In [1]:

```
# Importing Pandas and NumPy
import pandas as pd
import numpy as np
```

In [2]:

```
# Importing dataset
data = pd.read_csv('D:/Python/Dataset/nrippner-titanic-disaster-dataset/titanic.csv')
```

In [3]:

```
data
```

Out[3]:

	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 [4]:

```
data.describe()
```

Out[4]:

	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 [5]:

```
data.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 [6]:

```
data['sex'].value_counts()
```

Out[6]:

```
male      843
female    466
Name: sex, dtype: int64
```

In [7]:

```python
# Converting Male to 1 and Female to 0
data['sex'] = data['sex'].map({'male': 1, 'female': 0})
#The varaible was imported as a string we need to convert it to float
#data['sex'] =pd.to_numeric(data['sex'],errors='coerce')
```

In [8]:

```python
data
```

Out[8]:

	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	em
0	1.0	1.0	Allen, Miss. Elisabeth Walton	0.0	29.0000	0.0	0.0	24160	211.3375	B5	
1	1.0	1.0	Allison, Master. Hudson Trevor	1.0	0.9167	1.0	2.0	113781	151.5500	C22 C26	
2	1.0	0.0	Allison, Miss. Helen Loraine	0.0	2.0000	1.0	2.0	113781	151.5500	C22 C26	
3	1.0	0.0	Allison, Mr. Hudson Joshua Creighton	1.0	30.0000	1.0	2.0	113781	151.5500	C22 C26	
4	1.0	0.0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	0.0	25.0000	1.0	2.0	113781	151.5500	C22 C26	
...	...	...	...	...	...	...	...	...	...	...	
1305	3.0	0.0	Zabour, Miss. Thamine	0.0	NaN	1.0	0.0	2665	14.4542	NaN	
1306	3.0	0.0	Zakarian, Mr. Mapriededer	1.0	26.5000	0.0	0.0	2656	7.2250	NaN	
1307	3.0	0.0	Zakarian, Mr. Ortin	1.0	27.0000	0.0	0.0	2670	7.2250	NaN	
1308	3.0	0.0	Zimmerman, Mr. Leo	1.0	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 [9]:

```
data.describe()
```

Out[9]:

	pclass	survived	sex	age	sibsp	parch	f
count	1309.000000	1309.000000	1309.000000	1046.000000	1309.000000	1309.000000	1308.000
mean	2.294882	0.381971	0.644003	29.881135	0.498854	0.385027	33.295
std	0.837836	0.486055	0.478997	14.413500	1.041658	0.865560	51.758
min	1.000000	0.000000	0.000000	0.166700	0.000000	0.000000	0.000
25%	2.000000	0.000000	0.000000	21.000000	0.000000	0.000000	7.895
50%	3.000000	0.000000	1.000000	28.000000	0.000000	0.000000	14.454
75%	3.000000	1.000000	1.000000	39.000000	1.000000	0.000000	31.275
max	3.000000	1.000000	1.000000	80.000000	8.000000	9.000000	512.329

In [10]:

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

In [11]:

```
data
```

Out[11]:

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

1310 rows × 16 columns

In [12]:

```
data.describe()
```

Out[12]:

	pclass	survived	sex	age	sibsp	parch	f
count	1309.000000	1309.000000	1309.000000	1046.000000	1309.000000	1309.000000	1308.000
mean	2.294882	0.381971	0.644003	29.881135	0.498854	0.385027	33.295
std	0.837836	0.486055	0.478997	14.413500	1.041658	0.865560	51.758
min	1.000000	0.000000	0.000000	0.166700	0.000000	0.000000	0.000
25%	2.000000	0.000000	0.000000	21.000000	0.000000	0.000000	7.895
50%	3.000000	0.000000	1.000000	28.000000	0.000000	0.000000	14.454
75%	3.000000	1.000000	1.000000	39.000000	1.000000	0.000000	31.275
max	3.000000	1.000000	1.000000	80.000000	8.000000	9.000000	512.329

In [13]:

```
data['ticket'].value_counts()
```

Out[13]:

```
CA. 2343        11
1601             8
CA 2144          8
347077           7
S.O.C. 14879     7
                ..
364850           1
315083           1
350035           1
350404           1
31028            1
Name: ticket, Length: 929, dtype: int64
```

In [14]:

```
# We have created dummies for the below variables, so we can drop them
data = data.drop(['embarked'], 1)
```

In [15]:

```
data
```

Out[15]:

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

1310 rows × 15 columns

In [16]:

```
# Checking outliers at 25%,50%,75%,90%,95% and 99%
data.describe(percentiles=[.25,.5,.75,.90,.95,.99])
```

Out[16]:

	pclass	survived	sex	age	sibsp	parch	f
count	1309.000000	1309.000000	1309.000000	1046.000000	1309.000000	1309.000000	1308.000
mean	2.294882	0.381971	0.644003	29.881135	0.498854	0.385027	33.295
std	0.837836	0.486055	0.478997	14.413500	1.041658	0.865560	51.758
min	1.000000	0.000000	0.000000	0.166700	0.000000	0.000000	0.000
25%	2.000000	0.000000	0.000000	21.000000	0.000000	0.000000	7.895
50%	3.000000	0.000000	1.000000	28.000000	0.000000	0.000000	14.454
75%	3.000000	1.000000	1.000000	39.000000	1.000000	0.000000	31.275
90%	3.000000	1.000000	1.000000	50.000000	1.000000	2.000000	78.050
95%	3.000000	1.000000	1.000000	57.000000	2.000000	2.000000	133.650
99%	3.000000	1.000000	1.000000	65.000000	5.000000	4.000000	262.375
max	3.000000	1.000000	1.000000	80.000000	8.000000	9.000000	512.329

In [17]:

```
# Adding up the missing values (column-wise)
data.isnull().sum()
```

Out[17]:

```
pclass          1
survived        1
name            1
sex             1
age           264
sibsp           1
parch           1
ticket          1
fare            2
cabin        1015
boat          824
body         1189
home.dest     565
embarked_Q      0
embarked_S      0
dtype: int64
```

In [18]:

```
# Checking the percentage of missing values
round(100*(data.isnull().sum()/len(data.index)), 2)
```

Out[18]:

```
pclass          0.08
survived        0.08
name            0.08
sex             0.08
age            20.15
sibsp           0.08
parch           0.08
ticket          0.08
fare            0.15
cabin          77.48
boat           62.90
body           90.76
home.dest      43.13
embarked_Q      0.00
embarked_S      0.00
dtype: float64
```

In [88]:

```
data['home.dest'].value_counts()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, k
ey, method, tolerance)
   2896             try:
-> 2897                 return self._engine.get_loc(key)
   2898             except KeyError:

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHa
shTable.get_item()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHa
shTable.get_item()

KeyError: 'home.dest'

During handling of the above exception, another exception occurred:

KeyError                                  Traceback (most recent call last)
<ipython-input-88-114bb5a0bd52> in <module>
----> 1 data['home.dest'].value_counts()

~\Anaconda3\lib\site-packages\pandas\core\frame.py in __getitem__(self, key)
   2978             if self.columns.nlevels > 1:
   2979                 return self._getitem_multilevel(key)
-> 2980             indexer = self.columns.get_loc(key)
   2981             if is_integer(indexer):
   2982                 indexer = [indexer]

~\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, k
ey, method, tolerance)
   2897                 return self._engine.get_loc(key)
   2898             except KeyError:
-> 2899                 return self._engine.get_loc(self._maybe_cast_indexer
(key))
   2900         indexer = self.get_indexer([key], method=method, tolerance=t
olerance)
   2901         if indexer.ndim > 1 or indexer.size > 1:

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHa
shTable.get_item()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHa
shTable.get_item()

KeyError: 'home.dest'
```

In [20]:

```
# We have some unwanted columns , so we can drop them
data = data.drop(['home.dest','cabin','boat'], 1)
```

In [21]:

```
data
```

Out[21]:

	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	body	eml
0	1.0	1.0	Allen, Miss. Elisabeth Walton	0.0	29.0000	0.0	0.0	24160	211.3375	NaN	
1	1.0	1.0	Allison, Master. Hudson Trevor	1.0	0.9167	1.0	2.0	113781	151.5500	NaN	
2	1.0	0.0	Allison, Miss. Helen Loraine	0.0	2.0000	1.0	2.0	113781	151.5500	NaN	
3	1.0	0.0	Allison, Mr. Hudson Joshua Creighton	1.0	30.0000	1.0	2.0	113781	151.5500	135.0	
4	1.0	0.0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	0.0	25.0000	1.0	2.0	113781	151.5500	NaN	
...	...	...	...	...	...	...	...	...	...	...	
1305	3.0	0.0	Zabour, Miss. Thamine	0.0	NaN	1.0	0.0	2665	14.4542	NaN	
1306	3.0	0.0	Zakarian, Mr. Mapriededer	1.0	26.5000	0.0	0.0	2656	7.2250	304.0	
1307	3.0	0.0	Zakarian, Mr. Ortin	1.0	27.0000	0.0	0.0	2670	7.2250	NaN	
1308	3.0	0.0	Zimmerman, Mr. Leo	1.0	29.0000	0.0	0.0	315082	7.8750	NaN	
1309	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

1310 rows × 12 columns

In [22]:

```
data['age'].tail(50)
```

Out[22]:

```
1260    18.0
1261    63.0
1262     NaN
1263    11.5
1264    40.5
1265    10.0
1266    36.0
1267    30.0
1268     NaN
1269    33.0
1270    28.0
1271    28.0
1272    47.0
1273    18.0
1274    31.0
1275    16.0
1276    31.0
1277    22.0
1278    20.0
1279    14.0
1280    22.0
1281    22.0
1282     NaN
1283     NaN
1284     NaN
1285    32.5
1286    38.0
1287    51.0
1288    18.0
1289    21.0
1290    47.0
1291     NaN
1292     NaN
1293     NaN
1294    28.5
1295    21.0
1296    27.0
1297     NaN
1298    36.0
1299    27.0
1300    15.0
1301    45.5
1302     NaN
1303     NaN
1304    14.5
1305     NaN
1306    26.5
1307    27.0
1308    29.0
1309     NaN
Name: age, dtype: float64
```

In [23]:

```
data.describe()
```

Out[23]:

	pclass	survived	sex	age	sibsp	parch	f
count	1309.000000	1309.000000	1309.000000	1046.000000	1309.000000	1309.000000	1308.000
mean	2.294882	0.381971	0.644003	29.881135	0.498854	0.385027	33.295
std	0.837836	0.486055	0.478997	14.413500	1.041658	0.865560	51.758
min	1.000000	0.000000	0.000000	0.166700	0.000000	0.000000	0.000
25%	2.000000	0.000000	0.000000	21.000000	0.000000	0.000000	7.895
50%	3.000000	0.000000	1.000000	28.000000	0.000000	0.000000	14.454
75%	3.000000	1.000000	1.000000	39.000000	1.000000	0.000000	31.275
max	3.000000	1.000000	1.000000	80.000000	8.000000	9.000000	512.329

In [24]:

```
data.describe(percentiles=[.25,.5,.75,.90,.95,.99])
```

Out[24]:

	pclass	survived	sex	age	sibsp	parch	f
count	1309.000000	1309.000000	1309.000000	1046.000000	1309.000000	1309.000000	1308.000
mean	2.294882	0.381971	0.644003	29.881135	0.498854	0.385027	33.295
std	0.837836	0.486055	0.478997	14.413500	1.041658	0.865560	51.758
min	1.000000	0.000000	0.000000	0.166700	0.000000	0.000000	0.000
25%	2.000000	0.000000	0.000000	21.000000	0.000000	0.000000	7.895
50%	3.000000	0.000000	1.000000	28.000000	0.000000	0.000000	14.454
75%	3.000000	1.000000	1.000000	39.000000	1.000000	0.000000	31.275
90%	3.000000	1.000000	1.000000	50.000000	1.000000	2.000000	78.050
95%	3.000000	1.000000	1.000000	57.000000	2.000000	2.000000	133.650
99%	3.000000	1.000000	1.000000	65.000000	5.000000	4.000000	262.375
max	3.000000	1.000000	1.000000	80.000000	8.000000	9.000000	512.329

In [25]:

```
data.isnull().sum()
```

Out[25]:

```
pclass          1
survived        1
name            1
sex             1
age           264
sibsp           1
parch           1
ticket          1
fare            2
body         1189
embarked_Q      0
embarked_S      0
dtype: int64
```

In [26]:

```
data['age'].fillna(data['age'].mean(),inplace=True)
```

In [27]:

```
data['fare'].fillna(data['fare'].mean(),inplace=True)
```

In [28]:

```
data.isnull().sum()
```

Out[28]:

```
pclass          1
survived        1
name            1
sex             1
age             0
sibsp           1
parch           1
ticket          1
fare            0
body         1189
embarked_Q      0
embarked_S      0
dtype: int64
```

In [29]:

```
# Normalising continuous features
df = data[['sex','age','fare']]
```

In [30]:

```
normalized_df=(df-df.mean())/df.std()
```

In [31]:

```
data = data.drop(['sex','age','fare'], 1)
```

In [32]:

```
data = pd.concat([data,normalized_df],axis=1)
```

In [33]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1310 entries, 0 to 1309
Data columns (total 12 columns):
pclass        1309 non-null float64
survived      1309 non-null float64
name          1309 non-null object
sibsp         1309 non-null float64
parch         1309 non-null float64
ticket        1309 non-null object
body          121 non-null float64
embarked_Q    1310 non-null uint8
embarked_S    1310 non-null uint8
sex           1309 non-null float64
age           1310 non-null float64
fare          1310 non-null float64
dtypes: float64(8), object(2), uint8(2)
memory usage: 105.0+ KB
```

In [34]:

```
# Removing NaN in Survived rows
data = data[~np.isnan(data['survived'])]
```

In [35]:

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

In [36]:

```
survived
```

Out[36]:

```
38.19709702062643
```

In [37]:

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

In [38]:

```
# Putting feature variable to X
X = data.drop(['name','survived','ticket','body'],axis=1)

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

In [39]:

```
X.isnull().sum()
```

Out[39]:

```
pclass        0
sibsp         0
parch         0
embarked_Q    0
embarked_S    0
sex           0
age           0
fare          0
dtype: int64
```

In [40]:

```
# 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,
```

In [41]:

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

In [42]:

```
# 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:\Users\Vicky\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:
432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Speci
fy a solver to silence this warning.
  FutureWarning)
```

Out[42]:

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

In [43]:

```
# Predicted probabilities
y_pred = logsk.predict_proba(X_test)
```

In [44]:

```
# Predicted probabilities
y_pred1 = logsk.predict(X_test)
```

In [45]:

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

Out[45]:

```
76.33587786259542
```

In [46]:

```
logsk1 = LogisticRegression(C=10000,penalty='l1')
logsk1.fit(X_train, y_train)
```

```
C:\Users\Vicky\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:
432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Speci
fy a solver to silence this warning.
  FutureWarning)
```

Out[46]:

```
LogisticRegression(C=10000, class_weight=None, dual=False, fit_intercept=Tru
e,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l1',
                   random_state=None, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [47]:

```
# Predicted probabilities
y_pred1 = logsk1.predict_proba(X_test)
```

In [48]:

```
y_pred1
```

Out[48]:

```
array([[0.39286987, 0.60713013],
       [0.92442351, 0.07557649],
       [0.91191923, 0.08808077],
       [0.39401712, 0.60598288],
       [0.03604694, 0.96395306],
       [0.78627565, 0.21372435],
       [0.85648477, 0.14351523],
       [0.49035479, 0.50964521],
       [0.88571314, 0.11428686],
       [0.51616802, 0.48383198],
       [0.09139541, 0.90860459],
       [0.4032952 , 0.5967048 ],
       [0.84052779, 0.15947221],
       [0.55398624, 0.44601376],
       [0.45654732, 0.54345268],
       [0.18422854, 0.81577146],
       [0.91236422, 0.08763578],
       [0.96045032, 0.03954968],
```

In [49]:

```
metrics.accuracy_score( y_test, y_pred1)*100
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-49-eddbce20593e> in <module>
----> 1 metrics.accuracy_score( y_test, y_pred1)*100

~\Anaconda3\lib\site-packages\sklearn\metrics\classification.py in accuracy_
score(y_true, y_pred, normalize, sample_weight)
    174 
    175     # Compute accuracy for each possible representation
--> 176     y_type, y_true, y_pred = _check_targets(y_true, y_pred)
    177     check_consistent_length(y_true, y_pred, sample_weight)
    178     if y_type.startswith('multilabel'):

~\Anaconda3\lib\site-packages\sklearn\metrics\classification.py in _check_ta
rgets(y_true, y_pred)
     79     if len(y_type) > 1:
     80         raise ValueError("Classification metrics can't handle a mix
 of {0} "
---> 81                          "and {1} targets".format(type_true, type_pr
ed))
     82 
     83     # We can't have more than one value on y_type => The set is no m
ore needed

ValueError: Classification metrics can't handle a mix of binary and continuo
us-multioutput targets
```

In [50]:

```
y_pred2 = logsk.predict(X_test)
```

In [51]:

```
metrics.accuracy_score( y_test, y_pred2)*100
```

Out[51]:

```
76.33587786259542
```

In [52]:

```
logsk2 = LogisticRegression(C=100,penalty='elasticnet',solver='saga',l1_ratio=0)
logsk2.fit(X_train, y_train)
```

Out[52]:

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

In [53]:

```
y_pred3 = logsk.predict(X_test)
```

In [54]:

```
metrics.accuracy_score( y_test, y_pred3)*100
```

Out[54]:

```
76.33587786259542
```

In [55]:

```
logsk3 = LogisticRegression(C=10,penalty='elasticnet',solver='saga',l1_ratio=1)
logsk3.fit(X_train, y_train)
```

Out[55]:

```
LogisticRegression(C=10, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=1, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='elasticnet',
                   random_state=None, solver='saga', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [56]:

```
y_pred4 = logsk.predict(X_test)
```

In [57]:

```
metrics.accuracy_score( y_test, y_pred4)*100
```

Out[57]:

```
76.33587786259542
```

In [59]:

```
# Converting y_pred to a dataframe which is an array
y_pred_df = pd.DataFrame(y_pred1)
```

In [60]:

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

In [61]:

```
# Let's see the head
y_pred_1.head()
```

Out[61]:

	1
0	0.607130
1	0.075576
2	0.088081
3	0.605983
4	0.963953

In [62]:

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

In [63]:

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

In [64]:

```
y_test_df
```

Out[64]:

	survived	name
173	0.0	173
843	0.0	843
996	0.0	996
992	0.0	992
12	1.0	12
...	...	...
1191	0.0	1191
165	1.0	165
588	1.0	588
270	1.0	270
61	1.0	61

393 rows × 2 columns

In [65]:

```
# 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 [66]:

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

In [67]:

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

In [68]:

```
# Let's see the head of y_pred_final
y_pred_final.head()
```

Out[68]:

	survived	name	Survived_Prob
0	0.0	173	0.607130
1	0.0	843	0.075576
2	0.0	996	0.088081
3	0.0	992	0.605983
4	1.0	12	0.963953

In [69]:

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

In [70]:

```
# Let's see the head
y_pred_final.head()
```

Out[70]:

	survived	name	Survived_Prob	predicted
0	0.0	173	0.607130	1
1	0.0	843	0.075576	0
2	0.0	996	0.088081	0
3	0.0	992	0.605983	1
4	1.0	12	0.963953	1

In [71]:

```
from sklearn import metrics
```

In [74]:

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

Out[74]:

```
array([[208,  45],
       [ 46,  94]], dtype=int64)
```

In [75]:

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

Out[75]:

```
0.7684478371501272
```

In [76]:

```
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 [77]:

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

Out[77]:

```
0.8188976377952756
```

In [78]:

```
# Let us calculate specificity
TN / float(TN+FP)
```

Out[78]:

```
0.6762589928057554
```

In [80]:

```
# Calculate false postive rate - predicting survived when customer does not survived
print(FP/ float(TN+FP))
```

```
0.3237410071942446
```

In [81]:

```
# positive predictive value
print (TP / float(TP+FP))
```

```
0.8221343873517787
```

In [82]:

```
# Negative predictive value
print (TN / float(TN+ FN))
```

```
0.6714285714285714
```

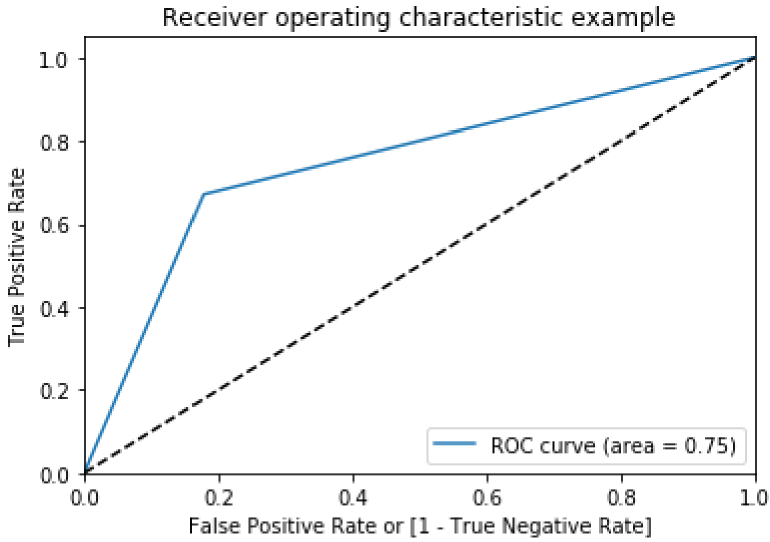
In [85]:

```
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 [87]:

```
# Importing matplotlib and seaborn
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
draw_roc(y_pred_final.survived, y_pred_final.predicted)
```



Out[87]:

```
(array([0.        , 0.17786561, 1.        ]),
 array([0.        , 0.67142857, 1.        ]),
 array([2, 1, 0], dtype=int64))
```

In [91]:

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

Out[91]:

	survived	name	Survived_Prob	predicted	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0	0.0	173	0.607130	1	1	1	1	1	1	1	0	0	0	
1	0.0	843	0.075576	0	1	0	0	0	0	0	0	0	0	0
2	0.0	996	0.088081	0	1	0	0	0	0	0	0	0	0	0
3	0.0	992	0.605983	1	1	1	1	1	1	1	1	0	0	0
4	1.0	12	0.963953	1	1	1	1	1	1	1	1	1	1	1

In [92]:

```
# Now let's calculate accuracy sensitivity and specificity for various probability cut
cutoff_df = pd.DataFrame( columns = ['prob','accuracy','sensi','speci'])
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 = metrics.confusion_matrix( y_pred_final.survived, y_pred_final[i] )
    total1=sum(sum(cm1))
    accuracy = (cm1[0,0]+cm1[1,1])/total1
    sensi = cm1[0,0]/(cm1[0,0]+cm1[0,1])
    speci = cm1[1,1]/(cm1[1,0]+cm1[1,1])
    cutoff_df.loc[i] =[ i ,accuracy,sensi,speci]
print(cutoff_df)
```

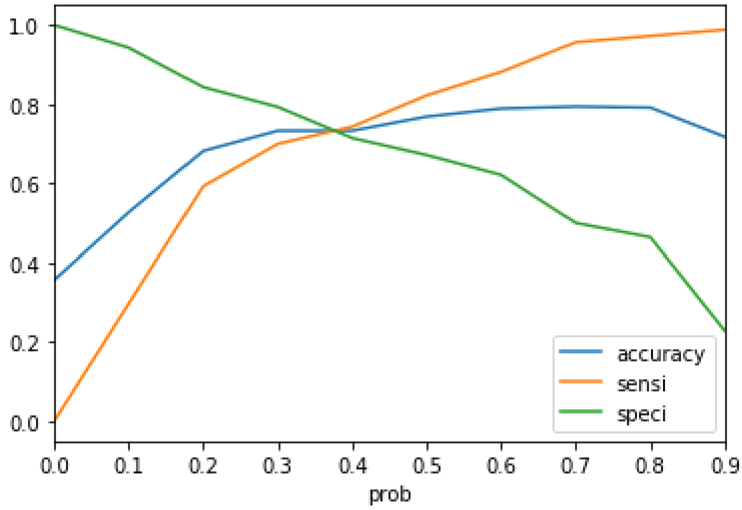
```
     prob  accuracy     sensi     speci
0.0   0.0  0.356234  0.000000  1.000000
0.1   0.1  0.526718  0.296443  0.942857
0.2   0.2  0.681934  0.592885  0.842857
0.3   0.3  0.732824  0.699605  0.792857
0.4   0.4  0.732824  0.743083  0.714286
0.5   0.5  0.768448  0.822134  0.671429
0.6   0.6  0.788804  0.881423  0.621429
0.7   0.7  0.793893  0.956522  0.500000
0.8   0.8  0.791349  0.972332  0.464286
0.9   0.9  0.717557  0.988142  0.228571
```

In [93]:

```
# Let's plot accuracy sensitivity and specificity for various probabilities.
cutoff_df.plot.line(x='prob', y=['accuracy','sensi','speci'])
```

Out[93]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x215b1629e88>
```



In [95]:

```
y_pred_final['final_predicted'] = y_pred_final.Survived_Prob.map( lambda x: 1 if x > 0
```

In [96]:

```
y_pred_final.head()
```

Out[96]:

	survived	name	Survived_Prob	predicted	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	fir
0	0.0	173	0.607130	1	1	1	1	1	1	1	1	0	0	0	
1	0.0	843	0.075576	0	1	0	0	0	0	0	0	0	0	0	
2	0.0	996	0.088081	0	1	0	0	0	0	0	0	0	0	0	
3	0.0	992	0.605983	1	1	1	1	1	1	1	1	0	0	0	
4	1.0	12	0.963953	1	1	1	1	1	1	1	1	1	1	1	

In [98]:

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

Out[98]:

```
0.732824427480916
```

In [100]:

```
metrics.confusion_matrix( y_pred_final.survived, y_pred_final.final_predicted )
```

Out[100]:

```
array([[177,  76],
       [ 29, 111]], dtype=int64)
```

In [ ]:

```

```