databricks

Model Risk Management

value_at_risk

Executive summary

This section reports the description of the model as specified by the model owner and reported on mlflow registry. The narrative should clearly articulate the business motivations behind this initiative, the desired business outcomes as well as its associated risk for the bank and / or its customers. This section will be used to assess the materiality of this model and may trigger different review processes and compliance requirements accordingly.

Model name	value_at_risk
Model creation date	2022-06-04
Model owner	antoine.amend@databricks.com
model	parent
parent	model

This is LAYER 1. This brings model description

databricks

Model submission request

This section reports the description of the model version submitted by the model owner and reported on mlflow registry. The narrative should clearly articulate the business motivations behind this new submission, and the desired benefits relative to previous model versions. Please ensure markdown is attached to your model submission on mlflow registry.

Model submission date	2023-02-17
Model version owner	antoine.amend@databricks.com
Model version	11
Model stage	Staging
model	version
version	model

This is LAYER 2 - brings submission context

Developmental history and conceptual soundness

This section reports the technical approach taken during the implementation of this particular model version. The narrative should clearly articulate the consideration behind the use of specific libraries and frameworks, the relevance of the data used throughout this exercise as well as the assumptions, constraints, dependencies, limitations and risks (ACDLR) as identified at the start of this project. Using empirical evidence, this section should clearly indicate why this particular experiment was proved to be the best model candidate and why other experiments or approaches were discarded. Finally, when applicable, the practitioner should be able to explain their strategies to ensure an explainable, fair and ethical use of data / AI. Please ensure markdown is attached to your model experiment on mlflow.

Execution time	2023-02-17T22:26:27.110000
Execution user	antoine.amend@databricks.com
Execution workspace	e2-demo-west.cloud.databricks.com
Execution type	NOTEBOOK
Execution code	/Repos/antoine.amend@databricks.com/value_at_risk/02_var_model
Execution code url	https://github.com/databricks-industry-solutions/value_at_risk.git

This is LAYER 3 - Brings technical overview

databricks

Model development, implementation and testing

This section dynamically pulls all the technical context around the implementation of the model itself. A given model registered on mlflow should have an associated experiment that can be linked to actual code at a given version. The goal is to document the approach taken by the model developer in the implementation of the model. We report all the technical metadata and specification of the artifact(s) logged on mlflow, the parameters used and output metrics.

Submitted artefacts

In this section, we report all binary artefacts that were stored alongside this model. Since a model may have multiple 'flavors' (or interpreter), we report each binary and their respective version.

Logged time	Artifact	Interpreter version
2023-02-18 05:26:28.949093	python_function	2.0.0

Developmental overview

This section will automatically retrieve the code associated with the model experiment. We report a databricks JOB output or a databricks NOTEBOOK markdown and their respective output cells. This becomes the responsibility of the model developer to document their approach with distinct sections and headers, from data sourcing and transformation, exploratory data analysis, feature selection, model selection and validation as well as model explainability when applicable. We recommend organizations to create template notebooks covering internal policies and external compliance requirements to ensure consistency and relevance of this documentation. Such policies will be seamlessly reported here.

markdown cell #0

Model building

We retrieve last 2 years worth of market indicator data to train a model that could predict our instrument returns. As our portfolio is made of 40 equities, we want to train 40 predictive models in parallel, collecting all weights into a single coefficient matrix for monte carlo simulations. We show how to have a more discipline approach to model development by leveraging MLFlow capabilities.

markdown cell #11

We create a temp directory where we may store some additional artifacts for our model

markdown cell #13

Compute returns

In the previous notebook, we already computed daily returns for each of our market indicators. Those will be used as features for our model, trying to predict investment returns for each of our instrument.

markdown cell #15

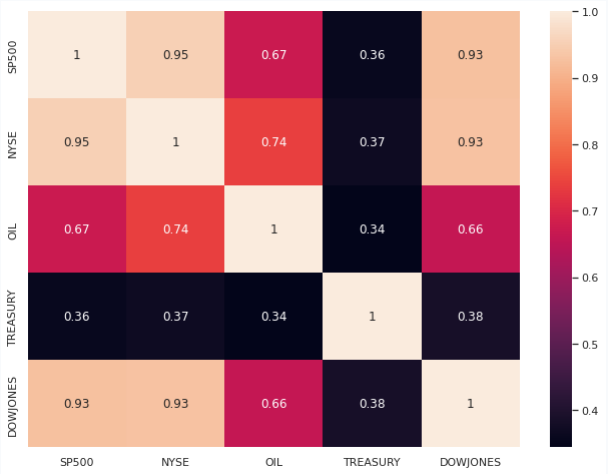
Let's compute daily returns of our investments. Given the size of a typical portfolio, we can leverage Window functions on spark to do so.

markdown cell #18

Create features

Risk models are complex and cannot really be expressed simply through the form of notebooks. This solution accelerator does not aim to build best financial model, but rather to walk someone through all processes to do so. The starting point to any good risk model will be to diligently study correlations between all different indicators (limited to 5 here for presentation purpose)

output cell #19



markdown cell #20

We join our market indicator data with stock returns to build an input dataset we can machine learn. We'll use`tempo` for this AS-OF join since our

timestamps may be differents in real life, with intra day tick data.

markdown cell #23

Building models

We show how any function or model can be easily wrapped as a `mlflow.pyfunc` model and registered as such on ml registry. Real life VAR models are obviously more complex than a simple linear regression described here and are not necessarily out of the box sklearn or keras models. Still, they should follow same ML development standard and can easily benefit from ml-flow functionalities as long as one can express model I/O as a form of `pd.Series`, `pd.DataFrame` or `np.array`

markdown cell #26

One can package an entire business logic (being statistical models or more AI models) as a simple`pyfunc`.

markdown cell #28

Such a model will be tracked, stored, registered and signature of the model enforced to prevent from data drift.

output cell #32

markdown cell #33

We can update our previous experiment with results of our prediction model (sum square of error)

markdown cell #35

The experiment captured now contains all libraries required to run in isolation and is linked to specific delta version to enable full reproducibility. By registering our model to ML registry, we make it available to downstream processes and backend jobs such as our next notebook focused on monte carlo simulations

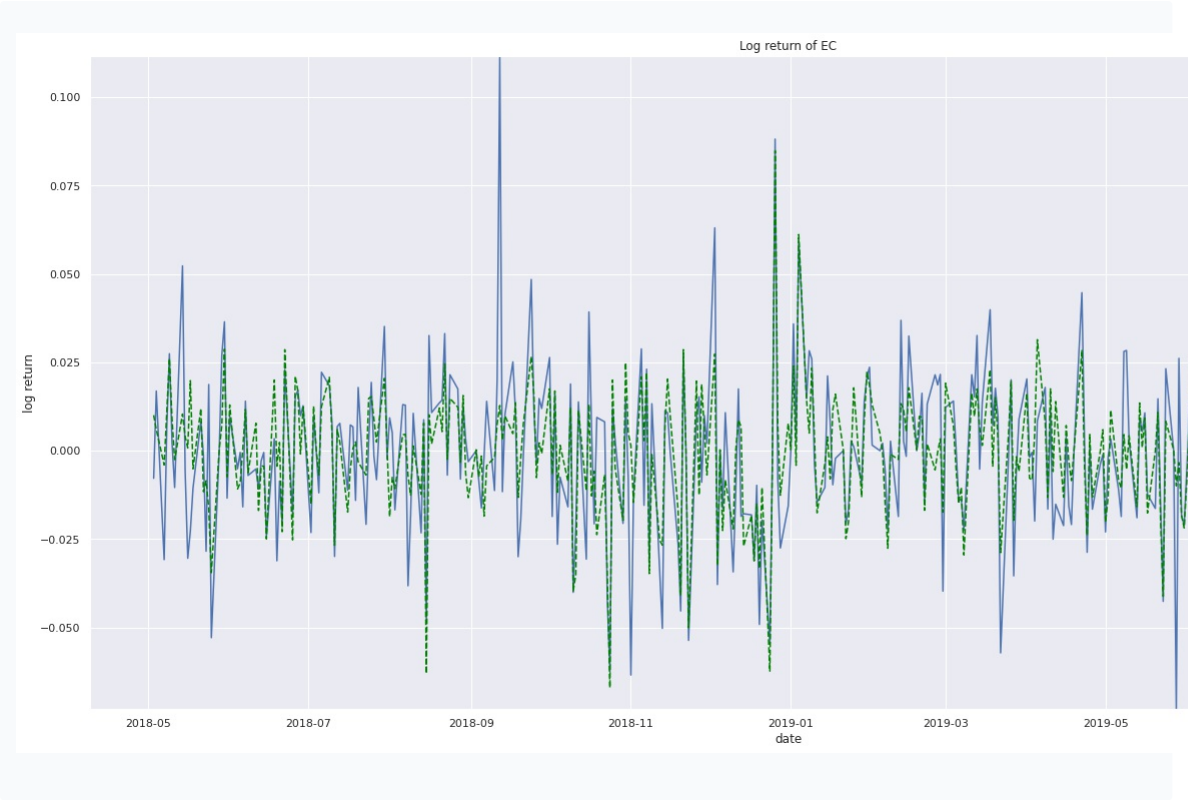
markdown cell #37

We can also promote our model to different stages programmatically. Although our models would need to be reviewed in real life scenario, we make it available as a production artifact for our next notebook and programmatically transition previous runs back to Archive.

markdown cell #41

Our model now production candidate, we can load our predictive logic as a simple user defined function and predict investment returns for every observed market condition

output cell #43



databricks

Model parameters

In this section, we report all the parameters used in the creation of this model. We highly recommend the use of mlflow auto-logging capability to ensure consistency of information reported from different teams and across different frameworks.

Param	Value
tickers	28

databricks

Model calibration

In this section, we report all the metrics logged in the creation of this model. We highly recommend the use of mlflow auto-logging capability to ensure consistency of information reported from different teams and across different frameworks.

Metric	Value
wsse	0.10339593800334608

Model dependencies

This section will retrieve all technical context surrounding the development of the model, the data set used, the input and output features, as well as infrastructure requirements and external libraries. This section will ensure model output can be reproduced under same conditions.

Infrastructure dependency

This section will programmatically retrieve the specification of the infrastructure used for the creation of the model. What environment was created, how many nodes were leveraged for distributed computing, what databricks runtime was used. We highly encourage users to leverage LTS versions of our runtimes.

Cluster name	value at risk
Cluster runtime	11.3.x-cpu-ml-scala2.12
Cluster instance type	i3.xlarge
Cluster number workers	1-6

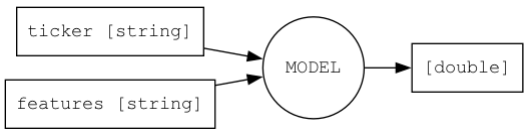
Libraries dependencies

Beside the cluster and infrastructure used for the creation of the model, specific libraries (open source or proprietary) may have been leveraged. This section will report all external dependencies (maven, pypi, custom packages) and their respective versions. We highly encourage users to install libraries at an infrastructure level rather than at a notebook level to ensure each library is properly tracked and reported here.

Could not find any associated libraries. Make sure dependencies are captured and installed as a cluster level (linked to an infrastructure rather than a notebook).

Input and output signatures

This section will programmatically represent the input features of the model and expected output signature. The transformations applied upfront should be documented as part of the developmental overview reported earlier.



Data dependencies

This section will report the different data sources used throughout this exercise. Using mlflow coupled with databricks notebooks, we should be able to track all data sources loaded through spark alongside their versions whenever possible. We highly encourage users to leverage delta format whenever possible to lock an experiment on a given data version we can easily time travel to.

Path	Format	Version
s3://databricks-e2demofieldengwest/b169b504-4c54-49f2-bc3a-adf4b128f36d/tables/b5eb7815-8f27-4cf7-bf3f-3a857a193054	delta	0
s3://databricks-e2demofieldengwest/b169b504-4c54-49f2-bc3a-adf4b128f36d/tables/f8cb6366-8c49-40be-8e97-2543a94589d1	delta	0

Model lineage

Whenever applicable, we will track the associated data lineage for every data dependency tracked in this experiments. Please refer to unity catalog to ensure lineage is captured end to end and reported here as a graphical representation.