附录

2020 级医疗数据处理实践-评测任务

1 个人信息

- 班级：大数据 2002
- 姓名：周华

2 任务

- 数据集：项目六
- 数据集文件：06.titanic-test.csv,06.titanic-train.csv
- 简要介绍：泰坦尼克号数据

3 代码部分

3.1 导入包

[1]:
```
# 导入常用包
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import random as rnd
%matplotlib inline

# 导入深度学习框架 pytorch
import torch
from torch.autograd import Variable
import torch.nn as nn
```

```
import torch.nn.functional as F
import torch.optim as optim

# 导入机器学习包
from sklearn.linear_model import LogisticRegression   # 逻辑回归
from sklearn.svm import SVC, LinearSVC                 #SVC
from sklearn.ensemble import RandomForestClassifier   # 随机森林
from sklearn.neighbors import KNeighborsClassifier    #KNN
from sklearn.naive_bayes import GaussianNB            # 贝叶斯
from sklearn.linear_model import Perceptron           # 感知机
from sklearn.linear_model import SGDClassifier        #SGD
from sklearn.tree import DecisionTreeClassifier       # 决策树

# 其他包
import warnings
warnings.filterwarnings("ignore")
```

3.2 数据读取

[2]:
```
train=pd.read_csv('./input/06.titanic-train.csv')
test=pd.read_csv('./input/06.titanic-test.csv')
combine = [train, test]
```

3.3 EDA（数据初探）

3.3.1 数据查看

查看前 5 行数据

[3]:
```
train.head(5)
```

[3]:
```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                      Name   Sex   Age  SibSp  \
0  Braund, Mr. Owen Harris  male  22.0      1
```

```
1  Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                            Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S
```

包含 12 个属性，分别为乘客 Id，是否幸存，客舱等级，姓名，性别，年龄，同代亲属数，不同代亲属数，船票编号，床票价格，客舱号，登船港口

查看数据行数列数

[4]:
```
train.shape
```

[4]:
```
(891, 12)
```

查看数据基本信息

[5]:
```
train.info()
```

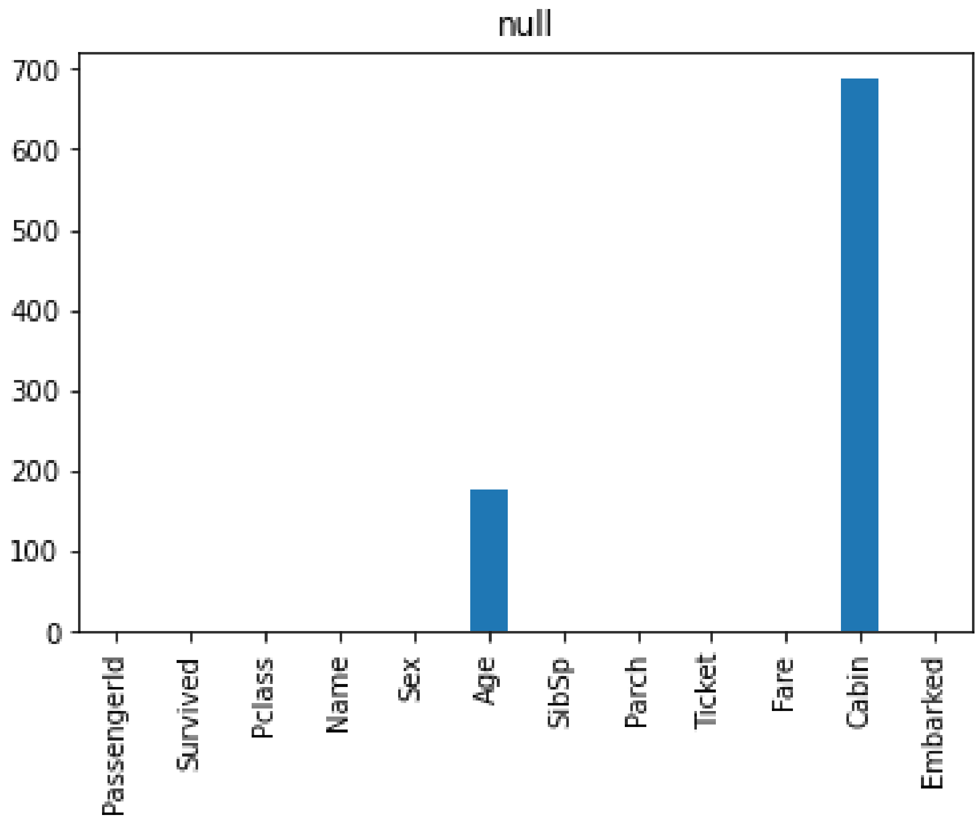
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null 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

查看缺失值

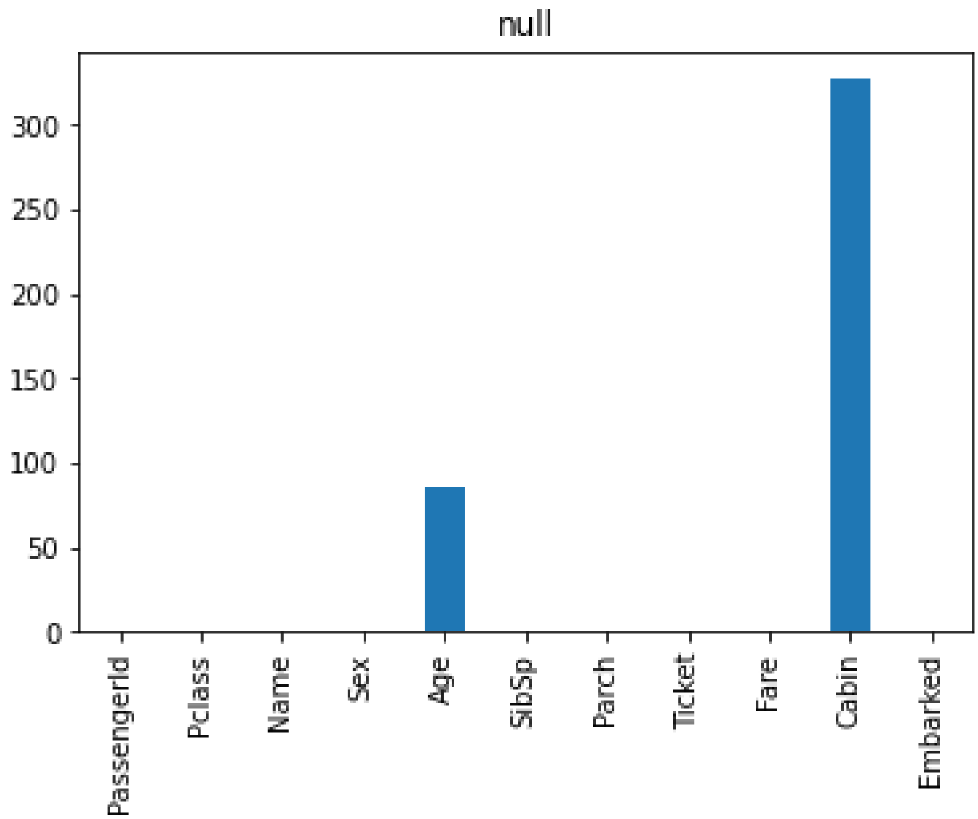
[6]:
```
train.isnull().sum().plot(kind='bar',title='null')
train.isnull().sum()
```

[6]:
```
PassengerId      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
```



[7]:
```
test.isnull().sum().plot(kind='bar',title='null')
test.isnull().sum()
```

[7]:
```
PassengerId      0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
```



age,cabin,embarke 含有缺失值

查看数据统计信息

[8]:
```
train.describe()
```

[8]:

	PassengerId	Survived	Pclass	Age	SibSp \
count	891.000000	891.000000	891.000000	714.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008
std	257.353842	0.486592	0.836071	14.526497	1.102743
min	1.000000	0.000000	1.000000	0.420000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000
50%	446.000000	0.000000	3.000000	28.000000	0.000000
75%	668.500000	1.000000	3.000000	38.000000	1.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000

```
          Parch         Fare
```

count	891.000000	891.000000
mean	0.381594	32.204208
std	0.806057	49.693429
min	0.000000	0.000000
25%	0.000000	7.910400
50%	0.000000	14.454200
75%	0.000000	31.000000
max	6.000000	512.329200

各个属性数值均合理

[9]:
```
train.describe(include=['O'])
```

[9]:

	Name	Sex	Ticket	Cabin	Embarked
count	891	891	891	204	889
unique	891	2	681	147	3
top	Braund, Mr. Owen Harris	male	347082	B96 B98	S
freq	1	577	7	4	644

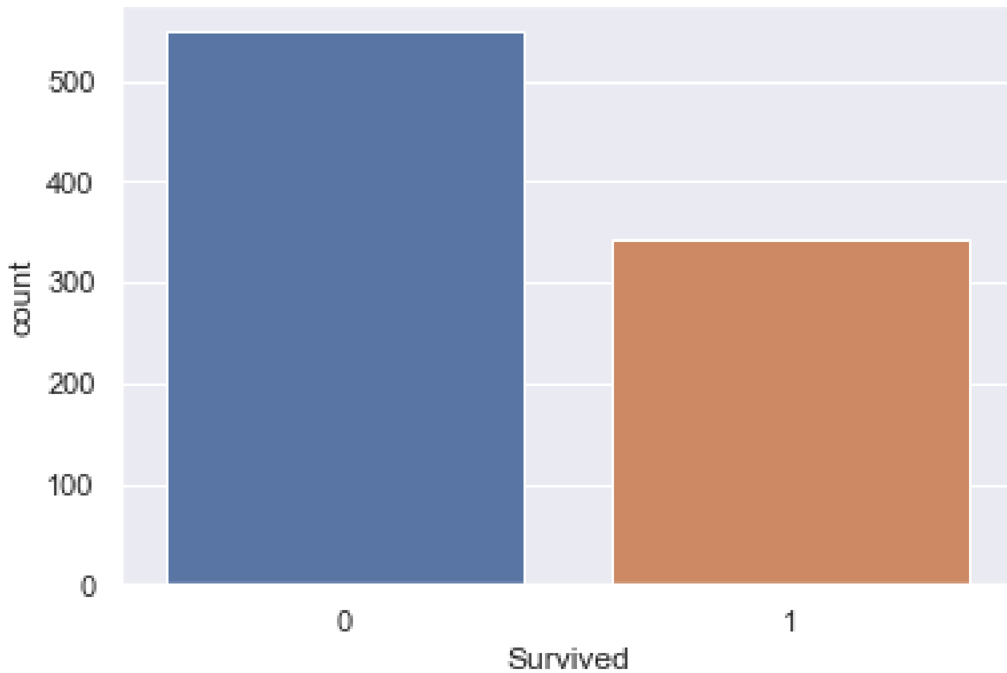
登船港口只有三种，S 最多

3.3.2 单个属性数据探索

Survived

[10]:
```
sns.set_theme(style="darkgrid")
sns.countplot(x=train.Survived,data=train)
train.Survived.value_counts()
```

[10]:
```
0    549
1    342
Name: Survived, dtype: int64
```



[11]:
```
train.Survived.value_counts()[1]/train.Survived.value_counts().sum()
```

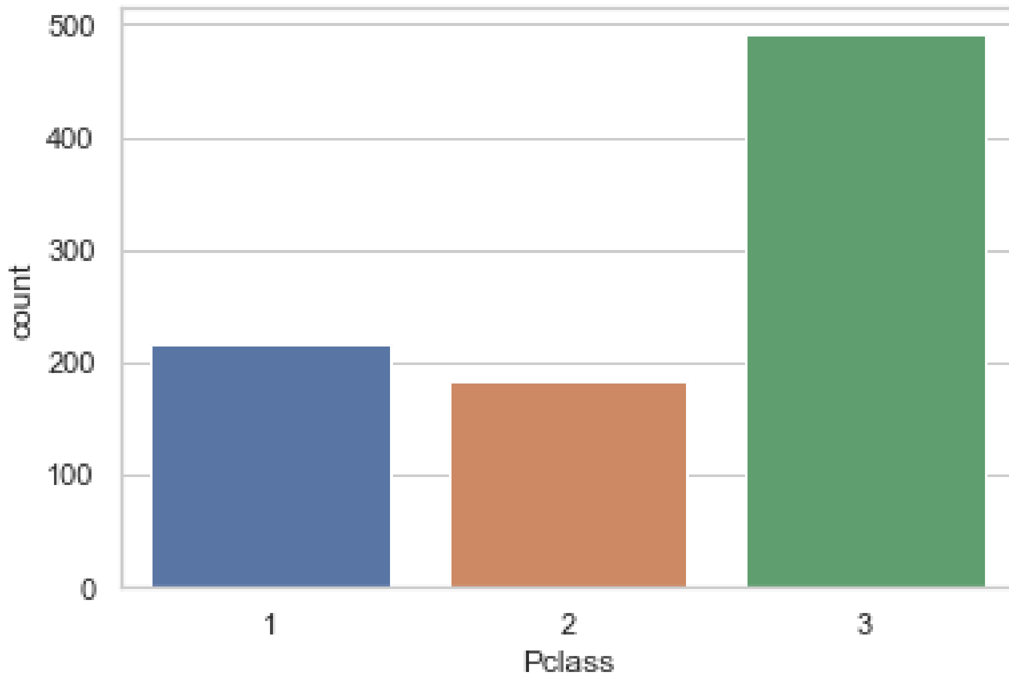
[11]: 0.3838383838383838

幸存者比例为 38.38%

Pclass

[12]:
```
sns.set_theme(style="whitegrid")
sns.countplot(x=train.Pclass,data=train)
train.Pclass.value_counts()
```

[12]:
```
3    491
1    216
2    184
Name: Pclass, dtype: int64
```



[13]:
```
train.Pclass.value_counts() / train.Pclass.value_counts().sum()
```

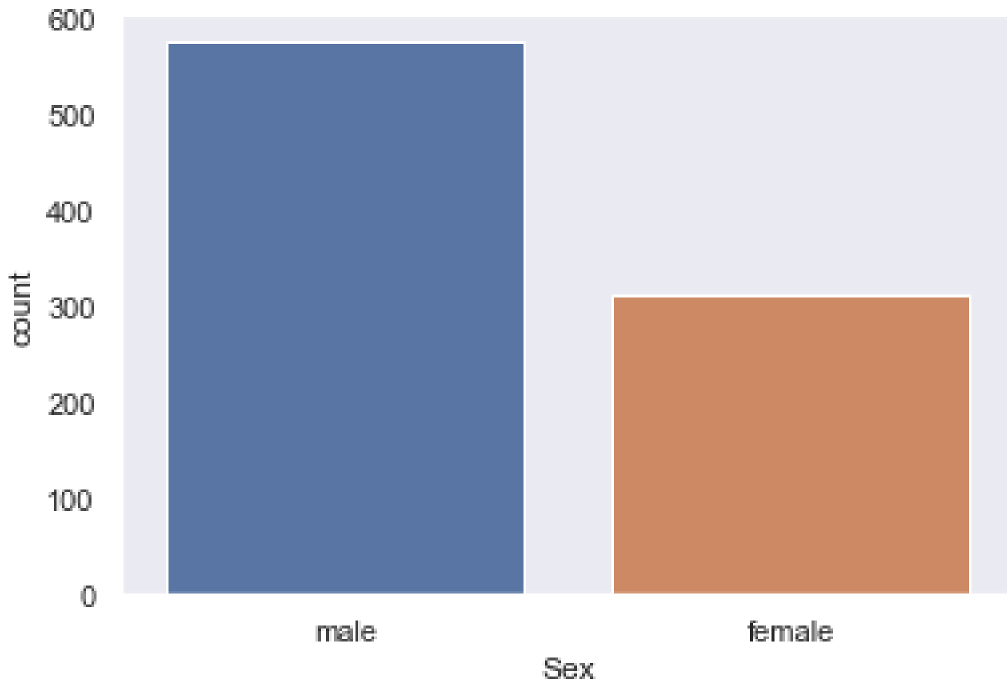
[13]:
```
3    0.551066
1    0.242424
2    0.206510
Name: Pclass, dtype: float64
```

三等仓人数最多，为 55.1%，一等舱为 24.2%，二等舱 20.7%

Sex

[14]:
```
sns.set_theme(style="dark")
sns.countplot(x=train.Sex,data=train)
train.Sex.value_counts()
```

[14]:
```
male      577
female    314
Name: Sex, dtype: int64
```



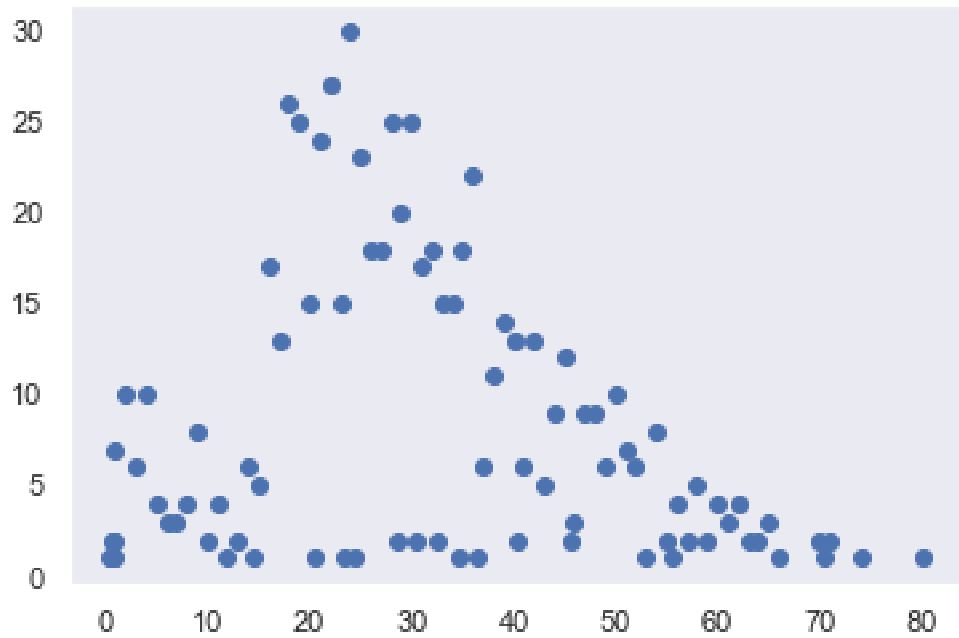
[15]:
```
train.Sex.value_counts()/train.Sex.value_counts().sum()
```

[15]:
```
male      0.647587
female    0.352413
Name: Sex, dtype: float64
```

男性更多，占 64.76%

Age

[16]:
```
plt.scatter(train.Age.value_counts().index,train.Age.value_counts())
plt.show()
train.Age.value_counts().sort_values(axis=0)
```



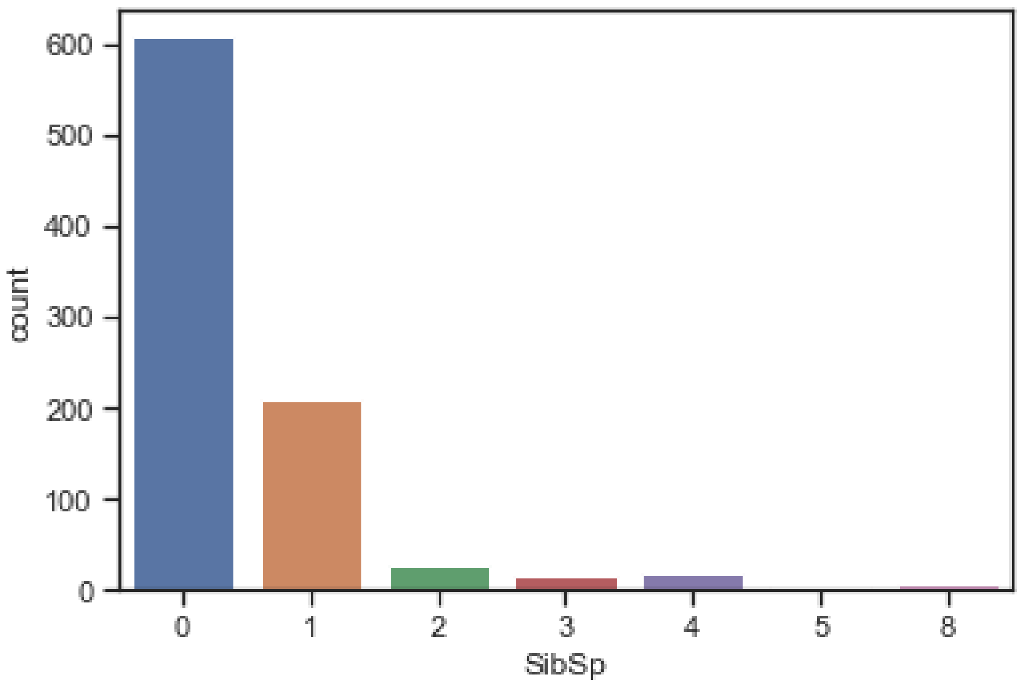
[16]:
```
74.00     1
34.50     1
0.42      1
0.67      1
66.00     1
         ..
28.00    25
19.00    25
18.00    26
22.00    27
24.00    30
Name: Age, Length: 88, dtype: int64
```

20 岁-40 岁的人比较多

SibSp

[17]:
```
sns.set_theme(style="ticks")
sns.countplot(x=train.SibSp,data=train)
train.SibSp.value_counts()
```

[17]:
```
0    608
1    209
2     28
4     18
3     16
8      7
5      5
Name: SibSp, dtype: int64
```



[18]:
```
train.SibSp.value_counts()/train.SibSp.value_counts().sum()
```

[18]:
```
0    0.682379
1    0.234568
2    0.031425
4    0.020202
3    0.017957
8    0.007856
5    0.005612
```

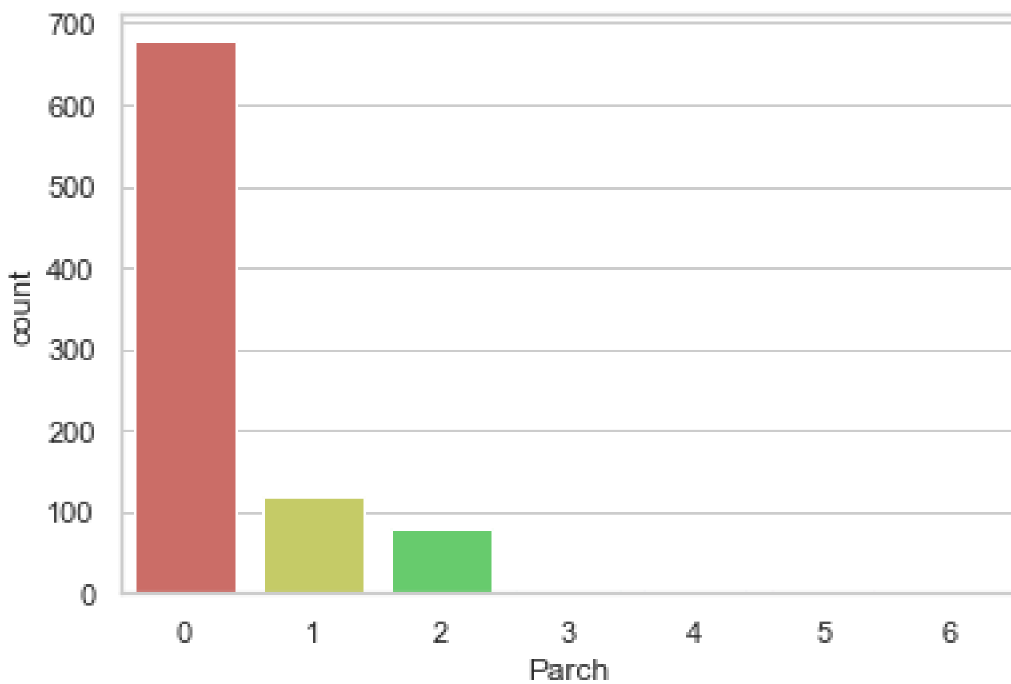
```
Name: SibSp, dtype: float64
```

68% 的人没有同级亲属，23% 的人有一个同级亲属，同级亲属有两个以上的很少

Parch

[19]:
```
sns.set_theme(style="whitegrid")
sns.countplot(x=train.Parch,data=train,palette=sns.color_palette("hls", 6))
train.Parch.value_counts()
```

[19]:
```
0    678
1    118
2     80
5      5
3      5
4      4
6      1
Name: Parch, dtype: int64
```



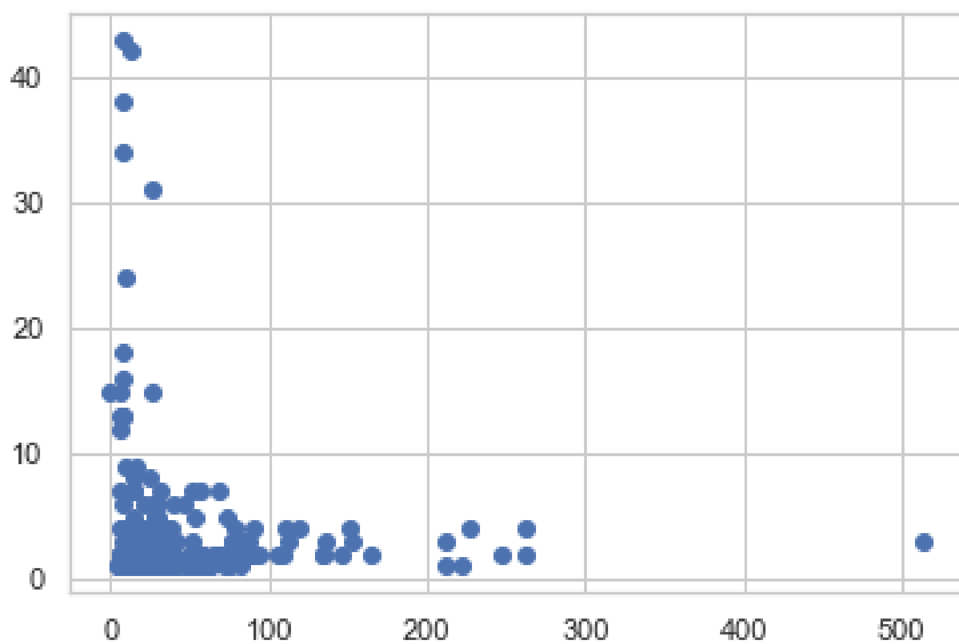
[20]:
```
train.Parch.value_counts()/train.Parch.value_counts().sum()
```

[20]:
```
0    0.760943
1    0.132435
2    0.089787
5    0.005612
3    0.005612
4    0.004489
6    0.001122
Name: Parch, dtype: float64
```

76.1% 的人没有非同级亲属，13% 的人有一个非同级亲属

Fare

[21]:
```
plt.scatter(train.Fare.value_counts().index,train.Fare.value_counts())
plt.show()
```

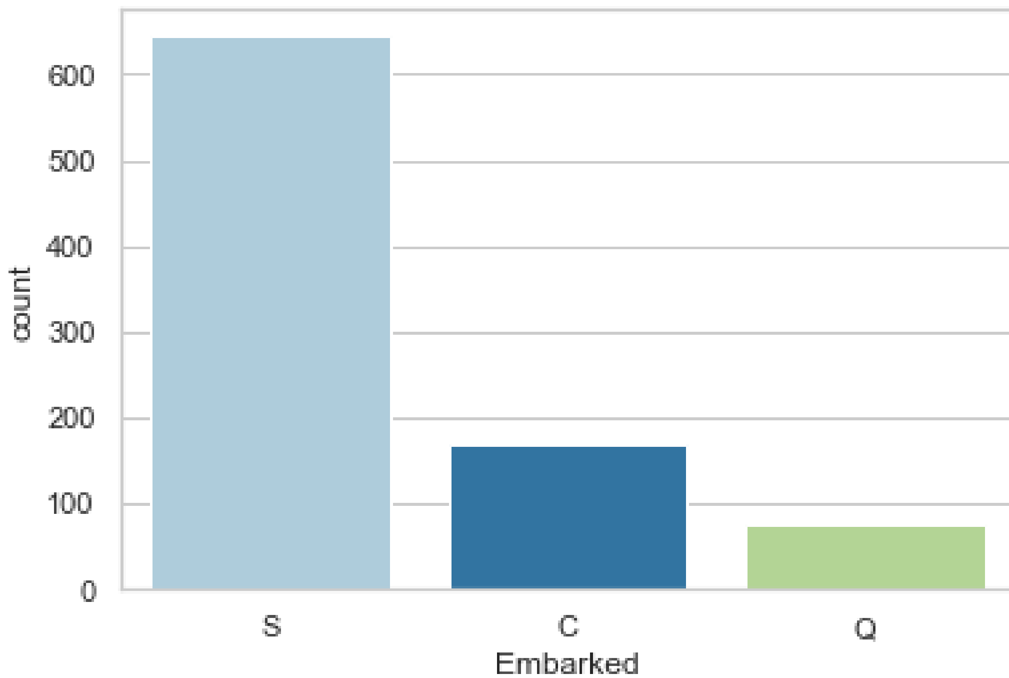


大部分的船票在 100 美元以下

Embarked

[22]:
```
sns.set_theme(style="whitegrid")
sns.countplot(x=train.Embarked,data=train,palette=sns.color_palette("Paired",3))
train.Embarked.value_counts()
```

[22]:
```
S    644
C    168
Q     77
Name: Embarked, dtype: int64
```



[23]:
```
train.Embarked.value_counts()/train.Embarked.value_counts().sum()
```

[23]:
```
S    0.724409
C    0.188976
Q    0.086614
Name: Embarked, dtype: float64
```

72% 的人从 S 上船，18% 从 C 上船，只有 8% 的人从 Q 上船

3.4 数据可视化（数据再探）

3.4.1 数据总体概览

[24]:
```
sns.pairplot(train.drop('PassengerId',axis=1))
```

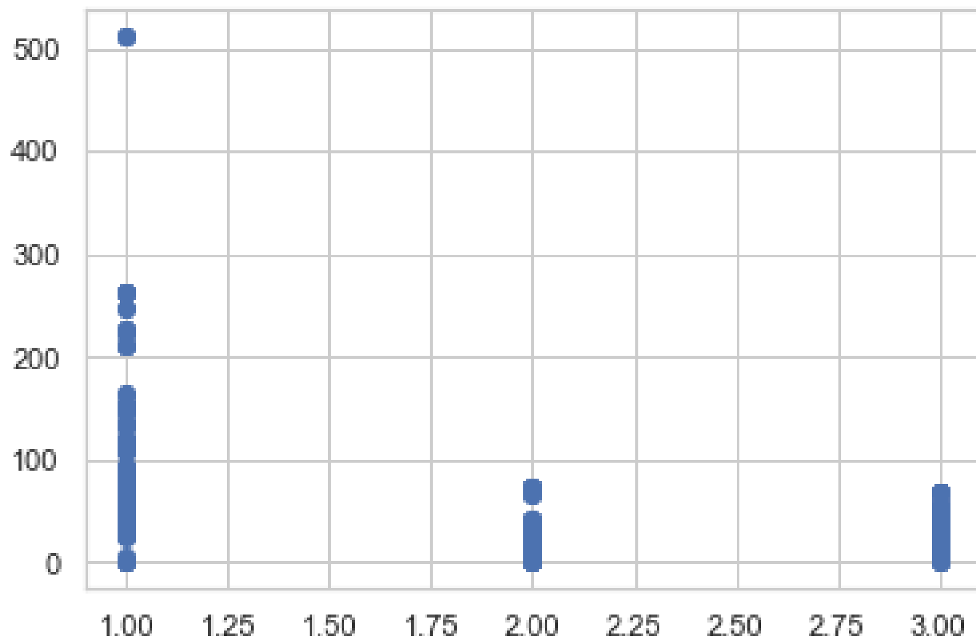
[24]:
```
<seaborn.axisgrid.PairGrid at 0x127a50c8>
```

3.4.2 二维数据

Fare 与 Pclass

[25]:
```
plt.scatter(train.Pclass,train.Fare)
train['Fare'].corr(train['Pclass'])
```

[25]: -0.5494996199439078

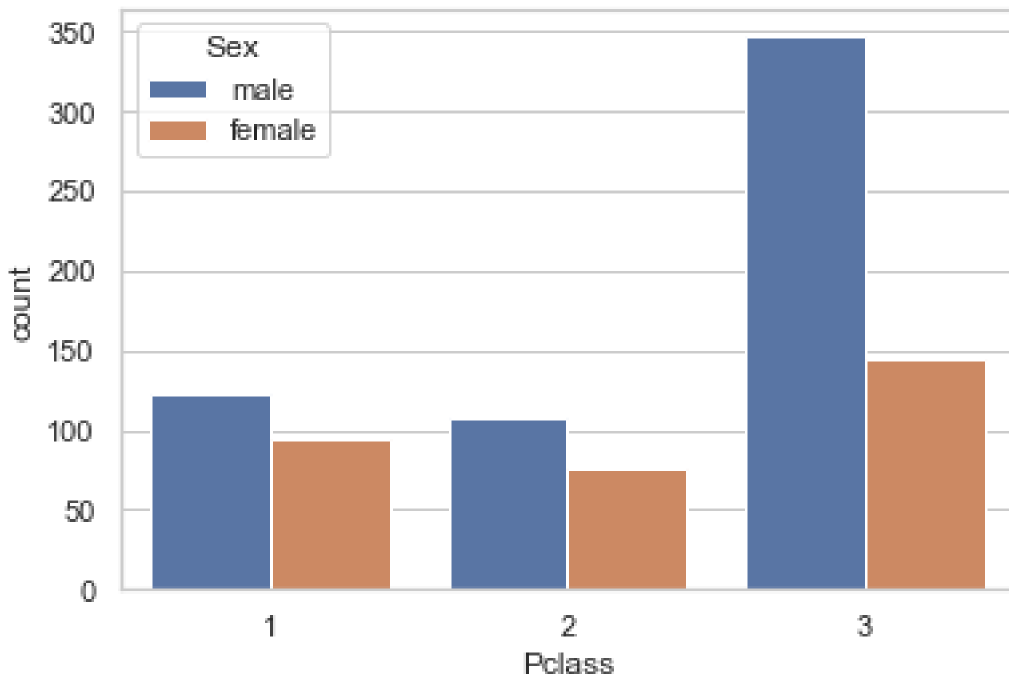


费用与船舱等级具有较高的相关性，费用越多，船舱等级可能越高

Sex 与 Pclass

[26]:
```
sns.countplot(x=train.Pclass, hue=train.Sex, data=train)
```

[26]: <AxesSubplot:xlabel='Pclass', ylabel='count'>



各个船舱，男性均多余女性

Pclass 与 Survived

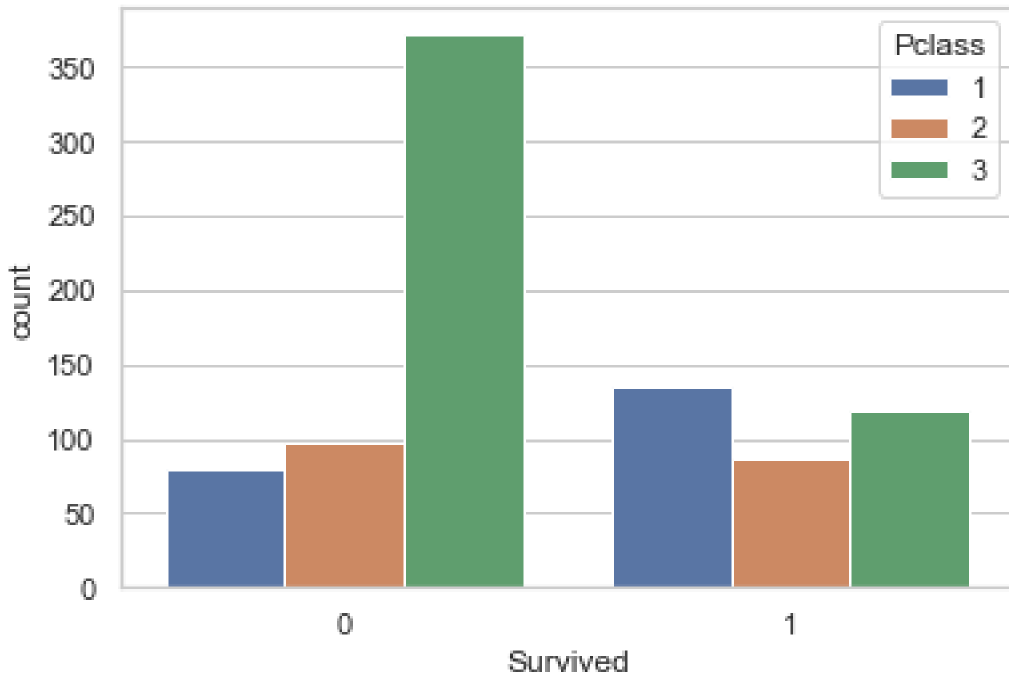
[27]:
```
train[['Pclass', 'Survived']].groupby(['Pclass'], as_index=False).mean().
↪sort_values(by='Survived', ascending=False)
```

[27]:

	Pclass	Survived
0	1	0.629630
1	2	0.472826
2	3	0.242363

[28]:
```
sns.countplot(x=train.Survived, hue=train.Pclass, data=train)
train['Survived'].corr(train['Pclass'])
```

[28]: -0.33848103596101536



一等舱的幸存率最高，三等舱幸存率最低，幸存率与船舱有重要关系

Sex 与 Survived

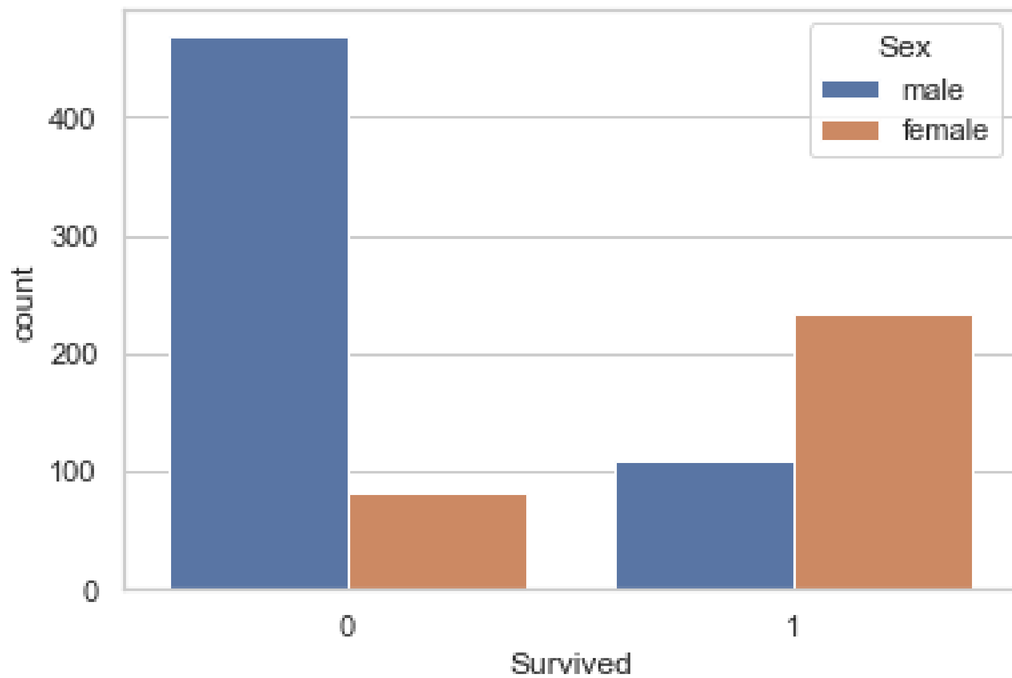
[29]:
```
train[["Sex", "Survived"]].groupby(['Sex'], as_index=False).mean().
    ↪sort_values(by='Survived', ascending=False)
```

[29]:

	Sex	Survived
0	female	0.742038
1	male	0.188908

[30]:
```
sns.countplot(x=train.Survived, hue=train.Sex, data=train)
```

[30]: <AxesSubplot:xlabel='Survived', ylabel='count'>

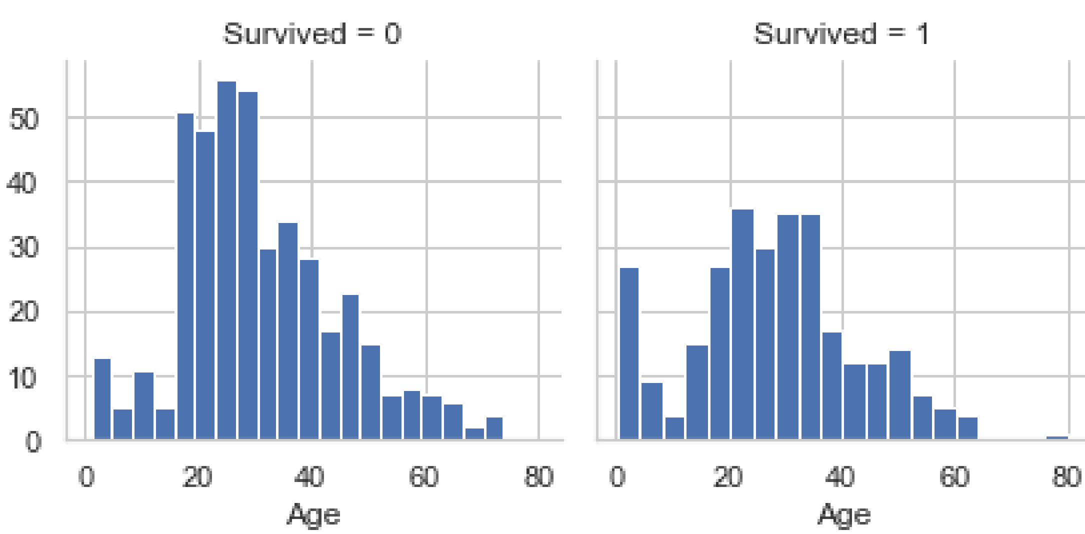


女性的生存率远高于男性，可以推断发生灾难时，许多男性把生存机会留给了女性

Age 与 Survived

[31]:
```
g = sns.FacetGrid(train, col='Survived')
g.map(plt.hist, 'Age', bins=20)
```

[31]: <seaborn.axisgrid.FacetGrid at 0x146949c8>



孩子和老人的生存率明显高于死亡率，而成年的死亡率明显高于生存率，可以推断发生灾难时，许多成年把生存机会留给了孩子和老人

SibSp 与 Survived

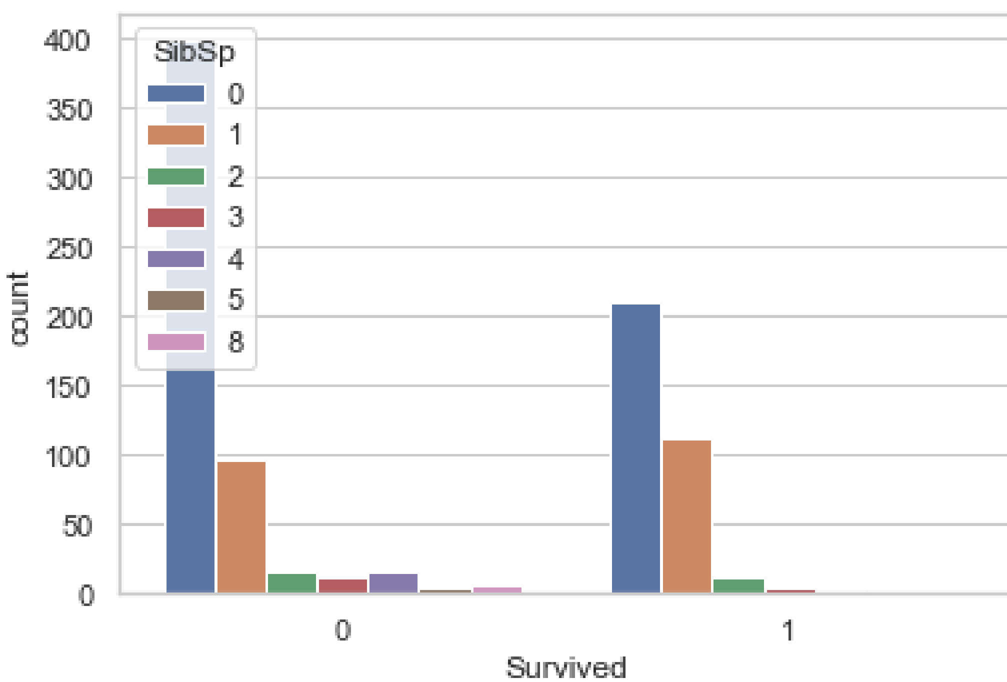
[32]:
```
train[["SibSp", "Survived"]].groupby(['SibSp'], as_index=False).mean().
↪sort_values(by='Survived', ascending=False)
```

[32]:

	SibSp	Survived
1	1	0.535885
2	2	0.464286
0	0	0.345395
3	3	0.250000
4	4	0.166667
5	5	0.000000
6	8	0.000000

[33]:
```
sns.countplot(x=train.Survived, hue=train.SibSp, data=train)
train['Survived'].corr(train['SibSp'])
```

[33]: -0.035322498885735559



相关性较弱

Parch 与 Survived

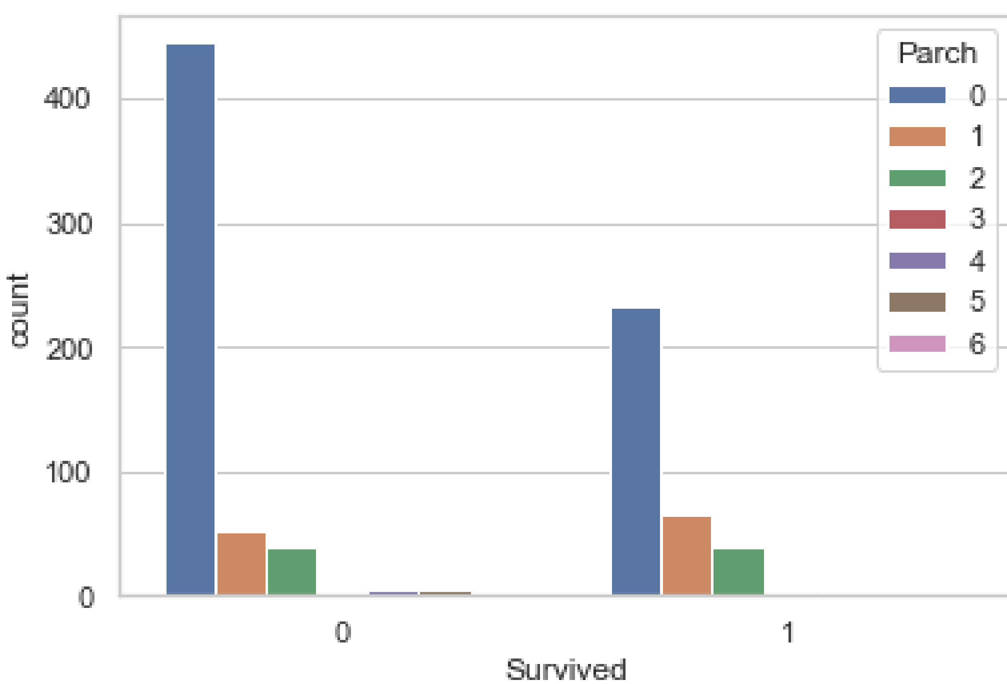
[34]:
```
train[["Parch", "Survived"]].groupby(['Parch'], as_index=False).mean().
 ↪sort_values(by='Survived', ascending=False)
```

[34]:

	Parch	Survived
3	3	0.600000
1	1	0.550847
2	2	0.500000
0	0	0.343658
5	5	0.200000
4	4	0.000000
6	6	0.000000

[35]:
```
sns.countplot(x=train.Survived, hue=train.Parch, data=train)
train['Survived'].corr(train['Parch'])
```

[35]: 0.08162940708348349

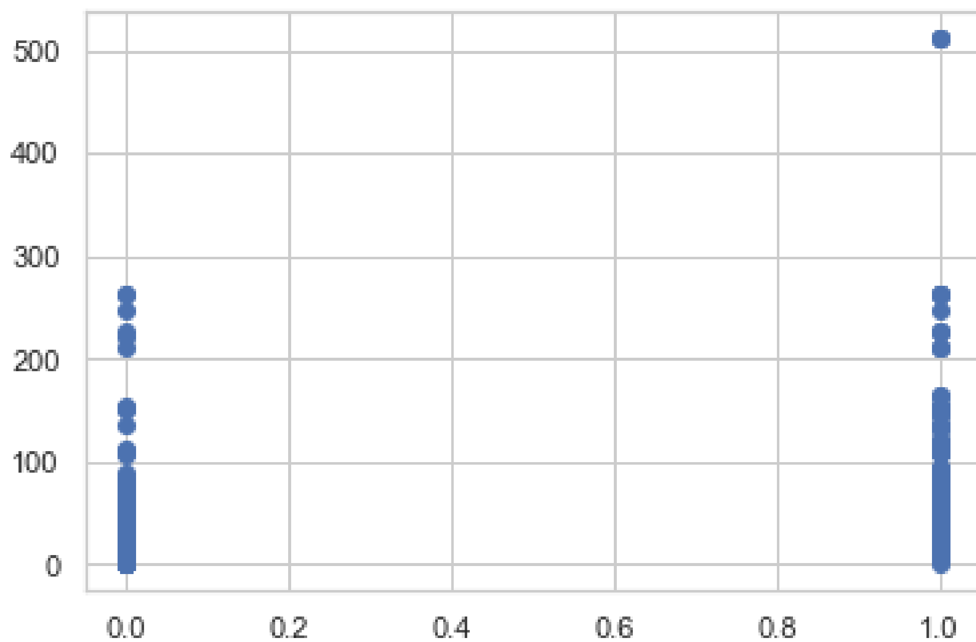


相关性较弱

Fare 与 Survived

[36]:
```
plt.scatter(train.Survived,train.Fare)
train['Survived'].corr(train['Fare'])
```

[36]:
```
0.2573065223849624
```



[37]:
```
train.groupby('Survived')['Fare'].mean()
```

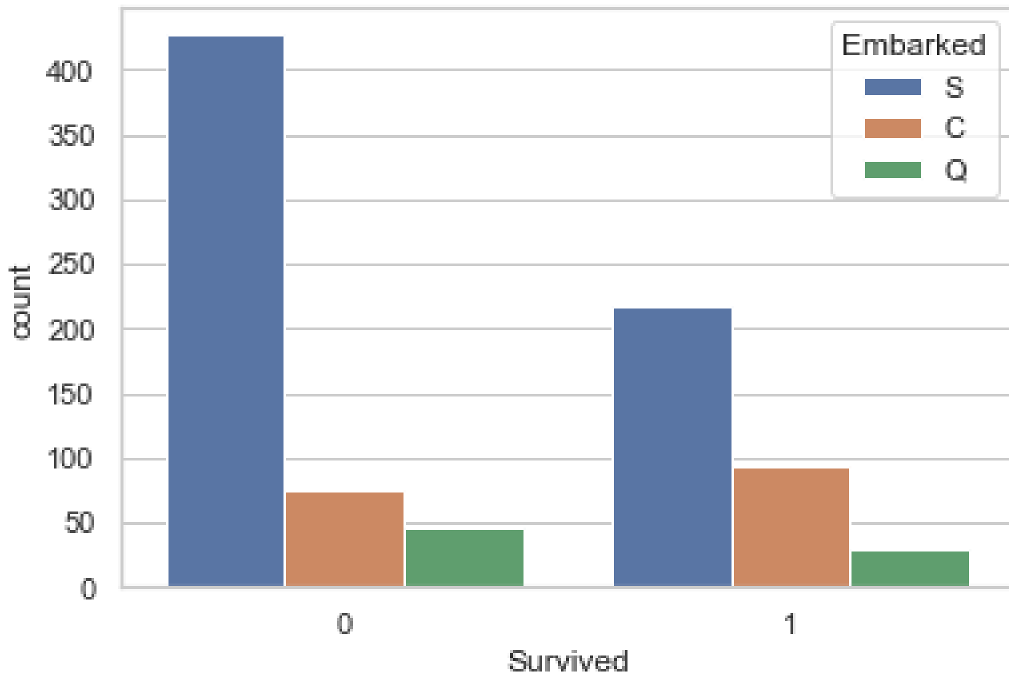
[37]:
```
Survived
0    22.117887
1    48.395408
Name: Fare, dtype: float64
```

船费更高的倾向于更高的的生存率

Embarked 与 Survived

[38]:
```
sns.countplot(x=train.Survived, hue=train.Embarked, data=train)
```

[38]: <AxesSubplot:xlabel='Survived', ylabel='count'>



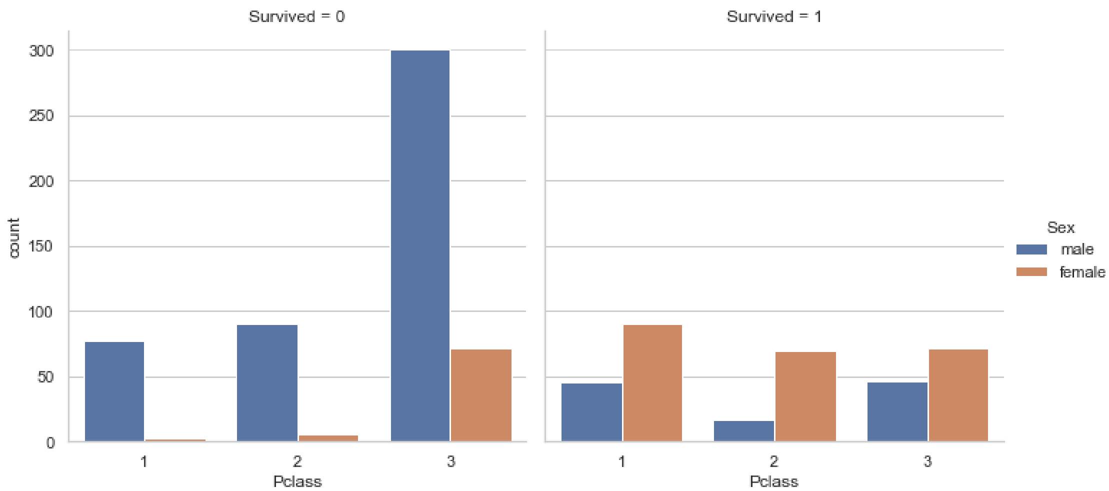
相关性较弱

3.4.3 三维数据

Sex 与 Survived,Pclass

[39]:
```
sns.catplot(x="Pclass",
            hue="Sex",
            col="Survived",
            data=train,
            kind="count")
```

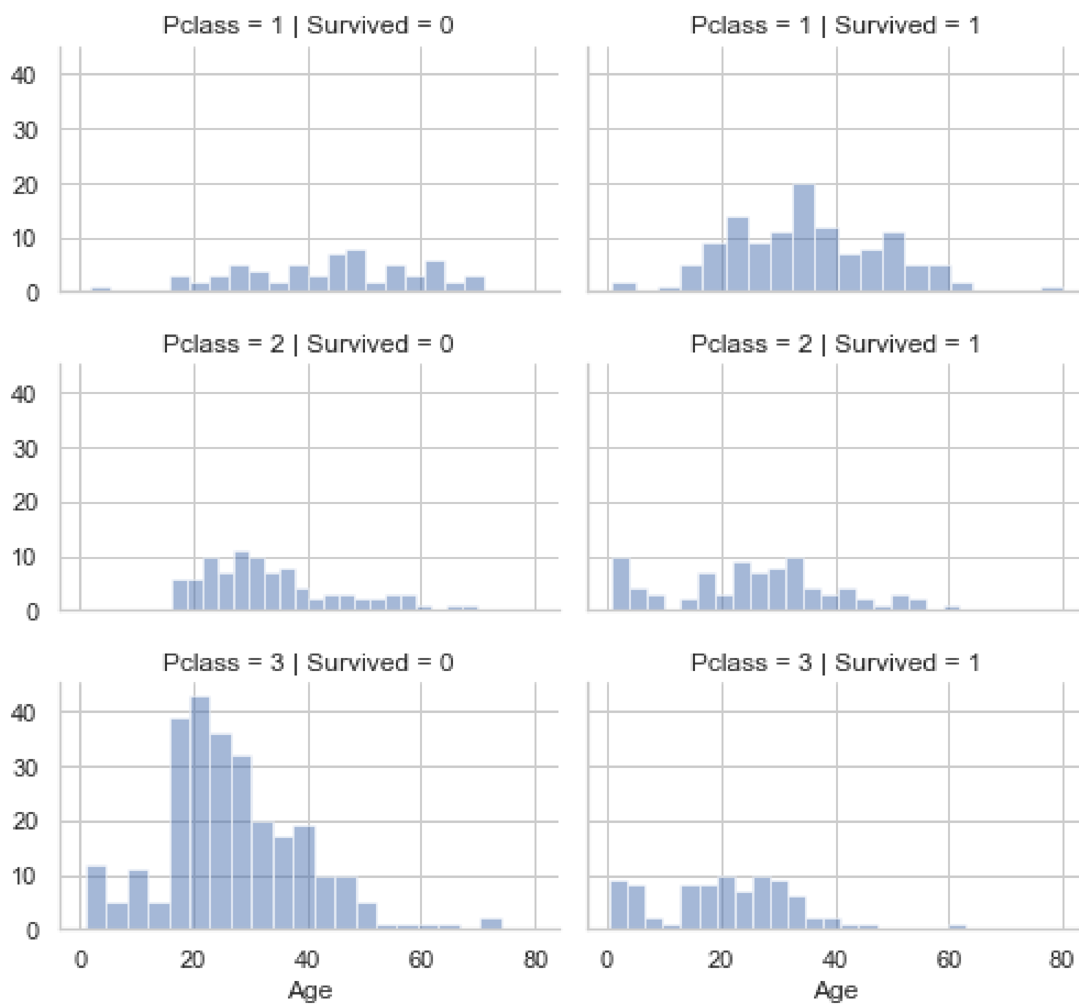
[39]: <seaborn.axisgrid.FacetGrid at 0x14d5cd48>



各个等级船舱，女性生存率高于男性; 高等级船舱生存率更高

Age 与 Survived,Pclass

[40]:
```
grid = sns.FacetGrid(train, col='Survived', row='Pclass', height=2.2, aspect=1.
↪6)
grid.map(plt.hist, 'Age', alpha=.5, bins=20)
grid.add_legend();
```

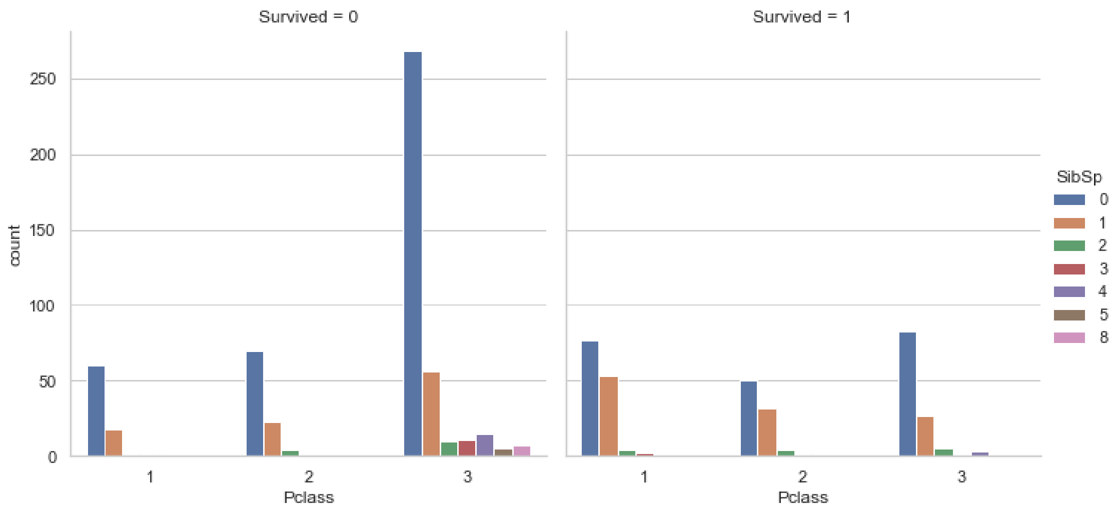


船舱一，死亡数较少；船舱二，死亡数中等；船舱三，孩子死亡少，成年死亡多

SibSp 与 Survived,Pclass

[41]:
```
sns.catplot(x="Pclass",
            hue="SibSp",
            col="Survived",
            data=train,
            kind="count")
```

[41]: <seaborn.axisgrid.FacetGrid at 0x150fdbc8>

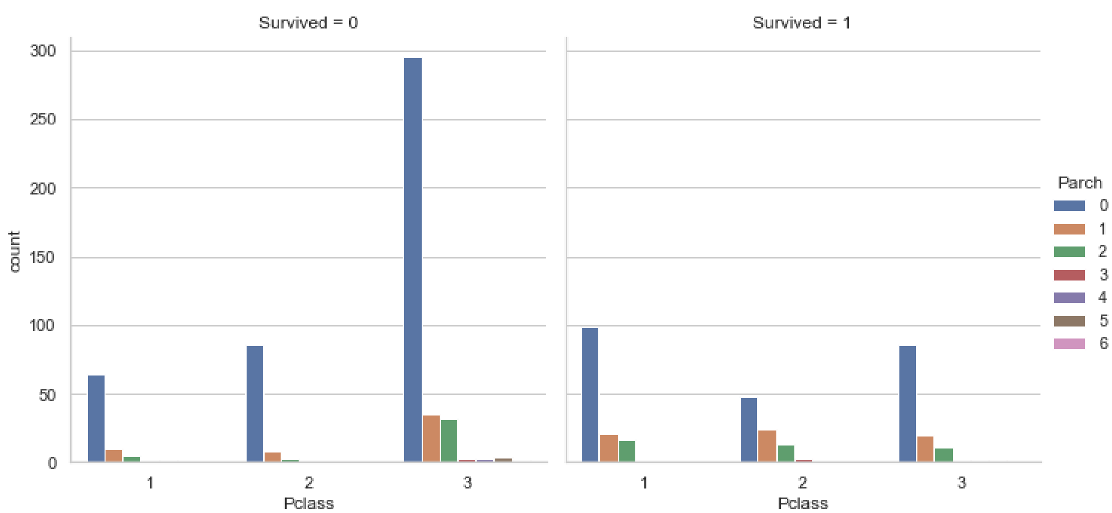


相关性较弱

Parch 与 Survived,Pclass

[42]:
```
sns.catplot(x="Pclass",
            hue="Parch",
            col="Survived",
            data=train,
            kind="count")
```

[42]: <seaborn.axisgrid.FacetGrid at 0x161d8d08>

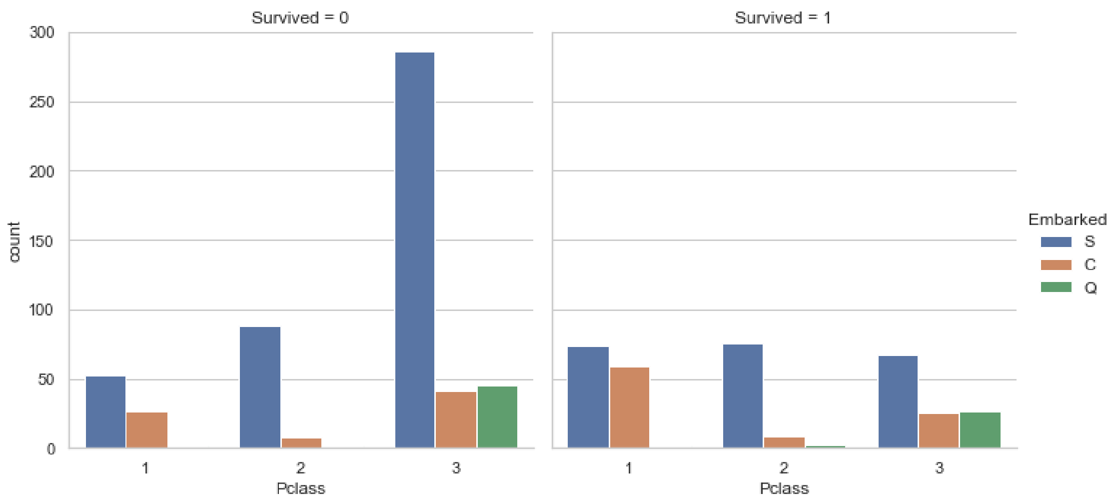


相关性较弱

Embarked 与 Survived,Pclass

[43]:
```
sns.catplot(x="Pclass",
            hue="Embarked",
            col="Survived",
            data=train,
            kind="count")
```

[43]: <seaborn.axisgrid.FacetGrid at 0x1632ac88>



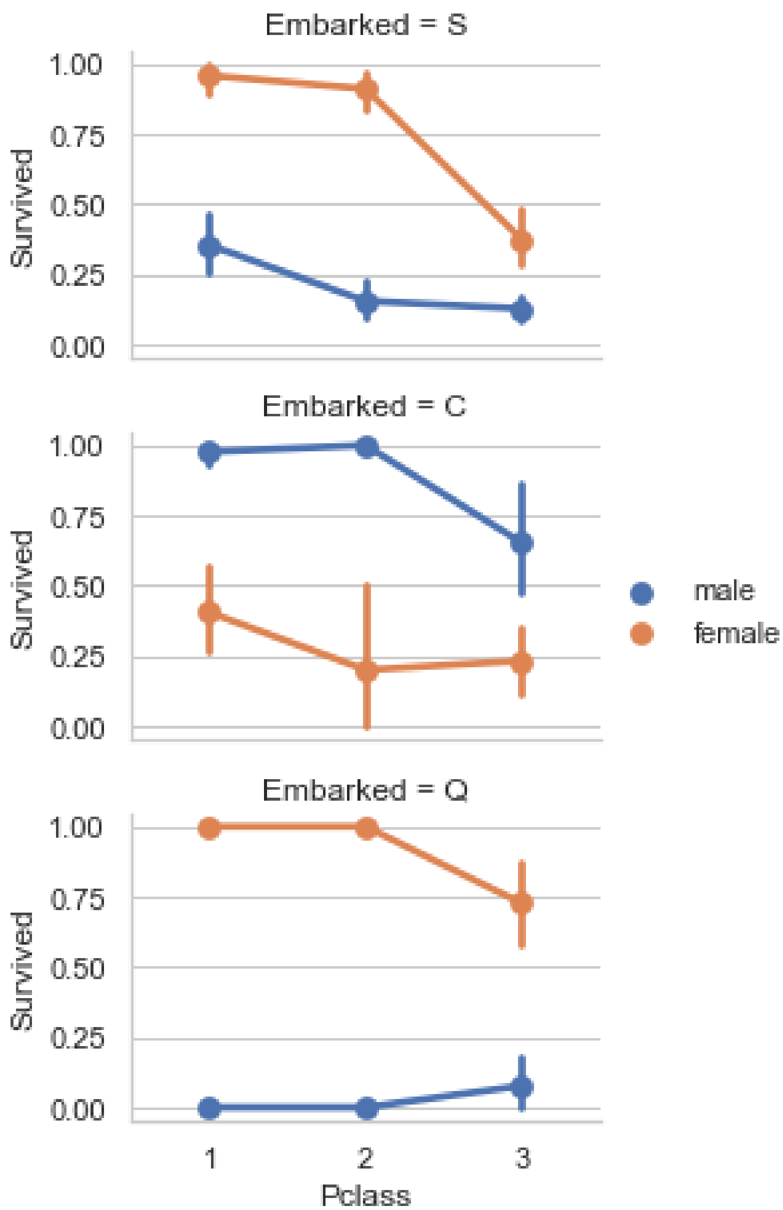
相关性较弱

3.4.4 四维数据

Embarked,Sex,Pclass 与 Survived

[44]:
```
grid = sns.FacetGrid(train, row='Embarked', height=2.2, aspect=1.6)
grid.map(sns.pointplot, 'Pclass', 'Survived', 'Sex', palette='deep')
grid.add_legend()
```

[44]: <seaborn.axisgrid.FacetGrid at 0x165af808>

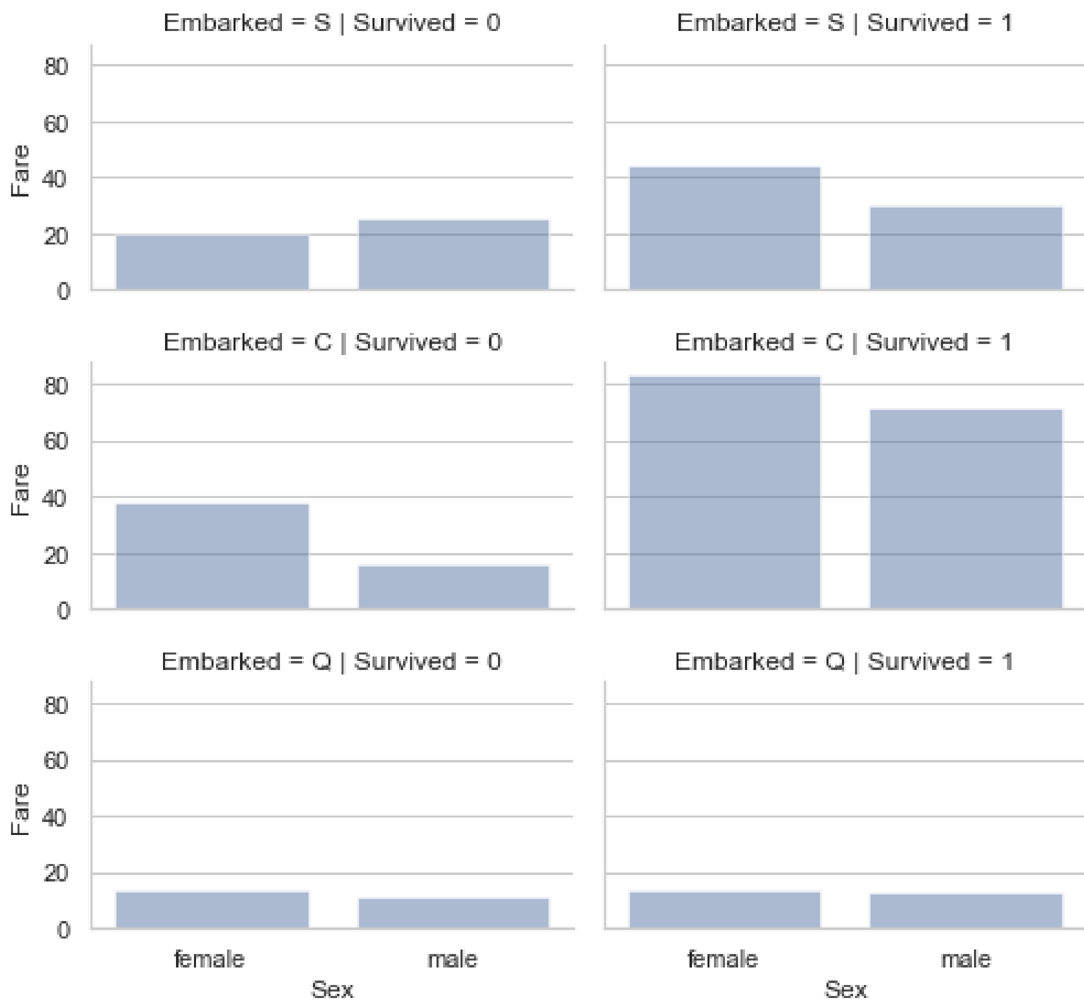


S 和 Q 上岸的人，女性生存率高于男性；一等舱，二等舱的生存率比三等舱高相关性较弱

Fare,Sex,Embarked 与 Survived

[45]:
```
grid = sns.FacetGrid(train, row='Embarked', col='Survived', size=2.2, aspect=1.
 ↪6)
grid.map(sns.barplot, 'Sex', 'Fare', alpha=.5, ci=None)
grid.add_legend()
```

[45]: <seaborn.axisgrid.FacetGrid at 0x16931cc8>



幸存的人倾向于更高的船费；Q 上岸的人倾向于更高的船费；男性倾向于更高的船费

3.5 特征工程

3.5.1 数据预处理

剔除 ticket,PassengerId,cabin ticket,PassengerId 信息无用，cabin 缺失值过多，将二者剔除

[46]:
```
(train.shape,test.shape)
```

[46]: ((891, 12), (418, 11))

[47]:
```python
train = train.drop(['Ticket', 'Cabin','PassengerId'], axis=1)
test = test.drop(['Ticket', 'Cabin','PassengerId'], axis=1)            # 剔除数据
```

[48]:
```python
(train.shape,test.shape)
```

[48]:
```
((891, 9), (418, 8))
```

Sex 装换为值

[49]:
```python
for i in range(train.shape[0]):
    if train['Sex'][i]=='male':
        train['Sex'][i]=0.5
    elif train['Sex'][i]=='female':
        train['Sex'][i]=-0.5

for i in range(test.shape[0]):
    if test['Sex'][i]=='male':
        test['Sex'][i]=0.5
    elif test['Sex'][i]=='female':
        test['Sex'][i]=-0.5

train['Sex']=train['Sex'].astype(float)
test['Sex']=test['Sex'].astype(float)
```

[50]:
```python
train['Sex']
```

[50]:
```
0       0.5
1      -0.5
2      -0.5
3      -0.5
4       0.5
       …
886     0.5
887    -0.5
888    -0.5
889     0.5
890     0.5
Name: Sex, Length: 891, dtype: float64
```

Embarked 缺失值填充并装换为值　考虑到 Embarked 只缺失了两个数据，考虑人工填充相邻值

[51]:
```
train[train['Embarked'].isna()]
```

[51]:
```
     Survived  Pclass                                       Name  Sex   Age  \
61          1       1                        Icard, Miss. Amelie -0.5  38.0
829         1       1  Stone, Mrs. George Nelson (Martha Evelyn) -0.5  62.0

     SibSp  Parch  Fare Embarked
61       0      0  80.0      NaN
829      0      0  80.0      NaN
```

[52]:
```
train[train['Survived']==1][train['Pclass']==1][train['Sex']==-0.
 ↪5][train['SibSp']==0][train['Parch']==0]  # 选出 Survived, Pclass, Sex, SibSp␣
 ↪parch 与缺失数据相同的行
```

[52]:
```
     Survived  Pclass                                         Name  Sex  \
11          1       1                     Bonnell, Miss. Elizabeth -0.5
61          1       1                          Icard, Miss. Amelie -0.5
194         1       1  Brown, Mrs. James Joseph (Margaret Tobin) -0.5
195         1       1                         Lurette, Miss. Elise -0.5
218         1       1                        Bazzani, Miss. Albina -0.5
256         1       1                Thorne, Mrs. Gertrude Maybelle -0.5
257         1       1                        Cherry, Miss. Gladys -0.5
258         1       1                            Ward, Miss. Anna -0.5
269         1       1                      Bissette, Miss. Amelia -0.5
290         1       1                 Barber, Miss. Ellen "Nellie" -0.5
306         1       1                      Fleming, Miss. Margaret -0.5
309         1       1               Francatelli, Miss. Laura Mabel -0.5
310         1       1              Hays, Miss. Margaret Bechstein -0.5
325         1       1                     Young, Miss. Marie Grice -0.5
337         1       1              Burns, Miss. Elizabeth Margaret -0.5
369         1       1                 Aubart, Mme. Leontine Pauline -0.5
380         1       1                       Bidois, Miss. Rosalie -0.5
504         1       1                       Maioni, Miss. Roberta -0.5
520         1       1                        Perreault, Miss. Anne -0.5
537         1       1                        LeRoy, Miss. Bertha -0.5
609         1       1                   Shutes, Miss. Elizabeth W -0.5
```

627	1	1	Longley, Miss. Gretchen Fiske	-0.5
641	1	1	Sagesser, Mlle. Emma	-0.5
708	1	1	Cleaver, Miss. Alice	-0.5
710	1	1	Mayne, Mlle. Berthe Antonine ("Mrs de Villiers")	-0.5
716	1	1	Endres, Miss. Caroline Louise	-0.5
730	1	1	Allen, Miss. Elisabeth Walton	-0.5
759	1	1	Rothes, the Countess. of (Lucy Noel Martha Dye…	-0.5
796	1	1	Leader, Dr. Alice (Farnham)	-0.5
829	1	1	Stone, Mrs. George Nelson (Martha Evelyn)	-0.5
842	1	1	Serepeca, Miss. Augusta	-0.5
862	1	1	Swift, Mrs. Frederick Joel (Margaret Welles Ba…	-0.5
887	1	1	Graham, Miss. Margaret Edith	-0.5

	Age	SibSp	Parch	Fare	Embarked
11	58.0	0	0	26.5500	S
61	38.0	0	0	80.0000	NaN
194	44.0	0	0	27.7208	C
195	58.0	0	0	146.5208	C
218	32.0	0	0	76.2917	C
256	NaN	0	0	79.2000	C
257	30.0	0	0	86.5000	S
258	35.0	0	0	512.3292	C
269	35.0	0	0	135.6333	S
290	26.0	0	0	78.8500	S
306	NaN	0	0	110.8833	C
309	30.0	0	0	56.9292	C
310	24.0	0	0	83.1583	C
325	36.0	0	0	135.6333	C
337	41.0	0	0	134.5000	C
369	24.0	0	0	69.3000	C
380	42.0	0	0	227.5250	C
504	16.0	0	0	86.5000	S
520	30.0	0	0	93.5000	S
537	30.0	0	0	106.4250	C
609	40.0	0	0	153.4625	S
627	21.0	0	0	77.9583	S

```
641  24.0     0     0   69.3000       C
708  22.0     0     0  151.5500       S
710  24.0     0     0   49.5042       C
716  38.0     0     0  227.5250       C
730  29.0     0     0  211.3375       S
759  33.0     0     0   86.5000       S
796  49.0     0     0   25.9292       S
829  62.0     0     0   80.0000     NaN
842  30.0     0     0   31.0000       C
862  48.0     0     0   25.9292       S
887  19.0     0     0   30.0000       S
```

[53]:
```python
train[train['Survived']==1][train['Pclass']==1][train['Sex']==-0.
↪5][train['SibSp']==0][train['Parch']==0]['Embarked'].value_counts() # 查看计数
```

[53]:
```
C    17
S    14
Name: Embarked, dtype: int64
```

[54]:
```python
train['Embarked'].fillna('C' ,inplace=True)     # 用 C 填充
```

[55]:
```python
train[train['Embarked'].isna()]
```

[55]:
```
Empty DataFrame
Columns: [Survived, Pclass, Name, Sex, Age, SibSp, Parch, Fare, Embarked]
Index: []
```

下面将 Embarked 装换为数值

[56]:
```python
train['Embarked'].value_counts()
```

[56]:
```
S    644
C    170
Q     77
Name: Embarked, dtype: int64
```

[57]:
```python
# 将数据集中的 C,S,Q 替换为数值
for i in range(train.shape[0]):
    if train['Embarked'][i]=='C':
```

```
        train['Embarked'][i]=-0.4
    elif train['Embarked'][i]=='S':
        train['Embarked'][i]=0.1
    elif train['Embarked'][i]=='Q':
        train['Embarked'][i]=0.6

for i in range(test.shape[0]):
    if test['Embarked'][i]=='C':
        test['Embarked'][i]=-0.4
    elif test['Embarked'][i]=='S':
        test['Embarked'][i]=0.1
    elif test['Embarked'][i]=='Q':
        test['Embarked'][i]=0.6

train['Embarked']=train['Embarked'].astype(float)
test['Embarked']=test['Embarked'].astype(float)
```

[58]:
```
train['Embarked']
```

[58]:
```
0       0.1
1      -0.4
2       0.1
3       0.1
4       0.1
       …
886     0.1
887     0.1
888     0.1
889    -0.4
890     0.6
Name: Embarked, Length: 891, dtype: float64
```

SibSp 与 Parch SibSp 与 Parch 表示的都是亲属，将其合并

[59]:
```
# 亲属包括自己，加 1
train['family']=train['SibSp']+train['Parch']+1
test['family']=test['SibSp']+test['Parch']+1
```

[60]:
```python
# 删除 SibSp, Parch
train = train.drop(['SibSp', 'Parch'], axis=1)
test = test.drop(['SibSp', 'Parch'], axis=1)
combine = [train, test]
```

[61]:
```python
test
```

[61]:
```
     Pclass                                          Name  Sex   Age  \
0         3                              Kelly, Mr. James  0.5  34.5
1         3              Wilkes, Mrs. James (Ellen Needs) -0.5  47.0
2         2                     Myles, Mr. Thomas Francis  0.5  62.0
3         3                              Wirz, Mr. Albert  0.5  27.0
4         3  Hirvonen, Mrs. Alexander (Helga E Lindqvist) -0.5  22.0
..      …                                           …  …   …
413       3                            Spector, Mr. Woolf  0.5   NaN
414       1                  Oliva y Ocana, Dona. Fermina -0.5  39.0
415       3                  Saether, Mr. Simon Sivertsen  0.5  38.5
416       3                           Ware, Mr. Frederick  0.5   NaN
417       3                     Peter, Master. Michael J  0.5   NaN

         Fare  Embarked  family
0      7.8292       0.6       1
1      7.0000       0.1       2
2      9.6875       0.6       1
3      8.6625       0.1       1
4     12.2875       0.1       3
..        …       …     …
413    8.0500       0.1       1
414  108.9000      -0.4       1
415    7.2500       0.1       1
416    8.0500       0.1       1
417   22.3583      -0.4       3

[418 rows x 7 columns]
```

Name 信息提取

[62]:
```python
# 获取称呼
for dataset in combine:
    dataset['call'] = dataset.Name.str.extract(' ([A-Za-z]+)\.', expand=False)

pd.crosstab(train['call'], train['Sex'])
```

[62]:

Sex	-0.5	0.5
call		
Capt	0	1
Col	0	2
Countess	1	0
Don	0	1
Dr	1	6
Jonkheer	0	1
Lady	1	0
Major	0	2
Master	0	40
Miss	182	0
Mlle	2	0
Mme	1	0
Mr	0	517
Mrs	125	0
Ms	1	0
Rev	0	6
Sir	0	1

[63]:
```python
# 将称呼统一归类并查看比例
for dataset in combine:
    dataset['call'] = dataset['call'].replace(['Lady', 'Countess','Capt',␣
 ↪'Col','Don', 'Dr', 'Major', 'Rev', 'Sir', 'Jonkheer', 'Dona'], 'other')
    dataset['call'] = dataset['call'].replace('Mlle', 'Miss')
    dataset['call'] = dataset['call'].replace('Ms', 'Miss')
    dataset['call'] = dataset['call'].replace('Mme', 'Mrs')

train[['call', 'Survived']].groupby(['call'], as_index=False).mean()
```

[63]:
```
     call  Survived
0  Master  0.575000
1    Miss  0.702703
2      Mr  0.156673
3     Mrs  0.793651
4   other  0.347826
```

[64]:
```
# 用数值替换称呼
call_mapping = {"Mr": 1, "Miss": 2, "Mrs": 3, "Master": 4, "other": 5}
for dataset in combine:
    dataset['call'] = dataset['call'].map(call_mapping)
    dataset['call'] = dataset['call'].fillna(0)

train.head()
```

[64]:
```
   Survived  Pclass                                               Name  Sex  \
0         0       3                            Braund, Mr. Owen Harris  0.5
1         1       1  Cumings, Mrs. John Bradley (Florence Briggs Th… -0.5
2         1       3                             Heikkinen, Miss. Laina -0.5
3         1       1       Futrelle, Mrs. Jacques Heath (Lily May Peel) -0.5
4         0       3                           Allen, Mr. William Henry  0.5

    Age     Fare  Embarked  family  call
0  22.0   7.2500       0.1       2     1
1  38.0  71.2833      -0.4       2     3
2  26.0   7.9250       0.1       1     2
3  35.0  53.1000       0.1       2     3
4  35.0   8.0500       0.1       1     1
```

[65]:
```
# 剔除 Name
train = train.drop(['Name'], axis=1)
test = test.drop(['Name'], axis=1)
combine = [train, test]
```

[66]:
```
train.head(3)
```

[66]:
```
   Survived  Pclass  Sex   Age    Fare  Embarked  family  call
0         0       3  0.5  22.0  7.2500       0.1       2     1
```

```
1           1        1  -0.5   38.0   71.2833       -0.4        2      3
2           1        3  -0.5   26.0    7.9250        0.1        1      2
```

3.5.2 缺失值处理

Fare 的填充及其归一化

[67]:
```python
test[test['Fare'].isnull()]
```

[67]:
```
     Pclass  Sex   Age  Fare  Embarked  family  call
152       3  0.5  60.5   NaN       0.1       1     1
```

[68]:
```python
# 用中位数进行填充
test['Fare'].fillna(test['Fare'].dropna().median(), inplace=True)
```

[69]:
```python
test[test['Fare'].isnull()]
```

[69]:
```
Empty DataFrame
Columns: [Pclass, Sex, Age, Fare, Embarked, family, call]
Index: []
```

[70]:
```python
#Fare 归一化
train['Fare']=(train['Fare']-train['Fare'].min())/train['Fare'].
 ↪max()-train['Fare'].min()
test['Fare']=(test['Fare']-test['Fare'].min())/test['Fare'].max()-test['Fare'].
 ↪min()
test['Fare']
```

[70]:
```
0      0.015282
1      0.013663
2      0.018909
3      0.016908
4      0.023984
         …
413    0.015713
414    0.212559
415    0.014151
416    0.015713
417    0.043640
Name: Fare, Length: 418, dtype: float64
```

age 的预测填充及其归一化 下面使用神经网络对 Age 进行预测

[71]:
```
# 搭建神经网络
class agenet(nn.Module):

    def __init__(self):
        super(agenet, self).__init__()

        self.conv1 = nn.Sequential(
            nn.Linear(6, 10)
        )
        self.conv2 = nn.Sequential(
           nn.Linear(10, 12)
        )
        self.conv3 = nn.Sequential(
           nn.Linear(12, 7)
        )
        self.conv4 = nn.Sequential(
           nn.Linear(7, 4)
        )

        self.outlayer = nn.Linear(4, 1)

    def forward(self, x):
        """

        :param x:
        :return:
        """

        x = F.relu(self.conv1(x))
        x = F.relu(self.conv2(x))
        x = F.relu(self.conv3(x))
        x = F.relu(self.conv4(x))
        x = self.outlayer(x)
        return x
```

生成训练集，测试集

[72]:
```python
# 生成训练集
x=pd.concat(
    [train.dropna()
    .drop('Survived',axis=1)
    .drop('Age',axis=1)
    ,test.dropna()
    .drop('Age',axis=1)]
    ,axis=0
    ,join='outer',ignore_index=True)

# 生成训练集标签
x_label=pd.concat(
    [train.dropna()['Age']
    ,test.dropna()['Age']]
    ,axis=0
    ,join='outer',ignore_index=True)

# 数据转换
x_label=torch.from_numpy(np.array(x_label)).to(torch.float32)
```

[73]:
```python
# 正常 test 测试集
pred=pd.concat(
    [
    train[train.isnull().values==True].drop('Survived',axis=1).
 ↪drop('Age',axis=1)
    ,
    test[test.isnull().values==True].drop('Age',axis=1)
    ]
    ,axis=0
    ,join='outer',ignore_index=True)

# 数据装换
pred=torch.from_numpy(np.array(pred)).to(torch.float32)
```

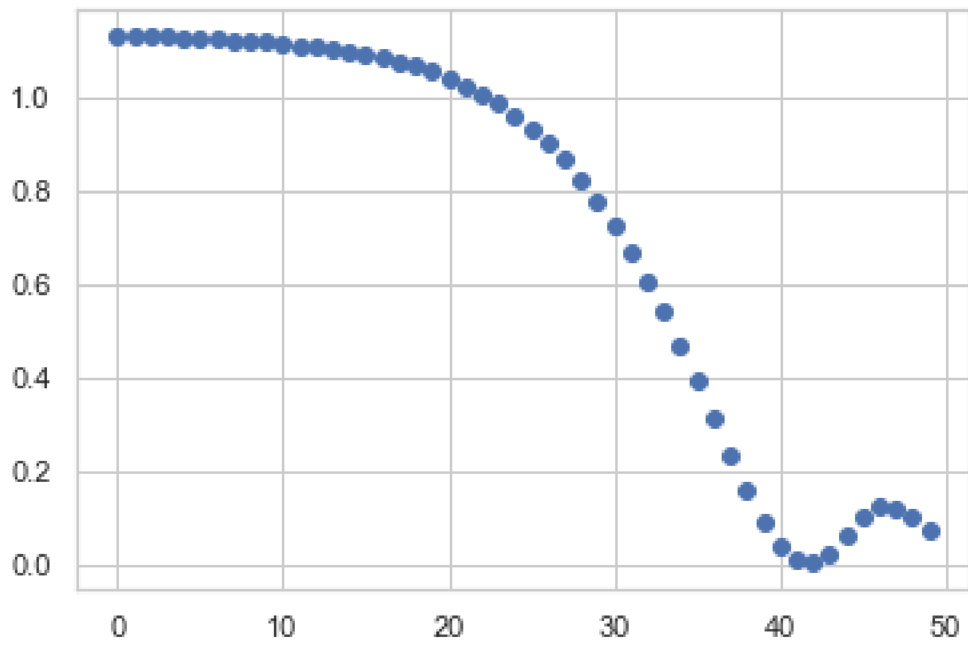
模型训练

[74]:
```
model=agenet()
optimizer = optim.Adam(model.parameters(), lr=1e-2)
train_tensor=torch.from_numpy(np.array(x[0:700])).to(torch.float32)
test_tensor=torch.from_numpy(np.array(x[700:])).to(torch.float32)
losses=[]       # 记录损失值
step=[]         # 记录训练次数
total_loss=0    # 总损失
for i in range(50):
    for j in range(700):
        temp=torch.zeros(1,6)
        temp[0]=train_tensor[j]
        model.train()
        optimizer.zero_grad()      # 剃度清零
        logits = model(temp)
        loss_fn=nn.MSELoss()       # 损失函数
        loss= loss_fn(logits,x_label[j])
        total_loss=+loss
    total_loss/=700
    total_loss.backward()          # 反向传播
    losses.append(total_loss.item())
    step.append(i)
    optimizer.step()    # 调节权重
```

效果可以，似乎有些过拟合

[75]:
```
plt.scatter(step,losses)
```

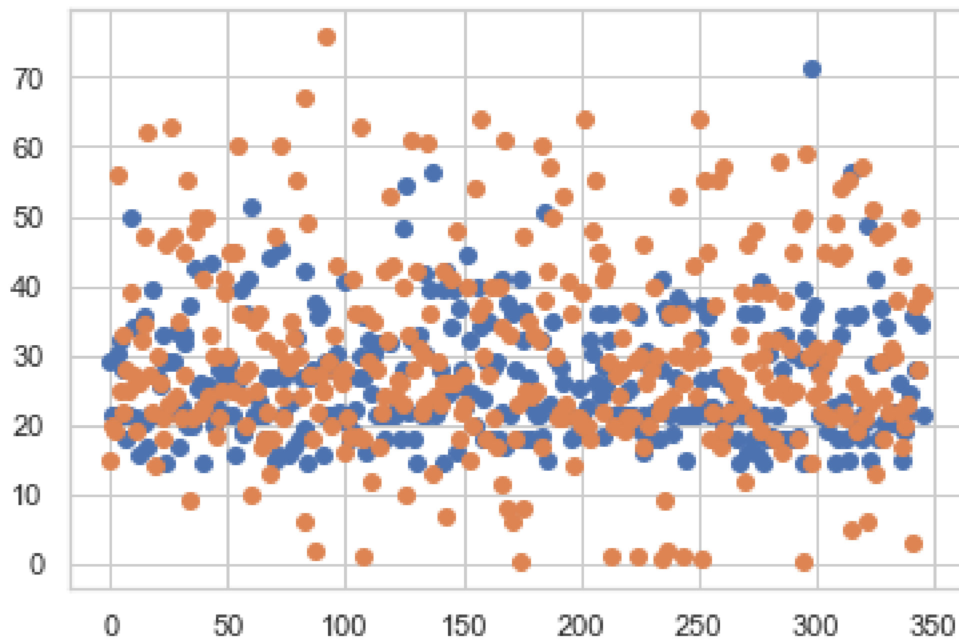
[75]: <matplotlib.collections.PathCollection at 0x150c22c8>



[76]:
```
# 预测测试集数据
predit=[]
for j in range(1046-700):
    temp1=torch.zeros(1,6)
    temp1[0]=test_tensor[j]
    predit.append(model(temp1).detach().numpy() )
```

[77]:
```
# 画图查看效果
plt.scatter(range(1046-700),predit)
plt.scatter(range(1046-700),x_label[700:])
```

[77]: <matplotlib.collections.PathCollection at 0x14cdcbc8>



似乎有些偏差，但并不影响，下面使用神经网络预测填充缺失值

[78]:
```
# 获取缺失值索引
index=pd.concat(
    [
    train[train.isnull().values==True]['Age']
    ,
    test[test.isnull().values==True]['Age']
    ]
    ,axis=0
    ,join='outer').index
```

[79]:
```
# 使用神经网络预测缺失值
predit_=[]
for j in range(263):
    temp1=torch.zeros(1,6)
    temp1[0]=pred[j]
    predit_.append(model(temp1).detach().numpy()[0,0] )
```

[80]:
```
train.isna().sum()
```

[80]:
```
Survived      0
Pclass        0
Sex           0
Age         177
Fare          0
Embarked      0
family        0
call          0
dtype: int64
```

[81]:
```
# 填充缺失值
train['Age'].fillna(pd.Series(predit_[0:177],index[0:177]),inplace=True)
test['Age'].fillna(pd.Series(predit_[177:],index[177:]),inplace=True)
```

[82]:
```
train.isnull().sum()
```

[82]:
```
Survived    0
Pclass      0
Sex         0
Age         0
Fare        0
Embarked    0
family      0
call        0
dtype: int64
```

[83]:
```
# 对数据进行归一化
train['Age']=(train['Age']-train['Age'].min())/train['Age'].max()-train['Age'].
 ↪min()
test['Age']=(test['Age']-test['Age'].min())/test['Age'].max()-test['Age'].min()
test['Age']
```

[83]:
```
0       0.281711
1       0.446184
2       0.643553
3       0.183026
4       0.117237
          …
```

```
413    0.111138
414    0.340921
415    0.334342
416    0.111138
417    0.383990
Name: Age, Length: 418, dtype: float64
```

3.5.3 模型训练与模型验证

[84]:
```
# 数据集划分
X_train = train.drop("Survived", axis=1)
Y_train = train["Survived"]
X_test  = test.copy()
X_train.shape, Y_train.shape, X_test.shape
```

[84]:
```
((891, 7), (891,), (418, 7))
```

逻辑回归

[85]:
```
# 逻辑回归
logreg = LogisticRegression()
logreg.fit(X_train[0:700], Y_train[0:700])
Y_pred = logreg.predict(X_test)
acc_log = round(logreg.score(X_train[700:], Y_train[700:]) * 100, 2)
acc_log
```

[85]:
```
82.2
```

[86]:
```
# 查看 Survived 与其他属性的相关性
coeff_df = pd.DataFrame(train.columns.delete(0))
coeff_df.columns = ['Feature']
coeff_df["Correlation"] = pd.Series(logreg.coef_[0])


coeff_df.sort_values(by='Correlation', ascending=False)
```

[86]:

	Feature	Correlation
3	Fare	0.558564
6	call	0.483823
4	Embarked	-0.261221
5	family	-0.265438

```
0    Pclass    -0.868191
2       Age    -1.634762
1       Sex    -2.272243
```

支持向量机

[87]:
```python
#SVM
svc = SVC()
svc.fit(X_train[0:700], Y_train[0:700])
Y_pred = svc.predict(X_test)
acc_svc = round(svc.score(X_train[700:], Y_train[700:]) * 100, 2)
acc_svc
```

[87]: 85.34

KNN

[88]:
```python
#KNN
knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(X_train[0:700], Y_train[0:700])
Y_pred = knn.predict(X_test)
acc_knn = round(knn.score(X_train[700:], Y_train[700:]) * 100, 2)
acc_knn
```

[88]: 80.1

朴素贝叶斯

[89]:
```python
# 朴素贝叶斯
gaussian = GaussianNB()
gaussian.fit(X_train[0:700], Y_train[0:700])
Y_pred = gaussian.predict(X_test)
acc_gaussian = round(gaussian.score(X_train[700:], Y_train[700:]) * 100, 2)
acc_gaussian
```

[89]: 81.68

感知机

[90]:
```python
# 感知机
perceptron = Perceptron()
perceptron.fit(X_train[0:700], Y_train[0:700])
Y_pred = perceptron.predict(X_test)
```

```
acc_perceptron = round(perceptron.score(X_train[700:], Y_train[700:]) * 100, 2)
acc_perceptron
```

[90]: 79.58

线性 SVC

[91]:
```
# 线性 SVC
linear_svc = LinearSVC()
linear_svc.fit(X_train[0:700], Y_train[0:700])
Y_pred = linear_svc.predict(X_test)
acc_linear_svc = round(linear_svc.score(X_train[700:], Y_train[700:]) * 100, 2)
acc_linear_svc
```

[91]: 82.72

随机剃度下降

[92]:
```
# 随机剃度下降
sgd = SGDClassifier()
sgd.fit(X_train[0:700], Y_train[0:700])
Y_pred = sgd.predict(X_test)
acc_sgd = round(sgd.score(X_train[700:], Y_train[700:]) * 100, 2)
acc_sgd
```

[92]: 78.01

决策树

[93]:
```
# 决策树
decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train[0:700], Y_train[0:700])
Y_pred = decision_tree.predict(X_test)
acc_decision_tree = round(decision_tree.score(X_train[700:], Y_train[700:]) *␣
↪100, 2)
acc_decision_tree
```

[93]: 75.92

随机森林

[94]:
```
# 随机森林
random_forest = RandomForestClassifier(n_estimators=100)
```

```python
random_forest.fit(X_train[0:700], Y_train[0:700])
Y_pred = random_forest.predict(X_test)
random_forest.score(X_train[700:], Y_train[700:])
acc_random_forest = round(random_forest.score(X_train[700:], Y_train[700:]) *␣
↪100, 2)
acc_random_forest
```

[94]: 81.15

[95]:
```python
# 模型比较
models = pd.DataFrame({
    'Model': ['Support Vector Machines', 'KNN', 'Logistic Regression',
              'Random Forest', 'Naive Bayes', 'Perceptron',
              'Stochastic Gradient Decent', 'Linear SVC',
              'Decision Tree'],
    'Score': [acc_svc, acc_knn, acc_log,
              acc_random_forest, acc_gaussian, acc_perceptron,
              acc_sgd, acc_linear_svc, acc_decision_tree]})
models.sort_values(by='Score', ascending=False)
```

[95]:

	Model	Score
0	Support Vector Machines	85.34
7	Linear SVC	82.72
2	Logistic Regression	82.20
4	Naive Bayes	81.68
3	Random Forest	81.15
1	KNN	80.10
5	Perceptron	79.58
6	Stochastic Gradient Decent	78.01
8	Decision Tree	75.92

3.6 模型调参（网格搜索）

选择随机森林，支持向量机，KNN，决策树网格搜索调参数

3.6.1 随机森林

n_estimators

[96]:
```
# n_estimatorsw 参数优化
scores_n=[]
for i in range(10,150,5):
    random_forest = RandomForestClassifier(n_estimators=i)
    random_forest.fit(X_train[0:700], Y_train[0:700])
    Y_pred = random_forest.predict(X_test)
    random_forest.score(X_train[700:], Y_train[700:])
    acc_random_forest = round(random_forest.score(X_train[700:], Y_train[700:])␣
 ↪* 100, 2)
    scores_n.append([i,acc_random_forest])
```

[97]:
```
scores_n
```

[97]:
```
[[10, 82.72],
 [15, 82.72],
 [20, 80.63],
 [25, 80.1],
 [30, 81.15],
 [35, 81.68],
 [40, 81.15],
 [45, 80.1],
 [50, 82.2],
 [55, 79.06],
 [60, 83.25],
 [65, 83.25],
 [70, 80.63],
 [75, 81.68],
 [80, 82.2],
 [85, 81.15],
 [90, 82.2],
 [95, 80.63],
 [100, 81.15],
 [105, 82.72],
 [110, 81.68],
 [115, 81.68],
 [120, 80.63],
 [125, 82.2],
```

```
 [130, 82.2],
 [135, 81.68],
 [140, 81.68],
 [145, 82.72]]
```

[98]:
```
scores_n
```

[98]:
```
[[10, 82.72],
 [15, 82.72],
 [20, 80.63],
 [25, 80.1],
 [30, 81.15],
 [35, 81.68],
 [40, 81.15],
 [45, 80.1],
 [50, 82.2],
 [55, 79.06],
 [60, 83.25],
 [65, 83.25],
 [70, 80.63],
 [75, 81.68],
 [80, 82.2],
 [85, 81.15],
 [90, 82.2],
 [95, 80.63],
 [100, 81.15],
 [105, 82.72],
 [110, 81.68],
 [115, 81.68],
 [120, 80.63],
 [125, 82.2],
 [130, 82.2],
 [135, 81.68],
 [140, 81.68],
 [145, 82.72]]
```

max_features

[99]:
```python
#max_features 参数优化
num_m=[]
scores_m=[]
for i in range(1,8):
    random_forest = RandomForestClassifier(n_estimators=85,max_features=i)
    random_forest.fit(X_train[0:700], Y_train[0:700])
    Y_pred = random_forest.predict(X_test)
    random_forest.score(X_train[700:], Y_train[700:])
    acc_random_forest = round(random_forest.score(X_train[700:], Y_train[700:])␣
↪* 100, 2)
    num_m.append(i)
    scores_m.append(acc_random_forest)
```

[100]:
```python
scores_m
```

[100]:
```
[81.15, 80.63, 82.2, 82.72, 80.1, 81.15, 81.68]
```

[101]:
```python
# 同时优化两个参数
scores_t=[]
for i in range(1,8):
    for j in range(10,100,5):
        random_forest = RandomForestClassifier(n_estimators=j,max_features=i)
        random_forest.fit(X_train[0:700], Y_train[0:700])
        Y_pred = random_forest.predict(X_test)
        random_forest.score(X_train[700:], Y_train[700:])
        acc_random_forest = round(random_forest.score(X_train[700:],␣
↪Y_train[700:]) * 100, 2)
        scores_t.append([i,j,acc_random_forest])
```

[102]:
```python
scores_t[35:40]
```

[102]:
```
[[2, 95, 81.68], [3, 10, 83.77], [3, 15, 80.1], [3, 20, 81.15], [3, 25, 82.72]]
```

3.6.2 KNN

[103]:
```python
#n_neighbors 参数优化
num_knn=[]
scores_knn=[]
for i in range(1,10):
```

```
    knn = KNeighborsClassifier(n_neighbors = i)
    knn.fit(X_train[0:700], Y_train[0:700])
    Y_pred = knn.predict(X_test)
    acc_knn = round(knn.score(X_train[700:], Y_train[700:]) * 100, 2)
    num_knn.append(i)
    scores_knn.append(acc_knn)
```

[104]:
```
scores_knn
```

[104]: [72.25, 81.15, 80.1, 81.15, 80.63, 81.68, 83.77, 82.2, 82.2]

3.6.3 决策树

[105]:
```
#max_depth 参数优化
scores_dec=[]
for i in range(1,20):
    decision_tree = DecisionTreeClassifier(max_depth=i)
    decision_tree.fit(X_train[0:700], Y_train[0:700])
    Y_pred = decision_tree.predict(X_test)
    acc_decision_tree = round(decision_tree.score(X_train[700:], Y_train[700:])␣
 ↪* 100, 2)
    scores_dec.append([i,acc_decision_tree])
```

[106]:
```
scores_dec
```

[106]:
```
[[1, 79.06],
 [2, 80.63],
 [3, 85.34],
 [4, 84.29],
 [5, 82.72],
 [6, 82.72],
 [7, 83.77],
 [8, 82.72],
 [9, 81.15],
 [10, 81.15],
 [11, 76.96],
 [12, 76.44],
 [13, 76.44],
```

 [14, 75.39],
 [15, 74.87],
 [16, 75.92],
 [17, 75.92],
 [18, 76.44],
 [19, 75.92]]
```

### 3.6.4 支持向量机

[107]:
```python
#C 参数优化
scores_svc=[]
for i in range(1,10):
    svc = SVC(C=i*0.1)
    svc.fit(X_train[0:700], Y_train[0:700])
    Y_pred = svc.predict(X_test)
    acc_svc = round(svc.score(X_train[700:], Y_train[700:]) * 100, 2)
    scores_svc.append([i*0.1,acc_svc])
```

[108]:
```python
scores_svc
```

[108]:
```
[[0.1, 83.77],
 [0.2, 83.77],
 [0.30000000000000004, 83.77],
 [0.4, 84.82],
 [0.5, 85.86],
 [0.6000000000000001, 84.82],
 [0.7000000000000001, 85.34],
 [0.8, 85.34],
 [0.9, 85.34]]
```

3.7 模型融合 (Stacking)

[109]:
```python
# 使用最优参数预测并查看 SVC 准确率
svc = SVC(C=0.5)
svc.fit(X_train[0:700], Y_train[0:700])
Y1 = svc.predict(X_train[700:])
acc_svc = round(svc.score(X_train[700:], Y_train[700:]) * 100, 2)
acc_svc
```
```

[109]: 85.86

[110]:
```
# 使用最优参数预测并查看 KNN 准确率
knn = KNeighborsClassifier(n_neighbors = 4)
knn.fit(X_train[0:700], Y_train[0:700])
Y_2 = knn.predict(X_train[700:])
acc_knn = round(knn.score(X_train[700:], Y_train[700:]) * 100, 2)
acc_knn
```

[110]: 81.15

[111]:
```
# 使用最优参数预测并查看随机森林准确率
random_forest = RandomForestClassifier(n_estimators=10,max_features=3)
random_forest.fit(X_train[0:700], Y_train[0:700])
Y_3 = random_forest.predict(X_train[700:])
acc_random_forest=random_forest.score(X_train[700:], Y_train[700:])*100
acc_random_forest
```

[111]: 80.6282722513089

[112]:
```
# 使用最优参数预测并查看决策树准确率
decision_tree = DecisionTreeClassifier(max_depth=3)
decision_tree.fit(X_train[0:700], Y_train[0:700])
Y_4 = decision_tree.predict(X_train[700:])
acc_decision_tree = round(decision_tree.score(X_train[700:], Y_train[700:]) *␣
↪100, 2)
acc_decision_tree
```

[112]: 85.34

[113]:
```
# 使用最优参数预测并查看逻辑回归准确率
logreg = LogisticRegression()
logreg.fit(X_train[0:700], Y_train[0:700])
Y_5 = logreg.predict(X_train[700:])
acc_log = round(logreg.score(X_train[700:], Y_train[700:]) * 100, 2)
acc_log
```

[113]: 82.2

[114]:
```
#Stacking 模型第二步
Y=Y1+Y_2+Y_3+Y_4+Y_5
```

[115]:
```
Y
```

[115]:
```
array([5, 2, 3, 0, 0, 0, 5, 1, 5, 4, 5, 1, 0, 1, 0, 0, 5, 5, 0, 0, 5, 0,
       1, 0, 0, 0, 5, 5, 0, 4, 5, 1, 0, 0, 0, 0, 0, 2, 0, 0, 1, 1, 5, 0,
       0, 3, 0, 5, 0, 1, 5, 4, 0, 0, 5, 5, 0, 0, 0, 3, 0, 0, 0, 5, 0, 5,
       2, 5, 0, 0, 0, 0, 5, 0, 5, 0, 0, 4, 0, 5, 3, 5, 2, 0, 0, 0, 4, 0,
       4, 1, 0, 0, 0, 0, 0, 0, 3, 4, 0, 4, 1, 5, 5, 4, 0, 0, 1, 4, 0, 5,
       0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 5, 0, 1, 4, 0, 0, 2, 5, 0, 5, 3, 5,
       0, 0, 0, 5, 0, 0, 1, 0, 0, 0, 5, 0, 0, 0, 0, 0, 2, 5, 0, 0, 5, 5,
       5, 3, 5, 0, 5, 0, 0, 0, 5, 0, 0, 5, 5, 1, 0, 4, 0, 5, 1, 0, 5, 3,
       0, 0, 0, 5, 5, 0, 5, 0, 0, 0, 1, 5, 3, 0, 1], dtype=int64)
```

[116]:
```
# 得出最终预测结果
Y[Y<3]=0
Y[Y>=3]=1
```

[117]:
```
Y
```

[117]:
```
array([1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0,
       0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1,
       0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1,
       1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1,
       0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0], dtype=int64)
```

[118]:
```
# 计算预测准确个数
(Y==Y_train[700:]).sum()
```

[118]:
```
164
```

[119]:
```
# 计算准确率
(Y==Y_train[700:]).sum()/191*100
```

```
[119]: 85.86387434554975
```