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_30 =np.load(r"C:\Users\barathy\Downloads\kannada_mnist\Kannada_MNIST_dat
data_xtr_30 =Train_data_30['arr_0']
data1_30=data_xtr_30.reshape(-1,28*28)

#Extracting the x_test data and reshapeing it to 2 dimensional array
Test_data_30=np.load(r"C:\Users\barathy\Downloads\kannada_mnist\Kannada_MNIST_datat
data_xt_30=Test_data_30['arr_0']
data2_30=data_xt_30.reshape(-1,28*28)

#Extracting the y_train data and reshapeing it to 2 dimensional array
Train_data_30=np.load(r"C:\Users\barathy\Downloads\kannada_mnist\Kannada_MNIST_data
data_ytr_30=Train_data_30['arr_0']
data3_30=data_ytr_30.reshape(-1,1)


# #Extracting the y_test data and reshapeing it to 2 dimensional array
Test_data_30=np.load(r"C:\Users\barathy\Downloads\kannada_mnist\Kannada_MNIST_datat
data_yt_30=Test_data_30['arr_0']
data4_30 =data_yt_30.reshape(-1,1)

# Initialize and fit PCA to reduce to 10 components
n_components = 30
pca = PCA(n_components=n_components)

# Fit and transform the training data
x_train_pca = pca.fit_transform(data1_30)
x_train_p30=x_train_pca

# Fit and transform the testing data
x_test_pca=pca.fit_transform(data2_30)
x_test_p30=x_test_pca
```

In [7]:
```
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_p30
x_test=x_test_p30
y_train=data3_30
y_test=data4_30
```

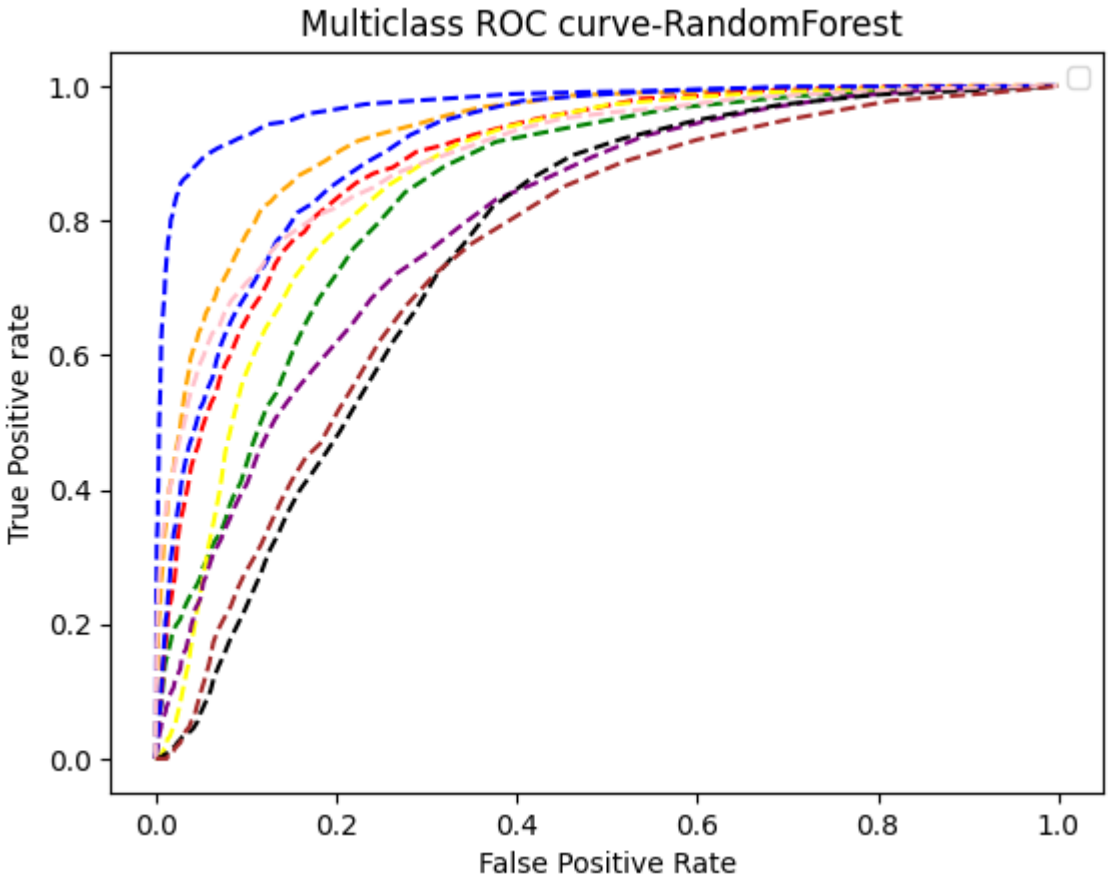
In [8]:
```
# performing predictions using Random Forest classifier
```

```
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-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.
```

```
*********Randomforest************
********Train*******
roc_auc: 0.8610816722222221
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:  0.4335
Recall:  0.4335
F1 Score:  0.4335
Confusion : [[649 144   5  35   3  15   4   3  87  55]
 [432 272   3  49  48  42   6   0  31 117]
 [  9   5 798  34  19 123   2   1   6   3]
 [ 16   2  72 660  18   9  55 105  60   3]
 [  2   0   1 223 270  42  88 151 132  91]
 [  2   1  30  30 275 303   7  28 207 117]
 [  5  24  47  56   1  10 517 263  47  30]
 [ 34  21  45 389  25  23 309 109  43   2]
 [ 41   4   0   1   4 308   1   0 541 100]
 [ 89 387   0  71  18 101   8   6 104 216]]
```



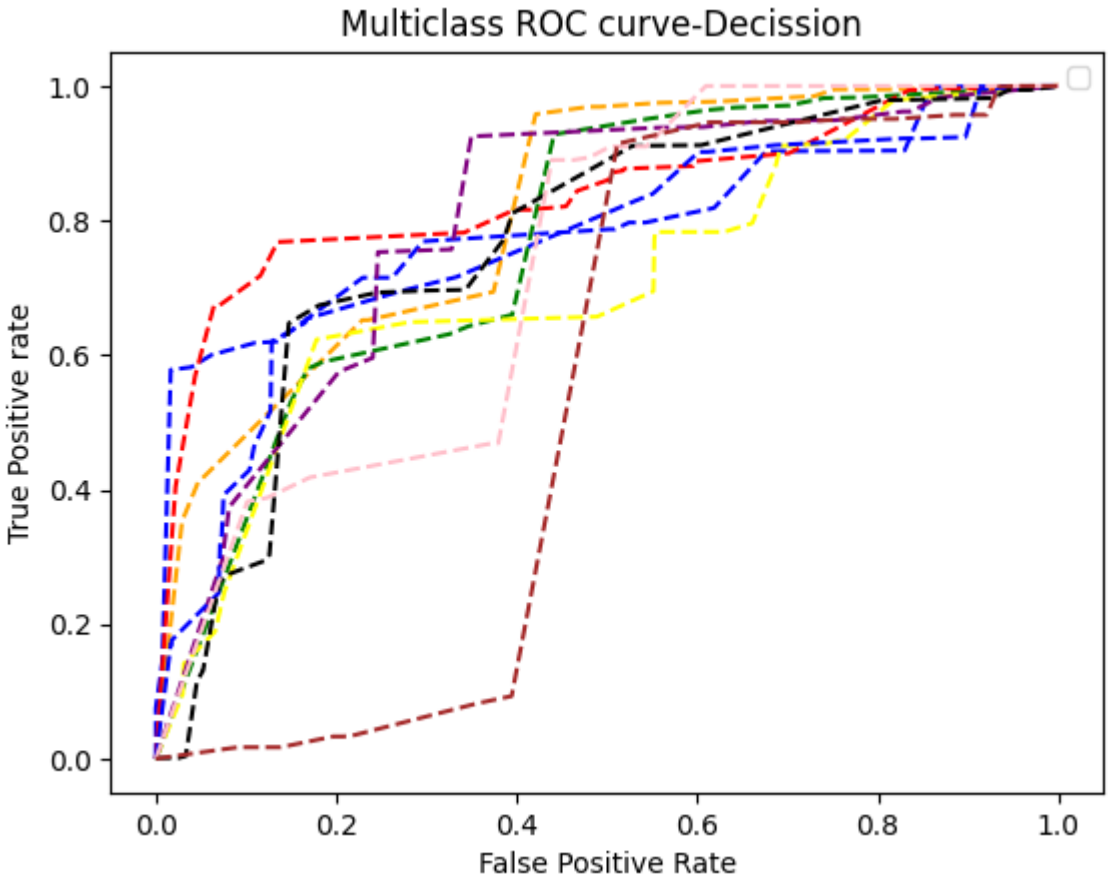
In [9]:

```
# 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 ')
```

```
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: 0.7524557666666667
Precision:  0.80855
Recall:  0.80855
F1 Score:  0.80855
Confusion : [[4474 1087    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  100  264   35   34]
 [   4  134    3  282 4902  276   10    7  364   18]
 [  44  198  181  100  155 4881    1   24  411    5]
 [  11  122   42  124    4  117 4932  610   20   18]
 [  38   64   48  337  109   13  478 4776   78   59]
 [ 214  278  511   12   11  104    3    5 4848   14]
 [  98  256  167   56  120  205   74   54  282 4688]]
*******Test*******
Precision:  0.36
Recall:  0.36
F1 Score:  0.36
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  56 178  94   4]
 [  4   8   7  86 141  85 164 130 375   0]
 [  1  13 160  30 168 374  24  16 201  13]
 [ 74 113  76  50   0  40 332 294  10  11]
 [ 90  69  43  38  10   3 382 264 100   1]
 [ 50  51   6   0   0 421   0   0 382  90]
 [ 46 821   0  17   1  35  16   2  48  14]]
```



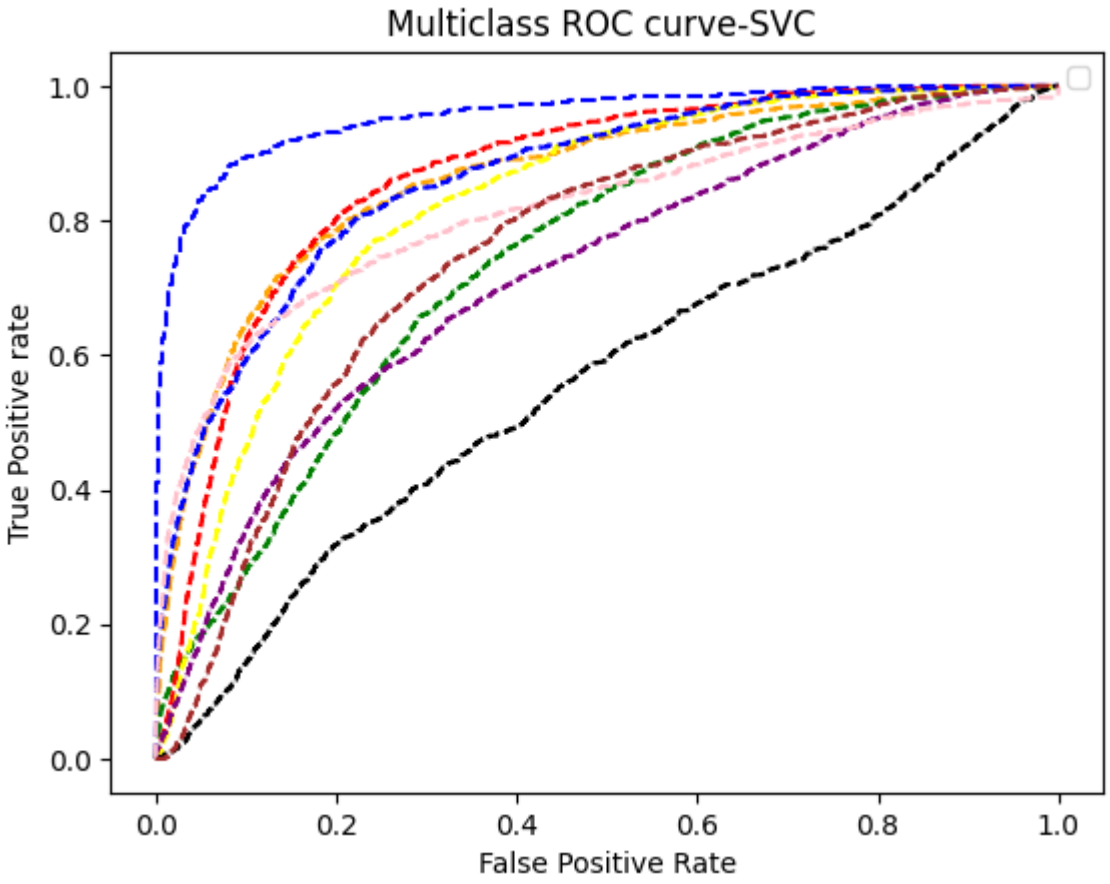
In [10]:

```
# 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 ')
```

```
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: 0.7959733888888889
Precision:  0.9909333333333333
Recall:  0.9909333333333333
F1 Score:  0.9909333333333333
Confusion : [[5882  103    0    5    0    0    3    0    5    2]
 [   3 5984    0    2    1    0    0    1    1    8]
 [  14    1 5976    8    0    0    0    0    0    1]
 [  10    1    0 5950    9    3    2   22    2    1]
 [   0    2    0    8 5975    8    1    2    2    2]
 [   0    0    0    5   18 5975    1    1    0    0]
 [   0    0    0    5    3    1 5931   49    0   11]
 [   1    5    0   17    2    1   88 5878    1    7]
 [   6    1    0    0    4    3    0    1 5984    1]
 [   2    1    0    4   13    1   55    3    0 5921]]
*******Test*******
Precision:  0.3906
Recall:  0.3906
F1 Score:  0.39060000000000006
Confusion : [[548 104   6  91   5  15   5  34  92 100]
 [381 144  17 122  58  47  21  46  50 114]
 [  5   0 802  22  20 136   5   6   3   1]
 [ 23   0  77 701  28  11  19 106  18  17]
 [  0   7   1 223 274  11 101 302  55  26]
 [  2   1  67  17 293 258  18  51  94 199]
 [ 17  26 127  70   4   9 395 293  57   2]
 [ 22  12  46 582  32  25 128 132  21   0]
 [ 82   3   3   1  31 293   3   3 460 121]
 [212 136   0 120  71 139   5  81  44 192]]
```



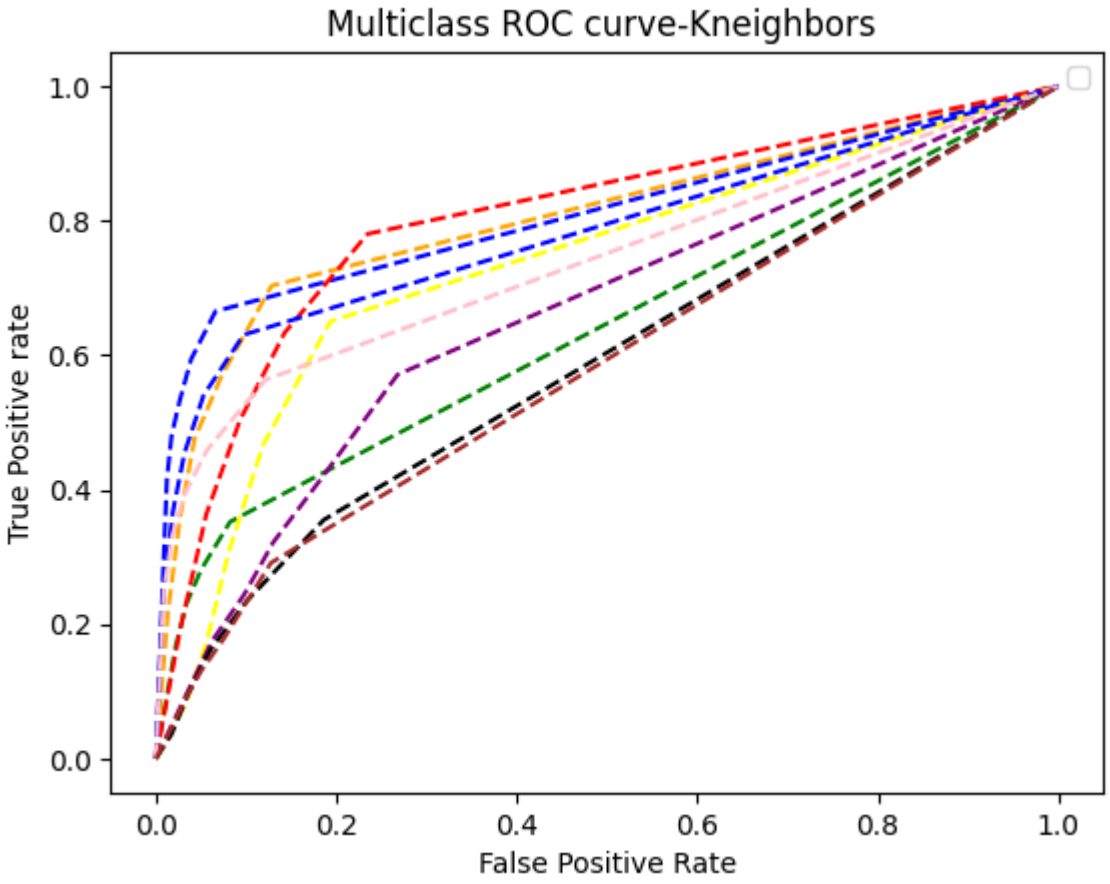
In [11]:

```
# 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 ')
```

```
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: 0.7139081055555556
Precision:  0.9893666666666666
Recall:  0.9893666666666666
F1 Score:  0.9893666666666666
Confusion : [[5827  157    0    4    1    0    1    0    8    2]
 [  10 5985    0    1    1    0    0    0    1    2]
 [  20    0 5971    7    1    1    0    0    0    0]
 [  13    4    1 5939   11    6    1   22    3    0]
 [   0    0    0    1 5995    3    0    0    0    1]
 [   0    0    1    5   10 5980    1    2    1    0]
 [   0    0    1    2    7    0 5923   52    0   15]
 [   0    8    0   22    8    0   83 5875    0    4]
 [  27    2    0    0    4    0    1    0 5963    3]
 [   2    1    0    1    7    1   68   10    6 5904]]
*******Test*******
Precision:  0.3806
Recall:  0.3806
F1 Score:  0.3806
Confusion : [[514 153   5  44  45  33   9  19  83  95]
 [302 241   1  68 180  69   1  17  33  88]
 [  5   2 563 130  30 252   3   7   4   4]
 [ 12   4  49 586 123  56  39 102  22   7]
 [  0   6   1 276 367  91  39 154  27  39]
 [  4  10  41  44 362 348  10  32  51  98]
 [  1  14 145 107   4  23 478 218   6   4]
 [ 23  11  73 327  44 100 231 168  23   0]
 [ 83  10   1   4  47 318   0   0 399 138]
 [ 98 171   0  87  31 352   9  32  78 142]]
```



In [12]:

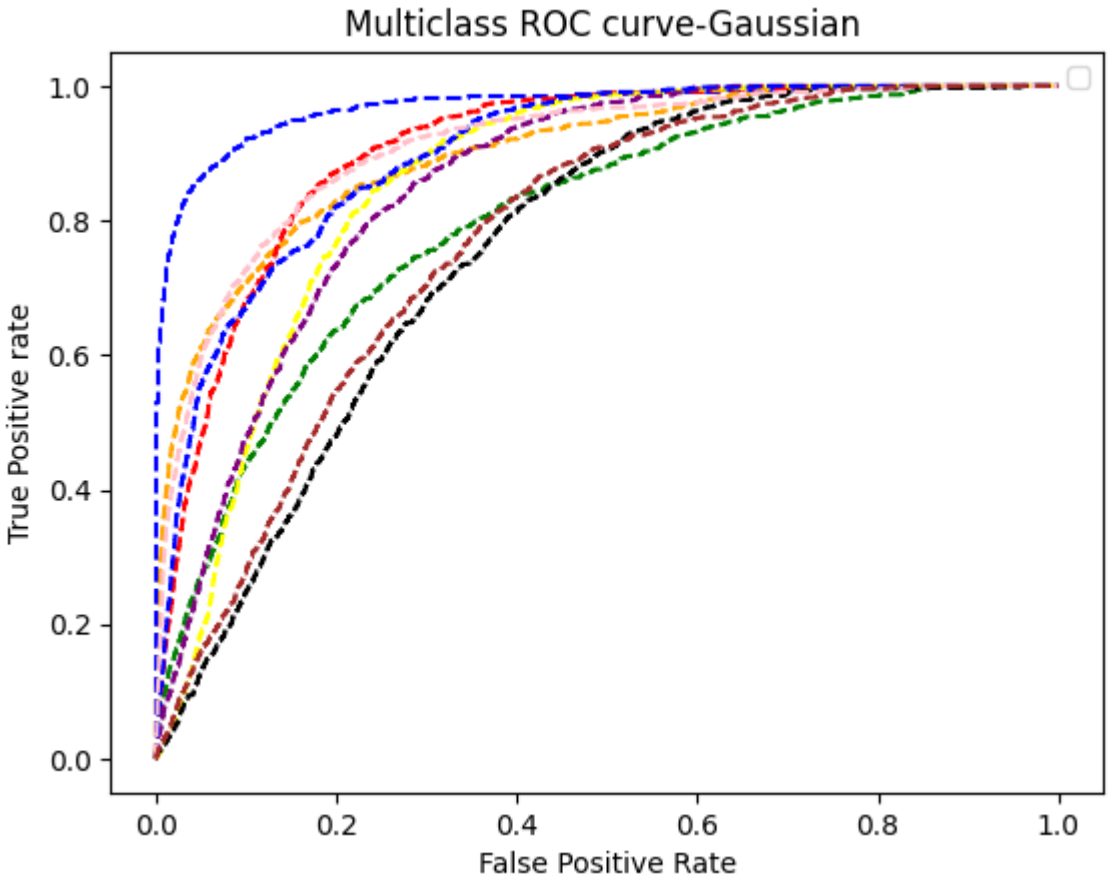
```
# performing predictions using gaussian
model= GaussianNB()
model.fit(x_train,y_train)
train_predict=model.predict(x_train)
test_predict=model.predict(x_test)
pred_prob=model.predict_proba(x_test)
fpr = {}
tpr = {}
thresh ={}

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: 0.8624150333333335
Precision:  0.90295
Recall:  0.90295
F1 Score:  0.90295
Confusion : [[5232  471   36   52    2    0    5   11  177   14]
 [ 129 5628    2   90   16    3    1    1   64   66]
 [ 104   17 5829   11    1   10    0   21    7    0]
 [ 137    6   41 5321   75   40   63  304    6    7]
 [  17    4    8   38 5563  231   10   54   46   29]
 [   2   23   46  118  147 5577    7   38   40    2]
 [   6    1   31   43   17   24 5374  490    5    9]
 [  57   13   13  211   45   13  843 4790    6    9]
 [ 301  175   21   50   35   16    2    6 5349   45]
 [  32    1   11    7  111    5   47  137  135 5514]]
*******Test*******
Precision:  0.4284
Recall:  0.4284
F1 Score:  0.4284
Confusion : [[611 183  33  13   3   5   1  17  93  41]
 [233 349  78  29  96  14   4  48  32 117]
 [  8   4 876   7   4  68   1  26   2   4]
 [ 26   2 107 396  26   4  46 363  30   0]
 [  3   3   0 129 132   7  19 502 151  54]
 [  0   1 143  26 195 173   5 125 166 166]
 [  7  15 131  15   2   0 542 273  12   3]
 [ 56  25  39  68  22   7 372 385  26   0]
 [ 68  13  43   0   1 114   1   3 650 107]
 [ 60 431   2  73  10  46  29  70 109 170]]
```


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

In [ ]: