The Data

age: continuous.

workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.

fnlwgt: continuous.

education: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.

education-num: continuous.

marital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.

occupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.

relationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.

race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black. sex: Female, Male.

capital-gain: continuous.

capital-loss: continuous.

hours-per-week: continuous.

native-country: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.

income: 0 indicates income “<=50K” and 1 indicates income “>50K”

Training/ Test Split

Split the data into a training and test set after first shuffling the examples so that they are in a random order. For any training/test split, a good rule of thumb is 80% for the training set and 20% for the test set.

(a) Decision Tree

Keep all hyperparameters fixed to see how well the model work on the training set and test set.

Accuracy of train: 0.9998894325123697

Accuracy of test: 0.8106135986733002

The model performs better on the training set (99.99%) than on the test set (81.06%), this means that the model is likely overfitting.

We will now try to use different types of pruning methods (pre and post) to reduce the model's likelihood of overfitting as much as possible.

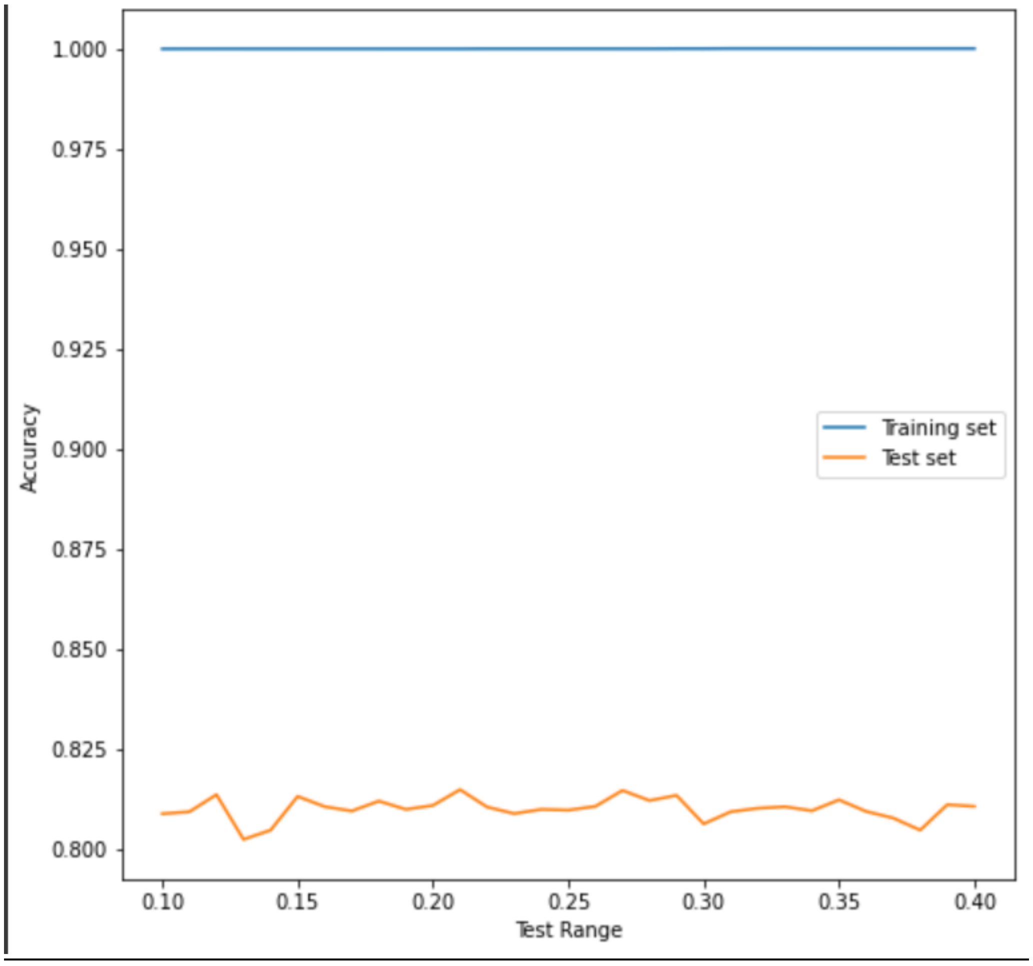
Pre pruning by way of hyperparameter tuning

The parameters to be controlled:

- Training set size
- Max depth
- Min sample split

The code for the last two were taken from an article on Kaggle. The code used to plot all the graphs was taken from this same article.

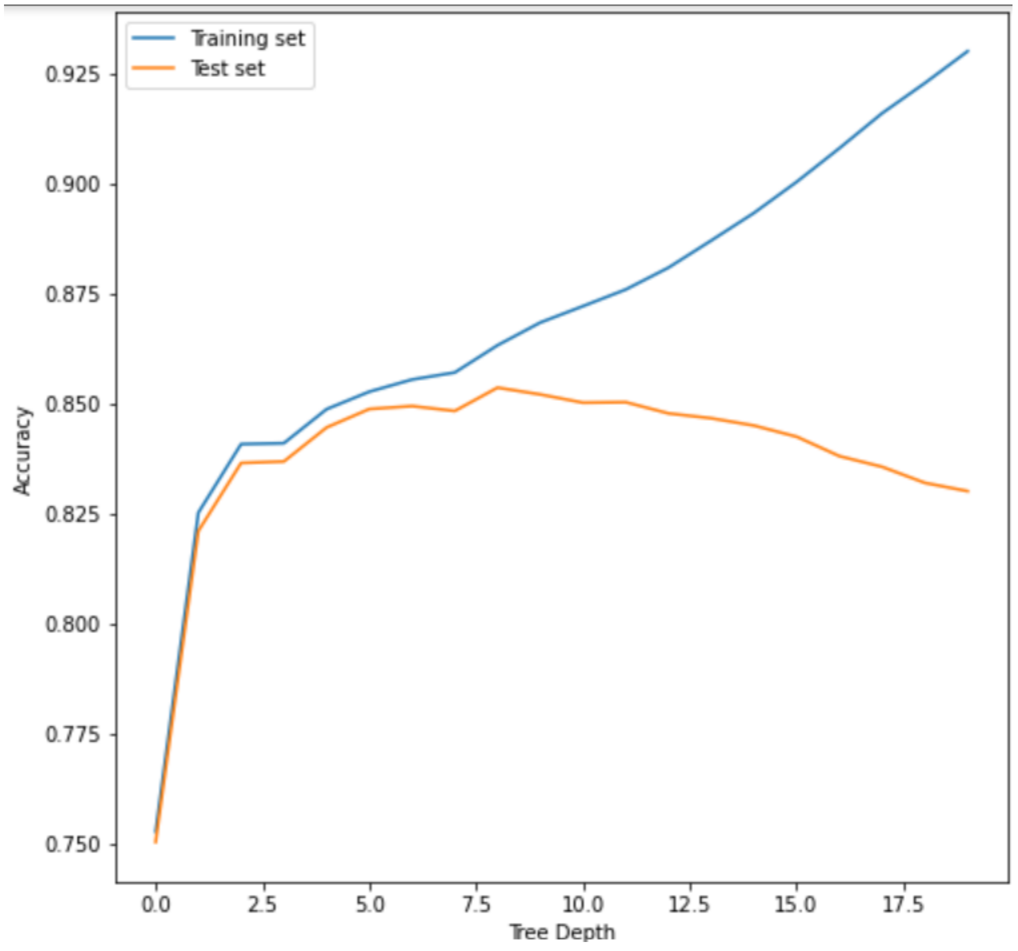
Training set size-



The training score is constant and close to 100%, while the validation score is much lower. A case of over-fitting. This is because the tree depth hasn't been limited so the model keeps creating new knobs until all the leaves are "pure".

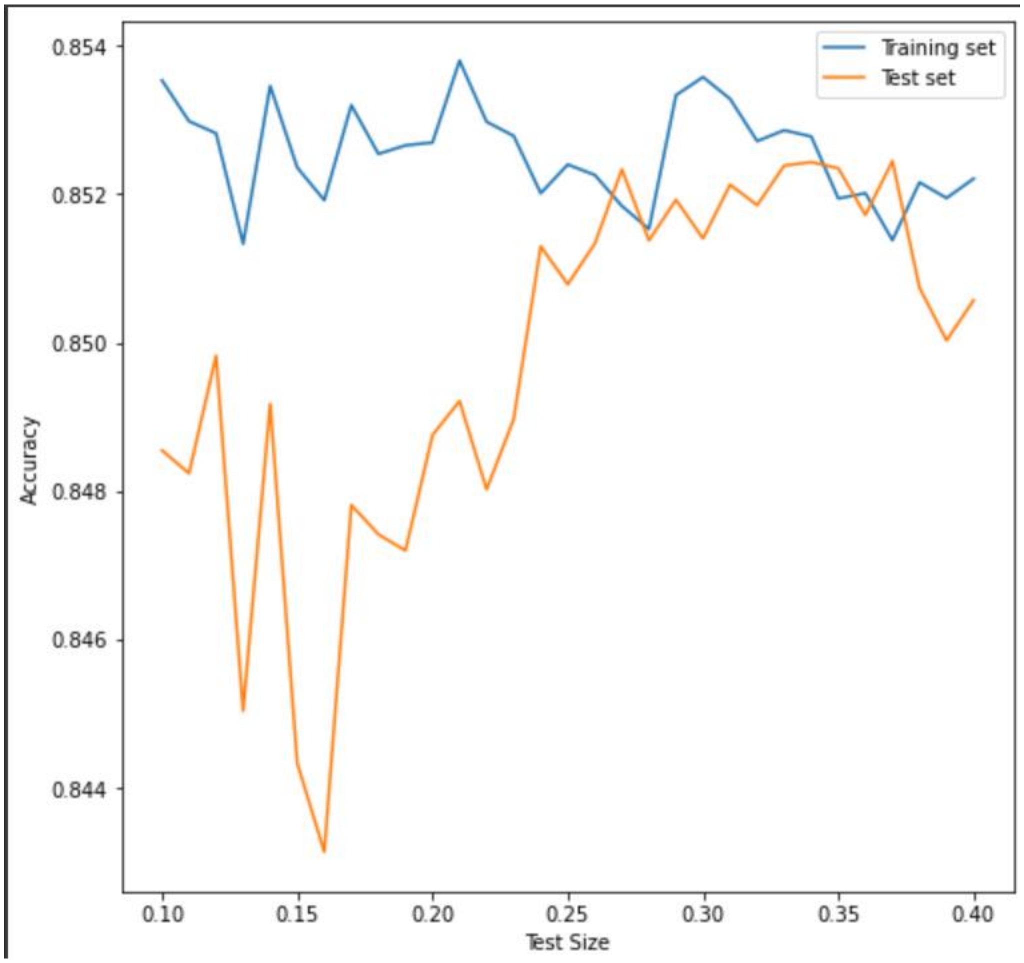
I will first find the max depth of the decision tree that gives the best possible outcome and then run the above code again.

Max Depth-



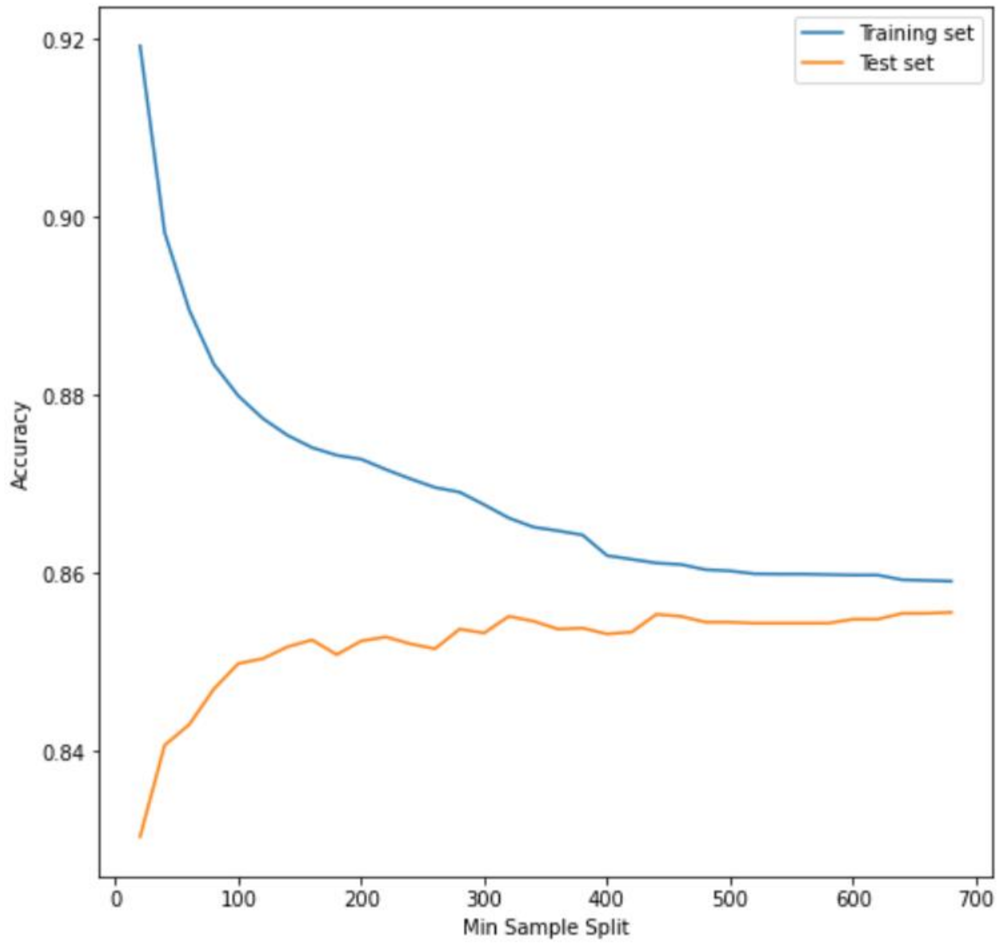
When max_depth is more than 6 (roughly) the model's likelihood of overfitting increases. The deeper the tree gets the model seems to overfit more.

Taking max depth as 6 running "Training set size" code again.



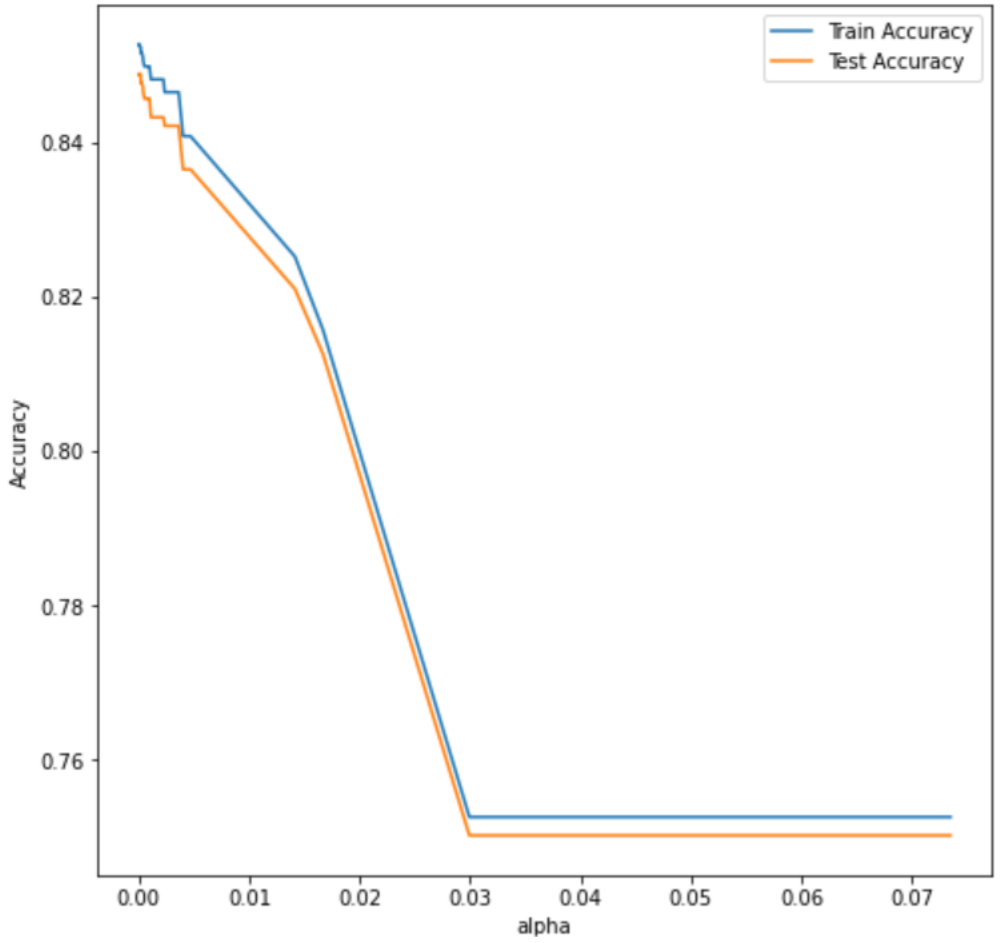
When test size is around 0.34 is when the model will achieve the highest accuracy scores while not overfitting. The model also seems less likely to overfit/underfit the data as the test size increases.

Min Sample Split-



The sample split is the minimum number of samples required to split an internal node. And even though the training accuracy decreases as we increase the sample split the chance of overfitting is significantly reduced. After sample_split>450 the likelihood of overfitting stays the same.

Post Pruning - Cost Complexity Pruning



There's little to no overfitting at any point but both the training and test accuracy seem to decrease as the value of alpha increase until around 0.03 and then the training and test accuracy stays constant (around 0.75) with a little bit of overfitting.

(b) Random Forest

Keep all hyperparameters fixed to see how well the model work on the training set and test set.

Accuracy of training set: 0.9998341487685546

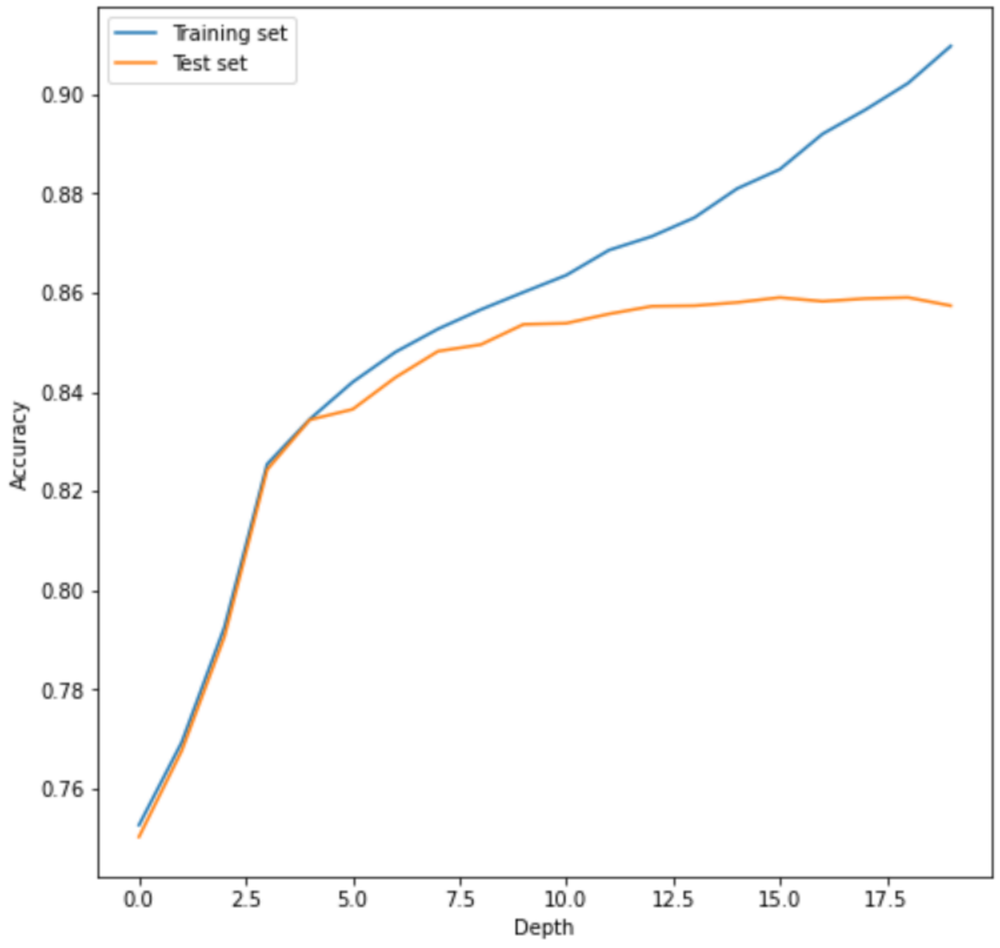
Accuracy of test set: 0.849640685461581

Training set is overfitting by 0.15 which is concerning.

Hyperparameter Tuning

- Max depth
- Random features - max_features
- Number of trees - n_estimators
- Training set size

Max depth



Accuracy of training set (**max_depth=5**): 0.8345081128894049

Accuracy of test set (**max_depth=5**): 0.834383637368712

Accuracy of training set (**max_depth=6**): 0.8419714183044475

Accuracy of test set (**max_depth=6**): 0.8364842454394693

Until the max depth is about 5 the training test error are very similar but after that the model soon starts to overfit.

For the rest of the hyperparameter tuning code the max_depth will be set to 5 because otherwise the training accuracy will always be close to 1.

Random features - max features

RandomForest3=RandomForestClassifier(max_features=**'sqrt'**, random_state=40, max_depth = 5)

Accuracy of train: 0.8345081128894049

Accuracy of test: 0.834383637368712

RandomForest4=RandomForestClassifier(max_features=**'log2'**, random_state=40, max_depth = 5)

Accuracy of train: 0.8144401138845122

Accuracy of test: 0.8119402985074626

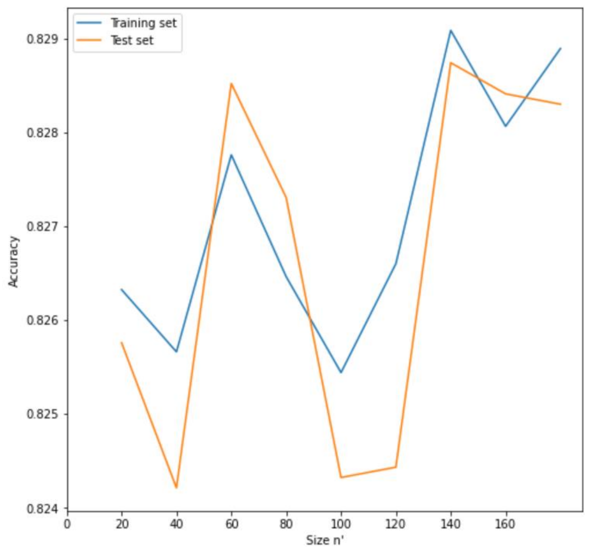
RandomForest5=RandomForestClassifier(max_features=**None**, random_state=40, max_depth = 5)

Accuracy of train: 0.8500152030295491

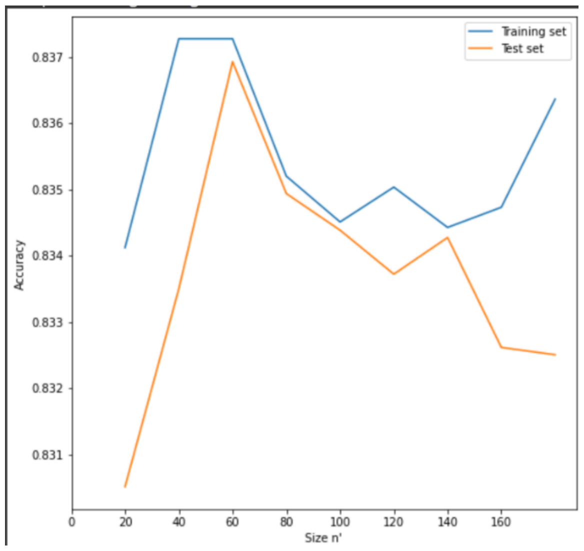
Accuracy of test: 0.8451077943615257

Even though max_features=None is when the model overfits the most, it is the best option as it gives the best training and test accuracy.

<u>Number of trees - n estimators</u>



Graph 1



Graph 2

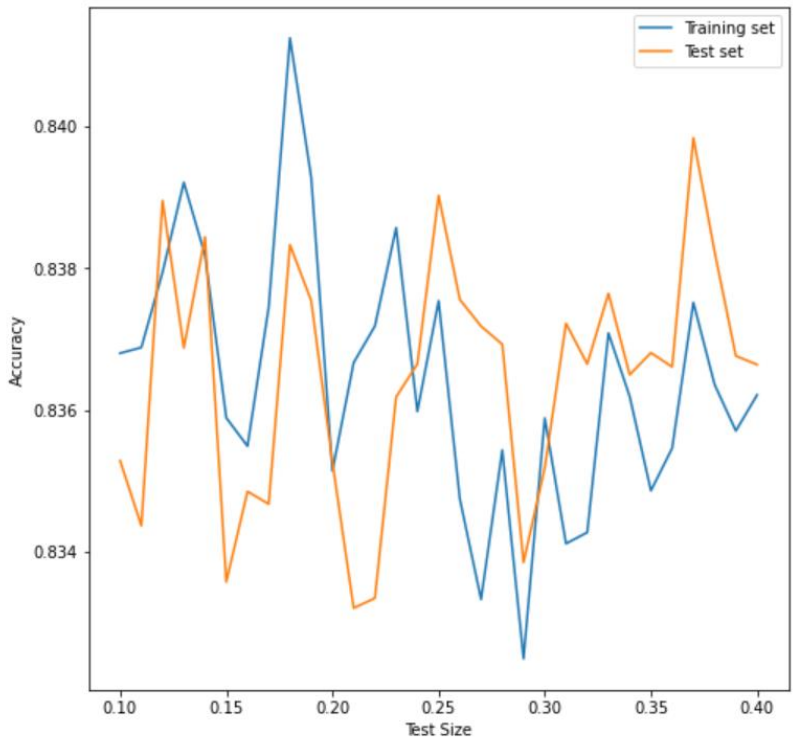
The difference between these two plots is that in the first one the max depth = 4.5 but in the second plot the max_depth = 5. There are a few points in the first plot where the models test and train accuracy seem to be exactly the same which looks better compared to the second model where it always seems to overfit (Although not by a lot). However, looking at the range of the test/training accuracy score, when max_depth is 5 it is much higher (0.832 and 0.837) compared to the first plot (0.824 and 0.829). Therefore, the model where max_depth is 5 would be the the one to choose.

Accuracy of train (**n_estimators = 75**): 0.8351438759432789

Accuracy of test (**n_estimators = 75**): 0.8352681039248203

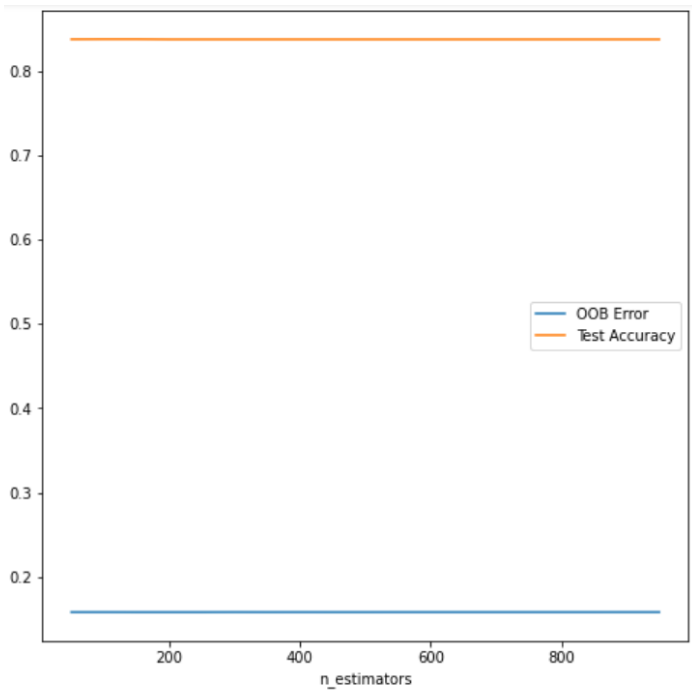
When max_depth=5 and n' = 75 (approximately) the training and test accuracy are high (0.8285) and with only a little bit of overfitting.

Training Set Size



When test size is around 24% the model's training and test accuracy is almost the same When test size < 24% the model is likely to overfit and when test size > 24% it's likely to underfit. But overall I don't think the test size matters a lot because the other hyperparameters are fixed at a position where the model could get the highest possible training and test accuracy while keeping the over and underfitting to a minimum therefore the accuracy scores here range between 0.832 and 0.842.

OOB Error



The test accuracy is much higher than the oob error which is what we want to achieve.

(c) Neural Networks

Keeping all other hyperparameters fixed

Accuracy of train: 1.0

Accuracy of test: 0.9694444444444444

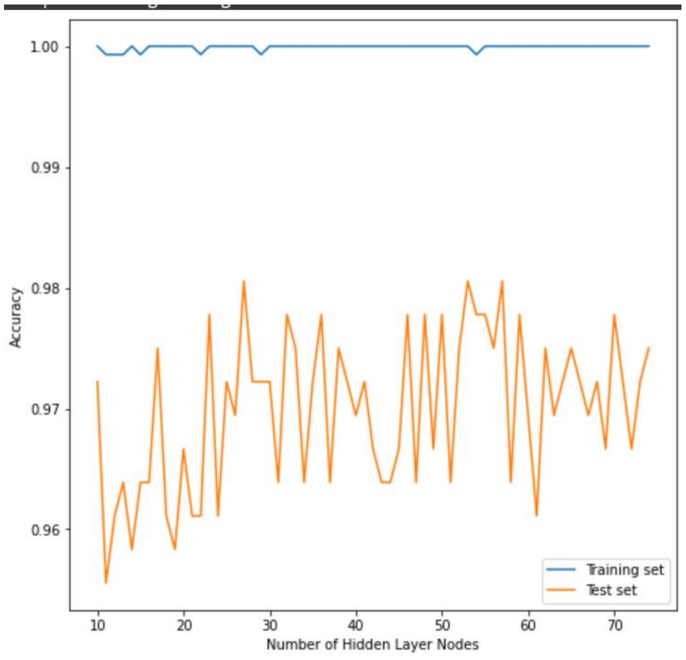
The accuracy of the training set may be so high because the default number of nodes in the hidden layer is very high (100). The model is still overfitting by around 0.031.

Hyperparameter Tuning

- Hidden layer nodes
- Nonlinearity choice
- Optimization algorithm
- Regularization Parameter

Going forward I would like to note that the assignment specifications recommended using only one hidden layer so that is exactly what I have done.

Hidden Layer Nodes



As the number of nodes in the hidden layer increases both the accuracy of the training and test set increases. The test accuracy remains closer to one no matter how many nodes there are in the hidden layer but the test accuracy fluctuates although within the same range (0.96 and 0.98).

Nonlinearity Choice

nn3 = MLPClassifier (max_iter=1300, alpha=1e-4,activation=**'relu'** ,solver='sgd', learning_rate_init=.01, random_state=40)

Accuracy of train: 1.0

Accuracy of test: 0.9694444444444444

nn4 = MLPClassifier(max_iter=1300, alpha=1e-4,activation=**'logistic'** ,solver='sgd', learning_rate_init=.01, random_state=40)

Accuracy of train: 0.9972164231036882

Accuracy of test: 0.9694444444444444

nn5 = MLPClassifier(max_iter=1300, alpha=1e-4,activation=**'tanh'** ,solver='sgd', learning_rate_init=.01, random_state=40)

Accuracy of train: 1.0

Accuracy of test: 0.9777777777777777

All of the models have the problem of overfitting but the best activation method is 'tanh' as it it gives the highest training accuracy when all other hyperparameters are fixed and only overfits by 0.023 wheras the other two neural network models overfit by around 0.03.

Optimization Algorithm

nn6 = MLPClassifier(max_iter=1300, alpha=1e-4,activation='relu' ,solver=**'sgd'**, learning_rate_init=.01, random_state=40)

Accuracy of train: 1.0

Accuracy of test: 0.9666666666666667

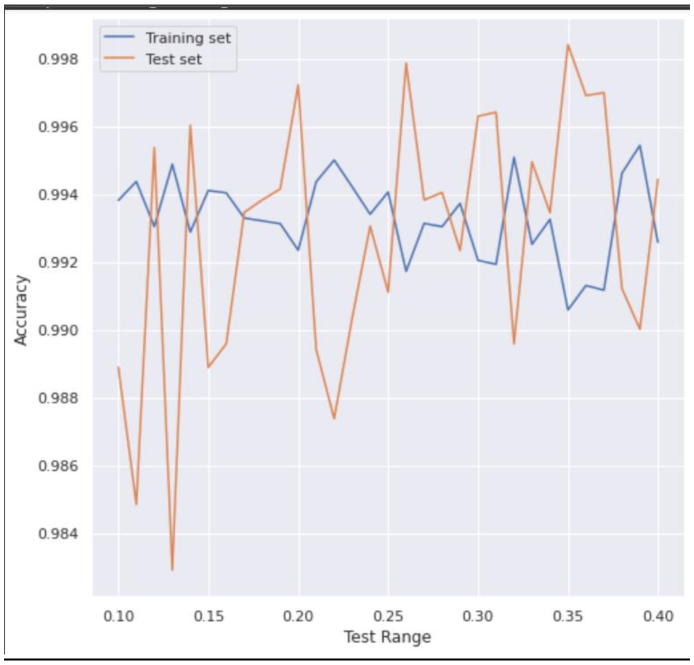
nn7 = MLPClassifier(max_iter=1300, alpha=1e-4,activation='relu' ,solver=**'adam'**, learning_rate_init=.01, random_state=40)

Accuracy of train: 1.0

Accuracy of test: 0.9611111111111111

'sgd' refers to stochastic gradient descent. This optimization algorithm gives a better result. As it overfits the least. But ‘adam’ also known as the stochastic gradient-based optimizer algorithm's likelihood of overfitting is not too far off from that.

Test Size



It looks like the proportion of the training and test model does not affect the Neural Network model as it the training and test accuracy seem to fluctuate between the same range no matter what the proportions are.

Comparing The Methods

Comparing how well decision trees, random forests and neaural networks work on the same set of data is hard as it really depends on the fixed hyperparameters. The first two methods gave a training/test score ranging between roughly 0.75 and 0.9. And the neural network models almost always had their training accuracy at 100% and test accuracy usually not below 96%. In conclusion, what I found most interesting is that as long as we're able to find out the values of hyperparameters where we will get the highest training/test accuracy with the least amount of overfitting, any of these three methods will be good.