TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP TP HỒ CHÍ MINH
KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN CUỐI KÌ

MACHINE LEARNING

Người thực hiện: **NGUYỄN TẤN ĐẠT– 21022831**

Lớp học phần **: DHKHMT16A - 420300371104**

Người hướng dẫn: Ths. Hồ Đắc Quán

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP TP HỒ CHÍ MINH
KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN CUỐI KÌ

MACHINE LEARNING

Người thực hiện: **NGUYỄN TẤN ĐẠT– 21022831**

Lớp học phần **: DHKHMT16A - 420300371104**

Người hướng dẫn: Ths. Hồ Đắc Quán

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

LỜI CẢM ƠN

Tôi xin bày tỏ lòng biết ơn đến sự hướng dẫn, chỉ dẫn, và hỗ trợ của Ths.Hồ Đắc Quán. đã cung cấp trong suốt quá trình thực hiện đồ án này. Những lời khuyên, ý kiến, và kiến thức mà thầy đã chia sẻ đã giúp tôi hoàn thành công việc một cách hiệu quả hơn.

Tôi cũng muốn bày tỏ lòng biết ơn đến tất cả những người đã đóng góp ý kiến, ý tưởng, và hỗ trợ trong dự án này. Sự đoàn kết và cống hiến của mọi người đã làm cho dự án trở thành một thành tựu đáng tự hào.

Dự án này đã mang lại cho tôi rất nhiều kiến thức, kỹ năng, và trải nghiệm quý báu. Tôi hy vọng rằng nó cũng sẽ có lợi ích và giá trị đối với mọi người như nó đã mang lại cho tôi.

Xin chân thành cảm ơn mọi người một lần nữa.

Trân trọng, Dat

ĐỒ ÁN ĐƯỢC HOÀN THÀNH
TẠI TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP TP HỒ CHÍ MINH

Tôi xin cam đoan đây là sản phẩm đồ án của riêng tôi và được sự hướng dẫn của Ths.Hồ Đắc Quán. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình. Trường đại học Công nghiệp TP Hồ Chí Minh không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày 30 tháng 10 năm 2023

Tác giả

Dat

(ký tên và ghi rõ họ tên)

Nguyễn Tấn Đạt

PHẦN ĐÁNH GIÁ CỦA GIẢNG VIÊN

_____

_____

_____

_____

_____

_____

_____

Tp. Hồ Chí Minh, ngày tháng năm
(kí và ghi họ tên)

TÓM TẮT

Ngân hàng đóng một vai trò quan trọng trong nền kinh tế thị trường. Chúng quyết định ai có thể nhận được tài chính và dưới những điều kiện gì, và có thể hỗ trợ hoặc phá vỡ các quyết định đầu tư. Để thị trường và xã hội hoạt động hiệu quả, cá nhân và doanh nghiệp cần có quyền tiếp cận tín dụng.

Thuật toán đánh giá tín dụng, dự đoán xác suất mặc định, là phương pháp mà ngân hàng sử dụng để quyết định việc cấp hoặc không cấp khoản vay. Bài toán này yêu cầu cải thiện tiêu chuẩn hiện hành trong việc đánh giá tín dụng, bằng cách dự đoán xác suất một người sẽ gặp khó khăn tài chính trong hai năm tới.

Mục tiêu của đề tài là xây dựng một mô hình giúp người vay có thể sử dụng để đưa ra những quyết định tài chính tốt nhất cho họ.

Give Me Some Credit

1.1 Giới thiệu về bài toán

Bài toán này là bài toán : Cải thiện tiêu chuẩn hiện hành

Bài toán này có ý nghĩa là : Giúp người vay có thể sử dụng để đưa ra những quyết định tài chính tốt nhất cho họ.

1.2 Phân tích yêu cầu của bài toán

1.2.1 Yêu cầu của bài toán

Xây dựng một mô hình giúp người vay vốn đưa ra quyết định vay hiệu quả phù hợp với tài chính của họ

1.2.2 Đề xuất các mô hình giải quyết bài toán

- Logistic Regression
- Decision Tree
- Random Forest
- XGBoost Classifier

1.3 Lý do chọn mô hình

1.3.1 Mô hình tổng quát

1. Logistic Regression:

- Lý do chọn: Logistic Regression là mô hình tuyến tính và được sử dụng rộng rãi trong các bài toán phân loại nhị phân. Nó ước lượng xác suất của một sự kiện xảy ra dựa trên một hoặc nhiều biến độc lập.

Decision Tree:

- Lý do chọn: Decision Tree hoạt động bằng cách phân chia tập dữ liệu thành các nhóm nhỏ hơn dựa trên các tiêu chí quyết định.

Random Forest:

- Lý do chọn: Random Forest là một mô hình kết hợp, sử dụng nhiều cây quyết định (Decision Trees) để đưa ra dự đoán.

XGBoost Classifier:

- Lý do chọn: XGBoost là một thuật toán boosting tree, nó tối ưu hóa cả về tốc độ và hiệu suất.

1.3.2 Đặc trưng của mô hình đề xuất

- Logistic Regression:

Ưu điểm: Mô hình này có tính giải thích cao, dễ triển khai và nhanh chóng.

- Decision Tree:

Ưu điểm: Mô hình này dễ hiểu và có thể trực quan hóa. Nó có khả năng xử lý dữ liệu số và phân loại.

- Random Forest:

Ưu điểm: So với một Decision Tree riêng lẻ, Random Forest giảm thiểu hiện tượng overfitting và cung cấp độ chính xác cao hơn. Nó cũng cho phép đánh giá tầm quan trọng của các đặc trưng.

- XGBoost Classifier:

Ưu điểm: XGBoost được thiết kế để nhanh chóng và hiệu quả, và thường cung cấp độ chính xác cao hơn so với các mô hình khác trong nhiều bài toán phân loại. Nó cũng hỗ trợ tính năng đánh giá đặc trưng và có nhiều tùy chỉnh.

1.4 Thực nghiệm

1.4.1 Dữ liệu

Dữ liệu được lấy ở :

https://www.kaggle.com/competitions/GiveMeSomeCredit/overview

Mô tả dữ liệu:

Tên cột	Mô tả	Kiểu dữ liệu
Variable Name	Tên biến	Chuỗi
Description	Mô tả biến	Chuỗi
SeriousDiqin2yrs	Người vay đã có lịch sử thanh toán chậm 90 ngày hoặc tệ hơn trong vòng 2 năm trước hay không	Boolean
Revolving UtilizationOfUnsecured Lines	Tổng số dư trên thẻ tín dụng và các khoản vay cá nhân không được thế chấp, ngoại trừ bất kỳ khoản nợ trả góp nào như khoản vay ô tô, chia cho tổng hạn mức tín dụng	Số thập phân
age	Tuổi của người vay theo năm	Số nguyên
NumberOfTime 30-59DaysPastDueNotWorse	Số lần người vay đã bị trễ hạn 30-59 ngày nhưng không tệ hơn trong vòng 2 năm qua	Số nguyên
DebtRatio	Khoản thanh toán nợ hàng tháng, tiền cấp dưỡng, chi phí sinh hoạt chia cho thu nhập gộp hàng tháng	Số thập phân
MonthlyIncome	Thu nhập hàng tháng của người vay	Số thực
NumberOfOpenCreditLinesAndLoans	Số khoản vay mở (như khoản vay ô tô hoặc thế chấp) và các khoản vay tín dụng (ví dụ: thẻ tín dụng)	Số nguyên
NumberOfTimes90DaysLate.	Số lần người vay bị trễ hạn 90 ngày hoặc hơn	Số nguyên
NumberRealEstateLoansOrLines	Số khoản vay bất động sản và các khoản vay thế chấp bao gồm cả các khoản vay thế chấp thế chấp	Số nguyên
NumberOfTime60-89DaysPastDueNot Worse	Số lần người vay bị trễ hạn 60-89 ngày nhưng không tệ hơn trong vòng 2 năm qua	Số nguyên
NumberOfDependents	Số người phụ thuộc trong gia đình không bao gồm bản thân họ (vợ / chồng, con cái, v.v.)	Số nguyên

1.4.2 Xử lý dữ liệu

Kiểm tra dữ liệu có bị thiếu hay không

```
train_df.isnull().sum()
```

```
SeriousDlqin2yrs                           0
RevolvingUtilizationOfUnsecuredLines       0
age                                        0
NumberOfTime30-59DaysPastDueNotWorse       0
DebtRatio                                  0
MonthlyIncome                          29731
NumberOfOpenCreditLinesAndLoans            0
NumberOfTimes90DaysLate                    0
NumberRealEstateLoansOrLines               0
NumberOfTime60-89DaysPastDueNotWorse       0
NumberOfDependents                      3924
dtype: int64
```

xử lý các cột dữ liệu bị thiếu:

```
# Điền giá trị thiếu
# Điền giá trị trung bình cho MonthlyIncome
train_df['MonthlyIncome'].fillna(train_df['MonthlyIncome'].mean(), inplace=True)
# Điền giá trị trung vị cho NumberOfDependents
train_df['NumberOfDependents'].fillna(train_df['NumberOfDependents'].mode()[0], inplace=True)
```

```
train_df.isnull().sum()
```

```
SeriousDlqin2yrs                       0
RevolvingUtilizationOfUnsecuredLines   0
age                                    0
NumberOfTime30-59DaysPastDueNotWorse   0
DebtRatio                              0
MonthlyIncome                          0
NumberOfOpenCreditLinesAndLoans        0
NumberOfTimes90DaysLate                0
NumberRealEstateLoansOrLines           0
NumberOfTime60-89DaysPastDueNotWorse   0
NumberOfDependents                     0
dtype: int64
```

Xử lý các dữ liệu bị trùng lặp

Xử lý dữ liệu trùng lặp

```
    # Đếm số hàng bị trùng
    print(f'Duplicated rows : {train_df.duplicated().sum()}')
```

```
Duplicated rows : 767
```

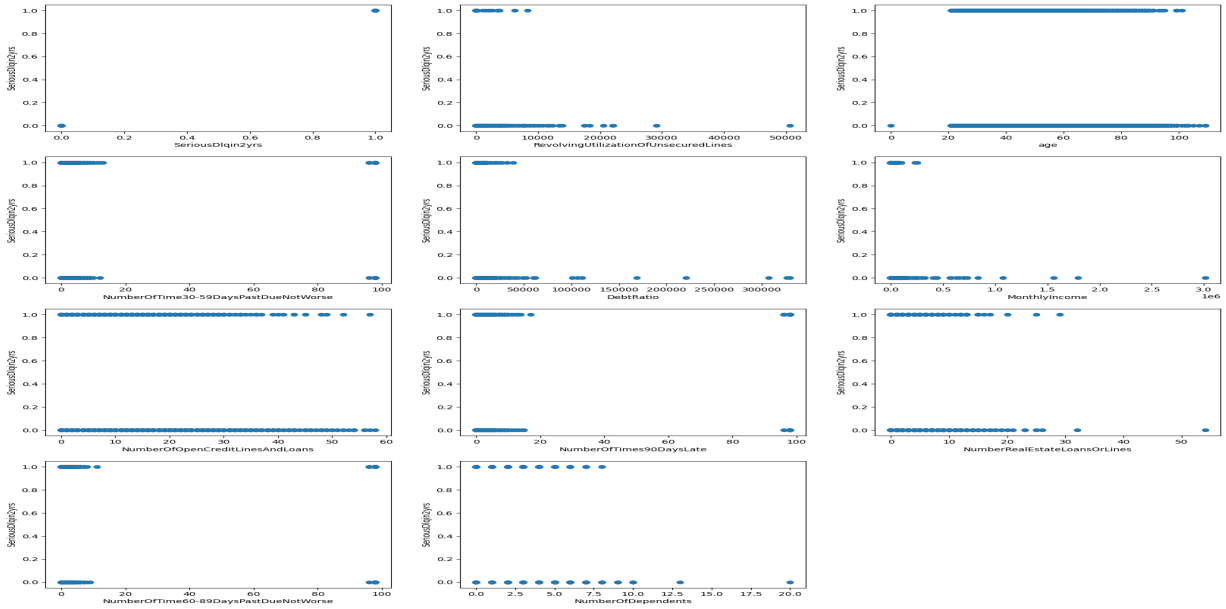
```
    # Xóa các hàng bị trùng
    train_df.drop_duplicates(inplace=True)
    # Kiểm tra lại
    print(f'Check again Duplicated rows : {train_df.duplicated().sum()}')
```

```
Check again Duplicated rows : 0
```

```
    train_df.shape
```

```
(149233, 11)
```

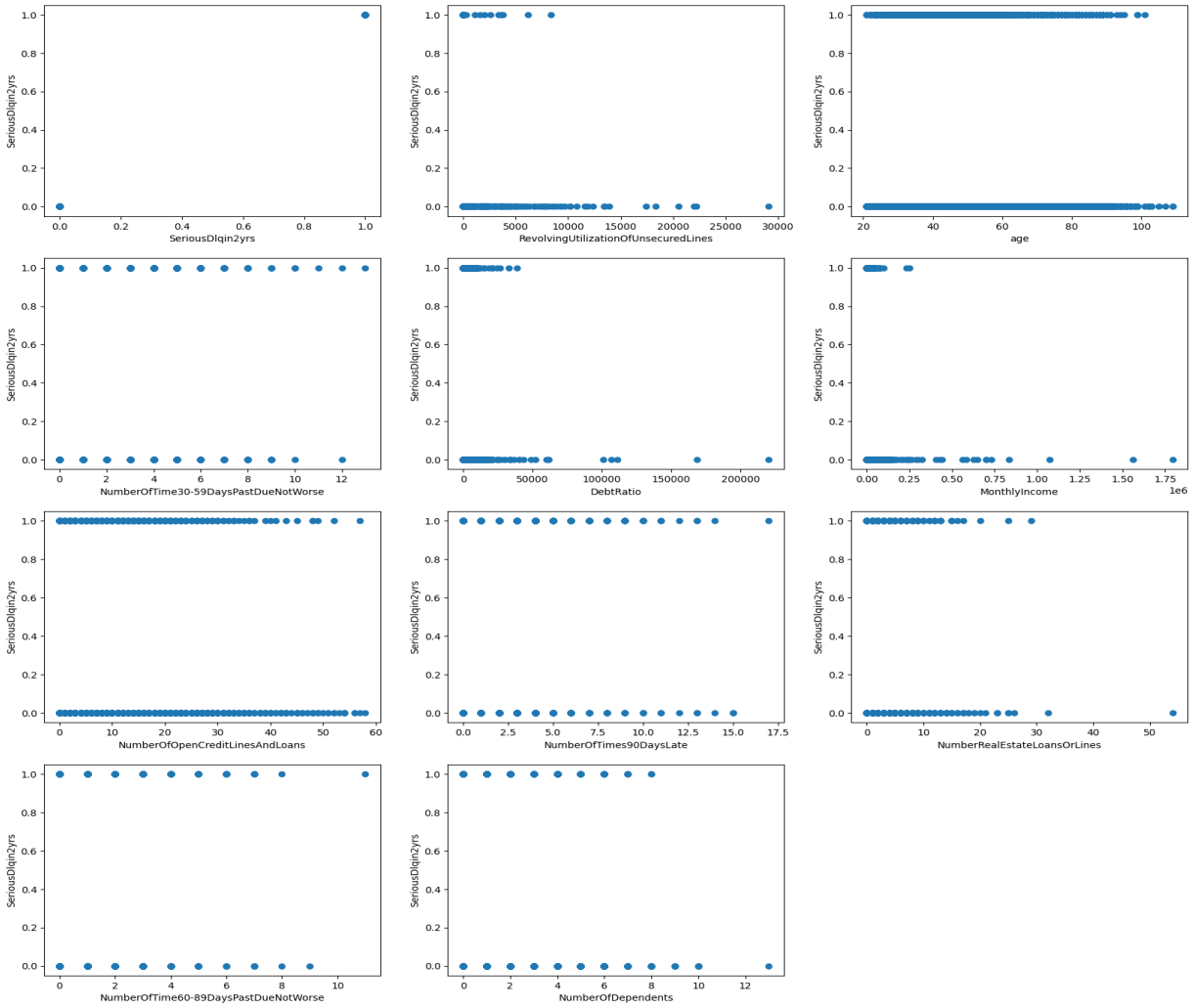
Xử lý điểm ngoại lai có giá trị bất thường :



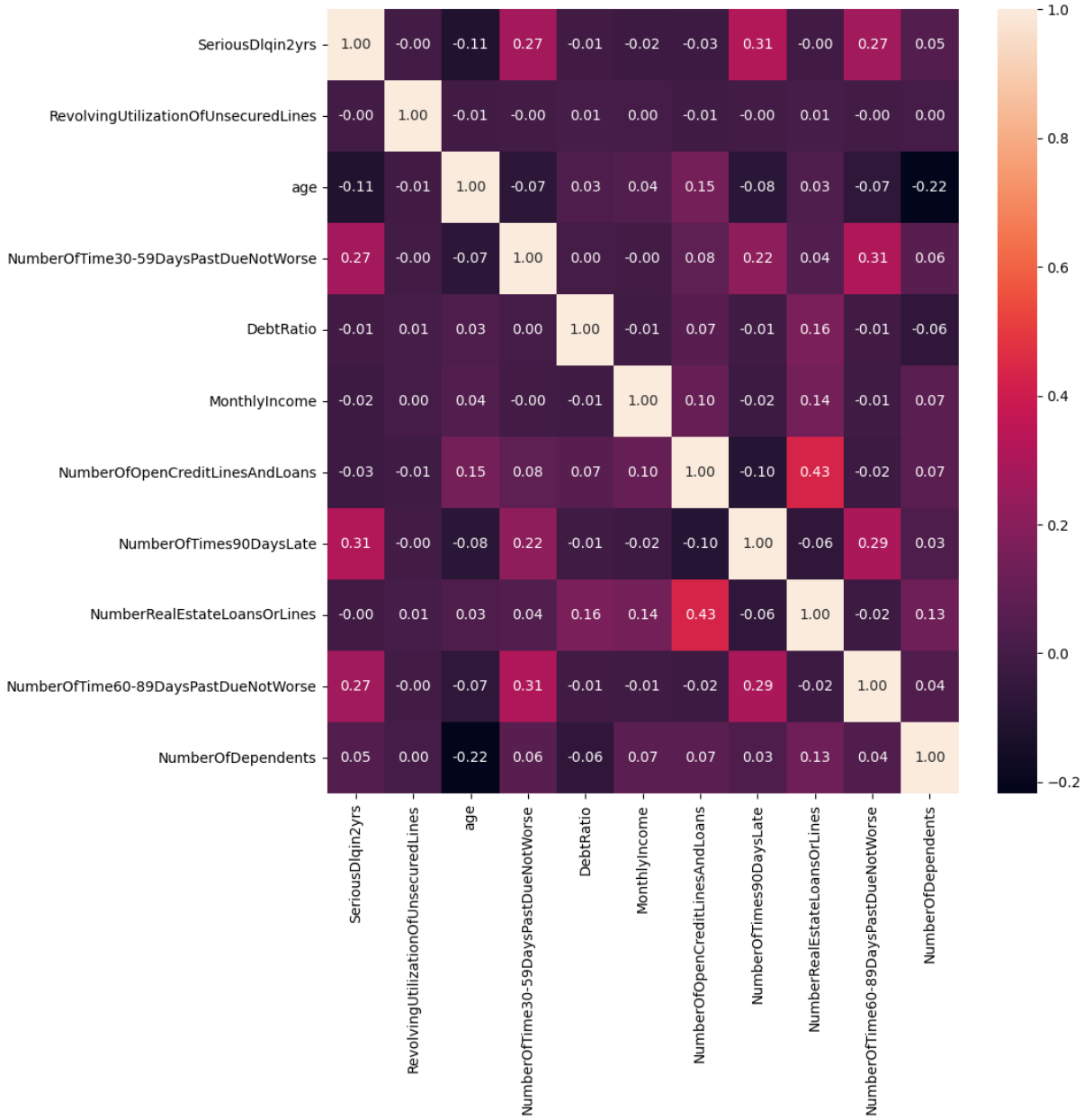
- Xoá bỏ điểm ngoại lai (Outlier)

```
train_df = train_df[train_df['RevolvingUtilizationOfUnsecuredLines'] < 50000]
train_df = train_df[train_df['age'] > 0]
train_df = train_df[train_df['NumberOfTime30-59DaysPastDueNotWorse'] < 90]
train_df = train_df[train_df['DebtRatio'] < 300000]
train_df = train_df[train_df['MonthlyIncome'] < 3000000]
train_df = train_df[train_df['NumberOfTimes90DaysLate'] < 90]
train_df = train_df[train_df['NumberOfTime60-89DaysPastDueNotWorse'] < 90]
train_df = train_df[train_df['NumberOfDependents'] < 20]
```

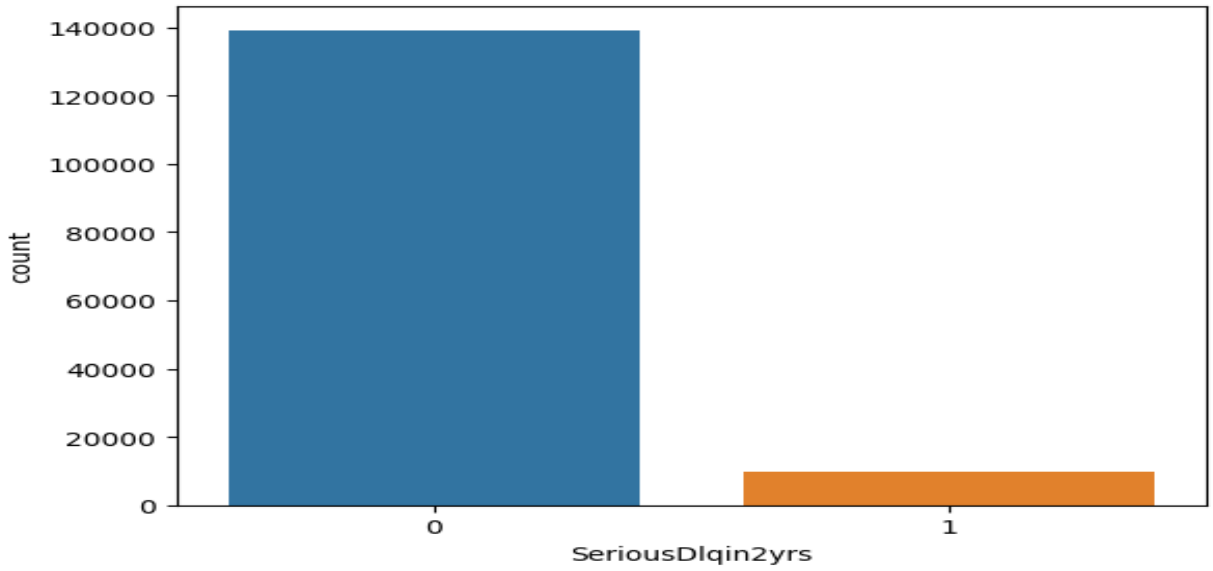
Dữ liệu sau khi loại bỏ điểm ngoại lai:



Đánh giá tương quan dữ liệu giữa các cột :



kiểm tra cân bằng dữ liệu



Dùng phương pháp SMOTE để xử lý sự mất cân bằng này:

1.4.3 Công nghệ sử dụng

Các thư viện được sử dụng :

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import  RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix, accuracy_score, f1_score
from sklearn.model_selection import GridSearchCV
import numpy as np
from sklearn.utils import resample
from scipy import stats
import pydotplus
from collections import deque
import math
import numpy as np
import pandas as pd
```

1.4.4 Cách đánh giá

Đánh giá bằng độ đo : confusion_matrix, accuracy_score, f1_score

1. Confusion Matrix (Ma trận nhầm lẫn):

Lý do chọn: Ma trận nhầm lẫn cung cấp một cái nhìn tổng quát về hiệu suất của mô hình. Nó chia thành bốn phần: True Positive (TP), False Positive (FP), True Negative (TN) và False Negative (FN).

Ưu điểm: Giúp ta thấy được số lượng các trường hợp được phân loại chính xác và không chính xác. Đặc biệt, trong những bài toán có sự mất cân bằng dữ liệu, việc nhìn vào ma trận nhầm lẫn giúp ta hiểu rõ hơn về hiệu suất của mô hình.

2. Accuracy Score (Độ chính xác):

Lý do chọn: Độ chính xác là tỷ lệ của số lượng dự đoán đúng trên tổng số trường hợp.

Ưu điểm: Đơn giản và dễ hiểu. Trong nhiều trường hợp, độ chính xác cung cấp cái nhìn sơ bộ về hiệu suất của mô hình.

Hạn chế: Đối với dữ liệu mất cân bằng, độ chính xác có thể không phản ánh chính xác hiệu suất của mô hình.

3. F1 Score:

Lý do chọn: F1 Score là trung bình điều hòa của Precision (độ chính xác của dự đoán tích cực) và Recall (tỷ lệ dự đoán đúng trong số các trường hợp thực sự tích cực).

Ưu điểm: F1 Score rất hữu ích trong các tình huống mà một trong hai giá trị Precision hoặc Recall quan trọng hơn hoặc khi dữ liệu mất cân bằng. Nó giúp đánh giá sự cân bằng giữa Precision và Recall, đặc biệt trong trường hợp dữ liệu có sự mất cân bằng giữa hai lớp.

1.5 Phân chia dữ liệu

Chia dữ liệu bằng train_test_split với test_size là 20%

- Chia dữ liệu thành 2 phần train và test

```
X_train, X_val, y_train, y_val = train_test_split(X_smote, y_smote, test_size=0.2, random_state=42)
```

1.6 Tìm tham số tối ưu cho mô hình

- Logistic Regression

```
# Tìm tham số tốt nhất cho logistic regression
param_grid = {'C': [0.1, 1, 10, 100, 1000]}
grid = GridSearchCV(LogisticRegression(), param_grid, refit=True, verbose=3)
grid.fit(X_train, y_train)
print(grid.best_params_)
print(grid.best_estimator_)
```

Tham số tối ưu:

```
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
[CV 5/5] END ..........................C=1000;, score=0.661 total time=   0.9s
{'C': 10}
LogisticRegression(C=10)
```

- Decision Tree

```
# Tìm tham số tốt nhất cho decision tree
param_grid = {'max_depth': [3, 5, 7, 9, 11, 13, 15]}
grid = GridSearchCV(DecisionTreeClassifier(), param_grid, refit=True, verbose=3)
grid.fit(X_train, y_train)
print(grid.best_params_)
print(grid.best_estimator_)
```

Tham số tối ưu:

```
{'max_depth': 15}
DecisionTreeClassifier(max_depth=15)
```

- Random Forest

```
# Tìm tham số tốt nhất cho random forest
param_grid = {'n_estimators': [100, 200, 300, 400,500]}
grid = GridSearchCV(RandomForestClassifier(), param_grid, refit=True, verbose=3)
grid.fit(X_train, y_train)
print(grid.best_params_)
print(grid.best_estimator_)
```

Tham số tối ưu:

```
{'n_estimators': 400}
RandomForestClassifier(n_estimators=400)
```

- XGBoost Classifier

```
# Tìm tham số tốt nhất cho xgboost
param_grid = {'n_estimators': [100, 200, 300, 400,500]}
grid = GridSearchCV(XGBClassifier(), param_grid, refit=True, verbose=3)
grid.fit(X_train, y_train)
print(grid.best_params_)
print(grid.best_estimator_)
```

Tham số tối ưu:

```
max_delta_step=None, max_depth=None, max_leaves=None,
min_child_weight=None, missing=nan, monotone_constraints=None,
n_estimators=500, n_jobs=None, num_parallel_tree=None,
predictor=None, random_state=None, ...)
```

1.6 Kết luận

Kết quả đạt được:

- Đối với mô hình tự xây dựng

  Độ đo của 4 mô hình có giá trị như sau:

  Logistic Regression:

```
    cost = (-y.dot(np.log(output)) - ((1 -
My Logistic Regression
Accuracy :  0.5013834986344688
F1 score :  0.6678953100130443
Confusion Matrix :
 [[    0 27751]
 [    0 27905]]
```

  Decision Tree

```
My Decision Tree
Accuracy :  0.6080566336064396
F1 score :  0.4109100729138536
Confusion Matrix :
 [[26234  1517]
 [20297  7608]]
```

  Random Forest

```
    y_pred = np.squeeze(stats.mode(tree_p
My Random Forest
Accuracy :  0.9127317809400604
F1 score :  0.9128273237970458
Confusion Matrix :
 [[25369  2382]
 [ 2475 25430]]
```

- Đối với mô hình sử dụng trong thư viện Sklearn

  Logistic Regression

```
Logistic Regression
Accuracy :  0.6942827368118442
F1 score :  0.6773122949420622
Confusion Matrix :
 [[20784  6967]
 [10048 17857]]
```

  Decision Tree

```
Decision Tree
Accuracy :  0.8705620238608596
F1 score :  0.8704689298043728
Confusion Matrix :
 [[24246  3505]
 [ 3699 24206]]
```

  Random Forest

```
Random Forest
Accuracy :  0.9144746298691965
F1 score :  0.9146127076382161
Confusion Matrix :
 [[25403  2348]
 [ 2412 25493]]
```

Ngoài ra đối với việc xử dụng mô hình có sẵn còn có mô hình XGBoost là mô hình khá mạnh và nổi tiếng được áp dụng rộng rải sau đâu là kết quả của mô hình này

```
XGBoost
Accuracy :  0.9044846916774472
F1 score :  0.903513866705387
Confusion Matrix :
 [[25450  2301]
 [ 3015 24890]]
```

:

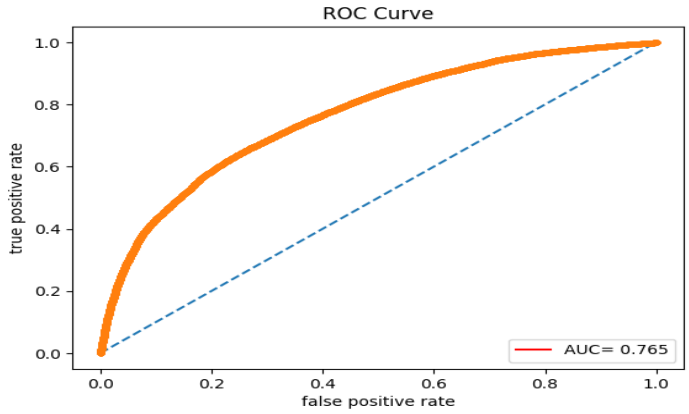
 Kết quả cho thấy độ chính xác của mô hình khá cao khoảng 90% nhưng vẫn thấp hơn mô hình Random Forest. Vì vậy mô hình tốt nhất là Random Forest sẽ được lưu lại và sử dụng để dự đoán file test_data

1.7. Kết Luận :

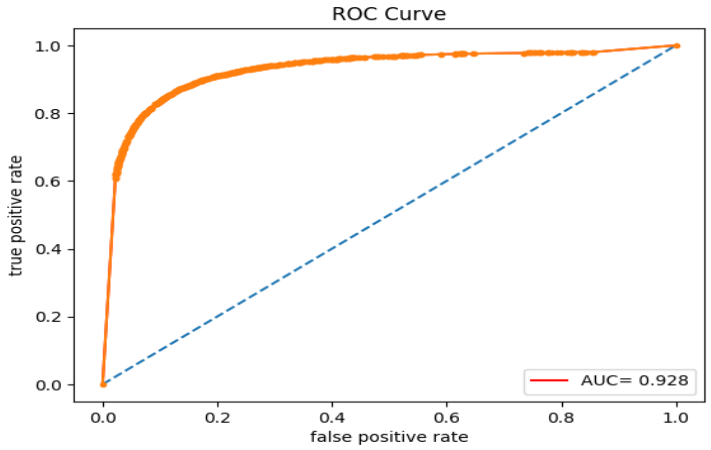
 Thông qua việc gọi mô hình trong thư viện SKlearn để so sánh kết quả với mô hình tự xây dựng, cho thấy kết quả của mô hình tự xây dựng có độ chính xác khá thấp so với mô hình trong thư viện SKlearn. Do đó, cần phải tìm hiểu thêm về các thuật toán để cải thiện mô hình. Duy nhất chỉ có mô hình Random Forest có độ chính xác khá cao so với các mô hình còn lại. Từ đó, có thể thấy được mô hình Random Forest là mô hình phù hợp nhất cho bài toán này.

- **Biểu đồ ROC**

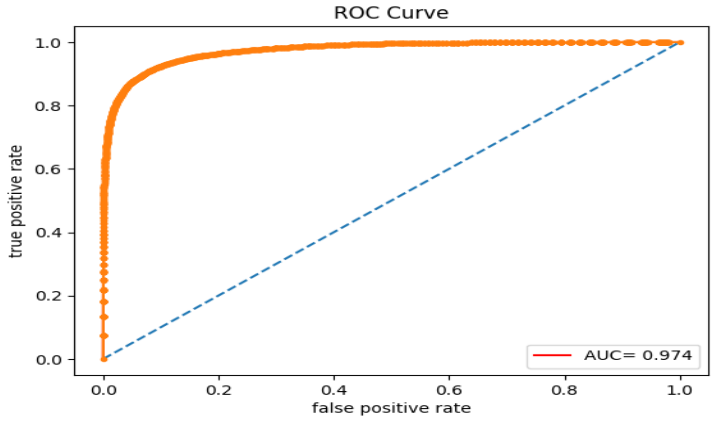
  Logistic regression:



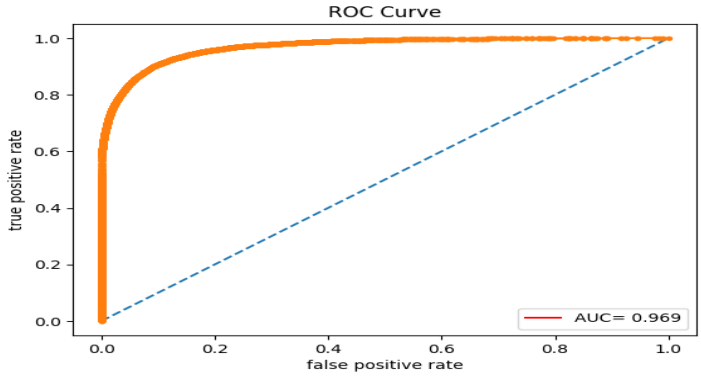
  Decision Tree:



Random Forest:



XGBoost:



→ kết luận biểu đồ ROC :

Giá trị AUC:

AUC là diện tích nằm dưới đường ROC.

Giá trị AUC nằm trong khoảng từ 0 đến 1.

AUC = 0.5: Hiệu suất của mô hình tương đương với một bộ phân loại ngẫu nhiên.

AUC > 0.5: Mô hình có hiệu suất tốt hơn bộ phân loại ngẫu nhiên.

AUC < 0.5: Mô hình có hiệu suất kém hơn bộ phân loại ngẫu nhiên.

AUC = 1: Mô hình phân loại hoàn hảo.

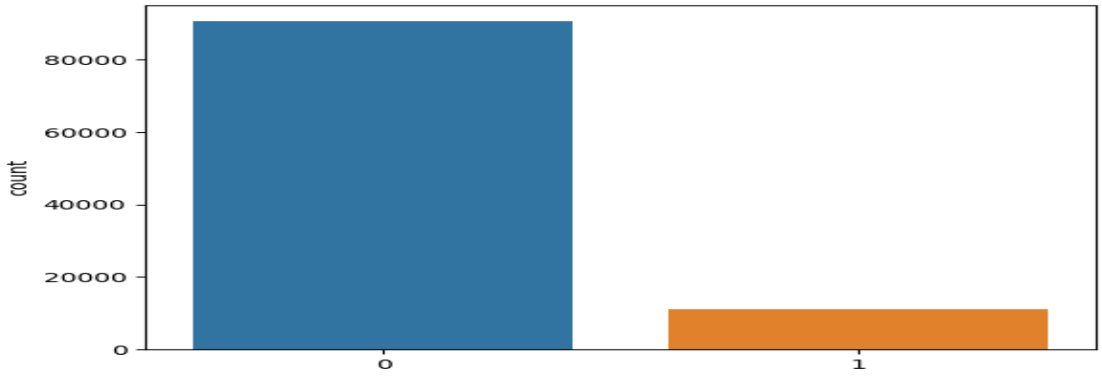
Đánh giá: Dựa vào các biểu đồ trên cho thấy mô hình Random Forest vẫn là mô hình có độ chính xác cao nhất cho bài toán phân loại này

→ Kết quả đạt được:

```
    test_data = test_df.drop('SeriousDlqin2yrs', axis=1)

    test_data = scaler.transform(test_data)

    model = pickle.load(open('model.pkl','rb'))
    y_pred = model.predict_proba(test_data)[:,1]
281]
```

```
    print(predict_target[:5])
    print(entry_df.head())
282]
..  [0.0875      0.125       0.0375      0.145       0.22378013]
       Id  Probability
    0   1     0.080807
    1   2     0.040719
    2   3     0.011968
    3   4     0.067640
    4   5     0.108264
```

```
    target = model.predict(test_data)
    print(pd.Series(target).value_counts())

    sns.countplot(x=target)
296]
..  0    90450
    1    11053
    dtype: int64
```



TÀI LIỆU THAM KHẢO

https://www.kaggle.com/code/simonpfish/comp-stats-group-data-project-final

https://www.kaggle.com/code/ashwinisaurabh/modeling-give-me-some-credit

https://www.kaggle.com/code/mifuniree/beginner-give-me-some-credit

https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

https://github.com/UserSaiVarma/Logistic-Regression-using-Gradient-Descent-without-Sklearn/blob/main/Logistic_Regression_using_Gradient_Descent.ipynb

https://github.com/hmahajan99/Decision-Tree-Implementation/blob/master/Decision%20Tree%20Implementation.ipynb