案例实战：员工离职预测模型搭建

模型搭建

1.数据读取与预处理

首先读取员工信息以及其交易离职表现，即是否离职记录，代码如下：

In [1]:

```
import pandas as pd
df = pd.read_excel('员工离职预测模型.xlsx')
df.head()
```

Out[1]:

	工资	满意度	考核得分	工程数量	月工时	工龄	离职
0	低	3.8	0.53	2	157	3	1
1	中	8.0	0.86	5	262	6	1
2	中	1.1	0.88	7	272	4	1
3	低	7.2	0.87	5	223	5	1
4	低	3.7	0.52	2	159	3	1

处理文本内容，代码如下：

In [2]:

```
df = df.replace({'工资': {'低': 0, '中': 1, '高': 2}})
df.head()
```

Out[2]:

	工资	满意度	考核得分	工程数量	月工时	工龄	离职
0	0	3.8	0.53	2	157	3	1
1	1	8.0	0.86	5	262	6	1
2	1	1.1	0.88	7	272	4	1
3	0	7.2	0.87	5	223	5	1
4	0	3.7	0.52	2	159	3	1

2.提取特征变量和目标变量

In [3]:

```
X = df.drop(columns='离职')
y = df['离职']
```

3.划分训练集和测试集

In [4]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=123)
```

4.模型训练及搭建

In [5]:

```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier(max_depth=3, random_state=123)
model.fit(X_train, y_train)
```

Out[5]:

```
DecisionTreeClassifier(max_depth=3, random_state=123)
```

模型预测及评估

1.直接预测是否离职

In [6]:

```
y_pred = model.predict(X_test)
print(y_pred[0:100])
```

```
[0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 1 1 0 1 0 1 0 0 1 0
 1 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0
 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 1 1 1 0 0 0]
```

In [7]:

```
# 通过构造DataFrame进行对比
a = pd.DataFrame()  # 创建一个空DataFrame
a['预测值'] = list(y_pred)
a['实际值'] = list(y_test)
a.head()
```

Out[7]:

	预测值	实际值
0	0	0
1	0	0
2	1	1
3	0	0
4	0	0

In [8]:

```
# 如果要查看整体的预测准确度，可以采用如下代码：
from sklearn.metrics import accuracy_score
score = accuracy_score(y_pred, y_test)
print(score)
```

```
0.9573333333333334
```

2.预测不离职&离职概率

其实分类决策树模型本质预测的并不是准确的0或1的分类，而是预测其属于某一分类的概率，可以通过如下代码查看预测属于各个分类的概率：

In [9]:

```
y_pred_proba = model.predict_proba(X_test)
print(y_pred_proba[0:5])
```

```
[[0.98526077 0.01473923]
 [0.98526077 0.01473923]
 [0.28600613 0.71399387]
 [0.98526077 0.01473923]
 [0.92283214 0.07716786]]
```

In [10]:

```
b = pd.DataFrame(y_pred_proba, columns=['不离职概率', '离职概率'])
b.head()
```

Out[10]:

	不离职概率	离职概率
0	0.985261	0.014739
1	0.985261	0.014739
2	0.286006	0.713994
3	0.985261	0.014739
4	0.922832	0.077168

如果想查看离职概率，即查看y_pred_proba的第二列，可以采用如下代码，这个是二维数组选取列的方法，其中逗号前的":"表示所有行，逗号后面的数字1则表示第二列，如果把数字1改成数字0，则提取第一列不离职概率。

In [11]:

```
y_pred_proba[:,1]
```

Out[11]:

```
array([0.01473923, 0.01473923, 0.71399387, ..., 0.01473923, 0.94594595,
       0.01473923])
```

3.特征重要性评估

In [12]:

```
model.feature_importances_
```

Out[12]:

```
array([0.        , 0.59810862, 0.14007392, 0.10638659, 0.00456495,
       0.15086592])
```

In [13]:

```
# 通过DataFrame进行展示，并根据重要性进行倒序排列，由特征变量对模型整体基尼系数下降的贡献来决定
features = X.columns  # 获取特征名称
importances = model.feature_importances_  # 获取特征重要性

# 通过二维表格形式显示
importances_df = pd.DataFrame()
importances_df['特征名称'] = features
importances_df['特征重要性'] = importances
importances_df.sort_values('特征重要性', ascending=False)
```

Out[13]:

	特征名称	特征重要性
1	满意度	0.598109
5	工龄	0.150866
2	考核得分	0.140074
3	工程数量	0.106387
4	月工时	0.004565
0	工资	0.000000