Machine Learning
Classification exercise report - Group 27

Borna Feldsar
Martin Matak
Soeren Nickel

November 24, 2017

1. Datasets

1.1 Dataset *KDD Cup 1998*

Source: https://www.kaggle.com/c/184702-tu-ml-ws-17-kdd-cup-1998

1.1.1 Characteristics of the dataset

number of samples	number of dimensions	number of classes	preprocessing needed
??	??	2	??

1.1.2 Preprocessing

...

1.2 Dataset *Breast cancer*

Source: https://www.kaggle.com/c/184702-tu-ml-ws-17-breast-cancer

1.2.1 Characteristics of the dataset

number of samples	number of dimensions	number of classes	preprocessing needed
??	??	2	??

1.2.2 Preprocessing

...

1.3 Dataset *Car evaluation*

Source: https://archive.ics.uci.edu/ml/datasets/car+evaluation

1.3.1 Characteristics of the dataset

number of samples	number of dimensions	number of classes	preprocessing needed
1728	6	4	yes

1.3.2 Preprocessing

...

1.4 Dataset *Detect Malicious Executable(AntiVirus)*

Source: https://archive.ics.uci.edu/ml/datasets/Detect+Malacious+Executable(AntiVirus)

1.4.1 Characteristics of the dataset

number of samples	number of dimensions	number of classes	preprocessing needed
373	513	2	yes

1.4.2 Preprocessing

...

2. Classifiers

We picked four different classifiers: Naive Bayes, SVM, Decision Tree and Random Forest. Below is explained our motivation why we decided to take those four classifiers.

2.1 Naive Bayes

Classifier which comes from family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features. Although it's mostly used for working with text classification, we picked it as one of our four classifiers. We believe that the conditional independence assumption actually holds and that's why we picked this classifier. Moreover, we assume that it will behave well on smaller data sets. Finally, we think that training time will be shorter than for some more complex classifiers - e.g. SVM.

2.2 Support Vector Machine - SVM

SVM is non-probabilistic binary linear classifier which can also perform a non-linear classification using the *kernel trick*, implicitly mapping their inputs into high-dimensional feature spaces. As one of the most popular and oldest [1] classifiers which is in this area, we had to pick it to try it out and learn more about it. We expect it will behave very well on data sets with only two classes. However, we are aware that we will have to come up with a solution for classifying categorical data with this classifier. Moreover, when used to classify in more then two classes, we expect longer runtime. Finally, we expect SVM to be good at dealing with small data sets, since only support vectors and not all the samples are used to constuct the separating hyperplane.

2.3 Decision Tree

Since we have two classifiers which we assume work well with small data sets, we had to pick one which would work good with large data sets in a reasonable time. That's why we picked this classifier. In addition to that, it is able to handle both numerical and categorical data so we will need less preprocessing when using this classifier. However, we must

[1]The original SVM algorithm was invented by Vladimir N. Vapnik and Alexey Ya. Chervonenkis in 1963.

Data class	Classified as *pos*	Classified as *neg*
pos	true positive (*tp*)	false negative (*fn*)
neg	false positive (*fp*)	true negative (*tn*)

Table 1: Confusion matrix

take care about the depth of the tree because it can easily happen that we finish up with over-complex tree that do not generalize well from the training data. That's why we don't expect it to work well with small data sets.

2.4 Random Forest

In order to solve the issue of overfitting of *Decision Tree* classifier, we decided to take an ensamble approach. Therefore, we expect to see the difference (at least on smaller data sets) in between *Random Forest* (RF) and *Decision Tree* (DT) in favour of *Random Forest* (RF) which is statistically significant. However, we are aware that RF models are black box and DT models are white box, but that difference for us is not important in this exercise. Finally, since RF is an ensamble approach, we assume that evaluation time will be greater than using DT as classifier.

2.5 Performance measures

Based on slides of this course on TUWEL and this paper, we decided to measure performance of our classifiers as follows.

In case of **binary classification** we provide *confusion matrix* described in Table 1. Using it, we compute then *accuracy* of our classifier as $\frac{tp+tn}{tp+tn+fp+fn}$ and *AUC* as $\frac{1}{2}(\frac{tp}{tp+fn}+\frac{tn}{tn+fp})$. With those measures, we will measure the overall effectiveness (avg) of the classifier and classifier's ability to avoid false classification (AUC).

Regarding the **multi-class classification**, since the greatest number of classes we have is 4, we provide confusion matrix for every of those classes. Moreover, we provide *average accuracy* of a classifier. Average accuracy is defined as average *accuracy* over all of the classes, i.e. $\frac{\sum_{i=1}^{l}\frac{tp_i+tn_i}{tp_i+fp_i+tn_i+fn_i}}{l}$ where l is number of different classes. Additionally, we provide *Error rate* $\frac{\sum_{i=1}^{l}\frac{fp_i+fn_i}{tp_i+fp_i+tn_i+fn_i}}{l}$ as well. Hence, we will see the average per-class effectiveness of a classifier and the average per-class classification error.

Since confusion matrix is provided, it should not be a problem for a reader to come up with some other more complex performance measure - e.g. *FScore* or any other performance measure based on confusion matrix if needed. Therefore, we believe that our performance measures and confusion matrix are sufficient.

3. Dataset *KDD Cup 1998*

Data class	Classified as *pos*	Classified as *neg*
pos	true positive (*tp*)	false negative (*fn*)
neg	false positive (*fp*)	true negative (*tn*)

Table 2: Confusion matrix of best performance

-	Accuracy	ACU	Training time [min]	Evaluation time [min]
param1, param2	1	1	1	1
param1, param2	1	1	1	1
param1, param2	2	1	1	1
param1, param2	1	1	1	1

Table 3: Performance measures with different parameters

3.1 Naive Bayes

Confusion matrix with best performance is in the Table 2. Performance measures with different parameters are in the Table 3.

3.2 Support Vector Machine - SVM

Confusion matrix with best performance is in the Table 4. Performance measures with different parameters are in the Table 5.

3.3 Decision Tree

Confusion matrix with best performance is in the Table 6. Performance measures with different parameters are in the Table 7.

3.4 Random Forest

Confusion matrix with best performance is in the Table 8. Performance measures with different parameters are in the Table 9.

3.5 Comparison of classifiers

4. Dataset *Breast cancer*

Data class	Classified as *pos*	Classified as *neg*
pos	true positive (*tp*)	false negative (*fn*)
neg	false positive (*fp*)	true negative (*tn*)

Table 4: Confusion matrix of best performance

-	Accuracy	ACU	Training time [min]	Evaluation time [min]
param1, param2	1	1	1	1
param1, param2	1	1	1	1
param1, param2	2	1	1	1
param1, param2	1	1	1	1

Table 5: Performance measures with different parameters

Data class	Classified as *pos*	Classified as *neg*
pos	true positive (*tp*)	false negative (*fn*)
neg	false positive (*fp*)	true negative (*tn*)

Table 6: Confusion matrix of best performance

-	Accuracy	ACU	Training time [min]	Evaluation time [min]
param1, param2	1	1	1	1
param1, param2	1	1	1	1
param1, param2	2	1	1	1
param1, param2	1	1	1	1

Table 7: Performance measures with different parameters

Data class	Classified as *pos*	Classified as *neg*
pos	true positive (*tp*)	false negative (*fn*)
neg	false positive (*fp*)	true negative (*tn*)

Table 8: Confusion matrix of best performance

-	Accuracy	ACU	Training time [min]	Evaluation time [min]
param1, param2	1	1	1	1
param1, param2	1	1	1	1
param1, param2	2	1	1	1
param1, param2	1	1	1	1

Table 9: Performance measures with different parameters