# 学习目标

|  |
| --- |
|  |

# 1.认识决策树,属性分类问题

|  |
| --- |
|  |

## 决策树的核心思想是:如果高效的做决策.注意:决策时,特征的顺序是很重要的.

## 那么怎么确定特征的先后顺序?请看下面的分类原理

# 2.决策树分类原理

## 新看看一个银行贷款数据

|  |
| --- |
|  |
|  |
|  |

## 仔细观察这个表格,发现凡是有房子的人最终都获得了贷款

|  |
| --- |
|  |

## 然后我们发现如果一个人没有房子,但是又工作,也有机会获得贷款

|  |
| --- |
|  |

## 可见只需要根据有没有房子,和有没有工作就可以很快决定是否贷款给某人,但是如果你把顺序搞错了,效率就很低,如仅仅根据年龄是不能够决定是否贷款给某人,然后即使根据年龄和贷款信用还是无法快速决定是否贷款给某人,

|  |
| --- |
|  |

## 上面两个条件都具备了,还需要看你没有没有工作,可见这样子决策需要做3次,效率很低

|  |
| --- |
|  |
|  |

.

### 那么我们如何利用数学方法来实现上面的高效决策?

|  |
| --- |
|  |
|  |
|  |

### 然后我们使用信息熵的公司对上面的银行贷款表进行计算

|  |
| --- |
|  |
|  |

### 信息:消除随机不确定性的东西

# 3.决策树API,在tree子模块下面,是一个分类器,说明决策树是做分类问题的

|  |
| --- |
|  |

## max\_depth参数如果不设置,他会尽可能多的取拟合效率数据,但是树的深度太深就会出现一个问题,就是决策树在训练集上面表现很好,但是在测试集上面表现就不那么好,也就是决策树的泛化能力不好.所以适当设置树的深度不让他过度的伸展可以提高准确率.

## 先用一个比较简单的例子来学习一下决策树,就是使用sklearn里面的鸢尾花数据集先来用一下,重要决策树和knn不一样的地方就是它不需要做特征工程,代码如下

|  |
| --- |
| *"""  简单决策树案例:用决策树来对鸢尾花进行分类,使用的是sklearn自带的数据集 """* from sklearn.datasets import load\_iris from sklearn.model\_selection import train\_test\_split from sklearn.tree import DecisionTreeClassifier  # 1.加载数据 iris = load\_iris() # 2.划分数据集 x\_train, x\_test, y\_train, y\_test = train\_test\_split(iris.data, iris.target, random\_state=10) # 3.决策树预估器流程 estimator = DecisionTreeClassifier(criterion="entropy",max\_depth=10) # 使用信息熵 estimator.fit(x\_train,y\_train) # 4.模型评估 # 方法1,直接比对真实值与预测值 y\_predict = estimator.predict(x\_test) print("比对真实值与预测值的结果", y\_test == y\_predict) # 方法2.计算准确率 accuracy = estimator.score(x\_test, y\_test) print("准确率为:", accuracy) |

### 效果.

|  |
| --- |
|  |

# 4.案例:泰坦尼克号乘客生存预测

|  |
| --- |
|  |

### 实例代码

|  |
| --- |
|  |
| *""" 决策树案例:泰坦尼克号乘客生存预测数据来做titanic.txt """* import pandas as pd from sklearn.feature\_extraction import DictVectorizer from sklearn.tree import DecisionTreeClassifier, export\_graphviz from sklearn.model\_selection import train\_test\_split from sklearn.tree import DecisionTreeClassifier  # 设置数据格式 pd.set\_option("display.max\_column", None) pd.set\_option("display.width", 200) # 1.加载数据 titanic = pd.read\_csv("./titanic.txt") # 1.2 确定特征值和目标值 features = titanic[["pclass", "age", "sex"]] target = titanic["survived"] # print(features,target) # 2.数据处理: # 1>age字段的缺失值处理 features["age"].fillna(features["age"].mean(), inplace=True) # 缺失值用平均值来填补,数据就地处理 # print(features["age"]) # 2>把特征值转换为字典的形式 features = features.to\_dict(orient="records") # 先把特征值转换为字典列表格式 # 3.划分数据集 x\_train, x\_test, y\_train, y\_test = train\_test\_split(features, target, random\_state=12) # 4.字典特征抽取 transfer = DictVectorizer() x\_train = transfer.fit\_transform(x\_train) x\_test = transfer.transform(x\_test) # print(transfer.get\_feature\_names\_out()) # 5.决策树的预估器流程 estimator = DecisionTreeClassifier(criterion="entropy") # 使用信息熵 # estimator = DecisionTreeClassifier(criterion="gini",max\_depth=10) # 基尼系数 estimator.fit(x\_train, y\_train) # 6.模型评估 # 方法1,直接比对真实值与预测值 y\_predict = estimator.predict(x\_test) print("比对真实值与预测值的结果", y\_test == y\_predict) # 方法2.计算准确率 accuracy = estimator.score(x\_test, y\_test) print("准确率为:", accuracy) # 决策树可视化 export\_graphviz(estimator, "./titanic\_tree.dot",feature\_names=transfer.get\_feature\_names\_out()) |

### 效果

|  |
| --- |
| 比对真实值与预测值的结果 248 True  810 True  790 True  27 True  436 True  ...  799 True  923 False  65 True  83 True  1226 False  Name: survived, Length: 329, dtype: bool  准确率为: 0.8085106382978723 |

# 5.决策树可视化

|  |
| --- |
|  |
|  |
|  |

## 查看决策树图形的网址: <http://webgraphviz.com/>

## 我们给上面的例子添加决策树可视化功能,完整代码如下

|  |
| --- |
| *"""  简单决策树案例:用决策树来对鸢尾花进行分类,使用的是sklearn自带的数据集 """* from sklearn.datasets import load\_iris from sklearn.model\_selection import train\_test\_split from sklearn.tree import DecisionTreeClassifier,export\_graphviz   # 1.加载数据 iris = load\_iris() print(iris) # 2.划分数据集 x\_train, x\_test, y\_train, y\_test = train\_test\_split(iris.data, iris.target, random\_state=10) # 3.决策树预估器流程 estimator = DecisionTreeClassifier(criterion="entropy",max\_depth=10) # 使用信息熵 estimator.fit(x\_train,y\_train) # 4.模型评估 # 方法1,直接比对真实值与预测值 y\_predict = estimator.predict(x\_test) print("比对真实值与预测值的结果", y\_test == y\_predict) # 方法2.计算准确率 accuracy = estimator.score(x\_test, y\_test) print("准确率为:", accuracy) # 决策树可视化 export\_graphviz(estimator,"./iris\_tree2.dot",feature\_names=iris.feature\_names) |

## 然后我们把生成的iris\_tree2.dot的内容复制到<http://webgraphviz.com/> ,点击Generate Graph,效果如下

|  |
| --- |
|  |

# 6.决策树总结

|  |
| --- |
|  |