In [1]:
```
from sklearn.datasets import fetch_openml
from sklearn.metrics import euclidean_distances
from sklearn.model_selection import train_test_split
import numpy as np

class KNN:
    def __init__(self, k):
        self.k = k

    def fit(self, X_train, y_train):
        self.X_train = X_train
        self.y_train = y_train

    def predict(self, X_test):
        y_pred = []
        distances = euclidean_distances(X_test, self.X_train)
        for i in range(X_test.shape[0]):
            indices = np.argsort(distances[i])[:self.k]
            k_nearest_labels = [self.y_train[j] for j in indices]
            y_pred.append(max(k_nearest_labels, key=k_nearest_labels.count))
        return y_pred

    def score(self,X_test, y_test):
        y_pred = self.predict(X_test)
        return (y_pred == y_test).mean()

# load mnist dataset
mnist = fetch_openml('mnist_784')
data = mnist.data
labels = mnist.target

# divide the data into train, validation, and test sets
train_data, val_and_test_data, train_labels, val_and_test_labels = train_test_split(data, labels, test_size=0.2)
val_data, test_data, val_labels, test_labels = train_test_split(val_and_test_data, val_and_test_labels, test_size=0.5)

# train KNN classifier on train_data
knn = KNN(k=5)
knn.fit(train_data, train_labels)

# evaluate training performance
train_acc = knn.score(train_data, train_labels)
print("Training Accuracy:", train_acc)

# evaluate validation performance
val_acc = knn.score(val_data, val_labels)
print("Validation Accuracy:", val_acc)

# evaluate testing performance
test_acc = knn.score(test_data, test_labels)
print("Testing Accuracy:", test_acc)
```

```
Training Accuracy: 0.9833214285714286
Validation Accuracy: 0.9718571428571429
Testing Accuracy: 0.9728571428571429
```

In [2]:
```
from sklearn.metrics import classification_report

# predict labels for test data
y_pred = knn.predict(test_data)

# generate classification report
print(classification_report(test_labels, y_pred, target_names=np.unique(labels)))
```

```
              precision    recall  f1-score   support

           0       0.98      0.99      0.98       712
           1       0.96      0.99      0.98       757
           2       0.99      0.96      0.97       688
           3       0.97      0.97      0.97       704
           4       0.98      0.96      0.97       690
           5       0.97      0.97      0.97       622
           6       0.97      0.99      0.98       685
           7       0.96      0.98      0.97       771
           8       0.99      0.93      0.96       691
           9       0.95      0.96      0.96       680

    accuracy                           0.97      7000
   macro avg       0.97      0.97      0.97      7000
weighted avg       0.97      0.97      0.97      7000
```

In [3]:
```
from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import TfidfVectorizer

class KNN:
    def __init__(self, k):
        self.k = k

    def fit(self, X_train, y_train):
        self.X_train = X_train
        self.y_train = y_train

    def predict(self, X_test):
        y_pred = []
        distances = euclidean_distances(X_test, self.X_train)
        for i in range(X_test.shape[0]):
            indices = np.argsort(distances[i])[:self.k]
            k_nearest_labels = [self.y_train[j] for j in indices]
            y_pred.append(max(k_nearest_labels, key=k_nearest_labels.count))
        return y_pred

    def score(self,X_test, y_test):
        y_pred = self.predict(X_test)
        return (y_pred == y_test).mean()

# load 20 newsgroups dataset
newsgroups = fetch_20newsgroups(subset='all')
data = newsgroups.data
labels = newsgroups.target

# preprocessing and vectorization of the data
vectorizer = TfidfVectorizer()
data = vectorizer.fit_transform(data)

# divide the data into train, validation, and test sets
train_data, val_and_test_data, train_labels, val_and_test_labels = train_test_split(data, labels, test_size=0.2)
val_data, test_data, val_labels, test_labels = train_test_split(val_and_test_data, val_and_test_labels, test_size=0.5)

# train KNN classifier on train_data
knn = KNN(k=5)
knn.fit(train_data, train_labels)

# evaluate training performance
train_acc = knn.score(train_data, train_labels)
print("Training Accuracy:", train_acc)

# evaluate validation performance
val_acc = knn.score(val_data, val_labels)
print("Validation Accuracy:", val_acc)

# evaluate testing performance
test_acc = knn.score(test_data, test_labels)
print("Testing Accuracy:", test_acc)
```

```
Training Accuracy: 0.9127752719554258
Validation Accuracy: 0.8238726790450929
Testing Accuracy: 0.8164456233421751
```

In [4]:
```
# predict labels for test data
y_pred = knn.predict(test_data)

# generate classification report
print(classification_report(test_labels, y_pred, target_names=newsgroups.target_names))
```

```
                          precision    recall  f1-score   support

             alt.atheism       0.74      0.91      0.81        75
           comp.graphics       0.78      0.75      0.76       119
 comp.os.ms-windows.misc       0.82      0.70      0.76       111
comp.sys.ibm.pc.hardware       0.68      0.73      0.71        82
   comp.sys.mac.hardware       0.80      0.75      0.77       103
          comp.windows.x       0.82      0.78      0.80        91
            misc.forsale       0.68      0.60      0.64        83
               rec.autos       0.88      0.86      0.87        92
         rec.motorcycles       0.86      0.87      0.87       101
      rec.sport.baseball       0.87      0.86      0.87       109
        rec.sport.hockey       0.89      0.95      0.92        96
               sci.crypt       0.84      0.91      0.87        99
         sci.electronics       0.81      0.71      0.76        94
                 sci.med       0.93      0.83      0.88        99
               sci.space       0.89      0.86      0.88        95
  soc.religion.christian       0.84      0.84      0.84        85
      talk.politics.guns       0.86      0.87      0.87       100
   talk.politics.mideast       0.81      0.97      0.88       105
      talk.politics.misc       0.70      0.83      0.76        84
      talk.religion.misc       0.77      0.69      0.73        62

                accuracy                           0.82      1885
               macro avg       0.81      0.81      0.81      1885
            weighted avg       0.82      0.82      0.82      1885
```