Chapter 9 - Ex 1: Admit to university?

Xem xét việc có được vào trường đại học hay không dựa trên bộ dữ liệu sinh viên 400 mẫu có tên là binary.csv

Yêu cầu: Hãy đọc dữ liệu từ tập tin này, áp dụng Logistic Regression để thực hiện việc xác định có được vào trường đại học hay không dựa vào các thông tin như: gre, gpa, rank.

1. Đọc dữ liệu, tiền xử lý dữ liệu nếu cần, trực quan hóa dữ liệu để thấy sự tương quan giữa các biến
2. Tạo X_train, X_test, y_train, y_test từ dữ liệu đọc được với tỷ lệ dữ liệu test là 0.2
3. Áp dụng thuật toán Logistic Regression
4. Kiểm tra độ chính xác. Đánh giá mô hình bằng kiểm tra underfiting và overfiting
5. Tìm kết quả Cho dữ liệu Test: X_now = [[600, 4, 2],[400, 3, 3]]

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import math
```

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

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 4 columns):
admit    400 non-null int64
gre      400 non-null int64
gpa      400 non-null float64
rank     400 non-null int64
dtypes: float64(1), int64(3)
memory usage: 12.6 KB
```

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

Out[4]:

	admit	gre	gpa	rank
count	400.000000	400.000000	400.000000	400.00000
mean	0.317500	587.700000	3.389900	2.48500
std	0.466087	115.516536	0.380567	0.94446
min	0.000000	220.000000	2.260000	1.00000
25%	0.000000	520.000000	3.130000	2.00000
50%	0.000000	580.000000	3.395000	2.00000
75%	1.000000	660.000000	3.670000	3.00000
max	1.000000	800.000000	4.000000	4.00000

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

Out[5]:

	admit	gre	gpa	rank
0	0	380	3.61	3
1	1	660	3.67	3
2	1	800	4.00	1
3	1	640	3.19	4
4	0	520	2.93	4

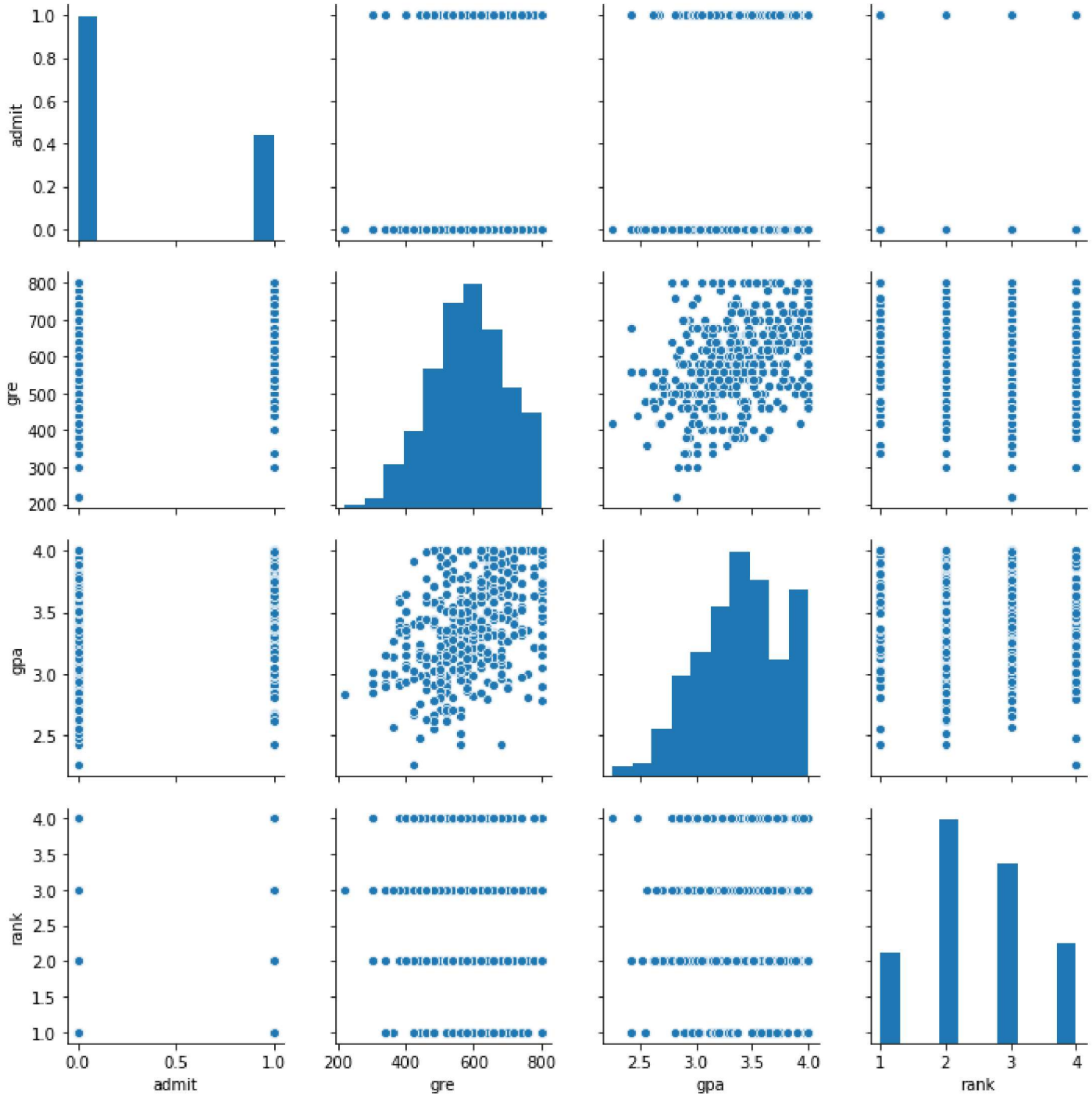
In [27]:
```
data.corr()
```

Out[27]:

	admit	gre	gpa	rank
admit	1.000000	0.184434	0.178212	-0.242513
gre	0.184434	1.000000	0.384266	-0.123447
gpa	0.178212	0.384266	1.000000	-0.057461
rank	-0.242513	-0.123447	-0.057461	1.000000

In [6]:
```
import seaborn as sns
```

In [7]:
```
sns.pairplot(data)
plt.show()
```



In [8]:
```
X = data[['gre', 'gpa', 'rank']]
X.head()
```

Out[8]:

	gre	gpa	rank
0	380	3.61	3
1	660	3.67	3
2	800	4.00	1
3	640	3.19	4
4	520	2.93	4

In [9]:
```
Y = data[['admit']]
```

```
Y.head()
```

Out[9]:

	admit
0	0
1	1
2	1
3	1
4	0

In [10]:
```
type(X)
```

Out[10]:
```
pandas.core.frame.DataFrame
```

In [11]:
```
X_train,X_test,Y_train,Y_test = train_test_split(X,
                                                 Y,
                                                 test_size=0.2)
```

In [12]:
```
from sklearn.linear_model import LogisticRegression
```

In [13]:
```
clf = LogisticRegression()
```

In [14]:
```
from sklearn.utils.validation import column_or_1d
```

In [15]:
```
clf.fit(X_train, column_or_1d(Y_train))
```

Out[15]:
```
LogisticRegression()
```

In [16]:
```
clf.intercept_
```

Out[16]:
```
array([-4.05455278])
```

In [17]:
```
clf.coef_
```

Out[17]:
```
array([[ 0.00236939,  0.92464962, -0.58525979]])
```

In [18]:
```
# Kiểm tra overfitting và underfitting
```

In [19]:
```
print('Score train: ', clf.score(X_train,Y_train))
```

```
Score train:  0.73125
```

In [20]:
```
print('Score test: ', clf.score(X_test,Y_test))
```

```
Score test:  0.625
```

In [ ]:
```
# Mô hình fit với training dataset hơn testing dataset.
# r^2 train > r^2 ~0.1 => có xu hướng overfitting
```

In [21]:
```
Yhat_train = clf.predict(X_train)
```

In [22]:
```
Yhat_test = clf.predict(X_test)
Yhat_test
```

Out[22]:
```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
       0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1], dtype=int64)
```

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

In [24]:
```
print("Accuracy is ", accuracy_score(Y_test,Yhat_test)*100,"%")
```

```
Accuracy is  62.5 %
```

In [25]:
```
# Đô chính xác của mô hình chưa cao
# Có giải pháp nào không?
```

In [26]:
```
X_now = [[600, 4, 2],[400, 3, 3]]
Y_now = clf.predict(X_now)
Y_now
```

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