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_15 =np.load(r"C:\Users\barathy\Downloads\kannada_mnist\Kannada_MNIST_dat
data_xtr_15 =Train_data_15['arr_0']
data1_15=data_xtr_15.reshape(-1,28*28)

#Extracting the x_test data and reshapeing it to 2 dimensional array
Test_data_15=np.load(r"C:\Users\barathy\Downloads\kannada_mnist\Kannada_MNIST_datat
data_xt_15=Test_data_15['arr_0']
data2_15=data_xt_15.reshape(-1,28*28)

#Extracting the y_train data and reshapeing it to 2 dimensional array
Train_data_15=np.load(r"C:\Users\barathy\Downloads\kannada_mnist\Kannada_MNIST_data
data_ytr_15=Train_data_15['arr_0']
data3_15=data_ytr_15.reshape(-1,1)


# #Extracting the y_test data and reshapeing it to 2 dimensional array
Test_data_15=np.load(r"C:\Users\barathy\Downloads\kannada_mnist\Kannada_MNIST_datat
data_yt_15=Test_data_15['arr_0']
data4_15 =data_yt_15.reshape(-1,1)

# Initialize and fit PCA to reduce to 10 components
n_components = 15
pca = PCA(n_components=n_components)

# Fit and transform the training data
x_train_pca = pca.fit_transform(data1_15)
x_train_p15=x_train_pca

# Fit and transform the testing data
x_test_pca=pca.fit_transform(data2_15)
x_test_p15=x_test_pca
```

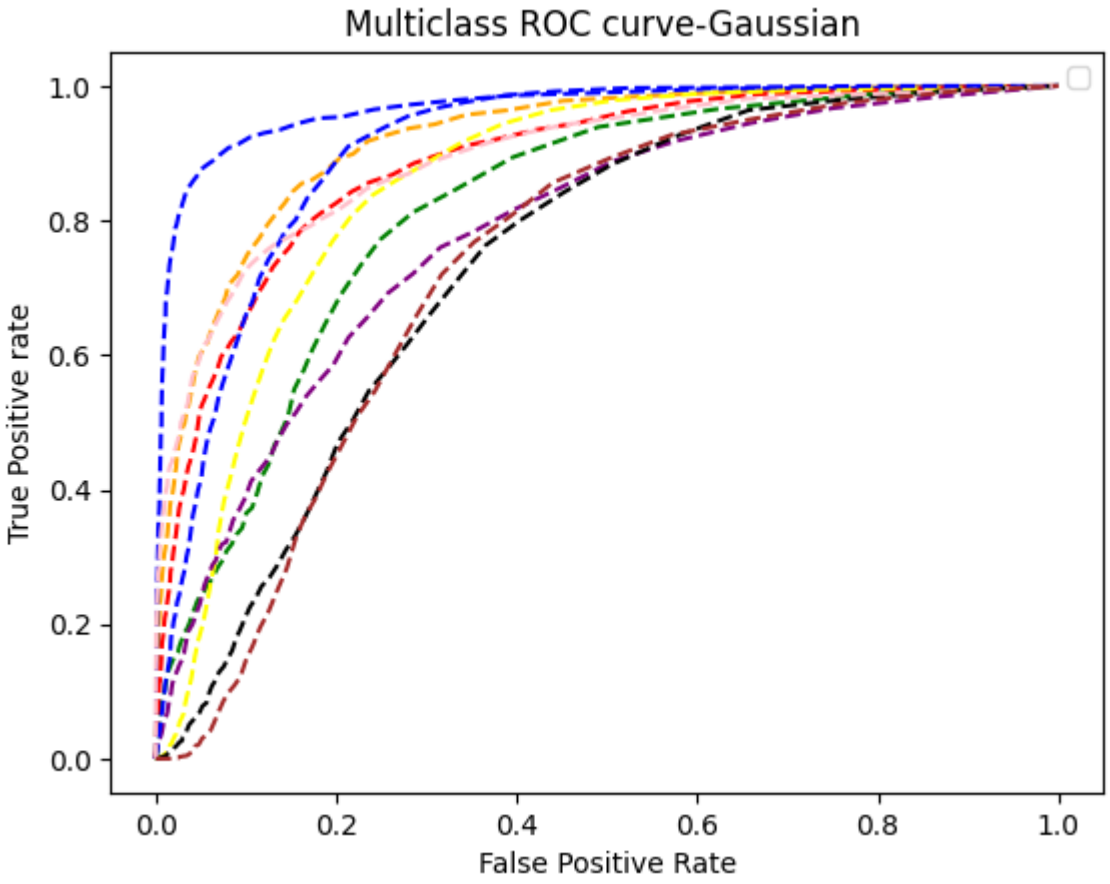
In [5]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import precision_score,recall_score,f1_score,confusion_matrix,
import sklearn.metrics as metrics
import matplotlib.pyplot as plt

x_train=x_train_p15
x_test=x_test_p15
y_train=data3_15
y_test=data4_15
```

In [4]:
```
# 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')
```

```
C:\Users\barathy\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\b
ase.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\m
etrics\_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\m
etrics\_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\m
etrics\_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\m
etrics\_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\m
etrics\_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\m
etrics\_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*******
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:  0.4005
Recall:  0.4005
F1 Score:  0.4005
Confusion : [[618 140   9  21   5  19  19   2  83  84]
 [432 228   5  56  70  43  10   0  31 125]
 [  7   5 778  32  17 149   1   3   2   6]
 [ 14   1  85 620  15   9  74 118  57   7]
 [  0   2   0 191 218  33 113 173 135 135]
 [  0   1  21  29 288 298   7  32 199 125]
 [  3  18  35  45   2   9 422 367  38  61]
 [ 30  20  57 341  24  15 332 121  58   2]
 [ 28   3   0   2  10 289   0   0 570  98]
 [ 82 476   0  49  15  88  14   9 135 132]]
```



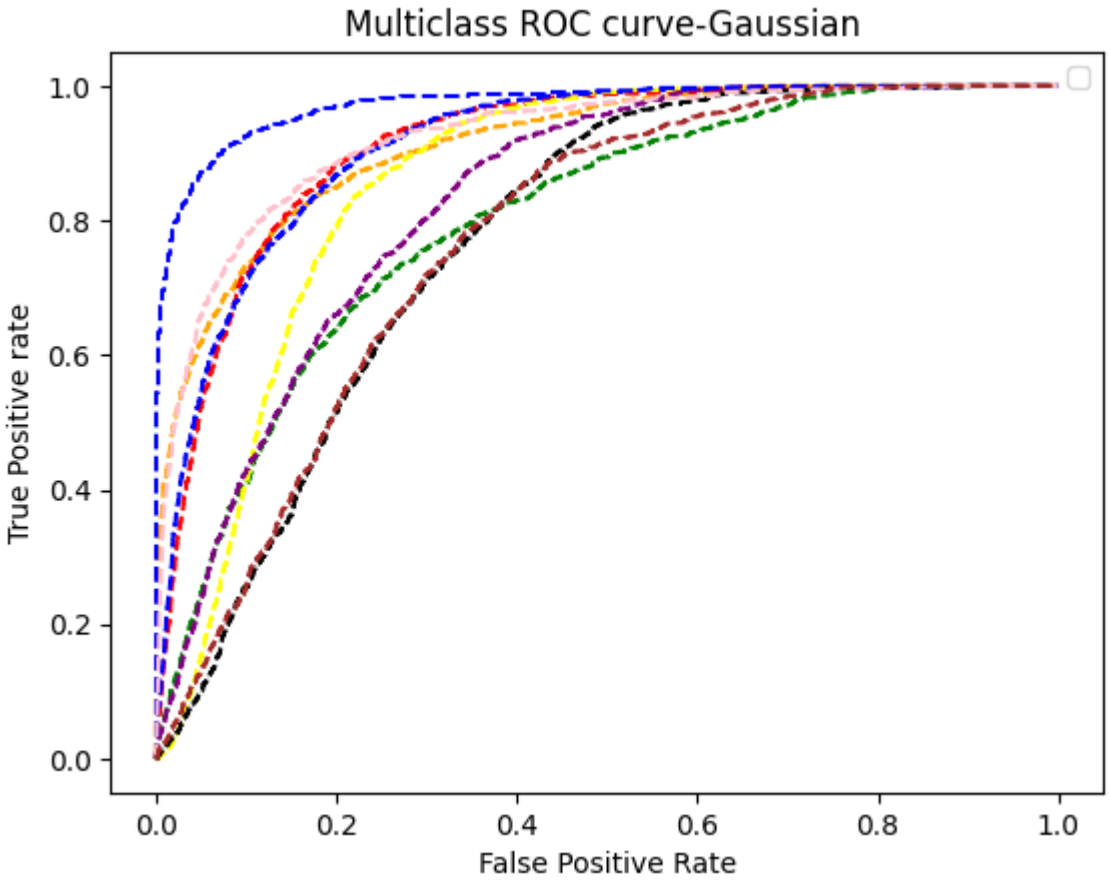
In [5]:

```
# 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 ')
```

```python
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')
```

```
C:\Users\barathy\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\u
tils\validation.py:1184: DataConversionWarning: A column-vector y was passed when a
1d array was expected. Please change the shape of y to (n_samples, ), for example us
ing ravel().
  y = column_or_1d(y, warn=True)
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\m
etrics\_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\m
etrics\_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\m
etrics\_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\m
etrics\_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\m
etrics\_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\m
etrics\_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(
```

```
******GaussianNB********
********Train*******
roc_auc: 0.8667841444444443
Precision:  0.88485
Recall:  0.88485
F1 Score:  0.88485
Confusion : [[5092  545   39   71    3    0    5   17  208   20]
 [ 178 5447    7  155   26    1    0    3  109   74]
 [ 109   19 5816   17    1   15    0   20    2    1]
 [ 138    6   36 5274   72   46   92  327    2    7]
 [  14    3    9   42 5575  214    2   69   41   31]
 [   1   24   66  140  178 5524    7   24   34    2]
 [  13    0   44   54   21   27 5253  579    4    5]
 [  62   13   23  231   57   11 1135 4459    3    6]
 [ 390  131   19   82   43   13    4   15 5236   67]
 [  87    2   15   14  137    3   36  137  154 5415]]
*******Test*******
Precision:  0.4287
Recall:  0.4287
F1 Score:  0.4287
Confusion : [[618 193  26   5   4   6  17   8  84  39]
 [264 374  36  24  81  16   8  32  32 133]
 [ 11   1 875   7   4  84   2  14   1   1]
 [ 20   4 104 389  21   1  50 375  36   0]
 [  2   1   0 129 111   3  35 514 159  46]
 [  0   0 149  14 235 157   3 137 151 154]
 [  4  19 155  18   1   0 547 240  12   4]
 [ 93  26  32  60  21   8 357 374  29   0]
 [ 52  11  33   0   1 102   0   0 705  96]
 [ 39 499   0  76   6  51  45  49  98 137]]
```



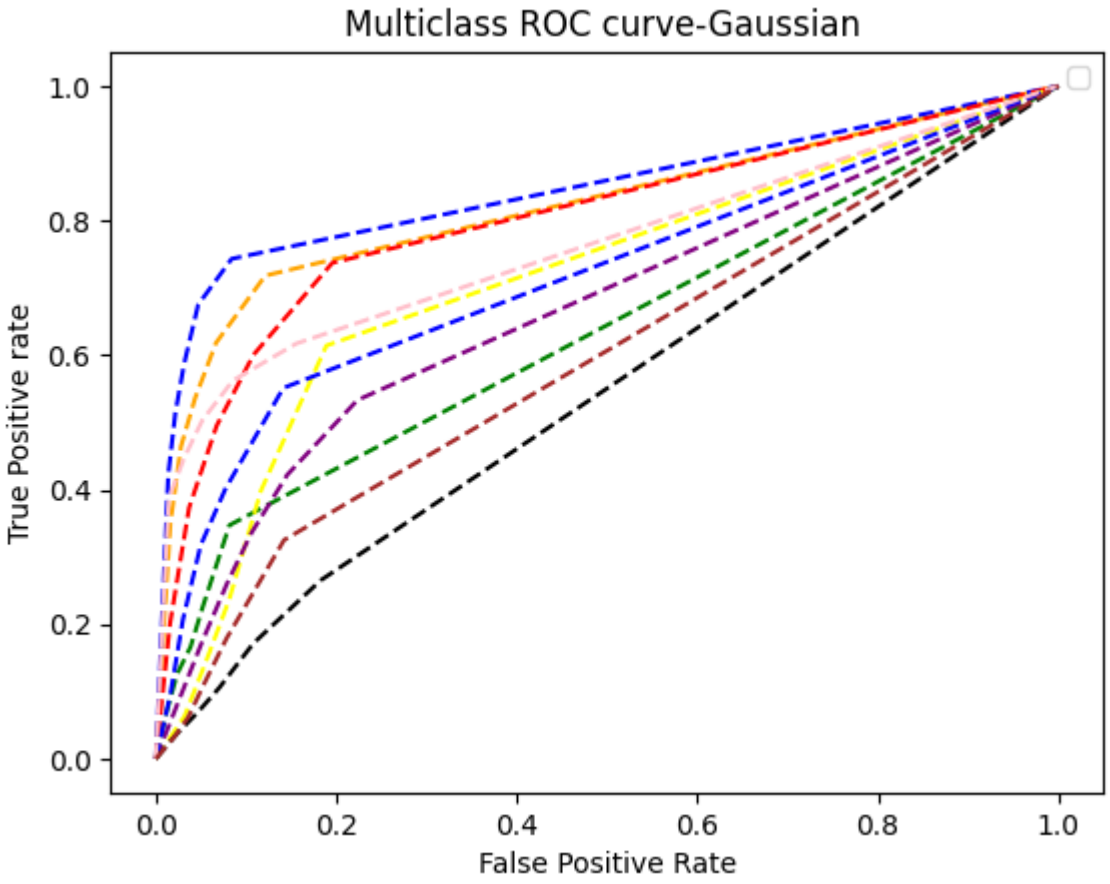
In [6]:

```
# 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')
```

```
C:\Users\barathy\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\n
eighbors\_classification.py:228: DataConversionWarning: A column-vector y was passed
when a 1d array was expected. Please change the shape of y to (n_samples,), for exam
ple using ravel().
  return self._fit(X, y)
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\m
etrics\_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\m
etrics\_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\m
etrics\_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\m
etrics\_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\m
etrics\_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\m
etrics\_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(
```

```
******KNeighborsClassifier********
********Train*******
roc_auc: 0.7085538055555555
Precision:  0.9838666666666667
Recall:  0.9838666666666667
F1 Score:  0.9838666666666667
Confusion : [[5792  174    2    8    4    0    1    1   14    4]
 [  14 5972    0    1    2    0    2    1    2    6]
 [  24    4 5957    7    0    5    0    1    0    2]
 [  16    8    5 5891   20   19    0   35    4    2]
 [   2    1    0    8 5957   25    1    1    0    5]
 [   0    0    0    9   21 5965    1    1    2    1]
 [   1    1    1   10    6    2 5873   88    0   18]
 [   3    6    0   34   10    1  118 5822    0    6]
 [  38    2    1    1    4    5    2    0 5945    2]
 [   4    0    0    4   33    1   76   20    4 5858]]
*******Test*******
Precision:  0.3723
Recall:  0.3723
F1 Score:  0.3723
Confusion : [[573 114  10  27  22  30   8   6  96 114]
 [352 181  13  62 199  65   6   8  34  80]
 [  5   1 637  79   6 249   4  10   1   8]
 [ 19   5  82 551  79  24  61 128  40  11]
 [  5   0   0 187 304  56 167 140  72  69]
 [  1   0  23  27 394 370  12  25  31 117]
 [  4  12 155  86   3  16 357 347  13   7]
 [ 34   7  76 302  35  45 262 118 112   9]
 [ 27   2   0   3  81 276   1   1 511  98]
 [ 62 273   0  37  45 278  16  23 145 121]]
```



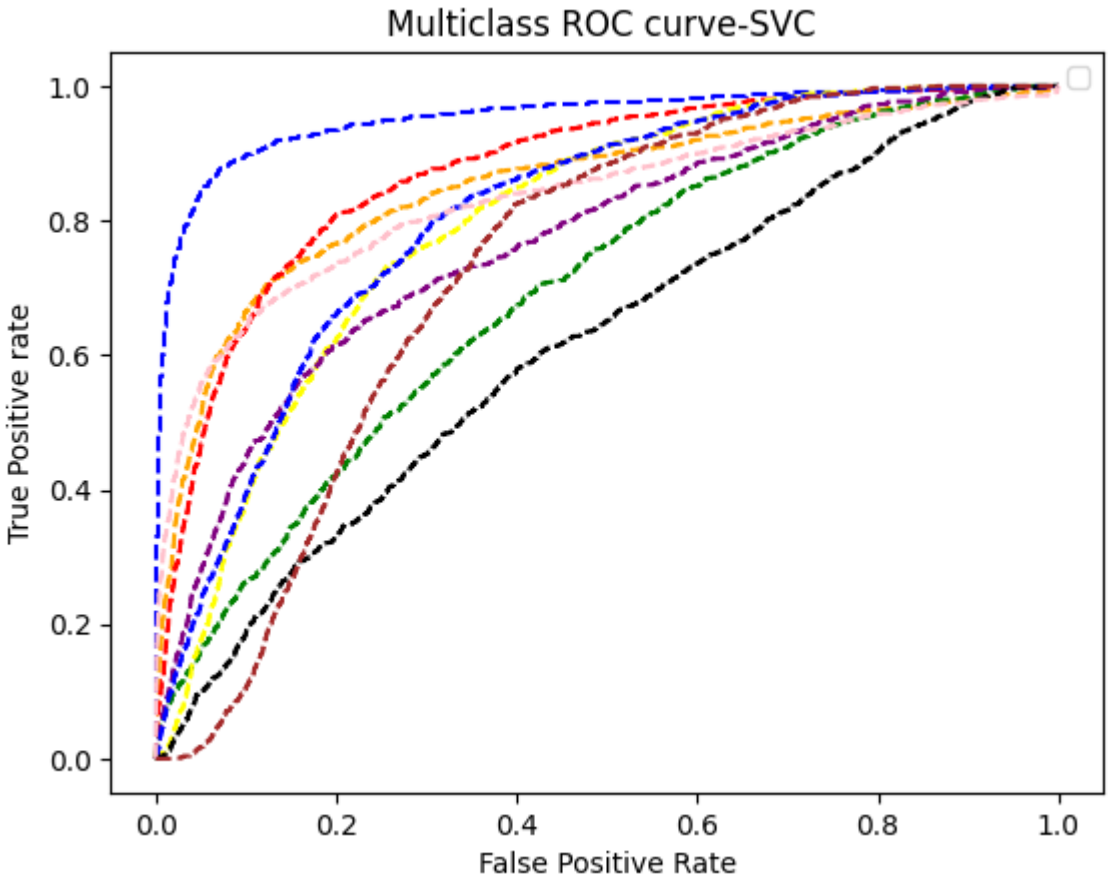
In [6]:

```
# 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')
```

```
C:\Users\barathy\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\u
tils\validation.py:1184: DataConversionWarning: A column-vector y was passed when a
1d array was expected. Please change the shape of y to (n_samples, ), for example us
ing ravel().
  y = column_or_1d(y, warn=True)
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\m
etrics\_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\m
etrics\_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\m
etrics\_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\m
etrics\_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\m
etrics\_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\m
etrics\_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(
```

```
******SVC********
********Train*******
roc_auc: 0.7921095333333332
Precision:  0.98085
Recall:  0.98085
F1 Score:  0.98085
Confusion : [[5802  152    0   12    3    0    1    1   25    4]
 [  24 5946    0    8    3    0    1    2    5   11]
 [  27    1 5957    9    0    3    0    1    0    2]
 [  17    6    4 5874   24   22    2   46    3    2]
 [   0    0    1   20 5929   35    2    4    5    4]
 [   0    1    0   12   43 5936    1    2    5    0]
 [   0    2    1   17    8    2 5856   99    1   14]
 [   6    6    0   38    7    2  157 5773    1   10]
 [  28    2    0    4    9    3    1    1 5946    6]
 [   5    2    0    8   37    3   72   31   10 5832]]
*******Test*******
Precision:  0.381
Recall:  0.381
F1 Score:  0.381
Confusion : [[574  97   8  32   6  12  37   6  94 134]
 [440 147  12  64  65  35  67  18  45 107]
 [  4   3 792  34  10 133  12   7   2   3]
 [ 19   1  80 622  26   8  52 130  39  23]
 [  1   0   0 178 215  19 163 188  51 185]
 [  2   0  34   9 286 333  19  54  60 203]
 [ 25  26  95  59   2   9 290 425  53  16]
 [ 26   8  63 426  23  24 187 186  54   3]
 [ 45   1   9   2  45 233   1   4 543 117]
 [138 343   0  67  52 115  12  95  70 108]]
```

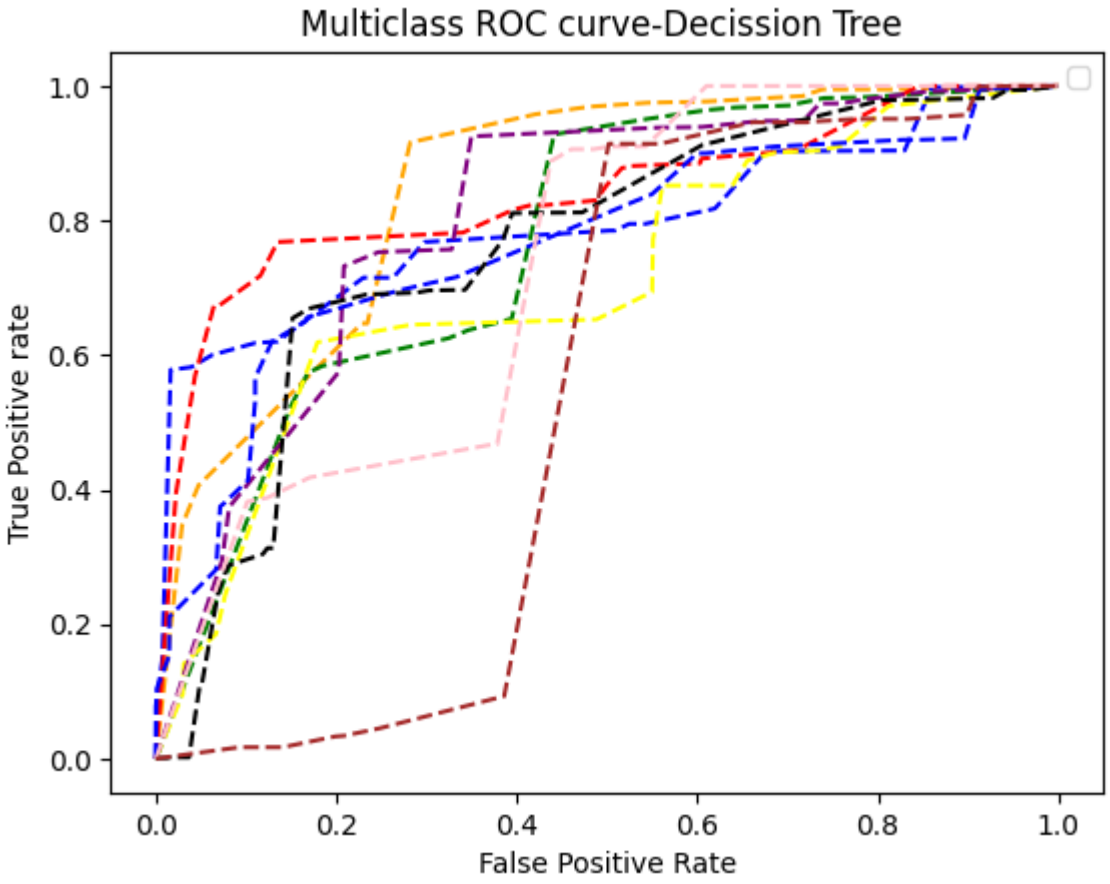


In [7]:
```
# performing predictions using Decision tree
model=DecisionTreeClassifier(max_depth=5)
model.fit(x_train,y_train)
train_predict=model.predict(x_train,)
test_predict=model.predict(x_test)
pred_prob=model.predict_proba(x_test)
fpr = {}
tpr = {}
thresh ={}

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 ')
```

```python
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.
C:\Users\barathy\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\m
etrics\_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\m
etrics\_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\m
etrics\_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\m
etrics\_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\m
etrics\_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\m
etrics\_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(
```

```
******Decision tree********
********Train*******
roc_auc: 0.7574038833333333
Precision:  0.8059833333333334
Recall:  0.8059833333333334
F1 Score:  0.8059833333333334
Confusion : [[4470 1078    4   20   10    9   10    4  219  176]
 [ 408 5252   30   13    2   14    1    1  214   65]
 [ 541   83 5012   26   10  250    1   52   12   13]
 [ 244  307   40 4731  188   55   78  286   35   36]
 [   4  133    3  282 4902  276    9    8  364   19]
 [  44  198  181  100  155 4881    2   23  411    5]
 [  11  119   42  124    4  117 4836  706   20   21]
 [  37   62   48  337  109   13  537 4717   78   62]
 [ 215  277  511   12   11  104    3    5 4848   14]
 [  99  233  167   56  120  205   72   56  282 4710]]
*******Test*******
Precision:  0.3615
Recall:  0.3615
F1 Score:  0.3615
Confusion : [[407 240  10   9   0   8  12   3  42 269]
 [ 78 523  28  24   1  23   3   5  41 274]
 [ 36  17 582 138  94 105   1   4   1  22]
 [ 45  14  23 569   5  12  61 173  94   4]
 [  4   8   7  85 141  85 145 150 375   0]
 [  1  13 160  30 168 374   6  34 201  13]
 [ 76 112  75  50   0  40 375 251  10  11]
 [ 90  69  43  38  10   3 398 248 100   1]
 [ 52  49   6   0   0 421   0   0 382  90]
 [ 46 821   0  17   1  35  16   2  48  14]]
```


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

In [ ]:

In [ ]: