

A World
Leading SFI
Research
Centre



MLGIG Team – Milk Lactose Prediction Data Challenge



Insight



SFI RESEARCH CENTRE FOR DATA ANALYTICS

[georgiana.ifrim thach.lenguyen timi.aderinola davide.serramazza] @insight-centre.org

4th International Workshop on Spectroscopy and Chemometrics 2024

30/04/2024

HOST INSTITUTIONS



PARTNER INSTITUTIONS



FUNDED BY:



About MLGIG Team

- Many thanks to the organisers, we had lots of fun and learning!
- **Georgiana** (Assoc Prof@UCD-CS), **Thach** (postdoc Insight@UCD-CS),
Timi (postdoc Insight@UCD-CS), **Davide** (PhD ML-Labs@UCD-CS)
- Our background is in ML/AI modeling, time series, XAI for time series;
no domain expertise in spectroscopy
- **Our code, results and slides are available** (Python Jupyter notebooks):
<https://github.com/mlgig/VM-challenge-lactose-prediction-2024>

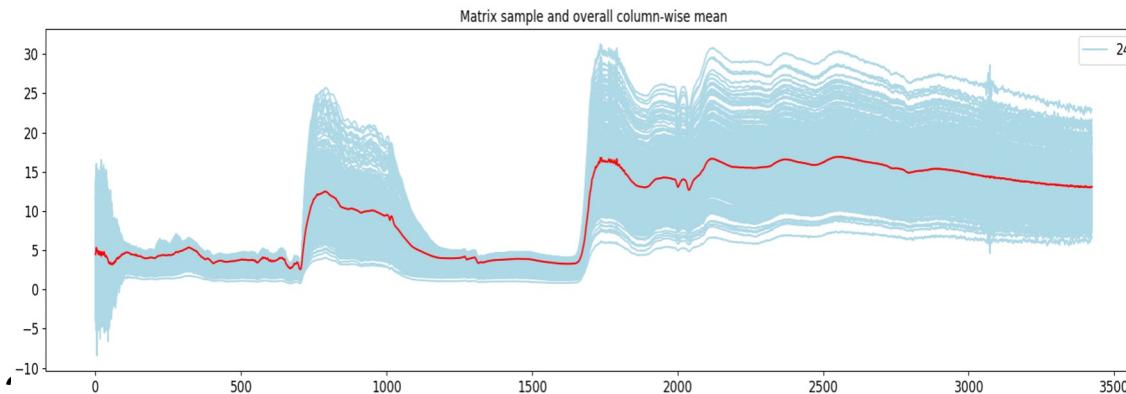
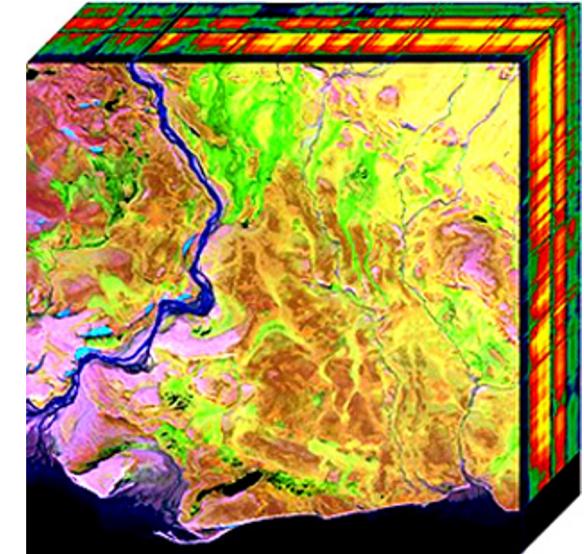
Outline

- Data Understanding and Preparation
- Modeling Approaches
- Results
 - Part 1: Best Predictive Model
 - Part 2: XAI to Improve the Best Predictive Model (Thach)

Data Understanding and Preparation

- Milk sample used to extract a **3d image (hyperspectral cube)**. Each pixel represented by a spectrum. **Our sample = 2d image, pixels on rows and spectrum on columns.**
- 64 samples to train (with targets)
- 8 samples to test (no targets)
- Target: lactose concentration measured in mg/mL
- **We represent each sample as a matrix with 300 rows and 3424 columns*.**

3D image
↓
2D image

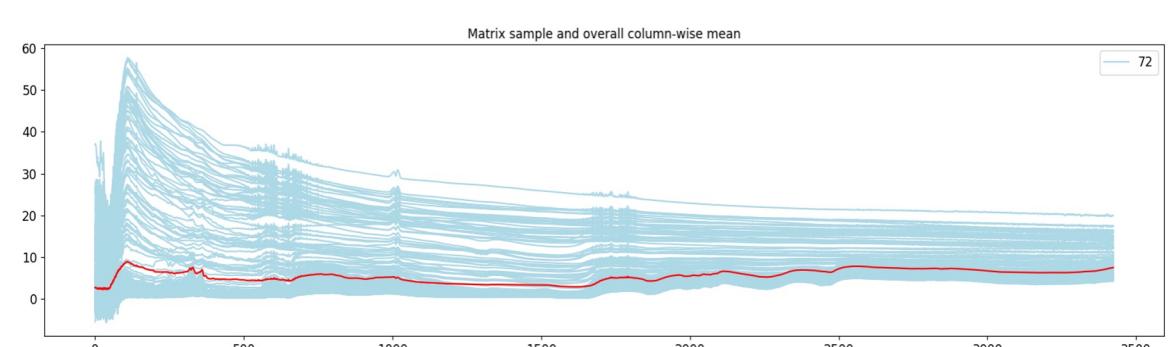
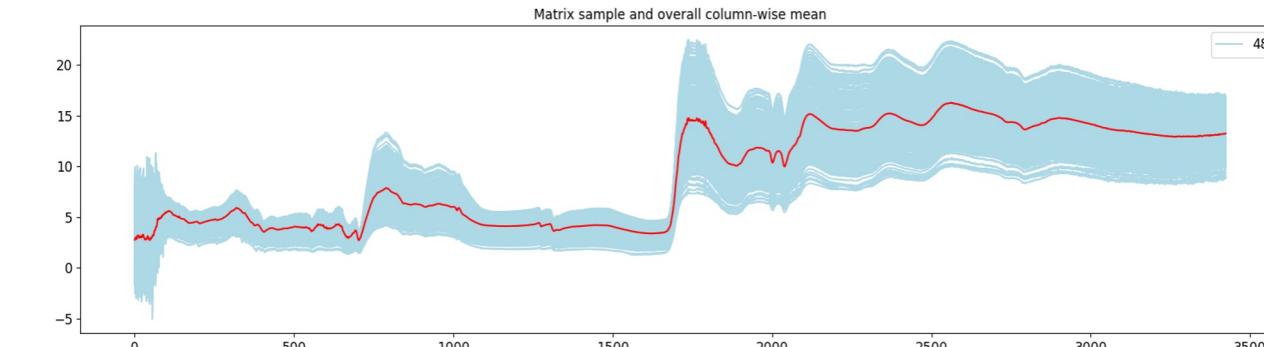
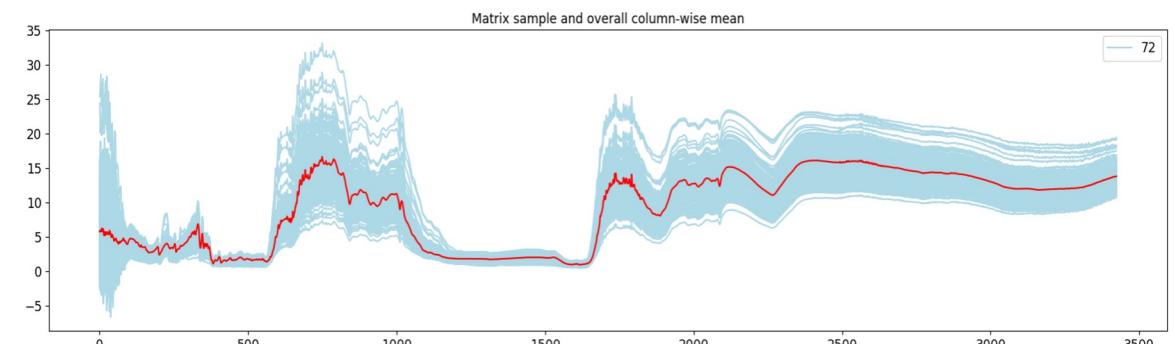
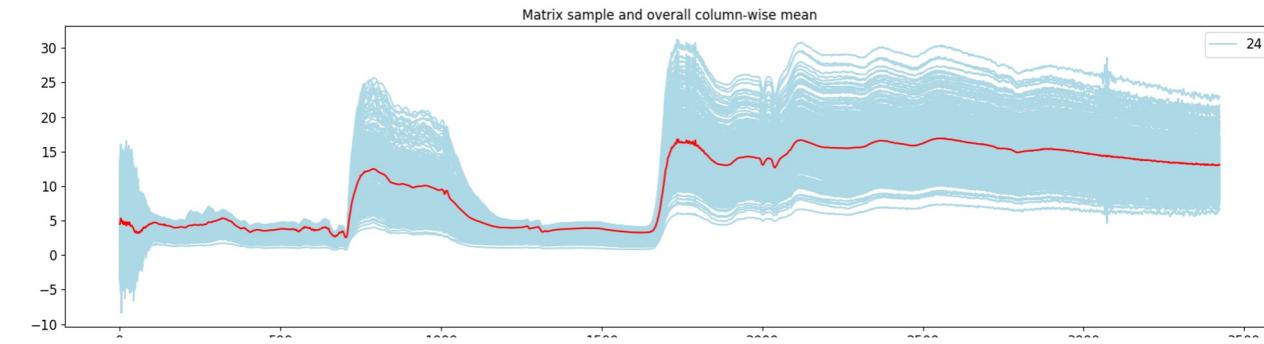
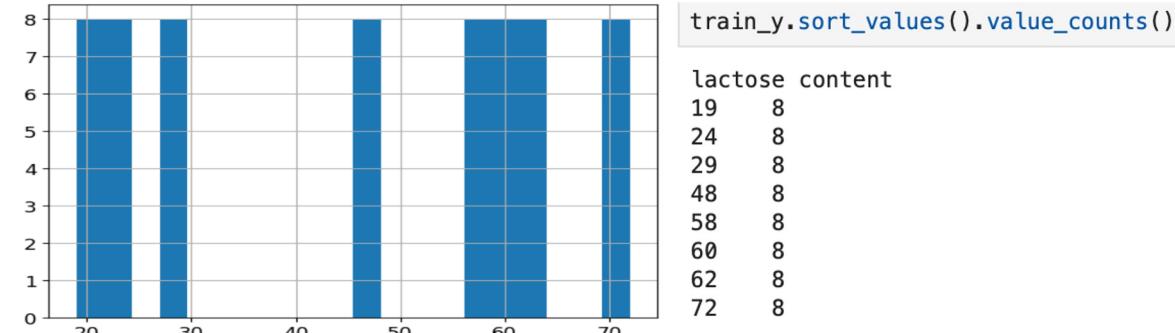


*Not all samples have the same number of rows, eg sample14 has 238 rows. We padded the rows with zeros up to 300 rows.

Data Understanding and Preparation

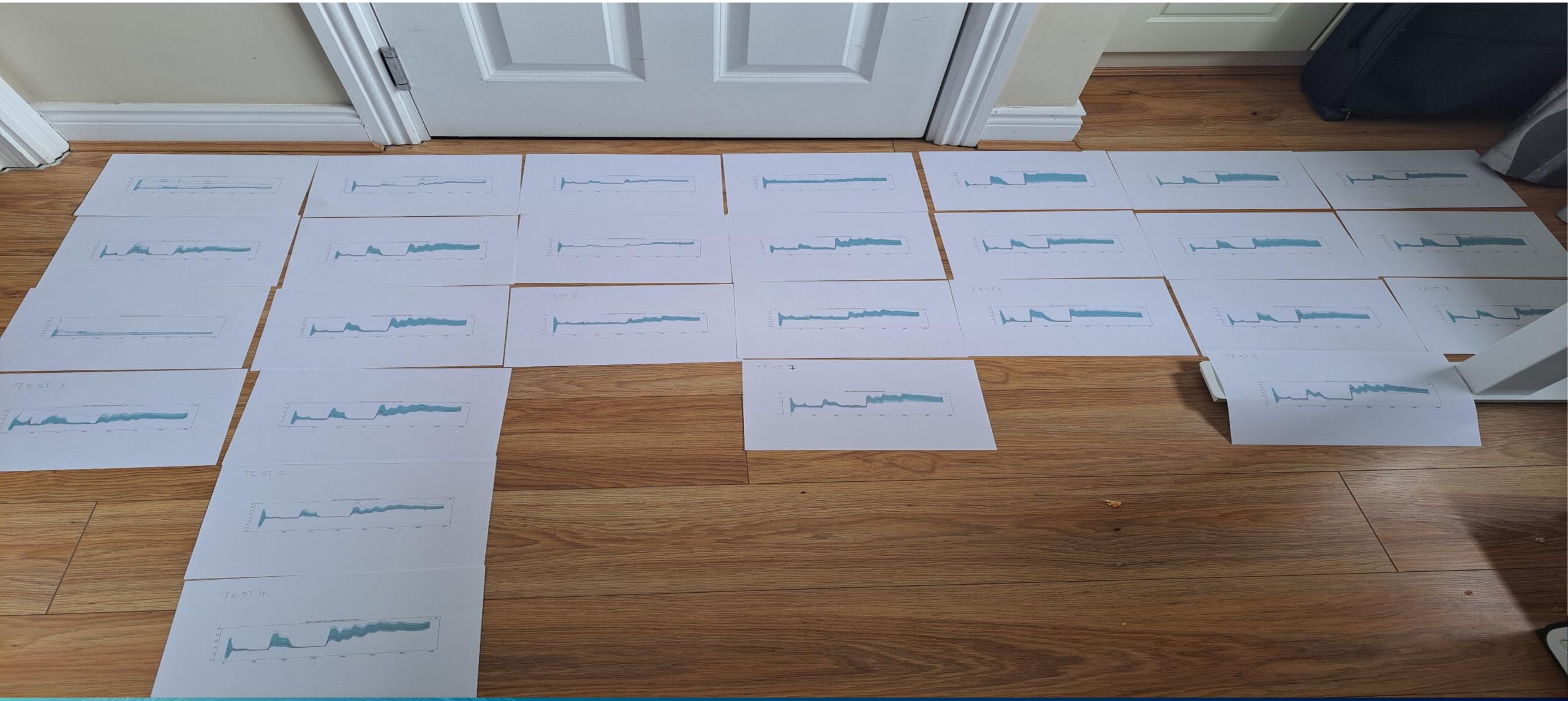
3 groups of train targets: 8 labels x 8 samples

- Low Lactose (19, 24, 29): 24 samples
- Medium Lactose (48): 8 samples
- High Lactose (58, 60, 62, 72): 32 samples



Data Understanding and Preparation

64 samples from 3 groups of lactose concentration (left most in the image is 72, 60/62, 58, 48, 29, 24, 19)



Data Understanding and Preparation

Challenges:

- 1. Storage:** Training data is about 700Mb. Can we reduce the data and still get good results?
- 2. Noise:** Feature selection: can we remove features that are noisy? Eg variance reduction, spectrum/column selection or pixel selection?
- 3. Outliers:** How to deal with outliers?
- 4. Small training set:** Number of samples is very low, only **64 samples to train**. This suggests preferring simpler models to complex DL models.

Modeling Approaches

What methods can we use to work with this data?

1. **Tabular methods:** if we concatenate the rows, each sample is a vector of features (temporal ordering discarded). Each vector will have 300 rows x 3424 columns = **1,027,200 features**. Can use all classic ML methods. Python library numpy, pandas, **sklearn**.
2. **Time series methods:** Each sample is a **time series (ordered features)** with multiple channels/rows. Python library: **aeon**, **captum**.
 - a. Multivariate time series regression (and channel selection methods).
 - b. Univariate time series regression (flatten the data to a single row).
3. **Deep Learning methods:** Models that work well with images, CNN, FCN, Resnet. Python library: **aeon**, **pytorch**.

Evaluation

- Regression task (numeric target): **RMSE and R2 as evaluation measures** for predictive quality. We keep track of **train+test runtime** to understand accuracy/efficiency tradeoffs.
- Single train/test 75/25 split to check what the model learns (shuffle the data before split), and where possible what are the important features.
- **4-fold Cross Validation (4CV) to compare different algorithms and estimate generalisation ability; model selection based on avgCV-RMSE**
- Select best model using CV -> train best model on the full training set -> predict on test.
- Extra sanity check: Look at predicted scores on test data and consistency across different models. Look at each test sample (plot image) versus predicted target.

Results - Summary

Summary of key findings about modeling, then detailed results.

- Tabular (with row concatenation) is similar to DL methods
 - < Multivariate TSR
 - < **Univariate TSR (with data flattened to a percentile)**
- DL methods we tested don't work well (small data, noise, outliers)
- Data denoising is important for this problem
- **Time series methods work well on this problem**
 - Methods that work in the time domain (interval-based, eg ROCKET) are better than those working in the frequency domain

Results - Summary

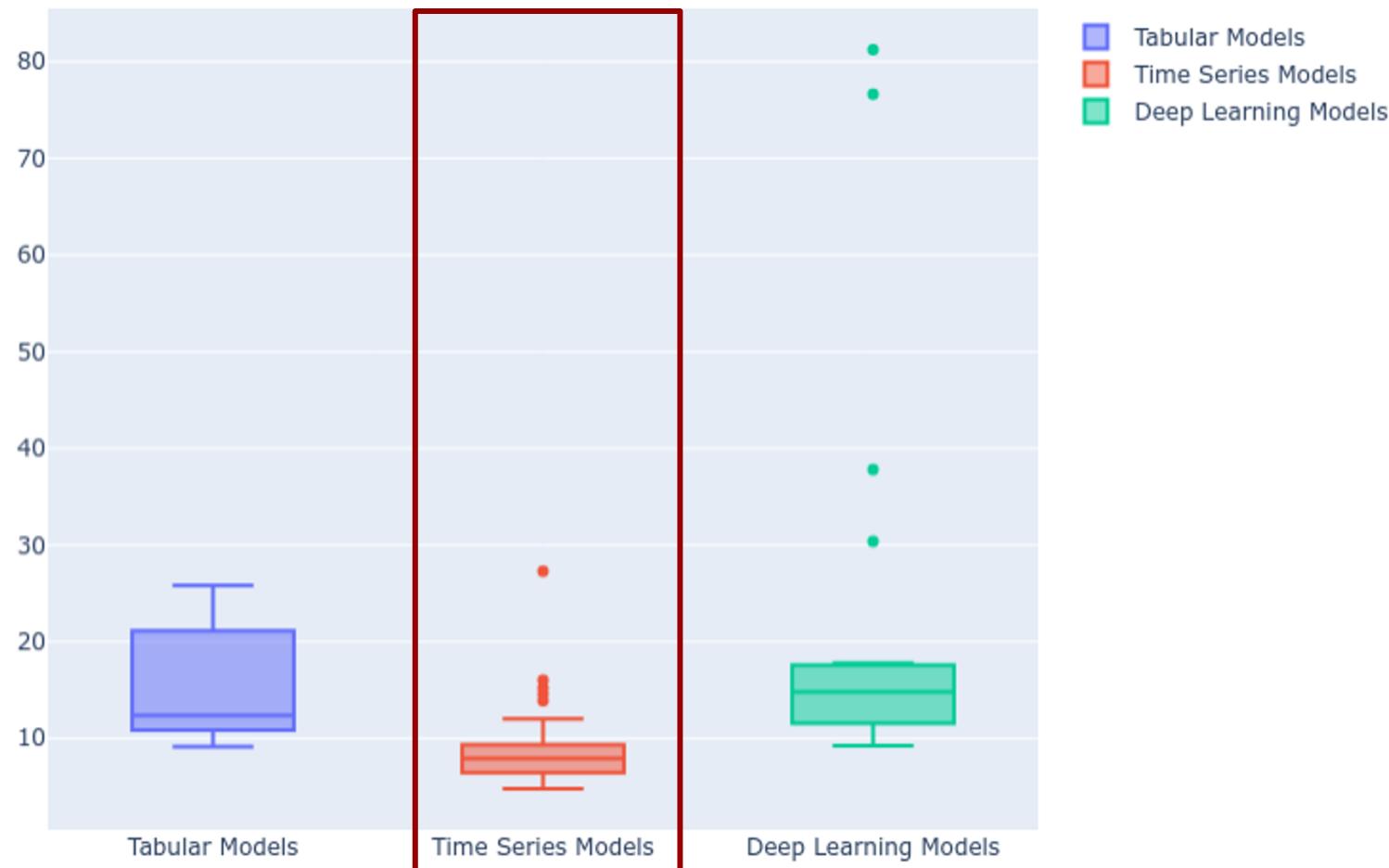
Boxplot of **4CV-RMSE** for models in each group

Tabular Models: Linear Regression, RidgeCV, PLSR

Time Series: 1NN-DTW, Catch22, DrCIF, Interval Forest, **ROCKET**, MiniROCKET, MultiROCKET, Random Interval, RISE, TSForest, TSFresh

Deep Learning: CNN, FCN, Inception Time, ResNet, TapNet

More detailed results [here](#)



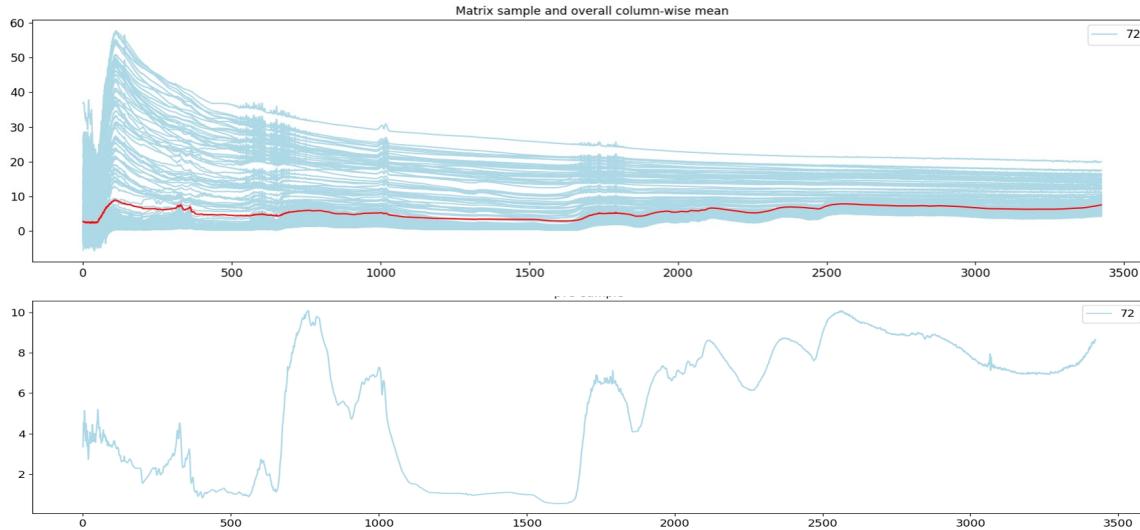
Results - Detailed

Baseline RMSE: 18.62 (for each sample in the training set, predict the average target computed on the training set)

| Method: Tabular Concatenate all rows/channels | 4CV avgRMSE | Predictions on test |
|---|------------------------|--|
| RidgeCV() | 19.19 | [56.34, 68.44, 42.64, 71.62, 6.39, 49.31, 36.04, 40.94] |
| KNeighborsRegressor() | 13.29 | [59.6, 52., 39.4, 60.4, 24., 57.6, 49., 49.6] |
| PLSRegression(n_components=5) | 17.32 | [71.23, 59.32, 37., 16, 61.66, 13.47, 50.01, 42.71, 46.31] |
| Best model: Noise cleaning + KNN <code>Pipeline(StandardScaler(), SelectFromModel(estimator=RidgeCV()), KNeighborsRegressor())])</code> | 12.08 | [61.6, 52., 29.8, 60.4, 26., 52.4, 36., 53.2] |

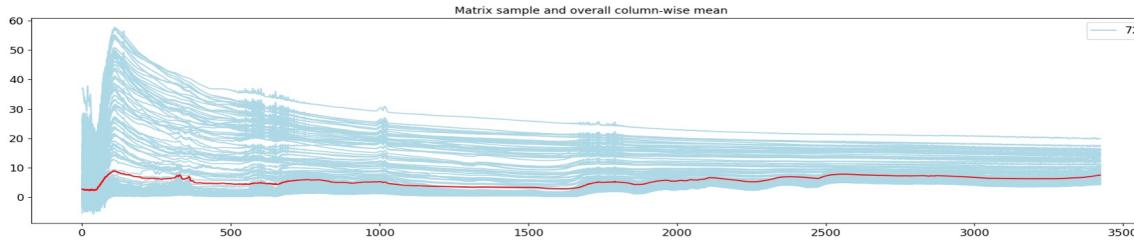
Results - Detailed

Data flattened to p75 =>



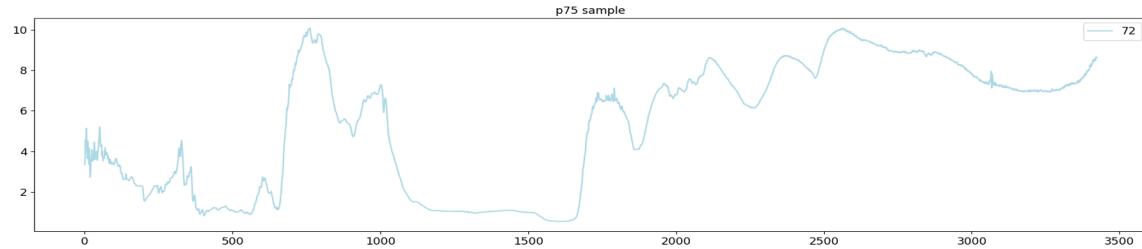
| Method: Tabular Flatten to percentile (p25, p50, p75) | 4CV avgRMSE | Predictions on test |
|--|------------------------|--|
| p25 RidgeCV() | 14.32 | [64.67, 62.55, 45.26, 66.77, 39.28, 49.76, 38.81, 34.52] |
| p50 PLSRegression(n_components=10) | 9.83 | [70.64, 64.9, 43.97, 67.14, 35.91, 42.91, 40.32, 38.49] |
| Best model: p75 RidgeCV() | 9.60 | [71.28, 66.38, 55.84, 63.8, 37.71, 52.86, 48.99, 35.64] |

Results



| Method: Multivariate Time series | 4CV avgRMSE | Predictions on test |
|--|----------------|---|
| All Channels <pre>Pipeline(Rocket(random_state=42)), StandardScaler(), SelectFromModel(estimator=RidgeCV()), RidgeCV())]</pre> | 7.13 | [73., 65.12, 41.17, 64.59, 25.79, 46., 46.25, 31.16] |
| <pre>Pipeline(Rocket(random_state=42)), StandardScaler(), SelectFromModel(estimator=RidgeCV()), ExtraTreesRegressor())]</pre> | 7.00 | [71.73, 61.05, 48.51, 61.37, 26.13, 54, 84, 50.46 23.57] |
| Subset of channels [p25, p50, p75] Best model: ROCKET + noise cleaning + ExtraTreesRegressor <pre>Pipeline(Rocket(random_state=42)), StandardScaler(), SelectFromModel(estimator=RidgeCV()), ExtraTreesRegressor())]</pre> | 5.76 | [70.98 59.4 54.11 61.16 21.1 53.88 52.06 23.64] |

Results - Detailed



| Method: Univariate Time Series Flatten to percentile (best results with p75) | 4CV avgRMSE | Predictions on test |
|--|----------------|---|
| Pipeline(Rocket(random_state=42)), StandardScaler(), RidgeCV())) | 5.13 | [73.4 62.48 51.78 64.54 24.5 47.43 47.18 25.59] |
| Pipeline(Rocket(random_state=42)), StandardScaler(), SelectFromModel(estimator=RidgeCV())), RidgeCV()) | 4.99 | [74.22 62.62 51.48 64.58 25.38 47.39 46.56 25.52] |
| Best model: ROCKET + Noise cleaning + ExtraTreesRegressor Pipeline(Rocket(random_state=42)), StandardScaler(), SelectFromModel(estimator=RidgeCV())), ExtraTreesRegressor()) | 4.71 | [72, 59.4, 51.56, 60.18, 22.51, 49.96, 48.72, 22.9] |

Results - Best Model

Best model: Samples flattened to p75

```
make_pipeline(Rocket(normalise=True, num_kernels=10000, random_state=142),
             StandardScaler(),
             SelectFromModel(RidgeCV()),
             ExtraTreesRegressor(n_estimators=100, random_state=142))
```

CV RMSE: 4.44, CV R2: 0.94

Predictions on test set: [71.86, 59.64, 53.14, 60.24, 22.78, 51.52, 49.24, 23.8]

Time to train + test (sec): 20.40 seconds

Results - Best Model

What seems to work well for this problem:

- 1. Outlier removal:** Flatten the time series to p75.
- 2. TS transformation:** Rocket time series transform to extract features from p75 time series
(convolution may have smoothing effects)
- 1. Noise reduction:** Scaling and feature selection
- 4. Learning algorithm:** A non-linear regressor ExtraTreesRegressor(), this is an ensemble of randomised decision trees inspired by Random Forests.
 - 1. Flatten to percentile**
 - 2. Rocket()**
 - 1. StandardScaler()**
SelectFromModel(RidgeCV())
 - 1. ExtraTreesRegressor()**

Results - Our Submissions

Our predictions (in the order presented in the test set):

1. Human guess: sol1
2. Best model: sol2
3. Best model + XAI + human: sol3
(presented in PART2)

Our best model avgCV: 4.44
Test error without outlier: 4.33

| Test-Id | TrueLabel | sol1-Human | sol2-ML | sol3-ML+XAI+human |
|-------------|-----------|-------------|------------------|-------------------|
| Sample39 | 72 | 72 | 71.86 | 72 |
| Sample29 | 62 | 60 | 59.64 | 60 |
| Sample50 | 60 | 24 | 53.14 | 48 |
| Sample03 | 58 | 62 | 60.24 | 60 |
| Sample19 | 29 | 29 | 22.78 | 24 |
| Sample30 | 48 | 58 | 51.52 | 58 |
| Sample67 | 19 | 19 | 23.8 | 24 |
| Sample42 | 24 | 48 | 49.24 | 48 |
| TestRMSE | | 15.77973384 | 9.8033336 | 10.47616342 |
| RMSE w/o 42 | | 14.22271825 | 4.3389465 | 6.568322247 |

Thank you for listening! Questions?

**PART 2 - XAI to Improve the Best Predictive Model
(presented by Thach)**