**Rolling window experiments**

New Code files:

*run\_arima\_ets\_rolling\_origin.py* -> runs the base arima and ets rolling window forecasts

*run\_ml\_method\_rolling\_origin.py* -> runs the ML reconciliation for the rolling windows

*run\_hts\_benchmarks\_rolling\_origin.py* -> runs the reconciliation benchmarks for the rolling window

*calculate\_rolling\_origin\_errors.py* -> this code will read all the base and reconciled forecasts and output a csv file with the percentage relative improvement of each reconciliation method [the order of the columns shows order of the method performance]

The forecasts are only for one-step ahead

* Each rolling window has a set of fitted values, actual values corresponding to the fitted data, forecasts 1-step and the actual values for the 1-step ahead forecasts
* For all reconciliation approaches the reconciliation approach is trained/ parameters are found after every 10th rolling window (we discussed this approach to reduce the time complexity for re-estimating parameters for each rolling-window)
  + For example, if we have 20 rolling windows for 1-10 rolling windows the parameters are found for the reconciliation method using the 1st windows fitted values. And the forecasts are reconciled using these parameters for all 1-10 rolling windows. Then the parameters are re-estimated from the 11th rolling window and these parameters used to reconcile the forecasts across 11-20th rolling windows.

For ML methods:

Case 1 -> Only the bottom level error as the validation loss

Case 2 -> Whole hierarchy is considered in the validation loss

**Prison**

**ARIMA**

****

**ETS**

****

**Labour**

**ARIMA**

****

**ETS**

****

**Tourism**

**ARIMA**

****

**ETS**

****

**Wikipedia**

**ARIMA**

****

**ETS**

****

**Running Time for ML method**

This table shows the average running time per rolling window when training and executing a trained ML reconciliation method. The average training time also includes the hyper-parameter tuning time.

|  |  |  |  |
| --- | --- | --- | --- |
| **Dataset** | **Base Model** | **Average training time** | **Average execution time once trained** |
| Prison | ARIMA | 25.47 minutes | 0.024 minutes |
|  | ETS | 22.22 minutes | 0.024 minutes |
| Labour | ARIMA | 15.65 minutes | 0.023 minutes |
|  | ETS | 15.09 minutes | 0.023 minutes |
| Tourism | ARIMA | 11.46 minutes | 0.028 minutes |
|  | ETS | 11.79 minutes | 0.023 minutes |
| Wikipedia | ARIMA | 77.79 minutes | 0.033 minutes |
|  | ETS | 72.34 minutes | 0.035 minutes |