Disciplina: Data Mining **Data:** 4/11/2016

Professor: Geraldo Zimbrão da Silva

Aluno: Leniel Macaferi **DRE:** 116005784

Mestrado em Engenharia de Dados e Conhecimento @ COPPE\UFRJ - Universidade Federal do Rio de Janeiro

Report
Classification

Classification is a data mining function that assigns items in a collection to target categories or classes. The goal of classification is to accurately predict the target class for each case in the data. For example, a classification model could be used to identify loan applicants as low, medium, or high credit risks.
https://docs.oracle.com/cd/B28359_01/datamine.111/b28129/classify.htm

1 - Problem

Classify the data contained in a spreadsheet called classification.xlsx using multiple classifiers having the Naive Bayes classifier as the baseline.

The expected result is to predict the classes for unlabelled\unclassified samples (spreadsheet rows) identified with a question mark ? in the column "Classe".

The output of the classifier must be in the format: id - Classe.

2 - Data

The data used for training the model comprised features (spreadsheet columns) X1 to X12 and Classe.

No data normalization was performed.

There are 10793 samples\instances in the spreadsheet from which 8618 are already classified into three classes [C, N, V] and 2175 need to be classified.

The following table displays the number of instances in each class along with the number of samples instances to be classified. The percentage in each group is also shown.

C	N	V	?
2767	3335	2516	2175
25.63%	30.89%	23.31%	20.15%

Table 1 - Number and percentage of samples\instances per class

3 - Development

3.1 - Tools

The following is the listing of tools used to execute this work:

Microsoft Excel

IDEs: Weka and Sublime Text

All the files including the input and output of for this work is stored on a Github repository @ https://github.com/leniel/DataMining

3.2 - Process

Data\number crunching was done in Microsoft Excel before serving as input to the classifiers' algorithms.

Data was partitioned into 2 groups: classified and unclassified instances (spreadsheet rows).

The first group containing only instances already classified was taken out and saved in the .csv format in a file called classification-training.csv. After that the CSV file was edited using Sublime Text 3 and its plugin Sublime-Text-Advanced-CSV. With the help of this plugin the column id was removed.

The second group containing only samples unclassified represented by an interrogation mark (?) was taken out and saved in the .CSV format in a file called classification-test.csv. The column id was removed as done previously.

Later on the two CSV files were imported into Weka.

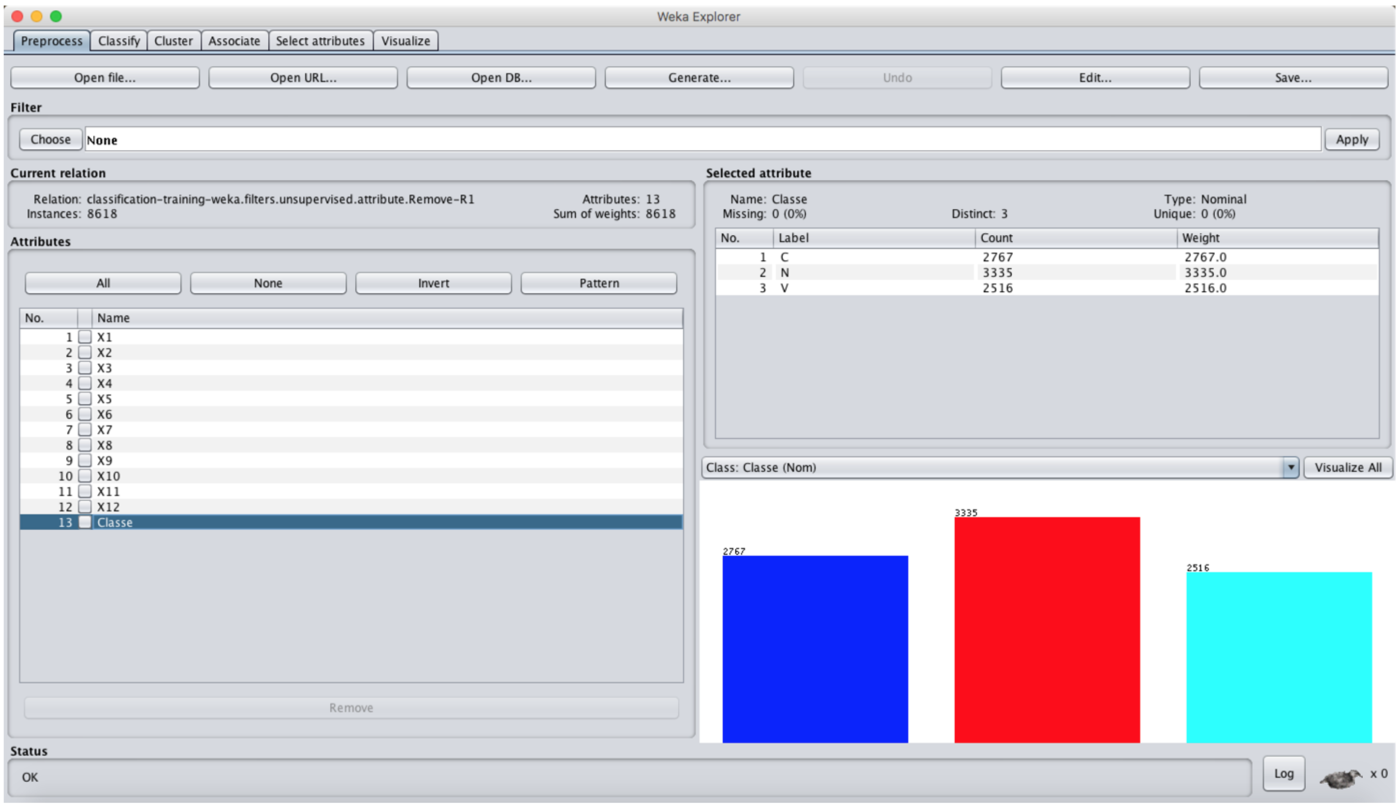
Weka is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes.

Waikato Environment for Knowledge Analysis
Version 3.8.0
(c) 1999 - 2016
The University of Waikato
Hamilton, New Zealand

Figure 1 - Weka’s GUI Chooser

Using Weka’s Explorer application the training data was imported for preprocessing. To play with the data I converted it to Weka’s own file format called arff (Attribute-Relation File Format) to make sure the data is ok. The new file is called classification-training.arff.

Training data was used to build the model from which prediction will be made using the test data.



Relation: classification-training-weka.filters.unsupervised.attribute.Remove-R1
Instances: 8618
Attributes: 13
Sum of weights: 8618

Name: Classe
Missing: 0 (0%)
Distinct: 3
Type: Nominal
Unique: 0 (0%)

No.	Label	Count	Weight
1	C	2767	2767.0
2	N	3335	3335.0
3	V	2516	2516.0

No.	Name
1	X1
2	X2
3	X3
4	X4
5	X5
6	X6
7	X7
8	X8
9	X9
10	X10
11	X11
12	X12
13	Classe

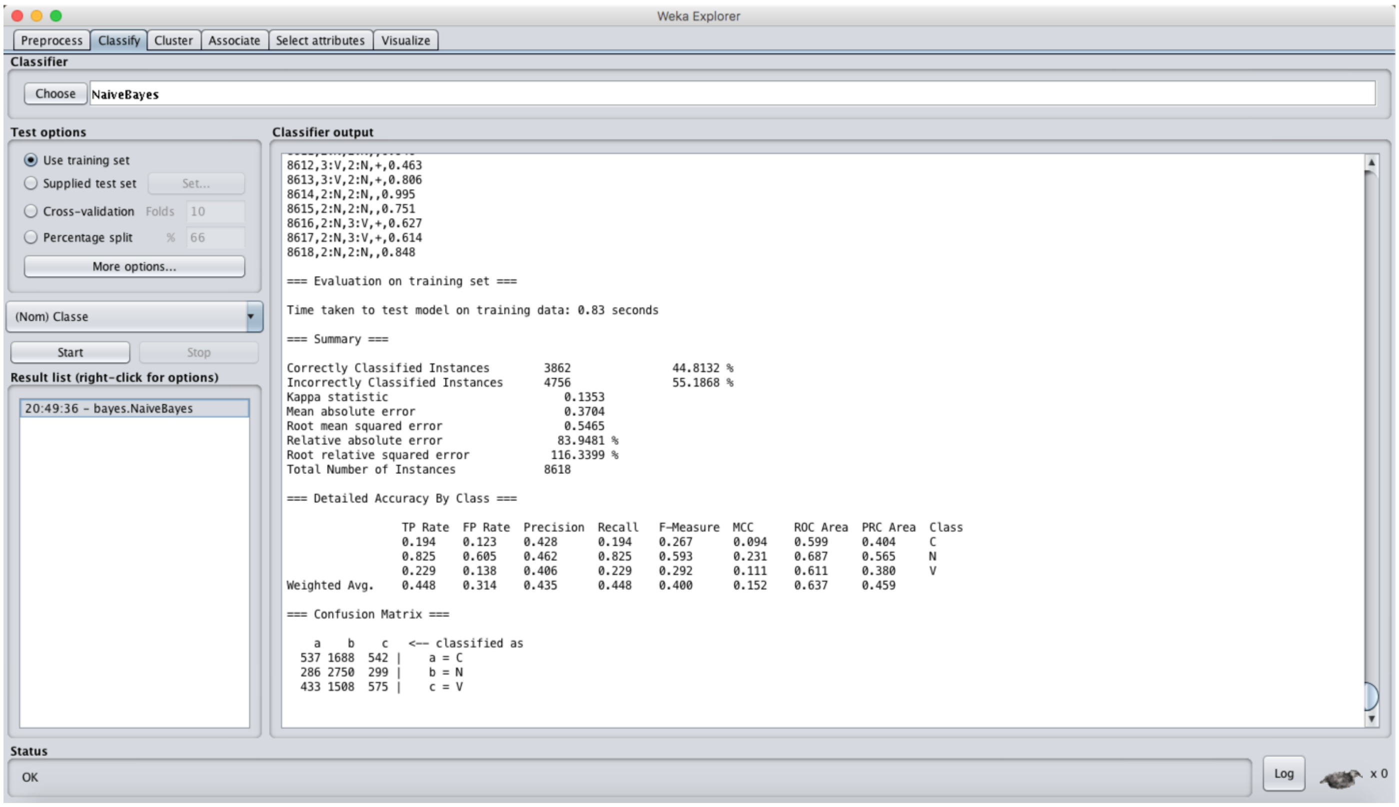
Figure 2 - Weka’s Preprocess tab\window listing attributes X1 to X12 and Classe

3.3 - Classification

Classification was performed using the Classify window in Weka.

A total of 8 different classifiers from multiple types were tested:

- Bagging
- Bayes Net
- IBk (KNN)
- J48
- Naive Bayes
- PART
- Random Forest
- SVM



Choose: NaiveBayes

(●) Use training set
( ) Supplied test set
( ) Cross-validation Folds 10
( ) Percentage split % 66

(Nom) Classe

20:49:36 - bayes.NaiveBayes

```
8612,3:V,2:N,+,0.463
8613,3:V,2:N,+,0.806
8614,2:N,2:N,,0.995
8615,2:N,2:N,,0.751
8616,2:N,3:V,+,0.627
8617,2:N,3:V,+,0.614
8618,2:N,2:N,,0.848

=== Evaluation on training set ===

Time taken to test model on training data: 0.83 seconds

=== Summary ===

Correctly Classified Instances        3862               44.8132 %
Incorrectly Classified Instances      4756               55.1868 %
Kappa statistic                          0.1353
Mean absolute error                      0.3704
Root mean squared error                  0.5465
Relative absolute error                 83.9481 %
Root relative squared error            116.3399 %
Total Number of Instances             8618

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.194    0.123    0.428      0.194    0.267      0.094    0.599     0.404     C
                 0.825    0.605    0.462      0.825    0.593      0.231    0.687     0.565     N
                 0.229    0.138    0.406      0.229    0.292      0.111    0.611     0.380     V
Weighted Avg.    0.448    0.314    0.435      0.448    0.400      0.152    0.637     0.459

=== Confusion Matrix ===

    a    b    c   <-- classified as
  537 1688  542 |    a = C
  286 2750  299 |    b = N
  433 1508  575 |    c = V
```

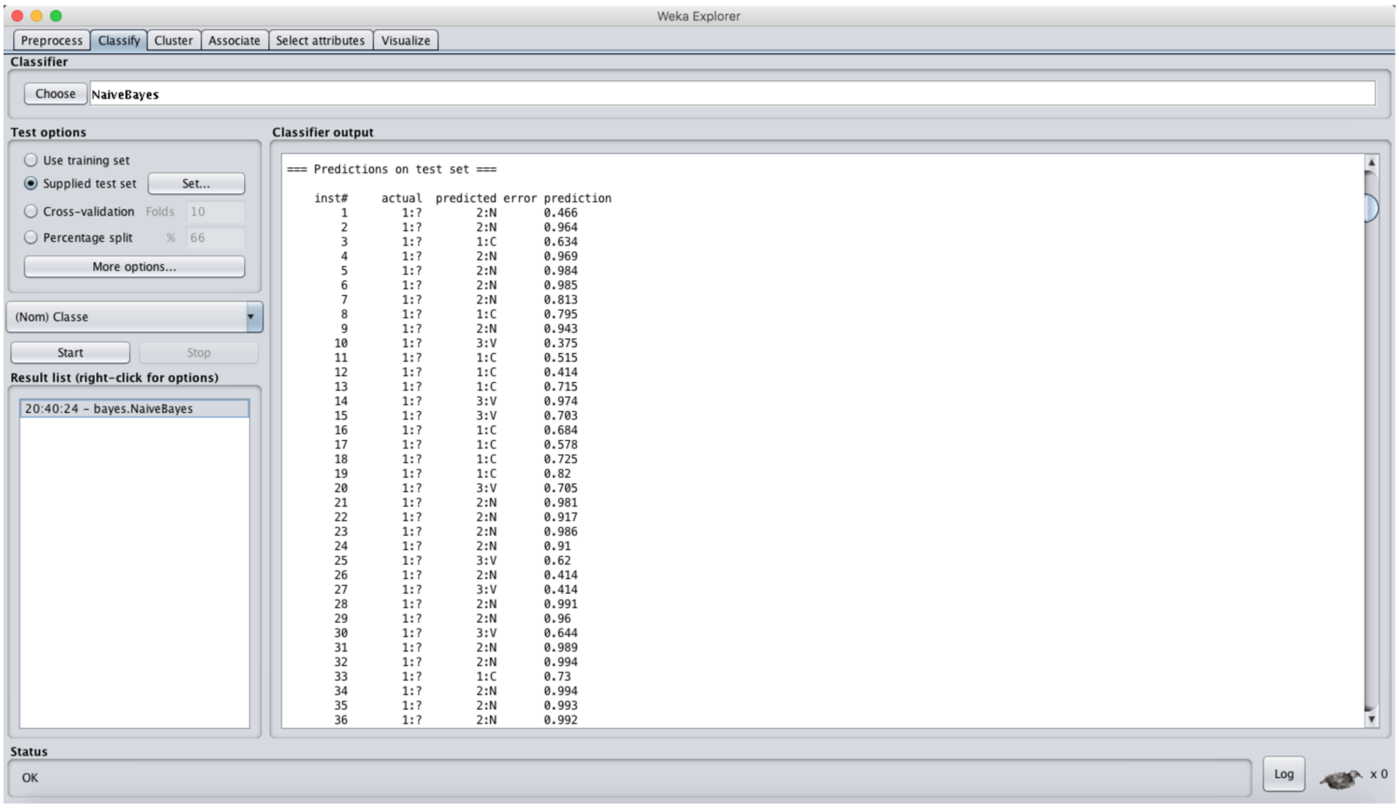
Status: OK

Figure 3 - Weka's Classify tab\window with Naive Bayes classifier chosen

For each classifier, the model was built using the training data (classification-training.arff) and classifier output data was taken out to a new spreadsheet called predictions.xlsx. In this spreadsheet further analysis of the output was performed.

3.4 - Predictions

For each classifier, the test file classification-test.arff was provided in Test Options (Supplied test set) in Weka.



Classifier: NaiveBayes

Test options: Supplied test set (selected); Cross-validation Folds 10; Percentage split % 66

(Nom) Classe

20:40:24 - bayes.NaiveBayes

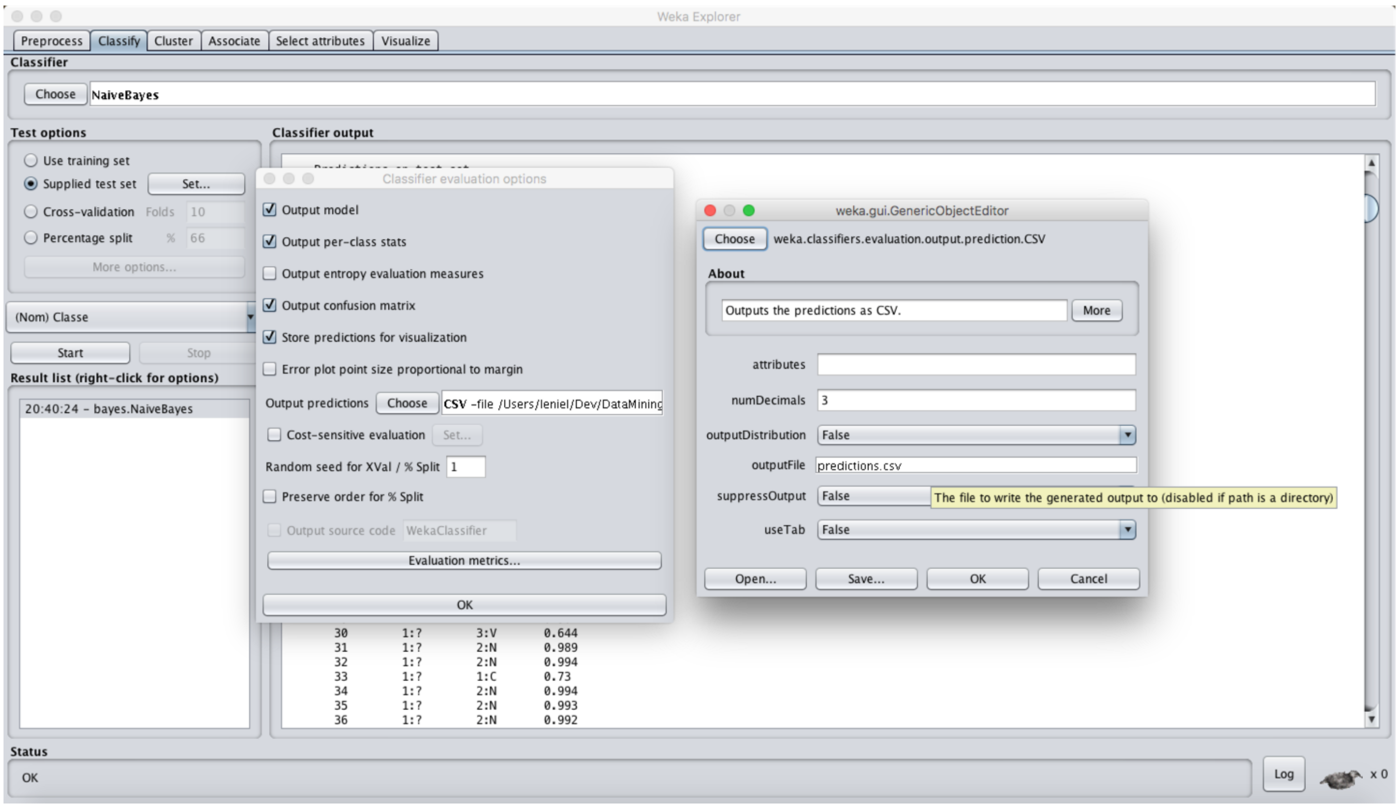
```
=== Predictions on test set ===

    inst#     actual  predicted error prediction
        1        1:?        2:N       0.466
        2        1:?        2:N       0.964
        3        1:?        1:C       0.634
        4        1:?        2:N       0.969
        5        1:?        2:N       0.984
        6        1:?        2:N       0.985
        7        1:?        2:N       0.813
        8        1:?        1:C       0.795
        9        1:?        2:N       0.943
       10        1:?        3:V       0.375
       11        1:?        1:C       0.515
       12        1:?        1:C       0.414
       13        1:?        1:C       0.715
       14        1:?        3:V       0.974
       15        1:?        3:V       0.703
       16        1:?        1:C       0.684
       17        1:?        1:C       0.578
       18        1:?        1:C       0.725
       19        1:?        1:C       0.82
       20        1:?        3:V       0.705
       21        1:?        2:N       0.981
       22        1:?        2:N       0.917
       23        1:?        2:N       0.986
       24        1:?        2:N       0.91
       25        1:?        3:V       0.62
       26        1:?        2:N       0.414
       27        1:?        3:V       0.414
       28        1:?        2:N       0.991
       29        1:?        2:N       0.96
       30        1:?        3:V       0.644
       31        1:?        2:N       0.989
       32        1:?        2:N       0.994
       33        1:?        1:C       0.73
       34        1:?        2:N       0.994
       35        1:?        2:N       0.993
       36        1:?        2:N       0.992
```

Status: OK

Figure 4 - Weka's Supplied test set option

While inside Weka's Classify windows, In More options... the output was set to a CSV file called predictions.csv according to Figure 5 below:



Classifier: NaiveBayes

Test options:
- Use training set
- [✓ Supplied test set]
- Cross-validation Folds 10
- Percentage split % 66

(Nom) Classe

Result list (right-click for options): 20:40:24 - bayes.NaiveBayes

Classifier evaluation options:
- [✓] Output model
- [✓] Output per-class stats
- [ ] Output entropy evaluation measures
- [✓] Output confusion matrix
- [✓] Store predictions for visualization
- [ ] Error plot point size proportional to margin
- Output predictions: CSV -file /Users/leniel/Dev/DataMining
- [ ] Cost-sensitive evaluation
- Random seed for XVal / % Split: 1
- [ ] Preserve order for % Split
- [ ] Output source code: WekaClassifier

weka.gui.GenericObjectEditor: weka.classifiers.evaluation.output.prediction.CSV

About: Outputs the predictions as CSV.

attributes	
numDecimals	3
outputDistribution	False
outputFile	predictions.csv
suppressOutput	False
useTab	False

The file to write the generated output to (disabled if path is a directory)

```
    30        1:?       3:V       0.644
    31        1:?       2:N       0.989
    32        1:?       2:N       0.994
    33        1:?       1:C       0.73
    34        1:?       2:N       0.994
    35        1:?       2:N       0.993
    36        1:?       2:N       0.992
```

Status: OK

Figure 5 - Setting the predictions output

The data in the predictions.csv was then transferred to the Predictions.xlsx.

4 - Results

The results are based on classification using both the training set versus 5-fold cross-validation for all the 8 classifiers listed in 3.3. It’s important to note that all classifications were run with default parameters.

The best prediction is shown in section 4.3.

4.1 - Correctly classified instances (%)

IBk (KNN) and Random Forest scored 100% when using the Training set.

SVN (47.96%) was the best performing classifier when using 5-fold cross-validation.

The chart in Figure 6 shows the output for each classifier.

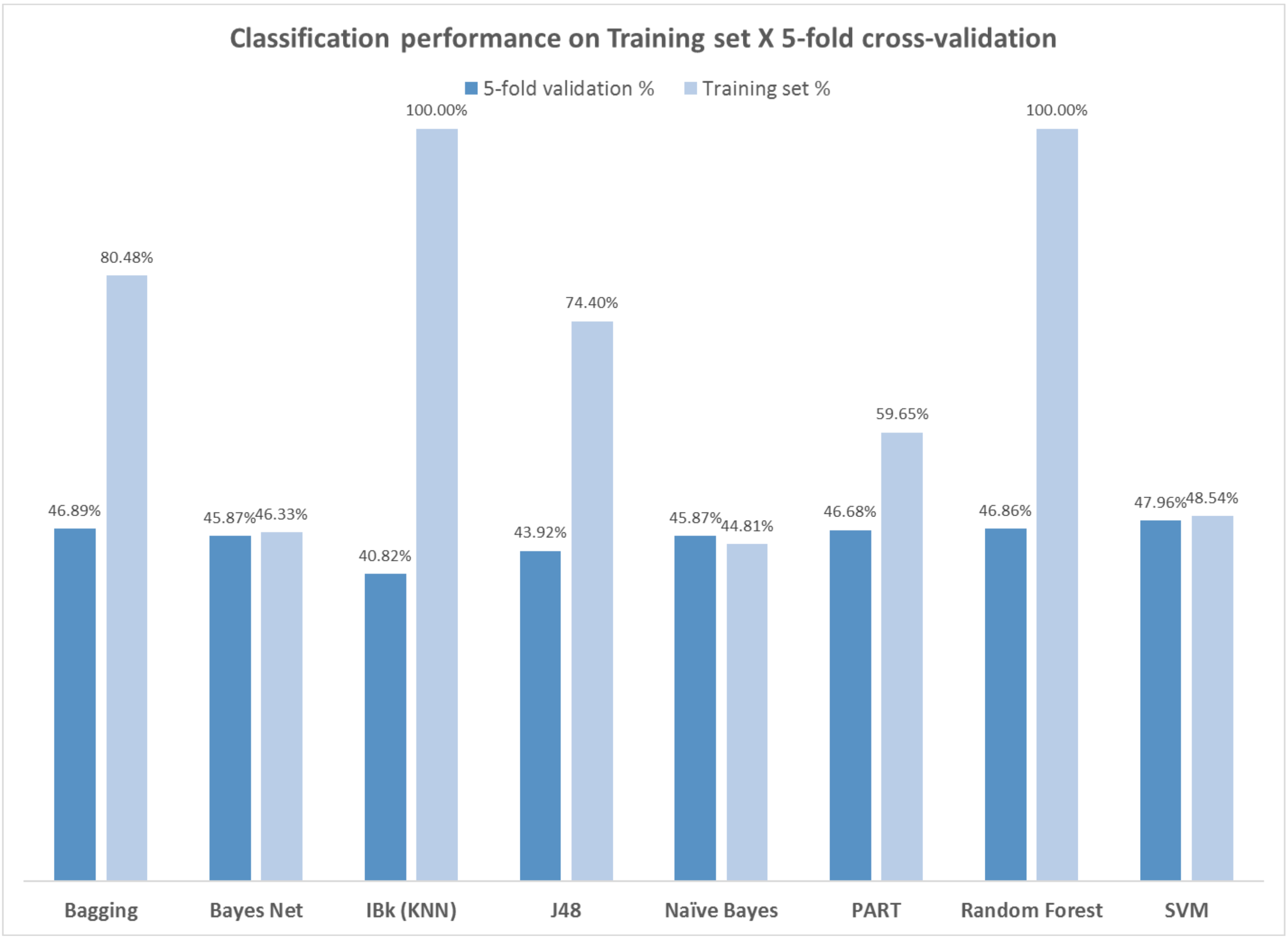


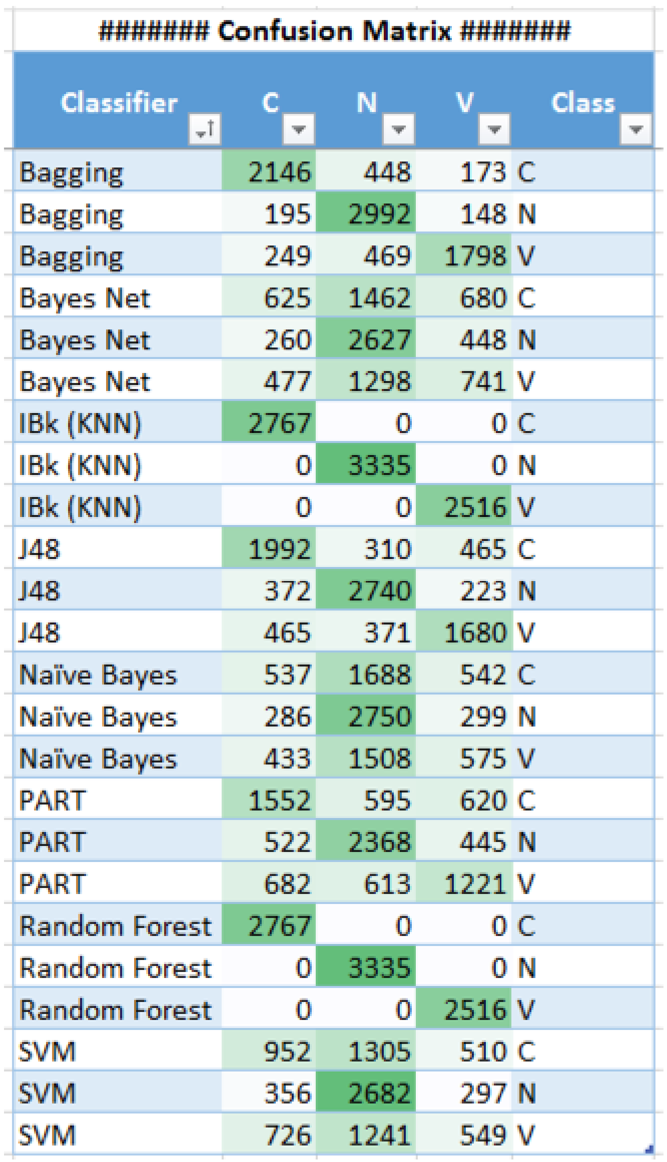
Figure 6 - Chart depicting the classifiers' accuracy

4.2 Confusion Matrix

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the correct values\classes are known.

Table 2 shows the confusion matrices obtained:

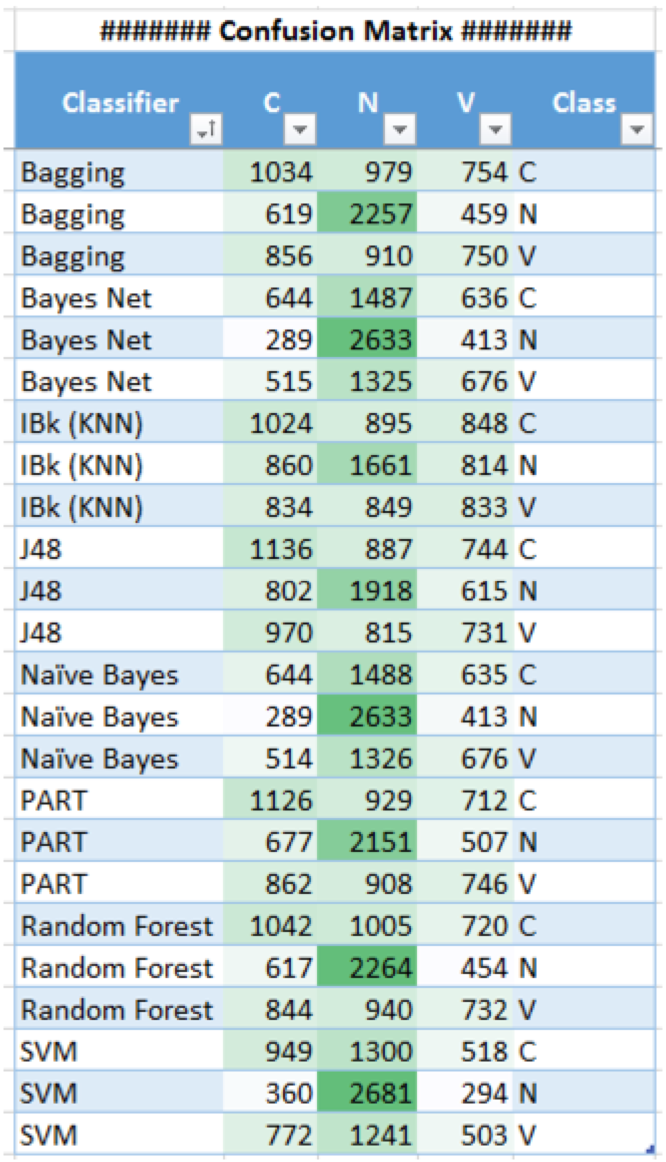
Using training set



####### Confusion Matrix #######

Classifier	C	N	V	Class
Bagging	2146	448	173	C
Bagging	195	2992	148	N
Bagging	249	469	1798	V
Bayes Net	625	1462	680	C
Bayes Net	260	2627	448	N
Bayes Net	477	1298	741	V
IBk (KNN)	2767	0	0	C
IBk (KNN)	0	3335	0	N
IBk (KNN)	0	0	2516	V
J48	1992	310	465	C
J48	372	2740	223	N
J48	465	371	1680	V
Naïve Bayes	537	1688	542	C
Naïve Bayes	286	2750	299	N
Naïve Bayes	433	1508	575	V
PART	1552	595	620	C
PART	522	2368	445	N
PART	682	613	1221	V
Random Forest	2767	0	0	C
Random Forest	0	3335	0	N
Random Forest	0	0	2516	V
SVM	952	1305	510	C
SVM	356	2682	297	N
SVM	726	1241	549	V

Using 5-folds cross-validation



####### Confusion Matrix #######

Classifier	C	N	V	Class
Bagging	1034	979	754	C
Bagging	619	2257	459	N
Bagging	856	910	750	V
Bayes Net	644	1487	636	C
Bayes Net	289	2633	413	N
Bayes Net	515	1325	676	V
IBk (KNN)	1024	895	848	C
IBk (KNN)	860	1661	814	N
IBk (KNN)	834	849	833	V
J48	1136	887	744	C
J48	802	1918	615	N
J48	970	815	731	V
Naïve Bayes	644	1488	635	C
Naïve Bayes	289	2633	413	N
Naïve Bayes	514	1326	676	V
PART	1126	929	712	C
PART	677	2151	507	N
PART	862	908	746	V
Random Forest	1042	1005	720	C
Random Forest	617	2264	454	N
Random Forest	844	940	732	V
SVM	949	1300	518	C
SVM	360	2681	294	N
SVM	772	1241	503	V

Table 2 - Classifiers' confusion matrices

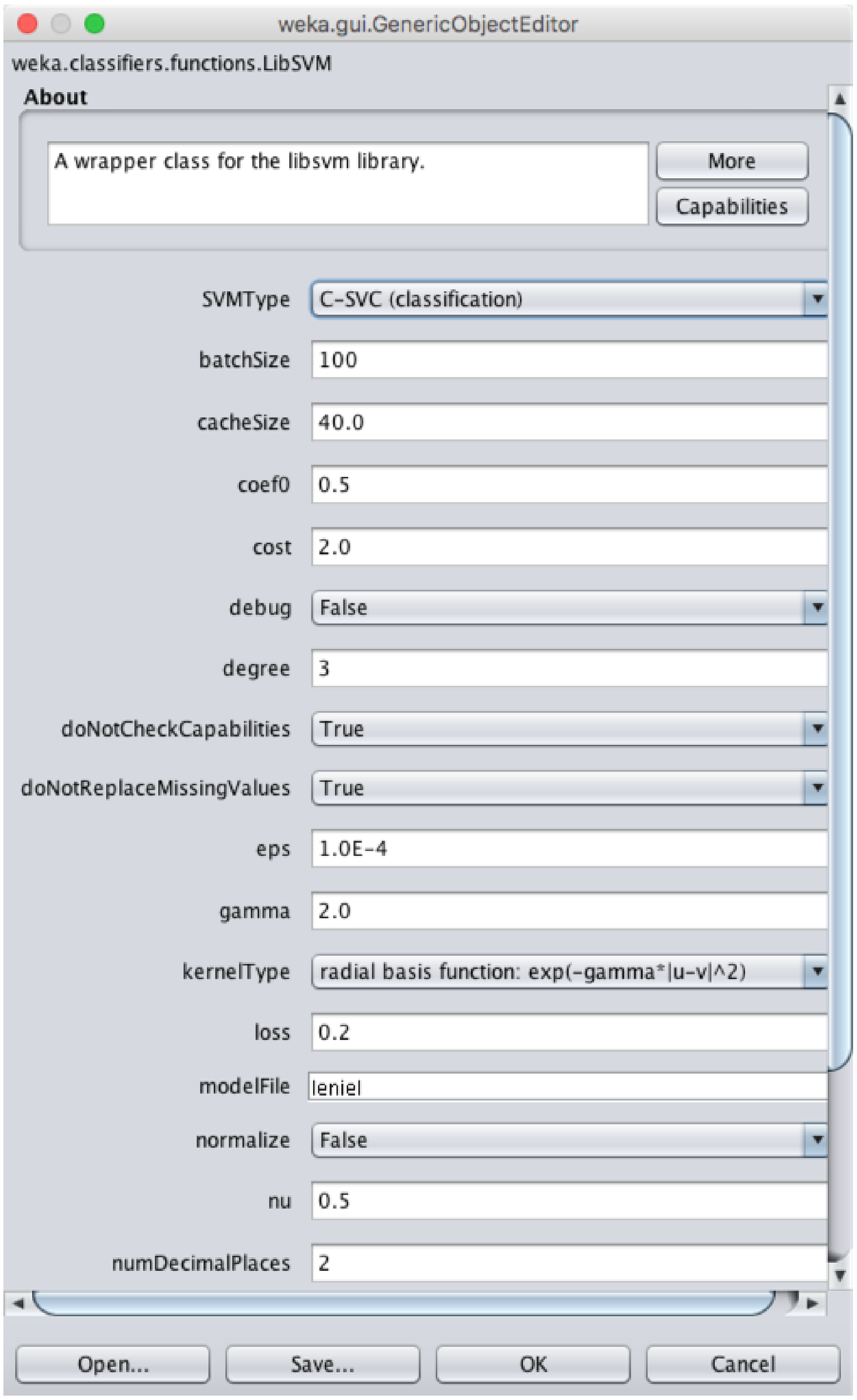
From the output we can see that IBk and Random Forest are the best performing ones when using the Training set alone.

SVM classified more N class samples correctly (2681) while IBk (KNN) was the worst classifying N samples (1661).

Using Conditional Formatting with Color Scales in Microsoft Excel it's easier to spot that the classifiers trained had best performance overall while classifying N class samples in dark green since they represent the group with more instances in the dataset (30.89% or almost 1/3 of all samples).

4.3 - Predictions

As the best performing classifier was the SVM one, I tweaked the default parameters and got a better result reaching 49% of correctly classified instances.


weka.gui.GenericObjectEditor
weka.classifiers.functions.LibSVM
About
More
Capabilities
Open...
Save...
OK
Cancel

A wrapper class for the libsvm library.

SVMType	C-SVC (classification)
batchSize	100
cacheSize	40.0
coef0	0.5
cost	2.0
debug	False
degree	3
doNotCheckCapabilities	True
doNotReplaceMissingValues	True
eps	1.0E-4
gamma	2.0
kernelType	radial basis function: exp(-gamma*|u-v|^2)
loss	0.2
modelFile	leniel
normalize	False
nu	0.5
numDecimalPlaces	2

Figure 7 - Tweaked SVM parameters to improve its efficacy

Given the amount of instances\data to be classified (2175), the predicted classes were stored in the file predictions.csv.

5 - Conclusion

As shown in this work, amongst the 8 classifiers tested on the given dataset, the best performing one was SVM.

Having Naive Bayes as the baseline and using 5 fold validation, SVM outperformed it by a margin of nearly 2% doing 180 more correctly classified instances.

In comparison with IBk (KNN) the worst performing classifier, SVM outperformed it by nearly 7% doing 615 more correctly classified instances.

Classifier	Correctly classified instances	Score\accuracy
SVM	4133	47.96%
Bagging	4041	46.89%
Random Forest	4038	46.86%
PART	4023	46.68%
Bayes Net	3953	45.87%
Naïve Bayes	3953	45.87%
J48	3785	43.92%
IBk (KNN)	3518	40.82%

Table 3 - Classifiers' score for 5-fold cross-validation

Classifiers accuracy vary according to the dataset at hand and that's why trying different classifiers on the same set of data is important.

The numbers are clear: the more the % of correctly classified instances the better is the probability of the classifier being a great fit for the data at hand.

For more serious work, fine tuning the classifiers is a must-do, that is, picking different sets of attributes, normalizing the data, changing the classifier's default parameters, etc can for sure have an influence in the result obtained. As a proof of that, the little change in some of the SVM parameters as shown in 4.3 allowed me to improve its score to 49%. That represents nearly 1% or 225 more correctly classified instances.

6 - References

Bagging
http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html

SVM
http://scikit-learn.org/stable/modules/svm.html

Random Forest
http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

PART
http://weka.sourceforge.net/doc.dev/weka/classifiers/rules/PART.html

Bayes Net
http://weka.sourceforge.net/doc.dev/weka/classifiers/bayes/BayesNet.html

Naive Bayes
http://scikit-learn.org/stable/modules/naive_bayes.html

J48
http://weka.sourceforge.net/doc.dev/weka/classifiers/trees/J48.html

IBk
http://www.cs.tufts.edu/~ablumer/weka/doc/weka.classifiers.IBk.html

Model evaluation
http://scikit-learn.org/stable/modules/model_evaluation.html

Cross-validation: evaluating estimator performance:
http://scikit-learn.org/stable/modules/cross_validation.html#cross-validation

Making predictions in Weka
http://weka.wikispaces.com/Making+predictions

Confusion Matrix
https://en.wikipedia.org/wiki/Confusion_matrix

Classifier comparison
http://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html