CS57800: Statistical Machine Learning
Homework 1

Name: Mohammad Haseeb
Purdue ID: mhaseeb@purdue.edu

September 22, 2018

1 4-Fold Cross Validation

I have implemented a separate class called *k_folds_generator* that takes as input the path to the data set and the number of folds to be generated and then returns all the folds. I use this class to generate the folds and then use the *tune_hyper_parameters* functions in the respective scripts for each of the *decisiontree.py* and *knn.py* to tune the hyper-parameters.

The idea behind my 4-fold cross validation is that I first generate 4 folds of the data using the *k_folds_generator*. I then iterate through *i=0...3* and in each iteration, I select the test data as $fold_i$ and the training data as $fold_{0..i-1} + fold_{i+1..i}$. From the training data, I get a 20% chunk of the data which serves as my validation set. I tune the parameters for the algorithms using the average validation set scores for each of the set of hyper-parameters.

1.1 Dealing with Unknown Feature Values

KNN is specifically very sensitive to outliers and I noted in some of the data examples that for a certain feature, the values could deviate drastically. In order to prevent the effect of the outliers, I normalized (using feature's min/max value) all the features in the range of $[0 - 10]$. This improved accuracy for both KNN and Decision Tree.

2 Decision Tree

Best performing value of max-depth: 44

ID3 algorithm for making the decision tree is implemented in the file named *decisiontree.py*. The file has two major parts:

- Hyper Parameter Tuning

This part calls the *tune_hyper_parameters* function which gets the max-depth hyper-parameter.

- Model training, prediction, and performance calculation

Once I have found the best value for the max-depth, I initialize a Decision Tree object with this depth and train it. I then calculate and print the difference performance metrics using this learned tree.

Note: I submitted my code with the *tune_hyper_parameters* function commented out as it takes more than 5 minutes to run. The value with which I run the remaining program is obtained from the results of this function.

2.1 Performance Metrics

The following table shows the average F-score and accuracy values for different sets of the data.

Fold 1	F1 Score	Accuracy
Training	*Not Calculated*	98.537414966
Validation	*Not Calculated*	54.2234332425
Test	*Not Calculated*	53.3496732026

Fold 2	F1 Score	Accuracy
Training	*Not Calculated*	92.3469387755
Validation	*Not Calculated*	53.6784741144
Test	*Not Calculated*	54.4934640523

Fold 3	F1 Score	Accuracy
Training	*Not Calculated*	99.9659863946
Validation	*Not Calculated*	52.7247956403
Test	*Not Calculated*	55.4738562092

Fold 4	F1 Score	Accuracy
Training	*Not Calculated*	97.4472430225
Validation	*Not Calculated*	54.2234332425
Test	*Not Calculated*	52.6101141925

I could not correctly implement the F1 Score for a multi-class label, therefore, I am not reporting any results for that. The results that I was getting from my implementation were very different from sklearn's results of the macro-score.

I had expected the decision tree to give a better accuracy as compared to KNN since decision tree's actually provide some generalization. However, it seems from the results that because the data was small and was a bit skewed, the decision tree does not really work well. The label '6' for wine-quality was the most prevalent and just by doing a random guess we could have gotten approx. 20% accuracy.

2.2 Hyper-parameter Tuning

I tuned the max-depth hyper-parameter for the decision tree by iterating over range $[44 - 59]$ and then checking the average validation set accuracy for that depth and then returning the value of the max-depth as the one with maximum accuracy.

Figure 1 shows the results for this. Note that I had sorted my results according to maximum score which is why we are seeing this graph. Note that the maximum depth of this tree after the ID3 algorithm terminated was 54, which explains why the accuracy for $depth = 59$ is same as $depth = 44$. Moreover, as expected, for a depth ≤ 36, the accuracy of the decision tree kept dropping as we under-fit the data.

3 KNN

Best performing value set of k and distance metric: k = 1 and distance metric = Manhattan Distance

The code for making the implementing the KNN algorithm is in the file named *knn.py.* The file has two major parts:

- Hyper Parameter Tuning

This part calls the *tune_hyper_parameters* function which gets the best set of k and distance metric hyper-parameters.

- Model training, prediction, and performance calculation

Once I have found the best set of hyper-parameters, I initialize a KNN object with these parameters and train it. I then calculate and print the difference performance metrics.

Note: I submitted my code with the *tune_hyper_parameters* function commented out as it takes more than 5 minutes to run. The value with which I run the remaining program is obtained from the results of this function.

3.1 Performance Metrics

The following table shows the average F-score and accuracy values for different sets of the data. This average is over all the 4 folds.

Fold 1	F1 Score	Accuracy
Validation	*Not Calculated*	60.6267029973
Test	*Not Calculated*	59.5588235294

Fold 2	F1 Score	Accuracy
Validation	*Not Calculated*	61.8528610354
Test	*Not Calculated*	58.5784313725

Fold 3	F1 Score	Accuracy
Validation	*Not Calculated*	55.9945504087
Test	*Not Calculated*	61.5196078431

Fold 4	F1 Score	Accuracy
Validation	*Not Calculated*	60.3542234332
Test	*Not Calculated*	60.277324633

I could not correctly implement the F1 Score for a multi-class label, therefore, I am not reporting any results for that. The results that I was getting from my implementation were very different from sklearn's results of the macro-score.

3.2 Hyper-parameter Tuning

I tuned the hyper-parameters by running KNN's on validation set for every pair of (k,distance_metric) where was in the range (1,79) and distance metric was either of Euclidean Distance, Manhattan Distance or Cosine Similarity. As can be seen in figures 2, 3, and 4, the best value of k was 1 and the best distance metric was Manhattan Distance. A value of 1 performed better than more k's as the data was small and contained a lot of values corresponding to the label '6'.

4 Questions

1. Depends on the number of features. If the number of features are large and the ID3 algorithm returns before that then the depth of the learned tree is less than the maximum-depth and then no matter what the number of features, it will not effect the decision tree made. If the number of features are small then we will get a smaller tree that may or may not generalize well i.e. it will under fit.

2. The main difference between Decision Tree and KNN is that KNN does not actually learn anything, it simply maintains the data set. On the other hand, the Decision Tree actually attempts to learn a hypothesis and generalize based on the training data using a number of smart techniques like Information Gain per attribute. Another difference between the two is that with Decision Tree, we can actually tell the path taken by the algorithm to reach a certain prediction. Finally, KNN's are highly sensitive to outliers whereas Decision Trees can become less effective of these given enough data.

3. In order convert the decision from a classification to a ranking model, we can simply assign weights to different class labels before running our testing data. Now, everytime we get a label for a new example, we can see the assigned weight of that example and then give it a rank accordingly. We can also store these weight in each of the decision tree's nodes and decide on which split to take while predicting based off of that feature's weight.

5 Additional Works

- Decision Tree: Best Max-depth = 44 and Best Avg Test Accuracy = 53.98%
- KNN: Best K = 1 and Best Distance Metric = Manhattan Distance and Best Avg Test Accuracy = 59.98%

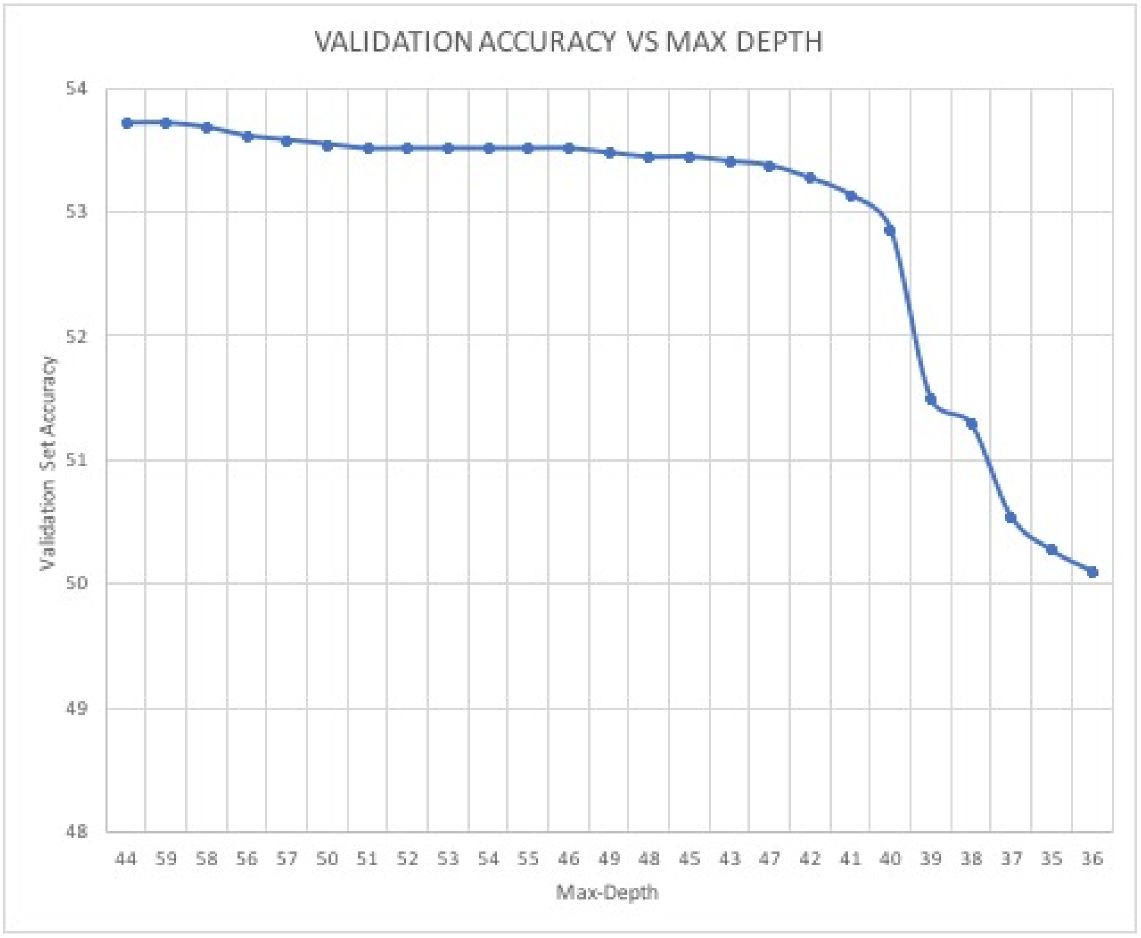

Figure 1: Decision Tree Max-Depth v.s. Validation Set Accuracy

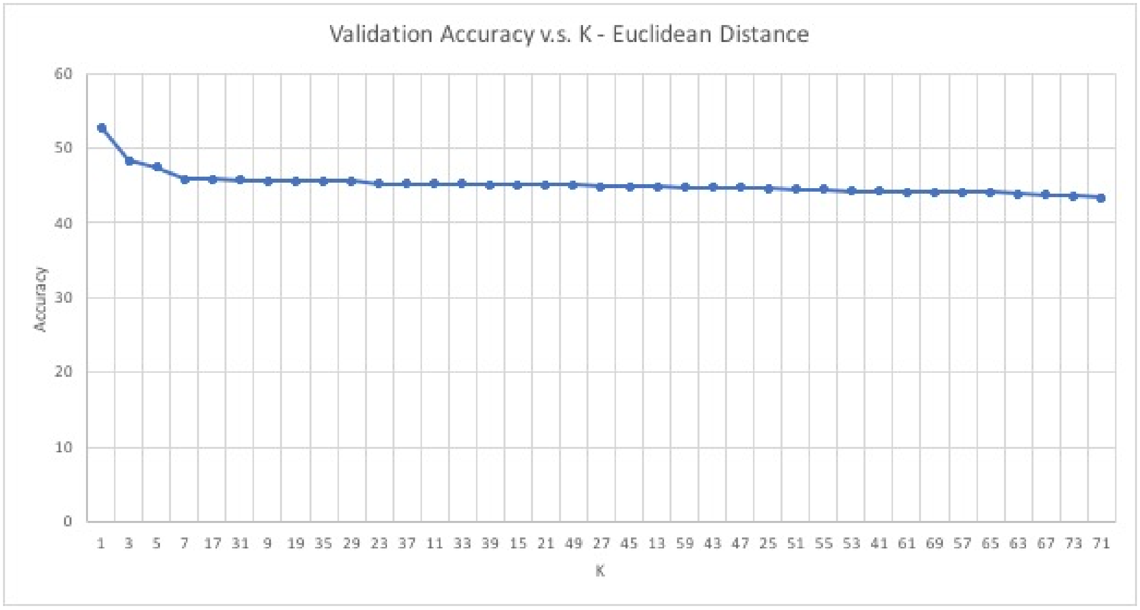

Figure 2: K v.s. Validation Set Accuracy with Euclidean Distance as distance metric

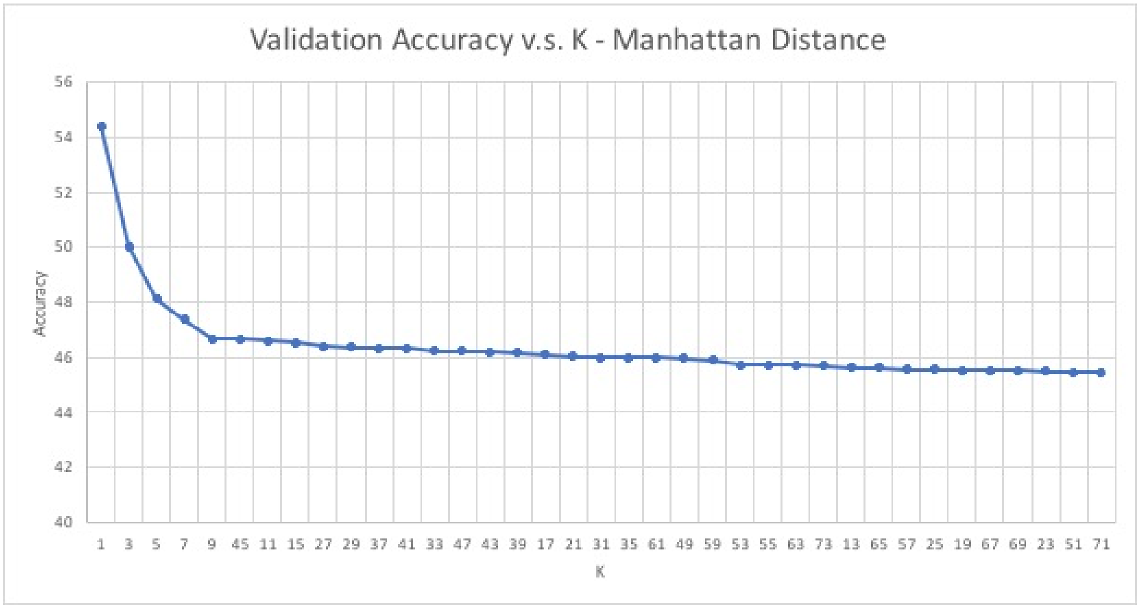

Figure 3: K v.s. Validation Set Accuracy with Manhattan Distance as distance metric

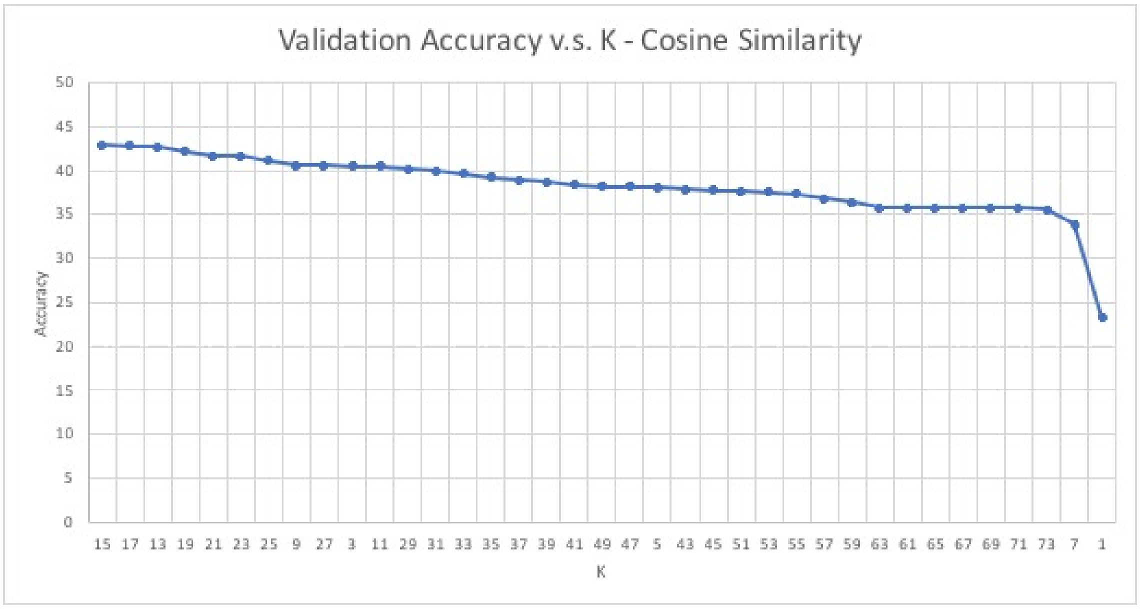

Figure 4: K v.s. Validation Set Accuracy with Cosine Similarity as distance metric