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

№1.3

Work with WEKA. SVM CLASSIFICATION AND EVALUATION

(link: https://docs.google.com/document/d/1ZnGXH5Qcp61A2tMVpXK29_wUx-hRZOep/edit)

The work was done by
Zvychaynaya Anastasia

The work was checked by
Alexander Oriekhov

Kyiv 2022

Execution of work:

1. SPAM FILTERING

1. Start up Weka, select the Explorer interface and load the preprocessed Spambase data set from Lab 1, where all attributes are converted to Boolean and randomize the instances. **Cheat:** If you did not save this data set, download it here.
2. Now it's time to train our classifiers. The task is to classify e-mails as spam or non-spam and we evaluate the performance of Logistic Regression and Support Vector Machines on this task. Go to the *Classify* tab and select *Choose > functions > SimpleLogistic*. Select the percentage split and set it to 10%. This is done in order to save us waiting while Weka works hard on a large data set.

   Click *Start* to train the model. Examine the *Classifier output* frame to view information for the model you've just trained and try to answer the following questions:

   - What is the percentage of correctly classified instances?

```
Classifier output

=== Evaluation on test split ===

Time taken to test model on test split: 0.02 seconds

=== Summary ===

Correctly Classified Instances        3727               90.0024 %
Incorrectly Classified Instances       414                9.9976 %
Kappa statistic                          0.7915
Mean absolute error                      0.1485
Root mean squared error                  0.2662
Relative absolute error                 31.3579 %
Root relative squared error             54.3506 %
Total Number of Instances             4141

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Clas
                 0,913    0,120    0,921      0,913    0,917      0,792    0,961     0,969     0
                 0,880    0,087    0,869      0,880    0,875      0,792    0,961     0,943     1
Weighted Avg.    0,900    0,107    0,900      0,900    0,900      0,792    0,961     0,959

=== Confusion Matrix ===

    a    b   <-- classified as
 2282  217 |    a = 0
  197 1445 |    b = 1
```

   - How do the regression coefficients for class 1 relate to the ones for class 0? Can you derive this result from the form of the Logistic Regression model?

   The ratio of the coefficient of class 1 to the coefficient of class 2 is -1. You can see comparative example below.

```
Class 0 :
0.93 +
[word_freq_make_binarized=1] * 0.12 +
[word_freq_all_binarized=1] * 0.17 +
[word_freq_3d_binarized=1] * -0.39 +
[word_freq_our_binarized=1] * -0.56 +
[word_freq_over_binarized=1] * -0.12 +
[word_freq_remove_binarized=1] * -1.15 +
```

```
Class 1 :
-0.93 +
[word_freq_make_binarized=1] * -0.12 +
[word_freq_all_binarized=1] * -0.17 +
[word_freq_3d_binarized=1] * 0.39 +
[word_freq_our_binarized=1] * 0.56 +
[word_freq_over_binarized=1] * 0.12 +
[word_freq_remove_binarized=1] * 1.15 +
```

- Write down the coefficients for class 1 for the attributes `[word_freq_hp_binarized]` and `[char_freq_$_binarized]`. Generally, we would expect the string `$` to appear in spam, and the string `hp` to appear in non-spam e-mails, as the data was collected from HP Labs.

For [word_freq_hp_binarized=1] is -1.46 and for [char_freq_$_binarized=1] is 0.92.

Do the regression coefficients make sense given that class 1 is spam? *Hint:* Consider the sigmoid function and how it transforms values into a probability between 0 and 1. Since our attributes are boolean, a positive coefficient can only increase the total sum fed through the sigmoid and thus move the output of the sigmoid towards 1. What can happen if we have continuous, real-valued attributes?

The sign can be positive or negative, it indicates the position of a single point in relation to the line (plane/hyperplane) that separates the classes. It is the sign of the logistic regression equation that allows us to attribute the object to the upper or lower subspace, and therefore to one of the classes. In linear regression, the sign is ignored, but in classification, we only need it!
The value of 0.92 thus obtained tells us that the *$* sign is spam, and -1.46 tells us that the *hp* is non-spam.

3. We will now train a Support Vector Machine (SVM) on our classification task. In the *Classify* tab, select *Choose > functions > SMO* (SMO stands for Sequential Minimal Optimization, which is an algorithm for training SVMs). Use the default parameters and click *Start*. This will train a linear SVM (which is quite similar to logistic regression). Again, examine the *Classifier output* frame and try answering the following:

- What is the percent of correctly classified instances? How does it compare to the result from Logistic Regression?

SVM gives a better result than the result of Logistic Regression, but the difference in accuracy is negligible (less than one percent).

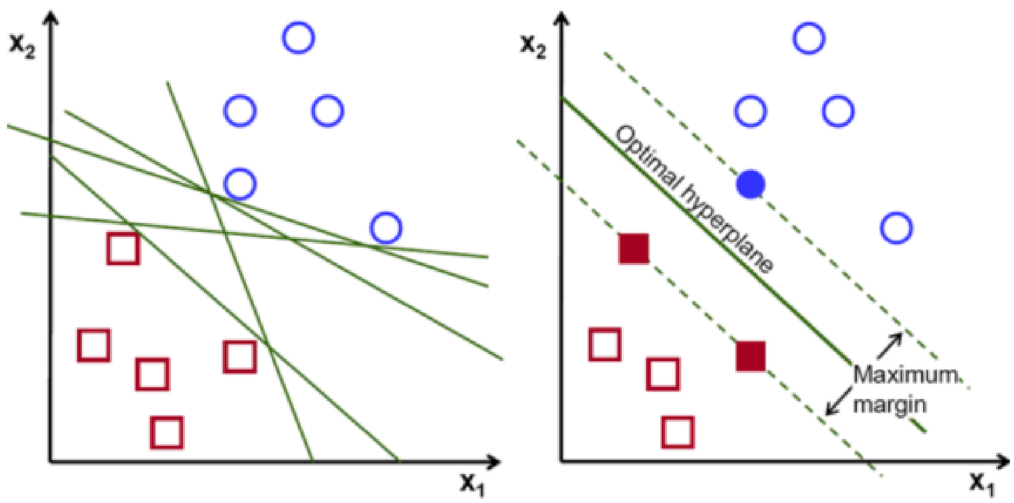
Using the default SVM:

```
Classifier output

Time taken to build model: 3.54 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0.16 seconds

=== Summary ===

Correctly Classified Instances        3766               90.9442 %
Incorrectly Classified Instances       375                9.0558 %
Kappa statistic                          0.8094
Mean absolute error                      0.0906
Root mean squared error                  0.3009
Relative absolute error                 19.1223 %
Root relative squared error             61.4422 %
Total Number of Instances             4141

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Clas
                 0,939    0,136    0,913      0,939    0,926      0,810    0,902     0,894     0
                 0,864    0,061    0,903      0,864    0,883      0,810    0,902     0,834     1
Weighted Avg.    0,909    0,106    0,909      0,909    0,909      0,810    0,902     0,871

=== Confusion Matrix ===

    a    b   <-- classified as
 2347  152 |    a = 0
  223 1419 |    b = 1
```

- What are the coefficients for the attributes `[word_freq_hp_binarized]` and `[char_freq_$_binarized]`? Compare these to the ones you found with Logistic Regression.

  For [word_freq_hp_binarized=1] is -2.0448 and for [char_freq_$_binarized=1] is 1.3794. This is more accurate than with a logistic regression model.

How does a linear SVM relate to Logistic Regression? *Hint:* Consider the classification boundary learnt in each model.



The left plot shows the decision boundaries of five possible linear classifiers. The solid line in the in the right plot represents the decision boundary of SVM classifier; this boundary not only separates the two classes but also stays as far away from the closest training instances as possible

2. PERFORMANCE ASSESSMENT #1

We will now look at a few ways of assessing the performance of a classifier. To do so we will introduce a new data set, the Splice data set. The classification task is to identify *intron* and *exon* boundaries on gene sequences. Read the description at the link above for a brief overview of how this works. The class attribute can take on 3 values: N, IE and EI. Now download the data sets below, converted into ARFF for you, and load the training set into Weka:

- splice_train.arff: training data
- splice_test.arff: test data

1. We'll also use a new classifier. Under the *Classify* tab, select *classifiers* > *lazy* > *IBk*. This is a K-nearest neighbour classifier.
2. In the *Test options* panel, select *Use training set* and hit *Start*.

Classifier output

```
=== Evaluation on training set ===

Time taken to test model on training data: 2.92 seconds

=== Summary ===

Correctly Classified Instances        2934               99.9659 %
Incorrectly Classified Instances         1                0.0341 %
Kappa statistic                          0.9994
Mean absolute error                      0.0007
Root mean squared error                  0.0107
Relative absolute error                  0.159  %
Root relative squared error              2.3499 %
Total Number of Instances             2935

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1,000    0,001    0,999      1,000    1,000      0,999    1,000     1,000     N
                 1,000    0,000    1,000      1,000    1,000      1,000    1,000     1,000     EI
                 0,999    0,000    1,000      0,999    0,999      0,999    1,000     1,000     IE
Weighted Avg.    1,000    0,000    1,000      1,000    1,000      0,999    1,000     1,000

=== Confusion Matrix ===

    a    b    c   <-- classified as
 1506    0    0 |    a = N
    0  715    0 |    b = EI
    1    0  713 |    c = IE
```

3. Observe the output of the classifier and consider the following:

   - What is the classification accuracy?

     Oh, the accuracy is so high. It is rare to face it in the real life.

   - Is this meaningful?

     Let me think, let me see the default parameters of the model - we are using KNN, where K = 1, so we are actually using nearest neighbor classification, the main problem of which is overfitting - so I am sure that we are facing overfitting.

- Why is testing on the training data a particularly bad idea for a 1-nearest neighbour classifier?

  Because of overfitting.

- Do you expect the performance of the classifier on a test set to be as good?

  I do not expect it, because our model is overfitted.

4. Now evaluate the classifier on the test set and check your expectations. In the *Test options* panel, select *Supplied test set* and load the file *splice_test.arff*. In the *Result list* panel, right-click on the classifier and select *Re-evaluate model on current test set*.

```
Classifier output

=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.52 seconds

=== Summary ===

Correctly Classified Instances         200               78.4314 %
Incorrectly Classified Instances        55               21.5686 %
Kappa statistic                          0.656
Mean absolute error                      0.1657
Root mean squared error                  0.3757
Relative absolute error                 41.5002 %
Root relative squared error             85.4537 %
Total Number of Instances              255

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,705    0,066    0,938      0,705    0,805      0,634    0,845     0,870     N
                 0,981    0,138    0,646      0,981    0,779      0,734    0,938     0,686     EI
                 0,815    0,100    0,688      0,815    0,746      0,674    0,879     0,675     IE
Weighted Avg.    0,784    0,088    0,825      0,784    0,787      0,663    0,871     0,791

=== Confusion Matrix ===

   a   b   c   <-- classified as
 105  24  20 |   a = N
   1  51   0 |   b = EI
   6   4  44 |   c = IE
```

Observe the output and consider the following:

- What would be the accuracy of the classifier, if all points were labelled as *N*? *Hint:* View the distribution of the *class* attribute of the test data. You can do this by loading the test data on the *Preprocess* tab, and selecting the *class* attribute in the *Attributes* panel.

If we load a test sample where all observations have the target label N, then the accuracy will be less because the label N is more often mistaken (44/149 vs 1/51 vs 10/54).

5. Now explore the effect of the *k* parameter. To do this, train the classifier multiple times, each time setting the *KNN* option to a different value. Try 5, 10, 100, 1000 and 10000 and test the classifier on the test set. *Hint:* To change the *KNN* option you need to bring up the options panel of the classifier.

K=5:

Classifier output

```
Time taken to test model on supplied test set: 0.74 seconds

=== Summary ===

Correctly Classified Instances         212               83.1373 %
Incorrectly Classified Instances        43               16.8627 %
Kappa statistic                          0.7292
Mean absolute error                      0.2066
Root mean squared error                  0.3116
Relative absolute error                 51.7215 %
Root relative squared error             70.8692 %
Total Number of Instances              255

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,752    0,028    0,974      0,752    0,848      0,716    0,954     0,968     N
                 0,981    0,133    0,654      0,981    0,785      0,741    0,976     0,878     EI
                 0,907    0,065    0,790      0,907    0,845      0,803    0,975     0,933     IE
Weighted Avg.    0,831    0,057    0,870      0,831    0,835      0,740    0,963     0,942

=== Confusion Matrix ===

   a   b   c   <-- classified as
 112  24  13 |   a = N
   1  51   0 |   b = EI
   2   3  49 |   c = IE
```

K=10:

Classifier output

```
Time taken to test model on supplied test set: 0.57 seconds

=== Summary ===

Correctly Classified Instances         218               85.4902 %
Incorrectly Classified Instances        37               14.5098 %
Kappa statistic                          0.7658
Mean absolute error                      0.2295
Root mean squared error                  0.3062
Relative absolute error                 57.4755 %
Root relative squared error             69.6444 %
Total Number of Instances              255

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,779    0,009    0,991      0,779    0,872      0,761    0,977     0,984     N
                 1,000    0,123    0,675      1,000    0,806      0,770    0,982     0,932     EI
                 0,926    0,055    0,820      0,926    0,870      0,834    0,985     0,956     IE
Weighted Avg.    0,855    0,042    0,891      0,855    0,858      0,778    0,980     0,967

=== Confusion Matrix ===

   a   b   c   <-- classified as
 116  22  11 |   a = N
   0  52   0 |   b = EI
   1   3  50 |   c = IE
```

K=100:

```
Classifier output

Time taken to test model on supplied test set: 0.8 seconds

=== Summary ===

Correctly Classified Instances         232               90.9804 %
Incorrectly Classified Instances        23                9.0196 %
Kappa statistic                          0.8505
Mean absolute error                      0.2795
Root mean squared error                  0.3165
Relative absolute error                 69.9925 %
Root relative squared error             71.9835 %
Total Number of Instances              255

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,859    0,000    1,000      0,859    0,924      0,847    0,997     0,998     N
                 1,000    0,079    0,765      1,000    0,867      0,839    0,994     0,978     EI
                 0,963    0,035    0,881      0,963    0,920      0,899    0,997     0,989     IE
Weighted Avg.    0,910    0,023    0,927      0,910    0,912      0,856    0,996     0,992

=== Confusion Matrix ===

   a   b   c   <-- classified as
 128  14   7 |   a = N
   0  52   0 |   b = EI
   0   2  52 |   c = IE
```

K=1000:

```
Classifier output

Time taken to test model on supplied test set: 0.73 seconds

=== Summary ===

Correctly Classified Instances         231               90.5882 %
Incorrectly Classified Instances        24                9.4118 %
Kappa statistic                          0.8256
Mean absolute error                      0.3541
Root mean squared error                  0.382
Relative absolute error                 88.6626 %
Root relative squared error             86.8871 %
Total Number of Instances              255

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1,000    0,208    0,871      1,000    0,931      0,831    0,989     0,993     N
                 0,731    0,010    0,950      0,731    0,826      0,799    0,993     0,977     EI
                 0,815    0,000    1,000      0,815    0,898      0,881    0,995     0,985     IE
Weighted Avg.    0,906    0,123    0,915      0,906    0,903      0,835    0,991     0,988

=== Confusion Matrix ===

   a   b   c   <-- classified as
 149   0   0 |   a = N
  14  38   0 |   b = EI
   8   2  44 |   c = IE
```

K=10000:

Classifier output

```
Time taken to test model on supplied test set: 0.91 seconds

=== Summary ===

Correctly Classified Instances         149               58.4314 %
Incorrectly Classified Instances       106               41.5686 %
Kappa statistic                          0
Mean absolute error                      0.3993
Root mean squared error                  0.4396
Relative absolute error                 99.9885 %
Root relative squared error             99.9966 %
Total Number of Instances              255

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1,000    1,000    0,584      1,000    0,738      ?        0,500     0,584     N
                 0,000    0,000    ?          0,000    ?          ?        0,500     0,204     EI
                 0,000    0,000    ?          0,000    ?          ?        0,500     0,212     IE
Weighted Avg.    0,584    0,584    ?          0,584    ?          ?        0,500     0,428

=== Confusion Matrix ===

   a   b   c   <-- classified as
 149   0   0 |   a = N
  52   0   0 |   b = EI
  54   0   0 |   c = IE
```

- How does the *k* parameter effect the results? *Hint:* Consider how well the classifier is generalising to previously unseen data, and how it compares to the base rate again.

In the beginning, we had a too small K value, which led to overfitting of the model. As long as the value of the parameter K increased, each time we had an increasingly accurate prediction; however, a critical moment came when the accuracy dropped, and in this situation we faced underfitting of the model.

Thus, it is important to find the optimal value of K.