**For binary mouse w/ noise and no behavior data set**

* After removing all but the last 15 attributes the neural net performed much better
  + Dec tree performed worse
* With all attributes including behavior
  + Neural net with 2 hidden layers classifies everything as a in conf matrix
  + Neural net with 20 hidden layers classifies everything as b in conf matrix
  + w/ 10 hidden layers and 1000 epochs all is classified as b
* Removing Behavior created some errors in confusion matrix for dec tree
* Rduced error pruning increases errors in confusion matrix for dec tree
  + Means less overfitting
* IBK performs ok about 70 % precision but not as good as dec tree and neural nets after removing all but last 15 attributes. Increasing k seems to make it worse
  + Using manhattan distance seems to work better
* May want to try adding more missing values
* Observations
  + Neural net performs well w/ 5 hidden layers and 100 epochs
  + Dec tree doesn’t perform very well
  + Ada boost does slightly better than J48
    - Changing J48 params (conf factor, unpruned, collapse tree) doesn’t make it perform
      * w/ these factors changed and 100 iterations performance increased slightly. Increasing to 1000 iterations doesn’t help
      * using resampling didn’t help
  + IBK does about the same as dec tree w/ 5 neighbors and manhattan distance. It performs better with a than b compared to dec tree. With 1 neighbor it doesn’t do well
  + SMO performs well w/ polykernel but has a lot of errors on a w/ RBFKernel

**For chess data set**

* Observations
  + Decision tree performs excellent, only 4 errors totals in test set
  + ANN performs exceelent, very few errors
  + IBK performs well but has about 70 errors total, a lot more than dec tree and ANN
  + SMO performs well but has about 50 errors total w/ polykernel
    - Rbfkernel has about 80 errors
  + ADA performs well

**Tic tac toe**

* Observations
  + Decision tree performs poorly
  + Boosting j48 performs well
    - Adding 10% noise really messes up boosting even on a. If you look at a in the J48 w/ noise it isn’t doing much worse but w/ ada it gets much worse w/ noise
  + ANN performs well
  + SMO performs well with poly kernel
    - w/ rbf kernel it missclassifes all the negative examples
  + IBK performs well w/ positive class not so well w/ negative class

**For tic tac toe split into training/test set**

* For J48 the cross validation does bad but the training set does well
  + They both perform exactly the same on the test set
* For ANN training set and cross validation does well
  + Both perform well on test set, exact same error values
* SMO performs well w/ polykernel w/ cross validate and training set
  + Both perform well on test set, exact same error values again
* SMO perofmrs poorly with rbf kernel on both cross val and training set. Misclassifies most of negative examples
  + Both misclassify negative examples in test set
* IBK w/ only 1NN performs perfect w/ traing set and good w/ cross val
  + Both perform well on test set. Better on positive class than negative class. Exact same error rates again.
* ADA performs perfect w/ training set, good w/ cross val
  + Both perform exactly same on test set

**General**

* By removing important attributes dec tree, k-NN, and ANN start performing worse
  + Maybe try adding missing values to see if we can exaggerate performance difference
* By adding noise to the data J48 performed worse but ADAboost performed even worse than J48
  + Goes along with fact that ADA weakness is noise in data. Having trouble reproducing this
* Good thing to add is how well alg performed on training set compared to testing set. So maybe compare the conf matrices of the two
* To test if data is linearly seperable we can use SVM
* Plot error rates against different attributes
* Dec tree probably performs well on chess data because it is all nominal. J48 uses ranges when splitting numeric data and that probably causes inaccuracies
* It seems that in general the better the j48 performs the better adaboost performs
* Sets w/ all nominal attributes aren’t handled well by algs if noise is added. The mouse set had numeric data so I think adding noise some of the algs were still able to handle it
* Need to rewatch lecture that talks about cross validation
* Error in weka? When doing auto 66% split it misclassifies a few examples after randomizing the order. Did this w/ j48
* Explain why you use FPR and FNR in learning curves instead of accuracy
* Can you re evaluate on a separate test set after performing cross validation
  + Sure, why not. The more data sets you have, the better. The point of having separate data sets in the first place is to avoid overfitting of the models you learn to the data and getting a misleading impression of the performance of the model. Using separate test sets and techniques like cross-validation ensures that you get a more accurate and reasonable picture of the performance of your model.
  + The separate test set can be used to confirm the performance estimate computed during cross-validation.

So, here is the scenario again: you have 100 labeled data

**Use training set**

* weka will take 100 labeled data
* it will apply an algorithm to build a classifier from these 100 data
* it applies that classifier AGAIN on these 100 data
* it provides you with the performance of the classifier (applied to the same 100 data from which it was developed)

**Use 10 fold CV**

* Weka takes 100 labeled data
* it produces 10 equal sized sets. Each set is divided into two groups: 90 labeled data are used for training and 10 labeled data are used for testing.
* it produces a classifier with an algorithm from 90 labeled data and applies that on the 10 testing data for set 1.
* It does the same thing for set 2 to 10 and produces 9 more classifiers
* it averages the performance of the 10 classifiers produced from 10 equal sized (90 training and 10 testing) sets

Let me know if that answers your question.