基于ModelArts的CART算法的实现和调用

实验目标

通过本案例的学习和课后作业的练习：

1. 学会搭建决策树模型；
2. 学会处理缺失值；
3. 学会使用网格搜索进行参数调优。

你也可以将本案例相关的 ipynb 学习笔记分享到 AI Gallery Notebook (https://marketplace.huaweicloud.com/markets/aihub/notebook/list/) 版块获得成长值 (https://marketplace.huaweicloud.com/markets/aihub/article/detail/?content_id=9b8d7e7a-a150-449e-ac17-2dcf76d8b492)，分享方法请查看此文档 (https://marketplace.huaweicloud.com/markets/aihub/article/detail/?content_id=8afec58a-b797-4bf9-acca-76ed512a3acb)。

案例内容介绍

CART假设决策树是二叉树，内部结点特征的取值为“是”和“否”，左分支是取值为“是”的分支，右分支是取值为“否”的分支。这样的决策树等价于递归地二分每个特征，将输入空间即特征空间划分为有限个单元，并在这些单元上确定预测的概率分布，也就是在输入给定的条件下输出的条件概率分布。

本案例推荐的理论学习视频：

- 《AI技术领域课程--机器学习》 决策树 (https://education.huaweicloud.com/courses/course-v1:HuaweiX+CBUCNXE086+Self-paced/courseware/f4092778ebec4ff1be33da5853ecaadf/3b6b2d586dbe4063ace3b63bcea0af59/)

注意事项

1. 如果您是第一次使用 JupyterLab，请查看《ModelArts JupyterLab使用指导》 (https://marketplace.huaweicloud.com/markets/aihub/article/detail/?content_id=03676d0a-0630-4a3f-b62c-07fba43d2857)了解使用方法；
2. 如果您在使用 JupyterLab 过程中碰到报错，请参考《ModelArts JupyterLab常见问题解决办法》 (https://marketplace.huaweicloud.com/markets/aihub/article/detail/?content_id=9ad8ce7d-06f7-4394-80ef-4dbf6cfb4be1)尝试解决问题。

实验步骤

1、导入数据集

In [1]:

```
import pandas as pd
import os
import moxing as mox

if not os.path.exists('Titanictrain.csv'):
    mox.file.copy('obs://modelarts-labs-bj4-v2/course/hwc_edu/machine_learning/datasets/CART/Titanictrain.csv',
                  'Titanictrain.csv')
data = pd.read_csv('Titanictrain.csv', sep=',')
data.head()
```

```
INFO:root:Using MoXing-v1.17.3-

INFO:root:Using OBS-Python-SDK-3.20.7
```

Out[1]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500

2、查看数据集信息

In [2]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
PassengerId    891 non-null int64
Survived       891 non-null int64
Pclass         891 non-null int64
Name           891 non-null object
Sex            891 non-null object
Age            714 non-null float64
SibSp          891 non-null int64
Parch          891 non-null int64
Ticket         891 non-null object
Fare           891 non-null float64
Cabin          204 non-null object
Embarked       889 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.6+ KB
```

3、处理缺失值

In [3]:

```
# 计算各特征缺失总数
total = data.isnull().sum().sort_values(ascending=False)
# 计算各特征缺失比例
percent = (data.isnull().sum() / data.isnull().count()).sort_values(ascending=False)
miss_data = pd.concat([total, percent], axis=1, keys=['Miss_Total', 'Miss_Percent'])
miss_data.head()
```

Out[3]:

	Miss_Total	Miss_Percent
Cabin	687	0.771044
Age	177	0.198653
Embarked	2	0.002245
Fare	0	0.000000
Ticket	0	0.000000

In [4]:
```
# 缺失值处理。
# 删除'Cabin’
del data['Cabin']
# 采用中位数填充缺失值
data['Age'] = data['Age'].fillna(data['Age'].median())
# 众数填充缺失值
data['Embarked'] = data['Embarked'].fillna(data['Embarked'].mode()[0])
# 查看数据情况
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>

RangeIndex: 891 entries, 0 to 890

Data columns (total 11 columns):

PassengerId    891 non-null int64

Survived       891 non-null int64

Pclass         891 non-null int64

Name           891 non-null object

Sex            891 non-null object

Age            891 non-null float64

SibSp          891 non-null int64

Parch          891 non-null int64

Ticket         891 non-null object

Fare           891 non-null float64

Embarked       891 non-null object

dtypes: float64(2), int64(5), object(4)

memory usage: 76.6+ KB
```

4、对乘客的Title进行处理

In [5]:
```
from sklearn.preprocessing import LabelEncoder

# 观察Name特征提取其中的Title称呼
data['Title'] = data['Name'].str.split(",", expand=True)[1].str.split(".", expand=True)[0]
# 将字符型变量做数值化处理
label = LabelEncoder()
data['Sex_Code'] = label.fit_transform(data['Sex'])
data['Title_Code'] = label.fit_transform(data['Title'])
data['Embarked'] = data['Embarked'].astype(str)
data['Embarked_Code'] = label.fit_transform(data['Embarked'])
# 考虑到PassengerId和Ticker为随机生成的变量，不作为影响目标变量的信息，因此特征选择时，将其去除
features = ['Pclass', 'Age', 'SibSp', 'Parch', 'Fare', 'Sex_Code', 'Title_Code', 'Embarked_Code', 'Survived']
data = data[features]
data.head()
```

Out[5]:

	Pclass	Age	SibSp	Parch	Fare	Sex_Code	Title_Code	Embarked_Code	Survived
0	3	22.0	1	0	7.2500	1	11	2	0
1	1	38.0	1	0	71.2833	0	12	0	1
2	3	26.0	0	0	7.9250	0	8	2	1
3	1	35.0	1	0	53.1000	0	12	2	1
4	3	35.0	0	0	8.0500	1	11	2	0

5、划分训练集和测试集

In [6]:
```
from sklearn.model_selection import train_test_split

X = data[['Pclass', 'Age', 'SibSp', 'Parch', 'Fare', 'Sex_Code', 'Title_Code', 'Embarked_Code']]
y = data[['Survived']]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
                                                    random_state=2)   # random_state为随机种子，确保每次划分的结果是相同的
```

6、训练模型

In [7]:
```
from sklearn.tree import DecisionTreeClassifier

dtc = DecisionTreeClassifier()
dtc.fit(X_train, y_train)
y_predict = dtc.predict(X_test)
```

7、模型评估

In [8]:

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score

# 模型评分：准确率，查全率，查准率，F1得分
accuracy_score = accuracy_score(y_test, y_predict)
recall_score = recall_score(y_test, y_predict)
precision_score = precision_score(y_test, y_predict)
f1_score = f1_score(y_test, y_predict)
print("DecisionTreeClassifier Results")
print("Accuracy      :", accuracy_score)
print("Recall        :", recall_score)
print("Precision     :", precision_score)
print("F1 Score      :", f1_score)
```

```
DecisionTreeClassifier Results

Accuracy      : 0.7541899441340782

Recall        : 0.6962025316455697

Precision     : 0.7333333333333333

F1 Score      : 0.7142857142857143
```

8、网格搜索

In [9]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import make_scorer

param = {'max_depth': [1, 3, 5, 7]}
# 采用网格搜索进行参数调优
gsearch = GridSearchCV(estimator=dtc, param_grid=param, cv=5, scoring='
f1')
gsearch.fit(X=X_train, y=y_train)
print("最优参数：{}".format(gsearch.best_params_))
print("最优模型：{}".format((gsearch.best_estimator_)))
print("模型最高分：{:.3f}".format(gsearch.score(X_test, y_test)))
```

```
最优参数：{'max_depth': 3}

最优模型：DecisionTreeClassifier(class_weight=None, criterion='gini',
max_depth=3,

            max_features=None, max_leaf_nodes=None,

            min_impurity_decrease=0.0, min_impurity_split=None,

            min_samples_leaf=1, min_samples_split=2,

            min_weight_fraction_leaf=0.0, presort=False, random_state
=None,

            splitter='best')

模型最高分：0.743
```

9、预测

In [10]:

```
from sklearn.tree import DecisionTreeClassifier
import numpy as np

# 选择最优模型进行预测
dtc = DecisionTreeClassifier(class_weight=None, criterion='gini', max_d
epth=3,
                             max_features=None, max_leaf_nodes=None,
                             min_samples_leaf=1, min_samples_split=2,
                             min_weight_fraction_leaf=0.0, presort=Fals
e, random_state=None,
                             splitter='best')
dtc.fit(X_train, y_train)
y_predict = dtc.predict(X_test)
# 打印预测结果
print('===================预测值=======================')
print(y_predict)
# 打印真实值
print('===================真实值=======================')
print(np.array(y_test).tolist())
```

```
===================预测值=======================

[0 0 1 0 1 0 0 0 0 0 0 1 1 0 0 1 0 0 1 0 0 1 0 1 0 1 1 0 0 0 0 0 0 0
0 1 1

 0 0 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 1 0 0 0 1
0 0 0

 1 1 0 1 1 0 1 0 0 0 1 1 0 0 1 0 0 0 0 0 0 1 0 1 0 0 1 0 1 1 0 1 1 1
0 0 0

 0 0 0 1 1 0 1 0 0 1 0 0 0 0 0 0 1 0 0 0 0 1 1 0 1 0 0 0 0 1 1 1 1 0
1 0 0

 0 1 0 1 0 0 1 0 0 1 1 0 0 0 0 0 1 1 0 0 1 0 0 1 1 1 0 0 0 0 1]

===================真实值=======================

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

以上是 CART 的实现方法，受限于篇幅原因，本案例未完全覆盖 CART 的全部操作，欢迎你将更全面的 CART 学习笔记分享到 AI Gallery Notebook (https://marketplace.huaweicloud.com/markets/aihub/notebook/list/) 版块获得成长值 (https://marketplace.huaweicloud.com/markets/aihub/article/detail/?content_id=9b8d7e7a-a150-449e-ac17-2dcf76d8b492)，分享方法请查看此文档 (https://marketplace.huaweicloud.com/markets/aihub/article/detail/?content_id=8afec58a-b797-4bf9-acca-76ed512a3acb)。

作业

请你利用本实验中学到的知识点，完成以下编程题：

1. 请你尝试修改 DecisionTreeClassifier() 函数的 criterion（衡量生成树的纯度） 参数的不同取值，看看该参数的修改对模型会有怎样的影响。 (https://marketplace.huaweicloud.com/markets/aihub/notebook/detail/?id=a4e4d696-843d-4321-90ba-909751ce29b8)
2. 请你尝试修改 DecisionTreeClassifier() 函数的 splitter（分裂点选择） 参数的不同取值，看看该参数的修改对模型会有怎样的影响。 (https://marketplace.huaweicloud.com/markets/aihub/notebook/detail/?id=0b99be3b-cb34-4e40-8ca9-929f27aa8a75)
3. 请你尝试修改 DecisionTreeClassifier() 函数的所有可调参数的不同取值，看看不同参数的不同取值组合，对模型会有怎样的影响。 (https://marketplace.huaweicloud.com/markets/aihub/notebook/detail/?id=087167c4-b6dc-45a0-a4b3-6f7adeadacb4)