A Work Project, presented as part of the requirements for the Award of a Master's Degree in Management from the Nova School of Business and Economics.

Implementing Machine Learning for Data Breach Detection

José Gama

33238

Work project carried out under the supervision of:

Patrícia Xufre

22-05-2020

Table of Contents

Abstract
Privata.ai is a User and Entity Behavior Analytics (UEBA) application used for the detection of data breaches in an organization. By tracking down the usual access to personal and sensitive data, it becomes much easier to detect an outlier. These anomalies could result in a real threat to the company's data security and must, therefore, be promptly detected and addressed. This paper focuses on the managerial challenges that arise from the increasing threat of data breaches and how machine learning could help in protecting organizations from them. For this purpose, large part of the challenge came from understanding the unique specificities of these attacks and finding an appropriate machine learning method to detect them. Given the fact that the data used to train the models was randomly generated, the results should be taken with caution. Nevertheless, the models used for this paper should be taken as a basis for the future development of the software.

Keywords: Machine Learning; User and Entity Behavior Analytics; Anomaly Detection

1. Introduction

1.1 Data Breaches and their effect on firms

Over recent years, data breaches have become an increasingly large threat for companies around the globe (Choong et al., 2017; Liu et al., 2018). Broadly defined, a data breach constitutes the intentional or unintentional leak of secure or private information to an untrusted environment (Cheng et al., 2017). For a company, these events could have severe consequences that could amount to very large costs.

These incidents can have various causes and even though data breaches are hard to prevent, they're not so difficult to anticipate (Irwin, 2020). They usually result from improper encryption and stolen credentials, one of the simplest and most common ways for cyber attackers to infiltrate a system when it is made of predictable and easy to decrypt passwords. They also come from configuration errors, when a software's technical vulnerabilities are exploited by attackers. This usually happens after a software provider discovers a vulnerability, urging their clients to apply a fix. If not applied promptly, attackers may take this opportunity to exploit the weakness in order to steal customer data. Furthermore, malwares are also a way with which cyber attackers can access confidential data. All it takes is to install a malware in a piece of software that contains a known vulnerability and exploit the rewards (Cheng et al., 2017; Romanosky et al., 2014).

Additionally, these data breaches can also come from inside the company. On the intentional side, employees may be tempted by the financial gain of selling data on the dark web or feel resentment towards the company, accessing the organization's systems for nefarious purposes. On the unintentional side, employees may just commit a mistake that results in a data breach, such as including the wrong person when sending an email, attaching the wrong document or losing a laptop (Cheng et al, 2017; Sanzgiri et al., 2016).

In fact, in order to understand the damage these attacks can have on a company, it's important to understand what's at risk. Khan et al. (2019) explored the effect that the loss of confidentiality could have for a company. Drawing from literature review, they identified how these events could result in a loss of competitive advantage, when records containing confidential trade secrets are stolen or customers switch to a competitor due to their loss of confidence in the company (Yayla et al., 2011). They also found identity theft to be one of repercussions these events may cause, since by appropriating someone's credentials, attackers may have access to more confidential information (Roberds et al., 2009). Moreover, the theft of customer data, could be used to commit fraud or launch personalized cyber-attacks, putting both customer integrity and the firm's reputation at risk (Miller, 2018; Graham, 2019).

Data breaches' costs are, by their very nature, hard to measure, due to the fact that their harms are intangible, risk-oriented and diffuse (Solove et al., 2016). In order to understand what costs a data breach can cause, Algarni et al. (2016) divided them into different categories. There are *incident investigation costs*, which are expenses aimed at assisting the organization in discovering the data breach, like forensic costs or audit and consulting costs. They also identified *crisis management costs*, which are costs derived from informing the public that personal information has been stolen,

such as notification activities and credit monitoring. Furthermore, there are *regulatory and industry sanction costs*, which depend on how compliant the organization is, potentially facing fines when not abiding to the established regulation. *Lawsuit costs* appear when a party that suffered damage from the data stolen decides to sue the organization. Finally, *opportunity costs*, which result from lost business opportunities and damaged reputation that come from the occurrence of these attacks.

In order to control the costs associated with these leaks, it is fundamental to follow incumbent data security regulation. Regulations outline the type of data that requires a notification for when a breach occurs (Karyda et al., 2016). At the moment, these regulations vary widely across the globe. In most countries, like Argentina, Belarus, Costa Rica, Egypt, Japan, Macau, Malaysia, Madagascar, Mauritius, Panama, Russia, and Saudi Arabia, data breach notifications aren't even mandatory (Kiener-Manu, 2020).

In the European Union, regulation on data breaches is both stricter and more thorough than most countries (Nieuwesteeg et al., 2018). The General Data Protection Regulation (GDPR), implemented in May 2018, introduced the general legislation that establishes rules for the protection and privacy of personal data. Under Article 33(1) GDPR[1], an organization must notify a data breach to the relevant supervisory authority under seventy-two hours since having been aware of it. This law is applicable in case any EU citizen's personal data is breached independently of whether or not the organization is inside the EU (Article 3(1) GDPR[1]; Publications Office, 2018; Tikkinen-Piri et al., 2018).

Despite all these efforts by legislators to keep these attacks from happening, the number of data breaches has kept rising with 2019 being reported as the year with the most data breaches yet (Sobers, 2020). According to IBM Security (2020), a data breach cost, on average, $3,92 million to an organization in 2019.

From the current state of things, there is both of an increasing number of these cyber-attacks and more regulation created to prevent them from happening. This shows that even though lawmakers are becoming more aware of the importance of this issue, the effectiveness of the legislation can still be put into question (Nieuwesteeg et al., 2018; Karyda et al., 2018). With continued efforts in trying to control this problem, it can be determined that large part of the blame can be attributed to the company management itself. In fact, most of these attacks happen because organizations are still negligent in the way they secure their data, leaving the door open for many of these incidents to occur (Hodge, 2020).

[1] Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation), OJ 2016 L 119/1.

1.2 Privata AI

Due to the rising need for companies to be able to properly combat these attacks, new data-oriented firms have emerged with the purpose of fixing this issue. Blockbird Ventures is a Portuguese firm founded in 2018 by José Figueiredo and Carlos Faria with the mission of disrupting businesses with innovative blockchain and cybersecurity solutions. The firm has gathered media attention for trying to bridge the gap between college students who have studied and are more acquainted with this type of technology and actual organizations, bringing their projects and business ideas to the forefront in the real-world (Bourbon, 2018).

From activities such as Ethereum Networks, end-to-end Decentralized Applications and Solidity Smart Contracts, the firm has also developed Privata.ai, an internally developed API-based software for protecting applications against data breaches. Unlike more well-known UEBA services with a higher market share, the software has unique features that differentiate it from competitors. Most UEBA services collect data from organizations at the network level using sources such as logs, network packets, flows, files, alerts or threat feeds (Shashanka et al., 2016; Lin et al., 2016). Data generated from these sources, although coming in large amounts, tends to contain a lot of noise, making it difficult to then fit it into a model that will provide meaningful conclusions. Privata.ai, on the other hand, focuses exclusively in data protection for web applications, only tracking down the access to data on a relational database containing personally identifiable information (PII) that should be monitored more closely. User access to these databases should be followed more closely since, according to Verizon (2019), on the one hand, 29% of data breaches in 2019 involved the use of stolen credentials by criminal hackers and, on the other hand, 34% of them were caused by internal actors. Having less data sources will mean that a lower amount of it will be collected but a lower amount of noise will also make it easier to create more meaningful models afterwards (Atla et al., 2011).

Due to the fact that machine learning hasn't been deployed on the software yet, the version that is now available in the market is a minimum viable product whose main function comes from flagging suspicious user behavior through alerts. The client can, therefore, configure rules for what they consider may be suspicious user behavior, such as excessive access to particular data or accessing sensitive data at a given time, and receive a notification whenever a user demonstrates that sort of behavior (see Appendix 1). As the product gets developed, the intent is to implement a machine learning solution that will automatically detect this anomalous behavior without the need for the client to configure their own rules.

1.3 Research Question

The main question posed by this paper is stated as:

"What is a proper machine learning framework for Privata.ai to be able to detect data breaches, given the user behavior collected by the software?"

The machine learning approach taken by Privata.ai is a very important issue for the firm, in the sense that it is the next step for the software they are developing. The better their algorithm works, the more likely it will be for the start-up to distinguish itself on the market. Thus far, the company has positioned itself in a different way from other User and Entity Behavior Analytics applications with a larger market share by focusing exclusively the very specific problem of detecting data breaches on web applications through machine learning. By having a more niche approach, the firm expects to respond to current business demands on protection against data theft. In this sense, the development of a competent machine learning system in this software is crucial for the product's success and company growth.

This paper aims at addressing the most efficient cybersecurity tools organizational management can use in order to fight data breaches, following specific advanced analytics techniques that would be useful to detect suspicious behavior occurring in a firm.

In Section 2, the Literature Review delves into research made on the development of similar software as well as illustrating how machine learning can be used in data breach detection. Section 3 regards the data and methodology used to train the models and Section 4 describes in depth the entire framework developed to tackle this issue. Section 5 showcases the results from the entire model and, finally, Section 6 comprises the conclusions and future research that come from this thesis.

2. Literature Review

User and Entity Behavior Analytics is a type of cybersecurity process that monitors the normal behavior patterns of entities within an enterprise. In this way, it becomes possible to flag suspicious behavior, which is to say behavior that falls out of the pattern set up by the users themselves. By investigating the anomalous behavior that has occurred, it becomes possible for organizations to detect much more easily potentially threatening behavior and protect themselves from possible harm (Lin, 2016; Brooke, 2020). This process has clear advantages for companies when it comes to data breach prevention and detection coming from firm insiders (Salitin et al., 2018), since the suspicious behavior detected by it could result in a leak of data to an untrusted environment.

While the benefits are clear, there are many challenges when it comes to implementing a proper of a UEBA system. At the center of a well-built platform is a well-tuned statistical analysis system. In order to get that, it is necessary to collect a large amount of data coming from various indicators from users, assets, applications or network locations (Lin, 2016). The data coming from all these sources will showcase a certain distribution and by flagging data points that fall outside this distribution is possible to detect when an anomalous behavior has occurred (Shashanka et al., 2016).

It is in this task of finding outliers within data that machine learning can become quite useful to integrate in a UEBA solution. Machine learning (ML) can be defined as "the science (and art) of programming computers so they can learn from data" (Géron, 2019). Machine Learning Algorithms (MLAs) can be used in order to recognize

patterns in data (Bishop, 2016; Stamp, 2017) and, henceforth, detect the data points that fall outside of the pattern (Escalante, 2005).

When dealing with a machine learning task, first, it is important to understand what type of data we are dealing with. Considering the available data being collected from a UEBA service is not labeled and has had barely any human supervision (Lin, 2016), the problem presented is classified as unsupervised learning (Ghahramani, 2003). With this type of learning, algorithms identify commonalities within the data, giving an insight about which data points can be considered outliers, the same ones representing anomalous behavior (Campos et al, 2016).

Even though unsupervised learning can be quite helpful, on its own, using a single modeling technique will probably not be enough for a UEBA system to work. In fact, the outliers given by these algorithms on their own will present a high false positive rate, which is to say a lot of events that don't constitute dangerous behavior will be flagged as such. This is due to the fact that human behavior is shown to have large fluctuations over time, making it easy for unusual but perfectly harmless behavior to be identified as a threat (Lin, 2016). In order to more accurately flag suspicious behavior, it is important to add to these algorithms another layer of complexity. This may come by combining data analytics with expert-driven knowledge of a particular field to make it more intuitive and easier to explain. Additionally, it may also be useful to score the risk of an event based on how far from normality it is and use customer feedback about risk scores to adjust and update the algorithms to work more appropriately next time they are implemented (Lukashin et al., 2019; Shashanka et al., 2016).

A large problem in implementing a successful method to this software comes from the fact that companies do not reveal which algorithms they are working with. Henceforth, it was necessary to research various methods that would correspond to the particularities that these sorts of attacks have. Data breaches are, by their very nature, hard to detect, taking, according to IBM (2020), 206 days for firms to do so. Studies have found that this is due to most companies' IT professionals having misplaced confidence in their systems, believing their data is well-secured, which causes data breaches to be detected months after the attack. The theft of secure or private information to an untrusted party can happen over a period of time instead of at a single point in time, making periodicity a factor to take into account. Additionally, even though a particular action taken by a user may not raise cause for concern, the collection of various seemingly harmless actions may threaten the security of an organization (Stolfo et al., 2008). This is aligned with the attacker's intentions of keeping their behavior as seamless as possible (Mookerjee et al., 2011). Below we try to relate these characteristics of a data breach to machine learning methods, so that we can find an appropriate approach in order to fix this problem.

First, we need to consider the importance of tracking the time at which each event in our system has happened. Henceforth, timestamps of all the events that constitute access to that data have an enormous importance in the proper defense against those attacks, as they allow for the periodicity of the actions to be taken into account (Ho et al., 2018; Grier, 2011). For this reason, anomaly detection in time series data should provide the most effective methods to successfully recognize such attacks.

Furthermore, there is also a need to identify which type of outlier is the one that corresponds to the sort of behavior that constitutes a data breach. From a classical point of view, an outlier is "an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism" (Hawkins, 1980). In a more modern context, other definitions of the event have been suggested in literature as well as characterizing different types of outliers as a way to better detect them. There are point outliers, which are pieces of data that showcase a rare behavior at a particular point in time when compared either to the other values in the time series (global outlier) or to its neighboring points (local outlier). Additionally, there are subsequent outliers which constitute consecutive points in time that become uncommon when their joint behavior is analyzed, even though each observation by itself is not necessarily a point outlier (Blázquez-García, 2020). With this in mind, it is important to understand that a data breach can happen both at a single time instance or over a period of time (Stolfo et al., 2008), so a method or a collection of methods that are able to detect both these types of outliers would be decisive to properly detect data breaches.

Taking all these factors into account, there have been methods developed that fulfill these specificities and can be useful in the successful detection and prevention of these attacks. Depending on whether the data is univariate or multivariate, different models can be deployed. On the one hand, univariate time series are ordered sets of observations where each one is recorded at a particular point in time (Moral et al.,2003). Multivariate time series, on the other hand, are ordered sets of k-dimensional vectors, each one recorded at a particular point in time (Reinsel, 2003). Blázquez-García, 2020). Considering that when building a machine learning pipeline for data breach detection multiple factors should be taken into account, the modeled time series tends to be multivariate.

For our purposes, this paper will build on the method developed by Munir et al. (2019), due to the fact that it is the one developed in current literature that tackles anomaly detection in a time series that can detect both point and subsequent outliers in multivariate data.

3. Data and Methodology

The goal of this work project is to develop an algorithm that can take data containing all the accesses into a relational database containing sensitive data as the input and present the anomalies in that model as the output, considering these outliers should constitute a data breach. The dataset used to test the model has been produced randomly by Blockbird Ventures representatives and not by an actual firm where the software has been installed. This means that the results produced by the model should be taken with caution and have limited interpretability. Nevertheless, the data generated does provide important insights as we get to test the features in the dataset Privata.ai would create once it had been installed and follow through with a methodology that would be effective in detecting data breaches once real data can be tested.

Each event in our given dataset corresponds to one access to the relational database that is being monitored. Each row illustrates a query that was run, characterized by different features that describe what the action consisted of. The features in our dataset can be described in the dictionary below. As far as the feature "Group", in particular, the prototype dataset was configured to be applicable to a medical facility, having the employees divided into four groups: Medics, Admins, Staff, and Nurses.

Feature	Description	Type
Query_id	Identification for each query	Numerical
Transaction_id	Table Primary Key	Numerical
User	User Id	Numerical
Group	User's role in the company	Categorical
Action	Action that is taken through the query (Read, Create, Update, Delete)	Categorical
Transaction_date	Date at which the action took place	Numerical
Timestamp	Timestamp at which the action took place (Date and Time)	Numerical
Table	Table that was accessed	Categorical
Column	Column that was accessed	Categorical
Row_count	Number of rows within the column that were accessed	Numerical

Table 1 – Data Dictionary

All the features in this dataset are GDPR compliant, as they are related to a user's behavior and, therefore, don't constitute personal data as defined in Article 4(1) GDPR[1]. With that in mind, the firm is legally entitled to collect this information and we can use these features to build our model.

3.1 Data Curation

The process of creating a useful machine learning algorithm has as its first step adapting the data that has been extracted in a way that can be applied to our model. As Zixuan Zhang (2019) establishes, only features that are relevant should be fitted into the model. Given that various of the features presented in the dataset present no meaning in terms of actual user behavior or are just redundant given that they present information already given in another feature, we will select only the ones that are actually useful to our model. Henceforth, the columns "Query_id", "Transaction_id" and "User" will be deleted due to the fact that they are merely identifying the event but contain no information on the user behavior and the column "Transaction_date" will be deleted due to the fact that the information in that column is already presented in "Timestamp". The lower number of features will make the model simpler and easier to understand (Azevedo, 2019).

Furthermore, there was a need to adapt our data into a time series, so that we could then use it appropriately in the model. As we had seen before, a time series is a set of observations described by one or more variables ordered by the time of its occurrence. In order to do that, we indexed the feature "Timestamp", which caused each data point to be ordered from the earliest to the latest event.

Afterwards, the current data had to be adapted into a data type that could be fitted into the model. As it is shown in Table 1, many of the features that will be fitted into the model are categorial, i.e. type of data that may be classified into groups by labels like gender or race (Agresti, 2003). However, this data type cannot be fitted into a machine learning model, having the need to be transformed prior to the training in order to be useful (Sarkar, 2019). For this purpose, we converted the labels that described these features into discrete numbers each one corresponding to a label, using the preprocessing technique "LabelEncoder" from the Scikit-Learn library.

The next step is the standardization of the numbers. Once the data is all numerical, there is still the problem that different features present values in widely different scales. This causes the model to run slower and be more unstable, as the algorithm's gradient may take much longer and oscillate much more before finding its global/ local minimum, damaging the learning process and obtaining worse results (Jaitley, 2019; Brownlee, 2019). In order to prevent this, we scaled each data point by converting them to z-scores, removing their mean and scaling to unit variance. This made all the features in our dataset more comparable and easier to fit into the model. After all these steps, the data preparation was done and the dataset was ready to be fit into a model.

4. Models

The goal of this paper is to describe an approach through which Privata.ai can properly detect data breaches using machine learning. Even though the data that we have to train our models is artificial, the processes described should be effective in the task of identifying suspicious outliers. In order to do that, we describe a three-model process through which we can thoroughly investigate where the outliers are and create a model that we'll be useful in detecting them with new data.

i. A clustering algorithm was built using our entire dataset, in order to group the data into different subsets depending on their behavior. By doing this, we will be able to analyze whether the number of clusters created by the algorithm is actually equal to the number of worker groups, demonstrating how the group each user is a part of affects his/her behavior;
ii. A feature selection technique in order to analyze which features are the most relevant and defining of these data points. This will be useful for feature engineering, since understanding which features are the most relevant and how they correlate with each other can help create more powerful features and build a model with better results;
iii. A more complex and robust outlier detection algorithm that can be used to determine both point and subsequent outliers. This model will be applied individually for each group, since it is expected that different worker groups would demonstrate different behavior from each other.

Each of these sub-models is further explained in the subsections below.

4.1 Clustering Algorithm

The first hypothesis that we want to prove is that the behavior of a user is different depending on the role that he/she undertakes in the company. In order to do that, we can create an algorithm that groups these different data points into different clusters, based on their similarities. This way, we can understand whether the amount of employee groups defined in our dataset is the same as the number of clusters, illustrating the pre-conceived notion that different employees would showcase different behavior.

Furthermore, this clustering algorithm also allows us to understand the statistical distribution of these data points in the entire dataset and compare that to the statistical distribution for each employee group. Given that outliers in terms of user behavior should be identified in the case a user demonstrates a strange behavior compared to others in the same group, it wouldn't make sense to simply spot the outliers from an entire dataset consisting of different users who would naturally represent very different behavior. In our case specifically, it would be like comparing the behavior of a doctor to that of an administrator. In statistical literature, this is called the Simpson's Paradox, in which a trend that is shown in different groups within a population is not verified in the population as a whole (Blyth, 1972). A clustering algorithm should, therefore, be effective in order to confirm this assumption.

The clustering algorithm chosen was the K-modes algorithm, an extension of the popular K-means algorithm that is used for discrete variables. The function of this model is to categorize each data point into a cluster based on the similarities it has with other data points. The number of clusters created by the algorithm is defined before training it. Using certain techniques, it is possible to determine how many clusters should be generated. The one applied was the elbow method, one of the most common techniques for this purpose. The method computes the change in the difference of observations from their cluster centroids with the increase of the number of clusters *k.* The number of clusters is then chosen once adding another cluster doesn't model the data any better (Bholowalia et al., 2014). Once the number of clusters is defined, we can then proceed to the implementation of the algorithm.

Mathematically, K-modes is an iterative algorithm in which each instance is assigned to one of *k* pre-defined clusters. First, *k* centroids are initialized by randomly selecting *k* data points from the dataset. Then, the dissimilarity measure between each data point and the centroids is calculated through the equation below, being that the smaller the number of mismatches, the more similar the data points are. Considering X and Y to be two different data points describing categorical features, the dissimilarity measure *d* between the two is given by:

$$d(X, Y) = \sum_{j=1}^{m} \delta(x_j, y_i) \quad (1)$$

where $$\delta(x_j, y_i) = \begin{cases} 0 & (x_j = y_i) \\ 1 & (x_j \neq y_i) \end{cases} \quad (2)$$

After determining which centroid each data point is closest to, the algorithm then proceeds to find a new set of centroids that may provide better results (lower dissimilarity measure) based on the mode of the values of the data points attributed for each cluster. More formally, a mode of $X = \{X_1, X_2, \ldots, X_n\}$ is a vector $Q = [q_1, q_2, \ldots, q_m]$ that minimizes

$$D(X, Q) = \sum_{i=1}^{n} d(X_i, Q) \quad (3)$$

where Q is not necessarily an element of X, n is the number of data points and m is the number of categorical features. In order to find the mode for a set that had been attributed to a cluster, we use the theorem below.

Theorem 1. *The function D(X, Q) is minimized iff* $f_r(A_j = q_j \,|X) \geq f_r(A_j = c_{k,j} \,|X)$ *for* $q_j \neq c_{k,j}$ *for all j =1,2,…, m.*

where $n_{c_{k,j}}$ is the number of objects having the *k*th category $c_{k,j}$ in attribute A_j and $f_r(A_j = c_{k,j} \,|X) = \frac{n_{c_{k,j}}}{n}$ is the relative frequency of category $c_{k,j}$ in X.

After the new centroids have been established, the dissimilarity measure is calculated again and the clusters to which each data point belongs to is updated based on the new nearest modes. Theorem 1 is then applied again for a number of iterations until no object in the dataset has changed clusters or a specified number of iterations has been reached (Huang, 1998).

The results for this clustering algorithm applied to our artificial dataset do not confirm the first hypothesis that the number of clusters is the same as the user groups inside the company but do confirm the second hypothesis that the distribution of data is different on these different groups. A more in-depth explanation of the results is given in section 5.1.

4.2 Feature Significance

After grouping the data through clusters and understanding their distribution, it is important to understand what features were the most significant in assigning each data point to its group. This is due to the fact that one of the most fundamental aspects of building a successful machine learning model is fitting meaningful features into the model. By understanding which features hold more significance in our dataset and comprehending the correlations between them, we might be able to optimize the current ones by combining them or simply creating new ones. This

process is called feature engineering and plays an important part of the machine learning project's success or failure (Zheng et al., 2018).

In fact, machine learning should be viewed as an iterative process in which we run the learner, analyze the results and then alter the data and/or the learner and repeat the process (Domingos, 2012). Martin Zinkevich (2020), data scientist at Google, outlines feature engineering as the second phase of creating a proper machine learning pipeline, due to the effectiveness of this process in improving the model in further iterations. This confirms the fact that the meaningfulness of the features is more important than the cleverness of the algorithm due to its description of the target intended for the model (Domingos, 2012).

However, due to the fact that the data that we have is not real, there wasn't much of a way to test how the model would work with potential new features. Nevertheless, there are still methods that we can apply to, on the one hand, do a more thorough analysis of our initial features and also have a more insightful view of their significance and how they relate to each other. This extra understanding of our data will be useful for us to be able to interpret the results of the current model and anticipate ways in which we could improve the data for future iterations.

Since feature importance techniques only function on labelled data, we labelled each data point with the cluster it was assigned to in the clustering algorithm. While we recognize that this approach makes the results of the feature importance methods dependent on the previous algorithm, there was no alternative way to implement them. The first method introduced is called permutation, which measures the importance of a feature by calculating the increase in the model's prediction error if that feature was taken out of the model (Altmann et al., 2010). In order to implement this method, we had to define a proper classifier and evaluation metric. The classifier used was the random forest (see Appendix 2), a tree ensemble method that predicts the category of a data point based on the result of a number of decision trees (Ho, 1995). The evaluating metric chosen was the accuracy score. Accuracy evaluates the proportion between the number of correct predictions and the total number of predictions made. Formally:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions made}} \qquad (4)$$

In this case, this metric will evaluate the number of times a data point was correctly assigned to a cluster through this algorithm. The permutation method will then evaluate the changes in accuracy for every time a feature is excluded, showcasing the importance that each feature has in the accuracy of the results. Due to the fact that every time the algorithm is ran, the predictions should be different, we decided to use the permutation method for a number of ten iterations and then average out the percentual changes in accuracy for a more conclusive set of results.

The second method introduced was the correlation matrix. This method presents the Pearson product-moment correlation coefficient between two variables in a map that can be easily interpretable in order to understand the correlation between two

variables. Mathematically, the correlation between two variables is computed by dividing the covariance of the two variables by the product of the standard deviation of the two variables. Formally, we first compute the covariance between two variables $X(X_1, X_2, \ldots, X_n)$ and $Y(Y_1, Y_2, \ldots, Y_n)$ through the equation below:

$$\text{Cov}(X,Y) = \frac{\sum_{i=1}^{n} (X_i - \bar{X})(Y_i - \bar{Y})}{n} \quad (5)$$

considering X_i and Y_i as individual observations from the variables X and Y respectively, $\bar{X}$ and $\bar{Y}$ as the mean for each variable respectively and n as the number of observations. Then, we compute the correlation through the equation:

$$\rho(X,Y) = \frac{\text{Cov}(X,Y)}{\sigma_X \sigma_Y} \quad (6)$$

Through these steps, we are able to understand how these variables correlate to one another, giving us hints to how we can combine them to possibly create more powerful features that will enhance future iterations of our model. Although the process of feature engineering will not be done for this first attempt at anomaly detection due to the nature of our dataset, this part of the overall machine learning pipeline is important for a better understanding of our current data and can be truly helpful in improving the features for future models. Since the process of feature engineering would be performed universally to all the data independently of the employee group, these methods were applied to the entire dataset.

4.3 Anomaly detection in a time series

As it's been established in the literature review section of this paper, the machine learning technique used in this case should be one for anomaly detection in time series data. After an extensive review of all the scientific methods that have been published to date for this particular task (Blázquez-García, 2020), we decided to apply the algorithm DeepAnT developed by Munir et al. (2019). This is due to its particular characteristics of being able to handle a multivariate time series and being proven to be able to detect both point and subsequent outliers, a necessary requirement given the nature of these attacks.

DeepAnT works in two different steps. The first step consists on a time series predictor. The goal here is to predict entire subsequences of data in a time series, given other data subsequences earlier in the time series. More easily explained, it is using past behavior in order to predict future behavior, taking into account different subsequences. The algorithm used in this predictor is a Convolutional Neural Network (CNN), a class of deep learning neural networks usually used for image and speech recognition and time series predictions. The architecture of these neural networks is explained more in depth in Appendix 3.

For the proper training of our CNN, we need to define important hyperparameters, i.e. parameters that need to be defined before the learning process begins. From the set of hyperparameters, there are the time windows for the training and testing of the neural network. Given that the intent of this algorithm is to learn from the data relating to a certain window of time by understanding the behavior of the data from a previous window of time, defining each of these windows' length is an important decision for the performance of the algorithm.

The first thing to do in order to train the CNN is to split the dataset in two sets, one representing the input and the other the target. In fact, the input of the CNN will be a matrix in which each row corresponds to a time series in the predefined training time window followed by another time series, a point in time ahead of the previous one. The outcome of the CNN will be the predicted matrix of results. This predicted matrix will have the same structure as the target matrix we created when splitting the data. Henceforth, it will have the same number of rows as the input matrix, in which each row also represents a time series where the next one is set one point in time ahead of the previous one. The time window for the time series in this matrix is the testing time window defined before the implementation of the algorithm. The representation of the input and target matrix is represented below.

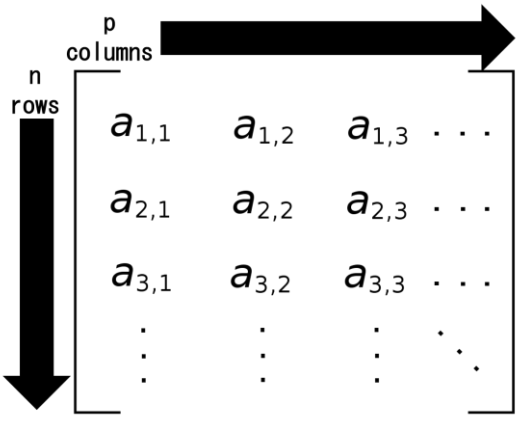

Figure 1 – Representation of the input matrix

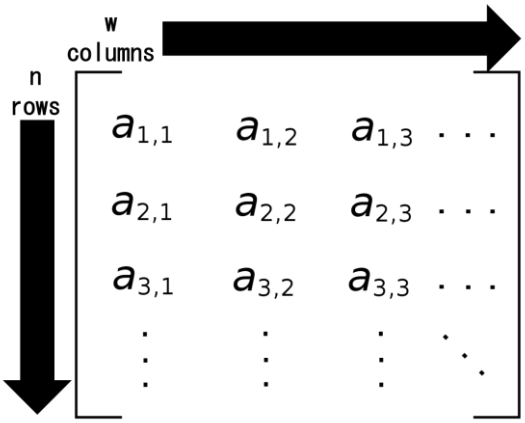

Figure 2 – Representation of the target matrix

where n is the number of time series that are being trained, p is the number of data points for each input time series and w is the number of data points for each target time series.

After the neural network has run for the first time and produced its results, the output will be compared to the actual target data that had been split before. Henceforth, we defined a loss function with the intent of evaluating the exactness of the result. The mean absolute error was the loss function employed, due to the fact that, even though it isn't the most used when the estimation algorithm is based on the gradient descent, it showcases more sensitivity towards outliers and, therefore, is more suited to our model. This loss function is the mean of the absolute difference between actual and predicted values, as shown as:

$$L(y,\hat{y}) = \frac{1}{N}\sum_{i=0}^{N}\left|y_i - \hat{y}_i\right| \quad (7)$$

where N is the number of observations, y is the set of actual observations and $\hat{y}$ is the set of predicted observations. After the calculation of the loss function, a new iteration of the algorithm will be computed where the neural network will change the weights that activate its neurons with the intent of decreasing the loss function and improving the overall result. This process will be repeated for a number of iterations until a global minimum, the lowest point in a loss function, is found. Once the neural network is fully trained, the weights of the optimal iteration, i.e. the optimal combination of the strength of connections between the units in our neural network, are saved. Therefore, the CNN has been properly adapted for our data and it should be able to properly predict user behavior in a certain window of time.

The robustness of this algorithm makes it difficult to thoroughly explain it. In fact, convolutional neural networks have a black box problem (Buhrmester et al., 2019), which means it is humanly impossible to understand what is going on inside a network and they become more difficult to interpret. Nevertheless, we should understand the input, output and architecture of this algorithm.

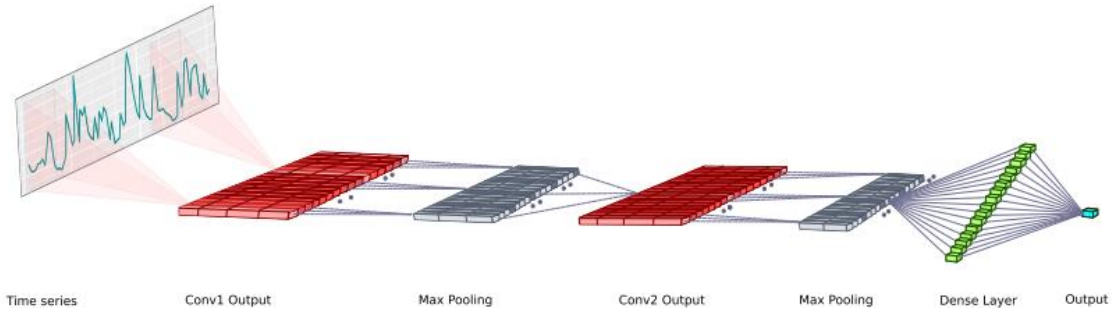

Figure 3 – Convolutional Neural Network for time series prediction. (Source: DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series)

As Figure 3 showcases, the architecture of a CNN is designed to take a large amount of data, in this case a matrix containing multiple time series, and through convolutional and pooling layers amplify what are the most defining characteristics in terms of user behavior. By properly generalizing the user behavior, we can more aptly be able to detect anomalies whenever the behavior shown is too atypical given these expected patterns.

The second step of the DeepAnT algorithm is called an anomaly detector. This is based on the Euclidean distance between the actual and predicted values for each data point at a given point in time. The Euclidean distance is computed as:

$$d(x,y) = \sqrt{\sum_{i=1}^{N}(x_i - y_i)^2} \quad (8)$$

where x and y are two set of points whose Euclidean distance is being measured and N is the number of observations within those sets. If the Euclidean distance between

two points surpasses a previously defined threshold, an anomaly is detected. For this implementation, we defined a 0,8 threshold for this anomaly detector.

Due to the simple logic of the problem, we should create one of these DeepAnT models for each user group inside a company, since it wouldn't make sense to compare the behavior between people with different positions and who would, therefore, naturally access different data and exhibit different behavior. That is not to say that the process followed in 4.1 is devoid of value. It still is quite important to assure that these differences in behavior can be found within the company and if not investigate why. Nevertheless, due to our preconceived idea of how these models should be implemented, we fitted the DeepAnT algorithm for each one of the groups found within the company. Hyperparameters such as the time windows for the input and target data, the learning rate and the number of iterations had to be optimized for each model through various attempts in order to get the best performance possible.

Once the processes described through these models in section 4 are concluded and we have found the anomalies in our dataset, the task of developing a mechanism that will actually detect data breaches in a firm is still far from over. In fact, this is only the first attempt of what is to become a much more effective data breach detection mechanism. As it's been established, labeled data is a luxury that data scientists aren't always able to have. However, once there is access to labeled data, the job becomes much simpler. The task following the detection of anomalies through these algorithms would be to investigate whether these outliers are actually data breaches. This would mean that clients would have to give their feedback on the anomalies found and, from that, data scientists could actually label their outliers as normal or abnormal. This would turn the problem into a semi-supervised approach, a machine learning method which combines a small amount of labeled data with a large amount of unlabeled data during training. This approach would help tuning the models into giving more accurate results and, from there, develop a product that showcases a stronger performance (Ruff et al, 2019).

Another important issue to take into account when building our model is the possibility of overfitting. This is a problem that happens when a model becomes too adapted to the training data that it works poorly when new data is tested in the model. For our case, it would happen if the algorithm detected patterns well enough for some user behavior but failed to grasp other behavior fluctuations, detecting plenty of abnormal behavior when in reality, it wasn't the case. A suggested solution for this potential threat is to collect data on a large enough time span so that the very likely fluctuations in human behavior are encapsulated by the model and the data becomes less likely to overfit.

5. Results

After understanding the reasoning behind each model and why they would be important in the detection of data breaches, we then proceed to show the results and verify whether the hypotheses that had been made are actually validated or not. In section 5.1, we showcase the number of clusters that were found in our unlabeled data and the statistical distribution of the various features on our dataset. In section 5.2, we review the results from the permutation technique and the correlation matrix. In section 5.3, we analyze the performance of the DeepAnT algorithm.

5.1 Clustering

The first hypothesis that we were considering when using the clustering algorithm was whether the role the employee has in the company substantially influences their behavior. By using the elbow method, we were able to determine how many clusters should be generated by the algorithm. By showing how adding a cluster changes the inertia, i.e., how far away the data points within a cluster are, we can analyze how many clusters we should create in the unsupervised algorithm. Once the increase in the number of clusters reaches a point where the inertia isn't significantly changed, then no more clusters should be added, since there are no longer meaningful differences between data points.

Considering the representation showcased in Figure 1, we can determine that this hypothesis is not validated in our dataset. The elbow method seems to suggest that the appropriate number of clusters in this population is three, which is different from the four groups of employees that had been defined. This result is not unexpected since the data that we have was generated randomly and not by actual employees, which makes it likely for the statistics represented by the data not to be realistic. Nevertheless, it is still an important first step to take when doing this data analysis with unlabeled data and three clusters is still a number close enough to the number of employee groups that we can determine that there are significant behavior differences between different users in our dataset.

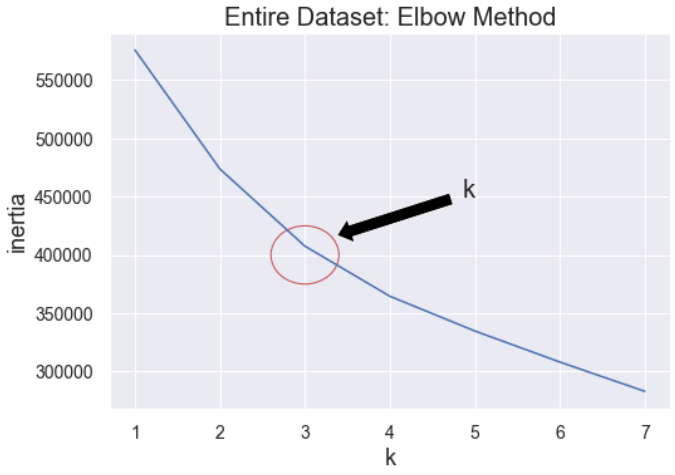

Figure 4 – Elbow method performed on the entire dataset

The second hypothesis we wanted to prove with this algorithm was the difference between the statistical distribution of the data within the entire dataset and the individual groups of employees within it. In order to do that, we fitted the entire dataset and the individual groups of data into a K-modes algorithm and visualized its distribution in order to confirm this hypothesis. In order to decide the number of clusters given to the individual groups of employees, the elbow method was performed in each one of them, all showcasing three clusters should be created for each case, as can be seen in Appendix 4. After fitting the data into the model, we decided to visualize it and analyze whether the distribution of the data differed from the different groups within the dataset. Since the data is multidimensional, the best way to visualize it is through a series of two-dimensional graphs where the axes are two of the features in our dataset.

As the figures in Appendix 5 show, the distribution of our data is different within all the different groups of employees, confirming the hypothesis that had been established in section 4.1. With these two hypotheses tested, the purpose of this model has been achieved. It is worth stressing that this is just an initial data analysis in order to confirm our pre-conceived notions on how the data should be distributed. Following the rules on what data should be fitted into a machine learning algorithm, we know that we shouldn't fit data from different populations in the same model. In this case, it is intuitive that the behavior demonstrated by a nurse should be different than the one from a staff member and that, in order to detect an anomaly, we should base the anomalous behavior in comparison to other nurses. Nevertheless, it is worth going through this initial stage to get a representation on how the data works before moving to more complex models like the one used individually for different employee groups in 4.3.

5.2 Feature Importance

The first method implemented in order to determine feature importance was the permutation, which evaluated the changes in accuracy once a feature is taken out of the model. The results are shown in Figure 5 and in Appendix 6.

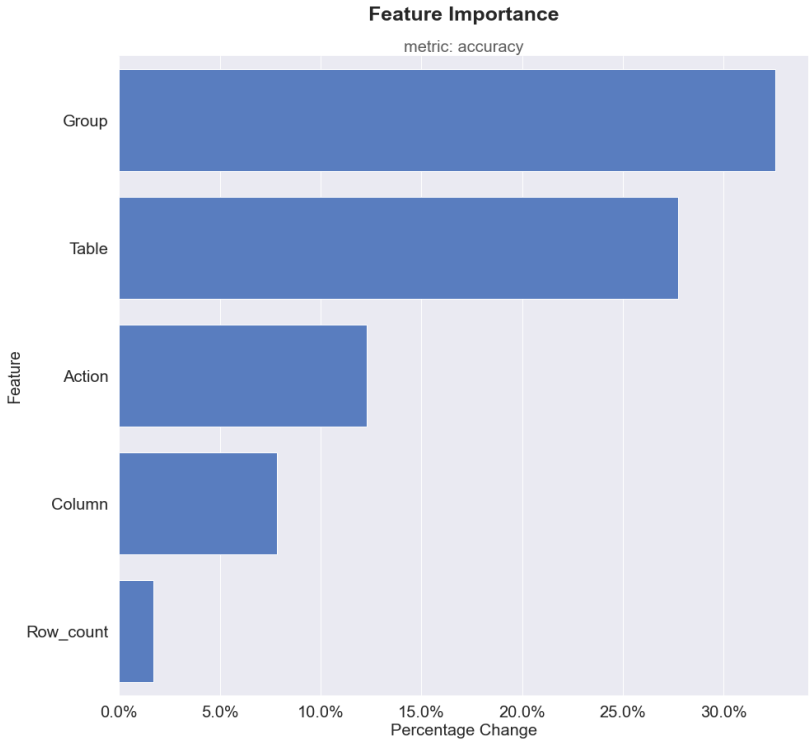

Figure 5 – Permutation method performed on the entire dataset

As the results showcase, the most defining features in our dataset are “Group” and “Table”. On the other hand, “Row_count” exhibits low importance. This could give some insight into how the less important features could become powerful if they were combined with more powerful ones. For instance, creating a feature that combined the “Table” accessed by the user with the “Row_count” in that query could prove to be more meaningful to our model and, therefore, improve the overall results. However, since this data was randomly generated, we should take these results with a lot of caution and only consider them as an example of how they would be interpreted if they were real. Given the constraints set upon us, this interpretation would have to be reviewed once real data was tested and new results were reached.

The second method tested for our feature significance analysis was the correlation matrix. The results for the method are presented in Figure 6.

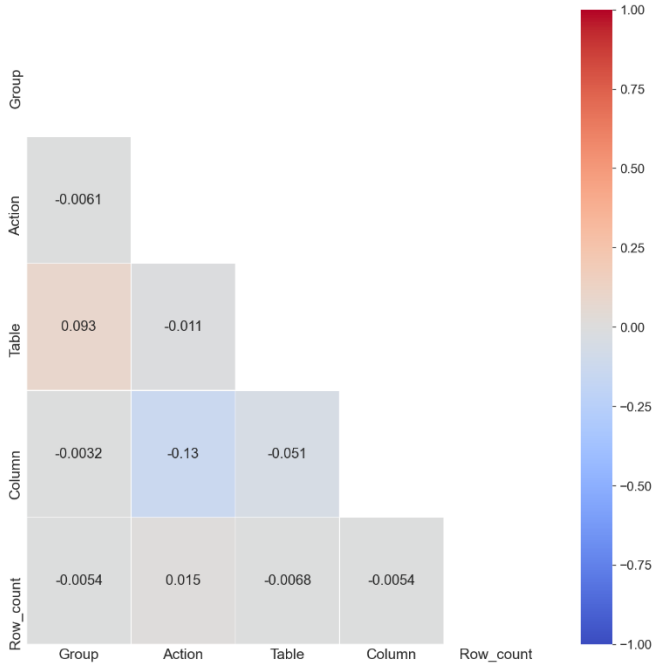

Figure 6 – Correlation Heatmap performed on the entire dataset

As is shown, most of features present very little correlation to each other. The only exceptions are the positive correlation between “Table” and “Group” and the negative correlation between “Action” and “Column”, which even then is practically null. Nevertheless, we are just measuring the linear correlation between variables. Again, this provides some insight into new possible features coming from the combination of these ones, but these results should be taken with caution.

5.3 DeepAnT

The DeepAnT algorithm was implemented in each one of the employee groups presented in our dataset: medics, nurses, administrators and staff. Due to the fact that the data on each one these groups was randomly generated, the interpretation that we make of this implementation should not be focused on the anomalies that were detected and whether they were data breaches or not, but instead on how the algorithm adapted itself to the data and whether it learned the behavior patterns within it. On that front, we want an algorithm that is capable of predicting changes in behavior without the threat of overfitting, since becoming too adapted to the training data would make it useless once new data was fitted into the model.

As explained in 4.3, different hyperparameters were trained for each model in order to get the lowest loss function possible and, after all the models had reached their best performing iteration, all of them had a different set of hyperparameters. This was already expected due to the fact that some employee groups had more data points and would, therefore, work better with different training and target time

windows. The metric used in order to analyze how well these hyperparameters performed in our model was the loss function defined in the implementation, the mean absolute error. After multiple iterations, the CNNs were trained to their optimal set of hyperparameters and loss function. The loss functions for all of the models in our dataset are represented in the Table 2 below.

The loss functions in our models don't present ideal results. In fact, considering that the threshold set was 0,8, these results will surely showcase a large number of outliers, which wasn't the intent of the model. Nevertheless, there are certain points that should be taken into consideration. First, we should acknowledge that all the data was generated randomly, not really representing the behavior of actual people, making it more difficult to find patterns. Furthermore, the solution could come from setting a higher threshold, although that's a decision that can only be taken once a thorough analysis of what the anomalies represent has been done which, at this point, we cannot do. Finally, an encouraging aspect is that the employee groups that had more data on them, presented a much better loss function. This suggests that with more representative data and a larger amount of it, the algorithm can present better results.

Employee Group	Number of records	Loss Function (MAE)
Medics	53 136	0,67
Nurses	6 794	0,8
Administrators	23 968	0,77
Staff	31 280	0,72

Table 2 – Loss function per model of employee group

In order to actually explore how well the algorithm adapts itself to the data, we plotted a time series representing the prediction of behavior given by the CNN versus the actual data of that time frame and the anomaly points that actually surpass the threshold set in the anomaly detector. Each time series shows a subsequence that is being examined for anomalies. In fact, the way this model works is by evaluating anomalous behavior in subsequences with the length of the target time window defined before training the model. In order to predict the anomalies on that time frame, the optimal neuron structure and the weights learned during the training of the CNN were used on a previous time window as lengthy as the training time window defined before the implementation.

The graphs for each one of the plotted time series can be seen in Appendix 7. The actual data on each time frame is represented by a blue line, the predicted data computed by the algorithm is represented by the black line and the anomalies are represented by red data points. In all the time series, we can see that the actual data was quite erratic, fluctuating a lot more abruptly than we had hoped. That explains the reason why the loss functions were so high to begin with. However, we can also see that the algorithm did a good job in understanding patterns in our data, even with its rapid fluctuations. We can see that whenever the target sequence shows a more consistent trend, the prediction made by our algorithm follows that trend, although not to same extremes. That is also a positive aspect of the behavior shown by the predicted sequences, since not mimicking exactly the inconsistent behavior of target sequences means that they didn't adapt too much to the training data and can be

used with new data when necessary. This provides some relief into the overfitting problem that machine learning models can have. Furthermore, as it was expected by the loss functions in our training model, the number of anomalies in each group were quite high, as can be seen by the large number of red dots. Nevertheless, we can also see that the number of red points is higher in the groups we had less data on, again showcasing the likelihood of the algorithm performing better once there is more data to train it on.

Once actual data is fit into this model, the next steps would be to evaluate the anomalies detected by the algorithm and adjust its hyperparameters, whether they would be the training and testing windows, the number of epochs, the learning rate or the threshold, in order to adapt the model into giving better results. Furthermore, we should also do some feature engineering and evaluate the results the model presents with other features as explained in 4.2 and label our data if possible, turning this into a semi-supervised learning problem as explained in 4.3. As Martin Zinkevich (2020) states, machine learning models are iterative processes and take their time in order to become effective. The most valuable asset for their success is data and only through that can we extract meaningful conclusions.

6. Conclusions and Future Research

In conclusion, the purpose of this paper was to define a machine learning framework that Blockbird Ventures could use in order to detect data breaches. On that front, it was demonstrated why each one of the presented steps was important and how they would provide meaningful results. The models created in 4.1 and 4.2 help us interpret our data much more clearly and try to make it as meaningful as possible for when it is fitted into a more robust algorithm. The more complex approach taken in 4.3 also displays some encouraging results, as it clearly comprehends patterns in data without overfitting the model.

The biggest and most obvious limitation in this entire process was the lack of real data. Blockbird is a new company that has launched its product into the market very recently. Therefore, it was quite difficult to get real data to fit in this model so soon. Establishing partnerships with other firms, integrating our software with theirs and extracting large amounts of data from a fairly large period of time are all things that take time and were just not possible for the time being. The coronavirus outbreak that happened during this period also didn't help in creating these partnerships, delaying some of the goals that were set for this time. Nevertheless, a machine learning framework has been developed and the way it works with data can be shown to future clients, which certainly adds value to the company.

For the future, the first and most obvious suggestion in order to better prove the purposed assignment is to test the established framework with real data coming from users inside a company. Furthermore, we also suggest special attention into the possibility of the model overfitting, which would make it adaptable to the training data but little useful for new data. In order to fix this problem, we recommend that a large sample of data be trained in this model, so that natural fluctuations in human behavior are incorporated into it and only actual anomalies get flagged by our

system. We also remind the next steps to follow once the anomalies are detected. Once we can understand how well current features perform in our model through the analysis performed in 4.2, we should go through the feature engineering process and try to create better ones in order to get better results. Additionally, labeling the data and turning this into a semi-supervised learning problem should also help tuning the model developed in 4.3. We also suggest continuous attentiveness into future scientific research on anomaly detection in time series techniques. For this case, we chose DeepAnT due to the combination of it being applicable to the necessary task and the dataset structure that we had. However, new machine learning methods are being developed and tested constantly and it is important to be aware of that since more effective methods may be developed in the future.

7. References

Cheng, L., Liu, F., & Yao, D. (2017). Enterprise data breach: causes, challenges, prevention, and future directions. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *7*(5), e1211.

Cheng, L., Liu, F., & Yao, D. (2017). Enterprise data breach: causes, challenges, prevention, and future directions. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *7*(5), e1211.

Liu, L., Han, M., Wang, Y., & Zhou, Y. (2018, June). Understanding data breach: A visualization aspect. In *International Conference on Wireless Algorithms, Systems, and Applications* (pp. 883-892). Springer, Cham.

Sen, R., & Borle, S. (2015). Estimating the contextual risk of data breach: An empirical approach. *Journal of Management Information Systems*, *32*(2), 314-341.

Wheatley, S., Maillart, T., & Sornette, D. (2016). The extreme risk of personal data breaches and the erosion of privacy. *The European Physical Journal B*, *89*(1), 1-12.

Algarni, A. M., & Malaiya, Y. K. (2016, May). A consolidated approach for estimation of data security breach costs. In *2016 2nd International Conference on Information Management (ICIM)* (pp. 26-39). IEEE.

Solove, D. J., & Citron, D. K. (2017). Risk and anxiety: A theory of data-breach harms. *Tex. L. Rev.*, *96*, 737.

Nieuwesteeg, B., & Faure, M. (2018). An analysis of the effectiveness of the EU data breach notification obligation. *Computer Law & Security Review*, *34*(6), 1232-1246.

Khan, F. S., Kim, J. H., Moore, R. L., & Mathiassen, L. (2019). Data Breach Risks and Resolutions: A Literature Synthesis.

Yayla, A. A., & Hu, Q. (2011). The impact of information security events on the stock value of firms: The effect of contingency factors. *Journal of Information Technology*, *26*(1), 60-77.

Roberds, W., & Schreft, S. L. (2009). Data breaches and identity theft. *Journal of Monetary Economics*, *56*(7), 918-929.

Graham, A. (2019). “The damaging after-effects of a data breach”, IT Governance Blog. Retrieved from https://www.itgovernance.co.uk/blog/the-damaging-after-effects-of-a-data-breach. Accessed 2020-05-14.

Miller, L. (2018). Cybersecurity insurance: Incentive alignment solution to weak corporate data protection. *Available at SSRN 3113771*.

Karyda, M., & Mitrou, L. (2016). Data Breach Notification: Issues and Challenges for Security Management. In *MCIS* (p. 60).

Kiener-Manu, K. (2020). Cybercrime Module 10 Key Issues: Data Breach Notification Laws. Retrieved from https://www.unodc.org/e4j/en/cybercrime/module-10/key-issues/data-breach-notification-laws.html. Accessed 2020-05-14.

Nieuwesteeg, B., & Faure, M. (2018). An analysis of the effectiveness of the EU data breach notification obligation. *Computer Law & Security Review*, *34*(6), 1232-1246.

Tikkinen-Piri, C., Rohunen, A., & Markkula, J. (2018). EU General Data Protection Regulation: Changes and implications for personal data collecting companies. *Computer Law & Security Review*, *34*(1), 134-153.

Sobers, R. (2020, March 3). 107 Must-Know Data Breach Statistics for 2020: Varonis. Retrieved from https://www.varonis.com/blog/data-breach-statistics/. Accessed 2020-05-14.

Irwin, L. (2020, April 8). The most common causes of data breaches and how you can spot them. Retrieved from https://www.itgovernance.eu/blog/en/the-most-common-causes-of-data-breaches-and-how-you-can-spot-them. Accessed 2020-05-14.

Romanosky, S., Hoffman, D., & Acquisti, A. (2014). Empirical analysis of data breach litigation. *Journal of Empirical Legal Studies*, *11*(1), 74-104.

Sanzgiri, A., & Dasgupta, D. (2016, April). Classification of insider threat detection techniques. In *Proceedings of the 11th annual cyber and information security research conference* (pp. 1-4).

IBM Security (2020). "Cost of a Data Breach Report 2019". Retrieved from https://www.ibm.com/downloads/cas/ZBZLY7KL. Accessed 2020-05-14.

Hodge, R. (2019). Welcome to the 2019 Data Breach Hall of Shame. Retrieved from https://www.cnet.com/news/2019-data-breach-hall-of-shame-these-were-the-biggest-data-breaches-of-the-year/. Accessed 2020-05-14.

Shashanka, M., Shen, M. Y., & Wang, J. (2016, December). User and entity behavior analytics for enterprise security. In *2016 IEEE International Conference on Big Data (Big Data)* (pp. 1867-1874). IEEE.

Brook, C. (2020, January 27). What is User and Entity Behavior Analytics? A Definition of UEBA, Benefits, How It Works, and More. Retrieved from https://digitalguardian.com/blog/what-user-and-entity-behavior-analytics-definition-ueba-benefits-how-it-works-and-more. Accessed 2020-05-14.

Lin, D. (2016). Applying Data Science to User and Entity Behavior Analytics.

Stamp, M. (2017). *Introduction to machine learning with applications in information security*. CRC Press.

Salitin, M. A., & Zolait, A. H. (2018, November). The role of User Entity Behavior Analytics to detect network attacks in real time. In *2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)* (pp. 1-5). IEEE.

Ghahramani, Z. (2003, February). Unsupervised learning. In *Summer School on Machine Learning* (pp. 72-112). Springer, Berlin, Heidelberg.

Campos, G. O., Zimek, A., Sander, J., Campello, R. J., Micenková, B., Schubert, E., ... & Houle, M. E. (2016). On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. *Data Mining and Knowledge Discovery*, *30*(4), 891-927.

Lukashin, A., Popov, M., Bolshakov, A., & Nikolashin, Y. (2019, October). Scalable Data Processing Approach and Anomaly Detection Method for User and Entity Behavior Analytics Platform. In *International Symposium on Intelligent and Distributed Computing* (pp. 344-349). Springer, Cham.

Blockbird Ventures - About Page - Meet the Team. (n.d.). Retrieved from https://blockbird.ventures/about/. Accessed 2020-05-14

Bourbon, M. J. (2018, June 8). Há uma startup portuguesa que quer ensinar a blockchain a diretores e gestores. Retrieved from https://expresso.pt/economia/2018-06-08-Ha-uma-startup-portuguesa-que-quer-ensinar-a-blockchain-a-diretores-e-gestores. Accessed 2020-05-14

Wolpert, D. H. (1996). The lack of a priori distinctions between learning algorithms. *Neural computation*, *8*(7), 1341-1390.

Kriegel, H. P., Schubert, E., & Zimek, A. (2017). The (black) art of runtime evaluation: Are we comparing algorithms or implementations?. *Knowledge and Information Systems*, *52*(2), 341-378.

Domingues, R., Filippone, M., Michiardi, P., & Zouaoui, J. (2018). A comparative evaluation of outlier detection algorithms: Experiments and analyses. *Pattern Recognition*, *74*, 406-421.

Han, J., Pei, J., & Kamber, M. (2011). *Data mining: concepts and techniques*. Elsevier.

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM computing surveys (CSUR)*, *41*(3), 1-58.

Böhmer, K., & Rinderle-Ma, S. (2016, October). Multi-perspective anomaly detection in business process execution events. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"* (pp. 80-98). Springer, Cham.

Schonenberg, H., Mans, R., Russell, N., Mulyar, N., & van der Aalst, W. (2008). Process flexibility: A survey of contemporary approaches. In *Advances in enterprise engineering I* (pp. 16-30). Springer, Berlin, Heidelberg.

Rinderle, S., Weber, B., Reichert, M., & Wild, W. (2005, September). Integrating process learning and process evolution–a semantics based approach. In *International Conference on Business Process Management* (pp. 252-267). Springer, Berlin, Heidelberg.

Euting, S., Janiesch, C., Fischer, R., Tai, S., & Weber, I. (2014, March). Scalable business process execution in the cloud. In *2014 IEEE International Conference on Cloud Engineering* (pp. 175-184). IEEE.

Wen, L., Van Der Aalst, W. M., Wang, J., & Sun, J. (2007). Mining process models with non-free-choice constructs. *Data Mining and Knowledge Discovery*, *15*(2), 145-180.

Böhmer, K., & Rinderle-Ma, S. (2017). Anomaly detection in business process runtime behavior--challenges and limitations. *arXiv preprint arXiv:1705.06659*.

Atla, A., Tada, R., Sheng, V., & Singireddy, N. (2011). Sensitivity of different machine learning algorithms to noise. *Journal of Computing Sciences in Colleges*, *26*(5), 96-103.

Bishop, C. M. (2006). Pattern Recognition and Machine Learning Springer-Verlag New York. *Inc. Secaucus, NJ, USA*, *2006*.

Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media.

Escalante, H. J. (2005). A comparison of outlier detection algorithms for machine learning. In *Proceedings of the International Conference on Communications in Computing* (pp. 228-237).

Chickowski, E. (2013, June 20). Why Are We So Slow To Detect Data Breaches? Retrieved from https://www.darkreading.com/attacks-breaches/why-are-we-so-slow-to-detect-data-breaches/d/d-id/1139970. Accessed 2020-05-14.

Mookerjee, V., Mookerjee, R., Bensoussan, A., & Yue, W. T. (2011). When hackers talk: Managing information security under variable attack rates and knowledge dissemination. *Information Systems Research*, *22*(3), 606-623.

Stolfo, S. J., Bellovin, S. M., Hershkop, S., Keromytis, A. D., Sinclair, S., & Smith, S. W. (Eds.). (2008). *Insider attack and cyber security: beyond the hacker* (Vol. 39). Springer Science & Business Media.

Grier, J. (2011). Detecting data theft using stochastic forensics. *digital investigation*, *8*, S71-S77.

Ho, S. M., Kao, D., & Wu, W. Y. (2018). Following the breadcrumbs: Timestamp pattern identification for cloud forensics. *Digital Investigation*, *24*, 79-94.

Blázquez-García, A., Conde, A., Mori, U., & Lozano, J. A. (2020). A review on outlier/anomaly detection in time series data. *arXiv preprint arXiv:2002.04236*.

Hawkins, D. M. (1980). *Identification of outliers* (Vol. 11). London: Chapman and Hall.

Reinsel, G. C. (2003). *Elements of multivariate time series analysis*. Springer Science & Business Media.

Moral, P., & González, P. (2003). Univariate Time Series Modelling. In *Computer-Aided Introduction to Econometrics* (pp. 163-224). Springer, Berlin, Heidelberg.

Irwin, L. (2019, March 12). The 6 most common ways data breaches occur. Retrieved from https://www.itgovernance.eu/blog/en/the-6-most-common-ways-data-breaches-occur. Accessed 2020-05-14.

Zhang, Z. (2019, May 11). Feature Selection Why & How Explained (Part 1). Retrieved from https://towardsdatascience.com/feature-selection-why-how-explained-part-1-c2f638d24cdb. Accessed 2020-05-14.

Azevedo, G. (2019, August 2). Feature selection techniques for classification and Python tips for their application. Retrieved from https://towardsdatascience.com/feature-selection-techniques-for-classification-and-python-tips-for-their-application-10c0ddd7918b. Accessed 2020-05-14.

Agresti, A. (2003). *Categorical data analysis* (Vol. 482). John Wiley & Sons.

Sarkar, D. (2019, March 27). Categorical Data. Retrieved from https://towardsdatascience.com/understanding-feature-engineering-part-2-categorical-data-f54324193e63. Accessed 2020-05-14.

Brownlee, J. (2019, August 6). How to use Data Scaling Improve Deep Learning Model Stability and Performance. Retrieved from https://machinelearningmastery.com/how-to-improve-neural-network-stability-and-modeling-performance-with-data-scaling/. Accessed 2020-05-14.

Jaitley, U. (2018, October 7). Why Data Normalization is necessary for Machine Learning models. Retrieved from https://medium.com/@urvashilluniya/why-data-normalization-is-necessary-for-machine-learning-models-681b65a05029. Accessed 2020-05-14.

Blyth, C. R. (1972). On Simpson's paradox and the sure-thing principle. *Journal of the American Statistical Association*, *67*(338), 364-366.

Huang, Z. (1998). Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data mining and knowledge discovery*, *2*(3), 283-304.

LeCun, Y., & Bengio, Y. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, *3361*(10), 1995.

Bholowalia, P., & Kumar, A. (2014). EBK-means: A clustering technique based on elbow method and k-means in WSN. *International Journal of Computer Applications*, *105*(9).

Zinkevich, M. (2019). Rules of Machine Learning: Best Practices for ML Engineering. Retrieved from https://developers.google.com/machine-learning/guides/rules-of-ml. Accessed April 27, 2020.

Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM*, *55*(10), 78-87.

Zheng, A., & Casari, A. (2018). *Feature engineering for machine learning: principles and techniques for data scientists*. " O'Reilly Media, Inc.".

Altmann, A., Toloşi, L., Sander, O., & Lengauer, T. (2010). Permutation importance: a corrected feature importance measure. *Bioinformatics*, *26*(10), 1340-1347.

Ho, T. K. (1995, August). Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition* (Vol. 1, pp. 278-282). IEEE.

Buhrmester, V., Münch, D., & Arens, M. (2019). Analysis of Explainers of Black Box Deep Neural Networks for Computer Vision: A Survey. *arXiv preprint arXiv:1911.12116*.

Ruff, L., Vandermeulen, R. A., Görnitz, N., Binder, A., Müller, E., Müller, K. R., & Kloft, M. (2019). Deep semi-supervised anomaly detection. *arXiv preprint arXiv:1906.02694*.

Verizon Enterprise. (2019). Data Breach Investigations Report. Retrieved from https://enterprise.verizon.com/resources/reports/2019-data-breach-investigations-report.pdf. Accessed 2020-05-14.

Sainath, T. N., Mohamed, A. R., Kingsbury, B., & Ramabhadran, B. (2013, May). Deep convolutional neural networks for LVCSR. In *2013 IEEE international conference on acoustics, speech and signal processing* (pp. 8614-8618). IEEE.

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5-32.

8. Appendices

8.1 Rules and alerts on Privata.ai

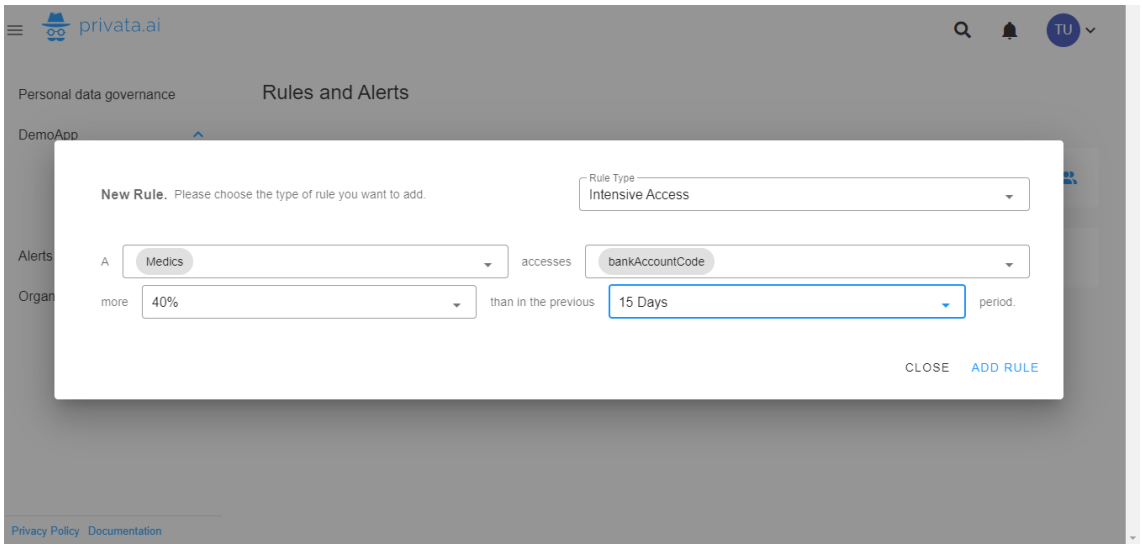

Figure 7 – Intensive Access Rule

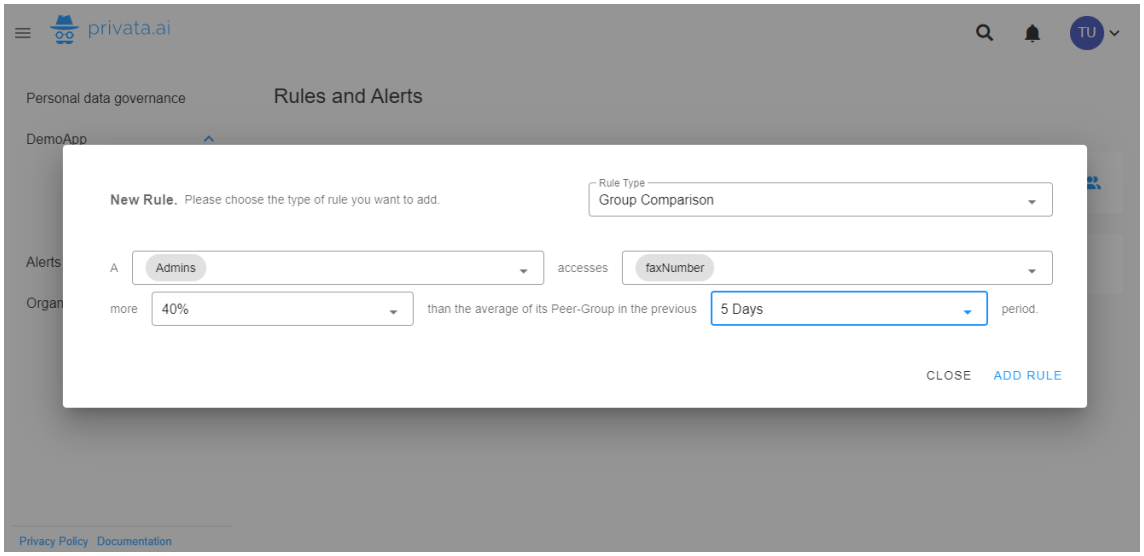

Figure 8 – Group Comparison Rule

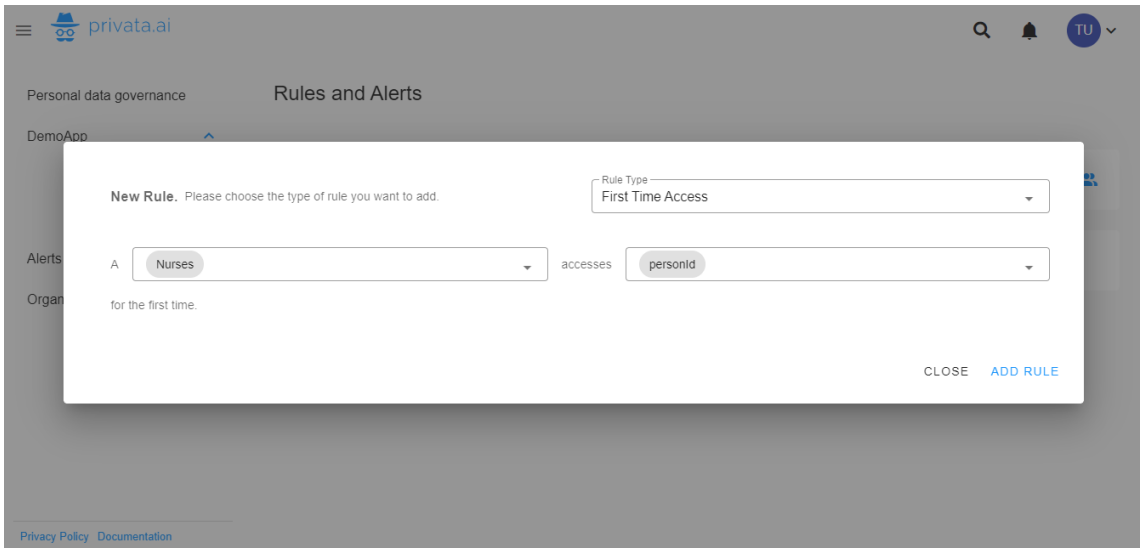

Figure 9 – First Time Access Rule

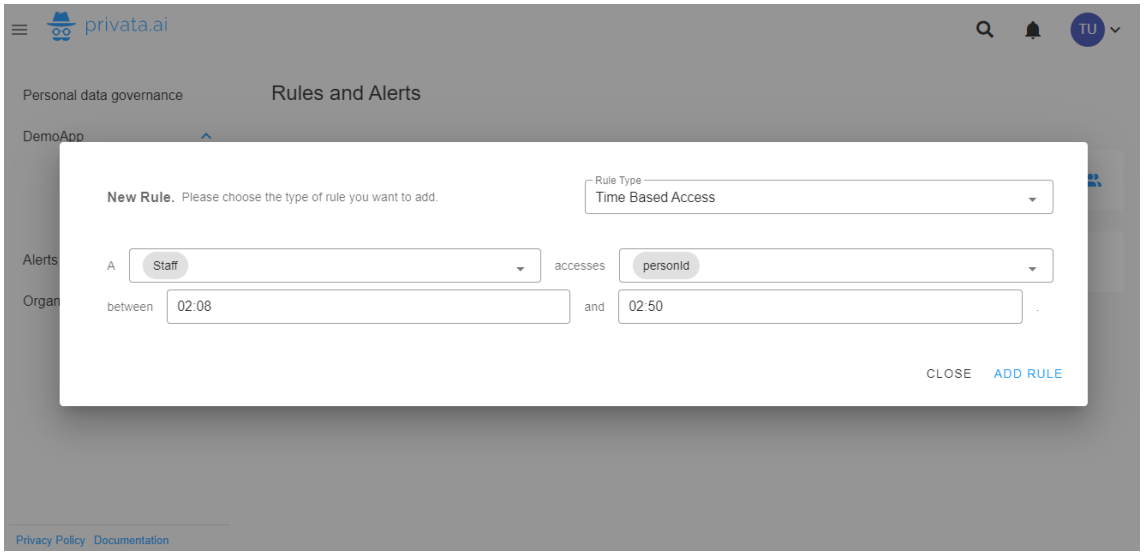

Figure 10 – Time Based Access Rule

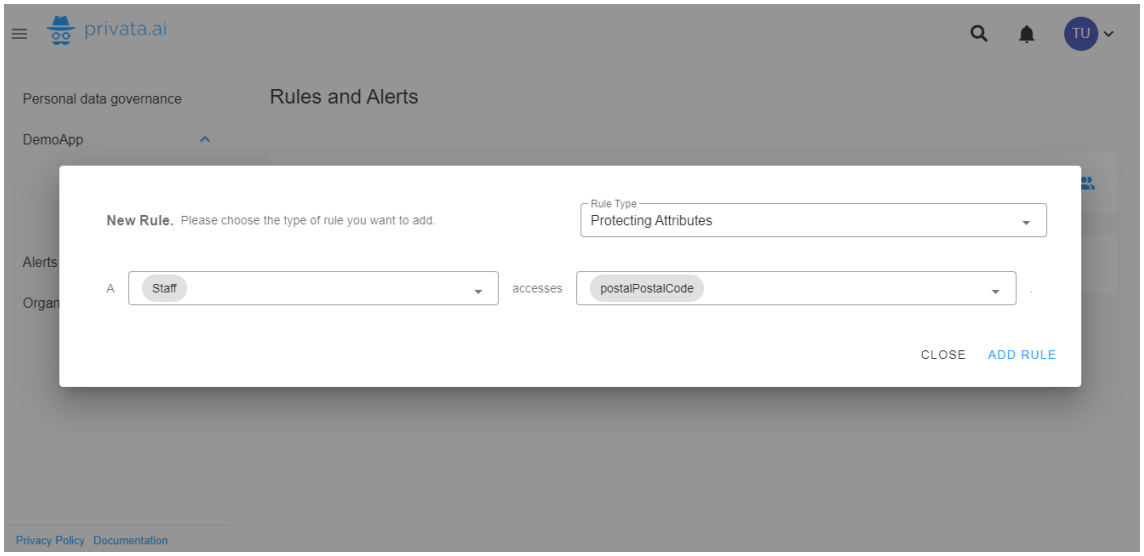

Figure 11 – Protecting Attributes Rule

8.2 Random Forest

(Retrieved from Breiman, 2001)

A random forest is a classifier consisting of a collection of tree-structured classifiers {h(x, kΘ), k = 1,...} where the {kΘ} are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input x.

Given an ensemble of classifiers $h_1(x)$, $h_2(x)$, . . . , $h_K(x)$, and with the training set drawn at random from the distribution of the random vector Y, X, define the margin function as

$$mg(X, Y) = av_k I(h_k(X) = Y) - \max_{j \neq Y} av_k I(h_k(X) = j). \quad (9)$$

where I(·) is the indicator function. The margin measures the extent to which the average number of votes at X, Y for the right class exceeds the average vote for any other class. The larger the margin, the more confidence in the classification. The generalization error is given by

$$PE^* = P_{X,Y}(mg(X, Y) < 0) \quad (10)$$

where the subscripts X, Y indicate that the probability is over the X, Y space.

In random forests, $h_k(X) = h(X, k\Theta)$. For a large number of trees, it follows from the Strong Law of Large Numbers and the tree structure that:

Theorem 2. As the number of trees increases, for almost surely all sequences Θ1,...PE* converges to

$$P_{X,Y}(P_\Theta(h(X,\Theta) = Y) - \max_{j \neq Y} P_\Theta(h(X,\Theta) = j) < 0). \quad (11)$$

This result explains why random forests do not overfit as more trees are added, but produce a limiting value of the generalization error.

8.3 Convolutional Neural Netwok

(Retrieved from Sainath et al., 2013)

In a fully-connected network like CNNs, each hidden activation hi is computed by multiplying the entire input V by weights W in that layer. However, in a CNN, each hidden activation is computed by multiplying a small local input (i.e. [v1, v2, v3]) against the weights W. The weights W are then shared across the entire input space, as indicated in the figure. After computing the hidden units, a maxpooling layer helps to remove variability in the hidden units (i.e. convolutional band activations), that exist due to speaking styles, channel distortions, etc. Specifically, each max-pooling unit receives activations from r convolutional bands, and outputs the maximum of the activations from these bands. Most CNN work in image recognition has the lower network layers be convolutional, while the higher network layers are fully connected. In this section, we will explore how many convolutional vs. fully connected layers are needed, what is the optimal number of hidden units per layer, what is the best pooling strategy, and the best input feature type for CNNs.

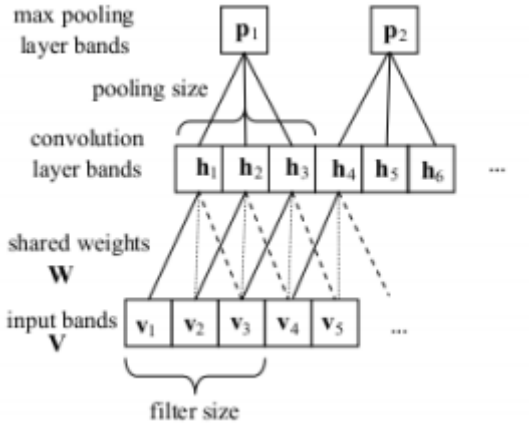

Figure 12 - Diagram showing a typical convolutional network architecture consisting of a convolutional and max-pooling layer. In this diagram, weights with the same line style are shared across all convolutional layer bands.

8.4 Elbow Method performed on different employee groups

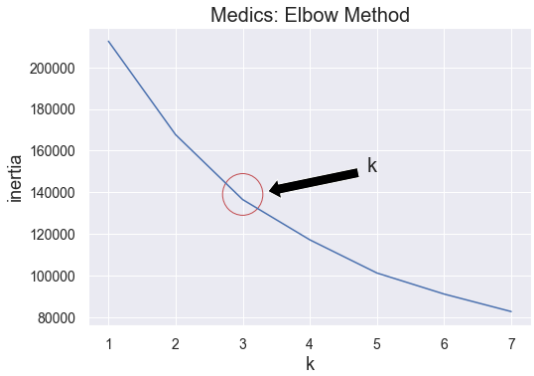

Figure 13 – Elbow method performed on Medics group

Figure 14 – Elbow method performed on Nurses group

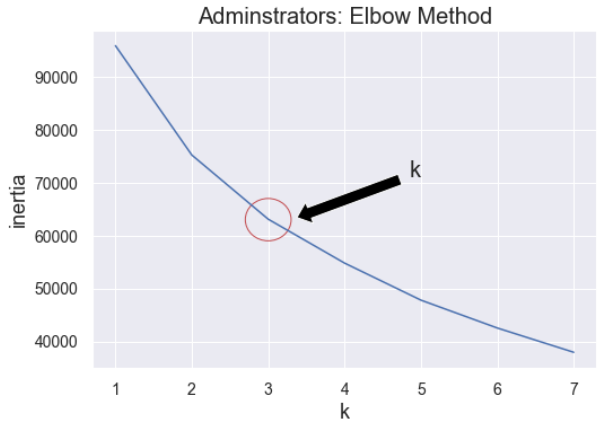

Figure 15 – Elbow method performed on Administrators group

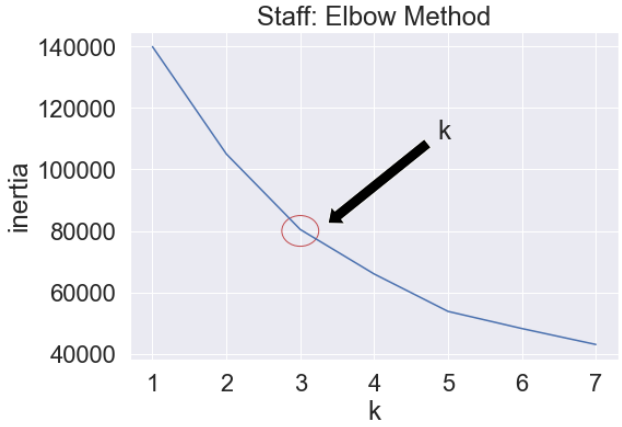

Figure 16 – Elbow method performed on Staff group

8.5 Data Distribution across employee groups with the defined number of clusters

Figure 19 – Data distribution across the entire dataset

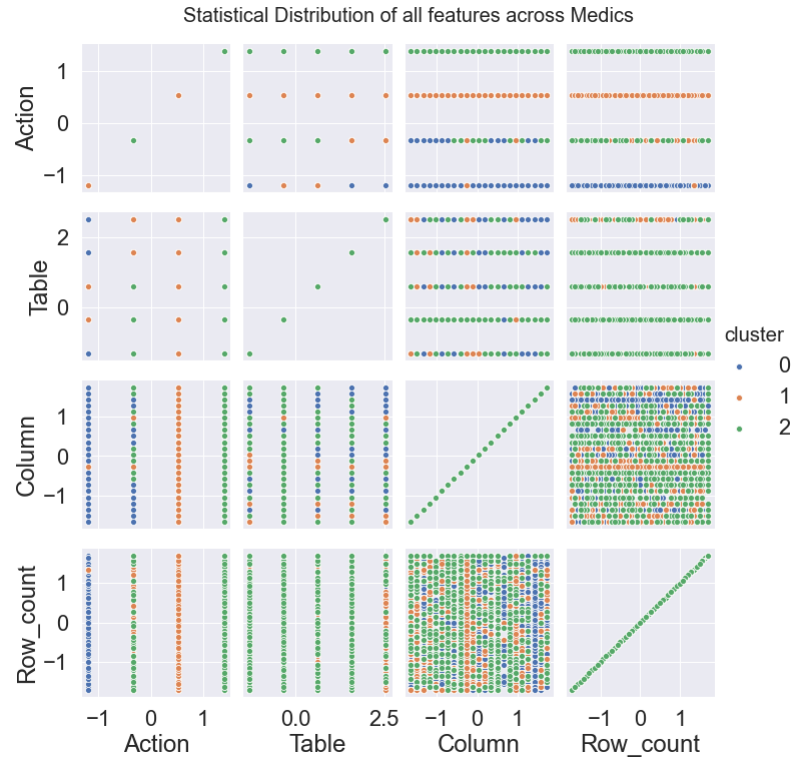

Figure 18 – Data distribution across Medics

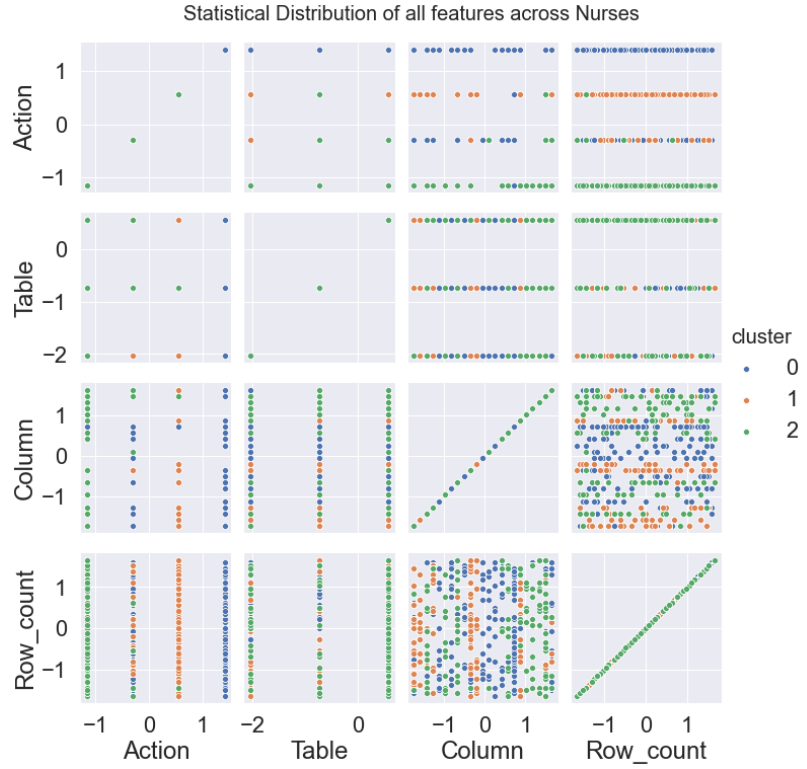

Figure 19 – Data distribution across Nurses

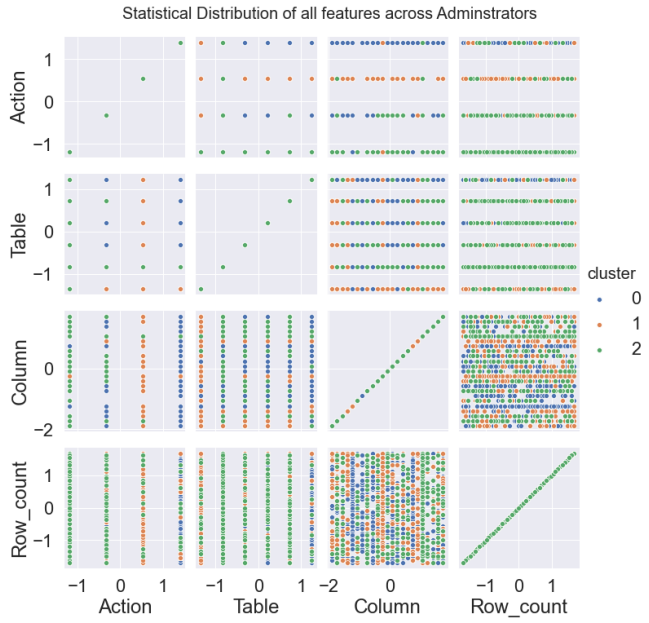

Figure 20 – Data distribution across Administrators

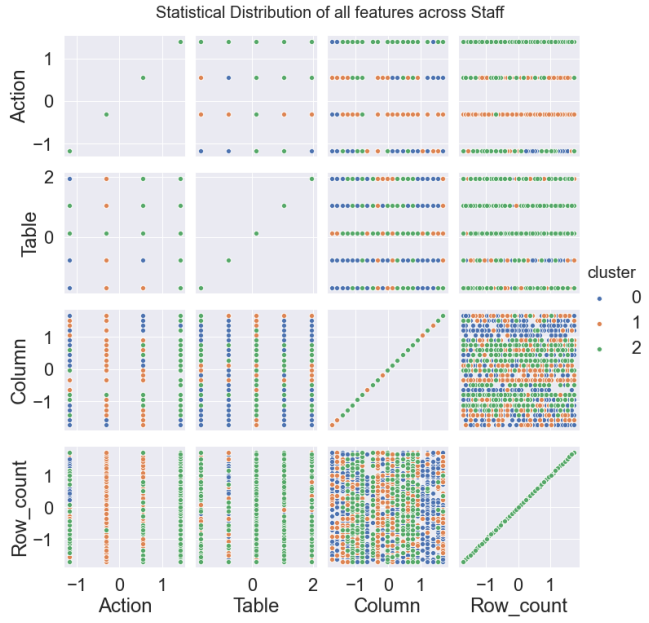

Figure 21 – Data distribution across Staff

8.6 Permutation

Variable	Change in percentage (%)
Group	32,57
Table	27,74
Action	12,29
Column	7,82
Row_count	1,69

Table 3 – Permutation (metric: accuracy)

8.7 DeepAnT representation in subsequence time series

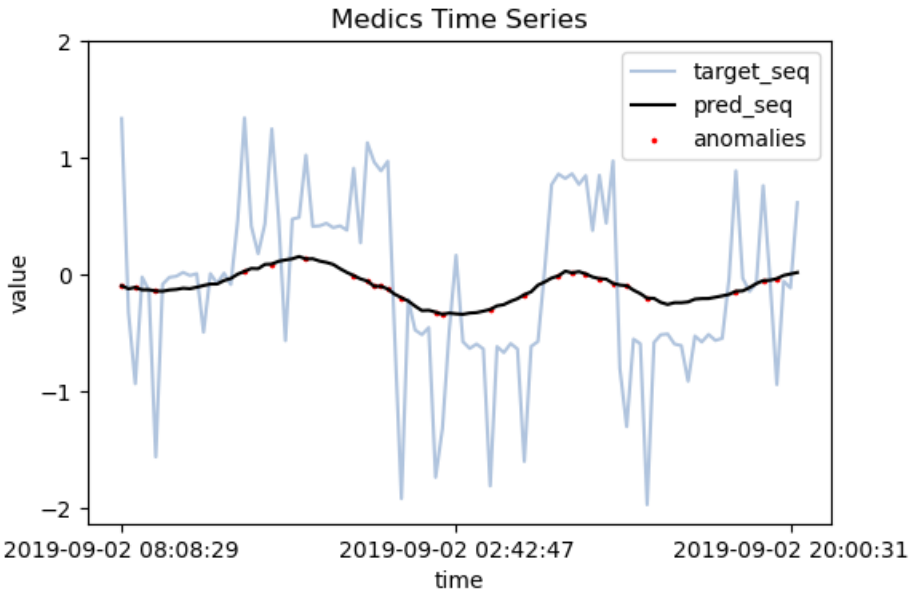

Figure 22 – Medics DeepAnT representation

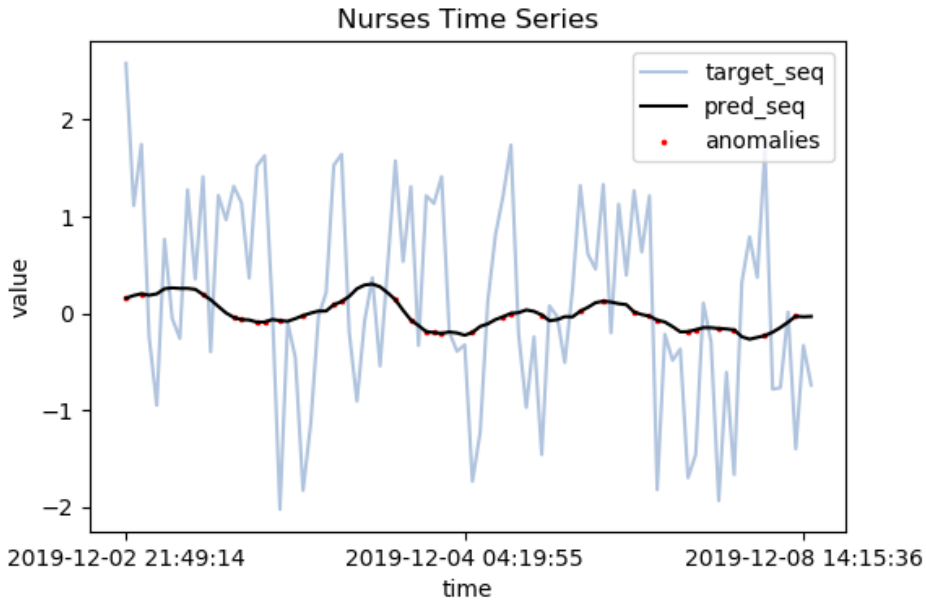

Figure 23 – Nurses DeppAnT representation

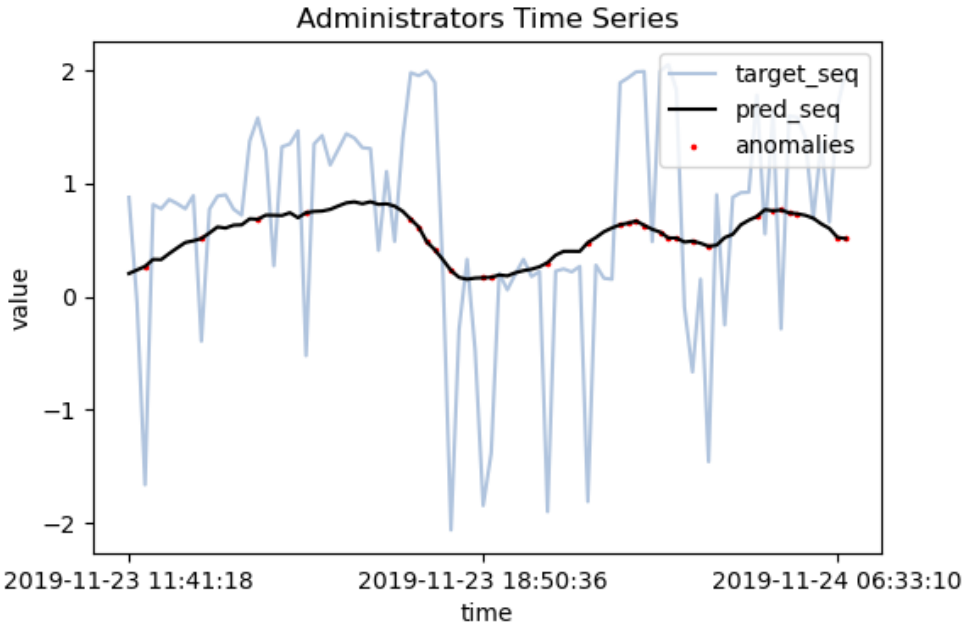

Figure 24 – Administrators DeppAnT representation

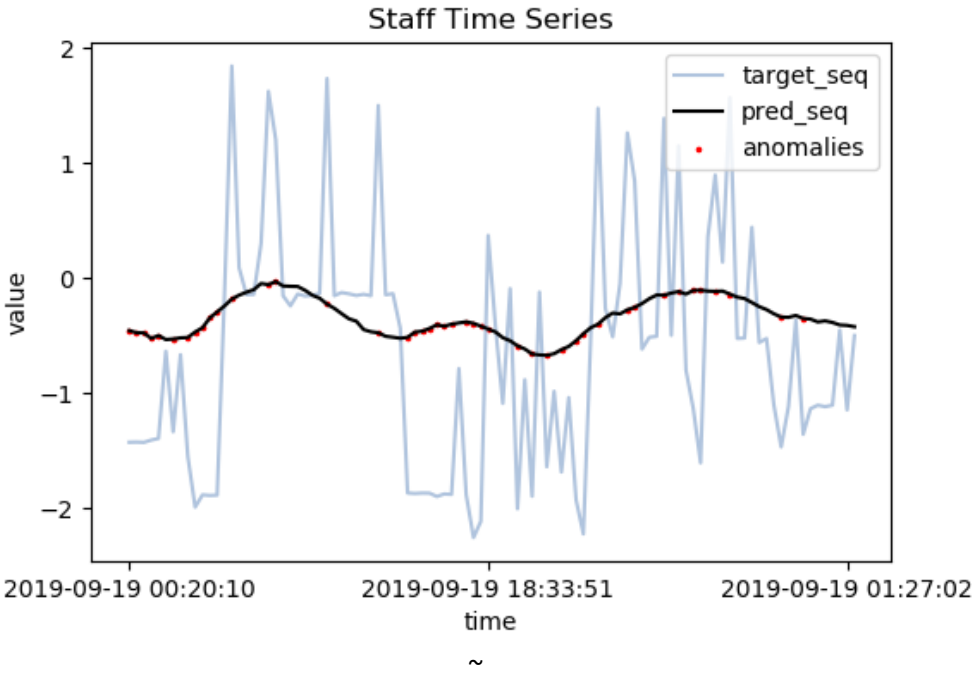

Figure 25 – Staff DeepAnT representation

8.8 Python packages used in this paper

Package	Version
pandas	0.25.3
numpy	1.17.4
matplotlib	3.1.1
sklearn	0.22.1
seaborn	0.10.0
kmodes	0.10.2
random	1.0.1
progressbar	3.51.3
keras	2.2.4
tensorflow	1.15.0

Table 4 – Python packages used in this paper