02 – SoH from EIS (Rashid et al.)

Chouaib (github.com/ChouaibB)

December 19, 2025

Contents

1. Data source: Rashid SoH-from-EIS dataset

For this notebook I use the public dataset:

> Rashid, Muhammad; Faraji-Niri, Mona; Sansom, Jonathan; Sheikh, Muhammad; Widanage, Dhammika; Marco, James (2023),
> **"Dataset for rapid state of health estimation of lithium batteries using EIS and machine learning: Training and validation"**,
> *Data in Brief*, 48, 109157, doi: 10.1016/j.dib.2023.109157.
> Available at: https://data.mendeley.com/datasets/mn9fb7xdx6/3
> Original data: **"DIB_Data"**, Mendeley Data, V3, doi: 10.17632/mn9fb7xdx6.3 (license: **CC0 1.0**).

Dataset highlights:

- 25 cylindrical Li-ion cells, aged from **SoH 100% down to 80%** in 5 steps (100, 95, 90, 85, 80%).
- At each SoH stage, reference performance tests (capacity / SoH) and **electrochemical impedance spectroscopy (EIS)** at multiple SOC and temperature conditions.
- Designed specifically to study **fast SoH estimation from EIS with machine learning**.

In this notebook I will:

- Build a **tidy table of EIS measurements** with associated SoH, SOC and temperature.
- Engineer compact **impedance features** (real/imaginary parts, magnitude, phase at selected frequencies).
- Train and evaluate **SoH regression models** using these EIS features.

Folder structure in this repo (after placing the dataset under `./data/`):

- `../data/DIB_Data/`
  - `.csvfiles/`
    - `Capacity_Check/` — capacity check results as CSV/Excel files.
    - `EIS_Test/` — EIS measurements as Excel (`.xls`) files (used in this notebook).
  - `.matfiles/`
    - `Capacity_Check/` — capacity check results as MATLAB files.
    - `EIS_Test/` — EIS measurements as MATLAB files (not used here).
  - `WholeDataRealSOH.mat` — MATLAB table summarising all tests (not used here).
  - `CapacityVsCycleNumber.xlsx` — capacity vs cycle reference.
  - `Table 1.docx` — description of the experimental matrix and test conditions.

2. EIS files: indexing & loading

In this section I locate all EIS test files, extract their metadata from the filenames, and load the spectra into a single long-format table.

2.1 List EIS test files EIS measurements are stored as Excel files under:

- `../data/DIB_Data/.csvfiles/EIS_Test/`

Each file corresponds to one EIS test at a specific (cell, SoH, SOC, temperature).

```
360
```

2.2 Parse metadata from filenames Filenames follow the pattern:

`Cell02_95SOH_15degC_05SOC_9505.xls`

From which I extract:

- `cell_id` → `02`
- `soh_pct` → `95`
- `temp_c` → `15`
- `soc_pct` → `05`
- `capacity_code` → `9505` (numeric code related to capacity / SoH, kept as extra metadata)

```
                                              filepath  cell_id  soh_pct  \
0  ../data/DIB_Data/.csvfiles/EIS_Test/Cell02_95S…        2       95
1  ../data/DIB_Data/.csvfiles/EIS_Test/Cell02_95S…        2       95
2  ../data/DIB_Data/.csvfiles/EIS_Test/Cell02_95S…        2       95
3  ../data/DIB_Data/.csvfiles/EIS_Test/Cell02_95S…        2       95
4  ../data/DIB_Data/.csvfiles/EIS_Test/Cell02_95S…        2       95

   temp_c  soc_pct  capacity_code
0      15        5           9505
1      15       20           9505
2      15       50           9505
3      15       70           9505
4      15       95           9505
```

2.3 Sanity checks on metadata I check that:

- There are 360 EIS tests in total.
- SoH, SOC and temperature cover the expected grids.
- There are multiple cells, so I can later split by `cell_id`.

```
Number of EIS tests: 360

Unique cells: [np.int64(2), np.int64(3), np.int64(4), np.int64(5), np.int64(6),
np.int64(12), np.int64(13), np.int64(14), np.int64(15), np.int64(17),
np.int64(18), np.int64(19), np.int64(20), np.int64(21), np.int64(22),
np.int64(23), np.int64(24), np.int64(25), np.int64(26), np.int64(28),
np.int64(29), np.int64(30), np.int64(31), np.int64(32)]

Unique SoH levels: [np.int64(80), np.int64(85), np.int64(90), np.int64(95),
np.int64(100)]

Unique temperatures (°C): [np.int64(15), np.int64(25), np.int64(35)]

Unique SOC levels (%): [np.int64(5), np.int64(20), np.int64(50), np.int64(70),
np.int64(95)]
```

2.4 Load EIS spectra Each EIS file contains one spectrum with three columns:

- column 1: frequency (Hz),
- column 2: real part of impedance Re(Z) (Ω),
- column 3: imaginary part of impedance Im(Z) (Ω).

I load all 360 files into a single long-format DataFrame where each row is a single (cell, SoH, SOC, T, frequency) point:

- `cell_id, soh_pct, temp_c, soc_pct, capacity_code`
- `frequency_hz, z_real_ohm, z_imag_ohm`

```
PosixPath('../data/DIB_Data/.csvfiles/EIS_Test/Cell02_95SOH_15degC_05SOC_9505.xl
s')
```

```
(         0        1        2
 0  10000.0  0.02995  0.03160
 1   7943.0  0.02827  0.02617
 2   6310.0  0.02704  0.02143
 3   5012.0  0.02599  0.01736
 4   3981.0  0.02499  0.01377,
 (61, 3))
```

	frequency_hz	z_real_ohm	z_imag_ohm	cell_id	soh_pct	temp_c	soc_pct	\
0	10000.0	0.02995	0.03160	2	95	15	5	
1	7943.0	0.02827	0.02617	2	95	15	5	
2	6310.0	0.02704	0.02143	2	95	15	5	
3	5012.0	0.02599	0.01736	2	95	15	5	
4	3981.0	0.02499	0.01377	2	95	15	5	

	capacity_code
0	9505
1	9505
2	9505
3	9505
4	9505

```
(21960, 8)
```

	count	mean	std	min	25%	\
frequency_hz	21960.0	797.097582	1951.144325	0.01000	0.316200	
z_real_ohm	21960.0	0.029719	0.007403	0.02261	0.025330	
z_imag_ohm	21960.0	0.000229	0.007416	-0.03523	-0.002455	
cell_id	21960.0	18.291667	9.149339	2.00000	12.750000	
soh_pct	21960.0	90.000000	7.217043	80.00000	85.000000	
temp_c	21960.0	25.000000	8.165152	15.00000	15.000000	
soc_pct	21960.0	48.000000	32.650399	5.00000	20.000000	
capacity_code	21960.0	9034.750000	721.318964	8046.00000	8307.250000	

	50%	75%	max
frequency_hz	10.000000	316.200000	10000.00000

```
z_real_ohm        0.027710     0.031280      0.08710
z_imag_ohm       -0.001167    -0.000125      0.03542
cell_id          19.500000    25.250000     32.00000
soh_pct          90.000000    95.000000    100.00000
temp_c           25.000000    35.000000     35.00000
soc_pct          50.000000    70.000000     95.00000
capacity_code  9067.000000  9612.750000  10000.00000

frequency_hz
61    360
Name: count, dtype: int64
```

2.5 Final count of complete EIS tests Here I summarise how many **distinct EIS tests** I have after loading and basic cleaning, and how this compares to the dataset description:

- number of cells,
- number of EIS tests (unique combinations of cell, SoH, SOC, temperature),
- SoH / SOC / temperature grids,
- typical number of frequency points per spectrum.

```
   n_cells  n_eis_tests           soh_levels         soc_levels temp_levels  \
0       24          360  80, 85, 90, 95, 100  5, 20, 50, 70, 95  15, 25, 35

   freq_points_per_spectrum
0                        61
```

3. SoH labels & basic exploration

In this dataset the SoH label is already encoded in the EIS filenames as a discrete percentage:

- `soh_pct` {80, 85, 90, 95, 100}.

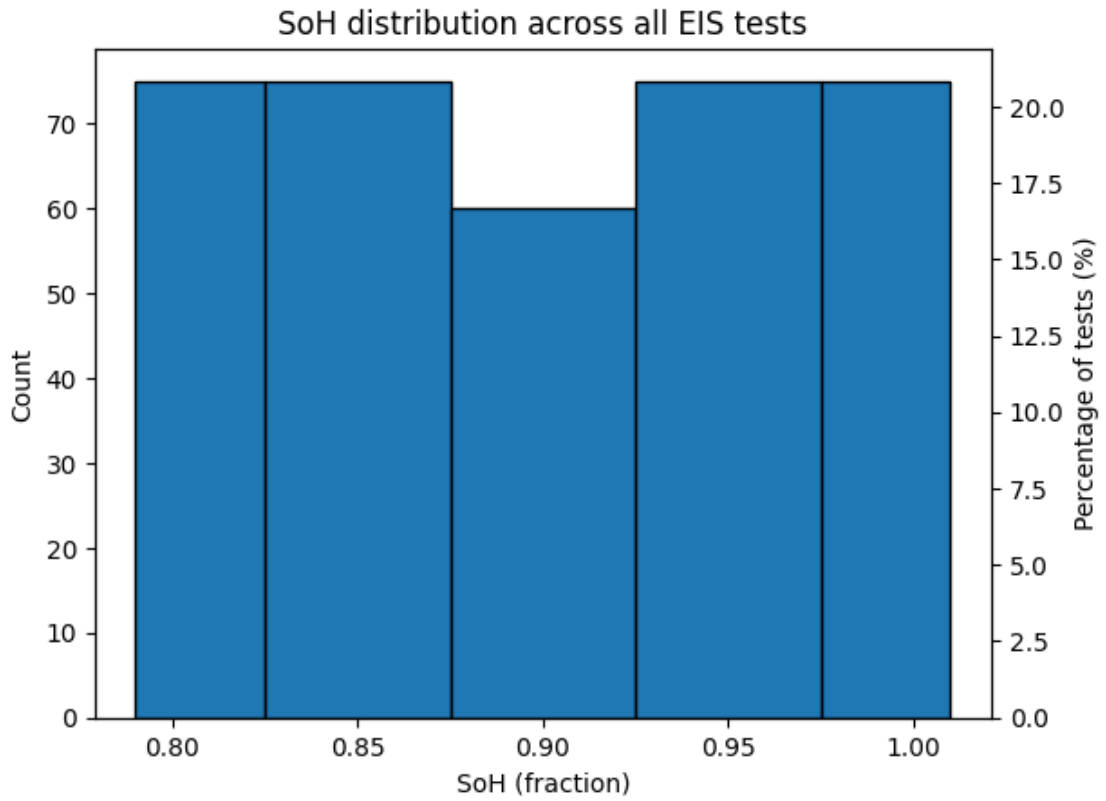
For the modelling part I will:

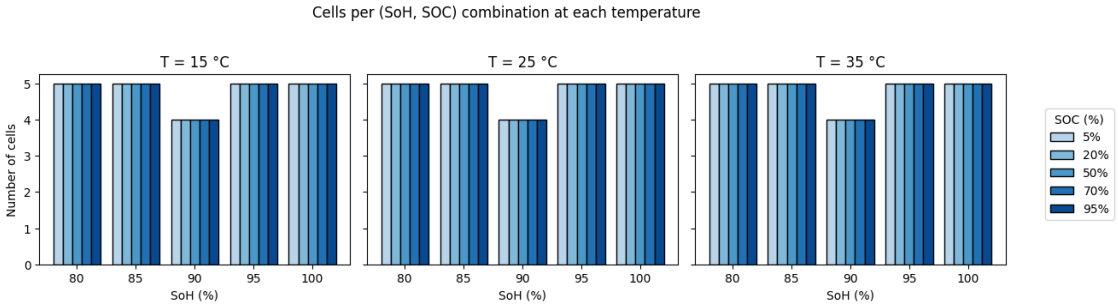
- use `soh_pct` (or `soh_frac = soh_pct / 100`) as the main regression target,
- treat SOC (`soc_pct`) and temperature (`temp_c`) as context features alongside the impedance features.

In a separate follow-up notebook **03 – Healthy vs Aged Classification from EIS** I will reuse this same dataset and features to derive a simple **healthy vs aged** label from SoH (e.g. using a 90% threshold) and focus on the binary classification view of the problem.

In this section I keep the exploration minimal and focus on two checks:

1. A histogram of SoH values across all EIS tests
   → shows how many spectra are available at each SoH level.
2. A 1×3 bar-plot figure ("Cells per (SoH, SOC) combination at each temperature")
   → summarises how the EIS tests cover the (SoH, SOC, T) grid and how many cells contribute to each combination.





4. Impedance feature engineering

Each EIS test corresponds to one spectrum:

- 61 frequency points (`frequency_hz`),
- real and imaginary parts of impedance (`z_real_ohm`, `z_imag_ohm`).

For the models I compress each spectrum into a small set of summary features. Per (cell, SoH, SOC, temperature) test I compute:

- **R_hf_ohm** – real part of Z at the highest frequency (approx. ohmic resistance),

- **R_lf_ohm** – real part of Z at the lowest frequency (approx. total resistance),
- **delta_R_ohm** – `R_lf_ohm - R_hf_ohm`,
- **Zmag_mean / Zmag_std / Zmag_min / Zmag_max** – summary statistics of |Z|,
- **phase_mean / phase_std** – summary statistics of the impedance phase.

On top of these global summaries, I sample the spectrum at a fixed set of frequencies (in Hz):

- `f_targets = [0.01, 0.1, 1.0, 10.0, 100.0, 1000.0, 10000.0]`

For each target frequency I find the closest available frequency in the measured spectrum and record:

- **Zmag_fX** – magnitude |Z| at that frequency,
- **phase_fX** – phase at that frequency.

This gives a compact "impedance fingerprint" per test that captures both global behaviour and the shape at a few physically meaningful frequencies.

The result is a modelling table `model_df` with one row per EIS test:

- metadata: `cell_id, soh_pct, soc_pct, temp_c`,
- impedance features: `R_hf_ohm, R_lf_ohm, delta_R_ohm, Zmag_* , phase_*`,
- SoH label for regression: `soh_frac = soh_pct / 100`.

Code 4.1 – define grouping and feature function

Code 4.2 – build feature table from all spectra

```
   cell_id  soh_pct  temp_c  soc_pct  R_hf_ohm  R_lf_ohm  delta_R_ohm  \
0        2       95      15        5   0.02995   0.07366      0.04371
1        2       95      15       20   0.03001   0.07916      0.04915
2        2       95      15       50   0.02965   0.03573      0.00608
3        2       95      15       70   0.02960   0.03803      0.00843
4        2       95      15       95   0.02958   0.04171      0.01213

   Zmag_mean  Zmag_std  Zmag_min  …  Zmag_f1p0  phase_f1p0  Zmag_f10p0  \
0   0.040826  0.016416  0.024201  …   0.042847   -0.282403    0.031430
1   0.041919  0.018060  0.024285  …   0.042731   -0.290515    0.031688
2   0.029907  0.003850  0.023819  …   0.030956   -0.031686    0.029673
3   0.030187  0.004274  0.023747  …   0.030942   -0.037886    0.029639
4   0.031920  0.005565  0.023771  …   0.034760   -0.067860    0.030048

   phase_f10p0  Zmag_f100p0  phase_f100p0  Zmag_f1000p0  phase_f1000p0  \
0    -0.123542     0.027523     -0.077613      0.024205       0.097449
1    -0.122719     0.027668     -0.079707      0.024291       0.095818
2    -0.047501     0.026807     -0.059273      0.023819       0.103972
3    -0.051610     0.026685     -0.058380      0.023747       0.103440
4    -0.092483     0.026772     -0.062605      0.023772       0.101259

   Zmag_f10000p0  phase_f10000p0
0       0.043538        0.812199
```

```
1       0.043579        0.811200
2       0.043398        0.818643
3       0.043320        0.818540
4       0.043262        0.817930

[5 rows x 27 columns]
```

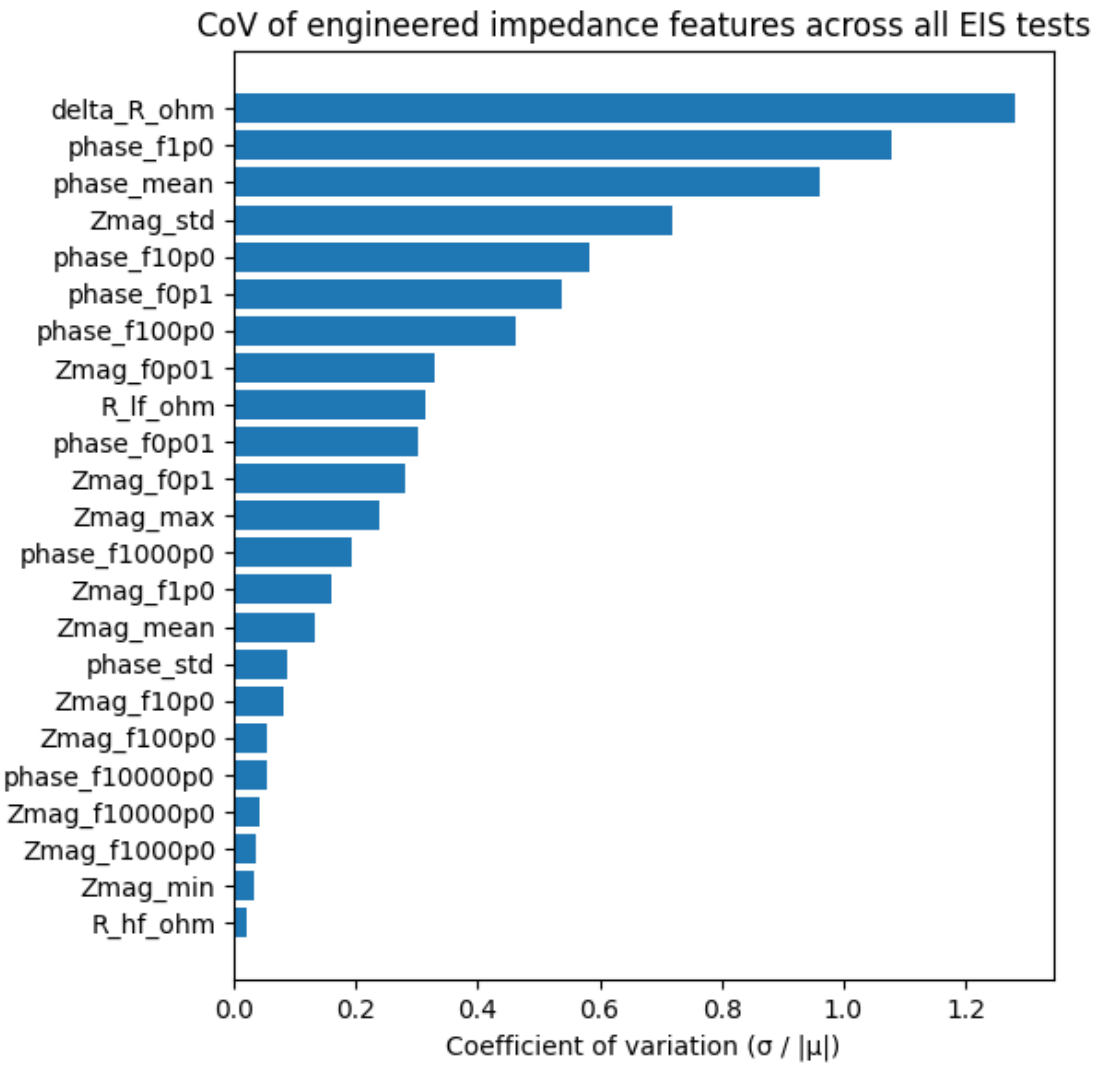
Code 4.3 – construct model_df with SoH label

```
   cell_id  soh_pct  temp_c  soc_pct  R_hf_ohm  R_lf_ohm  delta_R_ohm  \
0        2       95      15        5   0.02995   0.07366      0.04371
1        2       95      15       20   0.03001   0.07916      0.04915
2        2       95      15       50   0.02965   0.03573      0.00608
3        2       95      15       70   0.02960   0.03803      0.00843
4        2       95      15       95   0.02958   0.04171      0.01213

   Zmag_mean  Zmag_std  Zmag_min  …  phase_f1p0  Zmag_f10p0  phase_f10p0  \
0   0.040826  0.016416  0.024201  …   -0.282403    0.031430    -0.123542
1   0.041919  0.018060  0.024285  …   -0.290515    0.031688    -0.122719
2   0.029907  0.003850  0.023819  …   -0.031686    0.029673    -0.047501
3   0.030187  0.004274  0.023747  …   -0.037886    0.029639    -0.051610
4   0.031920  0.005565  0.023771  …   -0.067860    0.030048    -0.092483

   Zmag_f100p0  phase_f100p0  Zmag_f1000p0  phase_f1000p0  Zmag_f10000p0  \
0     0.027523     -0.077613      0.024205       0.097449       0.043538
1     0.027668     -0.079707      0.024291       0.095818       0.043579
2     0.026807     -0.059273      0.023819       0.103972       0.043398
3     0.026685     -0.058380      0.023747       0.103440       0.043320
4     0.026772     -0.062605      0.023772       0.101259       0.043262

   phase_f10000p0  soh_frac
0        0.812199      0.95
1        0.811200      0.95
2        0.818643      0.95
3        0.818540      0.95
4        0.817930      0.95

[5 rows x 28 columns]
```

4.4 Coefficient of variation of impedance features As a quick sanity check I look at the **coefficient of variation (CoV)** of the engineered impedance features across all EIS tests:

$$\text{CoV} = \text{std} \;/\; |\text{mean}|$$

Features with higher CoV vary more across the dataset and typically carry more useful information for the models than almost-constant features.



5. SoH regression models

In this section I predict SoH from the engineered impedance features.

- **Input features**: impedance features (`R_*`, `Zmag_*`, `phase_*`) plus SOC (`soc_pct`) and temperature (`temp_c`).
- **Target**: `soh_frac = soh_pct / 100`.

To avoid leakage between tests from the same physical cell, I use **nested grouped cross-validation**:

- **Outer loop**: 5-fold `GroupKFold` over `cell_id` (5 "M-fold" splits)
  → used to estimate model performance and variability across cell splits.
- **Inner loop**: 3-fold `GroupKFold` over `cell_id` inside each outer train split
  → used to select hyperparameters.

I compare three regression models:

1. **ElasticNet** (linear baseline with L1+L2 regularisation),
2. **XGBoostRegressor** (gradient boosting for tabular data),
3. **GaussianProcessRegressor** (non-parametric Bayesian regression with an RBF kernel).

Later I will take the **best-performing hyperparameters** for each model, refit on all data, and inspect the **parity plot** of GP and the **feature importance** for XGBoost.

5.1 Prepare features, target, groups

```
(24, (360, 24), (360,))
```

5.2 Metrics and hyperparameter grids

5.3 Running the models loop

```
=== Model: ElasticNet ===

ElasticNet outer CV:   0%|          | 0/5 [00:00<?, ?it/s]

  Fold 1: best inner MAE=0.0289 with params={'alpha': 0.001, 'l1_ratio': 0.5}
  Fold 2: best inner MAE=0.0545 with params={'alpha': 0.1, 'l1_ratio': 0.1}
  Fold 3: best inner MAE=0.0317 with params={'alpha': 0.001, 'l1_ratio': 0.1}
  Fold 4: best inner MAE=0.0449 with params={'alpha': 0.001, 'l1_ratio': 0.1}
  Fold 5: best inner MAE=0.0447 with params={'alpha': 0.001, 'l1_ratio': 0.1}

=== Model: XGBoost ===

XGBoost outer CV:   0%|          | 0/5 [00:00<?, ?it/s]

  Fold 1: best inner MAE=0.0434 with params={'n_estimators': 500, 'max_depth':
3, 'learning_rate': 0.1}
  Fold 2: best inner MAE=0.0472 with params={'n_estimators': 500, 'max_depth':
3, 'learning_rate': 0.05}
  Fold 3: best inner MAE=0.0367 with params={'n_estimators': 500, 'max_depth':
3, 'learning_rate': 0.05}
  Fold 4: best inner MAE=0.0403 with params={'n_estimators': 500, 'max_depth':
3, 'learning_rate': 0.1}
  Fold 5: best inner MAE=0.0408 with params={'n_estimators': 200, 'max_depth':
3, 'learning_rate': 0.1}

=== Model: GP ===

GP outer CV:   0%|          | 0/5 [00:00<?, ?it/s]

  Fold 1: best inner MAE=0.0326 with params={'length_scale': 10.0,
'noise_level': 0.01}
  Fold 2: best inner MAE=0.0430 with params={'length_scale': 10.0,
'noise_level': 0.01}
  Fold 3: best inner MAE=0.0286 with params={'length_scale': 10.0,
```

```
'noise_level': 0.01}
  Fold 4: best inner MAE=0.0377 with params={'length_scale': 10.0,
'noise_level': 0.01}
  Fold 5: best inner MAE=0.0402 with params={'length_scale': 10.0,
'noise_level': 0.01}
```

	model	outer_fold	best_params	\
0	GP	5	{'length_scale': 10.0, 'noise_level': 0.01}	
1	ElasticNet	5	{'alpha': 0.001, 'l1_ratio': 0.1}	
2	ElasticNet	2	{'alpha': 0.1, 'l1_ratio': 0.1}	
3	GP	2	{'length_scale': 10.0, 'noise_level': 0.01}	
4	XGBoost	5	{'n_estimators': 200, 'max_depth': 3, 'learnin…	
5	GP	3	{'length_scale': 10.0, 'noise_level': 0.01}	
6	XGBoost	1	{'n_estimators': 500, 'max_depth': 3, 'learnin…	
7	XGBoost	2	{'n_estimators': 500, 'max_depth': 3, 'learnin…	
8	GP	1	{'length_scale': 10.0, 'noise_level': 0.01}	
9	XGBoost	3	{'n_estimators': 500, 'max_depth': 3, 'learnin…	
10	ElasticNet	4	{'alpha': 0.001, 'l1_ratio': 0.1}	
11	ElasticNet	3	{'alpha': 0.001, 'l1_ratio': 0.1}	
12	GP	4	{'length_scale': 10.0, 'noise_level': 0.01}	
13	ElasticNet	1	{'alpha': 0.001, 'l1_ratio': 0.5}	
14	XGBoost	4	{'n_estimators': 500, 'max_depth': 3, 'learnin…	

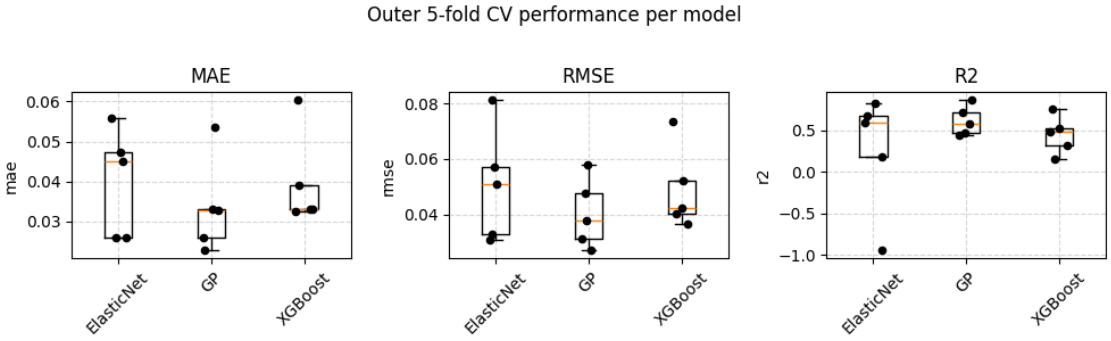
	mae	rmse	r2
0	0.022636	0.026865	0.868030
1	0.025800	0.030619	0.828568
2	0.025835	0.032980	0.680102
3	0.025895	0.031164	0.714351
4	0.032450	0.036305	0.758985
5	0.032707	0.047413	0.437996
6	0.032905	0.040188	0.524984
7	0.032959	0.042172	0.476920
8	0.033092	0.037768	0.580462
9	0.038925	0.051958	0.325101
10	0.045135	0.050831	0.596289
11	0.047411	0.057278	0.179817
12	0.053672	0.058010	0.474187
13	0.055824	0.081372	-0.947480
14	0.060430	0.073333	0.159722

6. Results & model comparison

In this section I compare the three models using the outer 5-fold grouped cross-validation results:

- **MAE** – mean absolute error on SoH (fraction),
- **RMSE** – root mean squared error on SoH,
- **R^2** – coefficient of determination.

Each box shows the distribution of the metric across the 5 outer folds for a given model.

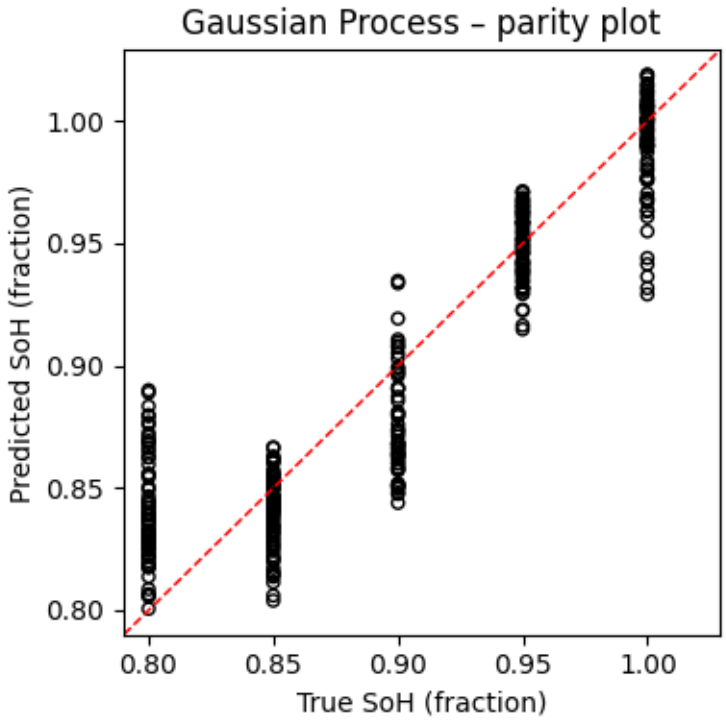


Overall, all three models achieve SoH errors on the order of **3–4% absolute SoH**. Across the 5 outer folds, the **Gaussian Process** model has the lowest MAE/RMSE and the highest, most stable R^2, followed by **XGBoost**. The **ElasticNet** baseline is clearly weaker and less stable (one split even has negative R^2), which suggests that a purely linear mapping from these impedance features to SoH is too restrictive for some cell splits.

7. Best model view: GP parity plot & XGBoost feature importance

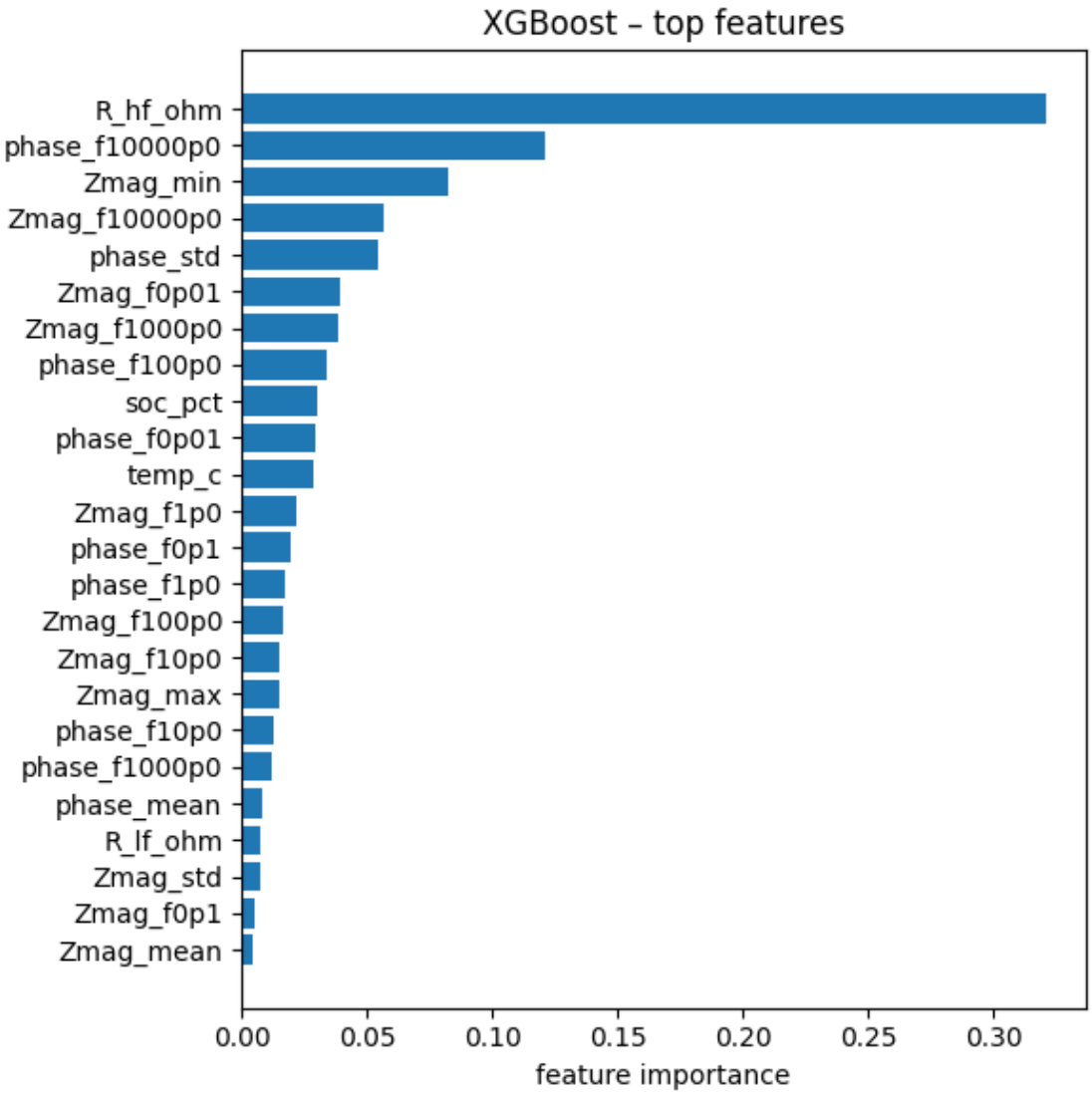
The Gaussian Process (GP) model achieved the best average performance across the outer folds. I first visualise its predictions versus the true SoH, then inspect feature importance for the XGBoost model as a complementary non-linear baseline.

7.1 GP parity plot (true vs predicted SoH)



The GP model tracks the discrete SoH levels well: most points lie close to the 1:1 line at each of the five SoH bands, with only modest vertical spread. Errors are slightly larger at the lowest SoH level, but overall the model stays close to the ideal diagonal across the whole 80–100% range.

7.2 XGBoost feature importance



XGBoost relies most strongly on the **high-frequency resistance** `R_hf_ohm` and the **phase and magnitude near 10 kHz**, with additional contribution from the minimum impedance (`Zmag_min`), low-frequency |Z|/phase, and the context variables `soc_pct` and `temp_c`. This suggests that both the ohmic region and the overall spread of the impedance spectrum carry useful information about SoH.