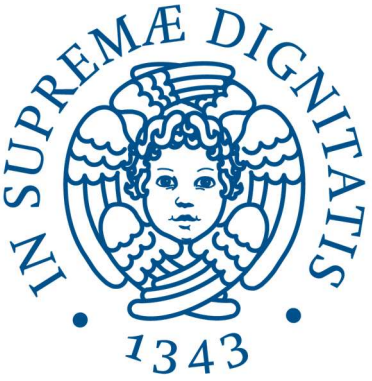


UNIVERSITÀ DI PISA

SECURE POS

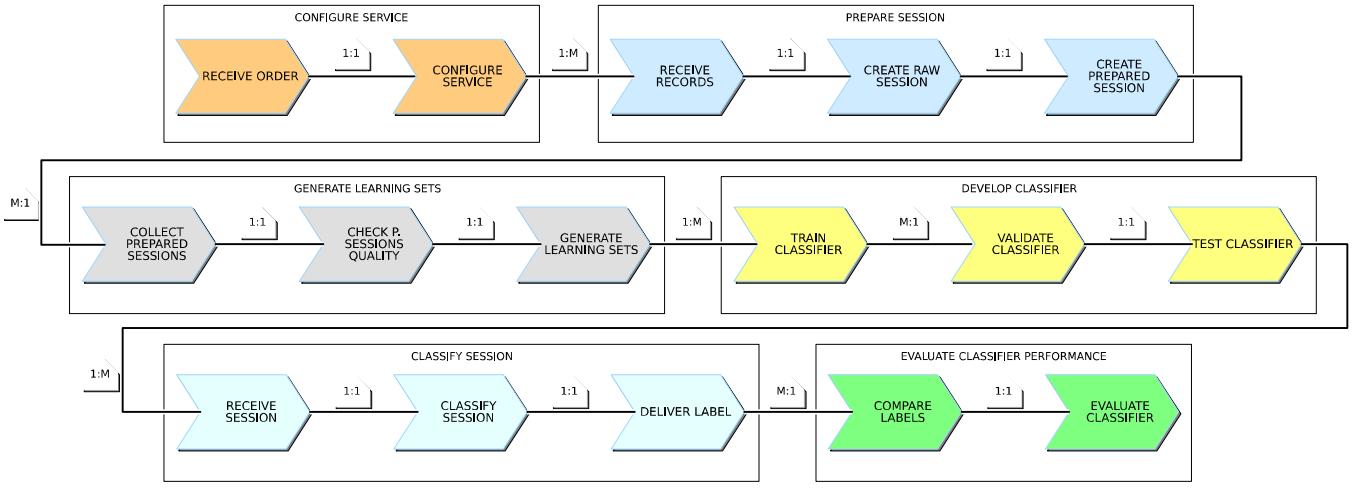
SOFTWARE SYSTEMS ENGINEERING PROJECT

A.Y. 2024/2025

JACOPO CARLON
MATTEO HALILAGA
GIOVANNI ENRICO LONI
LORENZO MANCINELLI
NICOLA RICCARDI

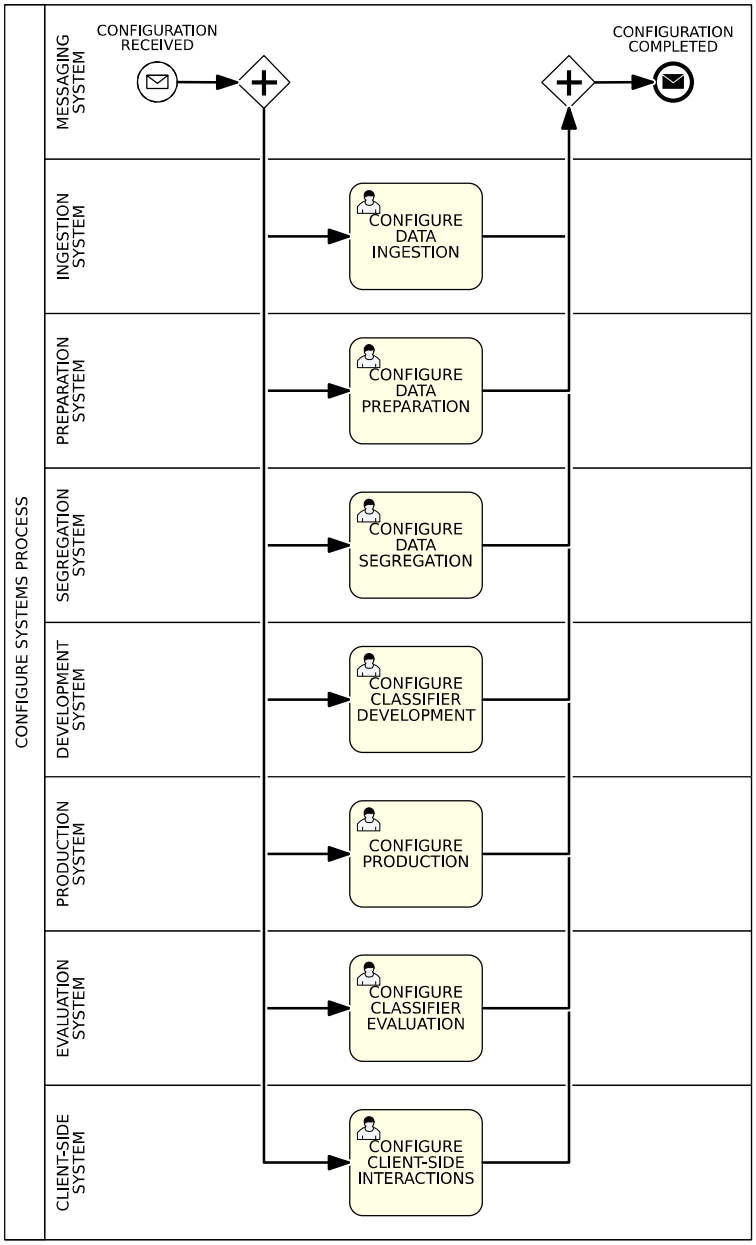
SOMMARIO

PROCESS LANDSCAPE DIAGRAM

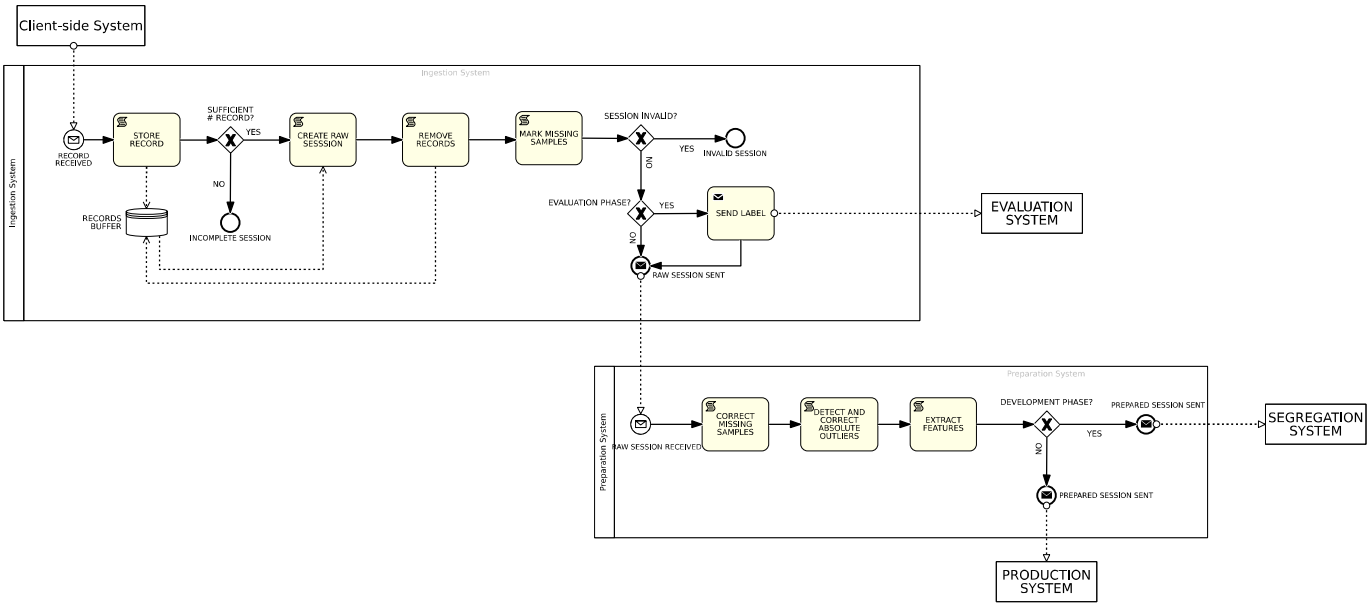


BPMN DIAGRAMS

CONFIGURE SYSTEMS PROCESS



PREPARE SESSION [Matteo]

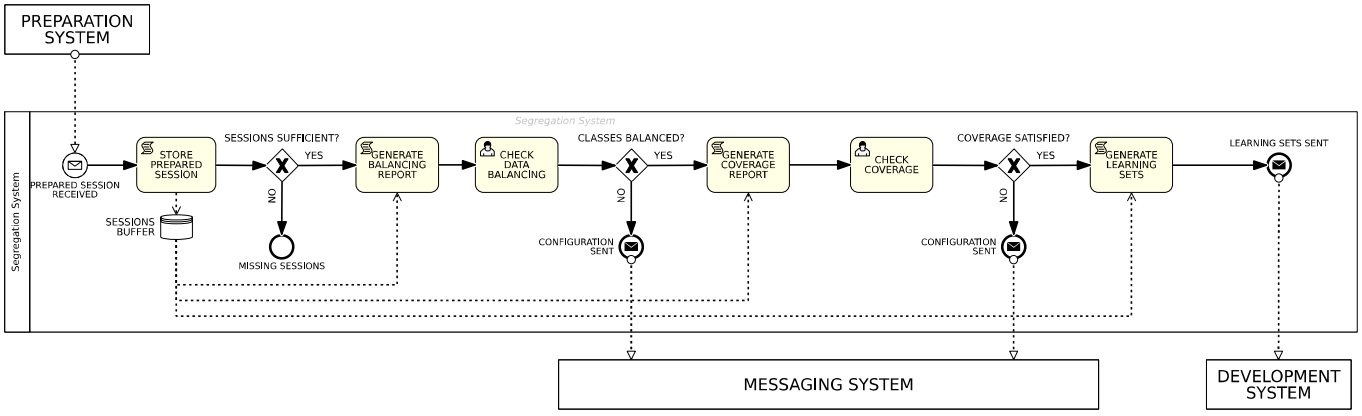


The system receives records from the client, stores them, and checks if there are enough records to create a raw session. Once this condition is reached, it generates the raw session, removes the records, and marks any missing samples.

Next, it verifies the validity of the session. If the system is in the evaluation phase, it sends a message containing the session token and the label to the evaluation system before forwarding the raw session to the preparation system; otherwise, it sends the raw session directly.

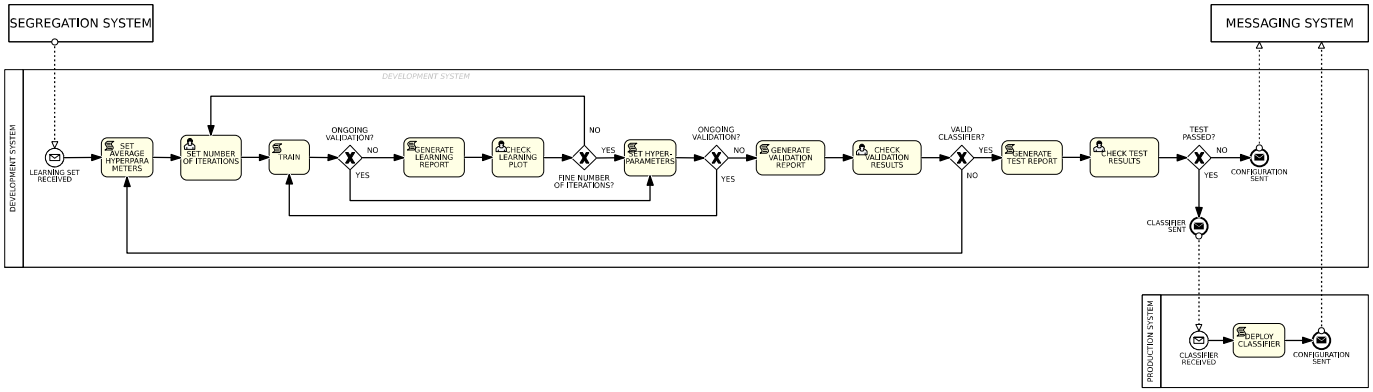
Upon receiving a raw session, the Preparation System corrects missing samples and addresses outliers in the data. Finally, it extracts relevant features and routes the session either to the segregation system if in the development phase, or to the production system otherwise.

GENERATE LEARNING SETS [Lorenzo]



This system receives the prepared sessions from the preparation system. It stores them and checks if they are enough to generate our learning sets. The system analyzes the balancing of the data of those sessions, and if data are unbalanced send a message in order to reconfigure the system. Otherwise, it checks the coverage. As before, if the coverage is not satisfied send a message to make a reconfiguration. Otherwise, it generates learning sets and sends them to the development system.

DEVELOP CLASSIFIER PROCESS [Nicola]



The system receives a learning set and sets starting average parameters.

A human must then set the number of iterations. A classifier is trained, and a learning plot is generated. A human checks the learning plot and decides if the number of iterations was enough. This sequence is repeated until the number of iterations is good.

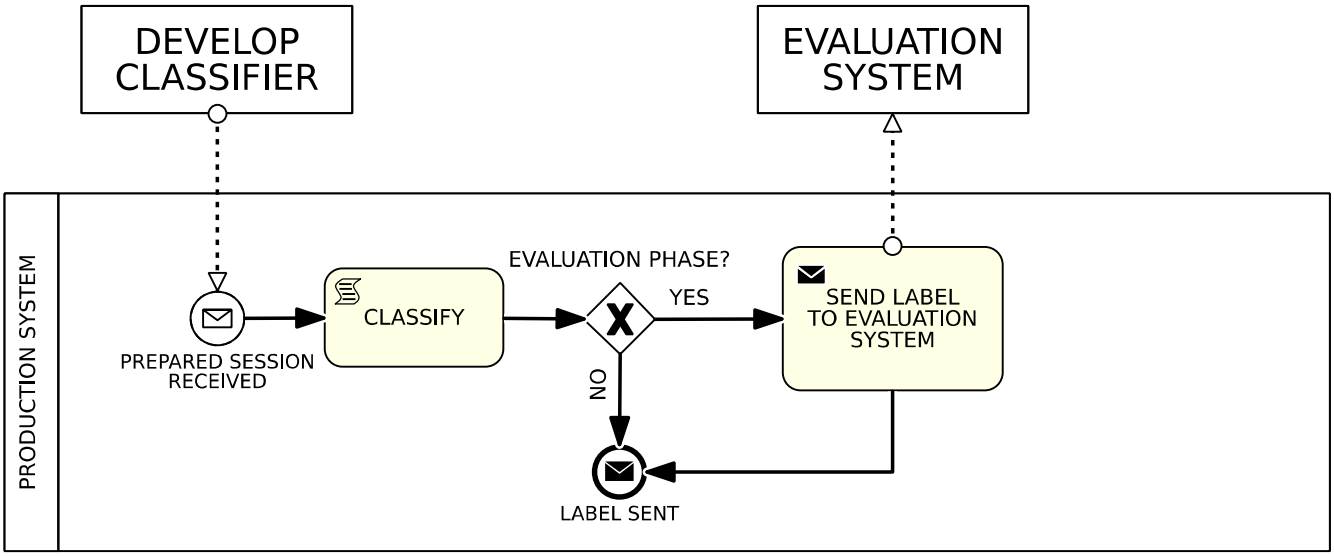
Then validation starts: a group of classifiers is generated by setting hyperparameters and training the model in a cycle.

When validation ends, a validation report is generated and a human analyses it. If there isn't any good classifier, the whole process restarts.

If there is a good classifier, it is tested, and a test report is generated. A human checks these results and decides if the test is passed. Then:

- If the test is passed, the fitted classifier is sent to the production system, that will deploy it and send a message to the messaging system to notify that the deployment succeeded.
- If the test is not passed, a message will be sent to notify that the development failed.

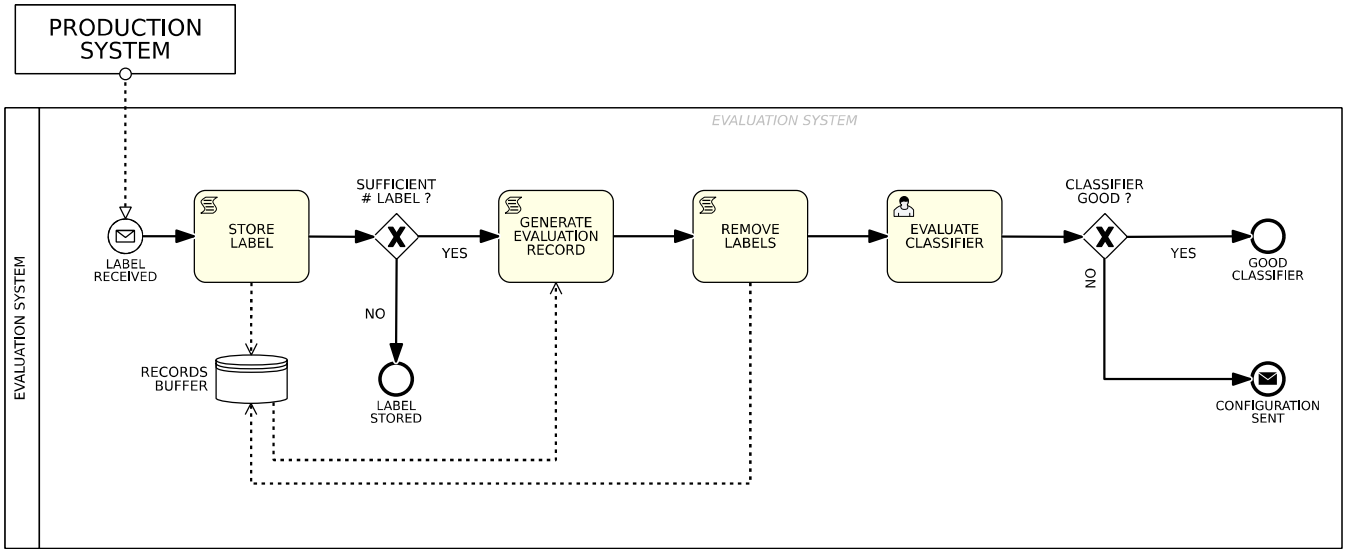
CLASSIFY SESSION [Enrico]



The system receives prepared sessions to be classified, thus the production system should already have deployed the classifier model.

After the classification, if the current phase is settled to 'Evaluation', the system sends the label given by the Classifier to the Evaluation System.

EVALUATE CLASSIFIER PROCESS [Jacopo]



This system receives some labels over time and starts collecting them in a buffer.

The system needs to have enough labels from both the classifier and the expert, so as to check if they match.

If it notices it has enough labels, it extracts them, deletes them from the DataBase, and generates an <EvaluationReport>.

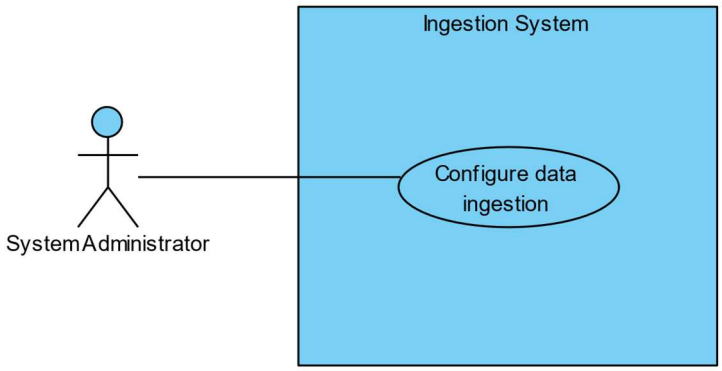
The Security Expert checks the report:

- If the classifier is good, everything works fine and there's nothing more to do.
- If the classifier is bad, a configuration message is sent in order to restart the whole training-balancing process.

ANALYSIS – USE CASES

PREPARATION/INGESTION SYSTEM [MATTEO]

Configure Ingestion System



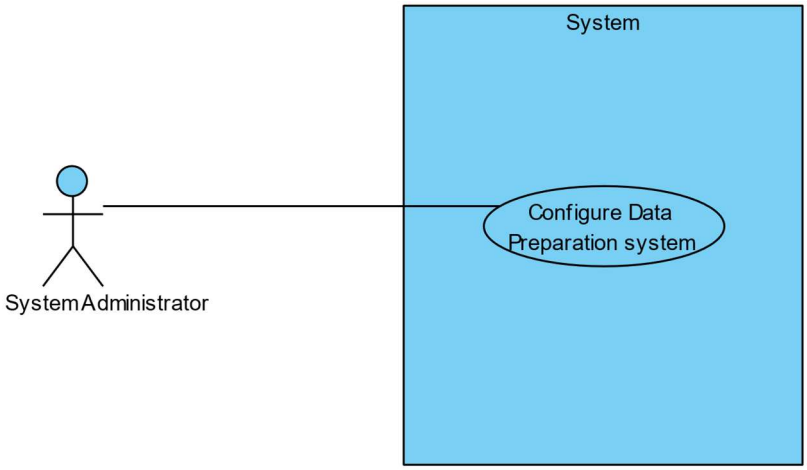
Scenario

1.The System Administrator open the configuration file through an editor.
2.The System Administrator update the configuration parameter using JSON.
3.The System Administrator save and close the configuration file.
4.The System Administrator restart the Ingestion System

Details

Name	Value
Preconditions	Configuration for data ingestion system required
Post-conditions	The Ingestion System is updated and restarted

Configure Data Preparation System



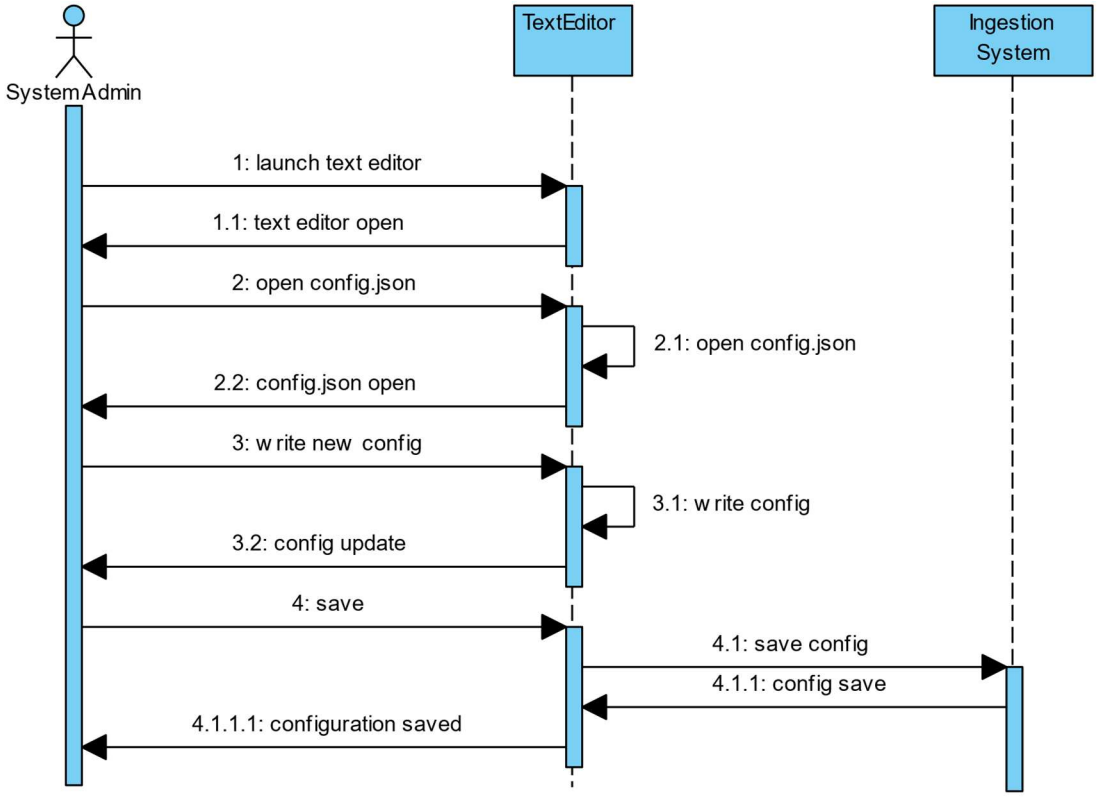
Scenario

1.The System administrator open the file config.json through a text editor
2.Update the configuration parameter then save, close and restart the preparation system

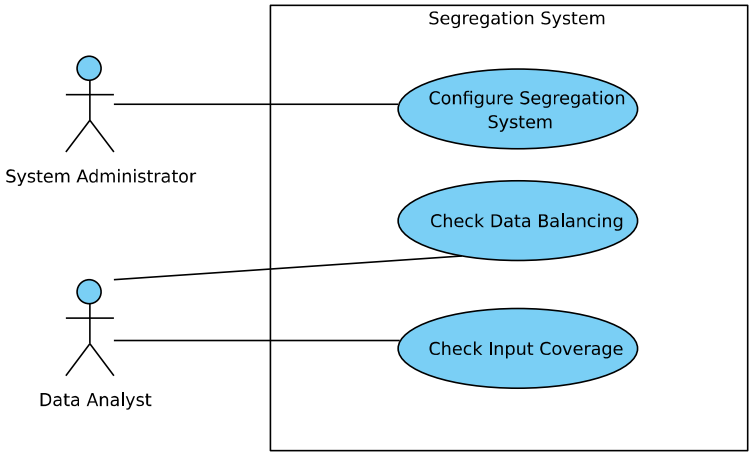
Details

Name	Value
Preconditions	Configuration for data preparation system required
Post-conditions	The Preparation System is updated and restarted

Sequence diagram of CONFIGURE <SYSTEM>



SEGREGATION SYSTEM [LORENZO]



Configure Segregation System

Scenario

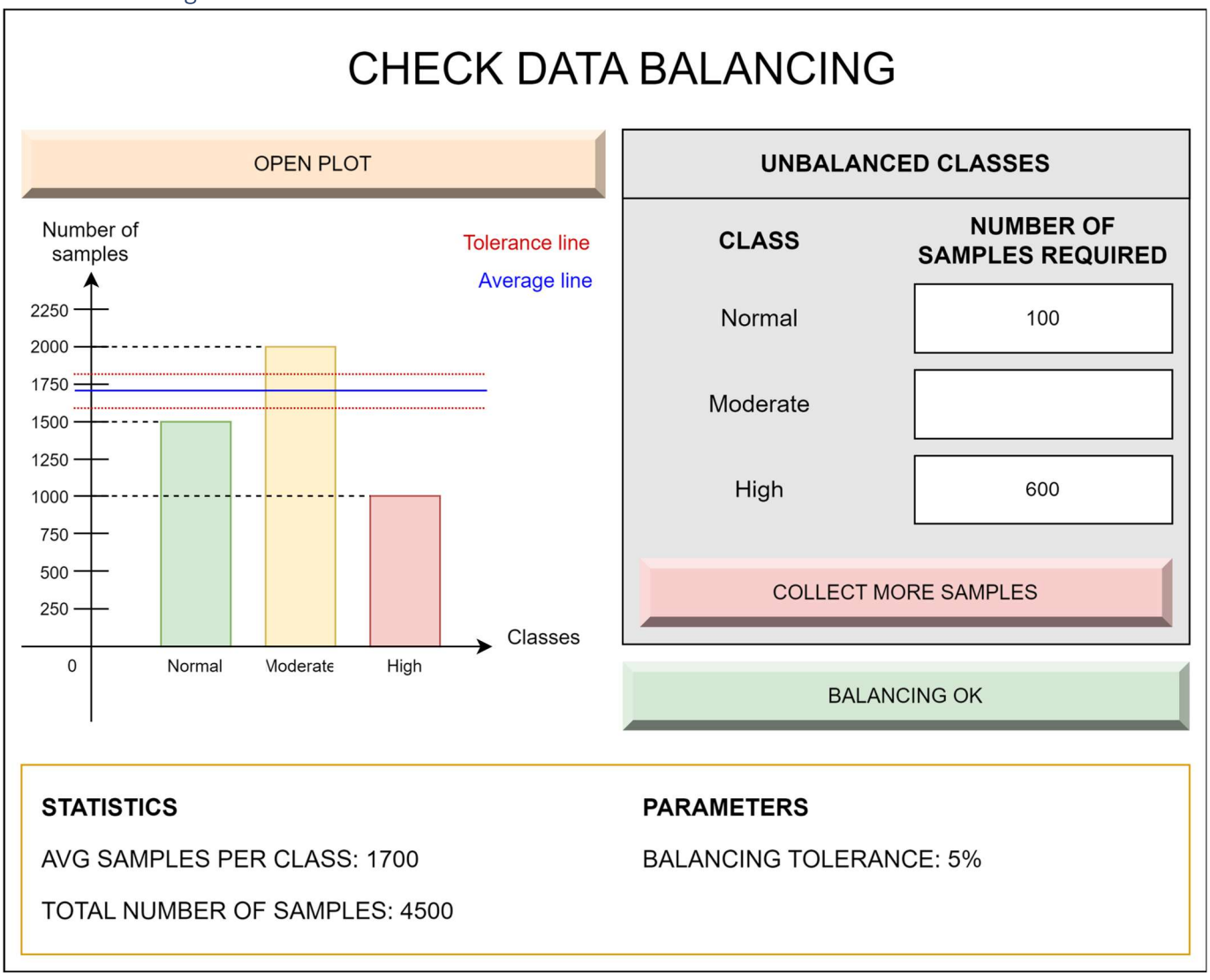
1. System Administrator launches Text Editor

2. System Administrator open configuration file (config.json)
3. System Administrator write configuration parameters in config.json
4. System Administrator save configuration file (config.json)
5. System Administrator close Text Editor

Details

Name	Value
Preconditions	A new configuration is requested for the Segregation System
Post-conditions	The Segregation System has been configured

Check Data Balancing



Scenario

1. Data Analyst requests to view the plot
2. for each class in the bar chart plot
2.1. Data Analyst checks the number of samples for that class
2.2. if number of samples is below the lower tolerance line

2.2.1. Data Analyst computes the number of required samples as (lower tolerance value - number of samples for that class)
2.2.2. Data Analyst write the number of required samples in the "number of required samples" field
end if
end for each
3. if all input classes are well balanced
3.1. Data Analyst clicks the button BALANCING OK
4. else
4.1. Data Analyst requests a new configuration clicking the button COLLECT MORE SAMPLES
end if

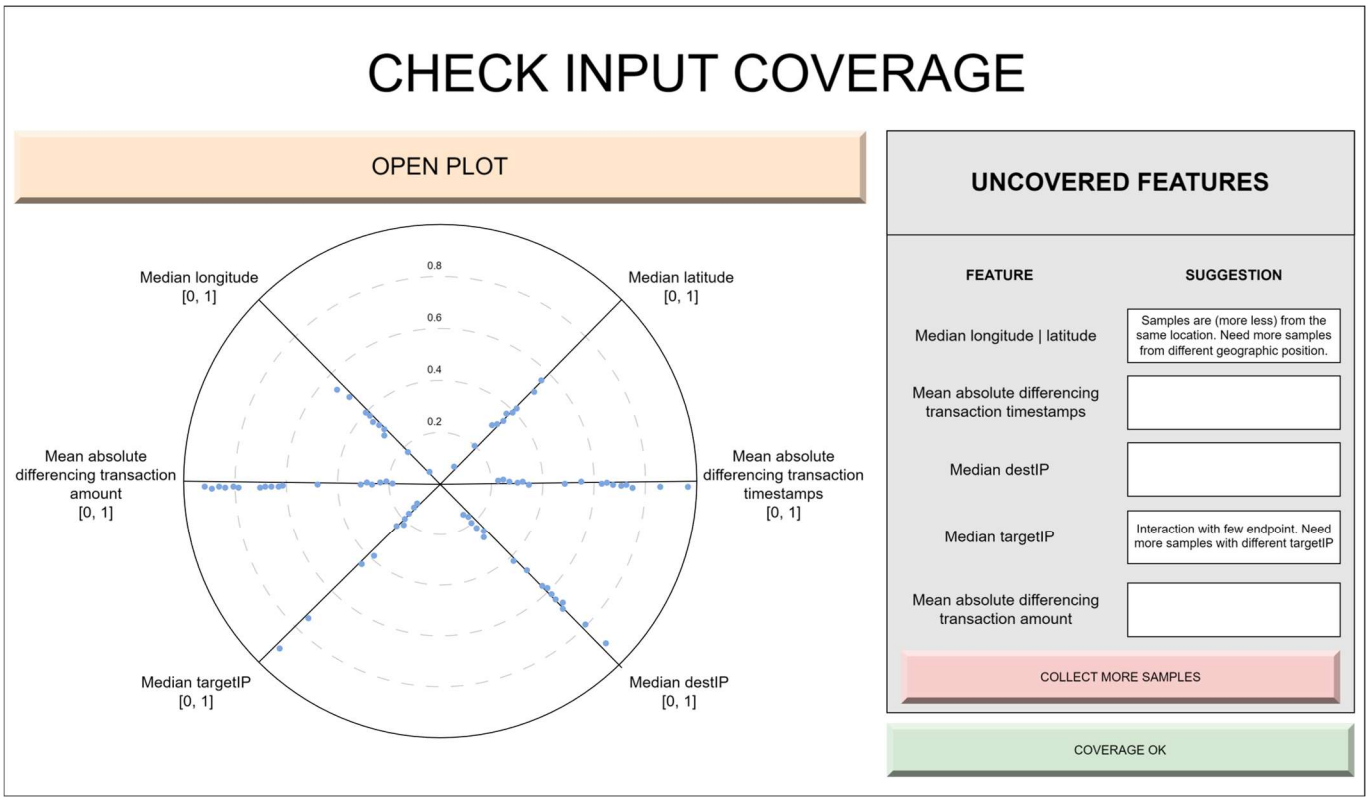
Details

Name	Value
Preconditions	1. The application has produced a plot that represent the label distribution 2. The application has stopped its execution waiting for user input
Post-conditions	Alternative flow 1: the system continue its execution Alternative flow 2: a new configuration is requested

Description

The Data Analyst needs to check the plot representing the distribution of the input labels (number of samples of each input class). The parameter to tell if classes are well balanced is the *tolerance*, in this case equal to 5%. If the number of instances of each class is different for at most 5% with respect of their average, then the classes are well balanced and the application can continue its flow. Otherwise, the Data Analyst needs to request a new configuration that ensure a good balance between the labels, specifying the number of samples for each class he needs.

Check Input Coverage



Scenario

1. Data Analyst requests to view the plot
2. for each feature in the radar plot
2.1. Data Analyst checks the data distribution
2.2. if data are not well distributed
2.2.1. Data Analyst insert a suggestion about the features that are not covered
end if
end for each
3. if all features have a good distribution
3.1. Data Analyst clicks the button COVERAGE OK
4. else
4.1. Data Analyst requests a new configuration clicking the button COLLECT MORE SAMPLES
end if

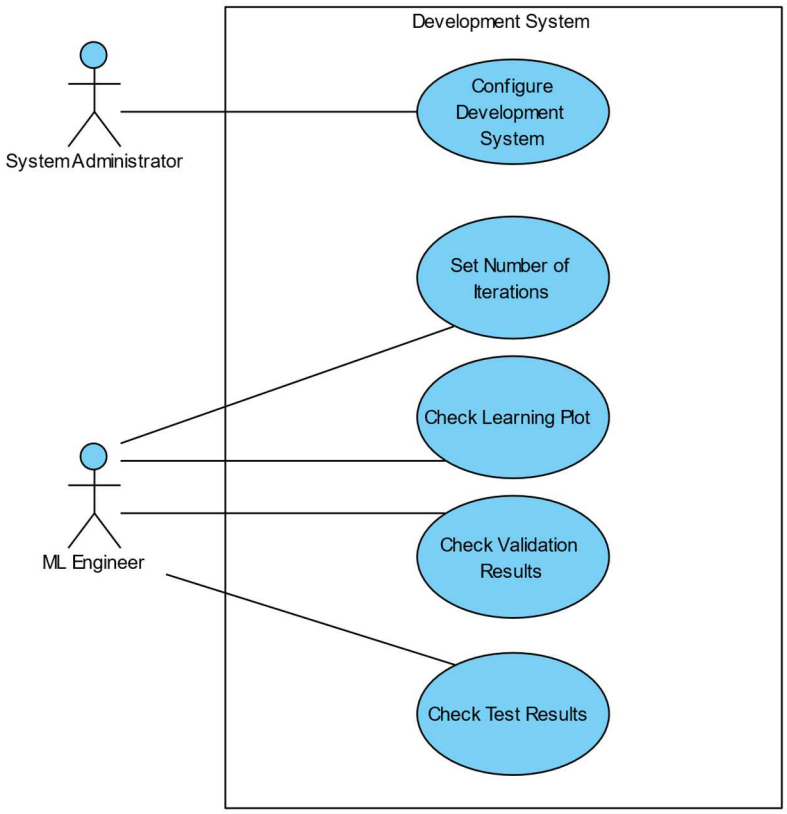
Details

Name	Value
Preconditions	1. The application has produced plot that represent input distribution 2. The application has stopped its execution waiting for user input
Post-conditions	Alternative flow 1: learning sets are generated Alternative flow 2: a new configuration is requested

Description

The Data Analyst needs to check if the input space of the network is definitively covered for each feature (by collecting other sessions, the distribution doesn't change). The input space is formed by some statistical variables: *Mean absolute differencing transaction timestamps*, *Mean absolute differencing transaction amount*, *Median longitude|latitude*, *Median targetIP* and *Median destIP*. The Data Analyst visualize a scatter radar plot. If there are many "bubbles" as the number of points (different number of sessions), then the data are well distributed and the learning sets can be generated. Otherwise, the Data Analyst needs to request a new configuration.

DEVELOPMENT SYSTEM [Nicola]

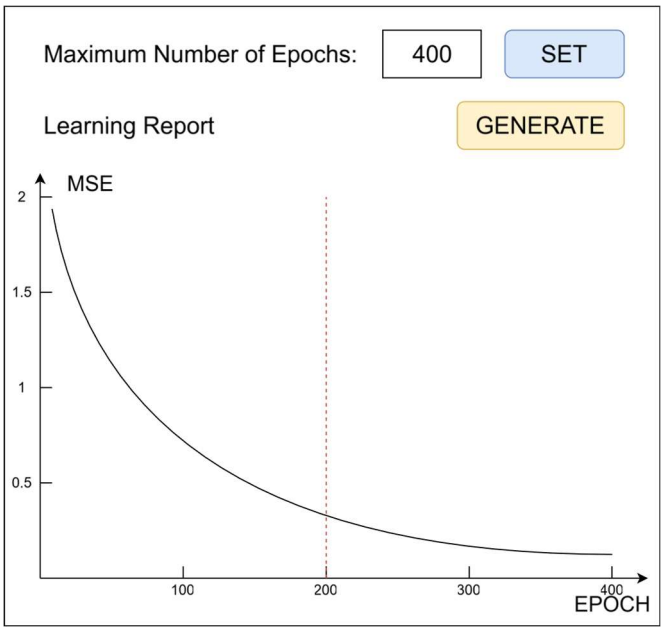


Configure Development System

1. System Administrator launches Text Editor
2. System Administrator opens the "developmentConfig.json" file
3. System Administrator writes or updates the "developmentConfig.json" file with new configuration parameters.
4. System Administrator saves the updated configuration file.
5. System Administrator closes the Text Editor.

Set Number of Iterations

ML Engineer determines the number of epochs for next training. If it is the first time for a specific classifier, they will choose from their experience, otherwise a learning curve of a preceding training is present, and ML Engineer will choose by analyzing the curve.
If the curve is flat after half of the iterations, the previous number of epochs is too high and will be reduced.
If the curve is too steep a the end, the previous number is too low and will be incremented.



Scenario

1. if there isn't a learning curve
1.1. ML Engineer writes in dedicated text block a number chosen by experience
2. else
2.1. ML Engineer looks at learning curve
2.2. ML Engineer check loss value after half the maximum number of iterations
2.3. if the loss curve is flat after half the number of epochs
2.3.1. ML Engineer reduces by a third the number in dedicated text block
2.4. else
2.4.1. ML Engineer checks the curve steepness at the end of the iterations
2.4.2 if the curve is too steep
2.4.2.1. ML Engineer enlarges by a third the number in dedicated text block
end if
end if
end if
3. ML Engineer clicks **SET**
4. SYSTEM saves the number in dedicated text block as maximum number of iterations.

Check Learning Plot

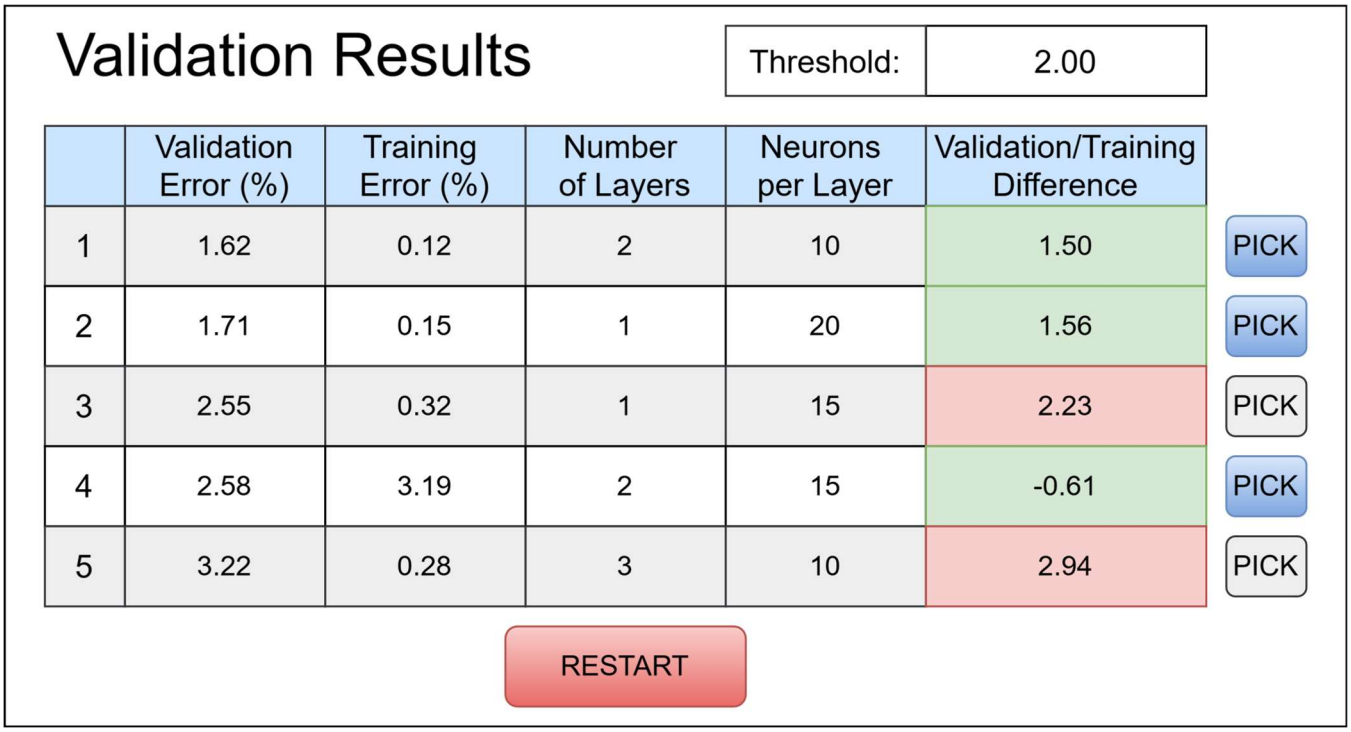
Scenario

1. ML Engineer clicks on **GENERATE**
2. SYSTEM trains a classifier and shows learning curve

Check Validation Results

ML Engineer chooses one of the hyperparameters configurations that were validated in a grid search. To do that, they must check the difference between validation and training error to avoid models that overfit (or underfit). Among the models that have that difference under a specific threshold, the Engineer should consider the two having the smallest validation

error. If these two models have a comparable error (in the same order of magnitude), ML Engineer should choose the less complex, otherwise simply the one with smaller error.



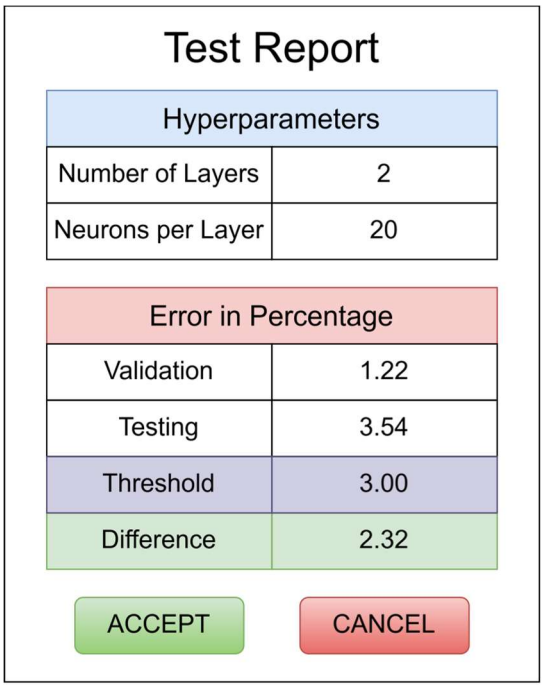
Scenario

1. ML Engineer starts a grid search
2. SYSTEM performs the grid search and shows a table with 5 best fitted models in ascending order of validation error
3. for each model in the table
3.1. ML Engineer checks the absolute value of the difference in validation and training error
end for each
4. if all models have a difference over a specified threshold
4.1. ML Engineer clicks **RESTART**
4.2. SYSTEM deletes all results
5. else if only one model is valid
5.1. ML Engineer clicks **PICK** button next to that model
5.2. SYSTEM saves model parameters on a file
6. else
6.1. ML Engineer calculate difference in validation error between first two valid models
6.2. ML Engineer calculate complexity of first two valid models
6.3. if the difference is an order of magnitude smaller than the error values AND the second model is less complex than the first
6.3.1. ML Engineer clicks **PICK** button next to second model
6.4. else
6.4.1. ML Engineer clicks **PICK** button next to first model
end if
6.5. SYSTEM saves model parameters on a file

end if

Check Test Results

ML Engineer looks at the testing results of the model selected from validation. If the difference between validation and testing error is under a prespecified generalization tolerance, then the model is good, and ML Engineer will click ACCEPT to declare a successful termination of development. Otherwise, they will issue the system to restart from scratch.



Scenario

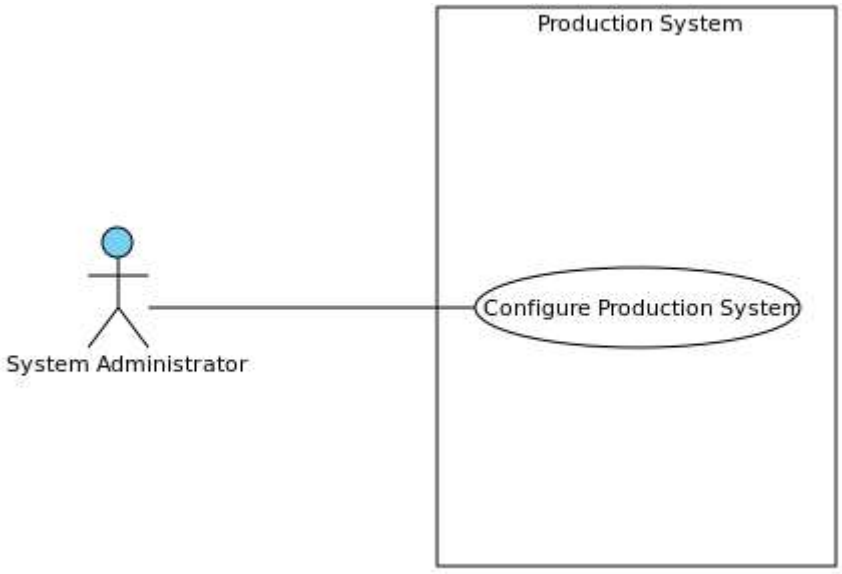
1. ML Engineer starts a test
2. SYSTEM executes the test using saved model and shows results, including hyperparameters values, test error, previous validation error and their difference
3. ML Engineer looks at the difference in validation and test error
4. if the difference is under the specified generalization tolerance threshold
4.1. ML Engineer clicks **ACCEPT** button
4.2. SYSTEM saves model parameters and flags it as ready for production
5. else
5.1. ML Engineer clicks **CANCEL** button
5.2. SYSTEM deletes all progress and restarts development
end if

Details

Name	Value
Preconditions	A file containing model hyperparameters must exist.
Postconditions	1. ML Engineer waits for a new training session and its new result. 2. SYSTEM starts deployment of the classifier.

PRODUCTION SYSTEM [ENRICO]

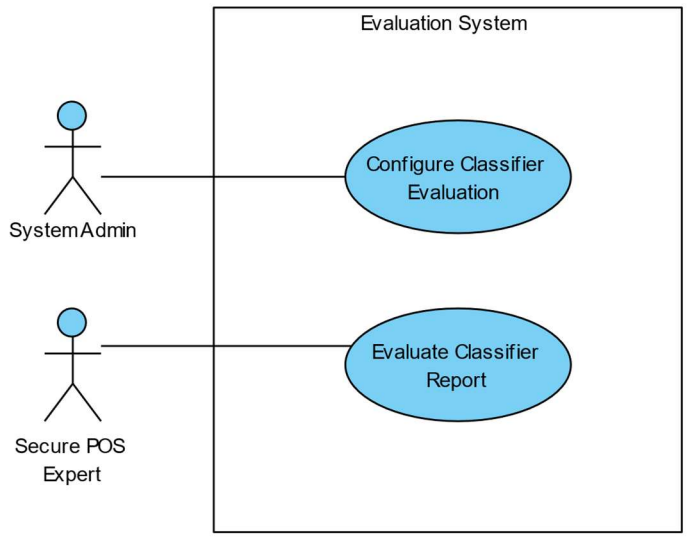
Configure Production System



1. System Administrator launches Text Editor
2. System Administrator opens the ”productionConfig.json” file
3. System Administrator writes or updates the ”productionConfig.json” file with new configuration parameters.
4. System Administrator saves the updated configuration file.
5. System Administrator closes the Text Editor.

Name	Value
Preconditions	1. A new configuration has been requested by Production System.
Post-conditions	1. The Production System has been configured.

EVALUATION SYSTEM [JACOPO]



Configure Classifier Evaluation

Details

Name	Value
Preconditions	1. The System Administrator has created a file called "config.json", and the corresponding validation scheme for this .json file. 2. A request has arrived for a new configuration of the monitoring system.
Post-conditions	1. The monitoring system has been newly configured.

Scenario

1.The System Administrator launches a test editor, able to open .json files .
2.The System Administrator opens the file "config.json".
3.The System Administrator writes the new configuration parameters into the config.json file, in json format, with respect of the specific validation schema.
4.The System Administrator saves the changes to the file "config.json".
5.The System Administrator closes the text editor that has been usin.
6.The System Administrator launches the Evaluation System.

Evaluate Classifier [JACOPO]

Expert label class	Anomaly detector label class	Result
1	2	✗
2	2	✓
3	3	✓
3	1	✗
4	5	✗
3	4	✗
2	2	✓

Ideal Max number of errors tolerated (thesis 1) :	5
Ideal Max number of consecutive errors tolerated (thesis 2) :	2

Total errors measured : 4

Max consecutive errors measured : 3

thesis1 : satisfied (4<5)

thesis2 : exceeded (3!< 2)

Scenario

1.The Secure POS Expert launches a text editor able to open .json files.
2.The Secure POS Expert opens the file "report.json".
3.The Secure POS Expert looks at the field containing a list of theses, and their satisfaction status:
3.1. If any one of the theses has been exceeded, then the classifier has failed its expectations.
3.1.1. Secure POS Expert must therefore send a new training request, therefore a new configuration request.
3.2. If all the theses have been satisfied, the classifier is good, and nothing must be sent.
4.The Secure POS Expert closes the text editor.

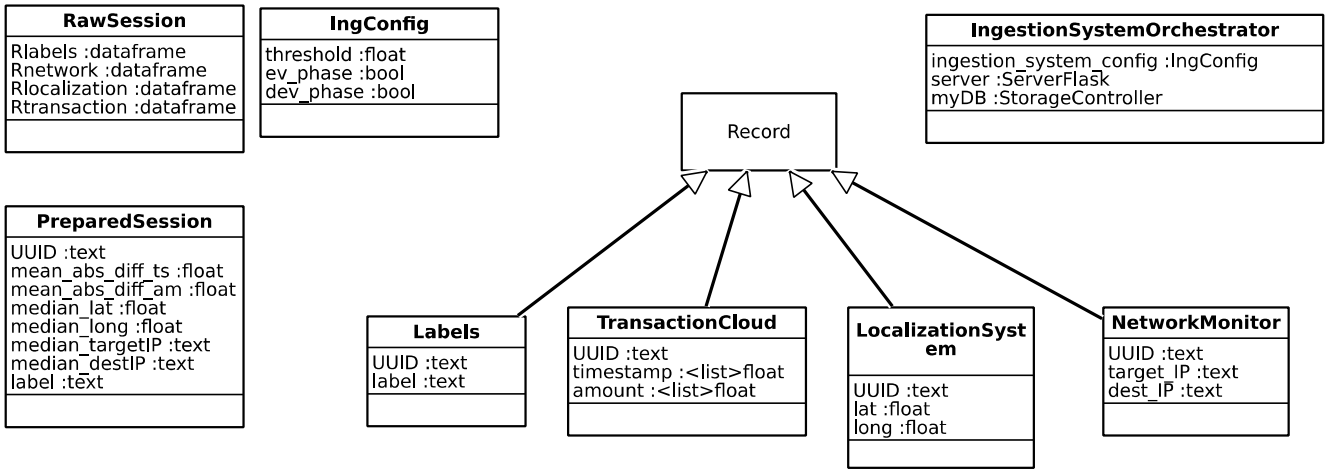
Details

Name	Value
Preconditions	1. The EVALUATION SYSTEM has generated an evaluation report, and has saved it in a "report.json" file.
Post-conditions	1. The evaluation report has been evaluated by the expert.

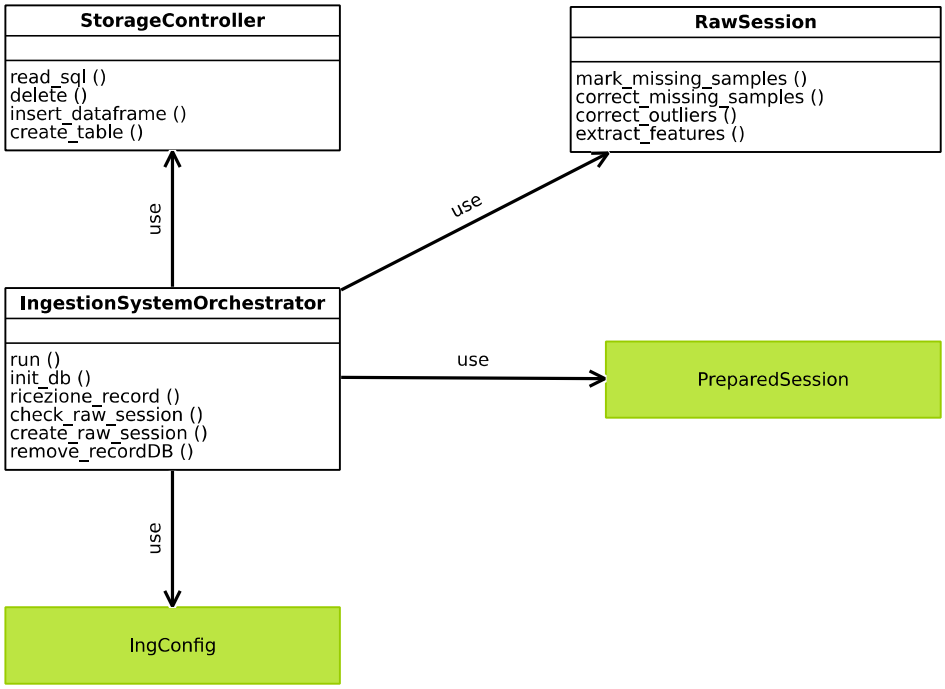
ANALYSIS – MODEL AND SEQUENCE DIAGRAM

PREPARATION PROCESS [MATTEO]

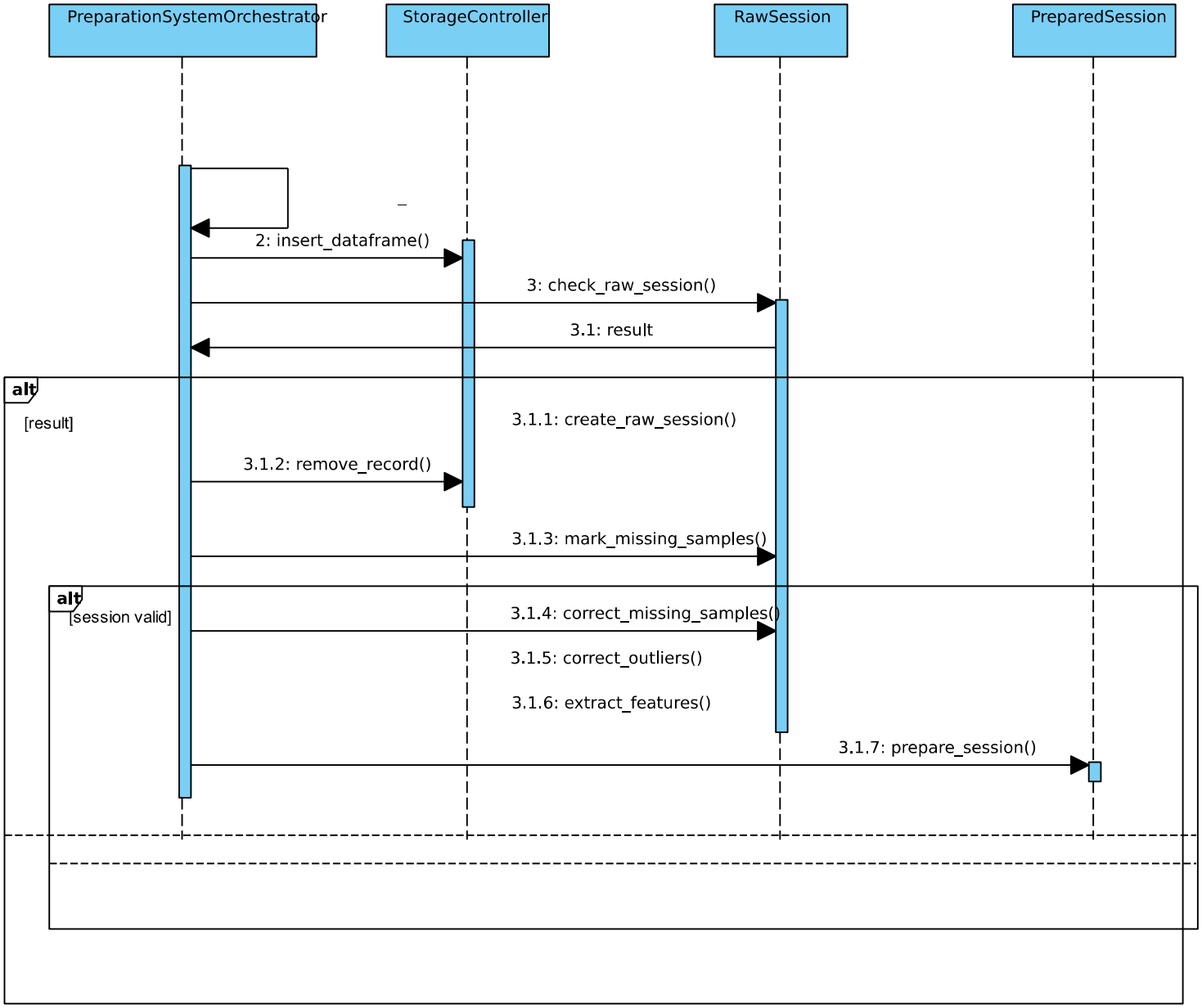
DATA MODEL



APPLICATION LOGIC

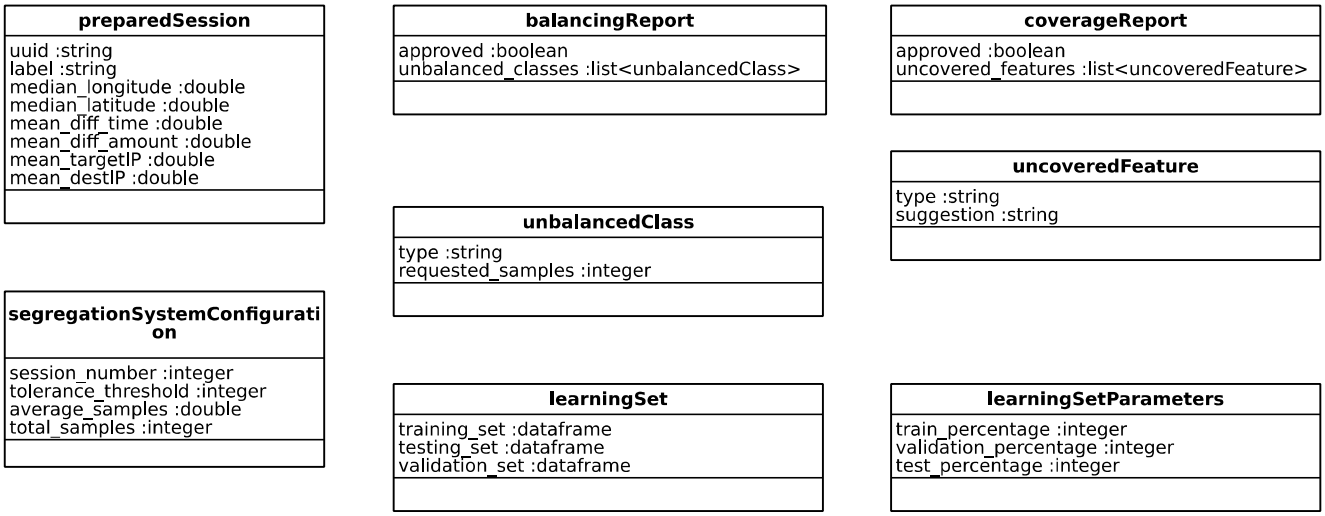


SEQUENCE DIAGRAM

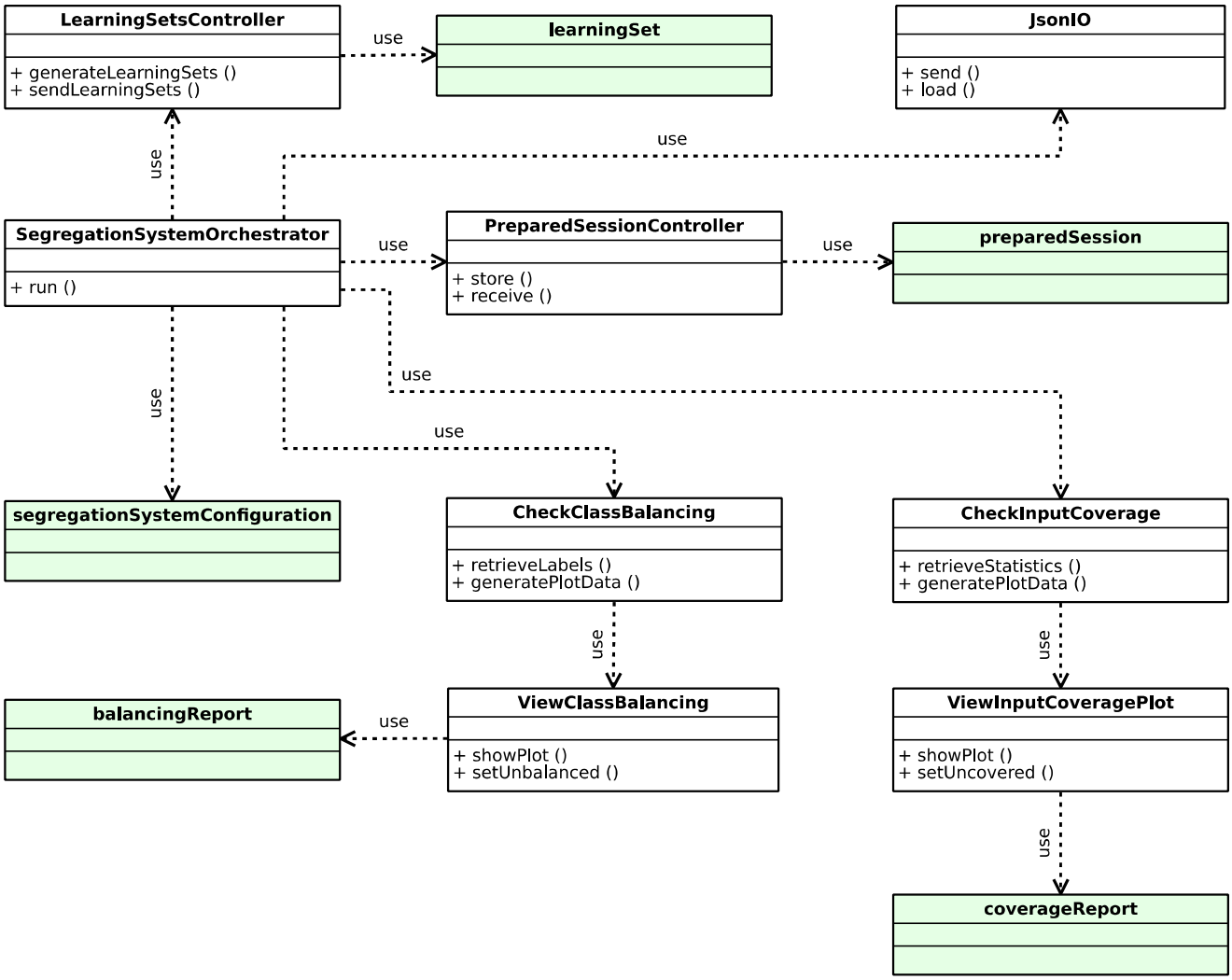


GENERATE LEARNING SETS PROCESS [LORENZO]

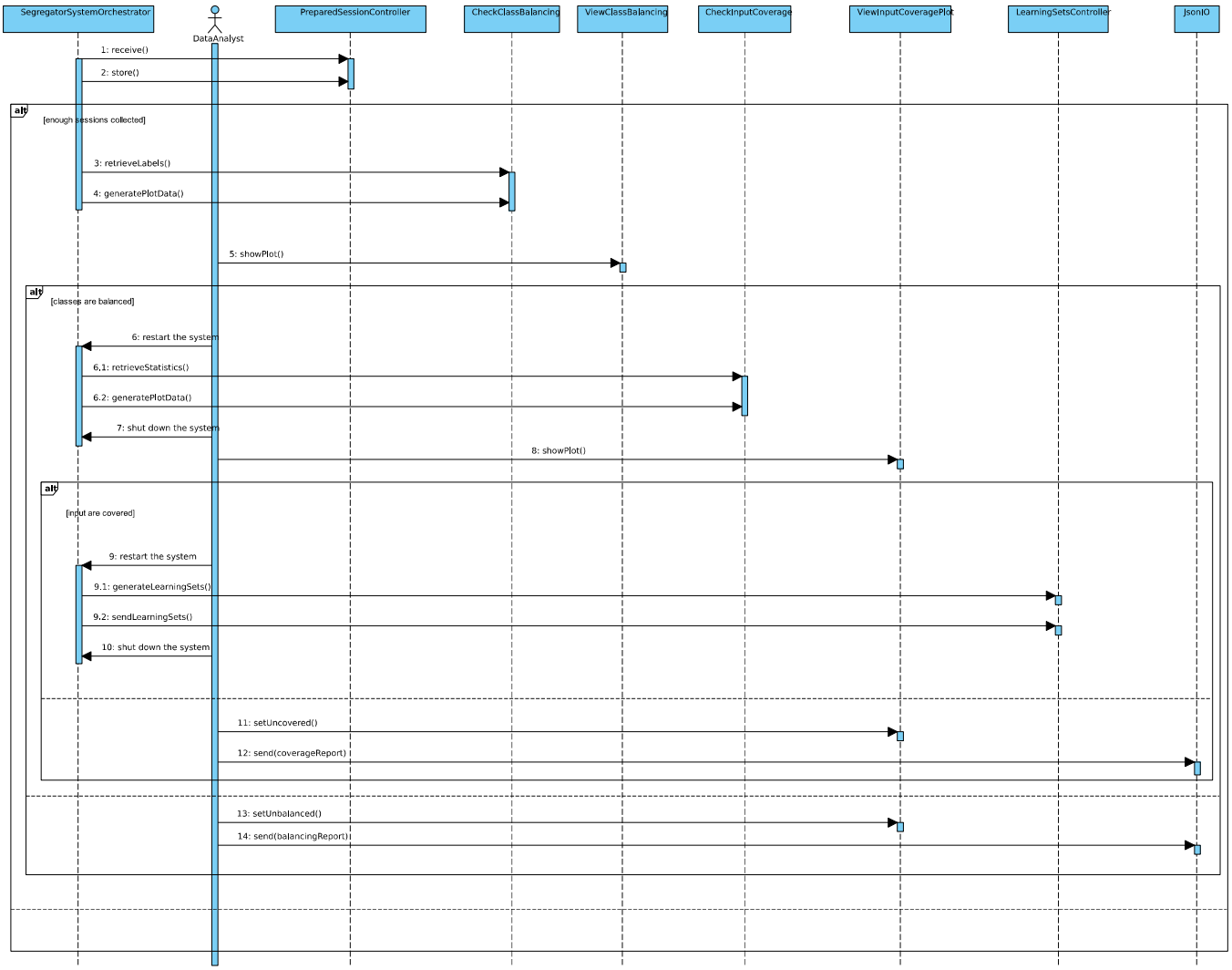
DATA MODEL



APPLICATION LOGIC

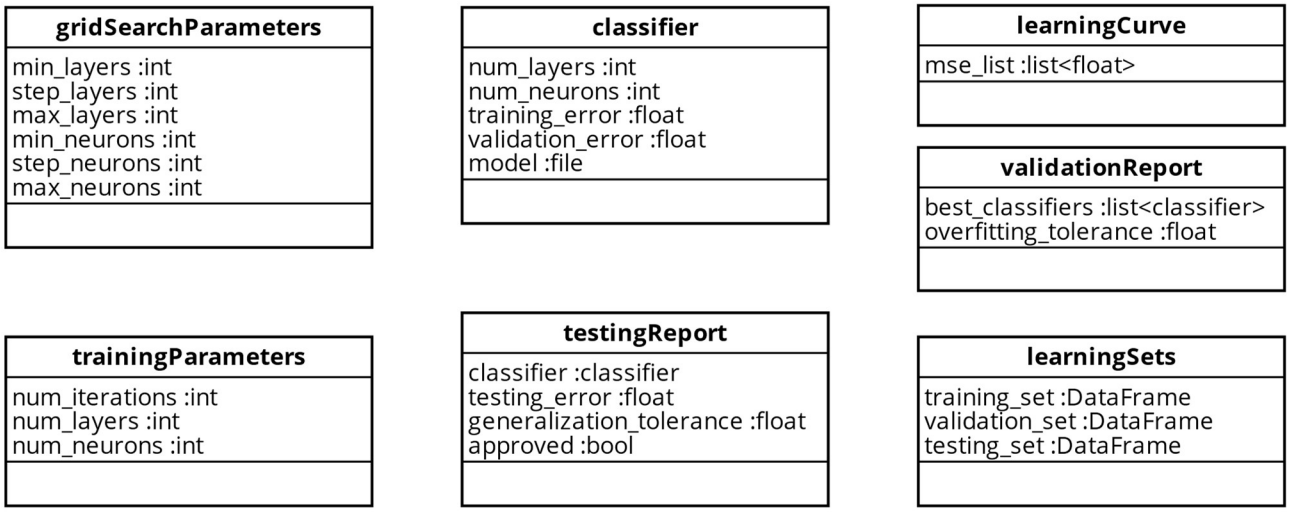


SEQUENCE DIAGRAM

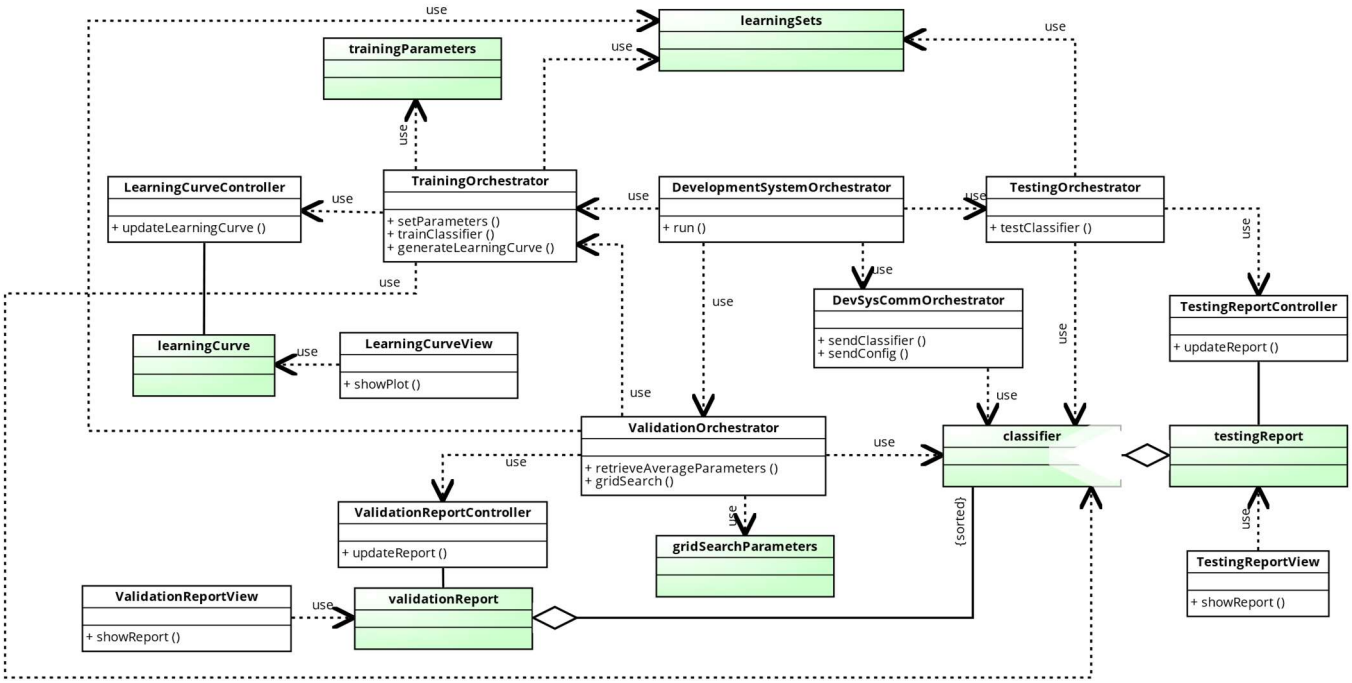


DEVELOP CLASSIFIER [NICOLA]

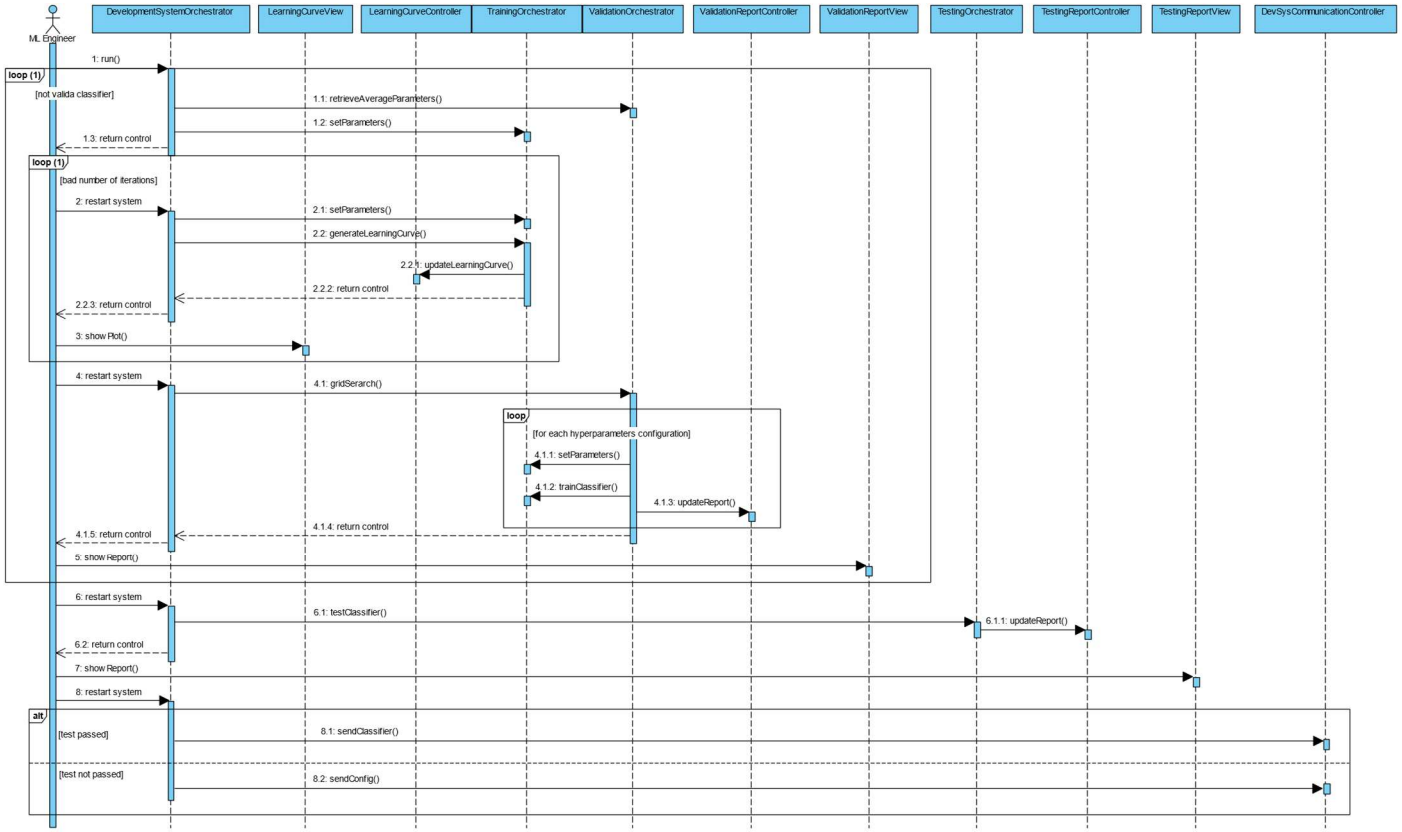
DATA MODEL



APPLICATION LOGIC

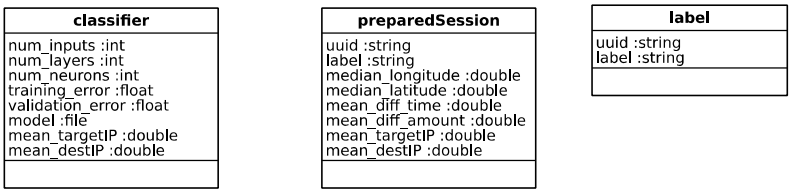


SEQUENCE DIAGRAM

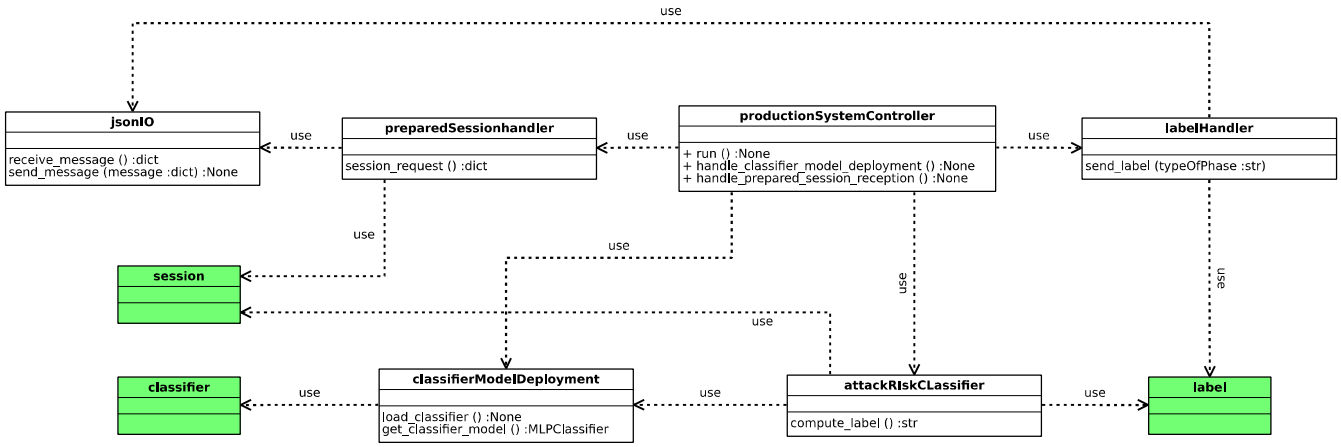


CLASSIFY SESSION [ENRICO]

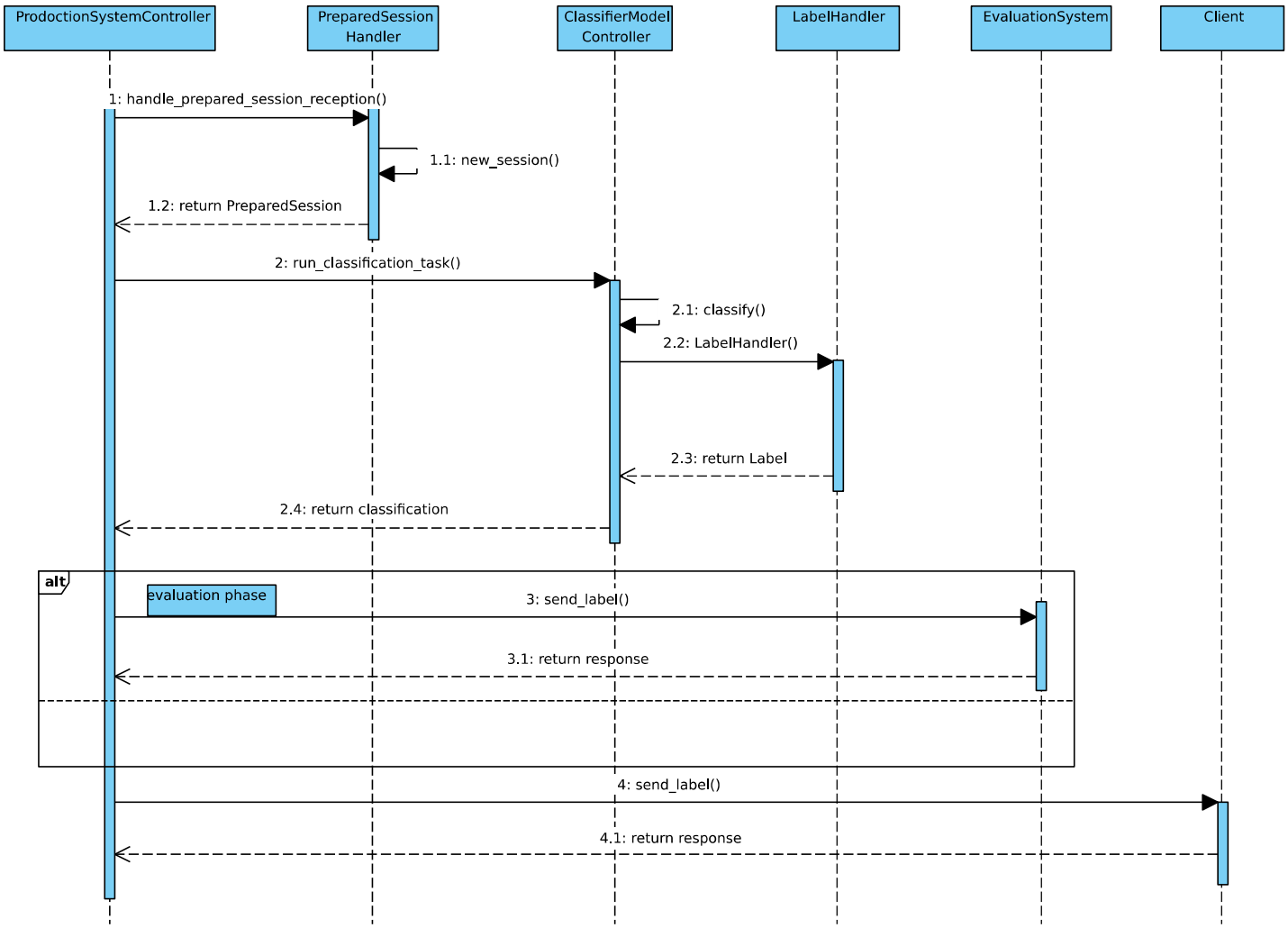
DATA MODEL



APPLICATION LOGIC

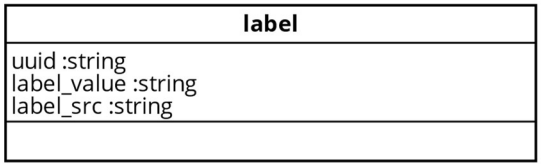


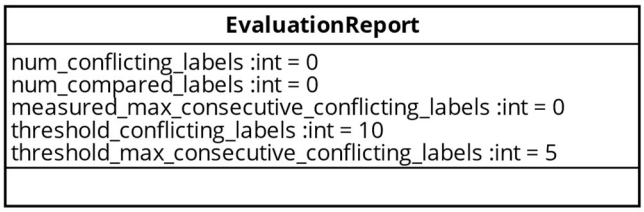
SEQUENCE DIAGRAM

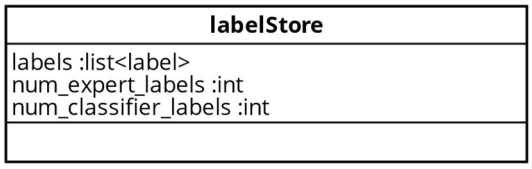


EVALUATE CLASSIFIER [JACOPO]

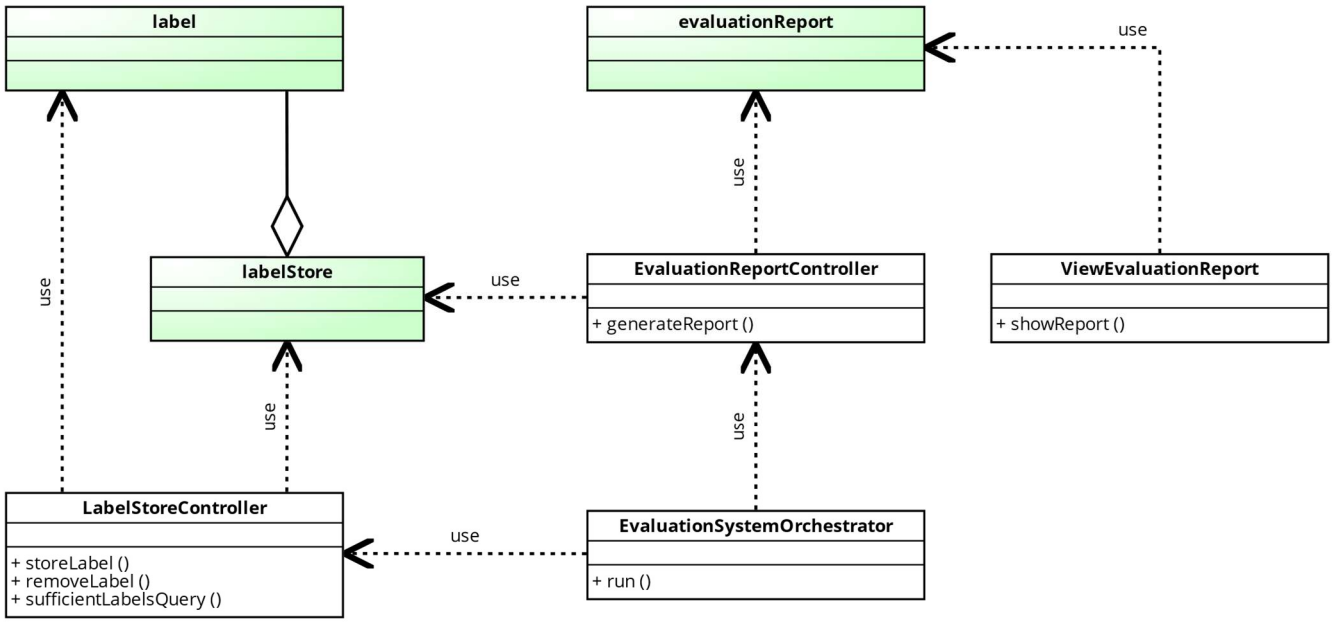
DATA MODEL



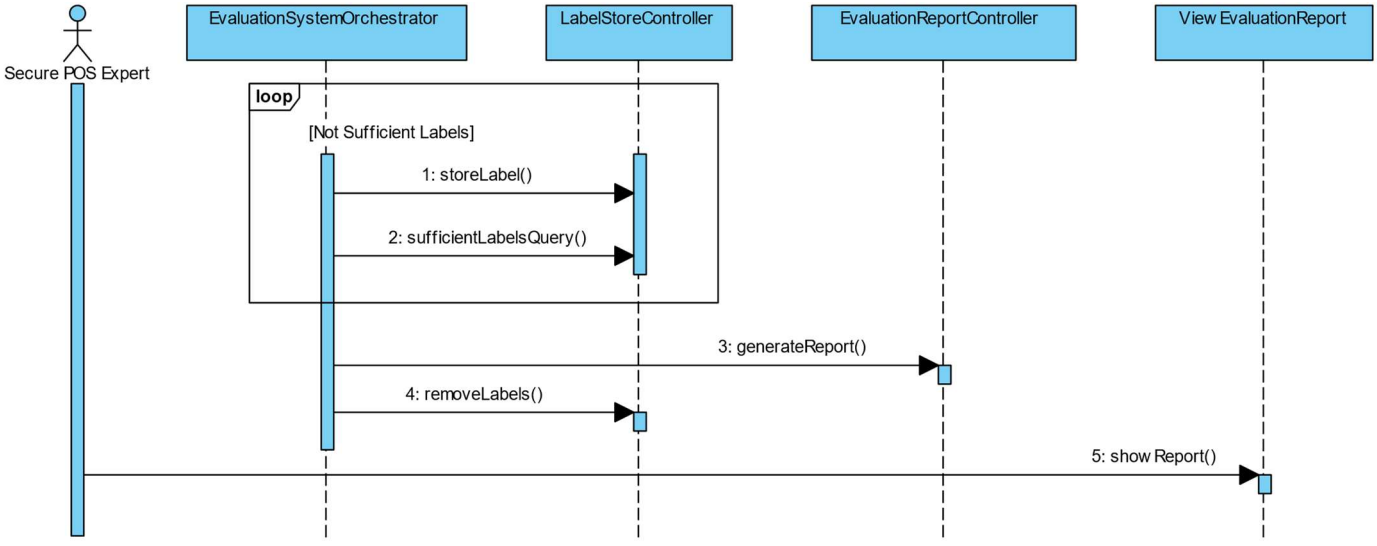




APPLICATION LOGIC



SEQUENCE DIAGRAM



DESIGN WORKFLOW

UTILITY CLASSES

Communications

ServerREST
api :Api app :Flask
run (:host, :port, :debug)

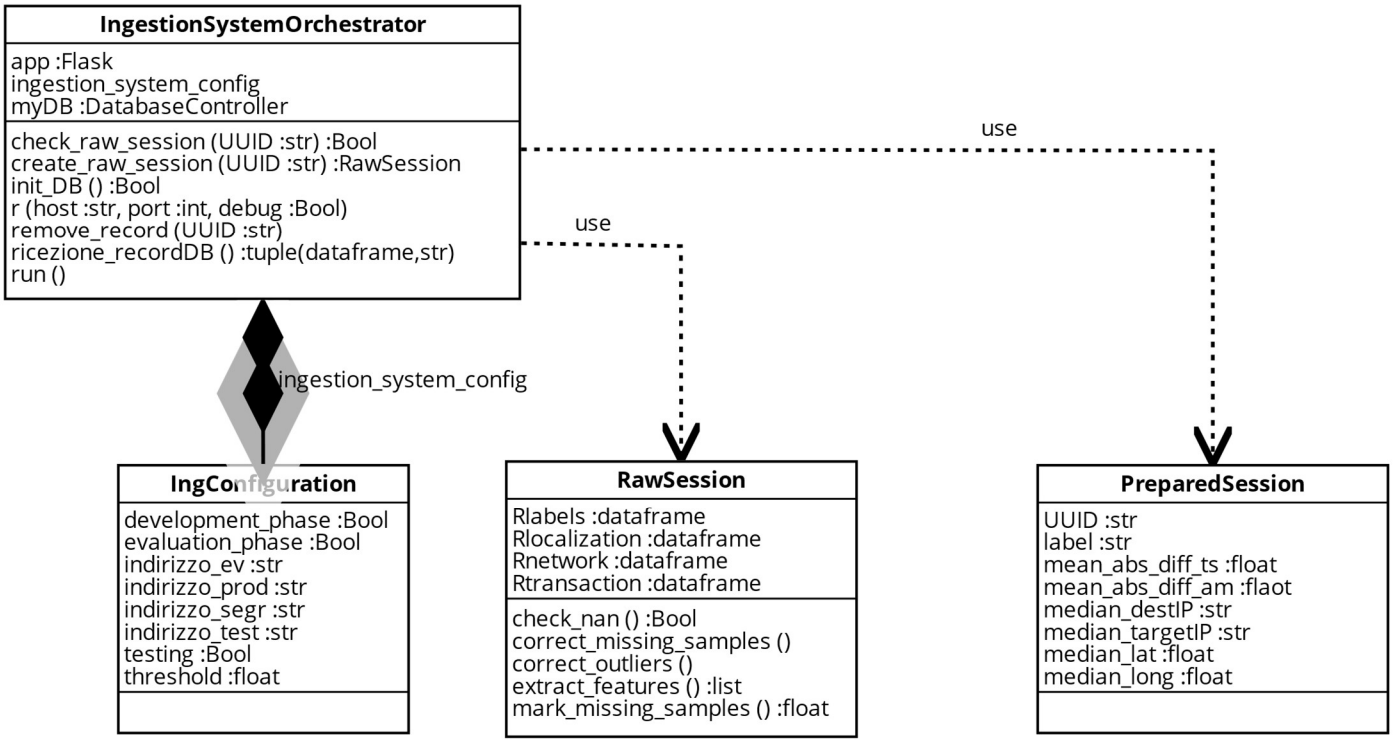
FileReceptionAPI
filepath :str
post ()

ReceiveJsonApi
handle_request :Optional[Callable[[dict],None] json_path_schema :str
post ()

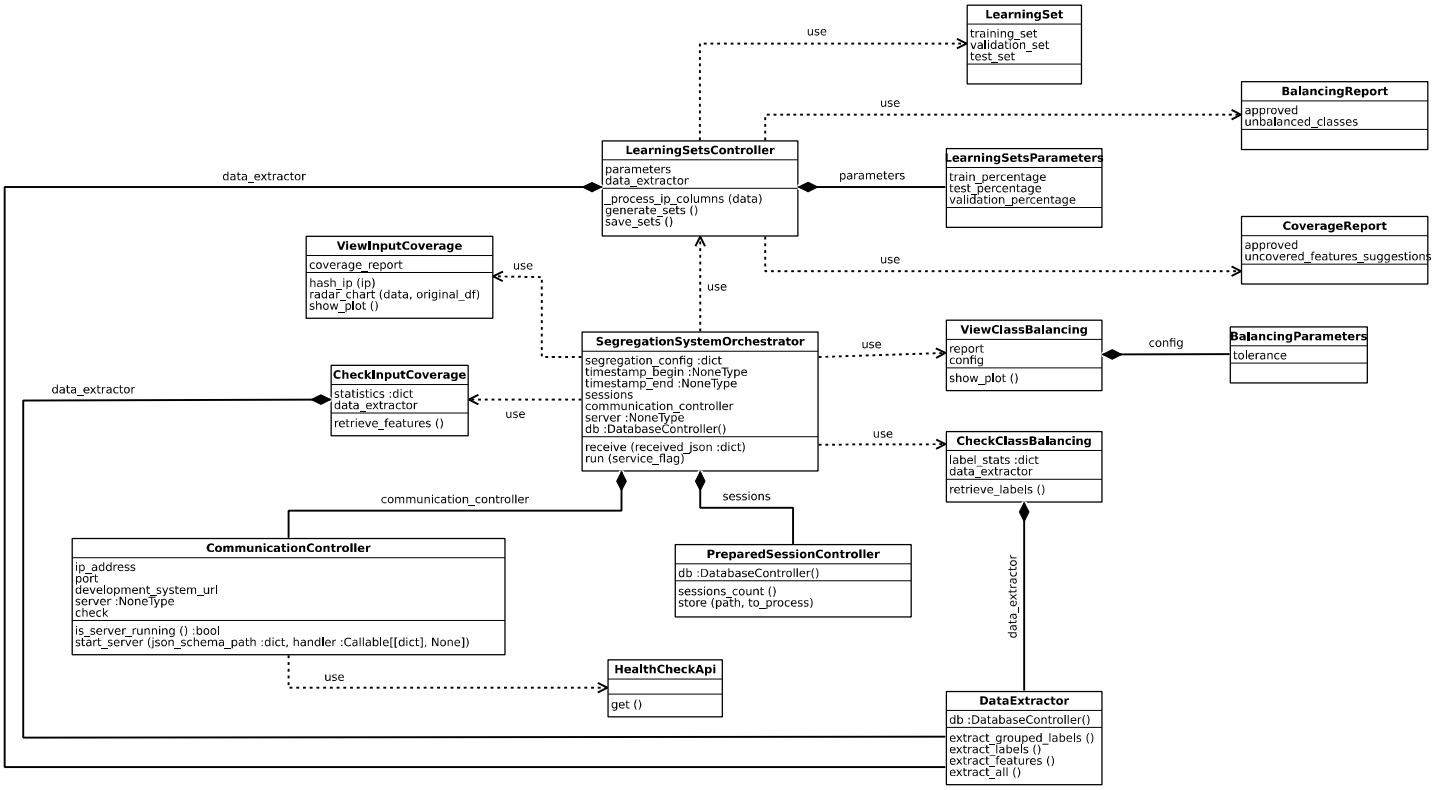
Database Controller

DatabaseController
create-table (query :str, params :list) :bool delete (query :str, params :list) :bool drop_database () drop_table (table :str) :bool insert_dataframe (dataframe :pd.DataFrame, table :str) :bool read_sql (query :str, params :list) update (query :str, params :list) :bool

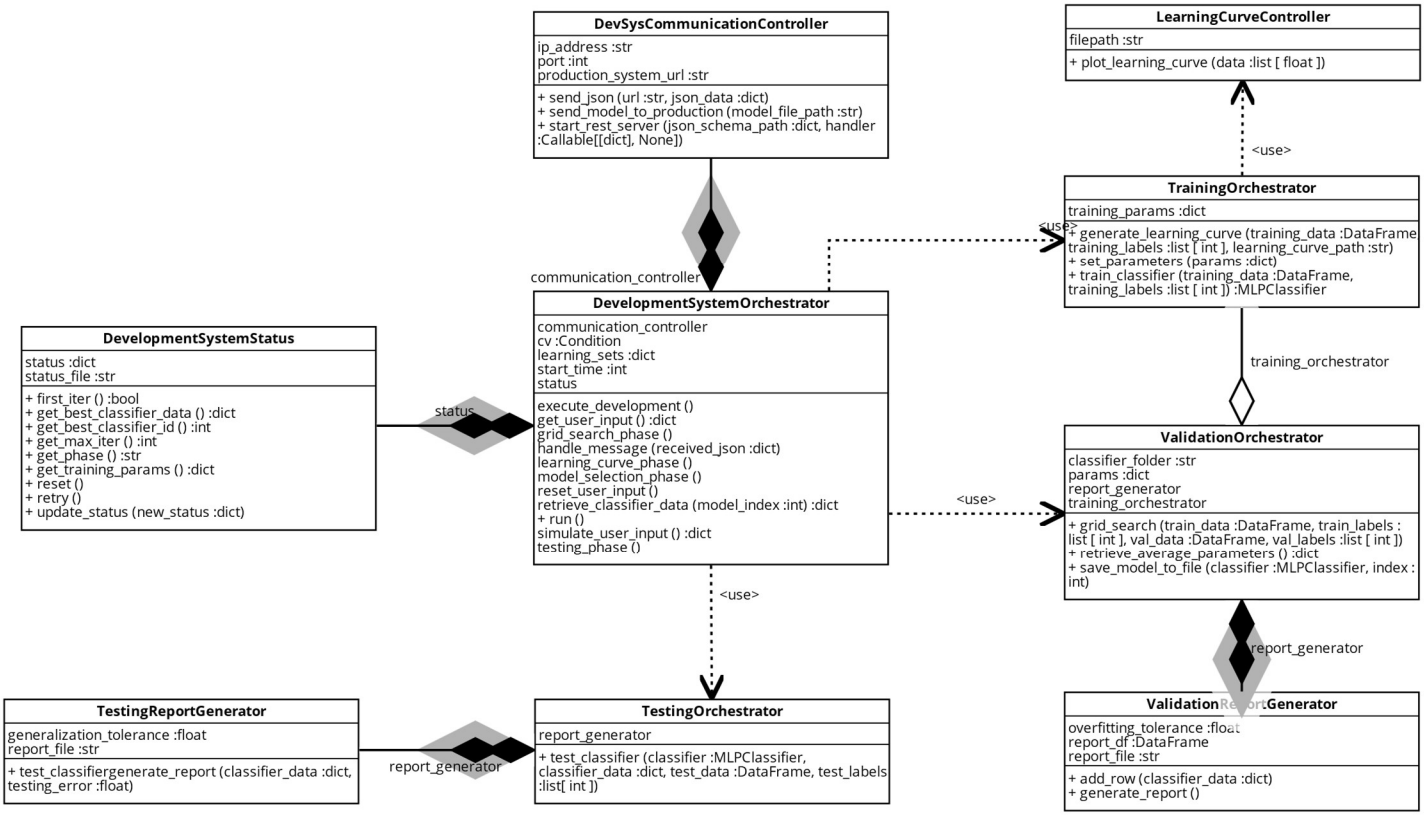
PREPARE/INGESTION SYSTEM [MATTEO]



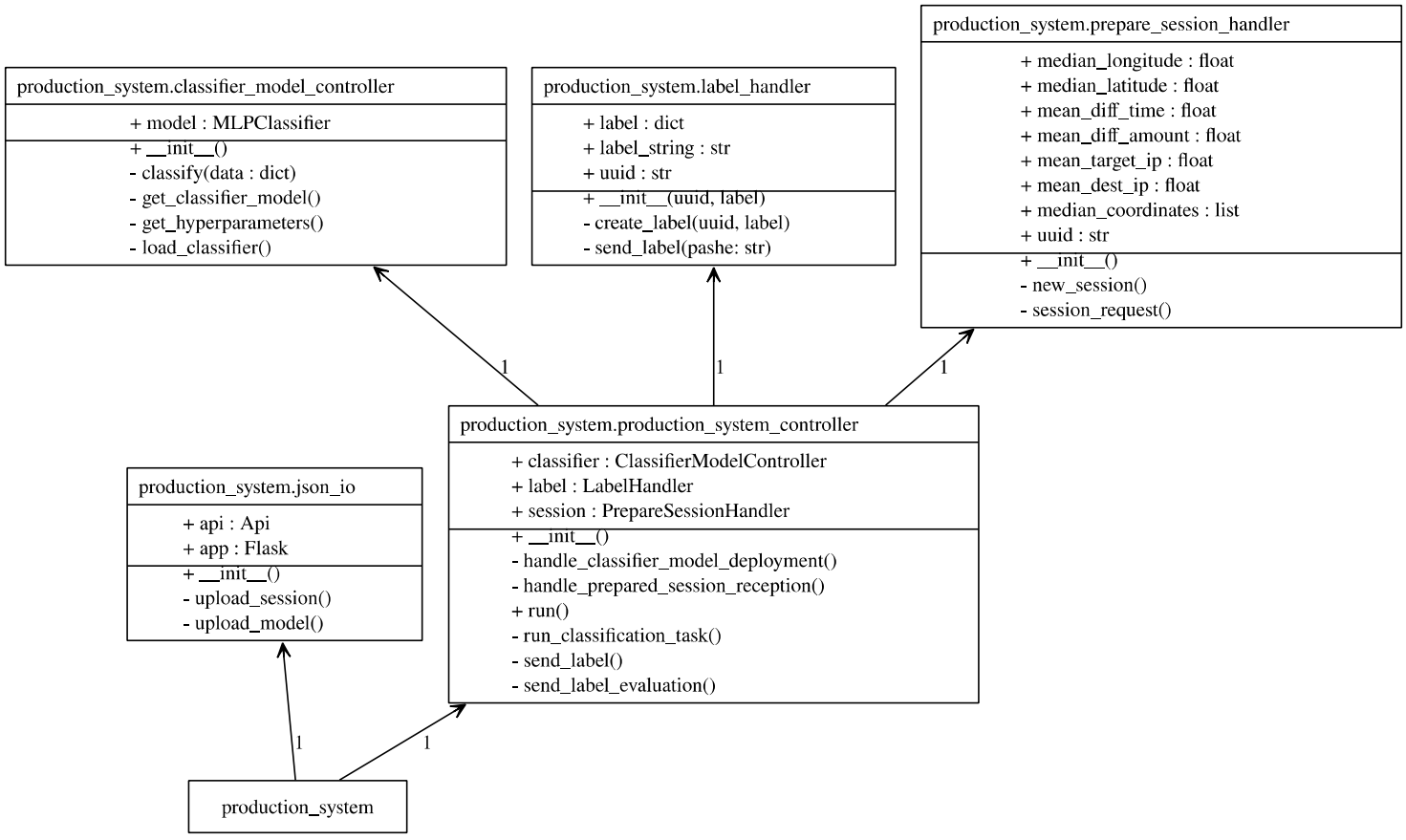
SEGREGATION SYSTEM [LORENZO]



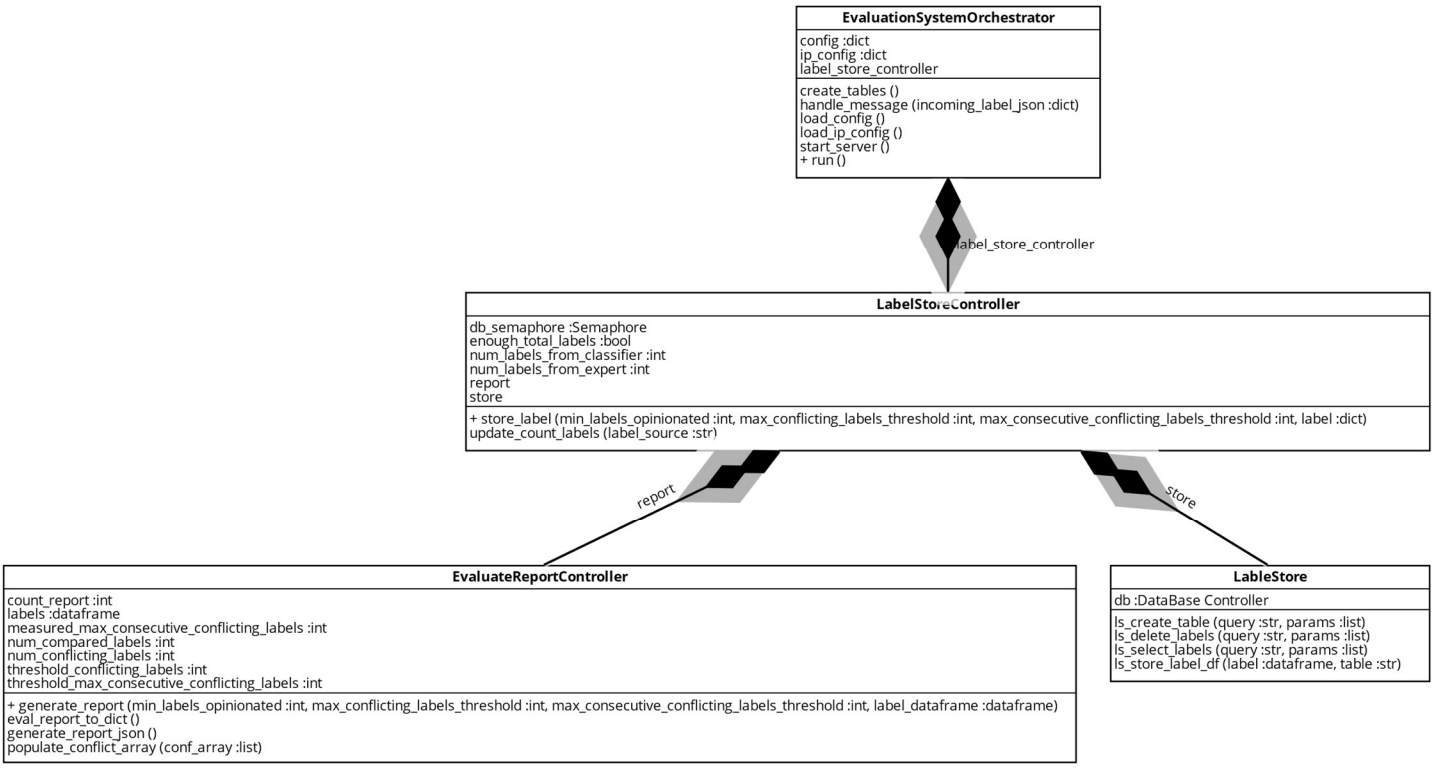
DEVELOPMENT SYSTEM [NICOLA]



PRODUCTION SYSTEM [ENRICO]



EVALUATION SYSTEM [JACOPO]



PRODUCTIVITY EVALUATION

NON-AUTOMATION

Cost Calculation: (iterations *) Probability * Automation-Cost * Relative-Salary

Base reference salary:

Employee	Salary (per year)	Relative salary
System Administrator	$ 94,356	1
Machine Learning Data Analyst	$ 107,000	1.13
Machine Learning Engineer	$ 166,833	1.77
Senior Cybersecurity Consultant (Secure POS Expert)	$ 201,000	2.13

PREPARE/INGESTION SYSTEM [MATTEO]

PREPARE/INGESTION SYSTEM: CONFIGURE

STEP	Cost Calculation	Salary Calculation
1. System Administrator launches Text Editor	1 x 1 x 1	1
2. System Administrator open configuration file (config.json)	1 x 1 x 1	1
3. System Administrator write configuration parameters in config.json	1 x 1 x 1	1
4. System Administrator save configuration file (config.json)	1 x 1 x 1	1
5. System Administrator close Text Editor	1 x 1 x 1	1
TOTAL		5

SEGREGATION SYSTEM [LORENZO]

SEGREGATION SYSTEM: CONFIGURE

STEP	Cost Calculation	Salary Calculation
1. System Administrator launches Text Editor	1 * 1 * 1	1
2. System Administrator open configuration file (config.json)	1 * 1 * 1	1
3. System Administrator write configuration parameters in config.json	1 * 1 * 1	1
4. System Administrator save configuration file (config.json)	1 * 1 * 1	1
5. System Administrator close Text Editor	1 * 1 * 1	1
TOTAL		5

CHECK DATA BALANGING

STEP	Cost Calculation	Salary Calculation
1. Data Analyst requests to view the plot	1 * 1 * 1.13	1.13
2. for each class in the bar chart plot		
2.1. Data Analyst checks the number of samples for that class	(3 *) 1 * 3 * 1.13	10.17
2.2. if number of samples is below the lower tolerance line		
2.2.1. Data Analyst computes the number of required samples as (lower tolerance value - number of samples for that class)	(3 * 0.1) * 3 * 1.13	1.02
2.2.2. Data Analyst write the number of required samples in the "number of required samples" field	(3 * 0.1) * 3 * 1.13	1.02
end if		
end for each		
3. if all input classes are well balanced		
3.1. Data Analyst clicks the button BALANCING OK	(0.73) * 1 * 1.13	0.95
4. else		
4.1. Data Analyst requests a new configuration clicking the button COLLECT MORE SAMPLES	(0.27) * 1 * 1.13	0.31
end if		
TOTAL		15.24

CHECK INPUT COVERAGE

STEP	Cost Calculation	Salary Calculation
1. Data Analyst requests to view the plot	1 * 1 * 1.13	1.13
2. for each feature in the radar plot		
2.1. Data Analyst checks the data distribution	6 * 4 * 1.13	27.12
2.2. if data are not well distributed		
2.2.1. Data Analyst insert a suggestion about the features that are not covered	(6 * 0.1) * 4 * 1.13	2.71

end if		
end for each		
3. if all features have a good distribution		
3.1. Data Analyst clicks the button COVERAGE OK	(0.53) * 1 * 1.13	0.60
4. else		
4.1. Data Analyst requests a new configuration clicking the button COLLECT MORE SAMPLES	(0.47) * 1 * 1.13	0.53
end if		
TOTAL		32.09

DEVELOPMENT SYSTEM [NICOLA]

DEVELOPMENT SYSTEM: CONFIGURE

Step	Cost Calculation	Salary Calculation
1. System Administrator launches Text Editor	(3 *) 1 * 1 * 1	3
2. System Administrator opens the configuration file	(3 *) 1 * 1 * 1	3
3. System Administrator writes or updates the configuration files with new configuration parameters.	(3 *) 1 * 1 * 1	3
4. System Administrator saves the updated configuration file.	(3 *) 1 * 1 * 1	3
5. System Administrator closes the Text Editor.	(3 *) 1 * 1 * 1	3
TOTAL		15

SET NUMBER OF ITERATIONS

Step	Cost Calculation	Salary Calculation
1. if there isn't a learning curve	0	0
1.1. ML Engineer writes in dedicated text block a number chosen by experience	0.4 * 4 * 1.77	2.832
2. else	0	0
2.1. ML Engineer looks at learning curve	0.6 * 0 * 1.77	0
2.2. ML Engineer check loss value after half the maximum number of iterations	0.6 * 4 * 1.77	4.248
2.3. if the loss curve is flat after half the number of epochs	0	0
2.3.1. ML Engineer reduces by a third the number in dedicated text block	0.15 * 3 * 1.77	0.7965
2.4. else	0	0
2.4.1. ML Engineer checks the curve steepness at the end of the iterations	0.45 * 4 * 1.77	3.186
2.4.2 if the curve is too steep	0	
2.4.2.1 ML Engineer enlarges by a third the number in dedicated text block end if	0.15 * 3 * 1.77	0.7965

end if end if		
3. ML Engineer clicks **SET**	1 * 1 * 1.77	1.77
4. SYSTEM saves the number in dedicated text block as maximum number of iterations.	0	0
TOTAL		13.629

CHECK LEARNING CURVE

Step	Cost Calculation	Salary Calculation
1. ML Engineer clicks on **GENERATE**	1 * 1 * 1.77	1.77
2. SYSTEM trains a classifier and shows learning curve	0	
TOTAL		1.77

CHECK VALIDATION REPORT

Step	Cost Calculation	Salary Calculation
1. ML Engineer starts a grid search	1 * 1 * 1.77	1.77
2. SYSTEM performs the grid search and shows a table with 5 best fitted models in ascending order of validation error	0	0
3. for each model in the table	0	0
3.1. ML Engineer checks the absolute value of the difference in validation and training error	(5 *) 1 * 2 * 1.77	17.7
end for each	0	0
4. if all models have a difference over a specified threshold	0	0
4.1. ML Engineer clicks **RESTART**	0.1 * 1 * 1.77	0.177
4.2. SYSTEM deletes all results	0	0
5. else if only one model is valid	0	0
5.1. ML Engineer clicks **PICK** button next to that model	0.3 * 1 * 1.77	0.531
5.2. SYSTEM saves model parameters on a file	0	0
6. else	0	0
6.1. ML Engineer calculate difference in validation error between first two valid models	0.6 * 3 * 1.77	1.062
6.2. ML Engineer calculate complexity of first two valid models	(3 *) 0.6 * 3 * 1.77	3.186
6.3. if the difference is an order of magnitude smaller than the error values AND the second model is less complex than the first	0	0
6.3.1. ML Engineer clicks **PICK** button next to second model	0.3 * 1 * 1 * 1.77	0.531
6.4. else	0	0

6.4.1. ML Engineer clicks **PICK** button next to first model end if	0.3 * 1 * 1 * 1.77	0.531
6.5. SYSTEM saves model parameters on a file end if	0	0
TOTAL		13.629

CHECK TESTING REPORT

Step	Cost Calculation	Salary Calculation
1. ML Engineer starts a test	1 * 1 * 1.77	1.77
2. SYSTEM executes the test using saved model and shows results, including hyperparameters values, test error, previous validation error and their difference	0	0
3. ML Engineer looks at the difference in validation and test error	1 * 2 * 1.77	3.54
4. if the difference is under the specified generalization tolerance threshold	0	0
4.1. ML Engineer clicks **ACCEPT** button	0.9 * 1 * 1.77	1.593
4.2. SYSTEM saves model parameters and flags it as ready for production	0	0
5. else	0	0
5.1. ML Engineer clicks **CANCEL** button	0.1 * 1 * 1.77	0.177
5.2. SYSTEM deletes all progress and restarts development	0	0
end if	0	0
TOTAL		7.08

PRODUCTION SYSTEM [ENRICO]

PRODUCTION SYSTEM: CONFIGURE

STEP	Cost Calculation	Salary Calculation
1. System Administrator launches Text Editor	1 x 1 x 1	1
2. System Administrator opens the ”productionConfig.json” file	1 x 1 x 1	1
3. System Administrator write configuration parameters in ”productionConfig.json”	1 x 1 x 1	1
4. System Administrator save configuration file ”productionConfig.json”	1 x 1 x 1	1
5. System Administrator close Text Editor	1 x 1 x 1	1
TOTAL		5

EVALUATION SYSTEM [JACOPO]

EVALUATION SYSTEM: CONFIGURE

Step	Cost Calculation	Salary Calculation
1.The Secure POS Expert launches a test editor, able to open .json files .	1 * 1 * 2.13	2.13
2.The Secure POS Expert opens the files : "eval_ambient_flags.json", “eval_config.json”, “eval_ip_configjson”.	(3 *) 2 * 1 * 2.13	6.39
3.The Secure POS Expert writes the new configuration parameters into the configuration files, in .json format, with respect of the respective specific validation schemas.	(3 *) 2 * 1 * 2.13	6.39
4.The Secure POS Expert saves the changes to the files.	(3 *) 1 * 1 * 2.13	6.39
5.The Secure POS Expert closes the text editor that has been using.	1 * 1 * 2.13	2.13
TOTAL		23.43

EVALUATE CLASSIFIER

Step	Cost Calculation	Salary Calculation
1.The Secure POS Expert launches the Evaluation System.	1 * 1 * 2.13	2.13
2.The Secure POS Expert launches a text editor able to open .json files.	1 * 1 * 2.13	2.13
3.The Secure POS Expert opens the file "report.json" corresponding to the test that is being evaluated.	1 * 1 * 2.13	2.13
4.The Secure POS Expert looks at the field containing a list of thresholds, and the respective measured values	3 * 1 * 2.13	6.39
4.1. If any one of the theses has been exceeded, then the classifier has failed its expectations.	0	0
4.1.1. Secure POS Expert must therefore send a new training request, therefore a new configuration request.	1 * 0.15 * 2.13	0.3195
4.2. Else	0	0
all the theses have been satisfied, the classifier is good, and nothing must be sent.	0 * 0.85 * 2.13	0
end if	0	0
5.The Secure POS Expert closes the text editor.	1 * 1 * 2.13	2.13
TOTAL		15.2295

TESTING

NON-RESPONSIVENESS

Non-Responsiveness was calculated by taking timestamps in each system at the beginning and at the end of their respective processes. These tests do not take into consideration the time needed to exchange messages between systems.

DEVELOPMENT SCENARIO

In this scenario a new classifier is developed and deployed. For this test, 150 sessions were required to create a learning set and the classifier hyperparameters were:

- Layers: minimum 1, maximum 3, step 1
- Neurons per layer: minimum 10, maximum 100, step 10

System	Mean Traversing Time	Probability of process continuing
Ingestion/Preparation	19.84 s (~132 ms per session)	1
Segregation	1.08 s	0.39
Development	74.31 s	0.90
Production	1.04 s	0
TOTAL	19.84 * 1 + 1.08 * 1 + 74.31 * 0.39 + 1.04 * 0.39 * 0.90 = **50.26 s**	

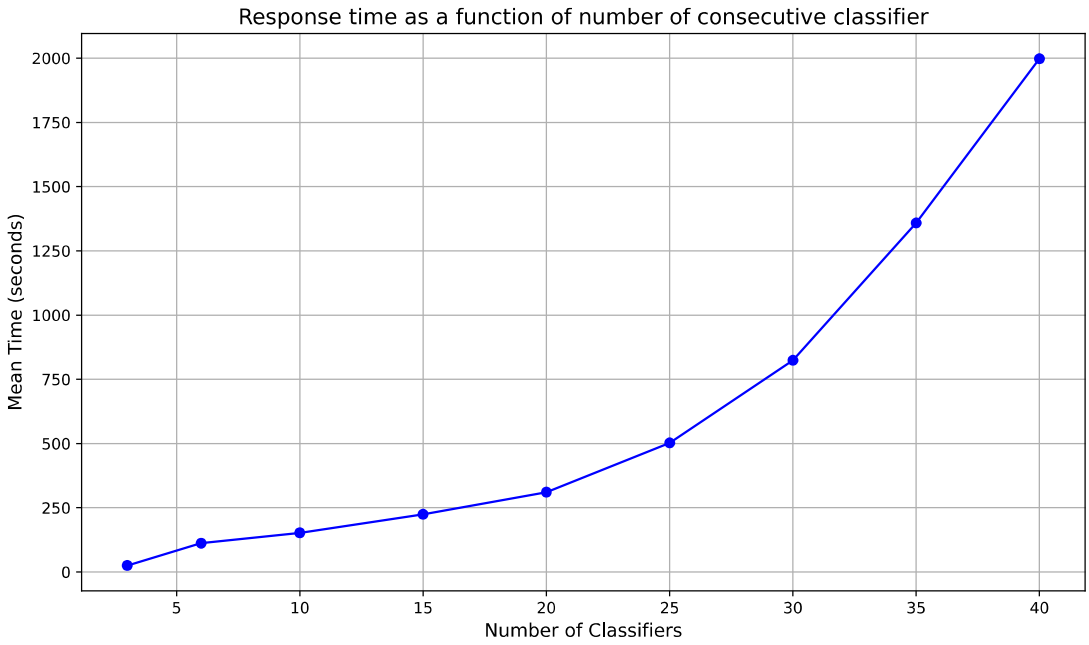
PRODUCTION SCENARIO

In this scenario an already deployed classifier is used to classify a session.

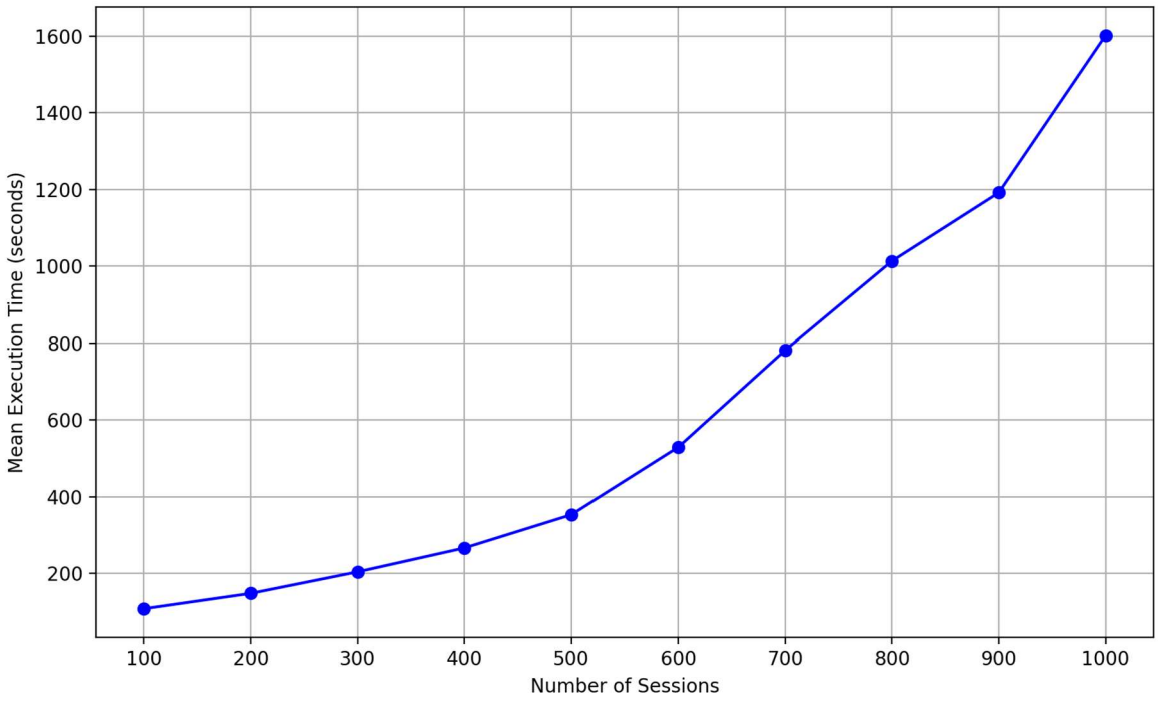
System	Mean Traversing Time	Probability of process continuing
Ingestion/Preparation	132.42 ms	1
Production	1.37 ms	0
TOTAL	132.42 + 1.37 = **133.79 ms**	

NON-ELASTICITY

DEVELOPMENT PHASE



PRODUCTION PHASE



EVALUATION SYSTEM STRESS-TEST

During testing, it was observed that the actual execution time of the evaluation system (which covers label store, db check, label query and report generation) is very short (less than 5 microseconds).

A test was developed in order to find a bound for *minimum incoming-packet period* .

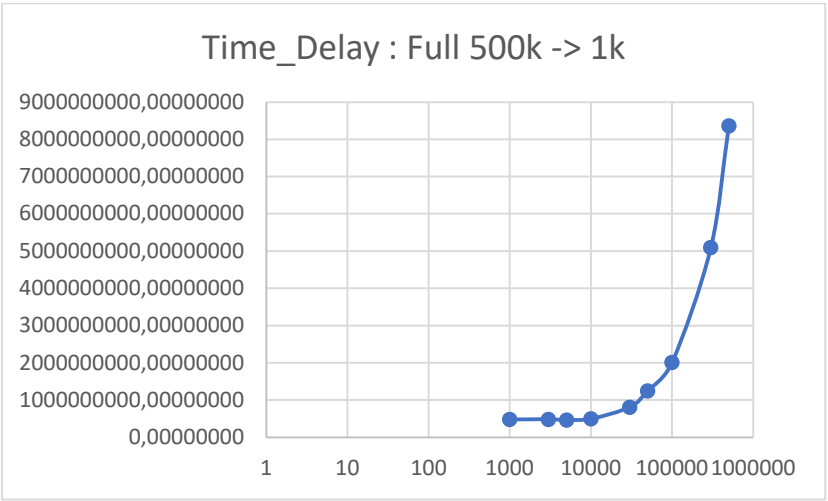
The test covered the following time intervals, in nanoseconds : 500k, 300k, 100k, 50k, 30k, 10k, 5k, 3k, 1k.

The times measured are the times intercurring between consecutive report generations, and during the test the parameter describing the “number of labels needed in order to generate a report” was a configuration constant.

Parameters where :

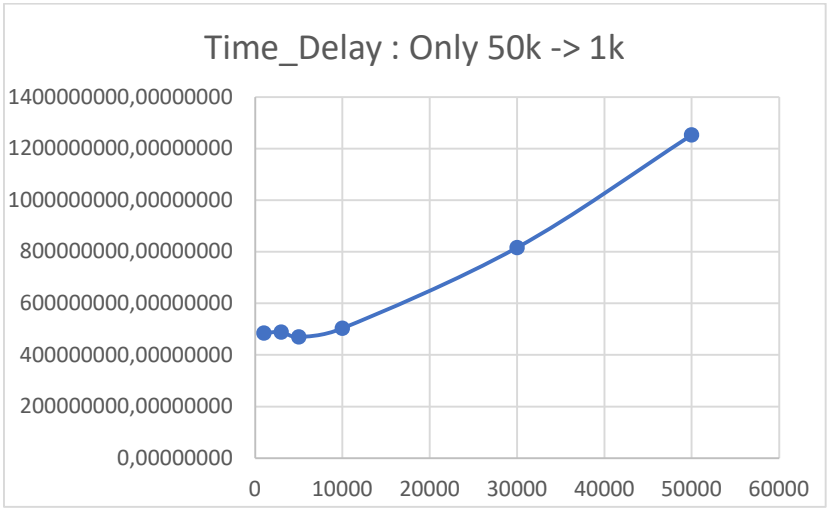
- num_labels_needed : 10 (meaning 10 from classifier and 10 from expert, with matching session_id)
- max_total_errors : 5
- max_consecutive_total_errors : 3

Each test consisted of 120 labels sent, with the given time period, and was repeated 15 times (for a total of 90 reports generated for each period value). The inter-report response period was then plotted, as can be seen in the following graphs.



If the system had no queueing problem at all, then the response time should be in the shape of : **a*gen_period + b**,

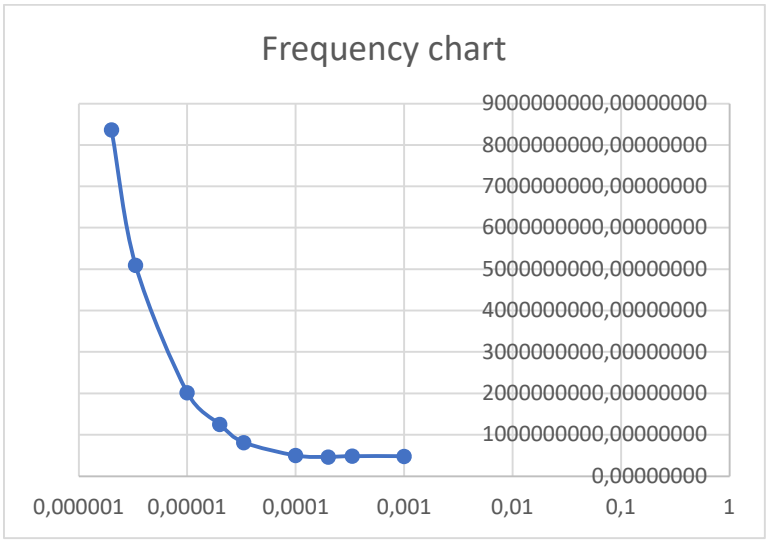
Therefore should appear to have a linear behaviour in a linear scale, or a (regular) exponential behaviour in the logarithmic scale. However, looking at the pot above, we can see that the time below 10k nanoseconds might seem to stop being as expected, so the plot was “zoomed” there :

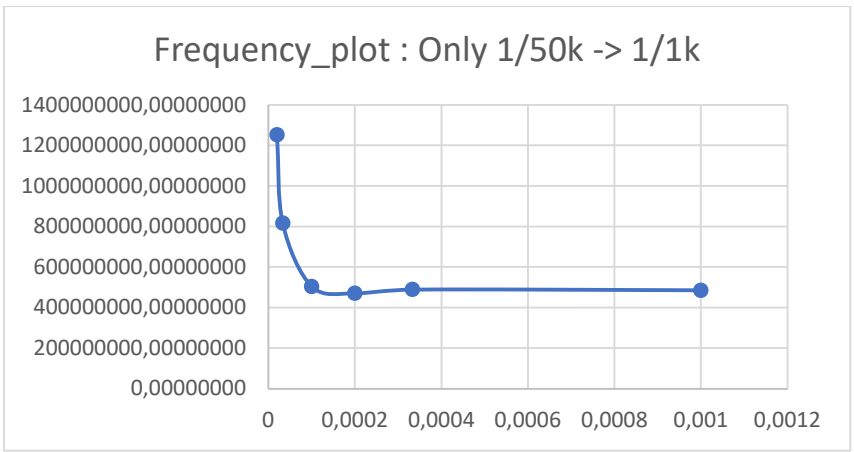


From this plot, which has linear axes, we can see a linear behaviour up to 10k ns, but then the values become constant.

Meaning that there was no longer gain in response-time to a reduction in input-period : a knee forms in the graph.

The same observation about expected-linear behaviour (not met) can be made if we extract the input-frequency instead of the input-period, as follows).





It should also be noted that a major block in the expansion of these tests is the fact that the system response is so short that in order to probe it we need very small times. Such times are so small that the main problem is not the system, but the Python request package, which doesn't manage to actually generate packages quickly enough for further probing.

NON-RESILIENCY

PREPARE/INGESTION SYSTEM

ID	Input	Consequence	Score
I1	Wrong data structure	Not detected, causes an internal error but doesn't propagate it.	**5**
I2	Missing label	Detected and filtered.	**4**
I3	Missing sample	Detected: if it is a time series, use interpolation; if it is a static record, try to find a previous record; otherwise, filter it.	**3**
I4	Duplicated records	Detected and solved automatically.	**1**
I5	Two (or more) labels arrive with same session_id	Detected: use the mode to determine the final label; if occurrences are equal, choose the first one.	**1**
I6	Missing record	Not detected, the system waits indefinitely.	**5**
I7	Outliers	Assign the relative upper or lower bound.	**1**
TOTAL			**20**

SEGREGATION SYSTEM

ID	Input	Consequence	Score
S1	Wrong prepared session schema.	Prepared session is discarded.	**3**
S2	Two (or more) prepared sessions with same UUID.	Only the first session is stored, the other(s) are discarded.	**3**
S3	Prepared sessions with wrong features (e.g., outliers).	The system processes them and generates the learning sets.	**5**
S4	Unbalanced labels.	The system is stopped.	**1**
S5	Features not well distributed.	The system is stopped.	**1**
TOTAL			**13**

DEVELOPMENT SYSTEM

ID	Input	Consequence	Score
D1	Missing set (training, validation or test) in learning set	Learning set is discarded, development doesn't start, and an alert is sent	**3**
D2	Malformed learning set (e.g. less labels than rows in a set)	Training/Validation/Testing fails and System crashes	**5**

D3	Too few records for training and/or validation	System trains only very bad classifiers and might remain stuck in validation loop	**5**
D4	ML Engineer passes malformed input	Error in input is detected, and an alert is sent	**3**
TOTAL			**16**

PRODUCTION SYSTEM

ID	Input	Consequence	Score
P1	Port already in use – server cannot start	Not detected, Production System Handler waits forever	**5**
P2	Classifier file corrupted before load	Detected: System does not load the corrupted file and waits until eventually it receives the correct file.	**1**
P3	Classifier file is not received	Not detected, Production System Handler waits forever	**5**
P4	Missing feature(s) in Prepared Session(s)	Detected and solved by discarding that Prepared Session(s).	**4**
S2	Two (or more) Prepared Sessions arrive with same uuid	Detected and solved by overwriting the older session.	**1**
TOTAL			**16**

EVALUATION SYSTEM

ID	Input	Consequence	Score
E1	Wrong label structure (fails validation).	Label is filtered out (not used), an alert is sent	**3**
E2	Bad label content (unexpected values).	Label is filtered out (not used), an alert is sent	**3**
E3	Labels come ouf-of-order (considering orders from expert vs classifier).	Label is inserted in DB and is used when both opinions have arrived.	**1**
E4	Two (or more) labels arrive with same session_id.	Causes db crash.	**5**
E5	Multiple label streams arrive, relative to different evaluation sessions.	Labels are all used together.	**5**
E6	Consecutive evaluation sessions are required by different clients (possible label_id collisions).	If the system is restarted with a proper flag, it cleans the database properly.	**1**
TOTAL			**18**

NON-INTEROPERABILITY

Feature	Prepare/Ingestion System	Segregation System	Development System	Production System	Evaluation System	Score
Can receive new input data while still working on a previous batch	YES	YES	NO	NO	YES	2
Can manage duplicate session id	YES	YES	YES	YES	NO	1
TOTAL						**3**