Imports

```
import numpy as np
from sklearn.cluster import KMeans
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.neighbors import NearestNeighbors
from keras.datasets import mnist
```

Definitons

```
def label_clusters(cluster_counts):
  num_clusters, num_labels = cluster_counts.shape
  cluster_labels = np.zeros(num_clusters, dtype=int)
  label_counts = np.sum(cluster_counts, axis=0)  # How many data points belong to each label

  # Iterate through elements in descending order based on the number of data points per label
  for ni_j in np.argsort(label_counts)[::-1]:
    # Find clusters with the maximum number of data points for this label
    candidate_clusters = np.where(cluster_counts[:, ni_j] == cluster_counts[:, ni_j].max())[0]
    # Assign the label to the first cluster that doesn't have a label or the label isn't assigned yet
    for cluster in candidate_clusters:
      if cluster_labels[cluster] == 0 or ni_j not in cluster_labels:
        cluster_labels[cluster] = ni_j
        break

  return cluster_labels
```

Labels clusters based on the maximum number of training data points with a specific label belonging to each cluster.

Args: cluster_counts: A numpy array where each row represents a cluster and each column represents a label, with the value at each position indicating how many training data points with that label belong to that cluster.

Returns: A numpy array where each element represents the assigned label for the corresponding cluster.

```
def calculate_accuracy(true_labels, predicted_labels):
  return accuracy_score(true_labels, predicted_labels)
```

Load Data

```
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

```
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1] * X_train.shape[2])
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1] * X_test.shape[2])
```

For using method like prediction, They want to 2d not 3d

```
distance_metrics = ['euclidean', 'manhattan', 'cosine']
```

```
for distance_metric in distance_metrics:
  print("Distance Metric:", distance_metric)
  accuracy_scores = []
  cluster_counts = np.zeros((10, 10), dtype=int)  # Initialize a table to store cluster counts

  for i in range(5):
    kmeans = KMeans(n_clusters=10, init="random", n_init=10, random_state=i, algorithm="lloyd")
    kmeans.fit(X_train)

    y_pred = kmeans.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    accuracy_scores.append(accuracy)

    # Count how many data points from each label belong to each cluster
    for j in range(len(y_train)):
      cluster_counts[kmeans.labels_[j], y_train[j]] += 1
    # Assign labels to clusters based on cluster_counts
    cluster_labels = label_clusters(cluster_counts)

    # Training Error
    y_train_pred = np.zeros_like(y_train)
    for j in range(len(y_train)):
      # Use 1-NN to assign training data point to a cluster based on labeled kmeans centers
      nbrs = NearestNeighbors(n_neighbors=1, algorithm='brute')
      nbrs.fit(kmeans.cluster_centers_)
      distances, indices = nbrs.kneighbors(X_train[j].reshape(1, -1))
      y_train_pred[j] = cluster_labels[indices[0][0]]

    train_confusion_matrix = confusion_matrix(y_train, y_train_pred)
    train_accuracy = calculate_accuracy(y_train, y_train_pred)
    print("Iteration", i+1, "- Accuracy:", accuracy)

    y_test_pred = np.zeros_like(y_test)

    for j in range(len(y_test)):
      nbrs = NearestNeighbors(n_neighbors=1, algorithm='brute')
      nbrs.fit(kmeans.cluster_centers_)
      distances, indices = nbrs.kneighbors(X_test[j].reshape(1, -1))
      y_test_pred[j] = cluster_labels[indices[0][0]]

    test_confusion_matrix = confusion_matrix(y_test, y_test_pred)
    test_accuracy = calculate_accuracy(y_test, y_test_pred)

    print("Training Accuracy:", train_accuracy)
    print("Training Confusion Matrix:\n", train_confusion_matrix)
    print("Test Accuracy:", test_accuracy)
    print("Test Confusion Matrix:\n", test_confusion_matrix)

  print("Mean Accuracy:", np.mean(accuracy_scores))
  print()
```

```
Distance Metric: euclidean
Iteration 1 - Accuracy: 0.1103
Training Accuracy: 0.5057666666666667
Training Confusion Matrix:
 [[5330    3   17    0   38  162  182   14  177    0]
 [2978 3717    9    0    6    5    8    9   10    0]
 [ 462  362 4193    0  174  329  211   69  158    0]
 [ 248  425  216    0  175 3921   57   48 1041    0]
 [ 331  159   37    0 3189    1  165 1941   19    0]
 [1131  166   15    0  378 1769  122  358 1482    0]
 [ 447  271   86    0   82   28 4913    1   90    0]
 [ 311  327   39    0 1796    5    4 3773   10    0]
 [ 490  296   54    0  193 1130   47  179 3462    0]
 [ 172  239   13    0 2903   85    8 2460   69    0]]
Test Accuracy: 0.5074
Test Confusion Matrix:
 [[876   0   2   0   4  48  30   3  17   0]
 [469 660   1   0   0   2   2   0   1   0]
 [107  59 707   0  30  65  25  11  28   0]
 [ 28  73  40   0  15 696   7   7 144   0]
 [ 38  30   5   0 559   0  36 310   4   0]
 [155  23   4   0  55 287  20  72 276   0]
 [ 74  28  18   0  22   2 795   1  18   0]
 [ 59  59  13   0 292   0   1 603   1   0]
 [ 63  34   7   0  31 207  10  35 587   0]
 [ 25  29   3   0 541   7   4 387  13   0]]
Iteration 2 - Accuracy: 0.1929
Training Accuracy: 0.3615333333333333
Training Confusion Matrix:
 [[5326   21   14    0   38  161  181    0  182    0]
 [3727 2977    9    0    6    5    8    0   10    0]
 [4660  355   70    0  173  330  212    0  158    0]
 [ 779  106   48    0  175 3920   58    0 1045    0]
 [ 217  314 1936    0 3188    1  167    0   19    0]
 [ 486  814  337    0  376 1763  123    0 1522    0]
 [ 569  230    1    0   83   28 4915    0   92    0]
 [ 393  286 3774    0 1793    5    4    0   10    0]
 [ 422  422  181    0  192 1134   47    0 3453    0]
 [ 308  118 2462    0 2899   85    8    0   69    0]]
Test Accuracy: 0.358
Test Confusion Matrix:
 [[873   5   3   0   4  48  30   0  17   0]
 [662 468   0   0   0   2   2   0   1   0]
 [789  84  11   0  30  65  25   0  28   0]
 [133   7   7   0  15 694   7   0 147   0]
 [ 36  38 309   0 559   0  35   0   5   0]
 [ 76 108  68   0  55 287  19   0 279   0]
 [ 89  31   1   0  22   2 794   0  19   0]
 [ 74  58 602   0 292   0   1   0   1   0]
 [ 58  45  35   0  31 207  10   0 588   0]
 [ 46  11 387   0 541   7   4   0  13   0]]
Iteration 3 - Accuracy: 0.0965
Training Accuracy: 0.35541666666666666
Training Confusion Matrix:
 [[5528   21   17    0   14  162  181    0    0    0]
 [3733 2978    9    0    9    5    8    0    0    0]
 [ 801  352 4196    0   69  328  212    0    0    0]
 [1781  106  219    0   48 3922   55    0    0    0]
```