|  |  |
| --- | --- |
| **Mini project:** Predict mua/không mua ở bank | |
| Họ và tên | Đặng Hưng Thịnh |
| MSSV | 22119137 |

|  |
| --- |
| **# Import các thư viện cần thiết**  *import pandas as pd*  *import numpy as np*  *from sklearn.model\_selection import GridSearchCV*  *from sklearn.preprocessing import StandardScaler, OneHotEncoder*  *from sklearn.compose import ColumnTransformer*  *from sklearn.pipeline import Pipeline*  *from sklearn.linear\_model import LogisticRegression*  *from sklearn.metrics import classification\_report, accuracy\_score, roc\_auc\_score, confusion\_matrix*  *import seaborn as sns*  *import matplotlib.pyplot as plt* |
| **# Đọc dữ liệu**  *file\_full = 'DATAKTRA/bank-additional/bank-additional/bank-additional-full.csv'*  *file\_test = 'DATAKTRA/bank-additional/bank-additional/bank-additional.csv'* |
| **# ; để chia giữa các cột (do là file csv cách nhau bở dấu ;**  *df\_full = pd.read\_csv(file\_full, sep=';')*  *df\_test = pd.read\_csv(file\_test, sep=';')* |
| *# Xác định đầu vào (features) và nhãn (label)*  *X\_train = train\_data.drop(columns=["y"])*  *y\_train = train\_data["y"].map({'no': 0, 'yes': 1})  # Chuyển nhãn thành số*  *X\_test = test\_data.drop(columns=["y"])*  *y\_test = test\_data["y"].map({'no': 0, 'yes': 1})* |

* *X\_train = train\_data.drop(columns=["y"])*

**+ Mục đích**: Lấy các cột đặc trưng (**features**) từ tập huấn luyện train\_data.

**+ Cách hoạt động**:

* train\_data là DataFrame chứa cả đặc trưng và nhãn.
* drop(columns=["y"]) loại bỏ cột "y" (cột nhãn) khỏi train\_data.
* Kết quả: X\_train chứa tất cả các cột đặc trưng, ngoại trừ cột "y".
* *y\_train = train\_data["y"].map({'no': 0, 'yes': 1})*

**+ Mục đích**: Chuyển đổi nhãn từ dạng chuỗi ("no", "yes") sang dạng số (0, 1) để phù hợp với mô hình học máy.

**+ Cách hoạt động**:

* train\_data["y"]: Truy cập cột "y" (cột nhãn) trong tập huấn luyện.
* map({'no': 0, 'yes': 1}): Ánh xạ giá trị "no" thành 0 và "yes" thành 1.
* Kết quả: y\_train là một Series chứa nhãn đã được chuyển đổi thành số.
* Tương tự cho tập test

|  |
| --- |
| **# Tiền xử lý dữ liệu**  *categorical\_features = X\_train.select\_dtypes(include=['object']).columns*  *numerical\_features = X\_train.select\_dtypes(exclude=['object']).columns* |
| *preprocessor = ColumnTransformer([*  *('num', StandardScaler(), numerical\_features),*  *('cat', OneHotEncoder(handle\_unknown='ignore'), categorical\_features)*  *])* |
| **# Xây dựng mô hình Logistic Regression**  *pipeline = Pipeline([*  *('preprocessor', preprocessor),*  *('classifier', LogisticRegression(max\_iter=1000, solver='liblinear', class\_weight='balanced'))*  *])* |
| **# Tối ưu tham số cho Logistic Regression bằng GridSearchCV**  *param\_grid = {*  *'classifier\_\_C': [0.001, 0.01, 0.1, 1, 10, 100]*  *}* |
| **# Tối ưu tham số cho Random Forest bằng GridSearchCV**  *grid\_search = GridSearchCV(pipeline, param\_grid, cv=5, scoring='accuracy', n\_jobs=-1)*  *grid\_search.fit(X\_train, y\_train)* |
|  |
| **# Lấy mô hình tốt nhất**  *best\_pipeline = grid\_search.best\_estimator\_*  *print(f"Best parameters: {grid\_search.best\_params\_}")* |
| ***Output cell:*** *Best parameters: {'classifier\_\_C': 0.001}* |
| **# Đánh giá mô hình trên tập test**  *y\_pred = best\_pipeline.predict(X\_test)*  *y\_pred\_proba = best\_pipeline.predict\_proba(X\_test)[:, 1]* |
| **# Accuracy**  *accuracy = accuracy\_score(y\_test, y\_pred)*  *print(f"Accuracy on test set: {accuracy:.4f}")* |
| **Output cell:** *Accuracy on test set: 0.8572* |
| **# Classification Report**  *report = classification\_report(y\_test, y\_pred)*  *print("Classification Report on test set:")*  *print(report)* |
| **Output cell:** Classification Report on test set:   |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | **Precision** | **Recall** | **F1- score** | **Support** | | **0** | **0.98** | **0.85** | **0.91** | **3668** | | **1** | **0.43** | **0.89** | **0.58** | **451** | | **accuracy** |  |  | **0.86** | **4119** | | **Macro avg** | **0.71** | **0.87** | **0.75** | **4119** | | **Weighted avg** | **0.92** | **0.86** | **0.88** | **4119** | |
| **# ROC-AUC**  *roc\_auc = roc\_auc\_score(y\_test, y\_pred\_proba)*  *print(f"ROC-AUC on test set: {roc\_auc:.4f}")* |
| **Output cell:** ROC-AUC on test set: 0.9344 |
| *from sklearn.metrics import ConfusionMatrixDisplay*  *ConfusionMatrixDisplay.from\_estimator(*  *best\_pipeline, X\_test, y\_test,*  *display\_labels=['No', 'Yes'],*  *cmap='Blues',*  *normalize='true'*  *)*  *plt.title('Normalized Confusion Matrix')*  *plt.show()* |
|  |