

Assignment 02

Analysis of data set 1

Data Description: The analytic methods can improve Human Resources (HR) management for companies with substantial number of employees. I tried to analyze how can companies benefit from machine learning methods applied to HR. I would like to present how to predict employee attrition with machine learning. For analysis I will use a data set created by IBM data scientists. In this data each sample (row) describes the employee with parameters like: age, department, distance from home, marital status, income, years at company. However, the attrition is unknown and we want to predict (compute) it with our machine learning model.

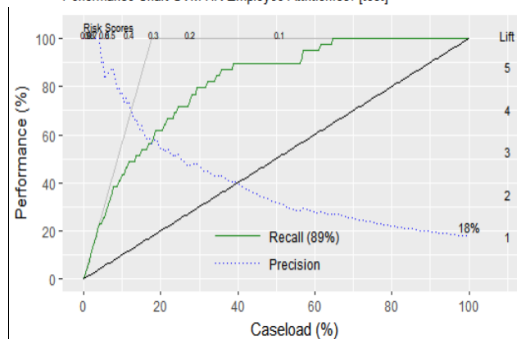
Algorithm 1: SVM

Comparison of the three kernels used:

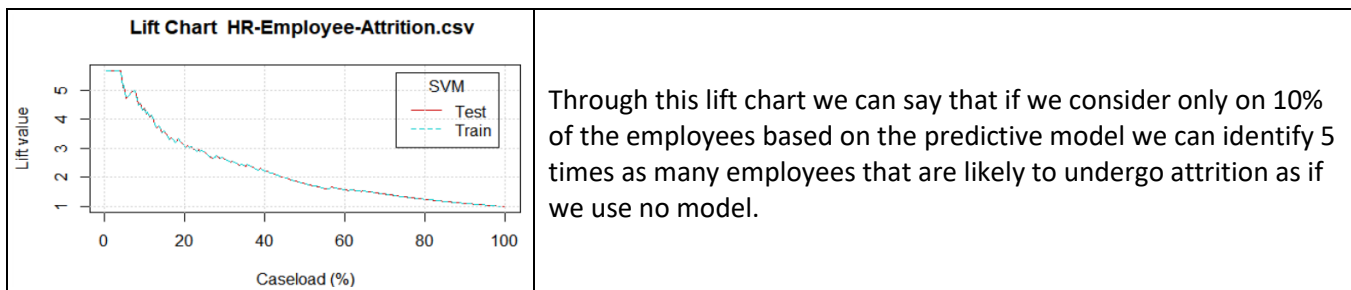
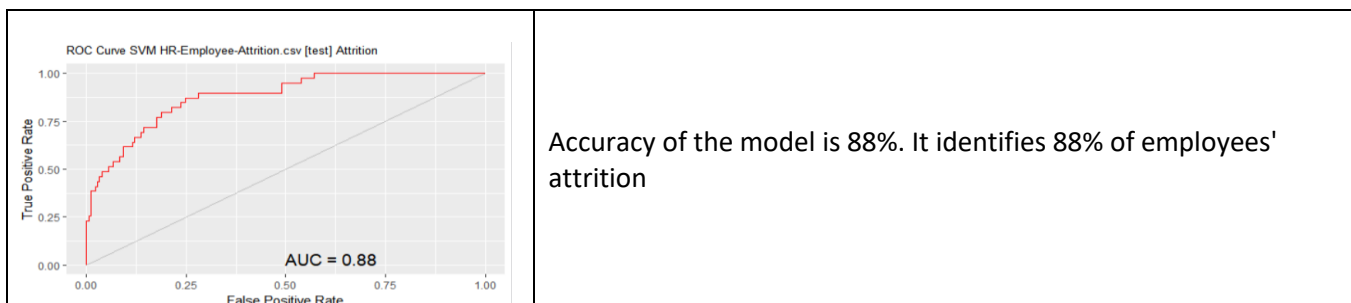
| Parameters | Linear Kernel | Radial Kernel | Laplacian Kernel |
|---|----------------------------------|----------------------------------|----------------------------------|
| Confusion matrix using training dataset | Predicted Actual No Yes Error | Predicted Actual No Yes Error | Predicted Actual No Yes Error |
| | No 82.7 1.5 1.7 | No 866 0 0.0 | No 866 0 0 |
| | Yes 8.7 7.1 55.2 | Yes 117 46 71.8 | Yes 163 0 100 |
| Overall Error | 10.2% | 11.3% | 15.8% |
| Average Class Error | 28.45% | 35.9% | 50% |
| Confusion matrix using test dataset | Predicted Actual No Yes Error | Predicted Actual No Yes Error | Predicted Actual No Yes Error |
| | No 79.6 2.7 3.3 | No 182 0 0.0 | No 182 0 0 |
| | Yes 10.0 7.7 56.4 | Yes 32 7 82.1 | Yes 39 0 100 |
| Overall Error | 12.7% | 14.4% | 17.6% |
| Average Class Error | 29.85% | 41.05% | 50% |

Since the error was least in linear kernel, I considered it for further analysis.

Performance Chart SVM HR-Employee-Attrition.csv [test]



Performance chart shows the tradeoff between the caseload (number of instances) and the performance of the model. It shows that the precision of the model decreases as we increase the caseload. When we use 100% of the test dataset, we can correctly predict 18% of employee's attrition. Whereas recall increases as well increase the caseload. The best recall obtained from this model is 89% i.e. we can identify 89% of employees who are likely to undergo attrition.



Algorithm 2: Decision Tree:

Comparison of Decision Tree before and after pruning

| Parameters | Before pruning | After Pruning |
|---|---|--|
| Confusion matrix using training dataset | Predicted Actual No Yes Error No 827 39 4.5 Yes 40 123 24.5 | Predicted Actual No Yes Error No 849 17 2.0 Yes 81 82 49.7 |
| Overall Error | 7.60% | 9.50% |
| Average Class Error | 14.50% | 25.85% |
| Confusion matrix using test dataset | Predicted Actual No Yes Error No 164 18 9.9 Yes 24 15 61.5 | Predicted Actual No Yes Error No 171 11 6.0 Yes 25 14 64.1 |
| Overall Error | 19.00% | 16.30% |
| Average Class Error | 35.70% | 35.05% |
| Complexity Parameter | CP nsplit rel error xerror xstd 1 0.0265849 0 1.00000 1.00000 0.071855 2 0.0214724 6 0.84049 0.98773 0.071495 3 0.0184049 9 0.76687 0.95706 0.070579 4 0.0168712 13 0.69325 0.95706 0.070579 5 0.0122699 17 0.62577 1.00000 0.071855 6 0.0061350 19 0.60123 1.06135 0.073598 7 0.0030675 28 0.54601 1.11043 0.074928 8 0.0023006 33 0.52761 1.16564 0.076359 9 0.0020450 41 0.50920 1.19632 0.077127 10 0.0015337 48 0.49080 1.19632 0.077127 11 0.0000100 52 0.48466 1.21472 0.077578 | CP nsplit rel error xerror xstd 0.026585 0 1.00000 1.00000 0.071855 0.021472 6 0.84049 0.98773 0.071495 0.018405 9 0.76687 0.95706 0.070579 0.016871 13 0.69325 0.95706 0.070579 0.012270 17 0.62577 1.00000 0.071855 0.010000 19 0.60123 1.03681 0.072912 |

From the table above, we can see that decision tree gives better result after pruning as the errors on the test set has reduced. For pruning the tree, I considered Complexity Parameter to be the key value. I pruned the tree till the value of CP reduced and were close for all the buckets. Complexity of the pruned tree is 0.01.

| | |
|---|--|
| <p>The chart shows Performance (%) on the left y-axis (0 to 100) and Lift on the right y-axis (1 to 5) against Caseload (%) on the x-axis (0 to 100). A solid green line represents Recall, which increases from 0% at 0% caseload to 67% at 100% caseload. A dotted blue line represents Precision, which starts at 100% for 0% caseload and decreases to 18% at 100% caseload. A diagonal grey line represents the baseline. A legend indicates Recall (67%) and Precision (18%).</p> | <p>When we use 100% of the test dataset, we can correctly predict 18% of employee’s attrition. Whereas recall increases as well increase the caseload. The best recall obtained from this model is 67% i.e. we can identify 67% of employees who are likely to undergo attrition. We can see the recall of this model is not good. It even falls below the baseline. So, we should not consider this model for prediction.</p> |
| <p>The chart shows Lift value on the y-axis (1.0 to 3.0) against Caseload (%) on the x-axis (0 to 100). Two lines are plotted: a solid red line for the Test set and a dashed blue line for the Train set. Both lines start at a lift of 1.0 for 0% caseload, peak at approximately 3.0 for 10% caseload, and then gradually decrease towards 1.0 as the caseload increases to 100%.</p> | <p>Through this lift chart we can say that if we can say although only on 10% of the employees based on the predictive model we can identify 4 times as many employees that are likely to undergo attrition as if we use no model, but it falls rapidly as we increase the size of our dataset.</p> |
| <p>The chart shows True Positive Rate on the y-axis (0.00 to 1.00) against False Positive Rate on the x-axis (0.00 to 1.00). A solid red line represents the model's performance, starting at (0,0) and ending at (1,1). A diagonal grey line represents the baseline. The area under the red curve is labeled as AUC = 0.64.</p> | <p>Accuracy of the model is 64%. It identifies 64% of employees' attrition. It also falls below the baseline after 80%.</p> |

Algorithm 3: Boosting

| | |
|--|--|
| <p>The chart shows Error on the y-axis (0.08 to 0.16) against Iteration 1 to 50 on the x-axis. A solid red line represents the training error, which starts at approximately 0.15 at iteration 1 and decreases steadily, reaching nearly 0.08 by iteration 50. A legend indicates '1 Train'.</p> | <p>We can see that training error reduces almost after every iteration. After 50 trees the error reached almost zero. Therefore, I used 50 trees for boosting.</p> |
|--|--|

| | |
|---|---|
| <p>Performance Chart Extreme Boost HR-Employee-Attrition.csv [test]</p> <p>This chart shows the tradeoff between caseload (number of instances) and model performance. The x-axis represents Caseload (%) from 0 to 100. The left y-axis represents Performance (%) from 0 to 100. The right y-axis represents Lift from 1 to 5. A green line shows Recall (88%), which increases with caseload. A blue dotted line shows Precision, which decreases as caseload increases. A black diagonal line represents random performance. A grey line at the top left indicates a Risk Score of 0.2.</p> | <p>Performance chart shows the tradeoff between the caseload (number of instances) and the performance of the model. It shows that the precision of the model decreases as we increase the caseload. When we use 100% of the test dataset, we can correctly predict 18% of employee's attrition. Whereas recall increases as well increase the caseload. The best recall obtained from this model is 88% i.e. we can identify 88% of employees who are likely to undergo attrition.</p> |
| <p>Lift Chart HR-Employee-Attrition.csv</p> <p>This chart shows the lift value on the y-axis (1 to 5) against the caseload percentage on the x-axis (0 to 100). It compares the performance of the Extreme Boost model (solid red line) with the Test (dashed red line) and Train (dashed blue line) datasets. The lift value starts high (around 5) at low caseload and decreases rapidly as the caseload increases, approaching a lift of 1 at 100% caseload.</p> | <p>Through this lift chart we can say that if we can say although only on 5% of the employees based on the predictive model we can identify more than 5 times as many employees that are likely to undergo attrition as if we use no model, but it falls rapidly as we increase the size of our dataset</p> |
| <p>ROC Curve Extreme Boost HR-Employee-Attrition.csv [test] Attrition</p> <p>This ROC curve shows the True Positive Rate on the y-axis (0.00 to 1.00) against the False Positive Rate on the x-axis (0.00 to 1.00). The red line represents the model's performance, which is significantly above the diagonal grey line representing random chance. The Area Under the Curve (AUC) is 0.87.</p> | <p>Accuracy of the model is 87%. It identifies 87% of employees' attrition</p> |

Model Comparison:

| | | |
|--|--|---|
| <p>Lift Chart HR-Employee-Attrition.csv [test]</p> <p>This chart compares the lift value (y-axis, 1.0 to 3.0) against caseload percentage (x-axis, 0 to 100) for three models: rpart (red solid line), ada (green dashed line), and ksvm (blue dotted line). ksvm and ada show higher lift values than rpart across most caseloads, indicating better performance.</p> | <p>Precision/Recall Plot HR-Employee-Attrition.csv [test]</p> <p>This chart compares precision (y-axis, 0.2 to 0.6) against recall (x-axis, 0.2 to 1.0) for the three models. ksvm (blue dotted line) generally maintains higher precision at higher recall values compared to rpart (red solid line) and ada (green dashed line).</p> | <p>Sensitivity/Specificity (tpr/fpr) HR-Employee-Attrition.csv</p> <p>This chart compares sensitivity (y-axis, 0.0 to 0.8) against specificity (x-axis, 0.0 to 1.0) for the three models. ksvm (blue dotted line) shows the highest sensitivity across most specificity levels, followed by ada (green dashed line) and rpart (red solid line).</p> |
| <p>From the lift chart for all the three models we can say that svm and boosting gives comparable result while decision tree is not very good for model generalization as it gives very few correct results</p> | <p>We can say from Precision/ Recall chart that for 80% of data boosting gives the best result while data from 90-100% svm performs better. We should not consider of decision tree in any of the cases.</p> | <p>Looking at Sensitivity/Specificity chart, we can say for 70% of data svm performs better than boosting while after that boosting is slightly better. Decision tree is not good in any case.</p> |

Conclusion:

Analyzing all the charts of various graphs we can say for this particular problem we should consider SVM or boosting as it shows the best results. Also, the accuracy of boosting and SVM are 87% and 89% respectively whereas through SVM we get only 64% accuracy.

Analysis of dataset 2

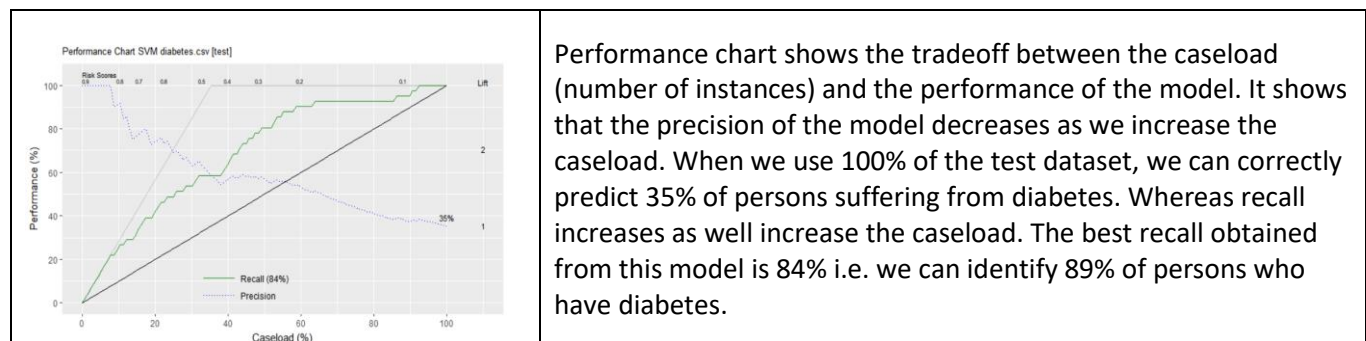
Data Description: Analytics play a huge role in Healthcare industry. Given the physical condition of a person it can be easily identified which disease is that person most likely to suffer. This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective is to predict based on diagnostic measurements whether a patient has diabetes. It consists of features like number of pregnancies, BMI, Blood Pressure, Skin Thickness and I am trying to predict whether the person is diabetic or not.

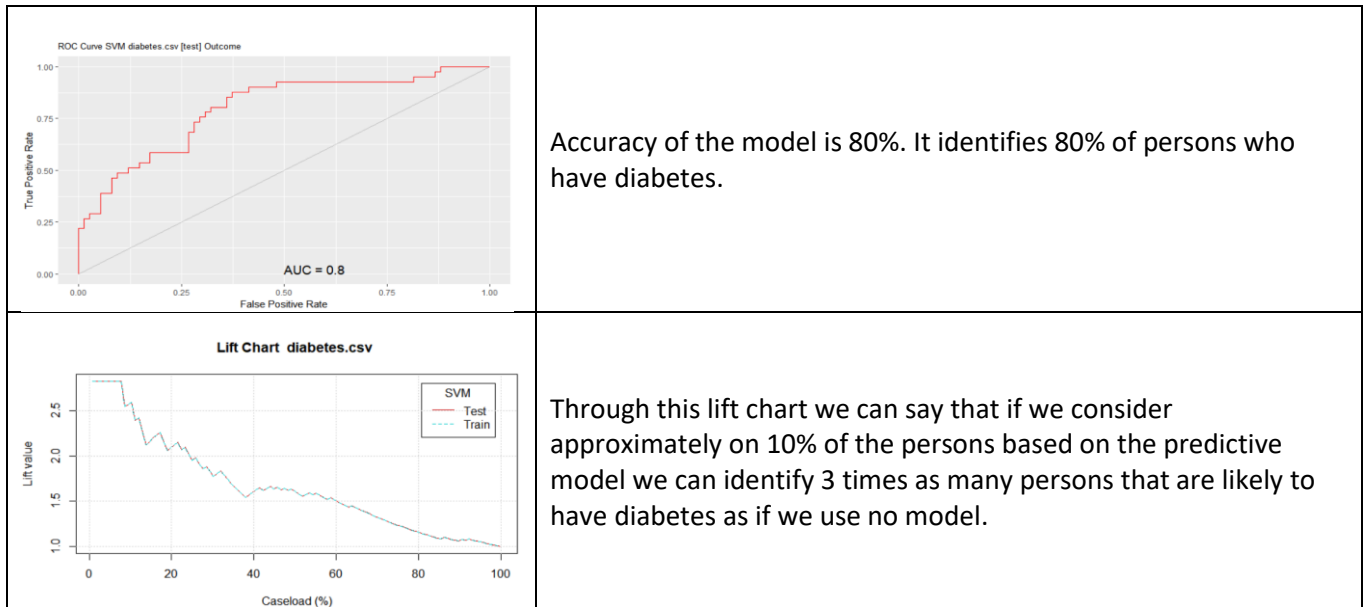
Algorithm 1: SVM

Comparison of the three kernels used:

| Parameters | Linear Kernel | Radial Kernel | Laplacian Kernel | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---------------|------------------|--|--|--------|---|---|-------|---|-----|----|------|---|----|-----|------|--|--|-----------|--|--|--------|---|---|-------|---|-----|----|------|---|----|-----|------|--|--|-----------|--|--|--------|---|---|-------|---|-----|----|------|---|----|-----|------|
| Confusion matrix using training dataset | <table><tr><td></td><td colspan="3">Predicted</td></tr><tr><td>Actual</td><td>0</td><td>1</td><td>Error</td></tr><tr><td>0</td><td>313</td><td>41</td><td>11.6</td></tr><tr><td>1</td><td>70</td><td>113</td><td>38.3</td></tr></table> | | Predicted | | | Actual | 0 | 1 | Error | 0 | 313 | 41 | 11.6 | 1 | 70 | 113 | 38.3 | <table><tr><td></td><td colspan="3">Predicted</td></tr><tr><td>Actual</td><td>0</td><td>1</td><td>Error</td></tr><tr><td>0</td><td>330</td><td>24</td><td>6.8</td></tr><tr><td>1</td><td>57</td><td>126</td><td>31.1</td></tr></table> | | Predicted | | | Actual | 0 | 1 | Error | 0 | 330 | 24 | 6.8 | 1 | 57 | 126 | 31.1 | <table><tr><td></td><td colspan="3">Predicted</td></tr><tr><td>Actual</td><td>0</td><td>1</td><td>Error</td></tr><tr><td>0</td><td>329</td><td>25</td><td>7.1</td></tr><tr><td>1</td><td>59</td><td>124</td><td>32.2</td></tr></table> | | Predicted | | | Actual | 0 | 1 | Error | 0 | 329 | 25 | 7.1 | 1 | 59 | 124 | 32.2 |
| | Predicted | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Actual | 0 | 1 | Error | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | 313 | 41 | 11.6 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 70 | 113 | 38.3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Predicted | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Actual | 0 | 1 | Error | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | 330 | 24 | 6.8 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 57 | 126 | 31.1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Predicted | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Actual | 0 | 1 | Error | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | 329 | 25 | 7.1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 59 | 124 | 32.2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Overall Error | 20.7% | 15% | 15.6% | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Average Class Error | 24.95% | 18.5% | 19.65% | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Confusion matrix using test dataset | <table><tr><td></td><td colspan="3">Predicted</td></tr><tr><td>Actual</td><td>0</td><td>1</td><td>Error</td></tr><tr><td>0</td><td>62</td><td>13</td><td>17.3</td></tr><tr><td>1</td><td>19</td><td>22</td><td>46.3</td></tr></table> | | Predicted | | | Actual | 0 | 1 | Error | 0 | 62 | 13 | 17.3 | 1 | 19 | 22 | 46.3 | <table><tr><td></td><td colspan="3">Predicted</td></tr><tr><td>Actual</td><td>0</td><td>1</td><td>Error</td></tr><tr><td>0</td><td>62</td><td>13</td><td>17.3</td></tr><tr><td>1</td><td>17</td><td>24</td><td>41.5</td></tr></table> | | Predicted | | | Actual | 0 | 1 | Error | 0 | 62 | 13 | 17.3 | 1 | 17 | 24 | 41.5 | <table><tr><td></td><td colspan="3">Predicted</td></tr><tr><td>Actual</td><td>0</td><td>1</td><td>Error</td></tr><tr><td>0</td><td>61</td><td>14</td><td>18.7</td></tr><tr><td>1</td><td>19</td><td>22</td><td>46.3</td></tr></table> | | Predicted | | | Actual | 0 | 1 | Error | 0 | 61 | 14 | 18.7 | 1 | 19 | 22 | 46.3 |
| | Predicted | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Actual | 0 | 1 | Error | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | 62 | 13 | 17.3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 19 | 22 | 46.3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Predicted | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Actual | 0 | 1 | Error | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | 62 | 13 | 17.3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 17 | 24 | 41.5 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Predicted | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Actual | 0 | 1 | Error | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | 61 | 14 | 18.7 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 19 | 22 | 46.3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Overall Error | 27.6% | 25.9% | 28.4% | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Average Class Error | 31.8% | 29.4% | 32.5% | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Since the error was least in radial kernel, I considered it for further analysis.





Algorithm 2: Decision Tree:

Comparison of Decision Tree before and after pruning

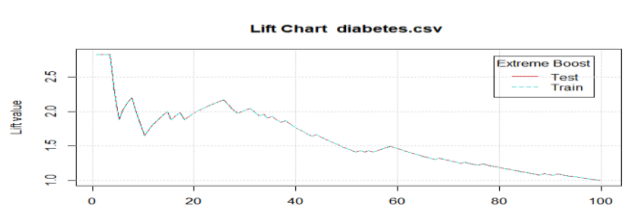
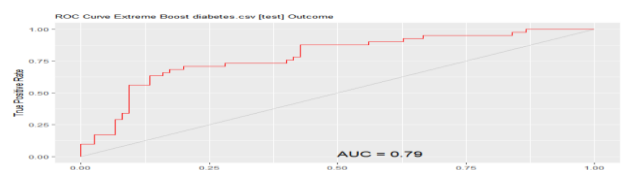
| Parameters | Before pruning | After Pruning |
|---|---|--|
| Confusion matrix using training dataset | Predicted Actual 0 1 Error 0 330 24 6.8 1 46 137 25.1 | Predicted Actual 0 1 Error 0 336 18 5.1 1 68 115 37.2 |
| | Overall Error | 13% |
| | Average Class Error | 15.95% |
| | Confusion matrix using test dataset | Predicted Actual 0 1 Error 0 64 11 14.7 1 14 27 34.1 |
| Overall Error | | 21.5% |
| Average Class Error | | 24.4% |
| Complexity Parameter | | CP nsplit rel error xerror xstd 1 0.3060109 0 1.00000 1.00000 0.060019 2 0.1038251 1 0.69399 0.74863 0.055202 3 0.0200364 2 0.59016 0.62295 0.051783 4 0.0136612 5 0.53005 0.66667 0.053060 5 0.0109290 7 0.50273 0.70492 0.054099 6 0.0095628 10 0.46995 0.72678 0.054661 7 0.0091075 14 0.43169 0.72678 0.054661 8 0.0027322 18 0.39344 0.76503 0.055593 9 0.0018215 20 0.38798 0.76503 0.055593 10 0.0000100 23 0.38251 0.76503 0.055593 |

From the table above, we can see that decision tree gives better result after pruning as the errors on the test set has reduced. For pruning the tree, I considered Complexity Parameter to be the key value. I pruned the tree till the value of CP reduced and were close for all the buckets. Complexity of the pruned tree is 0.01.

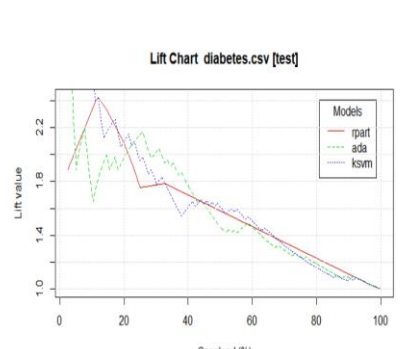
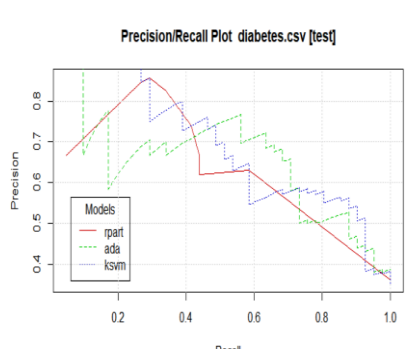
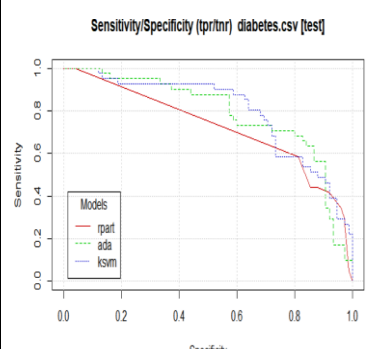
| | |
|--|---|
| <p>Performance Chart Decision Tree diabetes.csv [test]</p> <p>This chart shows the tradeoff between caseload and performance for a Decision Tree model. The x-axis represents Caseload (%) from 0 to 100. The left y-axis represents Performance (%) from 0 to 100. The right y-axis represents Lift from 1 to 2. A solid green line shows Recall (78%), which increases from 0% at 0% caseload to 78% at 100% caseload. A dotted blue line shows Precision, which starts at approximately 85% and decreases to approximately 35% at 100% caseload. A diagonal line represents the baseline performance.</p> | <p>When we use 100% of the test dataset, we can correctly predict 35% of people with diabetes. Whereas recall increases as well increase the caseload. The best recall obtained from this model is 78% i.e. we can identify 78% of people who are likely to have diabetes. We can see the recall of this model is not good.</p> |
| <p>Lift Chart diabetes.csv</p> <p>This chart shows the lift value for a Decision Tree model. The x-axis represents Caseload (%) from 0 to 100. The y-axis represents Lift value from 1.0 to 2.2. A solid blue line represents the Decision Tree model, which starts at a lift of approximately 1.8 at 0% caseload, peaks at approximately 2.4 at 10% caseload, and then decreases to approximately 1.0 at 100% caseload. A dotted blue line represents the baseline lift, which is constant at 1.0.</p> | <p>Through this lift chart we can say that if we can say although only on 10% of the employees based on the predictive model we can identify 2.4 times as many people that are likely to have diabetes as if we use no model.</p> |
| <p>ROC Curve Decision Tree diabetes.csv [test] Outcome</p> <p>This chart shows the ROC curve for a Decision Tree model. The x-axis represents False Positive Rate from 0.00 to 1.00. The y-axis represents True Positive Rate from 0.00 to 1.00. A solid red line represents the ROC curve, which starts at (0,0) and ends at (1,1). The area under the curve is labeled as AUC = 0.72.</p> | <p>Accuracy of the model is 72%. It identifies 72% of people who are likely to have diabetes.</p> |

Algorithm 3: Boosting

| | |
|---|--|
| <p>Training Error</p> <p>This chart shows the training error for a Boosting model. The x-axis represents Iteration 1 to 60. The y-axis represents Error from 0.12 to 0.24. A solid red line shows the training error, which starts at approximately 0.23 and decreases to approximately 0.12 after 60 iterations. The error curve is wiggly but generally decreases.</p> | <p>We can see that training error reduces almost after every iteration. It is a wiggly curve but after 60 iteration the error reached almost zero. Therefore, I used 60 iterations for boosting.</p> |
| <p>Performance Chart Extreme Boost diabetes.csv [test]</p> <p>This chart shows the tradeoff between caseload and performance for an Extreme Boost model. The x-axis represents Caseload (%) from 0 to 100. The left y-axis represents Performance (%) from 0 to 100. The right y-axis represents Lift from 1 to 2. A solid green line shows Recall (83%), which increases from 0% at 0% caseload to 83% at 100% caseload. A dotted blue line shows Precision, which starts at approximately 85% and decreases to approximately 35% at 100% caseload. A diagonal line represents the baseline performance.</p> | <p>Performance chart shows the tradeoff between the caseload (number of instances) and the performance of the model. It shows that the precision of the model decreases as we increase the caseload. When we use 100% of the test dataset, we can correctly predict 35% of people who might have diabetes. Whereas recall increases as well increase the caseload. The best recall obtained from this model is 83% i.e. we can identify 83% of people who are likely to have diabetes.</p> |

| | |
|---|---|
|  | <p>This lift chart is quite confusing to predict anything. We can assume on 25% of the employees based on the predictive model we can identify more than more than 2 times as many people that are likely to have diabetes as if we use no model, but it falls rapidly as we increase the size of our dataset</p> |
|  | <p>Accuracy of the model is 79%. It identifies 79% of people who are likely to have diabetes.</p> |

Model Comparison:

| | | |
|---|---|--|
|  |  |  |
| <p>From the lift chart for all the three models we can say that svm and boosting gives comparable result while decision tree is not very good for model generalization as it gives very few correct results</p> | <p>We can say from Precision/ Recall chart that for 50% of data svm gives the best result while data from 50-70% boosting performs better. We should not consider of decision tree in any of the cases.</p> | <p>Looking at Sensitivity/Specificity chart, we can say for 70% of data svm performs better than boosting while after that boosting is slightly better. Decision tree is not good in any case.</p> |

Conclusion:

Analyzing all the charts of various graphs we can say in order to predict if a person has diabetes or not from this data set, we should consider SVM or boosting as it shows the best results. Also, the accuracy of boosting and SVM are 79% and 80% respectively whereas through SVM we get only 72% accuracy.