Assignment 4
Decision trees and Random forests
Nikita Mehrotra (PhD18013)

1. Decision Trees

```
Accuracy score without hyperparameter tuning(DECISION TREE):
Accuracy Score on train data:  1.0
Accuracy Score on test data:  0.696
```

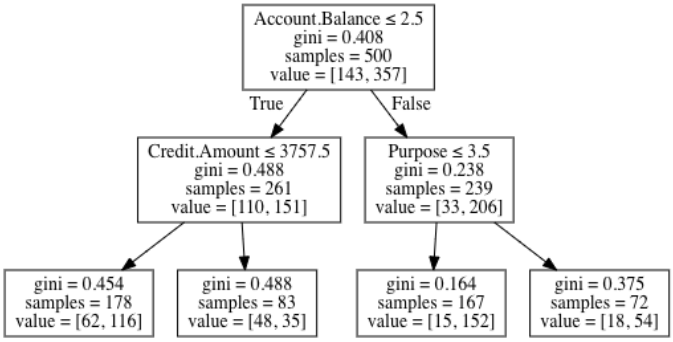
Accuracy Score (Without Hyper-parameter tuning)

```
{'criterion': 'gini',
 'max_depth': 7,
 'max_features': 8,
 'min_samples_leaf': 2,
 'min_samples_split': 2,
 'splitter': 'random'}
```

Best Parameters (Using GridSearchCV)

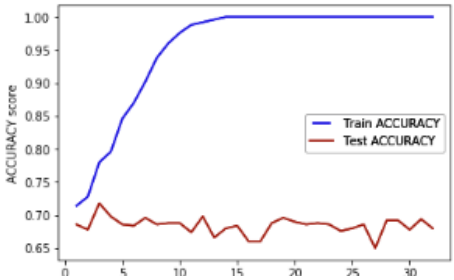
```
Accuracy Score with Hyperparameter tuning(DECISION TREE):
Accuracy Score on train data:  0.83
Accuracy Score on test data:  0.644
```

Accuracy Score (after Hyper-parameter tuning)

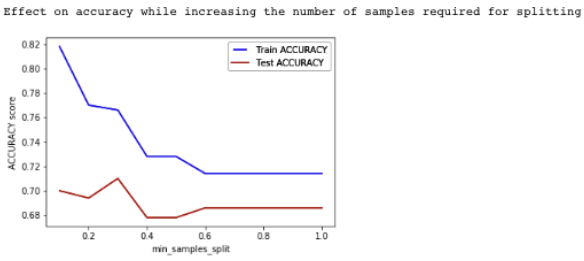


Sample Decision tree (of depth 2)

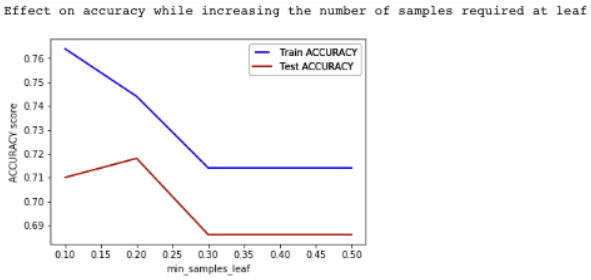
Plots showing the choices made during hyper parameter tuning



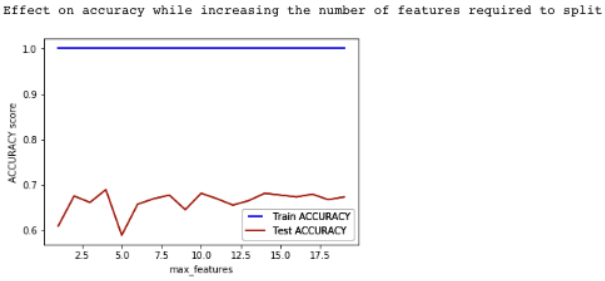
We see that our model overfits for large depth values. The tree perfectly predicts all of the train data, however, it fails to generalize the findings for new data



We can clearly see that when we consider 100% of the samples at each node, the model cannot learn enough about the data. This is an underfitting case.

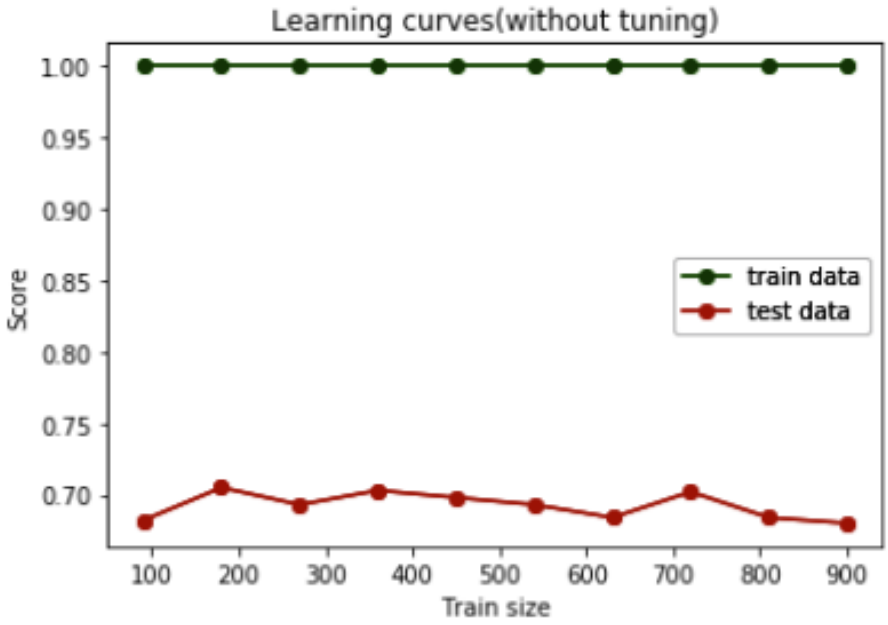


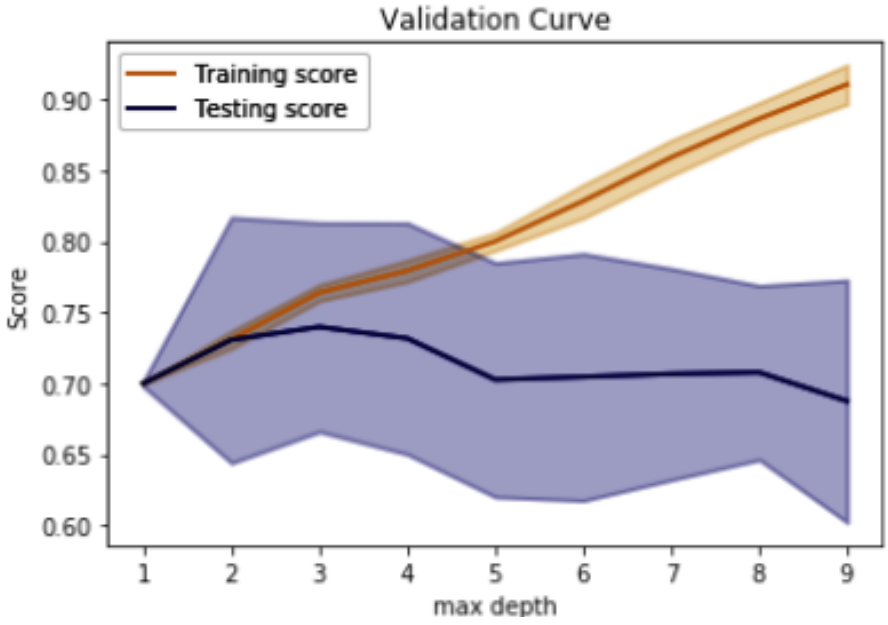
Increasing this value may cause underfitting.



This is also an overfitting case.

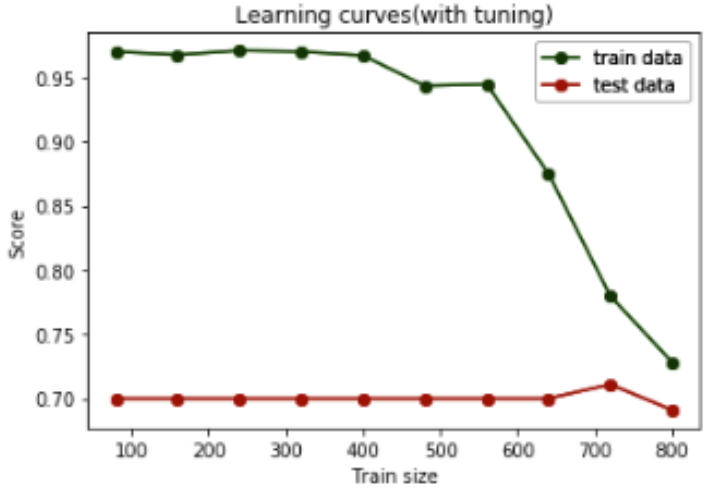
Learning and validation Curves without tuning

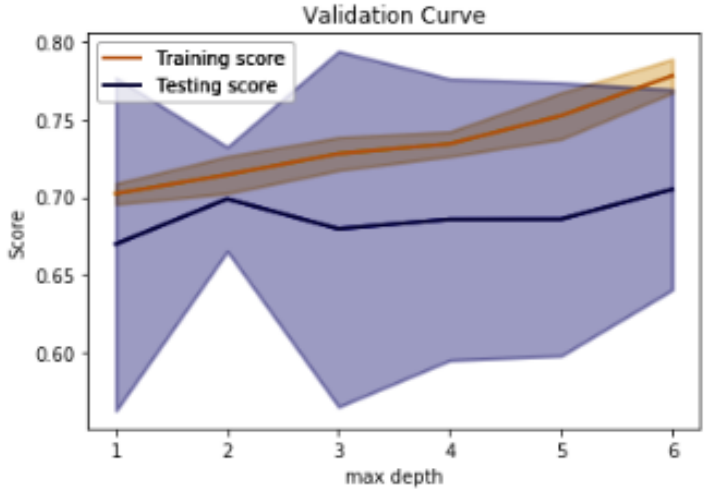


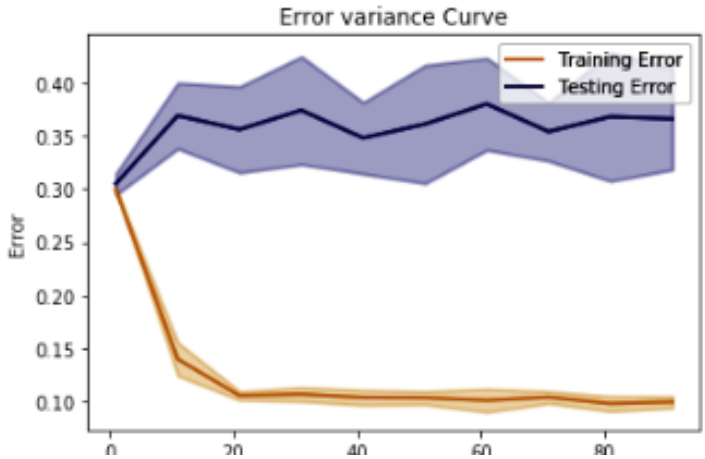


Type to enter a caption.

Curves with tuning







varaince error0.027999999999999987

2. Random Forests

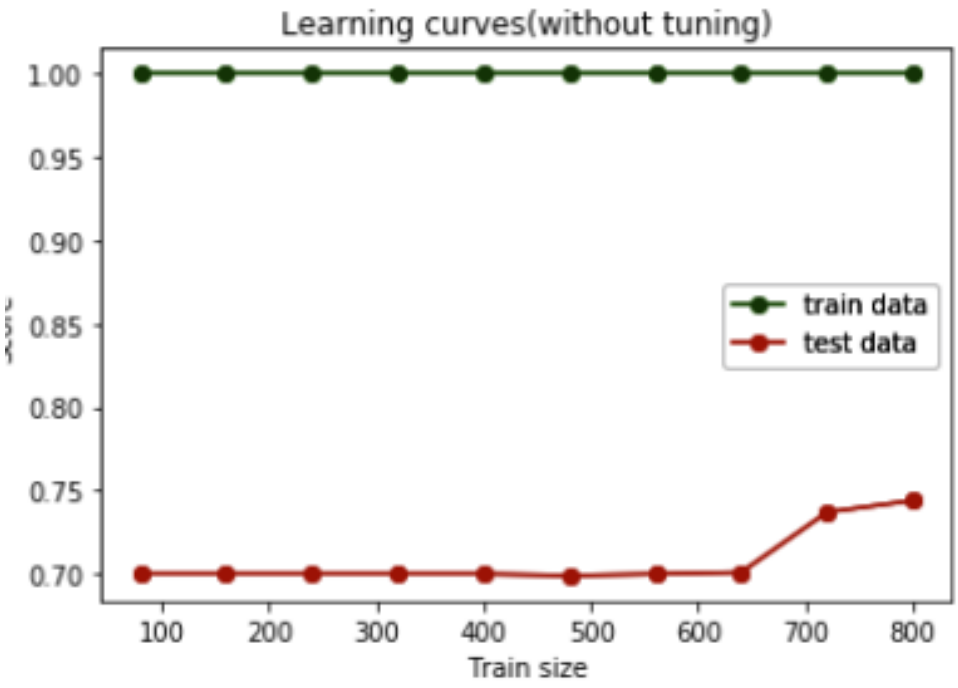
```
Accuracy Score without Hyperparameter tuning(Random forests):
Accuracy Score on train data:  1.0
Accuracy Score on test data:  0.732
```

Accuracy Score (Without Hyper-parameter tuning)

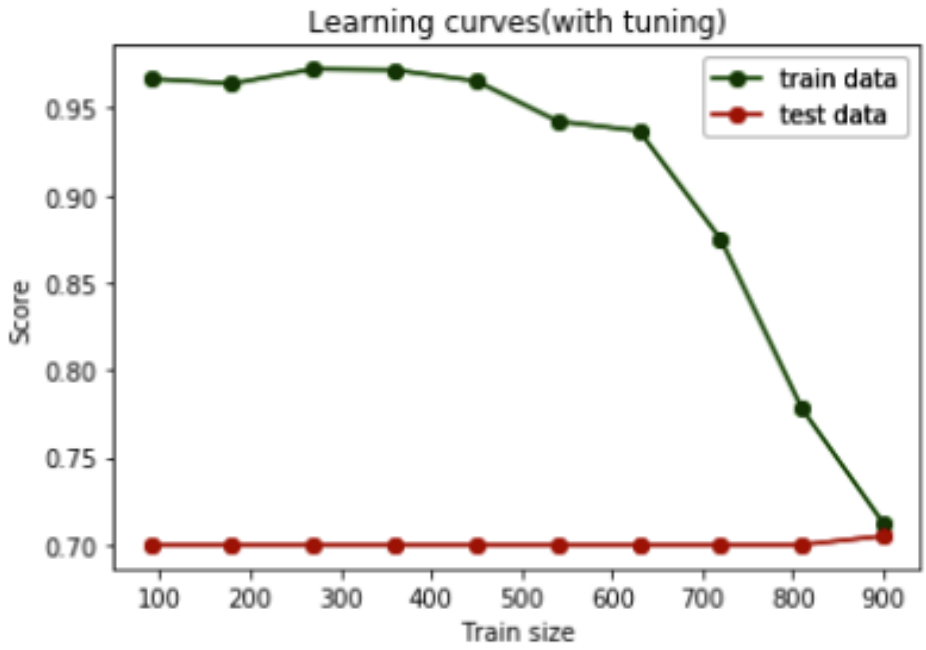
Accuracy Score with Hyperparameter tuning(RANDOM FOREST):
Accuracy Score on train data: 0.726
Accuracy Score on test data: 0.688

```
{'min_samples_split': 0.1,
 'min_samples_leaf': 0.1,
 'max_features': 7,
 'max_depth': 32.0,
 'criterion': 'gini',
 'bootstrap': True}
```

Best Parameters (Using Randomized GridSearchCV)

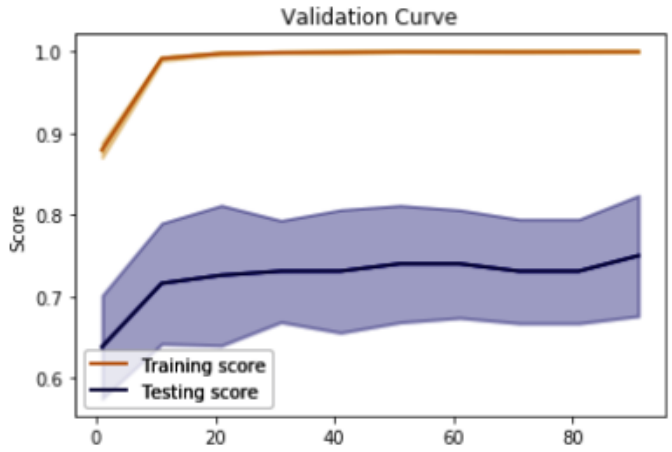


Type to enter a caption.



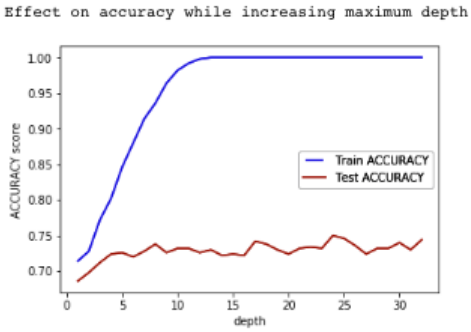
Type to enter a caption.

Validation curve(before tuning)

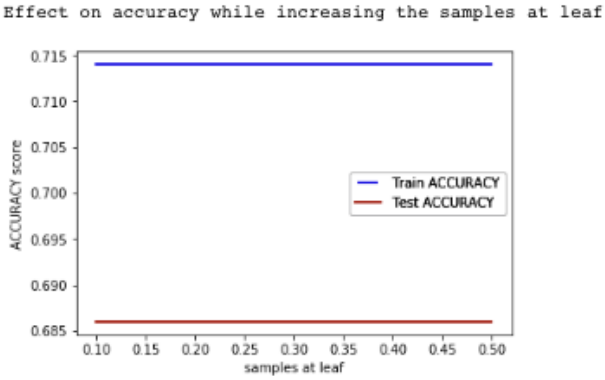


Type to enter a caption.

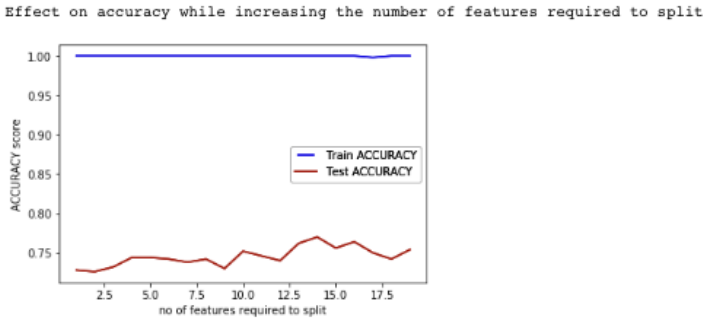
Justification for tuning hyper parameters



We see that our model overfits for large depth values.

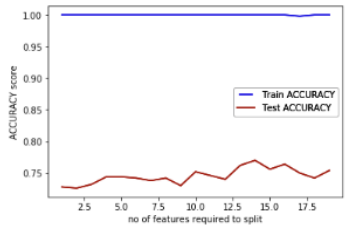


Increasing this value can cause underfitting.



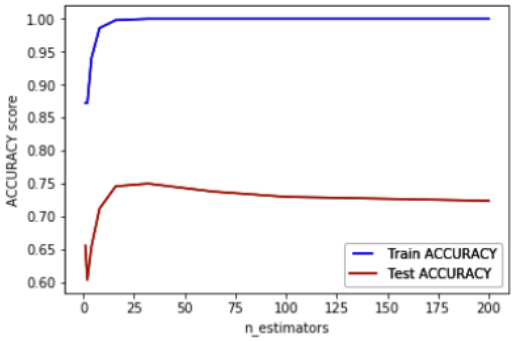
This is also an overfitting case

Effect on accuracy while increasing the number of features required to split



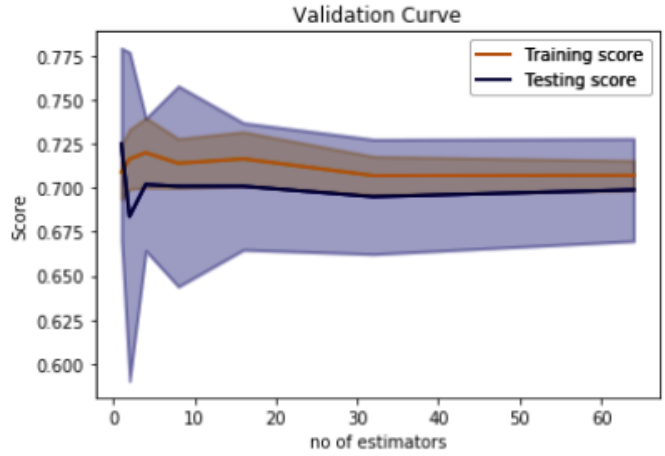
We can clearly see that when we require all of the samples at each node, the model cannot learn enough about the data. This is an underfitting case.

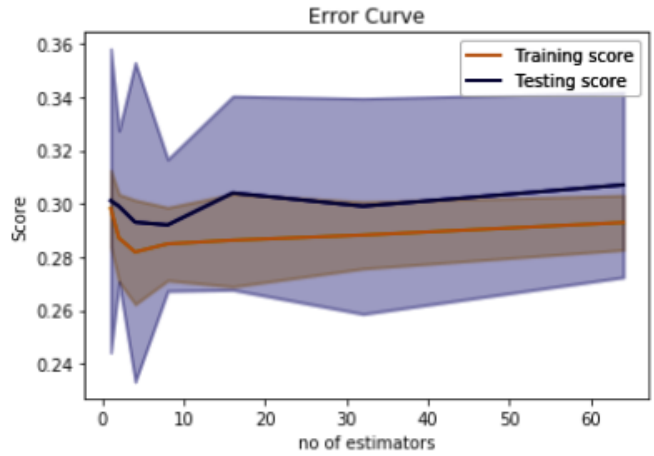
Effect on accuracy while increasing the number of estimators



We can see that for our data, we can stop at 32 trees as increasing the number of trees decreases the test performance.

Validation and error variance curve and error variance (after tuning)






```
varaince error 0.011401754250991 36
```

Type to enter a caption.

RANDOM FORESTS	DECISION TREES
Accuracy without tuning: Train:100 Test:73.6	Accuracy without tuning: Train:100% Tests:70%
Accuracy after tuning: Train: Test:	Accuracy after tuning: Train:74% Test:69%

Random forest is basically a ensemble of decision trees and it works better than decision trees, whose results are aggregated into one final result. Decision trees are prone to overfitting, so in order to remove overfitting we will set max depth. This will fix high variance issue but it will oversimplify the model which will lead to high bias. Ideally we like to minimise both of these problems, hence random forests are used. Random forest reduce variance by timing on different sample and by using subset of features for different decision trees.