COMP9417 - Assignment 1: Using Machine Learning Tools

Last revision: Fri Mar 24 09:40:11 AEDT 2017

1 Aims

The aim of this assignment is to enable you to ***apply*** different machine learning algorithms on a variety of datasets and produce a written ***analysis*** of the empirical results, based on your knowledge of machine learning. After completing this assignment you will be able to:

- set up replicated k-fold cross-validation experiments to obtain average performance measures of algorithms on datasets
- compare the performance of different algorithms against a base-line and each other
- aggregate comparative performance scores for algorithms over a range of different datasets
- propose properties of algorithms and their parameters, or datasets, which may lead to performance differences being observed
- suggest reasons for actual observed performance differences in terms of properties of algorithms, parameter settings or datasets.
- apply methods for data transformations and parameter search and evaluate their effects on the performance of algorithms

There are a total of 20 marks available. Each is worth 0.5 course mark, i.e., assignment marks will be scaled to a course mark out of 10 to contribute to the course total.

2 Submission

New Deadline: 23:59:59, Sunday April 9, 2017.

Late penalties: one mark will be deducted from the total for each day late, up to a total of five days. If six or more days late, no marks will be given. Recall the guidance regarding plagiarism in the course introduction: this applies to this assignment and if evidence of plagiarism is detected it may result in penalties ranging from loss of marks to suspension.

2.1 Overview of the assignment

The plan of the assignment is very straightforward. First, you will familiarise yourself with the open-source machine learning toolkit that you will use to complete the activity. Second, you will use the toolkit to configure experiments that will run machine learning algorithms on various real-world datasets. The toolkit will enable you to generate results from these experiments and save them to files which will form the main part of your submission. You will also have to answer a number of questions to check your understanding of what has occurred in the experiments. Your written answers will be saved to a text file which will form the remainder of your submission. You will not need to write any code for this assignment. The remainder of this document contains in Section 3 an introduction to the machine learning toolkit and in Section 4 the specification of the experiments that you will run and the questions to be answered.

2.2 Form of submission

There are 4 (four) questions in the assignment. You should submit 1 (one) ***text file*** containing Weka output for each question (details below), plus 1 (one) ***text file*** containing your written answers for *ALL* questions (details below), a total of 5 (five) files:

```
q1.out
q2.out
q3.out
q4.out
answers.txt
```

Files in any format other than text ***cannot be accepted***.
Submit your files using *give*. On a CSE Linux machine, type the following on the command-line:

```
$ give cs9417 ass1 q1.out q2.out q3.out q4.out answers.txt
```

Alternatively, you can submit using the web interface to `give`.
Please make sure the `answers.txt` file contains the correct numbers for all the questions!
Note: please keep written answers succinct and on point.

3 Getting started with the Weka Toolkit

This introduction is the starting point for Assignment 1, which requires the use of the Weka Machine Learning Toolkit. The aim of the assignment is to enable you to compare the performance of different machine learning algorithms on a varied range of datasets and produce an interpretation of the empirical results, a standard data mining task. By completing this introduction you will become sufficiently familiar with running Weka to complete the assignment. The assignment questions are in Section 4.

3.1 What is Weka ?

The Waikato Environment for Knowledge Analysis (Weka) is an open-source suite of machine learning tools implemented in Java at the University of Waikato in New Zealand. It is similar in design and capabilities to many commercially available machine learning toolkits, and is widely used in machine learning and data mining applications.
Please note: Weka is cross-platform software. You can run it under Linux, Windows or Mac OS X, at CSE or on your own machine. If you find any discrepancies in results on different platforms please let us know.

In this section is an introduction to the Weka toolkit using the two Weka GUIs that we will use in the assignment, the Weka *Explorer* and *Experimenter*.

Note: this section of the assignment is just an introduction and is not assessed – you don't have to submit anything ! The objective is to familiarise yourself with some of the capabilities of Weka that you will use for the assessed part of the aassignment in Section 4. If you are already familiar with Weka you can skip this introduction and go straight to the questions in Section 4.

3.2 Setting up Weka

The entire Weka package is around 109 MB uncompressed, depending on platform, and can be downloaded from the Weka website:

`www.cs.waikato.ac.nz/~ml/weka/index.html`.

If you are working on a non-CSE machine or you have enough disk space you can do this. Otherwise, if you need to save space on your CSE machine, you can link to the Weka software. Note: on this course we will use the most

recent stable release of Weka which is version 3-8; the latest release of this version is Weka 3-8-1 and can be downloaded from the Weka website.

At CSE:
In your home directory do the following. First, make a symbolic link from the CSE Weka directory to your home directory. Call it something like "myweka":

```
$ ln -s ~weka/weka-3-8-1 myweka
```

Do a quick check to see if this link looks OK:

```
$ ls -l myweka
```

You should see something like:

```
myweka -> /import/adams/1/weka/weka-3-8-1/
```

Then change to your local Weka directory:

```
$ cd myweka
```

And have a look at what there is:

```
$ ls
```

You should see a couple of Java `.jar` files, plus some other stuff including a README file and a PDF file: "WekaManual.pdf". This is a comprehensive and well-written document, although the organisation of it is a little strange — most users will probably not start off with the command-line interface — but there is a lot of good information in here, especially the sections on the main user-interfaces we will use (Explorer and Experimenter), plus details on the Weka file format (`.arff`) and much else besides. This plus the resources at the Weka project website (see above) are all useful for this assignment. Alternatively, you can click on "documention.html" to get to the manuals, etc.
You can either run Weka in your `myweka` directory:

```
$ java -jar weka.jar &
```

or set up your environment so you can run it from other directories. Here is one way to do the latter (assumes you are using bash):

```
$ WEKAHOME=~weka/weka-3-8-1
$ export WEKAHOME
$ CLASSPATH="$CLASSPATH:$WEKAHOME/weka.jar"
$ export CLASSPATH
```

Then check that Weka runs for you:

```
$ java -jar $WEKAHOME/weka.jar &
```

You should see a small window, the "Weka GUI Chooser", indicating that Weka is running[1]. You can choose different tools from this window: we will be using the Explorer to start with.

At home:
These instructions should work:

1. go to the Weka home page `http://www.cs.waikato.ac.nz/~ml/` and download the latest stable version `weka-3-8-1` from the Sourceforge site
2. follow the instructions and install it on your machine
3. if you're on Windows or Mac check the instructions given on the Web page

3.3 Getting started

Next you will need to copy the compressed tar file of datasets for this introduction `~myweka/datasets_intro.tgz` or `~myweka/datasets_intro.zip` to suitable destinations under your home directory. Here's one way:
First make a directory at the same level as `myweka` and call it, say, "experiments"

```
$ cd experiments
```

Now copy the data sets into this directory

```
$ cp ../myweka/datasets_intro.tgz .
```

Next uncompress and open the datasets archive file

```
$ tar xvfz datasets_intro.tgz
```

3.4 Introduction to Weka Explorer

To start Weka Explorer just click on the Explorer button in the Weka GUI Chooser. An introduction to this tool is in Section 4 of the "WekaManual.pdf" included in the Weka download package which you can view like

[1]You can ignore the error messages about missing database drivers, since we will not be using them in this assignment.

this:

```
$ xpdf ~/myweka/WekaManual.pdf
```

To test your knowledge of the Explorer, here are some things you can do:

1) [Preprocess tab] Load the iris data (`iris.arff`). How many values does the "class" attribute have in this data set ?

2) [Classify tab] Choose the rules classifier `ConjunctiveRule`, leaving the parameters at their default setting. Select a 70% split of the data into training and test sets. Run `ConjunctiveRule`. How many *incorrectly* classified instances are there in the test set ?

3) [Classify tab] Choose the decision tree classifier `J48`. Again, leave the parameters at their default setting and run `J48` with the same percentage training/test split as before. Does this give the same "Confusion Matrix" as you got with `ConjunctiveRule` ? What has changed ?

4) [Cluster tab] The default clusterer is `EM`. The default "numClusters" parameter is -1, which means that the algorithm will automatically determine the number of clusters in the data. If you use the entire training set to cluster the iris data, how many clusters are found ?

5) [Cluster tab] Try right-clicking in the "Result list" box and selecting "Visualize cluster assignments". You can inspect individual instances by clicking on the coloured crosses in the plot. Would you say this clustering is "correct" ?

6) [Preprocess and Classify tabs] Load the dataset `cpu.arff`. Click to select the attribute "vendor" in the "Attributes" panel, then click the "Remove" button under this panel – the attribute should have gone away. Switch to the classify tab and choose `LinearRegression` from the set of "functions". Select "Cross-validation" under "Test options" and click "Start". Now have a look at the "Linear Regression Model". From this model, which attribute is the most important (i.e., has the largest "weight") ?

The answers are in this footnote [2].

3.5 Introduction to Weka Experimenter

Weka Experimenter is an alternative GUI for Weka. It complements Weka Explorer which is designed for exploratory data analysis. Weka Experimenter

[2] **1)** 3; **2)** 15; **3)** No — now only two incorrectly classified "Iris-virginica " examples; **4)** 4; **5)** Difficult to say — the data have been labelled with 3 classes, but EM finds 4 clusters — this algorithm normally requires the user to say how many clusters there should be; **6)** CHMAX (1.1868).

allows you to set up, run and analyze comparative experiments in which different machine learning algorithms can be evaluated by their performance on a range of different datasets. This is main interface to the software you will be using for the assignment, although Weka Explorer will also be used. Help is available in the file "WekaManual.pdf". Follow the intructions below to get started.

3.5.1 Start Weka Experimenter

1. click on the Experimenter button in the Weka GUI Chooser
2. click on the Setup tab at the top of the Experiment Environment window
3. click on the "Simple" button to select Experiment Configuration Mode
4. click "New" to initialize the experiment

3.5.2 Adding datasets

Do the following in the Datasets panel of the Setup window:

1. check the "Use relative ..." checkbox
2. click on "Add new ..." to open a dialog window
3. navigate to your "experiments" directory
4. select `anneal.arff` and click "Open" to add it to the list
5. you should see `anneal.arff` displayed in the Datasets panel
6. now add in the same way two more datasets: `audiology.arff` and `vote.arff`
7. if you need to delete a dataset, select it in the Datasets panel and just click "Delete Selected"

3.5.3 Save and load Experiment Definitions

If you are setting up a complicated experiment you should regularly save the experiment definition so you can reload it to go back to a previous point in your work. Here's how:

1. select "Save ..." at the top of the Setup window

2. navigate to your "experiments" directory
3. choose a name, such as "myexpt.exp" (the ".exp" extension is required)
4. click on Save
5. to restore the experiment definition at any time, first select Open in the Setup window, then select "myexpt.exp" (or whatever you called it) in the dialog window

3.5.4 Results

You need a file to save the actual results from each run of a machine learning algorithm on a dataset:

1. in the Results Destination panel select ARFF file
2. click on Browse ... and navigate to your "experiments" directory
3. choose a name, such as "myexpt.arff" (the ".arff extension is required)
4. click on Save
5. you should now see your file in the Results Destination panel

3.5.5 Evaluation: setting up 10 fold cross validation

The results of interest to us will be generated under the control of parameters set via the Experiment Type and Iteration Control panels.

1. in the Experiment Type panel the default selection is Cross-Validation - leave this unchanged
2. the default number of folds is 10 - leave this unchanged
3. and the default method is "Classification" - leave this unchanged
4. in the Iteration Control panel the default number of repetitions is 10 - leave this unchanged
5. and the "Data sets first" button in the Iteration Control panel should be selected

3.5.6 Execution: additional schemes

Now you are ready to run some machine learning algorithms. In Weka terminology these are referred to as "schemes". Add the following schemes via the Algorithms panel:

1. click on Add new ...
2. the default scheme is ZeroR
3. click on OK to add ZeroR
4. you should see ZeroR is added
5. if we didn't want to use ZeroR we could click on Delete to remove it
6. since we *do* want to use it, we won't do that !
7. instead, click on Add new again
8. this time click on Choose
9. select J48 under "classifiers/trees" then click OK
10. now would be a good time to save your Experiment configuration (see above)

Notice that some schemes, like ZeroR, have no parameters. Others, like J48 have a number of parameters which you can change in the dialog box. However for now we will just use the default parameter settings.

Now we have finished setting up the experiment, so let's run it. This will be 10 repetitions of 10-fold cross-validation for each scheme on each of the datasets.

1. click on the Run tab at the top of the Experiment Environment window
2. click Start
3. In the Status bar at the bottom of the Experiment Environment window you should see an indication of the progress of the experiment
4. on completion, you should see in the Log window something like

```
10:27:26: Started
10:27:37: Finished
10:27:37: There were 0 errors
```

On your machine the runtime may be different, but it shouldn't take very long. Note: some algorithms can take a long time to run on certain datasets ! Running a complete cross-validation experiment can then be time-consuming. However, you can set it running and come back from time to time and check progress.

3.5.7 Analysis: compare schemes

Now we can use the Weka Experimenter to compare the schemes:

1. click on the Analyse tab at the top of the Experiment Environment window
2. click on Experiment to analyse the results of the current experiment
3. in the "Source" panel you should see "Got 600 results" ($10 \times$ 10-fold cross-validation for 2 algorithms on 3 datasets)
4. the next operations are applied in the "Configure test" panel
5. in the Comparison field box select "Percent_incorrect"
6. click "Select" in the Test base field, and in the popup box select "ZeroR"
7. click on the "Show std. deviations" check box (you should always do this!)
8. click on Perform test to automatically generate the comparisons with statistical significance estimates

In the Test output window you should see a table of the comparisons for each of the three schemes in the experiment. The *baseline* scheme is ZeroR; this tells you the *default* error rate, i.e., what the error is when you classify the test data just by predicting that each instance belongs to the most frequent class in the training set.

For each of the other schemes in the table you should see an entry containing the percent incorrect, the standard deviation in brackets and possibly an annotation symbol indicating a statistically significant result: A 'v' symbol means the result is significantly higher than the baseline classifier. A '*' symbol means the result is significantly lower than the baseline classifier. The default statistical significance level is 0.05. At the bottom of each column (apart from the first) is a count of the number of datasets on which the scheme in the column was significantly above/equal/below the baseline. The test for statistical significance is explained more in "WekaManual.pdf".

3.5.8 Saving results

Saving the current contents of the Test output window. First, if you want to start a new file:

1. click on Save output
2. navigate to your "experiments" directory
3. enter a file name such as "myexpt.res"
4. click on Save

Or, if you already have a result file, then

1. select it in the window, or
2. enter its file name, such as "myexpt.res"
3. click on Save
4. click on Append to add results to the designated output file

Note that if you click on "Select" next to Output format you get a dialog box, which you will need to expand to see all of the options. One thing that is very useful is that if you click on "Select" next to Advanced setup you get options to save output in formats such as LaTeX, HTML and CSV for import into a spreadsheet like Excel.
In experiments we often want to change the data in some way then evaluate the performance of a learning algorithm on this data. We can view this process abstractly as a "filtering" stage followed by a "classification" stage. Depending on the details of what we are trying to do, some care must be taken in order to set things up properly. For example, often we want to apply a filter to the training data while leaving the test data unaffected. However, Weka provides a class for this called the "FilteredClassifier". There are several parameters, but the two main ones concern the selection of the classifier and filter required. Let's try this out, using Weka's "Resample" filter as a way of investigating the effect on learning of using a subset of the training data.
As before, add the following schemes via the Algorithms panel:

1. click on Add new ...
2. select "Filtered Classifier" under "classifiers/meta"
3. a dialog box will appear

4. where it says "classifier" click on the "Choose" button
5. select "IBk" under "classifiers/lazy"
6. note that IBK is Weka's class for instance-based classifiers, or k-Nearest Neighbour (k-NN)
7. you can set the value of k by editing the KNN field
8. for now, just leave it at the default value of 1
9. now we need to set up the filter
10. select "Resample" under "filters/unsupervised/instance"
11. there are two parameters to the Resample filter that we will need
12. one is "sampleSizePercent", for now leave it at the default of 100%
13. the other is "noReplacement" which we will always set to "True" in this assignment
14. once the classifier and the filter are set up, click on OK
15. repeat these steps to add, in turn, the following:
16. "IBk" with 20% of the data
17. "ZeroR" with 100% of the data
18. now would be a good time to save your Experiment configuration (see above)

Now you can run the experiment and analyse it as before. What do you observe ? Is this what you expected ?

The Experimenter is quite powerful and will often be the main tool you use in Weka, particularly with some of the preprocessing tools used with the meta-classifier; you might want to play around a bit more to get a feel for what you can do with it.

In any case, you should now be ready to use Weka to complete the assignment, so read on.

4 Questions

Question 1

For this question the idea is to run the Experimenter with two different algorithms and a range of different sample sizes taken from the same training set to assess the effect of training sample size on error. You will use 2 different algorithms (plus the standard baseline, ZeroR) to generate two different sets of "learning curves" (you won't actually have to plot curves, but they will be recorded as tables) on 8 real-world datasets:

```
anneal.arff
audiology.arff
autos.arff
credit-a.arff
hypothyroid.arff
letter.arff
microarray.arff
vote.arff
```

Running the Experimenter [2 marks]

1. Start a new experiment, as explained in Section 3:
   (a) set a filename for results destination, like 'q1.arff'
   (b) the experiment should contain 10 repetitions of 10-fold cross-validation (CV);
   (c) ensure "Classification" and "Data sets first" are selected;
   (d) add the 8 datasets in the order in which they appear above;
   (e) add 5 versions of the FilteredClassifier, set up for each version as follows:
      - for the classifier, select IBk with default parameters, i.e., 1-Nearest Neighbour (1NN));
      - for the filter, select Resample with "noReplacement" set to True and "sampleSizePercent" set to, respectively, 10%, 25%, 50%, 75%, and 100%.
   (f) now add 1 more version of the FilteredClassifier as follows:
      - for the classifier, select ZeroR;
      - for the filter, select Resample with "noReplacement" set to True and "sampleSizePercent" set to 100%.

(g) check that you now have 6 entries in the Algorithms pane (5 of IBK and 1 of ZeroR).

2. Run the experiment:

   (a) if there are no errors from the run, continue

   (b) some error messages are not very clear, so you will need to check: for example, a common error is including a dataset that does not have a legal class value for a classifier learning algorithm.

3. Analyze the results as explained earlier:

   (a) ensure "Show std. deviations" is on in the "Configure test" panel

   (b) in "Output Format" ensure "Plain Text" is selected

   (c) compare algorithms by "Percent_incorrect" using ZeroR as the baseline classifier (select this as "Test base")

   (d) save these results to a new file called "q1.out"

   (e) Hint: if you use spreadsheets, you might like to also save these results in CSV format to a new file to slightly speed up some later analysis

4. Now go back to the Setup page and repeat the entire experiment, just replacing "IBk" with the decision tree learner "J48" — if you are careful, you should only need to change the selected classifier name in the experiment configuration, so only 5 changes will be required

5. Run and Analyze the experiment as before, just remembering this time to *append* your results to "q1.out" (and your CSV file, if you have one)

Results interpretation

Answer these questions in a new file called "answers.txt". Your answers must be backed up by referring to the results you saved in "q1.out". ***Please note:*** in this assignment simply writing down a description of the results is ***not sufficient***, and will not get any marks. When asked, you must attempt to ***explain why*** you think the results are as they are.

1a) [2 marks] Looking at the results for *both* IBk and J48 with respect to the baseline of ZeroR over all training set sizes, can you observe a "learning curve" effect, i.e., that the error changes dependent on the proportion of training data seen by the algorithm ? Is there any variation in this effect, between datasets and/or algorithms ? If you think there has been variation, give a brief description of what you observed.

1b) [**4 marks**] For each algorithm, IBk and J48, over all of the datasets, find the average change in error when moving from the default prediction (ZeroR) to learning from 10% of the training set as follows.

Let the error on ZeroR be err_0 and the error on 10% of the training set be err_{10}.

For each algorithm, calculate the percentage reduction in error relative to the default on each dataset as: $\frac{\text{err}_0 - \text{err}_{10}}{\text{err}_0} \times 100$. Sum these values and divide by 8 to get the mean of this value for the algorithm.

Now repeat exactly the same process by comparing IBk and J48, over all of the datasets, learning from 100% of the training set, compared to default. Organise your results by grouping them into a 2×2 table in your file "answers.txt", something like this:

	Mean error reduction relative to default	
Algorithm	after 10% training	after 100% training
IBk	Your result	Your result
J48	Your result	Your result

If you observe a positive reduction in error, is this larger than, smaller than, or about the same as what you would have expected after seeing 10% of the training data ? After seeing 100% ?

Is the effect more pronounced for IBk, or for J48 ? If you think it is more pronounced for one of the algorithms, suggest an explanation. If not, say why not. [Hint: if you also saved the results tables in CSV format you can save a little time by calculating the relative and percentage reductions in error for this question in a spreadsheet.]

Question 2

Dealing with noisy data is a key issue in machine learning. Unfortunately, even algorithms that have noise-handling mechanisms built-in, like decision trees, can overfit noisy data, unless their "overfitting avoidance" or regularization parameters are set properly.

Once again we will use the FilteredClassifier, where this time the filter will be Weka's AddNoise filter. As the name suggests this adds "class noise" by randomly changing the actual class value to a different one for a specified percentage of the training data. Here we will specify three arbitrarily chosen levels of noise: low (20%), medium (50%) and high (80%). The learning algorithm must try to "see through" this noise and learn the best model it can, which is then evaluated on test data *without* added noise.

We will also let the algorithm do a limited search using cross-validation for the best pruning parameters on each training set with Weka's built-in CVParameterSelection metalearner. This is based on the "wrapper" method, where for each one of a set of parameter values, the learning algorithm is run and performance is evaluated using cross-validation. The parameter values giving the best performance is selected.

To set this up using the CVParameterSelection metalearner requires entering a string defining a set of values for the parameter using Weka's alphabetic code for the parameter (similar to the flag for a Unix command), plus three numbers, namely the minimum, maximum and increment. For example, if we entered the string "M 1 5 1" this would define the set of values $\{1, 2, 3, 4, 5\}$ for J48's -M parameter which sets the minimum number of examples that can appear in a leaf node in a decision tree.

Running the Experimenter [2 marks]

1. Start by setting up a new experiment and select the following datasets:

   ```
   glass.arff
   primary-tumor.arff
   balance-scale.arff
   heart-h.arff
   ```

   (a) set a filename for results destination, like 'q2.arff'

   (b) the experiment should contain 10 repetitions of 10-fold cross-validation (CV);

   (c) ensure "Classification" and "Data sets first" are selected;

   (d) add the 4 datasets in the order in which they appear above;

(e) add 4 versions of the FilteredClassifier, set up for each version as follows:

- for the classifier, select CVParameterSelection from "classifiers/meta"; in the dialog ensure you set the classifier to be: '`J48 -C 0.01 -M 2`' and CVParameters to be: '`M 2 30 5`'
- for the filter, select AddNoise with "percent" set to, respectively, 0%, 20%, 50%, and 80%.

(f) now add 1 more version of the FilteredClassifier as follows:

- for the classifier, select J48 with default parameters;
- for the filter, select AddNoise with "percent" set to 50%.

(g) check that you now have 5 entries in the Algorithms pane (4 of J48 with CVParameterSelection and 1 of J48 with default parameters).

2. Run the experiment:

3. Analyze the results as explained earlier:

(a) ensure "Show std. deviations" is on in the "Configure test" panel

(b) in "Output Format" ensure "Plain Text" is selected

(c) compare algorithms by "Percent_incorrect" using the FilteredClassifier with J48 and AddNoise at 0% as the baseline classifier (select this as "Test base")

(d) save these results to a new file called "q2.out"

Results interpretation

Answer these questions in the file "answers.txt". Your answers must be backed up by referring to the results you saved in "q2.out".

2a) [2 marks] Looking at the error (Percent_incorrect) results for tree learning on these data sets as noise is increased, has learning managed to avoid overfitting at low, medium and high levels of added noise ?

2b) [1 mark] Is parameter selection helping with overfitting avoidance ? How can you assess this ?

Question 3

This question involves mining a data-set on California house prices from census data in the 1990s. We will be using regression to do this since the output is numeric. Since this problem involves using attribute or feature transformations we will need to do this using the Weka Explorer interface.

In the Explorer, click on the Pre-Process tab at the top of the Explorer window, then click on "Open file ..." and select "`houses.arff`", then click on Open. Now click on the Classify tab at the top of the Explorer window, choose LinearRegression, select 10-fold cross-validation, select "median_house_value" as the target (class) variable and click Start in the left side panel.

Linear Regression **[1 mark]** In the output panel you should see a regression equation with some results. Save the results list into a new file called "q3.out".

Return to the Pre-Process tab and choose the filter AddExpression under "filters/unsupervised/attribute". In the dialog box, click on More to see examples of the kind of attribute or feature transformation expressions you can add and apply here. Now set up a log transform to the class variable "median_house_value". Be sure to click "Apply" to actually transform the values in the training set (the transform just takes place on the data in main memory, so your original dataset is not changed, and you can save the transformed data to a new file if needed). Your new variable should appear in the attributes list. Now delete the old class variable, rerun the linear regression, and save (append) the results list into your "q3.out" file.

In the original research all variables (apart from latitude and longitude) were removed and replaced with a set of transformed versions. Transformations included squares, cubes, and logs of ratios. Experiment with at most two more sets of transformations to the variables, run linear regression on the transformed data, and save the output to your "q3.out" file.

Results interpretation Answer this in the file "answers.txt". Your answers must be backed up by referring to the results you saved in "q3.out".

3a) **[2 marks]** Have any of your variable transformations had any effect on the cross-validation error of the linear regression ? If so, state which transformations, and what the effect on error was.

3b) **[1 mark]** How do you account for such a change ?

Question 4

This question involves mining text data, for which Weka has a nice filter called "StringToWordVector". Each text example is a string of words with a class label, and the filter converts this to a vector of word counts in various formats. To save time this has already been done for you, reducing the set of words to a vocabulary size of 1000. The dataset contains $10,060$ "snippets", short sequences of words taken from Google searches, each of which has been labelled with one of 8 classes, such as business, sports, etc.

Using a vector representation for text fdata means that we can use many of the standard classifier learning methods. However, the dataset is highly "sparse", in the sense that for any example nearly all of its feature values are zero. To tackle this problem, we typically apply methods of feature selection, or dimensionality reduction. In this question you will investigate the effect of using the method of RandomProjections, a dimensionality reduction method. As in the previous question it is slightly easier to do these tasks with the Weka Explorer interface rather than the Experimenter.

Training text classifiers and dimensionality reduction [1 mark]
In the Explorer, load the file "`web_snippets.arff`". Click on the Classify tab at the top of the Explorer window and choose NaiveBayesMultinomial (MNB) from "bayes". In "Test options" select "Cross-validation". Be sure that "snip_class" is selected as the target variable. Since MNB has no parameters to set, click Start in the left side panel. When MNB has finished, right-click in the Result list and save this output into a new file called "q4.out".

Now return to the Preprocess tab and choose the filter unsupervised attribute filter "RandomProjection". The "numberOfAttributes" parameter should be set to 10 (the default) and all other parameters should be left at the default values. Now click on Apply and a dataset will be generated with the new attributes (denoted K1 to K10). Go to the Classify tab, select J48 and run it with the same settings as MNB above, saving (appending) the results to "q4.out". Now we cannot run Weka's MNB on this transformed dataset, so instead just select 'NaiveBayes' and run it with the same settings as MNB and J48, saving the results to "q4.out" as before. Repeat this process twice more, varying only the number of attributes found by "RandomProjection" to obtain new datasets of 100 and 500 attributes, respectively, and saving the results of cross-validated J48 and NaiveBayes to "q4.out". This should give a total of 7 results from the learning runs: 1 from MNB on the untransformed data, plus 2 each from each of the learning algorithms applied to data reduced to 10, 100 and 500 attributes.

Results interpretation **[2 marks]** Answer this in the file "answers.txt". Explain what happened. Your answer must be backed up by referring to the results you saved in "q4.out". Can you think of any characteristics of text classification in general and snippets in particular that would give you these results ?