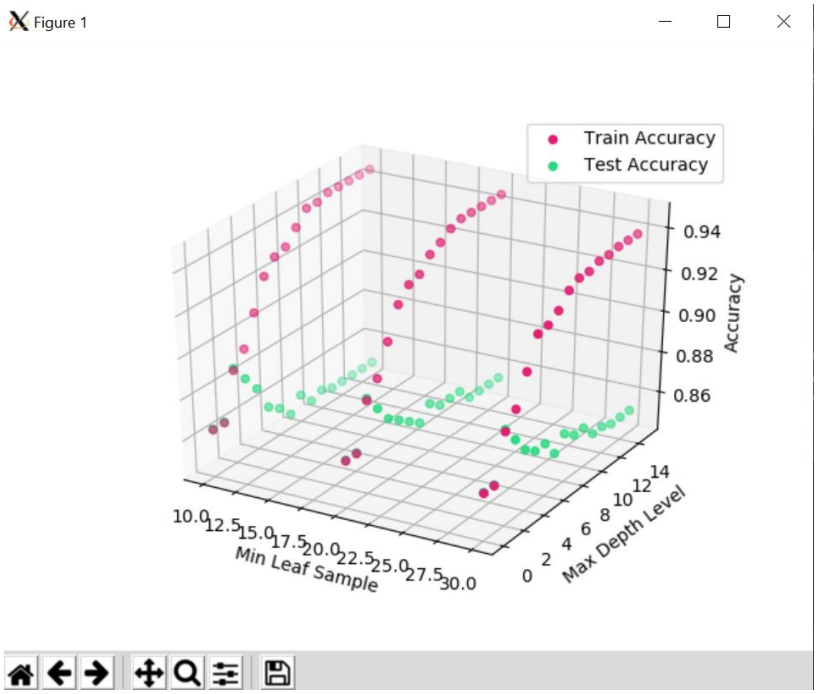
1A - 1B: See Code

1C:



1D:

Train Function:

D * (N^2) + N + log(p)

My function starts off by going through the the number features which is D times what is inside the loop. In the loop I loop through the number of rows which N rows, and inside that loop I split the data into node which goes through another N rows. This gives you the first part D * (N^2). In the next part I split the data again which has a runtime of N, after that I call my recursive function which takes log(p) where p is the max-depth

Predict Function:

log(p)

The predict function only has to go through the tree since all relevant information is stored at each node

2A - 2C: See code

2D: The holdout technique provide good values for Test Auc and the runtime wasn't too long, The robustness of the technique comes to question since it only partitions the dataset once. Compared to the other techniques the performance. In the K-fold technique the 2 fold does a little bit worse in terms of Test Auc but the runtime is more than half. As the fold start to increase the Test Auc starts to increase but when you transition from a 5-fold technique to 10-fold the

runtime doubles but provides a marginal increase in TestAuc. The Monte Carlo Validation was outperformed by the other techniques in terms of Test Auc but the trade-off came with runtime. The MC runtime was noticeably lower than the other techniques even when the number of folds increased from 5 to 10

3A: For all tests values SEE CODE: robust.py My choice for k-folds was 10, I decided to choose 10 folds because of the lower number of features so when the data was split into training and test data over the folds, there would be a good ratio of data to numbers features. So 10 folds will give a better representation compared to a 5 fold approach. The optimal parameters for knn was a k-value of 10 and for decision trees the optimal max depth is 5 and the optimal minimum leaf sample is 30

3B - 3C: See Code: robust.py

3D:
The Knn algorithm works well when trying to go for higher accuracy compared to Decision Trees, but loses out in Auc in comparison to Decision Tree

```
Optimal K Value 18
Optimal Max Depth 5
Optimal Min Leaf Sample 20
Optimal Type gini
------------
  Data Removed   Auc Knn   Acc Knn  Auc Tree  Acc Tree
0           0%  0.775792  0.866667  0.885598  0.775792
1         .01%  0.775644  0.866667  0.886080  0.775644
2         .05%  0.783577  0.864583  0.872011  0.783577
3          .1%  0.755848  0.862500  0.890806  0.755848
```