A Machine Learning Approach to Protect Electronic Devices from Damage Using the Concept of Outlier

Sunanda Das, Muhammad Sheikh Sadi, Md. Ahsanul Haque, Md. Milon Islam
Department of Computer Science and Engineering
Khulna University of Engineering & Technology
Khulna-9203, Bangladesh
sunanda@cse.kuet.ac.bd, sadi@cse.kuet.ac.bd, ahsanulhaque865@gmail.com, milonislam@cse.kuet.ac.bd

***Abstract*— Most of the appliances that we used in our everyday life are electronic devices, i.e. TV, Air Conditioner, Refrigerator, etc. Excessive voltage, current, temperature, etc. can harm the devices and in extreme cases, the devices can be completely damaged. We proposed a system to monitor the electrical behaviors of the devices in real-time. The system is trained with an unlabeled dataset and capable of identifying outliers. We have used a clustering technique, i.e. Density-Based Spatial Clustering of Applications with Noise (DBSCAN) to learn the label of a dataset and then apply machine learning algorithms, i.e. Support Vector Machine (SVM) and Decision Tree to predict the label of the new data. From the prediction, the system determines whether the device is operating in safe mode or not. In this work, we have achieved the accuracy of 98.61% in detecting outliers using SVM with 'rbf' kernel. Hence, if the device operates beyond safe mode, we can shut down the device.**

Keywords—Outlier; Outlier Detection; Clustering; Density-based Clustering; DBSCAN; Support Vector Machine; Decision Tree.

I. INTRODUCTION

People depend on the use of electronic devices in their everyday life. Most of the cases, the end users cannot identify the poor performance of the device and can only understand the weak performance when the device isn't working any longer. Hence, we need to find a way to protect these devices.

An outlier is an observation point or a set of observation points that have different properties and are inconsistent with other observations [1]. They are generated by any kind of disturbances in the system. Hence, outliers indicate the erroneous condition of the system. Outliers detection techniques try to find unusual patterns in the system. There are various approaches like distance-based approaches, depth-based approaches, density-based approaches, deviation-based approaches, etc. to detect outliers in the system [2].

In this work, a density-based approach, i.e. DBSCAN is used to detect outlier. DBSCAN is a hierarchical clustering method and can determine the number of clusters automatically. Moreover, DBSCAN can successfully identify arbitrary shaped cluster. This work describes an advanced system which can monitor the electrical characteristics of the device and identify disturbances like overvoltage, flicker, etc. This work considers these disturbances as the outlier and proposes an effective way to detect it. Then, supervised learning algorithms, i.e., SVM, Decision Tree are used for training a model to predict a new sample. Hence, we can decide whether the new sample data is an outlier or not.

The remaining part of the paper is organized as follows: Section II demonstrated the related works that have been done in this field. The proposed methodology with some theoretical explanations is investigated in Section III. The result is illustrated in details with some performance measures in Section IV. The conclusion of the paper is drawn in Section V.

II. RELATED WORK

For a system, outlier or anomaly may come at any time and can cause significant harm to the system. In case of protecting the system, we need to detect the outliers. In recent years, outlier detection has drawn considerable attention in the research community. For identifying outliers in feature spaces, Eskin et al. [3] proposed a system using different machine learning techniques. These techniques include one class SVM, K-nearest neighbor, and cluster-based estimation. They used these algorithms to identify the points that are in the sparse areas. Among the three algorithms, they obtained the detection rate of 98% and a false positive rate of 10% for SVM over the KDD Cup 1999 Data. Ester et al. [4] investigated the effectiveness of DBSCAN. They concluded that DBSCAN is good at detecting outlier and discovering a cluster of arbitrary shape. Their experiment on SEQUOIA 2000 benchmark data shows that DBSCAN outperforms the well-known algorithm CLARANS (Clustering Large Applications based on RANdomized Search) by a factor of at least 100 regarding efficiency.

For unsupervised anomaly detection, Kingsly Leung and Christopher Leckie [5] presented fpMAFIA, a new density-based and grid-based clustering algorithm over the 1999 KDD Cup data set. Their results indicate that their new approach fpMAFIA able to achieve a high detection rate while it suffers from a high false positive rate compared to the other methods. Bakar et al. [6] experimented different techniques like linear regression, control chart and Manhattan distance to detect outliers. Their experimental results demonstrate that the control chart technique gives reliable results in detecting outliers than linear regression. Ramaswamy et al. [7] demonstrated a profoundly efficient partition-based algorithm for discovering outliers. Their algorithm first partitions the input data set into disjoint subsets and as soon as it concludes that they do not include outliers then, the algorithm prunes entire partitions. Breunig et al. [8] talked about LOF (Local Outlier Factor). According to them, each object has its own LOF. The factor depends on how isolated the object is regarding its surrounding neighbors. Their outcome shows that the approach is very assuring in recognizing significant local outliers.

Arning et al. [9] presented a linear algorithm to detect deviation in data. Their investigation reveals that the effectiveness of the algorithm mainly depends on the

dissimilarity function. A dissimilarity function catches how different is a new data from the data items seen previously. For distinguishing outliers, Jiang et al. [10] suggested a two-phase clustering technique. In the first phase, they used a modified k-means algorithm. In the second phase, they applied an OFP (Outlier Finding Process) for finding outliers in the clusters that are generated by phase one. Hawkins et al. [11] suggested a Replicator Neural Networks (RNN) based outlier detection approach. An RNN is trained on sampled dataset so that it can predict the new data. This approach identifies cluster labels which are used to interpret the generated outliers.

III. The Proposed Methodology for Outlier Detection

The primary objective of the work is to determine the outliers of the electronic devices. Fig. 1 illustrates the detailed design of the proposed methodology. The main steps of the methodology include dataset preparation, preprocessing, DBSCAN clustering to detect the outliers and learn the label of the outlier and normal data, from that labeled dataset, predict new samples using Support Vector Machine (SVM) and Decision Tree.

A. Dataset Preparation

Ohm's law illustrates the relationship between current, voltage, and resistance. It informs us that the current i, passing through a circuit is directly proportional to the voltage v, and inversely proportional to the resistance r.

$$i = v/r \tag{1}$$

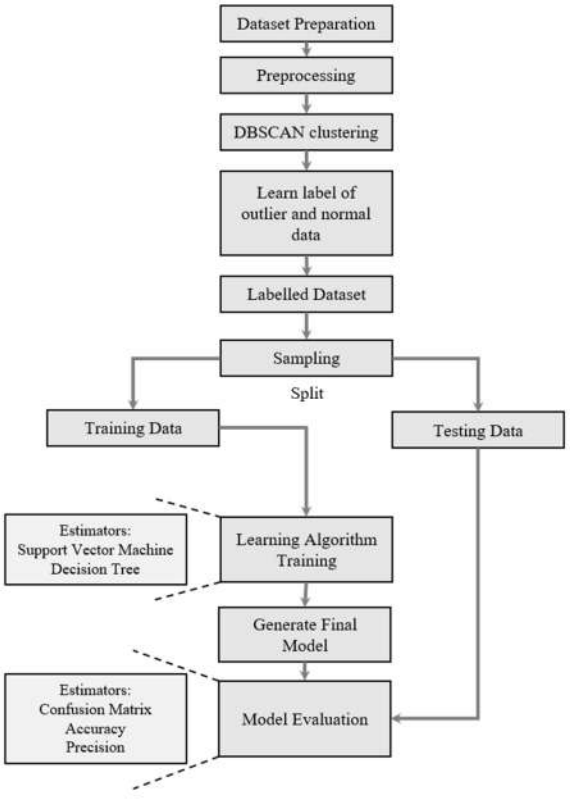


Fig. 1. Proposed methodology of the system.

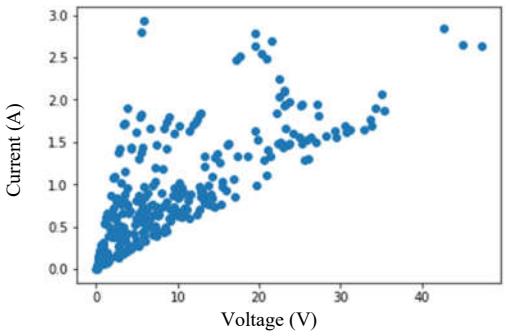


Fig. 2. The generated dataset for the system.

A synthetic dataset is prepared for the work. For preparing the dataset, we have used the (1) and randomly generate v and r and then calculate the corresponding i. The dataset contains two features, i.e., voltage (v) and current (i). Fig. 2 illustrates the generated dataset.

B. Preprocessing

The real-world data can be noisy and sometimes there are missing values in the dataset. The amount of noise and the number of missing values mainly depend on how we collect the data. A large amount of the datasets that are used to solve a machine learning problem need to be handled carefully [12]. For preprocessing, we have used z-score normalization. We get the rescaled features after using the standardization i.e. normalization. Standardizing the features is vital for our dataset as the dataset has measurements on different units. The z-score of a sample x is estimated as:

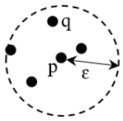
$$z = (x-\mu)/\sigma \tag{2}$$

where μ denotes the mean of the training samples, and σ denotes the standard deviation of the training samples.

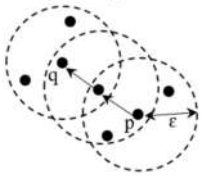
C. Clustering and Learning the label of the dataset

Clustering is mainly used for grouping objects that have the same kind of properties [13]. There are many techniques for clustering objects depending on dataset's perspective. In this work, DBSCAN is applied for clustering purpose. DBSCAN discovers neighbors, i.e. the member of the same cluster by using two parameters, i.e. MinPts and radius ε. The formation of a cluster depends on the notions of directly density-reachability, density-reachability and density-connectivity of points [14]. A cluster can be formed as the maximal set of 'density connected points' in the feature space.

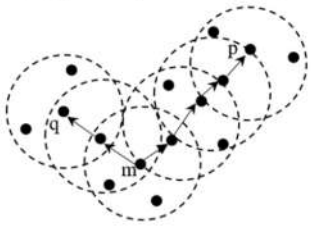
Let, consider two points p, q. From Fig. 3 it can be inferred that, q is directly density-reachable from p regarding ε if and only if q is in p's ε-neighborhood. q is density reachable from p regarding ε if there is a series of points p_1, …, p_n in such a way that $p_1 = p$, $p_n = q$ and for each i= 2, . . ., n it is true that p_i is directly density-reachable from p_{i-1} regarding ε. q is density connected from p regarding ε if and only if there is a point m such that q is density reachable from m and p is also density reachable from m [15].



(a) q is directly density reachable from p



(b) q is density reachable from p



(c) q is density connected from p

Fig. 3. The concepts of directly density reachability, density reachability and density connectedness of points.

DBSCAN defines different classes of points as followings

- Core point
- Border point
- Outlier

Let, u be another new point. u can be a core point if its neighborhood defined by radius ε contains at least or more points than the number of points defined by parameter MinPts. u can be a border point if it a member of a cluster and its ε neighborhood contain less points than MinPts but u is still density reachable by other points in that cluster. u will be an outlier if it is not a core point and at the same time not a border point. Fig. 4 represents the notion of core point, border point, and outlier. Hence, it can be realized that, all the points inside the same cluster are respectively density connected and if a point is density reachable from any member point of that cluster the point is also a member of that cluster as well.

We have used DBSCAN in our dataset. It finds the number of clusters automatically and also finds the outlier. It also returns the labels of the calculated clusters and outlier. Hence, we learn the label and use a predictive learning algorithm, i.e. SVM (Support Vector Machine) and Decision Tree to predict the label of the newly arrived data.

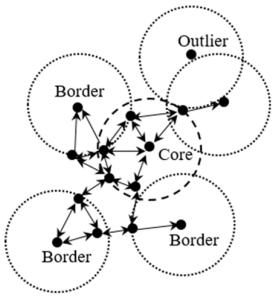


Fig. 4. The concept of core point, border point and outlier.

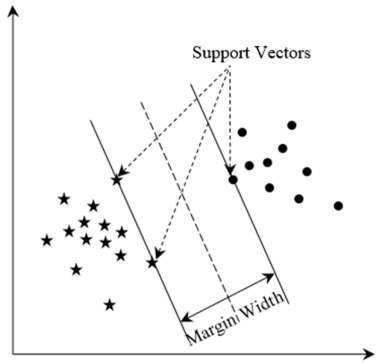


Fig. 5. Support vector points and margin width of hyperplane of SVM.

D. Prediction of outlier

Support Vector Machine (SVM) is a supervised learning algorithm in machine learning which is used for the classification problem. The main ideology behind the concept of SVM is finding the optimal hyperplane that confidently separates different classes [16].

The closest points to the hyperplane consist the support vector points. The distance between the support vector points and the hyperplane is called the margin. Fig. 5 shows support vector points and margin width of hyperplane for SVM. The support vector points play an important role for finding the hyperplane that maximizes margin width. All other points are irrelevant for finding the hyperplane.

Let, n be the number of dimensions. Thus, the equation of the hyperplane can be given as the following

$$\begin{aligned} y &= w_0 + w_1x_1 + w_2x_2 + w_3x_3 + \ldots \\ &= w_0 + \sum_{i=1}^{n} w_ix_i \\ &= w_0 + w^TX \\ &= b + w^TX \end{aligned} \tag{3}$$

where y is the outcome, b is the biased term ($b = w_0$), x_i are the attribute values. The weights w_i will be learned by the algorithm. In (3) w_i are the parameters that will determine the hyperplane [17].

Decision Tree is a very popular algorithm for classification in machine learning. To develop a tree, we need to choose features and conditions for splitting along with stopping criteria [18]. For classification, different split point is considered using a cost function. The split is selected that has the lowest cost. Gini score is considered as a cost function for classification.

$$G = sum(p_k * (1 - p_k)) \tag{4}$$

Here, p_k is the proportion of same class inputs present in a particular group. Gini score decides whether a split is good or not by estimating how mixed the response classes are in the groups formed by the split. When a group contains all the member from the same class then p_k is either 0 or 1 and $G = 0$. The performance of Decision Tree can be increased by pruning. It involves removing the branch that has less important features.

As we already generate label using DBSCAN, we train the dataset using SVM and Decision Tree. For model evaluation confusion matrix, accuracy and precision are used. Confusion matrix is used to decide the performance of a classification model. Accuracy, precision evaluates the

output quality of the classifier. Confusion matrix is formed using T_P, T_N, F_P, F_N. where,

T_P = *True Positive*
T_N = *True Negative*
F_P = *False Positive*
F_N = *False Negative*

For calculating accuracy (*Acc*) and precision (*Prec*), the following formulas are used.

$$Accuracy\ (Acc) = \frac{T_P + T_N}{T_P + T_N + F_P + F_N} \quad (5)$$

$$Precision\ (Prec) = \frac{T_P}{T_P + F_P} \quad (6)$$

IV. Result Analysis

The system is implemented in the python environment using 'scikit-learn' which is a machine learning library. The synthetic dataset that we generated by using (1), it is not possible to directly apply a supervised learning algorithm on the dataset as it has no label. We standardize the features of the dataset by using (2). Then, we apply DBSCAN on the dataset with MinPts = 3, radius ε = 0.3 and choose Euclidean metric for the DBSCAN to perform its clustering.

Fig. 6 illustrates the result of applying DBSCAN on the dataset. Here, the black points represent the outlier. All the other colored points are members of different normal classes. In addition, Fig. 6 also shows that there are three normal classes (class 0, class 1 and class 2). After performing DBSCAN on the dataset, we get the label of the normal data as well as the outlier. Now, as we know the label of the dataset, we apply supervised learning algorithms i.e. SVM and Decision Tree. Fig. 7 depicts the result of applying SVM and Decision Tree on the labeled dataset.

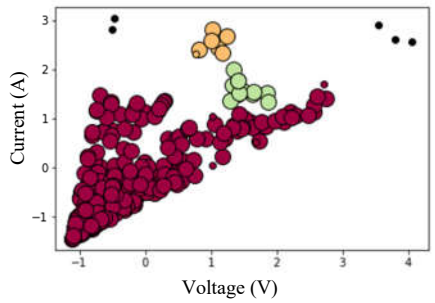
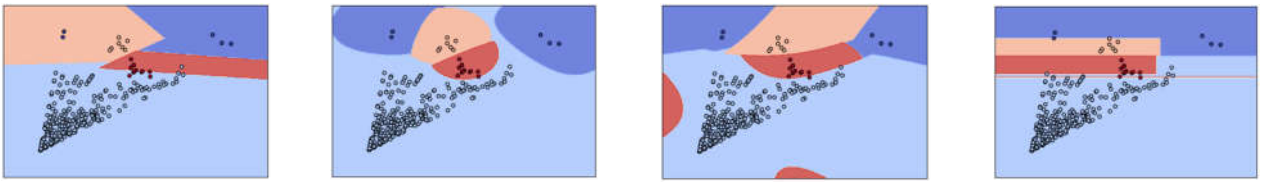


Fig. 6. Result of using DBSCAN (MinPts = 3, radius ε = 0.3, metric = 'euclidean') on the dataset.

Here, SVM and Decision Tree are used as classifier. For SVM, three different kernels i.e. linear, radial basis function ('rbf') and polynomial ('poly') are used. Different kernels are used to see which kernel can perform better. We used confusion matrix, accuracy and precision to evaluate the models. Table I and Table II represent the normalized confusion matrix for SVM and Decision Tree respectively. SVM with linear kernel predicts 50% of the outliers correctly and predict 50% as class 1.

All the other classifiers correctly predict the outlier. As our main concern is predicting the outlier, if a classifier predicts class 0 as class 1 or class 1 as class 2 it will cause no harm to the system.

The accuracy and precision are calculated using the (5) and (6) respectively. Fig. 8 illustrates the measure of accuracy and precision for training and testing phase of different classifiers. Table III contains the summary of accuracy and precision of different models. It shows that, in testing phase SVM with 'rbf' kernel has the highest accuracy and precision i.e. 98.61% and 99.60% respectively. Decision Tree classifier also performs well in the testing phase with the accuracy of 97.22% and precision of 94.59%.



(a) SVM (kernel = 'linear') (b) SVM (kernel = 'rbf', gamma = 0.5) (c) SVM (kernel = 'poly', degree = 3) (d) Decision Tree

Fig. 7. Result of applying different classifier on the labeled dataset (different colors represent different classes).

TABLE I. NORMALIZED CONFUSION MATRIX FOR SVM

	SVM (kernel = 'linear')				**SVM (kernel = 'rbf', gamma = 0.5)**				**SVM (kernel = 'poly', degree = 3)**			
	outlier	class 0	class 1	class 2	outlier	class 0	class 1	class 2	outlier	class 0	class 1	class 2
outlier	0.50	0.00	0.50	0.00	1.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00
class 0	0.00	0.98	0.00	0.02	0.00	1.00	0.00	0.00	0.00	0.98	0.00	0.02
class 1	0.00	0.00	1.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	1.00	0.00
class 2	0.00	0.80	0.00	0.20	0.00	0.20	0.00	0.80	0.00	0.20	0.00	0.80

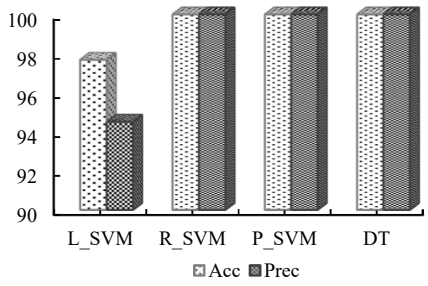
TABLE II. NORMALIZED CONFUSION MATRIX FOR DECISION TREE

	Decision Tree			
	outlier	class 0	class 1	class 2
outlier	1.00	0.00	0.00	0.00
class 0	0.00	0.98	0.00	0.02
class 1	0.00	0.00	1.00	0.00
class 2	0.00	0.00	0.20	0.80

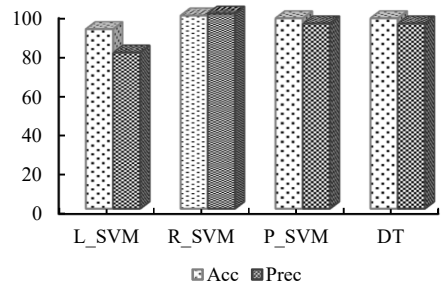
TABLE III. PERFORMANCE MEASURE INDICES OF OUTLIER DETECTION SYSTEM

Algorithm	**Kernel**	**Training Phase**		**Testing Phase**	
		Acc (%)	***Prec (%)***	***Acc (%)***	***Prec (%)***
SVM	Linear	97.68	94.51	91.67	79.71
	RBF	100	100	98.61	99.60
	Polynomial	100	100	97.22	94.59
Decision Tree	-	100	100	97.22	94.59

From Table III it is evident that if we compare the different models that have been employed on the system, SVM with 'rbf' kernel performs admirably in both training and testing phase with respect to other models.



(a) Training Phase



(b) Testing Phase

Fig. 8. Accuracy and precision for training and testing phase of different classifiers where L_SVM, R_SVM, P_SVM, DT represent SVM (kernel = 'linear'), SVM (kernel = 'rbf'), SVM (kernel = 'poly'), and Decision Tree respectively.

V. CONCLUSIONS

Protecting electronic devices from damage is very important as most of the devices are expensive. In this work, we focused on developing a system that not only detects outlier but also predicts whether a new current, voltage pair is an outlier or not. For outlier detection, a density-based clustering approach DBSCAN is used which can detect any arbitrary shaped cluster. Then, we develop models based on SVM and Decision Tree. We also provide a detailed performance measure for SVM with different kernels and Decision Tree. In the testing phase, SVM with 'rbf' kernel ends up with the highest accuracy of 98.61%. The performance shows that the proposed system will be quite helpful in protecting electronic devices.

REFERENCES

[1] V. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artificial intelligence review*, vol. 22, no. 2, pp. 85–126, 2004.

[2] H.-P. Kriegel, P. Kröger, and A. Zimek, "Outlier detection techniques," *Tutorial at KDD*, vol. 10, 2010.

[3] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo, "A geometric framework for unsupervised anomaly detection," in *Applications of data mining in computer security*, ed: Springer, pp. 77-101, 2002.

[4] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et* al., "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, pp. 226–231, 1996.

[5] K. Leung and C. Leckie, "Unsupervised anomaly detection in network intrusion detection using clusters," *in Proceedings of the Twenty-eighth Australasian conference on Computer Science-Volume 38*. Australian Computer Society, Inc., pp. 333–342, 2005.

[6] Z. A. Bakar, R. Mohemad, A. Ahmad, and M. M. Deris, "A comparative study for outlier detection techniques in data mining," in *Cybernetics and Intelligent Systems, 2006 IEEE Conference on.* IEEE, pp. 1–6, 2006.

[7] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," in *ACM Sigmod Record*, vol. 29, no. 2. ACM, pp. 427–438, 2000.

[8] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," in *ACM sigmod record*, vol. 29, no. 2. ACM, pp. 93–104, 2000.

[9] A. Arning, R. Agrawal, and P. Raghavan, "A linear method for deviation detection in large databases." in *KDD*, vol. 1141, no. 50, pp. 972–981, 1996.

[10] M.-F. Jiang, S.-S. Tseng, and C.-M. Su, "Two-phase clustering process for outliers detection," *Pattern recognition letters*, vol. 22, no. 6-7, pp. 691–700, 2001.

[11] S. Hawkins, H. He, G. Williams, and R. Baxter, "Outlier detection using replicator neural networks," in *International Conference on Data Warehousing and Knowledge Discovery*. Springer, pp. 170–180, 2002.

[12] M. R. Chmielewski and J. W. Grzymala-Busse, "Global discretization of continuous attributes as preprocessing for machine learning," 1996.

[13] P. Berkhin, "A survey of clustering data mining techniques," in *Grouping multidimensional data*. Springer, pp. 25–71, 2006.

[14] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et* al., "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, pp. 226–231, 1996.

[15] N. Schlitter, T. Falkowski *et* al., "Dengraph-ho: Density-based hierarchical community detection for explorative visual network analysis," in *Research and Development in Intelligent Systems XXVIII*. Springer, pp. 283–296, 2011.

[16] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and computing*, vol. 14, no. 3, pp. 199–222, 2004.

[17] K.-j. Kim, "Financial time series forecasting using support vector machines," *Neurocomputing*, vol. 55, no. 1-2, pp. 307–319, 2003

[18] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE transactions on systems, man, and cybernetics*, vol. 21, no. 3, pp. 660–674, 1991.