FEATURES TO BLAME FOR CONCEPT DRIFT

Yahel Jacobs (ID. 313385536), Gilad Daube (ID. 313451601)

Tabular Data Science course, BIU, Semester A, 2022

Abstract

We all know that a well-trained model can make good predictions. This kind of model is working under the assumption that both train and test data are driven from the same distribution. But, in the case of a concept drift the same well-trained model won't be able to achieve the same accuracy of prediction as before, to achieve that we need to train it again. This type of change might happen more than once, causing us to retrain each time. Currently, there are no automatic ways to find patterns between each event. In this paper, we are suggesting a way to provide some new information about future feature distribution and patterns suggested from the drift. Using that knowledge would make it easier to build a more functional model that will be able to make good predictions even when drifting occurs. Our solution will analyse each feature distribution using *goodness of fit* and make future-distributions-predictions using a *machine learning* algorithm. While we manage to single out the features to blame, we did not succeed using this knowledge to predict future behavior.

1 PROBLEM DESCRIPTION

1.1 Concept Drift

Predictive models are models that learn the current data it holds and based on that data predict answer the model isn't holding. A satisfying predictive model will answer correctly on a percentage of the questions the model face. Those models are based on the conceptions that historical data is relevant and foretelling on future data and the relation between the input question and output answer did not change. As we know from the real world, these guidelines are true for some but not all the problems we are facing.

In many problems throughout life the relationships between input data and output data changes over time, in which case the predictive model won't be able to map correctly the new coming data at the same satisfaction as before. The changes may be consequential, such as that the predictions made by a model trained on older historical data are no longer correct or as correct as they could be if the model was trained on more recent historical data. Concept drifts are the changes in the relationships between the input data and the output. The changes take many forms, the changes can be temporary, permanently, or cyclical, the changes can occur instantly or gradually. Different concept drifts detection and handling methods are used for each situation.

Indre Zliobaite in the 2010 paper titled "Learning under Concept Drift: An Overview" [1] provides a framework for thinking about concept drift and the decisions required by the machine learning practitioner, as follows:

- **Future assumption-** a designer needs to assume about the future data source.
- **Change type-** a designer needs to identify possible change patterns.
- **Learner adaptivity-** based on the change type and the future assumption, a designer chooses the mechanisms which make the learner adaptive.
- **Model selection-** a designer needs a criterion to choose a particular parametrization of the selected learner at every time step (e.g., the weights for ensemble members, the window size for variable window method).

1.2 The Problem

In a well-trained model there are a satisfying percentage of good output prediction based on the input data. When the model is no longer producing satisfying results, we can assume that there was a drift in one or more features in the given data. Retraining the data is a poor solution at the moment. Firstly, retraining the model is a costly procedure, and in order to keep our model effective the retrain cost should be minimal and lowest as possible. Secondly, in order to retrain our model post-drift there is need to be enough post-drift-data, and therefor there is a wait period of time, while the data is being collected after the model is starting to provide poor results.

In an always changing streaming data- such as stocks price, traffic time, etc. the data is ever changing and non-cyclical. There are many drifts in the data in short periods of time making prediction models useless. By the time the machine learning the current distributions the data might change, and by the time the model will retrain again, which might take a long time so the model will collect enough data, the model won't work to our satisfaction. We'll get that in the traditional way of predictions a rapidly changed data cannot be used in prediction models.

2 SOLUTION OVERVIEW

In our solution we tried to understand the cause for drifting from the data itself. The purpose is to find which features are responsible to the drift (or as we call them *features to blame*), and hope that finding the source will help us prevent the next drift.

2.1 Features to Blame

We want to distinguish between problematic features ("*features to blame*") and all the others. Let's start with a definition for these features; we will declare a feature as *feature to blame* if it's distribution is difference between training time and testing time. Meaning, the predictive model was trained on irrelevant data in terms of this feature. We found two ways to decide if a feature is a *blamed one*:

1. **Estimate the distribution-** first we need to analyse its distribution during training and testing. Finding the actual distribution is a very hard task, especially when it's not one of the common distributions. What we will do, is estimate whether the data came from a particular distribution. As we learned, this task called the *goodness of fit*. There are different tests to check the goodness-of-fit between an observed distribution and a theoretical distribution, two of them are *Pearson's chi-squared test* and *Kolmogorov-Smirnov (KS) test*. *Pearson's chi-squared test* finds out the relationship between the observed data and the expected data, derived from a theoretical distribution. The score is defined by the sum of the squared differences between the observed and expected values (Fig. 1), using that score it is possible to calculate the P-value[1]. *Kolmogorov-Smirnov (KS) test* report the maximum difference

[1]A p-value is a measure of the probability that an observed difference could have occurred just by random

$$\chi^2 = \sum_{i=1}^{k} (O_i - E_i)^2 / E_i \qquad D_n = sup_x |F_n(x) - F(x)|$$

Figure 1: Chi Squared and KS tests

between the two cumulative distributions (the empirical CDF of the observed data and the CDF of the theoretical distribution), and calculates a P-value from that and the sample sizes (Fig. 1).

2. **Visualization-** instead of analyze the distribution, we could just plot and compare the data for training and testing all together. This technique called *Probability plots*; the training data are plotted against the testing data in such a way that the points should form approximately a straight line (Fig. 2). Departures from this straight line indicate departures from the specified distribution, which means the distribution did changed.

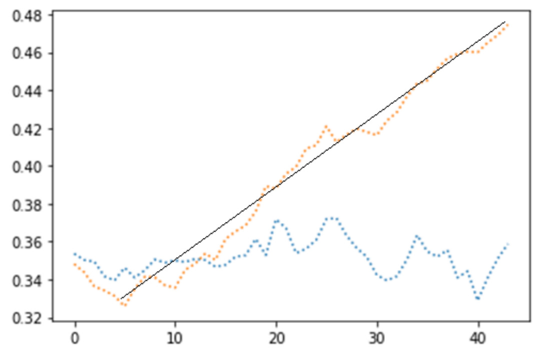


Figure 2: Plot training and testing together

2.2 Predict distribution

Our main goal is to try to prevent the predictive model's mistakes, which are based on data that was drifting. In order to do so, we need to find a way to predict which features will be blamed on the next drift and find their new distribution beforehand. In the solution, we used a *Convolutional Neural Network* (CNN) algorithm, which was trained on data before

chance. The lower the p-value, the greater the statistical significance of the observed difference. [2]

drift happened and labeled by the distribution after the drift. In our opinion, given enough data to train on, the model should be able to predict the future distribution for each feature. Using that prediction, one could customize the predictive model to take the new distribution under advisement in a way that will prevent drifting mistakes.

2.3 Our Solution

The solution consists of two parts: find and report the *features to blame* and predict for every feature its next distribution.

2.3.1 Features to blame

For each feature we will identify the best-suited distribution for both training and testing data, by comparing them to one another we could determine which features are to blame. After identifying them, we could report the results for each feature.

2.3.2 Predict next distribution

Using the ML algorithm described earlier, we will predict the distribution for every feature, assuming there is a change in the distribution.

2.3.3 The solution

As suggested before, our goal is to give a better understanding of the data distribution. Doing so by combining the two parts of our solution. We will create a report with all the features to blame we could find, giving the data analyst a perspective on how many features changed and which (over time it will create a pattern on some features). Also, it includes the predicted distribution for each feature, so the data analyst would be prepared when the next drift will come.

3 EXPERIMENTAL EVALUATION

To test both of our part we used large data bases that are bound to have drifts through them. Part one of the experiment was to our satisfaction, we found in the data the features that are drifting. Moreover, as noticed, these features were the features that had the impact on the overall drift in the data. To find the features, we divided the data by drifts and calculated the distribution prior and posterior the drift. The features with distribution change were flagged as the features that cause the overall drifts, *features to blame*. As can be seen in the report (Appendix Fig. 6) we constructed the features that contain change in distribution are being marked and labeled by their initial and recent distributions.

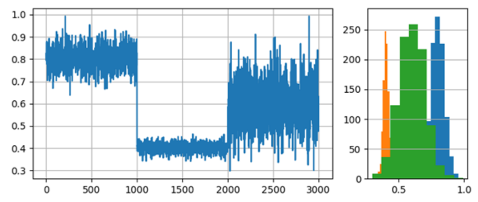


Figure 3: Visualization of a feature with two instant drifts

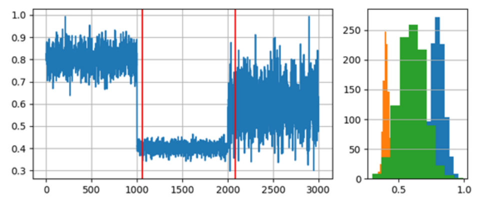


Figure 4: The feature after marking the estimated drift points

By flagging those features we came to the second part of our solution. These features were divided by their changes in distributions into masses of data and labeled with the distribution of the next mass (the first set with the distribution of second set, the second set with the distribution of the 3rd and so on). Tested with CNN learning algorithm we tested out theory; we labeled each mass of data by the distribution of the next mass as can be seen in the diagram. After the labeling of the we trained our net to predict the distribution after the drift from the current mass of data that is available. As we ran our net, we came to the understanding that our theory is wrong. We ran our net multiple times, tried different set of layers and sizes of nets but were never able to predict the next distribution for a set of data. As we can see (Fig. 5), trying to predict the distribution of the data after the drift from the data before the drift was true for **1 / number of distributions** which is simply random prediction. All and all changes we made to the prediction net never crossed the barrier of the random select of distribution causing us to the conclusion that our theory is wrong there is no pattern between sets of data and the distribution sub-sequential a drift.

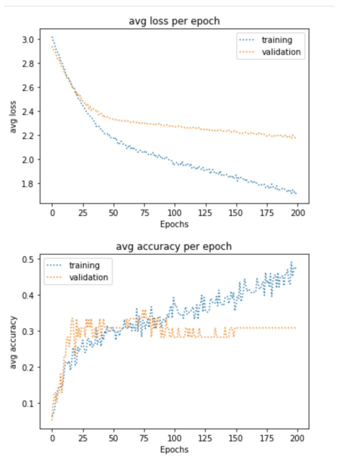


Figure 5: As can be seen the loss and accuracy are not minimizing

4 RELATED WORK

There is no right or wrong approach when dealing with Concept Drift. In some cases, the best way is to just detect the drift and re-train the model. In others, it's to learn the change and adapt it in real-time. In general, the work in this field can be divided into two main groups: Passive and Active, we briefly review the related work and solutions in each group.

- **Passive-** this type of approach is not about avoiding the drift but detecting and monitoring it. It can be done by monitoring the performance of the model over a long time, an easy solution and not hard to achieve. It is also possible to use an ML algorithm to detect it, like *ADWIN* (ADaptive WINdowing) or even using Microsoft Azure ML [3].

- **Active-** this type of approach is all about avoiding the drift, it can be done using one or more of the following solutions:

  1. **Online learning-** the model never stops to learn, every single sample is considered a *training* sample- this is the ideal way to avoid drifting. In our solution, using the prediction of the future distribution could make it possible to avoid drifting in advance, by also training with data from the predictive distribution.
  2. **Periodically re-train-** re-train the model on a constant period of time. A better way to do so is to monitor model performance and trigger a re-train when needed,

like while model performance degrades below a specified threshold.

3. **Feature dropping-** the idea is to build multiple models where one feature is used at a time keeping the target variable unchanged. For each model, the prediction on test data is checked and if the result is not up to the task, this feature might be considered as drifting and hence might be dropped. In our solution we deliver the features that should be dropped- they are the *features to blame*.

Our solution belongs to the "active" group. Finding the *features to blame* would make it very easy to decide which features to drop, moreover using ML algorithm to predict future distribution helps us avoid future drifting by adjusting the model accordingly (training on predictive distribution in advance). But, unlike these approaches by understanding our report-result it is possible to find patterns and understand why the drift happened.

5 CONCLUSION

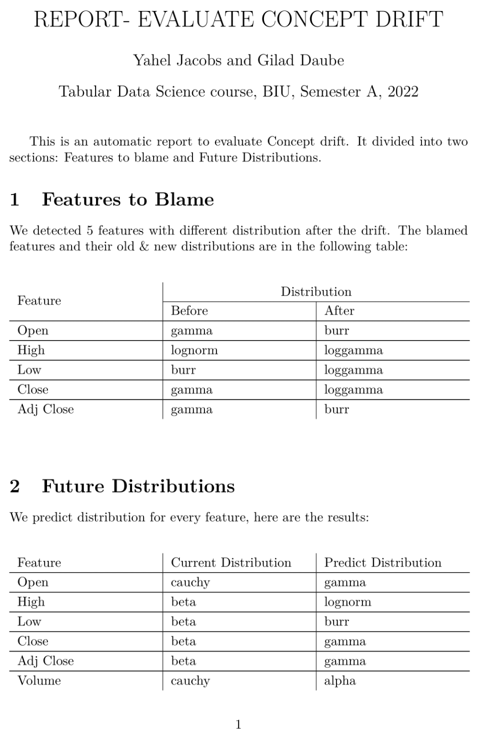
In conclusion, in case of a drift in the data- a trained prediction model will generate unsatisfying results for the data is no longer match the training of the model. Our initial thought was that by flagging the features that cause the drift, and by finding patterns between sets of data and the distribution after the drift; we will be able to tune the model without waiting for the data after the drift and retrain the model. After flagging the *features to blame* the prediction did not work as we hoped. We could not find a pattern in the behavior of the drifts and were not able to predict distribution. It seems as, in the data bases we studied, there are random distributions for sets of data. Even though we did not get the results we were hoping for and perhaps our theory was too vain, we did learn that in big sets of data that is frequently drifted there are changes in the distribution in the data features and to our knowledge the changes are not in a detectable pattern.

BIOGRAPHIES

[1]Learning under Concept Drift

[2]P-Value explanation

[3]Microsoft Azure ML



REPORT- EVALUATE CONCEPT DRIFT

Yahel Jacobs and Gilad Daube

Tabular Data Science course, BIU, Semester A, 2022

This is an automatic report to evaluate Concept drift. It divided into two sections: Features to blame and Future Distributions.

1 Features to Blame

We detected 5 features with different distribution after the drift. The blamed features and their old & new distributions are in the following table:

Feature	Distribution	
	Before	After
Open	gamma	burr
High	lognorm	loggamma
Low	burr	loggamma
Close	gamma	loggamma
Adj Close	gamma	burr

2 Future Distributions

We predict distribution for every feature, here are the results:

Feature	Current Distribution	Predict Distribution
Open	cauchy	gamma
High	beta	lognorm
Low	beta	burr
Close	beta	gamma
Adj Close	beta	gamma
Volume	cauchy	alpha

1

Figure 6: *features to blame* report