In [1]:

```
import numpy as np
from sklearn.decomposition import PCA

# Extracting the x_train data and reshapeing it to 2 dimensional array
Train_data_25 =np.load(r"C:\Users\barathy\Downloads\kannada_mnist\Kannada_MNIST_dat
data_xtr_25 =Train_data_25['arr_0']
data1_25=data_xtr_25.reshape(-1,28*28)

#Extracting the x_test data and reshapeing it to 2 dimensional array
Test_data_25=np.load(r"C:\Users\barathy\Downloads\kannada_mnist\Kannada_MNIST_datat
data_xt_25=Test_data_25['arr_0']
data2_25=data_xt_25.reshape(-1,28*28)

#Extracting the y_train data and reshapeing it to 2 dimensional array
Train_data_25=np.load(r"C:\Users\barathy\Downloads\kannada_mnist\Kannada_MNIST_data
data_ytr_25=Train_data_25['arr_0']
data3_25=data_ytr_25.reshape(-1,1)


# #Extracting the y_test data and reshapeing it to 2 dimensional array
Test_data_25=np.load(r"C:\Users\barathy\Downloads\kannada_mnist\Kannada_MNIST_datat
data_yt_25=Test_data_25['arr_0']
data4_25 =data_yt_25.reshape(-1,1)

# Initialize and fit PCA to reduce to 10 components
n_components = 25
pca = PCA(n_components=n_components)

# Fit and transform the training data
x_train_pca = pca.fit_transform(data1_25)
x_train_p25=x_train_pca

# Fit and transform the testing data
x_test_pca=pca.fit_transform(data2_25)
x_test_p25=x_test_pca
```

In [6]:

```
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
import warnings
warnings.filterwarnings('ignore')

x_train=x_train_p25
x_test=x_test_p25
y_train=data3_25
y_test=data4_25
```

In [7]:

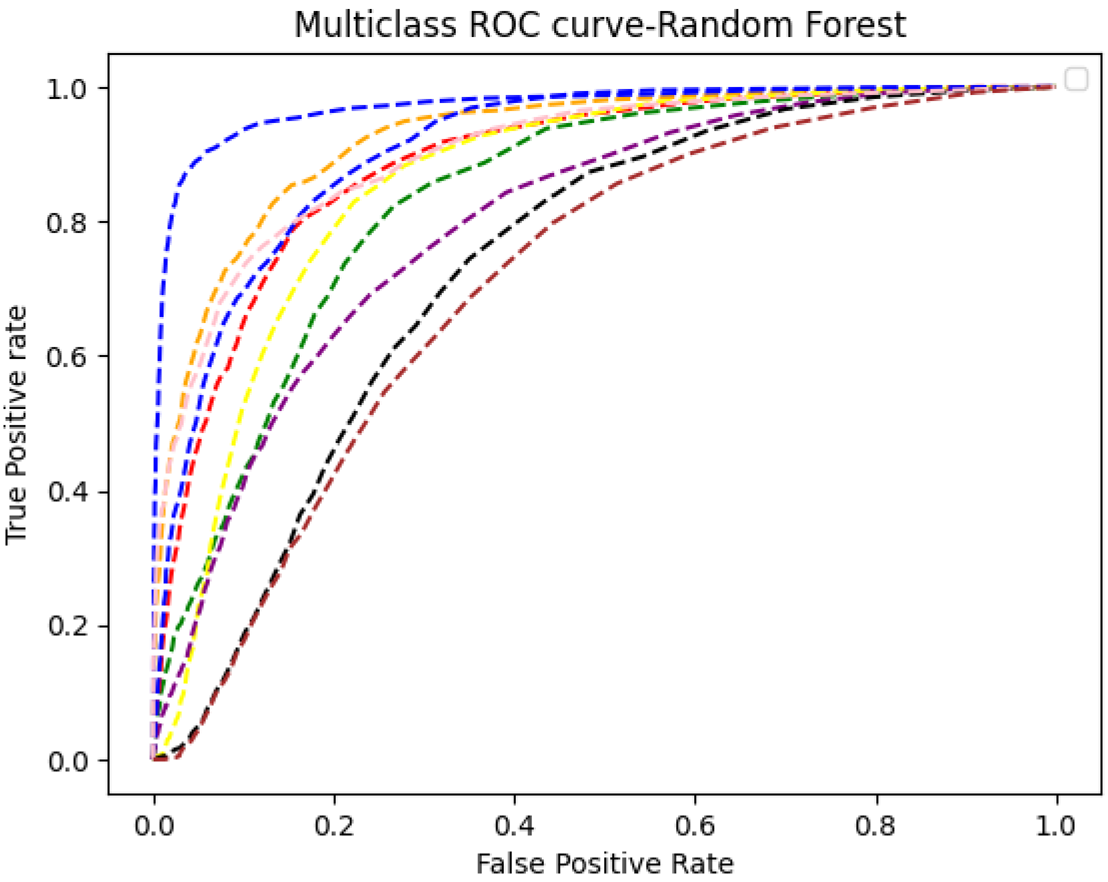
```
# performing predictions using Random Forest classifier
model=RandomForestClassifier(n_estimators=100)
```

```
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-Random Forest ')
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.
```

```
*********Randomforest************
********Train*******
roc_auc: 0.8532888166666668
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:  0.4214
Recall:  0.4214
F1 Score:  0.4214
Confusion : [[655 131   6  35   2  19   6   1  83  62]
 [391 259   4  57  53  49   9   0  30 148]
 [ 11   4 800  24  17 136   1   2   2   3]
 [ 15   1  73 652  22  10  58 116  50   3]
 [  1   2   1 247 251  28  68 176 136  90]
 [  1   1  31  27 286 298   5  33 190 128]
 [  5  26  37  52   1   9 524 271  24  51]
 [ 38  25  45 393  22  21 332  84  39   1]
 [ 43   1   2   1   4 310   0   3 540  96]
 [118 426   0  77  11  79  11  12 115 151]]
```



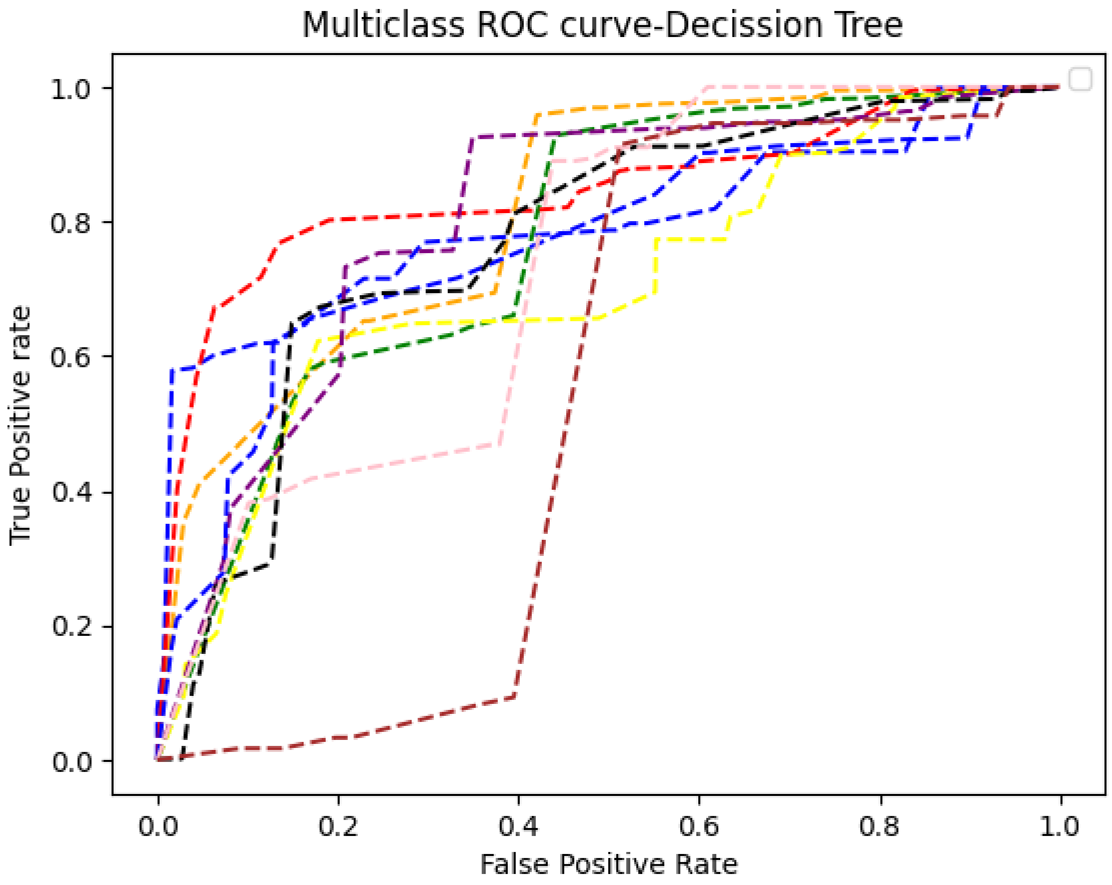
In [8]:

```
# performing predictions using Decision tree
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-Decission 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*******
roc_auc: 0.7536684499999999
Precision:  0.8083666666666667
Recall:  0.8083666666666667
F1 Score:  0.8083666666666667
Confusion : [[4475 1086    4   20   10    9   10    4  219  163]
 [ 396 5268   30   13    2   14    1    1  214   61]
 [ 540   83 5013   26   10  250    1   52   12   13]
 [ 244  309   40 4731  188   55  104  260   35   34]
 [   4  134    3  282 4902  276   10    7  364   18]
 [  44  198  181  100  155 4881    2   23  411    5]
 [  11  122   42  124    4  117 4959  583   20   18]
 [  38   64   48  337  109   13  517 4737   78   59]
 [ 214  278  511   12   11  104    3    5 4848   14]
 [  98  256  167   56  120  205   72   56  282 4688]]
*******Test*******
Precision:  0.3623
Recall:  0.3623
F1 Score:  0.3623
Confusion : [[411 241  10   9   0   8  12   3  42 264]
 [ 76 531  28  24   1  23   3   5  41 268]
 [ 36  18 582 138  94 105   1   4   1  21]
 [ 45  14  23 569   5  12  66 168  94   4]
 [  4   8   7  86 141  85 180 114 375   0]
 [  1  13 160  30 168 374  22  18 201  13]
 [ 74 113  76  50   0  40 362 264  10  11]
 [ 90  69  43  38  10   3 389 257 100   1]
 [ 50  51   6   0   0 421   0   0 382  90]
 [ 46 821   0  17   1  35  16   2  48  14]]
```



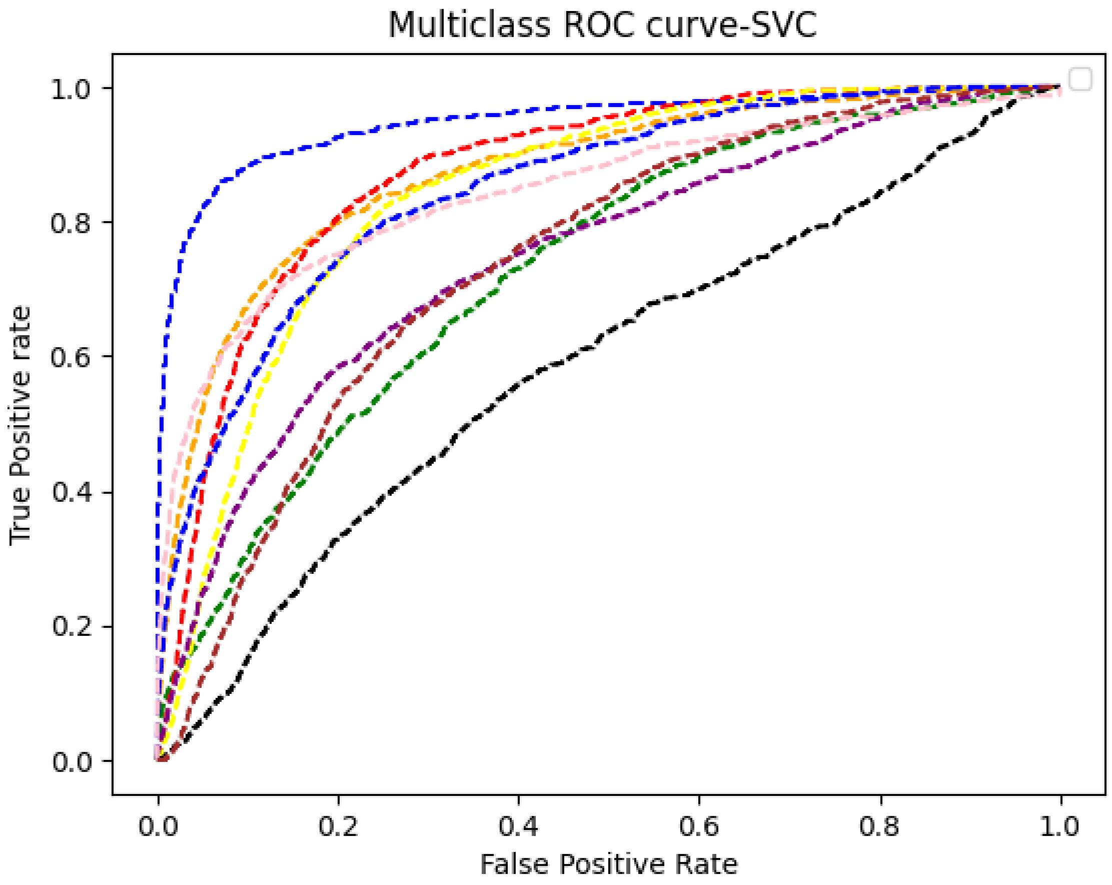
```
In [9]: # performing predictions using svc
        model=SVC(probability=True)
        model.fit(x_train,y_train)
        train_predict=model.predict(x_train,)
        test_predict=model.predict(x_test)
        pred_prob=model.predict_proba(x_test)
        fpr = {}
        tpr = {}
        thresh ={}

        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[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 ')
```

```python
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive rate')
plt.legend(loc='best')
plt.show
print('******SVC********')
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.
```

```
******SVC********
********Train*******
roc_auc: 0.8030229555555556
Precision:  0.9891833333333333
Recall:  0.9891833333333333
F1 Score:  0.9891833333333332
Confusion : [[5862  110    0   10    0    1    3    0   11    3]
 [   4 5975    0    5    2    0    0    3    1   10]
 [  13    1 5977    9    0    0    0    0    0    0]
 [  13    3    1 5937   15    6    1   21    2    1]
 [   0    1    0   12 5971    9    0    2    2    3]
 [   0    0    0    8   22 5968    1    1    0    0]
 [   0    0    1    8    3    1 5920   55    0   12]
 [   2    7    0   26    5    2   95 5856    1    6]
 [   7    1    0    0    3    3    0    1 5983    2]
 [   2    3    0    7   17    1   62    6    0 5902]]
*******Test*******
Precision:  0.402
Recall:  0.402
F1 Score:  0.402
Confusion : [[582  96   6  80   7  15   9  28  99  78]
 [383 154  31  94  61  41  52  37  45 102]
 [  8   0 788  30  19 131   4  14   5   1]
 [ 22   3  69 691  18  10  38 107  18  24]
 [  1   2   1 241 294   7  74 299  56  25]
 [  4   2  58  14 285 294  22  72  85 164]
 [ 17  32 135  75   2  10 384 296  45   4]
 [ 17  19  42 533  32  27 141 162  27   0]
 [ 53   3  11   1  43 255   2   3 509 120]
 [165 169   0 112  84 113   4 153  38 162]]
```



In [10]:

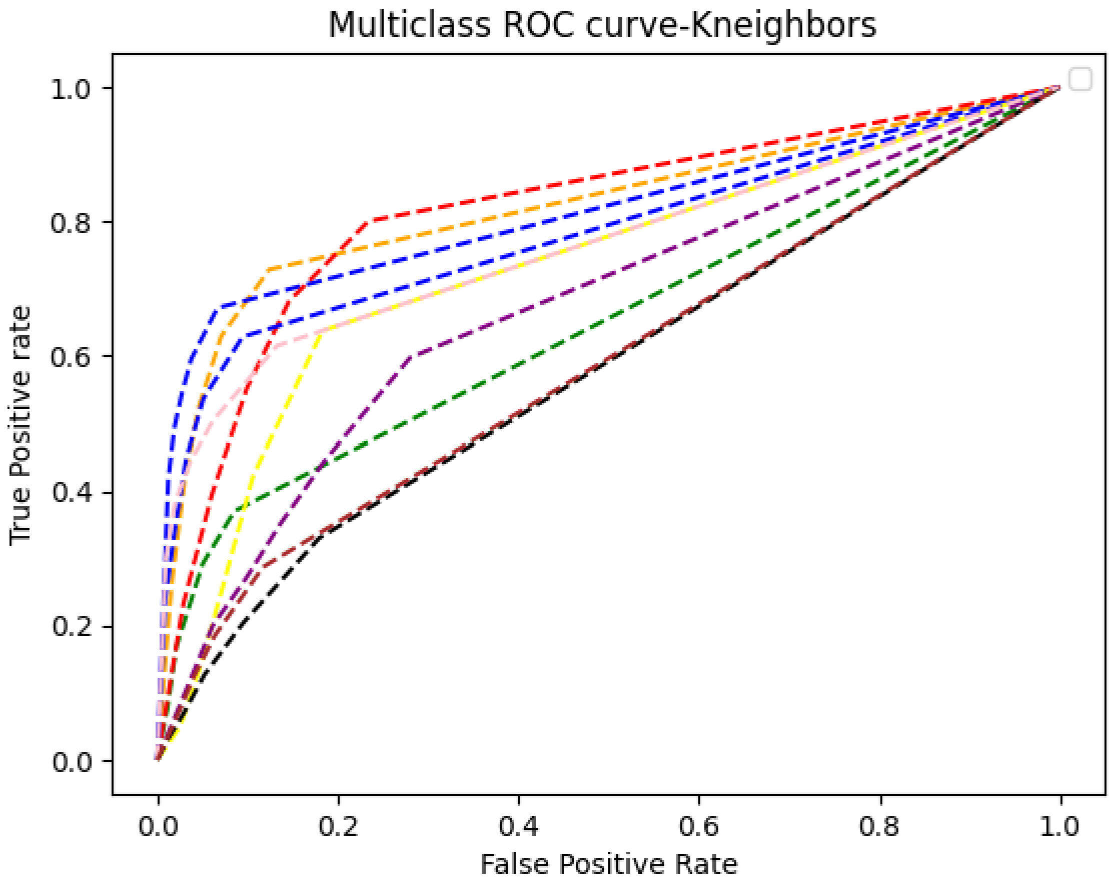
```
# performing predictions using Kneighbors
model=KNeighborsClassifier()
model.fit(x_train,y_train)
train_predict=model.predict(x_train,)
test_predict=model.predict(x_test)
pred_prob=model.predict_proba(x_test)
fpr = {}
tpr = {}
thresh ={}

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 ')
```

```python
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*******
roc_auc: 0.720558911111111
Precision:  0.9885
Recall:  0.9885
F1 Score:  0.9885
Confusion : [[5823  164    0    6    1    0    1    0    5    0]
 [   8 5987    0    1    1    0    1    0    0    2]
 [  18    3 5971    7    0    1    0    0    0    0]
 [  17    4    2 5932   13    6    0   23    3    0]
 [   0    0    0    4 5987    7    1    0    0    1]
 [   0    0    1    4   17 5975    1    1    1    0]
 [   0    0    2    4    9    0 5909   59    1   16]
 [   1    8    1   24   10    1   87 5864    0    4]
 [  26    2    0    0    3    2    0    0 5965    2]
 [   2    1    0    1   13    1   72    9    4 5897]]
*******Test*******
Precision:  0.3888
Recall:  0.3888
F1 Score:  0.3888
Confusion : [[566 140   3  49  25  39  10  23  83  62]
 [330 241   2  89 145  72   1  16  29  75]
 [  6   2 566  86  18 300   8   8   4   2]
 [ 16  15  44 637  62  48  42  94  34   8]
 [  4   5   0 338 321 105  37 113  43  34]
 [  1   7  31  32 381 372  16  27  37  96]
 [  7  14 158 116   5  24 464 200   8   4]
 [ 25  10  73 325  49 118 219 137  42   2]
 [ 66   2   1   7  51 308   0   4 454 107]
 [104 170   0  85  42 331   7  42  89 130]]
```



In [11]:

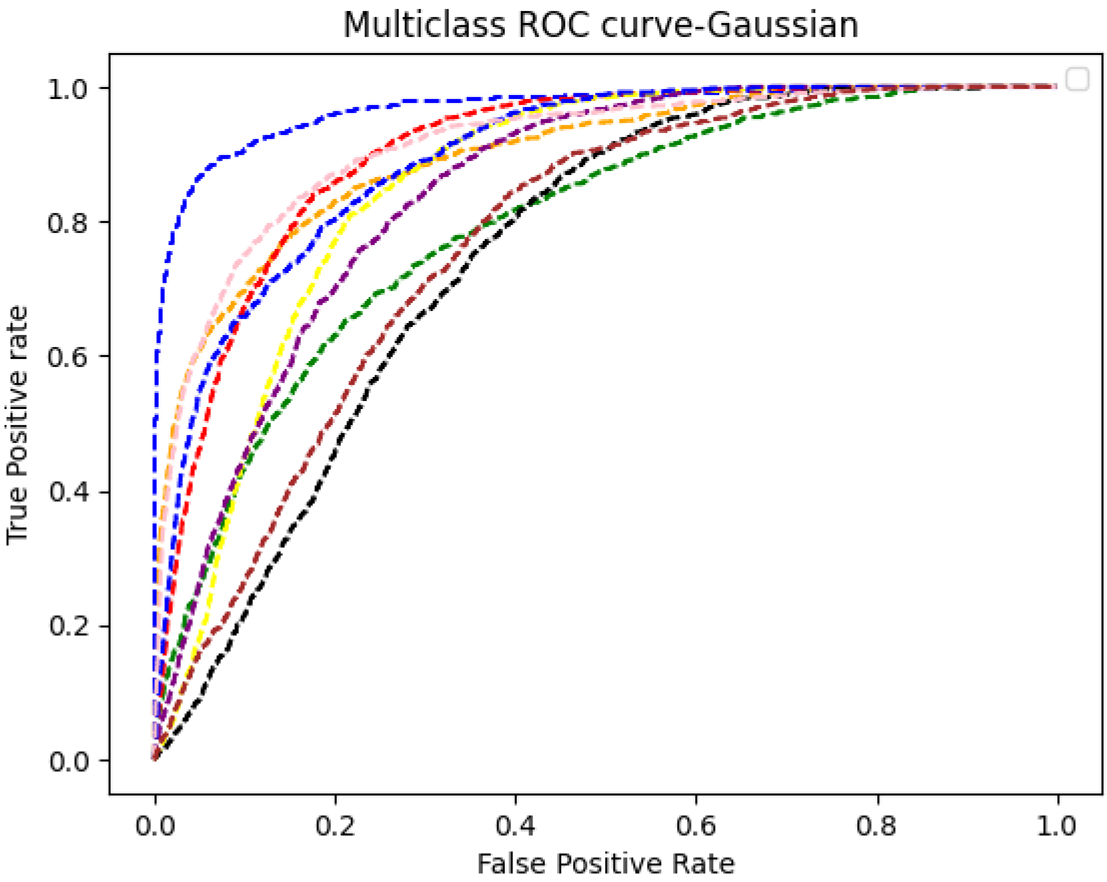
```
# performing predictions using gaussian
model= GaussianNB()
model.fit(x_train,y_train)
train_predict=model.predict(x_train)
test_predict=model.predict(x_test)
pred_prob=model.predict_proba(x_test)
fpr = {}
tpr = {}
thresh ={}

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*******
roc_auc: 0.8590247555555557
Precision:  0.89985
Recall:  0.89985
F1 Score:  0.89985
Confusion : [[5205  457   43   62    3    0    6   15  194   15]
 [ 136 5578    4  119   21    0    0    1   73   68]
 [  94   19 5839   13    1    7    1   19    6    1]
 [ 131    4   41 5309   75   40   63  328    5    4]
 [   9    3   10   41 5589  194    8   77   37   32]
 [   2   26   61  111  146 5553   10   47   41    3]
 [   5    1   28   44   18   33 5368  487    6   10]
 [  47   15   19  193   51   18  871 4777    2    7]
 [ 327  179   24   50   38   17    4   16 5282   63]
 [  30    1    8    5  115    5   37  149  159 5491]]
*******Test*******
Precision:  0.4246
Recall:  0.4246
F1 Score:  0.4246
Confusion : [[604 179  39  12   3   5   8  12  96  42]
 [243 347  73  23  93  15   5  52  31 118]
 [  8   4 871   8   3  75   1  25   1   4]
 [ 21   2 108 390  29   3  53 358  36   0]
 [  2   1   0 132 131   5  22 504 156  47]
 [  0   0 141  18 187 177   4 165 168 140]
 [  7  16 138  17   1   1 536 268  14   2]
 [ 78  28  38  76  21   9 361 359  30   0]
 [ 57  14  37   0   1 122   0   3 669  97]
 [ 67 440   1  85  10  55  33  47 100 162]]
```


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 [ ]: