In [65]:

```
!pip install sklearn
```

```
Requirement already satisfied: sklearn in c:\users\user\anaconda3\lib\site-p
ackages (0.0)
Requirement already satisfied: scikit-learn in c:\users\user\anaconda3\lib\s
ite-packages (from sklearn) (0.22.1)
Requirement already satisfied: joblib>=0.11 in c:\users\user\anaconda3\lib\s
ite-packages (from scikit-learn->sklearn) (0.13.2)
Requirement already satisfied: numpy>=1.11.0 in c:\users\user\anaconda3\lib
\site-packages (from scikit-learn->sklearn) (1.16.5)
Requirement already satisfied: scipy>=0.17.0 in c:\users\user\anaconda3\lib
\site-packages (from scikit-learn->sklearn) (1.3.1)
```

In [66]:

```
import pandas as pd
from sklearn.linear_model import LogisticRegression
```

In [67]:

```
titanic = pd.read_csv(r"C:\Users\USER\Desktop\Arun\titanic.csv")
```

In [68]:

```
titanic
```

Out[68]:

	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	e
0	1.0	1.0	Allen, Miss. Elisabeth Walton	female	29.0000	0.0	0.0	24160	211.3375	B5	
1	1.0	1.0	Allison, Master. Hudson Trevor	male	0.9167	1.0	2.0	113781	151.5500	C22 C26	
2	1.0	0.0	Allison, Miss. Helen Loraine	female	2.0000	1.0	2.0	113781	151.5500	C22 C26	
3	1.0	0.0	Allison, Mr. Hudson Joshua Creighton	male	30.0000	1.0	2.0	113781	151.5500	C22 C26	
4	1.0	0.0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.0000	1.0	2.0	113781	151.5500	C22 C26	
...	...	...	...	...	...	...	...	...	...	...	...
1305	3.0	0.0	Zabour, Miss. Thamine	female	NaN	1.0	0.0	2665	14.4542	NaN	
1306	3.0	0.0	Zakarian, Mr. Mapriededer	male	26.5000	0.0	0.0	2656	7.2250	NaN	
1307	3.0	0.0	Zakarian, Mr. Ortin	male	27.0000	0.0	0.0	2670	7.2250	NaN	
1308	3.0	0.0	Zimmerman, Mr. Leo	male	29.0000	0.0	0.0	315082	7.8750	NaN	
1309	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

1310 rows × 14 columns

In [62]:

```
titanic
```

Out[62]:

	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	e
0	1.0	1.0	Allen, Miss. Elisabeth Walton	female	29.0000	0.0	0.0	24160	211.3375	B5	
1	1.0	1.0	Allison, Master. Hudson Trevor	male	0.9167	1.0	2.0	113781	151.5500	C22 C26	
2	1.0	0.0	Allison, Miss. Helen Loraine	female	2.0000	1.0	2.0	113781	151.5500	C22 C26	
3	1.0	0.0	Allison, Mr. Hudson Joshua Creighton	male	30.0000	1.0	2.0	113781	151.5500	C22 C26	
4	1.0	0.0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.0000	1.0	2.0	113781	151.5500	C22 C26	
...	...	...	...	...	...	...	...	...	...	...	...
1305	3.0	0.0	Zabour, Miss. Thamine	female	NaN	1.0	0.0	2665	14.4542	NaN	
1306	3.0	0.0	Zakarian, Mr. Mapriededer	male	26.5000	0.0	0.0	2656	7.2250	NaN	
1307	3.0	0.0	Zakarian, Mr. Ortin	male	27.0000	0.0	0.0	2670	7.2250	NaN	
1308	3.0	0.0	Zimmerman, Mr. Leo	male	29.0000	0.0	0.0	315082	7.8750	NaN	
1309	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

1310 rows × 14 columns

In [6]:

```
titanic.head(10)
```

Out[6]:

	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	emb
0	1.0	1.0	Allen, Miss. Elisabeth Walton	female	29.0000	0.0	0.0	24160	211.3375	B5	
1	1.0	1.0	Allison, Master. Hudson Trevor	male	0.9167	1.0	2.0	113781	151.5500	C22 C26	
2	1.0	0.0	Allison, Miss. Helen Loraine	female	2.0000	1.0	2.0	113781	151.5500	C22 C26	
3	1.0	0.0	Allison, Mr. Hudson Joshua Creighton	male	30.0000	1.0	2.0	113781	151.5500	C22 C26	
4	1.0	0.0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.0000	1.0	2.0	113781	151.5500	C22 C26	
5	1.0	1.0	Anderson, Mr. Harry	male	48.0000	0.0	0.0	19952	26.5500	E12	
6	1.0	1.0	Andrews, Miss. Kornelia Theodosia	female	63.0000	1.0	0.0	13502	77.9583	D7	
7	1.0	0.0	Andrews, Mr. Thomas Jr	male	39.0000	0.0	0.0	112050	0.0000	A36	
8	1.0	1.0	Appleton, Mrs. Edward Dale (Charlotte Lamson)	female	53.0000	2.0	0.0	11769	51.4792	C101	
9	1.0	0.0	Artagaveytia, Mr. Ramon	male	71.0000	0.0	0.0	PC 17609	49.5042	NaN	

In [7]:

```
titanic.describe()
```

Out[7]:

	pclass	survived	age	sibsp	parch	fare	bo
count	1309.000000	1309.000000	1046.000000	1309.000000	1309.000000	1308.000000	121.0000
mean	2.294882	0.381971	29.881135	0.498854	0.385027	33.295479	160.8099
std	0.837836	0.486055	14.413500	1.041658	0.865560	51.758668	97.6969
min	1.000000	0.000000	0.166700	0.000000	0.000000	0.000000	1.0000
25%	2.000000	0.000000	21.000000	0.000000	0.000000	7.895800	72.0000
50%	3.000000	0.000000	28.000000	0.000000	0.000000	14.454200	155.0000
75%	3.000000	1.000000	39.000000	1.000000	0.000000	31.275000	256.0000
max	3.000000	1.000000	80.000000	8.000000	9.000000	512.329200	328.0000

In [11]:

```
titanic.tail()
```

Out[11]:

	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	emba
1305	3.0	0.0	Zabour, Miss. Thamine	female	NaN	1.0	0.0	2665	14.4542	NaN	
1306	3.0	0.0	Zakarian, Mr. Mapriededer	male	26.5	0.0	0.0	2656	7.2250	NaN	
1307	3.0	0.0	Zakarian, Mr. Ortin	male	27.0	0.0	0.0	2670	7.2250	NaN	
1308	3.0	0.0	Zimmerman, Mr. Leo	male	29.0	0.0	0.0	315082	7.8750	NaN	
1309	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

In [8]:

```
titanic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1310 entries, 0 to 1309
Data columns (total 14 columns):
pclass       1309 non-null float64
survived     1309 non-null float64
name         1309 non-null object
sex          1309 non-null object
age          1046 non-null float64
sibsp        1309 non-null float64
parch        1309 non-null float64
ticket       1309 non-null object
fare         1308 non-null float64
cabin        295 non-null object
embarked     1307 non-null object
boat         486 non-null object
body         121 non-null float64
home.dest    745 non-null object
dtypes: float64(7), object(7)
memory usage: 143.4+ KB
```

In [9]:

```
titanic['sex'].unique()
```

Out[9]:

```
array(['female', 'male', nan], dtype=object)
```

In [69]:

```
# Creating a dummy variable for the variable 'sex' and dropping the first one.
cont = pd.get_dummies(titanic['sex'],prefix='sex',drop_first=True)
#Adding the results to the master dataframe
titanic = pd.concat([titanic,cont],axis=1) #axis add column in titanic dataset
```

In [11]:

```
titanic
```

Out[11]:

	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	e
0	1.0	1.0	Allen, Miss. Elisabeth Walton	female	29.0000	0.0	0.0	24160	211.3375	B5	
1	1.0	1.0	Allison, Master. Hudson Trevor	male	0.9167	1.0	2.0	113781	151.5500	C22 C26	
2	1.0	0.0	Allison, Miss. Helen Loraine	female	2.0000	1.0	2.0	113781	151.5500	C22 C26	
3	1.0	0.0	Allison, Mr. Hudson Joshua Creighton	male	30.0000	1.0	2.0	113781	151.5500	C22 C26	
4	1.0	0.0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.0000	1.0	2.0	113781	151.5500	C22 C26	
...	...	...	...	...	...	...	...	...	...	...	
1305	3.0	0.0	Zabour, Miss. Thamine	female	NaN	1.0	0.0	2665	14.4542	NaN	
1306	3.0	0.0	Zakarian, Mr. Mapriededer	male	26.5000	0.0	0.0	2656	7.2250	NaN	
1307	3.0	0.0	Zakarian, Mr. Ortin	male	27.0000	0.0	0.0	2670	7.2250	NaN	
1308	3.0	0.0	Zimmerman, Mr. Leo	male	29.0000	0.0	0.0	315082	7.8750	NaN	
1309	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

1310 rows × 15 columns

In [70]:

```
#dropping sex column
titanic = titanic.drop(['sex'],1)
```

In [13]:

```
titanic
```

Out[13]:

	pclass	survived	name	age	sibsp	parch	ticket	fare	cabin	embar
0	1.0	1.0	Allen, Miss. Elisabeth Walton	29.0000	0.0	0.0	24160	211.3375	B5	
1	1.0	1.0	Allison, Master. Hudson Trevor	0.9167	1.0	2.0	113781	151.5500	C22 C26	
2	1.0	0.0	Allison, Miss. Helen Loraine	2.0000	1.0	2.0	113781	151.5500	C22 C26	
3	1.0	0.0	Allison, Mr. Hudson Joshua Creighton	30.0000	1.0	2.0	113781	151.5500	C22 C26	
4	1.0	0.0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	25.0000	1.0	2.0	113781	151.5500	C22 C26	
...	...	...	...	...	...	...	...	...	...	
1305	3.0	0.0	Zabour, Miss. Thamine	NaN	1.0	0.0	2665	14.4542	NaN	
1306	3.0	0.0	Zakarian, Mr. Mapriededer	26.5000	0.0	0.0	2656	7.2250	NaN	
1307	3.0	0.0	Zakarian, Mr. Ortin	27.0000	0.0	0.0	2670	7.2250	NaN	
1308	3.0	0.0	Zimmerman, Mr. Leo	29.0000	0.0	0.0	315082	7.8750	NaN	
1309	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

1310 rows × 14 columns

In [14]:

```
titanic['embarked'].unique()
```

Out[14]:

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

In [71]:

```
# Creating a dummy variable for the variable 'Contract' and dropping the first one.
cont = pd.get_dummies(titanic['embarked'],prefix='embarked',drop_first=True)
#Adding the results to the master dataframe
titanic = pd.concat([titanic,cont],axis=1)
```

In [25]:

```
titanic
```

Out[25]:

	pclass	survived	name	age	sibsp	parch	ticket	fare	cabin	embarked
0	1.0	1.0	Allen, Miss. Elisabeth Walton	29.0000	0.0	0.0	24160	211.3375	B5	S
1	1.0	1.0	Allison, Master. Hudson Trevor	0.9167	1.0	2.0	113781	151.5500	C22 C26	S
2	1.0	0.0	Allison, Miss. Helen Loraine	2.0000	1.0	2.0	113781	151.5500	C22 C26	S
3	1.0	0.0	Allison, Mr. Hudson Joshua Creighton	30.0000	1.0	2.0	113781	151.5500	C22 C26	S
4	1.0	0.0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	25.0000	1.0	2.0	113781	151.5500	C22 C26	S
...	...	...	...	...	...	...	...	...	...	...
1305	3.0	0.0	Zabour, Miss. Thamine	NaN	1.0	0.0	2665	14.4542	NaN	C
1306	3.0	0.0	Zakarian, Mr. Mapriededer	26.5000	0.0	0.0	2656	7.2250	NaN	C
1307	3.0	0.0	Zakarian, Mr. Ortin	27.0000	0.0	0.0	2670	7.2250	NaN	C
1308	3.0	0.0	Zimmerman, Mr. Leo	29.0000	0.0	0.0	315082	7.8750	NaN	S
1309	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

1310 rows × 16 columns

In [72]:

```
#dropping embarked column
titanic = titanic.drop(['embarked'],1)
```

In [28]:

```
titanic
```

Out[28]:

	pclass	survived	name	age	sibsp	parch	ticket	fare	cabin	boat	bo
0	1.0	1.0	Allen, Miss. Elisabeth Walton	29.0000	0.0	0.0	24160	211.3375	B5	2	Na
1	1.0	1.0	Allison, Master. Hudson Trevor	0.9167	1.0	2.0	113781	151.5500	C22 C26	11	Na
2	1.0	0.0	Allison, Miss. Helen Loraine	2.0000	1.0	2.0	113781	151.5500	C22 C26	NaN	Na
3	1.0	0.0	Allison, Mr. Hudson Joshua Creighton	30.0000	1.0	2.0	113781	151.5500	C22 C26	NaN	135
4	1.0	0.0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	25.0000	1.0	2.0	113781	151.5500	C22 C26	NaN	Na
...	...	...	...	...	...	...	...	...	...	...	
1305	3.0	0.0	Zabour, Miss. Thamine	NaN	1.0	0.0	2665	14.4542	NaN	NaN	Na
1306	3.0	0.0	Zakarian, Mr. Mapriededer	26.5000	0.0	0.0	2656	7.2250	NaN	NaN	304
1307	3.0	0.0	Zakarian, Mr. Ortin	27.0000	0.0	0.0	2670	7.2250	NaN	NaN	Na
1308	3.0	0.0	Zimmerman, Mr. Leo	29.0000	0.0	0.0	315082	7.8750	NaN	NaN	Na
1309	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na

1310 rows × 15 columns

In [17]:

```
titanic['body'].unique()
```

Out[17]:

```
array([ nan, 135.,  22., 124., 148., 208., 172., 269.,  62., 133., 275.,
       147., 110., 307.,  38.,  80.,  45., 258., 126., 292., 175., 249.,
       230., 122., 263., 234., 189., 166., 207., 232.,  16., 109.,  96.,
        46., 245., 169., 174.,  97.,  18., 130.,  17., 295., 286., 236.,
       322., 297., 155., 305.,  19.,  75.,  35., 256., 149., 283., 165.,
       108., 121.,  52., 209., 271.,  43.,  15., 101., 287.,  81., 294.,
       293., 190.,  72., 103.,  79., 259., 260., 142., 299., 171.,   9.,
       197.,  51., 187.,  68.,  47.,  98., 188.,  69., 306., 120., 143.,
       156., 285.,  37.,  58.,  70., 196., 153.,  61.,  53., 201., 309.,
       181., 173.,  89.,   4., 206., 327., 119.,   7.,  32.,  67., 284.,
       261., 176.,  50.,   1., 255., 298., 314.,  14., 131., 312., 328.,
       304.])
```

In [78]:

```
# We have created dummies for the below variables, so we can drop them
# string system cant read , so we are removing these columns
titanic = titanic.drop(['name'], axis=1)
```

In [79]:

```
titanic
```

Out[79]:

	pclass	survived	age	sibsp	parch	ticket	fare	sex_male	embarked_Q	emba
0	1.0	1.0	29.0000	0.0	0.0	24160	211.3375	0	0	
1	1.0	1.0	0.9167	1.0	2.0	113781	151.5500	1	0	
2	1.0	0.0	2.0000	1.0	2.0	113781	151.5500	0	0	
3	1.0	0.0	30.0000	1.0	2.0	113781	151.5500	1	0	
4	1.0	0.0	25.0000	1.0	2.0	113781	151.5500	0	0	
...	...	...	...	...	...	...	...	...	...	
1305	3.0	0.0	NaN	1.0	0.0	2665	14.4542	0	0	
1306	3.0	0.0	26.5000	0.0	0.0	2656	7.2250	1	0	
1307	3.0	0.0	27.0000	0.0	0.0	2670	7.2250	1	0	
1308	3.0	0.0	29.0000	0.0	0.0	315082	7.8750	1	0	
1309	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0	0	

1310 rows × 10 columns

In [80]:

```
titanic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1310 entries, 0 to 1309
Data columns (total 10 columns):
pclass        1309 non-null float64
survived      1309 non-null float64
age           1046 non-null float64
sibsp         1309 non-null float64
parch         1309 non-null float64
ticket        1309 non-null object
fare          1308 non-null float64
sex_male      1310 non-null uint8
embarked_Q    1310 non-null uint8
embarked_S    1310 non-null uint8
dtypes: float64(6), object(1), uint8(3)
memory usage: 75.6+ KB
```

In [81]:

```
#The varaible was imported as a string we need to convert it to float
#telecom['TotalCharges'] =telecom['TotalCharges'].convert_objects(convert_numeric=True) - i
titanic['ticket'] = pd.to_numeric(titanic['ticket'], errors='coerce')
#telecom['tenure'] = telecom['tenure'].astype(int).astype(float)
```

In [82]:

```
titanic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1310 entries, 0 to 1309
Data columns (total 10 columns):
pclass        1309 non-null float64
survived      1309 non-null float64
age           1046 non-null float64
sibsp         1309 non-null float64
parch         1309 non-null float64
ticket        957 non-null float64
fare          1308 non-null float64
sex_male      1310 non-null uint8
embarked_Q    1310 non-null uint8
embarked_S    1310 non-null uint8
dtypes: float64(7), uint8(3)
memory usage: 75.6 KB
```

In [83]:

```
titanic.describe()
```

Out[83]:

	pclass	survived	age	sibsp	parch	ticket	
count	1309.000000	1309.000000	1046.000000	1309.000000	1309.000000	9.570000e+02	1308.00
mean	2.294882	0.381971	29.881135	0.498854	0.385027	2.490391e+05	33.29
std	0.837836	0.486055	14.413500	1.041658	0.865560	4.426853e+05	51.75
min	1.000000	0.000000	0.166700	0.000000	0.000000	6.800000e+02	0.00
25%	2.000000	0.000000	21.000000	0.000000	0.000000	1.995000e+04	7.89
50%	3.000000	0.000000	28.000000	0.000000	0.000000	2.346040e+05	14.45
75%	3.000000	1.000000	39.000000	1.000000	0.000000	3.474680e+05	31.27
max	3.000000	1.000000	80.000000	8.000000	9.000000	3.101298e+06	512.32

In [84]:

```
#the reason for outliers 1)manual entry mistake - it will come as numerical perspective
# Checking for outliers in the continuous variables
out_titanic = titanic[['age','ticket','fare']]
```

In [85]:

```
# Checking outliers at 25%,50%,75%,90%,95% and 99% - just to check if there is any outlier
out_titanic.describe(percentiles=[.25,.5,.75,.90,.95,.99])
```

Out[85]:

	age	ticket	fare
count	1046.000000	9.570000e+02	1308.000000
mean	29.881135	2.490391e+05	33.295479
std	14.413500	4.426853e+05	51.758668
min	0.166700	6.800000e+02	0.000000
25%	21.000000	1.995000e+04	7.895800
50%	28.000000	2.346040e+05	14.454200
75%	39.000000	3.474680e+05	31.275000
90%	50.000000	3.652358e+05	78.050820
95%	57.000000	3.752406e+05	133.650000
99%	65.000000	3.101295e+06	262.375000
max	80.000000	3.101298e+06	512.329200

In [86]:

```
titanic.isnull().sum()
```

Out[86]:

```
pclass          1
survived        1
age           264
sibsp           1
parch           1
ticket        353
fare            2
sex_male        0
embarked_Q      0
embarked_S      0
dtype: int64
```

In [93]:

```
round(100*(titanic.isnull().sum()/len(titanic.index)), 2)
```

Out[93]:

```
pclass        0.0
survived      0.0
age           0.0
sibsp         0.0
parch         0.0
ticket        0.0
fare          0.0
sex_male      0.0
embarked_Q    0.0
embarked_S    0.0
dtype: float64
```

In [92]:

```
import numpy as np
titanic = titanic[~np.isnan(titanic['fare'])]
```

In [38]:

```
round(100*(titanic.isnull().sum()/len(titanic.index)), 2)
```

Out[38]:

```
pclass         0.00
survived       0.00
age            0.00
sibsp          0.00
parch          0.00
ticket        28.01
fare           0.10
sex_male       0.00
embarked_Q     0.00
embarked_S     0.00
dtype: float64
```

In [42]:

```
titanic = titanic[~np.isnan(titanic['fare'])]
```

In [43]:

```
round(100*(titanic.isnull().sum()/len(titanic.index)), 2)
```

Out[43]:

```
pclass        0.0
survived      0.0
age           0.0
sibsp         0.0
parch         0.0
ticket        0.0
fare          0.0
sex_male      0.0
embarked_Q    0.0
embarked_S    0.0
dtype: float64
```

In [44]:

```
titanic.describe()
```

Out[44]:

	pclass	survived	age	sibsp	parch	ticket	fare
count	752.000000	752.000000	752.000000	752.000000	752.000000	7.520000e+02	752.000000
mean	2.243351	0.398936	29.609486	0.501330	0.405585	2.623222e+05	31.335123
std	0.843607	0.490006	14.394254	0.886351	0.820263	4.913279e+05	41.335246
min	1.000000	0.000000	0.333300	0.000000	0.000000	6.800000e+02	0.000000
25%	1.000000	0.000000	21.000000	0.000000	0.000000	2.416000e+04	8.050000
50%	3.000000	0.000000	28.000000	0.000000	0.000000	2.319190e+05	14.454200
75%	3.000000	1.000000	38.000000	1.000000	1.000000	3.470820e+05	31.000000
max	3.000000	1.000000	80.000000	4.000000	5.000000	3.101298e+06	263.000000

In [94]:

```
scale = titanic[['age','ticket','fare']]
```

In [95]:

```
titanic.describe()
```

Out[95]:

	pclass	survived	age	sibsp	parch	ticket	fare
count	752.000000	752.000000	752.000000	752.000000	752.000000	7.520000e+02	752.000000
mean	2.243351	0.398936	29.609486	0.501330	0.405585	2.623222e+05	31.335123
std	0.843607	0.490006	14.394254	0.886351	0.820263	4.913279e+05	41.335246
min	1.000000	0.000000	0.333300	0.000000	0.000000	6.800000e+02	0.000000
25%	1.000000	0.000000	21.000000	0.000000	0.000000	2.416000e+04	8.050000
50%	3.000000	0.000000	28.000000	0.000000	0.000000	2.319190e+05	14.454200
75%	3.000000	1.000000	38.000000	1.000000	1.000000	3.470820e+05	31.000000
max	3.000000	1.000000	80.000000	4.000000	5.000000	3.101298e+06	263.000000

In [96]:

```
normalized_scale=(scale-scale.mean())/scale.std()
```

In [97]:

```
normalized_scale.describe()
```

Out[97]:

	age	ticket	fare
count	7.520000e+02	7.520000e+02	7.520000e+02
mean	1.092599e-16	-4.803708e-17	-1.553131e-16
std	1.000000e+00	1.000000e+00	1.000000e+00
min	-2.033880e+00	-5.325206e-01	-7.580727e-01
25%	-5.981196e-01	-4.847317e-01	-5.633237e-01
50%	-1.118145e-01	-6.187970e-02	-4.083905e-01
75%	5.829072e-01	1.725116e-01	-8.107439e-03
max	3.500738e+00	5.778169e+00	5.604536e+00

In [98]:

```
titanic = titanic.drop(['age','ticket','fare'], 1)
```

In [99]:

```
titanic = pd.concat([titanic,normalized_scale],axis=1)
```

In [100]:

```
titanic
```

Out[100]:

	pclass	survived	sibsp	parch	sex_male	embarked_Q	embarked_S	age	ticke
0	1.0	1.0	0.0	0.0	0	0	1	-0.042342	-0.48473
1	1.0	1.0	1.0	2.0	1	0	1	-1.993350	-0.30232
2	1.0	0.0	1.0	2.0	0	0	1	-1.918091	-0.30232
3	1.0	0.0	1.0	2.0	1	0	1	0.027130	-0.30232
4	1.0	0.0	1.0	2.0	0	0	1	-0.320231	-0.30232
...	...	...	...	...	...	...	...	...	.
1301	3.0	0.0	0.0	0.0	1	0	0	1.103948	-0.52855
1304	3.0	0.0	1.0	0.0	0	0	0	-1.049689	-0.52848
1306	3.0	0.0	0.0	0.0	1	0	0	-0.216023	-0.52849
1307	3.0	0.0	0.0	0.0	1	0	0	-0.181287	-0.52847
1308	3.0	0.0	0.0	0.0	1	0	1	-0.042342	0.10738

752 rows × 10 columns

In [101]:

```
titanic.describe()
```

Out[101]:

	pclass	survived	sibsp	parch	sex_male	embarked_Q	embarked_S	
count	752.000000	752.000000	752.000000	752.000000	752.00000	752.000000	752.000000	7
mean	2.243351	0.398936	0.501330	0.405585	0.62766	0.063830	0.765957	
std	0.843607	0.490006	0.886351	0.820263	0.48375	0.244612	0.423681	
min	1.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	-2
25%	1.000000	0.000000	0.000000	0.000000	0.00000	0.000000	1.000000	-
50%	3.000000	0.000000	0.000000	0.000000	1.00000	0.000000	1.000000	-
75%	3.000000	1.000000	1.000000	1.000000	1.00000	0.000000	1.000000	
max	3.000000	1.000000	4.000000	5.000000	1.00000	1.000000	1.000000	3

In [102]:

```
sur = (sum(titanic['survived'])/len(titanic['survived'].index))*100
```

In [103]:

```
sur
```

Out[103]:

```
39.8936170212766
```

In [104]:

```
from sklearn.model_selection import train_test_split
```

In [105]:

```
# Putting feature variable to X
#axis means column
X = titanic.drop(['survived'],axis=1)

# Putting response variable to y
y = titanic['survived']
```

In [106]:

```
y.head()
```

Out[106]:

```
0    1.0
1    1.0
2    0.0
3    0.0
4    0.0
Name: survived, dtype: float64
```

In [107]:

```
# Splitting the data into train and test
X_train, X_test, y_train, y_test = train_test_split(X,y, train_size=0.7,test_size=0.3,randc
```

In [110]:

```
# Let's run the model using the selected variables
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
logsk = LogisticRegression()
logsk.fit(X_train, y_train)#c=1/lambda
```

Out[110]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [111]:

```
y_pred1 = logsk.predict(X_test)
```

In [114]:

```
y_pred1
```

Out[114]:

```
array([0., 1., 0., 0., 0., 0., 1., 0., 0., 1., 0., 0., 0., 0., 0., 0., 1.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 1., 1., 0., 0., 0., 1.,
       0., 0., 0., 1., 0., 0., 1., 0., 0., 0., 1., 0., 0., 0., 1., 1., 0.,
       0., 0., 0., 1., 1., 1., 1., 0., 1., 0., 0., 1., 0., 0., 1., 0., 1.,
       0., 1., 0., 0., 1., 0., 1., 1., 0., 0., 0., 0., 0., 1., 0., 1., 1.,
       0., 0., 1., 0., 0., 0., 0., 0., 1., 0., 0., 0., 1., 0., 0., 1., 0.,
       0., 1., 0., 0., 0., 1., 0., 0., 1., 0., 0., 1., 0., 0., 0., 1., 0.,
       0., 1., 0., 0., 0., 0., 1., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0.,
       0., 1., 0., 1., 0., 1., 0., 1., 1., 1., 1., 0., 0., 0., 0., 0., 0.,
       1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 0., 0., 0., 1., 0., 1., 0.,
       0., 0., 1., 0., 0., 1., 0., 1., 0., 0., 0., 0., 0., 1., 1., 0., 1.,
       0., 0., 0., 0., 1., 0., 0., 1., 0., 0., 1., 1., 0., 0., 0., 1., 1.,
       0., 0., 0., 0., 1., 0., 1., 0., 1., 0., 1., 1., 1., 1., 0., 0., 0.,
       0., 1., 0., 0., 0.])
```

In [115]:

```
from sklearn import metrics
metrics.accuracy_score( y_test, y_pred1)*100
```

Out[115]:

```
77.43362831858407
```

In [116]:

```
y_pred = logsk.predict_proba(X_test)
```

In [117]:

```
y_pred
```

Out[117]:

```
array([[0.92736893, 0.07263107],
       [0.1983692 , 0.8016308 ],
       [0.88177822, 0.11822178],
       [0.54606494, 0.45393506],
       [0.86837756, 0.13162244],
       [0.79078965, 0.20921035],
       [0.31758141, 0.68241859],
       [0.93600272, 0.06399728],
       [0.82296392, 0.17703608],
       [0.40125721, 0.59874279],
       [0.90203191, 0.09796809],
       [0.90168758, 0.09831242],
       [0.69057886, 0.30942114],
       [0.88487459, 0.11512541],
       [0.75295831, 0.24704169],
       [0.86485844, 0.13514156],
       [0.21806665, 0.78193335],
       [0.70679244, 0.29320756],
```

In [118]:

```
# Converting y_pred to a dataframe which is an array
y_pred_df = pd.DataFrame(y_pred)
```

In [119]:

```
# Converting to column dataframe
y_pred_1 = y_pred_df.iloc[:,[1]]  #iloc - picking surived only
```

In [120]:

```
y_pred_df
```

Out[120]:

	0	1
0	0.927369	0.072631
1	0.198369	0.801631
2	0.881778	0.118222
3	0.546065	0.453935
4	0.868378	0.131622
...	...	...
221	0.828368	0.171632
222	0.250200	0.749800
223	0.873471	0.126529
224	0.506849	0.493151
225	0.907575	0.092425

226 rows × 2 columns

In [122]:

```
y_pred_1
```

Out[122]:

	1
0	0.072631
1	0.801631
2	0.118222
3	0.453935
4	0.131622
...	...
221	0.171632
222	0.749800
223	0.126529
224	0.493151
225	0.092425

226 rows × 1 columns

In [123]:

```
# Converting y_test to dataframe
y_test_df = pd.DataFrame(y_test)
```

In [124]:

```
# Putting CustID to index
y_test_df['Name'] = y_test_df.index
```

In [125]:

```
# Removing index for both dataframes to append them side by side
y_pred_1.reset_index(drop=True, inplace=True)
y_test_df.reset_index(drop=True, inplace=True)
```

In [127]:

```
y_pred_1
```

Out[127]:

	1
0	0.072631
1	0.801631
2	0.118222
3	0.453935
4	0.131622
...	...
221	0.171632
222	0.749800
223	0.126529
224	0.493151
225	0.092425

226 rows × 1 columns

In [128]:

```
# Appending y_test_df and y_pred_1
y_pred_final = pd.concat([y_test_df,y_pred_1],axis=1)
```

In [129]:

```
# Renaming the column
y_pred_final= y_pred_final.rename(columns={ 1 : 'Survived_Prod'})
```

In [130]:

```
y_pred_final
```

Out[130]:

	survived	Name	Survived_Prod
0	0.0	1289	0.072631
1	1.0	181	0.801631
2	0.0	639	0.118222
3	1.0	29	0.453935
4	0.0	814	0.131622
...	...	...	...
221	0.0	414	0.171632
222	1.0	342	0.749800
223	1.0	1017	0.126529
224	0.0	1231	0.493151
225	1.0	1120	0.092425

226 rows × 3 columns

In [131]:

```
# Creating new column 'predicted' with 1 if Churn_Prob>0.5 else 0
y_pred_final['predicted'] = y_pred_final.Survived_Prod.map( lambda x: 1 if x > 0.5 else 0)#
```

In [132]:

```
y_pred_final
```

Out[132]:

	survived	Name	Survived_Prod	predicted
0	0.0	1289	0.072631	0
1	1.0	181	0.801631	1
2	0.0	639	0.118222	0
3	1.0	29	0.453935	0
4	0.0	814	0.131622	0
...	...	...	...	...
221	0.0	414	0.171632	0
222	1.0	342	0.749800	1
223	1.0	1017	0.126529	0
224	0.0	1231	0.493151	0
225	1.0	1120	0.092425	0

226 rows × 4 columns

In [133]:

```
from sklearn import metrics
```

In [135]:

```
# Confusion matrix
confusion = metrics.confusion_matrix( y_pred_final.survived, y_pred_final.predicted )
confusion
```

Out[135]:

```
array([[118,  21],
       [ 30,  57]], dtype=int64)
```

In [136]:

```
#Let's check the overall accuracy.
metrics.accuracy_score( y_pred_final.survived, y_pred_final.predicted)
```

Out[136]:

```
0.7743362831858407
```

In [137]:

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

In [144]:

```
# Let's see the sensitivity of our logistic regression model
Sen=TP / float(TP+FN)
```

In [146]:

```
#Precision
Prec=TP / float(TP + FP)
```

In [145]:

```
Sen
```

Out[145]:

```
0.7972972972972973
```

In [147]:

```
Prec
```

Out[147]:

```
0.8489208633093526
```

In [148]:

```
F1 = 2 * ((Prec*Sen) / (Prec+Sen))
```

In [149]:

```
F1
```

Out[149]:

```
0.8222996515679443
```

In [150]:

```
# ROC Curve
```

In [154]:

```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

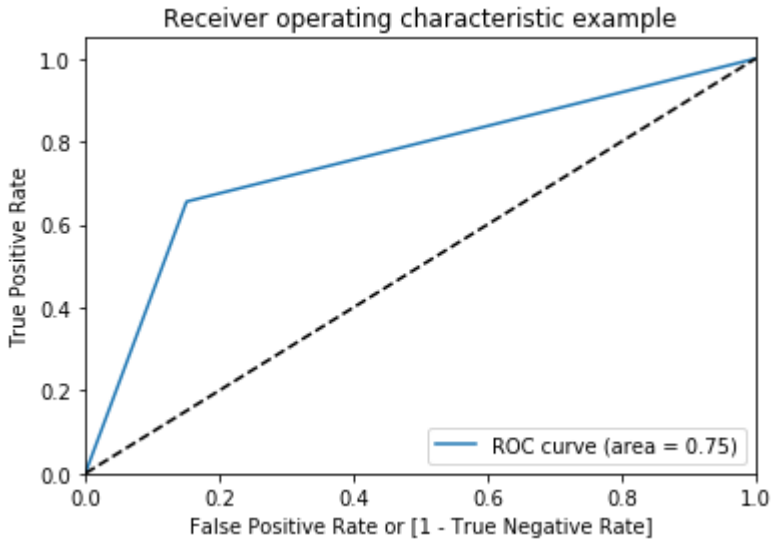
In [155]:

```
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=(6, 4))
    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 fpr, tpr, thresholds
```

In [156]:

```
draw_roc(y_pred_final.survived, y_pred_final.predicted)
```



Out[156]:

```
(array([0.        , 0.15107914, 1.        ]),
 array([0.        , 0.65517241, 1.        ]),
 array([2, 1, 0], dtype=int64))
```

In [ ]: