MACHINE LEARNING

ASSIGNMENT 4

PRESENTED BY: JOSE GIL

ID: 218 659 676

Part # 2

In [5]:
```
# Importing libraries
import pandas as pd
import warnings
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
from statistics import mean
from sklearn.metrics import accuracy_score, f1_score,confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_score,train_test_split
from sklearn.linear_model import SGDClassifier
from mpl_toolkits.mplot3d import axes3d
from sklearn import svm
from sklearn.ensemble import RandomForestClassifier
```

In [6]:
```
# importing training and testing datasets
import pandas as pd
train_dt1 = pd.read_excel('X_train.xlsx', header=None)
test_dt1 = pd.read_excel('X_test.xlsx',header=None)
```

```
train_dt2 = pd.read_excel('y_train.xlsx',header=None)
test_dt2 = pd.read_excel('y_test.xlsx',header=None)
```

In [7]:
```
#Printing shapes
print(train_dt1.shape,test_dt1.shape,train_dt2.shape,test_dt2.shape)
```

```
(7352, 561) (2947, 561) (7352, 1) (2947, 1)
```

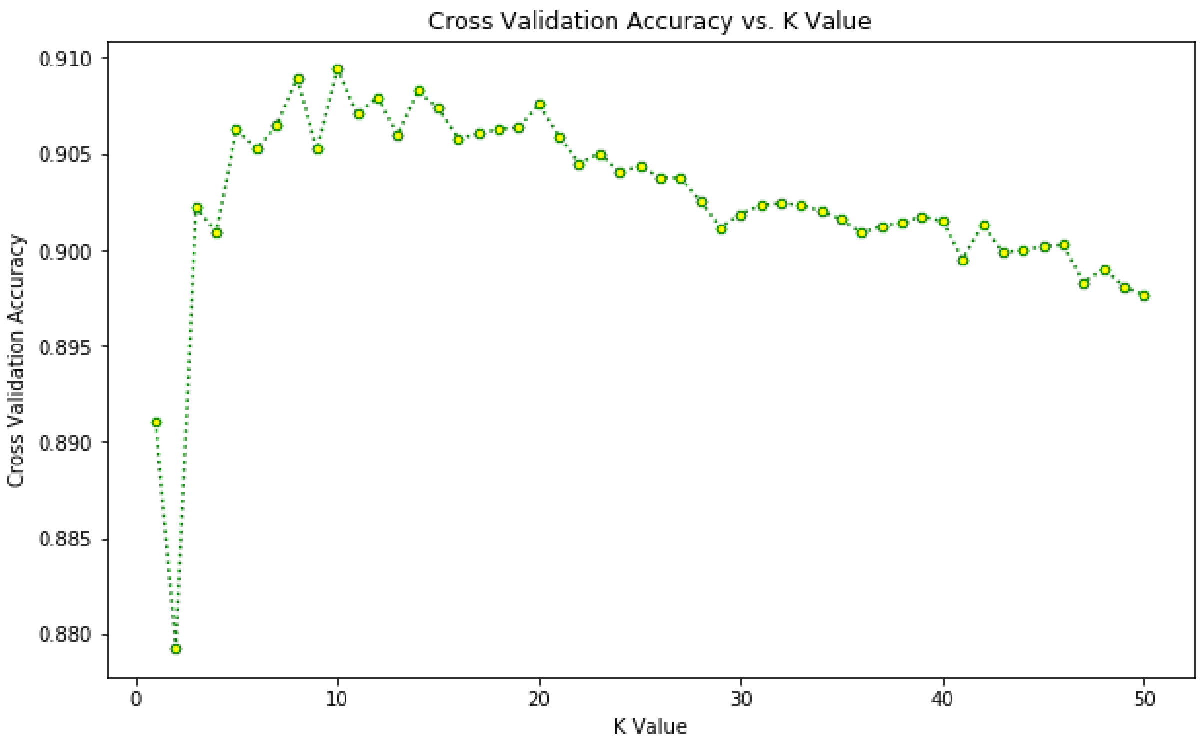
In [8]:
```
# fID for perfomance
SID=218659676
fID=SID%2
print(fID)
```

```
0
```

In [9]:
```
#Plotting
import warnings
value = []
for i in range(1, 51):
    warnings.simplefilter("ignore")
    knn = KNeighborsClassifier(n_neighbors = i)
    knn.fit(train_dt1,train_dt2)
    cv_scores = cross_val_score(knn, train_dt1, train_dt2, cv = 10,scor
ing='f1_weighted')
    avg_cross_val_score = mean(cv_scores)
    value.append(avg_cross_val_score)

plt.figure(figsize =(10, 6))
plt.plot(range(1, 51), value, color ='green', marker ='H', linestyle =
'dotted', markerfacecolor ='yellow', markersize = 5)
plt.title('Cross Validation Accuracy vs. K Value')
plt.ylabel('Cross Validation Accuracy')
plt.xlabel('K Value')
```

Out[9]:
```
Text(0.5, 0, 'K Value')
```



In [10]:
```
value = []
for i in range(1, 51):
    warnings.simplefilter("ignore")
    knn = KNeighborsClassifier(n_neighbors = i)
    knn.fit(train_dt1,train_dt2)
    cv_scores = cross_val_score(knn, train_dt1, train_dt2, cv = 10,scor
ing='f1_macro')
    avg_cross_val_score = mean(cv_scores)
    value.append(avg_cross_val_score)
```

In [11]:
```
with open('knn.txt', 'a') as f:
    for j in range(1,51):
        print(f"---------------------------------",file=f)
        print(f"{j}:{value[j-1]}",file=f)
```

In [12]:

```python
# Re-training the model
knn = KNeighborsClassifier(n_neighbors = 10)
warnings.simplefilter("ignore")
knn.fit(train_dt1, train_dt2)
pred = knn.predict(test_dt1)
con_mat = confusion_matrix(test_dt2, pred)
print ('Confusion Matrix :')
print(con_mat)
print ('Accuracy Score :',accuracy_score(test_dt2, pred)*100)
print('F1-Score :',f1_score(test_dt2, pred,average="macro")*100)
```

```
Confusion Matrix :
[[486   0  10   0   0   0]
 [ 36 431   4   0   0   0]
 [ 51  41 328   0   0   0]
 [  0   4   0 409  78   0]
 [  0   0   0  47 485   0]
 [  0   0   0   2   2 533]]
Accuracy Score : 90.66847641669494
F1-Score : 90.38079349608216
```

Part 3

In [13]:

```python
train_dt1.head()
```

Out[13]:

	0	1	2	3	4	5	6	7	8
0	0.288585	-0.020294	-0.132905	-0.995279	-0.983111	-0.913526	-0.995112	-0.983185	-0.923527
1	0.278419	-0.016411	-0.123520	-0.998245	-0.975300	-0.960322	-0.998807	-0.974914	-0.957686
2	0.279653	-0.019467	-0.113462	-0.995380	-0.967187	-0.978944	-0.996520	-0.963668	-0.977469
3	0.279174	-0.026201	-0.123283	-0.996091	-0.983403	-0.990675	-0.997099	-0.982750	-0.989302
4	0.276629	-0.016570	-0.115362	-0.998139	-0.980817	-0.990482	-0.998321	-0.979672	-0.990441

5 rows × 561 columns

In [14]:
```python
# fID for perfomance
SID=218659676
fID=SID%2
print(fID)
```

```
0
```

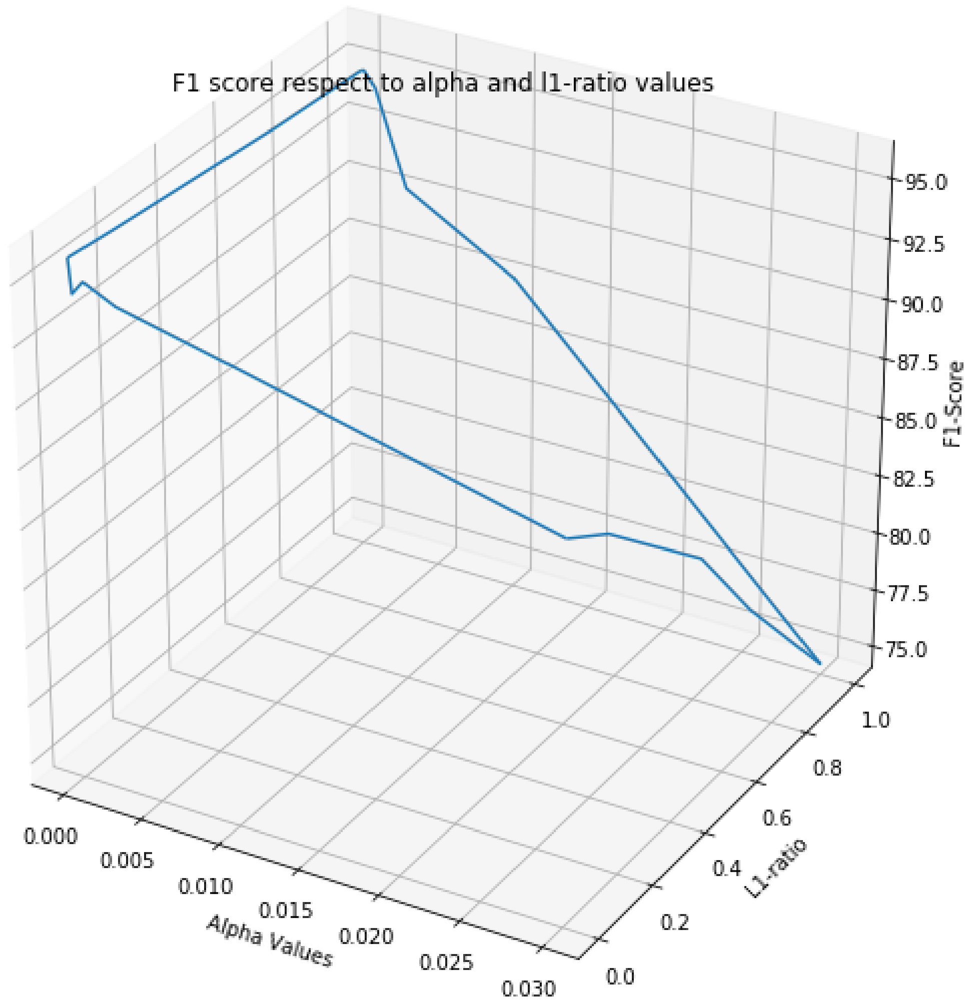
In [15]:
```python
#Parameters
alpha_val=[1e-4,3e-4,1e-3,3e-3,1e-2,3e-2]
l1=[0,.15,.5,.7,1]

f1=[]
with open('elastic.txt', 'a') as f:
    for i in l1:
        for j in alpha_val:
            value=[]
            warnings.simplefilter("ignore")
            elc_net = SGDClassifier(loss='log', penalty='elasticnet',al
pha=j, l1_ratio=i)
            elc_net.fit(train_dt1,train_dt2)
            predictor = elc_net.predict(test_dt1)
            cv_scores = cross_val_score(elc_net, train_dt1, train_dt2,
cv = 10,scoring='accuracy')
            avg_cross_val_score = mean(cv_scores)
            value.append(avg_cross_val_score)
            print(f"{i}  with  {j} is {value[0]}",file=f)
            print("---------------------------------",file=f)
            f1.append(round(f1_score(test_dt2, predictor,average="macr
o")*100,2))
```

As it can be identified in the graph, the best value of alpha is: 1e-4 and l1_ratio is: 0.5

In [16]:
```python
# Drawing Surface Plot
axis_x = np.array(alpha_val)
axis_y = np.array(l1)
```

```
ax_X, ax_Y = np.meshgrid(axis_x,axis_y)
ax_Z = np.array(f1).reshape(5,6)
fig = plt.figure(figsize=(10,10))
ax = fig.add_subplot(111, projection='3d')
ax.set_title('Surface plot')
ax.plot_wireframe(ax_X, ax_Y, ax_Z, rstride=10, cstride=10)
ax.set_ylabel('L1-ratio')
ax.set_xlabel('Alpha Values')
ax.set_zlabel('F1-Score')
ax.set_title('F1 score respect to alpha and l1-ratio values')
plt.show()
```



In [17]:
```
# Re-training the model
elc_net2 = SGDClassifier(loss='log', penalty='elasticnet',alpha=1e-4, l
1_ratio=0.5)
elc_net2.fit(train_dt1,train_dt2)
pred = elc_net2.predict(test_dt1)
```

```
print(pred)
con_mat = confusion_matrix(test_dt2, pred)
print ('Confusion Matrix :')
print(con_mat)
print ('Accuracy Score :',accuracy_score(test_dt2, pred)*100)
print('F1-Score :',f1_score(test_dt2, pred,average="macro")*100)
```

```
[5 5 5 ... 2 2 2]
Confusion Matrix :
[[490   1   5   0   0   0]
 [ 37 430   4   0   0   0]
 [  5   7 405   0   3   0]
 [  0   3   0 407  81   0]
 [  0   0   0   9 523   0]
 [  0   0   0   0   0 537]]
Accuracy Score : 94.74041398031898
F1-Score : 94.72210779381015
```

Part 4

In [18]:
```
# fID for perfomance
SID=218659676
fID=SID%3
print(fID)
```

```
2
```

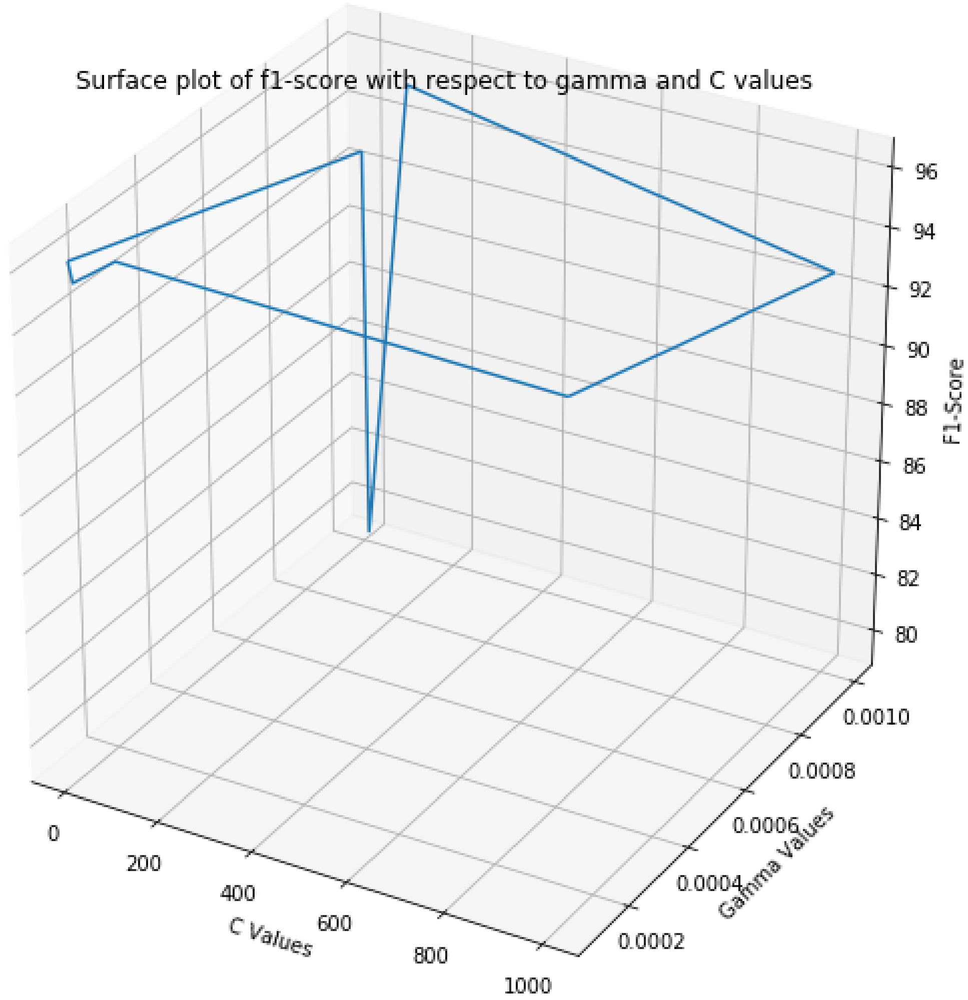
In [19]:
```
#Parameters
gamma=[1e-3,1e-4]
c=[1,10,100,1000]

from sklearn import svm
f1=[]
with open('svm.txt', 'a') as f:
    for i in c:
        for j in gamma:
            value=[]
```

```
            warnings.simplefilter("ignore")
            svm_mdl = svm.SVC(kernel='rbf',C=i,gamma=j)
            svm_mdl.fit(train_dt1,train_dt2)
            predictor = svm_mdl.predict(test_dt1)
            cv_scores = cross_val_score(svm_mdl, train_dt1, train_dt2,
cv = 10,scoring='accuracy')
            avg_cross_val_score = mean(cv_scores)
            value.append(avg_cross_val_score)
            print(f"{i}  with  {j} is {value[0]}",file=f)
            print("---------------------------------",file=f)
            f1.append(round(f1_score(test_dt2, predictor,average="macr
o")*100,2))
```

In [56]:

```
# Plotting the SVM
axis_x = np.array(c)
axis_y = np.array(gamma)
ax_X, ax_Y = np.meshgrid(axis_x,axis_y)
ax_Z = np.array(f1).reshape(2,4)
fig = plt.figure(figsize=(10,10))
ax = fig.add_subplot(111, projection='3d')
ax.set_title('Surface plot')
ax.plot_wireframe(ax_X, ax_Y, ax_Z, rstride=10, cstride=10)
ax.set_xlabel('Cost Parameter')
ax.set_ylabel('Gamma')
ax.set_zlabel('F1-Score')
ax.set_title('F1-score respect to Gamma and Cost Parameter')
plt.show()
```



After plotting the values, even though the graph is in 3D, the best best value identified of Cost Parameter is 1000 Gamma is: 1e-3.

Moreover, as it can be identified, if the cost parameter and gamma values is high, the f-score is also high, thus, there is a correlation and consistency. . It goes up with increase in values

In [20]:
```
# Re-training the model
svm_mdl2 = svm.SVC(kernel='rbf',C=1000,gamma=1e-3)
svm_mdl2.fit(train_dt1,train_dt2)
predictor = svm_mdl.predict(test_dt1)
print(pred)
con_mat = confusion_matrix(test_dt2, pred)
print ('Confusion Matrix :')
print(con_mat)
print ('Accuracy Score :',accuracy_score(test_dt2, pred)*100)
print('F1-Score :',f1_score(test_dt2, pred,average="macro")*100)
```

```
[5 5 5 ... 2 2 2]
Confusion Matrix :
[[490   1   5   0   0   0]
 [ 37 430   4   0   0   0]
 [  5   7 405   0   3   0]
 [  0   3   0 407  81   0]
 [  0   0   0   9 523   0]
 [  0   0   0   0   0 537]]
Accuracy Score : 94.74041398031898
F1-Score : 94.72210779381015
```

Part 5

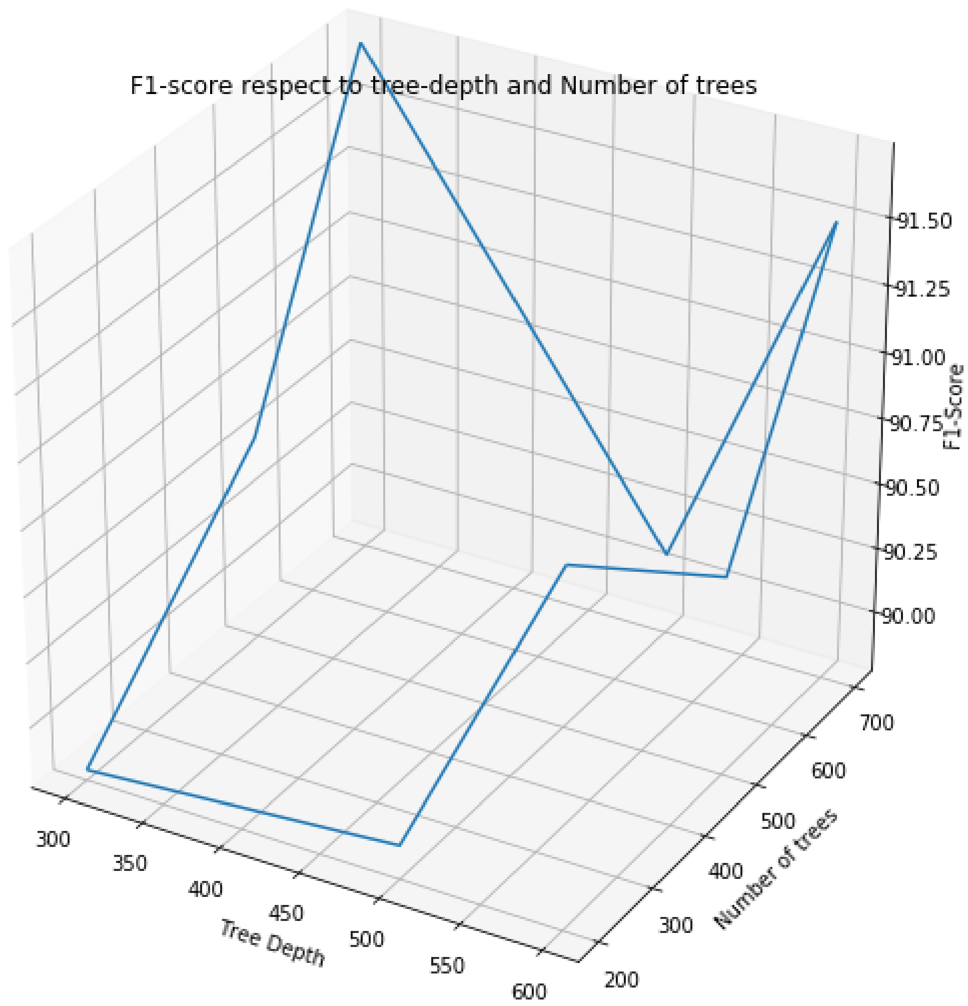
In [21]:
```
# fID for perfomance
SID=218659676
fID=SID%4
print(fID)
```

```
0
```

In [22]:
```
#Parameters
tree_depth=[300,500,600]
num_trees=[200,500,700]

f1=[]
with open('r_forest.txt', 'a') as f:
    for i in tree_depth:
        for j in num_trees:
            value=[]
            warnings.simplefilter("ignore")
            ran_fst = RandomForestClassifier(max_depth=i,max_leaf_nodes
=j)
            ran_fst.fit(train_dt1,train_dt2)
            predictor = ran_fst.predict(test_dt1)
            cv_scores = cross_val_score(ran_fst, train_dt1, train_dt2,
cv = 10,scoring='f1_weighted')
            avg_cross_val_score = mean(cv_scores)
            value.append(avg_cross_val_score)
            print(f"{i}  with  {j} is {value[0]}",file=f)
            print("---------------------------------",file=f)
            f1.append(round(f1_score(test_dt2, predictor,average="macr
o")*100,2))
```

In [23]:
```
axis_x = np.array(tree_depth)
axis_y = np.array(num_trees)
ax_X, ax_Y = np.meshgrid(axis_x,axis_y)
ax_Z = np.array(f1).reshape(3,3)
fig = plt.figure(figsize=(10,10))
ax = fig.add_subplot(111, projection='3d')
ax.set_title('Surface plot')
ax.plot_wireframe(ax_X, ax_Y, ax_Z, rstride=10, cstride=10)
ax.set_xlabel('Tree Depth')
ax.set_ylabel('Number of trees')
ax.set_zlabel('F1-Score')
ax.set_title('F1-score respect to tree-depth and Number of trees')
plt.show()
```



As it can be seen after plotting, the best value of tree-depth is: 300 and and number of trees are: 700

For instance, there are three dependent variables, which affects the consistency. Furthermore, in the graph it can be identified that the highest value of f-score and highest performance is given by the maximum number of trees and low tree-depth.

In [24]:

```
# Re-training the model
ran_fst = RandomForestClassifier(max_depth=300,max_leaf_nodes=700)
ran_fst.fit(train_dt1,train_dt2)
predictor = ran_fst.predict(test_dt1)
print(predictor)
con_mat = confusion_matrix(test_dt2, predictor)
print ('Confusion Matrix :')
print(con_mat)
print ('Accuracy Score :',accuracy_score(test_dt2, predictor)*100)
print('F1-Score :',f1_score(test_dt2, predictor,average="macro")*100)
```

```
[5 5 5 ... 2 2 2]
Confusion Matrix :
[[468  15  13   0   0   0]
 [ 43 418  10   0   0   0]
 [ 23  41 356   0   0   0]
 [  0   0   0 435  56   0]
 [  0   0   0  57 475   0]
 [  0   0   0   0   0 537]]
Accuracy Score : 91.24533423820834
F1-Score : 91.05417838424043
```