NAME - ARGHYADIP GHOSH

UID - 18BCS6081

SECTION - AIML-1 (GROUP-A)

CASE STUDY - 2 (TITANIC CHALLENGE)

In [285]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [286]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [287]:

```
# importing the train dataset
train=pd.read_csv("train.csv")
```

In [288]:

```
#importing the test dataset
test=pd.read_csv("test.csv")
```

In [289]:

```
train
```

Out[289]:

	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 [290]:

```
test
```

Out[290]:

	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

Let us consider all the attributes of the train dataset as the Target varriable:

In [291]:

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
PassengerId    891 non-null int64
Survived       891 non-null int64
Pclass         891 non-null int64
Name           891 non-null object
Sex            891 non-null object
Age            714 non-null float64
SibSp          891 non-null int64
Parch          891 non-null int64
Ticket         891 non-null object
Fare           891 non-null float64
Cabin          204 non-null object
Embarked       889 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 66.2+ KB
```

Checking for Outliers:

Outliers are extreme values that deviate from other observations on data , they may indicate a variability in a measurement, experimental errors or a novelty. In other words, an outlier is an observation that diverges from an overall pattern on a sample.

In [292]:

```
train.describe(percentiles=[.25,.2,.75,.90,.95,.99])
```

Out[292]:

	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

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
20%	179.000000	0.000000	1.000000	19.000000	0.000000	0.000000	7.854200
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
90%	802.000000	1.000000	3.000000	50.000000	1.000000	2.000000	77.958300
95%	846.500000	1.000000	3.000000	56.000000	3.000000	2.000000	112.079150
99%	882.100000	1.000000	3.000000	65.870000	5.000000	4.000000	249.006220
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

From the distribution shown above , we can see that there are no outliers in our data as the numbers are gradually increasing.

In [293]:

```
train.shape
```

Out[293]:

```
(891, 12)
```

Lets check whether any columns contain any NULL values

In [294]:

```
train.isnull().sum()
```

Out[294]:

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

It means that there are in total three columns named as : "Age" , "Cabin" , "Embarked" which has NULL values. so we will perform data cleaning one by one.

DATA PREPERATION

Survival as the target varriable

In [295]:

```
# here we will evaluate the survival ratio of the assengers on Titanic
# 0 means died
# 1 means survived
train.Survived.value_counts()
```

Out[295]:

```
0    549
1    342
Name: Survived, dtype: int64
```

In [296]:

```
# here we will find the survival percentage
train.Survived.value_counts(normalize=True)*100
```

Out[296]:

```
0    61.616162
1    38.383838
Name: Survived, dtype: float64
```

It means that there are total 549+342=891 passengers and only 342 (38%) survived and rest all have died.

In [297]:

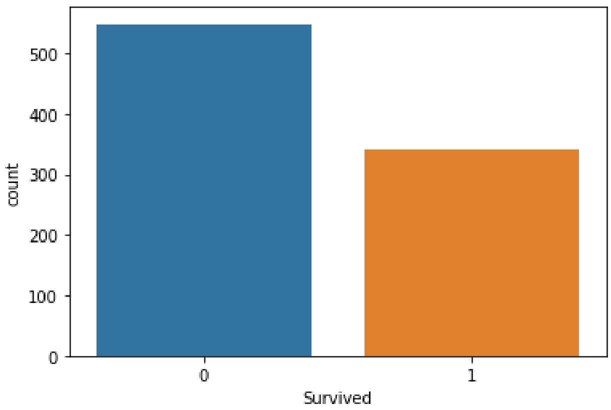
```
import seaborn as sns
```

In [298]:

```
sns.countplot(x="Survived",data=train)
```

Out[298]:

```
<matplotlib.axes._subplots.AxesSubplot at 0xd7d10f0>
```



Pclass as the target varriable:

In [299]:

```
# here we will evaluate the survivval ratio in comparison to Pclass of the
# people travelling
pd.crosstab(train.Pclass,train.Survived)
```

Out[299]:

Survived	0	1
Pclass		
1	80	136
2	97	87
3	372	119

In [300]:

```
train[["Pclass","Survived"]].groupby("Pclass").mean()*100
```

Out[300]:

Survived

Survived	Survived
Pclass	
1	62.962963
2	47.282609
3	24.236253

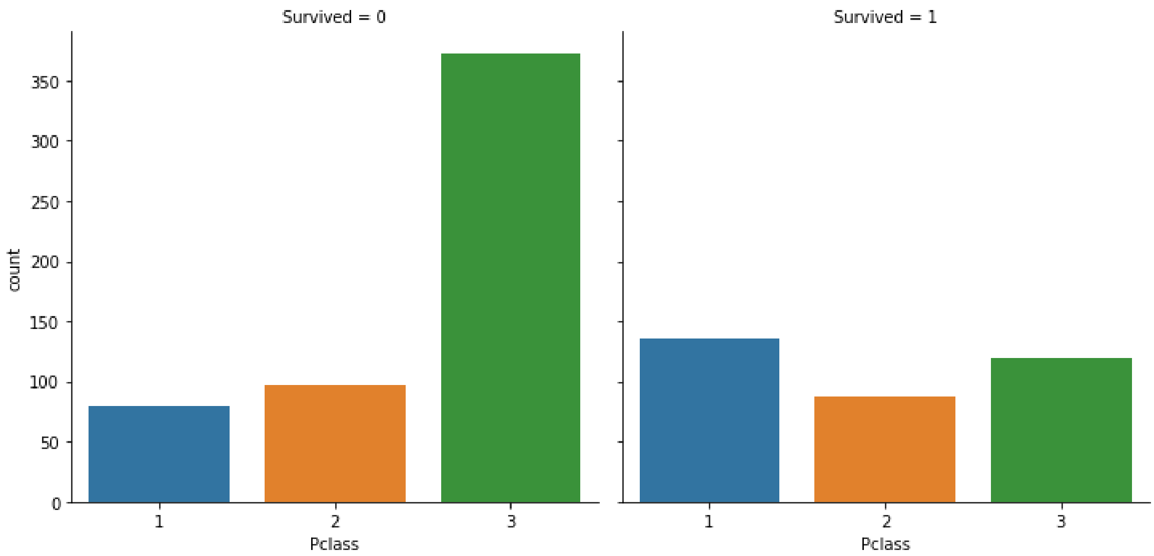
From this data , we can infer that there is 62.96% Survival chance for the people travelling in 1st class. It means that the people in the 3rd class are given the least priority and have died the most.

In [301]:

```
sns.factorplot(x="Pclass",data=train,col="Survived",kind="count")
```

Out[301]:

```
<seaborn.axisgrid.FacetGrid at 0xd7c6dd0>
```



Sex as the target varriable:

In [302]:

```
# here we will evaluate the survival ratio in comparison to the sex of
# the people
pd.crosstab(train.Sex,train.Survived)
```

Out[302]:

Survived	0	1
Sex		
female	81	233
male	468	109

In [303]:

```
pd.crosstab(train.Sex,train.Survived,normalize="index")*100
```

Out[303]:

Survived	0	1
Sex		
female	25.796178	74.203822

Survived	0	1
male	81.109185	18.890815

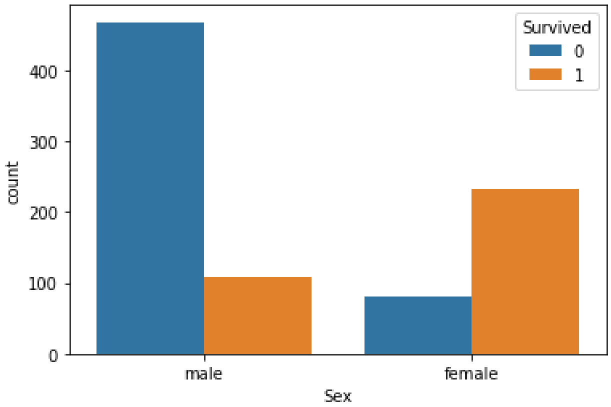
It means that almost Female survival probability is almost 74% which is almost 4 times that of the men. so we can infer that the females have more chances of survival.

In [304]:

```
sns.countplot(x="Sex",data=train,hue="Survived")
```

Out[304]:

```
<matplotlib.axes._subplots.AxesSubplot at 0xd9830f0>
```



Age as the target varriable:

The attribute "Age" contains NULL values so firstly we need to perform data cleaning here

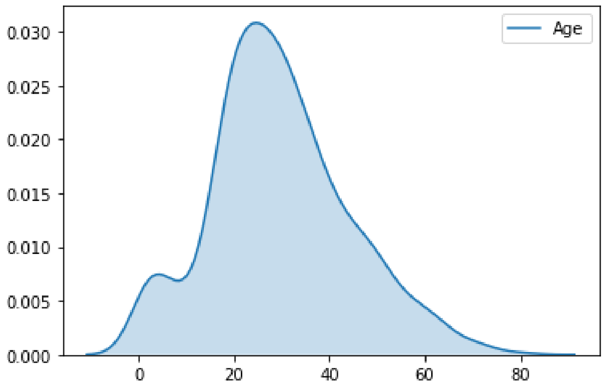
We will use KDEplot here as Age is a continuous varraible so Kernel Density Estimate (KDE) depicts the probability density of a continuous varriable.

In [305]:

```
sns.kdeplot(train.Age,shade=True)
```

Out[305]:

```
<matplotlib.axes._subplots.AxesSubplot at 0xdab09f0>
```



As Age has null values so we will fill it with its median values as we know that for a dataset with great outliers , it is advised to fill the Null values with median.

In [306]:

```
print("Median: "+str(train.Age.median()))
```

```
Median: 28.0
```

In [307]:

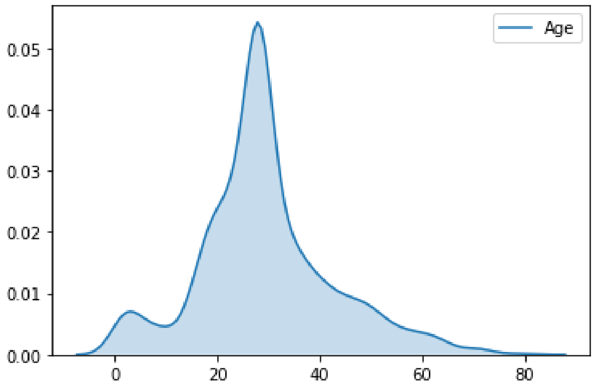
```
train["Age"].fillna(train.Age.median(),inplace=True)
```

In [308]:

```
sns.kdeplot(train.Age,shade=True)
```

Out[308]:

```
<matplotlib.axes._subplots.AxesSubplot at 0xdaf5f50>
```



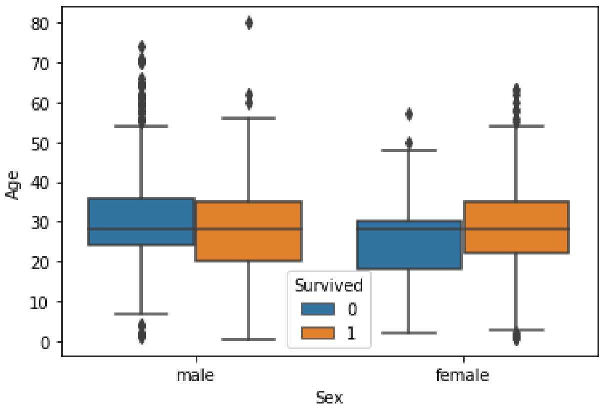
We see that the peak is very much close to 30 or in between 20 and 40. so from this we can say tht majority of the people on Titanic had age close to 30.

In [309]:

```
sns.boxplot(x="Sex",y="Age",data=train,hue="Survived")
```

Out[309]:

```
<matplotlib.axes._subplots.AxesSubplot at 0xdb31ab0>
```



From the boxplot we can infer that among all the people who had survived in Titanic are in between the age from 20 to 30.

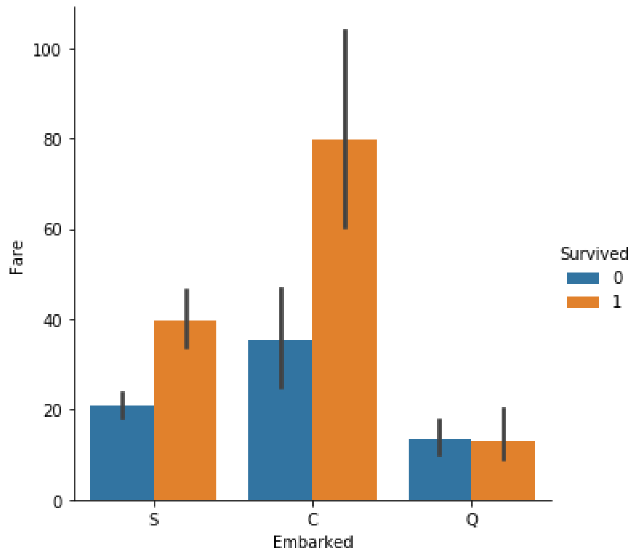
Fare as the target varriable:

In [310]:

```
sns.factorplot(x="Embarked",y="Fare",data=train,hue="Survived",kind="bar")
```

Out[310]:

```
<seaborn.axisgrid.FacetGrid at 0xdb54850>
```



From this we can infer that those people who had highly paid are likely to survive more.

Embarked as the target varriable:

We know that Embarked also has null varriables , so we need to perform data cleaning

So here we will fill the null values with Mode

In [311]:

```
train["Embarked"].fillna(train["Embarked"].mode()[0],inplace=True)
```

In [312]:

```
train.Embarked.count()
```

Out[312]:

```
891
```

In [313]:

```
pd.crosstab([train.Sex,train.Survived],[train.Pclass,train.Embarked])
```

Out[313]:

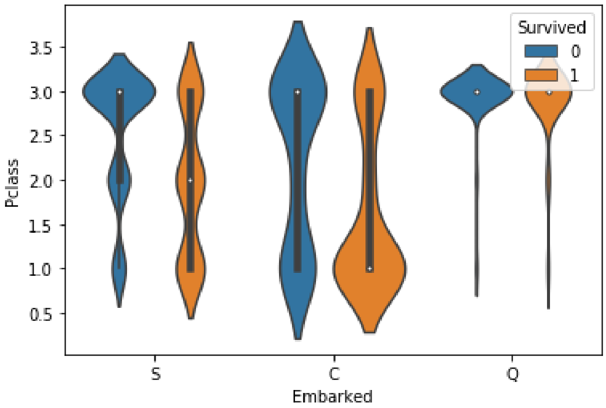
	Pclass	1			2			3		
	Embarked	C	Q	S	C	Q	S	C	Q	S
Sex	Survived									
female	0	1	0	2	0	0	6	8	9	55
	1	42	1	48	7	2	61	15	24	33
male	0	25	1	51	8	1	82	33	36	231
	1	17	0	28	2	0	15	10	3	34

In [314]:

```
sns.violinplot(x='Embarked' , y='Pclass' , data=train , hue='Survived' )
```

Out[314]:

<matplotlib.axes._subplots.AxesSubplot at 0xebb2c10>



We can see that those who embarked at C with First Class ticket had a good chance of Survival. Whereas for S, it seems that all classes had nearly equal probability of Survival. And for Q, third Class seems to have Survived and Died with similar probabilities.

SibSp as the target varriable:

In [315]:

```
# here we will evaluate the comparison between the individuals who have or not have
# siblings with the survival ratio
train[["SibSp","Survived"]].groupby("SibSp").mean()
```

Out[315]:

SibSp	Survived
0	0.345395
1	0.535885
2	0.464286
3	0.250000
4	0.166667
5	0.000000
8	0.000000

It means that those individuals having 1 or 2 siblins / spouses had the highest priority of survival followed by those who are alone

Parch :

In [316]:

```
# just like sibsp column we will perform similar analysis
train[["Parch","Survived"]].groupby("Parch").mean()
```

Out[316]:

Parch	Survived
0	0.343658
1	0.550847
2	0.500000
3	0.600000
4	0.000000

	Survived
Parch	
4	0.000000
5	0.200000
6	0.000000

From this we can infer that the individuals having 1,2 or 3 family memebers have higher probability to survive more.

W e can infer from Sibsp and Parch column that those individuls having more social relations with family members , siblinga have more chances of survival

In [317]:

```
train.head()
```

Out[317]:

	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

In [318]:

```
train["Cabin"].isnull().sum()
```

Out[318]:

```
687
```

In [319]:

```
display(train.Age.describe())
```

```
count    891.000000
mean      29.361582
std       13.019697
min        0.420000
25%       22.000000
50%       28.000000
75%       35.000000
max       80.000000
Name: Age, dtype: float64
```

In [320]:

```
def categorical_age(x):
    if 0 < x <= 22:
        return 0
    elif 22 < x <= 26:
        return 1
    elif 26 < x <= 36.5:
        return 2
    else:
        return 3

train['categorical_age'] = train['Age'].apply(categorical_age)
```

In [321]:

```
# Convert age into a categorical variables using the quartiles
display(train.Fare.describe())
```

```
count    891.000000
mean      32.204208
std       49.693429
min        0.000000
25%        7.910400
50%       14.454200
75%       31.000000
max      512.329200
Name: Fare, dtype: float64
```

In [322]:

```
def categorical_fare(x):
    if 0 < x <= 7.9:
        return 0
    elif 7.9 < x <= 14.5:
        return 1
    elif 14.5 < x <= 31:
        return 2
    else:
        return 3

train['categorical_fare'] = train['Fare'].apply(categorical_fare)
```

In [323]:

```
train.head()
```

Out[323]:

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

Converting Catagorical features:

We'll need to convert categorical features to dummy variables using pandas! Otherwise our machine learning algorithm won't be able to directly take in those features as inputs.

In [324]:

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 14 columns):
PassengerId         891 non-null int64
Survived            891 non-null int64
Pclass              891 non-null int64
```

```
Name                891 non-null object
Sex                 891 non-null object
Age                 891 non-null float64
SibSp               891 non-null int64
Parch               891 non-null int64
Ticket              891 non-null object
Fare                891 non-null float64
Cabin               204 non-null object
Embarked            891 non-null object
categorical_age     891 non-null int64
categorical_fare    891 non-null int64
dtypes: float64(2), int64(7), object(5)
memory usage: 80.1+ KB
```

In [325]:

```python
categorical_features = ['Pclass', 'Sex', 'Embarked']
train[categorical_features] = train[categorical_features].apply(lambda x: x.astype('category'))
```

In [326]:

```python
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 14 columns):
PassengerId         891 non-null int64
Survived            891 non-null int64
Pclass              891 non-null category
Name                891 non-null object
Sex                 891 non-null category
Age                 891 non-null float64
SibSp               891 non-null int64
Parch               891 non-null int64
Ticket              891 non-null object
Fare                891 non-null float64
Cabin               204 non-null object
Embarked            891 non-null category
categorical_age     891 non-null int64
categorical_fare    891 non-null int64
dtypes: category(3), float64(2), int64(6), object(3)
memory usage: 69.0+ KB
```

In [327]:

```python
# creating dummy variables
train = pd.get_dummies(data=train, columns=['Sex', 'Embarked', 'Pclass', 'categorical_age', 'catego
rical_fare'], drop_first=True)
```

In [328]:

```python
train.head()
```

Out[328]:

	PassengerId	Survived	Name	Age	SibSp	Parch	Ticket	Fare	Cabin	Sex_male	Embarked_Q	Embarked_S	Pclass_2
0	1	0	Braund, Mr. Owen Harris	22.0	1	0	A/5 21171	7.2500	NaN	1	0	1	0
1	2	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	38.0	1	0	PC 17599	71.2833	C85	0	0	0	0
2	3	1	Heikkinen, Miss. Laina	26.0	0	0	STON/O2. 3101282	7.9250	NaN	0	0	1	0
3	4	1	Futrelle, Mrs. Jacques	35.0	1	0	113803	53.1000	C123	0	0	1	0

	PassengerId	Survived	Name	Age	SibSp	Parch	Ticket	Fare	Cabin	Sex_male	Embarked_Q	Embarked_S	Pclass_2
3	4	1	Heath (Lily May Peel)	35.0	1	0	113803	53.1000	C123	0	0	1	0
4	5	0	Allen, Mr. William Henry	35.0	0	0	373450	8.0500	NaN	1	0	1	0

In [329]:

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 20 columns):
PassengerId            891 non-null int64
Survived               891 non-null int64
Name                   891 non-null object
Age                    891 non-null float64
SibSp                  891 non-null int64
Parch                  891 non-null int64
Ticket                 891 non-null object
Fare                   891 non-null float64
Cabin                  204 non-null object
Sex_male               891 non-null uint8
Embarked_Q             891 non-null uint8
Embarked_S             891 non-null uint8
Pclass_2               891 non-null uint8
Pclass_3               891 non-null uint8
categorical_age_1      891 non-null uint8
categorical_age_2      891 non-null uint8
categorical_age_3      891 non-null uint8
categorical_fare_1     891 non-null uint8
categorical_fare_2     891 non-null uint8
categorical_fare_3     891 non-null uint8
dtypes: float64(2), int64(4), object(3), uint8(11)
memory usage: 61.8+ KB
```

In [330]:

```
train.drop(['Name', 'Ticket', 'Age', 'Fare'], axis=1, inplace=True)
```

In [331]:

```
display(train.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 16 columns):
PassengerId            891 non-null int64
Survived               891 non-null int64
SibSp                  891 non-null int64
Parch                  891 non-null int64
Cabin                  204 non-null object
Sex_male               891 non-null uint8
Embarked_Q             891 non-null uint8
Embarked_S             891 non-null uint8
Pclass_2               891 non-null uint8
Pclass_3               891 non-null uint8
categorical_age_1      891 non-null uint8
categorical_age_2      891 non-null uint8
categorical_age_3      891 non-null uint8
categorical_fare_1     891 non-null uint8
categorical_fare_2     891 non-null uint8
categorical_fare_3     891 non-null uint8
dtypes: int64(4), object(1), uint8(11)
memory usage: 41.0+ KB
```

```
None
```

In [332]:

```
train.drop(["Cabin"],axis=1,inplace=True)
```

In [333]:

```
#keeping a copy of the initial dataset into another dataset
df=train.copy()
```

In [334]:

```
train.drop("PassengerId",axis=1,inplace=True)
```

In [335]:

```
train.head()
```

Out[335]:

	Survived	SibSp	Parch	Sex_male	Embarked_Q	Embarked_S	Pclass_2	Pclass_3	categorical_age_1	categorical_age_2	categorica
0	0	1	0	1	0	1	0	1	0	0	
1	1	1	0	0	0	0	0	0	0	0	
2	1	0	0	0	0	1	0	1	1	0	
3	1	1	0	0	0	1	0	0	0	1	
4	0	0	0	1	0	1	0	1	0	1	

Noe our dataset is ready to build Supervised learning Estimators

Supervised Learning Estimators(applying Logistic Regression):

In [336]:

```
x_train = train.drop(['Survived'],axis=1)
y_train = train['Survived']
```

In [337]:

```
import statsmodels.api as sm
```

In [338]:

```
logm1 = sm.GLM(y_train,(sm.add_constant(x_train)), family = sm.families.Binomial())
logm1.fit().summary()
```

Out[338]:

Generalized Linear Model Regression Results

Dep. Variable:	Survived	**No. Observations:**	891
Model:	GLM	**Df Residuals:**	877
Model Family:	Binomial	**Df Model:**	13
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-394.93
Date:	Mon, 24 Aug 2020	**Deviance:**	789.86
Time:	23:57:30	**Pearson chi2:**	905.
No. Iterations:	5		
Covariance Type:	nonrobust		

	coef	std err	z	P>|z|	[0.025	0.975]

const	2.9423	0.501	5.871	0.000	1.960	3.925
SibSp	-0.3736	0.123	-3.048	0.002	-0.614	-0.133
Parch	-0.1030	0.127	-0.809	0.418	-0.352	0.146
Sex_male	-2.7139	0.201	-13.522	0.000	-3.107	-2.321
Embarked_Q	-0.0409	0.398	-0.103	0.918	-0.821	0.739
Embarked_S	-0.4822	0.236	-2.047	0.041	-0.944	-0.020
Pclass_2	-0.7070	0.322	-2.193	0.028	-1.339	-0.075
Pclass_3	-1.7721	0.356	-4.985	0.000	-2.469	-1.075
categorical_age_1	-0.6219	0.348	-1.789	0.074	-1.303	0.059
categorical_age_2	-0.4952	0.238	-2.084	0.037	-0.961	-0.029
categorical_age_3	-1.2092	0.290	-4.175	0.000	-1.777	-0.641
categorical_fare_1	0.1709	0.313	0.545	0.585	-0.443	0.785
categorical_fare_2	0.7175	0.352	2.036	0.042	0.027	1.408
categorical_fare_3	0.6620	0.437	1.514	0.130	-0.195	1.519

In [339]:

```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
```

In [340]:

```
 from sklearn.feature_selection import RFE
rfe = RFE(logreg, 15) # running RFE with 13 variables as output
rfe = rfe.fit(x_train, y_train)
```

In [341]:

```
rfe.support_
```

Out[341]:

```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True])
```

In [342]:

```
list(zip(x_train.columns, rfe.support_, rfe.ranking_))
```

Out[342]:

```
[('SibSp', True, 1),
 ('Parch', True, 1),
 ('Sex_male', True, 1),
 ('Embarked_Q', True, 1),
 ('Embarked_S', True, 1),
 ('Pclass_2', True, 1),
 ('Pclass_3', True, 1),
 ('categorical_age_1', True, 1),
 ('categorical_age_2', True, 1),
 ('categorical_age_3', True, 1),
 ('categorical_fare_1', True, 1),
 ('categorical_fare_2', True, 1),
 ('categorical_fare_3', True, 1)]
```

In [343]:

```
 col = x_train.columns[rfe.support_]
```

In [344]:

```
x_train.columns[~rfe.support_]
```

Out[344]:

```
Index([], dtype='object')
```

In [345]:

```
x_train_sm = sm.add_constant(x_train[col])
logm2 = sm.GLM(y_train,x_train_sm, family = sm.families.Binomial())
res = logm2.fit()
res.summary()
```

Out[345]:

Generalized Linear Model Regression Results

Dep. Variable:	Survived	**No. Observations:**	891
Model:	GLM	**Df Residuals:**	877
Model Family:	Binomial	**Df Model:**	13
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-394.93
Date:	Mon, 24 Aug 2020	**Deviance:**	789.86
Time:	23:57:32	**Pearson chi2:**	905.
No. Iterations:	5		
Covariance Type:	nonrobust		

	coef	std err	z	P>|z|	[0.025	0.975]
const	2.9423	0.501	5.871	0.000	1.960	3.925
SibSp	-0.3736	0.123	-3.048	0.002	-0.614	-0.133
Parch	-0.1030	0.127	-0.809	0.418	-0.352	0.146
Sex_male	-2.7139	0.201	-13.522	0.000	-3.107	-2.321
Embarked_Q	-0.0409	0.398	-0.103	0.918	-0.821	0.739
Embarked_S	-0.4822	0.236	-2.047	0.041	-0.944	-0.020
Pclass_2	-0.7070	0.322	-2.193	0.028	-1.339	-0.075
Pclass_3	-1.7721	0.356	-4.985	0.000	-2.469	-1.075
categorical_age_1	-0.6219	0.348	-1.789	0.074	-1.303	0.059
categorical_age_2	-0.4952	0.238	-2.084	0.037	-0.961	-0.029
categorical_age_3	-1.2092	0.290	-4.175	0.000	-1.777	-0.641
categorical_fare_1	0.1709	0.313	0.545	0.585	-0.443	0.785
categorical_fare_2	0.7175	0.352	2.036	0.042	0.027	1.408
categorical_fare_3	0.6620	0.437	1.514	0.130	-0.195	1.519

When observing the p-values we can see that there are many insignificant features in our first model built with p-values more than 0.05 which marks them as insignificant. We can go for backward approach now by removing each of those features one by one, rebuilt and see for the model.

In [346]:

```
x_train_sm.drop(["Parch"],axis=1,inplace=True)
```

In [347]:

```
model2 = sm.GLM(y,(sm.add_constant(x_train_sm)),family=sm.families.Binomial())
model2.fit().summary()
```

Out[347]:

Generalized Linear Model Regression Results

Dep. Variable:	Survived	**No. Observations:**	891

Dep. Variable:	Survived	**No. Observations:**	891
Model:	GLM	**Df Residuals:**	878
Model Family:	Binomial	**Df Model:**	12
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-395.26
Date:	Mon, 24 Aug 2020	**Deviance:**	790.52
Time:	23:57:32	**Pearson chi2:**	909.
No. Iterations:	5		
Covariance Type:	nonrobust		

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
const	2.9940	0.498	6.009	0.000	2.017	3.971
SibSp	-0.3798	0.122	-3.101	0.002	-0.620	-0.140
Sex_male	-2.6823	0.196	-13.669	0.000	-3.067	-2.298
Embarked_Q	-0.0280	0.397	-0.071	0.944	-0.806	0.750
Embarked_S	-0.4816	0.235	-2.046	0.041	-0.943	-0.020
Pclass_2	-0.7662	0.315	-2.432	0.015	-1.383	-0.149
Pclass_3	-1.8646	0.339	-5.503	0.000	-2.529	-1.201
categorical_age_1	-0.6146	0.349	-1.763	0.078	-1.298	0.068
categorical_age_2	-0.4707	0.236	-1.998	0.046	-0.932	-0.009
categorical_age_3	-1.2176	0.289	-4.209	0.000	-1.785	-0.651
categorical_fare_1	0.1554	0.312	0.498	0.619	-0.457	0.768
categorical_fare_2	0.6350	0.338	1.878	0.060	-0.028	1.298
categorical_fare_3	0.5344	0.410	1.303	0.192	-0.269	1.338

In [348]:

```
x_train_sm.drop(["Embarked_Q"],axis=1,inplace=True)
```

In [349]:

```
model2 = sm.GLM(y,(sm.add_constant(x_train_sm)),family=sm.families.Binomial())
model2.fit().summary()
```

Out[349]:

Generalized Linear Model Regression Results

Dep. Variable:	Survived	**No. Observations:**	891
Model:	GLM	**Df Residuals:**	879
Model Family:	Binomial	**Df Model:**	11
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-395.26
Date:	Mon, 24 Aug 2020	**Deviance:**	790.53
Time:	23:57:33	**Pearson chi2:**	910.
No. Iterations:	5		
Covariance Type:	nonrobust		

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
const	2.9841	0.478	6.241	0.000	2.047	3.921
SibSp	-0.3801	0.122	-3.105	0.002	-0.620	-0.140
Sex_male	-2.6806	0.195	-13.766	0.000	-3.062	-2.299
Embarked_S	-0.4739	0.209	-2.270	0.023	-0.883	-0.065
Pclass_2	-0.7683	0.313	-2.451	0.014	-1.383	-0.154

Pclass_3	-1.8676	0.336	-5.555	0.000	-2.526	-1.209
categorical_age_1	-0.6137	0.348	-1.762	0.078	-1.296	0.069
categorical_age_2	-0.4733	0.233	-2.033	0.042	-0.930	-0.017
categorical_age_3	-1.2193	0.288	-4.229	0.000	-1.784	-0.654
categorical_fare_1	0.1608	0.303	0.531	0.596	-0.433	0.755
categorical_fare_2	0.6398	0.331	1.933	0.053	-0.009	1.289
categorical_fare_3	0.5400	0.402	1.343	0.179	-0.248	1.328

In [350]:

```
x_train_sm.drop(["Embarked_S"],axis=1,inplace=True)
```

In [351]:

```
model2 = sm.GLM(y,(sm.add_constant(x_train_sm)),family=sm.families.Binomial())
model2.fit().summary()
```

Out[351]:

Generalized Linear Model Regression Results

Dep. Variable:	Survived	**No. Observations:**	891
Model:	GLM	**Df Residuals:**	880
Model Family:	Binomial	**Df Model:**	10
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-397.84
Date:	Mon, 24 Aug 2020	**Deviance:**	795.68
Time:	23:57:33	**Pearson chi2:**	907.
No. Iterations:	5		
Covariance Type:	nonrobust		

	coef	std err	z	P>|z|	[0.025	0.975]
const	2.8109	0.467	6.025	0.000	1.896	3.725
SibSp	-0.4004	0.122	-3.274	0.001	-0.640	-0.161
Sex_male	-2.7330	0.194	-14.085	0.000	-3.113	-2.353
Pclass_2	-0.8850	0.309	-2.860	0.004	-1.491	-0.279
Pclass_3	-1.9294	0.333	-5.795	0.000	-2.582	-1.277
categorical_age_1	-0.6004	0.347	-1.731	0.084	-1.280	0.080
categorical_age_2	-0.4300	0.230	-1.867	0.062	-0.881	0.022
categorical_age_3	-1.2325	0.287	-4.292	0.000	-1.795	-0.670
categorical_fare_1	0.0249	0.294	0.085	0.932	-0.551	0.600
categorical_fare_2	0.5545	0.326	1.701	0.089	-0.085	1.194
categorical_fare_3	0.4553	0.396	1.150	0.250	-0.321	1.231

In [352]:

```
x_train_sm.drop(["categorical_age_1"],axis=1,inplace=True)
```

In [353]:

```
model2 = sm.GLM(y,(sm.add_constant(x_train_sm)),family=sm.families.Binomial())
model2.fit().summary()
```

Out[353]:

Generalized Linear Model Regression Results

Dep. Variable:	Survived	**No. Observations:**	891
Model:	GLM	**Df Residuals:**	881
Model Family:	Binomial	**Df Model:**	9
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-399.36
Date:	Mon, 24 Aug 2020	**Deviance:**	798.72
Time:	23:57:33	**Pearson chi2:**	909.
No. Iterations:	5		
Covariance Type:	nonrobust		

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
const	2.5979	0.447	5.810	0.000	1.722	3.474
SibSp	-0.3843	0.121	-3.174	0.002	-0.622	-0.147
Sex_male	-2.7304	0.193	-14.116	0.000	-3.110	-2.351
Pclass_2	-0.8896	0.309	-2.875	0.004	-1.496	-0.283
Pclass_3	-1.8833	0.331	-5.692	0.000	-2.532	-1.235
categorical_age_2	-0.2570	0.208	-1.237	0.216	-0.664	0.150
categorical_age_3	-1.0490	0.266	-3.939	0.000	-1.571	-0.527
categorical_fare_1	0.0375	0.293	0.128	0.898	-0.536	0.611
categorical_fare_2	0.5628	0.325	1.734	0.083	-0.074	1.199
categorical_fare_3	0.4757	0.394	1.208	0.227	-0.296	1.247

In [354]:

```
x_train_sm.drop(["categorical_age_2"],axis=1,inplace=True)
```

In [355]:

```
model2 = sm.GLM(y,(sm.add_constant(x_train_sm)),family=sm.families.Binomial())
model2.fit().summary()
```

Out[355]:

Generalized Linear Model Regression Results

Dep. Variable:	Survived	**No. Observations:**	891
Model:	GLM	**Df Residuals:**	882
Model Family:	Binomial	**Df Model:**	8
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-400.13
Date:	Mon, 24 Aug 2020	**Deviance:**	800.26
Time:	23:57:34	**Pearson chi2:**	906.
No. Iterations:	5		
Covariance Type:	nonrobust		

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
const	2.4183	0.420	5.751	0.000	1.594	3.242
SibSp	-0.3661	0.118	-3.107	0.002	-0.597	-0.135
Sex_male	-2.7373	0.193	-14.170	0.000	-3.116	-2.359
Pclass_2	-0.8620	0.308	-2.800	0.005	-1.465	-0.259
Pclass_3	-1.8462	0.328	-5.621	0.000	-2.490	-1.202
categorical_age_3	-0.8912	0.233	-3.825	0.000	-1.348	-0.435
categorical_fare_1	0.0543	0.292	0.186	0.852	-0.517	0.626
categorical_fare_2	0.5493	0.324	1.698	0.090	-0.085	1.184

categorical_fare_3	0.4916	0.392	1.253	0.210	-0.278	1.261

In [356]:

```
x_train_sm.drop(["categorical_fare_1"],axis=1,inplace=True)
```

In [357]:

```
model2 = sm.GLM(y,(sm.add_constant(x_train_sm)),family=sm.families.Binomial())
model2.fit().summary()
```

Out[357]:

Generalized Linear Model Regression Results

Dep. Variable:	Survived	**No. Observations:**	891
Model:	GLM	**Df Residuals:**	883
Model Family:	Binomial	**Df Model:**	7
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-400.15
Date:	Mon, 24 Aug 2020	**Deviance:**	800.29
Time:	23:57:34	**Pearson chi2:**	909.
No. Iterations:	5		
Covariance Type:	nonrobust		

	coef	std err	z	P>|z|	[0.025	0.975]
const	2.4557	0.370	6.638	0.000	1.731	3.181
SibSp	-0.3633	0.117	-3.110	0.002	-0.592	-0.134
Sex_male	-2.7391	0.193	-14.194	0.000	-3.117	-2.361
Pclass_2	-0.8543	0.305	-2.800	0.005	-1.452	-0.256
Pclass_3	-1.8564	0.324	-5.728	0.000	-2.492	-1.221
categorical_age_3	-0.8888	0.233	-3.820	0.000	-1.345	-0.433
categorical_fare_2	0.5122	0.255	2.009	0.044	0.013	1.012
categorical_fare_3	0.4526	0.332	1.363	0.173	-0.198	1.103

In [358]:

```
x_train_sm.drop(["categorical_fare_3"],axis=1,inplace=True)
```

In [359]:

```
model2 = sm.GLM(y,(sm.add_constant(x_train_sm)),family=sm.families.Binomial())
model2.fit().summary()
```

Out[359]:

Generalized Linear Model Regression Results

Dep. Variable:	Survived	**No. Observations:**	891
Model:	GLM	**Df Residuals:**	884
Model Family:	Binomial	**Df Model:**	6
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-401.06
Date:	Mon, 24 Aug 2020	**Deviance:**	802.12
Time:	23:57:34	**Pearson chi2:**	917.
No. Iterations:	5		
Covariance Type:	nonrobust		

Covariance Type:	nonrobust		

	coef	std err	z	P>|z|	[0.025	0.975]
const	2.8037	0.271	10.360	0.000	2.273	3.334
SibSp	-0.2833	0.099	-2.871	0.004	-0.477	-0.090
Sex_male	-2.7487	0.193	-14.270	0.000	-3.126	-2.371
Pclass_2	-1.0801	0.257	-4.198	0.000	-1.584	-0.576
Pclass_3	-2.1604	0.238	-9.092	0.000	-2.626	-1.695
categorical_age_3	-0.8738	0.232	-3.771	0.000	-1.328	-0.420
categorical_fare_2	0.2995	0.201	1.493	0.135	-0.094	0.693

In [360]:

```
x_train_sm.drop(["categorical_fare_2"],axis=1,inplace=True)
```

In [361]:

```
model2 = sm.GLM(y,(sm.add_constant(x_train_sm)),family=sm.families.Binomial())
result=model2.fit()
result.summary()
```

Out[361]:

Generalized Linear Model Regression Results

Dep. Variable:	Survived	**No. Observations:**	891
Model:	GLM	**Df Residuals:**	885
Model Family:	Binomial	**Df Model:**	5
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-402.17
Date:	Mon, 24 Aug 2020	**Deviance:**	804.35
Time:	23:57:35	**Pearson chi2:**	915.
No. Iterations:	5		
Covariance Type:	nonrobust		

	coef	std err	z	P>|z|	[0.025	0.975]
const	2.8984	0.267	10.871	0.000	2.376	3.421
SibSp	-0.2631	0.095	-2.756	0.006	-0.450	-0.076
Sex_male	-2.7652	0.193	-14.319	0.000	-3.144	-2.387
Pclass_2	-1.0660	0.257	-4.143	0.000	-1.570	-0.562
Pclass_3	-2.1912	0.238	-9.216	0.000	-2.657	-1.725
categorical_age_3	-0.8704	0.232	-3.755	0.000	-1.325	-0.416

Now all the features looks significant. Now we can go ahead and see for the Variance Inflation Factor for thes features and make a decision

In [362]:

```
# Check for the VIF values of the feature variables.
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

In [363]:

```
# Create a dataframe that will contain the names of all the feature variables that w used before a
nd their respective VIFs
vif = pd.DataFrame()
vif['Features'] = x_train_sm.columns
vif['VIF'] = [variance_inflation_factor(x_train_sm.values, i) for i in range(x_train_sm.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
```

```
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[363]:

	Features	VIF
0	const	7.03
4	Pclass_3	1.68
3	Pclass_2	1.52
5	categorical_age_3	1.12
2	Sex_male	1.04
1	SibSp	1.03

In [364]:

```
x_train_sm.drop("const",axis=1,inplace=True)
```

In [365]:

```
# Create a dataframe that will contain the names of all the feature variables that w used before a
nd their respective VIFs
vif = pd.DataFrame()
vif['Features'] = x_train_sm.columns
vif['VIF'] = [variance_inflation_factor(x_train_sm.values, i) for i in range(x_train_sm.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[365]:

	Features	VIF
1	Sex_male	2.35
3	Pclass_3	2.10
2	Pclass_2	1.27
0	SibSp	1.20
4	categorical_age_3	1.19

So the final features are "Sex_male" , "Pclass_3" , "Pclass_2" , "SibSP" ,"categorical_age_3".

In [366]:

```
y_train_pred = result.predict(sm.add_constant(x_train_sm)).values.reshape(-1)
```

In [367]:

```
y_train_pred[:10]
```

Out[367]:

```
array([0.08939138, 0.85383015, 0.6697825 , 0.93310007, 0.11324449,
       0.11324449, 0.32361891, 0.05482487, 0.6697825 , 0.82769109])
```

In [368]:

```
df.head()
```

Out[368]:

	PassengerId	Survived	SibSp	Parch	Sex_male	Embarked_Q	Embarked_S	Pclass_2	Pclass_3	categorical_age_1	categorical_age
0	1	0	1	0	1	0	1	0	1	0	

	PassengerId	Survived	SibSp	Parch	Sex_male	Embarked_Q	Embarked_S	Pclass_2	Pclass_3	categorical_age_1	categorical_age
1	2	1	1	0	0	0	0	0	0	0	
2	3	1	0	0	0	0	1	0	1	1	
3	4	1	1	0	0	0	1	0	0	0	
4	5	0	0	0	1	0	1	0	1	0	

In [369]:

```
df1=df[["PassengerId","Survived"]]
```

In [370]:

```
df1["Predicted_Survival"]=y_train_pred
```

In [371]:

```
df1.head()
```

Out[371]:

	PassengerId	Survived	Predicted_Survival
0	1	0	0.089391
1	2	1	0.853830
2	3	1	0.669783
3	4	1	0.933100
4	5	0	0.113244

Find the optimal cut-off point:

In [372]:

```
# Let's create columns with different probability cutoff
numbers = [float(x)/10 for x in range(10)]
for i in numbers:
    df1[i] = df1["Predicted_Survival"].map(lambda x: 1 if x > i else 0)
df1.head()
```

Out[372]:

	PassengerId	Survived	Predicted_Survival	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0	1	0	0.089391	1	0	0	0	0	0	0	0	0	0
1	2	1	0.853830	1	1	1	1	1	1	1	1	1	0
2	3	1	0.669783	1	1	1	1	1	1	1	0	0	0
3	4	1	0.933100	1	1	1	1	1	1	1	1	1	1
4	5	0	0.113244	1	1	0	0	0	0	0	0	0	0

In [373]:

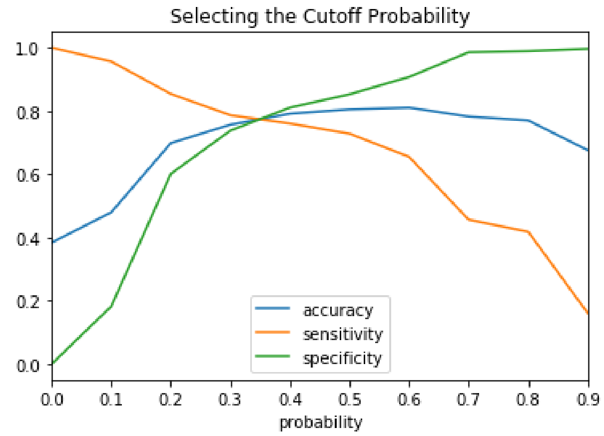
```
# Now let's calculate accuracy sensitivity and specificity for various probability cutoffs.
cutoff_df = pd.DataFrame(columns=['probability','accuracy','sensitivity','specificity'])
from sklearn.metrics import confusion_matrix
num = [0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9]
for i in num:
    cm1 = confusion_matrix(df1.Survived,df1[i])
    total1=sum(sum(cm1))
    accuracy = (cm1[0,0]+cm1[1,1])/total1
    sensi = cm1[1,1] / (cm1[1,0]+cm1[1,1])
    speci = cm1[0,0] / (cm1[0,0]+cm1[0,1])
    cutoff_df.loc[i] = [i,accuracy,sensi,speci]
cutoff_df
```

Out[373]:

	probability	accuracy	sensitivity	specificity
0.0	0.0	0.383838	1.000000	0.000000
0.1	0.1	0.479237	0.956140	0.182149
0.2	0.2	0.698092	0.853801	0.601093
0.3	0.3	0.756453	0.786550	0.737705
0.4	0.4	0.791246	0.760234	0.810565
0.5	0.5	0.804714	0.728070	0.852459
0.6	0.6	0.810325	0.654971	0.907104
0.7	0.7	0.782267	0.456140	0.985428
0.8	0.8	0.769921	0.418129	0.989071
0.9	0.9	0.675645	0.160819	0.996357

In [374]:

```
cutoff_df.plot.line(x='probability',y=['accuracy','sensitivity','specificity'])
plt.title('Selecting the Cutoff Probability')
plt.show()
```



From the above graph we can infer that the cut-off probability lies in between 0.3 and 0.4 so lets consder it to be 0.35

In [375]:

```
df1['Predicted'] = df1["Predicted_Survival"].map(lambda x : 1 if x > 0.35 else 0 )
```

In [376]:

```
df1.head()
```

Out[376]:

	PassengerId	Survived	Predicted_Survival	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	Predicted
0	1	0	0.089391	1	0	0	0	0	0	0	0	0	0	0
1	2	1	0.853830	1	1	1	1	1	1	1	1	1	0	1
2	3	1	0.669783	1	1	1	1	1	1	1	0	0	0	1
3	4	1	0.933100	1	1	1	1	1	1	1	1	1	1	1
4	5	0	0.113244	1	1	0	0	0	0	0	0	0	0	0

In [377]:

```
from sklearn import metrics
```

In [378]:

```
# Let's check the overall accuracy.
print(round(100* metrics.accuracy_score(df1.Survived,df1.Predicted),2))
confusion = confusion_matrix(df1['Survived'],df1['Predicted'])
print(confusion)
```

```
78.45
[[438 111]
 [ 81 261]]
```

Predicted not_survived survived Actual not_survived 438 111 survived 81 261

Metrics Beyond Simple Accuracy:

In [379]:

```
TP = confusion[1,1] # true positive
TN = confusion[0,0] # true negatives
FP = confusion[0,1] # false positives
FN = confusion[1,0] # false negatives
```

In [380]:

```
#Sensitivity
round(100*(TP / float(TP+FN)),2)
```

Out[380]:

```
76.32
```

In [381]:

```
#Specificity
round(100*(TN / float(TN+FP)),2)
```

Out[381]:

```
79.78
```

In [382]:

```
#False Positive Rate
round(100*(FP / float(FP+TN)),2)
```

Out[382]:

```
20.22
```

In [383]:

```
#Positive Predictive Value
round(100*(TP / float(TP+FP)),2)
```

Out[383]:

```
70.16
```

In [384]:

```
#Negative Predictive Value
round(100*(TN / float(TN+FN)),2)
```

Out[384]:

```
84.39
```

ROC CURVE

In [385]:

```
from sklearn import metrics
```

In [386]:

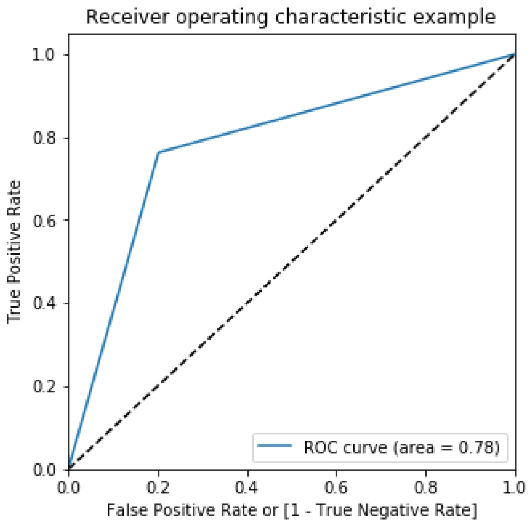
```
def draw_roc( actual, probs ):
    fpr, tpr, thresholds = metrics.roc_curve( actual, probs,
                                              drop_intermediate = False )
    auc_score = metrics.roc_auc_score( actual, probs )
    plt.figure(figsize=(5, 5))
    plt.plot( fpr, tpr, label='ROC curve (area = %0.2f)' % auc_score )
    plt.plot([0, 1], [0, 1], 'k--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate or [1 - True Negative Rate]')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic example')
    plt.legend(loc="lower right")
    plt.show()

    return None
```

In [387]:

```
fpr, tpr, thresholds = metrics.roc_curve(df1.Survived,df1.Predicted, drop_intermediate = False )
```

In [388]:

```
draw_roc(df1.Survived,df1.Predicted)
```



Precision And Recall

In [389]:

```
precision = TP / float(TP+FP)
```

In [390]:

```
recall = TP / float(TP+FN)
```

In [391]:

```
from sklearn.metrics import precision_score, recall_score
```

In [392]:

```
precision_score(df1.Survived,df1.Predicted)
```

Out[392]:

```
0.7016129032258065
```

In [393]:

```
recall_score(df1.Survived,df1.Predicted)
```

Out[393]:

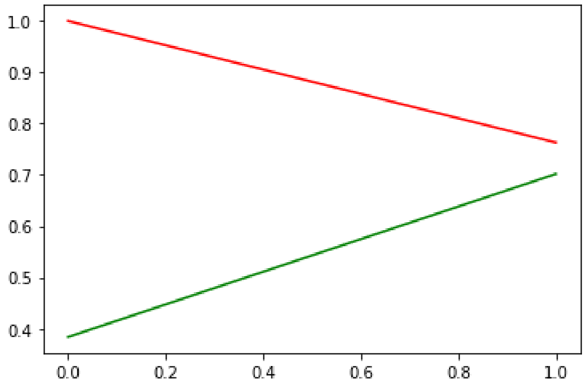
```
0.7631578947368421
```

In [394]:

```
from sklearn.metrics import precision_recall_curve
```

In [395]:

```
p, r, thresholds = precision_recall_curve(df1.Survived, df1.Predicted)
plt.plot(thresholds, p[:-1], "g-")
plt.plot(thresholds, r[:-1], "r-")
plt.show()
```



Test Dataset Computations:

In [396]:

```
test.head()
```

Out[396]:

	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

We need to keep the test dataset ready to run our previous model which we built and predict the passengers here in test dataset whether they survived or not. For this we can run the same transformations and set of operations that ran in train.

In [404]:

```
test.shape
```

Out[404]:

```
(418, 12)
```

In [405]:

```
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 12 columns):
PassengerId        418 non-null int64
Pclass             418 non-null int64
Name               418 non-null object
Sex                418 non-null object
Age                418 non-null float64
SibSp              418 non-null int64
Parch              418 non-null int64
Ticket             418 non-null object
Fare               417 non-null float64
Cabin              91 non-null object
Embarked           418 non-null object
categorical_age    418 non-null int64
dtypes: float64(2), int64(5), object(5)
memory usage: 31.1+ KB
```

In [406]:

```
print("Median: "+str(test.Age.median()))
```

```
Median: 27.0
```

In [407]:

```
test["Age"].fillna(test.Age.median(),inplace=True)
```

In [408]:

```
test["Embarked"].fillna(test["Embarked"].mode()[0],inplace=True)
```

In [409]:

```
display(test.Age.describe())
```

```
count    418.000000
mean      29.599282
std       12.703770
min        0.170000
25%       23.000000
50%       27.000000
75%       35.750000
max       76.000000
Name: Age, dtype: float64
```

In [410]:

```
def categorical_age(x):
    if 0 < x <= 23:
        return 0
    elif 23 < x <= 27:
        return 1
    elif 27 < x <= 36.5:
        return 2
    else:
        return 3
```

```
test['categorical_age'] = test['Age'].apply(categorical_age)
```

In [411]:

```
# Convert age into a categorical variables using the quartiles
display(test.Fare.describe())
```

```
count    417.000000
mean      35.627188
std       55.907576
min        0.000000
25%        7.895800
50%       14.454200
75%       31.500000
max      512.329200
Name: Fare, dtype: float64
```

In [412]:

```
def categorical_fare(x):
    if 0 < x <= 7.9:
        return 0
    elif 7.9 < x <= 14.5:
        return 1
    elif 14.5 < x <= 32:
        return 2
    else:
        return 3

test['categorical_fare'] = test['Fare'].apply(categorical_fare)
```

In [413]:

```
test.head()
```

Out[413]:

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

In [414]:

```
test.drop("Cabin",axis=1,inplace=True)
```

In [415]:

```
categorical_features = ['Pclass', 'Sex', 'Embarked']
test[categorical_features] = test[categorical_features].apply(lambda x: x.astype('category'))
```

In [416]:

```
test = pd.get_dummies(data=test, columns=['Sex', 'Embarked', 'Pclass', 'categorical_age', 'categori
```

```
test = pd.get_dummies(data=test, columns=['Sex', 'Embarked', 'Pclass', 'categorical_age', 'categorical_fare'], drop_first=True)
```

In [417]:

```
test.drop(['Name', 'Ticket', 'Age', 'Fare'], axis=1, inplace=True)
```

In [418]:

```
test.head()
```

Out[418]:

	PassengerId	SibSp	Parch	Sex_male	Embarked_Q	Embarked_S	Pclass_2	Pclass_3	categorical_age_1	categorical_age_2	catego
0	892	0	0	1	1	0	0	1	0	1	
1	893	1	0	0	0	1	0	1	0	0	
2	894	0	0	1	1	0	1	0	0	0	
3	895	0	0	1	0	1	0	1	1	0	
4	896	1	1	0	0	1	0	1	0	0	

Now basically we have done all the transformations in test as same as what we have done in train dataset including the imputing of missing values with new level,Binning the values,dummy variables for some features,maping for binary level features etc.

Now we can select only those selected columns with which we build the model and apply the model on the test dataset.

In [420]:

```
final_features=[]
final_features_index = x_train_sm.columns
for i in final_features_index:
    final_features.append(i)
```

In [421]:

```
final_features
```

Out[421]:

```
['SibSp', 'Sex_male', 'Pclass_2', 'Pclass_3', 'categorical_age_3']
```

In [1200]:

```
df_test = test[final_features]
df_final =test['PassengerId']
```

In [1201]:

```
df_test.head()
```

Out[1201]:

	SibSp	Sex_male	Pclass_2	Pclass_3	categorical_age_3
0	0	1	0	1	0
1	1	0	0	1	1
2	0	1	1	0	1
3	0	1	0	1	0
4	1	0	0	1	0

In [1202]:

```
y_test_pred = result.predict(sm.add_constant(df_test)).values.reshape(-1)
```

In [1203]:

```
y_test_pred[:10]
```

Out[1203]:

```
array([0.11324449, 0.39502962, 0.14146763, 0.11324449, 0.60924336,
       0.11324449, 0.6697825 , 0.23221064, 0.6697825 , 0.070165  ])
```

In [1204]:

```
df_final = pd.DataFrame(df_final)
df_final['Survived_Prob'] = y_test_pred
```

In [1205]:

```
#We can use the same cut-off probability of 0.4 to predict whether the passenger survived or not.
df_final['Survived'] = df_final['Survived_Prob'].map(lambda x : 1 if x > 0.35 else 0 )
```

In [1206]:

```
df_final.drop(['Survived_Prob'],axis=1,inplace=True)
```

In [1207]:

```
df_final.sort_values(by=['PassengerId'],ascending=True,inplace=True)
```

In [1208]:

```
df_final
```

Out[1208]:

	PassengerId	Survived
0	892	0
1	893	1
2	894	0
3	895	0
4	896	1
...	...	...
413	1305	0
414	1306	1
415	1307	0
416	1308	0
417	1309	0

418 rows × 2 columns

In [1209]:

```
df_final.shape
```

Out[1209]:

```
(418, 2)
```

In [1210]:

```
df_final.to_csv('Titanic.csv',header=True,index=False)
```

(: THANK YOU :)