Tên: PHẠM QUỲNH HƯƠNG

Lớp: K19414C

MSSV: K194141724

CREDIT RISK MODEL

1. Tạo đường dẫn

```
[1]  from google.colab import drive
     drive.mount('/content/gdrive', force_remount=True)
     path = "gdrive/My Drive/Credit Risk Model Subject/Data"
     # mount vào google drive thư mục có chứa data
```

- Đường dẫn: My drive/Credit Risk Model Subject/Data

2. Truy cập vào file data

```
[2]  import os
     path = "gdrive/My Drive/Credit Risk Model Subject/Data"
     print(os.path.isdir(path))
     print(os.path.isfile(path + "/khach_hang_ca_nhan.csv"))
     #print(os.path.isfile(path + "/german_credit_categori.csv"))
     # kiểm tra xem đường dẫn có tồn tại hay không
```

- File data “khach_hang_ca_nhan.csv”

3. Tải thư viện, nhập data

```
[3]  import pandas as pd #thư viện làm việc với data frame
     import seaborn as sns #vẽ hình
     import numpy as np # thư viện làm việc với con số
     import matplotlib.pyplot as plt #vẽ hình

     data = pd.read_csv(path + "/khach_hang_ca_nhan.csv") # đọc dữ liệu từ đương dẫn phía trên
      # .read_csv
      # .read_excel
```

- Đọc data set và lưu dưới dạng dataframe “data”.

4. In ra dữ liệu

```
[4] data
```

	default	account_check_status	duration_in_month	credit_history	purpose	credit_amount	savings	prese
0	0	1	6	4	4	1169	5	
1	1	2	48	2	4	5951	1	
2	0	4	12	4	7	2096	1	
3	0	1	42	2	3	7882	1	
4	1	1	24	3	0	4870	1	
...	...	...	...	...	...	...	...	
995	0	4	12	2	3	1736	1	
996	0	1	30	2	1	3857	1	
997	0	4	12	2	4	804	1	
998	1	1	45	2	4	1845	1	
999	0	2	45	4	1	4576	2	

1000 rows × 21 columns

- Dữ liệu có 1000 quan sát, 20 thuộc tính (biến độc lập) và 1 biến Target (biến phụ thuộc).

- Biến Target: 0 (No default); 1 (Default)

5. Copy bộ dữ liệu sang dataframe mới "data_2"

```
[5] from copy import deepcopy
    data_2 = deepcopy(data)
```

- Copy data set vào dataframe mới "data_2".

6. Xuất thông tin bộ dữ liệu (Tên thuộc tính, Số lượng quan sát không bị rỗng (NaN), kiểu dữ liệu từng thuộc tính)

```
data_2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 21 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   default                     1000 non-null   int64
 1   account_check_status        1000 non-null   int64
 2   duration_in_month           1000 non-null   int64
 3   credit_history              1000 non-null   int64
 4   purpose                     1000 non-null   int64
 5   credit_amount               1000 non-null   int64
 6   savings                     1000 non-null   int64
 7   present_emp_since           1000 non-null   int64
 8   installment_as_income_perc  1000 non-null   int64
 9   personal_status_sex         1000 non-null   int64
 10  other_debtors               1000 non-null   int64
 11  present_res_since           1000 non-null   int64
 12  property                    1000 non-null   int64
 13  age                         1000 non-null   int64
 14  other_installment_plans     1000 non-null   int64
 15  housing                     1000 non-null   int64
 16  credits_this_bank           1000 non-null   int64
 17  job                         1000 non-null   int64
 18  people_under_maintenance    1000 non-null   int64
 19  telephone                   1000 non-null   int64
 20  foreign_worker              1000 non-null   int64
dtypes: int64(21)
memory usage: 164.2 KB
```

- Tên thuộc tính:

+ *account_check_status*: số dư tài khoản

1 (<0 DM); 2 (0<=…<200 DM); 3 (>=200 DM); 4 (không có tài khoản).

+ *duration_in_month*: thời hạn hoàn trả (tính theo tháng)

+ *credit_history*: lịch sử tín dụng

0 (chưa ghi nhân khoản tín dụng nào); 1 (khách hàng hoàn trả đầy đủ); 2 (khách hàng đang vay tín dụng và hoàn trả hợp lệ cho đến nay); 3 (đã từng thanh toán chậm trễ trong quá khứ); 4 (có tài khoản quan trọng khác / các khoản tín dụng khác hiện có tại các ngân hàng khác); 5 (có nợ xấu ở ngân hàng khác).

+ *purpose*: mục đích vay tín dụng

0 (mua xe ô tô mới); 1 (mua xe ô tô cũ); 2 (mua thiết bị / nội thất); 3 (mua ti vi / radio); 4 (mua đồ gia dụng); 5 (sửa nhà); 6 (học phí / du học); 7 (du lịch); 8 (khóa đào tạo lại); 9 (kinh doanh); 10 (khác).

+ *credit_amount*: khoản tính dụng

+ *savings*: saving accounts / bonds

1 (< 100 DM); 2 (100 <=…<500 DM); 3 (500 <=…<1000 DM); 4 (>=1000 DM); 5 (không có / không biết).

+ *present_emp_since*: thời gian làm công việc hiện tại tính đến nay

1 (thất nghiệp); 2 (<1 năm); 3 (1<=…<4 năm); 4 (4<=…<7 năm); 5 (>= 7 năm).

+ *installment_as_income_perc*: tỷ lệ trả góp theo phần trăm thu nhập khả dụng

+ *personal_status_sex*: tình trạng hôn nhân và giới tính

1 (nam: ly hôn / ly thân); 2 (nữ: ly hôn / ly thân / kết hôn); 3 (nam: độc thân); 4 (nam: kết hôn / góa vợ); 5 (nữ: độc thân).

+ *other_debtors*: người đồng vay tín dụng / người bảo lãnh

1 (không có); 2 (có người đồng ký vay tín dụng); 3 (có người bảo lãnh).

+ *present_res_since*: thời gian cư trú tại nơi ở hiện tại tính đến nay

+ *property*: tài sản bảo đảm

1 (có nhà / đất); 2 (nếu không có nhà / đất thì có bảo hiểm nhân thọ / hợp đồng tiết kiệm nhà ở xã hội); 3 (nếu không có nhà / đất / bảo hiểm nhân thọ / hợp đồng tiết kiệm nhà ở xã hội thì có xe ô tô hoặc những tài sản bảo đảm khác, ngoại trừ các tài sản được liệt kê ở mục *savings (saving accounts / bonds*); 4 (không có / không biết).

+ *age*: tuổi (tính bằng năm)

+ *other_installment_plans*: những kế hoạch trả góp khác

1 (ngân hàng); 2 (cửa hàng); 3 (không có).

+ *housing*: hình thức sở hữu nhà ở

1 (thuê nhà ở); 2 (sở hữu nhà ở); 3 (được ở miễn phí, không sở hữu).

+ *credits_this_bank*: số lượng các khoản tín dụng hiện tại ở ngân hàng này

+ *job*: công việc

1 (thất nghiệp / không có tay nghề / không chính thức); 2 (không có tay nghề/ không chính thức); 3 (nhân viên có tay nghề / chính thức); 4 (quản lý / tự kinh doanh / nhân viên có tay nghề cao / nhân viên văn phòng).

+ *people_under_maintenance*: số lượng người phụ thuộc

+ *telephone*: điện thoại di động

1 (không); 2 (có, đăng ký dưới tên khách hàng).

+ *foreign_worker*: người lao động ngoại quốc

1 (có); 2(không).

- Tên biến Target:

+ *default*: gian lận

0 (không gian lận); 1 (có gian lận).

- Kết quả thể hiện 21 biến không chứa missing value (NaN), kiểu dữ liệu interger (số nguyên).

- Các biến numerical (số): *duration_in_month, credit_amount, installment_as_income_perc, present_res_since, age, credits_this_bank, people_under_maintenance.*

- Các biến categorical (phân loại): *account_check_status, credit_history, purpose, savings, present_emp_since, personal_status_sex, other_debtors, property, other_installment_plans, housing, job, telephone, foreign_worker, default.*

- Kiểu dữ liệu của toàn bộ thuộc tính: integer.

7. Đổi kiểu dữ liệu theo tính chất thuộc tính

+ Biến numerical (số): integer.

+ Biến categorical (phân loại): object.

```
[ ] for col in categorical_features:
        data_2[col] = data_2[col].astype('object')
```

```
[ ] data_2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 21 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   default                    1000 non-null   int64
 1   account_check_status       1000 non-null   object
 2   duration_in_month          1000 non-null   int64
 3   credit_history             1000 non-null   object
 4   purpose                    1000 non-null   object
 5   credit_amount              1000 non-null   int64
 6   savings                    1000 non-null   object
 7   present_emp_since          1000 non-null   object
 8   installment_as_income_perc 1000 non-null   int64
 9   personal_status_sex        1000 non-null   object
 10  other_debtors              1000 non-null   object
 11  present_res_since          1000 non-null   int64
 12  property                   1000 non-null   object
 13  age                        1000 non-null   int64
 14  other_installment_plans    1000 non-null   object
 15  housing                    1000 non-null   object
 16  credits_this_bank          1000 non-null   int64
 17  job                        1000 non-null   object
 18  people_under_maintenance   1000 non-null   int64
 19  telephone                  1000 non-null   object
 20  foreign_worker             1000 non-null   object
dtypes: int64(8), object(13)
memory usage: 164.2+ KB
```

8. Thống kê mô tả

8.1. Các biến numerical

```
[ ]  data_2[numerical_features].describe()
```

	duration_in_month	credit_amount	installment_as_income_perc	present_res_since	age	credits_this_bank	people_under_maintenance
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000
mean	20.903000	3271.258000	2.973000	2.845000	35.546000	1.407000	1.155000
std	12.058814	2822.736876	1.118715	1.103718	11.375469	0.577654	0.362086
min	4.000000	250.000000	1.000000	1.000000	19.000000	1.000000	1.000000
25%	12.000000	1365.500000	2.000000	2.000000	27.000000	1.000000	1.000000
50%	18.000000	2319.500000	3.000000	3.000000	33.000000	1.000000	1.000000
75%	24.000000	3972.250000	4.000000	4.000000	42.000000	2.000000	1.000000
max	72.000000	18424.000000	4.000000	4.000000	75.000000	4.000000	2.000000

- Quan sát bảng thống kê mô tả trên, ta thấy:

+ Biến *duration_in_month* có giá trị nhỏ nhất là 4 tháng và giá trị lớn nhất là 72 tháng (6 năm). Giá trị trung bình là 20.9 tháng.

+ Biến *credit_amount* có giá trị nhỏ nhất là 250 DM và giá trị lớn nhất là 18424 DM. Giá trị trung bình là 3271.258 DM.

Tương tự cho 5 biến tiếp theo.

8.2. Các biến categorical

```
[ ]  data_2[categorical_features].describe()
```

	account_check_status	credit_history	purpose	savings	present_emp_since	personal_status_sex	other_debtors	property	other_installment_plans	housing	job	telephone	foreign_worker
count	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
unique	4	5	10	5	5	4	3	4	3	3	4	2	2
top	4	2	4	1	3	3	1	3	3	2	3	1	1
freq	394	530	280	603	339	548	907	332	814	713	630	596	963

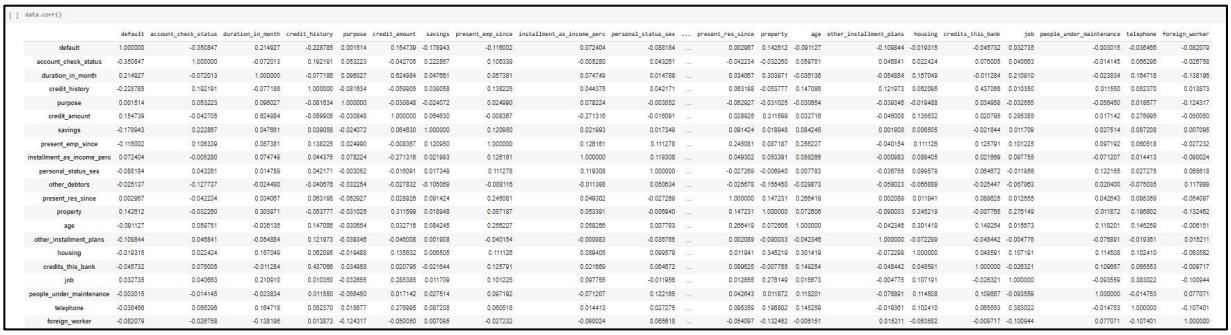
- Quan sát bảng thống kê mô tả trên, ta thấy:

+ Biến *account_check_status* có 4 yếu tố phân loại riêng biệt, yếu tố số (4) được phân loại nhiều nhất với tần suất là 394 lần trên tổng số 1000 quan sát (394 / 1000).

+ Biến *credit_history* có 5 yếu tố phân loại riêng biệt, yếu tố số (2) được phân loại nhiều nhất với tần suất là 530 lần trên tổng số 1000 quan sát (530 / 1000).

Tương tự cho 11 biến còn lại.

9. Ma trận tương quan giữa các biến và giữa biến *default* với các biến độc lập



```
[ ] pd.DataFrame(data.corr().iloc[1:,0])
```

	default
account_check_status	-0.350847
duration_in_month	0.214927
credit_history	-0.228785
purpose	0.001514
credit_amount	0.154739
savings	-0.178943
present_emp_since	-0.116002
installment_as_income_perc	0.072404
personal_status_sex	-0.088184
other_debtors	-0.025137
present_res_since	0.002967
property	0.142612
age	-0.091127
other_installment_plans	-0.109844
housing	-0.019315
credits_this_bank	-0.045732
job	0.032735
people_under_maintenance	-0.003015
telephone	-0.036466
foreign_worker	-0.082079

- Quan sát bảng sự tương quan (mức độ quan trọng) giữa các biến độc lập đối với biến *default*, ta thấy:

+ Có 13 biến tương quan nghịch (âm) với *default.* Khi biến độc lập tăng 1 giá trị thì biến *default* giảm tỷ lệ gian lận.Và ngược lại.

+ Có 7 biến tương quan thuận (dương) với *default.* Khi biến độc lập tăng 1 giá trị thì biến *default* tăng tỷ lệ gian lận.Và ngược lại.

- Sắp xếp theo mức độ giảm dần các biến có tương quan (tác động) đến *default:*

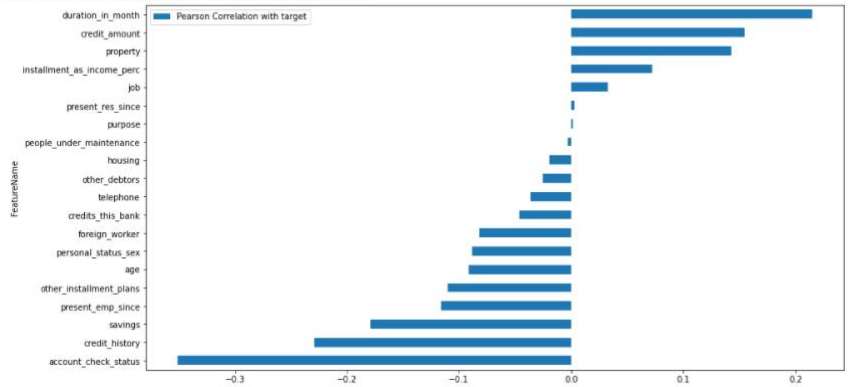
```
[ ] features_importances_corr = pd.DataFrame({'FeatureName': data[column_selected].columns[0:len(data[column_selected])-1], 'Pearson Correlation with target': correlation})
    features_importances_corr.reindex(features_importances_corr['Pearson Correlation with target'].abs().sort_values(ascending=False).index)
```

	FeatureName	Pearson Correlation with target
0	account_check_status	-0.350847
2	credit_history	-0.228785
1	duration_in_month	0.214927
5	savings	-0.178943
4	credit_amount	0.154739
11	property	0.142612
6	present_emp_since	-0.116002
13	other_installment_plans	-0.109844
12	age	-0.091127
8	personal_status_sex	-0.088184
19	foreign_worker	-0.082079
7	installment_as_income_perc	0.072404
15	credits_this_bank	-0.045732
18	telephone	-0.036466
16	job	0.032735
9	other_debtors	-0.025137
14	housing	-0.019315
17	people_under_maintenance	-0.003015
10	present_res_since	0.002967
3	purpose	0.001514

- Được thể hiện ở biểu đồ sau:

```
[ ] features_importances_corr.sort_values(by = ['Pearson Correlation with target']).plot(figsize=(15,8), x="FeatureName", y = "Pearson Correlation with target", kind="barh")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f71fff6fd10>
```



=> Biến *account_check_status* có tương quan cao nhất (35.08%) đến kết quả dự báo biến *default* và biến *purpose* có tương quan thấp nhất (0.15%).

- Tiếp theo ta tiến hành chạy mô hình và dự báo bằng 5 thuật toán khác nhau: Random Forest, Logistic Regression, Decision Tree Classifier, XGB Classifier, Ada Boost Classifier.

10. Random Forest model

10.1. Import thư viện

```
[ ] from sklearn.ensemble import RandomForestClassifier
    from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, roc_auc_score
    from sklearn.model_selection import cross_val_score
    from sklearn.model_selection import train_test_split
```

10.2. Tách training set - test set theo tỷ lệ 90% - 10%

```
[ ] X = data[features].values
    y = data[target].values
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.1, random_state=42)
```

- Random_state: dùng để cố định sự phân tách training set và test set, dễ dàng chạy lại mô hình nhiều lần nhưng vẫn giữ nguyên kết quả ban đầu.

10.3. Chạy mô hình và dự báo

```
[ ] RF_classifier = RandomForestClassifier()
    RF_classifier.fit(X_train, y_train.ravel())

    y_pred = RF_classifier.predict(X_test)
    print(confusion_matrix(y_test,y_pred))
    print(classification_report(y_test,y_pred))
    print('Random Forest accuracy: ', accuracy_score(y_test, y_pred))

    [[66  5]
     [11 18]]
                  precision    recall  f1-score   support

               0       0.86      0.93      0.89        71
               1       0.78      0.62      0.69        29

        accuracy                           0.84       100
       macro avg       0.82      0.78      0.79       100
    weighted avg       0.84      0.84      0.83       100

    Random Forest accuracy:  0.84
```

- Mô hình được xây dựng dựa trên tập training set (chiếm 90% bộ dữ liệu ban đầu "data_2").

- Mô hình được dự báo dựa trên tập test set (chiếm 10% bộ dữ liệu ban đầu).

- Ma trận hỗn loạn (Confusion Matrix) có giá trị:

	Predicted 0	Predicted 1
Actual 0	66	5
Actual 1	11	18

cho thấy có 66 quan sát được dự báo 0 (No default) và thật sự là 0; có 11 quan sát được dự báo 0 nhưng sự thật là 1 (Default); có 5 quan sát được dự báo 1 nhưng sự thật là 0; và có 18 quan sát được dự báo 1 và thật sự là 1.

- Classification report (báo cáo phân loại) cho thấy:

+ Có 71 quan sát actual 0; 29 quan sát actual 1. Tổng cộng 100 quan sát được dự báo.

+ Accuracy của mô hình là: 84%.

10.4. Mức độ quan trọng của các biến độc lập đối với mô hình Random Forest

```
[ ] importance_rf = RF_classifier.feature_importances_
    features_importances_rf = pd.DataFrame({'FeatureName': data.columns[0:len(data.columns)-1], 'Random Forest Feature Importance': importance_rf})
    features_importances_rf.sort_values(by=['Random Forest Feature Importance'], ascending=False)
```

	FeatureName	Random Forest Feature Importance
18	people_under_maintenance	0.141428
14	other_installment_plans	0.106183
10	other_debtors	0.101354
11	present_res_since	0.096770
6	savings	0.064195
3	credit_history	0.060375
9	personal_status_sex	0.054065
12	property	0.044498
7	present_emp_since	0.043348
17	job	0.043081
19	telephone	0.041328
8	installment_as_income_perc	0.039076
16	credits_this_bank	0.034532
0	default	0.026664
15	housing	0.025520
1	account_check_status	0.023294
4	purpose	0.018392
2	duration_in_month	0.018136
5	credit_amount	0.014052
13	age	0.003709

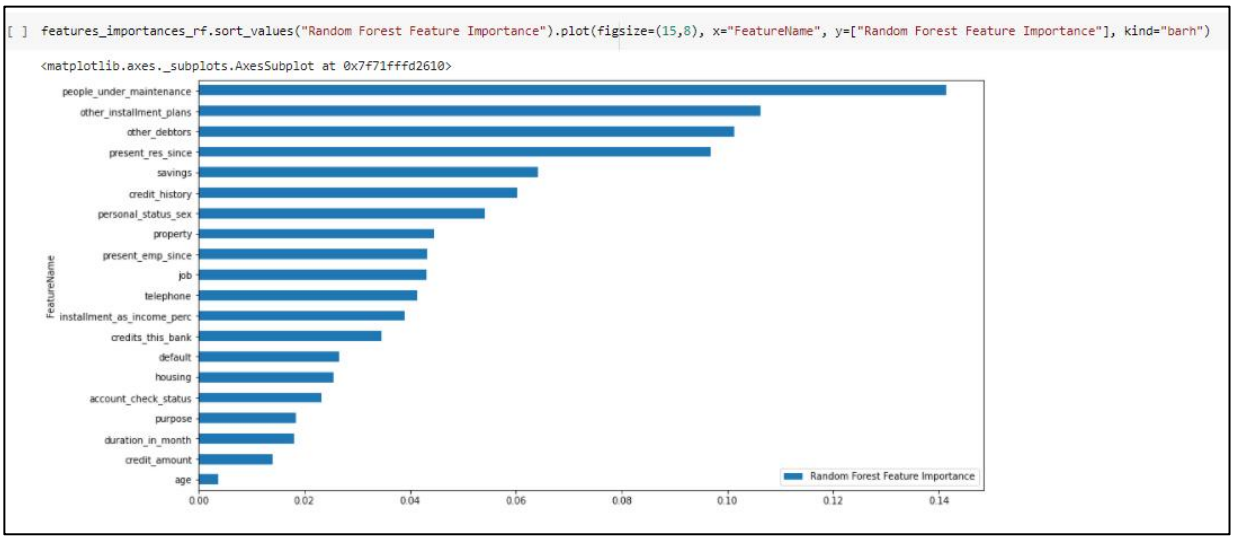
- Đối với mô hình Random Forest, bảng xếp hạng mức độ quan trọng của các biến độc lập lên dự báo như sau:

+ Biến *people_under_maintenance*: có mức độ tác động mạnh nhất (14.14%).

+ Biến *age:* có mức độc tác động yếu nhất (0.37%).

- Trực quan hóa như sau:

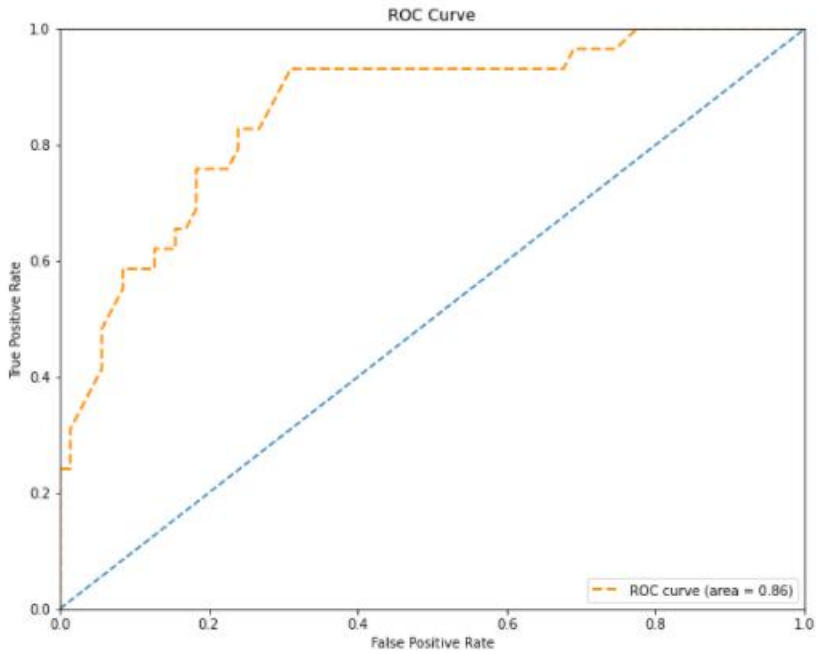
```
[ ] features_importances_rf.sort_values("Random Forest Feature Importance").plot(figsize=(15,8), x="FeatureName", y=["Random Forest Feature Importance"], kind="barh")
```



10.5. Biểu đồ ROC curve (Receiver Operating Characteristic curve)

```
[ ] y_pred_prob_test = RF_classifier.predict_proba(X_test)[:, 1]
    fpr, tpr, thres = roc_curve(y_test, y_pred_prob_test)
    roc_auc = auc(fpr, tpr)

    _plot_roc_curve(fpr, tpr, thres, roc_auc)
```



- AUC score của mô hình bằng 86%.

11. Logistic Regression model

11.1. Chạy mô hình và xuất kết quả dự báo

```
[ ] from sklearn.linear_model import LogisticRegression
    LR_classifier = LogisticRegression()
    LR_classifier.fit(X_train, y_train.ravel())

    y_pred = LR_classifier.predict(X_test)
    print(confusion_matrix(y_test,y_pred))
    print(classification_report(y_test,y_pred))
    print('Logistic Regression accuracy: ', accuracy_score(y_test, y_pred))
```

```
[[62  9]
 [14 15]]
              precision    recall  f1-score   support

           0       0.82      0.87      0.84        71
           1       0.62      0.52      0.57        29

    accuracy                           0.77       100
   macro avg       0.72      0.70      0.70       100
weighted avg       0.76      0.77      0.76       100
```

- Chạy mô hình trên tập training set.

- Dự báo mô hình trên tập test set.

- In bảng Confusion Matrix (ma trận hỗn loạn) có giá trị:

	Predicted 0	Predicted 1
Actual 0	62	9
Actual 1	14	15

cho thấy có 62 quan sát được dự báo 0 (No default) và thật sự là 0; có 14 quan sát được dự báo 0 nhưng sự thật là 1 (Default); có 9 quan sát được dự báo 1 nhưng sự thật là 0; và có 15 quan sát được dự báo 1 và thật sự là 1.

- Classification report (báo cáo phân loại) cho thấy:

+ Có 71 quan sát actual 0; 29 quan sát actual 1. Tổng cộng 100 quan sát được dự báo.

+ Accuracy của mô hình là: 77%.

11.2. Mức độ quan trọng của các biến độc lập đối với mô hình Logistic Regression

```
[ ] importance_lr = LR_classifier.coef_[0] #use coeffcient as importance
    features_importances_lr = pd.DataFrame({'FeatureName': data.columns[0:len(data.columns)-1], 'Logistic Regression Feature Importance': importance_lr})
    # features_importances_lr.sort_values(by=['Logistic Regression Feature Importance'], ascending=False)
    features_importances_lr.reindex(features_importances_lr['Logistic Regression Feature Importance'].abs().sort_values(ascending=False).index)
```

	FeatureName	Logistic Regression Feature Importance
10	other_debtors	-0.516709
3	credit_history	-0.307699
12	property	0.272344
17	job	-0.230898
7	present_emp_since	0.227679
19	telephone	0.114367
9	personal_status_sex	-0.081836
15	housing	-0.072438
8	installment_as_income_perc	-0.069165
16	credits_this_bank	0.067041
5	credit_amount	0.048193
2	duration_in_month	-0.035193
1	account_check_status	0.034150
4	purpose	-0.032651
11	present_res_since	0.023146
6	savings	-0.013894
0	default	-0.012294
14	other_installment_plans	-0.011871
13	age	-0.001558
18	people_under_maintenance	0.000079

- Đối với mô hình Logistic Regression, bảng xếp hạng mức độ quan trọng của các biến độc lập lên dự báo như sau:

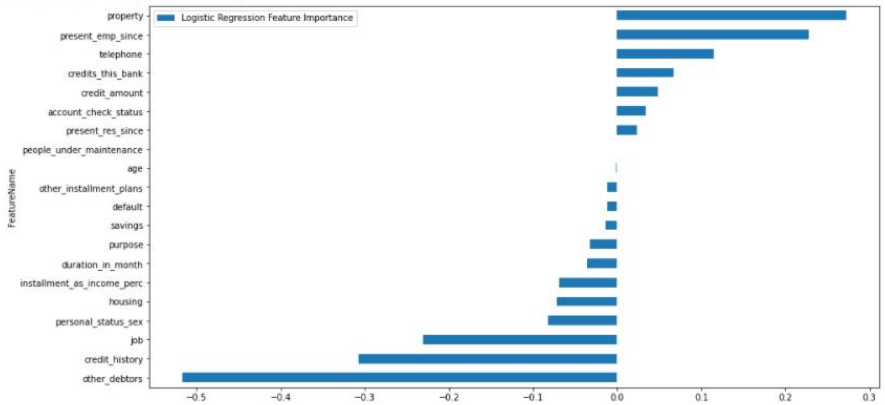
+ Biến *other_debtors*: có mức độ tác động mạnh nhất (51.67%).

+ Biến *people_under_maintenance:* có mức độc tác động yếu nhất (0.00%).

- Trực quan hóa mức độ quan trọng của các biến độc lập như sau:

```
[ ] features_importances_lr.sort_values("Logistic Regression Feature Importance").plot(figsize=(15,8), x="FeatureName", y=["Logistic Regression Feature Importance"], kind="barh")
```

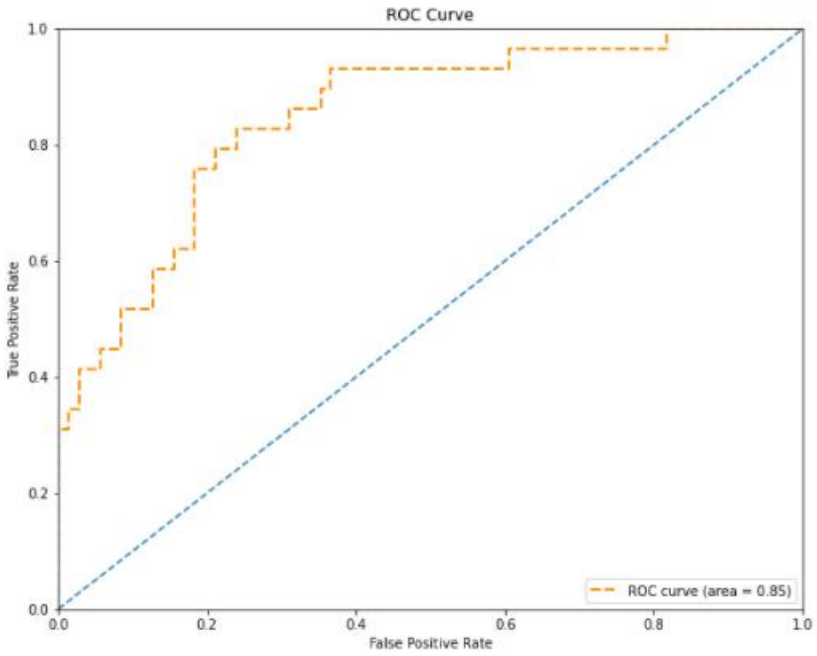
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f71ffe30f10>
```



11.3. Biểu đồ ROC curve (Receiver Operating Characteristic curve)

```
[ ] y_pred_prob_test = LR_classifier.predict_proba(X_test)[:, 1]
    fpr, tpr, thres = roc_curve(y_test, y_pred_prob_test)
    roc_auc = auc(fpr, tpr)

    _plot_roc_curve(fpr, tpr, thres, roc_auc)
```



- AUC của mô hình bằng 85%.

12. Decision Tree Classifier

12.1. Import thư viện, chạy mô hình và dự báo

```
[ ] from sklearn.tree import DecisionTreeClassifier
```

a. Prediction

```
[ ] DT_classifier = DecisionTreeClassifier()
    DT_classifier.fit(X_train, y_train.ravel())

    y_pred = DT_classifier.predict(X_test)
    print(confusion_matrix(y_test,y_pred))
    print(classification_report(y_test,y_pred))
    print('Decision Tree accuracy: ', accuracy_score(y_test, y_pred))
```

```
    [[57 14]
     [12 17]]
                  precision    recall  f1-score   support

               0       0.83      0.80      0.81        71
               1       0.55      0.59      0.57        29

        accuracy                           0.74       100
       macro avg       0.69      0.69      0.69       100
    weighted avg       0.75      0.74      0.74       100

    Decision Tree accuracy:  0.74
```

- Chạy mô hình trên tập training set.

- Dự báo mô hình trên tập test set.

- In bảng Confusion Matrix (ma trận hỗn loạn) có giá trị:

	Predicted 0	Predicted 1
Actual 0	57	14
Actual 1	12	17

cho thấy có 57 quan sát được dự báo 0 (No default) và thật sự là 0; có 12 quan sát được dự báo 0 nhưng sự thật là 1 (Default); có 14 quan sát được dự báo 1 nhưng sự thật là 0; và có 17 quan sát được dự báo 1 và thật sự là 1.

- Classification report (báo cáo phân loại) cho thấy:

+ Có 71 quan sát actual 0; 29 quan sát actual 1. Tổng cộng 100 quan sát được dự báo.

+ Accuracy của mô hình là: 74%.

12.2. Mức độ quan trọng của các biến độc lập đối với mô hình Decision Tree Classifier

```
[ ] importance_dt = DT_classifier.feature_importances_
    features_importances_dt = pd.DataFrame({'FeatureName': data.columns[0:len(data.columns)-1], 'Decision Tree Feature Importance': importance_dt})
    features_importances_dt.sort_values(by=['Decision Tree Feature Importance'], ascending=False)
```

	FeatureName	Decision Tree Feature Importance
18	people_under_maintenance	0.192852
14	other_installment_plans	0.127638
10	other_debtors	0.124167
11	present_res_since	0.087642
3	credit_history	0.064307
6	savings	0.053426
9	personal_status_sex	0.049282
19	telephone	0.044146
12	property	0.037065
16	credits_this_bank	0.035553
7	present_emp_since	0.034464
0	default	0.030738
17	job	0.027689
15	housing	0.021247
8	installment_as_income_perc	0.020634
4	purpose	0.019438
5	credit_amount	0.010175
1	account_check_status	0.008405
2	duration_in_month	0.007613
13	age	0.003520

- Đối với mô hình Decision Tree Classifier, bảng xếp hạng mức độ quan trọng của các biến độc lập lên dự báo như sau:

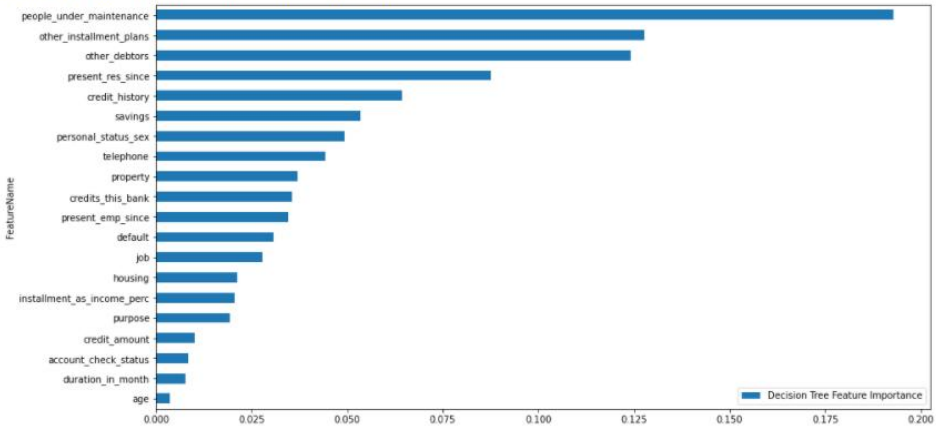
+ Biến *people_under_maintenance*: có mức độ tác động mạnh nhất (19.29%).

+ Biến *age:* có mức độc tác động yếu nhất (0.35%).

- Trực quan hóa mức độ quan trọng của các biến độc lập như sau:

```
[ ] features_importances_dt.sort_values("Decision Tree Feature Importance").plot(figsize=(15,8), x="FeatureName", y=["Decision Tree Feature Importance"], kind="barh")
```

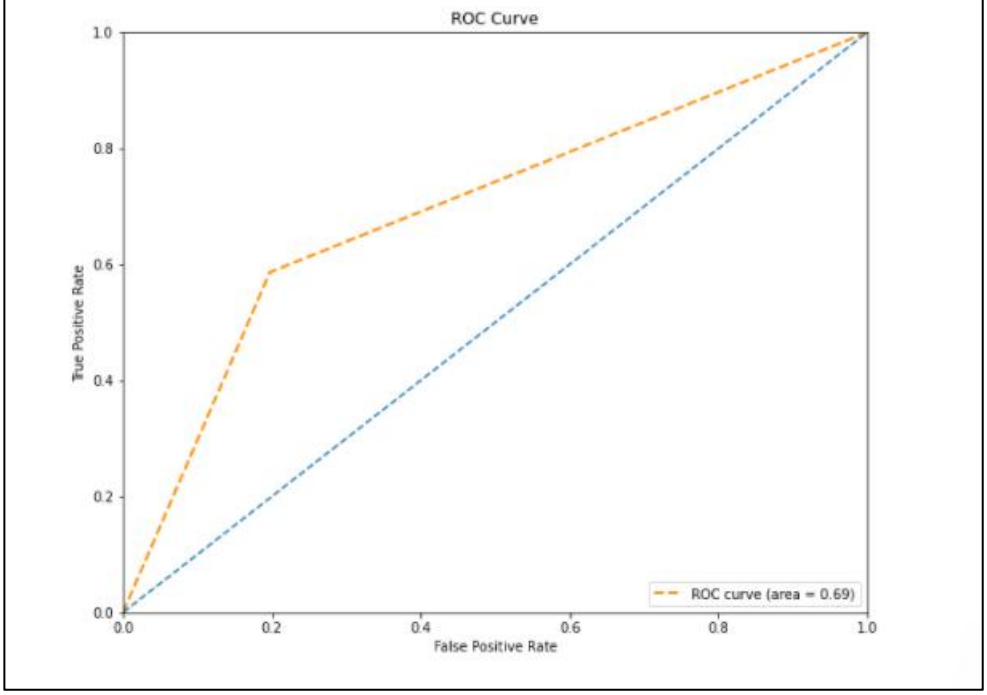
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f71fa898b50>
```



12.3. Biểu đồ ROC curve (Receiver Operating Characteristic curve)

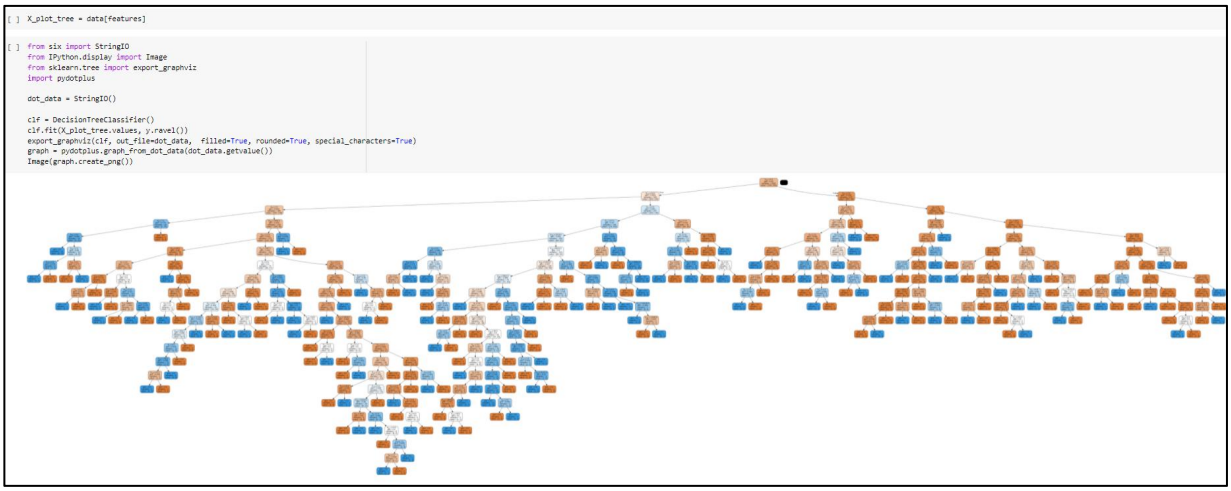
```
[ ]  y_pred_prob_test = DT_classifier.predict_proba(X_test)[:, 1]
     fpr, tpr, thres = roc_curve(y_test, y_pred_prob_test)
     roc_auc = auc(fpr, tpr)

     _plot_roc_curve(fpr, tpr, thres, roc_auc)
```



- AUC score của mô hình Decision Tree Classifier bằng 69%.

12.4. Trực quan hóa Decision Tree (cây quyết định)



- Trực quan hóa cây quyết định giúp quá trình giải thích mô hình và tìm ra quy luật dự báo một cách dễ dàng hơn.

13. XGB Classifier

13.1. Tải thư viện, chạy mô hình và dự báo

```
[ ] import xgboost as xgb
    from xgboost.sklearn import XGBClassifier
```

a. Prediction

```
[ ] XGB_classifier = XGBClassifier()
    XGB_classifier.fit(X_train, y_train.ravel())

    y_pred = XGB_classifier.predict(X_test)
    print(confusion_matrix(y_test,y_pred))
    print(classification_report(y_test,y_pred))
    print('XGBoost accuracy: ', accuracy_score(y_test, y_pred))
```

```
[[61 10]
 [11 18]]
              precision    recall  f1-score   support

           0       0.85      0.86      0.85        71
           1       0.64      0.62      0.63        29

    accuracy                           0.79       100
   macro avg       0.75      0.74      0.74       100
weighted avg       0.79      0.79      0.79       100

XGBoost accuracy:  0.79
```

- Chạy mô hình trên tập training set.

- Dự báo mô hình trên tập test set.

- In bảng Confusion Matrix (ma trận hỗn loạn) có giá trị:

	Predicted 0	Predicted 1
Actual 0	61	10
Actual 1	11	18

cho thấy có 61 quan sát được dự báo 0 (No default) và thật sự là 0; có 11 quan sát được dự báo 0 nhưng sự thật là 1 (Default); có 10 quan sát được dự báo 1 nhưng sự thật là 0; và có 18 quan sát được dự báo 1 và thật sự là 1.

- Classification report (báo cáo phân loại) cho thấy:

+ Có 71 quan sát actual 0; 29 quan sát actual 1. Tổng cộng 100 quan sát được dự báo.

+ Accuracy của mô hình là: 79%.

13.2. Mức độ quan trọng của các biến độc lập đối với mô hình XGB Classifier

```
[ ] importance_xgb = XGB_classifier.feature_importances_
    features_importances_xgb = pd.DataFrame({'FeatureName': data.columns[0:len(data.columns)-1], 'XGBoost Feature Importance': importance_xgb})
    features_importances_xgb.sort_values(by=['XGBoost Feature Importance'], ascending=False)
```

	FeatureName	XGBoost Feature Importance
10	other_debtors	0.204196
3	credit_history	0.070670
4	purpose	0.064103
7	present_emp_since	0.062391
17	job	0.060462
11	present_res_since	0.052110
9	personal_status_sex	0.048315
6	savings	0.044802
15	housing	0.044051
2	duration_in_month	0.042619
18	people_under_maintenance	0.041134
12	property	0.038650
14	other_installment_plans	0.037307
19	telephone	0.036189
1	account_check_status	0.035612
5	credit_amount	0.035233
8	installment_as_income_perc	0.027952
0	default	0.026582
13	age	0.015509
16	credits_this_bank	0.012114

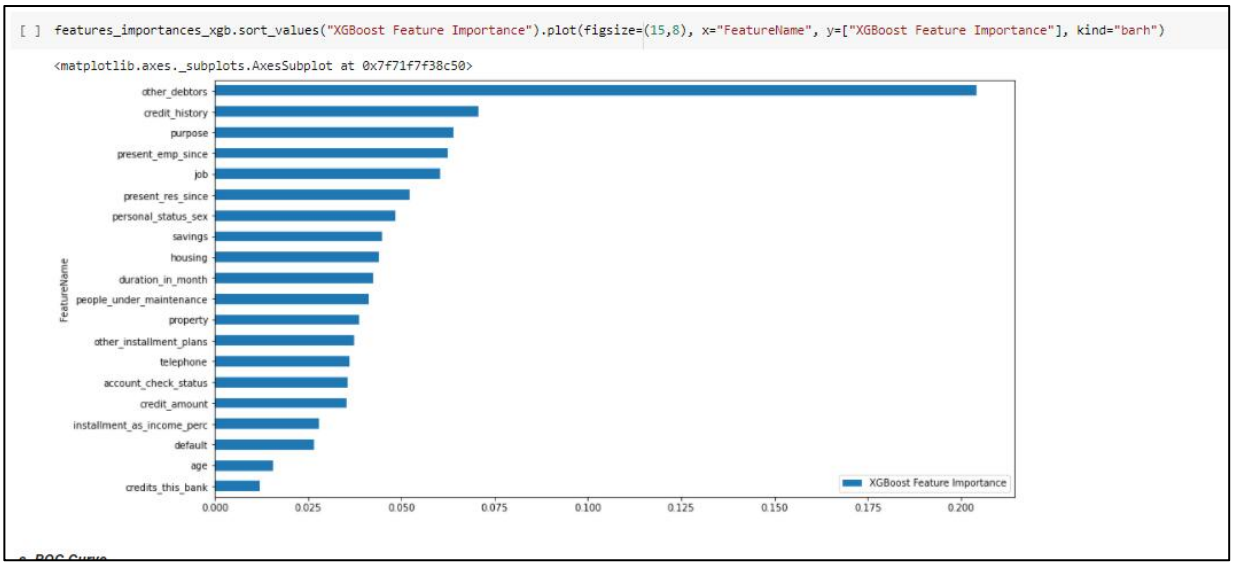
- Đối với mô hình XGB Classifier, bảng xếp hạng mức độ quan trọng của các biến độc lập lên dự báo như sau:

+ Biến *other_debtors*: có mức độ tác động mạnh nhất (20.42%).

+ Biến *credits_this_bank:* có mức độc tác động yếu nhất (1.21%).

- Trực quan hóa mức độ quan trọng của các biến độc lập như sau:

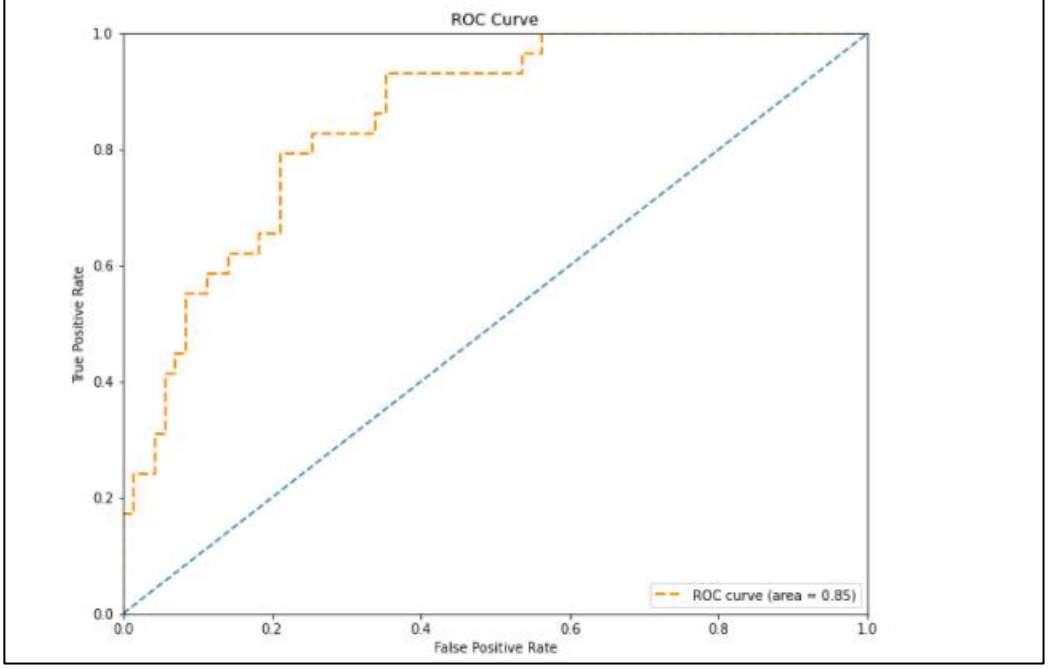
```
[ ] features_importances_xgb.sort_values("XGBoost Feature Importance").plot(figsize=(15,8), x="FeatureName", y=["XGBoost Feature Importance"], kind="barh")
```



13.3. Biểu đồ ROC curve (Receiver Operating Characteristic curve)

```
[ ] y_pred_prob_test = XGB_classifier.predict_proba(X_test)[:, 1]
    fpr, tpr, thres = roc_curve(y_test, y_pred_prob_test)
    roc_auc = auc(fpr, tpr)

    _plot_roc_curve(fpr, tpr, thres, roc_auc)
```



- AUC score của mô hình XGB Classifier bằng 85%.

14. Ada Boost Classifier

14.1. Tải thư viện, chạy mô hình và dự báo

```
[ ] from sklearn.ensemble import AdaBoostClassifier
```

a. Prediction

```
[ ] ada_classifier = AdaBoostClassifier()
    ada_classifier.fit(X_train, y_train.ravel())

    y_pred = ada_classifier.predict(X_test)
    print(confusion_matrix(y_test,y_pred))
    print(classification_report(y_test,y_pred))
    print('XGBoost accuracy: ', accuracy_score(y_test, y_pred))
```

```
[[59 12]
 [14 15]]
              precision    recall  f1-score   support

           0       0.81      0.83      0.82        71
           1       0.56      0.52      0.54        29

    accuracy                           0.74       100
   macro avg       0.68      0.67      0.68       100
weighted avg       0.73      0.74      0.74       100

XGBoost accuracy:  0.74
```

- Chạy mô hình trên tập training set.

- Dự báo mô hình trên tập test set.

- In bảng Confusion Matrix (ma trận hỗn loạn) có giá trị:

	Predicted 0	Predicted 1
Actual 0	59	12
Actual 1	14	15

cho thấy có 59 quan sát được dự báo 0 (No default) và thật sự là 0; có 14 quan sát được dự báo 0 nhưng sự thật là 1 (Default); có 12 quan sát được dự báo 1 nhưng sự thật là 0; và có 15 quan sát được dự báo 1 và thật sự là 1.

- Classification report (báo cáo phân loại) cho thấy:

+ Có 71 quan sát actual 0; 29 quan sát actual 1. Tổng cộng 100 quan sát được dự báo.

+ Accuracy của mô hình là: 74%.

14.2. Mức độ quan trọng của các biến độc lập đối với mô hình Ada Boost Classifier

```
[ ] importance_ada = ada_classifier.feature_importances_
    features_importances_ada = pd.DataFrame({'FeatureName': data.columns[0:len(data.columns)-1], 'AdaBoost Feature Importance': importance_ada})
    features_importances_ada.sort_values(by=['AdaBoost Feature Importance'], ascending=False)
```

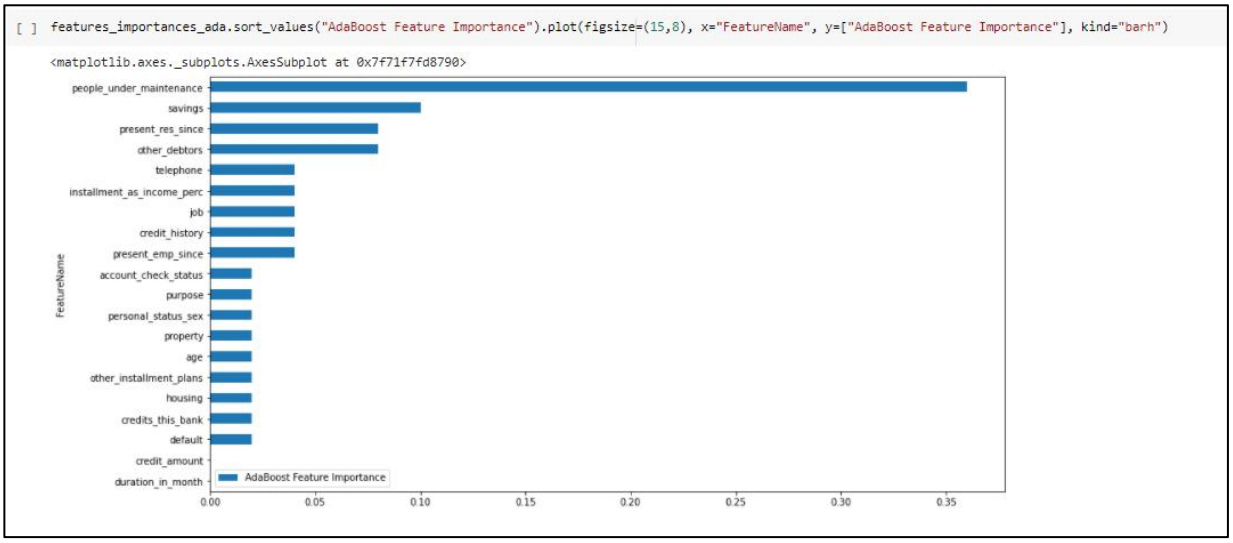
	FeatureName	AdaBoost Feature Importance
18	people_under_maintenance	0.36
6	savings	0.10
10	other_debtors	0.08
11	present_res_since	0.08
7	present_emp_since	0.04
17	job	0.04
8	installment_as_income_perc	0.04
19	telephone	0.04
3	credit_history	0.04
9	personal_status_sex	0.02
1	account_check_status	0.02
4	purpose	0.02
12	property	0.02
13	age	0.02
14	other_installment_plans	0.02
15	housing	0.02
16	credits_this_bank	0.02
0	default	0.02
5	credit_amount	0.00
2	duration_in_month	0.00

- Đối với mô hình Ada Boost Classifier, bảng xếp hạng mức độ quan trọng của các biến độc lập lên dự báo như sau:

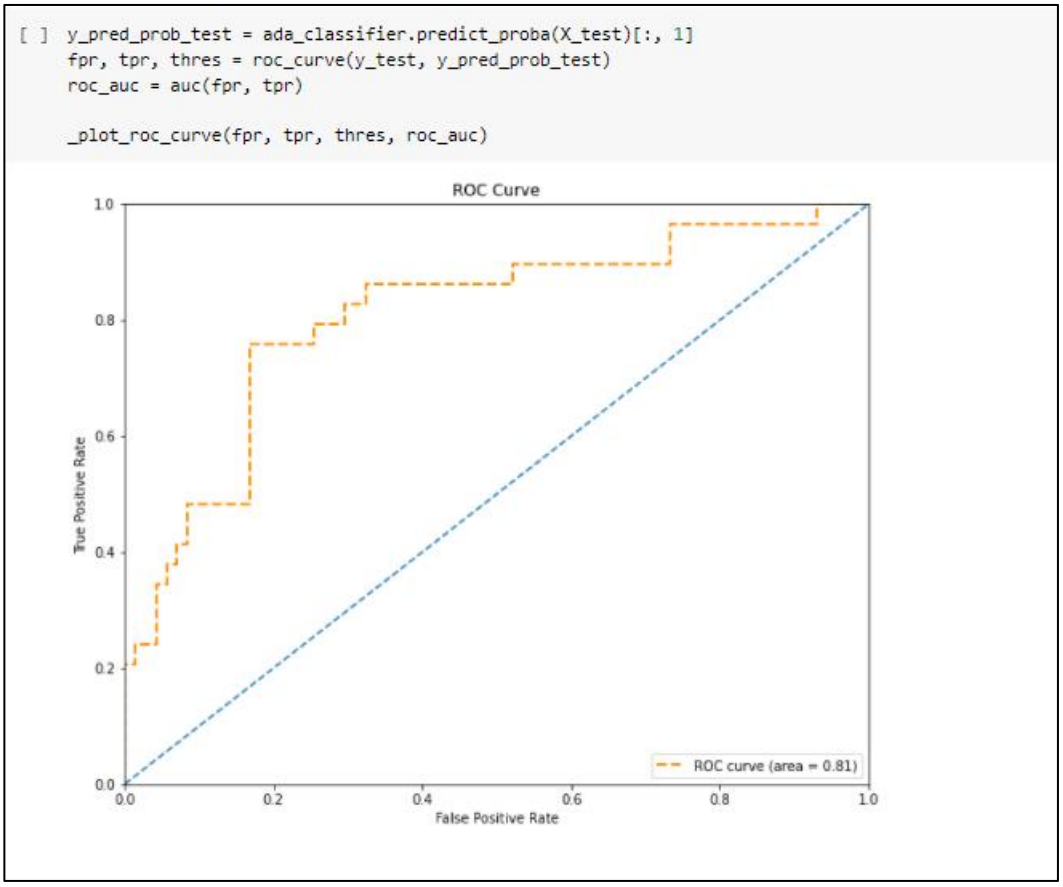
+ Biến *people_under_maintenance*: có mức độ tác động mạnh nhất (36%).

+ Biến *duration_in_month:* có mức độc tác động yếu nhất (0%).

- Trực quan hóa mức độ quan trọng của các biến độc lập như sau:



14.3. Biểu đồ ROC curve (Receiver Operating Characteristic curve)



- AUC score của mô hình Ada Boost Classifier bằng 81%.

15. Feature selected model (Mô hình được lựa chọn thuộc tính)