**特征回归**

**import** numpy **as** np  
**import** xlrd  
**from** sklearn.linear\_model **import** LogisticRegression  
**from** sklearn.model\_selection **import** train\_test\_split  
**from** sklearn.metrics **import** classification\_report  
  
  
**def** open\_excel(file):  
 *"""  
 打开excel文件获取数据* **:param** *file: 文件所在的位置* **:return***: 文件数据  
 """* **try**:  
 data = xlrd.open\_workbook(file)  
 **return** data  
 **except** Exception **as** e:  
 print(str(e))  
  
  
**def** split\_feature(row):  
 *"""  
 将该行特征处理后放入列表中* **:param** *row:一行特征数据* **:return***: 返回数据列表  
 """* app = []  
 **for** i **in** range(16):  
 app = app + [row[i]]  
 **return** app  
  
  
**def** loadDataSet(path, training\_sample, colnameindex=0, by\_name=**u'sheet1'**):  
 *"""  
 加载数据* **:param** *path: 数据文件存放路径* **:param** *training\_sample: 数据文件名* **:param** *colnameindex: 文件列名下标* **:param** *by\_name: 表名* **:return***: 数据集和类别标签  
 """* dataMat = [] *# 定义数据列表* labelMat = [] *# 定义标签列表* filename = path + training\_sample *# 形成特征数据的完整路径* data = open\_excel(filename) *# 打开文件获取数据* table = data.sheet\_by\_name(by\_name) *# 获得数据表* nrows = table.nrows *# 得到表数据总行数* colnames = table.row\_values(colnameindex) *# 某一行数据 ['user\_id', 'age\_range', 'gender', 'merchant\_id','label']* **for** rownum **in** range(1, nrows): *# 也就是从Excel第二行开始，第一行表头不算* row = table.row\_values(rownum) *# 取一行数据* **'''  
 判断2,3,6列数据是否为空，若为空则丢弃该行数据  
 '''  
 if** row[1] == **'' or** row[2] == **'' or** row[5] == **''**:  
 **continue  
 if** row:  
 app = split\_feature(row) *# 将特征值转化为列表* dataMat.append(app)  
 labelMat.append(float(row[16])) *# 获取类别标签* **return** dataMat, labelMat  
  
  
**def** show\_accuracy(a, b, tip):  
 *"""  
 计算准确率* **:param** *a: 真实类别* **:param** *b: 预测标签* **:param** *tip: 描述* **:return***: 准确率  
 """* acc = a.ravel() == b.ravel()  
 print(**"%s Accuracy:%.3f"** % (tip, np.mean(acc)))  
  
  
**def** main():  
 *"""  
 主函数* **:return***: null  
 """* path = **"E:\\"** training\_sample = **'featuredata.xls'** *# 特征数据文件* trainingSet, trainingLabels = loadDataSet(path, training\_sample) *# 取特征数据和标签数据* x = np.array(trainingSet) *# 将数据部分列表（list）格式转化为数组(array)格式* y = np.array(trainingLabels) *# 将标签部分的列表（list）格式转化为数组格式（array）* **'''  
 将数据分为训练数据和测试数据两部分  
 x\_train 训练数据  
 x\_test 测试数据  
 y\_train 训练数据标签  
 y\_test 测试数据标签  
 '''** train\_data, test\_data, train\_label, test\_label = train\_test\_split(x, y, random\_state=1, test\_size=0.3)  
 *#选择模型* clf = LogisticRegression()  
 *#把数据交给模型训练* clf.fit(train\_data, train\_label)  
 hat\_test\_label = clf.predict(test\_data)  
 print(classification\_report(test\_label, hat\_test\_label))  
  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 **"""  
 程序入口  
 """** main()

**from** numpy **import** \*  
**import** matplotlib.pyplot **as** plt  
**def** loadDataSet(path,training\_sample):  
 *'''  
 从文件中读入训练样本的数据，同上面给出的示例数据  
 下面第20行代码中的1.0表示x0 = 1  
 @param filename 存放训练数据的文件路径  
 @return dataMat 存储训练数据的前两列  
 @return labelMat 存放给出的标准答案（0,1）  
 '''* dataMat = []; labelMat = []  
 filename=path+training\_sample  
 fr = open(filename)  
 **for** line **in** fr.readlines():  
 line = line.strip(**'\n'**)  
 lineArr = line.strip().split(**' '**) *#文件中数据的分隔符* dataMat.append([1.0, float(lineArr[0]), float(lineArr[1])]) *#前两列数据* labelMat.append(int(lineArr[2])) *# 标准答案* **return** dataMat,labelMat  
  
  
**def** sigmoid(inX):  
 *'''  
 定义激活函数  
 '''* **return** 1.0/(1+exp(-inX))  
  
  
**def** gradAscent(dataMatIn, classLabels):  
 *'''  
 梯度上升求最优参数a，学习率0.001，迭代次数1000次  
 @***:param** *dataMatIn 文件中数据的前两列  
 @***:param** *classLabels 标准答案  
 @***:return** *weights 训练后的参数 3 x 1  
 '''* dataMatrix = mat(dataMatIn) *#转化成矩阵* labelMat = mat(classLabels).transpose() *#矩阵转置* m,n = shape(dataMatrix)  
 alpha = 0.001 *#学习率* maxCycles = 500  
 weights = ones((n,1)) *#3行 1列* **for** k **in** range(maxCycles): *# 计算权重* h = sigmoid(dataMatrix\*weights) *# 模型预测值, n x 1* error = (labelMat - h) *# 真实值与预测值之间的误差, n x 1* temp = dataMatrix.transpose()\* error *# 交叉熵代价函数对所有参数求偏导数, 3 x 1* weights = weights + alpha \* temp *# 更新权重* **return** weights  
  
  
**def** plotBestFit(weights,dataMat,labelMat1,labelMat2):  
 *'''  
 分类效果展示，画图部分  
 @***:param** *weights 回归系数  
 @***:param** *path 数据文件路径  
 @***:return** *null  
 '''  
 # dataMat,labelMat1=loadDataSet(path,testing\_sample)  
 # dataMat1,labelMat1=loadDataSet(path,training\_sample)* dataArr = array(dataMat)  
 n = shape(dataArr)[0] *#取行数* xcord1 = []; ycord1 = []  
 xcord2 = []; ycord2 = []  
 xcord3 = []; ycord3 = []  
 xcord4 = []; ycord4 = []  
 **for** i **in** range(n): *#将训练前的数据分类存储* **if** int(labelMat1[i])== 1:  
 xcord1.append(dataArr[i,1]); ycord1.append(dataArr[i,2])  
 **else**:  
 xcord2.append(dataArr[i,1]); ycord2.append(dataArr[i,2])  
 **for** i **in** range(n): *#将训练后的数据分类存储* **if** int(labelMat2[i])== 1:  
 xcord3.append(dataArr[i,1]); ycord3.append(dataArr[i,2])  
 **else**:  
 xcord4.append(dataArr[i,1]); ycord4.append(dataArr[i,2])  
 fig = plt.figure(**"LogisticRegression"**) *#新建一个画图窗口* ax = fig.add\_subplot(111) *#添加一个子窗口* ax.set\_title(**'Original'**)  
 ax.scatter(xcord1, ycord1, s=30, c=**'red'**, marker=**'s'**)  
 ax.scatter(xcord2, ycord2, s=30, c=**'green'**)  
 x = arange(-3.0, 3.0, 0.1) *#定义x轴* y = (-weights[0] - weights[1]\*x) / weights[2] *# x2 = f(x1) 定义y轴 a0\*1+a1\*x+a2\*y* ax.plot(x, y) *#画一条直线* plt.xlabel(**'X1'**); plt.ylabel(**'X2'**)  
  
 plt.figure(**"logisticRegression"**)  
 plt.title(**'Forecast'**)  
 plt.scatter(xcord3, ycord3, s=30, c=**'red'**, marker=**'s'**)  
 plt.scatter(xcord4, ycord4, s=30, c=**'green'**)  
 plt.plot(x,y)  
 plt.xlabel(**'X1'**);plt.ylabel(**'X2'**)  
 plt.show()  
  
**def** getResult(dataArr,A):  
 h = sigmoid(mat(dataArr)\*A) *#预测结果h(a)的值* H = []  
 **for** i **in** range(shape(h)[0]):  
 **if** h[i,0] > 0.5:  
 H.append(1)  
 **else**:  
 H.append(0)  
 **return** H