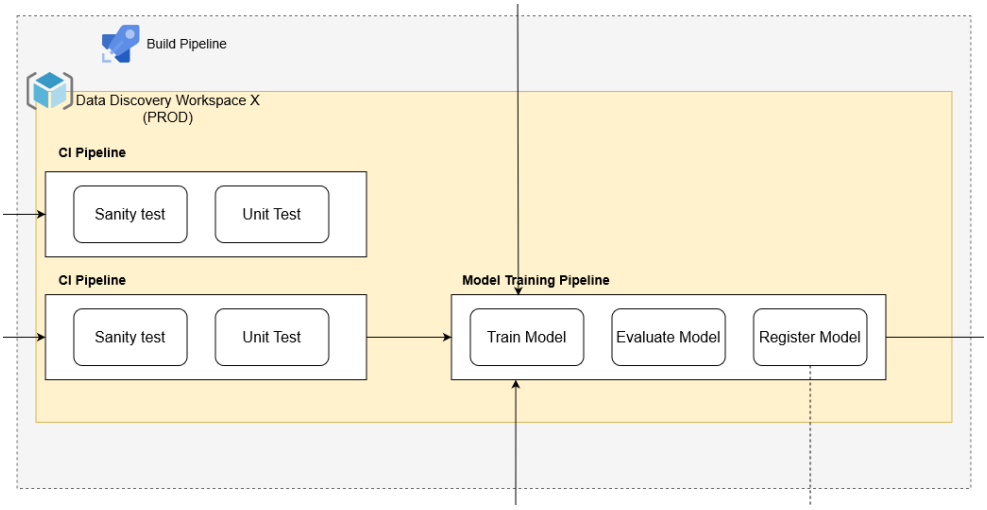
Pipeline design for model training, evaluation and serving

Model training pipeline is a part of *Build Pipeline* described in Design for Model Pipelines (MLOps).



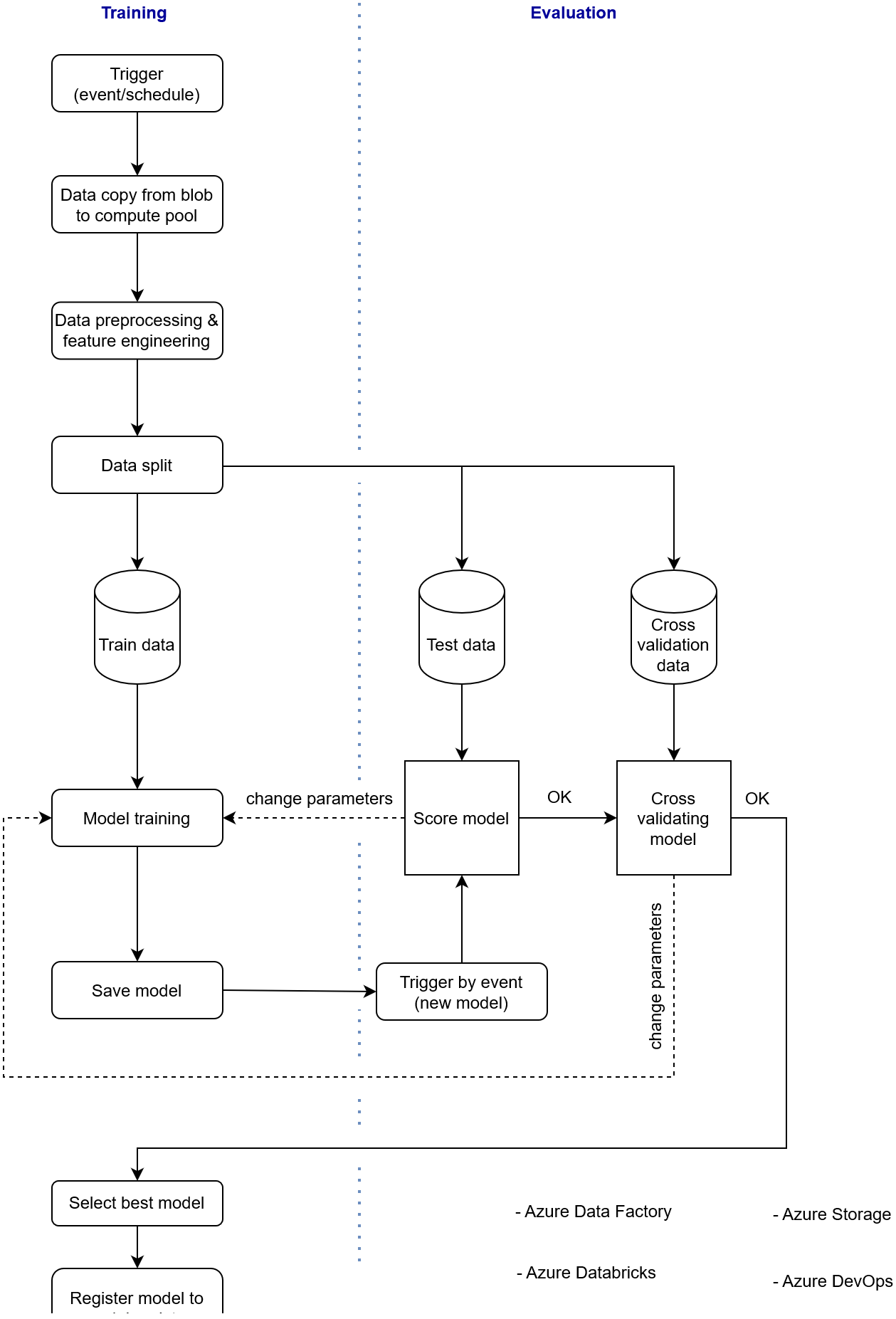
Model training pipeline consists of 3 parts:

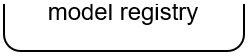
- Train model
- Evaluate model
- Register model

As was decided in the EAC S&D P2 sessions/workshops and can be seen from the diagram above *Model Serving* part of the pipeline belongs to Design for Model Pipelines (MLOps) so we will not discuss it here.

ML model pipeline

Let us begin with a diagram.


Training
Evaluation
Trigger (event/schedule)
Data copy from blob to compute pool
Data preprocessing & feature engineering
Data split
Train data
Test data
Cross validation data
Model training
change parameters
Score model
OK
Cross validating model
OK
Save model
Trigger by event (new model)
change parameters
Select best model
Register model to
- Azure Data Factory
- Azure Databricks
- Azure Storage
- Azure DevOps



- Azure ML

Now we can begin from the top of the diagram and explain the workflow going down. Here we have model training, scoring and registering parts.

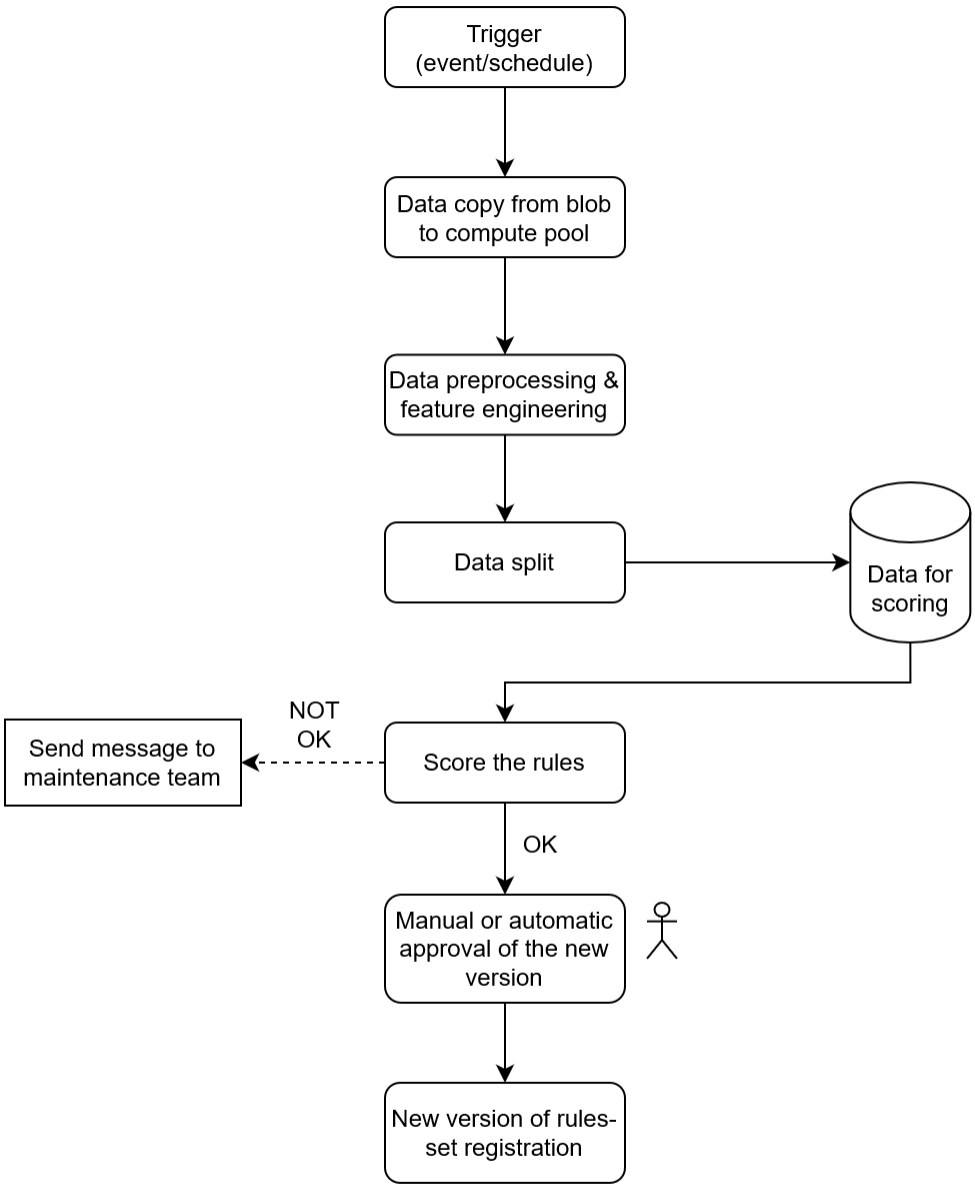
1. Training pipeline is started by the trigger(s) in Data Factory. The trigger could be a merge to master branch in Azure DevOps, also it can be scheduled to run periodically and the pipeline can be triggered by both.
2. After the trigger in Data Factory is hit - the second step is to gather/copy the data for the model from the Azure Data Lake(s) where the required data resides. After the copy is done - we have all the needed data in one place, stored in Databricks compute pool. The code which performs data copy resides in Databrics notebook, which is synchronised with Azure DevOps.
3. If previous step is successful the data then is being split into Train, Test and Cross validation subsets. It's done in Datatabricks which is synchronised with Azure DevOps.
4. When previous step completes - it starts model training step. Model training is done on the Train data subset from the previous step. Model training consists of experiments where model is trained using the predefined set of parameters. The scores collected from the experiments are saved. All these actions can be done with MLflow in Databricks or ML Service. As usual, the code resides in Databrics notebook, which is synchronised with Azure DevOps.
5. If the score result of the model meets requirements - it starts cross validation step. Now the experiments are performed on cross validation datasets.
6. Last two steps are repeated on the predefined parameters grid for the model.
7. After all possible combinations have been run - the best model is picked using best score results.
8. The best picked model is saved and registered in model registry at Databricks or Azure ML Service.

Let us end the description of the pipeline with a note - used resources and services as Databricks or Azure ML Service can be replaced with others like Synapse Analytics applying minimal architectural adjustments as the pipeline described here has a rather general form and is flexible to further evolution and development.

Rule-based solution pipeline

As there are quite a few differences between ML model pipeline and rule-based solution pipeline let us draw the separate one for the latest.

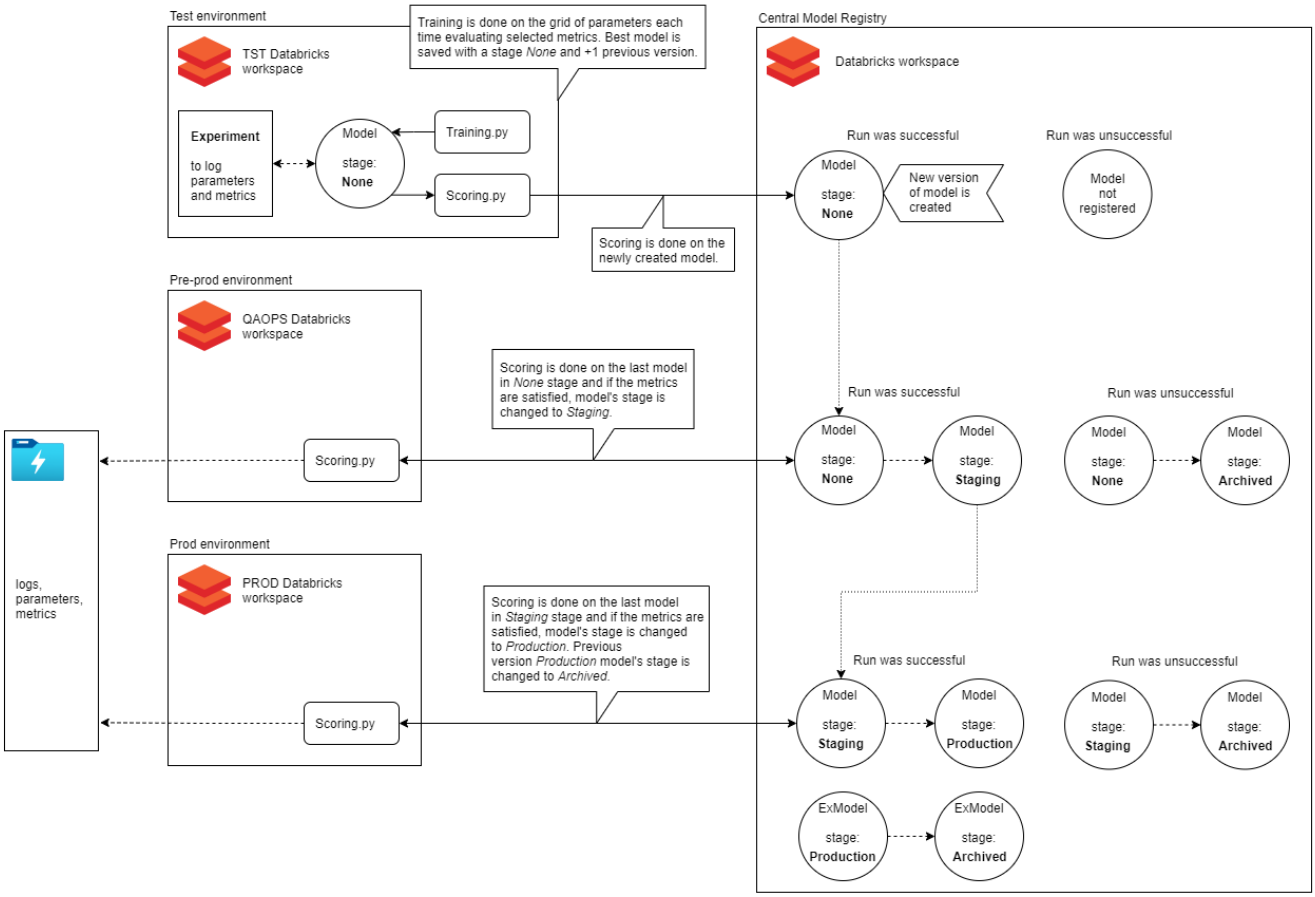
Scoring



- Azure Data Factory
- Azure Storage
- Azure Databricks
- Azure DevOps

The fist two steps of rule-based pipeline are identical to the ML model pipeline. The later differences are shown in the diagram and are self explanatory.

Model's stages and versions transitions design in Central Model Registry (CMR):



References

- https://docs.microsoft.com/en-us/azure/databricks/dev-tools/ci-cd/ci-cd-azure-devops
- https://techcommunity.microsoft.com/t5/educator-developer-blog/machine-learning-devops-mlops-with-azure-ml/ba-p/742150
- https://databricks.com/glossary/lambda-architecture