Titanic

April 5, 2024

[4]:
```
## Import Modules
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

[7]:
```
### Loading the Data
train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')
train.head()
```

[7]:
```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         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

   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
```

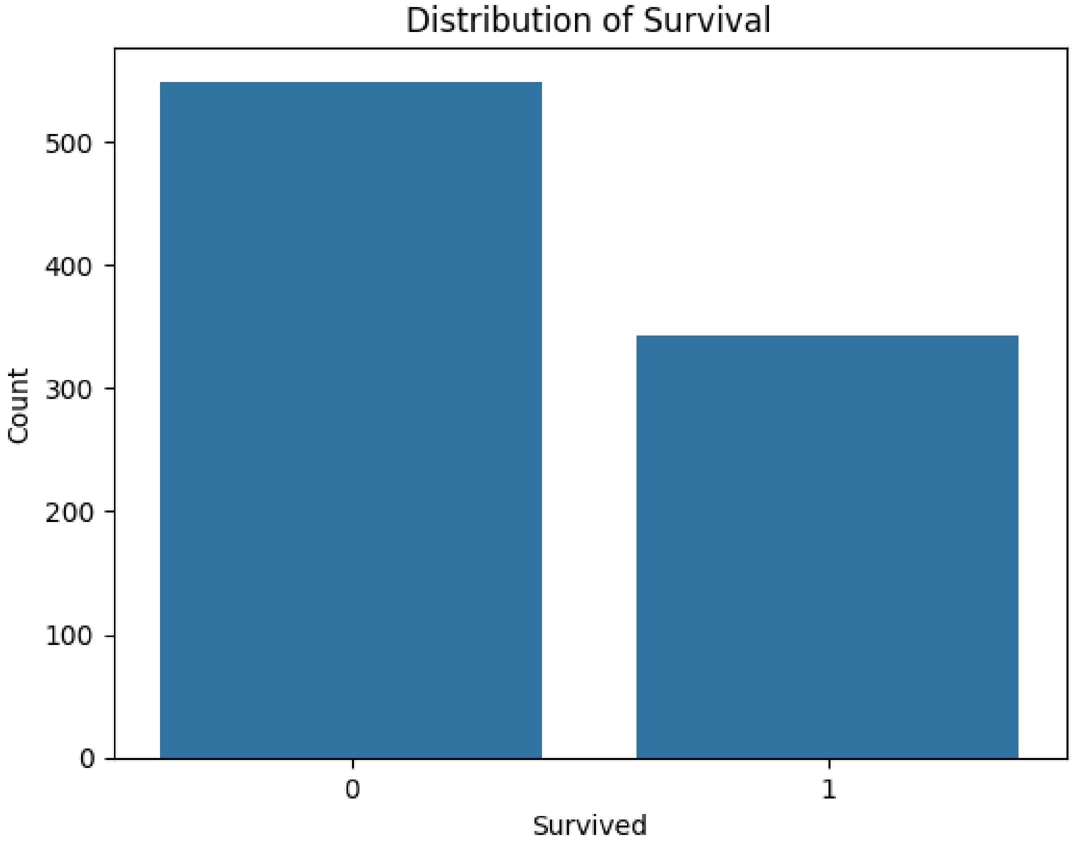
[10]:
```
## Statistical Info
train.describe()
```

```
[10]:        PassengerId    Survived      Pclass         Age       SibSp  \
      count   891.000000  891.000000  891.000000  714.000000  891.000000
      mean    446.000000    0.383838    2.308642   29.699118    0.523008
      std     257.353842    0.486592    0.836071   14.526497    1.102743
      min       1.000000    0.000000    1.000000    0.420000    0.000000
      25%     223.500000    0.000000    2.000000   20.125000    0.000000
      50%     446.000000    0.000000    3.000000   28.000000    0.000000
      75%     668.500000    1.000000    3.000000   38.000000    1.000000
      max     891.000000    1.000000    3.000000   80.000000    8.000000

                  Parch        Fare
      count  891.000000  891.000000
      mean     0.381594   32.204208
      std      0.806057   49.693429
      min      0.000000    0.000000
      25%      0.000000    7.910400
      50%      0.000000   14.454200
      75%      0.000000   31.000000
      max      6.000000  512.329200
```

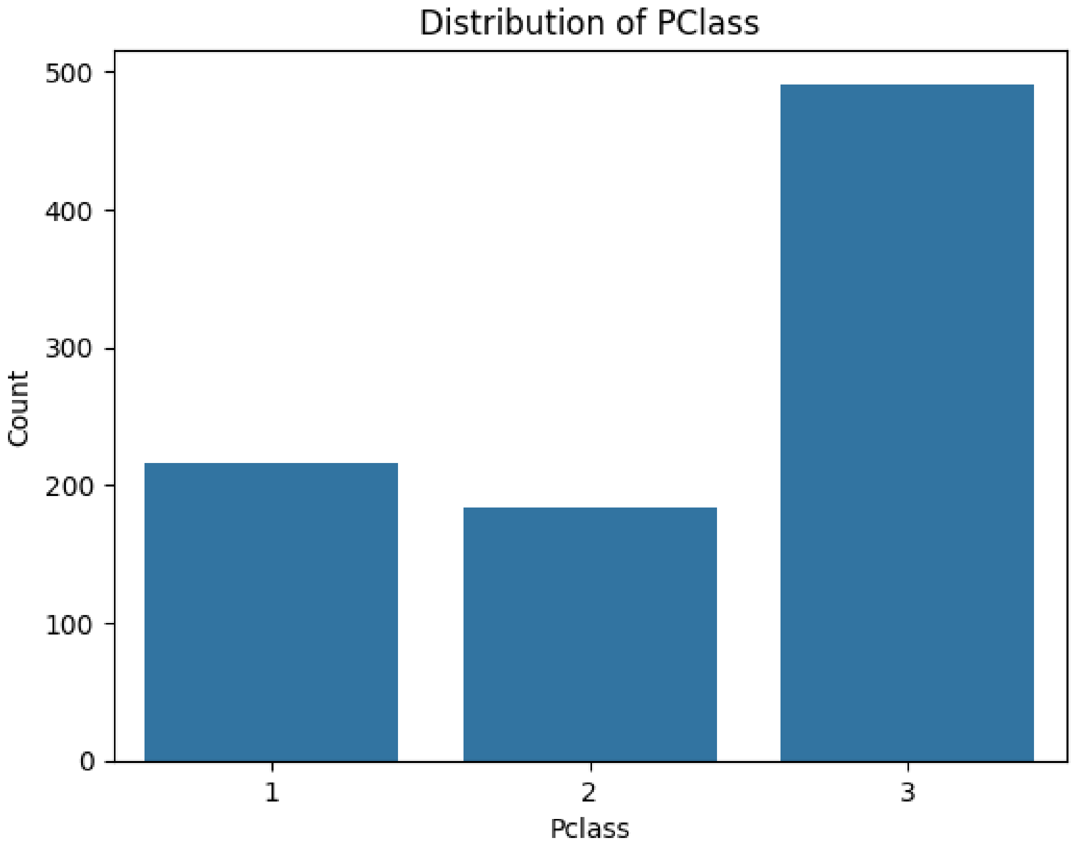
```
[13]: ## Datatype info
      train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

```
[15]: ## categorical attributes
      sns.countplot(data=train, x='Survived')

      # Add labels and title
      plt.xlabel('Survived')
      plt.ylabel('Count')
      plt.title('Distribution of Survival')

      # Show the plot
      plt.show()
```

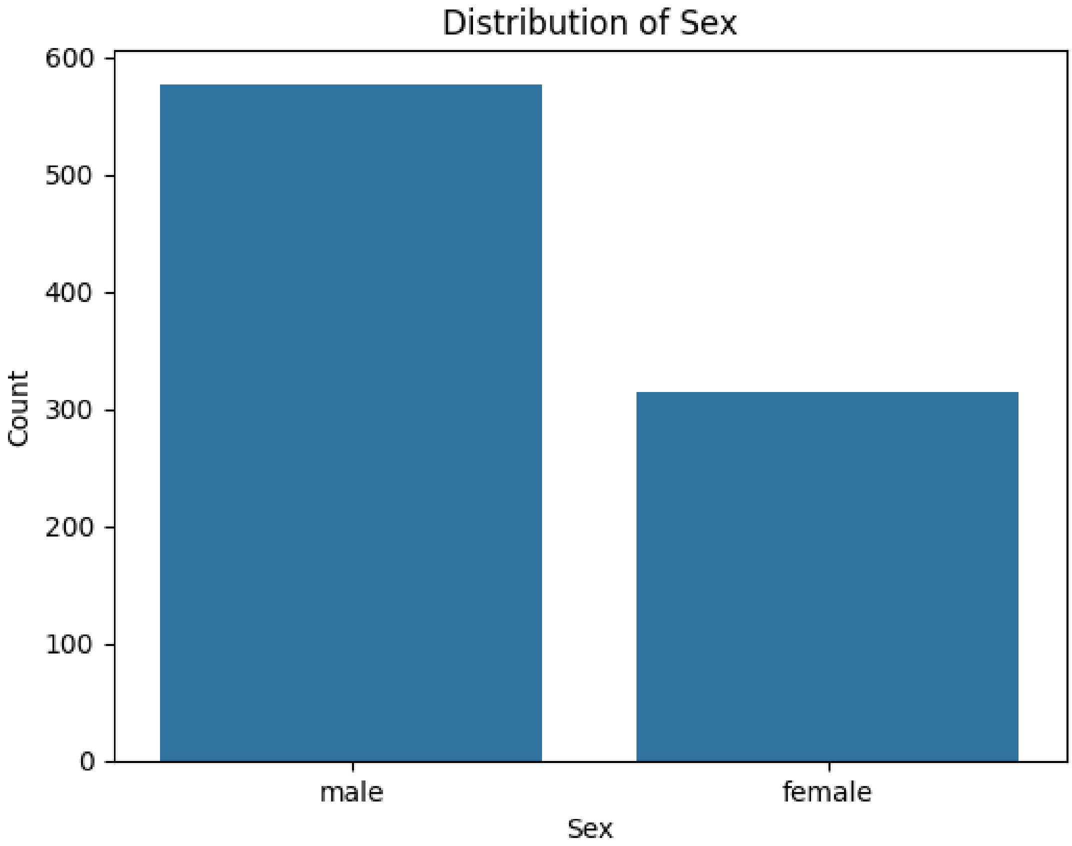


[17]:
```
sns.countplot(data=train, x='Pclass')

# Add labels and title
plt.xlabel('Pclass')
plt.ylabel('Count')
plt.title('Distribution of PClass')

# Show the plot
plt.show()
```



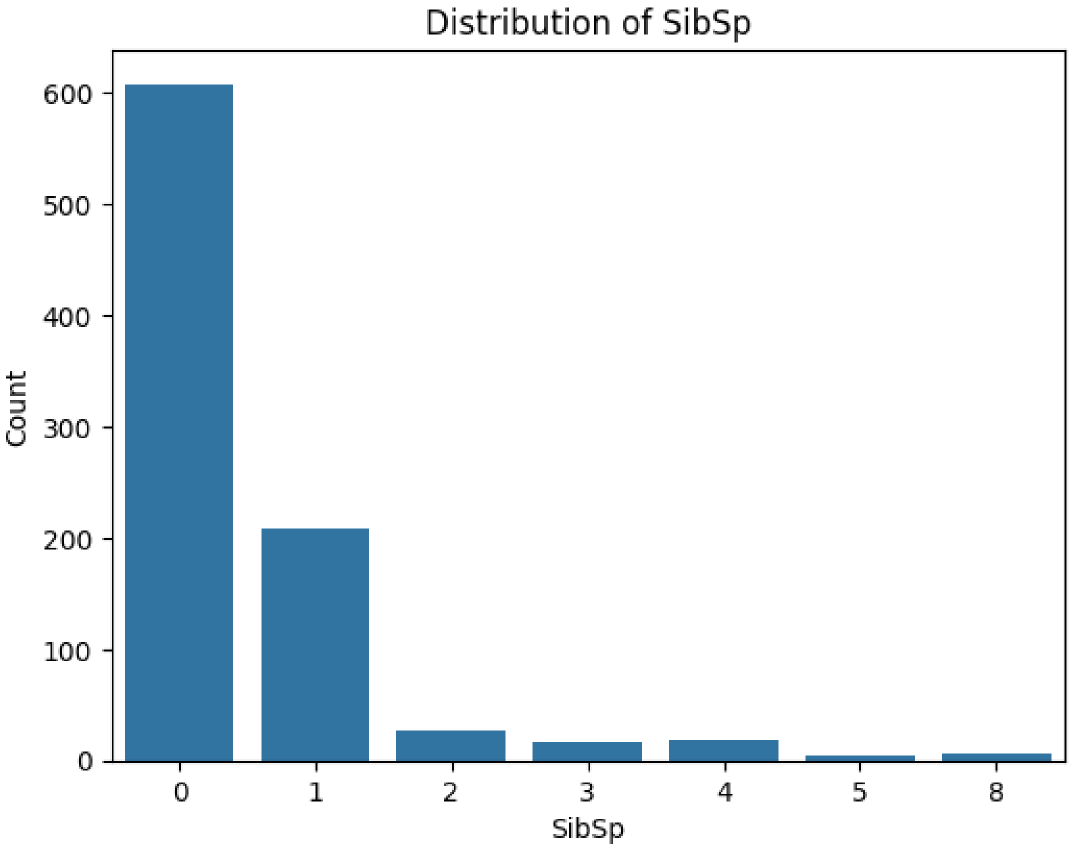
[18]:

```
sns.countplot(data=train, x='Sex')

# Add labels and title
plt.xlabel('Sex')
plt.ylabel('Count')
plt.title('Distribution of Sex')

# Show the plot
plt.show()
```

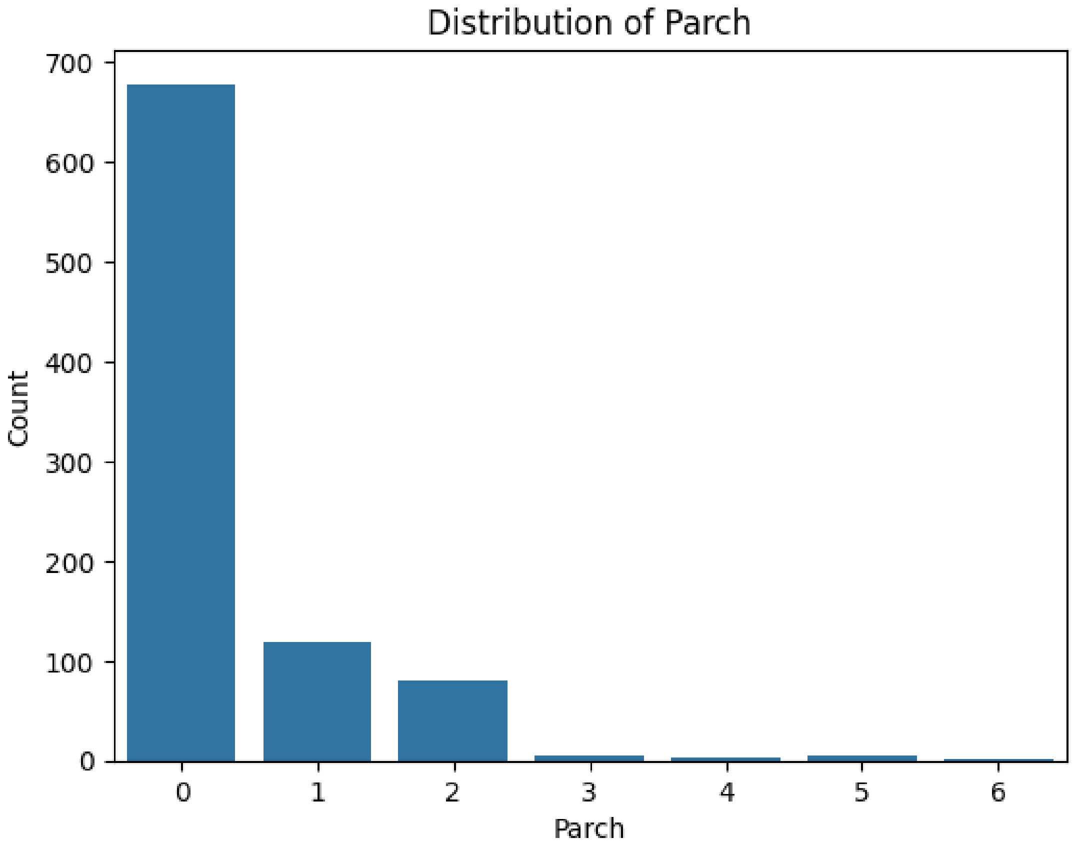


[19]:

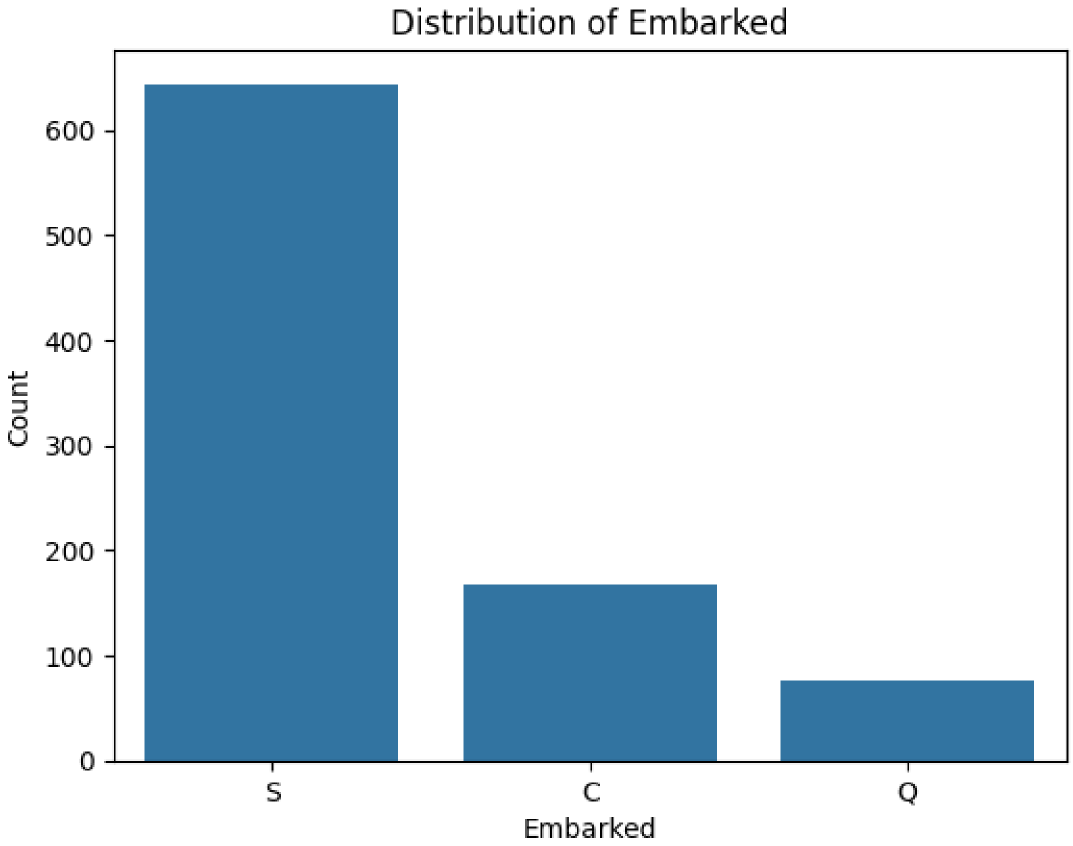
```
sns.countplot(data=train, x='SibSp')

# Add labels and title
plt.xlabel('SibSp')
plt.ylabel('Count')
plt.title('Distribution of SibSp')

# Show the plot
plt.show()
```



[20]:
```
sns.countplot(data=train, x='Parch')

# Add labels and title
plt.xlabel('Parch')
plt.ylabel('Count')
plt.title('Distribution of Parch')

# Show the plot
plt.show()
```

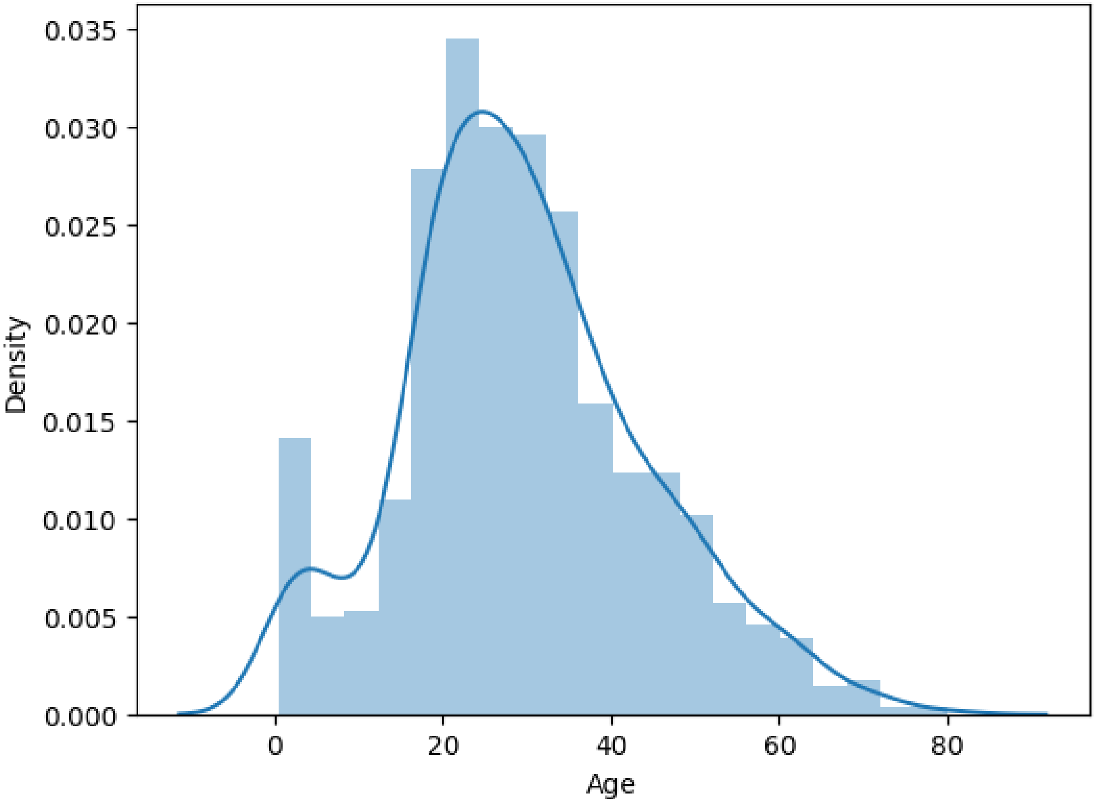


[21]:
```
sns.countplot(data=train, x='Embarked')

# Add labels and title
plt.xlabel('Embarked')
plt.ylabel('Count')
plt.title('Distribution of Embarked')

# Show the plot
plt.show()
```



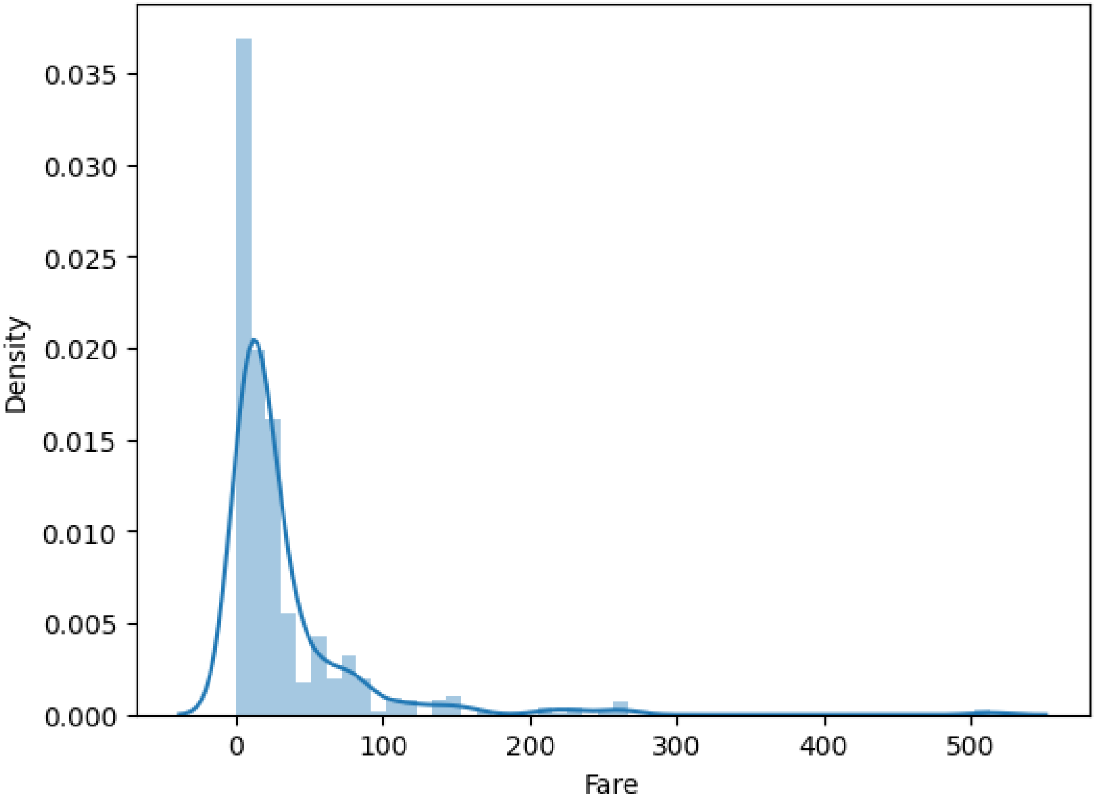
[22]:
```
## numerical attributes
sns.distplot(train['Age'])
```

[22]:
```
<Axes: xlabel='Age', ylabel='Density'>
```

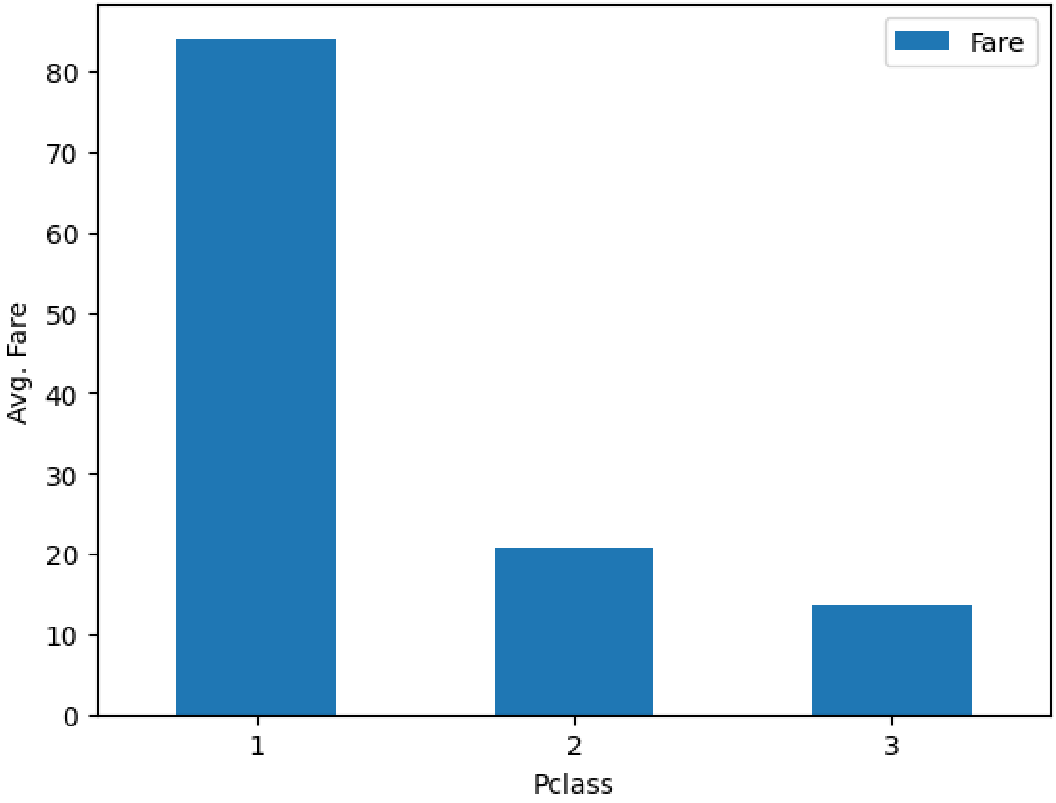


[23]:
```
sns.distplot(train['Fare'])
```

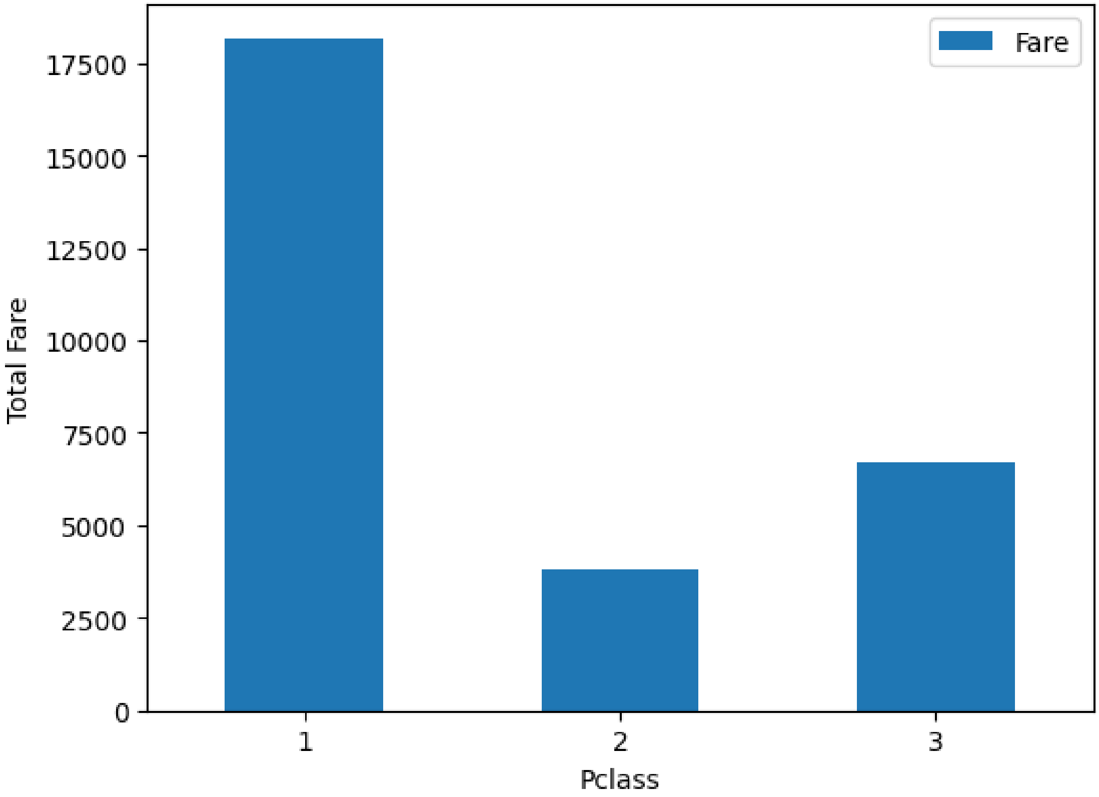
[23]:
```
<Axes: xlabel='Fare', ylabel='Density'>
```



[24]:
```
## Let us compare ticket classes by creating a new graph using a pivot table.



class_fare = train.pivot_table(index='Pclass', values='Fare')
class_fare.plot(kind='bar')
plt.xlabel('Pclass')
plt.ylabel('Avg. Fare')
plt.xticks(rotation=0)
plt.show()
```

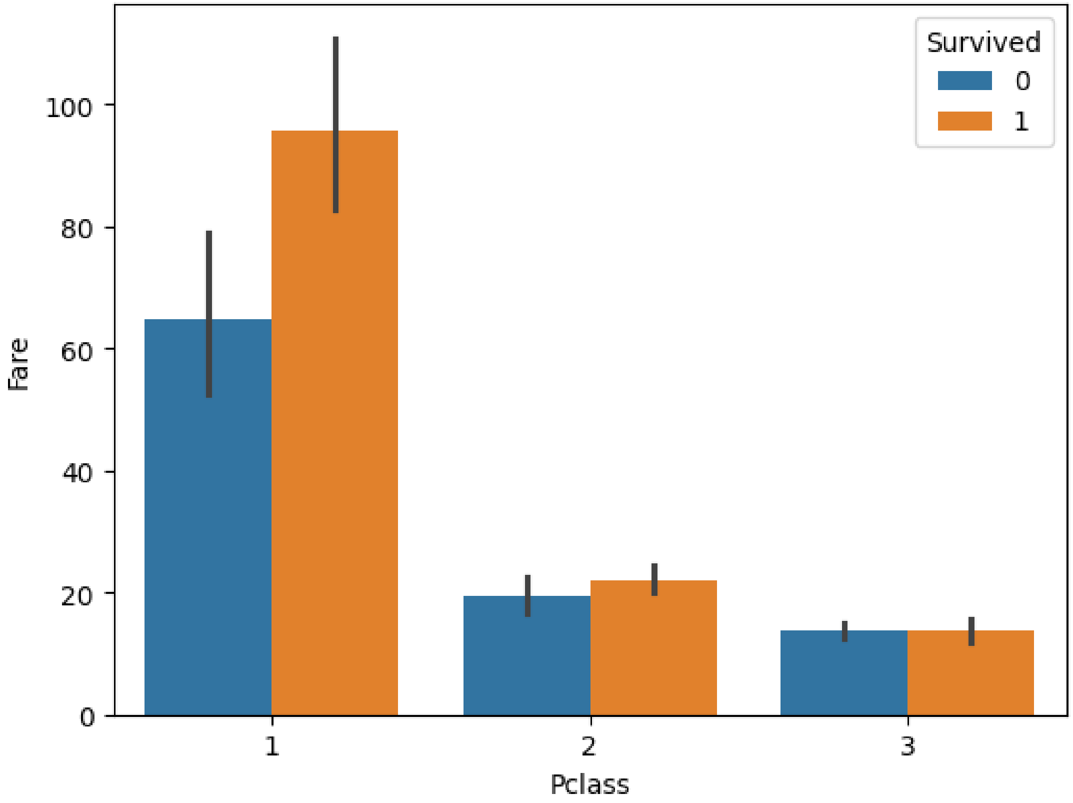


[25]:
```
## Let's compare Pclass by creating a new graph using a pivot table.

class_fare = train.pivot_table(index='Pclass', values='Fare', aggfunc=np.sum)
class_fare.plot(kind='bar')
plt.xlabel('Pclass')
plt.ylabel('Total Fare')
plt.xticks(rotation=0)
plt.show()
```



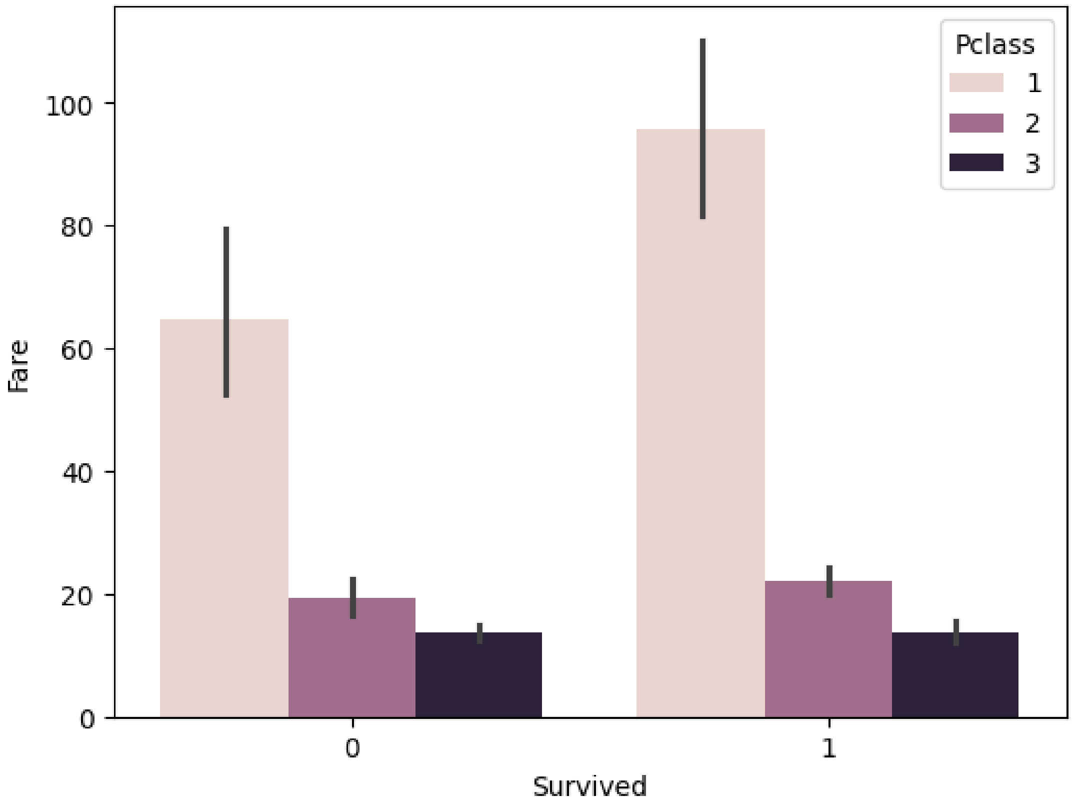
[26]:
```
## Let us display the difference between 'Pclass' and 'Survived' with the help␣
↪of a barplot.
sns.barplot(data=train, x='Pclass', y='Fare', hue='Survived')
```

[26]:
```
<Axes: xlabel='Pclass', ylabel='Fare'>
```



[28]:
```
## Let's change the horizontal and vertical axis of the graph.

sns.barplot(data=train, x='Survived', y='Fare', hue='Pclass')
```

[28]:
```
<Axes: xlabel='Survived', ylabel='Fare'>
```



[29]:
```
## Data Preprocessing

## We now combine the train and test datasets.

train_len = len(train)
# combine two dataframes
df = pd.concat([train, test], axis=0)
df = df.reset_index(drop=True)
df.head()
```

[29]:
```
   PassengerId  Survived  Pclass  \
0            1       0.0       3
1            2       1.0       1
2            3       1.0       3
3            4       1.0       1
4            5       0.0       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

   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
```

[30]:

```
df.tail()
```

[30]:

```
      PassengerId  Survived  Pclass                          Name     Sex  \
1304         1305       NaN       3            Spector, Mr. Woolf    male
1305         1306       NaN       1  Oliva y Ocana, Dona. Fermina  female
1306         1307       NaN       3  Saether, Mr. Simon Sivertsen    male
1307         1308       NaN       3           Ware, Mr. Frederick    male
1308         1309       NaN       3      Peter, Master. Michael J    male

       Age  SibSp  Parch              Ticket      Fare Cabin Embarked
1304   NaN      0      0           A.5. 3236    8.0500   NaN        S
1305  39.0      0      0            PC 17758  108.9000  C105        C
1306  38.5      0      0  SOTON/O.Q. 3101262    7.2500   NaN        S
1307   NaN      0      0              359309    8.0500   NaN        S
1308   NaN      1      1                2668   22.3583   NaN        C
```

[31]:

```
## find the null values
df.isnull().sum()
```

[31]:

```
PassengerId       0
Survived        418
Pclass            0
Name              0
Sex               0
Age             263
SibSp             0
Parch             0
Ticket            0
Fare              1
Cabin          1014
Embarked          2
dtype: int64
```

[32]:

```
# drop or delete the column
df = df.drop(columns=['Cabin'], axis=1)
```

[33]:

```
df['Age'].mean()
```

[33]:
```
29.881137667304014
```

[34]:
```
# fill missing values using mean of the numerical column
df['Age'] = df['Age'].fillna(df['Age'].mean())
df['Fare'] = df['Fare'].fillna(df['Fare'].mean())
```

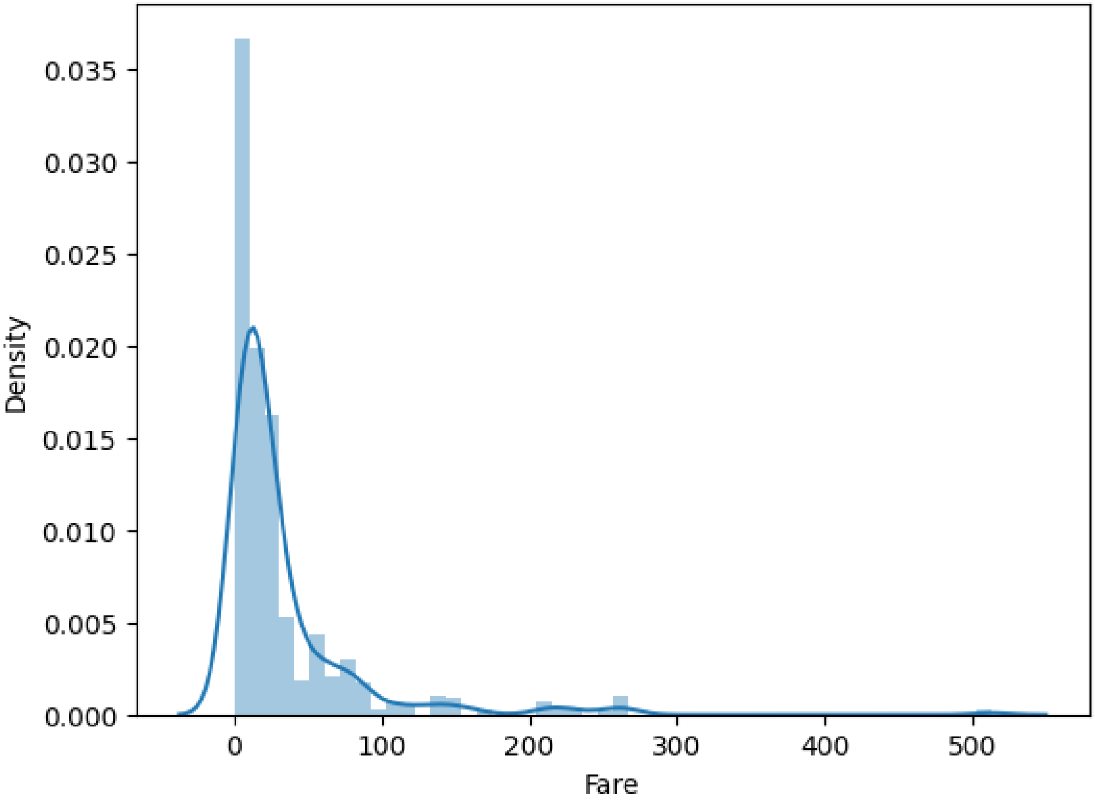
[35]:
```
df['Embarked'].mode()[0]
```

[35]:
```
'S'
```

[36]:
```
# fill missing values using mode of the categorical column
df['Embarked'] = df['Embarked'].fillna(df['Embarked'].mode()[0])
```

[37]:
```
sns.distplot(df['Fare'])
```

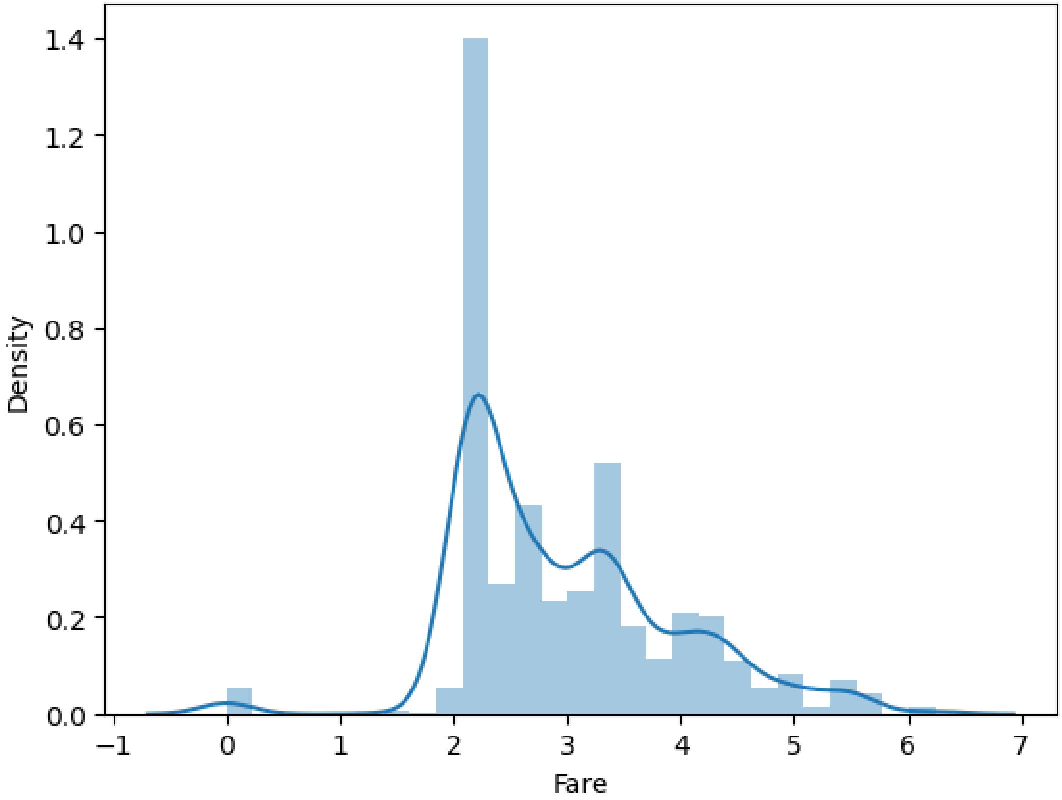
[37]:
```
<Axes: xlabel='Fare', ylabel='Density'>
```



[38]:
```
df['Fare'] = np.log(df['Fare']+1)
```

[39]:
```
sns.distplot(df['Fare'])
```

```
[39]: <Axes: xlabel='Fare', ylabel='Density'>
```



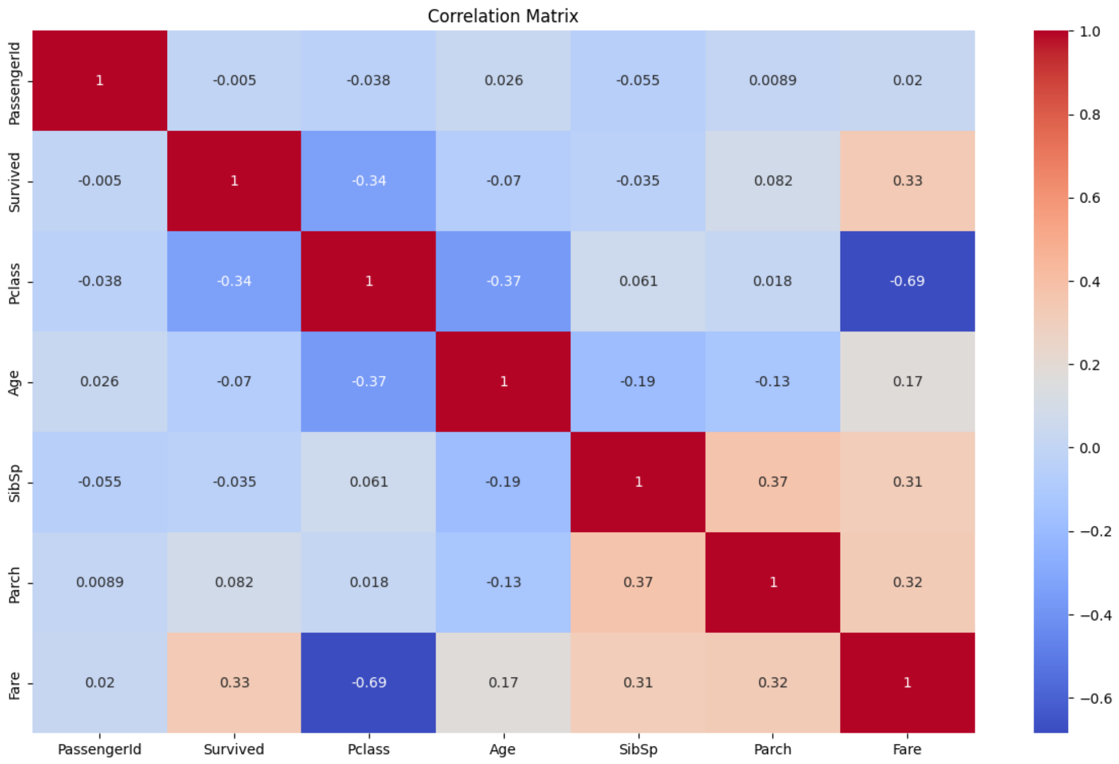
```
[40]: ## Correlation Matrix
corr = df.corr()
plt.figure(figsize=(15, 9))
sns.heatmap(corr, annot=True, cmap='coolwarm')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[40], line 2
      1 ## Correlation Matrix
----> 2 corr = df.corr()
      3 plt.figure(figsize=(15, 9))
      4 sns.heatmap(corr, annot=True, cmap='coolwarm')

File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\core\frame.
 ↪py:11022, in DataFrame.corr(self, method, min_periods, numeric_only)
  11020 cols = data.columns
  11021 idx = cols.copy()
> 11022 mat = data.to_numpy(dtype=float, na_value=np.nan, copy=False)
  11024 if method == "pearson":
```

```
  11025     correl = libalgos.nancorr(mat, minp=min_periods)

File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\core\frame.
 ↪py:1981, in DataFrame.to_numpy(self, dtype, copy, na_value)
   1979 if dtype is not None:
   1980     dtype = np.dtype(dtype)
-> 1981 result = self._mgr.as_array(dtype=dtype, copy=copy, na_value=na_value)
   1982 if result.dtype is not dtype:
   1983     result = np.array(result, dtype=dtype, copy=False)

File␣
 ↪~\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\core\internals\managers.
 ↪py:1693, in BlockManager.as_array(self, dtype, copy, na_value)
   1691         arr.flags.writeable = False
   1692 else:
-> 1693     arr = self._interleave(dtype=dtype, na_value=na_value)
   1694     # The underlying data was copied within _interleave, so no need
   1695     # to further copy if copy=True or setting na_value
   1697 if na_value is lib.no_default:

File␣
 ↪~\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\core\internals\managers.
 ↪py:1752, in BlockManager._interleave(self, dtype, na_value)
   1750     else:
   1751         arr = blk.get_values(dtype)
-> 1752     result[rl.indexer] = arr
   1753     itemmask[rl.indexer] = 1
   1755 if not itemmask.all():

ValueError: could not convert string to float: 'Braund, Mr. Owen Harris'
```

[41]:

```
# Calculate correlation matrix for numeric columns only
corr = df.corr(method='pearson', min_periods=1, numeric_only=True)

# Plot the correlation matrix
plt.figure(figsize=(15, 9))
sns.heatmap(corr, annot=True, cmap='coolwarm')

# Add title
plt.title('Correlation Matrix')

# Show the plot
plt.show()
```



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

```
[42]:    PassengerId  Survived  Pclass  \
      0            1       0.0       3
      1            2       1.0       1
      2            3       1.0       3
      3            4       1.0       1
      4            5       0.0       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

         Parch            Ticket      Fare Embarked
      0      0         A/5 21171  2.110213        S
      1      0          PC 17599  4.280593        C
      2      0  STON/O2. 3101282  2.188856        S
      3      0            113803  3.990834        S
      4      0            373450  2.202765        S
```

[43]:

```
## drop unnecessary columns
df = df.drop(columns=['Name', 'Ticket'], axis=1)
df.head()
```

[43]:

```
   PassengerId  Survived  Pclass     Sex   Age  SibSp  Parch      Fare  \
0            1       0.0       3    male  22.0      1      0  2.110213
1            2       1.0       1  female  38.0      1      0  4.280593
2            3       1.0       3  female  26.0      0      0  2.188856
3            4       1.0       1  female  35.0      1      0  3.990834
4            5       0.0       3    male  35.0      0      0  2.202765

  Embarked
0        S
1        C
2        S
3        S
4        S
```

[44]:

```
from sklearn.preprocessing import LabelEncoder
cols = ['Sex', 'Embarked']
le = LabelEncoder()

for col in cols:
    df[col] = le.fit_transform(df[col])
df.head()
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[44], line 1
----> 1 from sklearn.preprocessing import LabelEncoder
      2 cols = ['Sex', 'Embarked']
      3 le = LabelEncoder()

ModuleNotFoundError: No module named 'sklearn'
```

[45]:

```
pip install scikit-learn
```

```
Collecting scikit-learn
  Downloading scikit_learn-1.4.1.post1-cp312-cp312-win_amd64.whl.metadata (11
kB)
Requirement already satisfied: numpy<2.0,>=1.19.5 in
c:\users\sansk\appdata\local\programs\python\python312\lib\site-packages (from
scikit-learn) (1.26.3)
Requirement already satisfied: scipy>=1.6.0 in
c:\users\sansk\appdata\local\programs\python\python312\lib\site-packages (from
scikit-learn) (1.12.0)
Collecting joblib>=1.2.0 (from scikit-learn)
```

```
  Downloading joblib-1.3.2-py3-none-any.whl.metadata (5.4 kB)
Collecting threadpoolctl>=2.0.0 (from scikit-learn)
  Downloading threadpoolctl-3.4.0-py3-none-any.whl.metadata (13 kB)
Downloading scikit_learn-1.4.1.post1-cp312-cp312-win_amd64.whl (10.6 MB)
   ---------------------------------------- 0.0/10.6 MB ? eta -:--:--
   ---------------------------------------- 0.0/10.6 MB 960.0 kB/s eta 0:00:12
    --------------------------------------- 0.2/10.6 MB 3.1 MB/s eta 0:00:04
   -- ------------------------------------- 0.5/10.6 MB 4.7 MB/s eta 0:00:03
   ----- ---------------------------------- 1.6/10.6 MB 9.1 MB/s eta 0:00:01
   -------------- ------------------------- 3.9/10.6 MB 17.6 MB/s eta 0:00:01
   ------------------- -------------------- 5.2/10.6 MB 20.9 MB/s eta 0:00:01
   ------------------- -------------------- 5.2/10.6 MB 20.9 MB/s eta 0:00:01
   ------------------- -------------------- 5.2/10.6 MB 20.9 MB/s eta 0:00:01
   ------------------- -------------------- 5.2/10.6 MB 20.9 MB/s eta 0:00:01
   ------------------- -------------------- 5.2/10.6 MB 20.9 MB/s eta 0:00:01
   ------------------- -------------------- 5.2/10.6 MB 20.9 MB/s eta 0:00:01
   ----------------------- ---------------- 6.3/10.6 MB 11.9 MB/s eta 0:00:01
   ------------------------------ --------- 8.0/10.6 MB 14.2 MB/s eta 0:00:01
   ---------------------------------------  10.6/10.6 MB 19.3 MB/s eta 0:00:01
   ---------------------------------------- 10.6/10.6 MB 18.7 MB/s eta 0:00:00
Downloading joblib-1.3.2-py3-none-any.whl (302 kB)
   ---------------------------------------- 0.0/302.2 kB ? eta -:--:--
   ---------------------------------------- 302.2/302.2 kB ? eta 0:00:00
Downloading threadpoolctl-3.4.0-py3-none-any.whl (17 kB)
Installing collected packages: threadpoolctl, joblib, scikit-learn
Successfully installed joblib-1.3.2 scikit-learn-1.4.1.post1 threadpoolctl-3.4.0
Note: you may need to restart the kernel to use updated packages.
```

[46]:

```python
from sklearn.preprocessing import LabelEncoder

# Assuming 'df' is your DataFrame containing the Titanic dataset
# Make sure it's properly loaded

cols = ['Sex', 'Embarked']
le = LabelEncoder()

# Encode categorical columns
for col in cols:
    df[col] = le.fit_transform(df[col].astype(str))
```

[48]:

```python
df.head()
```

[48]:

```
   PassengerId  Survived  Pclass  Sex   Age  SibSp  Parch      Fare  Embarked
0            1       0.0       3    1  22.0      1      0  2.110213         2
1            2       1.0       1    0  38.0      1      0  4.280593         0
2            3       1.0       3    0  26.0      0      0  2.188856         2
3            4       1.0       1    0  35.0      1      0  3.990834         2
```

```
4            5       0.0       3    1  35.0      0      0  2.202765         2
```

[49]:
```
## Train-Test Split
train = df.iloc[:train_len, :]
test = df.iloc[train_len:, :]
train.head()
```

[49]:
```
   PassengerId  Survived  Pclass  Sex   Age  SibSp  Parch      Fare  Embarked
0            1       0.0       3    1  22.0      1      0  2.110213         2
1            2       1.0       1    0  38.0      1      0  4.280593         0
2            3       1.0       3    0  26.0      0      0  2.188856         2
3            4       1.0       1    0  35.0      1      0  3.990834         2
4            5       0.0       3    1  35.0      0      0  2.202765         2
```

[50]:
```
test.head()
```

[50]:
```
     PassengerId  Survived  Pclass  Sex   Age  SibSp  Parch      Fare  \
891          892       NaN       3    1  34.5      0      0  2.178064
892          893       NaN       3    0  47.0      1      0  2.079442
893          894       NaN       2    1  62.0      0      0  2.369075
894          895       NaN       3    1  27.0      0      0  2.268252
895          896       NaN       3    0  22.0      1      1  2.586824

     Embarked
891         1
892         2
893         1
894         2
895         2
```

[51]:
```
# input split
X = train.drop(columns=['PassengerId', 'Survived'], axis=1)
y = train['Survived']
X.head()
```

[51]:
```
   Pclass  Sex   Age  SibSp  Parch      Fare  Embarked
0       3    1  22.0      1      0  2.110213         2
1       1    0  38.0      1      0  4.280593         0
2       3    0  26.0      0      0  2.188856         2
3       1    0  35.0      1      0  3.990834         2
4       3    1  35.0      0      0  2.202765         2
```

[52]:
```
## Model Training
from sklearn.model_selection import train_test_split, cross_val_score
# classify column
def classify(model):
    x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.25,␣
 ↪random_state=42)
```

```
    model.fit(x_train, y_train)
    print('Accuracy:', model.score(x_test, y_test))

    score = cross_val_score(model, X, y, cv=5)
    print('CV Score:', np.mean(score))
```

[53]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
classify(model)
```

```
Accuracy: 0.8071748878923767
CV Score: 0.7833971502102819
```

[54]:
```
## Decision Tree
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
classify(model)
```

```
Accuracy: 0.7354260089686099
CV Score: 0.7699516665620488
```

[56]:
```
## Random Forest
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
classify(model)
```

```
Accuracy: 0.7847533632286996
CV Score: 0.811493314920595
```

[58]:
```
## Extra Trees:
from sklearn.ensemble import ExtraTreesClassifier
model = ExtraTreesClassifier()
classify(model)
```

```
Accuracy: 0.8026905829596412
CV Score: 0.7890214048082356
```

[ ]:
```
from xgboost import XGBClassifier
model = XGBClassifier()
classify(model)
```

[67]:
```
from catboost import CatBoostClassifier
model = CatBoostClassifier(verbose=0)
classify(model)
```

```
Accuracy: 0.8295964125560538
CV Score: 0.8226790534178645
```

[74]:
```
test.head()
```

```
[74]:      PassengerId  Survived  Pclass  Sex   Age  SibSp  Parch      Fare  \
      891          892       NaN       3    1  34.5      0      0  2.178064
      892          893       NaN       3    0  47.0      1      0  2.079442
      893          894       NaN       2    1  62.0      0      0  2.369075
      894          895       NaN       3    1  27.0      0      0  2.268252
      895          896       NaN       3    0  22.0      1      1  2.586824

           Embarked
      891         1
      892         2
      893         1
      894         2
      895         2
```

```
[75]: # input split for test data
      X_test = test.drop(columns=['PassengerId', 'Survived'], axis=1)
      X_test.head()
```

```
[75]:      Pclass  Sex   Age  SibSp  Parch      Fare  Embarked
      891       3    1  34.5      0      0  2.178064         1
      892       3    0  47.0      1      0  2.079442         2
      893       2    1  62.0      0      0  2.369075         1
      894       3    1  27.0      0      0  2.268252         2
      895       3    0  22.0      1      1  2.586824         2
```

```
[76]: pred = model.predict(X_test)
      pred
```

```
[76]: array([0., 0., 0., 1., 0., 0., 1., 0., 1., 0., 0., 0., 1., 0., 1., 1., 0.,
             0., 1., 1., 1., 0., 1., 1., 1., 0., 1., 1., 1., 0., 0., 0., 1., 0.,
             1., 0., 0., 0., 0., 0., 0., 0., 0., 1., 1., 0., 0., 0., 1., 1., 0.,
             0., 1., 1., 0., 0., 0., 0., 0., 1., 0., 1., 0., 1., 0., 1., 1., 0.,
             0., 1., 1., 0., 0., 0., 1., 1., 0., 1., 0., 1., 1., 0., 0., 0., 0.,
             0., 1., 1., 1., 1., 0., 0., 1., 0., 0., 0., 1., 0., 0., 0., 1., 0.,
             0., 0., 1., 0., 0., 0., 0., 0., 0., 1., 1., 1., 1., 0., 0., 1., 1.,
             1., 1., 0., 1., 0., 0., 1., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0.,
             1., 0., 0., 0., 0., 1., 0., 0., 1., 0., 0., 0., 0., 0., 1., 0., 0.,
             1., 0., 0., 1., 0., 1., 1., 1., 1., 1., 0., 0., 0., 0., 0., 1., 0.,
             0., 1., 0., 0., 0., 1., 1., 1., 1., 1., 0., 0., 1., 0., 1., 0., 1.,
             0., 0., 0., 0., 0., 0., 0., 1., 0., 1., 0., 0., 0., 1., 1., 0., 1.,
             0., 0., 1., 0., 1., 0., 0., 0., 0., 1., 0., 0., 1., 0., 1., 0., 1.,
             0., 1., 0., 1., 0., 0., 1., 0., 0., 0., 1., 0., 0., 1., 0., 1., 0.,
             1., 1., 1., 1., 0., 0., 0., 0., 1., 0., 1., 0., 1., 0., 1., 0., 0.,
             0., 0., 0., 1., 0., 0., 0., 1., 1., 0., 0., 0., 0., 0., 0., 0., 0.,
             1., 1., 0., 1., 0., 0., 0., 0., 0., 1., 1., 1., 1., 0., 0., 0., 0.,
             0., 0., 1., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 1., 1.,
             0., 1., 0., 0., 0., 1., 0., 1., 1., 1., 1., 0., 1., 0., 0., 0., 0.,
             1., 1., 0., 1., 0., 0., 0., 1., 0., 0., 1., 0., 0., 1., 0., 0., 0.,
             0., 0., 0., 1., 0., 1., 0., 1., 0., 1., 1., 0., 0., 0., 1., 0., 1.,
             0., 0., 1., 0., 1., 1., 1., 1., 0., 0., 0., 1., 1., 0., 1., 0., 0.,
             1., 1., 0., 0., 0., 1., 0., 0., 0., 1., 1., 1., 0., 0., 0., 0., 0.,
             1., 0., 0., 0., 1., 0., 1., 0., 0., 1., 0., 1., 0., 0., 0., 0., 0.,
             1., 1., 1., 1., 1., 0., 1., 0., 0., 0.])
```

[ ]: