MNIST Handwritten Digits Dataset

Classification and Clustering of MNIST handwritten digits dataset.
Identification of handwritten digits and clustering into classes.

Problem description

1. Classification
   Classifying the handwritten digits into one of the 10 classes.
   10 classes 0-9
2. Clustering
   Based on component analysis, clustering the images into digit sets.
   10 clusters

Algorithm Used

1. K Nearest Neighbours - classification
2. K Means clustering

Team members


```
1. Ravi Shankar Chegondi   16BCB0056
2. Kesav Swaroop Reddy     16BCB0100
3. Nishant Rohan Rodrigues 16BCE0098
```


In [1]:

```
# Libraries

import os

import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

from time import time, sleep
from tqdm import tqdm

from sklearn.neighbors import KNeighborsClassifier
from sklearn.cluster import KMeans
from sklearn.metrics import confusion_matrix

start = time()
```

Dataset Description:

Source: http://yann.lecun.com/exdb/mnist/

Dimension:

```
Training examples 1934
Testing examples  946
Image shape       32*32
```

Conversion:

Convert the binary image to a vector representation

Example:

Binary image file:

```
00000
00110
01100
```

Vector representation:

```
[0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0]
```

Missing data:

None

Attributes:

1024 pixel values - Binary 0 or 1

In [2]:

```
# Dataset

print('Training data')
print('Folder name: trainDigits')
print('Path: ./trainingDigits')
print('Training examples: ', len(os.listdir('./trainingDigits')))

print('\nTesting data')
print('Folder name: testDigits')
print('Path: ./testDigits')
print('Testing examples: ', len(os.listdir('./testDigits')))
```

```
Training data
Folder name: trainDigits
Path: ./trainingDigits
Training examples:  1934

Testing data
Folder name: testDigits
Path: ./testDigits
Testing examples:  946
```

In [3]:

```
'''
    Convert the binary image to a vector representation
    Example:
    image.txt
    00000
    00110
    01100
    Vector representation:
    image vector
    [0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0]
'''
# img2vector
# Args
# filename: file name
# Return
# returnVect: vector representation of image
def img2vector(filename):

    returnVect = []
    with open('./trainingDigits/' + filename) as f:
        lines = f.readlines()
    for line in lines:
        line = list(map(int, list(line.strip())))
        returnVect.append(line)

    returnVect = np.array(returnVect).reshape(1024)
    label = int(filename.split('_')[0])
```

```
    return returnVect, label
```

In [4]:

```
# Create training and testing dataset

X_train, Y_train = [], []
X_test, Y_test = [], []

# Training set
print('Generating trainig set')
sleep(2)
for file in tqdm(os.listdir('./trainingDigits')):
    data, label = img2vector(file)
    X_train.append(data)
    Y_train.append(label)

# Testing set
print('Generating testing set')
sleep(2)
for file in tqdm(os.listdir('./testDigits')):
    data, label = img2vector(file)
    X_test.append(data)
    Y_test.append(label)
```

```
Generating trainig set
```

```
100%|██████████| 1934/1934 [00:00<00:00, 3483.34it/s]
```

```
Generating testing set
```

```
100%|██████████| 946/946 [00:00<00:00, 3383.11it/s]
```

K Nearest Neighbours

Applying KNN Algorithm K=3.
Fitting the model with the training data
Prediction class [0-9] of testing data

In [5]:

```
# KNearestNeighbours

# Generating KNearestNeighbors classifier with K=3
knn_model = KNeighborsClassifier(n_neighbors=3)

# Fitting model with dataset and labels (training data)
knn_model.fit(X_train, Y_train)
```

Out[5]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=3, p=2,
           weights='uniform')
```

In [6]:

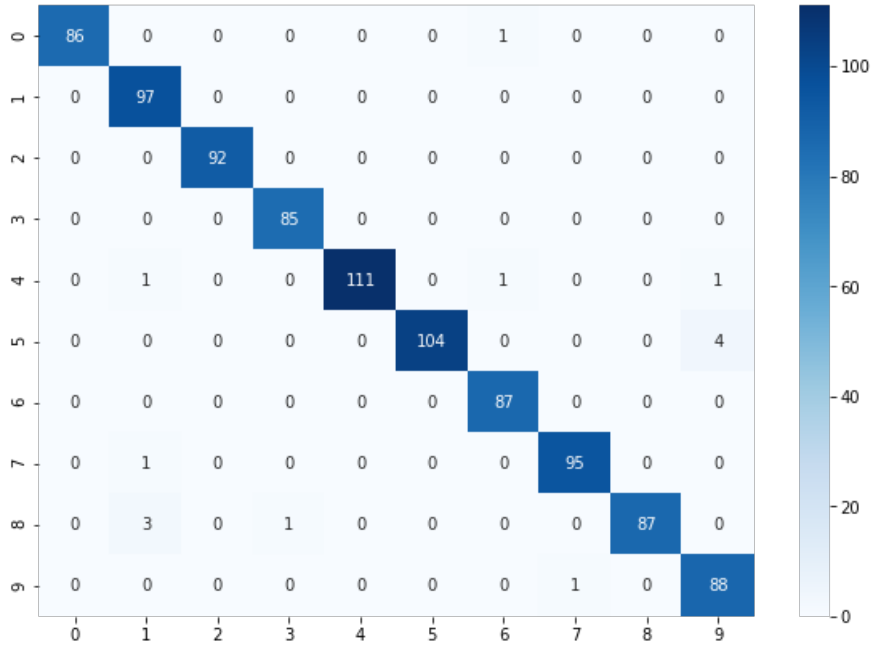
```
# Checking score of trainined model
score = knn_model.score(X_test, Y_test)
print('Accuracy: ', score*100)
```

```
Accuracy:  98.5200845665962
```

In [7]:

```
# Counfusion matrix

predict = knn_model.predict(X_test)
```

```
predict = knn_model.predict(X_test)
conf = confusion_matrix(Y_test, predict)

plt.figure(figsize=(10, 7))
sns.heatmap(conf, annot=True, fmt='g', cmap = 'Blues')
plt.show()
```



K Means Clustering

Applying K Means clustering K=10
Fitting the model with testing data
Clustering data into 10 classes

In [8]:

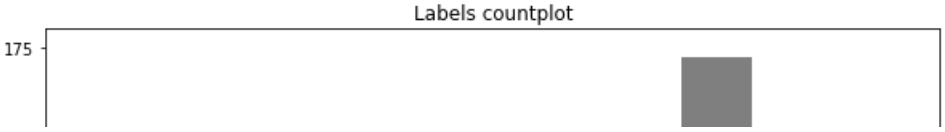
```
# Clustering

# Creating clustering model with n_clusters = 10 [0-9]
kmeans_model = KMeans(n_clusters=10, random_state=2019)

# Fitting model with test data
kmeans_model.fit(X_test)
```

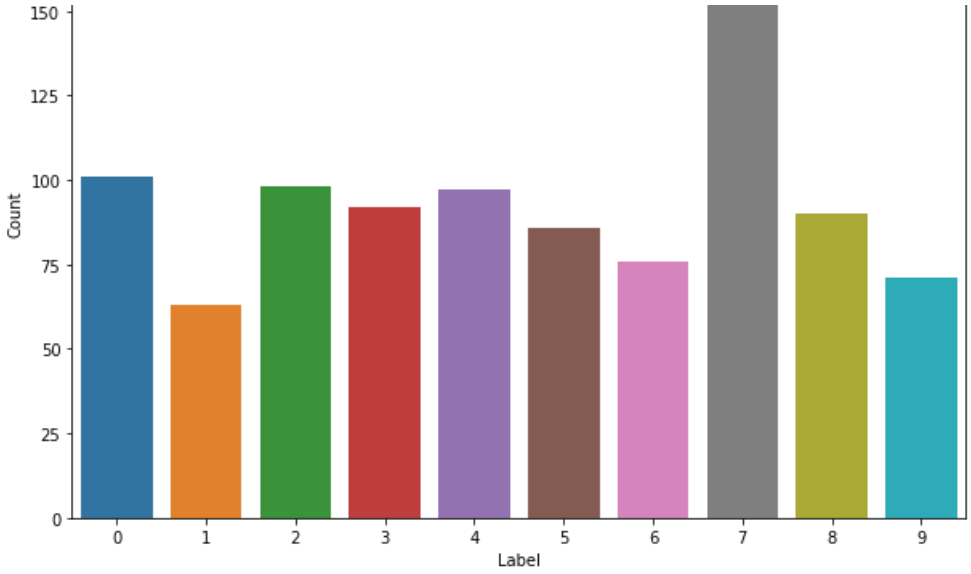
Out[8]:

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=10, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=2019, tol=0.0001, verbose=0)
```

In [9]:

```
# Countplot of clusered classes

labels = kmeans_model.labels_
plt.figure(figsize=(10, 7))
sns.countplot(labels)
plt.title('Labels countplot')
plt.xlabel('Label')
plt.ylabel('Count')
plt.show()
```



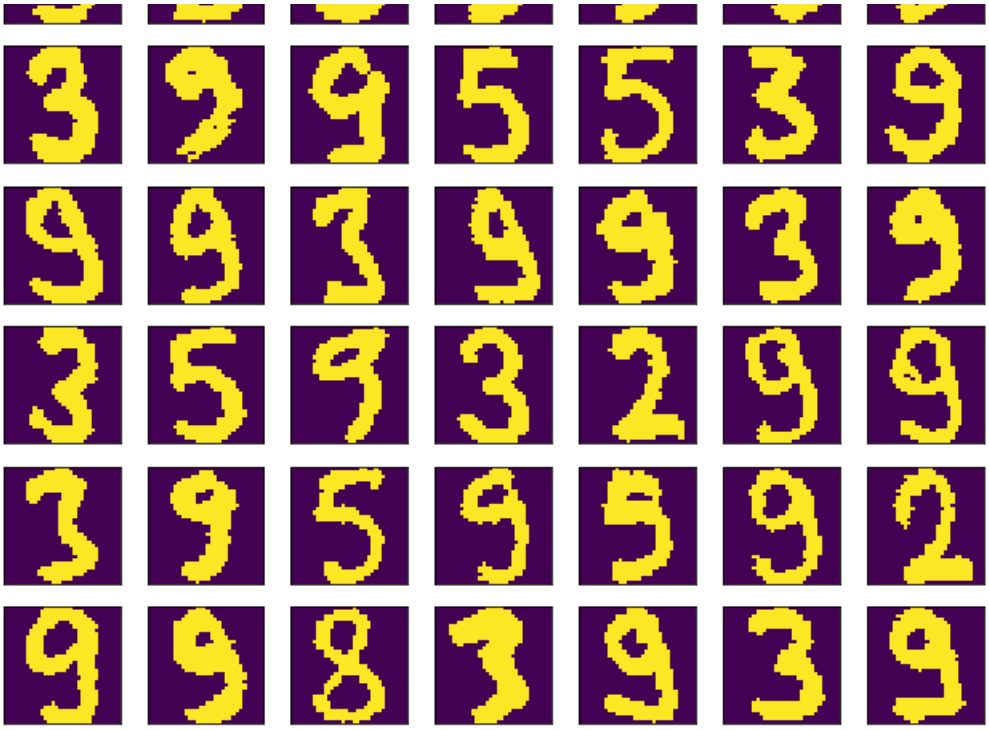


In [10]:

```
# Display images

'''
    Convert the binary image to a vector representation
    Example:
    image.txt
    00000
    00110
    01100
    Vector representation:
    image vector
    [0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0]
'''

# show_clustered_images
# Args
# label: label value (default=7)
# Return
# display 49 instances from the cluster
def show_clustered_images(label=7):

    plt.figure(figsize=(11, 11))
    count = 0
    for idx, label in enumerate(labels):
        if label == 7:
            plt.subplot(7, 7, count+1)
            plt.imshow(np.array(X_test[idx]).reshape(32, 32))
            plt.xticks([])
            plt.yticks([])
            count += 1
        if count==49:
            break

    plt.show()
```

In [11]:

```
# Display cluster (label = 7)

show_clustered_images(7)
```



In [12]:

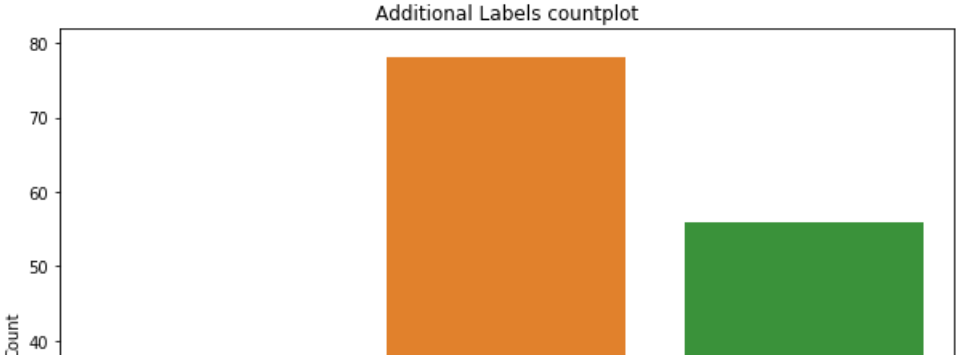
```
# Creating dataset with label 7

X_test_additional = []
for idx, label in enumerate(labels):
    if label == 7:
        X_test_additional.append(X_test[idx])

# Clustering label = 7 further
# Splitting into 3 classes

kmeans_model_2 = KMeans(n_clusters=3, random_state=2019)
kmeans_model_2.fit(X_test_additional)
```

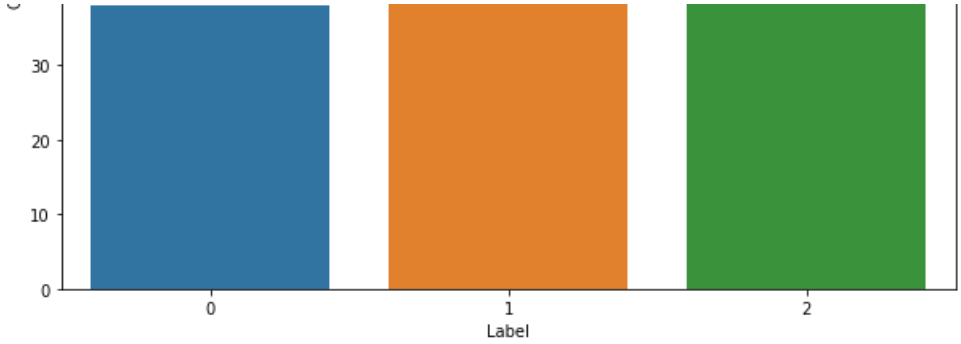
Out[12]:

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=3, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=2019, tol=0.0001, verbose=0)
```

In [13]:

```
# Countplot of additional clusters

labels_new = kmeans_model_2.labels_
plt.figure(figsize=(10, 7))
sns.countplot(labels_new)
plt.title('Additional Labels countplot')
plt.xlabel('Label')
plt.ylabel('Count')
plt.show()
```





In [14]:

```
# Kernel runtime

runtime = time() - start
print('Runtime:', runtime//60, 'mins', round(runtime%60, 2), 's')
```

```
Runtime: 0.0 mins 14.43 s
```

Conclusion

Classification

K Nearest Neighbours works very well for this data set.

Accuracy ~ 98.5%

It is able to classify the test data into the correct categories as the trainig set is large enough, and contains a number of sample cases of each class. This is a good model in terms of prediciton accuracy, but suffers from memory usage as the entire training dataset is loaded in the memory at all times. Also it has a high time complexity as the distance needs to be calculated for each and every data point in the testing dataset to every point in the training dataset for classification.

Clustering

K Means clustering does not work well on this dataset.

We get a vague result with a lot of `False` data in each cluster.

This is becaues of the architecture of the KMeans clustering algorithm. The algorithm is good a clustering similar data (Mean), but in this case, there are a lot of examples that appear similar thoughout, but have only a small difference. As the KMeans algorithm cannont extract such features and differentiate based on them, it does not make an ideal case for clustering this data.

References

1. Dataset - http://yann.lecun.com/exdb/mnist/
2. Classification - https://www.toppr.com/guides/biology/diversity-in-living-organism/classification-and-its-types/
3. Clustering - https://www.analyticsvidhya.com/blog/2016/11/an-introduction-to-clustering-and-different-methods-of-clustering/
4. K Nearest Neighbours - https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/
5. K Means clustering - https://www.datascience.com/blog/k-means-clustering