**在VUE-PYTHON-SHELL文件夹下**

**机器学习概论：**

机器学习常用数据：csv文件，而非数据库中存储的，不仅因为mysql性能瓶颈，而且还有格式的问题。numpy速度快是因为它释放了GIL锁(GIL全称Global Interpreter Lock),这个锁是为了多线程的安全性，但是也降低了效率，想解也解不掉了，而numpy没有加，所以它是真正的多线程，cpython有，而jpython下没有。

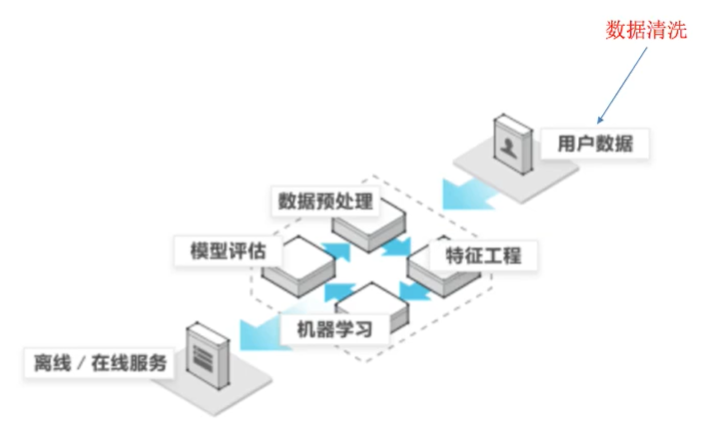
数据集结构构成：特征值+目标值

特征值对目标值有影响。

有些数据集没有目标值。

对特征的处理：pandas(读取和基本格式处理)，sklearn（特征处理）

比如对颜色的分类分为黄，白，黑，而计算机不好对这些字符串进行计算，所以需要将它们转换为数字

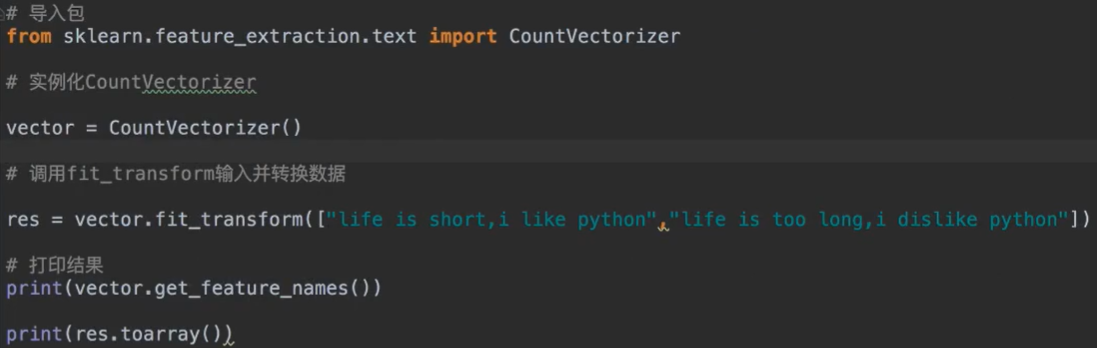


**特征工程：（SKLEARN）**

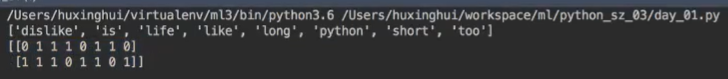
1.特征抽取

将文本，字符串等类型转换为数字类型。

使用sklearn中的方法进行特征抽取，对文本等数据进行特征值化，以方便计算机更好地理解数据。



返回的是两个数字类型的数组

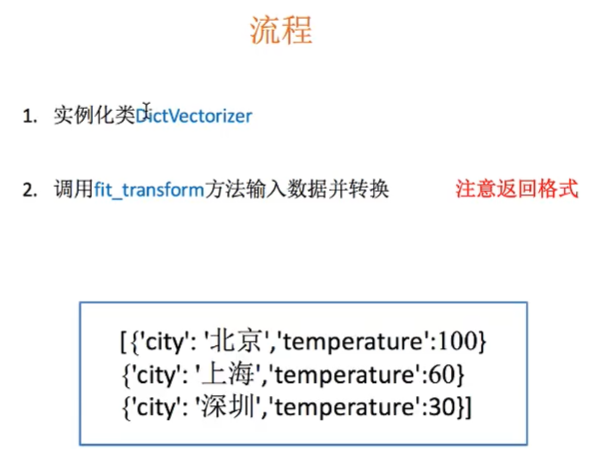


sklearn的特征抽取API是sklearn.feature\_extraction

1. 字典特征抽取

sklearn.feature\_extraction.DictVectorizer





代码：

from sklearn.feature\_extraction import DictVectorizer

def dictvec():

#实例化

dict=DictVectorizer()

#调用函数

data = dict.fit\_transform([{‘city’:’北京’,’temp’:10},{‘city’:’上海’,’temp’:12},{}])

print(data)

return None

if \_\_name\_\_==”main”:

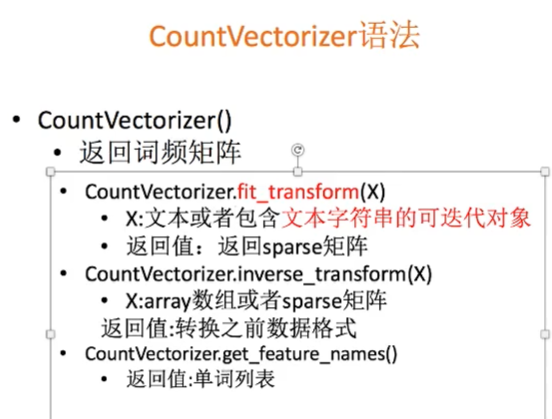
dictvec()

字典数据抽取：把字典中一些类别数据，分别转换为特征（数字0,1）

如果为数组形式类别，先转换为字典数据再按上面转换为特征

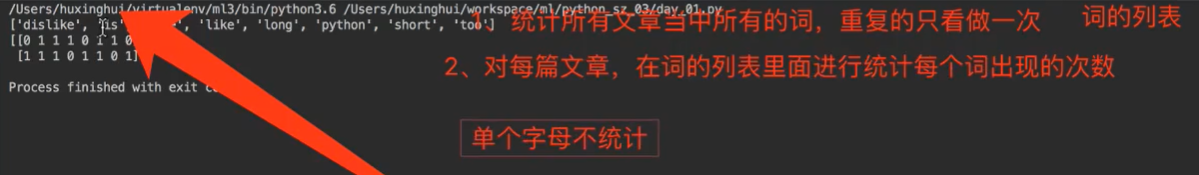
1. 文本特征抽取

sklearn.feature\_extraction.text.CountVectorizer



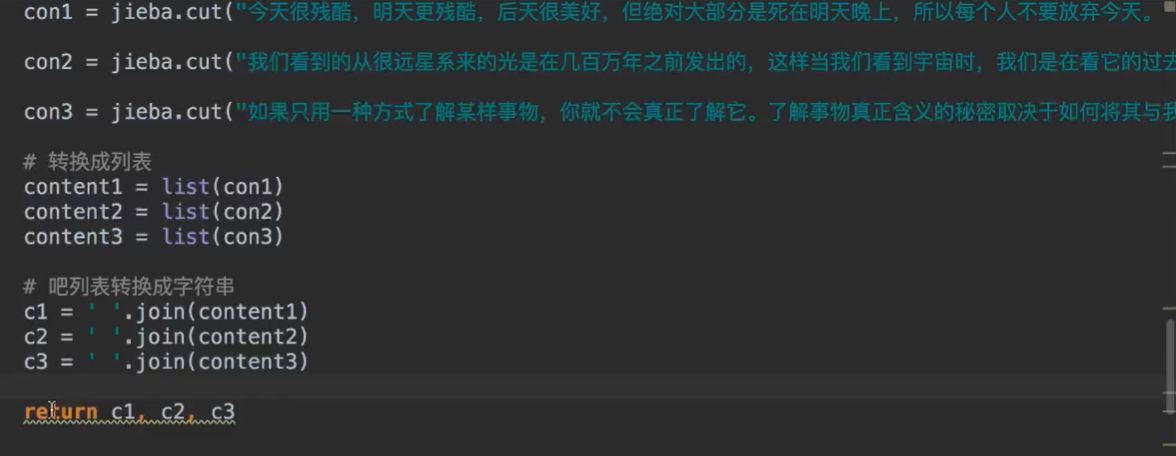
后续和上面一样。



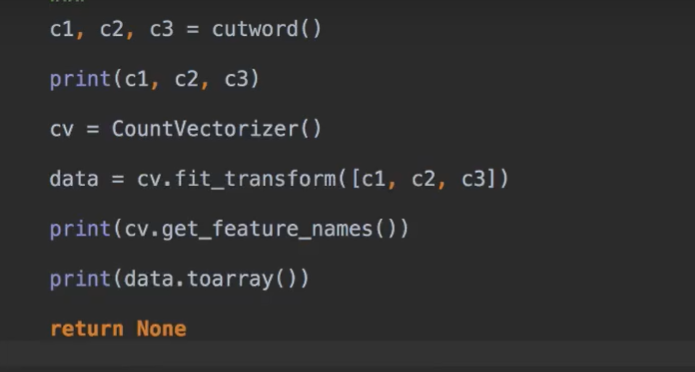


而如果对中文进行文本抽取，需要先进行分词操作。因为中文没有默认的空格分隔

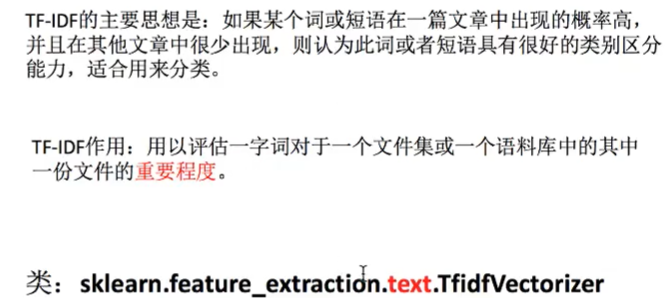
使用jieba库，pip install jieba，以下为cutword()函数

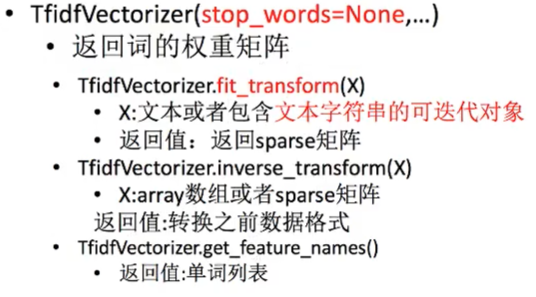


调用切字函数，得到三个划分好的中文字符串



文本特征分析常用：tf idf （词频 ，逆文档频率）



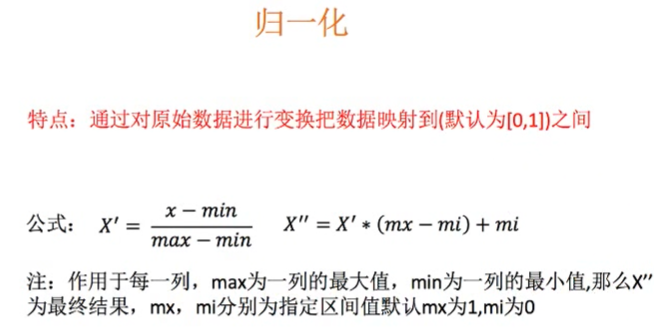


2.特征处理

数值型数据：标准缩放1.归一化，2.标准化3.缺失值

类别型数据：one-hot编码

时间类型：时间切分

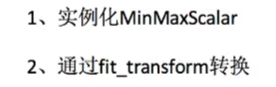


对该列（即该个特征下的那列数据进行计算）

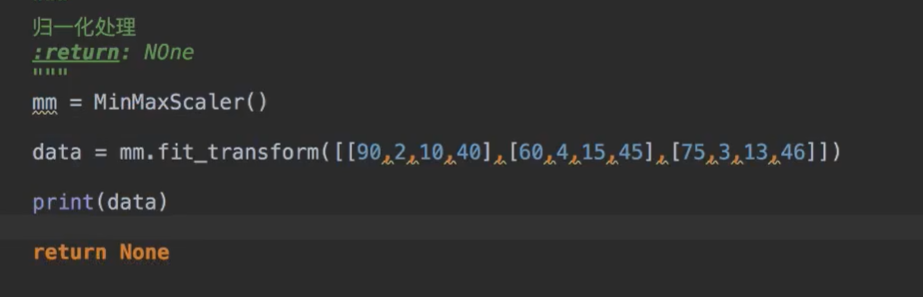
归一化API：sklearn.preprocessing.MinMaxScaler



步骤：



注意参数是二维数组



多个特征同等重要，需要归一化。因为有些特征的值本来就比较大，而其他特征值比较小，这样混合运算时肯定不对，所以将各个特征都归并至0,1之间才能让它们的重要性相同。

归一化目的：使得一个特征不会对最终结果产生较大的影响，但是易受异常点影响

标准化，不太受异常点影响，因此优先采用标准化。





**时间切分：**

df = pd.read\_csv(file\_path)

#把分开的时间字符串通过periodIndex的方法转化为pandas的时间类型

period = pd.PeriodIndex(year=df["year"],month=df["month"],day=df["day"],hour=df["hour"],freq="H")

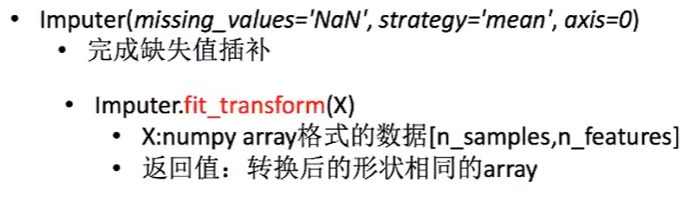
df["datetime"] = period

**缺失值插补：**

方法 ： 1.删除了，当这种缺失值很多时删除这一列

2.插补，可以通过这一列的平均值或者中位数来填充

API是sklearn.preprocessing.Imputer



在pandas里可以dropna ,fillna，这是当数据中的缺失值类型为np.nan.在sklearn中的缺失值处理也是如此，需要将所有的缺失值类型转换为np.nan才可以进行处理。

可以使用replace(“?”,np.nan)

具体操作

im=Imputer(missing\_values=’NaN’,stategy=’mean’,axis=0)#使用策略是mean-平均值填补，缺失值为nan，axis=0 表示按该列平均填充，0表示列，1表示行

data=im.fit\_transform([[1,2],[np.nan,3],[7,6]])

2.特征预处理

**降维：降低特征的个数，而不是二维数组的维的概念。是为了去除掉一些对目标结果无用的特征列，称之为降维。**

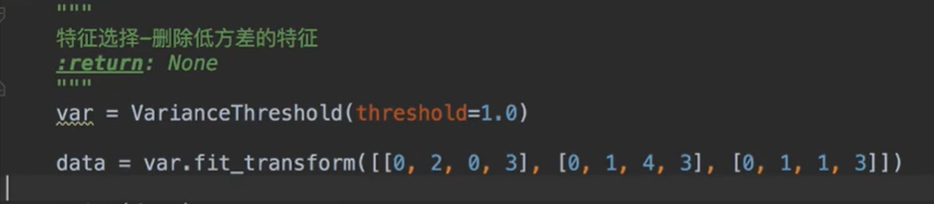
**降维方法：1.特征选择 2.主成分分析（特征几百个以上的时候可以用）**

1. 特征选择的原因：（1）冗余，特征之间有相关性（2）部分特征对结果有影响

方法有过滤式，嵌入式（正则化，决策树），包裹式，神经网络等。

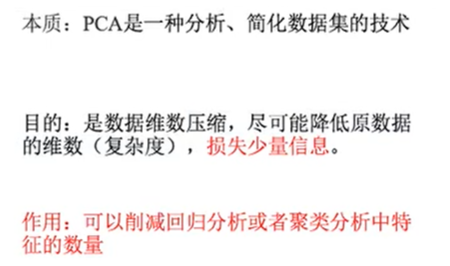
过滤式 是通过方差的大小来考虑所有样本在这个特征上的数据情况（说的啥？删除所有低方差数据） API是sklearn.feature\_selection.VarianceThreshold

这才是精华：删除某列低方差数据，设置一个方差的阈值，因为方差低，说明数据都差不多，不好，应该删掉这个特征列，当方差为0时，说明数据都相同。

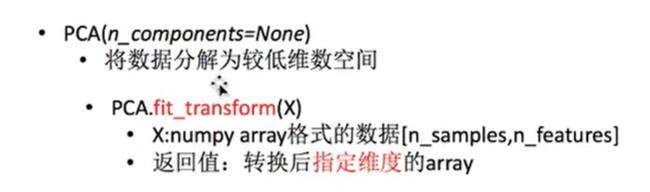


1. 主成分分析 API sklearn.decomposition

PCA使用条件:特征数量上百时，首先考虑需要用PCA来简化数据（减少特征，数据也要减少）。



即可以降低维数，同样可以表示原来维数所表示的信息，损耗不大。



n\_components最好填小数0-1,表示损耗后保留的精度（0.9-0.95即可），如果填整数，则表示的是最后留下的特征数，不好确定，故不采用。



实例1：用户购买分类



合并这几张表步骤：首先prior和products可以通过product\_id合并，从而有了order\_id,aisle\_id，然后合并后的表再与orders有相同的order\_id，故可再合并，最后再通过aisle\_id与aisles表合并

代码:

import pandas as pd

from sklearn.decomposition import PCA

#读取表数据

prior=pd.read\_csv(“./data/order\_products\_prior.csv”)

products=pd.read\_csv(“./data/products.csv”)

orders=pd.read\_csv(“./data/orders.csv”)

aisles=pd.read\_csv(“./data/aisles.csv”)

#合并表（用户-物品类别）

mg1=pd.merge(prior,products,on=[‘product\_id’,’product\_id’])#根据product\_id键合并

mg2=pd.merge(mg1,orders,on=[‘order\_id’,’order\_id’])

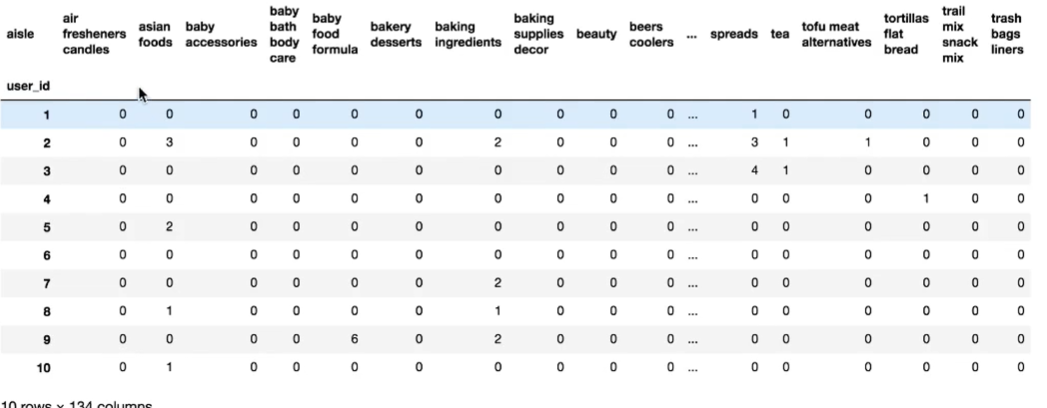
mt=pd.merge(mg2,aisles,on=[‘aisle\_id’,’aisle\_id’])

#print(mt.head(10))

#要建立一个行为用户id，列为物品种类

#使用交叉表来划分

cross=pd.crosstab(mt[‘user\_id’],mt[‘aisle’])#设置行为user\_id中的所有值,列为aisle中的所有值，而表中数据是对对应的进行统计



#0信息有点多，需要降维，使用PCA主成分分析

pca=PCA(n\_components=0.9)

data=pca.fit\_transform(cross)

**转换器 transform**

fit,transform的作用是依赖于调用它们的对象是什么，如

s=StandardScaler()

s.fit\_transform([[],[])) #此处为标准化，因为s是标准化对象

注意还可以使用另一个数组fit来的平均值标准差来转换一个数组

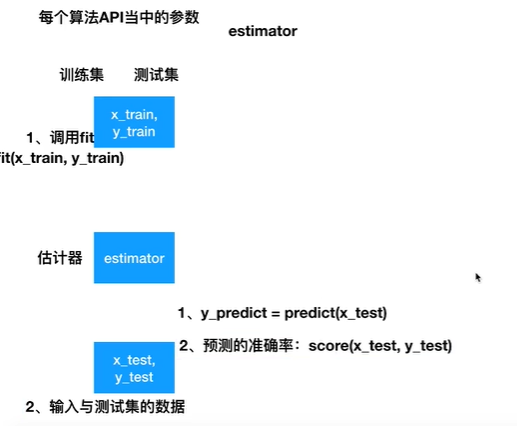
如s.fit([[1,2,3],[4,5,6]]) #这里求出平均值和标准差保存在s对象中，在下面transform应用来标准化

s.transform([[3,3,3],[4,4,4]])

**估计器**

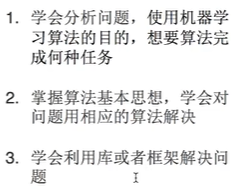
是算法API，也就是一些分类回归算法





**机器学习具体算法学习**

算法是核心，数据，计算是基础



入门基础：

离散型数据是区间不可分，连续型是区间内可分。

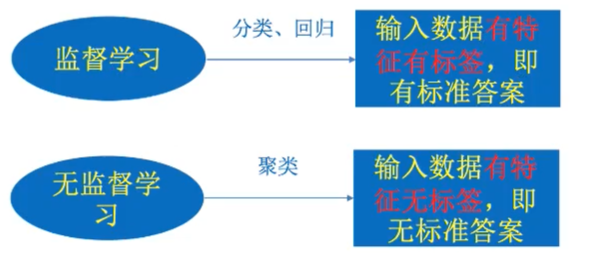
数据类型可作为选择不同算法应用的依据。

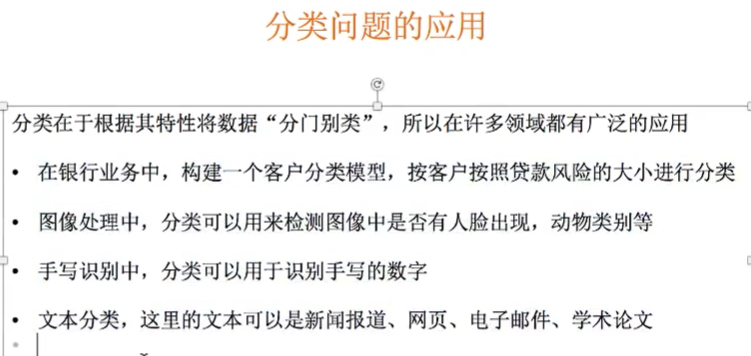


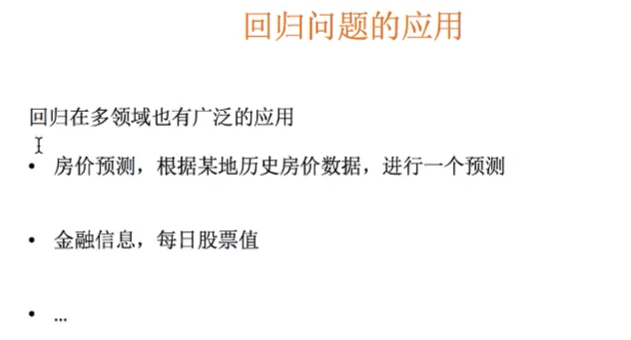
监督学习：有特征值+目标值，而无监督学习没有目标值。

分类：对应离散型数据

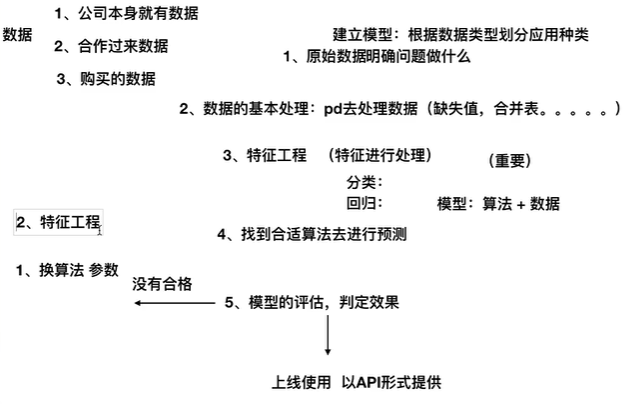
回归：对应连续型数据



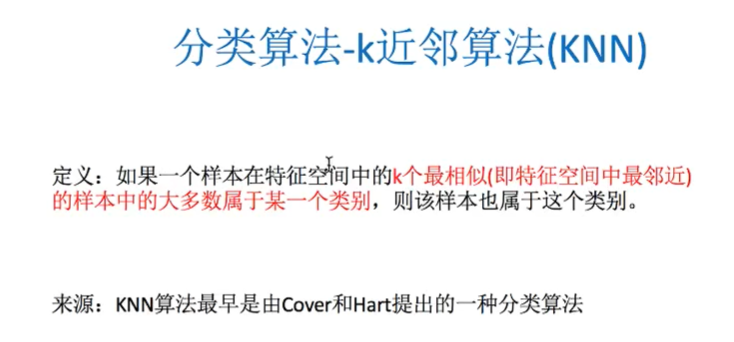




流程：

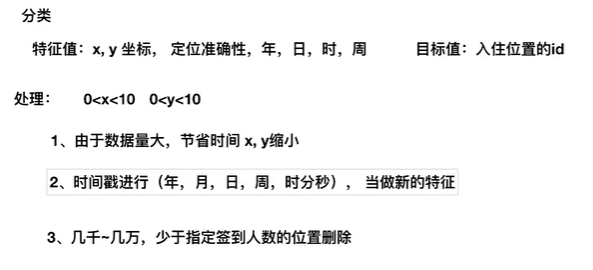


分类算法-k近邻：



k近邻算法需要标准化，因为要求欧式距离。

K近邻实战：

1. facebook预测社区签到：
2. 

data=pd.read\_csv(“./data/train.csv”)

print(data,head(10))

#处理数据

#首先缩小数据，不然太多了

data=data.query(“x>1.0&x<1.25&y>2.5&y<2.75”)

#再处理时间数据，把时间戳变成时间（如2016-10-19 12:12:12）

time\_value=pd.to\_datetime(data[‘time’],unit=’s’)

#构造新特征 如，年月日单独提出

#把日期格式转化为字典数据

time\_value=pd.DatetimeIndex(time\_value)

#从而可以提取出年月日

data[‘day’]=time\_value.day

data[‘hour’]=time\_value.hour

data[‘weekday]=time\_value.weekday

#把之前的时间戳列删除

data.drop([‘time’],axis=1) #按列删除，1表示列（与sklearn不同） 不知道参数是什么，可以点击进去看

#再把签到数量少于n的位置删除

place\_count = data.groupby('place\_id').count() #根据place\_id分组,通过count()统计特征非空的数目并写在每列下

tf=place\_count[place\_count.row\_id>3].reset\_index()

data=data[data[‘place\_id’].isin(tf.place\_id)]

# 取出数据当中的特征值和目标值

y = data['place\_id']

x = data.drop(['place\_id'], axis=1)#删除place\_id后剩下的为特征值，按列删除

# 进行数据的分割训练集合测试集 x特征值，y目标值，test\_size测试集的大小

#x\_train,训练集的特征值 x\_test,测试集的特征值 ,y\_train,训练集的目标值 ,y\_test,测试集的目标值。

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.25)

# 特征工程（标准化）

# std = StandardScaler()

# 对测试集和训练集的特征值进行标准化

# x\_train = std.fit\_transform(x\_train)

# x\_test = std.transform(x\_test)

# 进行算法流程

# 超参数

knn = KNeighborsClassifier()#参数默认是5

# fit，predict,score

knn.fit(x\_train, y\_train)

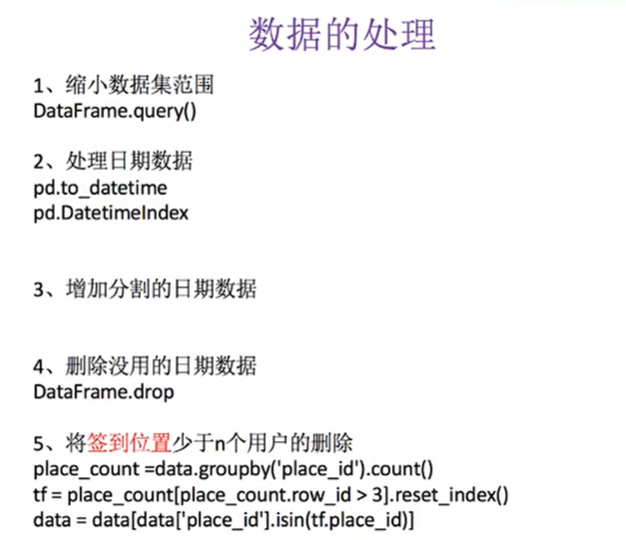
# 得出预测结果

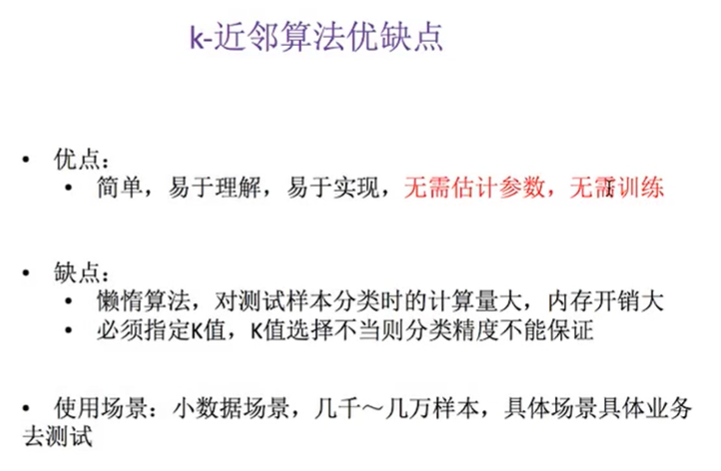
y\_predict = knn.predict(x\_test)#参数测试集的特征值

print("预测的目标签到位置为：", y\_predict)

#得出准确率

print("预测的准确率:", knn.score(x\_test, y\_test))





**subplot子图的用法**

plt.subplot(121) # 前面12代表共有一行两列，1表示现在正在绘制第一个图

**画散点图等其他图传参**

plt.scatter(data[:, 0], data[:, 1], c=label)

**数组与列表：**

python本身并没有数组类型，但是他的Numpy库中有数组类型。

Numpy数组中的所有元素的类型都必须相同的，而Python列表中的元素类型是任意的，

Numpy数组创建时，参数既可以是list，也可以是元组。例如：

>>> a=np.array((1,2,3))#参数是tuple

>>> b=np.array([6,7,8])#参数是list

>>> c=np.array([[1,2,3],[4,5,6]])#参数是二维list

除此之外，还可以使用numpy提供的其他方法创建一个数组，例如：

>>> arr1=np.arange(1,10,1)

>>> arr2=np.linspace(1,10,10)

data是二维数组，要取其第一列的数据作为一个一维数组的话就是data[:,0]

常用要将特征和目标分别输入的话就可以(data[:,0],data[:,1])

**数据预处理（数据清洗）：**

统计data里每一列是否有空值：

data.isnull().any()

统计data里每一列空值的个数：

**data.isnull().any().sum()**

但是有的时候，明明有空值却统计不出来度。

最近问我遇到的数据，空值的填充是null，这个需要转化一下才可以用上面的函数。

data = data.replace('null',**np.NaN**) #意思是只把NaN作为空值

然后你再继续用data.isnull().any()，ata.isnull().any().sum()就没问答题了。

如果这么做，你的问题还没解决，查看你内的缺失值的填充是什么，用np.NaN替代。、

另外的方法容：

np.any(np.isnan(data))

np.all(np.isfinite(data))

**df.isnull().sum().sort\_values(ascending=False).head(10) #进行排序并取前十个**

**#使用众数填充，mode是众数意思**

train\_df['Embarked'].fillna(train\_df['Embarked'].mode()[0], inplace = True)

**#使用平均数填充**

train\_df['Age'].fillna(train\_df['Age'].median(), inplace = True)

**#删除数据**

drop\_column = ['Cabin']train\_df.drop(drop\_column, axis=1, inplace = True)

**Pandas读取数据：**

df=pd.read\_csv(file,header=None)

df.head(10)

**X=df.iloc[0:100,[0,2]].values #注意iloc取地址也可以取一个范围，意思是取前100行（第0列和第二列）**

plt.scatter(X[0:50,0],X[0:50,1],color=’red’,maker=’o’,label=’setosa’) **#以第0列作为横轴，第1列作为纵轴。**

plt.xlabel(‘x’)

plt.ylabel(‘y’)

plt.legend(loc=’upper left’)

plt.show()

meshgrid函数理解：

[X,Y] = meshgrid(x,y) 将向量x和y定义的区域转换成矩阵X和Y,其中矩阵X的行向量是向量x的简单复制，而矩阵Y的列向量是向量y的简单复制.

假设x是长度为m的向量，y是长度为n的向量，则最终生成的矩阵X和Y的维度都是nm （注意不是mn）。即如果x为5，y为3，即他们产生的矩阵为3\*5，即3行5列的矩阵。且均为对应向量的复制，而非组合。

np.arrange(start,end,step) #产生从start到end，步长为step的数组

**定义模型**

使用一些模型都有相似的接口方法如下：

# 拟合模型

model.fit(X\_train, y\_train)

# 模型预测

model.predict(X\_test)

# 获得这个模型的参数

model.get\_params()

# 为模型进行打分

model.score(data\_X, data\_y) # 线性回归：R square； 分类问题： acc

**保存模型**

sklearn自带方法joblib

from sklearn.externals import joblib

# 保存模型

joblib.dump(model, 'model.pickle')

#载入模型

model = joblib.load('model.pickle')

**预测及评分**

# 预测

y\_predict = knnClf.predict(x\_test)

print("score on the testdata:",knnClf.score(x\_test,y\_test))

结构化数据 ：如表格

非结构化数据：如图片（识别猫）

**逻辑回归：**

Y=wtx+b 任务就是尽可能求出w，b

求得的y再用sigmoid函数（1/(1+e-z)）转化为0到1之间，从而作为概率。

为了优化w，b，需要一个代价函数

**机器学习：**

通过训练集求出f(x,q)=q1+q2x+q3x^3+…中的参数q

，通过误差函数L(q)=∑(f(x,q)-yi)^2 注:yi为实际值

尽可能降低误差

求出来参数q后，再把未知的（即测试集）的已知参数代入，再来计算出目标结果

**卷积神经网络：**

通常包含以下几种层：

卷积层（Convolutional layer），卷积神经网路中每层卷积层由若干卷积单元组成，每个卷积单元的参数都是通过反向传播算法优化得到的。卷积运算的目的是提取输入的不同特征，第一层卷积层可能只能提取一些低级的特征如边缘、线条和角等层级，更多层的网络能从低级特征中迭代提取更复杂的特征。

线性整流层（Rectified Linear Units layer, ReLU layer），这一层神经的活性化函数（Activation function）使用线性整流（Rectified Linear Units, ReLU）。

池化层（Pooling layer），通常在卷积层之后会得到维度很大的特征，将特征切成几个区域，取其最大值或平均值，得到新的、维度较小的特征。

全连接层（ Fully-Connected layer）, 把所有局部特征结合变成全局特征，用来计算最后每一类的得分。

**TensorFlow**

深度学习&机器学习框架

机器学习： 算法：神经网络（简单），回归

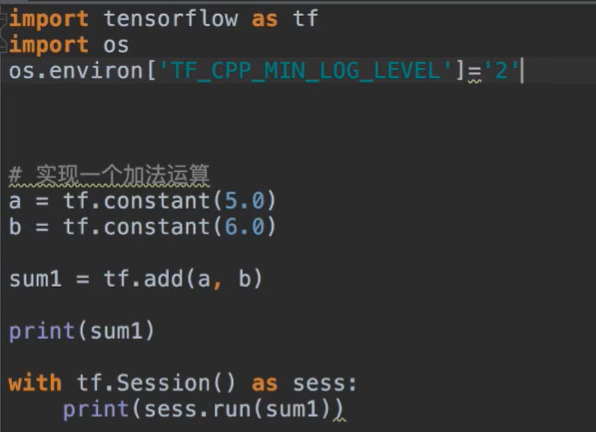
领域：

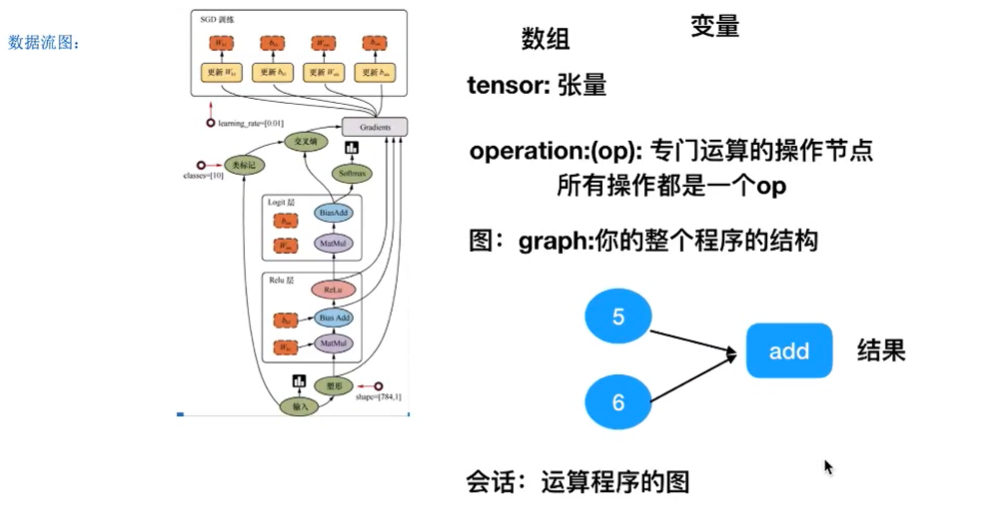
深度学习： 算法：神经网络（深度）

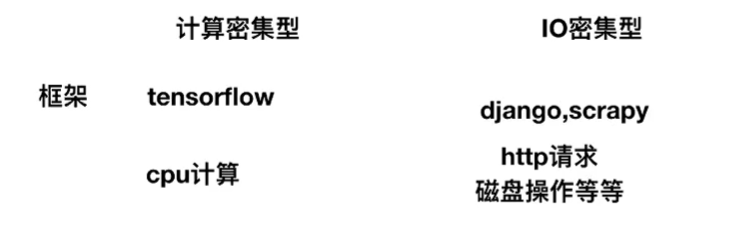
图像领域：卷积神经网络

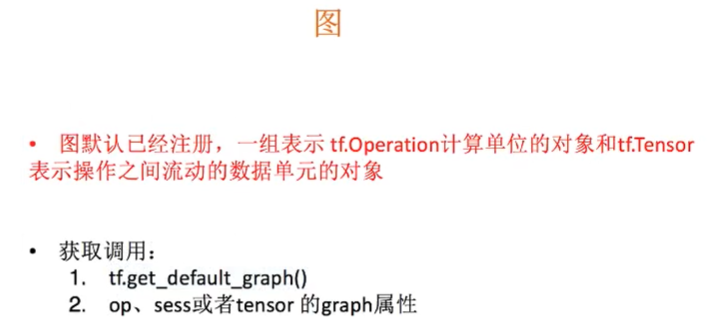
自然语言处理领域：循环神经网络

TensorFlow实现加法：（第二、三行是关闭警告，当使用pip安装时出现）









with 很重要