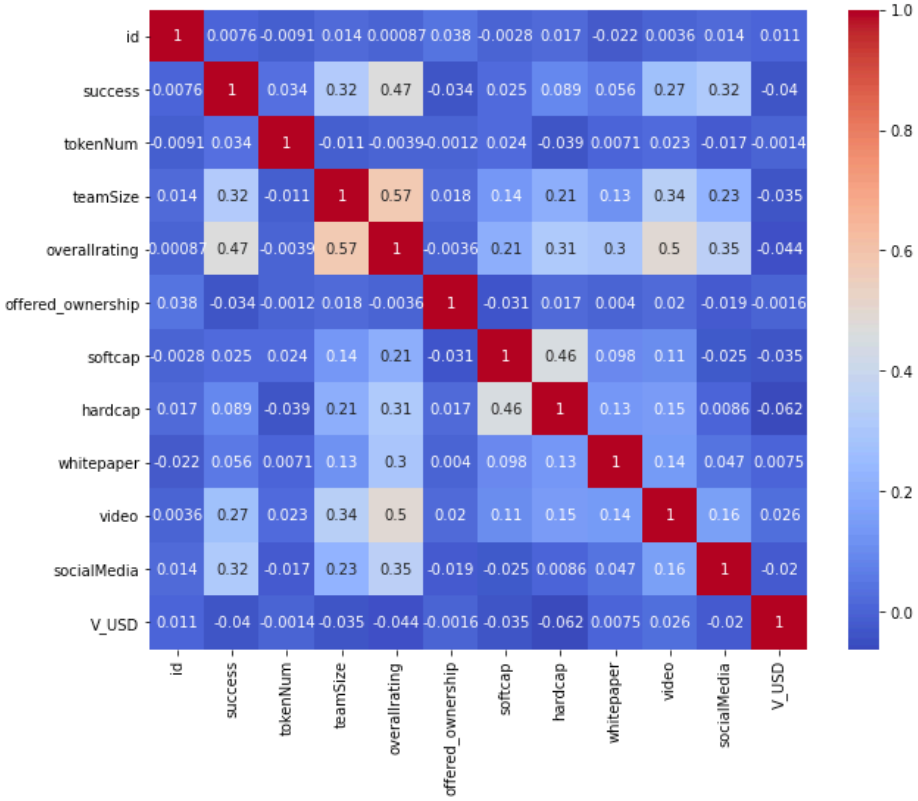


Fig 1.8

There are no correlations observed between most of the features however, moderately positive correlation has been observed between teamSize & overallrating and overallrating & success as well as video & overallrating.

Data Preparation –

Most dataset before being fitted into a model requires some amount of preprocessing or preparation to be done on them in order to address the inconsistencies present in them. This stage of a machine learning or a data science project takes the most amount of time. This activity is very crucial since if the anomalies in the data are not taken care of properly, it might lead to misleading results produced by the model during the prediction task.

- **tokenPrice –** The token price column as mention earlier in data preparation section is a very important column for the metric it represents. There are numerous inconsistencies in the data for this feature which have been handled step by step. First the special characters like “,”, “;”,”<”,”>” were taken care of.

The column values were then split on the ‘=’ operator dividing the string into two parts stored in two different columns. The column values were then swapped keeping string with ETH/USD in the right column(T2) and the other on the left(T1) as cited below.

tokenPrice
1 REDL = 1 USD
1 SLOGN= 0.0001 ETH
1 TNB = 0.00005 ETH
1GIF = 0.00969635 USD

Step 1

T1	T2
1 REDL	1 USD
1 SLOGN	0.0001 ETH
1 TNB	0.00005 ETH
1GIF	0.00969635 USD

Step 2

The numbers and characters in the string in T1 and T2 were then seperated to be stored in different columns resulting in –

V1	V2	D1
1	1	USD
1	0.0001	ETH
1	0.00005	ETH
1	0.00969635	USD

Step 3

There were several entries in this column which had a range. For those the mean of the range has been stored in column V2 as shown in the below example.

tokenPrice
0.95 - 1 USD
1 RTC = 0.8-1 USD
1 CBC = 0.50 - 0.75 USD
1 MIRA = 0.7-1.0 USD

Step 1

T1	T2
xxx	0.95 - 1 USD
1 RTC	0.8-1 USD
1 CBC	0.50 - 0.75 USD
1 MIRA	0.7-1.0 USD

Step 2

V1	V2	D1
	0.975	USD
1	0.9	USD
1	0.625	USD
1	0.85	USD

Step 3

After separating the token name, their values as well their accepting currencies and their values as shown above, we store the accepting currencies in the column **D1**. Now with most of the values in **D1** being **ETH**, **BTC**, **USD** and **EUR** we could convert all the price of the tokens in these accepting currencies to **USD**. For this, we imported the historical data for **ETH**, **BTC**, **EUR** to **USD** conversion and looked up on the basis of the End date in our dataset to fetch the conversion rate which when multiplied by **(V2/V1)** gave us the token prices in **USD** stored in a separate column – **V_USD**. The resultant data looks like the below.

V1	V2	D1	V_USD
	0.975	USD	0.975
1	0.9	USD	0.9
1	0.625	USD	0.625
1	0.85	USD	0.85

The token price column was then dropped along with the intermediate columns which were created for this processing retaining **V_USD** among these.

- **Duration-** In order to handle the time period of the ICO projects, the difference between the features **End date** and **Start date** were calculated and stored in the column called **duration**. The date columns were dropped after that.

- **Qualitative features –** Qualitative feature or catergorical variables are variables which hold non-numeric, descriptive values. Before modelling, the categorical data needs to be converted to numeric representations. This process was done using one hot encoding technique for the features- **categories**, **country**, **acceptingCurrency** and **platform**. Once the encoding was successfully done, these features were dropped from the dataset. The features **tokenType** and **tokeName** were omitted from encoding since **tokeName** is unique for each **ICO** project.

- **Missing values –** Mentioned below are the features and their count of missing values-

Feature Name	Number of missing values	Percentage Missing
tokenNum	247	15.4
offered_ownership	541	33.7
Duration	5	0.3
softcap	29	1.8
hardcap	29	1.8
whitepaper	29	1.8
video	29	1.8
socialMedia	29	1.8
V_USD	248	15.4

There are various methods of handling missing values. The simplest being imputing them with the mean or median of the respective features. However, doing so would result in the missing values of a feature imputed with the same value. This could potentially make the dataset skewed. Hence the technique applied was **KNN** imputation. This technique predicts for the missing values in the dataset using the nearest neighbor and replaces them with the credible predicted value.

After performing all the steps mentioned above out final dataset was ready for modelling.

Modelling –

Once the dataset was cleaned and ready for modelling. The dataset had to be divided into training set and test set.

The training set is used to train the model based on the underlying algorithm whereas the test set is used to test the accuracy of the model once trained. For this purpose, the class attribute (‘success’) is dropped from the test dataset whereas it is retained in the training dataset.

We split our dataset in the 70:30 ratio for training and testing respectively i.e. 70% of the data would be used for training whereas 30% will be used for testing. We also performed k-fold cross validation with 5 folds in place in order to assess the accuracy of the classifiers. In k-fold cross validation, the models are executed k times for training with 90% of data for training and 10% data for testing and the data shuffled for each execution.

The next step is to select the appropriate machine learning algorithms for our task. Since it is a classification problem with binary outcome that we are dealing with, we needed supervised machine learning algorithms to perform the task.

We selected the below algorithms for modelling. Hyperparameter tuning was done using the grid search technique (Jordan, 2017) in order to choose the optimal hyper parameters to be supplied to the learning algorithms.

- **Decision Tree -** One of the predictive modelling techniques used in statistics, data mining, and machine learning is decision tree, also known as induction of decision trees. It goes from observations about an item (represented in the branches) to inferences about the item's goal value using a decision tree (as a predictive model) (represented in the leaves) using the concept of entropy and information gain at each level.

  The decision tree classifier was built using the parameters – **{ max_depth=4, random_state=42}** and yielded an accuracy of **77.59%**. Below is the confusion matrix for the same.

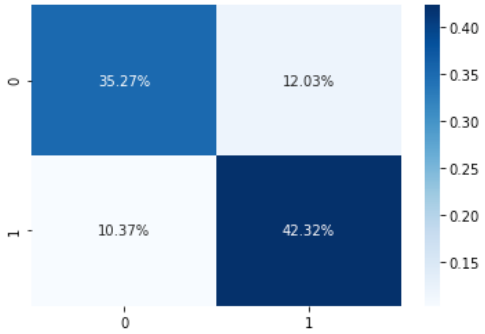


Fig 1.9 – Confusion matrix(Decision Tree)

Mentioned below is the complete decision tree generated by this classifier.

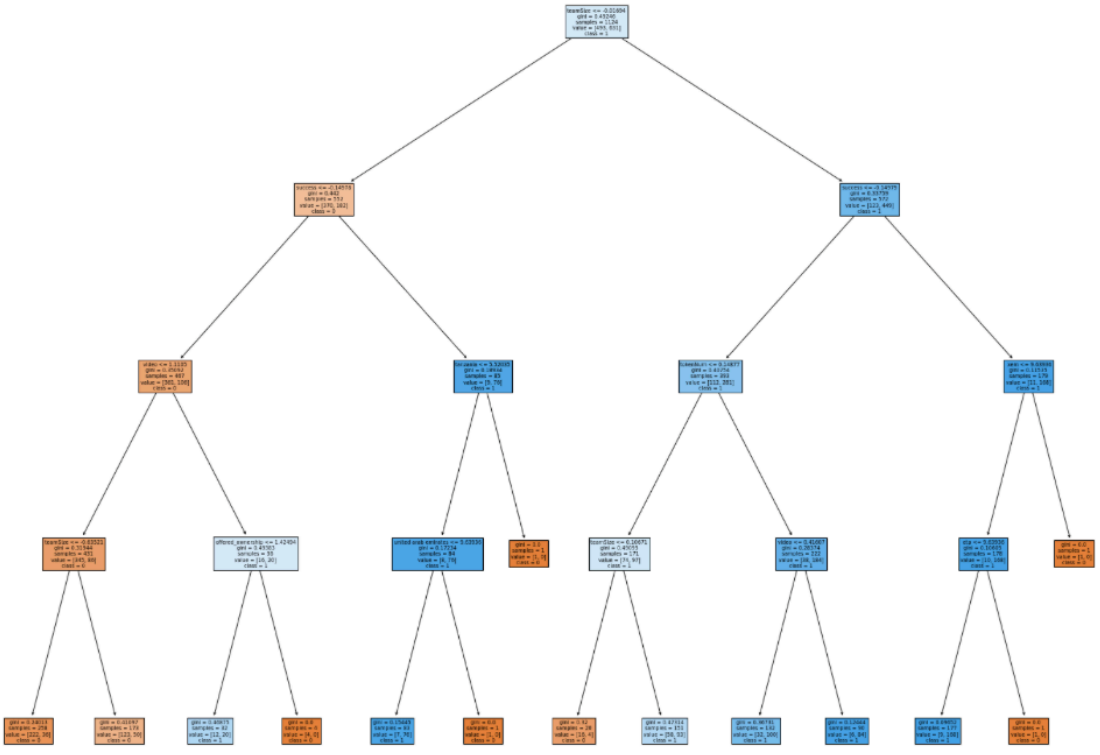


Fig 2.0 – Decision Tree Visual

- **Support Vector Machine (SVM) –** SVM is a learning method developed by Vapnik in 1995 to tackle the problem of pattern recognition in two classes. The technique is specified over a vector space, with the objective of locating a decision surface that best divides the data into two groups (Ayodele, 2010). SVM maps training examples to points in space in order to widen the distance between the two categories as much as possible. New instances are then mapped into the same space and classified according to which side of the gap they land on. Due to its capacity to eliminate the requirement for feature selection, SVM can obtain the best results in text classification (Alshutayri et al., 2016).

  The SVM classifier for this task was built using the parameters – **{ kernel='rbf', random_state= 42}** which yielded an accuracy of **66.8%**. Below is confusion matrix regarding the same.

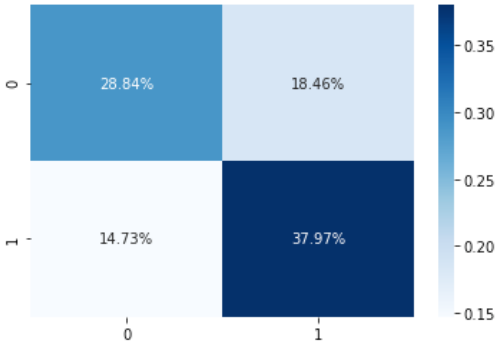


Fig 2.1 – Confusion matrix(SVM)

- **Random Forest -** Random forests, also known as random decision forests, are an ensemble learning technique for classification, regression, and other problems that employs a huge number of decision trees to train. The random forest's output for classification problems is the class picked by the majority of trees.

  The random forest classifier for this task was built using the parameters – **{ n_estimators=200,min_samples_leaf=2, random_state=42}** which yielded an accuracy of **75.31%**. Below is the confusion matrix regarding the same.

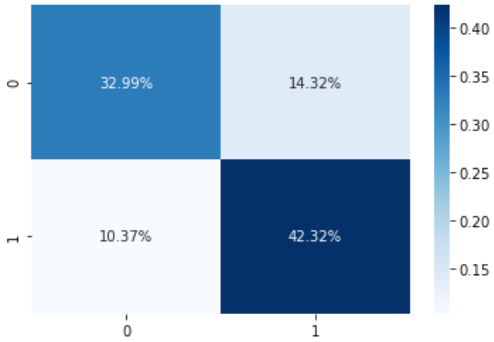


Fig 2.2 – Confusion matrix(Random Forest)

- **K Nearest Neighbors (KNN) -** The KNN algorithm implies that objects that are similar are near together i.e. comparable items are close together. The input consists of the k closest training examples in data set.(Harrison, 2018) The result of k-NN classification is a class membership. The object is assigned to the most common class among its k nearest neighbors, based on a majority vote of its neighbors. If k = 1, the item is simply assigned to the class of the item's closest neighbor.

  The KNN classifier was built using the parameters – **{ n_neighbors=42, leaf_size=20}** and yielded an accuracy of **68.04 %**. Below is the confusion matrix regarding the same.

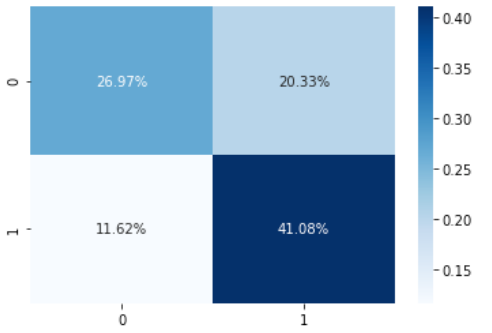


Fig 2.3 – Confusion matrix(KNN)

- **AdaBoost -** It is a meta-estimator that starts by fitting a classifier on the original dataset, then fits successive copies of the classifier on the same dataset, but adjusts the weights of erroneously classified instances such that future classifiers focus more on difficult cases. (Ref - Y. Freund, R. Schapire, “A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting”, 1995.)

The AdaBoost classifier was built using the parameters **{learning_rate= 0.2,n_estimators= 50,random_state=42}** which yielded an accuracy of **79.25%**. Below is the confusion matrix regarding the same.

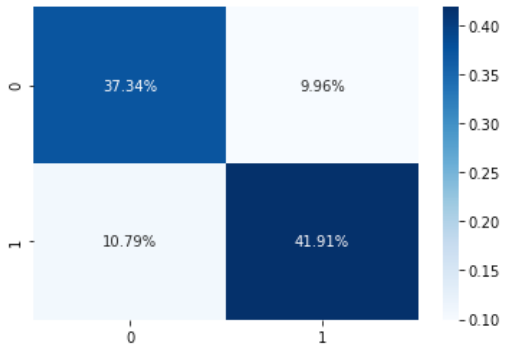


Fig 2.4 – Confusion matrix(AdaBoost)

Evaluation-

In this stage we compare the performance of the classifiers built above based on various key performance indicators in order to choose the best classifier for our task. As seen in Fig – 1.1 the class variable '**success**' has nearly equal proportion of occurrences of its values, which makes it a balanced dataset. In this case **Accuracy** is the best indicator to assess the performance. However, there are other key indicators also which should be taken into consideration since it is not perfectly balanced. They are –

- **Precision –** It is defined as the number of real positives among the total number of the model's predicted positive also known as Positive Predictive Value(PPV). A model with high precision will only forecast the positive class in instances where the outcome is extremely likely to be positive. Calculated as –

$$\text{Precision} = \frac{true\ positives}{true\ positives + false\ positives}$$

- **Recall –** It is defined as the ratio between the number of true positives and the total number of positives. A large portion of the positive samples are captured by a high recall model. Calculated as –

$$\text{Recall} = \frac{true\ positives}{true\ positives + false\ negatives}$$

- **F score -** The F-score, which is defined as the harmonic mean of the model's accuracy and recall, is a means of combining the model's precision and recall. Calculated as –

$$F = \frac{2 * Precision * Recall}{Precision + Recall}$$

The below table of results provides a combined view of these performance indicators for all the classifiers built for this task. The results from both the testing methods – percentage split in ratio of 70:30 as well as cross validation using 5 folds are nearly the same. Hence, we shall consider the results from percentage split.

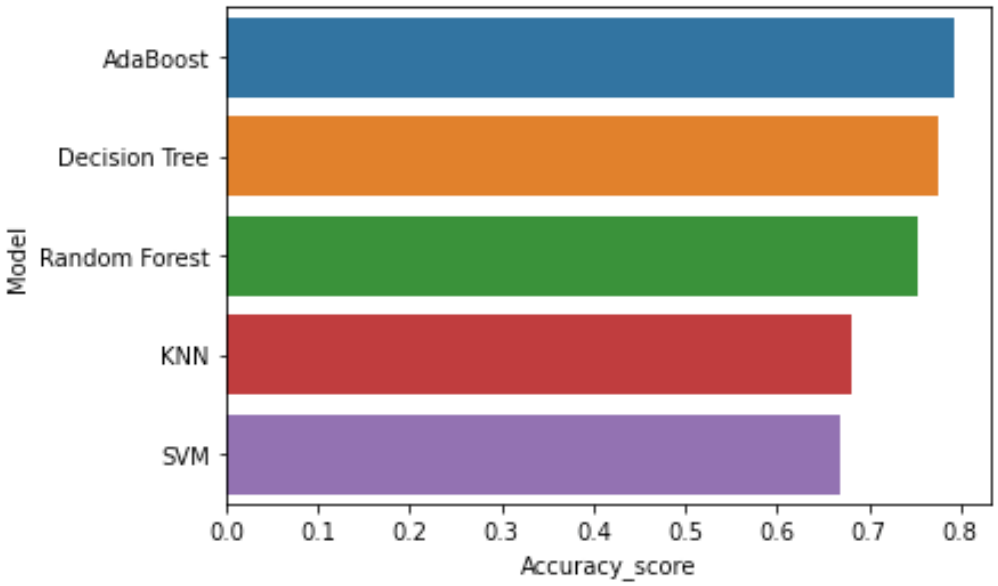
	Test Type	Accuracy %	Precision	Recall	F1 - Score
Decision Tree	Percentage Split (70:30)	77.59	0.78	0.8	0.79
	K-fold Cross Validation (K=5)	76.88	0.77	0.78	0.77
SVM	Percentage Split (70:30)	66.8	0.67	0.72	0.7
	K-fold Cross Validation (K=5)	66.3	0.65	0.69	0.71
Random Forest	Percentage Split (70:30)	75.31	0.75	0.8	0.77
	K-fold Cross Validation (K=5)	74.82	0.73	0.78	0.74
KNN	Percentage Split (70:30)	68.04	0.67	0.78	0.72
	K-fold Cross Validation (K=5)	68.36	0.64	0.75	0.68
AdaBoost	Percentage Split (70:30)	79.25	0.81	0.8	0.8
	K-fold Cross Validation (K=5)	79.33	0.83	0.81	0.85

It can be observed from this table that the AdaBoost classifier has performed the best in terms of all these performance indicators with Accuracy % = 79.25, Precision = 0.81, Recall = 0.80 and F Score = 0.80.

Conclusion –

For this task we have been able to demonstrate the different stages of a machine learning classification problem right from Explanatory Data Analysis (EDA) covered in data understanding to data preparation & preprocessing for modelling followed by building classifier models and evaluating those models. Our evaluation tells us that the AdaBoost classifier has outperformed the rest in all the aspects.

The overall classifier rankings in this task can be seen in the bar graph below.



References-

Frankenfield, J. 2020. *Intial Coin offering*.[online].[Accessed 17 August 2021]. Available from: https://www.investopedia.com/terms/i/initial-coin-offering-ico.asp

Jordan, J. 2017. *Hyperparameter tuning for machine learning models.*[online].[Accessed 17 August 2021] from: https://www.jeremyjordan.me/hyperparameter-tuning/

Ayodele, Taiwo. 2010. *Types of Machine Learning Algorithms*. 10.5772/9385.

A. Alshutayri, E. Atwell, A. Alosaimy, J. Dickins, M. Ingleby, and J. Watson. 2016. Arabic Language WEKA-Based Dialect Classifier for Arabic Automatic Speech Recognition Transcripts.In *Proceedings of VarDial'2016 Workshop on NLP for Similar Languages, Varieties and Dialects,* pp. 204-211

Harrison, O. 2018. *Machine learning basics with the K-Nearest Neighbor Algorithm.*[online].[Accessed 17 August 2021] from: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761