In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split,cross_val_score,GridSearchC\
import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
df=pd.read_csv("titanic_disaster.csv")
```

In [3]:

```
df.head()
```

Out[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabi
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	Na
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C8
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	Na
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C12
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	Na

In [4]:

```
df.describe()
```

Out[4]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [5]:

```
df.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
```

In [6]:

```
df.isnull().sum()/len(df)*100
```

Out[6]:

```
PassengerId     0.000000
Survived        0.000000
Pclass          0.000000
Name            0.000000
Sex             0.000000
Age            19.865320
SibSp           0.000000
Parch           0.000000
Ticket          0.000000
Fare            0.000000
Cabin          77.104377
Embarked        0.224467
dtype: float64
```

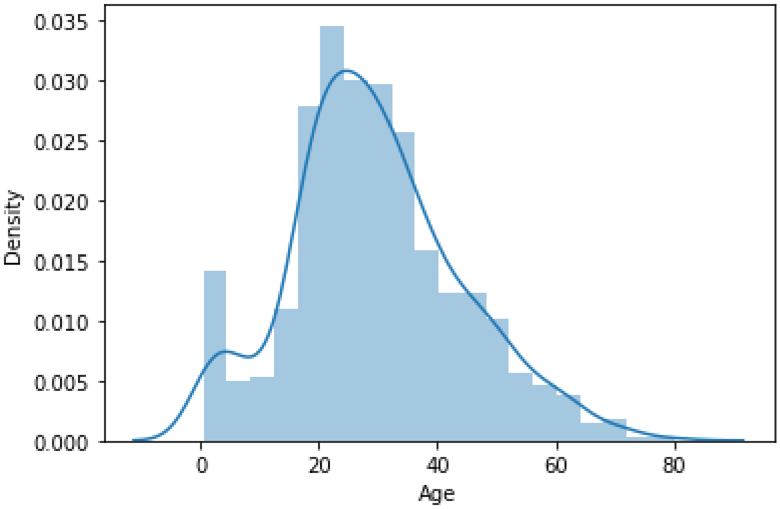
In [7]:
```
s=df['Embarked'].mode()
```

In [8]:
```
df['Embarked'].fillna(df.Embarked.mode()[0],inplace=True)
```

In [9]:
```
df.drop(['Cabin'],axis=1,inplace=True)
```

In [10]:
```
sns.distplot(df.Age)
```

Out[10]:
```
<AxesSubplot:xlabel='Age', ylabel='Density'>
```



In [11]:
```
df.Age.mean()
```

Out[11]:
```
29.69911764705882
```

In [13]:
```
df.Age.fillna(df.Age.mean(),inplace=True)
```

In [14]:
```
df.isnull().sum()
```

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

In [15]:

```
num_col=df.select_dtypes(include=['int','float']).columns
num_col
```

Out[15]:

```
Index(['PassengerId', 'Survived', 'Pclass', 'Age', 'SibSp', 'Parch', 'Fare'], d
type='object')
```

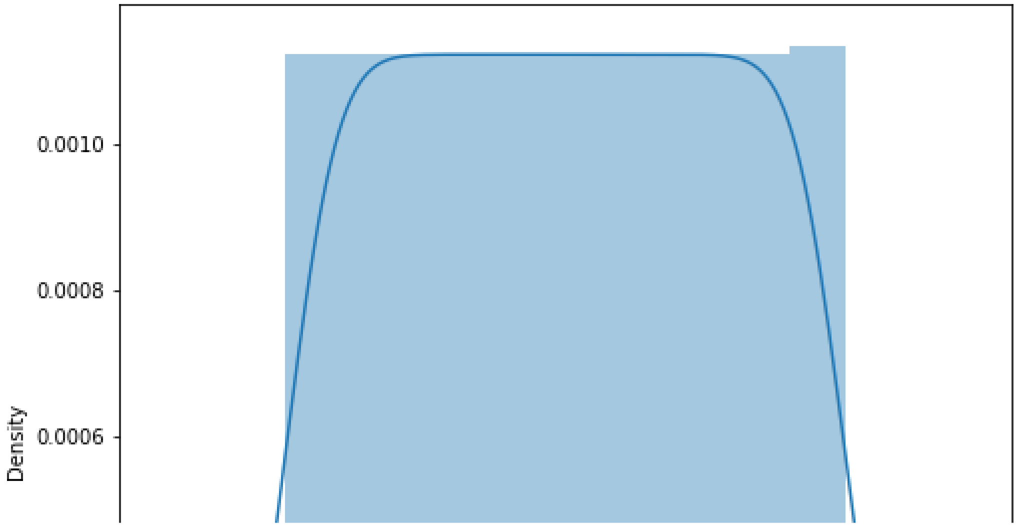
In [16]:

```
cat_col=df.select_dtypes(include='object').columns
```

In [17]:

```
for col in num_col:
    print(col)
    print("Skewness:",df[col].skew())
    print("Kurtosis",df[col].kurt())
    plt.figure(figsize=(8,8))
    sns.distplot(df[col])
    plt.show()
```

```
PassengerId
Skewness: 0.0
Kurtosis -1.1999999999999997
```

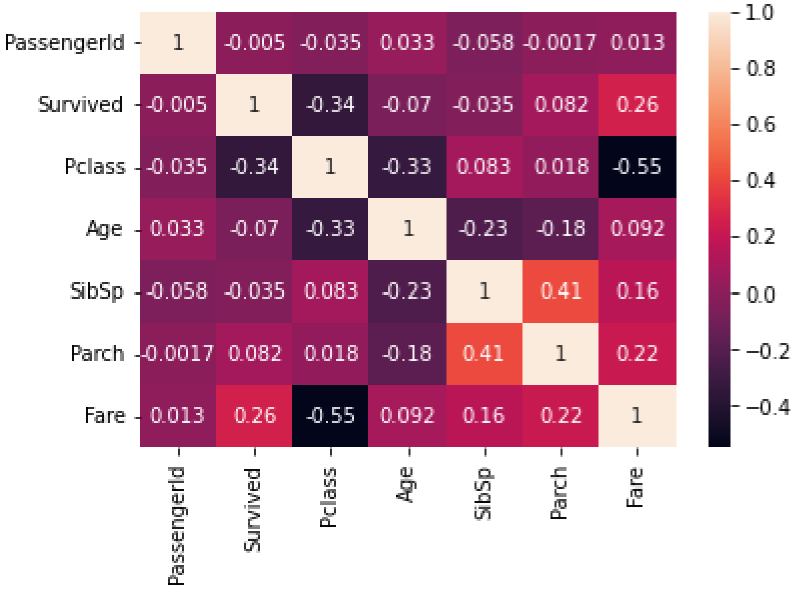


In [18]:

```
corr=df.corr()
```

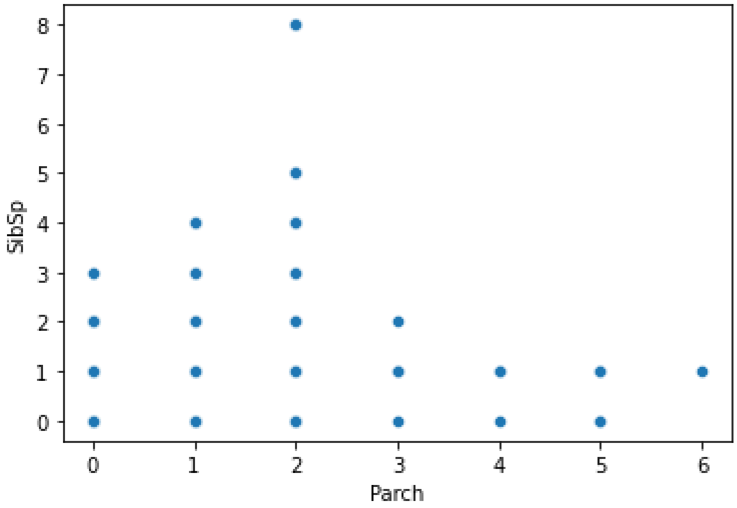
```
In [19]: sns.heatmap(corr,annot=True)
```

```
Out[19]: <AxesSubplot:>
```

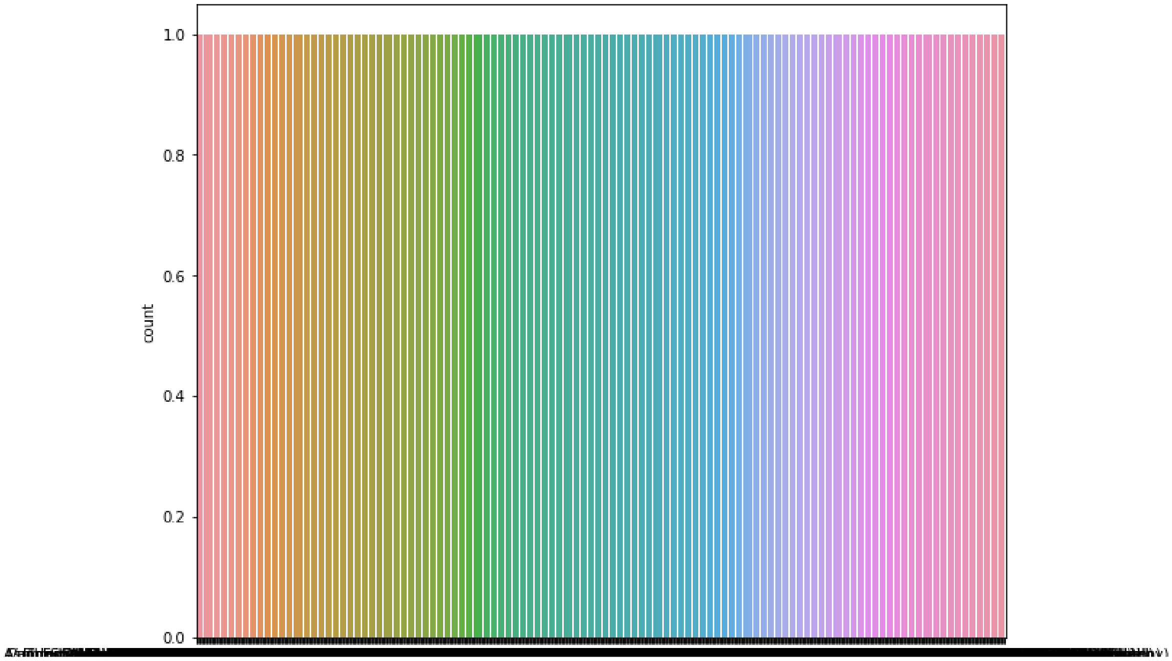


In [20]:
```
sns.scatterplot(x='Parch',y='SibSp',data=df)
```

Out[20]:
```
<AxesSubplot:xlabel='Parch', ylabel='SibSp'>
```



In [21]:
```
for i in cat_col:
    plt.figure(figsize=(10,8))
    sns.countplot(df[i])
    plt.show()
```



In [22]:
```
from sklearn.preprocessing import LabelEncoder
```

In [37]:
```
le=LabelEncoder()
```

In [38]:
```
from sklearn.preprocessing import StandardScaler
```

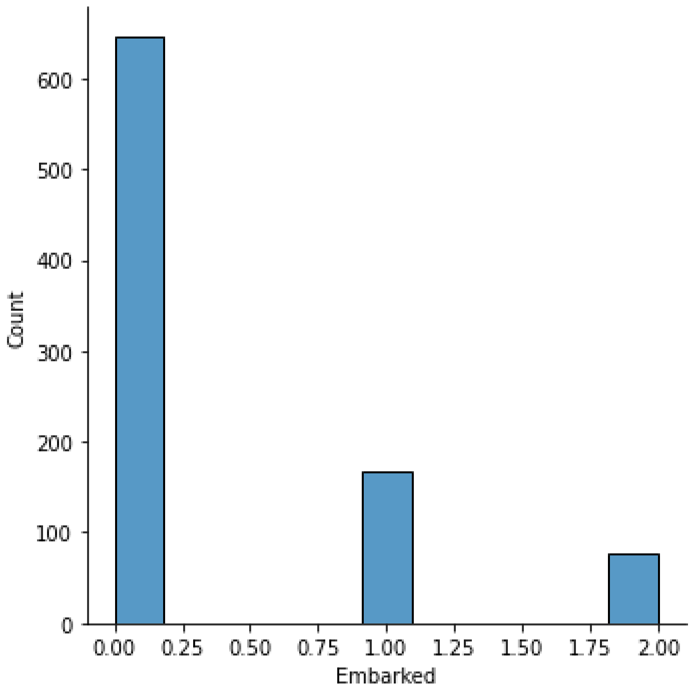
In [39]:
```
se=StandardScaler()
```

In [40]:
```
se.fit_transform(df[['Fare']])
```

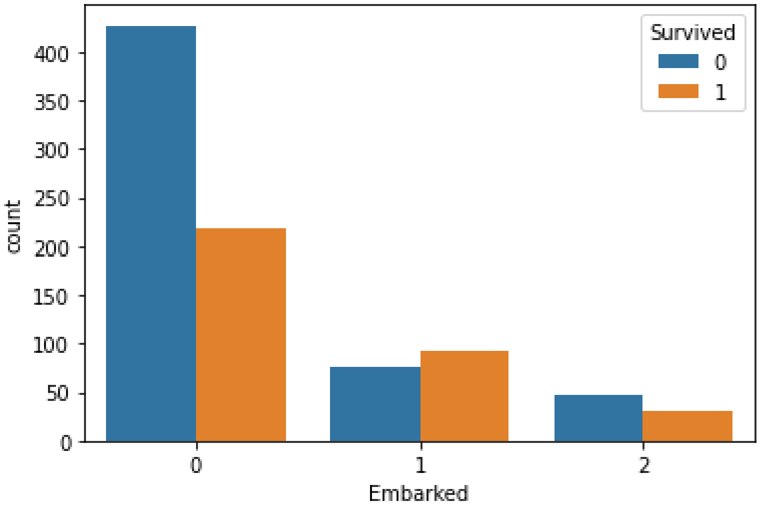
Out[40]:
```
array([[-5.02445171e-01],
       [ 7.86845294e-01],
       [-4.88854258e-01],
       [ 4.20730236e-01],
       [-4.86337422e-01],
       [-4.78116429e-01],
       [ 3.95813561e-01],
       [-2.24083121e-01],
       [-4.24256141e-01],
       [-4.29555021e-02],
       [-3.12172378e-01],
       [-1.13845709e-01],
       [-4.86337422e-01],
       [-1.87093118e-02],
       [-4.90279793e-01],
       [-3.26266659e-01],
       [-6.19988892e-02],
       [-3.86670720e-01],
       [-2.85997284e-01],
```

In [41]:
```
sns.displot(df['Embarked'])
```

Out[41]:
```
<seaborn.axisgrid.FacetGrid at 0x1e0e28bace0>
```



```
In [42]: sns.countplot('Embarked',hue='Survived' ,data=df)

Out[42]: <AxesSubplot:xlabel='Embarked', ylabel='count'>
```



```
In [43]: 
         df['Embarked'].value_counts()

Out[43]: 0    646
         1    168
         2     77
         Name: Embarked, dtype: int64
```

In [44]:
```
df
```

Out[44]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	0	22.000000	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	38.000000	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	1	26.000000	0	0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	1	35.000000	1	0	113803	53.1000
4	5	0	3	Allen, Mr. William Henry	0	35.000000	0	0	373450	8.0500
...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	0	27.000000	0	0	211536	13.0000
887	888	1	1	Graham, Miss. Margaret Edith	1	19.000000	0	0	112053	30.0000
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	1	29.699118	1	2	W./C. 6607	23.4500
889	890	1	1	Behr, Mr. Karl Howell	0	26.000000	0	0	111369	30.0000
890	891	0	3	Dooley, Mr. Patrick	0	32.000000	0	0	370376	7.7500

891 rows × 11 columns

In [45]:
```
df.replace({'Sex':{'male':0,'female':1},'Embarked':{'S':0,'C':1,'Q':2}},inplace=T
```

In [46]:

```
df.head()
```

Out[46]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarl
0	1	0	3	Braund, Mr. Owen Harris	0	22.0	1	0	A/5 21171	7.2500	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	38.0	1	0	PC 17599	71.2833	
2	3	1	3	Heikkinen, Miss. Laina	1	26.0	0	0	STON/O2. 3101282	7.9250	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	1	35.0	1	0	113803	53.1000	
4	5	0	3	Allen, Mr. William Henry	0	35.0	0	0	373450	8.0500	

In [50]:

```
df['Name']=le.fit_transform(df['Name'])
df['Ticket']=le.fit_transform(df['Ticket'])
```

In [51]:
```
df.head(20)
```

Out[51]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embark
0	1	0	3	108	0	22.000000	1	0	523	7.2500	
1	2	1	1	190	1	38.000000	1	0	596	71.2833	
2	3	1	3	353	1	26.000000	0	0	669	7.9250	
3	4	1	1	272	1	35.000000	1	0	49	53.1000	
4	5	0	3	15	0	35.000000	0	0	472	8.0500	
5	6	0	3	554	0	29.699118	0	0	275	8.4583	
6	7	0	1	515	0	54.000000	0	0	85	51.8625	
7	8	0	3	624	0	2.000000	3	1	395	21.0750	
8	9	1	3	412	1	27.000000	0	2	344	11.1333	
9	10	1	2	576	1	14.000000	1	0	132	30.0708	
10	11	1	3	727	1	4.000000	1	1	616	16.7000	
11	12	1	1	95	1	58.000000	0	0	38	26.5500	
12	13	0	3	729	0	20.000000	0	0	535	8.0500	
13	14	0	3	28	0	39.000000	1	5	333	31.2750	
14	15	0	3	840	1	14.000000	0	0	413	7.8542	
15	16	1	2	359	1	55.000000	0	0	153	16.0000	
16	17	0	3	682	0	2.000000	4	1	480	29.1250	
17	18	1	2	867	0	29.699118	0	0	151	13.0000	
18	19	0	3	839	1	31.000000	1	0	301	18.0000	
19	20	1	3	512	1	29.699118	0	0	184	7.2250	

In [53]:
```python
df
```

Out[53]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embar
0	1	0	3	108	0	22.000000	1	0	523	7.2500	
1	2	1	1	190	1	38.000000	1	0	596	71.2833	
2	3	1	3	353	1	26.000000	0	0	669	7.9250	
3	4	1	1	272	1	35.000000	1	0	49	53.1000	
4	5	0	3	15	0	35.000000	0	0	472	8.0500	
...	...	...	...	...	...	...	...	...	...	...	
886	887	0	2	548	0	27.000000	0	0	101	13.0000	
887	888	1	1	303	1	19.000000	0	0	14	30.0000	
888	889	0	3	413	1	29.699118	1	2	675	23.4500	
889	890	1	1	81	0	26.000000	0	0	8	30.0000	
890	891	0	3	220	0	32.000000	0	0	466	7.7500	

891 rows × 11 columns

In [58]:
```python
X = df.drop(columns=['PassengerId', 'Ticket', 'Survived'], axis=1)
y = df['Survived']
```

In [59]:
```python
from sklearn.model_selection import train_test_split
```

In [60]:
```python
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.20,random_state=12
```

In [61]:
```python
X_train.shape
```

Out[61]:
```
(712, 8)
```

In [62]:
```python
y_train.shape
```

Out[62]:
```
(712,)
```

In [63]:
```python
X_test.shape
```

Out[63]:
```
(179, 8)
```

In [64]:
```python
y_test.shape
```

Out[64]:
```
(179,)
```

In [71]:
```
from sklearn.metrics import accuracy_score
```

In [72]:
```
from sklearn.linear_model import LogisticRegression
```

In [73]:
```
clf=LogisticRegression()
```

In [82]:
```
clf.fit(X_train,y_train)
```

Out[82]:
```
▾ LogisticRegression
LogisticRegression()
```

In [83]:
```
print("Training Performance")
print(accuracy_score(y_train,y_train_pred))
print("Testing Performance")
print(accuracy_score(y_test,y_test_pred))
```

```
Training Performance
0.7865168539325843
Testing Performance
0.8156424581005587
```

In [84]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [85]:
```
#Model 1:Base Model
rf=RandomForestClassifier()
```

In [86]:
```
rf.fit(X_train,y_train)
```

Out[86]:
```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [87]:
```
#Prediting train and test data
y_pred_train=rf.predict(X_train)
y_pred_test=rf.predict(X_test)
```

In [88]:
```
y_pred_train
```

Out[88]:
```
array([1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1,
       1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0,
       1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0,
       1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0,
       1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0,
       1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0,
       1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1,
       0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1,
       0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0,
       1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1,
       0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1,
       1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0,
       0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0,
       1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0,
       1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1,
       1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0,
       1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0,
       1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0,
       0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1,
       0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1,
       0, 1, 1, 1, 1, 0, 0, 1], dtype=int64)
```

In [89]:
```
print("Train Data")
print(accuracy_score(y_train,y_pred_train))
print("Test Data")
print(accuracy_score(y_test,y_pred_test))
```

```
Train Data
1.0
Test Data
0.8435754189944135
```

In [90]:
```
#model 2
rf1=RandomForestClassifier(n_estimators=100,criterion="entropy",max_depth=12,min_
rf1.fit(X_train,y_train)
```

Out[90]:
```
▾                        RandomForestClassifier

RandomForestClassifier(criterion='entropy', max_depth=12, min_samples_split=1
0,
                       random_state=123)
```

In [91]:
```
#Prediting train and test data
y_pred_train=rf1.predict(X_train)
y_pred_test=rf1.predict(X_test)
```

In [92]:
```
print("Train Data")
print(accuracy_score(y_train,y_pred_train))
print("Test Data")
print(accuracy_score(y_test,y_pred_test))
```

```
Train Data
0.9213483146067416
Test Data
0.8659217877094972
```

In [93]:
```
param_grid={
    "n_estimators":[50,100],
    "criterion":["gini","entropy"],
    "max_depth":np.arange(1,10),
    "min_samples_split":[5,10,15,20,25,30,35,40],
    }
```

In [98]:
```
model=GridSearchCV(rf1,param_grid=param_grid,cv=5,scoring="accuracy",n_jobs=-1)
model.fit(X_train,y_train)
```

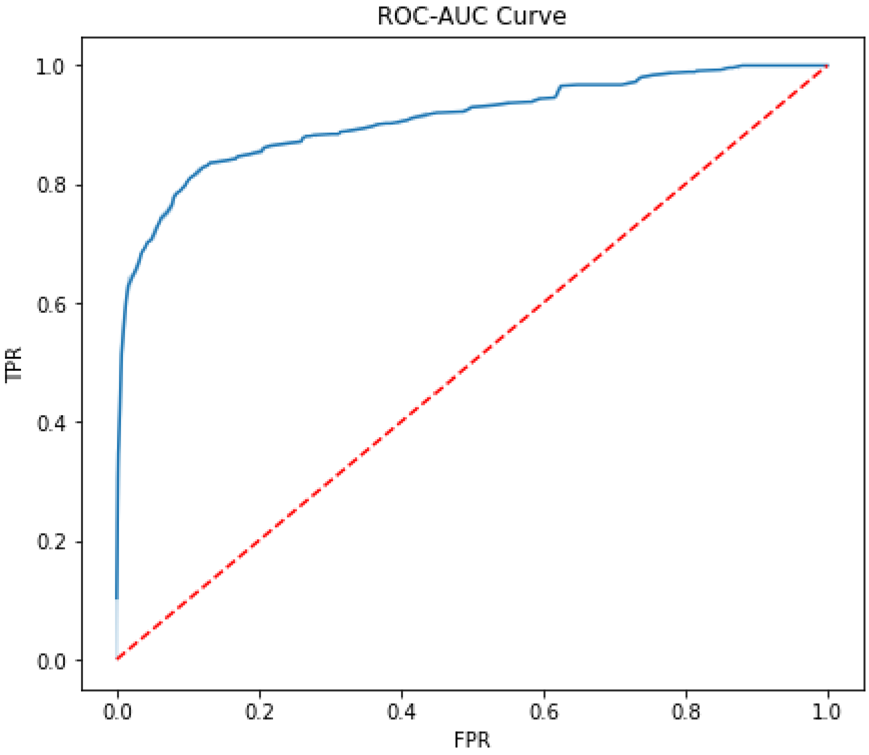
Out[98]:
```
▸                          GridSearchCV
▸               estimator: RandomForestClassifier
▾                     RandomForestClassifier

RandomForestClassifier(criterion='entropy', max_depth=12, min_samples_split=
10,
                       random_state=123)
```

In [99]:
```
train_proba=model.predict_proba(X_train)[:,1]
test_proba=model.predict_proba(X_test)[:,1]
```

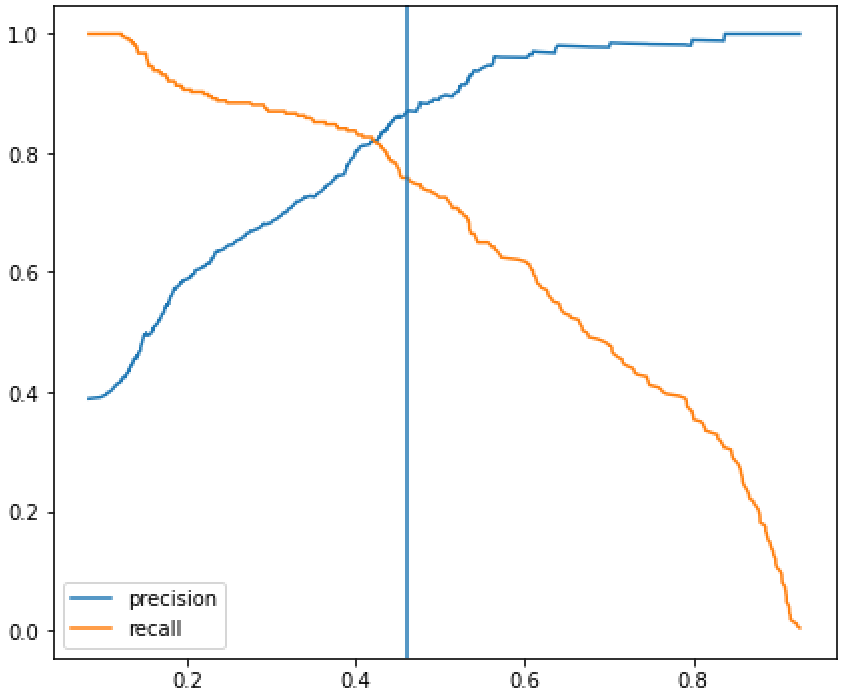
In [101]:
```
from sklearn import metrics
```

In [102]:

```
fpr,tpr,th=metrics.roc_curve(y_train,train_proba)
plt.figure(figsize=(7,6))
sns.lineplot(x=fpr,y=tpr)
sns.lineplot(x=[0.0,1],y=[0.0,1],color='red',linestyle="--")
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.title("ROC-AUC Curve")
plt.show()
```



In [103]:
```
p,r,th=metrics.precision_recall_curve(y_train,train_proba)
plt.figure(figsize=(7,6))
sns.lineplot(x=th,y=p[:-1],label="precision")
sns.lineplot(x=th,y=r[:-1],label='recall')
plt.axvline(0.46)
plt.show()
```



In [107]:
```
#model 2
rf2=RandomForestClassifier(n_estimators=100,criterion="entropy",max_depth=8,min_s
rf2.fit(X_train,y_train)
```

Out[107]:
```
▾                          RandomForestClassifier
RandomForestClassifier(criterion='entropy', max_depth=8, min_samples_split=10,
                       random_state=123)
```

In [108]:
```
y_pred_train=rf2.predict(X_train)
y_pred_test=rf2.predict(X_test)
```

In [109]:
```
print("Train Data")
print(accuracy_score(y_train,y_pred_train))
print("Test Data")
print(accuracy_score(y_test,y_pred_test))
```

```
Train Data
0.8974719101123596
Test Data
0.8659217877094972
```

In [ ]: