# Data Sets and Classification Problem description:

**1. Car evaluation Data Set (Multi-Variate response variable):**

This data helps to build a model that can evaluate cars according to various concept structures such as their buying price, maintenance costs, number of doors & persons, boot capacity, safety rating.

Every person faces this situation of evaluating and choosing a car before buying that suits their requirements more closely. Thus, I hope this classification model will prove to be very useful for everyone.

Response Variable: **car** (categories: unacc, acc, good, vgood)

Input independent variables (All Categorical):

**buying**: vhigh, high, med, low.  **maint**: vhigh, high, med, low.  **doors**: 2, 3, 4, 5more.   
**persons**: 2, 4, more.  **lug\_boot**: small, med, big.  **safety**: low, med, high.

**2. Credit Approval Data Set (Binary Classification problem):**

This data contains credit card application details of various applicants and deals with deciding whether to approve a credit card to a customer or not based on his several historical information. All attribute names and values have been changed to meaningless symbols to protect confidentiality of the data.

This dataset is interesting because there is a good mix of attributes -- continuous, nominal with small numbers of values, and nominal with larger numbers of values.

Response Variable: **A16** (categories: yes -> Approve; No -> Reject)

Input independent variables:

**A1**: b, a.  **A2**: continuous.  **A3**: continuous.  **A4**: u, y, l, t.  **A5**: g, p, gg.   
**A6**: c, d, cc, i, j, k, m, r, q, w, x, e, aa, ff.  **A7**: v, h, bb, j, n, z, dd, ff, o.  **A8**: continuous.   
**A9**: t, f.  **A10**: t, f.  **A11**: continuous.  **A12**: t, f.  A13: g, p, s.  **A14**: continuous.  **A15**: continuous.

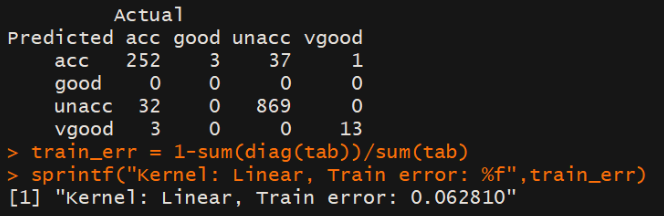
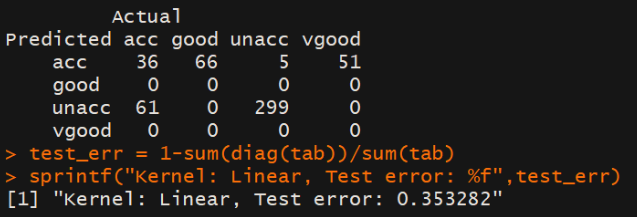
Error rates (train & test sets), Confusion Matrices and learning curves:

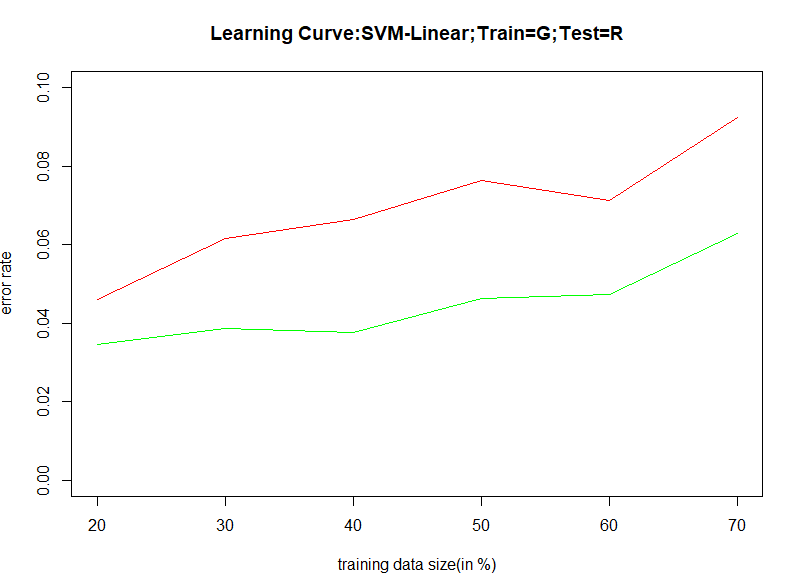
**Data-Set 1: Car evaluation problem:**

**Support Vector Machines model:**

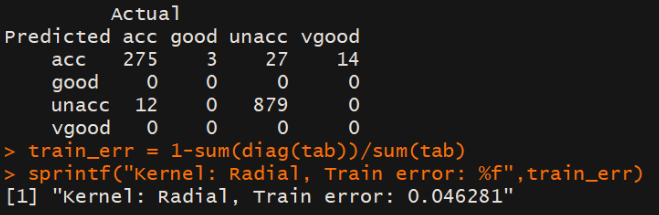
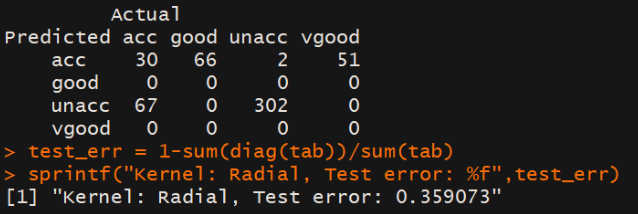
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Kernel** | **Train Error rate** | **Test Error rate** | **Accuracy (Test Set Prediction)** | **No. of Support vectors used** |
| Linear | 6.28 % | 35.33 % | 64.67 % | 304 |
| Radial | 4.63 % | 35.90 % | 64.10 % | 519 |
| Polynomial | 25.12 % | 41.31 % | 58.69 % | 635 |
| Sigmoid | 8.51 % | 37.64 % | 62.36 % | 537 |

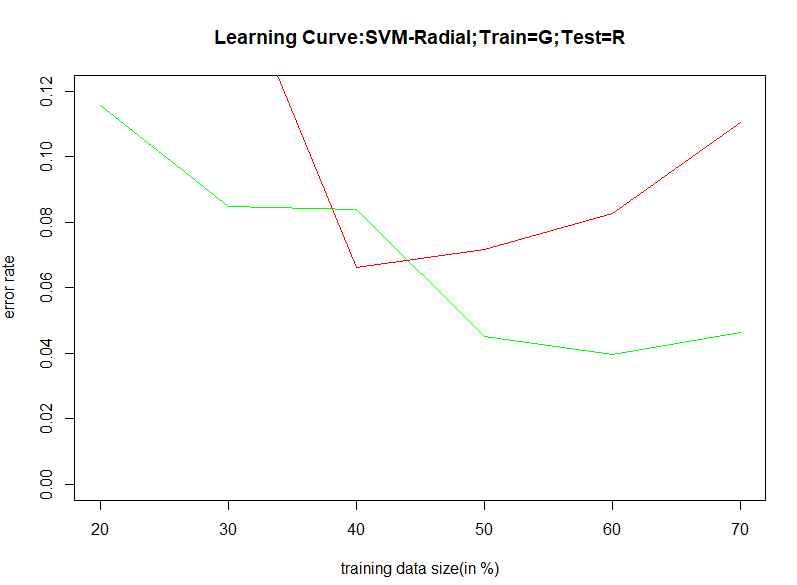
**Linear Kernel:**

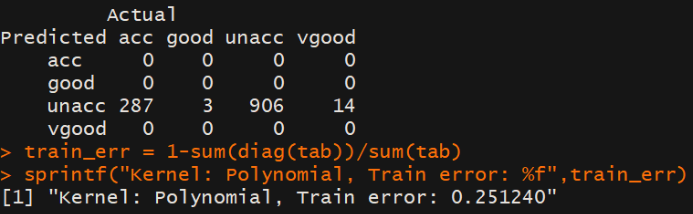
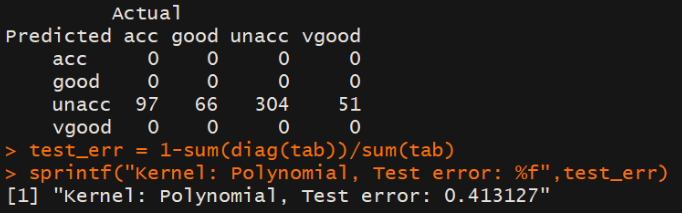


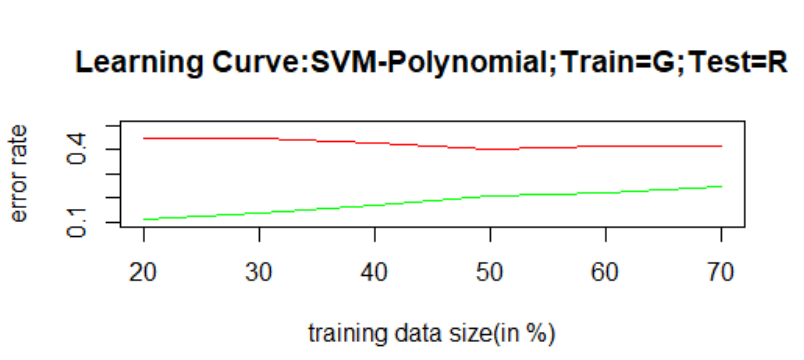
**Radial Kernel:**

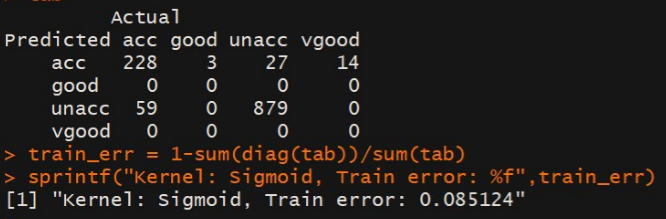
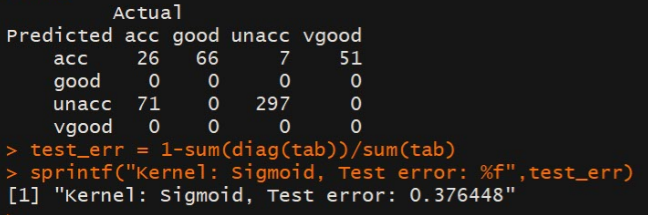


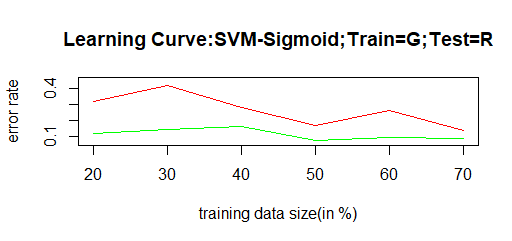
**Polynomial Kernel:**



**Sigmoid Kernel:**



**Comparison of various Kernels:**

As far as the error rates and confusion matrices are concerned, except for “Polynomial”, the other three “Linear”, “Radial” and “Sigmoid” have considerably lower values.

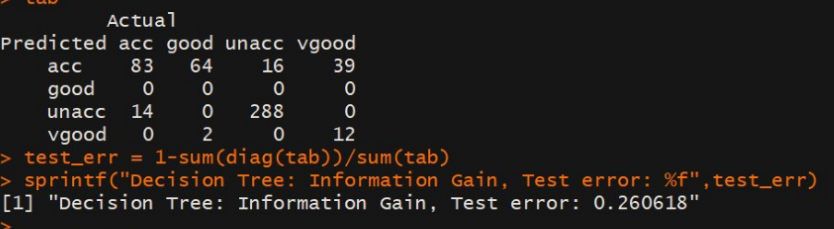
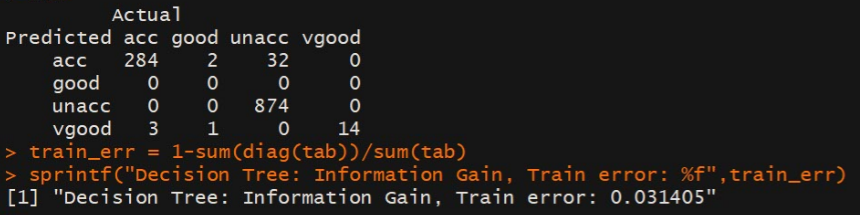
But when we look at their **learning curves**, “Linear” performs poor as its train & test error rates increases rapidly with increase in training size.

I would prefer “**Radial**” kernel as it has an overall better performance with **519** support vectors.

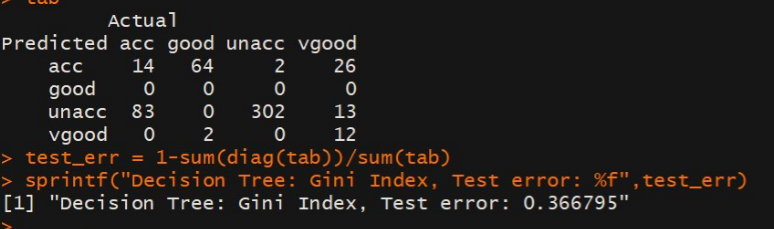
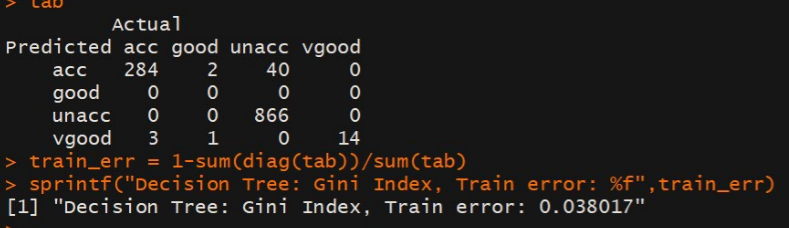
**Decision Trees:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Splitting Criterion** | **Train Error Rate** | **Test Error Rate** | **Accuracy (Test Set Prediction)** |
| Information Gain | 3.14 % | 26.06 % | 73.94 % |
| Gini Index | 3.80 % | 36.68 % | 63.32 % |

**Information Gain:**

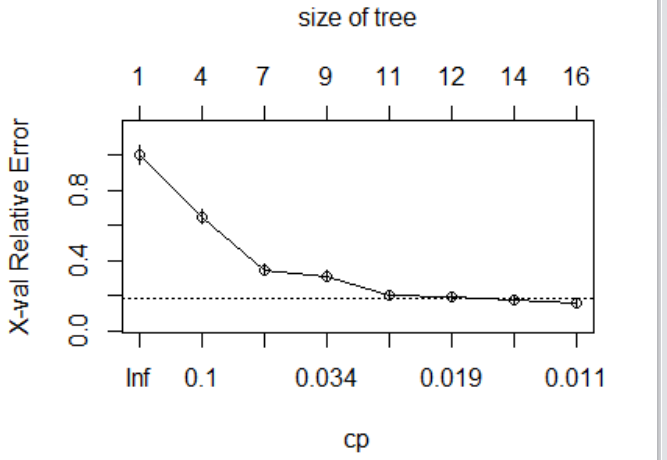


**Gini Index:**

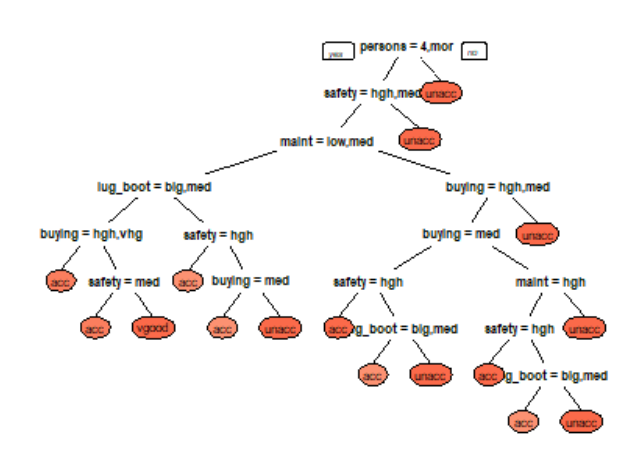


By looking at the above table, we can clearly see that “Information Gain” performs better than “Gini” with **10% more accuracy** in classifying the response variable correctly. So, we will **choose “Information Gain”** as our splitting criterion**.**

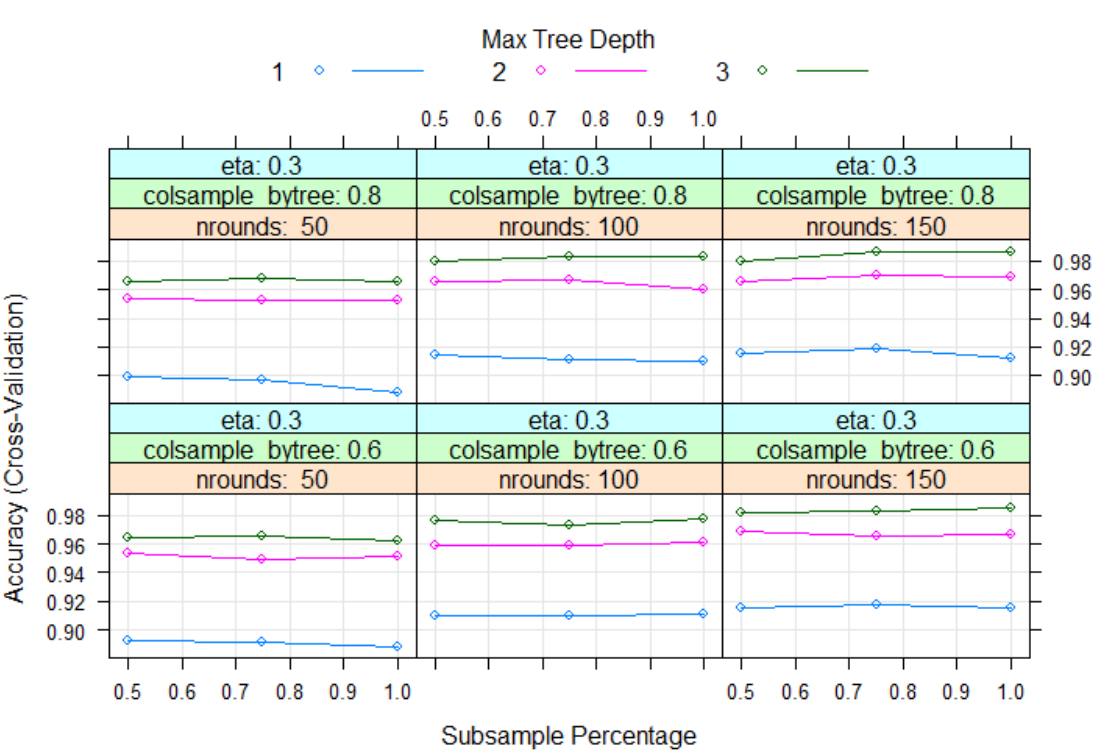
**POST-PRUNING: Complexity Parameter table (cp table) for Information Gain-Decision tree:**



As we can see, the tree grows fully with an “XERROR” value of 0.15789 for the final split which happens to be the lowest of all the other “XERROR” values”. Hence, any further pruning will result in information loss. For this reason**, this decision tree needs NO PRUNING**.

**Final Decision Tree**

**Extreme Boosted Decision Tree:**

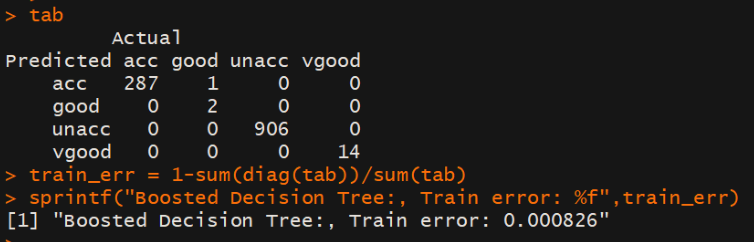
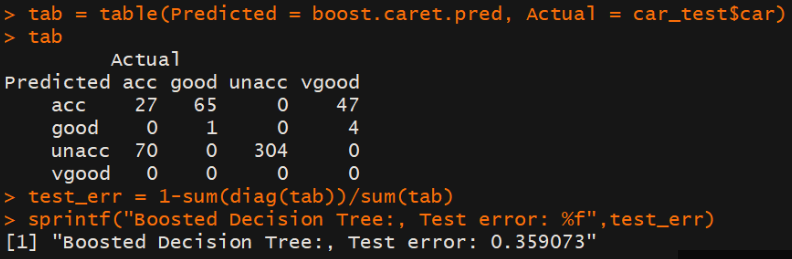


We are using the extreme boosting package to boost our decision tree.

Tuning parameter 'gamma' was held constant at a value of 0. Tuning parameter 'min\_child\_weight' was held constant at a value of 1. **Accuracy was used to select the optimal model using the largest value.**

**Pre-Pruning:** The final values used for the model were nrounds = 150, max\_depth = 3, eta = 0.4, gamma = 0, colsample\_bytree = 0.8, min\_child\_weight = 1 and subsample = 1. Also, we used a **10-fold cross validation method** to build this model.

|  |  |  |  |
| --- | --- | --- | --- |
| **Boosting Method** | **Train Error Rate** | **Test Error Rate** | **Accuracy (Test Set Prediction)** |
| Extreme Boost | 0.08 % | 35.91 % | 64.09 % |

Thus, the boosted decision tree produces an accuracy of 64 % with 95% confidence.

## Comparison between SVM-“Radial” Kernel vs Information Gain Decision Tree vs Boosted Decision Tree

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithm** | **Train Error Rate** | **Test Error Rate** | **Accuracy (Test Set Prediction)** |
| SVM – Radial Kernel | 4.63 % | 35.90 % | 64.10 % |
| Information Gain Decision Tree | 3.14 % | 26.06 % | 73.94 % |
| Boosted Decision Tree | 0.08 % | 35.91 % | 64.09 % |

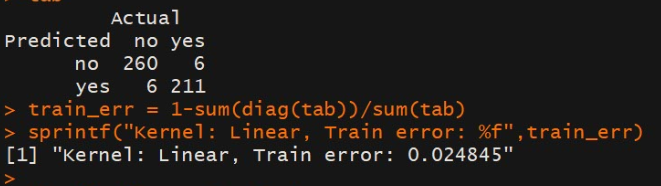
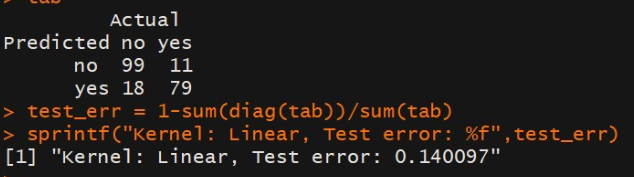
All the three algorithms perform equally good in classifying the response variable correctly**. But I would choose Information Gain Decision Tree** **algorithm over the other two due to its slightly higher Accuracy rate**.

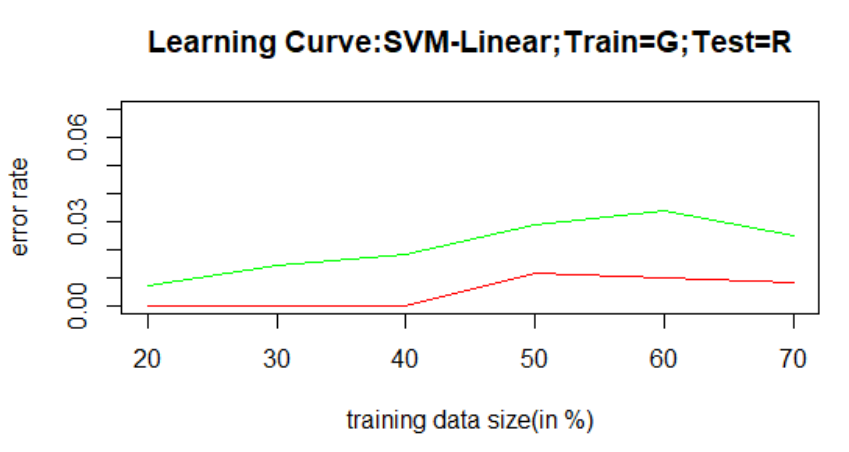
**Data Set 2: Credit Approval Data Set:**

**Support Vector Machines model:**

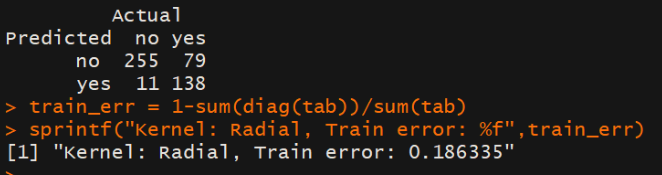
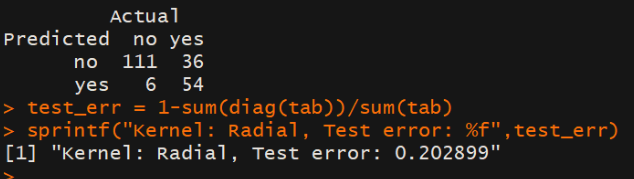
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Kernel** | **Train Error rate** | **Test Error rate** | **Accuracy (Test Set Prediction)** | **No. of Support vectors used** |
| Linear | 2.48 % | 14.01 % | 85.99 % | 200 |
| Radial | 18.63 % | 20.29 % | 79.71 % | 386 |
| Polynomial | 44.93 % | 43.48 % | 56.52 % | 434 |
| Sigmoid | 27.12 % | 36.23 % | 63.77 % | 419 |

**Linear Kernel:**

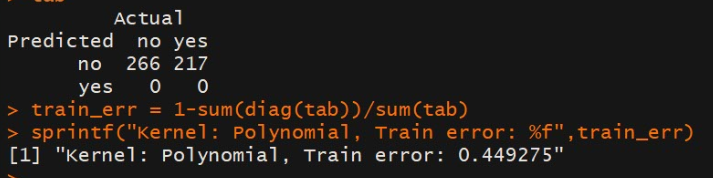
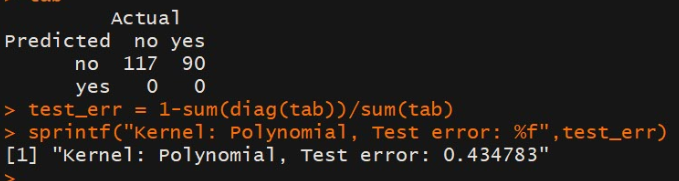


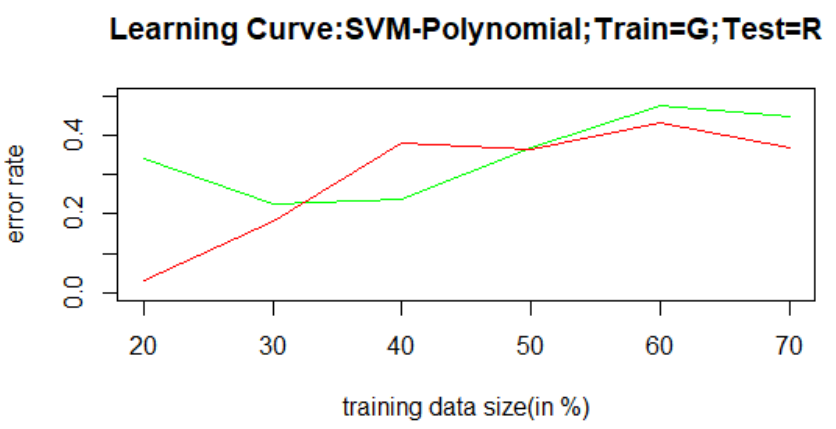
**Radial Kernel:**

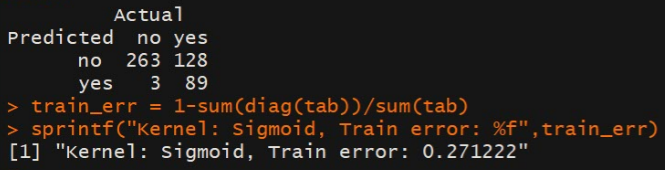
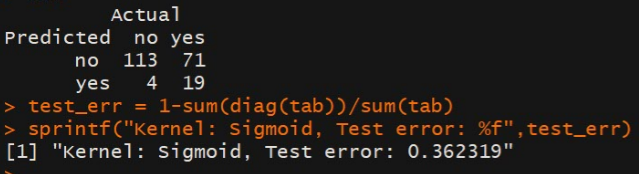


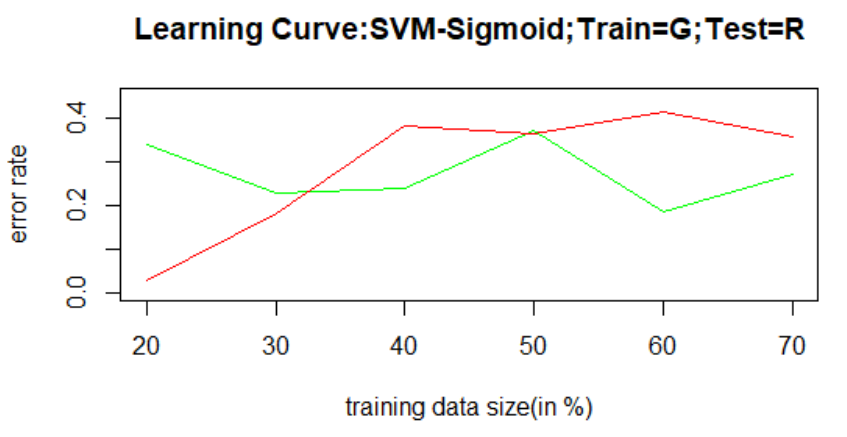
**Polynomial Kernel:**



**Sigmoid Kernel:**



**Comparison of various Kernels:**

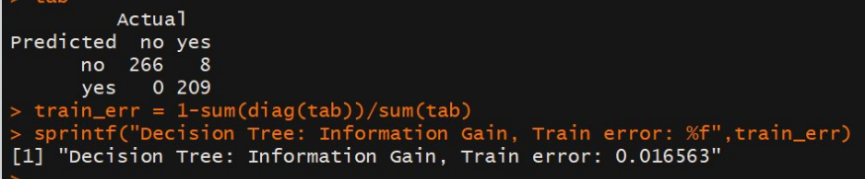
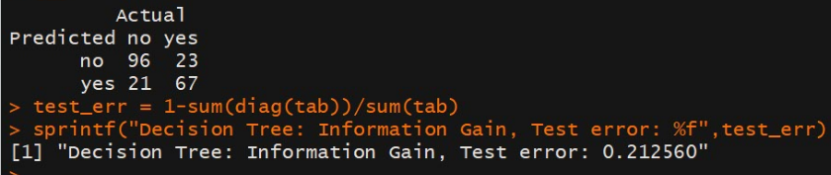
By looking at the error table and confusion matrices of all these kernels, we could certainly conclude that “**Linear**” kernel performs the best in classifying the response variable correctly with a top **accuracy of 86 %**

Though, the learning curves for both train and test sets converge with increasing training set size for all the four tested kernels, the learning curves of “Linear” kernel converges at the lower error rates. **Thus, these learning curves agree with our above findings.** We choose “**Linear**” Kernel**.**

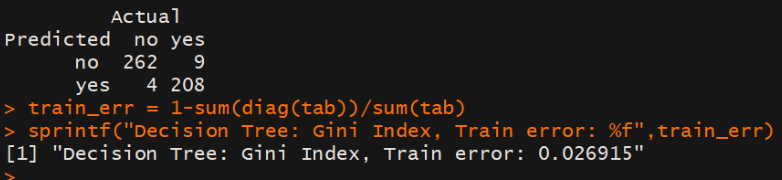
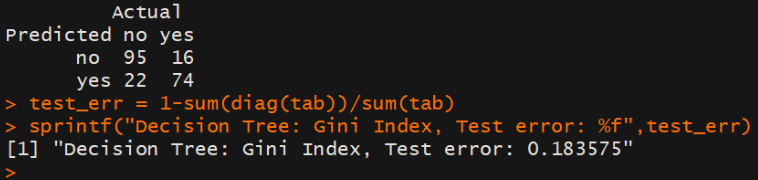
**Decision Trees:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Splitting Criterion** | **Train Error Rate** | **Test Error Rate** | **Accuracy (Test Set Prediction)** |
| Information Gain | 1.66 % | 21.26 % | 78.74 % |
| Gini Index | 2.69 % | 18.36 % | 81.64 % |

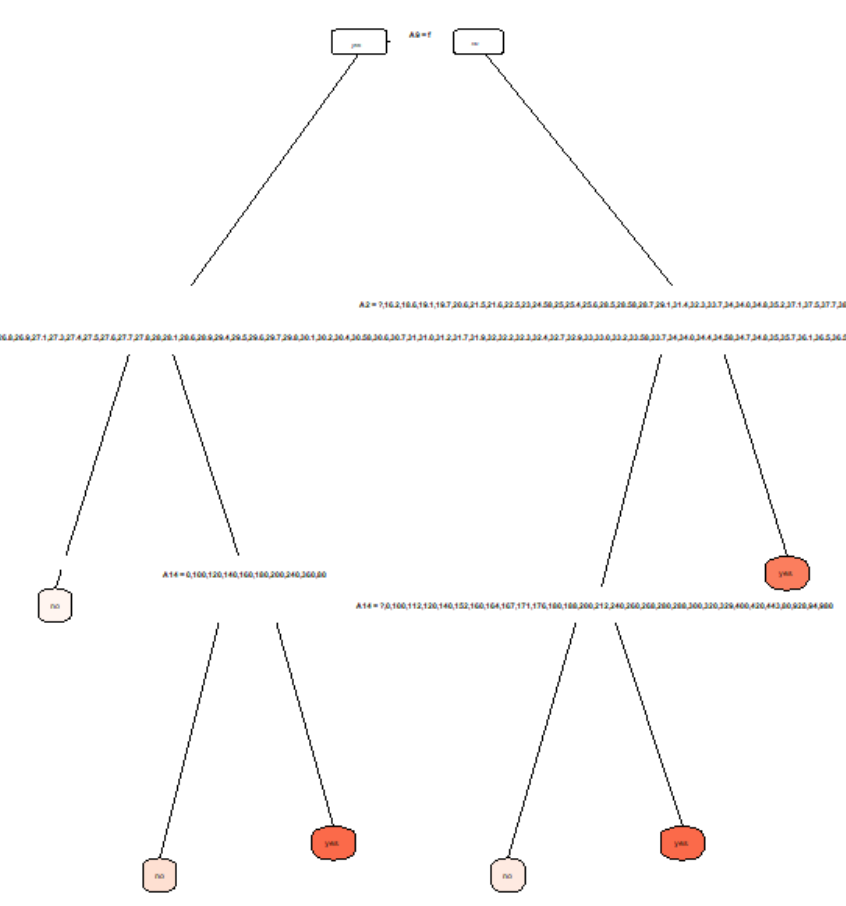
**Information Gain:**

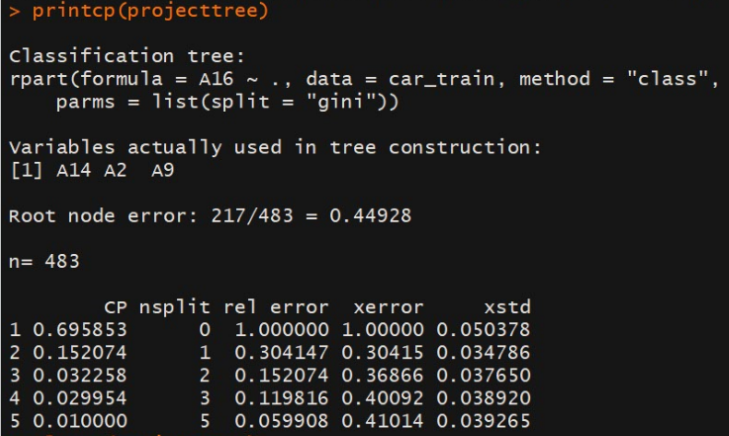
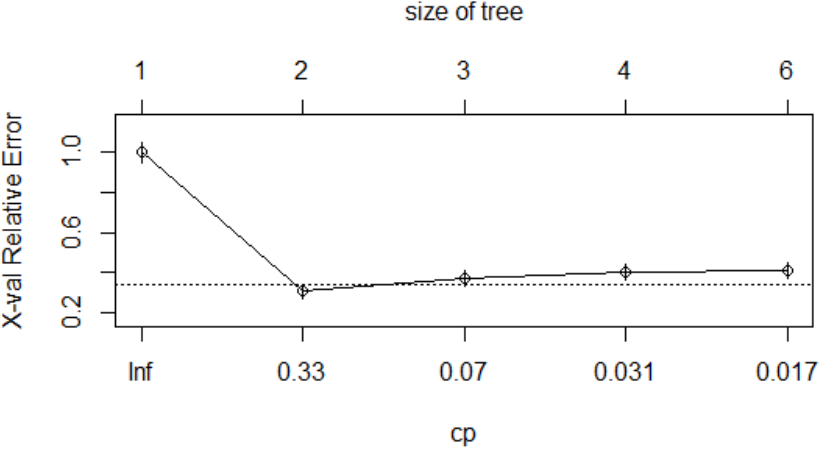
**Gini Index:**

By looking at the above table, we can see that the “Gini” performs slightly better than “Information Gain” by **3% more accuracy** in classifying the response variable correctly. So, we will **choose “Gini Index”** as our splitting criterion**.**

 **Decision Tree with Gini Index before Pruning**

**POST-PRUNING: Complexity Parameter table (cp table) for Gini Index-Decision tree:**

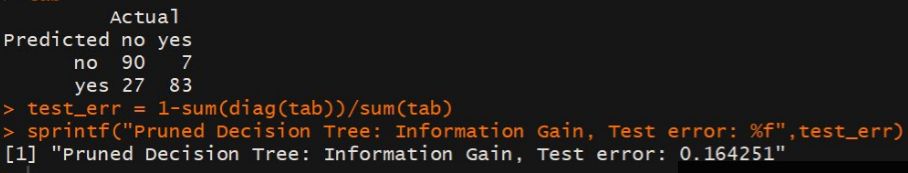
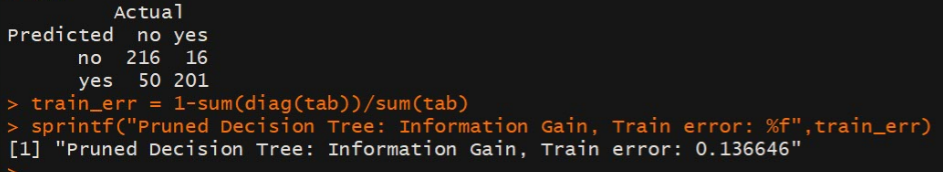
 

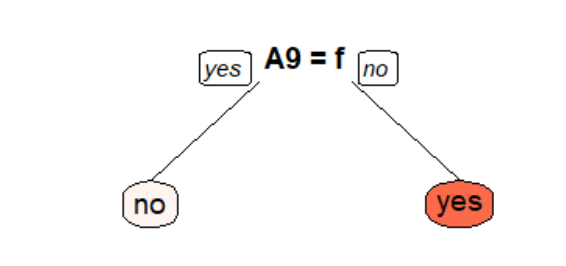
As we can see, the tree grows fully with an “XERROR” value of 0.41014 for the final split.

However, we notice that the “XERROR” value increases from a minimum value of 0.30415 corresponding to the CP = 0.152074 at split 1. It is safe to Prune this tree until the minimum “XERROR” value without incurring any information loss.

**Pruned Tree Accuracy**:

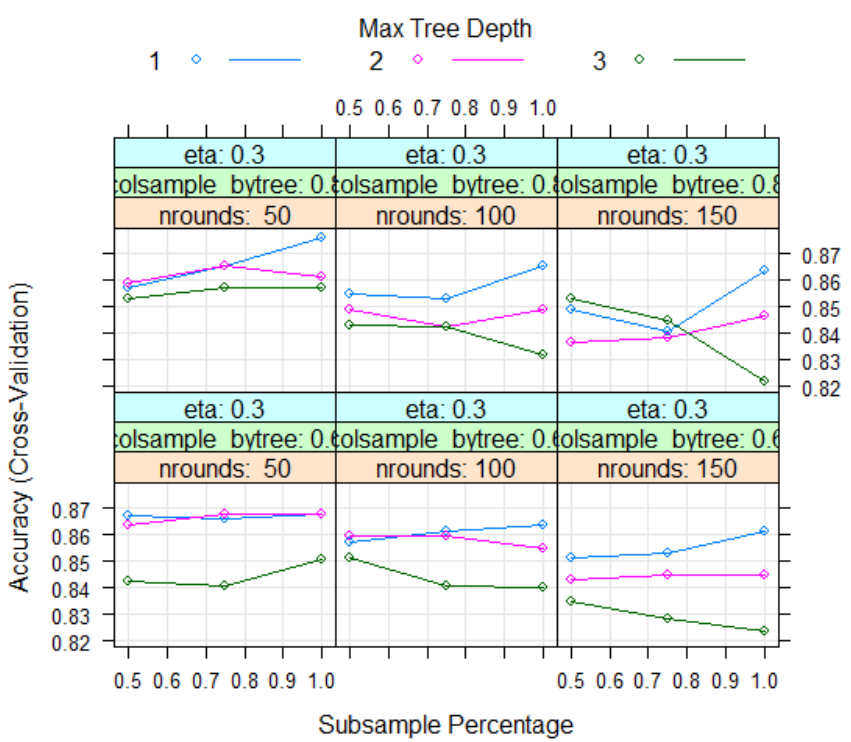
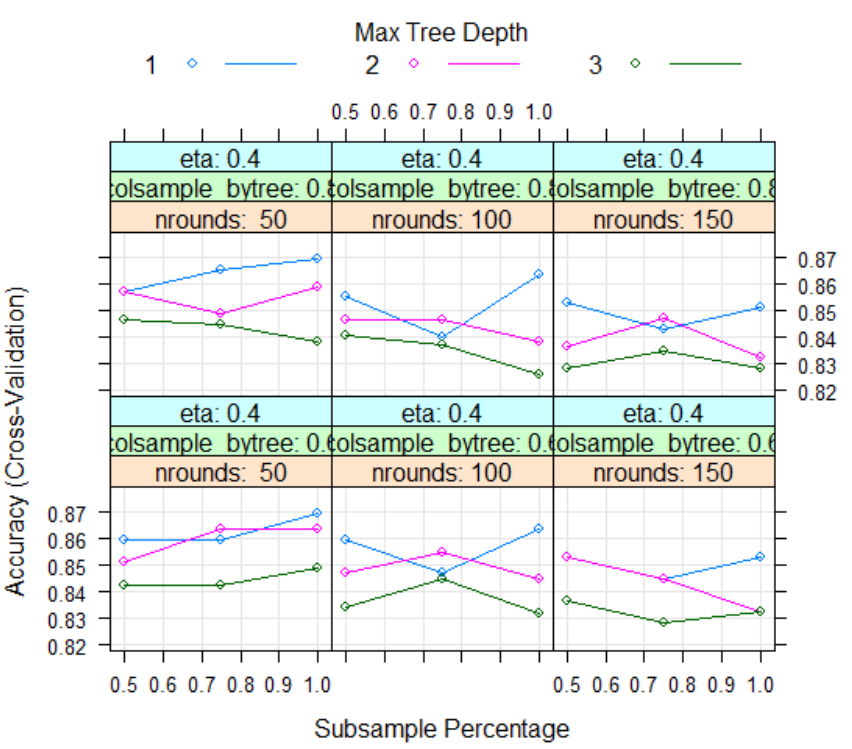
|  |  |  |  |
| --- | --- | --- | --- |
| **Splitting Criterion** | **Train Error Rate** | **Test Error Rate** | **Accuracy (Test Set Prediction)** |
| Gini Index | 13.66 % | 16.43 % | 83.57 % |



**Pruned Decision Tree**

Hence, Pruning helps avoid over-fitting the train-data and generalize well for future data.

**Extreme Boosted Decision Tree:**

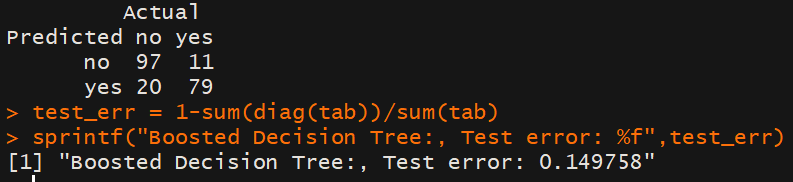
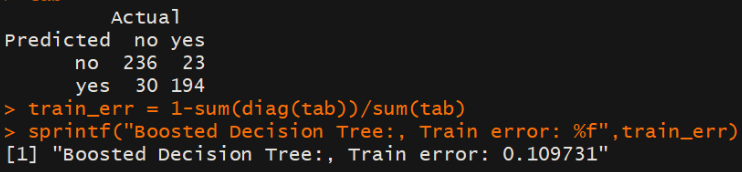
We are using the extreme boosting package to boost our decision tree.

Tuning parameter 'gamma' was held constant at a value of 0. Tuning parameter 'min\_child\_weight' was held constant at a value of 1. **Accuracy was used to select the optimal model using the largest value.**

**PRE-PRUNING:** The final values used for the model were nrounds = 50, max\_depth = 1, eta = 0.3, gamma = 0, colsample\_bytree = 0.8, min\_child\_weight = 1 and subsample = 1.

Also, we used a **10-fold cross validation method** to build this model.

|  |  |  |  |
| --- | --- | --- | --- |
| **Boosting Method** | **Train Error Rate** | **Test Error Rate** | **Accuracy (Test Set Prediction)** |
| Extreme Boost | 10.97 % | 14.97 % | 85.03 % |



Thus, the boosted decision tree produces an accuracy of **85.03%** with 95% confidence.

## Comparison between SVM-“Linear” Kernel vs Gini Index Decision Tree vs Boosted Decision Tree

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithm** | **Train Error Rate** | **Test Error Rate** | **Accuracy (Test Set Prediction)** |
| SVM – Linear Kernel | 2.48 % | 14.01 % | 85.99 % |
| Gini Index Decision Tree | 13.66 % | 16.43 % | 83.57 % |
| Boosted Decision Tree | 10.97 % | 14.97 % | 85.03 % |

All the three algorithms perform equally good in classifying the response variable correctly**. But I would choose SVM – Linear Kernel algorithm over the other two due to its slightly higher Accuracy rate**.