Logistic Regesstion and support Vector machine

In [1]:
```
#1 Import the libraries
import numpy as np;
import pandas as pd
import matplotlib as plt
```

In [2]:
```
#2 Read the training and test data into respective variables

train =pd.read_csv("train.csv")
test= pd.read_csv("test.csv")
```

In [3]:
```
#3 Take a peak at the data
train.head()
```

Out[3]:

	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 [4]:
```
#4 Choose some of the variables

df=train[['Survived','Pclass','Sex','Age','Fare']]
```

In [5]:
```
# 5 Encoding Gender values to 0 and 1

df['Sex']=df['Sex'].apply(lambda sex:1 if sex=="male" else 0)
```

```
C:\Users\intel\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  This is separate from the ipykernel package so we can avoid doing imports until
```

In [6]:
```
#6 Handling Missing Values - Data Imputation

df["Age"]=df["Age"].fillna(df["Age"].median()) #medain is robust to outliner
```

```
C:\Users\intel\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  This is separate from the ipykernel package so we can avoid doing imports until
```

In [7]:
```
#7 Set the Predicator and Response Variables
X=df.drop("Survived",axis=1)
Y=df["Survived"]
```

In [8]:
```
#8 Split the data into training and test sets
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.4,random_state=25)
```

In [9]:
```
#9 The Logistic Regression Model
from sklearn.linear_model import LogisticRegression
logit=LogisticRegression()
logit.fit(X_train,Y_train)
```

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

Out[9]:
```
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 [10]:
```
#10 Y Predictions
Y_pred =logit.predict(X_test)
```

In [11]:
```
#11 Confusion Matrix
from sklearn.metrics import confusion_matrix
confusion_matrix=confusion_matrix(Y_test,Y_pred)
confusion_matrix
```

Out[11]:
```
array([[182,  39],
       [ 45,  91]], dtype=int64)
```

In [12]:
```
#12 Accuracy Score
from sklearn.metrics import accuracy_score
accuracy_score(Y_test,Y_pred)
```

Out[12]:
```
0.7647058823529411
```

In [13]:
```
#13 Classification Report
from sklearn.metrics import classification_report
report =classification_report(Y_test,Y_pred)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.80      0.82      0.81       221
           1       0.70      0.67      0.68       136

    accuracy                           0.76       357
   macro avg       0.75      0.75      0.75       357
weighted avg       0.76      0.76      0.76       357
```

Using Support Vector machine

In [14]:
```
#using Support Vector machine
from sklearn import svm
SVM = svm.LinearSVC()
SVM.fit(X_train,Y_train)
```

```
C:\Users\intel\Anaconda3\lib\site-packages\sklearn\svm\base.py:929: ConvergenceWarning: Liblinear failed to co
nverge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
```

Out[14]:
```
LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
```

In [15]:
```
#10 Y Predictions
Y_pred_svm =SVM.predict(X_test)
```

In [16]:
```
#12 Accuracy Score
from sklearn.metrics import accuracy_score
accuracy_score(Y_test,Y_pred_svm)
```

Out[16]:
```
0.6582633053221288
```

In [ ]:
```

```