

Figure 1: Learning curve J48 for Chess data set

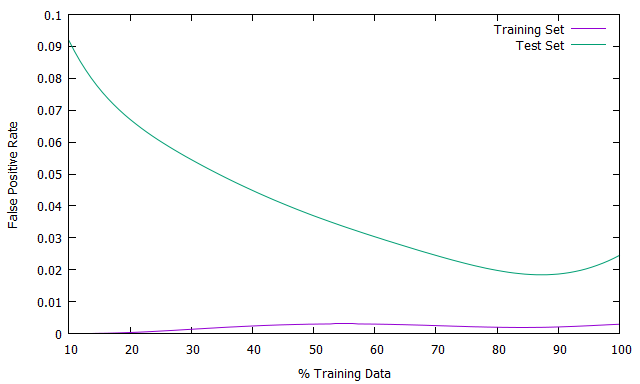


Figure 2: Learning curve for MultiLayer Perceptron for Chess Data set

* For ANN
  + You can see it has a steeper learning curve compared to some of the others. It needs more data to make an accurate model
* For polykernel
  + For SMO changing the exponent to 5 in polykernel options gives better results. Moving it to 15 gives 50% FPR on test set. 10 does better than 15 but worse than 5. 5 seems to be ideal.
  + w/ exponent of 5 it performs perfectly on training data. I think this means it has a wide margin that is also a perfect separator of the data.
  + Maybe use FNR for this because w/ change in exponent there is a 80% reduction in FNR compared to 50% reduction in FPR
* For RBFKernel
  + Changing gamma to 0.1 cuts FPR in half on test set.
  + Chaning gamma to 0.3 cuts FPR by factor of 10 on test set
  + Gamme to 0.5 does even better
  + With gamme of 1 it doesn’t perform as well as 0.5

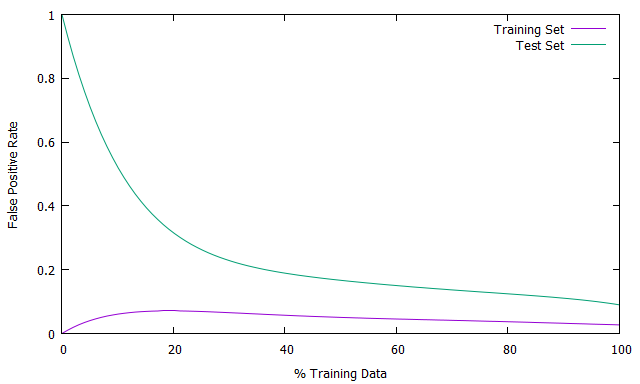


Figure : Learning curve for Nearest Neighbors algorithm for Chess data set

* For IBK
  + 1 overfits training data
  + 2 doesn’t overfit but doesn’t do any better on test set
  + 3 has about 20% less false positives
  + 4 and above does worse
  + Manhattan and Minkowski distance has a small effect
* For Ada it doesn’t make sense that we are getting 0 erros because our test error is continually decreasing
  + Perhaps it is because ada builds a model and subsequently tests that model on test set. There are many different models that could perform perfectly on training set but when each model is applied to the test set it gives different results. This is probably high variance. More in stanford packet.

------------------------------------------------------------------------------------------------------------------------

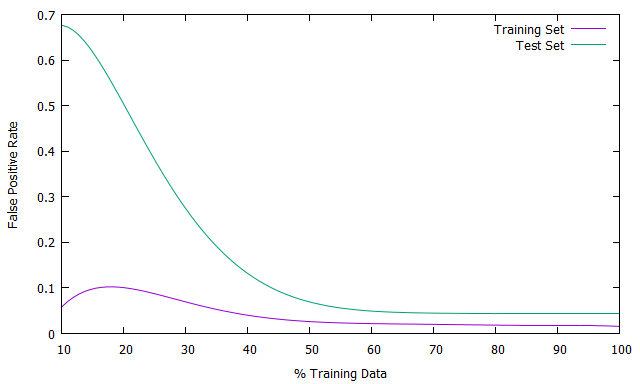


Figure : Multilayer Perceptron Learning curve for tic tac toe set

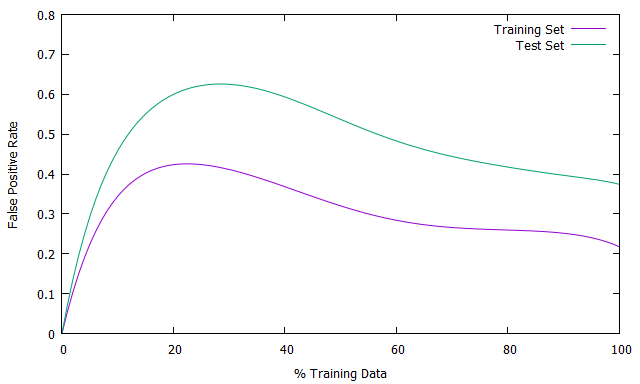


Figure: J48 Learning Curve for tic tac toe set

* For J48
  + Turning off pruning actually cuts FPR in half for test set.

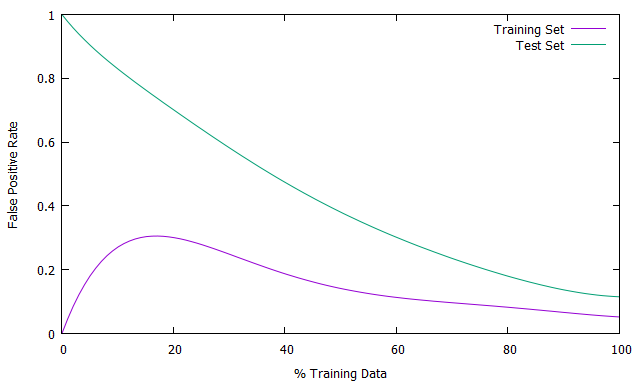


Figure : IBk learning curve for tic tac toe set

|  |  |  |
| --- | --- | --- |
| Model | Chess | Tic Tac Toe |
| J48 | 0.03 seconds |  |
| MultiLayer Perceptron | 0.5 seconds (2 hidden) (100 epoch) |  |
| SMO PolyKernel | 1.4 seconds (exponent = 5) |  |
| SMO RBFKernel | 3.44 seconds (gamma = 0.5) |  |
| ADABoostM1 | 0.25 seconds |  |
| IBk | 0 seconds |  |

Figure : Time to build model

|  |  |  |
| --- | --- | --- |
| Model | Chess | Tic Tac Toe |
| J48 | 0 seconds |  |
| MultiLayer Perceptron | 0.01 seconds |  |
| SMO PolyKernel | 0.18 seconds |  |
| SMO RBFKernel | 0.43 seconds |  |
| ADABoostM1 | 0.01 seconds |  |
| IBk | 1.1 seconds (3NN) |  |

Figure: Time to apply model

|  |  |  |
| --- | --- | --- |
|  | Chess | Tic Tac Toe |
| J48 | 21 kb |  |
| MultiLayer Perceptron | 35 kb |  |
| SMO PolyKernel | 776 kB |  |
| SMO RBFKernel | 792 kB |  |
| ADABoostM1 | 129 kB |  |
| Idk | 162 kB |  |

Figure : Space required for model