****

**Python与数据分析**

**学 院： 管理工程学院**

**专 业： 电子信息**

**学生姓名： 张飞宇**

**学 号： 22020210036**

**指导教师： 刘经纬**

**2020年12月**

#### 目 录

[一、数据抓取 3](#_Toc60042440)

[二、数据预处理 3](#_Toc60042441)

[（一）数据清洗 3](#_Toc60042442)

[（二）数据集成 4](#_Toc60042443)

[（三）数据变换 4](#_Toc60042444)

[（四）数据规约 5](#_Toc60042445)

[三、K-means算法 5](#_Toc60042446)

[（一）概念 5](#_Toc60042447)

[（二）算法流程 6](#_Toc60042448)

[（三）代码 6](#_Toc60042449)

[四、随机森林算法 8](#_Toc60042450)

[（一）随机森林算法由来 8](#_Toc60042451)

[（二）随机森林的建立过程 8](#_Toc60042452)

[（三）决策树概念 9](#_Toc60042453)

[（四）使用决策树算法的主要步骤 9](#_Toc60042454)

[五、KNN（K最近邻居算法）算法 9](#_Toc60042455)

[（一）原理 9](#_Toc60042456)

[（二）主要实现过程 9](#_Toc60042457)

[六、Bagging（套袋法）、Boosting（提升法） 10](#_Toc60042458)

[（一）bagging的算法 10](#_Toc60042459)

[（二）boosting的算法 10](#_Toc60042460)

[七、SVM算法 10](#_Toc60042461)

[八、岭回归 11](#_Toc60042462)

[（一）原理 11](#_Toc60042463)

[（二）缺点 11](#_Toc60042464)

[（三）适用情况 11](#_Toc60042465)

[九、爬虫 11](#_Toc60042466)

SVM独特的特点：使用训练集的一个子集来表示决策边界，该子集称作支持向量。

# 八、岭回归

## （一）原理

岭回归是一种专用于共线性数据分析的有偏估计回归方法，实质上是一种改良的最小二乘估计法，通过放弃最小二乘法的无偏性，以损失部分信息、降低精度为代价，获得回归系数更为符合实际、更可靠的回归方法，对病态数据的耐受性远远强于最小二乘法。

## （二）缺点

通常岭回归方程的R平方值会稍低于普通回归分析，但回归系数的显著性往往明显高于普通回归，在存在共线性问题和病态数据偏多的研究中有较大的实用价值。

## （三）适用情况

主要适用于过拟合严重或各变量之间存在多重共线性的时候

1.岭回归可以解决特征数量比样本量多的问题

2.岭回归作为一种缩减算法可以判断哪些特征重要或者不重要，有点类似于降维的效果

3.缩减算法可以看作是对一个模型增加偏差的同时减少方差

# 九、爬虫

**from** selenium.webdriver **import** Chrome  
**from** selenium.webdriver.common.keys **import** Keys  
**import** pandas **as** pd  
**import** time  
**import** requests  
**import** json *#传递对象的语法***import** re *#z正则***import** xlrd *#读取Excel的扩展工具***import** csv  
  
**import** matplotlib.pyplot **as** plt *#画图用的  
#魔法函数*%matplotlib inline  
  
result = {} *#建立空列表  
# 记录次数，格式为{nums:[name,自问自答次数，评论次数，签到次数]}***with** open(**'名单.csv'**, **'r'**,encoding=**'utf-8-sig'**) **as** f: *#encoding='utf-8-sig'编码* reader = csv.reader(f)  
 **for** row **in** reader:  
 result[row[0]]=[row[1],0,0,0]  
  
result  
  
browser = webdriver.Chrome() *#selenium的定义*browser.get(**"https://www.icourse163.org/"**)  
browser.get(**'https://www.icourse163.org/learn/CUEB-1450000234?tid=1461631472#/learn/content'**)  
time.sleep(2) *#暂停俩秒  
# 先把第一章的菜单合上*browser.find\_element\_by\_xpath(**r'//\*[@class="titleBox j-titleBox f-cb"]'**).click()  
  
**def** re(str\_of\_re:str):  
 str\_of\_re = str\_of\_re.lower()  
 nums = re.compile(**r"(?<=\+)\d{11}"**)  
 num = nums.findall(str\_of\_re)  
 **return** num  
**def** cal\_zwzd(nums):  
 **if** nums **in** result:  
 result[nums][1]+=1  
**def** cal\_pl(nums):  
 **if** nums **in** result:  
 result[nums][2]+=1  
**def** cal\_qd(nums):  
 **if** nums **in** result:  
 result[nums][3]+=1  
  
**def** start\_pa (i):  
 browser.get(**'https://www.icourse163.org/learn/CUEB-1450000234?tid=1461631472#/learn/content'**)  
 time.sleep(1)  
 chapter = browser.find\_elements\_by\_xpath(**r'//\*[@class="titleBox j-titleBox f-cb"]'**)  
 chapter = chapter[:-1]  
*# 第一个菜单合上* suc = 1*#安全作用* **while**(suc):  
 **try**:  
 browser.find\_element\_by\_xpath(**r'//\*[@class="titleBox j-titleBox f-cb"]'**).click()  
 suc = 0  
 **except**:  
 **pass***# 点开第i个菜单* chapter[i].click()  
*# 点开在里面找本章收获* time.sleep(1)  
 **try**:  
 questions = browser.find\_elements\_by\_xpath(**r'//\*[@class="j-name name f-fl f-thide"]'**)  
 **for** one\_question **in** questions:  
 **if "本章收获与问题是什么" in** one\_question.text:  
 one\_question.click()  
 **break  
 except**:  
 print(**f"第{i}个菜单有问题"**)  
 ye = 1  
 *# 看看有多少页* time.sleep(1)  
 a = browser.find\_elements\_by\_xpath(**r'//\*[@id="courseLearn-inner-box"]/div/div/div[3]/div[1]/div[2]/div/div/div[4]/div/div[1]/div[2]//a'**)  
 ye\_num = 1  
 **try**:  
 ye\_num = int(a[-2].text)  
 **except**:  
 print(**"只有一页"**)  
 **while**(ye <= ye\_num):  
 **try**:  
 *# 找所有自问自答* print(**f"正在爬第{ye}页"**)  
 answer = browser.find\_elements\_by\_xpath(**r'//div[contains(@class,"m-detailInfoItem")]/div[2]/p'**)  
 **for** one\_answer **in** answer:  
 **if "刘经纬" in** one\_answer.text:  
 **continue  
 elif '++' in** one\_answer.text **and '--'in** one\_answer.text **and '=='in** one\_answer.text:  
 nums= reg(one\_answer.text)  
 time.sleep(1)  
 **for** one\_num **in** range(len(nums)):  
 cal\_zwzd(nums[one\_num])  
 *# 找所有评论和签到  
 # 如果评论数很大就是签到，很小就是评论* comment = browser.find\_elements\_by\_xpath(**r'//div[contains(@class,"m-comment-pool")]'**)  
 qd\_list =[]  
 **for** comment\_num **in** range(len(comment)):  
 c = reg(comment[comment\_num].text)  
 **if** len(c)>10:  
 *# 就是签到* **for** qd\_num **in** range(len(c)):  
 cal\_qd(c[qd\_num])  
 **if** len(c)==20:  
*# 如果有评论数是20的就记下来，可能需要点下一页。* qd\_list.append(comment\_num)   
 **if** len(c)<=5:  
 *# 就是评论* **for** m **in** range(len(c)):  
 cal\_pl(c[m])   
 *# 读完了之后把每一个下一页都点开* next\_button = browser.find\_elements\_by\_xpath(**r'//a[contains(@class,"zbtn znxt")]'**)  
 **except**:  
 **pass  
 if** len(qd\_list)>0:  
 **try**:  
 **for** nxt **in** qd\_list:  
 next\_button[nxt].click()  
 time.sleep(1)  
 comment = browser.find\_elements\_by\_xpath(**r'//div[contains(@class,"m-comment-pool")]'**)  
 qd\_list\_again = []  
 **for** nxt **in** qd\_list:  
 c = reg(comment[nxt].text)  
 **if** len(c)==20:  
 qd\_list\_again.append(comment\_num)  
 **for** qd\_num **in** range(len(c)):  
 cal\_qd(c[qd\_num])  
 **if** len(qd\_list\_again)>0:  
 **for** nxt **in** qd\_list\_again:  
 next\_button[nxt].click()  
 time.sleep(1)  
 comment = browser.find\_elements\_by\_xpath(**r'//div[contains(@class,"m-comment-pool")]'**)  
 **for** nxt **in** qd\_list\_again:  
 c = reg(comment[nxt].text)  
 **for** qd\_num **in** range(len(c)):  
 cal\_qd(c[qd\_num])  
 **except**:  
 **pass** ye+=1  
 **try**:  
 next\_page = browser.find\_elements\_by\_xpath(**r'//a[contains(@class,"zbtn znxt")]'**)  
 next\_page[-1].click()  
 time.sleep(1)  
 **except**:  
 **pass** print(**"当前结果是："**)  
 print(result)  
  
**for** i **in** range(9):  
 print(i)  
 start\_pa(i)  
  
result  
  
df = pd.DataFrame.from\_dict(result,orient=**'index'**,columns=[**'姓名'**,**'自问自答次数'**,**'评论次数'**,**'签到次数'**])  
  
df  
df.to\_csv(**'ljw\_result.csv'**,encoding=**"utf-8-sig"**)  
  
df = pd.read\_csv(**'ljw\_result.csv'**,encoding=**'utf-8-sig'**)  
df.set\_index([**"Unnamed: 0"**], inplace=**True**)  
df.index.name=**''**df.head(30)  
  
df = df.drop(df[df[**'自问自答次数'**]==0].index)  
name = df[**'姓名'**]  
df = df.drop([**'姓名'**], axis=1)  
  
df.describe()  
  
**from** sklearn.model\_selection **import** train\_test\_split  
**from** sklearn.svm **import** SVC  
  
train\_X,test\_X= train\_test\_split(df,test\_size=0.7,random\_state=5)  
train\_X  
  
train\_label = [89,90,87,86,84,92,98,96,91,80,99,89,88,88,89,92,95,97,83,85,82,86,93,87,93,90,94,81,90,91]  
test\_X  
  
C = 0.1  
clf = SVC(kernel = **"linear"**, C = C)  
  
clf.fit(train\_X,train\_label)  
test\_pred = clf.predict(df)  
  
test\_pred  
  
df[**'label'**]=test\_pred  
df[**'姓名'**]=name  
  
df  
  
df.to\_csv(**'final\_result.csv'**,encoding=**"utf-8-sig"**)