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

In [2]:
```python
df = pd.read_csv('C:\\Users\\Anirudh\\Downloads\\titanic train.csv')
df
```

Out[2]:

	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	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

In [3]:
```python
df1 = pd.read_csv('C:\\Users\\Anirudh\\Downloads\\titanic test.csv')
df1
```

Out[3]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...
413	1305	3	Spector, Mr. Woolf	male	NaN	0	0	A.5. 3236	8.0500	NaN	S
414	1306	1	Oliva y Ocana, Dona. Fermina	female	39.0	0	0	PC 17758	108.9000	C105	C
415	1307	3	Saether, Mr. Simon Sivertsen	male	38.5	0	0	SOTON/O.Q. 3101262	7.2500	NaN	S
416	1308	3	Ware, Mr. Frederick	male	NaN	0	0	359309	8.0500	NaN	S
417	1309	3	Peter, Master. Michael J	male	NaN	1	1	2668	22.3583	NaN	C

418 rows × 11 columns

In [4]:
```python
df.corr()
```

```
C:\Users\Anirudh\AppData\Local\Temp\ipykernel_15664\1134722465.py:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  df.corr()
```

Out[4]:

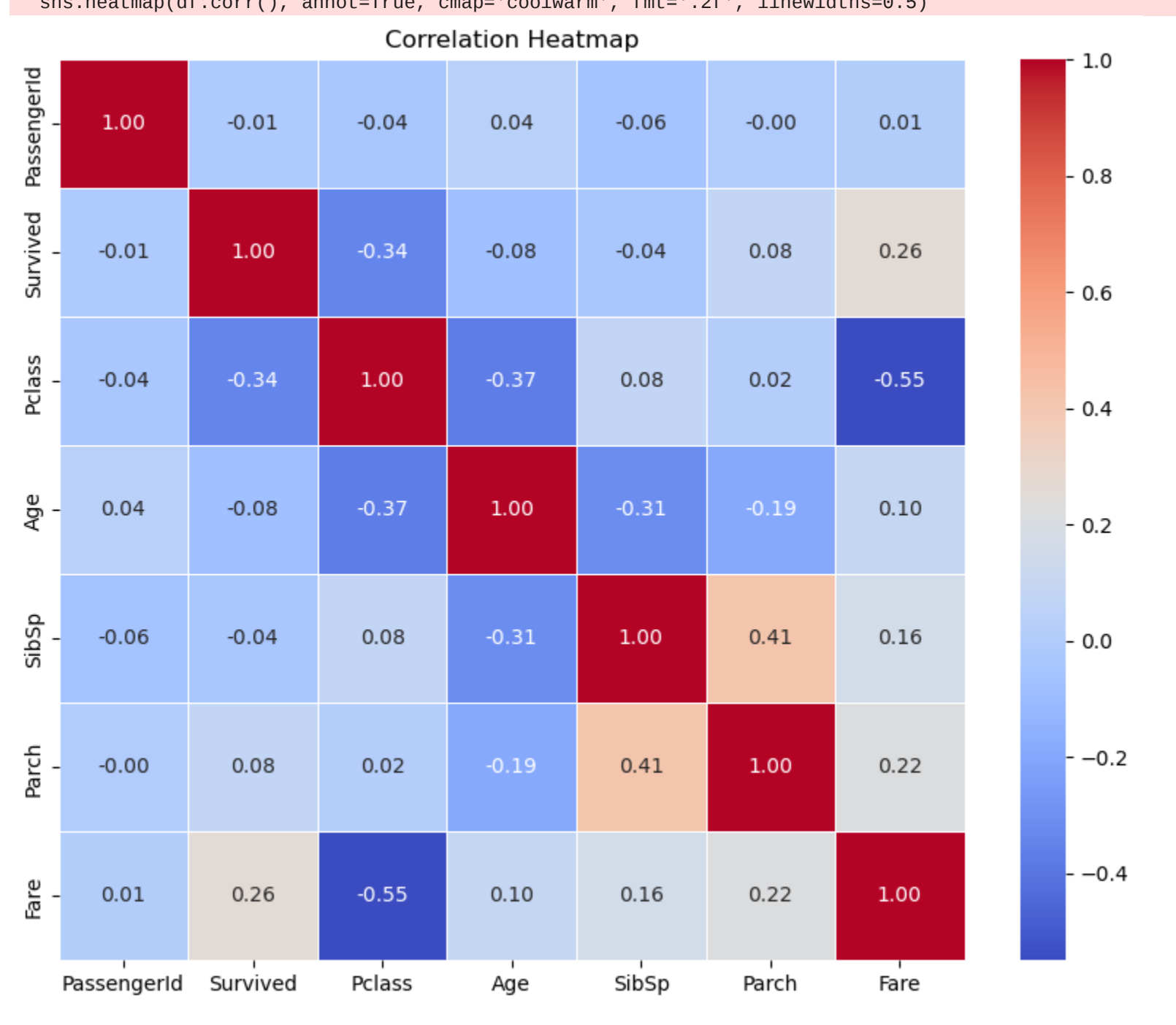
	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.005007	-0.035144	0.036847	-0.057527	-0.001652	0.012658
Survived	-0.005007	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307
Pclass	-0.035144	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.549500
Age	0.036847	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	0.096067
SibSp	-0.057527	-0.035322	0.083081	-0.308247	1.000000	0.414838	0.159651
Parch	-0.001652	0.081629	0.018443	-0.189119	0.414838	1.000000	0.216225
Fare	0.012658	0.257307	-0.549500	0.096067	0.159651	0.216225	1.000000

In [5]:
```python
plt.figure(figsize=(10, 8))  # Optional: Set the figure size
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)

# Optional: Add a title to the plot
plt.title("Correlation Heatmap")

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

```
C:\Users\Anirudh\AppData\Local\Temp\ipykernel_15664\1872074821.py:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  sns.heatmap(df.corr(), annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
```



In [6]:
```python
df['Pclass'].unique()
```

Out[6]:
```
array([3, 1, 2], dtype=int64)
```

In [7]:
```python
df['Pclass'].isna().sum()
```

Out[7]:
```
0
```

In [8]:
```python
df['Age'].isnull().sum()
```

Out[8]:
```
177
```

In [9]:
```python
df['Age'] = df['Age'].fillna(df['Age'].mean())
```

In [10]:
```python
df['Age'].isnull().sum()
```

Out[10]:
```
0
```

In [11]:
```python
df['Sex'] = df['Sex'].fillna(df['Sex'].mode())
```

In [12]:
```python
df['Sex'].isnull().sum()
```

Out[12]:
```
0
```

In [15]:
```python
df['Parch'].unique()
```

Out[15]:
```
array([0, 1, 2, 5, 3, 4, 6], dtype=int64)
```

In [16]:
```python
df['Embarked'].unique()
```

Out[16]:
```
array(['S', 'C', 'Q', nan], dtype=object)
```

In [17]:
```python
df['Embarked'] = df['Embarked'].fillna(df['Embarked'].mode())
```

In [18]:
```python
df['Fare'].isnull().sum()
```

Out[18]:
```
0
```

In [42]:
```python
df=df.drop(['PassengerId','Name','Ticket','Cabin','Parch'],axis=1)
```

In [43]:
```python
df1['Age'] = df1['Age'].fillna(df1['Age'].mean())
df1['Age'].isna().sum()
```

Out[43]:
```
0
```

In [44]:
```python
df1['Pclass'].unique()
```

Out[44]:
```
array([3, 2, 1], dtype=int64)
```

In [45]:
```python
df1['Pclass'].isna().sum()
```

Out[45]:
```
0
```

In [46]:
```python
df1['Age'].isnull().sum()
```

Out[46]:
```
0
```

In [48]:
```python
df1['Sex'].isnull().sum()
```

Out[48]:
```
0
```

In [49]:
```python
df1['Parch'].unique()
```

Out[49]:
```
array([0, 1, 3, 2, 4, 6, 5, 9], dtype=int64)
```

In [50]:
```python
df1=df1.drop(['PassengerId','Name','Ticket','Cabin','Parch'],axis=1)
```

In [51]:
```python
df1['Fare'].isnull().sum()
```

Out[51]:
```
0
```

In [53]:
```python
df1['Fare'] = df1['Fare'].fillna(df1['Fare'].mean())
```

In [54]:
```python
df1['Fare'].isnull().sum()
```

Out[54]:
```
0
```

In [55]:
```python
df1['Embarked'].isnull().sum()
```

Out[55]:
```
0
```

In [56]:
```python
from sklearn.preprocessing import LabelEncoder
```

In [57]:
```python
label_encoder = LabelEncoder()
```

In [58]:
```python
df["Sex"] = label_encoder.fit_transform(df["Sex"])
df
```

Out[58]:

	Survived	Pclass	Sex	Age	SibSp	Fare	Embarked
0	0	3	1	22.000000	1	7.2500	S
1	1	1	0	38.000000	1	71.2833	C
2	1	3	0	26.000000	0	7.9250	S
3	1	1	0	35.000000	1	53.1000	S
4	0	3	1	35.000000	0	8.0500	S
...	...	...	...	...	...	...	...
886	0	2	1	27.000000	0	13.0000	S
887	1	1	0	19.000000	0	30.0000	S
888	0	3	0	29.699118	1	23.4500	S
889	1	1	1	26.000000	0	30.0000	C
890	0	3	1	32.000000	0	7.7500	Q

891 rows × 7 columns

In [59]:
```python
df["Embarked"] = label_encoder.fit_transform(df["Embarked"])
df
```

Out[59]:

	Survived	Pclass	Sex	Age	SibSp	Fare	Embarked
0	0	3	1	22.000000	1	7.2500	2
1	1	1	0	38.000000	1	71.2833	0
2	1	3	0	26.000000	0	7.9250	2
3	1	1	0	35.000000	1	53.1000	2
4	0	3	1	35.000000	0	8.0500	2
...	...	...	...	...	...	...	...
886	0	2	1	27.000000	0	13.0000	2
887	1	1	0	19.000000	0	30.0000	2
888	0	3	0	29.699118	1	23.4500	2
889	1	1	1	26.000000	0	30.0000	0
890	0	3	1	32.000000	0	7.7500	1

891 rows × 7 columns

In [60]:
```python
df1["Embarked"] = label_encoder.fit_transform(df1["Embarked"])
df1
```

Out[60]:

	Pclass	Sex	Age	SibSp	Fare	Embarked
0	3	male	34.50000	0	7.8292	1
1	3	female	47.00000	1	7.0000	2
2	2	male	62.00000	0	9.6875	1
3	3	male	27.00000	0	8.6625	2
4	3	female	22.00000	1	12.2875	2
...	...	...	...	...	...	...
413	3	male	30.27259	0	8.0500	2
414	1	female	39.00000	0	108.9000	0
415	3	male	38.50000	0	7.2500	2
416	3	male	30.27259	0	8.0500	2
417	3	male	30.27259	1	22.3583	0

418 rows × 6 columns

In [61]:
```python
x_train = df.iloc[:,1:]
x_train
```

Out[61]:

	Pclass	Sex	Age	SibSp	Fare	Embarked
0	3	1	22.000000	1	7.2500	2
1	1	0	38.000000	1	71.2833	0
2	3	0	26.000000	0	7.9250	2
3	1	0	35.000000	1	53.1000	2
4	3	1	35.000000	0	8.0500	2
...	...	...	...	...	...	...
886	2	1	27.000000	0	13.0000	2
887	1	0	19.000000	0	30.0000	2
888	3	0	29.699118	1	23.4500	2
889	1	1	26.000000	0	30.0000	0
890	3	1	32.000000	0	7.7500	1

891 rows × 6 columns

In [62]:
```python
y_train = df.iloc[:,0]
y_train
```

Out[62]:
```
0      0
1      1
2      1
3      1
4      0
      ..
886    0
887    1
888    0
889    1
890    0
Name: Survived, Length: 891, dtype: int64
```

In [63]:
```python
y_train.unique()
```

Out[63]:
```
array([0, 1], dtype=int64)
```

In [64]:
```python
y_train.value_counts()
```

Out[64]:
```
0    549
1    342
Name: Survived, dtype: int64
```

In [65]:
```python
from imblearn.over_sampling import SMOTE

# Create SMOTE object
smote = SMOTE(random_state=42)

# Apply SMOTE to the training data
x_train_resampled, y_train_resampled = smote.fit_resample(x_train, y_train)
```

In [66]:
```python
y_train_resampled.value_counts()
```

Out[66]:
```
0    549
1    549
Name: Survived, dtype: int64
```

In [69]:
```python
df1
```

Out[69]:

	Pclass	Sex	Age	SibSp	Fare	Embarked
0	3	male	34.50000	0	7.8292	1
1	3	female	47.00000	1	7.0000	2
2	2	male	62.00000	0	9.6875	1
3	3	male	27.00000	0	8.6625	2
4	3	female	22.00000	1	12.2875	2
...	...	...	...	...	...	...
413	3	male	30.27259	0	8.0500	2
414	1	female	39.00000	0	108.9000	0
415	3	male	38.50000	0	7.2500	2
416	3	male	30.27259	0	8.0500	2
417	3	male	30.27259	1	22.3583	0

418 rows × 6 columns

In [70]:
```python
df1["Sex"] = label_encoder.fit_transform(df1["Sex"])
df1
```

Out[70]:

	Pclass	Sex	Age	SibSp	Fare	Embarked
0	3	1	34.50000	0	7.8292	1
1	3	0	47.00000	1	7.0000	2
2	2	1	62.00000	0	9.6875	1
3	3	1	27.00000	0	8.6625	2
4	3	0	22.00000	1	12.2875	2
...	...	...	...	...	...	...
413	3	1	30.27259	0	8.0500	2
414	1	0	39.00000	0	108.9000	0
415	3	1	38.50000	0	7.2500	2
416	3	1	30.27259	0	8.0500	2
417	3	1	30.27259	1	22.3583	0

418 rows × 6 columns

In [71]:
```python
df1
```

Out[71]:

	Pclass	Sex	Age	SibSp	Fare	Embarked
0	3	1	34.50000	0	7.8292	1
1	3	0	47.00000	1	7.0000	2
2	2	1	62.00000	0	9.6875	1
3	3	1	27.00000	0	8.6625	2
4	3	0	22.00000	1	12.2875	2
...	...	...	...	...	...	...
413	3	1	30.27259	0	8.0500	2
414	1	0	39.00000	0	108.9000	0
415	3	1	38.50000	0	7.2500	2
416	3	1	30.27259	0	8.0500	2
417	3	1	30.27259	1	22.3583	0

418 rows × 6 columns

In [72]:
```python
x_test = df1
x_test
```

Out[72]:

	Pclass	Sex	Age	SibSp	Fare	Embarked
0	3	1	34.50000	0	7.8292	1
1	3	0	47.00000	1	7.0000	2
2	2	1	62.00000	0	9.6875	1
3	3	1	27.00000	0	8.6625	2
4	3	0	22.00000	1	12.2875	2
...	...	...	...	...	...	...
413	3	1	30.27259	0	8.0500	2
414	1	0	39.00000	0	108.9000	0
415	3	1	38.50000	0	7.2500	2
416	3	1	30.27259	0	8.0500	2
417	3	1	30.27259	1	22.3583	0

418 rows × 6 columns

In [73]:
```python
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr = lr.fit(x_train_resampled,y_train_resampled)
y_pred = lr.predict(x_test)
y_pred
```

Out[73]:
```
array([0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0,
       1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1,
       1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1,
       1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1,
       1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 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,
       1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1,
       0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1,
       1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1,
       0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0,
       1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,
       0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1,
       0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0,
       0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0,
       1, 1, 0, 1, 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, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
       1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1,
       0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0],
      dtype=int64)
```

In [74]:
```python
from sklearn.metrics import accuracy_score

# Check the shapes of actual and predicted values
print(y_train_resampled.shape)  # Shape of actual values
print(y_pred.shape)  # Shape of predicted values

# If the shapes are different, adjust the dimensions of the predicted values
if y_train_resampled.shape != y_pred.shape:
    y_pred = y_train_resampled.squeeze()  # Remove any extra dimensions if present

# Calculate accuracy
score = accuracy_score(y_train_resampled, y_pred)
print("Accuracy:", score)
```

```
(1098,)
(418,)
Accuracy: 1.0
```

In [75]:
```python
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
dt = dt.fit(x_train_resampled,y_train_resampled)
ypr = dt.predict(x_test)
ypr
```

Out[75]:
```
array([0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1,
       1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1,
       1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1,
       1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1,
       0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1,
       1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0,
       0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0,
       1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1,
       1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1,
       0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 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, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0],
      dtype=int64)
```

In [76]:
```python
from sklearn.metrics import accuracy_score

# Check the shapes of actual and predicted values
print(y_train_resampled.shape)  # Shape of actual values
print(ypr.shape)  # Shape of predicted values

# If the shapes are different, adjust the dimensions of the predicted values
if y_train_resampled.shape != ypr.shape:
    ypr = y_train_resampled.squeeze()  # Remove any extra dimensions if present

# Calculate accuracy
score = accuracy_score(y_train_resampled, ypr)
print("Accuracy:", score)
```

```
(1098,)
(418,)
Accuracy: 1.0
```

In [77]:
```python
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators=100)
rf = rf.fit(x_train_resampled,y_train_resampled)
ypre = rf.predict(x_test)
ypre
```

Out[77]:
```
array([0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1,
       1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1,
       1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1,
       1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0,
       0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1,
       0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,
       1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0,
       0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0,
       1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1,
       0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 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, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
       0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 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, 0, 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, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0],
      dtype=int64)
```

In [78]:
```python
from sklearn.metrics import accuracy_score

# Check the shapes of actual and predicted values
print(y_train_resampled.shape)  # Shape of actual values
print(ypre.shape)  # Shape of predicted values

# If the shapes are different, adjust the dimensions of the predicted values
if y_train_resampled.shape != ypre.shape:
    ypre = y_train_resampled.squeeze()  # Remove any extra dimensions if present

# Calculate accuracy
score = accuracy_score(y_train_resampled, ypre)
print("Accuracy:", score)
```

```
(1098,)
(418,)
Accuracy: 1.0
```

In [ ]: