Nicholas Ross - Exercise 3

In [ ]:
```
import os
import numpy as np
from PIL import Image
import pandas as pd
from scipy.spatial import distance
from sklearn.neighbors import KNeighborsClassifier


def load_dataset(image_path):
    dataset = {
        'label' : [],
        'image' : []
    }

    for dir in os.listdir(image_path):
        if dir not in '.DS_Store':
                class_path = os.path.join(image_path, dir)
                for image in os.listdir(class_path):
                    dataset['image'].append(np.array(Image.open(os.path.j
                    dataset['label'].append(int(dir))

    return pd.DataFrame(dataset)

test_set = load_dataset('files/MNIST/MNIST/test')
train_set = load_dataset('files/MNIST/MNIST/train')
```

In [ ]:
```
# feature splits image into 16 tiles then calculates ratio of black to wh
def black_ratio(img):
    image_tile = []

    # create list of grid tiles
    for i in range(0,4):
        for j in range(0,4):
            image_tile.append(img[i * 7:(i+1) * 7, j * 7:(j+1) * 7])

    feature_ratio = []
    for i in range(len(image_tile)):
        # Count number of non  zero pixels(Black)
        feature_ratio.append(np.count_nonzero(np.asarray(image_tile)[i])/
    return np.asarray(feature_ratio)

def histogram_pixels(img):
    # count number of pixel not black from both horizontal and Vericale a
    return  np.concatenate( (np.count_nonzero(img, axis=1), np.count_nonz
```

Two features

Black_ratio: Images are divided into a grid of 16 boxes (7x7 pixels) each grid box I calculated the number of non black pixels and calculated the ratio of black to no black in each grid (each box being a feature in the vector)

Historgram_pixels: I calculated the number of non black pixels for the horizontal and vertical axis to form a two histograms the two feature vectors were combined. 28x28 image = 56 length feature vector

In [ ]:
```
# Create dataset with img vector transformed to new feature vector repres

# test_set['image'] = test_set['image'].apply(lambda img : histogram_pixe
# train_set['image'] = train_set['image'].apply(lambda img : histogram_pi

test_set['image'] = test_set['image'].apply(lambda img : black_ratio(img)
train_set['image'] = train_set['image'].apply(lambda img : black_ratio(im
```

In [ ]:
```
# calculate the mean vector for each class
def build_model(x):
    model = {}
    for i in range(10):
            model[i] = x[x['label']==i]['image'].mean()
    return model

# find the class based on the clossest mean vector
def classify(x, model):
    dists = np.zeros(10)

    for key in model:
        dists[key] = distance.cosine(x, model[key])

    # find the class with shortest distance to image
    return dists.argmin()


model = build_model(train_set)

# compute distance to nearest class for each image
predictions = test_set.apply(lambda x : classify(x['image'], model),axis

# create datatable with column of predicted class another column contain
mm = pd.concat([test_set['label'], predictions], axis=1)

# evaluate weather the predicted and true class are equal. Returns accuar
mm.apply(lambda x : 1 if x['label']==x[0] else 0, axis=1).sum()/(len(test_
```

Out[ ]:
```
0.6556
```

In [ ]:
```
#sklearn KNN implenentation
knn= KNeighborsClassifier(n_neighbors=5)
knn.fit(np.vstack(train_set['image']),train_set['label'])
knn.score(np.vstack(test_set['image']), test_set['label'])
```

Out[ ]:
```
0.8385
```