titanic

July 6, 2024

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

titanic_data = pd.read_csv("Titanic.csv")
titanic_data.head(10)
```

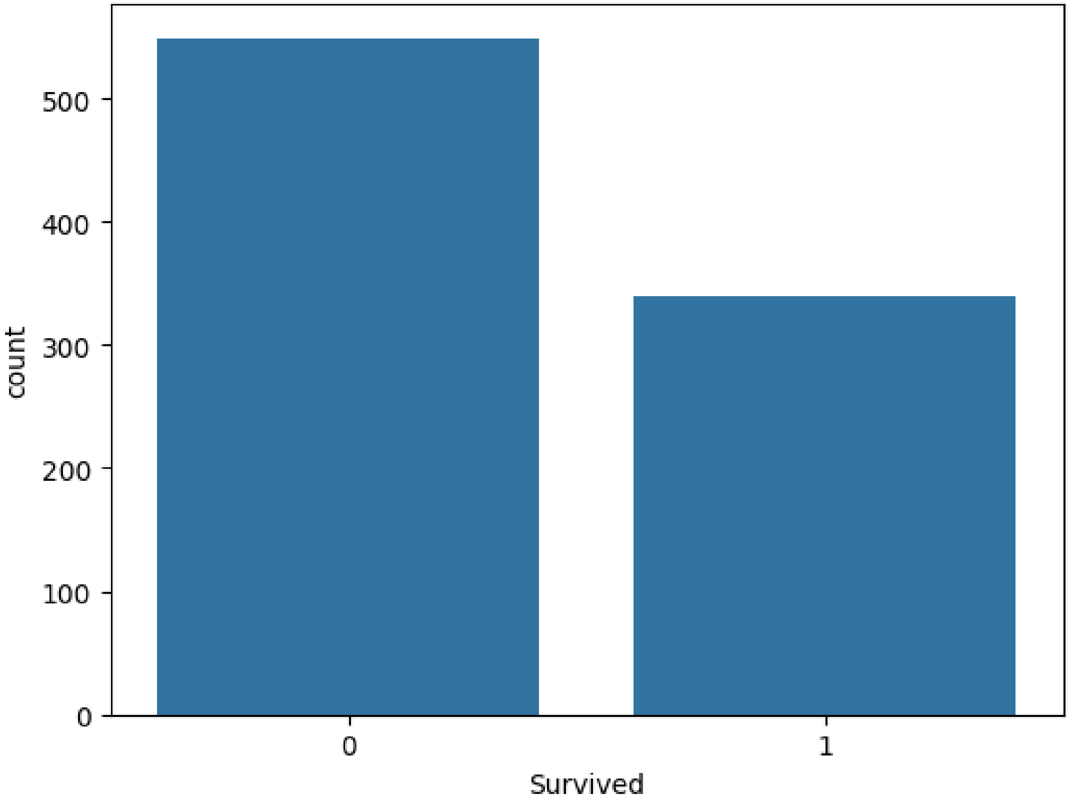
[1]:
```
   PassengerId  Survived  Pclass     Sex   Age  SibSp  Parch     Fare  \
0            1         0       3    Male  22.0      1      0   7.2500
1            2         1       1  female  38.0      1      0  71.2833
2            3         1       3  female  26.0      0      0   7.9250
3            4         1       1  female  35.0      1      0  53.1000
4            5         0       3    Male  35.0      0      0   8.0500
5            6         0       3    Male  60.0      0      0   8.4583
6            7         0       1    Male  54.0      0      0  51.8625
7            8         0       3    Male   2.0      3      1  21.0750
8            9         1       3  female  27.0      0      2  11.1333
9           10         1       2  female  14.0      1      0  30.0708

   Embarked
0         3
1         1
2         3
3         3
4         3
5         2
6         3
7         3
8         3
9         1
```

[2]:
```
print('# of passengers  in original data:' +str(len(titanic_data.index)))
```

```
# of passengers  in original data:889
```

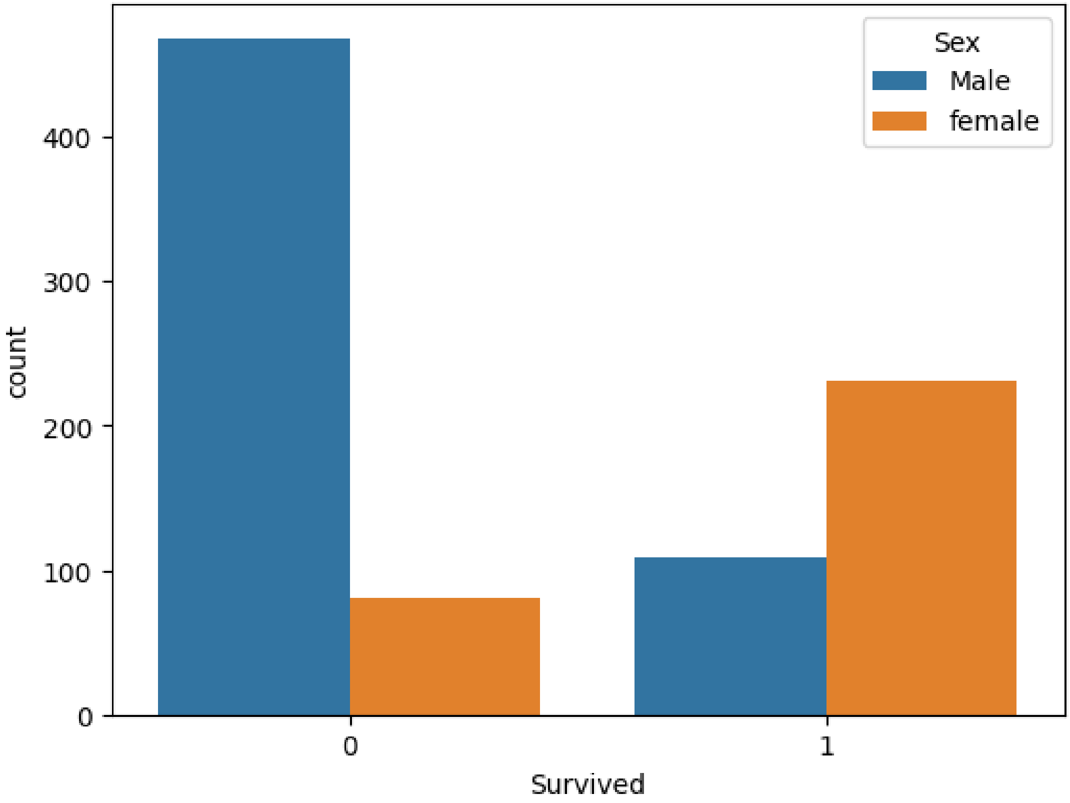
[3]:
```
sns.countplot(x="Survived", data=titanic_data)
```

[3]:
```
<Axes: xlabel='Survived', ylabel='count'>
```



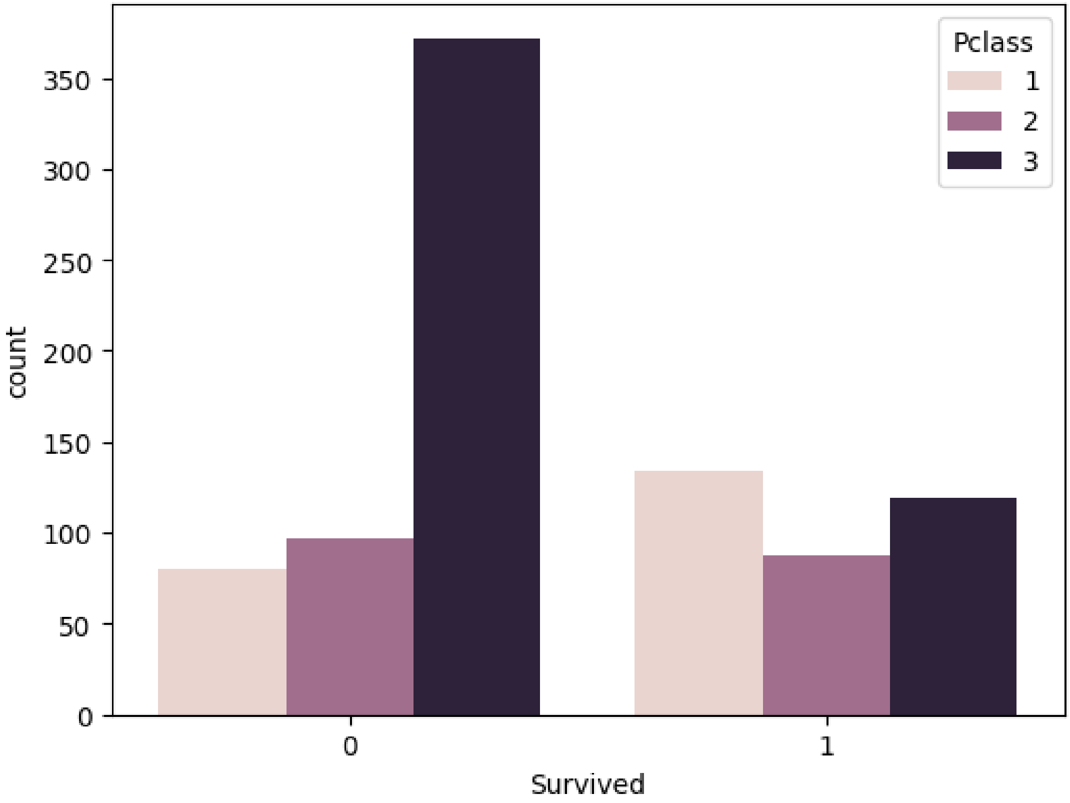
[4]:
```
sns.countplot(x="Survived",hue="Sex", data=titanic_data)
```

[4]:
```
<Axes: xlabel='Survived', ylabel='count'>
```



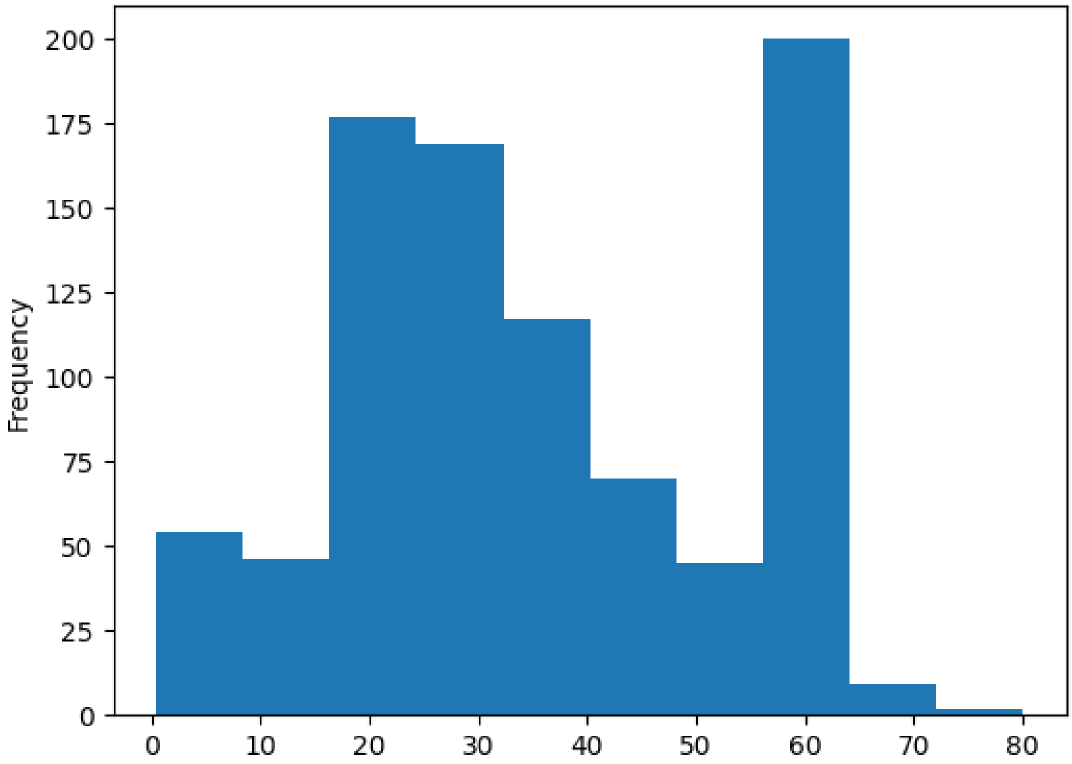
```
[5]: sns.countplot(x="Survived",hue="Pclass",data=titanic_data)
```

```
[5]: <Axes: xlabel='Survived', ylabel='count'>
```



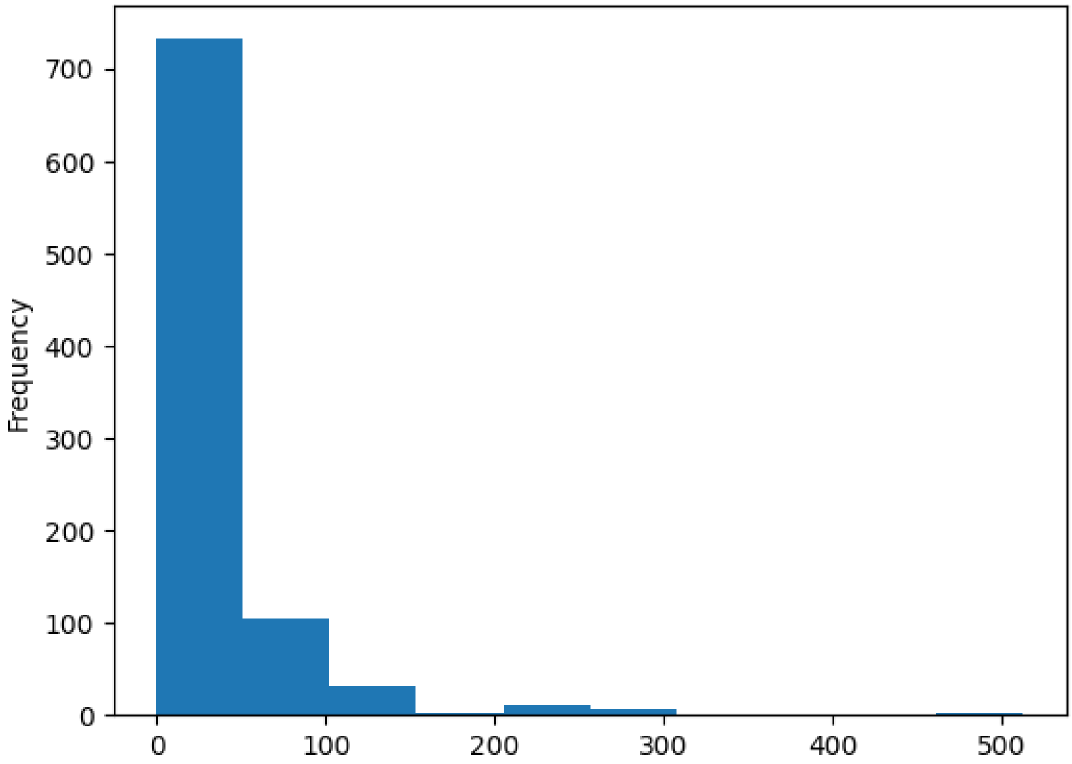
[6]:
```
titanic_data['Age'].plot.hist()
```

[6]:
```
<Axes: ylabel='Frequency'>
```



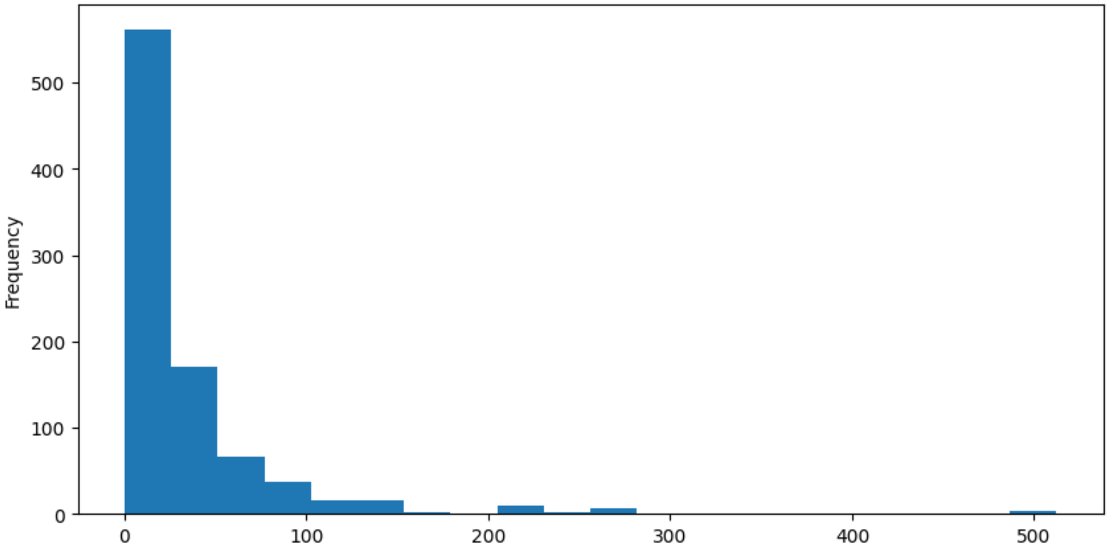
```
[7]: titanic_data['Fare'].plot.hist()
```

```
[7]: <Axes: ylabel='Frequency'>
```



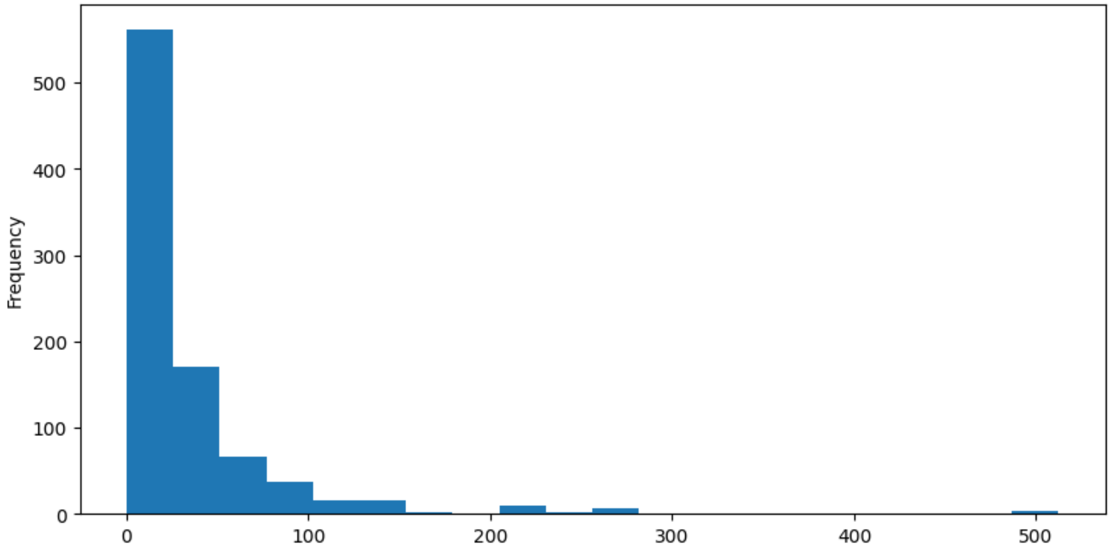
```
[55]: titanic_data['Fare'].plot.hist(bins=20, figsize=(10,5))
```

```
[55]: <Axes: ylabel='Frequency'>
```



```
[10]: titanic_data['Fare'].plot.hist(bins=20, figsize=(10,5))
```

```
[10]: <Axes: ylabel='Frequency'>
```

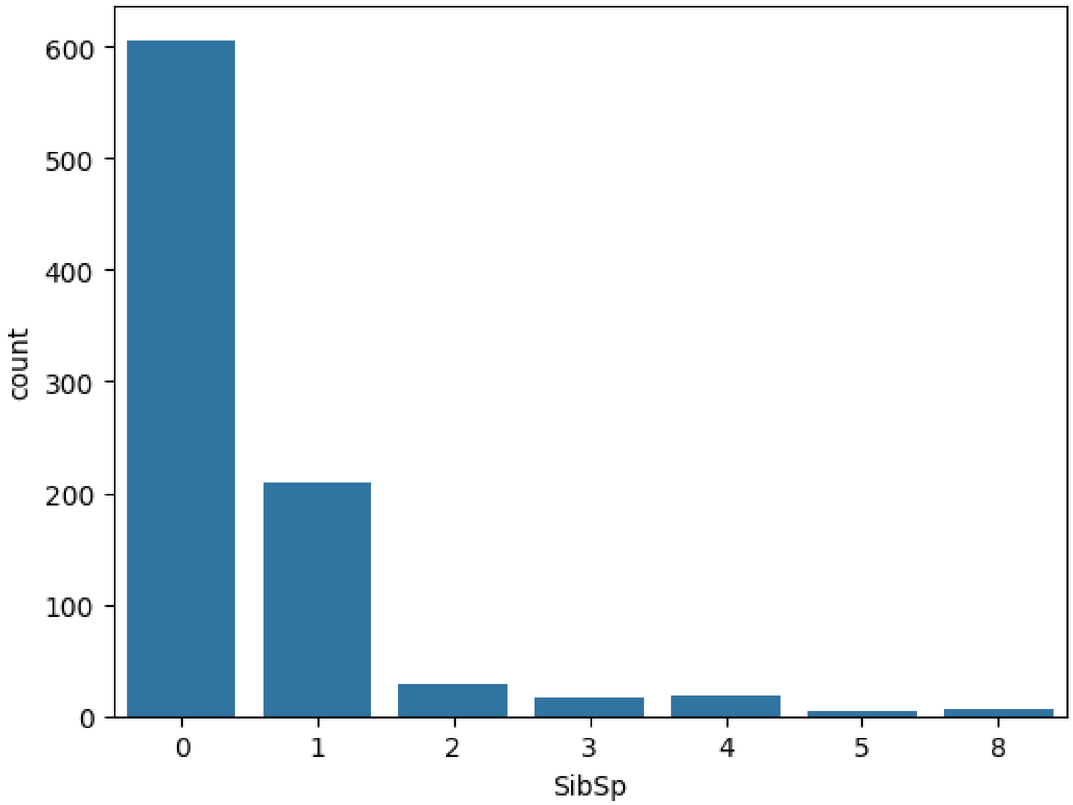


```
[11]: titanic_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 889 entries, 0 to 888
Data columns (total 9 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  889 non-null    int64  
 1   Survived     889 non-null    int64  
 2   Pclass       889 non-null    int64  
 3   Sex          889 non-null    object 
 4   Age          889 non-null    float64
 5   SibSp        889 non-null    int64  
 6   Parch        889 non-null    int64  
 7   Fare         889 non-null    float64
 8   Embarked     889 non-null    int64  
dtypes: float64(2), int64(6), object(1)
memory usage: 62.6+ KB
```

```
[12]: sns.countplot(x='SibSp', data=titanic_data)
```

```
[12]: <Axes: xlabel='SibSp', ylabel='count'>
```



[13]:
```
titanic_data.isnull()
```

[13]:
```
     PassengerId  Survived  Pclass    Sex    Age  SibSp  Parch    Fare  \
0          False     False   False  False  False  False  False   False
1          False     False   False  False  False  False  False   False
2          False     False   False  False  False  False  False   False
3          False     False   False  False  False  False  False   False
4          False     False   False  False  False  False  False   False
..             …         …      …     …     …     …     …     …
884        False     False   False  False  False  False  False   False
885        False     False   False  False  False  False  False   False
886        False     False   False  False  False  False  False   False
887        False     False   False  False  False  False  False   False
888        False     False   False  False  False  False  False   False

     Embarked
0       False
1       False
2       False
3       False
4       False
```

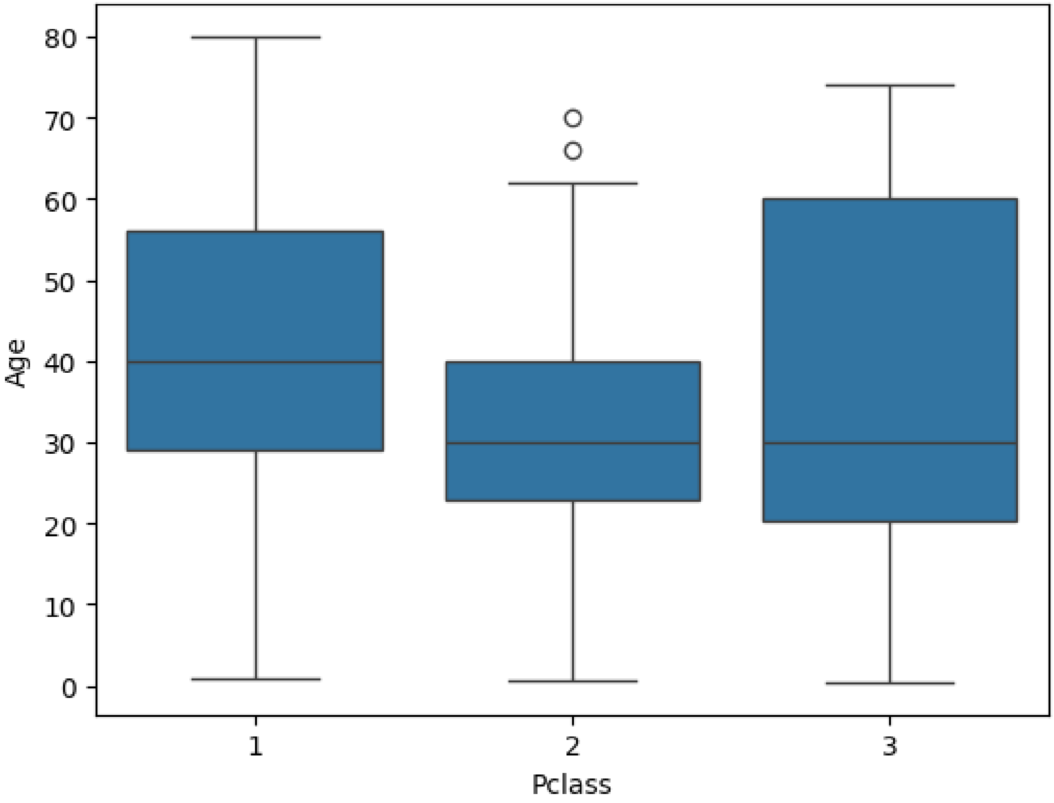
```
..        …
884     False
885     False
886     False
887     False
888     False

[889 rows x 9 columns]
```

[14]:
```
titanic_data.isnull().sum()
```

[14]:
```
PassengerId    0
Survived       0
Pclass         0
Sex            0
Age            0
SibSp          0
Parch          0
Fare           0
Embarked       0
dtype: int64
```

[15]:
```
sns.boxplot(x='Pclass', y='Age', data=titanic_data)
```

[15]:
```
<Axes: xlabel='Pclass', ylabel='Age'>
```



[16]:
```
titanic_data.head(5)
```

[16]:
```
   PassengerId  Survived  Pclass     Sex   Age  SibSp  Parch     Fare  \
0            1         0       3    Male  22.0      1      0   7.2500
1            2         1       1  female  38.0      1      0  71.2833
2            3         1       3  female  26.0      0      0   7.9250
3            4         1       1  female  35.0      1      0  53.1000
4            5         0       3    Male  35.0      0      0   8.0500

   Embarked
0         3
1         1
2         3
3         3
4         3
```

[17]:
```
titanic_data.dropna(inplace=True)
```

[18]:
```
titanic_data.isnull().sum()
```

[18]:
```
PassengerId    0
Survived       0
Pclass         0
Sex            0
Age            0
SibSp          0
Parch          0
Fare           0
Embarked       0
dtype: int64
```

[19]:
```
titanic_data.head(2)
```

[19]:
```
   PassengerId  Survived  Pclass     Sex   Age  SibSp  Parch     Fare  \
0            1         0       3    Male  22.0      1      0   7.2500
1            2         1       1  female  38.0      1      0  71.2833

   Embarked
0         3
1         1
```

[20]:
```
pd.get_dummies(titanic_data['Sex'])
```

[20]:
```
      Male  female
0     True   False
1    False    True
2    False    True
3    False    True
4     True   False
..     …       …
884   True   False
885  False    True
886  False    True
887   True   False
888   True   False

[889 rows x 2 columns]
```

[22]:
```
sex = pd.get_dummies(titanic_data['Sex'], drop_first=True)
sex.head(5)
```

[22]:
```
   female
0   False
1    True
2    True
3    True
4   False
```

[23]:

```
embark=pd.get_dummies(titanic_data["Embarked"],drop_first=True)
embark.head(5)
```

[23]:

```
       2      3
0  False   True
1  False  False
2  False   True
3  False   True
4  False   True
```

[24]:

```
Pcl=pd.get_dummies(titanic_data['Pclass'],drop_first=True)
Pcl.head(5)
```

[24]:

```
       2      3
0  False   True
1  False  False
2  False   True
3  False  False
4  False   True
```

[25]:

```
titanic_data=pd.concat([titanic_data,sex,embark,Pcl],axis=1)
```

[26]:

```
titanic_data.head(5)
```

[26]:

```
   PassengerId  Survived  Pclass     Sex   Age  SibSp  Parch     Fare  \
0            1         0       3    Male  22.0      1      0   7.2500
1            2         1       1  female  38.0      1      0  71.2833
2            3         1       3  female  26.0      0      0   7.9250
3            4         1       1  female  35.0      1      0  53.1000
4            5         0       3    Male  35.0      0      0   8.0500

   Embarked  female      2      3      2      3
0         3   False  False   True  False   True
1         1    True  False  False  False  False
2         3    True  False   True  False   True
3         3    True  False   True  False  False
4         3   False  False   True  False   True
```

[ ]:

```
titanic_data.drop(['Sex','Embarked','PassengerId'],axis=1,inplace=True)
```

[28]:

```
titanic_data.head()
```

[28]:

```
   PassengerId  Survived  Pclass     Sex   Age  SibSp  Parch     Fare  \
0            1         0       3    Male  22.0      1      0   7.2500
1            2         1       1  female  38.0      1      0  71.2833
2            3         1       3  female  26.0      0      0   7.9250
3            4         1       1  female  35.0      1      0  53.1000
```

```
4            5          0        3     Male  35.0      0      0   8.0500

   Embarked  female      2      3      2      3
0         3   False  False   True  False   True
1         1    True  False  False  False  False
2         3    True  False   True  False   True
3         3    True  False   True  False  False
4         3   False  False   True  False   True
```

[29]:
```
titanic_data.drop(['Pclass'],axis=1,inplace=True)
```

[30]:
```
titanic_data.head()
```

[30]:
```
   PassengerId  Survived     Sex   Age  SibSp  Parch     Fare  Embarked  \
0            1         0    Male  22.0      1      0   7.2500         3
1            2         1  female  38.0      1      0  71.2833         1
2            3         1  female  26.0      0      0   7.9250         3
3            4         1  female  35.0      1      0  53.1000         3
4            5         0    Male  35.0      0      0   8.0500         3

   female      2      3      2      3
0   False  False   True  False   True
1    True  False  False  False  False
2    True  False   True  False   True
3    True  False   True  False  False
4   False  False   True  False   True
```

[32]:
```
X = titanic_data.drop("Survived", axis=1)
Y = titanic_data["Survived"]
```

[34]:
```
from sklearn.model_selection import train_test_split
```

[39]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3,␣
 ↪random_state=1)
```

[40]:
```
from sklearn.linear_model import LogisticRegression
```

[41]:
```
logmodel=LogisticRegression()
```

[45]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import OneHotEncoder

# Assuming 'X' is your pandas DataFrame and contains categorical columns
ohe = OneHotEncoder(handle_unknown='ignore')  # Create OneHotEncoder instance
X_encoded = ohe.fit_transform(X.select_dtypes(include=['object']))  # Encode␣
 ↪categorical columns
```

```
# Convert the encoded data back to a DataFrame for easier handling
X_encoded_df = pd.DataFrame.sparse.from_spmatrix(X_encoded)

# Get feature names for one-hot encoded columns
# Assuming ohe.get_feature_names_out() is available in your sklearn version
feature_names = ohe.get_feature_names_out(X.select_dtypes(include=['object']).
  ↪columns)
X_encoded_df.columns = feature_names  # Assign feature names to encoded columns

# Concatenate the encoded categorical features with numerical features if any
X_final = pd.concat([X.select_dtypes(exclude=['object']), X_encoded_df], axis=1)

# Now proceed with splitting and model fitting
X_train, X_test, Y_train, Y_test = train_test_split(X_final, Y, test_size=0.3,␣
  ↪random_state=1)

# Ensure all column names are strings
X_train.columns = X_train.columns.astype(str)

logmodel = LogisticRegression()
logmodel.fit(X_train, Y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:768:
UserWarning: pandas.DataFrame with sparse columns found.It will be converted to
a dense numpy array.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[45]:

```
LogisticRegression()
```

[46]:

```
predictions = logmodel.predict(X_test)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:768:
UserWarning: pandas.DataFrame with sparse columns found.It will be converted to
a dense numpy array.
  warnings.warn(
[47]: from sklearn.metrics import classification_report
```

```
[48]: classification_report(Y_test,predictions)
```

```
[48]: '              precision    recall  f1-score   support\n\n           0
      0.86      0.87      0.86       166\n           1       0.78      0.76      0.77
      101\n\n    accuracy                           0.83       267\n   macro avg
      0.82      0.81      0.82       267\nweighted avg       0.83      0.83      0.83
      267\n'
```

```
[49]: from sklearn.metrics import confusion_matrix
```

```
[50]: c = confusion_matrix(Y_test, predictions)
```

```
[52]: confusion_matrix(Y_test, predictions)
```

```
[52]: array([[144,  22],
             [ 24,  77]])
```

```
[53]: from sklearn.metrics import accuracy_score
```

```
[54]: accuracy_score(Y_test, predictions)
```

```
[54]: 0.8277153558052435
```

```
[ ]:
```

```
[ ]:
```