# BUAN 6341.501: APPLIED MACHINE LEARNING

## 

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

## Data-Set 1: Car evaluation problem:

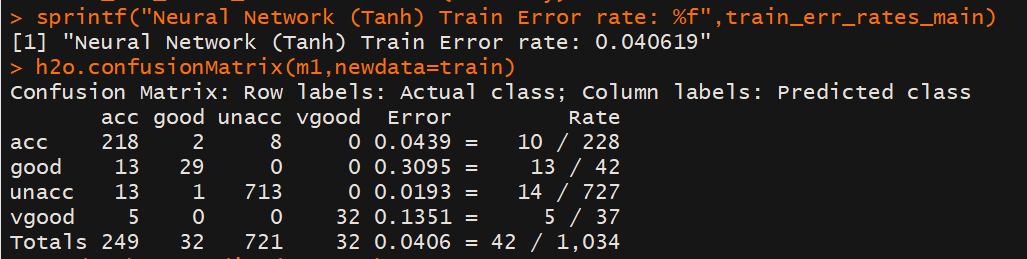
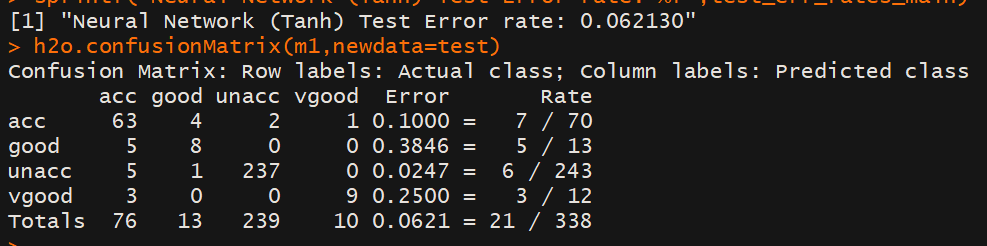
**Neural Networks model:**

**Cross Validation:** 5-fold cross-validations has been used as the control parameter to run the models **for both datasets**. Cross validation has been used to pick optimal weights to reduce over-fitting.

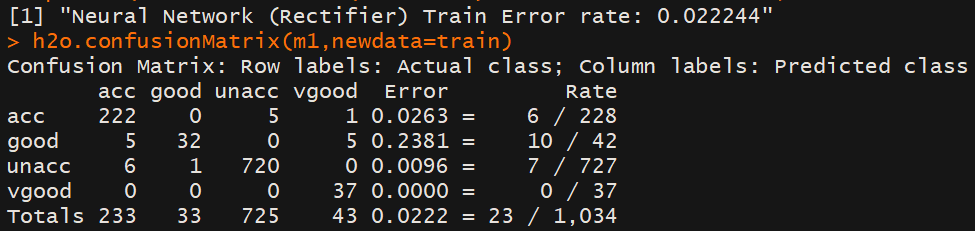
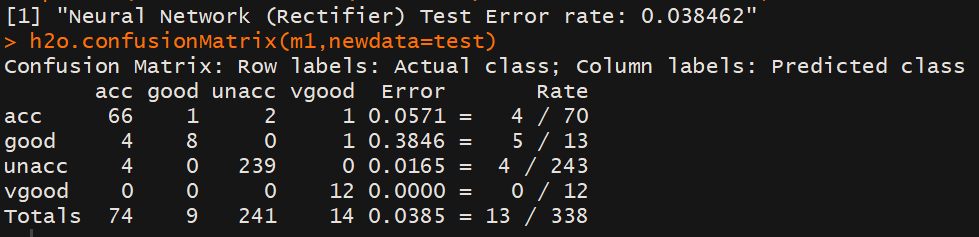
**Package and Regularization:** We have used H2o Deep Learning to build neural networks and set the L2 parameter to 6e-4 in order to perform Regularization.

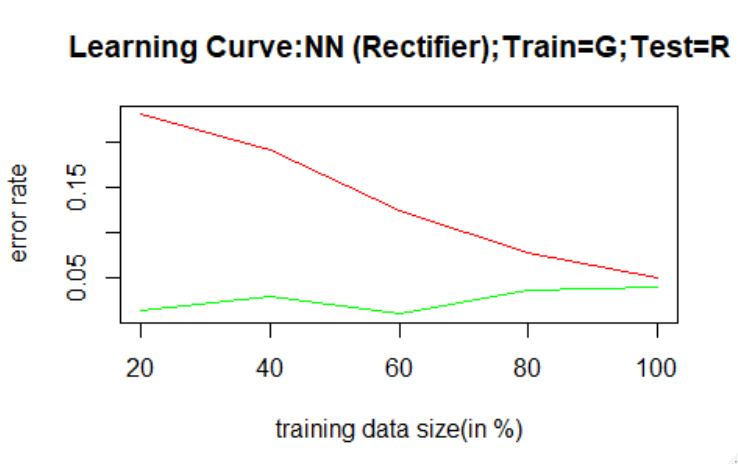
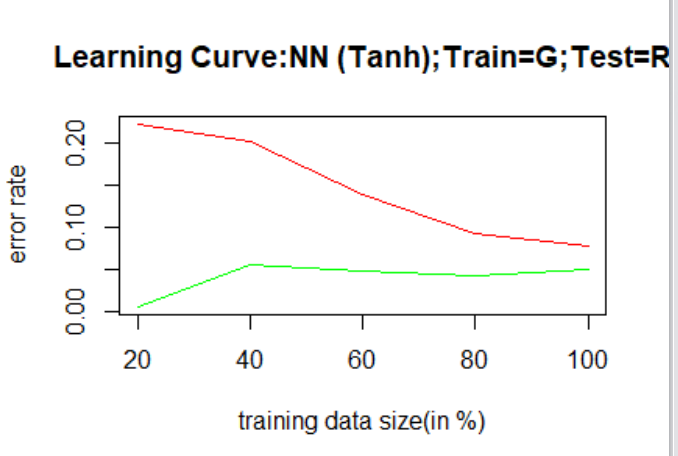
|  |  |  |  |
| --- | --- | --- | --- |
| **Activation** | **Train Error rate** | **Test Error rate** | **Accuracy (Test Set Prediction)** |
| Tanh | 4.06 % | 6.21 % | 93.79 % |
| Rectifier | 2.22 % | 3.84 % | 96.16 % |

**Tanh Activation: TRAIN ERROR TEST ERROR**

**Rectifier Activation: TRAIN ERROR TEST ERROR**



**Tanh vs Rectifier:**

As far as the Error Rates and Confusion matrices are concerned, the **Rectifier** activation outperforms the Tanh by a slight margin with an **additional 3% accuracy** and less error rate values.

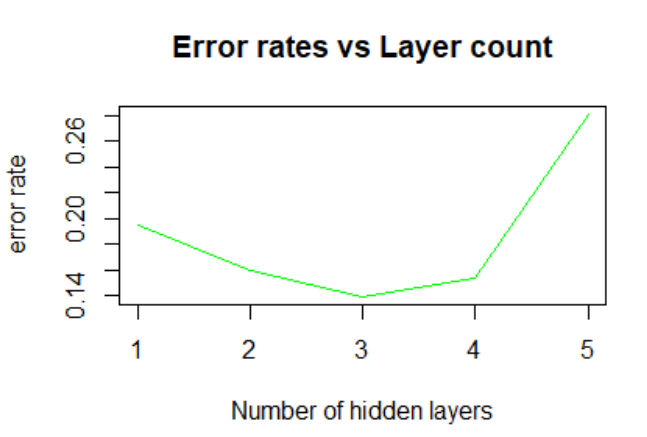
**HIGH BIAS CHECK:** Both the learning curves doesn’t exhibit High Bias as the Eout decreases steadily with increasing train data size. Also, the Ein values doesn’t increase steeply with increasing train data size.

**HIGH VARIANCE CHECK:** Both the learning curves doesn’t exhibit High variance as the Ein steadily increases with train data size and Eout steadily decreases with train data size.

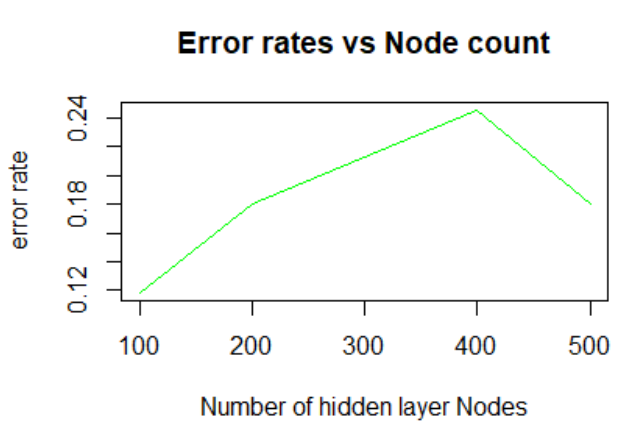
Thus, **we choose Rectifier Activation** since both the Ein and Eout values are less compared to Tanh.

## Experimentation with various parameters:

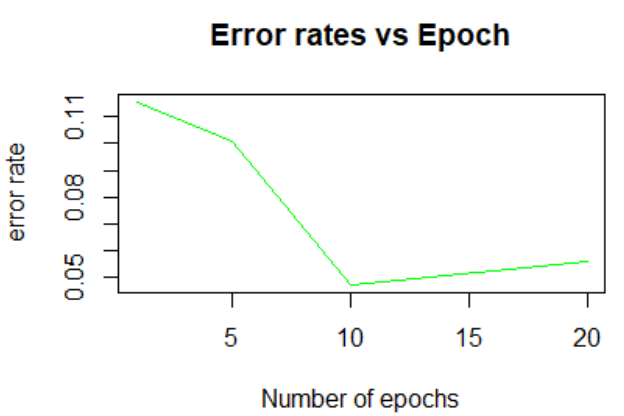
**a)** **Number of Layers:** The test error rate reaches minimum at 3 Layers before increasing. Hence, **we chose 3 layers.**



**b) Number of Nodes in each Layer: We chose 100 Nodes** in each layer since it produces the minimum test error rate.



**c) Number of Epochs: We chose Epoch = 10** as it proves to be an optimal option. An **epoch** is one complete presentation of the data set to be learned to a learning machine. Learning machines like feedforward neural nets that use iterative algorithms often need many epochs during their learning phase.



**KNN Model:**

**Caret Package -** For implementing KNN algorithm, we have used the CARET package on both the datasets. The KNN classifier works only the numeric variables, hence all the categorical predictor variables have been binarized using One Hot Encoding. Since, the algorithm uses distance as the metric to perform classification, we have normalized all the numeric variables to [0,1] range and the categorical predictors are already in [0,1] range.

**Choice of distance metric -**The distance metric that we have used here is the Euclidean distance. Euclidean distance works with real values inputs. And in this data, all the variables have been transformed to be able to use Euclidean distance as the distance metric for the KNN model.

**Cross Validation -** 10-fold cross-validations has been used as the control parameter to run the models. Cross validation has been used to picking an optimal value for the K parameter to reduce over-fitting.

Training Error Rate: **3.5%**

Test Error Rate: **3.08%,** Accuracy-(Test Data)**: 96.92%**

**Confusion Matrix (Test Data):**

|  |  |  |
| --- | --- | --- |
|  | Predicted (0) | Predicted (1) |
| Actual (0) | 148 | 6 |
| Actual (1) | 10 | 354 |

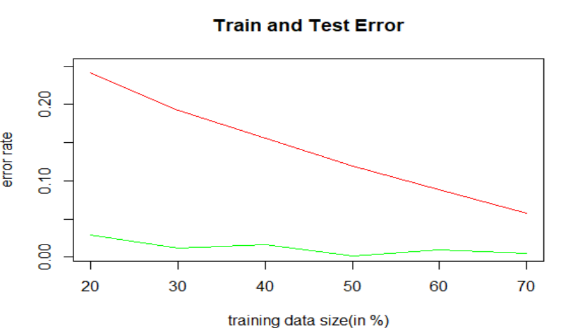
Learning Curves Experimentation: Training Data Vs Number of Neighbors K



**The algorithm selects the K with highest accuracy on cross-validated resamples of the training data as the optimal K.** **Usually optimal value of K is an odd number if the number of classes is binary.**

Thus, the Caret package that we used chose the ideal **K = 9.**

Error Rate: (Training Vs Test)



**Interpretation: HIGH BIAS CHECK:** Both the learning curves doesn’t exhibit High Bias as the Eout decreases steadily with increasing train data size. Also, the Ein values doesn’t increase steeply with increasing train data size.

**HIGH VARIANCE CHECK:** Both the learning curves doesn’t exhibit High variance as the Ein steadily increases with train data size and Eout steadily decreases with train data size.

## Comparison between Neural Networks(Rectifier) vs KNN(K=9) vs SVM-“Radial” Kernel vs Information Gain Decision Tree vs Boosted Decision Tree

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithm** | **Train Error Rate** | **Test Error Rate** | **Accuracy (Test Set Prediction)** |
| Neural Networks - Rectifier | 2.22 % | 3.84 % | 96.16 % |
| KNN | 3.5% | 3.08 % | 96.92 % |
| 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 % |

Comparing the Neural Net and KNN – we see that **KNN has a slightly higher accuracy**.

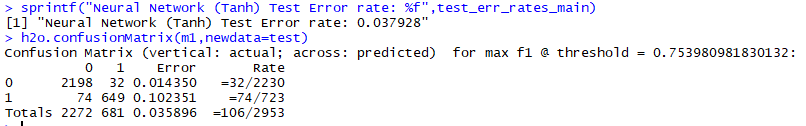
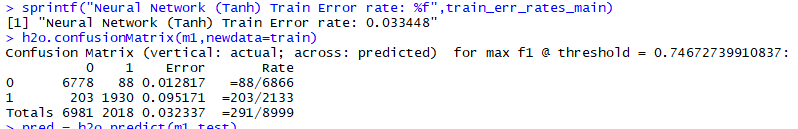
Comparing all the five models we see that **KNN is better** than all the models followed by the Neural Networks.

## Data-Set 2: HR Analytics:

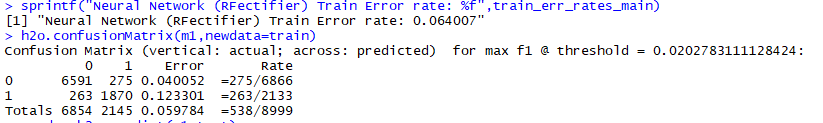
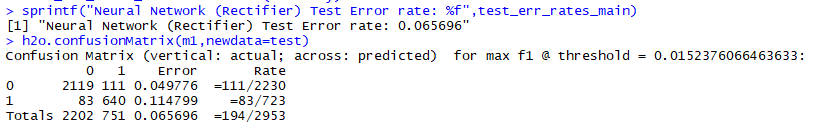
**Neural Networks model:**

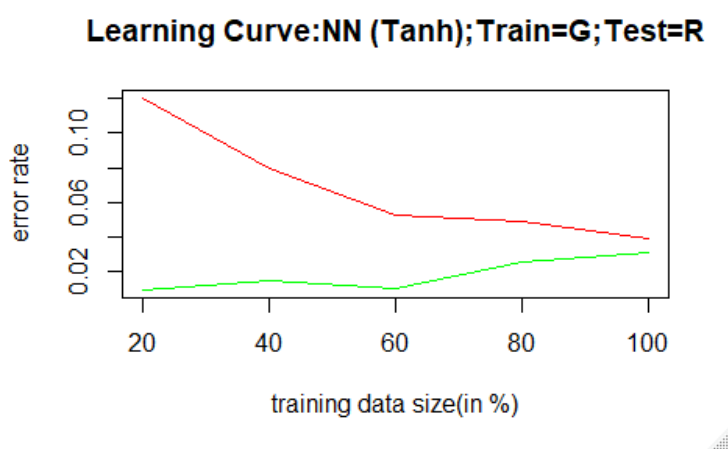
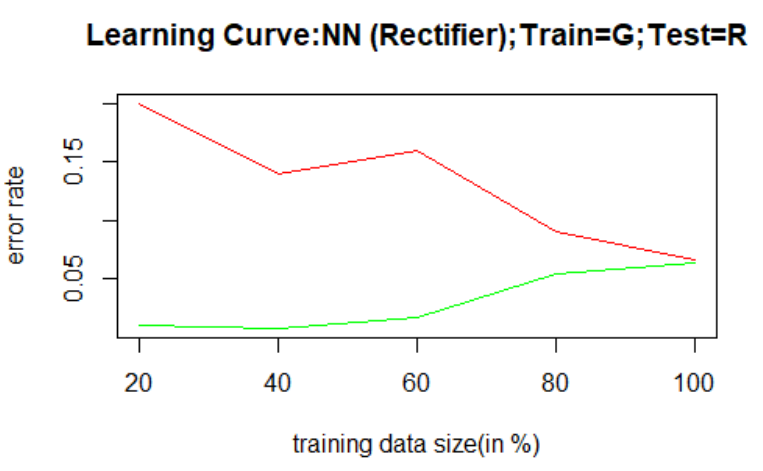
|  |  |  |  |
| --- | --- | --- | --- |
| **Activation** | **Train Error rate** | **Test Error rate** | **Accuracy (Test Set Prediction)** |
| Tanh | 3.34 % | 3.79 % | 96.21 % |
| Rectifier | 6.40 % | 6.58% | 93.42 % |

**Tanh Activation: TRAIN ERROR TEST ERROR**



**Rectifier Activation: TRAIN ERROR TEST ERROR**

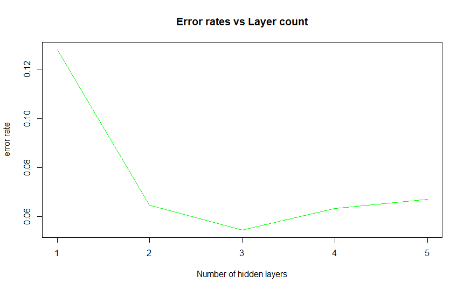
**Tanh vs Rectifier:**

As far as the Error Rates and Confusion matrices are concerned, the **Tanh activation outperforms** the Rectifier by a slight margin with an **additional 3% accuracy** and less error rate values.

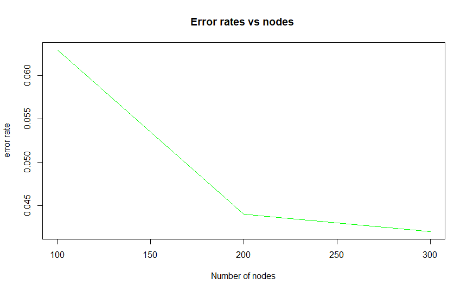
Thus**, we chose Tanh Activation** since both the Ein and Eout values are less compared to Rectifier function. The learning curves also reinforce the same.

## Experimentation with various parameters:

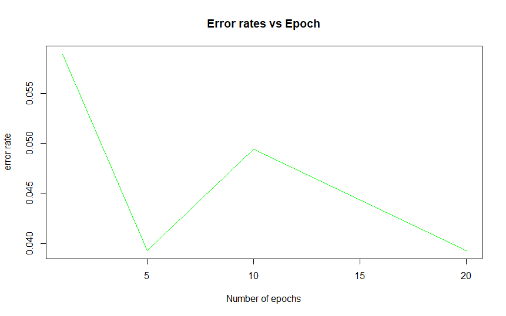
**a)** **Number of Layers:** The test error rate reaches minimum at 3 Layers before increasing. Hence, **we chose 3 layers.**



**b) Number of Nodes in each Layer: We chose 300 Nodes** in each layer since it produces the minimum test error rate.



**c)** **Number of Epochs: We chose Epoch = 5** as it proves to be an optimal option.



**KNN Model:**

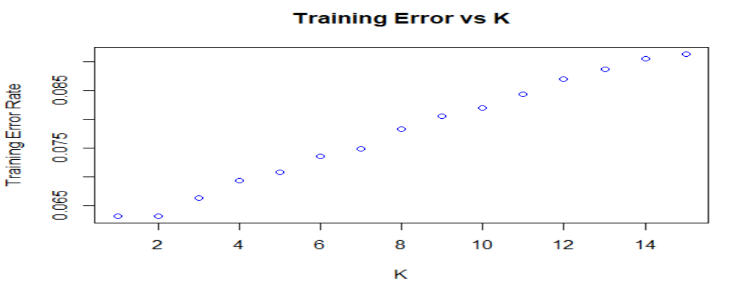
Training Error Rate: **6.3%**

Test Error Rate**: 5.91%,** Accuracy: **94.09%**

**Confusion Matrix (Test Data):**

|  |  |  |
| --- | --- | --- |
|  | Predicted (remained) | Predicted (left) |
| Actual (remained) | 3280 | 118 |
| Actual (left) | 148 | 953 |

Learning Curves Experimentation: Training Data Vs Number of Neighbors K

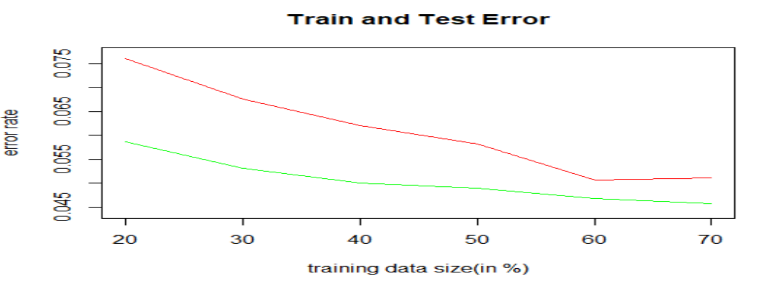


**The algorithm selects the K with highest accuracy on cross-validated resamples of the training data as the optimal K.** **Usually optimal value of K is an odd number if the number of classes is binary.**

Thus, the Caret package that we used chose the ideal **K = 5.**

*NOTE: The reference is available in the README file provided.*

Error Rate: Training Vs Test Data



Interpretation:

As the number of samples increases we see that the error rate decreases**.**

## Comparison between Neural Networks(Tanh) vs KNN(K=5) vs SVM-“Radial” Kernel vs Gini Index Decision Tree vs Boosted Decision Tree

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithm** | **Train Error Rate** | **Test Error Rate** | **Accuracy (Test Set Prediction)** |
| Neural Networks - Tanh | 3.34 % | 3.79 % | 96.21 % |
| KNN | 6.30 % | 5.91 % | 94.09 % |
| SVM – Radial Kernel | 3.78 % | 5.00 % | 95.00 % |
| Gini Index Decision Tree | 4.33 % | 5.09 % | 94.91 % |
| Boosted Decision Tree | 3.01 % | 3.33 % | 96.67 % |

Comparing the Neural Net and KNN – we see that **Neural networks has a higher accuracy**.

Comparing all the five models we see that **Boosting is better** than all the models followed by the Neural Networks.