数据预处理 3200104858

1 数据清洗

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scipy as scp
import warnings
warnings.filterwarnings("ignore")
```

```
data = pd.read_csv('train.csv')
data.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN

1.1 查看总体信息

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
```

```
 5   Age         714 non-null    float64
 6   SibSp       891 non-null    int64
 7   Parch       891 non-null    int64
 8   Ticket      891 non-null    object
 9   Fare        891 non-null    float64
 10  Cabin       204 non-null    object
 11  Embarked    889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

1.2查看缺失值

```
data.isnull().sum()
```

```
 assengerId       0
Survived          0
Pclass            0
Name              0
Sex               0
Age             177
SibSp             0
Parch             0
Ticket            0
Fare              0
Cabin           687
Embarked          2
dtype: int64
```

1.3 缺失值处理

1.3.1固定值替换

```
test=data[data['Age']==None]=0
```

```
test=data[data['Age']==np.nan]=0
```

数值列读取数据后，空缺值的数据类型为float64所以用None一般索引不到，比较的时候最好用np.nan

1.3.2 删除缺失行

```
test=data.dropna()
```

函数描述：DataFrame.dropna(*, axis=0, how=_NoDefault.no_default, thresh=_NoDefault.no_default, subset=None, inplace=False)

```
test=data.fillna(0)
```

函数描述：DataFrame.fillna(value=None, *, method=None, axis=None, inplace=False, limit=None, downcast=None)

1.3.3转换为新数据

检验Cabin缺失值部分数据的存活率与非缺失值之间是否存在显著差异，判断是否删除Cabin

```
Has_Cabin=data[data['Cabin'].isnull()==True]
No_Cabin=data[data['Cabin'].isnull()==False]
```

```
#Has_Cabin=Has_Cabin.copy()
#No_Cabin=No_Cabin.copy()
Has_Cabin['Cabin']=1
No_Cabin['Cabin']=0
data=pd.concat([Has_Cabin,No_Cabin])
data=data.reset_index()
#Has_Cabin['Cabin']
data.head()
```

	index	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250
2	4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500
3	5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583
4	7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750

1.3.4 中值填充

```
data['Age']=data['Age'].fillna(data['Age'].median())
```

1.3.5 众数填充

```
data['Embarked'].describe()
data['Embarked'].unique()
```

```
array(['S', 'Q', 'C', nan], dtype=object)
```

```
data['Embarked'] = data['Embarked'].fillna('S')
```

2.1重复值处理

```
data[data.duplicated()==True]
```

index	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embai

可知没有重复样本

```
test=data.drop_duplicates()
test=data.reset_index()
```

记录一下处理方法

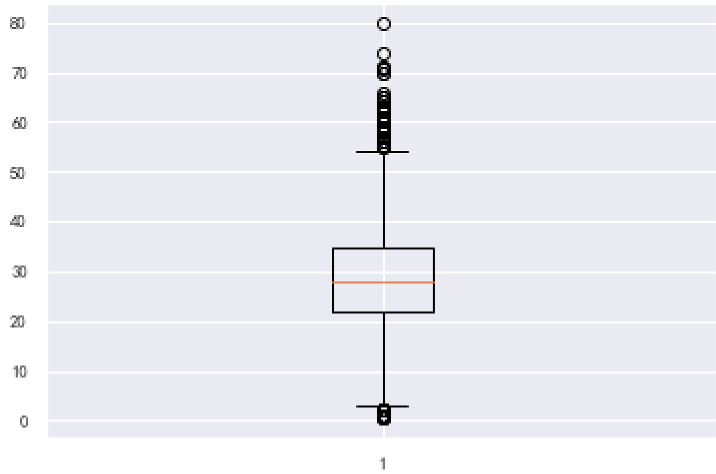
2.2 异常值处理

查看描述

```
data.describe()
```

	index	PassengerId	Survived	Pclass	Age	SibSp	Parch	F
count	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000	891.000
mean	445.000000	446.000000	0.383838	2.308642	29.361582	0.523008	0.381594	32.204
std	257.353842	257.353842	0.486592	0.836071	13.019697	1.102743	0.806057	49.693
min	0.000000	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000
25%	222.500000	223.500000	0.000000	2.000000	22.000000	0.000000	0.000000	7.910
50%	445.000000	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454
75%	667.500000	668.500000	1.000000	3.000000	35.000000	1.000000	0.000000	31.000
max	890.000000	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329

```
fig=plt.boxplot(data['Age'])
```



考虑到真实情况，不对这些年龄异常值做处理

3 数据变换

3.1 Sex处理

将Sex字段的文本数据转化为数字,其中female为0，male为1

```
data.loc[data['Sex']=='female','Sex']=0
data.loc[data['Sex']=='male','Sex']=1
```

3.2 Embarked处理

3.2.1 普通映射

将Embarked字段的文本数据转化为数字,其中S为0,C为1,Q为2

```
data.loc[data['Embarked']=='S','Embarked']=0
data.loc[data['Embarked']=='C','Embarked']=1
data.loc[data['Embarked']=='Q','Embarked']=2
```

3.2.2 独热编码

此处为了后续相关性矩阵的显示，仅做尝试，保存在test中

```
for feat in ["Age", "Embarked"]:
    x = pd.get_dummies(data[feat], prefix=feat)
    test = pd.concat([data, x], axis=1)
test['Embarked_0'].head()
```

```
0    1
1    1
2    1
3    0
4    1
Name: Embarked_0, dtype: uint8
```

3.3 姓名处理

对于姓名，主要根据其前缀来进行数据变换,首先提取前缀并查看可能性及数量

```
import re
def get_title(name):
    title_search = re.search(' ([A-Za-z]+)\.', name)

    if title_search:
        return title_search.group(1)
    return ""

titles = data["Name"].apply(get_title)
pd.value_counts(titles)
```

```
 Mr         517
Miss        182
Mrs         125
Master       40
Dr            7
Rev           6
Mlle          2
Major         2
Col           2
Jonkheer      1
Mme           1
Ms            1
Lady          1
Sir           1
Don           1
Capt          1
Countess      1
Name: Name, dtype: int64
```

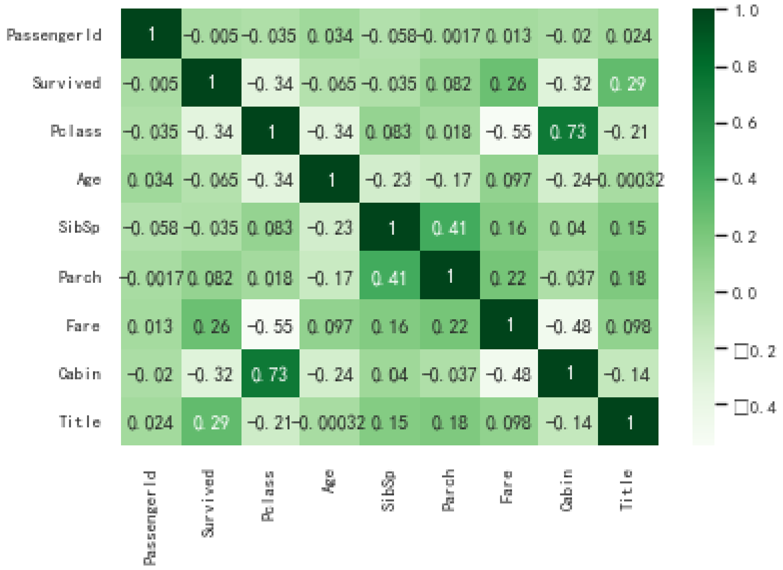
然后进行再编码

```
title_map = {"Mr":1, "Miss":2, "Mrs":3, "Master":4, "Dr":5, "Rev":6, "Mlle":7, "Col":8
```

```
titles = titles.map(title_map)
data["Title"] = titles
```

5 特征工程

```
import seaborn
cor=data.drop(['index'],axis=1).corr()
seaborn.set(font='SimHei',font_scale=0.8)
fig=seaborn.heatmap(cor,annot=True,cmap='Greens')
```



本次使用的数据集特征较少，通过计算皮尔逊相关系数并绘制热力图可知,仅Cabin和Pclass特征存在较大冗余，其他特征不存在明显的相关性，后续可考虑通过特征选择或者PCA降维进行特征的最优选择，本次作业仅考虑复现书上案例，因此不在此呈现。

```
data.to_csv('test_clear.csv')
```