exploratory_data_analysis

October 15, 2020

0.1 Build your own EDA of the Titanic dataset here!!!

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
sns.set()

import warnings
warnings.filterwarnings("ignore", category=FutureWarning)

df = pd.read_csv("~/data/titanic.csv")
```

[2]:
```
df.shape
```

[2]:
```
(891, 12)
```

[3]:
```
df.head
```

[3]:
```
<bound method NDFrame.head of      PassengerId  Survived  Pclass  \
0              1         0       3
1              2         1       1
2              3         1       3
3              4         1       1
4              5         0       3
..           …       …     …
886          887         0       2
887          888         1       1
888          889         0       3
889          890         1       1
890          891         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
..                                                  …     …   …    …
886                               Montvila, Rev. Juozas    male  27.0      0
887                        Graham, Miss. Margaret Edith  female  19.0      0
888            Johnston, Miss. Catherine Helen "Carrie"  female   NaN      1
889                               Behr, Mr. Karl Howell    male  26.0      0
890                                 Dooley, Mr. Patrick    male  32.0      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
..     …               …       …   …      …
886      0            211536  13.0000   NaN        S
887      0            112053  30.0000   B42        S
888      2        W./C. 6607  23.4500   NaN        S
889      0            111369  30.0000  C148        C
890      0            370376   7.7500   NaN        Q

[891 rows x 12 columns]>
```

[4]:
```
df.columns.values
```

[4]:
```
array(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'], dtype=object)
```

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

```
<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: 83.7+ KB
```

[6]:
```
df.isnull().sum()

#Conclusions:
# 1) Missing values in Age, Cabin and Embarked columns
# 2) More than 70% percent values are missing in Cabin column. Will have to␣
 ↪drop.
# 3) Few columns have inappropriate data types
```

[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]:
```
# Dropping cabin column

df.drop(columns = ['Cabin'], inplace = True)
```

[8]:
```
# Filling the missing values for Age with 'mean' strategy

df['Age'].fillna(df['Age'].mean(), inplace = True)
```

[9]:
```
# Filling the missing values for Embarked with the most appeared value in␣
 ↪embarked column (S)

df['Embarked'].value_counts()
df['Embarked'].fillna('S', inplace = True)
```

[10]:
```
# Finding categories for SibSp column

df['SibSp'].value_counts()
```

[10]:
```
0    608
1    209
2     28
4     18
3     16
```

```
8      7
5      5
Name: SibSp, dtype: int64
```

[11]:
```
# Finding categories for Parch column

df['Parch'].value_counts()
```

[11]:
```
0    678
1    118
2     80
5      5
3      5
4      4
6      1
Name: Parch, dtype: int64
```

[12]:
```
# Changing data types for Survived, PClass, Sex, Age and Embarked

df['Survived'] = df['Survived'].astype('category')
df['Pclass'] = df['Pclass'].astype('category')
df['Sex'] = df['Sex'].astype('category')
df['Age'] = df['Age'].astype('int')
df['Embarked'] = df['Embarked'].astype('category')
```

[13]:
```
# all empty value are filled and categories for the above 5 columns are changed
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 11 columns):
PassengerId    891 non-null int64
Survived       891 non-null category
Pclass         891 non-null category
Name           891 non-null object
Sex            891 non-null category
Age            891 non-null int64
SibSp          891 non-null int64
Parch          891 non-null int64
Ticket         891 non-null object
Fare           891 non-null float64
Embarked       891 non-null category
dtypes: category(4), float64(1), int64(4), object(2)
memory usage: 52.7+ KB
```

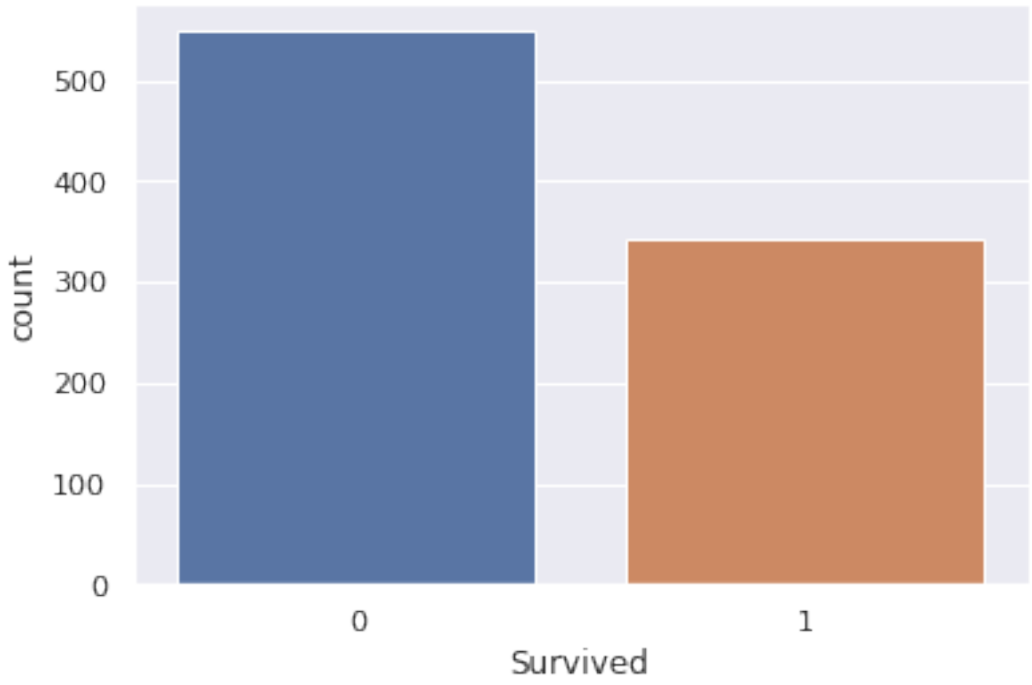
[14]:
```
# Five point summary
df.describe()
```

[14]:

	PassengerId	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	891.000000	891.000000
mean	446.000000	29.544332	0.523008	0.381594	32.204208
std	257.353842	13.013778	1.102743	0.806057	49.693429
min	1.000000	0.000000	0.000000	0.000000	0.000000
25%	223.500000	22.000000	0.000000	0.000000	7.910400
50%	446.000000	29.000000	0.000000	0.000000	14.454200
75%	668.500000	35.000000	1.000000	0.000000	31.000000
max	891.000000	80.000000	8.000000	6.000000	512.329200

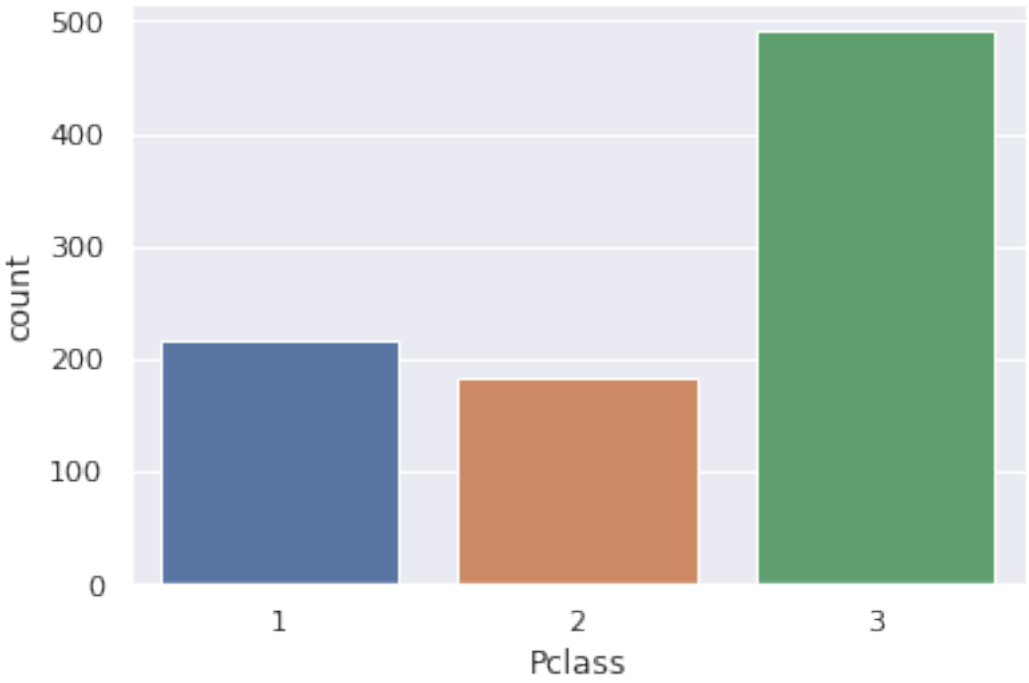
[15]:
```
# how many people survived among 891 passengers

sns.countplot(df['Survived'])
cnt = df['Survived'].value_counts()
print("Number of people did not survive and survived are respectively(out of␣
 ↪891 passengers):")
print(cnt)

ns = round((df['Survived'].value_counts().values[0]/891)*100)
s = round((df['Survived'].value_counts().values[1]/891)*100)

print("percentage of survival is:", s)
print("percentage of death is:", ns)

# Conclusion: more people died. less people survived.
```

```
Number of people did not survive and survived are respectively(out of 891
passengers):
0    549
1    342
Name: Survived, dtype: int64
percentage of survival is: 38.0
percentage of death is: 62.0
```



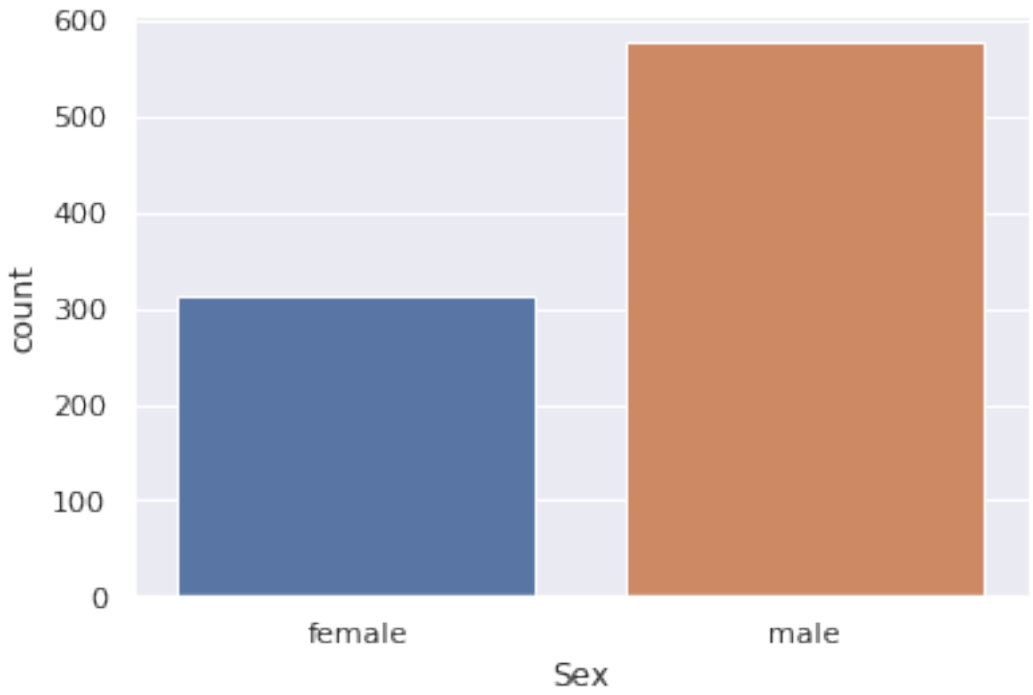
[16]:
```
# Number and Percentage of people who were travelling in class 1/2/3:"

sns.countplot(df['Pclass'])
pc = df['Pclass'].value_counts()
print("Number of people travelled in each Pclass(among 1/2/3):")
print(pc)

p1 = (df['Pclass'].value_counts().values[1]/891)*100
p2 = (df['Pclass'].value_counts().values[2]/891)*100
p3 = (df['Pclass'].value_counts().values[0]/891)*100

print("Percentage of people travelled in pclass 1:", p1)
print("Percentage of people travelled in pclass 2:", p2)
print("Percentage of people travelled in pclass 3:", p3)

# Conclusion: Pclass 3 had the most number of passengers.
```

```
Number of people travelled in each Pclass(among 1/2/3):
3    491
1    216
2    184
Name: Pclass, dtype: int64
Percentage of people travelled in pclass 1: 24.242424242424242
Percentage of people travelled in pclass 2: 20.65095398428732
Percentage of people travelled in pclass 3: 55.106621773288445
```



[17]:
```
# Number and percentage of male and female category:

sns.countplot(df['Sex'])

mf = df['Sex'].value_counts()
print("Number of male and female is:")
print(mf)

m = (df['Sex'].value_counts().values[0]/891)*100
f = (df['Sex'].value_counts().values[1]/891)*100

print("Percentage of male:", m)
print("Percentage of female:", f)

# Conclusion: most of the passengers were male.
```

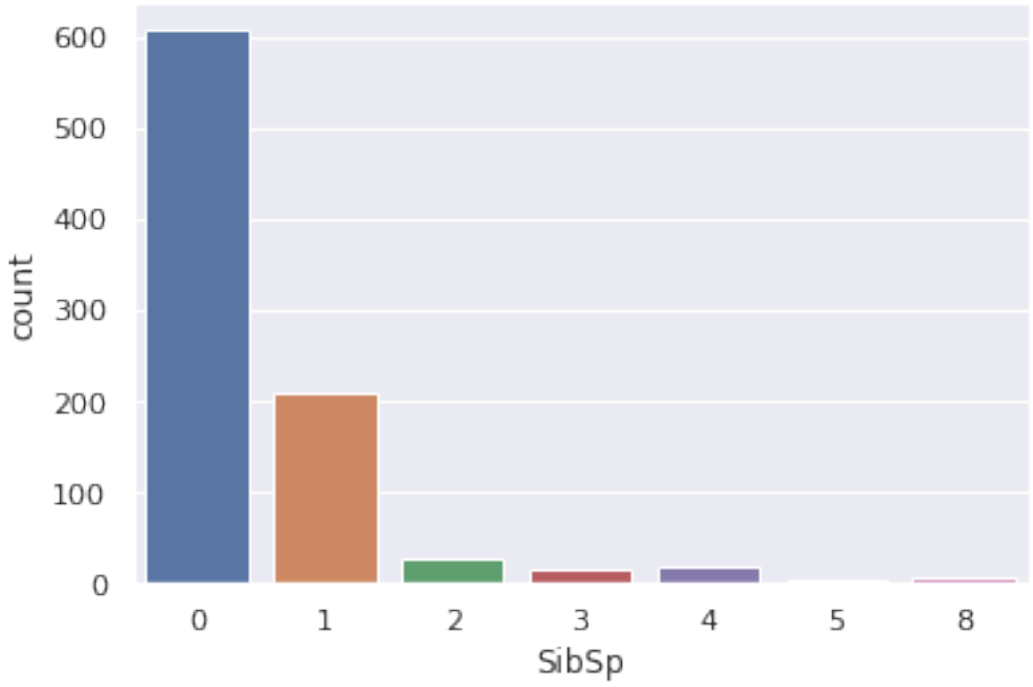
```
Number of male and female is:
male      577
female    314
Name: Sex, dtype: int64
Percentage of male: 64.75869809203144
Percentage of female: 35.24130190796858
```



[18]:
```
# Passenger counts with their number of siblings and spouse
sns.countplot(df['SibSp'])

ss = df['SibSp'].value_counts()
print("Passenger counts with their number of siblings and spouse:")
print(ss)

# percentage of passenger counts with their number of siblings and spouse
ssp = (df['SibSp'].value_counts()/891) * 100
print("Percentage of passenger counts with their number of siblings and spouse:
 ↪")
print(ssp)

# Conclusion: most of the people were travelling without accompanying their␣
 ↪siblings and spouses.
```

```
Passenger counts with their number of siblings and spouse:
0    608
1    209
2     28
4     18
3     16
8      7
5      5
```

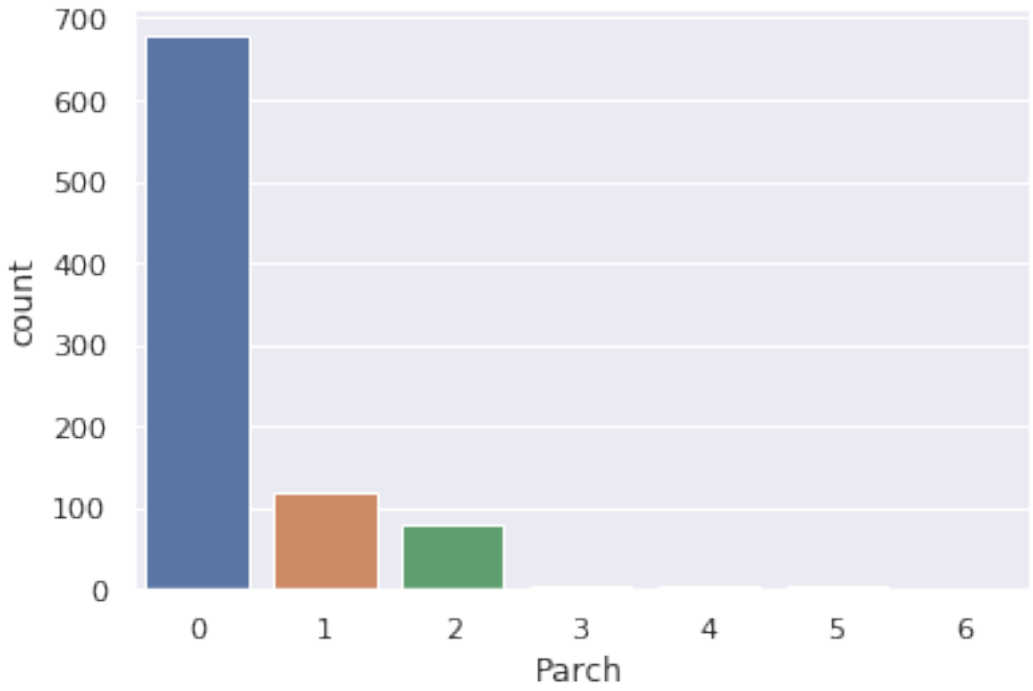
```
Name: SibSp, dtype: int64
Percentage of passenger counts with their number of siblings and spouse:
0    68.237935
1    23.456790
2     3.142536
4     2.020202
3     1.795735
8     0.785634
5     0.561167
Name: SibSp, dtype: float64
```



[19]:
```
# Passenger counts with their number of parents
sns.countplot(df['Parch'])

pc = df['Parch'].value_counts()
print("Passenger counts with their number of parents:")
print(pc)

# percentage of passenger counts with their number of parents
pcp = (df['Parch'].value_counts()/891) * 100
print("Percentage of passenger counts with their number of parents:")
print(pcp)
```

```
# Conclusion: most of the people were travelling without accompanying their␣
↪parents.
```

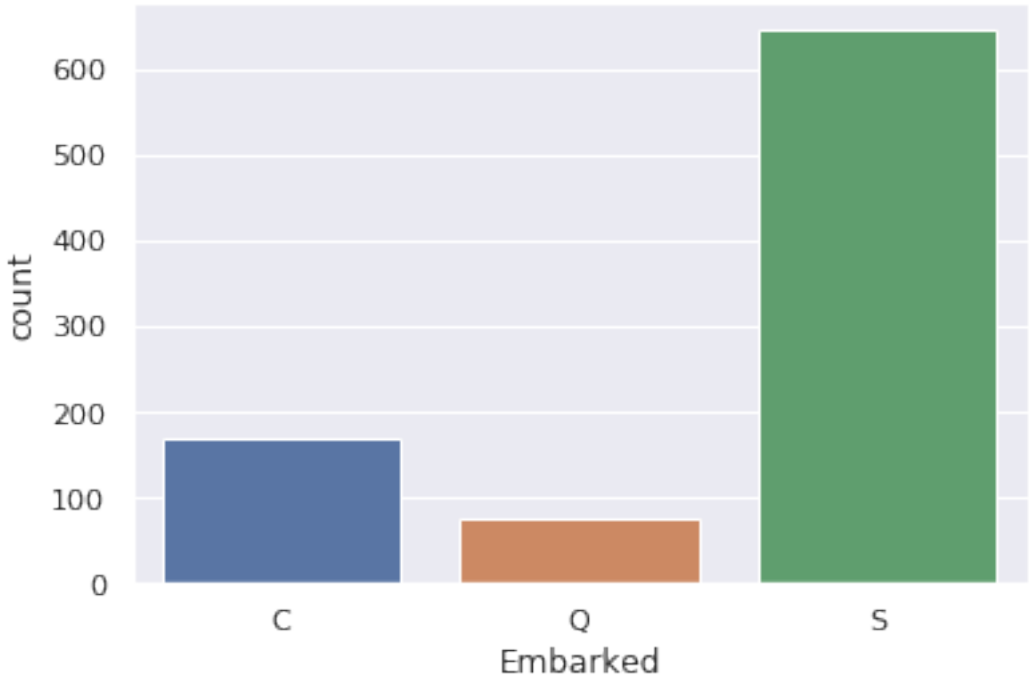
```
Passenger counts with their number of parents:
0    678
1    118
2     80
5      5
3      5
4      4
6      1
Name: Parch, dtype: int64
Percentage of passenger counts with their number of parents:
0    76.094276
1    13.243547
2     8.978676
5     0.561167
3     0.561167
4     0.448934
6     0.112233
Name: Parch, dtype: float64
```



[20]:
```
# Number of people travelling to each station(S/C/Q):
sns.countplot(df['Embarked'])
```

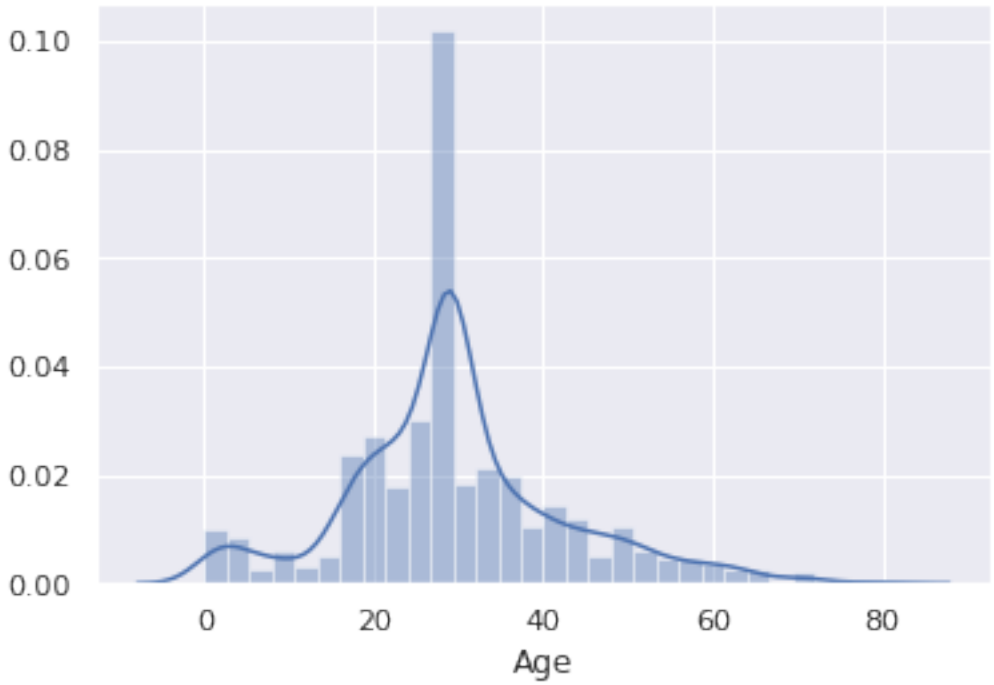
```
e = df['Embarked'].value_counts()
print("Number of people travelling to each station(S/C/Q):")
print(e)

# percentage of passenger counts with their number of parents
ep = (df['Embarked'].value_counts()/891) * 100
print("Percentage of people travelling to each station(S/C/Q):")
print(ep)

# Conclusion: most of the people were travelling to station S.
```

```
Number of people travelling to each station(S/C/Q):
S    646
C    168
Q     77
Name: Embarked, dtype: int64
Percentage of people travelling to each station(S/C/Q):
S    72.502806
C    18.855219
Q     8.641975
Name: Embarked, dtype: float64
```



[21]:
```
# Analyzing age column:

sns.distplot(df['Age'])

print("skew value is:", df['Age'].skew())
print("kurtosis value is:", df['Age'].kurt())

# Conclusion: The plot for age is very close to normal distribution.
# skew() value is 0.4595. values in the range of -0.5 -> 0.5 can be considered as normal distribution.
# So, it can be considered as a normal distribution.
# kurtosis is 0.9866 which means peakedness of the distribution is not very high. It is standard.
# Most of the people were in between the age range of 20 to 40. There were small children also.
# But there were less amount of elderly people.
```

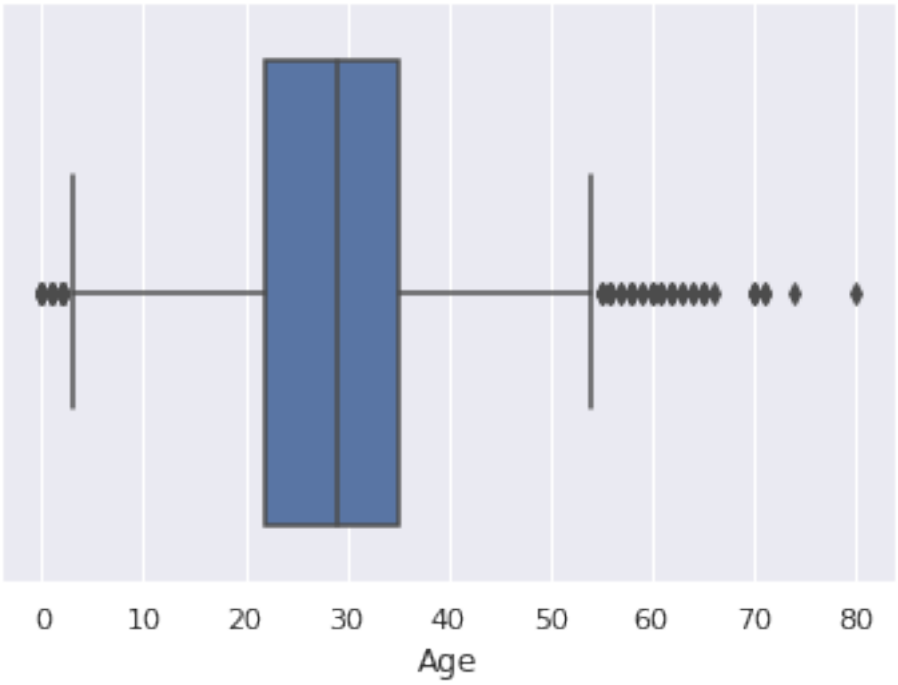
```
skew value is: 0.45956263424701577
kurtosis value is: 0.9865867453652877
```



[22]:
```
# Detecting outliers for age

sns.boxplot(df['Age'])
```

```
# Conclusion: People with ages greater than 55 and around 0 are being␣
↪considered as outliers.
```

[22]: <matplotlib.axes._subplots.AxesSubplot at 0x7f98e59bfdd0>

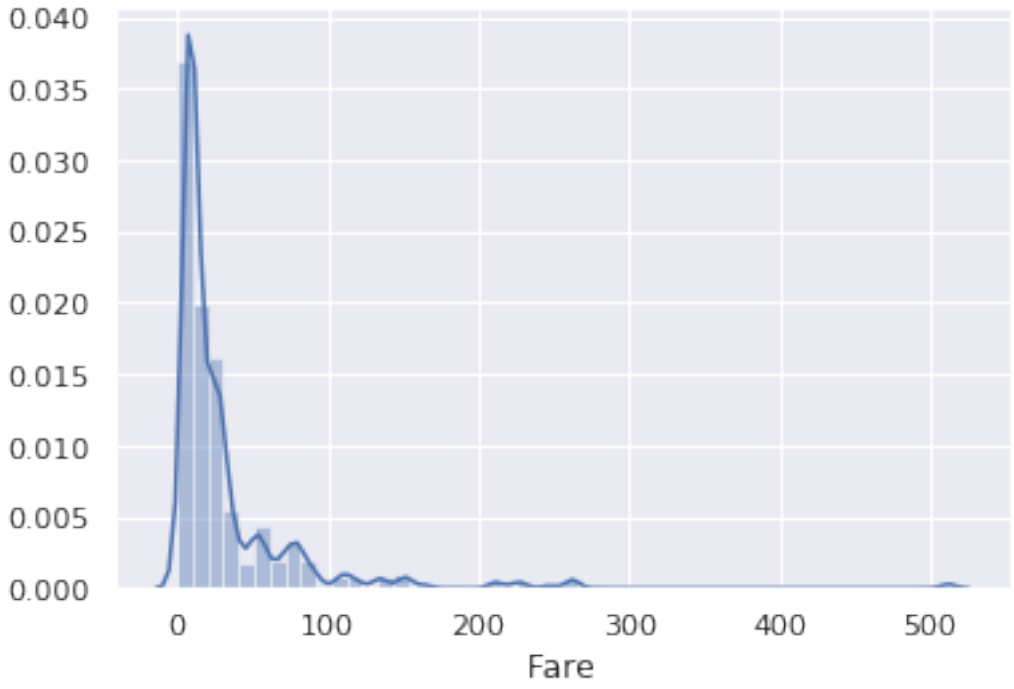


[23]:
```
# Number of people with different age range:

ss = df[(df['Age'] > 60) & (df['Age'] < 70)].shape[0]
print("People with age in between 60 and 70 are:", ss)

sf = df[(df['Age'] >= 70) & (df['Age'] <= 75)].shape[0]
print("People with age from 70 to 75 are:", sf)

gf = df[df['Age'] > 75].shape[0]
print("People with age greater than 75 are:", gf)

zo = df[(df['Age'] > 0) & (df['Age'] <= 1)].shape[0]
print("People with age range from 0 to 1:", zo)

# Conclusion: As the count between the age range between 60 and 70(15), 70 and␣
↪75(6), 0 and 1(7) are not negligible,
# probably it is not a good idea to identify them as outliers.
# So, deeper analysis is required for outlier detection.
```

```
People with age in between 60 and 70 are: 15
People with age from 70 to 75 are: 6
People with age greater than 75 are: 1
People with age range from 0 to 1: 7
```

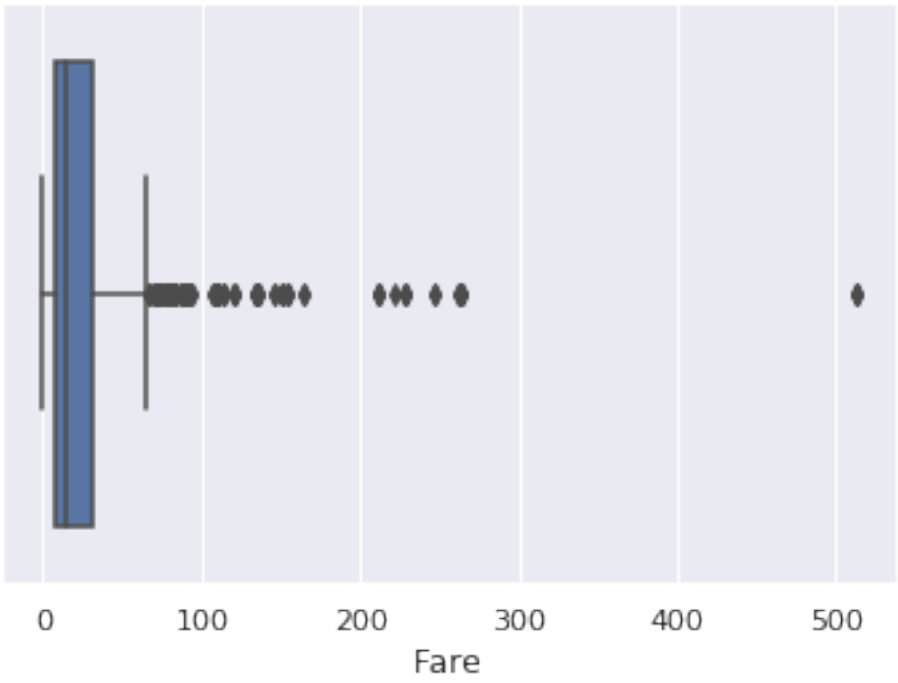
[24]:
```
# Analyzing Fare column

sns.distplot(df['Fare'])

print("skew value is:", df['Fare'].skew())
print("kurtosis value is:", df['Fare'].kurt())

# Conclusion: It has a high skew, high peak and can not be considered as normal␣
 ↪distribution.
# It seems that most of the people bought tickets with low price. That's why␣
 ↪the peak of that population is very high.
```

```
skew value is: 4.787316519674893
kurtosis value is: 33.39814088089868
```



[25]:
```
# Analyzing Fare column

sns.boxplot(df['Fare'])

# Conclusion: there are lot of outliers on the right side.
```

[25]: <matplotlib.axes._subplots.AxesSubplot at 0x7f98e58d6d50>



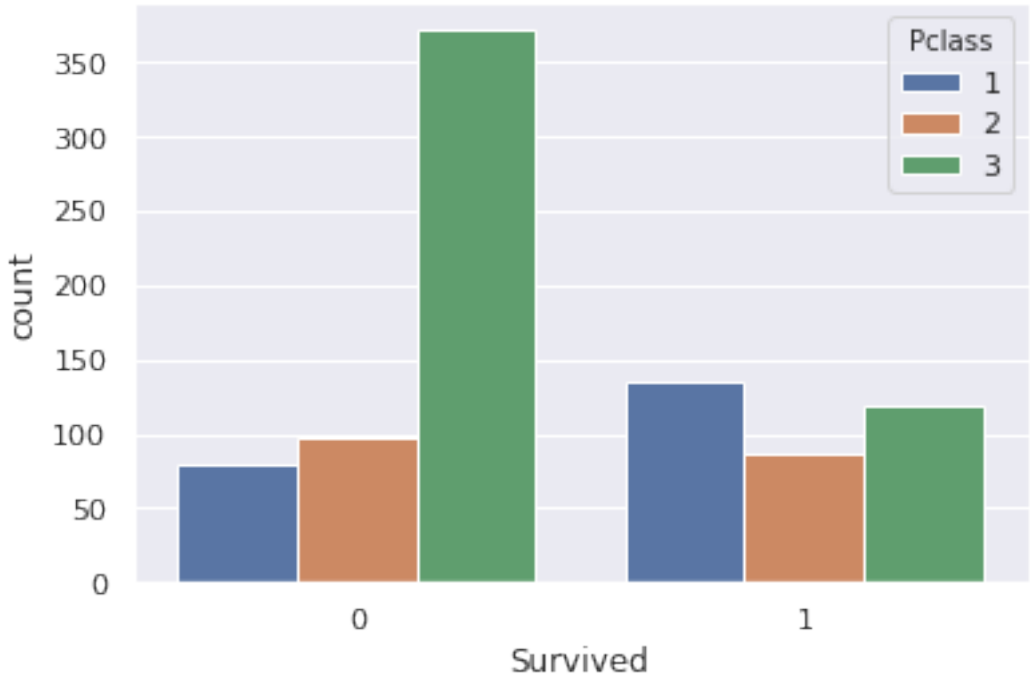
[26]:
```
# Number of people with different fares

tt = df[(df['Fare'] > 200) & (df['Fare'] < 300)].shape[0]
print("People with fare greater than $200 and less than $300 is:", tt)

gt = df[df['Fare'] > 300].shape[0]
print("Number of people with fare greater than $300 is:", gt)

# Conclusion: People with fare between $200 and $300 can not consideres as␣
 ↪outliers. Because there are 17 people in this
# range. But people with fare greater than $300 can be considered as outliers.␣
 ↪Because there are only 3 people in this
# range.
```

```
People with fare greater than $200 and less than $300 is: 17
Number of people with fare greater than $300 is: 3
```

[27]:
```
# Survival with Pclass

# Crosstab between Survived and Pclass
print("Crosstab table for Survived and Pclass:")
sns.countplot(df['Survived'], hue = df['Pclass'])
```

```
print(pd.crosstab(df['Pclass'], df['Survived']))

# percentage
print("Crosstab table in percentage for Survived and Pclass:")
print(pd.crosstab(df['Pclass'], df['Survived']).apply(lambda r: round((r/r.
 ↪sum())*100, 1), axis = 1))

# Conclusion: Most of the people of Pclass 3 died.
```

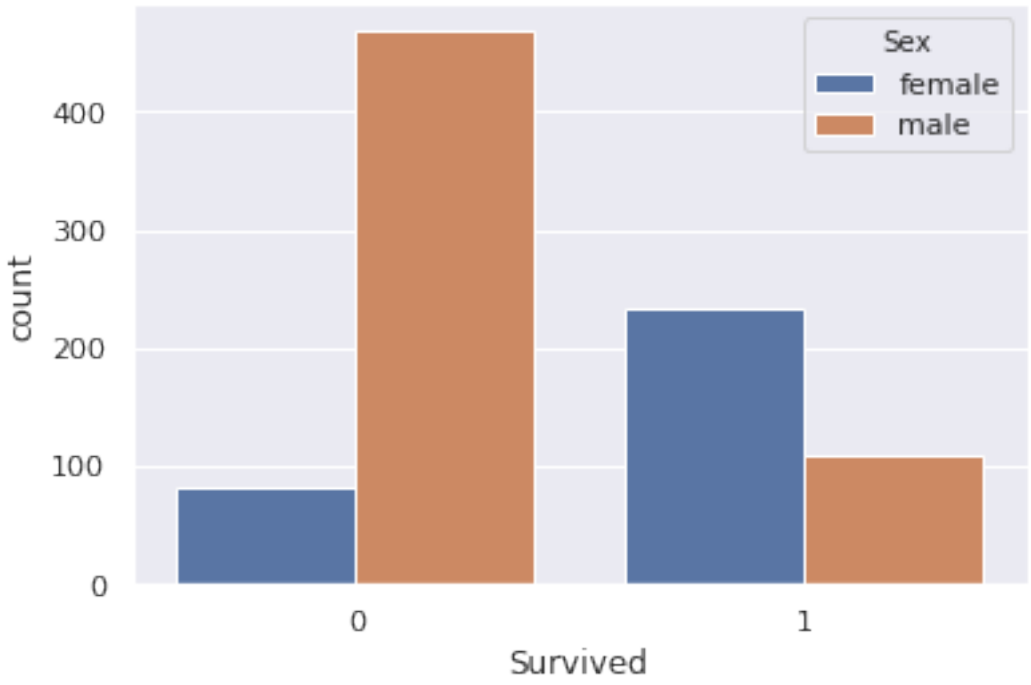
```
Crosstab table for Survived and Pclass:
Survived    0    1
Pclass
1          80  136
2          97   87
3         372  119
Crosstab table in percentage for Survived and Pclass:
Survived     0     1
Pclass
1         37.0  63.0
2         52.7  47.3
3         75.8  24.2
```



[28]:
```
# Survival with Sex
```

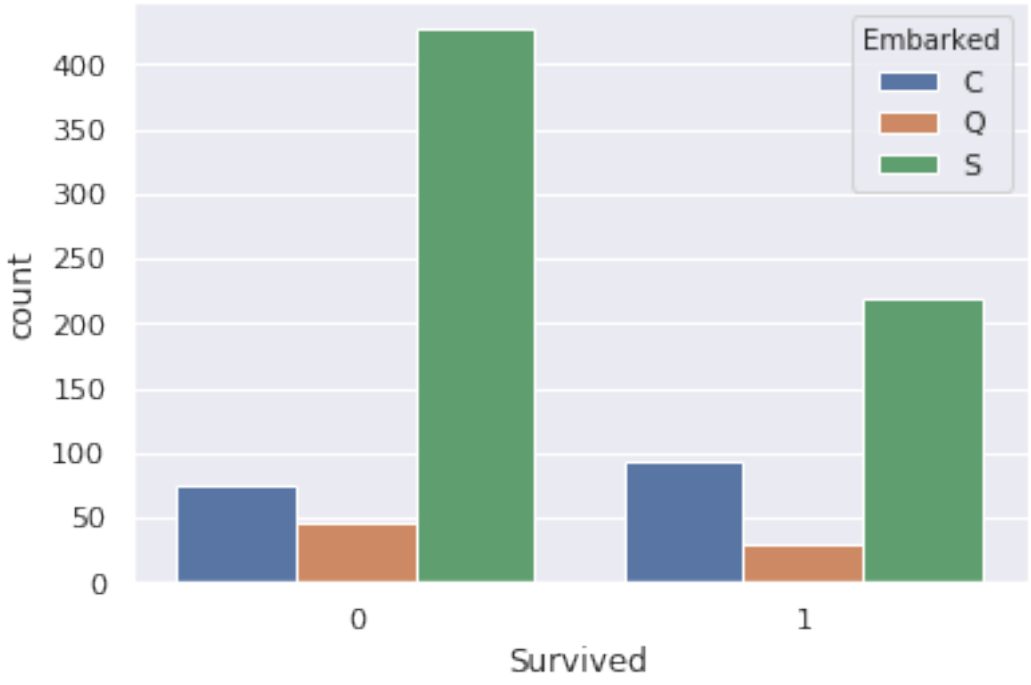
```
# Crosstab between Survived and Sex
print("Crosstab table for Survived and Sex:")
sns.countplot(df['Survived'], hue = df['Sex'])
print(pd.crosstab(df['Sex'], df['Survived']))

# percentage
print("Crosstab table in percentage for Survived and Sex:")
print(pd.crosstab(df['Sex'], df['Survived']).apply(lambda r: round((r/r.
 ↪sum())*100, 1), axis = 1))

# Conclusion: male died more compared to female.
```

```
Crosstab table for Survived and Sex:
Survived    0    1
Sex
female     81  233
male      468  109
Crosstab table in percentage for Survived and Sex:
Survived     0     1
Sex
female    25.8  74.2
male      81.1  18.9
```



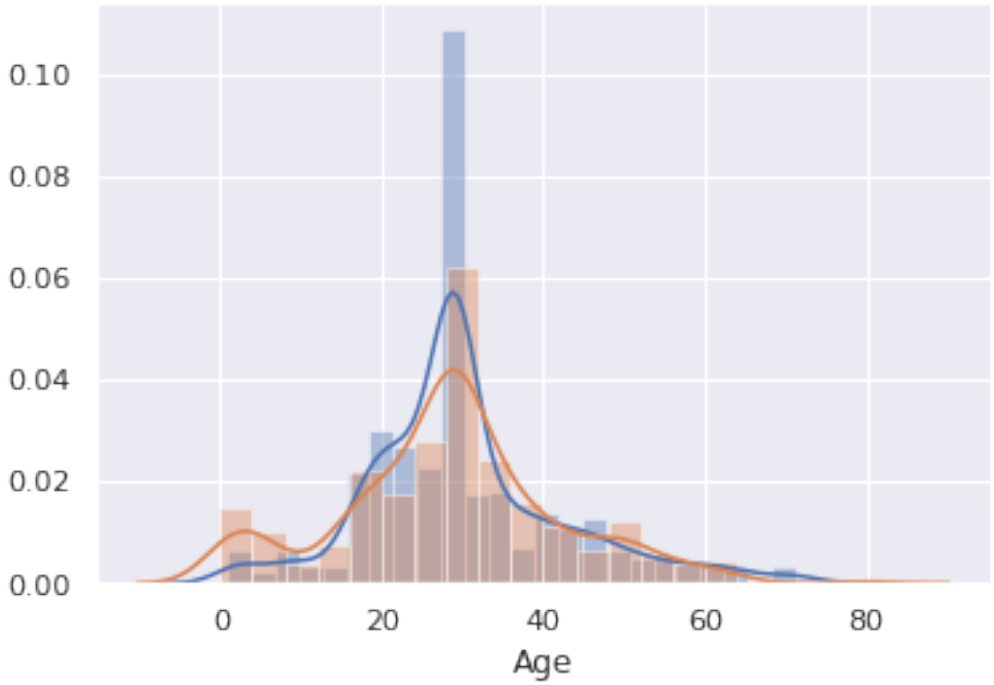
[29]:
```
# Survival with Embarked

# Crosstab between Survived and Embarked
print("Crosstab table for Survived and Embarked:")
sns.countplot(df['Survived'], hue = df['Embarked'])
print(pd.crosstab(df['Embarked'], df['Survived']))

# percentage
print("Crosstab table in percentage for Survived and Embarked:")
print(pd.crosstab(df['Embarked'], df['Survived']).apply(lambda r: round((r/r.
 ↪sum())*100, 1), axis = 1))

# Conclusion: survival rate for the people who were travelling to station C was␣
 ↪more than other two stations Q and S.
```

```
Crosstab table for Survived and Embarked:
Survived    0    1
Embarked
C          75   93
Q          47   30
S         427  219
Crosstab table in percentage for Survived and Embarked:
Survived     0     1
Embarked
C         44.6  55.4
Q         61.0  39.0
S         66.1  33.9
```



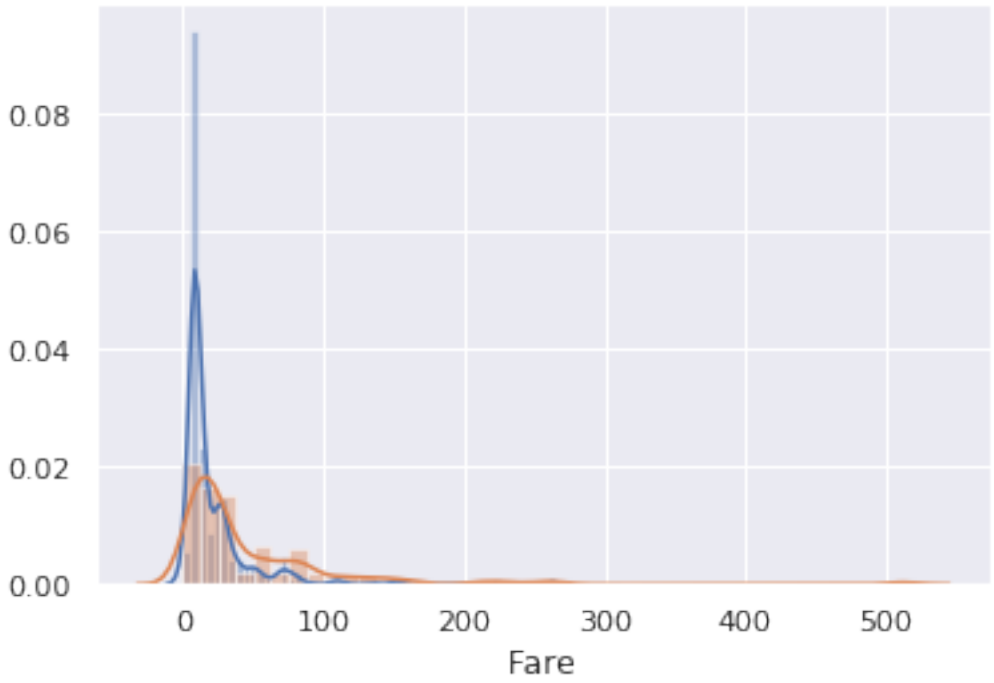
[30]:
```
# Survived with Age

sns.distplot(df[df['Survived'] == 0]['Age'])
sns.distplot(df[df['Survived'] == 1]['Age'])

# Conclusion: people of age around 0-15 had more chance to be saved.
# People of age around 15-35 had more chance to be dead.
# people of age around 35-40 had more probability to be saved.
# aged people whose age was greater than 60 was more likely to be dead
```

[30]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f98e7d398d0>
```



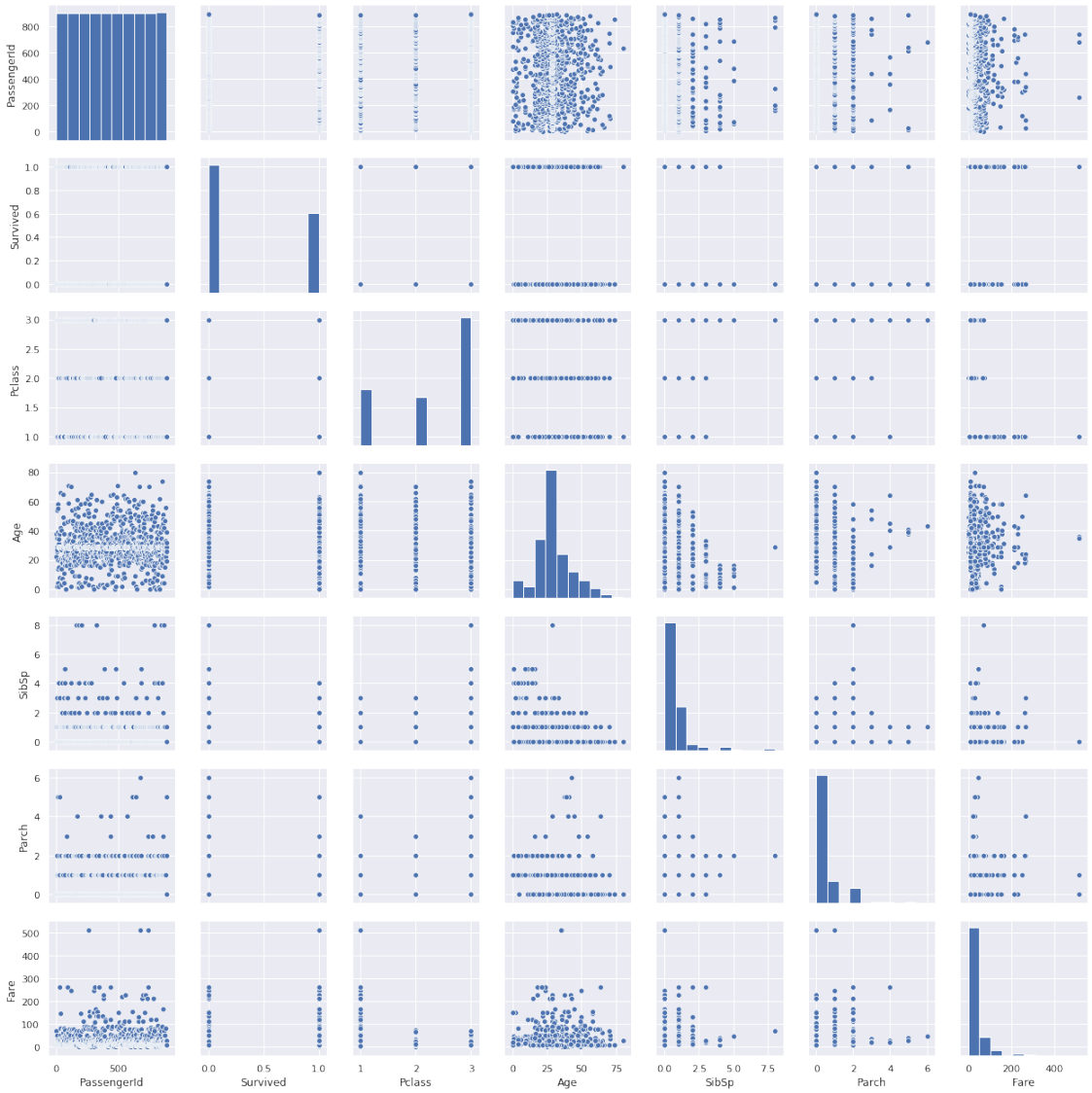
[31]:
```
# Survived with Fare

sns.distplot(df[df['Survived'] == 0]['Fare'])
sns.distplot(df[df['Survived'] == 1]['Fare'])

# Conclusion: More people with lower fare died. Survival rate has increased␣
 ↪with higher fare.
```

[31]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f98e5847210>
```



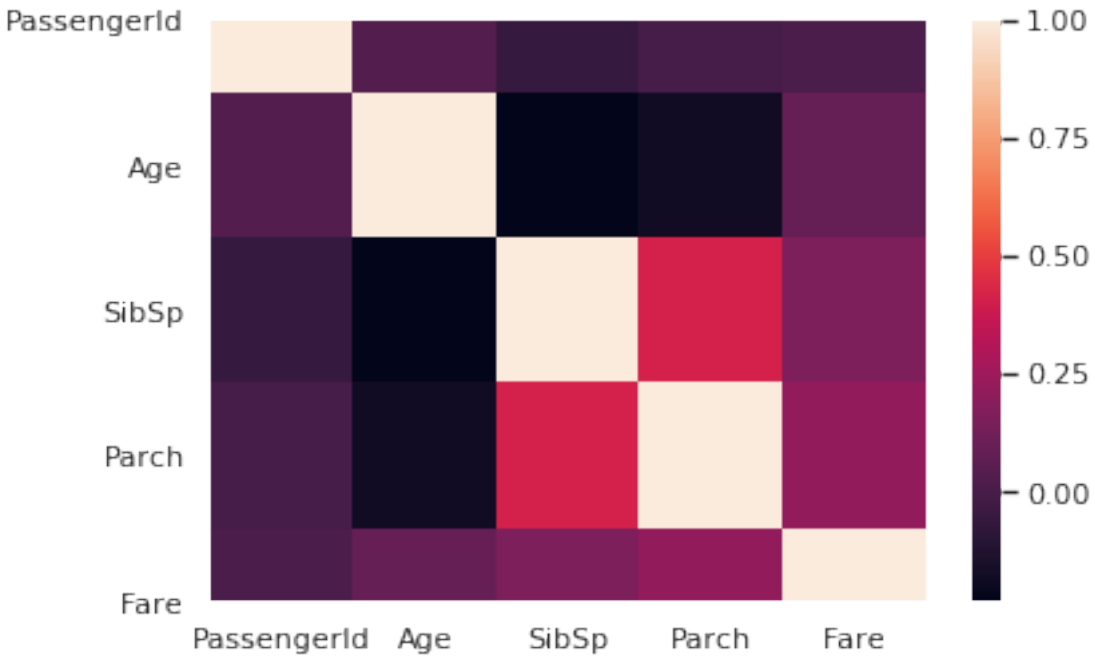
[32]:
```
sns.pairplot(df)
```

[32]: <seaborn.axisgrid.PairGrid at 0x7f98e56a0690>



[33]:
```
sns.heatmap(df.corr())

# Conclusion: SibSp and Parch have high correlation. Parch and Fare has high␣
 ↪correlation.
```

[33]: <matplotlib.axes._subplots.AxesSubplot at 0x7f98e3fe57d0>



[34]:
```
# A new column is created named family-size which will be the sum of SibSp and
 ↪Parch columns

df['family_size'] = df['Parch'] + df['SibSp']
```

[35]:
```
df.sample(5)
```

[35]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked	family_size
67	68	0	3	Crease, Mr. Ernest James	male	19	0	0	S.P. 3464	8.1583	S	0
452	453	0	1	Foreman, Mr. Benjamin Laventall	male	30	0	0	113051	27.7500	C	0
300	301	1	3	Kelly, Miss. Anna Katherine "Annie Kate"	female	29	0	0	9234	7.7500	Q	0
2	3	1	3	Heikkinen, Miss. Laina	female	26	0	0				
431	432	1	3	Thorneycroft, Mrs. Percival (Florence Kate White)	female	29	1	0				

```
2        0  STON/O2. 3101282   7.9250        S            0
431      0           376564  16.1000        S            1
```

[36]:
```
# A new feature has been engineered by the name of family type

def family_type(number):
    if number == 0:
        return "Alone"
    elif number > 0 and number <= 4:
        return "Medium"
    else:
        return "Large"

df['family_type'] = df['family_size'].apply(family_type)
```

[37]:
```
df.sample(5)
```

[37]:
```
     PassengerId Survived Pclass  \
608          609        1      2
530          531        1      2
126          127        0      3
638          639        0      3
646          647        0      3

                                                 Name     Sex  Age  SibSp  \
608  Laroche, Mrs. Joseph (Juliette Marie Louise La…  female   22      1
530                          Quick, Miss. Phyllis May  female    2      1
126                               McMahon, Mr. Martin    male   29      0
638            Panula, Mrs. Juha (Maria Emilia Ojala)  female   41      0
646                                Cor, Mr. Liudevit    male   19      0

     Parch         Ticket     Fare Embarked  family_size family_type
608      2  SC/Paris 2123  41.5792        C            3      Medium
530      1          26360  26.0000        S            2      Medium
126      0         370372   7.7500        Q            0       Alone
638      5        3101295  39.6875        S            5       Large
646      0         349231   7.8958        S            0       Alone
```

[38]:
```
# Dropping SibSp, Parch and family_size

df.drop(columns = ['SibSp', 'Parch', 'family_size'], inplace = True)
```

[39]:
```
df.sample(5)
```

[39]:
```
     PassengerId Survived Pclass                                 Name  \
639          640        0      3          Thorneycroft, Mr. Percival
597          598        0      3                 Johnson, Mr. Alfred
```

```
774          775         1       2  Hocking, Mrs. Elizabeth (Eliza Needs)
495          496         0       3                  Youseff, Mr. Gerious
386          387         0       3        Goodwin, Master. Sidney Leonard

        Sex  Age   Ticket     Fare Embarked family_type
639    male   29   376564  16.1000        S      Medium
597    male   49     LINE   0.0000        S       Alone
774  female   54    29105  23.0000        S      Medium
495    male   29     2627  14.4583        C       Alone
386    male    1  CA 2144  46.9000        S       Large
```

[40]:
```
# Analyzing family_type with Survived

print("Crosstab of family_type and Survived:")
print(pd.crosstab(df['family_type'], df['Survived']))

print("Crosstab with percentage of family_type and Survived:")
print(pd.crosstab(df['family_type'], df['Survived']).apply(lambda r: round((r/r.
 ↪sum())*100, 1), axis = 1))

# Conclusion: Death rate is highter for them who travelled with large family.
# Death rate is also high for them who travelled alone.
# Survival rate is highter for them who travelled with medium family.
```

```
Crosstab of family_type and Survived:
Survived       0    1
family_type
Alone        374  163
Large         40    7
Medium       135  172
Crosstab with percentage of family_type and Survived:
Survived        0     1
family_type
Alone        69.6  30.4
Large        85.1  14.9
Medium       44.0  56.0
```

[41]:
```
# Handling outliers in Age. Age almost follows normal distribution

df = df[df['Age'] < (df['Age']. mean() + 3 * df['Age'].std())]
df.shape
```

[41]:
```
(884, 10)
```

[42]:
```
# Handling outliers for Fare column. Fare does not follow normal distribution

# Finding quartiles
```

```
Q1 = np.percentile(df['Fare'],25)
Q3 = np.percentile(df['Fare'],75)

outlier_low =  Q1 - 1.5 * (Q3 - Q1)
outlier_high =  Q3 + 1.5 * (Q3 - Q1)

df= df[(df['Fare'] > outlier_low) & (df['Fare'] < outlier_high)]
df.shape
```

[42]:
```
(769, 10)
```

[43]:
```
# One hot encoding
# Columns to be transformed are Pclass, Sex, Embarked and family_type

pd.get_dummies(data = df, columns = ['Pclass', 'Sex', 'Embarked',␣
↪'family_type'], drop_first = True)
```

[43]:
```
     PassengerId Survived                                          Name  Age  \
0              1        0                       Braund, Mr. Owen Harris   22
2              3        1                        Heikkinen, Miss. Laina   26
3              4        1  Futrelle, Mrs. Jacques Heath (Lily May Peel)   35
4              5        0                      Allen, Mr. William Henry   35
5              6        0                              Moran, Mr. James   29
..           …      …                                             …  …
886          887        0                        Montvila, Rev. Juozas   27
887          888        1                 Graham, Miss. Margaret Edith   19
888          889        0     Johnston, Miss. Catherine Helen "Carrie"   29
889          890        1                        Behr, Mr. Karl Howell   26
890          891        0                          Dooley, Mr. Patrick   32

                Ticket     Fare  Pclass_2  Pclass_3  Sex_male  Embarked_Q  \
0            A/5 21171   7.2500         0         1         1           0
2    STON/O2. 3101282   7.9250         0         1         0           0
3               113803  53.1000         0         0         0           0
4               373450   8.0500         0         1         1           0
5               330877   8.4583         0         1         1           1
..                 …      …       …       …       …         …
886             211536  13.0000         1         0         1           0
887             112053  30.0000         0         0         0           0
888         W./C. 6607  23.4500         0         1         0           0
889             111369  30.0000         0         0         1           0
890             370376   7.7500         0         1         1           1

     Embarked_S  family_type_Large  family_type_Medium
0             1                  0                   1
2             1                  0                   0
3             1                  0                   1
```

```
4              1                   0                   0
5              0                   0                   0
..           ...                 ...                 ...
886            1                   0                   0
887            1                   0                   0
888            1                   0                   1
889            0                   0                   0
890            0                   0                   0

[769 rows x 13 columns]
```

[44]:
```
plt.figure(figsize =(15, 6))
sns.heatmap(df.corr(), cmap = 'summer')
```

[44]: <matplotlib.axes._subplots.AxesSubplot at 0x7f98e3fe54d0>