实战步骤说明文档

1 环境

代码在 Kaggle 的 Titanic 比赛下使用 Competition Notebook 在线运行。

2 初始处理

导入所需的 numpy 和 pandas 包。

通过 pd.read_csv 函数导入数据，使用 append 方法将训练数据与测试数据合并，再用 shape 属性打印出合并后数据集的大小，具体代码如下。

```
train_data = pd.read_csv("/kaggle/input/titanic/train.csv")
test_data = pd.read_csv("/kaggle/input/titanic/test.csv")
#合并训练数据与测试数据
full=train_data.append(test_data,ignore_index=True)
print(full.shape)
```

用 full.describe()函数查看数据集如下：

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	1309.000000	891.000000	1309.000000	1046.000000	1309.000000	1309.000000	1308.000000
mean	655.000000	0.383838	2.294882	29.881138	0.498854	0.385027	33.295479
std	378.020061	0.486592	0.837836	14.413493	1.041658	0.865560	51.758668
min	1.000000	0.000000	1.000000	0.170000	0.000000	0.000000	0.000000
25%	328.000000	0.000000	2.000000	21.000000	0.000000	0.000000	7.895800
50%	655.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	982.000000	1.000000	3.000000	39.000000	1.000000	0.000000	31.275000
max	1309.000000	1.000000	3.000000	80.000000	8.000000	9.000000	512.329200

可见存在数据缺失、数据异常的状况。

用 full.info()函数查看数据集每列的数据类型和总数如下：

```
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  1309 non-null   int64
 1   Survived     891 non-null    float64
 2   Pclass       1309 non-null   int64
 3   Name         1309 non-null   object
 4   Sex          1309 non-null   object
 5   Age          1046 non-null   float64
 6   SibSp        1309 non-null   int64
 7   Parch        1309 non-null   int64
 8   Ticket       1309 non-null   object
 9   Fare         1308 non-null   float64
 10  Cabin        295 non-null    object
 11  Embarked     1307 non-null   object
dtypes: float64(3), int64(4), object(5)
```

可见年龄、客舱号、登船港口以及票价的数据都有缺失。其中客舱号的缺失率很大。

3 数据清洗

3.1 数据预处理

年龄和船票价格的列为数值类型，填充方法是取各自的平均值进行填充。

```
#数据类型缺失
#Age
full['Age']=full['Age'].fillna(full['Age'].mean())#用平均值填充
#Fare
full['Fare']=full['Fare'].fillna(full['Fare'].mean())
```

港口和船舱号的列为字符串类型，填充方法是：港口用出现最频繁的值填充，船舱号由于缺失较多，由 U 表示 Unknown 填充。

```
#字符串类型缺失
#Embarked，缺失 2
full['Embarked']=full['Embarked'].fillna('S')#用出现最频繁的值填充
#Cabin，缺失值较多
full['Cabin']=full['Cabin'].fillna('U')#U 表示 Unknown
```

填充完毕后使用 full.describe()确认填充。

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	1309.000000	891.000000	1309.000000	1309.000000	1309.000000	1309.000000	1309.000000
mean	655.000000	0.383838	2.294882	29.881138	0.498854	0.385027	33.295479
std	378.020061	0.486592	0.837836	12.883193	1.041658	0.865560	51.738879
min	1.000000	0.000000	1.000000	0.170000	0.000000	0.000000	0.000000
25%	328.000000	0.000000	2.000000	22.000000	0.000000	0.000000	7.895800
50%	655.000000	0.000000	3.000000	29.881138	0.000000	0.000000	14.454200
75%	982.000000	1.000000	3.000000	35.000000	1.000000	0.000000	31.275000
max	1309.000000	1.000000	3.000000	80.000000	8.000000	9.000000	512.329200

3.2 特征工程

3.2.1 性别

将 male 映射为 1，female 映射为 0.

	Sex
0	1
1	0
2	0
3	0
4	1

3.2.2 登船港口

使用 get_dummy 来进行 one-hot 编码。因为港口有三种类别，因此生成了 3 列新的特征。

	Embarked_C	Embarked_Q	Embarked_S
0	0	0	1
1	1	0	0
2	0	0	1
3	0	0	1
4	0	0	1

3.2.3 客舱等级

提取方式同登船港口。

	Pclass_1	Pclass_2	Pclass_3
0	0	0	1
1	1	0	0
2	0	0	1
3	1	0	0
4	0	0	1

3.2.4 姓名

可以从姓名中提取出乘客的头衔和称谓，并将其分类。

最后得到：

	Master	Miss	Mr	Mrs	Officer	Royalty
0	0	0	1	0	0	0
1	0	0	0	1	0	0
2	0	1	0	0	0	0
3	0	0	0	1	0	0
4	0	0	1	0	0	0

3.2.5 家庭类别

根据 Parch+Sibsp+1 计算出的家庭成员的多少来为家庭的大小分类，用匿名函数 lambda 来进行条件判断来建立出家庭大小的类别。

	FamilySize	Family_Single	Family_Small	Family_Large
0	2	0	1	0
1	2	0	1	0
2	1	1	0	0
3	2	0	1	0
4	1	1	0	0

3.2.6 客舱号

提取客舱号的首字母来对船舱进行分类。

	Cabin_A	Cabin_B	Cabin_C	Cabin_D	Cabin_E	Cabin_F	Cabin_G	Cabin_T	Cabin_U
0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	1
3	0	0	1	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	1

具体代码可在项目中查看。

3.3 特征选择

采用“相关系数”筛选特征。用 corr()的方法得到相关系数矩阵：

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.005007	-0.038354	0.025731	-0.055224	0.008942	0.031416
Survived	-0.005007	1.000000	-0.338481	-0.070323	-0.035322	0.081629	0.257307
Pclass	-0.038354	-0.338481	1.000000	-0.366371	0.060832	0.018322	-0.558477
Age	0.025731	-0.070323	-0.366371	1.000000	-0.190747	-0.130872	0.171521
SibSp	-0.055224	-0.035322	0.060832	-0.190747	1.000000	0.373587	0.160224
Parch	0.008942	0.081629	0.018322	-0.130872	0.373587	1.000000	0.221522
Fare	0.031416	0.257307	-0.558477	0.171521	0.160224	0.221522	1.000000

将之前提取的有用的数据特征都联系起来，得出各种特征与“survived”的相关性矩阵（图为部分）：

	Age	PassengerId	Master	Miss	Mr	Mrs	Officer	Royalty	Pclass_1	Pclass_2	...
Age	1.000000	0.025731	-0.363923	-0.254146	0.165476	0.198091	0.162818	0.059466	0.362587	-0.014193	...
PassengerId	0.025731	1.000000	0.002254	-0.050027	0.014116	0.033299	0.002231	0.004400	0.026495	0.022714	...
Master	-0.363923	0.002254	1.000000	-0.110595	-0.258902	-0.093887	-0.029567	-0.015002	-0.084504	-0.016933	...
Miss	-0.254146	-0.050027	-0.110595	1.000000	-0.585809	-0.212435	-0.066899	-0.033945	-0.011733	-0.025440	...
Mr	0.165476	0.014116	-0.258902	-0.585809	1.000000	-0.497310	-0.156611	-0.079466	-0.099725	-0.038595	...

将相关系数矩阵中的“Survived”生存情况这一列提取出来，并按降序排列，得到各个特征与生存情况从[-1,1]的相关系数值。

```
Survived         1.000000
Mrs              0.344935
Miss             0.332795
Pclass_1         0.285904
Family_Small     0.279855
Fare             0.257307
Cabin_B          0.175095
Embarked_C       0.168240
Cabin_D          0.150716
Cabin_E          0.145321
Cabin_C          0.114652
Pclass_2         0.093349
Master           0.085221
Cabin_F          0.057935
Royalty          0.033391
Cabin_A          0.022287
FamilySize       0.016639
Cabin_G          0.016040
Embarked_Q       0.003650
PassengerId     -0.005007
Cabin_T         -0.026456
Officer         -0.031316
Age             -0.070323
Family_Large    -0.125147
Embarked_S      -0.149683
Family_Single   -0.203367
Cabin_U         -0.316912
Pclass_3        -0.322308
Sex             -0.543351
Mr              -0.549199
Name: Survived, dtype: float64
```

根据各个特征和生存情况的相关系数选择有代表性的特征作为模型输入（图为部分）：

	Master	Miss	Mr	Mrs	Officer	Royalty	Pclass_1	Pclass_2	Pclass_3	FamilySize	...	Cabin_E	Cabin_F	Cabin_G	Cabin_T	Cabin_U
0	0	0	1	0	0	0	0	0	1	2	...	0	0	0	0	1
1	0	0	0	1	0	0	1	0	0	2	...	0	0	0	0	0
2	0	1	0	0	0	0	0	0	1	1	...	0	0	0	0	1
3	0	0	0	1	0	0	1	0	0	2	...	0	0	0	0	0
4	0	0	1	0	0	0	0	0	1	1	...	0	0	0	0	1

4 构建模型

在合并数据集中 full_x1 中用数据框的 loc 属性根据索引值提取出原始数据和预测数据。再使用 train_test_split 函数从原始数据集中拆分出训练数据集和测试数据集。

```
#拆分原始数据集和预测集
sourceRow=891
source_x=full_x1.loc[0:sourceRow-1,:]#特征
source_y=full.loc[0:sourceRow-1,'Survived']#标签

pred_x=full_x1.loc[sourceRow:,:]

from sklearn.model_selection import train_test_split
#建立模型用的训练集和测试集
train_x,test_x,train_y,test_y=train_test_split(source_x,source_y,train_size=0.8)
```

使用随机森林算法建立模型并评估模型好坏：

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=100, max_depth=5, random_state=1)
model.fit(train_x, train_y)
model.score(test_x,test_y)
```

最后将预测输出到 submission.csv.

```
pred_y= model.predict(pred_x)
pred_y= pred_y.astype(int)

output = pd.DataFrame({'PassengerId': test_data.PassengerId, 'Survived': pred_y})
output.to_csv('submission.csv', index=False)
```