****

**《数据挖掘》**

**实验报告**

**Spaceship Titanic | Kaggle**

**姓 名：段欣然**

**学 号：202011081033**

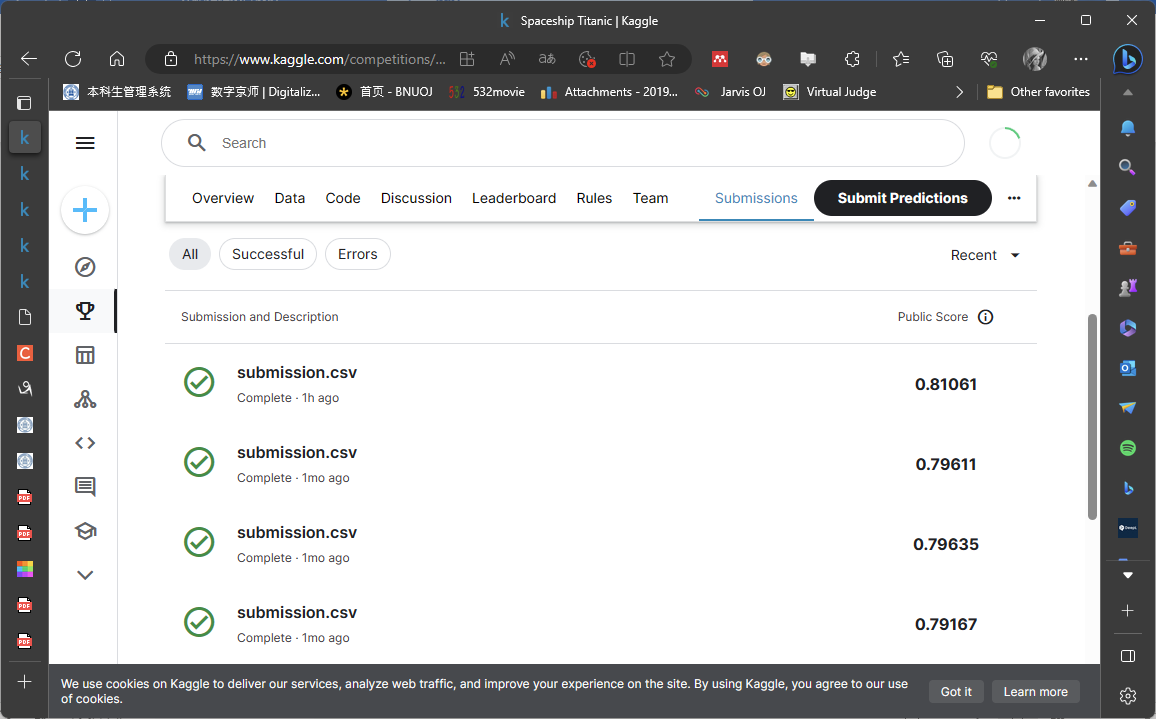
**专 业：计算机科学与技术**

**年 级：2020**

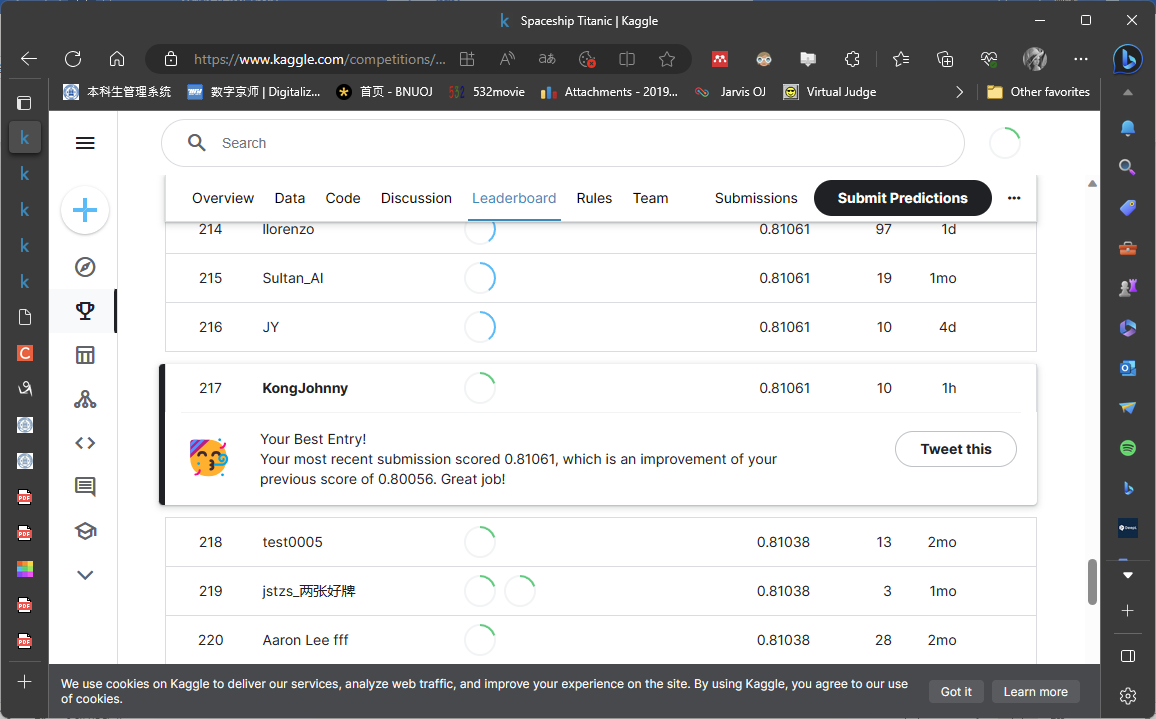
**完成日期： 2022年6月**

## 一、竞赛并提交的记录

1. 账号：[KongJohnny](https://www.kaggle.com/kongjohnny)
2. 提交的notebook网址：[notebook35d36512bc | Kaggle](https://www.kaggle.com/code/kongjohnny/notebook35d36512bc?scriptVersionId=134220274)
3. 提交记录的截图：



1. Leaderboard分数的截图，并标注。



## 二、数据挖掘方案

1. 问题定义

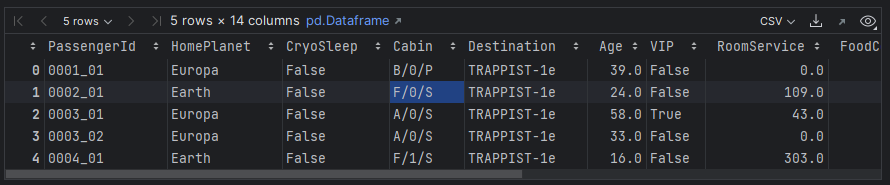
本次实验的问题是根据给定的数据集，构建一个模型来预测乘客是否会被运送到目的地。

1. 数据EDA与预处理

在这一部分，对原始数据进行了探索性数据分析（EDA）和预处理的步骤。

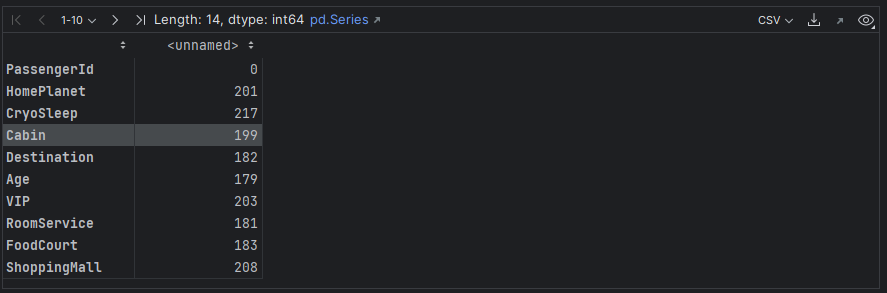
首先，使用pandas库读取了训练数据和测试数据，并使用head()方法查看了训练数据的前几行。

1. import pandas as pd
2. *#%%*
3. train\_data = pd.read\_csv('data/train.csv')
4. test\_data = pd.read\_csv('data/test.csv')
5. train\_data.head()



然后，使用isnull().sum()方法计算了训练数据中的缺失值数量。

1. train\_data.isnull().sum()



接下来，使用sklearn.impute.KNNImputer类对训练数据和测试数据中的缺失值进行了处理。具体地，将Transported和CryoSleep列中的数据转换为数值类型，并使用KNN插补方法对其他特征的缺失值进行填充。同时，对HomePlanet、Destination、Deck和Side列的缺失值进行了填充。

1. from sklearn.impute import KNNImputer
2. train\_data['Transported'] = train\_data['Transported']\*1
3. train\_data['CryoSleep'] = train\_data['CryoSleep']\*1
4. test\_data['CryoSleep'] = test\_data['CryoSleep']\*1
5. imputer = KNNImputer(n\_neighbors=5, weights='uniform', metric='nan\_euclidean')
6. train\_data[['Deck', 'Num', 'Side']] = train\_data['Cabin'].str.split('/', expand=True)
7. train\_data[['Age', 'RoomService', 'FoodCourt', 'ShoppingMall', 'Spa', 'VRDeck', 'CryoSleep', 'Num']] = imputer.fit\_transform(train\_data[['Age', 'RoomService', 'FoodCourt', 'ShoppingMall', 'Spa', 'VRDeck', 'CryoSleep', 'Num']])
8. train\_data['HomePlanet'].fillna('Earth', inplace=True)
9. train\_data['Destination'].fillna('TRAPPIST-1e', inplace=True)
10. train\_data['Deck'].fillna('F', inplace=True)
11. train\_data['Side'].fillna('P', inplace=True)
12. test\_data[['Deck', 'Num', 'Side']] = test\_data['Cabin'].str.split('/', expand=True)
13. test\_data[['Age', 'RoomService', 'FoodCourt', 'ShoppingMall', 'Spa', 'VRDeck', 'CryoSleep', 'Num']] = imputer.fit\_transform(test\_data[['Age', 'RoomService', 'FoodCourt', 'ShoppingMall', 'Spa', 'VRDeck', 'CryoSleep', 'Num']])
14. test\_data['HomePlanet'].fillna('Earth', inplace=True)
15. test\_data['Destination'].fillna('TRAPPIST-1e', inplace=True)
16. test\_data['Deck'].fillna('F', inplace=True)
17. test\_data['Side'].fillna('P', inplace=True)

随后，使用sklearn.preprocessing.LabelEncoder类对训练数据和测试数据中的Deck、Num和Side列进行了标签编码。

1. from sklearn.preprocessing import LabelEncoder
2. encoder = LabelEncoder()
3. train\_data['Deck'] = encoder.fit\_transform(train\_data['Deck'])
4. encoder = LabelEncoder()
5. train\_data['Num'] = encoder.fit\_transform(train\_data['Num'])
6. encoder = LabelEncoder()
7. train\_data['Side'] = encoder.fit\_transform(train\_data['Side'])
8. encoder = LabelEncoder()
9. test\_data['Deck'] = encoder.fit\_transform(test\_data['Deck'])
10. encoder = LabelEncoder()
11. test\_data['Num'] = encoder.fit\_transform(test\_data['Num'])
12. encoder = LabelEncoder()
13. test\_data['Side'] = encoder.fit\_transform(test\_data['Side'])

为了进一步进行特征工程，创建了一个名为AllSpending的新特征，该特征表示乘客在RoomService、FoodCourt、ShoppingMall、Spa和VRDeck上的总消费。

1. train\_data['AllSpending'] = train\_data['RoomService'] + train\_data['FoodCourt'] + train\_data['ShoppingMall'] + train\_data['Spa'] + train\_data['VRDeck']
2. test\_data['AllSpending'] = test\_data['RoomService'] + test\_data['FoodCourt'] + test\_data['ShoppingMall'] + test\_data['Spa'] + test\_data['VRDeck']

最后，使用sklearn.preprocessing.StandardScaler对训练数据和测试数据中的RoomService、Spa、VRDeck和AllSpending列进行了特征缩放，并使用pd.get\_dummies方法对特征进行了独热编码。

1. from sklearn.preprocessing import StandardScaler
2. train\_label = train\_data['Transported']
3. features = ['CryoSleep', 'RoomService', 'Spa', 'VRDeck', 'Deck', 'Side', 'AllSpending']
4. cols\_to\_norm = ['RoomService', 'Spa', 'VRDeck', 'AllSpending']
5. scaler = StandardScaler()
6. train\_data[cols\_to\_norm] = scaler.fit\_transform(train\_data[cols\_to\_norm])
7. test\_data[cols\_to\_norm] = scaler.transform(test\_data[cols\_to\_norm])
8. train\_input = pd.get\_dummies(train\_data[features])
9. test\_input = pd.get\_dummies(test\_data[features])
10. 特征工程

在特征工程阶段，进行了以下操作：

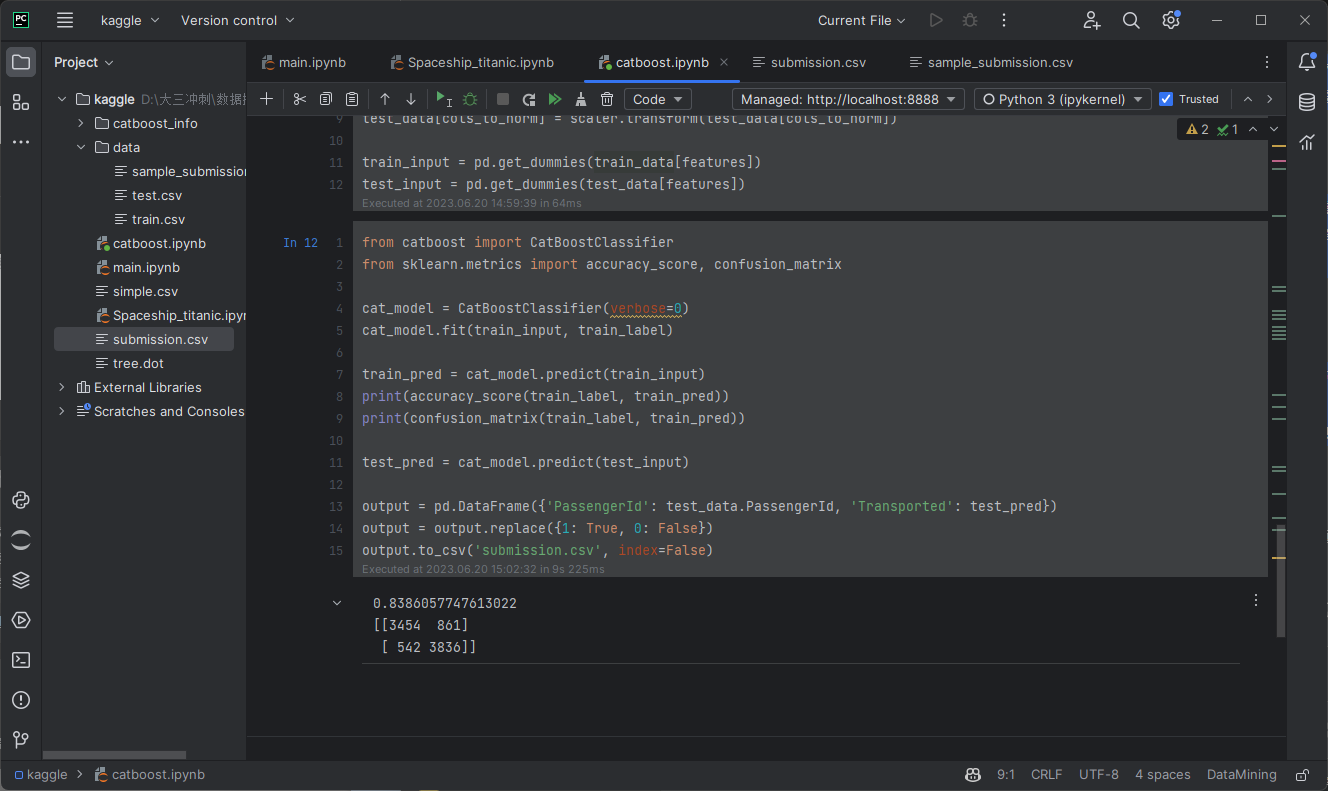
* 对Deck、Num和Side列使用了标签编码，将其转换为数值形式以便于模型训练；
* 创建了新的特征AllSpending，表示乘客在不同消费项目上的总消费。

相应代码已在上文给出。

1. 模型构建

在模型构建阶段，使用了CatBoostClassifier模型。首先，导入了CatBoostClassifier类和accuracy\_score、confusion\_matrix指标。然后，实例化了CatBoostClassifier模型，并使用训练数据进行了模型训练。接着，对训练数据进行了预测，并使用accuracy\_score和confusion\_matrix计算了训练集上的准确率和混淆矩阵。最后，对测试数据进行了预测，并将结果保存为submission.csv文件。

1. from catboost import CatBoostClassifier
2. from sklearn.metrics import accuracy\_score, confusion\_matrix
3. cat\_model = CatBoostClassifier(verbose=0)
4. cat\_model.fit(train\_input, train\_label)
5. train\_pred = cat\_model.predict(train\_input)
6. print(accuracy\_score(train\_label, train\_pred))
7. print(confusion\_matrix(train\_label, train\_pred))
8. test\_pred = cat\_model.predict(test\_input)
9. output = pd.DataFrame({'PassengerId': test\_data.PassengerId, 'Transported': test\_pred})
10. output = output.replace({1: True, 0: False})
11. output.to\_csv('submission.csv', index=False)



## 三、问题回答

1. 必答问题：你的方案里的哪一项技术（数据处理、模型改进）对提高分数产生了显著作用？结合理论知识对原因进行解释
2. 缺失值处理：在数据预处理阶段，使用sklearn.impute.KNNImputer对缺失值进行了处理。KNN插补是一种基于相似性的方法，它使用邻近样本的信息来估计缺失值。通过填充缺失值，我们避免了对带有缺失值的样本进行删除或简单填充的情况。这有助于保留更多有用的信息，可能提高了模型的性能。
3. 特征工程：在特征工程阶段，创建了新的特征AllSpending，表示乘客在不同消费项目上的总消费。这个新特征捕捉了乘客在多个消费项目上的总体消费情况。这可能有助于模型更好地理解乘客的消费行为，并提供更丰富的信息，从而提高了预测的准确性。
4. 模型选择：在模型构建阶段，选择了CatBoostClassifier作为分类器模型。CatBoost是一种基于梯度提升决策树（GBDT）的强大机器学习算法，具有处理类别特征、自动特征缩放和更好的泛化能力等优点。相比传统的GBDT算法，CatBoost能够更好地处理类别特征，无需额外的特征编码。因此，模型选择可能对提高分数产生显著作用。
5. 选答问题：你的方案的改进或者创新点是什么，并详细说明。
6. 标签编码的应用：在处理Deck、Num和Side列时，使用了sklearn.preprocessing.LabelEncoder进行标签编码。标签编码将类别特征转换为数值形式，使得模型能够处理这些特征。这种方法在处理有序类别特征时非常有用，可以将类别之间的顺序关系考虑到模型中。在这种情况下，通过标签编码将类别特征转换为数值特征，可能对模型的性能产生积极影响。
7. 特征缩放的应用：在特征工程的最后阶段，使用了sklearn.preprocessing.StandardScaler对RoomService、Spa、VRDeck和AllSpending等列进行了特征缩放。特征缩放对于基于距离或梯度的算法（如KNN、支持向量机和神经网络）尤其重要，它可以确保不同特征具有相似的尺度。通过特征缩放，可以避免某些特征对模型的训练产生过大的影响，提高了模型的稳定性和收敛速度。
8. 使用CatBoost模型进行训练：在模型构建阶段，选择了CatBoostClassifier作为分类器模型。CatBoost在梯度提升决策树（GBDT）的基础上，通过处理类别特征、自动特征缩放和更好的泛化能力等方面进行了改进。相比传统的GBDT算法，CatBoost能够更好地处理类别特征，无需额外的特征编码。这种模型选择可能对最终的预测性能和模型的鲁棒性有积极的影响。