CO 544- Machine Learning and Data Mining Project Specification

Introduction:

It is vital that credit card companies are able to identify fraudulent credit card transactions so that customers are not charged for items that they did not purchase. Such problems can be tackled with Data Science and its importance, along with Machine Learning, cannot be overstated. This project intends to illustrate the modelling of a data set using machine learning to find a validity of a credit card. This problem includes modeling past credit card transactions with the data of the ones that turned out to be invalied. This model is then used to recognize whether a new credit card transaction is valid or not. Our objective here is to detect 100% of the failure transactions while minimizing the incorrect fraud classifications. In this process, we have focused on analysing and preprocessing the dataset and making predictions over it. Because we have to create a model from a given dataset and then we can learn from that model. If we can learn the relationship from this model to the observed validity of the credit card, then we can use that for predicting on new dataset.

In this problem a dataset with input and output values is given and therefore this problem can be identified as a **supervised machine learning** problem. By looking at the dataset we can see the output variable a category. Therefore this can be identified as a **classification problem.** So our intended goal is to make accurate predictions for a given set of input data by training our model using a supervised machine learning classification algorithm, which gives the best accuracy based on how we build our model.

Experimental techniques and methods:

- **Handling the missing values**

```
#remove ? from df_dataset

col_list_obj = ['A1', 'A2', 'A3', 'A4', 'A6', 'A9']
for col_name in col_list:
  df_dataset = df_dataset[df_dataset[col_name] != '?']
```

Figure01 : Code part for handling missing values

When preprocessing the training dataset, all the rows which include missing values are removed when creating the model. Then our training dataset does not include any missing/null value with it.

- **Dealing with Boolean and Categorical Data**

```
#boolean convet to 1, 0
df_dataset.A8 = df_dataset.A8.astype(int)
df_dataset.A11 = df_dataset.A11.astype(int)
df_dataset.A13 = df_dataset.A13.astype(int)
df_dataset.head()
```

Figure02 : Code part for converting boolean values into integer

All the columns that are in boolean data type, are converted to a numerical value.
Therefore ‘TRUE’ values are converted to numeric 1 and ‘FALSE’ values are converted to numeric 0

```
#convert values to numaric
df_dataset.A1.replace(['a', 'b'], [0, 1], inplace=True)
df_dataset.A3.replace(['u', 'y', 'l'], [0, 1, 2], inplace=True)
df_dataset.A4.replace(['g', 'p', 'gg'], [0, 1, 2], inplace=True)
df_dataset.A6.replace(['c', 'd', 'cc', 'i', 'j', 'k', 'm', 'r', 'q', 'w', 'x', 'e', 'aa', 'ff'], [0, 1, 2, 3, 4, 5, 6, 7, 8 ,9 ,10 ,11 ,12 ,13], inplace=True)
df_dataset.A9.replace(['v', 'h', 'bb', 'j', 'n', 'z', 'dd', 'ff', 'o'], [0, 1, 2, 3, 4, 5, 6, 7, 8], inplace=True)
df_dataset.A15.replace(['g', 'p', 's'], [0, 1, 2], inplace=True)
df_dataset.A16.replace(['Failure','Success'], [0, 1], inplace=True)
```

Figure03 : Code part for converting categorical values into a numeric value

All the columns which have categorical data, are converted to a numerical value according to its data range.

After doing all this preprocessing, a dataset with numerical values in its columns was generated in order to analyse easily.

- **Attribute selection based on accuracies of individual attributes.**

```
#apply SelectKBest class to extract top 10 best features
bestfeatures = SelectKBest(score_func=f_classif, k=15)

fit = bestfeatures.fit(X,y)

dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(X.columns)

#concat two dataframes for better visualization
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['Feature','Score']  #naming the dataframe columns
print(featureScores.nlargest(16,'Score'))  #score of features
```

Figure04 : Code part for attribute selection

This is the given values for accuracy scores for each individual attribute.

	Feature	Score
10	A11	527.437498
7	A8	155.263362
11	A12	101.192569
9	A10	64.778755
4	A5	25.449401
1	A2	16.265080
2	A3	15.665484
3	A4	15.665484
6	A7	14.381885
8	A9	10.241500
14	A15	7.327621
13	A14	4.398846
12	A13	1.671202
5	A6	0.664319
0	A1	0.507370

Figure05 : Accuracy scores on attributes

All the columns were applied to SelectKBest class and give them a score according to how they contribute to training data.
Then we have ordered the attributes in descending order of the scores.

Then the Accuracy of the model is checked by removing the least scored attribute one by one at a time. This shows the importance of the feature sets and thereby we can select the most important attribute set for training our model.

```
selectedFeatures = featureScores.nlargest(16,'Score')
df_15F = X[selectedFeatures['Feature'].values]
df_15F.head()

#Accuracy

def randomForest(dataFrame, target):

  #Create a RF Classifier
  clf = RandomForestClassifier(n_estimators=100, max_depth=2,random_state=42)
  scores = cross_val_score(clf, dataFrame, target, cv=3)

  return scores.mean()

def returnScoreDataFrameModels(dataFrame):
  lists3 = []

  for i in [15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]:
    lists3.append(randomForest(dataFrame.iloc[:,0:(i)], y))

  rows = ["randomForest"]

  data = np.array([lists3])
  randomForestScore = pd.DataFrame(data=data, index=rows).transpose()

  return randomForestScore
```

Figure06 : Code part for calculate the accuracy of set of attributes

Following graph shows the Random Forest Score of features which was received after removing the least scored attribute one by one at a time . Here the x and y axis shows the attribute and the scores respectively.

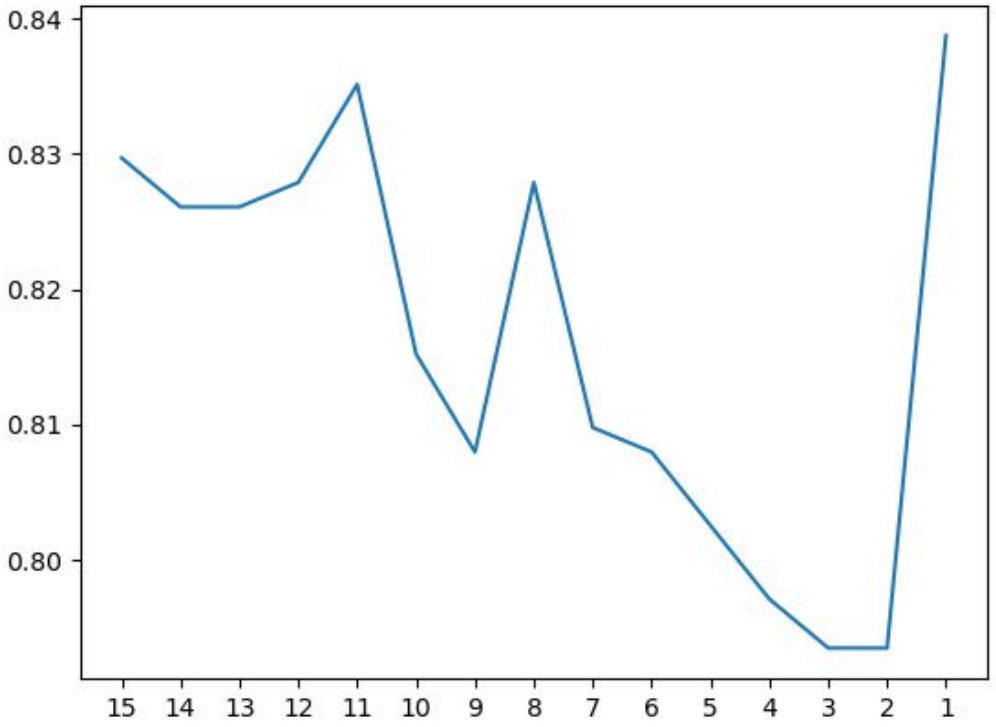

Figure07 : Feature Importance Over The Model

We can test this using any model but here we have tested it for the Random forest model since it gives the best accuracy for our dataset.

- **Check accuracy and Selecting Model**

```
def printClassificationResults(X_train, X_test, y_train, y_test):

  dt_clf = DecisionTreeClassifier(random_state=41,max_leaf_nodes=3)
  score_dt = cross_val_score(dt_clf, X_train, y_train, cv=3)
  dt_clf.fit(X_train, y_train)

  RF_clf = RandomForestClassifier(n_estimators=100, max_depth=2,random_state=42)
  score_RF = cross_val_score(RF_clf, X_train, y_train, cv=3)
  RF_clf.fit(X_train, y_train)


  y_pred_dt = dt_clf.predict(X_test)
  y_pred_RF = RF_clf.predict(X_test)


  # comparing actual response values (y_test) with predicted response values (y_pred)
  print("\t\t\t\t\t\t\t Testing\t Training")
  print("Decision Tree model accuracy(in %) \t\t\t:", round(metrics.accuracy_score(np.int64(y_test.values), y_pred_dt)*100,4) ,"\t", round(score_dt.mean()*100,2))
  print("Random forest model accuracy(in %) \t \t\t:" , round(metrics.accuracy_score(np.int64(y_test.values), y_pred_RF)*100,4) ,"\t", round(score_RF.mean()*100,2))
```

Figure08 : Code part for print accuracies of the model

We have tested some models, DecisionTreeClassifier and RandomForestClassifier to select the best model which gives the best accuracy to our model.
Here, we used a 30% of the preprocessed dataset as the testing data and others are training data. To get the accuracy score, we have used the “cross_val_score” function.

```
                                                    Testing         Training
Decision Tree model accuracy(in %)                : 83.5443         84.97
Random forest model accuracy(in %)                : 83.5443         86.61
```

Figure09 : Calculated accuracies of the models

This is the received accuracies by the two models. Since the Random Forest Model gives high accuracy in training and testing, it is selected as our model to get predictions.

- **Preprocessing the testing dataset**

All the steps in the preprocessing part we have done for the training set previously is used for this dataset as well.
Apart from them, for the rows that include missing values, we have applied some other techniques to handle them.

For each categorical data column we have chosen the most frequent value and those values are replaced with the missing value in the relevant columns.

```
#find frq of values in col for categorical data

most_frq_val = []
test_col_list_obj = ['A1', 'A3', 'A4', 'A6', 'A9', 'A15']

for col_name in test_col_list_obj:
  most_frq_val.append(test_dataset[col_name].mode()[0])
```

Figure10 : Code part for find most frequent values for categorical data

```
#categorical data replace with most frq val

i=0
for col_name in test_col_list_obj:
  test_dataset_copy[col_name].replace(['?'], [most_frq_val[i]], inplace=True)
  i=i+1
```

Figure11 : Code part for replacing missing values with frequent values - for categorical data

Missing values in the numerical columns are replaced with the mean value of that column.

```
#numaric val replace with mean of col

##################test_col_list_obj_num = ['A2', 'A5', 'A7', 'A10', 'A12', 'A14']
test_col_list_obj_num = ['A2', 'A5', 'A7','A8', 'A10','A11', 'A12','A13', 'A14']
for col_name in test_col_list_obj_num:
  test_dataset_copy[col_name] = test_dataset_copy[col_name].astype(object)
  test_dataset_copy[col_name].replace(['?'], [test_dataset[col_name].mean()], inplace=True)
  test_dataset_copy[col_name] = test_dataset_copy[col_name].astype(float)
```

Figure12 : Code part for replacing missing values with mean value - for numerical data

- **Make predictions**

After processing the testing set, the predictions are taken using the Random Forest Classifier.
Here for the training set we have selected all the columns for training as it gives a better accuracy.

```
X_train = X[selected_col_names]
X_test = test_dataset[selected_col_names]
Y_train = y


model = RandomForestClassifier(n_estimators=100, max_depth=2,random_state=42)
model.fit(X_train, Y_train)
predictions = model.predict(X_test)


print(predictions)
```

Figure13 : Code part for finding predictions

Results:

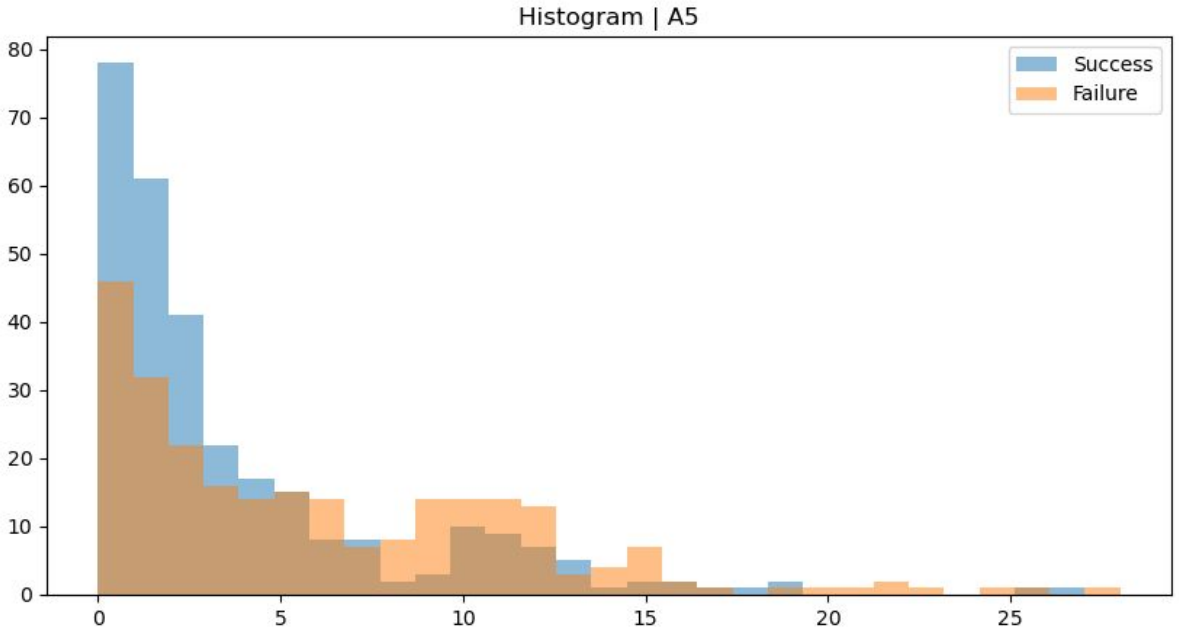

Figure14 : Histogram for Success and failures of the A5 attribute.

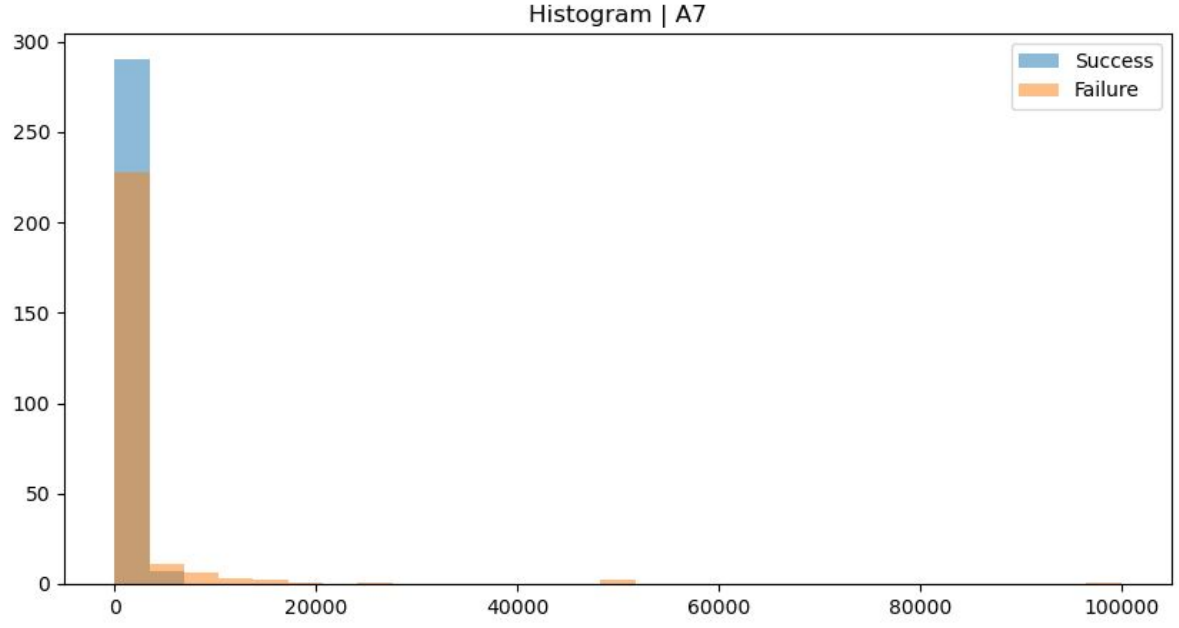

Figure15 : Histogram for Success and failures of the A7 attribute.

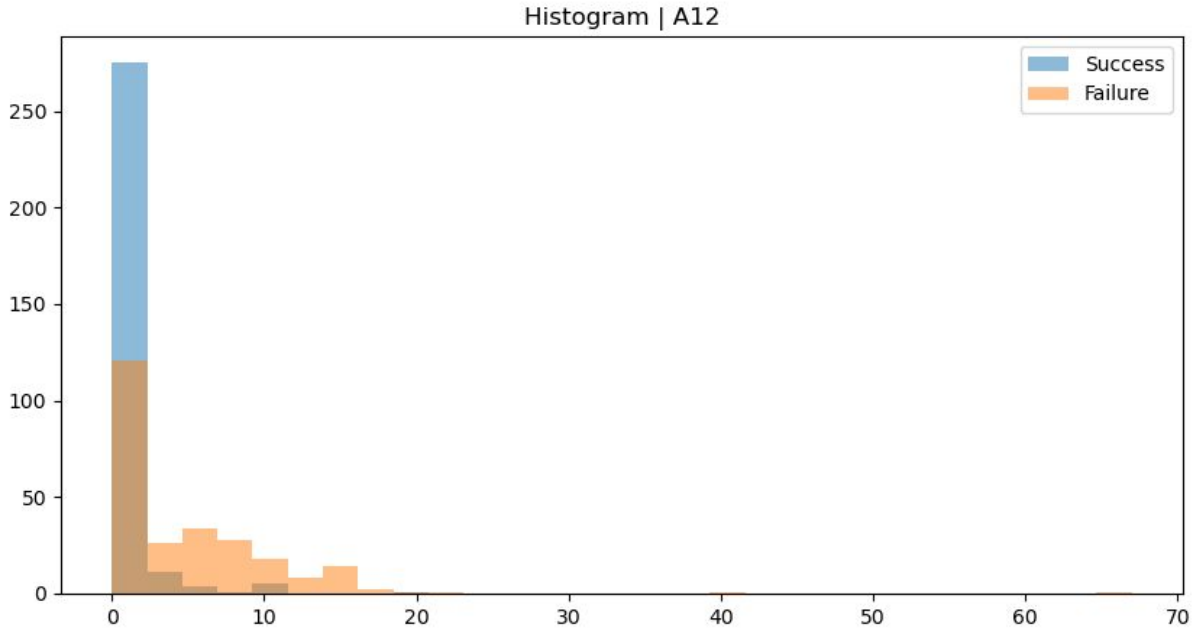

Figure16 : Histogram for Success and failures of the A12 attribute.

Here are the accuracy values given for the tested models by Decision Tree method and Random Forest method.

```
                                                Testing         Training
Decision Tree model accuracy(in %)            : 83.5443         84.97
Random forest model accuracy(in %)            : 83.5443         86.61
```

Figure17 : Calculated accuracies of the models

Since we have used the random forest model, we have expected 83% of testing accuracy and 86% of training accuracy after training the model.

```
def printClassificationResults(X_train, X_test, y_train, y_test):

  dt_clf = DecisionTreeClassifier(random_state=41,max_leaf_nodes=3)
  score_dt = cross_val_score(dt_clf, X_train, y_train, cv=3)
  dt_clf.fit(X_train, y_train)

  RF_clf = RandomForestClassifier(n_estimators=100, max_depth=2,random_state=42)
  score_RF = cross_val_score(RF_clf, X_train, y_train, cv=3)
  RF_clf.fit(X_train, y_train)

  log_RG = LogisticRegression(solver='lbfgs')
  score_LR = cross_val_score(log_RG, X_train, y_train, cv=3)
  log_RG.fit(X_train, y_train)

  svc_model = SVC()
  score_svc = cross_val_score(svc_model, X_train, y_train, cv=3)
  svc_model.fit(X_train, y_train)


  nb_clf = GaussianNB()
  score_nb = cross_val_score(nb_clf, X_train, y_train, cv=3)
  nb_clf.fit(X_train, y_train)


  y_pred_dt = dt_clf.predict(X_test)
  y_pred_RF = RF_clf.predict(X_test)
  y_pred_LG = log_RG.predict(X_test)
  y_pred_svc = svc_model.predict(X_test)
  y_pred_nb = nb_clf.predict(X_test)


  # comparing actual response values (y_test) with predicted response values (y_pred)
  print("\t\t\t\t\t\t\t Testing\t Training")
  print("Decision Tree model accuracy(in %) \t\t\t:", round(metrics.accuracy_score(np.int64(y_test.values), y_pred_dt)*100,4) ,"\t", round(score_dt.mean()*100,2))
  print("Random forest model accuracy(in %) \t \t\t:" , round(metrics.accuracy_score(np.int64(y_test.values), y_pred_RF)*100,4) ,"\t", round(score_RF.mean()*100,2))
  print("Logistic regression model accuracy(in %) \t \t\t:" , round(metrics.accuracy_score(np.int64(y_test.values), y_pred_LG)*100,4) ,"\t", round(score_LR.mean()*100,
  print("SVM accuracy(in %) \t\t\t:", round(metrics.accuracy_score(np.int64(y_test.values), y_pred_svc)*100,4) ,"\t", round(score_svc.mean()*100,2))
  print("Naive Bayes model accuracy(in %) \t\t\t:", round(metrics.accuracy_score(np.int64(y_test.values), y_pred_nb)*100,4) ,"\t", round(score_nb.mean()*100,2))
```


Figure18 : Test accuracies for different models

Apart from the Decision Tree model and Random Forest model, we used the Logistic Regression model, SVC model and Naive Bayes model to check the accuracy.
From all the models, the Random Forest model gave the highest accuracy.

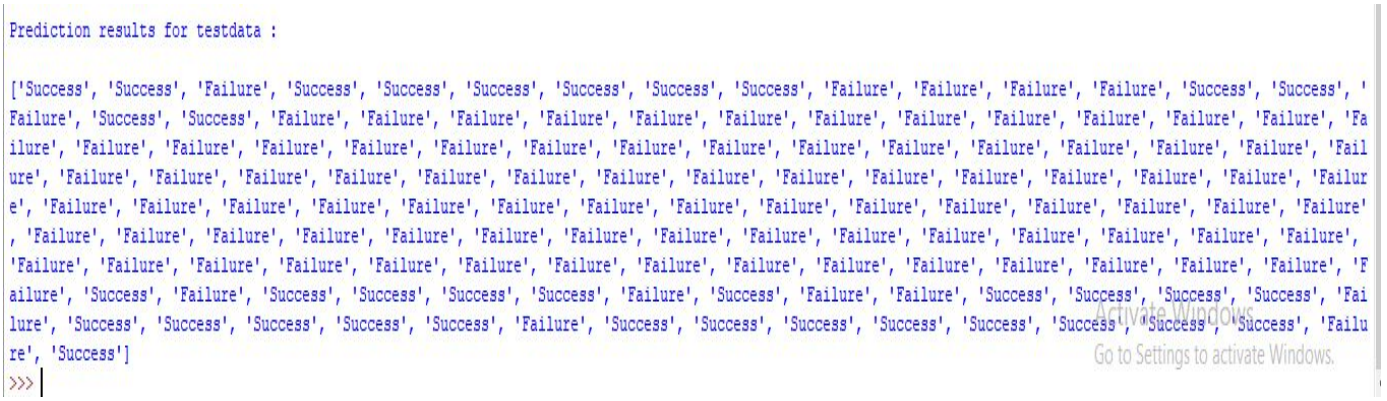

Figure19 : Resulted predictions for testdata.csv

Summary/Conclusions:

- According to the datasets given, the problem was identified as a supervised learning classification problem and we have to use Algorithms that are suitable for classification problems in the process of making predictions.
- In a nutshell, for our machine learning project first we have done the missing value handling for the training dataset by removing the rows with missing values.
- All the data types have been converted into integer/ floats in a way that we can analyse them easily.
- The boolean values and categorical data were replaced by numerical value.
- An accuracy score was calculated for each individual attribute and thereby an accuracy value was taken for each set of data that was generated by removing the least scored attribute one by one at a time.
- Modified dataset was applied to train some data models and the accuracy of predictions was tested for each model by splitting the dataset into two parts as training and testing.
- The algorithms that gives the best accuracy was chosen as our training model.

- From the results obtained from checking the accuracy of all the models (Random Forest model, Decision Tree model, Logistic Regression model, SVC model and Naive Bayes model), Random Forest model gave the highest accuracy. Therefore we used the Random Forest model to test and train the data.
- Thereby, the Random Forest algorithm was selected as the best algorithm.

- For the testing dataset, the missing values in numerical fields was replaced with the mean of the particular column. The missing values in categorical fields was replaced with the most frequent values of that particular column.
- After processing the testing dataset, the predictions were taken and around 85% of accuracy can be expected on those predictions according to previously calculated accuracy values.