UTS: ENGINEERING & INFORMATION TECHNOLOGY

SUBJECT NUMBER & NAME	**NAME OF STUDENT(s)** (PRINT CLEARLY)	**STUDENT ID(s)**
32130 Fundamentals of Data Analytics - Spring 2023	Aishwarya Rajesh Panhale	24587976

STUDENT EMAIL	**STUDENT CONTACT NUMBER**
aishwaryarajesh.panhale@student.uts.edu.au	0466394417

ASSESSMENT ITEM NUMBER & TITLE

Assignment 3
Data analytics in action

☐ I confirm that I have read, understood and followed the guidelines for assignment submission and presentation on page 2 of this cover sheet.
☐ I confirm that I have read, understood and followed the advice in the Subject Outline about assessment requirements.
☐ I understand that if this assignment is submitted after the due date it may incur a penalty for lateness unless I have previously had an extension of time approved and have attached the written confirmation of this extension.

Declaration of originality: The work contained in this assignment, other than that specifically attributed to another source, is that of the author(s) and has not been previously submitted for assessment. I understand that, should this declaration be found to be false, disciplinary action could be taken and penalties imposed in accordance with University policy and rules. In the statement below, I have indicated the extent to which I have collaborated with others, whom I have named.

Statement of collaboration:

Signature of student(s)_____ARP_____ Date _____10/11/2023_____

1. Problem Description

We have Two datasets provided, loan_data_training with 42453rows 72 columns and loan_data_unknown with 4717 rows 71 columns. The loan_data_training dataset includes TARGET column that has binary value (0 or 1) which indicated the payment difficulties of clients. The binary value 0 states that client faces difficulties in loan payment with more than X days on first Y instalments and 1 states that other cases.

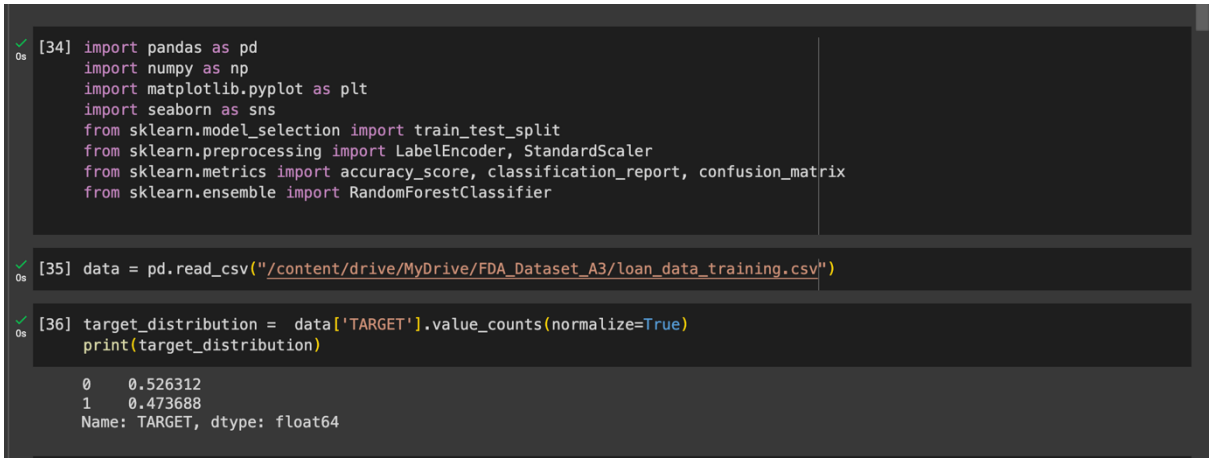
The task includes prediction of TARGET values to identify the client's loan application which results whether client should be given loan or not. The loan_data_unknown dataset which does not has TARGET column needs to be predicted using loan_data_training trained model and provide with TARGET column values. Determining the best model to predict TARGET values is based on the accuracy of the model during training the model.

2. Data Pre-processing

Data was pre-processed to have better accuracy when executing the classifier. The data pre-processing was carried out in python through google Colaboratory .

Step1:

Importing all the necessary libraries and importing the data.
I have also printed the distribution of TARGET column (0,1) count values .



```
[34] import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
     from sklearn.model_selection import train_test_split
     from sklearn.preprocessing import LabelEncoder, StandardScaler
     from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
     from sklearn.ensemble import RandomForestClassifier

[35] data = pd.read_csv("/content/drive/MyDrive/FDA_Dataset_A3/loan_data_training.csv")

[36] target_distribution =  data['TARGET'].value_counts(normalize=True)
     print(target_distribution)

     0    0.526312
     1    0.473688
     Name: TARGET, dtype: float64
```

Step2:

Loan_data_training has many missing values. Using this code, I first identified missing values in dataset to later clean the dataset to get better accuracy.

```
#identifying missing values
missing_values = data.isna().sum()
print("Missing values by column:")
print(missing_values[missing_values > 0])
```

```
Missing values by column:
AMT_ANNUITY                    1
AMT_GOODS_PRICE               48
NAME_TYPE_SUITE              159
CNT_FAM_MEMBERS                1
EXT_SOURCE_2                  91
EXT_SOURCE_3                8966
OBS_30_CNT_SOCIAL_CIRCLE     105
DEF_30_CNT_SOCIAL_CIRCLE     105
OBS_60_CNT_SOCIAL_CIRCLE     105
DEF_60_CNT_SOCIAL_CIRCLE     105
AMT_REQ_CREDIT_BUREAU_HOUR  6431
AMT_REQ_CREDIT_BUREAU_DAY   6431
AMT_REQ_CREDIT_BUREAU_WEEK  6431
AMT_REQ_CREDIT_BUREAU_MON   6431
AMT_REQ_CREDIT_BUREAU_QRT   6431
AMT_REQ_CREDIT_BUREAU_YEAR  6431
dtype: int64
```

Step3:

After the count of missing values, checking the histogram plot to identify the outliers. Verifying the columns as shown in the code screenshot to check if they have any impact on training the classifier. With the help of histogram, DAYS_EMPLOYED has few positive values and FLAG_DOCUMENT_* were multiple 0 values
they were identified.

```
data.hist(figsize=(30,30))
plt.show()
```

Step4:

For data cleaning and verifying the correlation of columns with TARGET column heatmap was created.

Step5:

Dropping columns that have no co-relation with TARGET column and will not affect the classifier training and accuracy;

Data.drop() function is used to drop multiple columns . Dataset includes numeric, float and string values. To differentiate datatypes of columns and resolving the null values to ensure better accuracy, for data pre-processing, I have used data.select_dtypes to separate the float64 , int64 values and object datatype values with the help of numerical_cols and categorical_cols. For categorical_cols replacing the null values with unique function- nunique() and for numerical_cols replacing null values with median calculation for every numeric column. Using fillna() function to fill the median value calculated as per the numeric column is

```
[ ] #data drop based on corelation through heatmap and histogram visuals
    data= data.drop(['EXT_SOURCE_3','FLAG_MOBIL'],axis=1)
    #multiple columns of FLAG_DOCUMENT and AMT_REQ_CREDIT_BUREAU_*
    data= data.drop([ 'FLAG_DOCUMENT_2','FLAG_DOCUMENT_4', 'FLAG_DOCUMENT_5', 'FLAG_DOCUMENT_6','FLAG_DOCUMENT_7',
                'FLAG_DOCUMENT_8', 'FLAG_DOCUMENT_9','FLAG_DOCUMENT_10', 'FLAG_DOCUMENT_11', 'FLAG_DOCUMENT_12','FLAG_DOCUMEN
                'FLAG_DOCUMENT_14', 'FLAG_DOCUMENT_15','FLAG_DOCUMENT_16', 'FLAG_DOCUMENT_17', 'FLAG_DOCUMENT_18','FLAG_DOCUM

[ ] #based on histogram diagram removing employee values that are positive
    #data = data[data['DAYS_EMPLOYED'] <= 0]

    #data['DAYS_EMPLOYED'].describe()

[ ] # Identifying datatypes of column as  numerical and categorical columns
    numerical_cols = data.select_dtypes(include=['float64', 'int64']).columns
    categorical_cols = data.select_dtypes(include=['object']).columns

    categorical_distribution = data[categorical_cols].nunique()
    #median for numeric values
    median_values = data[numerical_cols].median()

    # using median  for missing value
    data[numerical_cols] = data[numerical_cols].fillna(median_values)

[ ] #checking mising values
    data.isnull().sum()
```

Step6:

Transforming the categorical columns into numeric format for allowing machine learning classifiers to train and work on data. For this case, I preferred using label encoding because one-hot encoding after execution was affecting data and also it is memory efficient and optimizing the code. Label encoder is imported using “from sklearn.preprocessing import LabelEncoder” library.

```
label_encoder = LabelEncoder()
data['NAME_CONTRACT_TYPE'] = label_encoder.fit_transform(data['NAME_CONTRACT_TYPE'])
data['CODE_GENDER'] = label_encoder.fit_transform(data['CODE_GENDER'])
data['FLAG_OWN_CAR'] = label_encoder.fit_transform(data['FLAG_OWN_CAR'])
data['FLAG_OWN_REALTY'] = label_encoder.fit_transform(data['FLAG_OWN_REALTY'])
data['NAME_TYPE_SUITE'] = label_encoder.fit_transform(data['NAME_TYPE_SUITE'])
data['NAME_INCOME_TYPE'] = label_encoder.fit_transform(data['NAME_INCOME_TYPE'])
data['NAME_EDUCATION_TYPE'] = label_encoder.fit_transform(data['NAME_EDUCATION_TYPE'])
data['NAME_FAMILY_STATUS'] = label_encoder.fit_transform(data['NAME_FAMILY_STATUS'])
data['NAME_HOUSING_TYPE'] = label_encoder.fit_transform(data['NAME_HOUSING_TYPE'])
data['WEEKDAY_APPR_PROCESS_START'] = label_encoder.fit_transform(data['WEEKDAY_APPR_PROCESS_START'])
data['ORGANIZATION_TYPE'] = label_encoder.fit_transform(data['ORGANIZATION_TYPE'])

<ipython-input-47-78ca0b0e0c8c>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-v
  data['NAME_CONTRACT_TYPE'] = label_encoder.fit_transform(data['NAME_CONTRACT_TYPE'])
```

Step7:

To have balanced dataset for training using machine learning classifiers. Resample library is imported. In this case, upsampling is done for model to be unbiased with majority class and allowing minority class to be trained equally to provide better and balanced result. The TARGET value for 0-18066 and 1-17627 is of less difference so upscaling would be better option, to avoid any loss of data with down sampling.

```
from sklearn.utils import resample


print("BEFORE" , data["TARGET"].value_counts())

sam_data = data.copy()
df_major = sam_data [ sam_data["TARGET"] == 0]
df_minor = sam_data [ sam_data["TARGET"] == 1]

df_minor_upsampling = resample(df_minor, replace = True,
                               n_samples = len(df_major), random_state =23)

data = pd.concat([df_major, df_minor_upsampling])

print(data["TARGET"].value_counts())
```

```
BEFORE 0    18066
1    17627
Name: TARGET, dtype: int64
0    18066
1    18066
Name: TARGET, dtype: int64
```

Step8 :
I used StandardScaler library from scikit-learn to standardize the numerical columns dataset. This process is used in many machine learning algorithms, it ensures that all features are on a similar scale, which improves the algorithm. Dataset is split into training and testing module using train_test_split() function, it is used to train and test the model respectively and also test the performance on unseen data.
After splitting the data, the training and testing sets are standardized separately to prevent any data leakage

```
from sklearn.preprocessing import  StandardScaler
from sklearn.model_selection import train_test_split
scaler = StandardScaler()

X = data.drop(['TARGET', 'SK_ID_CURR'], axis=1)
y = data['TARGET']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=42)
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

3. Solving the problem

After data pre-processing and data split into testing and training module.the next step is to solve the problem by implementing different classifier to identify their accuracy, confusion matrix and ROC curve and select the best classifier for the dataset to predict TARGET values for unknown dataset.
I have implemented five classification

1. Random Forect Classifier
2. Decision Tres Classifier

3. SVM Model
4. K-Nearest Neigbour Classifier
5. Neural Networks- MLP classifier

Each classifier is trained on the training data, accuracy is calculated, and predictions were made on the test set. Performance of each is classifier is calculated using accuracy and F1 score metrics. Classification report is generated for all classifier along with ROC curve and Area Under Curve is shown.

- Random Forect Classifier

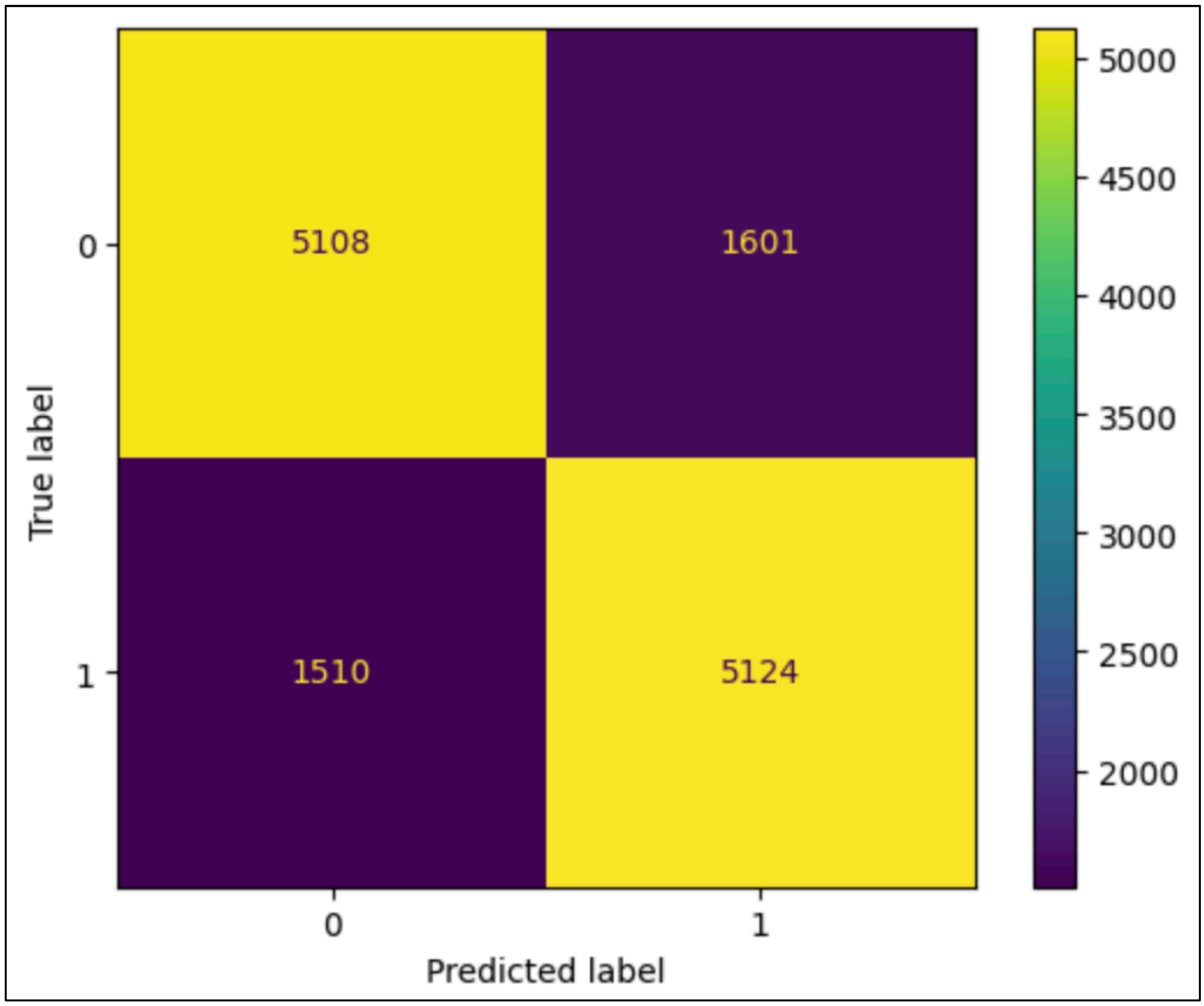
For executing random forest classifier essential libraries are imported such as "from sklearn.ensemble import RandomForestClassifier" and "from sklearn.metrics import accuracy_score". rf_model has classifier and its hyperparameters to get better accuracy which includes # max_depth=30: Maximum depth of the decision tree in each random forest.
min_samples_leaf=2 number of samples required for leaf node.
min_samples_split=10: number of samples to split an internal node.
n_estimators=300: decision trees in the random forest.
random_state=42: random seed for reproducibility.
Accuracy Score for rf_model is 0.76

```
#Random Forest Classifier

from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
#adding hyperparameters to get better accuracy than 62% which was with no parameters
rf_model = RandomForestClassifier(max_depth=30, min_samples_leaf=2, min_samples_split=10, n_estimators=300, random_state
# max_depth=30: Maximum depth of the decision tree in each random forest
# min_samples_leaf=2: Minimum number of samples required for leaf node.
# min_samples_split=10: Minimum number of samples required to split an internal node.
# n_estimators=300: The number of decision trees in the random forest.
# random_state=42: Set a random seed for reproducibility.
rf_model.fit(X_train, y_train)
y_rf_pred = rf_model.predict(X_test)
accuracy = accuracy_score(y_test, y_rf_pred)
print("Random Forest Classifier Accuracy:", accuracy)

print("Classification Report for Random Forest Classifier:\n", classification_report(y_test, y_rf_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_rf_pred))
```

```
Random Forest Classifier Accuracy: 0.7668440380723975
Classification Report for Random Forest Classifier:
              precision    recall  f1-score   support

           0       0.77      0.76      0.77      6709
           1       0.76      0.77      0.77      6634

    accuracy                           0.77     13343
   macro avg       0.77      0.77      0.77     13343
weighted avg       0.77      0.77      0.77     13343

Confusion Matrix:
 [[5108 1601]
 [1510 5124]]
```

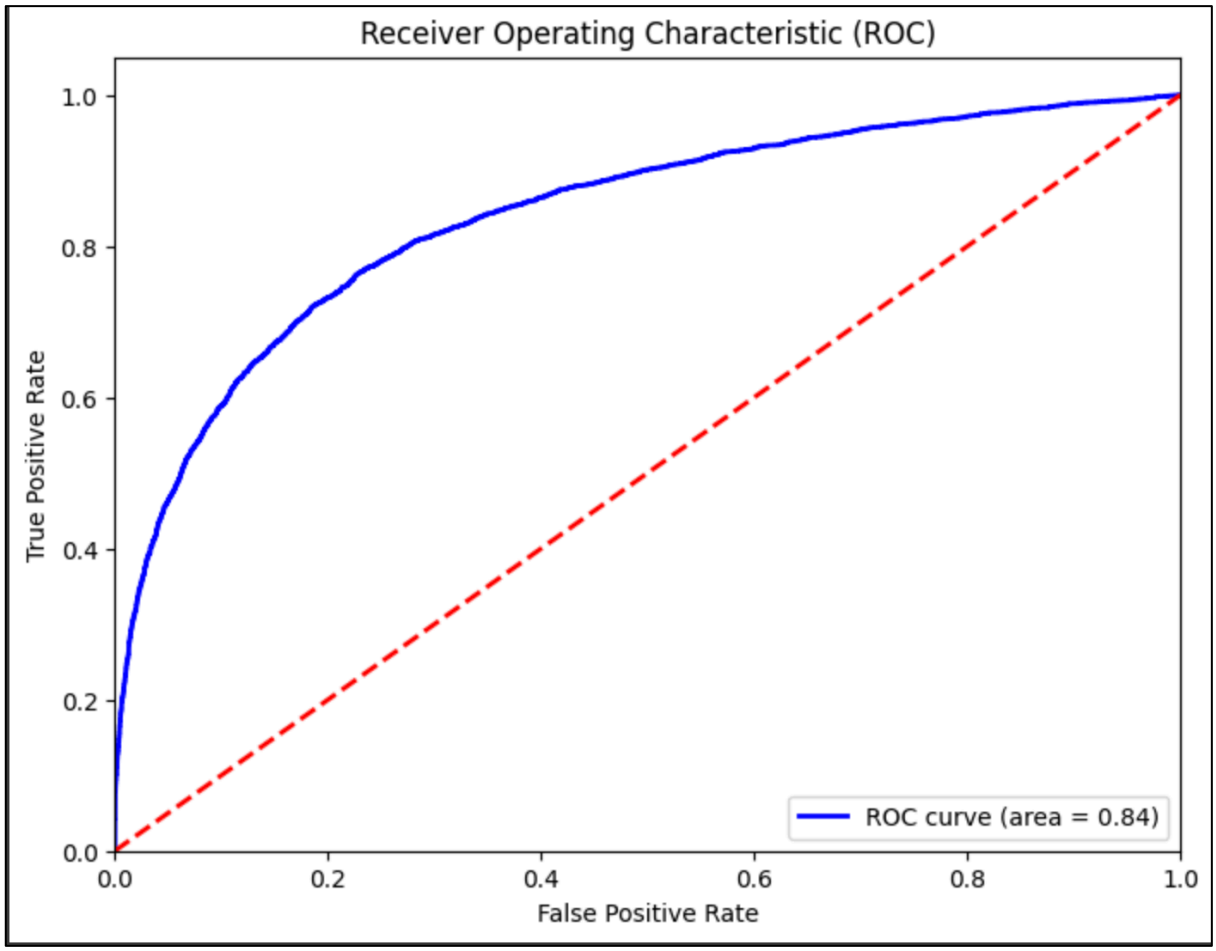
To plot Confusion matrix "from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay" is imported.

```
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.metrics import roc_curve, roc_auc_score, f1_score, auc

#defining the values of y_test and y_pred to confusion matrix
rf_confusion_matrix = confusion_matrix(y_test, y_rf_pred)

#to handle imblance in dataset we calculate f1_score
f1_rf= f1_score(y_test,y_rf_pred)

#displaying Confusion matrix
disp_rf= ConfusionMatrixDisplay(confusion_matrix=rf_confusion_matrix)
disp_rf.plot()
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()
```



To plot ROC curve “from sklearn.metrics import roc_curve, roc_auc_score” library is imported.
Area Under Curve for Random Forest is 0.84

```python
#random forest classifier ROC curve
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.ensemble import RandomForestClassifier

y_pred_prob_rf = rf_model.predict_proba(X_test)[:, 1]
#false positive and true positive rate to calculate roc curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob_rf)


roc_auc_rf = roc_auc_score(y_test, y_pred_prob_rf)


plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', lw=2, label='ROC curve (area = %0.2f)' % roc_auc_rf)
plt.plot([0, 1], [0, 1], color='red', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC)')
plt.legend(loc='lower right')
plt.show()
```



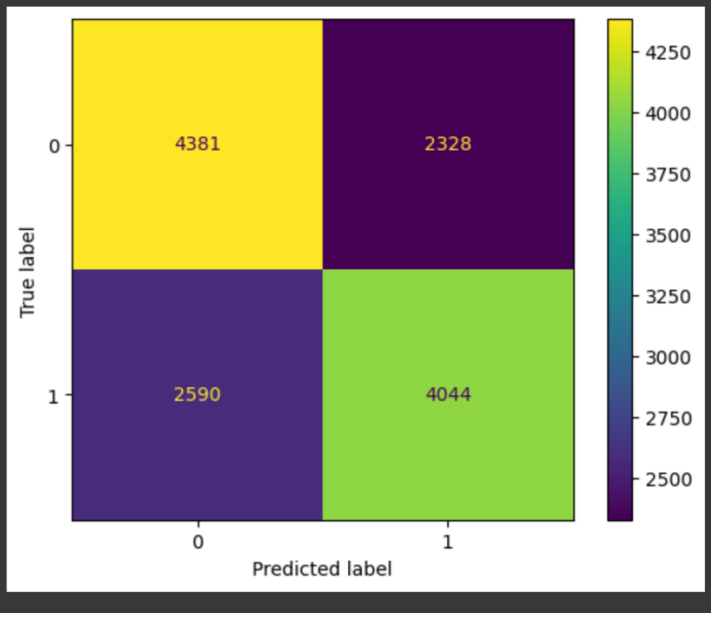
- Decision Tres Classifier

For executing Decision Tress classifier essential libraries are imported such as“from sklearn.tree import DecisionTreeClassifier”. dt_model has classifier and its hyperparameters to get better accuracy which includes 'entropy' is used for information

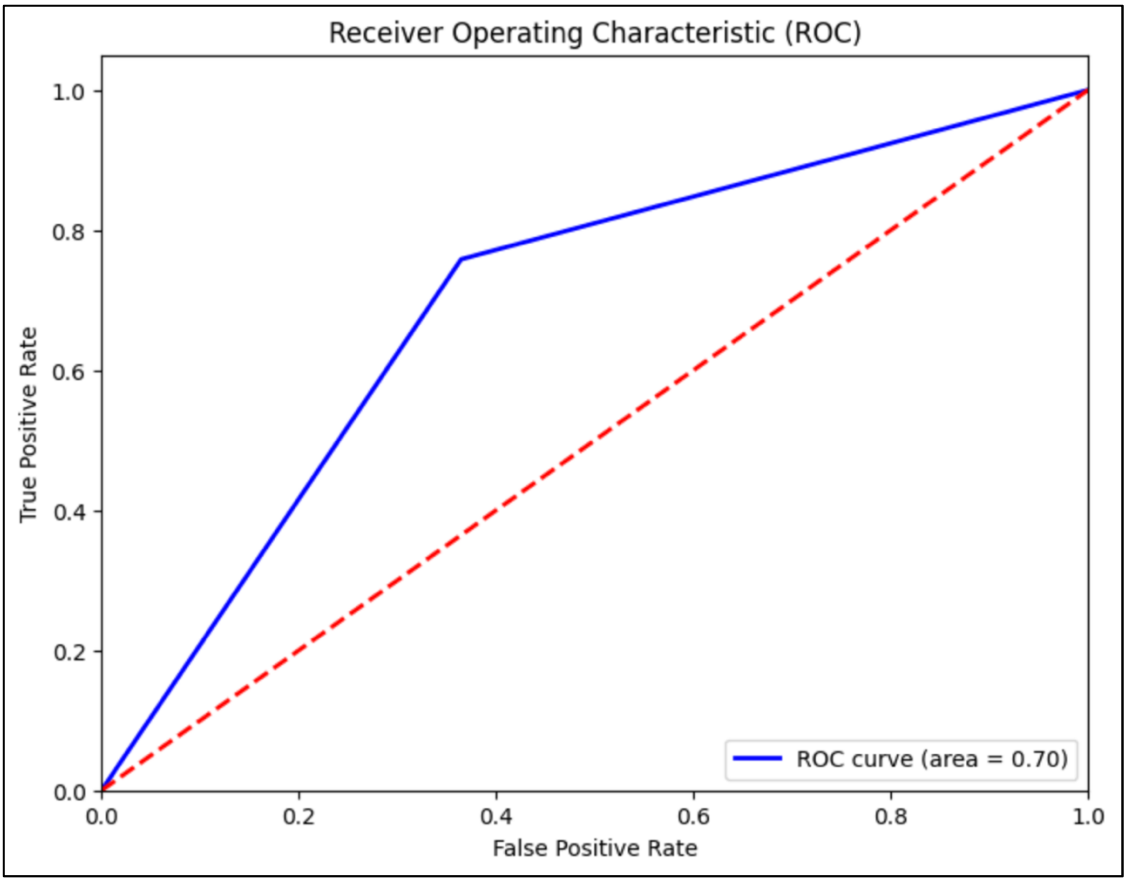
gain- criterion. Max Depth: 7, Min Samples Leaf: 2 leaf node. Min Samples Split: 2 samples required to split an internal node.
Accuracy for dt_model is 0.63

```
[23] #decsion tree classifier
     from sklearn.tree import DecisionTreeClassifier
     #adding hyper parameters to get higher accuracy
     # 'entropy' is used for information gain- criterion
     # Max Depth: 7 limits the depth of the tree
     # Min Samples Leaf: 2 minimum number of samples required to be a leaf node
     # Min Samples Split: 2  minimum number of samples required to split an internal node

     dt_model = DecisionTreeClassifier(criterion='entropy', max_depth=7, min_samples_leaf=2, min_samples_split=2, random_stat
     dt_model.fit(X_train, y_train)
     y_dt_pred = dt_model.predict(X_test)

     accuracy_dt = accuracy_score(y_test, y_dt_pred)
     print("Decision Tree Classifier Accuracy:", accuracy_dt)

     print("Classification Report for Decision Tree Classifier:\n", classification_report(y_test, y_dt_pred))


     #defining the values of y_test and y_pred to confusion matrix
     dt_confusion_matrix = confusion_matrix(y_test, y_dt_pred)


     f1_dt= f1_score(y_test,y_dt_pred)



     #displaying Confusion matrix
     disp_dt= ConfusionMatrixDisplay(confusion_matrix=dt_confusion_matrix)
     disp_dt.plot()
     plt.ylabel('True label')
     plt.xlabel('Predicted label')
     plt.show()
```

```
Decision Tree Classifier Accuracy: 0.6314172225136776
Classification Report for Decision Tree Classifier:
              precision    recall  f1-score   support

           0       0.63      0.65      0.64      6709
           1       0.63      0.61      0.62      6634

    accuracy                           0.63     13343
   macro avg       0.63      0.63      0.63     13343
weighted avg       0.63      0.63      0.63     13343
```



Area Under Curve for Decision Tress classifier is 0.70



- SVM Model

SVM model known as Support Vector Machine Model, powerful supervised machine learning algorithm used for both classification and regression tasks. Essential library was imported “from sklearn.svm import SVC”. SVM has types of kernel, I executed using Radial Basis Function(rbf) but the accuracy was less, whereas in liner kernel accuracy score was better and fast-processing.

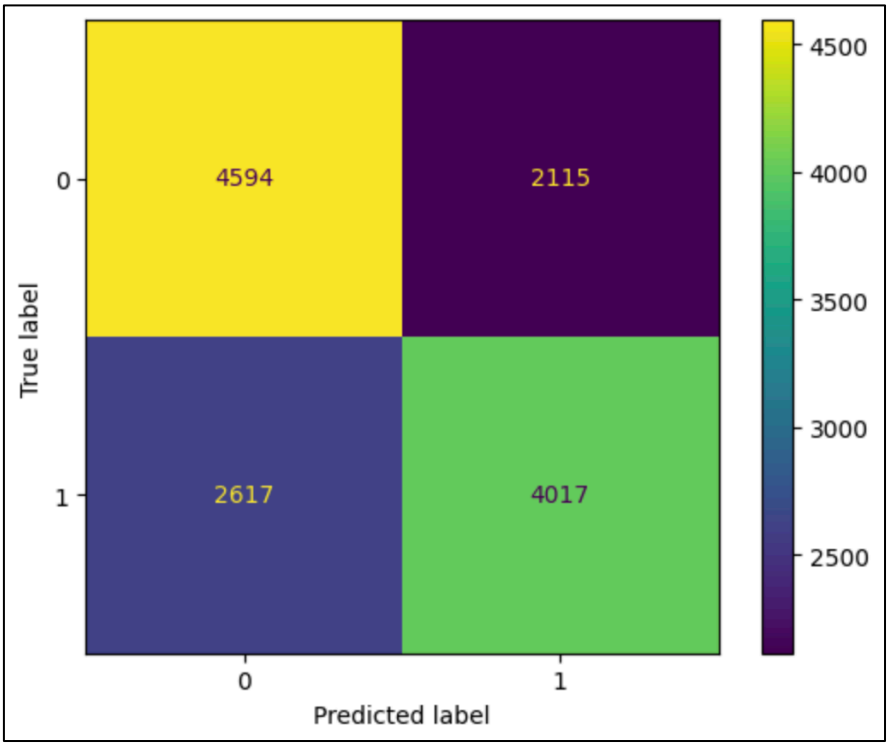
Hyper paramters -

svm_model = SVC(kernel='linear',random_state=42,probability= True)

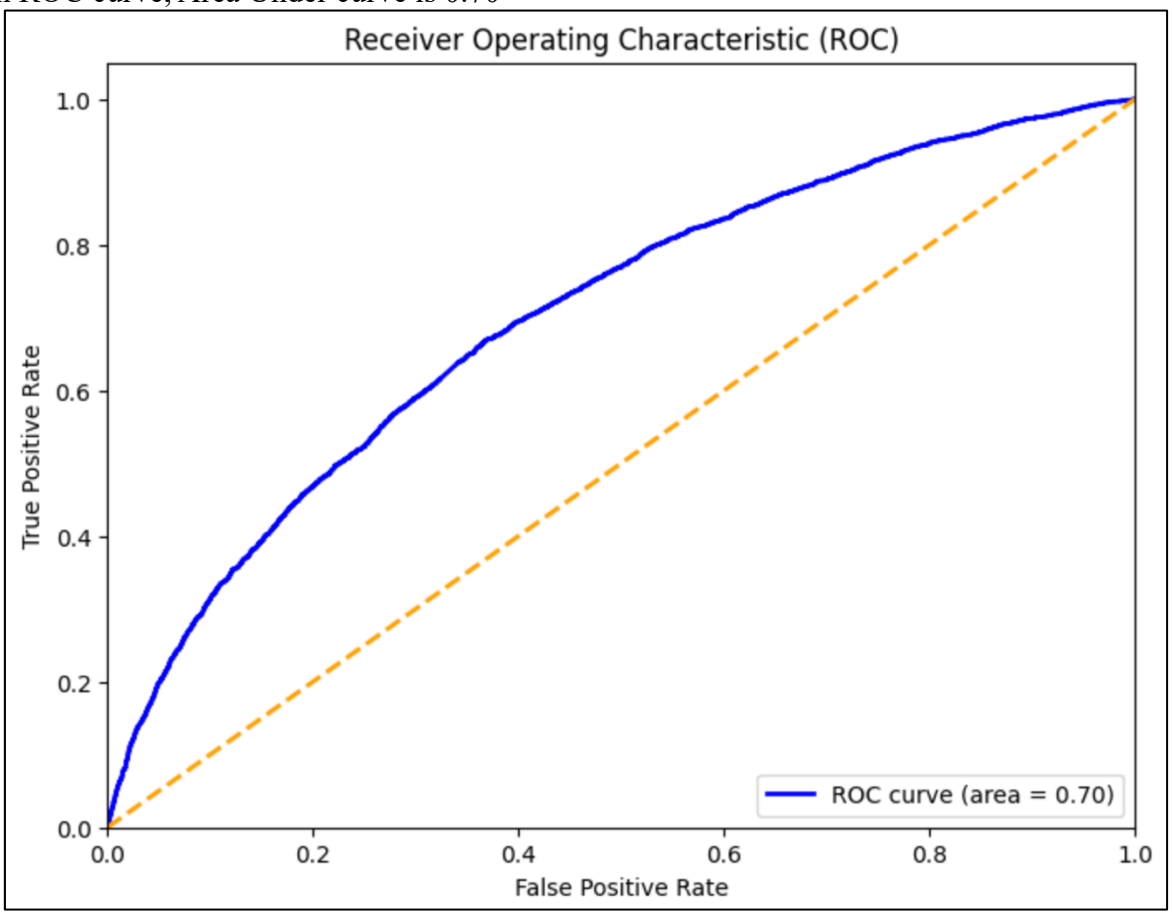
Accuracy for SVM model is 0.64

```
[25] #Support vector machine-svM
     from sklearn.svm import SVC

     svm_model = SVC(kernel='linear',random_state=42,probability= True)   #Linear Kernel
     svm_model.fit(X_train, y_train)
     y_svm_pred = svm_model.predict(X_test)

     accuracy_svm = accuracy_score(y_test, y_svm_pred)

     print ("Support Vector Classifier Accuracy:",accuracy_svm)

     print("Classification Report for SVM Classifier:\n", classification_report(y_test, y_svm_pred))

     #defining the values of y_test and y_pred to confusion matrix
     svm_confusion_matrix = confusion_matrix(y_test, y_svm_pred)


     f1_svm= f1_score(y_test,y_svm_pred)

     #displaying Confusion matrix
     disp_svm= ConfusionMatrixDisplay(confusion_matrix=svm_confusion_matrix)
     disp_svm.plot()
     plt.ylabel('True label')
     plt.xlabel('Predicted label')
     plt.show()
```

```
Support Vector Classifier Accuracy: 0.6453571160908341
Classification Report for SVM Classifier:
              precision    recall  f1-score   support

           0       0.64      0.68      0.66      6709
           1       0.66      0.61      0.63      6634
```



In ROC curve, Area Under curve is 0.70



- K-Nearest Neigbour Classifier

K-Nearest Neigbour Classifier as known as KNN classifier, is a simple and intuitive machine learning algorithm that can be used for classification and regression tasks. It is often used for its simplicity and ease of implementation.

“from sklearn.neighbors import KNeighborsClassifier” was imported for KNN classifier

knn_model = KNeighborsClassifier(n_neighbors=9, weights = 'distance', metric = 'manhattan', p= 1)

n_neighbors=9: Use 9 nearest neighbors to make predictions. AT first, 3 specified number of neighbors were used but were changed to get better accuracy

weights='distance': Give more weight to closer neighbors

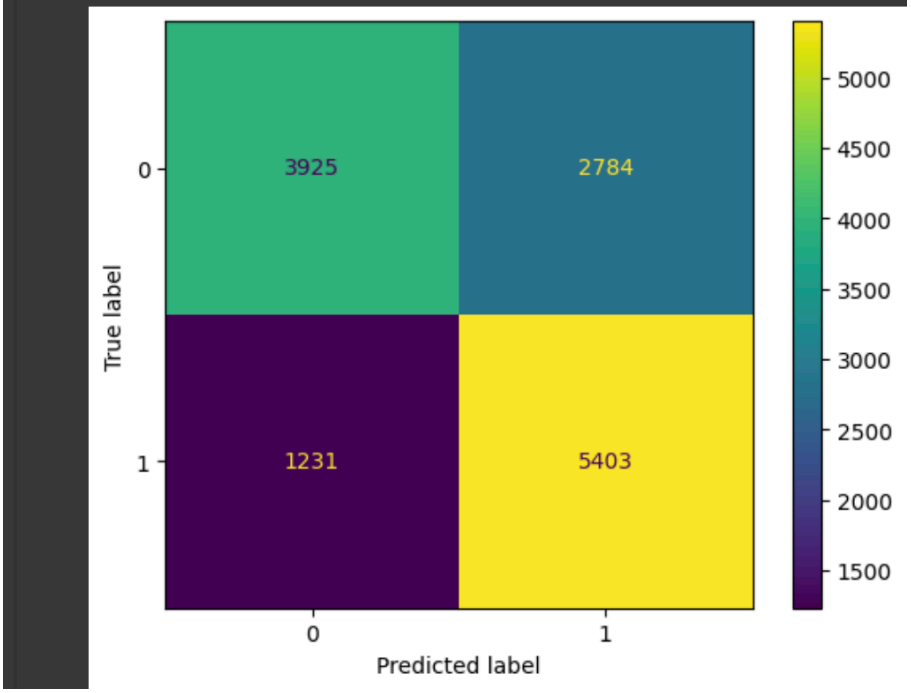
metric='manhattan': Use the Manhattan distance metric for measuring data point distances.

p=1: Equivalent to 'manhattan' distance metric

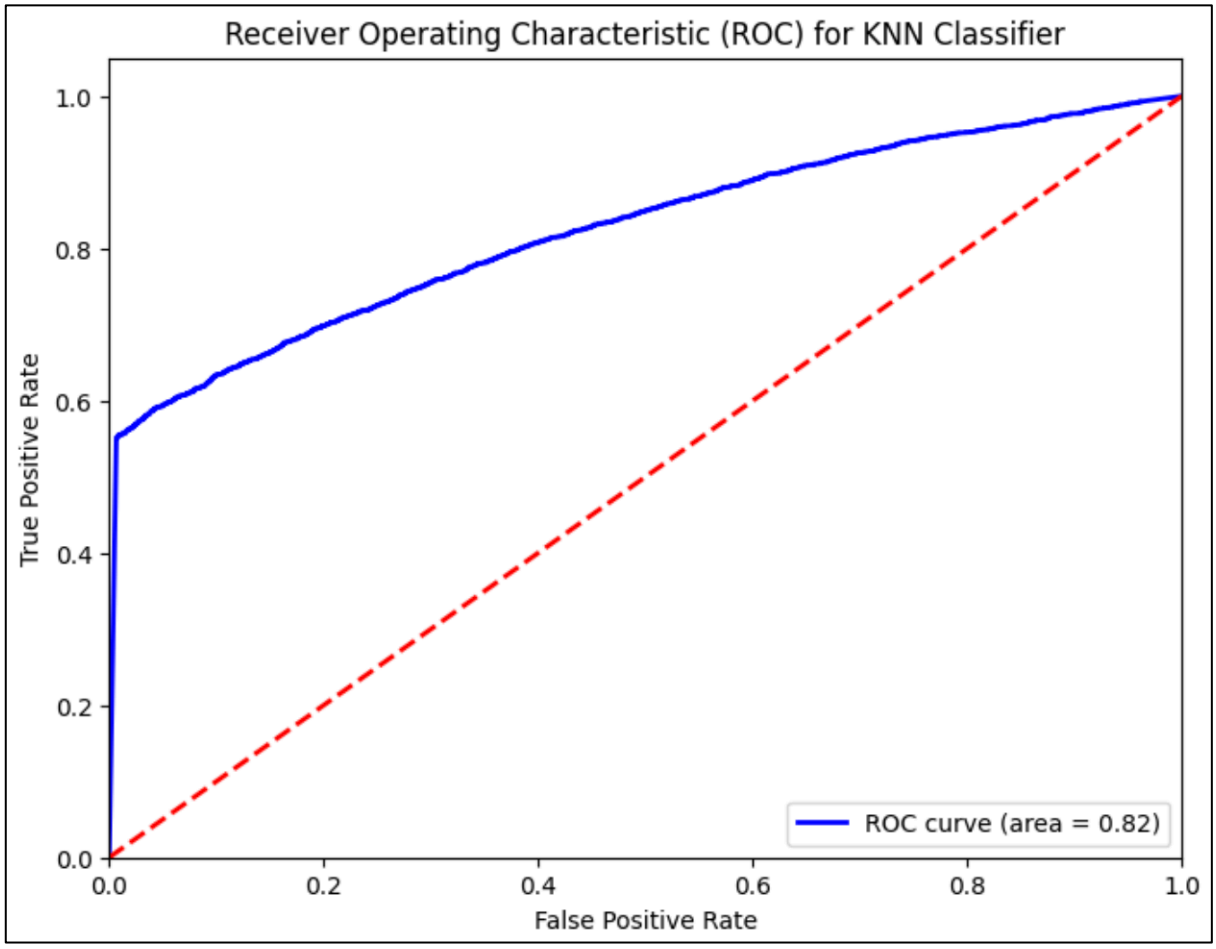
Accuracy Score for KNN classifier is 0.69

```
27] # KNN Classifier
   from sklearn.neighbors import KNeighborsClassifier
   #3 specified number of neighbors were used but were changed to get better accuracy
   knn_model = KNeighborsClassifier(n_neighbors=9, weights = 'distance', metric = 'manhattan', p= 1)
   # n_neighbors=9: Use 9 nearest neighbors to make predictions, controlling the balance between bias and variance.
   # weights='distance': Give more weight to closer neighbors, emphasizing local data structure.
   # metric='manhattan': Use the Manhattan distance metric for measuring data point distances.
   # p=1: Equivalent to using the 'manhattan' distance metric

   knn_model.fit(X_train, y_train)
   y_knn_pred = knn_model.predict(X_test)

   accuracy_knn = accuracy_score(y_test, y_knn_pred)
   print("KNN Classifier Accuracy:", accuracy_knn)

   print("Classification Report for KNN Classifier:\n", classification_report(y_test, y_knn_pred))

   #defining the values of y_test and y_pred to confusion matrix
   knn_confusion_matrix = confusion_matrix(y_test, y_knn_pred)

   f1_knn= f1_score(y_test,y_knn_pred)

   #displaying Confusion matrix
   disp_knn= ConfusionMatrixDisplay(confusion_matrix=knn_confusion_matrix)
   disp_knn.plot()
   plt.ylabel('True label')
   plt.xlabel('Predicted label')
   plt.show()

   # KNN ROC curve
   print("\n")

   y_pred_prob_knn = knn_model.predict_proba(X_test)[:, 1]

   fpr_knn, tpr_knn, thresholds_knn = roc_curve(y_test, y_pred_prob_knn)
   roc_auc_knn = roc_auc_score(y_test, y_pred_prob_knn)
```

```
KNN Classifier Accuracy: 0.6990931574608409
Classification Report for KNN Classifier:
              precision    recall  f1-score   support

           0       0.76      0.59      0.66      6709
           1       0.66      0.81      0.73      6634

    accuracy                           0.70     13343
   macro avg       0.71      0.70      0.70     13343
weighted avg       0.71      0.70      0.70     13343
```



In ROC curve, Area under Curve is 0.82



- Neural Networks- MLP classifier

MLP classifier known as Multilayer Perceptron is a type of neural network classifier used for various machine learning and deep learning tasks. Essential library is imported for Executing MLP classifier "from sklearn.neural_network import MLPClassifier". For better result, Proper preprocessing, hyperparameter tuning, and handling imbalanced datasets are important.

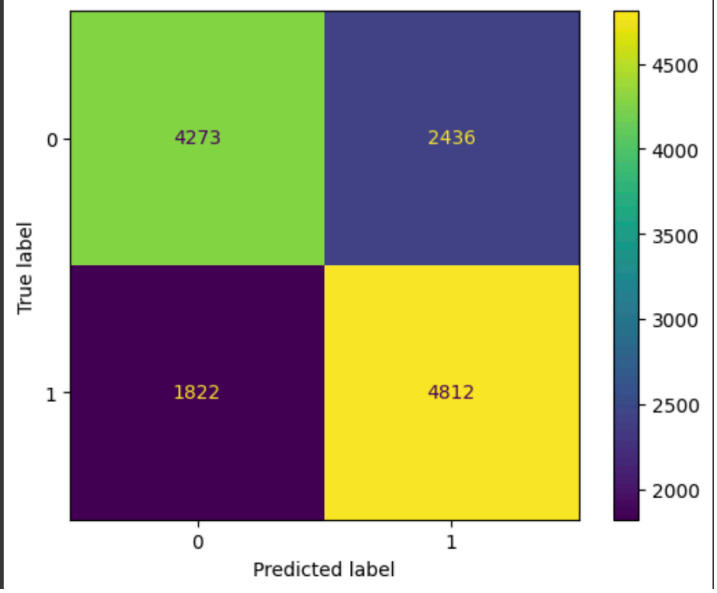
Hyper parameters were used for better accuracy

```
mlp_model = MLPClassifier(hidden_layer_sizes=(100, 50), max_iter=1000,
random_state=42)
#hidden_layer_sizes=(100, 50): with two hidden layers, containing 100 and 50 neurons,
respectively.
# max_iter=1000: Maximum number of iterations for the solver to converge.
```

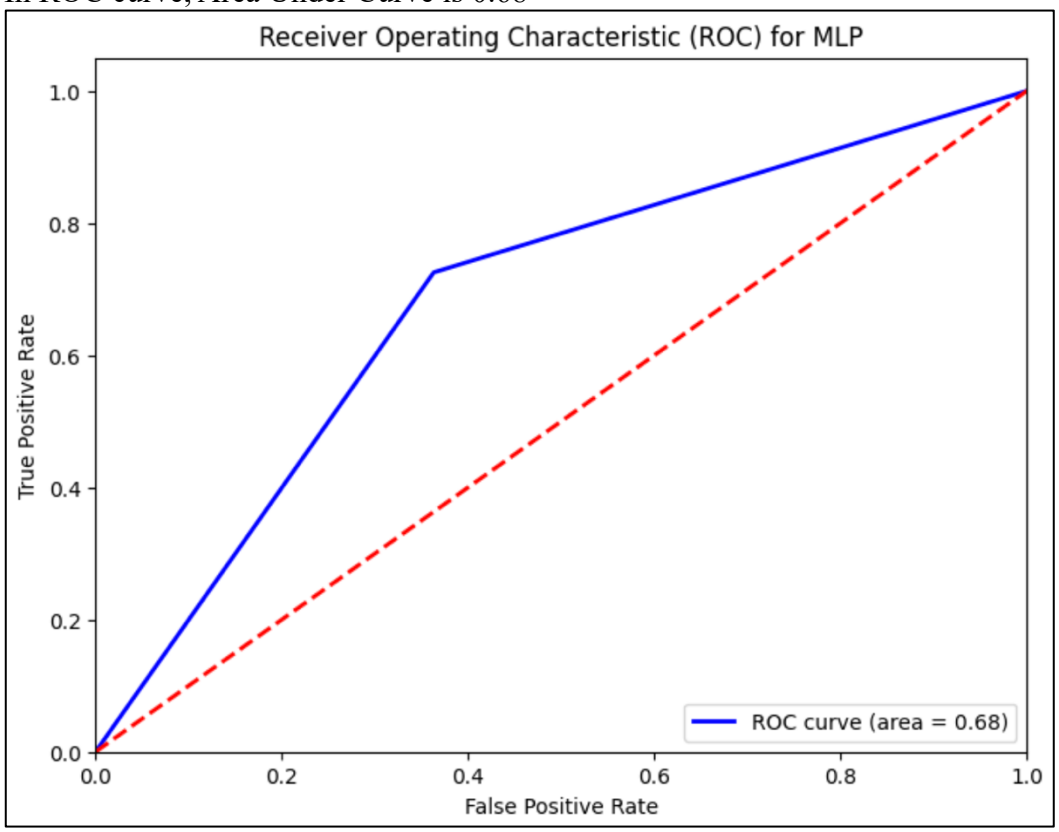
Accuracy for MLP classifier is 0.68

```
28] #Neural Networks using MLP classifier
    from sklearn.neural_network import MLPClassifier


    mlp_model = MLPClassifier(hidden_layer_sizes=(100, 50), max_iter=1000, random_state=42)
    #hidden_layer_sizes=(100, 50): with two hidden layers, containing 100 and 50 neurons, respectively.
    # max_iter=1000: Maximum number of iterations for the solver to converge.
    mlp_model.fit(X_train, y_train)
    y_mlp_pred = mlp_model.predict(X_test)

    accuracy = accuracy_score(y_test, y_mlp_pred)
    print("MLP Accuracy:", accuracy)
    print("Classification Report:\n", classification_report(y_test, y_mlp_pred))
    print("Confusion Matrix:\n", confusion_matrix(y_test, y_mlp_pred))

    #defining the values of y_test and y_pred to confusion matrix
    mlp_confusion_matrix = confusion_matrix(y_test, y_mlp_pred)

    f1_mlp= f1_score(y_test,y_mlp_pred)

    #displaying Confusion matrix
    disp_mlp= ConfusionMatrixDisplay(confusion_matrix=mlp_confusion_matrix)
    disp_mlp.plot()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.show()

    print("\n")
    #Neural Networks ROC curve

    fpr_mlp, tpr_mlp, thresholds_mlp = roc_curve(y_test, y_mlp_pred)
    roc_auc_mlp = roc_auc_score(y_test, y_mlp_pred)
```

```
MLP Accuracy: 0.6808813610132654
Classification Report:
              precision    recall  f1-score   support

           0       0.70      0.64      0.67      6709
           1       0.66      0.73      0.69      6634

    accuracy                           0.68     13343
   macro avg       0.68      0.68      0.68     13343
weighted avg       0.68      0.68      0.68     13343

Confusion Matrix:
 [[4273 2436]
 [1822 4812]]
```



In ROC curve, Area Under Curve is 0.68

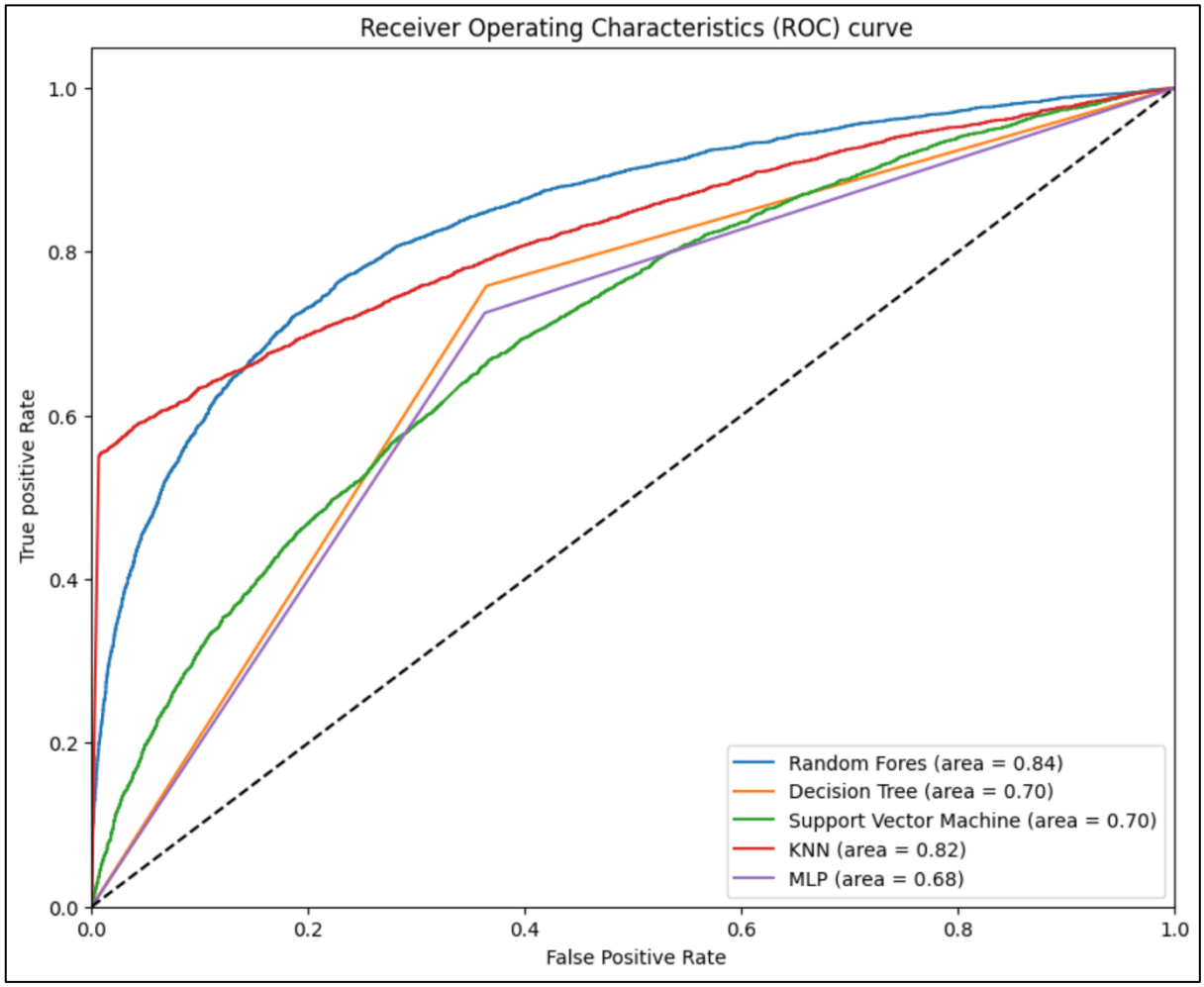


To get ROC curve for all classifier used for identifying which one should be selected for predicting in Unknown dataset I used- ProgressBar library. I assigned models variable which stores all classifiers value and prob_scores which stores all predicted value i.e y_pred_prob_[classifiername]
Code:
from IPython.core.display import ProgressBar
models= [rf_model,dt_model,svm_model,knn_model,mlp_model]
prob_scores = [y_pred_prob_rf, y_pred_prob_dt,y_pred_prob,y_pred_prob_knn,y_mlp_pred]
Model_names=['Random Fores','Decision Tree','Support Vector Machine','KNN','MLP']

All the ROC curve value is stored and plotted using for loop.

```
]  from sklearn.metrics import roc_curve, auc

   y_true = y_test

   plt.figure(figsize=(10,8))
   for name, scores in zip( Model_names, prob_scores):
     fpr , tpr, threshold = roc_curve(y_true, scores)
     roc_auc = auc(fpr, tpr)
     plt.plot(fpr, tpr, label = f"{name} (area = {roc_auc:.2f})")

   plt.plot([0,1],[0,1], 'k--') # Random Classifier line that passes through
   plt.xlim([0.0, 1.0])
   plt.ylim([0.0, 1.05])
   plt.xlabel(' False Positive Rate')
   plt.ylabel(' True positive Rate')
   plt.title( ' Receiver Operating Characteristics (ROC) curve')
   plt.legend(loc = "lower right")
   plt.show()
```

ROC curve and Area under curve of all five classifier



Loading Unknown dataset to predict target values and checking the info of columns using unknown_data.info()

```
[ ] #Load the unknown dataset
    unknown_data = pd.read_csv('/content/drive/MyDrive/FDA_Dataset_A3/loan_data_unknown.csv')

    unknown_data.info()

    <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 4716 entries, 0 to 4715
    Data columns (total 71 columns):
     #   Column                  Non-Null Count  Dtype
    ---  ------                  --------------  -----
     0   SK_ID_CURR              4716 non-null   int64
     1   NAME_CONTRACT_TYPE      4716 non-null   object
     2   CODE_GENDER             4716 non-null   object
     3   FLAG_OWN_CAR            4716 non-null   object
     4   FLAG_OWN_REALTY         4716 non-null   object
     5   CNT_CHILDREN            4716 non-null   int64
     6   AMT_INCOME_TOTAL        4716 non-null   float64
     7   AMT_CREDIT              4716 non-null   float64
     8   AMT_ANNUITY             4716 non-null   float64
     9   AMT_GOODS_PRICE         4710 non-null   float64
```

To identify missing value, isna().sum function is used to get sum of all the null values and their column name.

```
#identifying missing values
missing_values = unknown_data.isna().sum()
print("Missing values by column:")
print(missing_values[missing_values > 0])
```

```
Missing values by column:
AMT_GOODS_PRICE                 6
NAME_TYPE_SUITE                19
EXT_SOURCE_2                   12
EXT_SOURCE_3                 1022
OBS_30_CNT_SOCIAL_CIRCLE       13
DEF_30_CNT_SOCIAL_CIRCLE       13
OBS_60_CNT_SOCIAL_CIRCLE       13
DEF_60_CNT_SOCIAL_CIRCLE       13
AMT_REQ_CREDIT_BUREAU_HOUR    727
AMT_REQ_CREDIT_BUREAU_DAY     727
AMT_REQ_CREDIT_BUREAU_WEEK    727
AMT_REQ_CREDIT_BUREAU_MON     727
AMT_REQ_CREDIT_BUREAU_QRT     727
AMT_REQ_CREDIT_BUREAU_YEAR    727
dtype: int64
```

After the count of missing values, dropping the columns with no-corelation same as columns deleted in training dataset.
'EXT_SOURCE_3'
'AMT_REQ_CREDIT_BUREAU_HOUR','AMT_REQ_CREDIT_BUREAU_DAY','AMT_REQ_CREDIT_BUREAU_WEEK','AMT_REQ_CREDIT_BUREAU_MON','AMT_REQ_CREDIT_BUREAU_QRT','AMT_REQ_CREDIT_BUREAU_YEAR','TARGET' and FLAG_DOCUMENT_* were multiple 0 values

Second code is to differentiate the numerical and categorical columns using same technique of nunique for categorical columns and median value for numeric columns.

```
#data drop based on corelation through heatmap and histogram visuals
unknown_data= unknown_data.drop(['EXT_SOURCE_3','FLAG_MOBIL'],axis=1)
#multiple columns of FLAG_DOCUMENT and AMT_REQ_CREDIT_BUREAU_*
unknown_data= unknown_data.drop([ 'FLAG_DOCUMENT_2','FLAG_DOCUMENT_4', 'FLAG_DOCUMENT_5', 'FLAG_DOCUMENT_6','FLAG_DOCUMENT_7',
           'FLAG_DOCUMENT_8', 'FLAG_DOCUMENT_9','FLAG_DOCUMENT_10', 'FLAG_DOCUMENT_11', 'FLAG_DOCUMENT_12','FLAG_DOCUMENT_13',
           'FLAG_DOCUMENT_14', 'FLAG_DOCUMENT_15','FLAG_DOCUMENT_16', 'FLAG_DOCUMENT_17', 'FLAG_DOCUMENT_18','FLAG_DOCUMENT_19', 'FLAG_DO
```

```
#unknown is written before every variable to differentiate it from training dataset.
# Identifying datatypes of column as  numerical and categorical columns
unknown_numerical_cols = unknown_data.select_dtypes(include=['float64', 'int64']).columns
unknown_categorical_cols = unknown_data.select_dtypes(include=['object']).columns

unknown_categorical_distribution = unknown_data[unknown_categorical_cols].nunique()
#median for numeric values
unknown_median_values = unknown_data[unknown_numerical_cols].median()

# using median  for missing value
unknown_data[unknown_numerical_cols] = unknown_data[unknown_numerical_cols].fillna(unknown_median_values)
```

Label encoder for unknown dataset- categorical columns using “from sklearn.preprocessing import LabelEncoder” library.

```
label_encoder = LabelEncoder()
unknown_data['CODE_GENDER'] = label_encoder.fit_transform(unknown_data['CODE_GENDER'])
unknown_data['NAME_CONTRACT_TYPE'] = label_encoder.fit_transform(unknown_data['NAME_CONTRACT_TYPE'])
unknown_data['FLAG_OWN_CAR'] = label_encoder.fit_transform(unknown_data['FLAG_OWN_CAR'])
unknown_data['FLAG_OWN_REALTY'] = label_encoder.fit_transform(unknown_data['FLAG_OWN_REALTY'])
unknown_data['NAME_TYPE_SUITE'] = label_encoder.fit_transform(unknown_data['NAME_TYPE_SUITE'])
unknown_data['NAME_INCOME_TYPE'] = label_encoder.fit_transform(unknown_data['NAME_INCOME_TYPE'])
unknown_data['NAME_EDUCATION_TYPE'] = label_encoder.fit_transform(unknown_data['NAME_EDUCATION_TYPE'])
unknown_data['NAME_FAMILY_STATUS'] = label_encoder.fit_transform(unknown_data['NAME_FAMILY_STATUS'])
unknown_data['NAME_HOUSING_TYPE'] = label_encoder.fit_transform(unknown_data['NAME_HOUSING_TYPE'])
unknown_data['WEEKDAY_APPR_PROCESS_START'] = label_encoder.fit_transform(unknown_data['WEEKDAY_APPR_PROCESS_START'])
unknown_data['ORGANIZATION_TYPE'] = label_encoder.fit_transform(unknown_data['ORGANIZATION_TYPE'])
```

```
unknown_data.head(10)
```

	SK_ID_CURR	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	A
0	280139	0	1	1	1	2	135000.0	592560.0	31023.0	
1	284067	0	1	1	0	1	427500.0	450000.0	22018.5	
2	213484	0	0	0	0	0	225000.0	521280.0	35262.0	
3	430855	0	1	0	1	0	135000.0	1215000.0	35523.0	
4	342521	0	1	1	0	1	202500.0	355536.0	15790.5	
5	233048	0	1	1	1	0	112500.0	450000.0	23692.5	
6	441011	0	1	0	1	0	225000.0	325908.0	17811.0	
7	207597	0	1	0	1	1	270000.0	601470.0	32760.0	
8	282567	0	0	0	1	0	67500.0	95940.0	9342.0	
9	314049	0	0	0	1	0	189000.0	301464.0	23949.0	

Using standardscaler() in unknown dataset to split the data in unknown_data_X and unknown_data_test.
Using x_train_without_id which has no SK_ID_CURR in unknown dataset.

```
unknown_scaler= StandardScaler()
unknown_data_X = unknown_data.drop(['SK_ID_CURR'], axis = 1)
unknown_data_test = unknown_scaler.fit_transform(unknown_data_X)
```

```
# Convert the NumPy array back to a DataFrame
columns_no_SK_ID_CURR = [col for col in unknown_data.columns if col not in [ 'SK_ID_CURR']]
X_train_without_id = pd.DataFrame (unknown_data_test, columns=columns_no_SK_ID_CURR)
```

Importing JOBLIB library to pipeline and handle large data which will use machine learning model.In this case, I have used joblib.dump to randomforest classifier in my folder and with the help of joblib.load I have loaded the random forest classifier model into rf_mj Variable.

```
] import joblib

] joblib.dump(rf_model,'/content/drive/MyDrive/FDA_Dataset_A3/rf_model_joblib')
  ['/content/drive/MyDrive/FDA_Dataset_A3/rf_model_joblib']

] rf_mj= joblib.load('/content/drive/MyDrive/FDA_Dataset_A3/rf_model_joblib') #loading
```

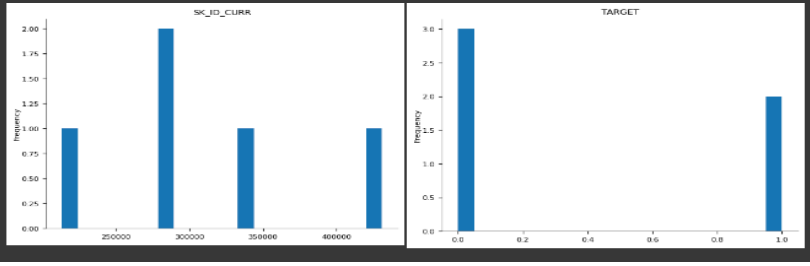
Using rf_mj which has random forest classifier to predict TARGET values for the unknown dataset using SK_ID_CURR as reference column. rf_mj uses the pre-trained Random Forest classifier to predict the TARGET values for the unknown dataset. After obtaining the predictions, predicted_df is created which has SK_ID_CURR and predicted TARGET value. This prediction is saved in predicted_dt and it is displayed using predicted_dt.head()

```
] #predicting unknown dataset TARGET value using SK_ID_CURR
  unknown_predictions = rf_mj.predict(X_train_without_id)

  predicted_df=pd.DataFrame({
    'SK_ID_CURR':unknown_data['SK_ID_CURR'],
     'TARGET':unknown_predictions,
  })

  predicted_df.head(5)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:432: U
  warnings.warn(
```

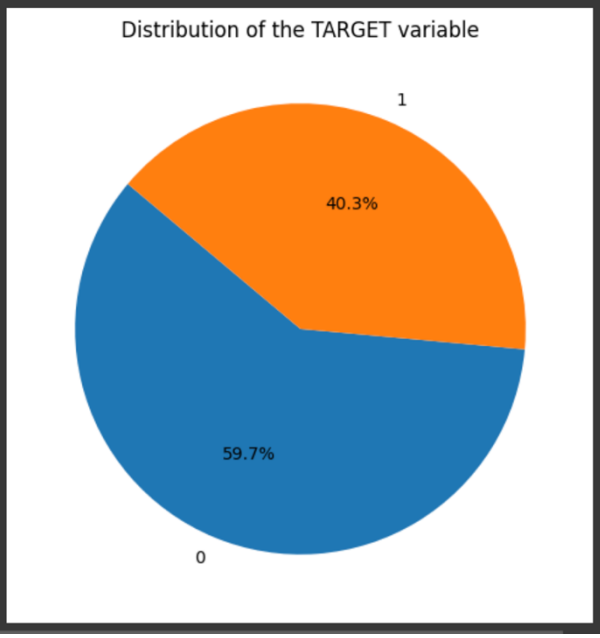
	SK_ID_CURR	TARGET
0	280139	1
1	284067	1
2	213484	0
3	430855	0
4	342521	0

Distributions



Plotting the TARGET value in pie chart using import matplotlib.pyplot as plt.
TARGET value 0 count is 59.7% and value 1 count is 40.3%.

```
import pandas as pd
import matplotlib.pyplot as plt

# Calculate the normalized value counts for the 'TARGET' column
unknown_target_distribution = predicted_df['TARGET'].value_counts(normalize=True)

# Creating  a pie chart for TARGET distribution (0,1)
plt.figure(figsize=(6, 6))
plt.pie(unknown_target_distribution, labels=unknown_target_distribution.index, autopct='%1.1f%%', startangle=140)
plt.title('Distribution of the TARGET variable')
plt.show()
```



Saving the SK_ID_CURR and TARGET Value of unknown dataset which is predicted in csv file- loan_data_predicted.csv

```
# Saving the predictions in  CSV file
predictions_path = '/content/drive/MyDrive/FDA_Dataset_A3/loan_data_predicted.csv'
predicted_df.to_csv(predictions_path, index=False)
```

Python code link:
https://colab.research.google.com/drive/1HLPGNHcGAf3f4kIKUBemgya02lw0El-j?usp=sharing

4. Justification of the classifier selected.

Random Forest classifier is the best classifier due to highest accuracy rate of 0.76 among all classifiers. Another factor is ROC curve of all classifiers, the area under curve for Random forest is 0.84 which higher compared to other classifier. This classifier execution was faster even with hyper parameters- RandomForestClassifier(max_depth=30, min_samples_leaf=2, min_samples_split=10, n_estimators=300, random_state=42). IN confusion matrix the true predicted value is more compared to other classifier, for true positive it is 5108 and 5124 respectively which shows the classifier has better results. Due to ensemble of decision trees

in random forest, each tree is built on a random subset of the training data which helps in reducing the overfitting of the classifier . The aggregation of multiple trees reduces variance, resulting to less prone to overfitting.

5. Reflection

This assignment of data analytics in action is knowledge centric and allowed me to understand how classifier can be implemented to predict data by training and testing them. The previous assignment gave insight on to how data should be understood and by that cleaning of data is very essential, this assignment gave insight on to how I can use the data to train and predict information for me based on the criteria required. When I first started working on the assignment I was trying to implement decision tree classifier as it is easiest way to understand data, I tried in knime without cleaning data the accuracy was 47% which shows that my classifier needs cleaned data, and I should first find out my columns correlation and then work on pre-processing. I started this assignment in python while data cleaning I removed DAYS_EMPLOYED positive values at first and EXT_SOURCE_2 but after feature importance I found out these values are important and had to perform trial and error, through which I am well-versed in python coding for data cleaning, working with classifier and predicting the unknown data using trained model. While using classifier I took help of very precise information provided through canvas modules and workshop notes and while creating confusion matrix for all classifier I was confused because on some website for python code learning they have true and predicted label on X and Y axis respectively, but I studied through YouTube video.

This assignment helped me to understand that coding to get desired outcome will come from various trial and error. Some classifiers such as SVM and MLP(neural network) were time consuming and sometime delayed the process for execution. The learning I gained through this assignment is very useful and helpful for real-world problems. This subject and the structured assignment interest me to learn more and due to which I have undertaken various courses in LinkedIn Learning related to python for data analytics and KNIME for Data Science