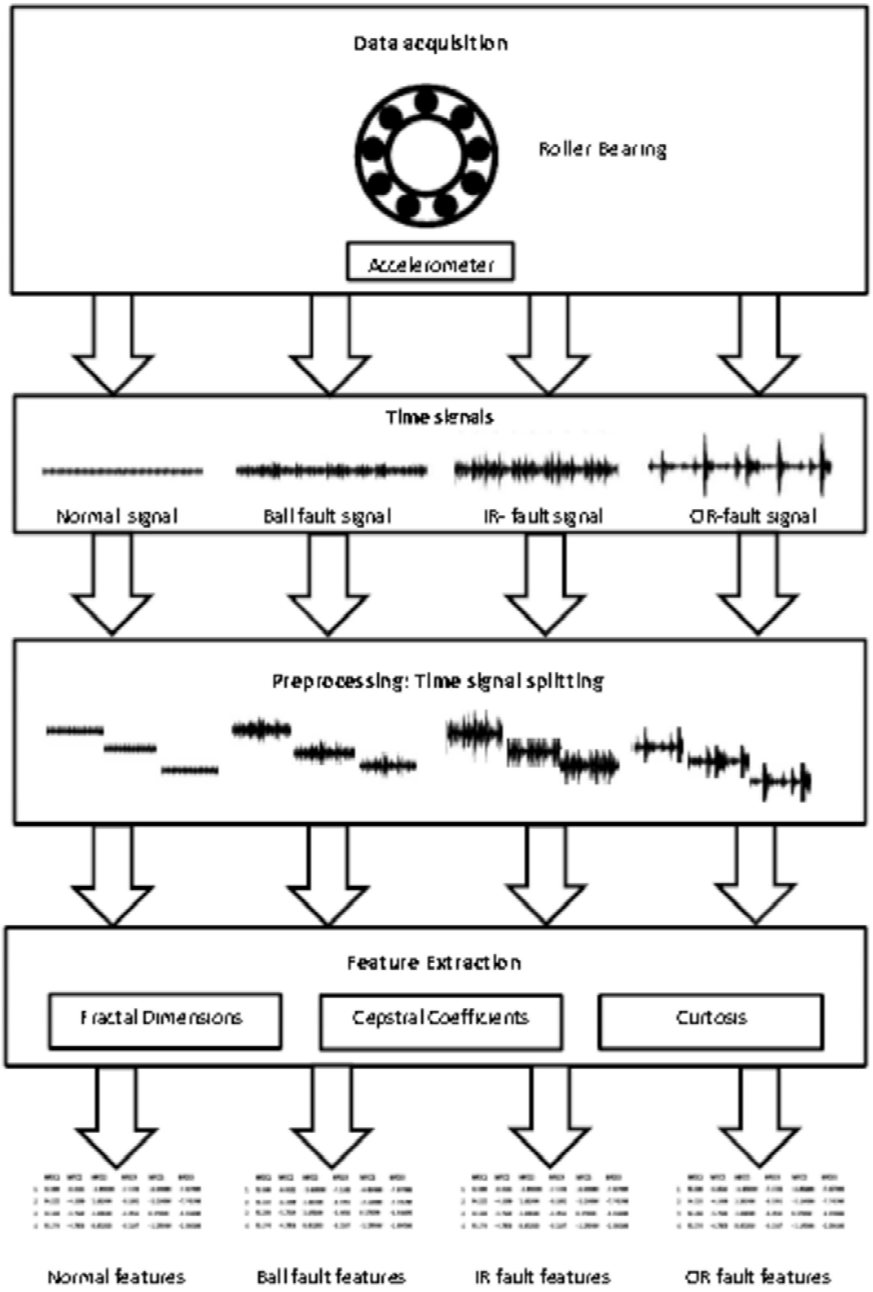
1 Experiments

Figure 1 provides an overview of the steps involved in the classification and evaluation approach of this thesis.



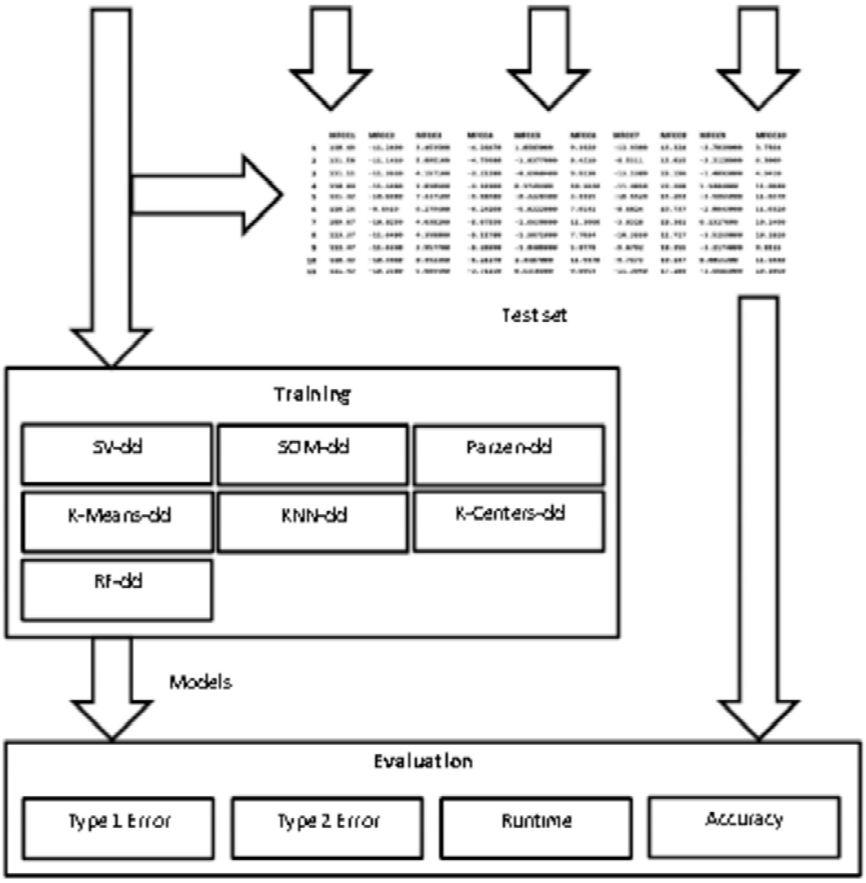


Fig. 1 Experimental Sequence

Data Acquisition was not part of this thesis, the roller bearing data sets available at (Case Western University) were used as benchmark. A brief overview of these datasets is given in 5.1. The preprocessing and feature extraction steps are summarized in 5.2 and chapter 5.3 describes the classification and evaluation sequences.

All experiments were implemented as MATLAB scripts. Except for the *Random Forest* classification, the *one-class classifiers* were realized with *mappings* from the *Dd_tools* toolbox provided by the TU Delft (Tax D. M., 2012). The *Random Forest* approach outlined in 5.3 was first evaluated based on an R-package (Liaw & Wiener, 2002) and then integrated into the *Dd_tools* framework.

1.1 Roller bearing datasets

Rotating Machines are very common in various industrial applications. In manufacturing, most machine failures are linked to bearing faults (Lou, Loparo, Discenzo, Yoo, & Twarowski, 2004). Consequently, much effort has gone into the development of classification and prediction techniques for bearing faults (Marwala, 2012), (Nelvamondo, Marwala, & Mahola, 2006), (Li, Chow, Tipsuwan, & Hung, 2000).

Roller bearings consist of two concentric rings, the inner and outer raceway (*IR* and *OR*), with a set of rolling elements running between their tracks, as illustrated in Figure 2.

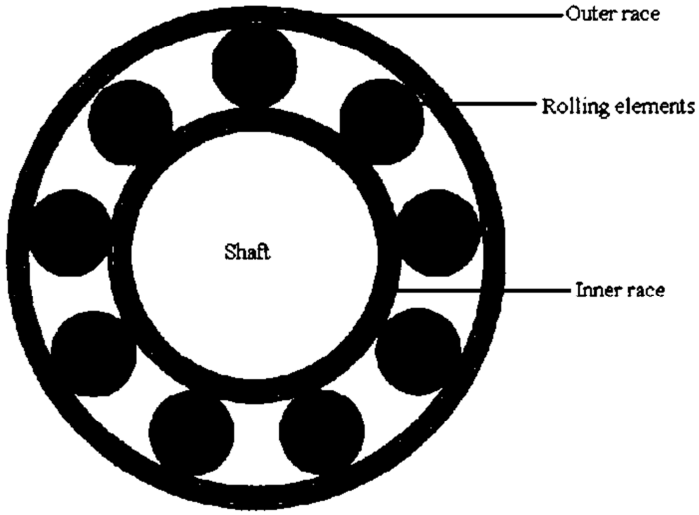


Fig. 2 Roller Bearing

The rolling elements are usually contained in a cage to prevent contact between elements.

Roller bearings generate vibration signals with characteristic shapes, depending on the conditions of the raceways and the rolling elements. Faults in one of these parts typically manifest themselves in characteristic frequencies in the vibration signal (Li, Chow, Tipsuwan, & Hung, 2000).

The *roller bearing* data collection provided by (Case Western University), includes vibration signals of four *roller bearing* conditions:

- Normal conditions
- Ball fault conditions
- Inner raceway fault conditions
- Outer raceway fault conditions

The individual datasets in the collection are defined by the parameters

- Position of the accelerometer for data acquisition (fan end or drive end)
- Rotation speed (1797rpm,1772rpm,1750rpm,1730rpm)
- Motor Load, correlated to the Rotation speed (0HP,1HP,2HP,3HP)
- Sample rate (12k and 48k)

The four data sets used for experiments in this thesis were acquired from the drive end, at a speed of 1797 and with a sample rate of 48k. Figure 3 shows sequences of each data set corresponding to the first five revolutions of the roller bearing.

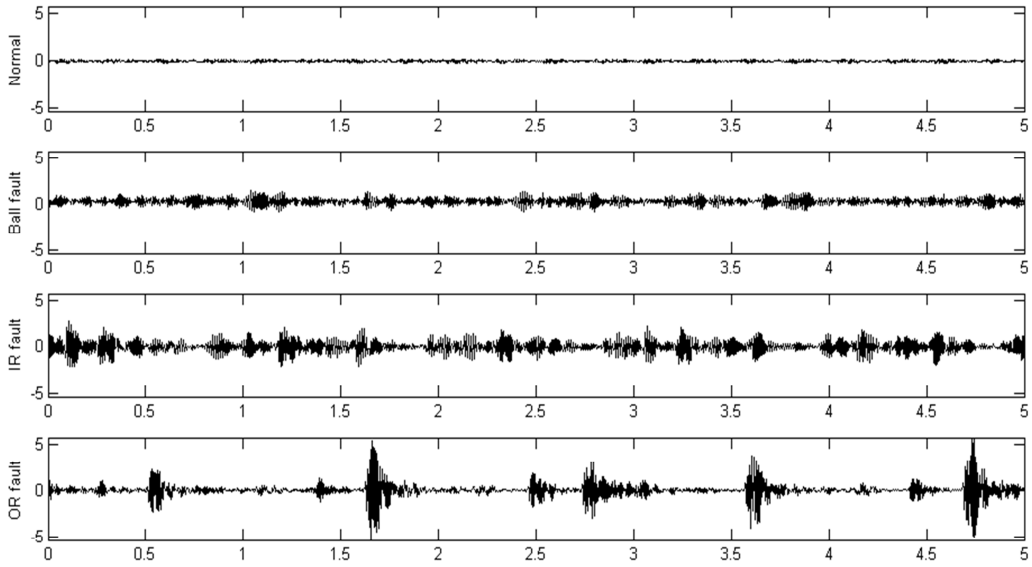


Fig. 3 Roller Bearing Sample Data

To simulate a *semi-supervised* scenario, only the normal condition data was used in classifier training and the three fault data sets were exclusively used for evaluation purposes.

1.2 Feature extraction

In a preprocessing step, the large data sets representing the normal and the three fault conditions were split into equal segments, with a length corresponding to five roller bearing revolutions. The *kurtosis* was calculated directly from each segment. For the extraction of MFD and MFCC, each segment was further split into 15 frames of equal length. Of each of these frames, 13 *MFCCs* and 13 *HFDs* were extracted. Table 1 illustrates the schema of a feature data set.

	$MFCC_1$	…	$MFCC_{13}$	MFD_1	…	MFD_{13}	*kurtosis*
Feature Vector 1	###	###	###	###	###	###	###
…	###	###	###	###	###	###	###
Feature Vector N	###	###	###	###	###	###	###

Table 1 Feature data set schema

1.2.1 Mel Frequency Cepstral Coefficients

The MATLAB package used for extracting the *MFCCs* (Ellis, 2005) is part of a collection designed for feature extraction in the speech recognition domain. It is highly adaptable through a large number of parameters, of which many require specific domain knowledge. For the purposes of this thesis, default settings proved to be sufficient for most of these parameters. The number of *Short Time Foruier Transform* frames was set to 15 and the number of *MFCCs* extracted from each of these frames was set to 13.

Figure 4 shows 13 *MFCCs* from the same frame of each dataset.

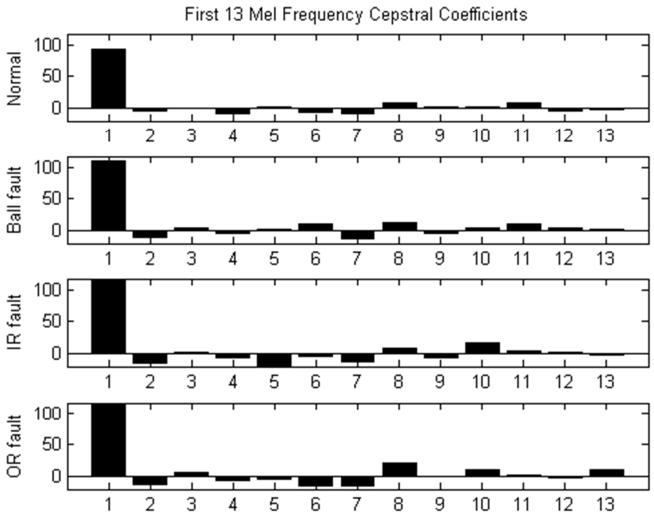


Fig. 4 Thirteen Mel Frequency Cepstral Coefficients

1.2.2 Higuchi Fractal Dimensions

The *Higuchi Fractal Dimension* method was implemented according to the definition in (Polychronaki, et al., 2010) . A Weierstrass function with known *fractal dimension* was used to verify correct behavior of the *HFD*-function.

Since no exact rules exist for the selection of the free parameter k_{max} , the parameter was varied within a certain range, based on experimental results (Polychronaki, et al., 2010). In order to get the same number of *HFDs* as *MFCCs*, each segment was split into 15 frames and for each frame 13 different *HFDs* where calculated by selecting a different value for k_{max} each time. With this approach, the best k_{max} values could be selected by *PCA* or similar *feature selection* methods in a subsequent step.

The graph in figure 5 illustrates the *HFD* values corresponding to $k_{max} = 6$ of the first 15 feature vectors of each set.

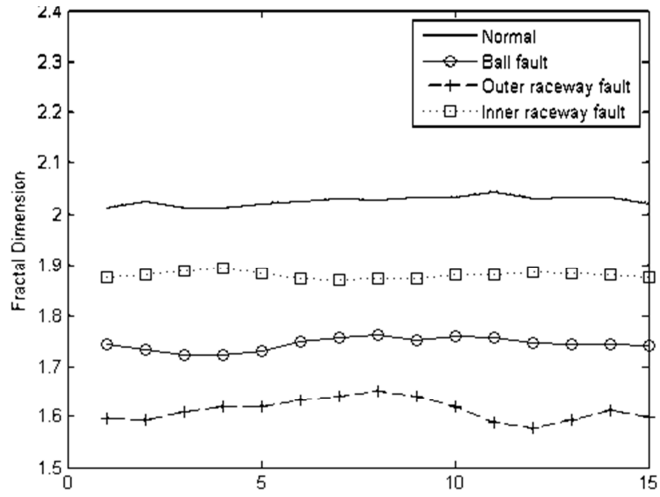


Fig. 5 15 Higuchi Fractal Dimensions with k=6

The *fractal dimensions* of the normal signal obviously exceed the expected fractal dimensions of a curve, which should range between one and two. The reason for this may be related to the specific shape of the normal signal, which looks like random noise and shows no discernible patterns. However, what really matters in the context of feature extraction are not absolute feature values, but values which are well suited to distinguish conditions. Figure 13 suggests that the *HFD* features are a good choice in this regard.

1.3 Random forest data description

The *outlier measure* of *Random Forest* as introduced in 4.6 is only defined for the training data set and cannot be applied to unseen test data. It can however be used to identify critical or implausible data in the training set. Such data can then be removed or modified before a retraining run.

Fig. 6 illustrates the *outlier measures* for a *random forest*, trained with 427 normal feature vectors.

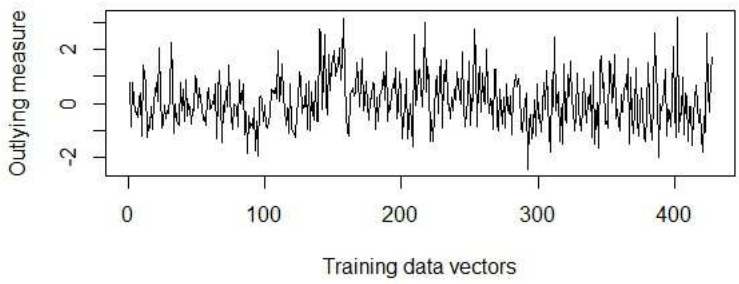


Fig. 6 Outlier Measures of Roller Bearing normal training data

According to a rule of thumb given in (Breiman & Cutler, Random Forests), data points with an *outlier measure* beyond a threshold of about $|10|$ require closer inspection. Figure 14 shows clearly, that the *outlier measures* for all training data points are smaller than $|10|$. Consequently, no further processing of the training data set was required.

Another useful feature which provides some insight into certain aspects of a training set, is the *variable importance* measure introduced in 4.2. Figure 7 illustrates the *variable importance* measures calculated during a *random forest* training with 427 normal feature vectors.

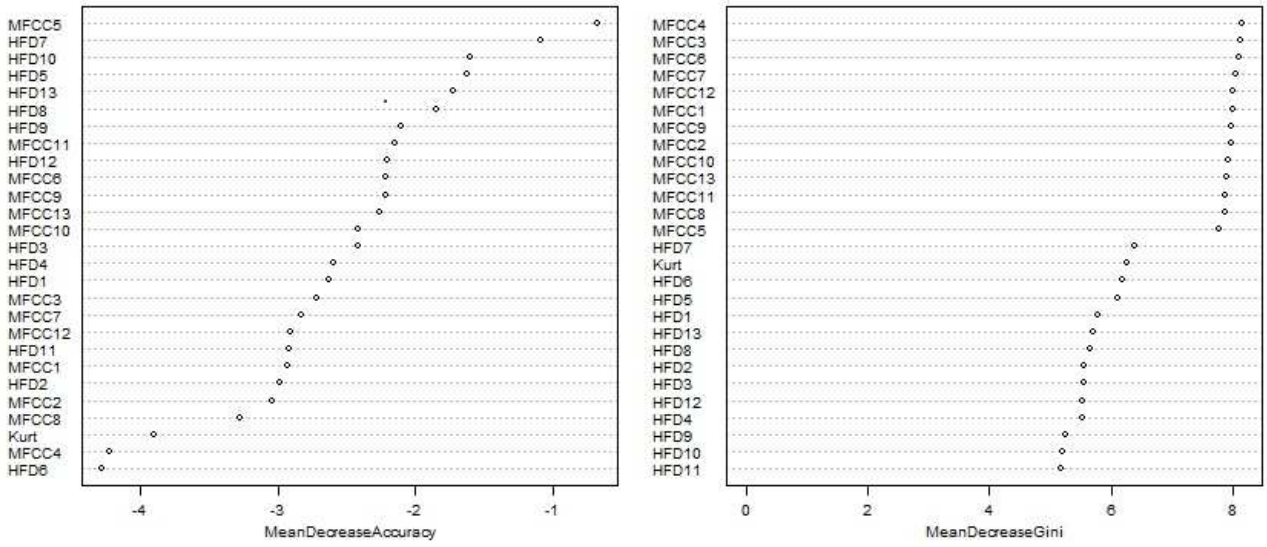


Fig. 7 Attribute Importance of Roller Bearing normal features training set

The *variable importance* can be defined as mean decrease in *accuracy* or as mean *gini* decrease, with both measures resulting in a different importance order of the attributes. An application of the *importance* measures as feature reduction method was evaluated in (1.4.2).

For the application of *random forests* in *semi-supervised outlier detection*, a *random forest one-class classifier* was implemented according to the generic approach introduced in 3. The classification sequence involves the following steps:

1. Training of a *random forest* in unsupervised mode (4.5) using only normal data
2. Calculation of a normal prototype $\boldsymbol{p_{normal}}$ (4.4)
3. Computation of the Euclidean distances $d(\boldsymbol{x}_n, \boldsymbol{p_{normal}})$ between test objects $\boldsymbol{x}_n$ and the normal prototype $\boldsymbol{p_{normal}}$
4. Calculation of a distance based threshold according to definition (3.3)
5. Classification of the test objects as defined by (3.1)

Figure 8 illustrates the Euclidean distances between the normal prototype and a test set with 120 samples, calculated during a test run.

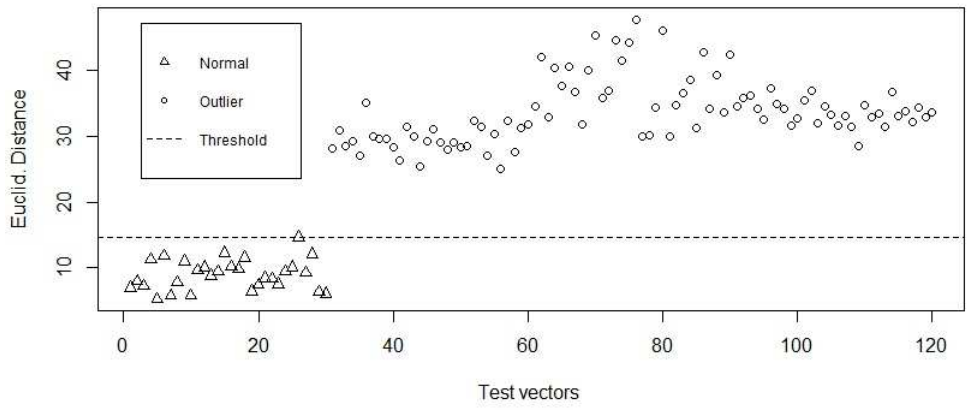


Fig. 8 Euclidean Distances between test data and normal prototype

The plot of Euclidean distances shows, that a constant classification threshold simply defined as the highest Euclidean distance among all distances between normal samples and the normal prototype (i.e. $F_{T-} = 0$) separates normal and outlier data perfectly.

1.4 Classification and evaluation

All experiments were conducted on an Intel(R) Core(TM)2 Duo CPU P7450 machine with a clock speed of 2.13GHz and 8129 MB of RAM.

Two experiments were defined to evaluate the outlier detection approach introduced in this thesis. In a first experiment, the classifiers were trained and tested repeatedly to estimate several performance measures. Goal of a second experiment was to evaluate the applicability of the *random forest variable importance* for feature reduction. The two experiments and the results are described below.

1.4.1 Performance experiment

One experimental sequence comprised the following steps:

- Construction of a training set with 368 objects, sampled without replacement from the normal features
- Construction of a balanced test set with 180 objects, by combining the normal objects not used for training with 30 objects sampled from each of the fault feature sets
- Training of each classifier with the training set
- Testing of each classifier using the test set
- Taking the training and testing runtime as well as the error type I (fraction of rejected normals) and error type II (fraction of accepted outliers) for each classifier

The complete sequence was run ten times.

To compare classification results, several performance measures were calculated for each classifier, averaged over the ten runs:

- training and testing time
- type I and type II errors
- $Accuracy = \frac{\#True\ Outliers + \#True\ Targets}{\#True\ Outliers + \#True\ Targets + \#False\ Outliers + \#False\ Targets}$

Table 4 contains a summary of the results of one complete experimental run, where F_{T-} was set to 0.1 for all classifiers.

	Parameter Settings	Average Training Runtime	Average Testing Runtime	Fraction of Rejected Normals (e 1)	Fraction of Accepted Outliers (e 2)	Accuracy
RF-dd	$\#trees = 100$ $\#nodes = 737$ mtry=5	0.9432 sec	0.0057 sec	0.1044	0	0.9478
SV-dd	Gaussian kernel $sigma = 6$	20,2460 sec	0.0093 sec	0.5300	0	0.735
K-means	$errtol = 10^{-5}$ $k = 5$	0,019 sec	0.005 sec	0.1422	0	0.9289

K-center	#tries=25
	K=5
		0,4289 sec	0.446 sec	0.1100	0	0.945

N-dd	-	0,4585 sec	0.013 sec	0.1278	0	0.9361
Parzen	$h = 1.5127$	0,2277 sec	0.0259 sec	1	0	0.5
Som	$2D, \#neurons = 675$ $h = [0.6,0.2,0.01]$ $\eta = [0.5,0.3,0.1]$	162,045 sec	0.0058 sec	0.1733	0	0.9133

Table 2 Experimental results

Figures 9 and 10 compare the time performance and accuracies of the classifiers.

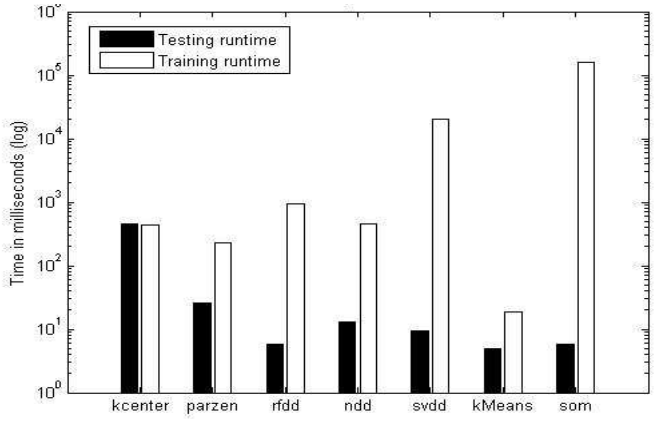


Fig. 9 Time Performance

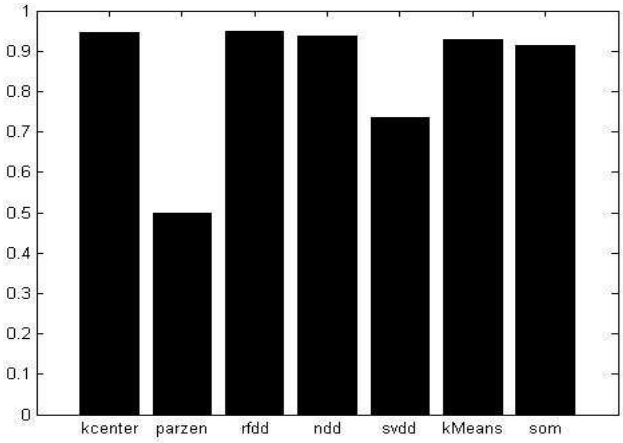


Fig. 10 Accuracies

A log-scale is used for the illustration of time performance values, since the training and testing times of the individual classifiers varied in a wide range. The *SOM-dd* had by far the longest average training time, followed by *SVDD*. Fastest in terms of average training time was *K-Means-dd*.

In a typical real world application, where classifiers would be trained offline and only once, the time needed to classify a new object is probably a more significant measure. Best in this category was

again the *K-Means-dd*, *K-center*, whose training and testing times were approximately equal, had by far the longest testing time.

Five of the seven classifiers achieved an accuracy score of more than 0.9, *rf-dd* had the best results with an accuracy of 0.9478. The two worst classifiers in terms of *accuracy* were *SVDD* and *Parzen-dd*. *Parzen-dd* rejected all objects of the balanced test set and thus achieved an accuracy of only 0.5. A feature scaling prior to training of the *Parzen-dd* did not improve performance significantly.

1.4.2 Feature reduction experiment

The feature reduction experiment involved the following steps:

- Construction of a training and a test set as in the performance experiment
- Training of a random forest classifier, calculation of variable importance
- In a loop starting with only the most important feature, adding the next most important feature in each iteration until the feature set is complete:
    - Training of each classifier (except random forest) using the reduced training set
    - Testing of each classifier using the reduced test set
    - Calculation of error type 1 and error type 2 for each classifier

This complete sequence was repeated several times. The resulting averaged accuracies for each reduced feature set are illustrated in figure 11.

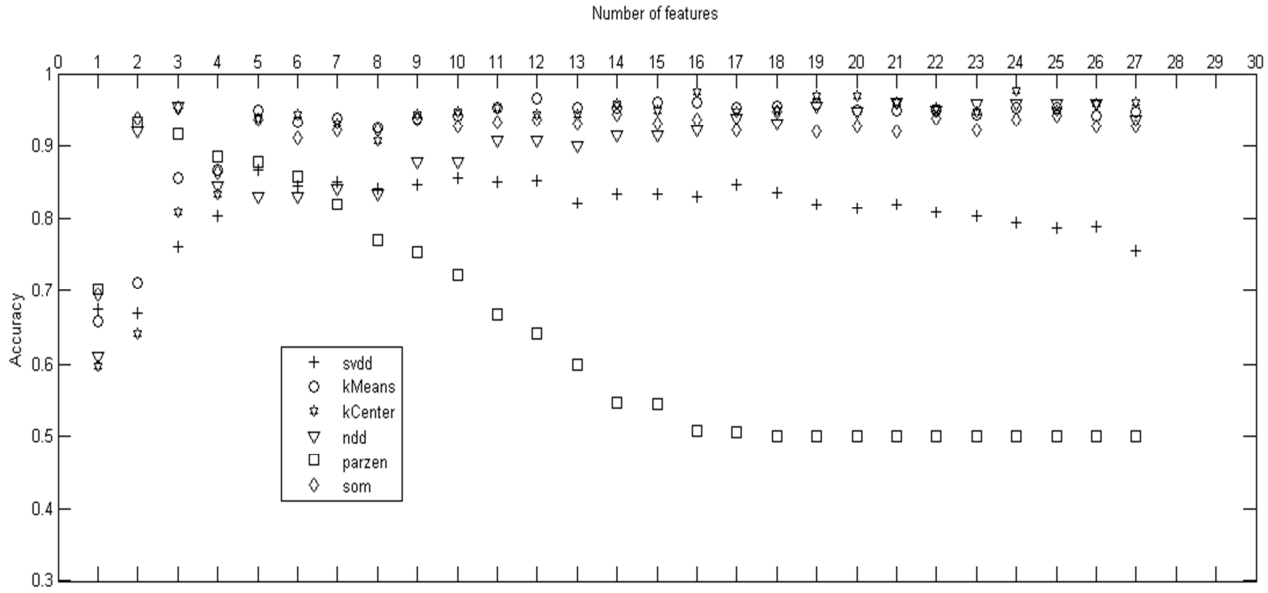


Fig. 11 Accuracies with reduced features

Figure 11 shows, that for most classifiers the number of features could be reduced considerably, without sacrificing accuracy. This could be an indication for redundancies in the feature set, which in turn would be the result of correlations between individual features.

The outcome of this experiment proofs, that the *random forest variable importance* can be used as an adequate feature reduction method.

2 Conclusions