Introduction

The given data set pertains to the historical demand for bike-sharing in Seoul. The data contains the count of rented bikes on every hour of a given day from December 2017 to November 2018. This data also provides different predictors that could explain the variance in demand of bikes for rent such as temperature, humidity, wind speed, Visibility, Dew point, Solar Radiation, Rainfall, snowfall, season, and whether the day is a holiday or functioning day. The objective is to predict how the bike renting going to be give the features or conditions such as climatic and date and season etc., so that proper planning can be done by the business teams.

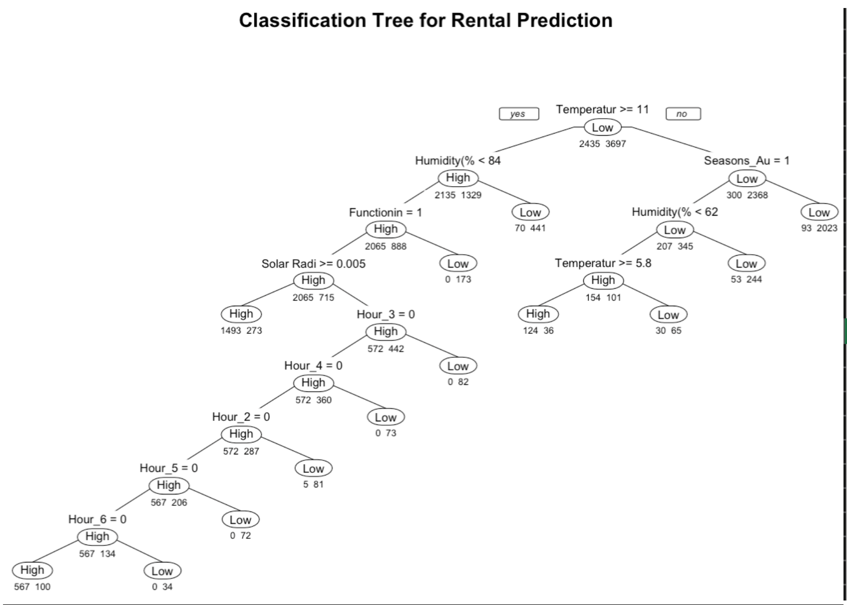
Binary classification threshold:

We can treat this problem as binary classification problem by converting the output variable ie., count of bikes into high or low. We can take the mean of the count of bikes rented(704.6) as the threshold and consider all the observations with count of bikes more than mean as high and observations with count of bikes less than mean as low.

Decision Tree Classification:

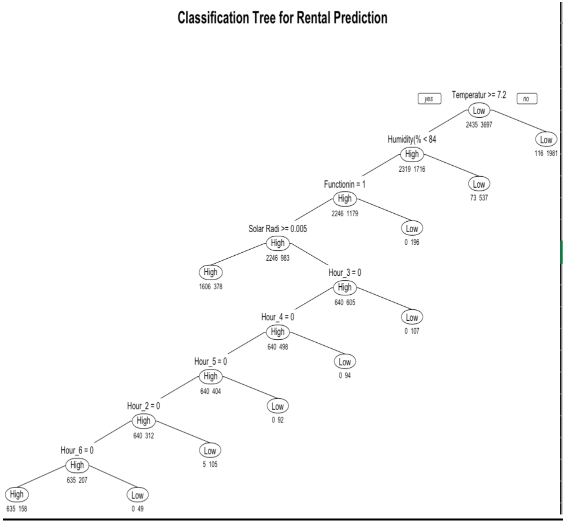
Classification algorithms can be easily applied to this problem as classification algorithm can have both categorical and numerical variables as inputs and usually categorical variable as output. Decision tree being one of the most used classification techniques is simple to use and interpret compared to other classification algorithms with easily understandable decision tree plots and hyperparameters. After converting the output variable to a categorical variable, we can safely use the decision tree algorithm to predict the bicycle rentals (High vs Low) given the other dependent variables. “Rpart” package has been used in R to model the decision trees for this problem with other ancillary packages to draw the decision tree etc., such as rpart.plot, caret.

The most important hyperparameter in decision tree algorithm is the metric that is used to measure the quality of information after any split. Two main such metrics are “Gini” and “Entropy”. While the range of Gini measure lies between 0 and 0.5, range of entropy lies between 0 and 1 in binary classification models. Though both try to measure the information gain in any split and go with the split that gives the highest information gain, calculation methodology is slightly different. Entropy uses logarithms, which put some pressure on algorithm leading to higher run time, whereas Gini uses basic calculus for the same which is not time consuming comparatively.



```
                              Training data        Testing data

             Accuracy : 0.8924               : 0.8661
               95% CI : (0.8843, 0.9)        : (0.8524, 0.8789)
  No Information Rate : 0.6029               : 0.5864
  P-Value [Acc > NIR] : < 2.2e-16            : < 2e-16

                Kappa : 0.7777               : 0.7254

Mcnemar's Test P-Value : 9.887e-10           : 0.02887

          Sensitivity : 0.8969               : 0.8574
          Specificity : 0.8894               : 0.8722
       Pos Pred Value : 0.8423               : 0.8255
       Neg Pred Value : 0.9291               : 0.8966
           Prevalence : 0.3971               : 0.4136
       Detection Rate : 0.3562               : 0.3546
 Detection Prevalence : 0.4229               : 0.4296
    Balanced Accuracy : 0.8931               : 0.8648

     'Positive' Class : High                 : High
```

Fig 1.1 Decision tree with Gini index & confusion matrix on train and test data



```
                              Training data        Testing data
             Accuracy : 0.881                : 0.863
               95% CI : (0.8726, 0.889)      : (0.8493, 0.875
  No Information Rate : 0.6029               : 0.5864
  P-Value [Acc > NIR] : < 2.2e-16            : < 2.2e-16

                Kappa : 0.7572               : 0.7223

Mcnemar's Test P-Value : < 2.2e-16           : 9.011e-11

          Sensitivity : 0.9203               : 0.8914
          Specificity : 0.8550               : 0.8430
       Pos Pred Value : 0.8070               : 0.8002
       Neg Pred Value : 0.9422               : 0.9167
           Prevalence : 0.3971               : 0.4136
       Detection Rate : 0.3655               : 0.3687
 Detection Prevalence : 0.4529               : 0.4608
    Balanced Accuracy : 0.8877               : 0.8672

     'Positive' Class : High                 : High
```

Fig 1.2 Decision tree with Entropy index & confusion matrix on train and test data

The above two figures (1.1, 1.2) depict the decision trees drawn using Gini index and using Entropy index. It is clear to see that algorithm with Gini index is performing well with higher training and testing accuracies compared to Entropy index. Also, it is worth wile to note the time taken by each index which are 0.04975605 secs for gini index and 0.05041599 secs for entropy index. This might have huge impact when working on big data sets with billions of records. Also, apparently it is not worthwhile to spend more time with entropy having lower performance than gini index.

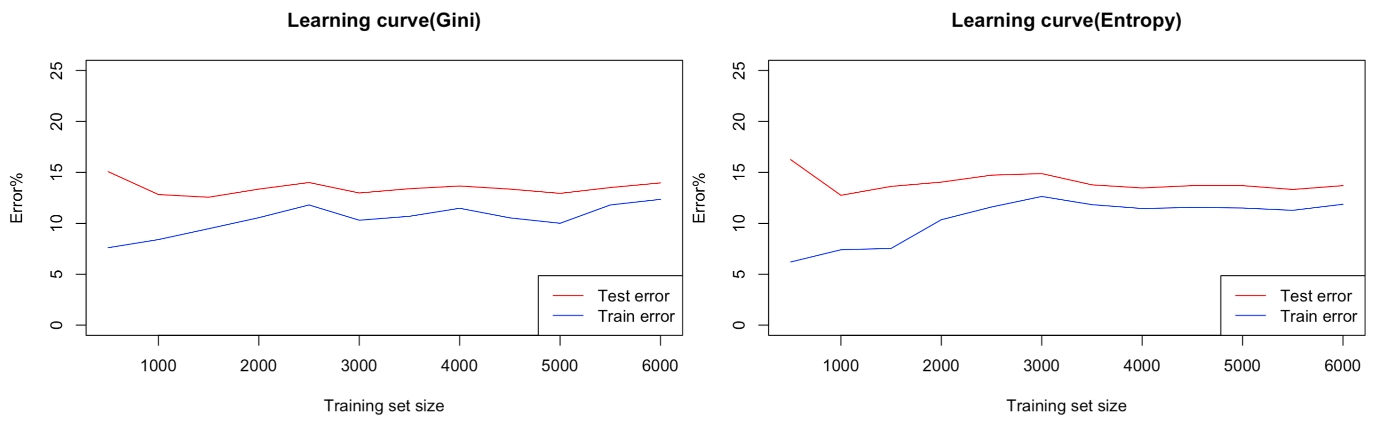


Fig 1.3 Learning curves for Gini and Entropy measures

Also it is interesting to note the learning curves from fig 1.3 for both the measures. Lerning in both the measures is done in an almost similay way. This shows that using either of the measures will have almost similar results irrespective of the size of the dataset we have. Hence as our model with Gini measure performed faster and better, we are using this for the rest of the experiments.

Experimentation on learning curve on depth of the tree:

We can control the maximum depth of the tree by modifying the maxdepth parameter in our model. Below learning curves captures how training and testing errors are at different levels of depth of the tree.

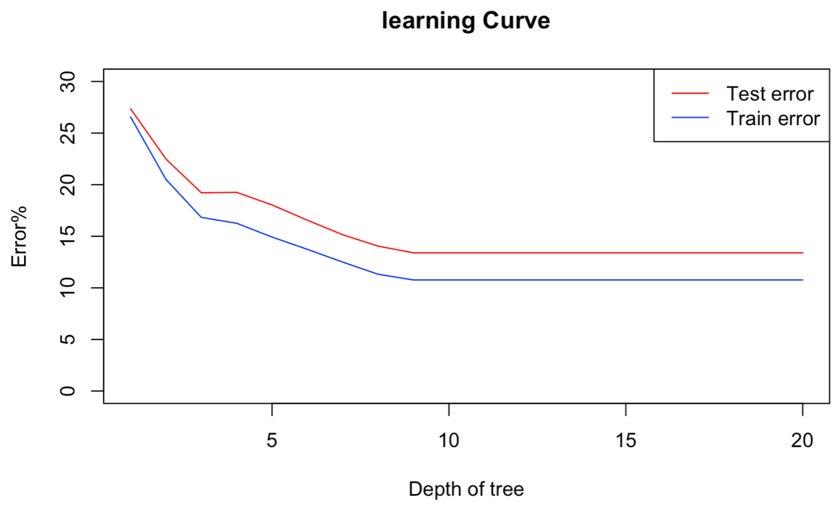


Fig 1.4 Learning curves for Gini and Entropy measures as function of depth of tree

This shows that as the depth of the tree increase, error came down leading to more generalization of the data. Apparently 9 levels of depth is optimal for this data set as from 10th level error doesn't seem to go down. It is interesting to note that both decision trees drawn with ginin and entropy index in fig.1.2 and 1.3 are having 9 levels as well.

Pre-Pruning:

We can use pre pruning techniques to reduce the complexity of the tree to improve generalization and accuracy. One of such techniques is pre determining the minimum number of observations required to split the node. If the observations in the node are less than the determined limit, further splitting is not allowed. Experimentation is done by taking minimum split as 500 and 300.

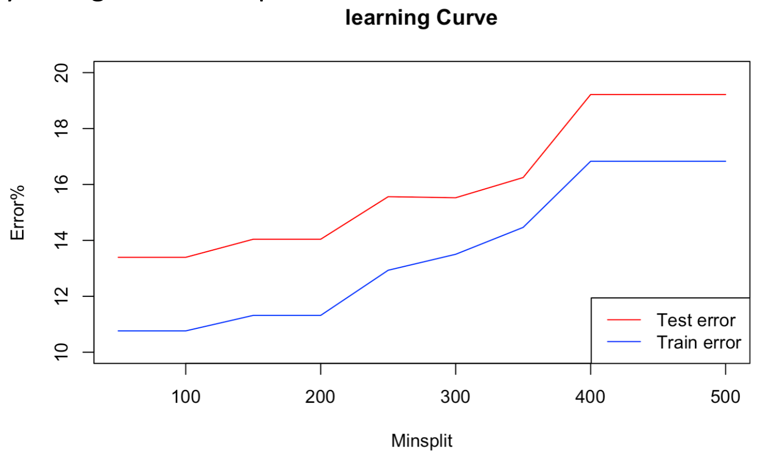
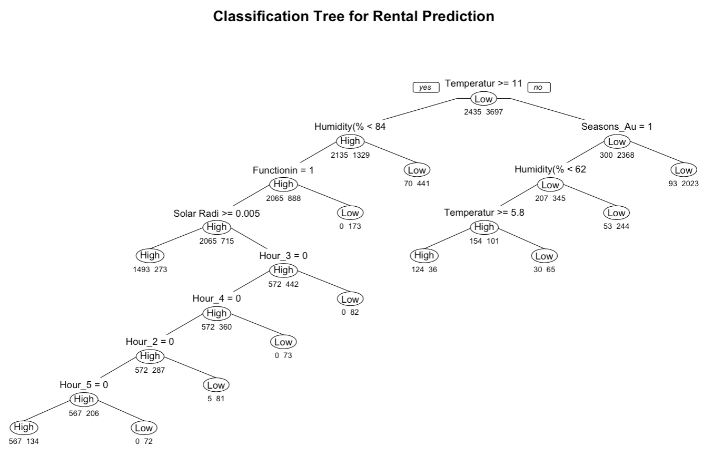


Fig 1.5 Learning curve as function of minsplits

It can be observed from above learning curve that until 200 minsplit, train and test errors are low. Even though the gap between test and train error is higher compared to cases between 300 and 400 split, overall error is high after 200 splits. Hence ideal minsplits that need to be chosen shall be 200.



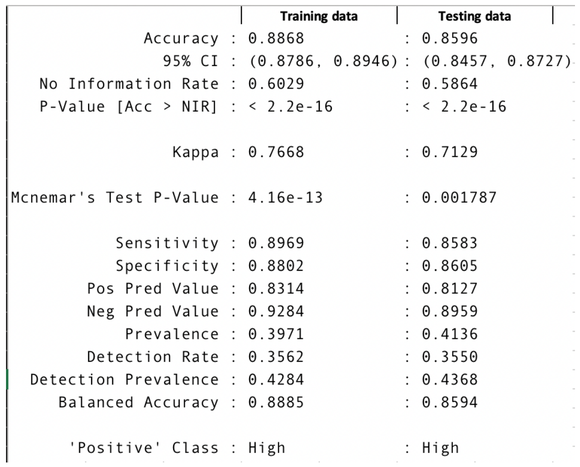

```
                         Training data          Testing data
          Accuracy : 0.8868               : 0.8596
            95% CI : (0.8786, 0.8946) : (0.8457, 0.8727)
   No Information Rate : 0.6029           : 0.5864
   P-Value [Acc > NIR] : < 2.2e-16        : < 2.2e-16

             Kappa : 0.7668               : 0.7129

Mcnemar's Test P-Value : 4.16e-13         : 0.001787

       Sensitivity : 0.8969               : 0.8583
       Specificity : 0.8802               : 0.8605
    Pos Pred Value : 0.8314               : 0.8127
    Neg Pred Value : 0.9284               : 0.8959
        Prevalence : 0.3971               : 0.4136
    Detection Rate : 0.3562               : 0.3550
  Detection Prevalence : 0.4284           : 0.4368
   Balanced Accuracy : 0.8885             : 0.8594

     'Positive' Class : High              : High
```

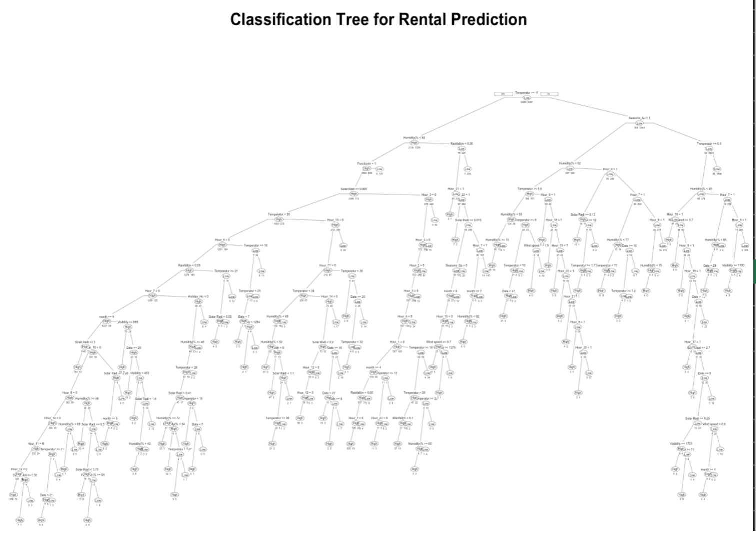
Fig 1.6 Decission tree and confusion matrix at 200 min splits

The above figure shows the actual tree drawn with minsplit set to 200 and the corresponding confusion matrix. Compared to model without specifying any minsplit, this model is having little high training accuracy but little less testing accuracy. This is further discussed in learning curves comparison section.

Post-Pruning with cross validation:

Decision tree can be pruned after drawing a tree with maximum size with k fold cross validation. Here 10-fold cross validation is used and a biggest possible tree is drawn. The model calculates the cross-validation error at different possible splits and assigns a complexity parameter (CP) to it. CP is another hyperparameter that controls the complexity of the decision tree. If the fit of the tree doesn't improve by factor of CP, the tree won't be drawn further.

We can select the complexity parameter at which cross validation error is least and use this to prune our big tree to small tree. As can be seen from the CP table generated by model, cross validation error is least when tree splits are 138. Hence we use this CP at 138 nodes and least cross validation error to rest the parameters to draw an optimum tree which is as follows(fig1.7). As anticipated, this could be the most optimum tree with highest training and testing accuracies.



	Training data	Testing data
Accuracy	0.9587	0.8965
95% CI	(0.9535, 0.9636)	(0.8842, 0.9079)
No Information Rate	0.6029	0.5864
P-Value [Acc > NIR]	<2e-16	< 2.2e-16
Kappa	0.9139	0.7853
Mcnemar's Test P-Value	0.5295	0.006362
Sensitivity	0.9503	0.8537
Specificity	0.9643	0.9267
Pos Pred Value	0.9460	0.8915
Neg Pred Value	0.9672	0.8998
Prevalence	0.3971	0.4136
Detection Rate	0.3774	0.3531
Detection Prevalence	0.3989	0.3961
Balanced Accuracy	0.9573	0.8902
'Positive' Class	High	High

Fig 1.7 Decision tree with best CP and confusion matrix.

<u>Comparison of learning curves for pre pruning vs post pruning vs no pruning(Gini)</u>

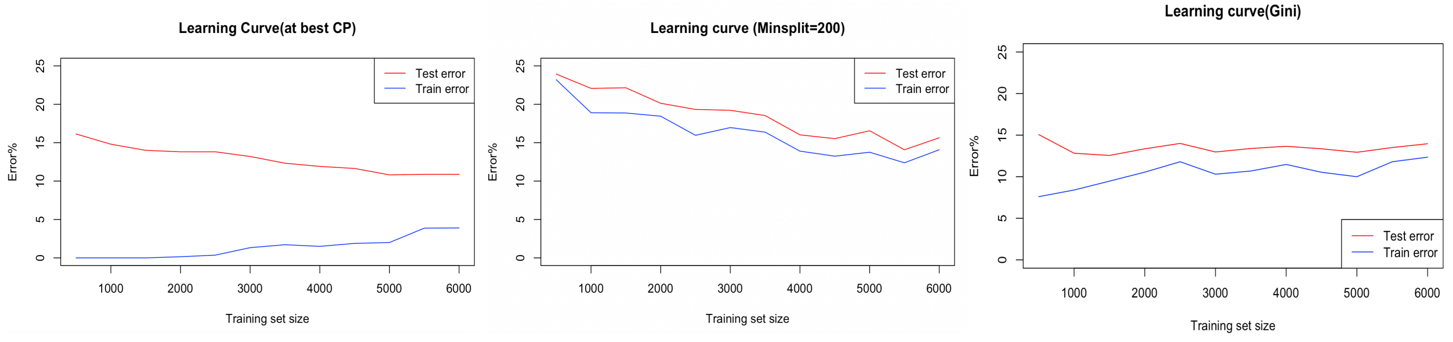


Fig 1.8 Comparison of learning curves.

The above figure shows the learning curves for pre pruning, post pruning and no pruning(Gini) models. Post pruning (at best CP) curve is having low overall training and testing errors but gap between them is high. This can be a case of high variance or overfitting to some extent. This can be solved or improved if we increase the number of observations to train the data. Learning curves of pre pruning and no pruning (Gini) are having less gap between train and test errors but overall error rates are high. This can be case of bias or underfitting. This can be solved if we can introduce more independent variables/features to the dataset.

Support vector Machines(SVM):

SVM algorithm can be easily applied to this problem as this is a binary classification problem and the features are numerical variables.

The SVM has 2 types of classification models – C-classification and nu-classification. Though both try to address the same problem they each have different hyper parameter named C and nu. Range of C has no boundaries whereas nu's range lies between 0-1 making it easy to modify to get better results.

C- Classification and nu classification with linear kernal:

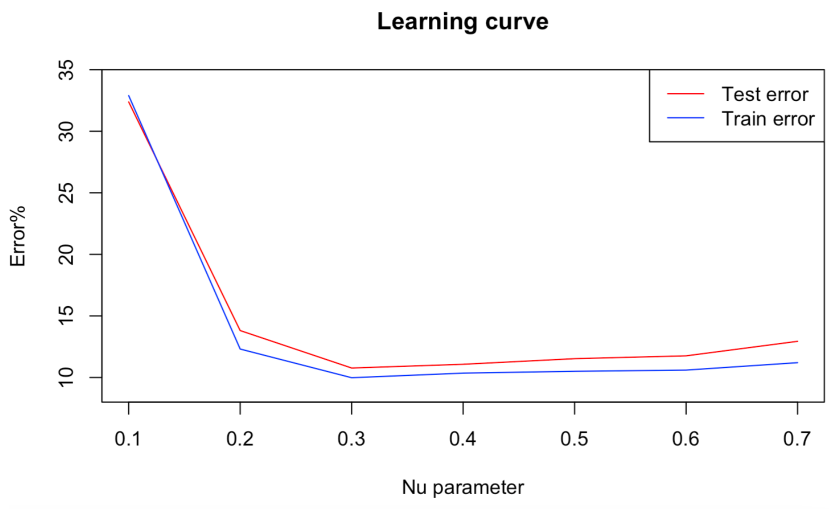
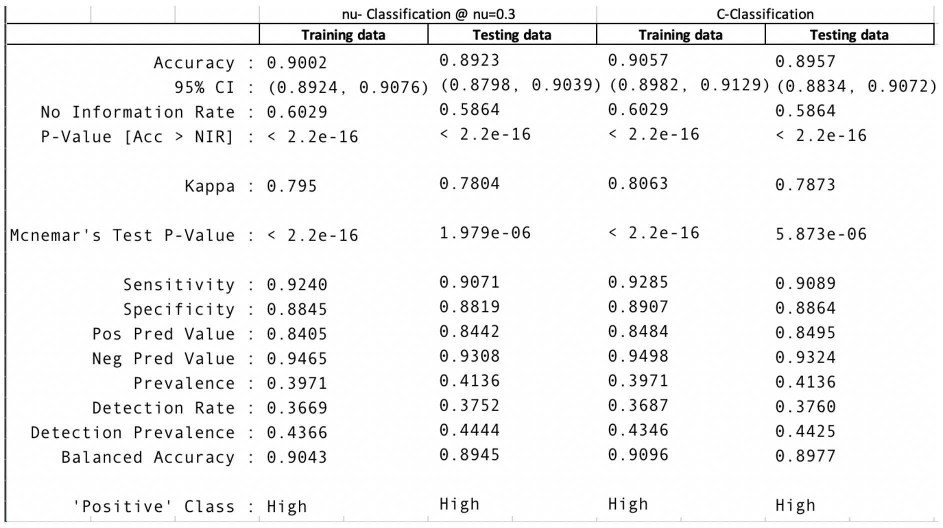


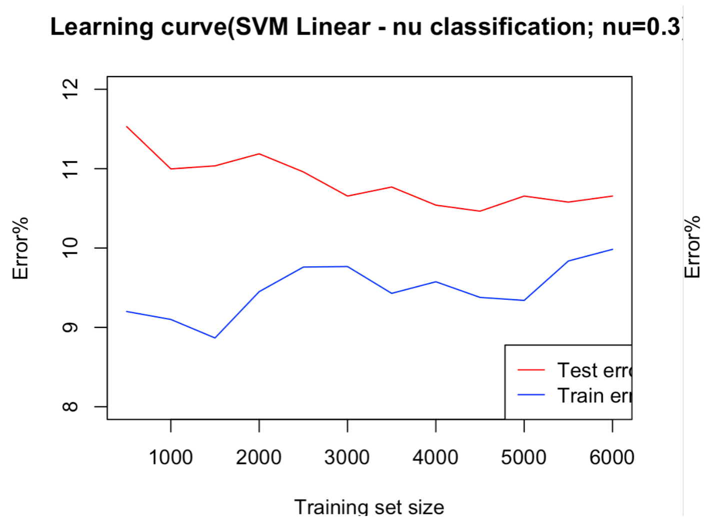
Fig 1.9 Learning curve as function on nu.

Above graph depicts the train and test error of SVM model at different values of nu. Apparently, nu at 0.3 can lead to best results with least train and test errors. We can use this to build our models



	nu- Classification @ nu=0.3		C-Classification	
	Training data	Testing data	Training data	Testing data
Accuracy :	0.9002	0.8923	0.9057	0.8957
95% CI :	(0.8924, 0.9076)	(0.8798, 0.9039)	(0.8982, 0.9129)	(0.8834, 0.9072)
No Information Rate :	0.6029	0.5864	0.6029	0.5864
P-Value [Acc > NIR] :	< 2.2e-16	< 2.2e-16	< 2.2e-16	< 2.2e-16
Kappa :	0.795	0.7804	0.8063	0.7873
Mcnemar's Test P-Value :	< 2.2e-16	1.979e-06	< 2.2e-16	5.873e-06
Sensitivity :	0.9240	0.9071	0.9285	0.9089
Specificity :	0.8845	0.8819	0.8907	0.8864
Pos Pred Value :	0.8405	0.8442	0.8484	0.8495
Neg Pred Value :	0.9465	0.9308	0.9498	0.9324
Prevalence :	0.3971	0.4136	0.3971	0.4136
Detection Rate :	0.3669	0.3752	0.3687	0.3760
Detection Prevalence :	0.4366	0.4444	0.4346	0.4425
Balanced Accuracy :	0.9043	0.8945	0.9096	0.8977
'Positive' Class :	High	High	High	High

Fig 1.10 Confusion matrix for nu and c classification.



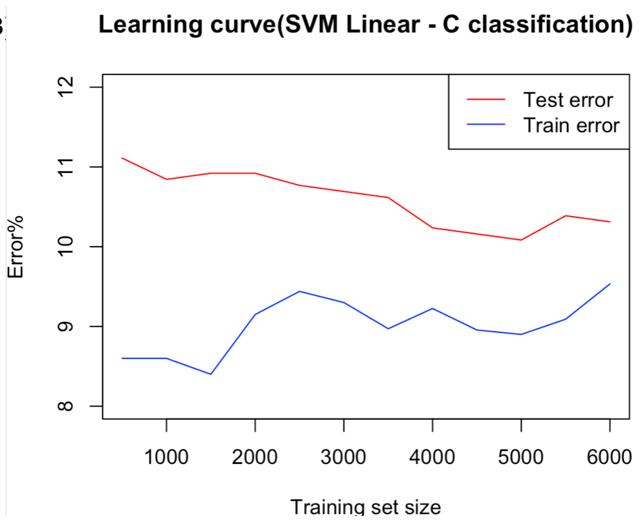


Fig 1.11 Learning cures for both models.

From the above figures 1.10 and 1.11 we can observe that even though c-classification's accuracies are slightly better than nu-classification, nu classification has less variance compared to C classification. This can be observed in the lesser gap between train and test error curves in fig.1.11. This explains the narrow confidence interval in the confusion matrix for nu- classification. Increasing the data set size can reduce the variance for c- classification model.

Apart from liner model, various kernels can be used to solve a classification problem with SVM such as polynomial and radial basis.

<u>Kernel methods– Polynomial & Radial Basis:</u>

We can use kernel trick to transform the features to higher dimensions so that data can be classified by drawing a hyperplane in a better way. Two of the most frequently used kernel methods are polynomial and radial basis kernel.
Polynomial kernel uses degrees as parameter to project the features into a greater number of dimensions to solve classification problem by drawing a hyperplane across. For example, if we have only 1 feature(x), we can 3rd degree polynomial to project the data into 3-dimensional space by converting to x^3, x^2, x.

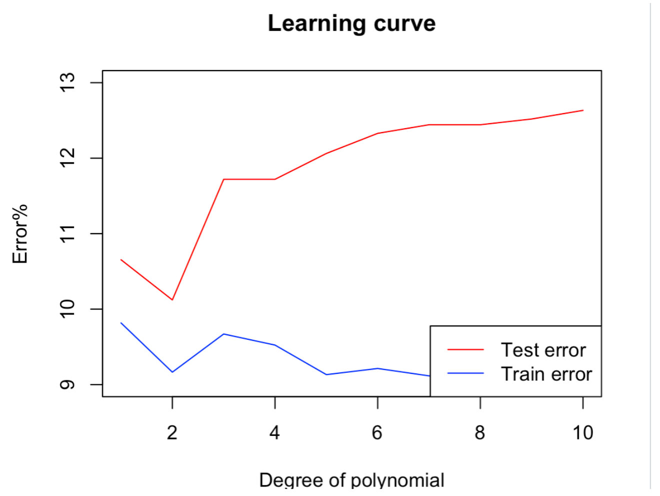


Fig 1.12 Learning cures for degrees of polynomial.

The above graph shows the error rates at different degrees of polynomial. Apparently 2 degrees is the optimum number for this data set as the error rates are low and gap between train and test errors is also low.

	Polynomial @ degree =2		Radial Basis	
	Training data	**Testing data**	**Training data**	**Testing data**
Accuracy :	0.9083	0.8988	0.9188	0.8992
95% CI :	(0.9009, 0.9155)	(0.8866, 0.9101)	(0.9117, 0.9255)	(0.887, 0.9104)
No Information Rate :	0.6029	0.5864	0.6029	0.5864
P-Value [Acc > NIR] :	< 2.2e-16	< 2.2e-16	< 2.2e-16	< 2.2e-16
Kappa :	0.8127	0.7948	0.8331	0.7948
Mcnemar's Test P-Value :	< 2.2e-16	4.642e-14	< 2.2e-16	7.724e-09
Sensitivity :	0.9491	0.9347	0.9462	0.9218
Specificity :	0.8815	0.8735	0.9007	0.8832
Pos Pred Value :	0.8407	0.8390	0.8626	0.8477
Neg Pred Value :	0.9633	0.9499	0.9621	0.9412
Prevalence :	0.3971	0.4136	0.3971	0.4136
Detection Rate :	0.3769	0.3866	0.3757	0.3813
Detection Prevalence :	0.4483	0.4608	0.4356	0.4498
Balanced Accuracy :	0.9153	0.9041	0.9235	0.9025
'Positive' Class :	High	High	High	High

Fig 1.13 Confusion matrix for 2nd degree polynomial and radial basis.

Apparently, accuracies of kernel methods are slightly better than the linear method as we instead of trying to classify the data with available features using a plane, we customized and projected the dimensions appropriately so that a hyperplane can classify the data more accurately.

Comparison of learning curves for Kernel methods:

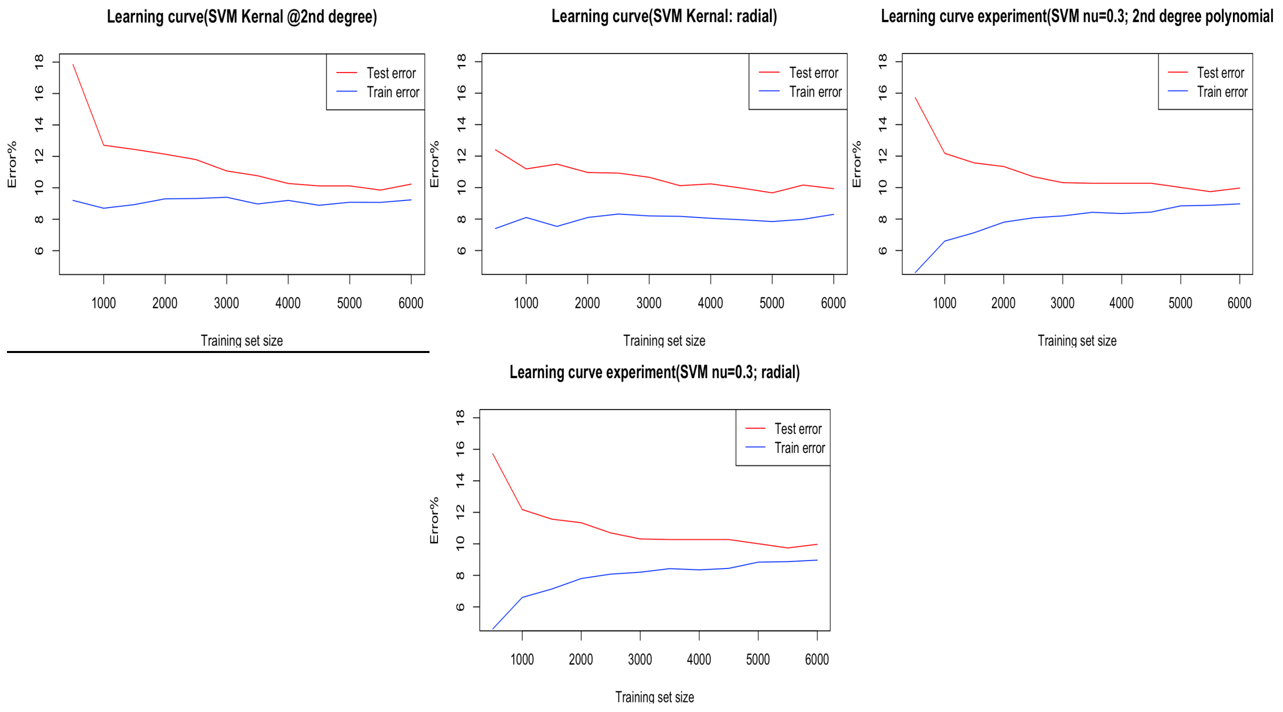


Fig 1.14 Learning curves for various kernel methods.

From the above figure we can see that except for radial model, learning curves are similar for all the other models with narrow gap between train and test error and similar overall error. There is high variance in radial model as the gap between train and test errors is high. Overall error can be further reduced if we can reduce the bias by introducing new features/independent variables.

Cross validation error – SVM:

We can use another metric to test the accuracy of the model called cross validation error. With this method training data set can be divided into k folds and training is done on k-1 folds and testing is done on excluded kth fold. This will help to understand how well our model generalizes on unknown new data.

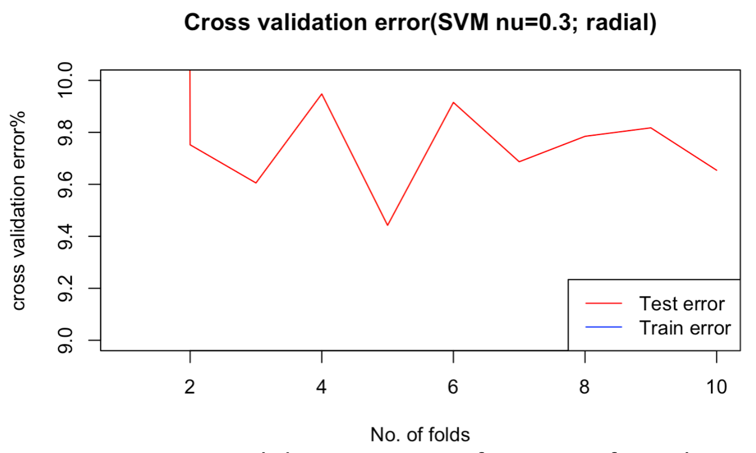


Fig 1.15 Cross validation error as function of number of folds.

Above graph shows that cross validation is almost around 9.8 at all folds of validations and pretty much consistent. This shows that the algorithm is stable and we can expect around 9.8% error on new or unseen data.