Explaining concept drifts in data streams and visual tools / time series predictions

ANILCAN POLAT, JONAS KRIEGEL, KATHRIN WILMS

Data streams and time series predictions deal with different types of data, one dynamic the other static. However, both data types have a temporal dimension, leading to different challenges. Data streams have to deal with concept drifts - with time the relationship between features and labels can change [6]. Time series predictions, where understanding why a machine learning model made its decision is important, a usual visualization method - saliency methods, start to fail [3]. In this report we aim to explain concept drifts by looking at what a context is and what type of and pattern of concept drift exist and a theoretical framework [8] to mitigate them. Afterwards we focus on time series predictions and discuss two methods to make them interpretable: Temporal Saliency Rescaling[3] and Dynamask[1] and come to the conclusion that Dynamask is the better method. We then propose to use time series predictions to help with concept drifts.

ACM Reference Format:
Anilcan Polat, Jonas Kriegel, Kathrin Wilms. 2024. Explaining concept drifts in data streams and visual tools / time series predictions . 1, 1 (January 2024), 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Looking at the application of Machine Learning (ML) models in security, healthcare and finance, having to deal with data streams - for example, financial transactions streaming in real-time, is not unusual. With data streams being dynamically and continuously flowing infinite series of elements [4], one issue becomes prominent - concept drifts. Concept drifts describe the problem where with time the relationship between features and label change [6]. The consequence to this is a drop in accuracy, making the model unreliable. Therefore models dealing with dynamical and evolving data streams have to be robust enough to recognize patterns and shifts in relationships over time. However, data-streams aren't the only challenge in data analytics. Just like data-streams, time series predictions involve a temporal dimension. The difference with time series predictions is that they're a collection of data of observations or measurements taken at sequential, discrete time points [5]. Therefore time series predictions don't represent a continuous flow of data over time, but static data. Applications of time series predictions are also found in high stake domains, especially to forecast trends. One example would be investors making decisions based on the analysis of time series prediction models analysing historical stock data. As these models play an important role in critical decision-making processes, transparency and interpretability are necessary for building trust and ensuring the responsible application of ML models. Commonly used methods are saliency methods, used to identify and emphasize the most relevant or important features for the prediction. However, since time series predictions have a temporal dimension, usual saliency methods stop working[3].

In this report we aim to explain concept drifts in data streams and visual tools / time series predictions. To achieve this, we first focus on data streams, take a more detailed look at concept

Author's address: Anilcan Polat, Jonas Kriegel, Kathrin Wilms, .

XXXX-XXXX/2024/1-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

drifts and how to detect and mitigate them. After that, we shift our focus to time series prediction, understand why usual saliency methods don't work and discuss two approaches trying to solve the issue - Temporal Saliency Rescaling and Dynamask. We then aim to bring the two topics together by proposing to use the idea behind Temporal Saliency Rescaling and Dynamask to help with data streams and concept drifts.

2 UNDERSTANDING CONCEPT DRIFTS IN DATA STREAMS

Concept Drifts are a type of distribution shift and therefore a form of "epistemic uncertainty", meaning the uncertainty is caused by a lack of knowledge[6]. In this case the lack of knowledge stems from the model not knowing the current state of the data. The underlying distribution of the data changed over time[6]. This can lead to less accuracy and an unreliable model.

To fully understand what a concept drift is, we first have to look at how a "concept" is defined in the first place. Based on (Geoffrey I. W. at al., 2016) paper "Characterizing concept drift" [7] it is described that traditionally, concept stands for a set of objects- either as a set of X values, where any object with those values belongs to the concept, or as a set of object instances. However, this excludes cases where different objects with the same values might belong to different concepts or where instances may only appear once, such as in data streams. Recent concept drift research uses a probabilistic definition[2, 7]:

$$\text{Concept} = P_t(\mathcal{X}) \tag{1}$$

P($\mathcal{X}$) stands for P(X,Y). P(Y) is the prior class probability for the class Y, meaning it doesn't take into account information provided by features. P(X|Y) represents the probability of observing feature set X given that the instance belongs to class Y. Now, both the prior class probabilities P(Y) and class conditional probabilities P(X|Y) determines the joint distribution P(X,Y)[]. t stands for the concept's time. Based on this definition a concept drift happens when, given the times t and z the distribution changes:

$$P_t(\mathcal{X}) \neq P_z(\mathcal{X}) \tag{2}$$

Unlike other types of distribution shift, concept drifts aren't caused by sampling biases, but a measurement bias[6], since relationships between features and labels change with time. To illustrate this, we can imagine a situation where an ML model is trained on historical stock data from a stable period. The model effectively predicts stock prices, but a sudden event, such as a pandemic starts impacting the world and market dynamics change. The ML model becomes unreliable. Once again, the uncertainty about the future is important to highlight. The pandemic isn't something that was known beforehand. Otherwise, we could create multiple models trained on separate data - with one including data from historical data collected while another pandemic was happening.
We can also formulize a concept drifts in two other ways [6]:

$$P_{Y|X}^{(train)}(y|x) \neq P_{Y|X}^{(deploy)}(y|x) \textbf{ and } P_X^{(train)}(y|x) = P_X^{(deploy)}(y|x) \tag{3}$$

In this case, the labels given the features are different but the features are the same. To illustrate this, we can imagine a scenario where in a spam filter, $P_{Y|X}^{(train)}$(spam|mail) changes at deployment because the user for a certain time period gains interest in a topic and therefore advertisement about it is not considered spam anymore. However, the types of emails being encountered remain the same.
Another situation would be when the features given the labels are different but the labels are the same.

$$P_{X|Y}^{(train)}(x|y) \neq P_{X|Y}^{(deploy)}(x|y) \textbf{ and } P_Y^{(train)}(x|y) = P_Y^{(deploy)}(x|y) \tag{4}$$

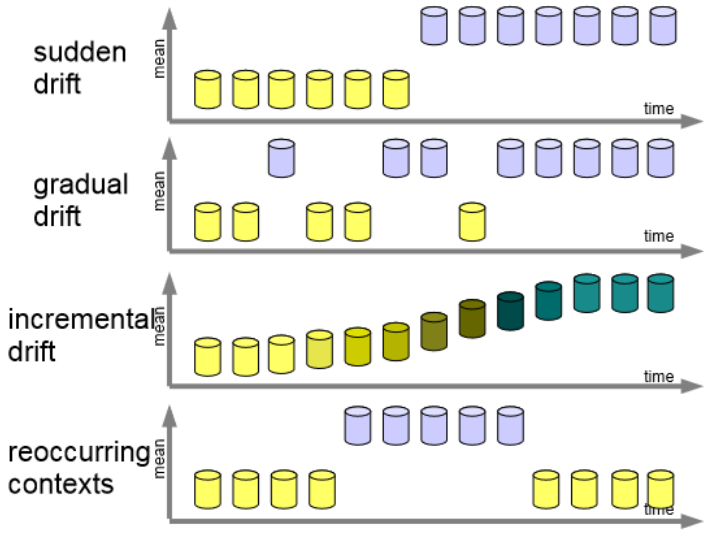

Fig. 1. The four structural types of Concept Drifts [8]

Here, the probability of the training feature X given the label Y is not the same as in the deployment, however the distribution of the label Y stays the same. To illustrate this we can imagine a model to predict a disease $P_{Y|X}^{(train)}$(Presence|Symptom). Due to a mutation in the virus new symptoms, like a fever, now is also a symptom for the sickness. Now other features should predict the presence of the disease, but since the model didn't notice the change of relationship between features and labels a concept drift occurs.

2.1 Structural types of Concept Drifts

Not only can we look at **what** concept drifts change, but also **when**. As shown in Figure 4, 4 types [8] can be pointed out.
The first type, sudden drift happens when the information source, the model was trained on S suddenly gets replaced at time t_0 by S'. One example for this drift would be a scenario where a ML model is used for online content recommendation. At time t_0 the model is accustomed to recommending articles related to cats. However, based on spontaneously getting a dog, the user suddenly is interested in dog articles.
The second type, gradual drift, explains the situation where the model samples from two sources S_1 and S_2, but the likelihood to rely on S_1 lessens, while the likelihood to rely on S_2 gets higher. Taking the example from before, in a gradual drift, the user doesn't get spontaneously a dog, but first thinks about it - slowly reading continuously more about dogs.
The third type, incremental drift, describes the situation where again two sources are being used, but the difference between the sources are so minimal, that only a longer time period shows the concept drift. Taking again the example from before, in an incremental drift, the user doesn't get spontaneously a dog, but finds interest in one specific cat breed, slowly starting to only look up articles about that breed.

The forth type reoccurring contexts, happens when time shows that concepts repeat after some time passed. However, the time passed isn't periodic but its uncertain when the context comes back. One example would be only looking up cat articles when having to look after the neighbors' cat. After looking at these four types of structural concept drifts, its important to mention that, just like the author of the paper as well mentioned, these aren't the only existing types. With data streams may being endless, there is an infinite number of pattern changes.

2.2 Detecting Concept Drifts

Based on the work of (Kush R. Varshney, 2022) book "Trustworthy Machine Learning" [6] if data from the new context is already available, two types of shift detection can be applied. The first one is data distribution-based shift detection, which directly checks how different the training data is to the new data. The other approach, classifier performance-based shift detection, compares performance metrics, such as accuracy, between the training data and the new data.
However, like mentioned at the beginning, this requires knowledge about the new data. If there is no (continuous) access to new data, it's best to assume there is a type of difference in data, based on the understanding of the problem. This again highlights how we have to assume the uncertainty of the future[6].

3 MILIGATING CONCEPT DRIFTS

Miligating concept drifts means the designer of the ML model has to resolve the four following design complications[8].
The first problem to be resolved, is having to make an assumption about the information source S_{t+1} the ML model samples from of the target data instance X_{t+1}. You can assume $S_{t+1} = S_t$, meaning the data source won't change drastically in the future. Another option is to approximate the information source based on X_{t+1}, by assessing how similiar X_{t+1} is to historical instances and estimate the information source based on this information. Another approach that is possible is to predict changes, this however, requires the creation of prediction rules to be able to guess what happens in the future and including the information in the learning process.
The next problem to be resolved is to recognize possible structural concept drifts, based on for example the patterns presented in 2.1. Once we identified the pattern we can move to the next problem, making the model adaptive to the change, meaning its *robust*. However, this is a challenging problem since training data collected in one context may not accurately represent the relationship in a different context. One way to address this can be assuming that features can be split into types: causal, meaning the feature is stable across environments, and spurious, meaning the feature is predictive in some but not all environments. The goal then is to create a model that relies on causal features, providing consistent prediction across various contexts and therefore the model is robust to all environments it could be used in[6]. Generally, the adjustment the internal workings of base learning components could improve the model.
Another option would be change the settings of the learning model based on the data, like adjusting weights for training samples. We can also focus on the way the model is trained, such as training set selection, manipulation or feature set manipulation, an example for this would be to carefully select unlabeled data from the new context and annotate it for example by human experts, giving more importance to the new data during training. While keeping the option to use older data, it may be given less weight in the process[6]. Optionally, you can also combine all the way to make the model adaptive to change[8].
Lastly the right model has to be selected, based on the estimation of the expected generalization error for a target instance at every time, meaning how well the model will perform on new, unseen

data. This choice as well depends on what we expect about the future data we'll get. Based on this assumption we then could choose, evaluate and adapt the model to the data. To estimate errors we could use theory to estimate how well the model will do based on its properties and the data or alternatively, we can practically estimate the performance by trying the model on different parts of the data. Then, we also have to choose whether the model adapt when a trigger signals a change or if it adapts over time without any triggers. Having a trigger is good when we can expect changes, while regularly evolving should be implemented based on how well the model is performing [8]. This choice also depends on what type of pattern we're expecting. With a incremental drift it would be better to use a regular evolving model, while with a sudden drift an adaptation on trigger would be more efficient[8].

3.1 Invariant risk minimization

One way to make a model robust to concept drifts is to use the invariant risk minimization (IRM) formulation for machine learning[6]. IRM can be used when dealing with data from different contexts and you want your model to prioritize important features that stay consistent across different scenarios while disregarding less important ones that might vary.

$$\begin{aligned} \hat{y}(\cdot) &= \arg\min_{f \in F} \sum_{e \in E} \frac{1}{n_e} \sum_{j=1}^{n_e} L\left(y_j^{(e)}, f\left(x_j^{(e)}\right)\right) \\ \text{such that } f &\in \arg\min_{g \in F} \frac{1}{n_e} \sum_{j=1}^{n_e} L\left(y_j^{(e)}, f\left(x_j^{(e)}\right)\right) \text{ for all } e \in E \end{aligned} \tag{5}$$

The goal is to minimize the model's prediction error, represented by F over all training samples from different contexts. f being the trained model and e represents different contexts in the data. n_e corresponds to the number of training samples from each environment. The inner summation calculates the error between predicted $\hat{y}(\cdot)$ and actually for each sample, where $x_j(e)$ and $y_j(e)$ are the features and labels of the j-th sample from environment e. The constraint in the second line reduces uncertainty and improve the model's generalizing out-of-distribution scenarios.

4 BENCHMARKING SALIENCY METHODS

Times series predictions are often used in high stakes applications like in healthcare or in the financial sector. Understanding what leads to a certain prediction of these machine learning models is a crucial step in building trust in these system for the users. Saliency methods are a popular approach for interpreting the importance of input features in deep learning models, but their effectiveness in the context of time series data is relatively unexplored [3]. Ismail et al developed a Benchmark to compare the performance of different saliency methods across diverse neural architectures. In the following sections, we will delve into the details of this benchmark, including the dataset design, evaluation metrics and the insights of the comparison.

4.1 Benchmark Design

4.1.1 Dataset Design. The dataset design in the benchmark study by Ismail et al involved the creation of multiple synthetic datasets to control and examine various design aspects commonly found in time series data. Each synthetic dataset is a binary classification problem and is characterized by various design aspects like signal position over the time axes, signal position over the features axes or signal value. In middle box dataset, for example, the discriminating signal is static over the time and feature axes and only changes with regard to the signal value. Ismail et al also introduced variations in classification difficulty by reducing the number of informative features, referred to as

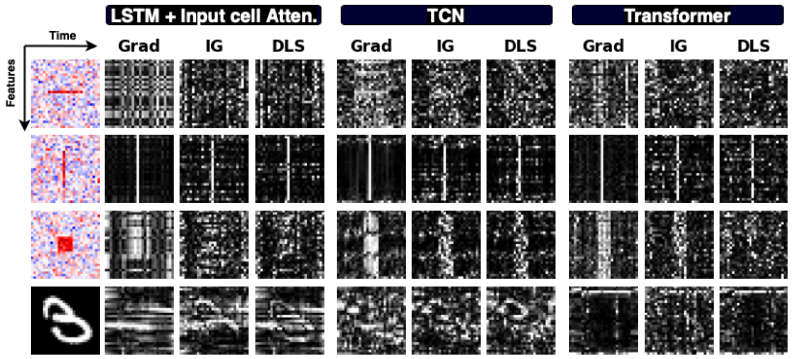

Fig. 2. The generated saliency maps Cite: Ismail Aya A., et al. (2020)

small box datasets. In addition to the synthetic datasets, the MNIST dataset was also included in the benchmark. It consists of black and white images of digits, so one image axis is treated as the time axis and the other as the feature axis.

4.1.2 Evaluation Metrics. For the evaluation of the feature importance Ismail et al use a modification-based approach. Initially, one of the saliency methods is applied, which attributes importance values for each feature across the different time steps within the dataset. After that, the obtained saliency values are sorted in descending order, creating a ranked list of features that reflects their perceived importance according to the chosen saliency method. The evaluation process continues by iteratively eliminating more and more features. In each iteration, the top k relevant features are selected based on their cumulative saliency, such that these make up a predetermined percentage of the total saliency. The selected features are then replaced with samples from the original distribution, which is known, as all datasets were generated synthetically. By replacing the eliminated features with samples from the original dataset, the requirement of the independent and identical distribution of the training and testing data is maintained. The model's accuracy is then calculated after each masking step, and repeated for different percentages of total saliency masked. In order to further assess the performance of the different saliency methods, the benchmark also compares the precision (whether all features identified as salient are genuinely informative) and the recall (whether the saliency method successfully identifies all informative features) [3]. It is crucial not to solely focus on the loss of accuracy after masking features because accuracy alone often does not provide a comprehensive understanding of the performance of saliency methods in identifying feature importance. For example, having the same level of accuracy after masking a feature identified as important by the saliency method does not necessarily mean that the saliency method has failed. Some neural architectures may use more features to make their predictions and therefore the performance does not necessarily deteriorate due to the loss of a single feature.

4.2 Benchmark Results

In the paper by Ismail et al, for space constraints they only include the evaluation of models that have an accuracy above 95% in the classification task. Despite this high accuracy of the models, many saliency methods struggled to identify informative features in the synthetic datasets for the respective neural architecture. This was particularly evident in cases like the Middle Box and Rare Feature datasets, where saliency methods faced challenges in highlighting relevant features [3]. Looking at Figure 2 we can see that no pair of saliency method a neural architecture delivers a saliency map which accurately shows the important features at the respective time steps for any

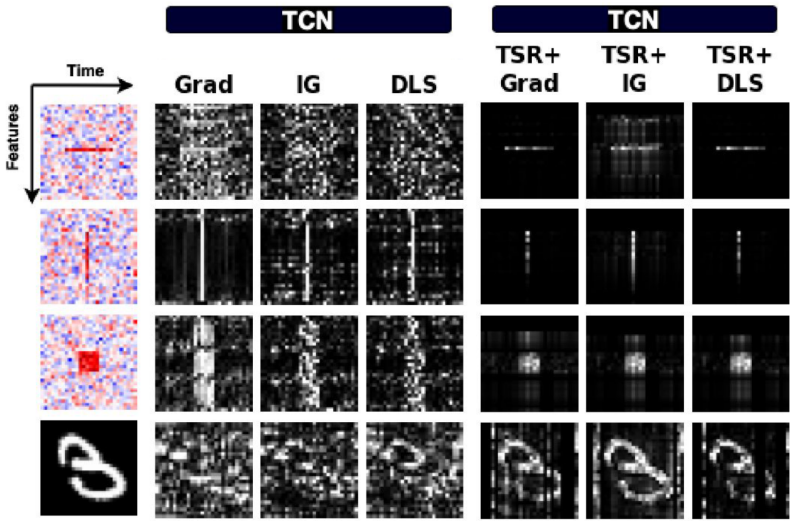

Fig. 3. Non TSR vs TSR Cite: Ismail Aya A., et al. (2020)

dataset. For the rare time dataset (second one from top) we can see that especially the saliency maps produced by grad focus on the right time step. Although this time step is also clearly emphasised by IG and DLS, they also highlight other time steps. However, differentiating the important features at the relevant time is a problem for all three methods. This problem can also be seen in the middlebox dataset, where GRAD roughly recognises the informative time steps for TCN and Transformer, but then basically highlights almost all features as relevant within those time steps.

4.3 Temporal Saliency Rescaling (TSR)

Temporal Saliency Rescaling (TSR) is a novel two-step methodology developed by [3] to augment the performance of saliency methods in the context of time series applications. It addresses the challenge of accurately identifying both time step importance and feature importance. The rationale behind TSR is to recognise relevant time steps and then to identify the important features within each step separated from the time component.

The first step of TSR involves the computation of time-relevance scores for each time step. This is achieved by quantifying the total change in saliency values when a specific time step is systematically masked. The aim is to identify the temporal significance of each time point, effectively capturing the changes in feature importance over the course of the time series. This first step leverages the strengths of existing interpretation methods, as Ismail et al observed that many of these existing methods demonstrate a high reliability in generating time-relevance scores. Following the calculation of time-relevance scores, the second step of TSR focuses on computing the feature-relevance score for each individual feature in time steps which are surpassing a predefined threshold α. This feature-relevance score is calculated using a similar approach to the first step, by masking each feature and measuring the resulting change in saliency values. The final (time, feature) importance score is derived from the product of the associated time and feature relevance scores, effectively capturing the nuanced interplay between temporal and feature-specific importance.

If you now compare the old saliency maps with the new ones generated with TSR, you can see a clear improvement. The Figure 3 on the left shows a segment of the Figure 2 with the three different saliency methods for the TCN neural net architecture. On the right are the saliency maps generated

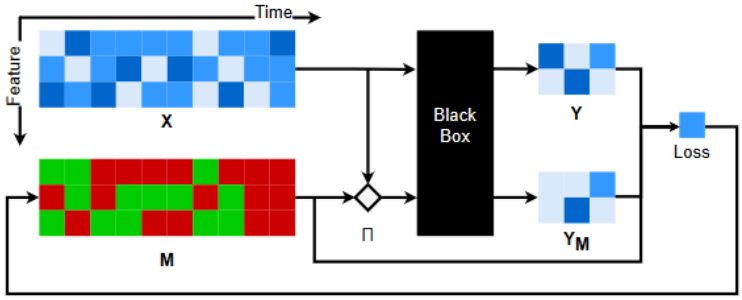

Fig. 4. Figure to describe how Dynamask works [1]

with the help of TSR. A direct comparison shows that these saliency maps are much more precise and mostly only highlight the relevant features.

4.4 Dynamask: Advanced Interpretation of Time Series Data

Crabbe and van der Schaar explain in their paper "Explaining Time Series Predictions with Dynamic Masks" [1] several challenge that appear when working with time series. One challenge would be to incorporate context, which is based on the temporal dependency. This means that an amount of sequential time steps are required to happen before another event takes place. Another challenge would be the large number of inputs, since the number of features gets multiplied with the number of time steps used by the model for making predictions. Being faced with the amount of information you'd get through a saliency map would be therefore overwhelming. Therefore saliency methods should be simple and clear. Crabbe and van der Schaar introduce "Parsimony" and "Legibility" to achieve this. Parsimony means that the method shouldn't pick more inputs than necessary to explain a prediction. Legibility means that when users look at the important scores, the analysis should be straightforward as possible. Analysing complex feature maps for long time series can become difficult and confusing and therefore would lessen the interpretability.
To deal with these challenges Crabbe and van der Schaar introduce Dynamask. It's a method for interpreting predictions in multivariate time series data using detailed importance scores for each feature at every time step. It operates by transforming a multivariate time series into an input matrix X, which is then utilized by a model to predict an output Y. A perturbation mask, termed as 'dynamask', and represented as M, mirrors X and contains saliency scores. The perturbed input X' is calculated using the formula $X' = X \odot M$, where $\odot$ denotes element-wise multiplication. The prediction of the model with this perturbed input is given by $Y' = f(X')$, where f represents the model function. The primary objective is to refine M to enhance the model's interpretability while minimally impacting its predictive accuracy. This iterative process of adjustment reveals the influence of changes in X' on Y', allowing for a continuous refinement of the saliency scores in M.

4.5 Structural types of Concept Drifts

To properly tackle the challenges mentioned at the beginning of this selection Dynamask ensures following things:
To incorporate context the mask is dynamic, meaning that the impact of features in adjacent times also is calculated and the values included in the perturbation for each future. A well-designed mask will reveal key features and time steps contributing to predictions, highlighting the dynamics within the time series data. To focus on only key features the mask only visualize features that have a saliency score of either 1 or 0. This ensures Dynmask to be parsimonious and legible. This also

helps with interpretability since it creates a proper understanding about what is really important.

Dynamask's approach is unique in that it preserves information about the temporal relationships between features, offering a more comprehensive understanding of time-dependent data. However, its computational intensity may limit its application in large-scale or real-time scenarios. Despite these constraints, Dynamask provides a balanced approach between interpretive depth and computational practicality, marking a significant advancement in the analysis of time series data.

5 DISCUSSIONS

In the following sections we will discuss two topics: Comparing Temperal Saliency Rescaling the "simpler" approach, and Dynamask the more "complex" approach. With the information we gained from the comparison we then try to think of a way to use this knowledge to potentially help with concept drifts in data streams.

5.1 Discussion: Temporal Saliency Rescaling vs Dynamask

After looking at two methods trying to make time series interpretable, we can compare them and try to conclude which one works better. Focusing on TSR first, it's a two-step method. In the first step, the goal is to understand the temporal significance. This is achieved through calculating time-relevance scores for each time step. Then, it calculates feature-relevance scores based on a predefined threshold.

Working with this method brings some challenges, such as deciding on the right threshold. If the threshold is too high then we might overlook some important feature and if it's too low we might add some noise. Moreover, while this may be a solvable issue there is another important issue to focus on. Temporal relevance is calculated by masking one time step and seeing how it influences the model's outcome. However, this is a greedy approach and undermines the complexity of time series, especially with multivariate time series. Only focusing on one time step at the time, loses you the relationship between time steps and the dynamic interplay between features. This is what the creators of Dynamask called context. TSR also fails at one more aspects the creators of Dynamask pointed out when creating a good saliency method for time series data. The result isn't "legible", since we end up with a saliency method where the analysis isn't quite straightforward since you end up with a variety of slightly highlighted points. As a user trying to understand the result, this would take a longer time to fully understand. We can also criticize the study itself. In the paper, time series data is presented as a picture with two dimensions: Time on the X - Axis and features on the Y-Axis. However, this isn't a accurate presentation of a time series. You lose temporal context, the relationships between data points over time aren't as clear. Time series are multi-dimensional and simply two dimensions to not accurately represent them. Therefore the results based on these pictures may not be actually meaningful.

Dynamask, on the other hand, creates a mask with a perturbed version of the input data. Based on how the perturbed version changes the outcome, the mask adapts saliency scores accordingly. In comparison to TSR this may be more complex and computational demanding, especially with more complex and bigger time series. However, since the method focuses on instances and not just on individual time steps, there is no immediate loss of information about temporal relationships. Therefore Dynamask appropriately incorporates context and leads to a more accurate result than TSR, with the potential trade-off in complexity. We also end up with a more clear saliency map, since either very important features or not important features are highlighted. This is easier to interpret for users.

To sum up, Temporal Saliency Rescaling might be first seen as the simpler and faster approach, however after taking a better look at how the study is conducted it is clear that one may can say

that one important part - actually keeping in mind the temporal context of features isn't properly incorporated. This is seen not only in how Temporal Saliency Rescaling works but also on what type of datasets it was tested. Dynamask on the other hand, by focusing on instances and also looking at the importance of features in adjacent times successfully incorporates context. It also works better as an interpretability method, since the mask has a clear communication about important and non-important features, which can't be said about the saliency maps produced by Temporal Saliency Rescaling. Therefore Dynamask should be chosen over Temporal Saliency Rescaling.

5.2 Mitigating of Concept Drifts through Time Series Prediction

Looking at the theoretical framework introduced in 3.1 on how to deal with concept drifts, we can take a look at how we could try to apply multiple time series predictions to help with concept drifts. To achieve this there has to be a trade off by introducing a delay and collecting the data as a (multivariate) time series. Then we can use the Dynamask method to understand what features in that time period are important and based on that predict or assume information about the near future. For example, for the first subproblem, where we have to make an assumption about the information source, the time series together with Dynamask can help us predict how it'll change in the future. The new available information can also help us identify the structural pattern of the concept drift. However, we still have to respect that with the infinite numbers of patterns and how complex they potentially could get, Dynamask eventually may also struggle to identify important features. Moreover, besides the trade-off with the delay, we also have to keep in mind the computational costs. This method to deal with concept drifts therefore wouldn't be able to be applied to time-critical situations, as well as fields where you can't afford the computational costs. However, this reflects the complex nature of concept drifts. With infinite patterns and the complexity data streams introduce, there can't be one solution to all. Instead there is a need for diverse strategies, fitting to different scenarios.

6 CONCLUSION

With data streams and time series predictions dealing with different types of data - one dynamic the other static, they do have things in common. Their data points are sequential and have a temporal dimension that can't be ignored when using machine learning models to interpret them, leading to different challenges. Looking at data streams the issue of concept drifts becomes prominent. With time the relationship between features and labels change and if the model can't adapt, it'll become inaccurate and unreliable. We looked at how to properly explain what a concept is, focusing on the probabilistic definition of concept (Concept = $P_t(X)$ and looked at multiple ways to formulize concept drifts. The complexity of concept drifts becomes even more clear when we realize that there is an infinite amount of structural patterns concept drifts can take on[8], besides the sudden, gradual, incremental drift and reoccurring context patterns we focused on. Moving on, we also looked at how to detect concept drifts. If there isn't any data from the new context already available it's difficult to use any shift detection method and therefore we have to assume one will exist and try to adapt the model accordingly [6]. This underlines the uncertainty when dealing with concept drifts. Trying to mitigate concept drift turns out to be a complex matter, with many factors to be kept in mind. From assumptions about the information source to choosing the right model, there are many design options to potentially mitigate concept drifts[8]. One method to make the model more robust is the Invariant Risk Minimization, making the model focus on important features that stay consistent across different scenarios[6].
Interpretability of ML models is needed for building trust and ensuring responsible decision making. However, usual saliency methods hit their limits when having to deal with time series data. Based on the paper of Ismail A. et al they struggle to tell which features are important or not at a specifc

time. If one feature is marked as important, then all other features at the time might also be marked as important, no matter their actual values. However, some methods do achieve to atleast pinpoint the important time step. The way the model is built, the model architecture, also has a big impact on how good the saliency map turns out to be. The paper of Crabbe and van der Schaar explain why time series data are a challenges have to be kept in mind when deailing with time series data. Four important aspects have to be ensured: Incorporating context, the number of inputs, not picking more inputs than necessary to explain a prediction and straightforward results.
We focused on two methods - TSR and Dynamask, both meant to highlight important features over time. However, we realize that the TSR method is too simple and greedy, losing temporal relationships, the context, by only focusing on one time step at a time and its impact. Dynamask on the other hand, while being more complex, properly computes this time-dependent context by focusing on time instances and letting the model learn which features are important.
Following this, we propose to use time series predictions to help with the issue of concept drifts in data streams by introducing a delay. However, this approach would only fit in non time critical fields and situations that can afford the computational costs. This highlights the diverse need for different strategies to deal with concept drifts, specialized for different situations.

REFERENCES

[1] Jonathan Crabbé and Mihaela van der Schaar. 2021. Explaining Time Series Predictions with Dynamic Masks. arXiv:2106.05303 [cs.LG]

[2] João Gama, Indrundefined Žliobaitundefined, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. 2014. A Survey on Concept Drift Adaptation. *ACM Comput. Surv.* 46, 4, Article 44 (mar 2014), 37 pages. https://doi.org/10.1145/2523813

[3] Aya Abdelsalam Ismail, Mohamed Gunady, Héctor Corrada Bravo, and Soheil Feizi. 2020. Benchmarking Deep Learning Interpretability in Time Series Predictions. arXiv:2010.13924 [cs.LG]

[4] Alessandro Margara and Tilmann Rabl. 2019. *Definition of Data Streams.* Springer International Publishing, Cham, 648–652. https://doi.org/10.1007/978-3-319-77525-8_188

[5] Teik Toe Teoh and Yu Jin Goh. 2023. *Time Series.* Springer Nature Singapore, Singapore, 65–85. https://doi.org/10.1007/978-981-99-4558-0_5

[6] Kush R. Varshney. 2022. *Trustworthy Machine Learning.* Independently Published, Chappaqua, NY, USA.

[7] Geoffrey I. Webb, Roy Hyde, Hong Cao, Hai Long Nguyen, and Francois Petitjean. 2016. Characterizing concept drift. *Data Mining and Knowledge Discovery* 30, 4 (April 2016), 964–994. https://doi.org/10.1007/s10618-015-0448-4

[8] Indre Zliobaite. 2010. Learning under Concept Drift: an Overview. *CoRR* abs/1010.4784 (2010). arXiv:1010.4784 http://arxiv.org/abs/1010.4784