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_20 =np.load(r"C:\Users\barathy\Downloads\kannada_mnist\Kannada_MNIST_dat
data_xtr_20 =Train_data_20['arr_0']
data1_20=data_xtr_20.reshape(-1,28*28)

#Extracting the x_test data and reshapeing it to 2 dimensional array
Test_data_20=np.load(r"C:\Users\barathy\Downloads\kannada_mnist\Kannada_MNIST_datat
data_xt_20=Test_data_20['arr_0']
data2_20=data_xt_20.reshape(-1,28*28)

#Extracting the y_train data and reshapeing it to 2 dimensional array
Train_data_20=np.load(r"C:\Users\barathy\Downloads\kannada_mnist\Kannada_MNIST_data
data_ytr_20=Train_data_20['arr_0']
data3_20=data_ytr_20.reshape(-1,1)


# #Extracting the y_test data and reshapeing it to 2 dimensional array
Test_data_20=np.load(r"C:\Users\barathy\Downloads\kannada_mnist\Kannada_MNIST_datat
data_yt_20=Test_data_20['arr_0']
data4_20 =data_yt_20.reshape(-1,1)

# Initialize and fit PCA to reduce to 10 components
n_components = 20
pca = PCA(n_components=n_components)

# Fit and transform the training data
x_train_pca = pca.fit_transform(data1_20)
x_train_p20=x_train_pca

# Fit and transform the testing data
x_test_pca=pca.fit_transform(data2_20)
x_test_p20=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
import warnings
warnings.filterwarnings('ignore')

x_train=x_train_p20
x_test=x_test_p20
y_train=data3_20
y_test=data4_20
```

In [3]:
```
# performing predictions using Random Forest classifier
model=RandomForestClassifier(n_estimators=100)
```

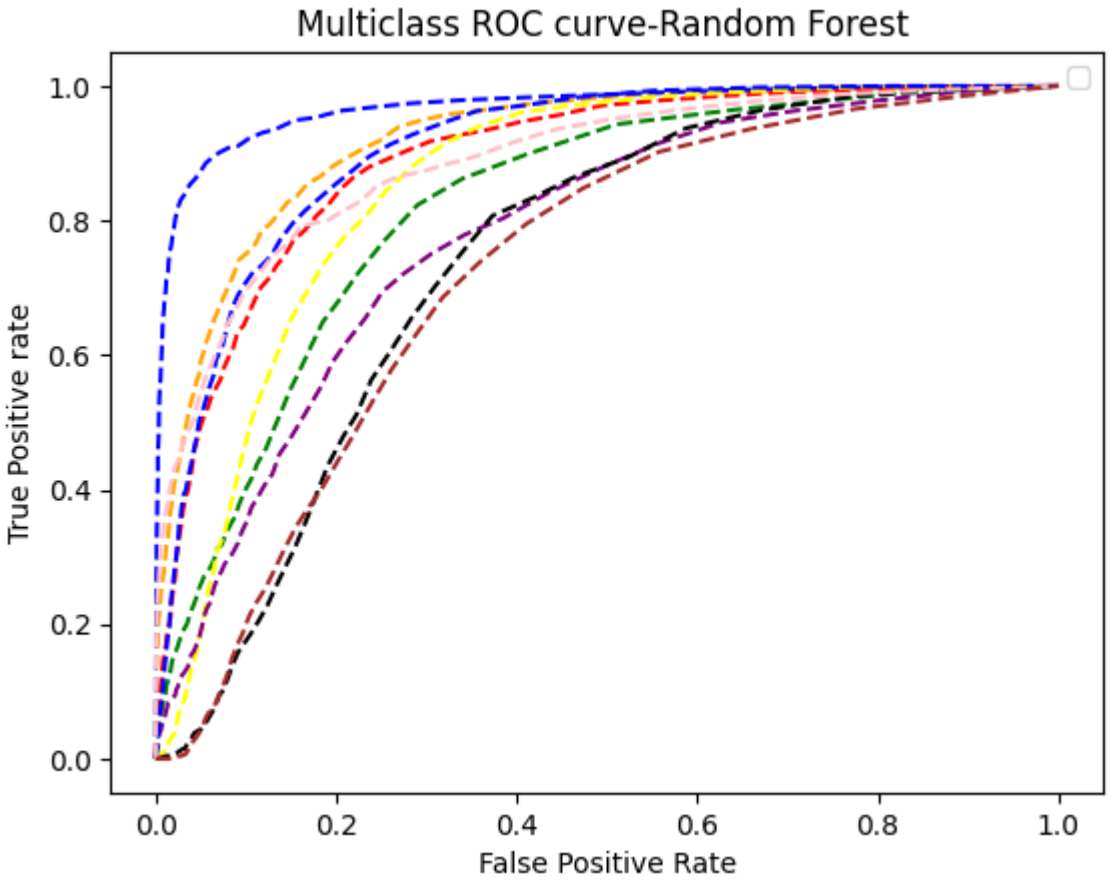
```
model.fit(x_train,y_train)
train_predict=model.predict(x_train,)
test_predict=model.predict(x_test)
pred_prob=model.predict_proba(x_test)
fpr = {}
tpr = {}
thresh ={}

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')
```

```
C:\Users\barathy\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\base.py:1151: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  return fit_method(estimator, *args, **kwargs)
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.
C:\Users\barathy\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\metrics\_classification.py:1521: UserWarning: Note that pos_label (set to 'positive') is ignored when average != 'binary' (got 'micro'). You may use labels=[pos_label] to specify a single positive class.
  warnings.warn(
C:\Users\barathy\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\metrics\_classification.py:1521: UserWarning: Note that pos_label (set to 'positive') is ignored when average != 'binary' (got 'micro'). You may use labels=[pos_label] to specify a single positive class.
  warnings.warn(
C:\Users\barathy\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\metrics\_classification.py:1521: UserWarning: Note that pos_label (set to 'positive') is ignored when average != 'binary' (got 'micro'). You may use labels=[pos_label] to specify a single positive class.
  warnings.warn(
C:\Users\barathy\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\metrics\_classification.py:1521: UserWarning: Note that pos_label (set to 'positive') is ignored when average != 'binary' (got 'micro'). You may use labels=[pos_label] to specify a single positive class.
  warnings.warn(
C:\Users\barathy\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\metrics\_classification.py:1521: UserWarning: Note that pos_label (set to 'positive') is ignored when average != 'binary' (got 'micro'). You may use labels=[pos_label] to specify a single positive class.
  warnings.warn(
C:\Users\barathy\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\metrics\_classification.py:1521: UserWarning: Note that pos_label (set to 'positive') is ignored when average != 'binary' (got 'micro'). You may use labels=[pos_label] to specify a single positive class.
  warnings.warn(
```

```
*********Randomforest************
********Train*******
roc_auc: 0.8497613166666668
Precision:  0.9999833333333333
Recall:  0.9999833333333333
F1 Score:  0.9999833333333333
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 5999    0    0    0    1    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:  0.4037
Recall:  0.4037
F1 Score:  0.4037
Confusion : [[621 139   3  42   6  23   7   1  82  76]
 [453 229   5  58  62  53   5   2  26 107]
 [  5   3 771  27  17 164   0   5   4   4]
 [ 12   2  79 654  17   7  52 128  45   4]
 [  2   0   1 244 231  37  83 158 126 118]
 [  0   0  32  30 266 283   5  33 211 140]
 [  3  26  28  47   1   7 507 321  36  24]
 [ 30  18  50 378  21  23 351  87  39   3]
 [ 37   2   0   1   8 339   0   1 511 101]
 [111 395   0  66  27  84  11  12 151 143]]
```



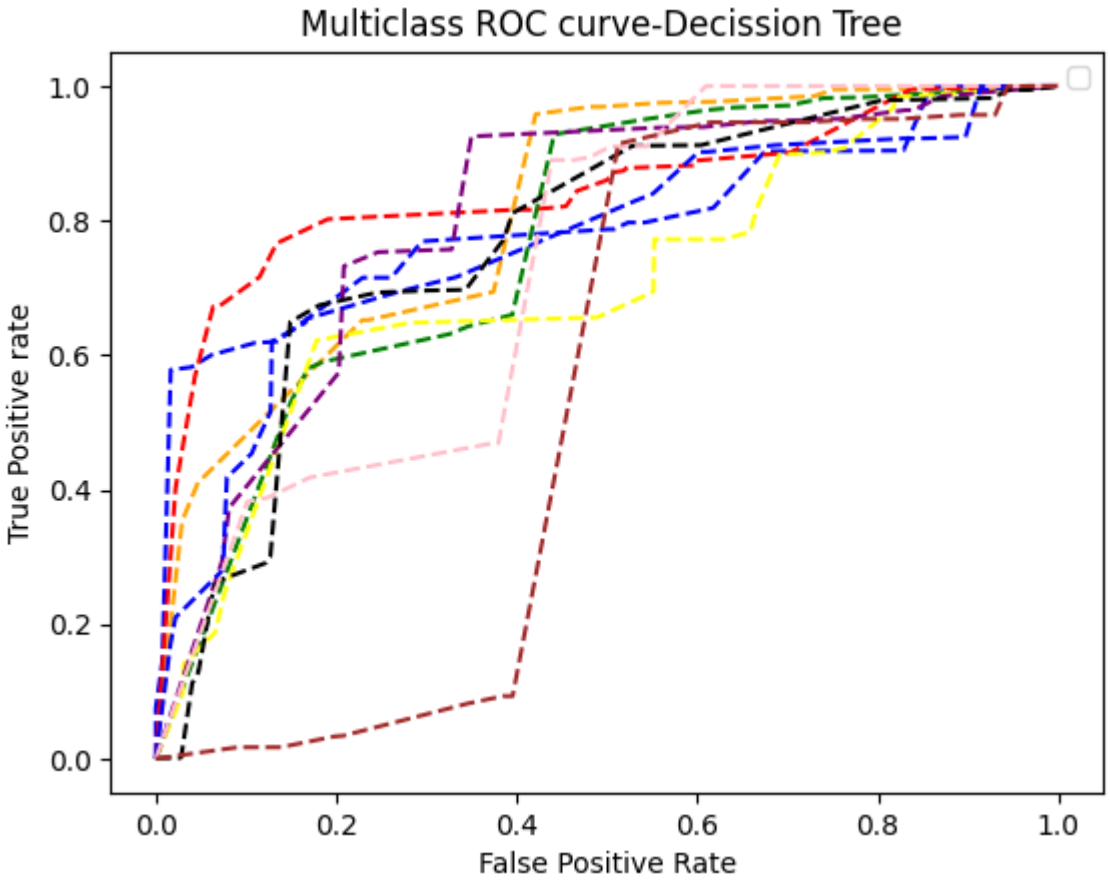
In [5]:

```
# 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: 0.7535443611111111
Precision:  0.8085666666666667
Recall:  0.8085666666666667
F1 Score:  0.8085666666666667
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  105  259   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 4925  617   20   18]
 [  38   64   48  337  109   13  470 4784   78   59]
 [ 214  278  511   12   11  104    3    5 4848   14]
 [  98  256  167   56  120  205   69   59  282 4688]]
*******Test*******
Precision:  0.3618
Recall:  0.3618
F1 Score:  0.3618
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  67 167  94   4]
 [  4   8   7  85 141  85 181 114 375   0]
 [  1  13 161  30 168 373  22  18 201  13]
 [ 74 113  76  50   0  40 358 268  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]]
```

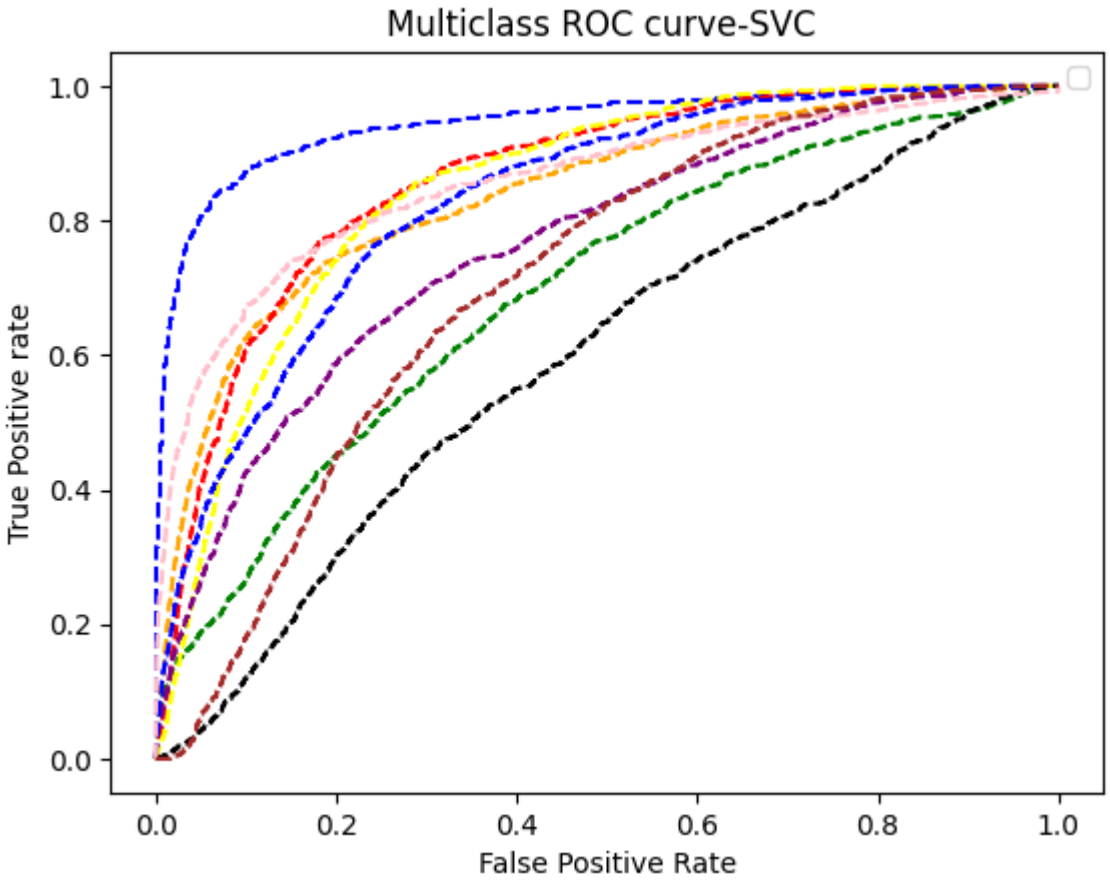


In [6]:
```
# performing predictions using svc
model=SVC(probability=True)
model.fit(x_train,y_train)
train_predict=model.predict(x_train,)
test_predict=model.predict(x_test)
pred_prob=model.predict_proba(x_test)
fpr = {}
tpr = {}
thresh ={}

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: 0.7946393777777778
Precision:  0.98695
Recall:  0.98695
F1 Score:  0.98695
Confusion : [[5862  113    0    9    2    0    3    0    9    2]
 [   4 5972    0    7    2    0    1    3    1   10]
 [  20    2 5968    8    0    1    0    0    0    1]
 [  15    3    3 5914   16   13    1   32    2    1]
 [   0    1    0   15 5952   21    1    2    4    4]
 [   0    0    0   10   25 5961    1    1    2    0]
 [   0    1    1   11    5    1 5898   71    0   12]
 [   3    5    0   27    6    2  128 5823    1    5]
 [  11    1    0    0    3    3    0    1 5979    2]
 [   3    2    0    5   21    3   65   10    3 5888]]
*******Test*******
Precision:  0.3886
Recall:  0.3886
F1 Score:  0.3886
Confusion : [[549 112   6  78   7  14  21  22 107  84]
 [440 154  23 101  61  39  54  36  42  50]
 [  3   1 778  37  22 136   7  11   2   3]
 [ 16   3  75 625  27  19  53 144  24  14]
 [  1   3   4 197 315   8 129 265  45  33]
 [  2   1  65   8 257 311  20  70  91 175]
 [ 30  32 133  71   2   8 387 290  38   9]
 [ 21  12  52 495  26  29 210 134  21   0]
 [ 50   3  13   3  58 222   4   5 531 111]
 [224 170   0  93  82 122   8 156  43 102]]
```



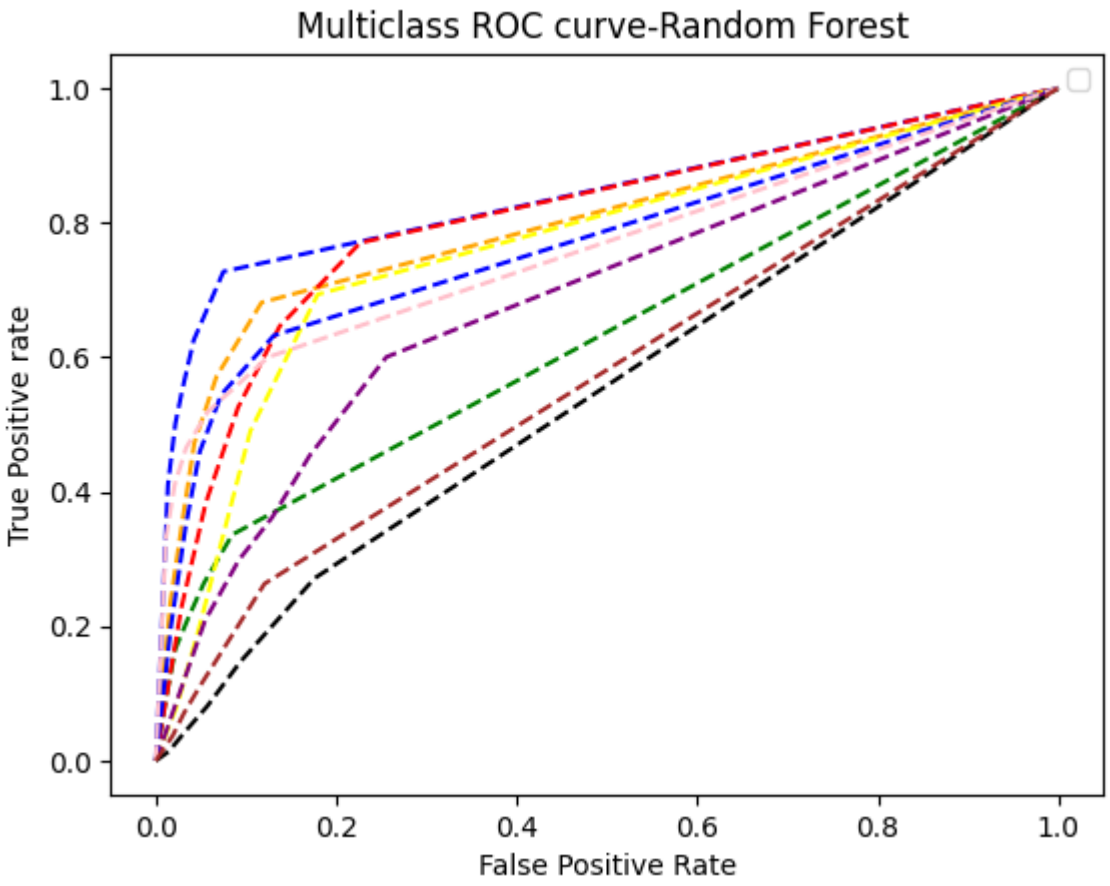
In [7]:

```
# 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-Random Forest ')
```

```
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: 0.71630105
Precision:  0.9870333333333333
Recall:  0.9870333333333333
F1 Score:  0.9870333333333333
Confusion : [[5816  164    1    6    2    0    2    1    7    1]
 [  10 5981    0    1    2    0    1    0    0    5]
 [  22    5 5963    6    0    2    1    1    0    0]
 [  18    5    2 5917   16   10    1   30    1    0]
 [   0    0    0    5 5979   12    0    1    0    3]
 [   0    0    1    5   17 5974    1    0    2    0]
 [   0    0    1    7    9    0 5894   74    0   15]
 [   3    6    0   26    8    2  105 5846    1    3]
 [  19    2    0    1    8    2    2    0 5965    1]
 [   4    0    0    0   23    1   71   11    3 5887]]
*******Test*******
Precision:  0.3854
Recall:  0.3854
F1 Score:  0.3854
Confusion : [[511 148   2  51  20  36  14  21  89 108]
 [336 231   7 107 135  73   0  12  25  74]
 [  4   1 579  68  15 319   5   2   2   5]
 [ 15   9  49 592  71  34  54 138  26  12]
 [  2   2   0 250 395  74 111 104  35  27]
 [  0   8  39  21 359 386  18  12  37 120]
 [  6  15 159  87   5  20 481 210   7  10]
 [ 18   5  59 380  38  91 281  89  36   3]
 [ 48   3   1   5  74 282   1   5 472 109]
 [ 95 202   0  87  48 323   9  42  76 118]]
```



In [8]:

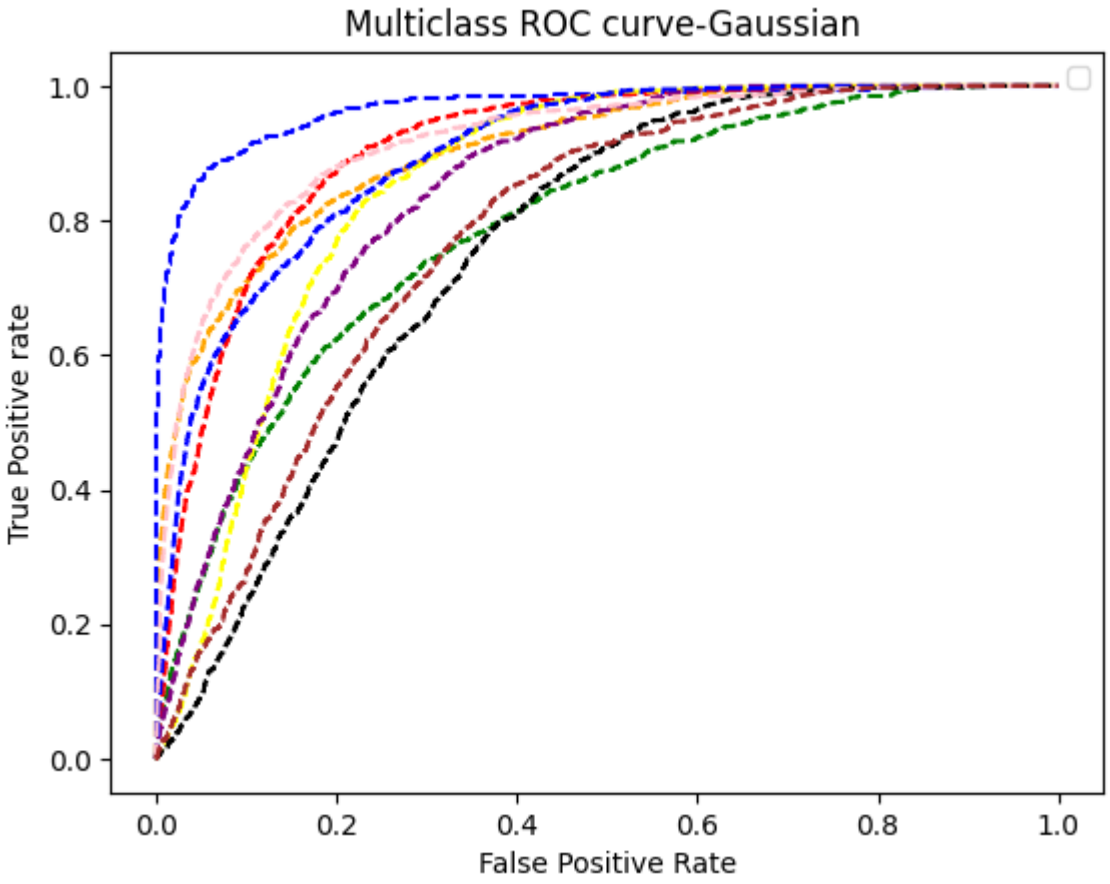
```
# performing predictions using gaussian
model= GaussianNB()
model.fit(x_train,y_train)
train_predict=model.predict(x_train)
test_predict=model.predict(x_test)
pred_prob=model.predict_proba(x_test)
fpr = {}
tpr = {}
thresh ={}

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: 0.8618007333333333
Precision:  0.8945833333333333
Recall:  0.8945833333333333
F1 Score:  0.8945833333333333
Confusion : [[5164  485   34   68    3    0    5   14  210   17]
 [ 168 5511    3  130   18    2    1    5   91   71]
 [ 132   15 5805   17    1    7    0   12   10    1]
 [ 145    7   37 5267   71   41   65  358    4    5]
 [  15    2   12   50 5585  192    6   70   39   29]
 [   1   25   59  145  174 5512    9   29   43    3]
 [  12    1   25   50   17   37 5360  485    5    8]
 [  57   14   16  233   50   16  885 4722    2    5]
 [ 295  170   21   51   40   17    6   15 5317   68]
 [  76    2   10    7  116    5   39  149  164 5432]]
*******Test*******
Precision:  0.4327
Recall:  0.4327
F1 Score:  0.4326999999999999
Confusion : [[604 183  32  12   4   5  12  14  94  40]
 [247 364  70  22  86  16   8  35  38 114]
 [ 10   3 865   7   5  85   1  22   1   1]
 [ 20   2 112 385  26   3  49 371  32   0]
 [  2   1   0 108 132   3  25 526 154  49]
 [  0   1 137  16 193 194   3 149 164 143]
 [  9  18 133  16   1   1 542 268  11   1]
 [ 67  28  32  71  21  11 353 387  30   0]
 [ 53  12  36   0   1 109   0   0 689 100]
 [ 53 437   0  99   8  46  37  45 110 165]]
```


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

In [ ]:

In [ ]: