[DA_Project TIMA] Task 11_Xây dựng mô hình dự đoán và tối ưu mô hình

1.1. Sử Dụng Ma Trận Tương Quan (Correlation Matrix)

Ma trận tương quan đánh giá mối quan hệ giữa các biến

- +1: Hoàn toàn tương quan thuận (khi một biến tăng, biến kia cũng tăng).
- -1: Hoàn toàn tương quan nghịch (khi một biến tăng, biến kia giảm).
- 0: Không có tương quan.

Code block

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

# Tạo ma trận tương quan
correlation_matrix = df.corr()
# Vẽ ma trận tương quan bằng heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Matrix')
plt.show()
```

1.2. p-value

- **p-value** là một chỉ số quan trọng trong thống kê dùng để kiểm tra giả thuyết về mối quan hệ giữa các biến. Khi đánh giá các thuộc tính trong học máy, p-value có thể giúp chúng ta xác định sự quan trọng của mỗi biến trong mô hình.
  - **p-value ⩽ 0.05**: Biến có mối quan hệ mạnh với biến mục tiêu.
  - **p-value > 0.05**: Biến không có mối quan hệ mạnh với biến mục tiêu.

Code block

```
import statsmodels.api as sm
# Giả sử X là ma trận đặc trưng và y là biến mục tiêu
X = df.drop(columns=['TS_CREDIT_SCORE_V2'])
y = df['TS_CREDIT_SCORE_V2']
# Thêm cột 1 vào X để tính hệ số chệch (intercept)
X = sm.add_constant(X)
# Fit mô hình hồi quy tuyến tính
model = sm.OLS(y, X).fit()
# In kết quả p-value
print(model.summary())
#Lấy ra các thuộc tính quan trọng
fe = model.summary()
results_as_html = fe.tables[1].as_html()
result = pd.read_html(results_as_html, header=0, index_col=0)[0]
feature_import = [i for i in result[result['P>|t|']<=0.05].index if i not in ['const']]
```

1.3. Sử dụng các mô hình Machine Learning

- Sử dụng các mô hình học máy như **Random Forest**, **XGBoost** hay **Lasso Regression, Hồi quy**

Code block

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import RandomForestRegressor
import pandas as pd
X = df.drop(columns=['TS_CREDIT_SCORE_V2'])
y = df['TS_CREDIT_SCORE_V2']
# Khởi tạo và huấn luyện mô hình RandomForest
# Ensure your target variable is continuous (not categorical)
rf = RandomForestRegressor(n_estimators=100)
rf.fit(X, y)
# Lấy tầm quan trọng của các đặc trưng
feature_importances = pd.DataFrame(rf.feature_importances_,
                                   index=X.columns,
                                   columns=['Importance']).sort_values('Importance', ascending=False)

print(feature_importances)
```

2. Tạo các thuộc tính mới

2.1 Mã hóa các biến phân loại

Code block

```
from sklearn.preprocessing import LabelEncoder, StandardScaler
# Mã hóa các cột phân loại
categorical_columns = df.select_dtypes(include=['object']).columns.tolist()
label_encoders = {}
for col in categorical_columns:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col].astype(str))  # Đảm bảo chuyển tất cả giá trị thành chuỗi
    label_encoders[col] = le
```

2.2 Chuẩn hóa dữ liệu

Code block

```
from sklearn.preprocessing import LabelEncoder, StandardScaler
scaler = StandardScaler()
df[list(set(df.columns)-set(['biến mục tiêu']))] = scaler.fit_transform(df[list(set(df.columns)-
set(['biến mục tiêu']))])
```

2.3 Biến Hóa Dữ Liệu Thời Gian

- **Tuổi khách hàng**: Dựa trên ngày sinh và ngày hiện tại, ta có thể tính được tuổi của khách hàng.
- **Số ngày vay còn lại**: Dựa trên ngày đăng ký vay và ngày kết thúc, ta có thể tính số ngày còn lại trong thời gian vay.

2.4. Tạo Các Thuộc Tính Mới Dựa Trên Các Tương Quan

- **Tỷ lệ giữa tiền vay ban đầu và tiền gốc còn lại**: Cột này sẽ giúp đánh giá tỷ lệ phần trăm của khoản vay còn lại so với tổng số tiền vay.
- **Tỷ lệ "Salary" và "LoanAmount"**: Đây là tỷ lệ giữa thu nhập và số tiền vay của khách hàng, có thể cho thấy khả năng thanh toán của khách hàng.

2.5. Sử Dụng Kỹ Thuật PCA (Principal Component Analysis)

- **PCA** giúp giảm số chiều của bộ dữ liệu mà vẫn giữ lại phần lớn thông tin quan trọng. Sử dụng PCA có thể giúp cải thiện hiệu suất mô hình nếu dữ liệu có nhiều đặc trưng có tương quan.

Code block

```python
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
# Chuẩn hóa dữ liệu trước khi áp dụng PCA
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df.drop(columns=['TS_CREDIT_SCORE_V2']))  # Giả sử 'Target' là biến mục tiêu
# Áp dụng PCA
pca = PCA(n_components=5)  # Giảm xuống 2 chiều
X_pca = pca.fit_transform(X_scaled)
# Chuyển đổi dữ liệu vào không gian PCA
df_pca = pd.DataFrame(X_pca, columns=['PC1', 'PC2', 'PC3', 'PC4', 'PC5'])
# In kết quả
df_pca
```

2.6 Traning model

- Model hồi quy Logistic

Code block

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder, StandardScaler

# Mã hóa các cột phân loại
categorical_columns = df.select_dtypes(include=['object']).columns.tolist()
label_encoders = {}
for col in categorical_columns:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col].astype(str))  # Đảm bảo chuyển tất cả giá trị thành chuỗi
    label_encoders[col] = le

#Chuẩn hóa dữ liệu
scaler = StandardScaler()
df[list(set(df.columns)-set(['HasLatePayment']))] = scaler.fit_transform(df[list(set(df.columns)-set(['HasLatePayment']))])
df = df.dropna()

#Chia dữ liệu thành tập huấn luyện và kiểm tra
X = df[list(set(df.columns)-set(['HasLatePayment']))]  # Các biến độc lập
y = df['HasLatePayment']  # Biến phụ thuộc
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Huấn luyện mô hình Logistic Regression
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

# Đánh giá mô hình
y_pred = model.predict(X_test)
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# In các hệ số của mô hình
coefficients = pd.DataFrame(model.coef_.flatten(), X_train.columns, columns=['Coefficient'])
print("\nCác hệ số của các thuộc tính quan trọng:")
print(coefficients.sort_values(by='Coefficient', ascending=False))
```

- Mô hình hồi quy tuyến tính

Code block

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder, StandardScaler

# Mã hóa các cột phân loại
categorical_columns = df.select_dtypes(include=['object']).columns.tolist()
label_encoders = {}
for col in categorical_columns:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col].astype(str))  # Đảm bảo chuyển tất cả giá trị thành chuỗi
    label_encoders[col] = le

#Chuẩn hóa dữ liệu
scaler = StandardScaler()
df[list(set(df.columns)-set(['Biến mục tiêu']))] = scaler.fit_transform(df[list(set(df.columns)-
set(['Biến mục tiêu']))])
df = df.dropna()

#Chia dữ liệu thành tập huấn luyện và kiểm tra
X = df[list(set(df.columns)-set(['Biến mục tiêu']))]  # Các biến độc lập
y = df['Biến mục tiêu']  # Biến phụ thuộc
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

#Khởi tạo và huấn luyện mô hình hồi quy tuyến tính
model = LinearRegression()
model.fit(X_train, y_train)

# Đánh giá mô hình
y_pred = model.predict(X_test)
#Đánh giá mô hình
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

#In kết quả
print(f'Mean Squared Error: {mse}')
print(f'Root Mean Squared Error: {rmse}')
print(f'R-squared: {r2}')

# In các hệ số của mô hình
coefficients = pd.DataFrame(model.coef_, X_train.columns, columns=['Coefficient'])
print("\nCác hệ số của các thuộc tính quan trọng:")
print(coefficients.sort_values(by='Coefficient', ascending=False))
```

Sử dụng một số kỹ thuật đánh giá thuộc tính và tạo thuộc tính mới để xây dựng các model sau:

1. Dự báo khả năng trả nợ đúng hạn của khách hàng (Loan Default Prediction)

- **Biến mục tiêu**: Loan_Status (Trạng thái khoản vay: "Trả đúng hạn" hoặc "Trễ hạn").
- **Cách tính**:

- **Trả đúng hạn**: Nếu số tiền còn lại (SoTienConLai) = 0 và ngày đáo hạn (ToDate) đã qua, hoặc khách hàng không có thanh toán trễ (HasLatePayment = 0).
- **Trễ hạn**: Nếu số tiền còn lại (SoTienConLai) > 0 và ngày đáo hạn (ToDate) đã qua, hoặc nếu có thanh toán trễ (HasLatePayment = 1).

2. Dự báo tỷ lệ rủi ro tín dụng của khách hàng (Credit Risk Classification)

- **Biến mục tiêu**: Risk_Level (Mức độ rủi ro: "Cao", "Trung bình", "Thấp").
- **Cách tính**:
  - **Cao**: Nếu điểm tín dụng (TS_CREDIT_SCORE_V2) dưới 500 và khách hàng có lịch sử nợ xấu (HasBadDebt = 1).
  - **Trung bình**: Nếu điểm tín dụng (TS_CREDIT_SCORE_V2) từ 500 đến 700 và không có nợ xấu.
  - **Thấp**: Nếu điểm tín dụng (TS_CREDIT_SCORE_V2) trên 700 và không có nợ xấu.

3. Dự báo khả năng khách hàng có nợ xấu (Bad Debt Prediction)

- **Biến mục tiêu**: BadDebt (Có nợ xấu: 0 - Không, 1 - Có).
- **Cách tính**:
  - **Có nợ xấu**: Nếu khách hàng có lịch sử nợ xấu (HasBadDebt = 1).
  - **Không có nợ xấu**: Nếu khách hàng không có lịch sử nợ xấu (HasBadDebt = 0).

4. Dự báo khả năng gia hạn khoản vay của khách hàng (Loan Renewal Prediction)

- **Biến mục tiêu**: Loan_Renewal (Gia hạn khoản vay: 0 - Không, 1 - Có).
- **Cách tính**:
  - **Có gia hạn**: Nếu ngày đáo hạn (ToDate) đã đến, nhưng khách hàng chưa thanh toán hết khoản vay (SoTienConLai > 0) và muốn gia hạn khoản vay.
  - **Không gia hạn**: Nếu khách hàng đã trả hết khoản vay (SoTienConLai = 0) hoặc không có yêu cầu gia hạn.

5. Dự báo số tiền vay còn lại của khách hàng (Remaining Loan Amount Prediction)

- **Biến mục tiêu**: Remaining_LoanAmount (Số tiền vay còn lại).
- **Cách tính**:
  - Sử dụng giá trị SoTienConLai nếu có sẵn để xác định số tiền còn lại trong khoản vay.

6. Dự báo thu nhập khả dụng để trả nợ của khách hàng (Available Income for Loan Repayment)

- **Biến mục tiêu**: Available_Income_For_Loan (Thu nhập khả dụng cho vay).
- **Cách tính**:
  - **Thu nhập khả dụng** = Salary - các khoản chi tiêu bắt buộc (Nếu có). Nếu không có chi tiêu bắt buộc, lấy giá trị thu nhập Salary làm thu nhập khả dụng.

7. Dự báo tỷ lệ thanh toán hàng tháng của khách hàng (Monthly Payment Rate)

- **Biến mục tiêu**: Payment_Rate (Tỷ lệ thanh toán hàng tháng).
- **Cách tính**:
  - **Tỷ lệ thanh toán hàng tháng** = (Tiền giải ngân / Số tháng vay).
  - Ví dụ: Nếu số tiền vay là 100 triệu và khách hàng vay trong vòng 12 tháng, tỷ lệ thanh toán hàng tháng = 100 triệu / 12 = 8.33 triệu/tháng.

8. Dự báo khả năng khách hàng trả nợ trước hạn (Prepayment Prediction)

- **Biến mục tiêu**: Prepayment (Trả trước hạn: 0 - Không, 1 - Có).

- **Cách tính**:
  - **Trả trước hạn**: Nếu số tiền còn lại (SoTienConLai) bằng 0 trước ngày đáo hạn (ToDate).
  - **Không trả trước hạn**: Nếu khách hàng không trả hết nợ trước hạn.

9. Dự báo tần suất vay của khách hàng (Loan Frequency Prediction)

- **Biến mục tiêu**: Loan_Frequency (Tần suất vay: số lần vay trong năm).
- **Cách tính**:
  - **Tần suất vay** = Số lần khách hàng đã vay trong năm (dựa trên các khoản vay trong cột LoanID hoặc FromDate và ToDate).

10. Dự báo loại hình vay mà khách hàng sẽ chọn (Loan Type Prediction)

- **Biến mục tiêu**: Loan_Type (Loại hình vay: Tiêu dùng, Thế chấp, Tín chấp).
- **Cách tính**:
  - **Loại vay tiêu dùng**: Nếu mục đích vay là tiêu dùng, thông qua các cột như ProductCreditName
  - **Loại vay thế chấp**: Nếu khách hàng có tài sản thế chấp như nhà cửa (CityNameHouseHold, DistrictNameHouseHold).
  - **Loại vay tín chấp**: Nếu không có tài sản thế chấp và khoản vay không phải cho tiêu dùng.