Extracting data from npz file

10 components

In [3]:

```
import numpy as np
from sklearn.decomposition import PCA
import warnings
warnings.filterwarnings('ignore')

# Extracting the x_train data and reshapeing it to 2 dimensional array
Train_data =np.load(r"C:\Users\barathy\Downloads\kannada_mnist\Kannada_MNIST_datata
data_xtr =Train_data['arr_0']
data1=data_xtr.reshape(-1,28*28)

# x_train=x_train/255
#Extracting the x_test data and reshapeing it to 2 dimensional array
Test_data=np.load(r"C:\Users\barathy\Downloads\kannada_mnist\Kannada_MNIST_datatase
data_xt=Test_data['arr_0']
data2=data_xt.reshape(-1,28*28)

# x_test=x_test/255

#Extracting the y_train data and reshapeing it to 2 dimensional array
Train_data=np.load(r"C:\Users\barathy\Downloads\kannada_mnist\Kannada_MNIST_datatas
data_ytr=Train_data['arr_0']
data3=data_ytr.reshape(-1,1)

# #Extracting the y_test data and reshapeing it to 2 dimensional array
Test_data=np.load(r"C:\Users\barathy\Downloads\kannada_mnist\Kannada_MNIST_datatase
data_yt=Test_data['arr_0']
data4 =data_yt.reshape(-1,1)
y_test=data4

# Initialize and fit PCA to reduce to 10 components
n_components = 10
pca = PCA(n_components=n_components)

# Fit and transform the training data
x_train_pca = pca.fit_transform(data1)
x_train_p=x_train_pca

# Fit and transform the testing data
x_test_pca=pca.fit_transform(data2)
x_test_p=x_test_pca
```

In [4]:

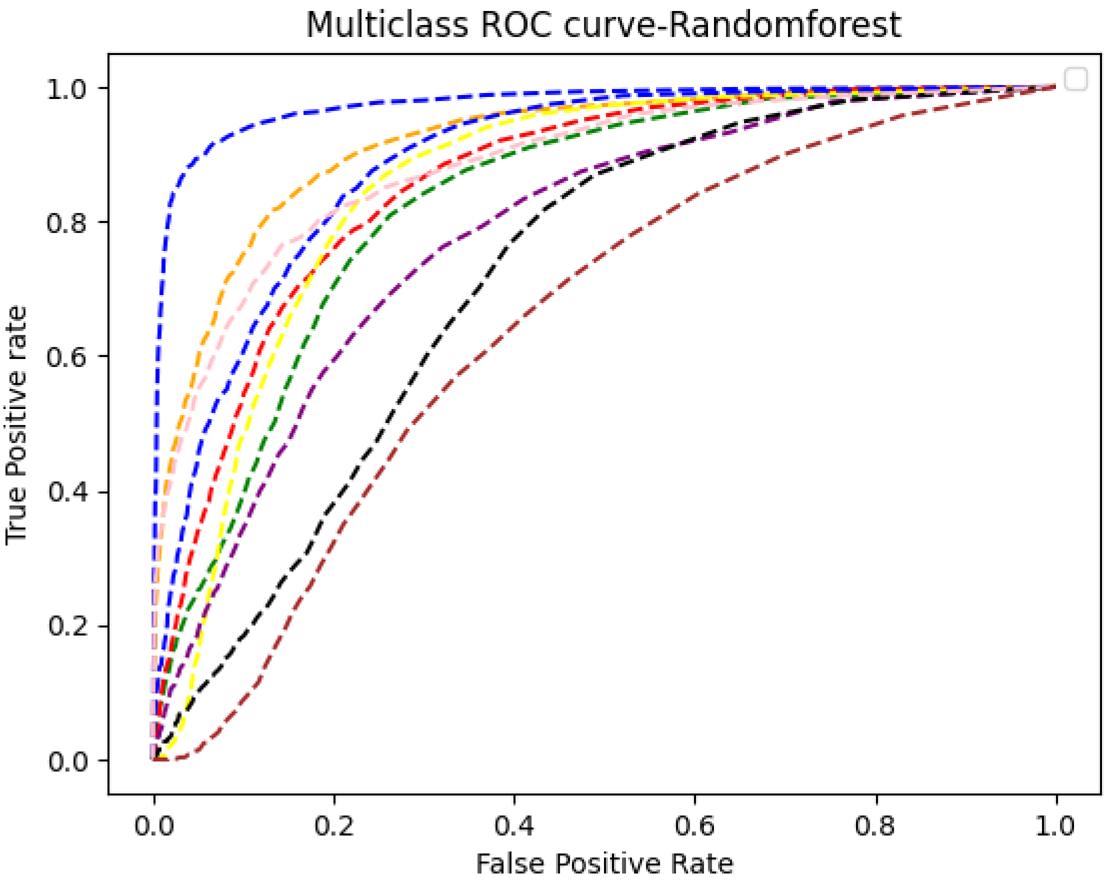
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import precision_score,recall_score,f1_score,confusion_matrix,
import matplotlib.pyplot as plt
import sklearn.metrics as metrics

x_train=x_train_p
x_test=x_test_p
y_train=data3
y_test=data4
model=RandomForestClassifier()
model.fit(x_train,y_train)
train_predict=model.predict(x_train,)
test_predict=model.predict(x_test)
pred_prob=model.predict_proba(x_test)
fpr = {}
tpr = {}
thresh ={}
n_class = 10
#looping the nclass in the target column
for i in range(n_class):
    fpr[i], tpr[i], thresh[i] = roc_curve(y_test, pred_prob[:,i], pos_label=i)
# evaluating the model using auc score
roc_auc=roc_auc_score(y_test,pred_prob,multi_class='ovr',average='macro')

# plotting
plt.plot(fpr[0], tpr[0], linestyle='--',color='orange')
plt.plot(fpr[1], tpr[1], linestyle='--',color='green')
plt.plot(fpr[2], tpr[2], linestyle='--',color='blue')
plt.plot(fpr[3], tpr[3], linestyle='--',color='red')
plt.plot(fpr[4], tpr[4], linestyle='--',color='yellow')
plt.plot(fpr[5], tpr[5], linestyle='--',color='purple')
plt.plot(fpr[6], tpr[6], linestyle='--',color='blue')
plt.plot(fpr[7], tpr[7], linestyle='--',color='black')
plt.plot(fpr[8], tpr[8], linestyle='--',color='pink')
plt.plot(fpr[9], tpr[9], linestyle='--',color='brown')
plt.title('Multiclass ROC curve-Randomforest')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive rate')
plt.legend(loc='best')
plt.show()
print('*********Randomforest************')
print('********Train*******')
print('roc_auc:',roc_auc)
print("Precision: ",precision_score(y_train,train_predict,pos_label='positive',aver
print("Recall: ",recall_score(y_train,train_predict,pos_label='positive',average='m
print("F1 Score: ",f1_score(y_train,train_predict,pos_label='positive',average='mic
print('Confusion :',confusion_matrix(y_train,train_predict))

print('*******Test*******')
print("Precision: ",precision_score(y_test,test_predict,pos_label='positive',averag
print("Recall: ",recall_score(y_test,test_predict,pos_label='positive',average='mic
print("F1 Score: ",f1_score(y_test,test_predict,pos_label='positive',average='micro
print('Confusion :',confusion_matrix(y_test,test_predict))
print('\n \n')
```

```
No artists with labels found to put in legend.  Note that artists whose label start
with an underscore are ignored when legend() is called with no argument.
```


Multiclass ROC curve-Randomforest
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

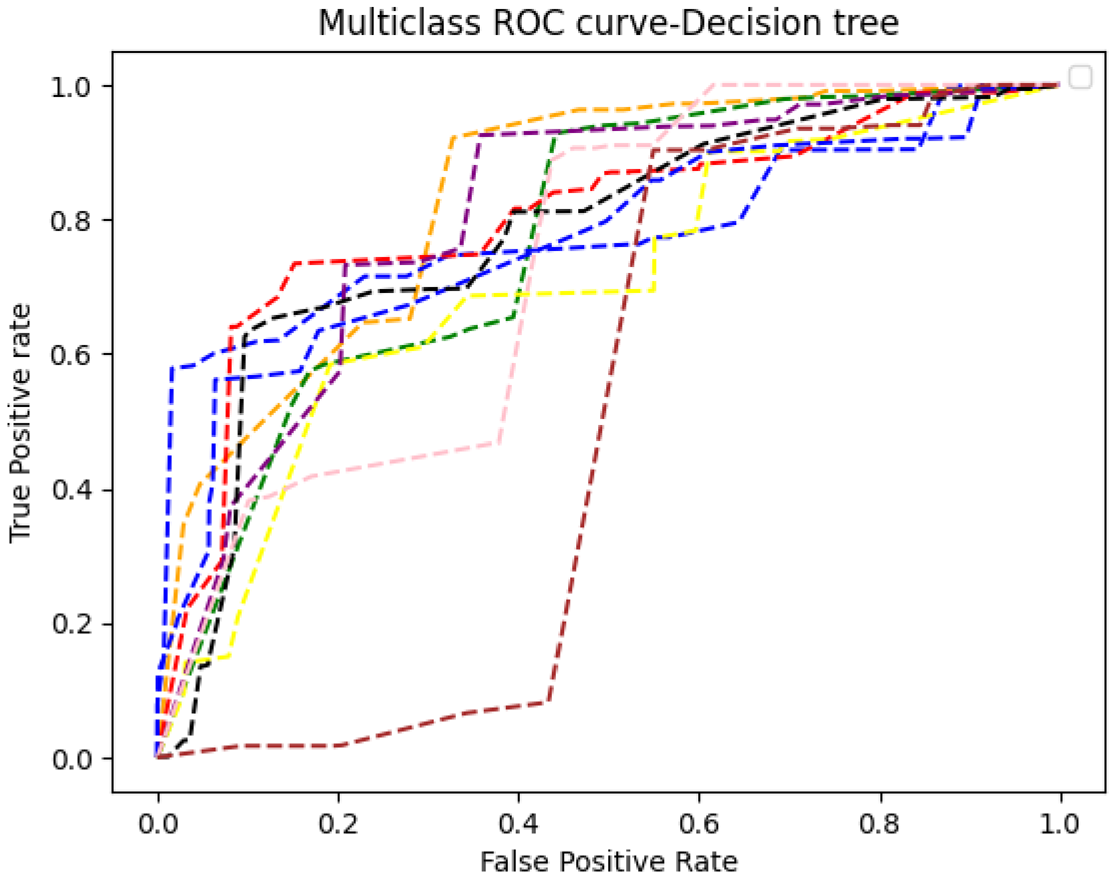
```
*********Randomforest************
********Train*******
Precision:  1.0
Recall:  1.0
F1 Score:  1.0
Confusion : [[6000    0    0    0    0    0    0    0    0    0]
 [   0 6000    0    0    0    0    0    0    0    0]
 [   0    0 6000    0    0    0    0    0    0    0]
 [   0    0    0 6000    0    0    0    0    0    0]
 [   0    0    0    0 6000    0    0    0    0    0]
 [   0    0    0    0    0 6000    0    0    0    0]
 [   0    0    0    0    0    0 6000    0    0    0]
 [   0    0    0    0    0    0    0 6000    0    0]
 [   0    0    0    0    0    0    0    0 6000    0]
 [   0    0    0    0    0    0    0    0    0 6000]]
*******Test*******
Precision:  0.3891
Recall:  0.3891
F1 Score:  0.3891
Confusion : [[615 151   8  35   4  21   7   2  65  92]
 [371 299   6  45  47  44  13   0  24 151]
 [  7   4 798  34  16 129   2   4   1   5]
 [ 15   2  79 545  39   9 118 122  59  12]
 [  1   2   0 203 220  35  95 250  91 103]
 [  1   1  29  19 298 244  19  21 208 160]
 [  3  51  18  52   3   9 414 380  22  48]
 [ 32  29  67 353  21  13 275 138  69   3]
 [ 31   5   0   0  15 303   0   1 540 105]
 [107 495   0  68  21  47   6   9 169  78]]
```

In [5]:

```python
model=DecisionTreeClassifier(max_depth=5)
model.fit(x_train,y_train)
train_predict=model.predict(x_train,)
test_predict=model.predict(x_test)
pred_prob=model.predict_proba(x_test)
fpr = {}
tpr = {}
thresh ={}
n_class = 10
#looping the nclass in the target column
for i in range(n_class):
    fpr[i], tpr[i], thresh[i] = roc_curve(y_test, pred_prob[:,i], pos_label=i)
# evaluating the model using auc score
roc_auc=roc_auc_score(y_test,pred_prob,multi_class='ovr',average='macro')
# plotting
plt.plot(fpr[0], tpr[0], linestyle='--',color='orange')
plt.plot(fpr[1], tpr[1], linestyle='--',color='green')
plt.plot(fpr[2], tpr[2], linestyle='--',color='blue')
plt.plot(fpr[3], tpr[3], linestyle='--',color='red')
plt.plot(fpr[4], tpr[4], linestyle='--',color='yellow')
plt.plot(fpr[5], tpr[5], linestyle='--',color='purple')
plt.plot(fpr[6], tpr[6], linestyle='--',color='blue')
plt.plot(fpr[7], tpr[7], linestyle='--',color='black')
plt.plot(fpr[8], tpr[8], linestyle='--',color='pink')
plt.plot(fpr[9], tpr[9], linestyle='--',color='brown')
plt.title('Multiclass ROC curve-Decision tree')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive rate')
plt.legend(loc='best')
plt.show()
print('******Decision tree********')
print('********Train*******')
print('roc_auc:',roc_auc)
print("Precision: ",precision_score(y_train,train_predict,pos_label='positive',aver
print("Recall: ",recall_score(y_train,train_predict,pos_label='positive',average='m
print("F1 Score: ",f1_score(y_train,train_predict,pos_label='positive',average='mic
print('Confusion :',confusion_matrix(y_train,train_predict))

print('*******Test*******')
print("Precision: ",precision_score(y_test,test_predict,pos_label='positive',averag
print("Recall: ",recall_score(y_test,test_predict,pos_label='positive',average='mic
print("F1 Score: ",f1_score(y_test,test_predict,pos_label='positive',average='micro
print('Confusion :',confusion_matrix(y_test,test_predict))
print('\n \n')
```

```
No artists with labels found to put in legend.  Note that artists whose label start
with an underscore are ignored when legend() is called with no argument.
```



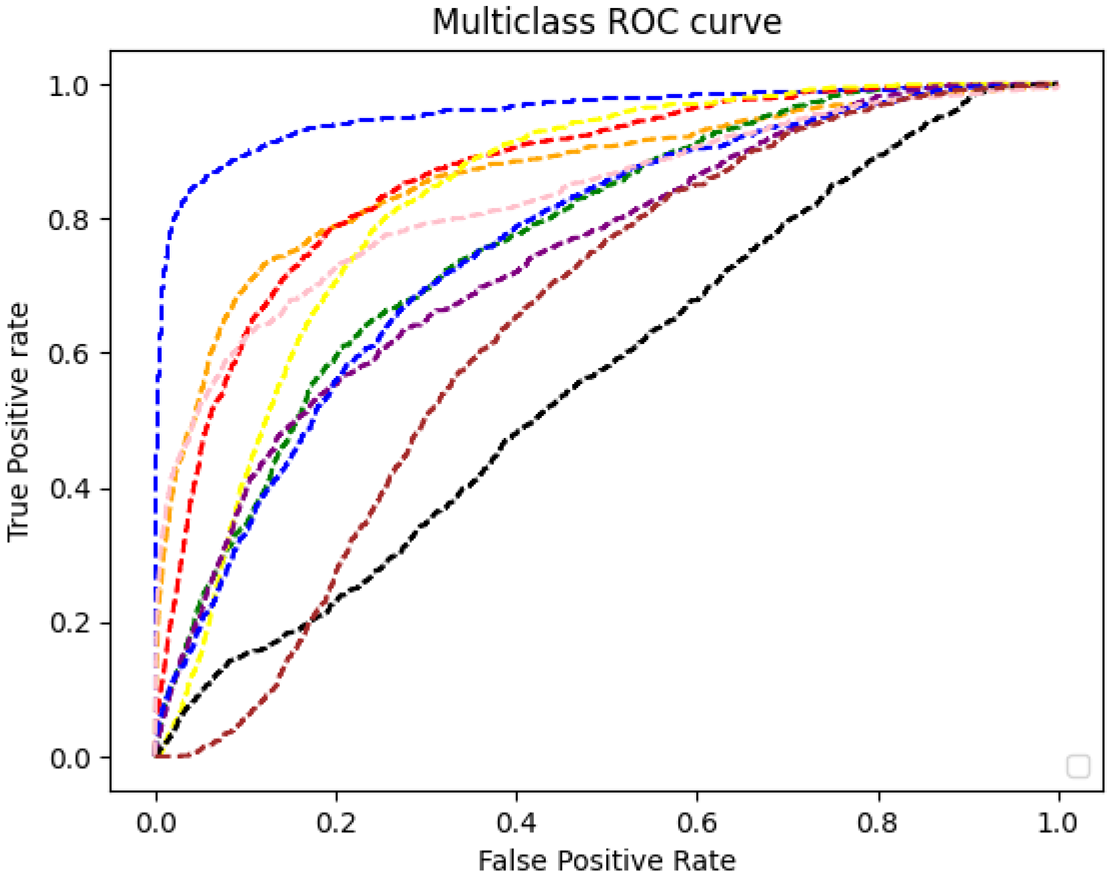
```
******Decision tree********
********Train*******
Precision:  0.8065166666666667
Recall:  0.8065166666666667
F1 Score:  0.8065166666666667
Confusion : [[4471 1077    4   31   10    9    0    3  219  176]
 [ 407 5254   30   14    2   14    1    0  214   64]
 [ 541   83 5013   38   10  249    1   40   12   13]
 [ 244  307   40 4905  188   55  110   80   35   36]
 [   4  133    3  284 4904  274   10    5  364   19]
 [  44  198  182  100  157 4878    5   20  411    5]
 [  11  119   42  170    4  117 4965  531   20   21]
 [  37   62   48  397  109   13  750 4444   78   62]
 [ 215  277  511   13   11  104    2    5 4848   14]
 [  99  234  168   66  120  204   56   62  282 4709]]
*******Test*******
Precision:  0.3508
Recall:  0.3508
F1 Score:  0.3508
Confusion : [[407 240  10  14   0   8   8   2  42 269]
 [ 77 524  28  28   1  23   3   1  41 274]
 [ 36  17 582 138  94 105   1   4   1  22]
 [ 45  14  23 640   5  12  92  71  94   4]
 [  4   8   7  96 141  85 182 102 375   0]
 [  1  13 161  30 168 373  25  15 201  13]
 [ 76 112  76  55   0  39 307 314  10  11]
 [ 90  69  43 352  10   3 194 138 100   1]
 [ 52  49   6   0   0 421   0   0 382  90]
 [ 46 821   0  25   1  35   8   2  48  14]]
```

In [6]:

```python
model=SVC(probability=True)
model.fit(x_train,y_train)
train_predict=model.predict(x_train,)
test_predict=model.predict(x_test)
pred_prob=model.predict_proba(x_test)
fpr = {}
tpr = {}
thresh ={}
n_class = 10
#looping the nclass in the target column
for i in range(n_class):
    fpr[i], tpr[i], thresh[i] = roc_curve(y_test, pred_prob[:,i], pos_label=i)
# evaluating the model using auc score
roc_auc=roc_auc_score(y_test,pred_prob,multi_class='ovr',average='macro')
# plotting
# plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.plot(fpr[0], tpr[0], linestyle='--',color='orange')
plt.plot(fpr[1], tpr[1], linestyle='--',color='green')
plt.plot(fpr[2], tpr[2], linestyle='--',color='blue')
plt.plot(fpr[3], tpr[3], linestyle='--',color='red')
plt.plot(fpr[4], tpr[4], linestyle='--',color='yellow')
plt.plot(fpr[5], tpr[5], linestyle='--',color='purple')
plt.plot(fpr[6], tpr[6], linestyle='--',color='blue')
plt.plot(fpr[7], tpr[7], linestyle='--',color='black')
plt.plot(fpr[8], tpr[8], linestyle='--',color='pink')
plt.plot(fpr[9], tpr[9], linestyle='--',color='brown')
plt.title('Multiclass ROC curve-SVC')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive rate')
plt.legend(loc='lower right')
plt.show()
print('roc_auc:',roc_auc)
print('******SVC********')
print('********Train*******')
print("Precision: ",precision_score(y_train,train_predict,pos_label='positive',aver
print("Recall: ",recall_score(y_train,train_predict,pos_label='positive',average='m
print("F1 Score: ",f1_score(y_train,train_predict,pos_label='positive',average='mic
print('Confusion :',confusion_matrix(y_train,train_predict))

print('*******Test*******')
print("Precision: ",precision_score(y_test,test_predict,pos_label='positive',averag
print("Recall: ",recall_score(y_test,test_predict,pos_label='positive',average='mic
print("F1 Score: ",f1_score(y_test,test_predict,pos_label='positive',average='micro
print('Confusion :',confusion_matrix(y_test,test_predict))
print('\n \n')
```

```
No artists with labels found to put in legend.  Note that artists whose label start
with an underscore are ignored when legend() is called with no argument.
```



```
0.7826503444444445
******SVC********
********Train*******
Precision:  0.9653666666666667
Recall:  0.9653666666666667
F1 Score:  0.9653666666666667
Confusion : [[5743  181    0   19    3    0    0    2   36   16]
 [  40 5900    1   20    6    1    2    1   13   16]
 [  32    4 5947    7    0    5    1    3    0    1]
 [  41   13    7 5775   34   42    9   64   11    4]
 [   3    4    0   26 5859   61    5    4   11   27]
 [   1    1    3   37   73 5872    1    3    7    2]
 [   3    3    2   39   14    4 5719  203    1   12]
 [   5   19    1   73   12    5  381 5483    9   12]
 [  49   11    1   34   16    5    3    1 5864   16]
 [  11    3    1   18   62    4   76   38   27 5760]]
*******Test*******
Precision:  0.3769
Recall:  0.3769
F1 Score:  0.37689999999999996
Confusion : [[589 145   3  40   3  13  31   3  31 142]
 [392 254  15  40  34  30  85   3  17 130]
 [  2   1 811  19  11 132   3  16   2   3]
 [ 20   1  53 575  35  11 123 103  44  35]
 [  0   0   0 121 248  22 239 204  52 114]
 [  0   0  56   4 281 241  41  10 104 263]
 [ 20  23  27  41   4   5 304 517  33  26]
 [ 33  29  71 388  21  12 210 144  91   1]
 [ 26   5   7   0  56 258   0   2 545 101]
 [116 394   0  64 118  42  15  35 158  58]]
```

In [7]:

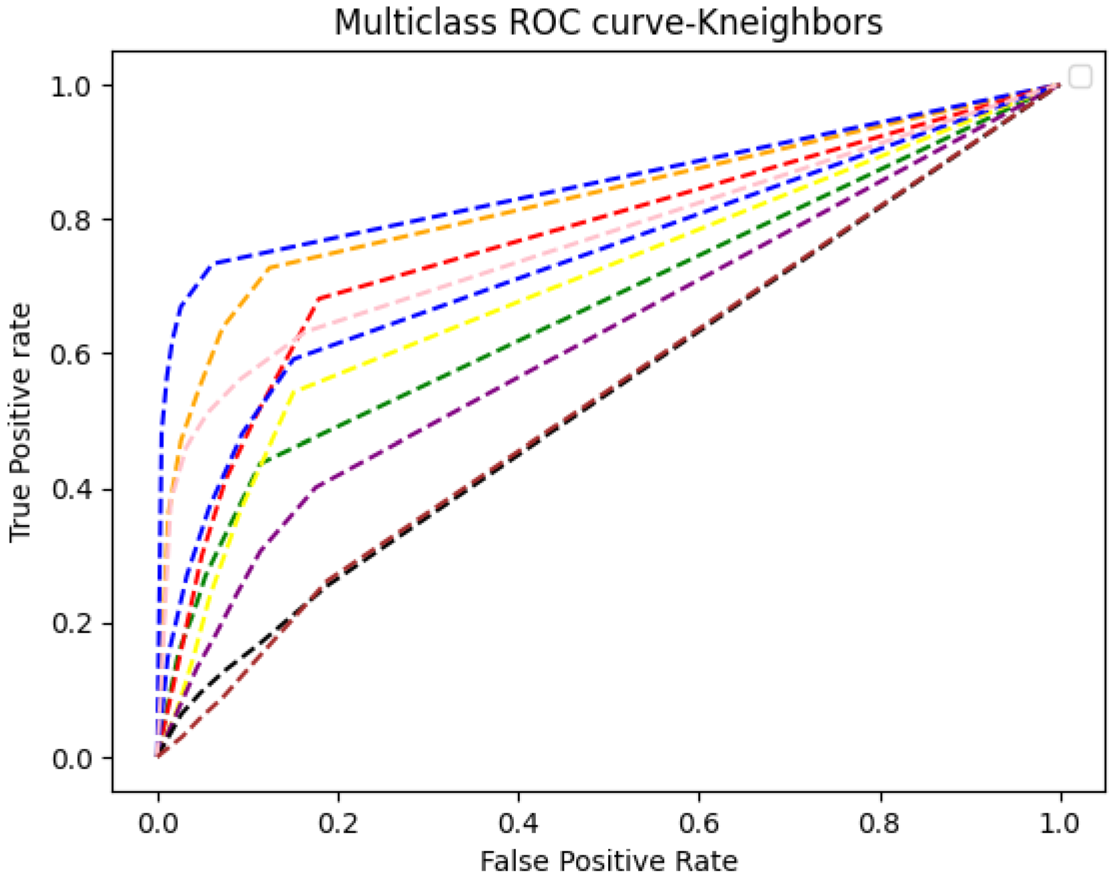
```python
model=KNeighborsClassifier()
model.fit(x_train,y_train)
train_predict=model.predict(x_train,)
test_predict=model.predict(x_test)
pred_prob=model.predict_proba(x_test)
fpr = {}
tpr = {}
thresh ={}

n_class = 10
#looping the nclass in the target column
for i in range(n_class):
    fpr[i], tpr[i], thresh[i] = roc_curve(y_test, pred_prob[:,i], pos_label=i)
# evaluating the model using auc score
roc_auc=roc_auc_score(y_test,pred_prob,multi_class='ovr',average='macro')

# plotting
plt.plot(fpr[0], tpr[0], linestyle='--',color='orange')
plt.plot(fpr[1], tpr[1], linestyle='--',color='green')
plt.plot(fpr[2], tpr[2], linestyle='--',color='blue')
plt.plot(fpr[3], tpr[3], linestyle='--',color='red')
plt.plot(fpr[4], tpr[4], linestyle='--',color='yellow')
plt.plot(fpr[5], tpr[5], linestyle='--',color='purple')
plt.plot(fpr[6], tpr[6], linestyle='--',color='blue')
plt.plot(fpr[7], tpr[7], linestyle='--',color='black')
plt.plot(fpr[8], tpr[8], linestyle='--',color='pink')
plt.plot(fpr[9], tpr[9], linestyle='--',color='brown')
plt.title('Multiclass ROC curve-Kneighbors')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive rate')
plt.legend(loc='best')
plt.show()
print('******KNeighborsClassifier********')
print('********Train*******')
print('roc_auc:',roc_auc)
print("Precision: ",precision_score(y_train,train_predict,pos_label='positive',aver
print("Recall: ",recall_score(y_train,train_predict,pos_label='positive',average='m
print("F1 Score: ",f1_score(y_train,train_predict,pos_label='positive',average='mic
print('Confusion :',confusion_matrix(y_train,train_predict))

print('*******Test*******')
print("Precision: ",precision_score(y_test,test_predict,pos_label='positive',averag
print("Recall: ",recall_score(y_test,test_predict,pos_label='positive',average='mic
print("F1 Score: ",f1_score(y_test,test_predict,pos_label='positive',average='micro
print('Confusion :',confusion_matrix(y_test,test_predict))
print('\n \n')
```

```
No artists with labels found to put in legend.  Note that artists whose label start
with an underscore are ignored when legend() is called with no argument.
```



```
******KNeighborsClassifier********
********Train*******
Precision:  0.9740666666666666
Recall:  0.9740666666666666
F1 Score:  0.9740666666666666
Confusion : [[5753  193    0   13    3    0    1    1   27    9]
 [  32 5940    0    6    4    1    0    1    7    9]
 [  29    6 5949   10    0    6    0    0    0    0]
 [  39   12    5 5822   27   35    3   43   11    3]
 [   2    3    0   16 5912   44    2    0    5   16]
 [   0    1    3   25   38 5923    0    3    4    3]
 [   2    3    3   16   11    2 5802  138    1   22]
 [   9   14    1   54   14    2  271 5622    4    9]
 [  41    8    0    7   12    7    2    0 5916    7]
 [  12    1    0    8   50    3   75   34   12 5805]]
*******Test*******
Precision:  0.3716
Recall:  0.3716
F1 Score:  0.3716
Confusion : [[578 159   3  35   8  15   8   5  60 129]
 [351 290   8  46  69  42  14   2  23 155]
 [  3   5 640  60  18 250   5  12   1   6]
 [ 30   5  51 478  71  15 134 128  63  25]
 [  4   2   0 156 309  56 205 170  48  50]
 [  0   4  24  13 338 263  46  21 103 188]
 [  7  19  51  68   4  22 409 383  15  22]
 [ 39  28  55 388  26  18 217 136  90   3]
 [ 24   5   0   0  43 305   1   1 516 105]
 [ 91 347   0  89  76 121  10   8 161  97]]
```

In [8]:

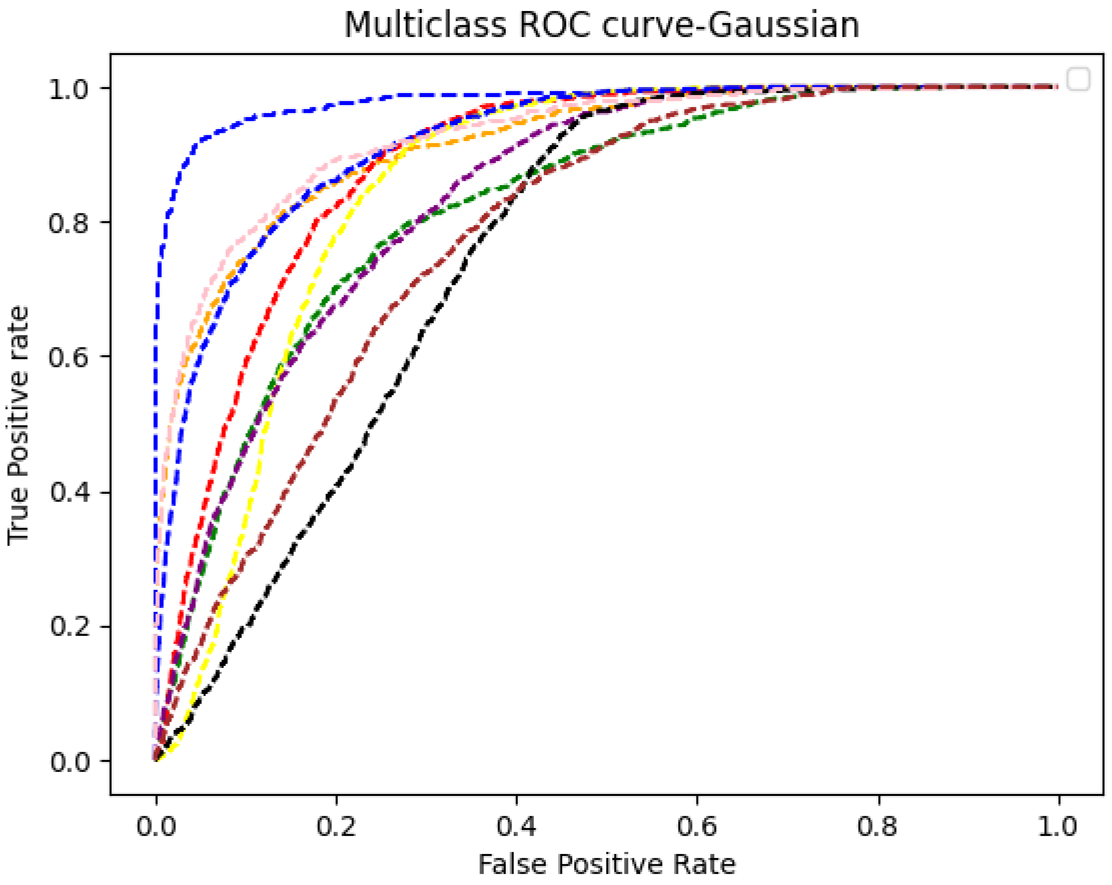
```
model= GaussianNB()
model.fit(x_train,y_train)
train_predict=model.predict(x_train)
test_predict=model.predict(x_test)
pred_prob=model.predict_proba(x_test)
fpr = {}
tpr = {}
thresh ={}

n_class = 10
#looping the nclass in the target column
for i in range(n_class):
    fpr[i], tpr[i], thresh[i] = roc_curve(y_test, pred_prob[:,i], pos_label=i)
# evaluating the model using auc score
roc_auc=roc_auc_score(y_test,pred_prob,multi_class='ovr',average='macro')
# plotting
plt.plot(fpr[0], tpr[0], linestyle='--',color='orange')
plt.plot(fpr[1], tpr[1], linestyle='--',color='green')
plt.plot(fpr[2], tpr[2], linestyle='--',color='blue')
plt.plot(fpr[3], tpr[3], linestyle='--',color='red')
plt.plot(fpr[4], tpr[4], linestyle='--',color='yellow')
plt.plot(fpr[5], tpr[5], linestyle='--',color='purple')
plt.plot(fpr[6], tpr[6], linestyle='--',color='blue')
plt.plot(fpr[7], tpr[7], linestyle='--',color='black')
plt.plot(fpr[8], tpr[8], linestyle='--',color='pink')
plt.plot(fpr[9], tpr[9], linestyle='--',color='brown')
plt.title('Multiclass ROC curve-Gaussian ')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive rate')
plt.legend(loc='best')
plt.show
print('******GaussianNB********')
print('********Train*******')
print('roc_auc:',roc_auc)
print("Precision: ",precision_score(y_train,train_predict,pos_label='positive',aver
print("Recall: ",recall_score(y_train,train_predict,pos_label='positive',average='m
print("F1 Score: ",f1_score(y_train,train_predict,pos_label='positive',average='mic
print('Confusion :',confusion_matrix(y_train,train_predict))

print('*******Test*******')
print("Precision: ",precision_score(y_test,test_predict,pos_label='positive',averag
print("Recall: ",recall_score(y_test,test_predict,pos_label='positive',average='mic
print("F1 Score: ",f1_score(y_test,test_predict,pos_label='positive',average='micro
print('Confusion :',confusion_matrix(y_test,test_predict))
print('\n \n')
```

```
No artists with labels found to put in legend.  Note that artists whose label start
with an underscore are ignored when legend() is called with no argument.
```

```
******GaussianNB********
********Train*******
Precision:  0.8678333333333333
Recall:  0.8678333333333333
F1 Score:  0.8678333333333333
Confusion : [[5120  507   35  100    4    0    2   34  169   29]
 [ 192 5376   13  179   26    1    0    3  128   82]
 [ 111   15 5677   17    1  136    0   40    2    1]
 [ 123    4   46 5443   83   45   76  169    2    9]
 [  12    3    7   96 5569  231    1   20   31   30]
 [   1   18   85  191  147 5507    5   19   25    2]
 [   6    1   52   65   21   26 4694 1128    2    5]
 [  65   12   29  195   89   13 1386 4201    3    7]
 [ 313  131   19  156   31   13    3    9 5247   78]
 [ 116   11   10   45  206    7   22  144  203 5236]]
*******Test*******
Precision:  0.441
Recall:  0.441
F1 Score:  0.441
Confusion : [[626 187  18  14   1   5   6  32  75  36]
 [255 395  20  22  65  15   0  59  28 141]
 [ 10   1 905   3   3  64   2  10   1   1]
 [ 19   5  65 478  13   0  91 296  33   0]
 [  0   2   0 231  80   7   4 491 132  53]
 [  0   0 127  28 248 183   2  86 159 167]
 [  3  14 149  37   0   3 601 183   7   3]
 [ 71  31  19 203  20   9 359 255  32   1]
 [ 54   9  32   0   1 114   0   2 698  90]
 [ 41 444   0 144   8  42  33   9  90 189]]
```


Multiclass ROC curve-Gaussian
True Positive rate
False Positive Rate
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

In [ ]:

In [ ]:

In [ ]: