

Python与金融数据挖掘(13)

文欣秀

wenxinxiu@ecust.edu.cn

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. 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

Pandas常用统计函数

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

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]]
```

Python应用领域

文本分析：Jieba、Nltk...

科学计算：Numpy、SciPy...

数据分析：Pandas、Matplotlib...

机器学习：Scikit-Learn、Keras...

深度学习：Pytorch、Mindspore...

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。

```
>>> pip install scikit-learn
```

机器学习分类



机器学习分类

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。如判别客户是否违规、判断垃圾邮件、判断电影类别等。

	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	100000	33	1	0	1	1
3	100000	25	0	0	1	1
4	200000	47	0	6000	2	1
5	200000	36	1	6000	1	1
6	200000	41	0	6000	1	1
7	200000	41	0	12000	0	1
8	200000	26	1	12000	0	1
9	200000	43	0	12000	0	0
10	200000	40	0	30000	0	0

电影分类问题

电影名称	打斗镜头	接吻镜头	电影类型
无问东西	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

- K近邻算法(K-Nearest Neighbor, **KNN**)
- 贝叶斯算法(Naive Bayes, **NB**)
- 支持向量机(Support Vector Machine, **SVM**)
- 决策树(Decision tree, **DT**)
- 逻辑回归(Logistic Regression, **LR**)

如何使用以上算法实现电影分类预测？

KNN算法

K近邻算法: K-Nearest Neighbor (KNN)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, 然后根据这**K**个样本的类别对未知样本进行分类。

电影分类问题

➤ 距离计算公式

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

➤ 距离计算案例

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

电影名称	打斗镜头	接吻镜头	电影类型
无问东西	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	?

电影分类问题

电影名称	与未知电影的距离
无问东西	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4部影片有3部为“爱情片”，预测新电影为“爱情片”

KNN算法基本步骤

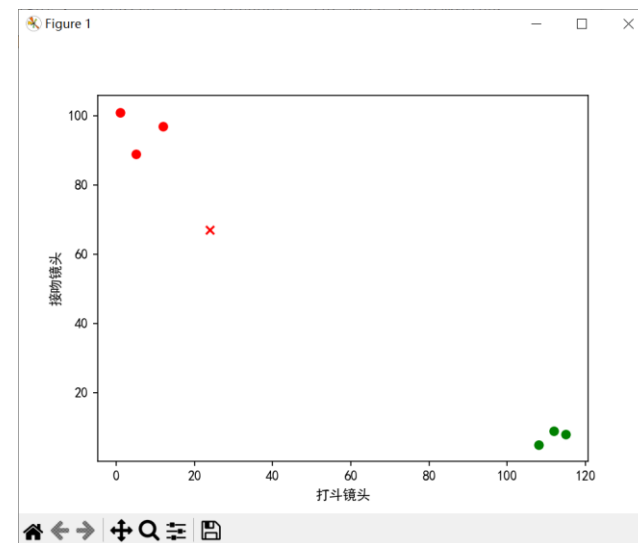
1. 计算已知样本空间中所有点与未知样本的距离;
2. 对所有距离按升序排列;
3. 确定并选取与未知样本距离最小的 k 个样本或点;
4. 统计选取的 k 个点所属类别的出现频率;
5. 把出现**频率最高**的类别作为预测结果, 即未知样本所属类别。

电影分类问题KNN算法代码（一）

```
import pandas as pd
import matplotlib.pyplot as plt
#读取数据集
data=pd.read_csv("movie.csv", encoding="gb2312")
#识别中文字体
plt.rcParams['font.sans-serif']=['SimHei']
```


电影分类问题KNN算法代码（二）

```
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',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',label='动作片')
plt.legend()
plt.xlabel('打斗镜头')
plt.ylabel('接吻镜头')
plt.scatter(24,67,color='r',marker='x')
plt.show()
```



电影分类问题KNN算法代码（三）

```
from sklearn.neighbors import KNeighborsClassifier
```

```
#KNeighborsClassifier为实现K近邻算法的一个类
```

```
knn=KNeighborsClassifier() #/模型初始化
```

```
X=data[['打斗镜头','接吻镜头']].values
```

```
y=data['电影类型'].values
```

```
预测类型为: ['爱情片']
```

```
>>>
```

```
knn.fit(X, y) #模型学习
```

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

电影分类问题K近邻算法代码

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

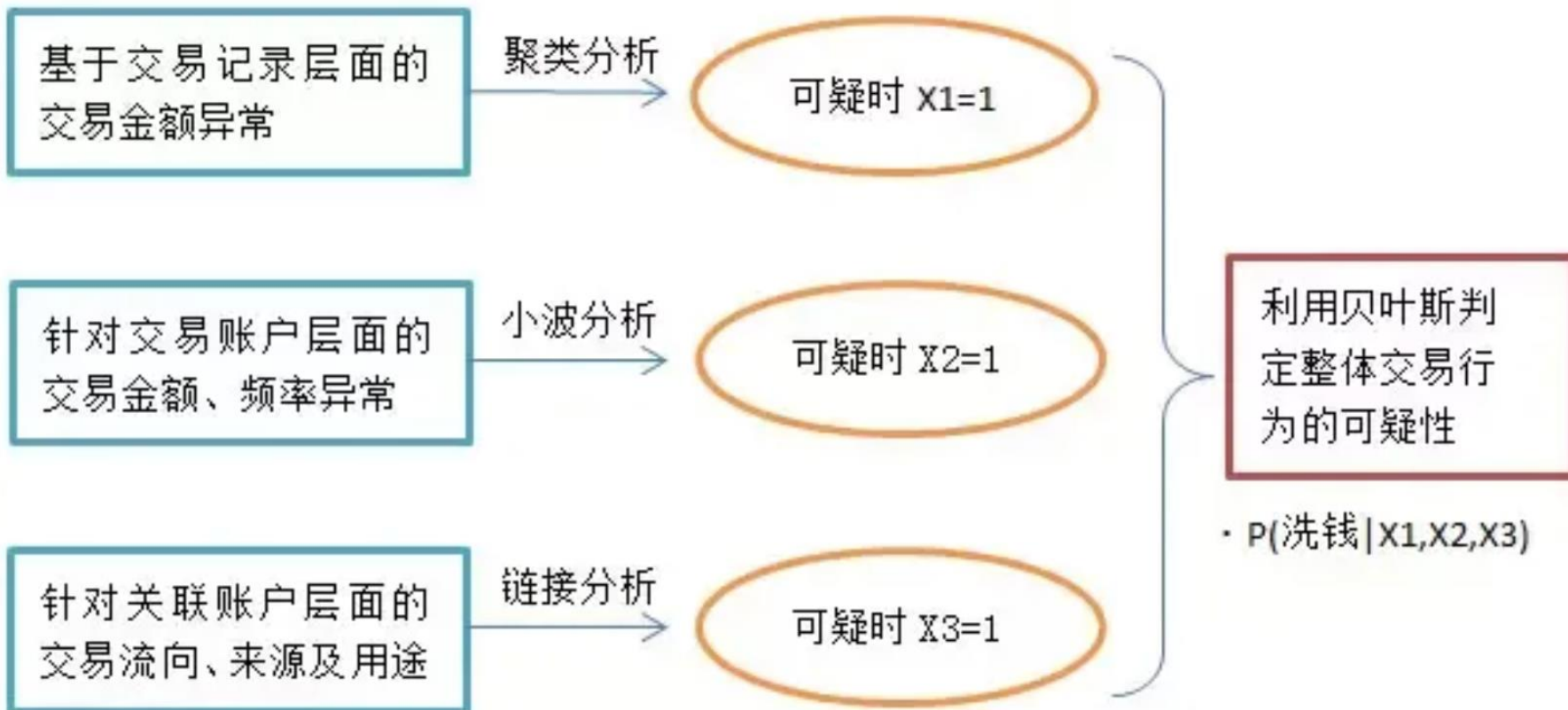
预测类型为: ['爱情片']
>>> |

思考题一

下面哪个选项不是机器学习中的分类算法？

- A. 贝叶斯
- B. 支持向量机
- C. 线性回归
- D. 决策树

案例分析



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

若一男生“帅 & 性格不好 & 不上进”，预测女生嫁还是不嫁？

案例分析

```
import pandas as pd
from sklearn.naive_bayes import GaussianNB
data=pd.read_csv('marry.csv',encoding='gb2312')
X=data[['颜值','性格','是否上进']].values
y=data['是否嫁'].values
model=GaussianNB()
model. fit(X,y)
print('预测类型为: ',model.predict([[1,0,0]]))
```

预测类型为: ['不嫁']
>>> |

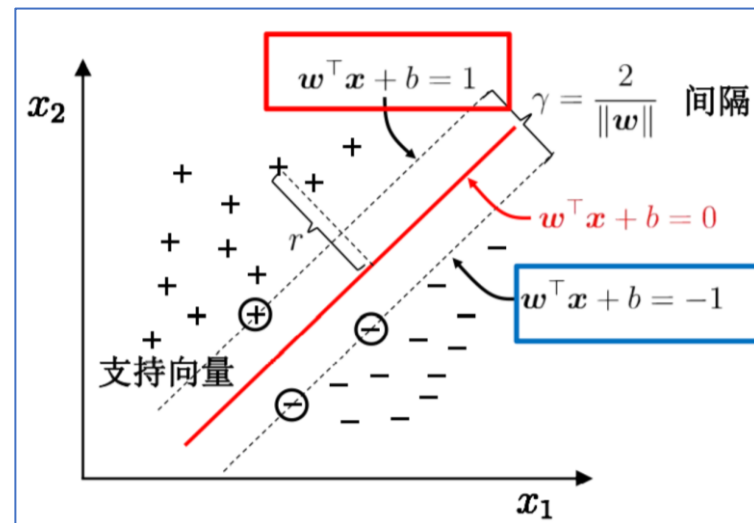
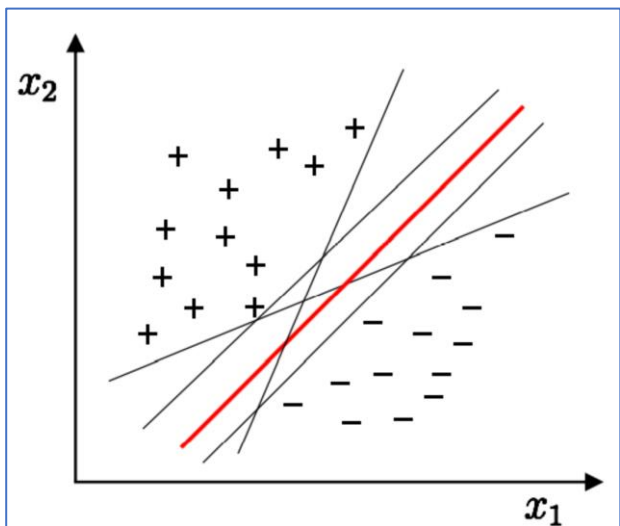
电影分类问题贝叶斯算法代码

```
import pandas as pd
from sklearn.naive_bayes import GaussianNB
data=pd.read_csv('movie.csv',encoding='gb2312')
X=data[['打斗镜头','接吻镜头']].values
y=data['电影类型'].values
model=GaussianNB()
model.fit(X,y)
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。



电影分类问题支持向量机算法代码

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

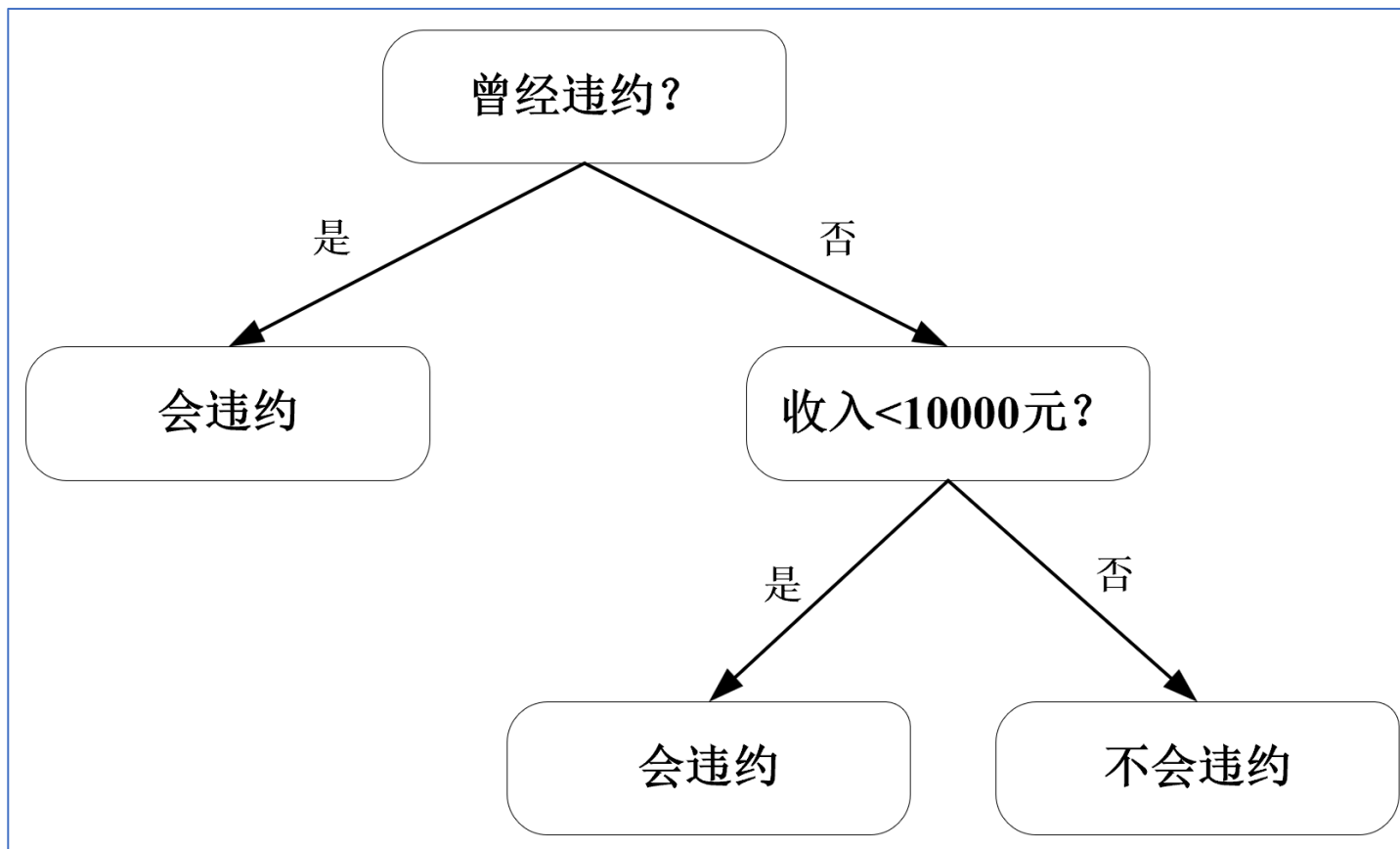
预测类型为: ['爱情片']

>>> |

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



电影分类问题决策树算法代码

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
data=pd.read_csv('movie.csv',encoding='gb2312')
X=data[['打斗镜头','接吻镜头']].values
y=data['电影类型'].values
model=DecisionTreeClassifier()
model.fit(X,y)
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。

电影分类问题逻辑回归算法代码

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

预测类型为: ['爱情片']

>>>

练习题



	A	B	C	D
1	Sex	Height	Weight	Type
2	1	180	75	normal
3	1	180	85	normal
4	1	180	90	overweight
5	1	180	100	overweight
6	1	175	90	overweight
7	1	175	80	overweight
8	1	175	65	normal
9	1	175	55	underweight
10	1	170	60	normal
11	1	170	70	normal
12	1	170	80	overweight
13	1	185	90	overweight

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

思考题二

问题：如何比较以上算法的准确率从而选择有效的算法？

方案：使用相同的数据，相同的方法来评估不同的算法，以便得到一个准确的结果。

	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	100000	33	1	0	1	1
3	100000	25	0	0	1	1
4	200000	47	0	6000	2	1
5	200000	36	1	6000	1	1
6	200000	41	0	6000	1	1
7	200000	41	0	12000	0	1
8	200000	26	1	12000	0	1
9	200000	43	0	12000	0	0
10	200000	40	0	30000	0	0

十折交叉验证

十折交叉验证 (**10-fold cross-validation**)：用来测试算法准确性。是常用的测试方法。将数据集分成10份，轮流将其中9份作为训练数据，1份作为测试数据进行试验。每次试验都会得出相应的正确率（或差错率）。10次的结果的正确率（或差错率）的平均值作为对算法精度的估计，一般还需要进行多次10折交叉验证（例如8次10折交叉验证），再求其均值，作为对算法准确性的估计。

K折交叉验证

Kfold()函数： sklearn 包中用于交叉验证的函数，一般情况将K折交叉验证用于模型调优。

函数功能： 在机器学习中，将数据集data分为训练集（**training set**）**A**和测试集（**test set**）**B**，在样本量不充足的情况下，为了充分利用数据集对算法效果进行测试，将数据集data随机分为**k**个包，每次将其中**1**个包作为**测试集**，剩下**k-1**个包作为**训练集**进行训练。

K折交叉验证

KFold(n_splits=10, shuffle=False, random_state=None)

n_splits : 整数, 表示交叉验证的折数 (即将数据集分为几份)。

shuffle : 布尔值, 表示是否要将数据打乱顺序后再进行划分, 若为True时, 每次划分的结果都不一样。

random_state : 默认为None。当shuffle为True时, random_state的值影响标签的顺序。

K折交叉验证案例分析

```
from sklearn.model_selection import KFold
import numpy as np
X = ['a','b','c','d']
kf = KFold(n_splits=4)
for train, test in kf.split(X):
    print(train, test) #打印索引
    print("-"*50)
    print(np.array(X)[train], np.array(X)[test])
    print("*"*50)
```

K折交叉验证案例分析

```
from sklearn. model_selection import KFold
import numpy as np
X = ['a','b','c','d']
kf = KFold(n_splits=4, shuffle=True)
for train, test in kf. split(X):
    print(train, test)  #打印索引
    print("-"*50)
    print(np. array(X)[train], np. array(X)[test])
    print("*"*50)
```

K折交叉验证案例分析

```
from sklearn. model_selection import KFold
import numpy as np
X = ['a','b','c','d']
kf = KFold(n_splits=4, shuffle=True, random_state=1)
for train, test in kf. split(X):
    print(train, test)  #打印索引
    print("-"*50)
    print(np. array(X)[train], np. array(X)[test])
    print("*"*50)
```


计算得分

cross_val_score()函数： 函数用于评估模型的性能

cross_val_score(estimator, X, Y, cv=kfold, scoring='accuracy'):

- **estimator:** : 估计器，也就是模型
- **X, Y:** 数据值，标签值
- **cv:** 交叉验证的折数，是一个整数或者是一个交叉验证迭代器
- **scoring:** 评价指标（准确度）

算法比较 (一)

```
import pandas as pd
from sklearn.model_selection import KFold
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.model_selection import cross_val_score
```

算法比较 (二)

```
# 导入数据
```

```
dataset = pd.read_excel('客户信息及违约表现.xlsx')
```

```
#显示数据维度
```

数据维度: 行 1000, 列 6

```
print('数据维度: 行 %s, 列 %s' % dataset.shape)
```

```
# 查看数据的前10行
```

```
print(dataset.head(10))
```

收入	年龄	性别	历史授信额度	历史违约次数	是否
503999	46	1	0	1	1
452766	36	0	13583	0	1
100000	33	1	0	1	1
100000	25	0	0	1	1
258000	35	1	0	0	1
933333	31	0	28000	3	1
665000	40	1	5000	1	1
291332	38	0	0	0	1
259000	45	1	0	1	1
1076666	39	1	71000	2	1

算法比较 (三)

```
print(dataset.describe()) # 统计描述数据信息
```

```
print(dataset.groupby('是否违约').size()) # 是否违约分布情况
```

```
# 分离数据集
```

```
array = dataset.values
```

```
X = array[:, 0:5]
```

```
Y = array[:, 5]
```

```
seed = 7
```

```
kfold = KFold(n_splits=10, shuffle=True, random_state=seed)
```

	收入	年龄	...	历史违约次数	是否违约
count	1.000000e+03	1000.000000	...	1000.000000	1000.000000
mean	8.046335e+05	36.913000	...	0.633000	0.399000
std	6.525200e+05	5.620055	...	0.945095	0.489938
min	1.000000e+05	25.000000	...	0.000000	0.000000
25%	4.165325e+05	32.000000	...	0.000000	0.000000
50%	5.580000e+05	37.000000	...	0.000000	0.000000
75%	1.000000e+06	41.000000	...	1.000000	1.000000
max	6.602542e+06	49.000000	...	5.000000	1.000000

是否违约	
0	601
1	399

算法比较（四）

```
# 算法审查  
models = {}  
models['LR'] = LogisticRegression(max_iter=10000)  
models['DT'] = DecisionTreeClassifier()  
models['KNN'] = KNeighborsClassifier()  
models['NB'] = GaussianNB()  
models['SVM'] = SVC()
```

算法比较 (五)

评估算法

for key in models:

#cross_val_score:得到K折验证中每一折的得分

cv_results = **cross_val_score**(models[key], X, Y, cv=kfold)

print('%s: %f (%f)' %(key, cv_results.**mean()**, cv_results.**std()**))

LR: 0.751000 (0.042767)
DT: 0.781000 (0.027731)
KNN: 0.596000 (0.070880)
NB: 0.595000 (0.065154)
SVM: 0.601000 (0.068037)

scikit-learn

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

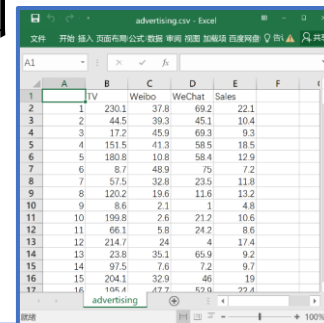
```
import pandas as pd
```

```
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
```

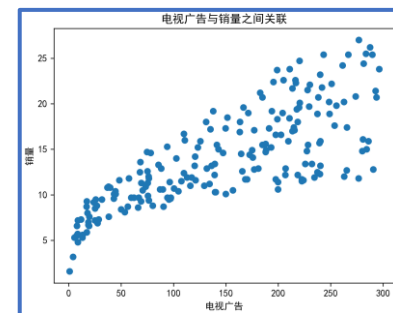
```
plt.rcParams['font.family']='SimHei'
```



	TV	Weibo	WeChat	Sales
1	230.1	37.8	69.2	22.1
2	44.5	39.3	45.1	10.4
3	17.2	45.9	69.3	9.3
4	151.5	41.3	58.5	18.5
5	180.8	10.8	58.4	12.9
6	8.7	48.9	75	7.2
7	57.5	32.8	23.5	11.8
8	120.2	19.6	11.6	13.2
9	8.6	2.1	1	4.8
10	109.8	2.6	21.2	10.6
11	66.1	5.8	24.2	8.6
12	214.7	24	4	17.4
13	23.8	35.1	65.9	9.2
14	97.5	7.6	7.2	9.7
15	204.1	32.9	46	19
16	194.4	47.7	57.8	22.4

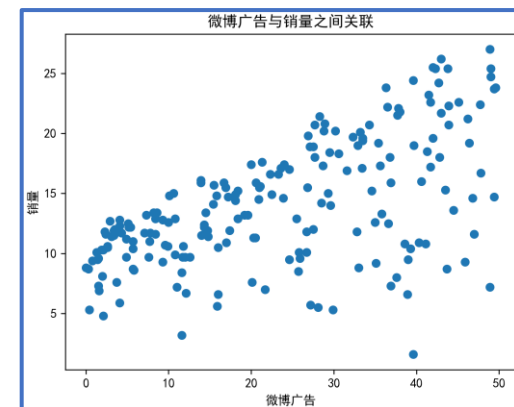
广告费用与销量线性回归模型（二）

```
#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()
```



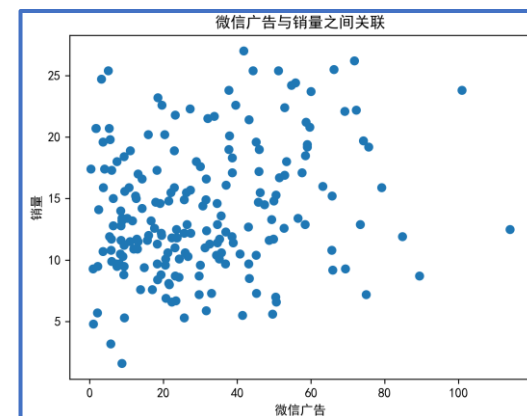
广告费用与销量线性回归模型 (三)

```
#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

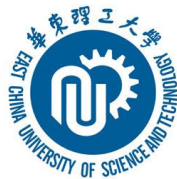
```
import numpy as np
```

```
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]  
>>>
```



谢 谢