Ex 3: Bank

- Sử dụng tập dữ liệu bank.csv chứa thông tin liên quan đến các chiến dịch tiếp thị trực tiếp - the direct marketing campaigns (dựa trên các cuộc gọi điện thoại) của một tổ chức ngân hàng Bồ Đào Nha. Thông thường, cần có nhiều contact cho cùng một khách hàng, để truy cập xem liệu có sản phẩm (tiền gửi ngân hàng có kỳ hạn - bank term deposit) sẽ được đăng ký (yes) hay không (no). Tập dữ liệu chứa một số thông tin khách hàng (như age, job...) và thông tin liên quan đến chiến dịch (chẳng hạn như contact hoặc communication type, day, month và duration của contact...).
- Đối với chiến dịch tiếp thị tiếp theo, công ty muốn sử dụng dữ liệu này và chỉ liên hệ với những khách hàng tiềm năng sẽ đăng ký tiền gửi có kỳ hạn, do đó giảm bớt nỗ lực cần thiết để liên hệ với những khách hàng không quan tâm. Để làm được điều này, cần tạo một mô hình có thể dự đoán liệu khách hàng có đăng ký tiền gửi có kỳ hạn hay không (y).

Yêu cầu:

- Đọc dữ liệu, tìm hiểu sơ bộ về dữ liệu. Chuẩn hóa dữ liệu nếu cần
- Tạo X_train, X_test, y_train, y_test từ dữ liệu chuẩn hóa với tỷ lệ dữ liệu test là 0.3
- Áp dụng Random Forest, Tìm kết quả.
- Kiểm tra độ chính xác
- Đánh giá mô hình.
- Ghi mô hình nếu mô hình phù hợp

Gợi ý:

In [1]:
```
import warnings
warnings.filterwarnings('ignore')
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE
from sklearn.preprocessing import StandardScaler, RobustScaler, MinMaxScaler
from collections import Counter
```

```
Using TensorFlow backend.
```

In [2]:
```
# Đọc dữ liệu. Tìm hiểu sơ bộ về dữ liệu
bank = pd.read_csv('bank.csv', sep = ';')
bank.head()
```

Out[2]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome
0	30	unemployed	married	primary	no	1787	no	no	cellular	19	oct	79	1	-1	0	unknown
1	33	services	married	secondary	no	4789	yes	yes	cellular	11	may	220	1	339	4	failure
2	35	management	single	tertiary	no	1350	yes	no	cellular	16	apr	185	1	330	1	failure
3	30	management	married	tertiary	no	1476	yes	yes	unknown	3	jun	199	4	-1	0	unknown
4	59	blue-collar	married	secondary	no	0	yes	no	unknown	5	may	226	1	-1	0	unknown

In [5]:
```
bank['y']=bank['y'].replace({'no': 0, 'yes': 1})
```

In [6]:
```
bank['month'].replace(['jan', 'feb', 'mar','apr','may',
                       'jun','jul','aug','sep','oct','nov','dec'],
                      [1,2,3,4,5,6,7,8,9,10,11,12], inplace  = True)
```

In [7]:
```
bank.shape
```

Out[7]:
```
(4334, 17)
```

In [8]:
```
# bank.info()
```

In [9]:
```
# Kiểm tra dữ liệu null
print(bank.isnull().sum())
# => Không có dữ liệu null
```

```
age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
pdays        0
previous     0
poutcome     0
y            0
dtype: int64
```

In [10]:
```
bank.describe()
```

Out[10]:

	age	balance	day	month	duration	campaign	pdays	previous	y
count	4334.000000	4334.000000	4334.000000	4334.000000	4334.000000	4334.000000	4334.000000	4334.000000	4334.000000
mean	40.991924	1410.637517	15.913936	6.176050	264.544301	2.806876	39.670974	0.544070	0.115828
std	10.505378	3010.612091	8.216673	2.374798	260.642141	3.129682	99.934062	1.702219	0.320056
min	19.000000	-3313.000000	1.000000	1.000000	4.000000	1.000000	-1.000000	0.000000	0.000000
25%	33.000000	67.000000	9.000000	5.000000	104.000000	1.000000	-1.000000	0.000000	0.000000
50%	39.000000	440.000000	16.000000	6.000000	186.000000	2.000000	-1.000000	0.000000	0.000000
75%	48.000000	1464.000000	21.000000	8.000000	329.000000	3.000000	-1.000000	0.000000	0.000000
max	87.000000	71188.000000	31.000000	12.000000	3025.000000	50.000000	871.000000	25.000000	1.000000

In [11]:
```
bank.describe(include=['O'])
```

Out[11]:

	job	marital	education	default	housing	loan	contact	poutcome
count	4334	4334	4334	4334	4334	4334	4334	4334
unique	12	3	3	2	2	2	3	4
top	management	married	secondary	no	yes	no	cellular	unknown
freq	942	2680	2306	4261	2476	3650	2801	3555

In [12]:
```
bank['y'].value_counts(0)
```

Out[12]:
```
0    3832
1     502
Name: y, dtype: int64
```

In [13]:
```
X = bank.drop(['y'], axis=1)
```

In [14]:
```
X.head()
```

Out[14]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome
0	30	unemployed	married	primary	no	1787	no	no	cellular	19	10	79	1	-1	0	unknown
1	33	services	married	secondary	no	4789	yes	yes	cellular	11	5	220	1	339	4	failure
2	35	management	single	tertiary	no	1350	yes	no	cellular	16	4	185	1	330	1	failure
3	30	management	married	tertiary	no	1476	yes	yes	unknown	3	6	199	4	-1	0	unknown
4	59	blue-collar	married	secondary	no	0	yes	no	unknown	5	5	226	1	-1	0	unknown

In [15]:
```
y = bank['y']
```

In [16]:
```
# Dữ liệu có sự chênh lệch giữa 0 và 1
```

In [17]:
```
# Chuẩn hóa dữ liệu phân loại (kiểu chuỗi)
from sklearn.preprocessing import OneHotEncoder
```

In [18]:
```
ohe = OneHotEncoder()
ohe = ohe.fit(X[['job', 'marital', 'education',
                 'default', 'housing', 'loan',
                 'contact', 'poutcome']])
X_ohe = ohe.transform(X[['job', 'marital', 'education',
                         'default', 'housing', 'loan',
                         'contact', 'poutcome']])
```

In [19]:
```
X_ohe
```

Out[19]:
```
<4334x31 sparse matrix of type '<class 'numpy.float64'>'
        with 34672 stored elements in Compressed Sparse Row format>
```

In [20]:
```
X_ohe_new = X_ohe.toarray()
```

In [21]:
```
ohe.get_feature_names(['job', 'marital', 'education',
                       'default', 'housing', 'loan',
                       'contact', 'poutcome'])
```

Out[21]:
```
array(['job_admin.', 'job_blue-collar', 'job_entrepreneur',
       'job_housemaid', 'job_management', 'job_retired',
       'job_self-employed', 'job_services', 'job_student',
       'job_technician', 'job_unemployed', 'job_unknown',
       'marital_divorced', 'marital_married', 'marital_single',
       'education_primary', 'education_secondary', 'education_tertiary',
       'default_no', 'default_yes', 'housing_no', 'housing_yes',
       'loan_no', 'loan_yes', 'contact_cellular', 'contact_telephone',
       'contact_unknown', 'poutcome_failure', 'poutcome_other',
       'poutcome_success', 'poutcome_unknown'], dtype=object)
```

In [22]:
```
X_ohe_new[:5]
```

Out[22]:
```
array([[0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 1., 0., 1.,
        0., 0., 1., 0., 1., 0., 1., 0., 1., 0., 0., 0., 0., 0., 1.],
       [0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 1., 0., 0.,
        1., 0., 1., 0., 0., 1., 0., 1., 1., 0., 0., 1., 0., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0.,
        0., 1., 1., 0., 0., 1., 1., 0., 1., 0., 0., 1., 0., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0.,
        0., 1., 1., 0., 0., 1., 0., 1., 0., 0., 1., 0., 0., 0., 1.],
       [0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0.,
        1., 0., 1., 0., 0., 1., 1., 0., 0., 0., 1., 0., 0., 0., 1.]])
```

In [23]:
```
X_ohe_df = pd.DataFrame(X_ohe_new,
                        columns=ohe.get_feature_names(['job', 'marital',
                                                       'education','default',
                                                       'housing', 'loan',
                                                       'contact', 'poutcome']))
```

In [24]:
```
X_ohe_df.head()
```

Out[24]:

	job_admin.	job_blue-collar	job_entrepreneur	job_housemaid	job_management	job_retired	job_self-employed	job_services	job_student	job_technician	...	hc
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	...	
2	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	
3	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	
4	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	

5 rows × 31 columns

In [25]:
```
X_new = pd.concat([X[['age', 'balance', 'day','month',
                      'duration', 'campaign',
                      'pdays', 'previous']], X_ohe_df],
                  axis=1)
```

In [26]:
```
# X_new.info()
```

In [27]:
```
# Build model
X_train, X_test, y_train, y_test = train_test_split(X_new, y,
                                                    test_size=0.3,
                                                    random_state=42)
```

In [28]:
```
model = RandomForestClassifier(n_estimators=100)
model.fit(X_train, y_train)
```

Out[28]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=None, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

In [29]:
```
model.score(X_new, y)
```

Out[29]:
```
0.9695431472081218
```

In [30]:
```
model.score(X_train, y_train)
```

Out[30]: 1.0

In [31]:
```
model.score(X_test, y_test)
```

Out[31]: 0.8985395849346657

In [32]:
```
# Đánh giá model
y_pred = model.predict(X_test)
```

In [33]:
```
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

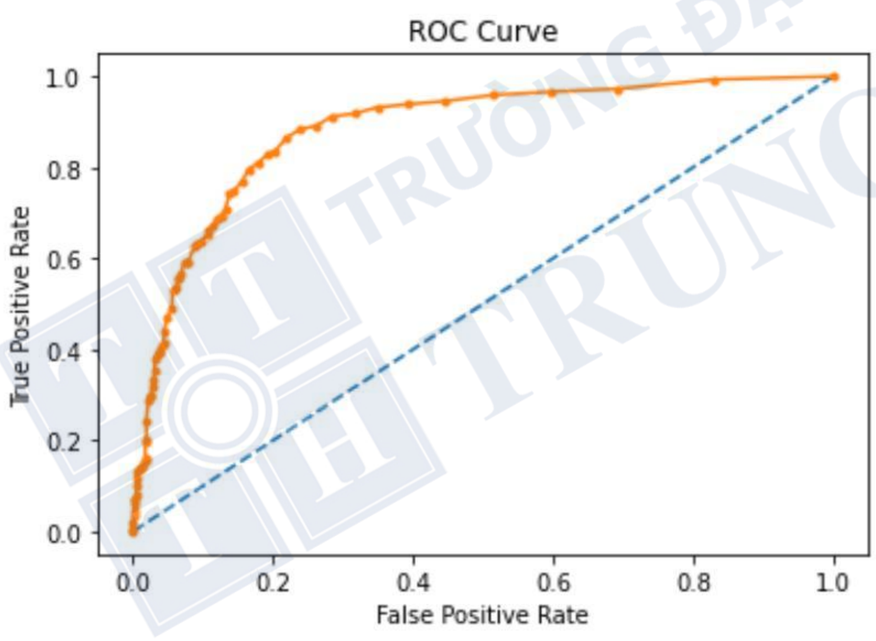
```
[[1122   32]
 [ 100   47]]
              precision    recall  f1-score   support

           0       0.92      0.97      0.94      1154
           1       0.59      0.32      0.42       147

    accuracy                           0.90      1301
   macro avg       0.76      0.65      0.68      1301
weighted avg       0.88      0.90      0.88      1301
```

In [34]:
```
# model dự đoán class 1 chưa được chính xác
from sklearn.metrics import roc_curve,auc
```

In [35]:
```
# Print ROC_AUC score using probabilities
probs = model.predict_proba(X_test)
```

In [36]:
```
scores = probs[:,1]
fpr, tpr, thresholds = roc_curve(y_test, scores)
```

In [37]:
```
plt.plot([0, 1], [0, 1], linestyle='--')
plt.plot(fpr, tpr, marker='.')
plt.title("ROC Curve")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.show()
```



In [38]:
```
auc(fpr, tpr)
```

Out[38]: 0.8845983800799935

In [ ]:
```
# Có giải pháp nào tốt hơn không?
```