DE GRUYTER
OPEN

JACSM 2017, Vol. 9, No. 1, pp. 5 - 22
10.1515/jacsm-2017-0001

ANOMALY DETECTION IN SERVER METRICS WITH USE OF ONE-SIDED MEDIAN ALGORITHM

Szymon Zacher [1], Przemysław Ryba[2]

[1] OVH SAS, Wrocław, Poland
szymon.zacher@corp.ovh.com

[2] Department of Systems and Computer Networks
Wrocław University of Science and Technology, Wrocław, Poland
przemyslaw.ryba@pwr.edu,pl

Abstract

In this paper we consider the problem of anomaly detection over time series metrics data took from one of corporate grade mail service cluster. We propose the algorithm based on one-sided median concept and present some results of experiments showing impact of parameters settings on algorithm performance. In addition we present short description of classes of anomalies discovered in monitored system. Proposed one-sided median based algorithm shows great robustness and good detection rate and can be considered as possible simple production ready solution.

Key words: anomaly detection, time series, one-sided median, server metrics

1 Introduction

Each system has to be monitored in order to gain certain level of stability and robustness. In the world of fast data exchange, behavior of complicated system can change rapidly during very small amount of time. Therefore not only monitoring of system has to be performed, but also constant analysis of collected data in order to detect each signs of instability or undefined behavior. In traditional approach these activities are performed by specialized and trained administrator who keeps track on incoming data and decide whether any changes of system behavior are anomalous or not. Hopefully in modern world of machine learning and data processing we do not have to rely completely on human work, we can craft a mechanism which can learn itself a normal state of system and detect any deviations from such model.

The most demanding part of such mechanism is definitely anomaly detection algorithm which has to be very sensitive for any deviation, but also re-

sistant for random noise coexisting in every data stream. Such algorithm should report anomaly with small rate of false positive alerts, because each alert triggers some human action or at least creates a urge for specialist on duty to check state of the infrastructure manually. Very important is also to choose algorithms with small memory and computation power requirements in order to create a possibility for simultaneous analysis of huge amount of time series data.

2 Available algorithms

Large variety of algorithms for performing anomaly detection over streaming time series data were presented in the literature. These algorithms can be assigned in one of 4 groups: cluster based methods, prediction models, density based methods and profile description methods.

Cluster based methods such as k-means [1] or k-medoids [2] works by discovering in historical data some clusters of probes and verifying whether current value of metric can be fitted in one of discovered set. For creation of cluster some parameters derived from probe has to be used. Example parameter can be day of week or hour of measurement. Similar approach is presented in works based on Support Vector Machine (SVM) [3] where a hyperplane separating different clusters of probes is created. There was also successful attempts to use Expectation-Maximization algorithm for profiling behavior of Hadoop clusters [4].

Profile description methods relies on learning by algorithm normal shape of analyzed time series and comparing collected data points with learned shape. Good example is Tiresias algorithm [5] used for monitoring performance of operating systems. Other solution can use specialized neural network for maintaining shape of checked series [6].

Another approach is to take into consideration density or distance between samples. In such methods all samples with number of neighbors below some threshold are taken as an anomalies. To use such method in streaming data, time sliding window for constant removal of old outdated samples, need to be introduced. The main work related to this subject makes use of ISB structure for storing neighborhood relation [7] or Yang algorithms which utilize some properties of sliding window [8].

Prediction approach is based on creating a model which will predict future value of data point and compare predicted value with real one. This comparison can be used to decide whether data point should be considered as anomaly or not. There are multiple different ways for creating a predicting model. Some examples of elementary ones are ARIMA model [9] and linear neural network. There are also some works comparing use of multilayer perceptron with linear neural network, naïve predictor and nearest cluster [10]. Other

solution can be Gibbs probes [11] or weighted maximal likelihood estimation [12].

3 Data streams source

All metrics series used in this work comes from one cluster of 70 mail servers processing around 26 million messages per day. In this paper as server metrics we will understand every time series containing some system specific variables such as local mail queue, number of concurrent connections or other simple system measurements like load average or size of free system memory. All data was collected by collectd system monitor and was stored in whisper datafiles. Samples was collected during five months period with granularity of one point per minute.

4 Data streams classification

Based on characteristic of time series data gathered in monitored system we distinguished following classes of time series:

4.1 Type I series – uniform series with rapid changes

In all Type I series seasonal effect does not exist or is negligibly small, additionally all existed anomaly have characteristic of rapid change with huge amplitude. Those kind of series describe systems witch workload is time independent or metrics whose should have always have the same value in correctly working system. As example we can use time series describing number of deferred e-mails on server inside mail infrastructure. Due to mail system specification such queue should be fixed at zero and each difference from this value can be considered as anomaly.

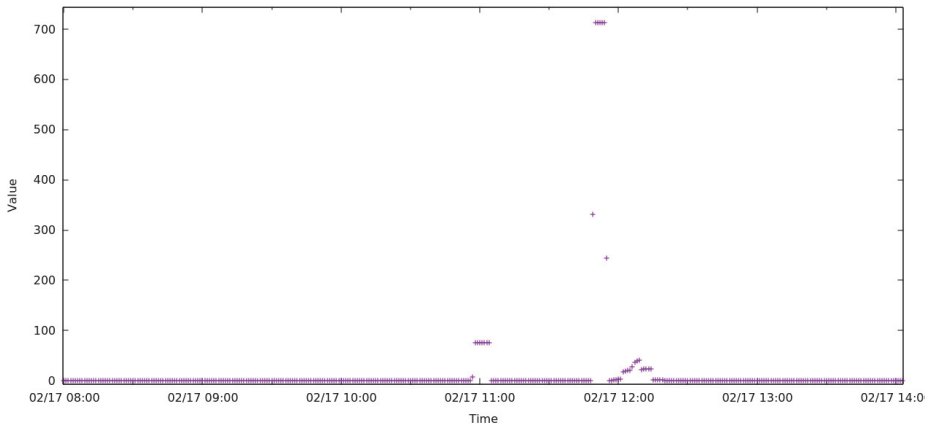


Figure 1. Type I series with stable fixed value

Other example of slightly more noisy series can be metric of active processed messages. Seasonality of such data is definitely small, therefore it can be considered as constant series with some random noise.

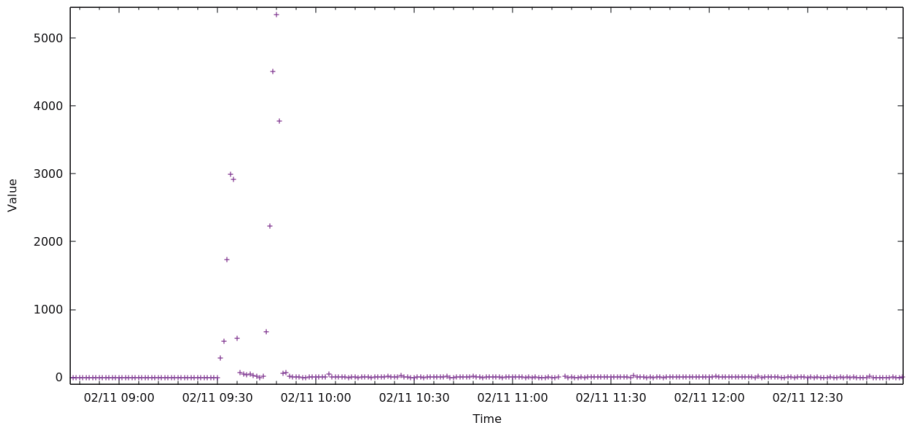


Figure 2. Type I series with very small seasonal effect

Those type of time series are characterized by high local consistency and complete time independency. Because of that there is no need to consider wider context of data. Each algorithm can just work on most recent examples and detect only local outliers.

4.2 Type II series – uniform series with slow changes

Very interesting example of time series is uniform and time independent series with very slow growing abnormal value. Such series can for example describe deferred message queue on output nodes caused generally by failure on the remote service providers side. Traffic to single provider is too small to cause rapid grow of deferred queue, but because of the long term characteristic of such failure, constant grow of queue on output nodes, can be possible.

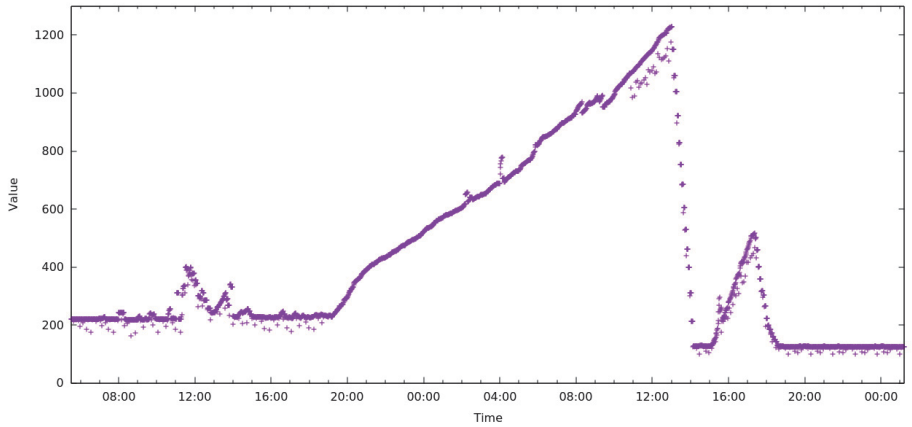


Figure 3. Type II series with slow growing anomaly

In production environment we cannot find any constant normal value for such metrics. This fact makes analysis much harder. Hopefully all series of this type have high level of local consistency.

4.3 Type III series – seasonal

Seasonal series are the most common ones, because they can describe typical standard working system with time dependent workload. Generally series of this kind have different average depending of the time of the day and day of the week. Let the example be number of messages originating from one of the input nodes.

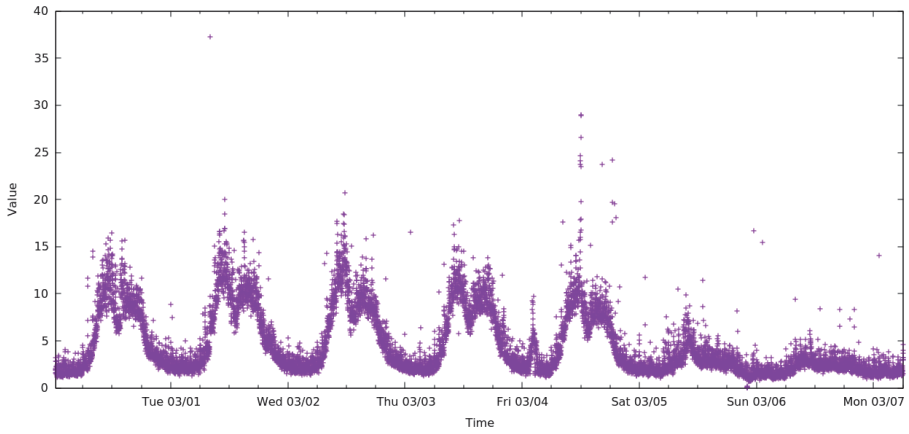


Figure 4. Type III seasonal series

Characteristic daily repeating pattern with two maximums around 11 and 16 o`clock can be seen. In addition to that there is possibility to observe general decrease of samples values during weekend or free days. It's worth to stressed out that the value of anomalous points at midnight 4th of the march is lower than normal workload during rush hours. The best algorithm should take time of measurement into consideration. Existence of local consistency in all series of this type is very promising. This means that all outliers points are anomalies. Thanks to that property there is possibility to use local consistency algorithm for anomaly detection.

4.4 Type IV series – seasonal not uniform series

Type IV of time series is a simple derivation of Type III series without local consistency. Loss of consistency can be caused by some repeating event which resulted in creation of peaks in series data. It is very important to note that such events are not anomalous and there is no need to generate alert for each occurrence. Good example of Type IV series is number of concurrent connections gathered from infrastructure input nodes.

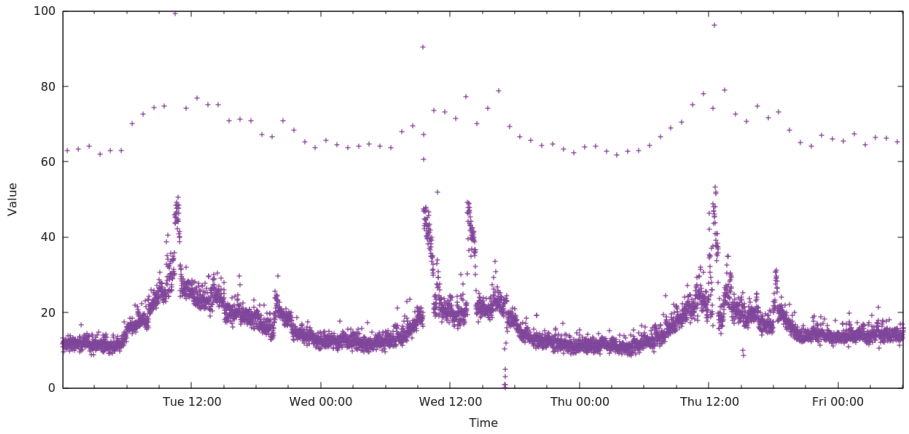


Figure 5. Type IV seasonal series with series of peaks

There is an easily distinguishable seasonal trend known from Type III series. Main difference is periodical rapid increase of value in every 15 minutes block, such points will be for sure marked as outlier but they are not an anomalies. Loss of local uniformity forces algorithm to consider wider context of data points in order to detect places where outlying point should occur. All of that makes this type of metric harder to analysis than Type III.

5 One sided median

The one sided median algorithm is based on quite simple prediction model, main assumption is that the median of series is almost always constant during single chosen time window. Described algorithm is based on work of S. Basu and M. Meckesheimer [13].

Consider time series Y containing ordered data points $y_1, y_2 \dots y_{n-1}, y_n$. For every time t we can define a k-length time window which will contains points

$$W_t^{(k)} = \{y_{t-k}, y_{t-k-1} \dots y_{t-1}\}, \tag{1}$$

We can also define series of differences between following points in time window $W_t^{(k)}$ let's denote

$$z_t = y_t - y_{t-1}, \tag{2}$$

than such series can be written as

$$Z_t^{(k)} = \{z_{t-k}, z_{t-k-1} \dots z_{t-1}\}, \tag{3}$$

Let's denote the median of $W_t^{(k)}$ as $\widetilde{W}$ and the median of $Z_t^{(k)}$ as $\tilde{Z}$. Finally in one sided median prediction value of data point at time t is estimated as

$$\tilde{y}_t^{(k)} = \widetilde{W} + \frac{k}{2}\tilde{Z}, \tag{4}$$

and the error of estimation is calculated using equation

$$\epsilon = \left| y_t^{(k)} - \tilde{y}_t^{(k)} \right|, \tag{5}$$

and compared to some threshold value τ. If the value of error ϵ is greater than threshold algorithm assume that the point is an anomaly and should be marked. Figure 6 shows concept of the current algorithm.

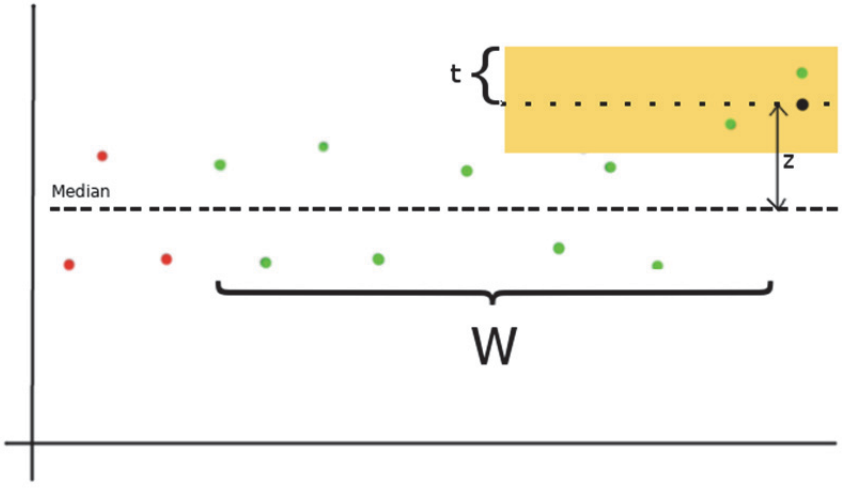


Figure 6. One sided median algorithm

There is also variant of algorithm which does not utilize differential module $Z_t^{(k)}$. In such approach value of data point can be estimated simply as median value:

$$\tilde{y}_t^{(k)} = \widetilde{W}. \tag{6}$$

Threshold value of τ has to be chosen very carefully and needs to match to the variability level of each series. Therefore we try to make it independent of series characteristic by using median absolute deviation parameter of time window. In our approach we assume that

$$\tau = MAD\left(W_t^{(k)}\right) * n, \tag{7}$$

where n is the value of allowed multiplication of MAD. Median absolute deviation parameters is robust measurement of variability of given sample and can be defined as shown in equation (4).

$$MAD\left(W_t^{(k)}\right) = median(y_t^{(k)} - median\left(W_t^{(k)}\right)), \tag{8}$$

6 Performance evaluation

6.1 Test procedure

To check efficiency and accuracy of proposed anomaly detection method we performed several experiments. Impact of presence of differential module, window length and threshold value on anomaly detection system performance were tested. Experiments were performed with all types of data series. We used number of detected anomalies as a metric of sensitivity. Each test result was manually reviewed and assessed against anomaly detection rate and number false positive alerts.

6.2 Presence of differential module

Evaluation of both models with (4) and without differential module (6) on the same sample series has shown that use of differential module results in huge increase in number of points detected as anomalies. Results of experiments are shown in Table 1. For the evaluation we used the threshold value as three times of the MAD parameter, and window length of 60 probes.

Table 1. Influence of differential module presence on number of points detected as anomalies

Series Type	Differential module present	Differential module absent
Type I	332	332
Type II	34582	34490
Type III	56256	9518
Type IV	68422	12043

However when we rely on manual assessment of results, introduction of differential module resulted in extraordinary increase of false positive rate. Example results are shown below on Figure 7 and Figure 8.

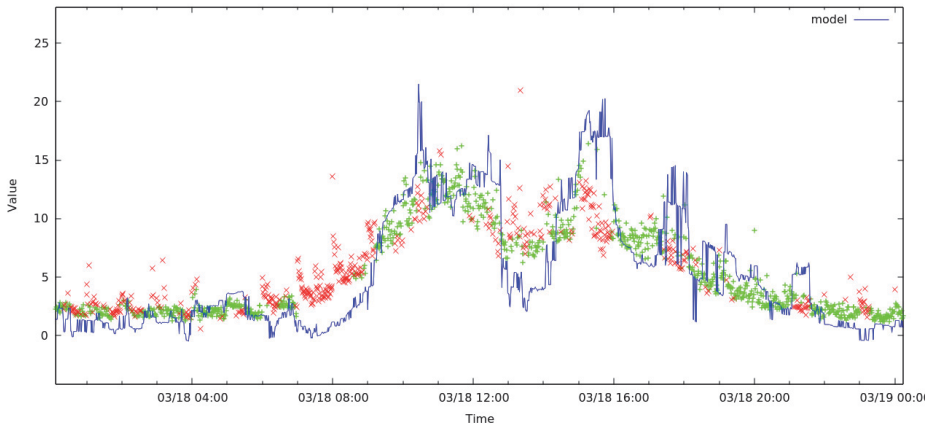


Figure 7. Estimation made by model with differential module enabled

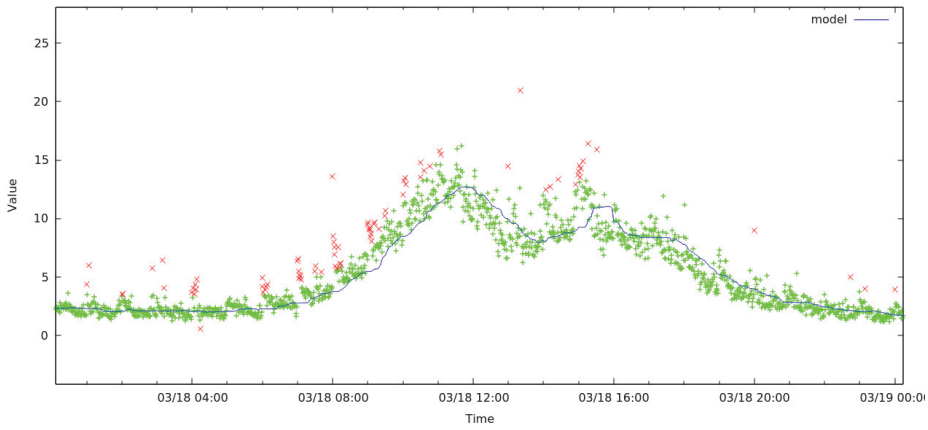


Figure 8. Estimation made by model with differential module disabled

Adding difference of following data points made median based estimation very variable, which lead to decrease of prediction accuracy. To show effect of this loss, test over a single generated series was performed. Evaluated series consists of stable value of zero with some random noise without any anomaly. Therefore estimated median should be fixed at the level of 0. As we may see in the Figure 9 estimated median has huge amplitude and vary a lot from the real median value.

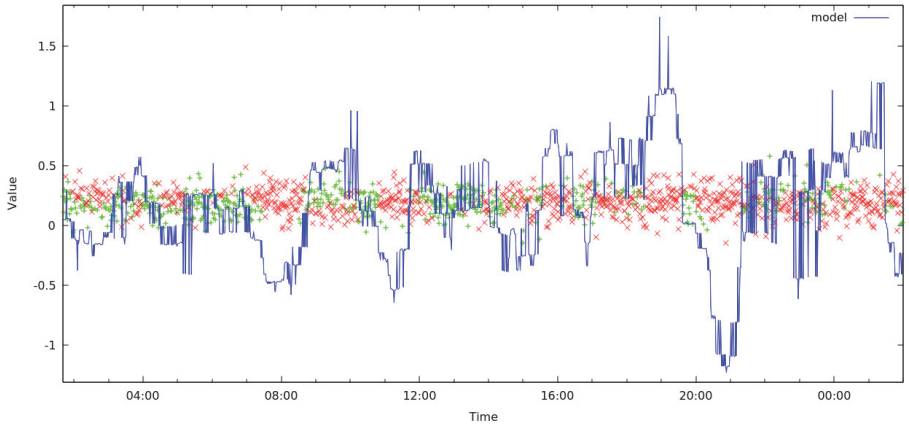


Figure 9. Effect of differential module on linear series with random noise

6.3 Sliding window length

Length of sliding window is key factor for gaining good prediction rate. Usage of small window can increase adaptivity of algorithm but will effect in increase of false positive rate. In the Table 2 number of points described as anomalies in function of window length and series type is presented. The threshold value was fixed at the level of three times of MAD value.

Table 2. Influence of time window length on number of points detected as anomalies

Series Type	**Window length**					
	15	30	45	60	75	**90**
Type I	385	328	364	332	327	329
Type II	34903	31584	33208	34490	35627	36233
Type III	10779	10140	9824	9518	9480	9687
Type IV	12214	11946	12040	12043	12210	12349
Series Type	**Window length**					
	105	120	135	150	165	**180**
Type I	333	312	312	312	312	312
Type II	36861	37036	37378	38077	38696	39515
Type III	9841	9997	10233	10825	11528	12349
Type IV	12566	12708	12905	13249	13590	13993
Series Type	**Window length**					
	210	240	270	300		
Type I	312	312	312	312		
Type II	40493	41284	41931	42647		
Type III	14078	15828	17289	18414		
Type IV	14986	16022	17193	18263		

In the Figures 10 and 11 algorithm reaction on anomaly is presented, chosen time window sizes are 15 and 60 respectively.

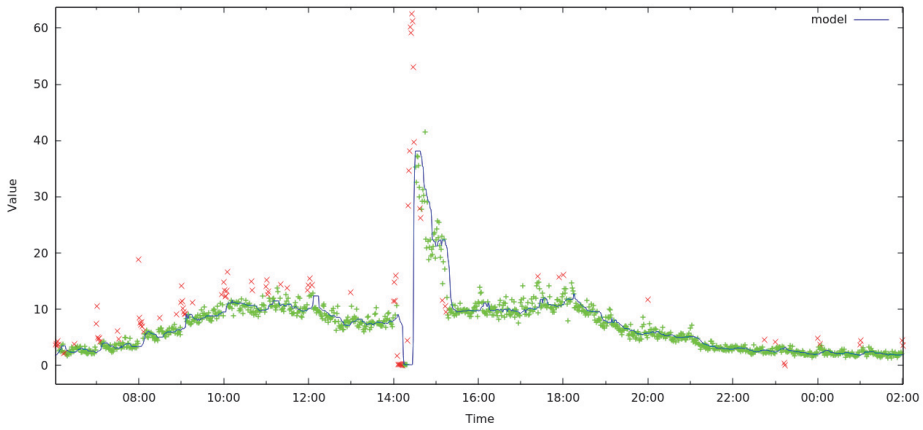


Figure 10. Anomaly detection with time window of 15 probes

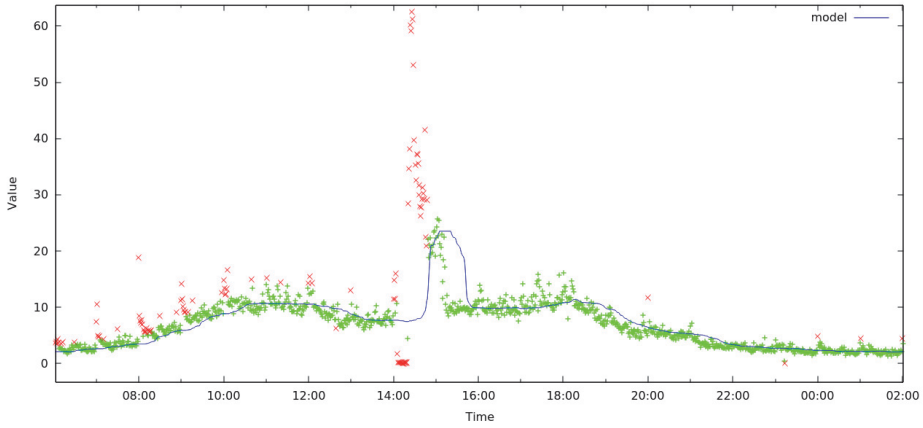


Figure 11. Anomaly detection with time window of 60 probes

For the small time window very fast adaptation of algorithm to anomalous level of metrics can be observed. Using larger size time window results in increasing of model stability.

However, time window can't be increased too much because at some point it will decrease a model accuracy, especially in time dependent time series such as type III and IV. On Figure 12 the same fragment of time series is presented, but this time window length is set to 360 points.

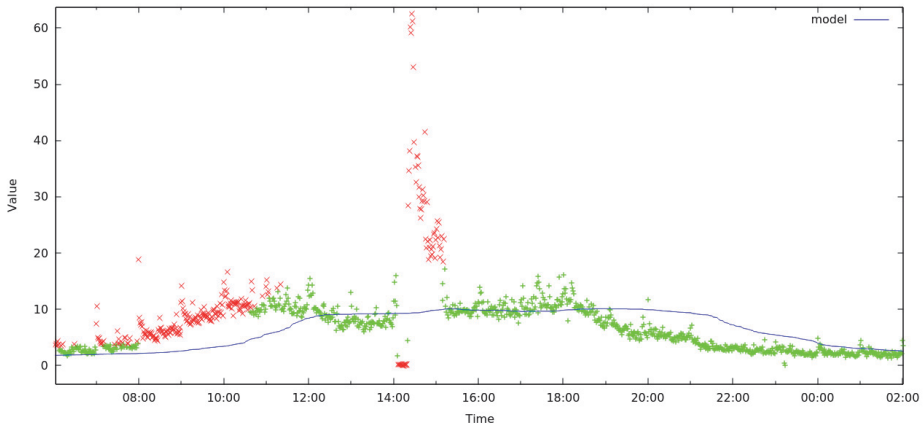


Figure 12. Too big time window

We can see decrease of prediction accuracy caused by not looking up to local trends of series. Estimated median value seems to be delayed in relation with real value. Loss of accuracy resulted in increase of false positive rate.

6.4 Threshold value

With the increase of threshold value used as multiply of MAD value, number of points described as anomalies start to decrease. Results of several test are shown in Table 3. Tests were performed with the window length of 60 probes.

Table 2. Influence of error threshold value on number of points detected as anomalies

Series Type	**Threshold value**					
	1	2	3	4	5	**6**
Type I	360	357	332	318	313	299
Type II	63697	43731	34490	29214	25668	23033
Type III	57995	19559	9518	5163	2950	1868
Type IV	57604	20701	12043	8851	7299	6342
Series Type	**Threshold value**					
	7			**8**		
Type I	290			289		
Type II	21048			19583		
Type III	1200			880		
Type IV	5664			5131		

On figures 13 and 14 two different threshold values are shown (3*MAD and 6*MAD).

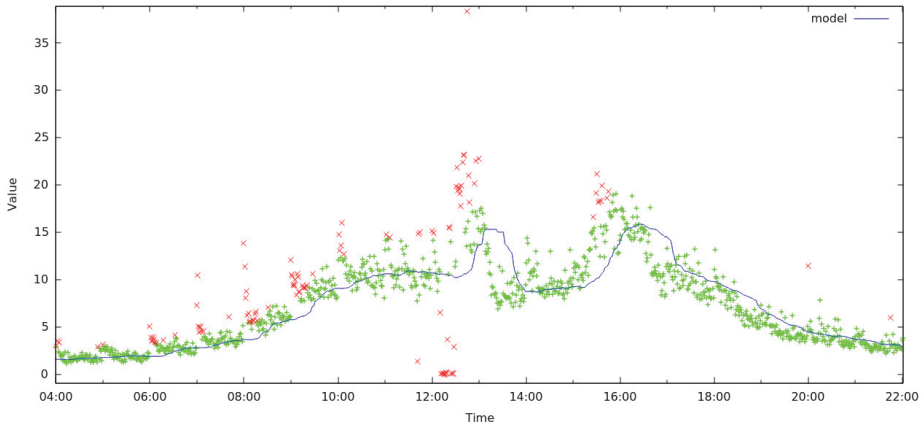


Figure 13. Threshold value set up to 3*MAD

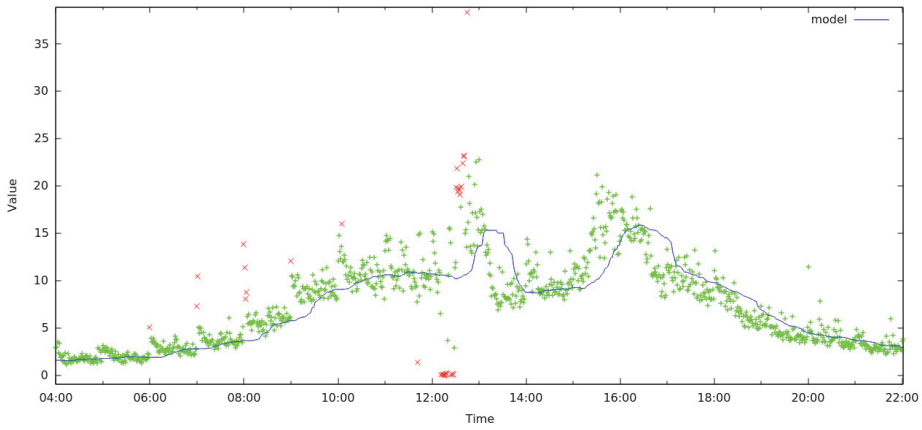


Figure 14. Threshold value set up to 6*MAD

We can see that increase of threshold results in decrease of sensitivity, however even use of 6*MAD level can properly detect critical anomalies.

6.5 Type of series

Usage of one sided median algorithm on all of analyzed Type I series showed that all existing anomalies can be easily discovered. Unfortunately one sided median algorithm can't be used to evaluate type II series. Two factor makes this kind of analysis really hard:

- Slow, gradual growth of anomaly is treated as a drift in data series values
- Presence of high level noise in the form of peaks determined by the source of this data series type.

High noise level is determined by the source of this series type. In some cases algorithm even mark good data points as anomalies which was caused by stabilized linear growing trend of anomaly.

It is really interesting that it is possible to discover anomalies in most common type III series with very good detection rate and small enough false positive rate by using well fitted sets of parameter. On the other hand use of one sided median algorithm over type IV series is limited due to marking every inconsistency as anomaly.

8 Conclusions

One sided median method of detection anomalies perform very well on metrics gathered from mail cluster. Based on the performed tests we can conclude, that this method is good choice for simple analysis of all kind uniform time series with rapid anomalies. We have shown that median absolute deviation (MAD) can be used as one factor of threshold value and allows model to be independent from series variability.

Results show also that use of differential module is not a good option for production series of metrics, as its introducing additional noise. It is possible that differential module can increase accuracy in series with constant visible trend such as constantly growing linear series.

Size of time window has to be correctly set to contain data points with low variance over median value. In case of Type III series appropriate value of this parameter is approximately 60 data points. We need to emphasis that in series of Type I time window length has no notable influence on the system performance. Number of points described as anomalies was similar between test results.

Threshold value has to be set as a compromise between sensitivity and false positive rate. For typical monitoring usage threshold value as three times the value of the parameter MAD should be enough to gain certain level of accuracy with good level of detection. For detecting only critical anomalies with very small level of false positive multiplication of 6 should be used.

This work shows that automatic anomaly detection over server metrics is possible even by using simple algorithms. One sided median is good enough to be used in production environment with success making administrators job simpler and systems more resistant to unpredicted failures.

References

1. Nairac A., Townsend N., Carr R., King S., Cowley P., Tarassenko L., 1999, *A System for the Analysis of Jet Engine Vibration Data* Integrated Computer Aided Engineering, Boulder, 6, 1, pp. 53–66.
2. Budalakoti S., Srivastava A., Akella R., Turkov E., 2006, *Anomaly Detection in Large Sets of High-dimensional Symbol Sequences,* Tech. Rep. NASA TM-2006-214553, NASA Ames Research Center
3. Eskin E., Arnold A., Prerau M., Portnoy L., Stolfo S., 2002, *A Geometric Framethe Analysis of Jet Engine Vibration Data* Integrated Computer Aided Engineering, Boulder, 6, 1, pp. 53–66.

4. Pan X., Tan J., Kavulya S., Gandhi R., Narasimhan P., 2010, *Ganesha: Black-Box Diagnosis of MapReduce Systems*, SIG-METRICS Performance Evaluation Review, 37, 3, pp. 8–13
5. Williams A.W., Pertet S. M., Narasimhan P., 2007, *Tiresias: Black-box Failure Prediction in Distributed Systems*, 21st Intl. Parallel and Distributed Processing Symposium (IPDPS), pp. 1-8
6. Silvestri G., Verona F., Innocenti M., Napolitano M., 1994*, Fault Detection using Neural Networks*, IEEE Intl. Conf. on Neural Networks, pp. 3796–3799
7. Angiulli F., Fassetti F., 2007, *Detecting Distance-based Outliers in Streams of Data*, 16th ACM Conf. on Information and Knowledge Management (CIKM), pp 811-820
8. Yang D., Rundensteiner E. A., Ward M.O., 2009, *Neighbor based Pattern Detection for Windows over Streaming Data*, 12th Intl. Conf. On Extending Database Technology: Advances in Database Technology (EDBT), pp 529-540
9. Tsay R.S., Pena D., Pankratz A.E., 2000, *Outliers in Multivariate Time Series*, Biometrika, 87, 4, pp. 789-804
10. Hill D. J., Minsker B. S., 2010, *Anomaly Detection in Streaming Environmental Sensor Data: A Data-driven Modeling Approach*, Environmental Modelling and Software, 25, 9, pp. 1014–1022
11. Justel A., Pena D., Tsay R.S, 2001, *Detection of Outlier Patches in Autoregressive Time Series*, Statistica Sinica, 11, 3, pp. 651-674
12. Luceno A., 1998, *Detecting Possibly Non-Consecutive Outliers in Industrial Time Seri*es, Journal of the Royal Statistical Society. Series B (Statistical Methodology), 60, 2, pp. 259-310
13. Basu S., Meckesheimer M., 2007, *Automatic Outlier Detection for Time Series: An Application to Sensor Data*, Knowledge and Information Systems – Special Issue on Mining Low-Quality Data, 11, 2, pp. 137-154

DE GRUYTER OPEN

JACSM 2017, Vol. 9, No. 1, pp. 23 - 47

10.1515/jacsm-2017-0002

FRIEDMAN AND WILCOXON EVALUATIONS COMPARING SVM, BAGGING, BOOSTING, K-NN AND DECISION TREE CLASSIFIERS

Vinai George Biju[1], Prashanth CM[2]

[1] Department of Computer Science and Engineering
Christ University Faculty of Engineering, India
vinai.george@christuniversity.in

[2] Department of Computer Science and Engineering
Sapthgiri College of Engineering, India

Abstract

This paper describes a number of experiments to compare and validate the performance of machine learning classifiers. Creating machine learning models for data with wide varieties has huge applications in predictive modelling across multiple domain of science. This work reviews state of the art techniques in machine learning classifiers methods with several extent of magnitude in statistics and key findings that will be helpful in establishing best methodological practices for class predictions. Comprehensive comparative review analysis with statistical validations for various machine learning algorithm for SVM, Bagging, Boosting, Decision Trees and Nearest Neighborhood algorithm on multiple data sets is carried out. Focus on the statistical analysis of the results using Friedman-Test and Wilcoxon Test as well as other interpretative metrics like classification rate, ROC, F-measure are evaluated to benchmark results.

Key words: bagging, boosting, SVM, KNN, decision tree

1 Introduction

Given the different types of input instances with output labels, predicting the output using machine learning tasks has been challenging for quite some time. The newly developed machine learning methods follows a rigorous criterion of analysis against previous approaches to verify its correctness of predictions. The results rely on choosing possibilities between output cases and empirical comparisons measuring the performance derived from the configuration parameters of the experiments. In order to set up on a firm conclusion on a radical learning technique, the statistical validation of

produced results is a requisite in current times. Many approaches have been proposed in present-years contributing towards optimized and transformed features and there by using well known machine learning techniques with out assuming independence or relationships among attributes making interpretable, dense and accurate learning models. Classification is mostly beneficial when the examples collected in a database can be used as the foundation for making future decisions; e.g., for judging risks for credit, analysing scientific data and for diagnosis of diseases taking biological data. Scientists have established extensive variety of classification algorithms namely decision tree, nearest neighbor, support vector machines, boosting, and bagging.

The comparative study should perhaps be done with utmost significance using a statistically adequate background. Pattern recognition with enhanced feature selection assigning groups or classes to data instances could be executed for either models that are based on supervised classification or models that extract relationships between objects and its properties namely clustering or unsupervised classification.

Even though plenty of work can be found in literature that describes more appropriate classifiers for particular tasks, only limited studies reflect a more systematic statistical analysis with regards to their performance. The typical initial outcome of this work is to find the performance of various machine learning classifiers under various parameter settings taking detailed input values from multiple data sets.

Evaluating classifiers giving priority to maximum accuracy alone under different classifier parameters for specific tuned data and values is usually not the best approach, because for a different dataset the result would be different for most of the cases. Since the key study in this work is evaluation of practical results comparing classifiers, the outcome of classifiers with generative models are compared to the discriminative models. Specifically the effect of varied data sets on average classifier classification results performed with wide-ranging experiments are explored. The behavior of feature combination and class labels can be briefly explained using the framework with some of the machine learning techniques like kNN, SVM, Boosted and Bagged Trees. Data scientists typically investigate with different classifiers taking varied features and data sets to compare with specialized guidelines. It should be dealt with caution that the detailed experiments carried out, not applying specific statistical tests could lead to invalid inferences. The degree to which the contending classifiers, disagree or agree on output class values deliver evidence about reliability of classification output over perceived input data sets. The fraction of class instances that are positive and correctly predicted is indicated by classifier sensitivity and likewise specificity is the fraction of negative class instances that are correctly predicted [1].

Performances are evaluated for CHAID, neural network and logistic regression for imbalanced data set executed in an actual marketing application of a bank in [2]. The classifier performance for k-NN, Naive Bayes, SVM, LDA and Decision Tree are evaluated using characteristics including specificity, sensitivity, classification accuracy, computational time and kappa in [3]. Analysis of ROC towards results in machine learning, describing various challenges and providing concise substitute methods to ROC analysis like Lift chart, Calibration chart, Detection error trade-off curve had been discussed in [4]. Sentiment analysis and opinion mining for business analytics and market research scrutinizing word-of-mouth data for movie reviews are explored using support vector machines, neural network and bayesian decision tree in [5]. In [6] the influence of lexically normalized, naive, and semantic features on the performance of classifier for various diseases have been assessed using support vector machines. Statistical tests for evaluations of machine learning algorithms on several data sets using Wilcoxon signed ranks test and Friedman test is detailed in [7] The raisins superiority for agriculture is graded by means of machine learning techniques after selecting the best features using feature selection based on correlation in [8]. Data from wireless kinematic sensors for the job of physical movement recognition is taken for comparing the performance of AdaBoostM1 as the classifier of meta level with base level classifier C4.5 Graft in [9]. Investigating classifier performance with optimization to categorize non-randomized readings and classification of biomedical quotations for text selection using organized reviews are studied in [10]. Classifiers namely Support vector machines, Conditional Random fields and Latent Dynamic conditional random fields are compared for user intention understanding in analysing web search engines was shown in [11]. Soil profiles were analysed, sampled, selected and predicted for taxonomic soil class after investigating the classification power of data mining classifiers in [12]. Chi-Square Methods and R- Square techniques were used for high dimensional curve fitting using machine learning in [36]. A review was carried out for forecasting the share trading from the stock market database using state of the art machine learning in [35].

2 Support vector machines with kernel evaluation

To classify instances of two classes using SVM, the input data x is mapped to higher dimensional space geometry $S = \phi(x)$ and then devising an optimal hyperplane denoted by $w \cdot S - b = 0$ separating the two classes [13]. The function is expressed as:

$$f(x) = \langle w, \Phi(x) \rangle + b \tag{1}$$

which acts as decision boundary and is evaluated thereby using the function Φ that maps x to S space which is in higher dimension [14]. The distance is maximized for the set of data points $\Phi(x_k)$ that are consistent on the training set with hyperplane characterized by (w,b). The vector w is represented by: $w=\sum_{k=1}^{m}\alpha_k^* y_k \Phi(x_k)$ and the quadratic optimization problem:

$$\max_{\alpha} W(\alpha)=\sum_{k=1}^{m}\alpha_k-\frac{1}{2}\sum_{k,l}^{m}\alpha_k\alpha_l y_k y_l\left(K(x_k,x_l)+\frac{1}{C}\delta_{k,l}\right) \tag{2}$$

is solved through α_k^* [15].

Table 1. Classification Output Parametrics for Support Vector Machine.

DataSet	Time	MAE	RMS	Prc	Rec	Fm	PRC	Class%
Supermarket	1.69	0.36	0.6	0.41	0.64	0.5	0.54	63.71
BreastCancer	0.11	0.3	0.55	0.67	0.7	0.68	0.63	69.58
Con-tactLenses	0.1	0.31	0.41	0.69	0.71	0.7	0.65	70.83
GermanCredit	1.54	0.25	0.5	0.74	0.75	0.74	0.68	75.1
PimaDiabetes	0.03	0.23	0.48	0.77	0.77	0.76	0.7	77.34
Glass	0.81	0.21	0.32	0.52	0.56	0.52	0.48	56.07
Hypothyroid	7.73	0.26	0.32	0.89	0.94	0.91	0.88	93.61
Ionosphere	0.36	0.11	0.34	0.89	0.89	0.88	0.83	88.6
Iris	0.13	0.23	0.29	0.96	0.96	0.96	0.94	96
Labor	0.21	0.11	0.32	0.89	0.9	0.89	0.85	89.47
Soybean	1.81	0.09	0.21	0.94	0.94	0.94	0.91	93.85
Vote	0.36	0.04	0.2	0.96	0.96	0.96	0.94	96.09
Weather	0.05	0.43	0.65	0.53	0.57	0.54	0.54	57.14
Segment	0.59	0.21	0.3	0.92	0.92	0.92	0.88	91.93
Weather	0.05	0.43	0.65	0.53	0.57	0.54	0.54	57.14
Segment	0.59	0.21	0.3	0.92	0.92	0.92	0.88	91.93

The solution to the above problem is established using the Lagrangian formulation and it is shown that $\sum_{k=1}^{m} y_k\alpha_k=0$ and $\forall_k, \alpha_k\geq 0$, where $\delta_{k,l}$ denotes Kronecker symbol with $K(x_k,x_l)=\langle\Phi(x_k),\Phi(x_l)\rangle$ representing the Gram matrix data set used for training. The predicted class label for each x can be computed after examining the sign of $f(x)$.

The mapped data $\Phi(x_k)$ could be contained in the smallest sphere of radius R. The radius margin bound $E \le 4R^2 \|w\|^2$ is evaluated to determine E i.e leave one out error bounds for SVMs. The distance S_p between a support vector $\Phi(x_p)$ that is mapped and the span of all other support vectors $E \le \sum_p \alpha_p^* S_p^2$ is used to devise methodically a tighter bound called span estimate. SVM with the variable S_p^2 and the quadratic slack variables ξ is dependent on $K_{sv} = \begin{pmatrix} K & 1 \\ 1^T & 0 \end{pmatrix}$ which is the the dot product between support vectors extended matrix by the equation $S_p^2 = 1/(K_{SV}^{-1})_{PP}$. We have made use of linear Kernel represented by $k(x^i, x^j) = \langle x^i \cdot x^j \rangle$ and quadratic kernel denoted $k(x^i, x^j) = (\langle x^i \cdot x^j \rangle + 1)^2$ for classifying instances using SVM.

If the number of instances are fewer than no of features representing the dimension space, it would result in an under par performance. It would definitely be an undetermined problem to find a hyperplane that fits the data in such cases. Then maximizing the margin with optimal parameters in SVM to find a solution will not be sufficient enough. Retaining only the features that are relevant, the dimensionality of the input space could be reduced [16].

The L1 soft-margin expression which is the fundamental problem for SVMs is solved by

$$\min \frac{1}{2} \| w \|^2 + C \sum_i \xi_k \text{ where } y_k (w \cdot z_k - b) \ge 1 - \xi_k, \xi_k \ge 0 \; \forall_k \tag{3}$$

This computational problem explained by its dual form through the kernel function implementing the non linear transformation.

$$\max \sum_i \alpha_i - \frac{1}{2} \sum_k \sum_j \alpha_k \alpha_j y_k y_j k(x_k, x_j) \text{ where } 0 \le \alpha_k \le C \; \forall_k \sum_k y_k \alpha_k = 0 \text{ where } k(x_k, x_j) = \phi(x_k) \cdot \phi(x_j) \tag{4}$$

Gaussian kernel represented by $k(x_k, x_j) = \exp\left(-\frac{\|x_k - x_j\|^2}{2\sigma^2} \right)$ and Polynomial kernel denoted by $k(x_k, x_j) = (1 + x_k \cdot x_j)^d$ are other popular kernel functions used in this paper.

Table 2. Classification Output Parametrics for Decision Stump.

DataSet	Time	MAE	RMS	Prc	Rec	Fm	PRC	Class%
Supermarket	0.13	0.42	0.46	0.68	0.64	0.65	0.64	64.4
BreastCancer	0.05	0.38	0.44	0.68	0.69	0.68	0.63	68.53
ContactLenses	0.01	0.23	0.36	0.71	0.71	0.68	0.71	70.83
GermanCredit	0.08	0.38	0.43	0.49	0.7	0.58	0.68	70
PimaDiabetes	0.05	0.38	0.44	0.72	0.72	0.72	0.68	71.88
Glass	0.01	0.18	0.3	0.21	0.45	0.28	0.34	44.86
Hypothyroid	0.21	0.03	0.12	0.95	0.95	0.95	0.95	95.39
Ionosphere	0.15	0.27	0.37	0.86	0.83	0.81	0.75	82.62
Iris	0.01	0.22	0.33	0.5	0.67	0.56	0.67	66.67
Labor	0.01	0.21	0.34	0.81	0.81	0.8	0.84	80.7
Soybean	0.01	0.08	0.2	0.13	0.28	0.16	0.21	27.96
Vote	0.01	0.08	0.2	0.96	0.96	0.96	0.93	95.63
Weather	0.02	0.49	0.59	0.29	0.29	0.29	0.58	28.57
Segment	0.11	0.21	0.32	0.11	0.3	0.16	0.28	30.4

3 DECISION TREES FOR INDEPENDENT OBSERVATIONS

A decision tree is a directed acyclic graph form of tree classifier. There is no incoming edges for the root of the tree and every internal node have outgoing edges with an incoming edge [17]. We apply binary decision trees in this study so that every node has outgoing edges either with number zero or two. The *leafnode* does not have any outgoing edges and is labeled with a class label. The splitting attribute X_n or predictor attribute is associated with each internal node. If X_n denotes a numerical attribute, then q_n which is the splitting predicate holds the form $X_n \le x_n$ and $x_n \in dom(X_n)$ where x_n is called the split point of node n. If X_n denotes a categorical attribute, then q_n holds in the form $X_n \in J_n$ where $J_n \subset dom(X_n)$ and J_n represents the splitting subset at the node n [18]. A classification tree is typically built using training data in two phases namely growing phase and pruning phase. The split selection techniques producing binary splits at each node is usually established on impurity-based method [19]. The problem of decision tree induction formally giving background terminology is indicated as follows: Let random variables be represented as $X_1,...,X_m$, C. The domain of X_i is

denoted as $dom(X_i)$ and $dom(C) = \{1, 2, ..., k\}$. The decision Tree classifier is represented as a function $d: dom(X_1) \times ... \times dom(X_m)$ $dom(C)$. Let the probability distribution be represented as $P(X', C')$ and a random record $t = \langle t.X_1, ...t.X_m, t.C \rangle$ be drawn from P where $\langle t.X_1, ...t.X_m \rangle \in X'$ and $t.C \in C'$ [20]

Decision tree learning induction using complete observations is as follows: For each data point and its neighbors $x_i, i = 1, ..., k$, along with a ranking associated as $\sigma_i \in \Omega$, the probability distribution is denoted as $P(. \mid x)$ on Ω which is locally constant. Since the observations are assumed to be independent and $\sigma = \{\sigma_1, ...\sigma_k\}$ with the parameters (θ, π), the probability is observed as

Table 3. Classification Output Parametrics for J48.

DataSet	Time	MAE	RMS	Prc	Rec	Fm	PRC	Class%
Supermarket	0.28	0.46	0.48	0.41	0.64	0.5	0.54	63.71
BreastCancer	0.11	0.37	0.43	0.75	0.76	0.71	0.65	75.52
ContactLenses	0.08	0.15	0.32	0.85	0.83	0.84	0.81	83.33
GermanCredit	0.39	0.35	0.48	0.69	0.71	0.69	0.66	70.5
PimaDiabetes	0.28	0.32	0.45	0.74	0.74	0.74	0.73	73.83
Glass	0.03	0.1	0.29	0.67	0.67	0.67	0.61	66.82
Hypothyroid	0.58	0	0.04	1	1	1	1	99.58
Ionosphere	0.37	0.09	0.29	0.92	0.92	0.91	0.88	91.45
Iris	0.06	0.04	0.16	0.96	0.96	0.96	0.92	96
Labor	0.05	0.32	0.47	0.75	0.74	0.74	0.68	73.68
Soybean	0.31	0.01	0.08	0.92	0.92	0.91	0.92	91.51
Vote	0.17	0.06	0.17	0.96	0.96	0.96	0.96	96.32
Weather	0.01	0.29	0.48	0.63	0.64	0.63	0.81	64.29
Segment	0.4	0.01	0.11	0.96	0.96	0.96	0.95	95.73

$$P(\sigma \mid \theta, \pi) = \prod_{i=1}^{k} \frac{\exp - \theta D(\sigma_i, \pi)}{\phi(\theta)} \tag{5}$$

The parameter (θ, π) has the maximum likelihood estimation for π given as $\hat{\pi} = \arg\min_{\pi} \sum_{i=1}^{k} D(\sigma_i, \pi)$ [21].

4 Aggregated bagging for bootstrap samples

Classifier optimization worked over estimation of error rate and model selection while learning from sample data sets can conclude in bias and over fitting [22]. This could result in an unstable classification model being generated and could be improved by the aggregation of classifiers. Bagged classification trees could solve to reduce misclassification error substantially in most of the applications and bench mark problems [23]

Table 4. Classification Output Parametrics for Bagging.

DataSet	Time	MAE	RMS	Prc	Rec	Fm	PRC	Class%
Supermarket	2.33	0.46	0.48	0.41	0.64	0.5	0.54	63.71
BreastCancer	0.09	0.38	0.45	0.64	0.69	0.64	0.69	69.23
ContactLenses	0.03	0.31	0.4	0.53	0.58	0.55	0.77	58.33
GermanCredit	0.28	0.33	0.42	0.73	0.75	0.73	0.77	74.7
PimaDiabetes	0.51	0.32	0.41	0.75	0.76	0.75	0.81	75.78
Glass	0.03	0.12	0.24	0.71	0.72	0.71	0.76	72.43
Hypothyroid	1.4	0	0.05	1	1	1	1	99.52
Ionosphere	0.47	0.14	0.26	0.91	0.91	0.91	0.95	91.17
Iris	0.09	0.05	0.17	0.94	0.94	0.94	0.98	94
Labor	0.29	0.3	0.38	0.84	0.84	0.84	0.86	84.21
Soybean	0.39	0.03	0.11	0.84	0.86	0.84	0.92	85.65
Vote	0.3	0.07	0.17	0.96	0.96	0.96	0.98	95.63
Weather	0.02	0.53	0.56	0.38	0.5	0.43	0.44	50
Segment	0.7	0.02	0.1	0.96	0.96	0.96	0.99	95.87

Let $L = \{(x_k, y_k), k = 1,..., N\}$ denotes learning from N observations of independent sample that comprise of predictors which are $q-$ dimensional vectors denoted by $x_k = (x_{k1},..., x_{kp}) \in R^P$. The learning sample have observations assumed to be identical distributed and random variables that are independent with a distinct distribution function F_L where $(x_1, y_1),..., (x_N,..., y_N) \sim F_L$. The class denoted y-values for the subsequent data is predicted by the classifier $C(x, L)$ from a set of vector of parameters x established through the learning sample L [24].

Let the distribution be denoted by $F_{x,y}$ for future observations represented by (x, y). To maintain stability for the classifiers C over averaged multiple learning samples, classifier C_A is aggregated for the observation x and is illustrated as $C_A(x) = E_{FL}C(x, L)$. The learning samples L and its expectation is distributed accordingly as F_L.

The aggregated rule $C_A(x)$ applying bootstrap as shown by $\hat{C}_A(x) = E_{FL}C(x, L^*)$, is measured by bagging where L^* denotes a random sample from the formulated distribution evaluated from samples and is denoted by the function $F_L.(x_1^*, y_1^*),...,(x_N^*, y_N^*) \sim F_L$.

Based on B the bootstrap samples, the bagged classifier $\hat{C}_A^B$ is computed as follows. Initially B samples L of size N are drawn randomly $L^{*(1)},...,L^{*(B)}$ with replacement.

The iterative algorithm for Bagging is shown as follows:

1. The bootstrap sample $L^{*(b)}$ is used to create the classifier C.
2. Classifier model is constructed iteratively for all bootstrap samples $b = 1,...,B$.
3. A new instance x is classified as,

$$\hat{C}_A^B = \arg\max_{j\in\{1,2\}} \sum_{b=1}^{B} \chi_{\{j\}}(C(x, L^{*(b)})) \tag{6}$$

we apply majority voting where χ is the indicator function

$$\chi_Z(x) = \begin{cases} 1 & x \in Z \\ 0 & else \end{cases} \tag{7}$$

To summarize performance of the bagged trees with smaller number of splits with smaller node size is found to be better in some data distributions than maximal unpruned trees and that the application requires careful tuning of the relevant classifier parameters while applying bagging [25].

5 Combining hypothesis with boosting

In boosting the weak rules or hypotheses which are moderately accurate are combined to design a classification rule that are highly accurate [26]. A single rule combined hypothesis is then linearly combined from these weak hypotheses. The predictive model function denoted by $f : \chi \to R$ is designed so that for example x and $f(x)$, the sign illustrated as (-1 or +1) indicates the predicted class and the magnitude $|f(x)|$ is evaluated as the confidence measure while creating a predictive model for learning [27].

The training sample S represented by: $S = \{(x_i, y_i)\}_{i=1}^{m}$ contains input features x and output label y.

Let D_1 be the distribution and is initialized for all $D_1(i) = 1/m$, $\forall i\, 1 \le i \le m$ .The weak hypothesis $h_t : \chi \to R$ is found and later $\alpha_t \in R$ is chosen to update the distribution $D_t, \forall i, 1 \le i \le m$ and $D_{t+1}(i) = \dfrac{D_1(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$. Here Z_t is selected with distribution D_{t+1}.

Finally the hypothesis is combined and returned as $f(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$ [28].

Table 5. Classification Output Parametrics for LogitBoost.

DataSet	Time	MAE	RMS	Prc	Rec	Fm	PRC	Class%
Supermarket	1.16	0.31	0.4	0.76	0.76	0.76	0.81	76.03
BreastCancer	0.09	0.36	0.44	0.7	0.72	0.71	0.72	72.38
ContactLenses	0.05	0.19	0.37	0.75	0.75	0.75	0.84	75
GermanCredit	0.22	0.36	0.43	0.68	0.71	0.68	0.75	70.8
PimaDiabetes	0.31	0.31	0.41	0.73	0.74	0.74	0.81	74.09
Glass	0.09	0.1	0.24	0.71	0.72	0.7	0.75	71.5
Hypothyroid	1.49	0.01	0.04	1	1	1	1	99.58
Ionosphere	0.23	0.14	0.28	0.91	0.91	0.91	0.95	91.17
Iris	0.2	0.05	0.18	0.94	0.94	0.94	0.94	94
Labor	0.06	0.15	0.31	0.89	0.9	0.89	0.91	89.47
Soybean	0.61	0.01	0.07	0.93	0.93	0.93	0.97	92.97
Vote	0.23	0.06	0.18	0.96	0.95	0.95	0.99	95.4
Weather	0.01	0.46	0.6	0.38	0.5	0.43	0.57	50
Segment	1.27	0.02	0.1	0.96	0.96	0.96	0.99	95.93

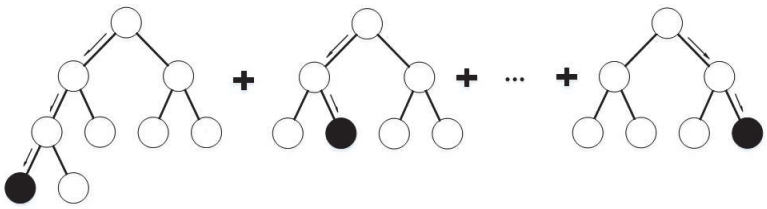


Figure 1. Boosted Tree Ensemble

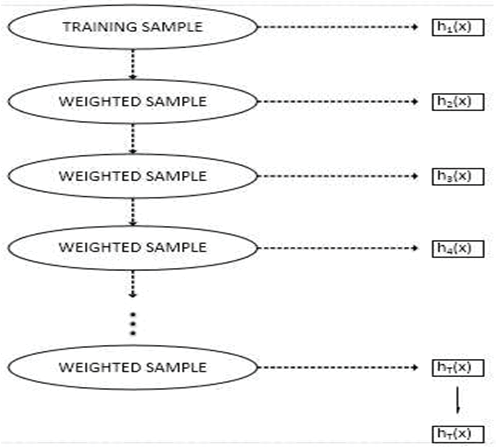


Figure 2. Weighted Boosting

Let the subset X_1 and X_0 be the examples for which p predicate holds true and does not hold true respectively. If π holds true, then $[[\pi]]$ be 1 corresponding to that predicate π and 0 otherwise.

The following values for W_b^j is evaluated when $b \in \{+1,-1\}$ and $j \in \{0,1\}$ for D_t, which represents the current distribution.

$$W_b^j = \sum_{i=1}^{m} D_t(i)\left[\left[x_i \in X_j \wedge y_i = b\right]\right] \tag{8}$$

Table 6. Classification Output Parametrics for AdaBoost.

DataSet	Time	MAE	RMS	Prc	Rec	Fm	PRC	Class%
Supermarket	1.42	0.33	0.41	0.74	0.75	0.74	0.79	74.86
BreastCancer	0.42	0.35	0.43	0.69	0.7	0.7	0.73	70.28
ContactLenses	0.12	0.36	0.41	0.72	0.71	0.69	0.7	70.83
GermanCredit	0.23	0.36	0.43	0.66	0.7	0.67	0.74	69.5
PimaDiabetes	0.2	0.31	0.42	0.74	0.74	0.74	0.8	74.35
Glass	0.01	0.18	0.3	0.21	0.45	0.28	0.34	44.86
Hypothyroid	0.37	0.03	0.12	0.91	0.93	0.92	0.97	93.21
Ionosphere	0.22	0.16	0.27	0.92	0.91	0.91	0.94	90.88
Iris	0.12	0.07	0.17	0.95	0.95	0.95	0.94	95.33
Labor	0.28	0.15	0.34	0.88	0.88	0.88	0.87	87.72
Soybean	0.01	0.08	0.2	0.13	0.28	0.16	0.21	27.96
Vote	0.23	0.06	0.19	0.95	0.95	0.95	0.99	95.4
Weather	0.02	0.49	0.63	0.53	0.57	0.54	0.52	57.14
Segment	0.03	0.21	0.32	0.11	0.3	0.16	0.28	30.4

W_b^j represents the weight of class b for the examples used for training in partition X_j which follows the distribution D_t. Setting $\alpha_t = 1$ and choosing $c_j = \frac{1}{2}\ln\left(\frac{W_{+1}^j}{W_{-1}^j}\right)$, the value for Z_t is minimized for a certain predicate. This background indicates that $Z_t = 2\sum_{j\in\{0,1\}}\sqrt{W_{+1}^j\, W_{-1}^j}$ and shows that for boosting's generalization error derives an upper bound

$$\Pr[H(x) \neq y] \leq \Pr[H(x) \neq y] + \tilde{O}\left(\sqrt{\frac{Td}{m}}\right) \tag{9}$$

The training examples are assumed to be generated over the probability distribution Pr[·] and the training sample generated over empirical probability distribution is denoted by P*r*[·].

Though the bound for d which is the space of VC-dimension for all likely base classifiers convert to be very feeble as the rounds T increases, prediction using AdaBoost will rapidly overfit with number of rounds which is usually moderate. Overfitting normally does not happen on the training examples by the notion of margins in the case of boosting. The margin $(x,y) = \frac{y\sum_t \alpha_t h_t(x)}{\sum_t \alpha_t}$

for the example (x, y) is based on the votes $h_t(x)$ along with α_t denoting the weights for all hypotheses [29].

The power of settlement for the base classifiers is indicated by the magnitude of the margin and the correct prediction combining votes is indicated by the sign it produces. The number of boosting rounds is independent on the bound and that the generalization error θ is maximum for the case as shown as:

$$\Pr[margin(x, y) \le \theta] + \tilde{O}\left(\sqrt{\frac{Td}{m\theta^2}}\right) \tag{10}$$

Table 7. Classification Output Parametrics for K Nearest Neighbor.

DataSet	Time	MAE	RMS	Prc	Rec	Fm	PRC	Class%
Supermarket	0.02	0.62	0.78	0.69	0.37	0.21	0.53	37.13
BreastCancer	0.02	0.33	0.51	0.7	0.72	0.7	0.69	72.38
ContactLenses	0.01	0.23	0.32	0.8	0.79	0.8	0.89	79.17
GermanCredit	0.02	0.28	0.53	0.72	0.72	0.72	0.67	72
PimaDiabetes	0.02	0.3	0.55	0.7	0.7	0.7	0.64	70.18
Glass	0.01	0.09	0.29	0.71	0.71	0.7	0.6	70.56
Hypothyroid	0.03	0.04	0.21	0.91	0.92	0.91	0.9	91.52
Ionosphere	0.01	0.14	0.37	0.87	0.86	0.86	0.81	86.32
Iris	0.01	0.04	0.17	0.95	0.95	0.95	0.93	95.33
Labor	0.01	0.19	0.41	0.83	0.83	0.83	0.79	82.46
Soybean	0.01	0.01	0.09	0.92	0.91	0.91	0.91	91.22
Vote	0.01	0.07	0.24	0.93	0.92	0.93	0.96	92.41
Weather	0.01	0.25	0.43	0.8	0.79	0.79	0.78	78.57
Segment	0.01	0.01	0.1	0.96	0.96	0.96	0.93	96.2

6 K-Nearest Neighbour with cost-distance metrics

The k-Nearest-Neighbours (kNN) is a effective but simple non-parametric classification technique. For classification of a data record t, a neighbourhood is formulated from its nearest k neighbours and retrieved using distance measure metric[30]. Considering weight-based distance, the class label for t is decided usually among the data records with majority voting in the neighbourhood of t [31].

Applying kNN requires choosing a suitable value for k, and the feat of classification is greatly dependent on the value of k. Since the kNN method is influenced by k and out of several ways of selecting the k value, a modest way is to execute the algorithm for several epochs with diverse k values and the one which supports the finest performance is chosen. In direction to kNN being not to be too much dependent on the selection of k, it is pre-eminent to observe sets of multiple nearest neighbours than rather just few k-nearest neighbour sets [32].

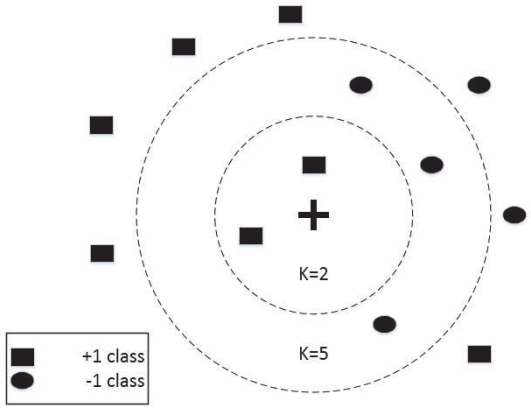


Figure 3. K-Nearest Neighbour

The extreme cost of kNN for classifying novel instances is mainly due to the reason that almost all computation happens during classification time rather than the training examples when first come across. To relieve the problem of heavy cost incurred storing the entire training set when it is very large, recent studies have attempted to eliminate the redundancy of the training set applied to k-Nearest-Neighbours classifier. kNN preserves the entire training data for classification and is a learning method which is case-based. Let $< Sim(d_i), Cls(d_i), Rp(d_i), Num(d_i) >$ represents the lower bound similarity of d_i to data values enclosed by N_i the class label of d_i, an illustration of d_i to itself and the data tuples enclosed by N_i respectively for the the model created M. The $Sim(d_i)$ value with minimum value is chosen, viz. representative with maximum density if equivalent maximal number of neighbours exist for more than one neighbour.

The classification algorithm is illustrated as follows:

1. For classification of a novel data tuple d_t, the similarity to every representation point in the model M is evaluated.

2. If only one representation point is covered for $d_t, < Sim(d_i), Cls(d_i), Rp(d_i), Num(d_i) >, d_t$ is classified as the grouping of d_j as followed by the Euclidean distance of d_j to d_t has a value less than Sim(d_j)
3. d_t is classified to be the category of the grouping with highest Num(d_j) if d_t is covered by minimum two symbolic diverse category, following the neighbourhood spanning the highest number of data tuples among the dataset used for training .
4. d_t is classified to be the category of grouping in which the boundary is nearest to d_t if there is no grouping in the model M that covers d_t.

The Euclidean distance of d_t to d_i subtracted with Sim(d_i) indicates the Euclidean distance of d_t to d_i represented with the closest boundary [33].

Let $\{(x_i, y_i)\}_{i=1}^n$ denote data used for training with *n* examples labeled using inputs $x_i \in R^d$ and class labels usually discrete y_i. The binary input matrix is used $y_{ij} \in \{0,1\}$ to show that the labels y_i and y_j match or otherwise. The goal learns a linear transformation which could be used to find squared distances: $D(x_i, x_j) = \| L(x_i, x_j) \|^2$. The cost function parameterized and concluding the distance metrics has significant terms penalizing heavy distances between each its target neighbors and input, while the other term penalizes minor distances between every inputs that does not form similar label and each input [34].

Precisely, the cost function is computed as:

$$y_t = \arg\min_{yt} \sum_j \eta_{tj} \| L(\overset{\sqcup}{x_i}, \overset{\sqcup}{x_j}) \|^2 + c \sum_{j, i=t \vee l=t} \eta_{tj}(1 - y_{il})[1 + \| L(\overset{\sqcup}{x_i}, \overset{\sqcup}{x_j}) \|^2 - \| L(\overset{\sqcup}{x_i}, \overset{\sqcup}{x_l}) \|^2] \quad (11)$$

7 Statistical significance using Friedman test and Wilcoxon test

Let *S* and *L* be the number of +ve class and -ve class in the data set, respectively; let S_+ denote the number of +ve classes that are correctly classified by a system, and S_- the number of +ve classes misclassified as -ve class. In the same way, let L_+ and L_- be the number of -ve classes classified by a system as +ve class and -ve class, respectively. These four values form a contingency table which summarizes the behavior of a system. The widely-used measures precision (*p)*, recall (*r*) and F_β are defined as follows:

$$p = \frac{S_+}{S_+ + L_+} \quad r = \frac{S_+}{S_+ + S_-} \quad F_\beta = \frac{(\beta^2 + 1)pr}{\beta^2 p + r} \tag{12}$$

We apply Friedman Test when we cannot assume that the data from each of groups are normally distributed populations. Blocks of data are assumed to be independent and the underlying variable in the data are mostly numeric in nature. When compared to F test, the Friedman rank test makes less stringent assumptions.The Friedman rank test concludes that the populations differs atleast from one of the other populations in variation, central tendency and shape. Friedman rank test also concludes if the input data groups have been generated from the whole original data set with the medians.

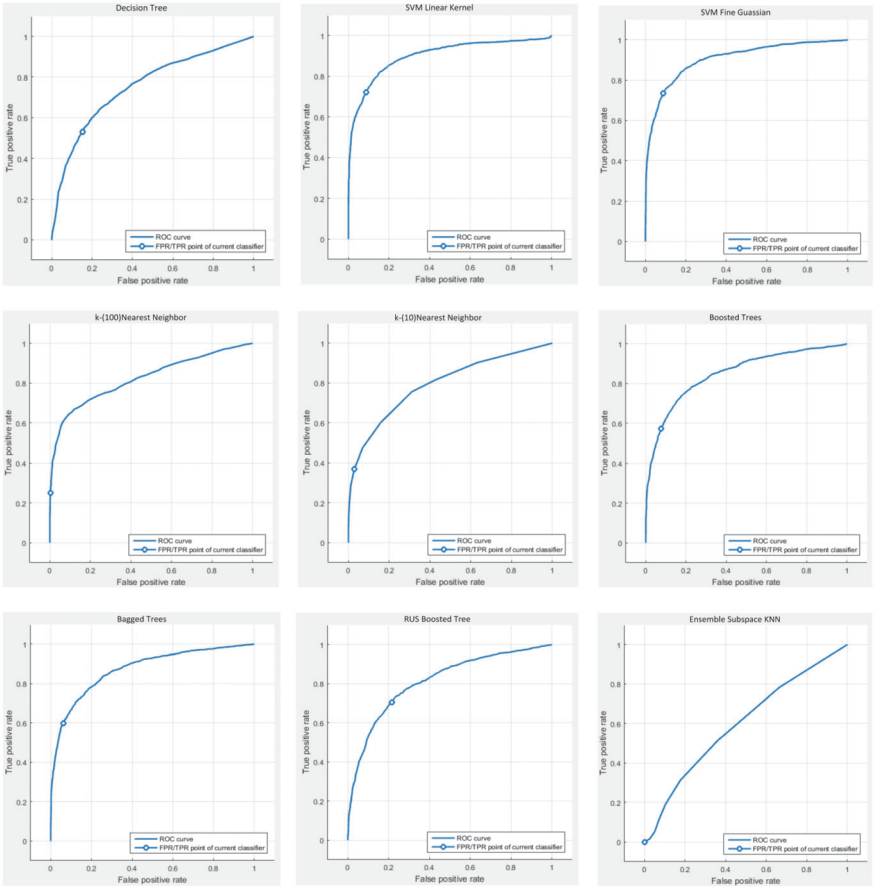


Figure 4. ROC evaluation for Decision Tree, SVM, KNN, Boosted and Bagged Tree Variants for Sparse Super Market Data.

Table 8. Friedman Test on Classifier Results.

N=14	Mean	StdD	Min	Max	FrdMR
SVM	79.95	14.63	56.07	96.09	4.68
KNN	79.67	15.45	37.13	96.2	3.75
DecisionStump	64.18	22.93	27.96	95.63	2.21
J48	81.59	13.2	63.71	99.58	5
Bagging	79.3	15.43	50	99.52	4.07
Adaboost	70.2	22.79	27.96	95.4	3.29
Logitboost	82.02	14.11	50	99.58	5
Bagging	79.3	15.43	50	99.52	4.07
Adaboost	70.2	22.79	27.96	95.4	3.29
Logitboost	82.02	14.11	50	99.58	5

Figure 5. Confusion Matrix evaluation for Decision Tree, SVM, KNN, Boosted and Bagged Tree Variants for Sparse Super Market Data.

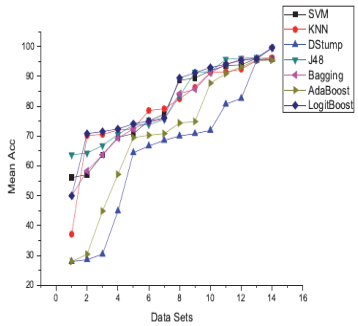


Figure 6. Classifier Mean Accuracy

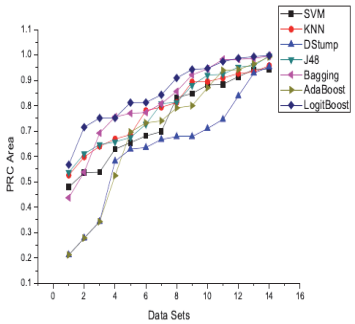


Figure 7. Classifier PRC Area

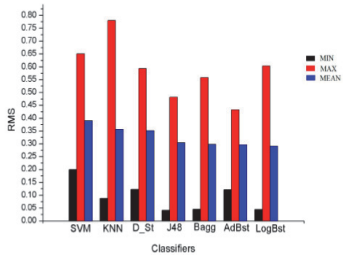


Figure 8. Classifier RMS

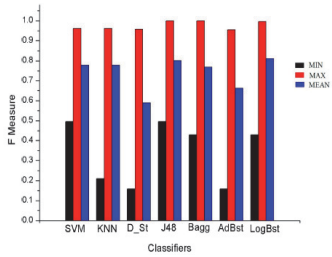


Figure 9. Classifier F-Measure

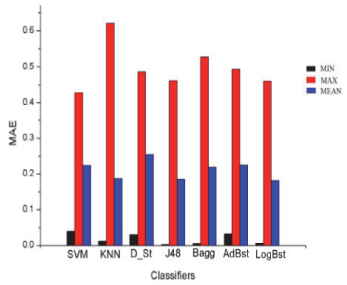


Figure 10. Classifier MAE

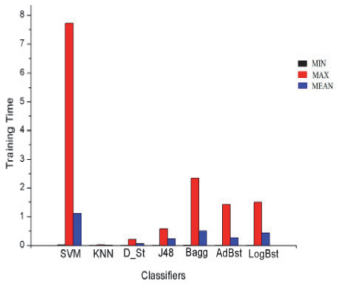


Figure 11. Classifier Training Time

The Classifier results are analysed using Friedman test under the following assumptions: One Data set is evaluated on three or more different classifiers. Training/Test set is generated as random sample from the population. Class/outcome variable is measured at the continuous or ordinal level. Samples are not necessarily normally distributed. The Wilcoxon signed rank test evaluates samples having size n greater than 10 observations and is evalu-

ated in pair of samples. Since W statistics is a non-parametric test, the multivariate normality is not essential to be assumed for the data. The Wilcoxon Signed Rank procedure evaluates under the illusion that the sample holds a frequency distribution that is symmetric and is from random population. The assumption which is symmetric never promises normality, as it is observed to have approximately the equal number of data points below and above the median.

The Wilcoxon technique evaluates a statistic for testing that is matched to an expected value. It is evaluated by summation of differences which is ranked along the deviation of every variable from a median. The Wilcoxon sign test compares the two dependent observations and quantifies the number of positive and negative differences.

Table 9. Wilcoxon Signed Ranks Test for Classifier Comparison.

Comparison	Ranking	Instances	MeanRank	SumOfRanks
J48-DecisionStump	Negative Ranks	2	5	10
	Positive Ranks	12	7.92	95
	Ties	0		
Bagging-DecisionStump	Negative Ranks	2	4.5	9
	Positive Ranks	11	7.45	82
	Ties	1		
Adaboost-DecisionStump	Negative Ranks	3	2.33	7
	Positive Ranks	7	6.86	48
	Ties	4		
Logitboost-DecisionStump	Negative Ranks	1	1	1
	Positive Ranks	13	8	104
	Ties	0		
Bagging-J48	Negative Ranks	8	7.25	58
	Positive Ranks	5	6.6	33
	Ties	1		
Logitboost-J48	Negative Ranks	6	7.5	45
	Positive Ranks	7	6.57	46
	Ties	1		
Adaboost-Bagging	Negative Ranks	8	7.75	62
	Positive Ranks	6	7.17	43
	Ties	0		
Logitboost-	Negative Ranks	4	4.75	19

Bagging	Positive Ranks	7	6.71	47
	Ties	3		
Logitboost-Adaboost	Negative Ranks	3	5.33	16
	Positive Ranks	10	7.5	75
	Ties	1		
KNN-SVM	Negative Ranks	9	6.22	56
	Positive Ranks	5	9.8	49
	Ties	0		
DecisionStump-SVM	Negative Ranks	11	7.73	85
	Positive Ranks	2	3	6
	Ties	0		
J48-SVM	Negative Ranks	4	6	24
	Positive Ranks	8	6.75	54
	Ties	2		
Bagging-SVM	Negative Ranks	9	6.22	56
	Positive Ranks	4	8.75	35
	Ties	1		
Adaboost-SVM	Negative Ranks	9	6.56	59
	Positive Ranks	3	6.33	19
	Ties	2		
Logitboost-SVM	Negative Ranks	6	5.33	32
	Positive Ranks	7	8.43	59
	Ties	1		
DecisionStump-KNN	Negative Ranks	10	8.3	83
	Positive Ranks	4	5.5	22
	Ties	0		
J48-KNN	Negative Ranks	5	7.6	38
	Positive Ranks	9	7.44	67
	Ties	0		
Bagging-KNN	Negative Ranks	6	7.33	44
	Positive Ranks	8	7.63	61
	Ties	0		
Adaboost-KNN	Negative Ranks	7	8.14	57
	Positive Ranks	6	5.67	34
	Ties	1		

Logitboost-KNN	Negative Ranks	5	6.6	33
	Positive Ranks	9	8	72
	Ties	1		

Table 10. Z Score and Significance on Wilcoxon Test

	KNN-SVM	DecStmp-SVM	J48-SVM	Bagg-SVM
Z	-0.22	-2.76	-1.177	-0.734
Asy.Sig	0.826	0.006	0.239	0.463
	Adbst-SVM	Logbst-SVM	DecStmp-KNN	J48-KNN
Z	-1.569	-0.943	-1.915	-0.91
Asy.Sig	0.117	0.345	0.056	0.363
	Bagg-KNN	Adbst-KNN	Logbst-KNN	J48-DecStmp
Z	-0.534	-0.804	-1.224	-2.668
Asy.Sig	0.594	0.422	0.221	0.008
	Bagg-DecStmp	Adbst-DecStmp	Logbst-DecStmp	Bagg-J48
Z	-2.551	-2.09	-3.233	-0.874
Asy.Sig	0.011	0.037	0.001	0.382
	Adbst-J48	Logbst-J48	Logbst-Bagg	Logbst-Adbst
Z	-1.977	-0.035	-1.245	-2.062
Asy.Sig	0.048	0.972	0.213	0.039

The significance is tested using the standard normal distributed z-value as shown in table 9 and table 10. The null hypothesis states that the median difference between pairs of classifier accuracy is zero. The null hypothesis is rejected when the significant value is less than 0.05 indicating one of the classifier outperforms the other. Here from table 11, Asy.Sig value of 0.826 indicates to accept the null hypothesis for KNN and SVM and Asy.Sig value of 0.006 shows that SVM and Decision Stump has statistically significant differences comparing the mean accuracy.

Conclusion

This work reviewed to assess various classification based machine learning techniques and investigated statistical evaluation measures to compare the results. Techniques for comparison and verification of classification results are Support Vector Machines, K-Nearest Neighbor, Decision Stump, J48,

Bagging, Logitboost and Adaboost. MAE, RMS, Precision, Recall, F-Measure, PRC-Area and Accuracy was considered for comparison of classifiers. Comparison of classifiers was executed using Weka 3 open source machine learning software and MATLAB 2016. Friedman and Wilcoxon test was executed using IBM SPSS and Data sets were taken from UCI machine learning repository. The statistical techniques used for validation of results are Friedman Test and Wilcoxon Signed Rank Test in non-parametric setting. Classification performance using SVM under Linear Kernel and Fine guassian framework was found much better than other classifiers for Sparse Supermarket Data. When the classifiers were compared with multiple data sets like Iris, Labor, Vote, German Credit, Breast Cancer, Glass and many others, Friedman Mean Rank was found high for J48 and LogitBoost. Pair wise comparison with statistical significance was evaluated using Wilcoxon Signed Ranks Test.

References

1. Labatut, Vincent, and Hocine Cherifi "Accuracy measures for the comparison of classifiers". *arXiv preprint arXiv*, 1207.3790, 2012.
2. Duman, Ekrem, Yeliz Ekinci, and Aydin Tanriverdi "Comparing alternative classifiers for database marketing: The case of imbalanced datasets". *Expert Systems with Applications,* 39.1, pp.48-53, 2012.
3. Aydemir, Onder, and Temel Kayikcioglu "Comparing common machine learning classifiers in low-dimensional feature vectors for brain computer interface applications". *International Journal of Innovative Computing, Information and Control* 9.3, pp.1145-1157, 2013.
4. Majnik, Matjaz, and Zoran Bosnic "ROC analysis of classifiers in machine learning: A survey". *Intelligent data analysis*, 17.3, pp.531-558, 2013.
5. Kim, Yoosin, Do Young Kwon, and Seung Ryul Jeong "Comparing machine learning classifiers for movie WOM opinion mining". *KSII Transactions on Internet and Information Systems* 9.8, pp.3178-3190, 2015.
6. Kotfila, Christopher, and Ozlem Uzuner "A systematic comparison of feature space effects on disease classifier performance for phenotype identification of five diseases". *Journal of biomedical informatics* 58, S92-S102, 2015.
7. Demsar, Janez "Statistical comparisons of classifiers over multiple data sets". *Journal of Machine learning research* 7.Jan, pp.1-30, 2006.
8. Mollazade, Kaveh, Mahmoud Omid, and Arman Arefi "Comparing data mining classifiers for grading raisins based on visual features". *Computers and electronics in agriculture* 84, pp.124-131, 2012.
9. Dalton, Anthony, and Gearoid OLaighin "Comparing supervised learning techniques on the task of physical activity recognition". *IEEE journal of biomedical and health informatics* 17.1, pp.46-52, 2013.
10. Bekhuis, Tanja, and Dina Demner-Fushman "Screening nonrandomized studies for medical systematic reviews: a comparative study of classifiers. Artificial intelligence in medicine 55.3, pp.197-207, 2012.

11. Deufemia, Vincenzo, et al. "Comparing classifiers for web user intent understanding". *Empowering Organizations. Springer International Publishing*, pp.147-159, 2016.
12. Taghizadeh-Mehrjardi, R., et al. "Comparing data mining classifiers to predict spatial distribution of USDA-family soil groups in Baneh region", Iran. *Geoderma* 253: 67-77, 2015.
13. Orru, Graziella, et al "Using support vector machine to identify imaging biomarkers of neurological and psychiatric disease: a critical review". *Neuroscience and Biobehavioral Reviews* 36.4, pp.1140-1152, .2012.
14. Qi, Zhiquan, Yingjie Tian, and Yong Shi "Robust twin support vector machine for pattern classification". *Pattern Recognition* 46.1, pp.305-316, 2013.
15. Geng, Yishuang, et al. "Enlighten wearable physiological monitoring systems: On-body rf characteristics based human motion classification using a support vector machine". *IEEE transactions on mobile computing* 15.3, pp.656-671, 2016.
16. Tehrany, Mahyat Shafapour, et al. "Flood susceptibility assessment using GIS-based support vector machine model with different kernel types". *Catena* 125, pp.91-101, 2015.
17. Azar, Ahmad Taher, and Shereen M. El-Metwally. "Decision tree classifiers for automated medical diagnosis". *Neural Computing and Applications* 23.7-8, pp.2387-2403, 2013.
18. Lajnef, Tarek, et al. "Learning machines and sleeping brains: automatic sleep stage classification using decision-tree multi-class support vector machines". *Journal of neuroscience methods* 250, pp.94-105, 2015.
19. Wang, Ran, et al. "Segment based decision tree induction with continuous valued attributes". *IEEE transactions on cybernetics* 45.7, pp.1262-1275, 2015.
20. Oliver, Jonathan J., and David J. "Hand On pruning and averaging decision trees. Machine Learning": *Proceedings of the Twelfth International Conference, Morgan Kaufmann*, pp.430-437, 1995.
21. Parvin, Hamid, Miresmaeil MirnabiBaboli, and Hamid Alinejad-Rokny. "Proposing a classifier ensemble framework based on classifier selection and decision tree". *Engineering Applications of Artificial Intelligence* 37, pp.34-42, 2015.
22. Simidjievski, Nikola, Ljupco Todorovski, and Saso Dzeroski "Predicting long-term population dynamics with bagging and boosting of process-based models". *Expert Systems with Applications* 42.22, pp.8484-8496, 2015.
23. Wang, Guan-Wei, Chun-Xia Zhang, and Gao Guo "Investigating the Effect of Randomly Selected Feature Subsets on Bagging and Boosting". *Communications in Statistics-Simulation and Computation* 44.3, pp.636-646, 2015.
24. Abdollahi-Arpanahi, R., et al. "Assessment of bagging GBLUP for whole-genome prediction of broiler chicken traits." *Journal of Animal Breeding and Genetics* 132.3, pp.218-228, 2015.
25. Hegde, Chiranth, Scott Wallace, and Ken Gray "Using Trees, Bagging, and Random Forests to Predict Rate of Penetration During Drilling". *SPE Middle East Intelligent Oil and Gas Conference and Exhibition. Society of Petroleum Engineers*, doi:10.2118/176792-MS, 2015.

26. Korytkowski, Marcin, Leszek Rutkowski, and Rafal Scherer "Fast image classification by boosting fuzzy classifiers". *Information Sciences* 327, pp.175—182, 2016.
27. Appel, Ron, Thomas J. Fuchs, Piotr Dollar, and Pietro Perona "Quickly Boosting Decision Trees-Pruning Underachieving Features Early". *In ICML* (3), pp.594-602, 2013.
28. Kim, Tae-Kyun, and Roberto Cipolla "Multiple classifier boosting and tree-structured classifiers". *Machine Learning for Computer Vision. Springer Berlin Heidelberg*, pp.163-196, 2013.
29. Ye, Jerry, Jyh-Herng Chow, Jiang Chen, and Zhaohui Zheng "Stochastic gradient boosted distributed decision trees". *In Proceedings of the 18th ACM conference on Information and knowledge management, ACM*, pp. 2061-2064, 2009.
30. Nowak, Bartosz A., et al. "Multi-class nearest neighbour classifier for incomplete data handling". *International Conference on Artificial Intelligence and Soft Computing. Springer International Publishing*, 2015.
31. Osth, John, William AV Clark, and Bo Malmberg "Measuring the Scale of Segregation Using k-Nearest Neighbor Aggregates". *Geographical Analysis* 47.1, pp.34-49, 2015.
32. Chavez, Edgar, et al. "Near neighbor searching with K nearest references". *Information Systems* 51, pp.43-61, 2015.
33. Bhulai, Sandjai "Nearest neighbour algorithms for forecasting call arrivals in call centers". *Intelligent Decision Technologies. Springer International Publishing*, pp. 77-87, 2015.
34. Blaszczyn'ski, Jerzy, and Jerzy Stefanowski "Neighbourhood sampling in bagging for imbalanced data". *Neurocomputing*, 150, pp.529-542, 2015.
35. Kamley S, Jaloree S, Thakur RS. "Performance Forecasting of Share Market using Machine Learning Techniques: A Review". *International Journal of Electrical and Computer Engineering*. 6(6):3196, 2016.
36. Vidyullatha P, Rao DR. Machine Learning Techniques on Multidimensional Curve Fitting Data Based on R-Square and Chi-Square Methods. *International Journal of Electrical and Computer Engineering*. 1;6(3):974, 2016.

DE GRUYTER
OPEN

JACSM 2017, Vol. 9, No. 1, pp. 49 - 63
10.1515/jacsm-2017-0003

ANALYSIS OF PERFORMANCE AND EFFICIENCY OF HARDWARE AND SOFTWARE FIREWALLS

Wojciech Konikiewicz[1], Marcin Markowski[2]

[1] Wroclaw University of Science and Technology,
Wybrzeze Wyspianskiego 27, 50-370 Wroclaw, Poland
wojciech.konikiewicz@onet.pl

[2] Department of Systems and Computer Networks,
Wroclaw University of Science and Technology,
Wybrzeze Wyspianskiego 27, 50-370 Wroclaw, Poland
marcin.markowski@pwr.edu.pl

Abstract

Firewalls are key elements of network security infrastructure. They should guarantee the proper level of security and, at the same time, the satisfying performance in order to not increase the packet delay in the network. In the paper, we present the comparative study on performance and security of a few firewall technologies including hardware, software and virtual solutions. Three important criteria are considered: the maximal throughput of firewall, the introduced delay and the ability to resist Denial of Service attacks. We report results of experiments, present analysis and formulate a few practical conclusions.

Key words: firewall, virtual firewall, network security, network performance, DoS attacks

1 Introduction

The security of telecommunication networks is one of the most important aspects of scientific research. When information is transmitted between users and servers, it often becomes the object of desire for unauthorized persons - the hackers, attempting to steal sensitive user data. Information security is especially important when the number of devices and end systems increases. IPS, IDS, anti-virus programs and, finally, firewalls are often placed on the border between the private and public networks. Choosing the proper security device is very important because it affects all the traffic passing between the local and external network.

Nowadays, firewalls are a mandatory part of the computer network in businesses, offices and other institutions. With the advancement of technology, these devices are constantly being developed. Their operation must be effective, quick and not noticeable to the potential users. There are many solutions available to protect the IT system. Some manufacturers provide their solutions for free, such as the programmable firewalls on Linux platforms, but there are also very expensive devices such as Cisco or Juniper hardware firewalls.

This paper focuses on selection of the best type of firewall for particular application. Optimal firewall should introduce the smallest packet latency in the network and, at the same time, provide a good protection level for user data. The goal of this work is to perform a comparative analysis of three types of firewall: two hardware solutions (Cisco ASA and Juniper), software solution installed on Linux (IPTables) and the virtual one(VyOS), implemented on a virtual machine. An analysis of the impact of individual firewalls on packet traffic in the network is based on bandwidth and server response time. We also analyze the level of resistance against the network attacks.

2 Firewall technologies

Firewall is a network device usually located at the border between two different (e.g. internal and external) computer networks. This is usually the place where the internal communication network of an enterprise is connected with the Internet. Its main task is to protect the network and data processed inside LAN. The firewall filters incoming and outgoing traffic. Thanks to certain rules, it is able to eliminate the unwanted traffic generated, for example, by an attacker. Firewalls control communication by deciding which packet is consistent with the security policy. Firewalls also isolate the restricted areas from the rest of the network. Firewall technologies can be divided into four basic groups: packet filtering, state control, network address translation (NAT), and proxy [2]:

- In *packed filtering* mode, device filters all incoming and outgoing packets, looking into the header information, i.e. IP addresses and port numbers. With defined Access Control List (ACL), only packets that are reflected in the security policy are allowed. It is important to start configuring the ACL with general (default) blocking rule, and after that to define which kind of traffic should be accepted. Filtering rules are usually defined separately for incoming and outgoing traffic [11].
- *Statefull firewall* is a powerful packet filtering technology, with control of the particular connection attributes. Unlike the packet filtering, it allows to monitor the connection status: whether the connection is in the

initiation, during data transfer, or in the termination state. Firewall tracks all the passing TCP sessions and drops packets, whose do not match any of known connections. Typically, the TCP rule is used for matching. This feature introduces a very high level of security, and it also offers satisfying transmission speed [14].

- *Network Address Translation* converts IP source (inside LAN) addresses into other (outside) addresses. This mechanism works on both sides, i.e. both outgoing and incoming packets are subject to this operation. This service does not have any built-in security services, but it allows to hide the internal architecture. Outbound packets live the local network with another IP address, so that the person or the external traffic tracking device is not able to see the local area network infrastructure [13].
- *Proxy Firewall* - this is a software package that gives an indirect access to the Internet. Communication on the network with the proxy server is split into two sessions: session between client and proxy service and session between proxy and remote destination server [14]. Client cannot connect directly to any server located in an external network.
- *Hybrid Firewall* is a combination of the above types of firewalls. In most applications it offers simultaneous packet filtering, the proxy services and allows to monitor the network traffic.

While the structure of the network is growing, the security devices evolve. At the turn of several years, three main types of firewall architecture (Figure 1) were created [11]:

- Hardware - a physical device that has its own resources: CPU, RAM, disk space. Similar to the router, having its own operating system.
- Software - a platform implemented on an existing operating system, using the resources of the server on which the OS is installed.
- Virtual - implemented as a virtual machine, most commonly used for packet filtering in SDN (Software Defined Networks) and for data protection in the cloud services. Thanks to the virtualization layer it is possible to change the hardware resources assigned to the machine [12].

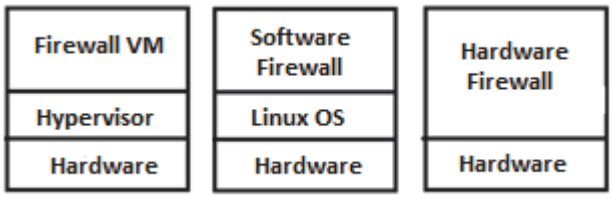


Figure 1. Comparison of firewall architectures

The hardware firewall is a stand-alone network device. It has dedicated components and the resources that it possesses are optimally tailored for correct and rapid work. Selecting a specific model of a hardware firewall, the manufacturers technical documentation should be carefully analyzed. An important feature of the hardware firewalls is that they are not dependent on third-part software. A software firewall is typically represented by a server with two network interfaces and a special application that is responsible for such functions as packet filtering, NAT or proxy. It controls the network traffic using configured bridge mode interfaces. All packets passing from the one subnet to the other are filtered according to the rules written by the administrator. Software firewalls do not have their dedicated resources. They use the resources of the operating system on which they are installed and cannot operate automatically. Software firewalls are very flexible, they can be extended with additional modules for proper operation, although their configuration is much more difficult since the majority of programs have only a textual interface. The advantage of the software firewall is that many free versions are available in the Internet. Virtual machines are running an environment monitored by the hypervisor. When multiple machines are running within a single virtual environment, a virtual network including all the physical network elements (routers, switches, and firewalls) is created [1]. Virtual firewall is responsible for the security of virtual host communication, but also for communication between the physical and virtual networks. Some virtual firewalls integrate additional network features such as VPN or QoS. Virtual firewalls do not have dedicated hardware resources but use the resources provided by the virtualization layer. The advantage of such solutions is the flexibility to change the hardware parameters of each machine.

3 Related Works

While surveying the scientific papers, we will not find an article or book comparing all types of firewalls. The main topics of the research are the optimization of device operation and the virtualization of particular elements of the backbone network. In [1] Author describes the use of the virtual gates and shows the basic differences between traditional and software firewalls. The advantages of non-physical applications, as well as the disadvantages of these technologies, are analyzed. The article does not present any exhaustive comparison, it just proposes the area of application of considered gate. Also the structure of the virtual network in which this device could be implemented have been proposed. In [2] Authors compare a few types of firewall technologies: packet filtering, statefull firewall, proxy, and hybrid firewall. The article does not contain any simulation data and therefore does not indicate the best system. Authors focused on the description on how the

firewall works and what its advantages are. The second part of the article deals with the subject of intrusion detection and prevention systems. The summary of the article is a table with the advantages and disadvantages of considered technologies.

Comparison of hardware and software firewall may be found in [3]. Cisco ASA 5500 (hardware), Check Point SPLAT (software) and Open BSD PF (software) were verified against the simulated DDoS (*Distributed Denial of Service*) attacks. Authors have shown that none of the firewalls are immune to this kind of threat. The results presented in the publication have been based on laboratory simulations and summarized in the table. According to the tests, all the firewalls showed similar performance, but SPLAT was the best one, able to survive 15 minutes attack. Another important parameter measured during this simulation was the CPU consumption level, best results were obtained for Cisco ASA. Devices listed in the above article are also the subject of research in papers [9] and [10]. Authors present a simulation-based comparison on the HTTP, FTP, UDP packet throughput and the number of possible connections. In [9] the security level of devices was also compared and some considerations on the degree of complexity of configuration, important when choosing a device by less experienced administrators, were presented. As the results have shown, both Cisco ASA and Check Point are doing very well with packet filtering, but Cisco hardware is the best one when taking into account the offered bandwidth. Paper [4] deals with the topic of firewalls, from definition to simulation. The study focuses on comparing commercial and free software firewalls. It includes both platforms configured under Unix operating systems (Linux, BSD, Solaris) and Windows (WS 2003). That work is based on an extensive simulation part, which is summarized by the graphs showing the packet delay dependence on the number of connections and the size of the packets. Summary of publications is a presentation of the disadvantages and advantages of each platform. The main advantage of the article is the well written theoretical part. The other articles discussing the subject of firewall comparison are [5] and [6]. Both compare hardware firewalls with software ones, but in [5] the considerations are purely theoretical. Author of [6] have investigated Cisco hardware firewall and platforms implemented on Linux. The comparison is based only on the data provided by the manufacturer and security tests made with basic security tools such as *nmap*. Very similar topic, a comparison of a firewall implemented on the Linux platform and the Cisco 2621 firewall, is addressed in [8]. That study shows the number of TCP packets passing through a device per unit of time. Definitely better results were obtained for Linux which, for the number of filtration principles 0-200, achieved two times higher bandwidth. Article [7] contains a comparison of the firewalls built into operating systems. Authors have generated identical traffic directed to two servers (Windows and Linux)

and investigated the CPU utilization. The results show that firewalls significantly affect the load of the platform on which they are implemented.

There are many works, publications and articles describing firewalls, but there is a restricted number of comparisons between all types of devices. Usually the hardware and software firewall comparison may be found. Since virtual firewalls are not yet very common then, in the literature, the architecture of the virtual systems is often considered. Comparisons mostly refer to Cisco devices as the leading physical ones, OpenBSD and Check Point as a software firewalls.

In this work we examine three types of firewalls: hardware, software, and virtual. We provide the comparative analysis and conclude, which of solutions ensure the best performance and the minimal impact on the network traffic.

4 Problem formulation and experimental setup

A network topology built of two computers and a traffic filter device (hardware firewall or dual-homed server with software/virtual firewall) was implemented for the experiments (Figure 2). One of the computers (SERVER) served as a server and was placed behind the firewall internal interface, the second one (PC) was placed in an external network zone. All analyzed firewalls were configured in the similar way in order to make the result comparable. The whole infrastructure was connected using category 5e UTP twisted pair copper cable.

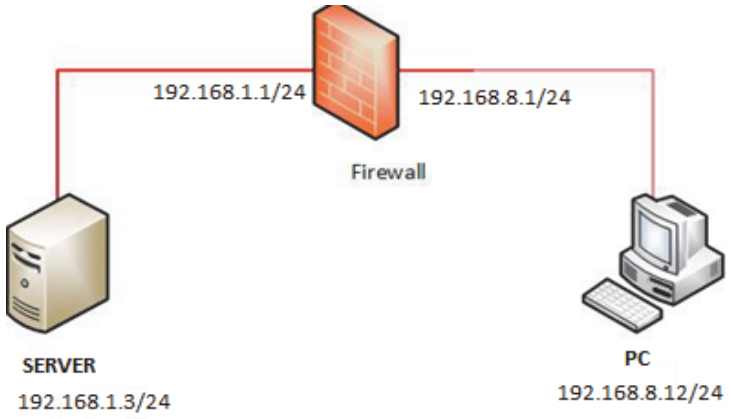


Figure 2. The network topology

Four firewalls were analyzed: IPTables, (software firewall), Juniper Netscreen 50 and Cisco ASA 5505 (hardware) and VyOS (virtual firewall). IPTables is a free software that is installed on the Linux operating system. It

has the ability to work from the second to the seventh ISO/OSI layer, then it can work as a comprehensive firewall. With the open license it is constantly being expanded with additional functionalities and support for additional protocols. The basic feature of IPTables used in this study is the packet filtering. It is based on the rules in the strings (equivalent of access-list), which are placed in the tables. The rules are the most important elements in the firewall configuration, because they determine whether the packet is accepted (ACCEPT) or rejected (DROP) [7].

The Juniper Netscreen 50 is a firewall with four Ethernet ports with a maximum throughput of 100 Mbps. The device supports two operating modes: transparent firewall and router with built-in firewall. In the former one, device acts as a second layer bridge and is invisible to other devices in the network. It filters packets according to established rules, but it has NAT disabled, because it cannot interfere with packet addressing as a second layer device. In the latter mode the firewall operates in the third layer and requires configuring the IP addresses of the individual interfaces. This allows NAT to be started [15]. An additional feature of Netscreen 50 is the ability to run the VPN functionality.

Cisco ASA 5505 has eight 10/100 Mbps network ports, two of them with a Power over Ethernet (PoE) functionality. Network interfaces of the firewall work in Layer 2 only, then it is impossible to configure IP addresses directly on the interfaces – they must be assigned to the appropriate virtual interfaces (VLANs). It is possible to assign each interface to another VLAN and isolate the subnets. VLANs can communicate with each other directly through the firewall, where packet filtering is applied. Devices on the same subnet exchange packets bypassing filtering. In order to divide the network into trusted and non-trusted interfaces, the security levels are defined and labeled from 0 to 100. The higher number, the higher security level. It is important that higher levels may access the lower-level interfaces, but not vice versa [16].

VyOS is a virtual platform with router and firewall functionalities, created in 2013 as a free network operating system. It is based on Debian and Quagga platform. VyOS configuration is provided through the CLI interface. It can be installed on virtual machines or on the cloud-based platforms. VyOS has been equipped with all the features of a physical firewall: packet filter, NAT service, VPN, and routing mechanisms. It is suitable for large and small networks as an alternative to physical devices, what remarkably reduce costs [17].

The common network diagnostic tools: *iperf*, *ping* and *hping* were used for the experiments. *Iperf* is a free network tool for measuring network bandwidth. It supports various protocols including: TCP and UDP. Thanks to the large number of parameters, it is very useful. For each performed test, it generates a report containing the connection throughput in the subsequent

time units [19]. *Ping* is a popular program used by the computer network administrators to diagnose the network performance, it is based on ICMP protocol. It allows to verify the connection between hosts, and measure the number of lost packets [18]. *Hping* is a tool for networks and devices analyzing. It can serve as a package generator and is often used for network audits. It supports protocols such as TCP and UDP. Additionally, it has features for sending files and the ability of package route tracking. *Hping* was originally created as a tool for the network testers, but is currently used by hackers as well [20], as able to carry out the *DoS* attacks (this option was used during experiments).

5 Experiments and Results

The goal of experiments was to obtain an comparative analysis of firewall solutions on their performance, efficiency and resistance to Denial of Service attacks. Considered criteria taken into account were: the throughput of firewall (in Mb/s), delay introduced by firewall and time of surviving during DoS attack.

For throughput investigations, *Iperf* tool was used to generate a high intensity traffic from PC to Server. In the consecutive experiments, different packed sizes l (in Bytes) were used, the intensity (in Mb/s) of generated traffic was always the same, equal to maximal possible line speed (around 100 Mb/s). The higher value of l, the smaller number of packets was sent during one second. As a baseline we have also measured a throughput in the direct connection between PC and Server (without firewall). Each single experiment lasted 60 seconds. Experiment with each packet size were performed a few times and results for each second were averaged. They are presented in Figures 3-6. It may be observed that the throughput of firewalls is unstable for l=200B and l=500B (Figure 3 and Figure 4). For l=200B the measured throughput was between 20 Mbps (for virtual firewall VyOs) and 80 Mbps. For l=500B (Figure 4) the significantly higher throughput was observed for VyOS (around 45 Mb/s), slighter improvement was noticed for the other firewalls, as well as for direct connection. Comparing results for both packet sizes it may be concluded, that the number of packets processed during one second is much about the same in case of VyOS – the higher number of packets, the higher throughput (in Mb/s).

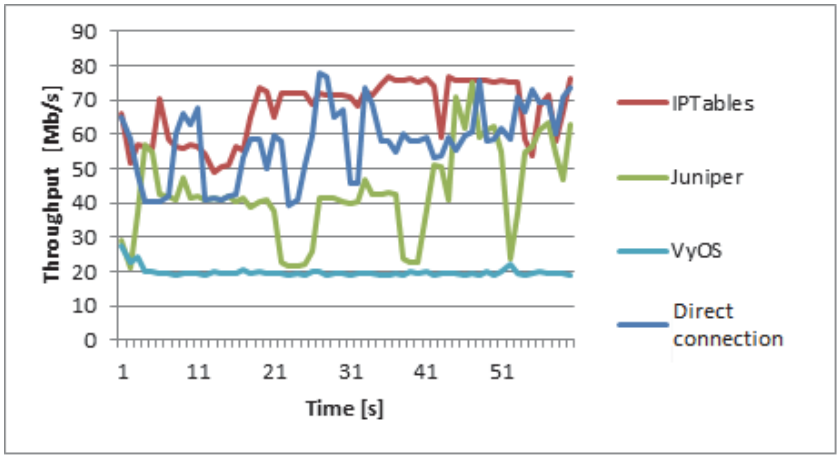


Figure 3. Comparison of the throughput for *l*=200B

Unexpectedly, the throughput offered by PC-based firewall (IPTables) was often higher than offered by dedicated network device (Juniper). The above observation is very interesting, since hardware firewalls are considered as offering much better performance in comparison with the multi-purpose computers. Also the throughput for the direct connection is very uneven, for *l*=200 the throughput of IPTables seems to be higher than throughput of direct connection. We may conclude, that for small packet size (and high number of packets per second) the performance of the PC network card or properties of TCP protocol (devices receives new packets and, at the same time, have to send acknowledgments of received packets) may hardly affect the results.

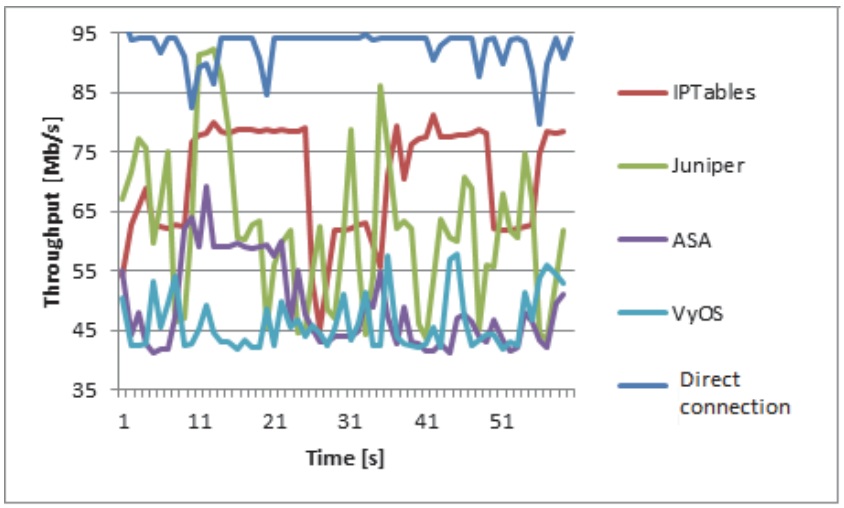


Figure 4. Comparison of the throughput for *l*=500B

Analyzing results for bigger packet sizes (Figure 5 and Figure 6), it can be observed that the throughput for ASA and IPTables are very close to the real bandwidth of the direct connection between computers. Those firewalls do not introduce any decrease in the network performance. A little bit worse and less stable results were obtained for Juniper. The lowest performance was noticed for the virtual firewall, where the value of throughput oscillated between 65 and 80 Mbps. For l=1500B, throughput of hardware firewall tends to be unstable. Comparing results presented in the Figures 5 and 6 we may conclude, that packet size equal to 1 kB was optimal in prepared testbed environment.

Average (taking into account values from each second of each experiment) values of throughput for all firewalls and sizes of packet are presented in the Figure 7. Improvement in the firewall performance with the growing size of

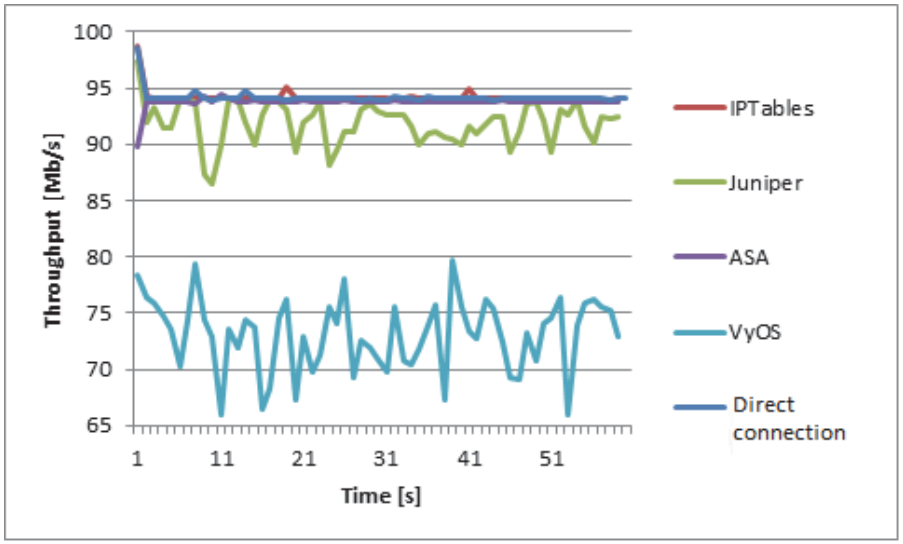


Figure 5. Comparison of the throughput for l=1000B

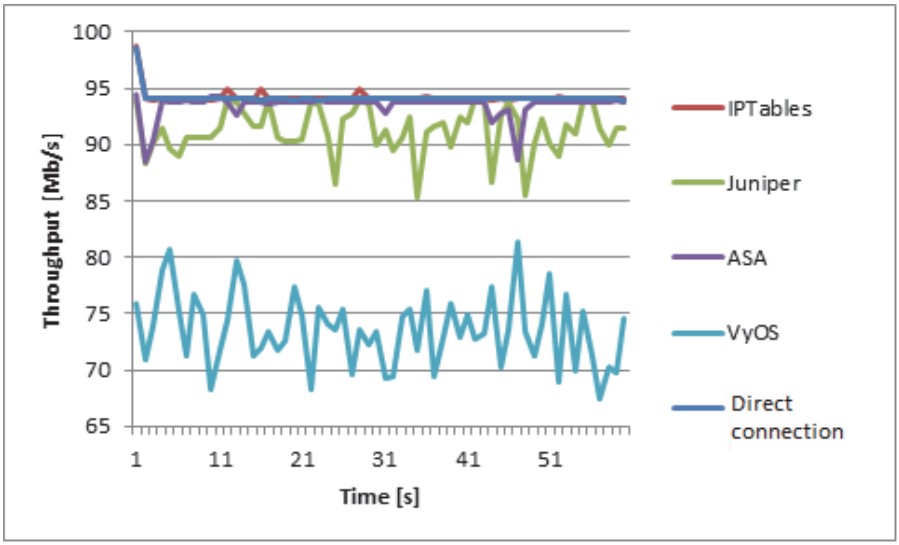


Figure 6. Comparison of the throughput for *l*=1500B

packet may be clearly seen. For *l*=200B all firewalls offered the least performance, but with the increase of the packet length the performance increased. For *l*=1000 and *l*=1500 the throughput reached a maximum value equal to the direct connection one. The graph shows that the slowest firewall turned out to be a virtual firewall, and hardware and software ones achieved very similar results.

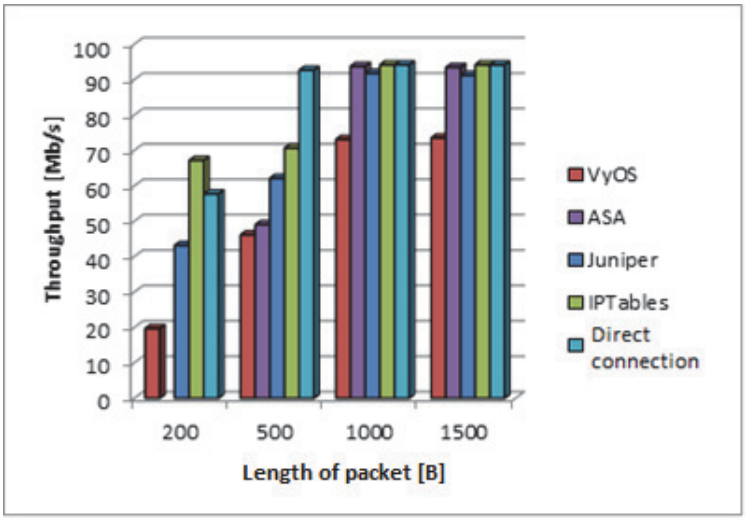


Figure 7. Average throughput for all firewalls

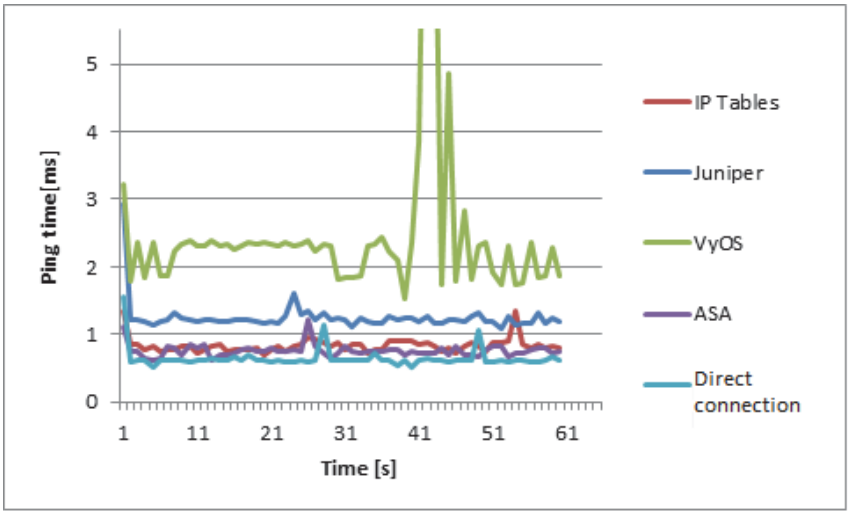


Figure 8. Ping response time for packet size 64B

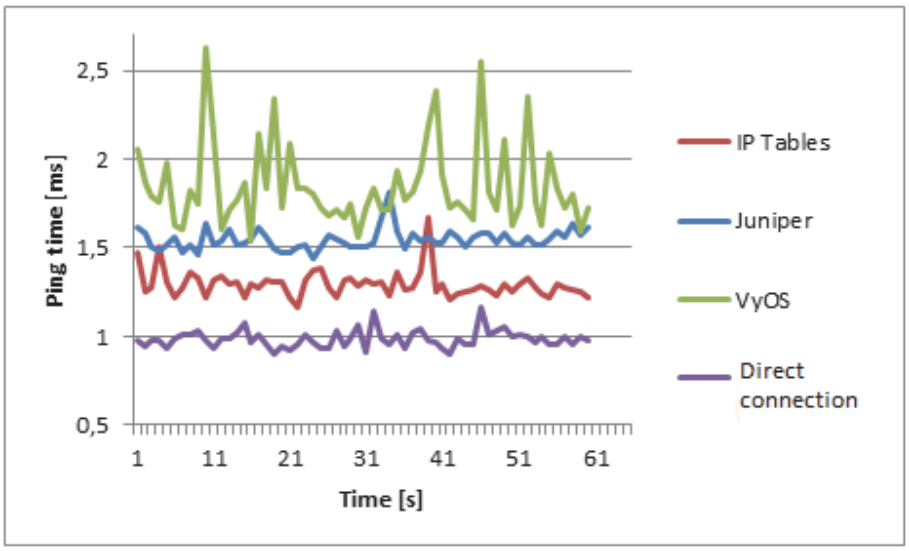


Figure 9. Ping response time for packet size 1000B

During the second part of the study the server response time was examined. The research was done using a *ping* program showing the time the packet reaches its destination. Results of these studies show which firewall introduces the greatest latency in the network. Experiment were performed in the following way. Each device was examined twice, with two packet sizes: 64 B and 1000 B. Each experiment (with each packed size) last for 60 seconds. As it may be observed in the results (Figures 8 and 9), the highest delay was observed for VyOS. Delay introduced by virtual firewall definitely

differs from the others. The smallest delays were observed for ASA and IPTables. The average response time for packet size equal to 64B was at the level of 1ms (for ASA, Juniper, IPTables) and 2.5ms for VyOS. In the latter study (ping size 1000B) the response time increased to 1.25ms for Juniper and IPTables, but we have observed a little decrease in the delay introduced by VyOS. Result for all examined firewall became proportionate.

Comparing results for all experiments it is clearly seen that virtual firewall may be pointed out as the worst solution, taking into account the performance and efficiency. It may be due to the fact that the virtual machine does not have its own built-in interfaces but uses communication interfaces of the physical machine on which it is installed. Transmitting packet through each physical port is there an additional delay. The virtual firewall could achieve better results when tested in the virtual network, which is its dedicated environment.

Finally, the ability to survive the Denial of Service attack was examined for all firewalls. DoS attack was carried out for 30 minutes with *hping3* tool. At the same time the availability of network connection to firewall was verified using *ping* requests. VyOS, ASA and Juniper remained available and operational during attacks. The CPU utilization around 100% was observed for each of them, but *ping* responses were received all the time during experiments. Unlike the others, IPTables stopped to response after 35 seconds, and hanged out after next 15 seconds. The restart of operating system and renewing of configuration was necessary in order to restore firewall functionality. It is worth to notice, that in case of software firewall the DoS attack was pointed at the operating system (Linux in this case), not at the firewall itself.

6 Conclusion

In the paper the performance and security of the hardware, software and virtual firewalls have been analyzed. The analysis was based on experiments in the prepared network environment. The considered criteria were: the throughput of the firewall, the introduced delay of network packets and the resistance to DoS attacks. A few important, practical conclusions were drawn from the results of experiments. It have been observed that the throughput of firewalls strongly depends on the size of packet transmitted over the network. Highest throughput, very close to the capacity of direct connection, was noticed for packets length equal and greater than 1 kB, for smaller packet lengths the throughput was considerably less. We may conclude that the optimal size of packet is 1 kB, while using network firewalls. Very interesting conclusion is the fact that the performance of the software based firewall was equal to the performance of hardware ones. In prepared physical environment the performance of virtual solution was lowest during all experiments.

Hardware and virtual firewalls turned out to be resistant to Denial of Service attacks. As documentation shows, they have built-in mechanisms for DoS protection. We became convinced that those mechanisms are effective. The level of security of the software firewall is, in fact, equal to the security level of the host operating system.

References

1. Ramaswamy Chandramouli, 2016, Secure Virtual Network Configuration for Virtual Machine (VM) Protection, NIST Special Publication 800-125B.
2. Wankhade A., Chatur P.N., 2014, Comparison of Firewall and Intrusion Detection System, International Journal of Computer Science and Information Technologies, Vol. 5 (1), pp. 674-678.
3. C. Sheth, R. Thakker, 2013, Performance Evaluation and Comparison of Network Firewalls under DDoS Attack, Computer Network and Information Security, Vol. 12, pp. 60-6.7
4. T. Höfler, C. Burkert and M. Telzer, 2004, "Comparative Firewall Study," Chemnitz Univeristy of Technology, Chemnitz.
5. Panchal R., 2005, *Firewalls: Hardware vs. Software*, SE 4C03.
6. Krajnik B., 2004, Firewalls with Filtering in Application Layer and Quality of Services, Engineer Diploma Thesis, Warsaw University of Technology.
7. Gouri Shankar Prajapati, Nilay Khare, 2015, A Comparative Study of Software Firewall on Windows and Linux Platform, International Journal of Computer and Technology, Vol. 14(8), pp. 5967-5978.
8. S. Patton, D. Doss and W. Yurcik, 2000, Open source versus commercial firewalls: functional comparison, Proceedings 25th Annual IEEE Conference on Local Computer Networks. LCN 2000, Tampa, FL, pp. 223-224.
9. C. Sheth and R. Thakker, 2011, Performance Evaluation and Comparative Analysis of Network Firewalls, 2011 International Conference on Devices and Communications (ICDeCom), Mesra, pp. 1-5.
10. Y. Yongxin, 2011, The comparative study on network firewalls performance, 2011 IEEE 3rd International Conference on Communication Software and Networks, Xi'an, pp. 427-430.
11. Shinder T.W., Shimonski R.J., Shinder D.L., 2003, The Best Damn Firewall Book Period" Syngress Publishing, Rockland.
12. Decusatis C., Mueller P., 2014, Virtual Firewall Performance as a Waypoint on a Software Defined Overlay Network, 2014 IEEE Intl Conf on High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Systems (HPCC,CSS,ICESS), Paris, pp. 819-822.
13. Comer D.E., 2012, Computer networks and Internets, Helion, Gliwice.
14. Krysiak K., 2005,. Sieci komputerowe. Kompendium. Helion, Gliwice.

15. NetScreen Technologies Inc., Juniper Netscreen Instaler's Guide, Version 4.
16. Gałęzowski G., 2010, Cisco ASA 5505 Podstawy konfiguracji, HAKING, Vol. 12/2010.
17. wiki.vyos.net
18. linux.die.net
19. iperf.fr/
20. blackmoreops.com, Denial-of-service Attack – DoS using hping3 with spoofed IP in Kali Linux, 2015.

DE GRUYTER
OPEN

JACSM 2017, Vol. 9, No. 1, pp. 65 - 78
10.1515/jacsm-2017-0004

chR: Dynamic Functional Constraints Checking in R

Konrad Grzanek

IT Institute, University of Social Sciences
9 Sienkiewicza St., 90-113 Lodz, Poland
kgrzanek@spoleczna.pl

Abstract

Dynamic typing of R programming language may issue some quality problems in large scale data-science and machine-learning projects for which the language is used. Following our efforts on providing gradual typing library for Clojure we come with a package chR - a library that offers functionality of run-time type-related checks in R. The solution is not only a dynamic type checker, it also helps to systematize thinking about types in the language, at the same time offering high expressivenes and full adherence to functional programming style.

Keywords: Formal software verification, software quality, dynamic type-checking, functional programming, category theory, R

1 Introduction

Writing software in dynamically typed programming languages requires as much attention with respect to types of expressions as when using a statically and strongly typed one ([2], [1]). One popular and apparently natural approach is to use gradual typing - a process of selectively adding type checks to expressions, mostly to the critical parts of computer programs. With the approach a programmer can decide where to put checks and which parts are so "obvious", that they do not have to be verified.

R programming language [5] has been growing in use in recent years, together with a growth of computer science and software engineering sub-domains to which it has been targeted: data science and statistical- (more broadly machine-) learning. Unfortunately, it lacks a decent type-checking solution. The great *assertthat* package [7] addresses a slightly different problem: putting generic run-time assertions into R codes. We need a package with the following properties:

– being deeply rooted in functional programming [4] and using notions from the category theory

– overall consistency with the dynamic and in a way Lispy nature of R programming language and adherence to functional programming style with purrr library [6]
– being as fast as possible, using checks that use as low-level elements of the R base (standard library) as possible
– expressiveness, ease of use, and extendability

Our previous work on dependent typing resulted in a *ch* library ([11], [12], [13]) for Clojure programming language ([8], [9]). Following that we decided to create a corresponding package for R. The package is called *chR* [10] and it is a subject of further analysis in this paper.

2 Essentials of the chR Library

The heart of our solution is *ch* procedure. In essence it executes a predicate (*pred*) on an argument *x*. If the predicate returns *false* value (or a value effectively) effectively equal to *false*, an error is raised. Otherwise *x* is returned. This behavior allows a greater composability and support for functional programming style. We may easily put any ch(eck) in a pipeline of data processing procedures, as will be presented further.

Procedure *ch* works also in a predicate-only mode. This mode is necessary when a ch(eck) is used as a sub-component of a larger one. Below we have the *ch* code together with an error generator (*errMessage*):

```
errMessage <- function(x) {
  r <- paste(capture.output(str(x)), collapse = "\n")
  paste0(" ch(eck) failed on\n", r)
}

#' Executes a ch(eck) of pred on x
#' @export
ch <- function(pred, x, asPred = FALSE) {
  r <- pred(x)
  if (asPred) return(r)
  if (!r)     stop(errMessage(x))
  x
}
```

Effective use of *chR* library starts with the following procedure that takes a predicate and returns a corresponding ch(eck). The returned ch(eck) takes an argument *x* and applies *ch* working by default in non-pred mode:

```
#' Returns a ch(eck) based on the pred
#' @export
chP <- function(pred) {
  function(x, asPred = FALSE) ch(pred, x, asPred)
}
```

For classes in R we have a *chInstance* ch(eck)s generator that uses a common *inherits* procedure belonging to R base:

```
#' Returns a \code{inherits(., cls)} ch(eck)
#' @export
chInstance <- function(class) chP(function(x)
  inherits(x, class))
```

An intrinsic property of an expressive language (including an embedded one) is composability. Our ch(eck)s compose. The three composition operators are logic-oriented and they reflect the most basic logical operations. We have negation:

```
#' Returns a ch(eck) that is a negation of the passed ch(eck)
#' @export
chNot <- function(c) chP(function(x)
  !c(x, asPred = TRUE))
```

Also, there is conjunction:

```
#' Returns a ch(eck) that &s all the passed ch(eck)s
#' @export
chAnd <- function(...) {
  chs  <- list(...)
  chP(function(x) {
    for (c in chs) if (!c(x, asPred = TRUE)) return (FALSE)
    TRUE
  })
}
```

and alternative:

```
#' Returns a ch(eck) that |s all the passed ch(eck)s
#' @export
chOr <- function(...) {
  chs <- list(...)
  chP(function(x) {
    for (c in chs) if (c(x, asPred = TRUE)) return (TRUE)
    FALSE
  })
}
```

In the two latter ones we assume a non-restricted number of composed ch(eck)s.

3 Fundamental Ch(eck)s

Many functional programming languages rooted in category theory ([3]), e.g. Haskell, use a unit type and unit value. Although R fully supports functional style of program-

ming, it does not unify notion of no-values. We made an arbitrary decision to treat NULL as unit value. The decision was based upon pragmatics in the technology. A corresponding ch(eck) follows:

```
#' \code{is.null} ch(eck)
#' @export
chUnit <- chP(is.null)

#' \code{!is.null} ch(eck)
#' @export
chSome <- chP(function(x) !is.null(x))
```

The *chSome* ch(eck) is an opposite to *chUnit*, as can be seen above. In R programming language we have both NULL as well as NA values. Thus, a separate ch(eck) for NAs is needed:

```
#' \code{is.na} ch(eck)
#' @export
chNA <- chAnd(chScalar, chP(is.na))
```

For two-element *Discriminated Union Types* we have the following *chEither* ch(eck):

```
#' Either ch(eck) where the left and right types are
#' expressed by checks cl and cr
#' @export
chEither <- function(cl, cr, x, asPred = FALSE)
  chOr(cl, cr)(x, asPred)
```

that used with *chUnit* forms a *Maybe* ch(eck):

```
#' Maybe ch(eck)
#' @export
chMaybe <- function(c, x, asPred = FALSE)
  chEither(chUnit, c, x, asPred)
```

Because R uses only vectorized values (and lists), we need ch(eck)s for scalars, hereby treated as one-element atoms (vectors):

```
#' Scalar \code{is.atomic} & \code{length == 1L}
#' value ch(eck)
#' @export
chScalar <- chP(function(x)
  is.atomic(x) && length(x) == 1L)
```

With the new ch(eck) we can define e.g. a ch(eck) for either a String vector of any length or a single String:

```
#' \code{is.character} ch(eck)
#' @export
chStrings <- chP(is.character)

#' \code{chScalar} & \code{chStrings} ch(eck)
#' @export
chString <- chAnd(chScalar, chStrings)
```

Another essential ch(eck) is for R functions:

```
#' \code{is.function} ch(eck)
#' @export
chFun <- chP(is.function)
```

This shortened presentation ends a section about the most common ch(ecks)s in *chR* library. For more, please read Appendix A.

4 Registry of Ch(eck)s

Additionally the *chR* library (like its ancestor *ch* for Clojure) provides a registry of ch(eck)s, that helps the programmer to understand, what kind of ch(eck)s an object or a collection of objects fulfill. The registry is an associative container that can be used to put a relation between a ch(eck) symbol (name) and the ch(eck):

```
CHSREG <- list()

#' Registeres the ch(eck) using an optional name
#' (ch(eck) name by default)
#' @export
chReg <- function(ch, name = NA) { # BEWARE: THREAD UNSAFE
  if (is.na(name)) name <- as.character(substitute(ch))
  CHSREG[[as.character(name)]] <<- as.function(ch)
  NULL
}
```

After we register selected ch(eck)s like below:

```
chReg(chUnit)
chReg(chScalar)
chReg(chString)
chReg(chStrings)
chReg(chFun)
```

we can ask the library about what kinds of ch(eck)s a given object fulfills:

```
chR::chs(1:10)

[1] "chAtomic"   "chInts"     "chNatInts"
[4] "chNumerics" "chPosInts"  "chSome"
[7] "chVector"
```

5 Supporting C++ Codes (via Rcpp)

Some of the ch(eck)s are better implemented in a low-level programming language. Thankfully R supports easy extensions in C++ written in effective library Rcpp. In *chR* the following procedure is defined to allow evaluation of predicates on vectors of any (presumably numeric) types:

```
template<typename V, typename F>
static inline bool everyInVector(const V xs,
                                 const F&& pred) {
  const int n = xs.size();
  for (int i = 0; i < n; i++)
    if (!pred(xs[i])) return false;

  return true;
}
```

The procedure is used to implement the predicates on vectors of doubles, as presented below:

```
//' Returns true iff all the xs are positive
//' @param xs vector to check
//' @return true or false
//' @export
// [[Rcpp::export]]
bool arePosDoubles(const DoubleVector xs) {
  return everyInVector(xs, [](double d)
    { return d > 0; });
}

//' Returns true iff all the xs are negative
//' @param xs vector to check
//' @return true or false
//' @export
// [[Rcpp::export]]
bool areNegDoubles(const DoubleVector xs) {
  return everyInVector(xs, [](double d)
    { return d < 0; });
}
```

```
//' Returns true iff all the xs are non-negative
//' @param xs vector to check
//' @return true or false
//' @export
// [[Rcpp::export]]
bool areNonNegDoubles(const DoubleVector xs) {
  return everyInVector(xs, [](double d)
    { return d >= 0; });
}
```

Accordingly, there are the preds for *IntegerVector*:

```
//' Returns true iff all the xs are positive
//' @param xs vector to check
//' @return true or false
//' @export
// [[Rcpp::export]]
bool arePosInts(const IntegerVector xs) {
  return everyInVector(xs, [](int n)
    { return n > 0; });
}

//' Returns true iff all the xs are negative
//' @param xs vector to check
//' @return true or false
//' @export
// [[Rcpp::export]]
bool areNegInts(const IntegerVector xs) {
  return everyInVector(xs, [](int n)
    { return n < 0; });
}

//' Returns true iff all the xs are naturals (>= 0)
//' @param xs vector to check
//' @return true or false
//' @export
// [[Rcpp::export]]
bool areNatInts(const IntegerVector xs) {
  return everyInVector(xs, [](int n)
    { return n >= 0; });
}
```

What's interesting is use of C++ lambdas in the procedures above. They are no-cost an highly expressive. Their use is possible in C++11 and above. Current Rcpp implementation supports that language standard.

6 Cases of Use in Production Setting

Our library is currently used in at least three commercial products. The usefulness of ch(eck)s can be seen in the following procedure, whose goal is to read employees' absences information in a business intelligence project:

```
readAbsences <- function(file) chDT({
  chString(file)
  absncs <- fread(file) %>%
    assertDTcolnames(ABSENCES_PROPS)

  for (p in ABSENCES_DATE_PROPS)
    set(absncs, j = p, value =
          parseDates(parse_character(absncs[[p]])))

  absncs %>% setDTcolorder(ABSENCES_PROPS)
  setkey(absncs, "Employee Number")
  absncs
})
```

The argument *file* is intended to be a String (*chString(file)* ch(eck)) and the result of the procedure is a *data.table* object (*chDT({...})* ch(eck)). Apparently, the system of ch(eck)s not only increases software correctness, but also has a positive impact on readability of codes.

References

1. Pierce B.C., 2002, *Types and Programming Languages, 1st Edition*, MIT Press, ISBN-10: 0262162091, ISBN-13: 978-0262162098
2. Pierce B.C., 2004, *Advanced Topics in Types and Programming Languages*, MIT Press, ISBN-10: 0262162288, ISBN-13: 978-0262162289
3. Awodey S., 2010, *Category Theory, Second Edition*, Oxford University Press
4. Bird R., Wadler R., 1988, *Introduction to Functional Programming*. Series in Computer Science (Editor: C.A.R. Hoare), Prentice Hall International (UK) Ltd
5. R Core Team, 2017, *R: A language and environment for statistical computing*, R Foundation for Statistical Computing, Vienna, Austria, https://www.R-project.org/
6. Henry L., Wickham H., 2017, *purrr: Functional Programming Tools*, R package version 0.2.4, https://CRAN.R-project.org/package=purrr
7. Wickham H., 2017, *assertthat: Easy Pre and Post Assertions*, R package version 0.2.0, https://CRAN.R-project.org/package=assertthat

8. Emerick Ch., Carper B., Grand Ch., 2012, *Clojure Programming*, O'Reilly Media Inc., ISBN: 978-1-449-39470-7

9. Fogus M., Houser Ch., 2014, *The Joy of Clojure*, Manning Publications; 2 edition, ISBN-10: 1617291412, ISBN-13: 978-1617291418

10. Grzanek K., 2017, *chR Github Repository*, https://github.com/kongra/chR

11. Grzanek K., 2016, *Low-Cost Dynamic Constraint Checking for the JVM*, Journal of Applied Computer Science Methods, 8(2), pp. 115-136, doi:10.1515/jacsm-2016-0008

12. Grzanek K., 2017, *ch GitHub Repository*, https://github.com/kongra/ch

13. Grzanek K., 2017, *ch Clojars Page*, https://clojars.org/kongra/ch

A Appendix: Selected Pre-defined Ch(eck)s

```
#' \code{is.logical} ch(eck)
#' @export
chBools <- chP(is.logical)

#' \code{chScalar} & \code{chBools} ch(eck)
#' @export
chBool <- chAnd(chScalar, chBools)

#' \code{is.integer} ch(eck)
#' @export
chInts <- chP(is.integer)

#' \code{chScalar} & \code{chInts} ch(eck)
#' @export
chInt <- chAnd(chScalar, chInts)

#' \code{is.double} ch(eck)
#' @export
chDoubles <- chP(is.double)

#' \code{chScalar} & \code{chDoubles} ch(eck)
#' @export
chDouble <- chAnd(chScalar, chDoubles)

#' \code{is.complex} ch(eck)
#' @export
chComplexes <- chP(is.complex)
```

```
#' \code{chScalar} & \code{chComplexes} ch(eck)
#' @export
chComplex <- chAnd(chScalar, chComplexes)

#' \code{is.numeric} ch(eck)
#' @export
chNumerics <- chP(is.numeric)

#' \code{chScalar} & \code{chNumerics} ch(eck)
#' @export
chNumeric  <- chAnd(chScalar, chNumerics)

#' \code{chInts} & > 0 ch(eck)
#' @export
chPosInts <- chAnd(chInts, chP(arePosInts))

#' \code{chInt} & > 0 ch(eck)
#' @export
chPosInt <- chAnd(chInt, chP(arePosInts))

#' \code{chDoubles} & > 0 ch(eck)
#' @export
chPosDoubles <- chAnd(chDoubles, chP(arePosDoubles))

#' \code{chDouble} & > 0 ch(eck)
#' @export
chPosDouble <- chAnd(chDouble, chP(arePosDoubles))

#' \code{chInts} & < 0 ch(eck)
#' @export
chNegInts <- chAnd(chInts, chP(areNegInts))

#' \code{chInt} & < 0 ch(eck)
#' @export
chNegInt <- chAnd(chInt, chP(areNegInts))

#' \code{chDoubles} & < 0 ch(eck)
#' @export
chNegDoubles <- chAnd(chDoubles, chP(areNegDoubles))

#' \code{chDouble} & < 0 ch(eck)
#' @export
chNegDouble <- chAnd(chDouble, chP(areNegDoubles))

#' \code{chInts} & >= 0 ch(eck)
```

```
#' @export
chNatInts <- chAnd(chInts, chP(areNatInts))

#' \code{chInt} & >= 0 ch(eck)
#' @export
chNatInt <- chAnd(chInt, chP(areNatInts))

#' \code{chDoubles} & >= 0 ch(eck)
#' @export
chNonNegDoubles <- chAnd(chDoubles, chP(areNonNegDoubles))

#' \code{chDouble} & >= 0 ch(eck)
#' @export
chNonNegDouble <- chAnd(chDouble, chP(areNonNegDoubles))

#' \code{chInt} & even? check
#' @export
chEvenInt <- chAnd(chInt, chP(function (x) x %% 2L == 0L))

#' \code{chInt} & odd? check
#' @export
chOddInt <- chAnd(chInt, chP(function (x) x %% 2L != 0L))

#' \code{is.list} ch(eck)
#' @export
chList <- chP(is.list)

#' \code{is.vector} ch(eck)
#' @export
chVector <- chP(is.vector)

#' \code{is.factor} ch(eck)
#' @export
chFactor <- chP(is.factor)

#' \code{is.data.frame} ch(eck)
#' @export
chDF <- chP(is.data.frame)

#' \code{data.table::is.data.table} ch(eck)
#' @export
chDT <- chP(data.table::is.data.table)

#' Returns a check for the data.table having exactly
#' n rows
```

```
#' @export
chDTn <- function(n) {
  chNatInt(n)
  chAnd(chDT, chP(function(dt) nrow(dt) == n))
}

#' Returns a check for the data.table having exactly
#' 0 or n rows
#' @export
chDT0n <- function(n) {
  chNatInt(n)
  chAnd(chDT, chP(function(dt) {
    nr <- nrow(dt)
    nr == 0L || nr == n
  }))
}

#' Ch(eck) for a single-row data.table
#' @export
chDT1 <- NULL

#' Ch(eck) for an empty or single-row data.table
#' @export
chDT01 <- NULL

#' \code{ggplot2::is.ggplot} ch(eck)
#' @export
chGgplot <- chP(ggplot2::is.ggplot)

#' \code{tibble::is.tibble} ch(eck)
#' @export
chTibble <- chP(tibble::is.tibble)

#' \code{is.array} ch(eck)
#' @export
chArray <- chP(is.array)

#' \code{is.atomic} ch(eck)
#' @export
chAtomic <- chP(is.atomic)

#' \code{is.recursive} ch(eck)
#' @export
chRecursive <- chP(is.recursive)
```

```
#' \code{is.object} ch(eck)
#' @export
chObject <- chP(is.object)

#' \code{is.matrix} ch(eck)
#' @export
chMatrix <- chP(is.matrix)

#' \code{is.table} ch(eck)
#' @export
chTable <- chP(is.table)

#' \code{is.environment} ch(eck)
#' @export
chEnv <- chP(is.environment)

#' \code{is.call} ch(eck)
#' @export
chCall <- chP(is.call)

#' \code{is.expression} ch(eck)
#' @export
chExpr <- chP(is.expression)

#' \code{is.symbol} ch(eck)
#' @export
chSymbol <- chP(is.symbol)

#' \code{s == ""} ch(eck) for String,
#' deliberately not chReg-ed
#' @export
chEmptyString <- chAnd(chString, chP(function(s) s == ""))

#' \code{s != ""} ch(eck) for String,
#' deliberately not chReg-ed
#' @export
chNonEmptyString <- chNot(chEmptyString)

#' Blank-ness ch(eck) for String,
#' deliberately not chReg-ed
#' @export
chBlank <- chAnd(chString, chP(function(s)
  is.na(readr::parse_character(s))))

#' Non blank-ness ch(eck) for String,
```

```
#' deliberately not chReg-ed
#' @export
chNonBlank <- chNot(chBlank)

#' \code{lubridate::is.Date} ch(eck)
#' @export
chDates <- chP(lubridate::is.Date)

#' \code{chScalar} & \code{chDates} ch(eck)
#' @export
chDate  <- chAnd(chScalar, chDates)
chReg(chDate)
```

DE GRUYTER
OPEN

JACSM 2017, Vol. 9, No. 1, pp. 79 - 88
10.1515/jacsm-2017-0005

APPLICATION OF MOLDFLOW SIMULATION IN INJECTION MOLDING OF PLASTIC TANK

Piotr Tutak

IT Instytut, Społeczna Akademia Nauk, Łódź, Polska
piotrtutak@wp.pl

Abstract

This article presents an application of moldflow simulation to optimize the injection molding process of charge air cooler plastic tank. The work shows the advantages of this kind of simulation software and information that it can provide. It also explains how big role today play simulation softwares and how they can improve product and reduce development cost.

Key words: simulation, moldflow, tank, charge air cooler, injection molding

1 Introduction

Engineering constructions during everyday use are subjected to various types of load, therefore before releasing a design for serial production at first it must be validated to ensure that it fulfills all required functions and it does not pose a threat to a user. This kind of validation usually involves performing a series of physical tests reflecting the phenomena that may occur during use in a given design. However, before building a prototype it is necessary first to check the concept itself. This can be done by experimental research which already require building expensive prototypes at this stage or to calculate rightness of the concept by means of theoretical methods: analytical or numerical. Experimental analysis in which the prototype is built is time-consuming and very expensive. This is particularly evident when various test variants of a given design concept are tested during the experimental test. Theoretical researches rely on the formulation of the corresponding mathematical description and then solution of such defined task. In case of analytical method for most cases it is difficult to get the exact solution. Such restrictions have forced the development of various numerical methods. At present, most of the calculations for structural engineering issues are done by using computer algorithms based on approximate methods, implemented in the form of programs written in different programming languages. Today

thanks to the easy algorithmization of approximation methods and the development of computer capabilities in recent years it is possible to perform computer simulations and obtain unattainable results of calculations for a given design concept and then optimize it from virtual validation level without the need of building expensive prototypes.

In the case of a charge air cooler (Figure 1) there are two types of computer simulations which are performed to validate the mechanical strength of this heat exchanger and also its tanks.

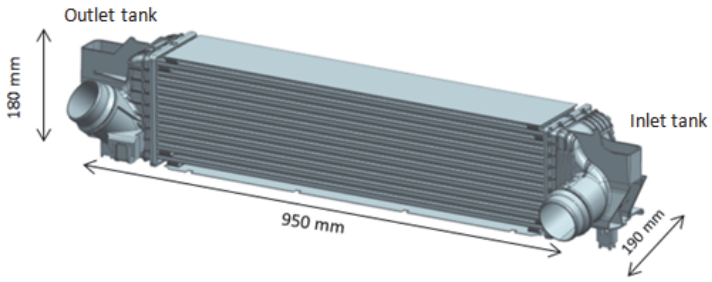
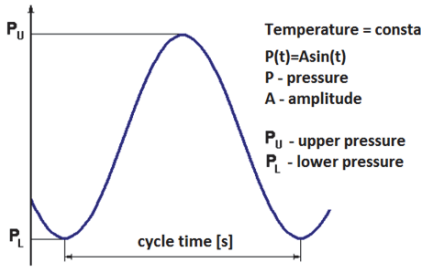
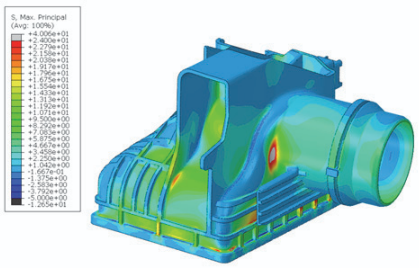


Figure 1. Charge Air cooler [own work]

One of them is a pressure cycle test simulation (Figure 2) checking resistance of its aluminum core and plastic tanks for sinusoidal changing hot air pressure in system. The pressure cycle test consists of applying cyclically changing air pressure provided to the inlet system of charge air cooler with specific temperature, frequency and through defined quantity of cycles. The test result is considered as positive if the leak level after test is in a given specification limit.





Pressure cycle test specification Pressure cycle test FEA stress result

Figure 2. Pressure cycle test [own work]

In this article we will focus on a second simulation, called moldflow which validates the design of the charge air cooler's tank from the injection mold level. Not only is the moldflow used to support and optimize the work of

plastics parts designers but also injection mold designers and injection molding technologists.

2 Moldflow simulation

Moldflow gives the opportunity of quick assessment of a just designed product at the stage of digital designing, which allows to eliminate all errors at this level. Thanks to moldflow simulation injection molding process has achieved impossible so far level of engineering. Virtual injection trials can be done without the need of building physical models. Figure 3 presents a typical injection molding machine.

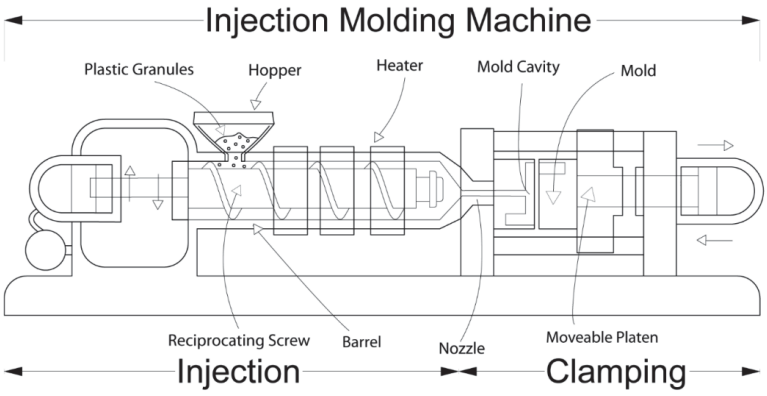


Figure 3. Injection molding machine [Wikipedia]

Table 1 shows an exemplary set of questions for which a group of engineers need to find answers to produce an optimized product. The moldflow simulation allows to find answers for these questions without the additional cost of building or modifying a prototype.

Table 1. Modlflow simulation feedback [own work]

Part designer	Mold designer	Injection technologist
- which material is optimal	- is cooling optimal	- is injection pressure proper
- is wall thickness correct	- where inlet of cooling should be placed	- what is the filling and clamp profile
- what will be deformation of part	- should channels be heated	- what are the machine settings
- cold or hot channels system	- are channels balanced	- is process stable
- will be cavity filled	- is BeCu insert necessary	- has been quality level reached
- where will be the weld lines	- where cavity degassing system should be placed	- can be cycle time reduced

Below there is a moldflow simulation of injection molding process executed to validate charge air cooler's inlet tank mold.

Figure 4 presents an object of simulation, the 3D model of charge air cooler's inlet tank, marked in green. The red rod is the injection nozzle through which material under pressure is provided to the mold. The performed simulation provides information on the influence of the assumed injection point position on the key parameters of the process.

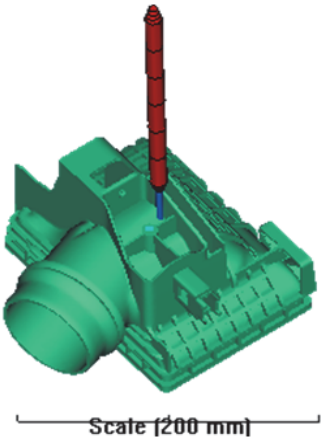


Figure 4. Inlet tank model (green) with injection nozzle (red) [own work]

Figure 5 shows the temperature of the material stream face during mold filling. Based on this result we can see that the mold filling process is correct because the material temperature is almost the same in the entire area.

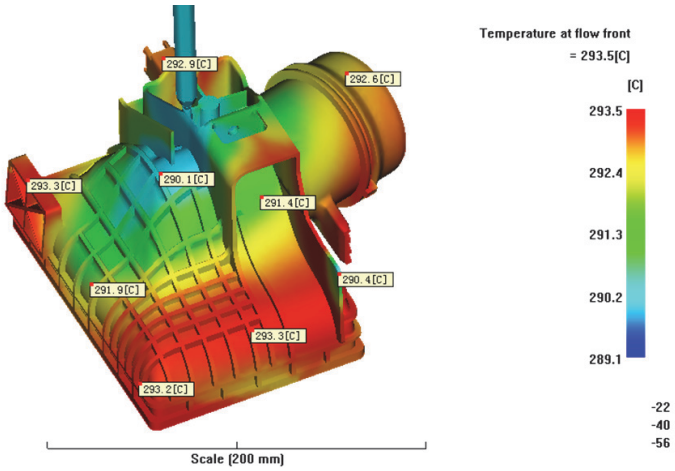


Figure 5. Temperature of material stream face [own work]

Figure 6 shows temperature distribution in the tank after material injection. Based on this we can see that temperature of the mold is heterogeneous what can result in uneven shrinkage of the material what in turn may cause significant deformation of such injected tank.

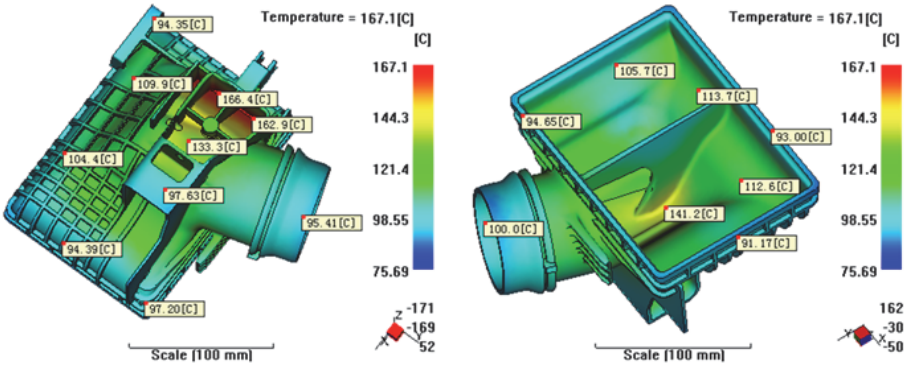


Figure 6. Temperature of tank after injection [own work]

To compensate the effect of uneven temperature distribution of mold, a cooling system needs to be implemented. In case of polyamide tanks the operating medium in cooling system is water. Mold temperature can be controlled by diameters of the cooling channels, their location in the mold as well as water pressure and its temperature, which for products made of polyamide should be approximately 80°C. Designed and analyzed cooling system is shown in Figure 7. It meets all requirements for cavity temperature (moving part of the mold on the clamp side) as well as the core temperature (stationary part of the mold on the injection side).

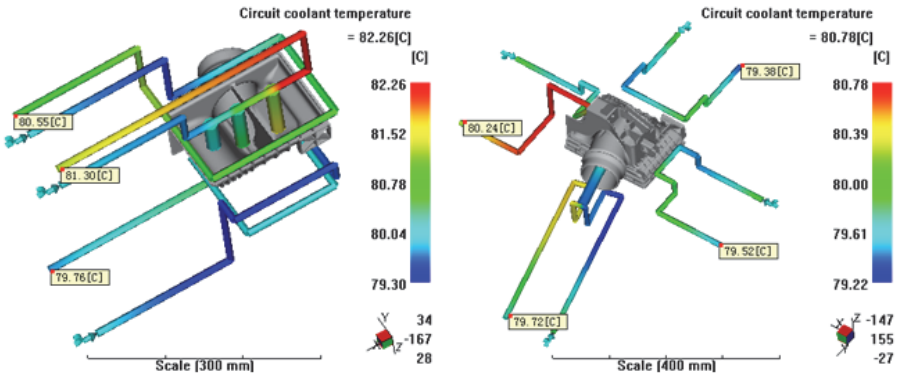


Figure 7. Designed cooling system [own work]

Moldflow analysis allows to specify the duration time of the injection process. The filling time for the analyzed tank is 2.047 seconds (Figure 8), where about 98% of the material is applied to the mold. Simulation also provides information on how the material flows and where it flows at the end, in this case it is the pipe and the tank foot area.

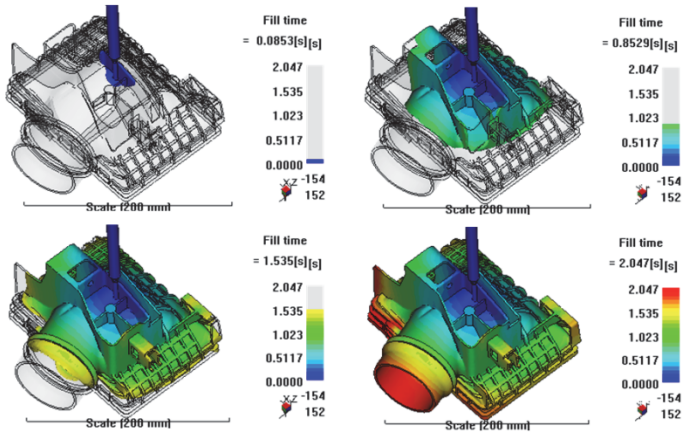


Figure 8. Mold filling phase [own work]

Based on the simulation result the required pressure value at the injection point has been defined as 30.1 MPa (Figure 8) and the duration of the phase 15.5 seconds during which 2% of the missing material is pressed. In addition, the program provides also the value of clamp force 88 tons (Figure 9) which ensures that the mold does not open during the material injection.

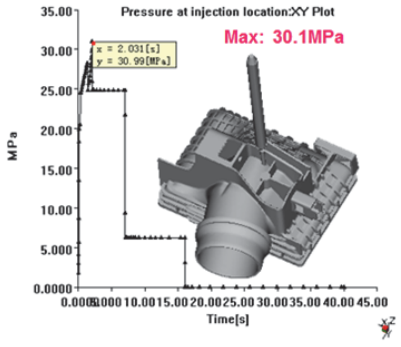


Figure 8. Required pressure in the injection point [own work]

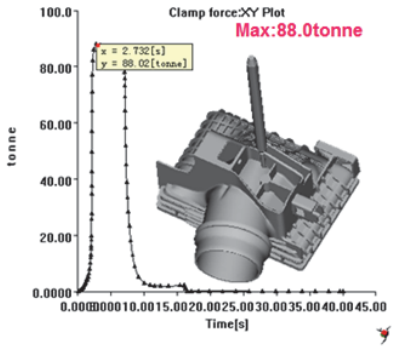


Figure 9. Required clamping force [own work]

The moldflow analysis also shows the orientation of glass fibers in the tank (Figure 10). Glass fibers in terms of strength for tank play a similar role like steel reinforcement in concrete used in construction constructions. Depending on the orientation of these fibers, there is a different shrinkage of material that affects the final shape of the product and its dimensions. When the fibers are stacked in parallel, the shrinkage in this area is about 0.5% but when fibers are perpendicular to each other, the shrinkage is about 1-1.5% to the initial length. The orientation of glass fibers is defined by the direction of the material flow during injection. If the analysis shows an aggregation of fibers in a given area (Figure 11) then such a place is characterized by worse strength properties and a tank's crack can occur in this place.

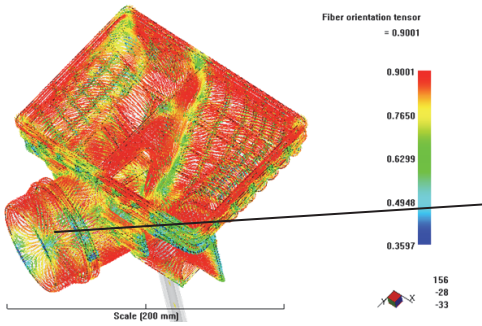


Figure 10. Orientation of glass fibers in tank [own work]

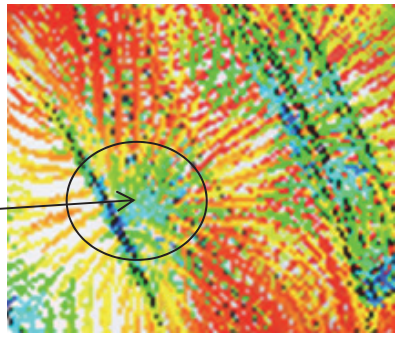


Figure 11. Aggregation of glass fibers [own work]

Another issue with negative impact on the mechanical strength of the injected parts is the fact of arising during the injection process weld lines. They are formed as a result of merging the faces of the two material streams coming from different directions having different temperature (Figures 12-13). If the injection process is well set the weld lines are not visible.

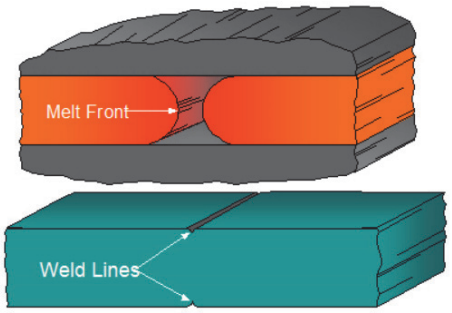


Figure 12. Weld line formation [www.mapeng.net]

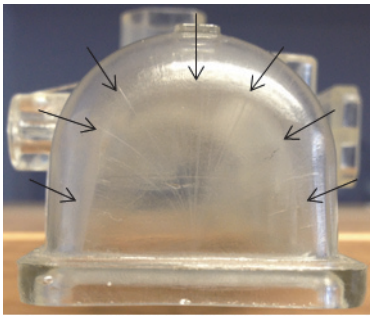


Figure 13. Weld lines [own work]

Moldflow tool for cavity filling process indicates where the potential locations of the weld lines could occur and how large they could be (Figure 14-15).

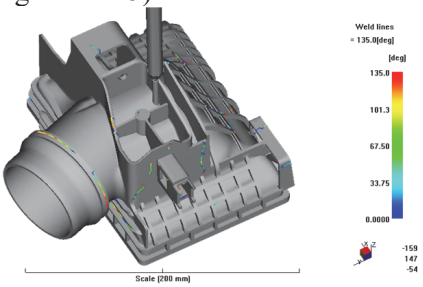


Figure 14. Weld lines location [own work]

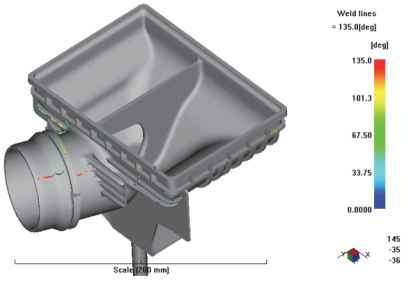


Figure 15. Weld lines location [own work]

This tool allows to verify if weld lines are located in the area where potential crack might occur. From the material durability point of view this kind of weld lines cause local weakness in the place of their occurrence. Based on weld lines location and pressure cycle simulation result decision is taken if a weld line in a given position is acceptable or not. If pressure cycle test simulation shows that in the area where the weld line is located there is also concentration of stress, the risk of crack in this area is very high. Moldflow provides also information about the deformations that may occur in the tank after injection. Figure 15 shows the final deformation of the analyzed tank in all directions.

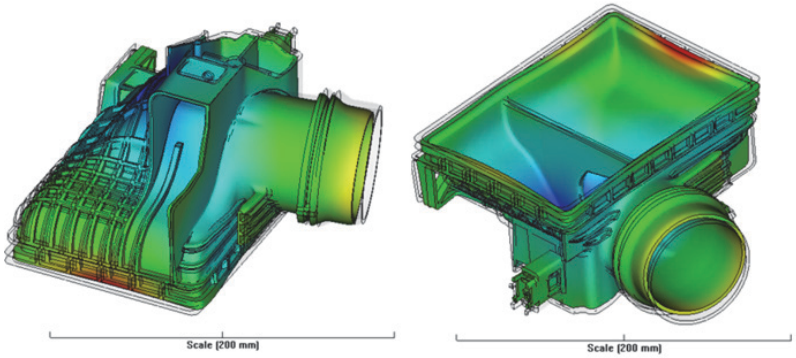


Figure 15. Tank deformation [own work]

The program also calculates the duration of each phase which is necessary to estimate the total cycle time.

3 Conclusion

This article presents an application of computer simulation optimizing the injection molding process based on the example of charge air cooler plastic tank. The presented moldflow analysis of the tank shows how important role simulation programs play today. So far, it was impossible to obtain the results of calculations for a given design concept. Now, by means of moldflow program optimization can be done from virtual validation level without the need of building expensive prototypes. It allows to create the most optimal products with minimal effort, improve quality, reduce development cost and control one of the most difficult technological processes, the injection molding process.

References

1. Aliabadi M., Brebbia C.A., *Computational methods in contact mechanics,* CMP and Elsevier Applied Science, London, 1992
2. Basmadjin D., *The Art of Modeling in Science and Engineering,* Chapman&Hall-CRC, 1999.
3. Champion E.R., *Finite Element Analysis in Manufacturing Engineering,* McGraw-Hill, New York, 1992.
4. Crandall S.H., *Engineering Analysis*, McGraw-Hill, New York, 1956
5. Huebner K.H., *The Finite Element Method for Engineers*, Wiley, New York, 1975.

6. Jenkins W.M., *Matrix and Digital Computer Methods Structural Analysis*, McGraw-Hill, New York, 1969.
7. Taler J.,*Teoria i praktyka identyfikacji procesów przepływu ciepła*, Ossolineum, Warszawa-Kraków-Wrocław, 1995.
8. Yang T. Y., *Finite element structural analysis, Prentice Hall*, New York, 1986.
9. www.autodesk.com, access 27.09.2017
10. www.robobat.pl, access 28.08.2017
11. www.pccpolska.pl, access 11.08.2017
12. www.ascented.com, access 27.08.2017
13. www.mat.net.pl, access 22.08.2017
14. www.moldmakingtechnology.com, access 22.09.2017
15. www.plasticstoday.com, access 22.08.2017