CS 334 - Homework 2

Alex Welsh

September 16th 2020

1 Question 1 - Decision Tree Implementation

a & b. The code for the Decision Tree Classifier (DTC) can be found in **dt.py**.

c. To see the accuracy landscape of how the criteria, max-depth, and min-leaf-sample, I created a Python notebook where i tested incremental chunks of the landscape so that in the event I lost the data, I wouldn't have to rerun the entire program from scratch. I copied the parameters and their accuracy's to **dt-model-accuracy.csv**. Next, I plotted two figures, one for the entropy criterion and one for the gini index criterion. The max-depth and min-leaf-samples make up the x and y axes and the accuracy is displayed on the z axis. The blue dots represent the training accuracy and the red dots represent testing accuracy using the test data set. What can be seen in the gini index graph is that smaller max depths and min leaf samples gave the highest accuracies. For the entropy calculations, a smaller min leaf sample was more important than the max depth as varying sizes of max depth gave high accuracies at low min leaf samples (Figure 1). The highest test accuracies were around .89 and .90 and the highest training accuracies were 1. The code for seeing the accuracy output of each model can be found in **q1.ipynb**. The code that makes the plots can be found at the bottom of **dt.py**.

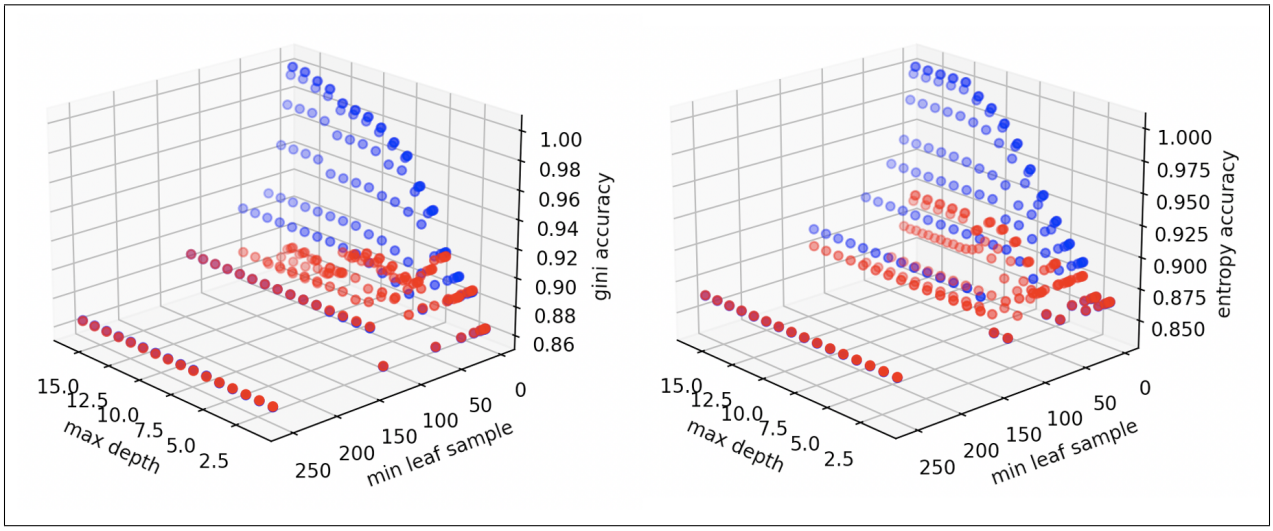

Figure 1. Effect of max depth and min leaf sample on gini and entropy training (blue) and test (red) accuracy.

d. The computational complexity of training the model is ndp. This is because when visualizing the tree that is formed, in each row of the tree (eg. first row has 1 node, 2nd row has 2 nodes, 3rd row has 4 nodes etc.) the split score is calculated for each sample for each feature. In the first node, you do nd calculations for the first split because you calcuate the split score of n samples and for each of their d features. Then, after the split, lets say the left node z samples and the right node would have $n - z$ samples. For both the left and right nodes, you calculate the split score for d features so the left node does zd calculations and the

right node does $(n-z)d$ calculations. Thus the total calculations per row is $zd+(n-z)d = nd$ calculations. For each "row" after, a similar idea is applied. Then, this is done for a total of p rows of the tree. Therefore, the final time complexity is ndk.

2 Question 2 - Exploring Model Assessment Strategies

a, b & c. The code that runs the models for holdout, K-fold Cross Validation (KFCV) and Monte-Carlo Cross Validation (MCCV) and reports their AUC's and run-time's can be found in **q2.py**.

d. For TrainAUC, all models have very high AUC values, around .94-.95 (Table 1). The 10-fold KFCV method seems to be the best when comparing it to the true test so that suggests that a bigger K-fold number would create a better model. This trend can also be seen as the 2 and 5-fold CV have lower validation AUC scores (Table 1). Holdout and MCCV seem to be worse in general than KFCV because of their lower AUC validation AUC scores. However, KFCV does seems to take longer for the 10-fold compared to the 10-fold MCCV (.077 vs .059) so if time is a concern for training more complex models, then choosing either KFCV or MCCV could be a factor (Table 1). It is also important to note that the holdout method was very, very fast so that is a benefit, though its validation AUC score is quite low. The code used to make Table 1 can be found in **q2.py**.

Table 1. Training, Testing and Run Time for Wine Data Set using Holdout, KFCV, and MCCV

Strategy	TrainAUC	ValAUC	Time
Holdout	0.9473319473319473	0.674181976445142	0.007349967956542969
2-fold	0.9527877063915793	0.7445940450658057	0.013366222381591797
5-fold	0.9573506753703059	0.7649418021560765	0.03730201721191406
10-fold	0.9559185882157399	0.7990751400920424	0.07763195037841797
MCCV w/ 5	0.9452108300017272	0.7444024020638642	0.030356884002685547
MCCV w/ 10	0.9432760834728257	0.7512471241388698	0.05940890312194824
True Test	0.9525016328307844	0.8042261353104726	0.0

3 Question 3 - Robustness of Decision Trees and K-NN

a. The optimal parameters for KNN and Decision Tree Classifiers (DTC) are as follows. For the KNN, the optimal K value was 21. For the DTC, the optimal parameters were criterion=entropy, max-depth=4, and min-leaf-sample=8. These parameters were used to train the 4 KNNs and 4 DTCs. I chose K=21 because i did a GridSearch for K=1 to K=41 and K=21 had the highest score. I also did a GridSearch for the DTC. I used the parameters criterion={'gini', 'entropy'}, max-depth={1 to 21}, and min-leaf-sample={1, 2, 4, 8, 16, 32, 64, 128, 256, 512}. I chose the min-leaf-samples this way to get a broad range of values because I thought that small changes won't make that big of a difference. The code that looks for the optimal hyper parameters can be found in **q3.ipynb**.

Table 2. Accuracy and AUC of KNN Models with Different Amounts of Training Data Dropout

	Accuracy	AUC
KNN All	0.8604166666666667	0.5040778498609824
KNN 5%	0.8645833333333334	0.5129749768303985
KNN 10%	0.8666666666666667	0.5076923076923077
KNN 20%	0.8625	0.4987951807228916

Table 3. Accuracy and AUC of DTC Models with Different Amounts of Training Data Dropout

	Accuracy	AUC
DTC All	0.86875	0.5672845227062094
DTC 5%	0.8770833333333333	0.6175162187210379
DTC 10%	0.8854166666666666	0.6482854494902688
DTC 20%	0.875	0.577386468952734

b & c. The code that fits the complete and dropped data sets, trains the models, and calculates the accuracy and AUC can be found in **q3.ipynb**.

d. For the KNN models in Table 2, dropping out different amounts of training data doesn't seem to affect the accuracy or the AUC when given the test data. The accuracy stays around .86 and the AUC stays around .5. For the DTC however, there does seem to be a small increase in the accuracy when some data is dropped. The accuracy increases from around .86 to .87/.88 (Table 3). The AUC also improves when comparing the drop 5% and drop 10% of data models to the All model which was trained with all of the training data. The AUC for drop 5% and drop 10% were .61 and .64, respectively. These were both higher than the baseline All model which had an AUC of .56. This increase in AUC stops at the drop 20% model where the AUC decreases back to .57. This finding suggest that dropping a little bit of the training data may help, but dropping too much will impact the success of the model. The tables can also be found in **q3.ipynb**.