报告

学号: 3021244270　　姓名: 黄琬婷　　班级:新工科一班

1. 目标

使用 python 完成对灾难数据集 Titanic Machine Learning 的全过程数据分析和预测。

2. 任务

按照以下实验步骤完成实验报告。

Data Dictionary

Variable	Definition	type	Explain
PassengerId	Id of passengers	int	Train data: 1-891 Test data: 892-1309
Survived	Survived or not	bool	0:No　1:Yes
Pclass	Ticket class	int	1: 1st　2: 2nd　3:3rd
Name	Name of passengers	string	
Sex	Sex of passengers	string	Male　female
Age	Age in years	int	
SibSp	# of siblings / spouses aboard the Titanic	int	
ParCh	# of parents / children aboard the Titanic	int	
Ticket	Ticket number	string	

Fare	Passenger fare	string	
Cabin	Cabin number	string	
Embarked	embarked Port of Embarkation	string	C = Cherbourg, Q = Queenstown, S = Southampton

Variable Notes

1. pclass: A proxy for socio-economic status (SES)

1st = Upper

2nd = Middle

3rd = Lower

2. age: Age is fractional if less than 1. If the age is estimated, is it in the form of xx.5

3. sibsp: The dataset defines family relations in this way...

Sibling = brother, sister, stepbrother, stepsister

4. Spouse = husband, wife (mistresses and fiancés were ignored)

5. parch: The dataset defines family relations in this way...

Parent = mother, father

Child = daughter, son, stepdaughter, stepson

Some children travelled only with a nanny, therefore parch=0 for them.

2.1 数据清洗

并不是每个变量都是有用的，也不是每个变量都适合建立模型，所以我们需要从原有的变量中提取信息来创建新的特征。

1) Name 变量

在 Name 变量中，每个人都有一个 Passenger 头衔。根据这个标题，我们可以用它来代替 Name 变量。比如：我们可以将标题进行划分，将低频标题合并成一个类。

For example: 'Dona', 'Lady', 'the Countess','Capt', 'Col', 'Don', 'Dr', 'Major', 'Rev', 'Sir', 'Jonkheer' can be merged.

Mlle <- 'Miss'

Ms <- 'Miss'

'Mme'<-'Mrs'

代码： Name.py

```
import pandas as pd
import re

#从 CSV 文件中读取训练和测试数据集
train = pd.read_csv('f:/train.csv')
test = pd.read_csv('f:/test.csv')

#从测试数据集中提取“PassengerId”来使用
PassengerId = test['PassengerId']
print(train.head(3)) #显示训练数据集的前 3 行

#将训练和测试数据集合并到一个列表中，以便于迭代
full_data = [train, test]

#从乘客姓名中提取标题
def get_title(name):
    title_search = re.search('([A-Za-z]+)\.', name) #使用正则表达式搜索
    #如果搜索到，返回；否则，返回一个空字符串
    if title_search:
```

```
        return title_search.group(1)
    return ""

#将 get_title 函数应用于两个数据集的“Name”列，创建一个新的“title”列
for dataset in full_data:
    dataset['Title'] = dataset['Name'].apply(get_title)

#将非常见标题分组为一个“稀有”类别，并对某些标题进行标准化
for dataset in full_data:
   dataset['Title']=dataset['Title'].replace(['Lady','Countess','Capt',
'Col', 'Don', 'Dr', 'Major', 'Rev', 'Sir', 'Jonkheer', 'Dona'], 'Rare')
    dataset['Title'] = dataset['Title'].replace('Mlle', 'Miss')
    dataset['Title'] = dataset['Title'].replace('Ms', 'Miss')
    dataset['Title'] = dataset['Title'].replace('Mme', 'Mrs')

print(train.head(3))
```

```
PS C:\Users\11298> & C:/Users/11298/AppData/Local/Programs/Python/Python310/python.exe c:/Users/11298/Desktop/Name.py
   PassengerId  Survived  Pclass                                               Name     Sex   Age  SibSp  Parch            Ticket     Fare Cabin Embarked
0            1         0       3                            Braund, Mr. Owen Harris    male  22.0      1      0         A/5 21171   7.2500   NaN        S
1            2         1       1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1      0          PC 17599  71.2833   C85        C
2            3         1       3                             Heikkinen, Miss. Laina  female  26.0      0      0  STON/O2. 3101282   7.9250   NaN        S
   PassengerId  Survived  Pclass                                               Name     Sex   Age  SibSp  Parch            Ticket     Fare Cabin Embarked Title
0            1         0       3                            Braund, Mr. Owen Harris    male  22.0      1      0         A/5 21171   7.2500   NaN        S    Mr
1            2         1       1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1      0          PC 17599  71.2833   C85        C   Mrs
2            3         1       3                             Heikkinen, Miss. Laina  female  26.0      0      0  STON/O2. 3101282   7.9250   NaN        S  Miss
```

2) Family size 变量:

我们可以在 SibSp 和 Parch 变量的基础上创建一个新的 Family size 变量。用柱状图列出家庭数量与存活率之间的关系。并对变量进行分化，例如大小{1,2-4,>4}或者其他.

代码： Familysize.py

```
import pandas as pd
import re

#从 CSV 文件中读取训练和测试数据集
train = pd.read_csv('f:/train.csv')
test = pd.read_csv('f:/test.csv')

#从测试数据集中提取“PassengerId”来使用
PassengerId = test['PassengerId']
print(train.head(3)) #显示训练数据集的前 3 行
```

```python
#将训练和测试数据集合并到一个列表中，以便于迭代
full_data = [train, test]

#从乘客姓名中提取标题
def get_title(name):
    title_search = re.search('([A-Za-z]+)\.', name) #使用正则表达式搜索
    #如果搜索到，返回；否则，返回一个空字符串
    if title_search:
        return title_search.group(1)
    return ""

#将 get_title 函数应用于两个数据集的“Name”列，创建一个新的“title”列
for dataset in full_data:
    dataset['Title'] = dataset['Name'].apply(get_title)

#计算包括乘客在内的家庭成员总数，并创建“family_num”列
for dataset in full_data:
    dataset['Family_num'] = 1 + dataset['SibSp'] + dataset['Parch']

#创建一个空的“FamilySize”列
for dataset in full_data:
    dataset['FamilySize'] = ''

#根据 family_num 大小进行分类
for dataset in full_data:
    dataset.loc[dataset['Family_num'] == 1, 'FamilySize'] = '1'
   dataset.loc[(dataset['Family_num'] > 1) & (dataset['Family_num'] <= 4),
'FamilySize'] = '2-4'
    dataset.loc[dataset['Family_num'] > 4, 'FamilySize'] = '>4'

print(train.head(3))
```

```
PS C:\Users\11298> & C:/Users/11298/AppData/Local/Programs/Python/Python310/python.exe c:/Users/11298/Desktop/Familysize.py
   PassengerId  Survived  Pclass                                               Name     Sex   Age  SibSp  Parch            Ticket     Fare Cabin Embarked
0            1         0       3                            Braund, Mr. Owen Harris    male  22.0      1      0         A/5 21171   7.2500   NaN        S
1            2         1       1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1      0          PC 17599  71.2833   C85        C
2            3         1       3                             Heikkinen, Miss. Laina  female  26.0      0      0  STON/O2. 3101282   7.9250   NaN        S
   PassengerId  Survived  Pclass                                               Name     Sex   Age  SibSp  Parch            Ticket     Fare Cabin Embarked Title  Family_num FamilySize
0            1         0       3                            Braund, Mr. Owen Harris    male  22.0      1      0         A/5 21171   7.2500   NaN        S    Mr          2        2-4
1            2         1       1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1      0          PC 17599  71.2833   C85        C   Mrs          2        2-4
2            3         1       3                             Heikkinen, Miss. Laina  female  26.0      0      0  STON/O2. 3101282   7.9250   NaN        S  Miss          1          1
PS C:\Users\11298>
```

3) More variables:

在 Cabin 变量中，我们可以得到甲板信息，也就是 Cabin 的第一个字母。

代码： MoreVariables.py

```
import pandas as pd
import re

#从 CSV 文件中读取训练和测试数据集
train = pd.read_csv('f:/train.csv')
test = pd.read_csv('f:/test.csv')

#从测试数据集中提取“PassengerId”来使用
PassengerId = test['PassengerId']
print(train.head(3)) #显示训练数据集的前 3 行

#将训练和测试数据集合并到一个列表中，以便于迭代
full_data = [train, test]

#从乘客姓名中提取标题
def get_title(name):
    title_search = re.search('([A-Za-z]+)\.', name) #使用正则表达式搜索
    #如果搜索到，返回；否则，返回一个空字符串
    if title_search:
        return title_search.group(1)
    return ""

#将 get_title 函数应用于两个数据集的“Name”列，创建一个新的“title”列
for dataset in full_data:
    dataset['Title'] = dataset['Name'].apply(get_title)

#将“Cabin”列值转换为字符串
for dataset in full_data:
    dataset['Cabin'] = dataset['Cabin'].apply(lambda x: str(x))

#将“客舱”列中的“nan”替换为“none”
for dataset in full_data:
    dataset['Cabin'] = dataset['Cabin'].replace('nan', 'none')

#提取“Cabin”值的第一个字符
for dataset in full_data:
   dataset['Cabin'] = dataset['Cabin'].apply(lambda x: x[0] if x != 'none'
else 'none')
```

```
print(train['Cabin'].head(5))
```

```
PS C:\Users\11298> & C:/Users/11298/AppData/Local
   PassengerId  Survived  Pclass
0            1         0       3
1            2         1       1  Cumings, Mrs. J
2            3         1       3
0    none
1       C
2    none
3       C
4    none
Name: Cabin, dtype: object
PS C:\Users\11298>
```

2.2 缺省值

方法:

1) 删除整行或整列

2）使用平均值

3）使用中位数

4) 预测

例如如下，你可以使用合适的方法来填充缺失的值：

a. missing Embarked

   use Median? Or same Passenger class and fare have same Embarked?

b. Missing Fare

c. Missing Age

   Prediction method

a.

代码： Embarked.py

```python
import pandas as pd

#从 CSV 文件中读取训练和测试数据集
train = pd.read_csv('f:/train.csv')
test = pd.read_csv('f:/test.csv')

#将训练和测试数据集合并到一个列表中，以便于迭代
full_data = [train, test]

freq_port=train.Embarked.dropna().mode()[0] #在 Embarked 列中查找最频繁出现的值
print(freq_port)
for dataset in full_data: #用最频繁的值填充 Embarked 列中缺失的值
    dataset['Embarked']=dataset['Embarked'].fillna(freq_port)

print(train.head(63))
```

```
PS C:\Users\11298> & C:/Users/11298/AppData/Local/Programs/Python/Python310/python.exe c:/Users/11298/Desktop/Embarked.py
S
    PassengerId  Survived  Pclass                                               Name     Sex   Age  SibSp  Parch            Ticket     Fare Cabin Embarked
0             1         0       3                            Braund, Mr. Owen Harris    male  22.0      1      0         A/5 21171   7.2500   NaN        S
1             2         1       1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1      0          PC 17599  71.2833   C85        C
2             3         1       3                             Heikkinen, Miss. Laina  female  26.0      0      0  STON/O2. 3101282   7.9250   NaN        S
3             4         1       1       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1      0            113803  53.1000  C123        S
4             5         0       3                           Allen, Mr. William Henry    male  35.0      0      0            373450   8.0500   NaN        S
..          ...       ...     ...                                                ...     ...   ...    ...    ...               ...      ...   ...      ...
58           59         1       2                    West, Miss. Constance Mirium  female   5.0      1      2       C.A. 34651  27.7500   NaN        S
59           60         0       3                 Goodwin, Master. William Frederick    male  11.0      5      2        CA 2144  46.9000   NaN        S
60           61         0       3                             Sirayanian, Mr. Orsen    male  22.0      0      0             2669   7.2292   NaN        C
61           62         1       1                               Icard, Miss. Amelie  female  38.0      0      0           113572  80.0000   B28        S
62           63         0       1                         Harris, Mr. Henry Birkhardt    male  45.0      1      0            36973  83.4750   C83        S
```

b.

代码：Fare.py

```python
import pandas as pd

#从 CSV 文件中读取训练和测试数据集
train = pd.read_csv('f:/train.csv')
test = pd.read_csv('f:/test.csv')
#将训练和测试数据集合并到一个列表中，以便于迭代
full_data = [train, test]

#用中位数进行补全 Fare
test['Fare'].fillna(test['Fare'].dropna().median(),inplace=True)

print(train.head(10))
```

```
PS C:\Users\11298> & C:/Users/11298/AppData/Local/Programs/Python/Python310/python.exe c:/Users/11298/Desktop/Fare.py
   PassengerId  Survived  Pclass                                               Name     Sex   Age  SibSp  Parch            Ticket     Fare Cabin Embarked
0            1         0       3                            Braund, Mr. Owen Harris    male  22.0      1      0         A/5 21171   7.2500   NaN        S
1            2         1       1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1      0          PC 17599  71.2833   C85        C
2            3         1       3                             Heikkinen, Miss. Laina  female  26.0      0      0  STON/O2. 3101282   7.9250   NaN        S
3            4         1       1       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1      0            113803  53.1000  C123        S
4            5         0       3                           Allen, Mr. William Henry    male  35.0      0      0            373450   8.0500   NaN        S
5            6         0       3                                   Moran, Mr. James    male   NaN      0      0            330877   8.4583   NaN        Q
6            7         0       1                            McCarthy, Mr. Timothy J    male  54.0      0      0             17463  51.8625   E46        S
7            8         0       3                     Palsson, Master. Gosta Leonard    male   2.0      3      1            349909  21.0750   NaN        S
8            9         1       3  Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)  female  27.0      0      2            347742  11.1333   NaN        S
9           10         1       2                Nasser, Mrs. Nicholas (Adele Achem)  female  14.0      1      0            237736  30.0708   NaN        C
```

c.

代码： Age.py

根据 Sex(0or1) 和 Pclass(1,2,3)两个特征来预测年龄

```
import pandas as pd
import numpy as np

#从 CSV 文件中读取训练和测试数据集
train = pd.read_csv('f:/train.csv')
test = pd.read_csv('f:/test.csv')

#将训练和测试数据集合并到一个列表中，以便于迭代
full_data = [train, test]

#将性别映射到数值（0 表示男，1 表示女）
for dataset in full_data:
    dataset['Sex']=dataset['Sex'].map({'female':1,'male':0}).astype(int)

#初始化一个 2x3 矩阵，用于存储基于性别和 Pclass 的 年龄中值
guess_ages=np.zeros((2,3))

#根据性别和 Pclass 预测缺失的“年龄”值
for dataset in full_data:
    for i in range(0,2):
        for j in range(0,3):
         guess_df=dataset[(dataset['Sex']==i)&(dataset['Pclass']==j+1)][
'Age'].dropna()
            age_guess=guess_df.median()
            guess_ages[i,j]=int(age_guess/0.5+0.5)*0.5
    for i in range(0,2):
        for j in range(0,3):
         dataset.loc[(dataset.Age.isnull())&(dataset.Sex==i)&(dataset.Pc
lass==j+1),'Age']=guess_ages[i,j]
    dataset['Age']=dataset['Age'].astype(int)

#创建 AgeBand 年龄阶段这一列，将年龄划分为不同段
train['AgeBand']=pd.cut(train['Age'],5)
#计算 AgeBand 的平均 Survived 率
```

```
age_band_survival = train[['AgeBand', 'Survived']].groupby(['AgeBand'],
as_index=False).mean().sort_values(by='AgeBand', ascending=True)

#将年龄划分为不同段
for dataset in full_data:
    dataset.loc[dataset['Age']<=16,'Age']=0
    dataset.loc[(dataset['Age']>16)&(dataset['Age']<=32),'Age']=1
    dataset.loc[(dataset['Age']>32)&(dataset['Age']<=48),'Age']=2
    dataset.loc[(dataset['Age']>48)&(dataset['Age']<=64),'Age']=3
    dataset.loc[dataset['Age']>64,'Age']

#从训练数据集中删除 AgeBand
train=train.drop(['AgeBand'],axis=1)

print(train.head(6))
```

```
PS C:\Users\11298> & C:/Users/11298/AppData/Local/Programs/Python/Python310/python.exe c:/Users/11298/Desktop/Age.py
c:\Users\11298\Desktop\Age.py:33: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or
 observed=True to adopt the future default and silence this warning.
  age_band_survival = train[['AgeBand', 'Survived']].groupby(['AgeBand'], as_index=False).mean().sort_values(by='AgeBand', ascending=True)
   PassengerId  Survived  Pclass                                               Name  Sex  Age  SibSp  Parch            Ticket     Fare Cabin Embarked
0            1         0       3                            Braund, Mr. Owen Harris    0    1      1      0         A/5 21171   7.2500   NaN        S
1            2         1       1  Cumings, Mrs. John Bradley (Florence Briggs Th...    1    2      1      0          PC 17599  71.2833   C85        C
2            3         1       3                             Heikkinen, Miss. Laina    1    1      0      0  STON/O2. 3101282   7.9250   NaN        S
3            4         1       1       Futrelle, Mrs. Jacques Heath (Lily May Peel)    1    2      1      0            113803  53.1000  C123        S
4            5         0       3                           Allen, Mr. William Henry    0    2      0      0            373450   8.0500   NaN        S
5            6         0       3                                   Moran, Mr. James    0    1      0      0            330877   8.4583   NaN        Q
PS C:\Users\11298>
```

2.3 特征工程

创建‘children’ and ‘mother’ 变量

Variable	Definition
children	0: child :Age<18 1: adult: Age>18
Mother	0: mother: female, adult, have one or more children, title is not ‘Miss’ 1: not mother

代码： 特征工程. py

```
import pandas as pd
import numpy as np
import re

#从 CSV 文件中读取训练和测试数据集
train = pd.read_csv('f:/train.csv')
test = pd.read_csv('f:/test.csv')

#将训练和测试数据集合并到一个列表中，以便于迭代
full_data = [train, test]

freq_port = train.Embarked.dropna().mode()[0]  #在 Embarked 列中查找最频繁出现的值
print(freq_port)
for dataset in full_data:
    # 用最频繁的值填充 Embarked 列中缺失的值
    dataset['Embarked'] = dataset['Embarked'].fillna(freq_port)

#将性别映射到数值（0 表示男，1 表示女）
for dataset in full_data:
    dataset['Sex']=dataset['Sex'].map({'female':1,'male':0}).astype(int)

#从乘客姓名中提取标题
def get_title(name):
    title_search = re.search('([A-Za-z]+)\.', name)  #使用正则表达式搜索
    #如果搜索到，返回；否则，返回一个空字符串
    if title_search:
        return title_search.group(1)
    return ""

#将 get_title 函数应用于两个数据集的“Name”列，创建一个新的“title”列
for dataset in full_data:
    dataset['Title'] = dataset['Name'].apply(get_title)

#将非常见标题分组为 稀有 类别，并对某些标题进行标准化
for dataset in full_data:
   dataset['Title']=dataset['Title'].replace(['Lady','Countess','Capt',
'Col', 'Don', 'Dr', 'Major', 'Rev', 'Sir', 'Jonkheer', 'Dona'], 'Rare')
    dataset['Title'] = dataset['Title'].replace('Mlle', 'Miss')
    dataset['Title'] = dataset['Title'].replace('Ms', 'Miss')
    dataset['Title'] = dataset['Title'].replace('Mme', 'Mrs')

#创建新的“儿童”列并根据年龄设置 0/1
```

```
for dataset in full_data:
    dataset['Children'] = 1
    for i in range(len(dataset)):
        if dataset['Age'][i] < 18:
            dataset.loc[i, 'Children'] = 0

#创建一个新的“母亲”列，并根据题目年龄、头衔、性别和船上父母/孩子的数量设置 0/1
for dataset in full_data:
    dataset['Mother'] = 1
    for i in range(len(dataset)):
        if dataset['Age'][i] >= 18 and dataset['Title'][i] != 'Miss' and
dataset['Sex'][i] == 'female' and dataset['Parch'][i] > 0:
            dataset.loc[i, 'Mother'] = 0

#初始化一个 2x3 矩阵，用于存储基于性别和 Pclass 的 年龄中值
guess_ages=np.zeros((2,3))

#根据性别和 Pclass 预测缺失的“年龄”值
for dataset in full_data:
    for i in range(0,2):
        for j in range(0,3):
         guess_df=dataset[(dataset['Sex']==i)&(dataset['Pclass']==j+1)][
'Age'].dropna()
            age_guess=guess_df.median()
            guess_ages[i,j]=int(age_guess/0.5+0.5)*0.5
    for i in range(0,2):
        for j in range(0,3):
         dataset.loc[(dataset.Age.isnull())&(dataset.Sex==i)&(dataset.Pc
lass==j+1),'Age']=guess_ages[i,j]
    dataset['Age']=dataset['Age'].astype(int)

#创建 AgeBand 年龄阶段这一列，将年龄划分为不同段
train['AgeBand']=pd.cut(train['Age'],5)
#计算 AgeBand 的平均 Survived 率
age_band_survival = train[['AgeBand', 'Survived']].groupby(['AgeBand'],
as_index=False).mean().sort_values(by='AgeBand', ascending=True)

#将年龄划分为不同段
for dataset in full_data:
    dataset.loc[dataset['Age']<=16,'Age']=0
    dataset.loc[(dataset['Age']>16)&(dataset['Age']<=32),'Age']=1
    dataset.loc[(dataset['Age']>32)&(dataset['Age']<=48),'Age']=2
    dataset.loc[(dataset['Age']>48)&(dataset['Age']<=64),'Age']=3
    dataset.loc[dataset['Age']>64,'Age']
```

```
for dataset in full_data:
    title_mapping = {"Mr": 1, "Miss": 2, "Mrs": 3, "Master": 4, "Rare": 5}
#将标记值映射为数值
    dataset['Title'] = dataset['Title'].map(title_mapping)
    dataset['Title'] = dataset['Title'].fillna(0) #用 0 填充 NaN
    dataset['Embarked'] = dataset['Embarked'].map({'S': 0, 'C': 1, 'Q':
2}).astype(int) #将标记值映射为数值

#从训练和测试数据集中删除不必要的列
train=train.drop(['AgeBand'],axis=1)
train_df = train.drop(['Ticket', 'Cabin', 'PassengerId', 'Parch', 'SibSp',
'Name'], axis=1).copy()
test_df = test.drop(['Ticket', 'Cabin', 'PassengerId', 'Parch', 'SibSp',
'Name'], axis=1).copy()

print(train_df.head(6))
```

```
c:\Users\11298\Desktop\特征工程.py:73: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future v
or or observed=True to adopt the future default and silence this warning.
  age_band_survival = train[['AgeBand', 'Survived']].groupby(['AgeBand'], as_index=False).mean().sort_values(by='AgeBand', ascending=True)
   Survived  Pclass  Sex  Age     Fare  Embarked  Title  Children  Mother
0         0       3    0    1   7.2500         0      1         1       1
1         1       1    1    2  71.2833         1      3         1       1
2         1       3    1    1   7.9250         0      2         1       1
3         1       1    1    2  53.1000         0      3         1       1
4         0       3    0    2   8.0500         0      1         1       1
5         0       3    0    1   8.4583         2      1         1       1
```

2.4 模型构建和预测

1.使用训练数据建立模型

2.利用测试数据验证模型。

3.使用错误率评价指标。

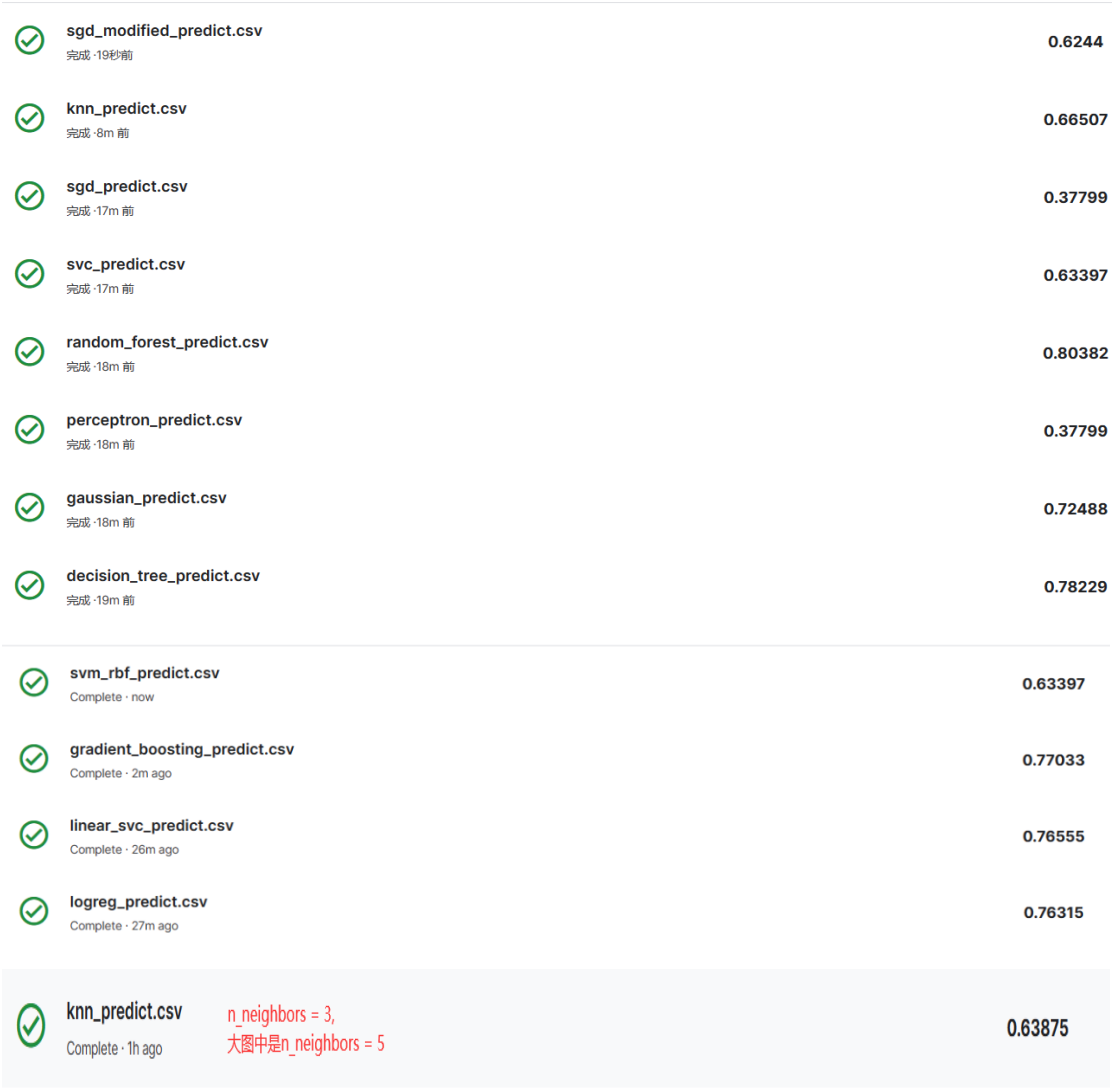
代码：test.py

使用多种方法对结果进行预测。

```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import
LogisticRegression,Perceptron,SGDClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC,LinearSVC
from sklearn.tree import DecisionTreeClassifier

#从 CSV 文件中读取训练和测试数据集
train = pd.read_csv('f:/processed_train.csv')
test = pd.read_csv('f:/processed_test.csv')

#从训练数据中提取特征 (X_train) 和目标变量 (Y_train)
X_train = train.drop(["Survived"], axis=1)  #去除目标变量列
Y_train = train["Survived"]  #目标变量列

#从测试数据中提取特征 (X_test)
X_test = test

#使用 K 近邻分类器建立模型
knn = KNeighborsClassifier(n_neighbors=5)  #使用 5 个近邻
knn.fit(X_train, Y_train)                   #在训练数据上拟合模型
Y_pred = knn.predict(X_test)               #对测试数据进行预测
acc_knn = round(knn.score(X_train, Y_train) * 100, 2)  #计算模型在训练数据上
的准确率

#创建包含 PassengerId 和预测结果 Survived 的 DataFrame
submission = pd.DataFrame({"PassengerId": test["PassengerId"], "Survived":
Y_pred})

#将预测结果保存为 CSV 文件
submission.to_csv('f:/knn_predict.csv', index=False)

#支持向量机分类器（SVC）
svc = SVC()
svc.fit(X_train,Y_train)
Y_pred = svc.predict(X_test)
acc_svc=round(svc.score(X_train,Y_train)*100,2)
submission=pd.DataFrame({"PassengerId":test["PassengerId"],"Survived":Y
_pred})
submission.to_csv('f:/svc_predict.csv',index=False)
```

```
#朴素贝叶斯分类器
gaussian = GaussianNB()
gaussian.fit(X_train,Y_train)
Y_pred = gaussian.predict(X_test)
acc_gaussian=round(gaussian.score(X_train,Y_train)*100,2)
submission=pd.DataFrame({"PassengerId":test["PassengerId"],"Survived":Y
_pred})
submission.to_csv('f:/gaussian_predict.csv',index=False)

#决策树分类器
decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train,Y_train)
Y_pred = decision_tree.predict(X_test)
acc_decision_tree=round(decision_tree.score(X_train,Y_train)*100,2)
submission=pd.DataFrame({"PassengerId":test["PassengerId"],"Survived":Y
_pred})
submission.to_csv('f:/decision_tree_predict.csv',index=False)

#随机森林分类器
random_forest = RandomForestClassifier(n_estimators=100)
random_forest.fit(X_train,Y_train)
Y_pred = random_forest.predict(X_test)
random_forest.score(X_train,Y_train)
acc_random_forest=round(random_forest.score(X_train,Y_train)*100,2)
submission=pd.DataFrame({"PassengerId":test["PassengerId"],"Survived":Y
_pred})
submission.to_csv('f:/random_forest_predict.csv',index=False)


#感知器分类器
perceptron = Perceptron()
perceptron.fit(X_train,Y_train)
Y_pred = perceptron.predict(X_test)
acc_perceptron = round(perceptron.score(X_train, Y_train) * 100, 2)
submission = pd.DataFrame({"PassengerId": test["PassengerId"],"Survived":
Y_pred})
submission.to_csv('f:/perceptron_predict.csv',index=False)

#随机梯度下降分类器
sgd = SGDClassifier()
sgd.fit(X_train, Y_train)
Y_pred = sgd.predict(X_test)
acc_sgd = round(sgd.score(X_train, Y_train) * 100, 2)
```

```
submission = pd.DataFrame({"PassengerId": test["PassengerId"],"Survived":
Y_pred})
submission.to_csv('f:/sgd_predict.csv',index=False)

#逻辑回归
logreg = LogisticRegression()
logreg.fit(X_train, Y_train)
Y_pred = logreg.predict(X_test)
acc_log = round(logreg.score(X_train, Y_train) * 100, 2)
submission = pd.DataFrame({"PassengerId": test["PassengerId"], "Survived":
Y_pred})
submission.to_csv('f:/logreg_predict.csv', index=False)

#线性支持向量分类
linear_svc = LinearSVC()
linear_svc.fit(X_train, Y_train)
Y_pred = linear_svc.predict(X_test)
acc_linear_svc = round(linear_svc.score(X_train, Y_train) * 100, 2)
submission = pd.DataFrame({"PassengerId": test["PassengerId"], "Survived":
Y_pred})
submission.to_csv('f:/linear_svc_predict.csv', index=False)

#具有修改损失参数的随机梯度下降（SGD 分类器）
sgd_modified = SGDClassifier(loss="modified_huber")
sgd_modified.fit(X_train, Y_train)
Y_pred = sgd_modified.predict(X_test)
acc_sgd_modified = round(sgd_modified.score(X_train, Y_train) * 100, 2)
submission = pd.DataFrame({"PassengerId": test["PassengerId"], "Survived":
Y_pred})
submission.to_csv('f:/sgd_modified_predict.csv', index=False)
```



一共通过 12 个不同模型来进行实验

分析数据：

（1） 随机森林分类器：成功率为 0.80382，表现最为优秀。这是一种集成学习方法，通过多个决策树的投票来提高分类性能，通常在各种类型的数据集上表现良好。

（2） 决策树分类器：成功率为 0.78229，表现较好。其是随机森林的基本组成部分，但相对于随机森林，决策树更容易过拟合。在一些情况下，通过调整树的深度等参数，可以提高性能。

（3） 梯度提升机：成功率为 0.77033，表现较好。这是一种通过逐步构建弱学习器的集成方法，在多轮迭代中逐渐提升模型性能。

（4） 线性支持向量分类 LinearSVC：成功率为 0.76555，表现较好。这表明在这个任务中，线性模型可能对于数据的线性关系有较好的拟合。

（5） 逻辑回归：成功率为 0.76315，表现较好。而这是一种广泛使用的线性分类器，通常在二分类问题上表现良好，也印证了（4）分析的结论。

（6） 朴素贝叶斯分类器：成功率为 0.72488。朴素贝叶斯可能对特征的独立性假设较为敏感。

（7） K 近邻分类器：成功率为 0.66507，表现较低。K 近邻中，使用五个近邻的成功率要

高于 3 个。

（8） 支持向量机的径向基函数核：成功率为 0.63397。径向基函数核在某些情况下可以更好地处理非线性关系，但需要调整参数以获得更好的性能，这是可以未来改进的方向。

（9） 支持向量机分类器：成功率为 0.63397。可能需要进一步调整参数或者特征工程。

（10） 具有修改损失参数的随机梯度下降分类器：成功率为 0.6244。表明在这个任务中，调整损失函数的参数对于性能提升的作用相对有限。

（11） 随机梯度下降分类器：成功率为 0.37799，表现最差。可能需要调整学习率、正则化项来提高性能。

（12） 感知器分类器：成功率为 0.37799，表现最差。感知器是一个简单的线性分类器，可能对于一些复杂的数据关系不够适用。

结论：

（1） 随机森林的效果最好，达到 0.80382；其他表现较好的模型包括决策树分类器、逻辑回归、梯度提升机和线性支持向量分类；随机梯度下降分类器和感知器分类器效果最差

（2） 加入附加条件的分类器会使得效果变优，例如：具有修改损失参数的随机梯度下降提升了普通随机梯度下降分类器的效果；linearSVC 的效果比普通 SVC 要好。
可继续尝试不同修正，例如随机梯度下降分类器，修改损失参数对成功率有一定提升，可以继续尝试不同的损失函数

（3） 集成化方法效果较为突出，同时，复杂的线性分类器在该项目中效果较好

提升准确率的方向：

（1） 模型选优：分别选取多种模型进行建模，根据模型评分进行初步比较，最终综合考虑多个性能指标来选择合适的预测模型

（2） 模型改进：对于成功率较低的模型，尝试调整模型的超参数，例如学习率、迭代次数、正则化参数等，以提高性能

（3） 特征挖掘与筛选：通过挖掘新的特征并测试选择不同特征时模型预测的准确性，以提高模型对数据的拟合能力，来选择最终训练模型的特征集合

（4） 数据预处理： 合理处理缺失值或异常值；
尝试不同的数据标准化或归一化方法，以确保数据在相同的尺度上

（5） 集成方法：将多个模型的预测结果结合起来，以提高整体性能