**CSCE 874 : DATA MINING**

**ASSIGNMENT 04**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Contribution %** | **Contributions** | **Signature & Date** |
| Bill Mutabazi | 25 | Documented the results of Weka experimenter, report. | **Bill Mutabazi**  03/18/2018 |
| Minal Khatri | 25 | Documented the results of Weka experimenter, report. | **Minal Khatri**  03/18/2018 |
| Nancy Pham | 25 | Documented the results of Weka explorer, report. | **Nancy Pham**  03/18/2018 |
| Venkata Krishna Mohan Sunkara | 25 | Documented the results of Weka explorer, report. | **Venkata Krishna Mohan Sunkara**  03/18/2018 |

**Classification**

**Introduction:**

Classification is a supervised learning process which tries to predict categorical class labels for a given data. It is a process of classifying data based on the training set and the class labels in a classifying attribute, the classification model uses these attributes in classifying new data. In general, the classification task is defined as:

“Given a collection of records (*x,y*), where *x* is a set of attributes and *y* is the class label, find a model that maps the set of attributes *x* to the class label *y*, such that previously unseen records can be assigned a class label as accurately as possible.” [1]

For this assignment, we use 5 different classification methods available in WEKA to classify data. We now describe the different algorithms used for the classification task:

I. **ZeroR:** It is the simplest classification method which relies on the target (class label) and ignores all predictors (attributes). This classifier simply predicts the majority category (class). Although there is no predictability power in ZeroR, it is useful for determining a baseline performance as a benchmark for other classification methods. [2]

II. **OneR**: OneR short for “One Rule” is a simple, yet an accurate classification algorithm that generates one rule for each predictor in data, then selects the rule with the smallest total error as its “one rule”. To create a rule for a predictor, it constructs a frequency table for each predictor (attribute) against the target (class). OneR algorithm produces rules that are slightly less accurate than state-of-the-art classification algorithms while producing rules that are simple for humans to interpret.One must choose an algorithm which has better accuracy along with high level of interpretability (trade off between complexity and interpretability). [3]

III. **Naive Bayes**: Naive Bayes is a simple technique for building classifiers that assign class labels to instances, represented as vectors of attributes, where class labels are drawn from a finite set. It is based on the Bayes theorem with strong independence between the attributes of an instance. This algorithm is more accurate in presence of large data and performs surprisingly well on large data. [4]

IV. **J48**: It is a predictive model that decides the target value (class label) of a new instance based on various attribute values of the available data. It is a decision tree algorithm which creates a decision tree based on the attribute values of the available training data. It creates a branch based on the most influential attribute at every step and builds a tree to classify the training instances. [5]

V. **Ibk**: Instance-based learning with parameter K is a non-parametric method used for classification. It is an implementation of K- nearest neighbors algorithm where the class label is determined by a majority vote of the k-nearest neighbors of an instance. It is also called “lazy learning” as the function is only approximated locally and all the computation is deferred. [6]

**Data Set:**

We utilize two different datasets to compare our results among different algorithms. The datasets are in the arff format.

1. **Contact-lenses.arff**: This is a small dataset of 24 instances and 4 nominal attributes along with a class-label. It consists of data related to the type of contact-lenses that must be given to a person based on four relevant attributes. All of the instances are complete and noise free.
2. **Iris.arff**: This is perhaps the best known data set to be found in the pattern recognition literature. Fisher's paper is a classic in the field and is referenced frequently to this day. (See Duda & Hart, for example). The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other. It consists of 4 continuous attributes along with a class-label.

**Evaluation Methods**:

We used 3 different evaluation methods for assessing the accuracy of different algorithms. All of these techniques are useful to get a better sense of the model about how it performs on previously unseen data. We now briefly describe the different evaluation methods used:

1. **Training Set**: This evaluation method calculates the classification rate on the training set itself. This gives an idea about how our model performs on the training set. In general, nothing much is known about the accuracy of a model on previously unseen test data.
2. **10-fold Cross-Validation**: This evaluation method captures the accuracy of a model better than other evaluation strategies as it considers the variance of data and helps in generalizing well on the previously unseen data. This method partitions the entire dataset into 10 equal-width bins and then trains on 9 bins and tests on the remaining bin. This process is then continued for 10 folds with each bin acting as a test bin during every fold. The final result is the average of 10 folds.
3. **Percentage split (set to 66%)**: This evaluation method splits the data in 2:1 fashion, such that 66% of the data is used for training the model and the remaining 34% is used for testing the model. This evaluation methods gives a better understanding of the accuracy of a model.

**Data Classification using WEKA Explorer:**

The contact-lenses dataset is first loaded into WEKA by selecting the Open file button. Then choose the classify window of Weka Explorer. Different classification methods can be selected using the choose button.

1. **ZeroR Classification**: The ZeroR classification algorithm is the most naive classifier of all and is used for determining baseline performance. This algorithm is available in Weka under the rules subsection of classifiers. It can be selected as ‘Choose’ → ‘weka’ → ‘classifiers’ → ‘rules’ → ‘ZeroR’. Figure 1 represents the ZeroR parameter editor (Generic Object Editor) along with the classification results for contact-lenses dataset. We observe a classification rate of 62.5% as it classifies all the instances to the maximum class (none,15). Here, we use training set as our evaluation method which is present in ‘Test options’ of the ‘Classify’ window.

|  |  |
| --- | --- |
|  |  |

Figure 1: ZeroR Classification

1. **OneR Classification**: The OneR classification algorithm is a simple classification algorithm which provides more interpretability as it considers only the best rule for classification. This algorithm is available in Weka under the rules subsection of classifiers. It can be selected as ‘Choose’ → ‘weka’ → ‘classifiers’ → ‘rules’ → ‘OneR’. Figure 2 represents the OneR parameter editor along with the classification results for contact-lenses dataset. Here, we use training set as our evaluation method.

|  |  |
| --- | --- |
|  |  |

Figure 2: OneR Classification

1. **Naive Bayes Classification**: The naive bayes method is a sophisticated algorithm for classification based on the bayes theorem. There are many algorithms based on bayes theorem but this algorithm is the most simple yet powerful classifier. It can be selected as ‘Choose’ → ‘weka’ → ‘classifiers’ → ‘bayes’ → ‘NaiveBayes’. Figure 3 represents the NaiveBayes parameter editor window along with the classification results for contact-lenses dataset. Here, we use 10 fold Cross-validation as our evaluation method which is present in ‘Test options’ of the ‘Classify’ window.

|  |  |
| --- | --- |
|  |  |

Figure 3: Naive Bayes Classification

1. **J48 Classification**: The J48 algorithm is a decision tree based algorithm which is used for classifying instances. It is considered as a powerful classifier and falls in to the family of tree classifiers. It can be selected as ‘Choose’ → ‘weka’ → ‘classifiers’ → ‘trees’ → ‘J48’. Figure 4.1 represents the J48 parameter editor along with the classification results for contact-lenses dataset. Here, we use percentage split (66%) as our evaluation method which is present in ‘Test options’ of the ‘Classify’ window.

|  |  |
| --- | --- |
|  |  |

Figure 4.1: J48 Classification

We can visualize the tree by selecting the model from the result list. Right click on the model name and select visualize tree option. It is then displayed in a separate window as shown in figure 4.2.

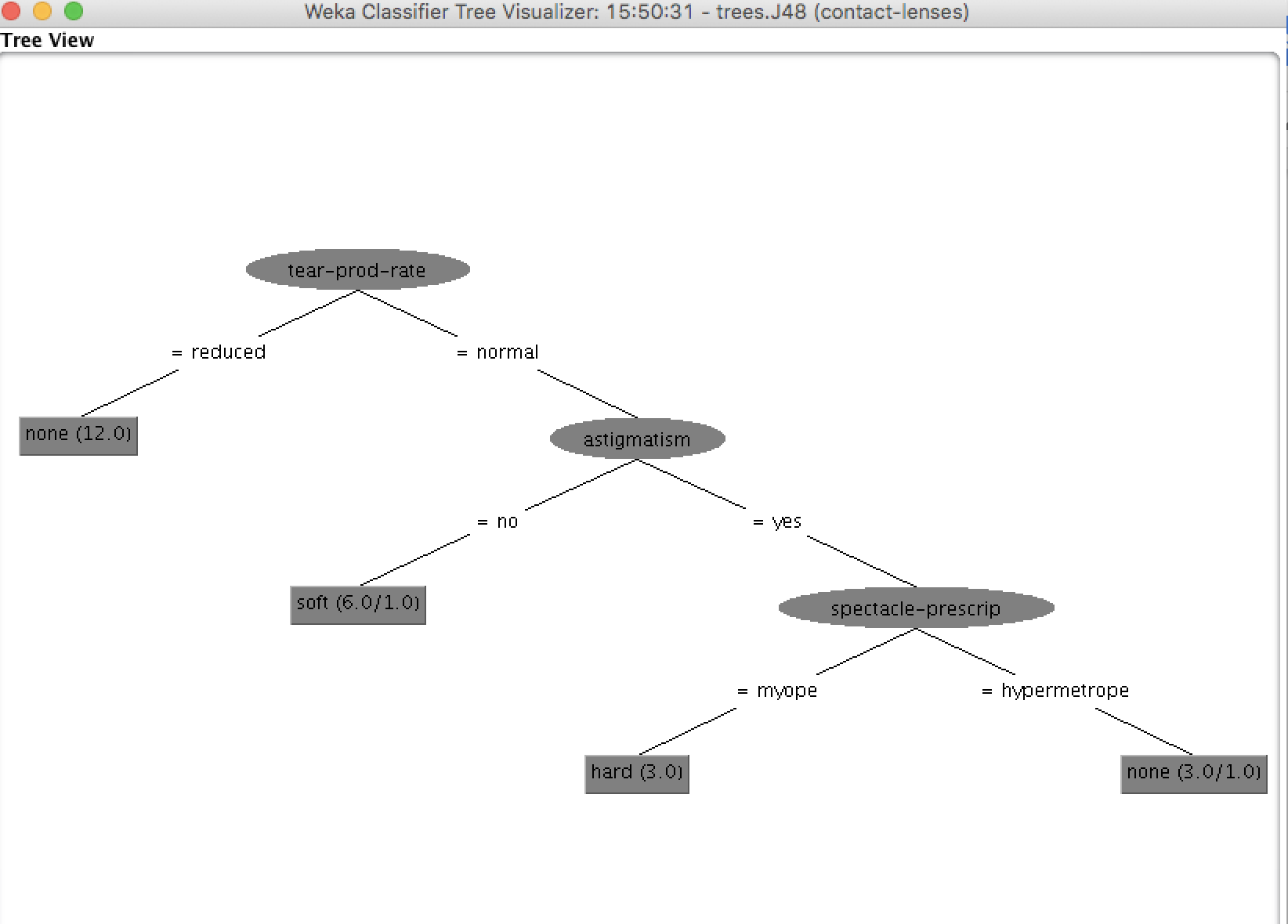


Figure 4.2: J48 tree visualization.

1. **Ibk Classification**: The Ibk algorithm is based on K-nearest neighbours classifier. It predicts a class label based on its neighbors. It assumes that instances which are close to each other tend to belong to the same class. It is present in the lazy classifiers subsection of Weka. It can be selected as ‘Choose’ → ‘weka’ → ‘classifiers’ → ‘lazy’ → ‘Ibk’. Figure 4.1 represents the Ibk parameter editor along with the classification results for contact-lenses dataset. Here, we use 10 fold Cross-validation as our evaluation method which is present in ‘Test options’ of the ‘Classify’ window.

|  |  |
| --- | --- |
|  |  |

Figure 5: Ibk Classification

We now perform the same classification on the discretized iris dataset and document our results in the ‘Results Assignment 4.csv’ file. The entire process of selecting the classifier remains the same. We represent the tree generated by J48 classifier on the discretized iris dataset using a percentage split of 66% in figure 6. In addition, it gives a one level decision tree as, decisions are made on the basis of one attribute

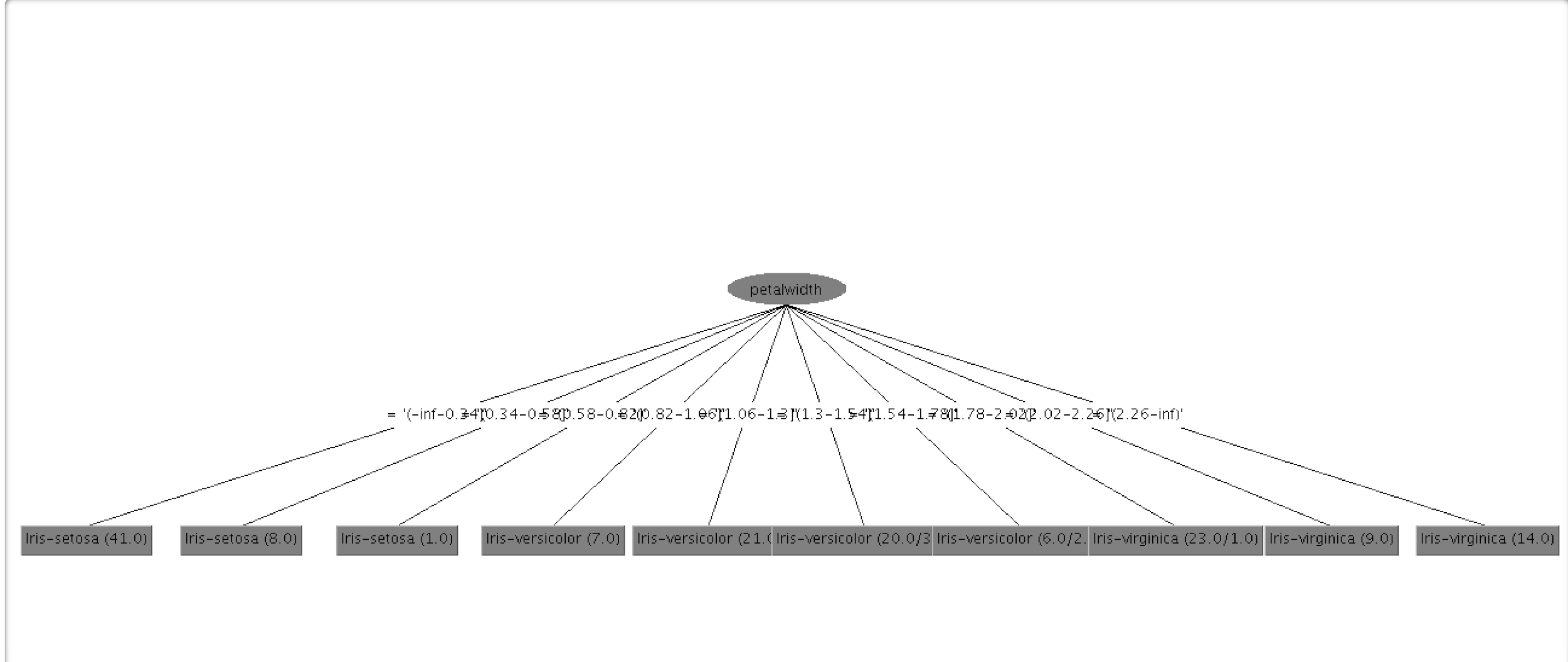


Figure 6: J48 tree visualization for Iris dataset.

**Data Classification Using Weka Experimenter:**

The Experimenter enables to set up large scale experiments, start them running , leave them and analyze the performance statistics that have been collected.It automates the experimental process.The results can be stored in either ARFF or CSV format.

***Step-1*** : Open the experimenter and we can see the following window.

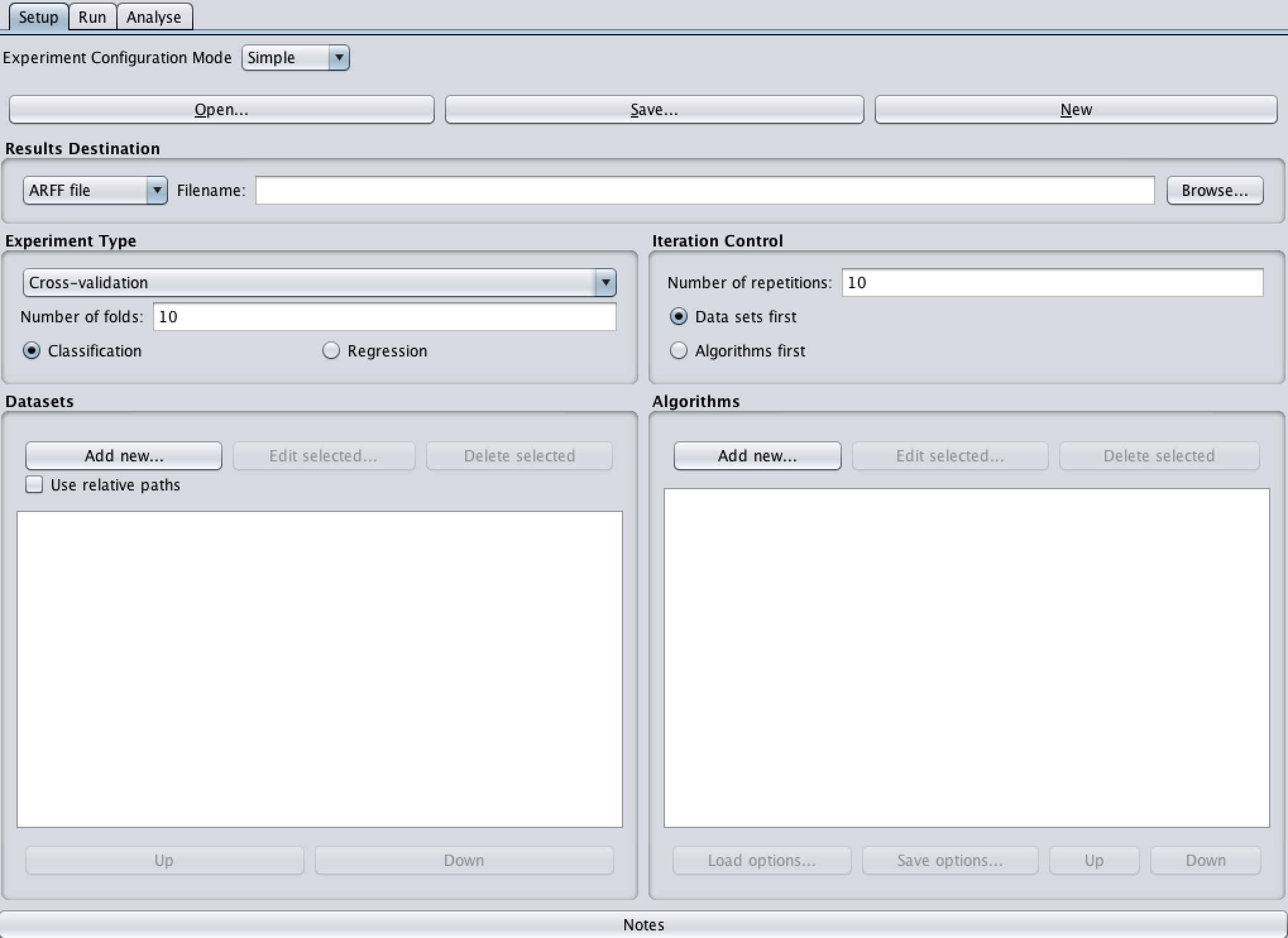
****

Figure 7: Experimenter Window

***Step-2*** *:* Click *New* to start a new experiment and after that we can set the default parameters for an Experiment.

1. **Result Destination:** By default, ARFF is the file format for result. But we can choose between

-ARFF file

-CSV file

-JDBC database

We, select the CSV file option. If the file name is left empty a temporary file will be created in the TEMP directory of the system or we can specify an explicit results file, by clicking on Browse and choosing a filename ‘RawResults.csv’ or by simply entering RawResults in the textbox. The advantage of ARFF or CSV files is that they can be created without any additional classes besides the ones from Weka. Also, ARFF type reduces the size of dataset files filters and their classifiers and allows the use of sparse data directly, which can make them very fast and easy to load.

The drawback is the lack of the ability to resume an experiment when interrupted, e.g., due to an error or the addition of dataset or algorithms. Especially with time-consuming experiments, this behavior can be annoying [7].

(b) **Experiment Type:** We can choose between the following three types:

1. *Cross -Validation :* performs stratified cross-validation with the given number of folds.
2. *Train/Test Percentage split (data randomized):* splits a dataset according to the given percentage into a train and a test file ,after the order of the data has been randomized and stratified [7].
3. *Train/Test Percentage split (order preserved):* We cannot specify explicit training and test files in the Experimenter so, this type unmerge previously merged train and test file into the two original files (we only needs to find out the correct percentage)

We set this parameter, to Cross-Validation with number of folds equal to 10.

(c) **Datasets:**

By clicking on ‘***Add new****’*, we can add dataset files either with an absolute path or with a relative one. We set this parameter to Contact-lens.arff.

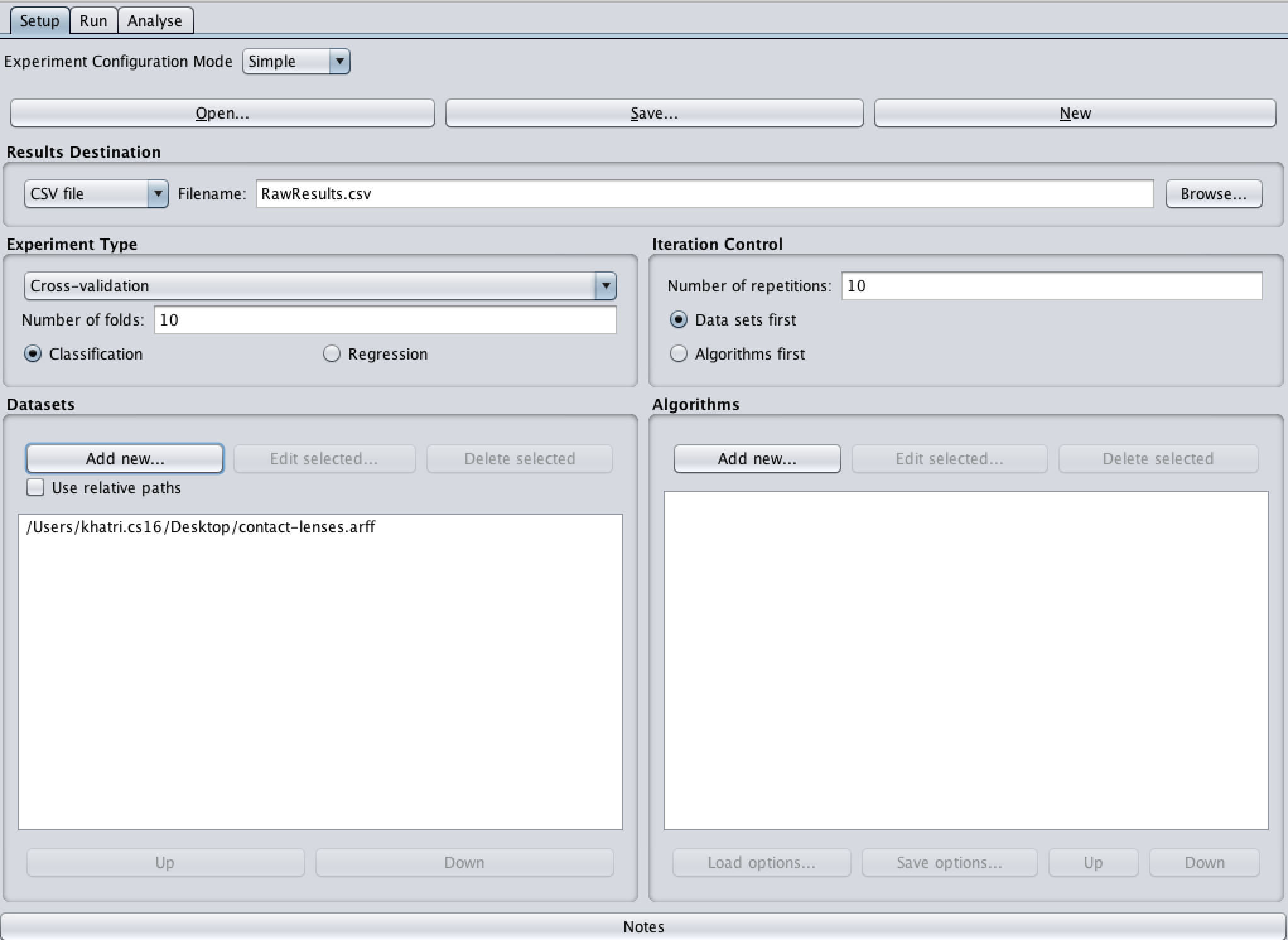


Figure 8: Loading the dataset in to Experimenter

(d) **Iteration Control:**

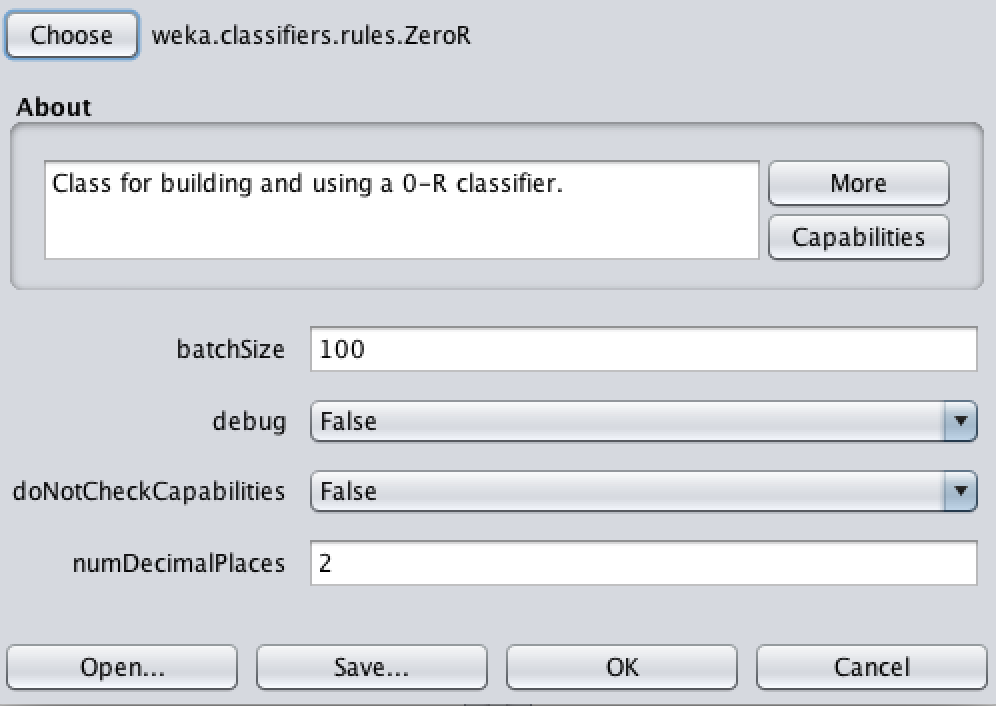
* *Number of Repetitions***:** In order to get statistically meaningful results, the default number of iterations is 10. In case of 10-fold cross-validation this means 100 calls of one classifier with training data and tested against test data [7].

* *Data set First :* In case of more than one dataset and algorithm. We select this option, if we want to iterate over data first.
* *Algorithms set First :* In case of more than one dataset and algorithm. We select this option, to switch from datasets being iterated over first to algorithms. This is the case if one stores the results in a database and wants to complete the results for all the datasets for one algorithm as early as possible [7].

In our case, we set it to default parameters which is 10 , number of iterations and Data set first.

(e) **Algorithms:**

New algorithms can be added via the *Add new* button. Clicking on it , will open a dialog box [7], which is set to the method ‘*Zero R’* by default.

**** Figure 9: Dialog box with ZeroR set to default

With the *Choose* button in fig 9., we can open the *GenericObjectEditor* and choose another classifier as shown in fig 10. After setting the classifier parameters, we click on OK to add it to the list of algorithms. We can multiple algorithms at the same type by using ‘Add new’ button repeatedly. In our case, we add OneR, J48 and Naive Bayes, as shown in fig 11.

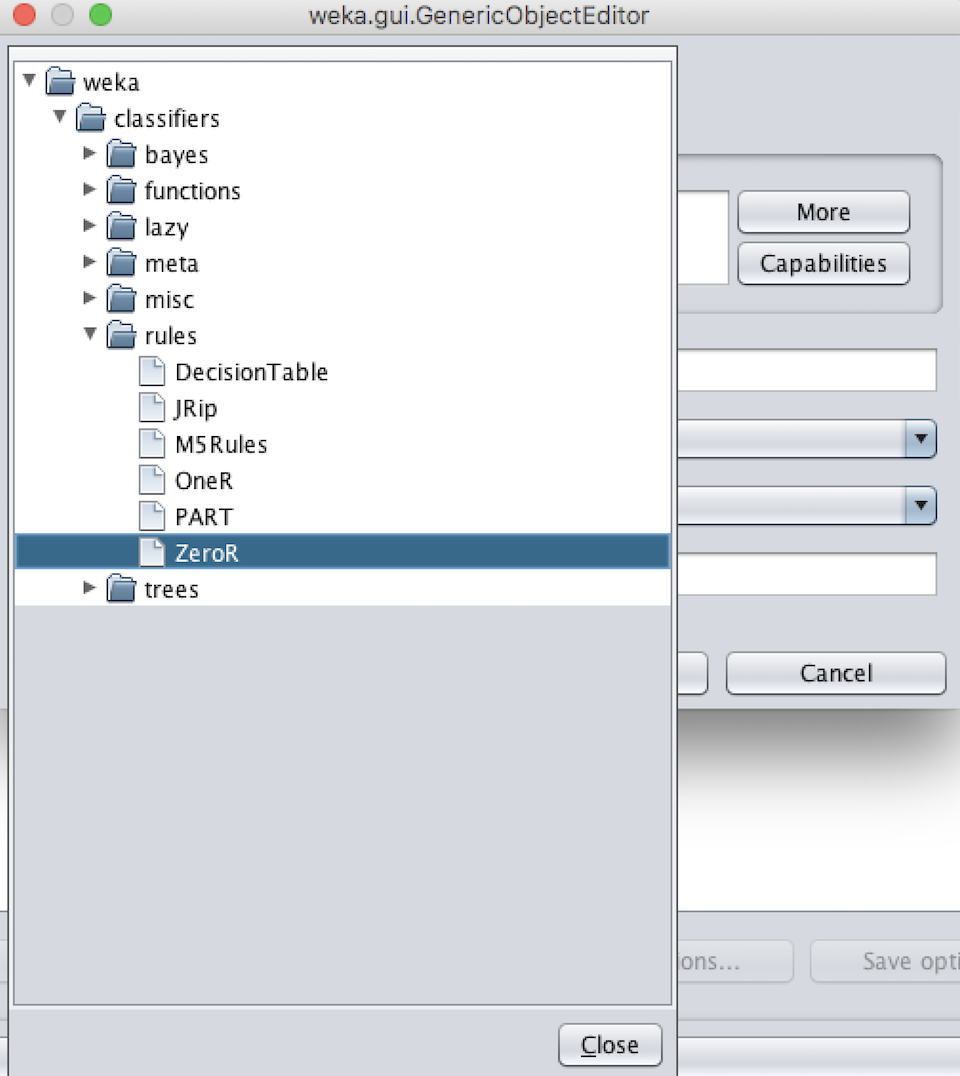


Figure 10: GenericObjectEditor

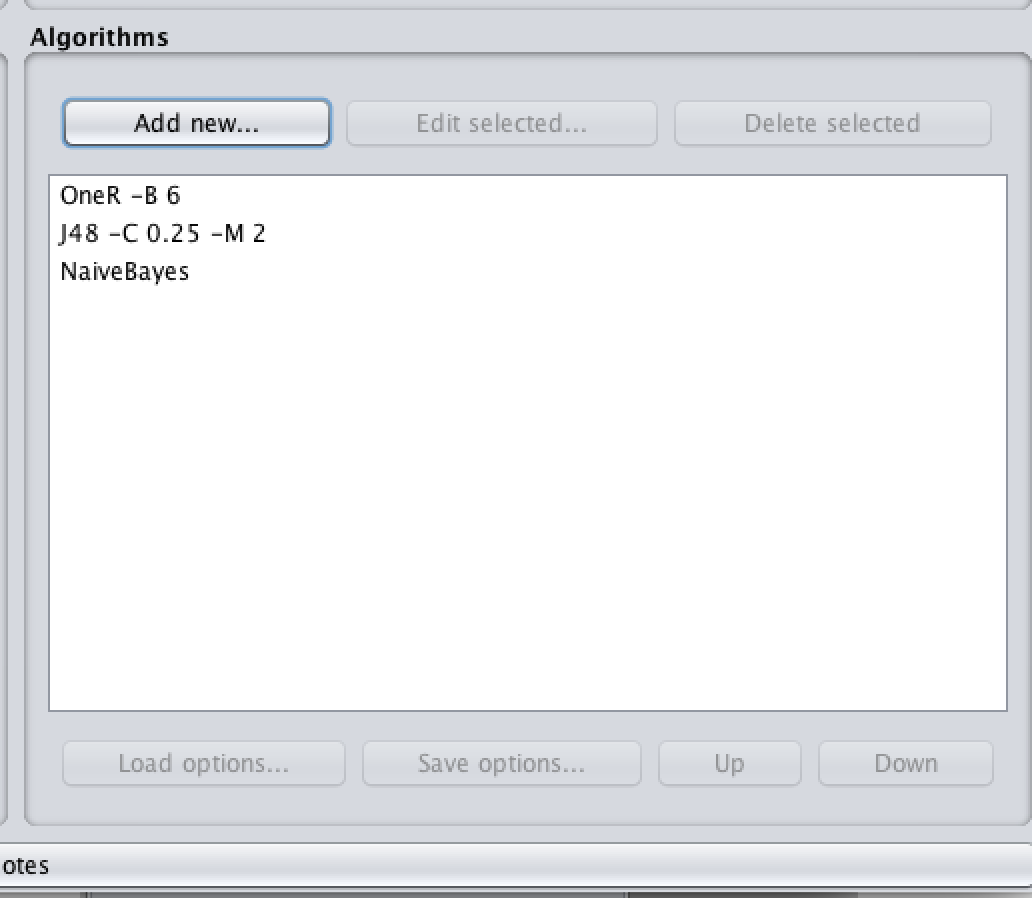


Figure 11. Adding algorithms using ‘Add new’ button

**Step-3:** To run the current experiment, click the Run tab at the top of the Experiment Environment window and we can see the interface fig 12. The current experiment performs 10 runs of 10-fold stratified cross-validation on the Contact-lens dataset using the OneR, J48 and Naive Bayes scheme.

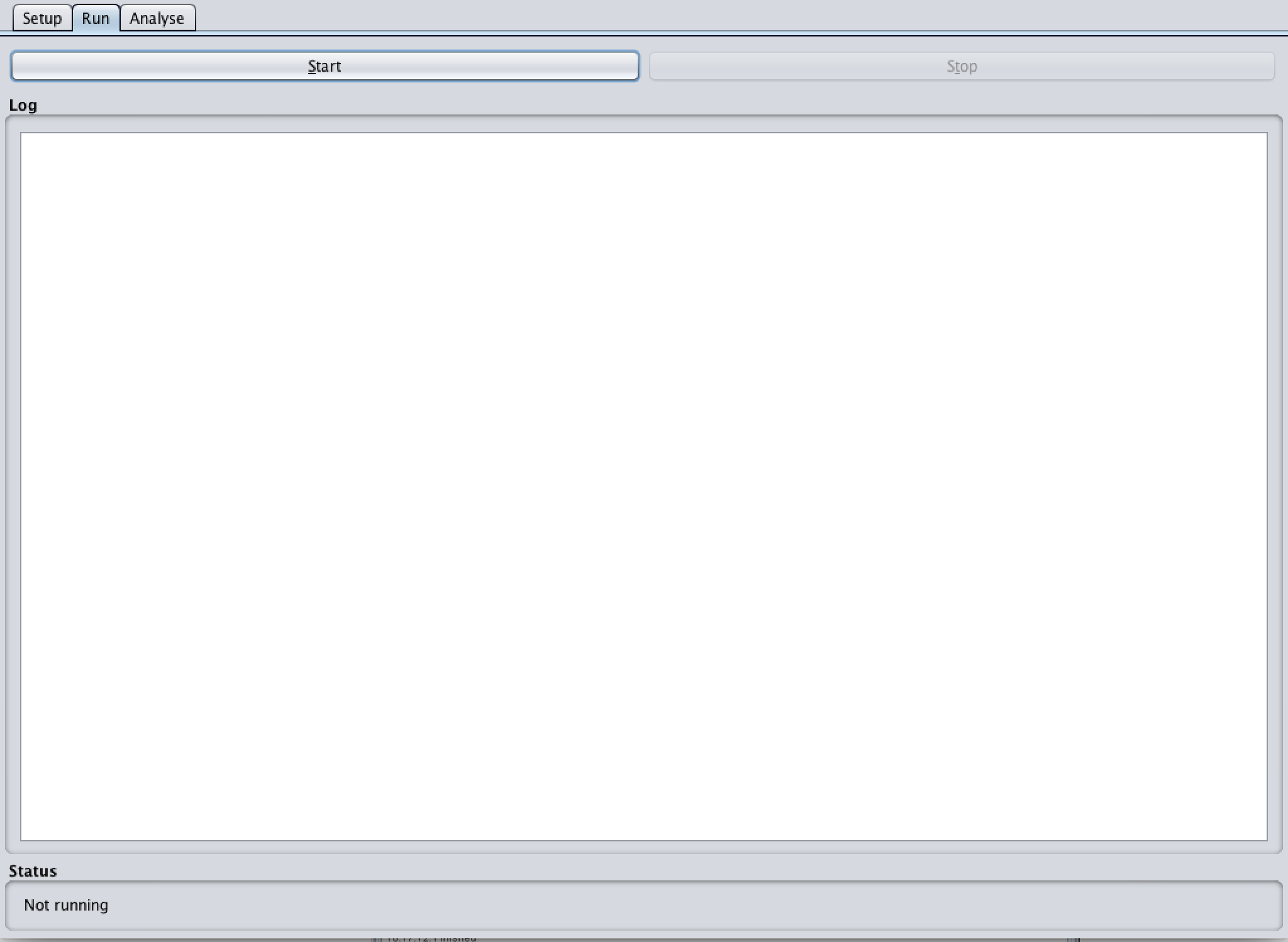


Figure 12. Run Interface

Now, we click *‘Start’* to run the experiment. If the experiment was defined correctly, the 3 messages shown in fig 13 will be displayed in the Log panel. The results of the experiment are saved to the file ‘RawResults.csv’.

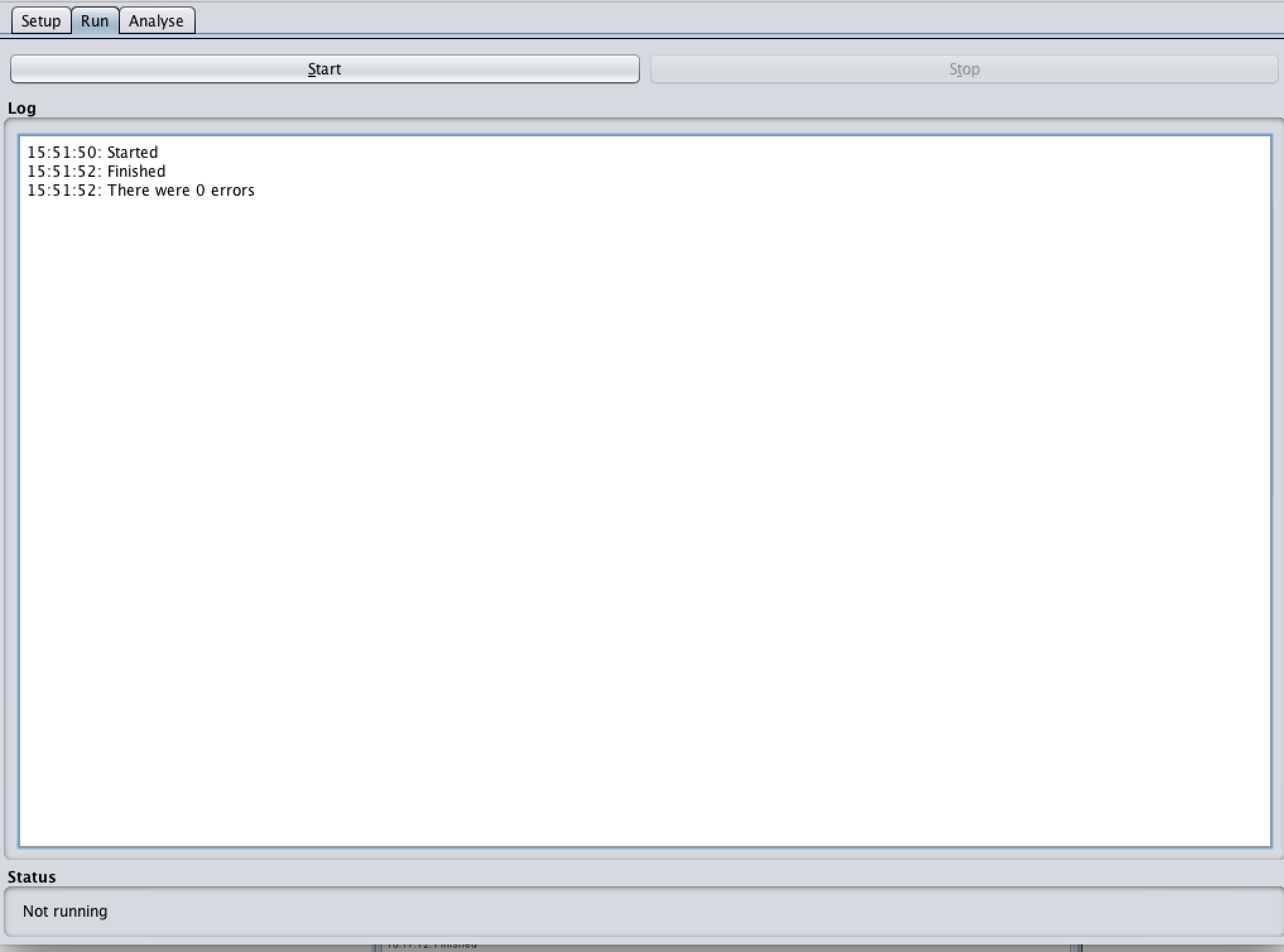


Figure 13. After running the experiment

**Results Evaluation:**

**1) Which classification method typically gives the highest accuracy?**

From our experiments and observations, we noticed that the classification method that usually gives the highest accuracy is Ibk. If we average it out, Ibk performs the best (85.1%)relative to the other methods. We also observed that Ibk classification method gives ~100% accuracy on the training set for both datasets. This is obvious, as IBK looks for the nearest neighbour of instances in the training set and if we evaluate using the training set the instances would exactly match. IBK is a domain sensitive algorithm, it will perform quite well, where the attributes in the neighborhood have similar characteristics.

Further, we know that the ZeroR classification method gives baseline accuracy for a dataset and IBK performs really well in comparison to that From our results of the contact-lens.arff dataset, the baseline accuracy is 62.5%, and IBK gives ~96% for cross validation and training set evaluation methods.

**2) Which method does not perform well and why?**

The method that does not perform well is ZeroR. We took averages for all the evaluation methods for same classification method for each data set . After that we took average of results for each classification method for both the data sets, and the average for the ZeroR method was 43.09%, which was the lowest. This might be because the ZeroR gives the baseline accuracy by classifying the instances into the majority class label. The given datasets are small and there might be a case that, the difference between the class label with majority of instances and other class labels is less.

**3) Why did we use discretization of the “iris” data set?**

We had to discretize the “iris” data set because all of its attributes are continuous except the class attribute. Moreover, many classification methods work well on datasets with nominal attributes. For instance, decision tree algorithms like J48, typically divide the values of attribute into different branches of the tree, therefore with continuous value attributes, it will be difficult to find a good partition and thus, this will affect the accuracy. Also when we discretize the attributes, the J48 classifier runs faster as it doesn’t need to decide the split point for continuous valued attributes.

For the iris.arff dataset, we had to discretize all the non-class attributes in Weka. Figures 14 and 15 represent the discretization process for iris dataset in weka. We discretize the continuous valued attributes into 10 equal-width bins by using the ‘Filters’ in the ‘preprocess’ window of Weka. We need to select ‘filters’ → ‘Unsupervised’ → ‘attribute’ → ‘Discretize’ and then a window as appears to change the parameters, we use the default parameters and make sure that all non-class attributes are nominal.

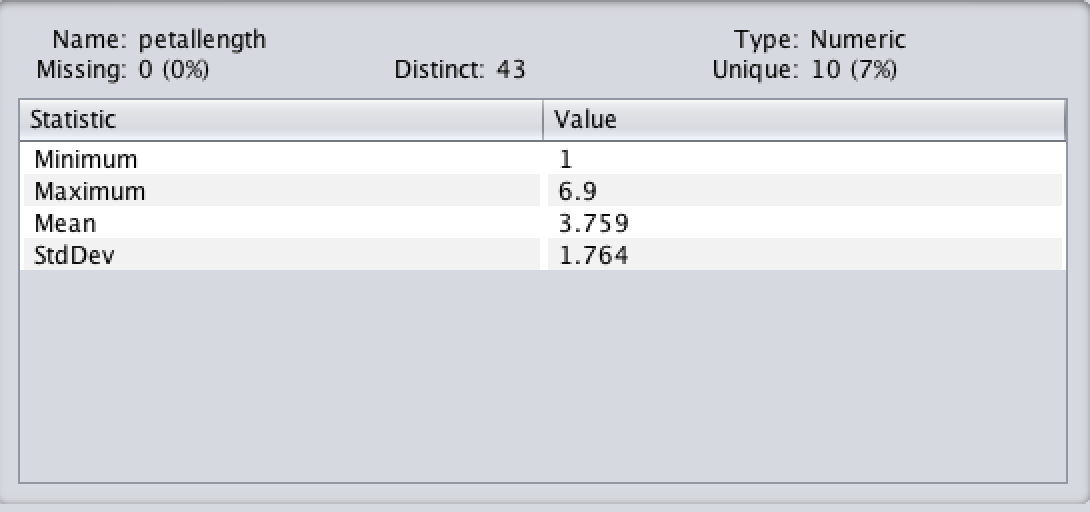


Figure 14: Numerical attribute before discretization.

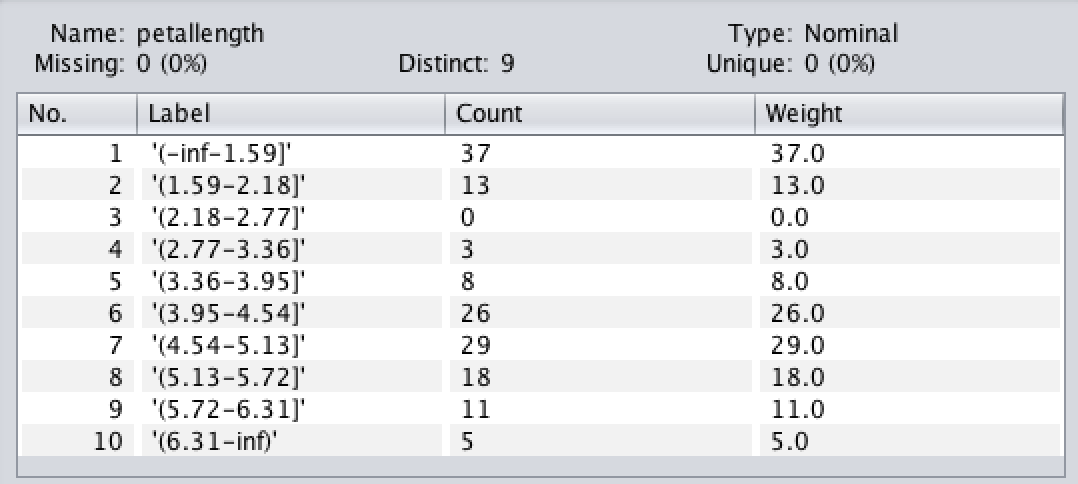


Figure 15: Nominal attribute after discretization.

**4) Does discretization and its method affect the results of classification and how?**

The goal of discretization is to reduce the number of values a continuous attribute assumes by grouping them into intervals or bins.

After running our experiments with and without discretization, we noticed that discretization does sometimes affect the results of classification. For J48, the number of leaves, confusion matrix, and partitions were different. However, for some cases, it was exactly the same. As an example, when we ran the J48 algorithm with cross-validation, it was different, but for percentage split, it was the same.

For other algorithms, such as OneR, we observed similar results with and without discretization. We think that the classification results without discretization is better in most cases because the dataset were small and the model has much more flexibility about where to split the attribute to make it nominal and this might help in achieving better accuracy.

**5) Which of the three evaluation methods overestimates the accuracy and why?**

Out of the three evaluation methods, we believe the**supplied*****training set evaluation method*** overestimates the accuracy. We believe this is because the model is trained using the training data set by pairing the input with expected output and while evaluating on the same data set, it maps the input to output with a higher accuracy than what it might give on the new data set. In addition, if we take the average for evaluation method supplied training set for all the classification methods on Contact-Lens.arff (84.16%) and iris.arff (84.13%) individually, we notice that the average is higher than the other two evaluation methods. Also, we can say that the model evaluated using training set overfits the data and doesn’t generalize well on previously unseen data.

**6) Which of the three evaluation methods underestimates the accuracy and why?**

Out of the three evaluation methods, **the percentage split evaluation** method underestimates the accuracy. We believe this is because there are less number of instances being used for training the classifier, in comparison to other two evaluation methods. Further, as we know that in percentage split method, the data set is split into two sets: (a) training set with 66% of the original dataset and (b) test set with remaining data. The training set is used to train the data while test set is used to test the model build using the training set.

In addition, if we take the average for percentage split evaluation method for all the classification methods on contact-Lens.arff (35%) and iris.arff (80.39%) individually, we notice that the average is lower than the other two evaluation methods. We note that for contact-lens.arff the average is much smaller than iris.arff. This might be because it is a much smaller dataset. Therefore, we cannot build a strong classifier using such a small number of instances. Also, if we look at the accuracy for ZeroR classifier on contact-lenses dataset using percentage split, it is the lowest because when we split the data, the first 2/3rd data consists mostly of instances that are classified as none, whereas, if we look at the other 1/3rd (test set) we notice that ZeroR classifies all the test data (8 instances) as none which gives us a less accuracy.

**References**

[1] Dr. Samal’s classification lecture slides from CSCE 874 course at the University of Nebraska Lincoln.

[2] S. Sayad, “ZeroR,” *chem-eng*. [Online]. Available: http://chem-eng.utoronto.ca/~datamining/dmc/zeror.htm. [Accessed: 18-Mar-2018].

[3] S. Sayad, “OneR,” *OneR*. [Online]. Available: http://www.saedsayad.com/oner.htm. [Accessed: 18-Mar-2018].

[4] “Naive Bayes classifier,” *Wikipedia - Naive Bayes Classifier*, 16-Mar-2018. [Online]. Available: https://en.wikipedia.org/wiki/Naive\_Bayes\_classifier. [Accessed: 18-Mar-2018].

[5] “Classification Methods,” *CLASSIFICATION METHODS*. [Online]. Available: http://www.d.umn.edu/~padhy005/Chapter5.html. [Accessed: 18-Mar-2018].

[6] “K-nearest neighbors algorithm,” *Wikipedia*, 28-Feb-2018. [Online]. Available: https://en.wikipedia.org/wiki/K-nearest\_neighbors\_algorithm. [Accessed: 18-Mar-2018].

[7] “Weka Experimenter Tutorial,” *EECS*. [Online]. Available: https://www.eecs.yorku.ca/course\_archive/2006-07/W/4412/doc/weka/ExperimenterTutorial-3.5.5.pdf. [Accessed: 18-Mar-2018].