Task 1

Inside the split() function I take the dataframe generated from the csv file and from it I create a new dataframe using only the labels column (Task 1.1, Line 17), I then create a second dataframe using everything from the original dataframe but the labels column (Task 1.2, Line 18), meaning I now have two dataframes, one with labels and one with feature vectors. I then further split the labels column to be either 1 for boot or 0 for sneaker, I then count the length of both of these dataframes to get the count of how many boots and sneakers are in the dataset. (Task 1.3, Lines 20→26). Finally, I use imshow() to draw one boot and one sneaker (Task 1.4, Lines 28→34).

Chart, histogram

Description automatically generatedChart, histogram

Description automatically generated

Task 2

In evaluate() I split the data using kfold selection (Task 2.1 & 2.2, Line 47), I also parameterise the sample size for future use (Task 2.3, Line 135). I measure the time taken for each split in the training (Task 2.4, Lines 79→83) and testing (Task 2.5, Lines 85→89) times. I then determine the confusion matrix for this fold (Task 2.6, Lines 97→102). Finally, I determine the min, max, and mean time taken for training per sample (Task 2.7, Lines 118→121), the min, max, and mean time taken for testing per sample (Task 2.8, Lines 124→126), and the min, max, and mean accuracy of the prediction (Task 2.9, Lines 110→112).

Task 3

First, I ran the code from task 2 using the Perceptron classifier (Task 3.1, Lines 146 & 147), splitting the data, training the classifier, using the classifier to predict, and then outputting the statistics for the run such as training time, testing time, and accuracy, doing all this for each fold in the cross-validation procedure. Next, I output the mean of the average accuracies that were collected for each of the folds (Task 3.2, Line 150). Finally, I re-ran the code from task 2 4 times, changing the sample size with each run to be 1000, 2500, 5000, and 7000. After each run, I also added the overall time the classifier took to both train and predict to a list. After the 4 runs, I graphed the results of sample size vs time for classifier to train and predict (Task 3.3, Lines 151 →164).

Chart, bar chart

Description automatically generated

Task 4

First, I defined a list of gammas to use with the radial basis kernel for SVM. I ran the code from task 2 for each gamma in the list, using the SVM classifier with the gamma parameter set to be the current element at the index of the gammas list. (Task 4.1 & 4.2, Lines 167 →171). After looping through each of the gammas, I print the mean of the mean accuracies of each of the gammas, and then picked the largest mean accuracy to determine the best gamma (Task 4.3 & 4.4, Lines 173 →178). Finally, I re-ran the code from task 2 4 times using the gamma with the greatest mean accuracy as a parameter, changing the sample size with each run to be 1000, 2500, 5000, and 7000. After each run, I also added the overall time the classifier took to both train and predict to a list. After the 4 runs, I graphed the results of sample size vs time for classifier to train and predict (Task 3.5, Lines 179 →193).

Chart, bar chart

Description automatically generated

Task 5

Chart, bar chart

Description automatically generatedFirst, I ran a range loop passing the current loop count for KNN’s k parameter. I ran the code from task 2 using k in the range 1, 5 as the KNN classifier with the k parameter. (Task 5.1 & 5.2, Lines 197 →199). After running through the loop, I print which k achieved the best mean accuracy for all the folds (Task 5.3, Line 201), the mean that said k achieved (Task 5.4, Line 201), as well as the mean of the means. Finally, I re-ran the code from task 2 4 times using the k figure with the greatest mean accuracy as a parameter, changing the sample size with each run to be 1000, 2500, 5000, and 7000. After each run, I also added the overall time the classifier took to both train and predict to a list. After the 4 runs, I graphed the results of sample size vs time for classifier to train and predict (Task 4.5, Lines 203 →218).

Task 6

First, I ran the code from task 2 using the Decision Tree classifier (Task 3.1, Lines 221 & 222), splitting the data, training the classifier, using the classifier to predict, and then outputting the statistics for the run such as training time, testing time, and accuracy, doing all this for each fold in the cross-validation procedure. Next, I output the mean of the average accuracies that were collected for each of the folds (Task 5.2, Line 225). Finally, I re-ran the code from task 2 4 times, changing the sample size with each run to be 1000, 2500, 5000, and 7000. After each run, I also added the overall time the classifier took to both train and predict to a list. After the 4 runs, I graphed the results of sample size vs time for classifier to train and predict (Task 5.3, Lines 226 →239).

Chart, bar chart

Description automatically generated

Task 7

The common trend between all the classifiers is that as the sample size increases, the time taken for training, testing or both increases. In both Perceptron and Decision Tree the time taken for training increases greatly with each increase in sample size, where testing stays relatively the same. In the case of KNN it is actually the reverse, where the Testing time increases greatly with each step up in sample size and the training times stay relatively the same. Finally, SVM shows the greatest time increase for both training and testing with each increase in sample size which does agree with the fact that SVM takes the longest of the 4 classifiers to run.

I would rank the classifiers like such:

1. KNN due to it having a high mean accuracy whilst still being relatively quick to process.
2. Perceptron due to the fact that it’s mean accuracy is close to the best and it is by far the quickest to process both training and testing.
3. Decision tree as again, it is close to the best mean accuracy, whilst remaining relatively fast compared to SVM.
4. SVM due to it having the longest processing times and poor accuracy.