**Advanced Analytics (AA) Models Monitoring Framework**

**v.1.0**

Contents

[1. Introduction 3](#_Toc108173395)

[1.1. Scope 4](#_Toc108173396)

[1.2. General provisions 4](#_Toc108173397)

[1.2.1. Levels of Monitoring 4](#_Toc108173398)

[1.2.2. Terms & Abbreviations 5](#_Toc108173399)

[2. Functional Level Monitoring 6](#_Toc108173400)

[2.1. Input Data 6](#_Toc108173401)

[2.1.1. Basic Scope 6](#_Toc108173402)

[2.1.2. Design Considerations 8](#_Toc108173403)

[2.1.3. Data integrity and data drift – case study 11](#_Toc108173404)

[2.1.4. Workflows 13](#_Toc108173405)

[2.1.5. Documentation 13](#_Toc108173406)

[2.2. Model & Outputs 14](#_Toc108173407)

[2.2.1. Basic Scope 14](#_Toc108173408)

[2.2.2. Design Considerations 15](#_Toc108173409)

[2.2.4. Workflows 20](#_Toc108173410)

[2.2.5. Documentation 20](#_Toc108173411)

[3. Operational Level Monitoring 20](#_Toc108173412)

[4. Logging and Alerting 22](#_Toc108173413)

[4.1. Logging 22](#_Toc108173414)

[4.2. Alerts 24](#_Toc108173415)

[5. Minimum requirements 24](#_Toc108173416)

[5.1. Monitoring and retraining frequency 25](#_Toc108173417)

[5.1.1. Significant change in business environment 25](#_Toc108173418)

[5.1.2. Statistical KPIs are not fulfilled 25](#_Toc108173419)

[5.1.3. Business KPIs are not fulfilled 26](#_Toc108173420)

[5.1.4. Data drift is detected 26](#_Toc108173421)

# **Introduction**

With recent developments on variety and volumes of collected data, more and more business processes start to rely on analytical models as an **integral part of decision-making**. Not only the number of analytical models has increased over the last years, but so did their complexity and, in turn, the level of sophistication involved in model development process.

However, in a general case, regardless of exact algorithms applied, amount of data used or other peculiarities of the model development, **any analytical model will degrade over time.** Such property reflects inevitable changes in underlying data generating processes (be it due to organic or inorganic reasons).

It therefore calls for **periodic intervention** in the model (in a form of a model re-fitting, adjustment or a complete re-development) to maintain analytical model “in fit” for solving defined business problem with accepted level of risk[[1]](#footnote-2). Whilst the necessity of such intervention is clear, yet another question remains unsolved – **which particular situations call for** **action and how to identify them consistently? Which stakeholders must undertake those interventions or be informed of their results?**

Answering above questions constitute the design of model monitoring in production – an equally important phase in the model lifecycle as the model development itself.

**Figure 1. Analytical Models Lifecycle (source:** [**SAS**](https://www.sas.com/en_us/insights/articles/analytics/modelops.html)**)**

Shape, arrow

Description automatically generated

Essentially, the **goals of monitoring** analytical models in production are[[2]](#footnote-3):

* To detect problems with the model and underlying / enabling systems **before** they start to generate negative business value,
* To **take action** by triaging and troubleshooting models or the inputs and systems that enable them,
* To ensure the model’s prediction process and its outputs / results are **transparent** to relevant stakeholders,
* Finally, to provide a **path for maintaining and improving the model.**

## **Scope**

The current framework covers the broad area of analytical models monitoring in several aspects.

First, it elaborates on the **choices**to be made while designing monitoring methodology for a given model,including:

* Validation of **input data**
* Validation of **intermediate / output artefacts** of the model
* Overseeing **model quality** from **statistical** and **business** points of view
* **Technical** model monitoring

As a next step, a guideline is provided for setting up a stable **monitoring governance** (i.e., processes, workflows and responsibilities involved in the monitoring process).

Finally, this document concludes with the description of **technologies** supporting selected monitoring methodology and governance.

It must be noted that the scope of present framework is bound solely to Retail Business analytical use-cases (developed either by HO AA&AI Tribe, RBRS AA Hub or NWBs) and do not cover respective topic of other use-case areas.

## **General provisions**

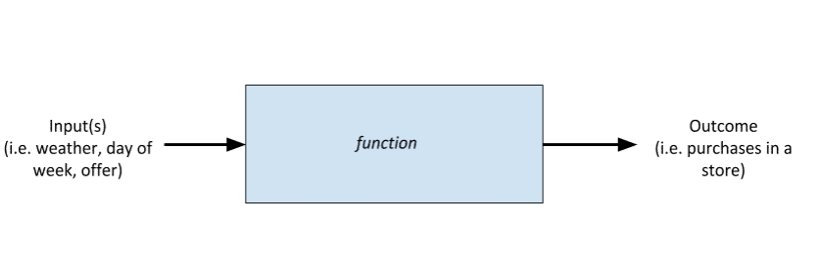
* + 1. **Levels of Monitoring**

For the purpose of the current framework, we adopt a **dual view** on the term **“analytical model”:**

**View 1:** AA model as a **calculation algorithm** with several components:

1. Input data and its characteristics:
   1. Number and list of input features
   2. Pre-processing logic performed to transform original features into the format expected by the model (e.g., changing scale, handling missing values, outliers, one-hot encoding of categorical variables)

**Figure 2. Analytical Models as a Calculation Algorithm (source: [])**



1. Mathematical / computational logic (i.e., model in a narrow sense) with defined architecture, (hyper)parameters or calculation flow
2. Any transformation of direct model output (parametrized or not) to arrive at the final model output

**View 2:** AA model as a **collection of software artefacts / codes**, which may or may not contain lower-level software units as their parts (e.g., classes, functions, packages).

In other words, when addressed holistically, a machine learning application isn’t just the model, but everything that enables executing the model in production, including infrastructure, input data, resources, and, optionally, surrounding upstream and/or downstream services.

Afore mentioned duality justifies the necessity for **several levels of monitoring** to be performed during the model lifecycle, addressed as functional and operational monitoring:

* **Functional** level monitoring – monitoring model performance (in statistical and business terms), inputs (data), and outputs (predictions),
* **Operational** level monitoring – monitoring at system and resource level.
  + 1. **Terms & Abbreviations**

| **Term / Abbreviation** | **Explanation** |
| --- | --- |
| Data drift | A change in distribution between the training data and production data, which is not related to data quality issues |
| False positive / negative | Any prediction, which does not match the original label (i.e., a prediction of 0 for an event labelled with 1 or a prediction of 1 for an event labelled with 0) |
| Negative event | An event, which is opposite to “positive” (i.e., labelled with 0) |
| Positive event | An event, which is labelled with 1, is called “positive” |
| ROC | Receiver Operating Characteristic |
| ROC AUC | Area Under ROC Curve |
| True positive / negative | Any prediction[[3]](#footnote-4), which matches the original label (e.g., a prediction of 1 for an event labelled with 1) |

# **Functional Level Monitoring**

## **Input Data**

* + 1. **Basic Scope**

Since any analytical solution rests on certain **assumptions**, it is expected that the properties of input data remain vastly compliant with the ones observed during model development. When this principle fails to hold, the model will sooner or later start producing unexpected, no longer valid, or even incorrect outputs (garbage in - garbage out principle).

It is therefore of utmost importance that respective stakeholders are duly notified of **data quality issues** or **organic / inorganic changes in data distribution (data drifts)**, which might impact the model performance in production.

Apart from statistical properties of underlying data, an analytical model expects inputs to be provided in a certain **format** and to specified **location,** which, if not satisfied, will not allow the calculation to proceed further. While such situation will generally be discovered way earlier than any changes in the data distribution (in particular, at the moment when computation crashes), it is advised to implement certain controls, which would simplify troubleshooting of respective format issues and notify functions in charge of: 1) data delivery and 2) model execution.

**Table 1** and **Table 2** summarize **basic / minimal scope** forboth aspects of input data monitoring:

* Data Integrity[[4]](#footnote-5)
* Data / Features Drifts

**Table 1. Data Integrity Checks[[5]](#footnote-6)**

| **Issue** | **Control Mechanism** | **Criticality Level** |
| --- | --- | --- |
| Data is unavailable at expected location (e.g., due to data loss / corruption at the source) | Check for presence (yes/no) and format (e.g. .csv, .parquet) of data files |  |
| Schema errors / changes (feature names) | Data scheme validation / check for syntax errors, column type and format errors in column names |  |
| Data type / format errors |  |
| Missing values in non-nullable columns; categories not in the defined range for categorical variable |  |
| Duplicated Data | Check for duplicates in primary key columns on a table basis |  |
| Unexplained pipeline failure | Investigate pipeline execution logs |  |

It shall be noted that the checks above must be complemented during the model development phase by other meaningful tests tailored to the exercise at hand. In fact, one deliverable of the model development process is an extended version of such table, describing all necessary monitoring activities.

**Table 2. Data Validation Checks[[6]](#footnote-7) (exemplary)**

| **Issue** | **Control Mechanism** | **Criticality Level** |
| --- | --- | --- |
| Feature Drifts | Measuring distribution changes between current and reference data set metrics via distribution tests / heuristic comparison:   * **Basic statistical metrics** such as mean/average value, share of missing values, standard deviation, min. and max. values comparison, correlation * **Continuous** features: divergence and distance tests (e.g. [Kullback–Leibler divergence](https://en.wikipedia.org/wiki/Kullback%E2%80%93Leibler_divergence), [Kolmogorov-Smirnov statistics](https://en.wikipedia.org/wiki/Kolmogorov%E2%80%93Smirnov_test), Population Stability Index (PSI), [Hellinger distance](https://en.wikipedia.org/wiki/Hellinger_distance), etc.) * **Categorical** features: [chi-squared test](https://en.wikipedia.org/wiki/Chi-squared_test), [entropy](https://en.wikipedia.org/wiki/Entropy_(information_theory)), the cardinality or frequency of the feature category |  |

* + 1. **Design Considerations**

While a part of control mechanisms, which were described in previous article, may seem fully deterministic, in fact all of them require further parametrization. In particular, one needs to ask himself**:**

* **How frequently** will I monitor selected metrics?
* On which **time horizon / data** do I calculate the **most recent** value of monitoring metric?
* On which **time horizon / data** do I calculate the **benchmark** value of monitoring metric?
* Are the data sets used for calculating metrics **comparable by design**[[7]](#footnote-8)?
* **How do I compare** most recent value of monitoring metric with the benchmark one?

Above considerations aim to complete the **design** of monitoring system from **methodological** point of view and shall be addressed **holistically**, as a choice in one category may materially impact the others. Nevertheless, in forthcoming part of the chapter we are covering each of these points with dedicated recommendations.

* + - 1. **Frequency**

The **frequency of input data monitoring** may be considered a function of **three parameters**:

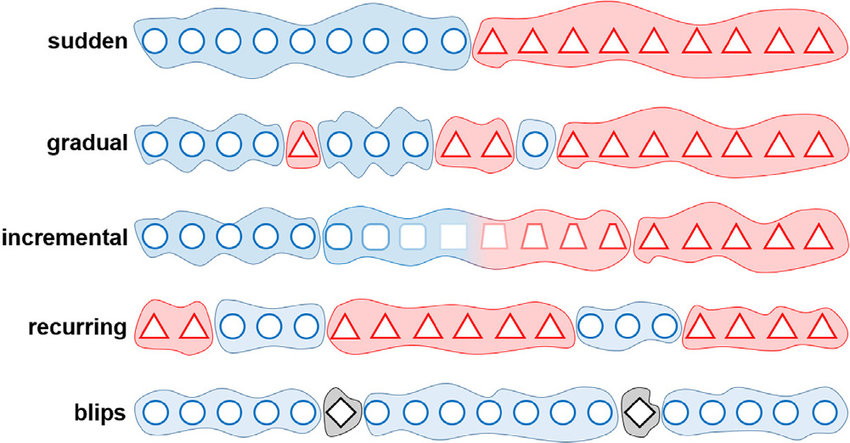
* + - 1. **Horizons**

Once more clarity on desired frequency of monitoring is acquired, the design of monitoring regime continues with selecting a proper **evaluation and benchmark horizons** for metrics under question. Alternatively, it might be understood as selecting the **underlying data basis** for respective metrics calculation.

Such choice remains under discretion of the model developer and vastly depend on **properties of underlying data generating process** (stability, degree of seasonality, etc.), as well as target variable definition. As a general recommendation, it is **advised to** investigate **historical behavior** of selected metrics on **development and more recent samples** and use this analysis to inform the decision on both horizons.

An important consideration to keep in mind at this point is the **variety of data drift types**, which might be present in the data, as illustrated in Figure 3 below.

**Figure 3. Types of Data / Concept Drift**[[8]](#footnote-9)



Observing or anticipating any type of those changes will further provide guidance for the model developer to decide on **combinations of recent / benchmark periods** to be compared within the monitoring. While it is not strictly necessary for monitoring system to watch out for all below defined data drifts, it is advised to account for, at least, several of them.

* + - 1. **Benchmarks & Critical Values**

At this point in time the monitoring methodology is vastly finalized, with only one point remaining to be clarified, namely: having a pair of KPI values from the recent observation period and a benchmark one, how does one conclude on **materiality of changes**?

To answer those questions again a multitude of factors shall be considered:

* **Behavior of selected metric in history**, preferrable a back-simulation of KPI on several overlapping / non-overlapping “monitoring horizons” is to be performed
* **Academic suggestions** for interpretation and threshold values of selected KPIs
* **Degree of model robustness** towards changes in underlying data

The general recommendation is to specify, **at least, two types of monitoring outcome** („green“ and „red“), which would further on guide the further actions to be taken („continue monitoring“ and „take action“ respectively). However, if a certain **differentiation of actions** makes sense, an additional assessment outcome („yellow“ or „amber“) might be introduced.

* + 1. **Data integrity and data drift – case study**

The goal of this case study is to show an example of problems that can be encountered during the production run of analytical models.

As can be seen from Table 1 and Table 2 above, data integrity and data validation checks are the first line of defense that should be executed to confirm desirable results. For this reason, this case study will demonstrate these checks in action on an artificial data set from Kaggle. The working environment used for creating this example was Databricks platform and all the code can be found on this [link](https://rbinternational.sharepoint.com/:f:/s/ACA/EjSrHMnfiBhGkO86DkriaqMBOfVSGje6E0wWvSG-dB7jHg?e=DL5DWQ).

The first check to be done when preparing for productive run of the model is to confirm the presence of all relevant data at locations that they are expected to be in.

| **Issue** | **Control Mechanism** | **Criticality Level** |
| --- | --- | --- |
| Data is unavailable at expected location (e.g. due to data loss / corruption at the source) | Check for presence (yes/no) and format (e.g. .csv, .parquet) of data files |  |

Such checks should be fully automated and included as a separate step in production pipeline, so that an alert is raised immediately, and a notification is provided to ML Ops to act upon this problem. Human intervention is then needed to check where exactly the problem occurred and to further fix this.

After successful completion of this step, we proceed with checking the content of the data itself. This would include checking if:

* schema is consistent with what the model is expecting
* missing values in columns exceed previously agreed thresholds
* duplicate rows are present
* other checks listed in Table 1

Confirming if data corresponds to the expected schema is done inherently through Apache Spark - after passing the expected schema for Spark to create a data frame, Spark automatically checks if the data is aligned with the provided schema. This way, we avoid problems that could arise from "feeding” incorrect data types to the model and breaking the prediction pipeline execution out-of-the-box. At the same time, it can also be done through various data validation libraries prior to initialization of the data load.

| **Issue** | **Control Mechanism** | **Criticality Level** |
| --- | --- | --- |
| Schema errors / changes (feature names) | Data scheme validation / check for syntax errors, column type and format errors in column names |  |
| Data type / format errors |  |

An example of schema inconsistency would be if the model expects the ’balance’ column to be an Integer and imported data stores it as a String for some reason. This would cause the pipeline to break immediately or at some later point in time. At the same time, if we are enforcing schemas from the beginning, we could investigate problems earlier or even set up automated solutions for some of them.

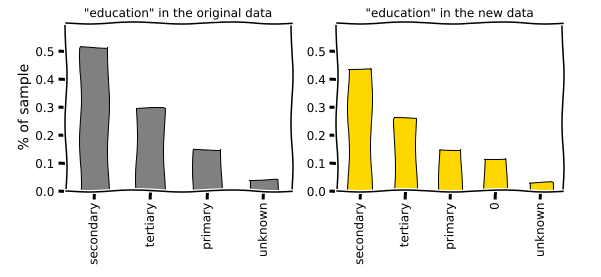
Apart from more technical integrity checks (i.e., the ones which are necessary for the model pipeline to successfully execute) we need to check for consistency of the provided data as well.

| **Issue** | **Control Mechanism** | **Criticality Level** |
| --- | --- | --- |
| Feature Drifts | Measuring distribution changes between current and reference data set metrics by [distribution tests](https://itrcweb.org/gsmc-1/Content/GW%20Stats/5%20Methods%20in%20indiv%20Topics/5%206%20Distributional%20Tests.htm) / heuristic comparison |  |

Taking missing values as an example: assuming delivered data contains a larger portion of missings, one might observe a worse model performance, resulting from the lack of information for the model to decide on (bias introduced by imputation). In Figure 4 we illustrate a change in ‘education’ column in our “toy” dataset – notice how a ‘0’ category has appeared as a result of careless imputation, when an unusually large share of missing values has not been properly investigated.

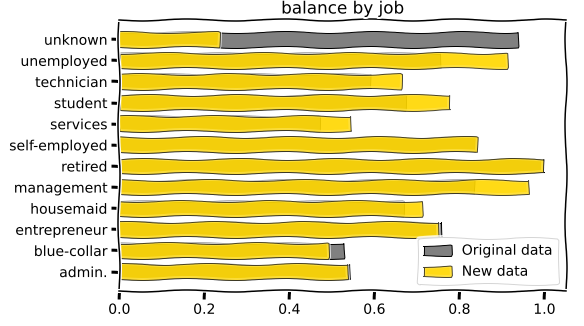
Duplicate rows could also be an example of data inconsistency where we, for some reason, get additional rows of exactly the same data, which is skewing the distribution and trueness of data.

**Figure 4. Introduction of missing values in data: before and after**



Finally, in production, we could encounter phenomena known as data drift. In this case, there is nothing wrong with our data (i.e., no duplicate rows or missing values) but we still see changes in the distribution of variables, which might affect the model performance. Contrary to data integrity problems, such situation is likely to occur due to an external event impacting the data generating process. This could be dangerous since it’s much harder to detect in the right way than, for example, duplicated rows, so that a possible model deterioration is only detected de-facto, when the target data becomes fully available. In Figure 4 we have illustrated how a generous distribution of “helicopter money” to at-risk employment categories would reflect in our banking dataset – balance distribution by employment category has changed.

**Figure 5. Change in data between deliveries due to external factors**



* + 1. **Workflows**

Process & Responsibilities Split & Actions

TBD after APEX release 3.0

* + 1. **Documentation**

Min. requirements on documentation / template

TBD after APEX release 3.0

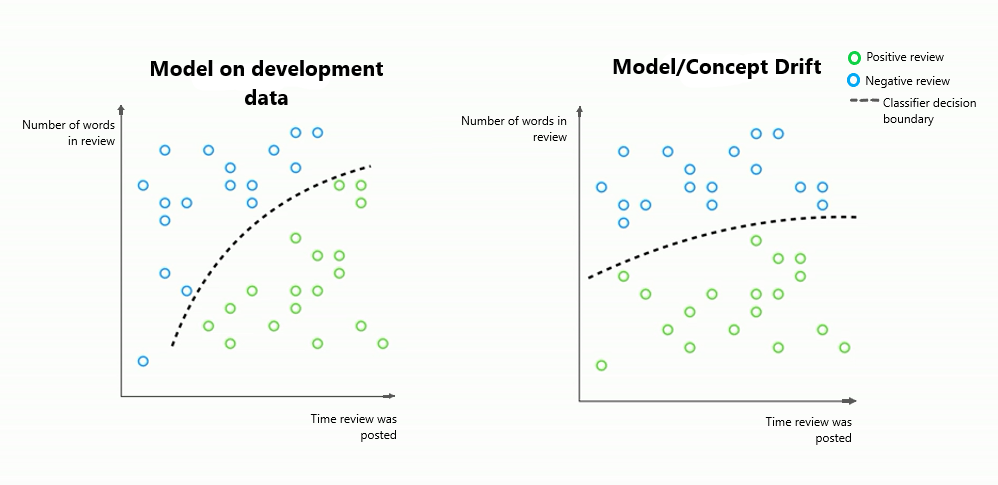
## **Model & Outputs**

* + 1. **Basic Scope**

Despite the best efforts invested in development process, any analytical model will tend to **deteriorate in performance over time.** Provided that the input to analytical solution is correct in both format and content, such deterioration will usually reflect the inevitable changes in processes/patterns being modelled, referred to as **concept drift**.

Model drift, or [concept drift](https://neptune.ai/blog/concept-drift-best-practices), happens when the **relationship between features and/or labels no longer holds** because the learned relationship/patterns have changed over time (see Figure 6for illustration).

**Figure 6. Model / Concept Drift Visualization[[9]](#footnote-10)**



Similar to input data drift, model/concept drift can happen with a **different speed and/or frequency**. Nonetheless, it is not guaranteed that the two drifts (data and concept drifts) will be observed concurrently. Validating if the change in the model performance is caused by skew in underlying data (or e.g. bugs in the data pipeline) shall be subject to additional investigation.

When it comes to **identifying a model drift**, several control mechanisms may be applied in production, either building upon model performance KPIs (as specified during model acceptance) or predictions properties directly (see Table 3).

**Table 3. Model / Concept Drift Checks**

| **Issue** | **Control Mechanism** | **Criticality Level** |
| --- | --- | --- |
| Performance Deterioration (i.e. significant worsening of model performance KPI(s)) | Comparison of current KPI values against benchmarks |  |
| Output / Label Drifts | Similar to Feature Drifts, see Table 2 |  |

* + 1. **Design Considerations**
       1. **Selection of KPIs**

As, by design, any analytical use-case is aiming to fulfill a certain **business objective**, it is important to continue checking in production, if it still **serves the original purpose with the same quality.** Provided that the model development and testing was performed in line with the [Model Development Rulebook](https://rbinternational.sharepoint.com/:w:/r/sites/ACA/AA%20Tribe/08%20Advanced%20Analytics%20Community/07%20Model%20Governance/AA%26AI%20Model%20Development%20Rule%20Book%20v1.2.docx?d=w3f9c03fc20534b3cb9da7de1e7921c02&csf=1&web=1&e=ccFjq1), the business KPIsto be monitored would be **already known prior to setup of monitoring** and reflect an **agreed state of model quality** at the moment of its **acceptance.**

This statement, however, does not preclude one from specifying additional metrics to be regularly scrutinized, provided that they follow certain minimum requirements:

* KPI is actively supporting the business objective of a use-case
* Can be collected in production with a **reasonable effort and time**
* Allows for **actionable alerting** on problems.

The same considerations, as given above, hold true for another “flavor” of KPIs to be monitored after use-case go-live – **statistical KPIs,** which signify the quality of the model from **data science perspective.**

A few additional remarks shall be made here regarding **supervised learning** models and **availability of “ground truth” in productive environment.**

In practice several scenarios for ground truth availability are possible depending on a given use-case and design of the business process (Table 4).

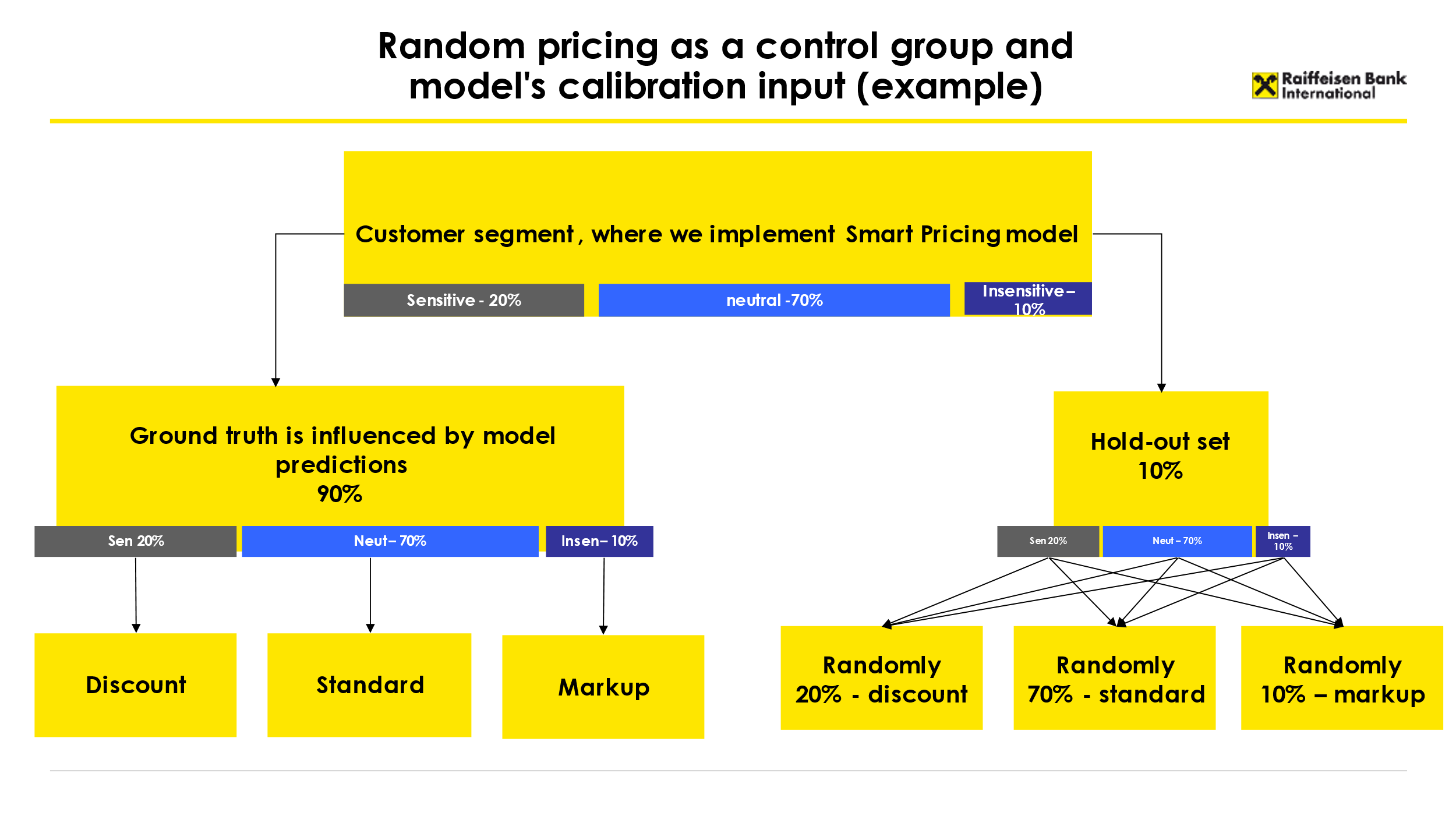
**Table 4. Ground Truth Availability Scenarios**

| **Scenario** | **KPIs Selection** |
| --- | --- |
| Actual labels are available **real-time** or (nearly) **at the same time when the prediction is done / used** | * Perform monitoring with the **same** model performance metrics, as used for the model selection / evaluation in **development phase** |
| Actual labels become available **substantially later** that respective prediction is made/used | * Perform monitoring with the **same** model performance metrics, as used for the model selection / evaluation in **development phase** **(delayed in time)** * Monitor **proxy metrics** - alternative signals that are correlated with the ground truth. Recommended if a **more dynamic** monitoring is required compared to the cadence of ground truth availability. Model predictions distribution may be seen as a special case of a proxy metric. |
| Actual labels are **not available** **/ observable / are compromised** | * Monitor **proxy metrics** (incl. model predictions distribution) |
| Ground truth is **available**, but **influenced by model predictions** (see Figure 7) | * Comparison of current KPI values against KPIs calculated on a **hold-out set** **/ global control group** where model’s predictions are **not followed** (see Figure 8) |

**Figure 7. Causal Influence on Ground Truth / Biased Ground Truth (loan approval process)[[10]](#footnote-11)**

Graphical user interface

Description automatically generated with medium confidence



**Figure 8. Example of hold-out set for Smart Pricing use case**

* + - 1. **Frequency**

Since methods for concept / predictions monitoring are vastly overlapping with the ones valid for data drifts, **all same principles** (frequency, horizons, benchmarks selection – see **Section 2.1** **Input Data**) are applicable for **both types of monitoring**.

Nevertheless, a few noticeable remarks are to be made on **potential differences** between the two monitoring frameworks:

* + The frequencies of model and data monitoring might be **different in both directions**:
    - Model / concept monitoring may be **less frequent** than data monitoring: such situation might arise in case when some of model quality KPIs (especially the ones, which have to do with business impact produced by the model) require **manual calculations** efforts.
    - Model / concept monitoring may be **more frequent** than data monitoring (e.g., monitoring conversion rate of campaign, which is based on model predictions, several throught the campaign duration).
  + Monitoring of model-level KPIs will be generally less homogenous compared to **data monitoring**. For instance, the **business** and **statistical KPIs** on the model level might be evaluated with a **different cadence**, depending on availability of respective data.

**Example:**

* A model, which predicts outstanding debt for the customer in other Banks, was trained on historical snapshots on Credit Bureau queries. Such queries contain data for entire Bank population and were purchased for the purpose of pre-selected/pre-approved lending.
* Under the target process, the Bank will stop purchasing Credit Bureau massively for every pre-selected / pre-approved campaign, but rather repeat this once in 6 months. The model predictions will be used instead of real Credit Bureau data for pre-selected offers and then will be followed up with a Credit Bureau inquiry only for converted leads.
* With such setup, the monitoring of model business quality (i.e., % of rejected clients, where only one knock-out based on the model prediction was triggered) may be monitored for every campaign, whereas statistical model quality could be assessed in full only once in 6 months, where CB data becomes available for the entire modelling population.

Last, but not least - **the more impact** (e.g. financial, reputational) is expected from the adverse model behavior, **the more frequently** it shall be monitored. From this perspective the frequency of (input) monitoring is closely related to the notion of **model materiality**, as described in [Model Development Rulebook](https://rbinternational.sharepoint.com/:w:/r/sites/ACA/AA%20Tribe/08%20Advanced%20Analytics%20Community/07%20Model%20Governance/AA%26AI%20Model%20Development%20Rule%20Book%20v1.2.docx?d=w3f9c03fc20534b3cb9da7de1e7921c02&csf=1&web=1&e=ccFjq1).

* + - 1. **Horizons**

Selection of evaluation and benchmark horizons for model KPIs is tightly following recommendations for design of data monitoring (please, refer to **Section 2.1.2.3 Horizons**).

On the other hand, one shall additionally consider **potential deviations**, similar to the ones listed in section 2.2.2.2 Frequencies:

* Difference between horizons used for **data** and **model** monitoring
* Difference between horizons used for **business** and **statistical** KPIs

in his/her decision on model KPIs calculation.

* + - 1. **Benchmarks & Critical Values**

A great similarity can be found between setting up critical values for data and model-level KPIs (with the former laid down in **Section 2.1.2.3** **Bench****marks & Critical Values**).

At the same time, apart from suggestions given for data monitoring metrics, one might consider an **additional benchmark option** for model KPI, such as **fixed “aspiration” / business objective** value. Since usually business stakeholders express their expectations towards minimum / desired level of model performance in terms of business KPIs, such value will generally be available before model development or latest after the model acceptance.

On top of this, another important difference shall be taken into account: contrary to data monitoring, model-level KPIs shall be equipped with **three** types of monitoring outcome („green“, „yellow“ or „amber“ and „red“), which are triggerting pre-defined actions / analysis on the side of model stakeholders.

* + 1. **Concept drifts – case study**

Concept drift results in a change in the relationship between input features and labels, for example, due to seasonality. This can quite easily lead to invalid model predictions and the current model. It is important to note that the sole presence of drift does not necessarily mean that we should retrain/replace the current model. In other words, if the current model still correctly predicts the labels even though there is a change in features, then there is no need for retraining/replacing the model. However, being aware of these changes is as important as adjusting our models to them, since in case there is not any reason for a change of model, we might still want to take some different business actions because of detected drifts in data.

Even though concept drifts are somewhat harder to detect and confirm since they imply a change in the relationship between both input features and labels, this is not impossible to do. However, an important thing to note is that detecting concept drifts, or any data drifts in general, is only possible when we have historical data to compare our current data to.

A good example of this process can be found on this [link](https://rbinternational.sharepoint.com/:f:/s/ACA/EkFYWlp8j9hBvZ_Z44e9Q3oB8AIYKsI7cr_5TL0jbycoeQ?e=EMWych) and includes checking for concept drifts in Airbnb listings on a monthly basis. Detecting these changes would come after doing all the necessary data integrity checks explained in the previous case study from this document. When we confirm data integrity, the next step is to check for the distribution of all the features which are of interest to us and compare it with a distribution of the same variables in our historical data. In some cases, simply using summary statistics (mean, median, variance, etc.) is enough to see more obvious changes in distributions. If that is not the case or if we want to have a more consistent metric, we can resort to using various tests (Population Stability Index, Kolmogorov-Smirnov, Chi-Squared, etc.) to help us determine whether our data has drifted.

In this case, we observe a drift in our label data and thus start our analysis. We conclude that the cause of this drift in the most recent month was due to data being captured during peak vacation season when Airbnb hosts have increased the price of their listings. In this case, our next course of action would be to retrain the model and include this recent month of data during training to try and capture this change in prices. After retraining, we compare the two models (new candidate and model in production) to see how they perform. If the new model does not meet our expectations, we avoid putting it in production and instead search for other ways to improve model performance like tuning hyperparameters of the current model, adding additional features, etc. For example, our problem might be solved by introducing the variable “month of the year” which would allow us to capture the temporary impact on listing prices.

In the end, concept drift is a phenomenon that should be solved on a case-by-case basis. As there is no one size fits all solution, we should take time to investigate after detecting its presence through the methods explained above.

* + 1. **Workflows**

Process & Responsibilities Split & Actions

TBD after APEX release 3.0

* + 1. **Documentation**

Min. requirements on documentation / template?

TBD after APEX release 3.0

# **Operational Level Monitoring**

As indicated in previous chapters, any AA model can be considered both from conceptual (statistical, business) point of view, as well as a specific IT application, which requires proper monitoring and maintenance.

In particular, when viewing analytical models from a software perspective, a topic of **operational level monitoring** is kicking in, which primarily encompasses the following questions:

* Does the application meet uptime requirements?
* Is it serving requests quickly enough?
* Is it efficiently utilizing resources?
* Does it meet scalability requirements?
* What are its serving limitations?
* How about changes in code dependencies, can the application handle it?
* Is it easy to identify the source of the problem, shall model execution fail? Why do certain actions (e.g. model retraining) take so long to execute compared to expectations / average execution time?

Table 5 summarizes dimensions to be looked into, while tracking model operational performance.

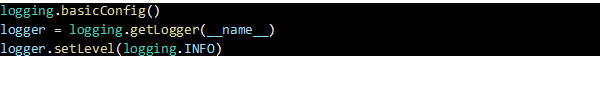
**Table 5. Model Operational Monitoring**

| **Dimension** | **Parameters** | **Responsible** | **Informed** |
| --- | --- | --- | --- |
| **System** | | | |
| **System Performance Metrics** | * **CPU/GPU utilization** per **type of process** (scoring / re-fitting) and **pipeline step** (data quality check, data validation, data pre-processing, etc.) * **Memory utilization** for when the model caches data or input data is cached in memory * **Response time (latency)** of the model server or prediction service[[11]](#footnote-12) * **Execution time** for every task / step of the pipeline * **Pipeline performance:** scheduled run time of a job, actual run time, how long it took to run, and the state of the job (successful, or failed job?) | MLOps | Data Engineer |
| **System reliability** | **Infrastructure and network uptime,** e.g.:   * how many clusters are running * which of the machines are running * SLAs | APEX DevOps | DevOps / MLOps |
| **Security** | **?** |  |  |
| **Pipelines** | | | |
| **Data & Model Pipelines** | **Input Data & Intermediate workflow steps:**   * are the **inputs** and **outputs** (i.e. number of files and file types) of every task / step in the pipeline as expected?   **Output data:**   * is the **output data schema** as expected by the machine learning model in terms of features and feature embeddings? * What’s the typical **file size** expected from an **output file**?   **Dependencies:** versions of each dependency the model runs on | MLOps | MLE  Data Engineer |
| **Costs** | | | |
| **Platform Costs** | * **Data storage costs** * **Cost of computational resources** | Retail AA PO  AA Hub Lead | Business PO (HO, NWB) |
| **Service Level Agreements / SLAs** | * **SLAs between distinct model-related services (e.g, prediction service) and consuming applications** | Retail AA PO  AA Hub Lead  Business PO (NWB) | NWB |

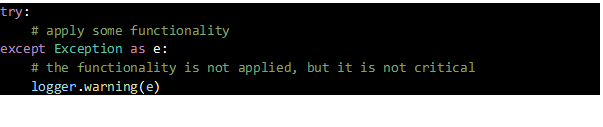
1. **Logging and Alerting**

## **Logging**

Creating logs for deployment pipelines is important in order to make bug fixing easier. If we go over from testing the deployment pipeline to stable operations we can at the same time change the log level[[12]](#footnote-13) from debug to info. For instance, in a stable operations environment our basic logging configuration could look like this:



Log messages are especially important for handling exceptions and if/else-conditions, like

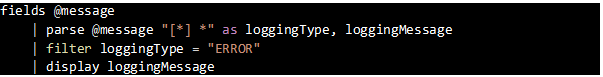


On debug level urls, file paths, data output, … should be logged in order to have a good overview on the pipeline while testing it in the first weeks after deployment. Log information on **critical turning points** of the pipeline, e.g. “Data loading”, “Data pre-processed”, “Model training started”, etc. are also nice to have at an info level and help a lot in a stable deployment pipeline for debugging. In Table 6 you find some examples of typical logged information and the corresponding log level.

**Table 6. Objects / artefacts to be logged in production**

| **Dimension** | **Objects / Artefacts** |
| --- | --- |
| **Data** | * Debug: Path to the dataset / source file(s), source file name * Debug: Source DB connection details (except any secret credentials), source table name(s) * Debug: Format of the file, file size, number of records loaded * Info: Dataset hash / data version * Debug: Data/format conversion details, if required * Warning/Error: Any exception or errors if file/source table is missing or “failed to fetch / load data” messages |
| **Models** | * Debug: Location of the model asset * Info: Model metadata (version, name, hyperparameters, signature) * Debug: info about configuration files * Info: Metrics * Info: Predictions / model output location |
| **Pipeline Events** | * Debug: CPU / GPU usage * Debug: Memory utilization * Info: run-id * Info: Server or node details where jobs are running * Info: Start / end time of events * Error: Failed/exception/error messages (e.g. out of memory during data processing) |

From a technical point of view, we have to find a clever way how to store logs for analysis (presumably via an S3 bucket) and, especially important for ML Engineers / Ops, to stream logs (presumably via Kafka). From a monitoring point of view having the ability to analyze and compare logs, query logs and do some graphical representations is of central importance. Below you find an example to query all log messages of log level “error”.



In a more mature state of our monitoring system, we could also think about building dashboards on top of the log queries. In the ML chapter a log library component will be developed in the near future, which should be used in all retail and non-retail use cases.

## **Alerts**

For a stable deployment process even more important than logging is an **alerting system**, to make sure that relevant stakeholders are **informed** of issues on time. **Logging** facilitates then help to **identify the cause** of those problems. Here the basic requirement is that the ML Ops team is informed when a job has failed. In a higher maturity state of our monitoring framework automated JIRA ticket could be created whenever a problem with the deployment pipeline occurs.

Make sure that the following conditions are fulfilled when building up the alerting system:

* **Alerts Minimization:**
  + Restrict scope of alerts to those situations that **require intervention** and do have the real business impact
  + Write out only the important information in the notification
  + **Limit alerts audience** to people in charge of maintaining respective process steps, in our case the ML Ops team
* **Alerts Validity:**
  + **Test** your alerts **before** they go into production
  + **Add context** to the alert by including **descriptive information** of the problem
  + Establish within the ML Ops team a **troubleshooting framework** (steps) for going from the notification to action to troubleshooting: **characterize alerts** (i.e. whether they were false positives, negatives, or true positives), record and automate **frequently taken actions.**

1. **Minimum requirements**

Another important part of proper monitoring of deployed models is having its minimum requirements clearly defined and documented. These requirements will include frequency of monitoring and retraining, putting model in production, etc. Purpose of this section is to give benchmark which should be followed, meaning that these requirements should be at least fulfilled but can be exceeded as well.

## **Monitoring and retraining frequency**

All models **must** be:

* monitored at least **once per quarter**

and

* retrained at least **once a year** or more frequently, if proves necessary based on monitoring results and data availability.

In particular, the model should be retrained if any one of these conditions holds true:

* Significant change in business environment impacting the model performance immediately or in future
* Statistical KPIs are not fulfilled
* Business KPIs are not fulfilled
* Data drift is detected

Keeping in mind that it might be unfeasible or unreasonable to retrain a model in selected situations (e.g. business continuity, problems with data (privacy), etc.), it stays at **discretion of the use-case owner** to take the final decision on re-training or refraining from doing so.

* + 1. **Significant change in business environment**

If new events occur that the model under question has not been trained on, it might be necessary to trigger retraining or even re-modeling. In the retail demand forecasting domain, examples of such events might be an introduction of a new public holiday, a new promotional activity or weather irregularity (Schelter et al., 2018)[[13]](#footnote-14).

* + 1. **Statistical KPIs are not fulfilled**

In case when actual labels / targets are available at the time of monitoring (or with a delay):

* Model should be retrained when the agreed metric drops by more than 10% (in comparison to the value calculated during the last training/re-training)

In case actual labels are not observable:

* Models should be retrained when the distribution of predictions changes significantly. This can be measured using the Population Stability Index (PSI):
  + PSI <= 0.1 – distribution hasn’t changed (green)
  + 0.1 < PSI < 0.25 – slight change in distribution, close examination required (yellow)
  + PSI >= 0.25 – significant change, retraining should be triggered (red)
    1. **Business KPIs are not fulfilled**

In case when continuous random offer testing is in place:

* Retraining should be considered when the agreed KPI uplift in the modelling cohort[[14]](#footnote-15) drops by more than 10% from what was observed during A/B testing

In case no random offer testing in place:

* Retraining should be considered when the agreed KPI uplift compared with Global Control Group[[15]](#footnote-16) in the modelling cohort drops by more than 10% than what was observed during model testing or the first production run
  + 1. **Data drift is detected**

In case of detected data drift, the model should be retrained in an attempt to capture the change in data. Data drift could be detected using Population Stability Index (PSI) in a similar manner to other statistical KPIs:

* PSI <= 0.1 – distribution hasn’t changed (green)
* 0.1 < PSI < 0.25 – slight change in distribution, close examination required (yellow)
* PSI >= 0.25 – significant change, retraining should be triggered (red)

It is strongly recommended to use PSI. However, alternative, but less accurate/stable approach would be using other statistical tests (I.e., Kolmogorov-Smirnov, Chi-Squared, etc.) to detect changes in distribution of newly delivered data compared to previously available data. For each test used a general rule of thumb would be:

* p >= 0.05 – distribution hasn’t changed (green)
* 0.01 < p < 0.05 – slight change in distribution, close examination required (yellow)
* p <= 0.01 – significant change, retraining should be triggered (red)

1. **Appendix**

Classification tasks

**Table 7. KPIs for statistical quality of the model**

| **KPI** | **Definition** | **Reference to Documentation** |
| --- | --- | --- |
| F1 score |  | <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html> |
| Gini Coefficient | Gini = 2\*AUROC – 1[[16]](#footnote-17) | https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc\_auc\_score.html |
| Recall[[17]](#footnote-18) | where:  tp = number of “true” positives,  fn = number of “false” negavites | <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html> |
| Precision[[18]](#footnote-19) | where:  tp = number of “true” positives,  fp = number of “false” positives | <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html> |
| Precision-Recall AUC | Area under precision-recall curve | <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_curve.html> |
| Balanced Accuracy | where:  tn = number of “true” negatives  tp, fn, rp = as previously defined | <https://scikit-learn.org/stable/modules/model_evaluation.html#balanced-accuracy-score> |
| Mean Absolute Percentage Error | where:  y = true value  = predicted value  n = number of samples | <https://scikit-learn.org/stable/modules/model_evaluation.html#mean-absolute-percentage-error> |
| Root Mean Squared Error |  | [Root-mean-square deviation - Wikipedia](https://en.wikipedia.org/wiki/Root-mean-square_deviation)  <https://scikit-learn.org/stable/modules/model_evaluation.html#mean-squared-error> |
| R squared | where: | <https://scikit-learn.org/stable/modules/model_evaluation.html#r2-score> |

regression

**Table 8. KPIs for model business value**

| **KPI** | **Definition** | **Reference to Documentation** |
| --- | --- | --- |
| Conversion Rate | A conversion rate records the percentage of users who have completed a desired action | [What is a Conversion Rate and why is it important? | Adjust](https://www.adjust.com/glossary/conversion-rate/) |
| Weighted-Average Margin | Average amount that a group of products or services contribute to paying down the fixed costs of a business | [Weighted average contribution margin definition — AccountingTools](https://www.accountingtools.com/articles/what-is-the-weighted-average-contribution-margin.html#:~:text=The%20weighted%20average%20contribution%20margin%20is%20the%20average%20amount%20that,for%20various%20amounts%20of%20sales.) |
| Volume | Total amount of products sold to targeted customers | [Sales volume: Definition, formula, and how to increase it (zendesk.com)](https://www.zendesk.com/blog/sales-volume/#:~:text=Sales%20volume%20refers%20to%20the%20number%20of%20units%20your%20company,a%20growing%20or%20contracting%20company.) |
| GI per Communication | Gross income per contacted customer | [Profit Margin Is An Essential KPI To Monitor In Your Business (forbes.com)](https://www.forbes.com/sites/melissahouston/2021/12/22/profit-margin-is-an-essential-kpi-to-monitor-in-your-business/?sh=2ccfb3211fef) |
| OPEX Decrease per Customer | Decrease in operating expenses per targeted customer | [Operating Expense Definition (investopedia.com)](https://www.investopedia.com/terms/o/operating_expense.asp) |
| Customer Feedback | Initial feedback to the offer customer received – e.g. interested/not interested | [How to measure customer satisfaction KPIs - GetFeedback](https://www.getfeedback.com/resources/csat/how-to-measure-customer-satisfaction-kpis/) |
| Daily Active Users | Measurement of the number of users who are active on an app or website each day | [Daily Active Users (DAU) — TOP Agency](https://topagency.com/glossary/daily-active-users-dau-definition/) |
| Speed of Process | Time it takes for the client to go through the entire application process | / |
| Share of Clients with 2+ Products | Share of clients who have had any non bank-initiated transaction in last 6 months and a minimum of 2 products from the defined categories | [Retail Customer Metrics\_New Definitions\_final\_v4.pptx (sharepoint.com)](https://rbinternational.sharepoint.com/:p:/r/sites/ACA-RETCustomerLifetimeValue/_layouts/15/Doc.aspx?sourcedoc=%7B44622E52-CF2D-4DE8-8885-B773F7597557%7D&file=Retail%20Customer%20Metrics_New%20Definitions_final_v4.pptx&action=edit&mobileredirect=true&DefaultItemOpen=1) |
| Share of Active Clients | Share of clients who are active according to the current definition | [Retail Customer Metrics\_New Definitions\_final\_v4.pptx (sharepoint.com)](https://rbinternational.sharepoint.com/:p:/r/sites/ACA-RETCustomerLifetimeValue/_layouts/15/Doc.aspx?sourcedoc=%7B44622E52-CF2D-4DE8-8885-B773F7597557%7D&file=Retail%20Customer%20Metrics_New%20Definitions_final_v4.pptx&action=edit&mobileredirect=true&DefaultItemOpen=1) |

1. The topic of model risk is not explicitly addressed in this document. For more information, please, refer to the [Model Risk Management Framework](https://rbinternational.sharepoint.com/:b:/r/sites/ACA/AA%20Tribe/08%20Advanced%20Analytics%20Community/07%20Model%20Governance/Model%20Risk%20Management%20Framework%20v1.0.pdf?csf=1&web=1&e=35yZhk). [↑](#footnote-ref-2)
2. Adapted from the source: https://neptune.ai/blog/how-to-monitor-your-models-in-production-guide [↑](#footnote-ref-3)
3. Please, note that some classification algorithms produce a score rather than predicted label, and such score needs be further translated into propensity of belonging to a positive or a negative class (from 0 to 1). It therefore remains an arbitrary choice of the modeler to decide, which score / propensity threshold shall be used for defining final prediction labels based on predicted probabilities. F1-optimized threshold is recommended as a default choice. [↑](#footnote-ref-4)
4. A concept of data integrity is a similar to that one of data quality. The latter naming is, however, avoided not to avoid confusion with a more narrow notion of data quality / DQIs used in RBI Group. [↑](#footnote-ref-5)
5. **Red** color indicates that corresponding issue, unless handled explicitly within the pipeline, would cause an interruption / break of the pipeline at an earlier or later stage. **Orange** color, in turn, means that the pipeline does not break technically, runs through, but will produce an incorrect output. If both situations are theoretically possible for a given underlying problem, the table will contain red and orange colors together. [↑](#footnote-ref-6)
6. For this table the meaning of criticality colors is the following: **Orange** – a non-critical feature drift is detected and logged, no notification is triggered, **Red** – feature drift is material and needs to be further investigated, send alert. [↑](#footnote-ref-7)
7. As an example, it is very likely that datasets 1) used for model development and 2) the one to which the model is applied, are **not comparable by design**, when only a part of the “application” sample is used to train an analytical model. For instance, a model providing a credit score for all customers with active transactional behavior is usually trained only a small sub-set of this population (i.e.. customers who ever took a credit product). This makes a direct comparison of the two samples meaningless for the monitoring purposes since by design they are not comparable. [↑](#footnote-ref-8)
8. <https://www.researchgate.net/publication/321627304_Online_Ensemble_Learning_with_Abstaining_Classifiers_for_Drifting_and_Noisy_Data_Streams> [↑](#footnote-ref-9)
9. Source: https://neptune.ai/blog/how-to-monitor-your-models-in-production-guide [↑](#footnote-ref-10)
10. Source: https://arize.com/blog/monitor-your-model-in-production/ [↑](#footnote-ref-11)
11. Not relevant for batch processing [↑](#footnote-ref-12)
12. DEBUG — fine-grained informational events that are most useful to troubleshoot a pipeline.

    INFO —informational messages that highlight the progress of the pipeline at a coarse-grained level.

    WARN —information on potentially harmful situations; including other run-time situations that are undesirable or unexpected, but not necessarily "wrong".

    ERROR —other run-time errors or unexpected conditions such as error events that might still allow the pipeline to continue running.

    CRITICAL / FATAL - Any error that is forcing a shutdown of the service or application to prevent data loss.

    The order of levels by severity is the following: DEBUG < INFO < WARN < ERROR < CRITICAL. [↑](#footnote-ref-13)
13. [on-challenges-in-machine-learning-model-management.pdf (amazon.science)](https://assets.amazon.science/7d/38/968b82c745bd9859a79dab0aade8/on-challenges-in-machine-learning-model-management.pdf) [↑](#footnote-ref-14)
14. Modeling cohort – subset of original data used for modeling [↑](#footnote-ref-15)
15. Global Control Group (GCG) – refers to subset of customers who will not receive any of the active campaigns [↑](#footnote-ref-16)
16. An exact specification of approximation algorithm, which stands behind sklearn implementation, goes beyond the scope of current document. [↑](#footnote-ref-17)
17. This metric will not be monitored stand-alone. Corresponding formula is given for the purpose of further specifying other KPIs. [↑](#footnote-ref-18)
18. This metric will not be monitored stand-alone. Corresponding formula is given for the purpose of further specifying other KPIs. [↑](#footnote-ref-19)