Software Requirement Specification

Version – 0.1

ML model monitoring system

Author(s)

Sriram Bhamidipati

Contents

[Abbreviations /definitions 2](#_Toc71842210)

[Background 2](#_Toc71842211)

[Requirements 3](#_Toc71842212)

[Generic 3](#_Toc71842213)

[Scalable 3](#_Toc71842214)

[Accurate 3](#_Toc71842215)

[Robust (Fault tolerant) 3](#_Toc71842216)

[Cost effective 3](#_Toc71842217)

[User friendly 3](#_Toc71842218)

[Future proof 3](#_Toc71842219)

[References 4](#_Toc71842220)

## Abbreviations /definitions

|  |  |
| --- | --- |
| Abbreviation | Comment/expansion |
| ML | Machine Learning |
| NN | Neural Networks |
| DL | Deep Learning |
| MLMS | Machine Learning Monitoring System  Can also cater to other classes of AI models such as Deep Learning/Neural Networks etc |
| HA | Highly Available |

# Background

Across the world, businesses are increasingly dependent on data and insights from Data to survive and thrive. The insights are of various types including exploratory and predictive/prescriptive, depending on the need and limitations such as time, data and other resources.

In their striving for business insights, many build/use Machine learning modes and some go beyond to use Neural networks and Deep learning models as well. These (ML/DL/NN) models play critical in business strategy and help companies thrive by providing valuable inputs

## Requirements basis for MLMS

This section discusses the rationale and basis for drawing requirements for the MLMS

All along, the guiding principle will be, reuse over reinvent

The requirements for MLMS are generally drawn from 2 types of sources

## Domain Specific Requirements

Inferences drawn by Domain experts . These are learnt hard by the industry and must be incorporated where possible to the maximum extent. While the requirements are from past, some may have temporal relevance and need to be considered. The purpose of this application is NOT to delve deeper into these requirements but make the application adaptive and extensible (should be able to stub out any requirements which are irrelevant for a given deployment and/or add new requirements with much ease , or even better, by adding assertions in a configurable format so no major code changes will be needed

## Generic monitoring system requirements

There is plethora of monitoring applications readily available and many of which are opensource. A quick search can yield a set of such tools. A careful evaluation is necessary either to borrow the source code as a baseline or draw the requirements . Making use of such has limitations related to GPL and monetization. Depending on such, we can at least draw the requirements rather than borrowing the codebase

# Detailed Requirements

From a birds-eye-view, we see the following requirements for a ML Monitoring system.

## Domain (ML) Specific

To avoid reinvention, these are drawn from the well researched documents (See References)

* Detect and report Dependency changes
* Data invariants hold in training and serving inputs, i.e. monitor Training/Serving Skew
* Training and serving features compute the same values
* Models are not too stale
* The model is numerically stable
* The model has not experienced dramatic or slow-leak regressions in training speed, serving latency, throughput, or RAM usage
* The model has not experienced a regression in prediction quality on served data
* Add more

## Generic

MLMS should be very generic and be able to support all classes of models (ML/NN/DL). The system should address the broadly identified challenges in well researched publications (see 4)

## Scalable

Depending on size of the company and other demands, there can be several Models running at a given time as part of bigger model and the MLMS needs to be auto (up/down) scalable for meeting such dynamic requirements without much of an effort. A lot of times paradigm of horizontal scaling is better than complex performant (deep monolithic) systems

## Accurate

There should be checks and balances not just for the model tuning but also for the MLMS. The checks and any deviations will be reported for manual/auto reviews/autocorrective measures

## Robust (Fault tolerant)

This requirement is aka HA system. Many times, there can be subcomponent failures over long periods of usage and due to latent bugs in the subsystems. While every effort should be made to ensure quality before deployment of the MLMS, there should be exhaustive exception handling which can catch such issues and recovers and makes the overall system highly available

***Redundancy*** (where necessary) is another paradigm which needs to be applied. For example, an ML pool may require a set of load balancers or name servers (DNS)

## Cost effective

Any system should justify its existence/use by its cost /benefit analysis. An MLMS can be an overkill at times but very economical for large scale ML deployments where ML performance is critical and timely corrections (for model performance) are very essential. MLMS should be designed/engineered to be very cost effective to the user which can make it a competitive and compelling case

Make use of Opensource products which are robust rather than costly licensed products

## User friendly

As humans, we prefer a very intuitive and simple usable systems and yet all the complexity is hidden under the hood. It should not trade off simplicity at any cost

1. Quick steps to configure and deploy any class of model
2. Extensive logging for any issues/red flags
3. A set of useful reports and alerting mechanisms
4. Little or no manual intervention (eg: auto scaling , horizontally scalable on demand)

## Future proof

The tools used by MLMS can be obsolete and the system should be loosely coupled and make no assumptions about any tool and be able to work with configurable set of tools. While it is indeed a challenge to be flexible enough to encompass any tool, the changes should be containable and easy to be incorporated when the need arises.

Conclusion

The above is a quick list of requirements which need to be drilled down further to make the MLMS a comprehensive and compelling application

All along the system should be developed with industry best practices such as

* Agile methodology
* State of the art Versioning systems
* RESTful Interfaces
* Logging/instrumentation
* Scalable and Highly available system design principles
* …

## References

#### https://devopscube.com/best-opensource-monitoring-tools/

#### <https://christophergs.com/machine%20learning/2020/03/14/how-to-monitor-machine-learning-models/>

#### <https://towardsdatascience.com/monitoring-machine-learning-models-62d5833c7ecc>