泰坦尼克号乘客生存数据预处理报告

根据自己写的论文进行参考。

1. 数据预处理概述

本报告详细描述了泰坦尼克号乘客生存数据集的预处理过程。预处理是数据分析的关键步骤，直接影响后续模型的性能。我们的预处理流程包括数据清洗、特征工程、特征编码和数据标准化等步骤。

2. 原始数据描述

原始数据包含以下特征:

- 乘客 ID、姓名、性别、年龄
- 舱位等级、同船亲属数量、船票信息
- 票价、船舱号码、登船港口
- 生存状态(标签)

3. 数据预处理步骤

3.1 数据清洗

```
import pandas as pd

import numpy as np

from sklearn.preprocessing import StandardScaler


# 加载数据

train_df = pd.read_csv('train.csv')

test_df = pd.read_csv('test.csv')


# 处理缺失值

def handle_missing_values(df):

    # 年龄用均值填充

    df['Age'].fillna(df['Age'].mean(), inplace=True)
```

```
    # 船舱号码用众数填充
    df['Cabin'].fillna(df['Cabin'].mode()[0], inplace=True)
    # 登船港口用众数填充
    df['Embarked'].fillna(df['Embarked'].mode()[0], inplace=True)
    return df

train_df = handle_missing_values(train_df)
test_df = handle_missing_values(test_df)
```

3.2 特征工程

```
# 特征工程函数
def feature_engineering(df):
    # 从姓名中提取称呼
    df['Title'] = df['Name'].apply(lambda x: x.split(',')[1].split('.')[0].strip())

    # 创建亲属总数特征
    df['FamilySize'] = df['SibSp'] + df['Parch']

    # 创建是否独自旅行特征
    df['IsAlone'] = (df['FamilySize'] == 0).astype(int)

    # 票价等级
    df['FareLevel'] = (df['Fare'] > df['Fare'].mean()).astype(int)

    # 年龄等级
    df['AgeLevel'] = (df['Age'] > df['Age'].mean()).astype(int)

    return df
```

```
train_df = feature_engineering(train_df)
test_df = feature_engineering(test_df)
```

3.3 特征编码

```
from sklearn.preprocessing import LabelEncoder

# 分类变量编码
def encode_features(df):
    # 性别编码
    le = LabelEncoder()
    df['Sex'] = le.fit_transform(df['Sex'])

    # 登船港口编码
    df['Embarked'] = le.fit_transform(df['Embarked'])

    # 称呼编码
    title_mapping = {'Mr': 1, 'Miss': 2, 'Mrs': 3, 'Master': 4, 'Dr': 5,
                     'Rev': 6, 'Col': 7, 'Major': 8, 'Mlle': 9, 'Countess': 10,
                     'Ms': 11, 'Lady': 12, 'Jonkheer': 13, 'Don': 14, 'Dona': 15,
                     'Mme': 16, 'Capt': 17, 'Sir': 18}
    df['Title'] = df['Title'].map(title_mapping)
    df['Title'].fillna(0, inplace=True)

    return df

train_df = encode_features(train_df)
test_df = encode_features(test_df)
```

3.4 数据标准化

```
# 数值特征标准化
def standardize_features(df):
    scaler = StandardScaler()
    numeric_features = ['Age', 'Fare', 'SibSp', 'Parch', 'FamilySize']
    df[numeric_features] = scaler.fit_transform(df[numeric_features])
    return df

train_df = standardize_features(train_df)
test_df = standardize_features(test_df)
```

3.5 特征选择

```
# 选择最终使用的特征
selected_features = ['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare',
                     'Embarked', 'Title', 'FamilySize', 'IsAlone',
                     'FareLevel', 'AgeLevel']

X_train = train_df[selected_features]
y_train = train_df['Survived']
X_test = test_df[selected_features]
```

4. 预处理结果分析

经过预处理后，我们得到了以下特征：

1. 原始特征：舱位等级、性别、年龄、票价等
2. 衍生特征：亲属总数、是否独自旅行、票价等级等
3. 编码特征：性别、登船港口、称呼等

所有数值特征都已标准化，消除了量纲影响。

5. 数据可视化代码示例

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns

# 生存率与性别的关系
plt.figure(figsize=(8, 5))
sns.barplot(x='Sex', y='Survived', data=train_df)
plt.title('Survival Rate by Gender')
plt.xlabel('Gender (0=Male, 1=Female)')
plt.ylabel('Survival Rate')
plt.show()

# 生存率与舱位等级的关系
plt.figure(figsize=(8, 5))
sns.barplot(x='Pclass', y='Survived', data=train_df)
plt.title('Survival Rate by Passenger Class')
plt.xlabel('Passenger Class')
plt.ylabel('Survival Rate')
plt.show()

# 年龄分布
plt.figure(figsize=(10, 6))
sns.histplot(data=train_df, x='Age', hue='Survived', kde=True, bins=30)
plt.title('Age Distribution by Survival Status')
plt.xlabel('Age')
plt.ylabel('Count')
plt.show()
```

6. 模型训练示例代码

```
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report

# 初始化逻辑回归模型
logreg = LogisticRegression(max_iter=1000)

# 交叉验证
cv_scores = cross_val_score(logreg, X_train, y_train, cv=5, scoring='accuracy')
print(f"Cross-validation accuracy scores: {cv_scores}")
print(f"Mean CV accuracy: {np.mean(cv_scores):.4f}")

# 训练最终模型
logreg.fit(X_train, y_train)

# 预测训练集
y_pred = logreg.predict(X_train)

# 分类报告
print("\nClassification Report:")
print(classification_report(y_train, y_pred))
```

7. 结论

通过系统的数据预处理，我们：

1. 处理了缺失值问题
2. 创建了有意义的衍生特征
3. 对分类变量进行了适当编码

4. 标准化了数值特征
5. 选择了最相关的特征集

这些预处理步骤为后续的建模分析奠定了良好基础，有助于提高模型的性能和解释性。