

Python与金融数据挖掘(14)

文欣秀

wenxinxiu@ecust.edu.cn

Pandas常用统计函数

函数	描述
<code>df.mean()</code>	计算样本数据的算术平均值
<code>df.value_counts()</code>	统计频数
<code>df.describe()</code>	返回基本统计量和分位数
<code>df.corr(sr)</code>	df与sr的相关系数
<code>df.count()</code> 、 <code>df.sum()</code>	统计每列(或行)数据的个数或总和
<code>df.max()</code> 、 <code>df.min()</code>	最大值和最小值
<code>df.idxmax()</code> 、 <code>df.idxmin()</code>	最大值、最小值对应的索引
<code>df.qantile()</code>	计算给定的四分位数
<code>df.var()</code> 、 <code>df.std()</code>	计算方差、标准差
<code>df.mode()</code>	计算众数
<code>df.cov()</code>	计算协方差矩阵

数据清洗案例

```
import pandas as pd  
data=pd.read_excel("info.xlsx","Group2",index_col=0)  
data1=data.dropna(thresh=6) # 每行至少6个非空值才保留  
print(data1)
```

	A	B	C	D	E	F	G	H
1	序号	性别	年龄	身高	体重	省份	成绩	月生活费
2	1	male	20	170	70	LiaoNing		800
3	2	male	22	180	71	GuangXi	77	1300
4	3	male		180	62	FuJian	57	1000
5	4	male	20	177	72	LiaoNing	79	900
6	5	male	20	172		ShanDong	91	
7	6	male	20	179	75	YunNan	92	950
8								
9	7	female	21	166	53	LiaoNing	80	1200
10	8	female	20	162	47	AnHui	78	1000
11	9	female	20	162	47	AnHui	78	1000
12	10	male	120	169	76	HeiLongJiang	88	1100

	性别	年龄	身高	体重	省份	成绩	月生活费
序号							
1.0	male	20.0	170.0	70.0	LiaoNing	NaN	800.0
2.0	male	22.0	180.0	71.0	GuangXi	77.0	1300.0
3.0	male	NaN	180.0	62.0	FuJian	57.0	1000.0
4.0	male	20.0	177.0	72.0	LiaoNing	79.0	900.0
6.0	male	20.0	179.0	75.0	YunNan	92.0	950.0
7.0	female	21.0	166.0	53.0	LiaoNing	80.0	1200.0
8.0	female	20.0	162.0	47.0	AnHui	78.0	1000.0
9.0	female	20.0	162.0	47.0	AnHui	78.0	1000.0
10.0	male	120.0	169.0	76.0	HeiLongJiang	88.0	1100.0

数据清洗案例

```
import pandas as pd
```

```
data=pd.read_excel("info.xlsx","Group2",index_col=0)
```

```
data1=data.fillna(method='ffill') #在列方向上以前一个值替换
```

```
print(data1)
```

	A	B	C	D	E	F	G	H
1	序号	性别	年龄	身高	体重	省份	成绩	月生活费
2	1	male	20	170	70	LiaoNing		800
3	2	male	22	180	71	GuangXi	77	1300
4	3	male		180	62	FuJian	57	1000
5	4	male	20	177	72	LiaoNing	79	900
6	5	male	20	172		ShanDong	91	
7	6	male	20	179	75	YunNan	92	950
8								
9	7	female	21	166	53	LiaoNing	80	1200
10	8	female	20	162	47	AnHui	78	1000
11	9	female	20	162	47	AnHui	78	1000
12	10	male	120	169	76	HeiLongJiang	88	1100

	性别	年龄	身高	体重	省份	成绩	月生活费
序号							
1.0	male	20.0	170.0	70.0	LiaoNing	NaN	800.0
2.0	male	22.0	180.0	71.0	GuangXi	77.0	1300.0
3.0	male	22.0	180.0	62.0	FuJian	57.0	1000.0
4.0	male	20.0	177.0	72.0	LiaoNing	79.0	900.0
5.0	male	20.0	172.0	72.0	ShanDong	91.0	900.0
6.0	male	20.0	179.0	75.0	YunNan	92.0	950.0
NaN	male	20.0	179.0	75.0	YunNan	92.0	950.0
7.0	female	21.0	166.0	53.0	LiaoNing	80.0	1200.0
8.0	female	20.0	162.0	47.0	AnHui	78.0	1000.0
9.0	female	20.0	162.0	47.0	AnHui	78.0	1000.0
10.0	male	120.0	169.0	76.0	HeiLongJiang	88.0	1100.0

提取数据存入数组

```
import pandas as pd  
data=pd.read_excel("info.xlsx","Group1")  
val=data[["身高","体重"]].values #提取数据存入数组  
print(val)
```

	A	B	C	D	E	F	G	H
1	序号	性别	年龄	身高	体重	省份	成绩	月生活费
2	21	female	21	165	45	Shanghai	93	1200
3	22	female	19	167	42	HuBei	89	800
4	23	male	21	169	80	GanSu	93	900
5	24	female	21	160	49	HeBei	59	1100
6	25	female	21	162	54	GanSu	68	1300
7	26	male	21	181	77	SiChuan	62	800
8	27	female	21	162	49	ShanDong	65	950
9	28	female	22	160	52	ShanXi	73	800
10	29	female	20	161	51	GuangXi	80	1250
11	30	female	20	168	52	JiangSu	98	700

```
[[165 45]  
 [167 42]  
 [169 80]  
 [160 49]  
 [162 54]  
 [181 77]  
 [162 49]  
 [160 52]  
 [161 51]  
 [168 52]]
```

scikit-learn

scikit-learn: 基于NumPy, SciPy, matplotlib, 可以实现数据预处理、**分类、回归、降维、聚类、模型选择**等常用的机器学习算法, 是数据挖掘和数据分析的一个简单有效工具。

机器学习分类: 有监督学习、无监督学习

机器学习分类

有监督学习：指数据中包括了想要预测的属性。

分类：对事物所属类别判断，类型数量已知。如判别鸟的种类、判断垃圾邮件、判断客户类别等。

电影分类问题

电影名称	打斗镜头	接吻镜头	电影类型
无问西东	1	101	爱情片
后来的我们	5	89	爱情片
前任3	12	97	爱情片
红海行动	108	5	动作片
唐人街探案	112	9	动作片
战狼2	115	8	动作片
新电影	24	67	?

常用分类算法

- KNN算法(k-近邻算法)
- 贝叶斯算法(NB)
- 支持向量机(SVM)
- ...

KNN算法

KNN算法： k近邻分类算法。基本思路是在特征空间中查找k个最相似或者距离最近的样本，然后根据k个最相似的样本对未知样本进行分类。

KNN算法基本步骤

1. 计算已知样本空间中所有点与未知样本的距离;
2. 对所有距离按升序排列;
3. 确定并选取与未知样本距离最小的 k 个样本或点;
4. 统计选取的 k 个点所属类别的出现频率;
5. 把出现频率最高的类别作为预测结果, 即未知样本所属类别。

电影分类问题

➤ 距离计算公式

$$d(A, B) = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}$$

电影名称	打斗镜头	接吻镜头	电影类型
无问西东	1	101	爱情片
后来的我们	5	89	爱情片
前任3	12	97	爱情片
红海行动	108	5	动作片
唐人街探案	112	9	动作片
战狼2	115	8	动作片
新电影	24	67	?

➤ 距离计算案例

$$d(\text{"新电影"}, \text{"无问东西"}) = \sqrt{(24 - 1)^2 + (67 - 101)^2} = 41.0$$

电影分类问题

电影名称	与未知电影的距离
无问西东	41.0
后来的我们	29.1
前任3	32.3
红海行动	104.4
唐人街探案	105.4
战狼2	108.5

若 $k=4$ ，距离最近的4部影片三部为“爱情片”
因此，新电影预测为“爱情片”

电影分类代码 (一)

```
import numpy as np  
  
import pandas as pd  
  
import matplotlib.pyplot as plt  
  
from sklearn import neighbors  
  
#读取数据集  
  
data=pd.read_csv("movie.csv", encoding="gb2312")
```

电影分类代码 (二)

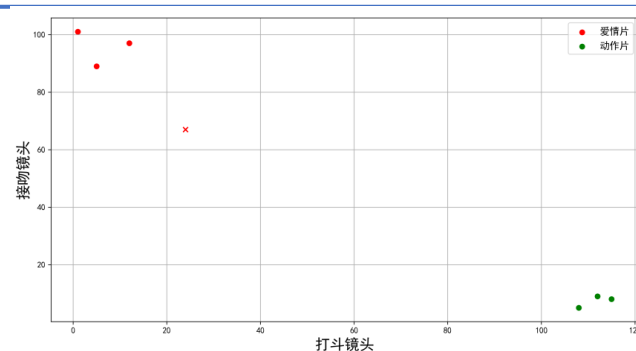
```
#Matplotlib识别中文字体
```

```
plt.rcParams['font.sans-serif']=['SimHei']
```

```
plt.figure(figsize=(20,8))
```

```
plt.scatter(data[data['电影类型']=='爱情片']['打斗镜头'], \  
            data[data['电影类型']=='爱情片']['接吻镜头'], \  
            color='r',marker='o',label='爱情片')
```

```
plt.scatter(data[data['电影类型']=='动作片']['打斗镜头'], \  
            data[data['电影类型']=='动作片']['接吻镜头'], \  
            color='g',marker='o',label='动作片')
```



电影分类代码 (三)

```
plt.grid()
```

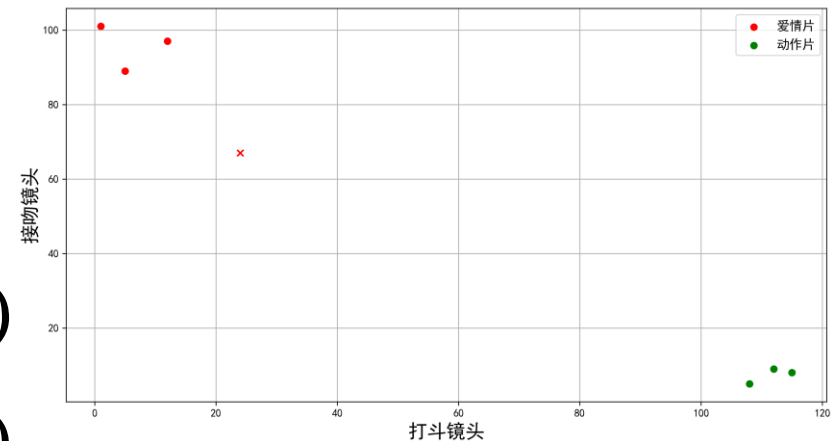
```
plt.legend(fontsize='large')
```

```
plt.xlabel('打斗镜头',fontsize=18)
```

```
plt.ylabel('接吻镜头',fontsize=18)
```

```
plt.scatter(24,67,color='r',marker='x')
```

```
plt.show()
```



电影分类代码（四）

#K近邻预测， KNeighborsClassifier为实现K近邻算法的一个类

knn=**neighbors.KNeighborsClassifier()** #/模型初始化

X=data[['打斗镜头','接吻镜头']].**values**

Y=data['电影类型'].**values**

knn.fit(X,Y) #/模型学习

print('预测类型为:', **knn.predict**([[24,67]]))#模型预测

will become False, the axis
minated, and the value None w
rue or False to avoid this wa
预测类型为: ['爱情片']

电影分类代码KNN模型

预测类型为: ['爱情片']

```
import numpy as np
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from warnings import simplefilter
simplefilter(action='ignore', category=FutureWarning) # 忽略所有警告
data=pd.read_csv('movie.csv',encoding='gb2312')
model= KNeighborsClassifier()
model.fit(data[['打斗镜头','接吻镜头']].values, data['电影类型'].values)
print('预测类型为: ',model.predict([[24,67]]))
```

思考题一

除KNN外，常用分类算法还有：

- 朴素贝叶斯算法(NB)
- 支持向量机(SVM)
- 决策树(DT)
- 逻辑回归(LR)

如何使用以上算法实现电影分类预测？

贝叶斯算法

贝叶斯分类算法：是统计学的一种分类方法，它是一类利用概率统计知识进行分类的算法。朴素贝叶斯（Naive Bayes，**NB**）是基于贝叶斯定理与特征条件独立假设的分类方法，可用于垃圾邮件分类、客户信用评估等问题。

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)} = \frac{P(A \cap B)}{P(B)}$$

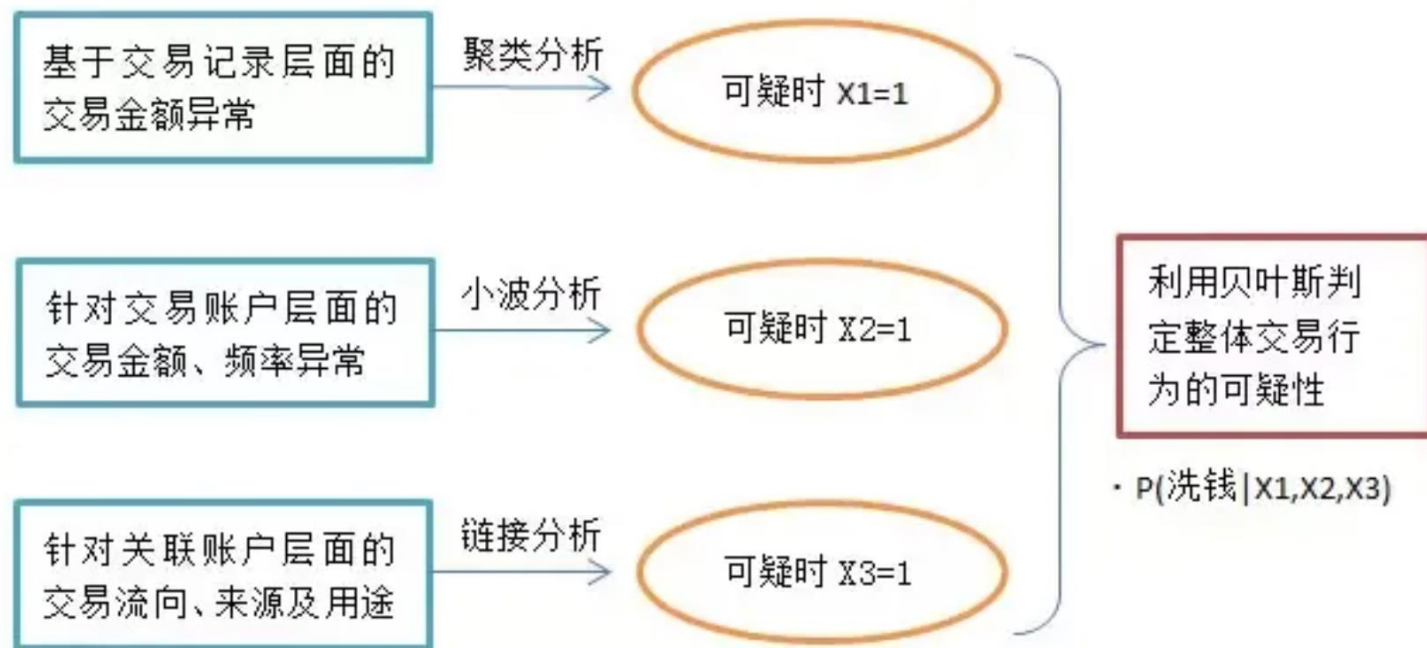
案例分析

颜值	性格	是否上进	是否嫁
帅	好	上进	嫁
不帅	好	一般	不嫁
不帅	不好	不上进	不嫁
帅	好	一般	嫁
不帅	好	上进	嫁
帅	不好	一般	不嫁
帅	好	不上进	嫁
不帅	不好	上进	不嫁
帅	不好	上进	嫁
不帅	好	不上进	不嫁

知乎 @Yolanda

男生“帅 & 性格不好 & 不上进”，预测女生嫁还是不嫁？

案例分析



电影分类代码贝叶斯模型

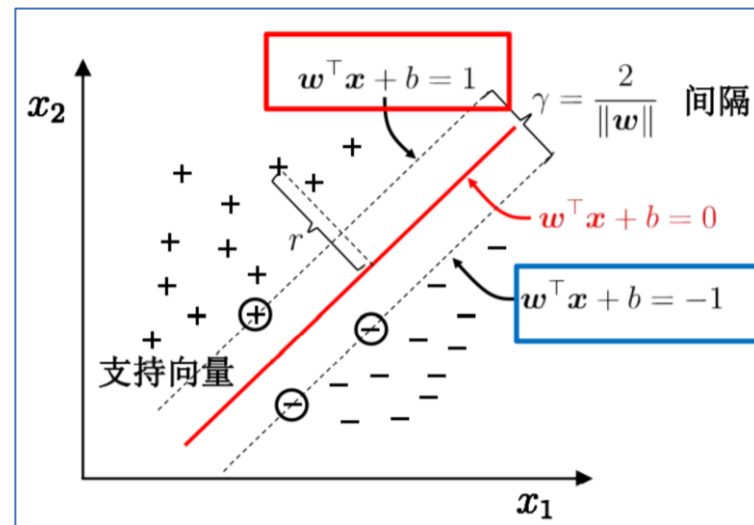
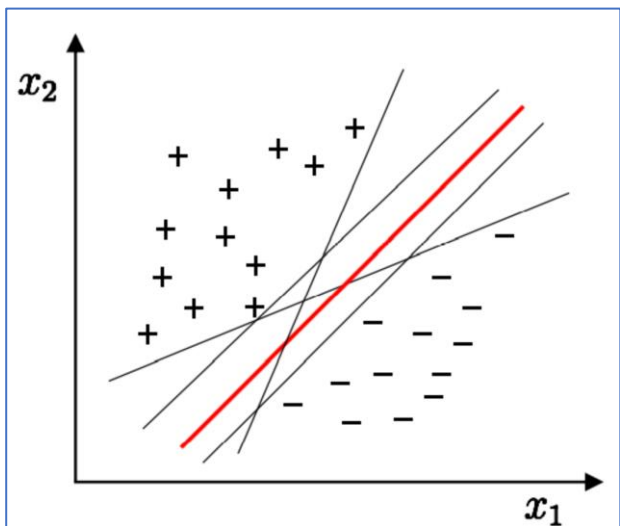
```
import numpy as np
import pandas as pd
from sklearn.naive_bayes import GaussianNB
data=pd.read_csv('movie.csv',encoding='gb2312')
model=GaussianNB()
model.fit(data[['打斗镜头','接吻镜头']].values, data['电影类型'].values)
print('预测类型为: ',model.predict([[24,67]]))
```

预测类型为: ['爱情片']

>>>

支持向量机(SVM)

支持向量机 (support vector machines, SVM) :是一种二分类模型, 它的基本模型是定义在特征空间上的间隔最大的线性分类器。SVM还包括核函数, 这使它成为实质上的非线性分类器。



电影分类代码支持向量机模型

```
import numpy as np
```

```
import pandas as pd
```

```
from sklearn.svm import SVC
```

```
data=pd.read_csv('movie.csv',encoding='gb2312')
```

```
model=SVC()
```

```
model.fit(data[['打斗镜头','接吻镜头']].values,data['电影类型'].values)
```

```
print('预测类型为: ',model.predict([[24,67]]))
```

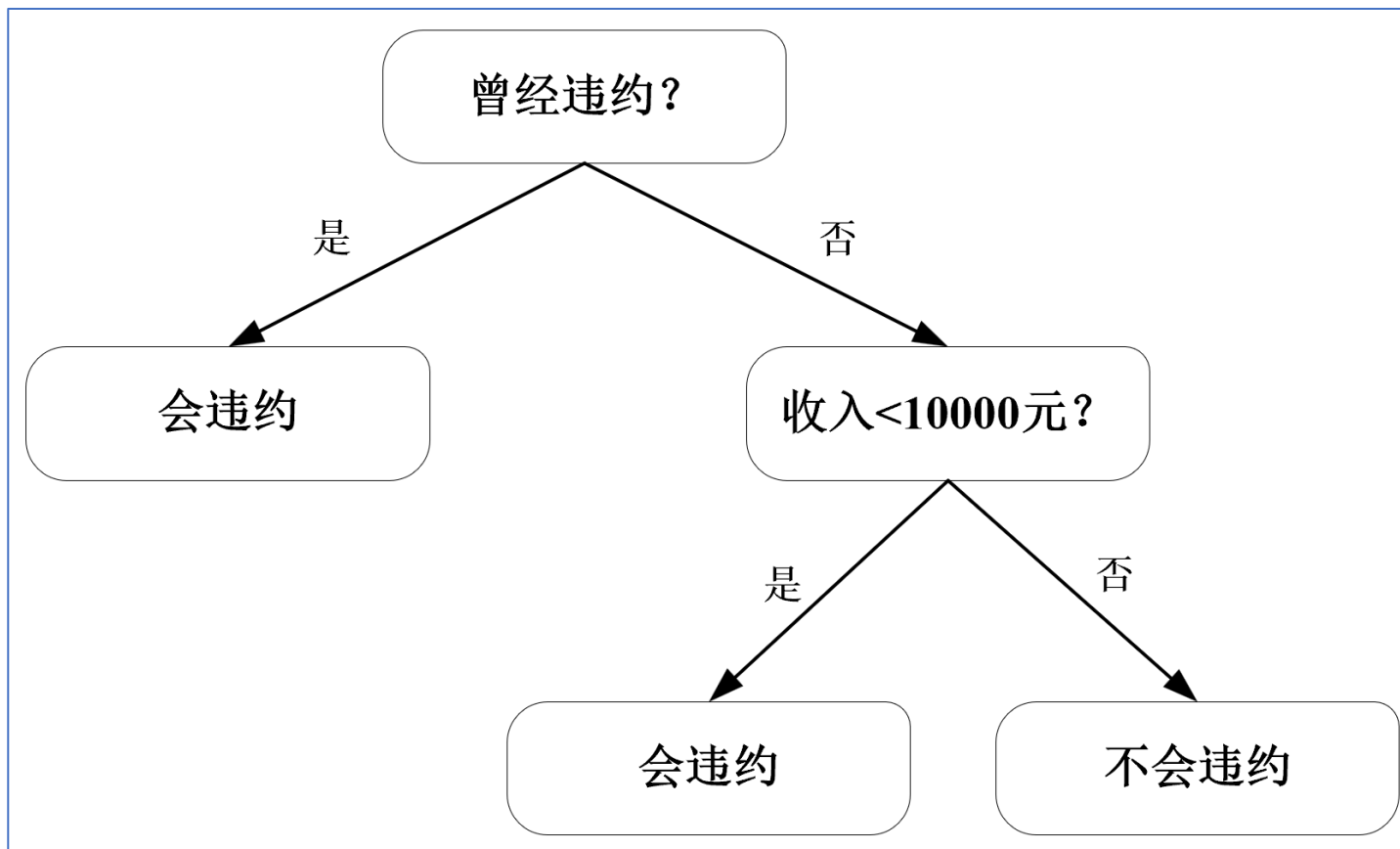
预测类型为: ['爱情片']

>>>

决策树(DT)

决策树(Decision Tree, DT)：是数据挖掘技术中的一种重要的分类与回归方法,它是一种以**树结构**(包括二叉树和多叉树)形式来表达的预测分析模型。其每个非叶节点表示一个特征属性上的测试，每个分支代表这个特征属性在某个值域上的输出，而每个叶节点存放一个类别。

案例分析



电影分类代码决策树模型

```
import numpy as np
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
data=pd.read_csv('movie.csv',encoding='gb2312')
model=DecisionTreeClassifier()
model.fit(data[['打斗镜头','接吻镜头']].values,data['电影类型'].values)
print('预测类型为: ',model.predict([[24,67]]))
```

预测类型为: ['爱情片']

>>>

逻辑回归

逻辑回归算法(Logistic Regression): 逻辑回归也被称为广义线性回归模型，它与线性回归模型的形式基本上相同，最大的区别就在于它们的因变量不同，如果是连续的，就是多重线性回归；如果是二项分布，就是Logistic回归，是一种分类算法。

电影分类代码逻辑回归模型

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
data=pd.read_csv('movie.csv',encoding='gb2312')
model=LogisticRegression()
model.fit(data[['打斗镜头','接吻镜头']].values,data['电影类型'].values)
print('预测类型为: ',model.predict([[24,67]]))
```

预测类型为: ['爱情片']

>>>

思考题二



如何使用KNN算法实现胖瘦预测？

思考题二

	A	B	C	D
1	性别	身高	体重	类型
2	1	180	75	正常
3	1	180	85	过重
4	1	180	90	过重
5	1	180	100	肥胖
6	1	175	90	肥胖
7	1	175	80	过重
8	1	175	60	正常
9	1	175	55	偏瘦
10	1	170	60	正常
11	1	170	70	过重
12	1	170	80	过重
13	1	185	90	过重
14	1	185	75	正常

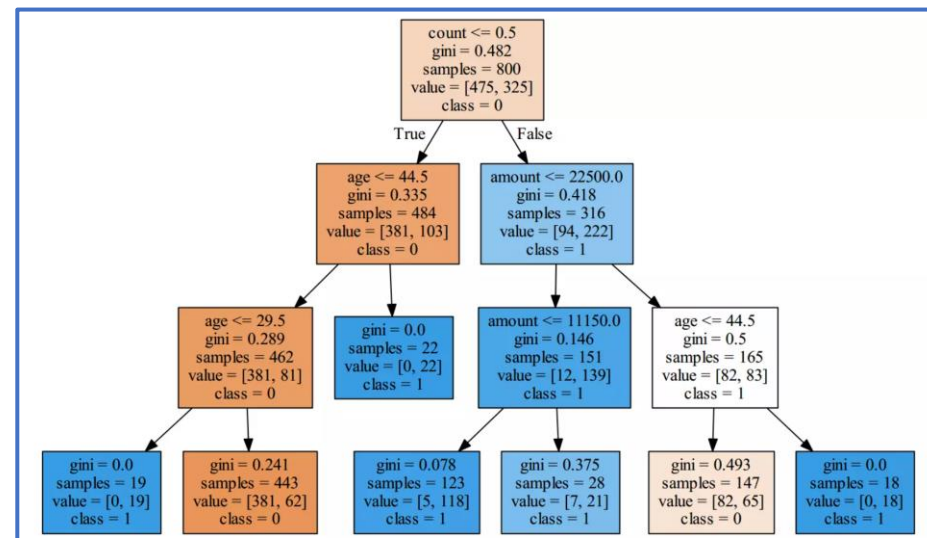
请输入您的身高: 173
请输入您的体重: 80
预测类型为: ['过重']
>>>

预测胖瘦代码**KNN**模型

```
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from warnings import simplefilter
simplefilter(action='ignore', category=FutureWarning) # 忽略所有警告
data=pd.read_csv('info.csv',encoding='gb2312')
model= KNeighborsClassifier()
model.fit(data[['性别','身高','体重']].values, data['类型'].values)
height=int(input("请输入您的身高: "))
weight=int(input("请输入您的体重: "))
print('预测类型为: ',model.predict([[1,height,weight]]))
```

思考题三

	A	B	C	D	E	F
1	收入	年龄	性别	历史授信额度	历史违约次数	是否违约
2	503999	46	1	0	1	1
3	452766	36	0	13583	0	1
4	100000	33	1	0	1	1
5	100000	25	0	0	1	1
6	258000	35	1	0	0	1
7	933333	31	0	28000	3	1
8	665000	40	1	5000	1	1
9	291332	38	0	0	0	1
10	259000	45	1	0	1	1
11	3076666	39	1	71000	2	1
12	695000	40	1	5000	0	1
13	600000	35	1	18000	3	1
14	440000	36	1	0	2	1
15	511999	35	0	0	2	1



决策树(DT)

基尼系数(gini)：用于计算一个系统中的失序现象。即系统的混乱程度。基尼系数越高，系统混乱程度越高，建立决策树模型的目的就是通过合适的分类来降低系统的混乱程度。

客户违约预测模型搭建

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
df = pd.read_excel('客户信息及违约表现.xlsx')
# 1.提取特征变量和目标变量
X = df.drop(columns='是否违约') #删除指定列
y = df['是否违约']
```

客户违约预测模型搭建

2.划分训练集和测试集，按8:2来划分

```
X_train, X_test, y_train, y_test = train_test_split(X, y,  
test_size=0.2, random_state=1) #每次划分的数据都一样
```

3.模型训练及搭建

```
clf = DecisionTreeClassifier(max_depth=3) #树最大深度为3  
clf = clf.fit(X_train, y_train)
```

1.直接预测是否违约

`print(y_pred)` #0为不违约， 1为预测会违约

38

客户违约预测模型搭建

# # 二、模型预测及评估		预测值	实际值
	0	0	0
	1	0	0
#创建一个空DataFrame, 将预测值和	2	0	0
	3	0	1
a = pd.DataFrame() #	4	0	1

a['预测值'] = list(y_pred)	195	0	0
	196	0	0
a['实际值'] = list(y_test)	197	0	0
	198	1	0
print(a)	199	1	1

客户违约预测模型搭建

查看模型预测准确度

```
from sklearn.metrics import accuracy_score
```

```
score = accuracy_score(y_pred, y_test)
```

```
print("{:.3f}".format(score))
```

0.825

分类决策树模型本质上预测的并不是准确的0或1的分类，
而是预测属于某一分类的频率。

客户违约预测模型搭建

2. 预测不违约&违约概率

```
y_pred_proba = clf.predict_proba(X_test)
```

```
print("不违约概率    违约概率")
```

```
print(y_pred_proba)
```

```
print(y_pred_proba[:,1]) #仅看违约
```

不违约概率	违约概率
[0.86004515	0.13995485]
[0.86004515	0.13995485]
[0.86004515	0.13995485]
[0.86004515	0.13995485]
[0.55782313	0.44217687]

第一行数据不违约概率大，因此预测为“不违约”

思考

商业实战中一般不会以准确度作为模型的评估标准，

那么，模型的评估标准是什么呢？

scikit-learn

有监督学习：指数据中包括了想要预测的属性。

分类：对事物所属类别判断，类型数量已知。如判别鸟的种类、判断垃圾邮件等。

回归：预测目标是连续变量，如根据父母身高预测孩子身高，根据企业财务指标预测收益等。

广告费用与销量问题

问题描述： 已知某公司近几年在电视、微博、微信的广告投入以及销售量情况，输入今年6月份的广告投入，预测公司的销售量。

广告费用与销量问题

advertising.csv - Excel

文件 开始 插入 页面布局 公式 数据 审阅 视图 加载项 百度网盘 告 共享

A1

	A	B	C	D	E	F
1		TV	Weibo	WeChat	Sales	
2	1	230.1	37.8	69.2	22.1	
3	2	44.5	39.3	45.1	10.4	
4	3	17.2	45.9	69.3	9.3	
5	4	151.5	41.3	58.5	18.5	
6	5	180.8	10.8	58.4	12.9	
7	6	8.7	48.9	75	7.2	
8	7	57.5	32.8	23.5	11.8	
9	8	120.2	19.6	11.6	13.2	
10	9	8.6	2.1	1	4.8	
11	10	199.8	2.6	21.2	10.6	
12	11	66.1	5.8	24.2	8.6	
13	12	214.7	24	4	17.4	
14	13	23.8	35.1	65.9	9.2	
15	14	97.5	7.6	7.2	9.7	
16	15	204.1	32.9	46	19	
17	16	105.1	17.7	52.0	22.1	

advertising

就绪

100%

广告费用与销量线性回归模型（一）

#1.从文件中读入数据

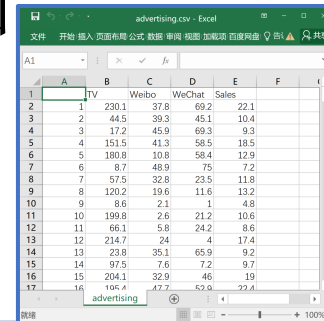
```
import pandas as pd
```

```
data = pd.read_csv('advertising.csv', index_col = 0)
```

#2.绘制自变量与目标变量之间的散点图

```
import matplotlib.pyplot as plt
```

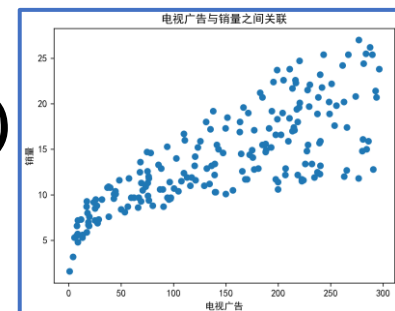
```
plt.rcParams['font.family']='SimHei'
```



	A	B	C	D	E	F
	TV	Weibo	WeChat	Sales		
1	1	230.1	37.8	69.2	22.1	
2	2	44.5	39.3	45.1	10.4	
3	3	17.2	45.9	69.3	9.3	
4	4	151.5	41.3	58.5	18.5	
5	5	180.8	10.8	58.4	12.9	
6	6	8.7	48.9	75	7.2	
7	7	57.5	32.8	23.5	11.8	
8	8	120.2	19.6	11.6	13.2	
9	9	8.6	2.1	1	4.8	
10	10	199.8	2.6	21.2	10.6	
11	11	66.1	5.8	24.2	8.6	
12	12	214.7	24	4	17.4	
13	13	23.8	35.1	65.9	9.2	
14	14	97.5	7.6	7.2	9.7	
15	15	204.1	32.9	46	19	
16	16	106.4	47.7	52.8	22.4	

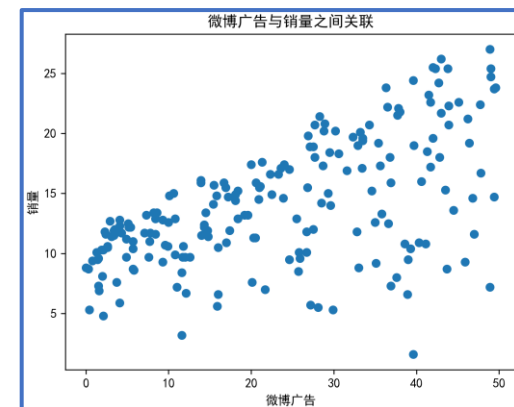
广告费用与销量线性回归模型（二）

```
#电视广告与销量之间的关联  
plt. scatter(data['TV'],data['Sales'])  
plt. xlabel("电视广告")  
plt. ylabel("销量")  
plt. title('电视广告与销量之间关联')  
plt. show()
```



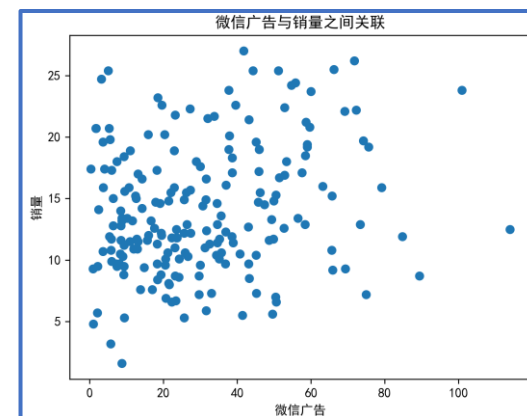
广告费用与销量线性回归模型 (三)

```
#微博广告与销量之间的关联  
plt.scatter(data['Weibo'],data['Sales'])  
plt.xlabel("微博广告")  
plt.ylabel("销量")  
plt.title('微博广告与销量之间关联')  
plt.show()
```



广告费用与销量线性回归模型（四）

```
#微信广告与销量之间的关联  
plt.scatter(data['WeChat'],data['Sales'])  
plt.xlabel("微信广告")  
plt.ylabel("销量")  
plt.title('微信广告与销量之间关联')  
plt.show()
```



广告费用与销量线性回归模型（五）

#3. 建立3个自变量与目标变量的线性回归模型

```
X = data.iloc[:,0:3].values.astype(float)
```

```
y = data.iloc[:,3].values.astype(float)
```

```
from sklearn.linear_model import LinearRegression
```

```
model = LinearRegression() #初始化模型
```

```
model.fit(X, y) #模型学习
```

广告费用与销量线性回归模型 (六)

#4. 加载预测数据

```
import numpy as np
```

```
new_X = np. array([[130.1,87.8,69.2]])
```

```
print("6月广告投入: ",new_X)
```

```
print("预期销售: ",model. predict(new_X) )#使用模型预测
```

```
6月广告投入: [[130.1 87.8 69.2]]  
预期销售: [25.37401071]  
>>>
```

身高预测案例分析



"儿童年龄,性别(0女1男),父亲身高,母亲身高,祖父身高,祖母身高,外祖父身高,外祖母身高"

```
x = np.array([[1, 0, 180, 165, 175, 165, 170, 165],\
               [3, 0, 180, 165, 175, 165, 173, 165],\
               [4, 0, 180, 165, 175, 165, 170, 165],\
               [6, 0, 180, 165, 175, 165, 170, 165],\
               [8, 1, 180, 165, 175, 167, 170, 165],\
               [10, 0, 180, 166, 175, 165, 170, 165],\
               [11, 0, 180, 165, 175, 165, 170, 165],\
               [12, 0, 180, 165, 175, 165, 170, 165],\
               [13, 1, 180, 165, 175, 165, 170, 165],\
               [14, 0, 180, 165, 175, 165, 170, 165],\
               [17, 0, 170, 165, 175, 165, 170, 165]])
```

儿童身高, 单位: cm

```
y = np.array([60, 90, 100, 110, 130, 140,\
               150, 164, 160, 163, 168])
```

预测身高

问题描述： 一个人的身高除了随年龄变大而增长之外，在一定程度上还受到遗传和饮食以及其他因素的影响，本文代码中假定受年龄、性别、父母身高、祖父母身高和外祖父母身高共同影响，并假定大致符合线性关系。

预测身高线性回归模型（一）

```
import copy
import numpy as np
from sklearn. linear_model import LinearRegression
"""儿童年龄,性别（0女1男）,父亲身高,母亲身高,祖父身高,祖母身高,外祖父身高,外祖母身高"""
X = np.array([[1, 0, 180, 165, 175, 165, 170, 165],\
              [3, 0, 180, 165, 175, 165, 173, 165],\
              [4, 0, 180, 165, 175, 165, 170, 165],\
              [6, 0, 180, 165, 175, 165, 170, 165],\
              [8, 1, 180, 165, 175, 167, 170, 165],\
              [10, 0, 180, 166, 175, 165, 170, 165],\
              [11, 0, 180, 165, 175, 165, 170, 165],\
              [12, 0, 180, 165, 175, 165, 170, 165],\
              [13, 1, 180, 165, 175, 165, 170, 165],\
              [14, 0, 180, 165, 175, 165, 170, 165],\
              [17, 0, 170, 165, 175, 165, 170, 165]])
y = np.array([60, 90, 100, 110, 130, 140, 150, 164, 160, 163, 168]) # 儿童身高, 单位: cm
```

预测身高线性回归模型（二）

```
model = LinearRegression()
```

```
model.fit(X, y) # 拟合
```

```
age=int(input("请输入您的年龄: "))
```

```
sex=int(input("请输入您的性别(1:男,0:女): "))
```

```
father=int(input("请输入您父亲的身高(厘米): "))
```

```
mother=int(input("请输入您母亲的身高(厘米): "))
```

预测身高线性回归模型 (三)

```
g1_father=int(input("请输入您祖父的身高(厘米): "))
g1_mother=int(input("请输入您祖母的身高(厘米): "))
g2_father=int(input("请输入您外祖父的身高(厘米): "))
g2_mother=int(input("请输入外祖母的身高(厘米): "))
alist=[age,sex,father,mother,g1_father,g1_mother,g2_father,g2_mother]
xs = np. array([alist])
item = copy. deepcopy(xs)  #深复制,若item改变, 不改变xs的值
print(item, ':', model. predict(item))  # 预测
```

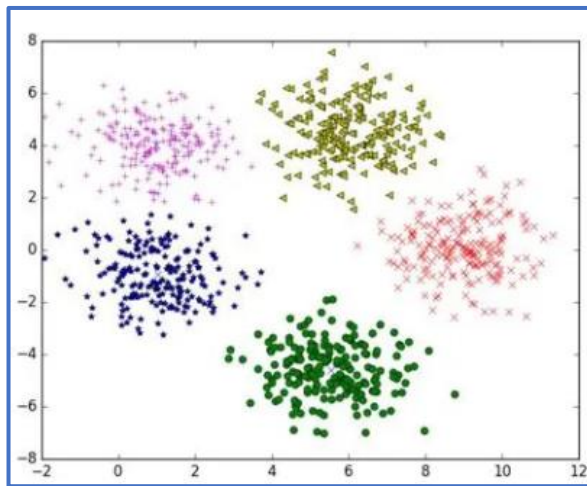

scikit-learn

无监督学习： 训练数据包括了输入向量 X 的集合，
但没有相应的目标变量。

常见问题： 数据降维、聚类问题

聚 类

聚类(Clustering Approach): 是按一定的距离或相似性系数将数据分成一系列相互区分的组，常用的经典聚类方法有**K-means**, K-medoids, isodata等。



K-Means聚类算法

- K-Means算法属于聚类分析中划分方法里较为经典的一种，由于该算法的效率 high，所以在对大规模数据进行聚类时被广泛应用。
- K-Means算法通过将样本划分 k 个簇类来实现数据聚类，该算法需要指定划分类的个数。

K- Means算法示例

例：对表中二维数据，使用k- means算法将其划分为2个簇，假设初始簇中心选为P7(4,5),P10(5,5)。

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
X	3	3	7	4	3	8	4	4	7	5
y	4	6	3	7	8	5	5	1	4	5

K- Means算法示例

- 根据题目,假设划分的两个簇分别为C1和C2,中心分别为(4,5)和(5,5),下面计算10个样本到这2个簇中心的距离,并将10个样本指派到与其最近的簇;

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
x	3	3	7	4	3	8	4	4	7	5
y	4	6	3	7	8	5	5	1	4	5

- 第一轮迭代结果如下:

属于簇C1的样本有:{P7,P1,P2,P4,P5,P8};

属于簇C2的样本有:{P10,P3,P6,P9};

- 重新计算新的簇中心,有:C1的中心为(3.5,5.167),C2的中心为(6.75,4.25);

K- Means算法示例

- 继续计算10个样本到新的簇的中心的距离，重新分配到新的簇中，第二轮迭代结果如下：

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
X	3	3	7	4	3	8	4	4	7	5
y	4	6	3	7	8	5	5	1	4	5

属于簇C1的样本有:{P1,P2,P4,P5,P7,P10};

属于簇C2的样本有:{P3,P6,P8,P9};

- 重新计算新的簇的中心，有:C1的中心为(3.67, 5.83)，C2的中心为(6.5,3.25);
- 继续计算10个样本到新的簇的中心的距离，重新分配到新的簇中，发现簇中心不再发生变化，算法终止。

K- Means聚类算法

Scikit-learn的Cluster类提供聚类分析的方法:

- 模型初始化

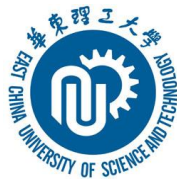
`kmeans=Kmeans(n_clusters)` #参数为簇的个数

- 模型学习

`kmeans.fit(X)` #参数为样本二维数组

鸢尾花(iris)数据集分析

Iris 数据集： 是一个经典数据集，在统计学习和机器学习领域都经常被用作示例。数据集内包含 3 类共 150 条记录，每类各 50 个数据，每条记录都有 4 项特征：**花萼长度**(Sepal_Length)、**花萼宽度**(Sepal_Width)、**花瓣长度**(Petal_Length)、**花瓣宽度**(Petal_Width)，可以通过这4个特征预测鸢尾花卉属于iris-setosa(**山鸢尾**)，iris-versicolour(**变色鸢尾**)，iris-virginica(**维吉尼鸢尾**)中的哪一品种。



谢 谢