第三次上机作业

18029100040

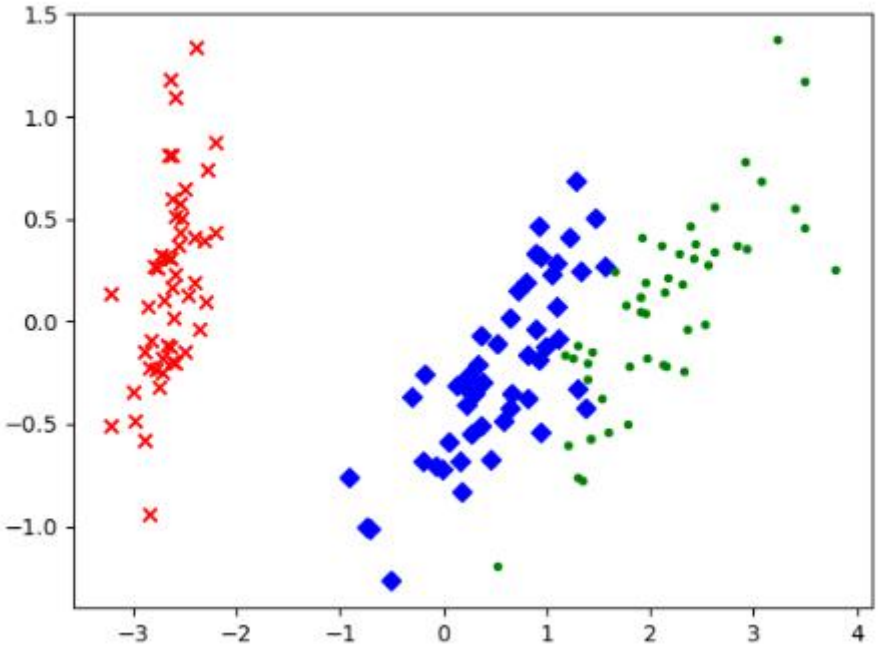
吴程锴

一、作业 26：PCA 实现高维数据可视化：

1.1 代码

```
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.datasets import load_iris
data = load_iris()
y = data.target
x = data.data
pca = PCA(n_components = 2)
reduced_x = pca.fit_transform(x)
red_x,red_y = [],[]
blue_x,blue_y = [],[]
green_x,green_y = [],[]
for i in range(len(reduced_x)):
    if y[i] == 0:
        red_x.append(reduced_x[i][0])
        red_y.append(reduced_x[i][1])
    elif y[i] == 1:
        blue_x.append(reduced_x[i][0])
        blue_y.append(reduced_x[i][1])
    else:
        green_x.append(reduced_x[i][0])
        green_y.append(reduced_x[i][1])
plt.scatter(red_x,red_y, c = 'r',marker = 'x')
plt.scatter(blue_x,blue_y, c = 'b',marker = 'D')
plt.scatter(green_x,green_y, c = 'g',marker = '.')
    plt.show()
```

1.2 结果



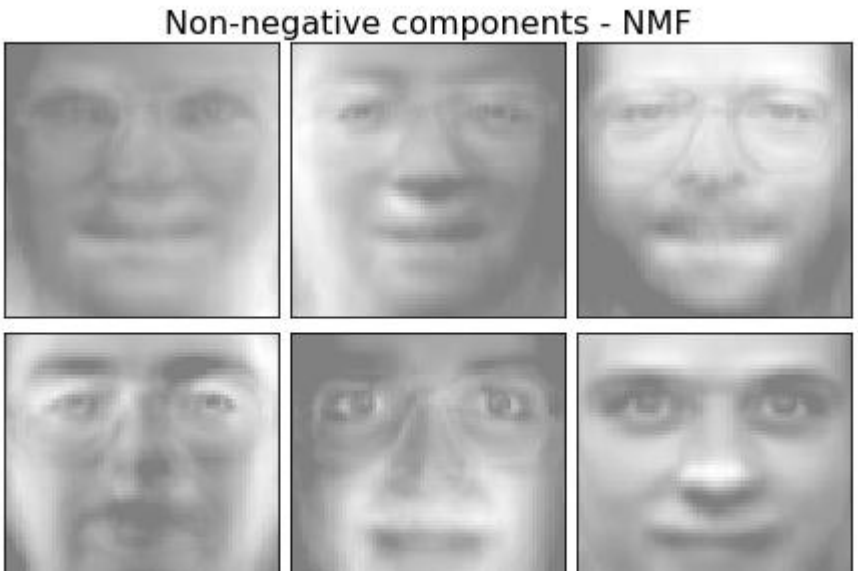
二、作业 27：降维之 NMF

2.1 代码

```
    import matplotlib.pyplot as plt
from sklearn import decomposition
from sklearn.datasets import fetch_olivetti_faces
from numpy.random import RandomState


n_row, n_col = 2, 3

n_components = n_row = n_col

image_shape = (64, 64)
dataset = fetch_olivetti_faces(shuffle=True, random_state=RandomState(0))
faces = dataset.data
def plot_gallery(title, images, n_col=n_col, n_row=n_row):
    plt.figure(figsize=(2. * n_col, 2.26 * n_row))
    plt.suptitle(title, size=16)
    for i, comp in enumerate(images):
```

```
        plt.subplot(n_row, n_col, i + 1)
        vmax = max(comp.max(), -comp.min())
        plt.imshow(comp.reshape(image_shape), cmap=plt.cm.gray, interpolation='nearest', vmin=-vmax,
                   vmax=vmax)
        plt.xticks(())
        plt.yticks(())
    plt.subplots_adjust(0.01, 0.05, 0.99, 0.93, 0.04, 0.)


# 创建特征提取的对象 NMF，使用 PCA 作为对
estimators = [('Eigenfaces - PCA using randomized SVD', decomposition.PCA(n_components=6, whiten=True)),
              ('Non-negative components - NMF', decomposition.NMF(n_components=6, init='nndsvda', tol=5e-3))]
# NMF 和 PCA 实例，将它们放在一个列表中

# 降维后数据点的可视化
for name, estimators in estimators:
    estimators.fit(faces)
    components_ = estimators.components_
    plot_gallery(name, components_[:n_components])
    plt.show()  # 可视化
```

2.2 结果

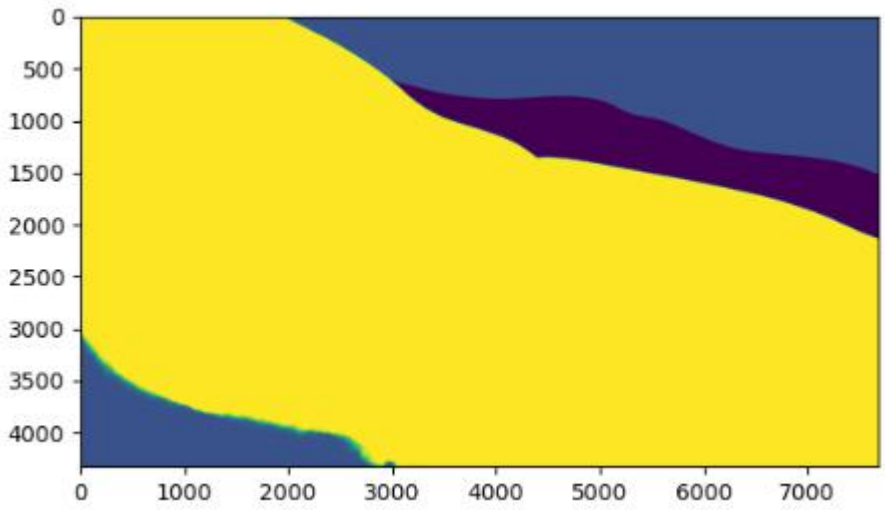


三、作业 28：图像分割

3.1 代码

```
import numpy as np
from PIL import Image  #加载 PIL 包，用于加载创建图片
from sklearn.cluster import KMeans  #加载 Kmeans 算法
import matplotlib.pyplot as plt  #绘制图像

def loadData(filePath):
    f = open(filePath, 'rb') #以二进制形式打开文件
    data = []
    img = Image.open(f)  #以列表的形式返回图片像素值
    m, n = img.size   #获取图片的大小
    for i in range(m):  #将每个像素点的 RGB 颜色处理到 0-1
        for j in range(n):
            x,y,z = img.getpixel((i,j))
            data.append([x/256.0, y/256.0, z/256.0]) #范围内并存入 data
    f.close()
    return np.mat(data), m, n #以矩阵的形式返回 data，以及图片大小

imgData,row,col = loadData('bull.jpg')
#加载 Kmeans 聚类算法
km = KMeans(n_clusters= 3) #其中 n clusters 属性指定了聚类中心的个数为 3

#聚类获取每个像素所属的类别
label = km.fit_predict(imgData)
label = label.reshape([row, col])
#创建一张新的灰度图保存聚类后的结果
pic_new = Image.new('L', (row, col))

#根据所属类别向图片中添加灰度值
# 最终利用聚类中心点的 RGB 值替换原图中每一个像素点的值，便得到了最终的分割后的图片
for i in range(row):
    for j in range(col):
        pic_new.putpixel((i, j), int(256 / (label[i][j] + 1)))

#以 JPEG 格式保存图片
pic_new.save("bull_r.jpg","JPEG")
plt.imshow(pic_new)
plt.show()
```

3.2 结果



四、作业 29：人体运动状态信息评级

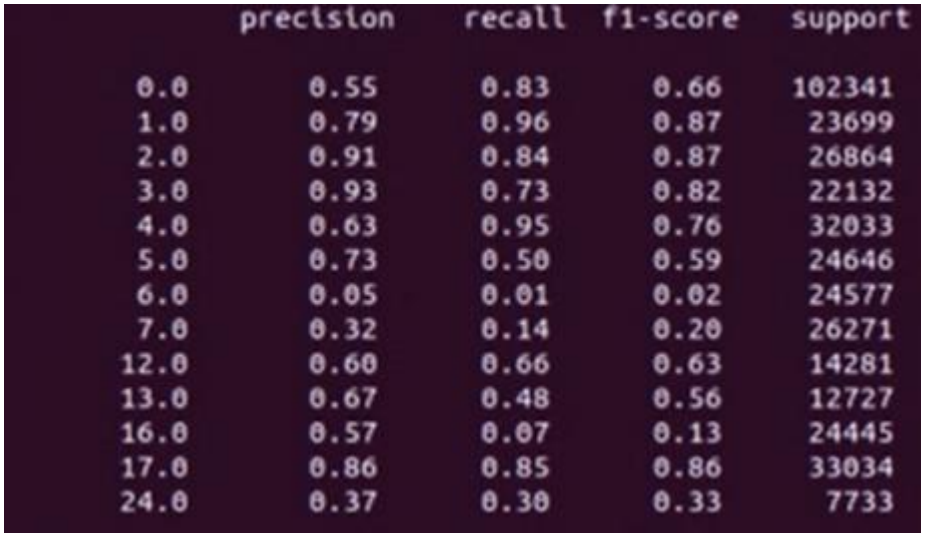
4.1 代码

```
import pandas as pd
import numpy as np

from sklearn.preprocessing import Imputer
from sklearn.cross_validation import train_test_split
from sklearn.metrics import classification_report

from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
```

```
11.
12. def load_datasets(feature_paths, label_paths):
13.     feature = np.ndarray(shape=(0,41))
14.     label = np.ndarray(shape=(0,1))
15.     for file in feature_paths:
16.         df = pd.read_table(file, delimiter=',', na_values='?', header=None)
17.         imp = Imputer(missing_values='NaN', strategy='mean', axis=0)
18.         imp.fit(df)
19.         df = imp.transform(df)
20.         feature = np.concatenate((feature, df))
21.
22.     for file in label_paths:
23.         df = pd.read_table(file, header=None)
24.         label = np.concatenate((label, df))
25.
26.     label = np.ravel(label)
27.     return feature, label
28.
29. if __name__ == '__main__':
30.     featurePaths = ['A/A.feature','B/B.feature','C/C.feature','D/D.feature','E/E.fea
    ture']
31.     labelPaths = ['A/A.label','B/B.label','C/C.label','D/D.label','E/E.label']
32.     x_train,y_train = load_datasets(featurePaths[:4],labelPaths[:4])
33.     x_test,y_test = load_datasets(featurePaths[4:],labelPaths[4:])
34.     x_train, x_, y_train, y_ = train_test_split(x_train, y_train, test_size = 0.0)
35.
36.     print('Start training knn')
37.     knn = KNeighborsClassifier().fit(x_train, y_train)
38.     print('Training done')
39.     answer_knn = knn.predict(x_test)
40.     print('Prediction done')
41.
42.     print('Start training DT')
43.     dt = DecisionTreeClassifier().fit(x_train, y_train)
44.     print('Training done')
45.     answer_dt = dt.predict(x_test)
46.     print('Prediction done')
47.
48.     print('Start training Bayes')
49.     gnb = GaussianNB().fit(x_train, y_train)
50.     print('Training done')
51.     answer_gnb = gnb.predict(x_test)
52.     print('Prediction done')
53.
54.     print('\n\nThe classification report for knn:')
```

```
    print(classification_report(y_test, answer_knn))
    print('\n\nThe classification report for DT:')
    print(classification_report(y_test, answer_dt))
    print('\n\nThe classification report for Bayes:')
        print(classification_report(y_test, answer_gnb))
```

4.2 结果



	precision	recall	f1-score	support
0.0	0.55	0.83	0.66	102341
1.0	0.79	0.96	0.87	23699
2.0	0.91	0.84	0.87	26864
3.0	0.93	0.73	0.82	22132
4.0	0.63	0.95	0.76	32033
5.0	0.73	0.50	0.59	24646
6.0	0.05	0.01	0.02	24577
7.0	0.32	0.14	0.20	26271
12.0	0.60	0.66	0.63	14281
13.0	0.67	0.48	0.56	12727
16.0	0.57	0.07	0.13	24445
17.0	0.86	0.85	0.86	33034
24.0	0.37	0.30	0.33	7733

五、作业 30：上证指数涨跌预测实例

5.1 代码

```
import pandas as pd
# ：支持高级大量的维度数组与矩阵运算，此外也针对数组运算提供大量的数学函数库
import numpy as np
# sklearn 下 svm：SVM 算法
from sklearn import svm
# sklearn 下 cross_validation：交叉验证
from sklearn import model_selection

# parse_dates=第 0 列解析为日期， index_col= 用作行索引的列编号）
data = pd.read_csv(r'000777.csv', encoding='gbk', parse_dates=[0],
                   index_col=0)
# DataFrame.sort_index(axis=0 (按 0 列排), ascending=True（升序），
# inplace=False（排序后是否覆盖原数据））data 按照时间升序排列
data.sort_index(0, ascending=True, inplace=True)

# 选取 5 列数据作为特征：收盘价 最高价 最低价 开盘价 成交量
# dayfeature：选取 150 天的数据
# featurenum：选取的 5 个特征*天数
```

```
# x：记录 150 天的 5 个特征值 y：记录涨或者跌
dayfeature = 150
featurenum = 5 * dayfeature
# data.shape[0]-dayfeature 意思是因为我们要用 150 天数据做训练，
# 对于条目为 200 条的数据，只有 50 条数据是有前 150 天的数据来训练的，
# 所以测试集的大小就是 200-150， 对于每一条数据，他的特征是前 150 天的所有特征数据，
# 即 150*5， +1 是将当天的开盘价引入作为一条特征数据
x = np.zeros((data.shape[0] - dayfeature, featurenum + 1))
y = np.zeros((data.shape[0] - dayfeature))

for i in range(0, data.shape[0] - dayfeature):
    # /将数据中的“收盘价”“最高价”“开盘价”“成交量”存入 x 数组中
    # u:unicode 编码 reshape:转换成 1 行，featurenum 列
    x[i, 0:featurenum] = np.array(data[i:i + dayfeature] \
                                      [[u'收盘价', u'最高价',
                                        u'最低价', u'开盘价', u'成交量
'']]).reshape((1, featurenum))
    x[i, featurenum] = data.ix[i + dayfeature][u'开盘价']
    # 最后一列记录当日的开盘价              ix :索引
for i in range(0, data.shape[0] - dayfeature):
    if data.ix[i + dayfeature][u'收盘价'] >= data.ix[i + dayfeature][u'开盘价']:
        y[i] = 1
    else:
        y[i] = 0
        # 如果当天收盘价高于开盘价，y[i]=1 代表涨，0 代表跌
# 创建 SVM 并进行交叉验证
clf = svm.SVC(kernel='rbf')
# 调用 svm 函数,并设置 kernel 参数，默认是 rbf，其它：‘linear’‘poly’‘sigmoid’
result = []
for i in range(5):
    # x 和 y 的验证集和测试集，切分 80 - 20 % 的测试集
    x_train, x_test, y_train, y_test = \
        model_selection.train_test_split(x, y, test_size=0.2)
    # 训练数据进行训练
    clf.fit(x_train, y_train)
    # 将预测数据和测试集的验证数据比对
    result.append(np.mean(y_test == clf.predict(x_test)))
print("svm classifier accuacy:")
    print(result)
```

5.2 结果

```
svm classifier accuacy:
[0.5635179153094463,0.5754614549402823, 0.5266015200868621, 0.5450597176981542, 0.5407166123778502]
```

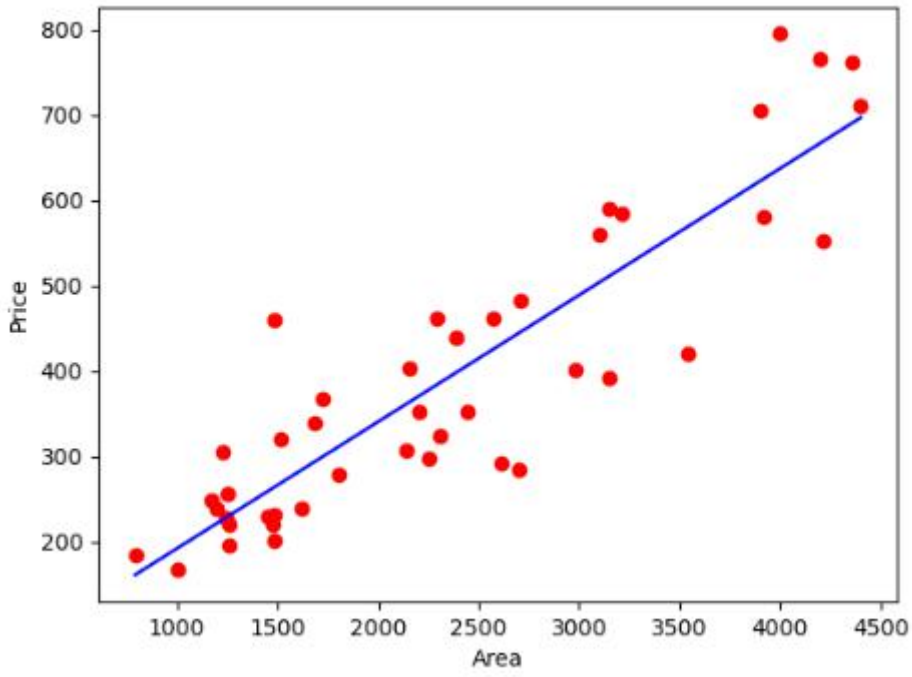
六、作业 31：线性回归+房价与房屋尺寸关系的线性拟合

6.1 代码

```
#表示 matplotlib 的 pyplot 子库，它提供了和 matlab 类似的绘图 API
import  matplotlib.pyplot as plt
#表示可以调用 sklearn 中的 linear_model 模块进行线性回归。
from sklearn import  linear_model
import numpy as np
#建立 datasets_X 和 datasets_Y 用来存储数据中的房屋尺寸和房屋成交价格。
datasets_X =[]
datasets_Y =[]
fr =open('prices.txt','r')
#一次读取整个文件。
lines =fr.readlines()
#逐行进行操作，循环遍历所有数据
for line in lines:
    #去除数据文件中的逗号
    items =line.strip().split(',')
    #将读取的数据转换为 int 型，并分别写入 datasets_X 和 datasets_Y。
    datasets_X.append(int(items[0]))
    datasets_Y.append(int(items[1]))
#求得 datasets_X 的长度，即为数据的总数。
length =len(datasets_X)
#将 datasets_X 转化为数组， 并变为二维，以符合线性回 归拟合函数输入参数要求
datasets_X= np.array(datasets_X).reshape([length,1])
#将 datasets_Y 转化为数组
datasets_Y=np.array(datasets_Y)

minX =min(datasets_X)
maxX =max(datasets_X)
#以数据 datasets_X 的最大值和最小值为范围，建立等差数列，方便后续画图。
X=np.arange(minX,maxX).reshape([-1,1])

linear =linear_model.LinearRegression()
linear.fit(datasets_X,datasets_Y)#调用线性回归模块，建立回归方程，拟合数据
#查看回归方程系数
print('Cofficients:',linear.coef_)
#查看回归方程截距
print('intercept',linear.intercept_)

#3.可视化处理
#scatter 函数用于绘制数据 点，这里表示用红色绘制数据点；
plt.scatter(datasets_X,datasets_Y,color='red')
```

```
#plot 函数用来绘制直线，这 里表示用蓝色绘制回归线；
#xlabel 和 ylabel 用来指定横纵坐标的名称
plt.plot(X,linear.predict(X),color='blue')
plt.xlabel('Area')
plt.ylabel('Price')
    plt.show()
```

6.2 结果



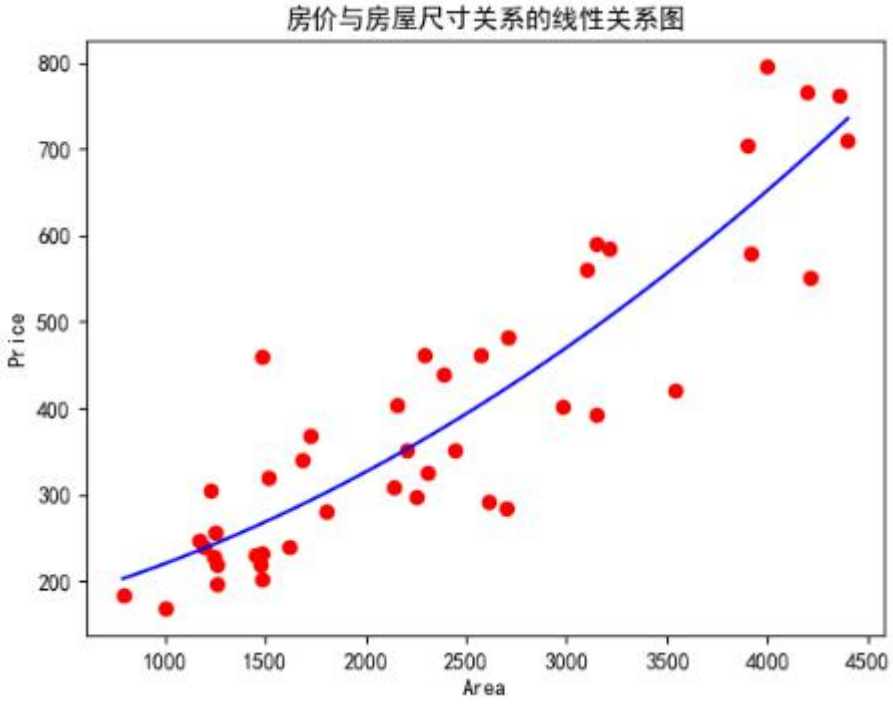
七、作业 32：多项式回归+房价与房屋尺寸的非线性拟合

7.1 代码

```
    #房价与房屋尺寸关系的非线性拟合（多项式回归）
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model
from sklearn.preprocessing import PolynomialFeatures as PF

plt.rcParams['font.sans-serif']='SimHei'
plt.rcParams['axes.unicode_minus']=False
data_x=[] #设房屋的尺寸（面积）为 data_x
data_y=[] #房价为 data_y
```

```
f=open('prices.txt','r')

lines=f.readlines()
for line in lines:
    items=line.strip().split(',')
    data_x.append(int(items[0]))
    data_y.append(int(items[1]))
length=len(data_x)
data_x=np.array(data_x).reshape([length,1])
data_y=np.array(data_y)
minX=min(data_x)
maxX=max(data_x)
x=np.arange(minX,maxX).reshape([-1,1])

poly_reg=PF(degree=2) #degree=2 表示建立 data_x 的二 次多项式特征 X_poly
x_poly=poly_reg.fit_transform(data_x)
linear=linear_model.LinearRegression()
linear.fit(x_poly,data_y)  #拟合 x，y

plt.scatter(data_x,data_y,color='red')
plt.plot(x,linear.predict(poly_reg.fit_transform(x)),color='blue')
plt.xlabel('Area')  #x 轴标签
plt.ylabel('Price')
plt.title('房价与房屋尺寸关系的线性关系图')
    plt.show()
```

7.2 结果


房价与房屋尺寸关系的线性关系图
Price
Area
800
700
600
500
400
300
200
1000
1500
2000
2500
3000
3500
4000
4500