Concept Drift Detection in Production Datasets

Submitted by:

Zeina Kandil 43-12170

Tutorial Group: "BI T03"

Supervised By:

Dr. Mervat Abuelkheir

13th of January, 2021

1. Introduction

Technology has changed the world in considerable manners. In the world of production the reliance on technology increases substantially with time; from the sales forecast and identifying the number of units that need to be produced, to the logistics involved in delivering and obtaining materials. Amongst the technologies used by manufacturing corporations is machine learning. Predictive models are used in more than one context: forecasting demand, forecasting cost, analyzing the best way to utilize machines, and predicting machine breakdowns among others. Inaccurate predictions come at a heavy cost to corporations and might result in the business incurring great loss; both financial loss and opportunity loss (Dubey, Gunasekaran, Childe, Blome, & Papadopoulos, 2019: 341; Lee, Lapira, Bagheri, & Kao, 2013: 39). Nevertheless, relying on predictive analytics does not ensure that these problems will simply cease to exist. The accuracy of these models does not have a guarantee for life. This is largely due to concept drift.

In predictive analytics and machine learning, concept drift by definition is when the statistical properties of the target variable, which the model is meant to predict, change with time in unforeseen ways. As a result, the model becomes significantly less accurate as time goes by (Ditzler & Polikar, 2010: 736; Lu et al., 2018: 1). This can be caused by a multitude of reasons.

The purpose of this paper is to identify a set of concept drift detection methods, the manners in which it can be dealt with, and how all of this can be applied to production datasets. This issue is of great importance given the increasing reliance on technology and especially predictive analytics to the manufacturing industry. What with globalization and increasing competition, firms are on the lookout for anything that will give them an edge over their competitors. Extracting accurate projections and automating analytical processes using the data they have is one way for them to keep up with the market and even surpass their competitors.

This paper starts by explaining the concept behind machine learning models and how they are developed. Then, it discusses model decay in a variety of forms, with the main focus being on concept drift. Next, it proceeds to delve into more details on concept drift and its types, as well as its detection methods. Naturally, it continues with methods to avoid or mitigate the

drift of concept and its effects on the model at hand, all the while giving examples of solutions proposed in previous research.

2. Literature Review

2.1 Model Development Methods

There are plenty of methods used to produce machine learning models. This section briefly explains some of the most commonly used methods that are relevant to this paper.

Ayodele (2010: 1) stated that machine learning algorithms can be classified as one of the following types: supervised learning, unsupervised learning, semi-supervised learning, reinforcement learning, transduction, and learning to learn. In supervised learning, input is mapped to a desired output like in classification problems. Unlike supervised learning, unsupervised learning makes inferences from datasets consisting of input data without labeled responses. It is used to find hidden patterns or in grouping inputs. As the name implies, semi-supervised learning takes a bit of both of its predecessors: it takes as input a combination of labeled and unlabeled data and draws a classifier from them. Next, there's reinforcement learning, which is a field in machine learning that focuses on how an algorithm should behave in situations in the real world. Using logic, transduction infers new outputs from new inputs and previous knowledge in the shape of training inputs and their outputs. It is very similar to supervised learning, however, it does not create a function. Finally, learning to learn is when an algorithm develops its own inductive methods based on its history and previous experiences.

Moreover, machine learning models are built following a standard sequence of steps. The first building block is data processing. It comprises data selection and feature engineering. Data selection is comprehensive and takes into consideration the data's type, quality, and format. Regarding feature engineering, it is the extraction of relevant attributes for the target variable, from raw data, for algorithms to work on. Now that adequate data has been collected and formatted appropriately, a model can be built and used for analysis, Modeling has its own set of steps, which involve the selection of suitable algorithms, using the training data to model, and making accurate predictions. Lastly, model validation is done to ensure that the model behaves correctly and determine its accuracy (Wei et al., 2019: 339-345).

2.2 Causes of Model Decay

Previously accurate machine learning models used for prediction can start giving inaccurate predictions. This can be caused by a variety of reasons. In this section, data drift, covariate shift, prior probability shift, and concept drift are all explained in an effort to help identify the culprits behind a previously functioning predictive model's decay. While the main focus of this paper is the drift in concept, it is important to be able to differentiate between it and other types of shift to avoid confusion.

2.2.1 Data Drift

In several circumstances and applications, the assumption that the data distribution remains static is invalid. As a result, the efficacy of many traditional algorithms does not stand the test of time (Ren, Liao, Zhu, & Li, 2018: 262). This phenomenon is called *data drift*. One real-life application that provides an example of data drift is well-logging. The majority of machine-learning technologies addressing that topic operate under the assumption that data collected from different wells all have a single probability distribution. However, due to the sedimentary differences in environment along with the usage of different well-logging methods, they tend to have different data distributions. Consequently, models trained on old wells do not report accurate predictions for other wells, with the reason being the drift in data (Liu et al., 2020: 1). Generally speaking, when the joint distributions for datasets for training and testing are not similar, it's called data drift or *dataset shift* (Liu, Lu, Liu, & Zhang, 2018: 4).

2.2.2 Covariate Shift

Covariate shift happens to be a subset of dataset shift. It is a cause of model decay that only occurs in "X → Y" scenarios. A shift in the independent variables "X" translates into covariate shift. In other words, the relation between "X" and "Y" remains unchanged (Liu et al., 2018: 4). In the context of multi-modal Gaussian distributions, covariate shift has two forms: mode collapse and boundary distortion. Mode collapse can be explained as the scenario when a large portion of the probability mass is concentrated on select few modes from the distribution, which in turn causes loss of diversity. The other phonomona, boundary distortion, reduces the diversity of the periphery of the learned distribution. (Santurkar, Schmidt, & Madry, 2018: 4482).

2.2.3 Prior Probability Shift

Again, this type of shift is a subcategory of dataset shift. However, unlike covariate shift, prior probability shift focuses on changes in the distribution of the target variable (Ren et al., 2018: 264). Thus, it is essentially the reverse of covariate shift. This particular type of dataset shift is associated with "Y → X" situations and is commonly linked to Naive Bayes. A prior probability is the probability derived using general knowledge without taking its attribute values into consideration. As for prior probability shift, the name is pretty self explanatory; a model can become tuned to a certain probability distribution following the dataset it was previously trained on. As a result, when it is tested on a dataset that should give off a different probability distribution it fails to do so (Sun, Tang, Zhu, & Yao, 2018: 2; Wang et al., 2013: 2).

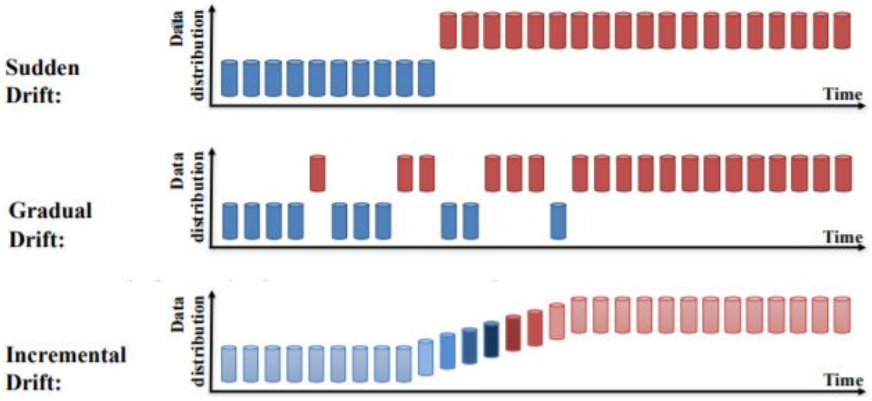
2.2.4 Concept Drift

Informally, concept drift is described as a change in the concept definition over time that is reflected in the data distribution of said concept (Elwell & Polikar, 2011: 1518). Concept drift can be defined by Bayes' theorem, $P(y|x) = P(x|y) \times P(y)/P(x)$, where the probabilistic value, $P(y|x)$, is the posterior probability, $P(x|y)$ is the class-conditional probability, $P(y)$ represents the prior probability and $P(x)$ the feature probability. According to the Bayes' theorem, true drift in concept is related to a change in the posterior probability. Given that $P(x)$ is the input value's data distribution, changes in the feature or x can be detected through it. Previous works call changes in the feature probability *virtual concept drift*. Nevertheless, monitoring such variations is inadequate. The class-conditional probability depicts the potentiality of finding a data point x in class y. In essence, after understanding Bayes' theorem, veritable concept drift is the dynamic disposition observed by the probabilistic value (Liu et al., 2018: 4; Ren et al., 2018: 263). Simply put, changes in the hidden context that result in drastic changes in the target concepts are what scholars deem as concept drift (Widmer & Kubat, 1996 : 70).

As it stands, there are numerous types of concept drift. Previous research states that the majority of existing streaming classifiers detect just one of the types (Ren et al., 2018: 262). In the coming sub-sections, different forms of concept drift will be explained to aid in understanding how to detect them later on.

2.2.4.1 Abrupt vs Gradual vs Incremental

Concept drift, if classified based on the severity of the rate of its occurrence, can fall into three categories: abrupt, gradual or incremental. Firstly, abrupt or sudden drift is when a new concept arises in a short period of time, instantly replacing the previous concept making it no longer applicable (Barros & Santos, 2018: 1). Secondly, gradual concept drift is when a new concept keeps replacing an old one over a period of time until the old concept is no longer relevant. The presence of the old concept decreases while the occurrence of the new concept increases until it becomes dominating (Escovedo, Koshiyama, Cruz, & Vellasco, 2018: 2; Hoens, Polikar & Chawla, 2012, 92). Lastly, incremental drift occurs when a new concept takes small incremental steps in replacing an old one. The diagram to the right illustrates data distribution against time for each of the three categories (Lu et al., 2018: 3).



2.2.4.2 Real vs Virtual

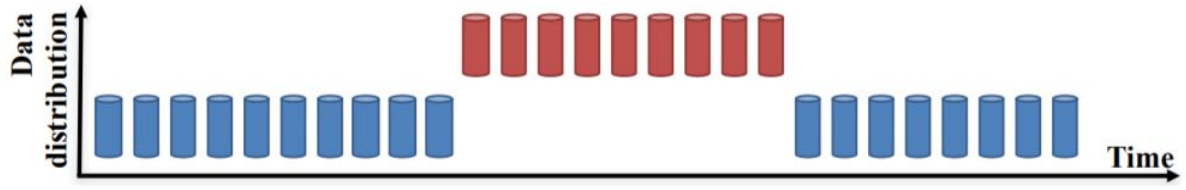
Furthermore, concept drift can be classified as either real drift or virtual drift. On one hand, virtual concept drift is when the underlying concept remains static regardless of variations in the distribution of the feature, P(x), or the a priori probability, P(y), over time (Lu et al., 2017: 2393). Nonetheless, this is problematic due to the fact that it can result in changes in errors in the model even though the concept itself has not changed. Often, the model will need to be retrained with the new data in the target concept. On the other hand, real concept drift is known as a change in the posterior distribution or in the class boundary. Furthermore, real concept drift can be labeled as *severe* if all instances have different classes before and after the drift. Apart from that, it is referred to as *intersected*. (Almeida, Oliveira, Britto, and Sabourin, 2018: 68; Hoens et al., 2012, 91-92; Ren et al., 2018: 263).

2.2.4.3 Adversarial Concept Drift

When drift is intentional with the goal of deteriorating the system it is called *adversarial concept drift*. The bulk of concept drift remedies treat the drift as a domain-independent issue for the method to be widely applicable. Their approach, normally, is to naively retrain the model. However, that makes them more susceptible to fall in traps made by active adversaries, who try to elude the deployed classification systems in ways that are undetectable by conventional concept drift detection systems (Sethi & Kantardzic, 2018: 1).

2.2.4.4 Recurring Concept Drift

In some cases, a concept that was once nullified by a new concept can make a comeback and become relevant once again. In certain topics, where concepts can be recurring, it makes sense to consider historical concepts and find the most relevant concept. Hence, the higher the chances of recurrence the more significant the historical concepts become. Note that recurring concepts can also drift abruptly, gradually, or incrementally. Their reappearance can be cyclical or random (Escovedo et al., 2018: 5; Widmer & Kubat, 1996: 16). The following diagram conveys how a concept can become recurring (Lu et al., 2018: 3).



2.3 Concept Drift Detection

The previous section indicated the effects of concept drift, which emphasizes the importance of identifying it; the quicker the better. This is done by *concept drift detectors*, which are programs that estimate the time of these changes to replace the base learner and ultimately improve overall accuracy. They seek to detect the point in time in which change occurs (Barros & Santos, 2018: 1: Lu et al., 2018: 4).

2.3.1 Active vs Passive Drift Detection Mechanisms

Typically drift detection mechanisms can be classified as either active or passive (Elwell & Polikar, 2011: 1520).

On one side, an active drift detection mechanism utilizes triggers to diagnose concept drift as a change in the statistics of the data-generating process (Liu et al., 2018: 2). Moreover, active approaches tend to abandon accumulated knowledge up to the point in time during which drift was detected (Losing, Hammer & Wersing, 2016: 292).

In non-stationary environments, a widely used technique for data classification is the sliding window approach. A dynamic sliding window is a method used to efficiently observe the data against its error history in a window that has all data since the previous incident of detected drift. This window is divided into two sub-windows over and over again and whenever their mean error surpasses a certain threshold the older window is disposed of or forgotten (Widmer & Kubat: 1996: 24). Consequently, the size of the sliding window is dependent on the drift rate. Hence, if a sliding window of modifiable size is used, it is considered an active approach, that is continuously on the lookout for any changes. Should any drift be detected, The existing model is updated with the purpose of decreasing the relevance of old data and emphasizing the significance of new ones. As a result, the window size becomes smaller with higher drift rates; the diminishing relevance of old data lessens the size of relevant data. Oppositely, when a model has little or rare occurrences of concept drift they tend to have larger window sizes. This is where the trade-off takes place; with a small window comes speed and efficiency whereas larger windows guarantee generalization in stable phases without drift (Ditzler & Polikar, 2010: 736; Losing et al., 2016: 292).

There are several algorithms that implement this concept in discussion. Amongst the instigators of the dynamic window size strategy is the FLOating Rough Approximation (FLORA) family of algorithms (Ditzler & Polikar, 2010: 736; Widmer & Kubat, 1996: 4). Recent algorithms that follow the FLORA framework include ADaptive sliding WINdowing (ADWIN) and Probabilistic Adaptive Windowing (PAW). ADWIN efficiently keeps track of the error history in a window that holds all data since the last bout of change. As for PAW, it relies heavily on random sampling. PAW extracts instances from the window, including recent and old data. The size of the window is dynamic. It utilizes ADWIN as its change detector and is often coupled with the k-Nearest Neighbors (kNN) classifier, obtaining low error rates in trials (Losing et al., 2016: 292; Ren et al., 2018: 264).

On the other side, a passive approach focuses on constantly updating the model even in scenarios where no drift has been detected, which helps prevent problems before they

happen. Furthermore, a passive detection method avoids having false positives and false negatives in drift detection systems. To the contrary of the sliding window technique, the computational cost of a passive concept drift detection method is somewhat constant, resulting in expensive delays should any sudden drift occurs (Ghazikhani, Monsefi & Yazdi, 2014: 51; Losing et al., 2016: 292).

Implementations of passive drift detection mechanisms mainly involve ensembles and on a more specific level: tree-based models. For example, there is the Very Fast Decision Tree (VFDT) that handles streaming data with the aid of decision trees. VFDTs add splits in increments as dictated by the Hoeffding bound, guaranteeing with high confidence that the sample will produce the same split attribute as the whole dataset (Losing et al., 2016: 292). To clarify, the Hoeffding bound derives confidence thresholds on the distribution's average, using statistical methods. Conveniently, the Hoeffding bounds stand regardless of the distribution itself. Thereby, when a certain confidence level is desired, one can directly compute the number of instances needed to achieve the same result that would be obtained should all of the data be used (Hoens et al., 2012: 94)

Another algorithm for the passive approach is the Dynamic Adaption to Concept Changes (DACC). The idea of DACC stems from the Dynamic Weighted Majority (DWM) method. The sample is split in two not necessarily equal partitions and a classifier of the worst partition is deleted randomly. The deletion occurs after a given number of iterations and is followed by a replacement. For a certain number of iterations, the new examples are exempted from this elimination process. More importantly, predictions are only made by the best classifier in the pool, which has a history of the highest accuracy. DACC is bound to perform adeptly with abrupt and incremental scenarios of concept drift (Losing et al., 2016: 292-293).

2.3.2 Ensemble Classifiers vs Single Classifiers in Concept Drift Detection

Coupling concept drift detection methods with base classifiers is a commonly used recipe when dealing with data streams. Some of the popular detectors that do that are the Drift Detection Method (DDM), the Early Drift Detection Method (EDDM), and the Statistical Test of Equal Proportions (STEPD) (Barros & Santos, 2018: 2).

DDM analyzes the error rate and its standard deviation to identify concept drift. It computes the standard deviation as $s_i = \sqrt{p_i \times (1 - p_i)/i}$, where p_i is the error rate for position i. To clarify, DDM considers the computed standard deviation as the tolerance zone (Escovedo et al, 2018: 4). Scholars claim that if the distribution of the sample subset remains constant, the error rate should have an inverse relationship with the size of the sample subset. Accordingly, should the error increase it is deduced that the data distribution has changed and that the current base learner is no longer reliable (Barros & Santos, 2018: 4; Lu et al., 2018: 4; Wang et al., 2013: 2).

Similarly, EDDM operates like DDM except that it observes the difference between two succeeding error rates, instead of the error rate itself. Thus, EDDM is more fit for detecting gradual concept drift, while DDM is more apt in identifying sudden concept drift (Barros & Santos, 2018: 4; Ren et al., 2018: 264).

On the other hand, STEPD is a bit disparate from DDM and EDDM. Yet, it is similar to DACC in the way that it splits the processed data over two partitions. However, that's where the similarities end. A statistical test of equal proportions is applied to the two windows, with continuity correction. Note that the windows are of different sizes. For the same concept, the accuracies should be the same in both the recent window and the older window. Hence, when the accuracies are significantly different, a warning is set off to alert the system of the occurrence of concept drift (Barros & Santos, 2018: 4-5; Escovedo et al., 2018: 4; Lu et al., 2018: 5; Yu & Abraham, 2017: 768).

Furthermore, a different approach to detect concept drift is via single classifiers. They adjust the existing model with the assistance of adaptive mechanisms like the sliding window approach and drift detectors. Algorithms that apply this technique include the previously discussed ADWIN as well as the unified instance selection algorithm (FISH). In order to adapt to changing environments, single classifiers continuously classify themselves. In comparison to ensemble classifiers, single classifiers consume much less time and memory, but they are marginally less accurate (Ren et al., 2018: 264).

Further still, some algorithms use a hybrid ensemble; they take features from more than one system. For instance, the Adaptive Classifier Ensemble (ACE) and Accuracy Updated Ensemble (AUE) can be merged into one hybrid mechanism. On one hand, ACE updates the model using a fixed-size data block whenever sudden drift is observed through the error rate

of a single classifier. On the other hand, AUE updates past ensembles with data from more recent blocks, incrementally (Ren et al., 2018: 264).

2.4 Addressing Concept Drift

Concept drift detection would be futile if no action was taken following its discovery. There are several approaches to remedy or mitigate the effects of concept drift. Several of them have constraints that render them useless in many scenarios. Recent studies have worked extensively on generating new approaches to handle concept drift. These techniques revolve around incremental learning, periodical retraining of the model, the use of neural networks, and ensemble learning. Occasionally, researchers experiment with a mechanism that uses a combination of these techniques. The following subsections go briefly through a few of these approaches.

2.4.1 Incremental Learning

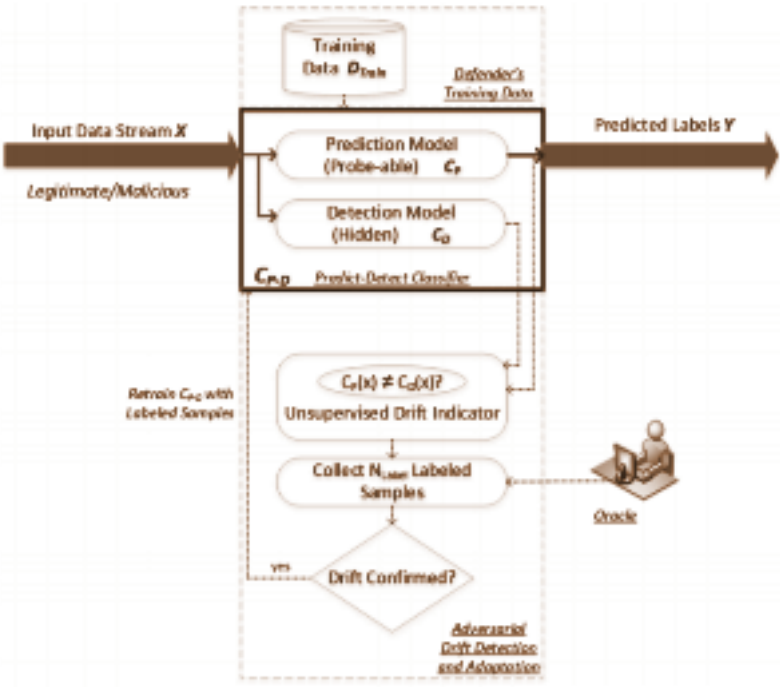
In the technical context, incremental learning is a method of machine learning where the model continuously learns from input data, increasing its knowledge and the model is adjusted whenever new examples emerge. That is to say, the model learns incrementally without needing to refer to previously seen data. Incremental learning is especially useful in cases where old data is not retrievable. However, this brings about the stability-plasticity dilemma (Mirza, Lin, & Liu, 2015: 316). Stability means retaining old knowledge that is still relevant such as historical concepts in cyclical environments. Whereas plasticity means the capacity to learn new knowledge. Achieving the right stability-plasticity balance is a challenge of its own (Elwell & Polikar, 2011: 1517). Previous research discusses different ways in which incremental learning can be used to assuage concept drift.

An environment is deemed non-stationary when its underlying data distribution varies over time. Most algorithms for learning in a non-stationary environment or *learning concept drift* work in limiting circumstances. They often only work on gradual or incremental drift, non-recurring concept drift, or scenarios where no new classes appear. In addition, usually, they were not tested on truly non-stationary environments. Learn^{++}.NSE, a member of the Learn^{++} family of algorithms, is a framework that places none of these restrictions. Although the Learn^{++} family is based on ensemble learning, a new classifier is trained with each arriving bit of data (Mirza et al., 2015: 316). Thus, it is also an incremental learning

mechanism. The focal point for this family of algorithms is that all members are incrementally trained classifiers with a weighted majority voting involved in some way or another. What differentiates them is how the voting weights are decided on and the conditions in place for choosing data for training ensemble members.

Learn^{++}.NSE is the only one, amongst its predecessors, that can learn in non-stationary environments. It follows a passive drift detection mechanism. After extensive research, the following was concluded regarding Learn^{++}.NSE. Firstly, the type of the base classifier does not affect its performance; be it MLP, SVM, or Naive Bayes. Secondly, it works better for slower rates of drift, and thirdly, no pruning results in superior results (Elwell & Polikar, 2011: 1517-1518).

On another spectrum, the Predict-Detect framework applies concepts of incremental learning to handle concept drift. This method was developed for streaming data with adversarial concept drift in mind (Sethi & Kantardzic, 2018: 1). It takes unlabeled data samples, X, as input and outputs the predicted labeled stream, Y. This framework has an unsupervised drift indication unit to detect any significant drifts in X. If it is detected, more labels, N_{Labels}, are collected from an independent Oracle, and are used to authenticate that concept drift has truly occured. If a consequential drop in accuracy of the predicted performance is observed, then adversarial concept drift has occurred. In that case, the labeled samples are used to retrain the model. Using incremental learning is advantageous because it allows old data to remain hidden from adversaries in case of an attack. Another plus for this system is that it uses unlabeled data efficiently; since the last labeled sample is not referred to unless there is a likelihood of drift, if drifts rarely occur, no drift will be indicated and thereby there is no wasteful usage of



labels. The diagram displayed provides an overview of the Predict-Detect stream classification framework (Sethi & Kantardzic, 2018: 8).

2.4.2 Periodical Retraining

Periodical retraining of the model is a convention to keep the model up to date and aware of new concepts (Yu & Abraham, 2018: 768). It can be performed on the dataset as a whole or on a sample subset of it and that is referred to as periodical retraining with sampling. Regarding adversaries, retraining can provide protection against adversarial attacks because of how it counters their ability to adapt to new detectors. In spite of that, periodical retraining does not provide a fool-proof mechanism against rivals who seek to poison the model with corrupt input. Non-stationarity makes an environment more vulnerable to attacks, since training provides a window for adversaries to station their strikes. Retraining the model periodically makes an environment even more susceptible to attacks considering that adversaries can anticipate retraining (Huang, Joseph, Nelson, Rubinstein, & Tygar, 2011: 1,9).

2.4.3 Online Neural Network

Supervised classifiers encounter challenges in the form of concept drift and class imbalance. While concept drift has been explained earlier, class imbalance is when the number of instances isn't distributed equally among classes; on the contrary, the imbalance is significant. In 2014, an online neural network model that addresses both issues was proposed. A forgetting approach is used to deal with concept drift. Meanwhile, an error function is used to assign weights to errors in different classes. On the other hand, preliminary models concentrated on batch solutions on sample subsets. The main difference between online and batch learning lies in the number of instances used to train the classifier: a single instance vs a batch (Ghazikhani et al., 2014: 51).

The proposed method for Imbalanced Data Stream Classification is an online MultiLayer Perceptron (MLP) Neural Network model, which has N inputs, H hidden, and M outputs. This approach defines the independent variable at training pattern p as $x^{(p)}$ and outputs $h_j^{(p)}$ at the hidden node. The first part of the neural network, the one in charge of handling concept drift, which is the forgetting function $f(p)$ has a general form of $f(p) = \alpha p + \beta$. Its job is to assign weights to recent instances to convert the existing model into a classification model

that abides with the new concepts. The second component is the error function. The error function used classifies imbalanced datasets into two classes, that have tangent hyperbolic activation functions. It amplifies error signals in the minority class. Thus, it permits handling of imbalanced datasets. The results of experimentation show that this online MLP model has shown greater accuracy on higher imbalance ratios. Moreover, it has shown greater efficiency than previous remedies for both class imbalance and concept drift (Ghazikhani et al., 2014: 51-61).

2.4.4 Ensemble Learning With Model Weighting

Ensemble learning, a supervised machine learning approach, involves merging multiple machine learning algorithms to enhance the predictive performance of the model. It is very popular to use ensemble learning when addressing concept drift. To help bypass troubles caused by concept drift, several models can be ensembled and the output would be a weighted average of outputs produced by distinct models. Moreover, ensemble methods tend to have the capacity to adapt to concept drift related problems better than single classifiers. (Lin, Deng, Kuo, & Chen, 2019: 56200; Mirza at al., 2015: 317; Ren et al., 2018: 261).

Early references mention an ensemble learning mechanism, that is based on a Diversity for Dealing with Drifts (DDD) (Barros & Santos, 2018: 2; Sun et al., 2018: 2). Provided that concept drift has been detected, the DDD generates two new classifiers and trains one of them on the datasets with high diversity and the other on the dataset with low diversity. Then, it integrates the new classifiers with the original classifiers. Unfortunately, its weighting scheme was designed to handle online concept drift only. Later on, in an application of ensemble learning concepts, multiple classifiers were used instead of a single classifier. The outcome proved that using multiple classifiers enhanced the performance of the model, in terms of handling concept drift (Lin et al., 2019: 56200).

In 2019, an ensemble learning method, a dynamic AdaBoost.NC with multiple subclassifiers for imbalance and drifts (DAMSID), based on DDD, was proposed. Like MLP, DAMSID is designed to handle concept drift as well as class imbalance. DAMSID is based on several improvements on the Adaptive Boosting (AdaBoost) ensemble learning algorithm. First, AdaBoost was consolidated with a negative correlation learning method, making AdaBoost.NC, that performed better with classification problems. Next, a Dynamic

AdaBoost.NC predictive model was created by adding to AdaBoost.NC, a feature that automatically adjusts the training parameters. This additional feature significantly reduced the training time and boosted the overall performance (Lin et al., 2019: 56200-56201). The latest version, DAMSID, will be discussed in more detail.

DAMSID consists of three stages. The first stage is to train an ensemble classifier. It is basically an incorporation of the Dynamic AdaBoost.NC along with SMOTE, where SMOTE is the component in one of the members of the Learn^{++} family of algorithms. Its role is managing class imbalance (Ditzler & Polikar, 2010: 737). With respect to the second stage, it is responsible of concept drift detection in imbalanced data. It does so by implementing a method called *Linear Four Rates*; the dataset is simulated with four levels of imbalance to test DAMSID's performance in each scenario. The third and final stage utilizes the Dynamic AdaBoost.NC, again, to build a model that works on the new concept (Lin et al., 2019: 56201).

Additionally, DAMSID was tested on a production dataset for condition-based maintenance (CBM), which forecasts when machines are likely to crash. This dataset is subject to concept drift because the fault patterns may change. Since the majority of companies cannot afford the infrastructure required for real-time online classifiers. Instead they usually have off-the-shelf offline classifiers. With the surge of Industrial IOT, manufacturing firms can continuously gather data from their machines, which generates big data. Given that DAMSID can be trained on offline classifiers, it solved the CBM concept drift dilemma. After testing, it was concluded that DAMSID was capable of detecting all drifts in concept effectively. With DAMSID the accuracy of predictions passed 90%. (Lin et al., 2019: 56198, 56206).

Another interesting approach that uses ensemble learning that was proposed prior to DAMSID is the ensemble of subset online sequential extreme learning machine (ESOS-ELM) (Lu et al., 2017: 2393: Lu et al., 2018: 13). Once again, this technique was built with both concept drift and class imbalance in mind. Unlike DAMSID, ESOS-ELM only works for data streams. There are three blocks that make up the ESOS-ELM system and they are as follows: the main ensemble block, the ELM-Store block, and the change detector (Mirza et al., 2015: 316).

Firstly, the main ensemble is also referred to as the short term memory. In it, many online sequential ELM networks are initialized. Then, a balanced subset from the original dataset is

used to train each OS-ELM network. Furthermore, ESOS-ELM handles the majority class sample with a single classifier. Whereas the minority class sample is processed by *m* classifiers, such that *m* is equivalent to the imbalance ratio. Each majority class sample is processed by a distinct classifier in a 1:1 relationship. This ensures that every single classifier in the ensemble was trained on a balanced subset of the unbalanced data stream. Secondly, the ELM-Store, which is also known as the long-term memory, is for saving old information. For the sake of benefitting from historical knowledge of recurring concepts, it should be maintained in an orderly fashion. With the main ensemble acting as a short-term memory, the ELM-Store's main focus needs to be on older data. It does so via batch classifiers that do not get retrained unless their ensemble performance drops below a certain threshold, signaling that concept drift has occurred. Thirdly, the change detector has to detect both sudden and gradual concept drift. Sudden drift is detected by monitoring the main ensemble against a predefined margin. Meanwhile, gradual drift is detected using a statistical decision theory (Mirza at al., 2015: 318-320).

It was concluded that ESOS-ELM can work accurately in stationary and non-stationary environments, regardless of the occurrence of concept drift. In addition, it even performed well with recurring concepts thanks to the ELM-Store unit, which proved to be computationally efficient (Mirza at al., 2015: 328).

2.5 Research Gap

From exploring the previous literature review, a research gap has been noticed. There was insufficient research dedicated to studying how concept drift detection in production datasets affects the success of manufacturing companies. As indicated in this paper concept drift methods often need to be case specific and tailored to the problem at hand. Recent technological advances have changed the commercial production world greatly. For example, automated recording of data has been enabled by sensors, storage, surveillance cameras, processing remote payments and communication technologies. This generates a data stream such as web logs and web page click streams. It is a scenario evolving huge amounts of data and is subject to concept drift for a multitude of reasons. Therefore, the research question that this paper aims to answer is: "What is the effect of accurate concept drift detection and mitigation in production datasets on the success of manufacturing firms?"

Hypothesis: Accurate concept drift detection and mitigation in production datasets has a positive effect on the success of manufacturing firms.

3. References

Almeida, P. R., Oliveira, L. S., Britto Jr, A. S., & Sabourin, R. (2018). Adapting dynamic classifier selection for concept drift. Expert Systems with Applications, 104, 67-85.

Ayodele, T. O. (2010). Types of machine learning algorithms. New advances in machine learning, 3, 19-48.

Barros, R. S. M., & Santos, S. G. T. C. (2018). A large-scale comparison of concept drift detectors. Information Sciences, 451, 348-370.

Ditzler, G., & Polikar, R. (2010, July). An ensemble based incremental learning framework for concept drift and class imbalance. In The 2010 International Joint Conference on Neural Networks (IJCNN) (pp. 1-8). IEEE.

Dubey, R., Gunasekaran, A., Childe, S. J., Blome, C., & Papadopoulos, T. (2019). Big data and predictive analytics and manufacturing performance: integrating institutional theory, resource‐based view and big data culture. British Journal of Management, 30(2), 341-361.

Elwell, R., & Polikar, R. (2011). Incremental learning of concept drift in nonstationary environments. IEEE Transactions on Neural Networks, 22(10), 1517-1531.

Escovedo, T., Koshiyama, A., da Cruz, A. A., & Vellasco, M. (2018). DetectA: abrupt concept drift detection in non-stationary environments. Applied Soft Computing, 62, 119-133.

Ghazikhani, A., Monsefi, R., & Yazdi, H. S. (2014). Online neural network model for non-stationary and imbalanced data stream classification. International Journal of Machine Learning and Cybernetics, 5(1), 51-62.

Hoens, T. R., Polikar, R., & Chawla, N. V. (2012). Learning from streaming data with concept drift and imbalance: an overview. Progress in Artificial Intelligence, 1(1), 89-101.

Huang, L., Joseph, A. D., Nelson, B., Rubinstein, B. I., & Tygar, J. D. (2011, October). Adversarial machine learning. In Proceedings of the 4th ACM workshop on Security and artificial intelligence (pp. 43-58).

Lee, J., Lapira, E., Bagheri, B., & Kao, H. A. (2013). Recent advances and trends in predictive manufacturing systems in big data environment. Manufacturing letters, 1(1), 38-41.

Lin, C. C., Deng, D. J., Kuo, C. H., & Chen, L. (2019). Concept drift detection and adaption in big imbalance industrial IoT data using an ensemble learning method of offline classifiers. IEEE Access, 7, 56198-56207.

Liu, A., Lu, J., Liu, F., & Zhang, G. (2018). Accumulating regional density dissimilarity for concept drift detection in data streams. Pattern Recognition, 76, 256-272.

Liu, H., Wu, Y., Cao, Y., Lv, W., Han, H., Li, Z., & Chang, J. (2020). Well Logging Based Lithology Identification Model Establishment Under Data Drift: A Transfer Learning Method. Sensors, 20(13), 3643.

Losing, V., Hammer, B., & Wersing, H. (2016, December). KNN classifier with self adjusting memory for heterogeneous concept drift. In 2016 IEEE 16th international conference on data mining (ICDM) (pp. 291-300). IEEE.

Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., & Zhang, G. (2018). Learning under concept drift: A review. IEEE Transactions on Knowledge and Data Engineering, 31(12), 2346-2363.

Lu, Y., Cheung, Y. M., & Tang, Y. Y. (2017, August). Dynamic Weighted Majority for Incremental Learning of Imbalanced Data Streams with Concept Drift. In IJCAI (pp. 2393-2399).

Mirza, B., Lin, Z., & Liu, N. (2015). Ensemble of subset online sequential extreme learning machine for class imbalance and concept drift. Neurocomputing, 149, 316-329.

Ren, S., Liao, B., Zhu, W., & Li, K. (2018). Knowledge-maximized ensemble algorithm for different types of concept drift. Information Sciences, 430, 261-281.

Santurkar, S., Schmidt, L., & Madry, A. (2018, July). A classification-based study of covariate shift in gan distributions. In International Conference on Machine Learning (pp. 4480-4489). PMLR.

Sethi, T. S., & Kantardzic, M. (2018). Handling adversarial concept drift in streaming data. Expert systems with applications, 97, 18-40.

Sun, Y., Tang, K., Zhu, Z., & Yao, X. (2018). Concept drift adaptation by exploiting historical knowledge. IEEE transactions on neural networks and learning systems, 29(10), 4822-4832.

Wang, S., Minku, L. L., Ghezzi, D., Caltabiano, D., Tino, P., & Yao, X. (2013, August). Concept drift detection for online class imbalance learning. In The 2013 International Joint Conference on Neural Networks (IJCNN) (pp. 1-10). IEEE.

Wei, J., Chu, X., Sun, X. Y., Xu, K., Deng, H. X., Chen, J., ... & Lei, M. (2019). Machine learning in materials science. InfoMat, 1(3), 338-358.

Widmer, G., & Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts. Machine learning, 23(1), 69-101.

Yu, S., & Abraham, Z. (2017, June). Concept drift detection with hierarchical hypothesis testing. In Proceedings of the 2017 SIAM International Conference on Data Mining (pp. 768-776). Society for Industrial and Applied Mathematics.