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.

Tables

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.

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

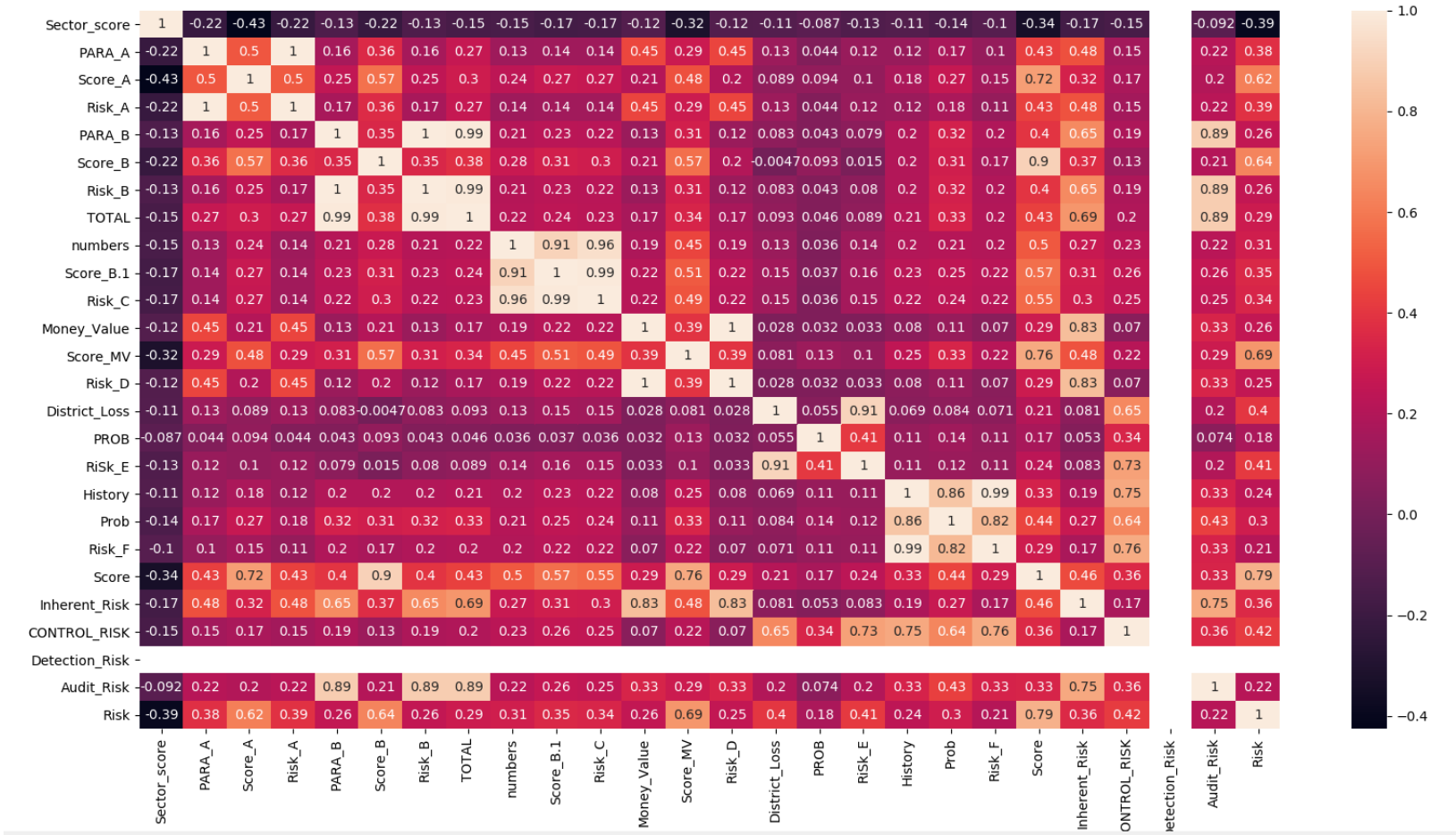

Figure 1 : Pair-plot of showing the distribution of the most correlated variable with RISK

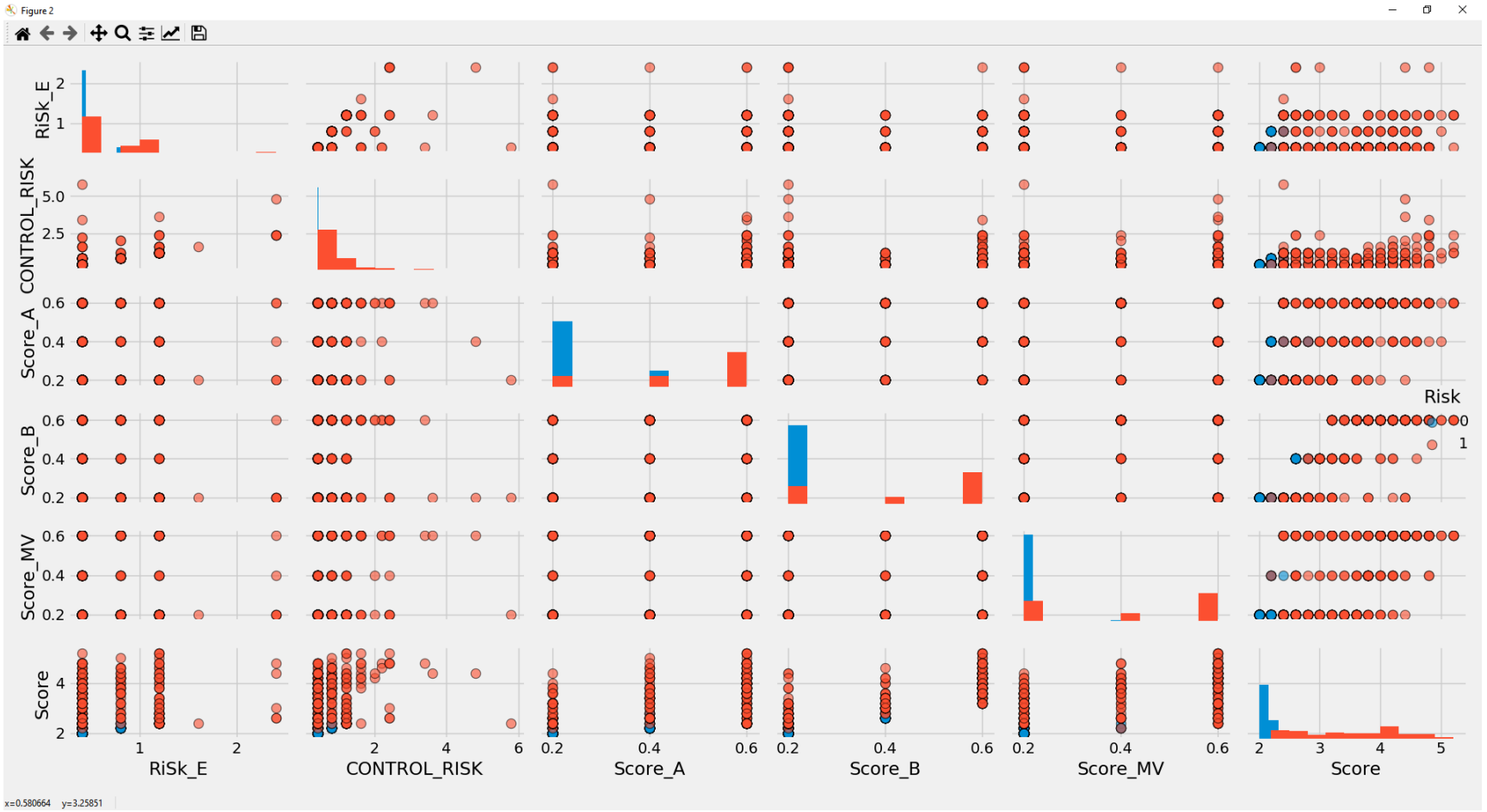

Figure 2: heatmaps showing the correlation between the 27 variables

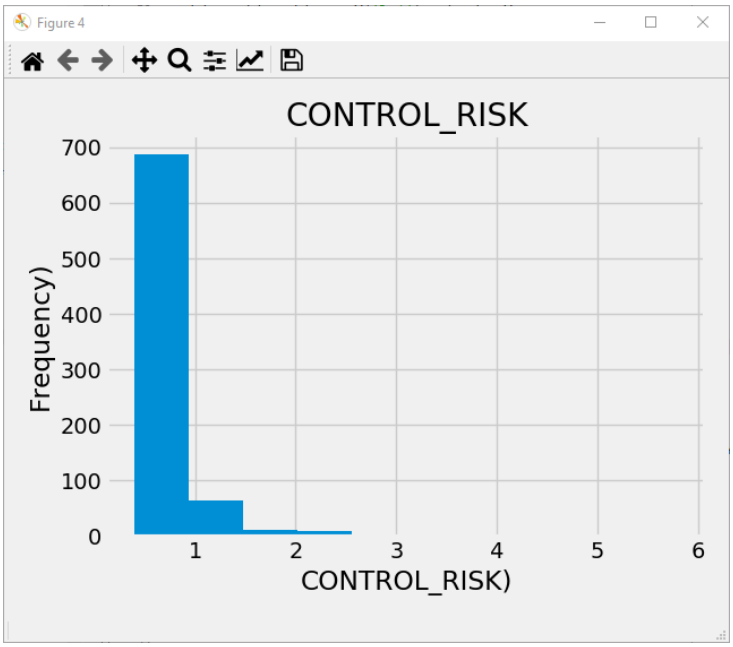

Figure 3: Histogram of pristine CONTROL _RISK feature

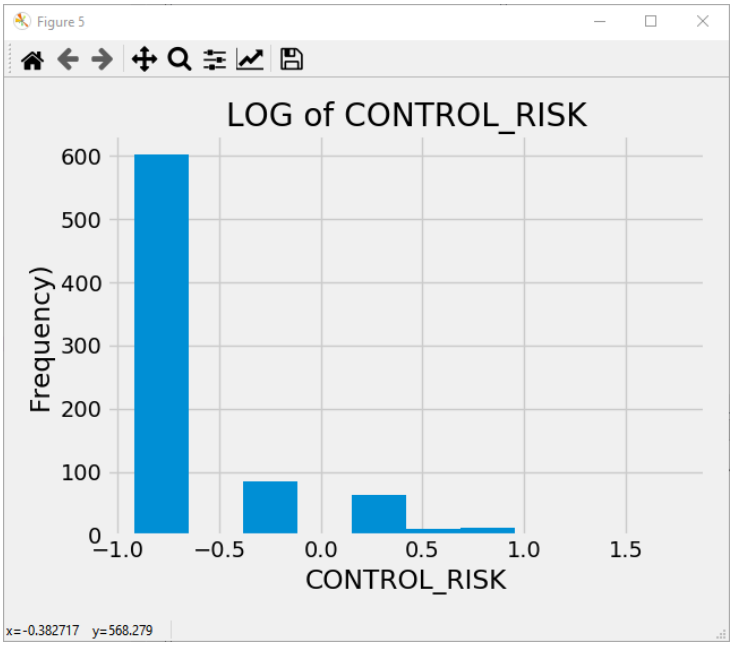

Figure 4: Histogram of CONTROL_RISK feature after transforming to the logarithm.

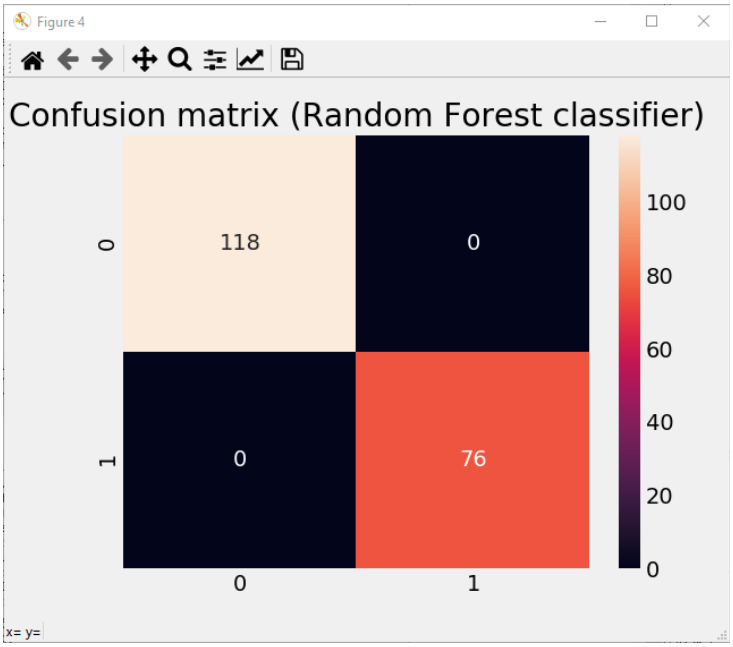

Figure 5: Confusion matrix for Random forest classifier

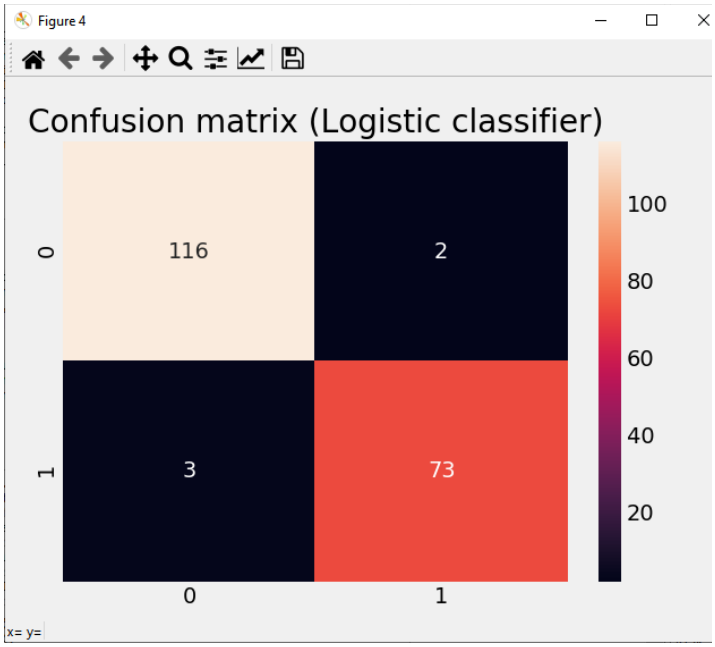

Figure 6: Confusion matrix for logistic regression classifier

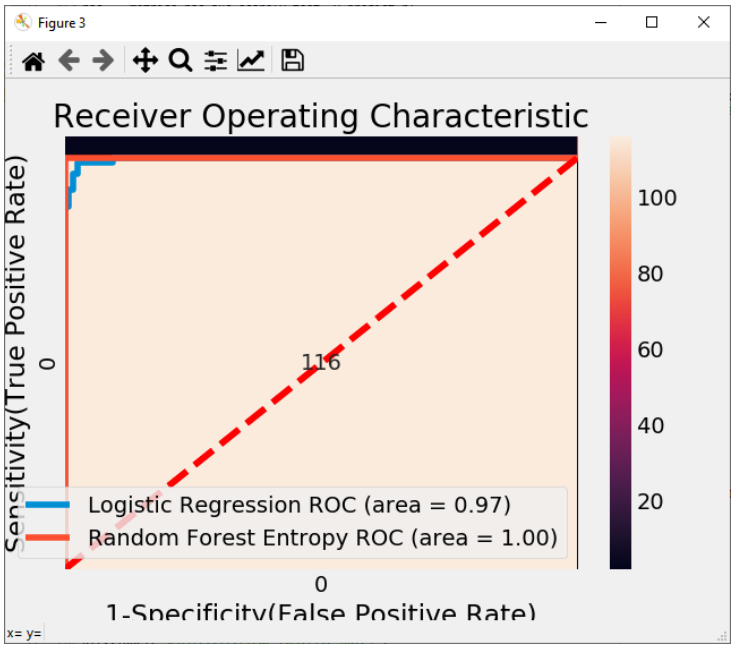

Figure 7 : AUC (Area under the curve) analysis for Logistic regression and random forest classifiers.

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()
```