# 1.0 Introduction

The sole aim of this exercise is to model the risk accrued by financial firms. This is based on the dataset provided in the interview.

The data frame is loaded with the Pandas python library as a data frame. It has 27 features; 26 out of it are dependent variables. The primary key in the dataset is the discretized Risk feature.

Table 1 shows a summary of the statistics of the datasets. There are 776 data samples, and RISK has a mean of 0.33 and a standard deviation(std) of 0.488.

The dataset is well -structured and clean. There is no need for the cleaning stage of the analysis, as would be in a typical data science project.

The processing done includes the is the replacement of the null value in the sum feature by the mean, removal of the location\_id feature since it serves no real purpose in the modeling or interpretation stage.

Risk as a primary/independent is also categorial and does not necessitate further processing.

# 2.0 Exploration data analysis (EDA)

Exploration data analysis is typically done to discern the properties and interrelation between variables and proffer the best course of action in terms of the methods/algorithms to apply to solve the problem.

Figure 1 shows the heatmap that depicts the correlation between the 27 variables. Posted along the diagonal are the correlations of the remaining 26 features with the primary key. The number of features is too high for any meaningful insight to be derived from creating pairs of plots from everything. Instead, a second image was created by plotting the seven (7) best-correlated variable with the Risk factor (Figure *2*).

Table 2 shows the correlation co-efficient sorted from highest to lowest. This data frame is merely created for visualization purposes in plotting the pairs and not used in the modeling and prediction stages.

A closer inspection of variables also indicates that most are highly skewed. Table 3 shows the skewness of the dataset. Positive values greater than 1 indicate positive skewness, while negative values less than -1 indicate negative skewness. An attempt is made to unskew some of the features by a logarithm transformation but it does not change the overall shape (Figure 3 and Figure 4).

# 3.0 Modeling

The next phase involved model creation, testing, and validation. The sklearn library (Python) is used heavily in this process. Since this is merely a binary classification problem. It was modeled with a simple classification algorithm (Logistic regression) and a machine learning approach (Random forest).

## 3.1 Logistic Regression and Random forest classification

Logistic regression is a statistical model that, in its basic form, uses a logistic function to model a binary dependent variable. In a binary logistic regression model, the dependent variable (Risk) has two levels (Wikipedia).

On the other hand, Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

## 3.2 Model Validation and Assessment

The confusion matrix is created to assess the model's sensitivity and accuracy. Figure 5 is the confusion matrix for the Logistic classification, while Figure *6* is the confusion matrix for the random forest classification. Both of them performed quite well, but the accuracy of the random forest is better than the Logistic regression.

Rate under the curve (ROC) analysis is implemented to validate the accuracy of the two binary classifiers and subsequently check the performance. Both algorithms did quite well with logistic regression having a ROC of 0.97 and Random forest, ROC of 1.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Features** | **Sector\_score** | **PARA\_A** | **Score\_A** | **Risk\_A** | **PARA\_B** | **Score\_B** | **Risk\_B** | **TOTAL** | **numbers** |
| count | 776 | 776 | 776 | 776 | 776 | 776 | 776 | 776 | 776 |
| mean | 20.184536 | 2.450194 | 0.35128 | 1.35102 | 10.79998 | 0.31314 | 6.3340 | 13.21848 | 5.0676546 |
| std | 24.319017 | 5.678870 | 0.174054 | 3.440446 | 50.08362 | 0.169804 | 30.0728 | 51.31282 | 0.264448 |
| min | 1.85 | 0 | 0.2 | 0 | 0 | 0.2 | 0 | 0 | 5 |
| 25% | 2.37 | 0.21 | 0.2 | 0.042 | 0 | 0.2 | 0 | 0.5375 | 5 |
| 50% | 3.89 | 0.875 | 0.2 | 0.175 | 0.405 | 0.2 | 0.081 | 1.37 | 5 |
| 75% | 55.57 | 2.48 | 0.6 | 1.488 | 4.16 | 0.4 | 1.8405 | 7.7075 | 5 |
| max | 59.85 | 85 | 0.6 | 51 | 1264.63 | 0.6 | 758.778 | 1268.91 | 9 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Features** | **History** | **Prob** | **Risk\_F** | **Score** | **Inherent\_Risk** | **CONTROL\_RISK** | **Detection\_Risk** | **Audit\_Risk** | **Risk** |
| count | 776 | 776 | 776 | 776 | 776 | 776 | 776 | 776 | 776 |
| mean | 0.10438 | 0.21675 | 0.05360 | 2.70257 | 17.680612 | 0.5726804 | 0.5 | 7.16815 | 0.393 |
| std | 0.53103 | 0.06798 | 0.30583 | 0.85892 | 54.740244 | 0.4445815 | 0 | 38.66749 | 0.488 |
| min | 0 | 0.2 | 0 | 2 | 1.4 | 0.4 | 0.5 | 0.28 | 0 |
| 25% | 0 | 0.2 | 0 | 2 | 1.5835 | 0.4 | 0.5 | 0.3167 | 0 |
| 50% | 0 | 0.2 | 0 | 2.4 | 2.214 | 0.4 | 0.5 | 0.5556 | 0 |
| 75% | 0 | 0.2 | 0 | 3.25 | 10.6635 | 0.4 | 0.5 | 3.2499 | 1 |
| max | 2.4 | 9 | 0.6 | 5.4 | 5.2 | 801.262 | 5.8 | 0.5 | 961.5 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Features** | **Score\_B.1** | **Risk\_C** | **Money\_Value** | **Score\_MV** | **Risk\_D** | **District\_Loss** | **PROB** | **RiSk\_E** |
| count | 776 | 776 | 776 | 776 | 776 | 776 | 776 | 776 |
| mean | 0.2237113 | 1.152963 | 14.137631 | 0.2909794 | 8.265434 | 2.5051546 | 0.206185 | 0.519072 |
| std | 0.0803517 | 0.537417 | 66.563533 | 0.1597452 | 39.97084 | 1.2286785 | 0.037508 | 0.2903118 |
| min | 0.2 | 1 | 0 | 0.2 | 0 | 2 | 0.2 | 0.4 |
| 25% | 0.2 | 1 | 0 | 0.2 | 0 | 2 | 0.2 | 0.4 |
| 50% | 0.2 | 1 | 0.095 | 0.2 | 0.018 | 2 | 0.2 | 0.4 |
| 75% | 0.2 | 1 | 5.63 | 0.4 | 2.235 | 2 | 0.2 | 0.4 |
| max | 0.6 | 5.4 | 935.03 | 0.6 |  | 561.018 | 6 | 0.6 |

Table 1: Table showing the statistic of all the variables in the dataset provided.

# Tables

|  |  |
| --- | --- |
| **Feature** | **Correlation co-efficient with RISK** |
| Sector\_score | -0.394130894 |
| PROB | 0.176912126 |
| Risk\_F | 0.214510784 |
| Audit\_Risk | 0.217112747 |
| History | 0.239452529 |
| Risk\_D | 0.25435453 |
| Risk\_B | 0.255286125 |
| Money\_Value | 0.256884494 |
| PARA\_B | 0.257029257 |
| TOTAL | 0.292021566 |
| Prob | 0.298639405 |
| numbers | 0.308140885 |
| Risk\_C | 0.342140482 |
| Score\_B.1 | 0.353802644 |
| Inherent\_Risk | 0.357020124 |
| PARA\_A | 0.378757707 |
| Risk\_A | 0.385066594 |
| District\_Loss | 0.403805745 |
| RiSk\_E | 0.411803498 |
| CONTROL\_RISK | 0.416473571 |
| Score\_A | 0.619725541 |
| Score\_B | 0.635768204 |
| Score\_MV | 0.688367421 |
| Score | 0.785995258 |
| Risk | 1 |
| Detection\_Risk | Nan |

Table 2: Correlation coefficient of dependent variables with RISK

|  |  |
| --- | --- |
| **Feature** | **Skewness** |
| Sector\_score | 0.769987 |
| PARA\_A | 8.505663 |
| Score\_A | 0.492813 |
| Risk\_A | 8.356859 |
| PARA\_B | 20.53829 |
| Score\_B | 0.960477 |
| Risk\_B | 20.50606 |
| TOTAL | 19.26174 |
| numbers | 6.742206 |
| Score\_B.1 | 3.55523 |
| Risk\_C | 3.995796 |
| Money\_Value | 10.53623 |
| Score\_MV | 1.29788 |
| Risk\_D | 10.52629 |
| District\_Loss | 2.231033 |
| PROB | 6.560046 |
| RiSk\_E | 3.011444 |
| History | 9.275458 |
| Prob | 4.348489 |
| Risk\_F | 10.39171 |
| Score | 1.055717 |
| Inherent\_Risk | 9.170031 |
| CONTROL\_RISK | 5.158719 |
| Detection\_Risk | 0 |
| Audit\_Risk | 20.10897 |
| Risk | 0.438822 |

Table 3: Table showing the data skewness

# 4.0 Figures

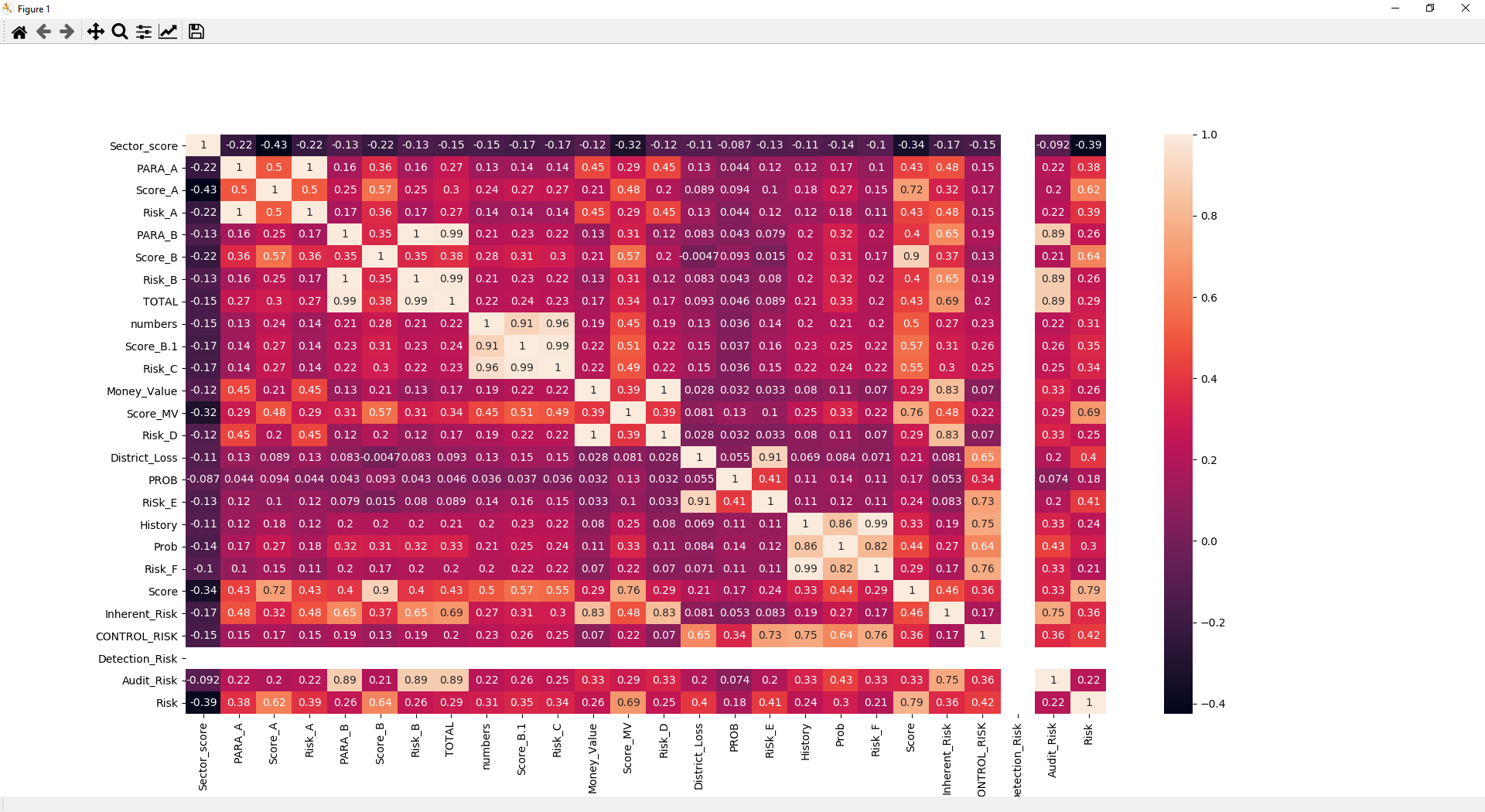


Figure 1 : Pair-plot of showing the distribution of the most correlated variable with RISK

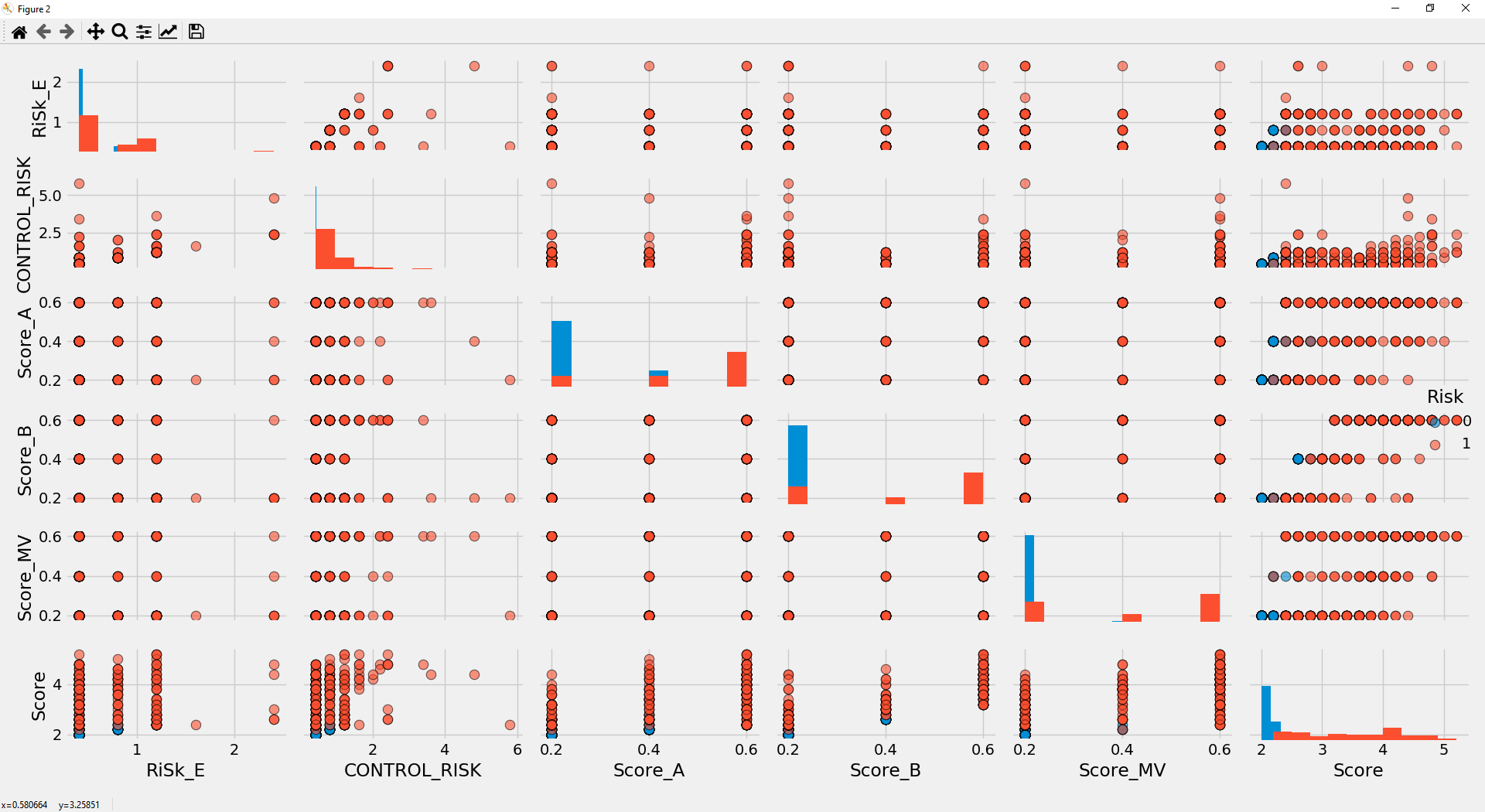


Figure 2: heatmaps showing the correlation between the 27 variables

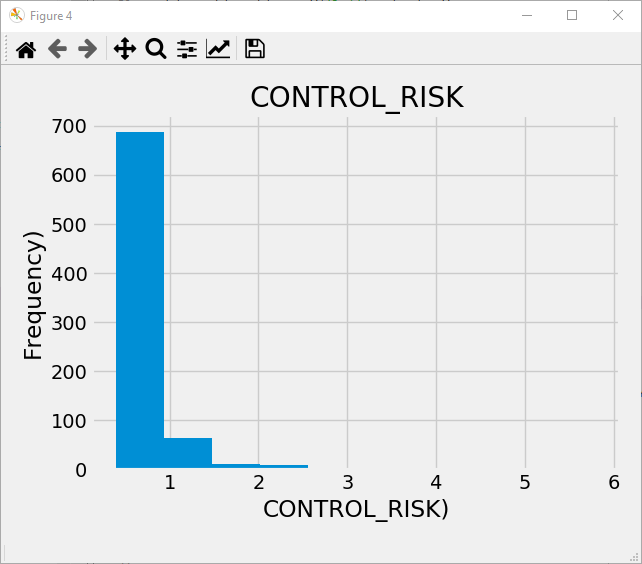


Figure 3: Histogram of pristine CONTROL \_RISK feature

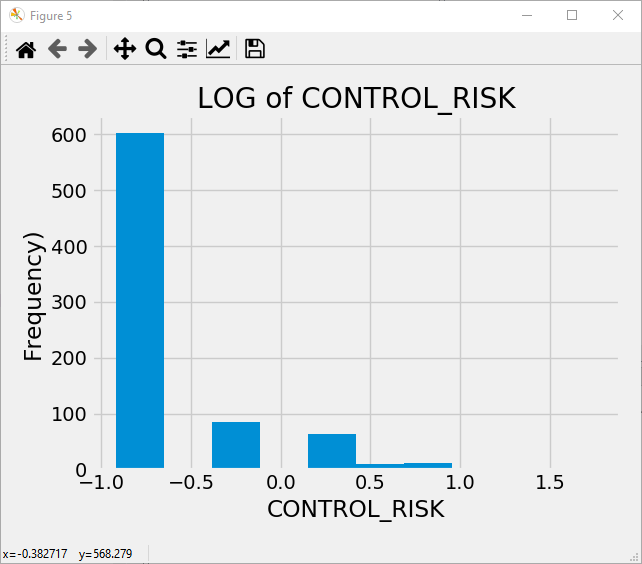


Figure 4: Histogram of CONTROL\_RISK feature after transforming to the logarithm.

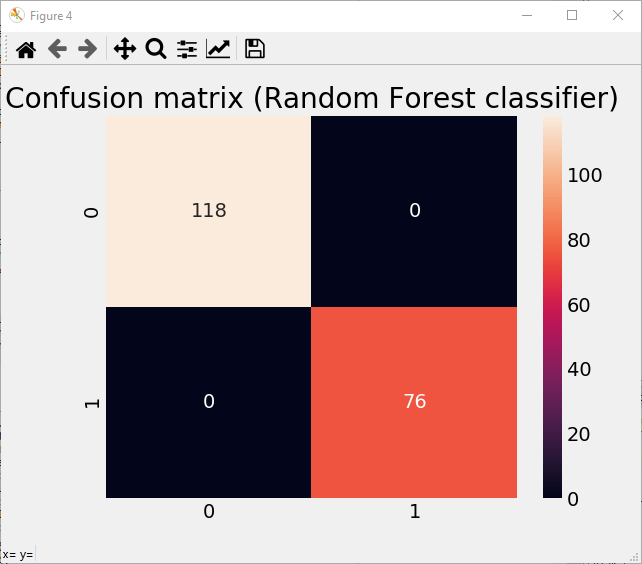


Figure 5: Confusion matrix for Random forest classifier

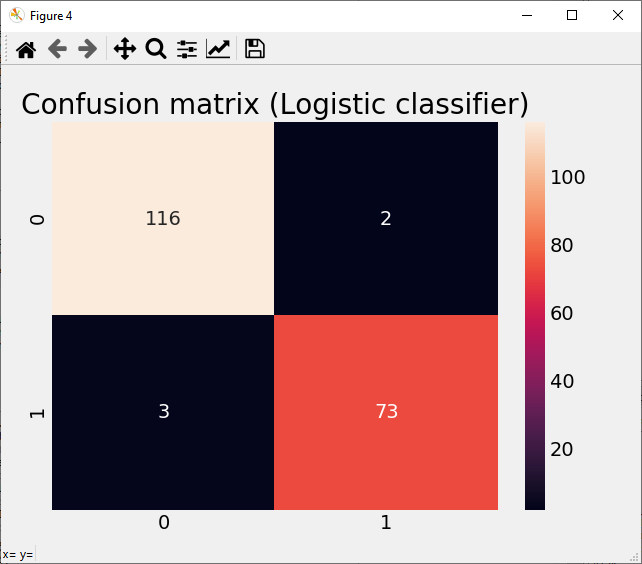


Figure 6: Confusion matrix for logistic regression classifier

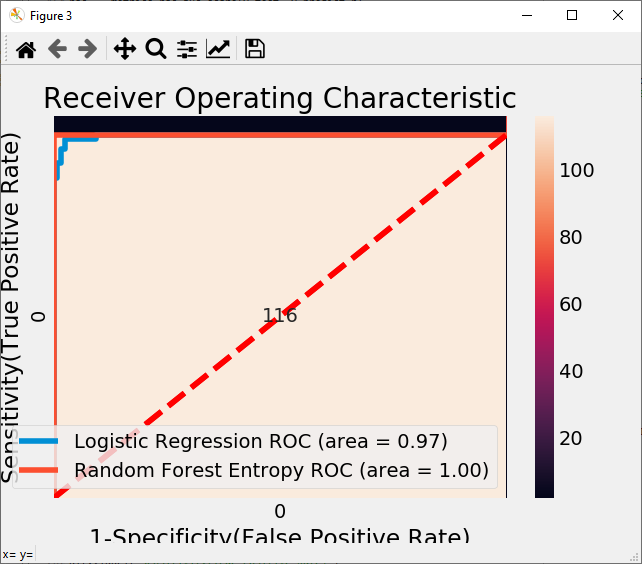


Figure 7 : AUC (Area under the curve) analysis for Logistic regression and random forest classifiers.

# 5.0 Appendix: Code

"""

@author: Dr.Ayodeji Babalola

"""

# IMPORT LIBSRARIES

#------------------------------------------------------------------

import pandas as pd

import numpy as np

import warnings

warnings.filterwarnings('ignore')

import matplotlib.pyplot as plt

from sklearn import model\_selection

from sklearn.linear\_model import LogisticRegression

import sklearn.metrics as metrics

from sklearn.ensemble import RandomForestClassifier,AdaBoostClassifier

import seaborn as sb

plt.close('all')

plotting\_flag = False

# Read the Data

#------------------------------------------------------------------

data = pd.read\_csv('audit\_risk.csv')

# Exploratory data Analysis

#------------------------------------------------------------------

data.head()

data.tail()

del\_cols =['LOCATION\_ID'] # not useful in the modeling process

data.drop(del\_cols, axis=1, inplace=True)

print(data.isna().sum()) # checking for null values

data['Money\_Value'].fillna((data['Money\_Value'].mean()), inplace=True) # test median

correlations\_data = data.corr()['Risk'].sort\_values()

df = data[correlations\_data[18:-1].keys()]

if (plotting\_flag == True):

sb.pairplot(df, hue = 'Risk', diag\_kind = 'hist',

plot\_kws = {'alpha': 0.6, 's': 80, 'edgecolor': 'k'},

size = 4)

corr = data.corr()

if (plotting\_flag == True):

sb.heatmap(corr,

xticklabels=corr.columns.values,

yticklabels=corr.columns.values,annot=True)

X=data.drop(['Risk'],axis=1)

if (plotting\_flag == True):

plt.figure()

plt.hist(data['CONTROL\_RISK'])

plt.xlabel('CONTROL\_RISK)')

plt.ylabel('Frequency)')

plt.title('CONTROL\_RISK')

plt.figure()

plt.hist(np.log(data['CONTROL\_RISK']))

plt.xlabel('CONTROL\_RISK')

plt.ylabel('Frequency)')

plt.title('LOG of CONTROL\_RISK')

data\_skew = data.skew()

Y=data['Risk']

# Train Test Split

#------------------------------------------------------------------

from sklearn.model\_selection import train\_test\_split,cross\_val\_score

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size = 0.25,stratify=Y, random\_state = 50)

# Feature Scaling

from sklearn.preprocessing import StandardScaler

sc\_X = StandardScaler()

X\_train\_scaled = pd.DataFrame(sc\_X.fit\_transform(X\_train))

X\_test\_scaled = pd.DataFrame(sc\_X.transform(X\_test))

# Applying Base Model : Logistic Regression

#------------------------------------------------------------------

model\_logistic = LogisticRegression();

model\_logistic.fit(X\_train\_scaled, y\_train)

# Cross Validation : Logistic Regression

#------------------------------------------------------------------

kfold = model\_selection.KFold(n\_splits=10, random\_state=7)

scoring = 'accuracy'

acc\_logi = cross\_val\_score(estimator = model\_logistic, X = X\_train\_scaled, y = y\_train, cv = kfold,scoring=scoring)

acc\_logi.mean()

# Model Evaluation : Logistic Regression

#------------------------------------------------------------------

y\_predict\_logi = model\_logistic.predict(X\_test\_scaled)

acc = metrics.accuracy\_score(y\_test, y\_predict\_logi)

roc = metrics.roc\_auc\_score(y\_test, y\_predict\_logi)

prec = metrics.precision\_score(y\_test, y\_predict\_logi)

rec = metrics.recall\_score(y\_test, y\_predict\_logi)

f1 = metrics.f1\_score(y\_test, y\_predict\_logi)

model\_eval\_logi\_regress = pd.DataFrame([['Logistic Regression',acc, acc\_logi.mean(),prec,rec, f1,roc]],

columns = ['Model', 'Accuracy','Cross Val Accuracy', 'Precision', 'Recall', 'F1 Score','ROC'])

# Applying Random Forest

#------------------------------------------------------------------

model\_random\_forest = RandomForestClassifier(n\_estimators = 100, oob\_score = True,criterion='entropy', random\_state = 45)

model\_random\_forest.fit(X\_train\_scaled, y\_train) # train the model

print('Score: ', model\_random\_forest.score(X\_train, y\_train))

# Model Evaluation : Random Forest

#------------------------------------------------------------------

acc\_rande = cross\_val\_score(estimator = model\_random\_forest, X = X\_train\_scaled, y = y\_train, cv = kfold, scoring=scoring)

acc\_rande.mean()

y\_predict\_r = model\_random\_forest.predict(X\_test\_scaled)

roc = metrics.roc\_auc\_score(y\_test, y\_predict\_r)

acc = metrics.accuracy\_score(y\_test, y\_predict\_r)

prec = metrics.precision\_score(y\_test, y\_predict\_r)

rec = metrics.recall\_score(y\_test, y\_predict\_r)

f1 = metrics.f1\_score(y\_test, y\_predict\_r)

model\_eval\_random\_forest = pd.DataFrame([['Random Forest',acc, acc\_rande.mean(),prec,rec, f1,roc]],

columns = ['Model', 'Accuracy','Cross Val Accuracy', 'Precision', 'Recall', 'F1 Score','ROC'])

model\_evals = model\_eval\_logi\_regress.append(model\_eval\_random\_forest, ignore\_index = True)

# CHOOSING THE BEST CLASSIFIER

#------------------------------------------------------------------

# Confusion Matrix

if (plotting\_flag == True):

plt.figure()

cm\_logi = metrics.confusion\_matrix(y\_test, y\_predict\_logi)

plt.title('Confusion matrix (Logistic classifier)')

sb.heatmap(cm\_logi,annot=True,fmt="d")

plt.show()

plt.figure()

cm\_r = metrics.confusion\_matrix(y\_test, y\_predict\_r)

plt.title('Confusion matrix (Random Forest classifier)')

sb.heatmap(cm\_r,annot=True,fmt="d")

plt.show()

# ROC Curve : 'Logistic Regression'

#------------------------------------------------------------------

model\_random\_forest.fit(X\_train\_scaled, y\_train) # train the model

y\_pred= model\_logistic.predict(X\_test\_scaled) # predict the test data

# Compute False postive rate, and True positive rate

fpr, tpr, thresholds = metrics.roc\_curve(y\_test, model\_logistic.predict\_proba(X\_test\_scaled)[:,1])

# Calculate Area under the curve to display on the plot

auc = metrics.roc\_auc\_score(y\_test,model\_logistic.predict(X\_test\_scaled))

if (plotting\_flag == True):

# Now, plot the computed values

plt.plot(fpr, tpr, label='%s ROC (area = %0.2f)' % ('Logistic Regression', auc))

# Custom settings for the plot

plt.plot([0, 1], [0, 1],'r--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('1-Specificity(False Positive Rate)')

plt.ylabel('Sensitivity(True Positive Rate)')

plt.title('Receiver Operating Characteristic')

plt.legend(loc="lower right")

plt.show()

model\_random\_forest.fit(X\_train\_scaled, y\_train) # train the model

y\_pred= model\_random\_forest.predict(X\_test\_scaled) # predict the test data

# Compute False postive rate, and True positive rate

fpr, tpr, thresholds = metrics.roc\_curve(y\_test, model\_random\_forest.predict\_proba(X\_test\_scaled)[:,1])

# Calculate Area under the curve to display on the plot

auc = metrics.roc\_auc\_score(y\_test,model\_random\_forest.predict(X\_test\_scaled))

# Now, plot the computed values

if (plotting\_flag == True):

plt.plot(fpr, tpr, label='%s ROC (area = %0.2f)' % ('Random Forest Entropy', auc))

# Custom settings for the plot

plt.plot([0, 1], [0, 1],'r--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('1-Specificity(False Positive Rate)')

plt.ylabel('Sensitivity(True Positive Rate)')

plt.title('Receiver Operating Characteristic')

plt.legend(loc="lower right")

plt.show()