BUAN 6341: Applied Machine Learning
Madhav Sampath Gubbala (MSG170330)

Introduction

The goal here is to use Support Vector Machine (SVM), Decision Trees, boosting algorithms to predict the respective target variables. This report includes data analysis, model building and experimentation by tuning the parameters. Through the experimentation the best model is identified.

In this assignment we use 2 data sets, the description of the respective data sets is given below:

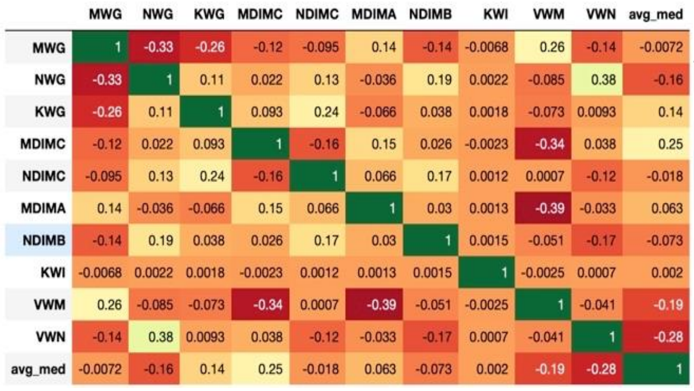
GPU Run Time:
This data set measures the running time of a matrix-matrix product A*B = C, where all matrices have size 2048 x 2048, using a parameterizable SGEMM GPU kernel with 241600 possible parameter combinations.
This dataset is used for implementing the Support Vector Machine (SVM),Cross Validation algorithms is downloaded from UCI Machine Learning repository. The details about the dataset are mentioned below:

- Dataset consists of 241600 observations on 14 variables.
- Dependent variable is the average of Run1 (ms), Run2 (ms), Run3 (ms), Run4 (ms) columns which is 'avg_med'.
- Data split is done by 70:30 ratio for training and testing data set with no missing values.

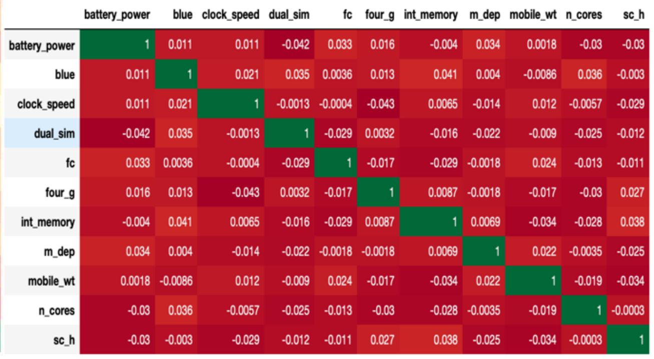
Mobile Price Classification:
Link: https://www.kaggle.com/iabhishekofficial/mobile-price-classification
This data set measures the Price of the mobile phone with different features with 2500 possible parameter combinations. The price of data set is classified into two fa
This dataset is used for implementing the Decision Trees, Boosting, Cross Validation algorithms is downloaded from Kaggle. The details about the dataset are mentioned below:

- Dataset consists of 2500 observations on 21 variables
- Dependent variable is the price of the mobile phone.
- Data split is done by 70:30 ratio for training and testing data set with no missing values.

Correlation: In SVM and Decision Tree algorithms correlation is important factor. For SVM, the effect is similar to that of multicollinearity in linear regression for linear kernel. For decision trees, no assumptions will be made by the algorithm on relation between features. If highly correlated information gain will be less.



	MWG	NWG	KWG	MDIMC	NDIMC	MDIMA	NDIMB	KWI	VWM	VWN	avg_med
MWG	1	-0.33	-0.26	-0.12	-0.095	0.14	-0.14	-0.0068	0.26	-0.14	-0.0072
NWG	-0.33	1	0.11	0.022	0.13	-0.036	0.19	0.0022	-0.085	0.38	-0.16
KWG	-0.26	0.11	1	0.093	0.24	-0.066	0.038	0.0018	-0.073	0.0093	0.14
MDIMC	-0.12	0.022	0.093	1	-0.16	0.15	0.026	-0.0023	-0.34	0.038	0.25
NDIMC	-0.095	0.13	0.24	-0.16	1	0.066	0.17	0.0012	0.0007	-0.12	-0.018
MDIMA	0.14	-0.036	-0.066	0.15	0.066	1	0.03	0.0013	-0.39	-0.033	0.063
NDIMB	-0.14	0.19	0.038	0.026	0.17	0.03	1	0.0015	-0.051	-0.17	-0.073
KWI	-0.0068	0.0022	0.0018	-0.0023	0.0012	0.0013	0.0015	1	-0.0025	0.0007	0.002
VWM	0.26	-0.085	-0.073	-0.34	0.0007	-0.39	-0.051	-0.0025	1	-0.041	-0.19
VWN	-0.14	0.38	0.0093	0.038	-0.12	-0.033	-0.17	0.0007	-0.041	1	-0.28
avg_med	-0.0072	-0.16	0.14	0.25	-0.018	0.063	-0.073	0.002	-0.19	-0.28	1



	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	sc_h
battery_power	1	0.011	0.011	-0.042	0.033	0.016	-0.004	0.034	0.0018	-0.03	-0.03
blue	0.011	1	0.021	0.035	0.0036	0.013	0.041	0.004	-0.0086	0.036	-0.003
clock_speed	0.011	0.021	1	-0.0013	-0.0004	-0.043	0.0065	-0.014	0.012	-0.0057	-0.029
dual_sim	-0.042	0.035	-0.0013	1	-0.029	0.0032	-0.016	-0.022	-0.009	-0.025	-0.012
fc	0.033	0.0036	-0.0004	-0.029	1	-0.017	-0.029	-0.0018	0.024	-0.013	-0.011
four_g	0.016	0.013	-0.043	0.0032	-0.017	1	0.0087	-0.0018	-0.017	-0.03	0.027
int_memory	-0.004	0.041	0.0065	-0.016	-0.029	0.0087	1	0.0069	-0.034	-0.028	0.038
m_dep	0.034	0.004	-0.014	-0.022	-0.0018	-0.0018	0.0069	1	0.022	-0.0035	-0.025
mobile_wt	0.0018	-0.0086	0.012	-0.009	0.024	-0.017	-0.034	0.022	1	-0.019	-0.034
n_cores	-0.03	0.036	-0.0057	-0.025	-0.013	-0.03	-0.028	-0.0035	-0.019	1	-0.0003
sc_h	-0.03	-0.003	-0.029	-0.012	-0.011	0.027	0.038	-0.025	-0.034	-0.0003	1

Task 1:

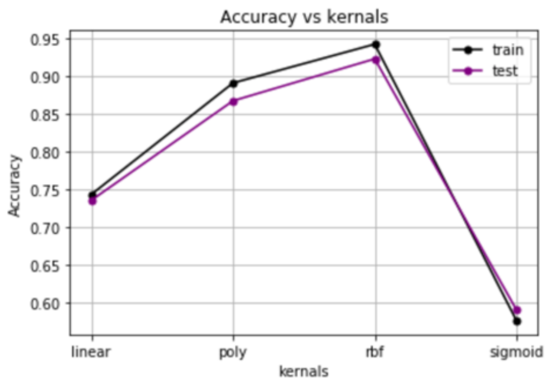
Import both the datasets and split them into train and test sets in 70:30 ratio. Both the datasets are converted into binary classification. As SVM heavily rely on the spatial data points, we have scaled the data using Standard Scaler. Another copy of the preprocessed data is not scaled because Decision Tree and Algorithm doesn't require the data to be scaled. Both the data sets don't have null values.

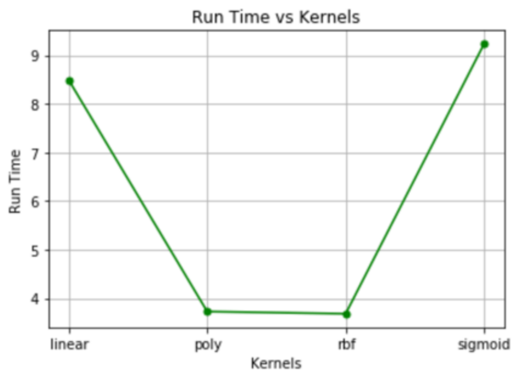
Task 2:

I imported SVM from sklearn package and implemented the SVM algorithm and drew the following conclusions

Support Vector Machine (SVM)

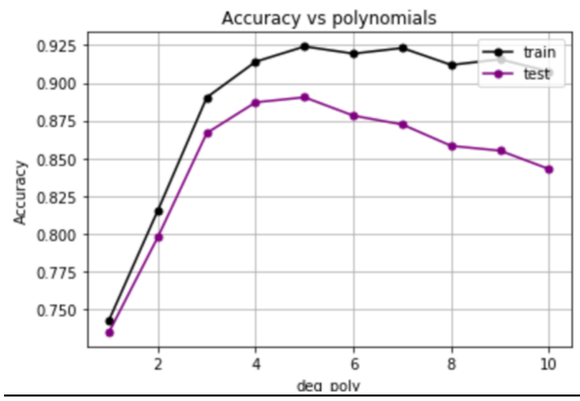
Experiment 1 : Varying Kernels

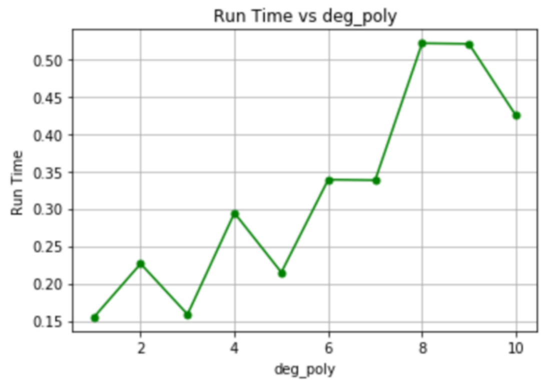




- Rbf kernel had highest accuracy with train and test data.
- Sigmoid had the worst accuracy with train and test data.
- Poly and rbf kernels run faster than other kernels.

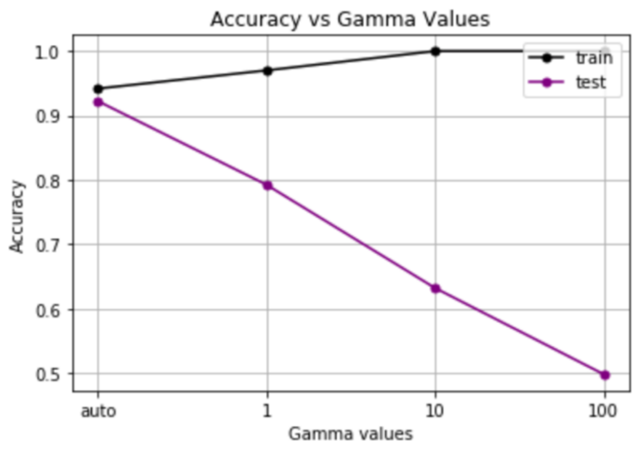
Experiment 2: Varying Polynomial Degree





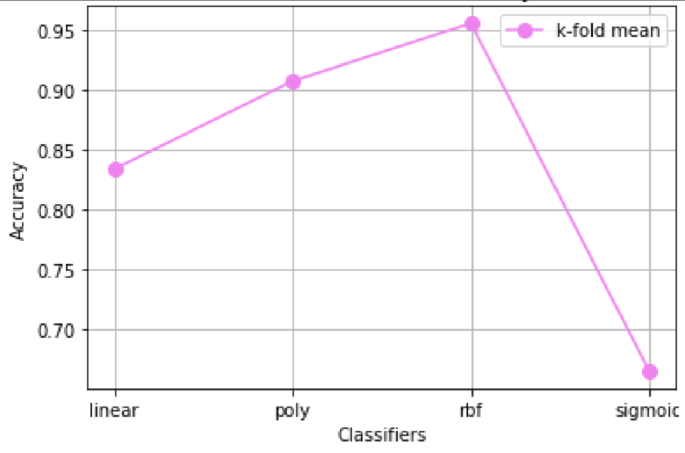
- It is evident that data is not linear as the accuracy increases with the increase in the degree of polynomial.
- It is obvious that Run time increases with the increase in the degree of polynomial, but optimal solution occurs at deg 3 as it has high accuracy and very low run time.

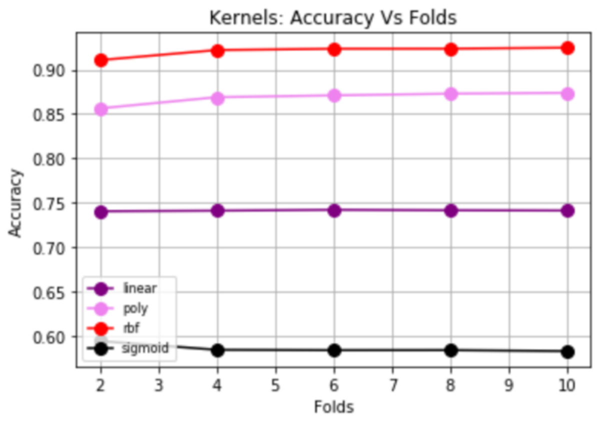
Experiment 3: Varying 'gamma' parameter for rbf kernel



- With the increase in gamma value the train data accuracy increases whereas out-sample error is decreased which states that the data is overfitting.
- Smaller the gamma values lower the variance and higher the bias.

Experiment 4: K-fold Cross validation by varying kernels and varying folds





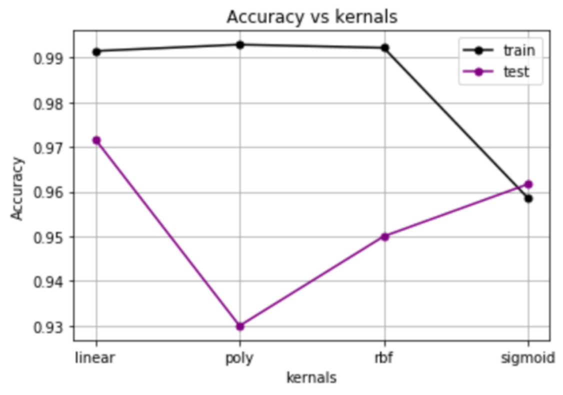
- Since we already got much higher accuracy for base model, as expected Cross validation didn't improve much.

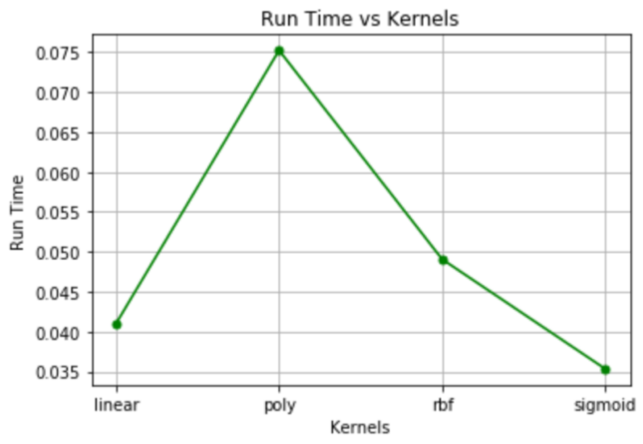
- There is no much difference in Accuracy by changing the folds.

Mobile Price Classification

Support Vector Machine (SVM)

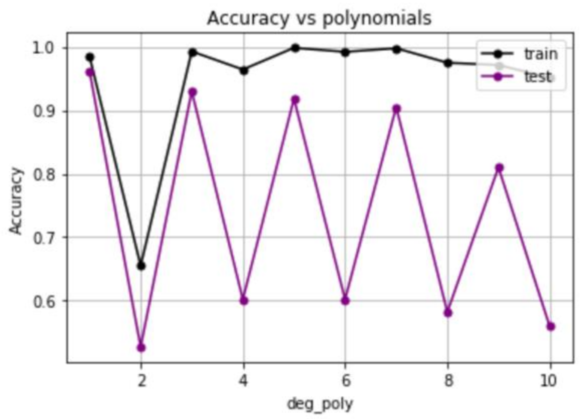
Experiment 1: Varying Kernels

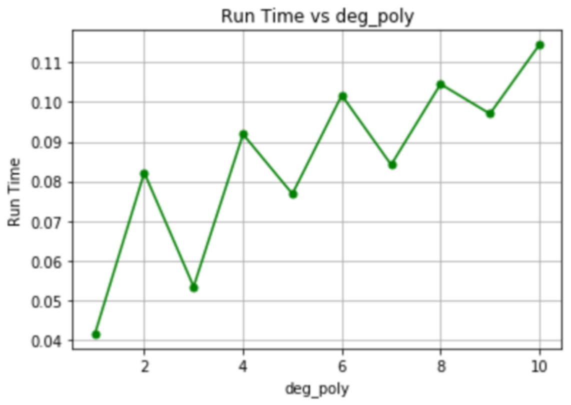




- Rbf and poly kernel had highest accuracy with train data than test data which implies that data is over fitted.
- Data is well separated linearly. So cross validation didn't improve much on the normal fit for any of the kernels.
- Sigmoid had the worst accuracy with train data and poly had worst accuracy with test data.
- Sigmoid and linear kernels run faster than other kernels.

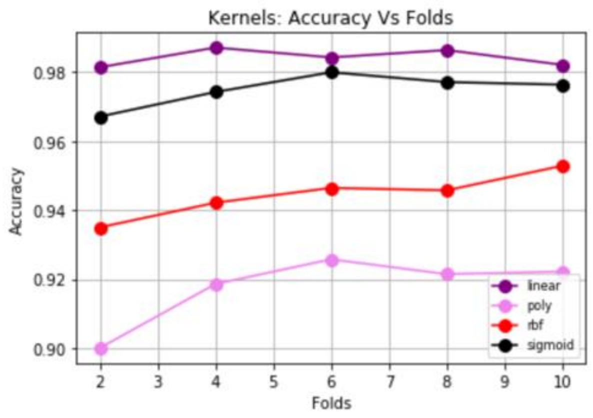
Experiment 2: Varying Polynomial Degree





- It is evident that data is linear as the accuracy decrease with the increase in the degree of polynomial.
- It is obvious that Run time increases with the increase in the degree of polynomial, but optimal solution occurs at deg 1 as it has high accuracy and very low run time.

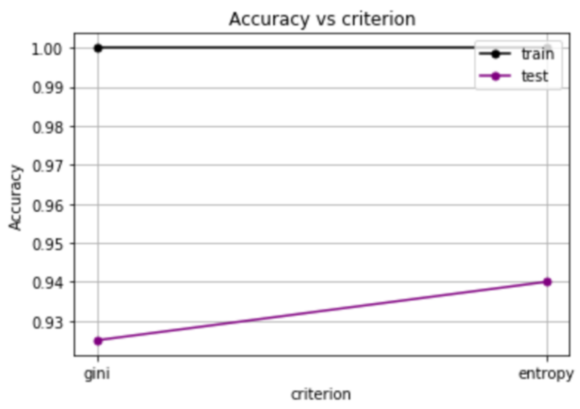
Experiment 3: K-fold Cross validation by varying kernels and varying folds

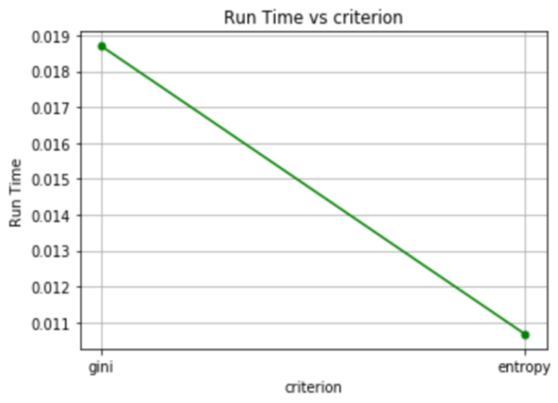


- Only in the linear kernel there is some variance in the accuracy with the increase in the number of folds.

Decision Tree and Boosting

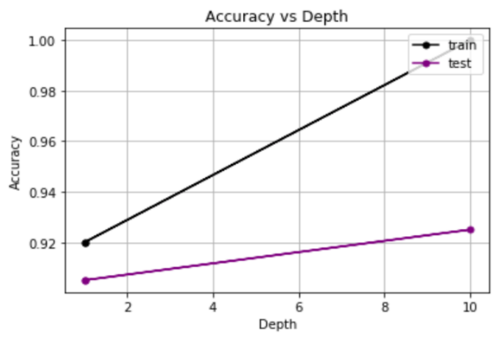
Experiment 1: Accuracy and Run time vs criterion:

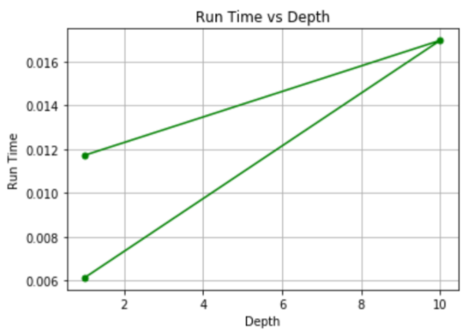




- No difference in accuracy for any of the metrics, but the test error is low.
- It might be due to not restricting any parameters for the tree thereby overfitting.
- Decision tree is Good when data is small.

Experiment 2: Accuracy and Run time vs Depth:

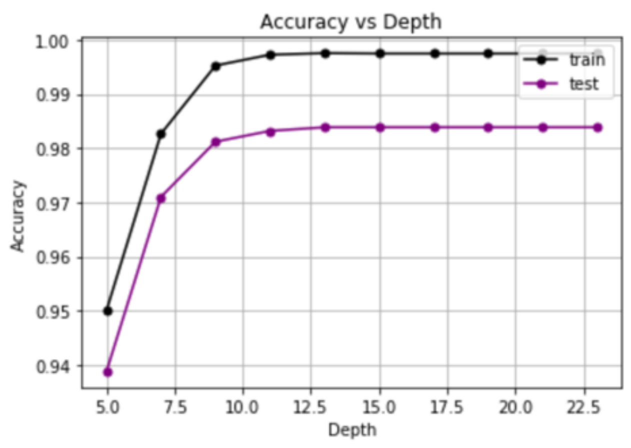




- Accuracy increases as the no of folds increase for both test and train data.

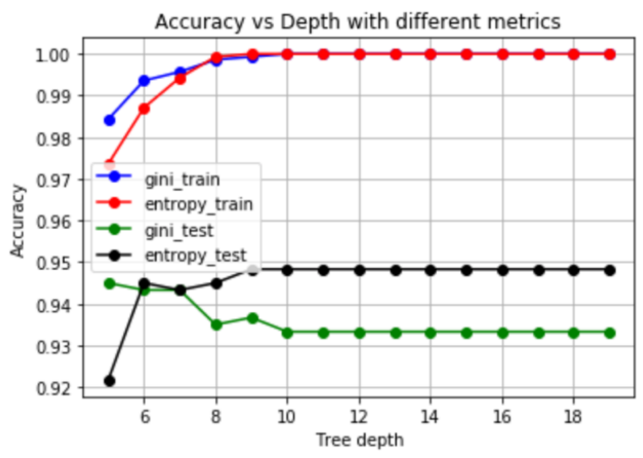
- Run time increases with the increase in no of folds.

<u>Experiment 3: Boosting Decision Tree</u>



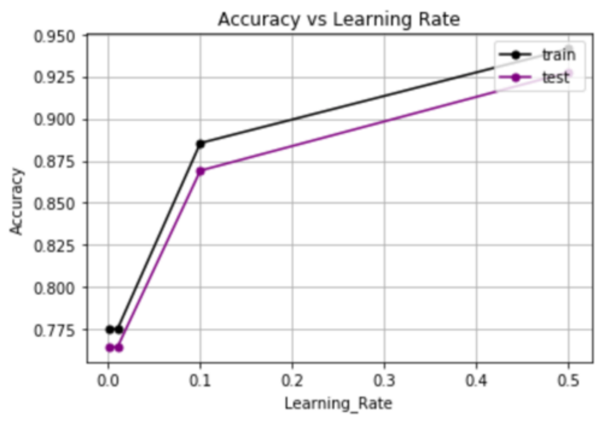
- Comparing the boosted and normal plot boosted had higher accuracy.
- In Boosted data over fitting occurs as the depth increases as the test data accuracy is increasing.
- Optimal value of depth is 12.

<u>Experiment 4: Accuracy vs Depth with different metrics:</u>

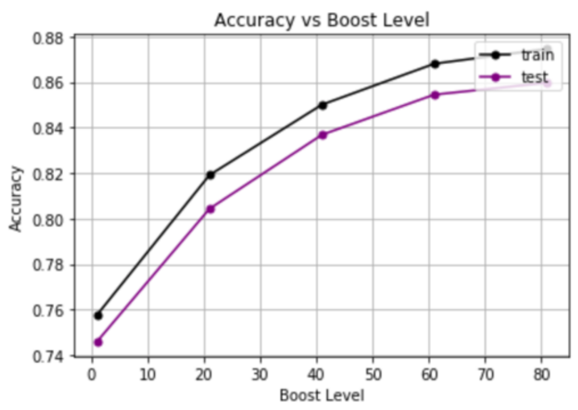


- As the depth increases, we can observe that train accuracy is increasing but the test accuracy remained same or decreased for gini metric
- This is can be an overfitting with increasing tree depth.
- Hence the optimal value of tree depth can be found at 6

<u>Experiment 5: Boosting – Learning Rate and Boosting Level</u>



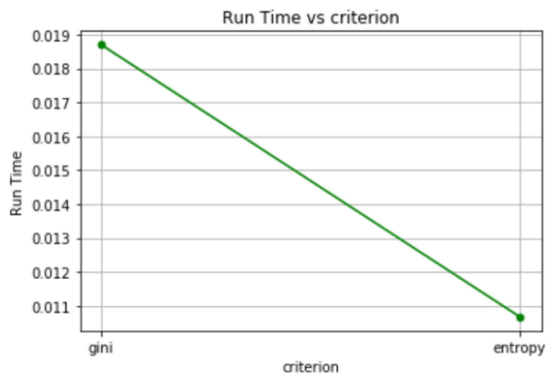
- Learning rate is the amount of overfitting in a way you could say with each new tree.
- So ,if it is very low (slow, gradual learning from one tree to next) obviously you will need a lot of trees to get a final reasonable model.
- Optimal learning can be found between 0.2 and 0.3 from the learning curve above.

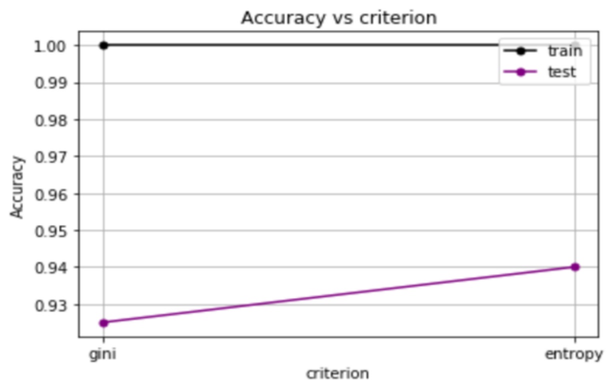


- There is no much increase in the accuracy beyond n estimator beyond 60.
- The reason is in the way that the boosted tree model is constructed, sequentially where each new tree attempts to model and correct for the errors made by the sequence of previous trees.
- The same can be confirmed from the hyper-parameter tuning where the n_estimator is 60.

Mobile Price Classification

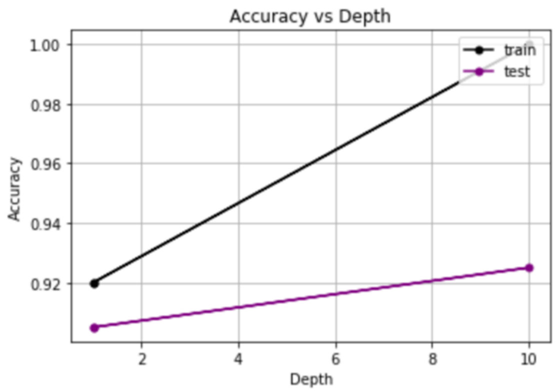
Experiment 1: Accuracy and Run time vs criterion:

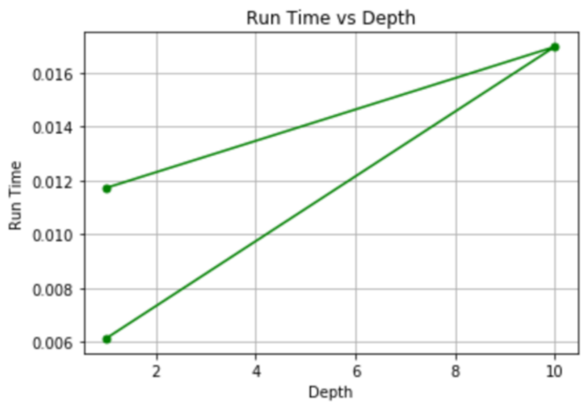




- No difference in selection criterion with respective to accuracy for both the indexes only the test accuracy is low for both.
- The run time for entropy is less when compared to Gini index.

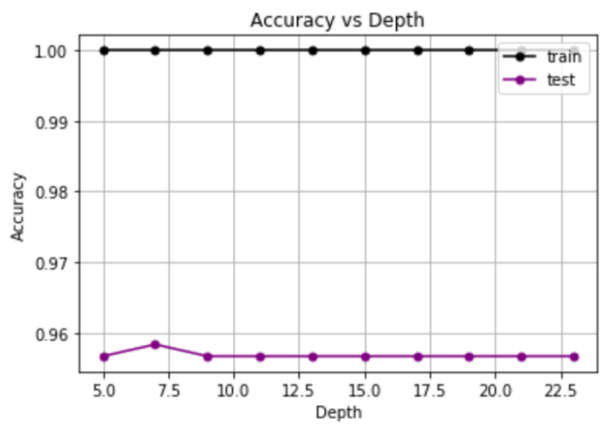
Experiment 2: Accuracy and Run time vs Depth:





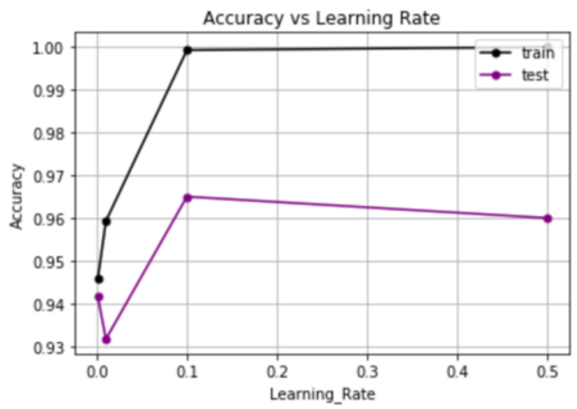
- It can be observed that the tree doesn't fit fully at lower depth, hence lower accuracy
- As the depth increases, the accuracy rises for both train and test

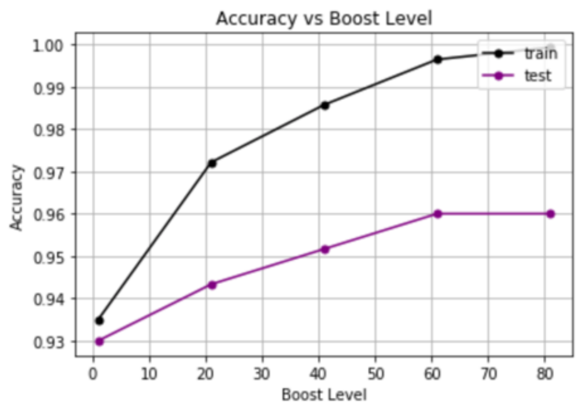
Experiment 3: Boosting



- From the plot, we can clearly observe that boosted tree has improved on decision tree
- But as the tree depth increases, there is no much difference in test and train accuracy.
- Optimal value of tree depth can be observed at 5

Experiment 4: Boosting – Learning Rate and Boosting Level





- If the learning rate is very low, you will need a lot of trees to get a final reasonable model.
- If you want to fit faster, or rather overfit itself, set a higher learning rate and low number of trees
- Optimal learning rate can be observed at 0.1 as the train and test accuracy are higher.

- As the boosting level increases, the accuracy keeps on increasing and stables at 60. Hence the learning curve gives 60 as optimal boosting level.

Task 5:
Cross Validation is implemented on both the datasets for different algorithms and performed experimentation for various folds along with their plots as shown in the above pages.

Performed hyper-parameter tuning using gridSearch:

Mobile Price Classification	GPU Run time
`{'model': 'XGBoost', 'parameter search time': '0:01:53.957000', 'accuracy': 0.974, 'test auc score': 0.998, 'training auc score': 1.0, 'parameters': {'colsample_bytree': 1.0, 'gamma': 0.5, 'learning_rate': 0.12307935650199768, 'm_child_weight': 1.0, 'max_depth': 2.0, 'n_estimators': 350.0, 'subsample': 0.55}}`	`'model': 'XGBoost', 'parameter search time': '5:54:32.412000', 'accuracy': 0.992, 'test auc score': 1.0, 'training auc score': 1.0, 'learning_rate': 0.13509889841790165, 'm_child_weight': 1.0, 'max_depth': 12.0, 'n_estimators': 525.0, 'subsample': 0.9}}`