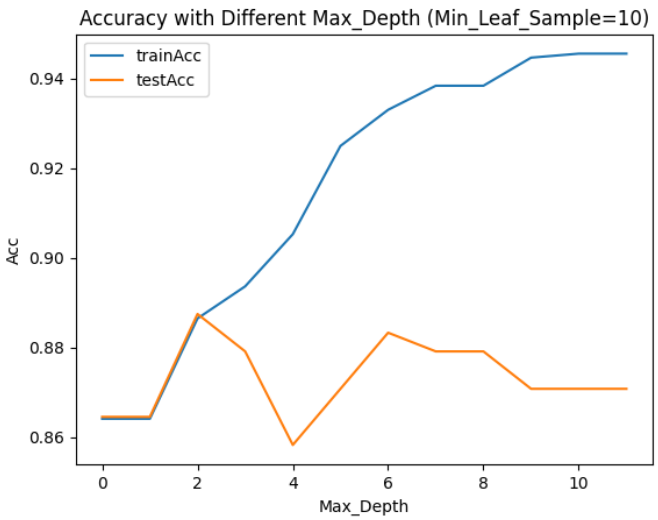
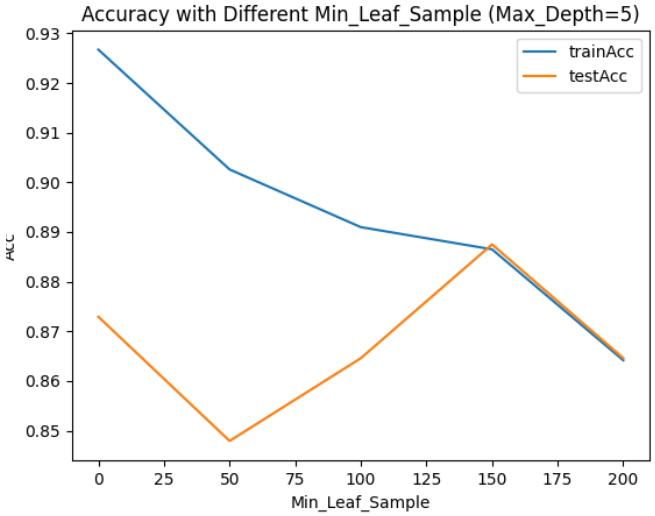
Q1

c)





d) Time complexity of the train function: *O(d * n * 2^p)*

Because the maximum depth is *p* and decision tree is a binary tree, the train function needs to construct *2^p* nodes maximum. In each node construction, there is a double loop that traverse all possible splits in all attributes to calculate the information gain or Gini index. Therefore, the time complexity of the train function is *O(d * n * 2^p)*.

Time complexity of the predict function: *O(n * p)*

For every sample predicted, the predict function would need to start from the root node and reach one of the leaf nodes, and compare the sample with every node traveled. At each level of the tree, one comparison is performed. If we assume the predicting size is also *n*, and since the maximum depth is *p*, the time complexity is *O(n * p)*.

Q2

d)

	Strategy	TrainAUC	ValAUC	Time
0	Holdout	0.932488	0.729671	0.004516
1	2-fold	0.955713	0.772213	0.006513
2	5-fold	0.952977	0.805180	0.021153
3	10-fold	0.954251	0.795174	0.045078
4	MCCV w/ 5	0.952036	0.734167	0.015069
5	MCCV w/ 10	0.945094	0.770655	0.028615
6	True Test	0.954458	0.840334	0.000000

From the table, we can see that all model assessment strategies, except the Holdout, yield Train AUC very close to the True Test's Train AUC. As for ValAUC, both k-fold and Monte Carlo cross validation show larger ValAUC than Holdout does. However, the True Test's ValAUC is larger than any other strategies. One thing noticeable here is that regardless of k-fold or Monte Carlo cross validation, increase of the k or s does necessarily result in the increase or decrease of the ValAUC. However, it does result in the increase of the Time taken to perform the assessment. When k and s is equal, Monte Carlo cross validation has a shorter assessment time than the k-fold cross validation does. In addition, the Holdout method has the minimum assessment time.

Q3

a) I used 5-fold cross validation, since 5-fold yields the highest ValAUC in the Q2.

Result of K-NN with different *K*:

K	AUC
1	0.677716
2	0.711388
3	0.739354
4	0.742082
5	0.752573
6	0.749524
7	0.749099
8	0.744104
9	0.749874
10	0.760000
11	0.757645
12	0.757849
13	0.757291
14	0.764350
15	0.759404

For K-NN: the optimal *k* is *k=14*.

Result of Decision Tree:

Max_Depth	AUC
1	0.678570
2	0.785775
3	0.820000
4	0.805977
5	0.800799
6	0.746771
7	0.739718
8	0.749599
9	0.734548
10	0.722412
11	0.706608
12	0.716070
13	0.733994
14	0.721036
15	0.702750

Figure 1 Different Max_Depth with Fixed Min_Sample_Leaf (1)

Min_Sample_Leaf	AUC
1	0.715609
25	0.830483
50	0.839103
75	0.821890
100	0.808831
125	0.792353
150	0.798362
175	0.782613
200	0.768268
225	0.756797
250	0.762358
275	0.748363
300	0.715134
325	0.737330
350	0.747496
375	0.747496
400	0.735213

Figure 2 Different Min_Sample_Leaf with no Max_Depth

For Decision Tree:

The optimal Max_Depth is *Max_Depth=3*, and the optimal Min_Sample_Leaf is *Min_Sample_Leaf=50.*

d)

model	Acc	AUC
knn_full	0.862500	0.518258
knn_95%	0.860417	0.510565
knn_90%	0.864583	0.519462
knn_80%	0.866667	0.514180
dt_full	0.887500	0.655978
dt_95%	0.887500	0.655978
dt_90%	0.887500	0.655978
dt_80%	0.885417	0.654773

From the table, we can see that both K-NN and Decision Tree are not very sensitive to the training datasets. As more data is removed from the training dataset, there is no big impact on both algorithms' accuracies and AUCs.