



Brain-Based Wheelchair Control

AA – 2024/2025

Software System Engineering

Alessandro Ascani

Francesco Taverna

Giovanni Ligato

Gabriele Pianigiani

Saverio Mosti

Table of contents

1	Process Landscape.....	4
2	Processes.....	6
2.1	Configure Service Process.....	6
2.2	Prepare Session Process	7
2.3	Generate Learning Sets Process	8
2.4	Develop Classifier Process	9
2.5	Classify Session Process.....	10
2.6	Evaluate Classifier Performance Process.....	10
3	– Use-Case Diagrams.....	11
3.1	Segregation System	11
3.2	Development System.....	11
3.3	Evaluation System	11
3.4	Preparation System	12
3.5	Ingestion System.....	12
3.6	Production System.....	12
4	– Use-Case Mock-ups.....	13
4.1	Check Data Balancing	13
4.2	Check Coverage (<i>Check Input Coverage</i>)	14
4.3	Set #iterations.....	14
4.4	Check Learning Plot	15
4.5	Check Validation Results.....	16
4.6	Check Test Results	17
4.7	Configure Evaluation System.....	17
4.8	Evaluate Classifier	18
4.9	Configure Segregation System	19
4.10	Configure Ingestion System.....	20
4.11	Configure Preparation System.....	20
4.12	Configure Development System.....	21
4.13	Configure Production System	22
5	– Data and Application Logic Modeling	23
5.1	Prepare Session.....	23
5.2	Generate Learning Set	26
5.3	Develop Classifier	27
5.4	Classify Session.....	29
5.5	Evaluate Classifier Performance	30
6	– Design.....	32
6.1	Ingestion System	32

6.2 Preparation System.....	33
6.3 Segregation System.....	35
6.4 Develop Classifier	37
6.5 Evaluate Classifier Performance	39
6.6 Production System	41
7. Testing.....	42
7.1 NON-INTEROPERABILITY	42
7.2 NON-RESILIENCY	43
7.3 NON-AUTOMATION	46
7.3 AUTOMATION RESPONSIVENESS-ELASTICITY of the Development phase	55
7.4 RESPONSIVENESS-ELASTICITY of the Production phase	59

1 Process Landscape

[Everyone]

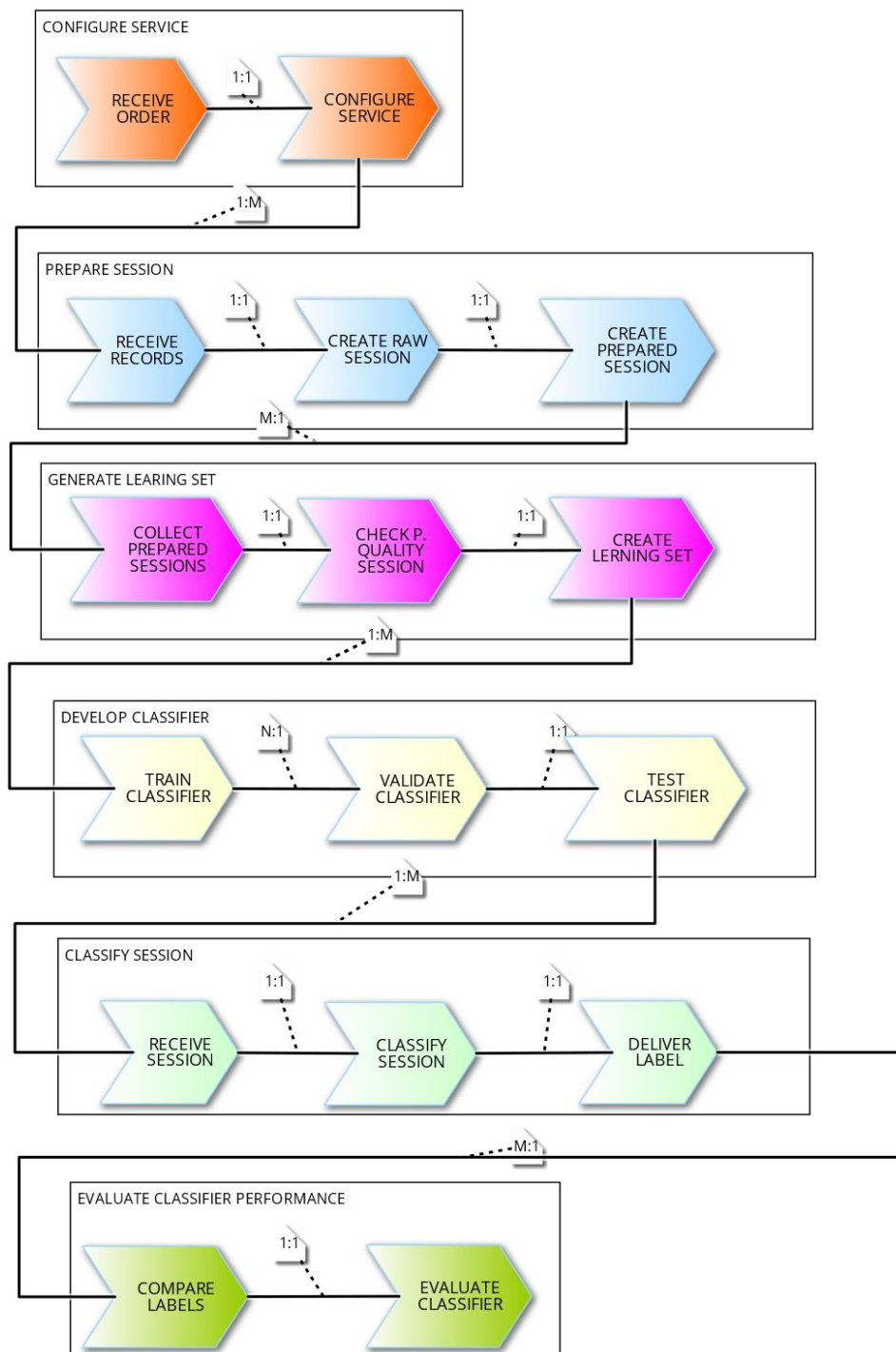


Figure 1 - Process Landscape Diagram.

In the general task of drawing the process landscape, we wanted to put attention on a particular cardinality between the following activities: “TRAIN CLASSIFIER” and “VALIDATE CLASSIFIER”.

These activities follow inside the “DEVELOP CLASSIFIER” sub-process. The cardinality between the two is N:1. We can note that there is a change of cardinality even if we are inside a sub-process because that one is a technical change in cardinality and not a business one.

We must train the classifier several times and after this process validate it by just choosing the best obtained instance regarding the hyper-parameters that we need to optimize.

2 Processes

2.1 Configure Service Process

[Everyone]

The configuration is sent to a person, and subsequently to the messaging system, which is not an actual system but rather a message on Teams from someone who observed that the classifier is not performing correctly. The individual responsible for the configuration may need to adjust the number of labels from 3 to 4, or vice versa. After making the necessary changes, the system must be powered down and restarted.

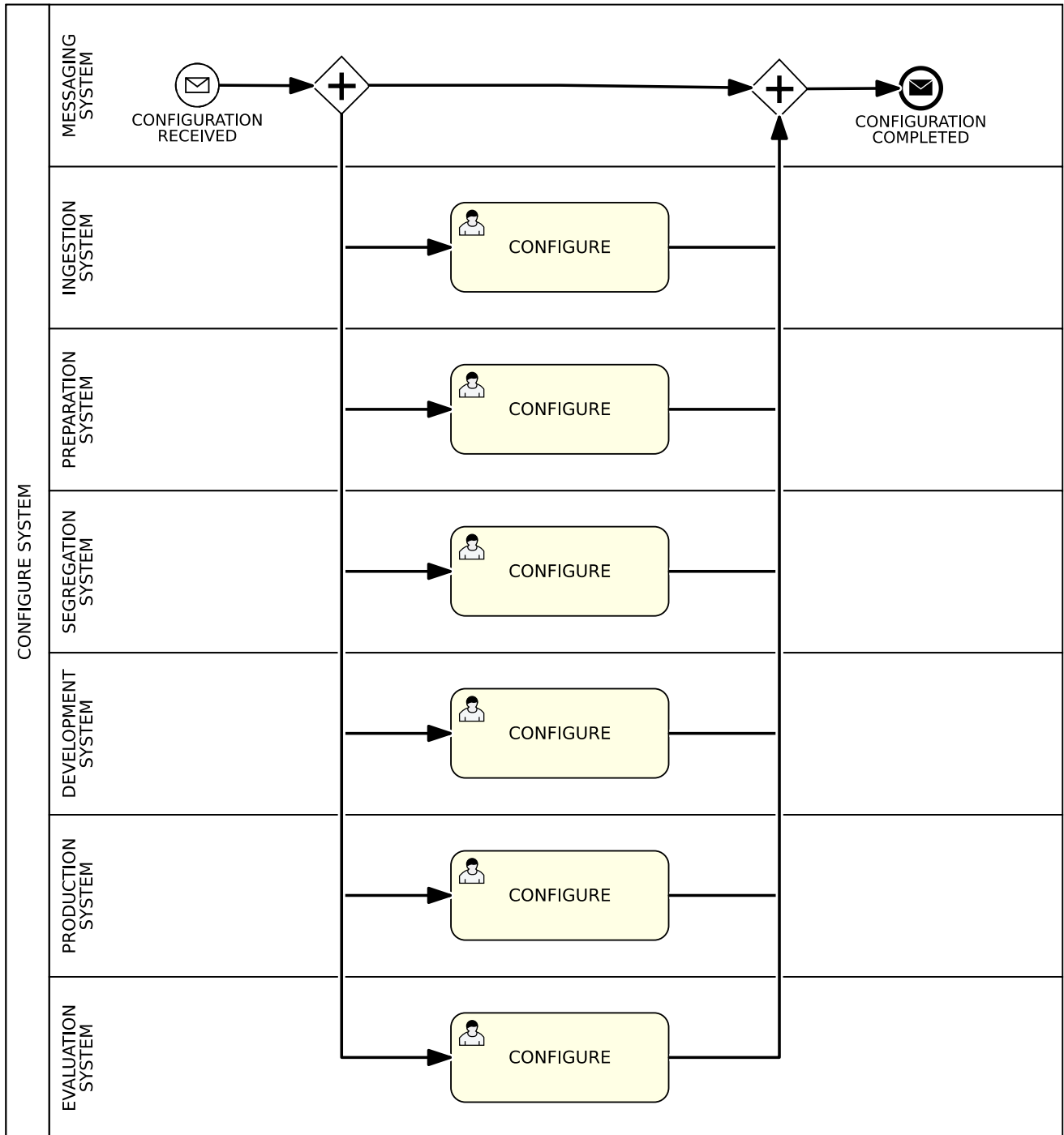


Figure 2 - Configure Systems Process BPMN.

2.2 Prepare Session Process

[Francesco Taverna]

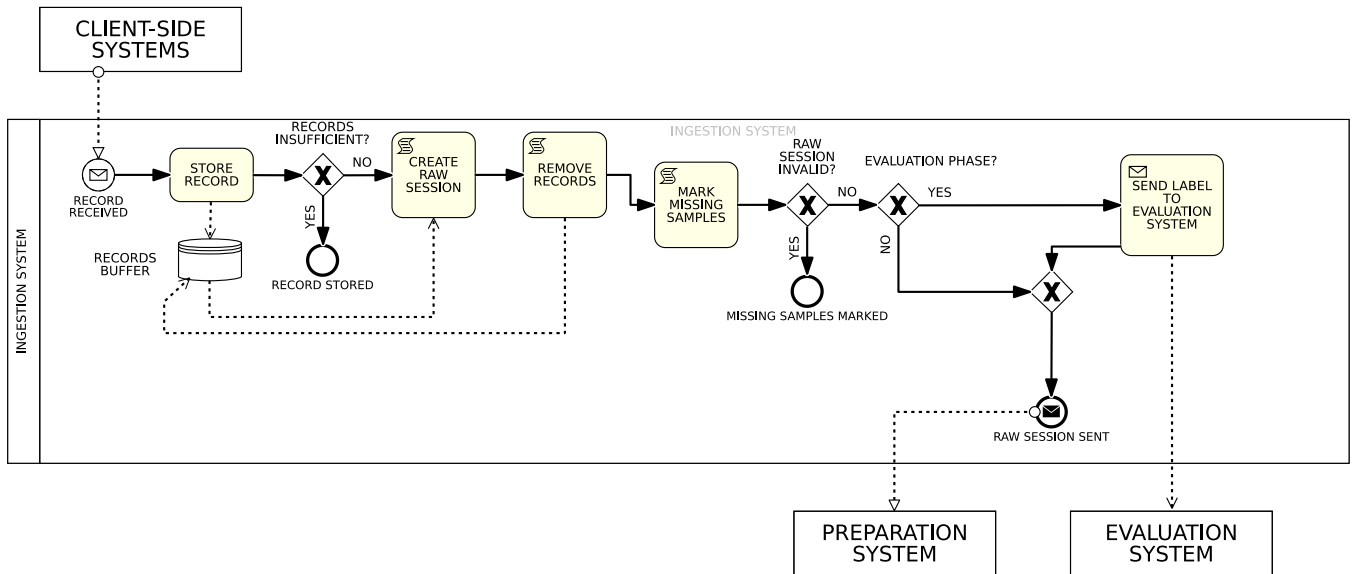


Figure 3- Prepare Session Process BPMN (ingestion system side).

The helmet record is a time series, consisting of a series of samples (numerical values) obtained by sampling a specific feature (e.g., heart signal) within a defined time interval (e.g., 0.5 seconds). A series of records—specifically four (three during the production phase, as labels are not included, and four during the development and evaluation phase, where the label is provided by an expert)—form a raw session.

Once a raw session is created, the records are removed from the database to avoid overloading it, as retaining them is no longer necessary.

Missing samples within a record are marked with a placeholder, and a certain number of missing samples is acceptable. However, if the helmet record contains too many missing samples and does not meet the required number of samples, the helmet record is deemed invalid, and the corresponding raw session is discarded (raw session invalid).

In the evaluation phase, if a raw session is created, the label is sent to the evaluation system, which compares it with the label provided by the expert.

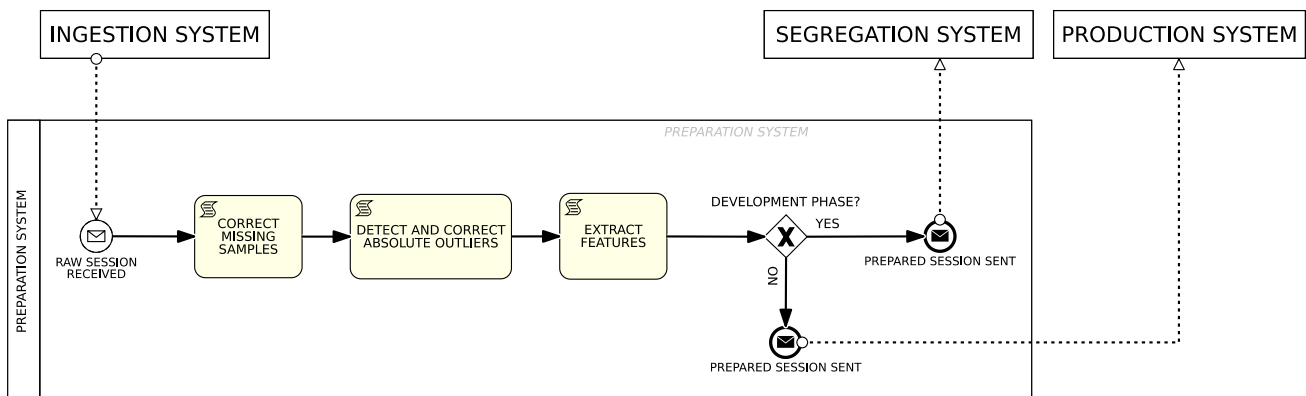


Figure 4 - Prepare Session Process BPMN (preparation system side).

In the **Correct Missing Samples** phase, all missing samples (those marked with the placeholder) in each record that constitutes the raw session are corrected through interpolation.

The **Detect and Correct Absolute Outliers** phase involves bringing out-of-range values back into the acceptable range of values (e.g., within a specific range of volts for an electrical signal). This is achieved by setting outlier values to the closest acceptable limit (e.g., min or max values).

In the **Extract Feature** phase, the raw session is processed in such a way that it retains the **activity** and **environment** features, along with the label (if available). At the same time, the **EEG Data** section is processed to extract four specific features, which correspond to the four brain signals that are being considered.

2.3 Generate Learning Sets Process

[Saverio Mosti]

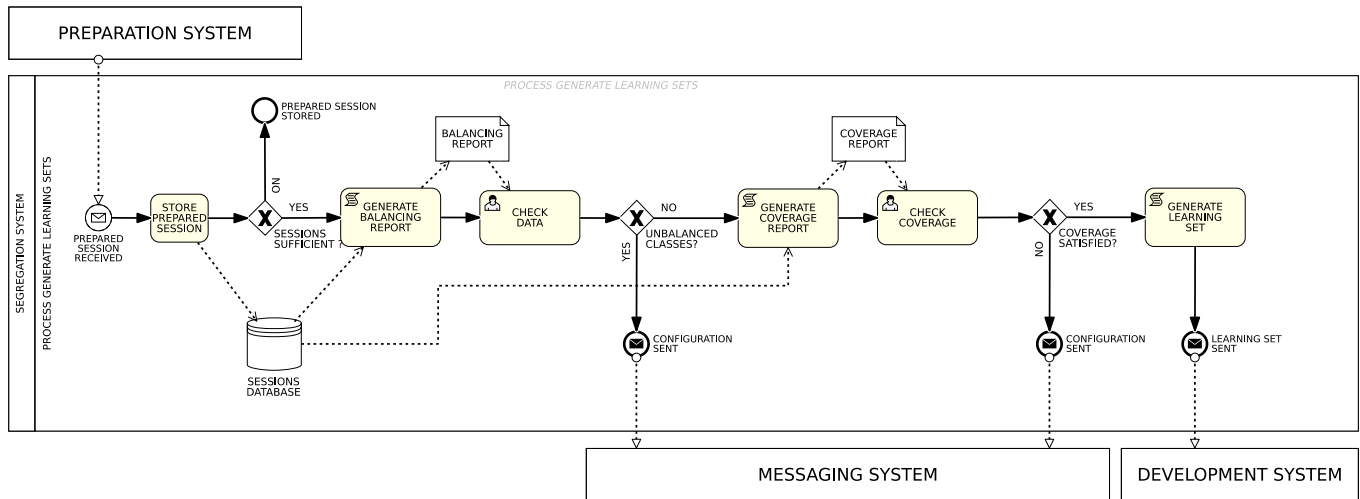


Figure 5 - Generate learning sets process BPMN

The segregation system does not stop if the number of sessions is insufficient; instead, it immediately returns to a waiting state for a new message.

If either of the two user checks (Check Data and Check Coverage) fails, the data collected so far is not discarded. Instead, it is retained in the hope that it will be corrected by the new sessions that will be received from the preparation system.

2.4 Develop Classifier Process

[Gabriele Pianigiani]

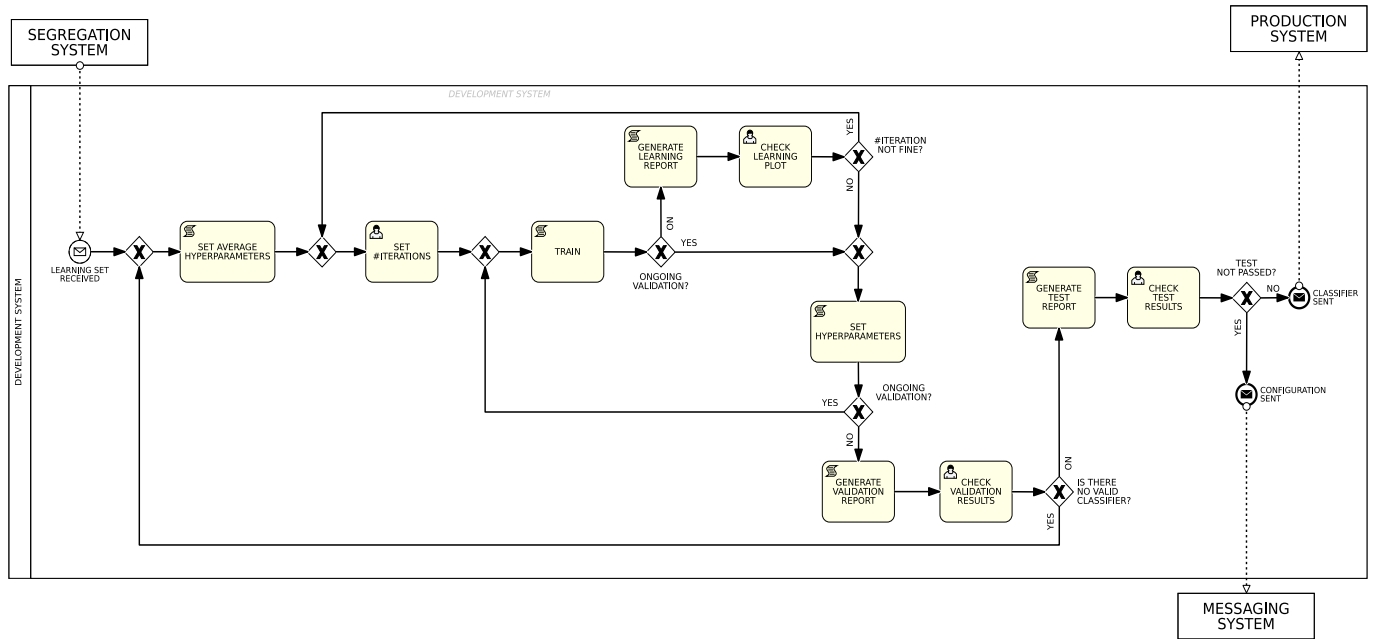


Figure 6 - Develop Classifier Process BPMN (development system).

[Alessandro Ascani]

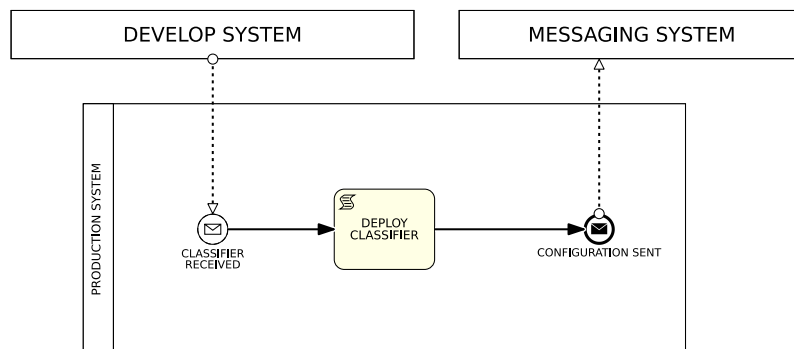


Figure 7 - Develop Classifier Process BPMN (production system).

2.5 Classify Session Process

[Alessandro Ascani]

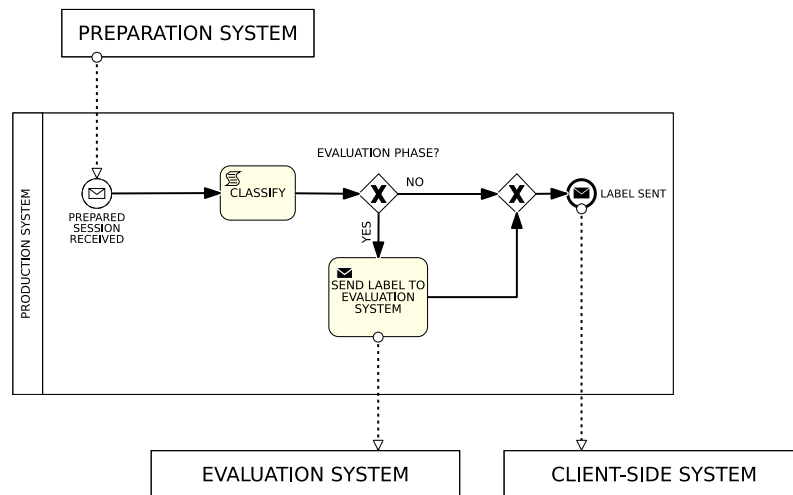


Figure 8 - Classify Session Process BPMN.

2.6 Evaluate Classifier Performance Process

[Giovanni Ligato]

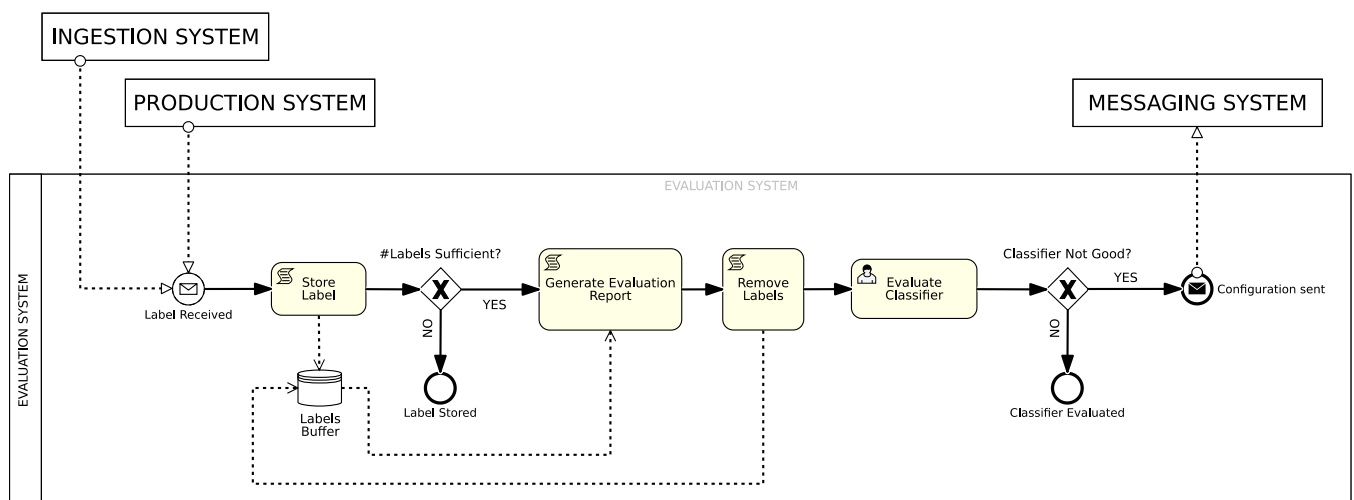


Figure 9 - Evaluate Classifier Performance Process BPMN.

3 – Use-Case Diagrams

3.1 Segregation System

[Saverio Mosti]

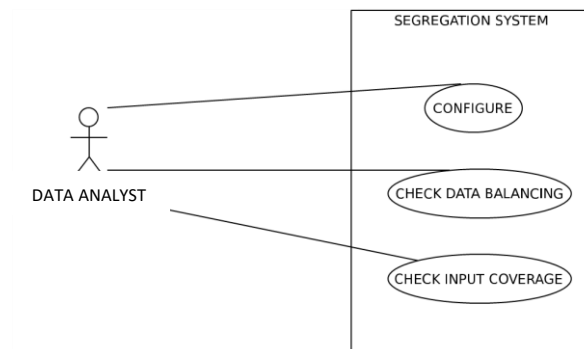


Figure 10 - Segregation System Use-Case Diagram.

In reality the HUMAN OPERATOR is a data analyst.

3.2 Development System

[Gabriele Pianigiani]

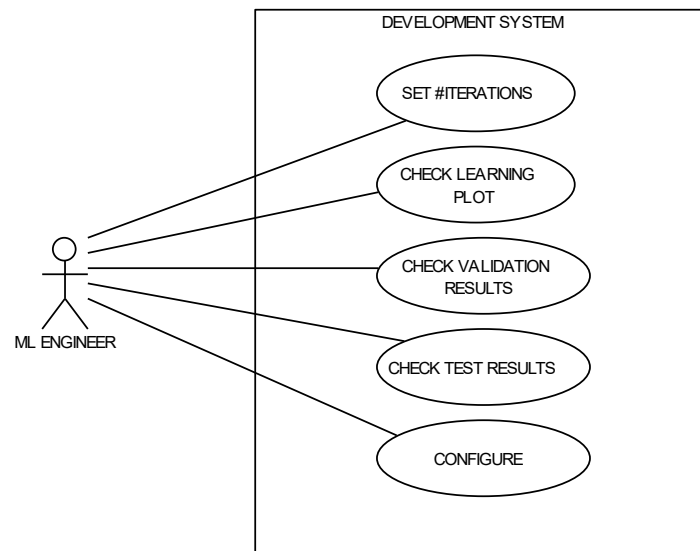


Figure 11 - Development System Use-Case Diagram.

3.3 Evaluation System

[Giovanni Ligato]

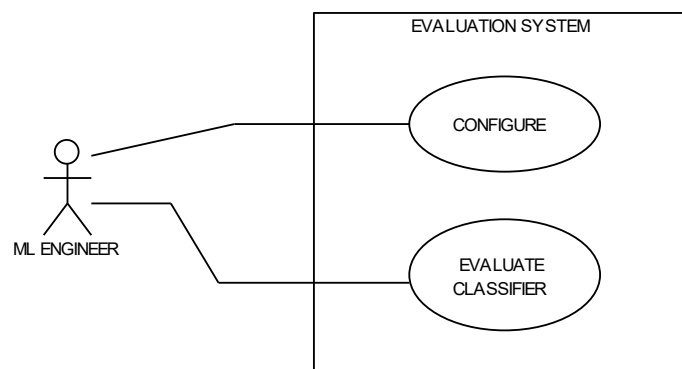


Figure 12 - Evaluation System Use-Case Diagram.

3.4 Preparation System

[Francesco Taverna]

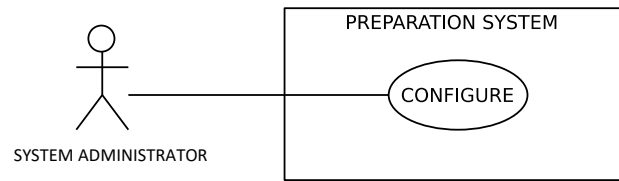


Figure 13 - Preparation System Use-Case Diagram.

3.5 Ingestion System

[Alessandro Ascani]

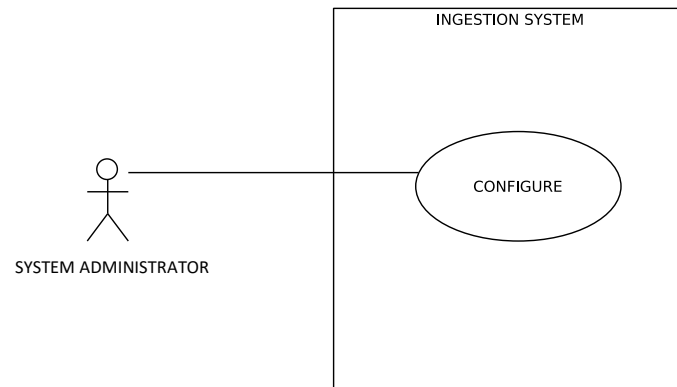


Figure 14 - Ingestion System Use-Case Diagram.

3.6 Production System

[Francesco Taverna]

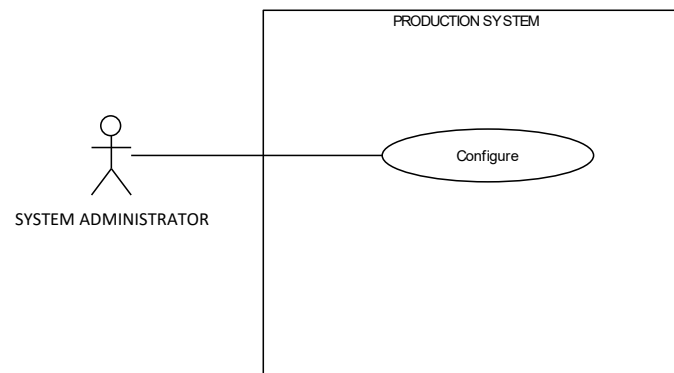


Figure 15 - Production System Use-Case Diagram.

4 – Use-Case Mock-ups

4.1 Check Data Balancing

[Saverio Mosti]

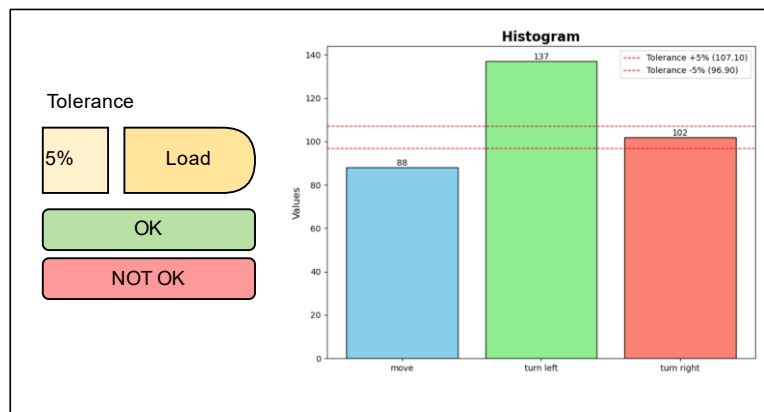







Figure 16 - Check Data Balancing Mock-up.

1. The use case starts when the  Data analyst launches the application.
2. SYSTEM loads the Check Data Balancing interface.
3. The  Data analyst presses the "LOAD" button.
4. for each class
4.1. if the bar level is lower or greater of the tolerance interval:
4.1.1. The  Data analyst presses the "NOT OK" button.
4.1.2. The  Data analyst requests a new configuration.
4.1.3. <i>The use case terminates.</i>
end if
5. end for each
6. The  Data analyst presses the "OK" button.

Pre-Condition	The data distribution report has been generated.
Pre-Condition	The application has saved the related plot image.

4.2 Check Coverage (*Check Input Coverage*)

[Saverio Mosti]

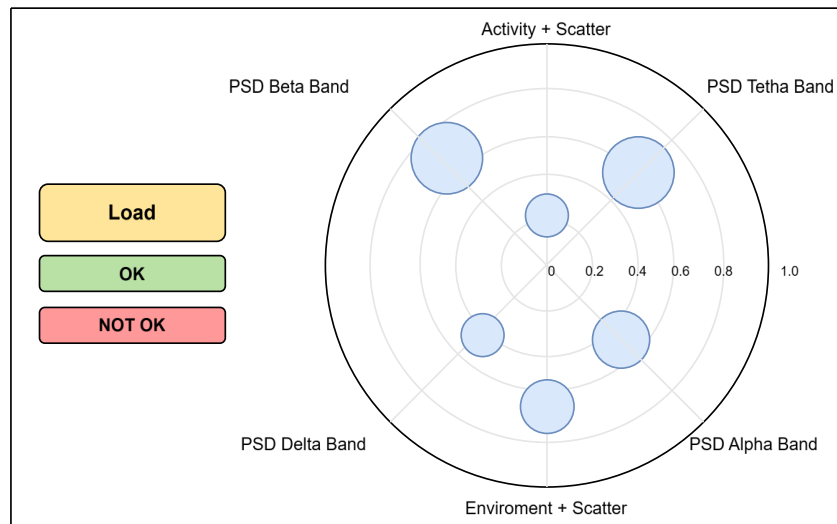







Figure 17 - Check Input Coverage Mock-up.

1. The use case starts when the  Data analyst launches the application.
2. SYSTEM loads the Check Input Coverage interface.
3. The  Data analyst presses the "LOAD" button.
4. for each feature
4.1. if the distribution of the values do not respect the expected distribution for the feature under examination. Multi-criteria for the PSD features (<i>Activity? Environment? Age of the customer?</i>).
4.1.1. The  Data analyst presses the "NOT OK" button.
4.1.2. The  Data analyst requests a new configuration.
4.1.3. <i>The use case terminates.</i>
end if
5. end for each
6. The  Data analyst presses the "OK" button.

Pre-condition	The coverage report has been generated.
Pre-Condition	The application has saved the related plot information.
Post-Condition	The learning sets are generated.




4.3 Set #iterations

[Gabriele Pianigiani]

Iterations settings

Insert number of iterations:

Figure 18 - Set #Iterations Mock-Up.

1. The use case starts when the  **ML ENGINEER** opens the interface for setting the number of iterations.
2. The  **ML ENGINEER** digits the number of iterations in the “Insert number of iterations” field.
3. The  **ML ENGINEER** presses the "Apply" button.









Post-Condition	The development system updates the number of iterations.
-----------------------	--

4.4 Check Learning Plot

[Gabriele Pianigiani]



Figure 19 - Check Learning Plot Mock-Up.

1. The use case starts when the  **ML ENGINEER** clicks on "Check Learning Plot"
2. **SYSTEM** shows a window with the Training error plot (MSE) against the number of iterations.
3. **if** the  **ML ENGINEER** sees that the line in the plot is flat for at least half of the iterations **then**
 - 3.1. The  **ML ENGINEER** reduces the number of iterations accordingly via the field and the button in the window
 - 3.2. The  **ML ENGINEER** press the “TRAIN AGAIN” button
 - 3.3. **SYSTEM** updates the plot with the new number of iterations
4. **else if** the  **ML ENGINEER** sees that the line in the plot is decreasing till the end
 - 4.1. The  **ML ENGINEER** increases the number of iterations accordingly via the field and the button in the window.
 - 4.2. The  **ML ENGINEER** press the “TRAIN AGAIN” button
 - 4.3. **SYSTEM** updates the plot with the new number of iterations
5. **else if** the number of iterations is fine
 - 5.1. The  **ML ENGINEER** presses the “VALIDATE” button.
6. **end if**

Pre-condition	The training has been completed.
Pre-condition	The training error plot has been generated.
Post-condition	The number of iterations has been updated.
Post-condition	The plot has been updated.

4.5 Check Validation Results

[Alessandro Ascani, Gabriele Pianigiani]

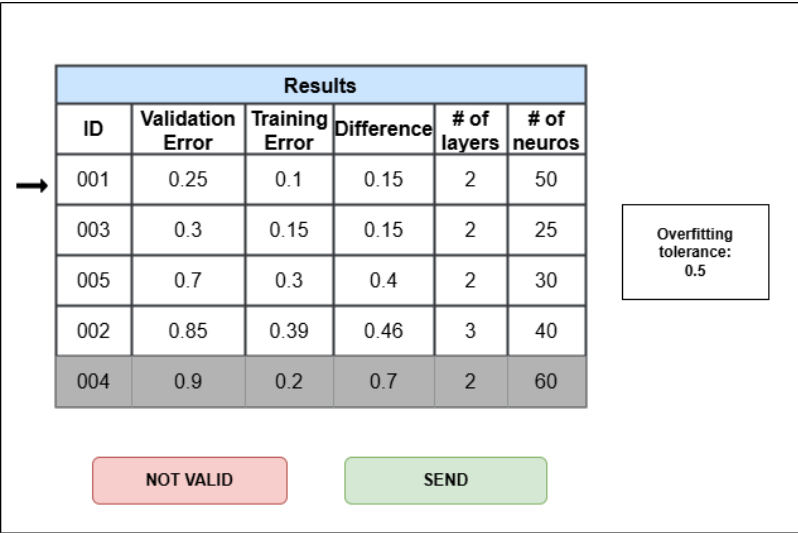


Figure 20 - Check Validation Results Mock-Up.

1. The use case starts when the SYSTEM visualizes the table results containing the top 5 classifier.
2. for each classifier in the report.
2.1. The ML ENGINEER checks if the difference is under the overfitting tolerance.
3. end for each
4. if the ML ENGINEER sees that there are no classifiers that satisfy the tolerance.
4.1. The ML ENGINEER presses the "NOT VALID" button.
5. else
5.1. The ML ENGINEER selects the best classifier in the report based on the value of the difference, number of layers, number of neurons.
5.2. The ML ENGINEER clicks on "SEND" button.
6. end if

Validation report: carry out a grid search, by training different networks, one for each point of the grid space. Finally, show a table with the top 5 classifiers, ordered by the final MSE on the validation set. In the table, for each classifier, show its validation error, training error, no. of layers, no. of neurons.

Find the best network, i.e., the lowest validation error, for which the difference between validation and training error is under the overfitting tolerance. Find the second- 2 best network. If the validation error between the two network is very similar, i.e. the difference is one order of magnitude w.r.t. their error, then select the network with the lowest complexity.

Pre-condition	The validation report has been generated.
Post-condition (flow 1)	The best classifier has been selected.
Post-condition (flow 2)	A new training started.





4.6 Check Test Results

[Gabriele Pianigiani]

Generalization Tolerance	0.19
Validation Error	0.61
Test Error	0.84
Test Error - Validation Error	0.23

TEST PASSED?

Figure 21 - Check Test Results Mock-Up.

1. The use case starts when the  ML ENGINEER opens the interface to check test results.
2. SYSTEM shows the check test result interface.
3. The  ML ENGINEER sees the color of the value in the row "Test Error - Validation Error".
4. if the color is red.
4.1 The  ML ENGINEER clicks on "NO" button.
5. else
5.1 The  ML ENGINEER clicks on "YES" button.
6. end if


Pre-Condition	The development system has generated the test report with the winner network.
Post-Condition	The winner classifier has been tested.

4.7 Configure Evaluation System


[Giovanni Ligato]

CONFIGURE [EVALUATION SYSTEM]


MINIMUM #LABELS

100 

TOTAL ERRORS


30 






MAX CONSECUTIVE ERRORS

1 

APPLY

Figure 22 - Configure Evaluation System Mock-Up.

1. The use case starts when the  ML ENGINEER opens the CONFIGURE EVALUATION SYSTEM interface.
--

2. **SYSTEM** displays the configuration page to the  **ML ENGINEER**.
3. The  **ML ENGINEER** sets the **MINIMUM #LABELS** field.
4. The  **ML ENGINEER** sets the **TOTAL ERRORS** field.
5. The  **ML ENGINEER** sets the **MAX CONSECUTIVE ERRORS** field.
6. The  **ML ENGINEER** presses the **APPLY** button to save the new configuration.

Post-Condition	Evaluation System Configured.
-----------------------	-------------------------------

4.8 Evaluate Classifier

[Giovanni Ligato]

EVALUATION REPORT

TOTAL ERRORS

THRESHOLDS

MAX CONSECUTIVE ERRORS

#SESSION	CLASSIFIER	EXPERT
1	move	move
2	move	turn left
3	turn right	turn right
...
100	turn left	turn left

AUTOMATIC CHECKS




ACTUAL # TOTAL ERRORS: 22 OK

ACTUAL # MAX CONSECUTIVE ERRORS: 1 OK

NOT GOOD

GOOD

Figure 23 - Evaluate Classifier Mock-Up.

1. The use case starts when the  **ML ENGINEER** opens the **EVALUATE CLASSIFIER** interface.
2. **SYSTEM** displays a table to the  **ML ENGINEER**, showing classifier and expert labels for each session to evaluate.
3.  **ML ENGINEER** checks if **ACTUAL # TOTAL ERRORS** and **ACTUAL # MAX CONSECUTIVE ERRORS** in the **AUTOMATIC CHECKS** session are labelled with **OK**.
 - 3.1. if both automatic checks are labelled with **OK**:

3.1.1.  **ML ENGINEER** clicks on the **GOOD** button.

3.2. **else**

3.2.1.  **ML ENGINEER** clicks on the **NOT GOOD** button.

Pre-Condition	Sufficient #Labels received (<i>Evaluation Report has been generated</i>).
Post-Condition	The classifier has been evaluated.

4.9 Configure Segregation System

[Saverio Mosti]

CONFIGURE SEGREGATION SYSTEM

Insert #sessions to collect:

100

Select balancing tolerance interval:

20

%

Select training set percentage:

75

%







Select validation set percentage:

15

%

Apply

Figure 24 - Configure Segregation System Mock-Up.

1. The use case starts when the  **Data analyst** opens the interface for configuring parameters.
2. The  **Data analyst** digits the number of sessions to collect in the “Insert #sessions to collect” field.
3. The  **Data analyst** digits the balancing tolerance interval in the “Select balancing tolerance interval” field.
4. The  **Data analyst** digits the training set percentage in the “Select training set percentage” field.
5. The  **Data analyst** digits the validation set percentage in the “Select validation set percentage” field.
6. The  **Data analyst** presses the "Apply" button.

Post-Condition	The segregation system updates the Segregation System configuration.
-----------------------	--

4.10 Configure Ingestion System

[Francesco Taverna]

Configuration parameters

Select current phase

development

production

evaluation

Select missing samples thresholdinterval

4

Select Production Sessions

5000

Select Evaluation Sessions

50




Service

True

False

Apply

Figure 25-Configure Ingestion System Mock-up

- | |
|---|
| 1. The use case starts when the  SYSTEM ADMINISTRATOR opens the interface to configure the ingestion system |
| 2. for each parameter: |
| 2.1 The  SYSTEM ADMINISTRATOR configures the parameter. |
| 3. end for each |
| 4. The  SYSTEM ADMINISTRATOR clicks on "Apply" button. |

Post-Condition	The Ingestion system updates the configuration parameters.
-----------------------	--

4.11 Configure Preparation System

[Francesco Taverna]

Configuration parameters




Development

Yes

No

Apply

Figure 26 - Configure Preparation System Mock-Up.










- | |
|---|
| 1. The use case starts when the  SYSTEM ADMINISTRATOR opens the interface for configuring parameters. |
| 2. The  SYSTEM ADMINISTRATOR selects the phase. |
| 3. The  SYSTEM ADMINISTRATOR presses the "Apply" button. |

4.12 Configure Development System

[Gabriele Pianigiani]

Configuration parameters	
Insert Overfitting Tolerance:	<input type="text" value="0.2"/>
Insert Min # of Layers:	<input type="text" value="12"/>
Insert Max # of Layers:	<input type="text" value="32"/>
Insert Variation step Layers:	<input type="text" value="4"/>
Insert Min # of Neurons:	<input type="text" value="4"/>
Insert Max # of Neurons:	<input type="text" value="20"/>
Insert Variation step Neurons:	<input type="text" value="4"/>
Insert Generalization tolerance:	<input type="text" value="0.2"/>
<input type="button" value="Apply"/>	

Figure 27 - Configure Development System Mock-Up.

1. The use case starts when the  **ML ENGINEER** opens the interface to configure the development system.
2. **SYSTEM** shows Development System configuration parameter interface
2. The  **ML ENGINEER** digits the Overfitting tolerance in the “Insert Overfitting tolerance” field.
3. The  **ML ENGINEER** digits the Min # of Layers in the “Insert Min # of Layers” field.
4. The  **ML ENGINEER** digits the Max # of Layers in the “Insert Max # of Layers” field.
5. The  **ML ENGINEER** digits the Variation step Layers in the “Insert Variation step Layers” field.
6. The  **ML ENGINEER** digits the Min # of Neurons in the “Insert Min # of Neurons” field.
7. The  **ML ENGINEER** digits the Max # of Neurons in the “Insert Max # of Neurons” field.
8. The  **ML ENGINEER** digits the Variation step Neurons in the “Insert Variation step Neurons” field.
9. The  **ML ENGINEER** digits the Generalization tolerance in the “Select Generalization tolerance” field.

10. The  **ML ENGINEER** presses the "Apply" button.

Post-Condition	The configuration of the development system has been updated.
-----------------------	---

4.13 Configure Production System

[Alessandro Ascani]

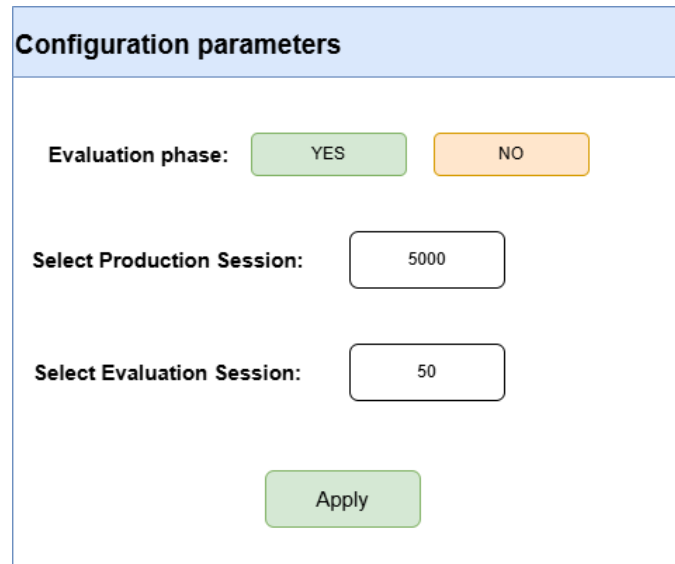





Figure 28 - Configure Production System Mock-up

- | |
|---|
| 1. The use case starts when the  SYSTEM ADMINISTRATOR opens the interface for configuring parameters. |
| 2. SYSTEM shows the Production System configuration parameters interface |
| 3. for each parameter: |
| 3.1 The  SYSTEM ADMINISTRATOR configures the parameter. |
| end for each |
| 4. The  SYSTEM ADMINISTRATOR presses the "Apply" button. |

Post-Condition	The configuration of the production system has been updated.
-----------------------	--

5 – Data and Application Logic Modeling

The parameters are manually configured by users in JSON files. These JSON files are then read when the application is launched, and the specific loadparameter function is executed.

It is not necessary to create a sequence diagram for this configuration process, as it simply involves the human user writing in the text file using a text editor.

5.1 Prepare Session

[Francesco Taverna]

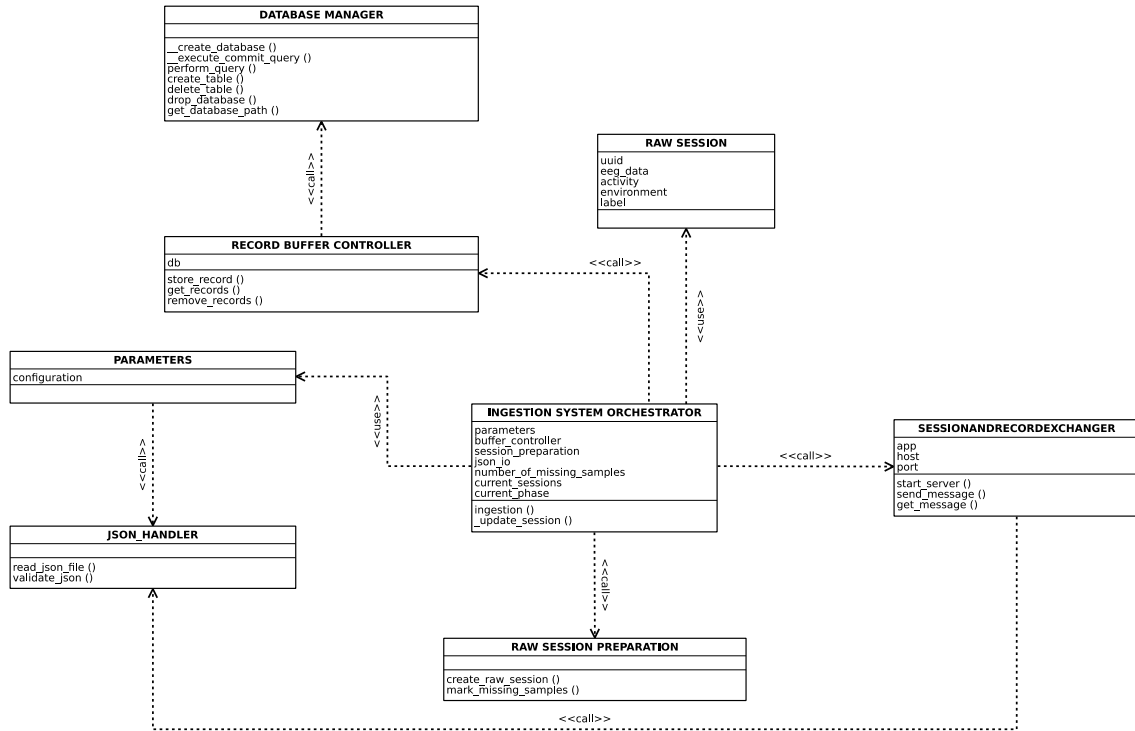


Figure 29 - UML Class Diagram of Prepare Session Process (Ingestion System).



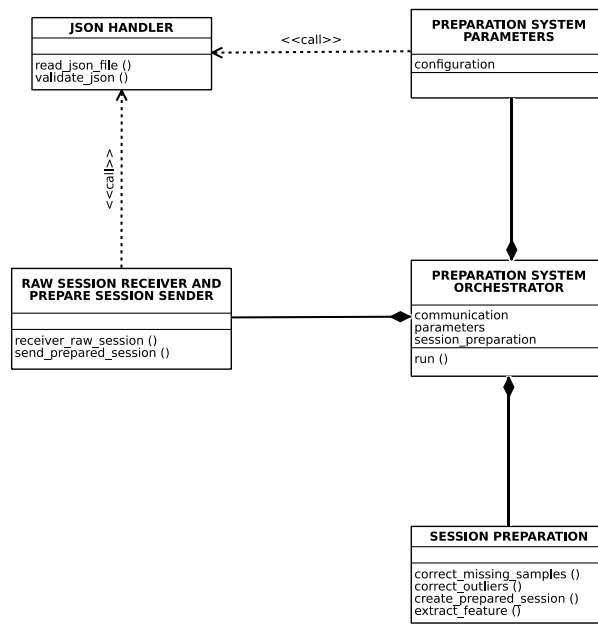


Figure 31 - UML Class Diagram of Prepare Session Process (Preparation System).

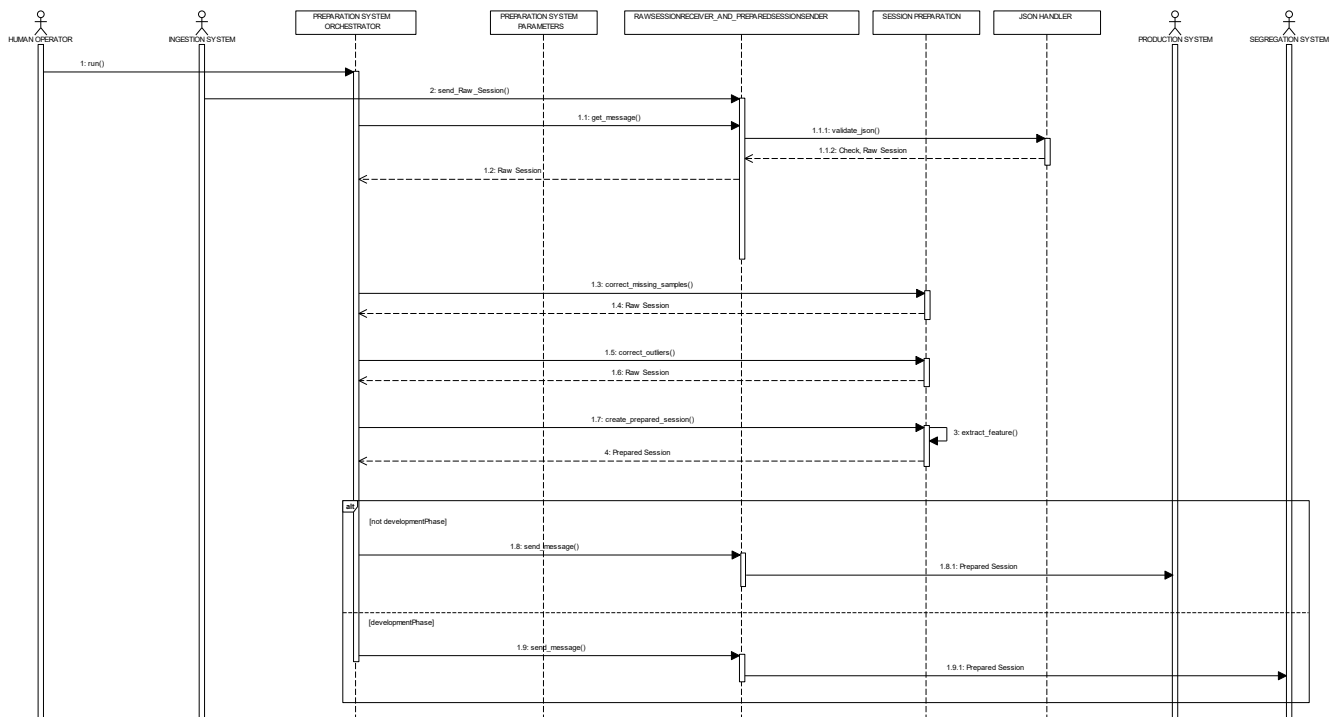


Figure 32 - Sequence Diagram of Prepare Session Process (Preparation System).

[Saverio Mosti]



5.3 Develop Classifier

[Gabriele Pianigiani]

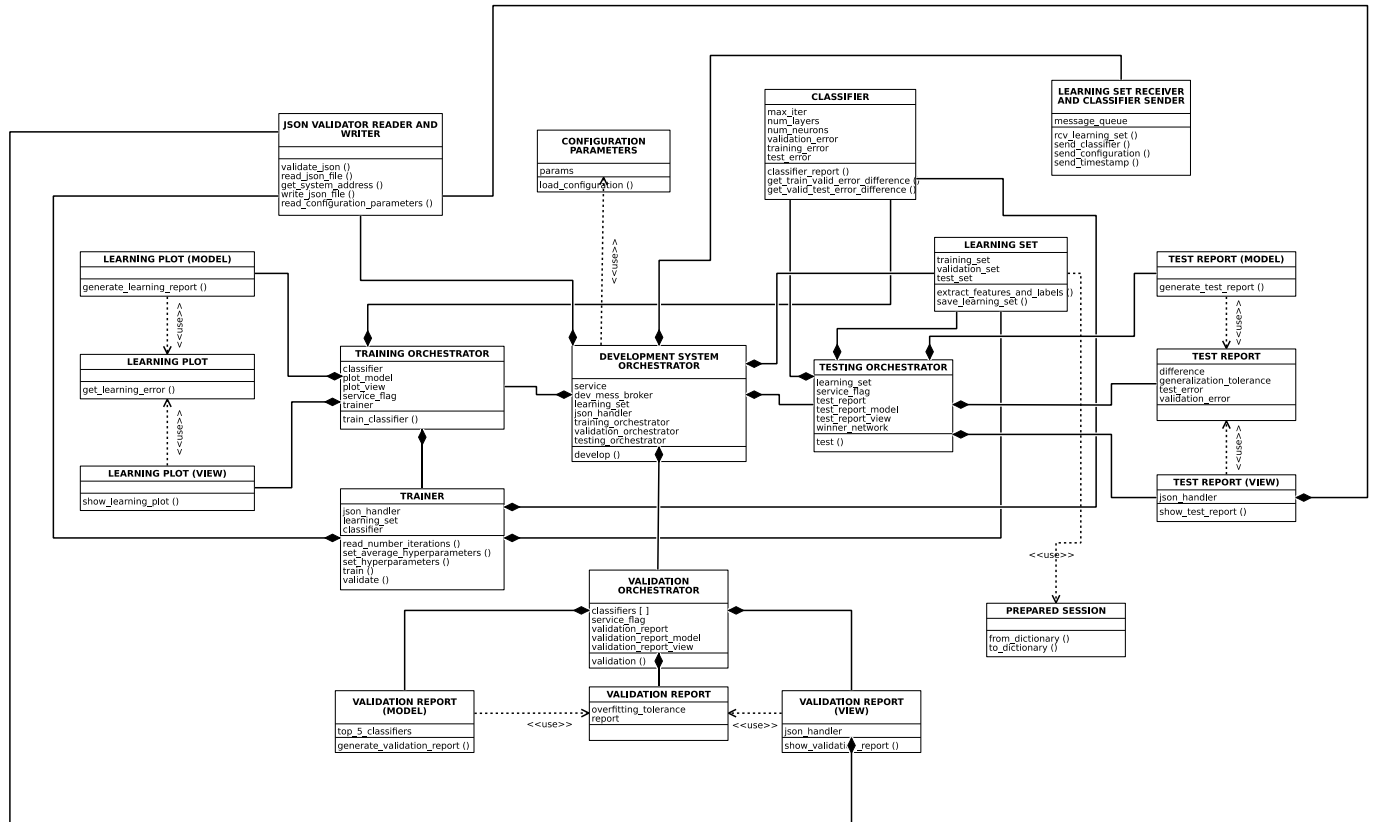


Figure 35- UML Class Diagram of the Develop Classifier Process

5.4 Classify Session

[Alessandro Ascani]

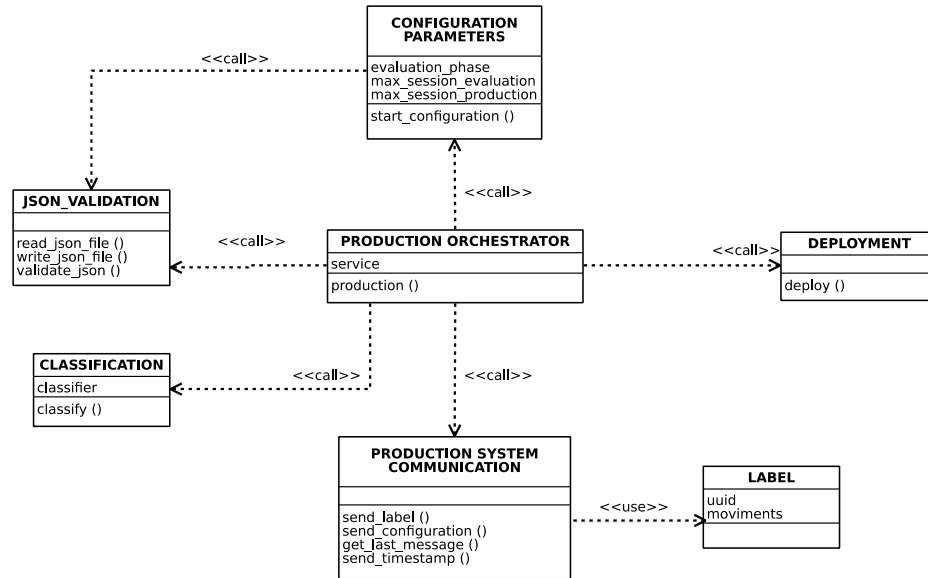


Figure 37 - UML Class Diagram of production system.

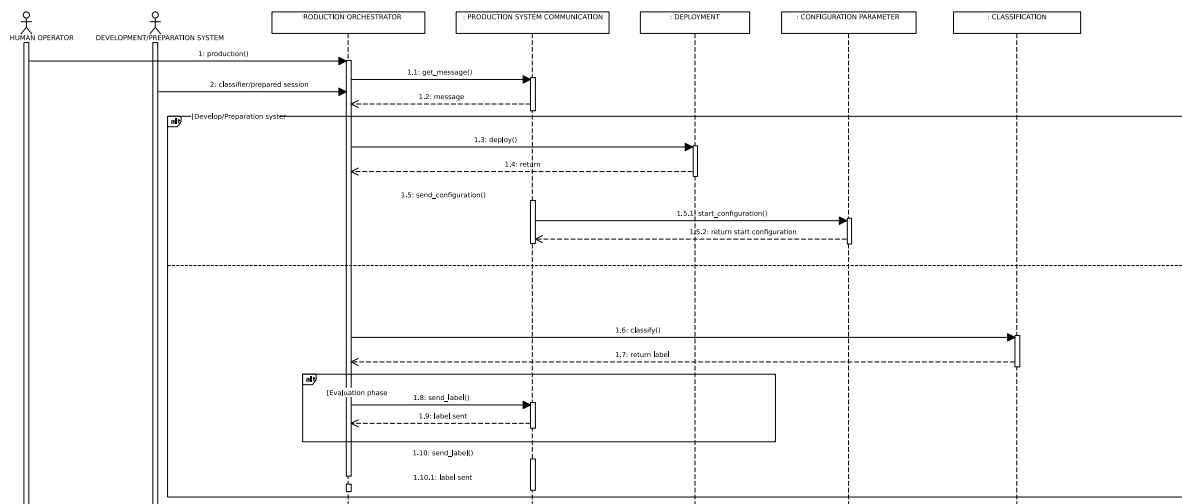


Figure 38 - Sequence diagram of classify and develop session.

5.5 Evaluate Classifier Performance

[Giovanni Ligato]

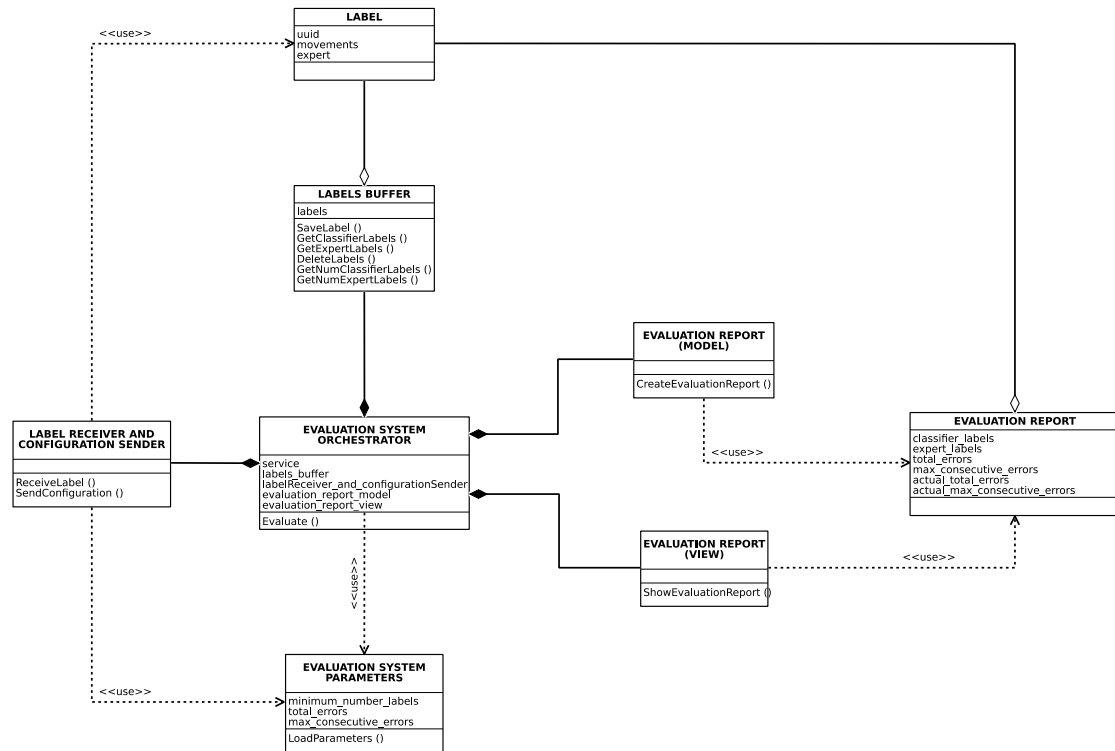


Figure 39 - UML Class Diagram of Evaluate Classifier Performance Process.

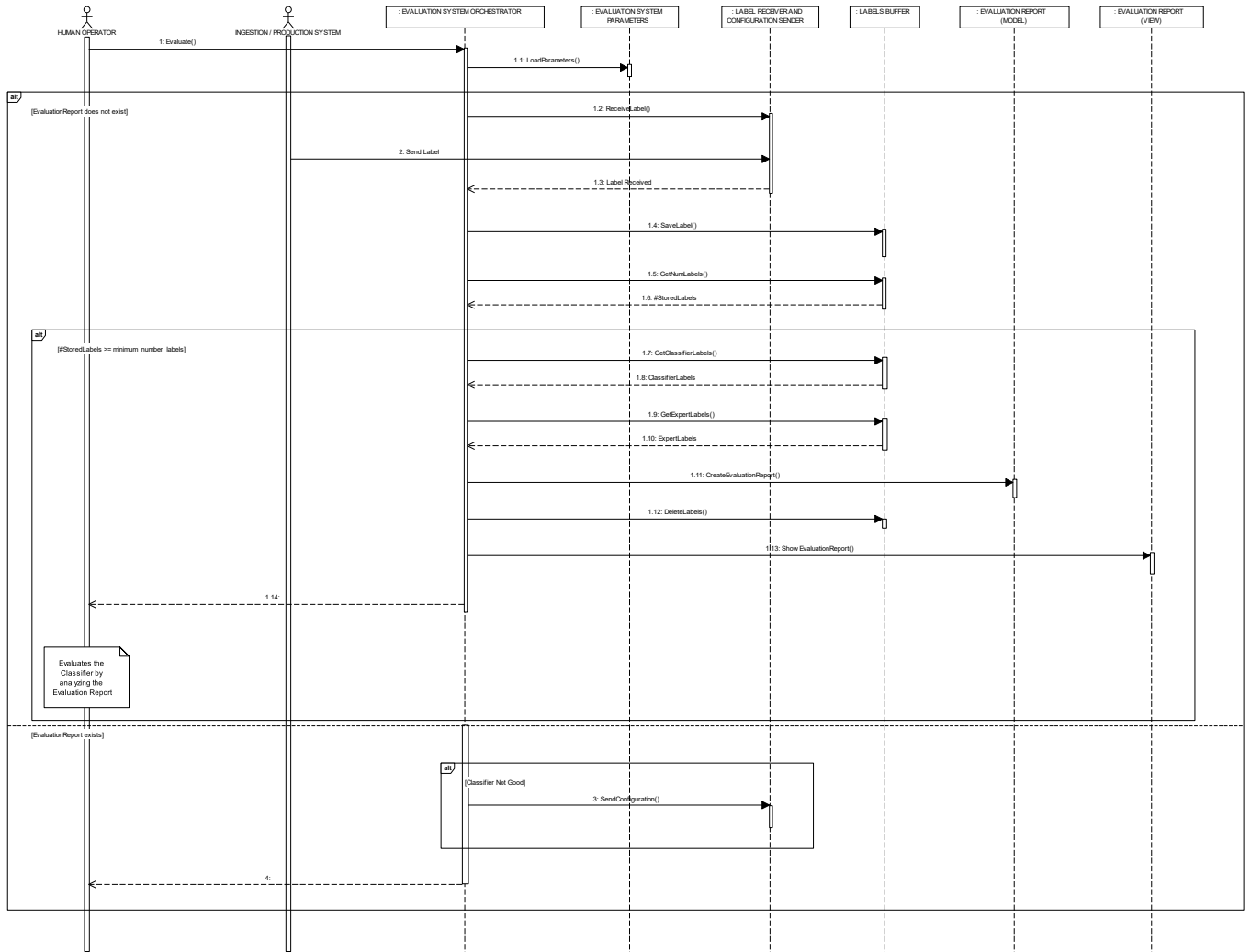


Figure 40 - Sequence Diagram of Evaluate Classifier Performance Process.

6 – Design

6.1 Ingestion System

[Francesco Taverna]

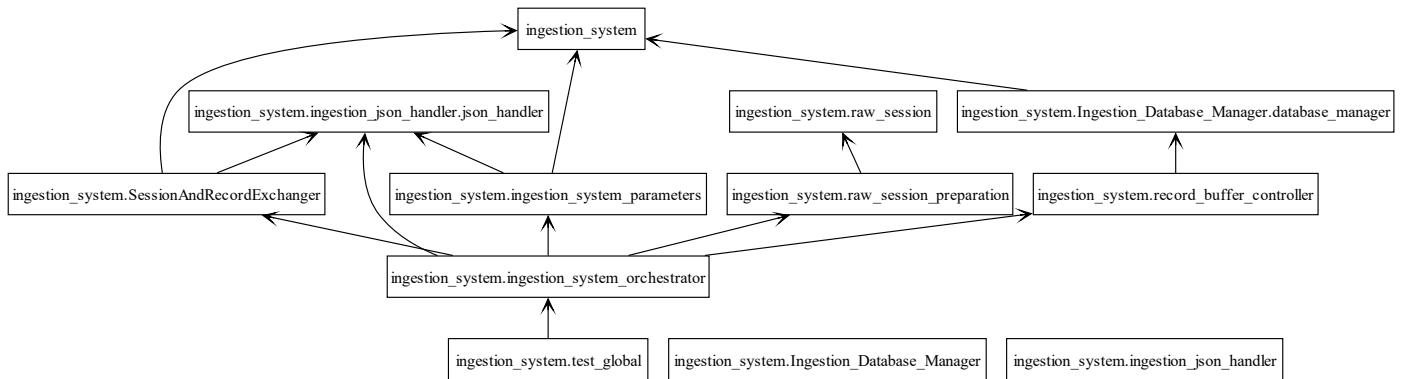


Figure 41 - Package Diagram of the Ingestion System.

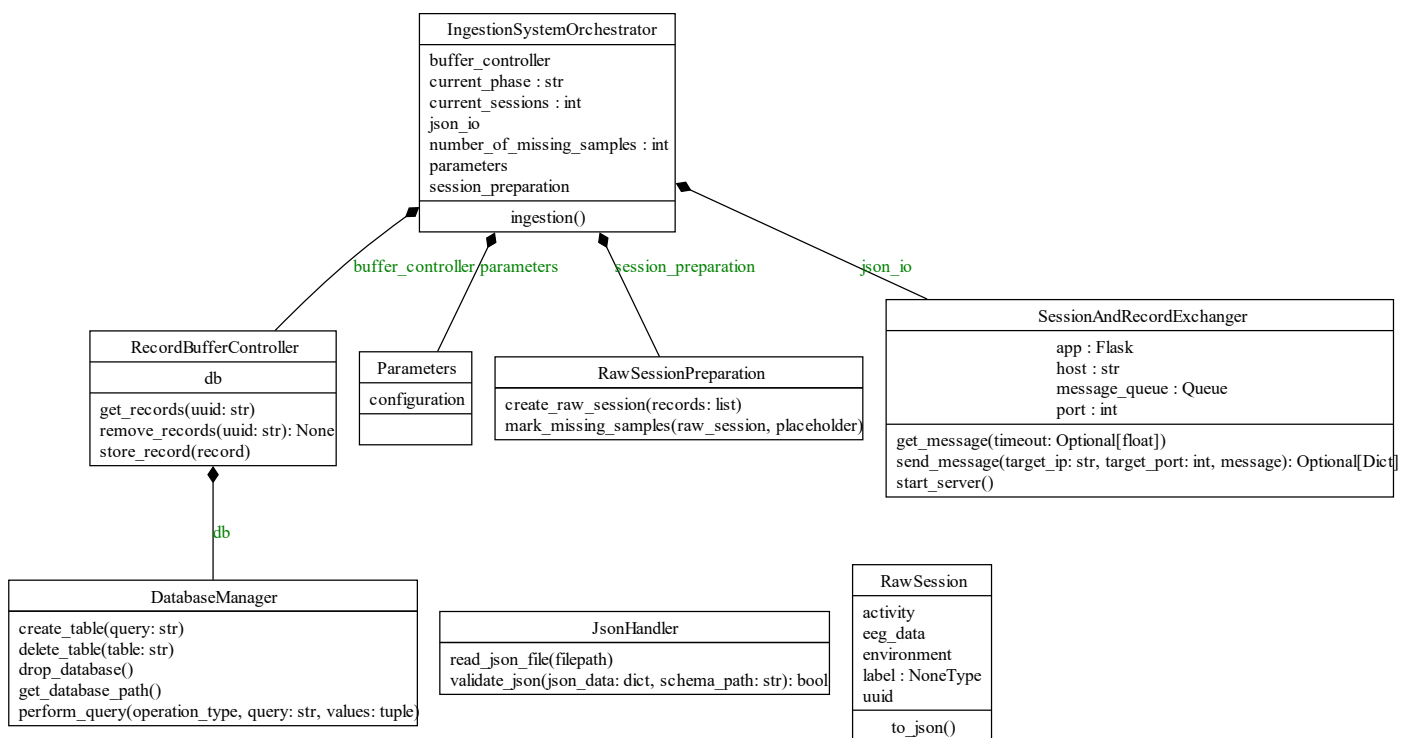


Figure 42 - UML Class Diagram of Ingestion System.

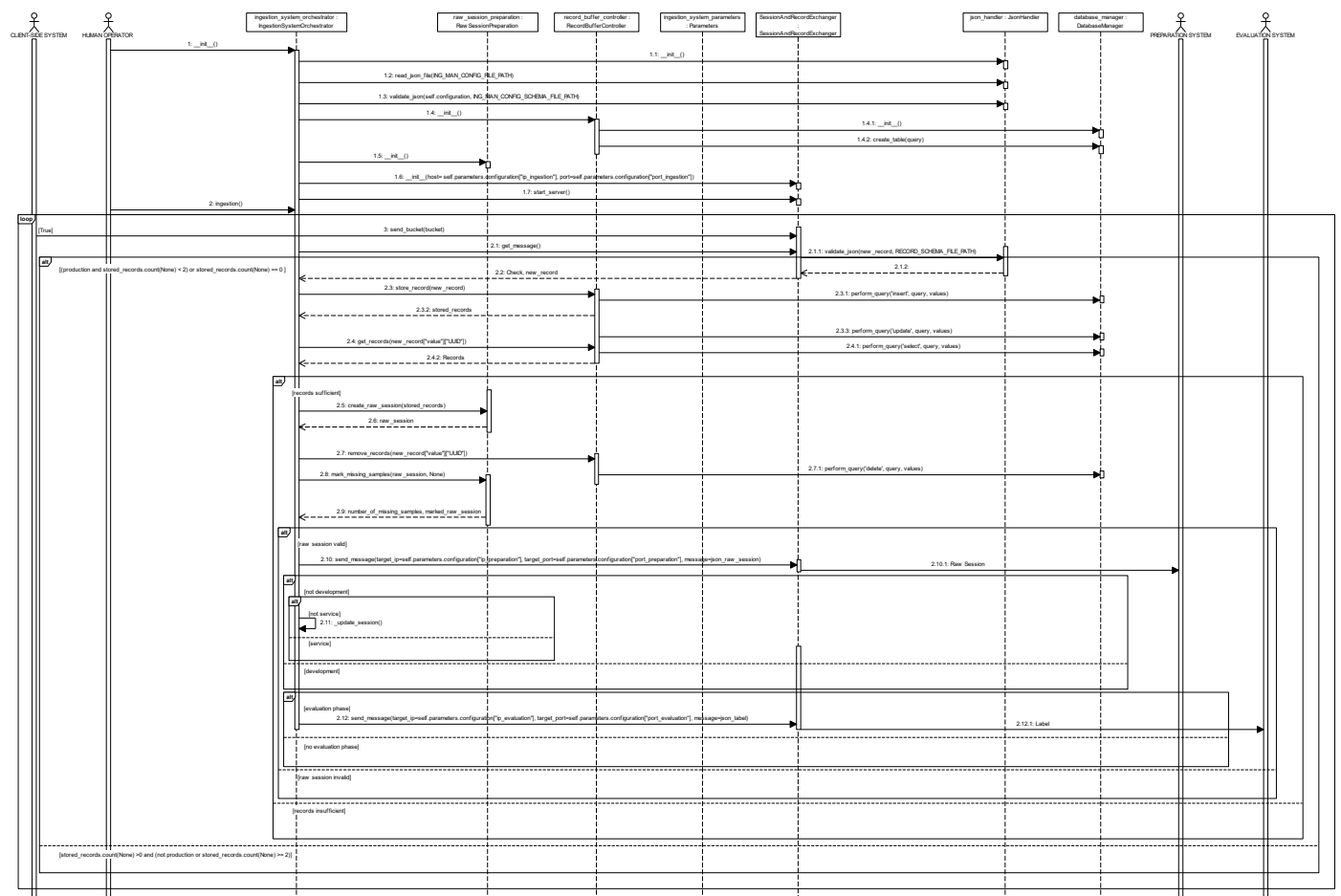


Figure 43 - Sequence Diagram of Ingestion System.

6.2 Preparation System

[Francesco Taverna]

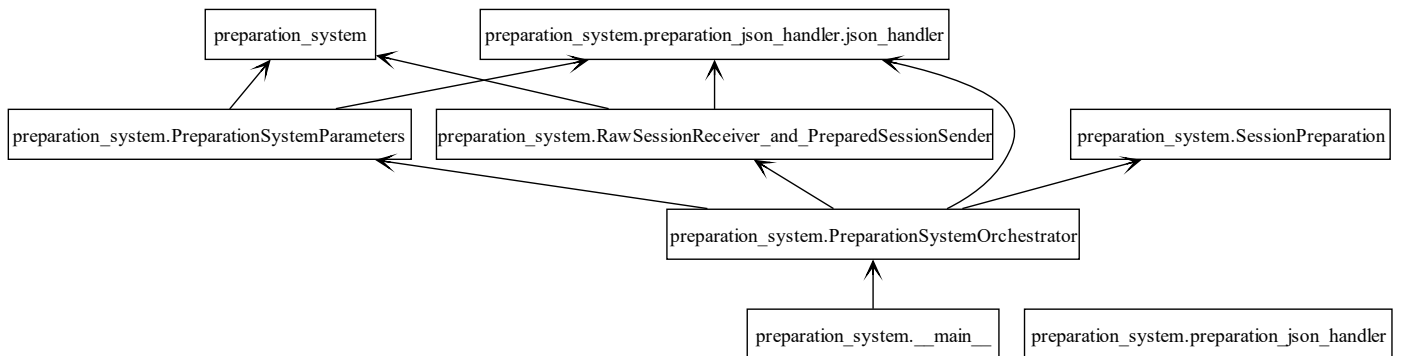


Figure 44 - Package Diagram of Preparation System

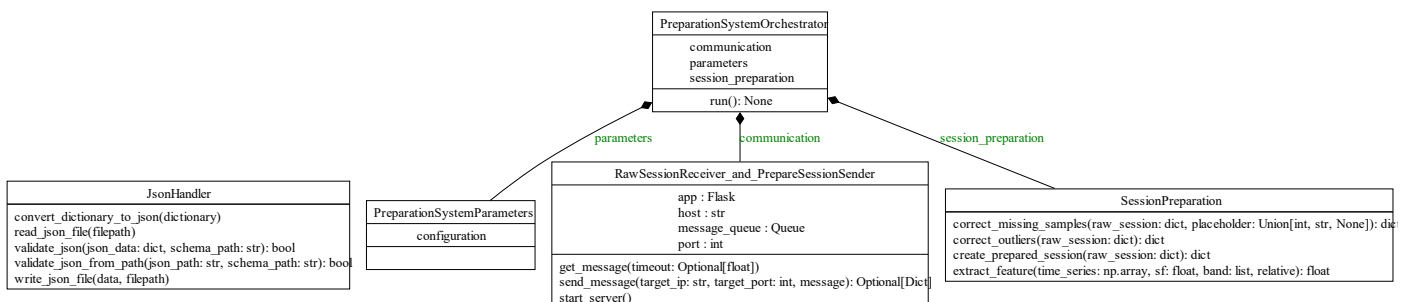


Figure 45 - UML Class Diagram of Preparation System

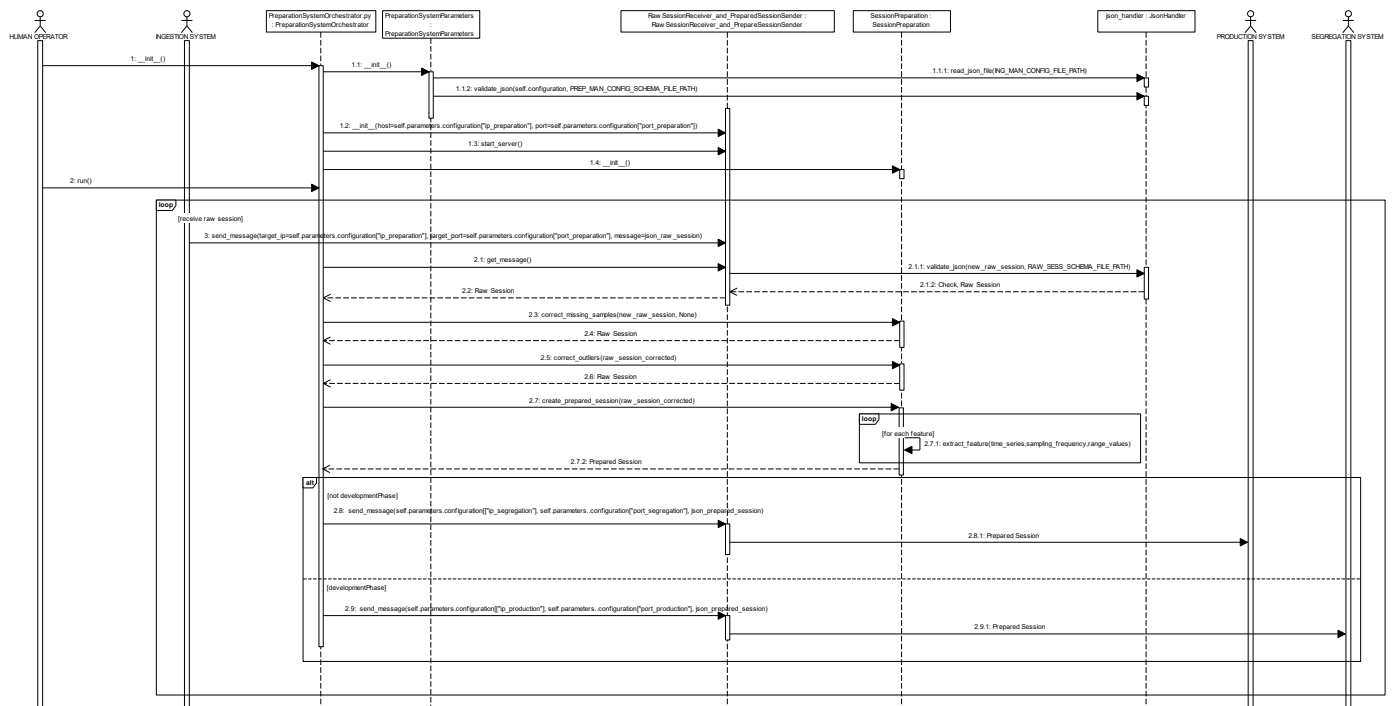


Figure 46 - Sequence Diagram of Preparation System.

6.3 Segregation System

[Saverio Mosti]

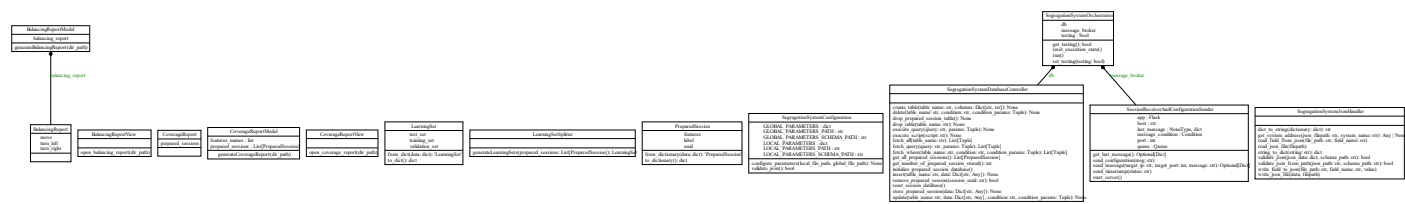


Figure 47 - UML Class Diagram of the Segregation System.

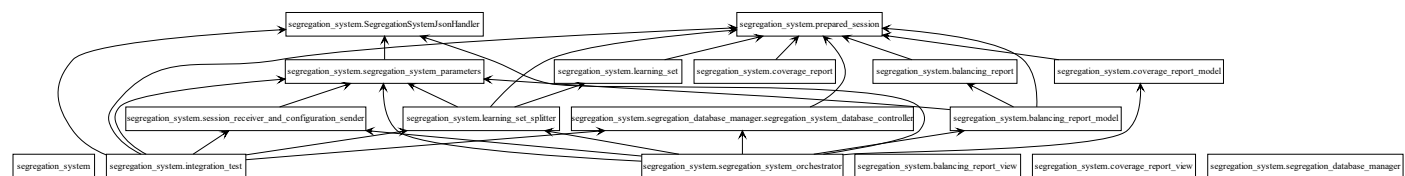


Figure 48 - Packages Diagram of the Segregation System

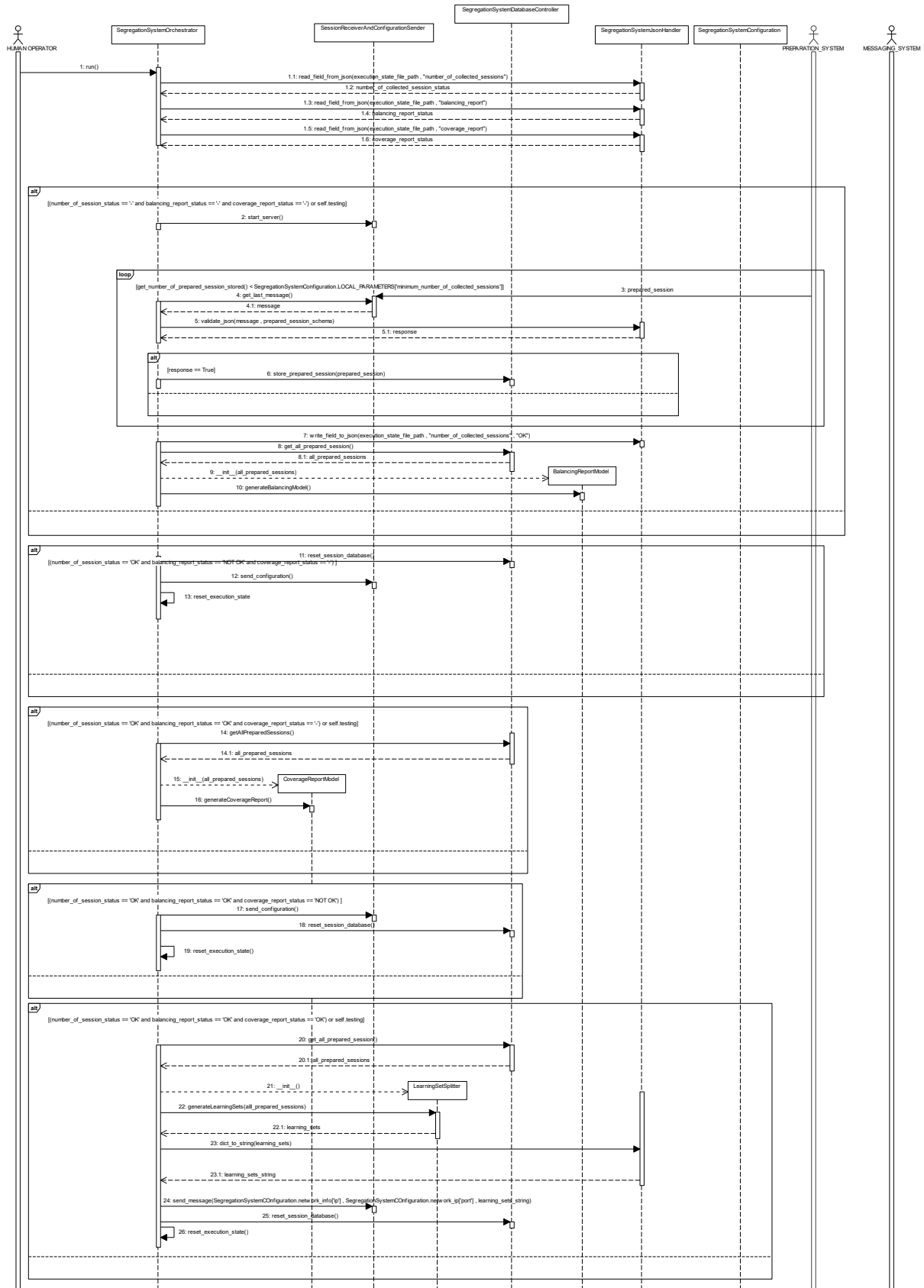


Figure 49 - Sequence Diagram of Segregation System.

6.4 Develop Classifier

[Gabriele Pianigiani]

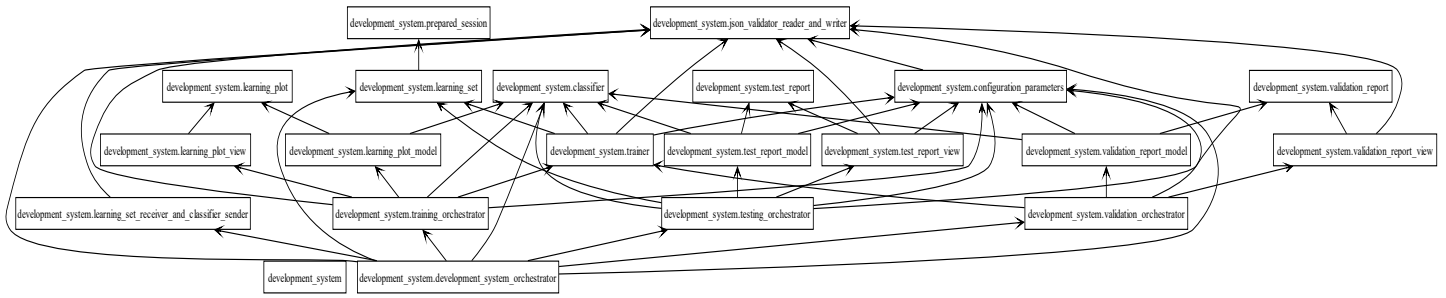


Figure 50 - Package Diagram of Develop Classifier.

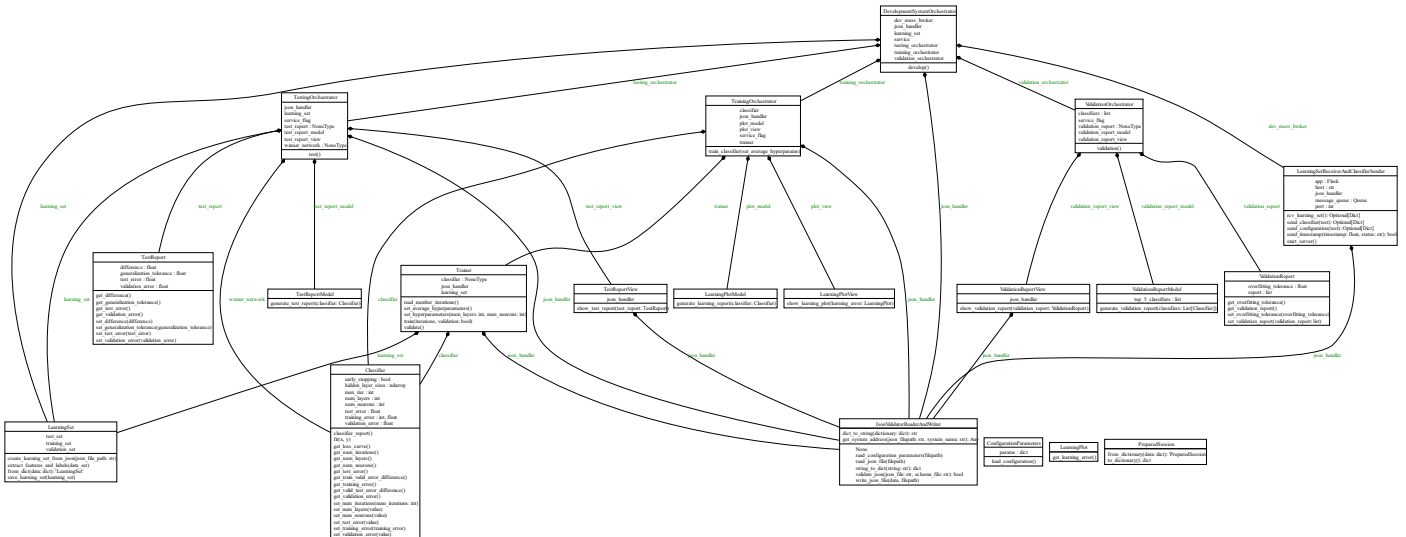


Figure 51 - UML Class Diagram of Develop Classifier.

6.5 Evaluate Classifier Performance

[Giovanni Ligato]

The following class diagram and package diagram are generated using the *pyreverse* tool made available by *pylint*.

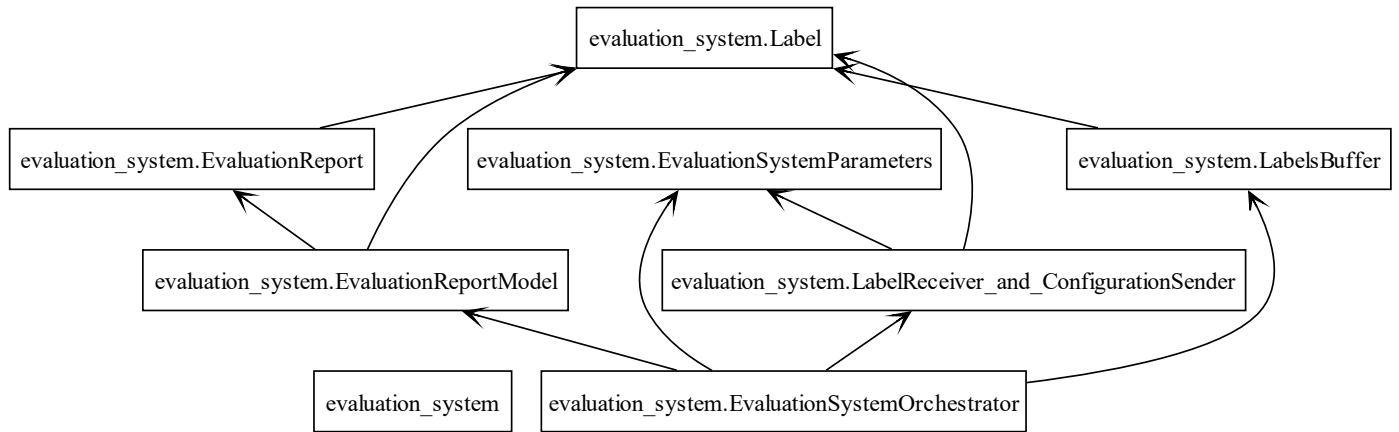


Figure 53 - Package Diagram of Evaluate Classifier Performance Process.

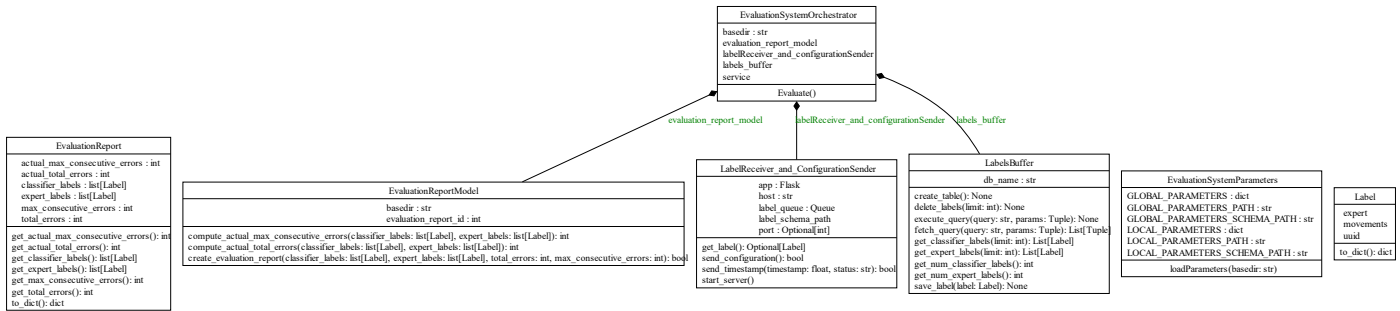


Figure 54 - UML Class Diagram of Evaluate Classifier Performance Process.

[Alessandro Ascani]

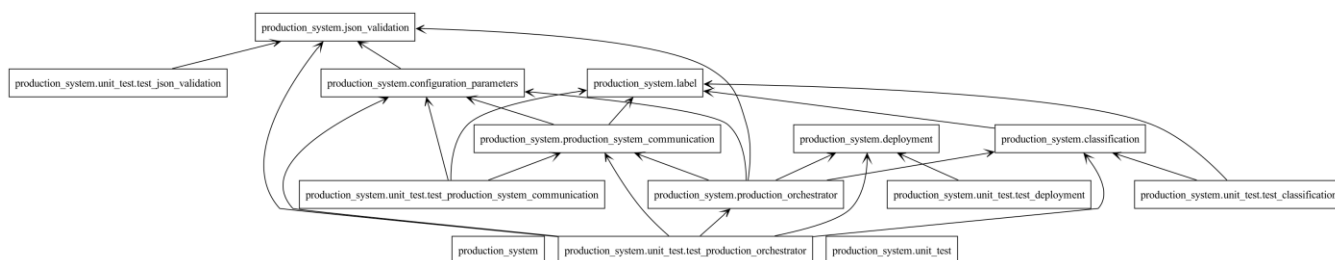


Figure 56 - Packages Diagram of Production System

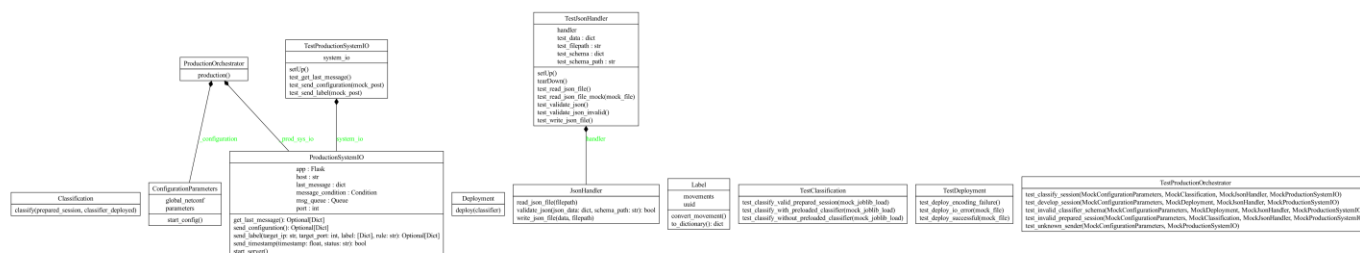


Figure 57 - UML Class Diagram of Production System

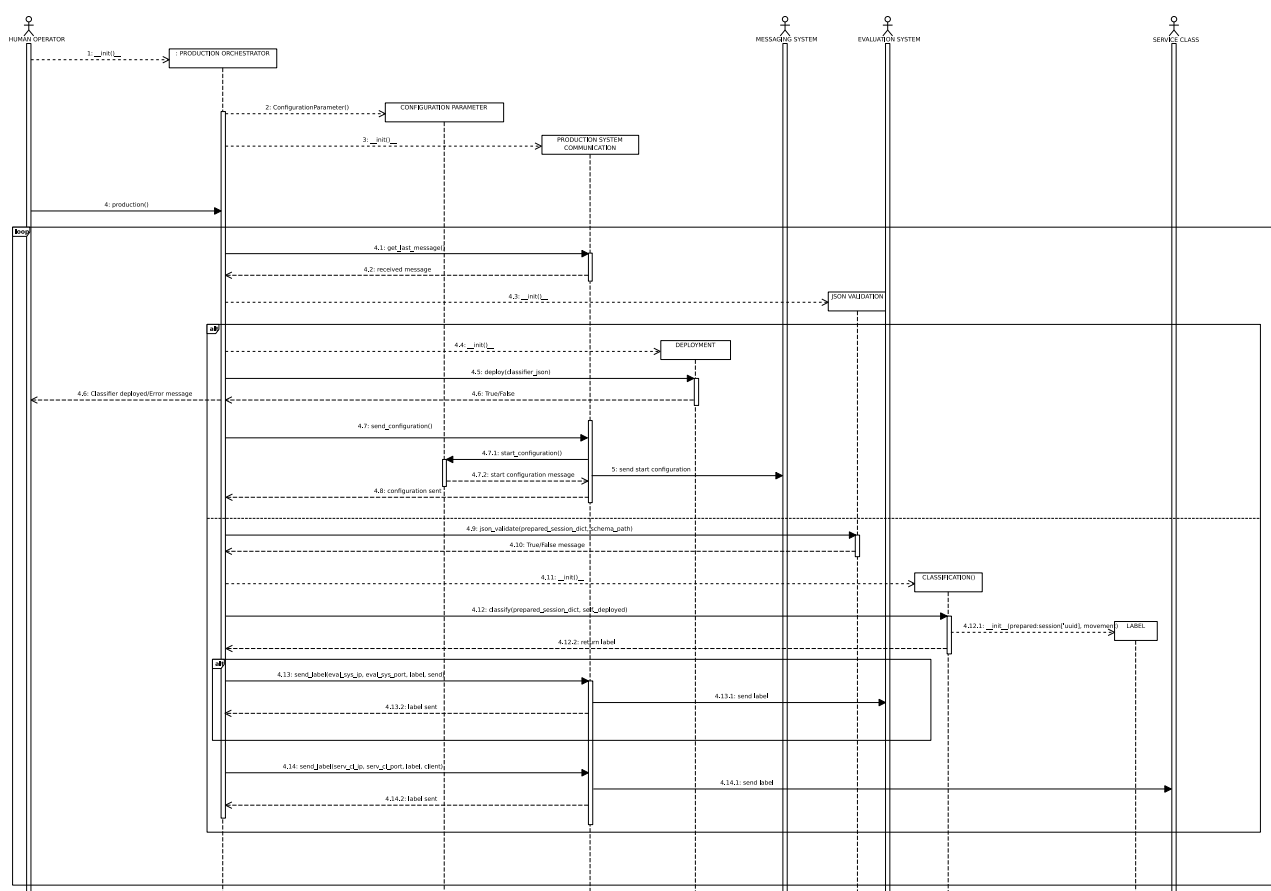


Figure 58 - Sequence diagram of the Production System

7. Testing

7.1 NON-INTEROPERABILITY

[Francesco Taverna]

Feature	Ingestion System	Preparation System	Segregation System	Development System	Production System	Evaluation System	Score
In case of unbalanced data, new data is waited (new sessions).	yes	yes	yes	n.a.	n.a.	n.a.	0
Multiple clients simultaneously send various records to the system.	no	no	no	no	no	no	6
If it is the evaluation phase, halt label classification and initiate a new development phase if the classifier performs poorly.	yes	yes	yes	yes	yes	yes	0
In case of insufficient number of labels, the evaluation system can request additional ones.	n.a.	n.a.	n.a.	n.a.	n.a.	no	1
TOTAL							7

7.2 NON-RESILIENCY

[Francesco Taverna]

Ingestion System

ID	Input	Consequence	Score
I1	Receive record with missing label in development/evaluation phase.	Record saved with missing label but not sent to the preparation system.	4
I2	Receive record with missing uuid.	Record saved with missing uuid but not sent to the preparation system.	4
I3	Receive record with missing time series samples under the threshold.	Detected, samples are marked.	1
I4	Receive record with missing time series samples over the threshold.	Detected, raw session is discarded.	4
I5	Receive sequentially records from different raw session.	Detected, records belonging to different raw sessions are saved in different rows of the database.	1
I6	Receive a record with wrong schema.	Detected, record is discarded.	4
I7	Raw session with wrong values (not in the list of valid values) for environment, label, activity.	Not detected: the prepared session will be sent with incorrect values.	5
TOTAL			28

[Francesco Taverna]

Preparation System

ID	Input	Consequence	Score
P1	Receive a raw session with wrong schema	Detected, raw session is discarded	4
P2	Raw session with wrong values (not in the list of valid values) for environment, label, activity	Not detected: prepared session will be sent with incorrect values	5
P3	Receive record with missing time series samples	Detected, samples are interpolated	1
TOTAL			10

[Saverio Mosti]

Segregation System

ID	Input	Consequence	Score
S1	Message is not sent by the Preparation System	The Segregation System keeps waiting for a message from the Preparation System.	5

S2	Wrong prepared session structure (<i>does not follow the schema</i>).	Detected and the prepared session is ignored .	4
S3	Wrong user response after evaluating the graph.	The Segregation System ignore the response and wait a correct one.	5
TOTAL			14

[Gabriele Pianigiani]

Development System

ID	Input	Consequence	Score
D1	Wrong dataset structure.	Wrong dataset error.	5
D2	Message wasn't sent by the Segregation System.	The Development System keep waiting for a message.	5
D3	Received a dataset containing sessions with wrong labels.	Not detected, the classifier is trained with wrong data.	5
D4	Wrong user response after evaluating the graph.	Validation error.	5
TOTAL			20

[Alessandro Ascani]

Production System

ID	Input	Consequence	Score
PS1	Wrong prepared session structure.	Detected: validation error.	4
PS2	Wrong classifier structure.	Not detected: the production system deploys the classifier with wrong structure.	5
PS3	Classifier trained with wrong prepared session's features.	Not detected: Classification executed with an incorrect trained classifier.	5
PS4	System receives a prepared session before to deploy a classifier.	Detected: classifier deployment error.	4
TOTAL			18

[Giovanni Ligato]

Evaluation System

ID	Input	Consequence	Score
E1	Insufficient Number of Labels Received.	The evaluation system becomes stuck while waiting for the minimum required number of	5




		labels, preventing further classifier evaluation.	
E2	Invalid Label Received.	The label fails validation against the schema, is identified as invalid, and is subsequently discarded.	4
E3	Duplicated Label Received.	It does not cause any issues in the database, as it results in no operation, effectively making no changes.	1
E4	Configuration Transmission Failure.	If the classifier is not good, the configuration restart message fails to reach the messaging system due to a connection break.	3
TOTAL			13

7.3 NON-AUTOMATION




Role's average salaries in Italy.

Role	Salary	Normalized Salary	Reference
Data Analyst	€30,000	1	https://www.payscale.com/research/IT/Job=Systems_Administrator/Salary
System Administrator	€32,431	1,081	https://www.payscale.com/research/IT/Job=Data_Analyst/Salary
ML Engineer	€32,670	1,089	https://www.payscale.com/research/IT/Job=Machine_Learning_Engineer/Salary




[Francesco Taverna]

Configure Ingestion System	COST CALCULATION (occurrence x cognitive x salary)	SCORE
The use case starts when the  <u>System administrator</u> opens the interface to configure the ingestion system.	1 x 1 x 1,081	1.081
The  <u>System administrator</u> configures the parameters.	5 x 4 x 1,081	21.62
The  <u>System administrator</u> clicks on "Apply" button.	1 x 1 x 1,081	1.081
TOTAL COST:		23.782




[Francesco Taverna]









Configure Preparation System	COST CALCULATION (occurrence x cognitive x salary)	SCORE
The use case starts when the  <u>System administrator</u> opens the interface for configuring parameters.	1 x 1 x 1,081	1.081
The  <u>System administrator</u> selects the phase.	1 x 1 x 1,081	1.081
The  <u>System administrator</u> presses the "Apply" button.	1 x 1 x 1,081	1.081
TOTAL COST:		3.243






[Gabriele Pianigiani]





Configure Development System	COST CALCULATION (occurrence x cognitive x salary)	SCORE
The use case starts when the  <u>ML Engineer</u> opens the interface to configure the development system.	1 x 1 x 1,089	1,089
SYSTEM shows Development System configuration parameter interface.		
The  <u>ML Engineer</u> for each parameter digits the value in the specific input field.	8 x 3 x 1,089	26,136
The  <u>ML Engineer</u> clicks on "Apply" button.	1 x 1 x 1,089	1,089
TOTAL COST:		28,314







[Gabriele Pianigiani]




Set number of iterations	COST CALCULATION (occurrence x cognitive x salary)	SCORE
1. The use case starts when the  <u>ML engineer</u> opens the interface for setting the number of iterations.	1 x 1 x 1,089	1,089
2.The  <u>ML Engineer</u> digits the number of iterations in the "Insert number of iterations" field.	1 x 3 x 1,089	3,267
3.The  <u>ML Engineer</u> clicks on the "Apply" button.	1 x 1 x 1,089	1,089
TOTAL COST:		5,445



Check Learning Plot	COST CALCULATION (occurrence x cognitive x salary)	SCORE
1. The use case starts when the  <u>ML Engineer</u> clicks on "Check Learning Plot"	$1 \times 1 \times 1,089$	1,089
2. SYSTEM shows a window with the Training error plot (MSE) against the number of iterations.		
3. if the  <u>ML Engineer</u> sees that the line in the plot is flat for at least half of the iterations then	$0,4 \times 3 \times 1,089$	1,307
3.1. The  <u>ML Engineer</u> reduces the number of iterations accordingly via the field and the button in the window	$0,4 \times 3 \times 1,089$	1,307
3.2. The  <u>ML Engineer</u> press the "TRAIN AGAIN" button	$0,4 \times 1 \times 1,089$	0,436
3.3. SYSTEM updates the plot with the new number of iterations		
4. else if The  <u>ML Engineer</u> sees that the line in the plot is decreasing till the end	$0,4 \times 3 \times 1,089$	1,307
4.1. The  <u>ML Engineer</u> increases the number of iterations accordingly via the field and the button in the window.	$0,4 \times 3 \times 1,089$	1,307
4.2. The  <u>ML Engineer</u> press the "TRAIN AGAIN" button	$0,4 \times 1 \times 1,089$	0,436
4.3. SYSTEM updates the plot with the new number of iterations		
5. else if The number of iterations is fine	$0,2 \times 3 \times 1,089$	0,653
5.1. The  <u>ML Engineer</u> presses the "VALIDATE" button.	$0,2 \times 1 \times 1,089$	0,218
6. end if		
TOTAL COST:		8,06

Check Validation Result	COST CALCULATION (occurrence x cognitive x salary)	SCORE
1. The use case starts when the SYSTEM visualizes the table results containing the top 5 classifier		
2. for each classifier in the report.		
2.1. The  <u>ML Engineer</u> checks if the difference is under the overfitting tolerance.	$5 \times 3 \times 1,089$	16,335
3. end for each		
4. if The  <u>ML Engineer</u> sees that there are no classifiers that satisfy the tolerance.	0,05	
4.1. The  <u>ML Engineer</u> presses the "NOT VALID" button	$0,05 \times 1 \times 1,089$	0,054
5. else	0,95	
5.1. The  <u>ML Engineer</u> selects the best classifier in the report based on the value of the difference, number of layers, number of neurons.	$0,95 \times 3 \times 1,089$	3,104
5.2. The  <u>ML Engineer</u> clicks on "SEND" button.	$0,95 \times 1 \times 1,089$	1,035
6. end if		
TOTAL COST:		20,528






Check Test Result	COST CALCULATION (occurrence x cognitive x salary)	SCORE
1. <i>The use case starts when the  ML Engineer opens the interface to check test results.</i>	1 x 1 x 1,089	1,089
2. SYSTEM shows the check test result interface		
3. The  ML Engineer sees the color of the value in the row "Test Error - Validation Error"	1 x 2 x 1,089	2,178
4. if the color is red	0,01	
4.1 The  ML Engineer clicks on "NO" button	0,01 x 1 x 1,089	0,011
5. else	0,99	
5.1 The  ML Engineer clicks on "YES" button	0,99 x 1 x 1,089	1,078
6. end if		
TOTAL COST:		4,356

Configure Segregation System	COST CALCULATION (occurrence x cognitive x salary)	SCORE
1. The use case starts when the  Data analyst opens the interface for configuring parameters.	1 x 1 x 1	1
2. The  Data analyst digits the number of sessions to collect in the field.	1 x 3 x 1	3
3. The  Data analyst selects the balancing tolerance interval in the "Select balancing tolerance interval" field.	1 x 3 x 1	3
4. The  Data analyst selects the training set percentage in the "Select training set percentage" field.	1 x 3 x 1	3
5. The  Data analyst selects the validation set percentage in the "Select validation set percentage" field.	1 x 3 x 1	3
6. The  Data analyst presses the "Apply" button.	1 x 1 x 1	1
Total Cost		14



Check Data Balancing	COST CALCULATION (occurrence x cognitive x salary)	SCORE
1. The use case starts when the  Data analyst launches the application.	1 x 1 x 1	1
2. SYSTEM loads the Check Data Balancing interface.		
3. The  Data analyst presses the "LOAD" button.	1 x 1 x 1	1
3. For each class		
4.1. if the bar level is lower or greater of the tolerance interval:	3 x 3 x 1	9
4.1.1. The  Data analyst presses the "NOT OK" button.	0,80 x 1 x 1	0,80





4.1.2. The  Data analyst requests a new configuration.	$0,80 \times 1 \times 1$	0,80
5. The  Data analyst presses the "OK" button.	$0,20 \times 1 \times 1$	0,20
Total Score		12,8

[Saverio Mosti]





Check Input Coverage	COST CALCULATION (occurrence x cognitive x salary)	SCORE
1. The use case starts when the  Data analyst launches the application.	$1 \times 1 \times 1$	1
2. SYSTEM loads the Check Input Coverage interface.		
3. The  Data analyst presses the "LOAD" button.	$1 \times 1 \times 1$	1
4. for each feature		
4.1. if the distribution of the values does not respect the expected distribution for the feature under examination.	$[6 \times 3] \times 4 \times 1$	56
4.1.1. The  Data analyst presses the "NOT OK" button.	$0,66 \times 1 \times 1$	0,66
4.1.2. The  Data analyst requests a new configuration.	$0,66 \times 1 \times 1$	0,66
end if		
5. end for each		
6. The  Data analyst presses the "OK" button.	$0,33 \times 1 \times 1$	0,33
Total Score		59,65


[Giovanni Ligato]

CONFIGURE EVALUATION SYSTEM	COST CALCULATION (occurrence x cognitive x salary)	SCORE
1. The use case starts when the  ML ENGINEER opens the CONFIGURE EVALUATION SYSTEM interface.	$1 \times 1 \times 1.089$	1.089
2. SYSTEM displays the configuration page to the  ML ENGINEER .		




3. The  ML ENGINEER sets the MINIMUM #LABELS field.	$1 \times 3 \times 1.089$	3.267
4. The  ML ENGINEER sets the TOTAL ERRORS field.	$1 \times 3 \times 1.089$	3.267
5. The  ML ENGINEER sets the MAX CONSECUTIVE ERRORS field.	$1 \times 3 \times 1.089$	3.267
6. The  ML ENGINEER presses the APPLY button to save the new configuration.	$1 \times 1 \times 1.089$	1.089
TOTAL COST:		11.979

[Giovanni Ligato]

EVALUATE CLASSIFIER	COST CALCULATION (occurrence x cognitive x salary)	SCORE
1. <i>The use case starts when the  ML ENGINEER opens the EVALUATE CLASSIFIER interface.</i>	$1 \times 1 \times 1.089$	1.089
2. SYSTEM displays a table to the  ML ENGINEER, showing classifier and expert labels for each session to evaluate.		
3.  ML ENGINEER checks if ACTUAL # TOTAL ERRORS and ACTUAL # MAX CONSECUTIVE ERRORS in the AUTOMATIC CHECKS session are labelled with OK .	$2 \times 2 \times 1.089$	4.356
3.1. if both automatic checks are labelled with OK :	0.14	
3.1.1.  ML ENGINEER clicks on the GOOD button.	$0.14 \times 1 \times 1.089$	0.152
3.2. else	0.86	

3.2.1.  ML ENGINEER clicks on the NOT GOOD button.	$0.86 \times 1 \times 1.089$	0.937
TOTAL COST:		6.534

[Alessandro Ascani]

Configure Production System	COST CALCULATION (occurrence x cognitive x salary)	SCORE
The use case starts when the  <u>System Administrator</u> opens the interfaces for configuring parameters	$1 \times 1 \times 1,081$	1,081
SYSTEM shows the Production System configuration parameters interface		
for each parameter		
The  <u>System Administrator</u> configures the parameters	$3 \times 3 \times 1,081$	9,729
end each		
The  <u>System Administrator</u> presses the "Apply" button.	$1 \times 1 \times 1,081$	1,081
TOTAL COST:		11,891

7.3 AUTOMATION RESPONSIVENESS-ELASTICITY of the Development phase

[Everyone]

The objective of this test was to evaluate the response times of the entire factory system when tasked with producing new classifiers. To assess its elasticity, an increasing number of classifiers were trained, and the results were plotted. The x-axis represents the increasing number of developed classifiers, while the y-axis depicts the average response time per classifier for the entire factory.

Testing Environment

The initial tests were conducted in a distributed environment across a network, with different components running on separate computers as follows:

- **Ingestion and Preparation system:** hosted on one computer.
- **Segregation system:** hosted on another computer.
- **Development system:** hosted on a third computer.
- **Production system:** hosted on a fourth computer.
- **Service class:** dedicated to rapidly transmitting records to the various systems, running on a separate computer.

To facilitate communication, the router ports of the respective locations were forwarded to the corresponding systems, and the public IPs of each router were shared. The systems were operated under heavy stress, with the CPU and other resources placed under significant load using specific stress-testing tools for the CPU and by opening numerous applications to increase RAM usage.

Initial Testing

The first test involved processing 500 sessions per classifier. However, this configuration proved inefficient, as only 12 classifiers were developed over approximately 5.5 hours. Due to the limited number of data points and the similarity of the behaviour to subsequent tests, the results of this test are not presented.

In the second test, the number of sessions per classifier was reduced to gather more data points and better observe the curve's behaviour. Over a six-hour testing period, 70 classifiers were successfully developed. The results, depicted in Figure 59, demonstrate a predominantly linear trend. It was observed that the average response time per classifier initially increased significantly but eventually followed a linear progression.

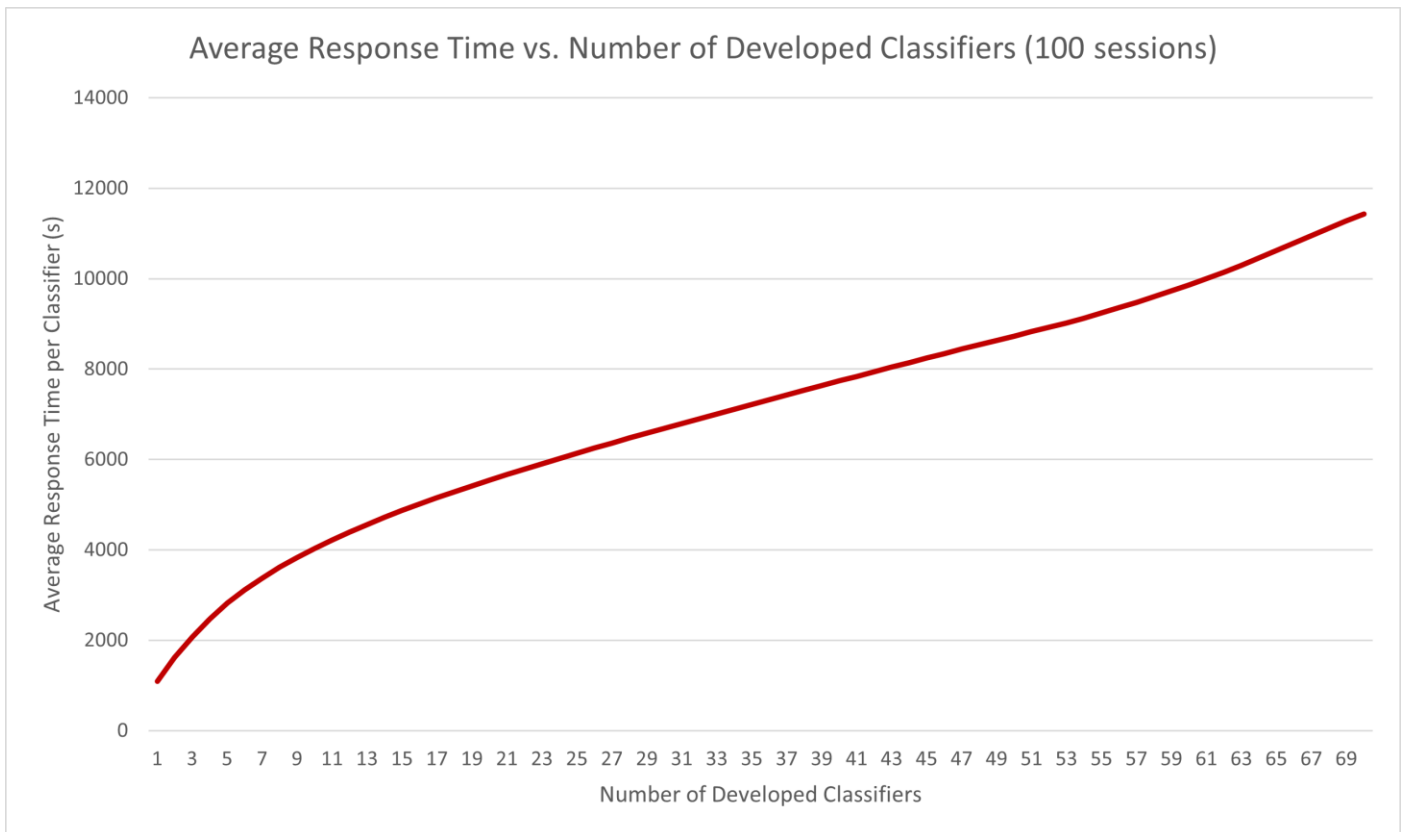


Figure 59 - Linear increase in average response time per classifier as the number of developed classifiers increases, based on the results from the second test under distributed conditions.

Controlled Environment Testing

To further analyse the observed behaviour, additional tests were conducted under controlled conditions. Initially, the rate of records transmitted by the service class was found to average only four records per second. This limitation was attributed to network latency, as the systems were distributed across different networks. To address this issue and ensure a more controlled environment, all systems were migrated to a single Ubuntu virtual machine equipped with one processor and 4 GB of RAM. This setup allowed better control over resource stress and ensured the machine was placed under heavy load during testing.

Under these conditions, the transmission rate increased tenfold to 40 records per second. To maximize data collection within a shorter timeframe, the number of sessions per classifier was further reduced to six records. During the execution of the tests, the virtual machine's resources were carefully monitored. It was observed that the swap area usage increased significantly until it reached 100%, causing the system to freeze completely. A forced shutdown was required, as no commands could be executed. This situation is illustrated in Figure 60.



Figure 60 - System resource utilization in the controlled virtual machine environment, showing 100% swap area usage, leading to a system freeze.

Despite these challenges, the results, shown in Figure 61, indicate a linear trend across the total number of developed classifiers.

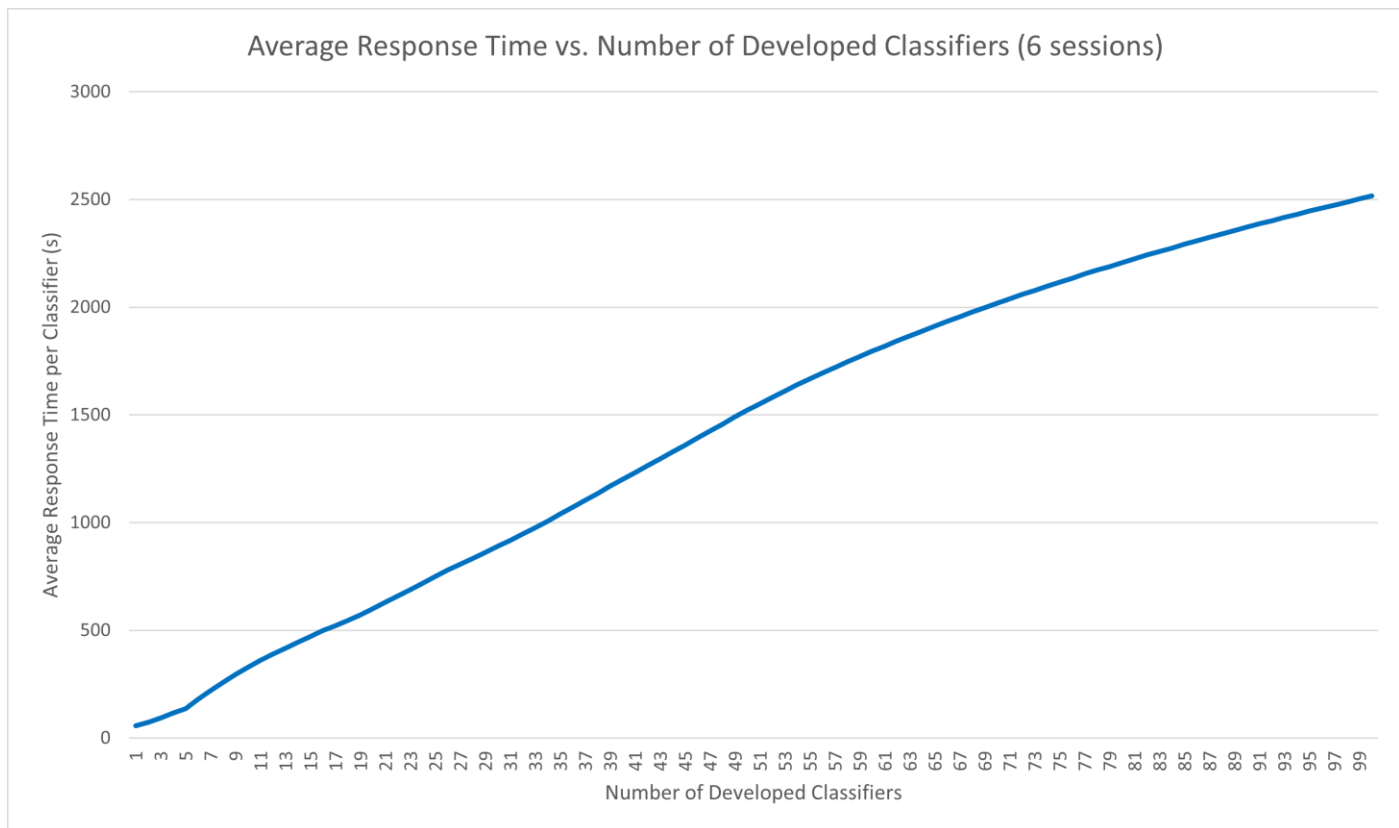


Figure 61 - Average response time per classifier in the controlled virtual machine environment, illustrating a consistent linear trend despite resource constraints.

Observations and Mathematical Analysis

The observed behaviour necessitated a deeper analysis to understand the underlying causes. The average response time, ($T_{\text{average},n}$) was computed as follows:

For n -th classifiers:

$$T_{\text{average},n} = \frac{\sum_{i=1}^n (t_{\text{end},i} - t_{\text{start},1})}{n}$$

Where:

- ($t_{\text{end},i}$): time of completion for the i -th classifier.
- ($t_{\text{start},1}$): start time of the first classifier.
- (n) total number of completed classifiers.

Each new classifier contributes to the numerator with the term ($t_{\text{end},i} - t_{\text{start},1}$), while the denominator increases by one.

To achieve a linear increase in the average response time, the completion time for each classifier (i) must remain constant and equal to t_{fixed} . This implies:

$$t_{\text{end},i} = t_{\text{start},1} + i \cdot t_{\text{fixed}}$$

Substituting this into the average formula:

$$T_{\text{average},n} = \frac{\sum_{i=1}^n [(t_{\text{start},1} + i \cdot t_{\text{fixed}}) - t_{\text{start},1}]}{n}$$

Simplifying further:

$$T_{\text{average},n} = \frac{\sum_{i=1}^n (i \cdot t_{\text{fixed}})}{n}$$

The numerator represents the sum of an arithmetic progression:

$$\sum_{i=1}^n (i \cdot t_{\text{fixed}}) = t_{\text{fixed}} \cdot \frac{n \cdot (n + 1)}{2}$$

Thus:

$$T_{\text{average},n} = \frac{t_{\text{fixed}} \cdot \frac{n \cdot (n + 1)}{2}}{n}$$

Simplifying further:

$$T_{\text{average},n} = t_{\text{fixed}} \cdot \frac{n + 1}{2}$$

Behaviour Analysis

The average response time grows linearly with n :

$$T_{\text{average},n} = \frac{t_{\text{fixed}}}{2} \cdot n + \frac{t_{\text{fixed}}}{2}$$

- The term $\frac{t_{\text{fixed}}}{2} \cdot n$ indicates an increase proportional to the number of completed classifiers.
- The constant term $\frac{t_{\text{fixed}}}{2}$ represents an initial offset.

The observed behaviour suggests that the time required for developing a single classifier remains constant, even under stress. This implies that the ingestion system consistently requires the same time to send sessions for classifier development. Systems downstream of the ingestion system are not heavily loaded, as in general they can process incoming data before new data arrives. Consequently, the ingestion system constitutes the bottleneck (*it is the slowest system*) of the factory, handling incoming records with constant processing time.

7.4 RESPONSIVENESS-ELASTICITY of the Production phase

This test evaluated the response time of the factory system in its capacity to classify labels for unseen sessions of records. The elasticity of the system was assessed by sending an increasing number of sessions to the factory for classification. To facilitate the test, the concept of a "bucket of sessions" was introduced. A bucket is defined as a collection of records corresponding to the sessions to be classified by the factory. The service class sends the records in each bucket to the ingestion system in random order. Once all records in the current bucket are sent, the service class prepares a new bucket containing an increased number of sessions, starting from one. As each bucket is conceptually independent, the start and end times of each bucket are used to compute the average production time.

Start time: The time at which the first record of a bucket is sent.

End time: The time at which the last label of all sessions in the bucket is received from the production system.

The average production time per session for each bucket is calculated as:

$$\text{Average Production Time per Session} = \frac{\text{End Time} - \text{Start Time}}{\text{Number of Sessions in Bucket}}$$

Test Setup

The test was conducted in a distributed environment over a network. Unlike the development phase tests, the following systems were involved in this phase:

- **Ingestion and preparation system:** hosted on one computer.
- **Production system:** hosted on another computer.
- **Service class:** hosted on a third computer.

As with previous tests, all systems were subjected to stress conditions during execution.

Observations

Unlike the development phase, where the number of sessions per classifier remained constant and the time required for classifier development remained uniform, the time required to classify buckets in this test varied. This variation was due to the increasing number of sessions in each subsequent bucket, which directly impacted both the sending and classification times.

After running the test for three hours, the results were plotted in Figure 62, which displays the average production time per session for buckets containing 1 to 90 sessions. The following observations were made:

1. Linear Trend for 1 to 70 Sessions:

Between 1 and 70 sessions, the curve exhibits a linear trend with a steady slope, indicating a proportional increase in production time as the number of sessions grows.

2. Increased Slope Beyond 70 Sessions:

For buckets containing 70 to 90 sessions, the curve remains linear but with a steeper slope. Initially, the ingestion system was considered the bottleneck. However, further analysis revealed that the bottleneck at this stage shifts to the production system.

- When the number of sessions becomes significant (after 70 sessions), the production system receives a large influx of sessions for classification. Each session requires the classifier to be loaded from the file system, which involves reading files of several tens of kilobytes.
- As the production system's memory becomes saturated, the system starts swapping memory pages to disk. This introduces a concurrent load on the disk, which now must both save memory pages to the swap area and read the classifier files for classification.
- This disk contention leads to increased average production times per session, which is reflected in the steeper slope observed in the graph beyond 70 sessions.

3. Responsiveness Threshold Unmet:

A dotted red line at 0.5 seconds on the plot represents the factory's target responsiveness threshold for classifying a single session. Unfortunately, the factory system is unable to meet this threshold in the current setup. Even when a bucket containing only a single session is sent, the response time exceeds 0.5 seconds due to the latency introduced by running the systems on different networks and computers.

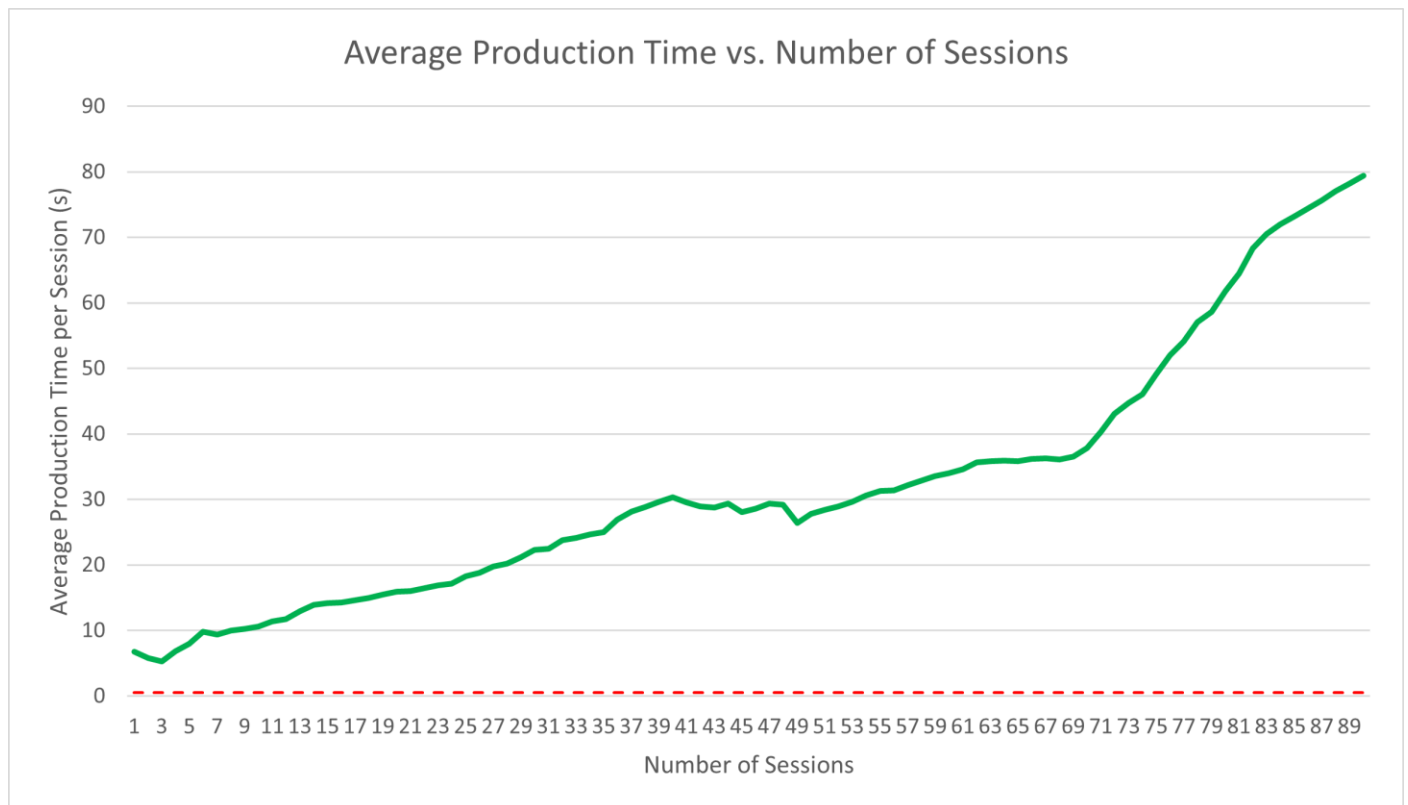


Figure 62 - Average production time per session as a function of the number of sessions in a bucket (1 to 90). The dotted red line represents the target responsiveness threshold of 0.5 seconds, which the factory system fails to meet under the current setup.