Ministry of Education and Science of Ukraine
National Technical University of Ukraine
«Igor Sikorsky Kyiv Polytechnic Institute»

№1.2

Work with WEKA. ATTRIBUTE SELECTION & REGRESSION

(https://docs.google.com/document/d/1C6K2UkkI4PvkyYjy3Cm_yAvkWQkXPiAO/edit)

The work was done by
Zvychaynaya Anastasia

The work was checked by
Alexander Oriekhov

Kyiv 2022

Execution of work:

1. LANDSAT DATA

The second data set to be used is the [Landsat](#) satellite imaging set. Acquaint yourself with this by reading [this description](#).

Most importantly, note that each instance is a 3 × 3 region of pixels with recordings in 4 different spectral bands for each pixel. The task is to classify the instances according to the soil type of the **centre** pixel.

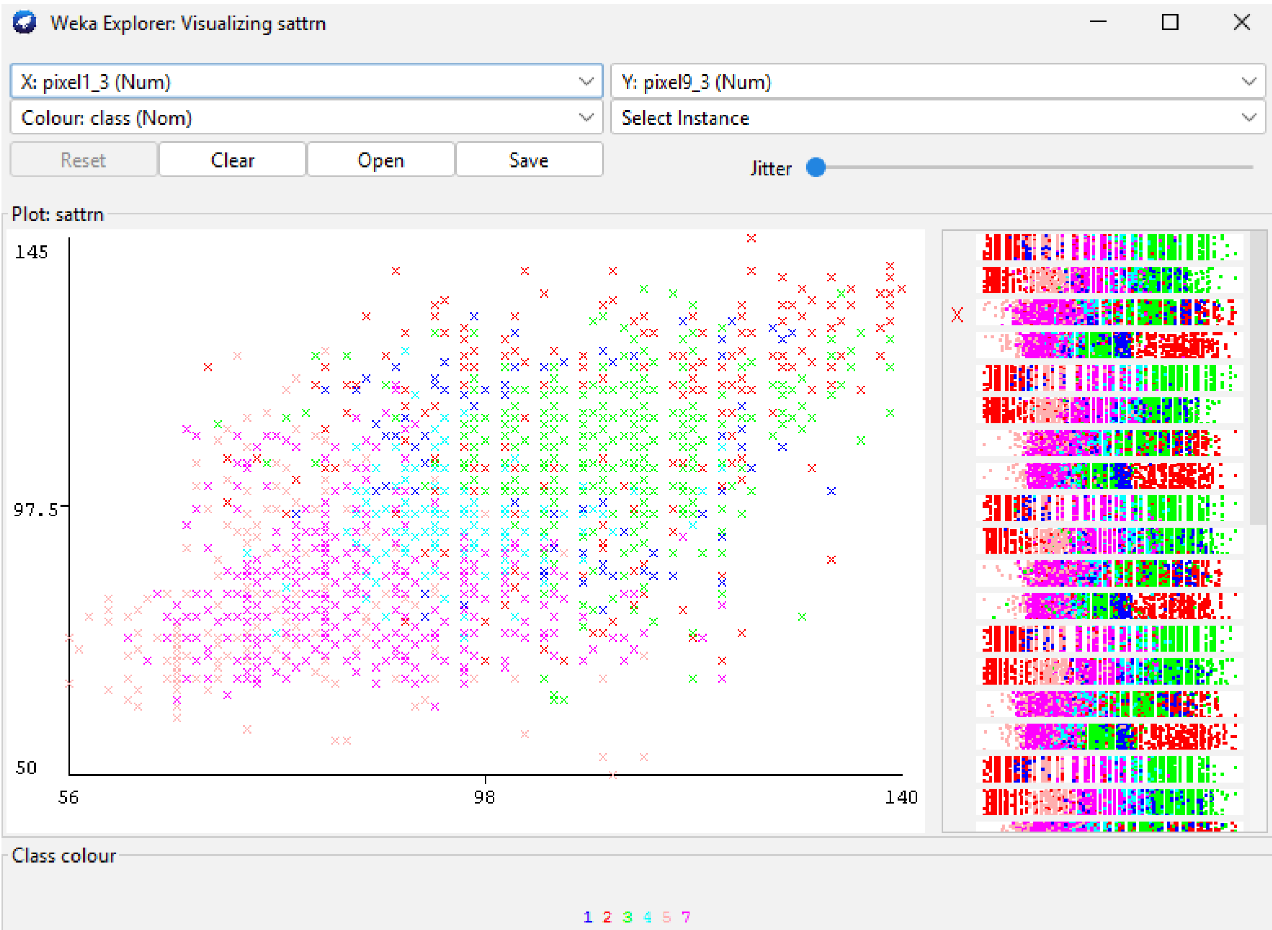
A slightly modified data set is used for this lab. Download the following file:

- [landsat.arff](#): landsat data

Load this dataset into Weka through the *Preprocess* tab. In this case no further preprocessing is required.

In any real world application it is worth visualising a data set before attempting to train a model with it. Bring up the *Visualize* tab in Weka. This plots all attributes against each other, which is useful for visually assessing correlations between attributes, or which attributes may be most important in discriminating classes. The colour of each point represents its class label in the training set.

In this case the large number of attributes can be stifling. Select any square on the plot to bring up the alternative interface.



1. Using the *X* and *Y* select boxes at the top, try plotting various pairs of attributes against each other to get a feel for the data.
2. Are there any significant correlations?

   Yes, I observe significant correlation between the pixelK_I and the pixelJ_I, where K, J = $\overline{1,9}$, I = $\overline{1,4}$.
3. Can you find a pair of attributes that seems particularly effective at discriminating between the classes? *In such a case there should only be a little mixing of the colours. Make a note of these for later.* **Hint:** recalling the format of the data, concentrate on pixel 5, the centre pixel, which we might expect to be most telling given it contains the soil type we are classifying.

   As the task is to classify the instances according to the soil type of the centre pixel, then let us focus on the pixel 5. So, we can see that plots with the pixel5_I and the pixelK_$\bar{I}$, where K = $\overline{1,9}$, I = $\overline{1,3}$, $\bar{I} = \begin{cases} 2,3 \ if \ I = 1 \\ 1,3 \ if \ I = 2 \\ 1,2 \ if \ I = 3 \end{cases}$ show us how we can discriminate the classes. This is also shown by graphs with the pixel5_1 and the pixelK_4.

We'll now attempt to build a Naive Bayes classifier over the data. We will apply our visual observations in a moment.

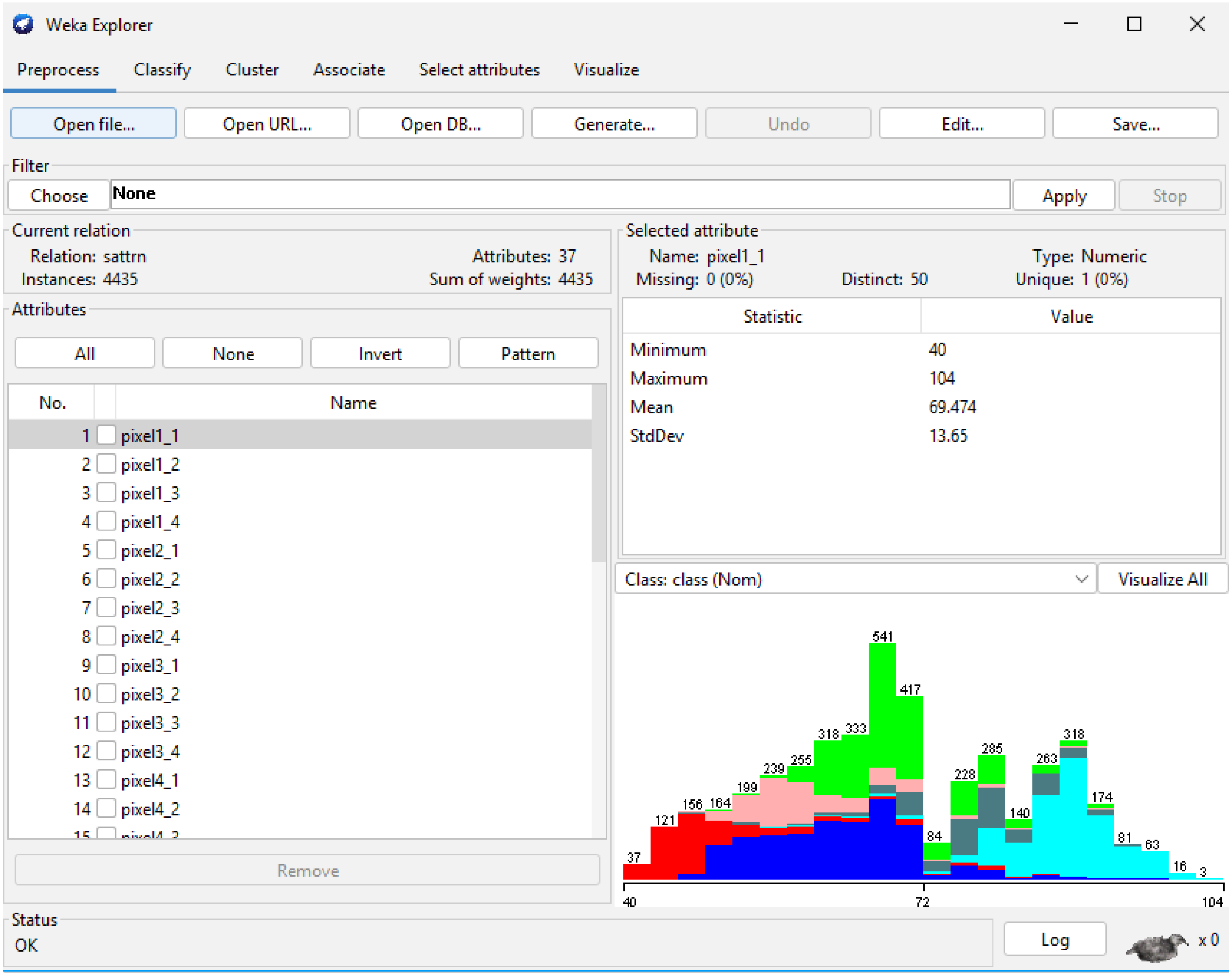
Return to the *Classify* tab and select the Naive Bayes classifier again if it is not already selected. Let's use 5-fold cross-validation (default option is 10). Hit *Start* to run the cross-validation.

Classifier output

```
Correctly Classified Instances        3526               79.5039 %
Incorrectly Classified Instances       909               20.4961 %
Kappa statistic                          0.7488
Mean absolute error                      0.0685
Root mean squared error                  0.2559
Relative absolute error                 25.4274 %
Root relative squared error             69.73   %
Total Number of Instances             4435

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,790    0,029    0,898      0,790    0,841      0,797    0,971     0,923     1
                 0,896    0,001    0,991      0,896    0,941      0,936    0,996     0,981     2
                 0,891    0,031    0,889      0,891    0,890      0,859    0,982     0,924     3
                 0,631    0,073    0,470      0,631    0,539      0,490    0,901     0,463     4
                 0,738    0,069    0,558      0,738    0,636      0,593    0,929     0,726     5
                 0,756    0,039    0,856      0,756    0,803      0,750    0,957     0,877     7
Weighted Avg.    0,795    0,037    0,820      0,795    0,803      0,764    0,962     0,855

=== Confusion Matrix ===

   a   b   c   d   e   f   <-- classified as
 847   1  35   0 189   0 |   a = 1
  17 429   0   3  28   2 |   b = 2
  17   0 856  82   1   5 |   c = 3
   8   0  64 262   9  72 |   d = 4
  54   3   0  13 347  53 |   e = 5
   0   0   8 197  48 785 |   f = 7
```

Now we'll apply the knowledge of our visual observations:

1. Return to the *Preprocess* tab.



2. Check the box next to the two attributes you identified as being most useful above. **Cheat:** If you're not sure, try **`pixel5_2` and `pixel5_4`.**

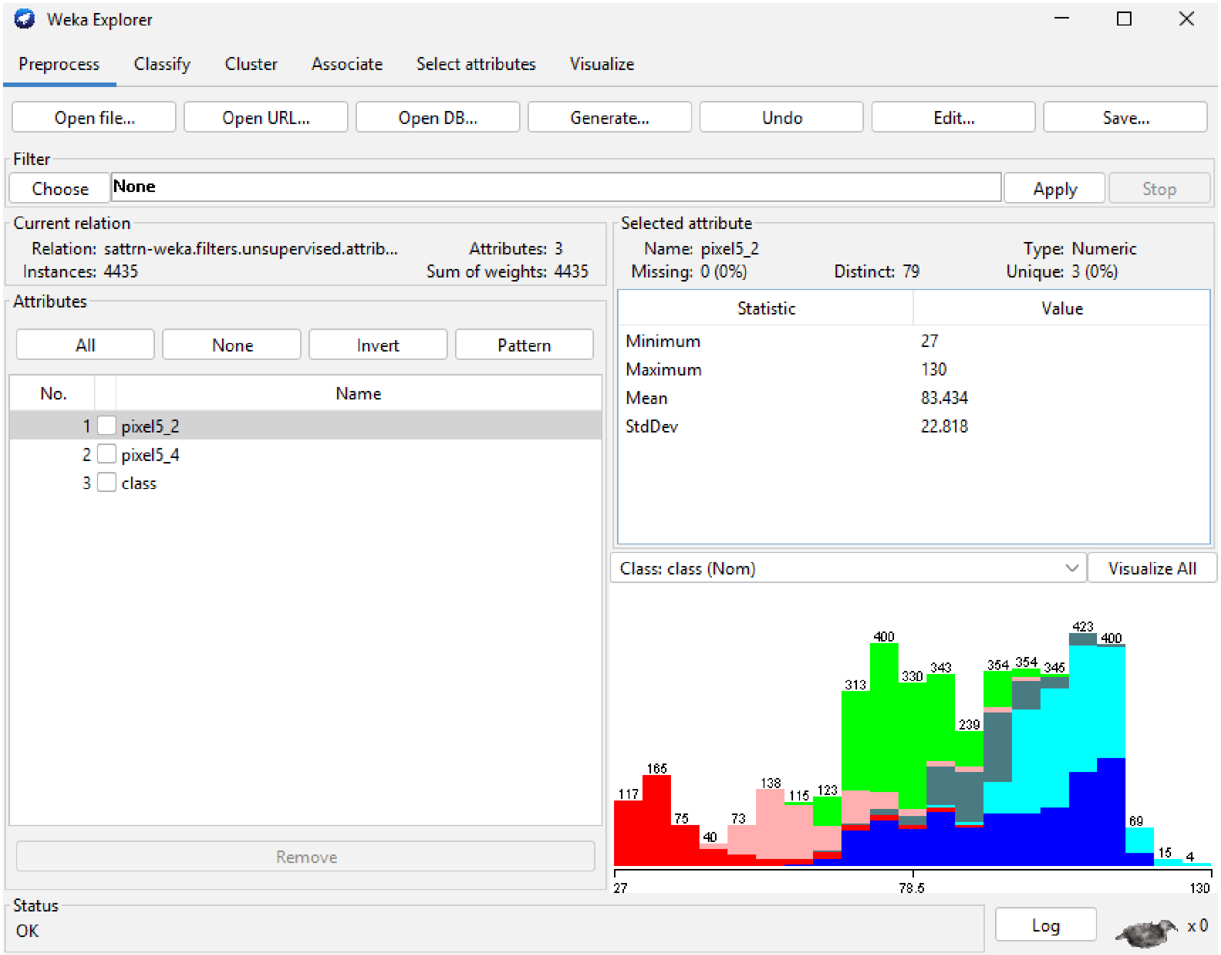
3. Check the box next to the `class` attribute.

4. Click the *Invert* button and then the *Remove* button. This should remove all attributes except for your two most important and the class label.

5. Return to the *Classify* tab and select the 5-fold cross-validation again.

I want to emphasize that the plot of pixel5_2 and pixel5_4 depicts group intersection significantly more than the plot of pixel5_1 and pixel5_2, therefore we have worst prediction (66% against 76%), and I do not khow why the author recommends us to try those pixels.

With pixel5_2 and pixel5_4:



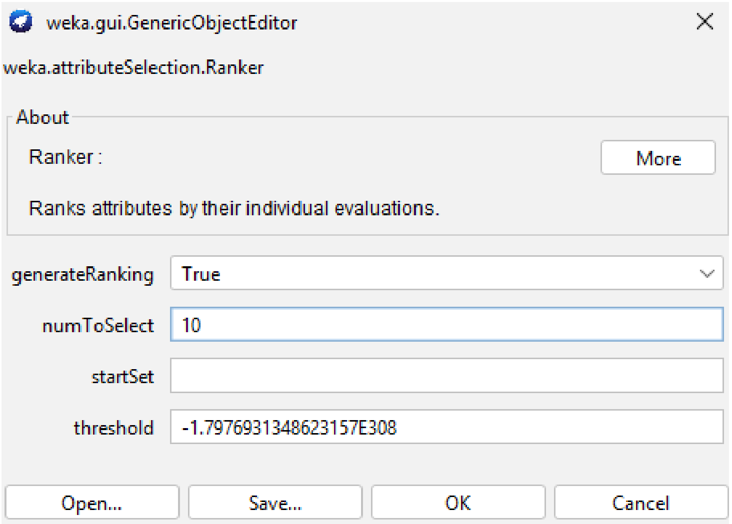

```
Classifier output

Correctly Classified Instances        2938               66.2458 %
Incorrectly Classified Instances      1497               33.7542 %
Kappa statistic                          0.5815
Mean absolute error                      0.137
Root mean squared error                  0.2692
Relative absolute error                 50.8643 %
Root relative squared error             73.3552 %
Total Number of Instances             4435

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,327    0,062    0,628      0,327    0,430      0,343    0,850     0,576     1
                 0,864    0,001    0,995      0,864    0,925      0,920    0,987     0,956     2
                 0,838    0,166    0,582      0,838    0,687      0,597    0,919     0,740     3
                 0,441    0,061    0,428      0,441    0,434      0,375    0,874     0,434     4
                 0,670    0,014    0,849      0,670    0,749      0,729    0,942     0,822     5
                 0,838    0,120    0,681      0,838    0,751      0,671    0,950     0,864     7
Weighted Avg.    0,662    0,086    0,675      0,662    0,649      0,581    0,915     0,733

=== Confusion Matrix ===

   a   b   c   d   e   f   <-- classified as
 351   0 487  45  13 176 |   a = 1
  21 414   0   5  26  13 |   b = 2
  83   0 805  68   0   5 |   c = 3
  26   0  79 183   0 127 |   d = 4
  57   2   5   4 315  87 |   e = 5
  21   0   7 123  17 870 |   f = 7
```

With pixel5_1 and pixel5_2:

```
Classifier output

Correctly Classified Instances        3405               76.7756 %
Incorrectly Classified Instances      1030               23.2244 %
Kappa statistic                          0.7135
Mean absolute error                      0.1054
Root mean squared error                  0.2348
Relative absolute error                 39.1332 %
Root relative squared error             63.9695 %
Total Number of Instances             4435

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,774    0,007    0,973      0,774    0,862      0,834    0,974     0,957     1
                 0,839    0,016    0,866      0,839    0,853      0,835    0,967     0,910     2
                 0,915    0,047    0,844      0,915    0,878      0,843    0,983     0,928     3
                 0,467    0,059    0,449      0,467    0,458      0,401    0,880     0,406     4
                 0,596    0,062    0,533      0,596    0,563      0,509    0,911     0,615     5
                 0,790    0,088    0,733      0,790    0,760      0,684    0,934     0,771     7
Weighted Avg.    0,768    0,046    0,782      0,768    0,771      0,726    0,950     0,814

=== Confusion Matrix ===

   a   b   c   d   e   f   <-- classified as
 830   3  43  15 136  45 |   a = 1
   0 402   0   3  41  33 |   b = 2
   1   0 879  76   0   5 |   c = 3
   7   0 103 194   0 111 |   d = 4
  13  59   0  13 280 105 |   e = 5
   2   0  17 131  68 820 |   f = 7
```

Attribute Selection via Information Gain: Weka has a whole range of functionalities for attribute selection. We will use Information Gain, i.e the mutual information between two variables. In attribute selection we are interested in the mutual information between the class and each of the other attributes.

- Go to the 'Select attributes' tab sheet
- Choose the Information Gain attribute selection method from the Attribute Evaluator box. Weka supports feature selection via information gain using the **InfoGainAttributeEval** Attribute Evaluator.
- Choose the Ranker search method in the Search Method box; you can indicate the number of attributes to retain to be 10 or so.



- Click 'Start'. Does the ranking make sense based on your previous observations? Which attributes obtain the highest score?

```
=== Attribute Selection on all input data ===

Search Method:
	Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 37 class):
	Information Gain Ranking Filter

Ranked attributes:
1.162772   18 pixel5_2
1.153942   17 pixel5_1
1.08395    21 pixel6_1
1.071975   22 pixel6_2
1.034403   13 pixel4_1
1.02718    14 pixel4_2
1.023388   20 pixel5_4
1.011337   29 pixel8_1
1.004091   16 pixel4_4
1.000472    5 pixel2_1

Selected attributes: 18,17,21,22,13,14,20,29,16,5 : 10
```

Yes, I talked about it! This rating is absolutely accurate.

- Would you logically expect pixel6_1 and pixel6_2 to be more important than pixel5_4? Go back to visualisation and plot pixel6_1 against pixel5_1, or pixel6_2 against pixel5_2. Can you spot a problem with information gain attribute selection here?

  Yes, this can be expected based because of plots and the generalized formula that I wrote above.

Run the classifier again and note the percentage of correctly classified instances:

- How does this compare to the result using all attributes?

In this case, we have 79.4814% accuracy versus 79.5039% for all attributes used. I think that the difference between the experiments is small, but if we have a huge number of features, then we must make a selection of attributes to avoid the curse of dimensionality.

- What does this tell you about these two particular attributes?

I am glad that the algorithm confirmed that the point5_1 and the point5_2 are in the top.

Classifier output

```
Correctly Classified Instances        3525               79.4814 %
Incorrectly Classified Instances       910               20.5186 %
Kappa statistic                          0.7485
Mean absolute error                      0.0685
Root mean squared error                  0.2559
Relative absolute error                 25.4218 %
Root relative squared error             69.7181 %
Total Number of Instances             4435

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,789    0,029    0,897      0,789    0,840      0,796    0,971     0,925     1
                 0,894    0,001    0,991      0,894    0,940      0,934    0,996     0,981     2
                 0,891    0,031    0,889      0,891    0,890      0,859    0,982     0,925     3
                 0,631    0,073    0,470      0,631    0,539      0,490    0,901     0,456     4
                 0,738    0,070    0,557      0,738    0,635      0,592    0,929     0,725     5
                 0,757    0,039    0,857      0,757    0,804      0,751    0,956     0,877     7
Weighted Avg.    0,795    0,037    0,820      0,795    0,803      0,764    0,962     0,855

=== Confusion Matrix ===

   a   b   c   d   e   f   <-- classified as
 846   1  36   1 188   0 |   a = 1
  18 428   0   3  28   2 |   b = 2
  17   0 856  82   2   4 |   c = 3
   8   0  63 262  10  72 |   d = 4
  54   3   0  13 347  53 |   e = 5
   0   0   8 196  48 786 |   f = 7
```

- Why might this approach be useful? **Hint:** consider larger, more complex problems.

Because with a huge dataset, we can face with the curse of dimensionality.

- What distribution is used by Weka to model the attribute values?

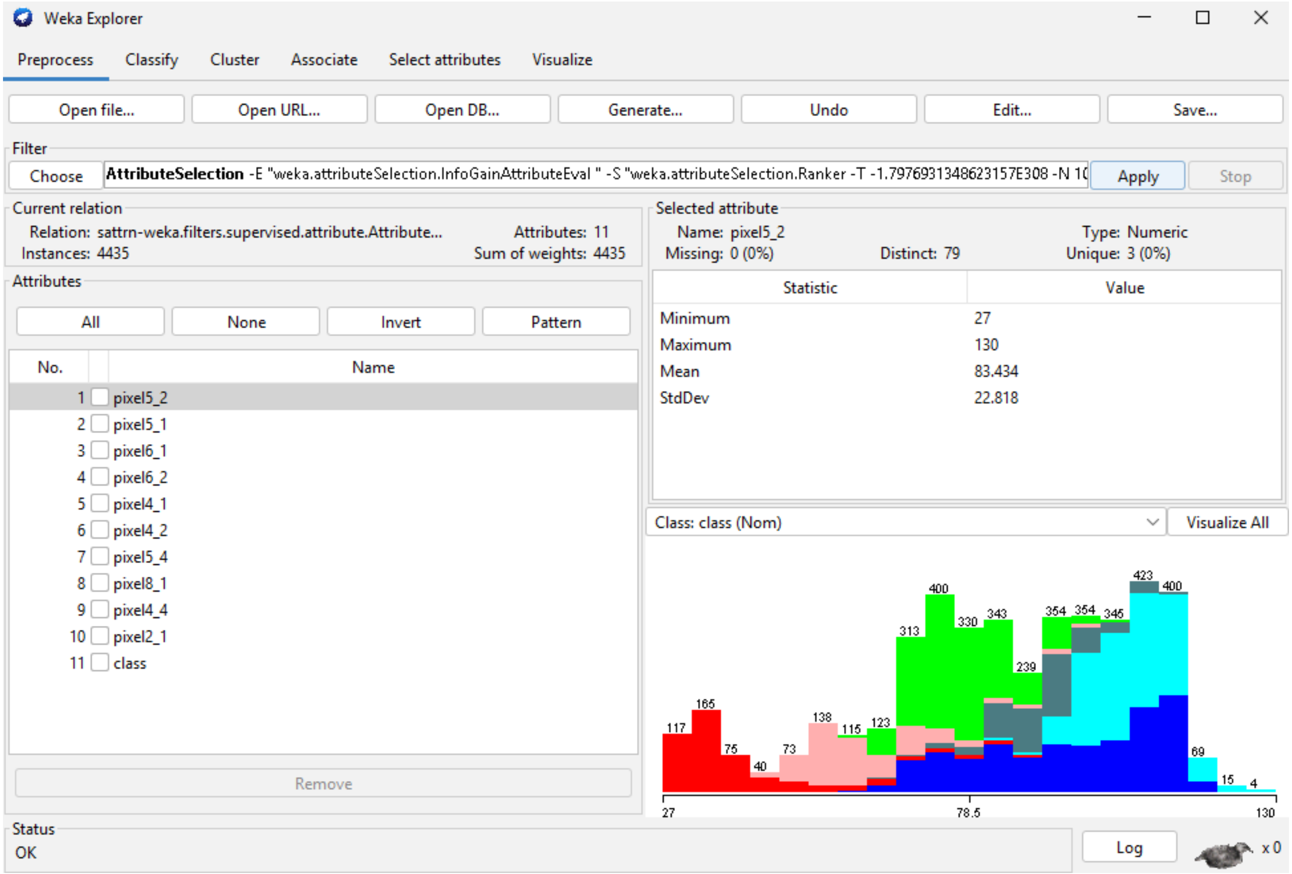
```
Ranked attributes:
1.162772   18 pixel5_2
1.153942   17 pixel5_1
1.08395    21 pixel6_1
1.071975   22 pixel6_2
1.034403   13 pixel4_1
1.02718    14 pixel4_2
1.023388   20 pixel5_4
1.011337   29 pixel8_1
1.004091   16 pixel4_4
1.000472    5 pixel2_1
```

As I understand, WEKA gives us the attribute importance weight (see left column), and the probability of each feature is proportional to its weight.

When you perform attribute selection in the 'Select attributes' tab sheet, you only get a list of ranked attributes. You may want the attribute selection to have an effect on all tabs for you to work on the reduced data set. To achieve this, you have to perform attribute selection in the 'Preprocess' tab sheet.

- Select the supervised 'AttributeSelectionFilter'.
- Set its settings as before (InfoGain + Ranker, numToSelect: 10)
- Apply the filter.

You have now replaced the working dataset for the whole weka environment.



- Train a NaiveBayes classifier on the reduced data.

Classifier output

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        3494               78.7824 %
Incorrectly Classified Instances       941               21.2176 %
Kappa statistic                          0.74
Mean absolute error                      0.0725
Root mean squared error                  0.2521
Relative absolute error                 26.892  %
Root relative squared error             68.6806 %
Total Number of Instances             4435

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,775    0,022    0,917      0,775    0,840      0,799    0,978     0,947     1
                 0,864    0,001    0,988      0,864    0,922      0,916    0,984     0,947     2
                 0,892    0,033    0,882      0,892    0,887      0,855    0,985     0,941     3
                 0,607    0,073    0,462      0,607    0,525      0,474    0,899     0,450     4
                 0,745    0,075    0,541      0,745    0,627      0,584    0,934     0,744     5
                 0,761    0,046    0,835      0,761    0,796      0,739    0,955     0,862     7
Weighted Avg.    0,788    0,038    0,816      0,788    0,797      0,757    0,963     0,858

=== Confusion Matrix ===

   a   b   c   d   e   f   <-- classified as
 831   0  32   7 199   3 |   a = 1
  17 414   0   4  35   9 |   b = 2
   3   0 857  98   0   3 |   c = 3
   7   0  72 252   3  81 |   d = 4
  47   5   1   7 350  60 |   e = 5
   1   0  10 177  60 790 |   f = 7
```

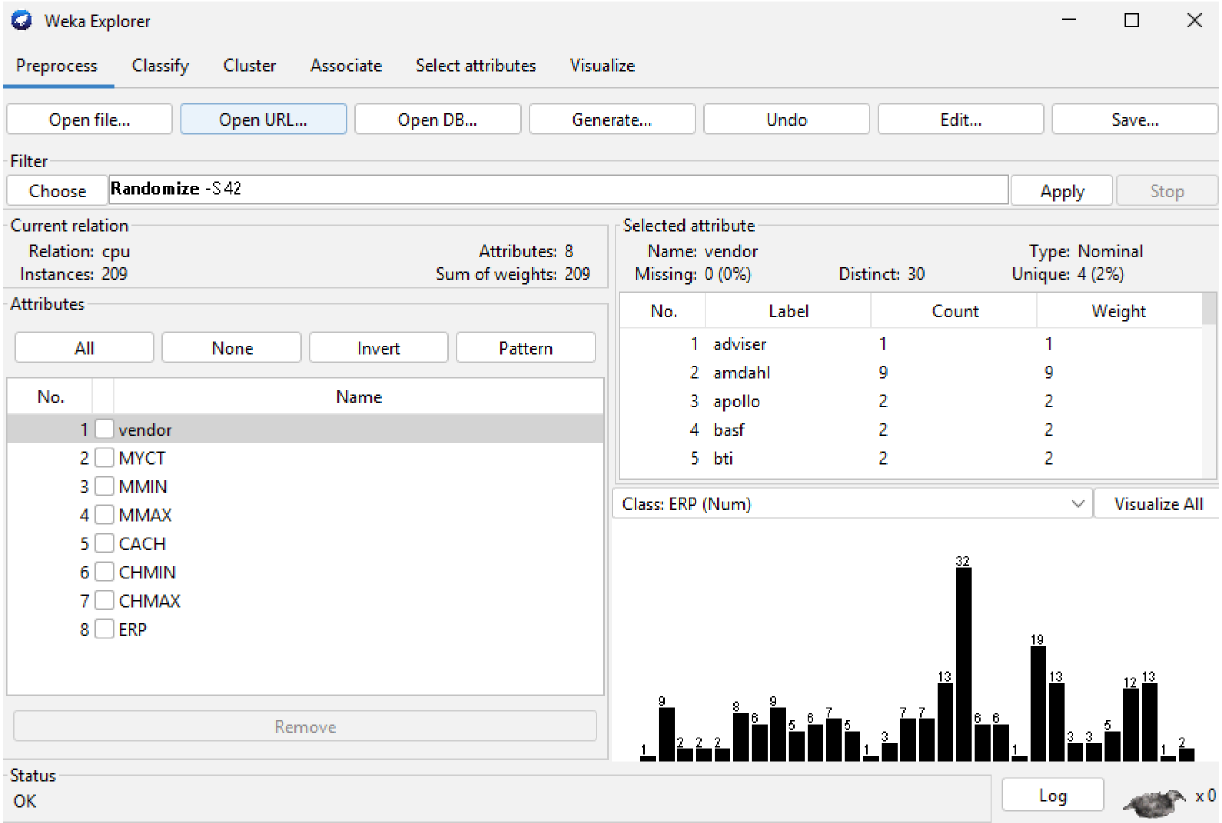
- Was the performance drop significant? Why?

As we have main attributes, we have not lost much in accuracy.

2. CPU PERFORMANCE

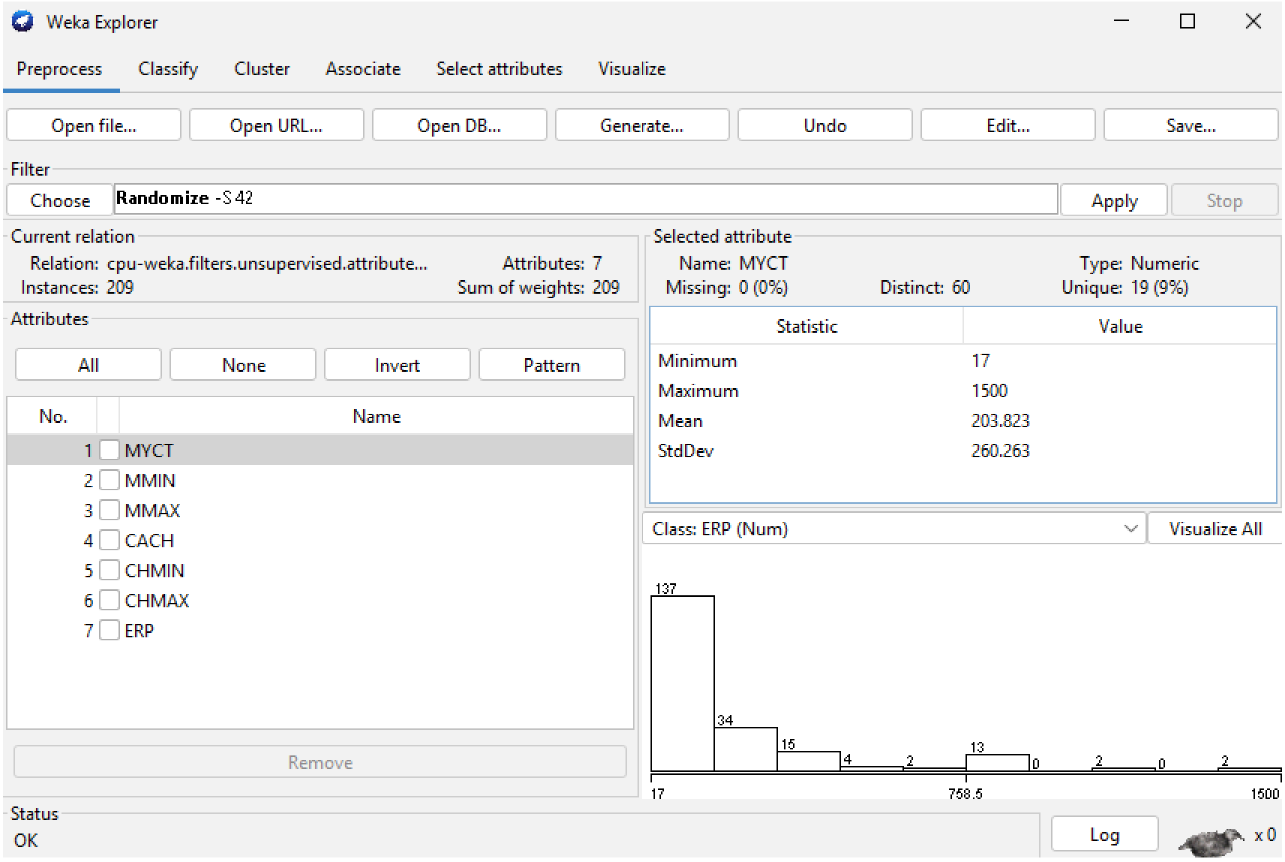
In this section we use a CPU Performance data set. The task is to predict the Estimated Relative Performance (ERP) based on a number of attributes, namely vendor, MYCT, MMIN, MMAX, CACH, CHMIN, CHMAX. More information about this data set can be found here.

1. Download the data set cpu.arff and load it into Weka as usual



2. Notice that the vendor attribute is nominal, not numeric. This will give problems when using a linear regression model. **For now we can simply remove this attribute.** Select the check box for *vendor* and click *Remove*.

```
Attribute Information:
%    1. vendor name: 30
%       (adviser, amdahl,apollo, basf, bti, burroughs, c.r.d, cambex, cdc, dec,
%        dg, formation, four-phase, gould, honeywell, hp, ibm, ipl, magnuson,
%        microdata, nas, ncr, nixdorf, perkin-elmer, prime, siemens, sperry,
%        sratus, wang)
%    2. MYCT: machine cycle time in nanoseconds (integer)
%    3. MMIN: minimum main memory in kilobytes (integer)
%    4. MMAX: maximum main memory in kilobytes (integer)
%    5. CACH: cache memory in kilobytes (integer)
%    6. CHMIN: minimum channels in units (integer)
%    7. CHMAX: maximum channels in units (integer)
%   8. ERP: estimated relative performance from the original article (integer)
```

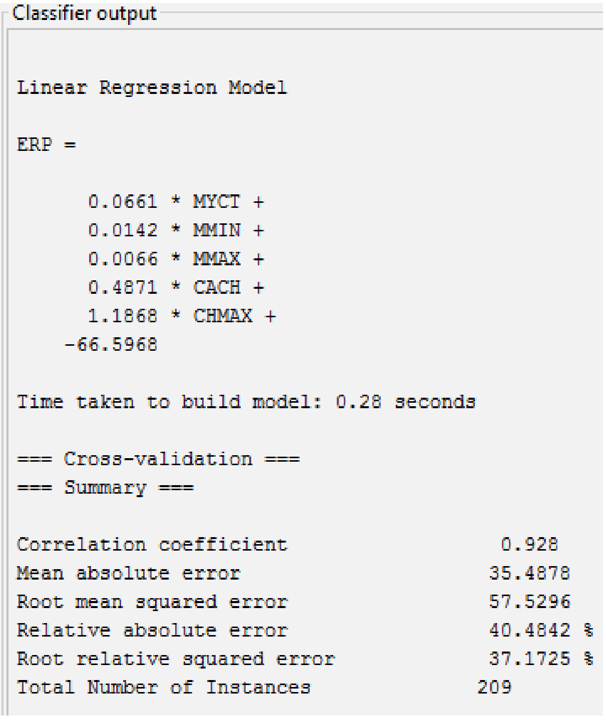


3. Now use the *Visualize* tab to explore the data. First look at plots of the input attributes against the **target variable ERP** (shown in the top row of scatterplots) and then look at plots of pairs of input attributes.
    - Do you think that ERP should be at least partially predictable from the input attributes?
    - Do any attributes exhibit significant correlations?

      In this case, it is difficult to emphasize any correlation, and I think that it is absent. Also, it is difficult to single out a specific attribute that should help in ERP forecasting, although I believe that MYCT is a less important feature.

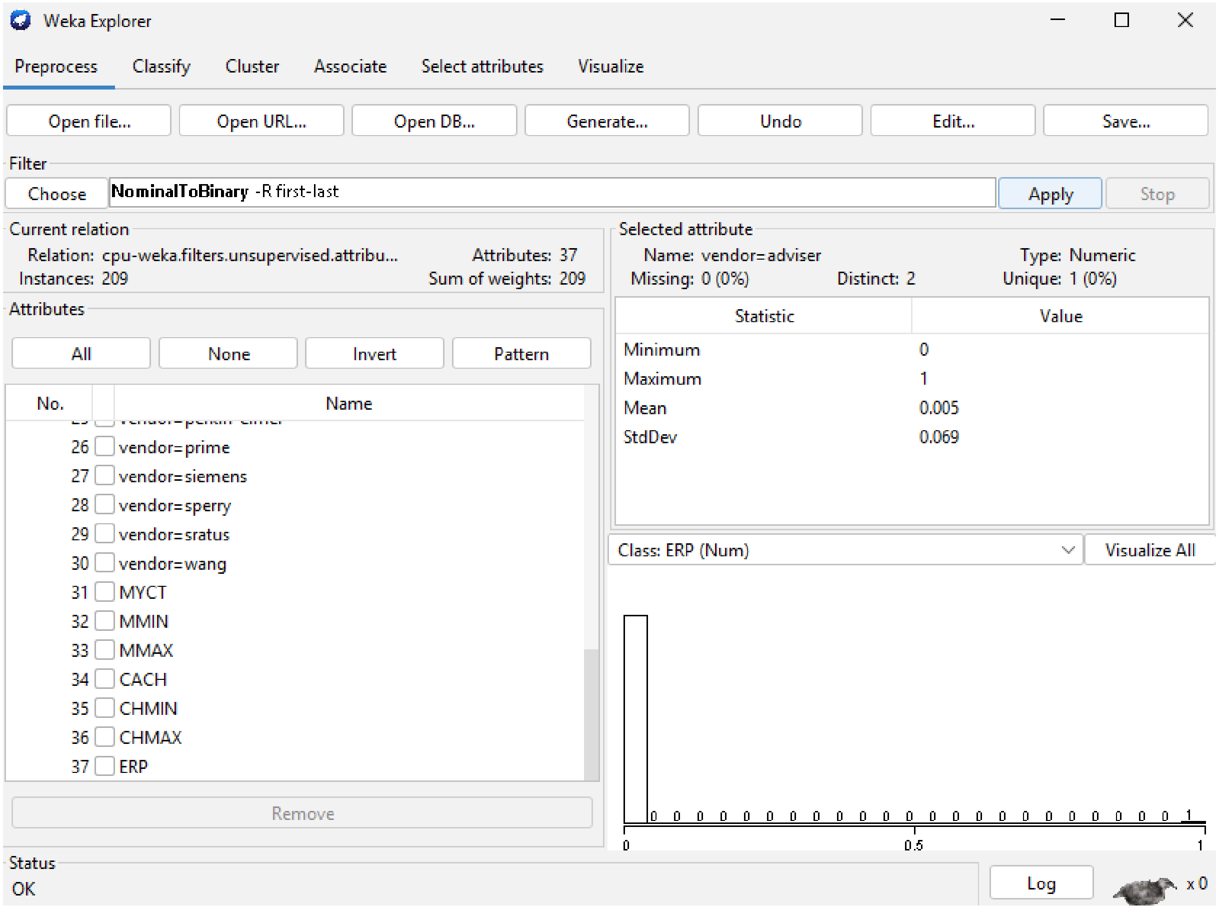
4. Now we have a feel for the data and we will try fitting a simple linear regression model to the data. On the *Classify* tab, select *Choose > functions > LinearRegression*. Use the default options and click *Start*. This will use 10-fold cross-validation to fit the linear regression model.

   Examine the results:
    - Record the Root relative squared error and the Relative absolute error. The Relative squared error is computed by dividing (normalizing) the sum of the squared prediction errors by the sum of the prediction errors obtained by always predicting the mean. The Root relative squared error is obtained by taking the square root of the Relative squared error. The Relative absolute error is similar to the Relative squared error, but uses absolute values rather than squares. See Table 5.8 in Witen and Frank (second edition). Therefore, if we have a relative error of 100%, the learned model is no better than this very dumb predictor.



```
Classifier output

Linear Regression Model

ERP =

      0.0661 * MYCT +
      0.0142 * MMIN +
      0.0066 * MMAX +
      0.4871 * CACH +
      1.1868 * CHMAX +
    -66.5968

Time taken to build model: 0.28 seconds

=== Cross-validation ===
=== Summary ===

Correlation coefficient                  0.928
Mean absolute error                     35.4878
Root mean squared error                 57.5296
Relative absolute error                 40.4842 %
Root relative squared error             37.1725 %
Total Number of Instances              209
```

5. Above we deleted the *vendor* variable. However, we can use nominal attributes in regression by converting them to numeric. The standard way of so doing is to replace the nominal variable with a bunch of binary variables of the form *"is_first_nominal_value, is_second_nominal_value"*and so on. Reload the unmodified data file cpu.arff. On the *Preprocess* tab select *Choose > filters > unsupervised > attribute > NominaltoBinary* and click *Apply*. This replaces the *vendor* variable with 30 binary variables and we now have 37 attributes (we started with 8).



6. Now train a linear regression model as in (4) and examine the results.
   - Record the Relative absolute error and the Root relative squared error.
   - Compare the performance to the one we had previously. Did adding the binarized *vendor* variable help?

Classifier output

```
Linear Regression Model

ERP =

   -132.1272 * vendor=adviser +
    -34.3319 * vendor=burroughs +
    -52.3128 * vendor=gould +
    -35.8202 * vendor=honeywell +
    -16.7597 * vendor=ibm +
   -144.1856 * vendor=microdata +
    -22.7172 * vendor=nas +
     41.5185 * vendor=sperry +
      0.0696 * MYCT +
      0.0167 * MMIN +
      0.0055 * MMAX +
      0.6304 * CACH +
     -1.5416 * CHMIN +
      1.6106 * CHMAX +
    -57.432

Time taken to build model: 0.07 seconds

=== Cross-validation ===
=== Summary ===

Correlation coefficient                  0.9252
Mean absolute error                     35.9725
Root mean squared error                 58.5821
Relative absolute error                 41.0372 %
Root relative squared error             37.8525 %
Total Number of Instances              209
```

Unfortunately, the categorization of the vendor feature did not help.

7. So far, we have made use of Linear Regression. One could also try fitting nonlinear models. If you have time, experiment with a non-linear predictor that has been discussed in class - a kNN regression *IBk*. Do you get better performance with this non-linear predictor?

Classifier output

```
              vendor=siemens
              vendor=sperry
              vendor=sratus
              vendor=wang
              MYCT
              MMIN
              MMAX
              CACH
              CHMIN
              CHMAX
              ERP
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

IB1 instance-based classifier
using 1 nearest neighbour(s) for classification


Time taken to build model: 0 seconds

=== Cross-validation ===
=== Summary ===

Correlation coefficient                  0.9467
Mean absolute error                     20.8278
Root mean squared error                 53.6354
Relative absolute error                 23.7602 %
Root relative squared error             34.6563 %
Total Number of Instances              209
```

Since our data is more complex for a linear predictor, we need a non-linear one, and of course it is expected to behave better.