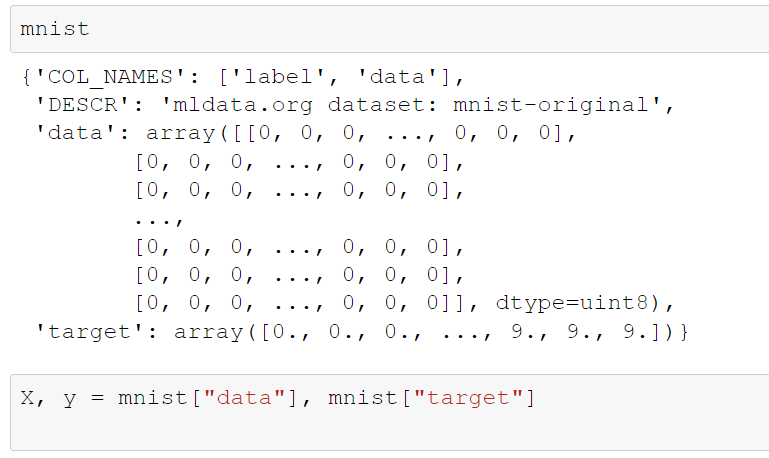
PCA-Dimensional Reduction

The popular MNIST data which is also called as “Hello World!” of Machine Learning is a data set of 70000 handwritten characters. This dataset is available in Scikit learn and can be fetched using the below code

>>from sklearn.datasets import fetch\_mldata #imports fetch\_mldata module from SKlearn library

>>mnist=fetch\_mldata("MNIST original") #Fetches the MNIST Dataset



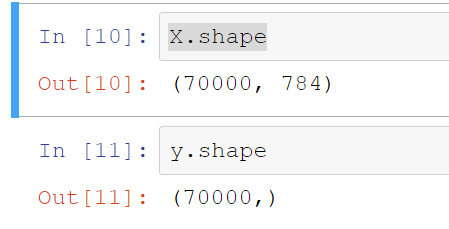
The first 60000 can be extracted as training set and the remaining 10000 can be used as testing set using the below code

>>X\_train, X\_test, y\_train,y\_test= X[:60000], X[60000:], y[:60000], y[60000:] #parse data separately

Then the dimensions of any data element can be checked using the below code

>>X.shape #shape of X

>>y.shape #shape of Y



We can visualize the non-decompressed or original data image by using the below python code using matplotlib package



As seen in the image mostly white part doesn’t have any information hence we can try to compress it by leaving some number of bits. To reduce the number of bits we have to lose only low variance information and contain 98% of the information available. So, we can use Principal Component Analysis technique to reduce the number of bits from 784 to 154 bits. I have used the below Scikit learn code to convert into 154 components

>>from sklearn.decomposition import PCA #imports PCA module from SKlearn package

>>pca = PCA(n\_components = 154) #performs PCA with 154 components

>>X\_mnist\_reduced = pca.fit\_transform(X) #fits the model with X and applies dimension redux

>>X\_mnist\_recovered=pca.inverse\_transform(X\_mnist\_reduced) #decompresses data with inverse transform



Clearly, we can decompress the data back by performing inverse transform, although it is not possible to regenerate the whole data back. Still we have successfully recovered most of the information.

Further we can leverage numpy function round to further reduce few more bits by rounding off

X\_mnist\_np=np.round(X\_mnist\_recovered \* 4, 0) / 4.0 #rounding off bits

