Logistic regression model , Decision tree model , Random forest model and Plotting ROC curves

```
# Importing the libraries
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import matplotlib.pyplot as plt
import seaborn as sns
import os, sys
!pip install kfinance\*\*(sys.stdin)
```

```
# Importing the data
!wget -qq https://drive.google.com/uc?export=download&id=1L-Ocwt_-HR-DbpqAxxw-Ovrcjuan_c
```

```
Age  Attrition  BusinessTravel  DailyRate  Department  DistanceFromHome  Education  EducationField  EmployeeCount  EmployeeNumber  ...  RelationshipSatisfaction  StandardHours  StockOptionLevel  TotalWorkingYears  Trai
0   41    Yes   Travel_Rarely       1102       Sales                 1          2  Life Sciences              1               1  ...                         1             80                  0                  8     0
1   49     No  Travel_Frequently        279  Research & Development                8          1  Life Sciences              1               2  ...                         4             80                  1                 10     3
2   37    Yes   Travel_Rarely       1373  Research & Development                2          2           Other              1               4  ...                         2             80                  0                  7     3
3   33     No  Travel_Frequently       1392  Research & Development                3          4  Life Sciences              1               5  ...                         3             80                  0                  8     3
4   27     No   Travel_Rarely        591  Research & Development                2          1        Medical              1               7  ...                         4             80                  1                  6     3

[5 rows x 35 columns]
```

```
# Data Cleaning
# Exploratory Data Analysis (EDA)

print(df.info())
df.describe(include='all')

class= df[df['Attrition'] == 'Yes']
print('Percentage of Employees Who Left: ', round((len(class)/(len(df))*100),2))
print('Number of Employees Who Left: ', len(class))
df = df.drop(['Over18', 'EmployeeCount', 'StandardHours'], axis = 1)
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Age                       1470 non-null   int64
 1   Attrition                 1470 non-null   object
 2   BusinessTravel            1470 non-null   object
 3   DailyRate                 1470 non-null   int64
 4   Department                1470 non-null   object
 5   DistanceFromHome          1470 non-null   int64
 6   Education                 1470 non-null   int64
 7   EducationField            1470 non-null   object
 8   EmployeeCount             1470 non-null   int64
 9   EmployeeNumber            1470 non-null   int64
 10  EnvironmentSatisfaction   1470 non-null   int64
 11  Gender                    1470 non-null   object
 12  HourlyRate                1470 non-null   int64
 13  JobInvolvement            1470 non-null   int64
 14  JobLevel                  1470 non-null   int64
 15  JobRole                   1470 non-null   object
 16  JobSatisfaction           1470 non-null   int64
 17  MaritalStatus             1470 non-null   object
 18  MonthlyIncome             1470 non-null   int64
 19  MonthlyRate               1470 non-null   int64
 20  NumCompaniesWorked        1470 non-null   int64
 21  Over18                    1470 non-null   object
 22  OverTime                  1470 non-null   object
 23  PercentSalaryHike         1470 non-null   int64
 24  PerformanceRating         1470 non-null   int64
 25  RelationshipSatisfaction  1470 non-null   int64
 26  StandardHours             1470 non-null   int64
 27  StockOptionLevel          1470 non-null   int64
 28  TotalWorkingYears         1470 non-null   int64
 29  TrainingTimesLastYear     1470 non-null   int64
 30  WorkLifeBalance           1470 non-null   int64
 31  YearsAtCompany            1470 non-null   int64
 32  YearsInCurrentRole        1470 non-null   int64
 33  YearsSinceLastPromotion   1470 non-null   int64
 34  YearsWithCurrManager      1470 non-null   int64
dtypes: int64(26), object(9)
memory usage: 402.1 KB
None
Age                         0
Attrition                   0
BusinessTravel              0
DailyRate                   0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EmployeeCount               0
EmployeeNumber              0
EnvironmentSatisfaction     0
Gender                      0
HourlyRate                  0
JobInvolvement              0
JobLevel                    0
JobRole                     0
JobSatisfaction             0
MaritalStatus               0
MonthlyIncome               0
MonthlyRate                 0
NumCompaniesWorked          0
Over18                      0
OverTime                    0
PercentSalaryHike           0
PerformanceRating           0
RelationshipSatisfaction    0
StandardHours               0
StockOptionLevel            0
TotalWorkingYears           0
TrainingTimesLastYear       0
WorkLifeBalance             0
YearsAtCompany              0
YearsInCurrentRole          0
YearsSinceLastPromotion     0
YearsWithCurrManager        0
dtype: int64
```

```
# We will change the datatype of our target variable column (change from object to category)
df_dummy['Attrition'].\*.dtype.\*\*(sys.stdin)
```

```
df_dummy.info()
df_dummy.describe(include='all')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Age                       1470 non-null   int64
 1   Attrition                 1470 non-null   category
 2   BusinessTravel            1470 non-null   object
 3   DailyRate                 1470 non-null   int64
 4   Department                1470 non-null   object
 5   DistanceFromHome          1470 non-null   int64
 6   Education                 1470 non-null   int64
 7   EducationField            1470 non-null   object
 8   EmployeeCount             1470 non-null   int64
 9   EmployeeNumber            1470 non-null   int64
 10  EnvironmentSatisfaction   1470 non-null   int64
 11  Gender                    1470 non-null   object
 12  HourlyRate                1470 non-null   int64
 13  JobInvolvement            1470 non-null   int64
 14  JobLevel                  1470 non-null   int64
 15  JobRole                   1470 non-null   object
 16  JobSatisfaction           1470 non-null   int64
 17  MaritalStatus             1470 non-null   object
 18  MonthlyIncome             1470 non-null   int64
 19  MonthlyRate               1470 non-null   int64
 20  NumCompaniesWorked        1470 non-null   int64
 21  Over18                    1470 non-null   object
 22  OverTime                  1470 non-null   object
 23  PercentSalaryHike         1470 non-null   int64
 24  PerformanceRating         1470 non-null   int64
 25  RelationshipSatisfaction  1470 non-null   int64
 26  StandardHours             1470 non-null   int64
 27  StockOptionLevel          1470 non-null   int64
 28  TotalWorkingYears         1470 non-null   int64
 29  TrainingTimesLastYear     1470 non-null   int64
 30  WorkLifeBalance           1470 non-null   int64
 31  YearsAtCompany            1470 non-null   int64
 32  YearsInCurrentRole        1470 non-null   int64
 33  YearsSinceLastPromotion   1470 non-null   int64
 34  YearsWithCurrManager      1470 non-null   int64
dtypes: int64(26), object(9)
memory usage: 402.1 KB
```

```
df_dummy.shape
```

```
(1470, 35)
```

```
# We will divide dataset into a smaller dataset and non-numerical columns into a numerical column
# Numerical column will be used for training and non-numerical column for plotting graphs

df_num = df_dummy[['Attrition', 'Department', 'Education', 'EducationField', 'JobRole', 'MaritalStatus', 'OverTime', 'Gender']]

df_final = pd.get_dummies(df_data, columns=cat_col, drop_first=True)
```

```
df_final.head()
```

```
Age  Attrition  DailyRate  DistanceFromHome  Education  EmployeeCount  EmployeeNumber  EnvironmentSatisfaction  HourlyRate  JobInvolvement  ...  MaritalStatus_Single  OverTime_Yes  EducationField_Human Resources  EducationField_Life Sciences  EducationField_Marketing
0   41    Yes       1102                 1          2              1               1                        2        94               3  ...                     1             1                               0                             1                         0
1   49     No        279                 8          1              1               2                        3        61               2  ...                     0             0                               0                             1                         0
2   37    Yes       1373                 2          2              1               4                        4        92               2  ...                     1             1                               0                             0                         0
3   33     No       1392                 3          4              1               5                        4        56               3  ...                     0             0                               0                             1                         0
4   27     No        591                 2          1              1               7                        1        40               3  ...                     0             1                               0                             0                         1

[5 rows x 35 columns]
```

```
# Add the Attrition column let us convert this into numerical form by using LabelEncoder

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df_num['Attrition'] = le.fit_transform(df_num['Attrition'])
df_num.head()
```

```
Age  Attrition  DailyRate  DistanceFromHome  Education  EmployeeCount  EmployeeNumber  EnvironmentSatisfaction  HourlyRate  JobInvolvement  ...  MaritalStatus_Single  OverTime_Yes  EducationField_Human Resources  EducationField_Life Sciences  EducationField_Marketing
0   41          1       1102                 1          2              1               1                        2        94               3  ...                     1             1                               0                             1                         0
1   49          0        279                 8          1              1               2                        3        61               2  ...                     0             0                               0                             1                         0
2   37          1       1373                 2          2              1               4                        4        92               2  ...                     1             1                               0                             0                         0
3   33          0       1392                 3          4              1               5                        4        56               3  ...                     0             0                               0                             1                         0
4   27          0        591                 2          1              1               7                        1        40               3  ...                     0             1                               0                             0                         1

[5 rows x 35 columns]
```

```
# Summary Statistics of Dataset
# Mean Analysis

df_num.describe()
```

```
Attrition  DailyRate  DistanceFromHome  Education  EmployeeCount  EmployeeNumber  EnvironmentSatisfaction  HourlyRate  JobInvolvement  JobLevel  ...  MaritalStatus_Single  OverTime_Yes  EducationField_Human Resources  EducationField_Life Sciences  EducationField_Marketing
count  1470.000000  1470.000000       1470.000000  1470.000000       1470.0  1470.000000              1470.000000  1470.000000     1470.000000  1470.000  ...           1470.000000   1470.000000                     1470.000000                   1470.000000                  1470.0
mean      0.161224   802.485714          9.192517     2.912925          1.0  1024.865306                 2.721769    65.891156        2.729932     2.063  ...              0.333333      0.282993                        0.016327                      0.412245                       0.0
std       0.367863   403.509100          8.106864     1.024165          0.0   602.024335                 1.093082    20.329428        0.711561     1.106  ...              0.471693      0.450567                        0.126616                      0.492451                       0.0
min       0.000000   102.000000          1.000000     1.000000          1.0     1.000000                 1.000000    30.000000        1.000000     1.000  ...              0.000000      0.000000                        0.000000                      0.000000                       0.0
25%       0.000000   465.000000          2.000000     2.000000          1.0   491.250000                 2.000000    48.000000        2.000000     1.000  ...              0.000000      0.000000                        0.000000                      0.000000                       0.0
50%       0.000000   802.000000          7.000000     3.000000          1.0  1020.500000                 3.000000    66.000000        3.000000     2.000  ...              0.000000      0.000000                        0.000000                      0.000000                       0.0
75%       0.000000  1157.000000         24.000000     4.000000          1.0  1555.750000                 4.000000    83.750000        3.000000     3.000  ...              1.000000      1.000000                        0.000000                      1.000000                       0.0
max       1.000000  1499.000000         29.000000     5.000000          1.0  2068.000000                 4.000000   100.000000        4.000000     5.000  ...              1.000000      1.000000                        1.000000                      1.000000                       0.0

8 rows × 26 columns
```

```
# Have to standardize transform our data because there are the values which has a high range.
# Feature scaling using MinMaxScaler
from sklearn.preprocessing import MinMaxScaler
sc = StandardScaler()
df_num[cols] = sc.fit_transform(df_num[cols])
```

```
df_num.shape
```

```
(1470, 26)
```

```
# Dividing our model:
y = df_num['Attrition']
X = df_num.drop(columns=['Attrition', 'DailyRate'])
```

```
# Splitting our data into training and test set:
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
print(X_train.shape), print(X_test.shape), print(y_train.shape), print(y_test.shape)
```

```
(1176, 1176, 294, 294)
```

```
# Building our model:
from sklearn.linear_model import LogisticRegression
# Fitting the model
model = LogisticRegression()
model.fit(X_train, y_train)
```

```
LogisticRegression
LogisticRegression()
```

```
# Predicting the values on test data using your model
y_test_predictions = model.predict(X_test)
```

```
y_test
```

```
Attrition
1247    0
768     0
918     0
1070    1
923     0
       ..
192     0
629     0
559     0
291     0
1162    0
Name: Attrition, dtype: int64
```

```
# Check on Confusion matrix
# Accuracy Score
# Classification Report
# Precision, Recall, F1-Score, Support, Accuracy, Macro average, & Weighted average
# Importing the libraries
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
y_test_predictions = model.predict(X_test)
cm = confusion_matrix(y_test, y_test_predictions)
print(cm)
accuracy = accuracy_score(y_test, y_test_predictions)
print('Accuracy: ', accuracy)
print(classification_report(y_test, y_test_predictions))
```

```
[[246   1]
 [ 45   2]]
Accuracy:  0.8401360544217687
              precision    recall  f1-score   support

           0       0.85       0.99       0.91       247
           1       0.67       0.04       0.08        47

    accuracy       0.84       0.84       0.84       294
   macro avg       0.76       0.51       0.49       294
weighted avg       0.82       0.84       0.77       294
```

```
# Importing the libraries
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
# Fitting the model on training data
model_dt = dt.fit(X_train,y_train)
# Predicting the values on test data
y_test_predictions_decision_tree = model_decision_predict(X_test)
y_test
```

```
Attrition  Attrition_Predicted  Prob_Attrition_score
1247          0                    0                 0.000000
768           0                    0                 0.000000
918           0                    0                 0.000000
1070          1                    1                 1.000000
923           0                    0                 0.000000
       ...  ...                  ...                 ...
192           0                    0                 0.000000
629           0                    0                 0.000000
559           0                    0                 0.000000
291           0                    0                 0.000000
1162          0                    0                 0.000000

[1176 rows x 3 columns]
```

```
y_test_predictions_decision_tree = model_decision_predict(X_test)
y_test_pd
```

```
Attrition  Attrition_Predicted  Prob_Attrition_score_dt
1247          0                    0                    0.000000
768           0                    0                    0.000000
918           0                    0                    0.000000
1070          1                    1                    1.000000
923           0                    0                    0.000000
       ...  ...                  ...                    ...
192           0                    0                    0.000000
629           0                    0                    0.000000
559           0                    0                    0.000000
291           0                    0                    0.000000
1162          0                    0                    0.000000

[1176 rows x 3 columns]
```

```
y_test_predictions_decision_tree[['Attrition_Predicted', 'Prob_Attrition_score']]
```

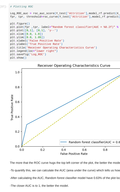
```
Attrition  Prob_Attrition_score_dt
1247          0                    0.000000
768           0                    0.000000
918           0                    0.000000
1070          1                    1.000000
923           0                    0.000000
       ...  ...                    ...
192           0                    0.000000
629           0                    0.000000
559           0                    0.000000
291           0                    0.000000
1162          0                    0.000000

[1176 rows x 3 columns]
```

```
y_test_pd['Attrition_Predicted', 'Prob_Attrition_score_dt']
y_test_pd.\*\*
```

```
# Importing the
!wg_REP_mod = rfc_pd_search_best('Attrition', model_dt_predict(X_test))
!wg_REP_mod = rfc_pd_search_best('Attrition', model_dt_predict(X_test))

plt.figure()
plt.plot(fpr, tpr, label='Decision tree classifier( = %f)' % roc_auc)
plt.plot([0, 1], [0, 1], '--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristics Curve')
plt.legend(loc='lower right')
plt.savefig('log_REP')
plt.show()
```



The more that the ROC curve hugs the top left corner of the plot, the better the model does at classifying the data into categories.

To quantify this, we can calculate the AUC (area under the curve) which tells us how much of the plot is located under the curve.

After calculating the AUC, Decision tree classifier model have 0.69% of the plot located under the curve.

The closer AUC is to 1, the better the model.

Building Decision tree model

```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
# Fitting the model on training data
model_dt = dt.fit(X_train,y_train)
# Predicting the values on test data
y_test_predictions_decision_tree = model_decision_predict(X_test)
y_test
```

```
Attrition  Attrition_Predicted  Prob_Attrition_score
1247          0                    0                 0.000000
768           0                    0                 0.000000
918           0                    0                 0.000000
1070          1                    1                 1.000000
923           0                    0                 0.000000
       ...  ...                  ...                 ...
192           0                    0                 0.000000
629           0                    0                 0.000000
559           0                    0                 0.000000
291           0                    0                 0.000000
1162          0                    0                 0.000000

[1176 rows x 3 columns]
```

```
y_test_predictions_decision_tree = model_decision_predict(X_test)
y_test_pd
```

```
Attrition  Attrition_Predicted  Prob_Attrition_score_dt
1247          0                    0                    0.000000
768           0                    0                    0.000000
918           0                    0                    0.000000
1070          1                    1                    1.000000
923           0                    0                    0.000000
       ...  ...                  ...                    ...
192           0                    0                    0.000000
629           0                    0                    0.000000
559           0                    0                    0.000000
291           0                    0                    0.000000
1162          0                    0                    0.000000

[1176 rows x 3 columns]
```

```
y_test_predictions_decision_tree[['Attrition_Predicted', 'Prob_Attrition_score']]
```

```
Attrition  Prob_Attrition_score_dt
1247          0                    0.000000
768           0                    0.000000
918           0                    0.000000
1070          1                    1.000000
923           0                    0.000000
       ...  ...                    ...
192           0                    0.000000
629           0                    0.000000
559           0                    0.000000
291           0                    0.000000
1162          0                    0.000000

[1176 rows x 3 columns]
```

```
y_test_pd['Attrition_Predicted', 'Prob_Attrition_score_dt']
y_test_pd.\*\*
```

```
# Importing the
!wg_REP_mod = rfc_pd_search_best('Attrition', model_dt_predict(X_test))
!wg_REP_mod = rfc_pd_search_best('Attrition', model_dt_predict(X_test))

plt.figure()
plt.plot(fpr, tpr, label='Random Forest classifier( = %f)' % roc_auc)
plt.plot([0, 1], [0, 1], '--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristics Curve')
plt.legend(loc='lower right')
plt.savefig('log_REP')
plt.show()
```


The more that the ROC curve hugs the top left corner of the plot, the better the model does at classifying the data into categories.

To quantify this, we can calculate the AUC (area under the curve) which tells us how much of the plot is located under the curve.

After calculating the AUC, Random forest classifier model have 0.69% of the plot located under the curve.

The closer AUC is to 1, the better the model.