Chapter 7 - Ex2: Adult Dataset - Full

- Adult Dataset được cung cấp bởi UCI (University of California, Irvine) được sử dụng để phát triển mô hình dự đoán Predictive Model Development.
- Bộ dữ liệu adult.data và adult.test chứa 48.842 mẫu và có 14 attributes/features. Dữ liệu này được dùng để xây dựng model dự đoán và kiểm tra một mẫu có thu nhập >50K USD hay không. ### Attribute Information:
- age: continuous.
- workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.
- fnlwgt: continuous.
- education: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.
- education-num: continuous.
- marital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.
- occupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.
- relationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
- race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.
- sex: Female, Male.
- capital-gain: continuous.
- capital-loss: continuous.
- hours-per-week: continuous.
- native-country: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.
- Class: >50K, <=50K.

Yêu cầu:

- Đọc dữ liệu adult.data, tiền xử lý dữ liệu.
- Xem xét tính cân bằng giữa hai loại mẫu. Trực quan hóa. Nhận xét.
- Nếu 2 loại mẫu này không cân bằng, hãy chọn một phương pháp cân bằng dữ liệu và thực hiện. Trực quan hóa kết quả.

In [1]:

```
# link tham khảo: https://towardsdatascience.com/under-sampling-a-performance-booster-on-imbalance
```

In [2]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
# Đọc dữ liệu, kiểm tra sơ bộ bau đầu, trực quan hóa, tiền xử lý dữ liệu
adult_train = pd.read_csv("adult/adult.data", header=None)
```

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

Out[4]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K

In [5]:
```
adult_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
0     32561 non-null int64
1     32561 non-null object
2     32561 non-null int64
3     32561 non-null object
4     32561 non-null int64
5     32561 non-null object
6     32561 non-null object
7     32561 non-null object
8     32561 non-null object
9     32561 non-null object
10    32561 non-null int64
11    32561 non-null int64
12    32561 non-null int64
13    32561 non-null object
14    32561 non-null object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

In [6]:
```
adult_train.to_csv("aldult_data.csv")
```

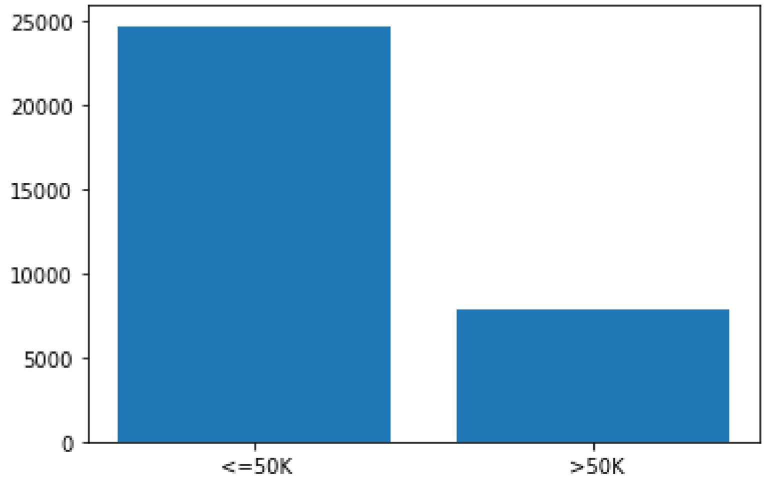
In [7]:
```
# Không có dữ liệu null
```

In [8]:
```
# Đếm theo loại: hiếm, phổ biến
occ = adult_train[14].value_counts()
occ
```

Out[8]:
```
<=50K    24720
>50K      7841
Name: 14, dtype: int64
```

In [9]:
```
plt.bar(occ.index.values, occ.values)
```

Out[9]:
```
<BarContainer object of 2 artists>
```



In [10]:
```
# Chuyển dữ liệu phân loại thành dạng numeric dùng Label encoder và dummy encoder
```

In [11]:
```
y_train = adult_train[14]
X_train = adult_train.drop([14], axis=1)
```

In [12]:
```
X_train.head(2)
```

Out[12]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States

In [13]:
```
y_train[:2]
```

Out[13]:
```
0      <=50K
1      <=50K
Name: 14, dtype: object
```

In [14]:
```
from sklearn.preprocessing import LabelEncoder
```

In [15]:
```
label_encoder = LabelEncoder()
y_train_l = label_encoder.fit_transform(y_train)
```

In [16]:
```
y_train_l[:2]
```

Out[16]:
```
array([0, 0])
```

In [17]:
```python
# Categorical boolean mask
categorical_feature_mask = X_train.dtypes==object
# filter categorical columns using mask and turn it into a list
categorical_cols = X_train.columns[categorical_feature_mask].tolist()
categorical_cols
```

Out[17]:
```
[1, 3, 5, 6, 7, 8, 9, 13]
```

In [18]:
```python
X_train_d = pd.get_dummies(data=X_train, columns=categorical_cols, drop_first=True)
```

In [19]:
```python
X_train_d.head(2)
```

Out[19]:

	0	2	4	10	11	12	1_ Federal-gov	1_ Local-gov	1_ Never-worked	1_ Private	...	13_ Portugal	13_ Puerto-Rico	13_ Scotland	13_ South	13_ Taiwan
0	39	77516	13	2174	0	40	0	0	0	0	...	0	0	0	0	0
1	50	83311	13	0	0	13	0	0	0	0	...	0	0	0	0	0

2 rows × 100 columns

Áp dụng thuật toán với dữ liệu gốc

In [20]:
```python
from sklearn.linear_model import LogisticRegression
```

In [21]:
```python
model = LogisticRegression()
```

In [22]:
```python
model.fit(X_train_d, y_train_l)
```

Out[22]:
```
LogisticRegression()
```

In [23]:
```python
from sklearn.metrics import confusion_matrix, accuracy_score
```

In [24]:
```python
y_pred = model.predict(X_train_d)
```

In [25]:
```python
accuracy_score(y_train_l, y_pred)
```

Out[25]:
```
0.7957679432449863
```

In [26]:
```python
cm = confusion_matrix(y_train_l, y_pred)
```

In [27]:
```python
cm
```

```
array([[23842,   878],
```

Out[27]:

```
       [ 5772,  2069]], dtype=int64)
```

In [28]:

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

In [29]:

```
print(classification_report(y_train_l, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.81      0.96      0.88     24720
           1       0.70      0.26      0.38      7841

    accuracy                           0.80     32561
   macro avg       0.75      0.61      0.63     32561
weighted avg       0.78      0.80      0.76     32561
```

In [30]:

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

Out[30]:

```
array([[0.45215974, 0.54784026],
       [0.63260177, 0.36739823],
       [0.80282842, 0.19717158],
       ...,
       [0.72915332, 0.27084668],
       [0.78767922, 0.21232078],
       [0.04969381, 0.95030619]])
```

In [31]:

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

Out[31]:

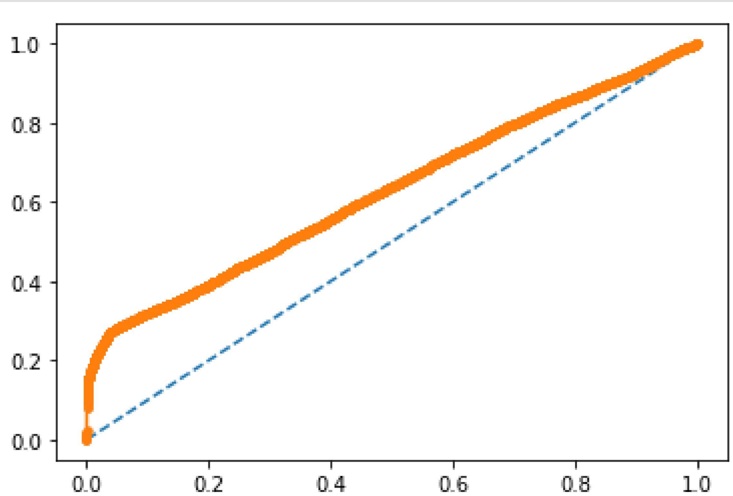
```
0.6291047901269116
```

In [32]:

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

In [33]:

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



Kết luận

- ROC_AUC thấp
- precision class 1 cao nhưng recall thấp

Undersampling

In [34]:
```
from collections import Counter
sorted(Counter(y_train_l).items())
```

Out[34]:
```
[(0, 24720), (1, 7841)]
```

In [35]:
```
# Vì lượng dữ liệu class 1 tương đối nhiều => do đó ta sẽ áp dụng Undersampling
# để giảm số mẫu của nhóm <=50k bằng với nhóm >50k
```

In [36]:
```
from sklearn.utils import resample
```

In [37]:
```
# có thể dùng cách resample
```

In [38]:
```
data_train = X_train_d
data_train[14] = y_train_l
```

In [39]:
```
data_0 = data_train[data_train[14]==0]
data_1 = data_train[data_train[14]==1]
```

In [40]:
```
display(data_0.shape, data_1.shape)
```

```
(24720, 101)
(7841, 101)
```

In [41]:
```
from sklearn.utils import resample
```

In [42]:
```
data_0_resample = resample(data_0,
                           replace = False, # sample without replacement
                           n_samples = data_1.shape[0], # match minority n
                           random_state = 27) # reproducible results
```

In [43]:
```
downsampled = pd.concat([data_0_resample, data_1])
downsampled.head()
```

Out[43]:

	0	2	4	10	11	12	1_ Federal-gov	1_ Local-gov	1_ Never-worked	1_ Private	...	13_ Puerto-Rico	13_ Scotland	13_ South	13_ Taiwan	Thaila
31749	22	199426	10	0	0	17	0	0	0	1	...	0	0	0	0	
24093	31	91964	13	0	0	40	0	0	0	1	...	0	0	0	0	

	0	2	4	10	11	12	1_ Federal-gov	1_ Local-gov	1_ Never-worked	1_ Private	...	13_ Puerto-Rico	13_ Scotland	13_ South	13_ Taiwan	Thail
21539	37	60313	9	0	0	40	0	0	0	1	...	0	0	0	0	
24582	30	85708	9	0	0	40	0	0	0	1	...	0	0	0	0	
622	65	109351	5	0	0	24	0	0	0	1	...	0	0	0	0	

5 rows × 101 columns

In [44]:
```
display(data_0_resample.shape, data_1.shape)
```

```
(7841, 101)
(7841, 101)
```

Áp dụng thuật toán với dữ liệu Undersampling

In [45]:
```
X_down = downsampled.drop(14, axis=1)
```

In [46]:
```
y_down= downsampled[14]
```

In [47]:
```
model_down = LogisticRegression()
```

In [48]:
```
model_down.fit(X_down, y_down)
```

```
c:\program files\python36\lib\site-packages\sklearn\linear_model\_logistic.py:764: ConvergenceWarn
ing: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

Out[48]:
```
LogisticRegression()
```

In [49]:
```
y_pred_d = model_down.predict(X_down)
```

In [50]:
```
accuracy_score(y_down, y_pred_d)
```

Out[50]:
```
0.6840326488968244
```

In [51]:
```
cm = confusion_matrix(y_down, y_pred_d)
```

In [52]:
```
cm
```

Out[52]:
```
array([[5263, 2578],
```

```
        [2377, 5464]], dtype=int64)
```

In [53]:
```
print(classification_report(y_down, y_pred_d))
```

```
              precision    recall  f1-score   support

           0       0.69      0.67      0.68      7841
           1       0.68      0.70      0.69      7841

    accuracy                           0.68     15682
   macro avg       0.68      0.68      0.68     15682
weighted avg       0.68      0.68      0.68     15682
```

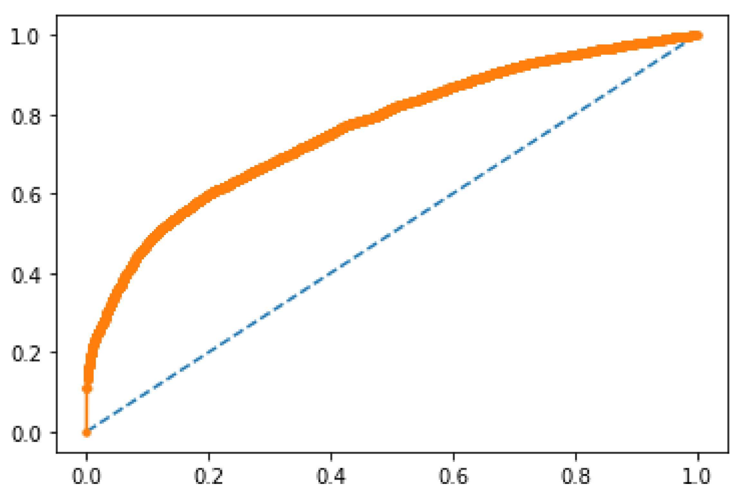
In [54]:
```
y_prob_d = model_down.predict_proba(X_down)
y_prob_d
```

Out[54]:
```
array([[0.47990917, 0.52009083],
       [0.33516886, 0.66483114],
       [0.6269155 , 0.3730845 ],
       ...,
       [0.3878897 , 0.6121103 ],
       [0.56140787, 0.43859213],
       [0.03907586, 0.96092414]])
```

In [55]:
```
roc_auc_score(y_down, y_prob_d[:, 1])
```

Out[55]:
```
0.7649199762119465
```

In [59]:
```
fpr, tpr, thresholds = roc_curve(y_down, y_prob_d[:, 1])
# plot no skill
plt.plot([0, 1], [0, 1], linestyle='--')
plt.plot(fpr, tpr, marker='.')
plt.show()
```



Kết luận

- ROC_AUC cao hơn so với dữ liệu gốc
- precision class 1 gần như dữ liệu gốc và recall cao hơn #### => Áp dụng Undersampling dữ liệu cho kết quả tốt hơn so với dữ liệu gốc ban đầu.

In [ ]: