Week3-学习笔记心得

简单整理了下本节课思维导图

Diagram, schematic

Description automatically generated

## 原理 为什么需要用户画像，对于业务来说有何帮助

用户画像可以帮助更好的提升业务，增加收益

## 用户画像的准则

Uuid –手机登陆，淘宝用户登陆，微信用户登陆

淘宝优酷的打通：

1.step1 统一标识

2.step2 给用户打标签

3.step3 基于标签指导业务

## 原理 有哪些维度可以用来设计用户标签

八字原则：用户消费行为分析  
2. 用户标签：年龄、性别、职业、学历  
3. 消费标签： 消费习惯  
4. 行为标签：浏览时长、次数  
5. 内容分析：用户浏览内容分析

## 原理 用户生命周期的三个阶段

1.获客：如何进行拉新（双十一，不是这个店的用户，也会收到信息）  
2.粘客：个性化推荐，搜索排序，场景运营等  
3.留客：流失率预测，分析关键节点降低流失率

## 数据流架构

A picture containing text

Description automatically generated

## 原理 标签从何而来

1. PGC 专家生产  
2. UGC 普通生产

标签是对高维事物的抽象（降维）

很多滴滴出事的情况，就是因为司机可以看到用户的标签，对用户的标签打的太好了，（比如说这是一个美女）

## 聚类算法

（不需要事先有经验，就可以进行训练, 是无监督的学习）

K-Means，EM聚类，Mean-Shift，DBSCAN，层次聚类，PCA

DBSCAN(Density-Based Spatial Clustering of Applications with Noise，具有噪声的基于密度的聚类方法)是一种很典型的密度聚类算法

### 1. 聚类算法 K-Means工作原理

（k个初始化的质心的位置选择对最后的聚类结果和运行时间都有很大的影响）

1. 随机选取K个点作为初始的类中心点

（对于K-Means算法，首先要注意的是k值的选择，一般来说，我们会根据对数据的先验经验选择一个合适的k值，如果没有什么先验知识，则可以通过交叉验证选择一个合适的k值或者从数据集中随机抽取）  
2. 将每个点分配到最近的类中心点，形成K个类，然后重新计算每个类的中心点  
3. 重复步骤2，直到类中心点不再发生变化；或者也可以设置最大迭代次数，只要达到最大迭代次数就会结束

#### K-Means优缺点

K-Means的主要优点有：

　　　　1）原理比较简单，实现也是很容易，收敛速度快。

　　　　2）聚类效果较优。

　　　　3）算法的可解释度比较强。

　　　　4）主要需要调参的参数仅仅是簇数k。

K-Means的主要缺点有：

　　　　1）K值的选取不好把握

　　　　2）对于不是凸的数据集比较难收敛

　　　　3）如果各隐含类别的数据不平衡，比如各隐含类别的数据量严重失衡，或者各隐含类别的方差不同，则聚类效果不佳。

　　　　4） 采用迭代方法，得到的结果只是局部最优。

　　　　5） 对噪音和异常点比较的敏感。

#### K-Means与KNN区别

K-Means是无监督学习的聚类算法，没有样本输出；而KNN是监督学习的分类算法，有对应的类别输出。KNN基本不需要训练，对测试集里面的点，只需要找到在训练集中最近的k个点，用这最近的k个点的类别来决定测试点的类别。而K-Means则有明显的训练过程，找到k个类别的最佳质心，从而决定样本的簇类别。

### 2.聚类算法EM

thinking：你炒了一份菜，想要把它平均分到两个碟子里，该怎么分？

主要步骤：

* 初始化参数、观察预期、重新估计
* 首先是先给每个碟子初始化一些菜量，然后再观察预期，这两个步骤实际上就是期望步骤（Expectation）
* 如果结果存在偏差就需要重新估计参数，这个就是最大化步骤（Maximization）

Diagram

Description automatically generated

EM 算法是一种求解最大似然估计的方法，

1.通过观测样本，来找出样本的模型参数

2.通过EM算法中的E步来进行观察，然后通过M步来进行调整A和B的参数，最后让碟子A和碟子B的参数不再发生变化为止

### 3.聚类算法GMM（GaussianMixtureModel 高斯混合模型）

用高斯概率密度函数（二维时也称为正态分布曲线）

常用的聚类算法，使用了EM算法进行迭代计算

模型思想：

任何一个曲线，无论多么复杂，都可以用若干个高斯曲线来无限逼近它

混合高斯就是通过求解多个单高斯模型，并通过一定的权重将多个单高斯模型融合成一个模型，即最终的混合高斯模型

假设混合高斯模型由K个高斯模型组成（即K个类），则GMM的概率密度函数



Text

Description automatically generated表示第k个高斯模型的概率密度函数，可以看成选定第k个模型后，该模型产生x的概率

p(k)=πk 是第k个高斯模型的权重，称作选择第k个模型的先验概率，且满足



GMM的目的就是找到一个合适的高斯分布（也就是确定高斯分布的参数μ,Σ），使得这个高斯分布能产生这组样本的可能性尽可能大

高斯混合模型是聚类，是机器学习中对“无标签数据”进行训练得到的分类结果（分类结果由概率表示，概率大者，则认为属于这一类）

### 4.Kmeans++

K-means 有三种改进算法：

K-means++,ISODATA和Kernel K-means

 首先需要明确的是上述四种算法都属于"**硬聚类**”算法，即数据集中每一个样本都是被100%确定得分到某一个类别中，与之相对的"**软聚类**”可以理解为每个样本是以一定的概率被分到某一个类别中

### 5.Faiss

Facebook开源的一款稠密向量检索工具

Faiss是Facebook AI团队开源的针对聚类和相似性搜索库，为稠密向量提供高效相似度搜索和聚类，支持十亿级别向量的搜索(具体就是给定一个向量，在所有已知的向量库中找出与其相似度最高的一些向量，本质是一个KNN(K近邻)问题)，是目前最为成熟的近似近邻搜索库。

它是一个能使开发者快速搜索相似多媒体文件的算法库。而该领域一直是传统的搜索引擎的短板。借助Faiss，Facebook 在十亿级数据集上创建的最邻近搜索（nearest neighbor search），比此前的最前沿技术快 8.5 倍，并创造出迄今为止学术圈所见最快的、运行于 [GPU](https://cloud.tencent.com/product/gpu?from=10680) 的 k-selection 算法。Facebook 人工智能实验室（FAIR） 借此创造了数个世界纪录，包括在十亿高维矢量上的构建的、世界最快的 k-nearest-neighbor 图。

## K-Means工具

### 工具 如何使用K-Means对球队进行聚类 球队

1。首先python 中提供了preprocessing这样一个类，preprocessing中有一个MinMaxScaler函数

from sklearn.cluster import KMeans  
 from sklearn import preprocessing  
 min\_max\_scale=preprocessing.MinMaxScaler()  
2. 针对MinMaxScaler这个函数，我们需要把我们最开始的的特征train\_x 提供进去，然后让它去fit,得到一个新的train\_x  
 train\_x=min\_max\_scaler.fit\_transform(train\_x)  
3. 对于新的到的train\_x 我们再用k-means算法对他进行预测  
 # kmeans算法  
 kmeans.fit(train\_x)  
 predict\_y = kmeans.predict(train\_x)  
  
 （注：聚类是无监督的学习，具体的含义还需要我们指定；  
 什么情况下会使用聚类：  
 1。人工打标签成本太高  
 2。如果不知道改怎么打标签，可以让机器先跑出几类，再去看下这每个类别到底代表什么含义，通过后面再做具体的分析，给类别一个有含义的标识。  
 也就是说缺乏一些经验知识的时候可以使用聚类）

* from sklearn.cluster import KMeans
* KMeans(n\_clusters=8, max\_iter=300)

n\_clusters：聚类个数，缺省值为8

max\_iter：执行一次k-means算法所进行的最大迭代数，缺省值为300

# coding: utf-8  
# 手肘法  
from sklearn.cluster import KMeans  
from sklearn import preprocessing  
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
# 输入数据  
data = pd.read\_csv('team\_cluster\_data.csv', encoding='gbk')  
train\_x = data[["2019国际排名","2018世界杯排名","2015亚洲杯排名"]]  
# 规范化到 [0,1] 空间  
min\_max\_scaler=preprocessing.MinMaxScaler()  
train\_x=min\_max\_scaler.fit\_transform(train\_x)  
# 统计不同K取值的误差平方和  
sse = []  
for k in range(1, 11):  
 # kmeans算法  
 kmeans = KMeans(n\_clusters=k)  
 kmeans.fit(train\_x)  
 # 计算inertia簇内误差平方和  
 sse.append(kmeans.inertia\_)  
x = range(1, 11)  
plt.xlabel('K')  
plt.ylabel('SSE')  
plt.plot(x, sse, 'o-')  
plt.show()  
# K增大到一定程度时，K增大对SSE减小的作用越来越小  
# 这里设置K=3  
kmeans = KMeans(n\_clusters=3)  
kmeans.fit(train\_x)  
predict\_y = kmeans.predict(train\_x)  
# 合并聚类结果，插入到原数据中  
result = pd.concat((data,pd.DataFrame(predict\_y)),axis=1)  
result.rename({0:u'聚类结果'},axis=1,inplace=True)  
print(result)  
# 将结果导出到CSV文件中  
#result.to\_csv("team\_cluster\_result.csv")

### 原理 手肘法判断K值

SSE指的是不同K取值的误差平方和  
手肘法的核心思想：  
随着聚类数K的增大，样本划分会更加精细，每个簇的聚合程度会逐渐提高，那么误差平方和SSE自然会逐渐变小。  
并且，当k小雨真实聚类数时，由于k的增大会大幅增加每个簇的聚合程度，故SSE的下降幅度会很大；  
当k到达真实聚类数时，再增加k所得的聚合程度回报会迅速变小，所以SSE的下降幅度会骤减，然后会随着  
K值的继续增大而趋于平缓，也就是说SSE和k的关系图是一个手肘的形状，而这个肘部对应的k值就是数据的真实聚类数。  
K-Means 手肘法：（看的是k=?时的斜率）  
 # 统计不同K取值的误差平方和  
 sse = []  
 for k in range(1, 11): 即（1，10]区间  
 # kmeans算法  
 kmeans = KMeans(n\_clusters=k)  
 kmeans.fit(train\_x)  
 # 计算inertia簇内误差平方和(当k=1的时候，误差平方和最大，当k=10的时候，误差平方和最小)  
 sse.append(kmeans.inertia\_)  
 x = range(1, 11)  
 plt.xlabel('K')  
 plt.ylabel('SSE')  
 plt.plot(x, sse, 'o-')  
 plt.show()  
当k=3或者k=4的时候，手肘变化会越来越小，这个点比较好

### K-Means 在图像分割中的应用

# 用K-Means对图像进行2聚类  
kmeans =KMeans(n\_clusters=2)  
kmeans.fit(img)  
label = kmeans.predict(img)  
# 将图像聚类结果，转化成图像尺寸的矩阵  
label = label.reshape([width, height])  
# 创建个新图像pic\_mark，用来保存图像聚类的结果，并设置不同的灰度值  
pic\_mark = image.new("L", (width, height))  
for x in range(width):  
 for y in range(height):  
 # 根据类别设置图像灰度, 类别0 灰度值为255， 类别1 灰度值为127  
 pic\_mark.putpixel((x, y), int(256/(label[x][y]+1))-1)  
  
K-Means图像分割的不足：（只相当于对颜色进行了分类）  
按照图像的灰度值定义距离，对图像的文理，内容缺乏理解  
  
采用深度学习模型：（效果比较好，可以把整个人的形状抠出来）  
100层 Tiramisu进行图像分割  
（全卷积 DenseNets)  
  
总结：K-Means是聚类的一种方式，可用于多种应用场景：足球队的应用，图像分割、人群挖掘等，最核心的是对距离的定义（用灰度值对距离进行定义）

## GMM 工具

GMM工具：

from sklearn.mixture import GaussianMixture

GaussianMixture(n\_components=1, covariance\_type="full")

n\_components：聚类个数，缺省值为1

covariance\_type：协方差类型，一共4种，默认为full

covariance\_type="full",完全协方差矩阵（元素都不为零）

covariance\_type="tied",相同的完全协方差矩阵（HMM会用到）

covariance\_type="diag",对角协方差矩阵（非对角为零，对角不为零）

covariance\_type="spherical",球面协方差矩阵（非对角为零，对角完全相同，球面特性）

## 原理 距离都有哪些表示方式

1。曼哈顿距离  
2。欧式距离  
3。余弦距离  
4。切比雪夫距离

## 原理 数据规范化的方法

任何设计到距离的都要做数据规范化

### 1。Min-max 最小最大规范化

新数值=（原数值-极小值）/ （极大值-极小值）

### Graphical user interface, chart Description automatically generated 2。Z-Score

新数值= （原数值-均值）/ 标准差

Diagram, engineering drawing

Description automatically generated

### 3。小数定标规范化(使用场景不多)

通过移动小数点位置来进行规范化  
   


## SimpleTagBased算法

### 工具 使SimpleTagBased算法进行TOP-N推荐

Delicious 数据集

https://grouplens.org/datasets/hetrec-2011/

1867名用户，105000个书签，53388个标签

https://grouplens.org/datasets/hetrec-2011/

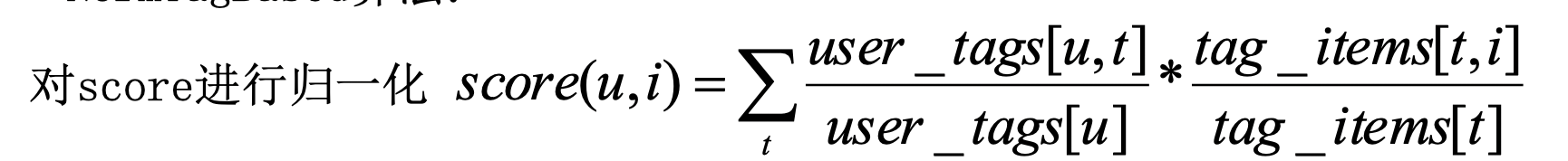
格式：userID bookmarkID tagID timestamp

(已经对这些数据打上了标签，如何利用用户标签做一些推荐呢？)  
SimpleTagBased算法：  
1。统计每个用户的常用标签（用户画像）  
2。对于每个标签，统计被打过这个标签次数最多的商品（Item的特征提取）  
3。对于一个用户，找到他常用的标签，然后找到具有这些标签的最热门的物品推荐给他（用户画像和Item之间的关联，基于内容的推荐）

Schematic

Description automatically generated  
  
用户u对商品i的兴趣，用户和商品之间的关联可能不止一个，可能有多个tag1 tag2 tag3,所以需要对每个tag都做  
计算，然后把所有的score累加起来  
  
以上算法可能存在一些问题：  
一个用户可能会打多个标签，也就是这个标签在用户画像中不明显，所以NormTagBased是对这个算法的改进  
  
数据结构的定义：  
用户打标签记录：records[i]={user,item,tag}  
用户打过的标签：user\_tags[u][t]  
用户打过标签的商品： user\_items[u][i]  
打上某标签的商品： tag\_items[t][i]  
某标签使用过的用户：tag\_user[t][u]

## NormTagBased算法



## TagBased-TFIDF算法

### 原理 什么是TF-IDF

什么是TF?  
TF： Term frequency 词频 TF = 单词次数/文档中总单词数  
一个单词的重要度和它在文档中出现的次数成正比。



IDF: Inverse Document Frequency 逆向文档频率  
一个单词在文档中的区分度。

这个单词这文档中出现的次数越少，证明这个单词的区分度就越大。IDF就越大。  
IDF=log(文档总数/（单词出现的文档数+1）)



如果一个tag很热门，就会导致user\_tags[t]很大，所以即使tag\_items[t,i]很小，最后导致score非常大，也就是说热门标签因为权重较大，所以不能正常反应用户个性化的兴趣。

这里借鉴TF-IDF的思想，使用tag\_users[t]表示标签t被多少个不同的用户使用



这里也就是对标签度做了一个区分度，如果使用的比较多，说明这个标签也没有什么区分度，如果使用的不多，但是在这个用户下使用的比较多，就说明在这个用户下区分度比较高。

## 原理 评测指标：准确率，召回率，精确率，F值

Table

Description automatically generated

准确率：我们去做一个预测，有可能预测结果为正，有可能预测结果为负  
 我们把预测结果为正的或者负的都对的数（都正确的数）/总的样本数---》》即正确率/准确率  
  
精确率： 只把预测结果为正的且实际也为正的/预测结果为正的（实际为正和为负的所有结果）  
  
准确率和精确率的不同：  
精确率只考虑预测结果为正的情况的准确率，

准确率考虑的维度更多，不光是预测结果为正的时候，还有预测结果为负的时候，只要预测结果是对的  
  
召回率：预测结果为正的情况/实际结果为正的情况下预测结果为正和为负的总和，也就是说实际为正的情况下，我们猜对的比例

## 如何给用户推荐标签

当用户u给物品i打标签时，可以给用户推荐和物品i相关的标签，方法如下：

* 方法1：给用户u推荐整个系统最热门的标签
* 方法2：给用户u推荐物品i上最热门的标签
* 方法3：给用户u推荐他自己经常使用的标签
* 将方法2和3进行加权融合，生成最终的标签推荐结果

## 原理 基于内容的推荐系统步骤

1. 标签化 为每个item抽取出features  
2. 特征学习（profile learning）通过以前用户喜欢或者不喜欢的item的特征数据，来学习该用户的喜好特征（profile）  
3. 生成推荐列表,计算score进行排序 通过用户的profile与候选的item的特征，来推荐相关性最大的item.

## 自动机器学习

### 工具 TPOT: 基于python的AutoML工具

TPOT https://github.com/EpistasisLab/tpot （6.2K）

* 目前只能做有监督学习
* 支持的分类器主要有贝叶斯、决策树、集成树、SVM、KNN、线性模型、xgboost
* 支持的回归器主要有决策树、集成树、线性模型、xgboost
* 数据预处理：二值化、聚类、降维、标准化、正则化等
* 特征选择：基于树模型、基于方差、基于F-值的百分比
* 可以实现通过代码来写代码，可以说是机器学习的机器学习，可以通过export()方法把训练过程导出为形式为sklearn pipeline的.py文件

TPOT可以解决：  
1。特征选择（基于树模型、基于方差、基于F-值的百分比），  
2。模型选择，  
3。数据预处理（二值化、聚类、降维、正则化等）  
但不包括数据清洗  
处理小规模数据非常快，大规模数据非常慢。可以先抽样小部分，使用TPOT

限制TPOT的使用的主要还是慢，未来的提升主要是效率提升，就可以让更多的人使用

我们只需要输入给TPOT一个清理后的data,TPOT就可以输出给我们什么是最优的一个模型，以及模型的参数。  
参数介绍：  
# generations 代表优化的迭代的次数  
# population\_size 代表遗传算法里个体的数量，采用遗传算法来找最优解，而这里最优解是个模型  
# verbosity tpot运行时能传递多少信息

Diagram

Description automatically generated

### pip国内的镜像

阿里云 http://mirrors.aliyun.com/pypi/simple/  
中国科技大学 https://pypi.mirrors.ustc.edu.cn/simple/  
豆瓣(douban) http://pypi.douban.com/simple/  
清华大学 https://pypi.tuna.tsinghua.edu.cn/simple/  
中国科学技术大学 http://pypi.mirrors.ustc.edu.cn/simple/  
  
使用方法：  
pip install tensorflow -i https://pypi.tuna.tsinghua.edu.cn/simple

### MNIST的10种解法：

Table

Description automatically generated

算法 工具  
Logistic Regression  
from sklearn.linear\_model import LogisticRegression  
  
CART，ID3（决策树）  
from sklearn.tree import DecisionTreeClassifier  
决策树学习采用的是自顶向下的递归方法，其基本思想是以信息熵为度量构造一颗熵值下降最快的树，  
到叶子节点处，熵值为0。其具有可读性、分类速度快的优点，是一种有监督学习。  
ID3 Cart C4.5区别：  
是因为对信息的判断尺度不一样，不同的标准会产生不同的算法，一个树的方式，不同的节点采用什么样的方式  
最早提及决策树思想的是Quinlan在1986年提出的ID3算法(采用信息增益的方式)  
和1993年提出的C4.5算法，  
以及Breiman等人在1984年提出的CART算法(采用信息增益率的方式)  
  
LDA(线形判别分析)  
from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis   
  
朴素贝叶斯  
from sklearn.naive\_bayes import BernoulliNB  
朴素贝叶斯的思想基础是这样的：  
对于给出的待分类项，求解在此项出现的条件下各个类别出现的概率，哪个最大，就认为此待分类项属于哪个类别。  
通俗来说，就好比这么个道理，你在街上看到一个黑人，我问你你猜这哥们哪里来的，你十有八九猜非洲。为什么呢？  
因为黑人中非洲人的比率最高，当然人家也可能是美洲人或亚洲人，但在没有其它可用信息下，我们会选择条件概率最大的类别，  
这就是朴素贝叶斯的思想基础。

贝叶斯定理（英语：Bayes' theorem）是[概率论](https://zh.wikipedia.org/wiki/%E6%A6%82%E7%8E%87%E8%AB%96" \o "概率论)中的一个[定理](https://zh.wikipedia.org/wiki/%E5%AE%9A%E7%90%86" \o "定理)，描述在已知一些条件下，某[事件](https://zh.wikipedia.org/wiki/%E4%BA%8B%E4%BB%B6_(%E6%A6%82%E7%8E%87%E8%AE%BA)" \o "事件 (概率论))的发生几率。比如，如果已知某癌症与寿命有关，使用贝叶斯定理则可以通过得知某人年龄，来更加准确地计算出他罹患癌症的几率

朴素贝叶斯分类器：在[机器学习](https://zh.wikipedia.org/wiki/%E6%9C%BA%E5%99%A8%E5%AD%A6%E4%B9%A0" \o "机器学习)中，朴素贝叶斯分类器是一系列以假设特征之间强（朴素）[独立](https://zh.wikipedia.org/wiki/%E7%B5%B1%E8%A8%88%E7%8D%A8%E7%AB%8B%E6%80%A7" \o "统计独立性)下运用[贝叶斯定理](https://zh.wikipedia.org/wiki/%E8%B4%9D%E5%8F%B6%E6%96%AF%E5%AE%9A%E7%90%86" \o "贝叶斯定理)为基础的简单[概率分类器](https://zh.wikipedia.org/w/index.php?title=%E6%A6%82%E7%8E%87%E5%88%86%E7%B1%BB%E6%B3%95&action=edit&redlink=1)。

#### 朴素贝叶斯3种常用模型：

1。高斯：对特征非常敏感，比如mnist例子种已经使用了Z-Score规范化，这里如果再使用高斯朴素贝叶斯的话，会发现识别率会非常低，可能会不到80%  
2。多项式  
3。伯努利

SVM（支持向量机）  
from sklearn import svm  
  
KNN(k-nearest-neighbor) (邻近算法)  
from sklearn.neighbors import KNeighborsClassifier  
  
Adaboost(自适应增强)  
from sklearn.ensemble import  AdaBoostClassifier  
AdaBoost方法的自适应在于：前一个分类器分错的样本会被用来训练下一个分类器。 AdaBoost方法对于噪声数据和异常数据很敏感。  
  
XGBoost  
from xgboost import XGBClassifier  
Xgboost是Boosting算法的其中一种，Boosting算法的思想是将许多弱分类器集成在一起，形成一个强分类器。  
因为Xgboost是一种提升树模型，所以它是将许多树模型集成在一起，形成一个很强的分类器。而所用到的树模型则是CART回归树模型。  
  
TPOT  
from tpot import TPOTClassifier  
  
keras  
import keras  
keras是基于python的深度学习库，是用python编写的高级神经网络API

同一个精确度，模型是可以选择多个的

### Google Cloud AutoML

Graphical user interface, application

Description automatically generated

### 华为AI训练集群

9月18日，华为发布计算战略，目前世界上训练速度最快的AI集群：Atlas 900

ResNet-50，Atlas 900 只用59.8秒就完成了整个训练

### 华为ModelArts

华为天才少年第一名研究：NAS 神经网络自动搜索

## 数据清洗

数据分析师80%的时间都花在数据清洗上

### 数据质量的准则：完全合一

完整性：数据是否存在空值，字段是否完善，是都有漏掉  
全面性：观察某一列的全部数值及特征值，是都存在单位、字段名与数值不匹配  
合法性: 数据类型、内容、大小合法  
唯一性：数据是否有重复

### python中如何进行数据清洗

DataFrame是Python中Pandas库中的一种数据结构，它类似excel，是一种二维表  
1. 缺失值：删除、采用均值、采用高频数据  
 dataframe函数 df['Age'].fillna(df['Age'].mean(), inplace=True)  
2. 空行  
 df.dropna(how='all',inplace=True)  
3. 列数据的单位不统一  
 比如说需要统一将磅（lbs）转化为千克（kgs）  
 # 获取 weight 数据列中单位为 lbs 的数据  
 rows\_with\_lbs = df['weight'].str.contains('lbs').fillna(False)  
 # 将 lbs转换为 kgs, 2.2lbs=1kgs  
 for i,lbs\_row in df[rows\_with\_lbs].iterrows():  
 # 截取从头开始到倒数第三个字符之前，即去掉lbs。  
 weight = int(float(lbs\_row['weight'][:-3])/2.2)（[:-3]）  
 df.at[i,'weight'] = '{}kgs'.format(weight)   
4. 非ASCII码  
 #删除非ASCII码字符

df['name'].replace({r'[^\x00-\x7F]+':''}, regex=True, inplace=True)

5. 一列有多个参数  
 # 切分名字，删除源数据列  
 df[['first\_name','last\_name']] = df['name'].str.split(expand=True)  
 df.drop('name', axis=1, inplace=True)  
 默认采用的空格进行分割，相当于df['name'].str.split(' ', expand=True)

6. 重复数据

如果存在重复记录，就使用 Pandas 提供的 drop\_duplicates() 来删除重复数据

# 删除重复数据行

df.drop\_duplicates(['first\_name','last\_name'],inplace=True)

### 工具 使用Python对Titanic数据进行清洗

Titanic <https://github.com/cystanford/Titanic_Data>

泰坦尼克海难是著名的十大灾难之一，究竟多少人遇难，各方统计的结果不一。

数据集格式为csv，一共有两个文件：

train.csv：训练集，包含特征信息，分类结果（存活与否）；

test.csv：测试集，只包含特征信息

字段 描述  
PassengerId 乘客编号  
Survived 是否幸存  
Pclas，有些特征标注的英文，船票等级  
Name 乘客姓名  
Sex 乘客性别  
Sibsp 亲戚数量（兄妹、配偶数）  
Parch 亲戚数量（父母、子女数）  
Ticket 船票号码  
Fare 船票价格  
Cabin 船舱  
Embarked 登陆港口

#数据探索

print(data.info())  
print(data.describe())  
#查看离散数据类型的分布  
print(data.describe(include=['O'])) o就是object类型，不是数字类型  
#显示数据大小  
print(dada.shape)  
#查看前五条数据  
print(data.head(5))  
#查看后五条数据  
print(data.tail(5))

缺失值处理：

怎么看确实值

1.train\_data.isnull().describe()

2.train\_data.isnull().sum()

# 使用平均年龄来填充年龄中的nan值  
train\_data['Age'].fillna(train\_data['Age'].mean(), inplace=True)  
test\_data['Age'].fillna(test\_data['Age'].mean(),inplace=True)  
# 使用票价的均值填充票价中的nan值  
train\_data['Fare'].fillna(train\_data['Fare'].mean(), inplace=True)  
test\_data['Fare'].fillna(test\_data['Fare'].mean(),inplace=True)  
  
#查看Embarked字段中取值的分布数量  
print(data['Embarked'].value\_counts())  
'''  
[5 rows x 12 columns]  
S 644  
C 168  
Q 77  
'''  
# 使用登录最多的港口来填充登录港口的nan值  
train\_data['Embarked'].fillna('S', inplace=True)  
test\_data['Embarked'].fillna('S',inplace=True)  
  
# 特征选择  
features = ['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked']

Cabin这个字段缺失太多了，不要了  
train\_features = train\_data[features]

编码的方式

1. get\_dummy()  
2.label encoded

3. dvec=DictVectorizer(sparse=False)  
DictVectorizer将非数字化的值数字化，比如将Embarked='S' 设为1 将Embarked != 'S' 设为0  
train\_features=dvec.fit\_transform(train\_features.to\_dict(orient='record'))  
print(dvec.feature\_names\_)

# 使用LR分类器

clf = LogisticRegression(max\_iter=100, verbose=True, random\_state=33, tol=1e-4)

clf.fit(train\_features, train\_labels)

# 获取特征重要性（更容易幸存的特征）

feature\_importance = clf.coef\_[0]

feature\_importance = 100.0 \* (feature\_importance / feature\_importance.max())

print(feature\_importance)

show\_feature\_importance(dvec.feature\_names\_, feature\_importance)

def show\_feature\_importance(feature\_list, feature\_importance):

# 设定阈值

fi\_threshold = 20

# 重要特征的下标

important\_idx = np.where(feature\_importance > fi\_threshold)[0]

# 特征名称

important\_features = [feature\_list[i] for i in important\_idx]

# 重要特征

sorted\_idx = np.argsort(feature\_importance[important\_idx])[::-1]

# 可视化

pos = np.arange(sorted\_idx.shape[0]) + 0.5

plt.subplot(1, 2, 2)

plt.title('Feature Importance')

plt.barh(pos, feature\_importance[important\_idx][sorted\_idx[::-1]], color='r',align='center')

plt.yticks(pos, [important\_features[i] for i in sorted\_idx[::-1]])

plt.xlabel('Relative Importance')

plt.draw()

plt.show()

## 思考与练习

### Thinking1 如何使用用户标签来指导业务（如何提升业务）

根据用户标签可以获取用户的一些特征，用户画像，精准营销，获取客户；根据我们的item特征，与用户特征做一些关联，推荐用户感兴趣的  
item,粘住客户；最后，也要做一些预测，比如说有一些客户可能会流失，及时采取一些措施。

### Thinking2 如果给你一堆用户数据，没有打标签。你该如何处理（如何打标签）

1。可以让机器去打标签，先跑出几类，再去看下这每个类别到底代表什么含义，通过后面再做具体的分析，给类别一个有含义的标识。  
2。看当下热门的标签是否符合

### Thinking3 准确率和精确率有何不同（评估指标）

精确率考虑的是预测结果为正的情况下的准确率  
准确率考虑的维度更多，不光考虑预测结果为正的，还有预测结果为负的时候，只有预测结果是对的都会考虑。

### Thinking4 如果你使用大众点评，想要给某个餐厅打标签。这时系统可以自动提示一些标签，你会如何设计（标签推荐）

1. 直接使用系统最热门的标签  
2.使用这家餐厅里相应物品上最热门的标签  
3.使用用户经常使用的标签  
将2和3进行加权融合，生成最终的标签推荐结果

### Thinking5 使用TPOT等AutoML工具，有怎样的好处和不足？

好处：

1. 可以解决

1）特征选择（基于树模型、基于方差、基于F-值的百分比）

2）模型选择  
 3）数据预处理（二值化、聚类、降维、标准化、正则化等）  
2. 处理小规模数据非常快

3.支持的分类器和回归器很多，有：

支持的分类器主要有贝叶斯、决策树、集成树、SVM、KNN、线性模型、xgboost

支持的回归器主要有决策树、集成树、线性模型、xgboost

4.可以实现通过代码来写代码，可以说是机器学习的机器学习，可以通过export()方法把训练过程导出为形式为sklearn pipeline的.py文件

不足：  
1.目前只能做有监督学习

2. 可以做特征模型选择，但不包括数据清洗  
3.处理小规模数据非常快，但大规模数据非常慢。可以先抽样小部分，使用TPOT

通过这节课的学习，感觉打开了新世界的大门，老师由简单的例子入手，逐步带我们深入了解什么是用户画像，及具体应用场景，用了什么样的模型和算法，感觉收获颇丰！内容真的很多，可以无限深入探究下去，这里就先整理到这里，以后在好好补充！

老师辛苦，这里谢谢老师！

最后，继续加油！希望以后找到大厂推荐算法工程师的工作！