Electrolysis-Agent-Report

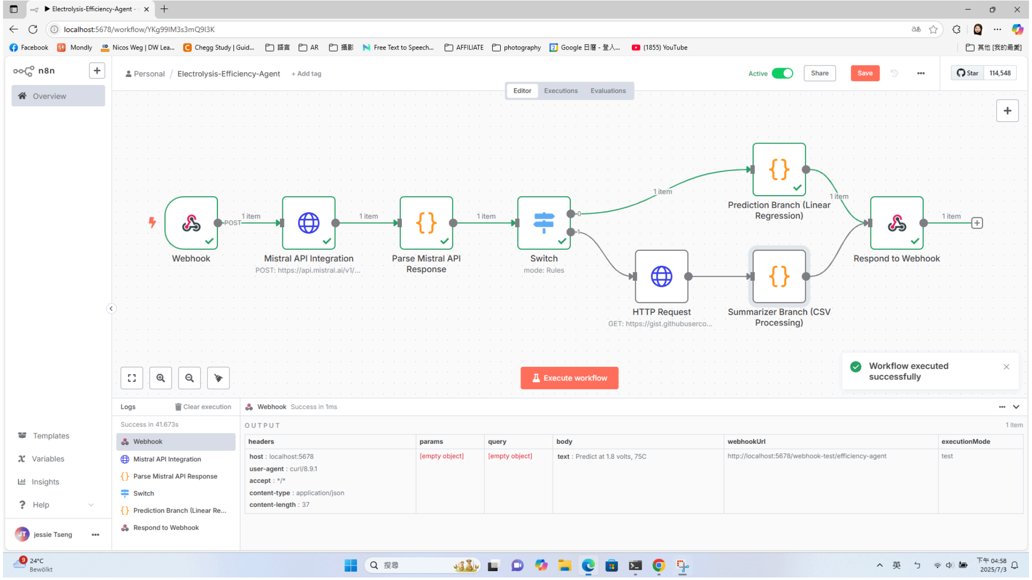
System Architecture

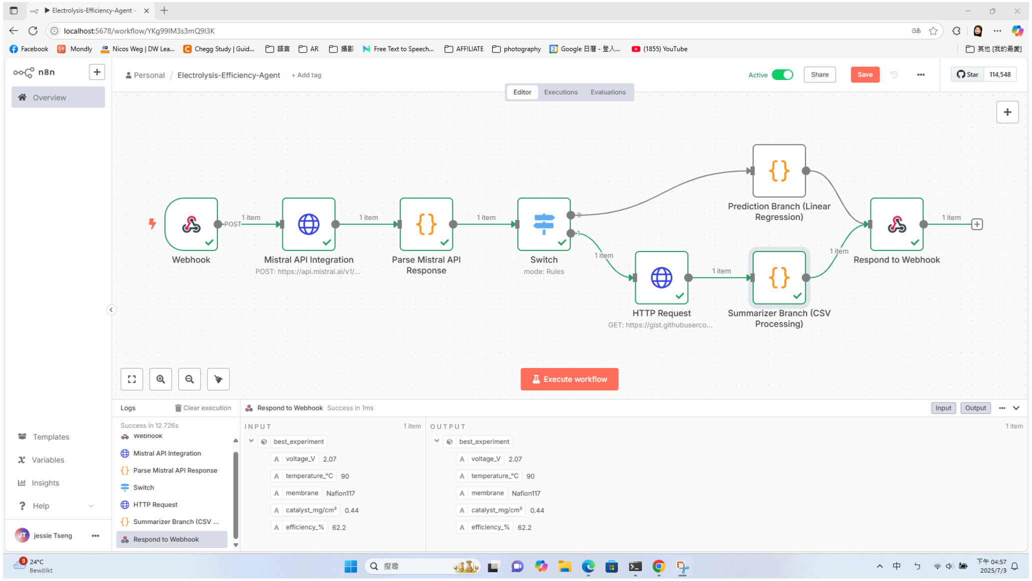
The system is built using **n8n** and consists of the following components:

1. **Webhook**– Accepts free-text requests from users.
2. **Mistral API Integration** – Sends the user's input to a hosted **'mistral-small-latest' LLM model** that interprets the request.
3. **Parse Mistral API Response** – Parses and validates the LLM's response into a clean JSON object
   - `intent` (either `"predict_run"` or `"summarize_best"`)
4. **Switch** – Directs the workflow based on the extracted `intent`.
5. **1.1 Prediction Branch** – Implements a pre-trained linear regression model with fixed coefficients.

   2.1 HTTP Requestre – trieves a public CSV file of 50 experimental runs.

   2.2 Summarizer Branch – parses the CSV content by splitting the string into rows and columns, converting each row into a JSON object, identifying the row with the **highest efficiency_%.**
6. **Respond to Webhook** – Returns JSON output.





Linear Regression Model

1. Data Preprocessing Steps:

Step	Description
Data Source	CSV loaded from public GitHub Gist
Cleaning	Renamed columns, stripped whitespace, and dropped missing values
Features Used	`voltage`, `temperature`
Target Variable	`efficiency`
Scaling	Standardized features using `StandardScaler`
Train/Test Split	80% train, 20% test with `random_state=42`

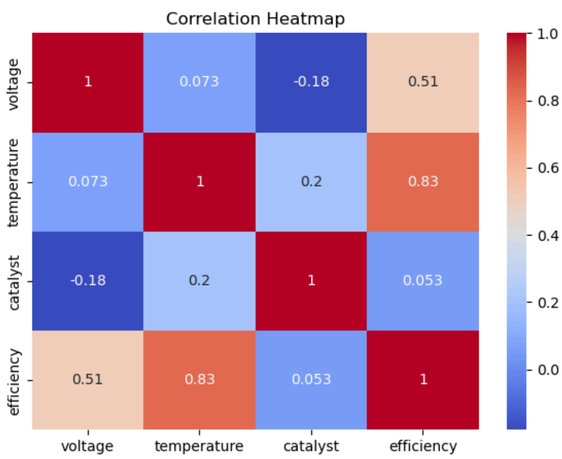


Fig 1. Correlation Heatmap: choose the features

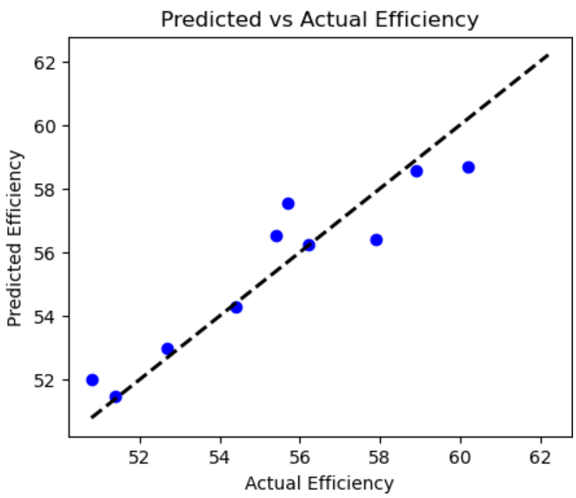
2. Model & Evaluation:

Step	Description
Model Type	Linear Regression inside a pipeline
Scaler Used	`StandardScaler` (removes mean and scales to unit variance)
Test RMSE	Computed using `mean_squared_error` on test set
Cross-Validation	5-fold `KFold` CV with `shuffle=True`
CV RMSE	Mean RMSE across all folds

$$\text{efficiency} = \beta_0 + \beta_1 \cdot \text{voltage} + \beta_2 \cdot \text{temperature}$$

Eq 1. LinearRegression equation, β0 is the **intercept,** β1 is the voltage **learned coefficient**, and β2 is the temperature **learned coefficients.**

> The model shown here uses voltage and temperature as inputs. Alternative models with more features (e.g. catalyst, membrane) were explored but not used in deployment.



- ✅ **Test RMSE**: ~1.04
- ✅ **5-Fold CV RMSE**: ~0.87

✅Final Regression Coefficients f or n8n: {'intercept': np.float64(22.1581), 'voltage': np.float64(10.2944), 'te mperature': np.float64(0.1918)}

Fig 2.Predicted vs Actual Plot

3. Exported Coefficients

These coefficients are recovered from the standardized model back to the **original scale**, suitable for integration:

```
const coefficients = {
  intercept: 22.1581,
  voltage: 10.2944,
  temperature: 0.1918
};
```

Example Inputs & Outputs

Prompt	Response
`Predict run at 1.85V and 80C`	`{"prediction": "58.53%", "parameters": {"voltage": 1.85, "temperature": 80}}`
`Summarize the best efficiency run`	`{"best_experiment": {"voltage_V": 2.07, "efficiency_%": 62.2, ...}}`
What is the expected efficiency at 1.7V and 60°C?	[{"prediction": "51.17%","parameters": {"voltage": 1.7,"temperature": 60},"model_info": "Linear Regression (voltage & temp)"}]