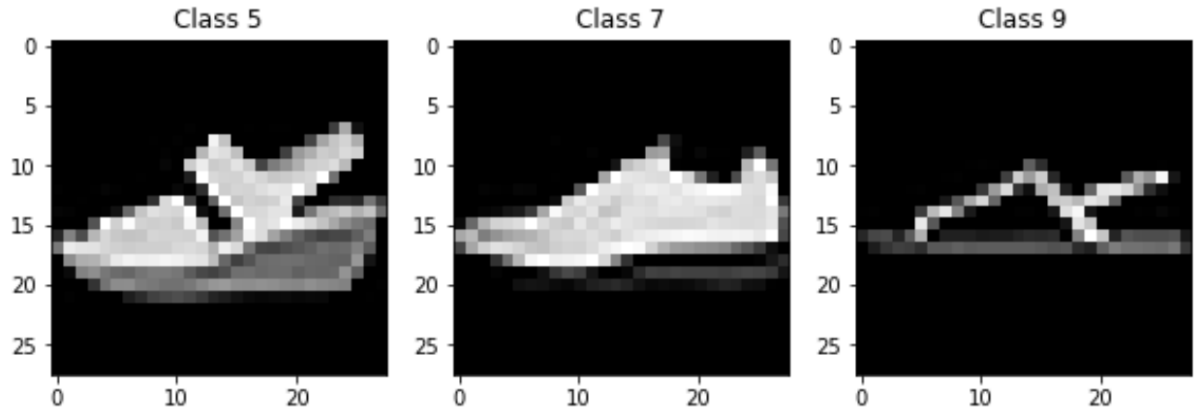
Assignment 2

Task 1



The data is read into a dataframe using pandas. Sandals, sneakers and ankle boots are selected within data while other data is ignored. Labels and data are split into two different variables. The first image for each label required is selected, reshaped and added to the diagram.

Task 2

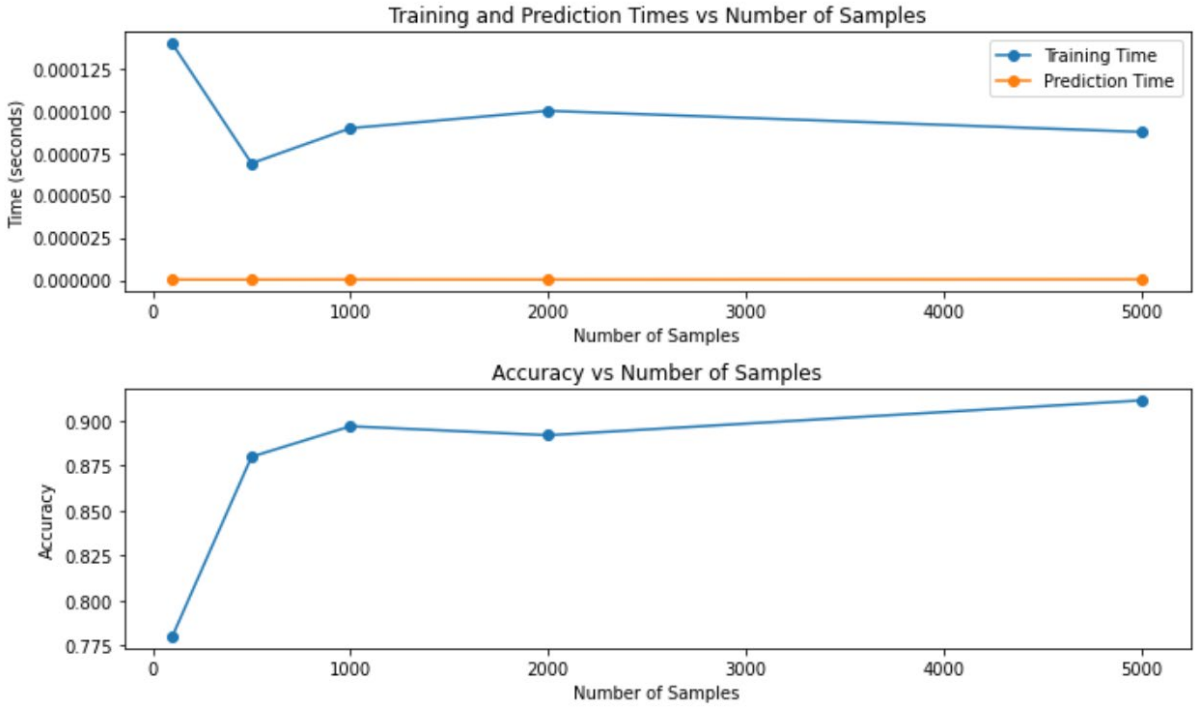
This function is the foundation of all other tasks in this assignment. It takes five parameters: the classifier to be tested, the feature vector data, the labels, the number of k-folds, and the number of samples to be tested.

Firstly, the KFold function is called and an instance stored in kf. Arrays are then defined for training times, prediction times, accuracy scores and confusion matrices.

Next, each fold is evaluated for the defined number of samples. The classifier is trained, predictions made on the test set, and the confusion matrix and accuracy calculated. The values of training time, prediction time, accuracy score, and confusion matrix are appended to the previously defined arrays for each fold.

Max, min, and mean values as well as all confusion matrices are then returned from the function in the form of a tuple. Only the first three, mean train time, mean predict time, and mean accuracy will be relevant for the tasks going forward.

Task 3



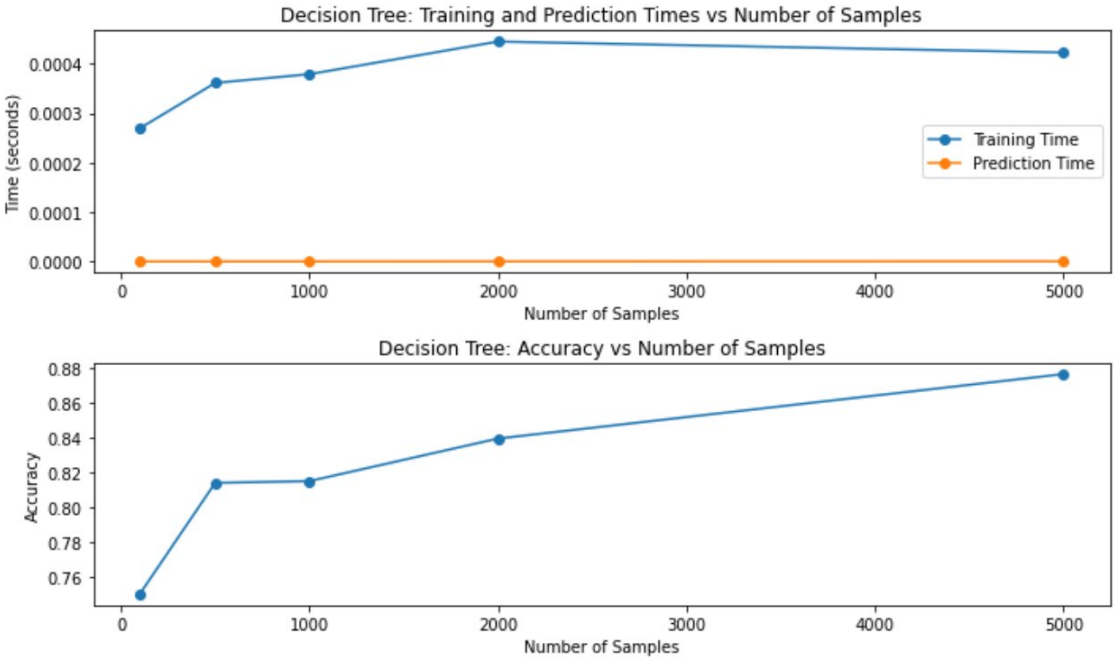
Task 3 tests the function written in task 2 with the perceptron classifier imported from sklearn.

Perceptron classifier is initialized, sample sizes are defined, and arrays are created to store mean values for different sample sizes.

For each sample size, the task2 function is called with k=5 number of folds. The results tuple is obtained with only relevant data which is appended to its correct array for each sample size.

The data of these arrays are then visualized in diagrams comparing training and prediction times with number of samples, as well as the accuracy vs number of samples.

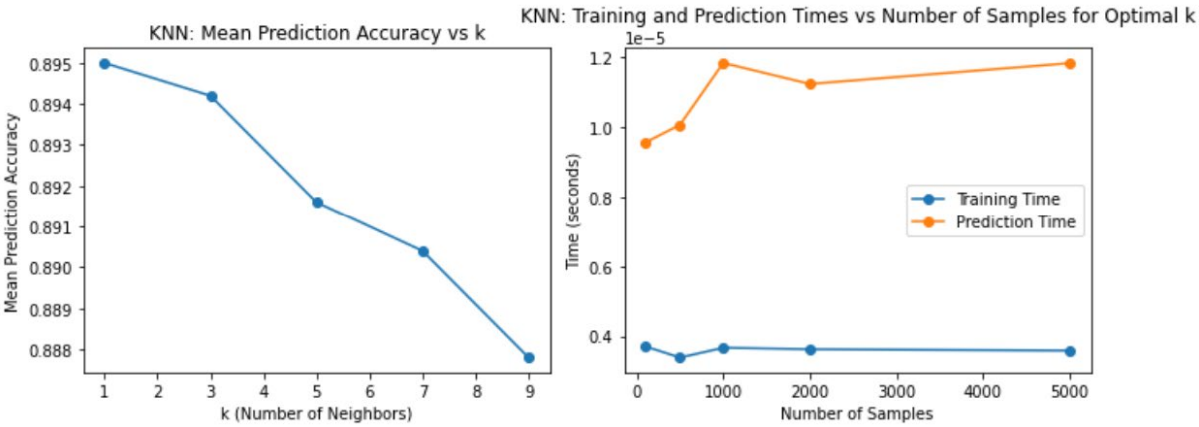
Task 4



Task 4 is identical to task3 in terms of methodology, only swapping the Perceptron classifier for the Decision Tree classifier.

The data is processed and displayed in exactly the same way, updating only the labels and titles of each graph.

Task 5

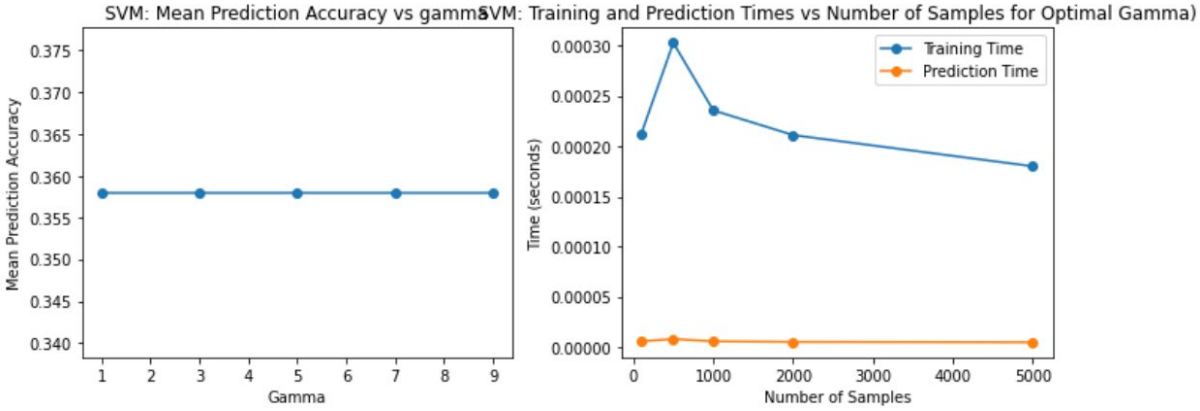


Task 5 adds complexity compared to the previous two by allowing for different values of k in its processing. An additional ‘for’ loop is added to initialize the classifier with a different value of k neighbors each time. This adds significant processing time compared to the previous methods.

The data storage is similar here, however the data required for graph plotting is only that of the optimal k value. The index of this optimal k value is identified using numpy.argmax()

function. That index is subsequently used to identify the data in the predefined arrays which should be used to plot the graphs.

Task 6



The process for task 6 is identical to that of task5, only this time the SVM classifier is used with the RBF kernel and the variable with different values being tested is 'gamma'.

The rest of the process for this task is identical to the previous one. One notable difference however is execution times. I tested this on my laptop and on a significantly more powerful machine and in both instances Task6 took significantly longer to execute than Task5.

Task 7

Training times for the Perceptron classifier are extremely low, as are the prediction times. As a linear classifier it doesn't tend to be heavy on resources. In this case the Decision Tree training and prediction times are similar to those of the Perceptron, however, this could easily increase with a larger and more complex tree. The k-Nearest Neighbours classifier has low training time as it memorises the dataset. The prediction however can be relatively high due to its finding of the nearest neighbours during prediction. The Support Vector Machine with the RBF kernel has by far the highest training and prediction times of the four classifiers. The added complexity should result in more accurate predictions, however, in my testing the accuracy rate was shockingly low.

My ranking of the classifiers is as follows:

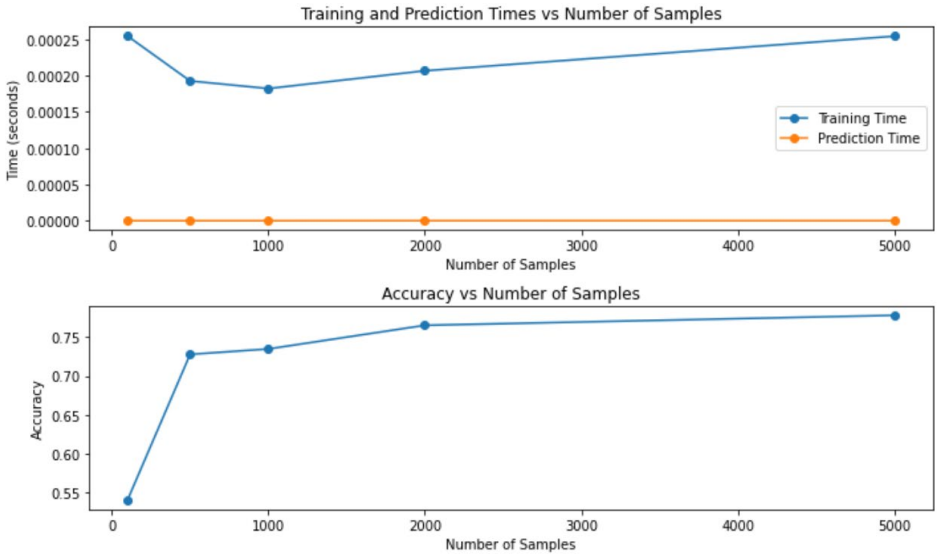
1. KNN – While training and prediction times are slightly higher than Perceptron and Decision Tree, it was the most consistently accurate classifier in my testing while still not having that long in training, prediction or execution times.
2. Perceptron – A simple but effective classifier. Low training and prediction times with accurate results.
3. Decision Tree – Able to handle more complex data while having low training and prediction times with accurate results.

4. SVM – I'm sure I must have made some mistake in my testing of this classifier because the results were abysmal. It took significantly longer than all other classifiers combined to train, predict, and execute. And when it had finally completed, the accuracy results were shocking, averaging at just over 30%.

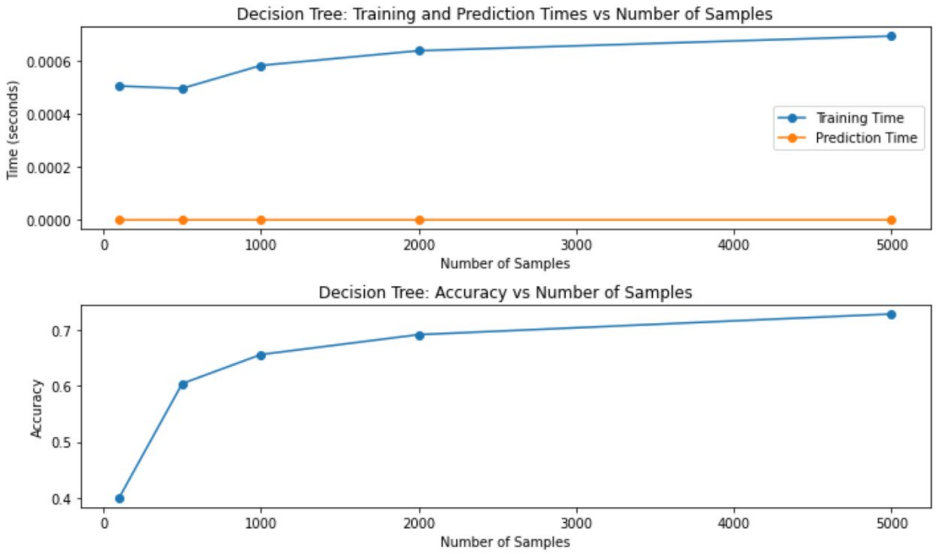
Sidenote

The graph above shows tests carried out on three labels of nine in the dataset. I also ran them using all labels. The results showed dips in the accuracy of all classifiers, however, the only classifier not to show a dramatic decline and retain a decent accuracy store is the k-Nearest Neighbours classifier.

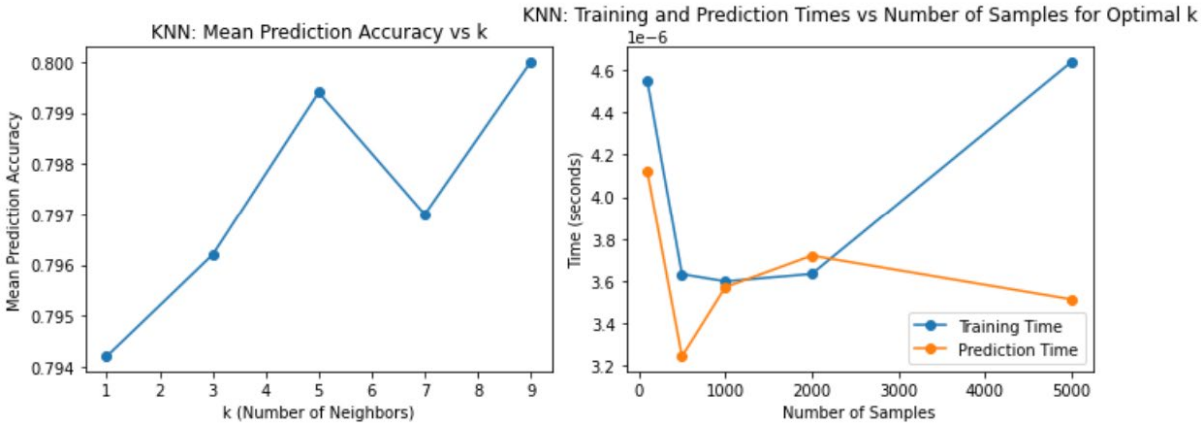
Perceptron:



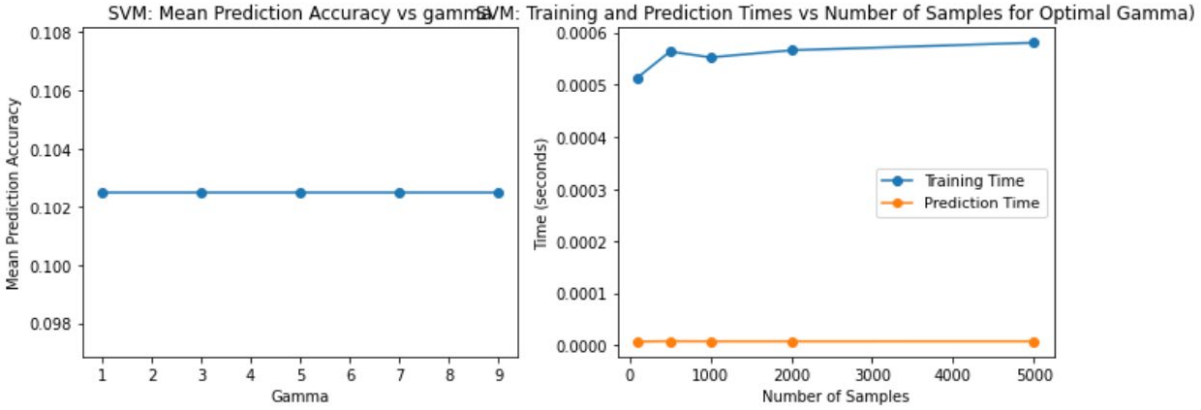
Decision Tree:



KNN:


KNN: Mean Prediction Accuracy vs k
Mean Prediction Accuracy
k (Number of Neighbors)
KNN: Training and Prediction Times vs Number of Samples for Optimal k
1e−6
Time (seconds)
Number of Samples
Training Time
Prediction Time

SVM:


SVM: Mean Prediction Accuracy vs gamma
Mean Prediction Accuracy
Gamma
SVM: Training and Prediction Times vs Number of Samples for Optimal Gamma)
Time (seconds)
Number of Samples
Training Time
Prediction Time