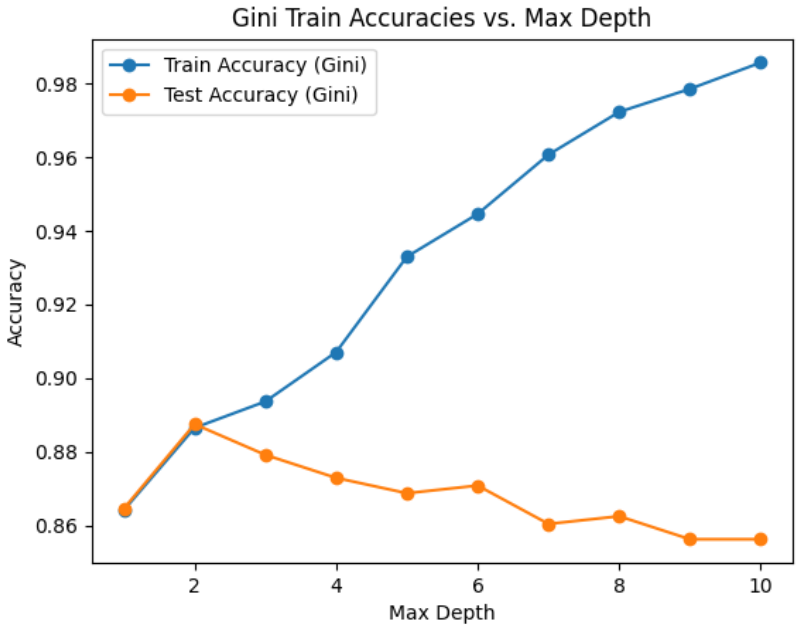
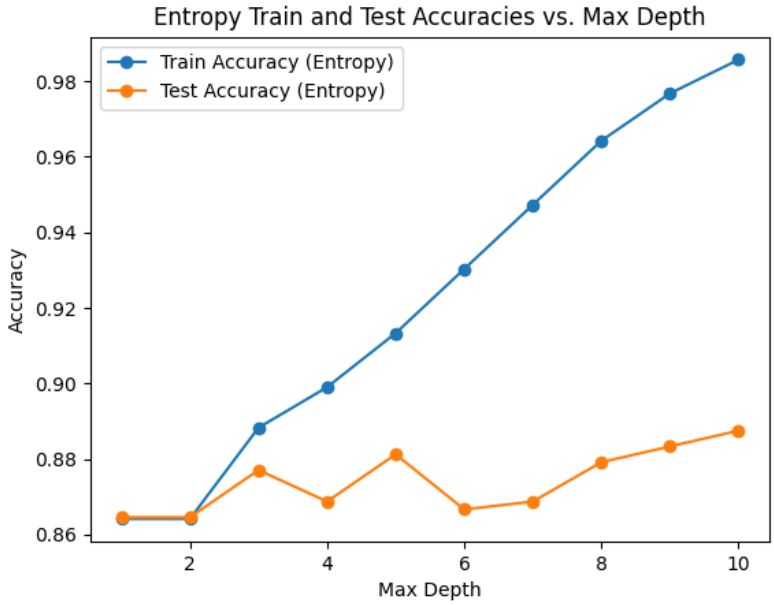
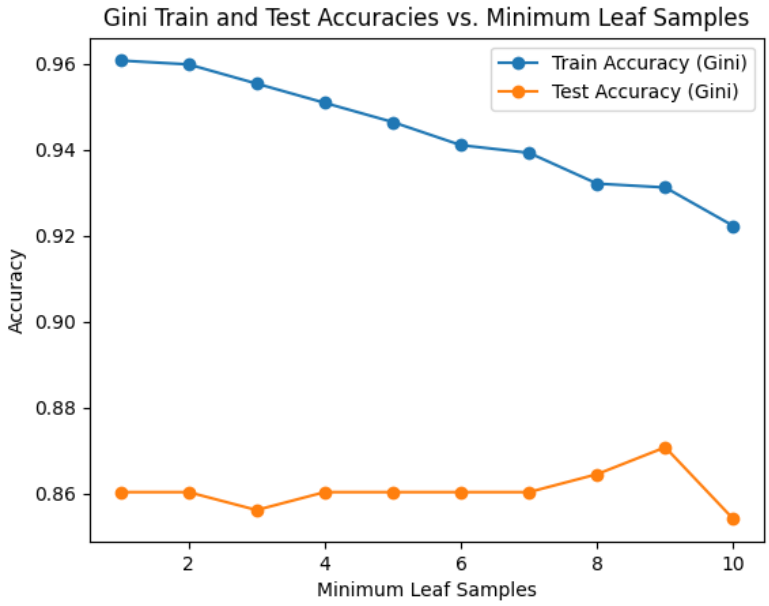
Dylan Parker
HW 2

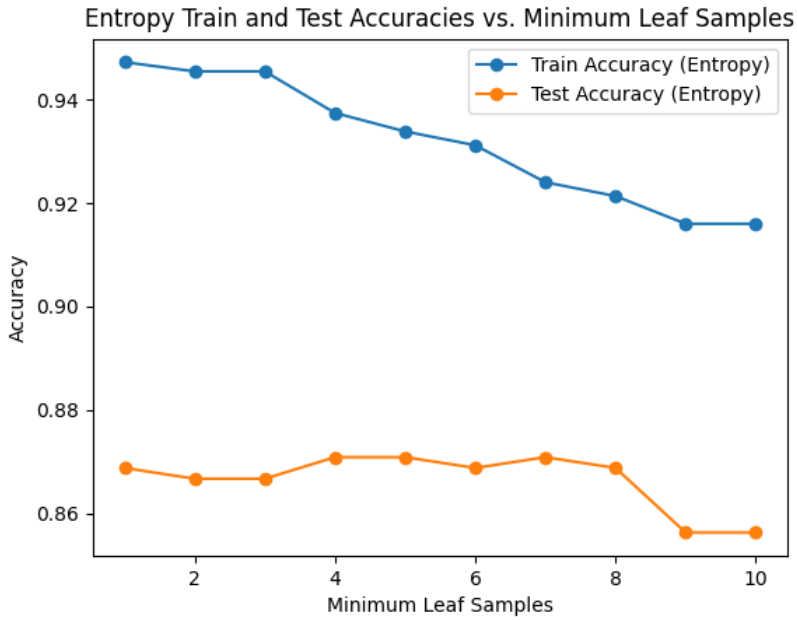
1d) As the max depth increases, the training accuracy when using the ‘gini’ or ‘entropy’ criteria also increases. Both start from an initial accuracy of 0.86, but by a max depth of 10 is reached the accuracy is around 0.98. However, the test accuracy when ‘gini’ is used also initially increases but later decreases after a max depth of 2 is reached. It starts with an accuracy of 0.86 at max depth=1, then peaks at 0.89 at max depth=2, and then decreases back to 0.86 by the max depth=10. On the other hand, the test accuracy when ‘entropy’ is used fluctuates a bit as the max depth grows larger, but the accuracy generally increases. It starts with an accuracy of 0.86 at max depth=1 and then finishes at an accuracy of 0.88 by the max depth=10.

As the minimum leaf samples (MLS) increase, the training accuracy when using the ‘gini’ or ‘entropy’ criteria decreases. Both start from an initial accuracy of above 0.94 at MLS=1, then decrease by at least 0.04 at MLS=10. However, the test accuracy when ‘gini’ is used stays relatively constant around 0.86 until it reaches 0.87 at MLS=9, then drops to 0.85 at MLS=10. On the other hand, the test accuracy when ‘entropy’ is used stays relatively constant around 0.87 until it reaches 0.85 from MLS=9 and 10.




Entropy Train and Test Accuracies vs. Max Depth
Train Accuracy (Entropy)
Test Accuracy (Entropy)
Accuracy
Max Depth
0.86
0.88
0.90
0.92
0.94
0.96
0.98
2
4
6
8
10


Gini Train and Test Accuracies vs. Minimum Leaf Samples
Train Accuracy (Gini)
Test Accuracy (Gini)
Accuracy
Minimum Leaf Samples
0.86
0.88
0.90
0.92
0.94
0.96
2
4
6
8
10



1e) The complexity of the train function recursively makes a decision tree. In terms of the training size (n), the number of features (d), and the maximum depth (p), the function evaluates every single possible split for each feature ‘d’, and then it iterates over each sample ‘n’ as a threshold to calculate the optimal ‘gini’ or ‘entropy’ score. This results in an O(d * n) time complexity, but at each threshold in order to calculate ‘gini’ or ‘entropy’, you have to separate the left and right splits for the ‘n’ samples to get an O(d * n^2) time complexity. However, This function continues for each left and right subtree up to the maximum depth ‘p’, so the total time complexity is O(p * d * n^2).

The predict function first iterates through each ‘n’ sample and then traverses through the whole tree up to the maximum depth ‘p’. This gives a time complexity of O(n *p).

2d) The accuracies when using Holdout, K-Fold Validation, or the Monte Carlo Cross Validations are above at least 94% accuracy for the training data. However, only the 10-Fold Cross Validation (The 5-Fold CV has the next highest training accuracy) returns a training accuracy higher than the True Test training accuracy. This could indicate that there is overfitting with the 10-Fold CV training data, which could negatively affect the 10-Fold CV test data accuracy. This is reflected in ValAUC, where the test accuracies of the 5-fold and 10-fold CVs are at around 78%, which is actually lower than the 2-fold CV, which is at 80% test accuracy. The time complexity for K-fold CV increased as each fold was added, with a 2-fold CV time of 0.005410 and a 10-fold time of 0.036884.

The MCCV with 5 vs. 10 also shows a similar trend, where the ValAUC for 10 is at 78% test accuracy while 5 is at 80% accuracy. The time complexity also increases as more splits are added from a time of 0.015902 at MCCV with 5, and a time of 0.031932 at MCCV with 10.

The Holdout technique took the shortest time of 0.004759, but it also gave the worst test accuracy 0f 0.730549.

Holdout validation is a simple and quick method for preliminary model assessment but may not provide a robust estimate of model performance. K-fold cross-validation reduces the impact of a single data split, making it suitable for general model evaluation and hyperparameter tuning. Monte Carlo cross-validation is valuable when you want to understand the variability in model performance across different data splits, particularly in cases of small or noisy datasets. From the test accuracies, the holdout was the least robust, while the k-fold CV and MCCV were more robust, though with different contexts one would be more preferable than the other to use.

	Strategy	TrainAUC	ValAUC	Time
0	Holdout	0.945685	0.730549	0.004759
1	2-fold	0.948012	0.800321	0.005410
2	5-fold	0.953358	0.780039	0.017133
3	10-fold	0.955193	0.786357	0.036884
4	MCCV w/ 5	0.952735	0.802054	0.015902
5	MCCV w/ 10	0.950025	0.783257	0.031932
6	True Test	0.954458	0.842632	0.000000

3a) I used k=5 for the k-Fold CV because it is a balance between 5 and 10 so it would be stable and not take too much time. Using k=5, it returned that:
Optimal KNN parameters: [n_neighbors: 4]
Optimal Decision Tree parameters: [criterion: entropy, max_depth: 3, min_samples_leaf: 1, min_samples_split: 2]

3b) Using the optimized hyperparameters for KNN I found that the AUC and ACC decreased as more of the dataset was removed. The AUC with the full dataset was 0.773, and after 20% was removed the AUC was 0.756. The ACC with the full dataset was 0.875, and after 20% was removed the ACC was 0.8625.

3c) Using the optimized hyperparameters for the DT I found that the AUC and ACC decreased as more of the dataset was removed. The AUC with the full dataset was 0.849, and after 20% was removed the AUC was 0.835. The ACC with the full dataset was 0.879, and after 20% was removed the ACC was 0.875.

3d)

AUC for optimized tests

Test Type	KNN(n=3) AUC	5% Removed AUC	10% Removed AUC	20% Removed AUC
KNN(n=3)	0.7733086190917516	0.7802780352177943	0.7792585727525487	0.7564411492122336
DT('criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 1, 'min_samples_split': 2)	0.8488600556070436	0.8532530120481927	0.8238554216867747	0.8352363299351252

Accuracies for optimized tests

Test Type	KNN(n=3) ACC	5% Removed ACC	10% Removed ACC	20% Removed ACC
KNN(n=3)	0.875	0.875	0.86875	0.8625
DT('criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 1, 'min_samples_split': 2)	0.8791666666666667	0.8791666666666667	0.86875	0.875

After checking the two algorithms, I found that while the accuracies of KNN and DT decrease slightly and are similar values as the training set decreases in size, the AUCs of the two are much more different. The AUC of KNN ranges from 0.76-0.78 while the AUC of DT ranges from 0.82-0.85. The AUC of DT is significantly larger than the AUC of KNN, however, they both follow the same trends as the training data is removed. When 5% is removed, the AUC of both the DT and KNN increases, however as more data is removed the AUC for both the DT and KNN decrease. These fluctuations indicate that the accuracy is not affected heavily as the dataset decreases, however, there is more variety in the AUC for both DT and KNN.