CSCI 447 Project 1: Introduction to Machine Learning using WEKA

David Rice davidrice363@gmail.com
Mason Weber mason.weber.cpe@gmail.com
Logan Esch logan.esch@msu.montana.edu

Abstract

We test the ability of ten different learning algorithms to produce classifiers for ten different multivariate data sets. The algorithms vastly exceeded our expectations for their accuracy to classify data. Between the algorithms studied we find that the dataset has a larger impact on the accuracy of the classifier than which algorithm produced it.

Contents

1 Problem Statement **3**
1.1 Hypothesis 3

2 Algorithm Descriptions **4**
2.1 Simple Nearest Neighbor 4
2.2 K-Nearest Neighbor (IBK) using k >2 4
2.3 Naïve Bayes 4
2.4 Logistic Regression 5
2.5 Deision Tree (J48) 5
2.6 Ripper (JRip) 5
2.7 Support Vector Machine(SMO) 5
2.8 Feedforward Neural Networks (MultilayerPerceptron) 6
2.9 Kernel Neural Network (RBFNetwork) 6
2.10 Ensemble (Adaboost) 6

3 Data Descriptions **6**
3.1 Adult 6
3.2 Car 7
3.3 Glass 7
3.4 Ionosphere 7
3.5 Iris 8
3.6 Liver 8
3.7 MAGIC Gamma Telescope 8
3.8 Seed 9
3.9 Vertebral 9
3.10 Wine 9

4 Experimental Approach **10**
4.1 Algorithm Tuning 10

5 Experiment Results **11**
5.1 Adult Data Set 11
5.2 Car Data Set 12
5.3 Glass Data Set 12
5.4 Ionosphere Data Set 12
5.5 Iris Data Set 13
5.6 Liver Data Set 14
5.7 MAGIC Telescope Data Set 14
5.8 Seed Data Set 14
5.9 Vertebral Data Set 15
5.10 Wine Data Set 16

6 Algorithm Behavior and Conclusions **16**

7 Summary **20**

8 Bibliography **22**

1. Problem Statement

A primary facet of machine learning is classification of data within a data set. Numerous algorithms have been devised to allow a program to learn and classify data; however there is no perfect algorithm. This is known as ”no free lunch”.

The Waikato Environment for Knowledge Acquisition (WEKA) is software that allows one to load data sets and run numerous machine learning algorithms on them while producing statistical tests and analysis.

This report experimentally investigates the performance of the 10 following machine learning algorithms within WEKA

1. Simple Nearest Neighbor (IB1)
2. K-nearest Neighbor (IBk) for k >2
3. Naïve Bayes
4. Logistic Regression
5. Decision Tree (J48)
6. Ripper (JRip)
7. Support Vector Machine (LibSVM or SMO)
8. Feedforward Neural Networks (Multilayer Perceptron)
9. Kernel Neural Network (RBFNetwork)
10. Ensemble (Adaboost)

Each algorithm is run on 10 different data sets[1] with the accuracy of the algorithms measured by the following three loss functions

1. Percent Correct
2. Root Mean Square Error
3. Area under the ROC Curve

1.1 Hypothesis

The following chart in Table 15 describes the preliminary, high-level hypothesis. The data set is as follows[2]

1. Adult	2. Car	3. Glass	4. Ionosphere
5. Iris	6. Liver	7. MAGIC Telescope	8. Seed
9. Vertebral	10. Wine		

1. From the UCI Machine Learning Repository
2. The data sets are described in section 3

Table 1: Algorithm Performance Accuracy Hypothesis

Algorithm	100% to 70%	69% to 50%	49% to 0%
Simple Nearest Neighbor	4, 7	1, 5, 6, 8, 9	2, 3
K-nearest Neighbor	1, 4, 7	5, 6, 8, 9	2, 3
Naïve Bayes	1, 6, 9	2, 3, 5, 8, 10	4, 7
Logistic Regression	1, 4, 6, 7	3, 5, 9, 10	2, 8
Decision Tree	2, 6, 7	1, 4, 5, 9	3, 8, 10
Ripper	2, 6, 7	1, 3, 4, 5, 9	8, 10
Support Vector Machine	4, 5, 7	1, 6, 8, 9	2, 3
Feedforward Neural Networks	2, 5	1, 3, 4, 8	6, 7, 9, 10
Kernel Neural Network	2, 5	1, 3, 4, 8	6, 7, 9, 10
Ensemble	2, 3, 4, 7, 10	1, 5, 6, 7, 9	None

Each number in the table above refers to the enumerated algorithms listed at the beginning of section 1.1.

2. Algorithm Descriptions

2.1 Simple Nearest Neighbor

Nearest Neighbor classifies by matching the input instance with the nearest training point. The algorithm can handle variable-sized hypothesis spaces, flexible decision boundaries, and is very efficient at learning (since it is just memorizing points). It suffers in higher dimensions since the area it needs to search is exponential in the number of features. Nearest Neighbor also does not handle noisy or irrelevant data well, since it has no way of detecting them or generalizing. The algorithm assumes that cases near each other tend to belong to the same class. (Kumar et al., 2008), (Abernethy, 2010)

2.2 K-Nearest Neighbor (IBK) using k >2

K-Nearest Neighbor is nearly identical to Simple Nearest Neighbor, except that instead of classifying based on the nearest instance it finds the k nearest instances and uses the majority class of those instances to classify the given instance. Variations exist that weight based on distance. K-Nearest Neighbor has all the advantages of Nearest Neighbor, but also can now handle some noise in the data if the majority of closest instances are correct. (Russell and Norvig)

2.3 Naïve Bayes

Naïve Bayes derives from the training data probabilities for each of the features that an instance is of a certain class. To keep the computation tractable, the algorithm assumes that all of the features are independent. This assumption allows the algorithm to handle datasets with high dimensionality and learn quickly. The assumption also causes the algorithm to perform poorly if independence of features does not hold. (har), (Russell and Norvig)

2.4 Logistic Regression

Logistic Regression attempts to fit a logistic or sigmoid curve to model the feature space using probabilistic classification given a binary classification. A sigmoid curve is good for binary classification due to how it is easy to tune and perfect the asymptotes of a sigmoid curve. This approach makes no assumptions about the distribution of classes in the feature space, is extendable to multiple classes, is quick to train and resists overfitting. Because of the shape of the curve, more confidence can be placed in the classification of inputs that fall further from the center of the curve. (Howbert, 2012)

2.5 Deision Tree (J48)

J48 works to develop a decision tree by picking the decision boundaries that result in the maximum information gain. The end result is a tree of rules for classifying instances. These trees are easy to understand and interpret. The algorithm is able to handle large datasets that can contain both categorical and numerical data, and can detect misclassified data. One drawback is that the greedy nature of the boundary picking means that the algorithm does not always produce an optimal solution. J48 attempts to minimize tree depth, which can lead to memorization of data if there is a feature with a unique value for each datum. The decision boundaries are also always perpendicular to the feature axis which for certain problems can lead to exponential complexity increases. (Bhargava et al., 2013)

2.6 Ripper (JRip)

JRip is an implementation of the RIPPER rule learning algorithm proposed by William Cohen in 1995. RIPPER is based on the IREP algorithm which works by greedily building up rules from the training set until the rule covers no negative examples. RIPPER then prunes rules based on the ratio of the difference positive to total examples (unlike IREP which prunes based on just the difference of positive to negative examples). RIPPER continues this process until the total description length of the rule is longer by an arbitrary length more than the smallest rule described thus far (also a change from IREP, which greedily stops when the error rate exceeds 50% for a rule, harming performance on smaller data sets). Finally RIPPER works on optimizing the rules it has generated by exploring pruning the generated rules to reduce the error set. (Cohen, 1995)

2.7 Support Vector Machine(SMO)

A Support Vector Machines (SVM) learns the linear boundary that maximizes the distance between classes. Focusing on the maximization of the boundary helps with generalization. SVMs can use the Kernel Trick to project data into higher dimensional space to allow linear separation when the data is not readily separable in the original space. While this technique can allow the SVM to generate a linear separator for very erratic data in the original space, it can lead to the inability to detect noise in the data. Because SVMs focus on the boundary between classes, they can often discard much of their data sets and retain only the examples that help define the boundaries. As such they have the power to model complex functions but can often avoid overfitting. (Russell and Norvig)

2.8 Feedforward Neural Networks (MultilayerPerceptron)

Feedforward Neural Networks are neural networks that can be arbitrarily large and deep. They consist of a series of nodes, each with a series of weighted inputs that feed into the nodes activation function which controls the nodes output. The nodes can be linked into layers with the outputs of the previous layer serving as inputs to the next layer, the outputs of the final layer serve as the outputs of the algorithm. Unlike many of the other algorithms we looked at, these networks can perform nonlinear regression. Feedforward Neural Networks can suffer from memorization rather than generalization of the training data if they contain too many nodes. Conversely to be used for universal approximation the network may require an infinite number of nodes. The effectiveness of the network on a particular problem can be dependent on the choice of activation function. Typical choices are 0/1 threshold or logistic functions. (Russell and Norvig)

2.9 Kernel Neural Network (RBFNetwork)

RBF Networks are another type of neural network. They consist of just three layers, an input layer, a hidden layer activated by a radial basis function (RBF) and a summation output layer. The radial function is typically Gaussian. RFB networks tend to be simpler than feedforward neural networks. Because they are only three layers RBF networks are easy to design. They also can handle noisy data and patterns that were not trained on. (Santos et al., 2013)

2.10 Ensemble (Adaboost)

Adaboost is an Ensemble algorithm that builds several models of other classifiers. It trains the classifiers by focusing on different parts of the problem. In other words, an Ensemble tries to find the class of x by combining numerous classifiers as a function of x. The algorithm will then do a weighted majority vote where the weights are based off of the performance of each classifier. The weighted majority vote works by adding all the weights associated with one class compared to another and returning whichever one is higher. To make sure an Ensemble does better than a single classifier, Adaboost attempts to have each classifier fail on different examples. If that happens, diversity is introduced that is driven by samples from data sets. Adaboost does this by still doing the weighted majority vote but also weighs the examples. The algorithm assumes that all classifiers are weak classifiers. It also relies on the diversity of the classifiers it selects; if it selects all the same type of classifier it will be limited by that classifiers biases. (Sheppard, 2015)

3. Data Descriptions

3.1 Adult

Abstract: Predict whether income exceeds $50K/yr based on census data.

- Data Set Characteristics: Multivariate
- Attribute Characteristics: Categorical, Integer
- Classes: 2

- Instances: 48842
- Attributes: 14

Sourced from (UCI, 1996)

3.2 Car

Abstract: Database used for testing constructive induction and structure discovery methods.

- Data Set Characteristics: Multivariate
- Attribute Characteristics: Categorical
- Classes: 4
- Instances: 1728
- Attributes: 6

Sourced from (UCI, 1997)

3.3 Glass

Abstract: Six types of glass; defined in terms of their oxide content

- Data Set Characteristics: Multivariate
- Attribute Characteristics: Real
- Classes: 6
- Instances: 214
- Attributes: 10

Sourced from (UCI, 1987a)

3.4 Ionosphere

Abstract: Classification of radar returns from the ionosphere

- Data Set Characteristics: Multivariate
- Attribute Characteristics: Integer, Real
- Classes: 2
- Instances: 351
- Attributes: 34

Sourced from (UCI, 1987b)

3.5 Iris

Abstract: Three classes of 50 instances each referring to a type of iris plant

- Data Set Characteristics: Multivariate
- Attribute Characteristics: Real
- Classes: 3
- Instances: 150
- Attributes: 4

Sourced from (UCI, 1988)

3.6 Liver

Abstract: Liver patient records used to divide in to groups of whether someone is a liver patient or not.

- Data Set Characteristics: Multivariate
- Attribute Characteristics: Integer, Real
- Classes: 2
- Instances: 583
- Attributes: 10

Sourced from (UCI, 2012a)

3.7 MAGIC Gamma Telescope

Abstract: Simulates registration of high energy gamma particles in an atmospheric Cherenkov telescope.

- Data Set Characteristics: Multivariate
- Attribute Characteristics: Real
- Classes: 2
- Instances: 19020
- Attributes: 11

Sourced from (UCI, 2007)

3.8 Seed

Abstract: Measurements of geometrical properties of kernels belonging to three different varieties of wheat.

- Data Set Characteristics: Multivariate
- Attribute Characteristics: Real
- Classes: 3
- Instances: 210
- Attributes: 7

Sourced from (UCI, 2012b)

3.9 Vertebral

Abstract: Values for six biomechanical features used to classify orthopaedic patients in to three or two classes.

- Data Set Characteristics: Multivariate
- Attribute Characteristics: Real
- Classes: 2 or 3
- Instances: 310
- Attributes: 6

Sourced from (UCI, 2011)

3.10 Wine

Abstract: Chemical analysis to determine the origin of wines

- Data Set Characteristics: Multivariate
- Attribute Characteristics: Integer, Real
- Classes: 3
- Instances: 178
- Attributes: 13

Sourced from (UCI, 1991)

4. Experimental Approach

Given the goal of finding which data sets perform best and worst on each algorithm, the experimental approach is a three step process

1. Convert the ten data files from the UCI Machine Learning Repository to .arff format to allow proper input in to WEKA
2. Run WEKA with the following settings:
   - Input the ten data sets set to run on the 10 algorithms.
   - Run each experiment as 10-fold cross-validation with an iteration control of 10 repetitions.
   - Tune each algorithm as discussed in section 4.1.
   - Run the program.
   - Analyze the results such that the performance of each algorithm according to the three loss function evaluation measures
3. Document and report the results in charts ranking algorithm performance on each data set organized from best to worst. Also note the distribution of error and give an analysis of why the algorithm performed the way it did.

4.1 Algorithm Tuning

Many algorithms need specific tuning to return the most reliable results. This tuning is done in WEKA under the *Algorithms* section and selecting the *Edit Selected* button. The tuning is as follows[3]

1. Simple Nearest Neighbor: No tuning available to set
2. K-nearest Neighbor: *KNN* = 3, *Cross-Validation* True, *Mean Squared* False, *Nearest Neighbor Search Algorithm* Linear NN Search, *Window Size* No limit to training instances
3. Naïve Bayes: *Kernel Estimator* Don't use, *Supervised Discretization* Don't use
4. Logistic Regression: *Maximum Iterations* No max, *Ridge Value* 1.0E-8
5. Decision Tree: *Binary Splits on Nominal Attributes* False, *Confidence Factor* 0.25, *Minimum instances per Leaf* 2, *Number of Folds* 3, *Include Subtree Raising* True, *Perform Pruning* True, *Laplace Leave Smoothening* True
6. Ripper: *Check Error Rate >= 1/2* True, *Number of Folds* 3, *Minimum Weight of Rule Instances* 2.0, *Optimization Runs* 2, *Perform Pruning* True
7. Support Vector Machine: *SVM Type* C-SVC (classification), *Coef0* 0.0, *Cost for C-SVC* 1.0, *Kernel Degree* 3, *Replace Missing Values* False, *Gamma* 1 / max_index, *Kernal Type* Radial Basis Function, *Loss Epison Value* 0.1, *Use Shrinking Hueristic* True, *Class Weights* 1

3. Only notable tunings are mentioned. Attributes such as random seed choices are left out

8. Feedforward Neural Networks: *Auto-Build* True, *Decay* False, *Hidden Layers* Attributes + Classes, *Learning Rate* 0.3, *Momentum* 0.2, *Normal to Binary Filer* true, *Normalize Attributes* true, *Normalize Numeric Class* True

9. Kernel Neural Network *Maximum Iterations for Discrete Classes* -1, *Minimum Standard Deviation* 0.1 *Number of Clusters* 2, *Ridge Value* 1.0E-8

10. Ensemble *Base Classifier* Decision Stump, *Number of Iterations* 10, *Use Resampling* False, *Weight Threshold for Pruning* 100

5. Experiment Results

Results are given by looking at each data set and ranking each algorithm based on the percentage of correctly classified data. Analysis and reasoning for the observed results is given in section 6.

5.1 Adult Data Set

We experienced difficulty with the running time of the Feedforward Neural Network and Support Vector Machine on this dataset. As such we changed the tunings on these algorithms as noted below.

Tested with Paired T-Tester (corrected). Confidence Interval of 95%. Comparing each algorithm to its percentage correct loss functions.

Table 2: Adult Data Set Algorithm Performance Metrics

Algorithm	Percent Correct, RMS, ROC	Remarks
Decision Tree	86.25, 0.33, 0.90	
Logistic Regression	85.05, 0.32, 0.90	
Support Vector Machine	84.86, 0.39, 0.75	c = 0.5
Ripper	84.50, 0.35, 0.75	
Ensemble	83.96, 0.34, 0.87	
Kernel Neural Network	83.54, 0.35, 0.88	
Naïve Bayes	83.43, 0.37, 0.89	
Feedforward Neural Networks	82.90, 0.34, 0.89	hiddenLayers = 5, trainingTime = 10
K-nearest Neighbor	81.92, 0.37, 0.82	
Simple Nearest Neighbor	79.47, 0.45, 0.72	

5.2 Car Data Set

Tested with Paired T-Tester (corrected). Confidence Interval of 95%. Comparing each algorithm to its percentage correct loss functions.

Table 3: Car Data Set Algorithm Performance Metrics

Algorithm	Percent Correct, RMS, ROC	Remarks
Feedforward Neural Networks	99.35, 0.05, 1.00	Runs slowest, high performance
K-nearest Neighbor	95.34, 0.20, 0.99	
Support Vector Machine	93.45, 0.32, 0.98	
Logistic Regression	93.28, 0.15, 1.00	
Decision Tree	92.44, 0.22, 0.99	
Kernel Neural Network	88.80, 0.20, 0.98	
Ripper	87.43, 0.33, 0.94	
Naïve Bayes	85.46, 0.23, 0.98	
Simple Nearest Neighbor	77.30, 0.34, 0.67	Runs most quickly
Ensemble	70.02, 0.33, 0.79	

5.3 Glass Data Set

Tested with Paired T-Tester (corrected). Confidence Interval of 95%. Comparing each algorithm to its percentage correct loss functions.

Table 4: Glass Data Set Algorithm Performance Metrics

Algorithm	Percent Correct, RMS, ROC	Remarks
Decision Tree	96.95, 0.10, 1.00	
Ripper	95.08, 0.10, 0.97	
Support Vector Machine	95.08, 0.32, 0.98	Very similar to Ripper
Feedforward Neural Networks	93.93, 0.09, 0.99	Runs very slowly
Logistic Regression	93.92, 0.12, 0.98	
K-nearest Neighbor	91.58, 0.15, 0.92	High std. deviation
Simple Nearest Neighbor	90.84, 0.16, 0.92	High std. deviation
Naïve Bayes	84.09, 0.20, 0.99	High std. deviation
Ensemble	81.53, 0.27, 0.69	
Kernel Neural Network	67.77, 0.12, 0.96	

5.4 Ionosphere Data Set

Tested with Paired T-Tester (corrected). Confidence Interval of 95%. Comparing each algorithm to its percentage correct loss functions.

Table 5: Ionosphere Data Set Algorithm Performance Metrics

Algorithm	Percent Correct, RMS, ROC	Remarks
Feedforward Neural Networks	92.32, 0.27, 0.92	Runs very slowly
Kernel Neural Network	91.03, 0.26, 0.95	
Support Vector Machine	90.89, 0.34, 0.95	
Decision Tree	89.46, 0.28, 0.92	
Ripper	89.04, 0.31, 0.89	
Ensemble	88.47, 0.27, 0.95	
Logistic Regression	88.05, 0.31, 0.87	
Simple Nearest Neighbor	86.77, 0.35, 0.83	
K-nearest Neighbor	86.33, 0.32, 0.90	
Naïve Bayes	82.48, 0.39, 0.94	High std. deviation

5.5 Iris Data Set

Tested with Paired T-Tester (corrected). Confidence Interval of 95%. Comparing each algorithm to its percentage correct loss functions.

Table 6: Iris Data Set Algorithm Performance Metrics

Algorithm	Percent Correct, RMS, ROC	Remarks
Logistic Regression	96.67, 0.10, 0.99	
Ensemble	96.33, 0.15, 0.97	
Support Vector Machine	96.33, 0.29, 0.98	
Feedforward Neural Networks	96.00, 0.10, 0.99	
Naïve Bayes	95.67, 0.12, 0.99	
Ripper	95.00, 0.14, 0.96	
Simple Nearest Neighbor	95.33, 0.13, 0.95	
K-nearest Neighbor	95.00, 0.13, 0.96	
Decision Tree	95.00, 0.14, 0.98	
Kernel Neural Network	94.67, 0.15, 0.99	

5.6 Liver Data Set

Tested with Paired T-Tester (corrected). Confidence Interval of 95%. Comparing each algorithm to its percentage correct loss functions.

Table 7: Liver Data Set Algorithm Performance Metrics

Algorithm	Percent Correct, RMS, ROC	Remarks
Logistic Regression	71.96, 0.42, 0.74	
Ensemble	71.36, 0.43, 0.70	Low Std. Deviation
Feedforward Neural Networks	71.31, 0.43, 0.72	Runs very slowly
Kernel Neural Network	70.55, 0.42, 0.71	
Ripper	69.38, 0.45, 0.55	
Support Vector Machine	68.52, 0.54, 0.50	
Decision Tree	68.46, 0.46, 0.55	
K-nearest Neighbor	64.53, 0.49, 0.61	
Simple Nearest Neighbor	64.44, 0.59, 0.59	
Naïve Bayes	55.79, 0.65, 0.73	Poor performance

5.7 MAGIC Telescope Data Set

Tested with Paired T-Tester (corrected). Confidence Interval of 95%. Comparing each algorithm to its percentage correct loss functions.

Table 8: MAGIC Telescope Data Set Algorithm Performance Metrics

Algorithm	Percent Correct, RMS, ROC	Remarks
Feedforward Neural Networks	85.7, 0.33, 0.91	
Decision Tree	85.13, 0.34, 0.90	
Ripper	84.72, 0.36, 0.82	
K-nearest Neighbor	83.06, 0.37, 0.86	
Simple Nearest Neighbor	80.99, 0.44, 0.78	
Support Vector Machine	79.14, 0.46, 0.75	
Logistic Regression	79.09, 0.38, 0.84	
Ensemble	78.83, 0.38, 0.85	Relatively high Std. Dev
Kernel Neural Network	78.45, 0.39, 0.83	
Naïve Bayes	72.68, 0.49, 0.76	

5.8 Seed Data Set

Tested with Paired T-Tester (corrected). Confidence Interval of 95%. Comparing each algorithm to its percentage correct loss functions.

Table 9: Seed Data Set Algorithm Performance Metrics

Algorithm	Percent Correct, RMS, ROC	Remarks
Support Vector Machine	95.48, 0.30, 0.98	
Logistic Regression	94.88, 0.14, 0.99	
Simple Nearest Neighbor	93.69, 0.18, 0.97	
Ensemble	93.45, 0.24, 0.95	
K-nearest Neighbor	92.74, 0.18, 0.99	
Feedforward Neural Networks	92.02, 0.13, 0.99	Longest running time
Decision Tree	91.43, 0.21, 0.97	High std. deviation
Naïve Bayes	90.71, 0.22, 0.99	
Kernel Neural Network	90.48, 0.20, 0.99	
Ripper	90.48, 0.23, 0.94	

5.9 Vertebral Data Set

Tested with Paired T-Tester (corrected). Confidence Interval of 95%. Comparing each algorithm to its percentage correct loss functions.

Table 10: Vertebral Data Set Algorithm Performance Metrics

Algorithm	Percent Correct, RMS, ROC	Remarks
Logistic Regression	85.00, 0.31, 0.93	High std. deviation
Support Vector Machine	84.92, 0.46, 0.71	High std. deviation
Ripper	82.02, 0.38, 0.79	High std. deviation
Simple Nearest Neighbor	81.55, 0.42, 0.81	High std. deviation
Decision Tree	81.37, 0.36, 0.88	High std. deviation
Feedforward Neural Networks	80.48, 0.32, 0.93	Longest running time
Kernel Neural Network	79.52, 0.36, 0.71	High std. deviation
Ensemble	78.63, 0.36, 0.89	
Naïve Bayes	78.06, 0.42, 0.88	High std. deviation
K-nearest Neighbor	77.98, 0.36, 0.84	High std. deviation

5.10 Wine Data Set

Tested with Paired T-Tester (corrected). Confidence Interval of 95%. Comparing each algorithm to its percentage correct loss functions.

Table 11: Wine Data Set Algorithm Performance Metrics

Algorithm	Percent Correct, RMS, ROC	Remarks
Support Vector Machine	98.76, 0.28, 0.99	
Feedforward Neural Networks	98.25, 0.08, 1.00	Very long running time
Kernel Neural Network	97.86, 0.05, 1.00	
Logistic Regression	97.69, 0.07, 1.00	Very long running time
Naïve Bayes	97.46, 0.08, 1.00	
K-nearest Neighbor	95.85, 0.13, 0.99	
Simple Nearest Neighbor	95.12, 0.14, 0.99	
Ripper	93.14, 0.16, 0.97	
Decision Tree	93.09, 0.18, 0.95	
Ensemble	89.53, 0.22, 0.98	

6. Algorithm Behavior and Conclusions

Figure 1 shows a beanplot for each algorithm of that algorithm's percent correct on each dataset (listed alphabetically). Initial inspection gives the intuition that all the algorithms performed well on some datasets and poorly on others. Their means are all visual similar, the Feedforward Neural Network algorithm has the highest mean at 89.226% correct, and the Naïve Bayes algorithm has the lowest at 82.583% correct.

Figure 2 shows a beanplot for each dataset of the algorithms accuracy on that data set. It is apparent from the figure that the datasets vary greatly. Liver was the toughest dataset for the algorithms, with a mean score of 67.63% correct. Iris, Seed and Wine were the easiest, with means of 95.6, 92.54 and 95.68 percent correct respectively. We can also see that the Car and Glass datasets had the largest spread, and thus the performance on those datasets will have the biggest impact on the overall algorithm ranking.

Running a Two Way Analysis of Variance to test for interaction between the dataset and the algorithm yields the Table 12. We can see that there is not sufficient evidence to conclude that there is a significant interaction between the dataset and the algorithm it is run on ($p = 0.1615$).

Table 12: Analysis of Variance Interaction Model

Predictor	Sum Sq	Df	F value	Pr(>F)
Dataset	6585.3	9	40.0485	2.696e-14
Algorithm	1022.1	9	6.2159	6.363e-05
Dataset:Algorithm	2043.3	81	1.3807	0.1615
Residuals	548.1	30		

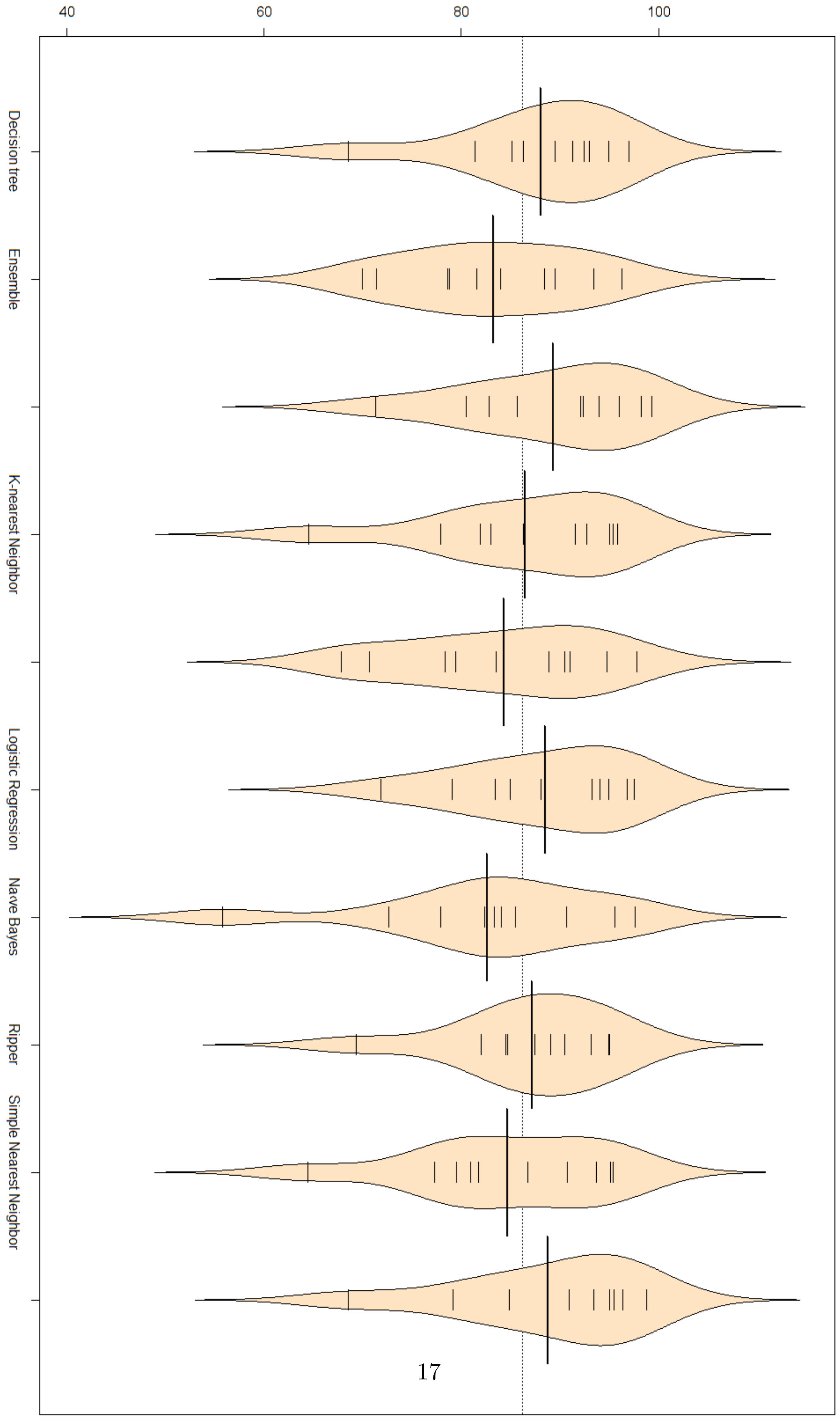

Figure 1: Comparison of Percent Correct Across Algorithms

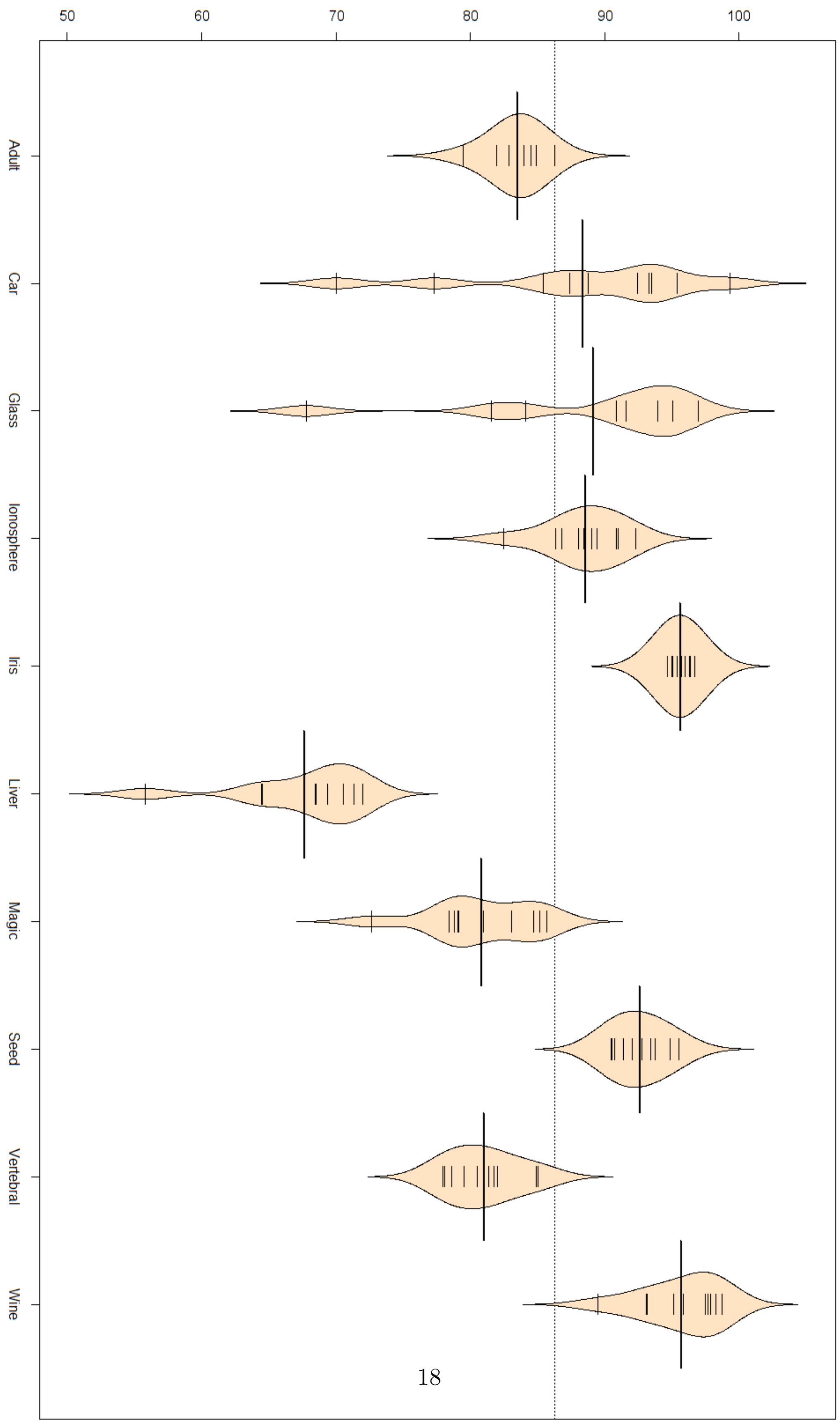

Figure 2: Comparison of Percent Correct Across Datasets

Failing to find an interaction, we test for the individual affects using an additive model in Table 13. We find strong evidence that both the dataset the algorithm was run on as well as the algorithm itself are significant (p <2.2e-16 and p=1.724e-05 respectively). Thus, contrary to our initial inspection above, both the dataset that the algorithm is performing on and the algorithm itself are significant to the accuracy of the classifier.

Table 13: Analysis of Variance Additive Model

Predictor	Sum Sq	Df	F value	Pr(>F)
Dataset	6585.3	9	31.3412	<2.2e-16
Algorithm	1022.1	9	4.8645	1.724e-05
Residuals	2591.5	111		

Examining the estimated coefficients in Table 14 we see that almost all of the data sets are found to be significantly different from the base dataset Adult when we hold the algorithm at it's mean. We also see that almost all the algorithms were found to be insignificant compared to the base algorithm Decision Tree when we hold the dataset at it's mean. The only algorithms found to be significantly different were Naïve Bayes and Simple Nearest Neighbor, which were found to be less accurate (p = 0.00111 and p = 0.02066 respectively). There is insufficient evidence to conclude that the other algorithms have different performance than the base algorithm decision tree when we hold the dataset at its mean.

Table 14: Coefficient Estimates

Predictor	Estimate	Std. Error	t value	Pr(¿—t—)
(Intercept)	85.80615	1.98770	43.169	¡ 2e-16 ***
DatasetCar	4.85000	1.87135	2.592	0.01083 *
DatasetGlass	5.68100	1.87135	3.036	0.00299 **
DatasetIonosphere	4.98750	1.87135	2.665	0.00884 **
DatasetIris	12.16300	2.16085	5.629	1.38e-07 ***
DatasetLiver	-15.80700	2.16085	-7.315	4.28e-11 ***
DatasetMagic	-2.65800	2.16085	-1.230	0.22127
DatasetSeed	9.09900	2.16085	4.211	5.19e-05 ***
DatasetVertebral	-2.46200	2.16085	-1.139	0.25700
DatasetWine	12.23800	2.16085	5.664	1.18e-07 ***
AlgorithmEnsemble	-7.55769	1.89519	-3.988	0.00012 ***
AlgorithmFeedforward Neural Network	1.50154	1.89519	0.792	0.42988
AlgorithmK-nearest Neighbor	-1.70231	1.89519	-0.898	0.37101
AlgorithmKernel Neural Network	-3.21769	1.89519	-1.698	0.09234 .
AlgorithmLogistic Regression	0.02923	1.89519	0.015	0.98772
AlgorithmNave Bayes	-6.34615	1.89519	-3.349	0.00111 **
AlgorithmRipper	-1.25615	1.89519	-0.663	0.50883
AlgorithmSimple Nearest Neighbor	-4.44923	1.89519	-2.348	0.02066 *
AlgorithmSupport Vector Machine	-0.69308	1.89519	-0.366	0.71528

Table 16 shows our results in the same format as our hypothesis was presented back in Table 15. Both Table 15 and the table enumerating the datasets has been copied below for the readers convenience. On the whole the algorithms vastly outperformed out expectations. None of them had an accuracy less than 50%.

1. Adult	2. Car	3. Glass	4. Ionosphere
5. Iris	6. Liver	7. MAGIC Telescope	8. Seed
9. Vertebral	10. Wine		

Table 15: Algorithm Performance Accuracy Hypothesis

Algorithm	100% to 70%	69% to 50%	49% to 0%
Simple Nearest Neighbor	4, 7	1, 5, 6, 8, 9	2, 3
K-nearest Neighbor	1, 4, 7	5, 6, 8, 9	2, 3
Naïve Bayes	1, 6, 9	2, 3, 5, 8, 10	4, 7
Logistic Regression	1, 4, 6, 7	3, 5, 9, 10	2, 8
Decision Tree	2, 6, 7	1, 4, 5, 9	3, 8, 10
Ripper	2, 6, 7	1, 3, 4, 5, 9	8, 10
Support Vector Machine	4, 5, 7	1, 6, 8, 9	2, 3
Feedforward Neural Networks	2, 5	1, 3, 4, 8	6, 7, 9, 10
Kernel Neural Network	2, 5	1, 3, 4, 8	6, 7, 9, 10
Ensemble	2, 3, 4, 7, 10	1, 5, 6, 7, 9	None

Table 16: Algorithm Performance Accuracy Results

Algorithm	100% to 70%	69% to 50%	49% to 0%
Simple Nearest Neighbor	1, 2, 3, 4, 5, 7, 8, 9	6	None
K-nearest Neighbor	1, 2, 3, 4, 5, 7, 8, 9	6	None
Naïve Bayes	1, 2, 3, 4, 5, 7, 8, 9	6	None
Logistic Regression	1, 2, 3, 4, 5, 6, 7, 8, 9	None	None
Decision Tree	1, 2, 3, 4, 5, 6, 7, 8, 9	None	None
Ripper	1, 2, 3, 4, 5, 7, 8, 9	6	None
Support Vector Machine	1, 2, 3, 4, 5, 7, 8, 9	6	None
Feedforward Neural Networks	1, 2, 3, 4, 5, 6, 7, 8, 9	None	None
Kernel Neural Network	1, 2, 4, 5, 6, 7, 8, 9	3	None
Ensemble	1, 2, 3, 4, 5, 6, 7, 8, 9	None	None

7. Summary

Upon examining algorithm performance over the datasets and returning to our hypothesis, one thing is clear we underestimated the performance on unrelated problems when defining the accuracy levels. If the performance scale shifts to 100 to 90, 89 to 80, and below 80, the predictions get slightly better.

We did not expect such a large difference between results from the Simple Nearest Neighbor and K-nearest Neighbor. For most datasets, the algorithms performed comparably, but on sets 2 and 9 there is a dramatic difference in performance. The Ensemble algorithm was the least surprising of the ten. We expected it perform do well on some sets and average on others. With the concept of an Ensemble being a majority vote of above-average classifiers, it was reasonable and correct to hypothesize that it would be consistent but never amazing.

We also did well if predicting some of the datasets that the Feedforward Neural Network would excel in. While also predicting that it would do poorly on sets it did well on, the likeness of the classes to a function for it to approximate is difficult to intuit.

The most surprising result was the overall performance of dataset 6, which we predicted most classifiers would to well on. When one problem gives trouble to all of the algorithms, it suggests complexities under the surface of the solution space, data quality, or approach.

Overall, we expected that some algorithms would outperform others on different fields of data. We saw just that as a demonstration of the No Free Lunch Theorem. We did not expect the average performance to be as high as it was.

8. Bibliography

References

(????), "Cs181 lecture 18: Naive bayes and em."

Abernethy, Michael (2010), "Data mining with weka, part 3: Nearest neighbor and server-side library." URL `http://www.ibm.com/developerworks/library/os-weka3/`.

Bhargava, Neeraj, Girja Sharma, Ritu Bhargava, and Manish Mathuria (2013), "Decision tree analysis on j48 algorithm for data mining." URL `http://www.academia.edu/4375403/Decision_Tree_Analysis_on_J48_Algorithm_for_Data_Mining`.

Cohen, William W. (1995), "Fast effective rule induction." URL `http://www.cs.utsa.edu/ bylander/cs6243/cohen95ripper.pdf`.

Howbert, Jeff (2012), "Machine learning logistic regression." URL `http://courses.washington.edu/css490/2012.Winter/lecture_slides/05b_logistic_regression.pd`

Kumar, Neeraj, Li Zhang, and Shree Nayar (2008), "What is a good nearest neighbors algorithm for finding similar patches in images?" URL `http://www1.cs.columbia.edu/CAVE/publications/pdfs/Kumar_ECCV08_2.pdf`.

Russell, Stuart and Peter Norvig (????), *Artificial Intelligence A Modern Approach*.

Santos, Rejane B., Markus Rupp, Santiago J. Bonzi, and Ana Maria F. Fileti (2013), "Comparison between multilayer feedforward neural networks and a radial basis function network to detect and locate leaks in pipelines transporting gas." URL `http://www.aidic.it/cet/13/32/230.pdf`.

Sheppard, John (2015). University Lecture.

UCI (1987a), "UCI Machine Learning Repository glass identification data set." URL `https://archive.ics.uci.edu/ml/datasets/Glass+Identification`.

UCI (1987b), "UCI Machine Learning Repository ionosphere data set." URL `https://archive.ics.uci.edu/ml/datasets/Ionosphere`.

UCI (1988), "UCI Machine Learning Repository iris data set." URL `https://archive.ics.uci.edu/ml/datasets/Iris`.

UCI (1991), "UCI Machine Learning Repository wine data set." URL `https://archive.ics.uci.edu/ml/datasets/Wine`.

UCI (1996), "UCI Machine Learning Repository adult data set." URL `https://archive.ics.uci.edu/ml/datasets/Adult`.

UCI (1997), "UCI Machine Learning Repository car evaluation data set." URL `https://archive.ics.uci.edu/ml/datasets/Car+Evaluation`.

UCI (2007), “UCI Machine Learning Repository magic gamma telescope data set.” URL `https://archive.ics.uci.edu/ml/datasets/MAGIC+Gamma+Telescope`.

UCI (2011), “UCI Machine Learning Repository vertebral column data set.” URL `https://archive.ics.uci.edu/ml/datasets/Vertebral+Column`.

UCI (2012a), “UCI Machine Learning Repository ilpd data set.” URL `https://archive.ics.uci.edu/ml/datasets/ILPD+(Indian+Liver+Patient+Dataset)`.

UCI (2012b), “UCI Machine Learning Repository seeds data set.” URL `https://archive.ics.uci.edu/ml/datasets/Seeds`.