Introduction:
In this part of the project, I will use the hierarchical k-means clustering to construct a dictionary of image patches and train a classifier or machine learning model based on the histogram of patches and test this model on the testing data. I will also seek improvement to this model to enhance its performance. I will break this part of the project into 5 parts:

- Part 1: Construct the dictionary of image patches and cluster each dataset in the dictionary to numerous centers
- Part 2: Construct a histogram of patches for each testing image
- Part 3: Train a decision tree classifier based on the histogram of patches and evaluate its performance
- Part 4: Seek improvement in the performance of the classifier
- Part 5: Compare this classifier to other high-ranking methods of MNIST

Part 1:
I use the hierarchical k-means to build a dictionary of image patches. At this time, I will keep the original data untouched.
Firstly, I construct a collection of 10 × 10 image patches by extracting these patches from the training images on an overlapping 4 × 4 grid, which means that each training image produces 16 overlapping patches and there is a total of 960000 training patches.
Secondly, for each training image, I choose one of these patches uniformly and randomly, and I subsample this dataset of 60000 patches uniformly and randomly to produce a dataset of 6000 elements.
Thirdly, I cluster this dataset to 50 centers, and I build 50 datasets corresponding to each cluster center by taking each element of the 60000 patch dataset, finding which of the cluster centers is closest to it, putting the patch in that center's dataset, and finally clustering each of these datasets to the 50 centers. (For codes, please refer to the file called "part2_code.py")

Part 2:
Now, my dictionary has 2500 entries. For each query image, I construct a set of 10 × 10 patches on an overlapping 4 × 4 grid. For each of the centers, I extract 9 immediate patches around it according to the coordinate of the centers, which means each test image will have 144 associated patches. I use my dictionary to find the closest center to each patch and construct a histogram of patches for each test image. (For codes, please refer to the file called "part2_code.py")

Part 3:
I employ the random forest classifier to train the model based on this histogram of patches and evaluate its performance. I train a random forest classifier on the 6000 selected MNIST images with their patches, test this model with the test dataset of 10000 MNIST images with their patches, and yield an accuracy of 87.6%.

Part 4:

After thorough consideration and testing, I reach a conclusion that increasing the size of patches will allow the accuracy to enhance. I change the size of each patch to 35x35 and keep everything else the same, including the number of estimators and the maximum depth of the random forest classifier, and the number of centers. I derive the following result:

Size of patches	10x10	35x35
Accuracy	87.6%	91.2%

Part 5:

My random forest classifier model can have an accuracy of up to 91.2%. Compared with other machine learning classifiers, my classifier is around the same performance as the linear classifiers, but still has a lower accuracy than most of the other classifiers, including KNN, boosted stumps/trees, non-linear classifiers, SVMs, and neural nets. The high-ranking machine learning models are found here: http://yann.lecun.com/exdb/mnist/