Applied Machine Learning - Assignment 2

Anurag Sethi

Oct 11, 2019

1 Introduction

This project focuses extensively on the application of popular machine learning classification techniques support vector machines, decision trees, ensemble methods. While algorithm application is a small but critical part of the project **the foremost motivation of the project is to get learning outcomes by varying control parameters of the algorithms and their effect on performance metric**. In the subsequent sections, we do feature engineering, and data precrocessing, section two and the following subsections are the implementation of Support Vector Machines, Decision Trees, Boosting algorithms on the *UCI Appliance Energy Consumption* dataset along with cross validation implementation with each alogorithm. In the last subsection is perhaps the most critical one with model performance comparison. In section three same series of algorithm implementations and model performance comparison are done for *Kickstarter Campaign Success Prediction dataset*. In last section, section 4, the project concluding remarks which summarize the key learning takeaways of the project are presented.

1.1 Datasets

The project makes use of two very interesting datasets **Appliance Energy Consumption Dataset** hosted at *UCI machine learning repository* a popular archive portal for open datasets for academic use. We create the new variable 'spike' where the recorded consumption exceeds 150 Kwh. By definition, our dependent variable indicates to the user potential spike in the electricity usage, such a working model **helps the users to anticipate spikes and regulate the appliance usage to reduce the energy consumption and subsequently reduce their carbon footprint**.

> The other dataset used for the project is the **Kickstarter Success Prediction dataset** hosted on *Kaggle*. There are multiple factors associated with selecting this particular dataset over others; from business point of view it is an interesting problem, crowdfunding is a recent phenomenon and it would be great insight if we know the factors that lead to success in crowdfunding campaigns. There can be many lessons for a person or a company aiming to launch their campaign, that could maximize their chances of raising money. Furthermore, this data-set had ample scope of *feature engineering*, which is always a great exercise, a model built with rudimentary parameter optimization with intelligent feature engineering can often beat a model built with advanced parameter optimization on raw features. Thirdly, the data-set was fairly modest sized, 350k data points, 16 raw features and an optimal class balance , fitting the scope of the project. Due to computational constraints for algorithms like SVM, a subset of 50k data points is used in the project.

1.1.1 Data Preprocessing - Feature Engineering

UCI Appliance Energy Consumption : The dataset has 29 features to start with, there is dependent variable Appliances showing the energy consumption by appliances of the house. The other features can be broadly divided into four further categories, Weather Related Features,Humidity related features,Temperature Related Features: each category capturing the various entities at different parts of the house, also there are features like light, date, rv1 and rv2 which are put together as miscellaneous. In the pre-processing step numeric features are scaled using the z-score, implemented using the standard scaler operand inbuilt in python.

Engineered Variable Name	Description
Month	Descriptor labelled 1 to 5 representing month
Is Weekend	Flag if the day is a weekend or weekday
Time of Day	Section of the Day eg. night time, evening time, work hours etc.

We engineer new feature from date variable, which are relevant based on the understanding of the context. There are listed in the table below. The rationale behind this is that intuitively appliance consumption is affected by seasons (captured by month), day of week (captured by isweekend), and section of the day (captured by time of the day). For our usage in this project, we selected set of 21 features (reduced after pairwise correlation test), In the logistic regression implementation in assignment one, we noted this set of variables gave the best classification result, though this doesn't mean that this set is the best set of variables to go for svm, tree based implementations but is a good starting point. Final feature set used is lights, RH1, T2, RH2, T3, T4, RH6, T7, RH7, T8, RH8, RH9, Tout, Pressmmhg, RHout, Windspeed, Tdewpoint, rv2, month, is-weekend,time-of-day-encoded.

The dependent variable is defined as *spike alert*; here category is defined as 1 when the consumption for the hour exceeds 150 kwh. In the dataset, the class split at this mark is 88.5 and 11.5 percent, which is a reasonable split from the prospective of modelling ease.

Kickstarter Campaign Success Prediction: The original set of raw variables can be obtained from here (https://www.kaggle.com/kemical/kickstarter-projects). Over these features, we have engineering a set of new features, these help us inject some domain knowledge in our prediction.

Engineered Variable Name	Description
Duration	Descriptor labelled 1 to 5 representing month
Is Weekend	Flag if the day is a weekend or weekday
Launched year, quarter	Year and seasonality information derived from date of launch
average amount per backer	Average amount each patron has given till now
goal level	Money campaign is targetting divided by mean goal of projects in the category
competition quotient	Total same category projects lanuched in the same month with given campaign
syllable count	Number of syllables in the campaign name

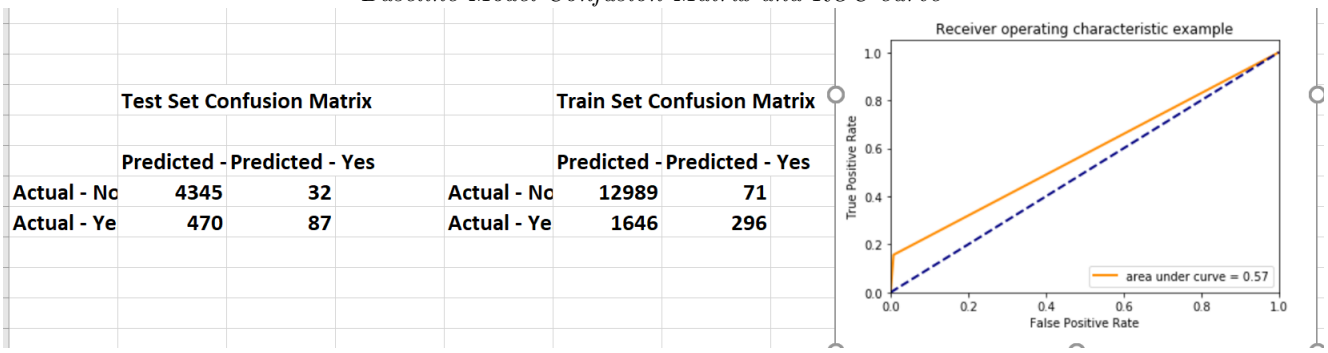
The categorical columns are converted using one hot encoding and we get 42 features in our set. To reduce the dimensionality furthur, univariate analysis of variance is performed with the dependent variable and 12 features are retained for modelling. The 12 features obtained are number of backers, average amount per backer, goal level, duration, quarter (all dummies), main category = 'some cateogry' exists (some dummies). In the next section we begin modelling different classification algorithms. Note both datasets are split in the 75-25 train -test split ratio.

2 Appliance Energy Consumption

In the subsequent subsections, we deeply explore various classification algorithms and predict spike in appliance energy usage. We see we have 11.6 % of classes marked as positive (2298 out of 19735) where the usage exceeds 150Kwh. We vary the control parameters and see the effect on training and out of sample errors. We also present ROC curves and confusion matrices as our means of evaluating the models.

2.1 Appliance Energy Consumption: Support Vector Machines

Baseline Model Confusion Matrix and ROC curve

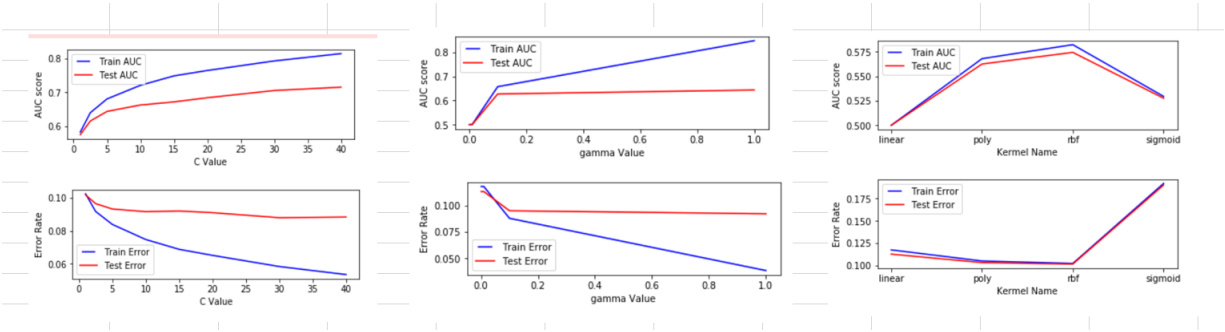


	Test Set Confusion Matrix			Train Set Confusion Matrix	
	Predicted -	Predicted - Yes		Predicted -	Predicted - Yes
Actual - No	4345	32	Actual - No	12989	71
Actual - Ye	470	87	Actual - Ye	1646	296

Baseline Model Confusion Matrix and ROC curve

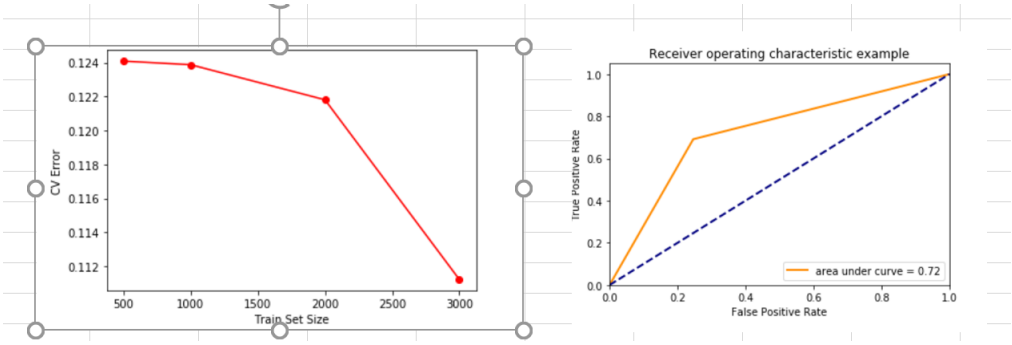
The first model implemented is support vector machines. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. Naturally for both appliance energy prediction and Kickstarter campaign success prediction

we will optimize parameters associated with the hyperplane which is the seperation boundry. **The report for both the datasets for all algorithm implementations follows similar blueprint, one baseline model, learning curves with AOC and Test Error, and optimized model by k-folds cross validation using parameter grid search**. Only in SVM, grid search wasn't implemented due to computational constraints. The baseline model performs poorly despite the fact it has high accuracy, Athough there are True Negatives there are lots of False Negatives as well, hence corresponding AOC score is very low.



Learning curves for different params

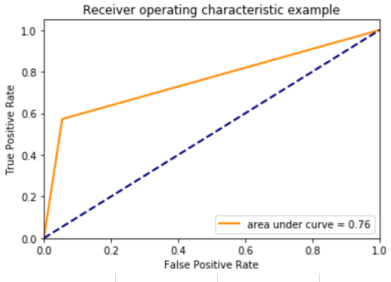
In the above plots we can see the learning curves with the varying values of c and gamma. C is the error penalty parameter of SVM and gamma is the coefficient of kernel, these parameters are more discussed in section 3.1. Both Area under curve and Error Test improve with increasing both these parameters. Also we can see train set size decrease E Cross Valid. The ROC curve below is replotted for a new model, created after cross validation and obtaining the improved parameters for gamma and c value. Here rbf kernel, radial basis kernel is the optimal kernel as we can see learning curve top right, it has least out of sample error and maximum area under the curve. The subsequent model built improves the AUC values significantly as it can be seen from the plot.



2.2 Appliance Energy Consumption: Decision Trees

Decision trees are easily the most intuitively simple to understand of all models presented today. We use sklearn library implementation of decision tree in python. **Before we begin in detail with our model we build a decision tree with default sklearn parameters as our baseline model. We get training set accuracy of 1, test set accuracy of 0.90 (which is very high) but a relatively poor AUC score of 0.77**

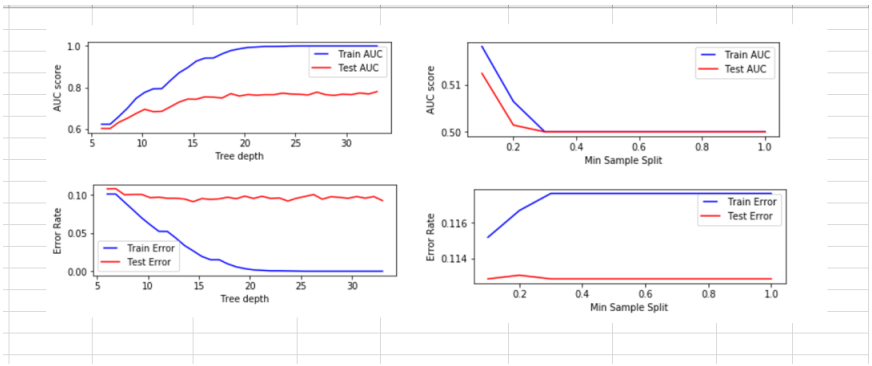
	TRAIN SET CONFUSION MATRIX			TEST SET CONFUSION MATRIX	
	Predicted - No	Predicted - Yes		Predicted - No	Predicted - Yes
Actual - No	13060	0	Actual - No	4137	240
Actual - Yes	0	1741	Actual - Yes	238	319



left: Test Error with different train size, right: ROC for optimized model

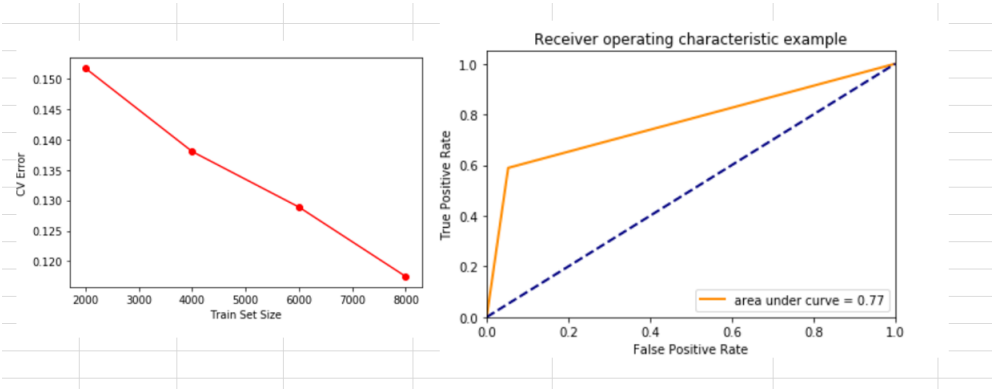
The decision tree classifier in sklearn give a lot of scope for tuning the levers of the algorithm, better called control parameters. Here we plot two performance parameters, train/test error and Area under the curve with

respect to two control parameters; max depth and minimum sample split. **We can use these parameters to prune to tree,** in other words control how deep the tree will grow. In our baseline model, the tree reached its' maximum length of 33, such a classifier will have the tendency to overfit. We first see the definition of max depth, and minimum sample split, max depth is the threshold for the maximum allowable depth of the tree. Minimum sample split is the minimum number of data points that are required to make a decision at any decision notes, hence **intuitively both control parameters control bias variance tradeoff**; we can increase bias by decreasing max length and increasing minimum sample split. By doing this we are also pruning the tree.



Confusion matrix and roc for baseline model

We vary the tree depth from 5 to 33 and observe AUC, in sample and out of sample errors, while train error is expected to decrease while increasing max depth,the test error ceases to decrease after around max tree depth 10. Correspondingly we see that around minimum sample ratio 0.3 there is no improvement in Area under the curve as well the out of sample error. It means it would be a good fit to have the ratio at 0.3 means, if there are less than 30 percent datapoints in a node there is no need to further split them. This number is slightly higher than expectation, one of the reason could be due of class imbalance, the decision tree could unnecessarily split the already classified dominant class possibly leading to overfitting. Below we plot the out of sample error with respect to size of the training data, here 20 percent of the data points are seperated and 4 fold cross validation is implemented. Here a sharp decline can be seen on out of sample error with increasing train size. 4 fold cross validation is chosen keeping in mind computational constraints and relatively small dataset.



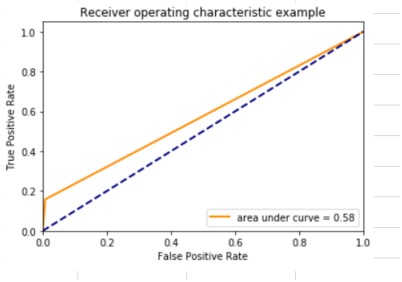
left: Test Error with different train size, right: ROC for optimized model

In the last step grid search cross validation is implemented and a pruned tree is implemented. In Grid search we provide a parameter grid and subsequently out of bag error is calculated after implementing class validation. ('criterion': 'gini', 'maxdepth': 5, 'maxfeatures': 21, 'minsamplessplit': 0.2599) *Here a pruned tree is constructed.* The library implementation gives the liberty of selecting the scoring metric, we have chosen AUC over other alternatives like False Positive rate, accuracy because, our baseline model has high accuracy and comparatively low AUC, all combinations of the grid of parameters (referring the code 96 different models) are executed, the combination with maximum CV Auc is reported above, we use this model fit to get AUC on out of sample data, and see slightly improved result at 0.78. Two important things to note here are that, we did not choose information gain or gini, accordingly to CV analysis, gini was suggested based on the higher AUC. However, in most cases, both the metrics gives similar results. **Perhaps the biggest takeway** is that the tree depth is largely reduced to 5, when we studied AUC and out of sample error variation independently wrt. to depth, we saw the tree showing 10-15 as the optimal value. This is an important lesson as when parameters are optimized in a group, they may converge differently compared to when they are optimized independently due to effects from other tuned parameters.

2.3 Appliance Energy Consumption: Boosting

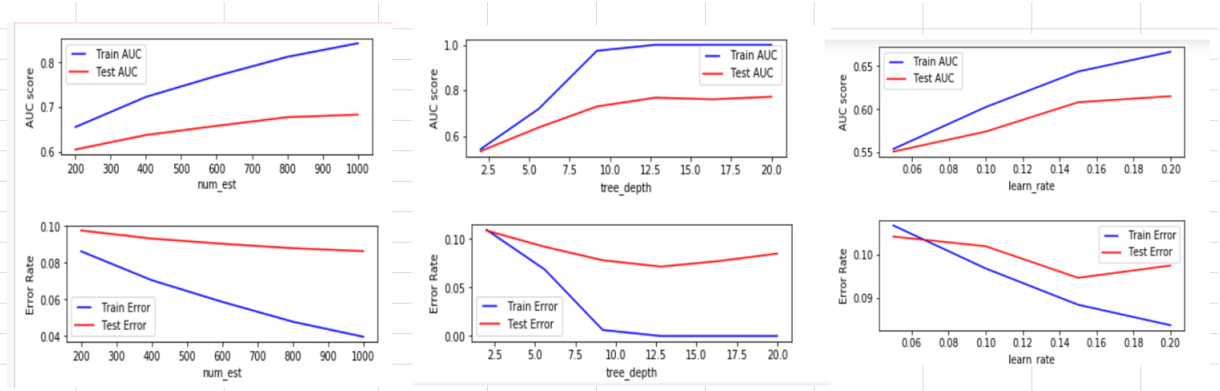
In the previous section individual learners were studied, the underlying fundamentals of boosting algorithms are such that they rely on individual weak learners to come together to form one robust strong classifier. The baseline model gives train set accuracy of 0.903 and test set accuracy of 0.890, this is a good output, however alarmingly the AUC is as low as 0.57. This is due to class imbalance, the baseline model in order to improve accuracy, predicts disproportionately high number of classes as zero, this results in a large number of false negatives, leading to low AUC. Below is the confusion matrix and ROC curve test set of baseline model.

	Predicted - No	Predicted - Yes
Actual - No	4344	33
Actual - Yes	469	88



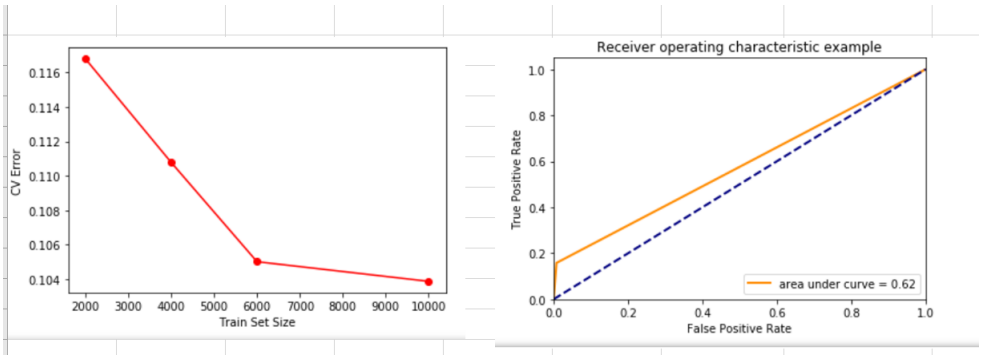
Confusion Matrix and ROC for baseline model

In Gradient boosting we take 3 instead of 2 parameters in the learning curves (as in decision trees) and see the effect on AUC and Out of Sample Error. **A key observation below is that the model does not overfit upon increasing the number of weak learners**. Also the test set performance is constant after a certain depth of the weak learner trees is reached. This is a general tendency of boosting algorithms, they don't generally overfit upon increasing the number of weak learners, in such a scenerio the learning of the classifier is gradual, marginally diminishing but still on the increasing trend.



Learning curves for different parameters

Similarly, looking at the learning curve with respect to learn rate, top right we can see that upon increasing the learning rate model performance out of sample increases, it is because for higher values of learn rate the model converges to optimal learner quickly without induced bias. Below we can see the learning curve with respect to number of training samples, it shows expected trend. Hence it can be inferred that in boosting algorithms one can afford to large number of classifiers and still significantly avoid the risk of overfitting, however strength of the individual classifiers may be detrimental to out of sample performance. After grid search following set of parameters were found to be optimal. ('learningrate': 0.15, 'maxfeatures': 'sqrt', 'nestimators': 400).



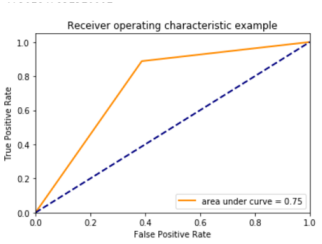
left: Test Error with different train size, right: ROC for optimized model

3 Kickstarter Campaign Success

In kickstarter campaign success, in the raw features, we do extensive features engineering as explained in the previous section. Campaigns suspended or cancelled are also labelled as unsuccessful, in total 16465 out of 40000 data points are class 1, that is successful project. This dataset has a more balanced class profile, what effect it has on the predictive process would also be interesting to see. Corresponding optimal boosting model is plotted below for ROC, we see a significant increase in area under the curve from the baseline model.

3.1 Kickstarter Campaign Success: Support Vector Machines

Support vector machines are sophisticated linear seperator's in a n dimension vector space, aimed to find the best classifier boundry between the two classes. **Like the learning curves of the tree based algorithms were plotted against ensemble parameters or parameters of individual tree based classifiers, here we tend to optimize the decision boundry, seperation and the factors influencing it like the shape of the kernel**. Foremost the baseline model is plotted, it gives fairly good results with Area under the curve as 0.75 and both train and test set error around 0.78.

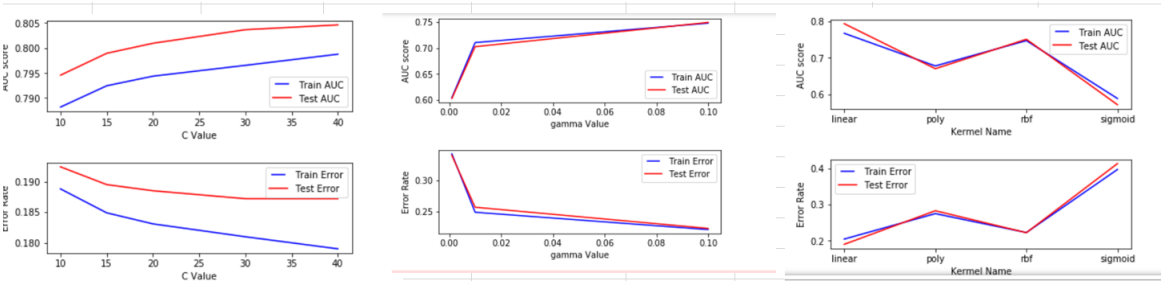


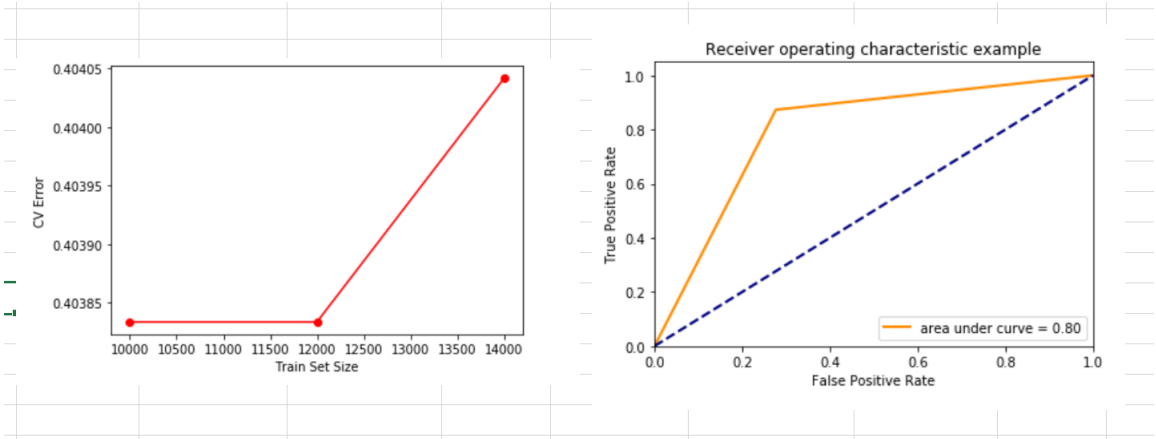
	Test Set Confusion Matrix	
	Predicted - No	Predicted - Yes
Actual - No	2446	1547
Actual - Yes	676	5333

	Train Set Confusion Matrix	
	Predicted - No	Predicted - Yes
Actual - No	7096	5030
Actual - Yes	1646	16228

ROC and Confusion matrix for baseline model

The three parameters presented herea are C value, gamma and kernel, there are few standard kernerls provided by sklearn, it is important to note that different kernels are suited for different kinds of problems. **It is not straightforward to visualize the shape that would fit in higher n dimensional shape** The way forward is to try the plot a learning curve for the available kernels and check the out of sample performance and area under the curve. We can see that both in terms of AUC score and test error linear kernel performs best and sigmoid kernel performs worst. C is the parameter for penalty for error term, less C would intuitively mean more bias in the model, Hence on increasing C value the learner becomes better without overfitting in general. Similarly gamma is the kernel coefficient, increasing it appears to improve the model performance. The exact role of gamma is not clear and was hard to find on internet a decent explanation describing this parameter.





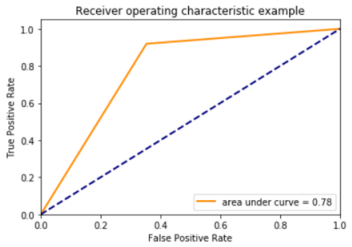
ROC and Confusion matrix for baseline model

Above left we can see the cross validation error largely remains unchanged (the scale of the graph presented is misleading). For SVM we have decided to not implement grid search to find optimal parameters because of computational constraints. In place of that based on individual cross validation learning curves and built a model for the most appropriate values of kernel gamma and C. (linear, 0.15, 60) and got the following ROC curve, much improved version over the baseline model.

3.2 Kickstarter Campaign Success: Decision Trees

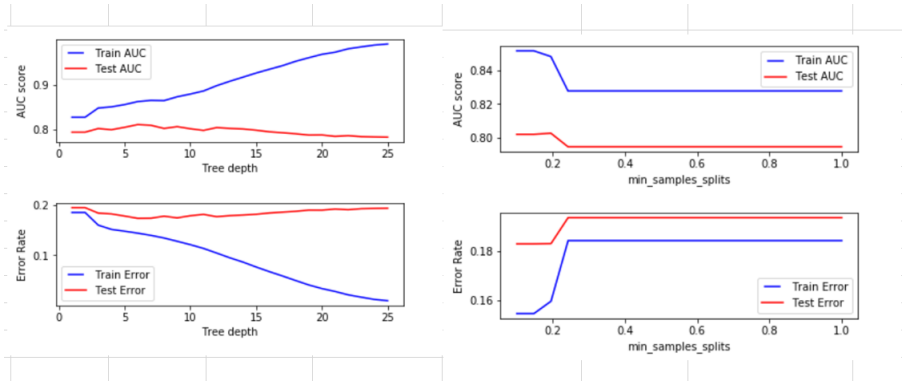
Decision trees are easily the most intuitively simple to understand of all models presented today. We use sklearn library implementation of decision tree in python. **Before we begin in detail with our model we build a decision tree with default sklearn parameters as our baseline model. We get training set accuracy of 1, test set accuracy of 0.90 (which is very high) but a relatively poor AUC score of 0.77**

	TRAIN SET CONFUSION MATRIX			TEST SET CONFUSION MATRIX	
	Predicted - No	Predicted - Yes		Predicted - No	Predicted - Yes
Actual - No	12430	0	Actual - No	2656	1466
Actual - Yes	3	17567	Actual - Yes	481	5397



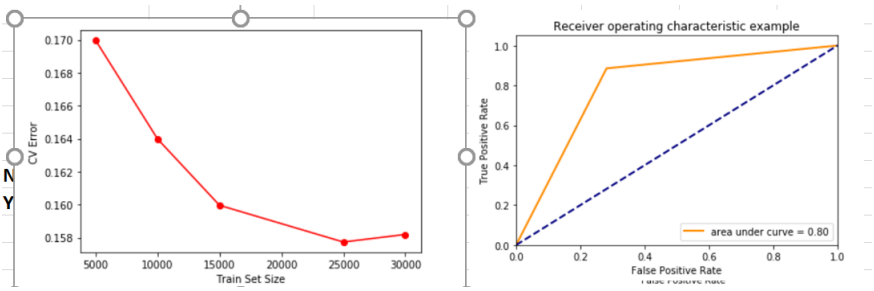
ROC and Confusion matrix for baseline model

Similar to before we see the out of sample results by varying some of the control parameters of the decision trees, max depth and minimum sample split. **The intuitive meaning of these two parameters and how they are instrumental in levering the bias variance trade-off and tree pruning has been previously covered in section 2.2** Here we can see that the Test AUC and Test Error are optimal at tree depth 8, implying increasing the allowable depth further may result in over-fitting. **Interestingly, by varying minimum sample split**, we see interesting results, beyond a certain threshold (above 0.2); both errors flatten out and the model doesn't change. It should be noted that the most optimal model in this case has low threshold of minimum sampling (this could be possible because of a more balanced class profile, there is more scope of further breaking down the tree.)



Learning curves for different parameters

In the end after running grid search we arrive at a set of optimal parameters after running all possible models from the grid. ('criterion': 'gini', 'maxdepth': 9, 'maxfeatures': 21, 'minsamplessplit': 0.1); again the cross validation instructs us to use Gini as the split criteria and limit depth to 9, split to a low value of 0.1 which was expected. As it can be seen below (right) this model shows noticeable improvement in AUC score to 0.8 (AUC was the optimization metric in grid search). Below left the size of the train set and its' effect on test error can be seen, here 10 CV cross validation is implemented, since this dataset is bigger, hence more number of folds, one can see beyond 25000 the out of sample error increases and the model develops the tendency to overfit.

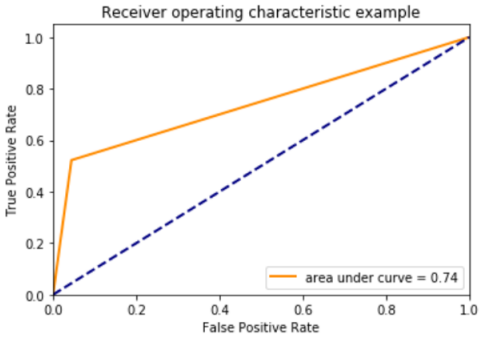


left: Test Error with varying train size, right : ROC for optimized model

3.3 Kickstarter Campaign Success: Boosting

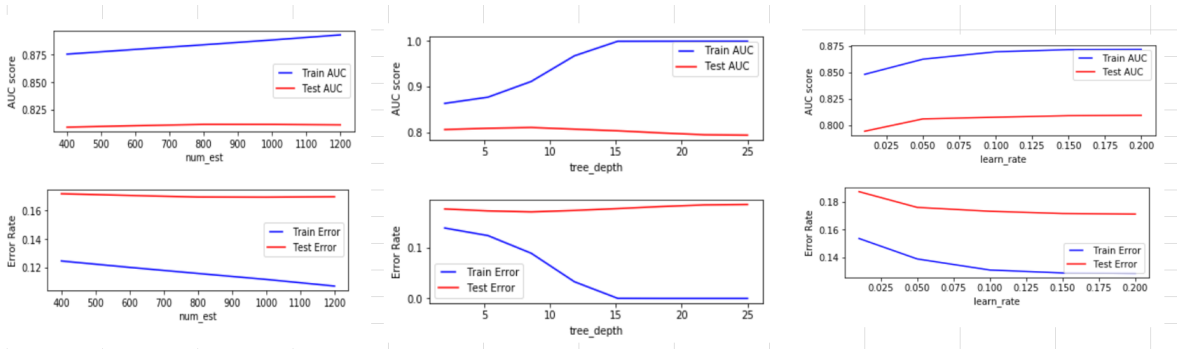
We have previously implemented boosting on the energy consumption dataset. While applying boosting algorithms to kickstarter campaign success prediction we see interesting results. The baseline model which is built without tuning itself gives good results, with Area under the reverse order characteristic curve as 0.74, test accuracy 0.70.

	Predicted - No	Predicted - Yes
Actual - No	2884	1268
Actual - Yes	495	5383



ROC and Confusion matrix for baseline model

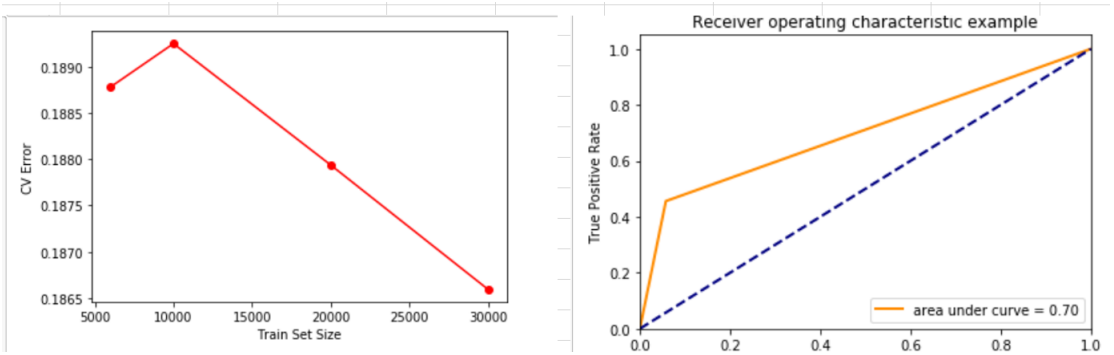
As compared to last appliance energy problem, this one gives more leeway to have individual learners to be of more depth and be better learners. One of the reasons could be class distribution, as said before it is harder problem to solve, with more train size, bigger feature set and most importantly class distribution. It is just there is more scope of *learning* from the train data. Similar to before curves plotted wrt learn rate show a grdual improveent with higher learning curve values. It is also important to note that among the three features max depth is again used for pruning.



learning curves for different parameters

Below left is the learning curve to visualize the change in the out of sample error with increasing sample size, there is an abnormality in the beginning steps with a spike the the test error, this could be due to the fact that training sample is small at that step as compared to the cross validated step with is 20 percent of the entire dataset. Later we see the usual behavior with the test error going down. The ROC curve on the bottom right showed the pruned tree ROC curve after doing the gridsearch cross validation for this boosting model. It is interesting to see that the test Area under the curve is less than the unpruned baseline model. It could be because pruning leads to bias in this model and tree depth is required. But this is counter-intuitive since this approach should also result in high cross validation error. Another explanation could be the cross validation sets are not representative of the test set. Following are the set of parameters we optimised using gridsearch.

(learningrate=0.2, maxdepth=3,maxfeatures='sqrt') A control variable not used in the learning curves above is max features, which is basically a methodology to select samples at the decision points. There are multiple possible options for this parameter like log base 2 of total sample count but as determined by grid search sqrt of sample count is the one used in the pruned tree.



left: test error for different train size, right : roc for optimized model

4 Model Comparison and Learning Outcomes

- Along with learning implementation of machine learning algorithms, one of the biggest focus of this assignment was the technique of cross validation, it is an invaluable mechanism of knowing on how the model will perform out of sample without going to the test data. It is also a leverage to optimize various controlling levels of the model, basically the input parameters using cross validation as a technique. A lot of model improvement can be done by engineering better features, but there is some scope in tuning parameters as well which was evident in the models built in this project. **Comparing the three models for two datasets maximum improvement in performance after cross validation and parameter tuning was observed in support vector machines**. One of the reasons can be that SVM is very sensitive to kernel coefficient, and error penalty term, in these particular modelling problems, these parameters have a more direct effect on the classification as compared to parameters of tree based classifiers.

- Compared to baseline models decision trees didn't require much tuning and on the default parameters did just as good a job. In boosting based models we were least prone to overfitting, it can be inferred in a general sense looking at the learning curves and cross validation scores.

- **One of the most important learning takeaway** from this project was the importance of selecting right evaluation metric and time of scoring and cross validating the models. For instance, the appliance electricity prediction, there was some imbalance in class (¿88 percent negative class) and the size of the dataset was small as well. A classifier could predict all negatives and have a fairly good accuracy score in such a scenerio (which was seen in case of SVM). Though such a solution is trivial and would not suffice the requirements. Hence it was better to have Area under the curve as more appropriate metric. It can significantly minimise those false negatives and was needed.