Business Intelligence

Name：杜会远

Student ID：2016302580130

1.Legal issues for DATA MINING

Data mining will undoubtedly encounter several serious problems, among which privacy problems will take the leading position. One of the most famous examples is about the Prism Program. The American government violated numerous kinds of laws at the home and the broad and overlooked almost all the present rules in our world so that they can grab what they need. Meanwhile, as for our daily life, sometimes our crawlers may break the intellectual property law.

2.Skewness

The measure of how asymmetric a distribution can be is called skewness.

Provided with 1,2,2,3,3,3,4,4,5, we have skewness that is 0.

Provided with 1,2,2,2,3,3,4,4,5,5,6, we have skewness that is about 3.54.

Provided with 1,2,2,3,3,4,4,5,5,5,6, we have skewness that is about -5.52.

3. Box plot analysis

When it comes to the performance of each branch, I suppose that ***Branch 3*** is the best. Half of the unit price data is between \$140 and \$220 and it can be seen that this branch pays the most attention to the expensive products and also sells several products whose unit price is lower than \$60.

75% of the products of ***Branch 1*** has the unit price that is lower than \$100 but it still has a few expensive products.

Half of the products of ***Branch 2*** has the unit price that is between \$60 and \$100 and we can assume that this branch also has some high-level products because 25% of its products have the unit price between \$160 and \$180.

Branch 4 has interesting data that seems like normal distribution.

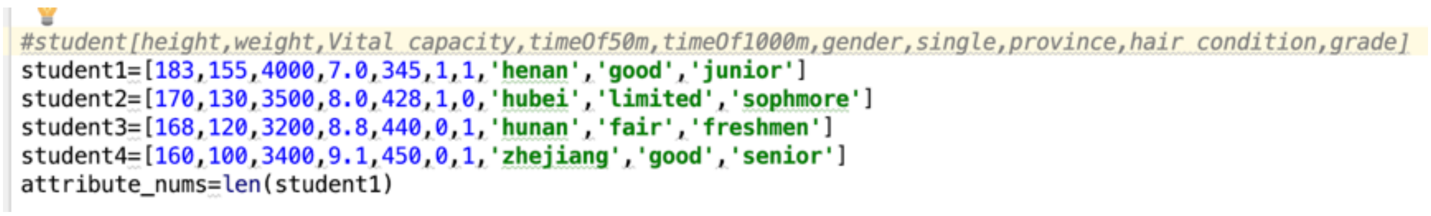
4. Dataset issues for Q-Q plot

Taking my practice into consideration, I used to modify the dataset with the help of `MinMaxScaler()` from `sklearn` , which is a common operation named normalization that transforms the data into a scale like `(0,1)` and also keep some of their attributes.

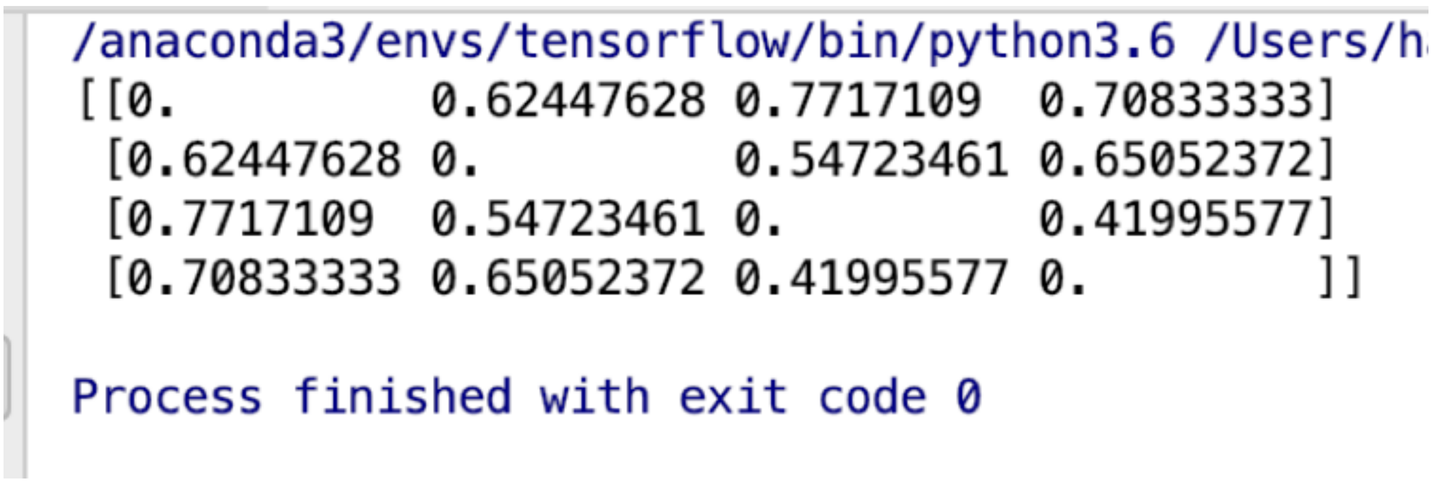
As for this problem, we may try to grab more data for the smaller one or wash some unsuitable data from the bigger one. Then I may find the **least common multiply** for the number of both datasets and transform the both the numbers to that degree. Then if necessary we can divide them and it seems that **the percentage of the data's position after sort** counts more.

5. Dissimilarity matrix

My data is about the sports performance of some students and the data concludes several types like **numeric attributes**, **binary attributes** and **ordinary attributes**. We have 4 students and each student has 10 attributes.



```
#student[height,weight,Vital capacity,timeOf50m,timeOf1000m,gender,single,province,hair condition,grade]
student1=[183,155,4000,7.0,345,1,1,'henan','good','junior']
student2=[170,130,3500,8.0,428,1,0,'hubei','limited','sophmore']
student3=[168,120,3200,8.8,440,0,1,'hunan','fair','freshmen']
student4=[160,100,3400,9.1,450,0,1,'zhejiang','good','senior']
attribute_nums=len(student1)
```



```
/anaconda3/envs/tensorflow/bin/python3.6 /Users/h
[[0.         0.62447628 0.7717109  0.70833333]
 [0.62447628 0.         0.54723461 0.65052372]
 [0.7717109  0.54723461 0.         0.41995577]
 [0.70833333 0.65052372 0.41995577 0.        ]]

Process finished with exit code 0
```

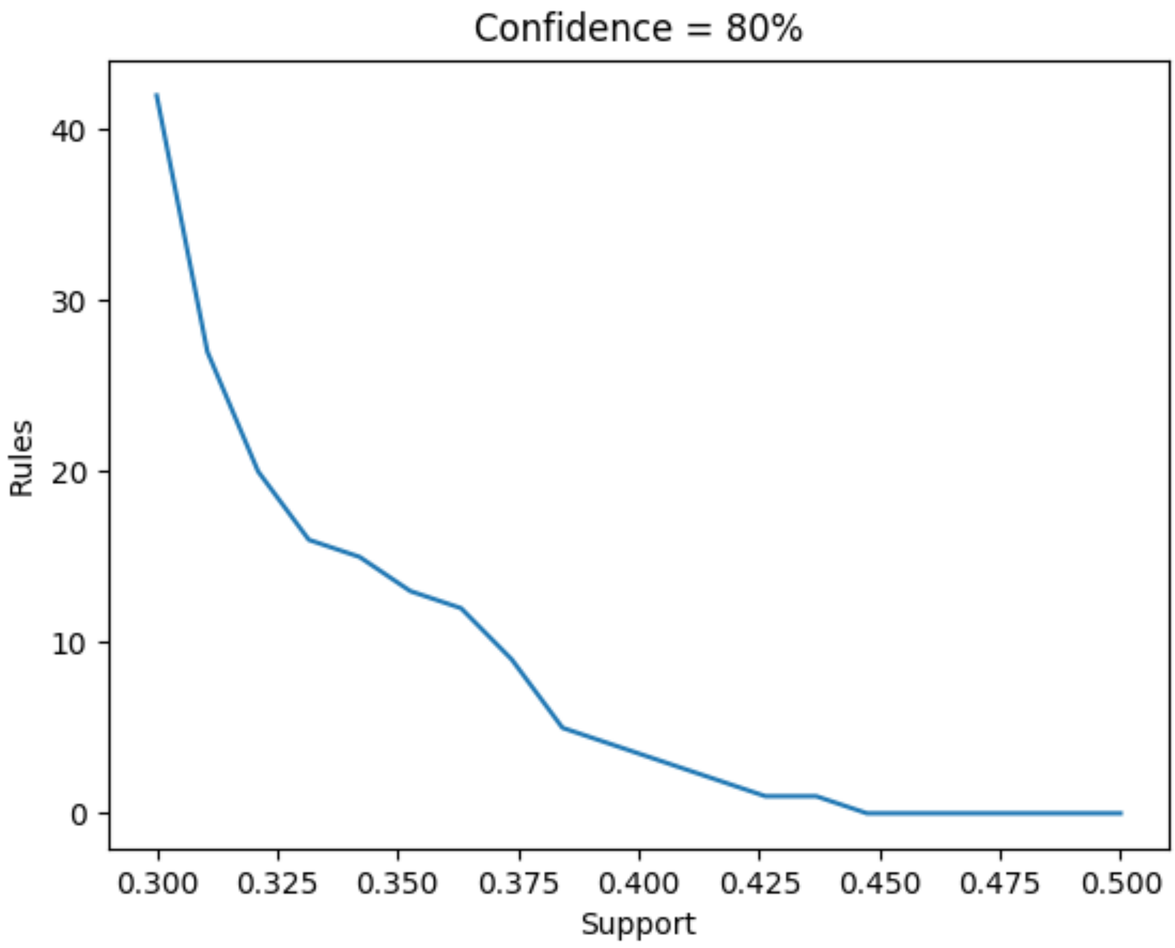
More details can be seen in `dissimilarity.py`

6. Apriori and FP-Growth

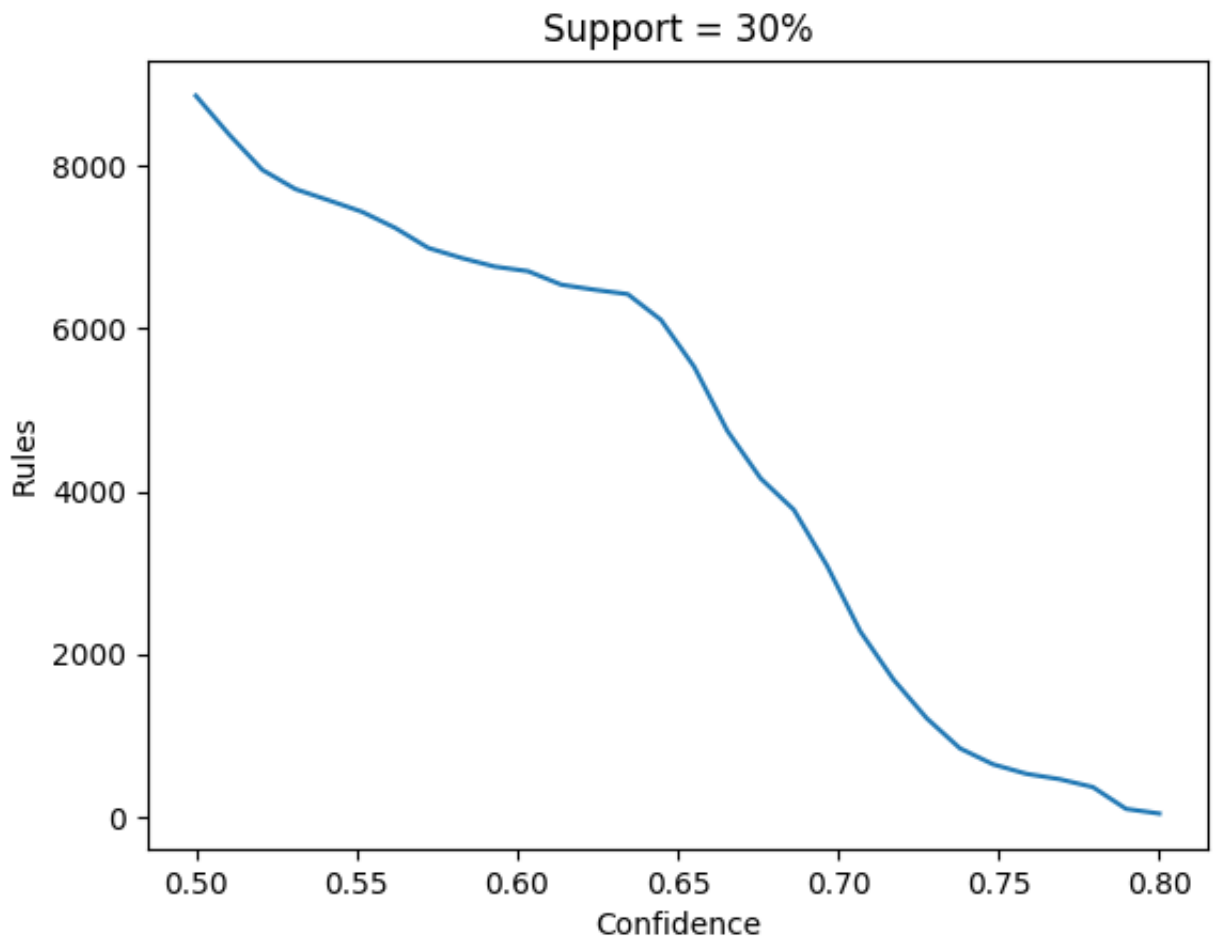
The data generation part is related to both algorisms. Assume that we have 100 products and we divide them into 5 kinds. For each kind of the products, I assume that the first one will appear randomly with the help of `possiblity_first` and `threshold`. Then we have `possibility1[i]` and `possibility2[i]` to imitate the possibility that you buy the i*th* product of this kind when you buy of not buy the first product of this kind.

I must say it's weird that there are few implementations for these two algorithms in common libraries. I suppose apriori is not that good for big data because it will iterate all the data in each loop but FP-Growth can be efficient to use for the search engines.

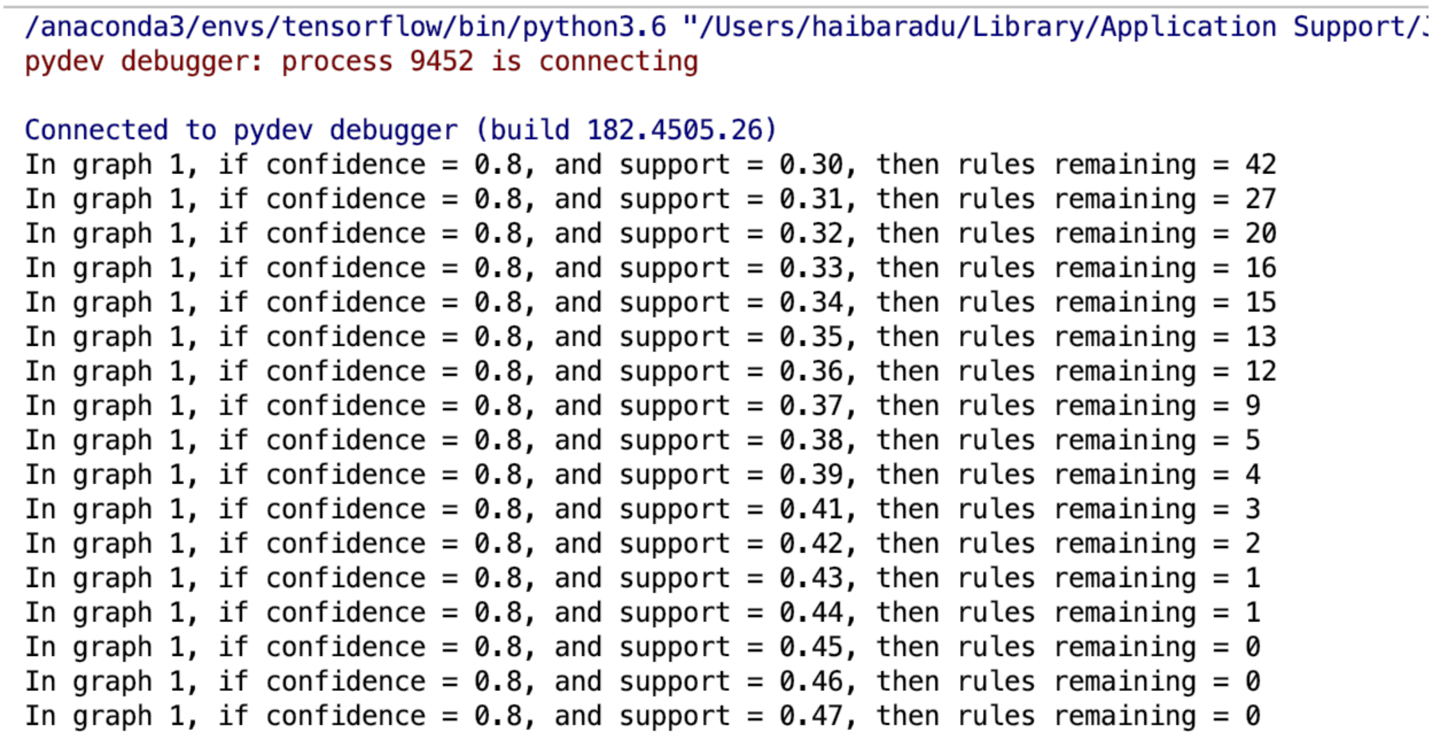
Here is part of the result of apriori algorithm.



Here is the result of FP-Growth algorithm.



As we can see, when the number of rules remaining is 20 and confidence ratio=0.8, we have support ratio=0.32.



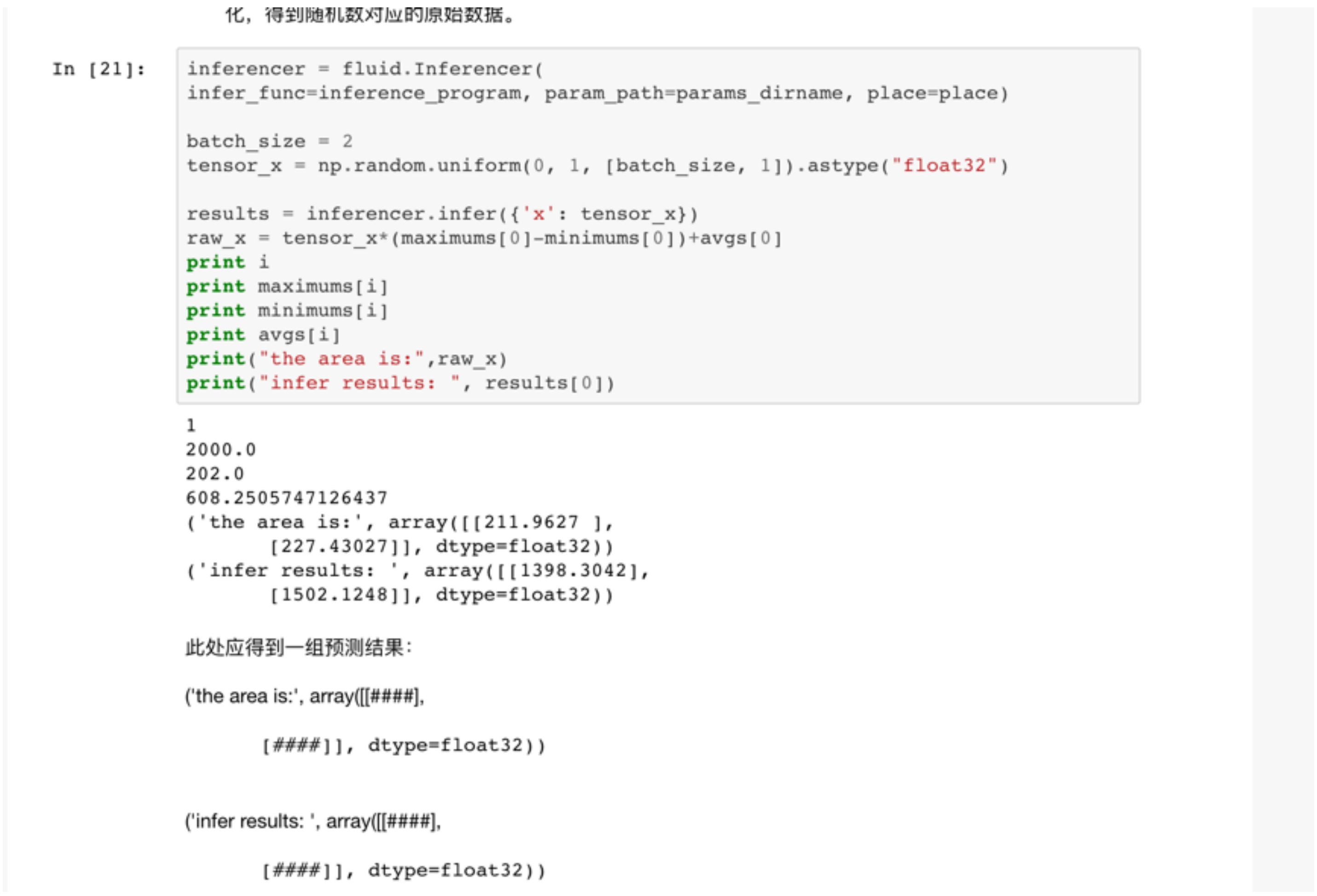
```
/anaconda3/envs/tensorflow/bin/python3.6 "/Users/haibaradu/Library/Application Support/
pydev debugger: process 9452 is connecting

Connected to pydev debugger (build 182.4505.26)
In graph 1, if confidence = 0.8, and support = 0.30, then rules remaining = 42
In graph 1, if confidence = 0.8, and support = 0.31, then rules remaining = 27
In graph 1, if confidence = 0.8, and support = 0.32, then rules remaining = 20
In graph 1, if confidence = 0.8, and support = 0.33, then rules remaining = 16
In graph 1, if confidence = 0.8, and support = 0.34, then rules remaining = 15
In graph 1, if confidence = 0.8, and support = 0.35, then rules remaining = 13
In graph 1, if confidence = 0.8, and support = 0.36, then rules remaining = 12
In graph 1, if confidence = 0.8, and support = 0.37, then rules remaining = 9
In graph 1, if confidence = 0.8, and support = 0.38, then rules remaining = 5
In graph 1, if confidence = 0.8, and support = 0.39, then rules remaining = 4
In graph 1, if confidence = 0.8, and support = 0.41, then rules remaining = 3
In graph 1, if confidence = 0.8, and support = 0.42, then rules remaining = 2
In graph 1, if confidence = 0.8, and support = 0.43, then rules remaining = 1
In graph 1, if confidence = 0.8, and support = 0.44, then rules remaining = 1
In graph 1, if confidence = 0.8, and support = 0.45, then rules remaining = 0
In graph 1, if confidence = 0.8, and support = 0.46, then rules remaining = 0
In graph 1, if confidence = 0.8, and support = 0.47, then rules remaining = 0
```

7. House Price Prediction

As for me, AiStudio is really like a new baby when compared with Kaggle and it's a disaster to install PaddlePaddle on my own laptop. The name of my online project is **House price predict** and my name is **HaibaraDu**.

Also here are some of the significant pictures.



化，得到随机数对应的原始数据。

In [21]:
```
inferencer = fluid.Inferencer(
infer_func=inference_program, param_path=params_dirname, place=place)

batch_size = 2
tensor_x = np.random.uniform(0, 1, [batch_size, 1]).astype("float32")

results = inferencer.infer({'x': tensor_x})
raw_x = tensor_x*(maximums[0]-minimums[0])+avgs[0]
print i
print maximums[i]
print minimums[i]
print avgs[i]
print("the area is:",raw_x)
print("infer results: ", results[0])
```

```
1
2000.0
202.0
608.2505747126437
('the area is:', array([[211.9627 ],
       [227.43027]], dtype=float32))
('infer results: ', array([[1398.3042],
       [1502.1248]], dtype=float32))
```

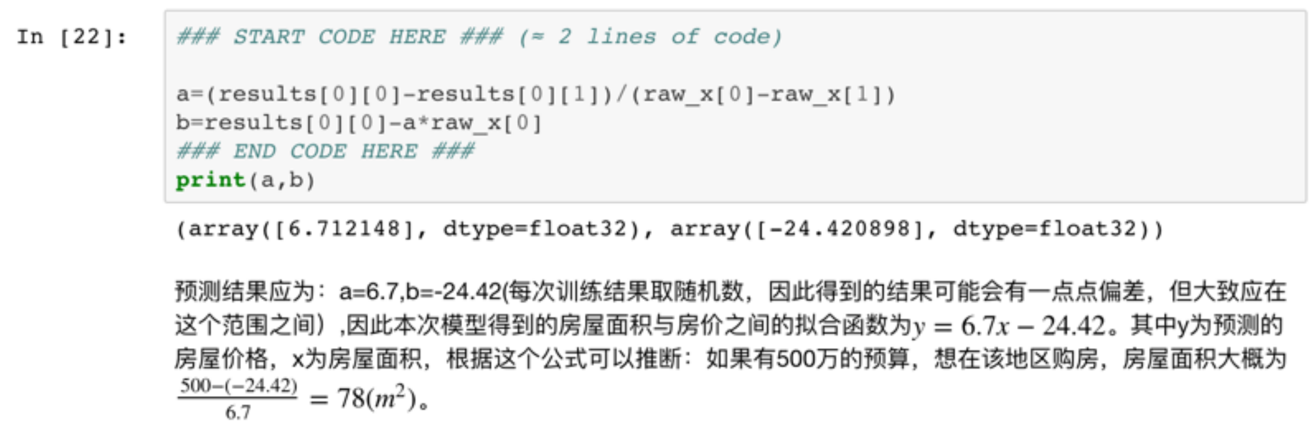
此处应得到一组预测结果：

('the area is:', array([[####],

```
       [####]], dtype=float32))
```

('infer results: ', array([[####],

```
       [####]], dtype=float32))
```



In [22]:
```
### START CODE HERE ### (≈ 2 lines of code)

a=(results[0][0]-results[0][1])/(raw_x[0]-raw_x[1])
b=results[0][0]-a*raw_x[0]
### END CODE HERE ###
print(a,b)
```

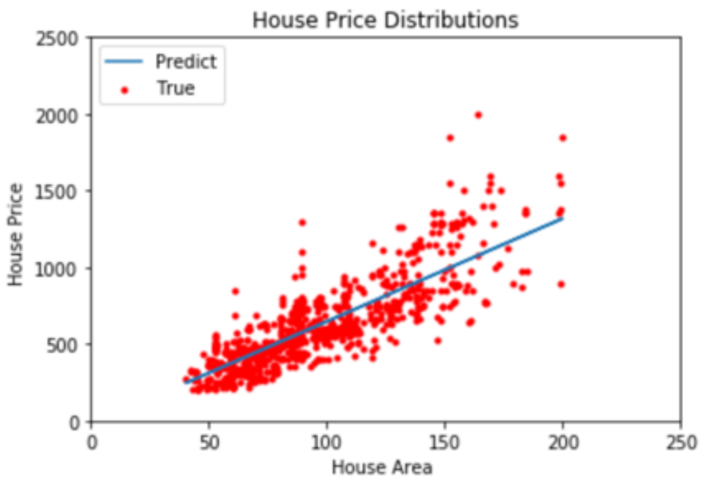
```
(array([6.712148], dtype=float32), array([-24.420898], dtype=float32))
```

预测结果应为：a=6.7,b=-24.42(每次训练结果取随机数，因此得到的结果可能会有一点点偏差，但大致应在这个范围之间）,因此本次模型得到的房屋面积与房价之间的拟合函数为$y = 6.7x - 24.42$。其中y为预测的房屋价格，x为房屋面积，根据这个公式可以推断：如果有500万的预算，想在该地区购房，房屋面积大概为$\frac{500-(-24.42)}{6.7} = 78(m^2)$。

```
In [23]:  import numpy as np
          import matplotlib.pyplot as plt

          def plot_data(data):
              x = data[:,0]
              y = data[:,1]
              y_predict = x*a + b
              plt.scatter(x,y,marker='.',c='r',label='True')
              plt.title('House Price Distributions')
              plt.xlabel('House Area ')
              plt.ylabel('House Price ')
              plt.xlim(0,250)
              plt.ylim(0,2500)
              predict = plt.plot(x,y_predict,label='Predict')
              plt.legend(loc='upper left')
              plt.savefig('result1.png')
              plt.show()

          ### START CODE HERE ### (≈ 1 lines of code)
              #读取数据#
          data = np.loadtxt('./datasets/data.txt',delimiter = ',')
          ### END CODE HERE ###
          plot_data(data)
```



> Looks as if I am lucky enough to miss the problems of PaddlePaddle mentioned by others?
>
> But if I can choose, I prefer Keras.
>
> LOL