Assignment 2: Project Report – SVM, Decision Tree, Boosting

Submitted By: **Shubham Murari**
Net ID: **SXM170056**
Course: **BUAN 6341.501 – Applied Machine Learning**

Problem Summary-

Data Set 1:

We have been provided with a regression problem for **'Appliances Energy Prediction'.** There are 19735 instances (records) with no missing values. The energy data was logged on a time basis where the Temperature and Humidity Conditions of a house were monitored with the help of sensors.

Data Set 2:

I have selected '**Bank Marketing Dataset'** from UC Irvine repository, following is the **link** for the same: https://archive.ics.uci.edu/ml/datasets/Bank+Marketing

Goal:

Apply ***SVM, Decision Tree, boosting algorithms*** in both datasets both by after ***train test split and cross validation*** and compare the respective results with appropriate reasoning.

Exploratory Analysis and Feature Engineering:

Before directly jump into any conclusion or analysis, there are certain steps that I kept in my mind before starting-

1. The exploratory analysis of the energy data is same as before, we have already selected best features and implementing various algorithms on that.
2. There are total **4521 instances** in the '**Bank Marketing Dataset'** and based upon correlation matrix we dropped few of them to name few - *poutcome_unknown', 'contact_unknown', 'marital_single', 'pdays','education_tertiary'*
3. There are some features which have different classes e.g. **Housing, Default, loan** which had Yes/No classes. I used **Binary Encoding** to convert them into 0/1 class.
4. Data was highly imbalanced due to which we used **Under Sampling** to make it balanced

Why I found this ‘Bank Marketing Dataset’ problem interesting:

The data is related with direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be ('yes') or not ('no') subscribed.

Due to my personal interest in the Financial/Banking domain, I opted for this problem. Interesting thing is that there are lots of features that we need to consider here before coming to any solution. Moreover, my current internship project is also related to the similar kind of domain. So, solving this problem would give me an opportunity to apply some domain knowledge that I gathered during the course. Let’s start with the Analysis!!

Training and Testing Scores for different Scenarios:

Results for Dataset 1: Energy Data-

Sr. No	**Model**	**Train Test Split**		**Cross Validation (k =10)**	
		Train Score	**Test Score**	**Train Score**	**Test Score**
1	SVC [Linear Kernel]	70.70%	69.95%	70.17%	73.39%
2	SVC [rbf Kernel]	96.57%	73.88%	95.49%	73.03%
3	SVC [Sigmoid kernel]	61.85%	62.97%	60.95%	73.28%
4	Decision Tree [Full Length]	100%	76.77%	100.00%	62.89%
5	Decision Tree [Pruned]	74.41%	72.72%	73.93%	73.79%
6	Boosting [XGBoost]	75.68%	74.05%	75.49%	73.69%
7	Boosting [Pruned XGBoost]	94.95%	81.74%	81.97%	75.16%

Below is the detailed analysis of different experimentations that were performed in order to get these results-

1. Analysis of results from SVM-

When we use the **Train, Test Split** we are getting the better result in the **Linear Kernel,** for **rbf Kernel** there is an **overfitting** which is not good for our data. Apart from this **Sigmoid Kernel** in this particular scenario is not giving that good result, however difference between Train and Test performance is less than 0.5 which is statistically good, but we can't consider that for the business purpose.

Now when we implemented Cross Validation- we observed that the Test Score for the **Linear Kernel** got **improved,** which shows that CV is helpful in best fitting the model and overcome the problem of overfitting. But again, the performance of rbf and Sigmoid did not get improved much. Test Score for Sigmoid a bit improved. But again, Linear Kernel is performing good as compared to other Kernels.

2. Analysis of results from Decision Tree (before and after pruning)-

For both **Test Train Split and Cross Validation;** well decision tree is giving us some very INTERESTING INSIGHTS. When we are using Train and Test Split and running the Full-Length tree with Gini criterion (without giving any parameter values) we are getting the Train Score of 100% which is very obvious and due to overfitting our test score is very poor 76.77% as compared to train score. Now as we know that we can play around with ***<u>3 important Pruning Parameters</u>*** which are: **Max Depth, Min Samples Leaf and Max Leaf Nodes.** This experimentation gives us the following values for our parameters-

<u>Best Pruning Parameters for Cross Validation</u>

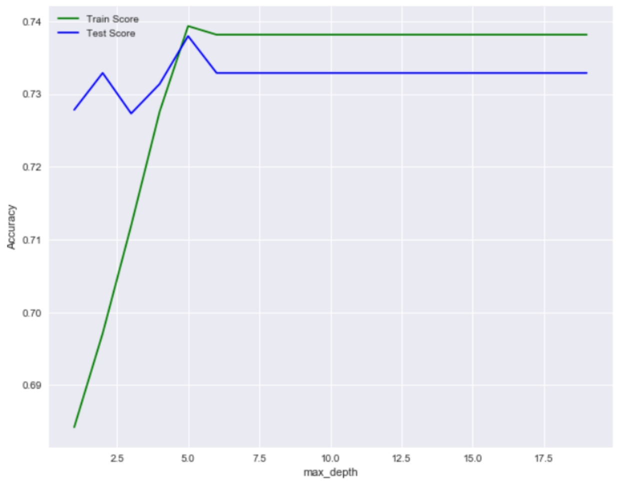
```
{'max_depth': 5, 'min_samples_leaf': 10, 'max_leaf_nodes': 28}
```

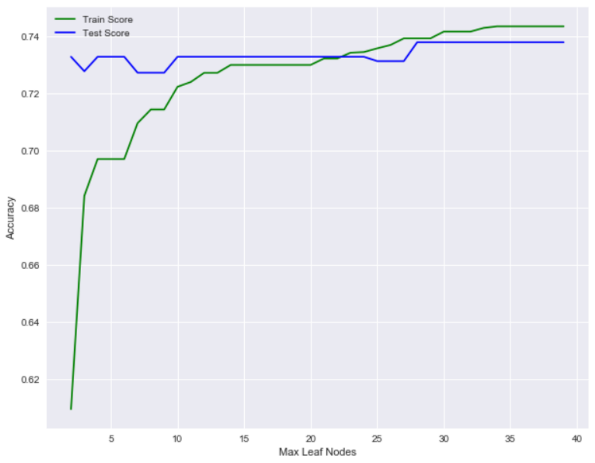
So, when I trained my Pruned Decision tree by feeding these values my results got optimized as you can see in the above table. By limiting these parameters my decision tree will classify each value in an optimal manner.

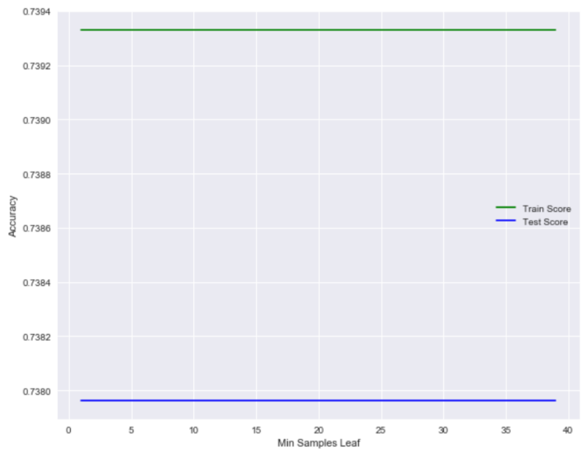
Following plots explain the above reasoning (basically the highlighted portion, these plots basically show that how my accuracy is getting better based upon optimal values of these Pruning Parameters)

As mentioned above **at Max Depth = 5 and Max Leaf Nodes = 28, Min Sample Leaf = 10** Train and Test Accuracies are comparable.

Learning Curves and explanation:



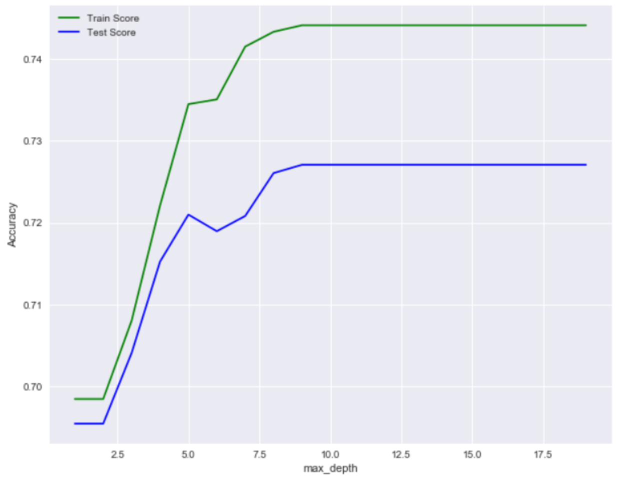


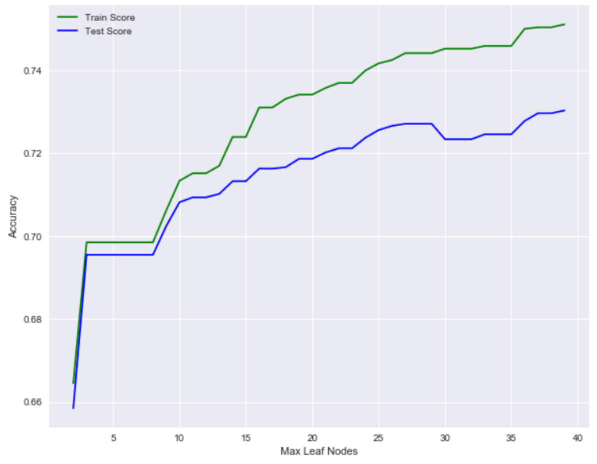


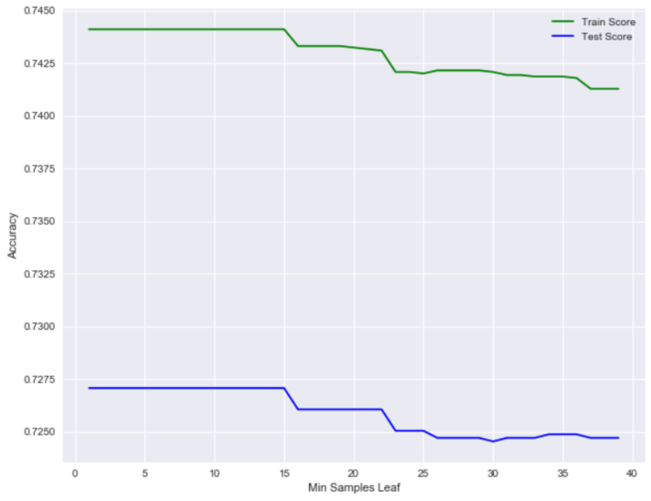
For **Min Sample Leaf = 10** Train and Test Accuracies are comparable.

Best Pruning Parameters for Train Test Split-

`{'max_depth': 9, 'min_samples_leaf': 10, 'max_leaf_nodes': 27}`







So these **Learning Curves** show the behaviour of Train and Test accuracies with respect to the **Pruning Parameters**. So for train and test split as you can see from the plot optimum scores are at **max depth =9, min sample leaf =10 and max leaf nodes = 27.**

So ultimately Pruning helps us to get rid of overfitting and optimal values of the pruning parameter is playing an important role in it, because these are basically the controlling factors which is responsible for the bias and variance trade off.

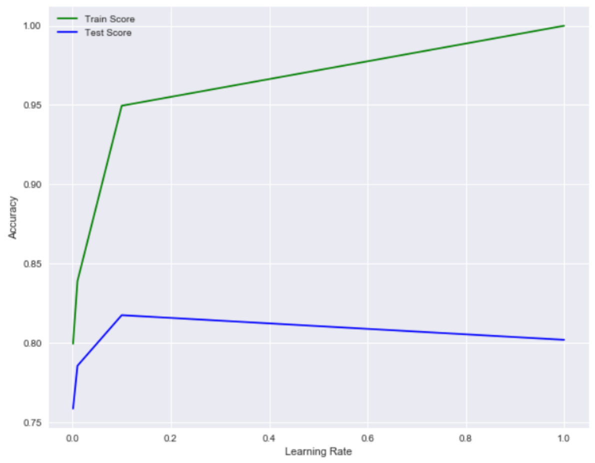
Using GINI criterion has the following advantage here:

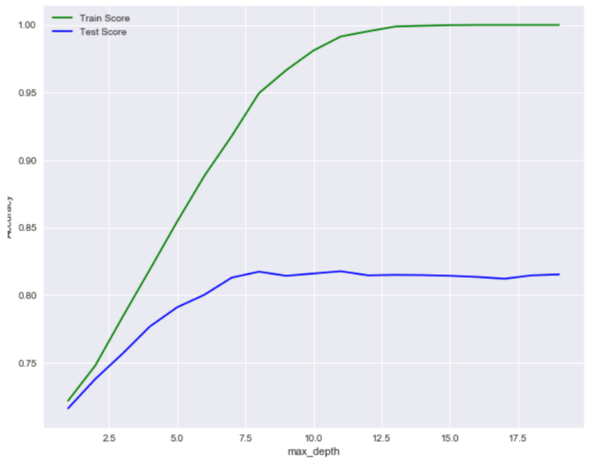
I would use the Gini impurity, as it doesn't require me to compute logarithmic functions, which are computationally intensive.

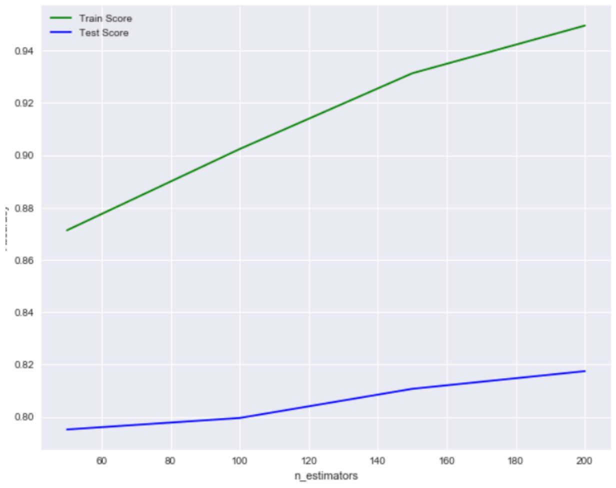
3. Analysis of results from Boosting (before and after pruning)-

In both the cases for Train and Test Split and as well as CV, boosted version of the trees improve our result significantly. Boosting in itself avoids overfitting. We can find the best parameters tweaking learning rate max depth and number of estimators which is how we experimented with the Pruning in XGBoost.

Which ultimately resulted in best train and test score for Pruned XGBoost for train test split, below is the Learning Curve for the pruned version of XGBoost which shows the optimal values of control parameters.







So these learning curves clearly show that at these values `{'learning_rate': 0.1, 'n_estimators': 200, 'max_depth': 8}` our train test accuracies are optimal

Overall, I will consider the Boosted XGBoost (Train Test Split) as my best model because for this my Train Score is 94.95% and Test Score is 81.74% which is better as compared to other models. It clearly demonstartes that Boosting can help to get better results and on the top of that if we are Pruning it optimizes the result more.

Results for Dataset 2: Bank Marketing Data

Sr. No	Model	Train Test Split		Cross Validation (k =10)	
		Train Score	Test Score	Train Score	Test Score
1	SVC [Linear Kernel]	78.73%	75.40%	77.20%	70.20%
2	SVC [rbf Kernel]	100.00%	48.88%	100.00%	0.00%
3	SVC [Sigmoid kernel]	47.87%	47.61%	47.55%	40.39%
4	Decision Tree [Full Length]	100.00%	75.39%	100.00%	70.20%
5	Decision Tree [Pruned]	83.96%	81.79%	71.22%	87.50%
6	Boosting [XGBoost]	90.95%	78.28%	89.33%	79.81%
7	Boosting [Pruned XGBoost]	100.00%	82.43%	44.45%	100.00%

1. Analysis of results from SVM

As we can observe in the above table for the banking data we used 3 kernel methods and among those 3 rbf and Sigmoid performed very badly, there is a overfitting in both cases that is mode is too flexible.

Again, the Linear Kernel is giving the best result for Train and Test Split.

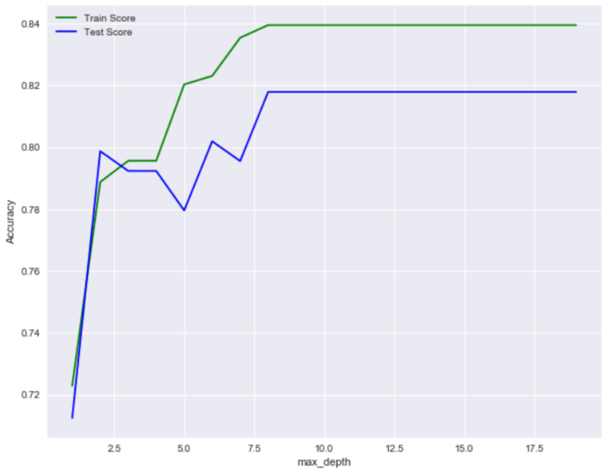
2. Analysis of results from Decision Tree (before and after pruning)-

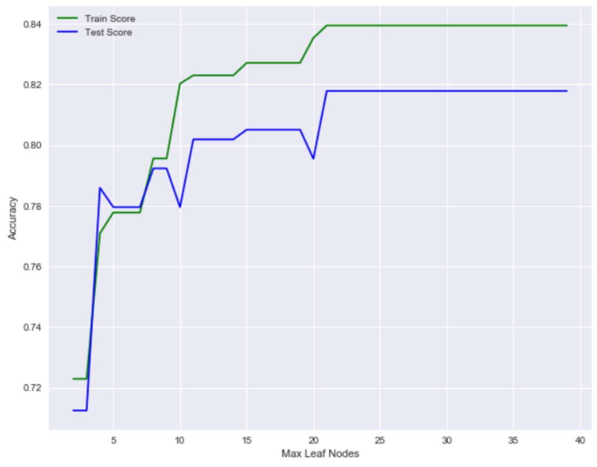
For both **Test Train Split and Cross Validation;** well decision tree is giving us some very INTERESTING INSIGHTS. When we are using Train and Test Split and running the Full-Length tree with Gini criterion (without giving any parameter values) we are getting the Train Score of 100% which is very obvious and due to overfitting, our test score is very poor 75.39% as compared to train score. Now as we know that we can play around with ***3 important Pruning Parameters*** which are: **Max Depth, Min Samples Leaf and Max Leaf Nodes.** This experimentation gives us the following values for our parameters-

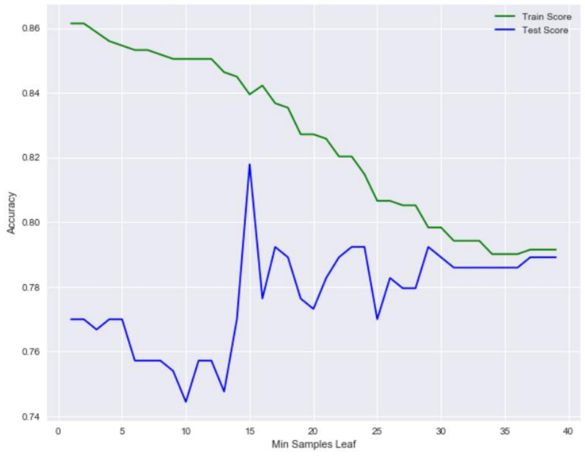
So basically, my Pruned Decision Tree is giving the best value for Train Test Split

Learning Curves and explanation-

```
{'max_depth': 8, 'min_samples_leaf': 15, 'max_leaf_nodes': 21}
```







So, when I trained my Pruned Decision tree by feeding these values my results got optimized as you can see in the above table. By limiting these parameters my decision tree will classify each value in an optimal manner.

Above plots explain the above reasoning (basically the highlighted portion, these plots basically show that how my accuracy is getting better based upon optimal values of these Pruning Parameters)

As mentioned above **at Max Depth = 8 and Max Leaf Nodes = 21, Min Sample Leaf = 15** Train and Test Accuracies are comparable.

3. Analysis of results from Boosting (before and after pruning)-

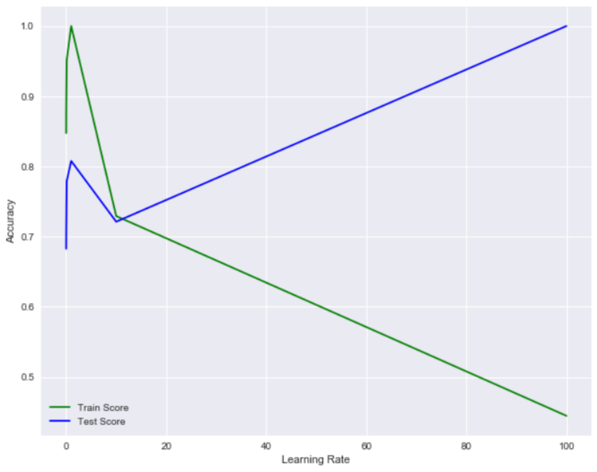
In both the cases for Train and Test Split and as well as CV, boosted version of the trees improve our result significantly. Boosting in itself avoids overfitting. We can find the best parameters tweaking learning rate max depth and number of estimators which is how we experimented with the Pruning in XGBoost.

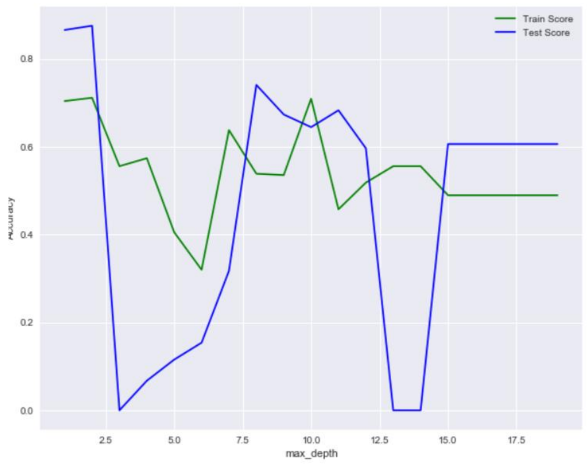
But for this particular **Bank Dataset, there is an interesting observation where I am getting 100% test accuracy for the pruned version of XGBoost Model after using Cross Validation.** Following parameters are playing an important role here:

```
{'learning_rate': 100, 'n_estimators': 50, 'max_depth': 6}
```

So, my learning rate is very high here, which means model learns fast on the training dataset and even by little training the model learns really well and gives better test accuracy on unseen data.

Learning Curves for this scenario (it follows the above explanation):





That’s why for me, Pruned version of XGBoost(CV) is the best model.

Conclusion:

Overall, I can say that using XGBoost is always beneficial in Machine Learning- it automatically handles the Regularization (it’s also called regularized form GBM, parallel processing, handles missing values, along with effective pruning of tree which results in optimal values of train and test scores.

****