In [90]:

```python
import os
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import preprocessing,svm
from sklearn.model_selection import train_test_split
from sklearn.model_selection import StratifiedKFold , cross_val_score,KFold
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score,f1_score,confusion_matrix
from sklearn.neighbors import KNeighborsClassifier


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

In [91]:

```python
directory = r'C:\Users\injam\Desktop\DM_Assignment_1\Cropped'
```

In [92]:

```python
from skimage import filters
from skimage import data, exposure, img_as_float
from skimage.color import rgb2gray
from skimage.io import imread
import numpy as np

def angle(dx, dy):
 """Calculate the angles between horizontal and vertical operators."""
 return np.mod(np.arctan2(dy, dx), np.pi)
hist=[]
label=[]
for index,name in enumerate(os.listdir(directory)):
    for image in os.listdir(os.path.join(directory,name)):
        img = imread(os.path.join(directory,name,image.strip()))
        gray_img = rgb2gray(img)
        angle_sobel = angle(filters.sobel_h(gray_img),filters.sobel_v(gray_img))
        Hist,_=exposure.histogram(angle_sobel, nbins=36)
        hist.append(Hist)
        label.append(index)
```

In [93]:

```python
X = np.array(hist)
Y = np.array(label)
```

In [94]:

```python
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, stratify=Y,rand
```

In [95]:

```python
scaler = preprocessing.StandardScaler().fit(X_train)
```

In [96]:

```python
X_scaled = scaler.transform(X_train)
```

In [97]:

```python
X_scaled.mean(axis=0)
```

Out[97]:

```
array([-6.98894494e-17,  2.45341088e-16, -2.29324756e-16,  4.51551365e-16,
       -5.78316994e-16, -6.12988713e-16,  8.69068023e-17, -4.06778436e-16,
        4.48184295e-17,  2.52985247e-16, -1.11386310e-16,  4.03047359e-16,
       -4.96142289e-16,  4.36809059e-16, -1.53611186e-16,  4.16424636e-16,
        2.61357420e-16,  6.55213588e-18,  2.61357420e-16, -1.79455722e-16,
        3.90580100e-16,  1.19758484e-16, -3.05766341e-17,  4.25888832e-16,
        5.74403912e-16, -3.99680289e-16, -2.32964831e-16,  3.01898761e-17,
        3.44214639e-16,  1.16027406e-16, -4.78487923e-16,  5.59843610e-16,
        1.68171488e-16, -4.68841723e-16, -1.45603020e-17, -2.67181541e-16])
```

In [98]:

```python
X_scaled.std(axis=0)
```

Out[98]:

```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1.])
```

In [99]:

```
X_test_scaled = scaler.transform(X_test)
```

In [100…

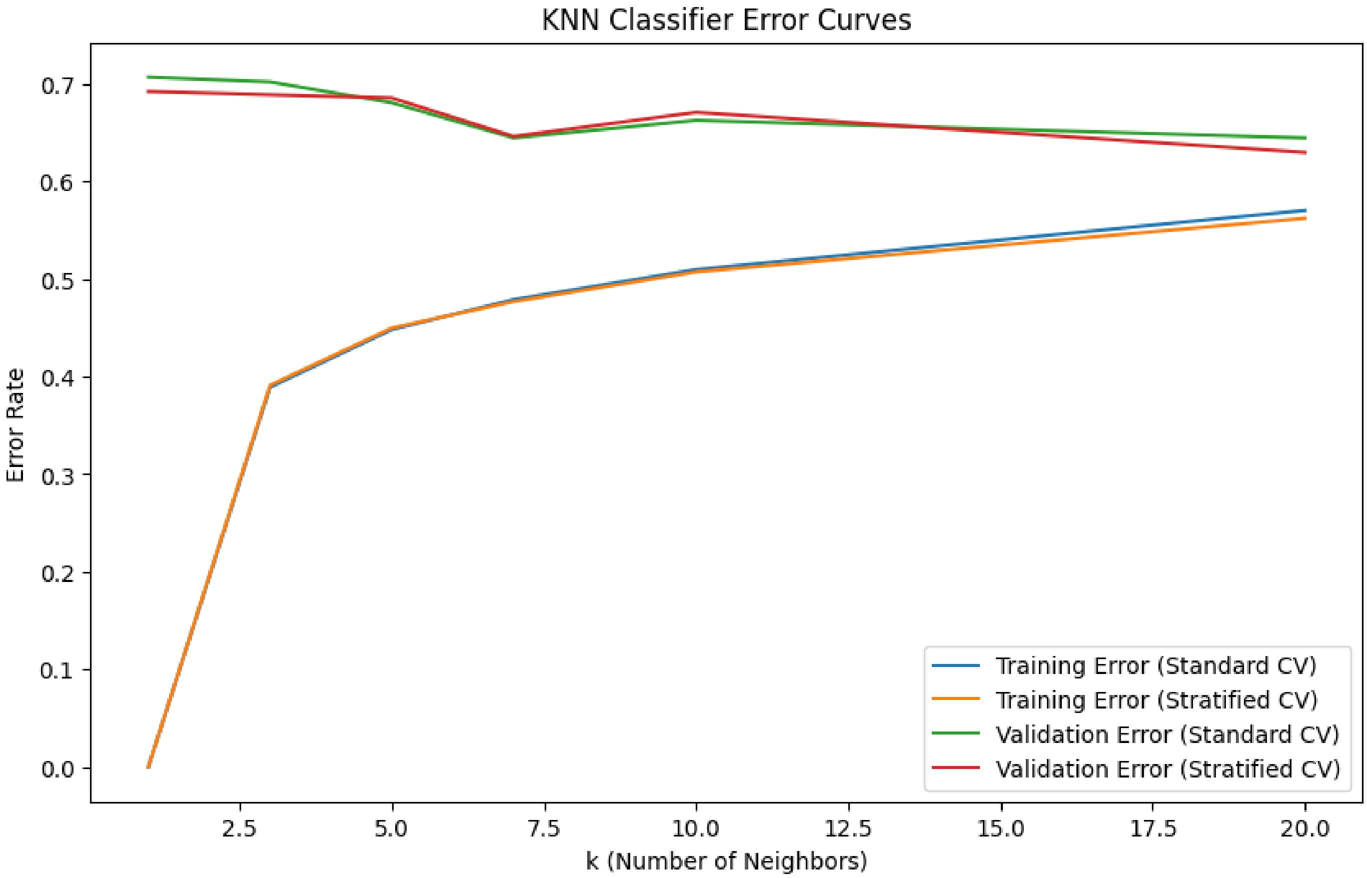
```
#standard 5 fold cross validation
train_errors=[]
val_errors=[]
kf = KFold(n_splits=5, shuffle=True, random_state=42)
k_val=[1,3,5,7,10,20]
for k in k_val:
    knn = KNeighborsClassifier(n_neighbors=k)
    train = []
    val= []
    for train_idx, val_idx in kf.split(X_scaled):
        x_train, x_val = X_scaled[train_idx], X_scaled[val_idx]
        Y_train, Y_val = y_train[train_idx], y_train[val_idx]
        knn.fit(x_train, Y_train)
        train_pred = knn.predict(x_train)
        val_pred = knn.predict(x_val)
        train_accuracy = accuracy_score(Y_train, train_pred)
        val_accuracy = accuracy_score(Y_val, val_pred)
        train.append(1 - train_accuracy)
        val.append(1 - val_accuracy)
    train_errors.append(np.mean(train))
    val_errors.append(np.mean(val))
```

In [101…

```
#stratified 5 fold cross validation
stratified_train_e=[]
stratified_val_e=[]
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

for k in k_val:
    knn = KNeighborsClassifier(n_neighbors=k)
    train = []
    val= []
    for train_idx, val_idx in skf.split(X_scaled,y_train):
        x_train, x_val = X_scaled[train_idx], X_scaled[val_idx]
        Y_train, Y_val = y_train[train_idx], y_train[val_idx]
        knn.fit(x_train, Y_train)
        train_pred = knn.predict(x_train)
        val_pred = knn.predict(x_val)
        train_accuracy = accuracy_score(Y_train, train_pred)
        val_accuracy = accuracy_score(Y_val, val_pred)
        train.append(1 - train_accuracy)
        val.append(1 - val_accuracy)
    stratified_train_e.append(np.mean(train))
    stratified_val_e.append(np.mean(val))
```

In [102…

```
plt.figure(figsize=(10, 6))
plt.plot(k_val, train_errors, label='Training Error (Standard CV)')
plt.plot(k_val, stratified_train_e, label='Training Error (Stratified CV)')
plt.plot(k_val, val_errors, label='Validation Error (Standard CV)')
plt.plot(k_val, stratified_val_e, label='Validation Error (Stratified CV)')

plt.title('KNN Classifier Error Curves')
plt.xlabel('k (Number of Neighbors)')
plt.ylabel('Error Rate')
plt.legend()

plt.show()
```



Which k has the lowest mean error for each curve? Standard Training error k=1 stratified Training error k=1 standard validation error k=7 Stratified validation error k=3

Comment about (1) the model complexity for k-Nearest Neighbor classifier in relation to k : Less - k=1 intermediate - for k= 3,5,7,10 high - k= 20

when/whether there is overfitting/underfitting. : overfits at k=1 having less training error and more validation error underfits at k=20 with more number of neighbours

In [103…

```
# k=3
Model = KNeighborsClassifier(n_neighbors = 3)
Model.fit(X_scaled, y_train)
p = Model.predict(X_test_scaled)
print("Test error when k=3 :" + str(1-(accuracy_score(y_test,p))))
```

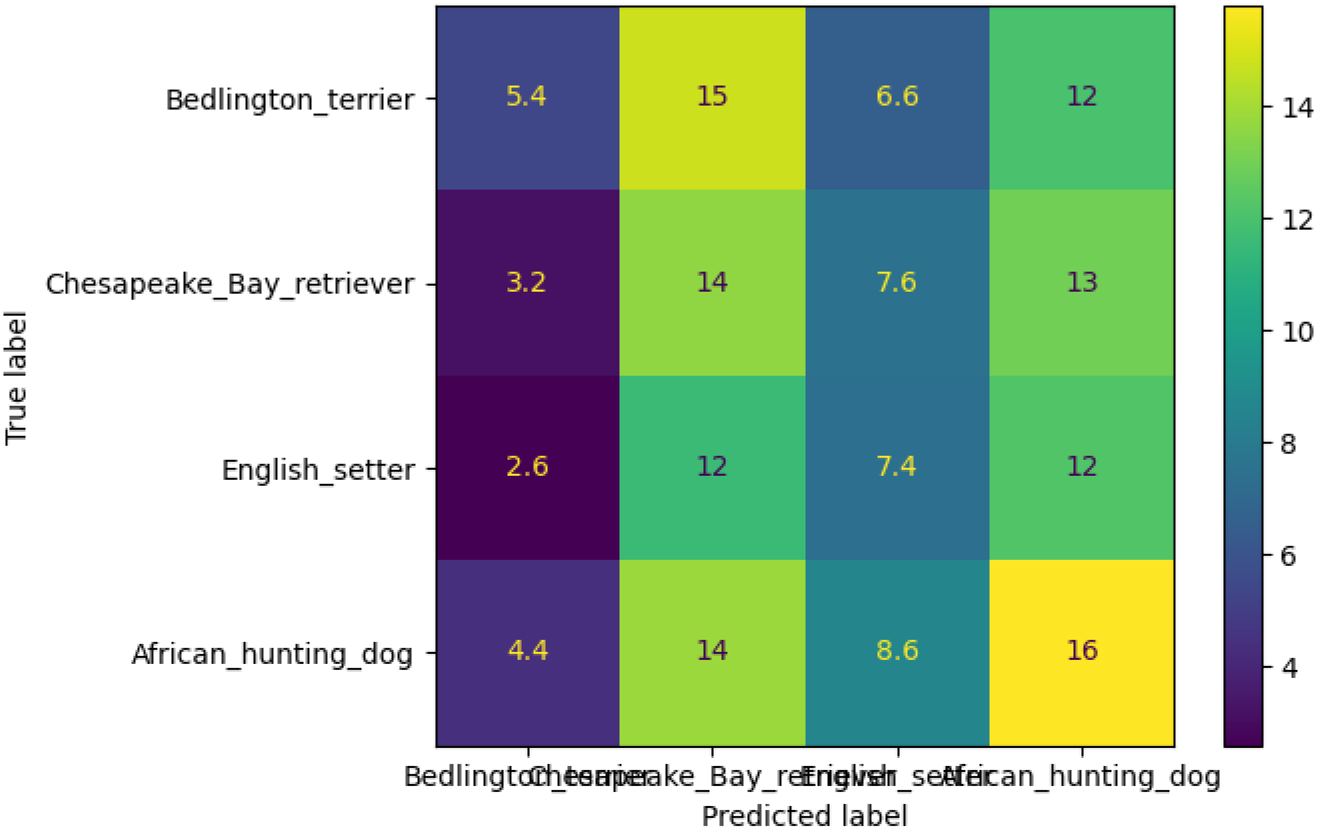
```
Test error when k=3 :0.7124183006535948
```

In [ ]:

In [104…

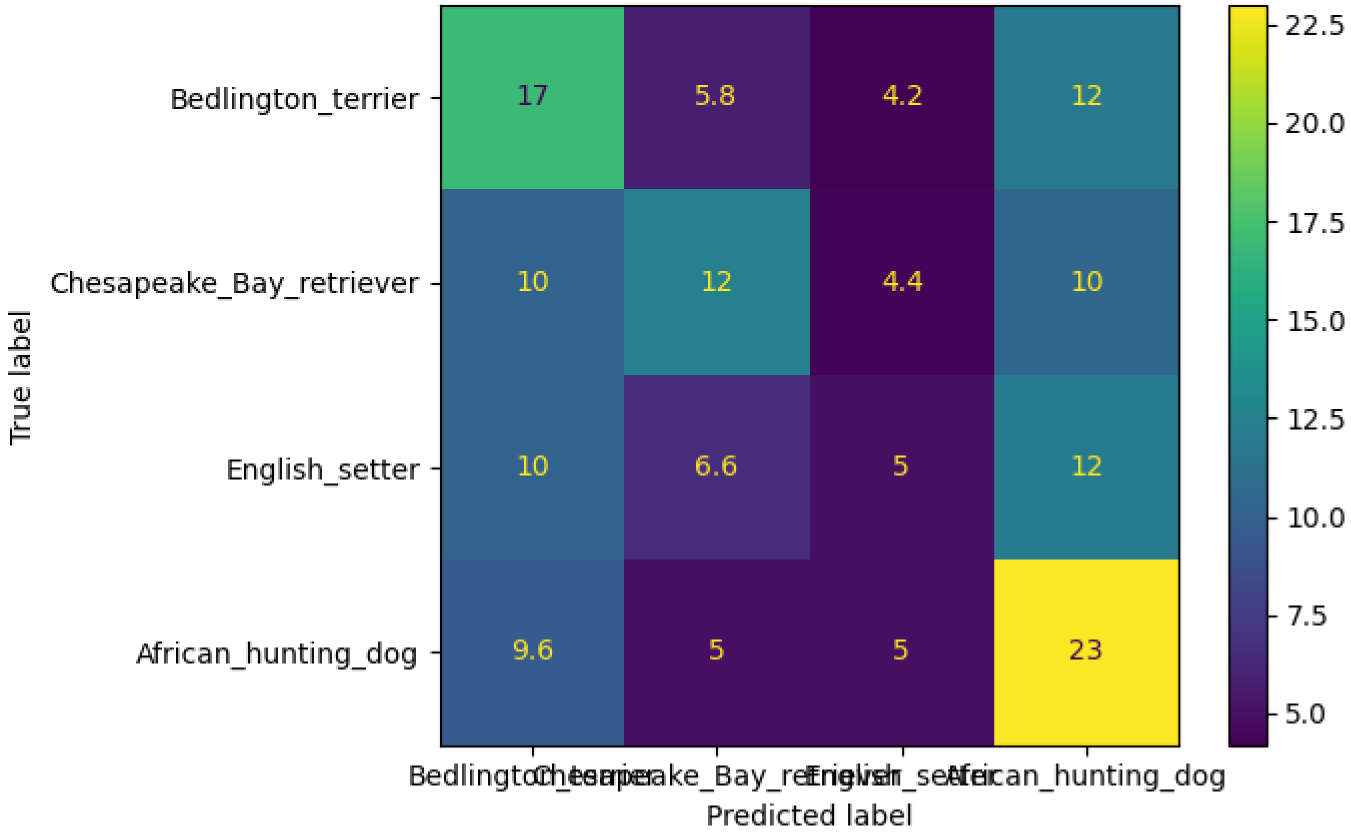
```
#https://www.w3schools.com/python/python_ml_confusion_matrix.asp

from sklearn import metrics
nn_model=MLPClassifier(hidden_layer_sizes=(10,10,10))
rand= RandomForestClassifier()
naiveb=GaussianNB()
for clf in [nn_model,rand,naiveb]:
    print(str(clf)+"\n\n")
    clf.fit(X_scaled,y_train)
    predictions=clf.predict(X_test_scaled)
    confusion_matrix = metrics.confusion_matrix(y_test, predictions)
    report=metrics.classification_report(y_test,predictions)
    print(report)
    truelabels,predictlabels,cm,val_a=[],[],[],[]
    for traini,testi in skf.split(X,Y):
        xtrain,xtest=X[traini],X[testi]
        ytrain,ytest=Y[traini],Y[testi]

        clf.fit(xtrain,ytrain)
        p=clf.predict(xtest)

        truelabels.extend(ytest)
        predictlabels.extend(p)
        val_a.append(metrics.accuracy_score(ytest,p))
        cm.append(metrics.confusion_matrix(ytest,p))
    print("Mean validation acc: "+str(np.mean(val_a)))
    cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = sum(cm)/len(cm), disp
    cm_display.plot()
    plt.show()
```

```
MLPClassifier(hidden_layer_sizes=(10, 10, 10))


              precision    recall  f1-score   support

           0       0.33      0.62      0.43        39
           1       0.26      0.14      0.18        37
           2       0.50      0.12      0.19        34
           3       0.34      0.42      0.38        43

    accuracy                           0.33       153
   macro avg       0.36      0.32      0.29       153
weighted avg       0.35      0.33      0.30       153

Mean validation acc: 0.27652218782249743
```

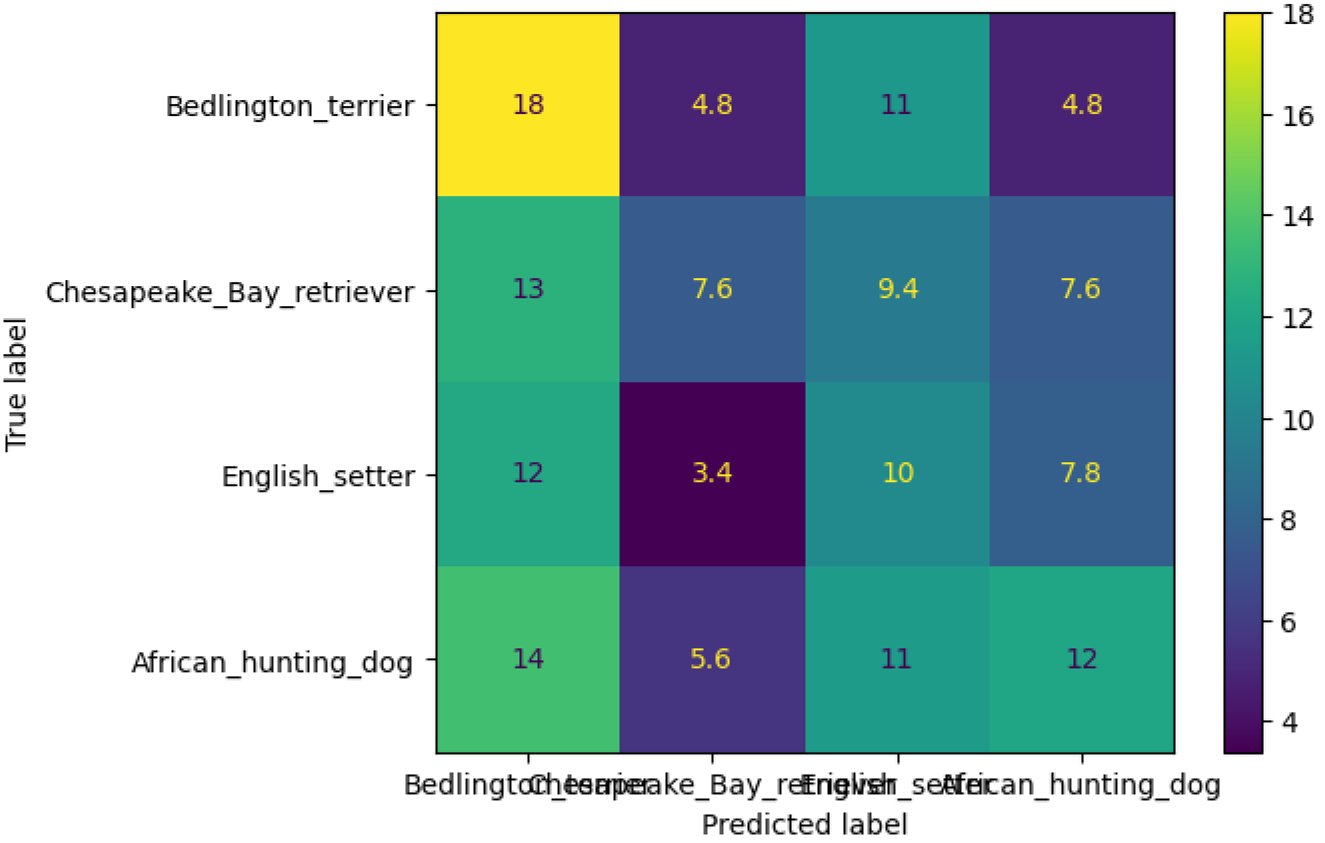


```
RandomForestClassifier()


              precision    recall  f1-score   support

           0       0.40      0.49      0.44        39
           1       0.23      0.16      0.19        37
           2       0.38      0.24      0.29        34
           3       0.41      0.56      0.48        43

    accuracy                           0.37       153
   macro avg       0.36      0.36      0.35       153
```

```
weighted avg       0.36      0.37      0.36       153

Mean validation acc: 0.37621259029927767
```



```
GaussianNB()

              precision    recall  f1-score   support

           0       0.36      0.62      0.45        39
           1       0.44      0.19      0.26        37
           2       0.27      0.24      0.25        34
           3       0.25      0.23      0.24        43

    accuracy                           0.32       153
   macro avg       0.33      0.32      0.30       153
weighted avg       0.33      0.32      0.30       153

Mean validation acc: 0.3145080839353285
```



Byvisually comparing (e.g., looking at the color on the diagonal values, etc.) the three confusion matrices (on the test set), which do you think is the best method? Why? Ans : randomforest -classifies 4 classes better than others

Based on the mean validation accuracies (from the 5-fold cross-validation) for the three methods. Which is the best method? Ans : randomforest (0.357765737874097)

Compute the accuracies for the three methods on the test set. Which is the best method? Ans : randomforest (0.36)

Compute the F-measure for the three methods on the test set. Which is the best method? Ans : randomforest is the best with weighted avg of 0.36

In [ ]: