#import all library files from lib.py

from lib import \*

# Reading data from train.csv

# the data is freely available on kaggle

data=pd.read\_csv("data/emnist-balanced-train.csv")

label=data.iloc[:,0]

def all\_same(items):

return len(set(items)) == 1

# Extract feature columns

feature\_cols = list(data.columns[1:])

# Separate the data into feature data and target data (X and y, respectively)

#X = data[feature\_cols]

X=data.iloc[:,1:]

y = label

# Apply PCA by fitting the data with only 60 dimensions

pca = PCA(n\_components=60).fit(X)

# Transform the data using the PCA fit above

X = pca.transform(X)

y = y.values

variance=pca.explained\_variance\_ratio\_

print(sum(variance))

# Shuffle and split the dataset into the number of training and testing points above

sss = model\_selection.StratifiedShuffleSplit(n\_splits=10, test\_size=0.4, random\_state=42)

for train\_index, test\_index in sss.split(X,y):

X\_train, X\_test = X[train\_index], X[test\_index]

y\_train, y\_test = y[train\_index], y[test\_index]

#Prediction can de done using this function

def predict\_output(X\_test=X\_test,y\_test=y\_test,X\_train=X\_train,y\_train=y\_train):

# Fit a KNN classifier on the training set

knn\_clf = KNeighborsClassifier(n\_neighbors=3, p=2)

knn\_clf.fit(X\_train, y\_train)

# X\_test=pca.transform(X\_test)

# Initialize the array of predicted labels

y\_pred = np.empty(len(y\_test), dtype=np.int)

start = time()

# Find the nearest neighbors indices for each sample in the test set

kneighbors = knn\_clf.kneighbors(X\_test, return\_distance=False)

# For each set of neighbors indices

for idx, indices in enumerate(kneighbors):

# Find the actual training samples & their labels

neighbors = [X\_train[i] for i in indices]

neighbors\_labels = [y\_train[i] for i in indices]

# if all labels are the same, use it as the prediction

if all\_same(neighbors\_labels):

y\_pred[idx] = neighbors\_labels[0]

else:

# else fit a SVM classifier using the neighbors, and label the test samples

svm\_clf = svm.SVC(C=0.5, kernel='rbf', decision\_function\_shape='ovo', random\_state=42)

svm\_clf.fit(neighbors, neighbors\_labels)

label = svm\_clf.predict(X\_test[idx].reshape(1, -1))

y\_pred[idx] = label

end = time()

print("Actual output:{}".format(y\_test))

print("Predicted output:{}".format(y\_pred))

print(accuracy\_score(y\_test, y\_pred))

print("Made predictions in {:.4f} seconds.".format(end - start))

return y\_pred

predict\_output(X\_test,y\_test,X\_train,y\_train)