A Machine Learning Approach to Database Failure Prediction

İsmet Karakurt[1, 2], Sertay Özer[1], Taner Ulusinan[1], Murat Can Ganiz[2]
[1]ExperTeam, R&D Center
[2]Marmara University, Computer Engineering Department
Istanbul, Turkey
{ismet.karakurt, sertay.ozer, taner.ulusinan}@experteam.com.tr, murat.ganiz@marmara.edu.tr

***Abstract*—In this study, we apply machine learning algorithms to predict technical failures that can be encountered in Oracle databases and related services. In order to train machine learning algorithms, data from log files are collected hourly from Oracle database systems and labeled with two classes; normal or abnormal. We use several data science approaches to preprocess and transform the input data from raw format to the format, which can be feed to the algorithms. After the preprocessing, several different machine learning classifiers are trained and evaluated on our datasets. Our results show that warnings that lead to failures which is dubbed as abnormal events can be predicted using supervised machine learning algorithms, in particular, the Random Forest algorithm, with a relatively satisfactory Recall (75.7%) and Precision (84.9%) which is visibly higher than the other classifiers.**

Keywords— machine learning; data science; oracle database; failure prediction;

I. INTRODUCTION

Early warning systems for detecting or better yet predicting relational database management systems is quite important since many critical processes and software in wide range of businesses such as finance highly depend on these systems. Especially the companies, which provide database management and consulting solutions to many different companies on scale, have minimum tolerance for such failures. Because of the importance, we develop and evaluate a machine learning based prediction system to predict database failures by looking their log files under the light of data science techniques and artificial intelligence [1, 2]. This system is designed in collaboration with such a company: Experteam (Uzman Bilişim Danışmanlık A.Ş.) - "which is an Oracle business partner" and the main data provider for our experiments. The prediction of these failures beforehand will definitely help the companies to save time, effort and money considering the resources spent for recovering in every single failure situation. It is as of great importance keep these database systems running by early detection of failures and preventing them to happen.

The machine learning algorithms were widely used in anomaly detections and still have an important role in different domains [1, 2]. However, majority of the work in anomaly detection is focused on cyber-attacks or other anomalies in the internet traffic, registry anomalies, break-ins or malicious behavior detection upon the database [3, 4, 5, 6, 7, 8, 9, 10, 11, 12]. Predicting system based failures on relational database management systems (RDMS), focusing on a particular RDBMS seems to a rarely studied topic, if any.

Getting related data ready for machine learning algorithms includes two main challenges. First one is to collect and communicate log data to a central prediction server from several RDMSs which can be installed in different physical locations possibly behind a firewall. The second challenge is to preprocess the hourly log data which is gathered from Oracle RDBMS and related services. The data must be clean and free from noise as much as possible in order to yield a better prediction results especially in supervised machine learning algorithms. The presence of such cases, leads to a poor classification output. In real life, pure clean data is impossible to obtain. There can be undesirable outliers, mislabeled instances, missing information, irrelevant features etc. [7, 10] Because of that, many researchers also apply on working with noisy data using different approaches such as clustering methods, learned probability distributions and unsupervised machine learning algorithms that are based on outlier detection [5, 6, 7, 8, 10]. Data preprocessing stage is one of the most important stages in the data science. The data science techniques come in handy in this part and help us to make data much more useful for learning algorithms. Also, the real-world data, obtained from the ExperTeam's own databases can be considered clean in terms of class labels since in this problem domain, warnings and critical conditions that yields to system failures generated by the system. They are not manually labeled. With some extra cleaning and transformation work on the data and proper feature selection techniques, we make the data ready to apply machine learning algorithms.

Feature selection methods help reduce the dimensionality and the number of possibly noisy features and generally help to improve performance of machine learning algorithms [2]. In this regard, we firstly use SelectKBest method from sklearn (scikit-learn) library in python [14] which selects the most relevant features from the feature set. Additionally, we implement Information Gain (IG) [15] based feature selection method. We observe that feature selection improves classification results in our experiments.

Different machine learning algorithms are used in our study. The learning algorithms that we used in this project includes

Support Vector Machines [16], Naïve Bayes [17], Logistic Regression [18], Random Forest [19] and Adaptive Boosting [20]. We conduct many experiments with these classification algorithms using 10-fold cross validation approach in order to estimate their performance over unseen data and to avoid overfitting.

II. Related Work

There are many studies about intrusion/anomaly detection in the literature. This subject draws attention since the early 1990s. Since then different anomaly detection approaches have been implemented on various subjects such as cyber-attack detection [3], anomaly detection on Internet backbone traffic [4], intrusion detection in the network environment [5], anomaly detection in windows registry [6] and more. In these studies, machine learning algorithms are widely used. The usage of machine learning algorithms is preferred over the traditional signature based systems since they are considered to be laborious to build and update [3, 4, 5, 6]. Most of the studies that we came across perform their work on incomplete and noisy data because of that the clean data is hardly a case and very expensive.

Eskin et al. [13] propose a unsupervised approach to anomaly detection algorithm using unlabeled data with the assumption that the number of normal instances are significantly larger than number of abnormal instances. Shon et al. [3] propose an enhanced SVM algorithm that combines supervised SVM with one-class SVM in order to detect zero day cyber-attacks in the internet traffic. They attempt to make use of high performance of supervised algorithms and abundant amount of unlabeled data for unsupervised algorithms. Ryan et al. [4] use neural network based machine learning algorithms to detect intrusions. The learning operation is made using traces that users left on the system such as command usage. Portnoy et al. [5] on the other hand, uses clustering methods in order to detect intrusions in the network. They argue that signature based detection approaches are not efficient on detecting new types of intrusions. Also, they defend the use of unsupervised algorithms i.e. clustering methods over supervised approaches due to the difficulty of labeling a very large dataset.

Hu et al. [7] propose new SVM approach which they called as robust-SVM to deal with noisy data. In their paper, they mention about that most of the methods make an assumption about the training samples being trustable and untainted. Based on the fact that this is hardly a case, they proceed in this approach.

Different from these studies, we apply machine learning algorithms to a relatively less studied topic of predicting failures in RDBMSs. Furthermore, we apply many machine learning algorithms since the results of algorithms are mostly unpredictable and the best performing algorithms in a particular domain should be selected based on experimentation. This is also known as the no free lunch theorem in the machine learning domain. Additionally, we also consider data preprocessing and feature selection methods as important as the choice of machine learning algorithms. In this regard, we put an additional effort in this.

III. Approach

Our data consist of logs that are collected from several different servers hourly during 170 days. As a result, we have 38,184 hourly observations in total. Each row in the data corresponds to an hour of several performance metrics calculated. In particular, we have a wide range of metrics that corresponds to the columns in our dataset table. We have 261 columns in total. In addition to these 261 metrics, we also have 'target id', which identifies the Oracle database or system the data collected, 'time stamp' which show the date time of the collected data and the 'output' which basically defines the class label. These 261 features cover as many different oracle services as possible, including the core RDBMS. As a result, for a particular hour, they are sparsely populated. Detailed information about these statistics is also accessible which informs us about their target type, metric name and metric type. For instance, a feature with 'A' code might represent a value recorded from the metric 'Response' of the target 'oracle_listener' with a metric column 'tnsPing'. Also 'target_id' represents the target systems that the record took place, 'time_stamp' represents the occurrence time and 'output' represents the tag label of the record which can be either '0' or '1', which corresponds to 'Normal' or 'Abnormal' class. Under certain circumstances, databases or related services produce warnings that lead to failures. Hourly logs that produce these warnings are labeled as abnormal.

There are different targets as it's mentioned above. Target means the place of the endpoint. Since the company provides solution for other companies for different purposes, the company has different end points in order to collect data. We predict warning situation independent from the target.

The number of abnormal instances is an important parameter for this classification. In other words, very similar to the other anomaly detection problems, there is a extremely skewed class distribution in our data, in which the abnormal instances consist of only a very small percentage of the overall data. In order to be able to train supervised machine learning algorithms such as classifiers, the number of abnormal instances should be somehow increased. Otherwise, the classification algorithms may not learn to distinguish or discriminate between normal and abnormal classes and as a result, will classify every coming instance as normal which is the overwhelmingly majority class. This is one of the key problems in applying supervised algorithms for anomaly detection. In order to avoid this, we employ several basic preprocessing approaches. For instance, we decrease the number of normal instances by sampling or employ oversampling by adding more of the minority class i.e. abnormal instances. This way, we have various datasets with different distribution of class values.

The original class distribution of the data shows us that the abnormal situations are faced very rarely. In 38,184 rows we have had only 1,216 abnormal instances. The pie chart in

Figure 1 shows the situation. The percentage of warnings that lead to abnormal situations is approximately 3% in the dataset.

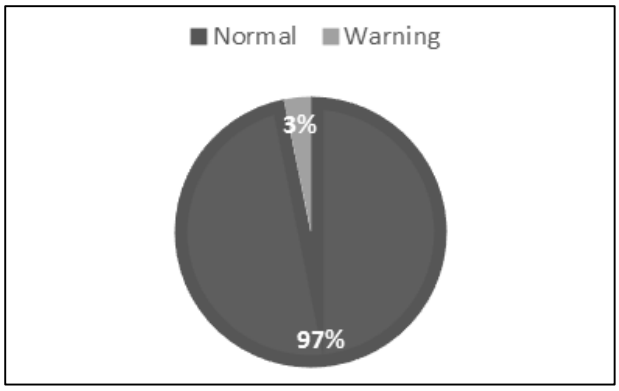

Figure 1: Class Distribution in our dataset

As mentioned, each of the 261 features in our dataset represents different target and metric combinations. The Table I shows these combinations for the top 20 features selected by Information Gain (IG).

TABLE I. TOP 20 FEATURES SELECTED BY INFORMATION GAIN

code	*Target type*	*Metric name*	*Metric column*
A150	oracle_database	instance_throughput	consistentreadgets_ps
A228	oracle_database	memory_usage	total_memory
A247	oracle_database	Service	elapsed_cs
A36	oracle_database	instance_efficiency	pxdwngrd25_ps
A111	oracle_database	dumpFull	dumpAvail
A113	oracle_database	dumpFull	dumpUsed
A46	Host	Load	activeMem
A239	oracle_database	scn_growth_statistics	current_scn
A238	oracle_database	Response	Status
A19	Host	host_storage_history	asm_unallocated
A18	Host	host_storage_history	asm_overhead
A16	Host	host_storage_history	asm_allocated
A246	oracle_database	service	cpu_cs
A13	Host	Filesystems	Available
A253	oracle_database	tbspAllocation	spaceUsed
A207	oracle_database	instance_throughput	redosize_ps
A205	oracle_database	instance_throughput	recurscalls_ps
A188	oracle_database	instance_throughput	networkbytes_ps
A187	oracle_database	instance_throughput	logreads_pt
A186	oracle_database	instance_throughput	logreads_ps

In addition to these, basic statistics for these features can be seen in Table II.

TABLE II. BASIC STATISTICS OF THE TOP 20 FEATURES

Code	*Min. value*	*Max. value*	*Average*
A150	1.47	2148008.69	108184.10
A228	9714.16	502617.65	63487.07
A247	21.75	19230854.61	136981.51
A36	9815.73	16101.76	13010.33
A111	23831706.66	451970944.00	136835775.60
A113	36409069.00	1310879800.00	402243008.27
A46	266933.33	101283017.33	34172052.31
A239	7134313056.00	1.14716288543e+13	3413992046379.36
A238	0.00	1.00	0.70
A19	6087.85	11653.64	9274.10
A18	9815.73	16101.76	13006.09
A16	39096.03	50947.86	44549.80
A246	0.00	17290144.03	76527.25
A13	60.35	755063.90	232356.17
A253	11941.133	197389.034	76399.519
A207	3.192	29685635.775	386118.253
A205	2.793	247396.551	13102.132
A188	44.457	74413713.082	1058300.781
A187	8.709	22488319.099	67124.888
A186	1.495	2149336.394	111663.743

We can see in the 'Table II' that the values corresponding to features have various numeric ranges. In this cases, normalizing the data is considered to be effective for improving the classification results [21]. Besides, putting the data on the same scale eases the operation of finding relations between features and output values considering that the output values have only '0's and '1's. Also in binary discriminative algorithms such as SVM, feature space creation from the training data is easier with the normalized data [16]. Normalization will lower the run-time of such algorithms. For these purposes, we use normalization methods found in preprocessing library of sklearn (scikit-learn) library in python. We normalize the values of our attributes between 0 and 1.

In the feature selection part, Shannon's Information Theory [15] is used in order to calculate information gain (IG) values. The features with higher IG value are selected. In this theory, $Info(D)$ is calculated for the class feature. After that, $Info_A(D)$ values are calculated for each feature in the data. IG values of the features are equal to difference between these $Info(D)$ and $Info_A(D)$ values.

For the class feature, the following calculation is made;

$$\text{Info(D)} = -\sum_{i=0}^{|C|} P(C_i)\ log_{10}\, P(C_i) \quad (1)$$

C_i : Occurrence of a class type in the data.

Apart from the class feature, the following calculation is made for each feature in the data;

$$\text{Info}_\text{A}\text{(D)} = \sum_{j=0}^{|v|} \frac{|D_j|}{|D|} \text{Info}(D_j) \quad (2)$$

v : Number of different occurrences in selected feature
D_j:The subset of the same occurrences in the selected feature
D :Size of the data

The formula below gives the information gain value for the feature 'A';

$$\text{IG}_\text{A} = \text{Info(D)} - \text{Info}_\text{A}\text{(D)} \quad (3)$$

The figure below shows the Information Gain (IG) curve of the features. While the 'y' axis shows the information gain values, the 'x' axis shows the number of features.

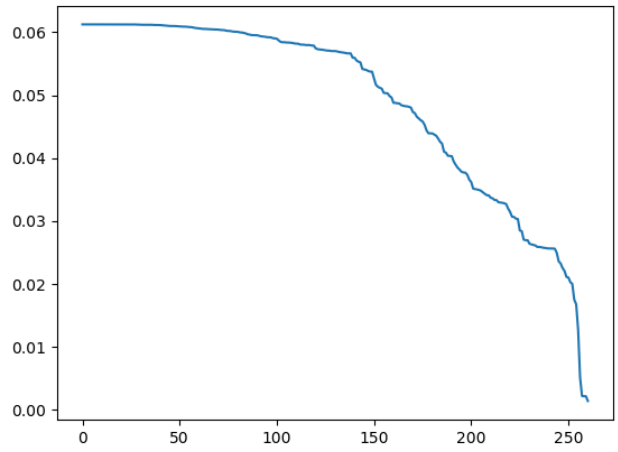

Figure 2: IG Values of the Features

Table III, shows the information gain values of the top 40 features that are sorted using their IG value.

TABLE III. IG VALUES OF THE TOP 40 FEATURES

Rank	*Code*	*IG value*
1	A150	0.0612405186943
2	A228	0.0612405186943
3	A247	0.0612405186943
...	...	...
22	A95	0.0612405186943
23	A244	0.0612247513574
...	...	…
28	A102	0.0612247513574
29	A189	0.0612089840205
30	A81	0.0612089840205
31	A88	0.0611932166836
32	A171	0.0611774493468
...	...	…
36	A158	0.0611774493468
37	A90	0.061171498064
38	A38	0.061145914673
39	A156	0.061145914673
40	A178	0.0611399633902

These values show us how well the features discriminate between two classes. IG is simply calculated using the value difference in separated classes. For instance, if a certain value for a feature has only single class occurrence or one of the classes is dramatically high in number and this situation is similar for the rest of values, then the IG value of this feature will be high. In our experiments, we chose different number of features according to IG curve in Figure 2 and examined their impact on the results.

IV. EXPERIMENT SETUP

The algorithms we used for classification are Support Vector Machines (SVM), Logistic Regression (LR), Multinomial Naive Bayes (Multinomial NB), Random Forest (RF) and Adaptive Boosting (AdaBoost) from the scikit-learn library of Python programming language.

RF and AdaBoost require several parameters. Default parameters which are programmed in scikit-learn library are used for these algorithms. The Table IV shows these parameters.

TABLE IV. ALGORITHMS AND PARAMETER SETTINGS

Algorithm	*Abbreviation*	*Parameter settings*
Multinomial Naive Bayes	Multinomial NB	-
Logistic Regression	LR	-
Support Vector Machines	SVM	-
Random Forest	RF	n_jobs=2
Adaptive Boosting	AdaBoost	min_samples_leaf=1, learning_rate=1, n_estimations=400, algorithms= "SAMME.R"

Three different experiment plans are created. First experiment plan uses whole data for both training and testing the algorithms. In the second experiment, whole data is split by using scikit-learn test_train_split [22] function as %90 testing %10 training data. In the third experiment, 10-fold cross validation (10-fold CV) is used to create training and testing data.

Additionally, five different normal and warning class distributions are used. Firstly, the abnormal instances are doubled with oversampling, and 1000, 5000, 10000, 20000 and 30000 of the normal instances that are sampled from original dataset respectively to form our dataset.

We use 'select_k_best' and Information Gain (IG) methods for feature selection. In the first two experiment settings, the top 40 features are chosen according to the IG values since IG leads to better classification results compare to k-best method. In the third experiment setting, the first breakdown point in the IG curve (Figure 2) is used to determine the number of features. The first visible breakdown point was after the 99th feature. So, the first 99 features are used in the experiments.

There were missing values (NaN values) in the dataset. We use two different approaches that are recommended in the literature in order to fill the missing values [23]. Firstly, we use an outlier value which does not exist in the data to fill the missing values. This value was '-10000' in our case. Secondly, we use the mean value of each feature to fill missing values. In our experiments, we see that the second approach works better in most of the cases.

We use several evaluation metrics in our study: Accuracy, F-measure (F1), Precision, and Recall. Accuracy is the percentage of true predictions. F1 score is the harmonic mean of the Precision and Recall. F1 is usually more useful than accuracy, especially when we have skewed class distribution. Precision tells us how many of the predictions were correct for a certain class. On the other hand, Recall tells us how many instances of a particular class are predicted correctly in the testset.

V. RESULTS

All algorithms were run for each experiment setting. You can see the evaluation results for the test on training data setting in Table V. As can be noticed from the AdaBoost results we can suspect from overfitting.

TABLE V. EVALUATION OF ALGORITHMS BY TEST ON TRAINING SET

Algorithms	*Accuracy*	*F1*	*Precision*	*Recall*
NB	%96.81	0.492	0.484	0.499
LR	%97.34	0.730	0.810	0.684
SVM	%97.20	0.725	0.788	0.686
RF	%99.86	0.988	0.998	0.978
AdaBoost	%100	1	1	1

In second experiment setting, we separate the training set and the test set by using only 10% of the data as training set and the rest as test set. Consequently, results are more realistic than the first setting which can be seen in Table VI. Although the results are pretty close, i.e. less than 1% difference, Random Forest (RF) algorithm is better than others in terms of accuracy and F1 score.

TABLE VI. 10% TRAINING SET AND 90% TEST SET

Algorithms	*Accuracy*	*F1*	*Precision*	*Recall*
NB	%96.83	0.491	0.484	0.500
LR	%97.20	0.652	0.835	0.602
SVM	%96.83	0.491	0.484	0.500
RF	%97.93	0.792	0.882	0.738
AdaBoost	%97.25	0.779	0.774	0.784

These results show the macro average F1, Precision and Recall scores of normal and warning classes. However, due to the nature of anomaly detection problems and their highly-skewed class distribution, it may be misleading to look these class averaged metrics. Actually, the important metrics to look out in our case is the F1 score or the Recall value of the abnormal class since we are trying to predict abnormal cases, in other words failures.

10-fold Cross Validation approach gives us more meaningful results in this aspect. Cross validation is a better approach to detect if a model is overfitting is prevented and much better forecaster of the real-world performance. 10-fold CV approach is applied to original data and five datasets with different sizes as described in the Experiment Setup section.

According to the third experiment's results, the best three algorithms are RF, AdaBoost and LR for the data that we study. In this experimental setup, we used different dataset because of the issue in the overall class distribution. Additionally, we reduce the number of features to 99 using IG. Another improvement over the first 2 experimental designs was using mean value for filling the empty values in the data.

Table VII shows the overall results for the original data with precision and recall values of the abnormal class.

TABLE VII. 10_FOLD CV – OVERALL RESULTS AND PRECISION AND RECALL FOR ABNORMAL CLASS

Algorithm	*Accuracy*	*F1 score*	*Precision A*	*Recall A*
RF	%98.37	0.869	0.849	0.757
AdaBoost	%98.80	0.897	0.738	0.756
LR	% 98.40	0.854	0.816	0.641
NB	% 96.99	0.557	0.794	0.071
SVM	% 98.05	0.817	0.764	0.559

Best recall (0.757) and Precision (0.849) of abnormal class is achieved by Random Forest (RF) algorithm which is evidently higher than the other algorithms.

In the remaining experiments we employ the settings which are described in the previous section to balance the class distribution.

TABLE VIII. 10_FOLD CV – ADABOOST RESULTS WITH INCREASING # OF INSTANCES IN NORMAL CLASS

Normal Class	*Algorithm*	*Accuracy*	*F1_score*	*Precision*	*Recall*
1K	AdaBoost	%100	1.0	1.0	1.0
5K	AdaBoost	%99.784	0.99755	0.99689	0.99823
10K	AdaBoost	%99.831	0.99731	0.99570	0.99895
20K	AdaBoost	%98.426	0.96102	0.94149	0.98326
30K	AdaBoost	%98.960	0.96433	0.94478	0.98629

As can be seen from Table VIII, performance metrics slightly decrease by increasing size of the normal class and henceforth skew of the class distribution. Same trend can be seen in Table IX for the LR algorithm, Table X for the NB algorithm, and Table XI for the SVM algorithm.

TABLE IX. 10_FOLD CV – LR RESULTS

	Algorithm	*Accuracy*	*F1_score*	*Precision*	*Recall*
1K	LR	%99.65	0.995	0.993	0.997
5K	LR	%98.80	0.986	0.989	0.983
10K	LR	%99.50	0.991	0.995	0.988
20K	LR	%96.91	0.919	0.923	0.915
30K	LR	%97.40	0.903	0.919	0.888

TABLE X. 10_FOLD CV – MULTINOMIAL NB RESULTS

	Algorithm	*Accuracy*	*F1_score*	*Precision*	*Recall*
1K	NB	%97.66	0.971	0.980	0.963
5K	NB	%98.92	0.987	0.991	0.983
10K	NB	%96.23	0.935	0.976	0.904
20K	NB	%92.23	0.718	0.902	0.661
30K	NB	%94.31	0.703	0.896	0.644

TABLE XI. 10_FOLD CV - SVM RESULTS

	Algorithm	*Accuracy*	*F1_score*	*Precision*	*Recall*
1K	SVM	%99.59	0.995	0.992	0.997
5K	SVM	%98.99	0.988	0.992	0.984
10K	SVM	%99.40	0.990	0.995	0.985
20K	SVM	%95.39	0.869	0.909	0.839
30K	SVM	%96.03	0.831	0.910	0.780

SVM is one of the most commonly preferred and used algorithms for the anomaly detection problems similar to the ours. But the solutions found by SVM was not as good as the ones found by RF and AdaBoost in this study. In the dataset with 30K normal instances, the recall for abnormal classes was only 0.569 for the model learned by SVM algorithm.

VI. Conclusion and Future Work

Early warning systems for detecting or predicting failures in relational database management systems (RDBMS) is important since many critical processes and software in wide range of businesses highly depend on these. The cost of these failures specifically high for certain companies such as ExperTeam (Uzman Bilişim Danışmanlık A.Ş.) which provide database management and consulting services to many different companies on scale. Early detection of these failures based on the logs of RDBMS and related system logs is of big importance. Because of this, we conduct extensive experiments for developing a machine learning based failure prediction system. The prediction of these failures beforehand will definitely help the companies to save time, effort and money considering the resources spent for recovering in every single failure situation.

We use several data science approaches to preprocess and transform the input data from raw format to the format which can be feed to the algorithms. After the preprocessing, several different machine learning classifiers are trained and evaluated extensively on our datasets. Our results show that warnings that lead to failures which is dubbed as abnormal events can be predicted using supervised machine learning algorithms, in particular Random Forest algorithm, with a relatively satisfactory Recall (75.7%) and Precision (84.9%) by Random Forest algorithm which is visibly higher than the other classifiers.

In the future work, we are planning to incorporate time series based regression and prediction approaches to the problem. Additionally, we are planning to improve the results by using temporal cumulative features.

Acknowledgments

ExperTeam is the trademark of Uzman Bilişim A.Ş. An initial part of this work has been done as a project work within the scope of introduction to data science and big data analytics class in the computer engineering department of the Marmara University with a group of four students; İsmet Karakurt, Ahmet Kaya, Soner Güzeloğlu, and Mehmethan Gür. Authors would like to thank for the anonymous reviewers for their valuable feedback.

References

[1] Chandola, V., Banerjee, A., and Kumar, V. 2009. Anomaly detection: A survey. ACM Comput. Surv. 41, 3, Article 15 (July 2009), 58 pages. DOI = 10.1145/1541880.1541882 , http://doi.acm.org/10.1145/1541880.1541882

[2] Tsai, C. F., Hsu, Y. F., Lin, C. Y., & Lin, W. Y. (2009). Intrusion detection by machine learning: A review. *Expert Systems with Applications*, *36*(10), 11994-12000.

[3] Shon, T., & Moon, J. (2007). A hybrid machine learning approach to network anomaly detection. *Information Sciences*, *177*(18), 3799-3821.

[4] de Urbina Cazenave, I. O., Köşlük, E., & Ganiz, M. C. (2011, June). An anomaly detection framework for BGP. In *Innovations in Intelligent Systems and Applications (INISTA), 2011 International Symposium on* (pp. 107-111). IEEE.

[5] Portnoy, L., Eskin, E., & Stolfo, S. (2001). Intrusion detection with unlabeled data using clustering. In *In Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001*.

[6] Heller, K. A., Svore, K. M., Keromytis, A. D., & Stolfo, S. J. (2003, November). One class support vector machines for detecting anomalous windows registry accesses. In *Proc. of the workshop on Data Mining for Computer Security* (Vol. 9).

[7] Hu, W., Liao, Y., & Vemuri, V. R. (2003, June). Robust Support Vector Machines for Anomaly Detection in Computer Security. In *ICMLA* (pp. 168-174).

[8] Hu, Y., & Panda, B. (2004, March). A data mining approach for database intrusion detection. In *Proceedings of the 2004 ACM symposium on Applied computing* (pp. 711-716). ACM.

[9] Sinclair, C., Pierce, L., & Matzner, S. (1999). An application of machine learning to network intrusion detection. In *Computer Security Applications Conference, 1999.(ACSAC'99) Proceedings. 15th Annual* (pp. 371-377). IEEE.

[10] Eskin, E. (2000). Anomaly detection over noisy data using learned probability distributions. In *In Proceedings of the International Conference on Machine Learning*.

[11] Lane, T., & Brodley, C. E. (1997, February). An application of machine learning to anomaly detection. In *Proceedings of the 20th National Information Systems Security Conference* (Vol. 377, pp. 366-380). Baltimore, USA.

[12] Kamra, A., Terzi, E., & Bertino, E. (2008). Detecting anomalous access patterns in relational databases. *The VLDB Journal—The International Journal on Very Large Data Bases*, *17*(5), 1063-1077.

[13] Eskin, E., Arnold, A., Prerau, M., Portnoy, L., & Stolfo, S. (2002). A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. Applications of data mining in computer security, 6, 77-102.

[14] Scikit-learn library in python (da:2017,June 29) Retrieved from: http://scikit-learn.org/stable/index.html

[15] Jaynes, E. T. (1957). Information theory and statistical mechanics. *Physical review*, *106*(4), 620.

[16] C.C. Chang and C.J. Lin, LIBSVM : a library for support vector machines, 2001.

[17] G. H. John and P. Langley. "Estimating continuous distributions in Bayesian classifiers," in Eleventh Conference on Uncertainty in Artificial Intelligence, San Mateo, pp. 338-345, 1995.

[18] Hosmer Jr, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). *Applied logistic regression* (Vol. 398). John Wiley & Sons.

[19] Breiman, Leo. "Random forests." *Machine learning* 45.1 (2001): 5-32.

[20] Schapire, R. E. (1999, July). A brief introduction to boosting. In *Ijcai* (Vol. 99, pp. 1401-1406).

[21] Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

[22] "train_test_split()" method from sklearn library (da:2017,June 29) Retrieved from : http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

[23] Pigott, T. D. (2009). Handling missing data. *The handbook of research synthesis and meta-analysis*, *2*, 399-416.