**Agenda:**

* Re-do correlation imputation with columns being imputed only if they have pairwise correlation >= 0.9 (Pearson’s correlation coefficient)
* Re-do analysis after dropping columns
* MLP and RNN fine-tuning with Optuna
* 95% thresholded results with fine-tuned models (cannot handle missing data)
* Discussion topics

**Clarification of model training process:**

When I train the models on the non-thresholded data, I am training on the original, non-imputed input data, which is why I cannot train many of the machine learning models on this dataset.

**Adjustments in data pre-processing and imputation**:

Based on feedback from last week’s meeting, I re-did the correlation imputation using a threshold of 0.9. In other words, a column with less than half its values missing was used to impute the values of another column with at least 75% of its values missing if the pairwise correlation coefficient between the two columns was at least 0.9.

Additionally, through taking a more detailed look at my individual features, I realised that my input dataset contained the following variables:

* Lifetime risk of maternal death (1 in: rate varies by country)
* Lifetime risk of maternal death (%)
* Number of maternal deaths
* Maternal disorders prevalence (age standardized) (per 100 000 population
* Maternal and neonatal disorders prevalence (age standardized) (per 100 000 population)

In my opinion, these variables provided very similar information to the maternity mortality ratio that I was training my model to estimate. I was wary of these columns being using by the model to cheat as it could use these columns to directly infer the mortality estimate instead of learning patterns in the other health-related and socioeconomic information. Therefore, I removed these variables and re-ran all my analyses.

For data with no threshold applied, standardisation produced a MAPE multiple times higher than without standardisation.

A screenshot of a computer

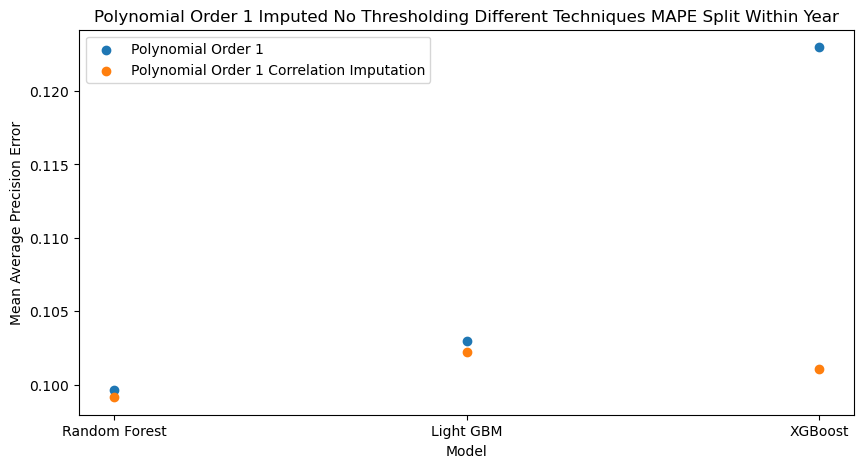
AI-generated content may be incorrect.

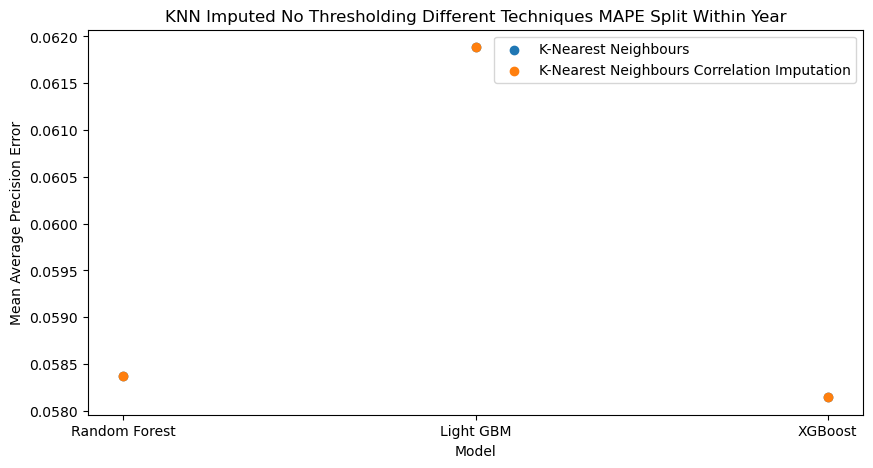
A screen shot of a computer

AI-generated content may be incorrect.

Polynomial orders 2 and 3 both had MAPEs above 10 as well.

Polynomial imputed, unthresholded data had MAPES generally between 0.09 and 0.15 regardless of whether correlation imputation was applied. After the changes made, using correlation imputation now decreases the MAPE on non-thresholded data for all polynomial imputation methods (no difference observed for KNN imputation). However, this difference appeared to always be less than 0.05.





The results from the data pre-processed with a 95% threshold will be presented in the following section.

**Neural Networks Fine-Tuning:**

At this point in my project, I have experimented with two types of neural network:

* Multi-Layer Perceptron (MLP)
* Recurrent Neural Network (RNN)

I implemented these using PyTorch functionalities, as they provide a lot of flexibility regarding how the network is constructed and trained. The MLP worked without further processing of the imputed datasets, but the recurrent neural network needed to see the dataset in a time series format. More specifically, it needed to track an identified subgroup over a specific sequence of time. I defined a unique identity that could be tracked over time as a specific combination of country name, dimension, and subgroup. This corresponded to a specific demographic (e.g. females, or lowest wealth quintile) from a specific country being tracked over time. I defined a sequence as consisting of 3 rows, or this entity being tracked over 3 consecutive years. I chose 3 years because it was approximately 1/9 of my dataset, allowing for many sequence permutations to be fed into the model.

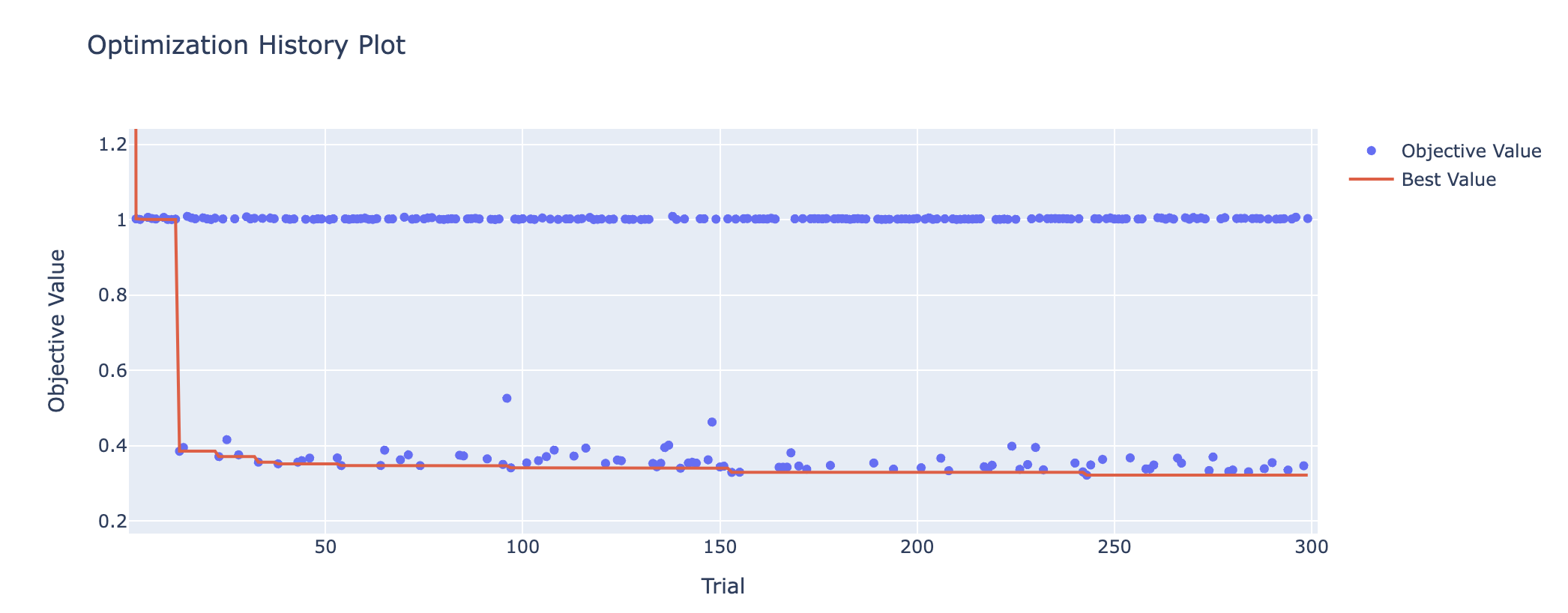
To get the best performance from a neural network, it is common practice to fine-tune their hyperparameters. I fine-tuned each of the networks on each of the imputation datasets for either un-processed data, scaled data, and correlation imputed data. I chose to fine-tune the following MLP hyperparameters:

* Number of hidden layers (1 to 4)
* Number of hidden units per layer (4 to 128)
  + There could be different numbers of hidden units per layer in the same network
* Learning rate (0.01 to 0.0001)
* Weight decay (0.01 to 0.0001)
  + L2 regularisation term
* Scheduler type (none, step learning rate, reduce learning rate on plateau)
  + Step: tested step sizes between 5 and 10 and factor decreases between 0.1 and 0.5
* Activation function (ReLU or tanh)

I tested the same hyperparameters with the RNN, with the caveat that the number of hidden units per layer was the same for the same model (due to the parameters accepted by this implementation).

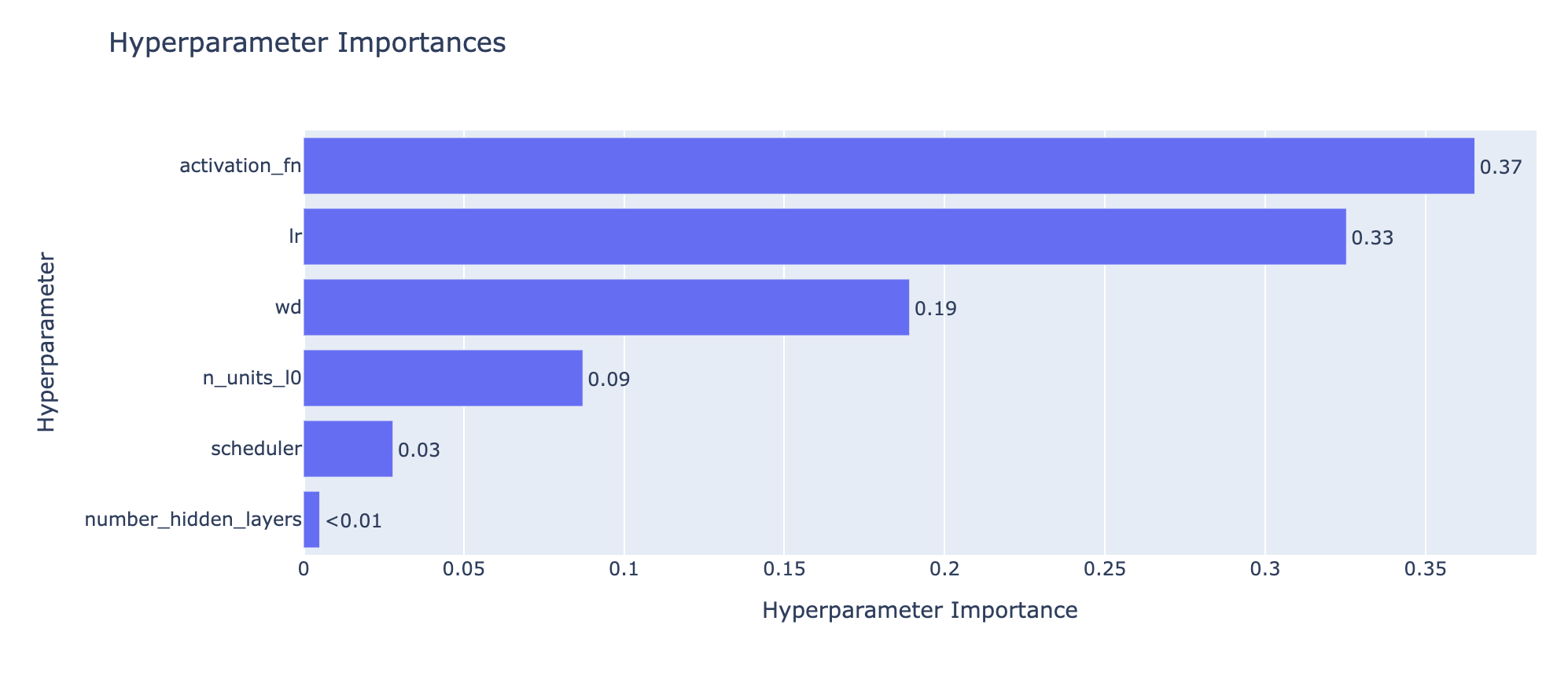
Both networks used the Adam optimiser, which could make parameter specific alterations to learning rate.

I ran 300 trials per model per dataset. Over the trials, I saw extremely high MAPE values, but often there were at least one combination of hyperparameters that produced a MAPE of less than 1. Often, as shown in the visualisation below, the fine tuners found a low MAPE against which to compare future trials relatively early in the process (before 50 trials had elapsed). This same trend was observed when fine-tuning the RNNs. The following plot shows the fine-tuning process for an MLP trained on kNN imputed data.

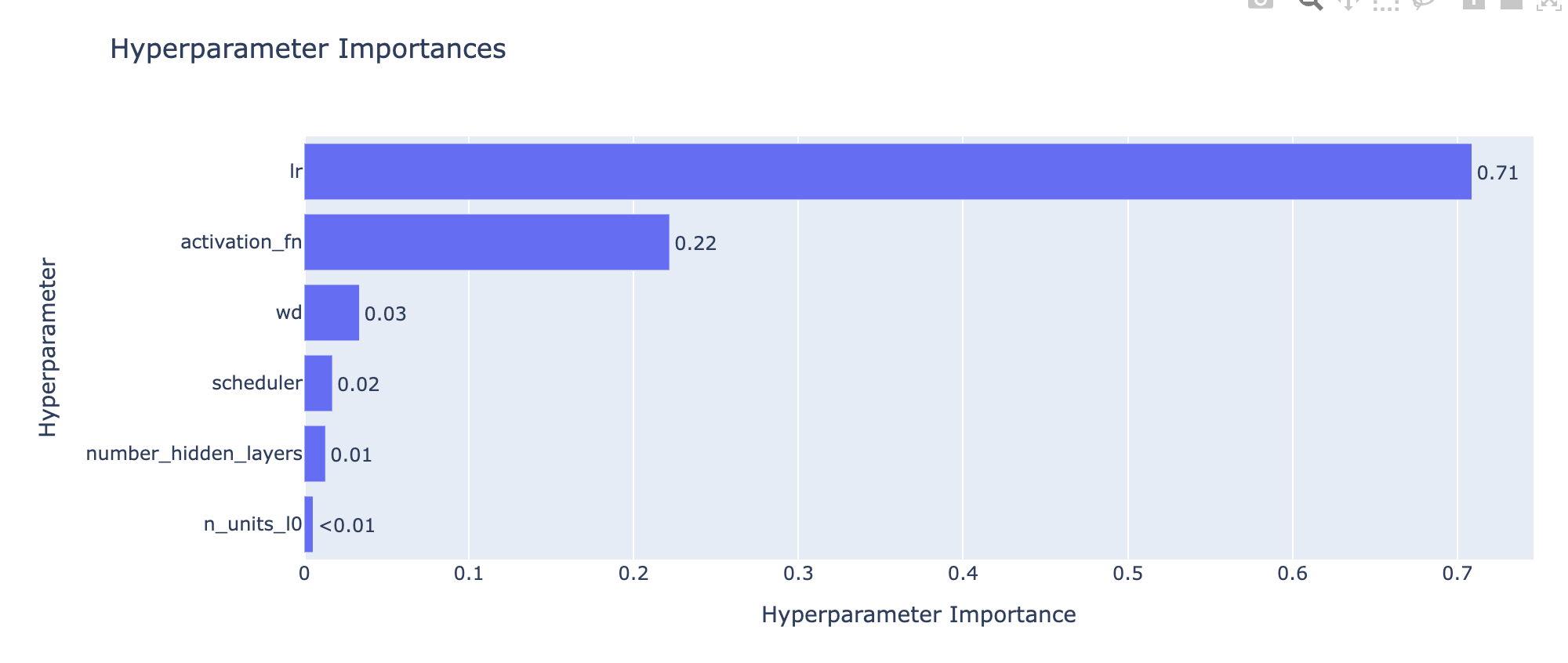


Optuna also plots the relative importance of different hyperparameters on reducing error in the model. Generally, the activation function or learning rate had the greatest importance, with weight decay and number of hidden units in the first hidden layer also sometimes having high importance, depending on the model. Each imputation method appeared to have slightly different distributions of importance, and there did not appear to be coherent trends for the MLP vs RNN. The number of hidden layers and scheduler type tended to have less of an effect, except for in an example shown below.

MLP, Polynomial Order 3:



MLP Polynomial Order 2



RNN, Miss Forest

A graph with blue bars

AI-generated content may be incorrect.

All the best non-scaled, no correlation imputation tuned RNNs had 4 hidden layers. There was more variation for the best MLPs, which had hidden layers ranging from 1 to 4. I can show more information if there are any questions. Both models tended to use the ReLU activation function. The number of hidden units in the RNNs ranged from in the 20s to in the 70s. The number of hidden units per layer in the MLPs did not appear to have a consistent trend (examples below).

90, 70, 50, 50

51, 7, 83

123, 32, 69, 94

53

113, 121, 100

**Results**:

The models trained on data with a 95% threshold was presented below.

A screen shot of a graph

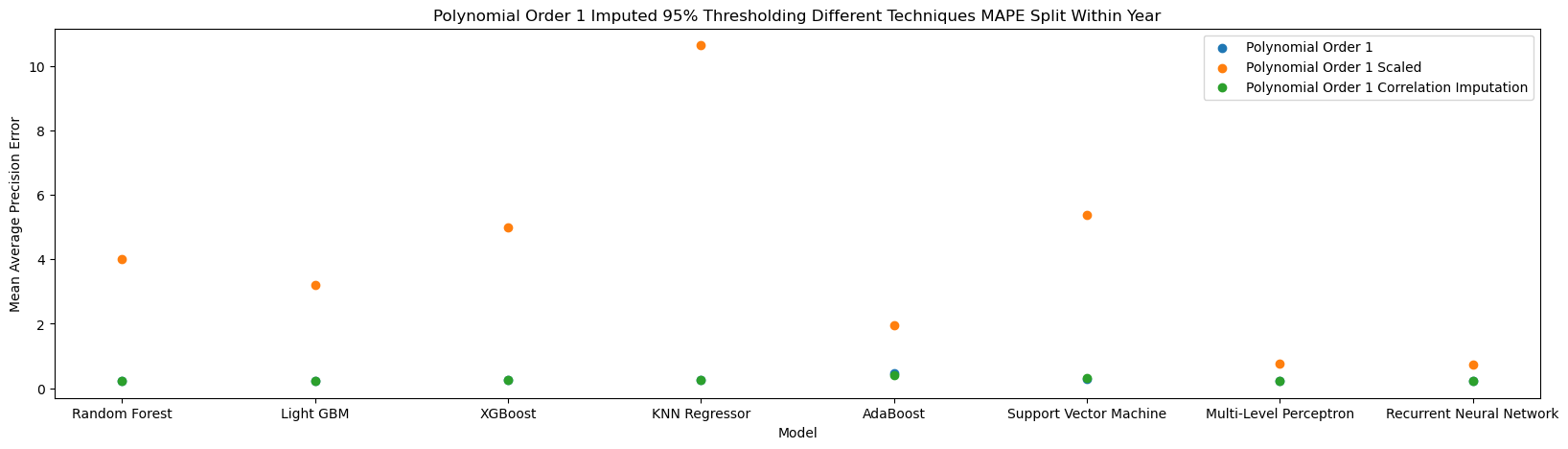
AI-generated content may be incorrect.

A screen shot of a graph

AI-generated content may be incorrect.

* No clear winner with either the KNN or Miss Forest imputed data
  + But ensemble-based methods and neural networks performed the best.
* Generally, the neural network
* Scikit learn’s random forest always performed among the best, with other models do better depending on imputation method of KNN or Miss Forest.

Scaling performed the worst for all 3 polynomial orders, with an example below.



I removed the scaled data to show the difference between the other two series more clearly.

A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

**A white screen with many dots

AI-generated content may be incorrect.**

* While there was no clear method winner for polynomial 1, correlation imputation tended to do better for polynomial 2 but did more poorly for polynomial 3.
* The neural networks were always among the best, with Scikit Learn’s Random Forest generally doing the best of the ensemble-based methods. AdaBoost performed poorly for these polynomial imputed datasets.

\* I need to finish running the fine-tuning process on the test/train split by year data.

**Discussion points**:

*Discussion of research aims*

Once I have my fine-tuned model, what do you see me using it for? In other words, what should I expect the results for this project to be? Ideas:

* Using SHAP to better understand most meaningful features to better avoid data biases in machine learning for public health
  + Does this differ for the different models chosen?
* Seeing how a change in the value of one of the features (e.g. increasing skilled birth attendance by 5%) would impact MMR to evaluate different policy approaches.

*How to incorporate uncertainty into my models.*

*How to choose an imputation method if the results are not directly comparable*

*Abstract*