Forecasting Banknote Flows in the BdI Branch Network: Speed Up with Machine Learning

Brandi M. [1] Fusaro M. [1] Laureti T. [2] Rocco G. [1]

[1] Currency Circulation Management Directorate

[2] Department of Economics, Engineering, Society and Business Organization - DEIM, Tuscia University

21 October 2019

Outline

Banknotes can be deposited in and withdrawn from all 35 BdI branches by Credit Institutions (CIs)

Banknotes can be deposited in and withdrawn from all 35 BdI branches by Credit Institutions (CIs)

Challenge

Plan the transfers of banknotes towards the Branches and the central hub in order to promptly satisfy the request.

Banknote Services: Overview

Banknotes can be deposited in and withdrawn from all 35 BdI branches by Credit Institutions (CIs)

Challenge

Plan the transfers of banknotes towards the Branches and the central hub in order to promptly satisfy the request.

Statistical challenge

Forecasting the deposited D and the withdrawn W banknotes.

Variables (W and D) are **stochastic** since they depend from the CIs behaviour that could reflect the market demand.

Aim of the study: Forecast W and D.

Variables (W and D) are **stochastic** since they depend from the CIs behaviour that could reflect the market demand.

Aim of the study: Forecast W and D.

Forecast process should be...

- ...able to treat numerous time series $\rightarrow$ **more than 400**
- ...update regularly (every week) $\rightarrow$ **fast**
- ...accurate $\rightarrow$ **low prediction error**
- ...automatized as much as possible $\rightarrow$ **no assumptions**

Looking at the Data

- **35** branches
- **7** denominations for D
- **6** denominations for W (€500 cannot be issued)

$\Bigg\}$ $\rightarrow$ **455** time series

Weekly time series from January 2009 to June 2019 (**546** data points for each time series).

A good property

Almost all the weekly time series show high seasonal patterns.

Outline

Classic approach vs ML

- Parametric econometrics assume that the data come from a generating process with the following form:

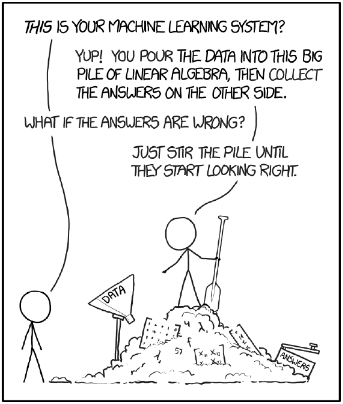
$$y = X\beta + \epsilon$$

- Machine Learning does not make any assumption on how the data have been generated:

$$y \approx f(X)$$

Why Machine Learning?

- Data intensive problem
- Data-driven approach
- Catch nonlinear relationship
- Oriented to forecast framework
- Easy implementation of ensemble methods



Issue: difficult interpretability of the model (in this case a not needed property).

Outline

Overview of the process

For each time series the following process is done

1. Split the data in train and test set ($h = 52$ weeks)

Overview of the process

For each time series the following process is done

1. Split the data in train and test set ($h = 52$ weeks)
2. Pre-process data (from $Y_t \rightarrow \tilde{Y}_t$) with STL decomposition

Overview of the process

For each time series the following process is done

1. Split the data in train and test set ($h = 52$ weeks)
2. Pre-process data (from $Y_t \to \tilde{Y}_t$) with STL decomposition
3. Forecast the future observations in two step:

Overview of the process

For each time series the following process is done

1. Split the data in train and test set ($h = 52$ weeks)
2. Pre-process data (from $Y_t \rightarrow \tilde{Y}_t$) with STL decomposition
3. Forecast the future observations in two step:
    1. Estimate the trend $\hat{T}_{t+h}$ in the test set with automatic choice of the ARIMA models

Overview of the process

For each time series the following process is done

1. Split the data in train and test set ($h = 52$ weeks)
2. Pre-process data (from $Y_t \to \tilde{Y}_t$) with STL decomposition
3. Forecast the future observations in two step:
    1. Estimate the trend $\hat{T}_{t+h}$ in the test set with automatic choice of the ARIMA models
    2. Run ML algorithms on $\tilde{Y}_t$ with hyperparameter auto-calibration and forecast $\hat{\tilde{Y}}_{t+h}$

Overview of the process

For each time series the following process is done

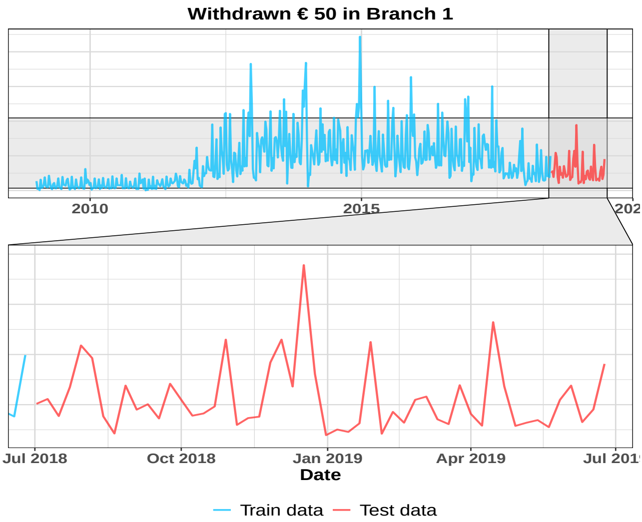
1. Split the data in train and test set ($h = 52$ weeks)
2. Pre-process data (from $Y_t \rightarrow \tilde{Y}_t$) with STL decomposition
3. Forecast the future observations in two step:
    1. Estimate the trend $\hat{T}_{t+h}$ in the test set with automatic choice of the ARIMA models
    2. Run ML algorithms on $\tilde{Y}_t$ with hyperparameter auto-calibration and forecast $\hat{\tilde{Y}}_{t+h}$
4. Final forecast the future values with $\hat{Y}_{t+h} = \hat{T}_{t+h} + \hat{\tilde{Y}}_{t+h}$

Overview of the process

For each time series the following process is done

1. Split the data in train and test set ($h = 52$ weeks)
2. Pre-process data (from $Y_t \rightarrow \tilde{Y}_t$) with STL decomposition
3. Forecast the future observations in two step:
    1. Estimate the trend $\hat{T}_{t+h}$ in the test set with automatic choice of the ARIMA models
    2. Run ML algorithms on $\tilde{Y}_t$ with hyperparameter auto-calibration and forecast $\hat{\tilde{Y}}_{t+h}$
4. Final forecast the future values with $\hat{Y}_{t+h} = \hat{T}_{t+h} + \hat{\tilde{Y}}_{t+h}$
5. Compare the forecast with the observed values

Example - Train and Test



Prepare data for Machine Learning

It is not usual to treat time series with ML algorithm due to the time dependencies (Bontempi et al, 2012, Dietterich, 2002).
In order to use them: pre-processing the data

Prepare data for Machine Learning

It is not usual to treat time series with ML algorithm due to the time dependencies (Bontempi et al, 2012, Dietterich, 2002).
In order to use them: pre-processing the data

Seasonal and Trend decomposition using Loess - STL (Cleveland, 1990)

$$Y_t = T_t + S_t + R_t$$

Prepare data for Machine Learning

It is not usual to treat time series with ML algorithm due to the time dependencies (Bontempi et al, 2012, Dietterich, 2002).
In order to use them: pre-processing the data

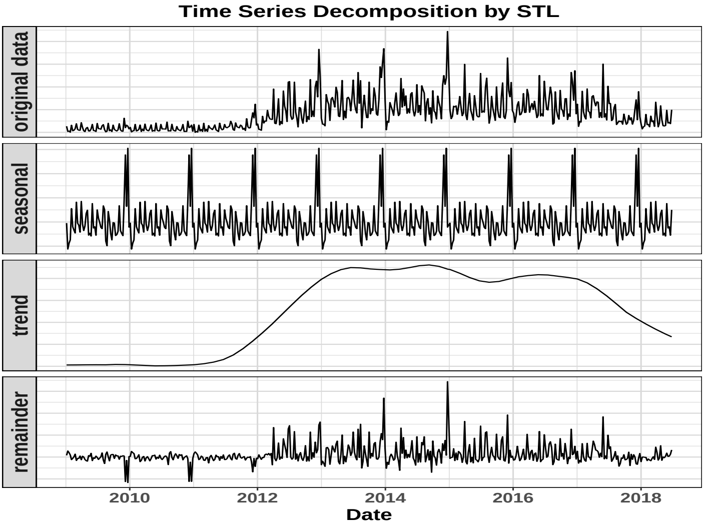
Seasonal and Trend decomposition using Loess - STL (Cleveland, 1990)

$$Y_t = T_t + S_t + R_t$$

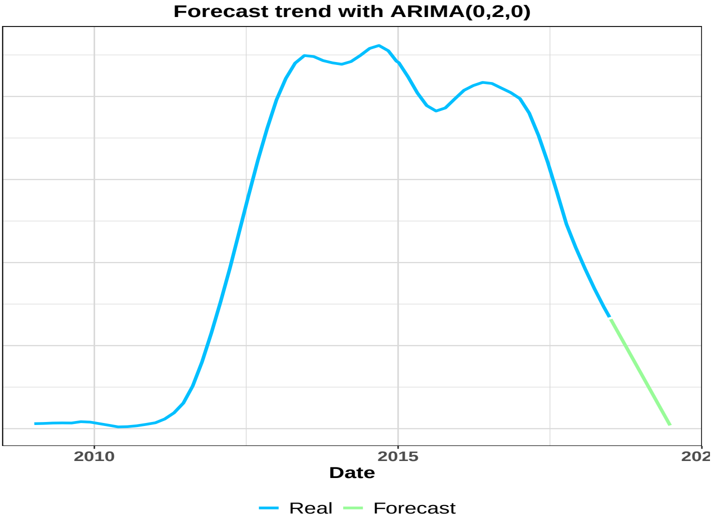
Apply ML methods (Laurinec, 2017) on

$$\tilde{Y}_t = S_t + R_t$$

STL decomposition: an example



Forecast Trend $\hat{T}_{t+h}$



Features

- Seasonal variable S_t
- Variable lag $\tilde{Y}_{t-1}, \tilde{Y}_{t-2}, \ldots$
- Fourier coefficients: $a_i \cos(\omega_i t)$, $b_i \sin(\omega_i t)$ for $i = 1, \ldots, K$ (Young, 1999)
- Dummy for Easter week (variable across years) E_t

Modeling $\tilde{Y}_t$ in ML algortihms

$$\tilde{Y}_t = f\left(S_t, \tilde{Y}_{t-1}, \tilde{Y}_{t-2}, \ldots, a_1, b_1, \ldots, E_t\right)$$

Algorithms

Using several ML algorithms such as:

- CART
- CTREE
- Bagging-CART
- Bagging-CTREE
- Random Forest
- Neural Networks

Compare forecasting performances with *classic* and probabilistic-based model such as:

- SARIMA
- Dynamic Harmonic Regression

Both are used with automatic choice of ARIMA models

Forecast performance evaluation

The evaluation of the forecast performance of the algorithms is measured with MASE (*Mean Absolute Scaled Error*) introduced in (Hyndman, 2006). Let $e_t = Y_t - \hat{Y}_t$ be the one-step-ahead forecast error. Then, a scaled error is defined as

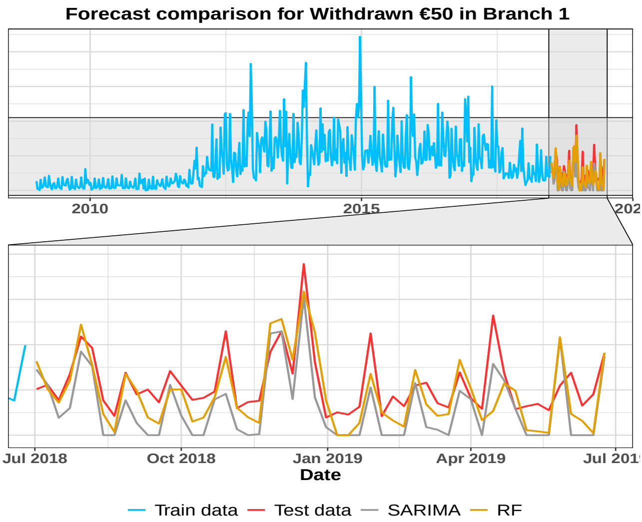
$$q_t = \frac{e_t}{\frac{1}{n-1}\sum_{i=2}^{n} |Y_i - Y_{i-1}|},$$

which is independent of the scale of the data. Mean absolute scaled error is defined as

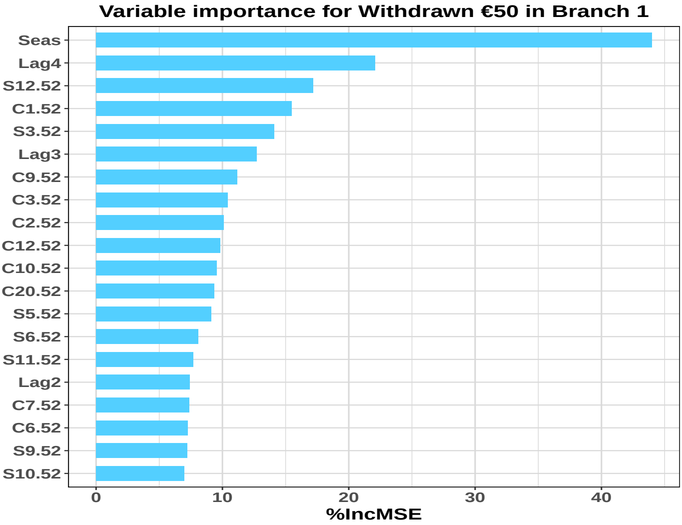
$$MASE = E(|q_t|)$$

Outline

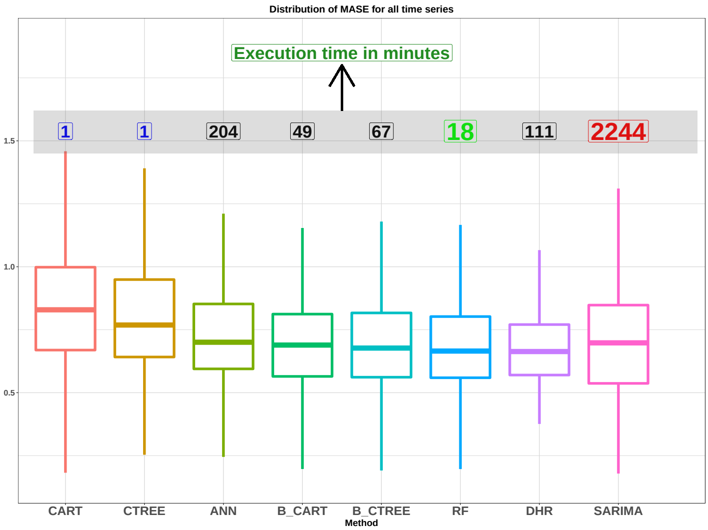
An example of forecasting



Output of Random Forest



Comparison of forecast perfomance



All the algorithms were executed on a PC with processor INTEL(R) Core(TM) i5-4300U 1.90GHz and 8,00 GB RAM.
R version 3.6.0

Conclusions

- The forecast performance of ML methods are in line with classic methods
- ML methods are time-saving
- No assumptions need to be verified

Next steps...

- ... a formal selection of variable lags
- ... moving from Easter dummy variable to a special dummy variable in order to include local holidays or events
- ... optimize the hyperparameter selection for ML algorithms (with rolling window subsample)
- ... multivariate modelization (VAR works very bad!)

THANK YOU

FOR

YOUR ATTENTION

How bagging with time dependencies?