untitled8

May 7, 2024

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[3]:
```
df = sns.load_dataset('titanic')
```

[4]:
```
df.head()
```

[4]:
```
   survived  pclass     sex   age  sibsp  parch     fare embarked  class  \
0         0       3    male  22.0      1      0   7.2500        S  Third
1         1       1  female  38.0      1      0  71.2833        C  First
2         1       3  female  26.0      0      0   7.9250        S  Third
3         1       1  female  35.0      1      0  53.1000        S  First
4         0       3    male  35.0      0      0   8.0500        S  Third

     who  adult_male deck  embark_town alive  alone
0    man        True  NaN  Southampton    no  False
1  woman       False    C    Cherbourg   yes  False
2  woman       False  NaN  Southampton   yes   True
3  woman       False    C  Southampton   yes  False
4    man        True  NaN  Southampton    no   True
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
```

```
 8   class        891 non-null    category
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

[41]:
```
df.isnull().sum()
```

[41]:
```
survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64
```

[43]:
```
print(df['age'].mode())
print(df['embarked'].mode())
print(df['embark_town'].mode())
```

```
0    24.0
Name: age, dtype: float64
0    S
Name: embarked, dtype: object
0    Southampton
Name: embark_town, dtype: object
```

[44]:
```
df['age'].fillna(value=24,inplace=True)
df['embarked'].fillna(value='S',inplace=True)
df['embark_town'].fillna(value='Southampton',inplace=True)
```

```
C:\Users\PUSHKAR\AppData\Local\Temp\ipykernel_14332\3508236087.py:1:
FutureWarning: A value is trying to be set on a copy of a DataFrame or Series
through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
```

```
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df['age'].fillna(value=24,inplace=True)
C:\Users\PUSHKAR\AppData\Local\Temp\ipykernel_14332\3508236087.py:2:
FutureWarning: A value is trying to be set on a copy of a DataFrame or Series
through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df['embarked'].fillna(value='S',inplace=True)
C:\Users\PUSHKAR\AppData\Local\Temp\ipykernel_14332\3508236087.py:3:
FutureWarning: A value is trying to be set on a copy of a DataFrame or Series
through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df['embark_town'].fillna(value='Southampton',inplace=True)
```

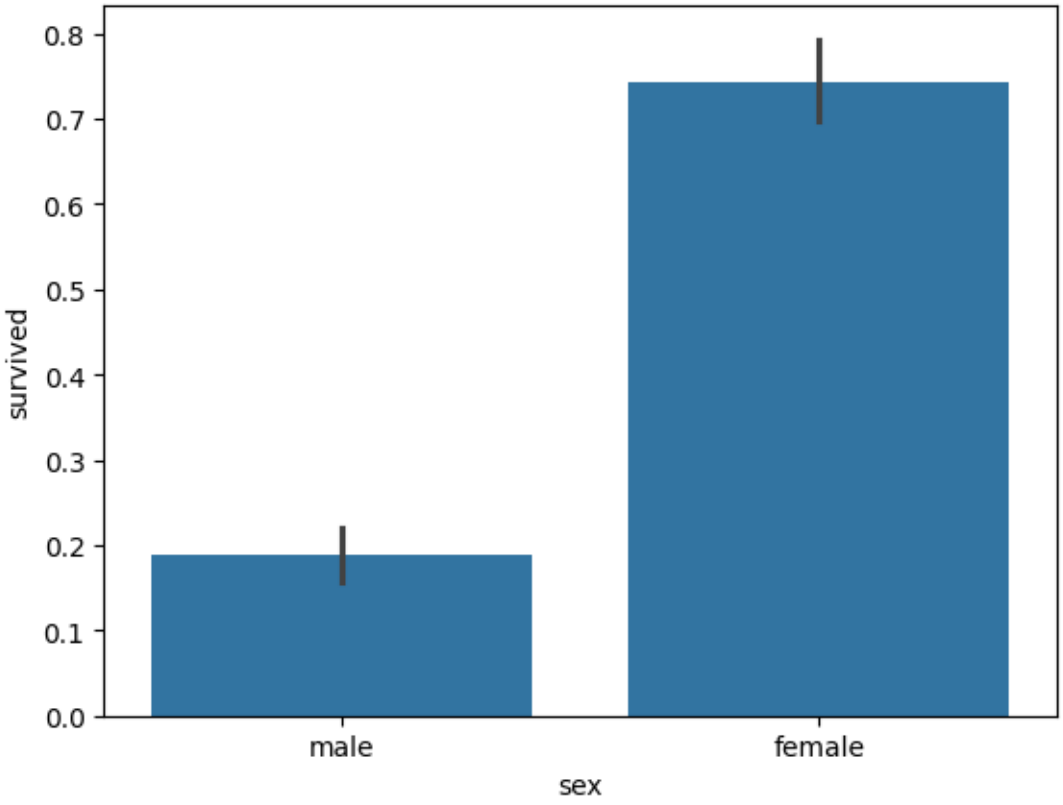
[48]:
```
df=df.drop(['deck'],axis=1,inplace=True)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[48], line 1
----> 1 df=df.drop(['deck'],axis=1,inplace=True)

AttributeError: 'NoneType' object has no attribute 'drop'
```

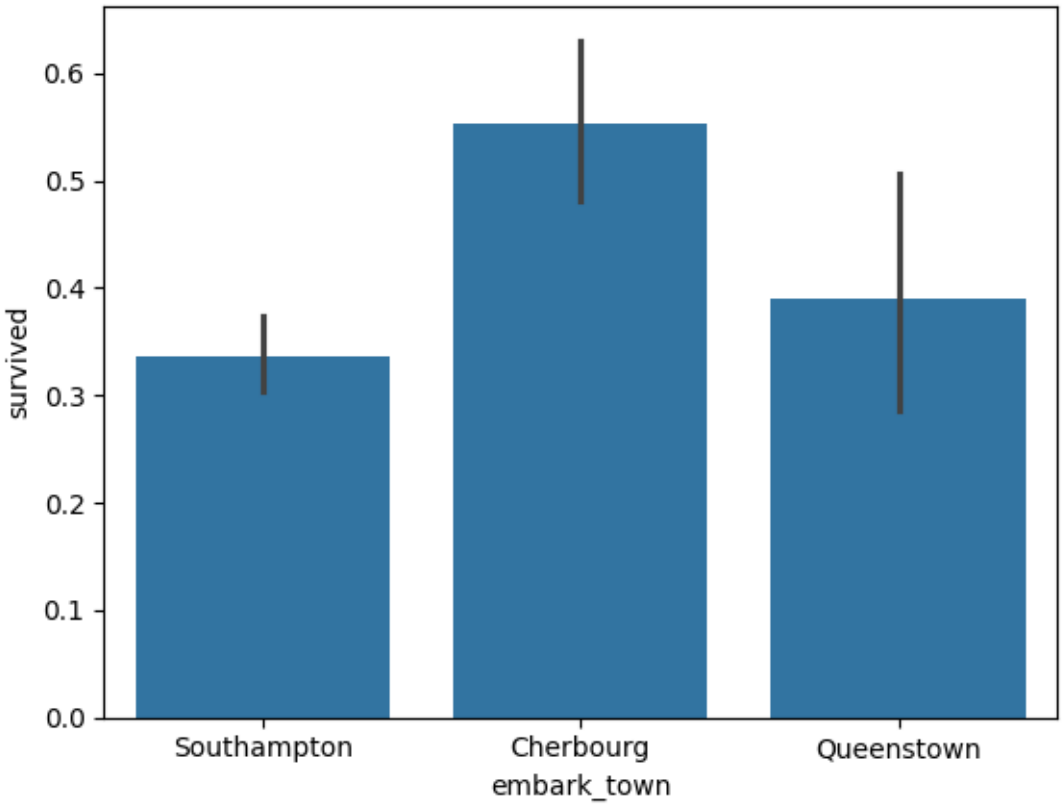
[32]:
```
sns.barplot(x='sex',y='survived',data=df)
```

[32]: <Axes: xlabel='sex', ylabel='survived'>



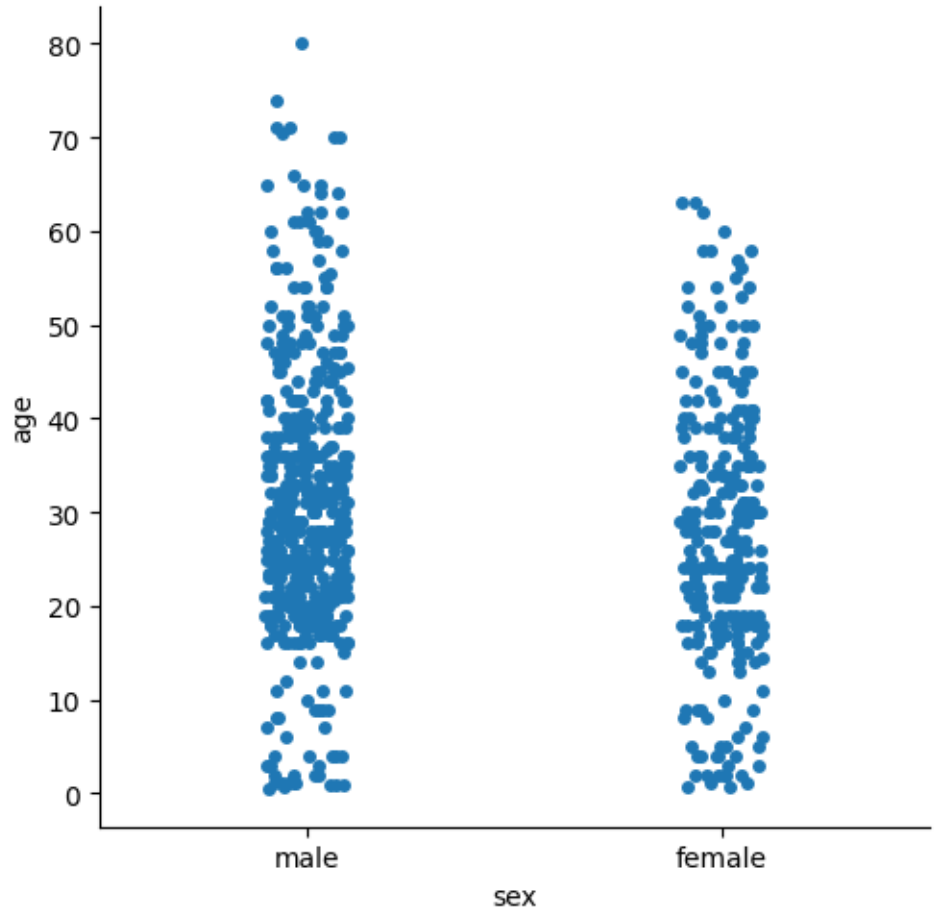
[33]:
```
sns.barplot(x='embark_town',y='survived',data=df)
```

[33]: <Axes: xlabel='embark_town', ylabel='survived'>



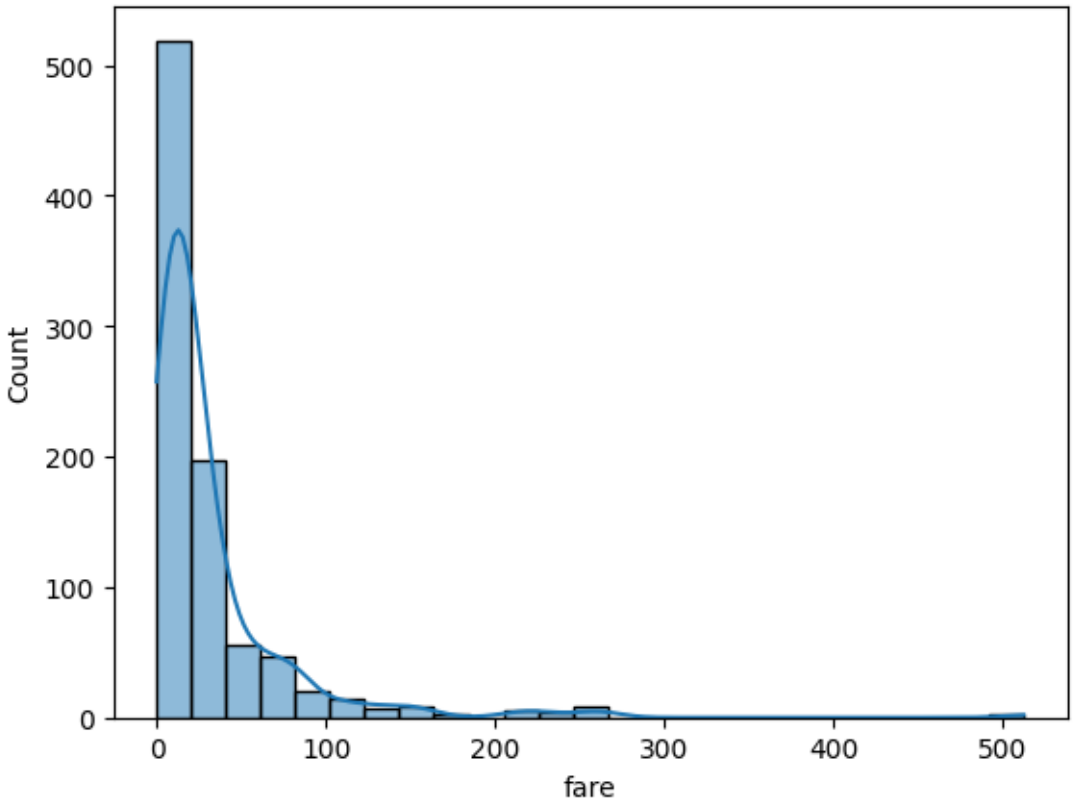
[35]:
```
sns.catplot(x='sex',y='age',data=df)
```

[35]: <seaborn.axisgrid.FacetGrid at 0x25702498ce0>



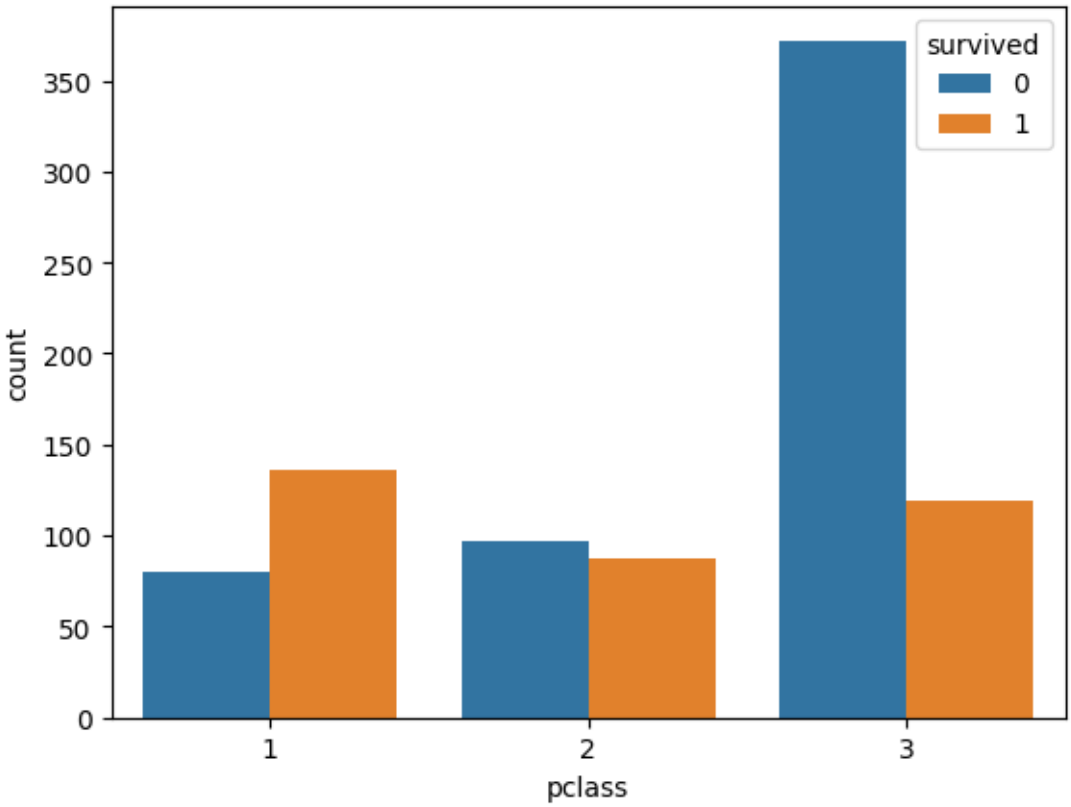
[37]:
```
sns.histplot(data=df['fare'],bins=25,kde=True)
```

[37]:
```
<Axes: xlabel='fare', ylabel='Count'>
```



[40]:
```
sns.countplot(x='pclass',hue='survived',data=df)
```

[40]: <Axes: xlabel='pclass', ylabel='count'>


survived
0
1
350
300
250
200
150
100
50
0
count
1
2
3
pclass

[ ]: