In [1]:

```
print ("Question 2")
```

```
Question 2
```

In [2]:

```
import pandas as pd
import numpy as np
from sklearn.cross_validation import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
```

```
C:\Users\Chandrima\Anaconda3\lib\site-packages\sklearn\cross_validation.p
y:41: DeprecationWarning: This module was deprecated in version 0.18 in fa
vor of the model_selection module into which all the refactored classes an
d functions are moved. Also note that the interface of the new CV iterator
s are different from that of this module. This module will be removed in
0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
```

In [3]:

```
# a: Import the train/test files from Titanic
train = pd.read_csv('F:/Annie/CornellMS/Semester 4/Machine Learning/HW1/Titanic/train.c
sv', delimiter=',')
test = pd.read_csv('F:/Annie/CornellMS/Semester 4/Machine Learning/HW1/Titanic/test.cs
v', delimiter=',')
```

In [4]:

```
# Check training features and values present
train.notna().sum()
```

Out[4]:

```
PassengerId    891
Survived       891
Pclass         891
Name           891
Sex            891
Age            714
SibSp          891
Parch          891
Ticket         891
Fare           891
Cabin          204
Embarked       889
dtype: int64
```

In [5]:

```
def clean_data(data):
    data = data.drop(columns=['Name'], axis=0) #Name has unique value for each passenge
r and will not affect the model
    data['Sex'] = data['Sex'].replace(['male', 'female'], [0,1])
    data['Age'] = data['Age'].fillna(round(data.Age.mean()))
    data['Embarked'] = data['Embarked'].replace(['S', 'C', 'Q', np.nan], [0, 1, 2, 3])
    tickets = data.Ticket.unique()
    tickets_dic = dict(zip(tickets, range(len(tickets))))
    cabin = data.Cabin.unique()
    cabin_dic = dict(zip(cabin, range(len(cabin))))
    data = data.replace({'Ticket': tickets_dic, 'Cabin': cabin_dic})
    return data
```

In [6]:

```
train_data = clean_data(train)
train_data
```

Out[6]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin
0	1	0	3	0	22.0	1	0	0	7.2500	0
1	2	1	1	1	38.0	1	0	1	71.2833	1
2	3	1	3	1	26.0	0	0	2	7.9250	0
3	4	1	1	1	35.0	1	0	3	53.1000	2
4	5	0	3	0	35.0	0	0	4	8.0500	0
5	6	0	3	0	30.0	0	0	5	8.4583	0
6	7	0	1	0	54.0	0	0	6	51.8625	3
7	8	0	3	0	2.0	3	1	7	21.0750	0
8	9	1	3	1	27.0	0	2	8	11.1333	0
9	10	1	2	1	14.0	1	0	9	30.0708	0
10	11	1	3	1	4.0	1	1	10	16.7000	4
11	12	1	1	1	58.0	0	0	11	26.5500	5
12	13	0	3	0	20.0	0	0	12	8.0500	0
13	14	0	3	0	39.0	1	5	13	31.2750	0
14	15	0	3	1	14.0	0	0	14	7.8542	0
15	16	1	2	1	55.0	0	0	15	16.0000	0
16	17	0	3	0	2.0	4	1	16	29.1250	0
17	18	1	2	0	30.0	0	0	17	13.0000	0
18	19	0	3	1	31.0	1	0	18	18.0000	0
19	20	1	3	1	30.0	0	0	19	7.2250	0
20	21	0	2	0	35.0	0	0	20	26.0000	0
21	22	1	2	0	34.0	0	0	21	13.0000	6
22	23	1	3	1	15.0	0	0	22	8.0292	0
23	24	1	1	0	28.0	0	0	23	35.5000	7
24	25	0	3	1	8.0	3	1	7	21.0750	0
25	26	1	3	1	38.0	1	5	24	31.3875	0
26	27	0	3	0	30.0	0	0	25	7.2250	0
27	28	0	1	0	19.0	3	2	26	263.0000	8
28	29	1	3	1	30.0	0	0	27	7.8792	0
29	30	0	3	0	30.0	0	0	28	7.8958	0
...	...	...	...	...	...	...	...	...	...	...
861	862	0	2	0	21.0	1	0	660	11.5000	0
862	863	1	1	1	48.0	0	0	661	25.9292	136

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin
863	864	0	3	1	30.0	8	2	148	69.5500	0
864	865	0	2	0	24.0	0	0	662	13.0000	0
865	866	1	2	1	42.0	0	0	663	13.0000	0
866	867	1	2	1	27.0	1	0	664	13.8583	0
867	868	0	1	0	31.0	0	0	665	50.4958	144
868	869	0	3	0	30.0	0	0	666	9.5000	0
869	870	1	3	0	4.0	1	1	8	11.1333	0
870	871	0	3	0	26.0	0	0	667	7.8958	0
871	872	1	1	1	47.0	1	1	222	52.5542	42
872	873	0	1	0	33.0	0	0	668	5.0000	115
873	874	0	3	0	47.0	0	0	669	9.0000	0
874	875	1	2	1	28.0	1	0	274	24.0000	0
875	876	1	3	1	15.0	0	0	670	7.2250	0
876	877	0	3	0	20.0	0	0	128	9.8458	0
877	878	0	3	0	19.0	0	0	671	7.8958	0
878	879	0	3	0	30.0	0	0	672	7.8958	0
879	880	1	1	1	56.0	0	1	276	83.1583	145
880	881	1	2	1	25.0	0	1	232	26.0000	0
881	882	0	3	0	33.0	0	0	673	7.8958	0
882	883	0	3	1	22.0	0	0	674	10.5167	0
883	884	0	2	0	28.0	0	0	675	10.5000	0
884	885	0	3	0	25.0	0	0	676	7.0500	0
885	886	0	3	1	39.0	0	5	16	29.1250	0
886	887	0	2	0	27.0	0	0	677	13.0000	0
887	888	1	1	1	19.0	0	0	678	30.0000	146
888	889	0	3	1	30.0	1	2	614	23.4500	0
889	890	1	1	0	26.0	0	0	679	30.0000	147
890	891	0	3	0	32.0	0	0	680	7.7500	0

891 rows × 11 columns

In [7]:

```
X = train_data.drop(columns=['Survived'], axis=0)
y = train_data['Survived']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state
= 0)
```

In [8]:

```
sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
```

In [9]:

```
lr = LogisticRegression()
lr.fit(X_train_std, y_train)
lr.coef_
```

Out[9]:

```
array([[-1.62829646e-04, -7.46489676e-01,  1.26253889e+00,
        -5.79799973e-01, -4.41016207e-01, -5.20773726e-02,
        -8.49053018e-02,  4.30454845e-02,  2.49439696e-01,
         1.64205111e-01]])
```

In [10]:

```
predictions = lr.predict(X_test)
print(confusion_matrix(y_test, predictions))
print(classification_report(y_test, predictions))
```

```
[[110   0]
 [ 67   2]]
             precision    recall  f1-score   support

          0       0.62      1.00      0.77       110
          1       1.00      0.03      0.06        69

avg / total       0.77      0.63      0.49       179
```

In [19]:

```
print ("Answer", "\n" ,"I used the coef_ function defined in scipy which outputs the co
efficient for features in the decision matrix. Then I choose those features which has a
feature weight of more than absolute(0.5). I had initially dropped names column as all
 values are unique and wouldn't have made any difference. There is an increase in recal
l even though precision remains the same.")
```

```
Answer
 I used the coef_ function defined in scipy which outputs the coefficient
for features in the decision matrix. Then I choose those features which ha
s a feature weight of more than absolute(0.5). I had initially dropped nam
es column as all values are unique and wouldn't have made any difference.
There is an increase in recall even though precision remains the same.
```

In [12]:

```
# The values influencing the positive and the negative class seems to be Pclass, Sex, S
ibSp and Parch.
# I chose is based on |lr.coef_[feature]| > .3
X = train_data.drop(columns=['SibSp', 'Parch', 'Fare', 'Embarked', 'PassengerId', 'Cabi
n'], axis=0)
y = train_data['Survived']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state
= 0)
X_train_std = sc.fit_transform(X_train)
lr.fit(X_train_std, y_train)
predictions = lr.predict(X_test)
print(classification_report(y_test, predictions))
print(confusion_matrix(y_test, predictions))
# We see the Precision for the following have increased after dropping the following co
lumns
```

```
             precision    recall  f1-score   support

          0       0.86      0.68      0.76       110
          1       0.62      0.83      0.71        69

avg / total       0.77      0.74      0.74       179

[[75 35]
 [12 57]]
```

In [15]:

```
# c: Train on entire training set and predict test set
X_train = train_data.drop(columns=['Survived', 'SibSp', 'Parch', 'Fare', 'Embarked', 'P
assengerId', 'Cabin'], axis=0)
X_test = clean_data(test).drop(columns=['SibSp', 'Parch', 'Fare', 'Embarked', 'Passenge
rId', 'Cabin'], axis=0)
y_train = train_data['Survived']
```

In [16]:

```
sc.fit(X_train)
X_train_std = sc.fit_transform(X_train)
lr.fit(X_train_std, y_train)
predictions = lr.predict(X_test)
```

In [17]:

```
final_out = pd.DataFrame(predictions, columns=['Survived'])
final_out['PassengerId'] =test['PassengerId']
final_out = final_out.set_index('PassengerId')
final_out.to_csv('F:/Annie/CornellMS/Semester 4/Machine Learning/HW1/Titanic/Titanic_Su
bmission.csv')
```