第四次上机作业

**18029100040**

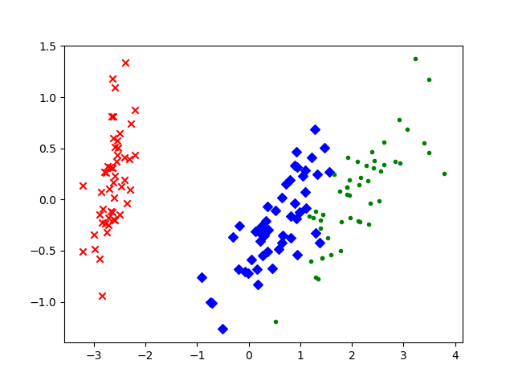
**吴程锴**

# 作业 26：PCA 实现高维数据可视化：

## 代码

1. **import** matplotlib.pyplot as plt
2. **from** sklearn.decomposition **import** PCA
3. **from** sklearn.datasets **import** load\_iris
4. data = load\_iris()
5. y = data.target
6. x = data.data
7. pca = PCA(n\_components = 2)
8. reduced\_x = pca.fit\_transform(x)
9. red\_x,red\_y = [],[]
10. blue\_x,blue\_y = [],[]
11. green\_x,green\_y = [],[]
12. **for** i **in** range(len(reduced\_x)):
13. **if** y[i] == 0:
14. red\_x.append(reduced\_x[i][0])
15. red\_y.append(reduced\_x[i][1])
16. **elif** y[i] == 1:
17. blue\_x.append(reduced\_x[i][0])
18. blue\_y.append(reduced\_x[i][1])
19. **else**:
20. green\_x.append(reduced\_x[i][0])
21. green\_y.append(reduced\_x[i][1])
22. plt.scatter(red\_x,red\_y, c = 'r',marker = 'x')
23. plt.scatter(blue\_x,blue\_y, c = 'b',marker = 'D')
24. plt.scatter(green\_x,green\_y, c = 'g',marker = '.')
25. plt.show()

## 结果



# 作业 27：降维之 NMF

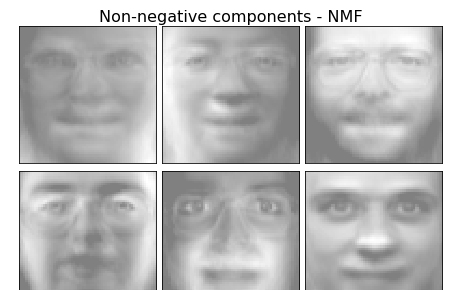
## 代码

1. **import** matplotlib.pyplot as plt
2. **from** sklearn **import** decomposition
3. **from** sklearn.datasets **import** fetch\_olivetti\_faces
4. **from** numpy.random **import** RandomState


8. n\_row, n\_col = 2, 3
10. n\_components = n\_row = n\_col
12. image\_shape = (64, 64)
13. dataset = fetch\_olivetti\_faces(shuffle=True, random\_state=RandomState(0))
14. faces = dataset.data
15. **def** plot\_gallery(title, images, n\_col=n\_col, n\_row=n\_row):
16. plt.figure(figsize=(2. \* n\_col, 2.26 \* n\_row))
17. plt.suptitle(title, size=16)
18. **for** i, comp **in** enumerate(images):
19. plt.subplot(n\_row, n\_col, i + 1)
20. vmax = max(comp.max(), -comp.min())
21. plt.imshow(comp.reshape(image\_shape), cmap=plt.cm.gray, interpolation='nearest', vmin=-vmax,
22. vmax=vmax)
23. plt.xticks(())
24. plt.yticks(())
25. plt.subplots\_adjust(0.01, 0.05, 0.99, 0.93, 0.04, 0.)

28. # 创建特征提取的对象NMF，使用PCA作为对
29. estimators = [('Eigenfaces - PCA using randomized SVD', decomposition.PCA(n\_components=6, whiten=True)),
30. ('Non-negative components - NMF', decomposition.NMF(n\_components=6, init='nndsvda', tol=5e-3))]
31. # NMF和PCA实例，将它们放在一个列表中
33. # 降维后数据点的可视化
34. **for** name, estimators **in** estimators:
35. estimators.fit(faces)
36. components\_ = estimators.components\_
37. plot\_gallery(name, components\_[:n\_components])
38. plt.show()  # 可视化

## 结果



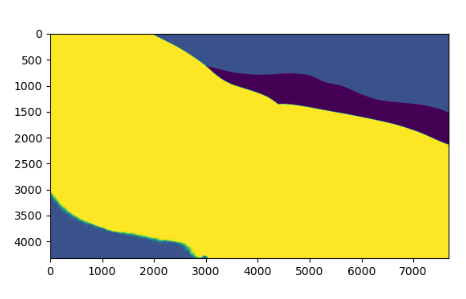
# 作业 28：图像分割

## 代码

1. **import** numpy as np
2. **from** PIL **import** Image  #加载PIL包，用于加载创建图片
3. **from** sklearn.cluster **import** KMeans  #加载Kmeans算法
4. **import** matplotlib.pyplot as plt  #绘制图像
6. **def** loadData(filePath):
7. f = open(filePath, 'rb') #以二进制形式打开文件
8. data = []
9. img = Image.open(f)  #以列表的形式返回图片像素值
10. m, n = img.size   #获取图片的大小
11. **for** i **in** range(m):  #将每个像素点的RGB颜色处理到0-1
12. **for** j **in** range(n):
13. x,y,z = img.getpixel((i,j))
14. data.append([x/256.0, y/256.0, z/256.0]) #范围内并存入data
15. f.close()
16. **return** np.mat(data), m, n #以矩阵的形式返回data，以及图片大小
18. imgData,row,col = loadData('bull.jpg')
19. #加载Kmeans聚类算法
20. km = KMeans(n\_clusters= 3) #其中n clusters属性指定了聚类中心的个数为3
22. #聚类获取每个像素所属的类别
23. label = km.fit\_predict(imgData)
24. label = label.reshape([row, col])
25. #创建一张新的灰度图保存聚类后的结果
26. pic\_new = Image.new('L', (row, col))
28. #根据所属类别向图片中添加灰度值
29. # 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
30. **for** i **in** range(row):
31. **for** j **in** range(col):
32. pic\_new.putpixel((i, j), int(256 / (label[i][j] + 1)))
34. #以JPEG格式保存图片
35. pic\_new.save("bull\_r.jpg","JPEG")
36. plt.imshow(pic\_new)
37. plt.show()

## 结果



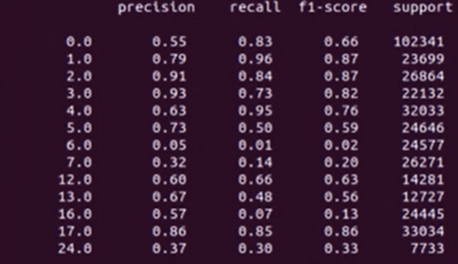


# 作业 29：人体运动状态信息评级

## 代码

1. **import** pandas as pd
2. **import** numpy as np
4. **from** sklearn.preprocessing **import** Imputer
5. **from** sklearn.cross\_validation **import** train\_test\_split
6. **from** sklearn.metrics **import** classification\_report
8. **from** sklearn.neighbors **import** KNeighborsClassifier
9. **from** sklearn.tree **import** DecisionTreeClassifier
10. **from** sklearn.naive\_bayes **import** GaussianNB
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))
22. **for** file **in** label\_paths:
23. df = pd.read\_table(file, header=None)
24. label = np.concatenate((label, df))
26. label = np.ravel(label)
27. **return** feature, label
29. **if** \_\_name\_\_ == '\_\_main\_\_':
30. featurePaths = ['A/A.feature','B/B.feature','C/C.feature','D/D.feature','E/E.feature']
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)
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')
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')
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')
54. **print**('\n\nThe classification report for knn:')
55. **print**(classification\_report(y\_test, answer\_knn))
56. **print**('\n\nThe classification report for DT:')
57. **print**(classification\_report(y\_test, answer\_dt))
58. **print**('\n\nThe classification report for Bayes:')
59. **print**(classification\_report(y\_test, answer\_gnb))

## 结果



# 作业 30：上证指数涨跌预测实例

## 代码

1. **import** pandas as pd
2. # ：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
3. **import** numpy as np
4. # sklearn下svm：SVM算法
5. **from** sklearn **import** svm
6. # sklearn下cross\_validation：交叉验证
7. **from** sklearn **import** model\_selection
9. # parse\_dates=第0列解析为日期， index\_col= 用作行索引的列编号）
10. data = pd.read\_csv(r'000777.csv', encoding='gbk', parse\_dates=[0],
11. index\_col=0)
12. # DataFrame.sort\_index(axis=0 (按0列排), ascending=True（升序）,
13. # inplace=False（排序后是否覆盖原数据））data 按照时间升序排列
14. data.sort\_index(0, ascending=True, inplace=True)
16. # 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
17. # dayfeature：选取150天的数据
18. # featurenum：选取的5个特征\*天数
19. # x：记录150天的5个特征值 y：记录涨或者跌
20. dayfeature = 150
21. featurenum = 5 \* dayfeature
22. # data.shape[0]-dayfeature意思是因为我们要用150天数据做训练，
23. # 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，
24. # 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，
25. # 即150\*5， +1是将当天的开盘价引入作为一条特征数据
26. x = np.zeros((data.shape[0] - dayfeature, featurenum + 1))
27. y = np.zeros((data.shape[0] - dayfeature))
29. **for** i **in** range(0, data.shape[0] - dayfeature):
30. # /将数据中的“收盘价”“最高价”“开盘价”“成交量”存入x数组中
31. # u:unicode编码 reshape:转换成1行，featurenum列
32. x[i, 0:featurenum] = np.array(data[i:i + dayfeature] \
33. [[u'收盘价', u'最高价',
34. u'最低价', u'开盘价', u'成交量']]).reshape((1, featurenum))
35. x[i, featurenum] = data.ix[i + dayfeature][u'开盘价']
36. # 最后一列记录当日的开盘价              ix :索引
37. **for** i **in** range(0, data.shape[0] - dayfeature):
38. **if** data.ix[i + dayfeature][u'收盘价'] >= data.ix[i + dayfeature][u'开盘价']:
39. y[i] = 1
40. **else**:
41. y[i] = 0
42. # 如果当天收盘价高于开盘价，y[i]=1代表涨，0代表跌
43. # 创建SVM并进行交叉验证
44. clf = svm.SVC(kernel='rbf')
45. # 调用svm函数,并设置kernel参数，默认是rbf，其它：‘linear’‘poly’‘sigmoid’
46. result = []
47. **for** i **in** range(5):
48. # x和y的验证集和测试集，切分80 - 20 % 的测试集
49. x\_train, x\_test, y\_train, y\_test = \
50. model\_selection.train\_test\_split(x, y, test\_size=0.2)
51. # 训练数据进行训练
52. clf.fit(x\_train, y\_train)
53. # 将预测数据和测试集的验证数据比对
54. result.append(np.mean(y\_test == clf.predict(x\_test)))
55. **print**("svm classifier accuacy:")
56. **print**(result)

## 结果

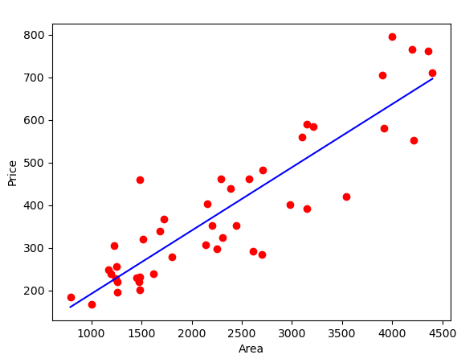


# 作业 31：线性回归+房价与房屋尺寸关系的线性拟合

## 代码

1. #表示matplotlib的pyplot子库，它提供了和matlab类似的绘图API
2. **import**  matplotlib.pyplot as plt
3. #表示可以调用sklearn中的 linear\_model模块进行线性回归。
4. **from** sklearn **import**  linear\_model
5. **import** numpy as np
6. #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。
7. datasets\_X =[]
8. datasets\_Y =[]
9. fr =open('prices.txt','r')
10. #一次读取整个文件。
11. lines =fr.readlines()
12. #逐行进行操作，循环遍历所有数据
13. **for** line **in** lines:
14. #去除数据文件中的逗号
15. items =line.strip().split(',')
16. #将读取的数据转换为int型，并分别写入datasets\_X和datasets\_Y。
17. datasets\_X.append(int(items[0]))
18. datasets\_Y.append(int(items[1]))
19. #求得datasets\_X的长度，即为数据的总数。
20. length =len(datasets\_X)
21. #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
22. datasets\_X= np.array(datasets\_X).reshape([length,1])
23. #将datasets\_Y转化为数组
24. datasets\_Y=np.array(datasets\_Y)
26. minX =min(datasets\_X)
27. maxX =max(datasets\_X)
28. #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。
29. X=np.arange(minX,maxX).reshape([-1,1])
31. linear =linear\_model.LinearRegression()
32. linear.fit(datasets\_X,datasets\_Y)#调用线性回归模块，建立回归方程，拟合数据
33. #查看回归方程系数
34. **print**('Cofficients:',linear.coef\_)
35. #查看回归方程截距
36. **print**('intercept',linear.intercept\_)
38. #3.可视化处理
39. #scatter函数用于绘制数据 点，这里表示用红色绘制数据点；
40. plt.scatter(datasets\_X,datasets\_Y,color='red')
41. #plot函数用来绘制直线，这 里表示用蓝色绘制回归线；
42. #xlabel和ylabel用来指定横纵坐标的名称
43. plt.plot(X,linear.predict(X),color='blue')
44. plt.xlabel('Area')
45. plt.ylabel('Price')
46. plt.show()

## 结果



# 作业 32：多项式回归+房价与房屋尺寸的非线性拟合

## 代码

1. #房价与房屋尺寸关系的非线性拟合（多项式回归）
2. **import** numpy as np
3. **import** matplotlib.pyplot as plt
4. **from** sklearn **import** linear\_model
5. **from** sklearn.preprocessing **import** PolynomialFeatures as PF
7. plt.rcParams['font.sans-serif']='SimHei'
8. plt.rcParams['axes.unicode\_minus']=False
9. data\_x=[] #设房屋的尺寸（面积）为data\_x
10. data\_y=[] #房价为data\_y
11. f=open('prices.txt','r')
13. lines=f.readlines()
14. **for** line **in** lines:
15. items=line.strip().split(',')
16. data\_x.append(int(items[0]))
17. data\_y.append(int(items[1]))
18. length=len(data\_x)
19. data\_x=np.array(data\_x).reshape([length,1])
20. data\_y=np.array(data\_y)
21. minX=min(data\_x)
22. maxX=max(data\_x)
23. x=np.arange(minX,maxX).reshape([-1,1])
25. poly\_reg=PF(degree=2) #degree=2表示建立data\_x的二 次多项式特征X\_poly
26. x\_poly=poly\_reg.fit\_transform(data\_x)
27. linear=linear\_model.LinearRegression()
28. linear.fit(x\_poly,data\_y)  #拟合x，y
30. plt.scatter(data\_x,data\_y,color='red')
31. plt.plot(x,linear.predict(poly\_reg.fit\_transform(x)),color='blue')
32. plt.xlabel('Area')  #x轴标签
33. plt.ylabel('Price')
34. plt.title('房价与房屋尺寸关系的线性关系图')
35. plt.show()

## 结果

