```
import numpy as np
from keras.datasets import mnist
from sklearn.cluster import KMeans, MiniBatchKMeans
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, normalize
from sklearn.neighbors import NearestNeighbors

def load_and_preprocess_data():
    (X_train, y_train), (X_test, y_test) = mnist.load_data()
    # Flatten and normalize the images
    X = np.vstack((X_train, X_test)).astype(np.float32)
    y = np.concatenate((y_train, y_test))
    X = X.reshape((X.shape[0], -1))
    scaler = StandardScaler()
    X = scaler.fit_transform(X)
    return X, y

def apply_kmeans(X, n_clusters=10, distance='euclidean'):
    if distance == 'euclidean':
        kmeans = KMeans(n_clusters=n_clusters, random_state=0)
    elif distance == 'manhattan':
        # We use MiniBatchKMeans as a workaround since it allows
batch_size specification which is necessary for large datasets
        kmeans = MiniBatchKMeans(n_clusters=n_clusters,
random_state=0, batch_size=10000)
    elif distance == 'cosine':
        # Normalize data for cosine similarity
        X = normalize(X)
        kmeans = KMeans(n_clusters=n_clusters, random_state=0)
    labels = kmeans.fit_predict(X)
    return labels, kmeans

def label_clusters(y, labels, n_clusters=10):
    cluster_labels = np.zeros(n_clusters, dtype=int)
    label_count_matrix = np.zeros((n_clusters, 10), dtype=int)
    for i in range(n_clusters):
        indices = np.where(labels == i)[0]
        cluster_labels[i] = np.bincount(y[indices],
minlength=10).argmax()
        label_count_matrix[i] = np.bincount(y[indices], minlength=10)
    return cluster_labels, label_count_matrix

def evaluate_accuracy(y_true, predicted_labels):
    conf_matrix = confusion_matrix(y_true, predicted_labels)
    accuracy = accuracy_score(y_true, predicted_labels)
    return conf_matrix, accuracy


X, y = load_and_preprocess_data()
```

```
distances = ['euclidean', 'manhattan', 'cosine']
results = {}

for distance in distances:
    print(f"\nUsing distance: {distance}")
    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=None) # test with 20% of data
    labels_train, kmeans = apply_kmeans(X_train, distance=distance)
    cluster_labels, label_count_matrix = label_clusters(y_train,
labels_train)

    # Print the label count matrix
    print("Label Count Matrix:")
    print(label_count_matrix)

    # Predict labels for training data
    predicted_labels_train = cluster_labels[labels_train]
    train_conf_matrix, train_accuracy = evaluate_accuracy(y_train,
predicted_labels_train)
    print("Training Confusion Matrix:")
    print(train_conf_matrix)
    print(f"\nTraining Accuracy: {train_accuracy:.4f}")

    # Use 1-NN to classify test data
    nn = NearestNeighbors(n_neighbors=1)
    nn.fit(kmeans.cluster_centers_)
    distances, indices = nn.kneighbors(X_test)
    predicted_labels_test = cluster_labels[indices.flatten()]

    test_conf_matrix, test_accuracy = evaluate_accuracy(y_test,
predicted_labels_test)
    print("Test Confusion Matrix:")
    print(test_conf_matrix)
    print(f"\nTest Accuracy: {test_accuracy:.4f}")
```

```
Using distance: euclidean

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/
_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(

Label Count Matrix:
[[ 207  117   36   69  723 2024   61  211 1575  139]
 [   8    4   29   76  767   74    1 3287  161 2015]
 [   8    7  124  161 3084  241   20 1158  266 2667]
 [  97   10 1925  684   75  223  305   14  207   21]
```

```
 [ 242   12  701   65   99   71 4252    3   21    2]
 [1185   23  571 3224    3 1717  113    6 2115  102]
 [2884    0   23    7   38   25  243   13   37   35]
 [  32 6124  669  436  429  369  496  400  939  261]
 [ 827    8 1475  952   27  195   28   12  105   17]
 [   6    0   32   73  239   42    4  728   56  306]]
Training Confusion Matrix:
[[2884   32  924 1185    8  207  242   14    0    0]
 [   0 6124   18   23    7  117   12    4    0    0]
 [  23  669 3400  571  124   36  701   61    0    0]
 [   7  436 1636 3224  161   69   65  149    0    0]
 [  38  429  102    3 3084  723   99 1006    0    0]
 [  25  369  418 1717  241 2024   71  116    0    0]
 [ 243  496  333  113   20   61 4252    5    0    0]
 [  13  400   26    6 1158  211    3 4015    0    0]
 [  37  939  312 2115  266 1575   21  217    0    0]
 [  35  261   38  102 2667  139    2 2321    0    0]]

Training Accuracy: 0.5180
Test Confusion Matrix:
[[ 750    3  224  318    2   57   49    4    0    0]
 [   0 1526    7    4    2   26    4    3    0    0]
 [   9  163  855  147   46   10  164   11    0    0]
 [   2  116  372  789   37   25   15   38    0    0]
 [  14  108   18    2  734  189   34  241    0    0]
 [   6   93  107  487   62  526   22   29    0    0]
 [  66  103   78   18    7   19 1061    1    0    0]
 [   4  100    3    3  296   38    0 1017    0    0]
 [   8  229   52  560   69  354   11   60    0    0]
 [   6   50    6   27  689   30    2  583    0    0]]

Test Accuracy: 0.5184

Using distance: manhattan

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/
_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 3 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(

Label Count Matrix:
[[ 138   15  979   61   55   56 4023    3   13    1]
 [   5    0   22   74  460   55    2 1148   57  806]
 [ 505  591 1624  591  411 1657  276  158 1363   78]
 [  43   10  256  196 2817  388   72  765  505 2141]
 [ 132    8 1505  824  162  322  146   17  341   84]
 [1658    3  332 3036    2 1912   90    4 1246   61]
 [  15    5   24   68 1248  269    1 2208  354 1982]
 [  42 5750  709  836  248  391  585  391 1514  247]
```

```
 [   0    0    0    7    2    7    0 1128    7  209]
 [2989    0   74    6   35   16  254   10   31   33]]
Training Confusion Matrix:
[[2989   42  132 1658   43  505  138   20    0    0]
 [   0 5750    8    3   10  591   15    5    0    0]
 [  74  709 1505  332  256 1624  979   46    0    0]
 [   6  836  824 3036  196  591   61  149    0    0]
 [  35  248  162    2 2817  411   55 1710    0    0]
 [  16  391  322 1912  388 1657   56  331    0    0]
 [ 254  585  146   90   72  276 4023    3    0    0]
 [  10  391   17    4  765  158    3 4484    0    0]
 [  31 1514  341 1246  505 1363   13  418    0    0]
 [  33  247   84   61 2141   78    1 2997    0    0]]

Training Accuracy: 0.4689
Test Confusion Matrix:
[[ 775   10   34  379    8  131   34    5    0    0]
 [   0 1332    2    1    6  152    2    0    0    0]
 [  16  192  384   90   70  434  265   14    0    0]
 [   3  202  208  771   67  137   20   34    0    0]
 [  16   61   49    1  707   99   13  438    0    0]
 [   5   91   73  474  103  411   12   71    0    0]
 [  59  154   35   22   13   81 1061    2    0    0]
 [   7   93    3    0  180   38    0 1140    0    0]
 [  14  397   79  300  122  362    8  112    0    0]
 [   6   51   21   21  498   11    0  708    0    0]]

Test Accuracy: 0.4701

Using distance: cosine

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/
_kmeans.py:870: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(

Label Count Matrix:
[[   2 2772  100   28   86   14   46  108  165   21]
 [ 198   23  216 3649    0 1536   19    5 1790   95]
 [ 178   29 4046  477  154  196 1234   56  238   52]
 [4257    0  115   39   41   78  172   29   60   51]
 [ 516  200  817 1031  511 2538  229  187 2404  108]
 [   5    0   19   53  897   56    1 2474  139 1388]
 [   0 3218   21  140   54   44   32   64  126   57]
 [   8    6  101  140 2405  271   18 1003  303 1811]
 [ 386   20  171   72  105  120 3735    0   55    6]
 [  10    9   12   85 1225  131    1 1920  190 1977]]
Training Confusion Matrix:
[[4257    2  178  198    8  516  386    5    0   10]
```

```
 [   0 5990   29   23    6  200   20    0    0    9]
 [ 115  121 4046  216  101  817  171   19    0   12]
 [  39  168  477 3649  140 1031   72   53    0   85]
 [  41  140  154    0 2405  511  105  897    0 1225]
 [  78   58  196 1536  271 2538  120   56    0  131]
 [ 172   78 1234   19   18  229 3735    1    0    1]
 [  29  172   56    5 1003  187    0 2474    0 1920]
 [  60  291  238 1790  303 2404   55  139    0  190]
 [  51   78   52   95 1811  108    6 1388    0 1977]]

Training Accuracy: 0.5548
Test Confusion Matrix:
[[1180    8    6   28    1   40   78    1    0    1]
 [   0 1591    1    4    1    2    0    0    0    1]
 [ 146  291  663   52   17   98   92    7    0    6]
 [  72  224   44  841   30  153   22   23    0   18]
 [  65  151   12    0  490   27   31  237    0  333]
 [ 112  224   35  372   64  401   41   38    0   42]
 [ 144  121  184    4    3    9  924    0    0    0]
 [  15  156    2    2  204   15    0  583    0  470]
 [  67  343   21  375   70  357   20   54    0   48]
 [  22  101    5   12  422    3    1  333    0  493]]

Test Accuracy: 0.5119
```

- `load_and_preprocess_data()`:
  - This function loads the MNIST dataset, combines training and test sets, and then flattens and normalizes the images.
- `apply_kmeans()`:
  - Depending on the metric argument, different types of KMeans clustering are applied.
  - Euclidean: The standard KMeans class is used.
  - Manhattan: MiniBatchKMeans is used with a large batch size as a workaround because the regular KMeans does not support Manhattan distance directly.
  - Cosine: Data is normalized using the normalize function to ensure that the angle between points is considered rather than the distance. The standard KMeans is then applied.
- `label_clusters()`:
  - This function assigns the most frequent true label to each cluster.
  - It creates a label count matrix that counts how many times each true label occurs in each cluster.
- `evaluate_accuracy()`:
  - Calculates the confusion matrix and accuracy score to evaluate the clustering performance.
- Clustering Loop:
  - For each metric, the dataset is split into training and test sets, and clustering is applied.

- It prints out the label count matrix for each cluster, evaluates training data accuracy, and then uses a 1-nearest neighbor (1-NN) approach to classify test data based on the nearest cluster center.

Distances

1. Euclidean Distance (L2 norm): The standard distance metric for many clustering algorithms, including KMeans, where the distance between two points is the square root of the sum of the squared differences between their coordinates.

2. Manhattan Distance (L1 norm): Sum of the absolute differences between points in all dimensions. MiniBatchKMeans is used here because the standard KMeans implementation does not support Manhattan distance.

3. Cosine Similarity: Measures the cosine of the angle between vectors (points) in a space. This is normalized in the data preprocessing step for this metric, making it sensitive to the direction of the data rather than the magnitude.

Confusion Matrices

The label count matrix counts occurrences of labels within clusters and does not involve any prediction. It shows the most representative label for each cluster.

In contrast, the training and test confusion matrices are used to evaluate accuracy and identify misclassifications. They use for calculating the accuracy.