Chapter4 - Exercise 1: Click Prediction

Cho dữ liệu Click_prediction_small (link: https://www.openml.org/d/1220 (https://www.openml.org/d/1220) hoặc sklearn.datasets import fetch_openml với data_id = 1220).

Yêu cầu: Hãy đọc dữ liệu từ tập tin này, áp dụng Naive Bayes để thực hiện việc xác định người dùng có click vào một quảng cáo hay không dựa trên các thông tin được cung cấp.

Yêu cầu:

1. Hãy chuẩn hóa dữ liệu cho phù hợp
2. Áp dụng Naive Bayes. Nhận xét kết quả.
3. Model trên có phù hợp không? Nếu không thì đưa ra phương án khác.

In [1]:

```
import pandas as pd
from sklearn.datasets import fetch_openml
```

In [2]:

```
data = fetch_openml(data_id=1220)

X = pd.DataFrame(
    data['data'],
    columns=data['feature_names']
).astype(float)

y = pd.Series(data['target']).astype(int)
```

In [3]:

```
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 39948 entries, 0 to 39947
Data columns (total 9 columns):
impression        39948 non-null float64
ad_id             39948 non-null float64
advertiser_id     39948 non-null float64
depth             39948 non-null float64
position          39948 non-null float64
keyword_id        39948 non-null float64
title_id          39948 non-null float64
description_id    39948 non-null float64
user_id           39948 non-null float64
dtypes: float64(9)
memory usage: 2.7 MB
```

In [4]:

```
#X = X.drop('user_id', axis=1)
```

In [5]:
```
X.head()
```

Out[5]:

	impression	ad_id	advertiser_id	depth	position	keyword_id	title_id	description_id	
0	1.0	8343295.0	11700.0	3.0	3.0	21264.0	27892.0	1559.0	
1	1.0	20017077.0	23798.0	1.0	1.0	35498.0	4.0	36476.0	5
2	1.0	21348354.0	36654.0	1.0	1.0	19975.0	36105.0	33292.0	116
3	1.0	20366086.0	33280.0	3.0	3.0	5942.0	4057.0	4390.0	87
4	1.0	6803526.0	10790.0	2.0	1.0	60593.0	25242.0	1679.0	121

In [6]:
```
y.head()
```

Out[6]:
```
0    0
1    1
2    0
3    0
4    0
dtype: int32
```

In [7]:
```
#target count: 1: click, 0: not click
y.value_counts()
```

Out[7]:
```
0    33220
1     6728
dtype: int64
```

In [8]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.25,
                                                    random_state=42)
```

In [9]:
```
from sklearn.naive_bayes import GaussianNB
import numpy as np
from sklearn.utils.validation import column_or_1d
```

In [10]:
```
#Create a Gaussian Classifier
model = GaussianNB()
# Train the model using the training sets
model.fit(X_train, y_train)
```

Out[10]:
```
GaussianNB(priors=None, var_smoothing=1e-09)
```

In [11]:
```
model.score(X_train, y_train)
```

Out[11]:
```
0.8086512466206068
```

In [12]:
```
model.score(X_test, y_test)
```

Out[12]:
```
0.8078502052668469
```

In [13]:

```
y_pred = model.predict(X_test)
```

In [14]:

```
from sklearn.metrics import accuracy_score
print("Accuracy is ", accuracy_score(y_test,y_pred)*100,"%")
```

```
Accuracy is  80.78502052668469 %
```

In [15]:

```
from sklearn.metrics import confusion_matrix
```

In [16]:

```
confusion_matrix(y_test, y_pred)
```

Out[16]:

```
array([[7992,  313],
       [1606,   76]], dtype=int64)
```

In [17]:

```
# Đánh giá model
from sklearn. metrics import classification_report, roc_auc_score, roc_curve
```

In [18]:

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.83      0.96      0.89      8305
           1       0.20      0.05      0.07      1682

    accuracy                           0.81      9987
   macro avg       0.51      0.50      0.48      9987
weighted avg       0.73      0.81      0.75      9987
```

In [19]:

```
y_prob = model.predict_proba(X_test)
y_prob
```

Out[19]:

```
array([[0.90958662, 0.09041338],
       [0.8898664 , 0.1101336 ],
       [0.85271593, 0.14728407],
       ...,
       [0.88644901, 0.11355099],
       [0.9088923 , 0.0911077 ],
       [0.76539178, 0.23460822]])
```

In [20]:

```
roc_auc_score(y_test, y_prob[:, 1])
```

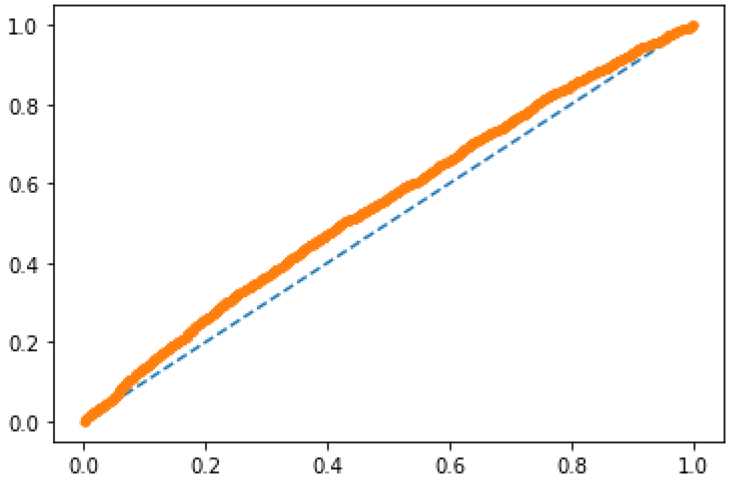
Out[20]:

```
0.549416529875775
```

In [21]:

```
import matplotlib.pyplot as plt
```

In [22]:
```
# calculate roc curve
fpr, tpr, thresholds = roc_curve(y_test, y_prob[:, 1])
# plot no skill
plt.plot([0, 1], [0, 1], linestyle='--')
plt.plot(fpr, tpr, marker='.')
plt.show()
```



In [1]:
```
# Dựa trên kết quả trên ta thấy precision và recall của class 1 đều rất thấp
# ROC thấp
# Dựa trên tất cả các đánh giá
# => model chưa phù hợp có thể do mất cân bằng dữ liệu ???
```

Resampling data

In [24]:
```
from imblearn.under_sampling import RandomUnderSampler
from imblearn.over_sampling import SMOTE
```

```
Using TensorFlow backend.
```

In [25]:
```
rus = RandomUnderSampler()
X_resampled, y_resampled = rus.fit_resample(X, y)
```

In [26]:
```
unique_elements, counts_elements = np.unique(y_resampled, return_counts=True)
print("Frequency of each category:")
print(np.asarray((unique_elements, counts_elements)))
```

```
Frequency of each category:
[[   0    1]
 [6728 6728]]
```

In [27]:
```
X_train_r, X_test_r, y_train_r, y_test_r = train_test_split(X_resampled,
                                                            y_resampled,
                                                    test_size=0.25,
                                                    random_state=42)
```

In [28]:
```
#Create a Gaussian Classifier
model_r = GaussianNB()
# Train the model using the training sets
model_r.fit(X_train_r, y_train_r)
```

Out[28]:
```
GaussianNB(priors=None, var_smoothing=1e-09)
```

In [29]:
```
y_pred_r = model.predict(X_test_r)
```

In [30]:
```
confusion_matrix(y_test_r, y_pred_r)
```

Out[30]:
```
array([[1633,   68],
       [1596,   67]], dtype=int64)
```

In [31]:
```
print(classification_report(y_test_r, y_pred_r))
```

```
              precision    recall  f1-score   support

           0       0.51      0.96      0.66      1701
           1       0.50      0.04      0.07      1663

    accuracy                           0.51      3364
   macro avg       0.50      0.50      0.37      3364
weighted avg       0.50      0.51      0.37      3364
```

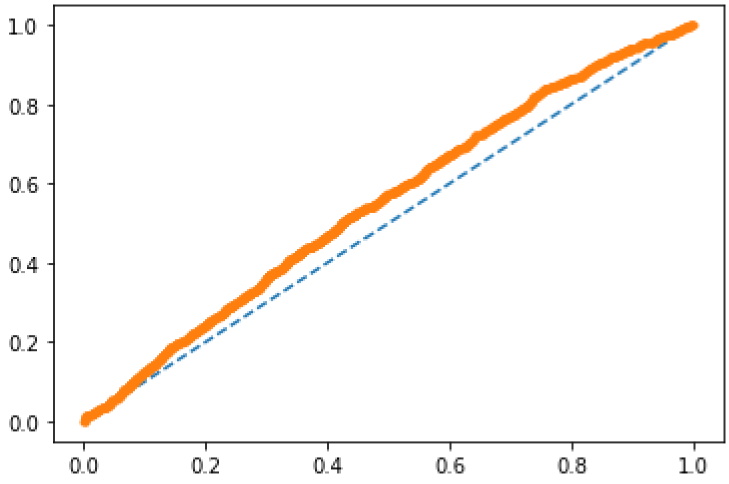
In [32]:
```
y_prob_r = model.predict_proba(X_test_r)
y_prob_r
```

Out[32]:
```
array([[9.14505765e-01, 8.54942351e-02],
       [9.10578135e-01, 8.94218651e-02],
       [9.50145867e-09, 9.99999990e-01],
       ...,
       [9.16230159e-01, 8.37698413e-02],
       [1.64622602e-02, 9.83537740e-01],
       [8.93668515e-01, 1.06331485e-01]])
```

In [33]:
```
roc_auc_score(y_test_r, y_prob_r[:, 1])
```

Out[33]:
```
0.5521173035705006
```

In [34]:
```
# calculate roc curve
fpr, tpr, thresholds = roc_curve(y_test_r, y_prob_r[:, 1])
# plot no skill
plt.plot([0, 1], [0, 1], linestyle='--')
plt.plot(fpr, tpr, marker='.')
plt.show()
```



In [35]:
```
# Kết quả với resampling cũng không tốt hơn
# => thay đổi thuật toán khác???
# Thống kê về các giải pháp và kết quả: https://www.openml.org/t/7295
```

In [36]:
```
# import pickle
# # Save to file in the current working directory
# pkl_filename = "pickle_model.pkl"
# with open(pkl_filename, 'wb') as file:
#     pickle.dump(model, file)
```

In [37]:
```
# with open(pkl_filename, 'rb') as file:
#     pickle_model = pickle.load(file)
```