Prediction using all Classification Models

In [20]:

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score,cohen_kappa_score

def classify_report(ytest,y_pred):
    cm = confusion_matrix(ytest, y_pred)
    print ("Confusion Matrix : \n", cm)
    error_rate = 1 - accuracy_score(ytest, y_pred)
    print ("Classification Report:\n")
    print(classification_report(ytest,y_pred))
    print("Accuracy: \n", accuracy_score(ytest, y_pred))
    print("Error Rate: \n",error_rate )
    print("Kappa Score: \n", cohen_kappa_score(ytest, y_pred))
```

In [49]:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score,cohen_kappa_score

#csv file
url = 'C:/Users/Prerna/Desktop/ML_jupyter_notenooks/datasets/titanic.csv'

#Creating a dataframe
dataframe = pd.read_csv(url).fillna(0)

#DATA CLEANING
#Dropping columns
dataframe = dataframe.drop('Name',axis=1)
dataframe = dataframe.drop('SexCode',axis=1)

# Create mapper
pclass_mapper = {"1st":1,"2nd":2,"3rd":3}
gender_mapper = {"male":1 ,"female":2}
survived_mapper = {0:1,1:2}

# Replace feature values with scale
dataframe["PClass"] = dataframe["PClass"].replace(pclass_mapper)
dataframe["Sex"] = dataframe["Sex"].replace(gender_mapper)
dataframe["Survived"] = dataframe["Survived"].replace(survived_mapper)

#Replacing missing values of Age with mean of age
dataframe["Age"] = np.where(dataframe['Age']==0,np.mean(dataframe['Age']),dataframe['Age'])


# Input features
x = dataframe.iloc[:, :3].values
# Output class
y = dataframe.iloc[:, 3].values


'''
xtrain = training features
xtest = testing features
ytrain = classes of training data
```

```python
ytest = classes of testing data
'''
#Splitting the data into training and test
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size = 0.3, random_state = 0)

#Create a model
gnb = GaussianNB()

#Fit a model
gnb.fit(xtrain, ytrain)

#Perform Predictions
y_pred_nb = gnb.predict(xtest)
classify_report(ytest,y_pred_nb)

```

```
Confusion Matrix :
 [[188  76]
 [ 33  97]]
Classification Report:

              precision    recall  f1-score   support

           1       0.85      0.71      0.78       264
           2       0.56      0.75      0.64       130

    accuracy                           0.72       394
   macro avg       0.71      0.73      0.71       394
weighted avg       0.75      0.72      0.73       394

Accuracy:
 0.7233502538071066
Error Rate:
 0.2766497461928934
Kappa Score:
 0.422784333754469
```

In [36]:

```
from sklearn.neighbors import KNeighborsClassifier

# decision tree model creation
knn = KNeighborsClassifier(n_neighbors=10)
knn.fit(xtrain,ytrain)
# predictions
y_pred_knn = knn.predict(xtest)

classify_report(ytest,y_pred_knn)
```

```
Confusion Matrix :
 [[242  22]
 [ 67  63]]
Classification Report:

              precision    recall  f1-score   support

           1       0.78      0.92      0.84       264
           2       0.74      0.48      0.59       130

    accuracy                           0.77       394
   macro avg       0.76      0.70      0.72       394
weighted avg       0.77      0.77      0.76       394

Accuracy:
 0.7741116751269036
Error Rate:
 0.2258883248730964
Kappa Score:
 0.4399297236863121
```

In [37]:

```python
from sklearn.ensemble import RandomForestClassifier
# random forest model creation
rfc = RandomForestClassifier(n_estimators=10)
rfc.fit(xtrain,ytrain)
# predictions
y_pred_rfc = rfc.predict(xtest)

classify_report(ytest,y_pred_rfc)

```

```
Confusion Matrix :
 [[238  26]
 [ 56  74]]
Classification Report:

              precision    recall  f1-score   support

           1       0.81      0.90      0.85       264
           2       0.74      0.57      0.64       130

    accuracy                           0.79       394
   macro avg       0.77      0.74      0.75       394
weighted avg       0.79      0.79      0.78       394

Accuracy:
 0.7918781725888325
Error Rate:
 0.20812182741116747
Kappa Score:
 0.500030950170226
```

In [38]:

```
from sklearn import tree

# decision tree model creation
dtc = tree.DecisionTreeClassifier()
dtc.fit(xtrain,ytrain)
# predictions
y_pred_dtc = dtc.predict(xtest)

classify_report(ytest,y_pred_dtc)
```

```
Confusion Matrix :
 [[244  20]
 [ 57  73]]
Classification Report:

              precision    recall  f1-score   support

           1       0.81      0.92      0.86       264
           2       0.78      0.56      0.65       130

    accuracy                           0.80       394
   macro avg       0.80      0.74      0.76       394
weighted avg       0.80      0.80      0.79       394

Accuracy:
 0.8045685279187818
Error Rate:
 0.19543147208121825
Kappa Score:
 0.5236016456769574
```

In [39]:

```python
from sklearn.svm import SVC

svclassifier = SVC(kernel='linear')
svclassifier.fit(xtrain, ytrain)
y_pred_svm = svclassifier.predict(xtest)

classify_report(ytest,y_pred_svm)
```

```
Confusion Matrix :
 [[215  49]
 [ 39  91]]
Classification Report:

              precision    recall  f1-score   support

           1       0.85      0.81      0.83       264
           2       0.65      0.70      0.67       130

    accuracy                           0.78       394
   macro avg       0.75      0.76      0.75       394
weighted avg       0.78      0.78      0.78       394

Accuracy:
 0.7766497461928934
Error Rate:
 0.2233502538071066
Kappa Score:
 0.5045441554729924
```

In [45]:

```python
from sklearn.ensemble import AdaBoostClassifier


seed = 5
num_trees = 200
adaboost = AdaBoostClassifier(n_estimators=num_trees, random_state=seed)
adaboost.fit(xtrain, ytrain)
y_pred_adb = adaboost.predict(xtest)

classify_report(ytest,y_pred_adb)
```

```
Confusion Matrix :
 [[238  26]
 [ 49  81]]
Classification Report:

              precision    recall  f1-score   support

           1       0.83      0.90      0.86       264
           2       0.76      0.62      0.68       130

    accuracy                           0.81       394
   macro avg       0.79      0.76      0.77       394
weighted avg       0.81      0.81      0.80       394

Accuracy:
 0.8096446700507615
Error Rate:
 0.19035532994923854
Kappa Score:
 0.5492540956099943
```

ROC Curve

In [52]:

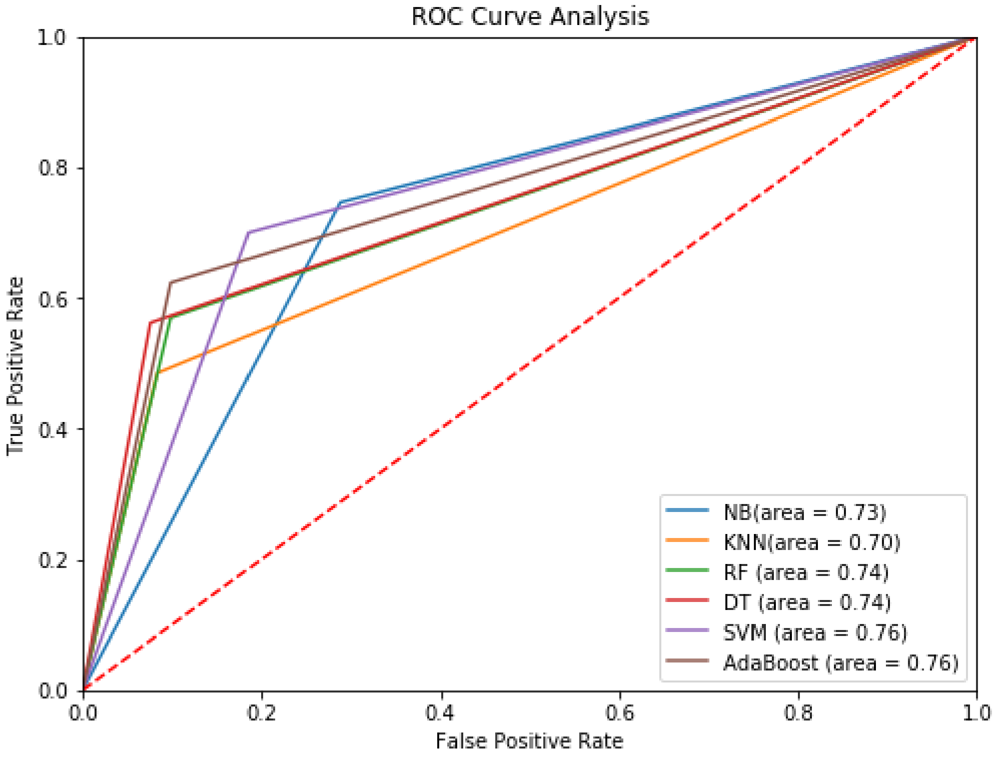
```
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve,roc_auc_score

fpr,tpr,thresholds = roc_curve(ytest,y_pred_nb,pos_label= 2,drop_intermediate = False)
fpr1,tpr1,thresholds1  = roc_curve(ytest,y_pred_knn,pos_label= 2,drop_intermediate = False)
fpr2,tpr2,thresholds2  = roc_curve(ytest,y_pred_rfc,pos_label= 2,drop_intermediate = False)
fpr3,tpr3,thresholds3  = roc_curve(ytest,y_pred_dtc ,pos_label= 2,drop_intermediate = False)
fpr4,tpr4,thresholds4  = roc_curve(ytest,y_pred_svm,pos_label= 2,drop_intermediate = False)
fpr5,tpr5,thresholds5  = roc_curve(ytest,y_pred_adb,pos_label= 2,drop_intermediate = False)


auc_score_nb = roc_auc_score(ytest,y_pred_nb)
auc_score_knn = roc_auc_score(ytest,y_pred_knn)
auc_score_rfc = roc_auc_score(ytest,y_pred_rfc)
auc_score_dtc= roc_auc_score(ytest,y_pred_dtc)
auc_score_svm= roc_auc_score(ytest,y_pred_svm)
auc_score_adb = roc_auc_score(ytest,y_pred_adb)


fig = plt.figure(figsize=(8,6))
plt.plot(fpr,tpr,label='NB(area = %0.2f)' %auc_score_nb)
plt.plot(fpr1,tpr1,label='KNN(area = %0.2f)' %auc_score_knn)
plt.plot(fpr2,tpr2,label='RF (area = %0.2f)' %auc_score_rfc)
plt.plot(fpr3,tpr3,label='DT (area = %0.2f)' %auc_score_dtc)
plt.plot(fpr4,tpr4,label='SVM (area = %0.2f)' %auc_score_svm)
plt.plot(fpr5,tpr5,label='AdaBoost (area = %0.2f)' %auc_score_adb)

plt.legend(loc = 'best')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.title('ROC Curve Analysis')
plt.show()
```


ROC Curve Analysis
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
NB(area = 0.73)
KNN(area = 0.70)
RF (area = 0.74)
DT (area = 0.74)
SVM (area = 0.76)
AdaBoost (area = 0.76)

In [ ]: