In [1]:
```
# import libraries
```

In [2]:
```
import pandas as pd
import numpy as np
```

In [3]:
```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [4]:
```
from sklearn.model_selection import train_test_split
```

In [5]:
```
from sklearn.linear_model import LogisticRegression
```

In [6]:
```
# Load data
```

In [7]:
```
titanic=pd.read_csv('train.csv')
```

In [8]:
```
titanic.head()
```

Out[8]:

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN

In [9]:
```
titanic.columns
```

Out[9]:
```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Gender', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

In [10]:
```
titanic.dtypes
```

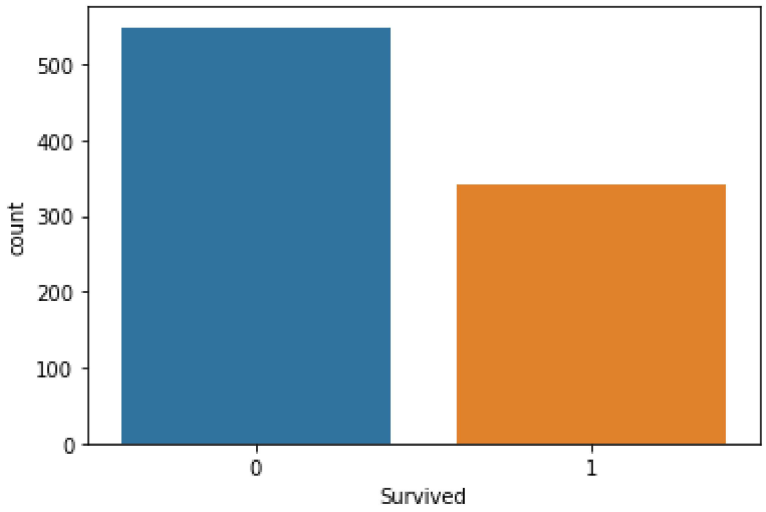
Out[10]:
```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Gender          object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
dtype: object
```

In [11]:
```
#Explaining Dataset
#survival : Survival 0 = No, 1 = Yes
#pclass : Ticket class 1 = 1st, 2 = 2nd, 3 = 3rd
#Gender : gender
#Age : Age in years
#sibsp : Number of siblings / spouses aboard the Titanic
#parch # of parents / children aboard the Titanic
#ticket : Ticket number fare Passenger fare cabin Cabin number
#embarked : Port of Embarkation C = Cherbourg, Q = Queenstown, S = Southampton
```

In [12]:
```
#Data analysis
```

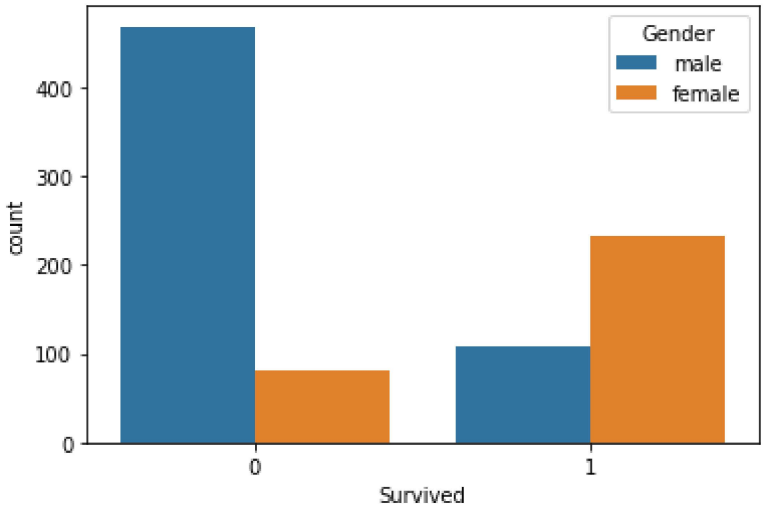
In [13]:
```
sns.countplot(x='Survived', data=titanic)
```

Out[13]:
```
<AxesSubplot:xlabel='Survived', ylabel='count'>
```



In [14]:
```
#male vs female survived
sns.countplot(x='Survived', data=titanic, hue='Gender')
```

Out[14]:
```
<AxesSubplot:xlabel='Survived', ylabel='count'>
```

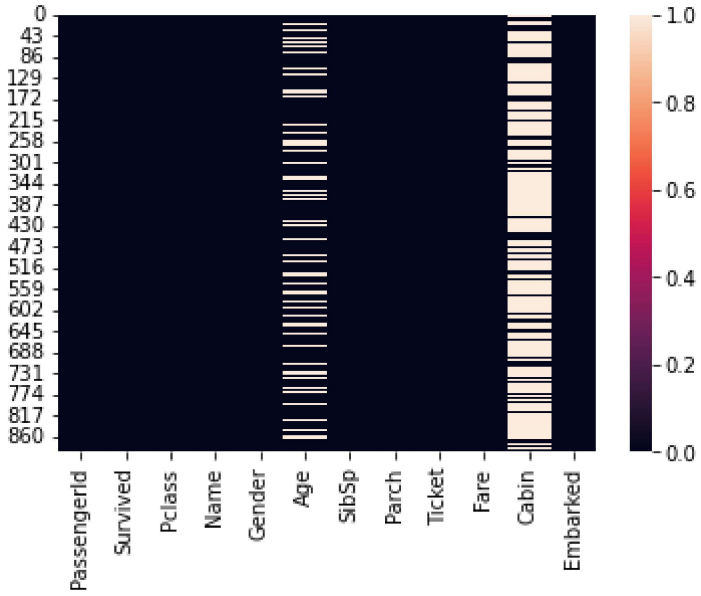


In [15]:
```
#check for null value
titanic.isnull().sum()
```

Out[15]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Gender           0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

In [16]:
```
#Visualize null values
sns.heatmap(titanic.isnull())
```

Out[16]:
```
<AxesSubplot:>
```



In [17]:
```
#find the % of null values in age column
```

```
(titanic['Age'].isna().sum()/len(titanic['Age']))*100
```

Out[17]: 19.865319865319865

In [18]:
```
#find the % of null values in Cabin column
(titanic['Cabin'].isna().sum()/len(titanic['Cabin']))*100
```

Out[18]: 77.10437710437711

In [19]:
```
#Data Cleaning
#Fill the missing values
#we will fill the missing values for age. In order to fill missing values we use mean
#For now we will fill the missing age by taking average of all age
```

In [20]:
```
titanic['Age'].fillna(titanic['Age'].mean(), inplace=True)
```

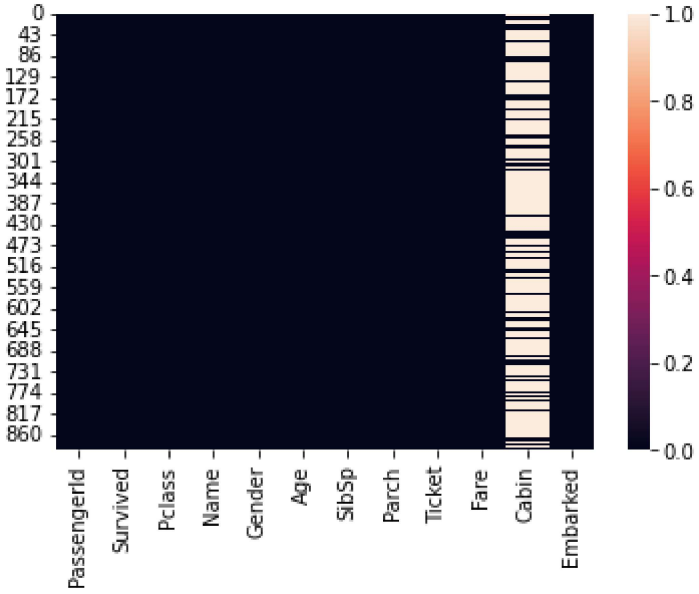
In [21]:
```
#We can verify that no more null data exist
#we will examine data by isnull mehtod which will return nothing
#verify null value
```

In [22]:
```
titanic['Age'].isna().sum()
```

Out[22]: 0

In [23]:
```
#Alternatively, visualize null values
sns.heatmap(titanic.isna())
```

Out[23]: <AxesSubplot:>



In [24]:
```
#It can be seen that the Cabin Column has huge no. of null values.
#Thankfully, Cabin info doesnt affect the Survival of pasengers and hence this column
```

In [25]:
```
#Drop cabin column
titanic.drop('Cabin', axis=1, inplace=True)
```

In [26]:
```
#see the contents of the data
titanic.head()
```

Out[26]:

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

In [27]:
```
#Remove remaining null values
titanic.dropna(inplace=True)
```

In [28]:
```
titanic.isnull().sum()
```

Out[28]:
```
PassengerId    0
Survived       0
Pclass         0
Name           0
Gender         0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       0
dtype: int64
```

In [29]:
```
#Preaparing Data for Model
#We need to convert all non-numerical columns to numeric. Please note this is required
```

In [30]:
```
titanic.dtypes
```

Out[30]:
```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Gender          object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Embarked        object
dtype: object
```

In [31]:
```
#We can see, Name, Gender, Ticket and Embarked are non-numerical.
#Name,Embarked and Ticket  are not useful for Machine Learning Prediction
#For Now we would convert Gender Column to dummies numerical values**
```

In [32]:
```
gender=pd.get_dummies(titanic['Gender'], drop_first=True)
```

In [33]:
```
titanic['Gender']=gender
```

In [34]:
```
titanic.drop(['Embarked','PassengerId','Name','Ticket'], axis=1, inplace=True)
```

In [35]:
```
titanic.head()
```

Out[35]:

	Survived	Pclass	Gender	Age	SibSp	Parch	Fare
0	0	3	1	22.0	1	0	7.2500
1	1	1	0	38.0	1	0	71.2833
2	1	3	0	26.0	0	0	7.9250
3	1	1	0	35.0	1	0	53.1000
4	0	3	1	35.0	0	0	8.0500

In [36]:
```
#Seperate Dependent and Independent variables
```

In [37]:
```
x=titanic[['Pclass','Gender','Age','SibSp','Parch','Fare']]
```

In [38]:
```
y=titanic['Survived']
```

In [39]:
```
x
```

Out[39]:

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

889 rows × 6 columns

In [40]:
```
y
```

Out[40]:
```
0      0
1      1
2      1
3      1
4      0
      ..
886    0
887    1
888    0
889    1
890    0
Name: Survived, Length: 889, dtype: int64
```

In [41]:
```
#Data Modelling
#Building Model using Logistic Regression
```

In [42]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3, random_state=3)
```

In [43]:
```
#Fit Logistic Regression
lr=LogisticRegression()
lr.fit(x_train,y_train)
```

Out[43]:
```
LogisticRegression()
```

In [44]:
```
LogisticRegression?
```

In [65]:
```
#predict
predict=lr.predict(x_test)
```

In [66]:
```
predict
```

Out[66]:
```
array([1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1,
       0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0,
       1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0,
       1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1,
       1, 1, 0], dtype=int64)
```

In [47]:
```
#Testing
#See how our model is performing
```

In [48]:
```
#print confusion matrix
from sklearn.metrics import confusion_matrix
pd.DataFrame(confusion_matrix(y_test,predict),columns=['Predicted No','Predicted Yes']
```

Out[48]:

	Predicted No	Predicted Yes
0	141	17
1	47	62

In [49]:
```
from sklearn.metrics import accuracy_score
```

In [50]:
```
accuracy_score(y_test,predict)
```

Out[50]:
```
0.7602996254681648
```

In [60]:
```
from sklearn.metrics import roc_auc_score
```

In [72]:
```
auc = roc_auc_score(y_test,predict)
```

In [73]:
```
auc
```

Out[73]:
```
0.7306062013703403
```

In [ ]:
```
# Visualisation of ROC_AUC with plot_metric
```

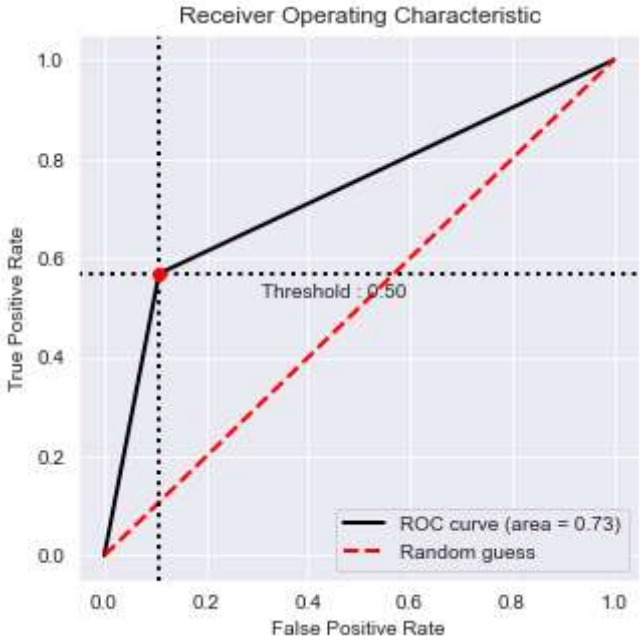
In [77]:
```
!pip install plot_metric
```

```
Collecting plot_metric
  Downloading plot_metric-0.0.6-py3-none-any.whl (13 kB)
Requirement already satisfied: scipy>=1.1.0 in c:\users\karthik a v\anaconda\lib\site
-packages (from plot_metric) (1.7.3)
Requirement already satisfied: numpy>=1.15.4 in c:\users\karthik a v\anaconda\lib\sit
e-packages (from plot_metric) (1.21.5)
Requirement already satisfied: pandas>=0.23.4 in c:\users\karthik a v\anaconda\lib\si
te-packages (from plot_metric) (1.4.2)
Requirement already satisfied: seaborn>=0.9.0 in c:\users\karthik a v\anaconda\lib\si
te-packages (from plot_metric) (0.11.2)
Collecting colorlover>=0.3.0
  Downloading colorlover-0.3.0-py3-none-any.whl (8.9 kB)
Requirement already satisfied: scikit-learn>=0.21.2 in c:\users\karthik a v\anaconda
\lib\site-packages (from plot_metric) (1.0.2)
Requirement already satisfied: matplotlib>=3.0.2 in c:\users\karthik a v\anaconda\lib
\site-packages (from plot_metric) (3.5.1)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\karthik a v\anaconda\lib
\site-packages (from matplotlib>=3.0.2->plot_metric) (3.0.4)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\karthik a v\anaconda
\lib\site-packages (from matplotlib>=3.0.2->plot_metric) (2.8.2)
Requirement already satisfied: packaging>=20.0 in c:\users\karthik a v\anaconda\lib\s
ite-packages (from matplotlib>=3.0.2->plot_metric) (21.3)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\karthik a v\anaconda\lib
\site-packages (from matplotlib>=3.0.2->plot_metric) (1.3.2)
Requirement already satisfied: cycler>=0.10 in c:\users\karthik a v\anaconda\lib\site
-packages (from matplotlib>=3.0.2->plot_metric) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\karthik a v\anaconda\lib
\site-packages (from matplotlib>=3.0.2->plot_metric) (4.25.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\karthik a v\anaconda\lib\sit
e-packages (from matplotlib>=3.0.2->plot_metric) (9.0.1)
Requirement already satisfied: pytz>=2020.1 in c:\users\karthik a v\anaconda\lib\site
-packages (from pandas>=0.23.4->plot_metric) (2021.3)
Requirement already satisfied: six>=1.5 in c:\users\karthik a v\anaconda\lib\site-pac
kages (from python-dateutil>=2.7->matplotlib>=3.0.2->plot_metric) (1.16.0)
Requirement already satisfied: joblib>=0.11 in c:\users\karthik a v\anaconda\lib\site
-packages (from scikit-learn>=0.21.2->plot_metric) (1.1.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\karthik a v\anaconda
\lib\site-packages (from scikit-learn>=0.21.2->plot_metric) (2.2.0)
Installing collected packages: colorlover, plot-metric
Successfully installed colorlover-0.3.0 plot-metric-0.0.6
```

In [79]:
```
from plot_metric.functions import BinaryClassification
```

In [83]:
```
# Visualisation with plot_metric
bc = BinaryClassification(y_test,predict, labels=['c1','c2'])

# Figures
plt.figure(figsize=(5,5))
bc.plot_roc_curve()
plt.show()
```


Receiver Operating Characteristic
True Positive Rate
False Positive Rate
Threshold : 0.50
ROC curve (area = 0.73)
Random guess
0.0
0.2
0.4
0.6
0.8
1.0

In [ ]: