PROJECT: Machine Learning

MNIST Classifier

Purpose of the Document

The document has 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.

Problem Statement

The purpose of the project is to train a model on the MNIST image database to detect images with 5 digits. This is done using one or more of the classification algorithms.

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

1. A collection of images of 70,000 handwritten digits is given as part of the Scikit-Learn "datasets" module. You need to import it into code.
2. Code template containing following 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 using Python Scikit-Learn that trains on data and calculates the accuracy score on the test data.

Project Guidelines

#	Exercises	Process
1	Initiation	Begin by extracting the ipynb file.
2	Exercise 1	Build a 5 vs not-5 binary classifier using KNN with number of neighbors as 4. (Hint: Use Scikit-Learn library KNeighboursClassifier) (Hint: Refer to 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.
3	Exercise 2	Build a 5 vs not-5 binary classifier using Logistic Regression. (Hint: Use Scikit-Learn library LogisticRegression) (Hint: Refer to the logistic regression tutorial taught earlier in the course) Print confusion matrix, accuracy score, and cross validation score. Note that it might take 10 minutes to print the accuracy score.
4	Exercise 3	Build a 5 vs not-5 binary classifier using SVMs (Hint: Refer to 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 only 10000 of the samples, not the entire 70000 images. (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))