****

**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](#_Toc60049000)

[二、数据预处理 3](#_Toc60049001)

[（一）数据清洗 3](#_Toc60049002)

[（二）数据集成 4](#_Toc60049003)

[（三）数据变换 4](#_Toc60049004)

[（四）数据规约 5](#_Toc60049005)

[三、K-means算法 5](#_Toc60049006)

[（一）概念 5](#_Toc60049007)

[（二）算法流程 6](#_Toc60049008)

[（三）代码 6](#_Toc60049009)

[四、随机森林算法 8](#_Toc60049010)

[（一）随机森林算法由来 8](#_Toc60049011)

[（二）随机森林的建立过程 8](#_Toc60049012)

[（三）决策树概念 9](#_Toc60049013)

[（四）使用决策树算法的主要步骤 9](#_Toc60049014)

[五、KNN（K最近邻居算法）算法 9](#_Toc60049015)

[（一）原理 9](#_Toc60049016)

[（二）主要实现过程 9](#_Toc60049017)

[六、Bagging（套袋法）、Boosting（提升法） 10](#_Toc60049018)

[（一）bagging的算法 10](#_Toc60049019)

[（二）boosting的算法 10](#_Toc60049020)

[七、SVM算法 10](#_Toc60049021)

[八、岭回归 11](#_Toc60049022)

[（一）原理 11](#_Toc60049023)

[（二）缺点 11](#_Toc60049024)

[（三）适用情况 11](#_Toc60049025)

[九、爬虫 11](#_Toc60049026)

# 一、数据抓取

数据抓取：常用抽取网页数据的方式有三种：**正则表达式、Beautiful Soup、lxml。**

|  |  |  |  |
| --- | --- | --- | --- |
| 三种方式的比较 | | | |
| 抓取方法 | 性能 | 使用难度 | 安装难度 |
| 正则表达式 | 快 | 困难 | 简单（内置模块） |
| Beautiful Soup | 慢 | 简单 | 简单（纯python） |
| Lxml | 快 | 简单 | 相对困难 |

# 二、数据预处理

**数据预处理**主要包括**数据清洗、数据集成、数据变换**和**数据规约**四个部分。

## （一）数据清洗

### 1、概念

删除原始数据集中的无关数据、重复数据、平滑噪声数据，处理缺失值、异常值等。

### ****2、数据清洗的步骤****

（1）缺失值处理（通过describe与len直接发现、通过0数据发现）

（2）异常值处理（通过散点图发现）

一般遇到缺失值，处理方式为：删除、插补、不处理。

插补的方式主要有：均值插补、中位数插补、众数插补、固定值插补、最近数据插补、回归插补、插值法等等。

一般遇到异常值，一般处理方式为：视为缺失值、删除、修补（平均数、中位数等等）。

## ****（二）数据集成****

将多个数据源合并存在一个一致的数据存储中，要考虑实体识别问题和属性冗余问题，从而将数据在最低层上加以转换、提炼和集成。

## ****（三）数据变换****

### 1、概念

即对数据规范化处理，主要包括以下几种：

（1）**简单函数变换**。（2）**数据规范化**。（3）**数据离散化**。（4）**属性构造。**

### 2、简单函数变换的常见方法

平方、开方、对数、差分运算等。

### ****3、数据规范化的常见方法****

（1）离差标准化（最小-最大标准化）--消除量纲（单位）影响以及变异大小因素的影响。（最小-最大标准化）

x1=（x-min）/（max-min）

（2）标准差标准化--消除单位影响以及变量自身变异影响。（零-均值标准化）

x1=（x-平均数）/标准差

（3）小数定标规范化--消除单位影响

x1= x/10\*\*(k)、k=log10(x的绝对值的最大值)

### 4、 ****数据离散化的常见方法****

   （1）等宽离散化。

   （2）等频率离散化。

   （3）一维聚类离散化。

### 5、****属性构造****

即通过现有的一个或多个属性数据经过处理变成新的属性数据。

## ****（四）数据规约****

### ****1、概念****

产生更小且保持数据完整性的新数据集。意义在于降低无效、错误数据；降低存储成本；少量且具有代表性的数据大幅加快，主要分为属性违约和数值违约。

### 2、****属性规约****

（1）属性合并或删除无关维，目标是寻找最小子集使子集概率分布尽可能与原来相同。

（2）常用方法

①合并属性 将就属性合并为新属性 {A1,A2,A3,B1,B2,C}——{A,B,C}  
  ②逐步向前选择 从空集开始，逐个加入最优属性，直到无最优或满足条件 {}—{A1}—{A1,A4}  
   ③逐步向后删除 从全集开始，每次删除最差属性，直到无最差或满足阈值  
  ④决策树归纳 利用决策树归纳能力进行分类，删除未出现的属性，即效果差的属性  
  ⑤主成分分析 用少量变量解释大部分变量，保留大部分信息，将相关性高的数据转为彼此独立

### 3、****数值规约****

通过选择替代的、较小的数据来较少数据量，包括有参数(回归、对数线性模型)和无参数方法(直方图、聚类、抽样)。

# 三、K-means算法

## （一）概念

K-means算法中的k表示的是聚类为k个簇，means代表取每一个聚类中数据值的均值作为该簇的中心，或者称为质心，即用每一个的类的质心对该簇进行描述。即，K-Means算法接受参数K；然后将样本数据集划分为K个聚类。获得的聚类需要满足：同一个聚类中的样本数据集相似度较高；而不同聚类中的样本数据集相似度较小。

算法思想为：以空间中K个点为中心进行聚类（即先从样本集中随机选取 k个样本作为簇中心），对最靠近他们的对象归类（所有样本与这 k个“簇中心”的距离，对于每一个样本，将其划分到与其距离最近的“簇中心”所在的簇中）。通过迭代的方法，逐次更新各聚类中心的值，直至得到最好的聚类结果。

## （二）算法流程

第一步，先从没有标签的元素集合A中随机取K个元素，作为K个子集各自的质心。

第二步，分别计算剩下的元素到K个子集质心的距离，根据距离将元素分别划分到最近的子集。

第三步，根据聚类结果，重新计算质心(计算方法为子集中所有元素各个维度的算术平均数)

第四步，将集合A中全部元素按照新的质心然后再重新聚类。

重复第四步，直到聚类结果不再发生变化。

## （三）代码

**import** math  
**import** random  
**import** pandas **as** pd  
  
*#计算距离***def** distance(a,b):  
 distan = math.sqrt((a[0] - b[0])\*\*2+(a[1] - b[1])\*\*2)  
 **return** distan  
  
*#D：数据集，k：簇数，maxIter：最大迭代次数，：簇划分的集合，C：质心集合***def** kmeans(D,k,maxIter):  
 **if**(k>=len(D)):  
 **return** D  
 C = []  
 U = []  
 *#初始化簇划分集合* **for** i **in** range(k):  
 U.append([])  
 *#从D中随机选择k个样本作为初始质心{u1,u2,...,uk}* ramdomNum = random.sample(range(0,len(D)),k)  
 **for** i **in** range(k):  
 C.append(D[ramdomNum[i]])  
 Iter = 0  
 tag = 0  
 temp = 0  
 C1 = C  
 *#计算各点到各质心的距离，划入距离最小的质心的簇。* **while**(Iter < maxIter):  
 i = 0  
 **while** (i < len(D)):  
 j = 0  
 distan = 1000  
 **while** (j < k):  
 temp = distance(D[i], C[j])  
 **if** (temp < distan):  
 tag = j  
 distan = temp  
 j = j + 1  
 U[tag].append(D[i])  
 i = i + 1  
 *# 在小于最大迭代次数时，找出新质心* C.clear()  
 Iter = Iter + 1  
 **if**(Iter < maxIter):  
 **break  
 for** num **in** range(k):  
 C.append(findNew(U[num]))  
 U[num].clear()  
 **if**(C1 == C & Iter >=5 ):  
 **break** C1 = C  
 **return** U  
  
*#找出新质心***def** findNew(L):  
 x = 0  
 y = 0  
 tup = ()  
 **for** i **in** range(len(L)):  
 x = x + L[i][0]  
 y = y + L[i][1]  
 tup = (x/len(L),y/len(L))  
 **return** tup

**if** \_\_name\_\_ ==**"\_\_main\_\_"**:  
 dataset = pd.read\_csv(**'data.csv'**,delimiter=**','**)  
 data = dataset.values.tolist()  
 D = []  
 **for** i **in** range(len(data)):  
 D.append(tuple(data[i]))  
 num = int(input(**"请输入簇数："**));  
 U = kmeans(D,num,10)  
 **print**(U)

# 四、随机森林算法

## （一）随机森林算法由来

集成学习（ensemble）思想是为了解决单个模型或者某一组参数的模型所固有的缺陷，从而整合起更多的模型，取长补短，避免局限性。随机森林就是集成学习思想下的产物，将许多棵决策树整合成森林，并合起来用来预测最终结果。

## （二）随机森林的建立过程

第一步：原始训练集中有N个样本，且每个样本有W维特征。从数据集D中有放回的随机抽取x个样本（Bootstraping方法）组成训练子集，一共进行w次采样，即生成w个训练子集。

第二步：每个训练子集形成一棵决策树，形成了一共w棵决策树。而每一次未被抽到的样本则组成了w个oob（用来做预估）。

第三步：对于单个决策树，树的每个节点处从M个特征中随机挑选m（m<M）个特征，按照结点不纯度最小原则进行分裂。每棵树都一直这样分裂下去，直到该节点的所有训练样例都属于同一类。在决策树的分裂过程中不需要剪枝。

第四步：根据生成的多个决策树分类器对需要进行预测的数据进行预测。根据每棵决策树的投票结果，如果是分类树的话，最后取票数最高的一个类别；如果是回归树的话，利用简单的平均得到最终结果。

## （三）决策树概念

决策树(decision tree)  是一种分类与回归方法，主要用于分类，决策树模型呈现树形结构，是基于输入特征对实例进行分类的模型。我认为决策树其实是定义在特征空间与类空间上的条件概率分布！

## （四）使用决策树算法的主要步骤

可以分为三部分： 1特征的选取 2 决策树的生成 3 决策树的修剪

当前决策树的主要流行的三个算法是 ID3算法 C4.5算法和 CART算法

# 五、KNN（K最近邻居算法）算法

## （一）原理

给定一个样本，k-最临近分类法搜索模式空间，找出最接近未知样本的k个训练样本。

通过计算每个训练数据到待分类元组的距离，取和待分类元组距离最近的K个训练数据，K个数据中哪个类别的训练数据占多数，则待分类元组就属于哪个类别。

## （二）主要实现过程

1.计算训练样本和测试样本中每个样本点的距离(常见的距离度量有欧式距离，马氏距离等)

2.对上面所有的距离值进行排序

3.选前k个最小距离的样本

4.根据这k个样本的标签进行投票，得到最后的分类类别

# 六、Bagging（套袋法）、Boosting（提升法）

## （一）bagging的算法

从原始样本集中使用Bootstraping方法随机抽取n个训练样本，共进行k轮抽取，得到k个训练集（k个训练集之间相互独立，元素可以有重复）。

对于k个训练集，我们训练k个模型（这k个模型可以根据具体问题而定，比如决策树，knn等）。

对于分类问题：由投票表决产生分类结果；对于回归问题：由k个模型预测结果的均值作为最后预测结果（所有模型的重要性相同）。

## （二）boosting的算法

对于训练集中的每个样本建立权值wi，表示对每个样本的关注度。当某个样本被误分类的概率很高时，需要加大对该样本的权值。

进行迭代的过程中，每一步迭代都是一个弱分类器。我们需要用某种策略将其组合，作为最终模型（例如AdaBoost给每个弱分类器一个权值，将其线性组合最为最终分类器。误差越小的弱分类器，权值越大）。

# 七、SVM算法

支持向量机（Support Vector Machine，简称SVM）可以理解为“使用了支持向量的算法”。支持向量机是一种基于分类边界的方法。

其基本原理是（以二维数据为例）：如果训练数据分布在二维平面上的点，它们按照其分类聚集在不同的区域。基于分类边界的分类算法的目标是：通过训练，找到这些分类之间的边界（如果是直线的，称为线性划分，如果是曲线的，称为非线性划分）。

SVM主要思想是针对两类分类问题，在高维空间中寻找一个超平面作为两类的分割，以保证最小的分类错误率。

SVM一个重要的优点是可以处理线性不可分的情况，在解决小样本、非线性及高维（避免维灾难）模式识别中表现出许多特有的优势，并能够推广应用到函数拟合等其他机器学习问题中。

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"**)