Chapter 3 - Ex2: Predicting Custumer Churn

- Cho dữ liệu Churn_Modelling.csv chứa thông tin của 10000 khách hàng của công ty.
- Là phụ trách bộ phận chăm sóc khách hàng bạn nhận thấy việc phải xây dựng một mô hình Machine Learning để dự đoán việc khách hàng sẽ ra đi hay ở lại. Công việc này vô cùng quan trọng vì giữ chân được khách hàng càng lâu doanh nghiệp của bạn sẽ càng tiết kiệm được chi phí và tăng doanh thu.

Gợi ý

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Đọc dữ liệu, tiền xử lý dữ liệu

In [2]:
```
data = pd.read_csv("Churn_Modelling.csv")
```

In [3]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
RowNumber          10000 non-null int64
CustomerId         10000 non-null int64
Surname            10000 non-null object
CreditScore        10000 non-null int64
Geography          10000 non-null object
Gender             10000 non-null object
Age                10000 non-null int64
Tenure             10000 non-null int64
Balance            10000 non-null float64
NumOfProducts      10000 non-null int64
HasCrCard          10000 non-null int64
IsActiveMember     10000 non-null int64
EstimatedSalary    10000 non-null float64
Exited             10000 non-null int64
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```

In [4]:
```
data.head()
```

Out[4]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance
0	1	15634602	Hargrave	619	France	Female	42	2	0.00
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86
2	3	15619304	Onio	502	France	Female	42	8	159660.80
3	4	15701354	Boni	699	France	Female	39	1	0.00
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82

In [5]:
```
# Dựa trên thông tin trên ta thấy các cột không dùng trong model là:
# RowNumber, CustomerId, Surname
# inputs: các cột còn lại trừ cột Exited
# output: cột Exited
```

In [6]:
```
X = data.iloc[:, 3:13]
y = data.iloc[:, 13]
```

In [7]:
```
X.head()
```

Out[7]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActive
0	619	France	Female	42	2	0.00	1	1	
1	608	Spain	Female	41	1	83807.86	1	0	
2	502	France	Female	42	8	159660.80	3	1	
3	699	France	Female	39	1	0.00	2	0	
4	850	Spain	Female	43	2	125510.82	1	1	

In [8]:
```
y.where(y==0).count() # khách hàng ở lại
```

Out[8]:
```
7963
```

In [9]:
```
y.where(y==1).count() # khách hàng ra đi
```

Out[9]:
```
2037
```

In [10]:
```
# Các thuộc tính phân loại
objects = [f for f in X.columns if X.dtypes[f] == 'object']
objects
```

Out[10]:
```
['Geography', 'Gender']
```

In [11]:
```
# Xem xét thuộc tính phân loại: Geography
X.groupby(by='Geography')['CreditScore'].count()
```

Out[11]:
```
Geography
France     5014
Germany    2509
Spain      2477
Name: CreditScore, dtype: int64
```

In [12]:
```
# Dựa trên kết quả ta thấy có 3 quốc gia
# => cần chuyển sang dữ liệu kiểu số
```

In [13]:
```
# Xem xét thuộc tính phân loại: Gender
X.groupby(by='Gender')['CreditScore'].count()
```

Out[13]:
```
Gender
Female    4543
Male      5457
Name: CreditScore, dtype: int64
```

In [14]:
```
# Dựa trên kết quả ta thấy có 2 giới tính
# => cần chuyển sang dữ liệu kiểu số
```

In [15]:
```
X_new = pd.get_dummies(X)
```

In [16]:
```
X_new.head()
```

Out[16]:

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSa
0	619	42	2	0.00	1	1	1	101348
1	608	41	1	83807.86	1	0	1	112542
2	502	42	8	159660.80	3	1	0	113931
3	699	39	1	0.00	2	0	0	93826
4	850	43	2	125510.82	1	1	1	79084

In [17]:
```
# from sklearn.preprocessing import StandardScaler
# sc = StandardScaler()
```

In [18]:
```
# X_new_1 = sc.fit_transform(X_new)
```

Áp dụng model, nhận xét kết quả

In [19]:
```
from sklearn.model_selection import train_test_split
```

In [20]:
```
X_train, X_test, y_train, y_test = train_test_split(X_new, y,
                                                    test_size = 0.2,
                                                    random_state = 0)
```

In [21]:
```
from sklearn.linear_model import LogisticRegression
```

In [22]:
```
lr = LogisticRegression()
```

In [23]:
```
lr.fit(X_train, y_train)
```

```
c:\program files\python36\lib\site-packages\sklearn\linear_model\logistic.py:43
2: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
```

Out[23]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [24]:
```
print('Train score: ', lr.score(X_train,y_train))
```

```
Train score:  0.788625
```

In [25]:
```
print('Test score: ', lr.score(X_test,y_test))
```

```
Test score:  0.786
```

In [26]:
```
yhat_test = lr.predict(X_test)
yhat_test
```

Out[26]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [27]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score
```

In [28]:
```
print("Test Accuracy is ", accuracy_score(y_test,yhat_test)*100,"%")
```

```
Test Accuracy is  78.60000000000001 %
```

In [29]:
```
from sklearn.metrics import confusion_matrix, precision_score, recall_score
```

In [30]:
```
cm = confusion_matrix(y_test, yhat_test)
```

In [31]:
```
cm
```

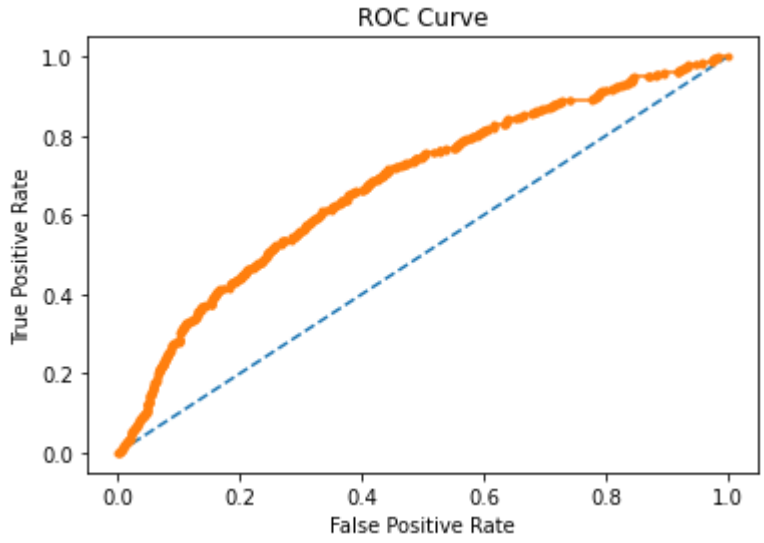
Out[31]:
```
array([[1544,   51],
       [ 377,   28]], dtype=int64)
```

In [32]:
```
from sklearn.metrics import roc_curve,auc
```

In [33]:
```
# Print ROC_AUC score using probabilities
probs = lr.predict_proba(X_test)
```

In [34]:
```
scores = probs[:,1]
fpr, tpr, thresholds = roc_curve(y_test, scores)
```

In [35]:
```
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 [36]:
```
auc(fpr, tpr)
```

Out[36]:
```
0.6781191222570533
```

Predicting new samples

In [37]:
```
# Cần phải chuẩn hóa dữ liệu để mẫu mới có cùng cấu trúc với dữ liệu đang có
```

In [38]:
```
columns = X_new.columns
columns
```

Out[38]:
```
Index(['CreditScore', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Geography_France',
       'Geography_Germany', 'Geography_Spain', 'Gender_Female', 'Gender_Male'],
      dtype='object')
```

In [39]:
```
new_samples = X.iloc[[0, 1, 2]]
```

In [40]:
```
new_samples =  pd.get_dummies(new_samples)
```

In [41]:
```
new_samples.columns
```

Out[41]:
```
Index(['CreditScore', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Geography_France',
       'Geography_Spain', 'Gender_Female'],
      dtype='object')
```

In [42]:
```
missing_cols = set(X_new.columns) - set(new_samples.columns)
missing_cols
```

Out[42]:
```
{'Gender_Male', 'Geography_Germany'}
```

In [43]:
```
for c in missing_cols:
    new_samples[c] = 0
# Ensure the order of column in the test set
# is in the same order than in train set
new_samples = new_samples[X_new.columns]
```

In [44]:
```
new_samples.columns
```

Out[44]:
```
Index(['CreditScore', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Geography_France',
       'Geography_Germany', 'Geography_Spain', 'Gender_Female', 'Gender_Male'],
      dtype='object')
```

In [45]:
```
#new_samples = sc.transform(new_samples)
```

In [46]:
```
new_predictions = lr.predict(new_samples)
new_predictions
```

Out[46]:
```
array([0, 0, 0], dtype=int64)
```

In [47]:
```
# Nhận xét kết quả
# Có giải pháp nào giúp cho kết quả cải thiện hơn không?
```

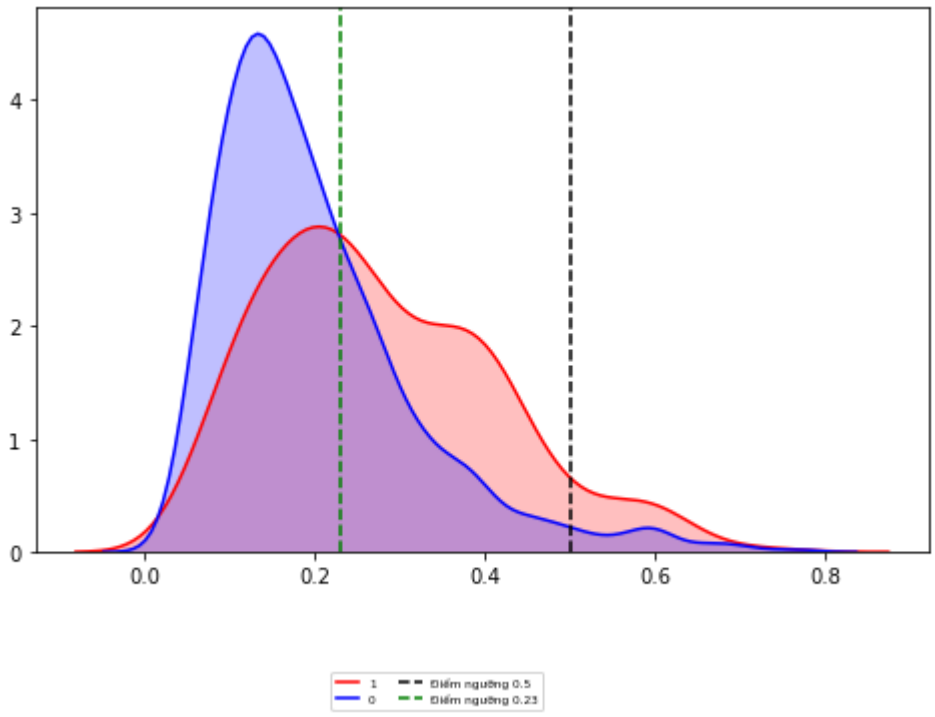
Giải pháp

Đề xuất 1: Điều chỉnh ngưỡng

In [48]:
```
# Dự đoán xác suất khách hàng bỏ đi
pos_label= 1
pos_index= np.where(lr.classes_ == pos_label)[0][0]
neg_index= np.where(lr.classes_ != pos_label)[0][0]
neg_label= lr.classes_[neg_index]
# Dự đoán xác xuất
y_predict_proba= lr.predict_proba(X_test)
# Tách xác xuất khách hàng bỏ đi
pos_proba= y_predict_proba[:, pos_index]
```

In [49]:
```
import seaborn as sns
```

In [50]:
```
# Trực quan phân phối xác suất khách hàng ra đi để điều chỉnh điểm ngưỡng
plt.figure(figsize= (8, 5))
sns.distplot(pos_proba[y_test== pos_label], label= pos_label,
             color= 'r', kde_kws={"shade": True}, hist=False, norm_hist= True)
sns.distplot(pos_proba[y_test== neg_label], label= neg_label,
             color= 'b', kde_kws={"shade": True}, hist=False, norm_hist= True)
plt.axvline(x= 0.5, linestyle= '--', color= 'black', label= 'Điểm ngưỡng 0.5')
plt.axvline(x= 0.23, linestyle= '--', color= 'g', label= 'Điểm ngưỡng 0.23', )
plt.legend(fontsize= 'xx-small', bbox_to_anchor=(0.3, -0.3, 0.3, 0.2),
           ncol=2, loc='lower center')
plt.show()
```



In [51]:
```
# Với ngưỡng mới: 0.23
```

In [52]:
```
y_hat_test_pro = lr.predict_proba(X_test)
```

In [53]:
```
y_hat_test_pro[:5]
```

Out[53]:
```
array([[0.79754585, 0.20245415],
       [0.64651623, 0.35348377],
       [0.82521651, 0.17478349],
       [0.90879767, 0.09120233],
       [0.8208282 , 0.1791718 ]])
```

In [54]:
```
y_hat_now = y_hat_test_pro[:,1] > 0.23
print(y_hat_now)
```

```
[False  True False ...  True False False]
```

In [55]:
```
print("Test Accuracy is ", accuracy_score(y_test,y_hat_now)*100,"%")
```

```
Test Accuracy is  66.95 %
```

In [56]:
```
cm1 = confusion_matrix(y_test, y_hat_now)
cm1
```

Out[56]:
```
array([[1109,  486],
       [ 175,  230]], dtype=int64)
```

In [57]:
```
fpr1, tpr1, thresholds1 = roc_curve(y_test, y_hat_now)
```

In [58]:
```
auc(fpr1, tpr1)
```

Out[58]:
```
0.6315995201052671
```

Predicting new samples with new threshold

In [59]:
```
lr.predict_proba(new_samples)[:,1]>0.23
```

Out[59]:
```
array([False,  True,  True])
```

In [60]:
```
# Nhận xét kết quả
# Có giải pháp nào giúp cho kết quả cải thiện hơn không?
```

Đề xuất 2: Resampling

In [61]:
```
from imblearn.over_sampling import SMOTE
method = SMOTE(kind='borderline1')
```

```
Using TensorFlow backend.
```

In [62]:
```
# Apply resampling to the training data only
X_resampled, y_resampled = method.fit_sample(X_train, y_train)
```

In [63]:
```
# Count the occurrences of fraud and no fraud and print them
occ_no = y_resampled[y_resampled==0].size
print(occ_no)
```

```
6368
```

In [64]:
```
occ_fraud = y_resampled[y_resampled==1].size
print(occ_fraud)
```

```
6368
```

In [65]:
```
# Continue fitting the model and obtain predictions
model = LogisticRegression()
model.fit(X_resampled, y_resampled)
```

```
c:\program files\python36\lib\site-packages\sklearn\linear_model\logistic.py:43
2: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
```

Out[65]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [66]:
```
# training score
model.score(X_resampled, y_resampled)
```

Out[66]:
```
0.6812971105527639
```

In [67]:
```
# testing score
model.score(X_test, y_test)
```

Out[67]:
```
0.666
```

In [68]:
```
# Get your performance metrics
y_pred = model.predict(X_test)
```

In [69]:
```
conf_mat = confusion_matrix(y_true=y_test, y_pred=y_pred)
print('Confusion matrix:\n', conf_mat)
```

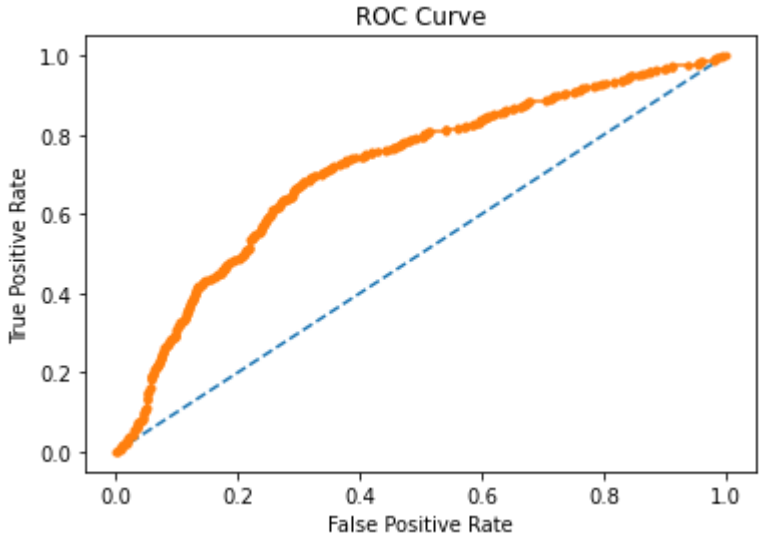
```
Confusion matrix:
 [[1045  550]
 [ 118  287]]
```

In [70]:
```
# Print ROC_AUC score using probabilities
probs = model.predict_proba(X_test)
```

In [71]:
```
from sklearn.metrics import roc_curve, auc
```

In [72]:
```
scores = model. predict_proba(X_test)[:,1]
fpr, tpr, thresholds = roc_curve(y_test, scores)
```

In [73]:
```
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 [74]:
```
auc(fpr, tpr)
```

Out[74]:
```
0.714701033321723
```

Predicting new samples

In [75]:
```
new_predictions = model.predict(new_samples)
new_predictions
```

Out[75]:
```
array([0, 1, 1], dtype=int64)
```

Kết luận:

- Kết quả nào phù hợp hơn với bài toán này? Tại sao?
- Nếu chưa tìm được giải pháp nào phù hợp hơn thì có thể nghĩ đến việc phải thay đổi thuật toán (sẽ học sau)