

## Chapter 23

# How To Work Through a Multiclass Classification Project

The Weka machine learning workbench is so easy to use that working through a machine learning project can be a lot of fun. In this section you will complete your first machine learning project using Weka, end-to-end. This gentle introduction to working through a project will tie together the key steps you need to complete when working through machine learning project in Weka. After completing this project, you will know:

- How to analyze a dataset and hypothesize data preparation and modeling algorithms that could be used.
- How to spot-check a suite of standard machine learning algorithms on a problem
- How to present final results.

Let's get started.

### 23.1 Tutorial Overview

This tutorial will gently walk you through the key steps required to complete a machine learning project. We will work through the following process:

- Load the dataset.
- Analyze the dataset.
- Evaluate algorithms.
- Present results.

You can use this as a template for the minimum steps in the process to work through your own machine learning project using Weka.

## 23.2 Load Dataset

In this tutorial, we will use the Iris Flowers Classification dataset. Each instance in the iris dataset describes measurements of iris flowers and the task is to predict which species of 3 iris flower the observation belongs. You can learn more about this dataset in Section [8.3.1](#).

- 1. Open the *Weka GUI Chooser*.



Figure 23.1: Weka GUI Chooser.

- 2. Click the *Explorer* button to open the *Weka Explorer*.
- 3. Click the *Open file...* button, navigate to the `data/` directory and select `iris.arff`. Click the *Open button*.

The dataset is now loaded into Weka.

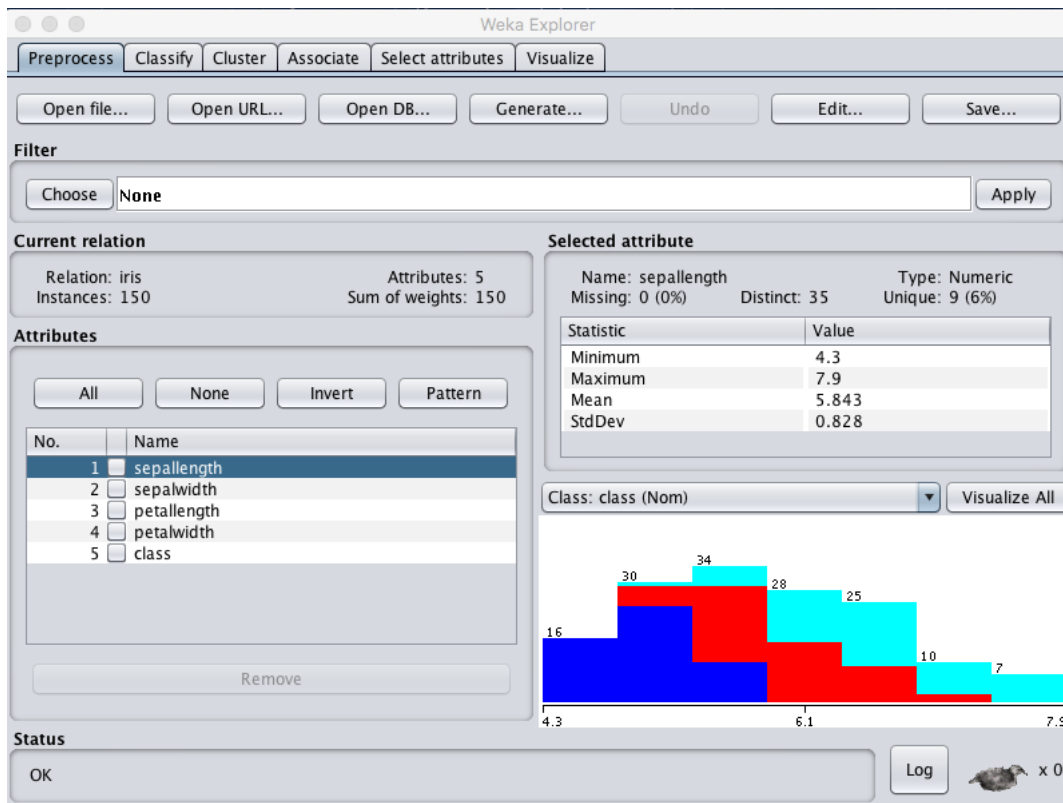


Figure 23.2: Weka Load Iris Flowers Dataset.

## 23.3 Analyze the Dataset

It is important to review your data before you start modeling. Reviewing the distribution of each attribute and the interactions between attributes may shed light on specific data transforms and specific modeling techniques that we could use.

### 23.3.1 Summary Statistics

Review the details about the dataset in the *Current relation* pane. We can notice a few things:

- The dataset is called iris.
- There are 150 instances. If we use 10-fold cross validation later to evaluate the algorithms, then each fold will be comprised of 15 instances, which is quite small. We may want to think about using 5-folds of 30 instances instead.
- There are 5 attributes, 4 inputs and 1 output variable.

There are a small number of attributes and we could investigate further using feature selection methods.

- Click on each attribute in the *Attributes* pane and review the summary statistics in the *Selected attribute* pane.

We can notice a few facts about our data:

- There are no missing values for any of the attributes.
- All inputs are numeric and have values in the same range between about 0 and about 8.
- The last attribute is the output variable called class, it is nominal and has three values.
- The classes are balanced, meaning that there is an equal number of instances in each class. If they were not balanced we may want to think about balancing them.

We may see some benefit from either normalizing or standardizing the data.

### 23.3.2 Attribute Distributions

- Click the *Visualize All* button and let's review the graphical distribution of each attribute.

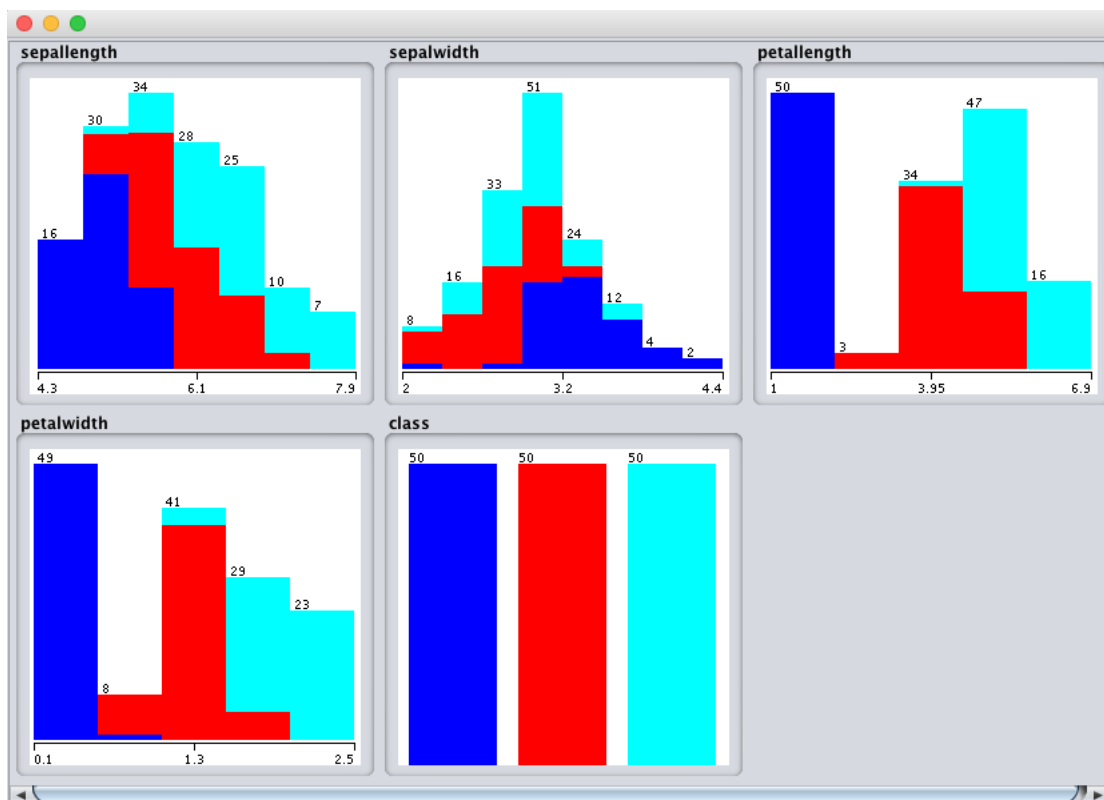


Figure 23.3: Weka Univariate Attribute Distribution Plots.

We can notice a few things about the shape of the data:

- We can see overlap but differing distributions for each of the class values on each of the attributes. This is a good sign as we can probably separate the classes.
- It looks like *sepalwidth* has a Gaussian-like distribution. If we had a lot more data, perhaps it would be even more Gaussian.

- It looks like the other 3 input attributes have nearly-Gaussian distributions with a skew or a large number of observations at the low end of the distribution. Again, it makes me think that the data may be Gaussian if we had an order of magnitude more examples.
- We also get a visual indication that the classes are balanced.

### 23.3.3 Attribute Interactions

- Click the *Visualize* tab and let's review some interactions between the attributes.
- Increase the window size so all plots are visible.
- Increase the *PointSize* to 3 to make the dots easier to see.
- Click the *Update* button to apply the changes.

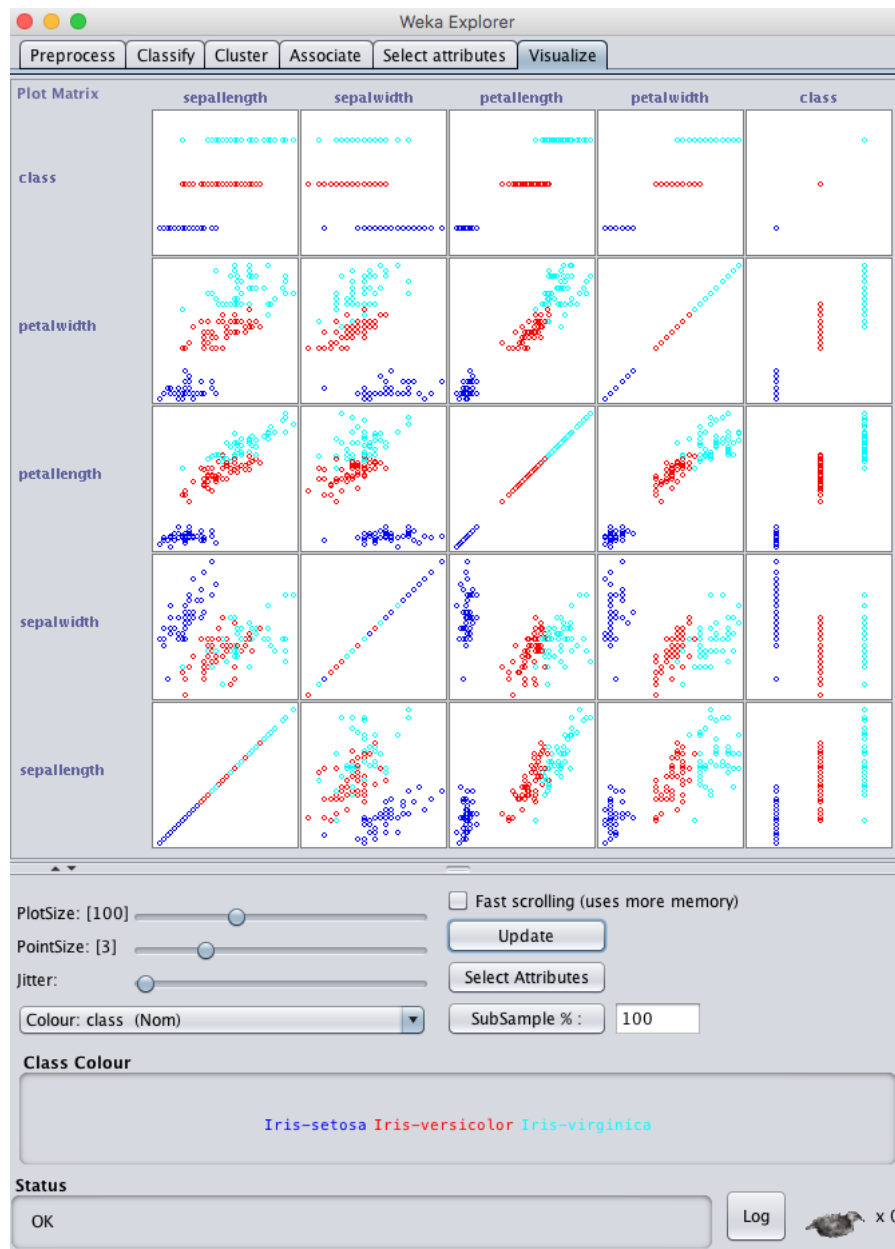


Figure 23.4: Weka Attribute Scatter Plot Matrix.

Looking across the graphs for the input variables, we can see good separation between the classes on the scatter plots. For example, *petalwidth* versus *sepalength* and *petalwidth* versus *sepalwidth* are good examples. This suggest that linear methods and maybe decision trees and instance-based methods may do well on this problem. It also suggest that we probably do not need to spend too much time tuning or using advanced modeling techniques and ensembles. It may be a straightforward modeling problem.

## 23.4 Evaluate Algorithms

Let's design a small experiment to evaluate a suite of standard classification algorithms on the problem.

- 1. Close the *Weka Explorer*.
- 2. Click the *Experimenter* button on the *Weka GUI Chooser* to launch the *Weka Experiment Environment*.

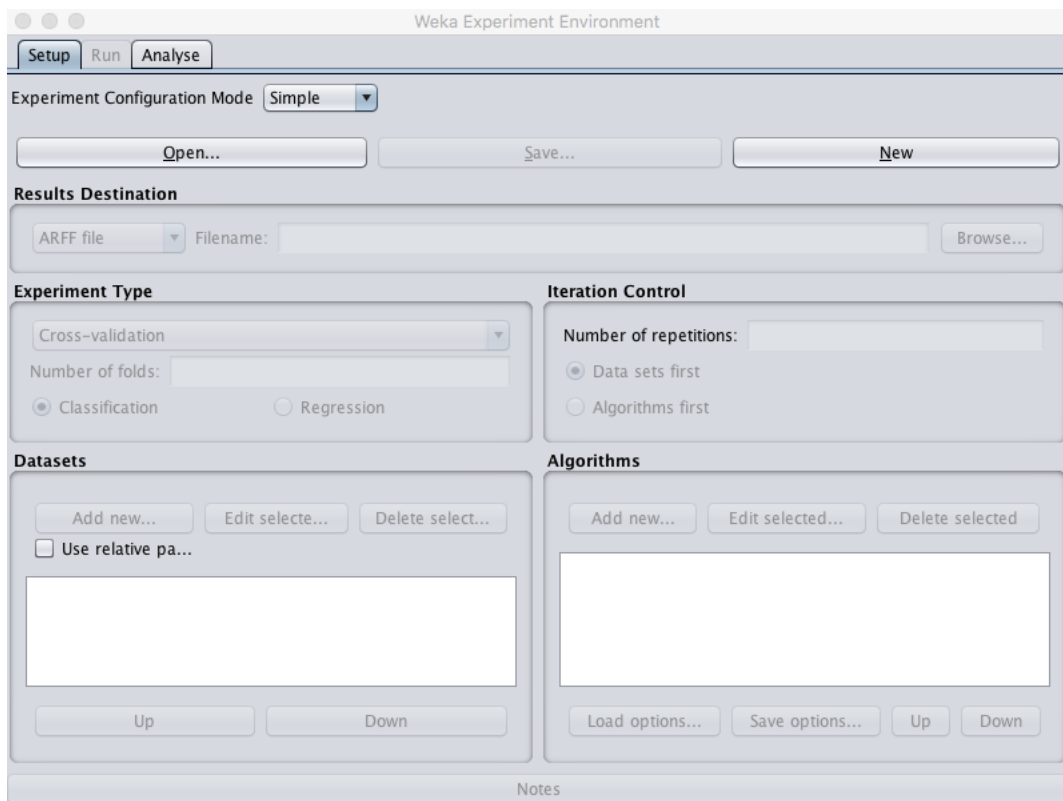


Figure 23.5: Weka Experiment Environment.

- 3. Click *New* to start a new experiment.
- 4. In the *Experiment Type* pane change the *Number of folds* from 10 to 5.
- 5. In the *Datasets* pane click *Add new...* and select `data/iris.arff` in your Weka installation directory.
- 6. In the *Algorithms* pane click *Add new...* and add the following 8 multiclass classification algorithms:
  - `rules.ZeroR`
  - `bayes.NaiveBayes`
  - `functions.Logistic`
  - `functions.SMO`
  - `lazy.IBk`
  - `rules.PART`
  - `trees.REPTree`

– *trees.J48*

- 7. Select *IBk* in the list of algorithms and click the *Edit selected...* button.
- 8. Change *KNN* from 1 to 3 and click the *OK* button to save the settings.

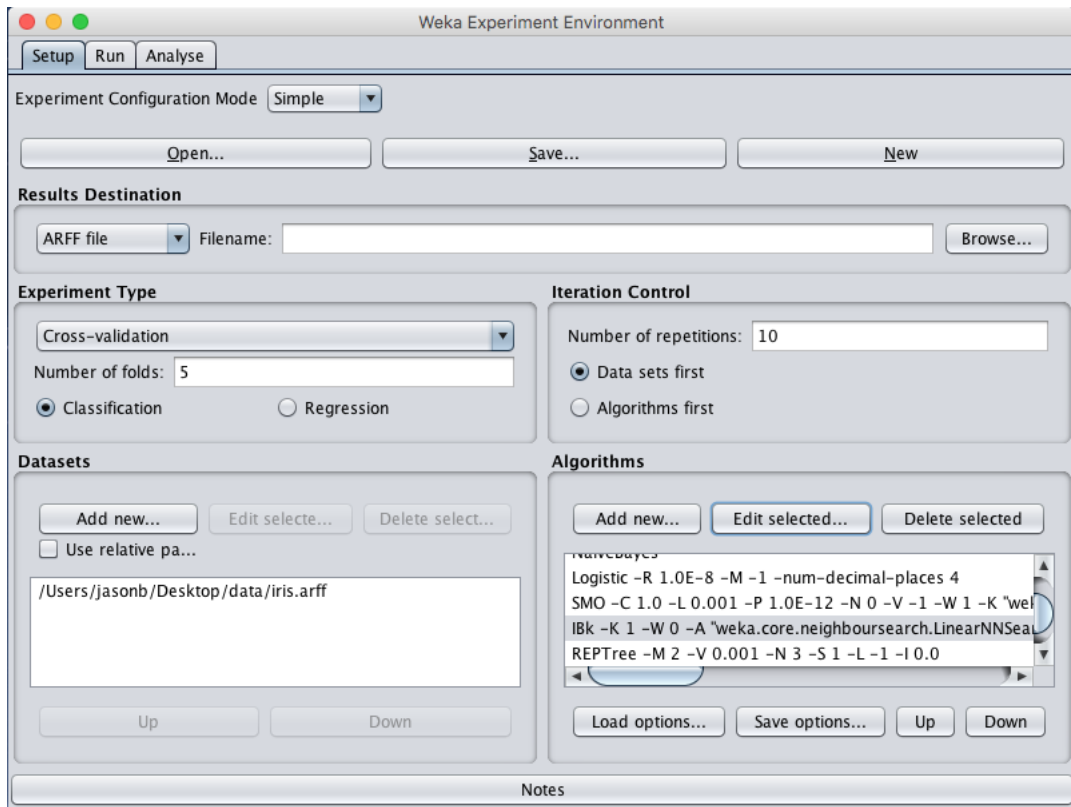


Figure 23.6: Weka Designed Algorithm Comparison Experiment.

- 9. Click on *Run* tab and click the *Start* button to run the experiment. The experiment should complete in just a few seconds.



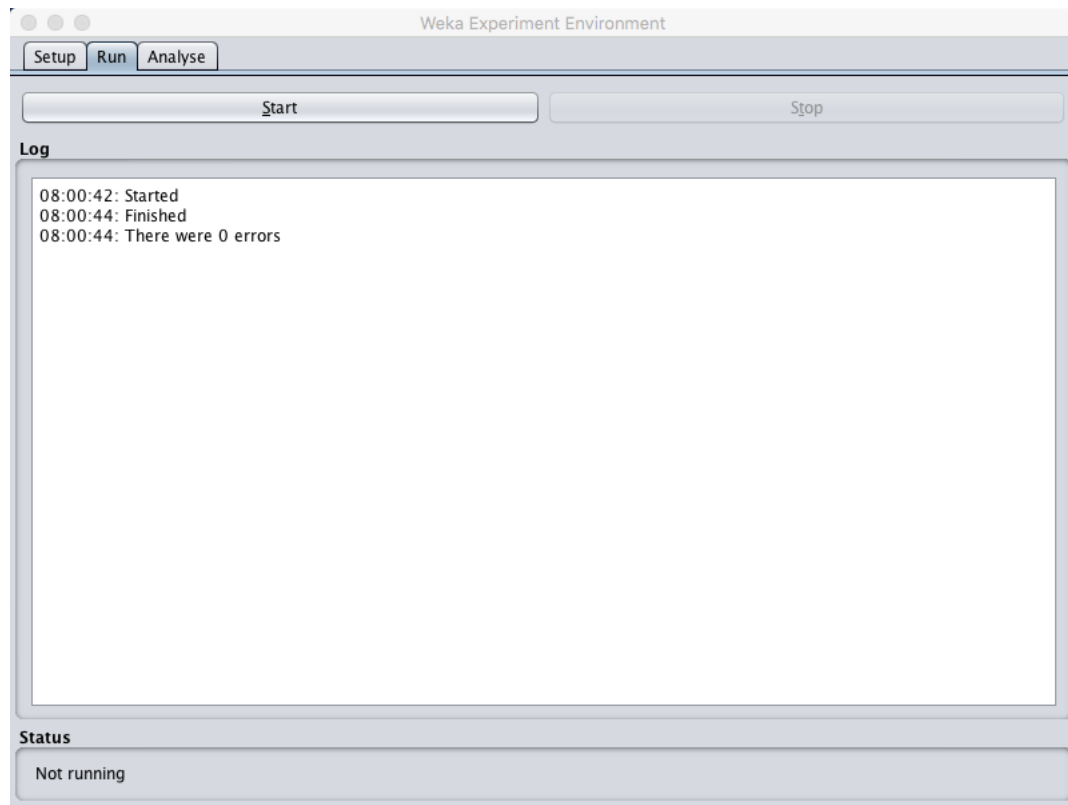


Figure 23.7: Weka Execute Weka Algorithm Comparison Experiment.

- 10. Click on *Analyse* tab. Click the *Experiment* button to load the results from the experiment.

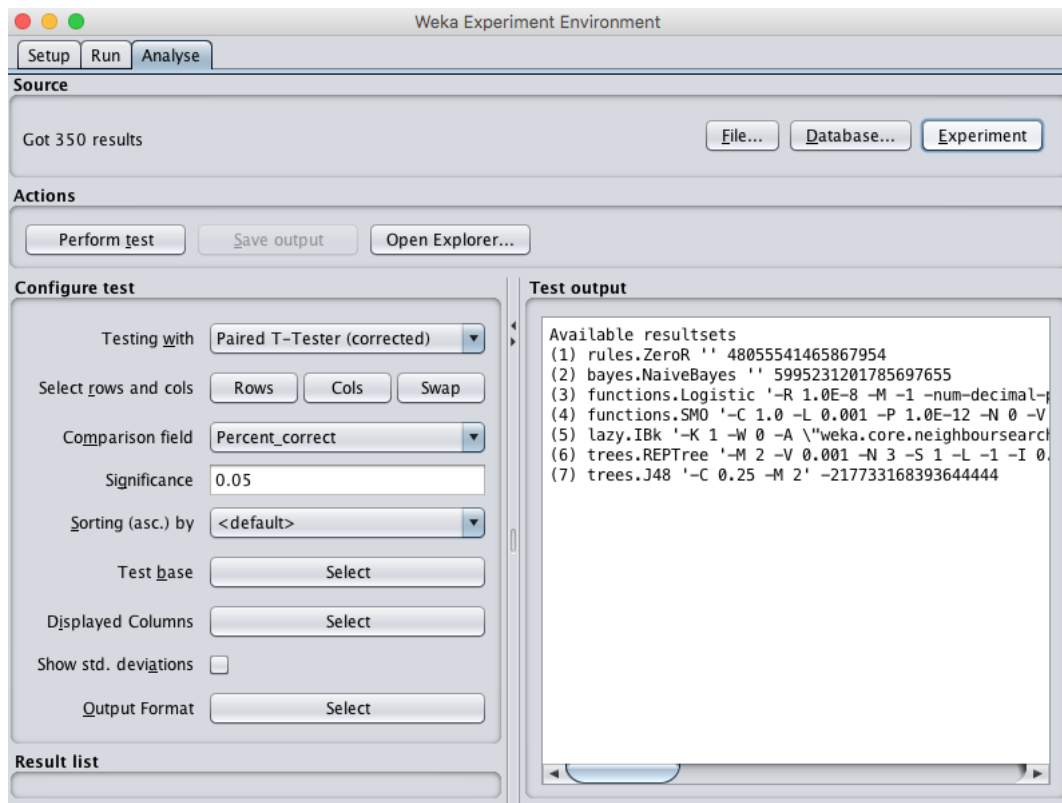


Figure 23.8: Weka Load Algorithm Comparison Experiment Results.

- 11. Click the *Perform test* button to perform a pairwise test comparing all of the results to the results for *ZeroR*.

Dataset	(1) rules.Ze	(2) bayes	(3) funct	(4) funct	(5) lazy.	(6) trees	(7) trees
iris	(50) 33.33	95.47 v	96.33 v	96.33 v	95.20 v	94.27 v	94.53 v
	(v/ /*)	(1/0/0)	(1/0/0)	(1/0/0)	(1/0/0)	(1/0/0)	(1/0/0)

Key:  
 (1) rules.ZeroR  
 (2) bayes.NaiveBayes  
 (3) functions.Logistic  
 (4) functions.SMO  
 (5) lazy.IBk  
 (6) trees.REPTree  
 (7) trees.J48

Listing 23.1: Results from Algorithm Comparison Experiment.

We can see that all of the models have skill. Each model has a score that is better than *ZeroR* and the difference is statistically significant. The results suggest both Logistic Regression and SVM achieved the highest accuracy. If we were to pick between the two, we would choose Logistic Regression if for no other reason that it is a much simpler model. Let's compare all of the results to the Logistic Regression results as the test base.

- 12. Click *Select* for the *Test base*, select *functions.Logistic* and click the *Select* button to choose the new test base. Click the *Perform test* button again to perform the new analysis.

Dataset	(3) function	(1) rules	(2) bayes	(4) funct	(5) lazy.	(6) trees	(7) trees
iris	(50) 96.33	33.33 *	95.47	96.33	95.20	94.27	94.53
	(v/ /*)	(0/0/1)	(0/1/0)	(0/1/0)	(0/1/0)	(0/1/0)	(0/1/0)

Key:  
 (1) rules.ZeroR  
 (2) bayes.NaiveBayes  
 (3) functions.Logistic  
 (4) functions.SMO  
 (5) lazy.IBk  
 (6) trees.REPTree  
 (7) trees.J48

Listing 23.2: Results from Algorithm Comparison Experiment With A New Base.

We now see a very different story. Although the results for Logistic look better, the analysis suggests that the difference between these results and the results from all of the other algorithms are not statistically significant. From here we could choose an algorithm based on other criteria, like understandability or complexity. From this perspective Logistic Regression and Naive Bayes are good candidates. We could also seek to further improve the results of one or more of these algorithms and see if we can achieve a significant improvement. If we change the *Significance* to less constraining values of 0.50, we can see that the tree and KNN algorithms start to drop away. This suggests we could spend more time on the remaining methods. Change *significance* back to 0.05. Let's choose to stick with Logistic Regression. We can collect some numbers we can use to describe the performance of the model on unseen data.

- 13. Check *Show std. deviations* to show standard deviations of accuracy scores.
- 14. Click the *Select* button for *Displayed Columns* and choose *functions.Logistic*, click *Select* to accept the selection. This will only show the results for the Logistic Regression algorithm.
- 15. Click *Perform test* to rerun the analysis.

We now have a final result we can use to describe our model.

Dataset	(3) functions.Logist
iris	(50) 96.33(3.38)
	(v/ /*)

Key:  
 (3) functions.Logistic

Listing 23.3: Final Algorithm Performance Results.

We can see that the estimated accuracy of the model on unseen data is 96.33% with a standard deviation of 3.38%.

## 23.5 Finalize Model and Present Results

We can create a final version of our model trained on all of the training data and save it to file.

- 1. Close the *Weka Experiment Environment*.
- 2. Open the *Weka Explorer* and load the `data/iris.arff` dataset.
- 3. Click on the *Classify* tab.
- 4. Select the *functions.Logistic* algorithm.
- 5. Change the *Test options* from *Cross Validation* to *Use training set*.
- 6. Click the *Start* button to create the final model.

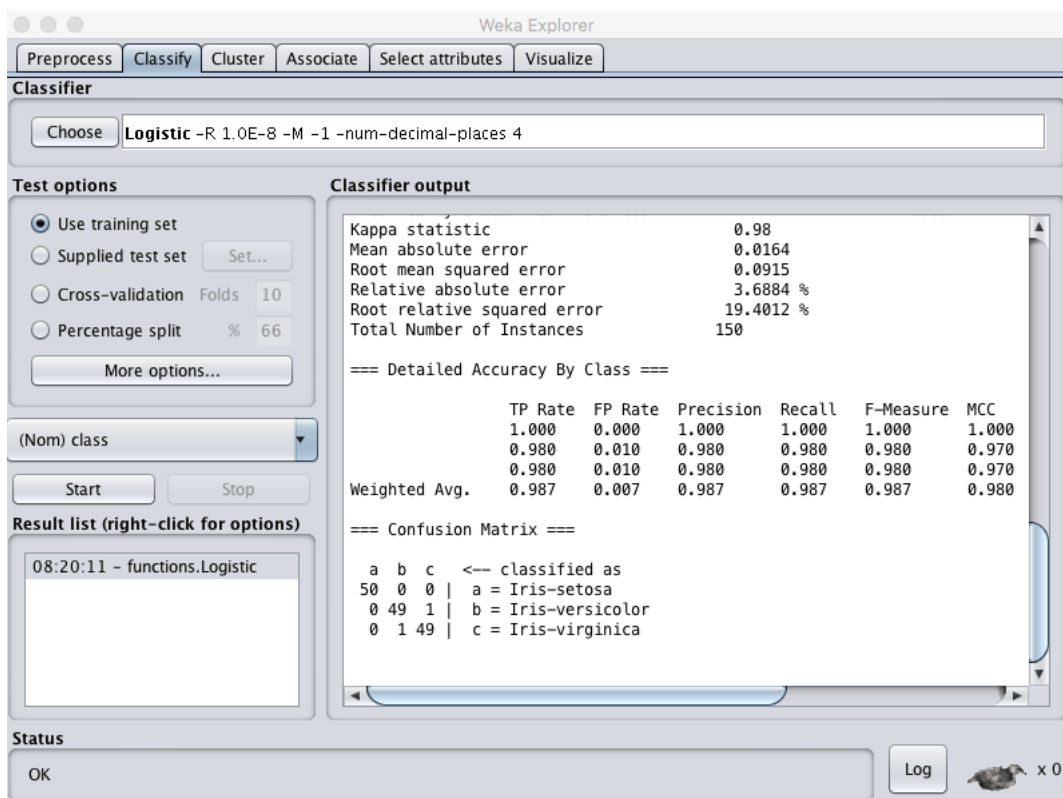


Figure 23.9: Weka Train Finalized Model on Entire Training Dataset.

- 7. Right click on the result item in the *Result list* and select *Save model*. Select a suitable location and type in a suitable name, such as *iris-logistic* for your model.

This model can then be loaded at a later time and used to make predictions on new flower measurements. We can use the mean and standard deviation of the model accuracy collected in the last section to help quantify the expected variability in the estimated accuracy of the model on unseen data. For example, we know that 95% of model accuracies will fall within two standard deviations of the mean model accuracy. Or, restated in a way we can explain to other people, we can generally expect that the performance of the model on unseen data will be 96.33% plus or minus  $2 \times 3.38$  or 6.76, or between 87.57% and 100% accurate.

## 23.6 Summary

In this project you completed your first machine learning project end-to-end using the Weka machine learning workbench. Specifically, you learned:

- How to analyze your dataset and suggest at specific data transform and modeling techniques that may be useful.
- How to spot-check a suite of algorithms on the problem and analyze their results.
- How to finalize the model for making predictions on new data and presenting the estimated accuracy of the model on unseen data.

### 23.6.1 Next

You need to practice applied machine learning to get better. In the next project you will work through a binary classification problem and investigate using multiple views of the dataset in an effort to improve performance.

# Chapter 24

## How To Work Through a Binary Classification Project

The fastest way to get good at applied machine learning is to practice on end-to-end projects. In this project you will discover how to work through a binary classification problem in Weka, end-to-end. After completing this project you will know:

- How to load a dataset and analyze the loaded data.
- How to create multiple different transformed views of the data and evaluate a suite of algorithms on each.
- How to finalize and present the results of a model for making predictions on new data.

Let's get started.

### 24.1 Tutorial Overview

This tutorial will walk you through the key steps required to complete a machine learning project. We will work through the following process:

1. Load the dataset.
2. Analyze the dataset.
3. Prepare views of the dataset.
4. Evaluate algorithms.
5. Finalize model and present results.

### 24.2 Load the Dataset

The problem used in this tutorial is the Pima Indians Onset of Diabetes dataset. You can learn more about this dataset in [Section 8.2.1](#).

- 1. Open the *Weka GUI Chooser*.
- 2. Click the *Explorer* button to open the *Weka Explorer*.
- 3. Click the *Open file...* button, navigate to the `data/` directory and select `diabetes.arff`. Click the *Open* button.

The dataset is now loaded into Weka.

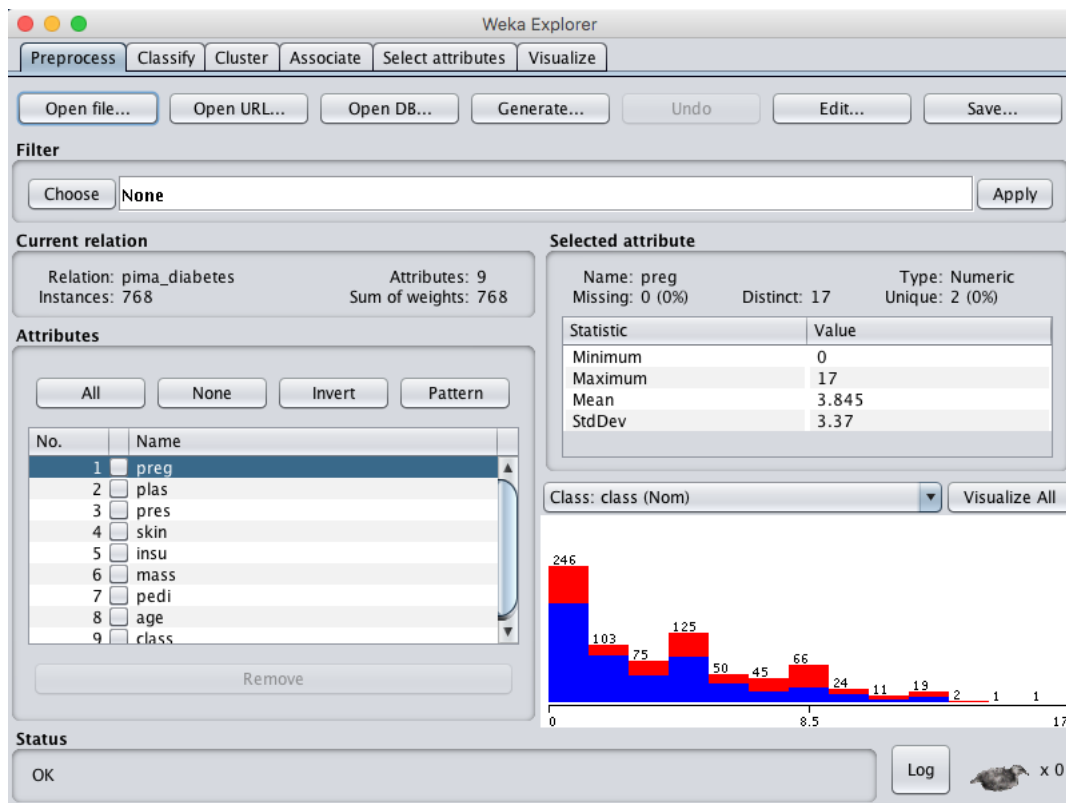


Figure 24.1: Weka Load Pima Indians Onset of Diabetes Dataset.

## 24.3 Analyze the Dataset

It is important to review your data before you start modeling. Reviewing the distribution of each attribute and the interactions between attributes may shed light on specific data transforms and specific modeling techniques that we could use.

### 24.3.1 Summary Statistics

Review the details about the dataset in the *Current relation* pane. We can notice a few things:

- The dataset has the name *pima\_diabetes*.
- There are 768 instances in the dataset. If we evaluate models using 10-fold cross validation then each fold will have about 76 instances, which is fine.

- There are 9 attributes, 8 input and one output attributes.
- Click on each attribute in the *Attributes* pane and review the summary statistics in the *Selected attribute* pane.

We can notice a few facts about our data:

- The input attributes are all numerical and have differing scales. We may see some benefit from either normalizing or standardizing the data.
- There are no missing values marked.
- There are values for some attributes that do not seem sensible, specifically: **plas**, **pres**, **skin**, **insu**, and **mass** have values of 0. These are probably missing data that could be marked.
- The class attribute is nominal and has two output values meaning that this is a two-class problem.
- The class attribute is unbalanced, 1 **positive** outcome to 1.8 **negative** outcomes, nearly double the number of negative cases. We may benefit from balancing the class values.

### 24.3.2 Attribute Distributions

Click the *Visualize All* button and let's review the graphical distribution of each attribute.

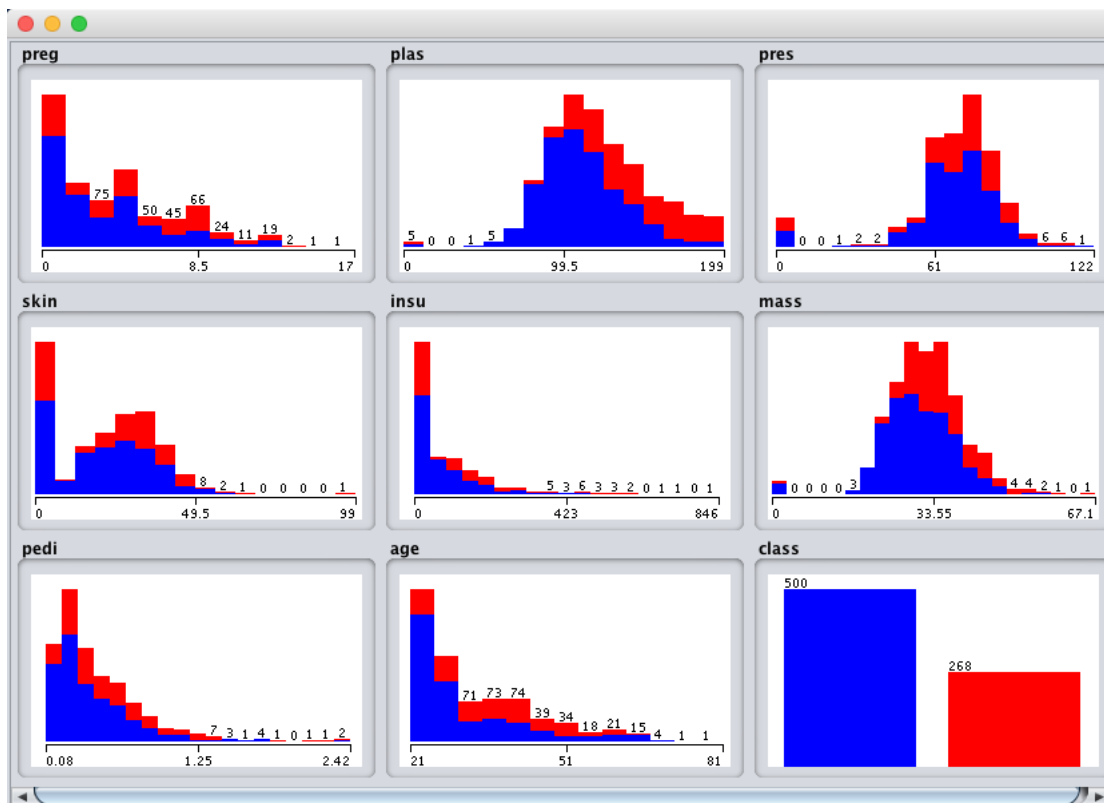


Figure 24.2: Weka Pima Indians Univariate Attribute Distributions.



We can notice a few things about the shape of the data:

- Some attributes have a Gaussian-like distribution such as `plas`, `pres`, `skin` and `mass`, suggesting methods that make this assumption could achieve good results, like Logistic Regression and Naive Bayes.
- We see a lot of overlap between the classes across the attribute values. The classes do not seem easily separable.
- We can clearly see the class imbalance graphically depicted.

### 24.3.3 Attribute Interactions

- Click the *Visualize* tab and let's review some interactions between the attributes.
- Increase the window size so all plots are visible.
- Increase the *PointSize* to 3 to make the dots easier to see.
- Click the *Update* button to apply the changes.

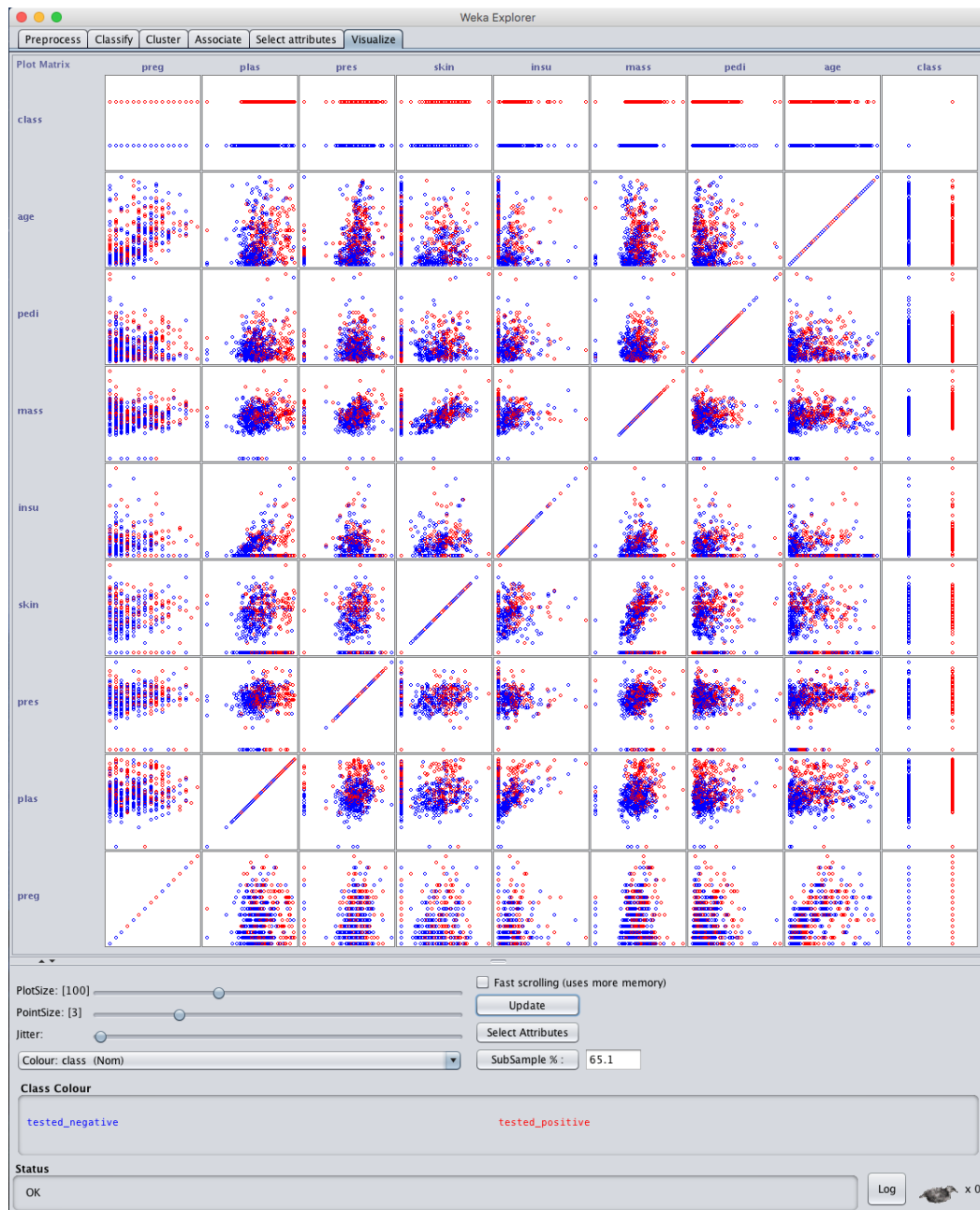


Figure 24.3: Weka Pima Indians Scatter Plot Matrix.

Looking across the graphs for the input variables, we can generally see poor separation between the classes on the scatter plots. This dataset will not be a walk in the park. It suggests that we could benefit from some good data transforms and creating multiple views of the dataset. It also suggests we may get benefits from using ensemble methods.

## 24.4 Prepare Views of the Dataset

We noted in the previous section that this may be a difficult problem and that we may benefit from multiple views of the data. In this section we will create varied views of the data, so

that when we evaluate algorithms in the next section we can get an idea of the views that are generally better at exposing the structure of the classification problem to the models. We are going to create 3 additional views of the data, so that in addition to the raw data we will have 4 different copies of the dataset in total. We will create each view of the dataset from the original and save it to a new file for later use in our experiments.

### 24.4.1 Normalized View

The first view we will create is of all the input attributes normalized to the range 0 to 1. This may benefit multiple algorithms that can be influenced by the scale of the attributes, like regression and instance-based methods.

1. In the *Weka Explorer* with the `data/diabetes.arff` file loaded.
2. Click the *Choose* button in the *Filter* pane and choose the *unsupervised.attribute.Normalize* filter.
3. Click the *Apply* button to apply the filter.
4. Click each attribute in the *Attributes* pane and review the min and max values in the *Selected attribute* pane to confirm they are 0 and 1.
5. Click the *Save...* button, navigate to a suitable directory and type in a suitable name for this transformed dataset, such as `diabetes-normalize.arff`.
6. Close the *Weka Explorer* interface to avoid contaminating the other views we want to create.

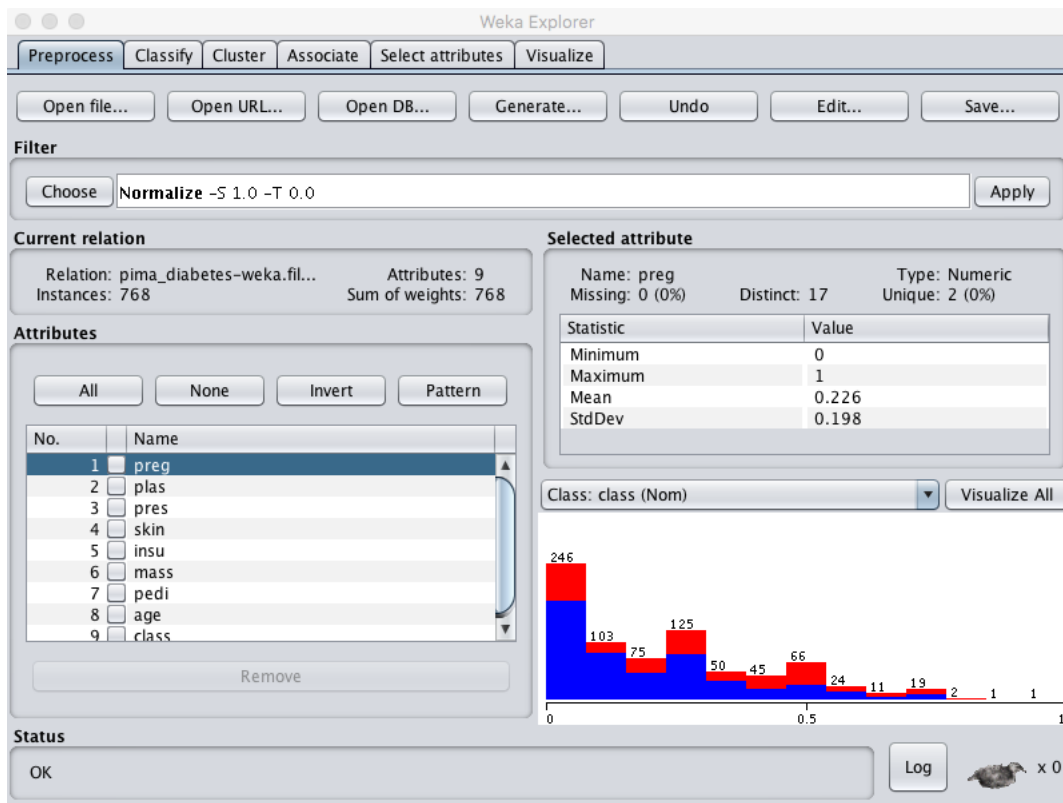


Figure 24.4: Weka Normalize Pima Indian Dataset.

### 24.4.2 Standardized View

We noted in the previous section that some of the attribute have a Gaussian-like distribution. We can rescale the data and take this distribution into account by using a standardizing filter. This will create a copy of the dataset where each attribute has a mean value of 0 and a standard deviation (mean variance) of 1. This may benefit algorithms in the next section that assume a Gaussian distribution in the input attributes, like Logistic Regression and Naive Bayes.

1. Open the *Weka Explorer*.
2. Load the Pima Indians onset of diabetes dataset `data/diabetes.arff`.
3. Click the *Choose* button in the *Filter* pane and choose the *unsupervised.attribute.Standardize* filter.
4. Click the *Apply* button to apply the filter.
5. Click each attribute in the *Attributes* pane and review the mean and standard deviation values in the *Selected attribute* pane to confirm they are 0 and 1 respectively.
6. Click the *Save...* button, navigate to a suitable directory and type in a suitable name for this transformed dataset, such as `diabetes-standardize.arff`.
7. Close the *Weka Explorer* interface.

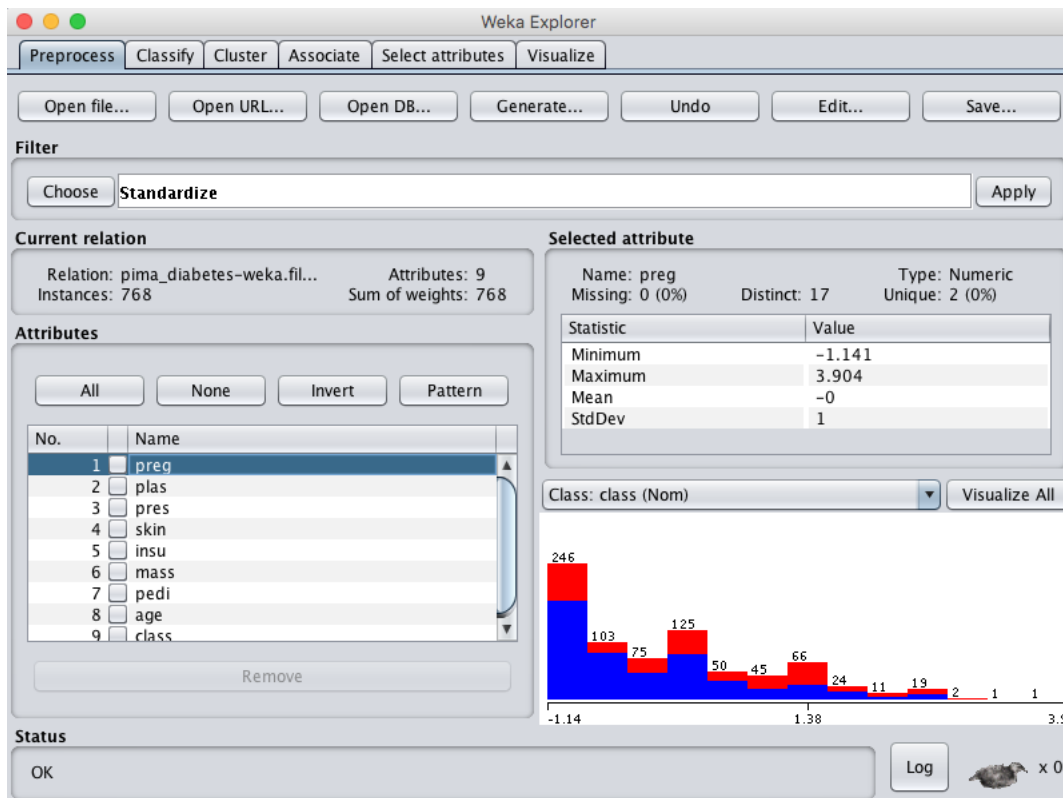


Figure 24.5: Weka Standardize Pima Indians Dataset.

### 24.4.3 Missing Data

In the previous section we suspected some of the attributes had bad or missing data marked with 0 values. We can create a new copy of the dataset with the missing data marked and then imputed with an average value for each attribute. This may help methods that assume a smooth change in the attribute distributions, such as Logistic Regression and instance-based methods. First let's mark the 0 values for some attributes as missing.

1. Open the *Weka Explorer*.
2. Load the Pima Indians onset of diabetes dataset `data/diabetes.arff`.
3. Click the *Choose* button for the Filter and select the *unsupervised.attribute.NumericalCleaner* filter.
4. Click on the filter to configure it.
5. Set the *attributeIndices* to 2-6
6. Set *minThreshold* to 0.1E-8 (close to zero), which is the minimum value allowed for each attribute.
7. Set *minDefault* to NaN, which is unknown and will replace values below the threshold.
8. Click the *OK* button on the filter configuration.

9. Click the *Apply* button to apply the filter.
10. Click each attribute in the *Attributes* pane and review the number of missing values for each attribute. You should see some non-zero counts for attributes 2 to 6.

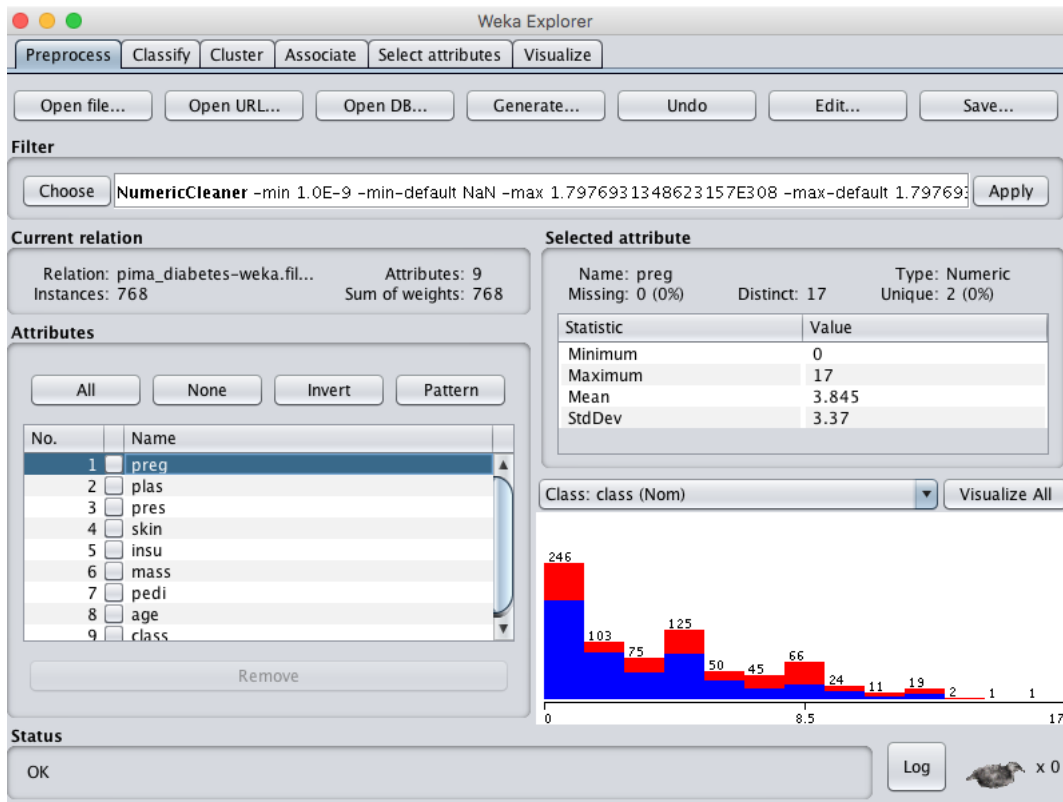


Figure 24.6: Weka Numeric Cleaner Data Filter For Pima Indians Dataset.

Now, let's impute the missing values as the mean.

1. Click the *Choose* button in the *Filter* pane and select *unsupervised.attribute.ReplaceMissingValues* filter.
2. Click the *Apply* button to apply the filter your dataset.
3. Click each attribute in the *Attributes* pane and review the number of missing values for each attribute. You should see all attributes should have no missing values and the distribution of attributes 2 to 6 should have changed slightly.
4. Click the *Save...* button, navigate to a suitable directory and type in a suitable name for this transformed dataset, such as `diabetes-missing.arff`.
5. Close the *Weka Explorer*.

Other views of the data you may want to consider investigating are subsets of features chosen by a feature selection method and a view where the class attribute is rebalanced.

## 24.5 Evaluate Algorithms

Let's design an experiment to evaluate a suite of standard classification algorithms on the different views of the problem that we created.

- 1. Click the *Experimenter* button on the *Weka GUI Chooser* to launch the *Weka Experiment Environment*.
- 2. Click *New* to start a new experiment.
- 3. In the *Datasets* pane click *Add new...* and select the following 4 datasets:
  - `data/diabetes.arff` (the raw dataset)
  - `diabetes-normalized.arff`
  - `diabetes-standardized.arff`
  - `diabetes-missing.arff`
- 4. In the *Algorithms* pane click *Add new...* and add the following 8 classification algorithms:
  - `rules.ZeroR`
  - `bayes.NaiveBayes`
  - `functions.Logistic`
  - `functions.SMO`
  - `lazy.IBk`
  - `rules.PART`
  - `trees.REPTree`
  - `trees.J48`
- 5. Select *IBk* in the list of algorithms and click the *Edit selected...* button.
- 6. Change *KNN* from *1* to *3* and click the *OK* button to save the settings.

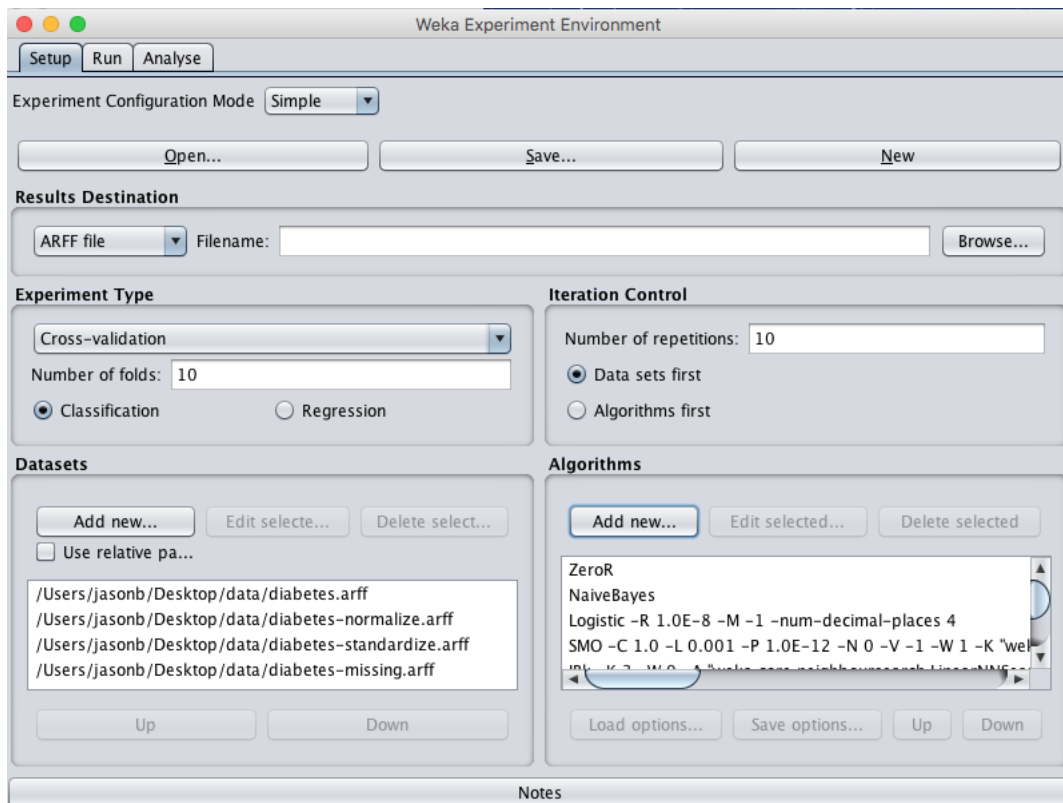


Figure 24.7: Weka Algorithm Comparison Experiment for Pima Indians Dataset.

- 7. Click on *Run* tab and click the *Start* button to run the experiment. The experiment should complete in just a few seconds.
- 8. Click on the *Analyse* tab. Click the *Experiment* button to load the results from the experiment.



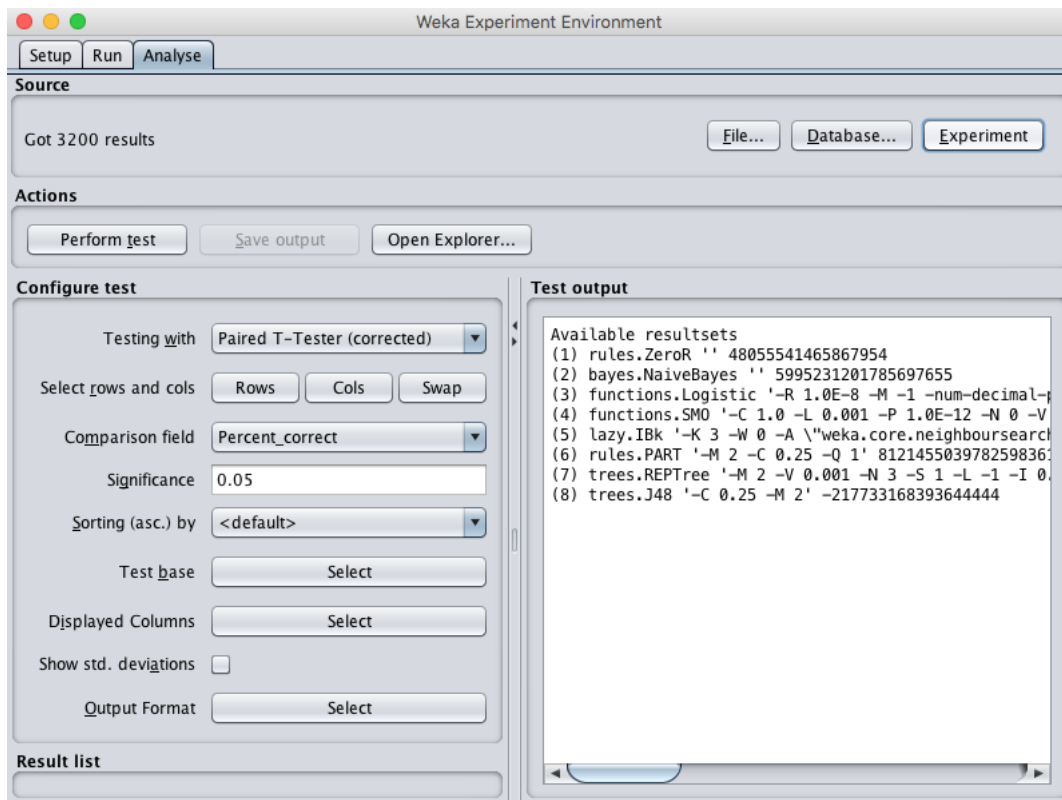


Figure 24.8: Weka Load Algorithm Comparison Experiment Results for Pima Indians Dataset.

- 9. Click the *Perform test* button to perform a pairwise test-test comparing all of the results to the results for *ZeroR*.

Dataset	(1) rules.Ze	(2) bayes	(3) funct	(4) funct	(5) lazy.	(6) rules	(7) trees	(8) trees
pima_diabetes	65.11	75.75 v	77.47 v	76.80 v	73.86 v	73.45 v	74.46 v	74.49 v
normalized	65.11	75.77 v	77.47 v	76.80 v	73.86 v	73.45 v	74.42 v	74.49 v
standardized	65.11	75.65 v	77.47 v	76.81 v	73.86 v	73.45 v	74.39 v	74.49 v
missing	65.11	74.81 v	76.86 v	76.30 v	73.54 v	73.03 v	73.70 v	74.69 v
	(v/ /*)	(4/0/0)	(4/0/0)	(4/0/0)	(4/0/0)	(4/0/0)	(4/0/0)	(4/0/0)

Key:  
 (1) rules.ZeroR  
 (2) bayes.NaiveBayes  
 (3) functions.Logistic  
 (4) functions.SMO  
 (5) lazy.IBk  
 (6) rules.PART  
 (7) trees.REPTree  
 (8) trees.J48

Listing 24.1: Results from the Algorithm Comparison Experiment.

We can see that all of the algorithms are skillful on all of the views of the dataset compared to *ZeroR*. We can also see that our baseline for skill is 65.11% accuracy. Just looking at the raw

classification accuracies, we can see that the view of the dataset with missing values imputed looks to have resulted in lower model accuracy in general. It also looks like there is little difference between the standardized and normalized results as compared to the raw results other than a few fractions of percent. It suggests we can probably stick with the raw dataset. Finally, it looks like Logistic regression may have achieved higher accuracy results than the other algorithms, let's check if the difference is significant.

- 10. Click the *Select* button for *Test base* and choose *functions.Logistic*. Click the *Perform test* button to rerun the analysis.

Dataset	(3) function	(1) rules	(2) bayes	(4) funct	(5) lazy.	(6) rules	(7) trees	(8) trees
pima_diabetes	77.47	65.11 *	75.75	76.80	73.86 *	73.45 *	74.46 *	74.49
normalized	77.47	65.11 *	75.77	76.80	73.86 *	73.45 *	74.42 *	74.49
standardized	77.47	65.11 *	75.65	76.81	73.86 *	73.45 *	74.39 *	74.49
missing	76.86	65.11 *	74.81	76.30	73.54 *	73.03 *	73.70 *	74.69
(v/ /*)		(0/0/4)	(0/4/0)	(0/4/0)	(0/0/4)	(0/0/4)	(0/0/4)	(0/4/0)

Key:

(1) rules.ZeroR  
 (2) bayes.NaiveBayes  
 (3) functions.Logistic  
 (4) functions.SMO  
 (5) lazy.IBk  
 (6) rules.PART  
 (7) trees.REPTree  
 (8) trees.J48

Listing 24.2: Results from the Algorithm Comparison Experiment With a New Test Base.

It does look like the logistic regression results are better than some of the other results, such as *IBk*, *PART*, *REPTree* and *ZeroR*, but not statistically significantly different from *NaiveBayes*, *SMO* or *J48*.

- 11. Check *Show std. deviations* to show standard deviations.
- 12. Click the *Select* button for *Displayed Columns* and choose *functions.Logistic*, click *Select* to accept the selection. This will only show results for the logistic regression algorithm.
- 13. Click *Perform test* to rerun the analysis.

We now have a final result we can use to describe our model.

Dataset	(3) functions.Logist
pima_diabetes	77.47(4.39)
normalized	77.47(4.39)
standardized	77.47(4.39)
missing	76.86(4.90)

```
(v/ /*) |
```

Key:

```
(3) functions.Logistic
```

Listing 24.3: Final Results for the Logistic Regression Algorithm.

We can see that the estimated accuracy of the model on unseen data is 77.47% with a standard deviation of 4.39%.

- 14. Close the *Weka Experiment Environment*.

## 24.6 Finalize Model and Present Results

We can create a final version of our model trained on all of the training data and save it to file.

1. Open the *Weka Explorer* and load the `data/diabetes.arff` dataset.
2. Click on the *Classify*.
3. Select the *functions.Logistic* algorithm.
4. Change the *Test options* from *Cross Validation* to *Use training set*.
5. Click the *Start* button to create the final model.
6. Right click on the result item in the *Result list* and select *Save model*. Select a suitable location and type in a suitable name, such as *diabetes-logistic* for your model.

This model can then be loaded at a later time and used to make predictions on new data. We can use the mean and standard deviation of the model accuracy collected in the last section to help quantify the expected variability in the estimated accuracy of the model on unseen data. We can generally expect that the performance of the model on unseen data will be 77.47% plus or minus ( $2 \times 4.39$ )% or 8.78%. We can restate this as between 68.96% and 86.25% accurate. What is surprising about this final statement of model accuracy is that at the lower end, the model is only a shade better than the *ZeroR* model that achieved an accuracy of 65.11% by predicting a negative outcome for all predictions.

## 24.7 Summary

In this project you completed a binary classification machine learning project end-to-end using the Weka machine learning workbench. Specifically, you learned:

- How to analyze your dataset and suggest at specific data transform and modeling techniques that may be useful.
- How to design and save multiple views of your data and spot-check multiple algorithms on these views.
- How to finalize the model for making predictions on new data and presenting the estimated accuracy of the model on unseen data.

### 24.7.1 Next

Next in the final project, you will work through a regression machine learning problem. This is the largest project and involves also using algorithm tuning and ensemble methods in an effort to achieve improved model performance.

# Chapter 25

## How to Work Through a Regression Machine Learning Project

The more time that you spend working through projects, the faster you will develop your skills in applied machine learning. As such, it is important to work on a suite of different problem types. In this project you will discover how to work through a regression problem in Weka, end-to-end. After completing this project you will know:

- How to load and analyze a regression dataset in Weka.
- How to create multiple different transformed views of the data and evaluate a suite of algorithms on each.
- How to finalize and present the results of a model for making predictions on new data.

Let's get started.

### 25.1 Tutorial Overview

This tutorial will walk you through the key steps required to complete a machine learning project in Weka. We will work through the following steps:

1. Load the dataset.
2. Analyze the dataset.
3. Prepare views of the dataset.
4. Evaluate algorithms.
5. Tune algorithm performance.
6. Evaluate ensemble algorithms.
7. Present results.

## 25.2 Load the Dataset

In this tutorial we will work on the Boston House Price dataset. In this dataset, each instance describes the properties of a Boston suburb and the task is to predict the house prices in thousands of dollars. You can learn more about this dataset in Section [8.4.2](#).

1. Open the *Weka GUI Chooser*.
2. Click the *Explorer* button to open the *Weka Explorer*.
3. Click the *Open file...* button, navigate to the `numeric/` directory and select `housing.arff`. Click the *Open* button.

The dataset is now loaded into Weka.

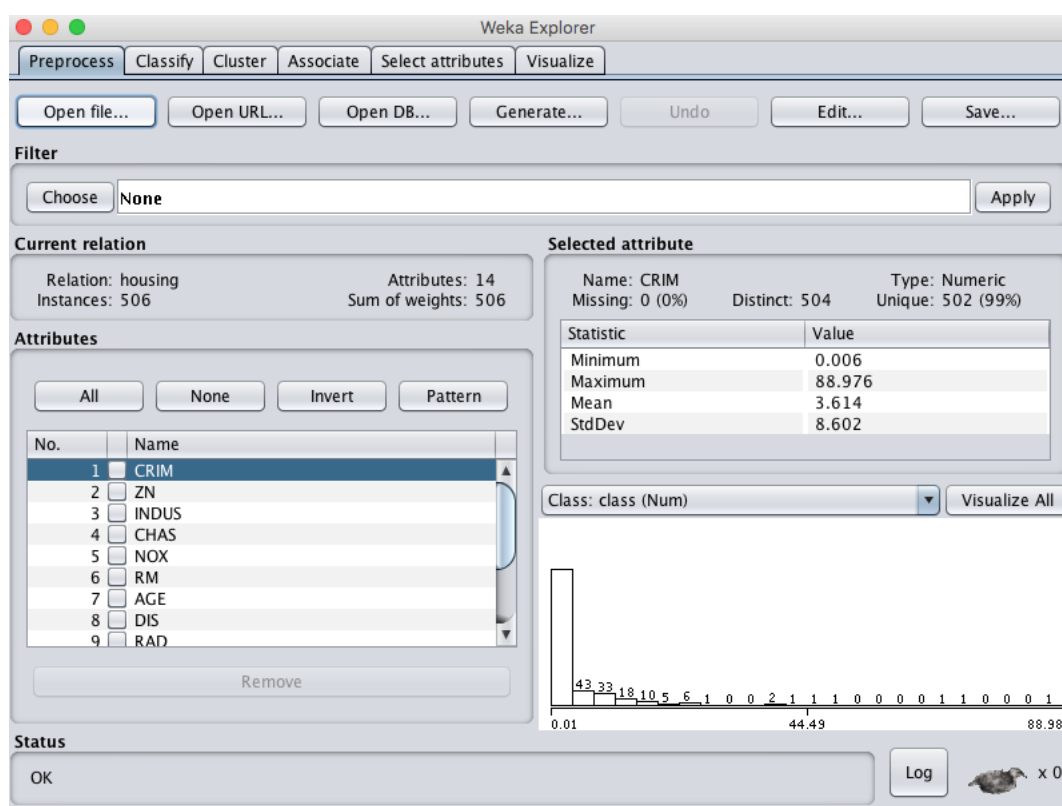


Figure 25.1: Weka Load the Boston House Price Dataset.

## 25.3 Analyze the Dataset

It is important to review your data before you start modeling. Reviewing the distribution of each attribute and the interactions between attributes may shed light on specific data transforms and specific modeling techniques that we could use.

### 25.3.1 Summary Statistics

Review the details about the dataset in the *Current relation* pane. We can notice a few things:

- The dataset is called housing.
- There are 506 instances. If we use 10-fold cross validation later to evaluate the algorithms, then each fold will be comprised of about 50 instances, which is fine.
- There are 14 attributes, 13 inputs and 1 output variable.
- Click on each attribute in the *Attributes* pane and review the summary statistics in the *Selected attribute* pane.

We can notice a few facts about our data:

- There are no missing values for any of the attributes.
- All inputs are numeric except one binary attribute, and have values in differing ranges.
- The last attribute is the output variable called class, it is numeric.

We may see some benefit from either normalizing or standardizing the data.

### 25.3.2 Attribute Distributions

- Click the *Visualize All* button and let's review the graphical distribution of each attribute.

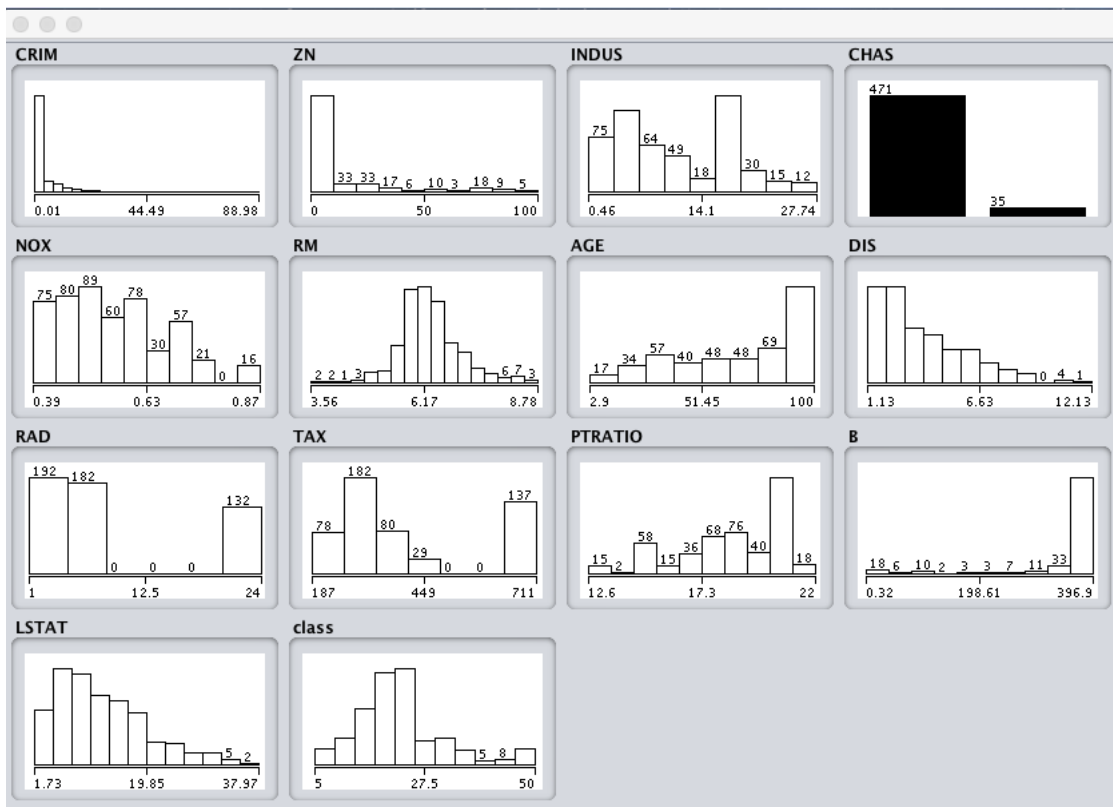


Figure 25.2: Weka Boston House Price Univariate Attribute Distributions.

We can notice a few things about the shape of the data:

- We can see that the attributes have a range of differing distributions.
- The `CHAS` attribute looks like a binary distribution (two values).
- The `RM` attribute looks like it has a Gaussian distribution.

We may see more benefit in using nonlinear regression methods like decision trees and such, than using linear regression methods like linear regression.

### 25.3.3 Attribute Interactions

- Click the *Visualize* tab and let's review some interactions between the attributes.
  1. Decrease the *PlotSize* to 50 and adjust the window size so all plots are visible.
  2. Increase the *PointSize* to 3 to make the dots easier to see.
  3. Click the *Update* button to apply the changes.



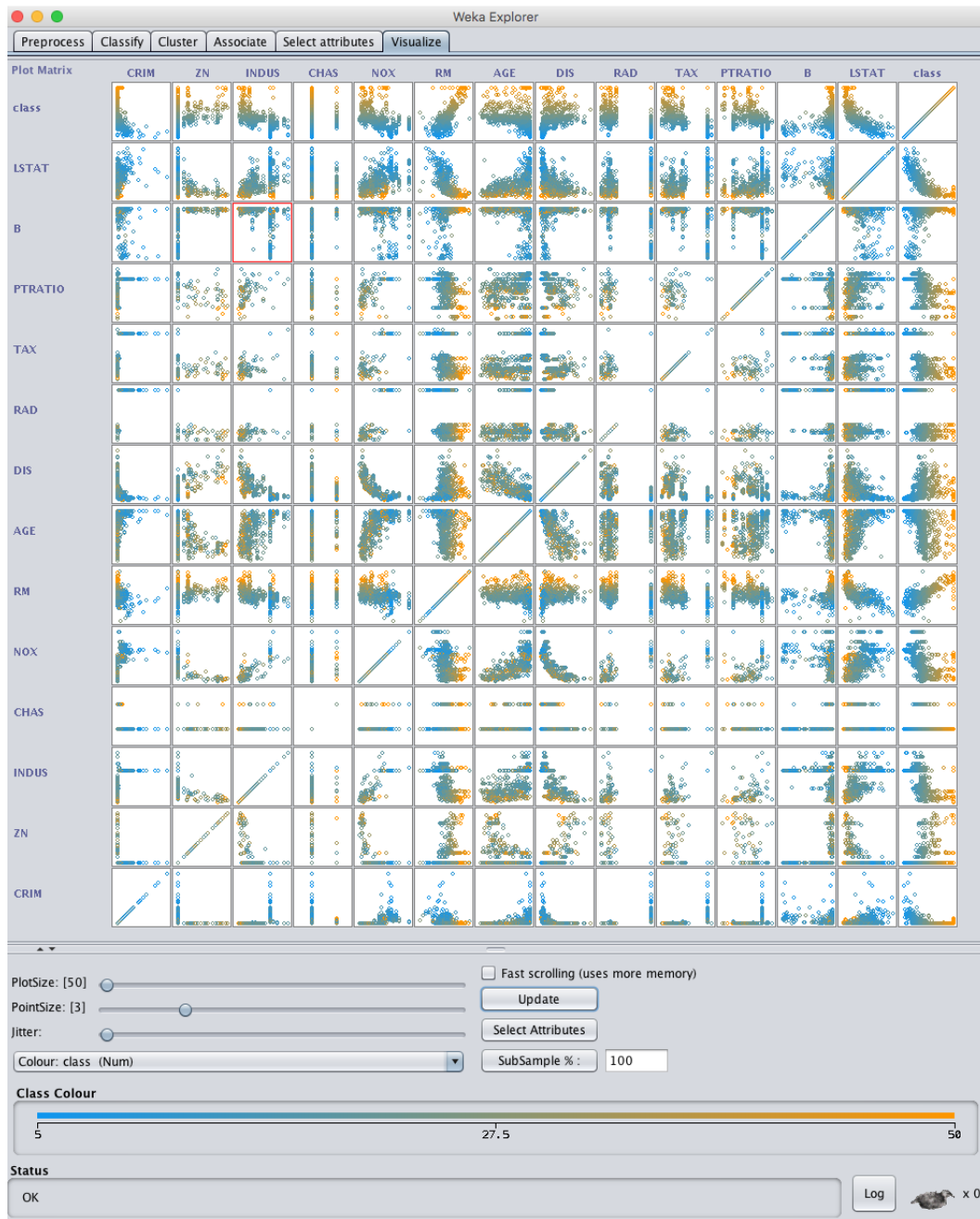


Figure 25.3: Weka Boston House Price Scatter Plot Matrix.

Looking across the graphs we can see some structured relationships that may aid in modeling such as DIS vs NOX and AGE vs NOX. We can also see some structure relationships between input attributes and the output attribute such as LSTAT and CLASS and RM and CLASS.

## 25.4 Prepare Views of the Dataset

In this section we will create some different views of the data, so that when we evaluate algorithms in the next section we can get an idea of the views that are generally better at exposing the structure of the regression problem to the models. We are first going to create a

modified copy of the original `housing.arff` data file, then make 3 additional transforms of the data. We will create each view of the dataset from our modified copy of the original and save it to a new file for later use in our experiments.

### 25.4.1 Modified Copy

The `CHAS` attribute is nominal (binary) with the values 0 and 1. We want to make a copy of the original `housing.arff` data file and change `CHAS` to a numeric attribute so that all input attributes are numeric. This will help with transforming and modeling the dataset.

- Locate the `housing.arff` dataset and create a copy of it in the same directory called `housing-numeric.arff`.
- Open this modified file `housing-numeric.arff` in a text editor and scroll down to where the attributes are defined, specifically the `CHAS` attribute on line 56.

```
@relation 'housing'
@attribute CRIM real
@attribute ZN real
@attribute INDUS real
@attribute CHAS { 0, 1}
@attribute NOX real
@attribute RM real
@attribute AGE real
@attribute DIS real
@attribute RAD real
@attribute TAX real
@attribute PTRATIO real
@attribute B real
@attribute LSTAT real
@attribute class real
@data
```

Figure 25.4: Weka Boston House Price Attribute Data Types.

Change the definition of the `CHAS` attribute from:

```
@attribute CHAS { 0, 1}
```

to:

```
@attribute CHAS real
```

The `CHAS` attribute is now numeric rather than nominal. This modified copy of the dataset `housing-numeric.arff` will now be used as the baseline dataset.

```
@relation 'housing'
@attribute CRIM real
@attribute ZN real
@attribute INDUS real
@attribute CHAS real
@attribute NOX real
@attribute RM real
@attribute AGE real
@attribute DIS real
@attribute RAD real
@attribute TAX real
@attribute PTRATIO real
@attribute B real
@attribute LSTAT real
@attribute class real
@data
```

Figure 25.5: Weka Boston House Price Dataset With Numeric Data Types.

### 25.4.2 Normalized Dataset

The first view we will create is of all the input attributes normalized to the range 0 to 1. This may benefit multiple algorithms that can be influenced by the scale of the attributes, like regression and instance-based methods.

1. Open the *Weka Explorer*.
2. Open the modified numeric dataset `housing-numeric.arff`.
3. Click the *Choose* button in the *Filter* pane and choose the *unsupervised.attribute.Normalize* filter.
4. Click the *Apply* button to apply the filter.
5. Click each attribute in the *Attributes* pane and review the min and max values in the *Selected attribute* pane to confirm they are 0 and 1.
6. Click the *Save...* button, navigate to a suitable directory and type in a suitable name for this transformed dataset, such as `housing-normalize.arff`.
7. Close the *Weka Explorer* interface.

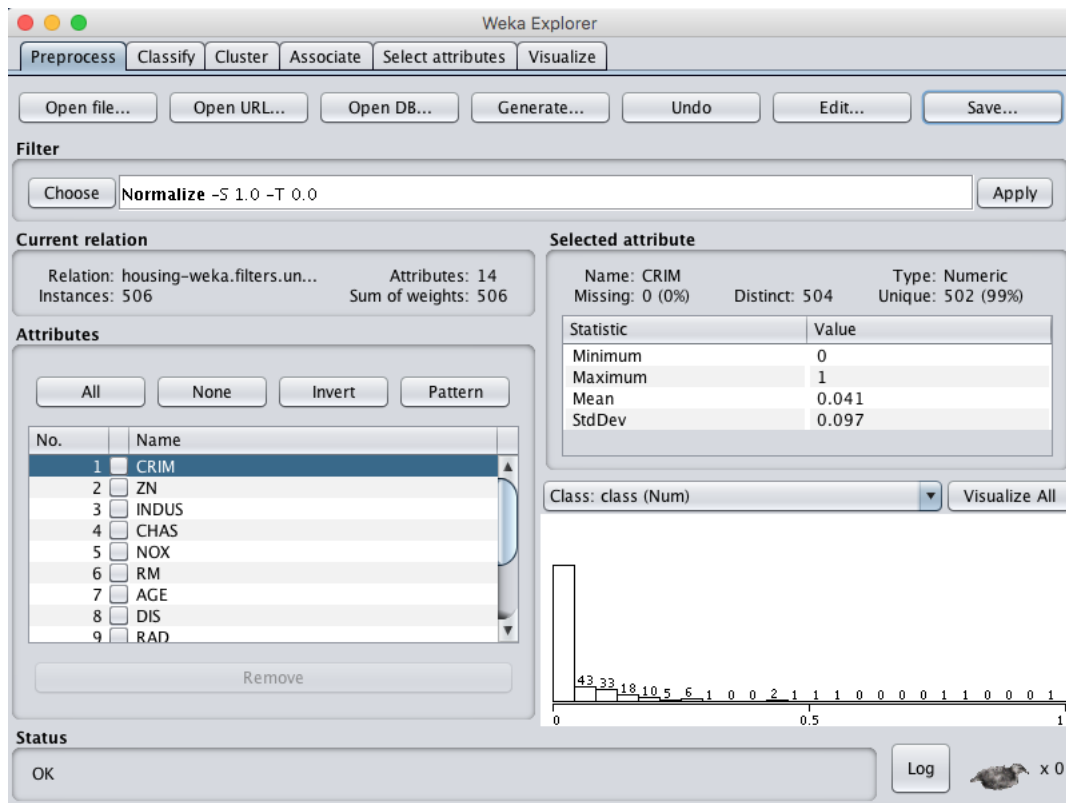


Figure 25.6: Weka Boston House Price Dataset Normalize Data Filter.

### 25.4.3 Standardized Dataset

We noted in the previous section that some of the attribute have a Gaussian-like distribution. We can rescale the data and take this distribution into account by using a standardizing filter. This will create a copy of the dataset where each attribute has a mean value of 0 and a standard deviation (mean variance) of 1. This may benefit algorithms in the next section that assume a Gaussian distribution in the input attributes, like Logistic Regression and Naive Bayes.

1. Open the *Weka Explorer*.
2. Open the modified numeric dataset `housing-numeric.arff`.
3. Click the *Choose* button in the *Filter* pane and choose the *unsupervised.attribute.Standardize* filter.
4. Click the *Apply* button to apply the filter.
5. Click each attribute in the *Attributes* pane and review the mean and standard deviation values in the *Selected attribute* pane to confirm they are 0 and 1 respectively.
6. Click the *Save...* button, navigate to a suitable directory and type in a suitable name for this transformed dataset, such as `housing-standardize.arff`.
7. Close the *Weka Explorer* interface.

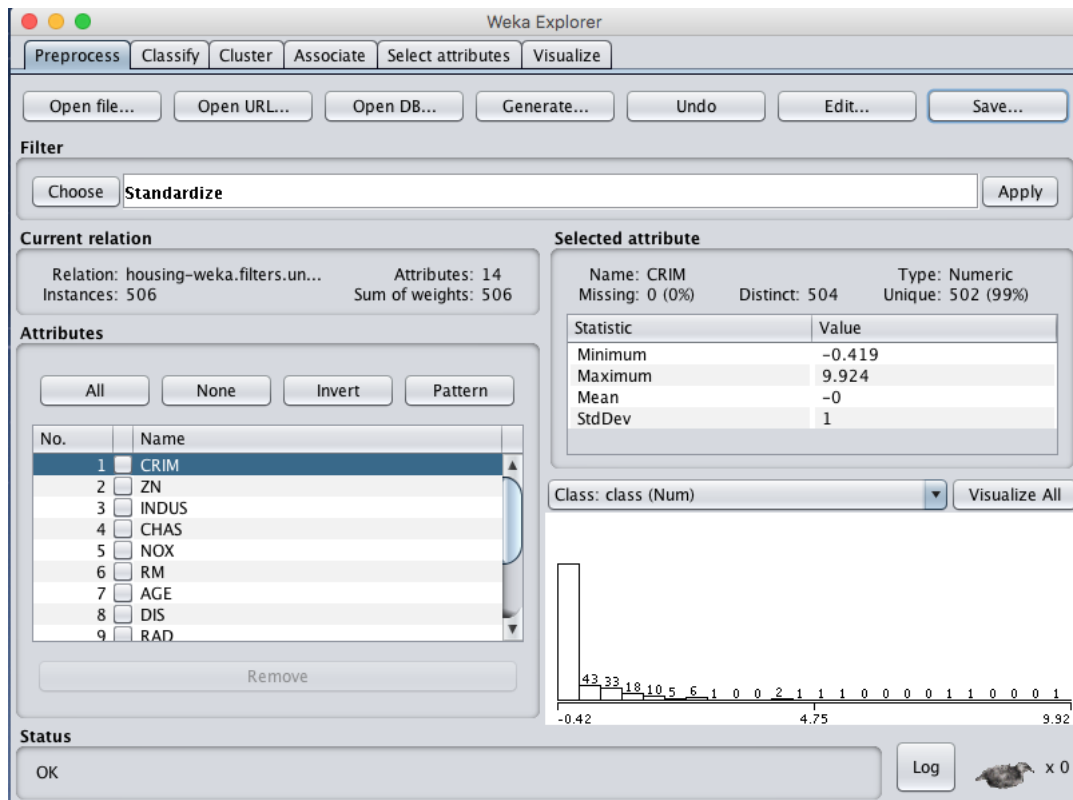


Figure 25.7: Weka Boston House Price Dataset Standardize Data Filter.

#### 25.4.4 Feature Selection

We are unsure whether all of the attributes are really needed in order to make predictions. Here, we can use automatic feature selection to select only those most relevant attributes in the dataset.

1. Open the *Weka Explorer*.
2. Open the modified numeric dataset `housing-numeric.arff`.
3. Click the *Choose* button in the *Filter* pane and choose the *supervised.attribute.AttributeSelection* filter.
4. Click the *Apply* button to apply the filter.
5. Click each attribute in the *Attributes* pane and review the 5 chosen attributes.
6. Click the *Save...* button, navigate to a suitable directory and type in a suitable name for this transformed dataset, such as `housing-feature-selection.arff`.
7. Close the *Weka Explorer* interface.

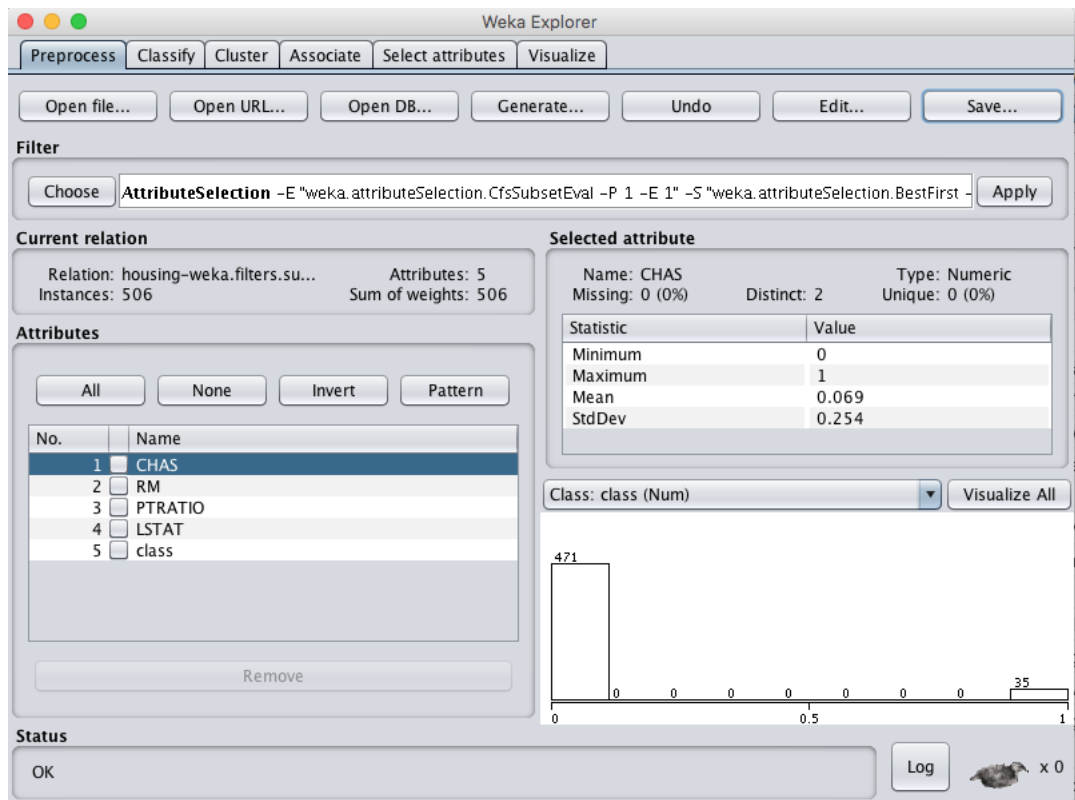


Figure 25.8: Weka Boston House Price Dataset Feature Selection Data Filter.

## 25.5 Evaluate Algorithms

Let's design an experiment to evaluate a suite of standard classification algorithms on the different views of the problem that we created.

- 1. Click the *Experimenter* button on the *Weka GUI Chooser* to launch the *Weka Experiment Environment*.
- 2. Click *New* to start a new experiment.
- 3. In the *Experiment Type* pane change the problem type from *Classification* to *Regression*.
- 4. In the *Datasets* pane click *Add new...* and select the following 4 datasets:
  - `housing-numeric.arff`
  - `housing-normalized.arff`
  - `housing-standardized.arff`
  - `housing-feature-selection.arff`
- 5. In the *Algorithms* pane click *Add new...* and add the following 6 regression algorithms:
  - `rules.ZeroR`
  - `bayes.SimpleLinearRegression`

- *functions.SMOreg*
  - *lazy.IBk*
  - *trees.REPTree*
- 6. Select *IBk* in the list of algorithms and click the *Edit selected...* button.
  - 7. Change *KNN* from *1* to *3* and click the *OK* button to save the settings.

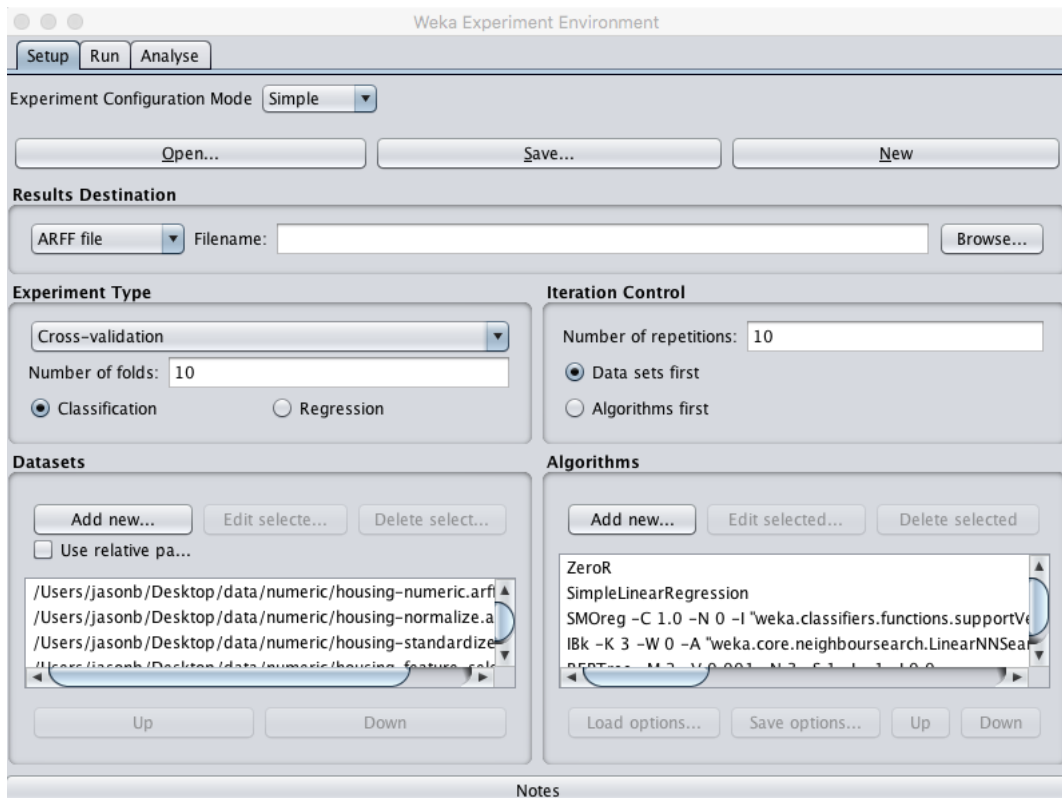


Figure 25.9: Weka Boston House Price Algorithm Comparison Experiment Design.

- 8. Click on *Run* tab and click the *Start* button to run the experiment. The experiment should complete in just a few seconds.
- 9. Click on the *Analyse* to open the *Analyse* tab. Click the *Experiment* button to load the results from the experiment.

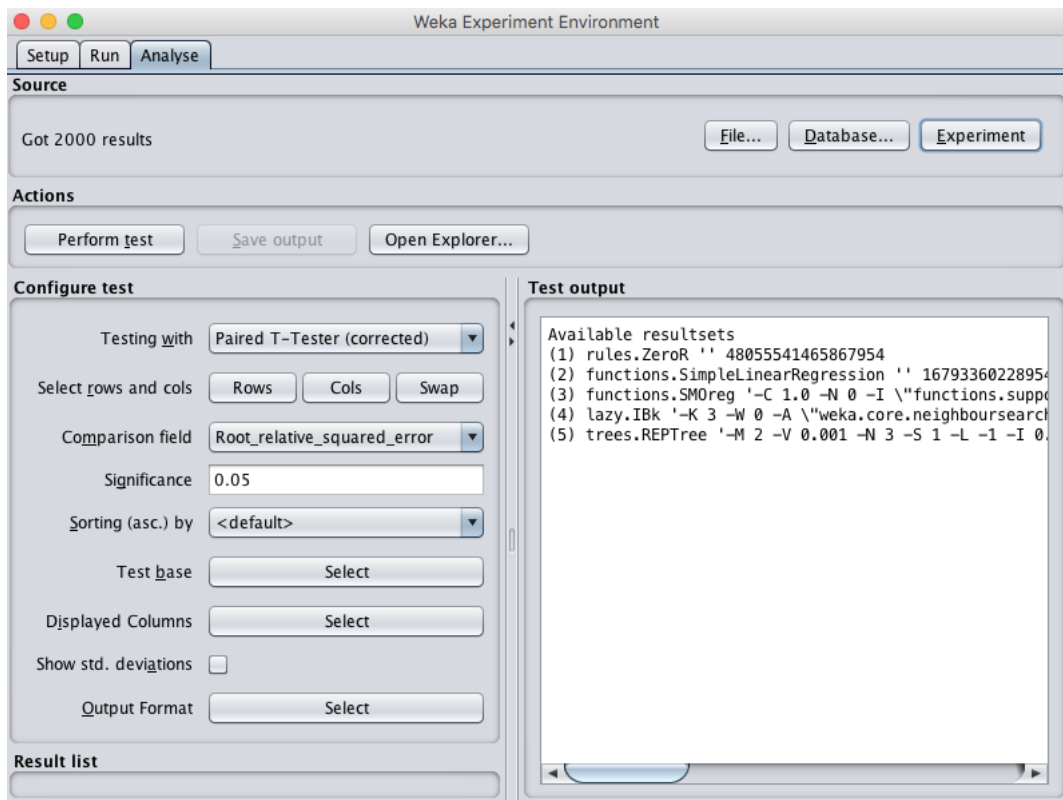


Figure 25.10: Weka Boston House Price Dataset Load Algorithm Comparison Experiment Results.

- 10. Change the *Comparison field* to *Root\_mean\_squared\_error*.
- 11. Click the *Perform test* button to perform a pairwise test comparing all of the results to the results for *ZeroR*.

Dataset	(1) rules.Z	(2) func	(3) func	(4) lazy	(5) tree
housing	9.11	6.22 *	4.95 *	4.41 *	4.64 *
normalized	9.11	6.22 *	4.94 *	4.41 *	4.63 *
standardized	9.11	6.22 *	4.95 *	4.41 *	4.64 *
feature-selection	9.11	6.22 *	5.19 *	4.27 *	4.64 *
-----					
	(v/ /*)	(0/0/4)	(0/0/4)	(0/0/4)	(0/0/4)
-----					
Key:					
(1) rules.ZeroR					
(2) functions.SimpleLinearRegression					
(3) functions.SMOreg					
(4) lazy.IBk					
(5) trees.REPTree					

Listing 25.1: Results From Algorithm Comparison Experiment.

Remember that the lower the RMSE the better. These results are telling. Firstly, we can see that all of the algorithms are better than the baseline skill of *ZeroR* and that the difference is