Name: Ronny Pena
Email: pena1164@gatech.edu

Assignment: Supervised Learning

Predicting congressmen party affiliations based on voting behavior.
Dataset: http://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records
Algorithms evaluated:

- ML Decision Tree - J48 - http://weka.sourceforge.net/doc.stable/weka/classifiers/trees/J48.html
- ML Support Vector - SMO - http://weka.sourceforge.net/doc.stable/weka/classifiers/functions/SMO.html
- ML Neural Network - MultilayerPerceptron - http://weka.sourceforge.net/doc.stable/weka/classifiers/functions/MultilayerPerceptron.html
- ML Nearest Neighbor - IBK - http://weka.sourceforge.net/doc.stable/weka/classifiers/lazy/IBk.html
- ML Boosting - AdaBoostM1 - http://weka.sourceforge.net/doc.stable/weka/classifiers/meta/AdaBoostM1.html

For this first problem, I chose the congress dataset because it mostly consists of boolean data. In some cases the votes contains "?" for when the vote is unknow. Based on this fact, it is easy to create a decision tree and visualize it. Decision trees are perfect for machine learning problems where the attribute values are binary and it contains many attributes (17 attributes in this case). This dataset contains missing values in some of the attributes and some empty records. All attributes are important for evaluating behavior so some of them will only be removed to test the performance of a given algorithm during the analysis. This dataset contains 435 records.

Weka's **ID3** algorithm implementation cannot be used because of empty records.

For my implementation, I used Weka GUI and my custom java test program. I am splitting the data into a training set and a test set 66% to 33% in Weka GUI and 70% to 30% in my java program. Since my java program receives different amount of training data, performance results are usually a little or higher lower than Weka GUI results.

The "Adult" dataset is another interesting classification problem. This dataset contains numeric attributes in addition to nominal values. **Dataset:** https://archive.ics.uci.edu/ml/datasets/Adult

Observations for Decision Tree:
When ReducedErrorPruned is used instead of ConfidenceFactor, the algorithm uses a smaller decision tree. It weka GUI run faster but the Mean squared error and Root mean squared error are lower. I chose ReducedErrorPruned because it was the simplest form of pruning. In the java version, the results about almost the same.

Observations for Support Vector Machine
As another experiment, I used SVM as a classifier and got worse results than the decision tree. It converts all missing values to normalized values. SVM algorithm had equal or worse performance for this dataset. I performed test with 17 attributes and with 3 of the major attributes: "water-project-cost-sharing", "physician-fee-freeze", "mx-missile" but also reduced performance to Correctly Classified Instances 113 81.295 % and Incorrectly Classified Instances 26 18.705 %

Observations for Neural Network:
As another experiment, I used a standard neural network. Performance was better but 1.25 seconds to build the model. Had to use a training time of 500 epochs. This generated neural network has better performance than the decision tree but since the data doesn't have much noise a decision tree will perform faster. The network has one hidden layer. Removing data attribute AKA units from the dataset speeds up the performance but increases the error rate. I still need to test this data set with a manually created neural network and compare the performance. I've included the execution time for both decision tree and the neural network. They are included in the observation results.

Observations for KNN:
As another experiment, I used IBK because you can select the number of neighbors, IB1 uses one neighbor. Performance for IB1 and IBK is the same using the dataset. in KNN, reducing the number of data attributes AKA dimensions improves the performance for KNN. I performed test with 17 attributes and with 3 of the major attributes: “water-project-cost-sharing”, “physician-fee-freeze”, “mx-missile”, there was a 4% performance increase. Error rate went from 91.89% to 96.62. Increasing the number of neighbors to 3 did not change the outcome.

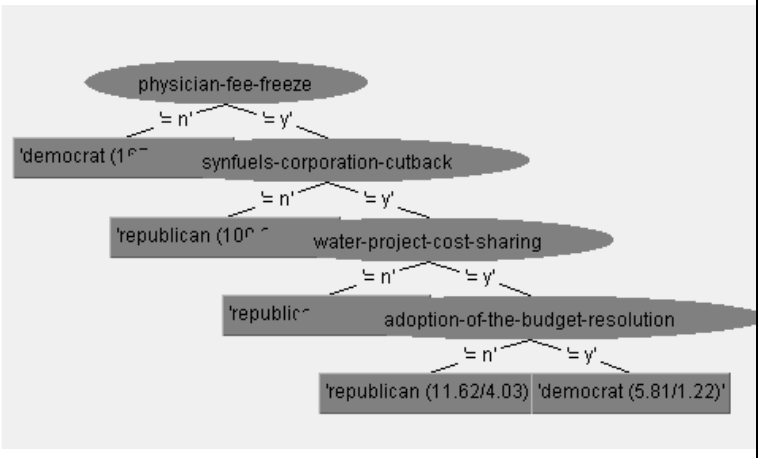
Observations for Boosting:
As another experiment, I used AdaBoostM1 combined with j48 did not provide any improvement. Also tried with multilayerperceptron, same result. I tried j48 with pruning and confidences, same results. I tried with percentage split and cross validation, same result. I believe this is due to the number of records, a data set with more instances would have a different outcome.

Results with decision tree:

Options used for ReducedErrorPruned enabled
WEKA GUI
Options: J48 -R -N 3 -Q 1 -M 2

Time taken to build model: 0 seconds
=== Summary ===

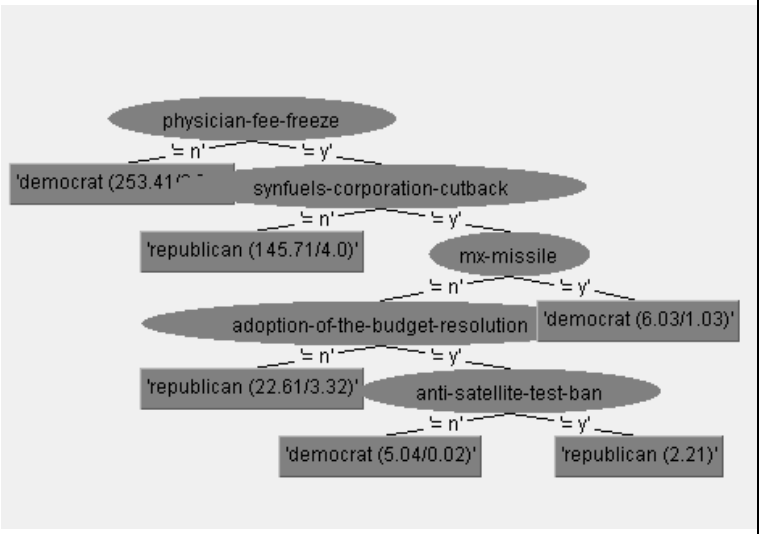
Correctly Classified Instances	144	**97.2973 %**
Incorrectly Classified Instances	4	2.7027 %
Kappa statistic	0.9447	
Mean absolute error	0.0586	
Root mean squared error	0.1571	
Relative absolute error	12.2266 %	
Root relative squared error	31.6927 %	
Total Number of Instances	148	
Number of Leaves :	5	
Size of the tree :	9	



Options used for confidenceFactor enabled:
WEKA GUI
Options: J48 -C 0.25 -M 2

Time taken to build model: 0.01 seconds
=== Summary ===

Correctly Classified Instances	144	97.2973 %
Incorrectly Classified Instances	4	2.7027 %
Kappa statistic	0.9447	
Mean absolute error	0.0608	
Root mean squared error	0.1539	
Relative absolute error	12.6846 %	
Root relative squared error	31.0328 %	
Total Number of Instances	148	
Number of Leaves :	6	
Size of the tree :	11	



Java Program with ReducedErrorPrunded enabled
classifier.setOptions(new String[] {"-R", "-N", "3", "-Q", "1", "-M", "2"});

Dataset size: 435
Training Set size: 304

Java Program with confidenceFactor enabled:
classifier.setOptions(new String[] {"-C", "0.25", "-M", "2"});

Dataset size: 435
Training Set size: 304

Test Set size: 131

Results
======

Correctly Classified Instances	128	**97.7099** %
Incorrectly Classified Instances	3	2.2901 %
Kappa statistic	0.952	
Mean absolute error	0.0489	
Root mean squared error	0.1571	
Relative absolute error	10.3096 %	
Root relative squared error	32.2176 %	
Total Number of Instances	131	

Execution time for testing: 14

Number of leaves: 7
Size of the tree: 13

Test Set size: 131

Results
======

Correctly Classified Instances	127	**96.9466** %
Incorrectly Classified Instances	4	3.0534 %
Kappa statistic	0.9358	
Mean absolute error	0.0553	
Root mean squared error	0.1622	
Relative absolute error	11.6426 %	
Root relative squared error	33.2552 %	
Total Number of Instances	131	

Execution time for testing: 12

Number of leaves: 5
Size of the tree: 9

Results for Support Vector Machines:

Options with NormalizedPolyKernel
Scheme:weka.classifiers.functions.SMO -C 1.0 -L 0.001 -P 1.0E-12 -N 0 -M -V -1 -W 1 -K "weka.classifiers.functions.supportVector.NormalizedPolyKernel -C 250007 -E 2.0"

Number of support vectors: 83
Number of kernel evaluations: 158150 (82.524% cached)
Time taken to build model: 0.26 seconds

=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances	144	**97.2973** %
Incorrectly Classified Instances	4	2.7027 %
Kappa statistic	0.9447	
Mean absolute error	0.0541	
Root mean squared error	0.1442	
Relative absolute error	11.2919 %	
Root relative squared error	29.0772 %	
Total Number of Instances	148	

Options with PolyKernel
Scheme:weka.classifiers.functions.SMO -C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions.supportVector.PolyKernel -C 250007 -E 1.0"

Number of kernel evaluations: 17182 (78.602% cached)
Time taken to build model: 0.1 seconds

=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances	143	**96.6216** %
Incorrectly Classified Instances	5	3.3784 %
Kappa statistic	0.9311	
Mean absolute error	0.0338	
Root mean squared error	0.1838	
Relative absolute error	7.0536 %	
Root relative squared error	37.0738 %	
Total Number of Instances	148	

Java Program

```
classifier.setOptions(new String[] {"-C", "1.0", "-L", "0.001", "-P",
"1.0E-12", "-N", "0", "-M", "-V", "-1",
            "-W", "1", "-K",
"weka.classifiers.functions.supportVector.NormalizedPolyKernel
-C 250007 -E 2.0"}); // set
```

Dataset size: 435
Training Set size: 304
Test Set size: 131

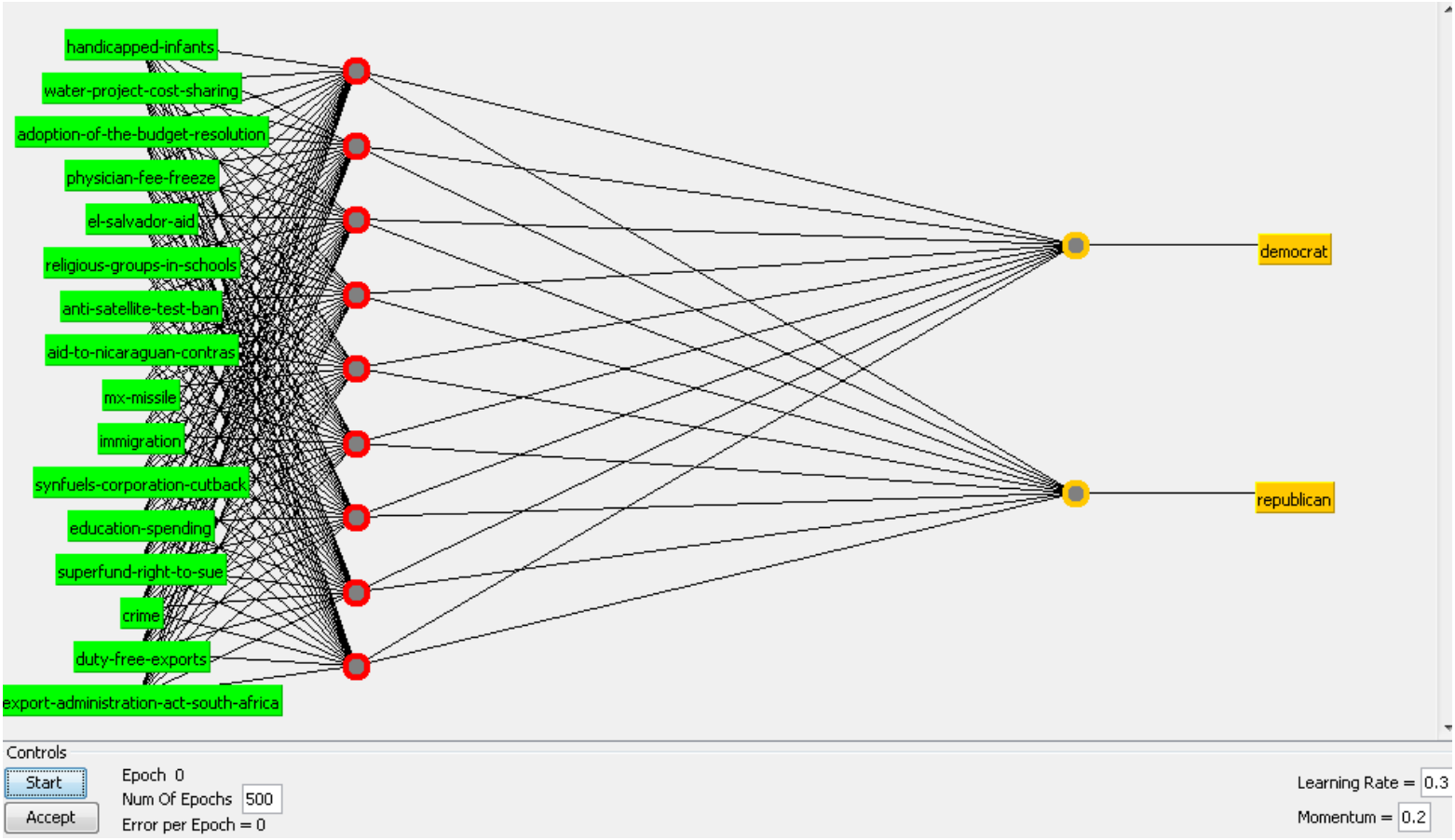
Results
======

Correctly Classified Instances	127	**96.9466** %
Incorrectly Classified Instances	4	3.0534 %
Kappa statistic	0.9362	

Mean absolute error 0.0543 Root mean squared error 0.1613 Relative absolute error 11.4431 % Root relative squared error 33.086 % Total Number of Instances 131 Execution time for testing: 23	

Results for Neural Network:

Generated network with auto build:



Weka options: MultilayerPerceptron -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a Time taken to build model: 1.25 seconds === Evaluation on test split === === Summary === Correctly Classified Instances 146 **98.6486** % Incorrectly Classified Instances 2 1.3514 % Kappa statistic 0.9724 Mean absolute error 0.0222	**Java Program** classifier.setOptions(new String[] {"-L", "0.3", "-M", "0.2", "-N", "500", "-V", "0", "-S", "0", "-E", "20", "-H", "a"}); Dataset size: 435 Training Set size: 304 Test Set size: 131 Results ====== Correctly Classified Instances 125 **95.4198** % Incorrectly Classified Instances 6 4.5802 % Kappa statistic 0.9044 Mean absolute error 0.047

Root mean squared error 0.1134 Relative absolute error 4.6257 % Root relative squared error 22.8741 % Total Number of Instances 148	Root mean squared error 0.2026 Relative absolute error 9.9075 % Root relative squared error 41.5567 % Total Number of Instances 131 Execution time for testing: 20

Results for KNN:

Weka options: IB1 Instances: 435 Attributes: **17** Time taken to build model: 0 seconds === Evaluation on test split === === Summary === Correctly Classified Instances 136 **91.8919%** Incorrectly Classified Instances 12 8.1081 % Kappa statistic 0.8371 Mean absolute error 0.0811 Root mean squared error 0.2847 Relative absolute error 16.9286 % Root relative squared error 57.4345 % Total Number of Instances 148	**Weka options:** IB1 Instances: 435 Attributes: **5** Time taken to build model: 0 seconds === Evaluation on test split === === Summary === Correctly Classified Instances 141 **95.2703%** Incorrectly Classified Instances 7 4.7297 % Kappa statistic 0.9039 Mean absolute error 0.0473 Root mean squared error 0.2175 Relative absolute error 9.875 % Root relative squared error 43.8663 % Total Number of Instances 148
Weka options: IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\"" Instances: 435 Attributes: **17** Time taken to build model: 0 seconds === Evaluation on test split === === Summary === Correctly Classified Instances 136 **91.8919%** Incorrectly Classified Instances 12 8.1081 % Kappa statistic 0.8364 Mean absolute error 0.078 Root mean squared error 0.2541 Relative absolute error 16.2899 % Root relative squared error 51.2431 % Total Number of Instances 148	**Weka options:** IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\"" Instances: 435 Attributes: **5** Time taken to build model: 0 seconds === Evaluation on test split === === Summary === Correctly Classified Instances 143 **96.6216%** Incorrectly Classified Instances 5 3.3784 % Kappa statistic 0.9311 Mean absolute error 0.0716 Root mean squared error 0.1693 Relative absolute error 14.9535 % Root relative squared error 34.1423 % Total Number of Instances 148
Java options: default options Dataset size: 435 Training Set size: 304 Test Set size: 131	**Java options:** default options Using removed attributes Dataset size: 435 Training Set size: 304 Test Set size: 131

Results
======

Correctly Classified Instances	122	**93.1298 %**
Incorrectly Classified Instances	9	6.8702 %
Kappa statistic	0.857	
Mean absolute error	0.0764	
Root mean squared error	0.2482	
Relative absolute error	16.0936 %	
Root relative squared error	50.9105 %	
Total Number of Instances	131	

Execution time for testing: 118

Results
======

Correctly Classified Instances	127	**96.9466 %**
Incorrectly Classified Instances	4	3.0534 %
Kappa statistic	0.9362	
Mean absolute error	0.0531	
Root mean squared error	0.1659	
Relative absolute error	11.1921 %	
Root relative squared error	34.013 %	
Total Number of Instances	131	

Execution time for testing: 68

Results for Boosting:

Weika GUI Options:
AdaBoostM1 -P 100 -S 1 -I 10 -W weka.classifiers.trees.J48 -- -R -N 3 -Q 1 -M 2

Number of performed Iterations: 7

Time taken to build model: 0.01 seconds

=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances	144	97.2973 %
Incorrectly Classified Instances	4	2.7027 %
Kappa statistic	0.9447	
Mean absolute error	0.0372	
Root mean squared error	0.1535	
Relative absolute error	7.7726 %	
Root relative squared error	30.9615 %	
Total Number of Instances	148	

Weka GUI Options:
AdaBoostM1 -P 100 -S 1 -I 10 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2

Number of performed Iterations: 9

Time taken to build model: 0.02 seconds

=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances	143	96.6216 %
Incorrectly Classified Instances	5	3.3784 %
Kappa statistic	0.9308	
Mean absolute error	0.0345	
Root mean squared error	0.1695	
Relative absolute error	7.2119 %	
Root relative squared error	34.1865 %	
Total Number of Instances	148	

Weika GUI Options:
AdaBoostM1 -P 100 -S 1 -I 50 -W weka.classifiers.functions.MultilayerPerceptron -- -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a

Number of performed Iterations: 2

Time taken to build model: 3.54 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances	418	96.092 %
Incorrectly Classified Instances	17	3.908 %
Kappa statistic	0.9178	
Mean absolute error	0.0404	
Root mean squared error	0.1847	
Relative absolute error	8.5248 %	
Root relative squared error	37.9273 %	
Total Number of Instances	435	

Java program options:
classifier.setOptions(new String[] {"-P", "100", "-S", "1", "-I", "10", "-W", "weka.classifiers.trees.J48",
"--", "-R", "-N", "3", "-Q", "1", "-M", "2"});

Results
======

Correctly Classified Instances	128	97.7099 %
Incorrectly Classified Instances	3	2.2901 %
Kappa statistic	0.952	
Mean absolute error	0.0397	
Root mean squared error	0.1508	
Relative absolute error	8.3602 %	
Root relative squared error	30.9295 %	
Total Number of Instances	131	

Execution time for testing: 3

Web Resources:

Discussion about data source generation:
http://programmers.stackexchange.com/questions/191025/machine-learning-with-sample-data-set

Where to find data:
http://stats.stackexchange.com/questions/33475/where-can-i-find-datasets-usefull-for-testing-my-own-machine-learning-implementa

Data set considerations:
Boston housing data: http://tunedit.org/repo/UCI/numeric/housing.arff
Census data / Income prediction: https://archive.ics.uci.edu/ml/datasets/Adult
Car Evaluation: https://archive.ics.uci.edu/ml/datasets/Car+Evaluation
Cancer prediction: https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29
Contraceptive choices: https://archive.ics.uci.edu/ml/datasets/Contraceptive+Method+Choice
Internet Ad remover: https://archive.ics.uci.edu/ml/datasets/Internet+Advertisements
Credit score: https://archive.ics.uci.edu/ml/datasets/Statlog+(German+Credit+Data)

Dataset libraries:

- http://archive.ics.uci.edu/ml/
- https://archive.ics.uci.edu/ml/datasets.html
- http://azure.microsoft.com/en-us/documentation/articles/machine-learning-use-sample-datasets/
- http://www.kdnuggets.com/datasets/index.html
- http://mldata.org/

Java Libraries consideration:

	Weka	Java-ML	MLlib	mahout
Decision trees with some form of pruning	Y	does have RandomForst but requires weka plugin for the rest of the algorithms	Y	?
Neural networks	Y	?	?	Y
Boosting with decision tree	Y	?	?	?
Support Vector Machines	Y but easier to use SMO instead of LibSVM	Y	Y	Y
k-nearest neighbors	Y	Y	?	?

Choosing and evaluating ML algorithms:
http://machinelearningmastery.com/how-to-evaluate-machine-learning-algorithms/
http://machinelearningmastery.com/how-to-choose-the-right-test-options-when-evaluating-machine-learning-algorithms/

ML Library guide:
http://daoudclarke.github.io/machine%20learning%20in%20practice/2013/10/08/machine-learning-libraries/
http://machinelearningmastery.com/java-machine-learning/
http://www.javaworld.com/article/2461485/big-data/5-ways-to-add-machine-learning-to-java-javascript-and-more.html

Java ML Libraries:
http://java-ml.sourceforge.net/
http://java-ml.sourceforge.net/content/weka-classifier
https://mahout.apache.org/
Weka:

- http://www.cs.waikato.ac.nz/ml/weka/
- http://en.wikipedia.org/wiki/Weka_(machine_learning)
- http://weka.wikispaces.com/Use+Weka+in+your+Java+code
- http://www.cs.waikato.ac.nz/ml/weka/mooc/dataminingwithweka/

https://github.com/cgearhart/students-filters
http://spark.apache.org/mllib/
MathLab: http://www.mathworks.com/machine-learning/

Tour of ML algorithms: http://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/

Machine learning guide: http://graphics.stanford.edu/~mdfisher/MachineLearning.html

Java Machine learning library - how to use weka classifiers: http://java-ml.sourceforge.net/content/weka-classifier

Use Weka in your Java code: http://weka.wikispaces.com/Use+Weka+in+your+Java+code

Weka JavaDoc: http://weka.sourceforge.net/doc.stable/index.html

Using SVM in Weka:

- http://weka.wikispaces.com/LibSVM
- http://stackoverflow.com/questions/18410900/how-to-use-svm-in-weka

Tutorials:
WEKA Data Mining Tutorial for First Time and Beginner Users: https://www.youtube.com/watch?v=m7kpIBGEdkI

Data Mining with Weka (1.6: Visualizing your data): https://www.youtube.com/watch?v=U-1sTxmHE5U

Weka Tutorial 13: Stacking Multiple Classifiers (Classification) - Boosting:
https://www.youtube.com/watch?v=Nje8mblA7bs

Improve Machine Learning Results with Boosting, Bagging and Blending Ensemble Methods in Weka:
http://machinelearningmastery.com/improve-machine-learning-results-with-boosting-bagging-and-blending-ensemble-methods-in-weka/

Weka Tutorial 35: Creating Training, Validation and Test Sets (Data Preprocessing):
https://www.youtube.com/watch?v=uiDFa7iY9yo

Data mining with WEKA, Part 3: Nearest Neighbor and server-side library:
http://www.ibm.com/developerworks/library/os-weka3/

Classification via Decision Trees in WEKA
http://facweb.cs.depaul.edu/mobasher/classes/ect584/weka/classify.html

Piazza Forum Resources:
https://piazza.com/class/hyk7vwylnyc66k?cid=17
https://piazza.com/class/hyk7vwylnyc66k?cid=52

Testing server:
https://support.cc.gatech.edu/facilities/general-access-servers