
Tech
Mahindra
Connected World. Connected Experiences.

Analytical Model Version Control

(Let's Leverage the investment)

A Tech Paper by Tech Mahindra Analytics CoE

Author: Venkatesan Ramakrishnan, Principal Consultant

Table of Contents

Objective

The focus of this article is to, discuss on addressing the increasing need for analytical model version control. There are many blogs/solutions on this topic, which recommends GIT DVC as a solution. This article will focus more on leveraging the investment made on existing analytics infrastructure than introducing a new component/ investments. This article does assume that there is analytics infrastructure already available and that needs to be efficiently leveraged.

Prelude

While creating an analytical model, Data Scientists work with various methods (depending on the objective of the problem), and determine the best fit method to resolve a problem in hand. This process though sounds simple, there are various activities carried out like:

1. Exploratory Data Analysis
2. Computation
3. Imputations
4. Null/Blank value treatments
5. Hypothesis validations
6. Hyper parameter fine tuning
7. Model Validations

Once the above steps are completed, the model is ready for deployment (means ready to run with new data of the same structure) in a periodic manner (production runs). The model over a period of time tend to show accuracy issues, due to data processing issues and new patterns in data. This is the point that calls for tuning and re-deployment which, again has to go through the above steps.

While it looks easier said than done, getting the right version of the model code and data used to build the model is always a challenge. Some of the challenges that is practically faced in every analytics projects are:

1. Resource churn - who built is known but person not available
2. Data availability - Data used to build the model is not available due to various constraints and no standards
3. No proper versioning of the model - Don't know if the code available is the final version
4. No proper model metadata availability - Hyper parameters, data structure etc., need to go through the code to understand it

While Analytical model code can be part of any standard code repository, the challenge is more towards, which data has been used to build the model. At this stage Analytical model version control into consideration.

Anatomy of an Analytical Model

An Analytical model consists of:

1. Data
2. Program code (SAS, R, Python, Java etc.)
3. Hyper Parameter details (embedded in coding)
4. Computations and Imputations (embedded as part of extraction layer or code)
5. Statistical libraries used (embedded as part of code)

Analytical Model Version Control – Approach

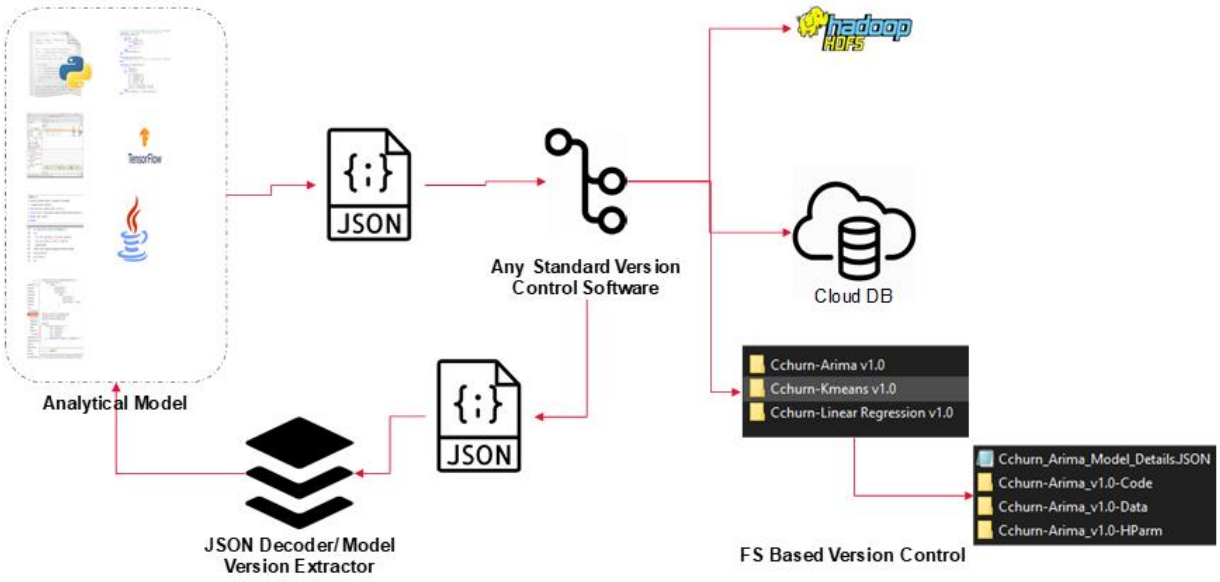
There are many articles out there detailing how to use GIT DVC for effective version controlling of analytical models. But here the focus is how to leverage the existing investment to the fullest for Analytical Model version control. The suggestion assumes there is a cloud strategy in place or Big Data setup available, (it is possible to do this setup without both, but that calls for lot more control on storage, files, directory, data convertors, data compressors etc.).

Unlike any project codebase version control, Analytical model version control predominantly focuses on storing the version of analytical model code that is put into production along with the data that was used to build the model. The challenge is more on the part of storing the version of data that is finally used to build the model. The below illustration shows how this can be achieved.

The solution proposed is a simple one, which leverages JSON and PowerShell (Windows) or shell script (Linux) to store and retrieve the data along with enterprise standard version control tool. To implement the solution there may be a additional lines of coding that need to be added to the existing analytical model code. The JSON file should be created with the following details:

1. Hyper parameter details

2. Data Location details are stored in cloud or Big Data Environment or directory details
3. Create a Model version extractor which can read the JSON file and extract the respective model code and the data, along with the hyper parameter details



The advantages of the above architecture are:

1. Extensible wireframe
2. Leverage the File systems/ Big Data/ Cloud's built-in security and compression features
3. Ability to handle variety of data
4. No need to procure any additional licensed software
5. Provide better control over Analytical model

The following points should be considered to take the right decision, before going for Analytical Model version control:

1. Is there a Big data and/ or Cloud policy available
2. Can Big Data and/ or Cloud be leveraged for version controlling the analytical model
3. Should we opt for COTS product or build a framework
4. Is there an in-house talent available for building the analytical model version control

The below table provides some guidelines on deciding the best way forward:

Feature	Cloud	On Prem Big Data	On Prem File System
Built-in Security Feature	Y	Y	Y
Dual layer of Security (O/S and Environment Security)	Y	Y	N
Additional cost involved while downloading the data and related model	Y	N	N
Custom Development of Analytics Model Management	Y	Y	Y

Conclusion

The above solution is only a recommendation. Whenever open source analytical languages like Python or R is adapted for delivering analytical models the above strategy can be very handy. While this is a framework, a deeper discussion concerning the practicality and maintainability of this framework needs to be assessed upfront. Tech Mahindra has substantial experience on the above framework and used this framework in their CoE for better demonstration of analytics solutions and reusability of the models across various domains.

Disclaimer

Thank You

Visit us at techmahindra.com