In [3]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

Data Loading and processing.

In [70]:
```
#Load the dataset
titanic_data = pd.read_csv('titanic.csv')
```

In [71]:
```
titanic_data.head()
```

Out[71]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabi
0	892	0	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	Nal
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	Nal
2	894	0	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	Nal
3	895	0	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	Nal
4	896	1	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	Nal

In [72]:
```
titanic_data.shape
```

Out[72]:
```
(418, 12)
```

In [73]:
```
titanic_data.index
```

Out[73]:
```
RangeIndex(start=0, stop=418, step=1)
```

In [74]:
```
titanic_data.columns
```

Out[74]:
```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

In [75]:
```
titanic_data.describe()
```

Out[75]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	418.000000	418.000000	418.000000	332.000000	418.000000	418.000000	417.000000
mean	1100.500000	0.363636	2.265550	30.272590	0.447368	0.392344	35.627188
std	120.810458	0.481622	0.841838	14.181209	0.896760	0.981429	55.907576
min	892.000000	0.000000	1.000000	0.170000	0.000000	0.000000	0.000000
25%	996.250000	0.000000	1.000000	21.000000	0.000000	0.000000	7.895800
50%	1100.500000	0.000000	3.000000	27.000000	0.000000	0.000000	14.454200
75%	1204.750000	1.000000	3.000000	39.000000	1.000000	0.000000	31.500000
max	1309.000000	1.000000	3.000000	76.000000	8.000000	9.000000	512.329200

In [76]:
```
titanic_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  418 non-null    int64
 1   Survived     418 non-null    int64
 2   Pclass       418 non-null    int64
 3   Name         418 non-null    object
 4   Sex          418 non-null    object
 5   Age          332 non-null    float64
 6   SibSp        418 non-null    int64
 7   Parch        418 non-null    int64
 8   Ticket       418 non-null    object
 9   Fare         417 non-null    float64
 10  Cabin        91 non-null     object
 11  Embarked     418 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 39.3+ KB
```

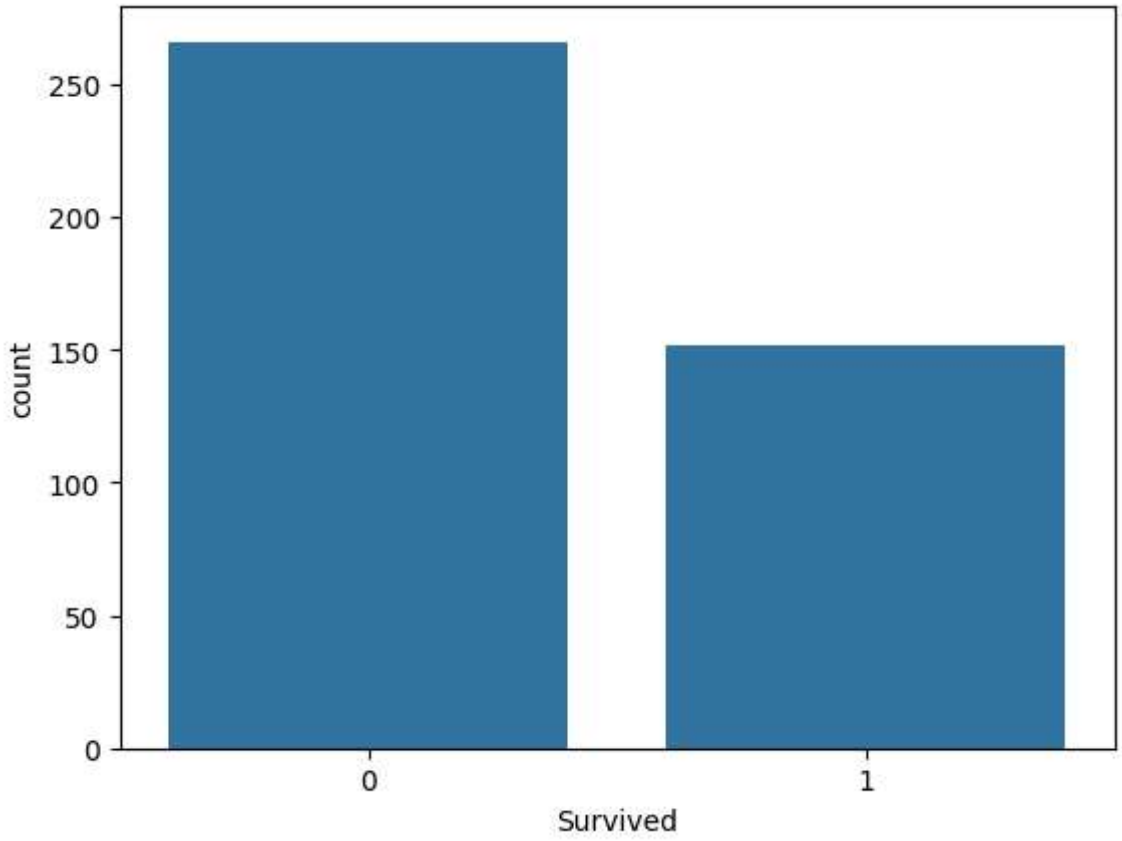
Data Analysis

In [77]:
```
#how many survived?
titanic_data['Survived'].value_counts()
```

Out[77]:
```
Survived
0    266
1    152
Name: count, dtype: int64
```

In [78]:
```
#visualizing data
sns.countplot(x='Survived',data=titanic_data)
```

Out[78]:
```
<Axes: xlabel='Survived', ylabel='count'>
```

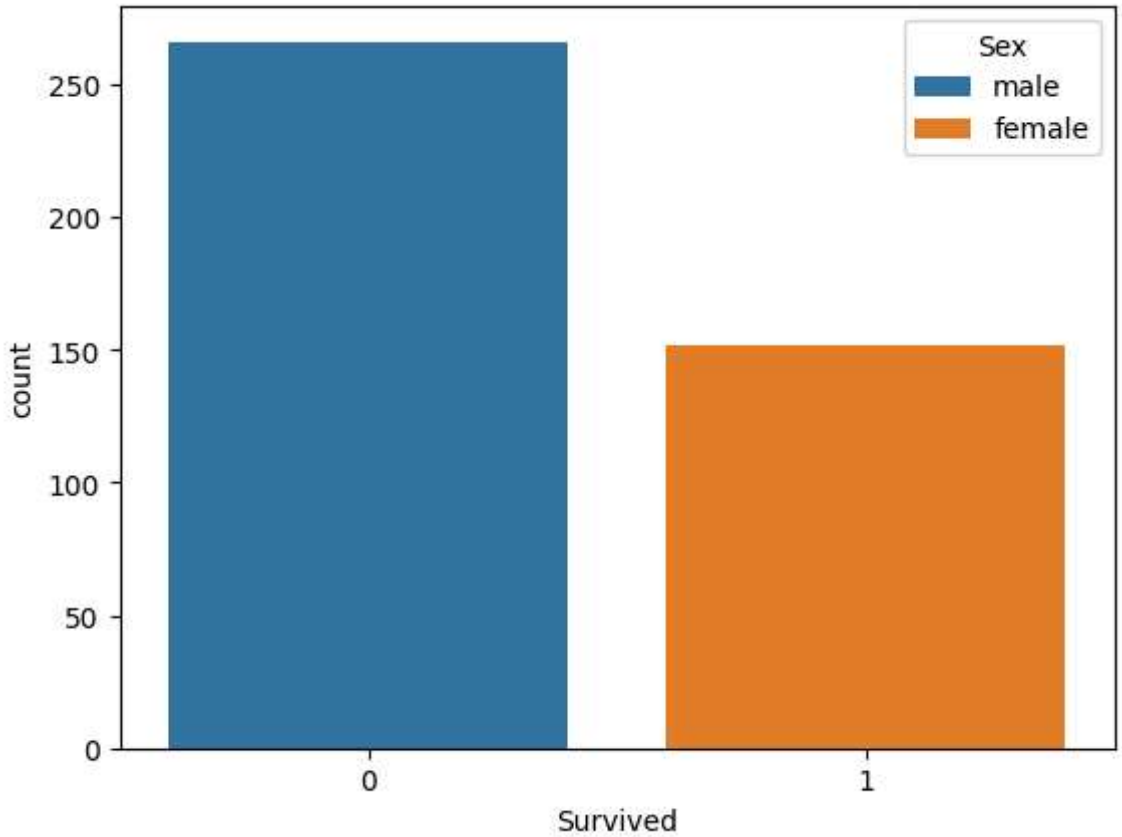


In [79]:
```
titanic_data['Sex'].value_counts()
```

Out[79]:
```
Sex
male      266
female    152
Name: count, dtype: int64
```

In [80]:
```
#Male vs female Suevived?
sns.countplot(x='Survived',data=titanic_data,hue='Sex')
```

Out[80]:
```
<Axes: xlabel='Survived', ylabel='count'>
```

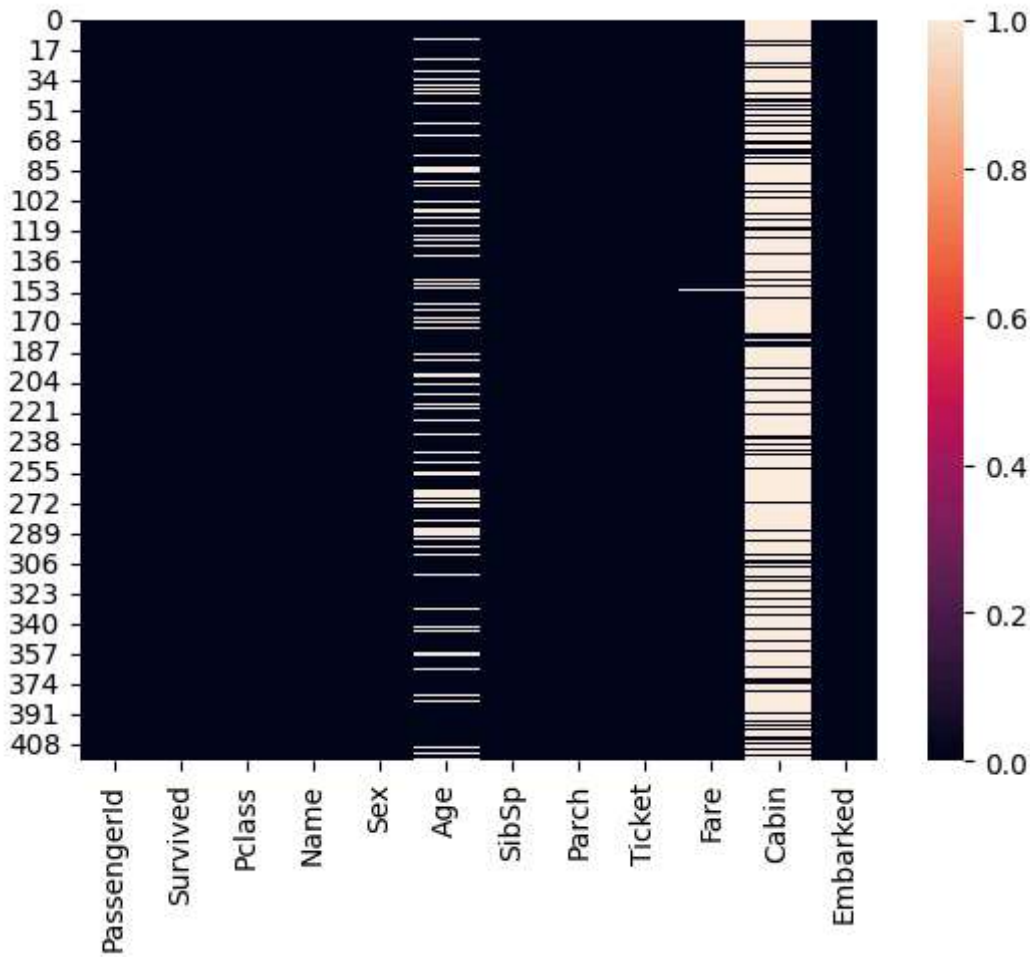


In [81]:
```
#check null values
titanic_data.isnull().sum()
```

Out[81]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
```

In [82]:
```
#visualize null values
sns.heatmap(titanic_data.isnull())
```

Out[82]: <Axes: >



In [83]:
```
#find the % of null valus in age column
(titanic_data['Age'].isnull().sum()/len(titanic_data['Age']))*100
```

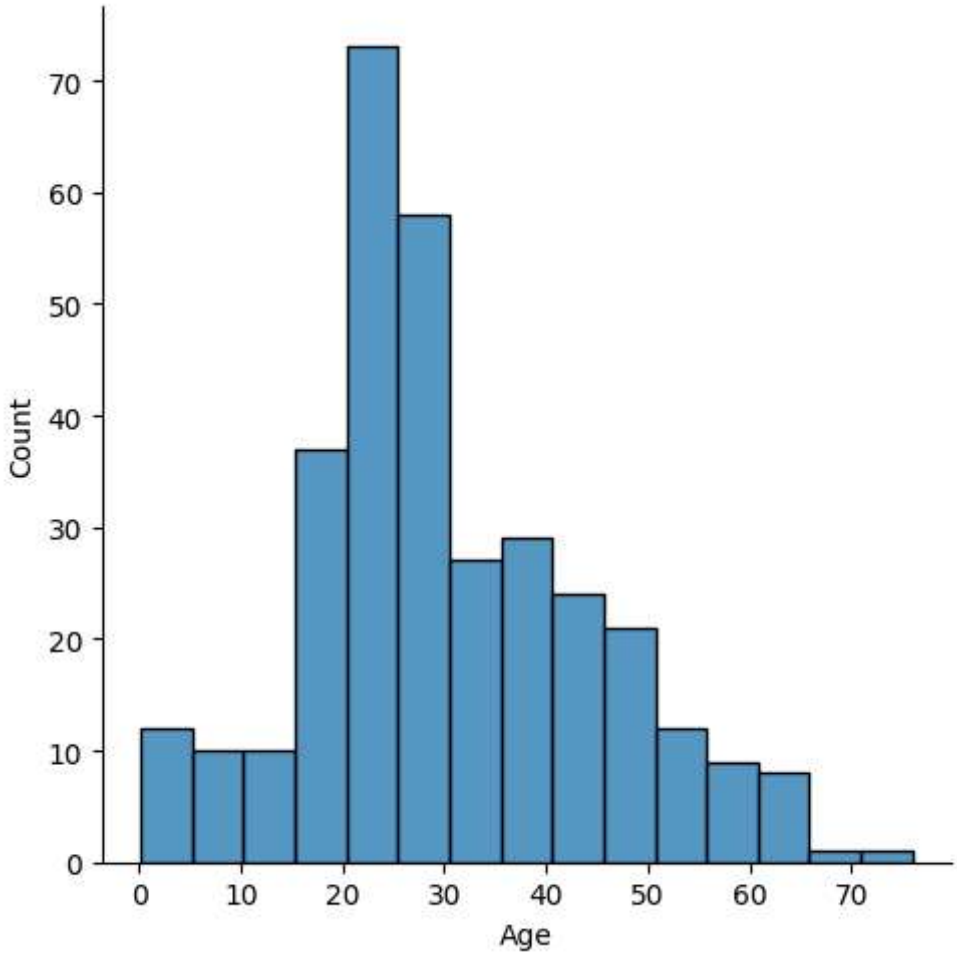
Out[83]: 20.574162679425836

In [84]:
```
#find the % of null valus in Fare column
(titanic_data['Fare'].isnull().sum()/len(titanic_data['Fare']))*100
```

Out[84]: 0.23923444976076555

In [85]:
```
# distribution of age column
sns.displot(x='Age',data=titanic_data)
```

Out[85]: <seaborn.axisgrid.FacetGrid at 0x26151a1d1c0>



Data cleaning

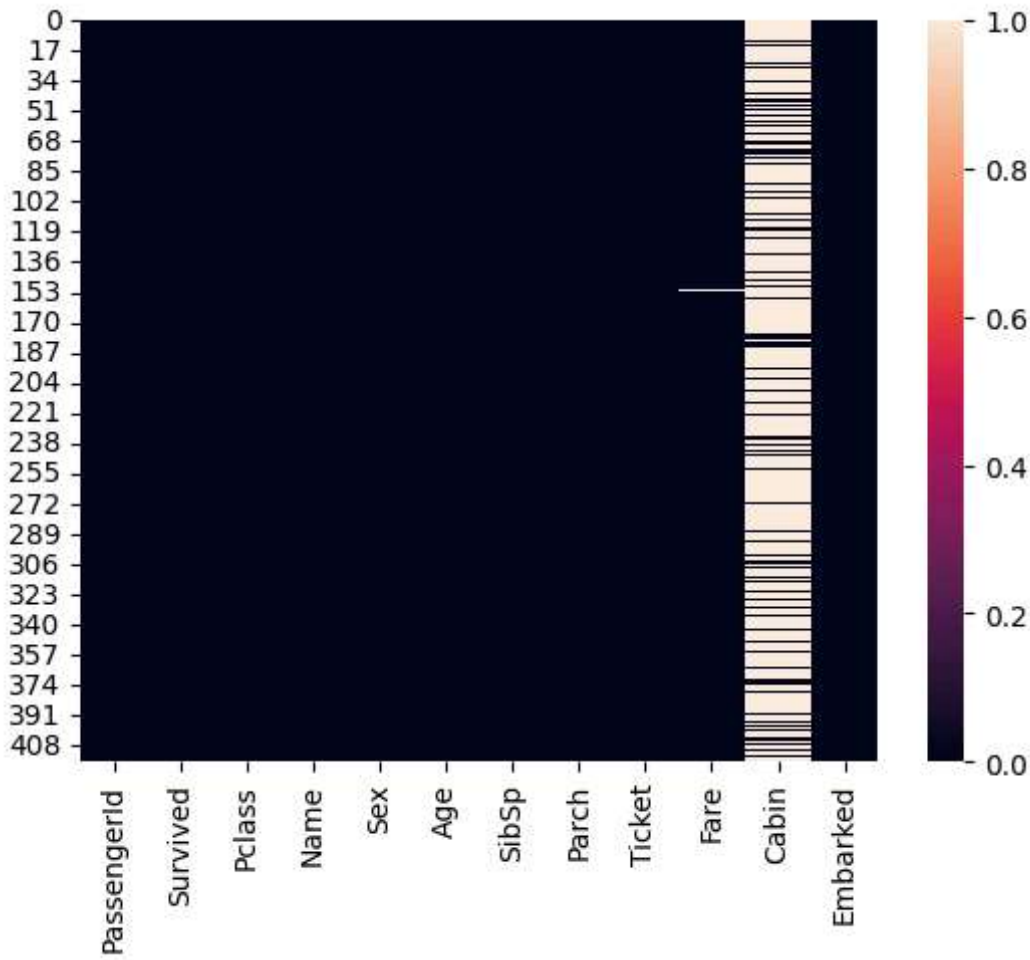
In [86]:
```
#fill age column
titanic_data['Age'].fillna(titanic_data['Age'].mean(),inplace=True )
```

In [87]:
```
#verify null value
titanic_data['Age'].isnull().sum()
```

Out[87]: 0

In [88]:
```
sns.heatmap(titanic_data.isnull())
```

Out[88]: <Axes: >



In [89]:
```
#drop fare column
titanic_data.drop('Fare', axis=1,inplace=True)
```

In [90]:
```
titanic_data.head()
```

Out[90]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Cabin	Embar
0	892	0	3	Kelly, Mr. James	male	34.5	0	0	330911	NaN	
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	NaN	
2	894	0	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	NaN	
3	895	0	3	Wirz, Mr. Albert	male	27.0	0	0	315154	NaN	
4	896	1	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	NaN	

In [91]:
```
#check for the non numeric column
gender=pd.get_dummies(titanic_data['Sex'],drop_first=True)
```

In [92]:
```
titanic_data['Gender']=gender
titanic_data.head()
```

Out[92]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Cabin	Embar
0	892	0	3	Kelly, Mr. James	male	34.5	0	0	330911	NaN	
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	NaN	
2	894	0	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	NaN	
3	895	0	3	Wirz, Mr. Albert	male	27.0	0	0	315154	NaN	
4	896	1	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	NaN	

In [93]:
```
#drop the column which are not required
titanic_data.drop(['Name','Sex','Ticket','Embarked','Cabin'],axis=1,inplace=Tr
titanic_data.head()
```

Out[93]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Gender
0	892	0	3	34.5	0	0	True
1	893	1	3	47.0	1	0	False
2	894	0	2	62.0	0	0	True
3	895	0	3	27.0	0	0	True
4	896	1	3	22.0	1	1	False

In [95]:
```
#Seperate dependent and independent variables
x=titanic_data[['PassengerId','Pclass','Age','SibSp','Parch','Gender']]
y=titanic_data['Survived']
```

In [97]:
```
y
```

Out[97]:
```
0      0
1      1
2      0
3      0
4      1
      ..
413    0
414    1
415    0
416    0
417    0
Name: Survived, Length: 418, dtype: int64
```

Data modeling

In [98]:
```
#import train test split method
from sklearn.model_selection import train_test_split
```

In [99]:
```
#train test split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, rando
```

In [101]:
```
#import logistic regression
from sklearn.linear_model import LogisticRegression
#Fit logistic regression
log_reg = LogisticRegression()
log_reg.fit(x_train, y_train)
```

```
C:\Users\ANURADHA KAR\AppData\Roaming\Python\Python39\site-packages\sklearn\l
inear_model\_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (s
tatus=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 (https://sciki
t-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-regres
sion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession)
  n_iter_i = _check_optimize_result(
```

Out[101]:
```
LogisticRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [103]:
```
#prediction
predict = log_reg.predict(x_test)
```

In [104]:
```
#import confusion matrix, classification report,
from sklearn.metrics import accuracy_score, confusion_matrix, classification_r
```

In [106]:
```
pd.DataFrame(confusion_matrix(y_test,predict),columns=['Predicted No','Predict
```

Out[106]:

	Predicted No	Predicted Yes
Actual No	50	0
Actual Yes	0	34

In [111]:
```
accuracy = accuracy_score(y_test,predict)
print("Accuracy:", accuracy)
```

```
Accuracy: 1.0
```

In [107]:
```
print(classification_report(y_test,predict))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        50
           1       1.00      1.00      1.00        34

    accuracy                           1.00        84
   macro avg       1.00      1.00      1.00        84
weighted avg       1.00      1.00      1.00        84
```