HAROKOPIO UNIVERSITY
SCHOOL OF DIGITAL TECHNOLOGY
DEPARTMENT OF INFORMATICS AND TELEMATICS
POSTGRADUATE PROGRAM INFORMATICS AND TELEMATICS
COURSE WEB TECHNOLOGIES AND APPLICATIONS

Anomaly detection on multivariate time series using ensemble techniques

Master Thesis

by

Anastasios Iliopoulos

Athens, 2023

Table of Contents

Abstract

Anomaly Detection in multivariate time series is a major problem in many fields. As the number of anomalies in real data is much smaller than the size of data it is difficult problem for categorization algorithms to solve. Also due to the variety of anomalies it is necessary to be used algorithms specially designed corresponding to the specific type of anomalies. Basic methods based on Deep Learning such as LSTM, Autoencoder, Convolutional-Autoencoder, have shown positive results in anomaly detection. The major challenge algorithms face when they applied to multivariate time series, relies on the fact that the anomaly can arise from one feature or a small subset of the total features. The detection is done based on a dynamic threshold which compared with Anomaly Score that the algorithm assign to every point. If the Anomaly Score is greater than the threshold then the point is labeled as anomalous otherwise it is labeled as normal. In order to increase effectiveness, we use ensemble techniques based on these (basic) algorithms. We apply the Majority Voting technique where five algorithms are used and the result concluded based on the majority (every vote has a weight of 1). Then we apply a variation of Majority Voting where a semi-supervised learning is done and in the final stage a Logistic Regressor is responsible to combine the basic algorithms. Also we use the idea of Feature Bagging taken from Random Forests, adjusted on time series data for the anomaly detection problem and extended using the method PCA to rotate the space so that the models can more easily identify anomalies. We call this method Feature Bagging with Rotation. Finally we will create models using the technique of Feature Bagging with Rotation and we will apply the Stacking method with a Logistic Regressor as meta-learner. The results are mixed and there is not a solution that fits all. However, results prove that ensemble techniques have better performance in many cases than the basic algorithms. More specific, the algorithm using Feature Bagging with Rotation performs better than the basic models and achieves up to 2 % better results. If we apply stacking on models that created with Feature Bagging with Rotation, with Logistic Regressor as meta-learner, performance is raised up to 10 %. The data that we used are datasets from the SKAB (Skoltech Anomaly Benchmark), which contain anomalies that have been generated to simulate different conditions.

keywords: Anomaly Detection, Ensemble, Deep Learning, Time Series, Multivariate

Chapter 1

Introduction

1.1 Machine Learning, Time Series and Anomaly Detection

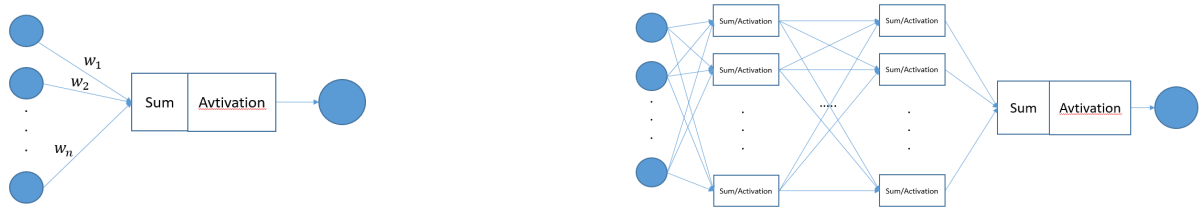
Machine learning is a subfield of statistics. Through the use of statistical methods, algorithms are trained on historical data to make predictions. Other applications that machine learning is used are classification, value prediction and clustering.

Training can take several forms and some of them characterize the nature of training. We apply supervised learning when we supply the model with labels. The model trained in order to be able to predict these labels. In supervised learning we tell the model to find any relationship exists between the input data and labels. Another form of training is unsupervised learning which is exactly the opposite from the previous form. Model never gets the labels so it is required to find the patterns between data without any knowledge of the desired outcome. Semi-supervised learning is επαινοςcalled a form that lies between the previous two. In this form a part of labels supplied to the model. Finally the reinforcement learning refers to a form in which the model interact with an environment. When the model does a desired action then we reward it, otherwise we penalize it. The goal of the model is to earn as many points of reward as it can.

While a model get historical data as input, the model compute parameters/variables named weights. So this process is called training and we say that the model is learning. When the model finish this process it means that it has computed every weight so it is capable of making predictions in new data. After training the model may be overfitted or underfitted or well generalized. A model is called overfitted if it is aligned to the training data, but when we feed it with new data it generates big errors. On the other hand, underfitted is called a model if it is unable to capture any relationship as a result it generates again big errors. The ideal scenario for a model is to be

generalized which means it never generates errors or at least they are really small and is closer to the desired output. To explain better these terms lets give an example. Consider a student that is going to give an exam. Lets assume that the student memorize every past exam but he didn't understand or learn anything. If the student face a different problem than those he memorized then the student will not pass the exam. So the student is overfitted. On the other hand if the student didn't study then he will not pass the exam again. So in this situation we say that the student is underfitted. Generally we want our models to be generalized in order to learn the relationships between data and be able to perform well in any new data.

Deep Learning is a subfield of machine learning and is the field that study the neural networks. The most simple neural network is a network that consists of some neurons in successive layers, it is call Feed Forward Neural Network and it is used in the context of supervised classification problem. Let $\mathcal{X}$ be a set of data where every element X is in the form of $X = (x_1, x_2, ..., x_n)$ and let $\mathcal{L} = \{l_1, l_2, ..., l_k\}$ the set of labels such that every element of this set $X \in \mathcal{D}$ maps to $X_i \mapsto l_j$. Then we define $z = w_1 \cdot x_1 + w_2 \cdot x_2 + ... + w_n \cdot x_n + b \Rightarrow Z = W^T \cdot X + B$. Next we apply a function (which is called activation function) h. Finally we will get the prediction $\hat{y} = h(z)$. If instead of the output add another round of computations (or many more rounds) and after that we add the output layer then these layers (rounds) called hidden layers Σχ. 1.1.



(α') Input - Output

(β') Multiple Hidden Layers

Σχήμα 1.1: Neural Networks

Time series is a term that is defined as a sequence of values indexed with time. Machine learning can used in the context of time series to predict the next value in the future. Other applications regarding time series are classification in which model tries to classify a time series or a part of it and clustering in which model tries to group the data. Another application of using machine learning on time series data is the anomaly detection. In essence, much of real data is represented through time series which is data from systems, phenomena, or organs. If during the evolution (regarding time) of an operation, phenomenon or process there is something wrong then we get an anomaly. Then it is vital to get these points (anomalous points), and automatically recognize if a value is anomaly or not.

The increase in the amount of data makes it difficult for humans to detect anomalies, and that is why we use algorithms to make the detection efficient and fast. It is therefore necessary to find models that detect ”abnormalities” during the evolution of a process. There are many problems that anomaly detection tries to solve such as detecting bank fraud or early detection of tool malfunctions. Other applications relate to the health sector, such as the detection of various diseases or viruses. Applications also exist in industry such as fault detection or the detection of incorrect operation of a machine. Also money transactions and general fraud detection lies under the anomaly detection. These are just a few of the examples we can give in relation to anomaly detection.

1.2 Time Series and Stochastic Processes

In the study of phenomena, values emerge that appear in time sequence, one after the other. This family is called time series and formally we write $X(t), t \in \mathbb{R}$. If for each moment we get more than one value we say that we have a multivariate time series and it is denoted by $(X_i)_t, \quad t \in \mathbb{R}, \quad i \in \mathbb{I}$ where $X_{i,t}$ is the i component of in time t. Otherwise, if for every moment we get a single value times series is called univariate time series. For example if we have a sensor that gives us a values for a phenomenon then the result will be a univariate time series. On the hand if we have many sensors then the result will be a multivariate time series. Len Ω be a sample space, we say that the family $X(\omega, t)$ where $t \in \mathbb{R}, \quad \omega \in \Omega$ is a stochastic process and for a moment of time $t \in \mathbb{R}$ the $X_t(\omega)$ is a random variable [1].

A time series is said to have **Trend** if its mean μ is not constant and it increases or decreases with time. Also it is said that it has **Seasonality** if there is a periodic repetition of its fluctuations and is usually observed on an annual or monthly basis. Let be a time sereis, we define autocorrelation ρ_τ for a lag τ and the correlation coefficient two elements of the time series that are τ time steps apart and is estimated from the time series as

$$\rho_\tau = \text{Corr}(x_t, x_{t+\tau}) = E[(x_t - \mu)(x_{t+\tau} - \mu)]$$

and correlation coefficient

$$r_\tau = \frac{\rho_\tau}{\rho_0} \quad , \quad -1 \leq \rho_\tau \leq 1$$

Correlation make sense if time series is stationary.

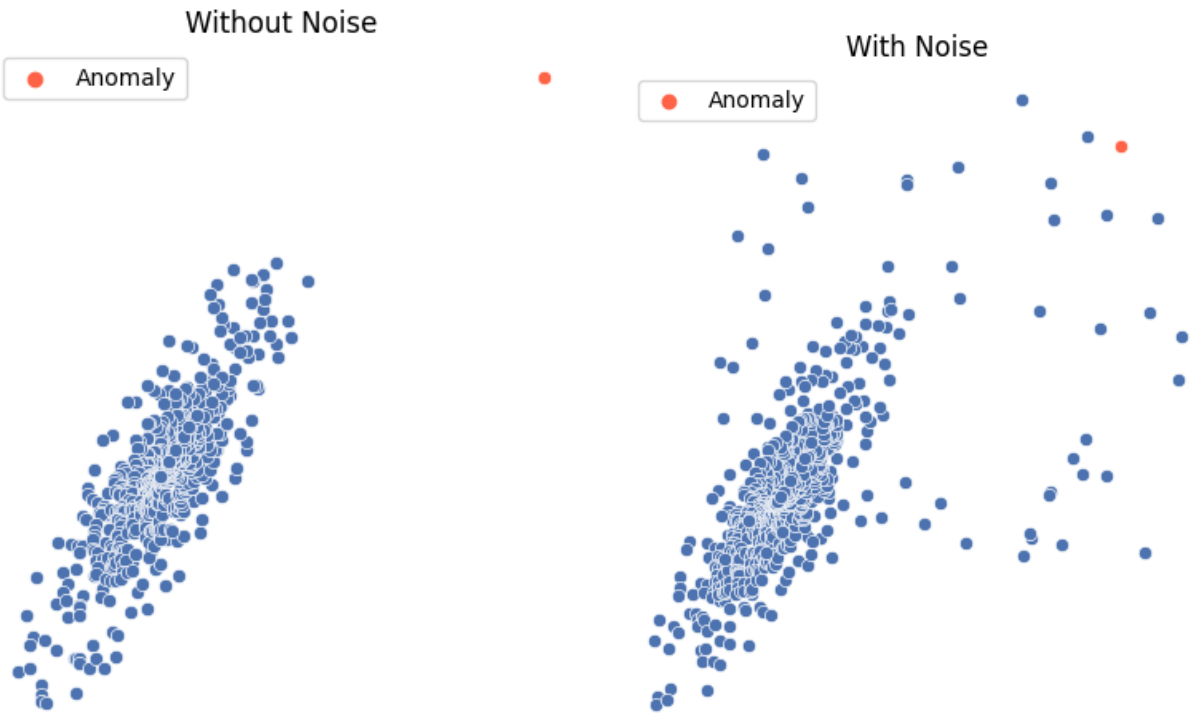
A time series that has no trend (ie has a mean of 0), no seasonality, constant variance, and constant autocorrelation is called **Στάσιμη** [2]. Noise ϵ_t is a stochastic process that is uncorrelated with time and comes from a distribution with constant mean $\mu = 0$ and finite variance. Thus noise is a stationary stochastic process.

1.3 The problem of Anomaly Detection

The goal of Machine Learning is to identify the abnormal points of a time series in the future so that the necessary actions can be taken, for example if it is a fault, the fault should be fixed, or if it is an illegal entry into a system it should be dealt with. To go deeper into this problem we must first define what an Anomaly is. The term Anomaly is not easy to define. Therefore, several attempts have been made to define this term. According to Hawkins [3] an Anomaly is ”an observation that deviates so significantly from other observations as to arouse suspicion that it was produced by some other mechanism”. Then comes the term outlier. Many do not distinguish between these two terms and consider them equivalent. Thus we have definitions such as Grubbs [4] who says that ”Outlier observations, or outliers, are those which appear to deviate significantly from the other members of the sample in which they occur”. And finally we get Aggarwal’s definition which combines the two definitions into one as follows ”Outlying points are often also referred to as non-normal, inconsistent, abnormal or anomalous, in the Data Mining and Statistics Literature”. We will consider the term Anomaly to be equivalent to the term Outlier point. Here we have to distinguish the term Anomaly from the term Novelty which refers to data that is acceptable but does not resemble the rest of our sample [5]. However, most algorithms are used to find both, so no further distinction will be made here. Finally, the main and most important problem in the definition of Anomaly is the concept of Noise. Noise are points that lie on the border between normal data and Anomalies, which could be called anomalies but are not points of interest. For this reason it is difficult to distinguish them Fig. 1.2. Some give the definitions *Strong Anomaly* and *Weak Anomaly* [6] to distinguish the two concepts.

Thus, taking into account the above, we accept the following definition [7]:

”An anomaly is an observation or sequence of observations that deviates significantly from the general distribution of the data. The total anomalies are a very small part of the total data.”



(α') Anomaly without Noise

(β') Anomaly with Noise

Σχήμα 1.2: Anomaly with and without Noise

1.4 Anomaly Types

Anomalies fall into three categories:

- **Point Anomalies** are called the points which deviate significantly from the rest of the sample. For example, a huge financial transaction that differs from the rest is called a Point Anomaly.
- **Collective Anomalies** are called the points which may be normal if we observe them individually but all together have an abnormal behavior. For example, if a customer of a bank withdraws 10 euros every 2 seconds from the bank, then these withdrawals are Collective Anomalies. 10 euros or 500 euros are normal amounts for a withdrawal but a series of such withdrawals is abnormal behavior.
- **Context Anomalies** or Context-dependent abnormalities are those points which are normal in a particular context but are reported as abnormal in another. For example the temperature $150°C$ is a normal temperature for someone to cook their food in an oven but it is an abnormal value for the weather of a country.

1.5 Anomaly Detection

Having defined what is an Anomaly, we can talk about Anomaly Detection. We define a function φ such that:

$$\varphi\colon \mathbb{R}^n \to \mathbb{R}$$
$$\varphi(x) \mapsto \gamma$$

where γ is Anomaly Score, $x \in X \subseteq \mathbb{R}^n$ and X is the data set. Anomaly Score is defined as the measure to describe how much anomalous is a point, or how much this point deviates from the rest data set. We can transform it to a measure that belongs to interval $[0, 1]$ and then we can say that it is the probability of a point to be anomalous.

To finally arrive at whether a point is an anomaly or not we define a threshold. A threshold is called a $\delta \in \mathbb{R}$, where all the points with an anomaly score below δ will be the normal points while those with an anomaly score above δ will be the anomalous points. Thus we define the binary anomaly detection method as:

$$\varphi_{\text{binary}}\colon \mathbb{R}^n \to \{Normal, Anomaly\} \quad \text{ή} \quad \{0, 1\}$$
$$\varphi_{\text{binary}}(x) \mapsto \begin{cases} Anomaly \quad \text{ή} \quad 1, \quad \text{if} \quad \varphi(x) > \delta \\ Normal \quad \text{ή} \quad 0, \quad \text{otherwise} \end{cases}$$

Anomaly detection is not an ordinary two-class classification process. Naturally anomalies are less than 1% which makes anomaly detection very different from two class classification. The difference lies in the fact that even if we say that all the points in a data set are normal points then the success of this model will be 99%. But in reality the model will perform very poorly because it did not catch a single anomaly. For this reason we should tackle the problem with two well-known metrics: the F1 score and AUC (area under curve).

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{F}_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

and

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad \text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

AUC is the area under the plot obtained by plotting on two vertical x,y axes, FPR on the x-axis and TPR on the y-axis. In essence, it shows how well the model can distinguish one class from another.

Anomaly detection is not handled in the same way for all problems. Although we mentioned the difference in relation to two-class classification problems, we should say that anomaly detection problems (and thus their approaches) differ from each other depending on the type of anomalies they contain. This is clearly seen in the context of the time series we study in this thesis. For example, let's assume that we have a machine and we record in order among others the following temperatures in degrees Celsius: 35,34,34,35,36,37,36,90,92,92,93,92,94. If we approached the problem as if we had Point Anomalies then we probably wouldn't find any Anomalies because these temperatures are very normal for an engine. But if we take into account the time dependence of the temperatures then we observe an abnormal sudden rise from 36 degrees to 90. In the time series data we usually find Collective Anomalies or Anomalies that depend on the context (context anomalies).

Chapter 2

Background

Anomaly detection is a subject that has many difficulties and is an evolving field. In the literature we will notice that the methods used can be categorized into 3 categories:

- Statistical methods
- Machine Learning methods
- Deep Learning methods

Statistical methods assume that anomalies generated from a statistical model. In contrast, in machine learning methods, the anomaly generation mechanism is considered as a black box and anomalies are detected based on the data. Finally, the third category concerns techniques based exclusively on neural networks, which are often considered part of Machine Learning, which is why there is often no separation between the second and third categories. Several methods have been presented by Markou, M. & Singh, S. [5, 8] and by Chandola we have a very valuable research [9, 10] who mentions algorithms of all three categories.

2.1 Statistical methods

- **Autoregressive Model:**

  This model is considered linear where the value X_t at time t is based on the (finite) set of size p of the previous values of the (stochastic) process and an error ϵ, that is:

$$X_t = \sum_{i=1}^{p} a_i X_{t-i} + c + \epsilon_t$$

p is also called a window of length p and the model is called an Autoregressive Model of order p or simply $AR(p)$. The ”error” values are considered uncorrelated and have mean $\mu = 0$ and constant variance σ.

The coefficients $a_1, ..., a_p$ and c can be approximated during training by solving the corresponding linear equations using the least square method. ϵ , which represents the anomaly score, can then be calculated as the difference between the ”predicted” value and the actual value of X at time t . The model assumes that the stochastic process is stationary, so if we do not have a stationary process then it should be transformed into one.

- **Moving Average Model:**

  While the previous model considers a X_t value as a linear combination of the previous p values, the Moving Average model considers this value as a linear combination of the previous q forecasts' errors, so:

$$X_t = \sum_{i=1}^{q} a_i \epsilon_{t-i} + \mu + \epsilon_t$$

  where μ is the mean value of X and the coefficients $a_1, ..., a_q$ are calculated during training as before. q is the size of the window and for this model (of order q) we write $MA(q)$. The difference with the previous model is that it considered the previous values from X_t known, however in this model are considered unknown and predicted at each step thus producing the errors $\epsilon_t, ...\epsilon_{t-q}$.

- **Autoregressive Moving Average Model:**

  This model is a combination of the previous two and is often used on univariate time series data. denoted by $ARMA(p, q)$ and defined as:

$$X_t = \sum_{i=1}^{p} a_i X_{t-i} + \sum_{i=1}^{q} b_i \epsilon_{t-i} + \epsilon_t$$

  $X(t)_{t\in\mathbb{R}}$ is an $ARMA(p, q)$ process if it is stationary. Actually this model uses fewer variables than the previous models but the main difficulty is in choosing the appropriate p and q. The larger these values of these two parameters are, the more likely the model will be overfitted. This will result in many False Negatives, i.e. the model will not be able to recognize that something is an anomaly and decide that it is a normal point. On the other hand, if we

choose very small values, we run the risk of getting many False Positives, i.e. the model will characterize several points as anomalies which in reality are not. In any case the model is not reliable.

- **ARIMA Model:**

  The major challenge faced by previous algorithms is that many processes are not stationary. This model addresses this problem by generalizing the Autoregressive Moving Average model by adding an additional parameter d which defines how many times we will take the differences for a time series. That is:

$$d = 0, \quad X'_t = X_t$$
$$d = 1, \quad X'_t = X_t - X_{t-1}$$
$$d = 2, \quad X'_t = (X_t - X_{t-1}) - (X_{t-1} - X_{t-2})$$

  This transformation allows us to either remove the trend the time series may have, or to remove seasonality. It is applied as many times as we want until the time series is where we want it. Finally, the anomalous points are detected as in the previous algorithms (according to the error we get from the difference between the predicted value and the actual one).

- **Exponential Smoothing Model:**

  Unlike previous models that are linear, this model uses a non-linear approach. This model is defined as follows:

$$X_{t+1} = aX_t + a(1-a)X_{t-1} + a(1-a)^2X_{t-2}, ..., +a(1-a)^N X_{t-N}$$

  where $a \in [0, 1]$, so the prediction X_{t+1} is a linear combination of the prior values with exponential weights. The parameter a is called the weight reduction rate, and the smaller it is, the more weight is given to terms that are further away (regrading time). This model assumes that X is stationary.

  Variations exist to deal with non-stationarity by introducing additional parameters.

  double exponential smoothing

  To deal with the problem of the trend in stochastic processes, the double exponential smoothing

model is introduced, which is a generalization of the previous model. For the definition we have:

$$S_0 = X_0$$
$$B_0 = X_1 - X_0$$

και

$$S_t = aX_t + (1-a)(S_{t-1} + B_{t-1})$$
$$B_t = \beta(S_t - S_{t-1}) + (1-\beta)B_{t-1}$$

where $0 \leq a \leq 1$ and $0 \leq \beta \leq 1$

and finally the model **triple exponential smoothing** is introduced to tackle the problem of seasonality [11]

2.2 Machine Learning methods

Οι μέθοδοι μηχανικής μάθησης προσπαθούν να εντοπίσουν τις ανωμαλίες χωρίς να θεωρούν ότι προέρχονται από κάποιο μοντέλο. Βασίζονται στο γεγονός ότι δεν χρειάζεται να ξέρουν ο τρόπος που παράχθηκαν τα δεδομένα. Έτσι διαφοροποιούνται πολύ από τα προηγούμενα μοντέλα.

- **K-Means Clustering**

  Ο K-means μπορεί είναι ένα πολύ γνωστός αλγόριθμος στα πλαίσια της μη-επιβλεπόμενης εκπαίδευσης, ωστόσο μπορεί να χρησιμοποιηθεί και στα πλαίσια της ανίχνευσης ανωμαλιών. Έστω μία διαδικασία (X_N) αρχικά ορίζουμε ένα παράθυρο μήκους w. Στη συνέχεια μετακινούμε το παράθυρο κατά γ και σε κάθε πέρασμα παίρνουμε μία υπακολουθία. Έτσι δημιουργείται ένα σύνολο τ.ω $\mathcal{S} \subseteq \mathbb{R}^{(N-w)\times w}$ με

$$\mathcal{S} = \left\{(X_0, X_1, ..., X_w)^T, (X_{0+\gamma}, X_{1+\gamma}, ..., X_{w\gamma})^T, ..., (X_{N-w}, X_{N-w+1}, ..., X_N)^T\right\}$$

  Ο k-means έχει σαν παράμετρο τον αριθμό των clusters (συστάδων) που θέλουμε να σχηματίσει. Αφού επιλέξουμε τον αριθμό αυτόν εφαρμόζουμε τον αλγόριθμο στο σύνολο $\mathcal{S}$ και ο οποίος θα υπολογίσει τα κέντρα των clusters, έστω C το σύνολο των κέντρων. Κάθε

υπακολουθία από το σύνολο $\mathcal{S}$ ανήκει σε ένα cluster, η απόσταση από το κέντρο μας δίνει έναν αριθμό τον οποίο ορίζουμε να είναι το σκορ ανωμαλίας. Έτσι για κάθε υπακολουθία ορίζεται το σκορ ανωμαλίας $\epsilon_i = d(s_i, c_i)$, όπου $s_i \in \mathcal{S}$ και $c_i \in C$. Η απόσταση d είναι η συνάρτηση απόστασης.

Ο εντοπισμός των ανωμαλιών γίνεται ορίζοντας ένα φράγμα δ και επιλέγοντας κάθε υπακολουθία με απόσταση μεγαλύτερη από αυτό το φράγμα να είναι ανωμαλία.

Οι Keogh και Jessica έδειξαν ότι αυτή η μέθοδος δεν είναι αξιόπιστη [12]. Αυτό συμβαίνει διότι τα κέντρα των clusters που επιλέγονται μοιάζουν με τα κέντρα που προκύπτουν από μια διαδικασία γνωστή ως Random Walk. Αυτό σημαίνει ότι θα μπορούσαμε να πάρουμε τα κέντρα από ένα Random walk αντί του συνόλου C που είχαμε προηγουμένως. Έγιναν πολλαπλά πειράματα με διάφορα σύνολα δεδομένων και με διαφορετικές αποστάσεις όπως τις Manhattan, L_∞ και Mahalanobis. Σε κανένα πείραμα δεν βγήκε διαφορετικό αποτέλεσμα. Αυτή η δυσκολία είναι προς εξερεύνηση και απασχολεί πολλούς επιστήμονες. Ωστόσο αυτός ο αλγόριθμος θεωρείται θεμελιώδης και γι'αυτό το λόγο έπρεπε να γίνει μια αναφορά.

- **DBSCAN**

Ένας άλλος αλγόριθμος που χρησιμοποιείται σε προβλήματα clustering είναι ο DBSAN ο οποίος χρησιμοποιείται για τον εντοπισμό ανωμαλιών [13]. Ο αλγόριθμος αυτός στηρίζεται στην πυκνότητα των σημείων και έχει μέσα του την έννοια των ανωμαλιών. Χωρίζει δηλαδή το σύνολο των δεδομένων σε τρεις κατηγορίες: στα κύρια σημεία, στα συνοριακά σημεία και στα ανώμαλα σημεία. Οι σημαντικές παράμετροι είναι δύο το ϵ και μ, τα οποία είναι η απόσταση που ορίζει αν ένα σημείο είναι γειτονικό ή όχι και ο ελάχιστος αριθμός σημείων που πρέπει να έχει ένα cluster, αντίστοιχα.

Έστω λοιπόν ένα σύνολο δεδομένων $\mathcal{S}$ οι ϵ – Γείτονες ενός σημείου $s_i \in \mathcal{S}$ είναι το σύνολο $\epsilon - \text{Neighbors}(s_i) = \left\{ s_j : s_j \in \mathcal{S}, \quad s_j \neq s_i, \quad d(s_i, s_j) \leq \epsilon \right\}$, όπου d είναι η συνάρτηση απόστασης.

Ένα σημείο λέγεται είναι κύριο αν $|\epsilon - \text{Neighbors}(s_i)| \geq \mu$.

Ένα σημείο s_i είναι συνοριακό αν

$$\exists s_j \in \mathcal{S} : s_j \neq s_i, \quad s_j \in \epsilon - \text{Neighbors}(s_i), \quad |\epsilon - \text{Neighbors}(s_j)| = 1$$

Τέλος ένα σημείο s_i είναι ανωμαλία αν δεν είναι τίποτα από τα παραπάνω, δηλαδή δεν είναι ούτε κύριο σημείο ούτε συνοριακό σημείο.

Στις χρονοσειρές μπορούμε να εργαστούμε όπως και στον k-means δηλαδή ορίζοντας υπακολουθίες. Η δυσκολία στον DBSCAN είναι να επιλεχθούν κατάλληλα ϵ και μ.

- **Local Outlier Factor**

Ο συγκεκριμένος αλγόριθμος δεν στηρίζεται στην πυκνότητα των σημείων αλλά στη λογική των Κοντινότερων Γειτόνων (Nearest Neighbors) ενώ ταυτόχρονα στηρίζεται και στις τοπικές ακραίες τιμές (Local Outliers).

Έστω $\mathcal{S}$ το σύνολο δεδομένων, για ένα σημείο s_i ορίζουμε το LOF (παράγοντας τοπικής ακραίας τιμής) ως εξής:

(i) **ευρεση του συνόλου των k-Κοντινών Γειτόνων.** Έστω $k \in \mathbb{N}_+$ υπολογίζουμε τις αποστάσεις των σημείων από το s_i και τις βάζουμε στη σειρά από τη μικρότερη στη μεγαλύτερη. Στη συνέχεια επιλέγουμε την k-οστή απόσταση. Αυτή η απόσταση λέγεται $k-$ απόσταση του s_i και γράφουμε $\delta_k(s_i)$. Το σύνολο λοιπόν των k-Κοντινών Γειτόνων του s_i είναι το

$$N_k(s_i) = \left\{ s \in \mathcal{S} : d(s_i, s) \leq \delta_k(s_i) \right\}$$

και η Απόσταση Προσβασιμότητας ενός σημείου s_j είναι

$$RD_k(s_i, s_j) = max\{\delta_k(s_j), d(s_i, s_j)\}$$

(ii) **υπολογίζουμε την Πυκνότητα Προσβασιμότητας.** Για ένα σημείο s_i η πυκνότητα προσβασιμότητας της $k-$ απόστασης του είναι

$$LRD_k(s_i) = \left(\frac{\sum_{s \in N_k(s_i)} RD_k(s_i, s)}{|N_k(s_i)|} \right)^{-1}$$

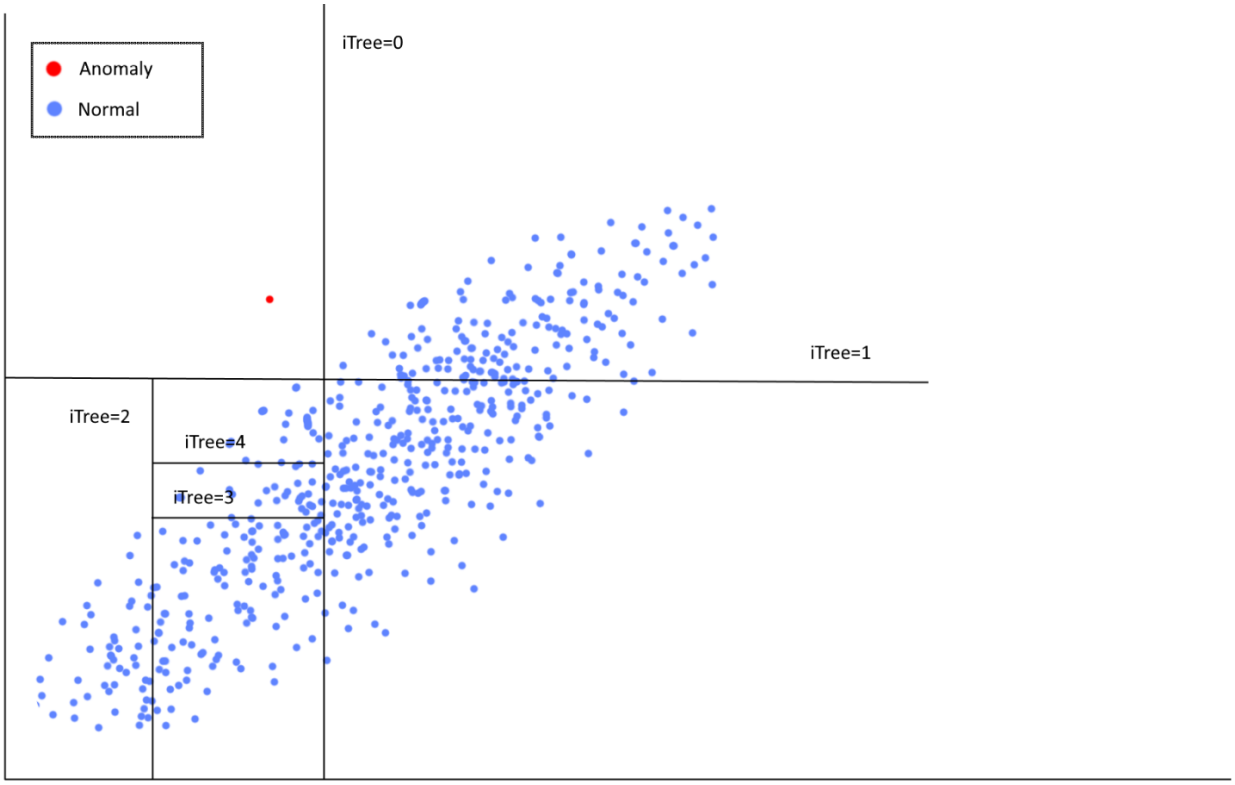
(iii) **τελικά υπολογίζουμε τον LOF.**

$$LOF(s_i) = \frac{\sum_{s \in N_k(s_i)} \frac{LRD_k(s)}{LRD_k(s_i)}}{|N_k(s_i)|}$$

Το LOF χρησιμοποιείται σαν σκορ ανωμαλίας έτσι θέτοντας ένα φράγμα μπορούμε να αποφασίσουμε αν ένα σημείο είναι ανωμαλία ή όχι.

Οι δυσκολίες που αντιμετωπίζουμε με αυτόν τον αλγόριθμο είναι αρκετές. Καταρχήν πρέπει να βρούμε το κατάλληλο k και την κατάλληλη συνάρτηση απόστασης d. Αν και η Ευκλείδεια απόσταση δουλεύει αρκετά καλά πολλές φορές δεν μας δίνει αξιόπιστα αποτελέσματα. Επιπλέον ο αλγόριθμος αναφέρεται στην ”τοπικότητα”. Μία άλλη δυσκολία είναι ότι ο χρόνος στις χρονοσειρές δίνει σημαντικές πληροφορίες κάτι το οποίο δεν το εκμεταλλεύεται ο αλγόριθμος. Τέλος είναι κοστίζει πολύ υπολογιστικά διότι η πολυπλοκότητά του είναι $O(n^2)$.

- **Isolation Forest**



Σχήμα 2.1: Isolation Forest

Ο Isolation Forest παρουσιάστηκε από τον Liu [14] και είναι μία μέθοδος που χωρίζει τον χώρο πολλαπλές φορές μέχρι να απομονώσει τα σημεία. Είναι μία ensemble μέθοδος που χρησιμοποιεί τα iTrees (τα οποία είναι μία δυαδική διαμέριση του χώρου) για το διαχωρισμό του χώρου. Συνήθως οι ανωμαλίες απομονώνονται σε μόλις λίγα βήματα. Στο Σχ. 2.1 βλέπουμε μια ανωμαλία που απομονώθηκε σε μόλις 2 βήματα. Αντίθετα για να απομονωθούν τα φυσιολογικά σημεία απαιτούνται αρκετά βήματα.

Η μέθοδος υλοποιείται σε δύο βήματα: πρώτον δημιουργούμε n iTrees κατά την εκπαί-

δευση και στο δεύτερο βήμα της αξιολόγησης υπολογίζουμε το σκορ ανωμαλίας.

Για να εφαρμόσουμε τον Isolation Forest στο πλαίσιο τον χρονοσειρών αρχικά θα πρέπει να ορίσουμε ένα παράθυρο και στη συνέχεια να εφαρμόσουμε τον αλγόριθμο στο σύνολο των υπακολουθιών. Το σκορ ανωμαλίας βγαίνει από το μέσο όρο των μηκών των δέντρων, δηλαδή το βάθος του Forest.

Οι κύριες δυσκολίες αυτής τη μεθόδου είναι τρεις, το μέγεθος του παραθύρου, ο αριθμός των δέντρων και η παράμετρος contamination. Εάν το μέγεθος του παραθύρου είναι μικρό τότε δεν θα υπάρχουν αρκετά δεδομένα ώστε να μπορέσει να φτιαχτεί ένα καλό μοντέλο. Από την άλλη αν είναι μεγάλο, τότε τα παλαιότερα χρονικά δεδομένα θα έχουν το ίδιο βάρος με τα νεότερα. Επιπλέον αν ο αριθμός των δέντρων είναι μεγάλος τότε είναι πιο κοντά στην αναμενόμενη τιμή, όμως, το κύριο πρόβλημα είναι ότι αυξάνεται η υπολογιστική πολυπλοκότητα διότι ο αλγόριθμος τρέχει σε $O(Lw^2)$, όπου L είναι ο αριθμός των δέντρων που κατασκευάζονται και w είναι το μέγεθος του παραθύρου. Τέλος πολλές υλοποιήσεις βασίζονται σε μια παράμετρο που λέγεται contamination, η οποία δείχνει το ποσοστό των ανωμαλιών που περιέχει το σύνολο δεδομένων μας. Η παράμετρος αυτή επηρεάζει το φράγμα για τον εντοπισμό των ανωμαλιών έτσι εάν κάνουμε κακή επιλογή κινδυνεύουμε να πάρουμε αρκετά false positives ή false negatives.

- **Άλλες τεχνικές.**

  Μία ενδιαφέρουσα τεχνική που πρέπει να αναφερθεί είναι μια παραλλαγή του αλγόριθμου SVM που ονομάζεται **OC-SVM** [15, 16]. Τέλος έχουμε και τον γνωστό αλγόριθμο **XGBoost** που είναι μια Boosting τεχνική [17].

2.3 Μέθοδοι Βαθιάς Μηχανικής Μάθησης

Οι μέθοδοι αυτοί μοιάζουν με τις μεθόδους της Μηχανικής Μάθησης διότι δεν θεωρούν ότι τα δεδομένα προέρχονται με κάποιο μηχανισμό από πίσω. Θεωρούν το μηχανισμό σαν μαύρο κουτί και προσπαθούν να κάνουν προβλέψεις εμπειρικά. Σε αυτήν την ενότητα θα δούμε μεθόδους που στηρίζονται στα Νευρωνικά Δίκτυα.

- **Multiple Layer Perceptron.**

  Η βασικότερη αρχιτεκτονική των νευρωνικών δικτύων είναι το MLP. Εδώ θα ορίσουμε ένα παράθυρο μήκους w. Στη συνέχεια θα πάρουμε τις υπακολουθίες και θα κάνουμε πρόβλεψη με τον εξής τρόπο: για την υπακολουθία $(x_{i-w}, x_{i-w+1}, ..., x_i)$ μπορούμε να προβλέ-

ψουμε την επόμενη τιμή $\hat{y} = f(x_{i+1})$. Εναλλακτικά μπορούμε να προβλέψουμε μεγαλύτερο διάστημα μήκους m, $\hat{y} = f((x_{i+1}, x_{i+2}, ..., x_{i+m}))$. Το διάστημα μήκος m λέγεται παράθυρο πρόβλεψης.

Για τον εντοπισμός ανωμαλιών θέτουμε ένα φράγμα δ και για κάθε πρόβλεψη παίρνουμε το λάθος $\epsilon_i = \hat{y}_i - x_i$ το οποίο θα χρησιμοποιηθεί ώς σκορ ανωμαλίας και έχουμε:

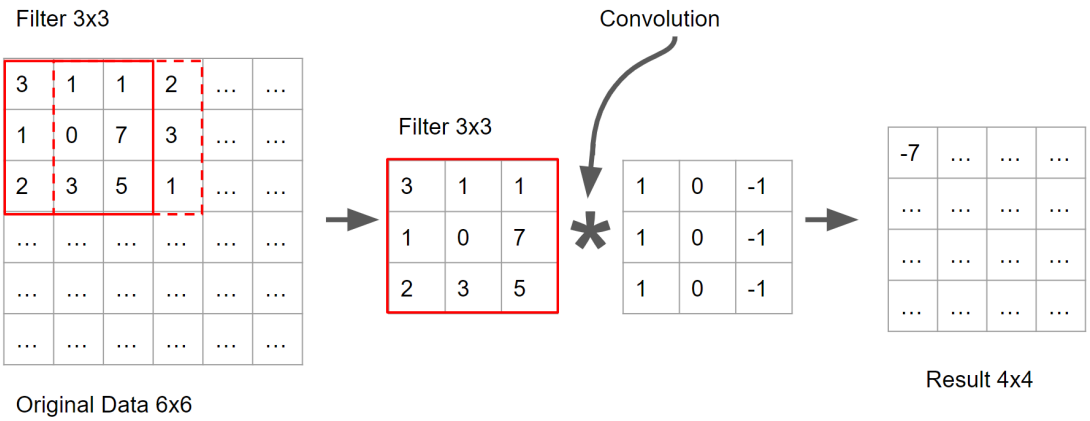
$$\varphi(x_i) = \begin{cases} 1 & \text{, Αν } \epsilon_i \geq \delta \\ 0 & \text{, Αλλιώς} \end{cases}$$

Ο αλγόριθμος αυτός έχει πολλές παραμέτρους άρα και αρκετές δυσκολίες. Αρχικά έχουμε το βάθος του δικτύου και το πλάτος του, δηλαδή τον αριθμό των κρυμμένων επιπέδων και τον αριθμό των κόμβων σε κάθε επίπεδο αντίστοιχα. Στη συνέχεια θα πρέπει να ορίσουμε το μήκος του παραθύρου και τέλος έχουμε την συνάρτηση βελτιστοποίησης.

- **Convolutional Neural Networks** Τα convolutional νευρωνικά δίκτυα έχουν γνωρίσει μεγάλη επιτυχία στον τομέα της αναγνώρισης εικόνας, αναγνώρισης αντικειμένων, στην κατηγοριοποίηση κλπ. Ωστόσο μπορούμε να τα χρησιμοποιήσουμε και στον εντοπισμό ανωμαλιών. Σε αντίθεση με τα MLP δίκτυα τα οποία είναι πλήρως συνδεδεμένα (άρα έχουμε και πολλές παραμέτρους) τα convolutional δίκτυα είναι μερικώς συνδεδεμένα κάτι το οποίο τους επιτρέπει να έχουν λιγότερες μεταβλητές να υπολογίσουν, κάνοντάς τα πιο γρήγορα στην εκπαίδευση ενώ ταυτόχρονα τους επιτρέπει να έχουν μεγαλύτερο βάθος. Επιπροσθέτως τα δίκτυα αυτά χρησιμοποιούν μια τεχνική που λέγεται pooling για την αποφυγή της υπερ-εκπαίδευσης (overfitting). Μία άλλη τεχνική που χρησιμοποιείται συχνά ονομάζεται Batch Normalization [18] η οποία είναι μια εναλλακτική για την dropout δηλαδή μειώνει τη σημαντικότητα αρχικοποίησης τιμών.

  Για να εκπαιδευτεί ένα Convolutional Network εφαρμόζεται ένα φίλτρο (ένας πίνακας δηλαδή $n \times k$, το οποίο συνήθως είναι 3×3, 5×5) το οποίο μετατοπίζεται ώστε να σκανάρει όλα τα δεδομένα και να εφαρμοστεί ο τελεστής της συνέλιξης (convolution) Σχ. 2.2.

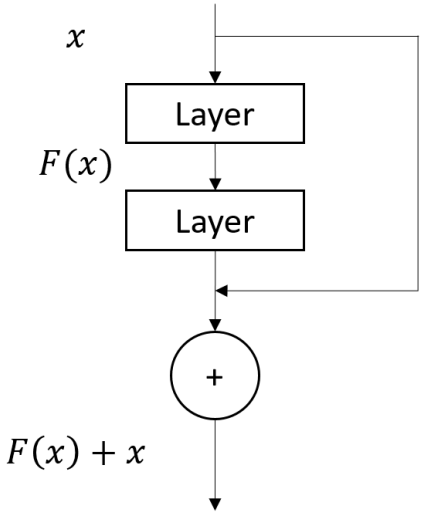
  Η χρήση των convolutional στον εντοπισμό ανωμαλιών γίνεται κατά τον ίδιο τρόπο με πριν, δηλαδή ορίζουμε ένα παράθυρο και η εκπαίδευση γίνεται πάνω στις υπακολουθίες. Ο εντοπισμός γίνεται με τον ορισμό ενός φράγματος και τελικά η απόφαση παίρνεται πάνω στα λάθη ανάμεσα στις προβλέψεις και στην πραγματική τιμή, δηλαδή έχουμε ανωμαλία αν $f((x_{i-w}, x_{i-w+1}, ..., x_i)) - x_{i+1} \geq \delta$



Σχήμα 2.2: Filter of Convolutonal Layer

Οι δυσκολίες που αντιμετωπίζει αυτή η μέθοδος είναι κατά κύριο λόγο η αρχιτεκτονική. Εδώ έχουμε μία πολύ μεγάλη δυσκολία διότι η αρχιτεκτονική του δικτύου παίζει μεγάλο ρόλο, δηλαδή έχει σημασία αν θα χρησιμοποιήσουμε dropout, Batch Normalization ή/και max-pooling. Επιπλέον έχει σημασία ο αριθμός και το μέγεθος των πυρήνων που θα χρησιμοποιηθούν σε κάθε convolutional επίπεδο. Πυρήνας ονομάζεται ένα φίλτρο που σαρώνει ένα convolutional επίπεδο το οποίο εξάγει διάφορα χαρακτηριστικά. Τέλος σημαντικός παράγοντας είναι και το βάθος το οποίο θα πρέπει να αποφασιστεί κατά τον ορισμό του δικτύου.

- **Residual Neural Network**



Σχήμα 2.3: Residual Network

Τα residual δίκτυα είναι μια επέκταση των Convolutional. Τα δίκτυα αυτά συνδυάζουν ένα αποτέλεσμα από τα προηγούμενα επίπεδα με το αποτέλεσμα ενός μεταγενέστερου επιπέδου Σχ. 2.1 το block αυτό ονομάζεται residual block [19]. Για να ορίσουμε ένα residual block

θεωρούμε ένα x ώς η είσοδος του, και φ ένα convolutional block, το οποίο αποτελείται από convolutional επίπεδα, max-pooling, batch normalization κλπ. Η έξοδος ενός residual block είναι $y = x + \varphi(x)$. Συνήθως συμπεριλαμβάνεται και μία συνάρτηση ενεργοποίησης Ψ και τελικά έχουμε $y = \Psi(x + \varphi(x))$.

Τα residuals δίκτυα χρησιμοποιούνται για να λύσουν το πρόβλημα της ”εξαφάνισης της παραγώγου” που προκύπτει από τα Convolutional δίκτυα λόγω του βάθους τους. Στην ουσία το πρόβλημα αυτό αναφέρεται σε παραγώγους οι οποίες είναι τόσο μικρές που στην ουσία μηδενίζονται (εξαφανίζονται).

- **LSTM - Long Short Term Memory Network.**

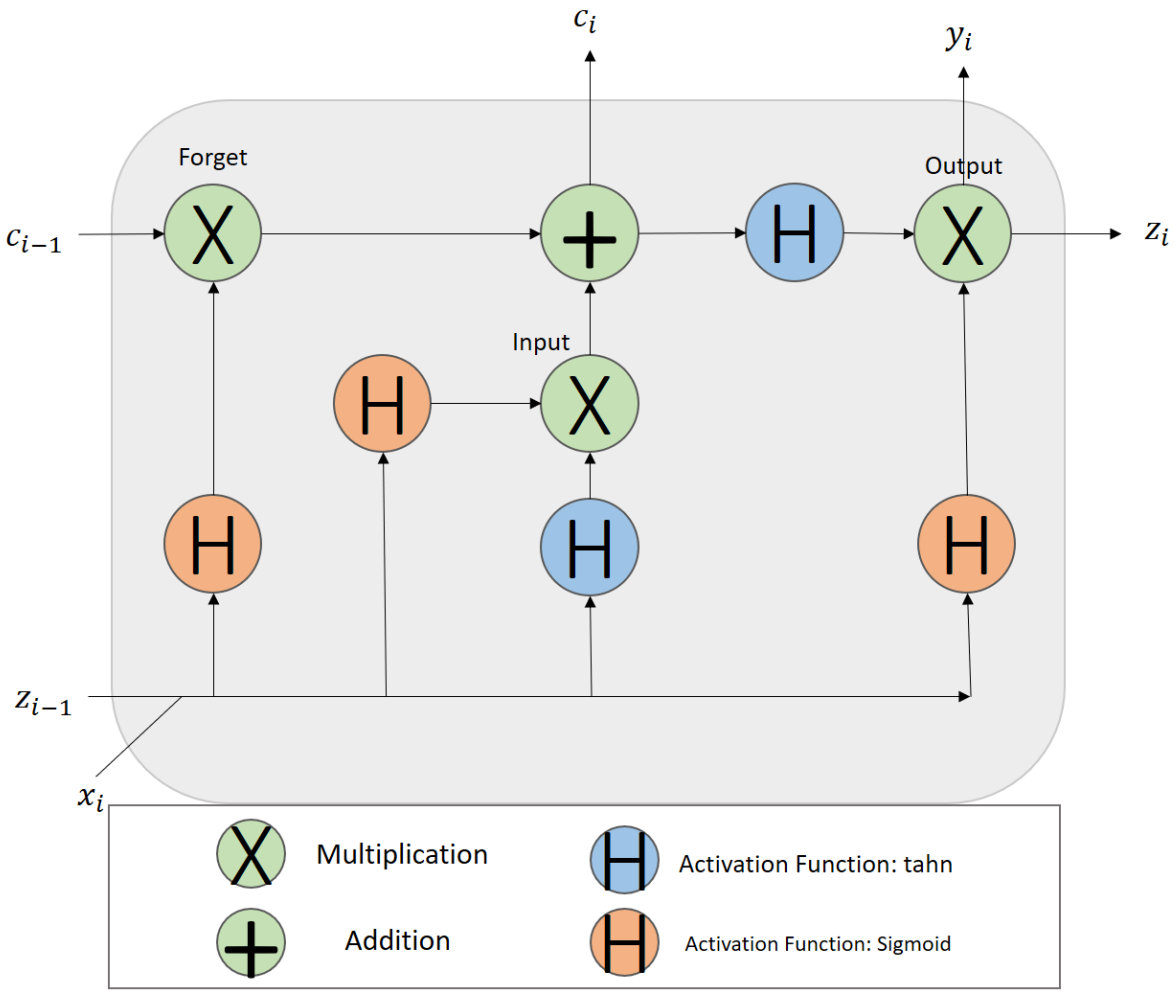
Ένα δίκτυο γνωστό για την εφαρμογή του σε σειριακά (ακολουθιακά) δεδομένα είναι το δίκτυο LSTM το οποίο ανήκει στην οικογένεια των Recurrent Νευρωνικών Δικτύων. Σε αντίθεση με τα προηγούμενα όπου τα δεδομένα είχαν μια συνεχή πορεία προς τα εμπρός, ένα Recurrent δίκτυο διαθέτει μία σύνδεση με τα δεδομένα που ήδη έχουν περάσει, κάτι το οποίο του επιτρέπει να αντλεί πληροφορία σύμφωνα με άλλα δεδομένα και να συνδυάζει τη γνώση. Επίσημα λοιπόν η έξοδος που παράγεται είναι η εξής:

$$y_i = h({x_i}^T \cdot w_x + {y_{i-1}}^T \cdot w_y + b)$$

όπου h είναι η συνάρτηση ενεργοποίησης, x_i είναι ένα στοιχείο από το σύνολο των δεδομένων, y_{i-1} είναι μία ”προηγούμενη” προβλεπόμενη τιμή, w_x, w_y είναι τα βάρη και b μία σταθερά.

Ένα δίκτυο LSTM είναι λοιπόν μια παραλλαγή ενός τέτοιου δικτύου [20] στην οποία περιλαμβάνονται δύο διανύσματα c_i και z_i. Η z_i προστίθεται με το x_i και καλείται short term ενώ η c_i πολλαπλασιάζεται με το x_i και καλείται long term. Επιπλέον υπάρχουν 3 ”πύλες” που ρυθμίζουν την πληροφορία που θα πεταχτεί (forget πύλη), την πληροφορία που θα περάσει (input πύλη) και την πληροφορία που θα αποτελέσει την έξοδο (output πύλη). Αυτός ο σχεδιασμός έχει ώς αποτέλεσμα να αποθηκεύεται σημαντική πληροφορία κατά την πρόσθεση ώστε να χρησιμοποιηθεί πιο μετά (long term) και με τη χρήση του πολλαπλασιασμού (κατά σημείο) επιτυγχάνεται η διαγραφή πληροφορίας που μας είναι άχρηστη σύμφωνα με το μοντέλο από τον long term όρο. Τέλος η πύλη εξόδου αποφασίζει ποιο κομμάτι του long-term όρου θα είναι η έξοδος Σχ. 2.4.

Τα βήματα που εκτελεί το LSTM είναι τα εξής:



Σχήμα 2.4: LSTM

(i) Καταρχήν θα μεταφερθούμε στην forget πύλη όπου θα αποφασιστεί τι πληροφορία θα πεταχτεί. Παίρνει την είσοδο x_i και την προηγούμενη έξοδο z_{i-1}, και με την συνάρτηση ενεργοποίησης Sigmoid (έστω h_s) έχουμε $f_i = h_s(W_f \cdot [z_{i-1,x_i}] + b_f)$. Σαν τελευταίο βήμα όταν πάμε στην πύλη forget πολλαπλασιάζουμε με την παλιά κατάσταση και έχουμε $F_i = f_i \cdot C_{i-1}$

(ii) Στη συνέχεια αποφασίζεται τι πληροφορία θα κρατήσει (input πύλη). Αρχικά εφαρμόζεται μία συνάρτηση ενεργοποίησης sigmoid στην είσοδο x_i και στην προηγούμενη έξοδο z_{i-1} η οποία θα αποφασίσει ποιες τιμές πρέπει να ανανεωθούν $u_i = h_s(W_u \cdot [z_{i-1}, x_i] + b_u)$. Και δεύτερον θα εφαρμοστεί μία συνάρτηση ενεργοποίησης tanh (έστω $h_{\tanh}$), $\hat{C}_i = h_{\tanh}(W_C \cdot [z_{i-1}, x_i] + b_C)$ για την ανανέωση. Τελικά θα πάρουμε μετά την εφαρμογή της πύλης input $I_i = u_i \cdot \hat{C}_i$

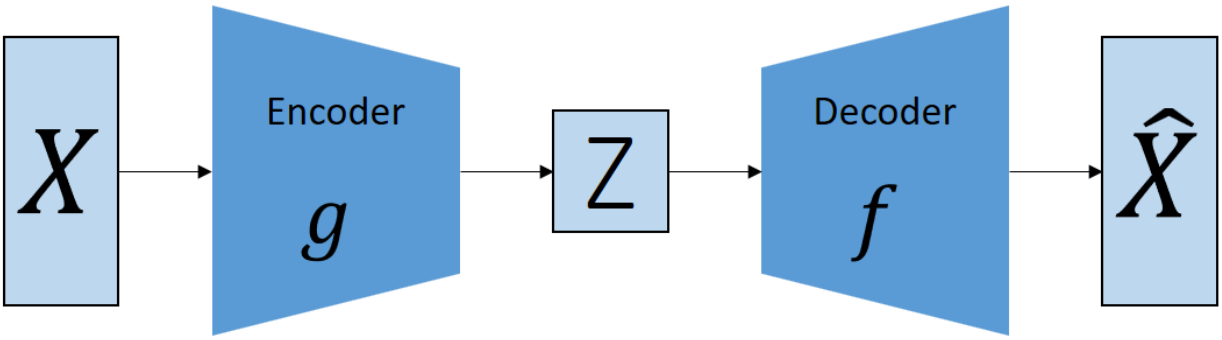
(iii) Στην συνέχεια συνδυάζουμε τα δύο προηγούμενα βήματα δηλαδή εφαρμόζουμε μία πρόσθεση και έχουμε $C_i = F_i + I_i$

(iv) Εφόσον έχουν γίνει τα προηγούμενα βήματα τώρα είναι η στιγμή που θα αποφασιστεί η έξοδος. Αρχικά περνάμε το προηγούμενο αποτέλεσμα από τη συνάρτηση ενεργοποίησης tanh ώστε να περιορίσουμε τις τιμές στο διάστημα -1, 1 παίρνοντας $r_i = h_{\tanh}(C_i)$ και μετά θα περάσουμε την είσοδο x_i και την προηγούμενη έξοδο από την συνάρτηση ενεργοποίησης sigmoid για να πάρουμε $o_i = h_s(W_o \cdot [z_{i-1}, x_i] + b_o)$. Μετά τον πολλαπλασιασμό θα πάρουμε την έξοδο από την πύλη output $O_i = o_i \cdot r_i$

(v) κατά τη διάρκεια της διαδικασία αυτής αποθηκεύονται οι τιμές των C_i, $O_i = z_i$ για μελλοντική χρήση

- **Autoencoder.**

Η μέθοδος autoencoder υλοποιείται σε δύο βήματα. Στο πρώτο βήμα το μοντέλο κωδικοποιεί την είσοδο και στο δεύτερο βήμα αποκωδικοποιεί την είσοδο Σχ. 2.5. Η κύρια ιδέα πίσω από αυτόν τον αλγόριθμο είναι ότι αν προβάλουμε τα σημεία σε έναν χώρο λιγότερων διαστάσεων τα φυσιολογικά δεδομένα θα διαφέρουν σημαντικά από τις ανωμαλίες. Οπότε αν κάνουμε την αντίστροφη διαδικασία, τα σημεία που έχουν σημαντικές διαφορές σε σχέση με την αρχική τους κατάσταση θα είναι ανωμαλίες. Αυτή η ιδέα κάνει τον autoencoder πολύ καλή επιλογή για την ανίχνευση ανωμαλιών.



Σχήμα 2.5: Autoencoder

Τέτοιου τύπου δίκτυα ανήκουν στην οικογένεια των δικτύων εμπρόσθιας τροφοδότησης ή αλλιώς feed-forward (όπως ήταν και ο αλγόριθμος MLP που είδαμε νωρίτερα) σε αντίθεση με τα LSTM. Το πρώτο βήμα της διαδικασίας ονομάζεται κωδικοποιητής (Encoder) ενώ το δεύτερο αποκωδικοποιητής (Decoder). Επίσημα έχουμε:

Έστω X το σύνολο δεδομένων, f η συνάρτηση decoder και g η συνάρτηση encoder, έχουμε:

$$\hat{X} = f(g(X))$$

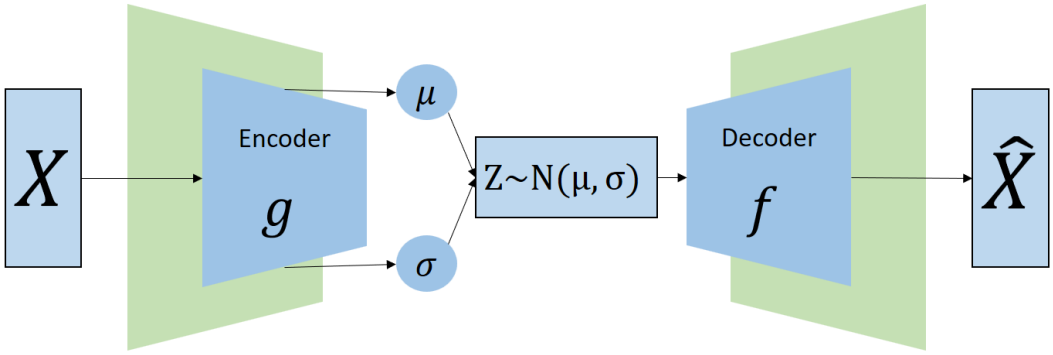
Η συνάρτηση βελτιστοποίησης θα προσπαθήσει να μειώσει τη διαφορά ανάμεσα στο $\hat{X}$ και X, δηλαδή:

$$\min_{w_f, w_g} \|X - \hat{X}\|_2 = \min_{w_f, w_g} \|X - f(g(X))\|_2$$

Τελικά από κάθε x_i παίρνουμε ένα $\hat{x}_i$ και έχουμε το λάθος $\epsilon = |x_i - \hat{x}_i|$ το οποίο το χρησιμοποιούμε σαν σκορ ανωμαλίας. Ορίζοντας ένα φράγμα δ μπορούμε να αποφασίσουμε τι είναι ανωμαλία και τι όχι.

Συχνά τα autoencoder συνδυάζονται με άλλα δίκτυα, για παράδειγμα CNN δίκτυα ή LSTM. Με αυτή την σύνθεση το ρόλο των encoder/decoder παίρνουν αυτά τα δίκτυα και το αποτέλεσμα είναι να πάρουμε τα πλεονεκτήματα και των δύο τεχνικών. Η διαδικασία είναι η εξής: αρχικά τα δεδομένα περνάνε από το δίκτυο (πρώτο βήμα Encoding) ώστε να μειωθούν οι διαστάσεις και στη συνέχεια στο ίδιο δίκτυο με την αντίστροφη αρχιτεκτονική (δεύτερο βήμα Decoding) ώστε να ξαναπάρουμε ένα στοιχείου του αρχικού χώρου. Οι ανωμαλίες εντοπίζονται όπως είπαμε πριν.

- **Variational Autoencoder και LSTM-VAE**



Σχήμα 2.6: LSTM-VAE

Παραλλαγή του autoencoder είναι ο Variational autoencoder (ή Auto-Encoding Variational Bayes [21]) όπου επιλέγεται δειγματοληπτικά ένα Z από μία κανονική κατανομή, ενώ ο encoder και ο decoder είναι η posterior $q(z; x)$ και η prior $p(\hat{x}; z)$.

Το δίκτυο LSTM-VAE συνδυάζει έναν Variational Autoencoder με ένα LSTM Σχ. 2.6 το οποίο στην πραγματικότητα είναι μια παραλλαγή του LSTM-AE.

- **Άλλες τεχνικές.**

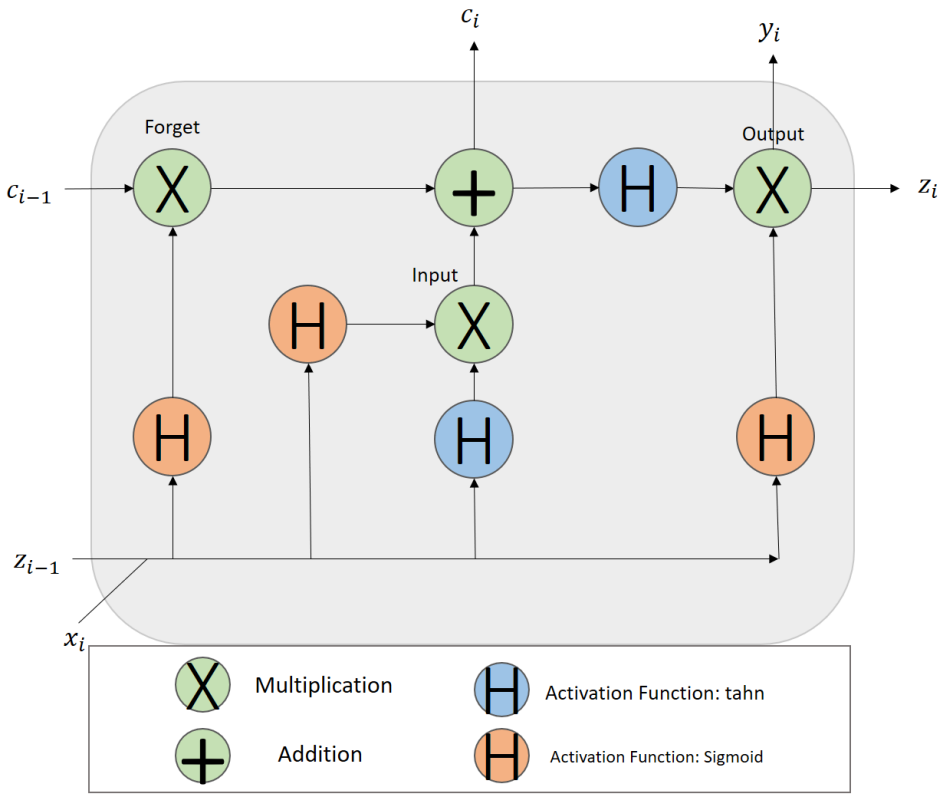
Επιπλέον πριν κλείσουμε αυτήν την ενότητα θα πρέπει να κάνουμε μια αναφορά σε δίκτυα που αξίζει να αναφερθούν όπως το WaveNet [22] και μια παραλλαγή του LSTM το

GRU [23]. Στην περίπτωση του WaveNet η χρονοσειρά εισάγεται όπως ακριβώς και σε ένα convolutional δίκτυο, ορίζοντας ένα παράθυρο το οποίο ορίζει υπακολουθίες οι οποίες με τη σειρά τους δίνονται στο μοντέλο. Η η κυριότερη διαφορά του WaveNet με το convolutional είναι ότι το WaveNet χρησιμοποιεί διαστελλόμενα φίλτρα Σχ. 2.7.

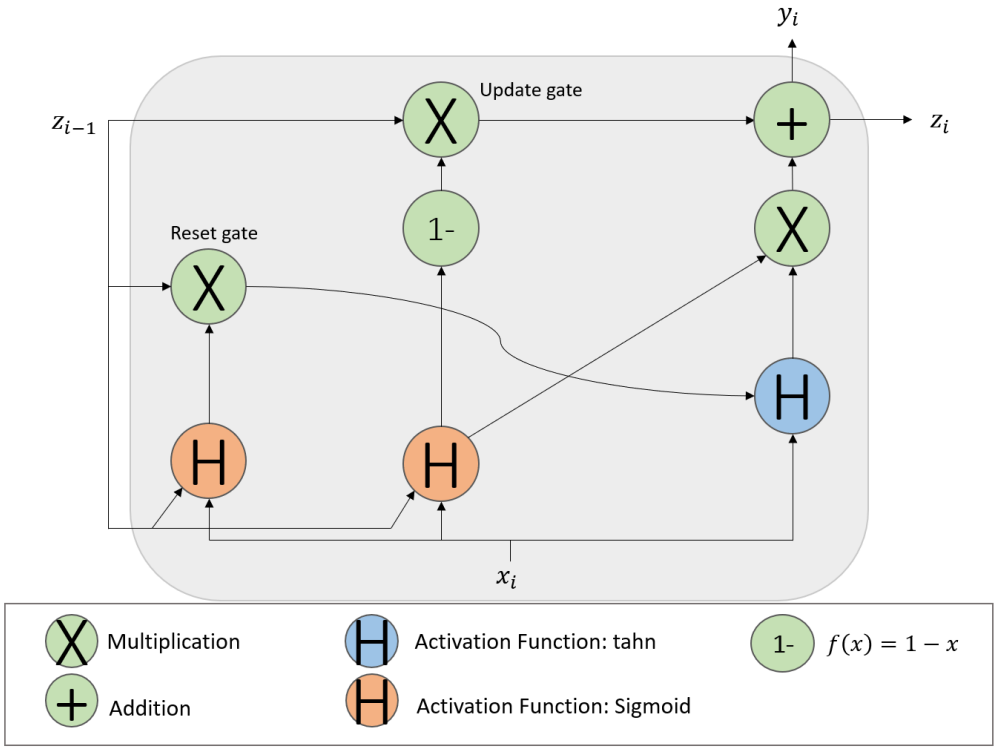
(α') Without dilation (Χωρίς διαστολή) (β') With dilation (Με διαστολή)

Σχήμα 2.7: no dilation vs dilation

Από την άλλη το GRU είναι μία παραλλαγή του LSTM, οι χρονοσειρές εισάγονται κατά τον ίδιο τρόπο με το LSTM ορίζοντας ένα παράθυρο μήκους w. Η ομοιότητα με το LSTM είναι μεγάλη και στη βιβλιογραφία φαίνεται να έχουν τις ίδιες επιδόσεις. Η μεγαλύτερη διαφορά είναι ότι το GRU έχει πιο απλή δομή Σχ. 2.8. Ως εκ τούτου έχει λιγότερες μεταβλητές και άρα εκπαιδεύεται σε λιγότερο χρόνο.



(α') LSTM



(β') GRU

Σχήμα 2.8: LSTM cell vs GRU cell

Chapter 3

Ανίχνευση εξαιρέσεων σε δεδομένα χρονοσειρών με χρήση ensemble τεχνικών

3.1 Ensemble

Στο κεφάλαιο αυτό θα δούμε κάποιες τεχνικές ensemble οι οποίες θα συγκριθούν με τους βασικούς αλγόριθμους στους οποίους στηρίζονται. Ensemble ονομάζονται οι μέθοδοι machine learning που βασίζονται σε άλλα μοντέλα. Στην ουσία συγκεντρώνουν τα αποτελέσματα άλλων μοντέλων και τα συνδυάζουν σε ένα τελικό αποτέλεσμα. Η ομάδα των μοντέλων και ο συνδυασμός τους ονομάζεται Ensemble. Τα ensemble μοντέλα έχουν χρησιμοποιηθεί αρκετά σε διάφορους τύπους προβλημάτων, όπως στην κατηγοριοποίηση ή στην πρόβλεψη τιμών. Γνωστοί αλγόριθμοι όπως οι Random Forest και Majority Voting έχουν μεγάλη επιτυχία σε τέτοιου είδους προβλήματα. Αλλά και πιο σύνθετοι αλγόριθμοι όπως το Bagging, το boosting και το Stacking έχουν χρησιμοποιηθεί πολύ. Η μέθοδος Bagging αποτελείται από δύο βήματα. Στο πρώτο βήμα έχουμε κάποια αρχικά μοντέλα και επιλέγουμε μόνο μερικά από τα δεδομένα για κάθε ένα μοντέλο ώστε να εκπαιδευτεί σε περιορισμένο αριθμό παρατηρήσεων. Αυτό έχει ως αποτέλεσμα το κάθε μοντέλο να βρίσκει διαφορετικά μοτίβα στο σύνολο των δεδομένων. Στο δεύτερο βήμα συγκεντρώνουμε τα αποτελέσματα και καταλήγουμε σε μία πρόβλεψη η οποία προκύπτει από το συνδυασμό των επιμέρους προβλέψεων. Στη μέθοδο boosting εκπαιδεύουμε τα μοντέλα κατά σειρά το ένα μετά το άλλο, δίνοντας στο ένα μοντέλο την έξοδο του προηγούμενου. Έτσι ένα μοντέλο εκπαιδεύεται από τα λάθη του προηγούμενού του με αποτέλεσμα όσο η διαδικασία προχωράει τα μοντέλα που βρίσκονται προς το τέλος να γίνονται πιο ισχυρά. Στην τεχνική stacking, σε αντίθεση με το boosting, όλα τα μοντέλα εκπαιδεύονται παράλληλα, ενώ η διαφορά με την

τεχνική bagging είναι ότι η εκπαίδευση όλων των μοντέλων γίνεται σε όλα τα δεδομένα από το σύνολο δεδομένων εκπαίδευσης. Στο τελικό στάδιο ενός stacking βρίσκεται ένα μοντέλο το οποίο θα εκπαιδευτεί ώστε να συνδυάζει σωστά τα αποτελέσματα των υπόλοιπων μοντέλων. Το τελικό μοντέλο ονομάζεται μετα-μοντέλο.

Παρόλο που οι ensemble τεχνικές είναι διαδεδομένες, στο χώρο της ανίχνευσης ανωμαλιών σε χρονοσειρές πολλών μεταβλητών δεν έχουν μελετηθεί τόσο πολύ. Σε αυτή τη διπλωματική δίνουμε ensemble τεχνικές που έχουν χρησιμοποιηθεί για άλλου είδους προβλήματα ενώ ταυτόχρονα συγκρίνουμε τα αποτελέσματα με τους βασικούς αλγόριθμους που χρησιμοποιήθηκαν για να συνθέσουν τα ensemble. Επίσης θα δώσουμε και μία υλοποίηση η οποία δεν βρέθηκε στην βιβλιογραφία της ανίχνευσης ανωμαλιών, τουλάχιστον όσον αφορά την προσωπική μας αναζήτηση. Τελικά θα χρησιμοποιήσουμε τις παραπάνω τεχνικές όπως το bagging και το stacking. Αρχικά θα δημιουργήσουμε μοντέλα από την τεχνική bagging δίνοντας και στη συνέχεια θα δώσουμε τις εξόδους τους σε ένα μετα-μοντέλο σύμφωνα με την τεχνική του stacking.

Στις παρακάτω τεχνικές που θα δούμε έχουμε χρησιμοποιήσει παντού τα ίδια μοντέλα τα οποία είναι: Autoencoder, LSTM, Autoencoder με LSTM (LSTM-AE), Convolutional Autoencoder (CONV-AE) και ένα LSTM-VAE.

3.2 Δεδομένα

Τα dataset που έχουν χρησιμοποιηθεί είναι από το SKAB (Skoltech Anomaly Benchmark) [24]. Τα dataset έχουν παραχθεί από μία συσκευή η οποία περιλαμβάνει:

- Ένα σύστημα κυκλοφορίας νερού.
- Ένα σύστημα ελέγχου της κυκλοφορίας νερού.
- Ένα σύστημα για την παρακολούθηση της κατάστασης του νερού στο σύστημα κυκλοφορίας.
- Μία τεχνολογία (TSN) για την καταγραφή του χρόνου.
- Τέλος ένα σύστημα για την αποθήκευση δεδομένων.

Επίσης η συσκευή αυτή διαθέτη βαλβίδες και μηχανισμούς ώστε να δημιουργεί μη-φυσιολογική λειτουργία αλλά και αισθητήρες.

Τα δεδομένα είναι χρονοσειρές πολλών μεταβλητών. Κάθε γραμμή μέσα στα dataset συμβολίζει μία καταγραφή μίας δεδομένης χρονικής στιγμής η οποία αποτελείται από 8 μεταβλητές,

τη χρονική στιγμή που έγινε η παρατήρηση και την ένδειξη αν είναι ανωμαλία ή όχι:

- datetime - Είναι η χρονική στιγμή που καταγράφεται η παρατήρηση.
- Accelerometer1RMS - Επιτάχυνση (μονάδα g)
- Accelerometer2RMS - Επιτάχυνση (μονάδα g)
- Current - Δείχνει το ρεύμα (μονάδα Ampere)
- Pressure - Πίεση (μονάδα Bar)
- Temperature - Θερμοκρασία (Βαθμοί Κελσίου)
- Thermocouple - Θερμοκρασία (Βαθμοί Κελσίου)
- Voltage - Τα βολτ του μηχανήματος (μονάδα Volt)
- RateRMS - Ρυθμός κυκλοφορίας ροής του ρευστού μέσα στο βρόγχο (Μονάδα Λίτρα/Λεπτό)
- anomaly - Δείχνει αν το σημείο είναι ανωμαλία (0 ή 1)

Τα δεδομένα προσομοιώνουν διάφορες καταστάσεις (κάθε σύνολο δεδομένων περιγράφει διαφορετική κατάσταση και συνολικά έχουμε επιλέξει 31 σύνολα δεδομένων) όπως για παράδειγμα Προσομοίωση διαρροής υγρού και προσθήκη, Απότομη συμπεριφορά ανισορροπίας, Απότομη αύξηση/μείωση υγρού, Αργή αύξηση υγρού, Αύξηση υγρού υψηλής θερμοκρασίας και άλλα. Ανάλογα με την περίπτωση έχει φτιαχτεί διαφορετικό σύνολο δεδομένων όπου αρχικά το σύστημα δουλεύει σε φυσιολογικές συνθήκες και στη συνέχεια γίνεται η προσομοίωση ενός σεναρίου όπου παρατηρούμε τις ανωμαλίες.

Το SKAB λοιπόν είναι μία συλλογή από δεδομένα τα οποία μπορούν να χρησιμοποιηθούν για την αξιολόγηση τεχνικών και μοντέλων στο πλαίσιο της έρευνας στην Ανίχνευση Ανωμαλιών σε χρονοσειρές πολλών μεταβλητών. Οι συγγραφείς στο [24] έχουν πειραματιστεί με αρκετά από τα μοντέλα που έχουμε ήδη περιγράψει όπως Convolutional Autoencoder, LSTM, LSTM Variational Autoencoder, Isolation forest και άλλα. Τα αποτελέσματα που πέτυχαν δείχνουν ότι πετυχαίνουν 0.79 F1-Score με το Convolutional Autoencoder στην πρώτη θέση ενώ στην τελευταία θέση βρίσκεται το Isolation Forest με 0.4 F1-Score, Πίνακας 3.1α'. Στο [25] προτείνεται μία νέα αρχιτεκτονική Νευρωνικού Δικτύου από την οποία προκύπτει ένα υβριδικό Δίκτυο που προέρχεται από τα δίκτυα Capsule και LSTM. Οι συγγραφείς χρησιμοποιούν τα datasets του SKAB για να αξιολογήσουν την αρχιτεκτονική τους. Και εδώ παρατηρούμε τα αποτελέσματα να κυμαίνονται στα

ίδια επίπεδα με ένα LSTMCaps στην πρώτη θέση με 0.74 F1-Score ενώ το Isolation Forest βρίσκεται στην τελευταία θέση με 0.4 F1-Score Πίνακας 3.1β'. Τέλος στο [26] προτείνεται ένα μοντέλο που ονομάζεται TRL-CPC το οποίο μαθαίνει να αναπαριστά τη χρονοσειρά μέσω του Contrastive Learning. Contrastive Learning λέγεται η εκπαίδευση ενός μοντέλου όπου τα δείγματα συγκρίνονται μεταξύ τους και το μοντέλο μαθαίνει να βρίσκει τις ομοιότητες των χαρακτηριστικών των δειγμάτων που ανήκουν στην ίδια κλάση. Οι συγγραφείς καταφέρνουν να πετύχουν ένα 0.70 F1-Score με το TRL-CPC στο σύνολο δεδομένων του SKAB.

Model Title	F1 Score
Conv-AE	0.79
MSET	0.73
LSTM-AE	0.68
T-squared+Q (PCA)	0.67
LSTM	0.64
MSCRED	0.64
LSTM-VAE	0.56
T-squared	0.56
Autoencoder	0.45
Isolation forest	0.4

(α') SKAB Results

Model Title	F1 Score
LSTMCaps	0.74
MSET	0.73
LSTMCapsV2	0.71
MSCRED	0.7
LSTM	0.67
Conv-AE	0.66
LSTM-AE	0.65
LSTM-VAE	0.56
Autoencoder	0.45
Isolation forest	0.4

(β') Hybridization of Capsule and LSTM Networks

Πίνακας 3.1: Αποτελέσματα από άλλες ερευνητικές εργασίες

3.3 Majority Voting

Η λογική του Majoriy Voting στηρίζεται στο εξής: Αν κάνουμε μια ερώτηση σε 1000 ανθρώπους τότε κατά πάσα πιθανότητα η πλειοψηφία να βρει τη σωστή απάντηση. Αυτό ονομάζεται "η σοφία της πλειοψηφίας". Στα μαθηματικά αυτό δείχνεται με τη βοήθεια της διωνυμικής κατανομής: Έστω X_i ανεξάρτητες και ισόνομες τυχαίες μεταβλητές με $X_i \sim bernoulli(p), \quad i = 1, 2, ..., 1000$ όπου η πιθανότητα σωστής απάντησης είναι $p = 51\%$. Οι X_i αναλογούν στους ανθρώπους και η πιθανότητα 51% αναλογεί στην ικανότητά τους να απαντάνε σωστά. Δηλαδή είναι ελάχιστα καλύτεροι από το να απαντούσαν στην τύχη γυρίζοντας ένα νόμισμα. Τότε η πιθανότητα πάνω από τους μισούς να απαντήσουν σωστά είναι $P(X \geq 501) = 1 - P(X \leq 500)$. Από τη διωνυμική κατανομή ξέρουμε ότι $P(X \leq 500) = \sum_{n=0}^{499} \binom{1000}{n} (\frac{51}{100})^n (1 - \frac{51}{100})^{1000-n} \approx 25\%$. Οπότε έχουμε ότι $P(X \geq 501) \approx 75\%$. Το οποίο σημαίνει ότι έχουμε 75% πιθανότητα περισσότεροι από 500 άτομα να απαντήσουν σωστά. Όσο αυξάνουμε τα άτομα τόσο πιο πολύ μεγαλώνει αυτή η πιθανότητα (για παράδειγμα αν έχουμε 10000 άτομα η πιθανότητα αυτή βρίσκεται κοντά στο 97%).

Στην τεχνική αυτή χρησιμοποιούμε όλα τα μοντέλα που προαναφέρθηκαν σε αυτό το κεφάλαιο σαν βασικά μοντέλα πρόβλεψης, τα οποία είναι πολύ καλύτερα από ένα απλό 51% στις προβλέψεις τους. Η διαδικασία είναι:

(i) Εκπαιδεύουμε όλα τα μοντέλα σε ένα κομμάτι των δεδομένων μας.

(ii) Στη συνέχεια κάνουμε προβλέψεις σε όλα τα δεδομένα μας. Για κάθε σημείο το κάθε βασικό μοντέλο παράγει μία ετικέτα (αν είναι ανωμαλία 1 αλλιώς 0)

(iii) Για κάθε σημείο έχουμε μια 5-άδα $(X_1, X_2, X_3, X_4, X_5)$ όπου τα $X_i \in \{0, 1\}$ συμβολίζουν την ετικέτα που δίνει ο αλγόριθμος i στο σημείο αυτό.

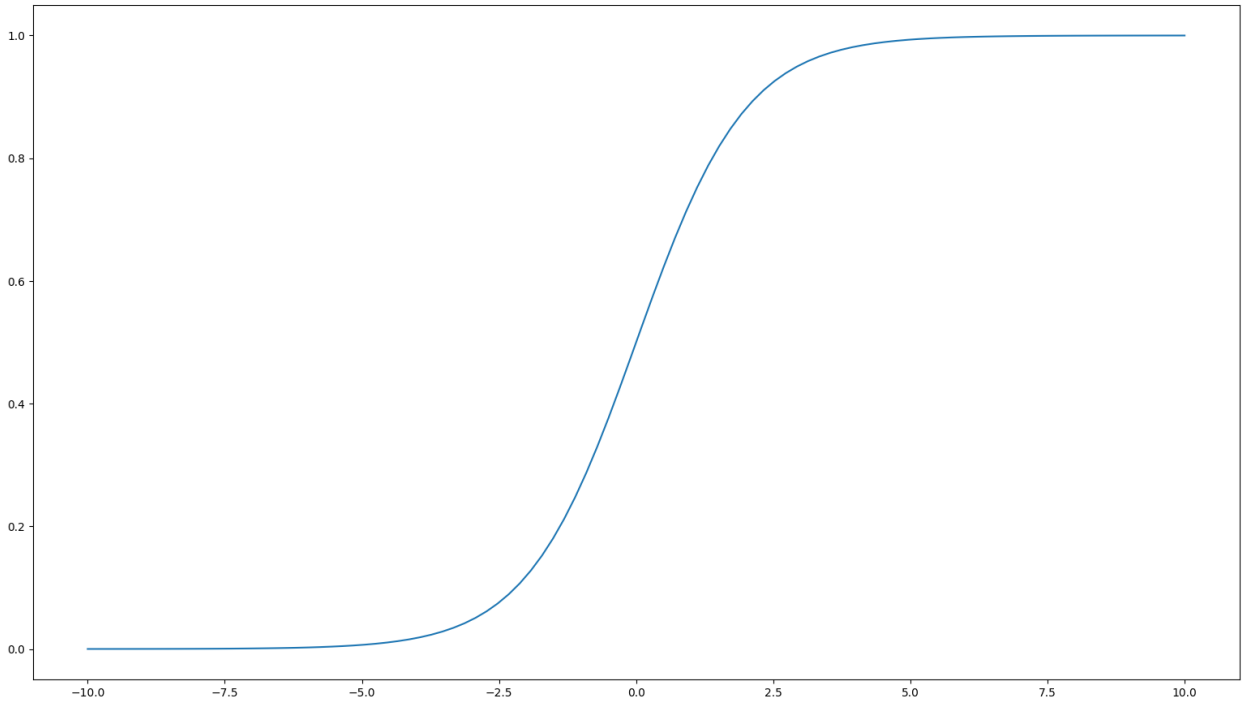
(iv) Αν τα 1 είναι περισσότερα από τα 0 τότε το σημείο χαρακτηρίζεται ως ανωμαλία.

Σύμφωνα με αυτόν τον αλγόριθμο αν ένα μοντέλο αδυνατεί να προσαρμοστεί σε κάποιο σύνολο δεδομένων ενώ οι άλλοι αλγόριθμοι τα πηγαίνουν καλύτερα τότε το σημείο δεν θα χαθεί και αν είναι ανωμαλία θα εντοπιστεί. Λόγω της φύσης των ανωμαλιών τα σύνολα που διαθέτουμε έχουν αρκετές διαφορές μεταξύ τους. Αυτό θα είχε ως αποτέλεσμα ένας αλγόριθμος μόνος του να μπορούσε να δει κάποια μοτίβα σε κάποια σύνολα ενώ να αποτύγχανε σε κάποια άλλα. Η μέθοδος του Majority Voting λύνει αυτό το πρόβλημα χρησιμοποιώντας τη γνώση από 5 διαφορετικά μοντέλα. Ο αλγόριθμος αυτός μπορεί να χρησιμοποιηθεί με μια παραλλαγή και να δίνουμε ένα βάρος στις απαντήσεις των βασικών αλγορίθμων, έπειτα ανάλογα το βάρος θα

βγαίνει το τελικό συμπέρασμα. Εμείς επιλέξαμε να κρατήσουμε σε κάθε αλγόριθμο να δίνουμε το ίδιο βάρος οπότε όλες οι απαντήσεις μετράνε το ίδιο.

3.4 Logistic Regression

Η μέθοδος αυτή είναι πολύ δημοφιλής στην κατηγοριοποίηση. Εδώ θα χρησιμοποιήσουμε αυτή τη μέθοδο στο περιβάλλον της ημι-επιβλεπόμενης μάθησης όπου αρχικά θα εκπαιδεύσουμε τα βασικά μοντέλα και στη συνέχεια θα εκπαιδεύσουμε ένα logistic regression μοντέλο πάνω στα σκορ ανωμαλίας που έχουμε εξάγει από τα μοντέλα. Στην ουσία θέλουμε το μοντέλο μας να μάθει να συνδυάζει τη γνώση που του δίνουν οι βασικοί αλγόριθμοι ώστε να μπορεί να προβλέψει αν ένα σημείο είναι ανωμαλία.



Σχήμα 3.1: Sigmoid

Το logistic regression μοντέλο χρησιμοποιεί την σιγμοειδή (Sigmoid) συνάρτηση Σχ. 3.1 η οποία είναι $\sigma(x) = \frac{1}{1+\exp(-x)}$. Ο αλγόριθμος εκτιμάει την πιθανότητα $\hat{p} = \sigma(X^T\theta)$ και η πρόβλεψη είναι η

$$\hat{y} = \begin{cases} 0, & \hat{p} < 0.5 \\ 1, & \hat{p} \geq 0.5 \end{cases}$$

Η ensemble μέθοδος λοιπόν εκτελείται σε δύο βήματα.

- Αρχικά παίρνουμε ένα κομμάτι των δεδομένων για εκπαίδευση και το χωρίζουμε σε δύο

υποσύνολα ξένα μεταξύ τους έστω A και B

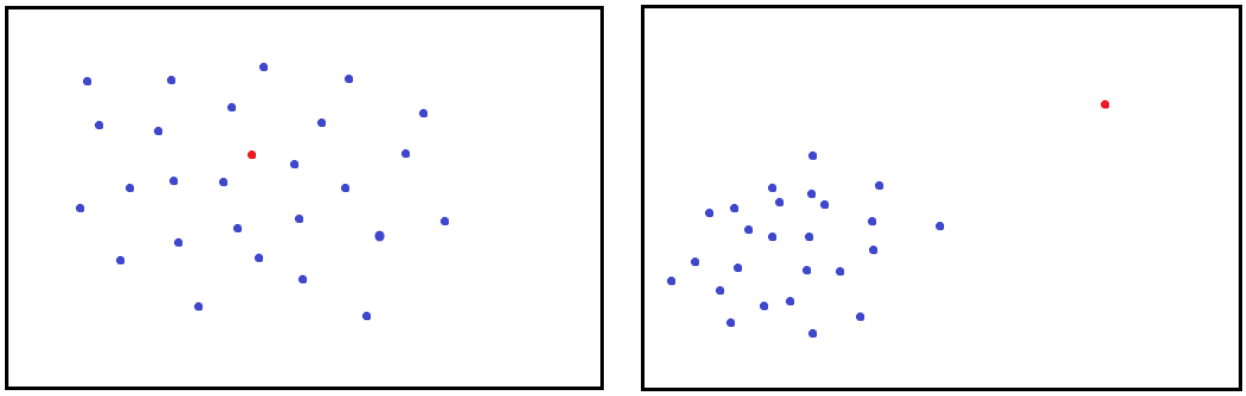
- Στη συνέχεια εκπαιδεύουμε τα βασικά μοντέλα στο σύνολο A χωρίς να δουν καθόλου το σύνολο B. Αυτό γίνεται για να μην υπάρχει διαρροή πληροφορίας ανάμεσα στα μοντέλα και στο logistic regression.
- Έπειτα κάνουμε προβλέψεις στο σύνολο B με τα βασικά μοντέλα και φτιάχνουμε ένα σύνολο B'. Έτσι το σύνολο μας αποτελείται από στοιχεία τα οποία έχουν ως διαστάσεις τις προβλέψεις των βασικών μοντέλων.
- Τέλος εκπαιδεύουμε το logistic regressor στα στοιχεία του B' δίνοντάς του ταυτόχρονα και τις πραγματικές ετικέτες τους (για αυτό το λόγο στην τεχνική αυτή μιλάμε για ημιεπιβλεπόμενη μάθηση).

Τελικά το logistic regression θα έχει μάθει να βασίζεται στις προβλέψεις των βασικών μοντέλων. Ωστόσο κρατώντας ξεχωριστά την εκπαίδευση των βασικών μοντέλων και την εκπαίδευση του logistic regression καταφέρνουμε ουσιαστικά να δώσουμε στις προβλέψεις κάθε βασικού μοντέλου ένα βάρος. Συνδυαστικά δηλαδή στο τέλος αν ένα μοντέλο δεν είναι τόσο αξιόπιστο, το logistic regression δεν θα λαμβάνει πολύ την άποψη του σε αντίθεση με την προηγούμενη μέθοδο που κάθε μοντέλο είχε την ίδια αντιμετώπιση με τα υπόλοιπα (δίνοντας βάρος 1 σε όλα).

Η ανωμαλία τελικά προκύπτει από την πιθανότητα που εξάγεται μετά την πρόβλεψη σύμφωνα με αυτό που δείξαμε παραπάνω. Δηλαδή ότι υπολογίζεται κάτω από 0.5 είναι φυσιολογικό σημείο ενώ ότι υπολογίζεται πάνω από 0.5 είναι ανώμαλο σημείο. Ανάλογα την περίπτωση θα μπορούσαμε να πειράξουμε το φράγμα του 0.5 για να μετακινήσουμε το όριο απόφασης. Αυτό θα μας έδινε περισσότερα False Positives ή False Negatives αν μειώναμε ή αυξάναμε το φράγμα. Για παράδειγμα αν πρόκειται για την αντιμετώπιση νοσημάτων σε έναν πληθυσμό θα θέλαμε ιδανικά να μην χαρακτηρίσει το μοντέλο μας κάτι φυσιολογικό ενώ στην πραγματικότητα δεν είναι. Στο παράδειγμα που δώσαμε ένα χαμηλό φράγμα θα ταίριαζε καλύτερα διότι είναι πολύ επιβλαβές να περάσει κάτι επικίνδυνο. Επιπλέον με την ανθρώπινη συμμετοχή θα μπορούσαν να ελέγχονται οι ανωμαλίες (δεδομένου ότι είναι λίγες) και να αποφασίζεται τελικά αν ήταν ανωμαλία ή όχι από τον άνθρωπο. Αντίθετα εάν έχουμε ένα ηλεκτρονικό κατάστημα και οι ανωμαλίες εκεί είναι κακόβουλοι χρήστες τότε με ένα πολύ χαμηλό φράγμα να κάναμε δύσχρηστο το σύστημά μας το οποίο θα είχε ως αποτέλεσμα την απομάκρυνση του κόσμου.

3.5 Feature Bagging

Η μέθοδος αυτή έχει προταθεί από τους Lazarevic και Kumar [27] ως μία μέθοδος ανίχνευσης ανωμαλιών σε χρονοσειρές πολλών διαστάσεων. Οι συγγραφείς κάνουν αναφορά σε αλγορίθμους οι οποίοι προσπαθούν να ανιχνεύσουν ανωμαλίες σε δεδομένα πολύ μεγάλων διαστάσεων ενώ ταυτόχρονα θεωρούν ότι είναι άσκοπο να γίνεται μία τέτοια προσπάθεια διότι η σημασία της ομοιότητας (ή της απόστασης) χάνει το νόημά της όσο αυξάνεται ο αριθμός των διαστάσεων. Λόγω του μεγάλου πλήθους τον διαστάσεων τα δεδομένα είναι πολύ μακρυά μεταξύ τους. Αυτό σημαίνει ότι όσο μεγαλώνουν οι διαστάσεις τόσο πιο πολύ απομακρύνονται τα σημεία το ένα από το άλλο και τόσο όλα τα σημεία απομονώνονται που τελικά θεωρούνται όλα ανωμαλίες. Έτσι προτείνεται η μέθοδος Feature Bagging η οποία παίρνει ένα υποσύνολο των διαστάσεων ώστε να αποφύγει το φαινόμενο αυτό.



(α') Η ανωμαλία δεν ξεχωρίζει από τα δεδομένα (β') Η ανωμαλία ξεχωρίζει από τα δεδομένα

Σχήμα 3.2: Δύο διαφορετικές προβολές των δεδομένων σε δύο διαστάσεις

Από την άλλη αν θεωρήσουμε ότι οι περισσότερες διαστάσεις εισάγουν θόρυβο στο σύνολό μας, τότε είναι πιο πιθανό να βρεθούν οι ανωμαλίες μέσα από ένα υποσύνολο των διαστάσεων. Έτσι λοιπόν ο αλγόριθμος αυτός μπορεί να χρησιμοποιηθεί ώστε να μπορέσει να εντοπίσει τις ανωμαλίες κοιτάζοντας κάποιες από τις διαστάσεις και όχι όλες. Στο Σχ. 3.2 βλέπουμε αυτή ακριβώς τη λογική, στην πρώτη εικόνα φαίνεται ότι δεν μπορούμε να ξεχωρίσουμε την ανωμαλία από τα υπόλοιπα δεδομένα, στη δεύτερη εικόνα όμως, βλέπουμε μία προβολή των δεδομένων σε δύο διαστάσεις όπου η ανωμαλία απομονώνεται.

Είναι γνωστό ότι οι βασικοί αλγόριθμοι στις ensemble τεχνικές πρέπει να είναι ικανά μοντέλα αλλά και να έχουν μια διαφορετικότητα μεταξύ τους ώστε να μπορέσουν να συνθέσουν ένα αποδοτικό ensemble μοντέλο. Η ικανότητα των μοντέλων φυσικά παίζει σημαντικό ρόλο, αλλά και η διαφορετικότητα διότι αν τα μοντέλα είναι διαφορετικά τότε δεν θα κάνουν τα ίδια λάθη. Το οποίο σημαίνει ότι ένα λάθος που κάνει ένα μοντέλο θα καλύπτεται από τα υπόλοιπα. Με αυτή

τη μέθοδο πετυχαίνουμε επιπλέον να δώσουμε διαφορετικά σύνολα δεδομένων στα μοντέλα, ως εκ τούτου κάθε μοντέλο αποκτάει ιδιαίτερα χαρακτηριστικά. Έτσι ο αλγόριθμος αυτός πετυχαίνει να διαφοροποιεί τα βασικά τους μοντέλα. Ο κλασικός αλγόριθμος bagging δεν καταφέρνει να δώσει ένα αποδοτικό ensemble από βασικούς αλγόριθμους που βασίζονται στα τοπικά χαρακτηριστικά των δεδομένων. Αυτό διότι τέτοιοι αλγόριθμοι δεν είναι ευαίσθητοι σε δειγματοληπτικές μεθόδους. Ωστόσο με τον feature bagging πετυχαίνουμε να διαφοροποιηθούμε από τον κλασικό bagging στο γεγονός ότι αντί να επιλέγουμε σημεία, επιλέγουμε features, στο οποίο είναι πολύ ευαίσθητοι αυτοί οι αλγόριθμοι, εφόσον η επιλογή των features παίζει σημαντικό ρόλο στον υπολογισμό αποστάσεων.

Η διαδικασία για την υλοποίηση αυτής της μεθόδου είναι η εξής:

- Αρχικά παίρνουμε ως δεδομένο ένα σύνολο δεδομένων $\mathcal{S} = \{X_1, X_2, ...X_n\}$ όπου $X_i \in \mathbb{R}^d$, με d να είναι το πλήθος των διαστάσεων των X_i.
- Στο πρώτο βήμα της διαδικασίας επιλέγουμε το πλήθος των βασικών αλγόριθμων που θέλουμε να χρησιμοποιήσουμε. Εμείς εδώ έχουμε επιλέξει όλοι οι αλγόριθμοι να έχουν την ίδια αρχιτεκτονική. Έστω T το πλήθος τους, για κάθε έναν από αυτούς εκτελούμε τα επόμενα βήματα.
  - Επιλέγουμε έναν τυχαίο αριθμό N_t από μία ομοιόμορφη κατανομή ανάμεσα στο $\lfloor d \rfloor$ και $(d-1)$. Εδώ το t συμβολίζει τον $t-$ οστό αλγόριθμο.
  - Στη συνέχεια επιλέγουμε χωρίς επανάθεση τυχαία N_t features και δημιουργούμε ένα σύνολο από features $\mathcal{F}$
  - Εκπαιδεύουμε τον αλγόριθμο εντοπισμού ανωμαλιών Alg_t στα δεδομένα κρατώντας μόνο τα χαρακτηριστικά $\mathcal{F}$.
  - και παίρνουμε για κάθε σημείο το σκορ ανωμαλίας του αλγόριθμου t AnomalyScore_t
- Τελικά για κάθε σημείο έχουμε T σκορ ανωμαλίας (ένα για κάθε αλγόριθμο),
  $\text{AnomalyScore}_1, \text{AnomalyScore}_2, ..., \text{AnomalyScore}_T$
  και επιλέγουμε το τελικό σκορ ανωμαλίας εφαρμόζοντας μία συνάρτηση συλλογής. Δηλαδή
  $\text{AnomalyScore} = \text{Agg}(\text{AnomalyScore}_1, \text{AnomalyScore}_2, ..., \text{AnomalyScore}_T)$.

Η επιλογή της συνάρτηση συλλογής παίζει και αυτή ένα ρόλο στην απόδοση του αλγόριθμου. Μία πολύ γνωστή είναι η Breadth-First η οποία επιλέγει το στοιχείο με το μεγαλύτερο σκορ του πρώτου αλγόριθμου σαν ανωμαλία, στη συνέχει επιλέγει το μεγαλύτερο σκορ του δεύτερου

αλγόριθμου σαν ανωμαλία εώς ότου φτάσει στον τελευταίο αλγόριθμο. Έπειτα πηγαίνει στο δεύτερο μεγαλύτερο στοιχείο του τελευταίου αλγόριθμου και το επιλέγει σαν ανωμαλία και εκτελεί το ίδιο μέχρι να γυρίσει στον πρώτο. Η διαδικασία αυτή γίνεται εώς ότου φτάσουμε ένα συγκεκριμένο αριθμό/ποσοστό ανωμαλιών. Ο τρόπος αυτός αν και είναι πολύ δημοφιλής έχει την εξής αδυναμία, θα βρίσκει πάντα ανωμαλίες ακόμα και αν το σύνολο δεδομένων έχει ελάχιστες καμία, ή ακόμα και αν τα σκορ ανωμαλίας είναι πολύ μικρά, μέχρι να πετύχει τον απαιτούμενο αριθμό. Άλλες συναρτήσεις είναι είναι ο μέσος όρος και το άθροισμα. Εμείς εδώ για τα πειράματά μας έχουμε επιλέξει το majority voting το οποίο χρησιμοποιείται και από τους Random Forest. Τελικά μετά την επιλογή ενός φράγματος έχουμε τον εντοπισμό των ανωμαλιών.

3.6 Feature Bagging with Rotation

Η μέθοδος η οποία προτείνουμε ως μια καλή λύση στον εντοπισμό ανωμαλιών για χρονοσειρές πολλών μεταβλητών είναι μία γενίκευση της προηγούμενης μεθόδου. Η ιδέα έχει προκύψει από τη γνωστή μέθοδο κατηγοριοποίησης Rotation Forest [28] η οποία επεκτείνει τα Random Forest. Ωστόσο δεν έχει υλοποιηθεί κάτι παρόμοιο στο πλαίσιο του εντοπισμού ανωμαλιών σε χρονοσειρές πολλών μεταβλητών (τουλάχιστον κατά την δική μας έρευνα).

Για να περιγράψουμε τη μέθοδο αυτή θα πρέπει πρώτα να κάνουμε μια περιγραφή του αλγόριθμου PCA. Ο PCA (Principal Component Analysis) είναι ένας αλγόριθμος γνωστός στη βιβλιογραφία για τη μείωση των διαστάσεων [29]. Έστω n η διαστάσεις τότε ο αλγόριθμος προσπαθεί να βρει $k \leq n$ n-διάστατα διανύσματα τα οποία είναι ορθογώνια μεταξύ τους και ταυτόχρονα περιγράφουν/αντιπροσωπεύουν τα υπόλοιπα δεδομένα. Στη συνέχεια τα δεδομένα προβάλλονται στο χώρο που δημιουργούν αυτά τα διανύσματα έτσι προκύπτει μία αναπαράσταση των δεδομένων σε έναν χώρο λιγότερων διαστάσεων χωρίς να χάνεται πληροφορία. Η διαδικασία είναι η εξής:

(i) Αρχικά τα δεδομένα κανονικοποιούνται. Η κανονικοποίηση των δεδομένων βοηθάει στο να μην επιτρέψει τις πολύ υψηλές τιμές να κυριαρχήσουν και να επηρεάσουν το αποτέλεσμα.

(ii) Για κάθε διάσταση υπολογίζεται η μέση τιμή.

(iii) Στη συνέχεια υπολογίζεται ο πίνακας συσχέτισης (covariance matrix) για όλες τις διαστάσεις. Δηλαδή $\text{COV}(X, Y) = \frac{1}{m} \cdot \sum_{i=1}^{m} (X - \mu_X)(Y - \mu_Y)$

(iv) Έπειτα υπολογίζονται τα ιδιοδιανύσματα και οι ιδιοτιμές του προηγούμενου πίνακα. Δηλαδή $\det(A - \lambda I) = 0$. Τα διανύσματα αυτά λέγονται και Principal Components ενώ οι τιμές

αναπαριστούν τη ”σημαντικότητα”.

(v) Βάζουμε σε φθίνουσα σειρά τα διανύσματα ανάλογα με τις ιδιοτιμές. Δηλαδή πρώτα μπαίνει το διάνυσμα με τη μεγαλύτερη ιδιοτιμή, μετά αυτό με την αμέσως επόμενη μεγαλύτερη ιδιοτιμή κοκ. Τα διανύσματα που επιλέγονται είναι τα k πρώτα.

(vi) Τέλος παίρνουμε τον πίνακα με τα ιδιοδιανύσματα και πολλαπλασιάζουμε τον ανάστροφο του, με τον αρχικό πίνακα δεδομένων ώστε να γίνει η μετατροπή, άρα:

$$\begin{pmatrix} \text{new data} \end{pmatrix} = \begin{pmatrix} p_1 & \dots & p_k \end{pmatrix}^T \begin{pmatrix} \text{data} \end{pmatrix}$$

Έτσι λοιπόν καταφέρνουμε να μειώσουμε τις διαστάσεις από n σε k. Αν λοιπόν κρατήσουμε όλα τα ιδιοδιανύσματα και δεν αφήσουμε κανένα απέξω είναι σαν να στρέφουμε τους άξονες των δεδομένων. Αυτή η ιδέα δίνει τη δυνατότητα στο μοντέλο να δει τα δεδομένα από άλλη ”οπτική” γωνία επιτρέποντάς του να αυξήσει την απόδοσή του. Επιπλέον χρησιμοποιείται το Feature Bagging ώστε να έχουμε όλα τα πλεονεκτήματα που είδαμε προηγουμένως. Αρχικά επιλέγουμε τυχαία ένα υποσύνολο από το σύνολο των Features στη συνέχεια εφαρμόζεται ο αλγόριθμος PCA σε αυτό το υποσύνολο. Παίρνουμε τα νέα δεδομένα και εκπαιδεύουμε έναν βασικό αλγόριθμο. Αυτό γίνεται για κάθε βασικό αλγόριθμο που πρόκειται να συνθέσει το ensemble μοντέλο. Στο τέλος μαζεύονται όλα τα αποτελέσματα και εφαρμόζεται μια συνάρτηση συλλογής. Παρακάτω δίνουμε αναλυτικά τη διαδικασία που ακολουθεί ο αλγόριθμος.

Έστω $\mathcal{X}$ μεγέθους N το σύνολο δεδομένων που διαθέτουμε για να κάνουμε την εκπαίδευση του μοντέλου. Τα στοιχεία του $\mathcal{X}$ έχουν n διαστάσεις οπότε ο πίνακας των δεδομένων, έστω A, είναι διάστασης $N \times n$. Έστω D το σύνολο των Features και L το πλήθος τον αλγορίθμων που θα χρησιμοποιηθούν. Για κάθε βασικό αλγόριθμο έχουμε:

- Επιλέγουμε έναν τυχαίο αριθμό F_l από μία ομοιόμορφη κατανομή ανάμεσα στο $\lfloor n \rfloor$ και $(n-1)$. Εδώ το l συμβολίζει τον $l-$ οστό αλγόριθμο.
- Στη συνέχεια επιλέγουμε χωρίς επανάθεση τυχαία N_t features και δημιουργούμε ένα σύνολο από features $F \subseteq D$ και παίρνουμε μόνο αυτά τα features για τα δεδομένα μας.
- Επιλέγουμε ένα τυχαίο αριθμό K και χωρίζουμε το σύνολο F σε K υποσύνολα. Τα υποσύνολα αυτά μπορεί να είναι ξένα ή και όχι. Για να αυξήσουμε όμως τη διαφορετικότητα ανάμεσα στους βασικούς αλγόριθμους επιλέγουμε να είναι ξένα. Έτσι τα δεδομένα κάθε υποσυνόλου αποτελείται από $M = n/K$ features.

- Επιλέγουμε ένα ποσοστό του συνόλου $\mathcal{X}$. Στο επόμενο βήμα θα εξηγήσουμε γιατί.
- Πάνω σε αυτά τα υποσύνολα εφαρμόζουμε το PCA και αποθηκεύουμε τους συντελεστές των principal components, έστω $a_{l,k}^1, ...a_{l,k}^M$, όπου με k συμβολίζουμε το k-οστό υποσύνολο από τα K. Στο προηγούμενο βήμα έχουμε επιλέξει ένα ποσοστό επί του συνόλου. Αυτό έγινε για ενισχύσουμε τη διαφορετικότητα ανάμεσα στους συντελεστές σε περίπτωση που τυχαία επιλεγούν ίδια features ανάμεσα στους βασικούς αλγόριθμους. Έτσι ακόμα και ίδιες επιλογές να γίνουν από δύο βασικούς αλγόριθμους οι συντελεστές θα είναι διαφορετικοί, και τελικά θα πετύχουμε τη διαφορετικότητα που θα θέλαμε.
- Από το προηγούμενο βήμα παράγουμε έναν πίνακα στροφής για τον βασικό αλγόριθμο l:

$$R_l = \begin{pmatrix} a_{l,1}^1, ...a_{l,1}^M & [0] & \cdots & [0] \\ [0] & a_{l,k}^1, ...a_{l,k}^M & \cdots & [0] \\ \vdots & \vdots & \ddots & \vdots \\ [0] & [0] & \cdots & a_{l,K}^1, ...a_{l,K}^M \end{pmatrix}$$

- Στο προηγούμενο βήμα πρέπει να προσέξουμε ότι εφόσον τα K-υποσύνολα έχουν επιλέξει τυχαία τα features τα a_i δεν είναι στη σειρά. Οπότε σε αυτό το βήμα τοποθετούμε με τα a_i με τη σειρά που παρατηρούνται τα αντίστοιχα features. Έστω R_l^{ordered} ο πίνακας με τη σωστή σειρά.
- Τελικά παίρνουμε τα νέα δεδομένα πολλαπλασιάζοντας $X_{\text{new}}^l = X_l R_l^{\text{ordered}}$ όπου X_l είναι τα αρχικά δεδομένα που θα δίναμε στον αλγόριθμο l πριν την εφαρμογή του PCA. Ο αλγόριθμος l εκπαιδεύεται σε αυτά τα δεδομένα.
- Σαν τελικό βήμα της διαδικασίας αφού έχουμε εκπαιδεύσει L βασικούς αλγόριθμους, εφαρμόζουμε μία συνάρτηση συλλογής. Εμείς αποφασίσαμε να ακολουθήσουμε το παράδειγμα των Random Forest και να χρησιμοποιήσουμε το Majority Voting.

Στη διαδικασία αυτή θα παρατηρήσουμε ότι το ποσοστό των στοιχείων που κρατάμε για να πετύχουμε τη διαφορετικότητα ανάμεσα στους βασικούς αλγόριθμους και ο αριθμός K είναι παράμετροι του ensemble.

Ο αλγόριθμος αυτός όπως θα δούμε και παρακάτω φαίνεται να δίνει πολύ καλά αποτελέσματα. Ένα μειονέκτημα που έχει είναι οι πολλές παράμετροι που πρέπει να επιλέξουμε. Ο αλγόριθμος πέρα από τις παραμέτρους των βασικών αλγόριθμων που πρέπει να επιλεχθούν, έχει επιπλέον την παράμετρο K και το ποσοστό των στοιχείων. Επιπλέον το μεγαλύτερο μειονέκτημά του είναι ο μεγάλος χρόνος εκπαίδευσης. Ο χρόνος εκπαίδευσής του εξαρτάται από 2 παράγοντες. Αρχικά έχουμε το χρόνο εκπαίδευσης κάθε βασικού υπο-μοντέλου και μετά έχουμε το χρόνο που χρειάζεται ο PCA ο οποίος είναι εξαιρετικά χρονοβόρος. Έτσι ο χρόνος που χρειάζεται ένα τέτοιο ensemble να εκπαιδευτεί είναι αρκετά μεγάλος.

3.7 Stacking Feature Bagging with Rotation Models

Όπως είπαμε και πριν στην τεχνική που ονομάζεται Stacking εκπαιδεύονται όλα τα μοντέλα παράλληλα. Με βάση την έξοδο των μοντέλων στο πρώτο βήμα εκπαιδεύεται ένα μετα-μοντέλο. Για την υλοποίηση της τεχνικής αυτής εμείς χρησιμοποιούμε στο πρώτο στάδιο μοντέλα που έχουν κατασκευαστεί από τη μέθοδο Feature Bagging with Rotation και στη συνέχεια χρησιμοποιούμε έναν Logistic Regressor ως μετα-μοντέλο. Η διαδικασία είναι η εξής:

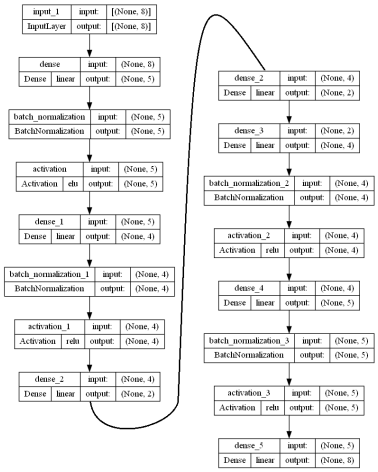
- Αρχικά επιλέγουμε ένα υποσύνολο των δεδομένων στο οποίο θα γίνει η εκπαίδευση. Το υποσύνολο αυτό το χωρίζουμε σε δύο υποσύνολα, A και B. Στο A θα εκπαιδευτούν τα επιμέρους μοντέλα ενώ στο B θα εκπαιδευτεί το μετα-μοντέλο.
- Επιλέγουμε έναν αριθμό $k > 1$
- Για κάθε βασικό μοντέλο κατασκευάζονται k_n μοντέλα με την τεχνική Feature Bagging with Rotation όπως περιγράψαμε νωρίτερα (όπου n είναι το n-οστό βασικό μοντέλο).
- Κάθε μοντέλο εκπαιδεύεται και παράγει ένα score ανωμαλίας.
- Με αυτά τα επιμέρους μοντέλα κάνουμε προβλέψεις σε δεδομένα που δεν έχουν εκπαιδευτεί τα μοντέλα δηλαδή στο σύνολο B. Έτσι για κάθε σημείο παίρνουμε $k_n * n$ features (χαρακτηριστικά) όπου k_n ο αριθμός που επιλέξαμε στην αρχή και n ο αριθμός των αρχικών μας μοντέλων. Έτσι κατασκευάζεται ένα νέο σύνολο από το το σύνολο δεδομένων B, έστω B' και για κάθε στοιχείο του έχουμε ότι $b_i = (SA_{k_n,n})_{k_n,n} \in \mathbb{R}^{k_n \times n}$, όπου με SA συμβολίζουμε το σκορ ανωμαλίας που έχει δώσει το k_n-οστό μοντέλο το οποίο έχει παραχθεί από το n-οστό βασικό μοντέλο μέσω της τεχνικής Feature Bagging with Rotation.
- Στο σύνολο B' εφαρμόζουμε επιβλεπόμενη μάθηση εκπαιδεύοντας έναν Logistic Regressor.

Αυτή η μέθοδος συνδυάζει όλα τα παραπάνω και το αποτέλεσμα είναι να πάρουμε ένα πολύ ισχυρό μοντέλο όπως θα δούμε και παρακάτω. Η αδυναμία αυτής της τεχνικής είναι ότι διατηρεί όλες τις δυσκολίες από τις παραπάνω τεχνικές και ο χρόνος εκπαίδευσης είναι κατά πολύ μεγαλύτερος.

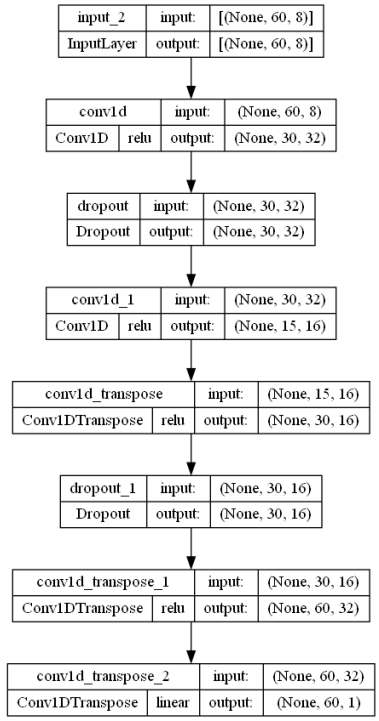
3.8 Αρχιτεκτονικές, Υπερπαράμετροι και Μεθοδολογία

Η εκπαίδευση για κάθε μοντέλο γίνεται σε κάθε σύνολο δεδομένων ξεχωριστά όπως και η πρόβλεψη. Όλα τα μοντέλα εκπαιδεύονται στα ίδια δεδομένα. Τα τελικά αποτελέσματα περιλαμβάνουν τις επιδόσεις ως προς όλα τα σύνολα δεδομένων. Επίσης έχουμε βγάλει αποτελέσματα και για τα βασικά μοντέλα ώστε να μπορεί να γίνει η σύγκριση μεταξύ τους. Η εκπαίδευση είναι χωρίς-επίβλεψη (εκτός από τις τεχνικές που περιλαμβάνουν έναν Logistic Regressor που το μοντέλο εκπαιδεύεται με μερική-επίβλεψη) οπότε κανένα μοντέλο δεν γνωρίζει την πληροφορία ”anomaly”. Η πληροφορία ”anomaly” χρησιμοποιείται στο τέλος ώστε να υπολογίσουμε την απόδοση κάθε αλγόριθμου. Οι αρχιτεκτονικές και οι παράμετροι που χρησιμοποιούνται στα απλά μοντέλα εφαρμόζονται και στα μοντέλα που συνθέτουν τις ensemble τεχνικές. Τέλος στα δεδομένα δεν γίνεται κάποια προεπεξεργασία εκτός από κάποιου είδους κανονικοποίησης και δεν εφαρμόζεται η τεχνική του ”ανακατέματος” (shuffling) διότι αυτό θα αφαιρούσε από τα δεδομένα μας την έννοια της χρονικής σειράς. Για να συγκρίνουμε τους αλγόριθμους υπολογίζεται το συνολικό F1 σκορ και το συνολικό AUC (ως προς όλα τα δεδομένα) για τον κάθε αλγόριθμο ενώ ταυτόχρονα παρέχουμε και τον confusion πίνακα.

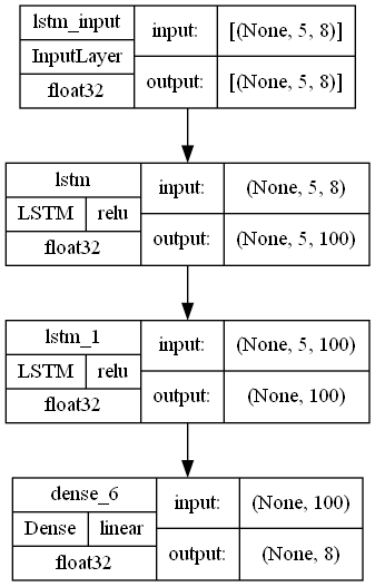
Έστω ότι τα σύνολα δεδομένων (κάθε σύνολο είναι μία χρονοσειρά πολλών μεταβλητών) μπορεί να αναπαρασταθεί με $(t_0, t_1, ...t_n)$ Το σύνολο εκπαίδευσης κάθε συνόλου δεδομένων είναι από τη στιγμή t_0 έως τη στιγμή t_{399} και το σύνολο του τεστ είναι $(t_{400}, t_{401}, ...t_n)$. Στις περιπτώσεις που έχει χρησιμοποιηθεί ο Logistic Regressor το σύνολο εκπαίδευσης των επιμέρους μοντέλων αποτελείται από τα σημεία $(t_0, t_1, ...t_{399})$, το σύνολο εκπαίδευσης με ημι-επιβλεπόμενη μάθηση του Logistic Regressor αποτελείται από τα σημεία $(t_{400}, t_{401}, ...t_{599})$ και τα υπόλοιπα σημεία χρησιμοποιούνται σαν τεστ (Το μέσο μέγεθος των χρονοσειρών είναι περίπου 1100 χρονικές στιγμές). Τα βασικά μοντέλα που επιλέξαμε είναι Autoencoder, Convolutional Autoencoder, LSTM, LSTM Autoencoder και LSTM Variational Autoencoder και στο Σχ. 3.3 μπορούμε να δούμε τις αρχιτεκτονικές τους.



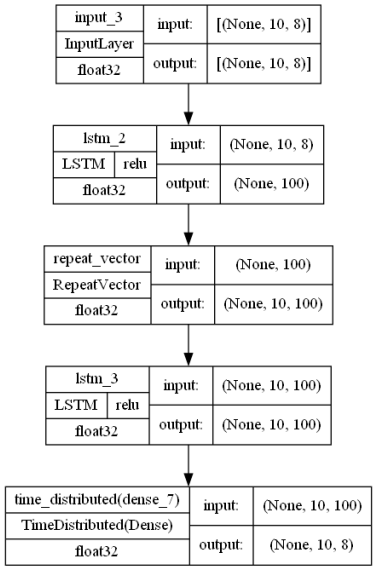
(α') Autoencoder



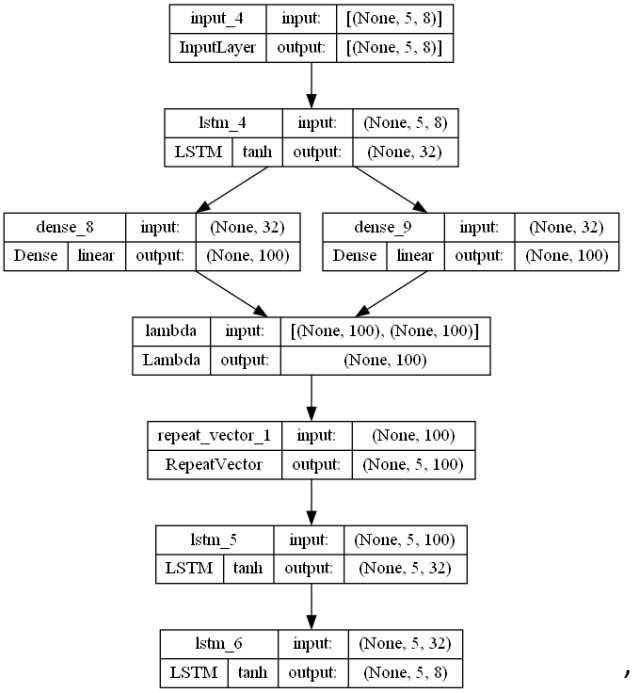
(β') Convolutional Autoencoder



(γ') LSTM



(δ') LSTM Autoencoder



(ε') LSTM Variational Autoencoder

Σχήμα 3.3: Αρχιτεκτονικές Βασικών Μοντέλων

Το μέγεθος του παραθύρου που χρησιμοποιούμε είναι 60 χρονικές στιγμές για τον Convolutional Autoencoder, 5 για τον LSTM, 10 για τον LSTM Autoencoder και 5 για τον LSTM Variational Autoencoder. Για την τεχνική Feature Bagging χρησιμοποιήσαμε 17 estimators. Για την τεχνική Feature Bagging with Rotation χρησιμοποιήθηκαν 17 estimators, το K το θέσαμε 2 και το ποσοστό των δεδομένων που χρησιμοποιήθηκαν για τον αλγόριθμο PCA ήταν στο 75%. Τέλος στο Stacking των Feature

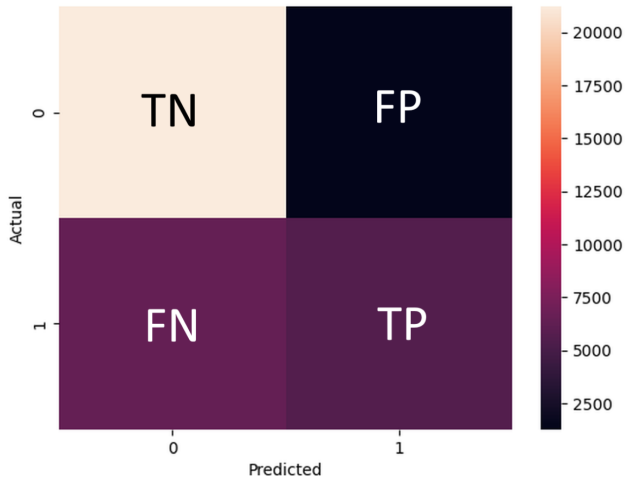
Bagging with Rotation με μετα-μοντέλο έναν Logistic Regressor χρησιμοποιήθηκαν 12 estimators για κάθε μία από τους 5 βασικούς αλγόριθμους όπου οι παράμετροι για κάθε μία από αυτές ήταν K=2 και το ποσοστό στο 80%. Να θυμίσουμε εδώ ότι οι βασικές τεχνικές και οι τεχνικές Majority Voting και Logistic Regression χρησιμοποιούν όλα τα features ενώ στις υπόλοιπες τεχνικές

Chapter 4

Αποτελέσματα

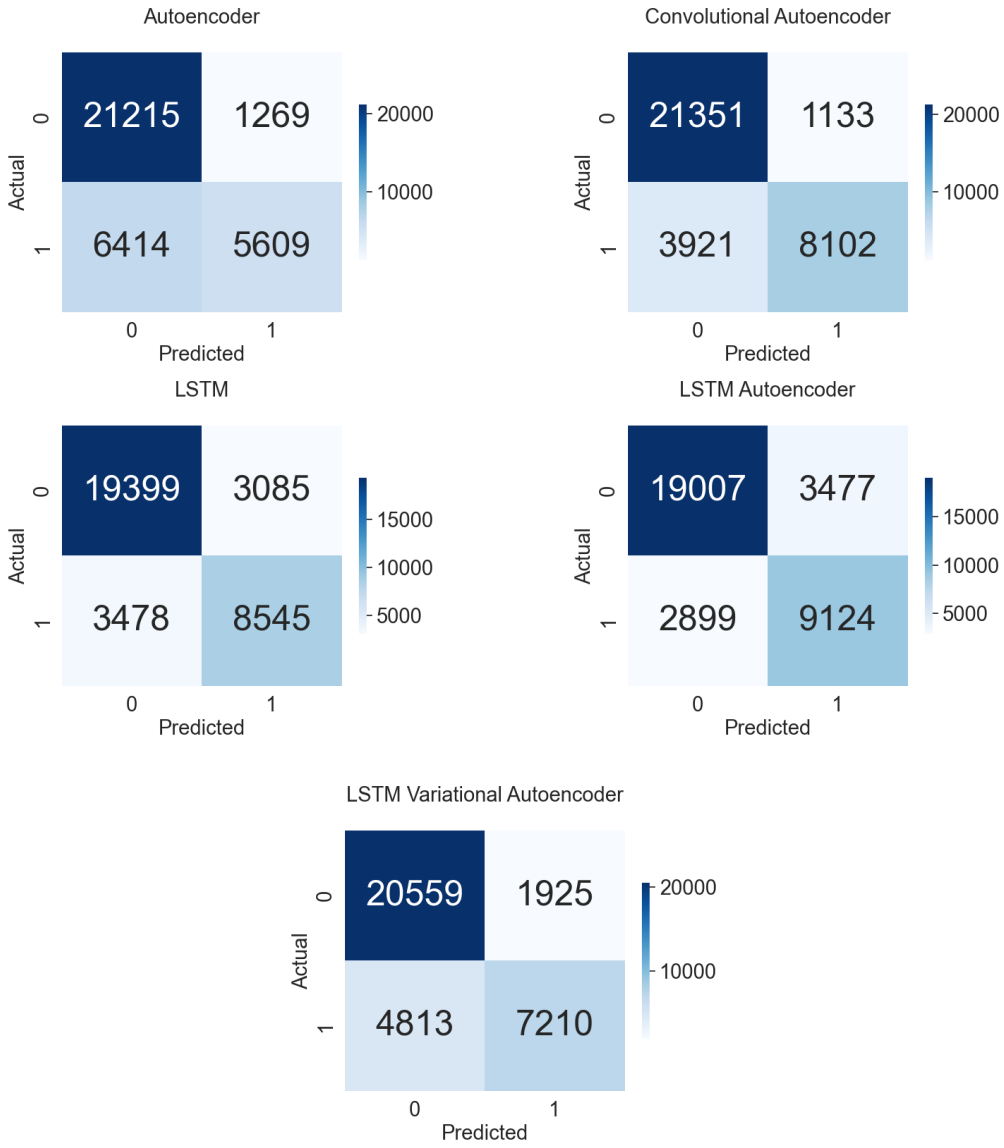
4.1 Αποτελέσματα

Οι παραπάνω τεχνικές εφαρμόστηκαν σε 31 σύνολα δεδομένων τα οποία προσομοιώνουν διαφορετικές καταστάσεις. Για κάθε ένα από αυτά τα σύνολα τα μοντέλα εκπαιδεύονται σε ένα κομμάτι τους και στη συνέχεια κάνουν τις προβλέψεις σε ολόκληρα τα σύνολα. Στο πλαίσιο του εντοπισμού των ανωμαλιών δεν μπορούμε να μετρήσουμε την απόδοση ενός μοντέλου με την κλασική μετρική "Accuracy" η οποία δίνεται από το Accuracy $= \frac{TP+TN}{TP+FP+TN+FN}$ όπου TP, FP, TN, FN είναι τα True Positives, False Positives, True Negative, False Negatives αντίστοιχα. Οπότε επιλέξαμε να μετρήσουμε τις αποδόσεις με βάση τα F1 score και το AUC (οι ορισμοί δόθηκαν σε προηγούμενο κεφάλαιο). Στη συνέχεια θα δώσουμε τα αποτελέσματα από τις τεχνικές που αναφέραμε ως προς το σύνολο των δεδομένων, πρώτα όμως, θα πρέπει να εξηγήσουμε τι είναι ένας Confusion Πίνακας.



Σχήμα 4.1: Confusion Matrix

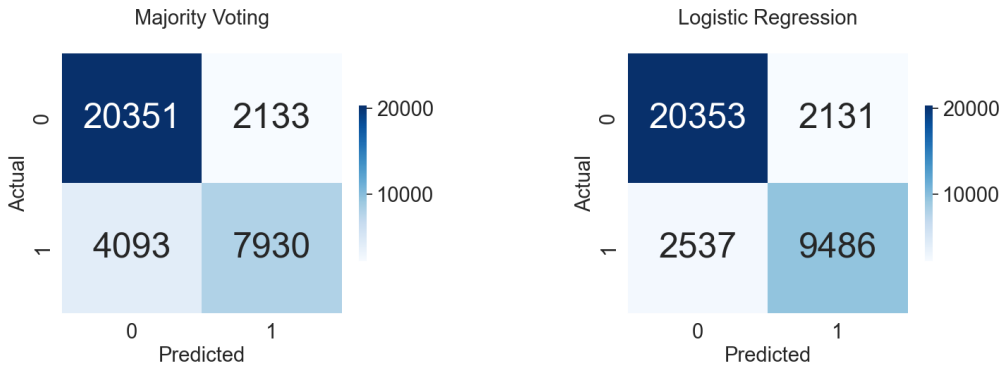
Ένας Confusion Πίνακας αποτελείται από 4 περιοχές τα True Positives, True Negative, False Positives και False Negative. Στην αριστερή πλευρά του πίνακα μπορούμε να διαλέξουμε ανάμεσα στα πραγματικά δεδομένα (πάνω γραμμή τα πραγματικά φυσιολογικά σημεία ενώ στην κάτω γραμμή έχουμε έχουμε τα πραγματικά Ανώμαλα δεδομένα). Στην κάτω πλευρά του πίνακα μπορούμε να διαλέξουμε ανάμεσα στις προβλέψεις (στην αριστερή στήλη έχουμε τις προβλέψεις των φυσιολογικών σημείων ενώ στη δεξιά τις προβλέψεις των ανώμαλων σημείων). Στο Σχ. 4.1 βλέπουμε έναν confusion matrix όπου αντί για αριθμούς έχουμε σημειώσει σε κάθε περιοχή τι δείχνει. Σημειώνουμε εδώ ότι το ιδανικό μοντέλο θα είχε στις περιοχές των FP και FN τον αριθμό μηδέν.



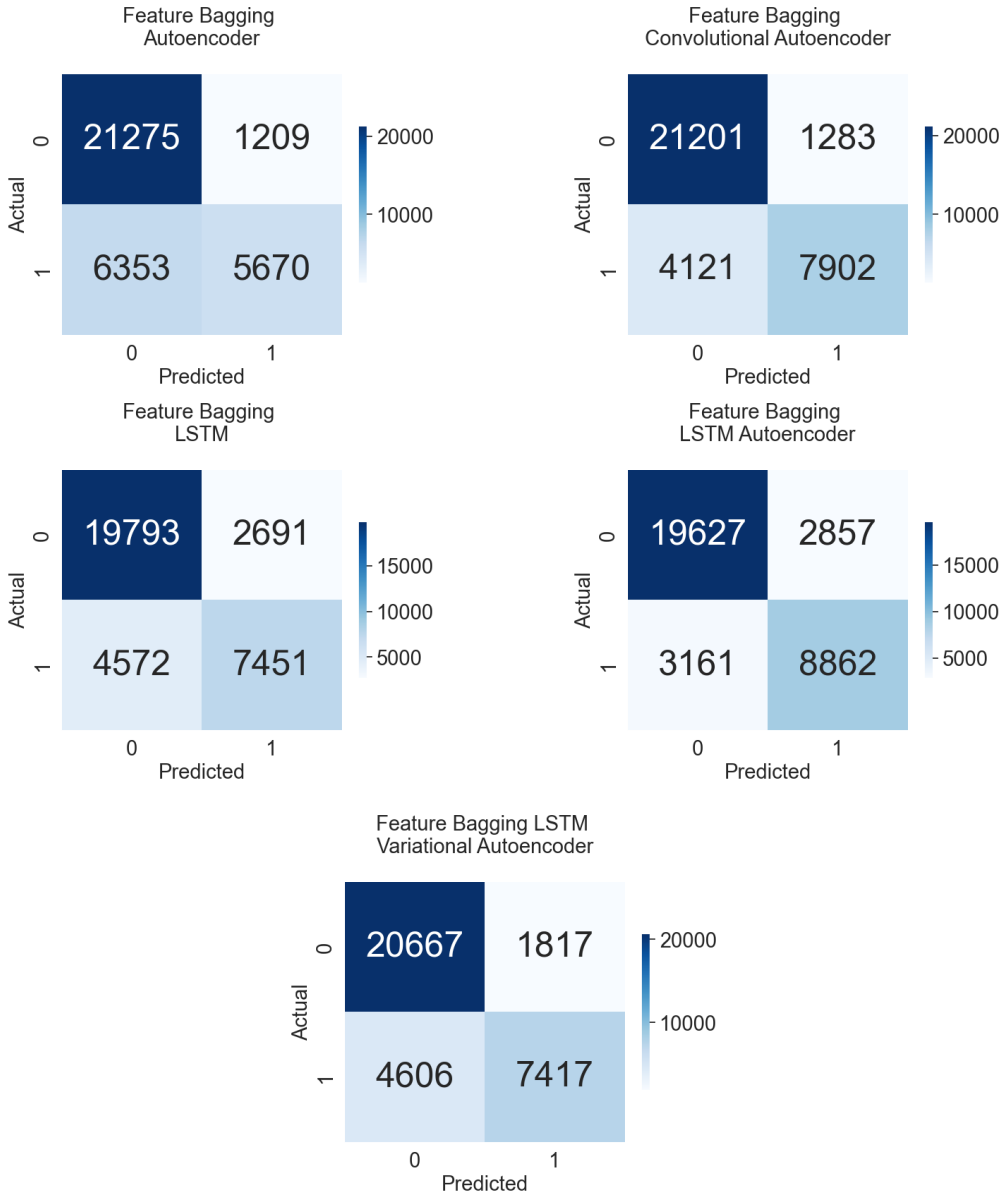
Σχήμα 4.2: Βασικά Μοντέλα

Παράλληλα με τις τεχνικές εκπαιδεύσαμε και τα βασικά μοντέλα ώστε να έχουμε ένα μέτρο σύγκρισης στη συνέχεια. Στο Στο Σχ. 4.2 βλέπουμε τα αποτελέσματα που είχαν τα βασικά

μοντέλα. Παρατηρούμε ότι τα μοντέλα πετυχαίνουν καλές επιδόσεις ειδικότερα ο convolutional autoencoder και ο LSTM autoencoder.



Σχήμα 4.3: Majority Voting - Logistic Regression



Σχήμα 4.4: Feature Bagging

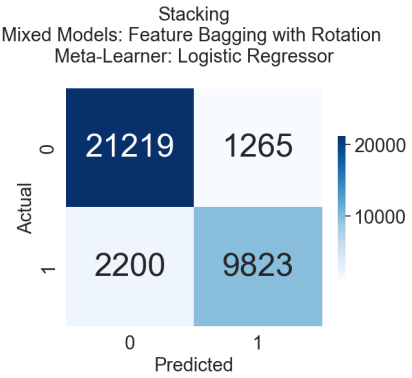
Στη συνέχεια δίνουμε τα αποτελέσματα για τις τεχνικές που αναφέραμε προηγουμένως. Ξεκινάμε με τον Majority Voting και την τεχνική όπου εφαρμόζουμε ημι-επιβλεπόμενη μάθηση με Logistic Regression Σχ. 4.3. Παρατηρούμε ότι και οι δύο αλγόριθμοι είναι πολύ κοντά στα σημεία που δεν είναι ανωμαλίες όμως διαφέρουν στα ανώμαλα σημεία. Όπως βλέπουμε ο Logistic Regression πετυχαίνει καλύτερα αποτελέσματα εκεί.

Στη συνέχεια παρουσιάζουμε τα αποτελέσματα της τεχνικής Feature Bagging Σχ. 4.4. Παρατηρούμε ότι εδώ ανάλογα με τον βασικό αλγόριθμο που έχει χρησιμοποιηθεί υπάρχει μια ποικιλία στα αποτελέσματα. Στην επόμενη ενότητα θα δούμε μία σύγκριση ανάμεσα σε κάθε βασικό αλγόριθμο και στην τεχνική Feature Bagging που του αντιστοιχεί.

Σχήμα 4.5: Feature Bagging with Rotation

Στο Σχ. 4.5 παρουσιάζουμε τα αποτελέσματα της τεχνικής Feature Bagging with Rotation

όπου με τη βοήθεια του αλγόριθμου PCA στρίβουμε τους άξονες. Παρατηρούμε και εδώ ότι ανάλογα με τον βασικό αλγόριθμο που έχει χρησιμοποιηθεί υπάρχει μια ποικιλία στα αποτελέσματα.



Σχήμα 4.6: Stacking, Mixed Models: Feature Bagging with Rotation, Meta-Learner: Logistic Regressor

Τέλος παραθέτουμε τα αποτελέσματα από το Stacking που χρησιμοποιήσαμε σε μοντέλα κατασκευασμένα από την τεχνική Feature Bagging with Rotation με μετα-μοντέλο έναν Logistic Regressor Σχ. 4.6

4.2 Συγκρίσεις

Στους παρακάτω πίνακες γίνεται μία σύγκριση ανάμεσα στα βασικά μοντέλα και τις τεχνικές. Όπως είπαμε και προηγουμένως χρησιμοποιούμε τις μετρικές F1 score και AUC για να αξιολογήσουμε τα μοντέλα. Με βάση αυτές τις μετρικές παρατηρούμε ότι η τεχνική Feature Bagging δίνει αρκετά καλά αποτελέσματα σε κάποιες περιπτώσεις ενώ η τεχνική Feature Bagging with Rotation κερδίζει σε όλες τις συγκρίσεις εκτός από την περίπτωση που το βασικό μοντέλο είναι ο LSTM. Επιπλέον ο Majority Voting είναι περίπου στη μέση (κάτι που το περιμέναμε εφόσον στηρίζεται στην πλειοψηφία) ενώ ο αλγόριθμος με το Logistic Regression κερδίζει τους βασικούς αλγόριθμους καθώς στηρίζεται στο γεγονός ότι μαθαίνει να κάνει την καλύτερη επιλογή στηριζόμενος στην πρόβλεψη των βασικών. Επιπλέον είναι ο μόνος αλγόριθμος που γνωρίζει έστω και λίγες ετικέτες από τα δεδομένα. Τέλος αξίζει να αναφερθεί ότι δεν μπορούμε να κάνουμε συγκρίσεις μεταξύ των Feature Bagging τεχνικών διότι δεν στηρίζονται στους ίδιους βασικούς αλγόριθμους. Το ίδιο συμβαίνει και με την τεχνική Feature Bagging with Rotation.

Model Title	F1 Score	AUC
Feature Bagging with Rotation Autoencoder	62.59 %	72.31 %
Feature Bagging Autoencoder	59.99 %	70.89 %
Autoencoder	59.35 %	70.50 %

Πίνακας 4.1: Autoencoder

Παρατηρούμε στους αλγόριθμους Autoencoder, LSTM Autoencoder και LSTM Variational Autoencoder ο Feature Bagging έχει καλύτερη επίδοση από το αντίστοιχο βασικό μοντέλο και ο Feature Bagging with Rotation καλύτερη επίδοση από τον Feature Bagging ενώ σε σχέση με τα βασικά μοντέλα η επίδοση βελτιώνεται 2% Πίνακας 4.1 Πίνακας 4.4 Πίνακας 4.5. Στην περίπτωση του Convolutional Autoencoder το Feature Bagging with Rotation συνεχίζει να έχει καλύτερη επίδοση όμως σε αντίθεση με τα προηγούμενα ο Feature Bagging όχι Πίνακας 4.3. Ο μόνος βασικός αλγόριθμος που έχει καλύτερη επίδοση από τις αντίστοιχες εκδόσεις του Feature Bagging και Feature Bagging with Rotation είναι ο LSTM Πίνακας 4.3.

Model Title	F1 Score	AUC
Feature Bagging with Rotation Convolutional Autoencoder	78.73 %	83.15 %
Convolutional Autoencoder	76.22 %	81.17 %
Feature Bagging Convolutional Autoencoder	74.51 %	80.00 %

Πίνακας 4.2: Convolutional Autoencoder

Model Title	F1 Score	AUC
LSTM	72.25 %	78.67 %
Feature Bagging with Rotation LSTM	70.74 %	77.49 %
Feature Bagging LSTM	67.23 %	75.00 %

Πίνακας 4.3: LSTM

Model Title	F1 Score	AUC
Feature Bagging with Rotation LSTM Autoencoder	76.41 %	81.90 %
Feature Bagging LSTM Autoencoder	74.65 %	80.50 %
LSTM Autoencoder	74.10 %	80.21 %

Πίνακας 4.4: LSTM Autoencoder

Model Title	F1 Score	AUC
Feature Bagging with Rotation LSTM Variational Autoencoder	70.14 %	77.04 %
Feature Bagging LSTM Variational Autoencoder	69.78 %	76.80 %
LSTM Variational Autoencoder	68.15 %	75.70 %

Πίνακας 4.5: LSTM Variational Autoencoder

Παρατηρούμε στους Πίνακες 4.1-4.5 ότι η τεχνική Feature Bagging with Rotation ανταποκρίνεται καλύτερα σε όλα τα βασικά μοντέλα εκτός από την περίπτωση του LSTM όπου σε αυτή την περίπτωση αποτυγχάνει και η τεχνική Feature Bagging. Εδώ να θυμίσουμε πάλι ότι η εκπαίδευση έγινε με 17 estimators για κάθε περίπτωση, με K=2 και ποσοστό δεδομένων στο 75 %.

Model Title	F1 Score	AUC
Convolutional Autoencoder	76.22 %	81.17 %
LSTM Autoencoder	74.10 %	80.21 %
LSTM	72.25 %	78.67 %
Majority Voting	71.81 %	78.23 %
LSTM Variational Autoencoder	68.15 %	75.70 %
Autoencoder	59.35 %	70.50 %

Πίνακας 4.6: Majority Voting

Ο Majority Voting έχει τοποθετηθεί στη μέση του πίνακα με μικρή διαφορά από το LSTM ενώ παράλληλα έχει πετύχει καλύτερη επίδοση πάνω από 8% από τον Autoencoder Πίνακας 4.3. Τέλος ο Logistic Regressor πετυχαίνει καλύτερα αποτελέσματα από όλους τους βασικούς αλγόριθμους Πίνακας 4.7

Model Title	F1 Score	AUC
Logistic Regression	80.25 %	84.71 %
Convolutional Autoencoder	76.22 %	81.17 %
LSTM Autoencoder	74.10 %	80.21 %
LSTM	72.25 %	78.67 %
LSTM Variational Autoencoder	68.15 %	75.70 %
Autoencoder	59.35 %	70.50 %

Πίνακας 4.7: Logistic Regression

Τέλος την καλύτερη επίδοση πετυχαίνει ο αλγόριθμος που εφαρμόζουμε το Stacking των Feature Bagging with Rotation με μετα-μοντέλο ένα Logistic Regressor Πίνακας 4.8

Model Title	F1 Score	AUC
Stacking with Logistic Regression	85.00 %	88.03 %
Convolutional Autoencoder	76.22 %	81.17 %
LSTM Autoencoder	74.10 %	80.21 %
LSTM	72.25 %	78.67 %
LSTM Variational Autoencoder	68.15 %	75.70 %
Autoencoder	59.35 %	70.50 %

Πίνακας 4.8: Stacking with Logistic Regression

Chapter 5

Συμπεράσματα

5.1 Συμπεράσματα και Περαιτέρω επέκταση της εργασίας

Στην εργασία αυτή υλοποιήσαμε διάφορες ensemble τεχνικές για την ανίχνευση ανωμαλιών σε χρονοσειρές πολλών μεταβλητών. Το πρόβλημα αυτό έχει προσεγγιστεί από διάφορους αλγόριθμους τόσο από στατιστικές τεχνικές όσο και από Μηχανικής μάθησης και βαθιάς μηχανικής μάθησης μοντέλα. Εμείς πήραμε δεδομένα κάποια βασικά μοντέλα βαθιάς μηχανικής μάθησης και στηριχθήκαμε σε αυτά ώστε να υλοποιήσουμε ensemble τεχνικές ώστε να πετύχουμε ακόμα καλύτερα αποτελέσματα. Τα ensemble μοντέλα που κατασκευάστηκαν αρχικά ήταν ένας Majority Voting ο οποίος είχε μια μέτρια επίδοση και ένας αλγόριθμος με ημι-επιβλεπόμενη εκπαίδευση όπου σαν τελικό μοντέλο χρησιμοποιήθηκε ένας Logistic Regressor. Στη συνέχεια υλοποιήθηκε ένα αλγόριθμος που ονομάζεται Feature Bagging ο οποίος μοιάζει πολύ με την τεχνική που χρησιμοποιείται στους Random Forest. Ο Feature Bagging έδωσε κάποια καλά αποτελέσματα σε κάποιες περιπτώσεις έναντι των βασικών μοντέλων. Στη συνέχεια από μια ιδέα που προέρχεται από τους Rotation Forest χρησιμοποιήθηκε ο PCA πάνω στον Feature Bagging με αποτέλεσμα να παραχθεί ένας αλγόριθμος οποίος στις περισσότερες περιπτώσεις είχε καλύτερη επίδοση από όλα τα παραπάνω και από τους βασικούς αλγόριθμους. Τέλος χρησιμοποιήθηκε μία stacking τεχνική με επιμέρους μοντέλα κατασκευασμένα από την τεχνική Feature Bagging with Rotation και μετα-μοντέλο έναν Logistic Regressor όπου πιάσαμε την καλύτερη επίδοση από όλες τις προηγούμενες τεχνικές. Οι ensemble τεχνικές είναι γνωστό ότι βελτιώνουν κατά πολύ τις επιδόσεις και λύνουν αρκετά προβλήματα. Τα παραπάνω αποτελέσματα μας δείχνουν ότι μπορούν να βοηθήσουν πάρα πολύ και στην ανίχνευση ανωμαλιών. Με τις ensemble μεθόδους μπορούμε να ξεπεράσουμε αρκετά τα απλά μοντέλα και να παραχθεί ένα υπερ-μοντέλο το οποίο θα εντο-

πίζει αρκετά αξιόπιστα τις ανωμαλίες. Ειδικότερα ο αλγόριθμος Feature Bagging with Rotation μπόρεσε να έχει 2% καλύτερη επίδοση από τα βασικά μοντέλα σε κάποιες περιπτώσεις ενώ ο Feature Bagging όταν ήταν καλύτερος από ένα απλό μοντέλο είχε καλύτερη επίδοση περίπου 1% σε μία περίπτωση ενώ στις άλλες δύο είχε λιγότερο από 1%. Τελικά όταν εφαρμόσαμε το Stacking πετύχαμε περίπου 10 % καλύτερη επίδοση σε σχέση με τα βασικά μοντέλα.

Από τα παραπάνω μπορούμε να πούμε ότι αξίζει η περαιτέρω επέκταση της εργασίας και η προσπάθεια ανίχνευσης ανωμαλιών με ensemble τεχνικές καθώς φαίνεται ότι τα ensemble μοντέλα μπορούν να δώσουν λύσεις. Οι περισσότερες από τις τεχνικές χρειάζονται κάποια συνάρτηση συγκέντρωσης των αποτελεσμάτων οπότε θα μπορούσε αυτό να αποτελέσει αντικείμενο έρευνας ώστε με την κατάλληλη συνάρτηση να μπορούν να συνδυαστούν τα αποτελέσματα των βασικών μοντέλων πιο αποτελεσματικά. Οι Shahzad και Lavesson έχουν κάνει μία έρευνα [30] στην οποία ερευνούν παραλλαγές του Majority Voting και φαίνεται ότι ο Majority Voting πετυχαίνει καλύτερα αποτελέσματα σαν αλγόριθμος αλλά θα ήταν ενδιαφέρον να ερευνηθεί αν μπορούν οι παραλλαγές να χρησιμοποιηθούν σαν συναρτήσεις επιλογής των τεχνικών που περιγράψαμε και αν μπορούν να δώσουν καλύτερα αποτελέσματα.

Επιπλέον τα σύνολα δεδομένων αποτελούνται από λίγα features, δηλαδή οι χρονοσειρές πολλών μεταβλητών αποτελούνται από λίγες μεταβλητές (συνολικά 8), αυτό σημαίνει ότι οι αλγόριθμοι αυτοί θα μπορούσαν να ερευνηθούν και σε παραπάνω διαστάσεις. Όσο αυξάνονται οι διαστάσεις τόσο πιο πολύ χάνεται το νόημα της ανωμαλίας κάτι που κάνει δύσκολο τον εντοπισμό, όμως πιστεύουμε ότι οι αλγόριθμοι αυτοί θα μπορούσαν να αντεπεξέλθουν πολύ καλά ειδικότερα οι αλγόριθμοι Feature Bagging και Feature Bagging with Rotation οι οποίοι μπορούν να εκμεταλλευτούν τις πολλές διαστάσεις.

Ένα άλλο κομμάτι σημαντικό για αυτούς τους αλγόριθμους είναι το κομμάτι της ταχύτητας. Παρατηρήσαμε ότι όσο αυξάνεται η πολυπλοκότητα τόσο αυξάνεται και ο χρόνος που χρειάζονται τα μοντέλα για να εκπαιδευτούν. Αυτό σημαίνει ότι είναι αδύνατον να πάμε σε πολύ μεγάλο αριθμό βασικών μοντέλων. Ο μεγάλος αριθμός βασικών μοντέλων θα μας έδινε τη δυνατότητα να έχουμε καλύτερες επιδόσεις όμως κάτι τέτοιο ήταν αδύνατον δεδομένου του χρόνου που έπαιρνε για παράδειγμα ένα Feature Bagging with Rotation να εκπαιδευτεί. Έτσι λοιπόν αξίζει να διερευνηθεί παραπάνω το πώς θα μπορούσε αυτό να γίνει σε πολύ λιγότερο χρόνο ώστε να μπορέσει ένας τέτοιος αλγόριθμος να αποτελείται από πολλά μικρά βασικά μοντέλα.

Βιβλιογραφία

[1] W. Wei, *Time Series Analysis: Univariate and Multivariate Methods*. Time Series Analysis: Univariate and Multivariate Methods, Pearson Addison Wesley, 2006.

[2] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*. OTexts, 2018.

[3] D. M. Hawkins, *Identification of outliers*, vol. 11. Springer, 1980.

[4] F. E. Grubbs, "Procedures for detecting outlying observations in samples," *Technometrics*, vol. 11, no. 1, pp. 1–21, 1969.

[5] M. Markou and M. Singh, "Novelty detection: A review—part 1: Statistical approaches," *Signal Processing*, vol. 83, pp. 2481–2497, 12 2003.

[6] E. M. Knorr and R. T. Ng, "Finding intensional knowledge of distance-based outliers," in *Vldb*, vol. 99, pp. 211–222, Citeseer, 1999.

[7] M. Braei and S. Wagner, "Anomaly detection in univariate time-series: A survey on the state-of-the-art," *arXiv preprint arXiv:2004.00433*, 2020.

[8] M. Markou and S. Singh, "Novelty detection: a review—part 2:: neural network based approaches," *Signal processing*, vol. 83, no. 12, pp. 2499–2521, 2003.

[9] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, 07 2009.

[10] V. Chandola, *Anomaly detection for symbolic sequences and time series data*. University of Minnesota, 2009.

[11] R. J. Hyndman, A. B. Koehler, R. D. Snyder, and S. Grose, "A state space framework for automatic forecasting using exponential smoothing methods," *International Journal of Forecasting*, vol. 18, no. 3, pp. 439–454, 2002.

[12] E. Keogh and J. Lin, “Clustering of time-series subsequences is meaningless: Implications for previous and future research,” *Knowledge and Information Systems*, vol. 8, pp. 154–177, 08 2005.

[13] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD’96, p. 226–231, AAAI Press, 1996.

[14] F. T. Liu, K. M. Ting, and Z.-H. Zhou, “Isolation forest,” in *2008 Eighth IEEE International Conference on Data Mining*, pp. 413–422, 2008.

[15] V. Vapnik and A. Y. Chervonenkis, “A class of algorithms for pattern recognition learning,” *Avtomat. i Telemekh*, vol. 25, no. 6, pp. 937–945, 1964.

[16] B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt, “Support vector method for novelty detection,” in *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS’99, (Cambridge, MA, USA), p. 582–588, MIT Press, 1999.

[17] T. Chen and C. Guestrin, “XGBoost,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, aug 2016.

[18] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” 2015.

[19] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *CoRR*, vol. abs/1512.03385, 2015.

[20] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Computation*, vol. 9, pp. 1735–1780, 11 1997.

[21] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” 2013.

[22] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, “Wavenet: A generative model for raw audio,” 2016.

[23] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using RNN encoder-decoder for statistical machine translation,” *CoRR*, vol. abs/1406.1078, 2014.

[24] I. D. Katser and V. O. Kozitsin, “Skoltech anomaly benchmark (skab).” https://www.kaggle.com/dsv/1693952, 2020.

[25] A. Elhalwagy and T. Kalganova, “Hybridization of capsule and lstm networks for unsupervised anomaly detection on multivariate data,” *arXiv preprint arXiv:2202.05538*, 2022.

[26] T. Pranavan, T. Sim, A. Ambikapathi, and S. Ramasamy, “Contrastive predictive coding for anomaly detection in multi-variate time series data,” *arXiv preprint arXiv:2202.03639*, 2022.

[27] A. Lazarevic and V. Kumar, “Feature bagging for outlier detection,” in *booktitle=Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining,*, vol. 21, pp. 157–166, 01 2005.

[28] J. Rodríguez, L. Kuncheva, and C. Alonso, “Rotation forest: A new classifier ensemble method,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, pp. 1619–30, 11 2006.

[29] J. Han, M. Kamber, and J. Pei, *Data mining concepts and techniques, third edition*. Morgan Kaufmann Publishers, 2012.

[30] R. K. Shahzad and N. Lavesson, “Comparative analysis of voting schemes for ensemble-based malware detection,” *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 4, no. 1, pp. 98–117, 2013.