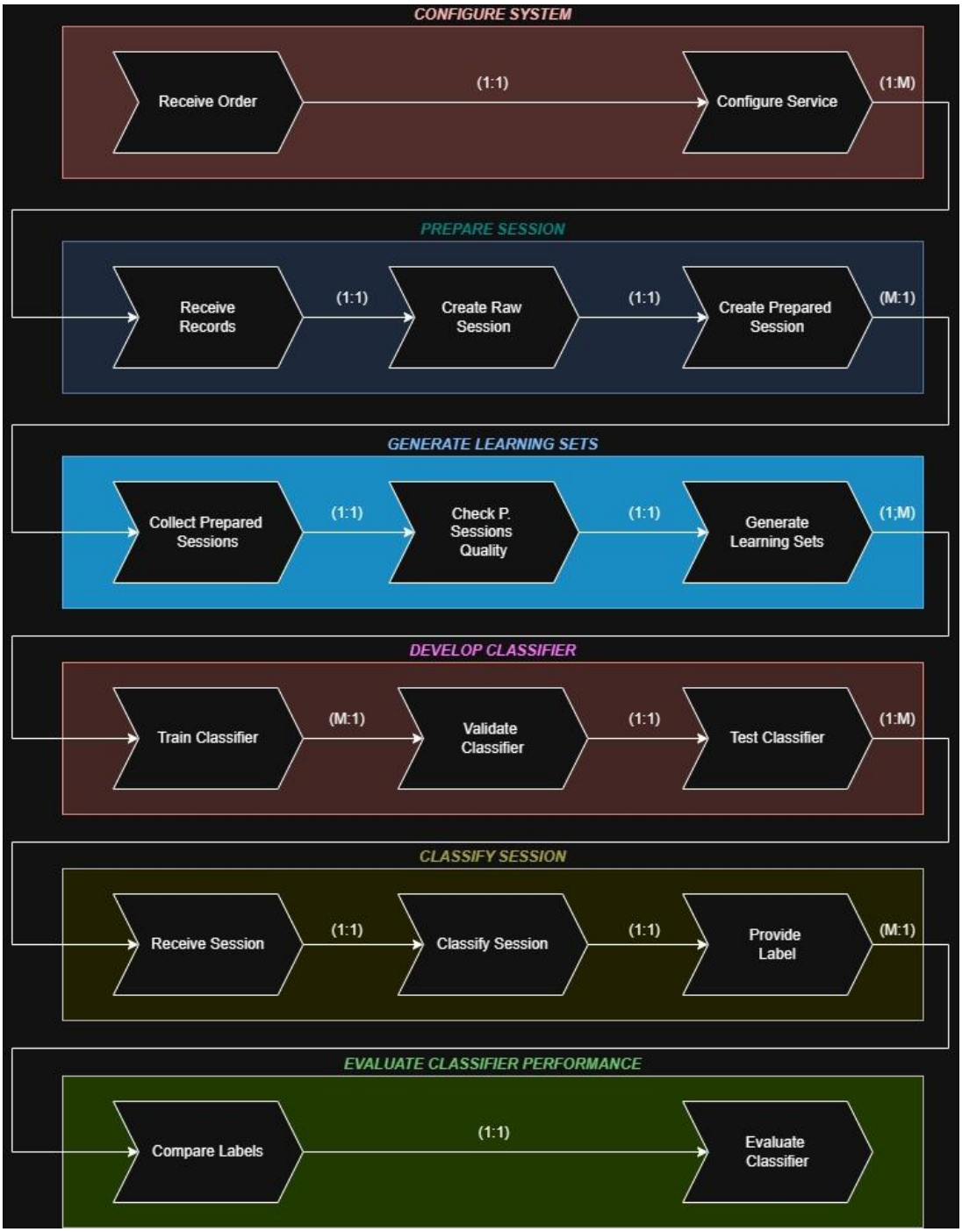
Project Documentation for Process Mining and Intelligence
SPEECH EMOTION RECOGNITION

Group Members

Rojan Shrestha
Nimra Tahir
Francesco Panattoni
Ronald Omoding

University of Pisa
Department of Information Engineering
Academic Year 2025/2026

1. Process Landscape

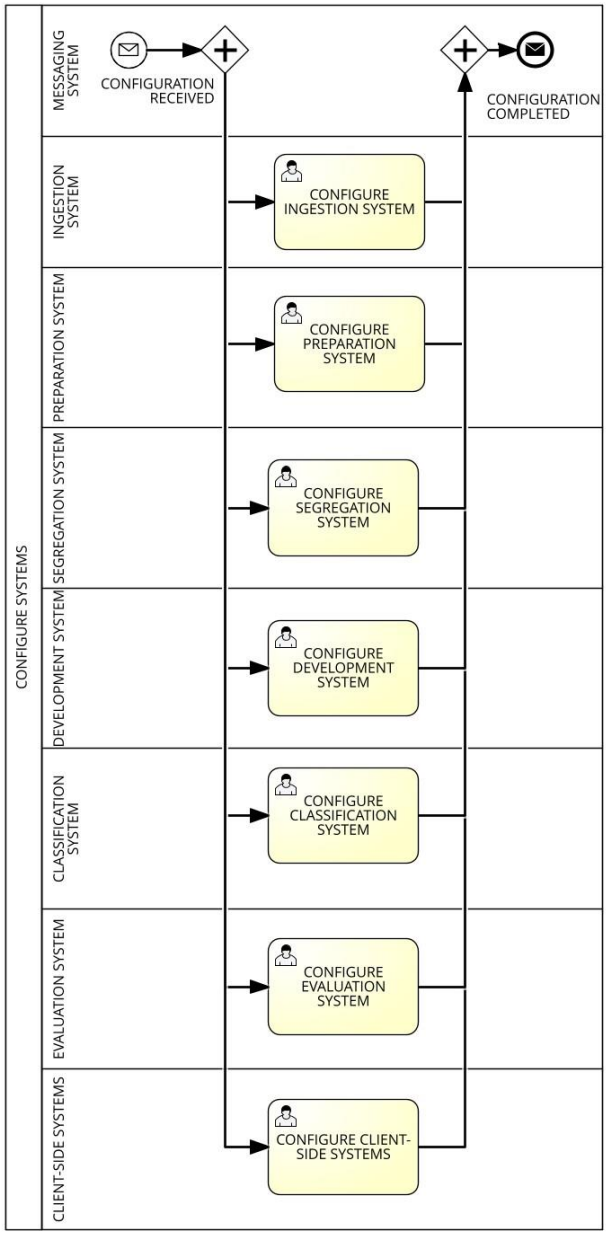


Cardinality Explanation:

1. **Receive Order** → **Configure Service** (**1:1**) - Each order received results in exactly one configuration.
2. **Configure Service** → **Receive Records** (**1:M**) - A single service configuration enables the system to collect many incoming records from (multiple) client-side sources.
3. **Receive Records** → **Create Raw Session** (**1:1**) - Each set of records received results in exactly one raw session. For SER, a raw session includes record sets (Audio Record, Calendar, Profile, and Annotator).
4. **Create Raw Session** → **Create Prepared Session** (**1:1**) - One raw session can be transformed into one prepared session (Feature Vector).
5. **Create Prepared Session** → **Collect Prepared Sessions** (**M:1**) - Multiple prepared sessions are collected to generate the learning sets.
6. **Collect Prepared Sessions** → **Check Sessions Quality** (**1:1**) - Each prepared session collected undergoes a quality check individually.
7. **Check Sessions Quality** → **Generate Learning Sets** (**1:1**) - A batch of quality-checked sessions generates a single learning set.
8. **Generate Learning Sets** → **Train Classifier** (**1:M**) - A single generated learning set may support several training iterations (epochs) during classifier development to explore many **combinations of hyperparameters** i.e. **Grid Search**.
9. **Train Classifier** → **Validate Classifier** (**N:1**) - Multiple training iterations feed into a single validation phase, where all trained variants are assessed through one validation process.
10. **Validate Classifier** → **Test Classifier** (**1:1**) - Each validated classifier is tested individually.
11. **Test Classifier** → **Receive Session** (**1:M**) - One tested classifier can classify multiple received sessions.
12. **Receive Session** → **Classify Session** (**1:1**) - Each session is classified individually.
13. **Classify Session** → **Provide Label** (**1:1**) - Each session classification results in exactly one output label.
14. **Provide Label** → **Compare Labels** (**M:1**) - Multiple provided labels from different classification sessions are used together in one evaluation step, which compares them to the corresponding human-provided labels.
15. **Compare Labels** → **Evaluate Classifier** (**1:1) -** Each evaluation cycle produces one overall classifier evaluation result. Even though multiple label comparisons occur inside the cycle, they collectively generate a single evaluation.

2. Skeleton Factory

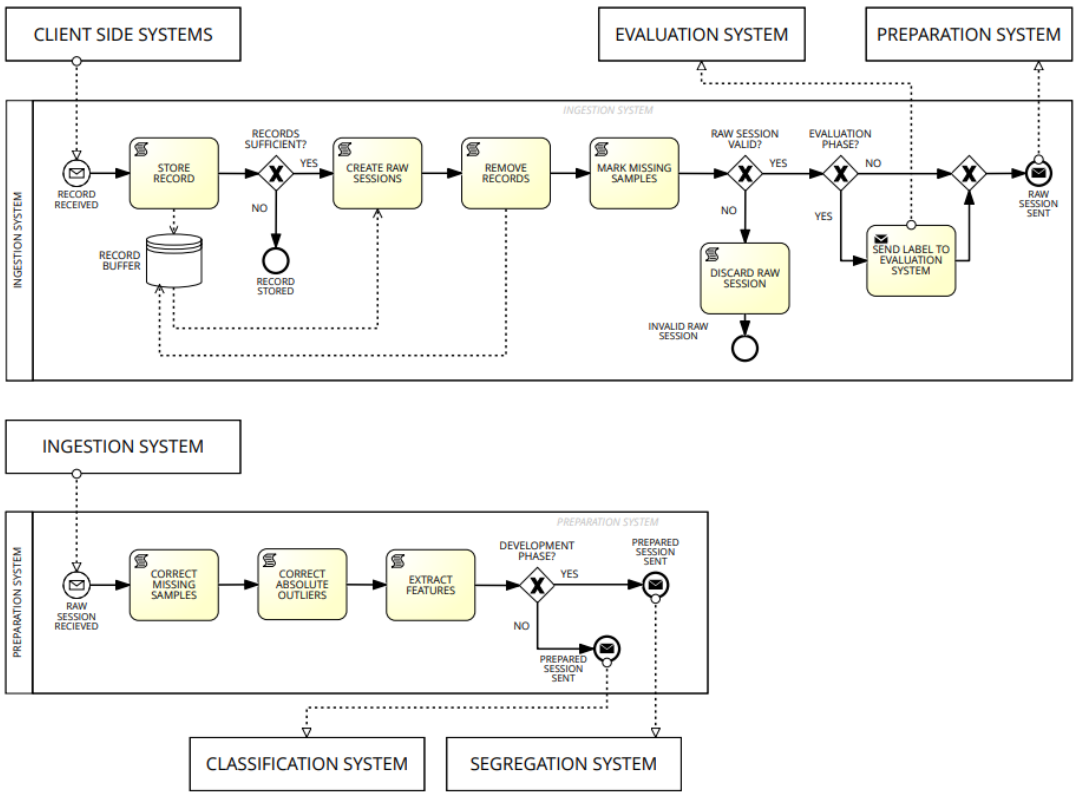
2.1 Configure Systems



This phase defines all global settings for the ML factory - configures data sources, feature extraction rules, hyperparameter ranges, and quality thresholds, ensuring that every downstream process operates with consistent and valid parameters.

2.2 Prepare Session

PREPARE SESSION

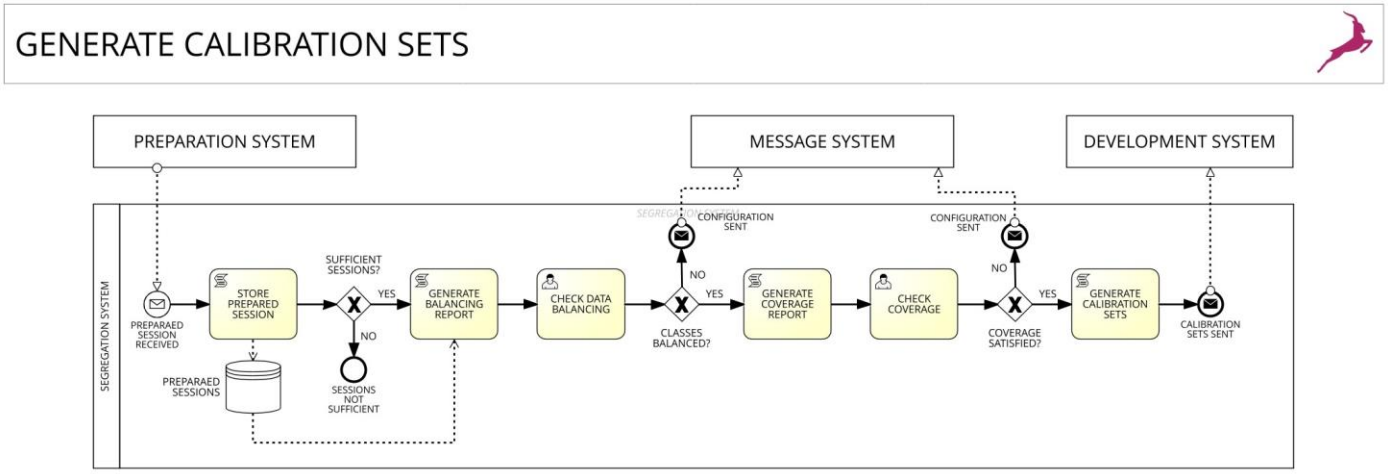


Ingest raw audio data, clean it, and extract features to produce structured sessions ready for model training.

The workflow checks "Evaluation Phase" inside the Ingestion System before sending the raw session onward, while the Preparation System contains a separate "Development Phase" gateway. Why is the phase decision split across two systems instead of being evaluated once and propagated as a single control flow condition?

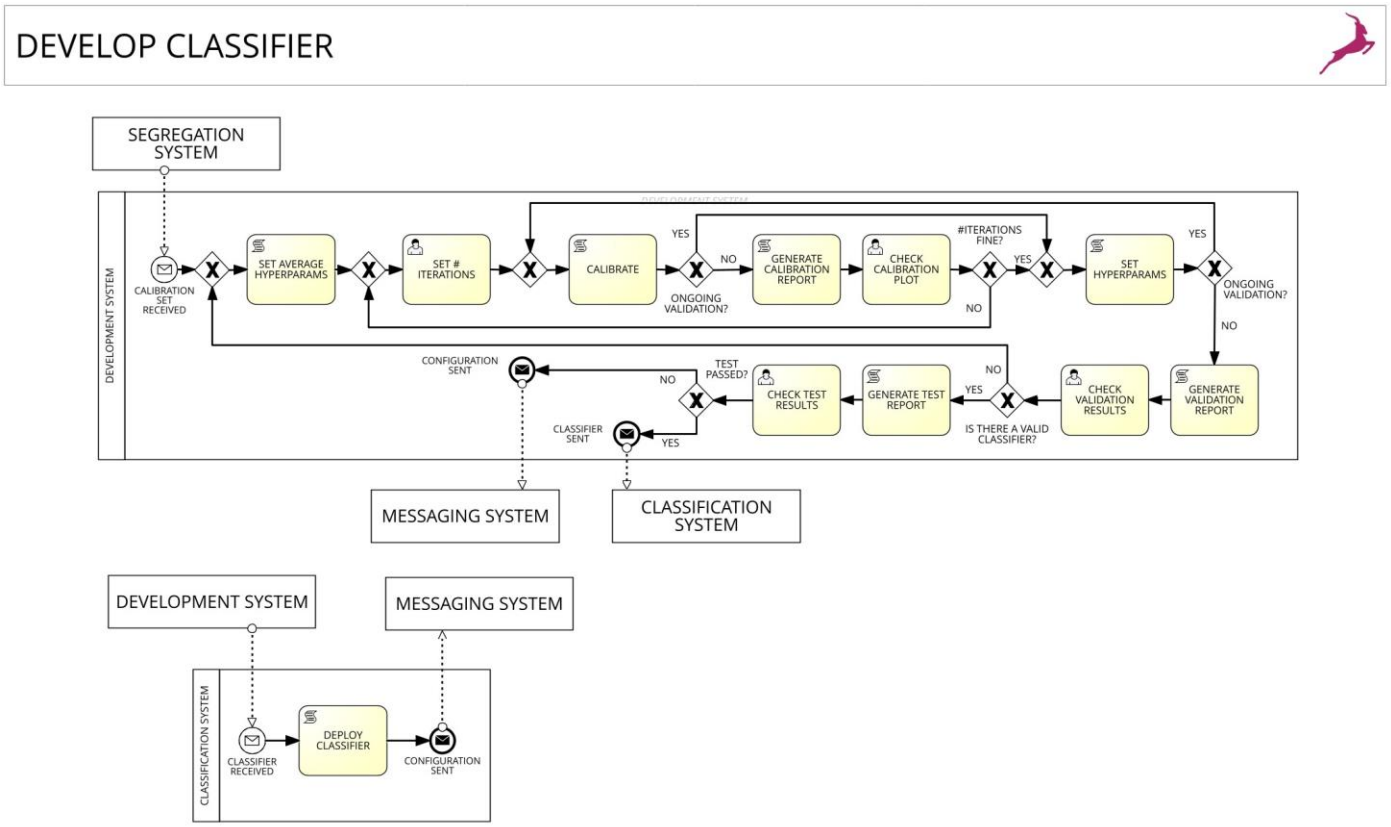
The phase decision is split across the two systems because each system controls a different stage of the lifecycle: Ingestion routes raw sessions for evaluation, while Preparation applies development-specific processing.

2.3 Generate Labeled Sets



Splits prepared sessions into training, validation, and test sets after verifying class balance and input-space coverage. Ensures that the resulting learning sets are sufficient, valid, and representative for reliable classifier training and evaluation.

2.4 Develop Classifier



Trains multiple hyperparameter configurations, validates them to select the best-performing classifier under overfitting constraints, and tests the selected model to confirm generalization. Produces a validated,

production-ready classifier.

We have three tasks: “Generate training validation report”, “Generate test report” and “Generate validation reports”. What is the exact difference between them and why do we need all three?

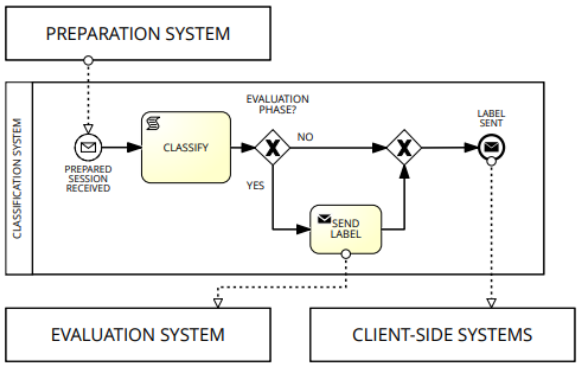
Training report = training-time diagnostics (loss curves, over/underfitting) for data scientists. **Test report** = technical metrics on a hold-out set for ML/engineering go/no-go. **Validation report** = business-level evaluation (KPIs, A/B, fairness) for product/PO; we should rename them to make this split explicit.

In Develop Classifier, the arrows labelled “ONGOING VALIDATIONS” loop back into the flow around “Train model” and “Set final hyperparameters”. Do these loops mean post-deployment monitoring, or are they just part of the normal development cycle?

During the **Develop Classifier** process, ongoing validations allow the data scientist to iterate retraining and reassessing the model multiple times - without restarting the full workflow. Once performance is confirmed and hyperparameters are finalized, we generate the deployment configuration. From that point onward, continuous monitoring and performance checks are managed within the **Evaluate Classifier Performance** phase.

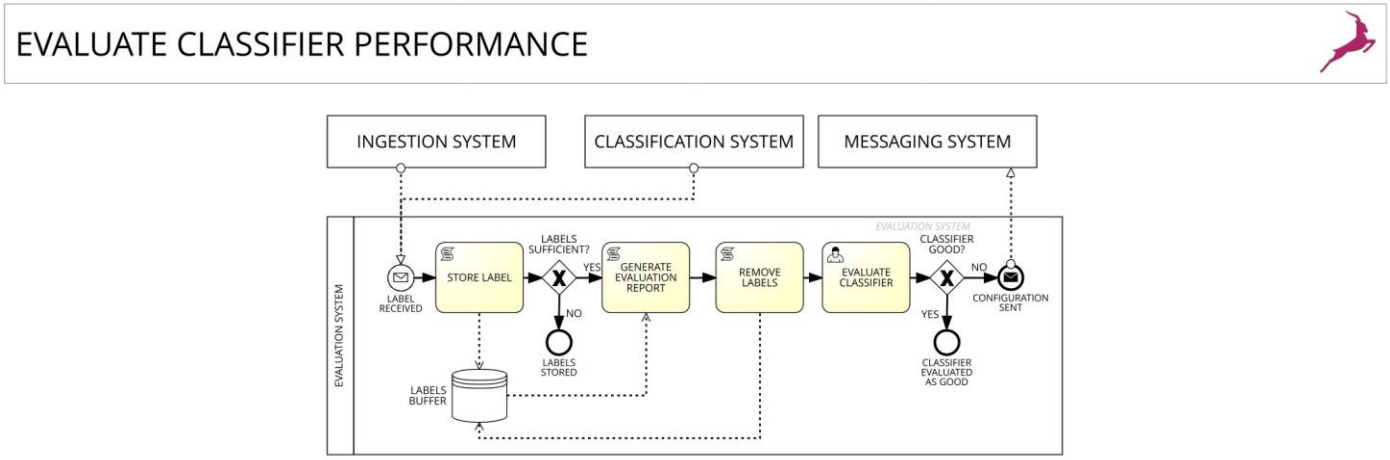
2.5 Classify Session

CLASSIFY SESSION



Use the deployed classifier to infer labels for new, unseen sessions in the production environment, providing real-time predictions on operational data.

2.6 Evaluate Classifier Performance



Regularly checks how well the classifier is performing by comparing its predictions to the correct labels, making sure its accuracy stays acceptable over time.

What is the actual value of storing the labels? Are we doing it for a specific reason, or is it just a legacy step?

We should check the logs to see if these stored labels are ever actually reused. If they're just being archived and never touched again, it means this step is probably creating overhead without providing any real benefit, and we should reconsider its purpose.

3. Data Modeling

Raw Inputs (client-side systems → records):

1. Audio Sensor → UUID, Audio Samples (numeric amplitude values of the waveform), Sampling Rate and Duration
2. Calendar → UUID, activity: {sport, meditation, work, home, relax}, period of day: {morning, afternoon, evening, night}
3. Profile → UUID, speaker ID, age, gender and language: {Mandarin, English}
4. Annotation → UUID, labeler ID, emotion: {Angry, Happy, Neutral, Sad, Surprise}, confidence (optional) and timestamp (optional)

One session = {Audio Record + Calendar+ Profile+ Annotator}

Output:

5. emotion: {Angry, Happy, Neutral, Sad, Surprise}

Features (net input, derived in the Preparation System - BPMN):

Prosodic features:

(a) rmse (short-term energy)
(b) zeroCrossingRate
(c) chroma_stft}

Spectral features:

(d) spectralCentroid
(e) spectralBandwidth
(f) SpectralRolloff

MFCC features (20-dimensional):

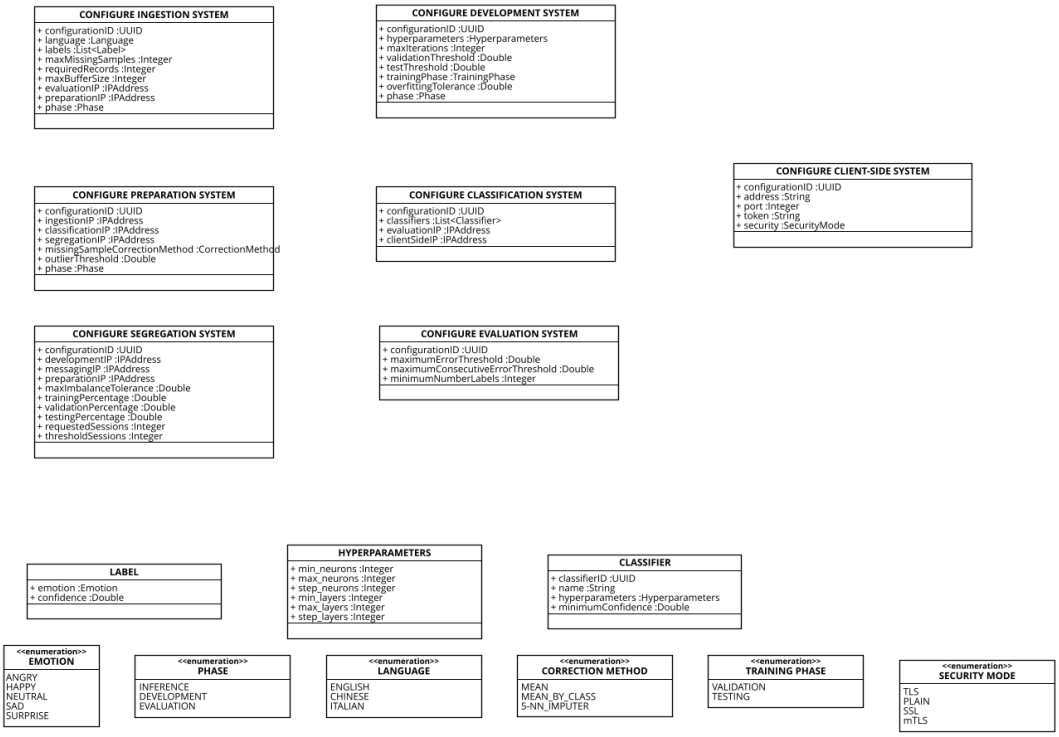
(g) mfcc1 – mfcc20

NOTE:

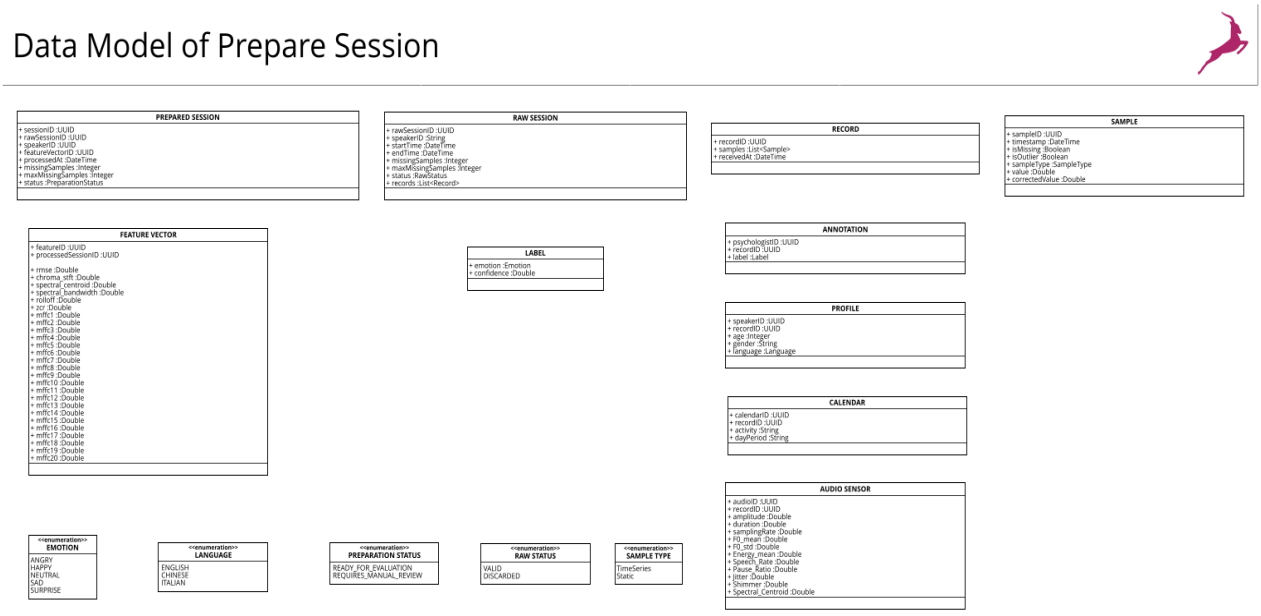
1. Audio Sample = 1 numeric amplitude value at a time step
2. Sample is the smallest atomic unit of input data

3.1 DATA MODEL of CONFIGURE SYSTEM (Francesco)

Data Model of Configure Systems



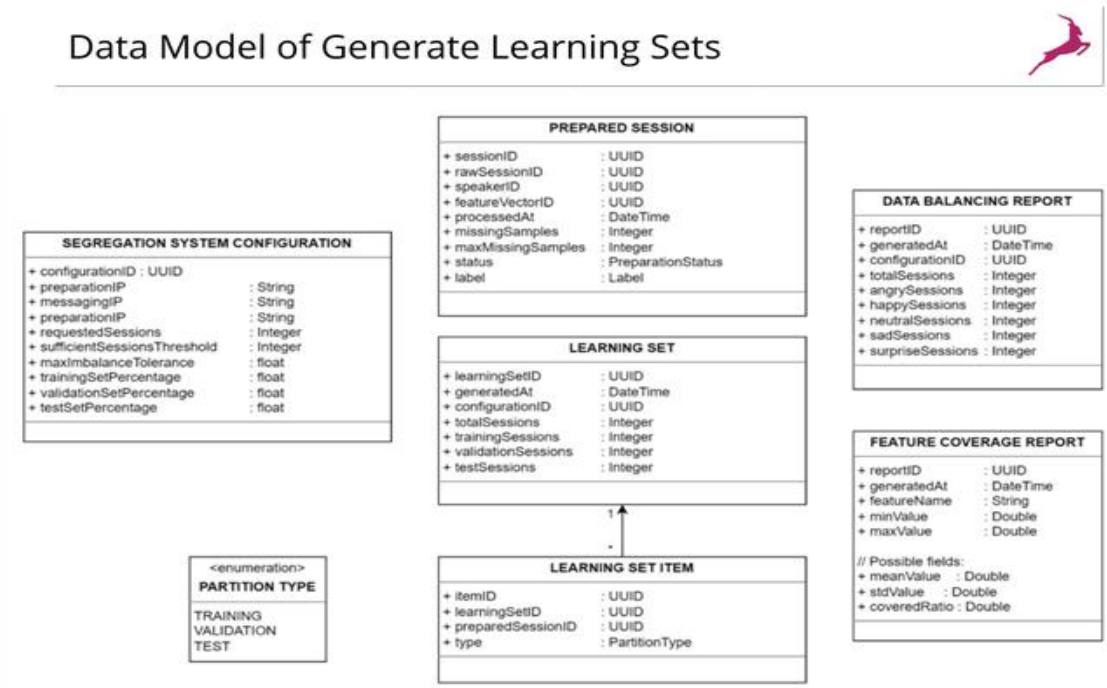
3.2 DATA MODEL of PREPARE SESSION (Francesco)



Feedback:

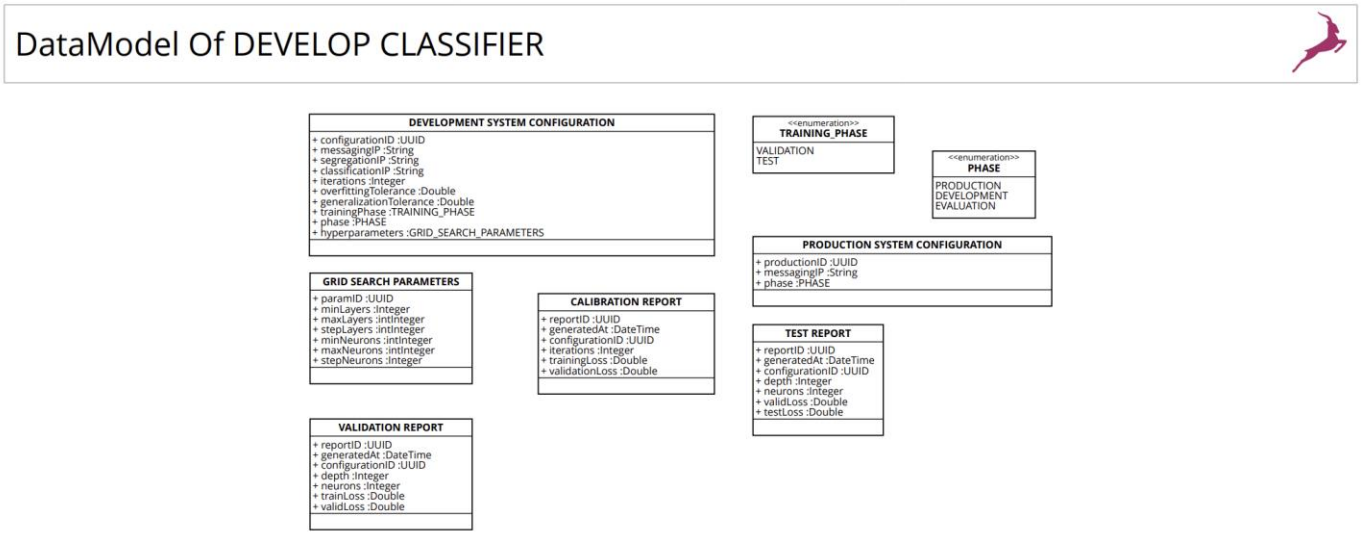
1. Annotation, Profile, Callender and Audio Sensor, all are Record but only Audio Sensor can have deviceID so deviceID as a field in Record is unclear
2. Samples usually refer to timeseries. If the values of Annotation,Profile, Calender are considered static samples and that they have a missing value, it means they affect the validity of the raw session. Since the only purpose of the raw session is to create a prepared session with feature vectors. Referring to **TASK Check Input Coverage**, the features are only dependent upon the audio samples, so it is unclear why the values of Annotation, Profile, Callender are considered samples.
3. Mffc1-mffc20 can be grouped together as a single field - list (Mffc)
4. Why Record and Annotation both have Label fields? “label: Label” only makes sense if Label is an enum but Label is a class per the model and should be referenced with UUID
5. Maybe label a separate field in Prepared Session?
6. PreparedSession.status can be “Evaluation: Boolean” same with Raw Status

3.3 DATA MODEL of GENERATE LEARNING SETS (Rojan)



1. Why configurationID ? To say “This report was generated using configuration X”
2. // Possible fields: FEATURE COVERAGE REPORT : + meanValue: Double, + stdValue: Double, + coveredRatio: Double
3. Do we need trainingSessions, validation Sessions, and test Sessions count?
4. Should LABEL be a separate entity? (OR)

3.4 DATA MODEL of DEVELOP CLASSIFIER (Nimra)

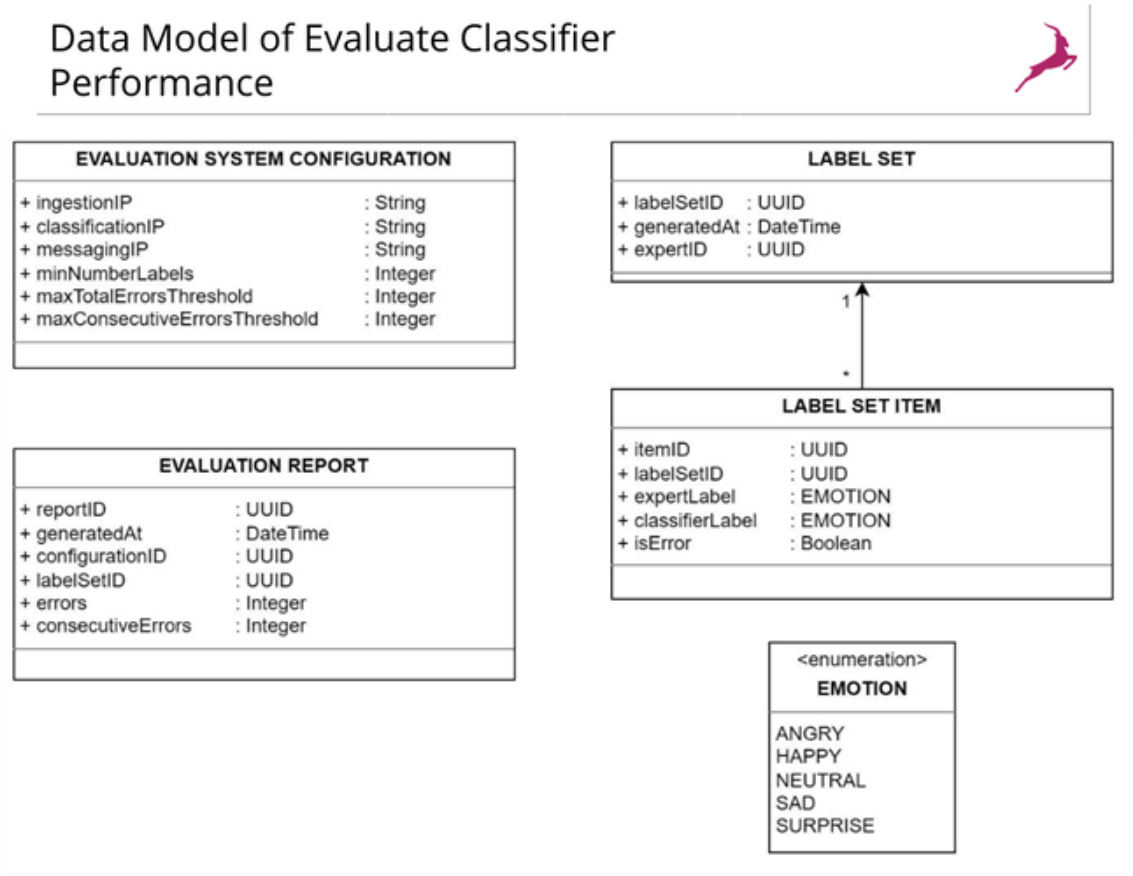


- ***DEVELOPMENT SYSTEM CONFIGURATION*** is correctly modeled as the core object of this task: it holds the IPs of the connected systems, the iteration budget, and the tolerance thresholds. All reports (CALIBRATION REPORT, VALIDATION REPORT, TEST REPORT) are linked back to it via configurationID, so every result can be traced to the exact configuration used during that run.
- Splitting results into three report classes matches the factory: calibration, validation and test are distinct phases with different purposes.
    - Calibration focuses on iterations and the pair (trainingLoss, validationLoss) over time
    - Validation report stores grid-search results (depth, neurons, losses per configuration)
    - Test report captures the final chosen configuration with a single pair of validLoss and testLoss.
- In **CALIBRATION REPORT** and **VALIDATION REPORT**, losses and hyperparameter values are modeled as arrays [0..*]. This reflects the fact that one *calibration/validation run produces a full learning curve over many iterations*, not just a **single scalar**, and is aligned with how the "Check Calibration Plot" task uses these curves.
- **GRID SEARCH PARAMETERS** collects the search space (minLayers, maxLayers, stepLayers, minNeurons, maxNeurons, stepNeurons) instead of mixing these with live configuration fields. This keeps the development configuration focused on "what was actually run" while allowing the same parameter grid to be reused for multiple experiments if needed.
- Enumerations (TRAINING_PHASE, PHASE) ensure that both internal training status (validation/test) and global lifecycle status (development/production) remain explicit and valid.
- **PRODUCTION SYSTEM CONFIGURATION** only stores what is strictly needed for deployment (productionID, messagingIP, phase), which is coherent with the Deploy Classifier part of the factory. It reuses the same PHASE enum, so the production configuration can be interpreted together with the development configuration in dashboards or logs.
- All report entities use reportID : UUID and generatedAt : DateTime, while configurations use configurationID / productionID. This ensures that every training/validation/test run can be audited later (e.g., "which configuration generated this validation report on this date?").
- Every metric can be traced back to the exact configuration and hyperparameters, supporting reproducibility, debugging, and regulatory transparency.

3.5 DATA MODEL of CLASSIFY SESSION (Ronald)

Data Model of Classify Session

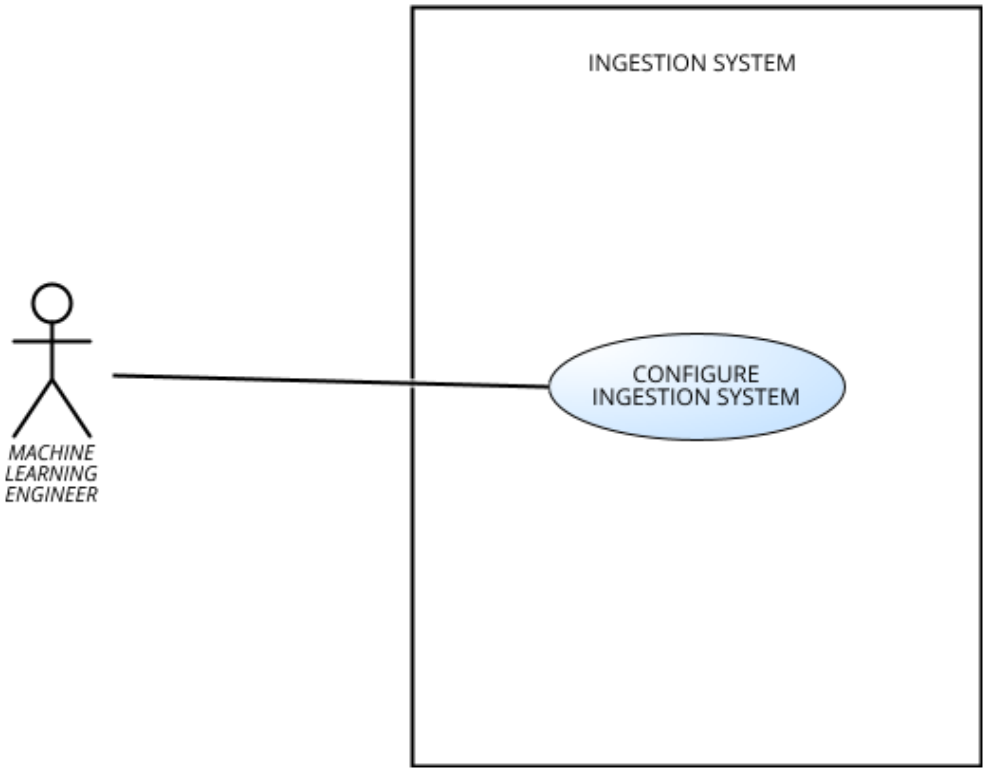
3.6 DATA MODEL of EVALUATE CLASSIFIER PERFORMANCE (Rojan)

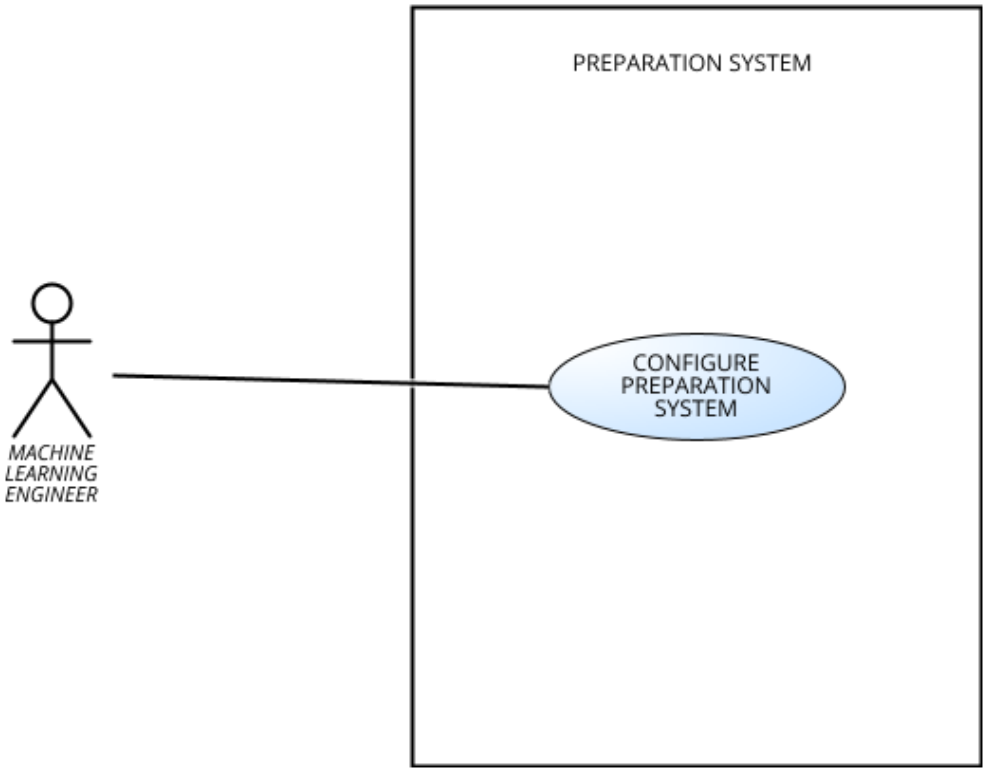


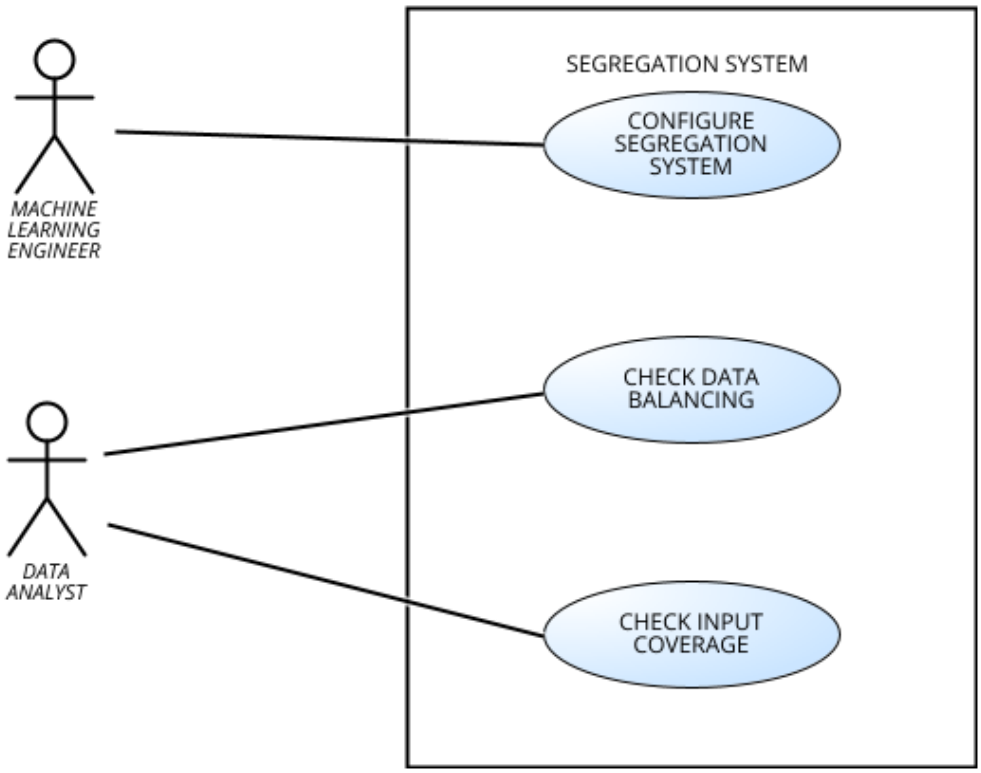
1. Prof says “For each Process” so Configure Service would have all configuration models but since they are used in other process as well should we include each configuration model also in respective process? Ex: Segregation config model in both Configure Service and GENERATE LEARNING SETS?

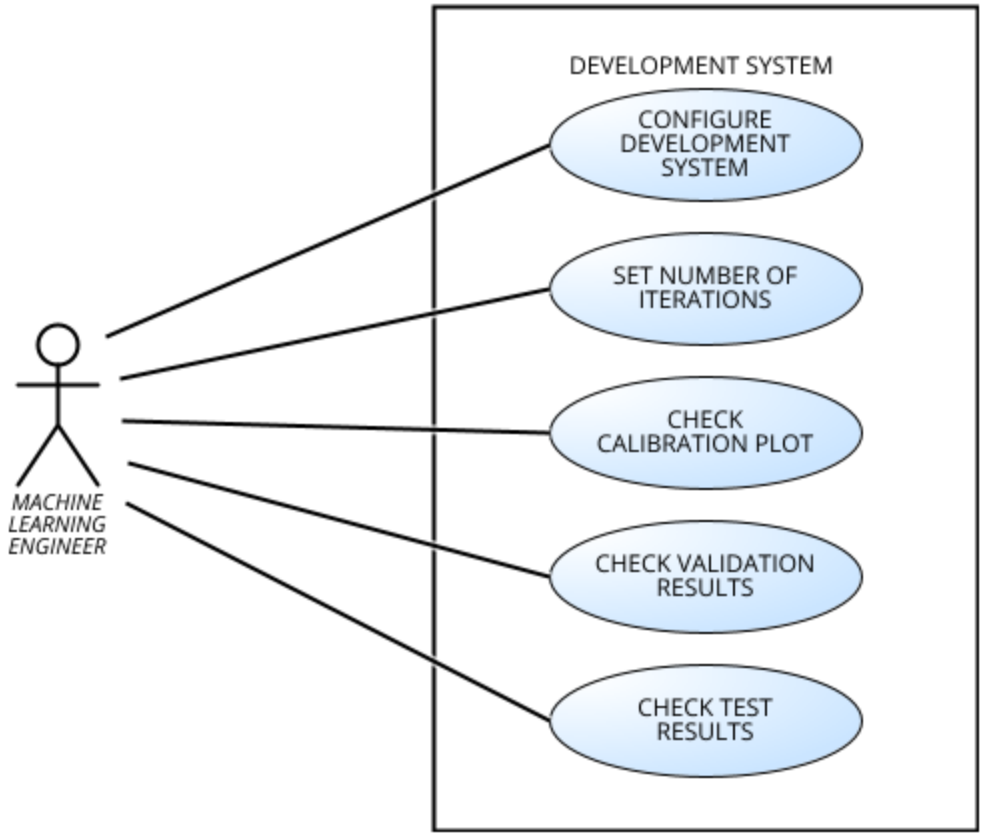
4. Task Modeling

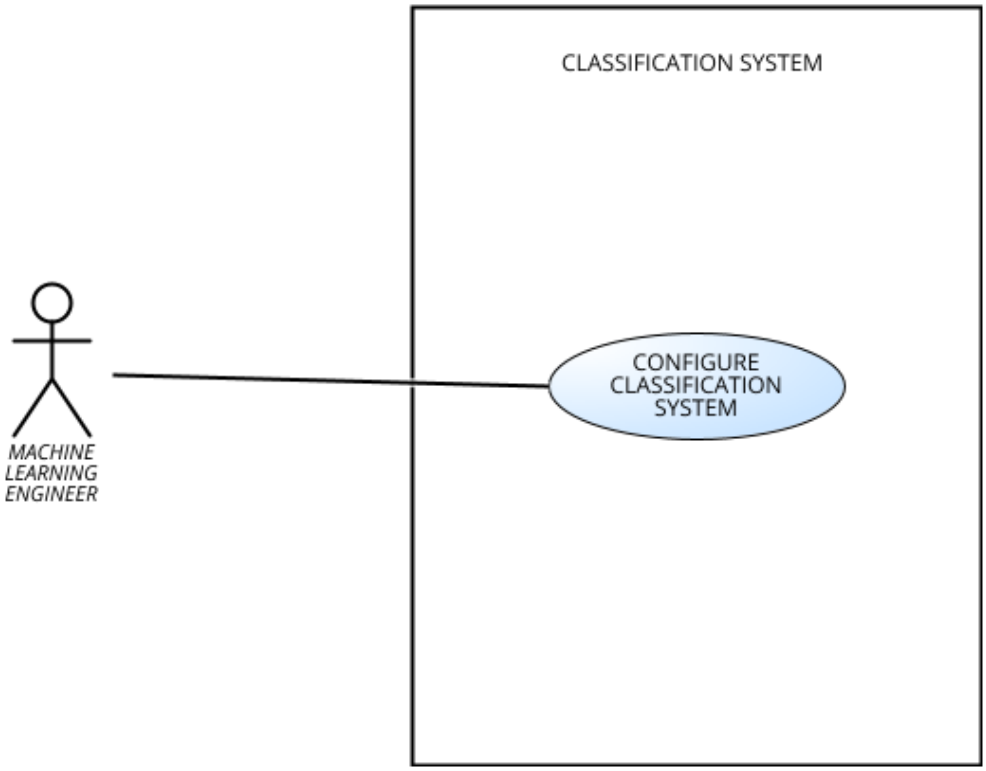
4.1 Use Cases

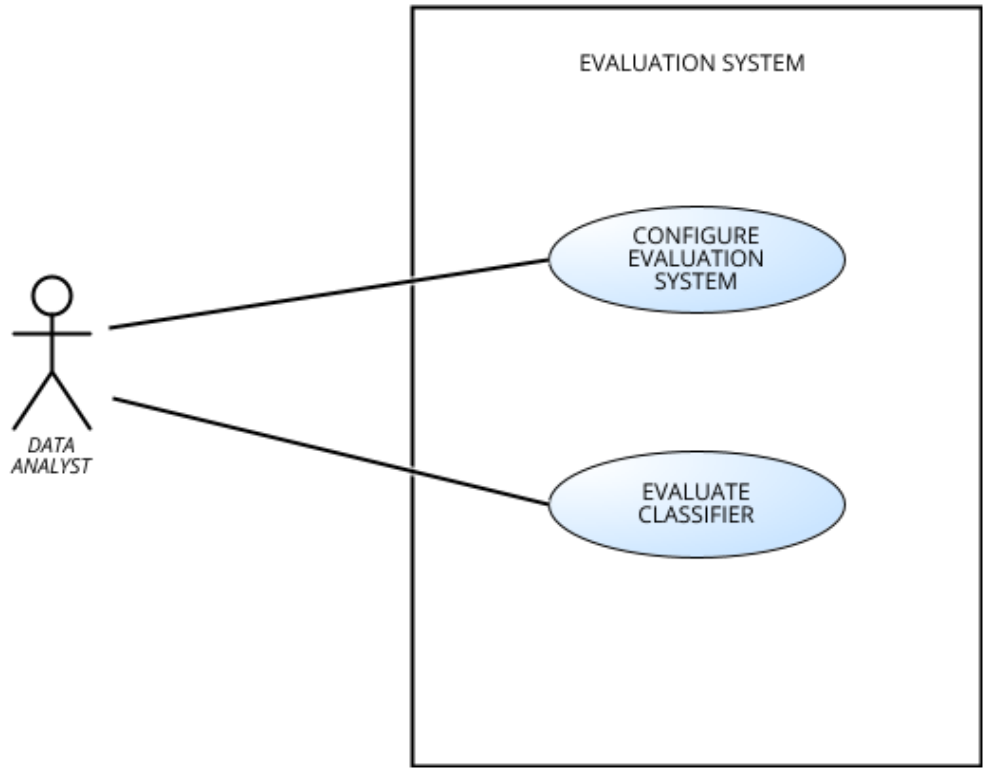


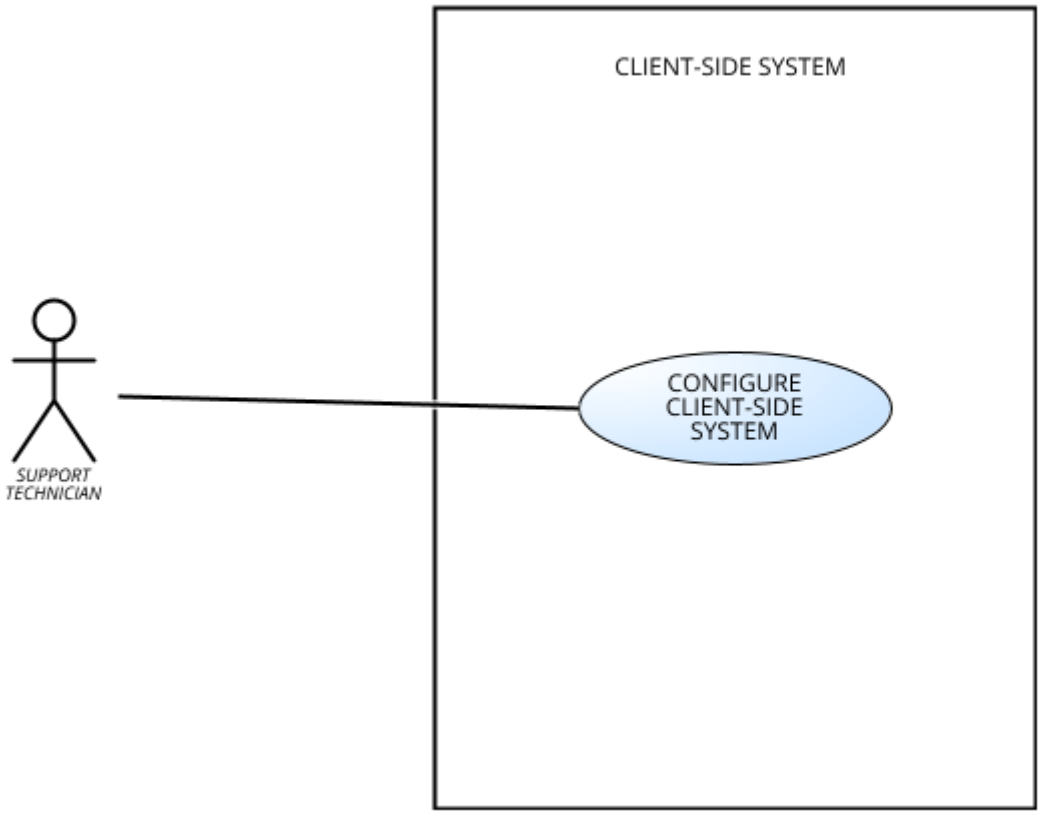



SEGREGATION SYSTEM
CONFIGURE SEGREGATION SYSTEM
CHECK DATA BALANCING
CHECK INPUT COVERAGE
MACHINE LEARNING ENGINEER
DATA ANALYST


DEVELOPMENT SYSTEM
CONFIGURE DEVELOPMENT SYSTEM
SET NUMBER OF ITERATIONS
CHECK CALIBRATION PLOT
CHECK VALIDATION RESULTS
CHECK TEST RESULTS
MACHINE LEARNING ENGINEER


CLASSIFICATION SYSTEM
CONFIGURE CLASSIFICATION SYSTEM
MACHINE LEARNING ENGINEER


EVALUATION SYSTEM
CONFIGURE EVALUATION SYSTEM
EVALUATE CLASSIFIER
DATA ANALYST


CLIENT-SIDE SYSTEM
CONFIGURE CLIENT-SIDE SYSTEM
SUPPORT TECHNICIAN

4.2 Normalized Salaries

ACTOR	Description	LINK
Data Analyst	Collects, organizes, and cleans data; analyzes data for patterns/trends; builds databases; creates visualizations; and communicates insights.	https://www.payscale.com/research/IT/Job=Data_Analyst/Salary
Machine Learning Engineer	Builds ML systems, implements ML algorithms, analyzes data statistically, runs experiments/tests, maintains and improves ML systems, and bridges data scientists with engineering teams.	https://www.payscale.com/research/IT/Job=Machine_Learning_Engineer/Salary
Domain Expert (Psychologist)	Examines audio recordings, segmenting them into distinct utterances when needed, and assigns the appropriate emotional label to each segment. Specializes in interpreting vocal cues and emotional nuances to ensure accurate ground-truth annotations.	https://www.payscale.com/research/IT/Job=Psychologist/Salary
Support Technician	Install, configure and maintain hardware/software and network for clients/devices. Fix equipment issues, perform upgrades, set up new devices, assist	https://www.payscale.com/research/IT/Job=Support_Technician%2C_Information_Technology_(

	users, and manage system inventory.	

Normalized salaries are calculated using the following method:

1. Determine the yearly salary;
2. Find the lowest salary (System Engineer);
3. Divide this amount by the lowest salary (used as the baseline).

4.3 Task

STEP COST = OCCURENCE × COGNITIVE × SALARY

OCCURRENCE is the number of executions of the step.

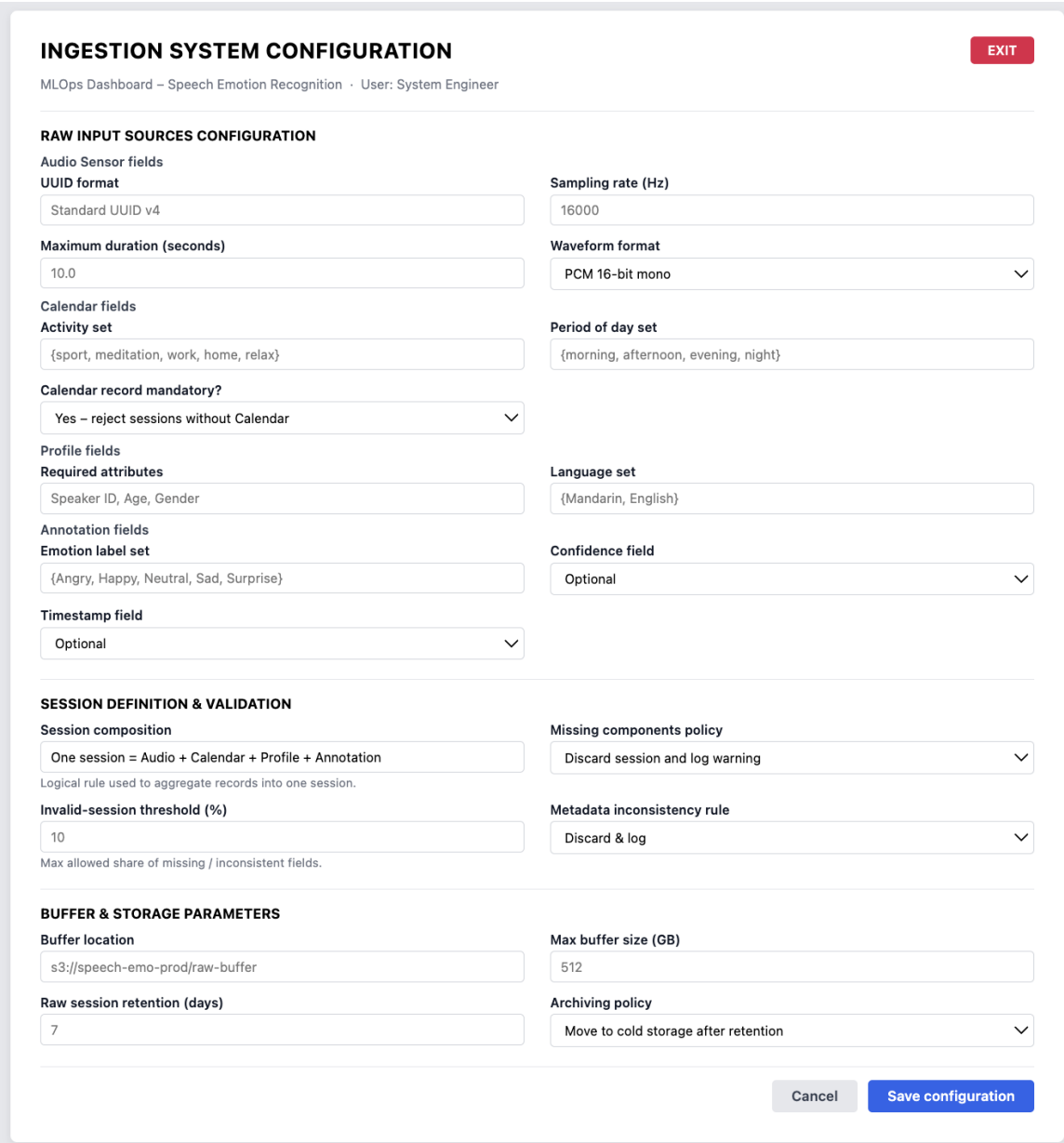
COGNITIVE is the cognitive effort. It can have these values:

- **Remember (1):** Perform the step by recalling a previous instance of the same action.
- **Understand (2):** Perform the step by identifying a value within a set of predefined categories.
- **Apply (3):** Perform the step by following a company-defined procedure.
- **Analyze (4):** Perform the step by determining unknown categories.

SALARY is the normalized job salary (calculated in previous page).

4.3.1. Configure the Ingestion System (Nimra)

	STEP	O	CE	O × CE
1	**USER** opens the "*Ingestion System Configuration form*"	1	1	1
2	**SYSTEM** loads the configuration interface with default / current parameters			
3	**USER** reviews the raw-input sources configuration (Audio Sensor, Calendar, Profile, Annotation)	1	2	2
3.1	**USER** edits/confirms the Audio Sensor fields (UUID, sampling rate, max duration, waveform format, etc...)	1	3	3
3.2	**USER** edits/confirms the Calendar fields (activity set, period of day, mandatory flag)	1	3	3
3.3	**USER** edits/confirms the Profile fields (speaker ID, age, gender, language set)	1	3	3
3.4	**USER** edits/confirms the Annotation fields (labeler ID, emotion set, confidence/timestamp options)	1	3	3
4	**USER** reviews the minimum number of Records	1	2	2
4.1	**USER** edits/confirms the minimum number of Records	1	3	3
5	**USER** reviews the max missing Samples	1	2	2
5.1	**USER** edits/confirms the max missing Samples	1	3	3
6	**USER** reviews the IP Addresses	1	2	2
6.1	**USER** set Evaluation IP Address	1	3	3
6.2	**USER** set Preparation IP Address	1	3	3
7	**USER** selects *Save to store the configuration*	1	1	1
8	**SYSTEM** validates the configuration and shows a confirmation message			
9	**USER** closes the configuration interface	1	1	1
TOTAL (O × CE) COST		35		
USER is a Machine Learning Engineer: Normalized Salary		× 1.67		
TOTAL HUMAN COST		**58.45**		



FEEDBACK: These are the parameters we think should be in the mockup considering they are used in Ingestion system (refer to gateways in skeleton factory)

1. required_records* - Minimum number of records needed before creating a raw session.
2. max_missing_samples*- Upper bound on allowable missing samples in a raw session. If exceeded, the raw session is discarded.
3. evaluation_sys_addr* -Address/endpoint of the Evaluation System for sending labels during evaluation phase.
4. preparation_sys_addr* + client sys addr - Address/endpoint of the Preparation System to which raw sessions are forwarded.

5. phase - Indicates the active lifecycle phase: development, production, evaluation

Remove Session Definition & Validation and Buffer & Storage Parameters.

YOU DON'T HAVE TO REDO THE COST TABLE. WE ALREADY UPDATED.

4.3.2. Configure the Preparation System (Ronald)

	STEP	O	CE	O × CE
1	**USER** opens the "*Preparation System Configuration*" form	1	1	1
2	**SYSTEM** loads the configuration interface with default parameters			
3	**USER** reviews the connected system IP addresses	1	2	2
3.1	**USER** edits/confirms Ingestion System IP address	1	3	3
3.2	**USER** edits/confirms Classification System IP address	1	3	3
3.3	**USER** edits/confirms Segregation System IP address	1	3	3
4	**USER** reviews the preprocessing parameters	1	2	2
4.1	**USER** opens Missing Samples Correction Method field	1	1	1
4.2	**USER** edits/confirms the Missing Samples Correction Method	1	3	3
4.3	**USER** edits/confirms the minimum outlier threshold	1	3	3
4.4	**USER** edits/confirms the maximum outlier threshold	1	3	3
4.5	**USER** opens Feature Extraction Method field	1	1	1
4.6	**USER** edits/confirms the Feature Extraction Method	1	3	3
5	**USER** opens Phase Selection field	1	1	1
5.1	**USER** selects Phase (development / production / evaluation)	1	3	3
6	**USER** selects *Save Configuration* to store the configuration	1	1	1
7	**SYSTEM** validates the configuration and shows a confirmation message			
8	**USER** closes the configuration interface	1	1	1
TOTAL (O × CE) COST		34		
USER is a Machine Learning Engineer: Normalized Salary		× 1.67		
TOTAL HUMAN COST		**56.78**		

Preparation System Configuration

System Addresses

Ingestion System IP Address

192.168.1.101

Classification System IP Address

192.168.1.102

Segregation System IP Address

192.168.1.103

Phase Selection

Development | Production | Evaluation

Preprocessing Parameters

Missing Samples Correction Method

Drop Missing

Absolute Outlier Detection Threshold

Minimum: -2.5

Maximum: 2.5

Feature Extraction Method

Principal Component Analysis (PCA)

Close | Save Configuration

4.3.3. Configure the Segregation System (Rojan)

	STEP	O	CE	O × CE
1	**USER** opens the "*Segregation System Configuration*" form	1	1	1
2	**SYSTEM** loads the configuration interface with default parameters			
3	**FOR EACH** input fields of the System Addresses (IP address of Development, Messaging and Preparation System)	3		
3.1	**USER** edits/confirms the IP address		3	9
4	**FOR EACH** input fields of the Session Sufficiency (Number of Requested Sessions and Sufficient Sessions Threshold)	2		
4.1	**USER** edits/confirms the parameter		3	6
5	**USER** edits/confirms the Class Balancing Tolerance	1	3	3
6	**FOR EACH** input fields of the learning set split percentages (Training, Validation and Test)	3		
6.1	**USER** edits/confirms the percentages (%)		3	9
7	**USER** selects *Save* to store the configuration	1	1	1
8	**SYSTEM** validates the configuration and shows a confirmation message			
9	**USER** closes the configuration interface	1	1	1
TOTAL (O × CE) COST		30		
USER is a Machine Learning Engineer: Normalized Salary		× 1.67		
TOTAL HUMAN COST		**50.1**		

SEGREGATION SYSTEM CONFIGURATION

EXIT

SYSTEM ADDRESSES

Development System IP: 172.16.32.1

Messaging System IP: 172.16.32.2

Preparation System IP: 172.16.32.3

SESSION SUFFICIENCY

Number of Requested Sessions: 500

Sufficient Sessions Threshold: 500

BALANCING PARAMETERS

Max Class Imbalance Tolerance (%): 5

LEARNING SET SPLIT PERCENTAGES

Training Set (%): 70

Validation Set (%): 15

Test Set (%): 15

SAVE

4.3.4. Configure the Development System (Ronald)

	STEP	O	CE	O × CE
1	**USER** opens the "*Development System Configuration*" form	1	1	1
2	**SYSTEM** loads the configuration interface with default parameters			
3	**USER** reviews the connected system IP addresses	1	2	2
3.1	**USER** edits/confirms Segregation System IP address	1	3	3
3.2	**USER** edits/confirms the Messaging System IP address	1	3	3
3.3	**USER** edits/confirms Classification System IP address	1	3	3
4	**USER** reviews the hyperparameter settings	1	2	2
4.1	**USER** opens Hyperparameter Grid field	1	1	1
4.2	**USER** edits/confirms *min_layers*	1	3	3
4.3	**USER** edits/confirms *max_layers*	1	3	3
4.4	**USER** edits/confirms *step_layers*	1	3	3
4.5	**USER** edits/confirms *min_neurons*	1	3	3
4.6	**USER** edits/confirms *max_neurons*	1	3	3
4.7	**USER** edits/confirms *step_neurons*	1	3	3
4.8	**USER** edits/confirms Maximum Number of Iterations	1	3	3
4.9	**USER** edits/confirms Overfitting Tolerance	1	3	3
5	**USER** reviews the validation parameters	1	2	2
5.1	**USER** edits/confirms the Validation Threshold	1	3	3
6	**USER** reviews the test parameters	1	2	2
6.1	**USER** edits/confirms the Test Pass Threshold	1	3	3
7	**USER** selects *Save Configuration* to store the configuration	1	1	1
8	**SYSTEM** validates the configuration and shows a confirmation message			
9	**USER** closes the configuration interface	1	1	1
TOTAL (O × CE) COST		51		
USER is a Machine Learning Engineer: Normalized Salary		× 1.67		
TOTAL HUMAN COST		**85.17**		

Development System Configuration

System IP Addresses

Segregation System IP Address

192.168.1.110

Messaging System IP Address

192.168.1.111

Classification System IP Address

192.168.1.112

Hyperparameter Grid

Min Layers	Max Layers	Step Layers
1	5	1

Min Neurons	Max Neurons	Step Neurons
32	256	32

Training Parameters

Maximum Number of Iterations

1000

Overfitting Tolerance

0.05

Validation Parameters

Validation Threshold

0.85

Test Parameters

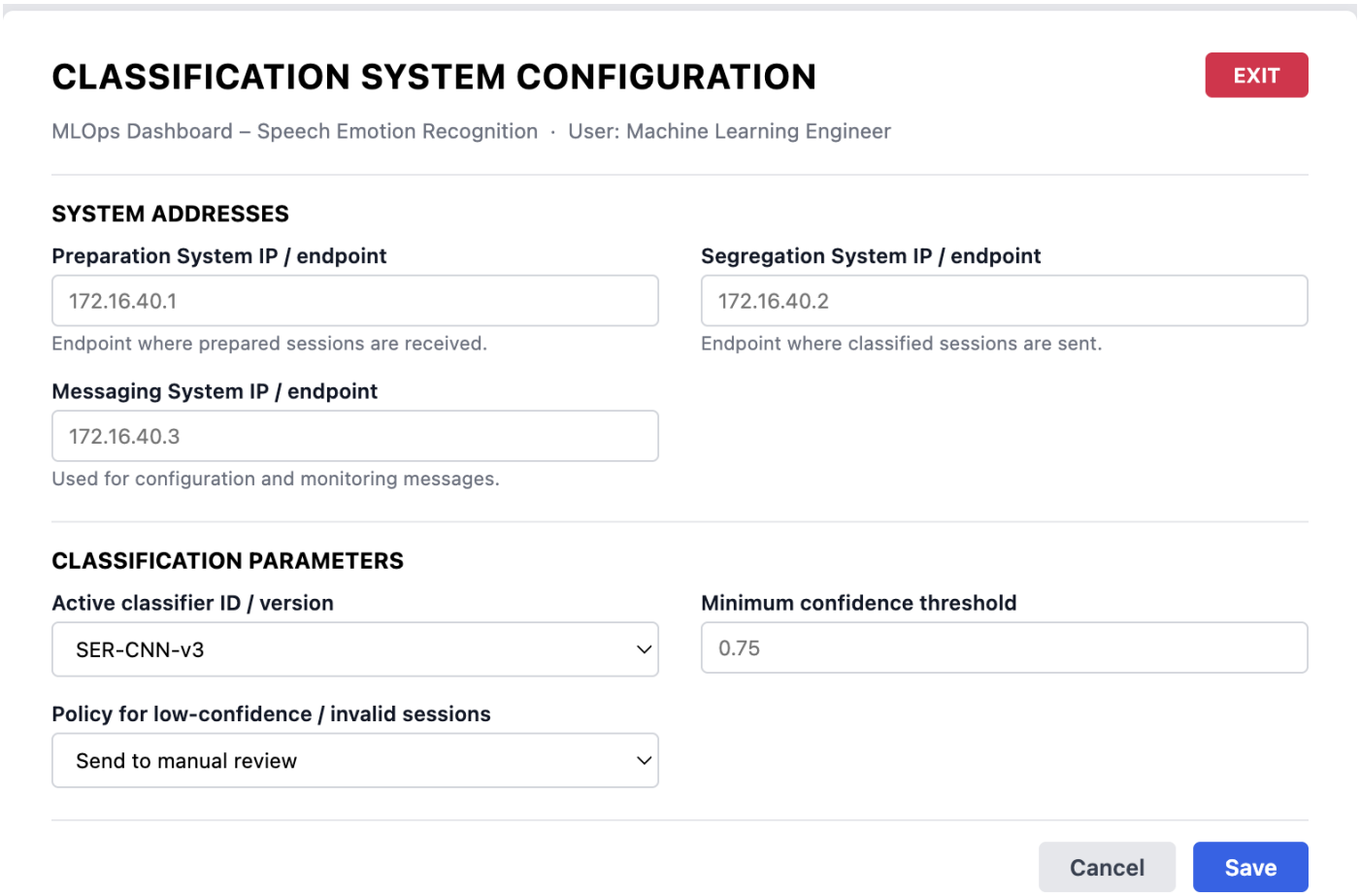
Test Pass Threshold

0.90

Close | Save Configuration

4.3.5. Configure the Classification System (Nimra)

	STEP	O	CE	O × CE
1	**USER** opens the "*Classification System Configuration form*".	1	1	1
2	**SYSTEM** loads the configuration interface with current/default parameters.			
3	**FOR EACH** input fields of the System Addresses (IP address of Development, Messaging, Preparation, Evaluation and Client-Side System)	5		
3.1	**USER** edits/confirms the IP address		3	15
4	**USER** edits/confirms the lifecycle phase	1	3	3
5	**USER** selects *Save to store the configuration*.	1	1	1
6	**SYSTEM** validates the configuration and shows a confirmation message.			
6.1	**USER** closes the configuration interface.	1	1	1
TOTAL (O × CE) COST		21		
USER is a Machine Learning Engineer: Normalized Salary		× 1.67		
TOTAL HUMAN COST		**35.07**		



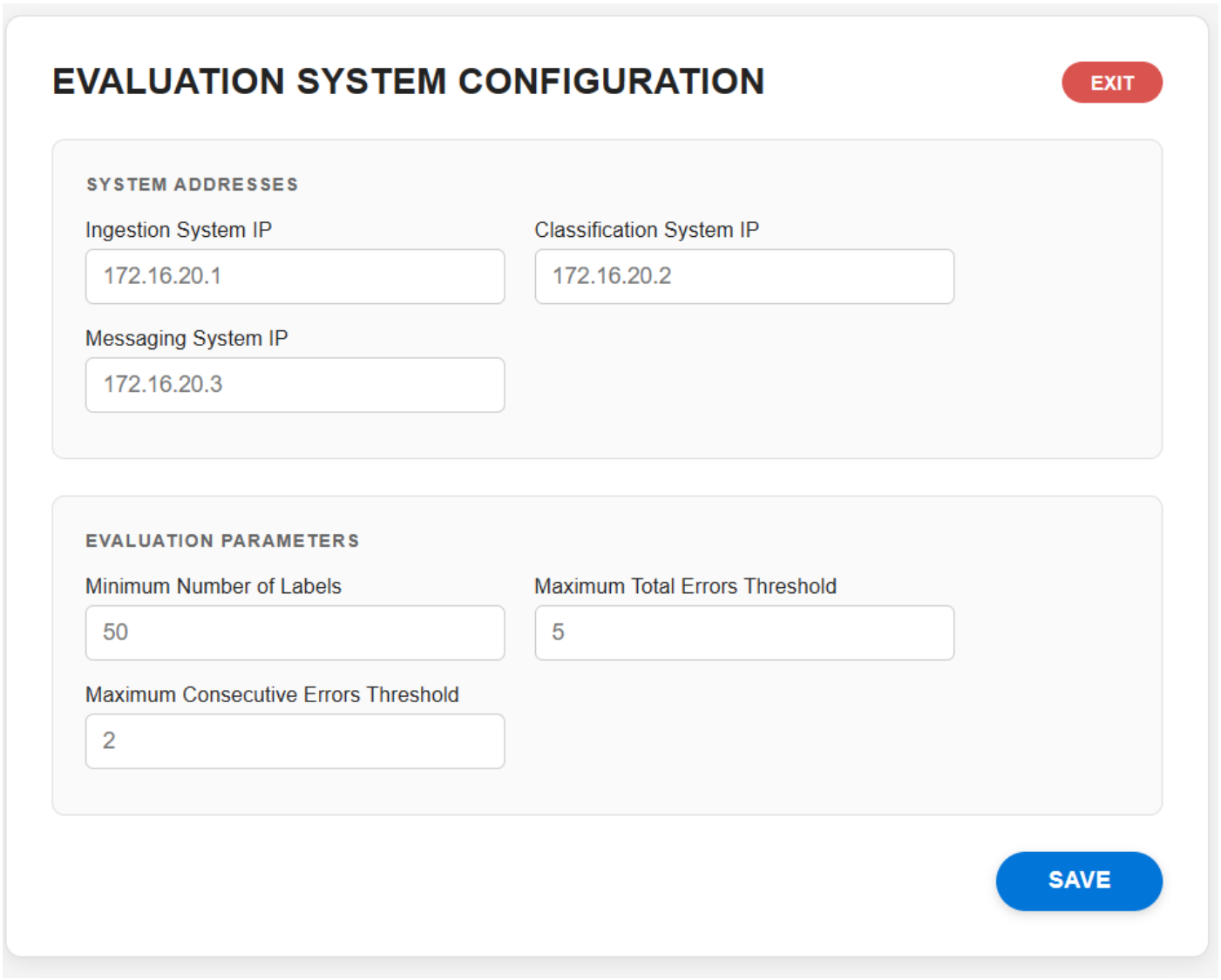
FEEDBACK

Parameter	Purpose
evaluation_sys_addr	Where to send labels during evaluation phase
client_side_sys_addr	Where to send labels during normal production
phase	Controls routing (evaluation vs normal mode)

YOU DON'T HAVE TO REDO THE COST TABLE. WE ALREADY UPDATED.

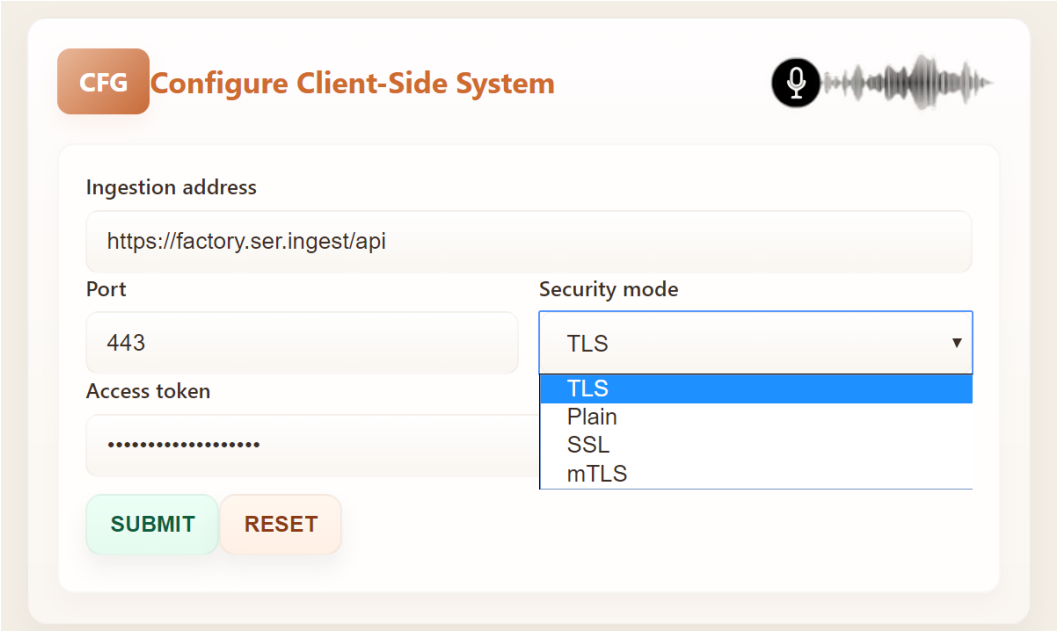
4.3.6. Configure the Evaluation System (Rojan)

	STEP	O	CE	O × CE
1	**USER** opens the *Evaluation System Configuration* form	1	1	1
2	**SYSTEM** loads the configuration interface with default parameters			
3	**FOR EACH** input fields of the System Addresses (IP address of Ingestion, Classification and Messaging System)	3		
3.1	**USER** edits/confirms the IP address		3	9
4	**FOR EACH** input fields of the Evaluation parameters (Minimum Number of Labels, Maximum Total Errors Threshold and Maximum Consecutive Errors Threshold)	3		
4.1	**USER** edits/confirms the parameter		3	9
5	**USER** selects *Save to store the configuration*	1	1	1
8	**SYSTEM** validates the configuration and shows a confirmation message			
9	**USER** closes the configuration interface	1	1	1
TOTAL (O × CE) COST		21		
USER is a Data Analyst: Normalized Salary		× 1.45		
TOTAL HUMAN COST		**30.45**		


EVALUATION SYSTEM CONFIGURATION
EXIT
SYSTEM ADDRESSES
Ingestion System IP
172.16.20.1
Classification System IP
172.16.20.2
Messaging System IP
172.16.20.3
EVALUATION PARAMETERS
Minimum Number of Labels
50
Maximum Total Errors Threshold
5
Maximum Consecutive Errors Threshold
2
SAVE

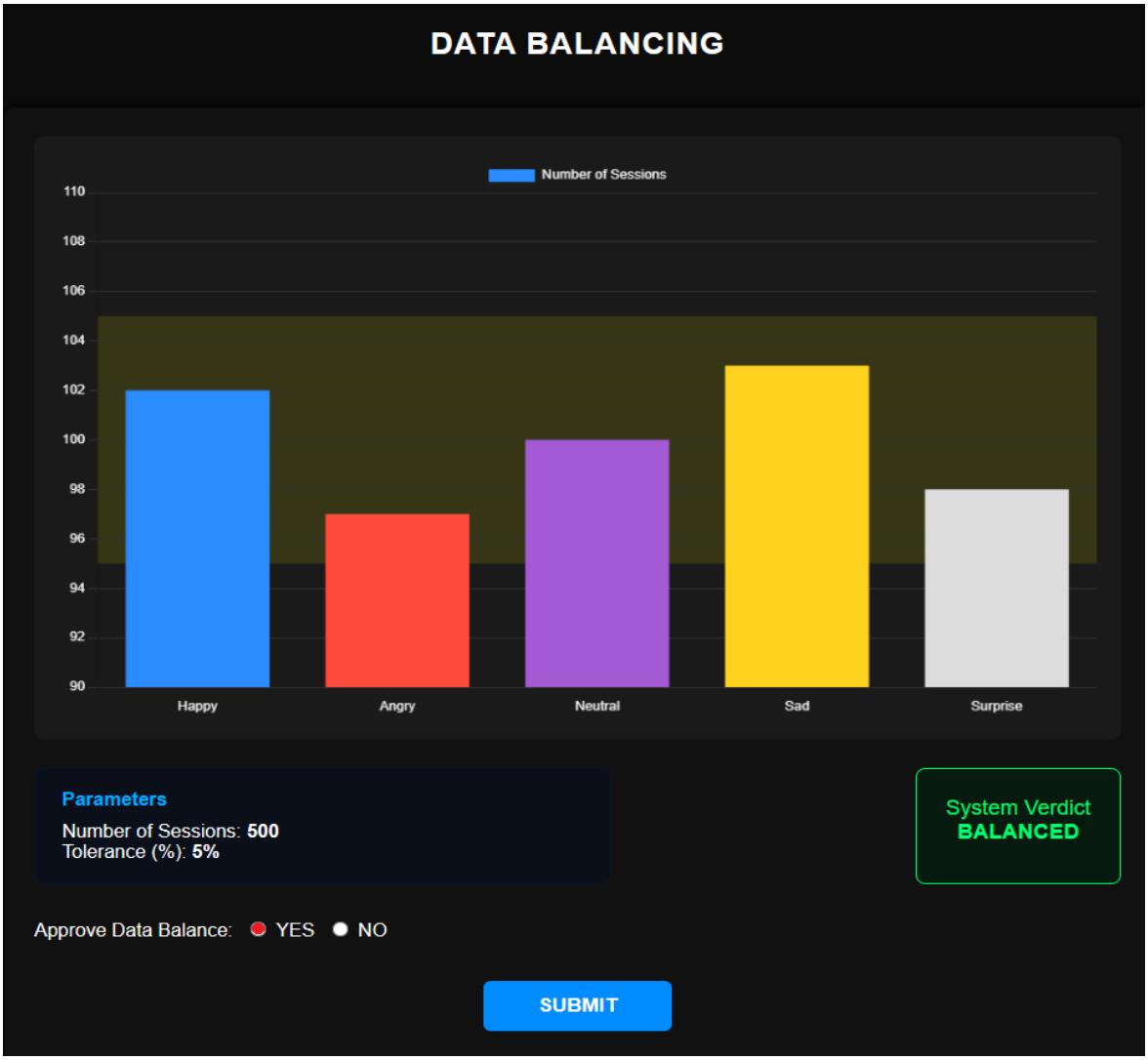
4.3.7. Configure the Client-Side System (Francesco)

	STEP	O	CE	O × CE
1	**USER** opens the "*Configure Client-Side System form*"	1	1	1
2	**SYSTEM** displays current configuration			
3	**USER** reviews the ingestion address	1	2	2
3.1	**USER** sets the ingestion address	1	3	3
4	**USER** reviews the port	1	2	2
4.1	**USER** sets the port	1	3	3
5	**USER** sets the access token	1	3	3
6	**USER** selects the security mode	1	1	1
6.1	**USER** sets the security mode	1	3	3
7	**USER** selects **SUBMIT**	1	1	1
8.1	**IF** the configuration is correct	0.95		
8.1.1	**SYSTEM** displays a confirmation notice			
8.1.2	**SYSTEM** *saves the new configuration*			
8.2	**ELSE**	0.05		
8.2.1	**SYSTEM** displays an error message and aborts			
9	**USER** closes the form	1	1	1
TOTAL (O × CE) COST		20		
USER is a Support Technician: Normalized Salary		× 1		
TOTAL HUMAN COST		**20**		



4.3.8. Check Data Balancing (Rojan)

	STEP	O	CE	O × CE
1	**USER** opens the data balancing form	1	1	1
2	**SYSTEM** loads the report (a bar chart with five bars, one for each emotion class)			
3	**FOR EACH *class***	5		2
3.1	**USER** reads the number of sessions of the *class*		3	15
3.2	**USER** reads the tolerance band (±5%)		3	15
3.3	**USER** compares the *class* sessions count with the tolerance band		3	15
3.4	**USER** determines if the *class* is **BALANCED**		3	15
4.1	**IF** data is balanced	0.2		
4.1.1	**USER** selects **YES**		1	0.2
4.2	**ELSE**	0.8		
4.2.1	**USER** selects **NO**		1	0.8
4.2.2	**SYSTEM** shows "*Send Configuration*" section			
4.2.3	**USER** fills required fields		3	2.4
5	**USER** selects **SUBMIT**	1	1	1
6	**SYSTEM** shows confirmation and stores the outcome			
7	**USER** closes the form	1	1	1
TOTAL (O × CE) COST		68.4		
USER is a Data Analyst: Normalized Salary		× 1.45		
TOTAL HUMAN COST		**99.18**		

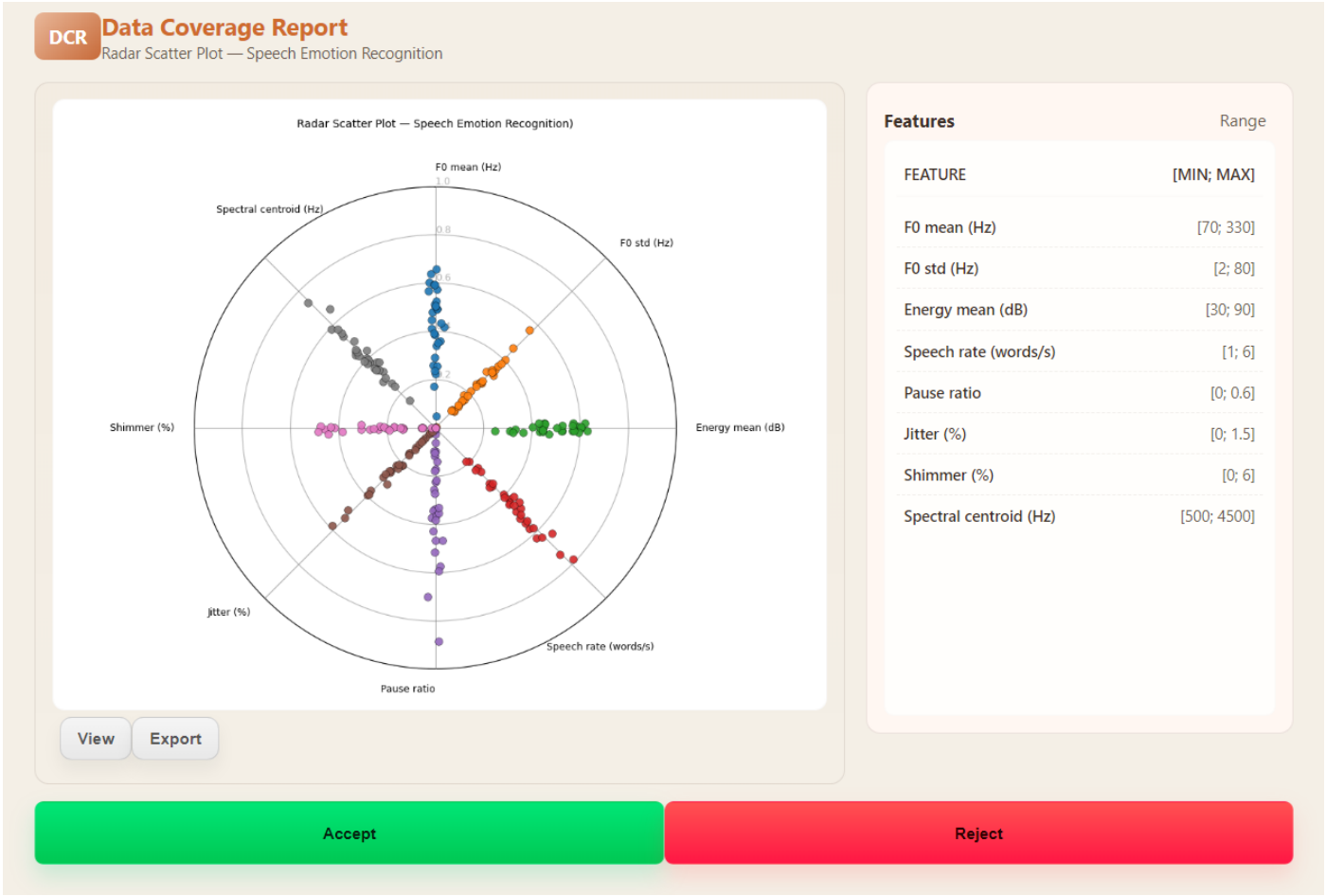

DATA BALANCING
Number of Sessions
110
108
106
104
102
100
98
96
94
92
90
Happy
Angry
Neutral
Sad
Surprise
Parameters
Number of Sessions: 500
Tolerance (%): 5%
System Verdict
BALANCED
Approve Data Balance: YES NO
SUBMIT

4.3.9. TASK Check Input Coverage (Francesco)

	STEP	O	CE	O × CE
1	**USER** opens the "*Coverage Report*"	1	1	1
2	**SYSTEM** loads the report (a radar scatter plot)			
3.1	**FOR EACH** feature	8		
3.1.1	**USER** check if the feature on the radar plot has the correct distribution on the radius		4	32
4.1	**IF** the input coverage is satisfied	0.33		
4.1.1	**USER** selects **ACCEPT**		1	0.33
4.2	**ELSE**	0.67		
4.2.1	**USER** selects **REJECT**		1	0.67
5	**SYSTEM** shows confirmation and stores the outcome			
6	**USER** closes the form	1	1	1
TOTAL (O × CE) COST		35		
USER is a Data Analyst: Normalized Salary		× 1.45		
TOTAL HUMAN COST		**50.75**		

Choosen features:

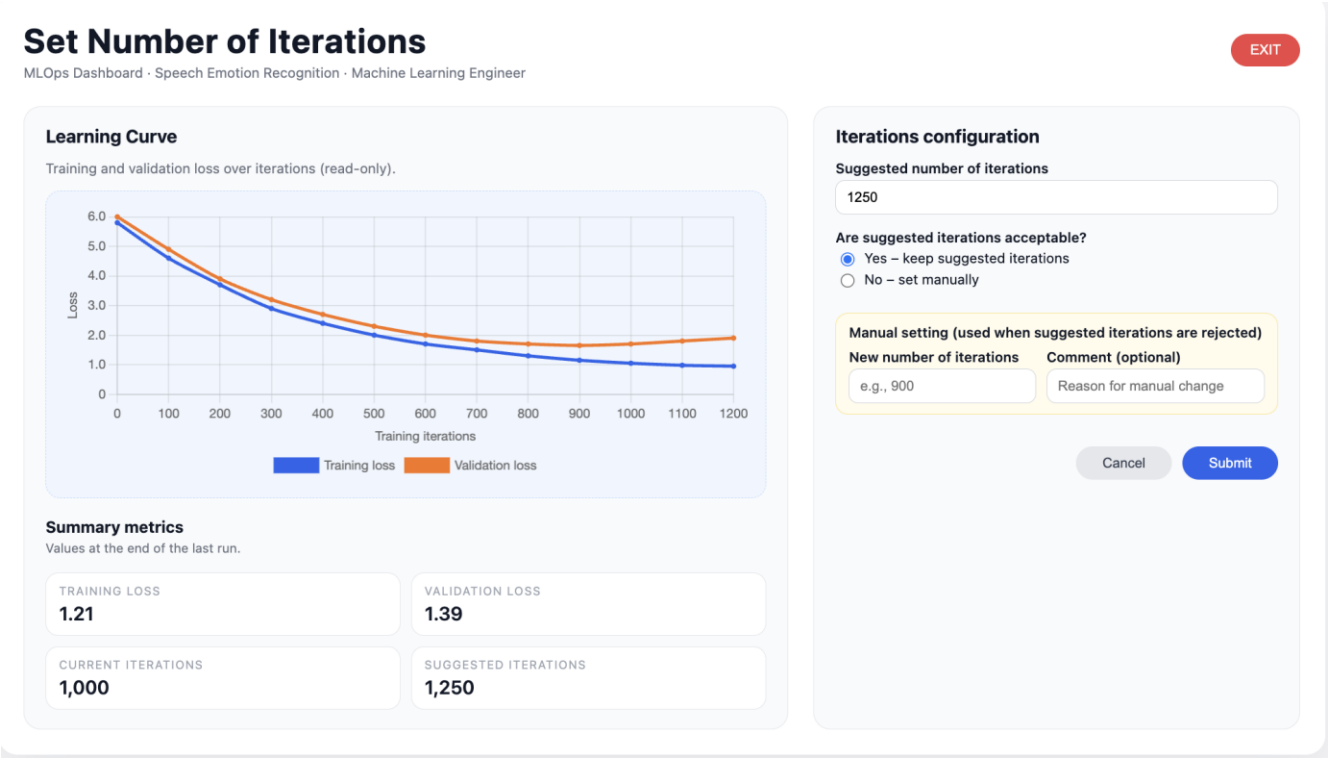
1. **F0 mean (Hz):** Average of the fundamental frequency of the utterance. Interpretable as the overall "tone" of the voice;
2. **F0 std (Hz):** Standard deviation of the F0. Measures tonal variability (flat vs. modulated intonation);
3. **Energy mean (dB):** Average of perceived energy. Indicates arousal (high = anger/enthusiasm, low = sadness);
4. **Speech rate (words/syllables per second):** Rate of speech calculated per word or syllable. Interpretable as agitation vs. calm;
5. **Pause ratio (duration of silence / total duration in seconds):** Percentage of silent time. Useful for distinguishing frequent pauses or fragmented speech;
6. **Jitter (relative, %):** Cycle-to-cycle variation in frequency. Indicates instability/tension in the voice;
7. **Shimmer (relative, %):** Cycle-to-cycle variation in amplitude. Reports tremor or vocal fatigue;
8. **Spectral centroid (Hz):** "Center of mass" of the spectrum. Measures the brightness/timbre of the voice (higher = "brighter").



In this example, the graph shows that the data covers the entire acoustic feature space well, with no gaps or marked outliers (except for something in Pause Ratio and Spectral Centroid), and reveals which features exhibit greater or lesser variability (little for standard F0, much for speech rate and spectral centroid). The plot serves to verify feature completeness, variability, and quality before modeling.

3.3.10. TASK Set Number of Iterations (Nimra)

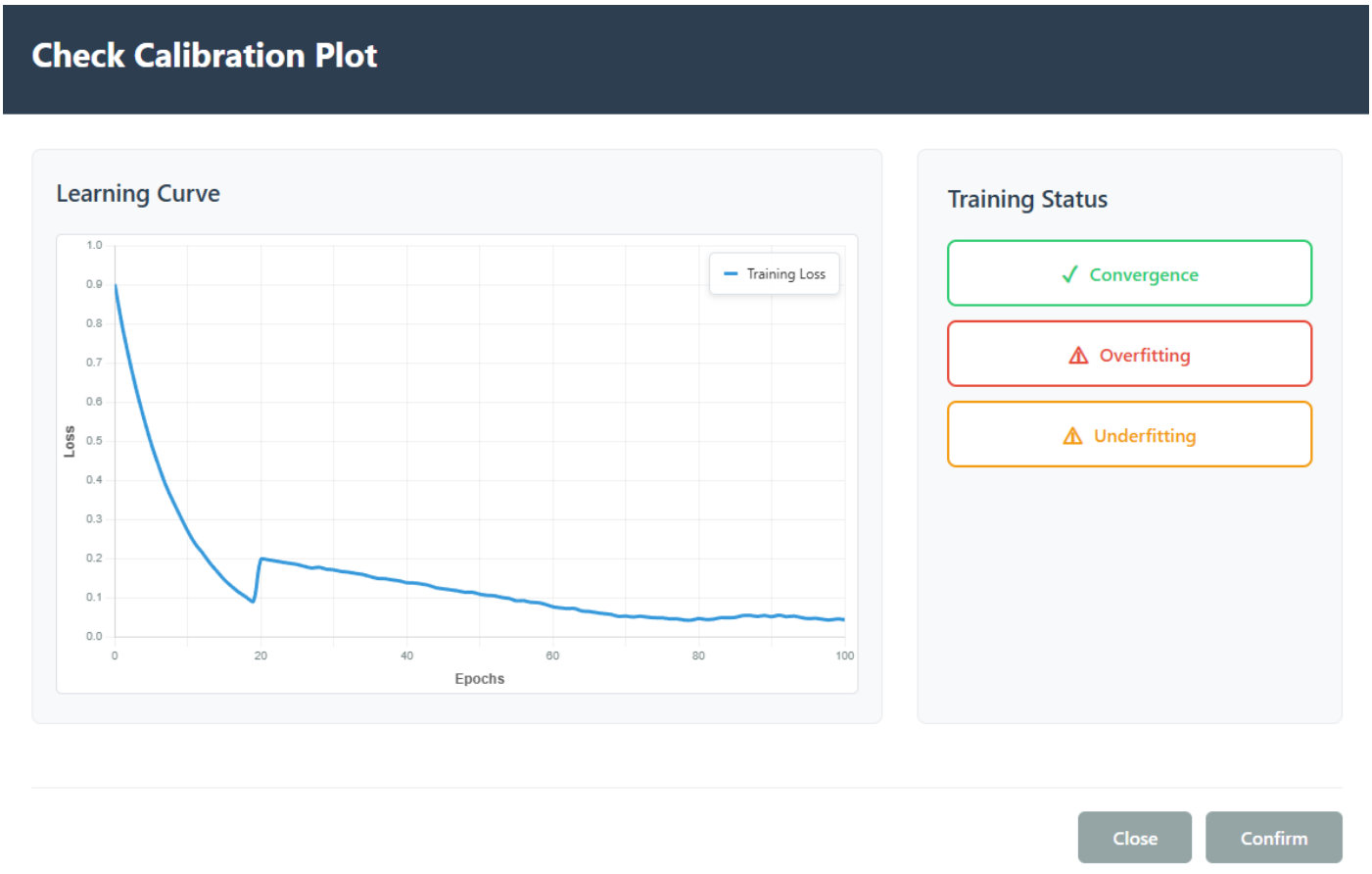
	STEP	O	CE	O × CE
1	**USER** opens *"Set Number of Iterations"* form	1	1	1
2	**SYSTEM** loads a default/current number of iterations			
3	**USER** see the previous number of iterations	1	2	2
4	**USER** edits/confirms the number of iterations based on task complexity and personal experience	1	3	3
4	**USER** selects "Submit" button to confirm the number of iterations	1	1	1
5	**SYSTEM** shows confirmation and stores new value			
6	**USER** closes the form	1	1	1
TOTAL (O × CE) COST		8		
USER is a Machine Learning Engineer: Normalized Salary		× 1.67		
TOTAL HUMAN COST		**13.36**		



FEEDBACK:
No graph just a set num of iteration (Please refer to past proj)

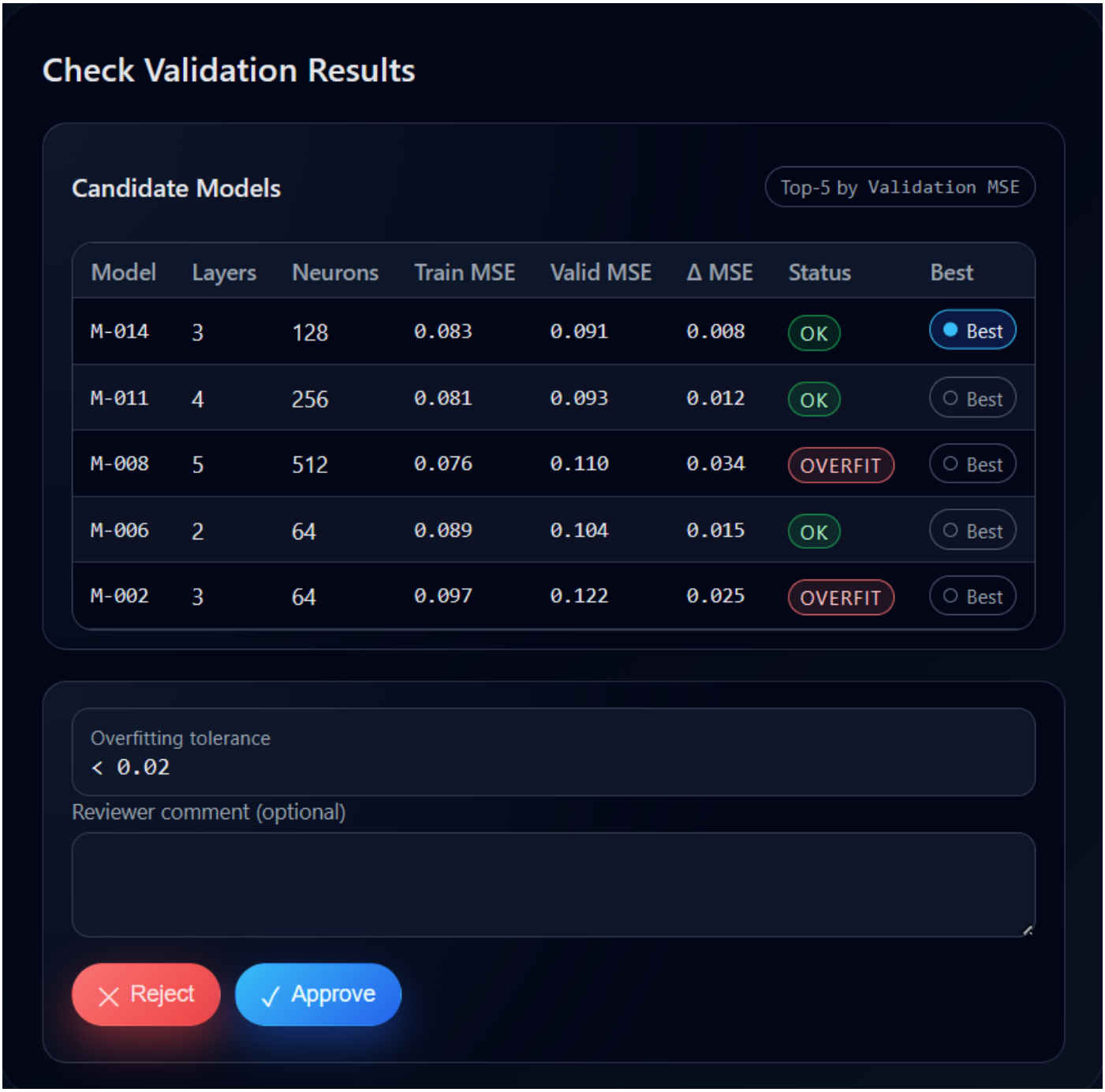
3.3.11. TASK Check Calibration Plot (Ronald)

STEP	ACTION	O	CE	O × CE
1	**USER** opens the *"Check Calibration Plot"* form	1	1	1
2	**SYSTEM** loads the Learning Curve plot (training loss curve)			
3	**USER** inspects Learning Curve plot for convergence, overfitting or underfitting	1	4	4
3.1	**IF** the Learning Curve plot converges (loss stabilizes)	0.4		
3.1.1	**USER** confirms by selecting *"Convergence"* button		1	0.4
3.1.2	**SYSTEM** generates and shows validation report			
3.2	**ELSE**	0.6		
3.2.1	**USER** selects *"Overfitting"* or *"Underfitting"* button		1	0.6
4	**USER** selects *"submit"*	1	1	1
5	**SYSTEM** shows confirmation message for submission			
6	**USER** closes form	1	1	1
TOTAL (O × CE) COST		8		
USER is a ML Engineer: Normalized Salary		× 1.67		
TOTAL HUMAN COST		**13.36**		



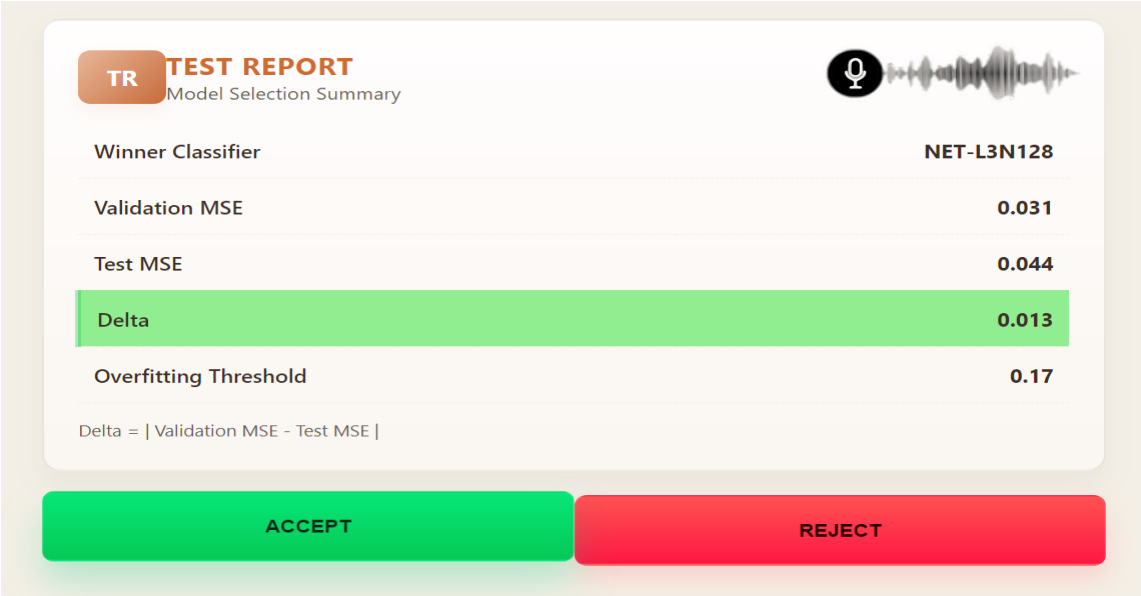
3.3.12. TASK Check Validation Results (Rojan)

	STEP	O	CE	O × CE
1	**USER** opens validation results form	1	1	1
2	**SYSTEM** loads validation report			
3	**FOR EACH** model	5		
3.1	**USER** checks whether delta MSE (the difference between Validation and Training MSE) for given model is under the overfitting tolerance		3	15
3.2	**USER** marks the model **OK** or **Overfit** based on the delta MSE		3	15
4.1	**IF** validation results acceptable	0.95		
4.1.1	**USER** identifies the best network (the lowest validation error marked **OK**)		3	2.85
4.1.2	**USER** identifies the second-best network (the second lowest validation error marked **OK**)		3	2.85
4.1.3	**USER** compares the validation error between the two networks		3	2.85
4.1.4	If the error is very similar (the difference is one order of magnitude w.r.t. their error), **USER** selects the **network with the lowest complexity** otherwise selects **the best network**		4	3.8
4.2	**ELSE**	0.05		
4.2.1	**USER** rejects the result		1	0.05
4.2.2	**SYSTEM** shows "*Reason for Rejection*" section			
4.2.3	**USER** provides the feedback for the rejection		3	0.15
5	**USER** selects **SUBMIT**	1	1	1
6	**SYSTEM** shows confirmation and stores decision			
7	**USER** closes the form	1	1	1
TOTAL (O × CE) COST		45.55		
USER is a **ML Engineer: Normalized Salary**		× 1.67		
TOTAL HUMAN COST		**76.0685**		


Check Validation Results
Candidate Models
Top-5 by Validation MSE
Model Layers Neurons Train MSE Valid MSE Δ MSE Status Best
M-014 3 128 0.083 0.091 0.008 OK Best
M-011 4 256 0.081 0.093 0.012 OK Best
M-008 5 512 0.076 0.110 0.034 OVERFIT Best
M-006 2 64 0.089 0.104 0.015 OK Best
M-002 3 64 0.097 0.122 0.025 OVERFIT Best
Overfitting tolerance
< 0.02
Reviewer comment (optional)
Reject
Approve

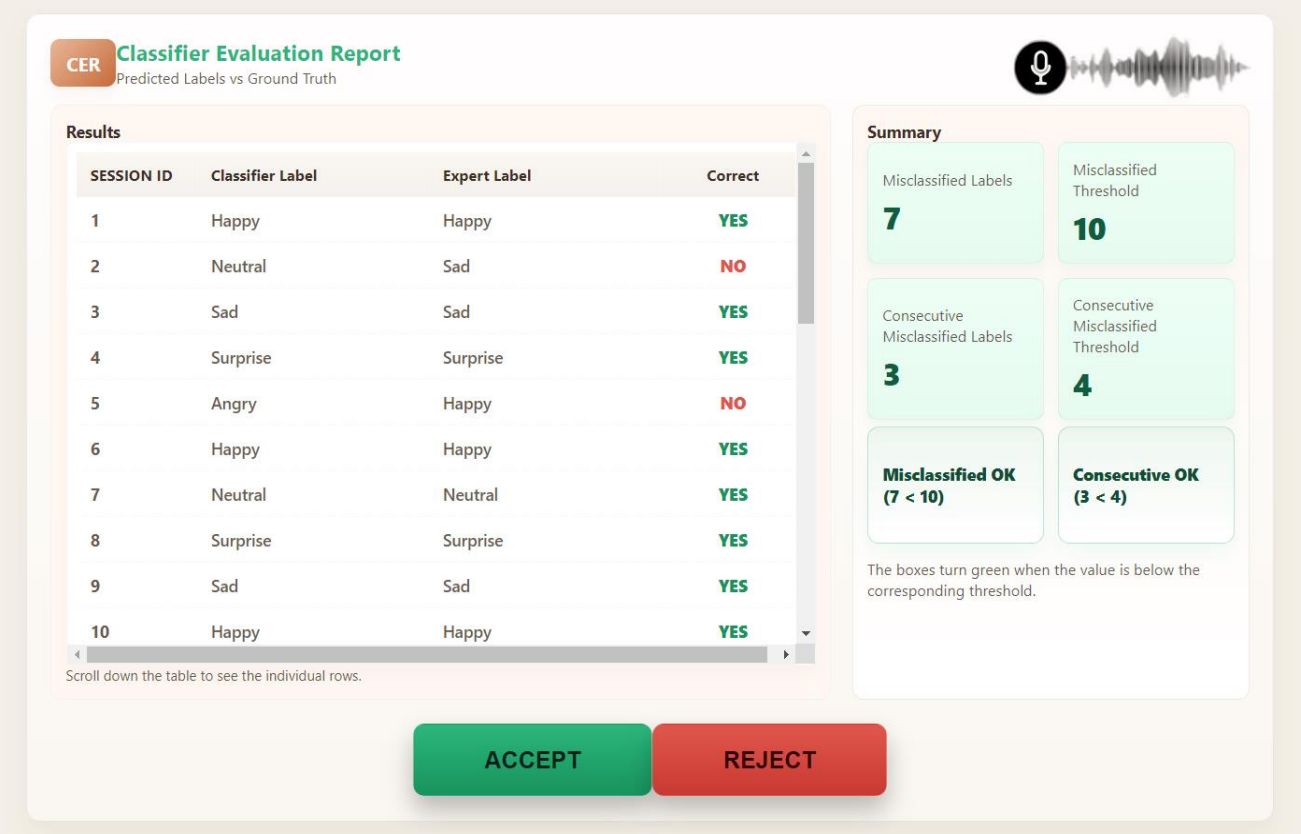
3.3.13. TASK Check Test Results (Francesco)

	STEP	O	CE	O × CE
1	**USER** opens "*Test Results form*"	1	1	1
2	**SYSTEM** shows the test results			
3	**USER** checks Delta	1	3	3
4	**USER** checks overfitting tolerance	1	3	3
5	**USER** checks if the difference between the test results and the validation results (Delta) is below the overfitting tolerance	1	3	3
6.1	**IF** the test results are not satisfactory.	0.99		
6.1.1	**USER** selects **ACCEPT**		1	0.99
6.1.2	**SYSTEM** saves *Classifier*			
6.2	**ELSE**	0.01		
6.2.1	**USER** selects **REJECT**		1	0.01
6.2.2	**SYSTEM** sends configuration message to messaging system			
7	**SYSTEM** shows confirmation and stores decision			
8	**USER** closes the form		1	1
TOTAL (O × CE) COST		12		
USER is a ML Engineer: Normalized Salary		× 1.67		
TOTAL HUMAN COST		**20.04**		



3.3.14. TASK Evaluate Classifier (Francesco)

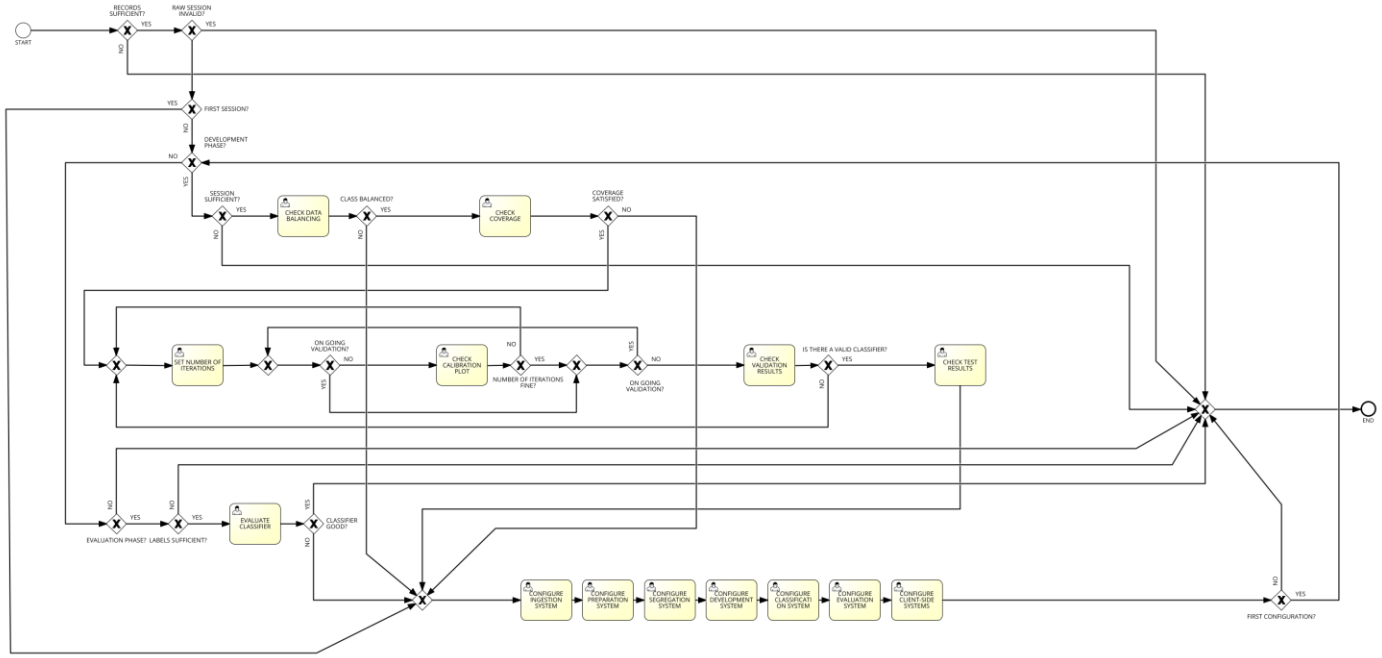
	STEP	O	CE	O × CE
1	**USER** opens the "*Evaluate Classifier Performance form*"	1	1	1
2	**SYSTEM** displays a session table containing the expert label (ground truth) and the classifier label (predicted). Any mismatch between the two indicates an error.			
3	**USER** reads total errors number	1	3	3
4	**USER** reads total errors number threshold	1	3	3
5	**USER** checks if the total errors number is below the threshold	1	3	3
6	**USER** reads total consecutive errors	1	3	3
7	**USER** reads total consecutive errors threshold	1	3	3
8	**USER** checks if the total consecutive errors number is below the threshold	1	3	3
9.1	**IF** a least one error number is above its threshold	0.14		
9.1.1	**USER** selects **REJECT**		1	0.14
9.1.2	**SYSTEM** sends configuration message to messaging system			
9.2	**ELSE**	0.86		
9.2.1	**USER** selects **ACCEPT**		1	0.86
9.2.2	**SYSTEM** marks the *Classifier* as Good			
10	**SYSTEM** shows confirmation and stores decision			
11	**USER** closes the form	1	1	1
TOTAL (O × CE) COST		21		
USER is a Data Analyst: Normalized Salary		× 1.45		
TOTAL HUMAN COST		**30.45**		



Documentation on features:
https://drive.google.com/file/d/1TGj5K0cTLhr3NYvcmxfeUKfatDMmIdQ2/view?usp=sharing

5 Simulation

5.1 Collapsed Workflow



5.2 Token Definition

We choose a Session as a Token because the simulation engine represents each process instance by a moving execution marker. That marker must carry the session identifier through the workflow so the simulator can account for branching logic, resource use, and frequencies. Modeling the session as a token ensures the simulator can track and route each instance correctly, especially when probabilities or conditional paths depend on whether it is the first occurrence or a later one.

5.3 AS-IS Simulation

This are the **Uniform Distribution (±5%)** for Human Tasks' Costs (in alphabetical order):

- **CHECK CALIBRATION PLOT (13.36)** → [12.692; 14.028]
- **CHECK COVERAGE (50.75)** → [48.2125; 53.2875]
- **CHECK DATA BALANCING (99.18)** → [94.221; 104.139]
- **CHECK TEST RESULTS (20.04)** → [19.038; 21.042]
- **CHECK VALIDATION RESULTS (76.0685)** → [72.2651; 79.8719]
- **CONFIGURE CLASSIFICATION SYSTEM (35.07)** → [33.3165; 36.8235]
- **CONFIGURE CLIENT-SIDE SYSTEMS (20)** → [19; 21]
- **CONFIGURE DEVELOPMENT SYSTEM (85.17)** → [80.9115; 89.4285]
- **CONFIGURE EVALUATION SYSTEM (30.45)** → [28.9275; 31.9725]
- **CONFIGURE INGESTION SYSTEM (58.45)** → [55.5275; 61.3725]
- **CONFIGURE PREPARATION SYSTEM (56.78)** → [53.941; 59.619]
- **CONFIGURE SEGREGATION SYSTEM (50.1)** → [47.595; 52.605]
- **EVALUATE CLASSIFIER (30.45)** → [28.9275; 31.9725]
- **SET NUMBER OF ITERATIONS (13.36)** → [12.692; 14.028]

This are the **percentages** of the gateways (in alphabetical order):

GATEWAYS	YES	NO	MOTIVATION
CLASS BALANCED?	20%	80%	Documentation: "Balanced classes in 5 iterations, then 4 occurrences ko and 1 ok, i.e. 1/5 = 20% chance ok".
CLASSIFIER GOOD?	14%	86%	Documentation: "Evaluation fine for 6 iterations, 6 occurrences ok and 1 ko, the 1/7 = 14% ok".
COVERAGE SATISFIED?	33%	67%	Documentation: "Input coverage in 3 iterations, 2 occurrences ko and 1 ok, then 1/3 = 33% ok".
DEVELOPMENT PHASE?	9%	91%	In this gateway, out of 5550, 500 are in the development phase
EVALUATION PHASE?	1%	99%	In this gateway, out of 5550, 50 are in the evaluation phase
FIRST CONFIGURATION?	0.05%	99.95%	Only 5 out of 10,000 input tokens represent a first session, corresponding to the five classifiers that each require exactly one initial session.
FIRST SESSION?	0.05%	99.95%	Only 5 out of 10,000 input tokens represent a first session, corresponding to the five classifiers that each require exactly one initial session.
IS THERE A VALID CLASSIFIER?	95%	5%	Documentation: "Validation result 95% fine".
LABELS SUFFICIENT?	71%	29%	Given 5550 valid sessions and the assumption that these produce five

			proper classifiers, the preceding gateway probabilities imply that 71% of all sessions must contain sufficient labels to satisfy the documentation's assumptions.
NUMBER OF ITERATIONS FINE?	80%	20%	Documentation: "Number of iterations in 5 iterations, then 4 occurrences ko and 1 ok, i.e. 1/5 = 20% chance ok".
ON GOING VALIDATION?	97.5%	2.5%	For the selected hyperparameter grid, the model has 5 possible layer counts (1–5 in steps of 1) and 8 possible neuron settings (32–256 in steps of 32), yielding 40 total configurations. Since only one configuration is under ongoing validation, 39 out of 40 are already validated. This corresponds to 97.5% marked as "Yes" for ongoing validation and 2.5% marked as "No."
RAW SESSION INVALID?	10%	90%	We assume that 90% of the times the raw session is valid.
RECORDS SUFFICIENT?	10%	90%	We assume that 90% of the times the raw session is valid.
SESSION SUFFICIENT?	99%	1%	We set the proportion of sessions with sufficient labels to 99% because, under the document's assumptions, roughly 545 such sessions would be required to obtain five final good classifiers. In this scenario we begin with only 500, which is already below the required amount.