# TITANIC SURVIVAL PREDICTION

September 24, 2024

## 1 Loading Libraries

[61]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import OneHotEncoder
```

[62]:
```
data=pd.read_csv("Titanic-Dataset.csv")
data.head(10)
```

[62]:

| | PassengerId | Survived | Pclass | Name | Sex | Age | SibSp | Parch | Ticket | Fare | Cabin | Embarked |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 3 | Braund, Mr. Owen Harris | male | 22.0 | 1 | 0 | A/5 21171 | 7.2500 | NaN | S |
| 1 | 2 | 1 | 1 | Cumings, Mrs. John Bradley (Florence Briggs Th… | female | 38.0 | 1 | | | | | |
| 2 | 3 | 1 | 3 | Heikkinen, Miss. Laina | female | 26.0 | 0 | | | | | |
| 3 | 4 | 1 | 1 | Futrelle, Mrs. Jacques Heath (Lily May Peel) | female | 35.0 | 1 | | | | | |
| 4 | 5 | 0 | 3 | Allen, Mr. William Henry | male | 35.0 | 0 | | | | | |
| 5 | 6 | 0 | 3 | Moran, Mr. James | male | NaN | 0 | | | | | |
| 6 | 7 | 0 | 1 | McCarthy, Mr. Timothy J | male | 54.0 | 0 | | | | | |
| 7 | 8 | 0 | 3 | Palsson, Master. Gosta Leonard | male | 2.0 | 3 | | | | | |
| 8 | 9 | 1 | 3 | Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg) | female | 27.0 | 0 | | | | | |
| 9 | 10 | 1 | 2 | Nasser, Mrs. Nicholas (Adele Achem) | female | 14.0 | 1 | | | | | |

```
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
5      0            330877   8.4583   NaN        Q
6      0             17463  51.8625   E46        S
7      1            349909  21.0750   NaN        S
8      2            347742  11.1333   NaN        S
9      0            237736  30.0708   NaN        C
```

## 2 Drop the unnecessary columns from the data

[63]:
```
data_new=data.drop(["PassengerId","Name","Cabin","Ticket"],axis=1)
data_new.head(10)
```

[63]:

| | Survived | Pclass | Sex | Age | SibSp | Parch | Fare | Embarked |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 3 | male | 22.0 | 1 | 0 | 7.2500 | S |
| 1 | 1 | 1 | female | 38.0 | 1 | 0 | 71.2833 | C |
| 2 | 1 | 3 | female | 26.0 | 0 | 0 | 7.9250 | S |
| 3 | 1 | 1 | female | 35.0 | 1 | 0 | 53.1000 | S |
| 4 | 0 | 3 | male | 35.0 | 0 | 0 | 8.0500 | S |
| 5 | 0 | 3 | male | NaN | 0 | 0 | 8.4583 | Q |
| 6 | 0 | 1 | male | 54.0 | 0 | 0 | 51.8625 | S |
| 7 | 0 | 3 | male | 2.0 | 3 | 1 | 21.0750 | S |
| 8 | 1 | 3 | female | 27.0 | 0 | 2 | 11.1333 | S |
| 9 | 1 | 2 | female | 14.0 | 1 | 0 | 30.0708 | C |

## 3 Calculate Statistical values

[64]:
```
data_new.describe().round(3)
```

[64]:

| | Survived | Pclass | Age | SibSp | Parch | Fare |
|---|---|---|---|---|---|---|
| count | 891.000 | 891.000 | 714.000 | 891.000 | 891.000 | 891.000 |
| mean | 0.384 | 2.309 | 29.699 | 0.523 | 0.382 | 32.204 |
| std | 0.487 | 0.836 | 14.526 | 1.103 | 0.806 | 49.693 |
| min | 0.000 | 1.000 | 0.420 | 0.000 | 0.000 | 0.000 |
| 25% | 0.000 | 2.000 | 20.125 | 0.000 | 0.000 | 7.910 |
| 50% | 0.000 | 3.000 | 28.000 | 0.000 | 0.000 | 14.454 |
| 75% | 1.000 | 3.000 | 38.000 | 1.000 | 0.000 | 31.000 |
| max | 1.000 | 3.000 | 80.000 | 8.000 | 6.000 | 512.329 |

## 4 Checking Null values and Filling them

[65]:
```
data_new.isnull().sum()
```

[65]:
```
Survived      0
Pclass        0
Sex           0
Age         177
SibSp         0
Parch         0
Fare          0
Embarked      2
dtype: int64
```

[66]:
```
data_new.dtypes
```

[66]:
```
Survived      int64
Pclass        int64
Sex          object
Age         float64
SibSp         int64
Parch         int64
Fare        float64
Embarked     object
dtype: object
```

[67]:
```
data_new["Embarked"].unique()
```

[67]:
```
array(['S', 'C', 'Q', nan], dtype=object)
```

[68]:
```
data_new["Embarked"]=data_new["Embarked"].fillna(method="ffill")
```

[69]:
```
data_new["Embarked"].unique()
```

[69]:
```
array(['S', 'C', 'Q'], dtype=object)
```

[70]:
```
ord_data={"S":0,"C":1,"Q":2}
data_new["Embarked"]=data_new["Embarked"].map(ord_data)
```

[71]:
```
data_new.dtypes
```

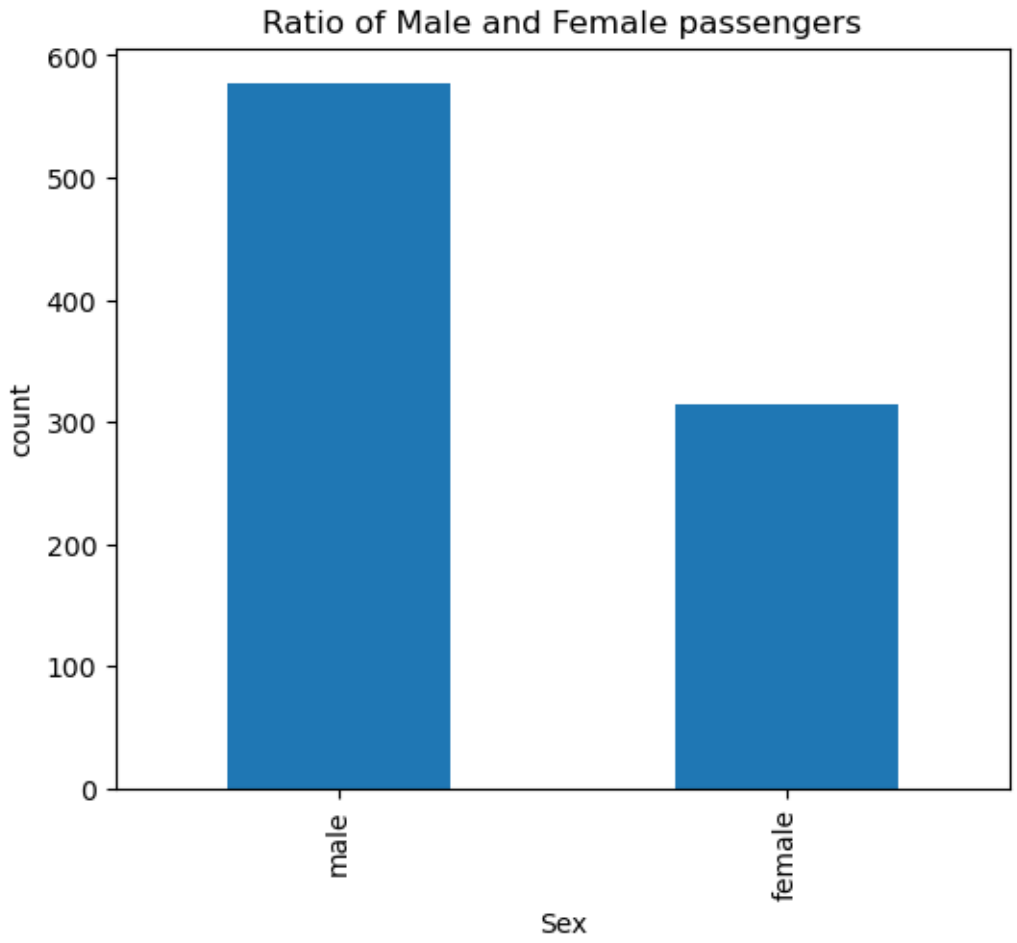
[71]:
```
Survived      int64
Pclass        int64
Sex          object
Age         float64
SibSp         int64
Parch         int64
Fare        float64
Embarked      int64
dtype: object
```

[72]:
```
data_new["Age"]=data_new["Age"].fillna(data_new["Age"].mean())
data_new["Embarked"]=data_new["Embarked"].fillna(data_new["Embarked"].mean())
data_new.isnull().sum()
```

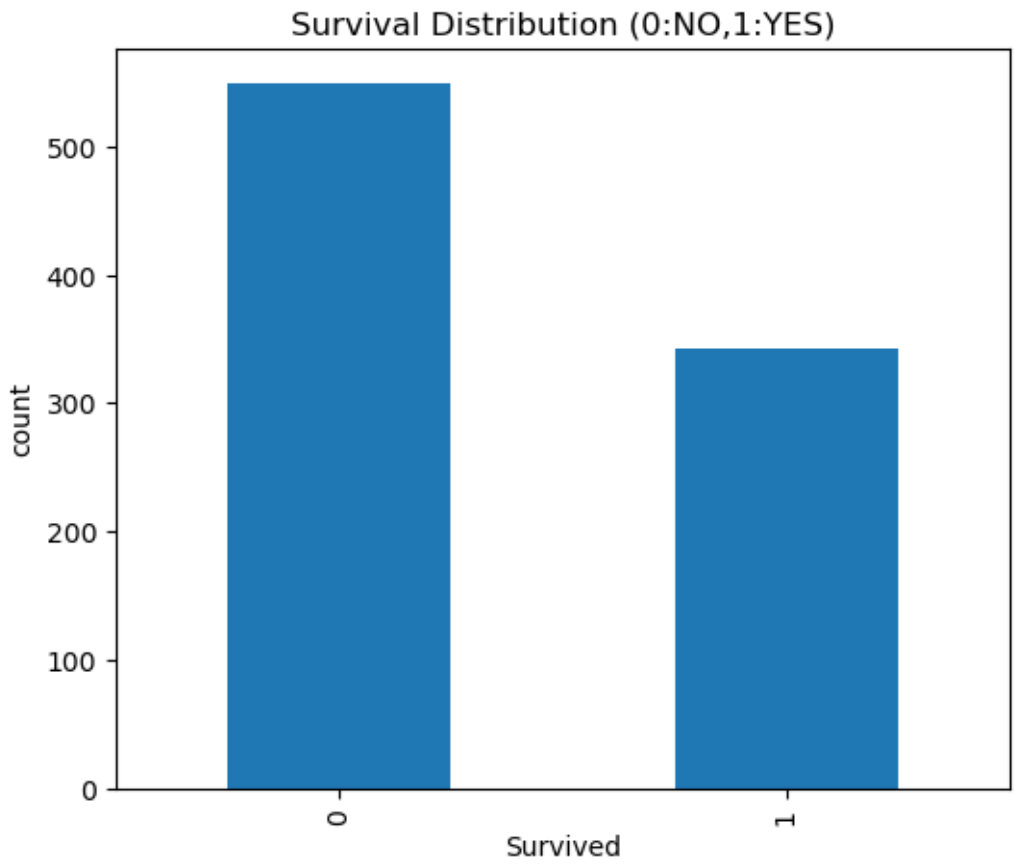
[72]:
```
Survived    0
Pclass      0
Sex         0
Age         0
SibSp       0
Parch       0
Fare        0
Embarked    0
dtype: int64
```

## 5 Visualization of Passenger Survival

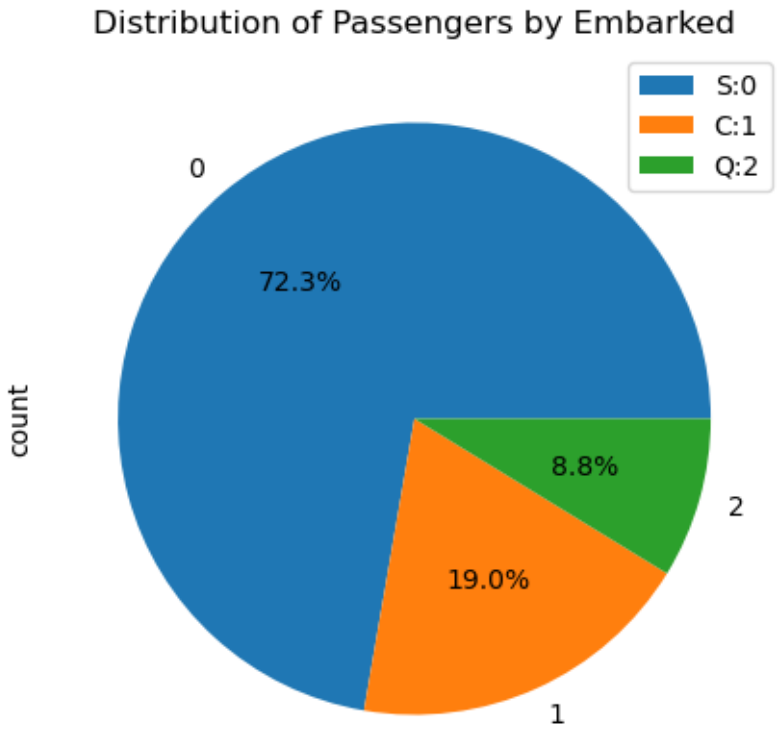
[73]:
```
ax=data_new["Sex"].value_counts().plot(kind="bar",x="Survived",figsize=(6,5))
plt.title("Ratio of Male and Female passengers")
plt.ylabel("count")
plt.xlabel("Sex")
plt.show()
```



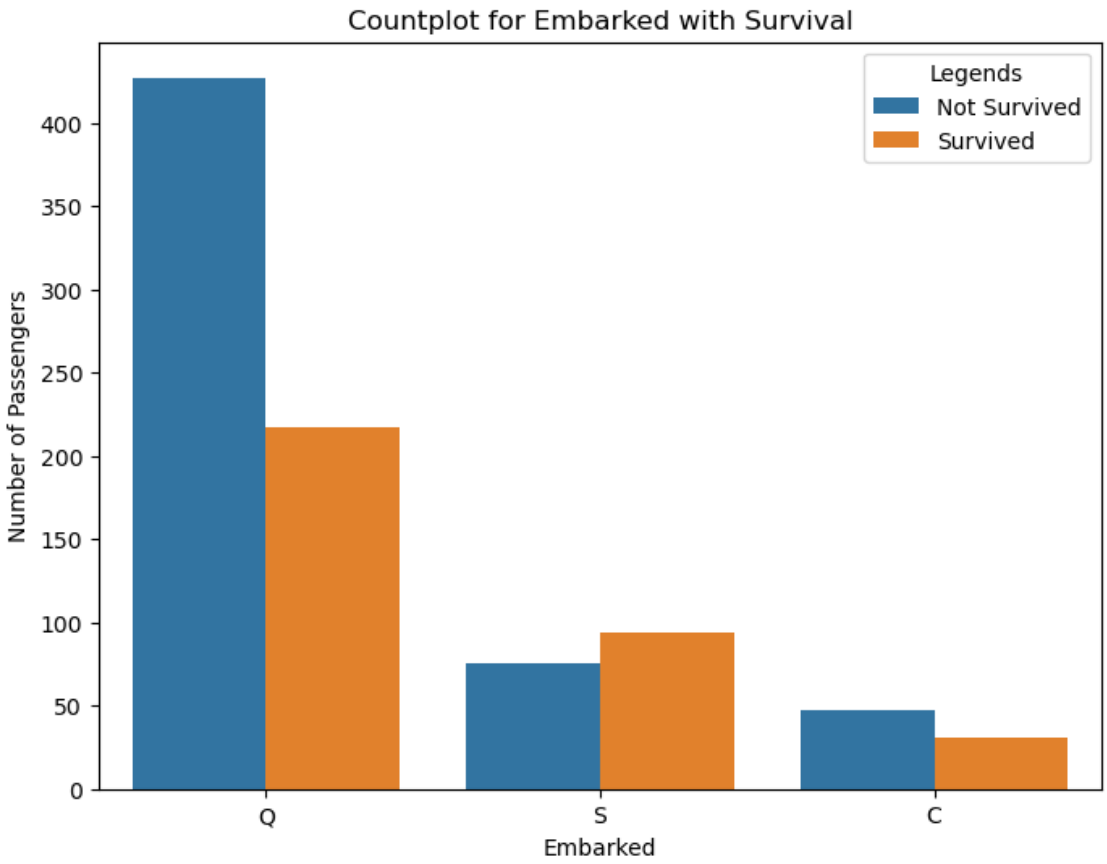

[19]:
```
ax=data_new["Survived"].value_counts().plot(kind="bar",figsize=(6,5))
plt.title("Survival Distribution (0:NO,1:YES)")
plt.ylabel("count")
plt.xlabel("Survived")
plt.show()
```




[24]:
```
ax=data_new["Embarked"].value_counts().
  ↪plot(kind="pie",figsize=(6,5),autopct='%1.1f%%')
plt.title("Distribution of Passengers by Embarked")
plt.legend(["S:0","C:1","Q:2"])
plt.show()
```




[21]:
```
fig, ax = plt.subplots(figsize=(8, 6))
sns.countplot(data=data_new, x="Embarked", hue="Survived", ax=ax)
ax.set_title("Countplot for Embarked with Survival")
ax.set_xlabel("Embarked")
ax.set_xticklabels(["Q", "S", "C"])
ax.set_ylabel("Number of Passengers")
ax.legend(title="Legends", labels=["Not Survived", "Survived"])
plt.show()
```




[90]:
```
data_new["Sex"]=data_new["Sex"].apply({"male":1,"female":0}.get)
data_new.head(10).round(3)
```

[90]:

| | Survived | Pclass | Sex | Age | SibSp | Parch | Fare | Embarked |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 3 | None | 22.000 | 1 | 0 | 7.250 | 0 |
| 1 | 1 | 1 | None | 38.000 | 1 | 0 | 71.283 | 1 |
| 2 | 1 | 3 | None | 26.000 | 0 | 0 | 7.925 | 0 |
| 3 | 1 | 1 | None | 35.000 | 1 | 0 | 53.100 | 0 |
| 4 | 0 | 3 | None | 35.000 | 0 | 0 | 8.050 | 0 |
| 5 | 0 | 3 | None | 29.699 | 0 | 0 | 8.458 | 2 |
| 6 | 0 | 1 | None | 54.000 | 0 | 0 | 51.862 | 0 |
| 7 | 0 | 3 | None | 2.000 | 3 | 1 | 21.075 | 0 |
| 8 | 1 | 3 | None | 27.000 | 0 | 2 | 11.133 | 0 |
| 9 | 1 | 2 | None | 14.000 | 1 | 0 | 30.071 | 1 |

## 6 Splitting and Training the Cleaned data

[91]:
```
x=data_new.drop(["Survived"],axis=1)
y=data_new["Survived"]
```

[98]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.8)
```

[99]:
```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=5)
```

[113]:
```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy='mean')
x_train = imputer.fit_transform(x_train)
x_test = imputer.fit_transform(x_test)
```

[110]:
```
knn.fit(x_train,y_train)
```

[110]:
```
KNeighborsClassifier()
```

[112]:
```
predictions=knn.predict(x_test)
print(predictions)
```

```
[0 0 1 0 0 0 1 1 1 0 0 0 1 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
 0 0 0 1 1 0 1 0 0 0 0 1 0 0 1 0 0 0 0 0 0 1 0 0 0 0 1 1 1 1 0 1 0 0 0 0 1
 1 0 1 0 0 1 0 1 1 0 0 1 0 0 0 0 0 0 1 0 1 0 0 1 1 1 0 0 0 0 0 1 0 0 0 0 0 1
 1 1 0 0 1 0 1 0 0 0 0 1 0 0 1 0 1 1 1 0 0 1 0 1 0 0 1 0 0 0 1 1 0 1 1 1 0
 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0 1 0 1 0 0 0 1 0 1 0 0]
```

## 7 Accuracy score and confusion metrices

[117]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
ac=accuracy_score(y_test,predictions)
cm=confusion_matrix(y_test,predictions)
```

[118]:
```
print(cm)
```

```
[[92 23]
 [28 36]]
```

[119]:
```
print(ac)
```

```
0.7150837988826816
```

[120]:
```
from sklearn.tree import DecisionTreeClassifier
tree=DecisionTreeClassifier()
```

[121]:
```
tree.fit(x_train,y_train)
```

[121]:
```
DecisionTreeClassifier()
```

[122]:
```
predictions=tree.predict(x_test)
print(predictions)
```

```
[0 0 1 0 0 0 1 0 1 0 0 0 0 0 1 1 0 0 0 1 0 1 0 0 1 0 0 0 0 0 1 1 0 0 1 0 0
 0 0 0 1 1 0 1 1 0 1 1 1 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 1 0 0 0 0 0 0 0 1
 1 0 1 0 0 1 0 0 1 0 0 0 1 0 0 0 0 1 1 1 0 0 1 0 0 0 0 0 0 0 0 1 0 1 1 0 0
 0 1 0 0 1 1 1 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 1 1 1 0 1 1 0 0 0 1 1 0 0 0 0 0
 0 1 0 0 0 0 0 1 1 0 0 0 1 1 1 1 1 0 0 1 0 0 0 0 0 0 0 0 1 1 1 0 1]
```

[123]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
ac=accuracy_score(y_test,predictions)
cm=confusion_matrix(y_test,predictions)
```

[124]:
```
print(cm)
```

```
[[89 26]
 [28 36]]
```

[125]:
```
print(ac)
```

```
0.6983240223463687
```

[126]:
```
from sklearn.svm import SVC
svm=SVC()
```

[127]:
```
svm.fit(x_train,y_train)
```

[127]:
```
SVC()
```

[129]:
```
predictions=svm.predict(x_test)
print(predictions)
```

```
[0 0 1 0 0 0 0 0 1 0 0 0 1 0 0 1 1 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0
 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1
 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
 0 0 0 0 1 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0 1 1 0 1 0 0 0
 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0]
```

[130]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
ac=accuracy_score(y_test,predictions)
cm=confusion_matrix(y_test,predictions)
```

[131]:
```
print(cm)
```

```
[[106   9]
 [ 42  22]]
```

[132]:
```
print(ac)
```

```
0.7150837988826816
```