SA18225211 李志豪

I did the data analysis part in my Curriculum design.
Include 2 data set: train.csv and test.csv.
Question description

The sinking of the RMS Titanic is one of the most infamous shipwrecks in history. On April 15, 1912, during her maiden voyage, the Titanic sank after colliding with an iceberg, killing 1502 out of 2224 passengers and crew. This sensational tragedy shocked the international community and led to better safety regulations for ships.

One of the reasons that the shipwreck led to such loss of life was that there were not enough lifeboats for the passengers and crew. Although there was some element of luck involved in surviving the sinking, some groups of people were more likely to survive than others, such as women, children, and the upper-class.

In this challenge, we ask you to complete the analysis of what sorts of people were likely to survive. In particular, we ask you to apply the tools of machine learning to predict which passengers survived the tragedy.

1 Data screening

The following are the required libraries

```
import re
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

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

Observe the first few lines of data at the training level

```
train_data = pd.read_csv('data/train.csv')
test_data = pd.read_csv('data/test.csv')

sns.set_style('whitegrid')
train_data.head()
```

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.25		S
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 3101282	7.925		S
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803	53.1	C123	S
5	0	3	Allen, Mr. William Henry	male	35	0	0	373450	8.05		S
6	0	3	Moran, Mr. James	male		0	0	330877	8.4583		Q
7	0	1	McCarthy, Mr. Timothy J	male	54	0	0	17463	51.8625	E46	S
8	0	3	Palsson, Master. Gosta Leonard	male	2	3	1	349909	21.075		S
9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27	0	2	347742	11.1333		S
10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14	1	0	237736	30.0708		C
11	1	3	Sandstrom, Miss. Marguerite Rut	female	4	1	1	PP 9549	16.7	G6	S
12	1	1	Bonnell, Miss. Elizabeth	female	58	0	0	113783	26.55	C103	S
13	0	3	Saundercock, Mr. William Henry	male	20	0	0	A/5. 2151	8.05		S
14	0	3	Andersson, Mr. Anders Johan	male	39	1	5	347082	31.275		S
15	0	3	Vestrom, Miss. Hulda Amanda Adolfina	female	14	0	0	350406	7.8542		S
16	1	2	Hewlett, Mrs. (Mary D Kingcome)	female	55	0	0	248706	16		S
17	0	3	Rice, Master. Eugene	male	2	4	1	382652	29.125		Q
18	1	2	Williams, Mr. Charles Eugene	male		0	0	244373	13		S
19	0	3	Vander Planke, Mrs. Julius (Emelia Maria Vandemoortele)	female	31	1	0	345763	18		S
20	1	3	Masselmani, Mrs. Fatima	female		0	0	2649	7.225		C

The data contains the following fields:

- PassengerID
- Survived
- Pclass
- Name
- Sex
- Age
- SibSp
- Parch
- Ticket
- Fare
- Cabin
- Embarked

Overview of data information

```
train_data.info()
print("-" * 40)
test_data.info()
```

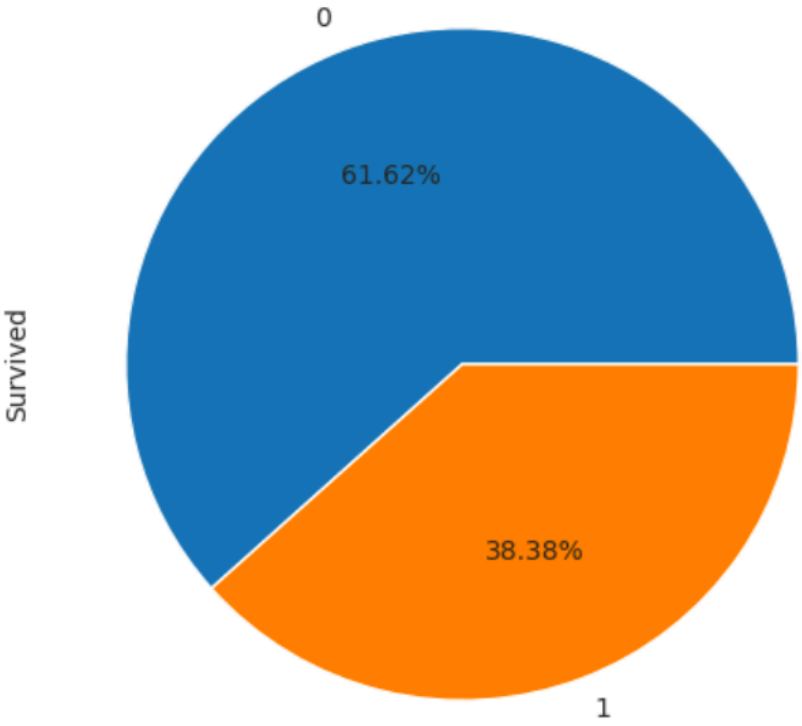
```
/Users/lizhihao/.conda/envs/Titanic/bin/python /Users/lizhihao/PycharmProjects/Titanic/test.py
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
PassengerId    891 non-null int64
Survived       891 non-null int64
Pclass         891 non-null int64
Name           891 non-null object
Sex            891 non-null object
Age            714 non-null float64
SibSp          891 non-null int64
Parch          891 non-null int64
Ticket         891 non-null object
Fare           891 non-null float64
Cabin          204 non-null object
Embarked       889 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.6+ KB
```

```
----------------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
PassengerId    418 non-null int64
Pclass         418 non-null int64
Name           418 non-null object
Sex            418 non-null object
Age            332 non-null float64
SibSp          418 non-null int64
Parch          418 non-null int64
Ticket         418 non-null object
Fare           417 non-null float64
Cabin          91 non-null object
Embarked       418 non-null object
dtypes: float64(2), int64(4), object(5)
memory usage: 36.0+ KB
```

It can be seen from the above that there are missing values for the features of Age, Cabin, heating and Fare, and corresponding treatment methods are provided below.

Below is a pie chart to plot the proportion of survival

```
train_data['Survived'].value_counts().plot.pie(autopct = '%1.2f%%')
plt.show()
```



2 The missing value processing method

When analyzing data, pay attention to whether there are missing values.

Some machine learning algorithms can handle missing values, such as neural networks, and some cannot. For missing values, the following processing methods are commonly used:

(1) if the data set is large but has few missing values, the rows with missing values can be deleted.

(2) if the attribute is less important than learning, mean or mode can be assigned to the missing value. So where do we go from there to the boarding of this property (there are three boarding points), missing two values, which can be assigned by the mode

train_data.Embarked[train_data.Embarked.isnull()] = train_data.Embarked.dropna().mode().values

(3) for nominal attributes, you can assign a value representing the missing, such as' U0 '. Because the absence itself may represent some implied information. For example, the missing property of Cabin may mean there is no Cabin.

#replace missing value with U0

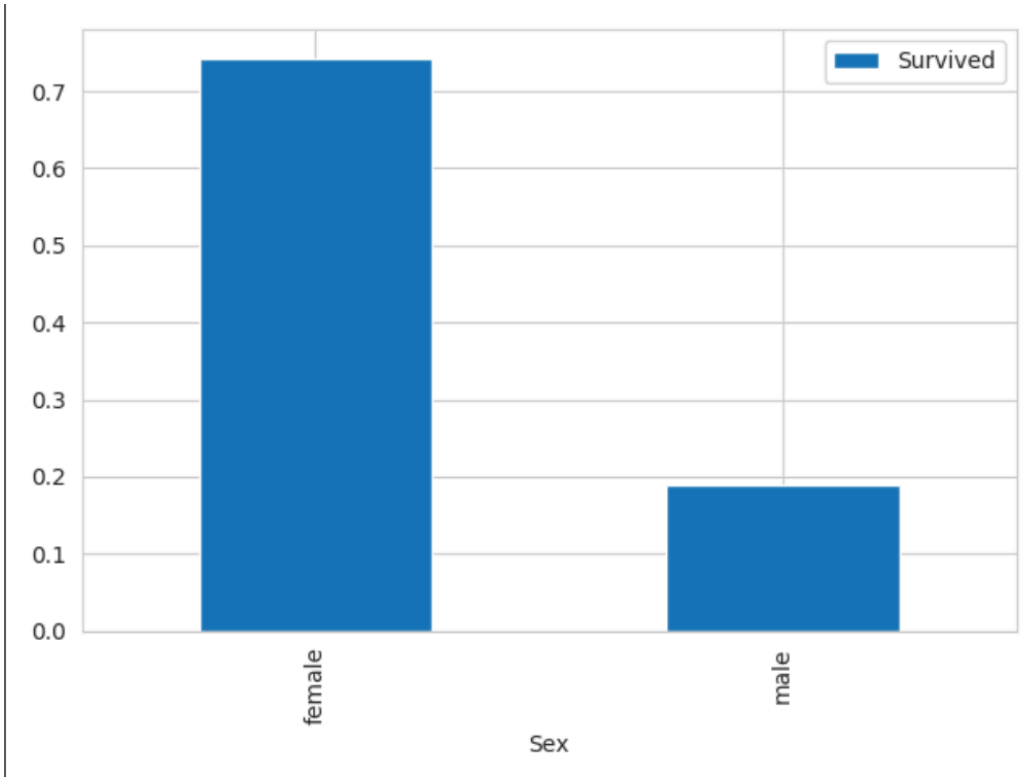
train_data['Cabin'] = train_data.Cabin.fillna('U0') # train_data.Cabin[train_data.Cabin.isnull()]='U0'

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
PassengerId    891 non-null int64
Survived       891 non-null int64
Pclass         891 non-null int64
Name           891 non-null object
Sex            891 non-null object
Age            714 non-null float64
SibSp          891 non-null int64
Parch          891 non-null int64
Ticket         891 non-null object
Fare           891 non-null float64
Cabin          891 non-null object
Embarked       891 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.6+ KB
```

3 Analyze data relations

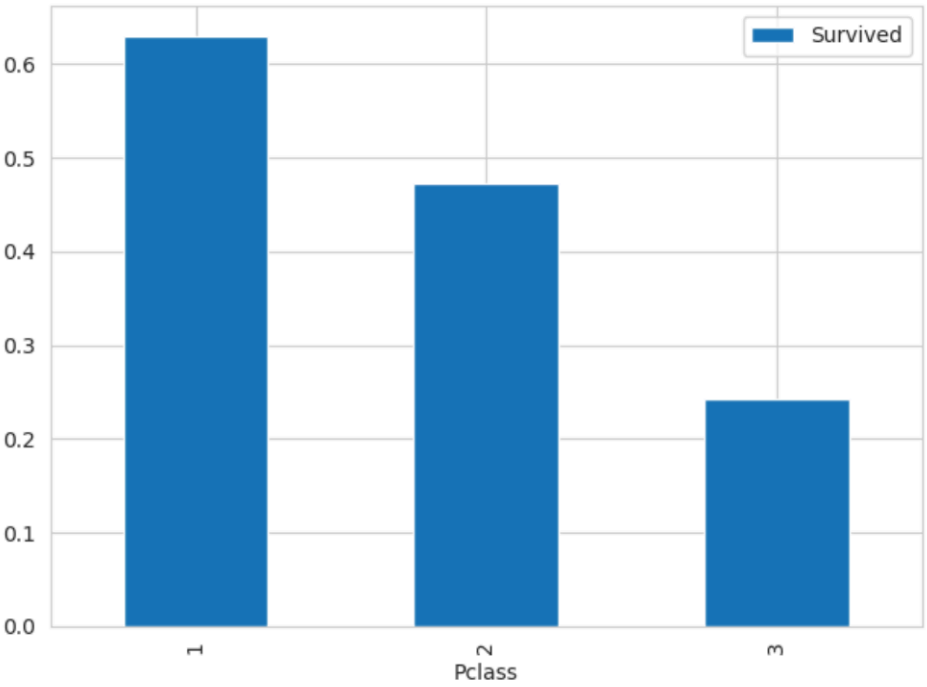
1. The relationship between gender and survival

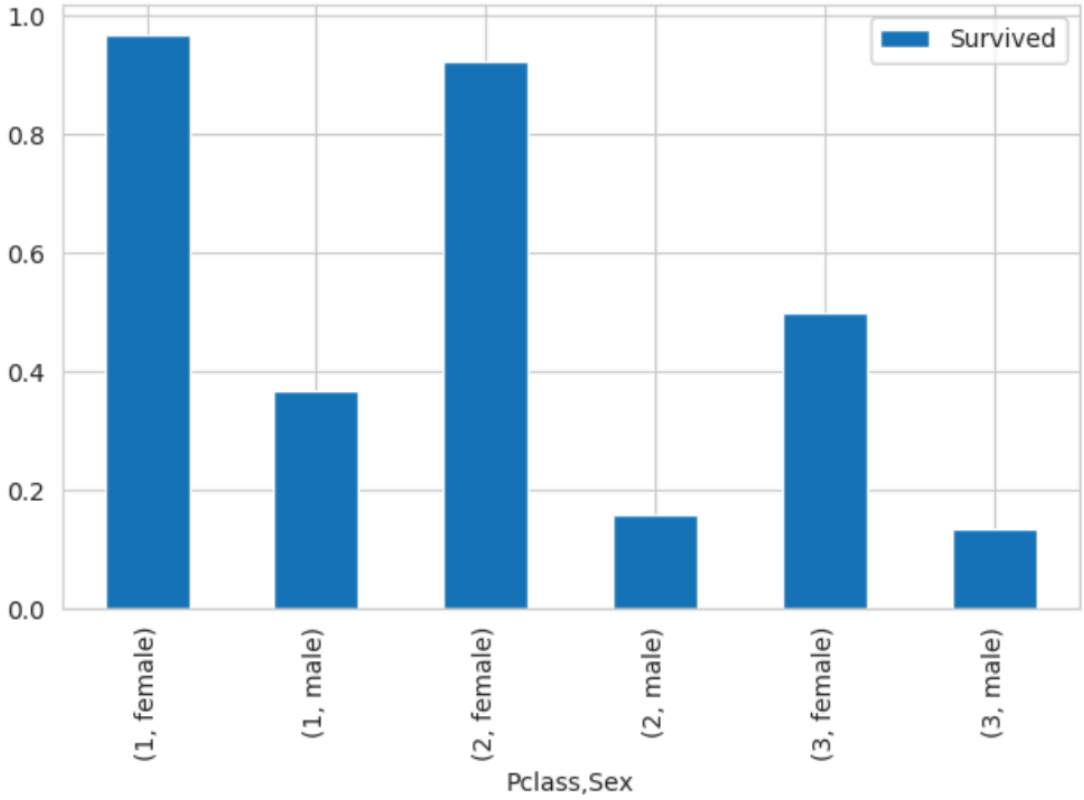
```
train_data.groupby(['Sex','Survived'])['Survived'].count()
train_data[['Sex','Survived']].groupby(['Sex']).mean().plot.bar()
plt.show()
```



2. The relationship between cabin class and survival

```
train_data.groupby(['Pclass','Survived'])['Pclass'].count()
train_data[['Pclass','Survived']].groupby(['Pclass']).mean().plot.bar()
train_data[['Sex','Pclass','Survived']].groupby(['Pclass','Sex']).mean().
plot.bar()

train_data.groupby(['Sex', 'Pclass', 'Survived'])['Survived'].count()
plt.show()
```

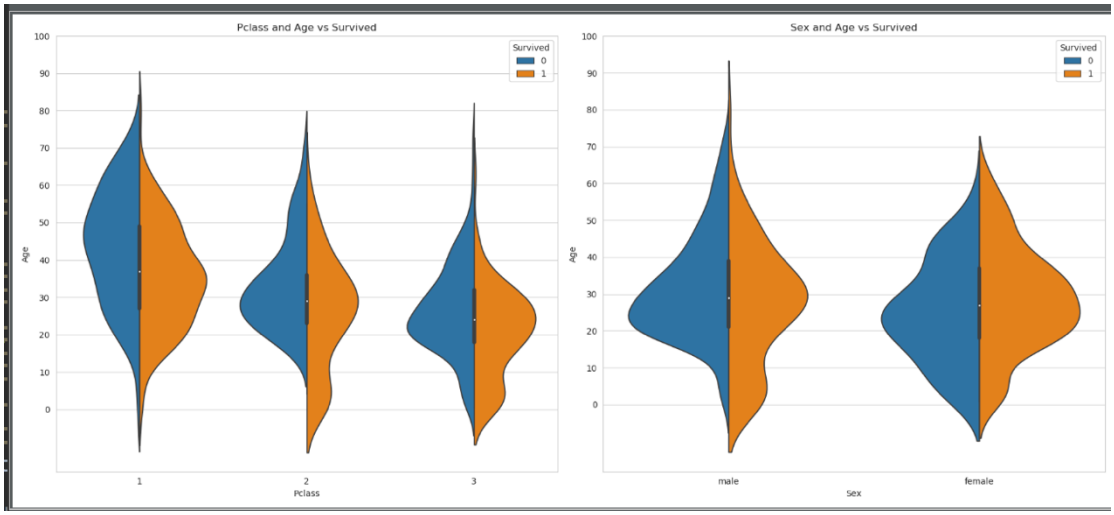




3. The relationship between age and survival

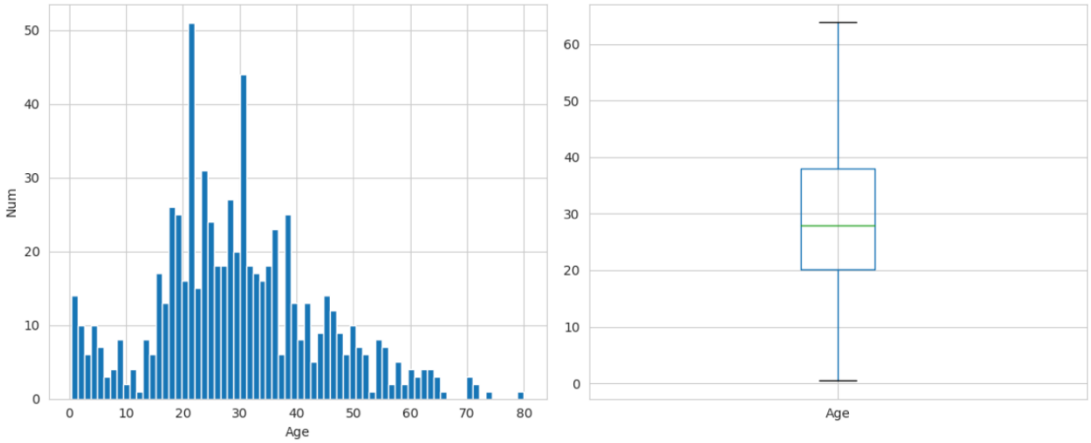
(1) the relationship between age distribution and survival under different cabin classes and different genders

```
fig, ax = plt.subplots(1, 2, figsize = (18, 8))
sns.violinplot("Pclass", "Age", hue="Survived", data=train_data,
split=True, ax=ax[0])
ax[0].set_title('Pclass and Age vs Survived')
ax[0].set_yticks(range(0, 110, 10))

sns.violinplot("Sex", "Age", hue="Survived", data=train_data, split=True,
ax=ax[1])
ax[1].set_title('Sex and Age vs Survived')
ax[1].set_yticks(range(0, 110, 10))
plt.show()
```



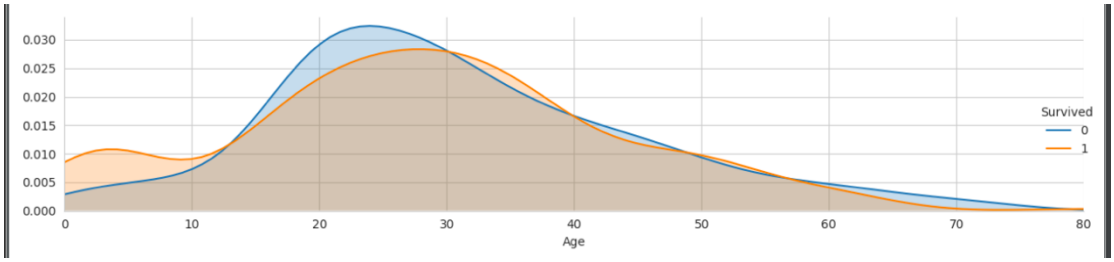
(2) analyze the overall age distribution

```
plt.figure(figsize=(12,5))
plt.subplot(121)
train_data['Age'].hist(bins=70)
plt.xlabel('Age')
plt.ylabel('Num')

plt.subplot(122)
train_data.boxplot(column='Age', showfliers=False)
plt.show()
```



(3) the relationship between survival and non-survival under different ages

```
facet = sns.FacetGrid(train_data, hue="Survived",aspect=4)
facet.map(sns.kdeplot,'Age',shade= True)
facet.set(xlim=(0, train_data['Age'].max()))
facet.add_legend()
plt.show()
```

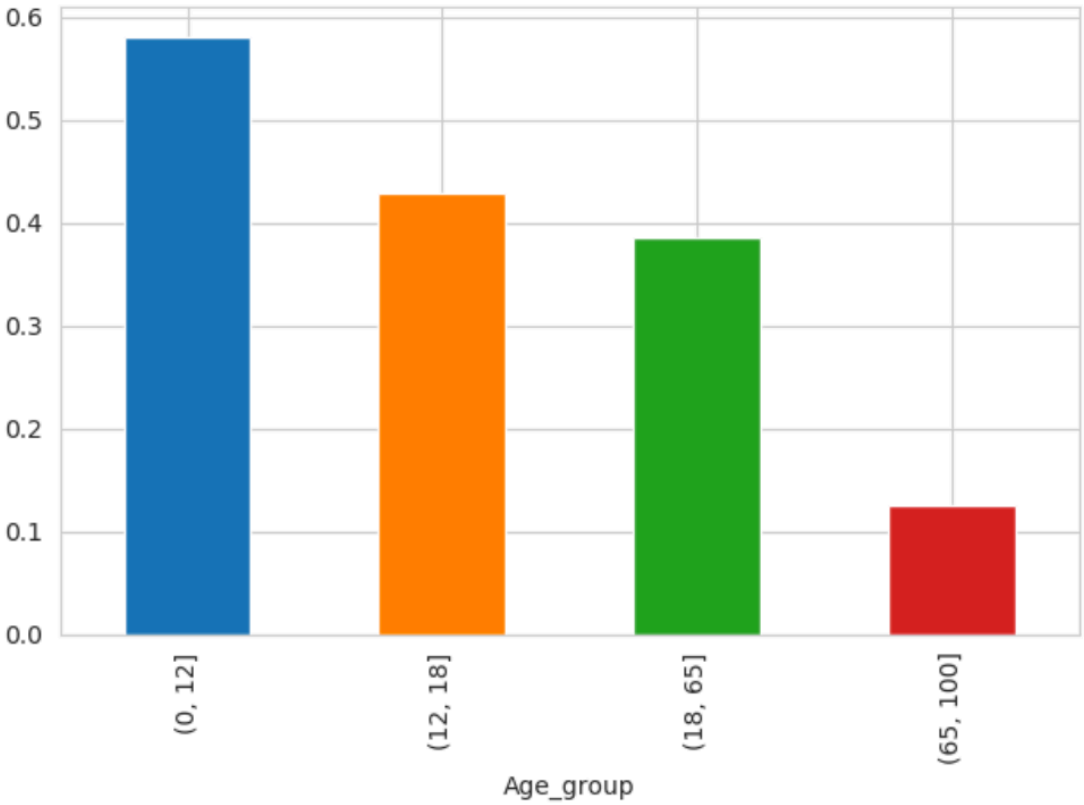


(4) survival average rate at different ages

The sample was 891, with an average age of about 30 years old, standard deviation of 13.5 years old, minimum age of 0.42 and maximum age of 80. Passengers were divided into children, teenagers, adults and the elderly according to their age to analyze the survival of the four groups

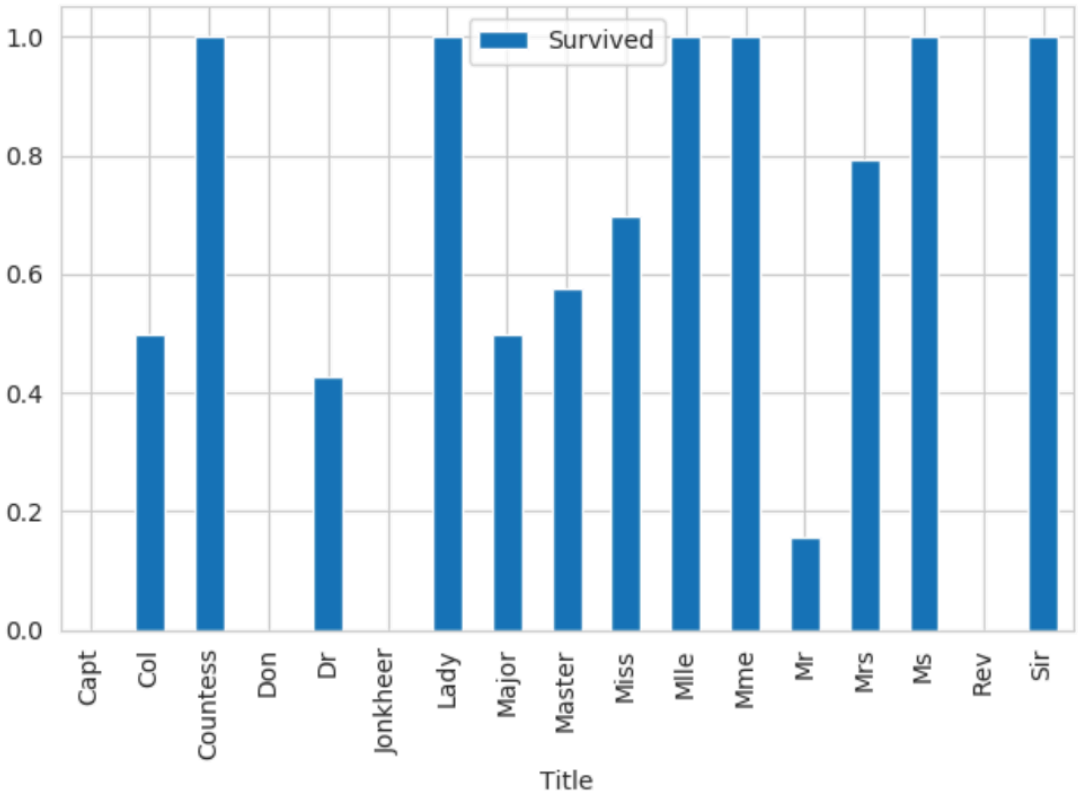
```
train_data['Age'].describe()
bins = [0, 12, 18, 65, 100]
train_data['Age_group'] = pd.cut(train_data['Age'], bins)
by_age = train_data.groupby('Age_group')['Survived'].mean()
by_age
```

```
by_age.plot(kind = 'bar')
plt.show()
```



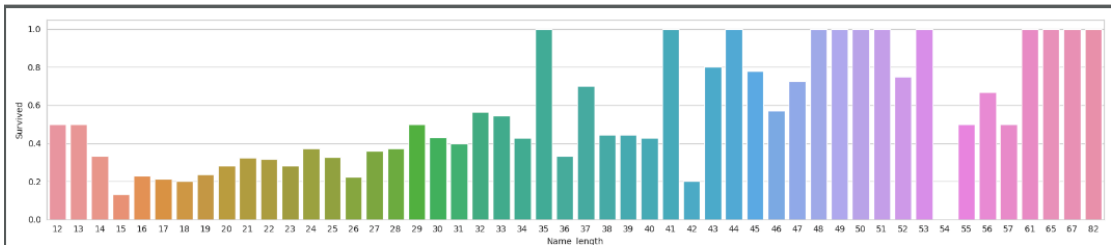
4. (1) the relationship between appellation and survival

```
train_data['Title'] = train_data['Name'].str.extract(' ([A-Za-z]+)\.',
expand=False)

pd.crosstab(train_data['Title'], train_data['Sex'])
train_data[['Title','Survived']].groupby(['Title']).mean().plot.bar()
plt.show()
```



(2) the relationship between name length and survival

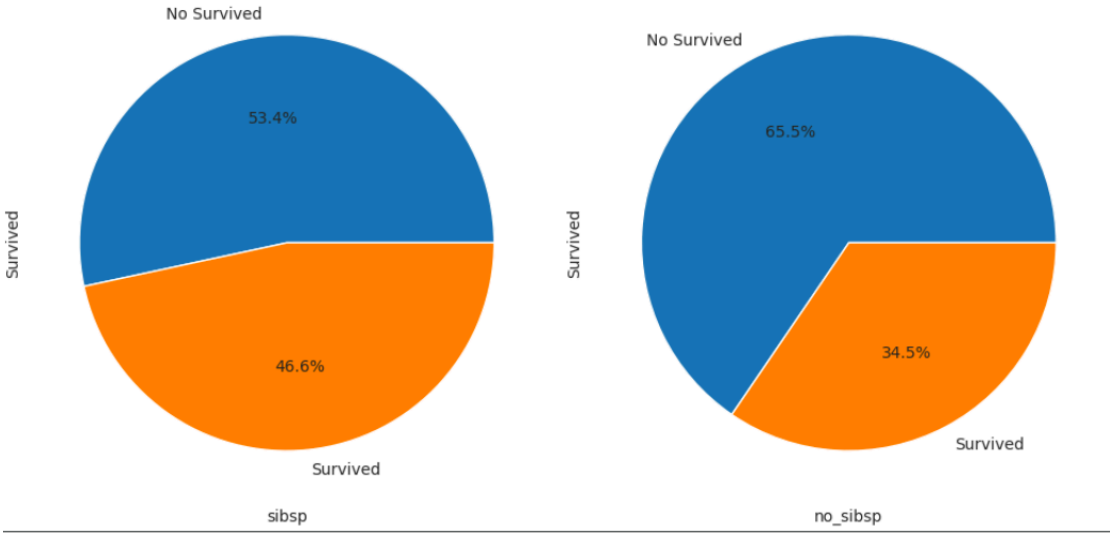
```
fig, axis1 = plt.subplots(1,1,figsize=(18,4))
train_data['Name_length'] = train_data['Name'].apply(len)
name_length =
train_data[['Name_length','Survived']].groupby(['Name_length'],as_index=Fals
e).mean()
sns.barplot(x='Name_length', y='Survived', data=name_length)
plt.show()
```



5. The relationship between siblings and survival

```
# 将数据分为有兄弟姐妹的和没有兄弟姐妹的两组:
sibsp_df = train_data[train_data['SibSp'] != 0]
```

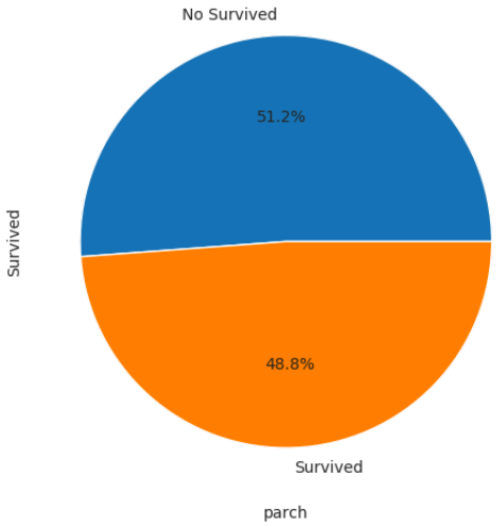
```
no_sibsp_df = train_data[train_data['SibSp'] == 0]

plt.figure(figsize=(10,5))
plt.subplot(121)
sibsp_df['Survived'].value_counts().plot.pie(labels=['No Survived',
'Survived'], autopct = '%1.1f%%')
plt.xlabel('sibsp')

plt.subplot(122)
no_sibsp_df['Survived'].value_counts().plot.pie(labels=['No Survived',
'Survived'], autopct = '%1.1f%%')
plt.xlabel('no_sibsp')

plt.show()
```

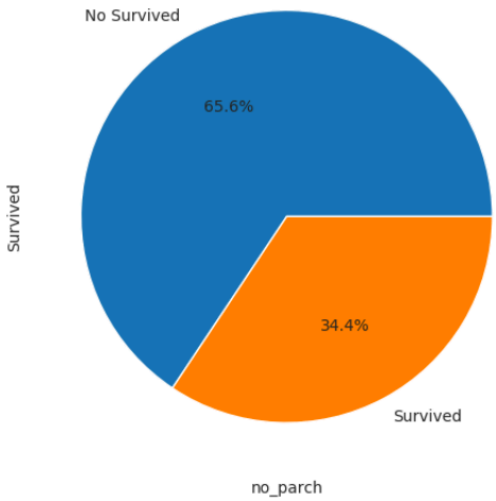


6. The relationship between having parents or children and survival

```
parch_df = train_data[train_data['Parch'] != 0]
no_parch_df = train_data[train_data['Parch'] == 0]

plt.figure(figsize=(10,5))
plt.subplot(121)
parch_df['Survived'].value_counts().plot.pie(labels=['No Survived',
'Survived'], autopct = '%1.1f%%')
plt.xlabel('parch')

plt.subplot(122)
no_parch_df['Survived'].value_counts().plot.pie(labels=['No Survived',
'Survived'], autopct = '%1.1f%%')
plt.xlabel('no_parch')
```

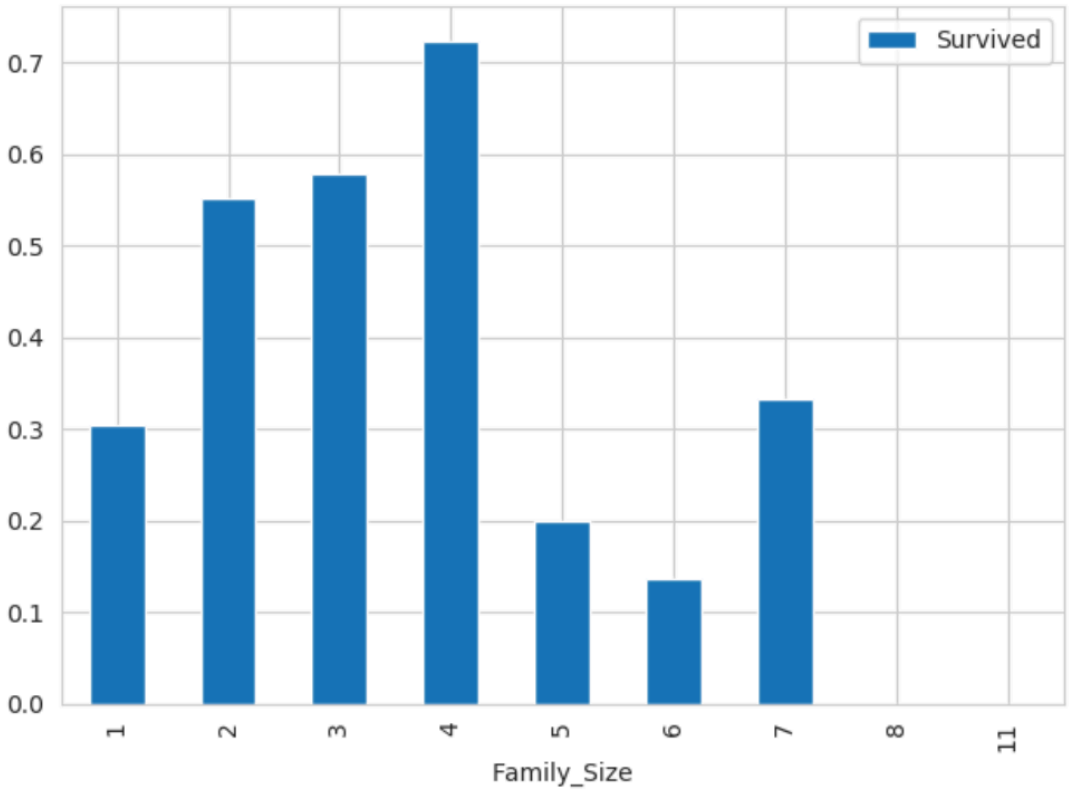
```
plt.show()
```





7. The relationship between family size and the survival

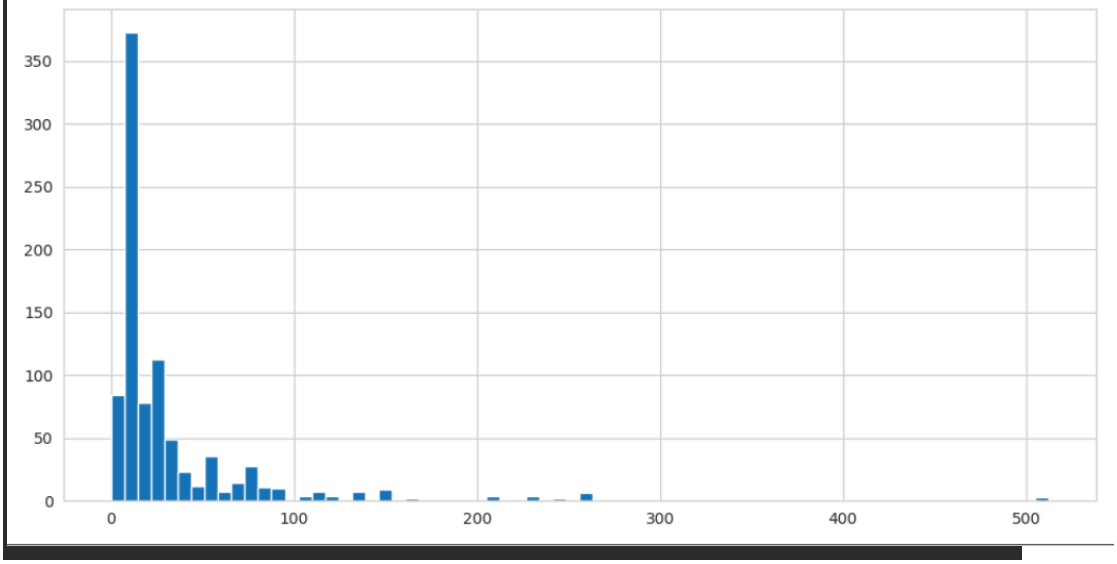
```
train_data['Family_Size'] = train_data['Parch'] + train_data['SibSp'] + 1
train_data[['Family_Size','Survived']].groupby(['Family_Size']).mean().pl
ot.bar()
plt.show()
```

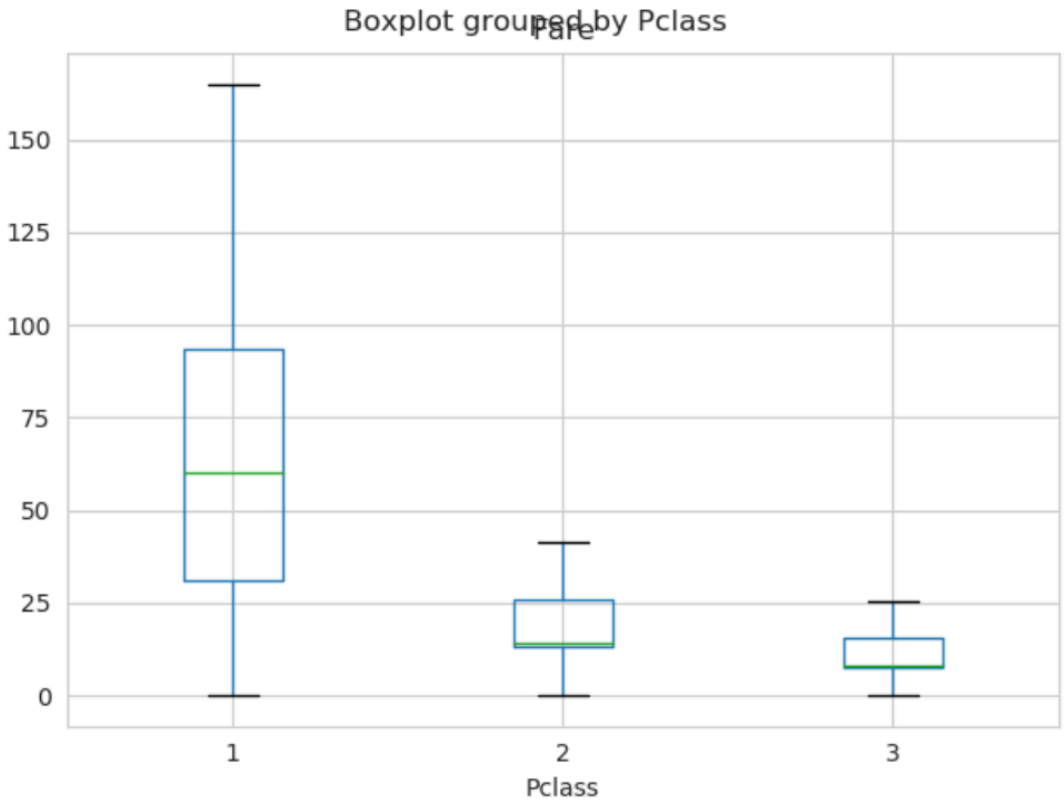


8. The relationship between fare distribution and survival

(1) first, plot the distribution of fare

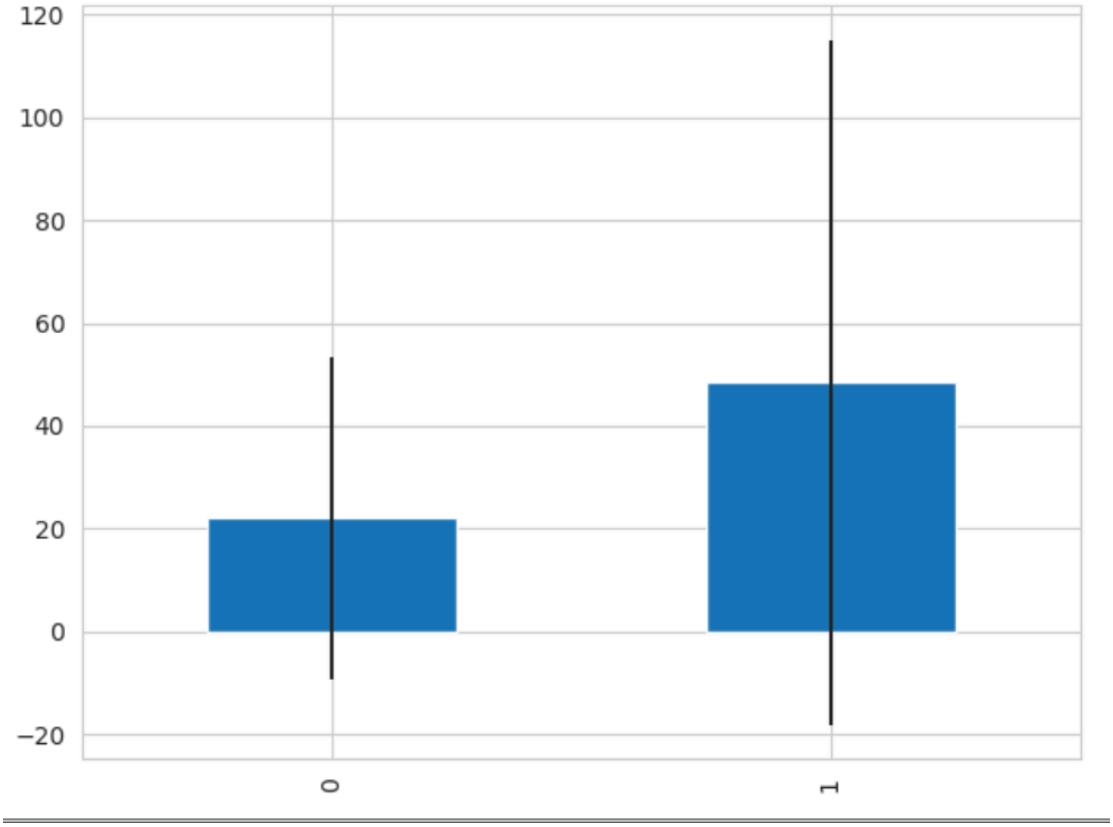
```
train_data.boxplot(column='Fare', by='Pclass', showfliers=False)
plt.show()
```





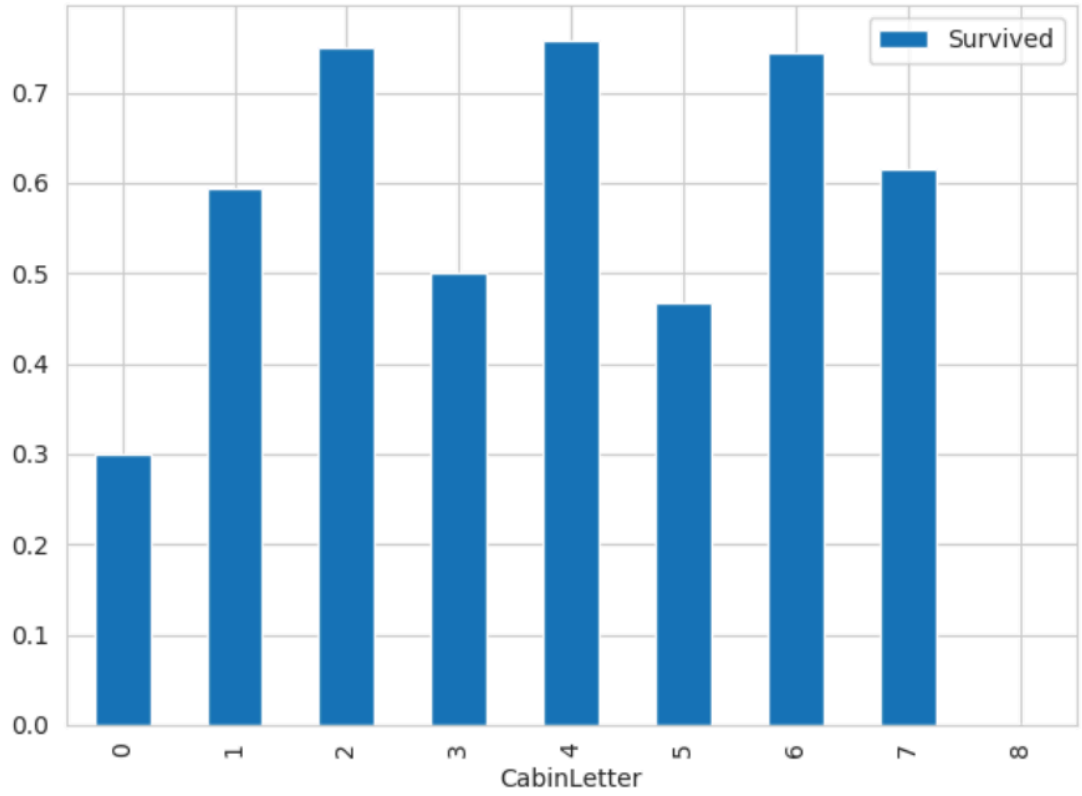
(2) the relationship between ticket average and variance and survival

```
train_data['Fare'].describe()
fare_not_survived = train_data['Fare'][train_data['Survived'] == 0]
fare_survived = train_data['Fare'][train_data['Survived'] == 1]

average_fare = pd.DataFrame([fare_not_survived.mean(),
fare_survived.mean()])
std_fare = pd.DataFrame([fare_not_survived.std(),
fare_survived.std()])
average_fare.plot(yerr=std_fare, kind='bar', legend=False)

plt.show()
```



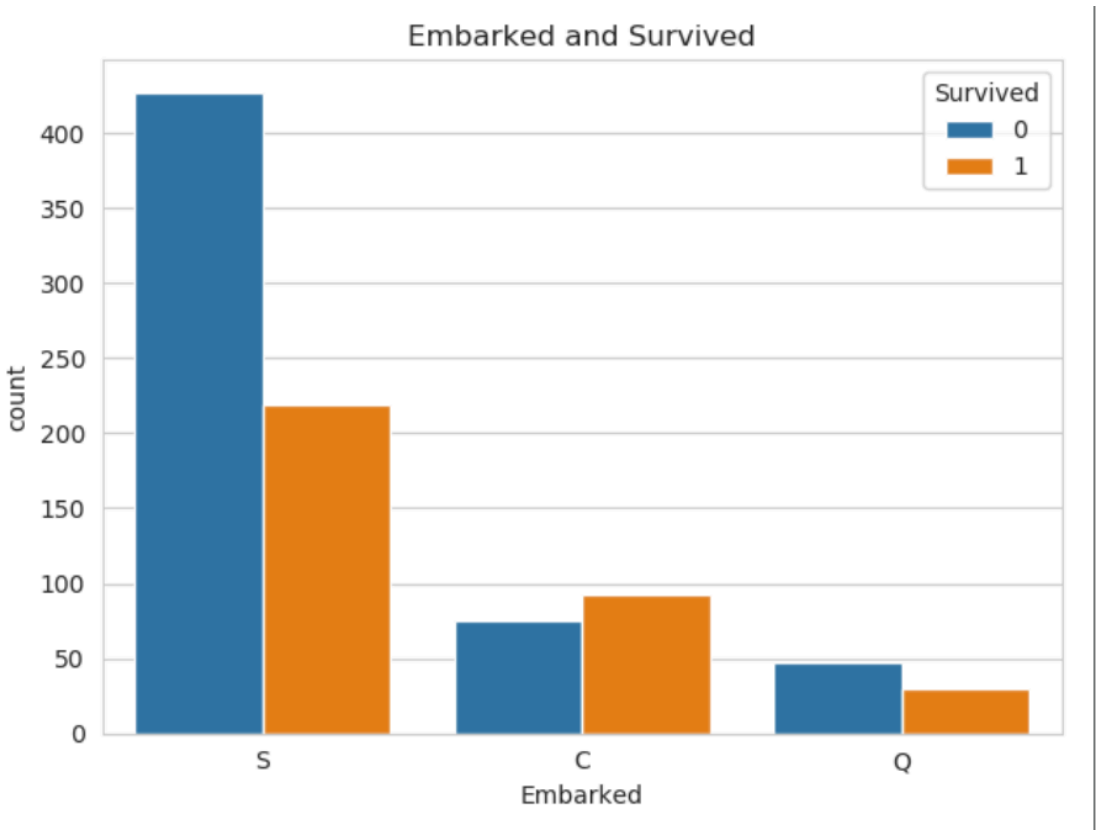
9. The relationship between cabin type and survival

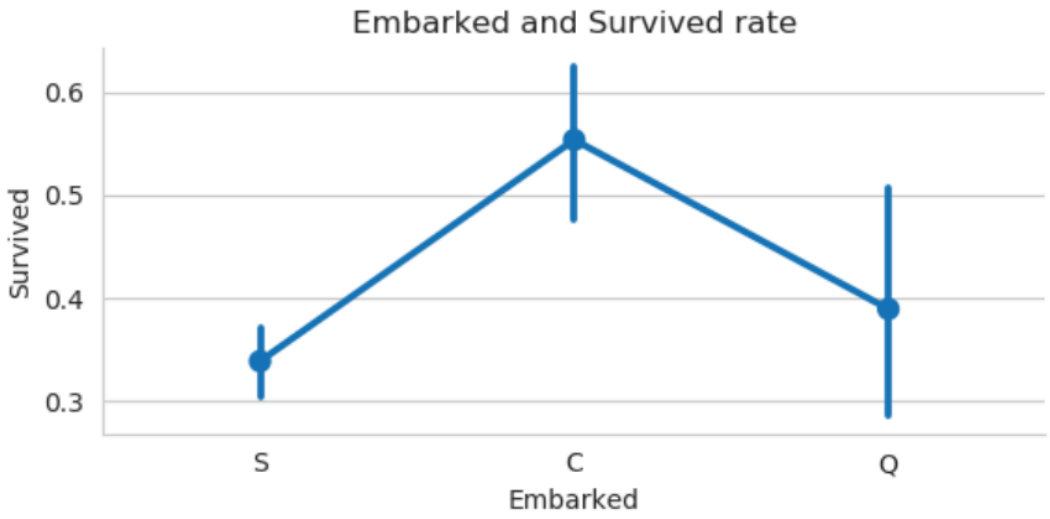
```
# Replace missing values with "U0"
train_data.loc[train_data.Cabin.isnull(), 'Cabin'] = 'U0'
train_data['Has_Cabin'] = train_data['Cabin'].apply(lambda x: 0 if x ==
'U0' else 1)
train_data[['Has_Cabin','Survived']].groupby(['Has_Cabin']).mean().plot.b
ar()
#plt.show()
# create feature for the alphabetical part of the cabin number
train_data['CabinLetter'] = train_data['Cabin'].map(lambda x:
re.compile("([a-zA-Z]+)").search(x).group())
# convert the distinct cabin letters with incremental integer values
train_data['CabinLetter'] = pd.factorize(train_data['CabinLetter'])[0]
train_data[['CabinLetter','Survived']].groupby(['CabinLetter']).mean().pl
ot.bar()
plt.show()
```



10. The relationship between port and survival

```
sns.countplot('Embarked', hue='Survived', data=train_data)
plt.title('Embarked and Survived')

sns.factorplot('Embarked', 'Survived', data=train_data, size=3, aspect=2)
plt.title('Embarked and Survived rate')
plt.show()
```


Embarked and Survived
Survived
0
1
count
400
350
300
250
200
150
100
50
0
S
C
Q
Embarked


Embarked and Survived rate
Survived
0.6
0.5
0.4
0.3
S
C
Q
Embarked