

EECS 349 Problem Set 1

Due 11:59PM Tuesday, Apr 12

Updated Mar 30 21:30:00 CDT 2016

This assignment consists of two parts. A Python programming warm-up, and some machine learning experiments with Weka.

Submission Instructions

You'll turn in your homework as a single zip file, in Canvas. Specifically:

1. Create a text file with your completed code for the Python warm-up below. Name this file `ps1.py`.
2. Create a *text* file with your answers to the [questions](#) below. Be sure you have answered all the questions. Name this file `ps1.txt`.
3. Create a file containing your Weka model ([instructions](#) below). Be sure this file can be loaded into Weka and that it runs. Name this file `ps1.model`.
4. Create a text file in ARFF format with your predicted labels for the test set ([instructions](#) below). Name this file `ps1.arff`.
5. Create a single ZIP file containing:
 - o `ps1.py`
 - o `ps1.txt`
 - o `ps1.model`
 - o `ps1.arff`
6. Turn the zip file in under Problem Set 1 in Canvas.

Python warm-up (2 points)

Complete the three functions in [node_hw1.py](#), using the comments in the file as a guide for what to do. (you can also define helper functions with other names, but make sure you don't change the names of the three functions you're asked to implement). For convenience, the "tester" function provides a rudimentary test of each method. Use Python 2.7. Turn in your .py file in Canvas, as per the submission instructions specified above.

Weka Experiments (8 points)

In this part of the assignment you will run a machine learning experiment using [Weka](#), an open source framework for machine learning and data mining. You will generate a model that predicts the quality of wine based on its chemical attributes. You will train the model on the supplied training data and use the model to predict the correct output for unlabeled test data.

Download and Install Weka

Weka is available for Windows, Mac, and Linux from <http://www.cs.waikato.ac.nz/ml/weka/>. Click on the "Download" link on the left-hand side and download the Stable GUI version, which is currently 3.6. You may also wish to download a Weka manual from the "Documentation" page.

Note

Some points that may help with Weka installation on Mac OS X:

1. The Weka download page lists a version that works with JVM 1.6 and one that comes bundled with JVM 1.7. If you have a JVM version higher than 1.7 already installed on your Mac, you should download the Weka version bundled with JVM 1.7.
2. Installing Weka on a Mac simply requires copying the downloaded contents to a new folder under *Applications*. In other words, there is no .dmg file to run.

Download the Dataset

The dataset files are here:

- [train.arff](#) (labeled training set, 1890 instances)
- [test.arff](#) (unlabeled test set, 810 instances)

This dataset is adapted from:

P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. *Modeling wine preferences by data mining from physicochemical properties*. In Decision Support Systems, Elsevier, 47(4):547-553. ISSN: 0167-9236.

This dataset contains data for 2700 white variants of the Portuguese "Vinho Verde" wine. For each variant, 11 chemical features were measured. Each of these is a numeric attribute. They are:

- fixed acidity
- volatile acidity
- citric acid
- residual sugar
- chlorides numeric
- free sulfur dioxide
- total sulfur dioxide
- density
- pH
- sulphates
- alcohol

Each variant was tasted by three experts. Their ratings have been combined into a single quality label: "good" or "bad" Therefore this is a *binary classification* problem with *numeric attributes*.

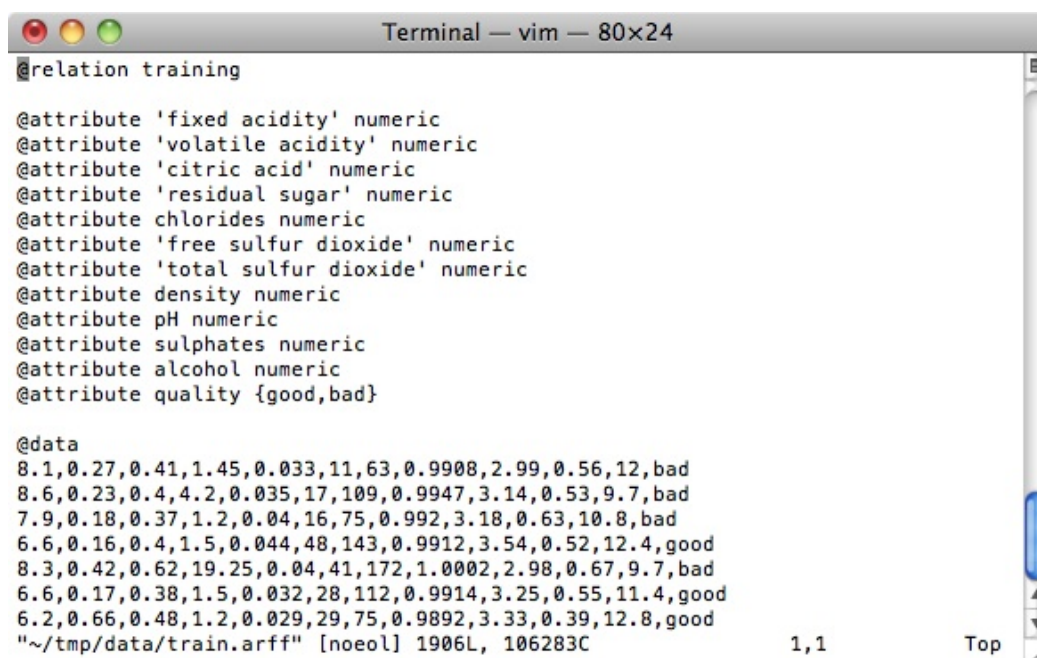
The dataset has been randomly split into a training set (1890 variants) and a test set (810 variants). The training set contains both chemical features and quality labels. The test set contains only the chemical features.

Examine the Data

It is a good idea to inspect your data by hand before running any machine learning experiments, to ensure that the dataset is in the correct format and that you understand what the dataset contains. The following sections will familiarize you with the data and introduce some tools in Weka.

The ARFF Format

View `train.arff` and `test.arff` in a text editor. You should see something like this:



```
Terminal — vim — 80x24
@relation training

@attribute 'fixed acidity' numeric
@attribute 'volatile acidity' numeric
@attribute 'citric acid' numeric
@attribute 'residual sugar' numeric
@attribute chlorides numeric
@attribute 'free sulfur dioxide' numeric
@attribute 'total sulfur dioxide' numeric
@attribute density numeric
@attribute pH numeric
@attribute sulphates numeric
@attribute alcohol numeric
@attribute quality {good,bad}

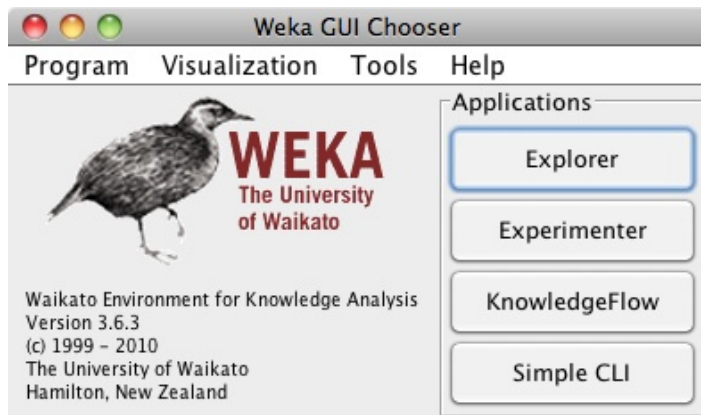
@data
8.1,0.27,0.41,1.45,0.033,11,63,0.9908,2.99,0.56,12,bad
8.6,0.23,0.4,4.2,0.035,17,109,0.9947,3.14,0.53,9.7,bad
7.9,0.18,0.37,1.2,0.04,16,75,0.992,3.18,0.63,10.8,bad
6.6,0.16,0.4,1.5,0.044,48,143,0.9912,3.54,0.52,12.4,good
8.3,0.42,0.62,19.25,0.04,41,172,1.0002,2.98,0.67,9.7,bad
6.6,0.17,0.38,1.5,0.032,28,112,0.9914,3.25,0.55,11.4,good
6.2,0.66,0.48,1.2,0.029,29,75,0.9892,3.33,0.39,12.8,good
~/tmp/data/train.arff" [noeol] 1906L, 106283C 1,1 Top
```

The files are in ARFF (Attribute-Relation File Format), a text format developed for Weka. At the top of each file you will see a list of attributes, followed by a data section with rows of comma separated values, one for each instance. The text and training files look similar, except that the last value for each training instance is a quality label and the last value for each test instance is a question mark, since these instances are unlabeled.

For this assignment you will not need to deal with the ARFF format directly, as Weka will handle reading and writing ARFF files for you. In future experiments you may have to convert between ARFF and another data format. (You can close the text editor.)

The Weka ARFF Viewer

Run Weka. You will get a screen like the following:



From the *Tools* menu choose *ArffViewer*. In the window that opens, choose *File*→*Open* and open one of the data files. You should see something like the following (see [important note](#) below):

No.	fixed acidity Numeric	volatile acidity Numeric	citric acid Numeric	residual sugar Numeric	chlorides Numeric	free sulfur dioxide Numeric	total sulfur dioxide Numeric	density Numeric	pH Numeric
1	8.1	0.27	0.41	1.45	0.033	11.0	63.0	0.9908	2.99
2	8.6	0.23	0.4	4.2	0.035	17.0	109.0	0.9947	3.14
3	7.9	0.18	0.37	1.2	0.04	16.0	75.0	0.992	3.18
4	6.6	0.16	0.4	1.5	0.044	48.0	143.0	0.9912	3.54
5	8.3	0.42	0.62	19.25	0.04	41.0	172.0	1.0002	2.98
6	6.6	0.17	0.38	1.5	0.032	28.0	112.0	0.9914	3.25
7	6.2	0.66	0.48	1.2	0.029	29.0	75.0	0.9892	3.33
8	6.5	0.31	0.14	7.5	0.044	34.0	133.0	0.9955	3.22
9	6.2	0.66	0.48	1.2	0.029	29.0	75.0	0.9892	3.33
10	6.4	0.31	0.38	2.9	0.038	19.0	102.0	0.9912	3.17
11	6.8	0.26	0.42	1.7	0.049	41.0	122.0	0.993	3.47
12	7.6	0.67	0.14	1.5	0.074	25.0	168.0	0.9937	3.05
13	7.2	0.32	0.36	2.0	0.033	37.0	114.0	0.9906	3.1
14	5.8	0.27	0.2	14.95	0.044	22.0	179.0	0.9962	3.37
15	7.3	0.28	0.43	1.7	0.08	21.0	123.0	0.9905	3.19
16	6.5	0.39	0.23	5.4	0.051	25.0	149.0	0.9934	3.24
17	7.3	0.24	0.39	17.95	0.057	45.0	149.0	0.9999	3.21
18	7.3	0.24	0.39	17.95	0.057	45.0	149.0	0.9999	3.21
19	7.4	0.18	0.31	1.4	0.058	38.0	167.0	0.9931	3.16
20	6.2	0.45	0.26	4.4	0.063	63.0	206.0	0.994	3.27
21	6.2	0.46	0.25	4.4	0.066	62.0	207.0	0.9939	3.25
22	6.9	0.19	0.35	5.0	0.067	32.0	150.0	0.995	3.36
23	6.6	0.25	0.29	1.1	0.068	39.0	124.0	0.9914	3.34
24	6.2	0.16	0.33	1.1	0.057	21.0	82.0	0.991	3.32
25	7.0	0.47	0.07	1.1	0.035	17.0	151.0	0.991	3.02
26	6.2	0.35	0.03	1.2	0.064	29.0	120.0	0.9934	3.22
27	6.4	0.26	0.24	6.4	0.04	27.0	124.0	0.9903	3.22
28	6.7	0.25	0.13	1.2	0.041	81.0	174.0	0.992	3.14

Here you see the same data as in the text editor, but parsed into a spreadsheet-like format. Although you will not need the ArffViewer for this assignment, it is a useful tool to know about when working with Weka. (You can close the ArffViewer window.)

Important Note

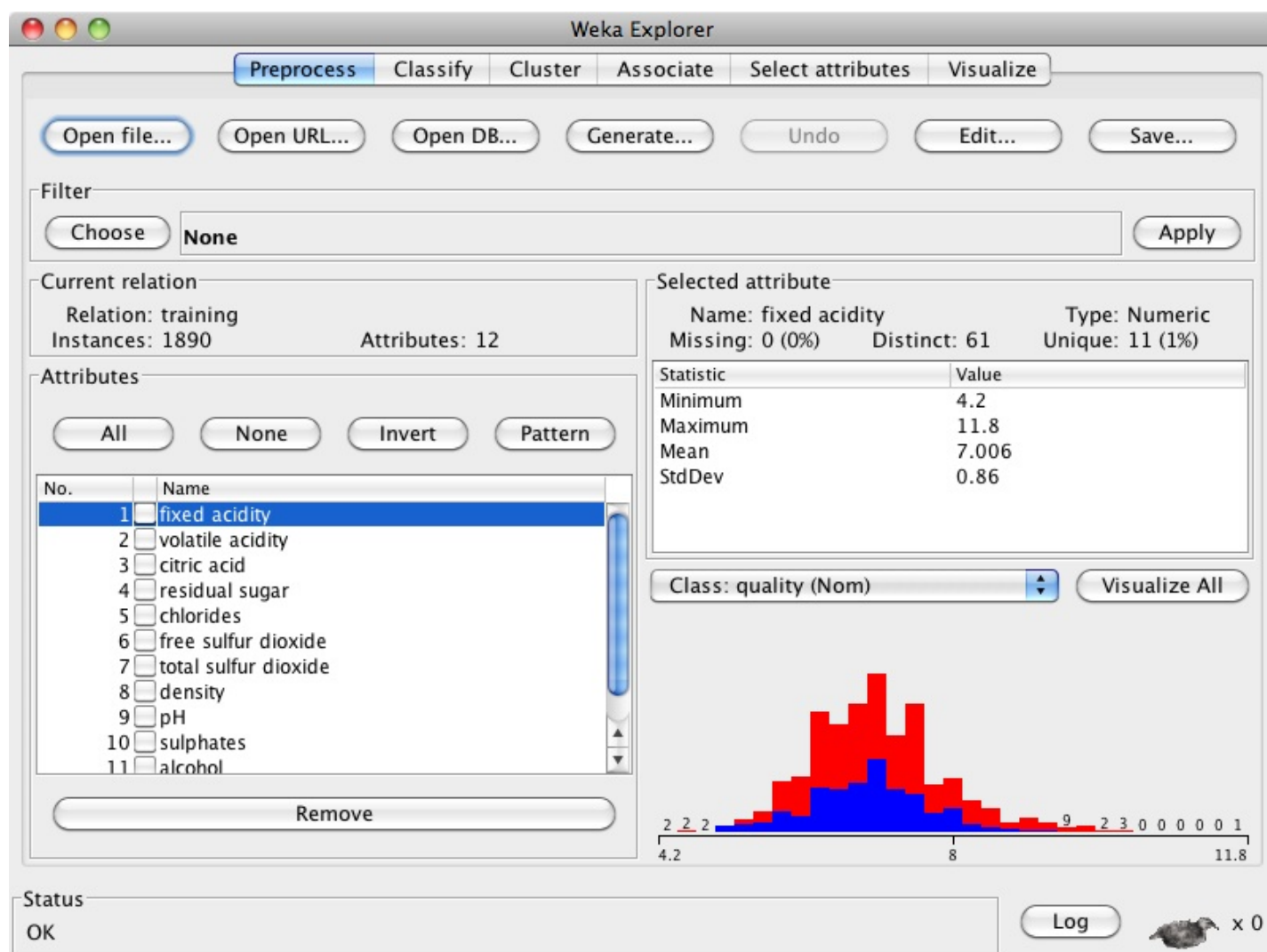
You may find that the ARFF files are grayed out and that the *All Files* option needs to be selected from the *File Format* dropdown menu for the files to be selectable. However, the ARFF Viewer may still not read the files properly. If such is the case, it is likely that a `.txt` extension got appended to the filename when the files were downloaded. However, even if the files are downloaded without `.txt` getting appended or an inadvertently added `.txt`

extension is removed, the ARFF Viewer may have trouble reading the files properly. The following steps should resolve the issue:

1. View the ARFF in your Web browser by clicking on the link in the instructions or open the downloaded ARFF file in a text editor.
2. Copy all the text and paste it to a new text file.
3. If you copied the ARFF contents from the downloaded ARFF file, it is recommended that you do not overwrite the downloaded ARFF file when saving the new file on the next step. Instead, delete the downloaded ARFF file.
4. Save the new text file with a **.arff** extension, carefully making sure that a **.txt** extension does not get appended.
5. Open the newly saved ARFF file in the Weka ARFF Viewer to verify the Viewer can display the file in the manner illustrated in the image above.

The Weka Explorer

From the Weka GUI Choose click on the *Explorer* button to open the Weka Explorer. The Explorer is the main tool in Weka, and the one you are most likely to work with when setting up an experiment. For the remainder of this assignment you will work within the Weka Explorer. The Explorer should open to the "Preprocess" tab. The Preprocess tab allows you to inspect and modify your dataset before passing it to a machine learning algorithm. Click on the button that says "Open file..." and open `train.arff`. You should see something like this:



The attributes are listed in the bottom left, and summary statistics for the currently selected attribute are shown on the right side, along with a histogram. Click on each attribute (or use the down arrow key to move through them) and look at the corresponding histogram. You will notice that many numeric attributes have a "hump" shape; this is a common pattern for numeric attributes drawn from real-world data.

You will also notice that some attributes appear to have outliers on one or both sides of the distribution. The proper treatment of outliers varies from one experiment to another. For this assignment you can leave the outliers alone.

Now answer [Question #1](#).

Classifier Basics

In this section you will see how to train a classifier on the data.

Baseline Classifier

Click on the "Classify" tab. Choose *ZeroR* as the Classifier if it is not already chosen (it is under the "rules" subtree when you click on the "Choose" button). When used in a classification problem, *ZeroR* simply chooses the majority class. Under "Test options" select "Use training set", then click the "Start" button to run the classifier. You should see something like this:

The screenshot shows the Weka Explorer window with the 'Classify' tab selected. The 'Classifier' dropdown is set to 'ZeroR'. Under 'Test options', 'Use training set' is selected. The 'Start' button has been clicked, and the 'Classifier output' pane displays the following information:

zeroR predicts class value: bad
Time taken to build model: 0 seconds

=== Evaluation on training set ===
=== Summary ===

Metric	Value	Percentage
Correctly Classified Instances	1179	62.381 %
Incorrectly Classified Instances	711	37.619 %
Kappa statistic	0	
Mean absolute error	0.4694	
Root mean squared error	0.4844	
Relative absolute error	100	%
Root relative squared error	100	%
Total Number of Instances	1890	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC
0	0	0	0	0	0	0
1	1	1	0.624	1	0.768	0
Weighted Avg.	0.624	0.624	0.389	0.624	0.479	0

=== Confusion Matrix ===

a \ b	0	1
0	711	0
1	0	1179

--- classified as ---
a = good
b = bad

The 'Result list' on the left shows '02:07:30 - rules.ZeroR' as the selected result. The 'Status' bar at the bottom indicates 'OK'.

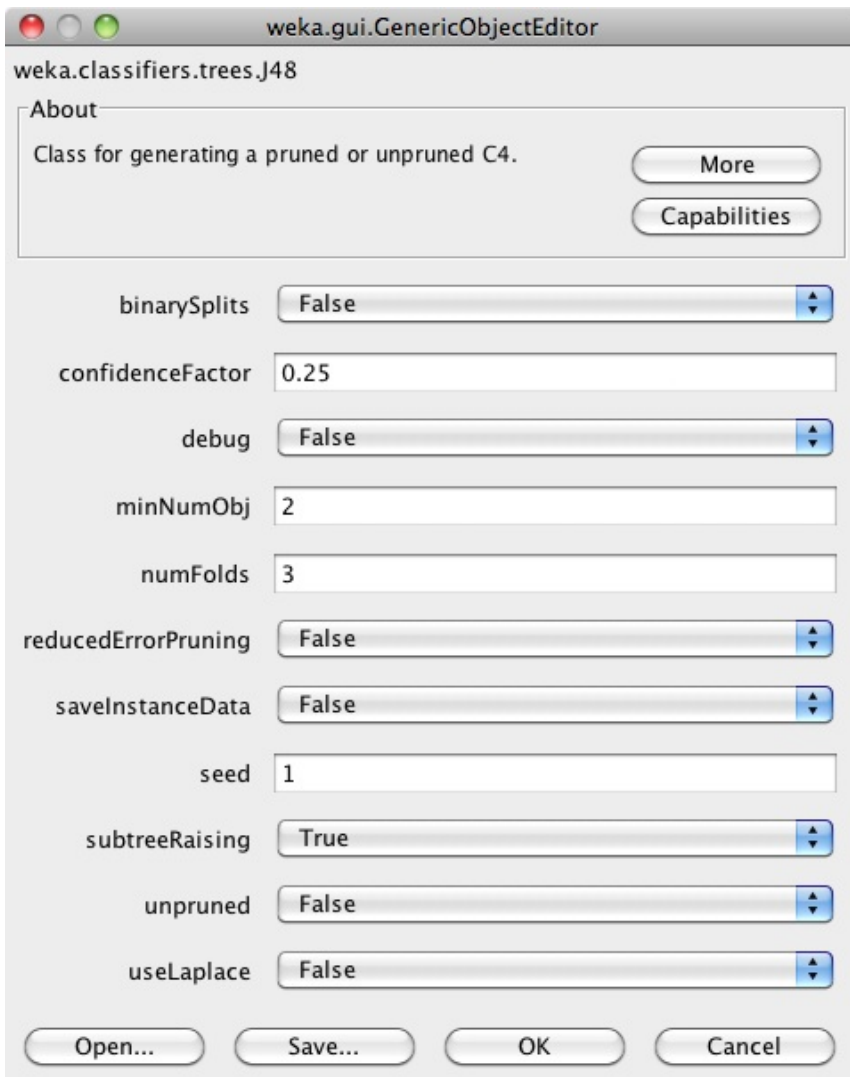
The classifier output pane displays information about the model created by the classifier as well as the evaluated performance of the model. In the Summary section, the row "Correctly Classified Instances" reports the accuracy of the model.

Now answer [Question #2](#).

Decision Trees

J48 is the Weka implementation of the C4.5 decision tree algorithm.

Click on the "Choose" button and select *J48* under the "trees" section. Notice that the field to the right of the "Choose" button updates to say "J48 -C 0.25 -M 2". This is a command-line representation of the current settings of *J48*. Click on this field to open up the configuration dialog for *J48*.



Each classifier has a configuration dialog such as this that shows the parameters of the algorithm as well as buttons at the top for more information. When you change the settings and close the dialog, the command line representation updates accordingly. For now we will use the default settings, so hit "Cancel" to close the dialog.

Under "Test options" select "Use training set", then click the "Start" button to run the classifier. After the classifier finishes, scroll up in the output pane. You should see a textual representation of the generated decision tree.

Now answer [Question #3](#).

Scroll back down and record the percentage of Correctly Classified Instances. Now, under "Test options", select "Cross-validation" with 10 folds. Run the classifier again and record the percentage of Correctly Classified Instances.

In both cases, the final model that is generated is based on all of the training data. The difference is in how the accuracy of that model is estimated.

Now answer [Question #4](#).

Build Your Own Classifier

This is the main part of the assignment. Search through the classifiers in Weka and run some of them on the training set. You may want to try varying some of the classifier parameters as well. Choose the one you feel is most likely to generalize well to unseen examples--namely the unlabeled examples in the test set. Feel free to use validation strategies other than 10-fold cross-validation.

When you have built the classifier you want to submit, move on to the following sections.

Saving the Model

To export a classifier model you have built:

1. Right-click on the model in the "Result list" in the bottom left corner of the Classify tab.
2. Select "Save model".
3. In the dialog that opens, ensure that the File Format is "Model object files"
4. Save the model using the naming convention given in the [submission instructions](#) (e.g. ps1.model).

In order to grade your assignment it must be possible to load your model file in Weka and run it on a labeled version of `test.arff`. You can load your model by right-clicking in the Result list pane and selecting "Load model".

Generating Predictions

To generate an ARFF file with predictions for the test data, perform the following steps from within the Classify tab:

1. Under "Test options" select "Supplied test set".
2. Click on the "Set..." button.
3. In the "Test Instances" dialog that opens click "Open file...".
4. Open `test.arff`.
5. Close the Test Instances dialog.
6. Right-click on your model in the Result list and select "Re-evaluate model on current test set". Your output will look something like the picture below. Notice that the output contains a bunch of NaNs. This is because the test data is unlabeled and therefore Weka cannot compute the accuracy.
7. Right-click again on your model and select "Visualize classifier errors".
8. In the dialog that opens, click on the "Save" button.
9. Save the ARFF file using the naming convention given in the [submission instructions](#) (e.g. ps1.arff).

Weka Explorer

Preprocess **Classify** Cluster Associate Select attributes Visualize

Classifier
Choose **J48 -C 0.25 -M 2**

Test options
☐ Use training set
☒ Supplied test set **Set...**
☐ Cross-validation Folds **10**
☐ Percentage split % **66**
More options...

Classifier output

```

a b <-- classified as
586 125 | a = good
140 1039 | b = bad

=== Re-evaluation on test set ===

User supplied test set
Relation: test
Instances: unknown (yet). Reading incrementally
Attributes: 12

=== Summary ===

Total Number of Instances          0
Ignored Class Unknown Instances    810

=== Detailed Accuracy By Class ===

          TP Rate  FP Rate  Precision  Recall  F-Measure  ROC
          0         0         0         0         0         ?
          0         0         0         0         0         ?
Weighted Avg.  NaN      NaN      NaN      NaN      NaN      NaN

=== Confusion Matrix ===

a b <-- classified as
0 0 | a = good
0 0 | b = bad

```

Result list (right-click for options)
19:10:52 - trees.J48

Status
OK

Log x 0

Now answer [Questions #5 through #7](#).

Try Another Data Set

You will now build a classifier for a second data set concerning the evaluation of cars, following which you will answer only the last two [questions](#). (You do not have to answer Questions #1 through #8 again.)

In order to answer the questions, perform the following steps:

Download the Car Evaluation Dataset

The car evaluation dataset files are here (see [important note](#) below)::

- [car_train.data](#) (labeled training set, 1190 instances)
- [car_train.names](#) (auxiliary file for training set)
- [car_test.data](#) (unlabeled test set, 510 instances)
- [car_test.names](#) (auxiliary file for test set)

This dataset is adapted from:

Car Evaluation Database, which was derived from a simple hierarchical decision model originally developed for the demonstration of DEX, M. Bohanec, V. Rajkovic: *Expert system for decision making*. Sistemica 1(1), pp. 145-157, 1990.).

Important Note

The main data file for the car evaluation data set ends in a **.data** extension and has an associated auxiliary data file ending in a **.names** extension. However, the usage for the **.data** file is the same as for the **.arff** file you are already familiar with, including the [important note](#) applicable to the wine evaluation data set files, though you have to pay special attention to the following:

1. When opening the main data file (`car_train.data` or `car_test.data`) in the Weka Explorer, the *C4.5 data files (*.data)* option needs to be selected from the *File Format* dropdown menu.
2. The auxiliary data files (`car_train.names` and `car_test.names`) must be located in the same folder as the main data files. (You do not need to take any action on these auxiliary data files other than to keep them in the same folder as the main data files, but inspecting the contents of the files should help you interpret how the main data files work.)

Build Classifiers

You will perform four experiments, measuring the 10-fold cross-validation accuracy of two types of classifiers (call them classifiers A and B) on two data sets (cars and wine). You can choose A and B however you like -- they can be different classifiers (nearest-neighbor vs. decision trees) or the same classifier with different settings (different numbers of nearest neighbors, for example). Your goal is to choose settings such that the A classifier performs great for wine evaluation, but poorly for car evaluation, and vice-versa for classifier B. In other words, you should strive to find a value as large as you can for the expression below:

$$wine_acc(A) + car_acc(B) - wine_acc(B) - car_acc(A)$$

where $wine_acc(A)$ refers to the accuracy of Classifier A on the wine data set, and $car_acc(B)$ refers to the accuracy of Classifier B on the car data set, and so on.

Note: You do not need to obtain the largest possible quantity for the above expression, and it is okay to use classifiers we have discussed in class as long as you can achieve some positive value for the above expression (a value of 2% is sufficient for the assignment).

Note that you will only need to use the training data (`car_train.data`) for this task. The test data (`car_test.data`) is provided for your personal reference, should you choose to try your car evaluation classifier on it to gain experience. Therefore, you do not need to perform the steps under [Generating Predictions](#) for this task.

Now answer the remaining [questions](#).

Questions

Put concise answers to the following questions in a text file, as described in the [submission instructions](#).

1. Which attributes appear to have outliers? (Do not worry too much about being precise here. The point is for you to inspect the data and interpret what you see.)
2. What is the *accuracy* - the percentage of correctly classified instances - achieved by *ZeroR* when you run it on the training set? Explain this number. How is the accuracy of *ZeroR* a helpful baseline for interpreting the performance of other classifiers?

3. Using a decision tree Weka learned over the training set, what is the most informative single feature for this task, and what is its influence on wine quality?
 4. What is 10-fold cross-validation? What is the *main* reason for the difference between the percentage of Correctly Classified Instances when you measured accuracy on the training set itself, versus when you ran 10-fold cross-validation over the training set? Why is cross-validation important?
 5. What is the "command-line" for the model you are submitting? For example, "*J48 -C 0.25 -M 2*". What is the reported accuracy for your model using 10-fold cross-validation?
 6. In a few sentences, describe how you chose the model you are submitting. Be sure to mention your validation strategy and whether you tried varying any of the model parameters.
 7. A Wired article from several years ago on the 'Peta Age' suggests that increasingly huge data sets, coupled with machine learning techniques, makes model building obsolete. In particular it says: *This is a world where massive amounts of data and applied mathematics replace every other tool that might be brought to bear. Out with every theory of human behavior, from linguistics to sociology. Forget taxonomy, ontology, and psychology...* In a short paragraph (about four sentences), state whether you agree with this statement, and why or why not.
 8. Briefly explain what strategy you used to obtain the Classifiers A and B that performed well on one of the car or wine data sets, and not the other.
 9. What is the key difference about the output space for the car task, as compared to the wine task?
-