# **PROJECT: Build an MNIST Classifier**

The purpose of this document is to specify the requirements for the project “Build an MNIST Classifier.” Apart from specifying the functional and non-functional requirements for the project, it also serves as an input for project scoping.

**Project Objective**

The purpose of the project is to use one or more of the classification algorithms to train a model on the MNIST image database to detect images which contain the digit 5 or not.

**Project Description and Scope**

You are provided with the following resources that can be used as inputs for your model:

1. A collection of images of 70000 handwritten digits as part of the Scikit-Learn “datasets” module (one needs to just import it into code)

2. Code template containing these code blocks :

a. Import modules (part 1)

b. Plot functions (part 2)

c. Binary classifier using SGDClassifier class (part 3)

d. Predict and validate using cross validation (part 4)

e. Print Confusion Matrix, precision and recall (part 5)

You are expected to write the code for a binary classification model (after part 5 / cell no 108) using Python Scikit-Learn that trains on the data and calculates the accuracy score on the test data.

**Project Guidelines**

Begin by extracting the ipynb file.

Exercise 1 : Build a 5 vs not-5 binary classifier using KNN with no of neighbours as 4.

(hint : use Scikit-Learn libary KNeighboursClassifier)

(hint : look at the KNN tutorial taught earlier in the course)

Print accuracy score of the model

Note that it might take 10 minutes to print the accuracy score

Exercise 2 : Build a 5 vs not-5 binary classifier using Logistic Regression

(hint : use Scikit-Learn libary LogisticRegression)

(hint : look at the Logistic Regression tutorial taught earlier in the course)

Print confusion matrix, accuracy score and cross validation score (hint : see code above as to how we do this)

Note that it might take 10 minutes to print the accuracy score

Bonus Exercise : Build a 5 vs not-5 binary classifier using SVMs

(hint : look at the SVM tutorial taught earlier in the course)

Print accuracy score

Note that it might take 10 minutes to print the accuracy score

To make the processing faster, you could work with 10000 of the samples only, not the entire 70000 images. (change figures in Cell 91. )

(hint : X\_train, X\_test, y\_train, y\_test = X[30000:37000], X[37000:40000], y[30000:37000], y[37000:40000])

(hint : shuffle\_index = np.random.permutation(7000))

**Pre-requisites**

To execute this project you will need:

* Python 3.5
* Scikit-Learn library
* Jupyter notebook

**Programming hint for the student :-**

For ideas, look at tutorials accompanying various modules taught in the course.