Network Intrusion Detection on Apache Spark with Machine Learning Algorithms

Elif Merve Kurt[1,2(✉)] and Yaşar Becerikli[1,3(✉)]

[1] Computer Engineering Department, Kocaeli University, Umuttepe Campus, 41380 Kocaeli, Turkey
ybecerikli@kocaeli.edu.tr

[2] Ctech Information Technology Incorporated Company, Teknopark Istanbul, Istanbul, Turkey
elif.merve.kurt@ctech.com.tr

[3] Forensic Computing Department, Ankara Group Presidency, Council of Forensic Medicine, Ankara, Turkey

Abstract. The continuous increase in internet-based services makes network traffic data larger and more complex day by day. This makes it increasingly difficult to detect network attacks, and therefore requires more efficient and faster data processing methods to ensure network security. For this purpose, many intrusion detection systems have been developed and development works are continuing.

This study; by comparing the performance of machine learning algorithms on the same network data, aims to establish a reference source for the developed intrusion detection systems. In this study; all data of KDD Cup'99 were run on Logistic Regression, Support Vector Machine, Naive Bayes and Random Forest from machine learning algorithms using Apache Spark a big data technology; and the results were analyzed comparatively.

Keywords: Network attack · Intrusion detection system · KDD Cup'99 Big data · Apache spark · Machine learning

1 Introduction

The advancement in computer technologies and the acceleration of computer networks make the development of internet based services compulsory. With the development of internet-based services, network traffic data became increasingly complex in terms of management. This makes network attacks difficult to detect and concurrently threatens network security. Although more effective methods of data processing are developed to protect against threats, attackers are constantly creating new attacks. This cyclical situation makes network attacks the focus of researchers.

Intrusion detection systems, which have been in continuous development since the 1980 are used in combination with various learning algorithms; they can be examined in two main sections: Information Based and Behavioral Based [1]. The use of pattern classification in detecting attacks is applied in various fields. Attacks can be better captured by using pattern classification, information and behavior based system logic. There are two main methods [2]: Supervised Learning and Unsupervised Learning.

In this study; KDD Cup'99 was selected as the experimental data. For faster data processing, big data technology Apache Spark was used. Information based intrusion detection system has been established using Logistic Regression, Support Vector Machine, Naive Bayes and Random Forest machine learning algorithms. The output of this study is performance analysis of intrusion detection systems developed with different machine learning algorithms. The performed comparative analysis will be a reference source for researchers. In the first section of the study, the purpose and the subject are explained; in the second part, related works are mentioned. In the following sections; the system architecture, technologies dataset and algorithms used in system, discussion, conclusion and references are included.

2 Related Works

With the increase in network traffic data, it has become increasingly difficult to detect network attacks. Resolving this problem with faster and more effective methods has become the main focus of researchers [3–7].

In literature studies vastly use Apache Spark as a big data processing tool because of its ability to rapidly analyze network traffic data and capture attack traffic [3, 5, 7]. It is emphasized by researchers that the Apache Spark is a fairly suitable tool for machine learning algorithms that requires repetition [5].

In literature, feature extraction algorithms well known for extracting qualified features have been utilized. These qualified features have been given as the dataset to the network intrusion detection system in which classification based methods are used [3, 4]. The mechanism of the system is as follows; after the information of the available network data is learned by the system, the learned information of the new incoming network data is classified as either normal or attack [6].

Overall, well known machine learning algorithms Logistic Regression [3, 4, 6], Support Vector Machine [3, 6], Random Forest [3, 6], Gradient Boosted Decision Tree Algorithms [3, 4], Naive Bayes [3] and Decision Tree Algorithms [6] are frequently used as a classification method in network intrusion detection system studies.

The KDD Cup'99 dataset which consists of data from DARPA (which is a real time dataset) was used as an experimental dataset for performance comparison verification for classification based methods [3, 6, 7]. This experimental dataset has become a standard in literature. Forest Cover Type and Internet advertising data are also among the experimental datasets used [5].

In this study, the Apache Spark Machine Learning Library was used for attack detection from big network traffic. Unlike similar studies, the algorithm extensions and data structures that came with the latest release of Apache Spark are used. Machine learning algorithms Logistic Regression, Support Vector Machine, Naive Bayes and Random Forest which are proved to be suitable for network data structure have been used in the system to enable comparability with previous studies. For the same purpose, speed and accuracy tests were performed by changing the parameters on the machine learning algorithms using the all KDD Cup'99 dataset. The algorithms are reported with parameters, giving the most optimal results in terms of time and accuracy.

In other studies, more efficient intrusion detection systems were tried to be established in terms of speed and accuracy by operations such as feature selection performed by various algorithms used on experimental dataset. The purpose of this study is not to create a more efficient and faster system, but rather to reveal the development that has occurred in the existing systems. The results clearly show how effective the development of Apache Spark tool is in detecting attacks on big network data [3].

3 System Architecture

The architecture of the developed system consists of training and prediction stages in general (see Fig. 1). The system is expected to detect the learned attacks.

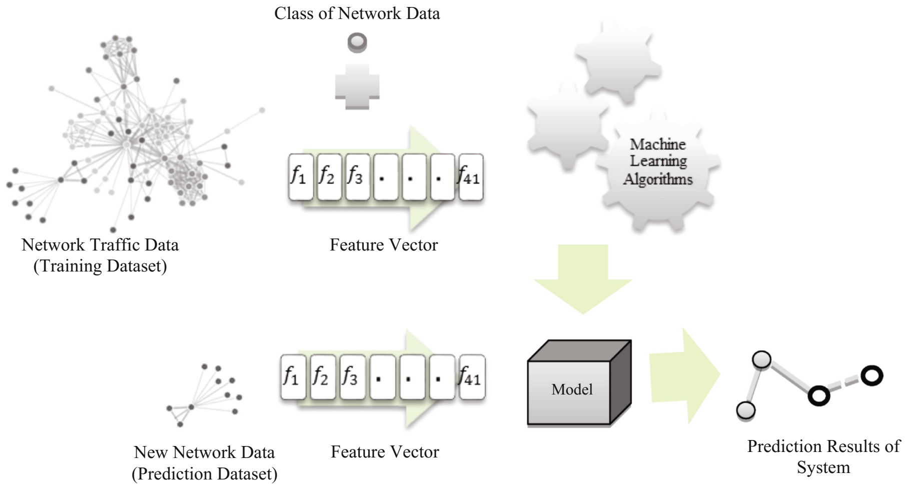
Class of Network Data
Machine Learning Algorithms
f_1 f_2 f_3 . . . f_{41}
Network Traffic Data (Training Dataset)
Feature Vector
f_1 f_2 f_3 . . . f_{41}
Model
New Network Data (Prediction Dataset)
Feature Vector
Prediction Results of System

Fig. 1. Network intrusion detection system architecture

The training dataset is applied to Logistic Regression, Support Vector Machine, Naive Bayes, Random Forest machine learning algorithms. A model for each algorithm is created. Prediction dataset without the class label is given to the generated models and the predicted class outputs of each algorithm are obtained. The accuracy of the algorithm is calculated by the prediction outputs. The processes performed while the system is being developed are: data preprocessing, training and prediction.

Data Preprocessing: The dataset is separated as the feature vector generated by extracting the features that will be effective in identifying the problem and the class label (There is no class label in the prediction dataset). Categorical features are converted to numerical features. Available dataset are separated in a ratio of 60% training dataset and 40% prediction dataset. The training data consists class label; however the prediction dataset does not.

Training Phase: is the step where the machine is being trained. At this step; it is extremely important to use enough, quality data and to select the algorithm that is suitable for the problem. The machine builds a model of itself with the knowledge obtained from the presented data and the machine learning algorithm rules determined for its use. In other words, all the knowledge that the machine knows about the type of data that will help to predict the new incoming data class is called the model. The mentioned model structure is the output of the training phase.

Prediction Phase: When the machine is compared with an unlabeled data, it begins to work on the model it creates using the algorithm rules determined for that data type and tries to predict the class of the new data with the knowledge previously learned. The output of the prediction phase is the class label for which the machine has predicted from the new data.

4 Technologies, Dataset and Algorithms Used in System

4.1 Big Data Technology: Apache Spark

Apache Spark; is an open source code platform written in Scala language, designed for big data processing, allowing parallel processing on datasets. Compared with another big data processing technology, Hadoop consists of two components MapReduce and HDFS (Hadoop Distributed File System). It is a MapReduce alternative that provides big data parallel processing. Compared to MapReduce it performs memory operations 100 times faster [8]. Apache Spark supports many programming languages. It provides an easy to use API for Scala, Python, Java, and SQL. Also, with the included machine learning library MLlib, machine learning algorithms can run on more than one machine, so that faster analysis can be performed on big data.

4.2 Dataset: KDD Cup’99

KDD Cup’99 is a dataset used in the “Third International Knowledge Discovery and Data Mining Tools” organization to create a network intrusion detector. It includes a standard set of network data that encompasses a wide range of fraudulent attacks on military network environments. The dataset was prepared by Stolfo et al. in 1999 and has become the most widely used dataset for the evaluation of anomaly detection. The KDD Cup’99 training dataset contains approximately 4.900.000 single link vectors. A link vector consist of 41 features and is labeled as normal or special attack type (the dataset contains 23 types of link types in total). It is believed that newly developed attacks can be detected with the knowledge learned from the registered attacks. According to researchers, established information is enough for the detection of unknowns.

Attacks Types Contained Within: Attack types in the KDD Cup'99 dataset can be basically examined in four main sections [9]:

- **DoS-Denial of Service Attack:** In this type of attack; the attacker creates some very busy or very busy memory resources by doing some calculations to handle logical requests; denies the users access to the machine.
- **U2R-User to Root Attack:** These attacks are a type of exploitation made by deciphering passwords or social engineering. An attacker could exploit some vulnerabilities to gain root access to the system; starts attacking by accessing a normal user account on the system.
- **R2L-Remote to Local Attack:** An attacker who exploits some security vulnerabilities to provide local access, attack to target machine by send a packet over the network to the machine like a normal user.
- **Probing Attack:** It is an attempt to gather information about the computer network for a specific purpose by passing security measures (Table 1).

Table 1. KDD Cup'99 attack types

Dos	Probe	R2L	U2R
smurf	portsweep	ftp_write	buffer_overflow
teardrop	ipsweep	guess_passwd	perl
neptune	satan	imap	loadmodule
back	nmap	multihop	rootkit
pod		phf	
land		spy	
		warezclient	
		warezmaster	

Features Contained Within: Features in the KDD Cup'99 dataset can be basically examined in three main sections [9]:

- **Basic features:** This category covers all the features available from the TCP/IP connection. Many of these features are significant delay reason in attack detection.
- **Traffic features:** These features are calculated according to a window time interval and are examined in two groups:
  - **Same host:** It only examines the connections within the last 2 s that have the same destination host with the instant connection and it computes statistics about protocol behaviors such as service.
  - **Same service:** It only examines the connections within the last 2 s that have the same service with the instant connection.
- **Content features:** Unlike most DoS and Probing attacks, R2L and U2R attacks do not have frequently repetitive pattern sequences. While DoS and Probing attacks contain many connections in very short time periods, R2L and U2R attacks are embedded in the data part of the package and normally only contain a single connection. When detecting such attack types, some features are needed to look at suspicious behavior in data fragments (such as failed logins). These are called content features (Table 2).

Table 2. KDD Cup'99 attributes

No	Attribute	No	Attribute	No	Attribute
1	duration lenght	15	lsu_attempted	29	same_srv_rate
2	protocol_type	16	lnum_root	30	diff_srv_rate
3	service	17	lnum_file_creations	31	srv_diff_host_rate
4	flag	18	lnum_shells	32	dst_host_count
5	src_bytes	19	lnum_access_files	33	dst_host_srv_count
6	dst_bytes	20	lnum_outbound_cmds	34	dst_host_same_srv_rate
7	land	21	is_host_login	35	dst_host_diff_srv_rate
8	wrong_fragment	22	is_guest_login	36	dst_host_same_src_port_rate
9	urgent	23	count	37	dst_host_srv_diff_host_rate
10	hot	24	srv_count	38	dst_host_serror_rate
11	num_failed_logins	25	serror_rate	39	dst_host_srv_serror_rate
12	logged_in	26	srv_serror_rate	40	dst_host_rerror_rate
13	lnum_compromised	27	rerror_rate	41	dst_host_srv_rerror_rate
14	lroot_shell	28	srv_rerror_rate		

4.3 Machine Learning Algorithms

Machine learning algorithms using specific parameters determine the decision rules which carry the correct decision making feature on the new data. They object to find the most suitable model that characterizes the data best, because the better the model characterizes the data, the better decision-making mechanism works.

4.3.1 Logistic Regression

Logistic regression is used for problems whose dependent variable is categorical. The purpose is to create a model of the relation between dependent and independent variables [10]. The ease of mathematical interpretation of the model makes this method eligible. The independent variable is "x", the dependent variable is "Y" and the conditional average of the dependent variable is "$\pi(x) = E(Y|x)$". Logistic regression model function is shown in Eq. (1). If the category number of Y is 2, the conditional average value range will be "$0 \leq E(Y|x) \leq 1$".

$$\pi(x) = E(Y|x) = \frac{e^{\beta_0 + \beta_1 x}}{1 + e^{\beta_0 + \beta_1 x}} \tag{1}$$

The likelihood of encountering the dependent variable is "$P(Y = 1|x) = \pi(x)$" and not encountering the dependent variable is "$P(Y = 0|x) = 1 - \pi(x)$". The conditional average must be transformed to logit to take the form of linear expression. This transformation is displayed at Eq. (2).

$$g(x) = ln\left[\frac{\pi(x)}{1 - \pi(x)}\right] = ln\left[\frac{P(Y = 1|x)}{P(Y = 0|x)}\right] = \beta_0 + \beta_1 x \tag{2}$$

When the category number of dependent variable Y is larger than 2, multinomial logistic regression will step in. Multiclass classification will be carried out by combining binary classifier. For instance if Y has {0, 1, 2} categories and Y = 0 is the reference category, two logistic model will be such that "Y = 0 against Y = 1" and, "Y = 0 against Y = 2". This method is known as one-vs-all.

4.3.2 Support Vector Machine

Support Vector Machine (SVM) is one of the most effective machine learning algorithms. It is frequently used for solving complex classification problems. Although it was initially designed to classify two classes of linearly separable data, today it is used to classify data that are composed of more than two classes which cannot be separated linearly. SVM is based on the prediction of the hyper plane, which is the decision function that will determine the data classes [11].

Assume the dataset that can be linearly separated is expressed as $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$. n is the total number of classes in the dataset. As $y_i \in \{+1, -1\}$, y_i values hold the class label of x_i values (i = 1, …, n) (see Fig. 2) [12]. There are many hyper planes that can separate the dataset. The aim here is to find a hyper plane that maximizes the distance between the closest points separating the two classes. H_0 is the optimal hyper plane. H_1 and H_2, known as support vectors, are the vectors that determine the decision boundary width. H_1 and H_2, are obtained by Eq. (3) and the optimal hyper plane H_0 is obtained by Eqs. (4) and (5) [13]. In these equations, "w" is weight vector and "b" is bias.

$$|w.x_i + b| = 1 \tag{3}$$

$$w.x_i + b \geq 1, \quad \forall x_i \in (y = +1) \tag{4}$$

$$w.x_i + b \leq 1, \quad \forall x_i \in (y = -1) \tag{5}$$

For binary classification problems using the dataset that can be linearly separated, the SVM decision function obtained as a result of a number of optimization operations is shown in Eq. (6) [13]. "λ" is the Lagrange multiplier and (*) is the expression of the optimum values.

$$f(x) = sign\left(\sum_{i=1}^{n} y_i \lambda_i^* (x.x_i) + b^*\right) \tag{6}$$

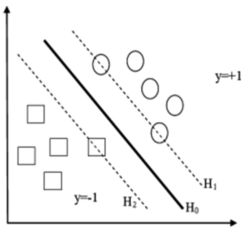

Fig. 2. Linearly Separable Dataset

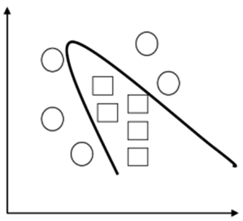
Fig. 3. Not Linearly Separable Dataset

The current dataset may not be linearly separated as in Fig. 3. The solution in such classification problems is to move the data to a larger dimensional space called the feature space. In the new space, each x variable is "$\phi(x)$" feature vector that is expressed as "$\phi(x) = \alpha_1\phi_1(x), \alpha_2\phi_2(x), \ldots, \alpha_n\phi_n(x)$". After changing the data space, the decision function is obtained by using kernel functions expressed as "$K(x, y) = \phi(x).\phi(y)$". The obtained decision function is shown in Eq. (7). Kernel functions can be linear, polynomial or radial basis [13].

$$f(x) = sign\left(\sum\nolimits_{i=1}^{n} y_i\lambda_i^* K(\mathrm{x}, x_i) + b^*\right) \tag{7}$$

In multi-class SVM problems, datasets are classified using more than one binary SVM. There are many methods recommended for multi-class SVM problems, the most important ones being one-vs-one and one-vs-all approaches [14].

4.3.3 Naive Bayes

Naive Bayes is a probability-based, widely used method. It assumes that each feature is independent. Modeling is easy and works better with big datasets. The learning model is created based on the probabilities of the dataset belonging to classes [15]. Naive Bayes produces the posterior probability P(y|x), by using P(x|y) and P(y) priori probabilities shown in Eq. (8) which is called the Bayesian rule.

$$P(y|x) = \frac{P(x|y).P(y)}{P(x)} \tag{8}$$

$$P(y|x) = P(x_1|y).P(x_2|y).P(x_3|y)\ldots P(x_n|y).P(y) \tag{9}$$

Events that are independent of the probability of occurrence with Bayes are examined. "$x_1, x_2, \ldots, x_n$" in Eq. (9) show independent events. The posterior probability is computed for each class y. The class that produces the max value is determined as the class label of data.

4.3.4 Random Forest

It is one of the widely used methods of collective classification. In this method, multiple classifiers are created instead of a single classifier. The mentioned classifiers are tree type classifiers. Instead of producing a single decision tree, the decisions of a multitude of multi variable decision trees trained with different training dataset are all combined. Therefore, it is referred to by researchers as the "collection of tree-type classifiers" [16, 17]. It surpasses other methods of collective classification in terms of both speed and accuracy. Its competence and correctness makes it a practical classifier [18]. Steps to create a Random Forest model are:

(1) With the selections made through the training dataset, a new training data is generated. T is the training dataset, T_k is the new training data generated for the tree to be created. Two-thirds of T_k will be bootstrapping samples for the decision tree. The remaining 1/3 will be used as OOB (Out of bag) data. When the tree is created, the classifier will be tested using OOB data and errors will be computed.

(2) Two parameters must be defined by the user before the algorithm starts to operate [19]: “m”, is the number of variables used in each node to perform best partitioning in the tree structure. When the value of m is chosen as the square root of the total number of variables, the best results are achieved [20]. “N” is the number of tree generations that is required. In researches, in most cases 500 trees are sufficient [23].
(3) From the new training data, a decision tree is created by random property selection method. Pruning is not carried out in trees [21, 22]. The algorithm becomes more effective classifier when not pruned compared to other tree type classifiers. The Random Forest algorithm uses the Classification and Regression (CART) algorithm to generate a decision tree [22]. The CART algorithm uses particular partitioning algorithms when deciding tree nodes. The Random Forest uses “gini” from these partitioning algorithms [17]. The Random Forest algorithm process steps are repeated until N value which is defined by the user.

At the end of each process cycle, the generated classifier is tested with the initially allocated OOB data and the OOB error is obtained. This error value shows the effect of the variables used in the classifier. While predicting a class belonging to a new incoming data in the random forest algorithm, first the new data is assigned to each node in the forest generated during the modeling phase. Then the class outputs produced by each tree are recorded. The class that receives the most votes is determined as the class of the new incoming data.

5 Discussion

The developed software was run on an Ubuntu operating system. The developments were made using the DataFrame based API of Apache Spark Machine Learning Library (Mllib).

The properties of Apache Spark which is configured with 4 GB executer and driver memory is as follows:

- Spark version: 2.2.1 (Dec 01 2017)
- Package type: Pre-built for Apache Hadoop 2.7 and later

The dataset used is the KDD Cup’99 dataset that contains in total “4898431” network link data. The data were used as training and prediction dataset by a separation ratio of 60%–40%.

Logistic Regression was applied using the one-vs-all method with default over fitting parameters. *Support Vector Machine* was implemented using default over fitting parameters and one-vs-all method with linear kernel function. *Naive Bayes* is implemented using default parameters. *Random Forest* parameters used are as folows; the number of random forest trees (NumTrees) is “300”, the homogeneity criterion (Impurity) is “gini”, the number of features to be considered for each node partition (FeatureSubsetStrategy) is “sqrt”.

The software output is shown in Table 3 where the algorithm-based comparative results can be seen. The comparative evaluation of the results obtained from the used machine learning algorithms in terms of accuracy, training time, prediction time and difference between training-prediction time are as indicated below:

- **Accuracy and performance metrics:**
  Logistic Regression > Random Forest > Support Vector Machine > Naive Bayes
- **Training time:**
  Logistic Regression > Support Vector Machine > Random Forest > Naive Bayes
- **Prediction time:**
  Random Forest > Support Vector Machine > Logistic Regression > Naive Bayes
- **Difference between training-prediction time:**
  Logistic Regression > Support Vector Machine > Random Forest > Naive Bayes

Table 3. Machine learning algorithms performance evaluations

Algorithm	Accuracy	Precision	Recall	F-Measure	Training time (h)	Prediction time (h)
Logistic Regression	0.991	0.989	0.995	0.992	4.041	0.089
Support Vector Machine	0.939	0.928	0.939	0.933	1.419	0.092
Naive Bayes	0.809	0.914	0.809	0.858	0.016	0.026
Random Forest	0.980	0.972	0.985	0.978	0.412	0.297

If the accuracy and performance metrics are to be evaluated, it is seen that the highest correct prediction rate belongs to the Logistic Regression, Random Forest and Support Vector Machine algorithms. On the other hand, in the Logistic Regression algorithm, the training time and difference between training-prediction time has the highest values compared to other algorithms. This situation raises doubts as to whether the Logistic Regression algorithm is suitable for detecting attacks on the network.

When the obtained results in the study are compared with the results of previous intrusion detection system studies, it is concluded that Apache Spark has become increasingly more efficient with newer releases in detecting attacks on big network data.

In summary, the algorithms and the data used in similar studies in literature have been reviewed and their results were compared with the results of the Apache Spark tool used in this study. For future studies, the developed software can be carried out with different parameters through different machine learning algorithms with different network datasets. And comparative studies can be carried out with the results. Especially, with the developed software, intrusion detection systems using unsupervised classification algorithms that can detect newly encountered attacks can be developed. In terms of processing speed, machines with better properties will yield faster results. By analyzing the 41 features in the dataset, it is possible to increase the processing speed by removing data with less effect on classification. The study contributes to the literature in terms of examining and summarizing the performance of the new big data methods in the latest release of Apache Spark over network data.

6 Conclusion

The implemented study is an introduction to intrusion detection systems. The obtained results show that the Apache Spark tool has become increasingly effective in detecting attacks on big network data. This study is a guide for big data researchers and provide a reference source for the developed intrusion detection systems by comparing the performance of machine learning algorithms on network data.

As a continuation of this research, the next step would be to extract the features that have less effect on the classification in the dataset, followed by performance comparisons between the performed and the previously performed classification.

References

1. Çevik, M.: Intrusion detection with pattern classification. Ph.D. thesis, Istanbul Technical University, Institute of Science and Technology (2005)
2. Becerikli, Y.: Advanced pattern recognition. Doctorate Lecture, Computer Engineering Departmant, Kocaeli University, Kocaeli, Turkey (2016)
3. Gupta, G.P., Kulariya, M.: A framework for fast and efficient cyber security network intrusion detection using apache spark. Procedia Comput. Sci. **93**(Supplement C), 824–831 (2016)
4. Siddique, K., Akhtar, Z., Lee, H.G., Kim, W., Kim, Y.: Toward bulk synchronous parallel-based machine learning techniques for anomaly detection in high-speed big data networks. Symmetry **9**(9), 197 (2017)
5. Harifi, S., Byagowi, E., Khalilian, M.: Comparative study of apache spark MLlib clustering algorithms. In: Tan, Y., Takagi, H., Shi, Y. (eds.) DMBD 2017. LNCS, vol. 10387, pp. 61–73. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-61845-6_7
6. Jeong, H.-D.J., et al.: A search for computationally efficient supervised learning algorithms of anomalous traffic. In: Barolli, L., Enokido, T. (eds.) IMIS 2017. AISC, vol. 612, pp. 590–600. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-61542-4_58
7. Oh, S.W., Kim, H.S., Lee, H.S., Kim, S.J., Park, H., You, W.: Study on the multi-modal data preprocessing for knowledge-converged super brain. In: 2016 International Conference on Information and Communication Technology Convergence (ICTC), pp. 1088–1093. IEEE (2016)
8. Lightning-fast cluster computing. https://spark.apache.org/. Accessed 14 Mar 2018
9. Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A.: A detailed analysis of the KDD CUP 99 data set. In: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, CISDA 2009, pp. 1–6. IEEE (2009)
10. Intrusion Detector Learning. http://archive.ics.uci.edu/ml/machine-learning-databases/kddcup99-mld/task.html. Accessed 08 Jan 2018
11. Vapnik, V.: The Nature of Statistical Learning Theory. Springer, New York (2013). https://doi.org/10.1007/978-1-4757-3264-1
12. Özkan, Y.: Data Mining Methods. Papatya Publishing, Istanbul (2008)

13. Osuna, E., Freund, R., Girosi, F.: Support Vector Machines: Training and Applications. Massachusetts Institute of Technology, Cambridge (1997)
14. Pöyhönen, S.: Support vector machine based classification in condition monitoring of induction motors. Helsinki University of Technology (2004)
15. Ilhan Omurca, S.: Machine learning. Master Lecture, Computer Engineering Departmant, Kocaeli University, Kocaeli, Turkey (2016)
16. Akar, Ö., Güngör, O.: Classification of multispectral images using random forest algorithm. J. Geod. Geoinf. **1**, 139–146 (2012)
17. Özdarıcı Ok, A., Akar, Ö., Güngör, O.: Classification of crops in agricultural lands using random forest classification method. In: TUFUAB 2011 VI. Technical Symposium, Antalya, Turkey (2011)
18. Gislason, P.O., Benediktsson, J.A., Sveinsson, J.R.: Random forests for land cover classification. Pattern Recogn. Lett. **27**(4), 294–300 (2006)
19. Pal, M.: Random forest classifier for remote sensing classification. Int. J. Remote Sens. **26** (1), 217–222 (2005)
20. Breiman, L.: Manual on setting up, using, and understanding random forests v3.1. Statistics Department, University of California Berkeley, CA, USA (2002)
21. Archer, K.J., Kimes, R.V.: Empirical characterization of random forest variable importance measures. Comput. Stat. Data Anal. **52**(4), 2249–2260 (2008)
22. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
23. Hasan, M.A.M., Nasser, M., Pal, B., Ahmad, S.: Support vector machine and random forest modeling for intrusion detection system (IDS). J. Intell. Learn. Syst. Appl. **06**, 45–52 (2014)