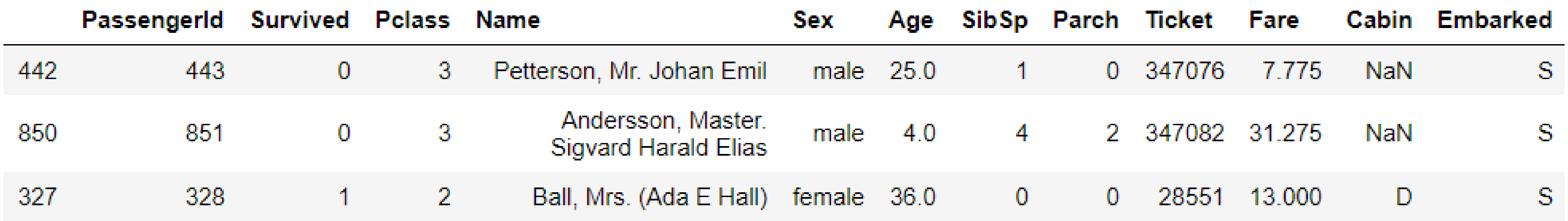
Kaggle 鐵達尼號生存預測

陳子農

```
import numpy as np
import matplotlib.pyplot as mp
from scipy import stats
import pandas as pd
import seaborn as sns
sns.set(style='darkgrid')

import sklearn.preprocessing as sp
import sklearn.ensemble as se
import sklearn.metrics as sm
import sklearn.model_selection as ms

import string
import warnings
warnings.filterwarnings('ignore')

SEED = 1000
```

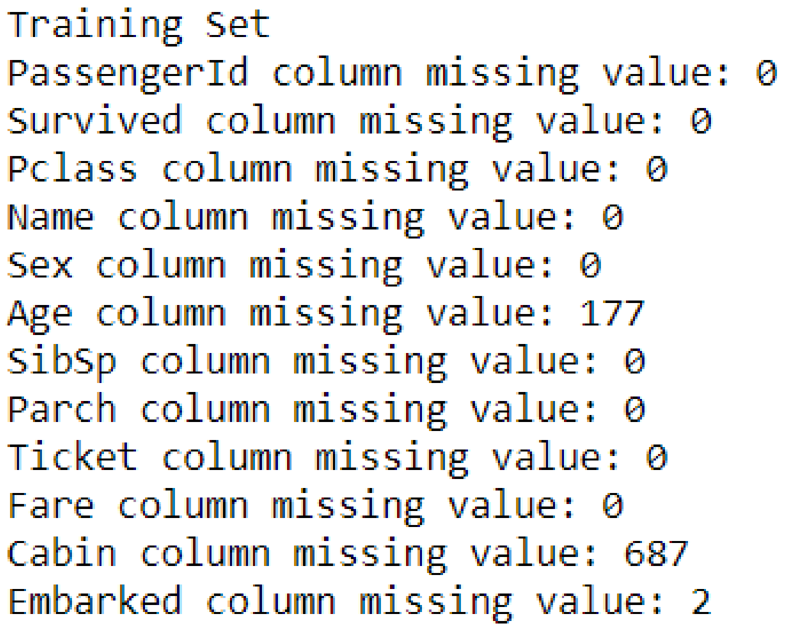
```
def concat_df(train_data, test_data):
   return pd.concat([train_data, test_data],
sort=True).reset_index(drop=True)


def divide_df(all_data):
   return all_data.loc[:890], all_data.loc[891:].drop(['Survived'], axis=1)


df_train = pd.read_csv('train.csv')
df_test = pd.read_csv('test.csv')
df_all = concat_df(df_train, df_test)
```

```
df_train.name = 'Training Set'
df_test.name = 'Test Set'
df_all.name = 'All Set‘
dfs = [df_train, df_test]
df_train.sample(3)
```

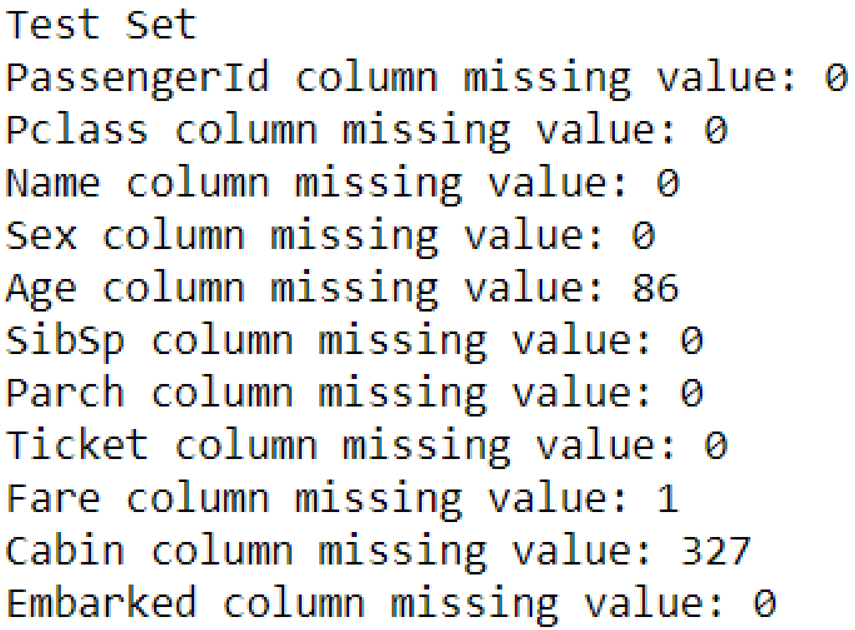


	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
442	443	0	3	Petterson, Mr. Johan Emil	male	25.0	1	0	347076	7.775	NaN	S
850	851	0	3	Andersson, Master. Sigvard Harald Elias	male	4.0	4	2	347082	31.275	NaN	S
327	328	1	2	Ball, Mrs. (Ada E Hall)	female	36.0	0	0	28551	13.000	D	S

```
def display_missing(df):
    for col in df.columns.tolist():
        print('{} column missing value: {}'.format(col, df[col].isnull().sum()))
    print('\n')
for df in dfs:
    print('{}'.format(df.name))
    display_missing(df)
```

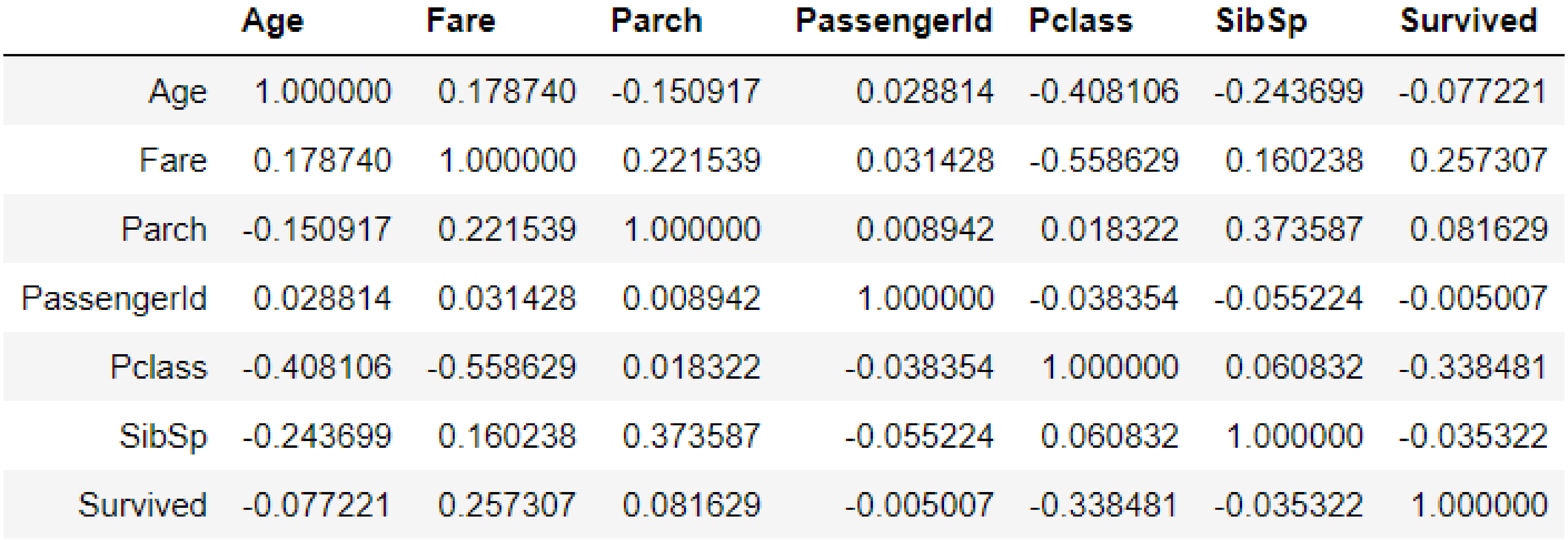


```
Training Set
PassengerId column missing value: 0
Survived column missing value: 0
Pclass column missing value: 0
Name column missing value: 0
Sex column missing value: 0
Age column missing value: 177
SibSp column missing value: 0
Parch column missing value: 0
Ticket column missing value: 0
Fare column missing value: 0
Cabin column missing value: 687
Embarked column missing value: 2
```



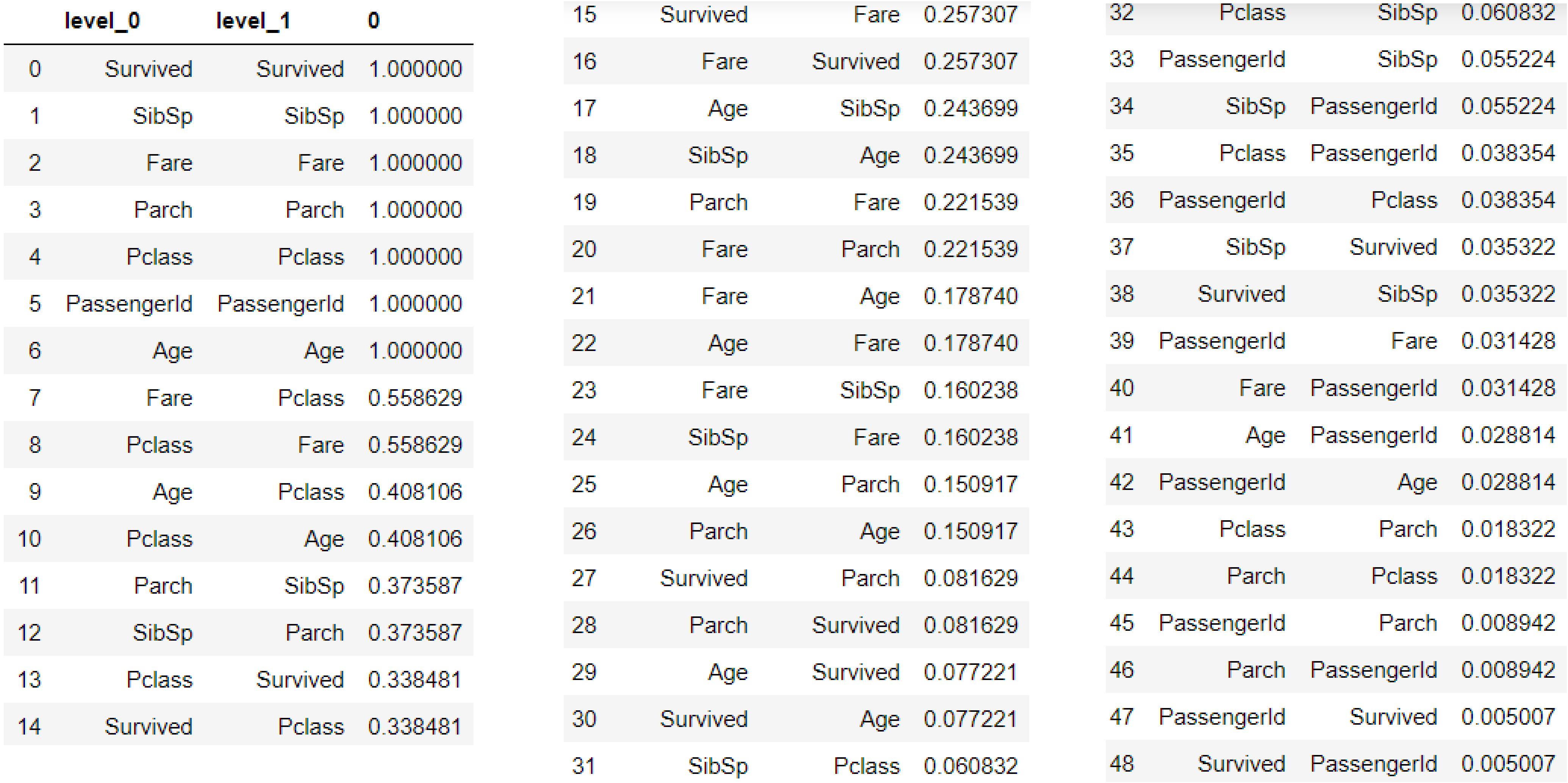
```
Test Set
PassengerId column missing value: 0
Pclass column missing value: 0
Name column missing value: 0
Sex column missing value: 0
Age column missing value: 86
SibSp column missing value: 0
Parch column missing value: 0
Ticket column missing value: 0
Fare column missing value: 1
Cabin column missing value: 327
Embarked column missing value: 0
```

df_all.corr()



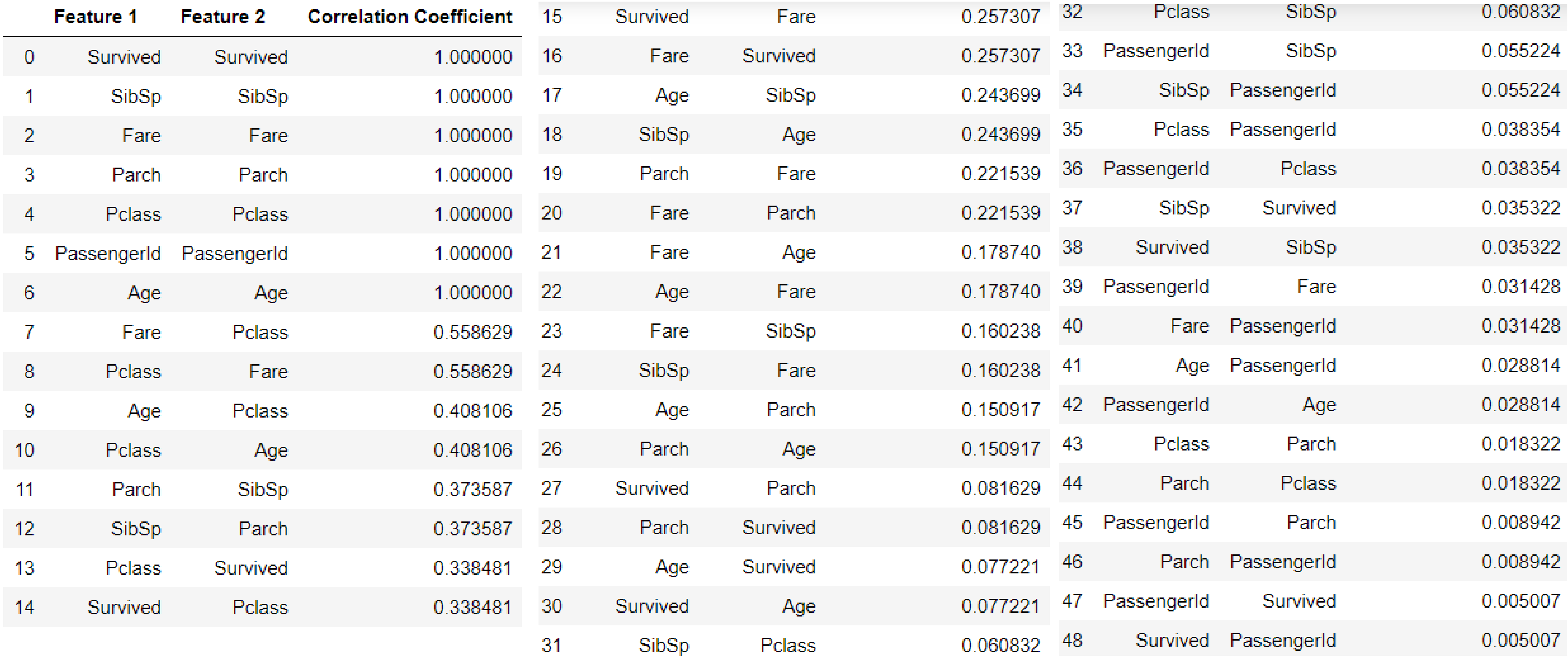
	Age	Fare	Parch	PassengerId	Pclass	SibSp	Survived
Age	1.000000	0.178740	-0.150917	0.028814	-0.408106	-0.243699	-0.077221
Fare	0.178740	1.000000	0.221539	0.031428	-0.558629	0.160238	0.257307
Parch	-0.150917	0.221539	1.000000	0.008942	0.018322	0.373587	0.081629
PassengerId	0.028814	0.031428	0.008942	1.000000	-0.038354	-0.055224	-0.005007
Pclass	-0.408106	-0.558629	0.018322	-0.038354	1.000000	0.060832	-0.338481
SibSp	-0.243699	0.160238	0.373587	-0.055224	0.060832	1.000000	-0.035322
Survived	-0.077221	0.257307	0.081629	-0.005007	-0.338481	-0.035322	1.000000

```
df_all_corr=df_all.corr().abs().unstack().sort_values(kind='quicksort',
ascending=False).reset_index()
df_all_corr
```



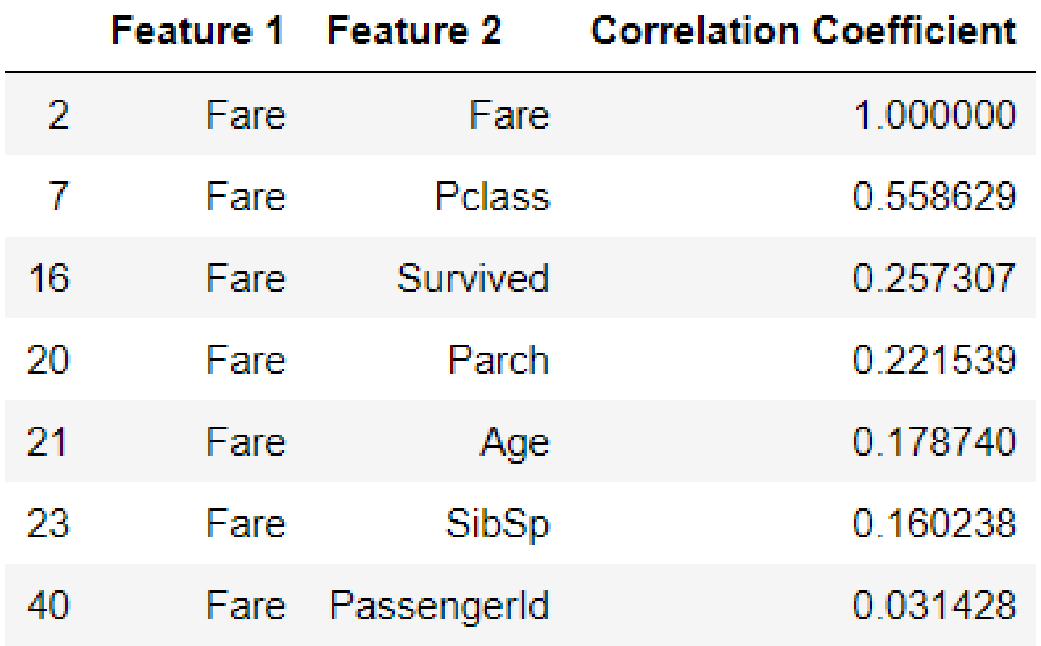
	level_0	level_1	0
0	Survived	Survived	1.000000
1	SibSp	SibSp	1.000000
2	Fare	Fare	1.000000
3	Parch	Parch	1.000000
4	Pclass	Pclass	1.000000
5	PassengerId	PassengerId	1.000000
6	Age	Age	1.000000
7	Fare	Pclass	0.558629
8	Pclass	Fare	0.558629
9	Age	Pclass	0.408106
10	Pclass	Age	0.408106
11	Parch	SibSp	0.373587
12	SibSp	Parch	0.373587
13	Pclass	Survived	0.338481
14	Survived	Pclass	0.338481
15	Survived	Fare	0.257307
16	Fare	Survived	0.257307
17	Age	SibSp	0.243699
18	SibSp	Age	0.243699
19	Parch	Fare	0.221539
20	Fare	Parch	0.221539
21	Fare	Age	0.178740
22	Age	Fare	0.178740
23	Fare	SibSp	0.160238
24	SibSp	Fare	0.160238
25	Age	Parch	0.150917
26	Parch	Age	0.150917
27	Survived	Parch	0.081629
28	Parch	Survived	0.081629
29	Age	Survived	0.077221
30	Survived	Age	0.077221
31	SibSp	Pclass	0.060832
32	Pclass	SibSp	0.060832
33	PassengerId	SibSp	0.055224
34	SibSp	PassengerId	0.055224
35	Pclass	PassengerId	0.038354
36	PassengerId	Pclass	0.038354
37	SibSp	Survived	0.035322
38	Survived	SibSp	0.035322
39	PassengerId	Fare	0.031428
40	Fare	PassengerId	0.031428
41	Age	PassengerId	0.028814
42	PassengerId	Age	0.028814
43	Pclass	Parch	0.018322
44	Parch	Pclass	0.018322
45	PassengerId	Parch	0.008942
46	Parch	PassengerId	0.008942
47	PassengerId	Survived	0.005007
48	Survived	PassengerId	0.005007

```
df_all_corr.rename(columns={'level_0':'Feature 1', 'level_1':'Feature 2',
0:'Correlation Coefficient'}, inplace=True)
df_all_corr
```



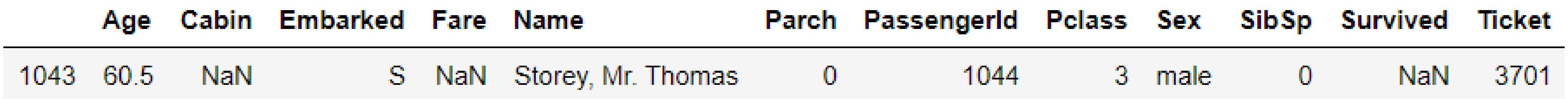
	Feature 1	Feature 2	Correlation Coefficient
0	Survived	Survived	1.000000
1	SibSp	SibSp	1.000000
2	Fare	Fare	1.000000
3	Parch	Parch	1.000000
4	Pclass	Pclass	1.000000
5	PassengerId	PassengerId	1.000000
6	Age	Age	1.000000
7	Fare	Pclass	0.558629
8	Pclass	Fare	0.558629
9	Age	Pclass	0.408106
10	Pclass	Age	0.408106
11	Parch	SibSp	0.373587
12	SibSp	Parch	0.373587
13	Pclass	Survived	0.338481
14	Survived	Pclass	0.338481
15	Survived	Fare	0.257307
16	Fare	Survived	0.257307
17	Age	SibSp	0.243699
18	SibSp	Age	0.243699
19	Parch	Fare	0.221539
20	Fare	Parch	0.221539
21	Fare	Age	0.178740
22	Age	Fare	0.178740
23	Fare	SibSp	0.160238
24	SibSp	Fare	0.160238
25	Age	Parch	0.150917
26	Parch	Age	0.150917
27	Survived	Parch	0.081629
28	Parch	Survived	0.081629
29	Age	Survived	0.077221
30	Survived	Age	0.077221
31	SibSp	Pclass	0.060832
32	Pclass	SibSp	0.060832
33	PassengerId	SibSp	0.055224
34	SibSp	PassengerId	0.055224
35	Pclass	PassengerId	0.038354
36	PassengerId	Pclass	0.038354
37	SibSp	Survived	0.035322
38	Survived	SibSp	0.035322
39	PassengerId	Fare	0.031428
40	Fare	PassengerId	0.031428
41	Age	PassengerId	0.028814
42	PassengerId	Age	0.028814
43	Pclass	Parch	0.018322
44	Parch	Pclass	0.018322
45	PassengerId	Parch	0.008942
46	Parch	PassengerId	0.008942
47	PassengerId	Survived	0.005007
48	Survived	PassengerId	0.005007

```
df_all_corr[df_all_corr['Feature 1']=='Fare']
```



	Feature 1	Feature 2	Correlation Coefficient
2	Fare	Fare	1.000000
7	Fare	Pclass	0.558629
16	Fare	Survived	0.257307
20	Fare	Parch	0.221539
21	Fare	Age	0.178740
23	Fare	SibSp	0.160238
40	Fare	PassengerId	0.031428

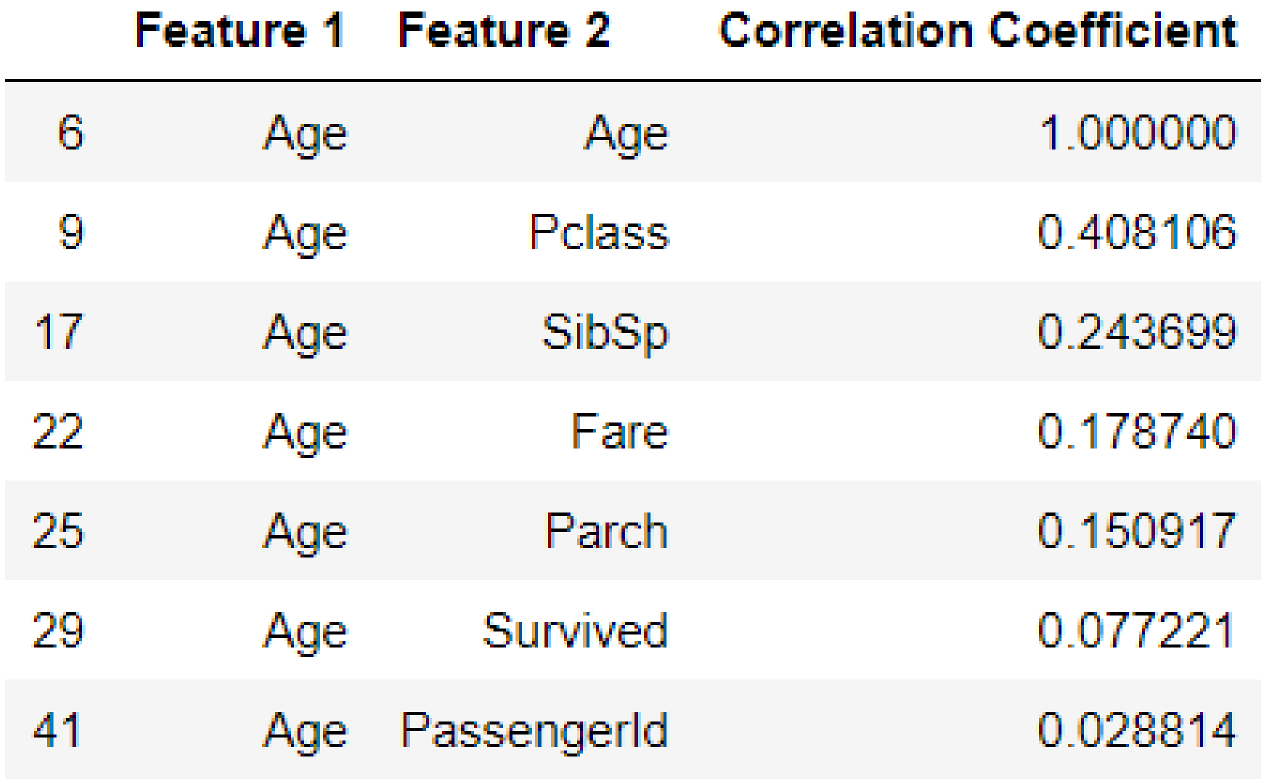
```
shapiro_age, p = stats.shapiro(df_all.loc[:,['Fare']].dropna())
df_all[df_all['Fare'].isnull()]
```



	Age	Cabin	Embarked	Fare	Name	Parch	PassengerId	Pclass	Sex	SibSp	Survived	Ticket
1043	60.5	NaN	S	NaN	Storey, Mr. Thomas	0	1044	3	male	0	NaN	3701

```
med_fare =
df_all.groupby(['Parch','Pclass','SibSp']).Fare.median()[0][3][0]
df_all['Fare'] = df_all['Fare'].fillna(med_fare)
```

```
df_all_corr[df_all_corr['Feature 1']=='Age']
```



	Feature 1	Feature 2	Correlation Coefficient
6	Age	Age	1.000000
9	Age	Pclass	0.408106
17	Age	SibSp	0.243699
22	Age	Fare	0.178740
25	Age	Parch	0.150917
29	Age	Survived	0.077221
41	Age	PassengerId	0.028814

```
shapiro_age, p = stats.shapiro(df_all.loc[:,['Age']].dropna())
print(p)        5.74782790807582e-11


female_age = df_all.loc[df_all['Sex']=='female',['Age']].dropna()
male_age = df_all.loc[df_all['Sex']=='male',['Age']].dropna()
# female_age.head()
Tt, p = stats.ttest_ind(female_age.loc[:,'Age'], male_age.loc[:,'Age'])
print(p)        0.03958770007671348


stat, p, med, tbl = stats.median_test(female_age.loc[:,'Age'],
male_age.loc[:,'Age'])
print(p)        0.39247502235548404
```

```
female_1 =
df_all.loc[(df_all['Pclass']==1)&(df_all['Sex']=='female'),['Age']].dropna()
female_2 =
df_all.loc[(df_all['Pclass']==2)&(df_all['Sex']=='female'),['Age']].dropna()
female_3 =
df_all.loc[(df_all['Pclass']==3)&(df_all['Sex']=='female'),['Age']].dropna()
stat, p = stats.f_oneway(female_1, female_2, female_3)
print(p)     [1.27578628e-18]
male_1 =
df_all.loc[(df_all['Pclass']==1)&(df_all['Sex']=='male'),['Age']].dropna()
male_2 =
df_all.loc[(df_all['Pclass']==2)&(df_all['Sex']=='male'),['Age']].dropna()
male_3 =
df_all.loc[(df_all['Pclass']==3)&(df_all['Sex']=='male'),['Age']].dropna()
stat, p = stats.f_oneway(male_1, male_2, male_3)
print(p)     [1.01425403e-28]
```

```
age_by_pclass_sex = df_all.groupby(['Sex','Pclass']).median()['Age']
for pclass in range(1, 4):
    for sex in ['female','male']:
        print('Median age of pclass {} {}s: {}'.format(pclass, sex,
age_by_pclass_sex[sex, pclass]))
print('Median age of all pessengers:{}'.format(df_all['Age'].median()))
df_all['Age'] = df_all.groupby(['Sex','Pclass'])['Age'].apply(lambda
x:x.fillna(x.median()))
```

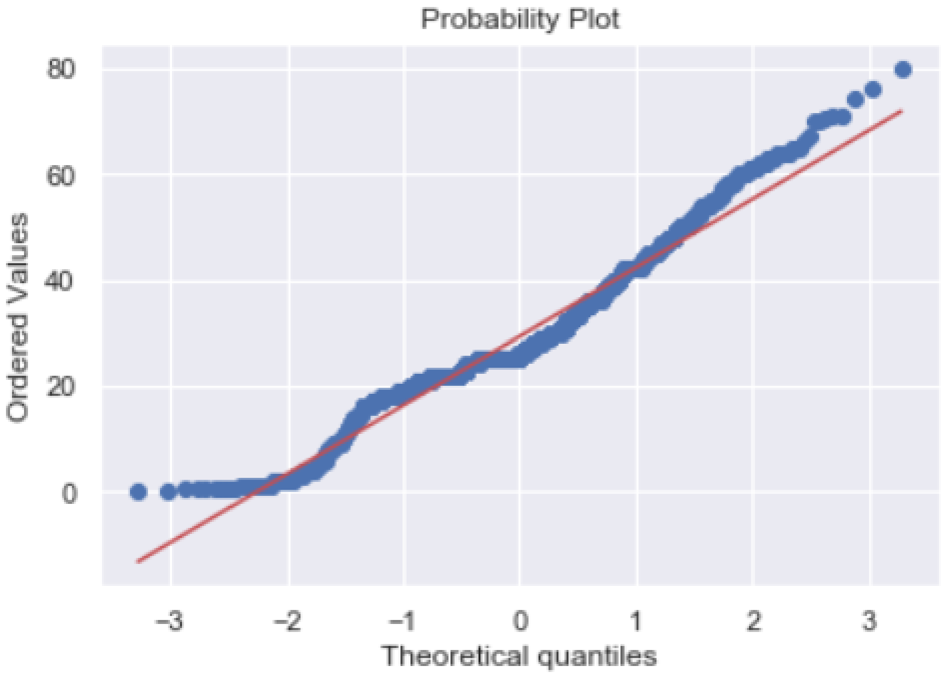
```
Median age of pclass 1 females: 36.0
Median age of pclass 1 males: 42.0
Median age of pclass 2 females: 28.0
Median age of pclass 2 males: 29.5
Median age of pclass 3 females: 22.0
Median age of pclass 3 males: 25.0
Median age of all pessengers:28.0
```

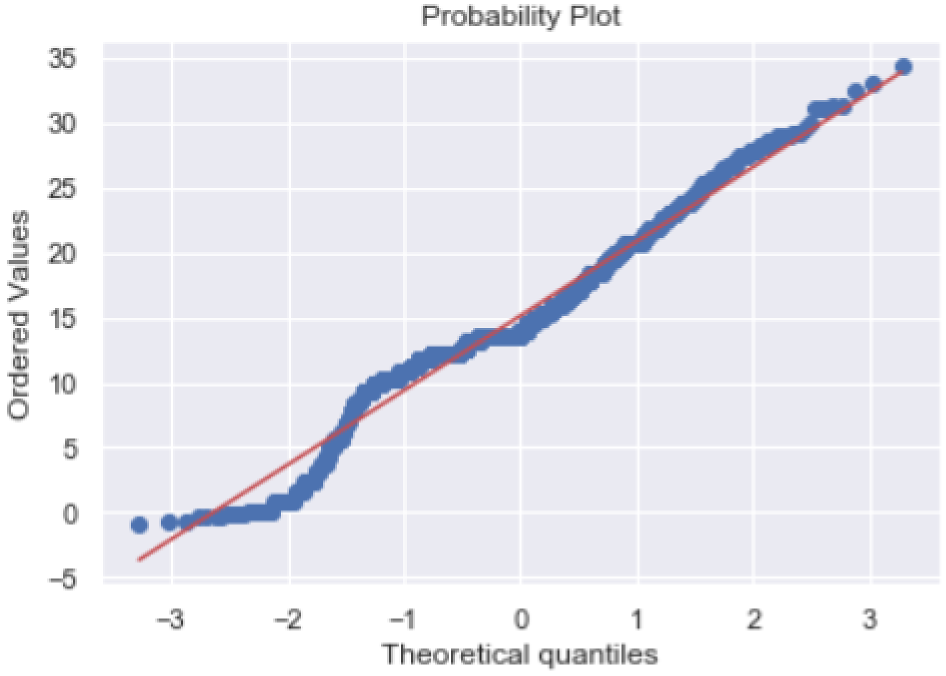
```
age_all = df_all.loc[:,'Age']
stats.probplot(age_all,plot=mp)
mp.show()


x, lambda_box = stats.boxcox(df_all.loc[:,'Age'])
print('lambda={}'.format(lambda_box))
```

```
lambda=0.751393574356586
```

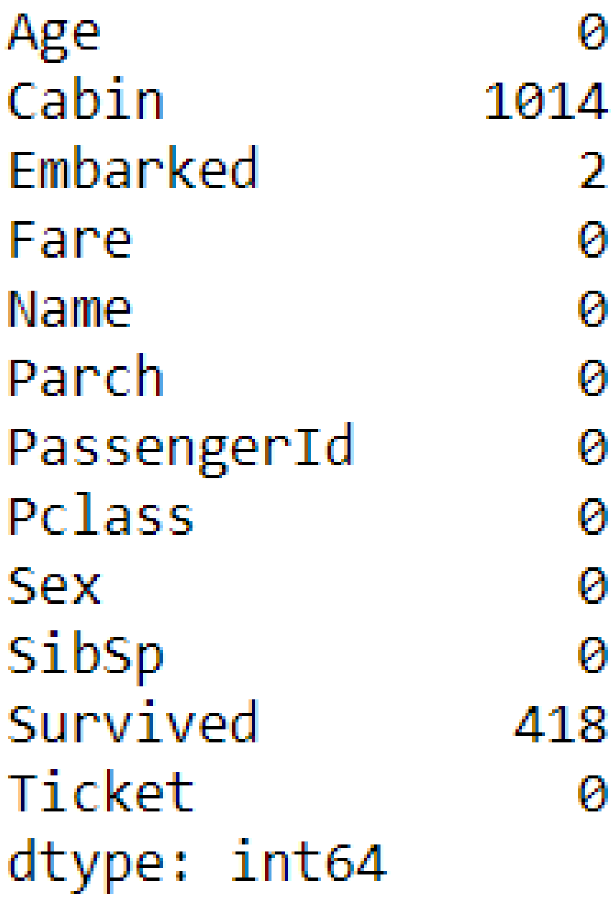
```
stats.probplot(x, plot=mp)
mp.show()
df_all.loc[:,'Age'] = x
print(x)
```

```
[12.2466548 19.14168134 14.06253422 ...
19.34375915 13.61550726 13.61550726]
```





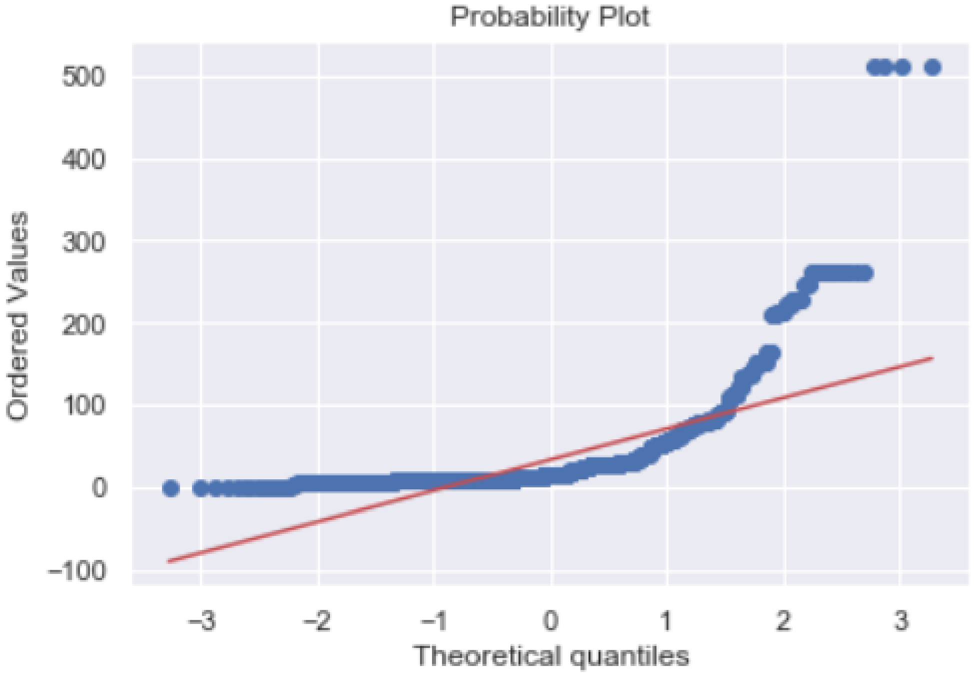
df_all.isnull().sum()

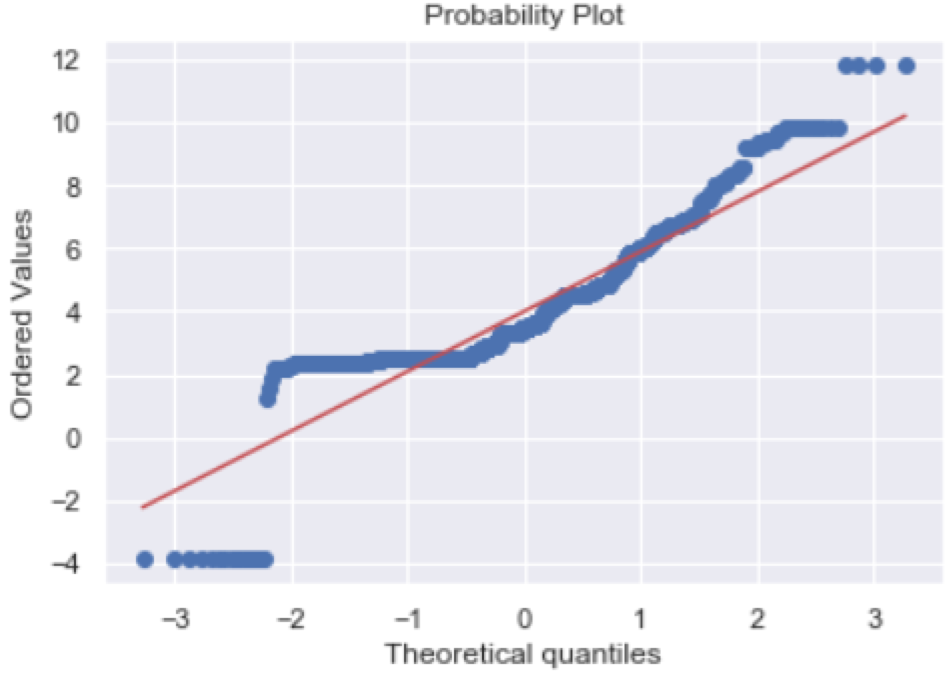


```
Age               0
Cabin          1014
Embarked          2
Fare              0
Name              0
Parch             0
PassengerId       0
Pclass            0
Sex               0
SibSp             0
Survived        418
Ticket            0
dtype: int64
```

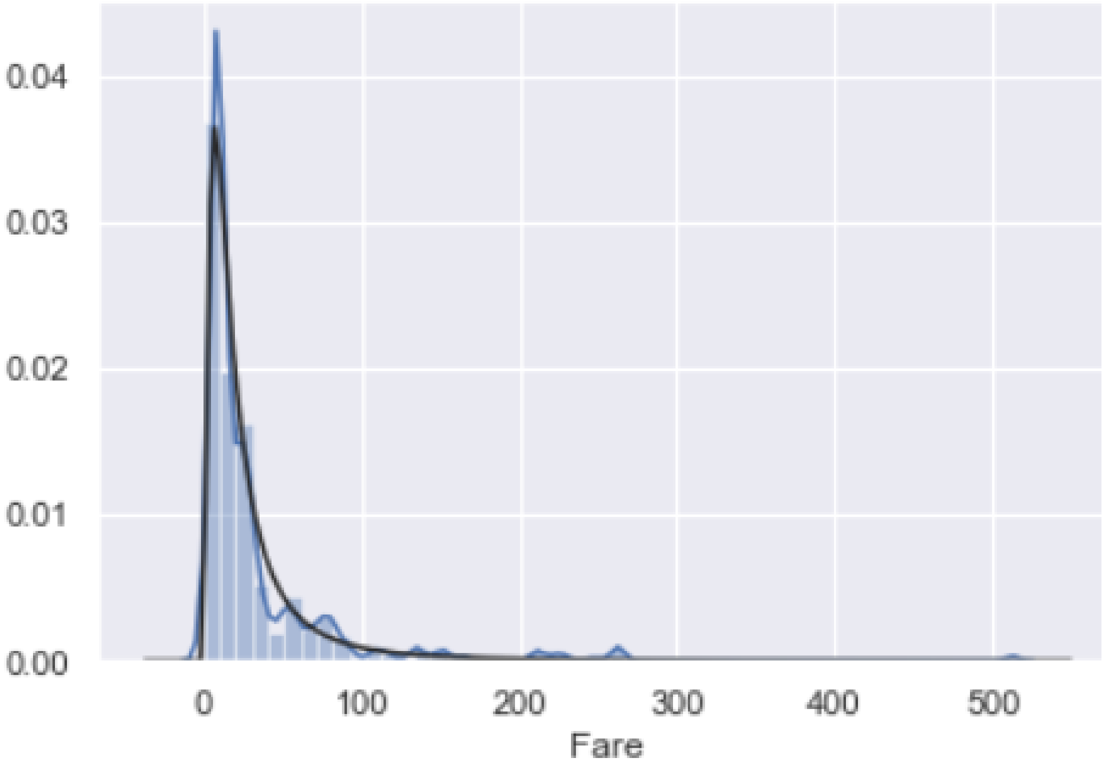
```
fare_all = df_all.loc[:,'Fare']
stats.probplot(fare_all, plot=mp)
mp.show()


x, lambda_box = stats.boxcox(df_all.loc[:,'Fare']+0.001)
stats.probplot(x, plot=mp)
mp.show()


sns.distplot(df_all['Fare'],fit=stats.exponweib)
print(stats.exponweib.fit(df_all['Fare']))
```

```
(106.18175044619605, 0.25252609240692425,
-1.5922706193161118, 0.0323692260660689)
```

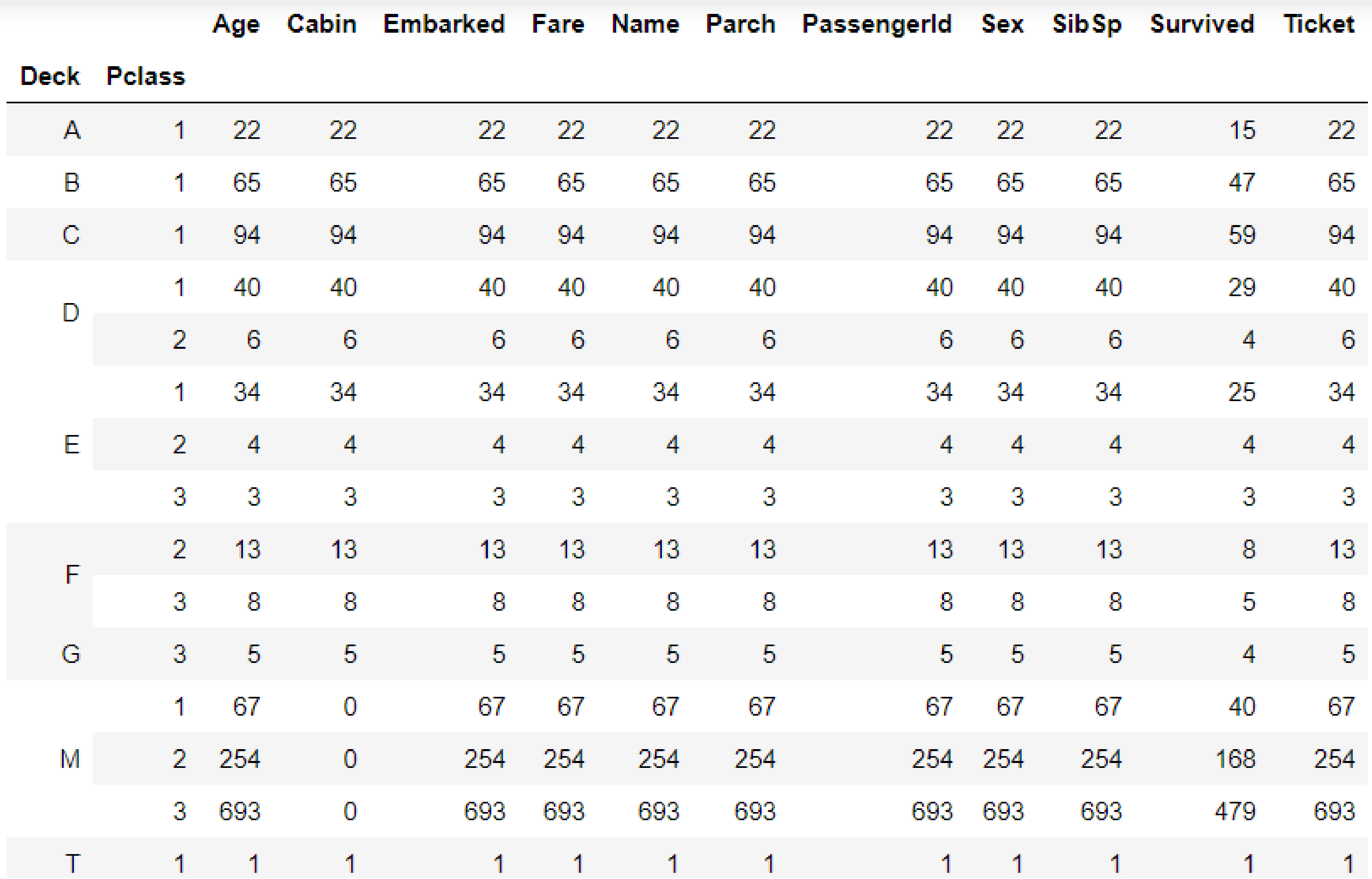




```
df_all.loc[:,'Fare'] = stats.exponweib.cdf(df_all['Fare'],
stats.exponweib.fit(df_all['Fare'])[0],
stats.exponweib.fit(df_all['Fare'])[1],
stats.exponweib.fit(df_all['Fare'])[2],
stats.exponweib.fit(df_all['Fare'])[3])
```

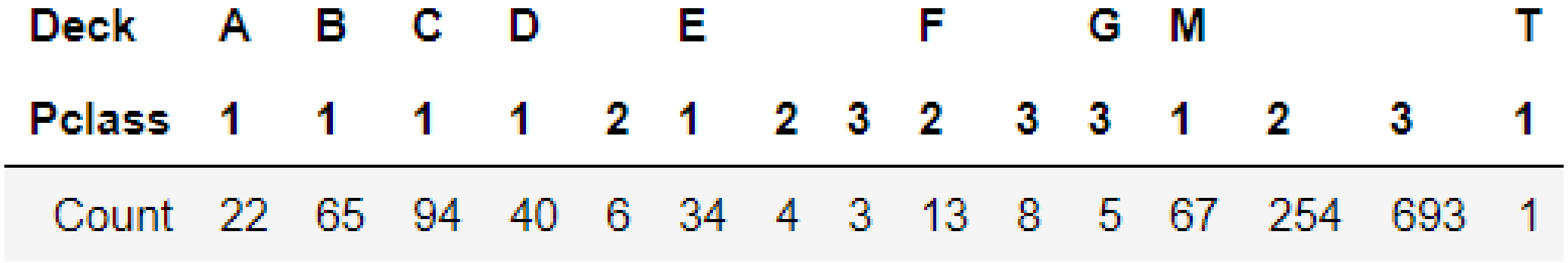


```
df_all['Embarked'] = df_all['Embarked'].fillna('S')
df_all['Deck'] = df_all['Cabin'].apply(lambda s: s[0] if pd.notnull(s) else 'M')
df_all.groupby(['Deck','Pclass']).count()
```



		Age	Cabin	Embarked	Fare	Name	Parch	PassengerId	Sex	SibSp	Survived	Ticket
Deck	**Pclass**											
A	1	22	22	22	22	22	22	22	22	22	15	22
B	1	65	65	65	65	65	65	65	65	65	47	65
C	1	94	94	94	94	94	94	94	94	94	59	94
D	1	40	40	40	40	40	40	40	40	40	29	40
	2	6	6	6	6	6	6	6	6	6	4	6
E	1	34	34	34	34	34	34	34	34	34	25	34
	2	4	4	4	4	4	4	4	4	4	4	4
	3	3	3	3	3	3	3	3	3	3	3	3
F	2	13	13	13	13	13	13	13	13	13	8	13
	3	8	8	8	8	8	8	8	8	8	5	8
G	3	5	5	5	5	5	5	5	5	5	4	5
M	1	67	0	67	67	67	67	67	67	67	40	67
	2	254	0	254	254	254	254	254	254	254	168	254
	3	693	0	693	693	693	693	693	693	693	479	693
T	1	1	1	1	1	1	1	1	1	1	1	1

```
df_all_decks = df_all.groupby(['Deck','Pclass']).count().drop(columns=/
['Age','Cabin','Embarked','Fare','Parch','PassengerId','Sex','SibSp',
'Survived','Ticket']).rename(columns={'Name':'Count'}).transpose()
df_all_decks
```

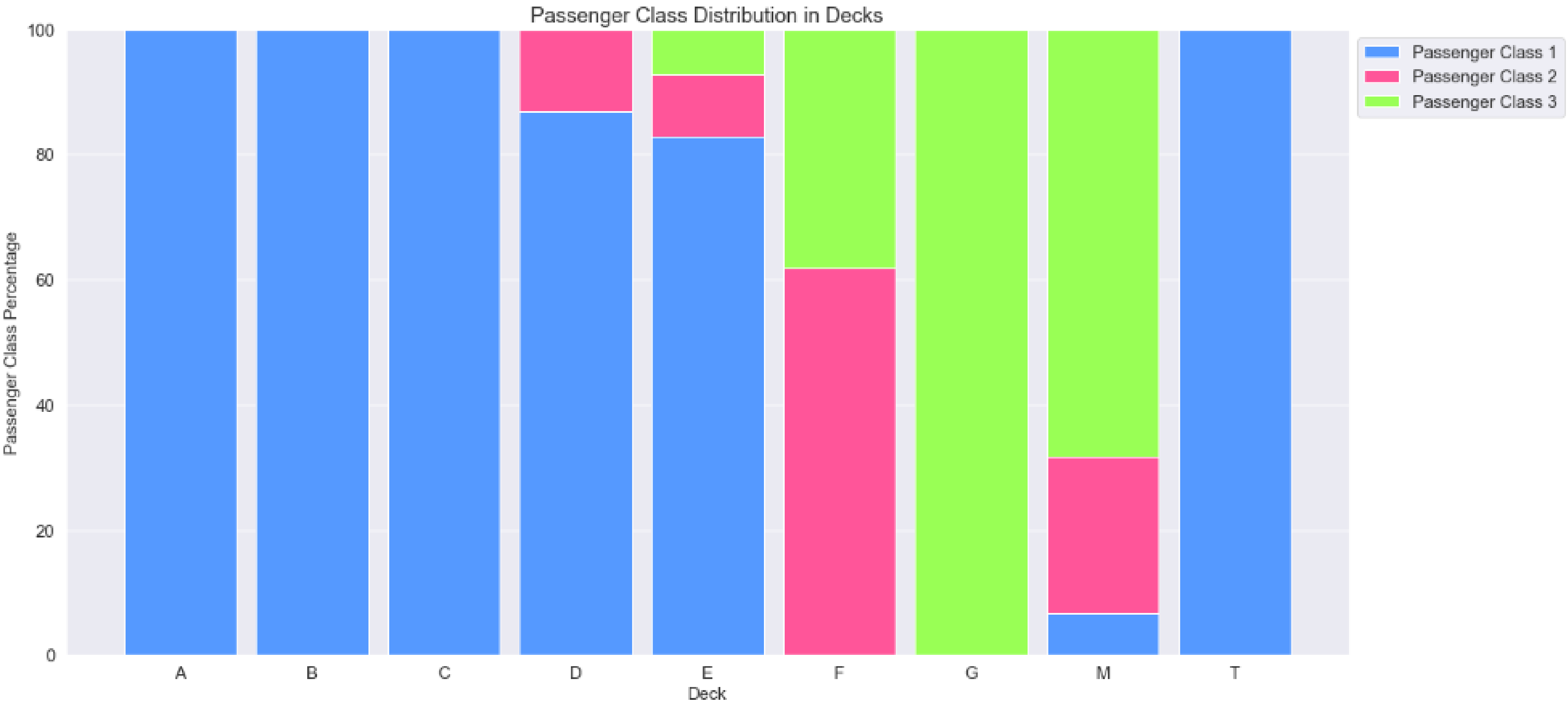


Deck	A	B	C	D		E			F		G	M			T
Pclass	1	1	1	1	2	1	2	3	2	3	3	1	2	3	1
Count	22	65	94	40	6	34	4	3	13	8	5	67	254	693	1

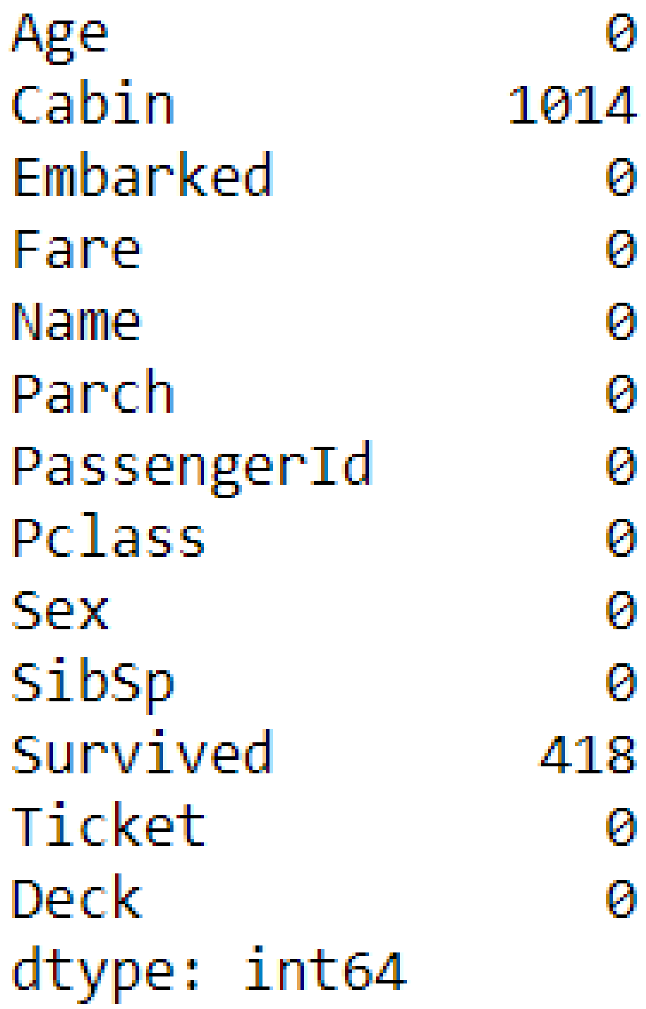
```
def get_pclass_dist(df):
    deck_counts = {'A':{}, 'B':{}, 'C':{}, 'D':{}, 'E':{}, 'F':{}, 'G':{}, 'M':{}, 'T':{}}
    decks = df.columns.levels[0]
    for deck in decks:
        for pclass in range(1, 4):
            try:
                count = df[deck][pclass][0]
                deck_counts[deck][pclass] = count
            except KeyError:
                deck_counts[deck][pclass] = 0
    df_decks = pd.DataFrame(deck_counts)
    deck_percentages = {}
    for col in df_decks:
        deck_percentages[col] = [(count / df_decks[col].sum()) * 100 for count in df_decks[col]]
    return deck_counts, deck_percentages
all_deck_count, all_deck_per = get_pclass_dist(df_all_decks)
all_deck_count, all_deck_per
```

```
({'A': {1: 22, 2: 0, 3: 0},
'B': {1: 65, 2: 0, 3: 0},
'C': {1: 94, 2: 0, 3: 0},
'D': {1: 40, 2: 6, 3: 0},
'E': {1: 34, 2: 4, 3: 3},
'F': {1: 0, 2: 13, 3: 8},
'G': {1: 0, 2: 0, 3: 5},
'M': {1: 67, 2: 254, 3: 693},
'T': {1: 1, 2: 0, 3: 0}},
{'A': [100.0, 0.0, 0.0],
'B': [100.0, 0.0, 0.0],
'C': [100.0, 0.0, 0.0],
'D': [86.95652173913044, 13.043478260869565, 0.0],
'E': [82.92682926829268, 9.75609756097561, 7.317073170731707],
'F': [0.0, 61.904761904761905, 38.095238095238095],
'G': [0.0, 0.0, 100.0],
'M': [6.607495069033531, 25.04930966469428, 68.34319526627219],
'T': [100.0, 0.0, 0.0]})
```

```
def display_surv_dist(per):
    df_per = pd.DataFrame(per).T
    deck_names = ['A','B','C','D','E','F','G','M','T']
    back_count = np.arange(len(deck_names))

    pclass1 = df_per[0]
    pclass2 = df_per[1]
    pclass3 = df_per[2]
    mp.figure(figsize=(20,10))
    mp.title('Passenger Class Distribution in Decks', fontsize=18)
    mp.xlabel('Deck',fontsize=15)
    mp.ylabel('Passenger Class Percentage',fontsize=15)
    mp.xticks(back_count,deck_names)
    mp.tick_params(labelsize=15)
    mp.bar(back_count, pclass1, width=0.85, color='#5599ff', edgecolor='White', label='Passenger Class 1')
    mp.bar(back_count, pclass2, bottom=pclass1, width=0.85, color='#ff5599', edgecolor='White', label='Passenger Class 2')
    mp.bar(back_count, pclass3, bottom=pclass1+pclass2, width=0.85, color='#99ff55', edgecolor='White', label='Passenger Class 3')
    mp.legend(bbox_to_anchor=(1, 1), prop={'size': 15})
    mp.show()
display_surv_dist(all_deck_per)
```

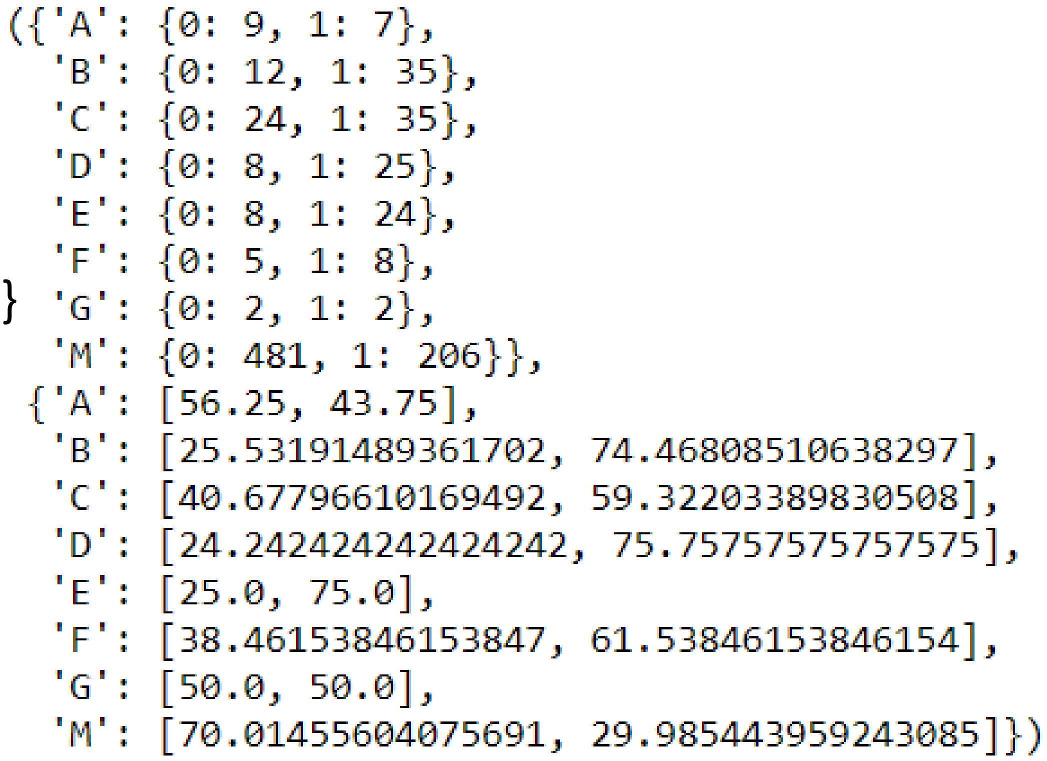

Passenger Class Distribution in Decks
Passenger Class Percentage
0
20
40
60
80
100
A
B
C
D
E
F
G
M
T
Deck
Passenger Class 1
Passenger Class 2
Passenger Class 3

```
idx = df_all[df_all['Deck']=='T'].index
df_all.loc[idx,'Deck'] = 'A‘
df_all.isnull().sum()
```



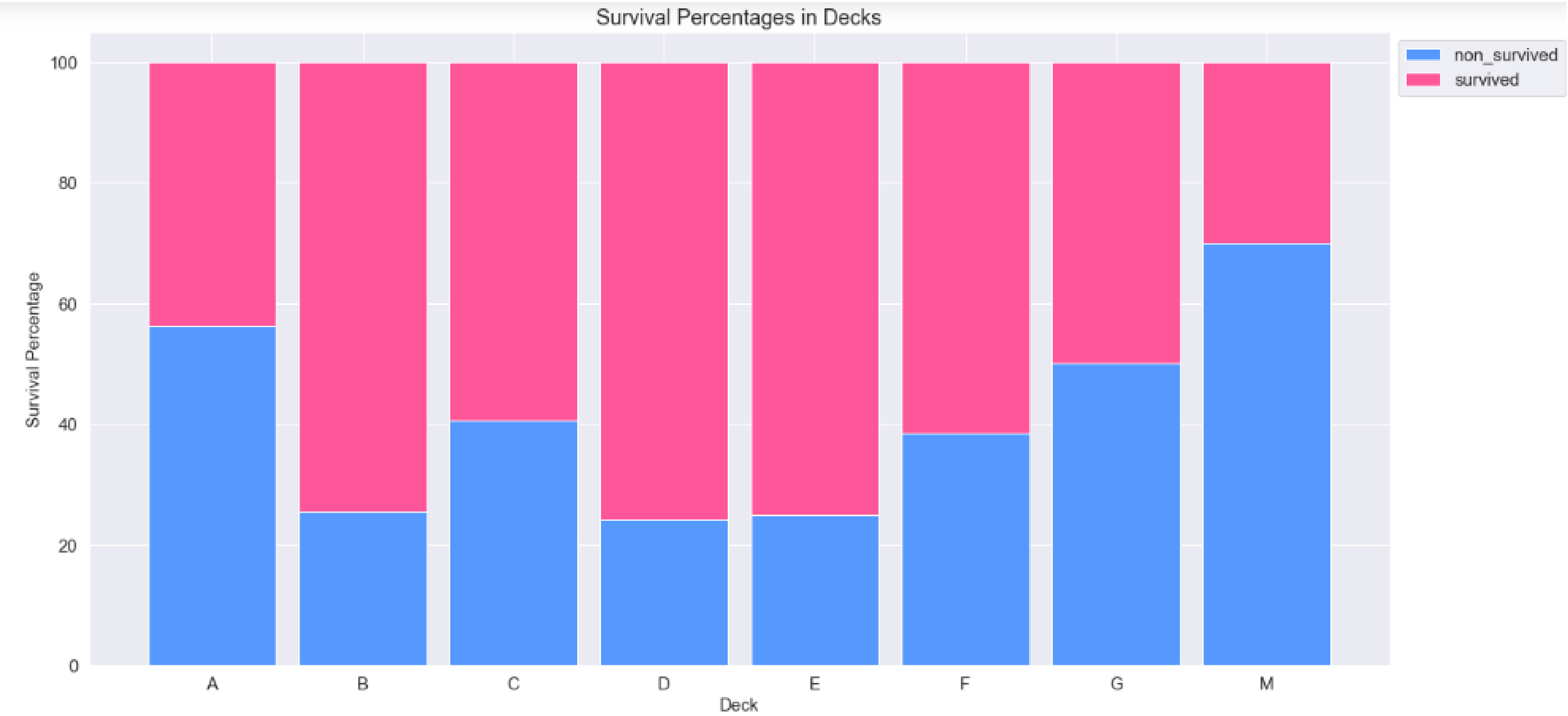
```
Age               0
Cabin          1014
Embarked          0
Fare              0
Name              0
Parch             0
PassengerId       0
Pclass            0
Sex               0
SibSp             0
Survived        418
Ticket            0
Deck              0
dtype: int64
```

```
df_all_decks_survived =
df_all.groupby(['Deck','Survived']).count().drop(columns=['Age','Cabin',
'Embarked','Name','PassengerId','Pclass','Sex','SibSp','Ticket']).rename(columns={
'Name':'Count'}).T

def get_survived_dist(df):
    decks = df.columns.levels[0]
    surv_counts = {'A':{},'B':{},'C':{},'D':{},'E':{},'F':{},'G':{},'M':{}}
    for deck in decks:
        for surv in range(2):
            surv_counts[deck][surv] = df[deck][surv][0]
    df_surv = pd.DataFrame(surv_counts)
    surv_per = {}
    for col in df_surv.columns:
        surv_per[col] = [(count / df_surv[col].sum()) * 100 for count in df_surv[col]]
    return surv_counts, surv_per
all_surv_count, all_surv_per = get_survived_dist(df_all_decks_survived)
all_surv_count, all_surv_per
```



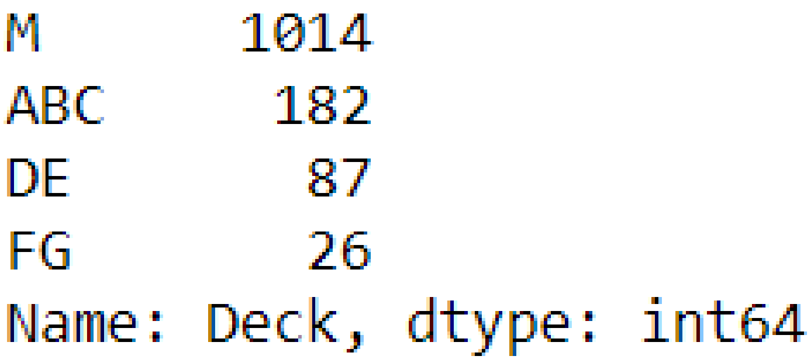
```
({'A': {0: 9, 1: 7},
  'B': {0: 12, 1: 35},
  'C': {0: 24, 1: 35},
  'D': {0: 8, 1: 25},
  'E': {0: 8, 1: 24},
  'F': {0: 5, 1: 8},
  'G': {0: 2, 1: 2},
  'M': {0: 481, 1: 206}},
 {'A': [56.25, 43.75],
  'B': [25.53191489361702, 74.46808510638297],
  'C': [40.67796610169492, 59.32203389830508],
  'D': [24.242424242424242, 75.75757575757575],
  'E': [25.0, 75.0],
  'F': [38.46153846153847, 61.53846153846154],
  'G': [50.0, 50.0],
  'M': [70.01455604075691, 29.985443959243085]})
```

```
def display_surv_dist(per):
    df_surv_per = pd.DataFrame(per).T
    deck_names = ['A','B','C','D','E','F','G','M']
    bar_count = np.arange(len(deck_names))

    survived = df_surv_per[1]
    non_survived = df_surv_per[0]

    mp.figure(figsize=(20,10))
    mp.title('Survival Percentages in Decks', fontsize=18)
    mp.xlabel('Deck',fontsize=15)
    mp.ylabel('Survival Percentage',fontsize=15)
    mp.xticks(bar_count,deck_names)
    mp.tick_params(labelsize=15)
    mp.bar(bar_count, non_survived, width=0.85, color='#5599ff', edgecolor='White', label='non_survived')
    mp.bar(bar_count, survived, bottom=non_survived, width=0.85, color='#ff5599', edgecolor='White', label='survived')
    mp.legend(bbox_to_anchor=(1, 1), prop={'size': 15})
    mp.show()
display_surv_dist(all_surv_per)
```


Survival Percentages in Decks
non_survived
survived
Survival Percentage
0
20
40
60
80
100
A
B
C
D
E
F
G
M
Deck

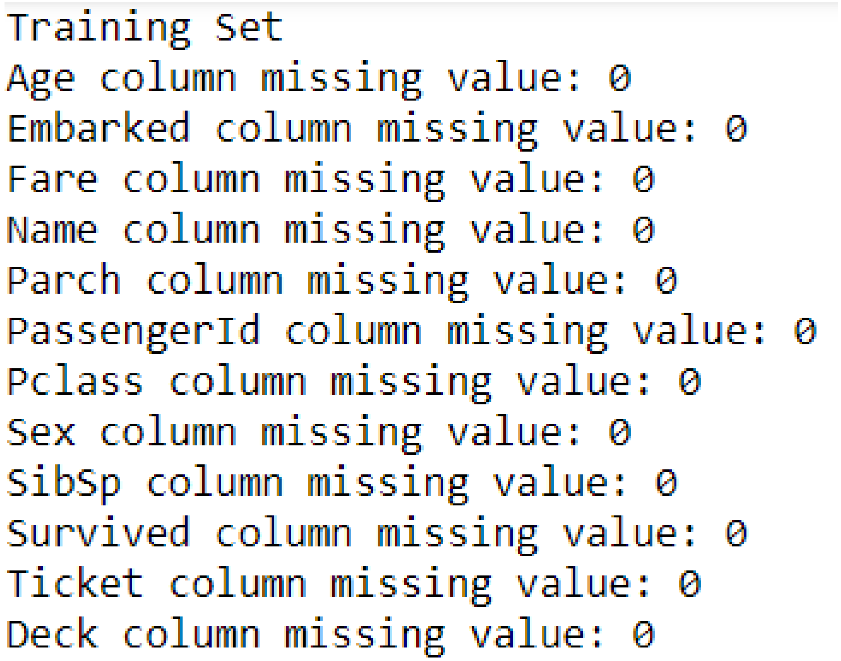
```
df_all['Deck'] = df_all['Deck'].replace(['A','B','C'],'ABC')
df_all['Deck'] = df_all['Deck'].replace(['D','E'],'DE')
df_all['Deck'] = df_all['Deck'].replace(['F','G'],'FG')
```

```
df_all['Deck'].value_counts()
```

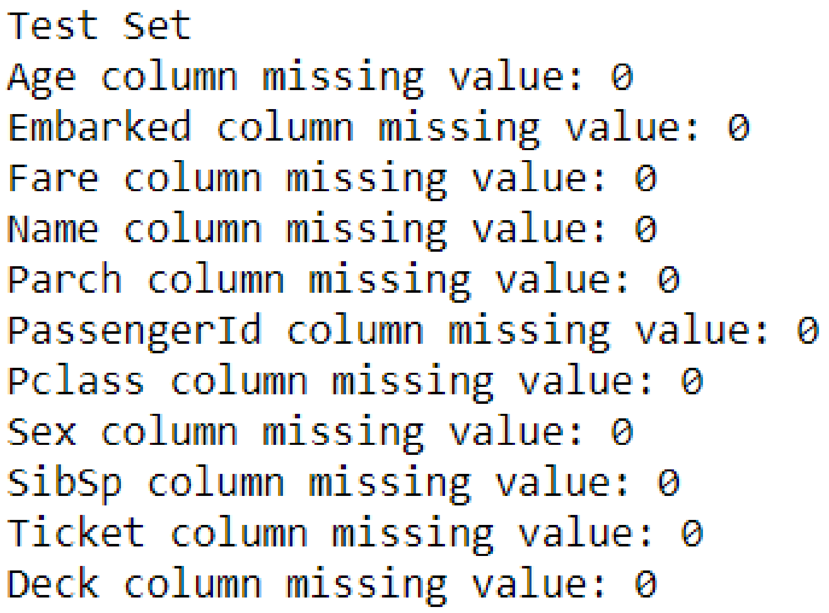


```
M      1014
ABC     182
DE       87
FG       26
Name: Deck, dtype: int64
```

```
df_all.drop('Cabin',inplace=True,axis=1)
df_train, df_test = divide_df(df_all)
df_train.name = 'Training Set'
df_test.name = 'Test Set'
dfs = [df_train, df_test]

for df in dfs:
    print('{}'.format(df.name))
    display_missing(df)
```

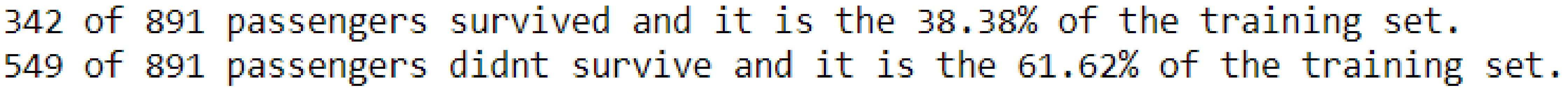


```
Training Set
Age column missing value: 0
Embarked column missing value: 0
Fare column missing value: 0
Name column missing value: 0
Parch column missing value: 0
PassengerId column missing value: 0
Pclass column missing value: 0
Sex column missing value: 0
SibSp column missing value: 0
Survived column missing value: 0
Ticket column missing value: 0
Deck column missing value: 0
```

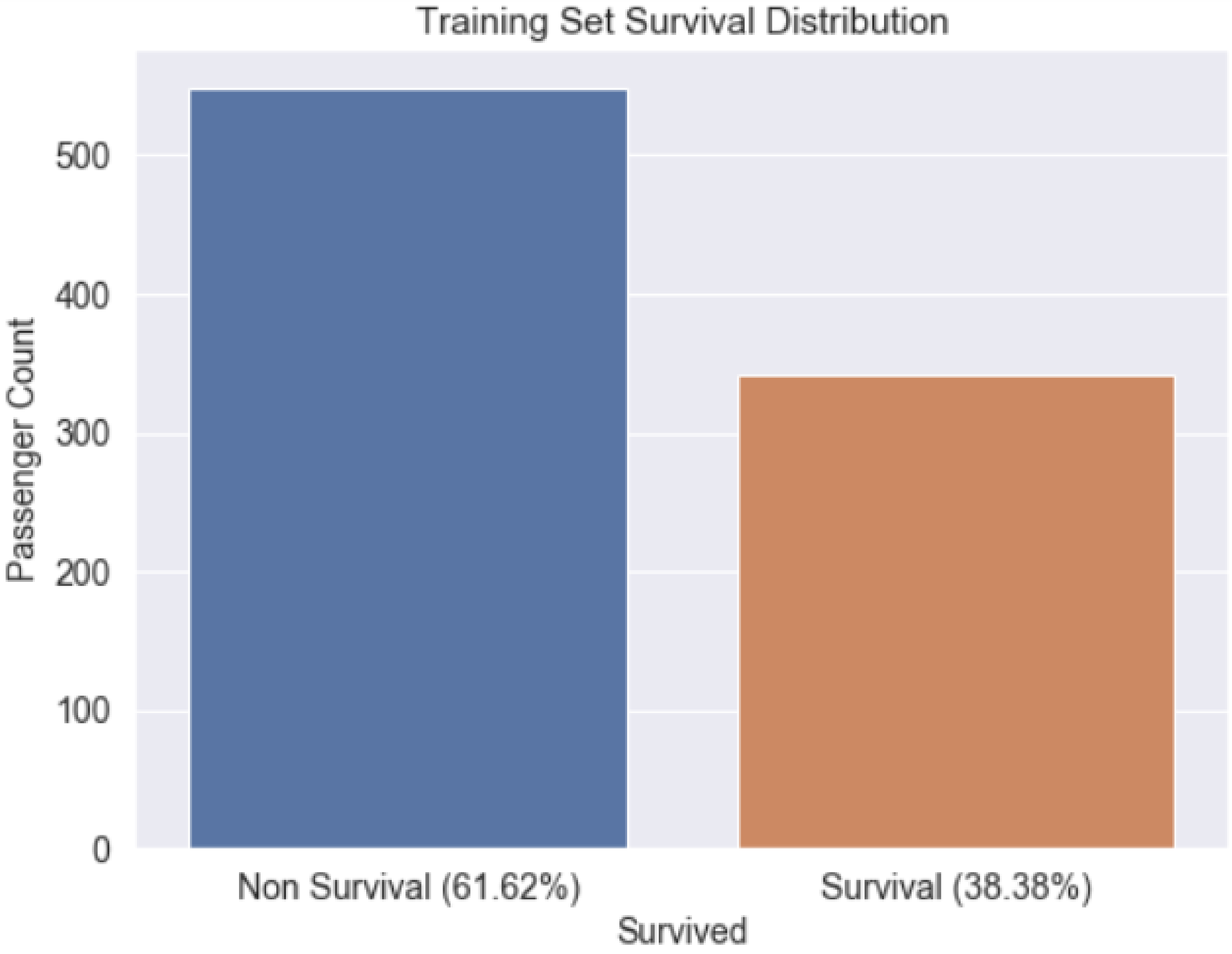


```
Test Set
Age column missing value: 0
Embarked column missing value: 0
Fare column missing value: 0
Name column missing value: 0
Parch column missing value: 0
PassengerId column missing value: 0
Pclass column missing value: 0
Sex column missing value: 0
SibSp column missing value: 0
Ticket column missing value: 0
Deck column missing value: 0
```

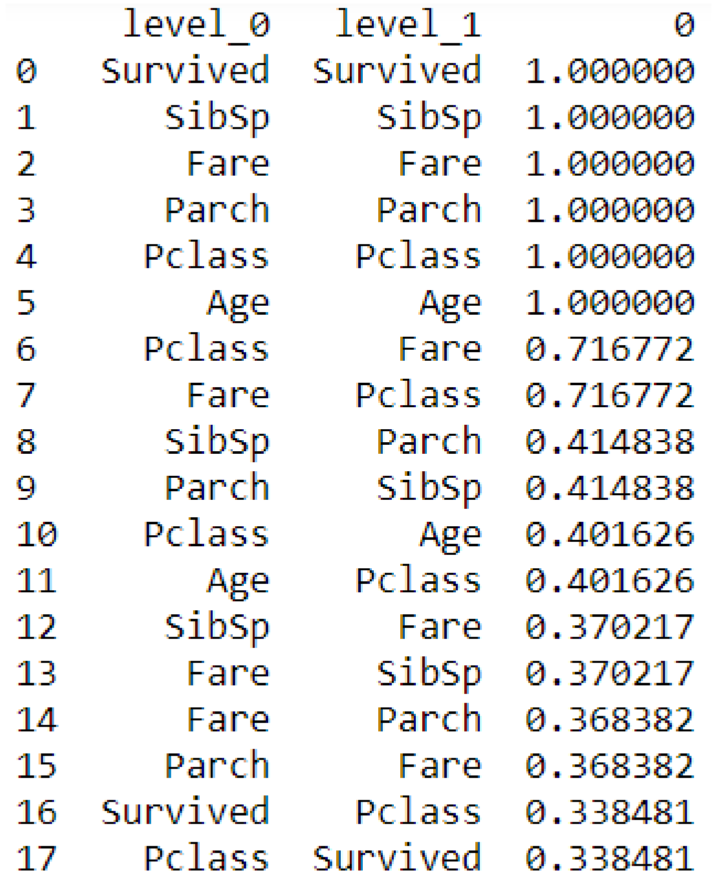
```python
survived = df_train['Survived'].value_counts()[1]
non_survived = df_train['Survived'].value_counts()[0]
sur_per = survived / df_train.shape[0] * 100
nonsur_per = non_survived / df_train.shape[0] * 100
print('{} of {} passengers survived and it is the {:.2f}% of the training set.'.format(survived,
df_train.shape[0],sur_per))
print('{} of {} passengers didnt survive and it is the {:.2f}% of the training set.'.format(non_survived,
df_train.shape[0],nonsur_per))

mp.figure(figsize=(10,8))
sns.countplot(df_train['Survived'])
mp.title('Training Set Survival Distribution', fontsize=15)
mp.xlabel('Survived',fontsize=14)
mp.ylabel('Passenger Count',fontsize=14)
mp.xticks((0,1),['Non Survival ({:.2f}%)'.format(nonsur_per), 'Survival ({:.2f}%)'.format(sur_per)])
mp.tick_params(labelsize=14)

mp.show()
```



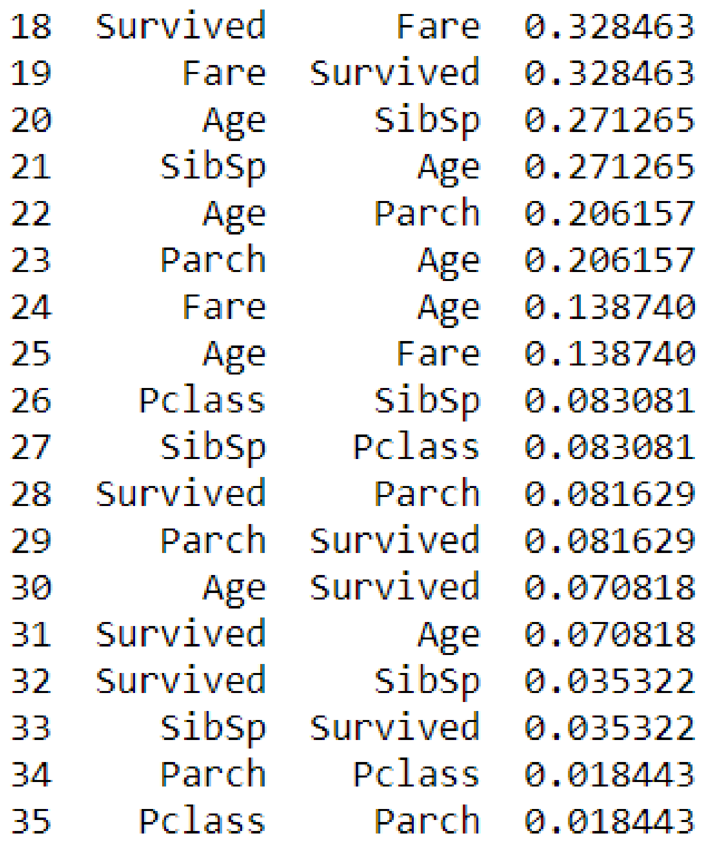
```
342 of 891 passengers survived and it is the 38.38% of the training set.
549 of 891 passengers didnt survive and it is the 61.62% of the training set.
```



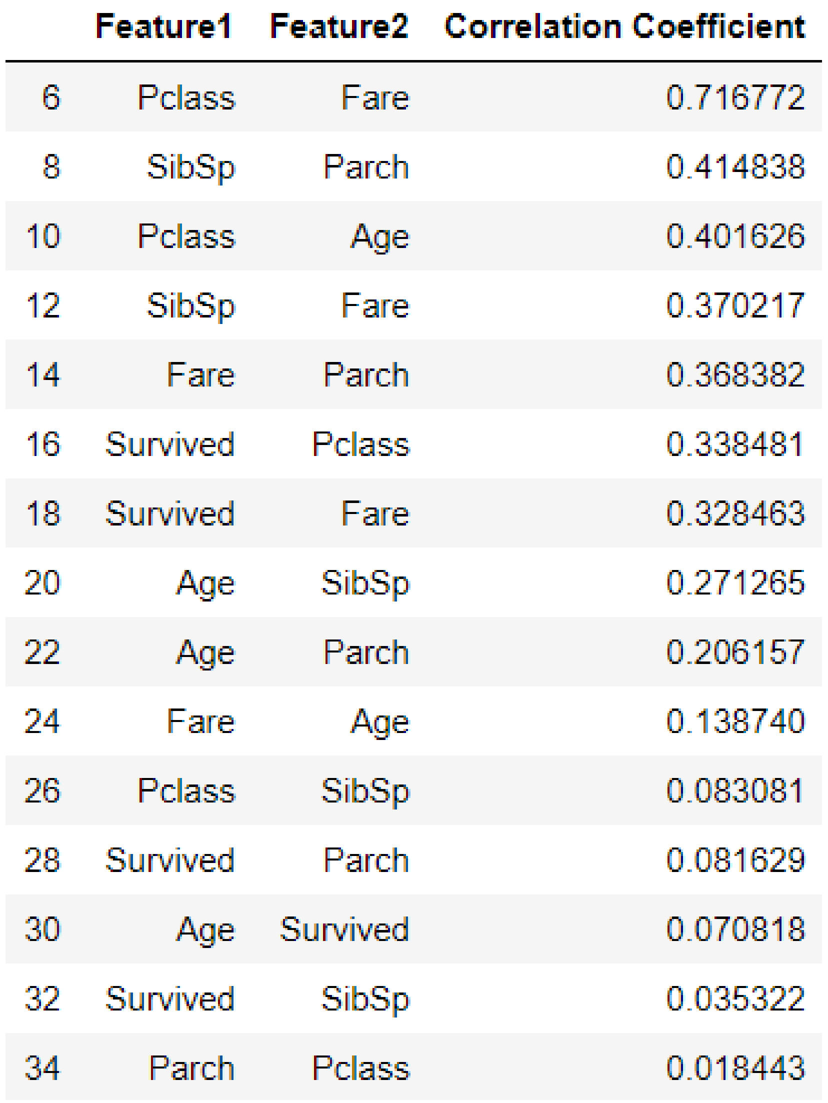
```
df_train_corr = df_train.drop(['PassengerId'],axis=1).corr().abs()
.unstack().sort_values(kind='quicksort',ascending=False).reset_index()

print(df_train_corr)


df_train_corr.rename(columns={'level_0':'Feature1','level_1':'Feature2',
0:'Correlation Coefficient'},inplace=True)
df_train_corr.drop(df_train_corr.iloc[1::2].index, inplace=True)
df_train_corr_nd = df_train_corr.drop(df_train_corr[
df_train_corr['Correlation Coefficient'] == 1.0].index)

print(df_train_corr_nd)
```



```
      level_0   level_1         0
0    Survived  Survived  1.000000
1       SibSp     SibSp  1.000000
2        Fare      Fare  1.000000
3       Parch     Parch  1.000000
4      Pclass    Pclass  1.000000
5         Age       Age  1.000000
6      Pclass      Fare  0.716772
7        Fare    Pclass  0.716772
8       SibSp     Parch  0.414838
9       Parch     SibSp  0.414838
10     Pclass       Age  0.401626
11        Age    Pclass  0.401626
12      SibSp      Fare  0.370217
13       Fare     SibSp  0.370217
14       Fare     Parch  0.368382
15      Parch      Fare  0.368382
16   Survived    Pclass  0.338481
17     Pclass  Survived  0.338481
```

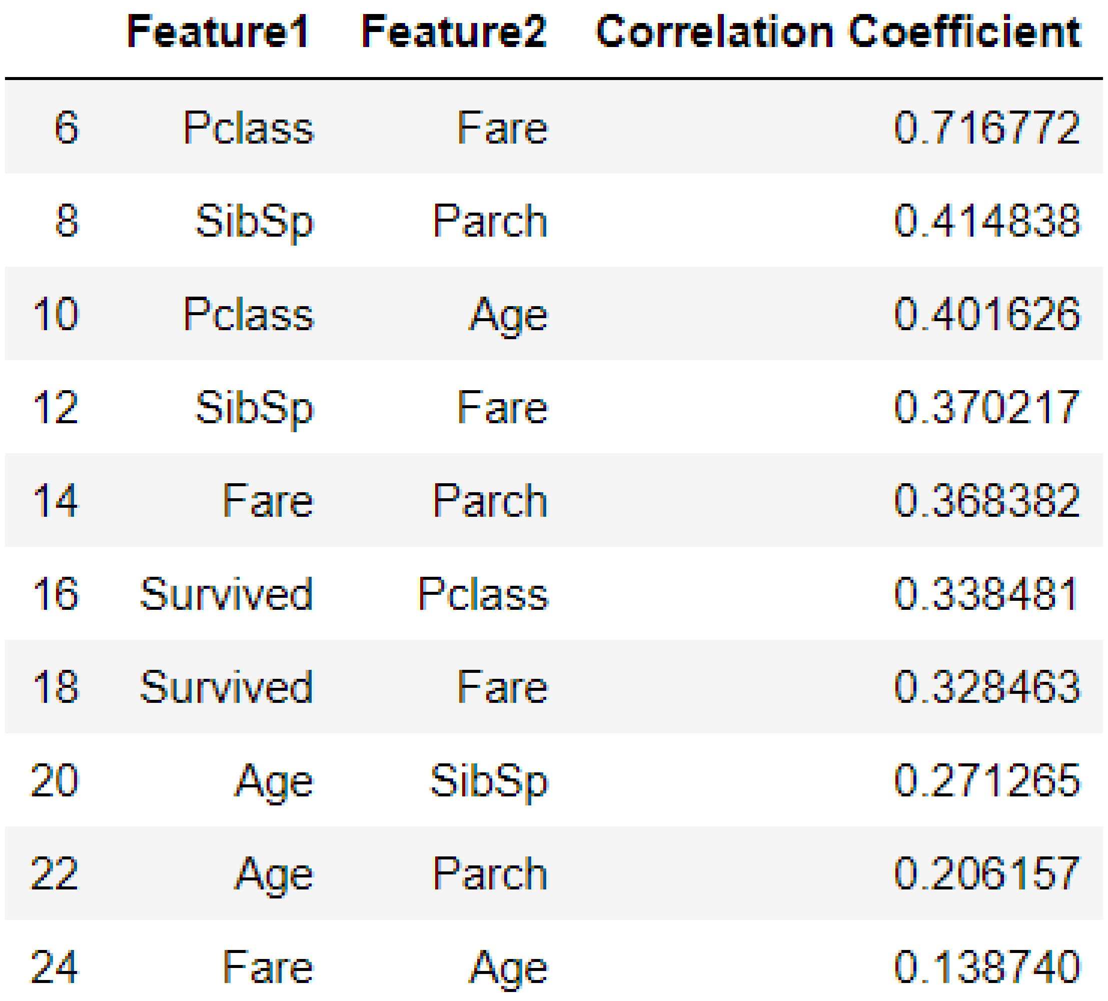


```
18   Survived      Fare  0.328463
19       Fare  Survived  0.328463
20        Age     SibSp  0.271265
21      SibSp       Age  0.271265
22        Age     Parch  0.206157
23      Parch       Age  0.206157
24       Fare       Age  0.138740
25        Age      Fare  0.138740
26     Pclass     SibSp  0.083081
27      SibSp    Pclass  0.083081
28   Survived     Parch  0.081629
29      Parch  Survived  0.081629
30        Age  Survived  0.070818
31   Survived       Age  0.070818
32   Survived     SibSp  0.035322
33      SibSp  Survived  0.035322
34      Parch    Pclass  0.018443
35     Pclass     Parch  0.018443
```



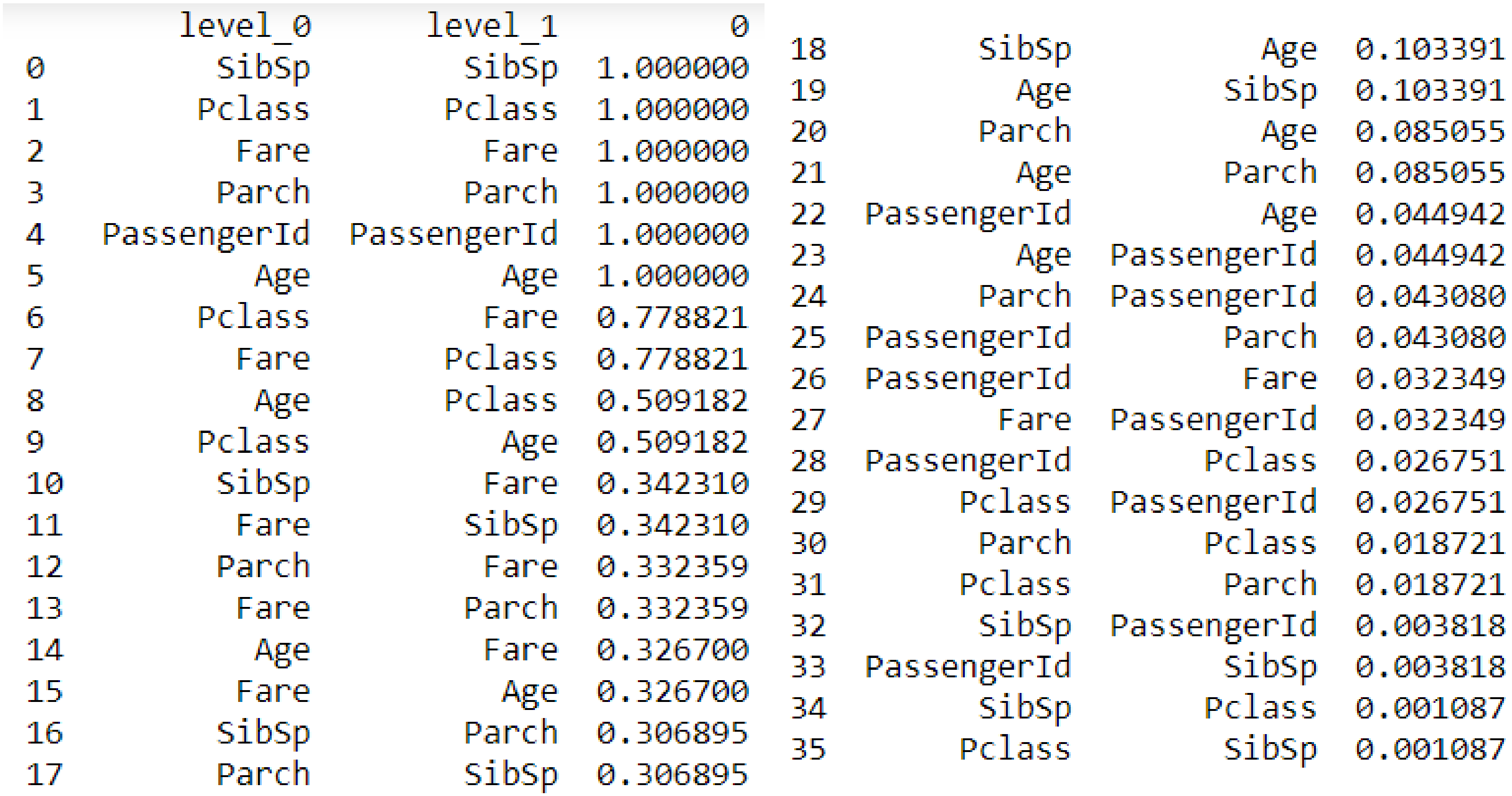
	Feature1	Feature2	Correlation Coefficient
6	Pclass	Fare	0.716772
8	SibSp	Parch	0.414838
10	Pclass	Age	0.401626
12	SibSp	Fare	0.370217
14	Fare	Parch	0.368382
16	Survived	Pclass	0.338481
18	Survived	Fare	0.328463
20	Age	SibSp	0.271265
22	Age	Parch	0.206157
24	Fare	Age	0.138740
26	Pclass	SibSp	0.083081
28	Survived	Parch	0.081629
30	Age	Survived	0.070818
32	Survived	SibSp	0.035322
34	Parch	Pclass	0.018443

```
corr = df_train_corr_nd['Correlation Coefficient'] > 0.1
df_train_corr_nd[corr]
```

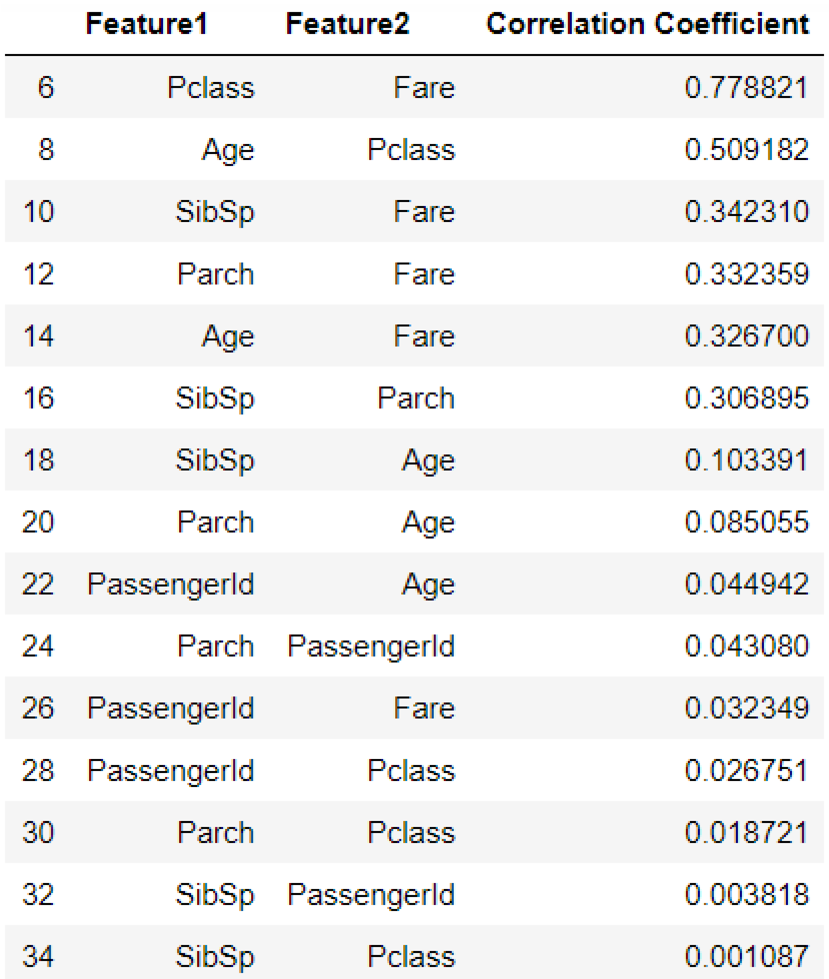


	Feature1	Feature2	Correlation Coefficient
6	Pclass	Fare	0.716772
8	SibSp	Parch	0.414838
10	Pclass	Age	0.401626
12	SibSp	Fare	0.370217
14	Fare	Parch	0.368382
16	Survived	Pclass	0.338481
18	Survived	Fare	0.328463
20	Age	SibSp	0.271265
22	Age	Parch	0.206157
24	Fare	Age	0.138740

```
df_test_corr =
df_test.corr().abs().unstack().sort_values(kind='quicksort',
ascending=False).reset_index()

print(df_test_corr)

df_test_corr.rename(columns={'level_0':'Feature1','level_1':'Feature2',
0:'Correlation Coefficient'},

inplace=True)

df_test_corr.drop(df_test_corr.iloc[1::2].index, inplace=True)

df_test_corr_nd =
df_test_corr.drop(df_test_corr[df_test_corr['Correlation Coefficient']
== 1.0].index)

df_test_corr_nd
```

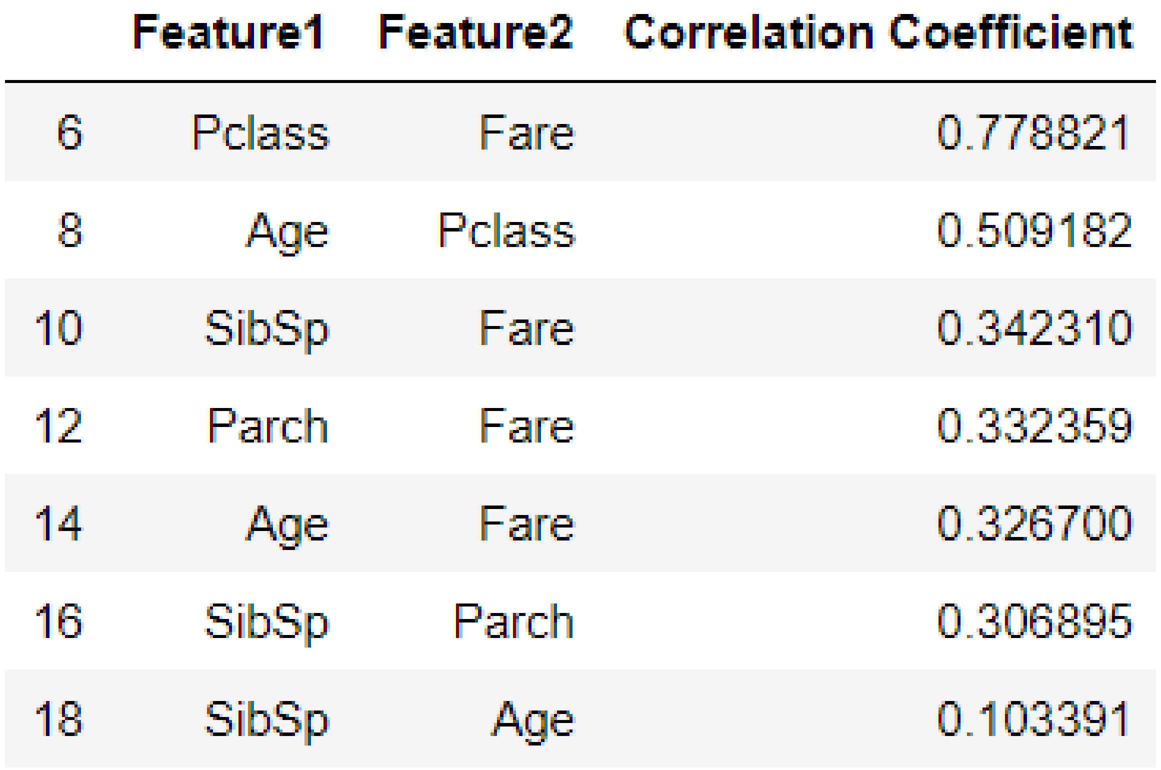


```
         level_0      level_1         0
0          SibSp        SibSp  1.000000
1         Pclass       Pclass  1.000000
2           Fare         Fare  1.000000
3          Parch        Parch  1.000000
4    PassengerId  PassengerId  1.000000
5            Age          Age  1.000000
6         Pclass         Fare  0.778821
7           Fare       Pclass  0.778821
8            Age       Pclass  0.509182
9         Pclass          Age  0.509182
10         SibSp         Fare  0.342310
11          Fare        SibSp  0.342310
12         Parch         Fare  0.332359
13          Fare        Parch  0.332359
14           Age         Fare  0.326700
15          Fare          Age  0.326700
16         SibSp        Parch  0.306895
17         Parch        SibSp  0.306895
18         SibSp          Age  0.103391
19           Age        SibSp  0.103391
20         Parch          Age  0.085055
21           Age        Parch  0.085055
22   PassengerId          Age  0.044942
23           Age  PassengerId  0.044942
24         Parch  PassengerId  0.043080
25   PassengerId        Parch  0.043080
26   PassengerId         Fare  0.032349
27          Fare  PassengerId  0.032349
28   PassengerId       Pclass  0.026751
29        Pclass  PassengerId  0.026751
30         Parch       Pclass  0.018721
31        Pclass        Parch  0.018721
32         SibSp  PassengerId  0.003818
33   PassengerId        SibSp  0.003818
34         SibSp       Pclass  0.001087
35        Pclass        SibSp  0.001087
```



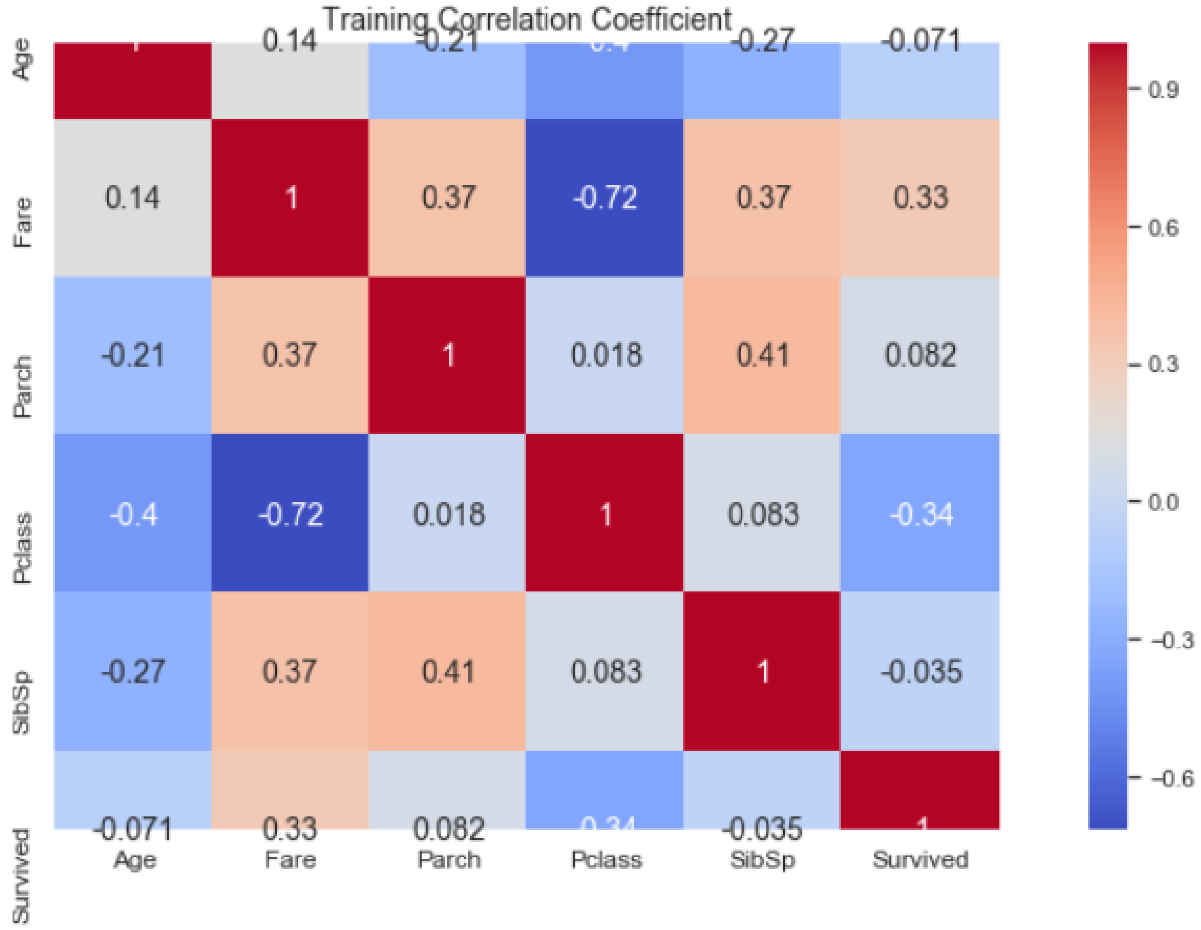
	Feature1	Feature2	Correlation Coefficient
6	Pclass	Fare	0.778821
8	Age	Pclass	0.509182
10	SibSp	Fare	0.342310
12	Parch	Fare	0.332359
14	Age	Fare	0.326700
16	SibSp	Parch	0.306895
18	SibSp	Age	0.103391
20	Parch	Age	0.085055
22	PassengerId	Age	0.044942
24	Parch	PassengerId	0.043080
26	PassengerId	Fare	0.032349
28	PassengerId	Pclass	0.026751
30	Parch	Pclass	0.018721
32	SibSp	PassengerId	0.003818
34	SibSp	Pclass	0.001087

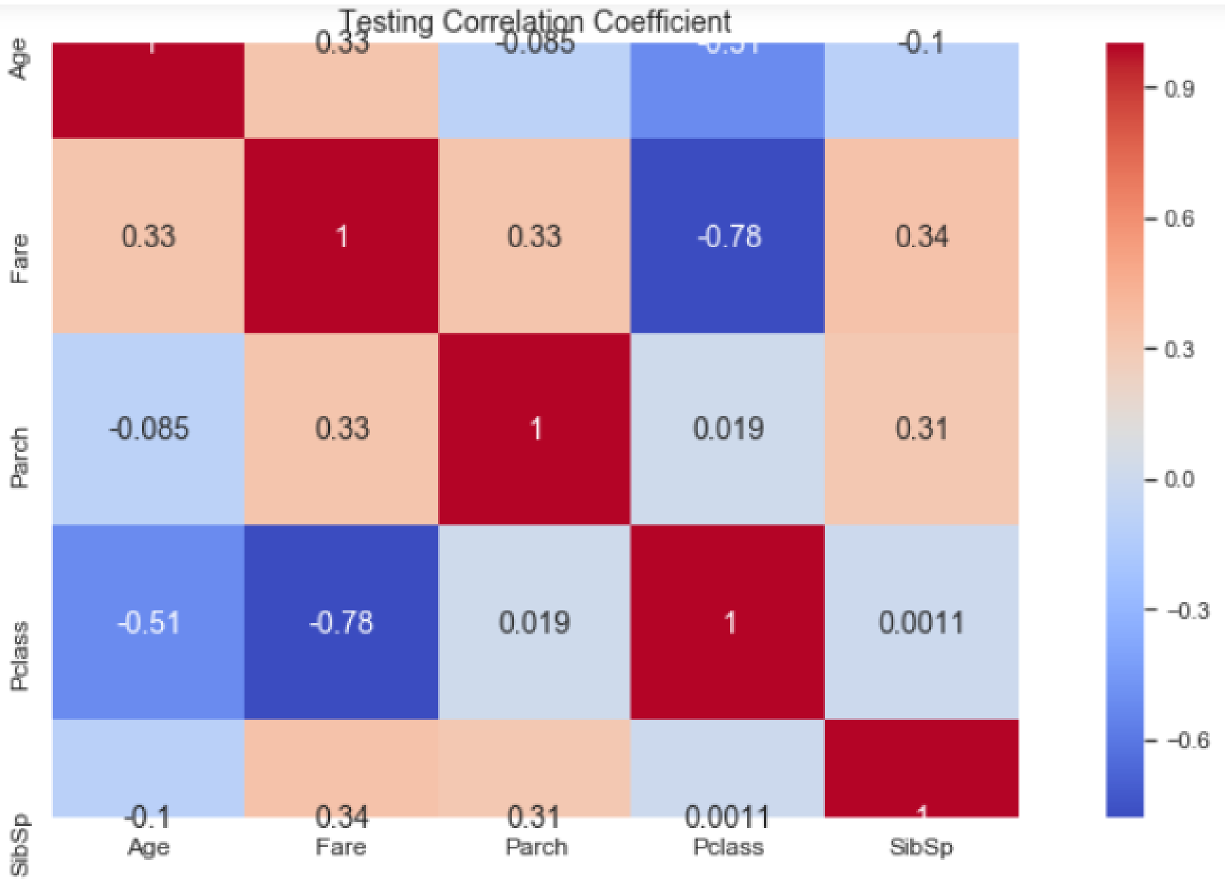
```
corr = df_test_corr_nd['Correlation Coefficient'] > 0.1
df_test_corr_nd[corr]
```



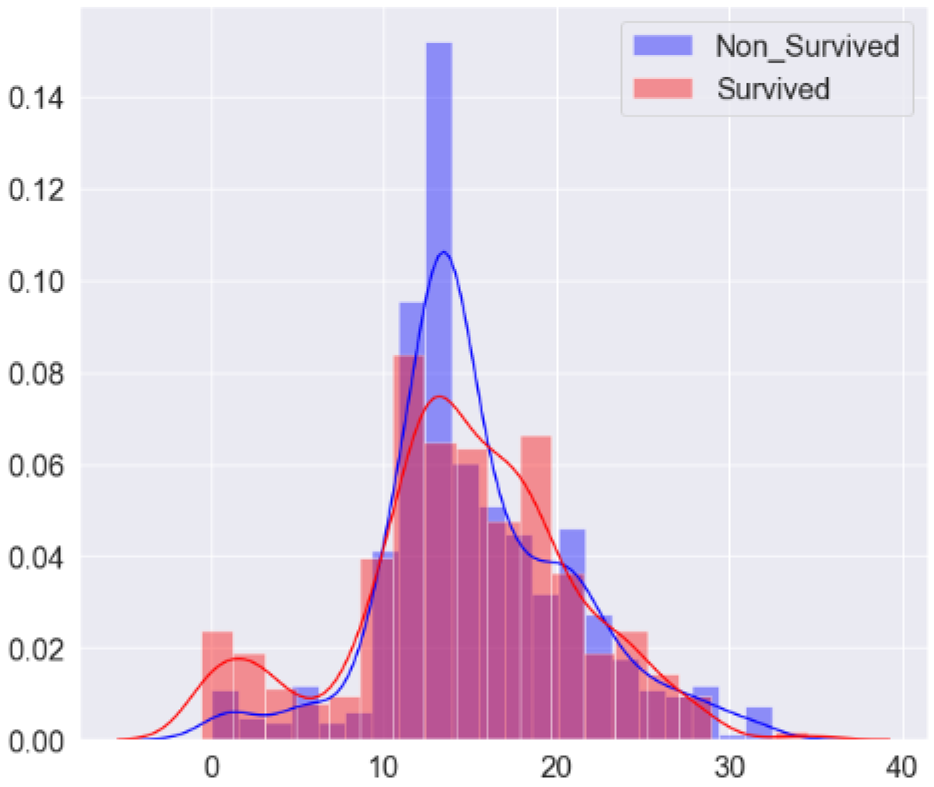
	Feature1	Feature2	Correlation Coefficient
6	Pclass	Fare	0.778821
8	Age	Pclass	0.509182
10	SibSp	Fare	0.342310
12	Parch	Fare	0.332359
14	Age	Fare	0.326700
16	SibSp	Parch	0.306895
18	SibSp	Age	0.103391

```
fig, axs = mp.subplots(nrows=2, figsize=(15, 15))

sns.heatmap(df_train.drop(['PassengerId'],axis=1).corr(), ax=axs[0],
cmap='coolwarm',  square=True,  annot=True, annot_kws={'size':14})
sns.heatmap(df_test.drop(['PassengerId'],axis=1).corr(), ax=axs[1],
cmap='coolwarm',  square=True,  annot=True, annot_kws={'size':14})

for i in range(2):
    axs[i].tick_params(labelsize=14)

axs[0].set_title('Training Correlation Coefficient', fontsize=14)
axs[1].set_title('Testing Correlation Coefficient', fontsize=14)

mp.show()
```

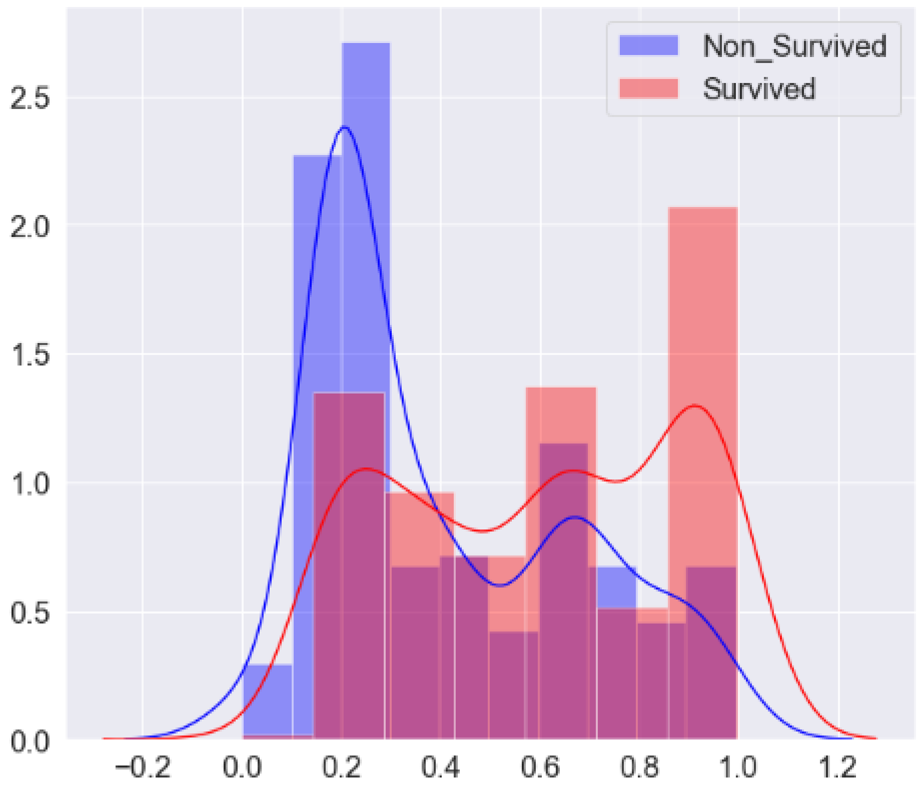

Training Correlation Coefficient
Age
Fare
Parch
Pclass
SibSp
Survived
1
0.14
-0.21
-0.4
-0.27
-0.071
0.14
1
0.37
-0.72
0.37
0.33
-0.21
0.37
1
0.018
0.41
0.082
-0.4
-0.72
0.018
1
0.083
-0.34
-0.27
0.37
0.41
0.083
1
-0.035
-0.071
0.33
0.082
-0.34
-0.035
1
Age
Fare
Parch
Pclass
SibSp
Survived
0.9
0.6
0.3
0.0
−0.3
−0.6


Testing Correlation Coefficient
Age
Fare
Parch
Pclass
SibSp
1
0.33
-0.085
-0.51
-0.1
0.33
1
0.33
-0.78
0.34
-0.085
0.33
1
0.019
0.31
-0.51
-0.78
0.019
1
0.0011
-0.1
0.34
0.31
0.0011
1
Age
Fare
Parch
Pclass
SibSp
0.9
0.6
0.3
0.0
−0.3
−0.6

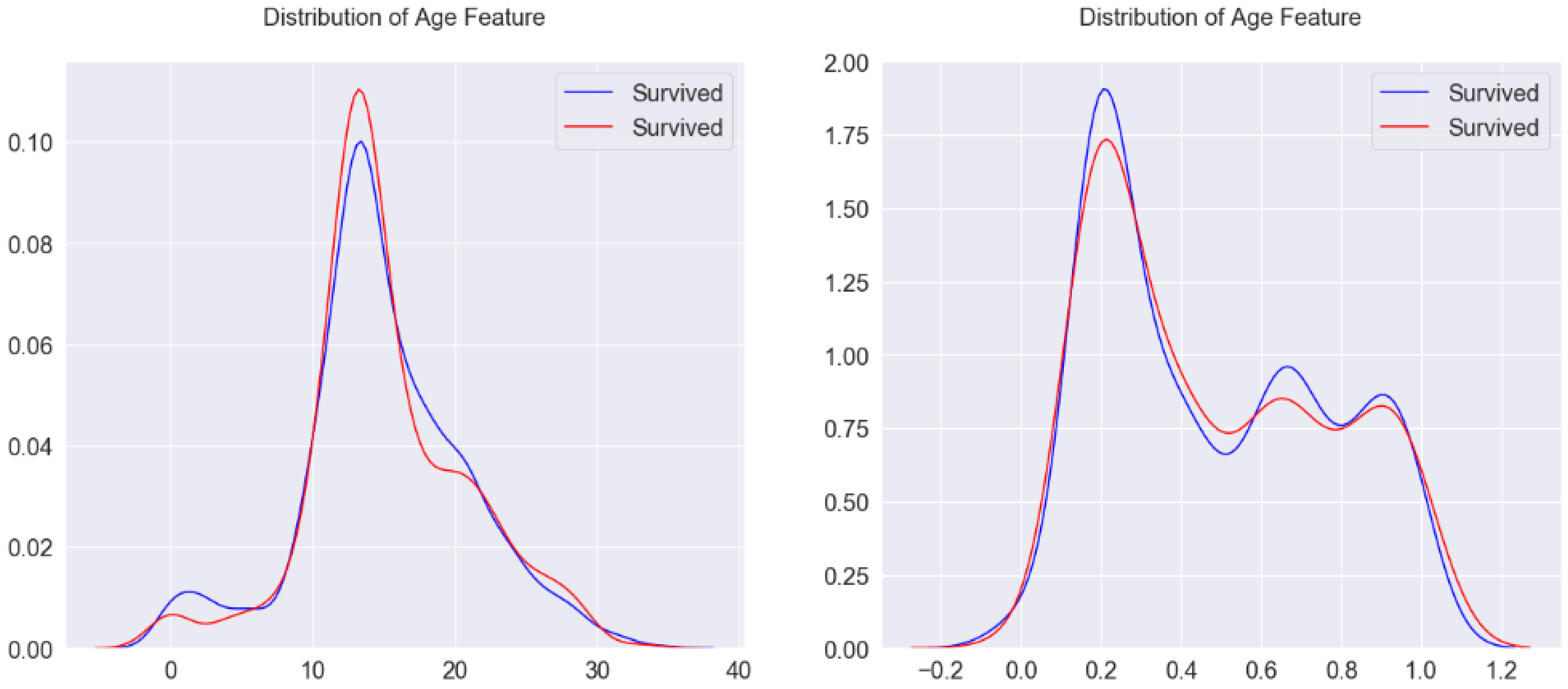
```
cont_feature = ['Age', 'Fare']
surv = df_train['Survived'] == 1
fig, axs = mp.subplots(ncols=2, nrows=2, figsize=(20, 20))
mp.subplots_adjust(right=1)

for i, feature in enumerate(cont_feature):
    sns.distplot(df_train[~surv][feature], label='Non_Survived', hist=True, color='#0000ff', ax=axs[0][i])
    sns.distplot(df_train[surv][feature], label='Survived',hist=True, color='#ff0000' ,ax=axs[0][i])

    sns.distplot(df_train[feature], label='Survived',hist=False, color='#0000ff' ,ax=axs[1][i])
    sns.distplot(df_test[feature], label='Survived',hist=False, color='#ff0000' ,ax=axs[1][i])

    axs[0][i].set_xlabel('')
    axs[1][i].set_xlabel('')

    for j in range(2):
        axs[i][j].tick_params(labelsize=20)
    axs[0][i].legend(loc='upper right', prop={'size': 20})
    axs[1][i].legend(loc='upper right', prop={'size': 20})
    axs[0][i].set_title('Distribution of Survival in {}'.format(feature), size=20, y=1.05)
    axs[1][i].set_title('Distribution of {} Feature'.format('Age'), size=20, y=1.05)
mp.show()
```



Distribution of Survival in Age

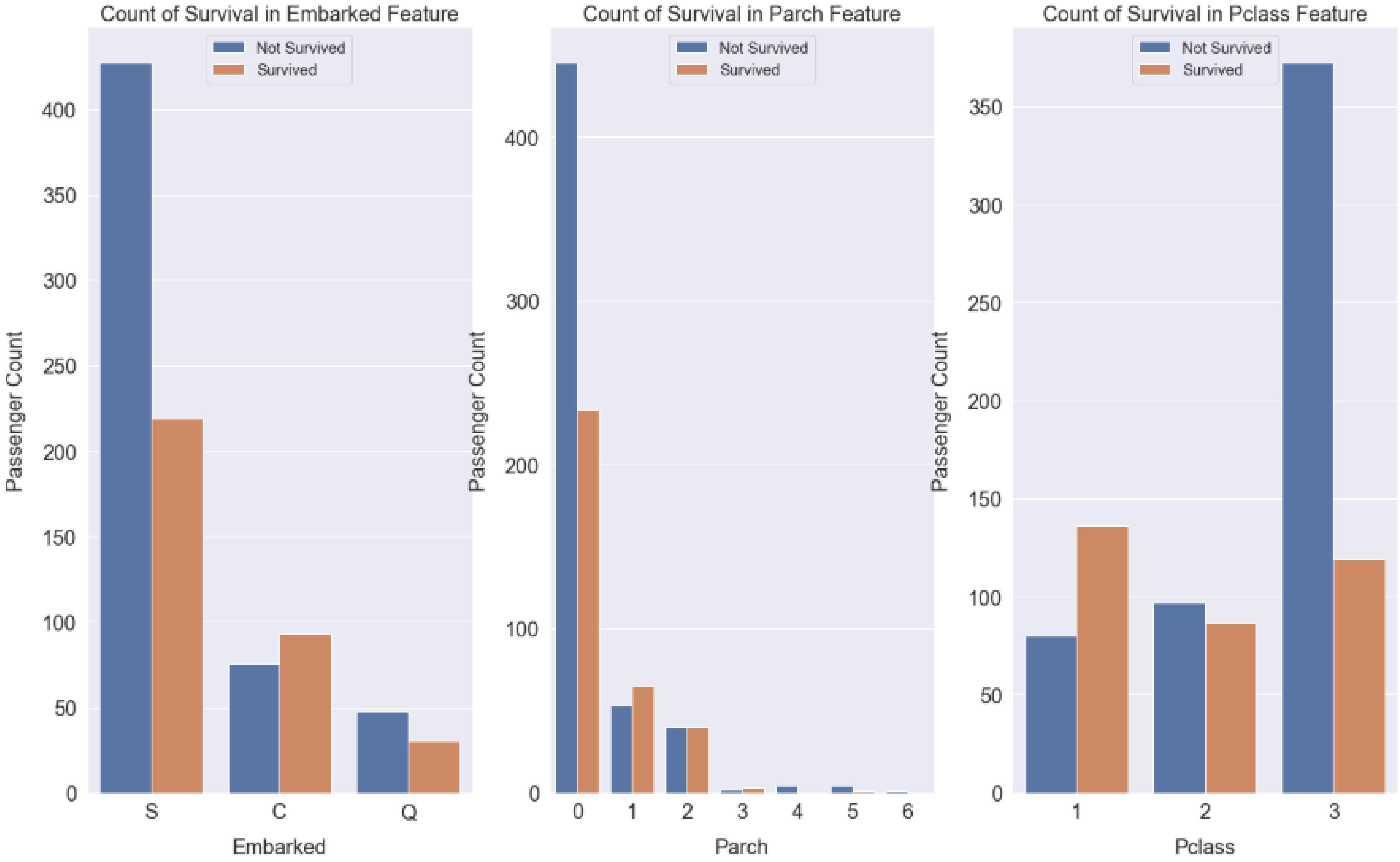


Distribution of Survival in Fare

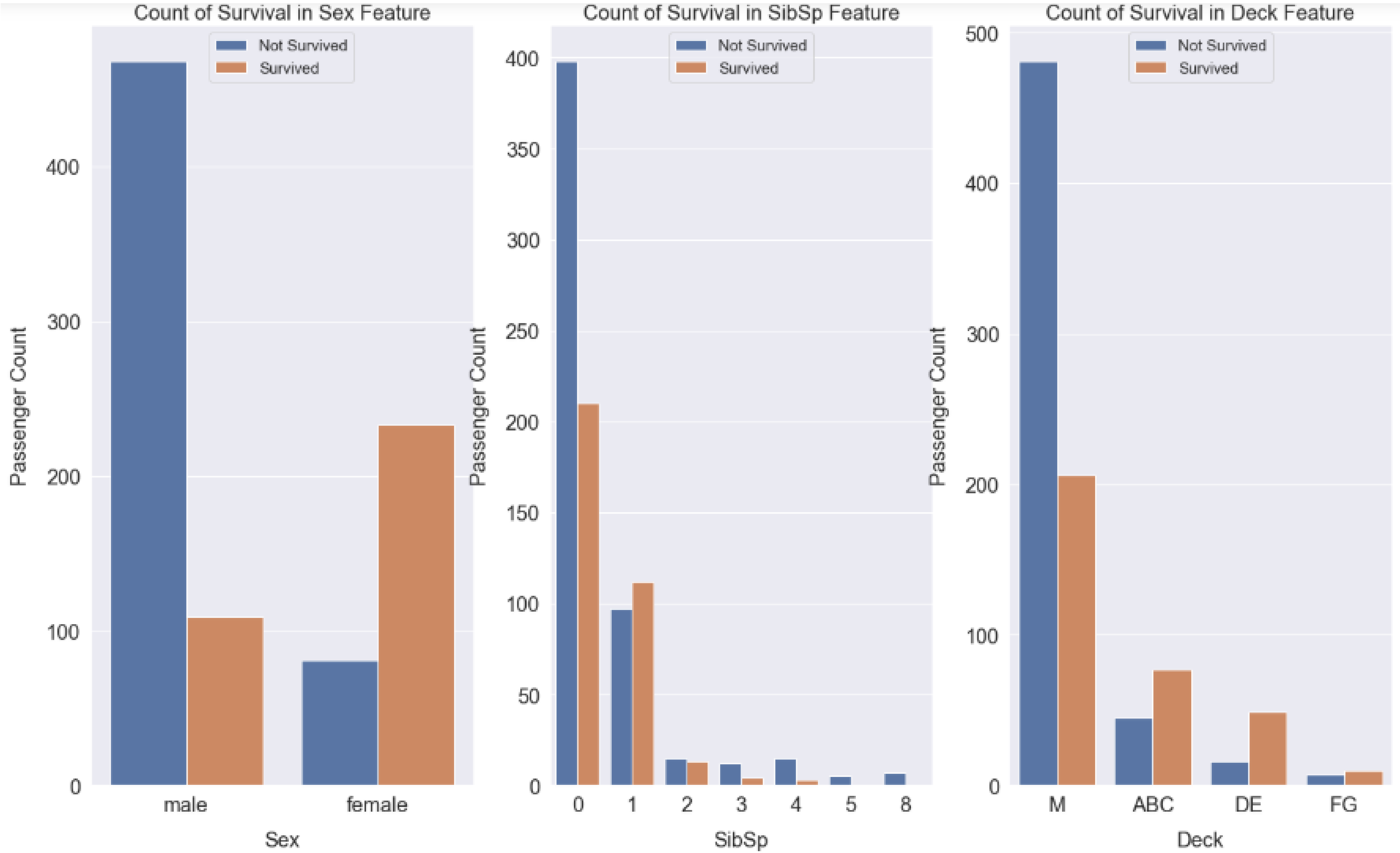


Distribution of Age Feature

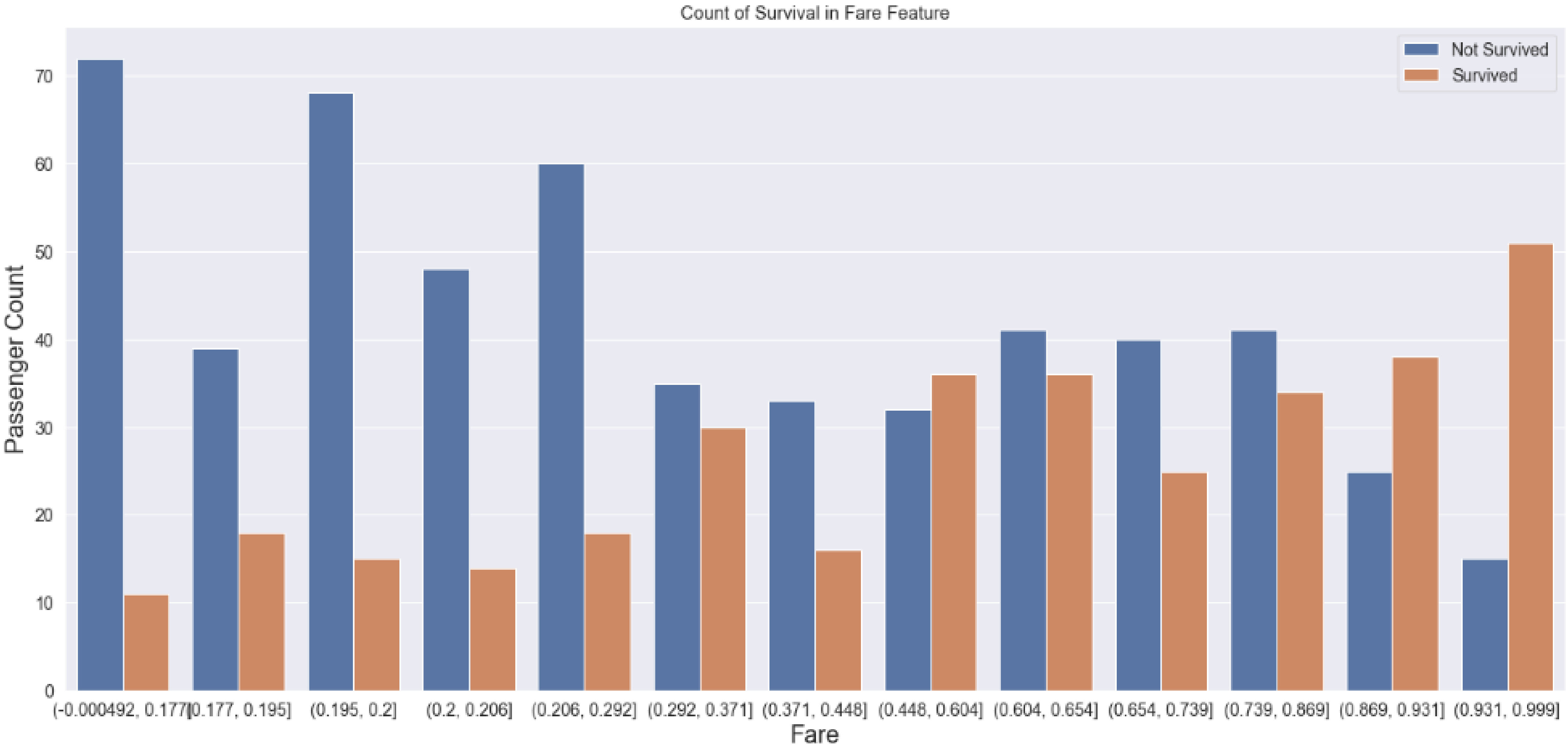
Distribution of Age Feature



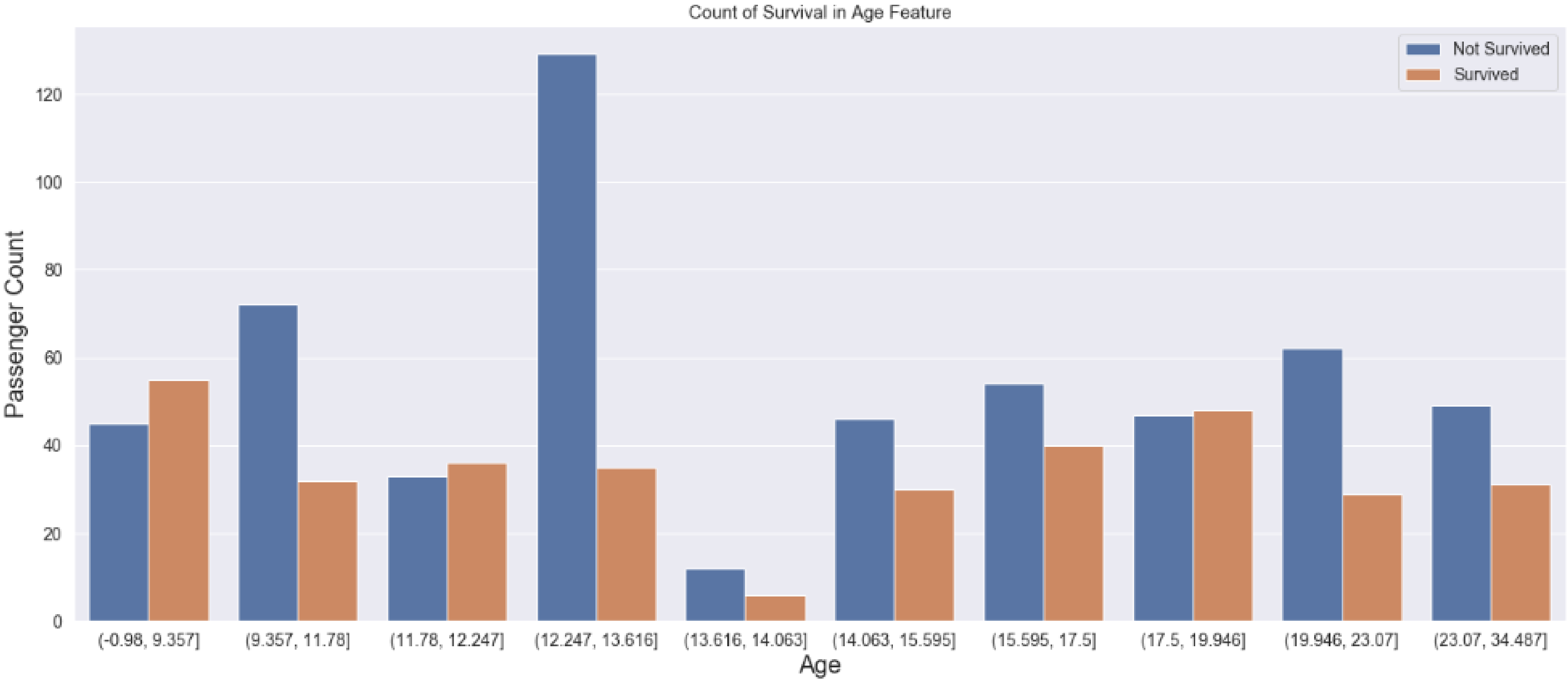
Count of Survival in Embarked Feature
Not Survived
Survived
Passenger Count
0
50
100
150
200
250
300
350
400
S
C
Q
Embarked
Count of Survival in Parch Feature
Not Survived
Survived
Passenger Count
0
100
200
300
400
0
1
2
3
4
5
6
Parch
Count of Survival in Pclass Feature
Not Survived
Survived
Passenger Count
0
50
100
150
200
250
300
350
1
2
3
Pclass


Count of Survival in Sex Feature
Not Survived
Survived
Passenger Count
0
100
200
300
400
male
female
Sex
Count of Survival in SibSp Feature
Not Survived
Survived
Passenger Count
0
50
100
150
200
250
300
350
400
0
1
2
3
4
5
8
SibSp
Count of Survival in Deck Feature
Not Survived
Survived
Passenger Count
0
100
200
300
400
500
M
ABC
DE
FG
Deck

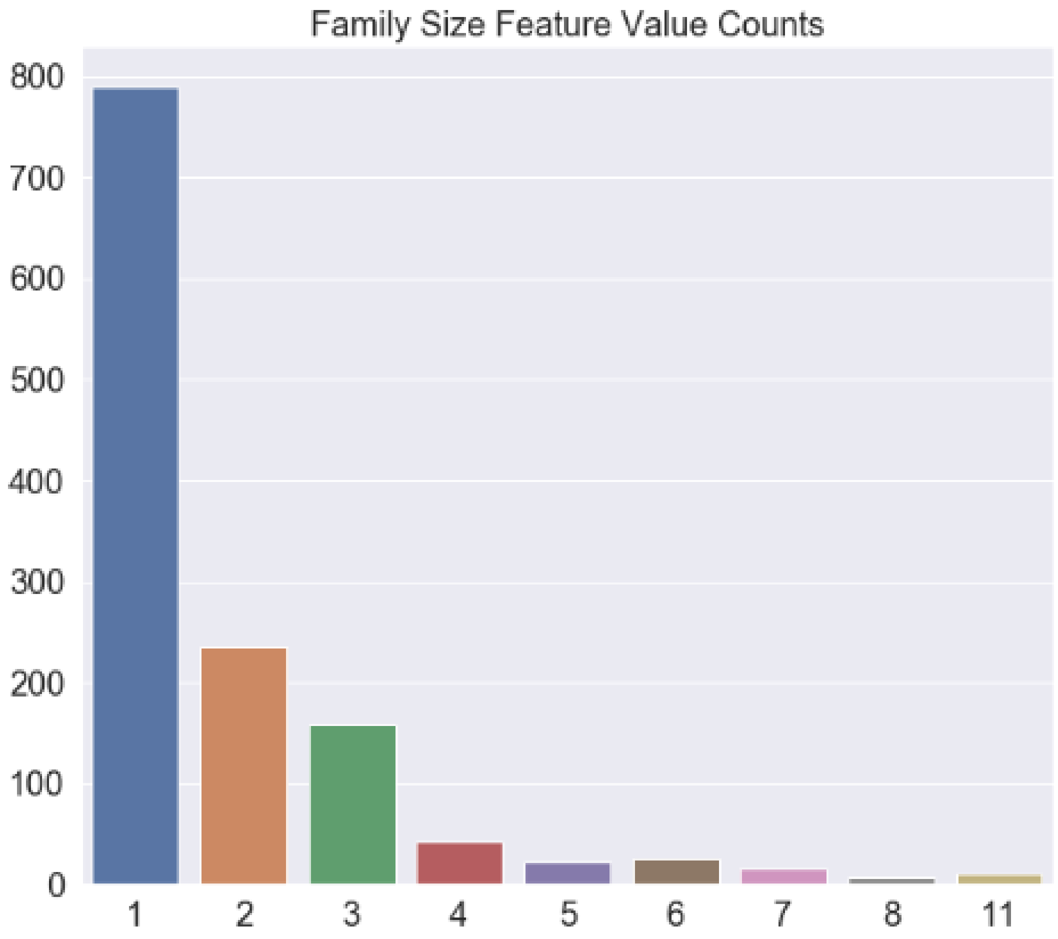
```
df_all = concat_df(df_train, df_test)
df_all['Fare'] = pd.qcut(df_all['Fare'],13)

fig, axs = mp.subplots(figsize=(22, 10))
sns.countplot(x='Fare', hue='Survived', data=df_all)

mp.xlabel('Fare', fontsize=20)
mp.ylabel('Passenger Count', fontsize=20)
mp.tick_params(labelsize=14)
mp.legend(['Not Survived', 'Survived'], loc='upper right',
prop={'size':14})
mp.title('Count of Survival in {} Feature'.format('Fare'), fontsize=15)
mp.show()
```


Count of Survival in Fare Feature
Not Survived
Survived
Passenger Count
70
60
50
40
30
20
10
0
(-0.000492, 0.177]
(0.177, 0.195]
(0.195, 0.2]
(0.2, 0.206]
(0.206, 0.292]
(0.292, 0.371]
(0.371, 0.448]
(0.448, 0.604]
(0.604, 0.654]
(0.654, 0.739]
(0.739, 0.869]
(0.869, 0.931]
(0.931, 0.999]
Fare

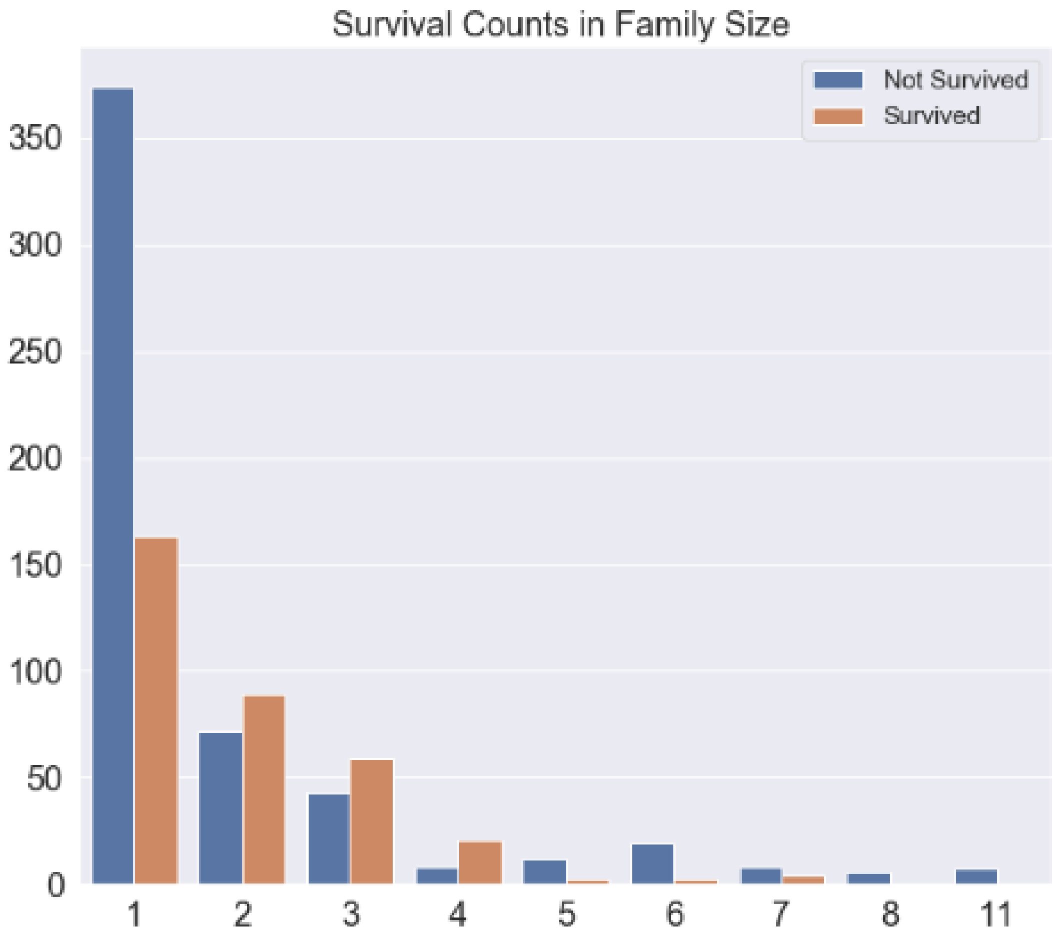
```
df_all['Age'] = pd.qcut(df_all['Age'], 10)

fig, axs = mp.subplots(figsize=(22, 9))
sns.countplot(x='Age', hue='Survived', data=df_all)

mp.xlabel('Age', fontsize=20)
mp.ylabel('Passenger Count', fontsize=20)
mp.tick_params(labelsize=14)
mp.legend(['Not Survived', 'Survived'], loc='upper right',
prop={'size':14})
mp.title('Count of Survival in {} Feature'.format('Age'), fontsize=15)
mp.show()
```


Count of Survival in Age Feature
Not Survived
Survived
Passenger Count
0
20
40
60
80
100
120
(-0.98, 9.357]
(9.357, 11.78]
(11.78, 12.247]
(12.247, 13.616]
(13.616, 14.063]
(14.063, 15.595]
(15.595, 17.5]
(17.5, 19.946]
(19.946, 23.07]
(23.07, 34.487]
Age

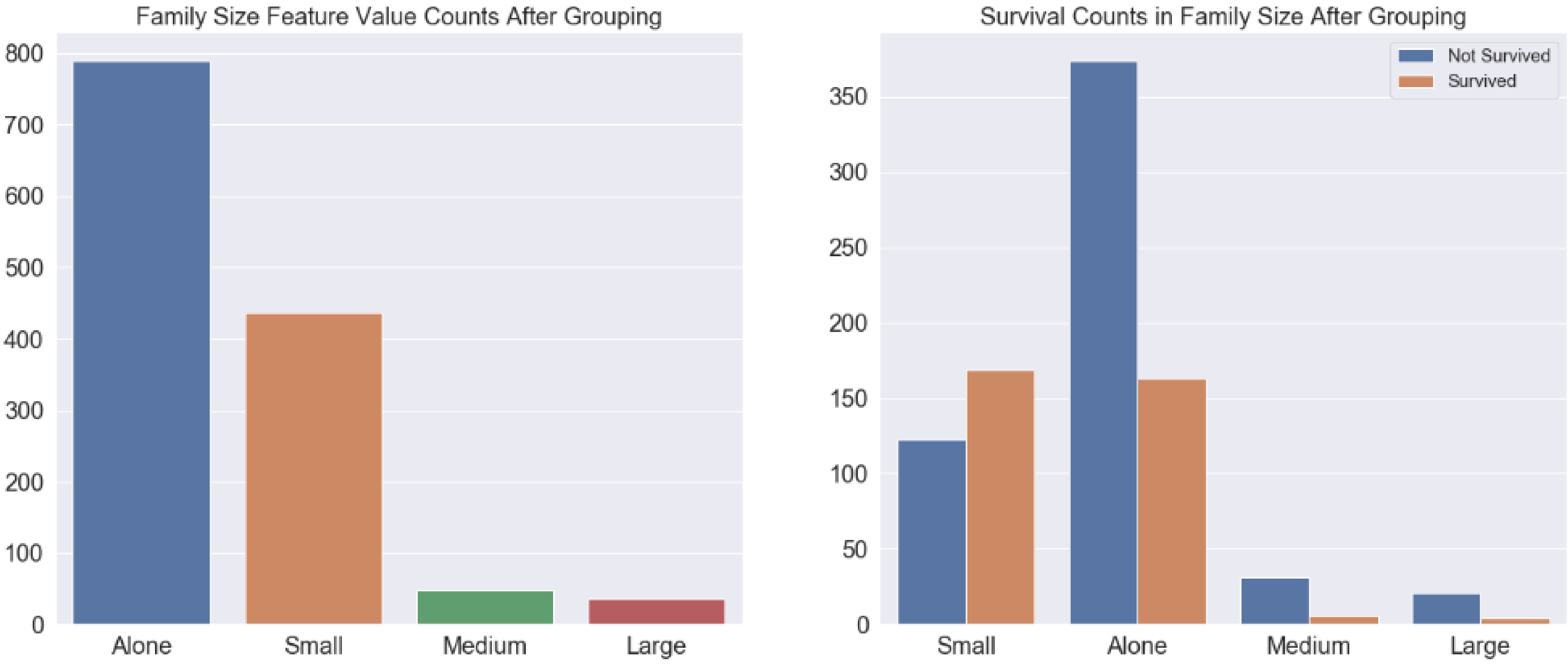
```
df_all['Family_Size'] = df_all['SibSp'] + df_all['Parch'] + 1
fig, axs = mp.subplots(nrows=2, ncols=2, figsize=(20, 20))
mp.subplots_adjust(right=1)

sns.barplot(x=df_all['Family_Size'].value_counts().index, y=df_all['Family_Size'].value_counts().values, ax=axs[0][0])
sns.countplot(x='Family_Size', hue='Survived', data=df_all, ax=axs[0][1])
axs[0][0].set_title('Family Size Feature Value Counts', size=20)
axs[0][1].set_title('Survival Counts in Family Size ', size=20)
family_map = {1:'Alone', 2:'Small', 3:'Small', 4:'Small', 5:'Medium', 6:'Medium', 7:'Large', 8:'Large', 11:'Large'}
df_all['Family_Size_Group'] = df_all['Family_Size'].map(family_map)

sns.barplot(x=df_all['Family_Size_Group'].value_counts().index, y=df_all['Family_Size_Group'].value_counts().values, ax=axs[1][0])
sns.countplot(x='Family_Size_Group', hue='Survived', data=df_all, ax=axs[1][1])
axs[1][0].set_title('Family Size Feature Value Counts After Grouping', size=20)
axs[1][1].set_title('Survival Counts in Family Size After Grouping', size=20)
for i in range(2):
    axs[i][1].legend(['Not Survived', 'Survived'], loc='upper right', prop={'size':15})
    for j in range(2):
        axs[i][j].tick_params(labelsize=20)
        axs[i][j].set_xlabel('')
        axs[i][j].set_ylabel('')
mp.show()
```



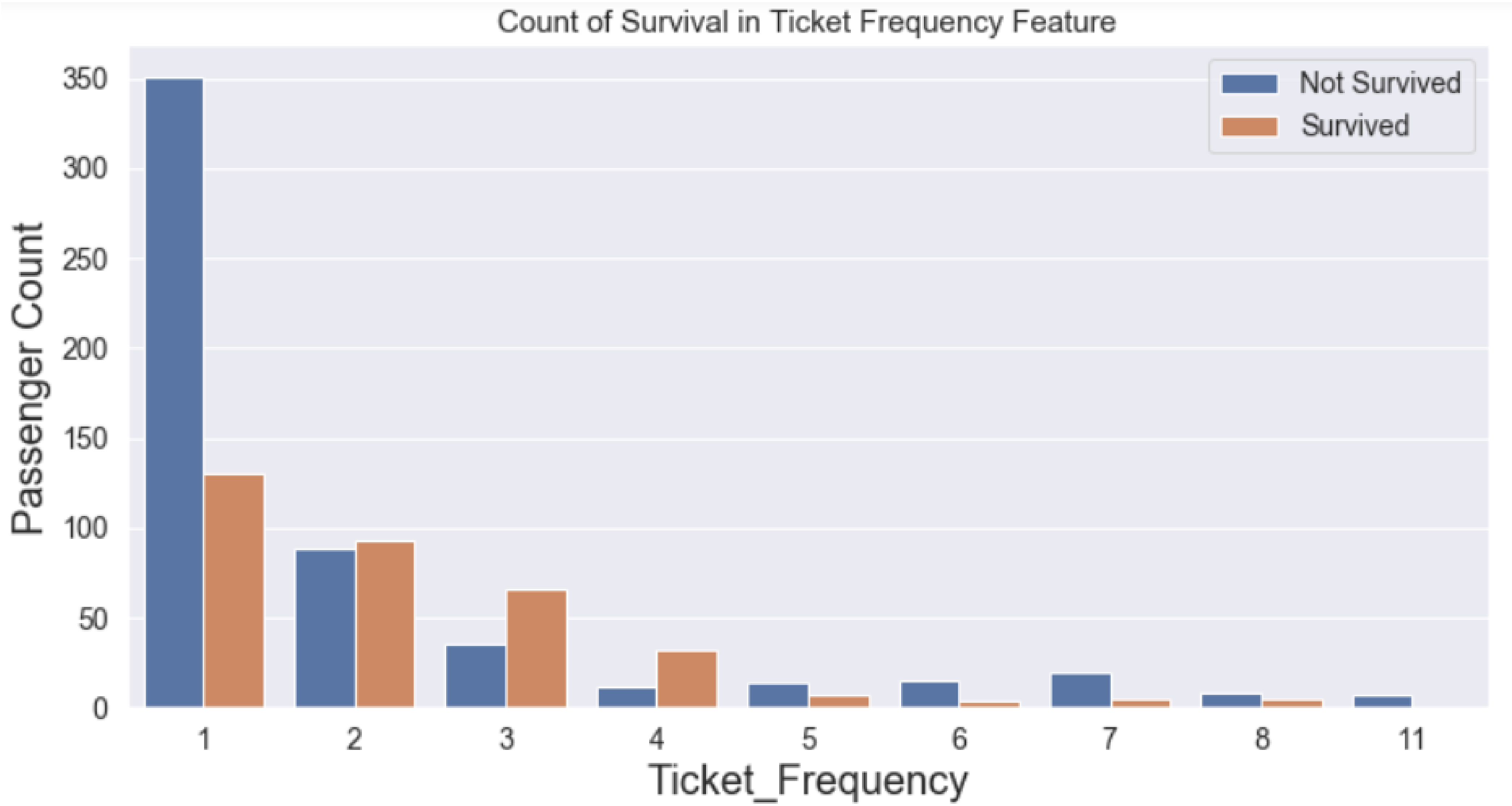
Family Size Feature Value Counts



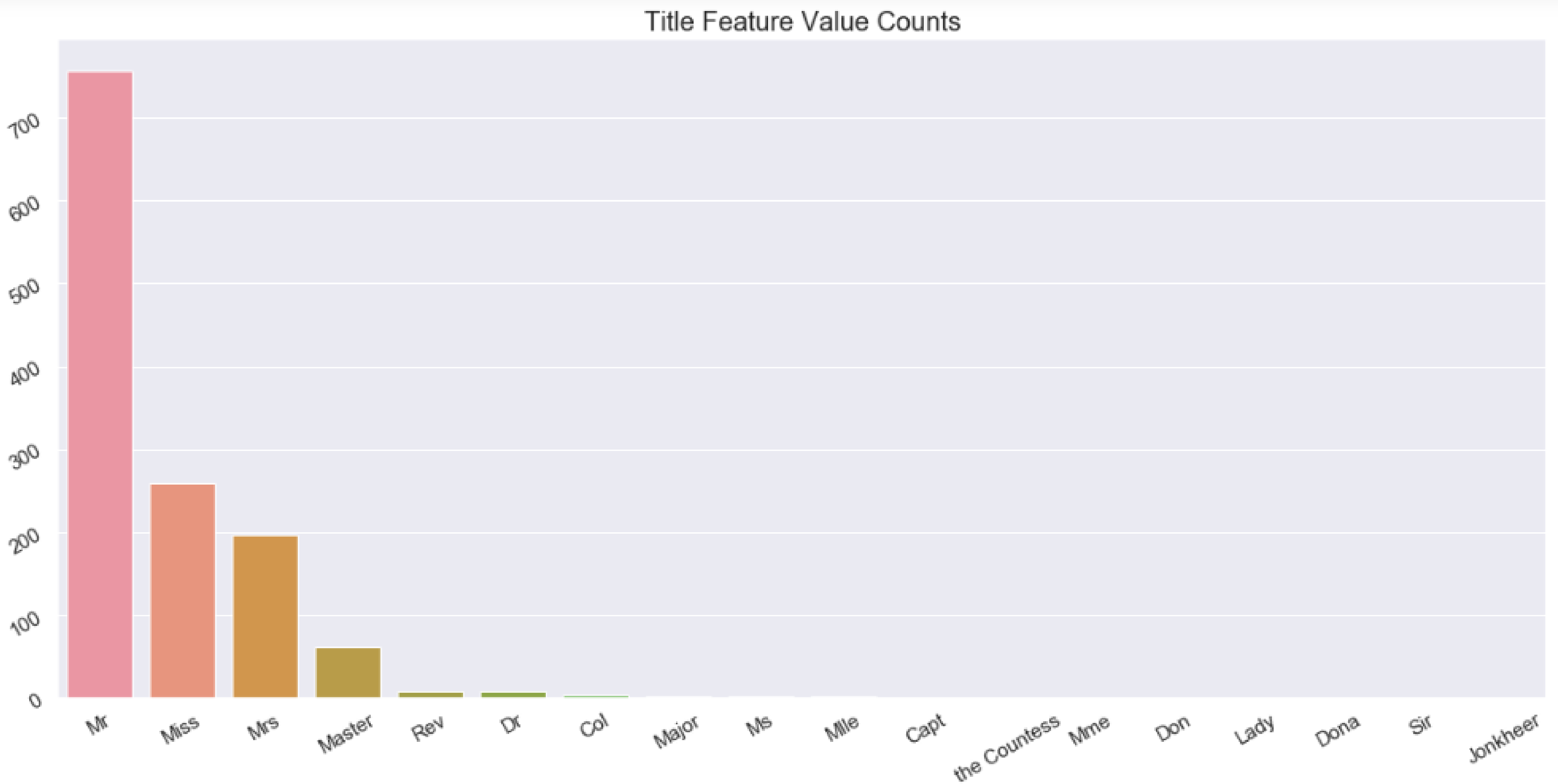
Survival Counts in Family Size

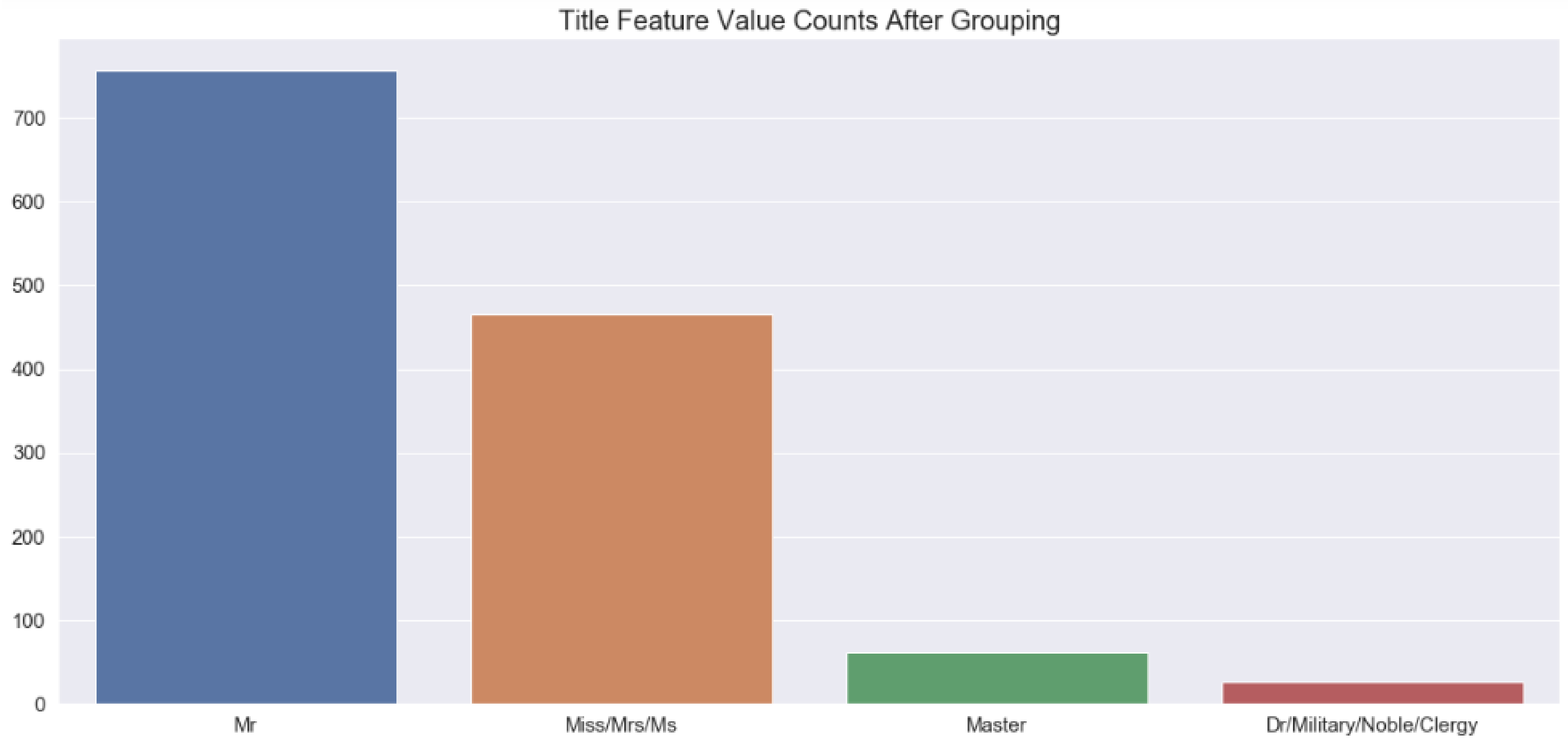

Family Size Feature Value Counts After Grouping
800
700
600
500
400
300
200
100
0
Alone
Small
Medium
Large
Survival Counts in Family Size After Grouping
Not Survived
Survived
350
300
250
200
150
100
50
0
Small
Alone
Medium
Large

```
df_all['Ticket_Frequency'] = df_all.groupby('Ticket')['Ticket'].transform('count')
fig, axs = mp.subplots(figsize=(12, 6))
sns.countplot(x='Ticket_Frequency', hue='Survived', data=df_all)

mp.xlabel('Ticket_Frequency', fontsize=20)
mp.ylabel('Passenger Count', fontsize=20)
mp.tick_params(labelsize=14)
mp.legend(['Not Survived', 'Survived'], loc='upper right', prop={'size':14})
mp.title('Count of Survival in {} Feature'.format('Ticket Frequency'), fontsize=15)
mp.show()
```

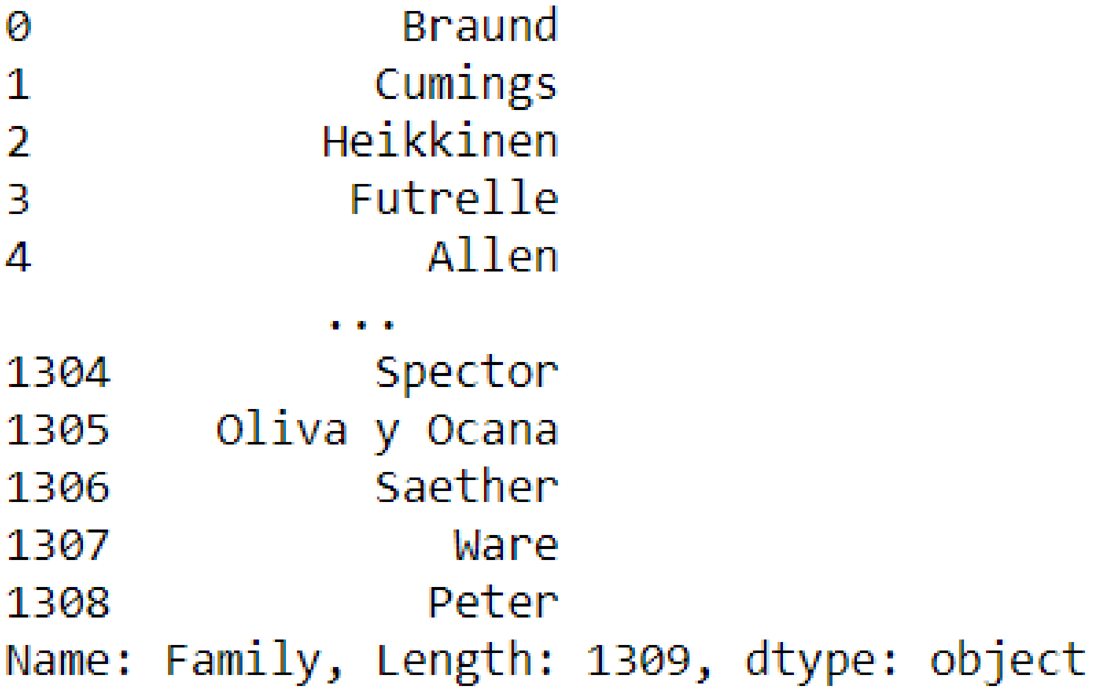

Count of Survival in Ticket Frequency Feature
Not Survived
Survived
Passenger Count
350
300
250
200
150
100
50
0
1
2
3
4
5
6
7
8
11
Ticket_Frequency

```
df_all['Title'] = df_all['Name'].str.split(', ', expand=True)[1].str.split('.', expand=True)[0]
df_all['Is_Merried'] = 0
df_all['Is_Merried'].loc[df_all['Title'] == 'Mrs'] == 1

fig, axs = mp.subplots(nrows=2, figsize=(20, 20))
sns.barplot(x=df_all['Title'].value_counts().index, y=df_all['Title'].value_counts().values, ax=axs[0])

axs[0].tick_params(labelrotation=30, labelsize=15)
axs[1].tick_params(labelsize=15)
axs[0].set_title('Title Feature Value Counts', size=20)
df_all['Title'] = df_all['Title'].replace(['Mlle', 'Mrs', 'Miss', 'Mme', 'Ms', 'Lady',
                          'the Countess', 'Dona'], 'Miss/Mrs/Ms')
df_all['Title'] = df_all['Title'].replace(['Don', 'Rev', 'Dr', 'Major', 'Sir', 'Col',
                          'Capt', 'Jonkheer'], 'Dr/Military/Noble/Clergy')
sns.barplot(x=df_all['Title'].value_counts().index, y=df_all['Title'].value_counts().values, ax=axs[1])
axs[1].set_title('Title Feature Value Counts After Grouping', size=20)
mp.show()
```


Title Feature Value Counts
0
100
200
300
400
500
600
700
Mr
Miss
Mrs
Master
Rev
Dr
Col
Major
Ms
Mlle
Capt
the Countess
Mme
Don
Lady
Dona
Sir
Jonkheer

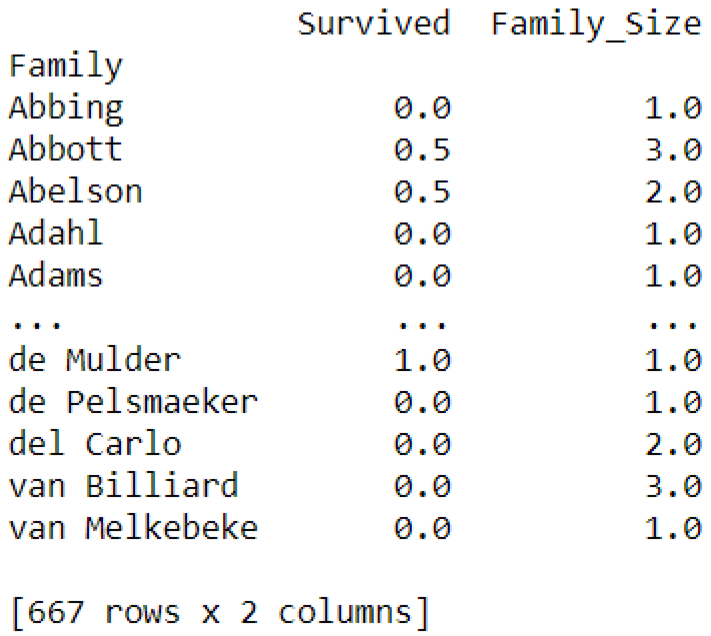

Title Feature Value Counts After Grouping
700
600
500
400
300
200
100
0
Mr
Miss/Mrs/Ms
Master
Dr/Military/Noble/Clergy

```
def create_families(data):
    families = []
    for i in range(len(data)):
        name = data.iloc[i]
        if '(' in name:
            name_no_backet = name.split('(')[0]
        else:
            name_no_backet = name
        family = name.split(',')[0]
        for c in string.punctuation:
            family = family.replace(c,'').strip()

        families.append(family)
    return families
df_all['Family'] = create_families(df_all['Name'])
df_all['Family']
```

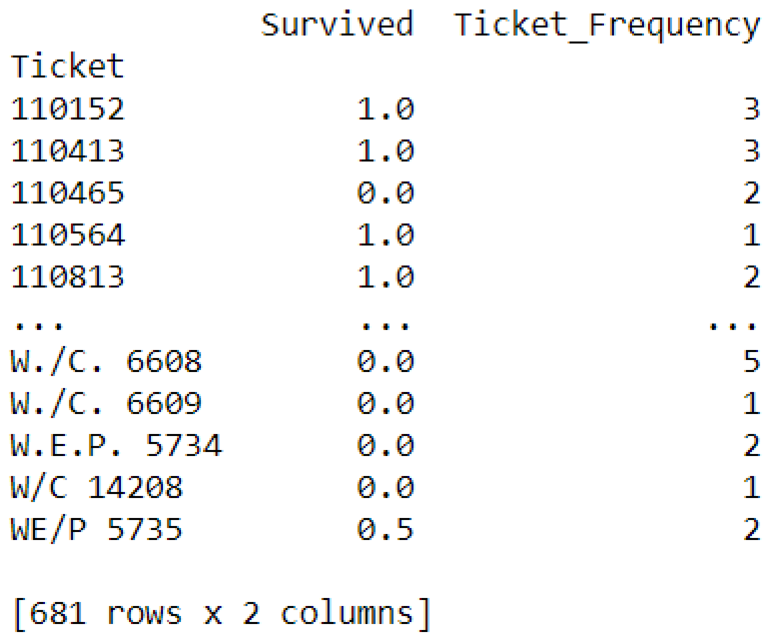


```
0                Braund
1               Cumings
2             Heikkinen
3              Futrelle
4                 Allen
                ...
1304            Spector
1305      Oliva y Ocana
1306            Saether
1307               Ware
1308              Peter
Name: Family, Length: 1309, dtype: object
```

```
df_train = df_all.loc[:890]
df_test = df_all.loc[891:]
dfs = [df_train, df_test]
non_unique_families = [x for x in df_train['Family'].unique() if x in df_test['Family'].unique()]
non_unique_tickets = [x for x in df_train['Ticket'].unique() if x in df_test['Ticket'].unique()]

df_family_survival_rate = df_train.groupby('Family')['Survived', 'Family', 'Family_Size'].median()
df_ticket_survival_rate = df_train.groupby('Ticket')['Survived', 'Ticket', 'Ticket_Frequency'].median()
print(df_family_survival_rate)
print(df_ticket_survival_rate)
```



```
              Survived  Family_Size
Family
Abbing             0.0          1.0
Abbott             0.5          3.0
Abelson            0.5          2.0
Adahl              0.0          1.0
Adams              0.0          1.0
...                ...          ...
de Mulder          1.0          1.0
de Pelsmaeker      0.0          1.0
del Carlo          0.0          2.0
van Billiard       0.0          3.0
van Melkebeke      0.0          1.0

[667 rows x 2 columns]
```

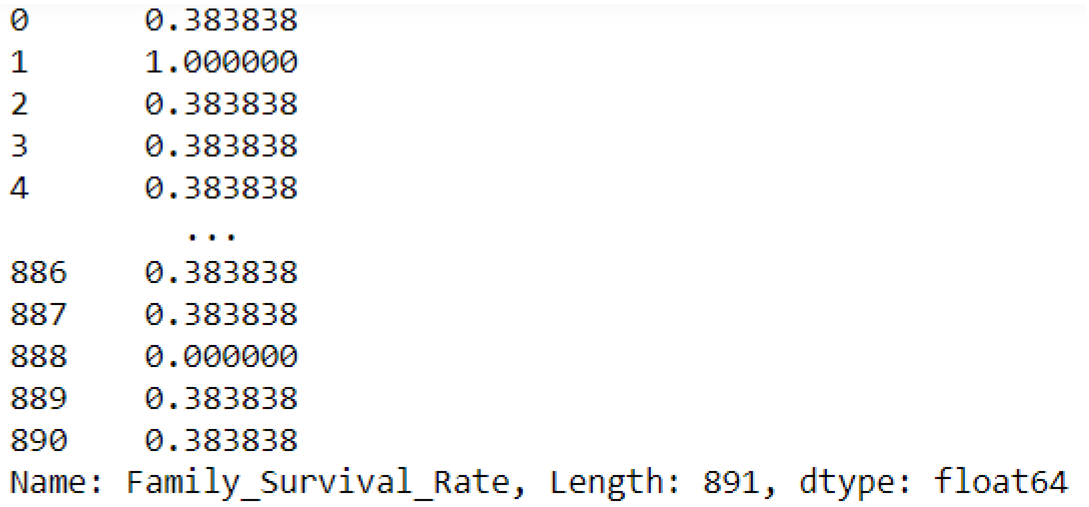


```
             Survived  Ticket_Frequency
Ticket
110152            1.0                 3
110413            1.0                 3
110465            0.0                 2
110564            1.0                 1
110813            1.0                 2
...               ...               ...
W./C. 6608        0.0                 5
W./C. 6609        0.0                 1
W.E.P. 5734       0.0                 2
W/C 14208         0.0                 1
WE/P 5735         0.5                 2

[681 rows x 2 columns]
```

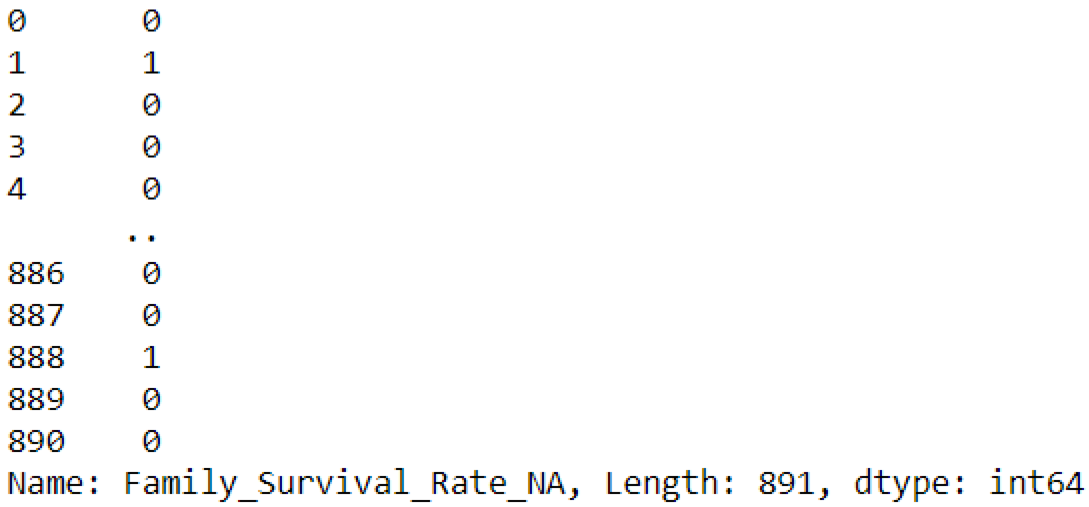
```
family_rate = {}
ticket_rate = {}
for i in range(len(df_family_survival_rate)):
    if df_family_survival_rate.index[i] in non_unique_families and df_family_survival_rate.iloc[i, 1] > 1:
        family_rate[df_family_survival_rate.index[i]] = df_family_survival_rate.iloc[i, 0]
for i in range(len(df_ticket_survival_rate)):
    if df_ticket_survival_rate.index[i] in non_unique_tickets and df_ticket_survival_rate.iloc[i, 1] > 1:
        ticket_rate[df_ticket_survival_rate.index[i]] = df_ticket_survival_rate.iloc[i, 0]
```

```
mean_survival_rate = df_train['Survived'].mean()
print(mean_survival_rate)      0.3838383838383838
train_family_survival_rate = []
train_family_survival_rate_NA = []
test_family_survival_rate = []
test_family_survival_rate_NA = []
for i in range(len(df_train)):
    if df_train['Family'][i] in family_rate:
        train_family_survival_rate.append(family_rate[df_train['Family'][i]])
        train_family_survival_rate_NA.append(1)
    else:
        train_family_survival_rate.append(mean_survival_rate)
        train_family_survival_rate_NA.append(0)
for i in range(len(df_test)):
    if df_test['Family'].iloc[i] in family_rate:
        test_family_survival_rate.append(family_rate[df_test['Family'].iloc[i]])
        test_family_survival_rate_NA.append(1)
    else:
        test_family_survival_rate.append(mean_survival_rate)
        test_family_survival_rate_NA.append(0)
```

```
df_train['Family_Survival_Rate'] = train_family_survival_rate
df_train['Family_Survival_Rate_NA'] = train_family_survival_rate_NA
df_test['Family_Survival_Rate'] = test_family_survival_rate
df_test['Family_Survival_Rate_NA'] = test_family_survival_rate_NA
print(df_train['Family_Survival_Rate'])
print(df_train['Family_Survival_Rate_NA'])
```



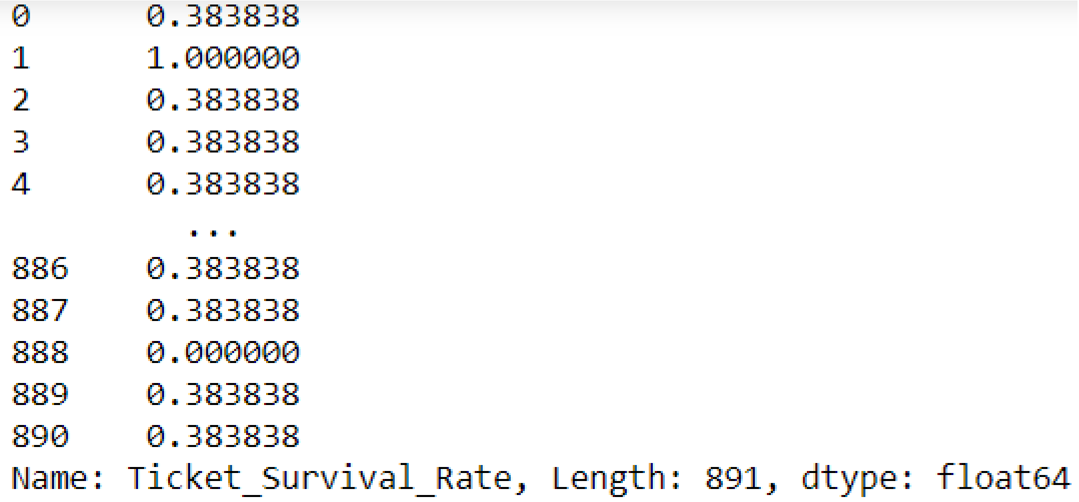
```
0      0.383838
1      1.000000
2      0.383838
3      0.383838
4      0.383838
         ...
886    0.383838
887    0.383838
888    0.000000
889    0.383838
890    0.383838
Name: Family_Survival_Rate, Length: 891, dtype: float64
```



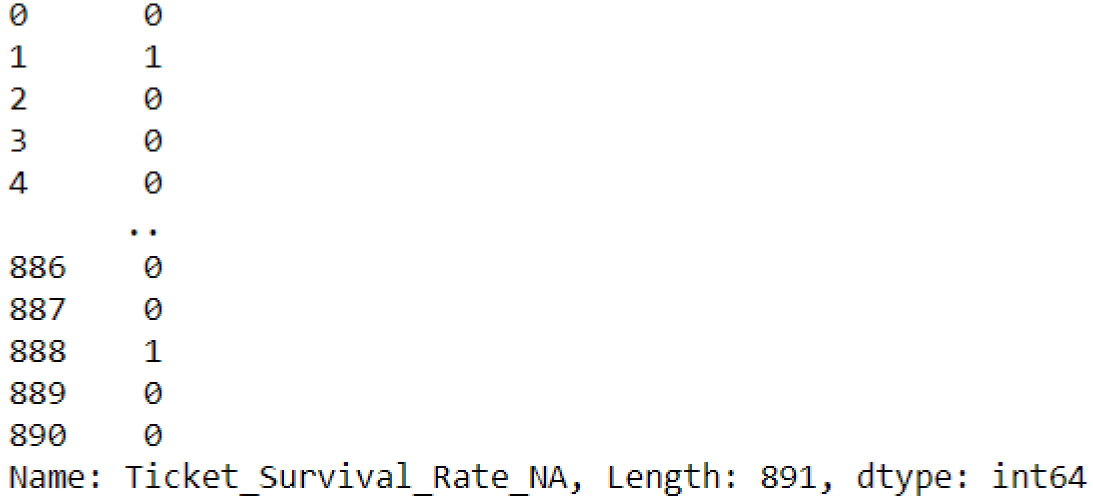
```
0      0
1      1
2      0
3      0
4      0
      ..
886    0
887    0
888    1
889    0
890    0
Name: Family_Survival_Rate_NA, Length: 891, dtype: int64
```

```
train_ticket_survival_rate = []
train_ticket_survival_rate_NA = []
test_ticket_survival_rate = []
test_ticket_survival_rate_NA = []

for i in range(len(df_train)):
    if df_train['Ticket'][i] in ticket_rate:
        train_ticket_survival_rate.append(ticket_rate[df_train['Ticket'][i]])
        train_ticket_survival_rate_NA.append(1)
    else:
        train_ticket_survival_rate.append(mean_survival_rate)
        train_ticket_survival_rate_NA.append(0)

for i in range(len(df_test)):
    if df_test['Ticket'].iloc[i] in ticket_rate:
        test_ticket_survival_rate.append(ticket_rate[df_test['Ticket'].iloc[i]])
        test_ticket_survival_rate_NA.append(1)
    else:
        test_ticket_survival_rate.append(mean_survival_rate)
        test_ticket_survival_rate_NA.append(0)
```

```
df_train['Ticket_Survival_Rate'] = train_ticket_survival_rate
df_train['Ticket_Survival_Rate_NA'] = train_ticket_survival_rate_NA
df_test['Ticket_Survival_Rate'] = test_ticket_survival_rate
df_test['Ticket_Survival_Rate_NA'] = test_ticket_survival_rate_NA
print(df_train['Ticket_Survival_Rate'])
print(df_train['Ticket_Survival_Rate_NA'])
```



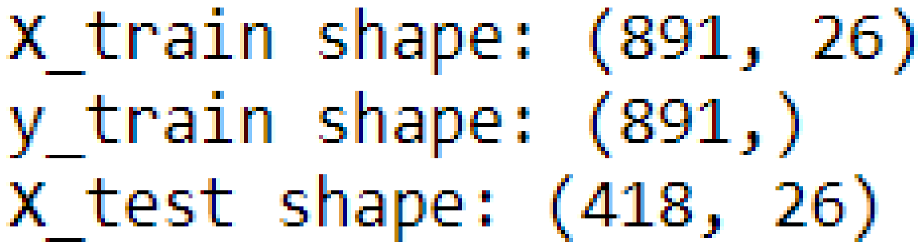
```
0      0.383838
1      1.000000
2      0.383838
3      0.383838
4      0.383838
         ...
886    0.383838
887    0.383838
888    0.000000
889    0.383838
890    0.383838
Name: Ticket_Survival_Rate, Length: 891, dtype: float64
```



```
0      0
1      1
2      0
3      0
4      0
      ..
886    0
887    0
888    1
889    0
890    0
Name: Ticket_Survival_Rate_NA, Length: 891, dtype: int64
```

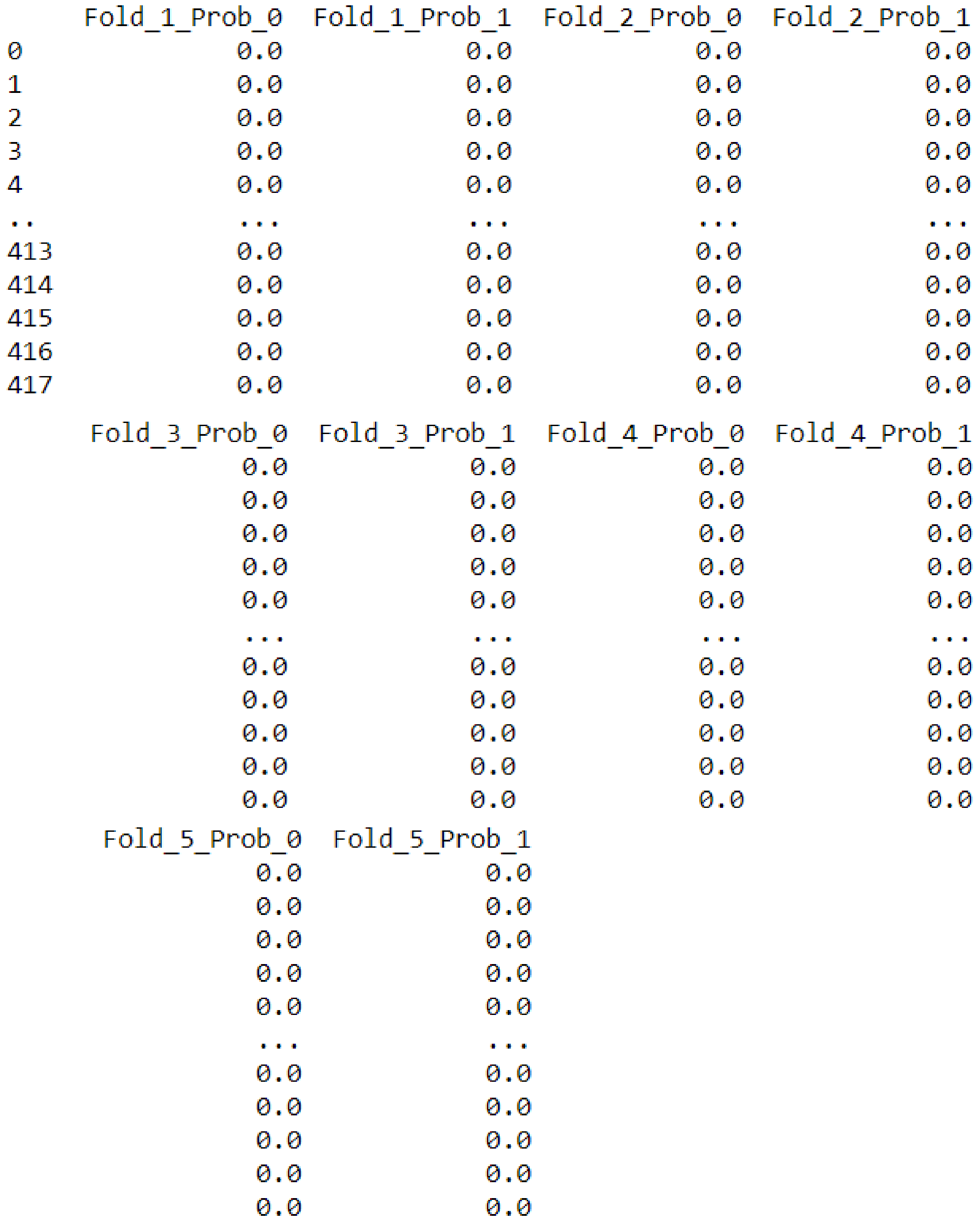
```
for df in [df_train, df_test]:
    df['Survival_Rate'] = (df['Family_Survival_Rate'] + df['Ticket_Survival_Rate']) / 2
    df['Survival_Rate_NA'] = (df['Family_Survival_Rate_NA'] + df['Ticket_Survival_Rate_NA']) / 2
non_numeric_features = ['Embarked', 'Sex', 'Deck', 'Title', 'Family_Size_Group', 'Age', 'Fare']
for df in dfs:
    for feature in non_numeric_features:
        df[feature] = sp.LabelEncoder().fit_transform(df[feature])
cat_features = ['Pclass', 'Sex', 'Deck', 'Embarked', 'Title', 'Family_Size_Group']
encoded_features = []

for df in dfs:
    for feature in cat_features:
        encoded_feat = sp.OneHotEncoder().fit_transform(df[feature].values.reshape(-1, 1)).toarray()
        n = df[feature].nunique()
        cols = ['{}_{}'.format(feature, n) for n in range(1, n + 1)]
        encoded_df = pd.DataFrame(encoded_feat, columns=cols)
        encoded_df.index = df.index
        encoded_features.append(encoded_df)
df_train = pd.concat([df_train, *encoded_features[:6]], axis=1)
df_test = pd.concat([df_test, *encoded_features[6:]], axis=1)
```

```
df_all = concat_df(df_train, df_test)
drop_features = ['Deck', 'Embarked', 'Family', 'Family_Size', 'Family_Size_Group', 'Survived',
 'Name', 'Parch', 'PassengerId', 'Pclass', 'Sex', 'SibSp', 'Ticket', 'Title','Ticket_Survival_Rate',
'Family_Survival_Rate', 'Ticket_Survival_Rate_NA','Family_Survival_Rate_NA']
df_all.drop(columns=drop_features, inplace=True)
X_train = sp.StandardScaler().fit_transform(df_train.drop(columns=drop_features))
y_train = df_train['Survived'].values
X_test = sp.StandardScaler().fit_transform(df_test.drop(columns=drop_features))

print('X_train shape: {}'.format(X_train.shape))
print('y_train shape: {}'.format(y_train.shape))
print('X_test shape: {}'.format(X_test.shape))
```

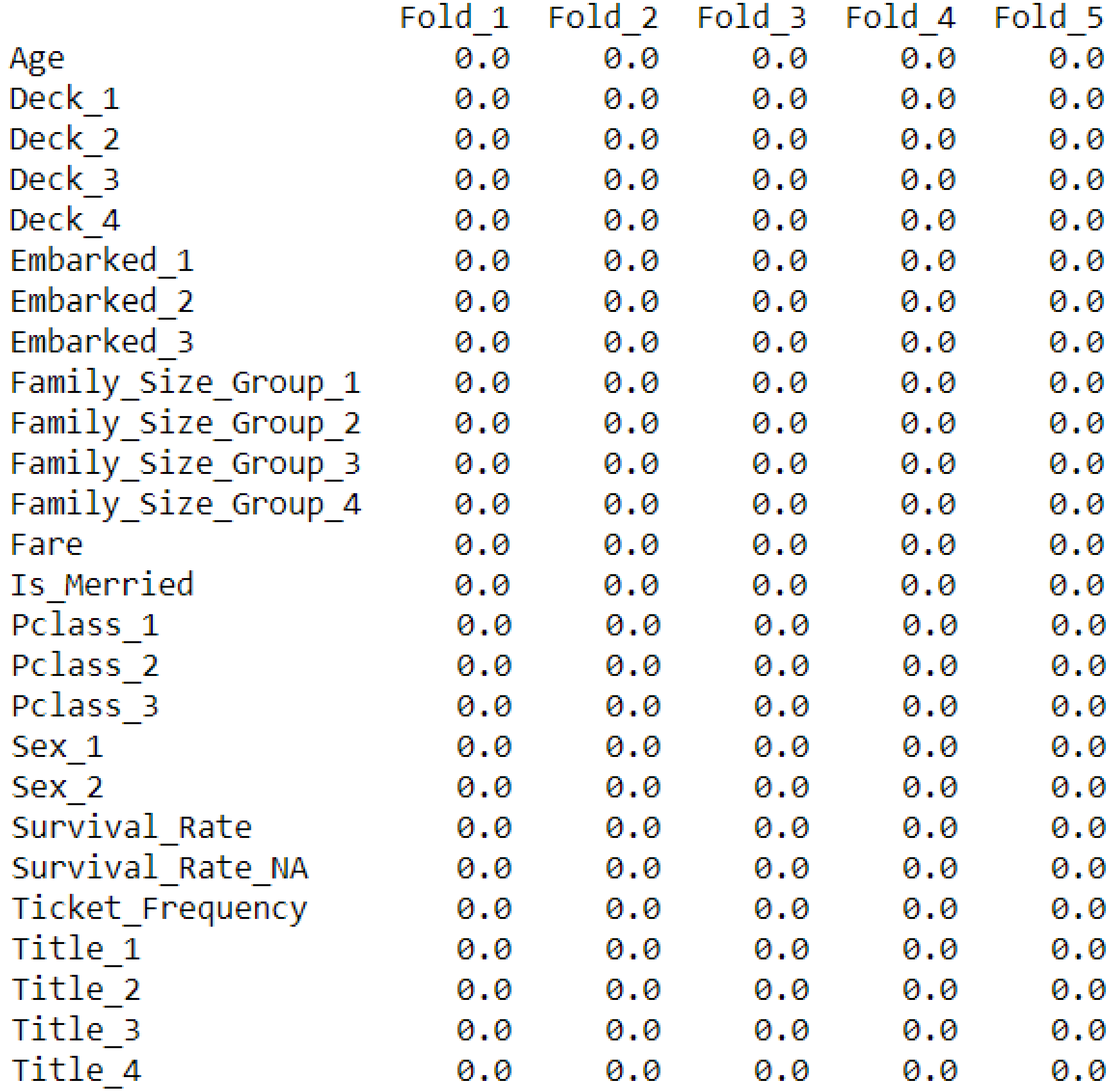


```
X_train shape: (891, 26)
y_train shape: (891,)
X_test shape: (418, 26)
```

```
single_best_model = se.RandomForestClassifier(n_estimators=1100, max_depth=5,
min_samples_split=4, min_samples_leaf=5,oob_score=True, random_state=SEED,
n_jobs=-1, verbose=1)


leaderboard_model = se.RandomForestClassifier(n_estimators=1750, ax_depth=7,
min_samples_split=6, min_samples_leaf=6,oob_score=True, random_state=SEED,
n_jobs=-1, verbose=1)


N = 5

probs = pd.DataFrame(np.zeros((len(X_test), N * 2)),
columns=['Fold_{}_Prob_{}'.format(i, j) for i in range(1, N + 1) for j in range(2)])

importances = pd.DataFrame(np.zeros((X_train.shape[1], N)),
columns=['Fold_{}'.format(i) for i in range(1, N + 1)], index=df_all.columns)

skf = ms.StratifiedKFold(n_splits=N, random_state=N, shuffle=True)

print(probs)

print(importances)
```



```
     Fold_1_Prob_0  Fold_1_Prob_1  Fold_2_Prob_0  Fold_2_Prob_1
0              0.0            0.0            0.0            0.0
1              0.0            0.0            0.0            0.0
2              0.0            0.0            0.0            0.0
3              0.0            0.0            0.0            0.0
4              0.0            0.0            0.0            0.0
..             ...            ...            ...            ...
413            0.0            0.0            0.0            0.0
414            0.0            0.0            0.0            0.0
415            0.0            0.0            0.0            0.0
416            0.0            0.0            0.0            0.0
417            0.0            0.0            0.0            0.0
      Fold_3_Prob_0  Fold_3_Prob_1  Fold_4_Prob_0  Fold_4_Prob_1
               0.0            0.0            0.0            0.0
               0.0            0.0            0.0            0.0
               0.0            0.0            0.0            0.0
               0.0            0.0            0.0            0.0
               0.0            0.0            0.0            0.0
               ...            ...            ...            ...
               0.0            0.0            0.0            0.0
               0.0            0.0            0.0            0.0
               0.0            0.0            0.0            0.0
               0.0            0.0            0.0            0.0
               0.0            0.0            0.0            0.0
       Fold_5_Prob_0  Fold_5_Prob_1
                 0.0            0.0
                 0.0            0.0
                 0.0            0.0
                 0.0            0.0
                 0.0            0.0
                 ...            ...
                 0.0            0.0
                 0.0            0.0
                 0.0            0.0
                 0.0            0.0
                 0.0            0.0
```

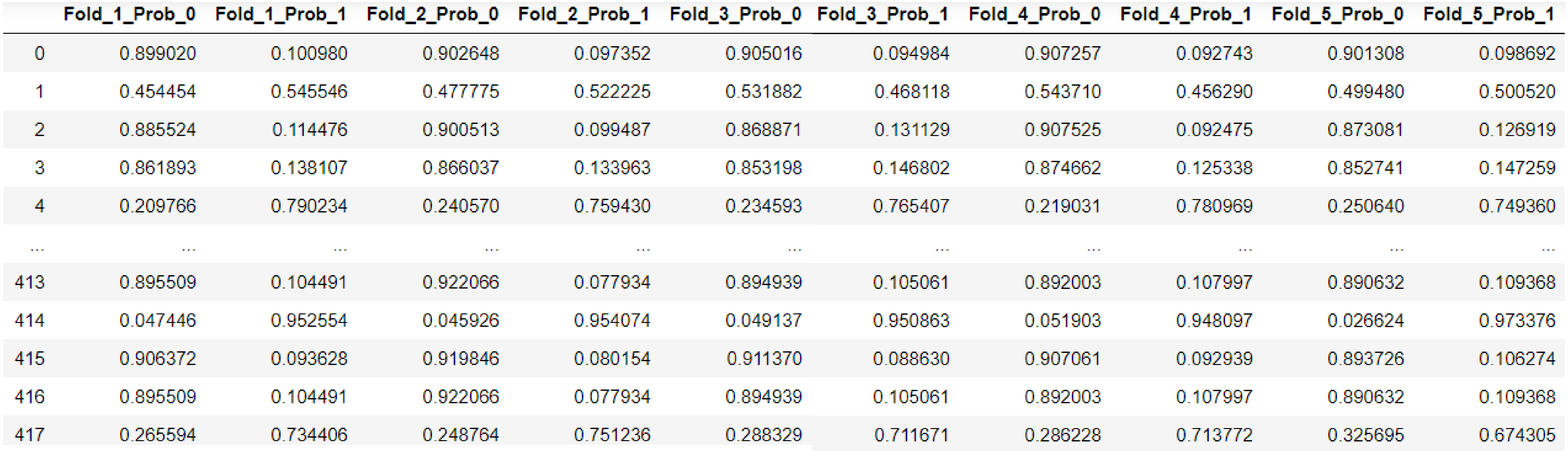


```
                      Fold_1  Fold_2  Fold_3  Fold_4  Fold_5
Age                      0.0     0.0     0.0     0.0     0.0
Deck_1                   0.0     0.0     0.0     0.0     0.0
Deck_2                   0.0     0.0     0.0     0.0     0.0
Deck_3                   0.0     0.0     0.0     0.0     0.0
Deck_4                   0.0     0.0     0.0     0.0     0.0
Embarked_1               0.0     0.0     0.0     0.0     0.0
Embarked_2               0.0     0.0     0.0     0.0     0.0
Embarked_3               0.0     0.0     0.0     0.0     0.0
Family_Size_Group_1      0.0     0.0     0.0     0.0     0.0
Family_Size_Group_2      0.0     0.0     0.0     0.0     0.0
Family_Size_Group_3      0.0     0.0     0.0     0.0     0.0
Family_Size_Group_4      0.0     0.0     0.0     0.0     0.0
Fare                     0.0     0.0     0.0     0.0     0.0
Is_Merried               0.0     0.0     0.0     0.0     0.0
Pclass_1                 0.0     0.0     0.0     0.0     0.0
Pclass_2                 0.0     0.0     0.0     0.0     0.0
Pclass_3                 0.0     0.0     0.0     0.0     0.0
Sex_1                    0.0     0.0     0.0     0.0     0.0
Sex_2                    0.0     0.0     0.0     0.0     0.0
Survival_Rate            0.0     0.0     0.0     0.0     0.0
Survival_Rate_NA         0.0     0.0     0.0     0.0     0.0
Ticket_Frequency         0.0     0.0     0.0     0.0     0.0
Title_1                  0.0     0.0     0.0     0.0     0.0
Title_2                  0.0     0.0     0.0     0.0     0.0
Title_3                  0.0     0.0     0.0     0.0     0.0
Title_4                  0.0     0.0     0.0     0.0     0.0
```

```
oob = 0
fprs, tprs, scores = [], [], []
for fold, (trn_idx, val_idx) in enumerate(skf.split(X_train, y_train), 1):
    print('Fold {}\n'.format(fold))
    leaderboard_model.fit(X_train[trn_idx], y_train[trn_idx])
    trn_fpr, trn_tpr, trn_thresholds = sm.roc_curve(y_train[trn_idx], leaderboard_model.predict_proba(X_train[trn_idx])[:, 1])
    trn_auc_score = sm.auc(trn_fpr, trn_tpr)

    val_fpr, val_tpr, val_thresholds = sm.roc_curve(y_train[val_idx], leaderboard_model.predict_proba(X_train[val_idx])[:, 1])
    val_auc_score = sm.auc(val_fpr, val_tpr)

    scores.append((trn_auc_score, val_auc_score))
    fprs.append(val_fpr)
    tprs.append(val_tpr)
    probs.loc[:, 'Fold_{}_Prob_0'.format(fold)] = leaderboard_model.predict_proba(X_test)[:, 0]
    probs.loc[:, 'Fold_{}_Prob_1'.format(fold)] = leaderboard_model.predict_proba(X_test)[:, 1]
    importances.iloc[:, fold - 1] = leaderboard_model.feature_importances_

    oob += leaderboard_model.oob_score_ / N
    print('Fold {} OOB Score: {}\n'.format(fold, leaderboard_model.oob_score_))
print('Average OOB Score: {}'.format(oob))
```

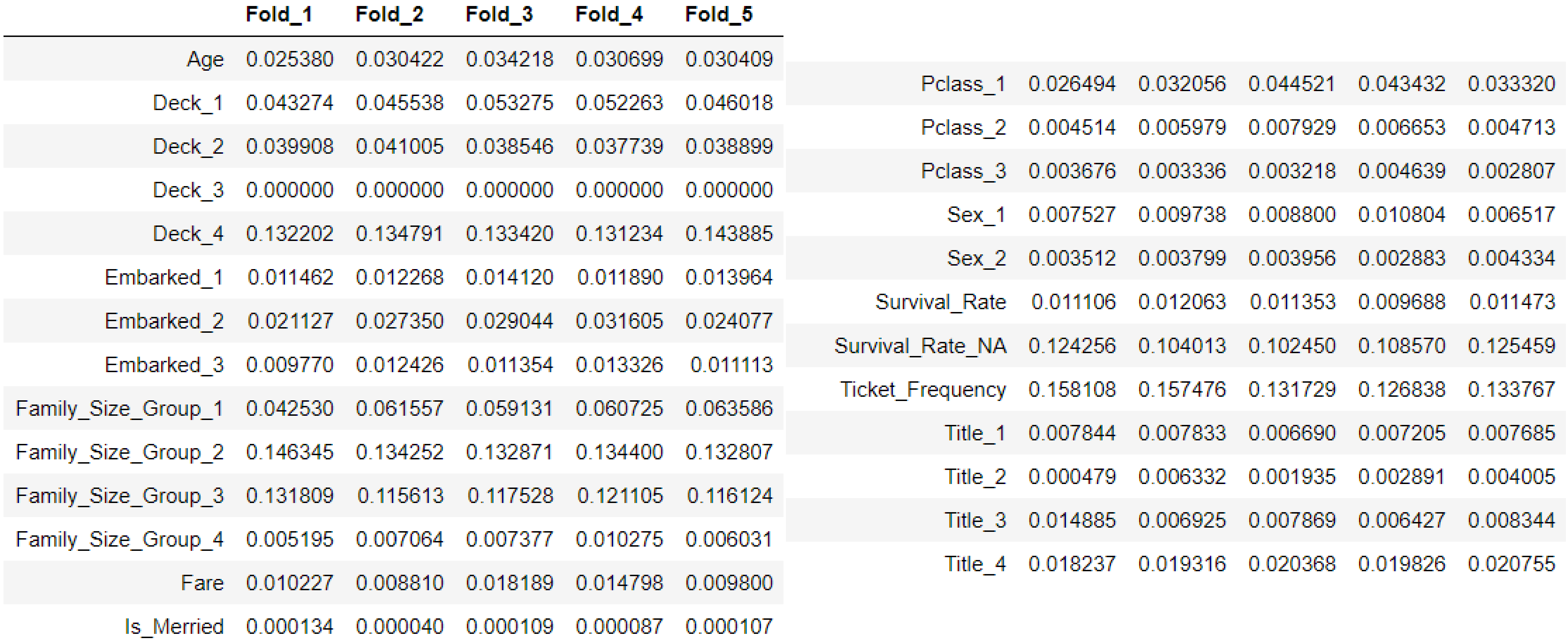
```
Fold 1 OOB Score: 0.8567415730337079
Fold 2 OOB Score: 0.846910112359550б
Fold 3 OOB Score: 0.8274894810659187
Fold 4 OOB Score: 0.828892005610098l
Fold 5 OOB Score: 0.8431372549019608
Average OOB Score: 0.8406340853942473
```

probs



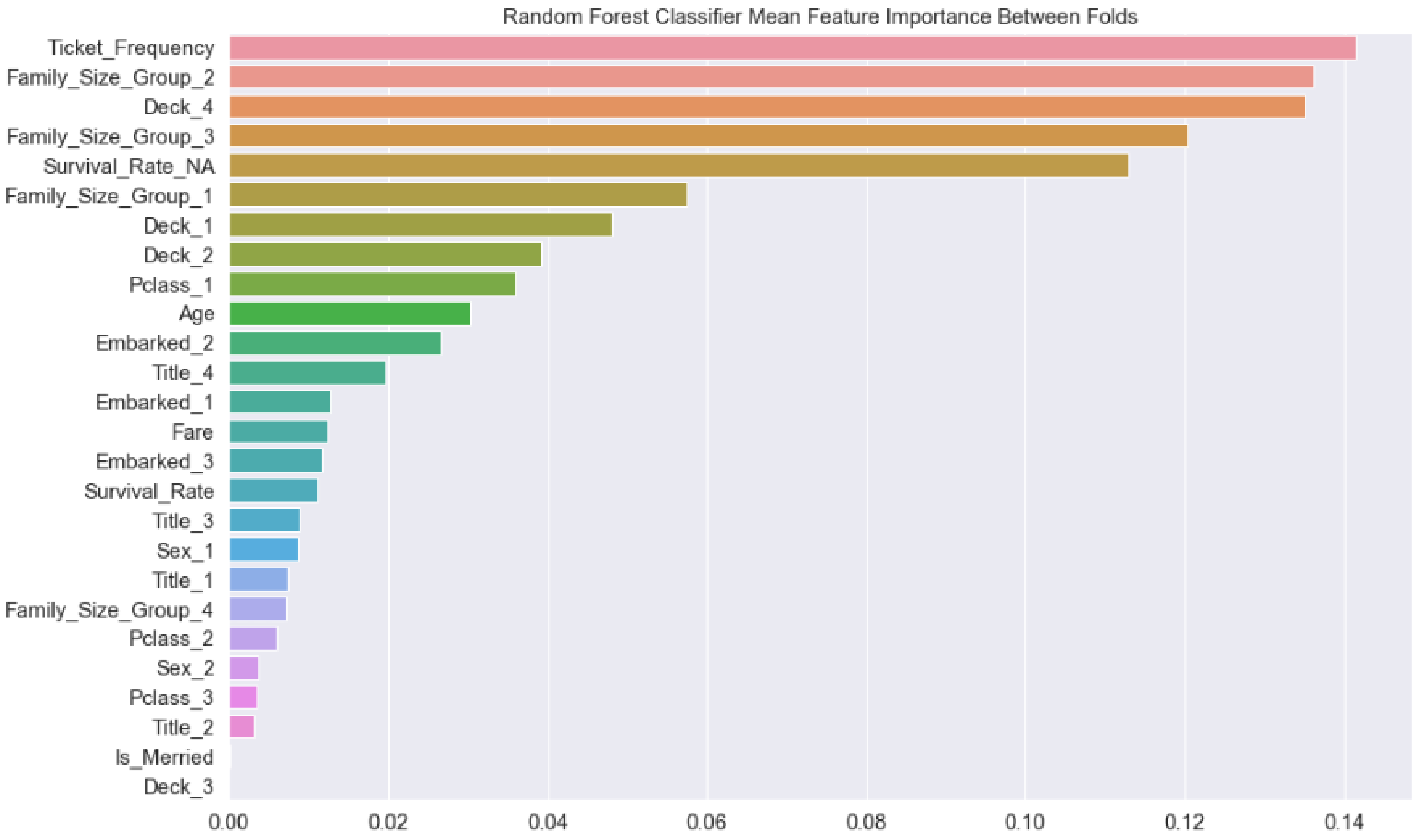
	Fold_1_Prob_0	Fold_1_Prob_1	Fold_2_Prob_0	Fold_2_Prob_1	Fold_3_Prob_0	Fold_3_Prob_1	Fold_4_Prob_0	Fold_4_Prob_1	Fold_5_Prob_0	Fold_5_Prob_1
0	0.899020	0.100980	0.902648	0.097352	0.905016	0.094984	0.907257	0.092743	0.901308	0.098692
1	0.454454	0.545546	0.477775	0.522225	0.531882	0.468118	0.543710	0.456290	0.499480	0.500520
2	0.885524	0.114476	0.900513	0.099487	0.868871	0.131129	0.907525	0.092475	0.873081	0.126919
3	0.861893	0.138107	0.866037	0.133963	0.853198	0.146802	0.874662	0.125338	0.852741	0.147259
4	0.209766	0.790234	0.240570	0.759430	0.234593	0.765407	0.219031	0.780969	0.250640	0.749360
...	...	...	...	...	...	...	...	...	...	...
413	0.895509	0.104491	0.922066	0.077934	0.894939	0.105061	0.892003	0.107997	0.890632	0.109368
414	0.047446	0.952554	0.045926	0.954074	0.049137	0.950863	0.051903	0.948097	0.026624	0.973376
415	0.906372	0.093628	0.919846	0.080154	0.911370	0.088630	0.907061	0.092939	0.893726	0.106274
416	0.895509	0.104491	0.922066	0.077934	0.894939	0.105061	0.892003	0.107997	0.890632	0.109368
417	0.265594	0.734406	0.248764	0.751236	0.288329	0.711671	0.286228	0.713772	0.325695	0.674305

importances

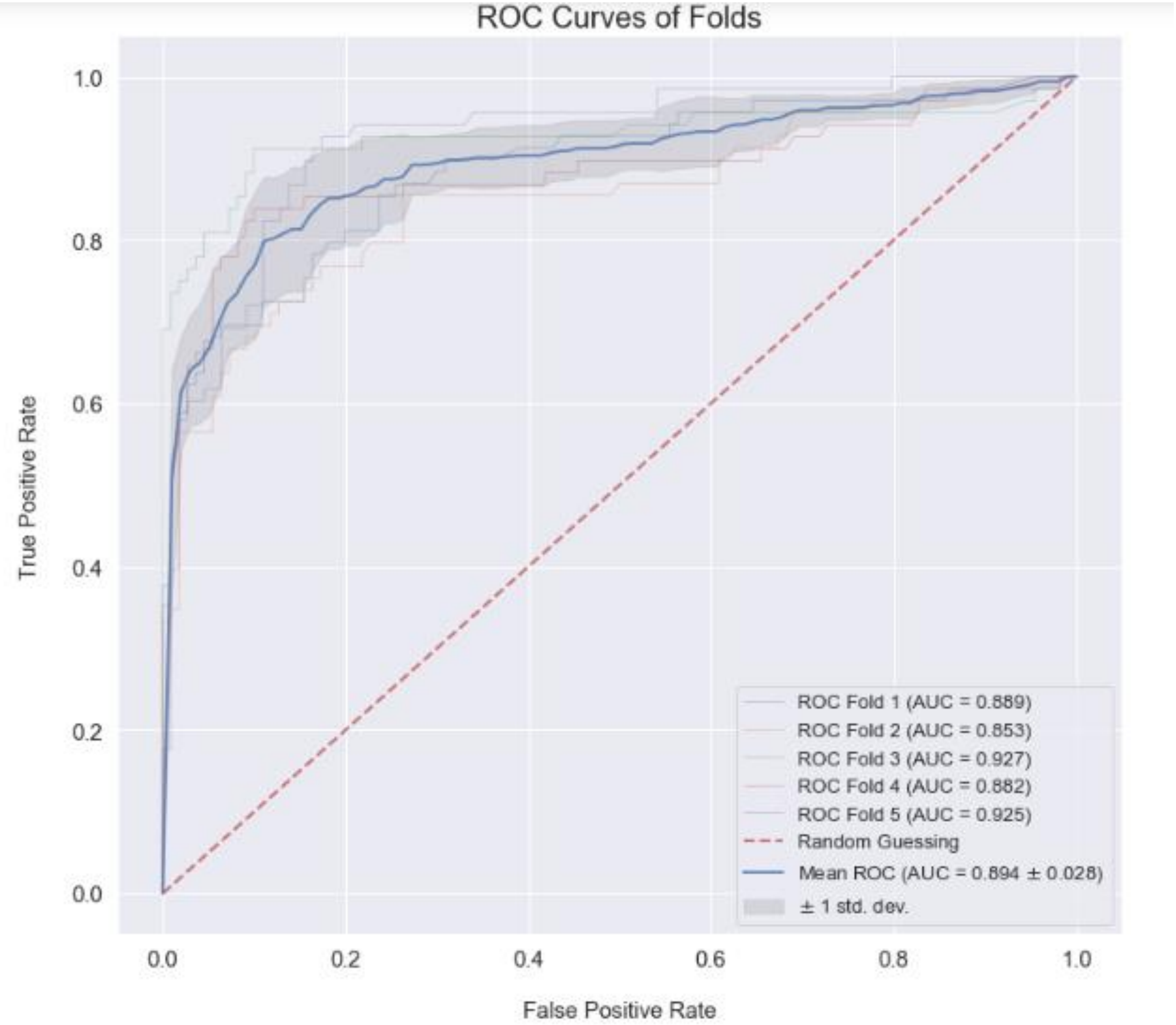


	Fold_1	Fold_2	Fold_3	Fold_4	Fold_5
Age	0.025380	0.030422	0.034218	0.030699	0.030409
Deck_1	0.043274	0.045538	0.053275	0.052263	0.046018
Deck_2	0.039908	0.041005	0.038546	0.037739	0.038899
Deck_3	0.000000	0.000000	0.000000	0.000000	0.000000
Deck_4	0.132202	0.134791	0.133420	0.131234	0.143885
Embarked_1	0.011462	0.012268	0.014120	0.011890	0.013964
Embarked_2	0.021127	0.027350	0.029044	0.031605	0.024077
Embarked_3	0.009770	0.012426	0.011354	0.013326	0.011113
Family_Size_Group_1	0.042530	0.061557	0.059131	0.060725	0.063586
Family_Size_Group_2	0.146345	0.134252	0.132871	0.134400	0.132807
Family_Size_Group_3	0.131809	0.115613	0.117528	0.121105	0.116124
Family_Size_Group_4	0.005195	0.007064	0.007377	0.010275	0.006031
Fare	0.010227	0.008810	0.018189	0.014798	0.009800
Is_Merried	0.000134	0.000040	0.000109	0.000087	0.000107
Pclass_1	0.026494	0.032056	0.044521	0.043432	0.033320
Pclass_2	0.004514	0.005979	0.007929	0.006653	0.004713
Pclass_3	0.003676	0.003336	0.003218	0.004639	0.002807
Sex_1	0.007527	0.009738	0.008800	0.010804	0.006517
Sex_2	0.003512	0.003799	0.003956	0.002883	0.004334
Survival_Rate	0.011106	0.012063	0.011353	0.009688	0.011473
Survival_Rate_NA	0.124256	0.104013	0.102450	0.108570	0.125459
Ticket_Frequency	0.158108	0.157476	0.131729	0.126838	0.133767
Title_1	0.007844	0.007833	0.006690	0.007205	0.007685
Title_2	0.000479	0.006332	0.001935	0.002891	0.004005
Title_3	0.014885	0.006925	0.007869	0.006427	0.008344
Title_4	0.018237	0.019316	0.020368	0.019826	0.020755

```
importances['Mean_Importance'] = importances.mean(axis=1)
importances.sort_values(by='Mean_Importance', inplace=True, ascending=False)

mp.figure(figsize=(15, 20))
sns.barplot(x=importances['Mean_Importance'], y=importances.index, data=importances)

mp.xlabel('')
mp.tick_params(labelsize=15)
mp.title('Random Forest Classifier Mean Feature Importance Between Folds', size=15)
mp.show()
```


Random Forest Classifier Mean Feature Importance Between Folds
Ticket_Frequency
Family_Size_Group_2
Deck_4
Family_Size_Group_3
Survival_Rate_NA
Family_Size_Group_1
Deck_1
Deck_2
Pclass_1
Age
Embarked_2
Title_4
Embarked_1
Fare
Embarked_3
Survival_Rate
Title_3
Sex_1
Title_1
Family_Size_Group_4
Pclass_2
Sex_2
Pclass_3
Title_2
Is_Merried
Deck_3
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14

```
def plot_roc_curve(fprs, tprs):
    tprs_interp = []
    mean_fpr = np.linspace(0, 1, 100)
    aucs = []
    fig, ax = mp.subplots(figsize=(15, 15))

    for i, (fpr, tpr) in enumerate(zip(fprs, tprs), 1):
        tprs_interp.append(np.interp(mean_fpr, fpr, tpr))
        tprs_interp[-1][0] = 0.0
        roc_auc = sm.auc(fpr, tpr)
        aucs.append(roc_auc)
        ax.plot(fpr, tpr, lw=1, alpha=0.3, label='ROC Fold {} (
        AUC = {:.3f})'.format(i, roc_auc))

    ax.plot([0, 1], [0, 1], lw=2, alpha=0.8, color='r', linestyle='--',
    label='Random Guessing')

    mean_tpr = np.mean(tprs_interp, axis=0)
    mean_tpr[-1] = 1.0
    mean_auc = sm.auc(mean_fpr, mean_tpr)
    std_auc = np.std(aucs)
    ax.plot(mean_fpr, mean_tpr, c=‘b’, lw=2, alpha=0.8, label=‘Mean ROC (
    AUC = {:.3f} $\pm$ {:.3f})'.format(mean_auc, std_auc))

    std_tpr = np.std(tprs_interp, axis=0)
    tprs_upper = np.minimum(mean_tpr + std_tpr, 1)
    tprs_lower = np.maximum(mean_tpr - std_tpr, 0)
    ax.fill_between(mean_fpr, tprs_upper, tprs_lower,
    color='gray', alpha=0.2, label='$\pm$ 1 std. dev.')

    ax.set_xlabel('False Positive Rate', size=15, labelpad=20)
    ax.set_ylabel('True Positive Rate', size=15, labelpad=20)
    ax.tick_params(labelsize=15)
    ax.set_xlim([-0.05, 1.05])
    ax.set_ylim([-0.05, 1.05])

    ax.set_title('ROC Curves of Folds', size=20)
    ax.legend(loc='lower right', prop={'size': 13})

    mp.show()
plot_roc_curve(fprs, tprs)
```


ROC Curves of Folds
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
ROC Fold 1 (AUC = 0.889)
ROC Fold 2 (AUC = 0.853)
ROC Fold 3 (AUC = 0.927)
ROC Fold 4 (AUC = 0.882)
ROC Fold 5 (AUC = 0.925)
Random Guessing
Mean ROC (AUC = 0.894 ± 0.028)
± 1 std. dev.

```
y_pred = probs['pred'].astype(int)
final = pd.DataFrame()
final['PassengerId'] = df_test['PassengerId']
final['Survived'] = y_pred.values
final.to_csv('final_submit.csv', header=True, index=False)
```