CAP4770 Assignment 3

P1. Use the MNIST dataset and build a binary classifier to detect 3 versus 5. You can assume 3 is the positive class and 5 is the negatively class. Note that, for training, you only need a subset of the original training dataset, which are those corresponding to 3s and 5s. Similarly for testing, you will use the subset of the original test dataset that corresponds to 3s and 5s.

For this part, use the SGDClassifier with the default hyperparameters unless mentioned otherwise.

(a) Use cross_val_score() to show the accuracy of prediction under cross validation.

(b) Use cross_val_predict() to generate predictions on the training data. Then, generate the following:
   - The confusion matrix
   - The precision score
   - The recall score
   - The F_1 score

(c) Use cross_val_predict() to generate the prediction scores on the training set. Then, plot the precision and recall curves as functions of the threshold value.

(d) Based on the curves, what will be a sensible threshold value to choose? Generate predictions under the chosen threshold value. Evaluate the precision and recall scores using the predictions.

(e) Plot the ROC curve and evaluate the ROC AUC score.

(f) Try the RandomForestClassifier. Plot the ROC curve and evaluate the ROC AUC score.

(g) Repeat part (f) with feature scaling using StandardScaler().

P2. Build a multiclass classifier that distinguishes three classes: 3, 5, and others (i.e., neither 3 nor 5). Do this by train three binary classifiers: one that distinguishes between 3 and 5, one that distinguishes 3 and others, and one that distinguishes between 5 and others. Use the SGDClassifier for each the binary classifiers. For prediction, given the image of a digit, count the number of duels won as follows:

- Assume the digit is 3. Pass it to the 3-vs-5 classifier and 3-vs-others classifier, and count the number of wins by 3.
- Assume the digit is 5. Pass it to the 3-vs-5 classifier and 5-vs-others classifier, and count the number of wins by 5
- Assume the digit is 'others'. Pass it to the 3-vs-others classifier and 5-vs-others classifier, and count the number of wins by 'others'.

Whichever gives the most wins, the assumed digit will be the predicted class for the input. If there is tie, break the tie randomly.

P3. Use the KNeighborsClassifier, which has built-in support for multiclass classification, to classify all the 10 digits. Try to build a classifier that achieves over 97% accuracy on the test set. The KNeighborsClassifier works quite well for this task if you find the right hyperparameters. Use a grid search on the weights and n_neighbors hyperparameters.

Once you find a good set of hyperparameters, please conduct error analysis on the training dataset. In particular, find the confusion matrix, display it as an image using matshow(), and discuss the kinds of errors that your model makes.

Please submit a pdf file that contains your code and results, and the Jupyter Notebook if you use it.

Hint: You can build on the code that we discussed during the lectures, which can be downloaded from the github page: https://github.com/ageron/handson-ml2. Most of the code for P3 is related to Exercise 1.

P4. See the separate the PDF file ‘A4-P4.pdf’.