Chapter 6 - Exercise 1: 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 Decision Tree, Tìm kết quả.
- Kiểm tra độ chính xác
- Trực quan hóa Decision Tree
- Đánh giá mô hình.
- Ghi mô hình nếu mô hình phù hợp

Gợi ý:

In [1]:
```
import warnings
warnings.filterwarnings('ignore')
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
from collections import Counter
from sklearn.metrics import classification_report,confusion_matrix,accuracy_scor
```

```
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	d
0	30	unemployed	married	primary	no	1787	no	no	cellular	19	oct	
1	33	services	married	secondary	no	4789	yes	yes	cellular	11	may	
2	35	management	single	tertiary	no	1350	yes	no	cellular	16	apr	
3	30	management	married	tertiary	no	1476	yes	yes	unknown	3	jun	
4	59	blue-collar	married	secondary	no	0	yes	no	unknown	5	may	

In [3]:
```
# bank = bank.rename(columns={
#                            'y': 'Target'
#                            })
```

In [4]:
```
# bank['Target']=bank['Target'].replace({'no': 0, 'yes': 1})
```

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]:
```
(4521, 17)
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4521 entries, 0 to 4520
Data columns (total 17 columns):
age          4521 non-null int64
job          4521 non-null object
marital      4521 non-null object
education    4521 non-null object
default      4521 non-null object
balance      4521 non-null int64
housing      4521 non-null object
loan         4521 non-null object
contact      4521 non-null object
day          4521 non-null int64
month        4521 non-null int64
duration     4521 non-null int64
campaign     4521 non-null int64
pdays        4521 non-null int64
previous     4521 non-null int64
poutcome     4521 non-null object
y            4521 non-null int64
dtypes: int64(9), object(8)
memory usage: 600.6+ KB
```

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
count	4521.000000	4521.000000	4521.000000	4521.000000	4521.000000	4521.000000	4521.000000
mean	41.170095	1422.657819	15.915284	6.166777	263.961292	2.793630	39.766645
std	10.576211	3009.638142	8.247667	2.378380	259.856633	3.109807	100.121124
min	19.000000	-3313.000000	1.000000	1.000000	4.000000	1.000000	-1.000000
25%	33.000000	69.000000	9.000000	5.000000	104.000000	1.000000	-1.000000
50%	39.000000	444.000000	16.000000	6.000000	185.000000	2.000000	-1.000000
75%	49.000000	1480.000000	21.000000	8.000000	329.000000	3.000000	-1.000000
max	87.000000	71188.000000	31.000000	12.000000	3025.000000	50.000000	871.000000

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

Out[11]:

	job	marital	education	default	housing	loan	contact	poutcome
count	4521	4521	4521	4521	4521	4521	4521	4521
unique	12	3	4	2	2	2	3	4
top	management	married	secondary	no	yes	no	cellular	unknown
freq	969	2797	2306	4445	2559	3830	2896	3705

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

Out[12]:
```
0    4000
1     521
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	du
0	30	unemployed	married	primary	no	1787	no	no	cellular	19	10	
1	33	services	married	secondary	no	4789	yes	yes	cellular	11	5	
2	35	management	single	tertiary	no	1350	yes	no	cellular	16	4	
3	30	management	married	tertiary	no	1476	yes	yes	unknown	3	6	
4	59	blue-collar	married	secondary	no	0	yes	no	unknown	5	5	

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]:
```
<4521x32 sparse matrix of type '<class 'numpy.float64'>'
        with 36168 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',
       'education_unknown', '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., 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., 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., 0., 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., 0., 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., 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
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
2	0.0	0.0	0.0	0.0	1.0	0.0	0.0
3	0.0	0.0	0.0	0.0	1.0	0.0	0.0
4	0.0	1.0	0.0	0.0	0.0	0.0	0.0

5 rows × 32 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 = DecisionTreeClassifier()
model.fit(X_train, y_train)
```

Out[28]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                       max_features=None, 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, presort=False,
                       random_state=None, splitter='best')
```

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

Out[29]:
```
0.9608493696084937
```

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

Out[30]:
```
1.0
```

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

Out[31]:
```
0.8695652173913043
```

In [32]:
```
# Có hiện tượng overfitting
```

In [33]:
```
# Đánh giá model
y_pred = model.predict(X_test)
```

In [34]:
```
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

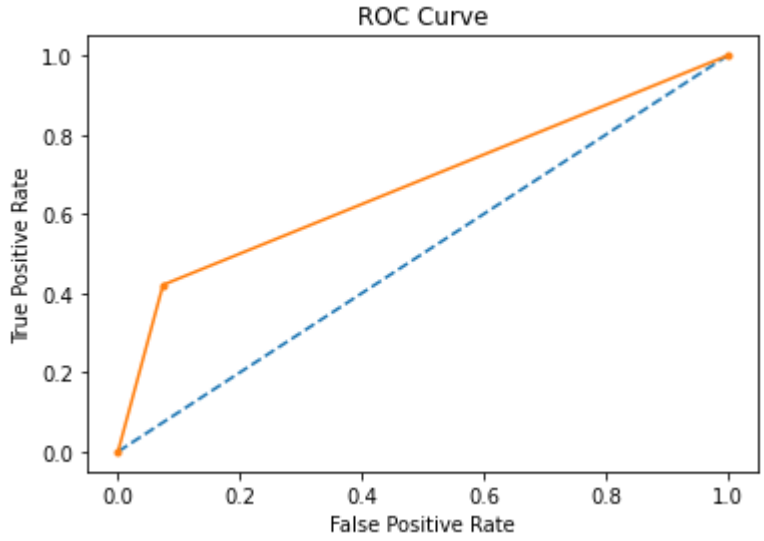
```
[[1116   89]
 [  88   64]]
              precision    recall  f1-score   support

           0       0.93      0.93      0.93      1205
           1       0.42      0.42      0.42       152

    accuracy                           0.87      1357
   macro avg       0.67      0.67      0.67      1357
weighted avg       0.87      0.87      0.87      1357
```

In [35]:
```
# model dự đoán class 1 chưa được chính xác
from sklearn.metrics import roc_curve,auc
```

In [36]:
```
# Print ROC_AUC score using probabilities
probs = model.predict_proba(X_test)
```

In [37]:
```
scores = probs[:,1]
fpr, tpr, thresholds = roc_curve(y_test, scores)
```

In [38]:
```
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 [39]:
```
auc(fpr, tpr)
```

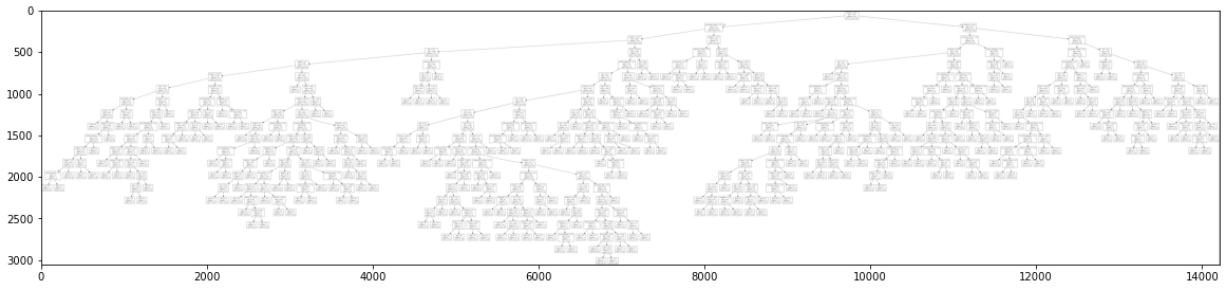
Out[39]:
```
0.6735968552085608
```

In [40]:
```
from IPython.display import Image
from sklearn import tree
import pydotplus
```

In [41]:
```
dot_data = tree.export_graphviz(model, out_file='bank.txt',
                                feature_names=X_new.columns
                                )
#graph = pydotplus.graph_from_dot_data(dot_data)
#Image(graph.create_png())
```

In [42]:
```
# dùng để mở và xem kết quả
# https://dreampuf.github.io/GraphvizOnline/
# hoặc http://viz-js.com/
import matplotlib.pyplot as plot
import imageio
photo_data = imageio.imread("bank.png")
plot.figure(figsize = (20, 20))
plot.imshow(photo_data)
```

Out[42]:
```
<matplotlib.image.AxesImage at 0x214074a7da0>
```



In [43]:
```
# Có giải pháp nào tốt hơn không?
```

In [44]:
```
X_resampled, y_resampled = SMOTE().fit_resample(X_train,y_train)
```

In [45]:
```
standard_scalar = StandardScaler()
X_train_sc_resampled = standard_scalar.fit_transform(X_resampled)
X_test_sc = standard_scalar.transform(X_test)
```

In [46]:
```
tree_model = DecisionTreeClassifier()
```

In [47]:
```
tree_model.fit(X_train_sc_resampled,y_resampled)
```

Out[47]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                       max_features=None, 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, presort=False,
                       random_state=None, splitter='best')
```

In [48]:
```
tree_model.score(X_train_sc_resampled, y_resampled)
```

Out[48]:
```
1.0
```

In [49]:
```
tree_model.score(X_test_sc, y_test)
```

Out[49]:
```
0.8607221812822402
```

In [50]:
```
# Cũng còn overfitting
```

In [51]:
```
y_pred1=tree_model.predict(X_test_sc)
```

In [52]:
```
print(confusion_matrix(y_test, y_pred1))
print(classification_report(y_test, y_pred1))
```

```
[[1100  105]
 [  84   68]]
              precision    recall  f1-score   support

           0       0.93      0.91      0.92      1205
           1       0.39      0.45      0.42       152

    accuracy                           0.86      1357
   macro avg       0.66      0.68      0.67      1357
weighted avg       0.87      0.86      0.86      1357
```

In [53]:
```
# Kết quả không cải thiện nhiều
```

In [ ]:
```

```