International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 05 | May 2020 www.irjet.net p-ISSN: 2395-0072

A Survey on Machine Learning Model Monitoring for Performance Assessment

Yogita M Pattan, Prof. Smitha G R

Department of Information Science and Engineering, RV College of Engineering
R V Vidyaniketan Post, Mysore Road, Bengaluru - 560 059, Karnataka, India

---

Abstract - *In the recent decade, Machine Learning models are being used for a wide range of applications across the world. Once a Machine Learning model is developed, it is not the end of its lifecycle. A very important phase of its lifecycle is monitoring the model's performance based on the real-time environment where is it under operation. It is essential to monitor the model as long as it is being used in order to exploit the model to its maximum and make sure that is performing well with respect to its intended purpose. Model monitoring is as important as any other phase because the model may degrade over time with unstable environment, deviate from its intended purpose leading to invalid and inaccurate outputs which can cause adverse effects on the outcomes of decisions which rely on these models. A wide range of factors cause model performance degradation and a number of challenges are faced when a model is being monitored. These factors and challenges must be understood to tackle them effectively. This paper provides information on the factors which influence the behaviour of the models, lists out the challenges that can be faced when monitoring, provides some of the model monitoring metrics that allows quantifying the model performance. Lastly, this survey paper presents a few model monitoring methods that have been developed and provides information on the functionality of these monitoring tools.*

Key Words: **Machine Learning, Model Monitoring, Model drift, concept drift, data distribution, Monitoring metrics**

1. INTRODUCTION

A Machine Learning model can be described as a mathematical representation of a real world process. This model takes certain input, processes the input and produces some estimated output. Every model undergoes a lifecycle beginning with the Development phase where is model is developed, followed by the Testing phase where rigorous testing is carried out on the model, the Deployment phase where the model is exposed to real the world, and finally the Monitoring phase where this intelligent system must be tracked.

Model monitoring is a stage in the lifecycle of a model where the model is monitored to determine the continued stability of the model, to assess the performance of the model in the real world environment and maintain a predetermined desired level of performance. Long term changes in the model include changes of distribution in the features with respect to the training time, referred to as concept drift, which would require retraining the model. Short term changes like Errors in the features and crashes must also be taken into consideration while model monitoring.

Models deal with huge amounts of real-time data. With an unstable environment, the models may degrade overtime in terms of their prediction power as they are being used, which is also known as Model Drift. Model drifts are caused due to:

1. Unseen data: Usually, during the development of machine learning models, they are trained mostly on a small percentage of total data. This is usually due to the insufficiency of quality data available for training models accurately. Even though the model may be developed in order to be able to generalize the data, there can be outlier data points that cause the model to predict erroneous outputs.
2. Fluctuation in the environment and associations between the variables: A model is developed by tuning the model parameters and variables at the time of creation. With the evolving environment, the associations between these variables may cause the model to not perform at all.
3. Upstream data changes: This refers to the changes in the operational data. The developers of the model have no control over the system which provides the data. Any significant change to certain parts of the data will lead to consequences on the model performance as the aggregated data will be deteriorated.

Two types of monitoring methods are known – Proactive Model Monitoring and Reactive Model Monitoring. Proactive Model Monitoring requires defining a set of key indicators such as data patterns against which the model's performance is tracked. Whereas, Reactive Model Monitoring performs a Root-Cause-Analysis (RCA) on any erroneous output provided by the model and determines how it can be rectified.

This paper is structured as follows. Section II discusses the challenges faced during model monitoring. Section III lists some of the performance metrics to quantify model monitoring. Section IV discusses a model monitoring procedure from [1], Section V discusses a model monitoring java toolkit and it's functionality from [2].

2. MODEL MONITORING CHALLENGES

Multi-Language Code Base

Most of the libraries for Machine learning are written in Python, for example: numpy, sklearn. Whereas most of the data pre-processing use Java based systems like Apache Spark. Due to such heterogeneous code base causes a challenge as the automatic error detecting tools can either check java or python. Having a monitoring system for all kinds of code base is one of the major challenges.

Model Retraining Decision

When a model underperforms, there can be a huge number of factors which cause the degradation in performance. In certain cases, the models may be required to be retrained on an evolved dataset or even may have to be remodelled. Deciding on when this has to be carried out on the model is a challenge as it requires careful evaluation of the models and observing their deviation.

Model Boundary Settings

A model may be trained for a given range of input data. Any data point beyond the range can be considered as an outlier. This range cannot be fixed, taking into consideration the change in input data over time. Hence, the change in input data pattern must be monitored and consequently, these changes have to be incorporated into the model to adapt to the changing data.

3. PERFORMANCE METRICS FOR MODEL MONITORING

There are a number of metrics that one can use to evaluate the performance of a model. Appropriate metric must be chosen based on the following factors like the type of model, the business objective.

Some of the common metrics used include False Positive, False Negative, Precision, Recall, F1 Score which are particularly used for classification models, whereas Accuracy, Mean Absolute Error, and Mean Squared Error are used for Regression models. Some of the key figures that are measured for a Model Monitoring Framework are discussed here.

3.1 Population Stability Index (PSI)

PSI is a measure which determines the change in the characteristics of dataset overtime. A model is trained on a static data set but is being used in a dynamic environment. It is important to measure the change in data to determine the model produces accurate outcome.

PSI is calculated as shown where $Expected_i$ is the % of responders in the i^{th} decile of training data and $Actual_i$ is the % of observations in the i^{th} decile of the real-time data.

$$PSI = \sum_{i=1}^{n} \left\{ (Actual_i - Expected_i) ln \left(\frac{Actual_i}{Expected_i} \right) \right\}$$

If the PSI value is less than 0.1, there is an insignificant change in the data. If the PSI value is between 0.1-1.25, there is a minor shift in the data characteristic. If the PSI value is greater than 0.25, there has been a major shift in the data characteristic.

3.2 Kolmogorov-Smirnov (KS) Test

The KS-statistic is a measure of degree of separation between distribution function of two samples. If a model is capable of dividing the data into two separate groups, then the KS-statistic value is 100. Higher the KS value, better the performance of model.

3.3 Receiver Operating Characteristic (ROC) and Lift Curve

ROC curve is a graphical representation which displays the capability of a model to classify binary events. The Area Under Curve (AUC) represents an estimate of the quality of the classifier. It is constructed by plotting the true positive against the false positive by varying the threshold value.

Lift curve provides a quick estimate of the efficacy of a classifier. They are particularly applicable in market models.

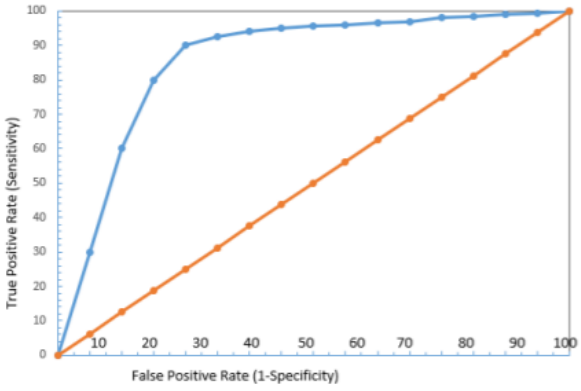

Fig -1: ROC Curve

Table -1: Comparison between ROC Curve and KS Test

ROC Curve	*KS Test*
Indicator valid for assessing the overall performance of model	Is being used as dissimilarity metric for assessing the model's power
Higher the AUC_ROC, better the classifier	Higher the AUC_KS, better the classifier

ROC curve is constructed by plotting True Positive Rate against False Positive Rate	KS statistic is the maximum difference between True Positive Rate and False Positive Rate
AUC_ROC = 0.5 +AUC_KS	

4. MODEL MONITORING FOR DATA STREAMS

Concept drift detection is a noted difficulty in a machine learning model. Fabio Pinto, Marco O. P. Sampaio, Pedro Bizarro in [1] discuss SAMM (Streaming system for Automatic Model Monitoring) as an automatic model monitoring system which detects concept drift. It is able to detect sudden fluctuations in addition to providing the explanation for the change. SAMM becomes very useful when the model monitoring process becomes tedious and difficult due to unforeseen performance.

SAMM operates by evaluating a threshold and a signal that triggers alarm when it crosses the threshold value to detect a concept drift. It also produces explanation reports for these deviations. SAMM does this by detecting local changes in the stream of outcomes produced by the model.

The next section describes the different components of SAMM and how they are integrated to function as a whole.

4.1 Signal Computation

A signal value S is evaluated for each incoming data. It includes an estimate of resemblance between a reference window R and a target window T. T contains the last n_t data points collected and R contains the data corresponding to a reference period prior to the target window T. The signal S is defined as a measure M of similarity between the target window and the reference window, S = M(R, T). [1] present their results by adopting Jensen-Shannon Divergence (JSD) due to its appealing properties.

4.2 Threshold Computation

The proposed method to evaluate the threshold for a signal above which an alarm is triggered is briefed. To compute the threshold value in streaming scenarios, preceding perceived instances are kept in memory which in turn requires maintaining a cluster of samples and carrying out operations on these samples like sorting. This method produces highly accurate estimates. Hence, the method adopted is a lighter approach that incorporates interpolation based on set of bins updated in a linear pass and a simple percentile update approach.

An algorithm called SPEAR (Streaming Percentile EstimAtoR) is designed taking into consideration the Time and Space efficiency and Streaming implementation. This algorithm has been tested with real datasets to prove its usage suitable with SAMM.

4.2 Alarm Report

An alarm is triggered when the signal crosses the threshold. A report is created by comparing the events in windows T and R through measure of heterogeneity. A Gradient Boosted Decision Trees (GBDT) model is used to obtain an alarm score and a measure of feature importance to obtain a rank for events that caused the deviation in the model scores.

5. MODEL MONITOR (M^2)

Troy Raeder, Nitesh V. Chawla in [2] discuss Model Monitor as a java toolkit for robustly assessing machine learning models in a varying environment. It helps the researchers to combat probable shifts in data distribution.

The fundamental capabilities of this toolkit include:

1. Identification of Distribution shift: It allows highlighting the features of the testing data which causes drift by conducting sensitivity study on the features and determining the features which are more likely to vary between the testing and training data. These features can be isolated and processed to reduce its impact on the model.
2. Exploration of hypothetical scenarios: This allows the user to introduce changes in the data beforehand based on how the data may fluctuate overtime in a hypothetical scenario. Later on, allows them to compare the results before and after these changes were injected.
3. Comparison of Classifiers: A model may be tested against multiple data sets and distribution shift. For each distribution shift, the model is tested with all the data sets and calculated an average rank for the model hence portraying the relative performance of the model.

Performance Measures

M^2 supports most of the standard performance measures like Accuracy, Precision, Recall, F1 score. Apart from these, there are performance measures implemented that can be particularly used for classification models- Brier Score and Negative Cross Entropy.

Methods for identifying and computing the shifts in distribution are a part of the tool. It provides the Hellinger Distance and Kolmogorov-Smirnov (KS) test. The Friedman test and Bonferroni-Dunn test are used to obtain a comparative analysis on the behavior of multiple classifiers.

6. CONCLUSIONS

Organizations use Machine learning models for predicting estimated values which guide them towards taking major decisions. Hence, a model must perform at its desired level while maintaining its stability. This paper presents SAMM, an

automatic monitoring system of ML models for data streams which has been validated with real world data sets, M^2 which is the only publicly available software that leverages multiple performance measures and allows learning under varying distributions. This paper also presents some challenges that need to be considered, evaluate different performance metrics based on the scenario that aid in monitoring process and adopt complex methods proposed by different works. Some of these challenges and performance measures involve quantifying understanding the data that the model has to process. Thus, a potential research direction could be on data monitoring for a better model monitoring. Due to complexity of the mentioned algorithms, mathematics and other related formulas have not been presented and readers are advised to view the original papers for more details.

ACKNOWLEDGEMENT

A lot of analysis and reading was done prior to the curation of the content of the given paper. This would definitely not be possible under the guidance of our mentor, Prof. Smitha G R, who constantly guided and gave a direction in the due course of the given paper. We would also like to thank our Head of Department, Dr. B.M. Sagar, who gave us this opportunity to work on this paper.

REFERENCES

[1] Fabio Pinto, Marco O. P. Sampaio, Pedro Bizarro, “Automatic Model Monitoring For Data Streams,”KDD-ADF-2019,Anchorage, Alaska, August 2019.

[2] Troy Raeder, Nitesh V. Chawla, “Model Monitor(M^2):Evaluating, Comparing, and Monitoring Models,” Journal of Machine Learning Research 10 (2009) 1387-1390, July 2009.

[3] Sebastian Schelter, Felix Biesmann, Tim Januschowski, Davis Salinas, Stephan Seufert, Gyuri Szarvas, “On Challenges in Machine Learning Model Management” Amazon Research, 2018.

[4] Corinna Cortes, Xavier Gonzalvo, Vitaly Kuznetsov, Mehryar Mohri, and Scott Yang. 2017. Adanet: Adaptive structural learning of artificial neural networks. In Proceedings of the 34th International Conference on Machine Learning, vol. 70. JMLR, 874–883.

[5] Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Karro, and D Sculley. 2017. Google vizier: A service for black-box optimization. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 1487–1495.

[6] Denis Baylor, Eric Breck, Heng-Tze Cheng, Noah Fiedel, Chuan Yu Foo, Zakaria Haque, Salem Haykal, Mustafa Ispir, Vihan Jain, Levent Koc, et al. TFX: A TensorFlow-Based Production-Scale Machine Learning Platform. KDD, pages 1387–1395, 2017.

[7] Joos-Hendrik Bose, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Dustin Lange, David Salinas, Se-¨bastian Schelter, Matthias Seeger, and Yuyang Wang. Probabilistic Demand Forecasting at Scale. PVLDB, 10(12):1694–1705, 2017.

[8] H. Yan, X. Shen, X. Li, and M. Wu, “An improved ant algorithm for job scheduling in grid computing,” presented at Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, vol. 5, pp. 2957-2961, 2005.

[9] Nick Hynes, D Sculley, and Michael Terry. The data linter: Lightweight, automated sanity checking for ml data sets. NIPS Workshop on Machine Learning Systems, 2017.

[10] H. Witten, E. Frank, L. Trigg, M. Hall, G. Holmes, and S.J. Cunningham. Weka: Practical Machine Learning Tools and Techniques with Java Implementations. ICONIP/ANZIIS/ANNES, 99:192– 196, 1999.

[11] Ambika Kaul, Saket Maheshwary, and Vikram Pudi. 2017. AutolearnâĂTAutomated feature generation and selection. In 2017 IEEE International Conference on Data Mining (ICDM). IEEE, 217–226

[12] Lars Kotthoff, Chris Thornton, Holger H Hoos, Frank Hutter, and Kevin LeytonBrown. 2017. Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA. The Journal of Machine Learning Research 18, 1 (2017), 826–830.

[13] Fatemeh Nargesian, Horst Samulowitz, Udayan Khurana, Elias B Khalil, and Deepak S Turaga. 2017. Learning Feature Engineering for Classification.. In IJCAI. 2529–2535.