


Comparative Study on Time Series Forecasting

Student:
Yacine Ben Baccar

School Supervisor:
François ROEUFF

Master's Coordinator:
Erwann LePennec
Florence d'Alché-Buc

Company Supervisor:
Bertrand LAMY
Jacques DOAN HUU

Department of Information Technologies
October 9, 2019

Abstract

Today, there are plenty of various forecasting models for Time Series with each one requiring proper data preprocessing and analysis to provide a usable prediction. The aim of this report is to conduct a comparative study on the most commonly used Time Series estimators in order to benchmark their performance on a wide variety of series from different fields (economics, finance, meteorology, etc...) and compare them to the in-house estimator developed by **SAP** called **SAPTF**. All the implemented models are automated, making hyper-parameter search a part of the model, this is done so that it can be used without any prior knowledge of the models, nor the datasets on which they will be applied on.

Acknowledgements

The successful completion of this work could not be possible without the help and support from a lot of individuals. I will take this opportunity to thank them for their help.

I wish to express my sincere gratitude and respect for both my supervisors **Bertrand Lamy** and **Jacques Doan Huu**. I am grateful for their valuable guidance, support and encouragement. I would also like to thank wholeheartedly all my colleagues, at SAP, that I had the pleasure to work and interact with and for making me feel welcome since the first day.

I owe a lot of what I have achieved to my parents and my brother for their constant love and support. They have always been there to guide and encourage me to a have positive outlook and a can-do attitude towards life's challenges.

At last, I want to thank the entire faculty and staff of the schools that I have attended in the past few years which have shaped me into the engineer that I am now.

Abbreviations

ARIMA Auto Regressive Integrated Moving Average

ARMA Auto Regressive Moving Average

AR Auto Regressive

DETS Double Exponential Smoothing

ETS Exponential Smoothing

FFT Fast Fourier Transform

GRU Gated Recurrent Unit

LSTM Long Short Term Memory

MAE Mean Average Percentage Error

MAPE Mean Absolute Percentage Error

MA Moving Average

MSE Mean Squared Error

NN Neural Network

P2 Pearson Squared Correlation

RMSE Root Mean Square Error

RNN Recurrent Neural Network

SARIMA Seasonal Auto Regressive Integrated Moving Average

SETS Simple Exponential Smoothing

SMAPE Symmetric Mean Average Percentage Error

SVR Single Vector Regression

TETS Triple Exponential Smoothing

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Time Series modeling is a dynamic research area which has attracted the attention of the research community in the past decades. The main objective of time series modeling is to collect, analyze and rigorously study the past and develop an appropriate model which describes the inherent structure of the series. This model is then used to explain the Time Series and predict future values for it, i.e **forecasting**.

Time Series forecasting is one of the most common and used learning tasks, every day businesses use time series forecasting for a wide variety of purposes such as forecasting daily stock prices, forecasting foreign currency exchange rates, forecasting rates of unemployment; meteorologists use it to provide an estimate of the wind speeds, daily maximum and minimum temperatures and approximate the precipitations. All these and many other tasks show the importance of time series and the importance of having a good estimation of the future as it might be paramount for businesses to prepare for potential spike/dip of their sales and prepare for it or to avoid catastrophe when observing meteorological data.

Time series forecasting is the act of predicting the future after careful consideration and analysis of the past, due to the indispensable importance of this task in numerous fields such as finance, meteorology, business, science and engineering. Preparing an adequate model to fit and then predict the series is not an obvious task as each signal/series has its own properties and dependencies on exogenous parameters that cannot be easily represented in the model. Over the years, an abundance of research and models have been proposed by academics, statisticians and economists to improve the forecasting accuracy. As a result, various Time series models have been put to work/improved, but this abundance of models does not necessarily mean that these models are ubiquitously applicable. The most popular and used approaches are still statistical models like **ARIMA** and **ETS** and machine learning regression models (appropriately applied to Time Series) like **SVR**. But, in the past few years, **RNN** have become more and more competitive to the traditional approaches and provide much more expandability on the previously mentioned models. They can be used with a wide variety of data (exogenous) that can enrich the insight provided by time series.

Chapter 2

Basic Concepts of Time Series

2.1 What is a Time Series ?

Every sequential set of datapoints constitutes a Time Series. Each Time Series has two main components: **time** and **value** assigned to the corresponding time step. One of the most common time series example is the daily closing price of a stock, in this graph below we represent the closing price of SAP share between the years 2007 and 2019.

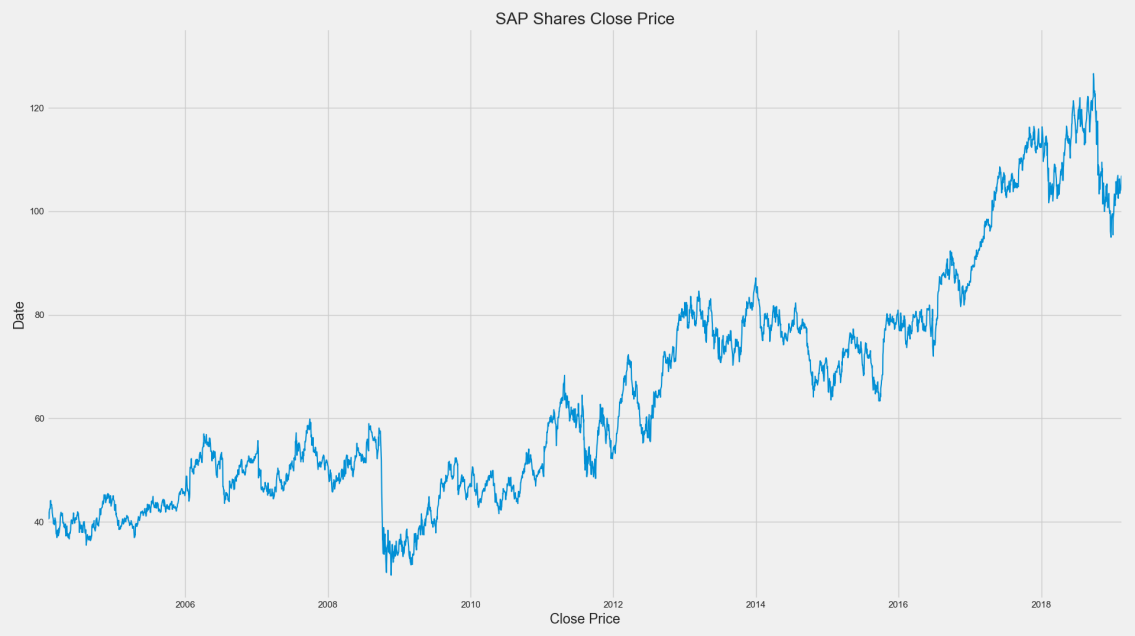


Figure 2.1: SAP shares close price between 2007 and 2019

A time series that records the measurements of a single phenomenon/variable is called univariate, whereas the measurements of multiple phenomena/variables is called multivariate. In this study, we will focus on forecasting models that deal with univariate time series. An other way to differentiate between time series is the way they are recorded, as we have discrete time series and continuous time series.

Discrete time series, are measurement taken (sampled) at regularly spaced time steps (monthly, quarterly, yearly), these time series are very common (the tracking of the unemployment rate which is usually measured every quarter). As for the continuous time series, they tend to be more dense (as they are continuously recorded) and vary a lot faster than discrete time series, an example of these time series are EEG (Electroencephalogram) or the measurments of sensors. In practice all time series are discrete but they differ by their granularity and dependency on past values.

2.2 Types of Time Series

There exists many classification criteria for time series, in this part we will focus on time dependency and stationarity.
Time dependency refers to the influence of past values on the newly observed values of the recorded variable/phenomenon, we can divide time series into two categories: long memory time series and short memory time series. Long memory time series are those that have an auto-correlation function that decreases slowly, these time series describe a process that has slowly changing behavior. Long memory time series can be found usually in meteorological, geological data; an example of long memory time series is the evolution of the mean temperature on planets. Short term time series describe a process with a fast turnover and have an auto-correlation function that decrease rapidly as the longer we are from the present the less useful to the future the measure is. A typical example of these time series are financial data, like stock prices.
For the second classification criteria: stationarity, we have stationary and non-stationary time series. Stationary time series, are processes that have statistical properties (mean, variance) that do not depend on time. Time series that do not fit the aforementioned description are called, non-stationary, these time series are very common in finance and the retail sector. These types of time series can be tough to predict without any type of pre-processing, that is why we use multiple techniques to stationarize these processes to be able to get better forecasts.

2.3 Why do we do Time Series Analysis ?

Time series analysis is done for mainly two reasons:

- Understanding the behavior of the process, by studying its past records, to be able to model and identify the main parameters that influence the time series and identify its components.

- Forecasting the future values of the series, using an adequate model that has been trained on past values.

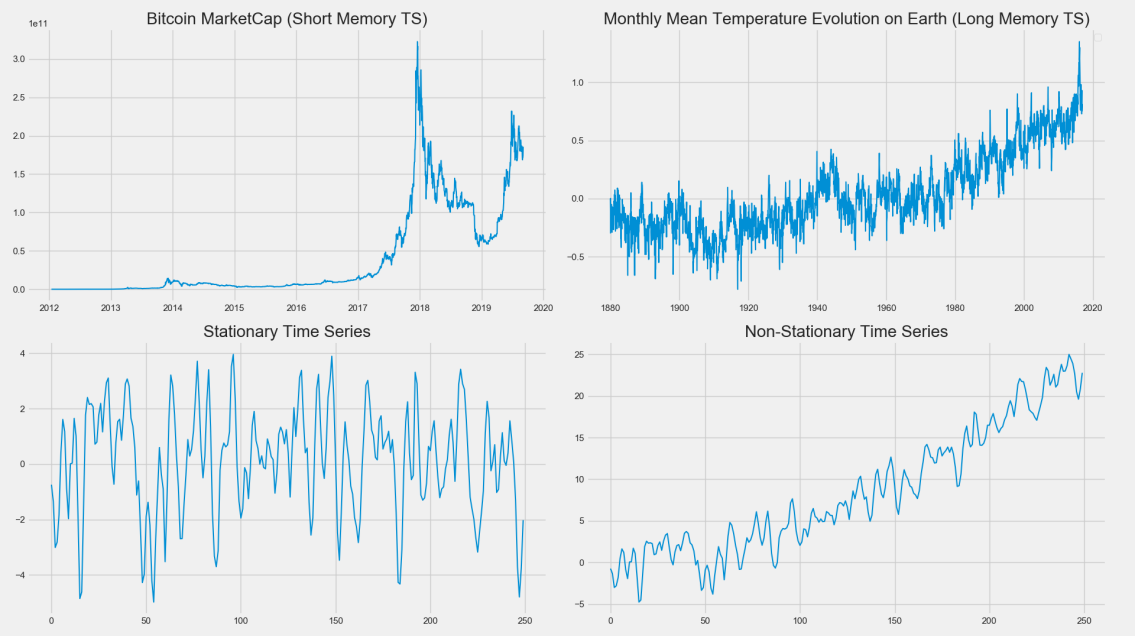


Figure 2.2: Example of Long/Short memory, Stationary/Non-Stationary Time Series

2.4 Time Series Components

$$y(t) = T(t) + S(t) + C(t) + \varepsilon(t) \tag{2.1}$$
$$y(t) = T(t) \times S(t) \times C(t) \times \varepsilon(t) \tag{2.2}$$
$$\tag{2.3}$$

$y(t)$ represents the the measure captured at time step t, $T(t)$ is the overall **trend** of the series, $S(t)$ describes the **seasonal** aspect of the time series, $C(t)$ is the **cyclic** component of the observation and ε_t represents the irregular patterns within the series, **residuals**.

Figure 2.2 represents the sales of a retail store between the years 2013 and 2015, in this time series we can observe the aforementioned components (Figure 2.4). The trend varies slightly in this case but the seasonal part can be inferred from the behavior of the series and some common sense. We can see a spike of sales at the end of each year which is something to be expected as these dates correspond to the holidays, where most retail stores make the most sales.

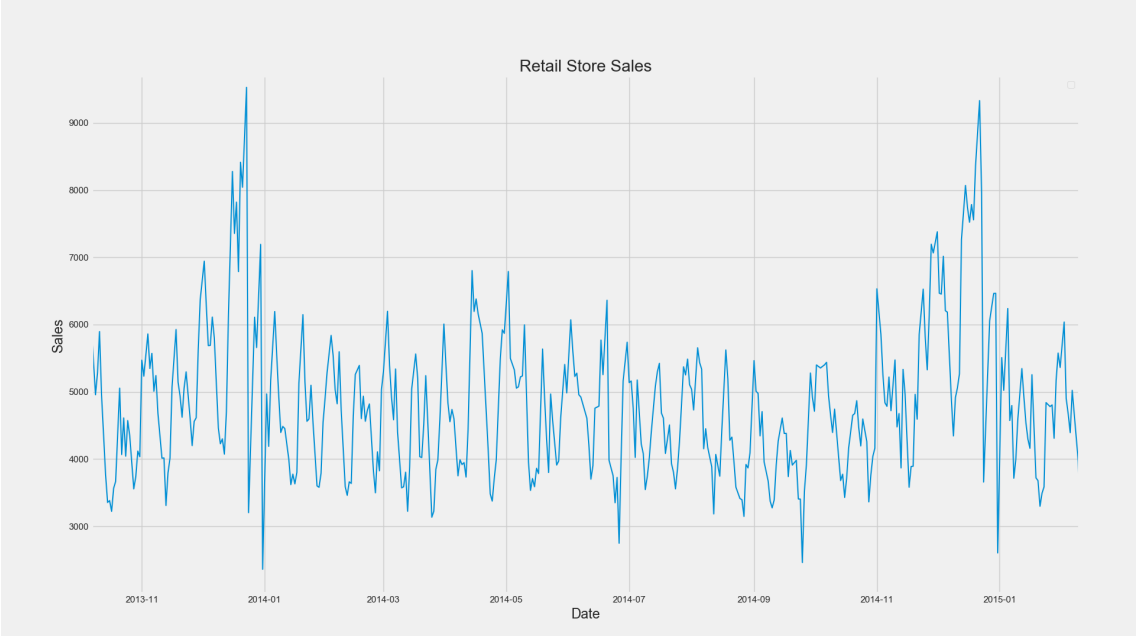


Figure 2.3: Sales of retail store

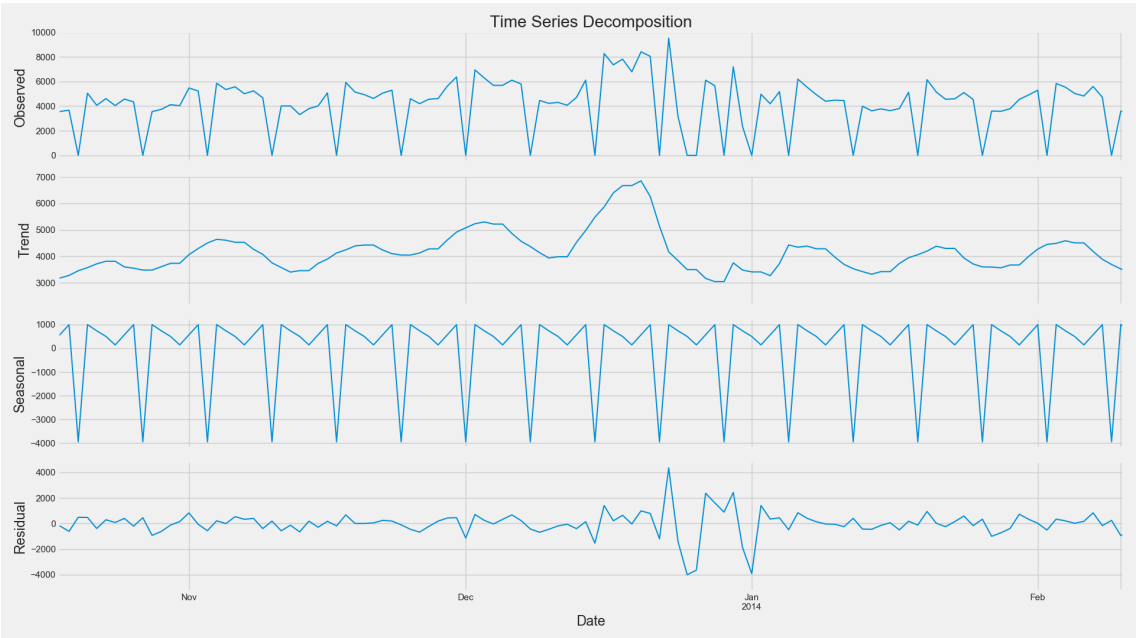


Figure 2.4: Decomposition of the sales of retail store into its components: Trend, Seasonal and Residual

Chapter 3

Forecasting Models

3.1 How to Forecast Time Series ?

Time series forecasting is different from the standard regression tasks as we have an important constraint that we need to account for: order. The chronological order of the data makes it harder for an estimator to learn an overall model that can be used for a long time, as patterns may appear for some time and then disappear or the entire distribution of the data might change. The prediction of stock price is a relevant example of this problem as the behavior of the stock might change based on a new regulation which can change the entire distribution of the data going further.

3.1.1 Time Series Forecasting

Forecasting is the main reason we do time series analysis, the fundamental idea is to try and use the past observations to predict the future. The model that describes best the data will later be used to predict the future based on past records. A forecasting model is a functional representation that describes a time series, on this basis we will forecast the future. For univariate time series we have two main variables:

- Endogenous variables, which are the past values of the series.
- Exogenous variables (or explicative variables) that are external factors that can influence the value of the time series, these variables have to be deterministic (like calendar data) otherwise we will be using other time series to predict our own which is the case of multivariate time series forecasting.

In this case study we will **only be using endogenous variables**.
After identifying the best model that can fit the data, we can now start forecasting future values of the series. Here we have two approaches, we can forecast one step ahead or multiple step ahead.

3.1.2 One step ahead

$$\hat{y}_{t+1} = f(y_0, y_1, ..., y_{t-1}, y_t) \tag{3.1}$$

One step ahead strategy is very straight forward using the past values of the series the task will be to predict the next value of the signal.

3.1.3 Multi-Step

For the multi-step strategy however, things start to be a bit complicated as we are faced with two choices either we forecast all the values of the series up to y_{t+h} or we only forecast the last value of the horizon y_{t+h}.

Iterative

$$\begin{aligned} \hat{y}_{t+1} &= f(y_0, y_1, ..., y_t) \\ \hat{y}_{t+2} &= f(y_0, y_1, ..., y_t, \hat{y}_{t+1}) \\ &\vdots \\ \hat{y}_{t+h-1} &= f(y_0, y_1, ..., y_t, \hat{y}_{t+1}, .., \hat{y}_{t+h-2}) \\ \hat{y}_{t+h} &= f(y_0, y_1, ..., y_t, \hat{y}_{t+1}, .., \hat{y}_{t+h-2}, \hat{y}_{t+h-1}) \end{aligned} \tag{3.2}$$

The iterative approach is used generally by auto-regressive models, where each prediction is based on the past records which make it difficult for the model to predict values that are far away in the future directly (this is the case of the **AR** model), described by the equation (3.2). The weakness of this method is that it propagates the error committed in earlier forecasts to the future which might render the quality of long term forecasts unreliable.

Direct

$$\hat{y}_{t+h} = f(y_0, y_1, ..., y_t) \tag{3.3}$$

The direct approach uses the past values collected up to the present to forecast directly a future value, this process is described above in the equation (3.3). This approach is going to be used in our off the shelf deep learning estimators.

3.1.4 Cross Validation for Time Series

The traditional forecasting approach for time series is to train the model/estimator on a portion of the data (Train dataset) then forecast on a fixed sized horizon, and then compute the performance of the estimator on the test dataset.
The aforementioned strategy is not adequate for time series especially for those with huge number of datapoints. This comes from the fact that we cannot assume that the distribution of the series won't change with time.
On the other hand the fact that some external/exogenous variables have an effect that we cannot necessarily model/use when we make our predictions making the task of forecasting on a relatively wide horizon unreliable.

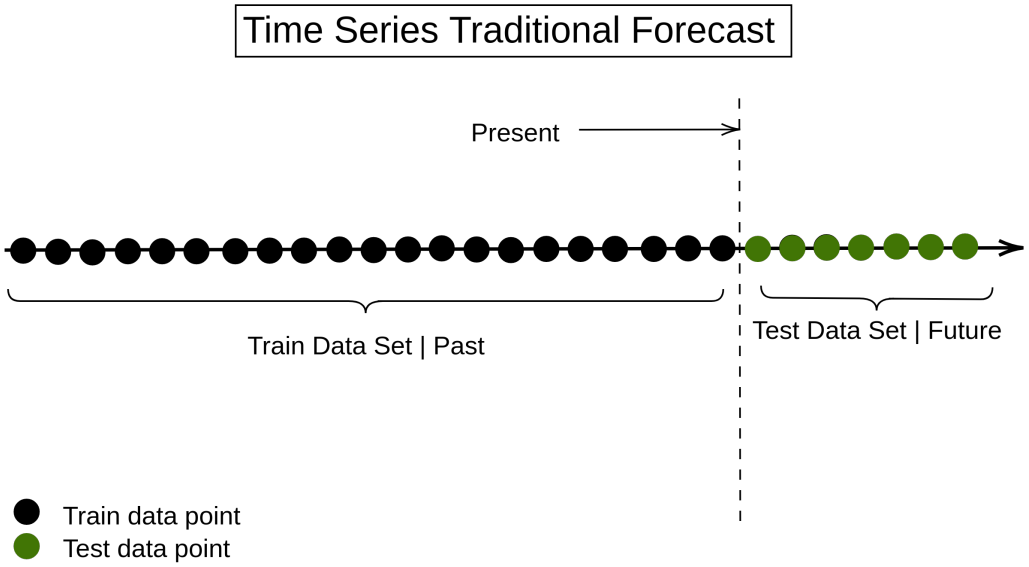


Figure 3.1: Traditional Train Test Forecasting Approach

So to prevent such issues, we use and approach called: **Walk Forward Validation**. This strategy that takes into account the granularity/frequency of the Time Series when predicting unseen values. From a business/practical point of view this is the better approach/implementation of the Time Series Forecasting problem as it is able to provide better predictions (using the maximum amount of information that it has up to the 'present'). As we assume that we can have good performances from our estimators up to a specific **sub-horizon**, that can be inferred from the granularity of the dataset, and then retrain the model on the same train dataset and the observations included in that **sub-horizon** that we've already predicted. This increases the knowledge of the estimators as they make their predictions and counter an issue that is rampant in the traditional approach which is adapting the model to the change in the overall trend of the series.

This strategy is inspired by the a campaign that used long time series. After examining closely the result we found that for this kind of datasets even with test ratio of 5% we are still forecasting a huge number of points at a time (3000 points for one case). For this reason we thought of this approach, where we provide the same number of forecasted points but we assume that after an inferred sub-horizon, based on the term (Short, Medium or Long Term) on which we forecast, the real data are available for use. Thus making it possible for the estimator to use this knowledge to adjust the learned model and provide better forecasts for the next sub-horizon. In our implementation, we have added a configuration section for time series cross validation [**TimeSeriesCV**] (Figure A.1), we put forth 3 terms: short term, medium term and long term. Based on this choice, we will select the appropriate sub-horizon on which we will forecast before retraining (exp: if the time series is daily and we specify that we want to make a short term prediction, then our cross validation method will infer a sub-horizon of 7 days, thus forecasting 7 values before retraining the model). Furthermore, we added a windowing option

that helps keep the forecast time similar at each fold. This option keeps the number of points fed at each train phase constant (Figure 3.2). This approach is done to emulate a real process of forecasting within a business and to provide an insight on the robustness/adaptability of the model over time.

From a practical/business point of view, this strategy makes a lot of sense as businesses generally make predictions on a small number of datapoints. In our implementation, we will provide 3 choices for the computation for the **sub-horizon**: **S**hort **T**erm, **M**edium **T**erm, **L**ong **T**erm.

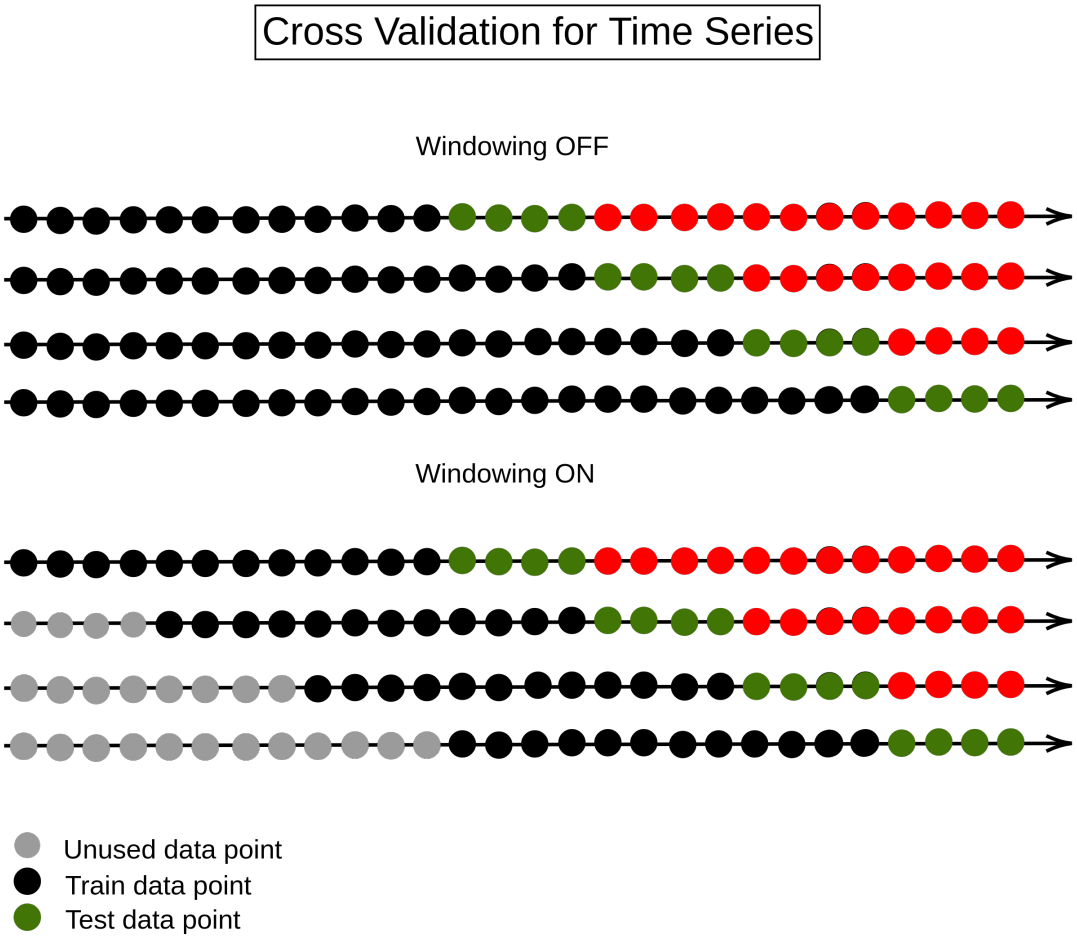


Figure 3.2: Cross Validation for Time Series

3.2 SAP Time Series Forcasting (SAPTF)

SAP uses its own time series forecasting model that will be used as a reference when studying the rest of candidate models, in terms of performance and speed. The aim of this study is to compare SAPTF to most commonly used estimators in time series forecasting, to identify its weaknesses and strengths. SAPTF is an automated model that is able to leverage exogenous variables to provide better forecasts. However, in the context of this study we will only be using endogenous variables. SAPTF uses an additive model to decompose time series as follows:

$$y(t) = T(t) + S(t) + F(t) + \varepsilon(t)$$

Where $y(t)$ is the original time series, $T(t)$ is the trend, $S(t)$ the seasonal component, $F(t)$ describes the fluctuations and $\varepsilon(t)$ is the residual component of the model. Unlike other estimators, SAPTF models components sequentially starting with the trend, seasonal component then the fluctuations that produces the lower error possible. SAPTF tests multiple trend models to find the one that best fits the train data and is able to handle sub-daily periods (seconds). Another capability, of this estimator is its ability to explain the model and provide the importance of each feature it uses during the estimation phase.

3.3 ARMA and its variants

3.3.1 AR(p)

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} \tag{3.4}$$

$$y_t = c + \sum_{k=1}^{p} \phi_k y_{t-k} \tag{3.5}$$

The auto-regressive model is the simplest model for time series, it comes from the idea that any value of a time series can be predicted based on the past recorded values. Mathematically, this formulation means that the future value of the time series, y_t, will be the linear combination of the past values with an intercept (3.5, 3.6), with $\phi_{i\in[1..p]}$ the weight of each observation. The parameter p of AR(p) determines the number of lags that will be used in predicting the value of y_t.

3.3.2 MA(q)

$$y_t = \mu + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q} \tag{3.6}$$

$$y_t = \mu + \sum_{k=1}^{q} \theta_k \varepsilon_{t-k} \tag{3.7}$$

$$\mu = \frac{1}{q} \sum_{t=1}^{q} y_{t-i} \tag{3.8}$$

The moving average model is similar to the auto-regressive model, except that the intercept is the average of the most recent q values and instead of performing an auto-regressive model on the observed records we regress on the residuals (difference between the average and the observed values). As described in the equations (3.7, 3.8, 3.9) the MA component is the linear combination of the latest q residuals with $\theta_{i\in[1..q]}$ the weight of each error correction. The component $\varepsilon_{t-i}, i \in [1, q]$ is called a random shock which is assumed to follow a normal distribution with zero mean and a constant variance σ^2.

A useful characteristic of an MA(1) model is that, it is the equivalent of an infinite AR(∞) model. The Δ operator will be further explained in the ARIMA subsection (3.2.5).

$$\begin{aligned}
y_t &= (1 - \theta_1 \Delta)\varepsilon_t \\
(1 - \theta_1 \Delta)^{-1} y_t &= (1 - \theta_1 \Delta)^{-1}(1 - \theta_1 \Delta)\varepsilon_t \\
(1 - \theta_1 \Delta)^{-1} y_t &= \varepsilon_t \\
\sum_{k=0}^{\infty} (1 - \theta_1 \Delta)^k y_t &= \varepsilon_t \\
MA(1) &\iff \mathrm{AR}(\infty)
\end{aligned}$$

3.3.3 ARMA(p,q)

$$y_t = c + \varepsilon_t + \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} \tag{3.9}$$

The ARMA(p,q) model is the combination of the previously mentioned AR(p) and MA(q), mathematically the model is simply the sum of these models as described by (3.10). But the weakness of these models is that they assume that the time series that we are working on are **stationary** which is not the case in most real life cases. Thus, we need to make sure that any time series modeled with this function needs to be **stationary**, for this purpose, we use the differencing operation to stationarize the series. This operation is handled by the ARIMA model which generalizes the ARMA model for non-stationary time series.

3.3.4 ARIMA(p,d,q)

Differencing (or back-shift) operator :

$$\Delta y_t = y_{t-1} \tag{3.10}$$
$$(1 - \Delta) y_t = y_t - y_{t-1} \tag{3.11}$$
$$\begin{aligned} \Delta^d y_t &= \Delta^{d-1} y_{t-1} \\ &= y_{t-d} \end{aligned} \tag{3.12}$$

With this operator we can now rewrite the AR(p), MA(q) and ARMA(p,q) model (for the sake of simplicity we'll ignore the constant term). Re-writing these formulas using the back-shift operator will simplify the mathematical representation of the ARIMA model.

AR(p)

$$\begin{aligned} y_t &= \phi_1 y_{t-1} + \phi_1 y_{t-2} + \ldots + \phi_p y_{t-p} + \varepsilon_t \\ y_t &= \sum_{i=1}^{p} \phi_i y_{t-i} + \varepsilon_t \\ y_t &= \sum_{i=1}^{p} \phi_i \Delta^i y_t + \varepsilon_t \\ y_t &= \prod_{i=1}^{p} (1 - \phi_i' \Delta) y_t + \varepsilon_t \\ \varepsilon_t &= (1 - \prod_{i=1}^{p} (1 - \phi_i' \Delta)) y_t \end{aligned}$$

$$\Phi_p(\Delta) y_t = \varepsilon_t \ , \ \Phi_p(\Delta) = (1 - \prod_{i=1}^{p} (1 - \phi_i' \Delta)) \tag{3.13}$$

MA(q)

$$
\begin{aligned}
y_t &= \theta_1\varepsilon_{t-1} + \theta_1\varepsilon_{t-2} + \ldots + \theta_p\varepsilon_{t-q} + \varepsilon_t \\
y_t &= \sum_{i=1}^{q} \theta_i\varepsilon_{t-i} + \varepsilon_t \\
y_t &= \sum_{i=1}^{q} \theta_i\Delta^i\varepsilon_t + \varepsilon_t \\
y_t &= \prod_{i=1}^{q} (1 - \theta_i'\Delta)\varepsilon_t + \varepsilon_t \\
y_t &= (1 + \prod_{i=1}^{q} (1 - \theta_i'\Delta))\varepsilon_t \\
y_t &= \Theta_q(\Delta)\varepsilon_t \ , \ \Theta_q(\Delta) = (1 + \prod_{i=1}^{q} (1 - \theta_i'\Delta))
\end{aligned} \tag{3.14}
$$

ARMA(p,q)

$$
\begin{aligned}
y_t &= \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{i=1}^{q} \theta_i\varepsilon_{t-i} + \varepsilon_t \\
y_t\Phi(\Delta) &= \varepsilon_t\Theta(\Delta)
\end{aligned} \tag{3.15}
$$

The most important characteristic of the ARIMA model is that it can handle non-stationary time series thanks to the d parameter which defines the number of differencing performed on the datapoints (3.13). Thus the ARMA model will be applied on the differentiated variable (3.18) and not on the original record.

$$z_t = (1 - \Delta)^d y_t \tag{3.16}$$

$$\Phi_p(\Delta)z_t = \varepsilon_t\Theta_q(\Delta) \tag{3.17}$$

$$\Phi_p(\Delta)(1 - \Delta)^d y_t = \varepsilon_t\Theta_q(\Delta) \tag{3.18}$$

Even though the ARIMA model is more equipped to handle larger array of time series (non seasonal and non stationary) it still lacks the ability to handle seasonal data as it does not model this component of the series in its mathematical formulation.

3.3.5 SARIMA$(p, d, q)(P, D, Q)^s$

The SARIMA model is a generalization of ARIMA, introduced in 1970 by Box and Jenkins, that is equipped to handle the seasonality component of periodic time series. In their generalized proposition, Box and Jenkins perform a second differentiation on the seasonal component of the time series. Thus, we add four more parameters to

the model which will be $(p, d, q)(P, D, Q)^s$. The model will be presented as follows:

$$\begin{aligned} y_t - y_{t-s} = (1-\Delta^s)y_t &= z_t \\ \underbrace{\Phi_P(\Delta^s)}_{\text{Seasonal AR}} \quad \underbrace{(1-\Delta^s)^D z_t}_{\text{Seasonal Differencing}} &= \varepsilon_t \quad \underbrace{\Theta_q(\Delta^s)}_{\text{Seasonal MA}} \end{aligned} \tag{3.19}$$

$$\Phi_p(\Delta)\Phi_P(\Delta^s)(1-\Delta^s)^D z_t = \varepsilon_t \Theta_S(\Delta^s)\Theta_q(\Delta^s) \tag{3.20}$$

$$\Phi_p(\Delta)\Phi_P(\Delta^s)(1-\Delta)^d(1-\Delta^s)^D y_t = \varepsilon_t \Theta_S(\Delta^s)\Theta_q(\Delta^s) \tag{3.21}$$

The SARIMA model is a generalization of all the previously mentioned models, that handles seasonal and non-stationary time series, as we can find a our back to any estimator with right choice of parameters $(p, d, q)(P, D, Q)^s$:

- SARIMA$(p, 0, 0)(0, 0, 0)^0$ describes an AR(p) model
- SARIMA$(0, 0, q)(0, 0, 0)^0$ describes an MA(q) model
- SARIMA$(p, 0, q)(0, 0, 0)^0$ describes an ARMA(p,q) estimator
- SARIMA$(p, d, q)(0, 0, 0)^0$ describes an ARIMA(p,d,q) estimator

A brief Summary of the family of auto-regressive estimators.

	Stationary	Non-Stationary	Seasonality
AR	✓	✗	✗
MA	✓	✗	✗
ARMA	✓	✗	✗
ARIMA	✓	✓	✗
SARMA	✓	✓	✓

Table 3.1: Methodology for Auto-regressive models

3.3.6 Auto-correlation and Partial auto-correlation function

Determining the time dependency/correlation between the recent records and past ones is an important step in time series modeling and forecasting because it helps to come up with an adequate model that describes best the records we posses and generate robust forecasts.

$$\mathrm{R}_{YY}(t_i, t_j) = \mathbb{E}[\,Y_i\, Y_j] \tag{3.22}$$

The auto-correlation function (3.4) computes the correlations that exists between the time series and its lags, at various points in time. The results of this function is a graph called correlogram (Figure (3.3)), this graph is an important part of tuning the parameters of the ARIMA model as it can reveal the presence of periodic pattern that might be hidden by noise. We also use the partial auto-correlation function as another way to determine time dependencies between recent and past values, which provides us with the partial correlation between the records at different time steps (Figure (3.4)).

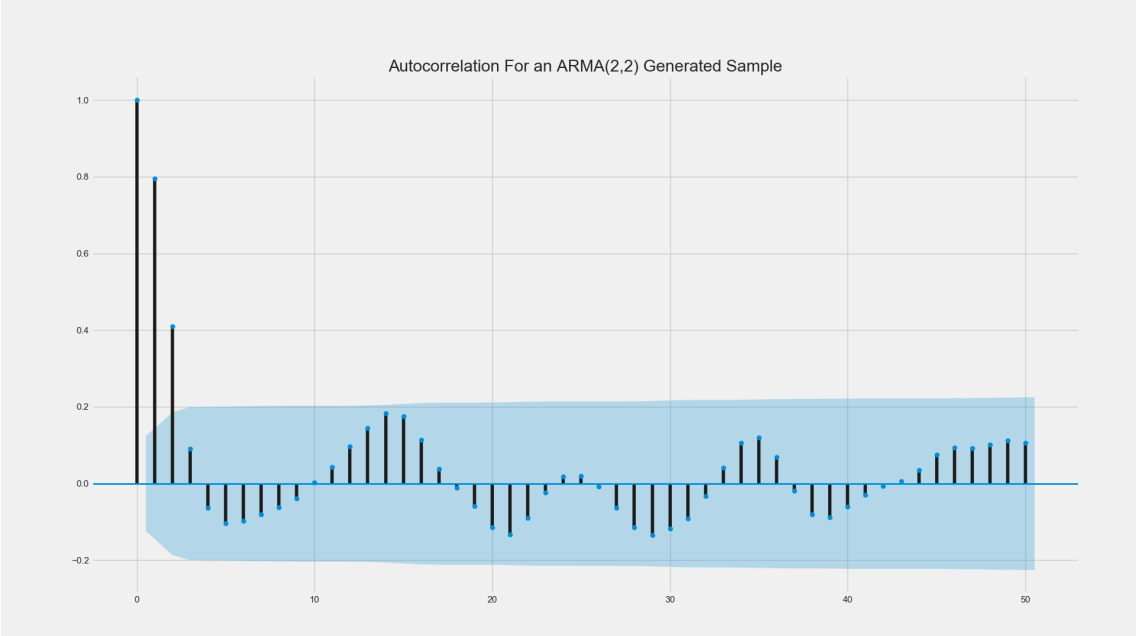


Figure 3.3: Example of an auto-correlation function on sample generated by an ARMA(2,2) process

3.3.7 Implementation

Since we aim to automate the process of forecasting, we implemented a helper function that will search for the best hyper-parameters that will provide the best results. During this process we will limit the dataset on which we will be fitting our models to the maximum value between 100 points and the latest $0.6\times$ the size of the train dataset, this is done to limit the time that each candidate spends on fitting the data, thus accelerating the process. This choice guarantees that we have a minimum number of points to fit the model on. This helper function, that we'll call AutoSARIMA, will first try and find a seasonal component in the time series based on the granularity of the data provided, using the date field. If this field is not provided then it uses a signal processing technique to extract the period. This is done by extracting the periodogram of the series. A simple inverting function will return the period. Once this value, is detected we conduct further tests to make sure that this value is not erroneous, where we fix a threshold of 3.5 periods that need to appear in the train dataset to conclude that this values is indeed a period. Once that is done and we know that we will be using SARIMA and not ARIMA, we run the PACF function to infer the most optimal AR order for the seasonal and non-seasonal part of SARIMA. The seasonal AR component P is detected by limiting the maximum number of lags in the PACF function to the value of the period and for the non-seasonal AR component p, we use a higher maximum lag parameter which is roughly a fourth of the size of the train data. Furthermore, we fix the value of the MA order for both the seasonal and non-seasonal component to 1. Having a higher order MA yields little to no improvement during the experiments that we conducted. This approach is further consolidated by testing 3 values of the differencing order d and D. Thus instead of running, a huge grid search that will

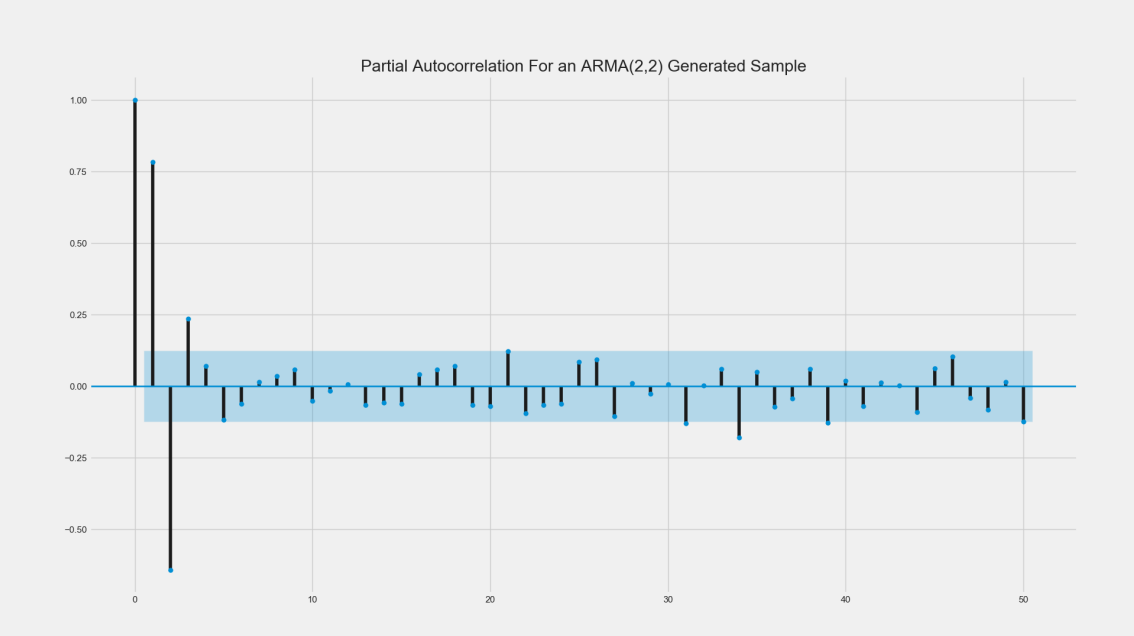


Figure 3.4: Example of a partial auto-correlation function on sample generated by an ARMA(2,2) process

take hours to yield any results (which we did at first, and led to an SARIMA model that can run for hours before providing a forecast), we use few heuristics and test a smaller number of parameter combinations which speeds the process. With this helper function, the average time for an automated SARIMA model is around 90 seconds with an average test MAPE performance of around 20%.
The SARIMA model was not re-implemented from scratch as we used the implement version of the estimator from the `statsmodels` library.

3.4 Exponential Smoothing and its variants

3.4.1 Simple Exponential Smoothing

$$\hat{y}_{t+1} = \alpha \sum_{i=0}^{t} (1-\alpha)^i y_{t-i} \tag{3.23}$$

The simple exponential smoothing model is named adequately as it describes the records as a repetitive revision of the forecasting function based on newly available values. Exponential smoothing, assigns exponentially decreasing weights to past values based on their distance from the present. Thus, newly observed records will have a more important impact on the forecast than past ones. The functional representation of the simple exponential smoothing model is detailed in the (3.23) equation.

3.4.2 Holt's Linear Trend

$$\hat{y}_{t+h|t} = \ell_t + hb_t \tag{3.24}$$

$$\ell_t = \alpha y_t + (1-\alpha) \underbrace{(\ell_{t-1} + b_{t-1})}_{\hat{y}_t} \tag{3.25}$$

$$b_t = \beta^* (\ell_t - \ell_{t-1}) + (1-\beta^*) b_{t-1} \tag{3.26}$$

Holt extended the simple exponential smoothing model to handle time series with trends. His formulation, include a forecast function (3.24) and two smoothing equations (3.25, 3.26). Where l_t refers to the level of the series at time step t (based on a weighted average of the actual value and the one step ahead forecast for time t), as for the trend equation it is also the weighted average of the previous estimated trend (computation of the slope) and the estimated trend at time step t. Though this model provides more description of the observed time series, it has a major flaw. The trend of this estimator is strictly monotone (always increasing/ decreasing) in the future, which something that is not entirely true on real world data, as continuous growth is very rare. For this purpose, we add a new parameter to this formulation to **dampen** the trend of the model over time (to resemble a logistic growth/decay). This is change to the formulation, is described in the ϕ parameter or the damping parameter (3.27) which will make sure that the trend converges to a constant after a few time steps. When $\phi = 0$, we get the classic Holt's linear trend model; otherwise if $0 < \phi < 1$ the forecasts will converge to $\ell_t + \frac{\phi b_t}{1-\phi}$.

$$\begin{aligned} \hat{y}_{t+h|t} &= \ell_t + \left(\phi + \phi^2 + \cdots + \phi^h\right) b_t \\ \ell_t &= \alpha y_t + (1-\alpha)(\ell_{t-1} + \phi b_{t-1}) \\ b_t &= \beta^* (\ell_t - \ell_{t-1}) + (1-\beta^*) \phi b_{t-1} \end{aligned} \tag{3.27}$$

3.4.3 Triple Exponential Smoothing - Holt-Winters

$$\hat{y}_{t+h|t} = \ell_t + (\phi + \phi^2 + \ldots + \phi^h)b_t + s_{t+h-m(k+1)} \tag{3.28}$$
$$\ell_t = \alpha\,(y_t - s_{t-m}) + (1-\alpha)\,(\ell_{t-1} + \phi b_{t-1}) \tag{3.29}$$
$$b_t = \beta^*\,(\ell_t - \ell_{t-1}) + (1-\beta^*)\,\phi b_{t-1} \tag{3.30}$$
$$s_t = \gamma\,(y_t - \ell_{t-1} - \phi b_{t-1}) + (1-\gamma)s_{t-m} \tag{3.31}$$

Even tough Holt's Linear Trend model extended the scope of application of the exponential smoothing estimators it still lacked the ability to capture a major aspect of the series component: seasonality. That is why Holt and Winters further extended the aforementioned model to reflect periodic patterns that may be present in time series. This model is defined by a forecasting function (3.28) and three smoothing functions (3.29, 3.30, 3.31). ℓ_t and b_t refer respectively to the level and the trend equations (like in 3.25 and 3.26), however the new addition s_t is the seasonal component of the model (3.31) that will help the estimator capture periodic patterns in the series. Due to the seasonal component of the model, there are two variations of the model:

- An additive model (3.28, 3.29, 3.30, 3.31), where the seasonal component is expressed in absolute terms (subtracted from the level component) (3.29) and then added back to the forecast at the end (3.28). This approach is generally adopted when the periodic variation are constant throughout the records.

- A multiplicative model (3.32, 3.33, 3.34, 3.35), where the periodic variations are proportional to the level of the series. In this approach, we adjust the level component of the model (3.33) by dividing the seasonal term by the level of the series and we adjust the forecast at the end by multiplying the estimated value by the seasonal component (3.32).

$$\hat{y}_{t+h|t} = (\ell_t + (\phi + \phi^2 + \ldots + \phi^h)b_t)s_{t+h-m(k+1)} \tag{3.32}$$
$$\ell_t = \alpha\frac{y_t}{s_{t-m}} + (1-\alpha)\,(\ell_{t-1} + \phi b_{t-1}) \tag{3.33}$$
$$b_t = \beta^*\,(\ell_t - \ell_{t-1}) + (1-\beta^*)\,\phi b_{t-1} \tag{3.34}$$
$$s_t = \gamma\frac{y_t}{(\ell_{t-1} - \phi b_{t-1})} + (1-\gamma)s_{t-m} \tag{3.35}$$

A brief Summary of the family of exponential smoothing estimators.

	Trend	Seasonality	Multiplicative Seasonality
SETS	✓	✗	✗
DETS	✓	✗	✗
TETS	✓	✓	✓

Table 3.2: Methodology for exponential smoothing

3.4.4 Implementation

We apply the same logic that we used when we created the helper function for the SARIMA model that automated the optimal parameter search. In the case of the TEST model, we created an AutoHolt function that will be the helper function that will find the best parameters for the model in a timely fashion. Here we have fewer heuristics. For this helper, we use also the periodogram technique to detect the period of the time series and conduct the same tests to make sure that this pattern is not temporary, we need to see at least 3.5 periods repeated in the train dataset to make the conclusion that the time series is in fact periodic. Here, we will use the heuristic proposed by Rob Hyndman to limit the damping parameter ϕ to]0.8,1[, this will helps chose more specific value for this parameter as we will choose values from that interval starting with 0.8 up to 1 with a step of 0.5. As for the rest of the parameters (α, β, γ) we will test values from [0.1,0.9] with a step of 0.1. Furthermore, we will test the additive and multiplicative modes of the model to see which on fits best. Once all the testing is done, we chose the best performing model. In this implementation, we used the R2 score as the performance metric.
We used the TETS model implemented in `statasmodels` library.

3.5 Fourier Transform Based Forecaster

The Fourier Transform based forecaster, is a decomposition based model. It decomposes the signal to its building blocks which will be later used to forecast and recreate the signal on a specified horizon H.

3.5.1 FFT Estimator, how does it work?

The fundamental idea behind the FFT estimator is that we will be using the discrete Fourier decomposition to extract the most important components of the series, then recreate a generalized version of the observations based on the extracted components then apply the inverse Fourier transform on those components.

1. De-trend the time series
2. Run the FFT decomposition on the de-trended time series
3. Filter the low-amplitude, high-frequency components
4. Forecast on each individual component
5. Run the inverse FFT on the new components.
6. Add the trend back to the forecasts.

$$Y_t = \sum_{n=0}^{N-1} y_n \exp^{-\frac{i2\pi}{N}tn} \tag{3.36}$$

$$= \sum_{n=0}^{N-1} y_n [\cos(\frac{2\pi tn}{N}) - i\sin(\frac{2\pi tn}{N})]$$

$$y_t = \frac{1}{N}\sum_{k=0}^{N-1} Y_k \exp^{i\frac{2\pi kt}{N}} \tag{3.37}$$

The equation (3.36) describes the discrete Fourier decomposition of the a series $y_t | t \in [0..T]$, whereas the equation (3.37) is the inverse transform of the discrete Fourier decomposition. The second step of the FFT prediction model is used to avoid over-fitting and avoid carrying the noise that may have found its way in the original records to the forecasts.

3.5.2 Forecasting on the number of daily births in Quebec data set with FFT

In this example we will be using an FFT estimator on the number of daily births in Quebec data set, with a Holdout of 5%, which is the equivalent to a horizon of 255. This data set is a big data set in the context of uni-variate time series forecasting with 5000 points. A portion of the test set is portrayed in Figure (3.5)

where we can see that overall the predictions are not very far from the targets iFFT prediction model is used to avoid over-fitting and avoid carrying the noisven*** that the horizon is very large and the distribution of the train and test might change over time.
We will detail the best practices concerning the FFT estimator use in the sixth chapter.

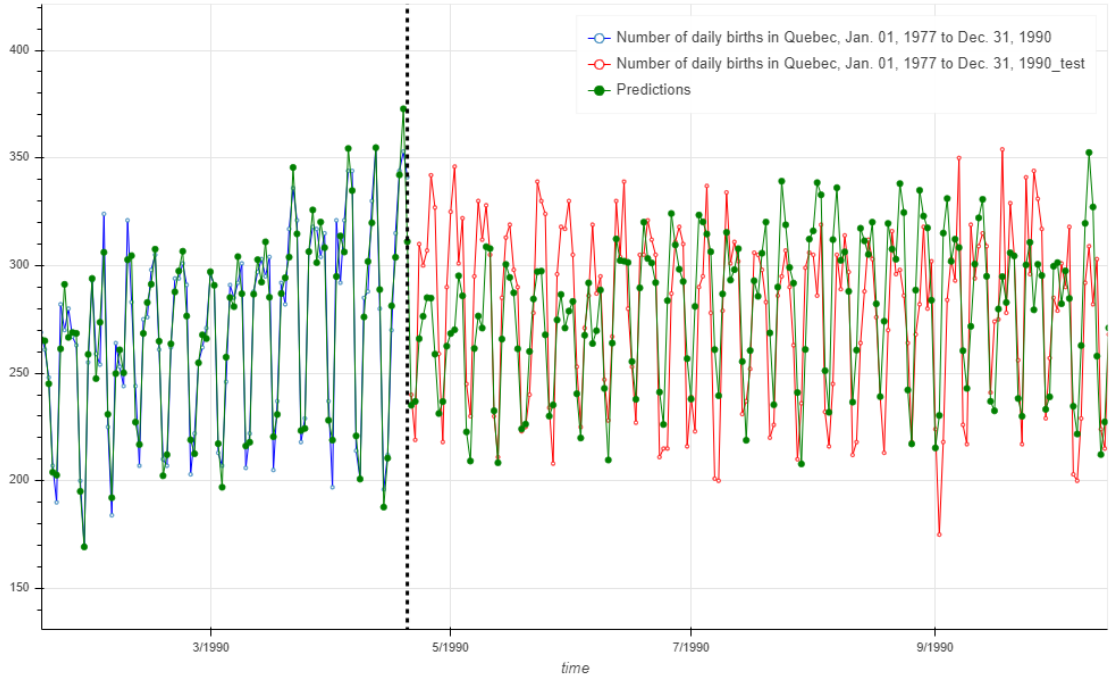


Figure 3.5: Forecast results on the number of daily births in Quebec data set

	Train			Test			Horizon
	MAPE	RMSE	R2	MAPE	RMSE	R2	
Ho=0.025	3.4	10.37	0.938	23.684	67.825	-1.438	127
Ho=0.050	3.418	10.4	0.937	10.137	32.54	0.41	255
Ho=0.100	3.437	10.437	0.936	23.345	67.496	-1.41	511

Table 3.3: Results on the Number of daily births is Quebec data set with different holdout [0.025, 0.05, 0.1]

3.5.3 Implementation

The method proposed above is not an estimator, as the forecasting part of the model is done by another time series estimator (on each harmonic). It is simply a technique that leverages the discrete fourier decomposition to simplify the model and divide the complexity of the signal on several harmonics. For this reason, this model will not be featured in Chapter 6 where we will exposes some of the results of the candidate models on a few datasets.

3.6 Facebook Prophet

Prophet is an open source project for time series analysis and forecasting, developed by Facebook. It provides a useful API, in Python and R, for off the shelf forecasting for people who lack deep expertise in time series modeling. Prophet provides a semi-automatic forecasting model with the ability to tune the model based on the user's prior knowledge and expertise, this approach is called, in the paper that details the model used behind Prophet, 'analyst in the loop'.

Prophet uses an additive time series model with three components:

1. A piece-wise linear or logistic growth function for modeling the **trend** component in the series (3.38)
2. A **periodic** component modeled by the (3.39) $s(t)$ function.
3. A holiday component, which is not present is most cases in the Prophet model as it has to be added manually by the user to be activated. This component adds external features to the time series that can help the user get better forecasts and properly predict for rare events.
4. An error components which accounts for the errors made when forecasting.

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t \tag{3.38}$$

$$g(t) = \frac{C(t)}{1 + \exp(-(k + \mathrm{a}(t)\mathrm{T}\delta)(t - (m + \mathrm{a}(t)\mathrm{T}\gamma)))} \tag{3.39}$$

$$s(t) = \sum_{n=1}^{N} \left(a_n \cos\left(\frac{2\pi nt}{P}\right) + b_n \sin\left(\frac{2\pi nt}{P}\right) \right) \tag{3.40}$$

$$\tag{3.41}$$

3.6.1 Useful add-ons to the traditional regression additive model

Prophet uses a modified trend detection model which leverages a piece-wise linear component to be able to adjust the level of the time series when there is a shift in the trend slope, to avoid missing out on local pattern in favor of the overall pattern in the series. Furthermore, the trend uses a logistic growth like function to model the non periodic evolution of the signal over time. This function has a useful parameter $C(t)$ which caps the value of the observations to a maximum or a minimum value (similar to the damped parameter for the TETS models). An example of the usefulness of this component is in predicting real time series such as the evolution of the number of users of a specific service, this value cannot exceed the number of humans on earth; or when predicting a stock price, we know for a fact that the price of a stock cannot be lower than 0. These small constraints on the model can go a long way to make sure that the model makes sense and provide explainable forecasts.

Another useful add-on, is the ability to detect changes in the trend's direction or slope and to leverage that in creating piece-wise linear models, that better describe local trends, and provide further analysis and insight on the behavior of the series. Facebook's tool also handles multiplicative seasonality and sub-daily frequencies, but these two options are not handled automatically and have to be provided manually.
One weakness of prophet, is that it cannot handle time series without timestamps as it uses dates to infer frequency and build the model, this weakness is dealt with in our code by returning the mean value of the train data set as the forecast. This problem is pain point for the prophet model, which consistently provides good forecasts, as it is not able to handle records with no timestamps which are quite a few in our case.

A brief Summary of the capability of Prophet:

	Sub-daily Frequency		Seasonality		Capped trend	Change-point detection
	Automatic	Manual	Additive	Multiplicative		
Prophet	✗	✓	✓	✓	✓	✓

3.7 Support Vector Regression (SVR)

3.7.1 Theoretical formulation of SVR

Usually Support Vector Machine models are used for binary classification as the main idea behind it, is to find a hyperplane that can maximally separate two classes. However, other variation of the model called SVR (or SVM for Regression) derived from the mathematical formulation of the SVM has been derived to generalize the scope of application of the model to regression. A subtle difference between the formulation of SVM and SVR is the outputs which are now continuous and not discrete and the loss function used in the quadratic programming problem, the $\varepsilon-$intensive loss function defined as follows:

$$L_\varepsilon(y, f(x,w)) = \begin{cases} 0 & \text{if } |y - f(x,w)| \leq \varepsilon \\ |y - f(x,w)| & \text{otherwise} \end{cases} \tag{3.42}$$

The associated quadratic programming problem can be written as follows:

$$\begin{aligned} &\text{Minimize} \quad \mathcal{Q}(w, \Psi, \Psi^*) = \tfrac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}(\Psi_i + \Psi_i^*) \\ &\text{s.t} \qquad\quad y_i - w^T\phi(x_i) - b \leq \varepsilon + \Psi_i; \forall i \in [1..N] \\ &\qquad\qquad\; w^T\phi(x_i) + b - y_i \leq \varepsilon + \Psi_i^* \\ &\qquad\qquad\; \Psi_i \geq 0 \\ &\qquad\qquad\; \Psi_i^* \geq 0 \end{aligned} \tag{3.43}$$

To solve (3.41) we use the Lagrange multipliers and we get the optimal hyperplane:

$$\begin{aligned} y(x) &= \sum_{i=1}^{S}(\alpha_i - \alpha_i^*)K(x, x_i) + b_{opt} \\ \text{with} \quad &\alpha_i \geq 0 \;\forall\; i \in [1..S] \\ &C \geq \alpha_i^* \;\forall\; i \in [1..S] \\ &\text{K the support vector kernel function} \end{aligned} \tag{3.44}$$

3.7.2 Implementation

For the purpose of our experiments, since we are manipulating uni-variate time series, we need to create the features that will be fed to the model to make predictions. We will use a moving window of size N, which will be determined automatically by our model based on some characteristics of the time series and the size of the forecasting horizon, this window will be fed to the model with the target value y_{t+H} (Figure 3.6). In all of our experiments we have used sklearn's SVR implementation, used with a linear kernel and $\varepsilon = 10^{-2}$.

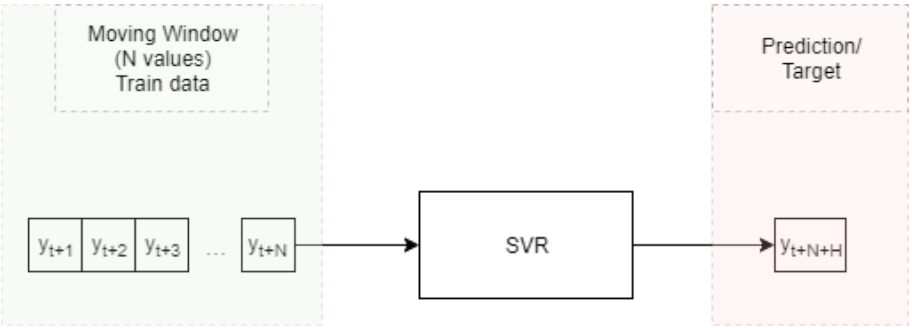


Figure 3.6: SVR model design

3.8 Deep Learning Approach

3.8.1 Neural Network

Neural Networks are being used a lot more frequently now, in time series forecasting, especially for long series. This is because NN are able to model complex non-linear models and self-adjust to the data. In this portion of the study, we will describe the methodology employed in our experiments. We will assume that the reader have basic knowledge of deep learning approaches and methodology as we will only provide an overview of the useful units and architecture to this task.

3.8.2 Recurrent Neural Network (RNN)

The aim of our methodology, for deep learning architecture, is to provide an off the shelf RNN model that is automatically configured and tuned based on the datasset provided, using heuristics adopted in our exploration phase, that provided the best results. Out of the RNN units that are present in literature, we've selected two units to constitute the layers of our models. These two units are the most widespread and used: **Gated Recurrent Unit** and **Long Short Term Memory**.

Long Short Term Memory (LSTM)

$$\begin{aligned} f_t &= \sigma_g\left(W_f x_t + U_f h_{t-1} + b_f\right) \\ i_t &= \sigma_g\left(W_i x_t + U_i h_{t-1} + b_i\right) \\ o_t &= \sigma_g\left(W_o x_t + U_o h_{t-1} + b_o\right) \\ c_t &= f_t \odot c_{t-1} + i_t \odot \underbrace{\sigma_h\left(W_c x_t + U_c h_{t-1} + b_c\right)}_{\text{candidate cell state } \tilde{c}_t} \\ h_t &= o_t \odot \sigma_h\left(c_t\right) \end{aligned} \tag{3.45}$$

The LSTM cell has two components that handle its state, the hidden state h_t and the internal state c_t. The hidden state corresponds to the short-term memory whereas the cell state corresponds to the long-term memory. One LSTM cell consists of a cell unit and three gates (input, forget and output). The gates use a sigmoid σ_g activation function and the input and cell state are usually transformed by tanh σ_h. The gating mechanism can hold information for long duration, this unit supports the

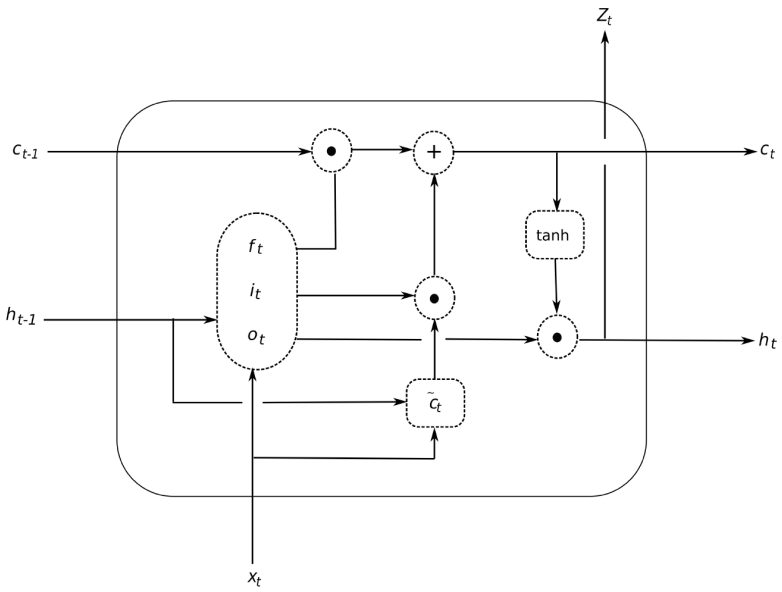


Figure 3.7: LSTM unit

Constant Error Carousel (CEC) which solves the problem of vanishing and exploding gradients. The addition of the CEC to the LSTM unit allows it to learn long-term patterns. In the equation (3.46), h_t denotes the hidden state vector of the cell at time step t, which captures the information that needs to be propagated further in the future. c_t and $\tilde{c}_t$ are the cell state and the candidate cell state at time step t. U_f, U_i, U_o, U_c are the weight matrices of the forget, input, output and cell state gate respectively. Similarly, W_f, W_i, W_o, W_c and b_f, b_i, b_o, b_c are the wight matrices and the bias vectors of the current input. In the expression of c_t, i_t and f_t determines how much of the previous information to retain in the current cell state and how much of the current context to propagate to the future. If the forget gate is set to 1 then the previous cell state should be retained and if it is set to 0 then nothing of the current state should be retained.

Gated Recurrent Unit (GRU)

$$\begin{aligned} z_t &= \sigma_g\left(W_z x_t + U_z h_{t-1} + b_z\right) \\ r_t &= \sigma_g\left(W_r x_t + U_r h_{t-1} + b_r\right) \\ h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \sigma_h\left(W_h x_t + U_h\left(r_t \odot h_{t-1}\right) + b_h\right) \end{aligned} \tag{3.46}$$

The GRU is another variant of a recurrent unit and is relatively simpler then the LSTM cell, which makes it faster in computation. The GRU unit has only two gating mechanism: the update and reset gates. The update gate combines the role of the forget and input gate of the LSTM unit. Furthermore, unlike LSTM, GRU has only one state component h_t. In the equation (3.47) u_t, r_t refer to the update and reset gates respectively. Whereas h_t denotes the current hidden state at time step t, in terms of the candidate hidden state $\tilde{h}_t$. The reset gate decides how much of the previous information of the previous hidden state to contribute to the candidate

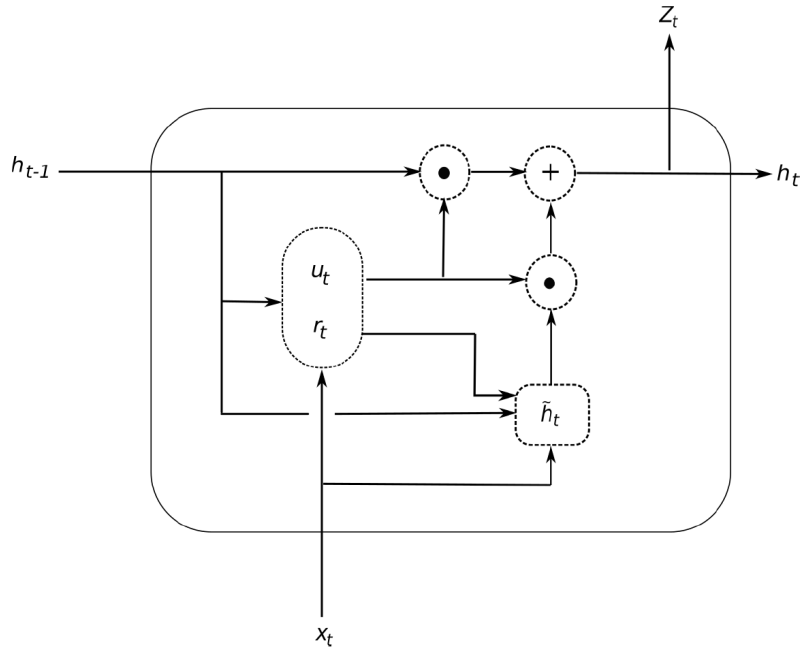


Figure 3.8: GRU unit

current one. Then for the computation of the current hidden state, the value of the update gate will be used to compute a weighted average of the current candidate hidden state and the previous hidden state.

3.8.3 RNN models architecture

In our experiments we opted for the Sequence to Sequence with Dense Layer architecture that uses a moving window as input and the y_{t+h} as the target. We use this architecture to forecast values up to the horizon h (predict h points), this choice is motivated by the fact that we want to study the predictive power of the model up to horizon h without any additional error that can be propagated by the fact that we re-introduce predicted values as input for the next prediction.

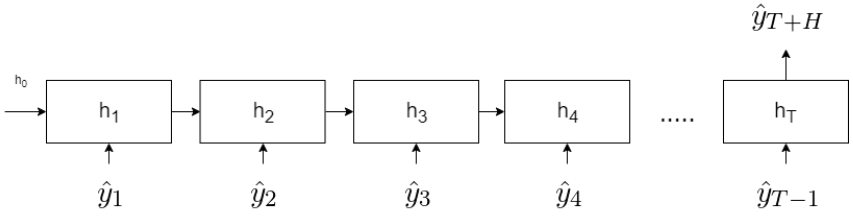
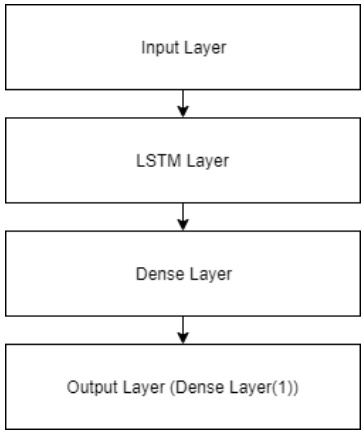


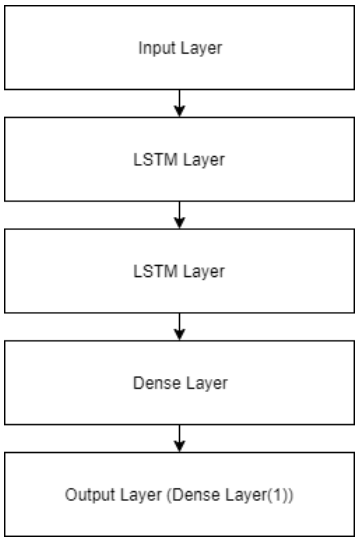
Figure 3.9: Sequence to Sequence with Dense Layer Architecture

During our benchmarking phase, we've experimented with four main models. All of them are automatically configured based on heuristics. Once the model is selected we start by configuring the architecture so that it handles best the given dataset, starting from the choice of the moving window size, the number of RNN

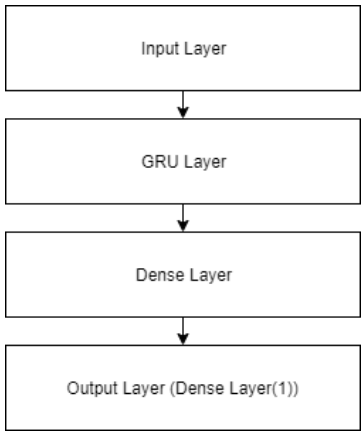
units per layer and the number of neurons in the Dense Layers. Once the model is properly configured, we train the model and then forecast up to the fixed horizon h.



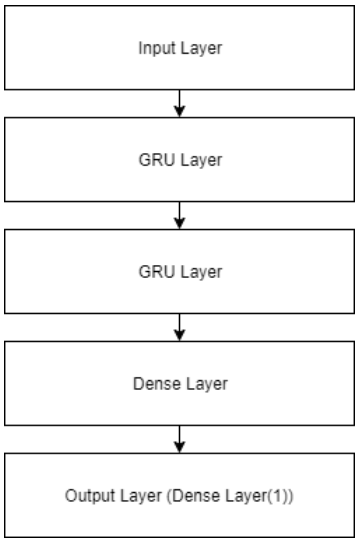
(a) LSTM Layer + Dense Layer



(b) LSTM Layer + LSTM Layer + Dense Layer



(c) GRU Layer + Dense Layer



(d) GRU Layer + GRU Layer + Dense Layer

Figure 3.10: The RNN model used during the experimentation phase

3.8.4 Wavelets + Neural Networks

In this section of the study we added a specific deep learning architecture that uses different level of wavelet decomposition to enrich the features that we can extract from a time series. We extract at most 4 levels of decomposition, these are used to extract more global patterns that we might lose sight of when using small moving windows.
This model is inspired by the wavelet model described in the thesis of **Chong Tan** :

Financial Time Series Forecasting Using Improved Wavelet Neural Network of May 2009, who proposed a similar architecture for "unpredictable time series".

3.8.5 Overview of the architecture

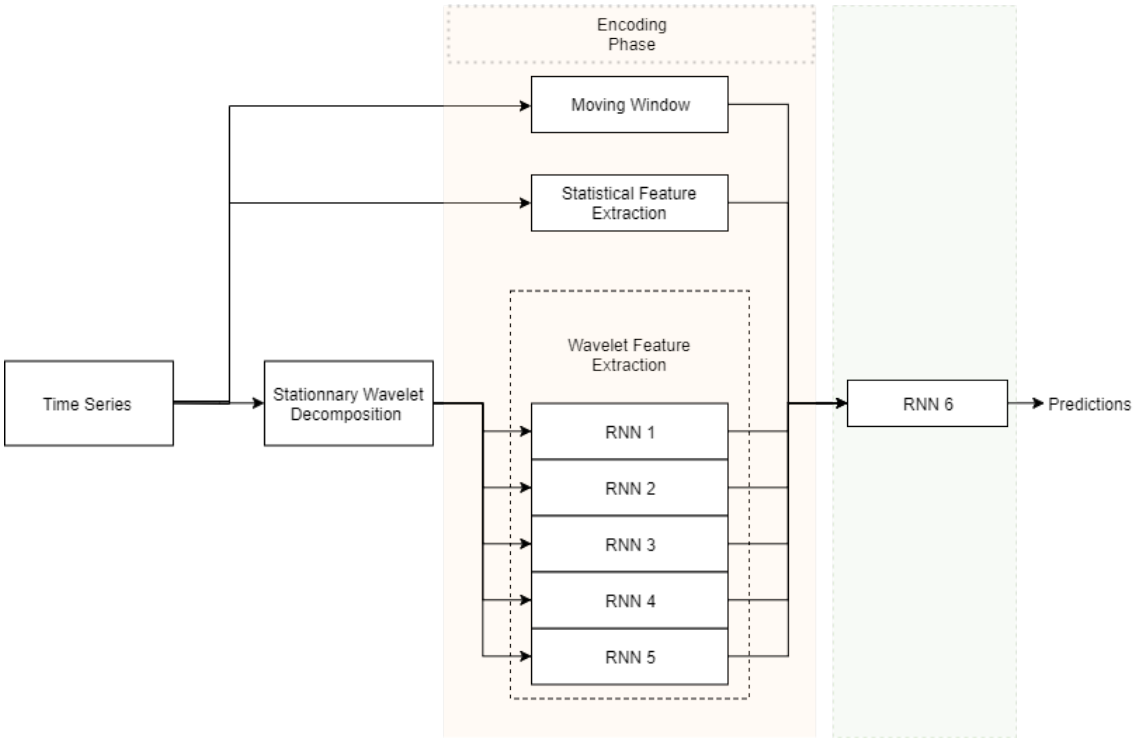


Figure 3.11: Design of the wavelet + RNN architecture

The basic principle of this architecture is straight forward, we decompose the time series to 4 different level of sub-sampling to extract global patterns that might be difficult for the network to perceive (similar to convolution when dealing with images). After decomposing the time series into an approximation of level 4 A_4 and 4 detail components D_1, D_2, D_3, D_4, we feed this input as a moving window to their respective recurrent neural network to extract useful features. At the same time, the model is also being fed statistical features of each of the moving window (train records of the time series) and the original records as moving windows. Then after all the features have been extracted (orange area), we concatenate the outputs of all the encoders and feed it to an RNN (a simple LSTM + Dense Layer) model to provide a forecast. In this implementation we used five statistical features : mean, mean average deviation, variance, kurtosis, skewness.
For all our intended purposes, we used the `pywt v:1.0.3` library for all our wavelet decomposition and transformation operations.

3.8.6 Implementation

When testing deep learning models, it was very obvious from the beginning that the choice of the size of the moving window is paramount as it is very correlated to the

performance of the model. This led us try different strategies. The first strategy was a naive one, where we the size of the window was hard coded and fixed for all datasets, which makes little sense when trying to automate a model.
This led us to a smarter approach, which calculated the size of the moving window based on the size of the train data. Though, we've seen that with some periodic datasets, deep learning models were able to capture the periodic pattern of the series but when the forecast charts were plotted these cycles have been shifted and were not in sync with the original observations. This observation and with a few other experiments led us to compute a window size based on the periodicity of the time series (if it exists) and the length of the forecast horizon. The main idea is to identify the period of the time series using the same methods in the SARIMA/ETS helpers (based on the periodogram). We use the value of the period as a jumping point for the size of the moving window. Then perform a few tests to consolidate our choice. The first test is to make sure that we have at least 3 values in a single moving window. The second is to test if the value of the period is not too small compared to the length of the horizon as this might limit the ability of the model to provide good forecasts. For this reason, we have to make sure that if the value of the period is not at least $\frac{\text{length of the horizon}}{2}$ then the new size of the scrolling widow will be $T \times \left\lceil \frac{Horizon}{2T} \right\rceil$ (the closest multiple of the period that satisfies the previous condition). In addition to the size of moving windows, we limit for small datasets the units used in LSTM layers and the number of neurons in the dense layer to speed up the computation as adding more parameters to the model will only make it more complex without any tangible gains. This engineering part of model implementation and experimentation proved very important in making the deep learning candidates competitive and on some datasets even better then the most used models. The wavelet model will not be a member of the candidates in our experiments in Chapter 6, as it is designed to be used for unpredictable time series. However, we will provide some forecasts done with this model separately.
The SVR model uses the same strategy in selecting the value of its moving window. All deep learning models were implemented using `keras` with a `tensorflow v1.13` backend, and they are set for 500 epochs with a checkpoint callback to save the best performing model and an early stopping callback function that stops the training phase if the performance have not increased for 30 consecutive epochs.

3.8.7 Hardware specification

All models have been run a single machine with an Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz with 8 cores (x86_64), 32GB of RAM and a single NVIDIA Quadro K1200 GPU with 4GB of dedicated memory.

Chapter 4

Time Series pre-processing techniques (Transformers)

4.1 ”New” pipeline for time series forecasting

Traditional machine and deep learning pipelines use the scikit-learn convention where we can stack (in series) pre-processors/transformers and the last attribute of the pipeline is the estimators (Figure 4.1).
Things are slightly different for univariate time series forecasting because changing the space of the training data set will change the space of future values since the only features we posses are the records, thus when we apply our transformers and make predictions these values are not necessarily correct because we need to revert back to the original space so that we can compute the performance of the model. For this purpose, we ”modify” the standard scikit-learn pipeline to include inverse transformation so that we can be able to extract the ”real” predicted future values. This time series pipeline will not feature an estimator at the end but a series of 'inverse' transformers to revert back to the original data space. Another reason for this change is that SAP benchmarking tool uses a similar pipeline for running models and to compute correctly the performance of each model the forecast and the real test values need to be in same value space.

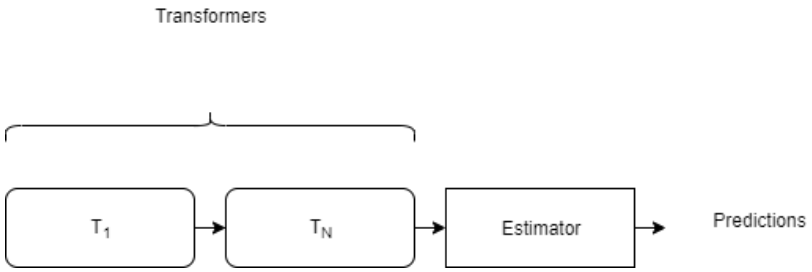


Figure 4.1: Time Series Pipeline

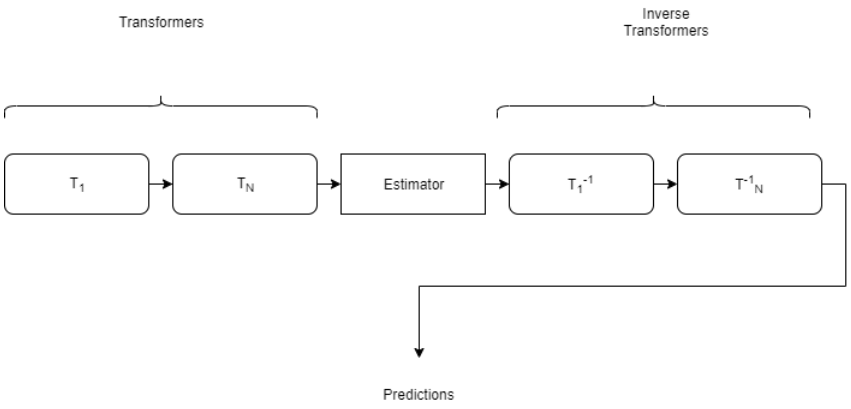


Figure 4.2: Time Series Pipeline

4.2 Data scaling

4.2.1 Log scaling

This transformer applies the log function on the values of the time series, when the creation of the pipeline is automated it is best used after a scaler such as Min Max Scaler to avoid any having values that do not belong the definition interval of the log function $(]0, +\infty[)$. This transformer is useful in some cases, where the outlier records are part of the series and convey an important information about the observed variable/phenomenon, as it shrinks the the value space and reduce the range of values and the skewness of the data set.

4.2.2 Min Max scaling

The Min Max Scaler is a scaling method used to express the values within the series in regard to its distance from the minimal and maximal value, in pre-specified range. When this transformer is applied with a range of $[a, b]$ the maximal value of the series will be set to b whereas the minimal value will be set to a. The scaler is mathematically formulated as follows:

$$
\begin{aligned}
Y_{tscaled} &= \frac{y_t - y_{min}}{y_{max} - y_{min}} \\
y_{max} &= \max_{i \in [1..T]} y_i \\
y_{min} &= \min_{i \in [1..T]} y_i
\end{aligned}
\tag{4.1}
$$

We used this scaler for two main reasons:

1. Since we are running our pipeline in an automated fashion, and some of them contain the log transformer we need to make sure to feed our model usable records. Thus we include the min max scaling with a range [a,b], such that $[a, b] \subset \mathbb{R}_+$. This use-case of the scaling technique is purely for stability concerns.

2. The second reason to include this technique is because it significantly accelerates the SVR model, which is very sensitive to the scale of the input data. The difference of performance, in forecast time, of the SVR model without scaling and with scale is huge.

4.3 Outlier removal

For most regression tasks, we use a plethora of techniques to remove outliers from the training data set that can bias our predictions. In time series, and for an automated model, this technique might be a double edged sword, why?
Removing outliers from data sets might be a good idea, because sometimes its origin can be faulty measuring equipment or mishandling of the data; but in other cases it can be an anomaly or a rare event. In former case, the removing outlier can be a useful technique to get better forecasting results, however in the latter case the whole forecasting task might be meaningless. For example, having the data set of incoming internet connection of a website reveal three points that lie at the tail of the value's distribution, if we remove these point then our estimator will lose its purpose of predicting the next spike in incoming internet connection. The point of this paragraph is to serve as a note for when to use (or not) an outlier removal technique, for an automated estimators.

4.3.1 Hampel filter

In theory the Hampel filter is a simple outlier removal/replacement technique, yet very effective. This technique is a configurable-width scrolling window that will run through the values of the time series. At each step, the filter will compute the median value and estimates the window's standard deviation σ (i.e: $\sigma = 1.4826\text{MAD}$, MAD = Mean Absolute Deviation). A point is considered to be an outlier if its value is larger than 3σ. The value of the outlier is then replaced by the average of the previous and next observation. An example of the hampel filter applied to a time series with artificially added outliers (Figure 4.3).

4.4 Noise reduction and signal smoothing

These techniques are used when the time series is drowned in noise and thus making it hard to analyze and hamper our ability to extract useful insights in modeling the records and can even hide periodic or cyclic patterns. These techniques are to be used carefully, because if we do not smooth the signal appropriately (aggressive smoothing) we might lose important information or remove patterns for the series.

4.4.1 Savitzky-Golay Filter

The smoothing process of the Savitzky-Golay filter is done by fitting low order polynomial functions on a width-configurable scrolling window that will run through the entire values of the time series. The length of the scrolling window and the order

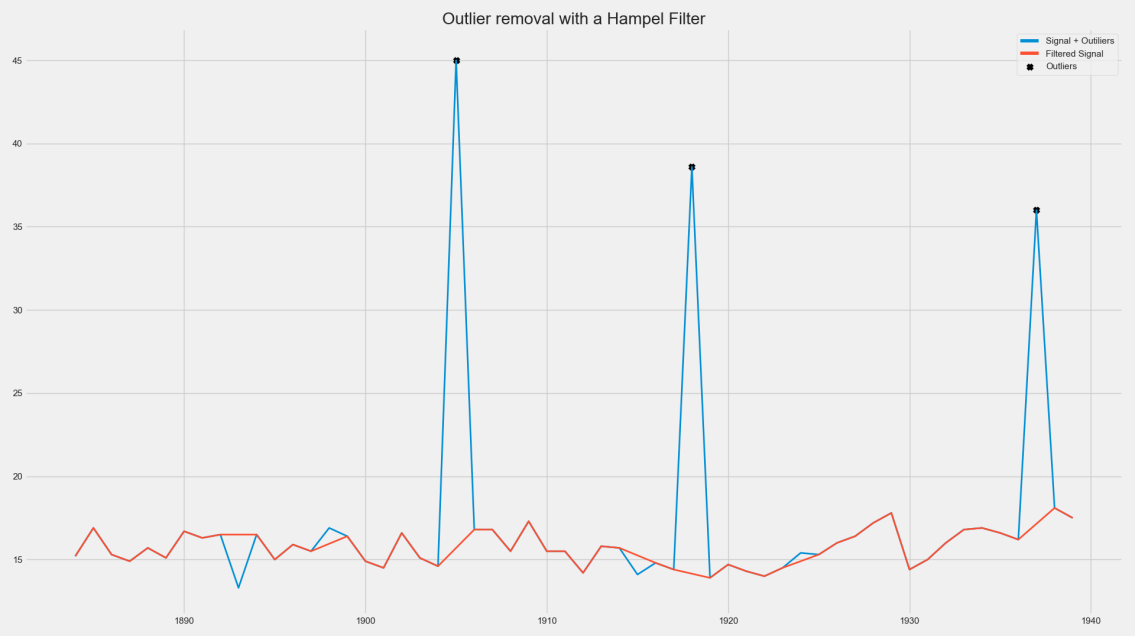


Figure 4.3: Outlier removal with a Hampel filter

of the polynom that will be fitted is configurable, in the example below (Figure 4.4) we used a window of size 15 with an polynomial order of 2. Keep in mind that the bigger the rolling window is, the smoother the signal will be; and the bigger the polynomial order the noisier the time series will remain.

4.5 De-trending

The de-trender is particular kind of transformer as within the process of fitting and transforming our data, we perform multiple polynomial regressions (more on this later). The purpose of the de-trender transformer is to stationarize the time series as best as possible.

4.5.1 How to detect the trend?

For trend detection we handle two **modes**:

- **Additive**, where the trend will be subtracted from the original observation and then added back with the `inverse_transform` method.
- **Multiplicative**, where the records will be divided by the overall trend of the series and then multiplied back with the `inverse_transform`.

4.5.2 Which trend detection mode to choose?

This transformer can be set up manually by the use of the campaign configuration file. Since we are trying to automate our model's pipeline, the mode will be detected

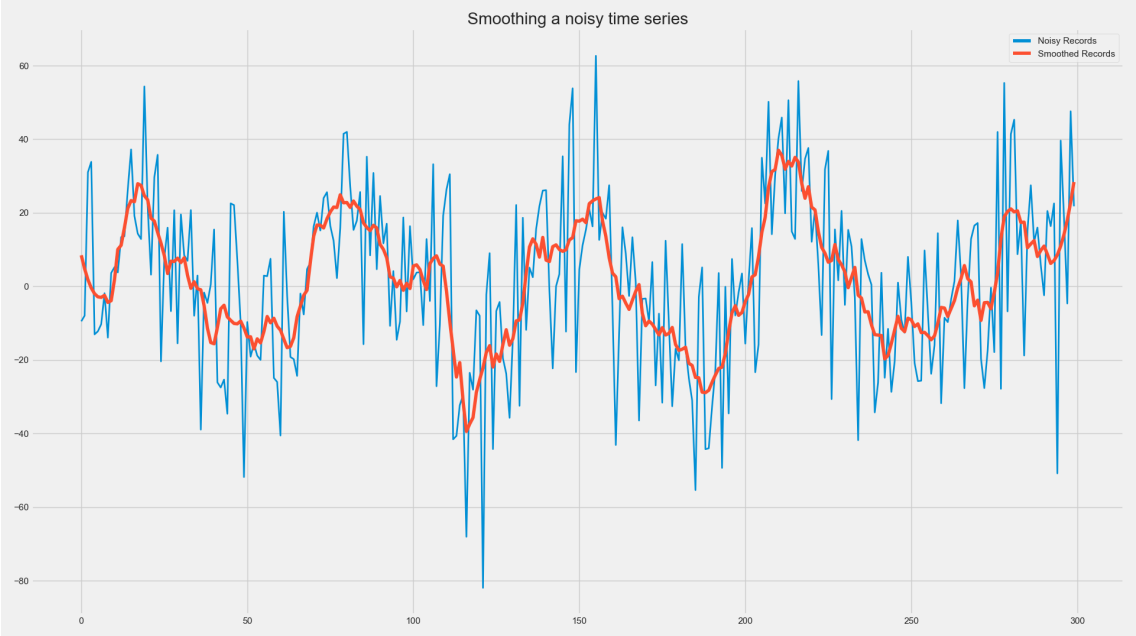


Figure 4.4: Smoothing a Time Series with a Savitzky-Golay Filter

automatically using statistical tests. The main idea is simple, we will perform 3 ad-fuller tests for each mode (in the worst case scenarios).
The testing process goes as follows:

1. We perform the ad-fuller test on the trend free time series for both modes, if one of the two tests return different answers to the question : **'Are the tested records stationary? (True or False)'**, then we return the mode that has answered True, otherwise the results are inconclusive and we move on to the second test.

2. On the trend free time series, we compute the scrolling average of the series. Then we perform the same test, as in the first step, on the new series. If the results are still inconclusive we move on to the third and last test.

3. In this test instead of testing on the rolling mean we will test on the scrolling variance. We are still testing our data based on the test in step 1. In this case, if the results are still inconclusive we compare the p-values of the tests performed and we return the mode that got the lower p-value in the test hypothesis process (ad-fuller test).

4.5.3 How to compute the trend?

The trend computation/prediction is a polynomial regression problem, but time series records can present some trend changes (growth, decay, stability, ...) so linear regression might not be able to capture this phenomena correctly (and thus adequately stationarize the data). For this reason, we test multiple polynomial regressions with different orders to get the best fit. But, we must make sure to not overfit the trend, that is why we limit the order (low orders, 3 maximum)of the

polynom that will estimate the trend and pick the best performing model by using an order penalizing loss function $\mathcal{L}_{trend}$ (4.2).

$$\mathcal{L}_{trend} = \frac{1}{4T}\sum_{t=1}^{T}(y_t - \hat{y}_t)^2 + \frac{3}{4} \times \text{ order} \tag{4.2}$$

Chapter 5

Forecast Metrics

5.1 Things to keep in mind when evaluating forecasts

5.2 A Few Forecast Performance Metrics

5.2.1 The Mean Absolute Error (MAE)

$$\text{MAE} = \frac{1}{N}\sum_{t=1}^{N}|y_t - \hat{y}_t| \tag{5.1}$$

Properties:

- It measures the average absolute deviation of forecasted values from original ones.
- It is also termed as the Mean Absolute Deviation (**MAD**).
- Affected by any transformation of the data
- Extreme forecast errors are not penalized by the **MAE**.
- MAE should be as close as possible to 0 for a perfect forecast.

5.2.2 The Mean Absolute Percentage Error (MAPE)

$$\text{MAPE} = \frac{100}{N}\sum_{t=1}^{N}|\frac{y_t - \hat{y}_t}{y_t}| \tag{5.2}$$

Properties:

- This metric measures the percentage of average absolute error that occurred when forecasting
- It is independent of the scale of measurement (e.g. MinMaxScaling), but affected by data transformation (e.g. Log Scaling)
- Does not penalize extreme forecast errors
- Provide a measure that is independent from the values observed in the series
- Ineffective in data sets where the original measures contain zeros (MAPE $\rightarrow \infty$)*
- Easily interpretable by people with no statistical background

* This issue can be avoided by using the modified formula of MAPE, SMAPE; but still the performance metric might diverge if the forecast and the real values are both 0.

$$\text{SMAPE} = \frac{100}{N} \sum_{t=1}^{N} \frac{2|\hat{y}_t - y_t|}{|y_t| + |\hat{y}_t|} \tag{5.3}$$

5.2.3 The Mean Squared Error (MSE)

$$\text{MSE} = \frac{1}{N} \sum_{t=1}^{N} (y_t - \hat{y}_t)^2 \tag{5.4}$$

Properties:

- It is a measure of the average squared deviation of the forecasted values
- It penalizes extreme errors during the forecast
- MSE emphasizes that overall forecast error is in fact much affected by large individual errors (outliers)
- It is sensitive to the change of scale and data transformation
- It is not easily interpretable metric as the value depends on the dataset and the values present within the Time Series (noisy data or the presence of outliers can hinder the judging of a model's performance)
- This metric is not comparable across all datasets

Another variate of the MSE is the **Root Mean Squared Error (RMSE)** which has the same properties as MSE and is formulated as follows:

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{t=1}^{N} (y_t - \hat{y}_t)^2} \tag{5.5}$$

5.2.4 The R2 Score

$$\text{R2} = 1 - \frac{\sum_{t=1}^{N}(y_t - \hat{y}_t)^2}{\sum_{t=1}^{N}(y_t - \bar{y}_t)^2} \tag{5.6}$$

Properties:

- Sensitive to extreme values (outliers)
- It is used to assess the goodness of fit in a regression model compared to a baseline approach. R2 = 0, then the performance is equivalent to the baseline approach (here the mean value), R2 > 0 then the performance is better then the baseline and R2 < 0 then the performance is worse then the baseline.
- It cannot indicate if a regression model provides an adequate fit to your data (a good model can have a low R2 score while a biased model can have a high R2 score, this can be observed mostly Long Time Series with a huge horizon to forecast on)
- Provides a measurement that is independent of the values observed in the series (useful for comparing the performance of a model on different data sets with different scales/range of values)

5.2.5 Theil's U-Statistics

$$\text{U} = \frac{\sqrt{\sum_{t=1}^{N}(y_t - \hat{y}_t)^2}}{\sqrt{\sum_{t=1}^{N} y_t^2} + \sqrt{\sum_{t=1}^{N} \hat{y}_t^2}} \tag{5.7}$$

Properties:

- It is a normalized measure of total forecast error.
- $0 \leq U \leq 1$; $U = 0$ means that the forecasted values fit perfectly with the real values.
- This measure is affected by changes in scale and data transformations.

5.2.6 The Pearson Squared Correlation

$$P2 = \text{Corr}^2(\hat{y}, y) \tag{5.8}$$

Properties:

- Provides a measure of the link between the actual and the forecasted values
- The performance measure is bounded [0,1], which can provide good insight as to whether a model have good performance or not.
- P2 = 1, means the forecasted values have the same distribution as the real values
- This measure however can be misleading because if the correlation is equal to -1 then the P2 score will be equal to 1.

Chapter 6

Experimental Results and Discussions

In this section of the study we will be presenting the results of the campaigns run with the aforementioned engineering and models. All the models have been run on 189 datasets with different characteristics and lengths. All of the tested estimators are automated and were not manually tuned during any of these experiment. Here we will present the performance of each model on a handful of data sets, then the overall performance of the models on all 189 data sets in terms of performance (MAPE metric) and in terms of forecast time. The latter metric is important because we need to provide a good forecast but also in a reasonable amount of time otherwise the model is useless, even if it produces the best results, but takes hours to compute. All the graphs related to the following experiences will be appended in the appendix section.
Below, we will find the results exposed in tables, where each table is an experience where we use each of the proposed model on the data setsusing the traditional forecast method. **Ho** refers to the Hold Out portion of the data on which will forecast (the size of the holdout portion is the number of out of sample forecasting done by the model, we refer to this value as **horizon**). In this section all values of the MAPE metric are expressed in percentages.

6.1 Experiments and Results

	Pipelines	Train			Test			Horizon
		MAPE	RMSE	R2	MAPE	RMSE	R2	
Ho=0.10	SAPTF	**1.59**	**147.975**	**0.997**	5.311	845.188	-0.442	45
	Prophet	4.97	244.284	0.994	4.393	660.186	0.119	
	TETS	2.85	190.319	0.996	**2.291**	**362.571**	**0.734**	
	SARIMA	2.14	151.68	0.997	6.579	987.19	-0.968	
	SVR	3.54	248.927	0.993	5.24	839.145	-0.422	
	GRU	3.42	309.52	0.989	3.956	627.748	0.203	
	sGRU	4.56	446.604	0.978	5.29	877.224	-0.554	
	LSTM	3.14	298.171	0.99	10.14	1631.62	-4.377	
	sLSTM	3.14	295.751	0.99	5.75	923.99	-0.724	
Ho=0.15	SAPTF	**1.57**	**135.284**	**0.997**	4.1	709.97	0.252	68
	Prophet	5.06	241.069	0.993	**2.96**	**503.336**	**0.624**	
	TETS	2.42	166.57	0.996	4	602.852	0.46	
	SARIMA	2.149	144.146	0.997	3.175	511.8	0.611	
	SVR	3.08	220.413	0.992	5.105	764.338	0.133	
	GRU	3.21	276.382	0.988	6.08	900.708	-0.2037	
	sGRU	3.15	281	0.987	7.19	1026.818	-0.564	
	LSTM	3.21	286.201	0.987	5.56	867.256	-0.115	
	sLSTM	2.78	264.615	0.989	4.6	710.512	0.25	
Ho=0.20	SAPTF	**1.65**	**132.706**	**0.997**	3.4	**557.078**	**.75093**	91
	Prophet	5.062	234.734	0.992	5.18	771.896	0.521	
	TETS	2.142	153.444	0.996	2.92	495.71	0.802	
	SARIMA	2.243	145.418	0.997	**2.424**	393.91	0.875	
	SVR	2.7	217.609	0.988	4.19	663.097	0.647	
	GRU	3.41	291.818	0.979	9.41	1309.985	-0.377	
	sGRU	3.37	288.459	0.98	9.27	1292.168	-0.34	
	LSTM	3.25	283.523	0.98	8.87	1242.809	-0.239	
	sLSTM	3.47	298.228	0.978	6.03	923.83	0.315	

Table 6.1: Estimators results on the Australian Monthly Electricity Production data set with different holdout [0.1, 0.15, 0.2]

	Pipelines	Train			Test			Horizon
		MAPE	RMSE	R2	MAPE	RMSE	R2	
Ho=0.10	SAPTF	**7.51**	542.455	0.9	11.08	1235.79	0.209	45
	Prophet	10.14	**537.97**	**0.916**	11.31	1128.058	0.34	
	TETS	10.669	633.318	0.884	13.877	1487.525	-0.145	
	SARIMA	10.318	578.026	0.903	11.306	1263.42	0.173	
	SVR	12.25	792.057	0.796	**9.14**	**1035.719**	**0.44**	
	GRU	10.246	673.47	0.85	19.086	1847.255	-0.766	
	sGRU	13.506	842.277	0.77	9.398	1048.66	0.43	
	LSTM	12.24	768.018	0.809	13.986	1378.98	0.015	
	sLSTM	11.83	860.52	0.76	25.085	2439.6	-2.08142	
Ho=0.15	SAPTF	8.659	548.846	0.88	**11.16**	**1108.19**	**0.46**	68
	Prophet	9.729	**493.52**	**0.916**	12.87	1194.355	0.37	
	TETS	9.19	521.94	0.906	37.52	3459.23	-4.25	
	SARIMA	9.099	515.29	0.909	13.79	1256.457	0.306	
	SVR	12.25	850.386	0.71	14.47	1510.77	-0.0019	
	GRU	8.25	616.01	0.849	20.33	2114.78	-0.96	
	sGRU	**7.225**	523.929	0.89	12.79	1344.207	0.20679	
	LSTM	17.81	1064.876	0.55	15.628	1497.339	0.015	
	sLSTM	9.83	769.879	0.764	41.944	3880.9	-5.61	
Ho=0.20	SAPTF	10.34	573.987	0.855	22.7	2251.214	-1.41	91
	Prophet	9.65	**474.96**	**0.9**	11.99	1147.45	0.372	
	TETS	8.98	511.659	0.885	**9.937**	**1073.389**	**0.45**	
	SARIMA	9.319	511.838	0.88521	16.935	1756.786	-0.47	
	SVR	10.11	659.98	0.769	29.9	2806.869	-2.75	
	GRU	9.559	684.439	0.75	28.71	2727.697	-2.54	
	sGRU	**8.189**	628.59	0.79	31.417	3118.89	-3.63	
	LSTM	10.27	725.41	0.72	28.796	2749.69	-2.6	
	sLSTM	8.478	586.67	0.818	28.81	2734.25	-2.56	

Table 6.2: Estimators results on the Chocolate data set with different holdout [0.1, 0.15, 0.2]

	Pipelines	Train			Test			Horizon
		MAPE	RMSE	R2	MAPE	RMSE	R2	
Ho=0.10	SAPTF	**6.33**	19.27	0.797	8.397	26.23	0.61	182
	Prophet	6.35	**19.2**	**0.8**	**6.848**	**22.627**	**0.709**	
	TETS	7.55	22.81	0.716	52.53	148.097	-11.42	
	SARIMA	6.76	20.58	0.768	7.775	25.98	0.617	
	SVR	8.83	27.387	0.6	10.72	35.85	0.27	
	GRU	8.7	25.975	0.64	9.647	31.288	0.445	
	sGRU	9.668	28.94	0.55	11.66	37.716	0.194	
	LSTM	8.92	27.119	0.608	12.15	38.59	0.156	
	sLSTM	15.24	41.94	0.06	15.166	47.869	-0.298	
Ho=0.15	SAPTF	6.4	19.45	0.779	8.667	27.949	0.59	273
	Prophet	6.33	**19.016**	**0.789**	**6.7**	**22.58**	**0.73**	
	TETS	7.7	22.85	0.696	11.598	35.51	0.34	
	SARIMA	6.8	20.548	0.753	9.55	30.418	0.5166	
	SVR	**6.32**	19.92	0.768	7.418	24.84	0.677	
	GRU	6.867	20.73	0.749	7.23	25.457	0.66	
	sGRU	7.4	22.16	0.713	8.17	28.446	0.577	
	LSTM	10.638	30.857	0.44	12.955	45.477	-0.08	
	sLSTM	7.036	21.42	0.73	8.23	28.45	0.577	
Ho=0.20	SAPTF	6.395	19.48	0.775	6.867	23.17	0.71	364
	Prophet	**6.35**	**19.01**	**0.786**	**6.508**	**21.816**	**0.745**	
	TETS	6.86	20.2	0.758	9.126	31.054	0.485	
	SARIMA	6.83	20.543	0.75	10.949	36.627	0.28	
	SVR	6.476	21.39	0.73	7.884	27.14	0.606	
	GRU	6.9	20.72	0.748	7.25	24.866	0.669	
	sGRU	7.076	20.88	0.74	7.51	25.879	-3.63	
	LSTM	7.074	21.23	0.736	7.347	24.766	0.67	
	sLSTM	7.275	21.869	0.72	7.81	26.86	0.61	

Table 6.3: Estimators results on the Number of daily births is Quebec data set with different holdout [0.1, 0.15, 0.2]

	Pipelines	Train			Test			Horizon
		MAPE	RMSE	R2	MAPE	RMSE	R2	
Ho=0.10	SAPTF	21.267	137.658	0.8	39.495	256.818	-0.825	3
	Prophet	5.45	27.156	0.99	13.696	94.86	0.75	
	TETS	**1.87**	**9.487**	**0.999**	**1.8**	**16.267**	**0.99**	
	SARIMA	12.415	90.403	0.914	4.415	28	0.978	
	SVR	32.756	237.656	0.415	29.23	260.32	-0.8757	
	GRU	25.85	254.525	0.329	41.09	312.51	-1.7	
	sGRU	26.5	263.195	0.28	39.286	308.567	-1.635	
	LSTM	27.59	269.419	0.248	41.062	317.689	-1.79	
	sLSTM	26.057	258.696	0.307	39.349	303.498	-1.549	
Ho=0.15	SAPTF	27.826	176.319	0.528	33.609	418.41	-0.606	5
	Prophet	4.837	28.58	0.987	12.34	127.24	0.85	
	TETS	**1.97**	**9.68**	0.998	1.487	16.045	**0.997**	
	SARIMA	9.935	68.096	0.932	**0.9**	**7.42**	**0.999**	
	SVR	28.42	212.016	0.349	39.279	494.24	-1.24	
	GRU	23.508	186.82	0.494	41.81	524.528	-1.52	
	sGRU	26.39	201.55	0.412	43.14	541.95	-1.69	
	LSTM	22.378	190.33	0.475	45.58	553.775	-1.81	
	sLSTM	20.52	182.807	0.516	45.51	550.386	-1.778	
Ho=0.20	SAPTF	28.216	158.34	0.582	34.03	412.52	-0.935	7
	Prophet	2.43	10.936	0.998	18.02	166.16	0.685	
	TETS	**1.777**	**7.96**	**0.998**	**1.68**	**17.099**	**0.996**	
	SARIMA	14.014	97.034	0.844	8.559	88.662	0.91057	
	SVR	17.45	138.48	0.706	17.147	214.6	0.476	
	GRU	17.3	110.916	0.811	24.589	267.42	0.186	
	sGRU	41.98	262.37	-0.05	37.525	484.539	-1.67	
	LSTM	40.577	263.398	-0.06	38.13	492.83	-1.763	
	sLSTM	40.18	241.018	0.11	37.45	468.72	-1.499	

Table 6.4: Estimators results on the Cola Sales data set with different holdout [0.1, 0.15, 0.2]

	Pipelines	Train			Test			Horizon
		MAPE	RMSE	R2	MAPE	RMSE	R2	
Ho=0.10	SAPTF	N/A	6.048	0.84	37.48	**6.707**	**0.77**	3
	Prophet	N/A	5.72	0.859	49.098	7.898	0.68	
	TETS	N/A	**4.93**	**0.895**	291.42	46.23	-9.89	
	SARIMA	N/A	4.168	0.925	48.867	6.845	0.761	
	SVR	N/A	5.71	0.85	47.07	9.97	0.49	
	GRU	N/A	5.666	0.85	**29.355**	7.37	0.72	
	sGRU	N/A	5.619	0.856	40.439	7.555	0.709	
	LSTM	N/A	5.625	0.855	30.91	7.786	0.69	
	sLSTM	N/A	5.566	0.858	33.956	8.15	0.66	
Ho=0.15	SAPTF	N/A	6.22	0.83	31.28	**6.41**	**0.79**	5
	Prophet	N/A	5.81	0.85	38.59	7.21	0.738	
	TETS	N/A	**5.055**	**0.888**	196.034	37.062	-5.889	
	SARIMA	N/A	4.27	0.92	67.03	13.669	0.0628	
	SVR	N/A	6.742	0.8	28.115	8.22	0.66	
	GRU	N/A	6.19	0.83	20.455	6.93	0.758	
	sGRU	N/A	6.137	0.834	**19.05**	6.535	0.785	
	LSTM	N/A	6.269	0.827	26.677	7.17	0.742	
	sLSTM	N/A	6.63	0.806	25.74	7.28	0.734	
Ho=0.20	SAPTF	N/A	6.206	0.838	22.61272	6.341	0.757	7
	Prophet	N/A	**5.829**	**0.857**	28.4	6.59	0.737	
	TETS	N/A	6.977	0.795	153.29	31.8	-5.1	
	SARIMA	N/A	4.307	0.92	56.15	15.3	-0.41	
	SVR	N/A	6.742	0.84	28.63	7.16	0.69	
	GRU	N/A	5.99	0.84	29.91	6.61	0.736	
	sGRU	N/A	5.98	0.84	31.61	6.71	0.727	
	LSTM	N/A	6.379	0.819	22.716	**6.22**	**0.765**	
	sLSTM	N/A	6.35	0.82	**21.47**	6.24	0.764	

Table 6.5: Estimators results on the Temperature data set with different holdout [0.1, 0.15, 0.2]

	Pipelines	Train			Test			Horizon
		MAPE	RMSE	R2	MAPE	RMSE	R2	
Ho=0.10	SAPTF	0.715	127.74	0.989	**0.598**	**109.929**	**0.99**	3
	Prophet	**0.65**	**118.5**	**0.99**	0.98	158.565	0.98	
	TETS	1.179	247.289	0.96	4.066	682.805	0.695	
	SARIMA	4.717	2091.693	-1.8	1.6399	297.976	0.94	
	SVR	1.427	281.928	0.948	1.566	254.03	0.957	
	GRU	1.739	290.277	0.94	2.425	452.107	0.866	
	sGRU	5.785	1124.92	0.173	5.725	1321.53	-0.14	
	LSTM	5.476	1087.338	0.227	5.578	1310.2	-0.122	
	sLSTM	6.037	1206.19	0.049	5.69	1477.576	-0.427	
Ho=0.15	SAPTF	0.688	128.46049	0.988	1.746	287.92	0.95	5
	Prophet	**0.652**	**119.779**	**0.99**	1.259	221.72	0.97	
	TETS	1.246	261.28	0.95	**1.163**	**185.46**	**0.98**	
	SARIMA	5.388	2408.349	-2.995	1.664	280.13	0.954	
	SVR	1.669	322.587	0.92	3.72	595.526	0.793	
	GRU	6.177	1178.62	-0.015	7.8	1400.656	-0.139	
	sGRU	6.06	1134.995	0.058	7.47	1389.18	0.12	
	LSTM	5.4	1085.258	0.138	6.47	1495.55	-0.299	
	sLSTM	5.78	1129.739	0.066	6.808	1603.24	-0.493	
Ho=0.20	SAPTF	1.06	189.3	0.976	3.14	520.635	0.8	7
	Prophet	**0.44**	**85.192**	**0.995**	3.625	568.198	0.76	
	TETS	1.05	174.848	0.98	3.36	565.335	0.765	
	SARIMA	5.69	2487.945	-3	3.09	503.519	0.81	
	SVR	1.536	312.80419	0.93	**2.055**	**383.636**	**0.892**	
	GRU	4.599	812.71653	0.54	5.379	874.458	0.438	
	sGRU	5.66	1105.38316	0.157	5.59	1115.157	0.087	
	LSTM	4.98	1159.037	0.073	5.105	1312.44	-0.263	
	sLSTM	6.012	1213.53	-0.015	4.89	1300.677	-0.241	

Table 6.6: Estimators results on the Quarterly retail turnorver data set with different holdout [0.1, 0.15, 0.2]

	Pipelines	Train			Test			Horizon
		MAPE	RMSE	R2	MAPE	RMSE	R2	
Ho=0.10	SAPTF	**2.865**	16.709	0.956	9.507	47.59	0.41	15
	Prophet	3.036	**15.945**	**0.97**	4.8	**28.68**	**0.786**	
	TETS	3.069	16.6	**0.97**	6.01	29.89	0.767	
	SARIMA	5.109	34.119	0.869	**4.187**	23.53	0.856	
	SVR	5.157	28.83	0.878	9.14	45.875	0.45	
	GRU	3.684	21.32	0.933	7.706	37.938	0.626	
	sGRU	3.875	22.998	0.922	7.999	39.34	0.597	
	LSTM	3.456	20.016	0.94	7.64	37.35	0.637	
	sLSTM	3.369	19.79	0.94	8.19	39.318	0.598	
Ho=0.15	SAPTF	**2.58**	15.229	0.966	15.435	82.44	-0.756	23
	Prophet	2.878	**14.738**	**0.977**	4.18	29.969	0.767	
	TETS	5.85	30.78	0.9	**3.418**	**25.209**	0.835	
	SARIMA	4.34	34.407	0.874	5.874	32.343	0.729	
	SVR	4.82	29.389	0.833	9.516	55.803	0.195	
	GRU	3.86	22.13	0.905	5.485	33.1	0.716	
	sGRU	3.5	19.85	0.92	5.7	31.825	0.738	
	LSTM	3.58	20.82	0.916	5.68	34.1	0.699	
	sLSTM	6.696	39.338	0.7	9.3	61.636	0.018	
Ho=0.20	SAPTF	**2.679**	15.97	0.964	11	67.08	-0.364	30
	Prophet	2.9	**14.7**	**0.97**	4.99	32.73	0.675	
	TETS	3.17	16.49	**0.97**	**3.82**	**23.437**	**0.83**	
	SARIMA	4.425	34.86	0.874	4.125	26.248	0.79	
	SVR	5.145	30.292	0.782	9.87	56.658	0.026	
	GRU	3.115	18.517	0.918	5.83	39.279	0.532	
	sGRU	10.3	59.96	0.146	13.97	67.75	-0.39	
	LSTM	9.216	52.38	0.348	11.22	58.76	-0.046	
	sLSTM	8.548	47.21	0.47	8.739	57.23	0.006	

Table 6.7: Estimators results on the quarterly beer production in Australia data set with different holdout [0.1, 0.15, 0.2]

	Avg Perf Train	Avg Perf Test	Avg Diff Perf Test Train
SAPTF	12.263	16.955	4.691
Prophet	18.818	22.277	3.458
SARIMA	15.519	22.982	7.463
TETS	13.449	65.917*	45.677*
SVR	13.449	21.158	7.708
LSTM	17.41	23.187	5.776
GRU	16.623	21.914	5.29
sLSTM	17.634	24.762	7.128
sGRU	16.459	21.997	5.538

Table 6.8: Average performance of the candidates on 189 data set, the performance metric is MAPE

The high value observed as the average performance of the TETS on 189 datasets (Table 6.8) is biased by the fact that on a very few datasets where the model is unable to detect a period within the time series which causes the estimator performs very poorly. This is apparent in the boxplot presented below (Figure 6.3). These outliers change drastically the value of the average performance.

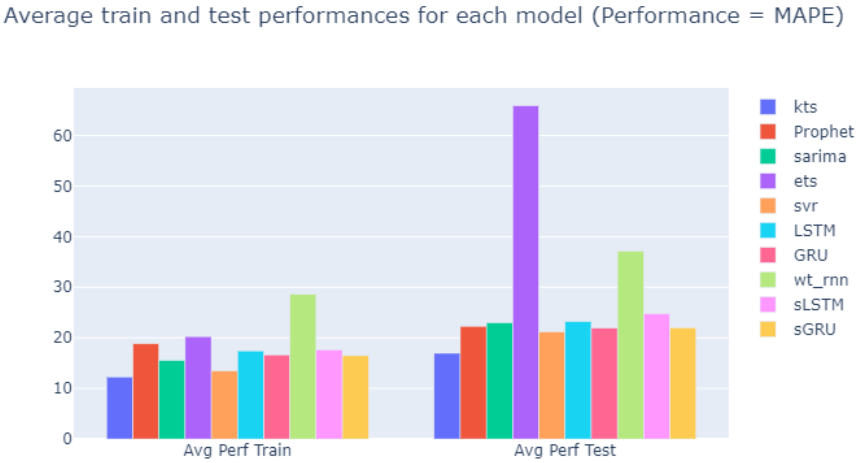


Figure 6.1: Overall performance measure in MAPE of all candidate models on 189 data set

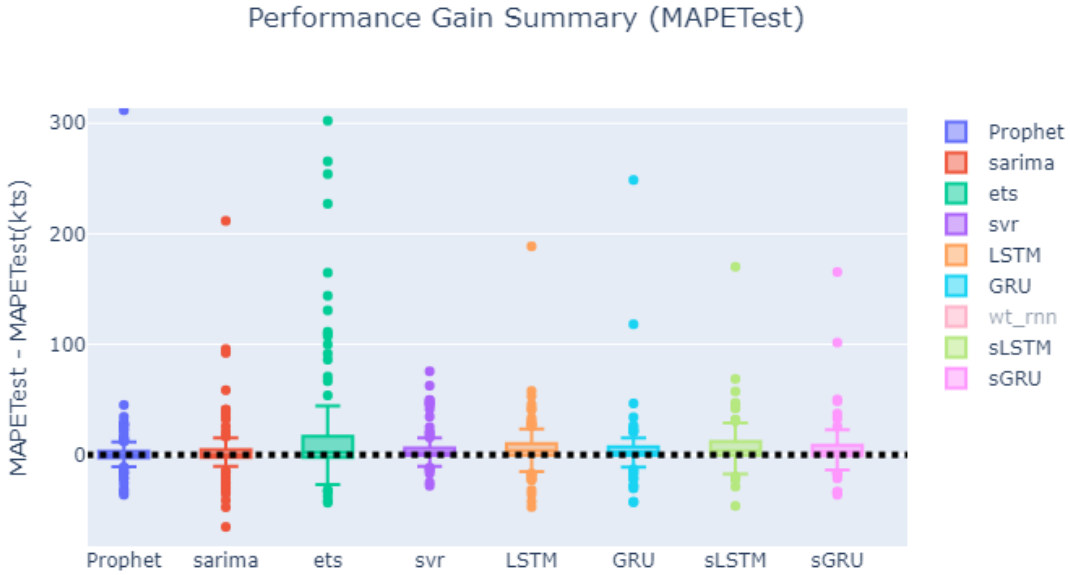


Figure 6.2: Relative performance of all models to SAPTF (MAPE metric)

	Avg Forecast time
Prophet	3.785
SARIMA	27.98
TETS	89.27
SVR	0.0377
LSTM	11.34
GRU	11.57
sLSTM	15.008
sGRU	14.19

Table 6.9: Average relative performance on forecast time (SAPTF forecast time)/Candidate forecast time, for small data sets

	Avg Forecast time
Prophet	0.86
SARIMA	23.62
TETS	30.7
SVR	0.469
LSTM	4.875
GRU	6.17
sLSTM	7.7
sGRU	8.2

Table 6.10: Average relative performance on forecast time (SAPTF forecast time)/Candidate forecast time, for small Large datasets

6.2 Takeaways

The key points to retain from these experiments is that stochastic models (SARIMA and TETS) struggle with handling time series with multiple seasonalities without any exogenous variables, the temperature dataset is a good example of this weakness. For time series, where it is hard to extract a seasonal component SARIMA (which gets reduced to ARIMA) and TETS (to DETS) cannot provide medium or long term predictions, as the predicted values tend to converge to the mean of the latest observed values.
Prophet has a performance that is very comparable to SAPTF model but struggles to automatically detect sub-daily periods and with datasets that does not have indexes in the datetime format.
As for deep learning models, they struggle to provide good forecasts for short time series as these models are by design made to capture complex patterns. RNNs are capable of modeling seasonality directly if the series in the dataset contains homogeneous seasonal patterns. So for deep learning models, it is essential that we have an adequate length of time series and a homogeneous pattern (no frequent sudden change-points).
However, SAPTF, SARIMA, TETS are the only explainable models. This is the strong suit of stochastic models compared to deep or even some machine learning models where we can provide good forecasts but we cannot necessarily explain why.

Overall, as showed in Table 6.8, SAPTF provides a better forecast on average (Average TestMAPE = 16.9%), even though it may struggle with some datasets, followed by Prophet, SARIMA, SVR and sGRU which have very similar performance (around 22% MAPE on average). And in last place comes, TETS but this does not reflect its true capability, as the value is biased by outliers. TETS, struggled with less than 5 datasets out of the 189 tested, for which it could not find a period. Thus the model reverted to the DETS that performed very poorly. But as we can see on the few example shown in Tables 6.1 to 6.7 TETS consistently provides excellent results.

These metrics are not the only factor that make a forecaster useful or not, the forecast time is an important value to keep in mind. This can be the reason why we choose a model over another in a production context. When handling short time series we see that SVR is 80% faster then SAPTF(the reference) whereas the rest of the estimators are slower than this estimator, with prophet coming second with 3 times slower. But these values (Table 6.9), even though they seem huge, only reflect a gain of a few seconds since the baseline is already small. But for the larger time series, where the number of datapoints per time series can exceed 1000, here we find once again SVR at the top (55% gain in forecast time), followed by Prophet(14% gain in forecast time) then SAPTF. The last two models in term of forecast time are automated SARIMA and automated TETS as they perform hyper-parameter search which is computationally heavy.

Chapter 7

Best Practices

In this part we will provide the reader with general best practices for univariate time series forecasting. This study puts forth an implementation of an automated version of the most commonly used statistical, machine and deep learning models, for the purpose of providing an easy to use interface that requires no prior knowledge about time series analysis nor forecasting. An example of how we create a model is provided in the appendix section (Figure A.1).

7.1 Stochastic Models

For stochastic models, the approach to automation is well established by using statistical tests or grid search approach for the detection of the best parameters. These two approaches are combined in our implementation and we have added some useful insight that might not be easy to detect, by using some signal processing techniques like periodograms. This approach helps us reduce the search space of the best parameters but also provides pertinent, useful and robust models. For example, in our datasets we encountered a few with no `timestamp` or `datetime` format index which makes period detection in some cases impossible using heuristics based on dates or long if we try to test multiple candidates, whereas using signal processing approaches we can detect multiple cycles in one go. Which strengthens the usability of the model put forth.
In the stochastic case, these models provide an explainable model with good performances. However, for time series that present multiple periods, they struggle to provide a good forecast. For this reason, we advise the use of exogenous variables to help the model adapt to the change of frequencies or using other models. Furthermore, for long time series, paired with the automatic search for the best parameters, providing a forecast can take a lot of time as these models are sometimes slow to optimize. The default space of parameters of our grid search is 9 (we test 9 SARIMA model and put them in competition, then we select the best performing model); this is done by limiting the search space by inferring some parameters based on statistical tests and/or signal processing techniques.
PS: The user have the option to override the standard configuration and provide his own search space for any model.

7.2 Machine/Deep Learning Approaches

These models are capable of modeling seasonality directly if the series in the data set possess homogeneous seasonal patterns. Machine and deep learning estimators use the same input (the moving window with targets at distance of horizon h). We've put forth some heuristics that gave the best performances in regard to all the others; the size of the window is a very sensitive parameter that is entangled with the size of the data set, the inherent characteristics of the series and how far in the future we are forecasting. That is why we use period detection techniques to characterize the signal and to choose an appropriate size of the moving window based on the value of the period of the signal (if it exists) and the size of the horizon (the window size must be a multiple of the period). This is done to make sure that we are not using a size of window that is too small compared to the horizon (Figure A.2). An example of why we made this choice, is that if we have a time series of yearly barley production, for example, from the 1800s to the late 2010s and if the horizon is way bigger then the moving window, we will be using values of the 1800s to predict values in the 1900s which is absurd as some effects that are not necessarily modeled in the series (advance in agriculture techniques for example) can drastically change the distribution values of the observations. Another useful practice is to limit the number of units and neurons used (for deep learning models) when dealing with smaller datasets to avoid over-fitting and useless computation, since deep learning model rarely perform well on small datasets which is a weakness that is inherent to deep learning. One of the reasons NNs perform poorly is that the time series themselves are either too short to be modeled using complex models or the characteristics of the time series have changed over time so that even long time series may not contain relevant patterns that can be used to predict the future. For deep learning approaches, it is imperative that the series have an adequate length and somewhat conserves its characteristics over time. The main weakness of these models is explainability, as we still struggle to provide justifications on the behavior of these models.

7.3 Performance Metrics

For the performance metrics, we strongly advise the use of at least **three** different metrics to assess the performance of the model. Some metrics may provide the wrong insight on the quality of the model compared to another metric. An example, which was striking in our experiments, is we had a MAPE value of 0.68% and R2 score of -0.23. So based on the R2 score the model performs poorly but based on the MAPE value the model is very close to the ground truth. After a close examination of the time series, we understand why we have such values as the time series varies slightly (in value) between train and test data sets, while predicting the mean value would have provided a better R2 score (Figure A.3). For this reason, and as we don't assume any prior about the time series that we handle, we have used 3 distinct performance metrics in our experimental results and discussion in Chapter 6.

Chapter 8

Conclusion

The aim of this study is to compare the performances most commonly used time series estimators with SAP's own model, SAPTF. This study was conducted on 189 datasets with different sizes and from a wide variety of fields (financial, business, meteorological, etc...). An aggregate of the data shows that SAPTF performs better, on average, then the rest of the other candidates. However, a few weakness have been discovered through this study and proposal have been made to correct them.

These campaign have been automatically launched using SAP's internal benchmarking tool, that provides an exhaustive comparison between the models across different datasets. Some performance issues have been encountered, when using deep learning models, that have been tackled by refining some engineering of the data. In addition, forecast time was a barrier at the beginning for stochastic models, as the grid search that we performed was computationally heavy. In order, to improve the forecasting time we adopted a few heuristics that helped us drastically reduce the forecasting time, making these models reasonably quick.

This study provides the strengths and weaknesses of the most used time series forecasting estimators and offers a new outlook on a new engineering approach for deep learning models that makes rival the other candidates. Experiments have shown that, on average, SAPTF comes on top in terms of performance (MAPE) followed very closely by SARIMA, Prophet, SVR and sGRU.

During this study another question was raised on how we can detect changes in the trend of time series. To this effect, we proposed an approach that leverages regression trees to identify these regions (Figure A.4).

8.1 Going Further

With the addition of exogenous variables, this work can be a precursor to the creation of an automated time series forecasting library. Furthermore, to build on the results of this study, we can suggest a recommender system that applies the best forecasting model based the characteristics of the given datasets.

References

[1] Jason Brownlee. *Statistical Methods for Machine Learning: Discover how to Transform Data into Knowledge with Python*. 1.4. 2019.

[2] Kasun Bandara Hansika Hewamalage Christoph Bergmeir. "Recurrent Neural Networks for Time Series Forecasting: Current Status and Future Directions". In: (2019).

[3] Tibshirani R. Hastie T. Friedman J. *The Elements of Statistical Learning*. en. 2nd edition. New York: Springer, 2017, pp. 389–415.

[4] Rob J Hyndman and George Athanasopoulos. *Forecasting: Principles and Practice*. 2nd ed. OTexts, 2018.

[5] J. Hyuk Han. *Comparing Models for Time Series Analysis*. en. Tech. rep. University of Pennsylvania, 2018.

[6] S. Makridakis, E. Spiliotis, and Assimakopoulos. *Statistical and Machine Learning forecasting methods : Concerns and ways forward*. en. 2018. URL: https://doi.org/10.1371/journal.pone.0194889.

[7] Robert H. Shumway and David S. Stoffer. *Time series analysis and its applications*. 4th ed. Springer, 2015.

[8] B. Taylor S.J. Letham. "Forecasting at scale". en. In: *PeerJ Preprints* 5.3190 (2017). URL: https://doi.org/10.7287/peerj.preprints.3190v2.

[9] Agus Widodo, Mohamad Ivan Fanany, and Indra Budi. "Enriching time series datasets using Nonparametric kernel regression to improve forecasting accuracy". In: *ICACSIS 2011 - 2011 International Conference on Advanced Computer Science and Information Systems, Proceedings* (Jan. 2011). DOI: 10.6084/m9.figshare.1609661.

Appendix A

Appendix

```
[Campaign]
Name = Campaigns/Preds
Description =
Result_file = test_campaign.json
Executor_type = SequentialExecutor
Models =  kts, svr, sarima
HoldOutRatios = 0.1,0.15,0.2

[Models]
sarima = sarimax.SARIMAX
svr = TSTransformers.MinMaxScaling, svr.SVRForecaster

[Datasets]
Tsk = K:/Development/Benchmarks/Benchmarks/TimeSeriesBench/Predicta-
ble/R_ozone-la.tsk_ts

[Report]
Type = reportcampaign
Output_file = Campaigns/Preds/test_campaign.html
Result_files = Campaigns/Preds/test_campaign.json
Indicators = r2Test, ForecastTime
DatasetIndicators = NumberLines
DatasetColorIndicator = TargetMean
CompareOn = r2Test
CompareTimeOn = ForecastTime
CompareRobustnessOn = r2

[Algorithms]
Reference = svr
Algorithm = svr
Algorithm = sarima

[TimeSeriesCV]
CV = False
Term = ST
Windowing = True
```

Figure A.1: A configuration file example, that will be used to automatically create and tune a given model (sarima and svr in this case)

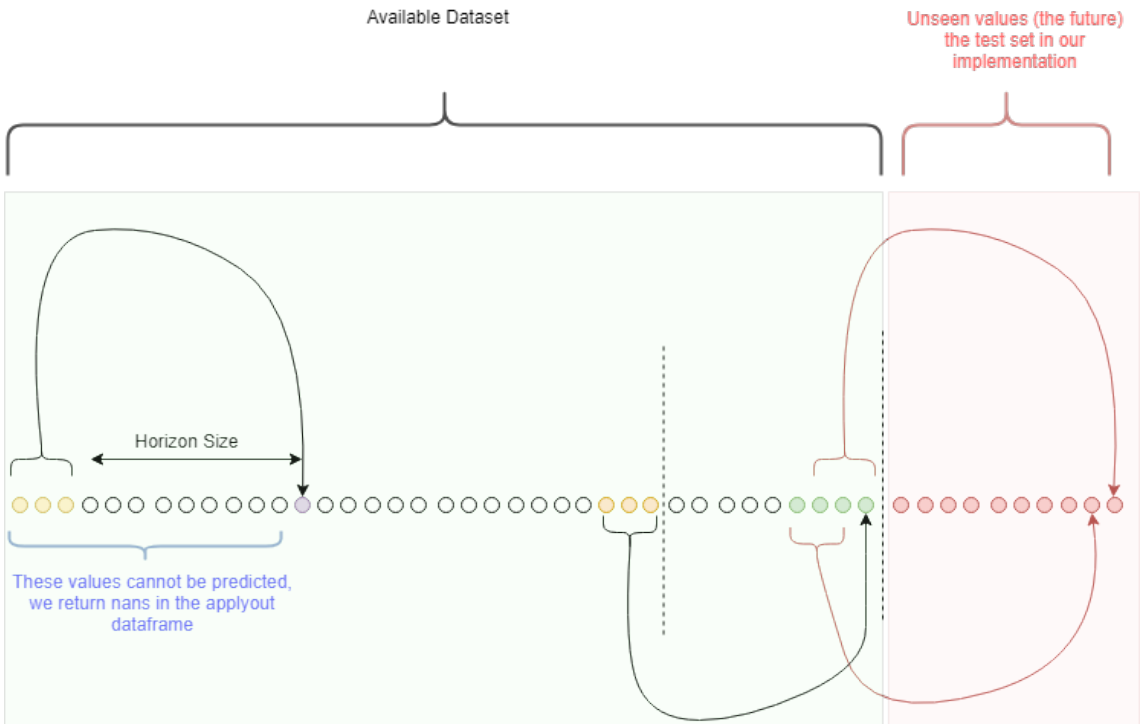


Figure A.2: An overview of the relationship between the window size and the horizon size

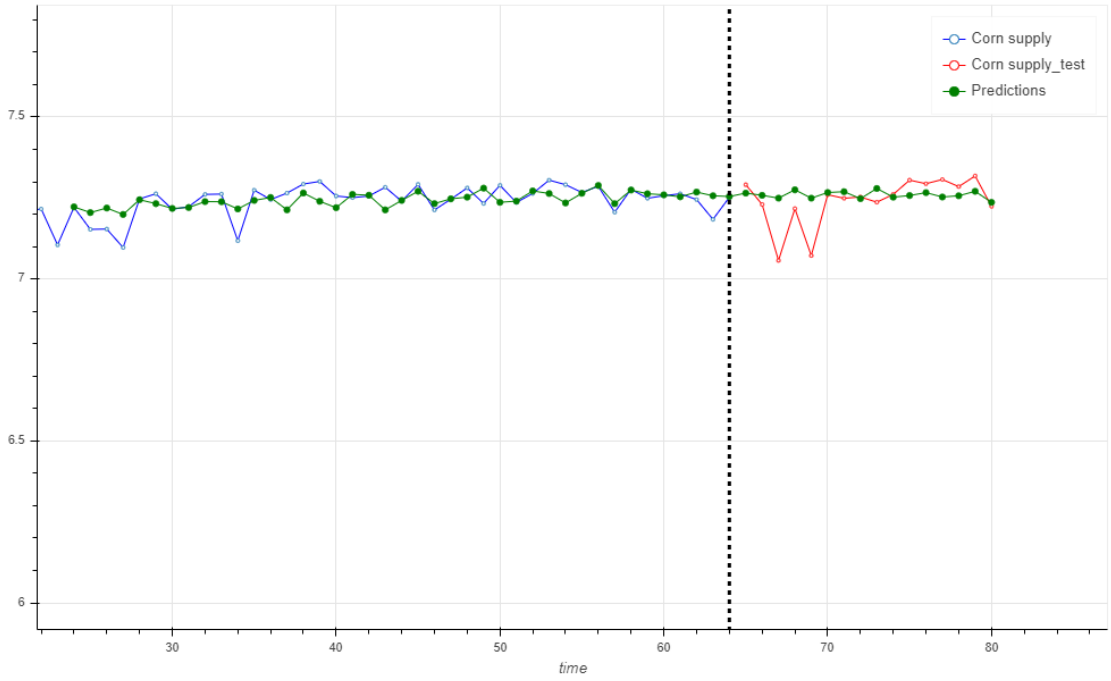


Figure A.3: Case of the inconclusiveness of the performance metrics R2Test=-0.23, MAPETest=0.68%

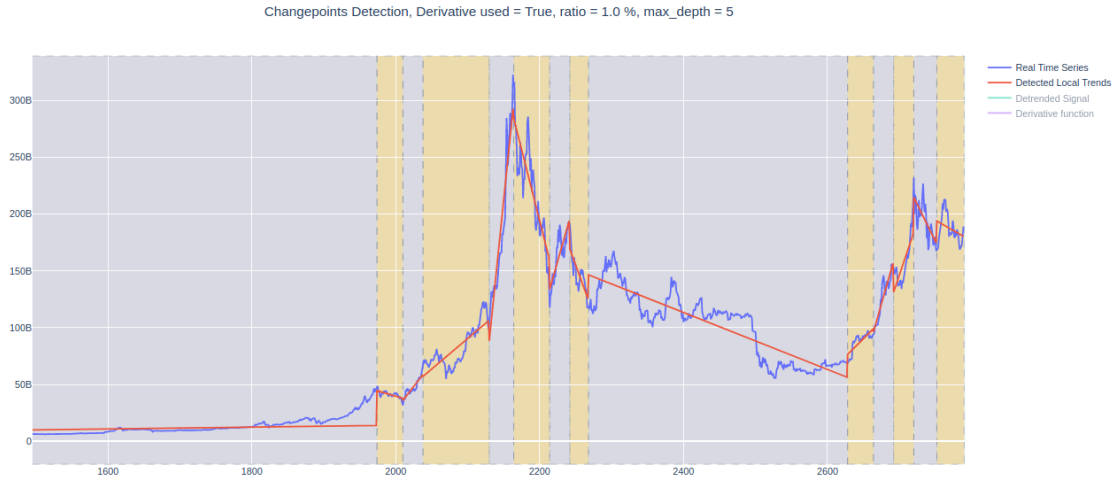


Figure A.4: The proposed model for changepoint detection

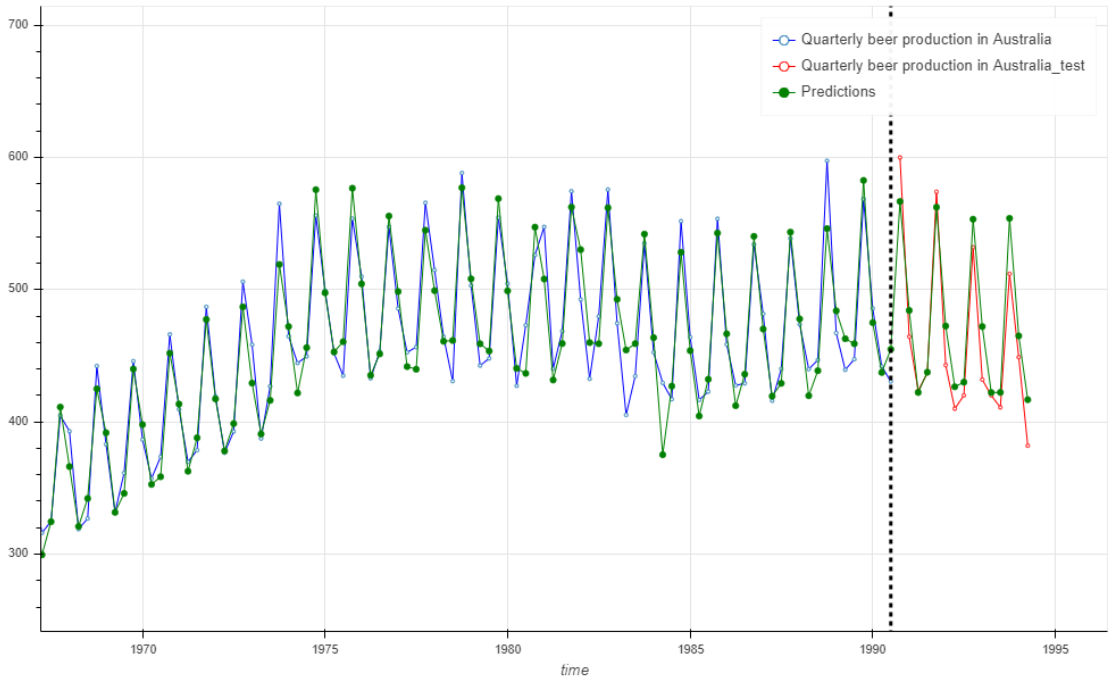


Figure A.5: Quarterly beer production in Australia: SARIMA(7,0,1)(2,0,1)4, Ho=0.10

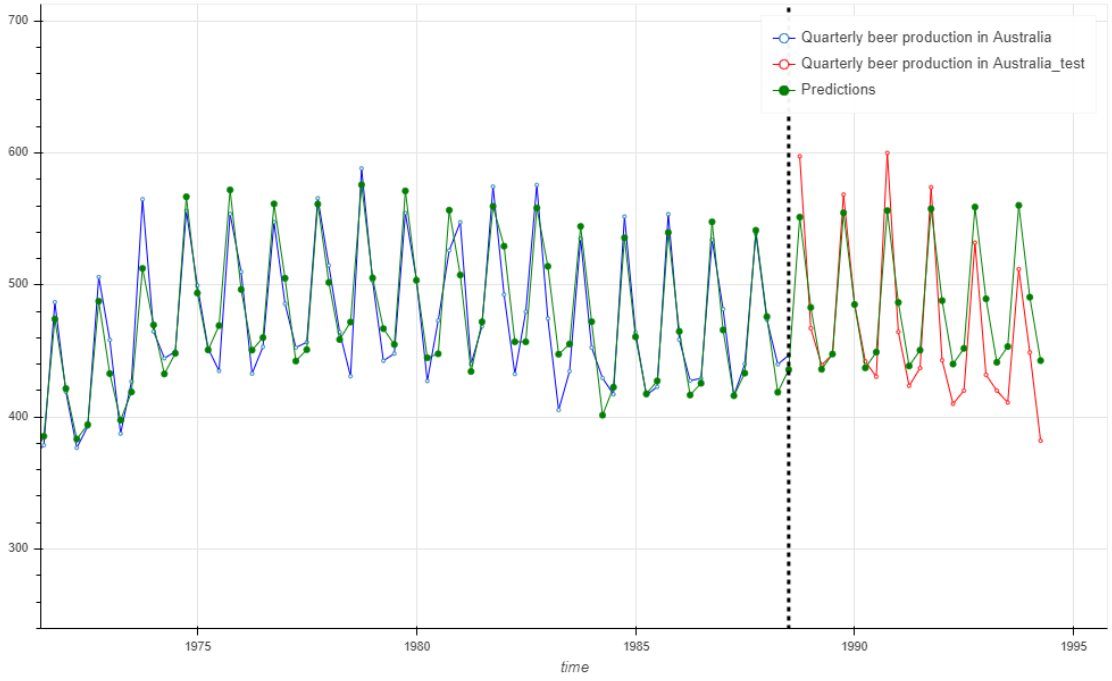


Figure A.6: Quarterly beer production in Australia: SARIMA(2,0,1)(2,0,1)4, Ho=0.15

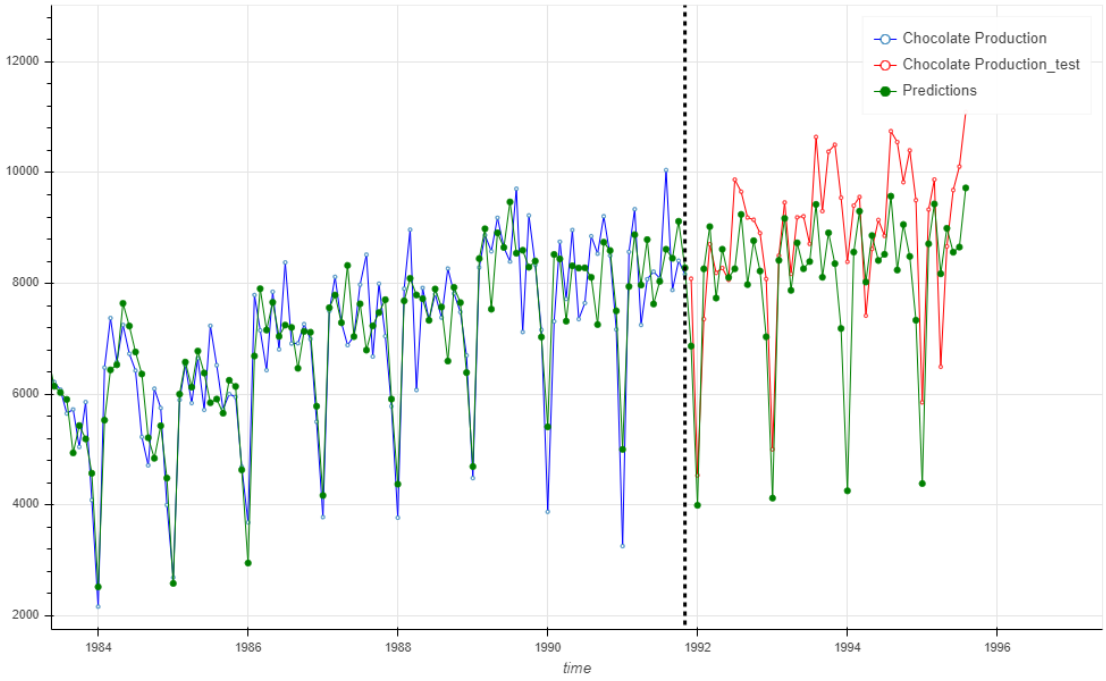


Figure A.7: Chocolate production : SARIMA(4,0,1)(2,0,1)12, Ho=0.10

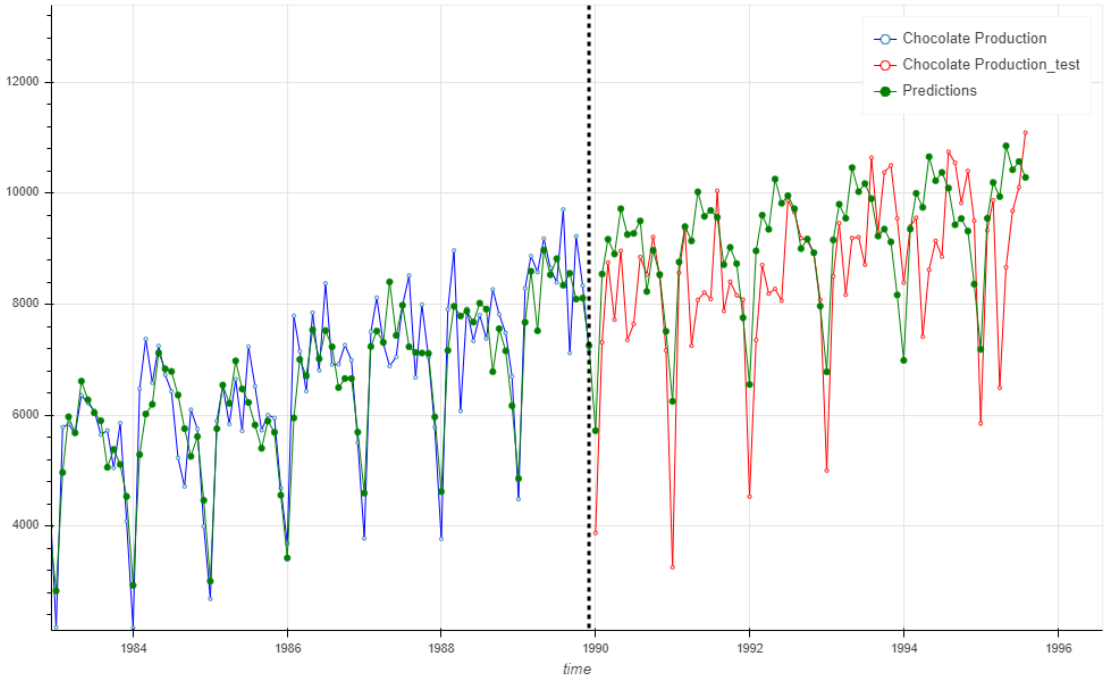


Figure A.8: Chocolate production : SARIMA(2,0,1)(2,0,1)12, Ho=0.15

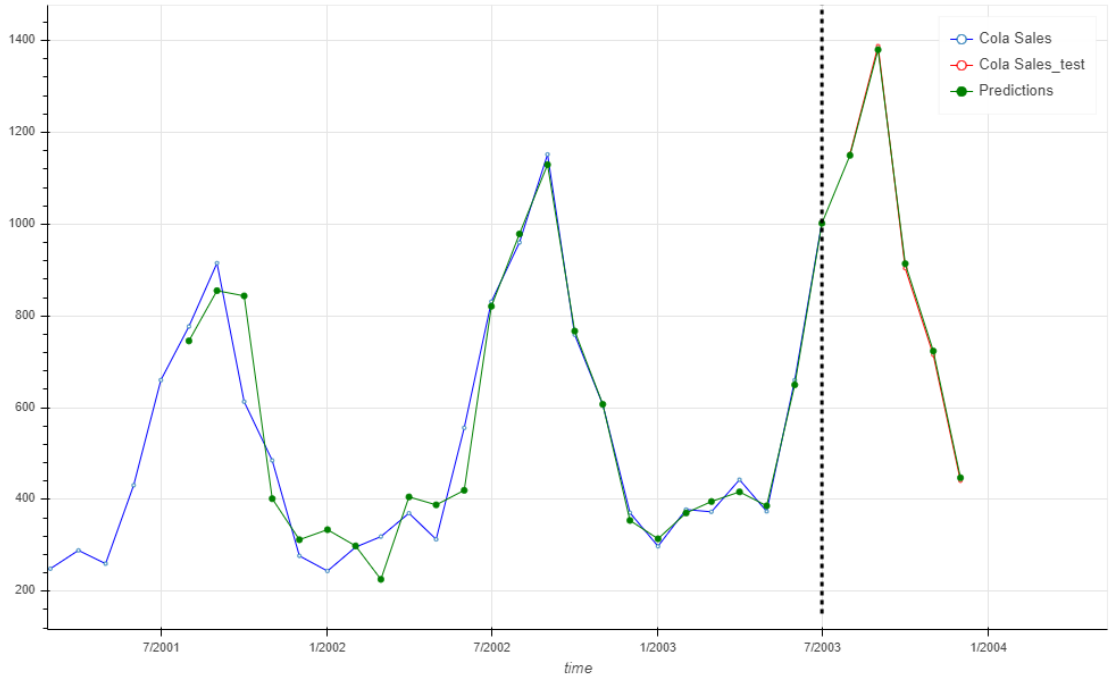


Figure A.9: Cola Sales : SARIMA(7,0,1)(5,0,1)12, Ho=0.15

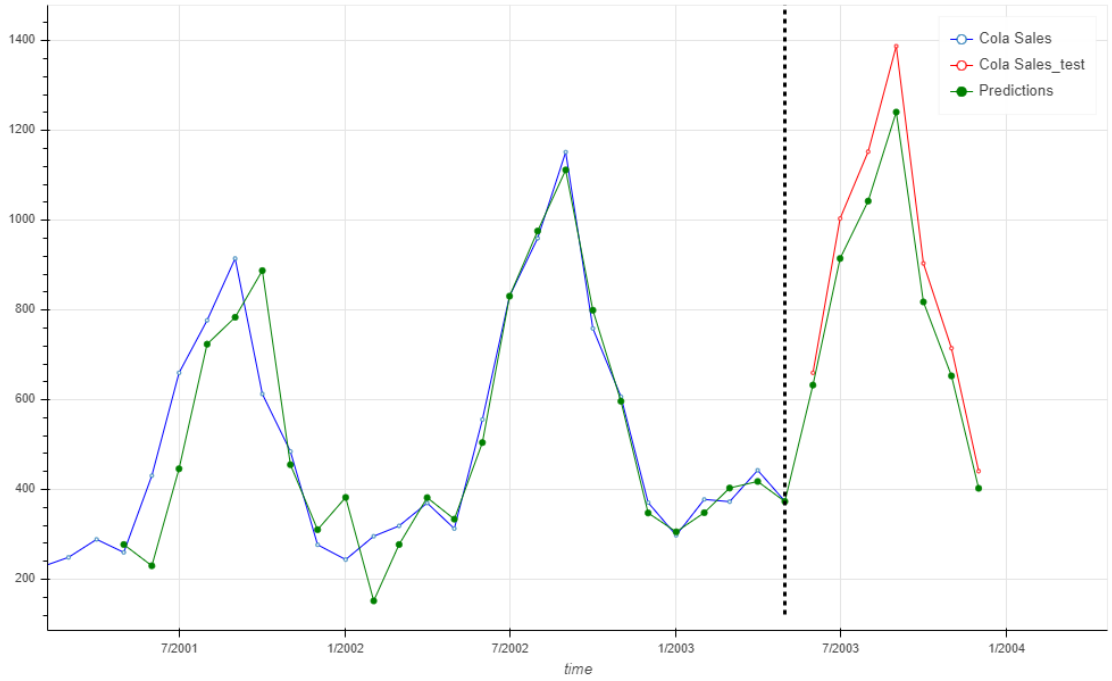


Figure A.10: Cola Sales : SARIMA(4,0,1)(4,0,1)12, Ho=0.20

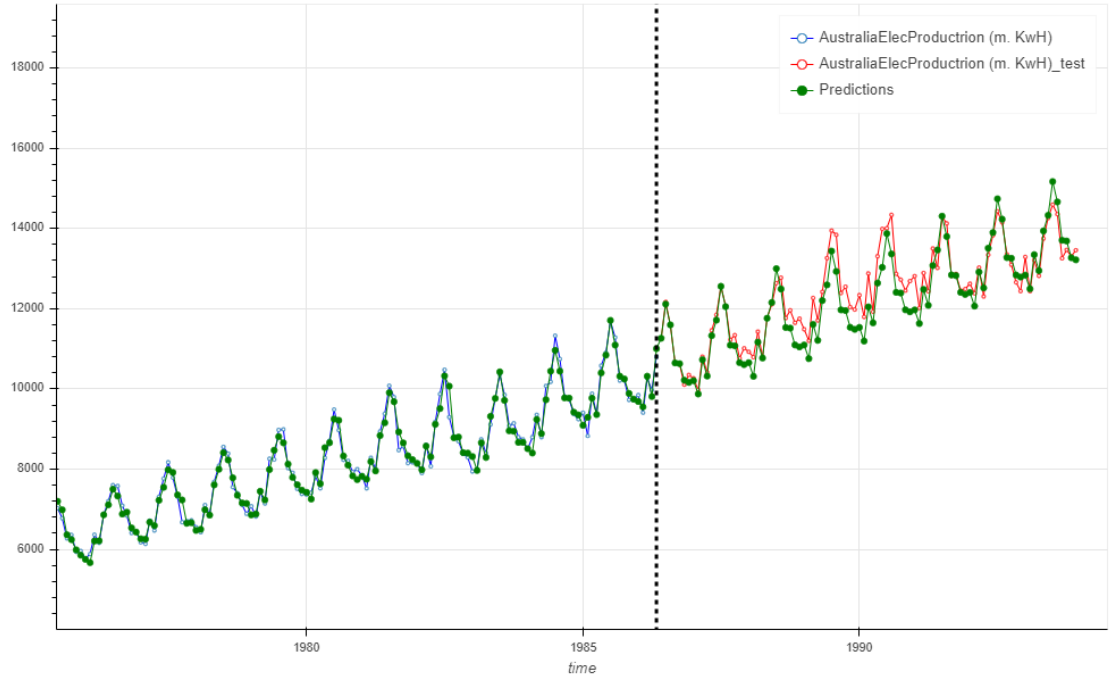


Figure A.11: Electricity production in Australia : SARIMA(2,0,1)(2,0,1)12, Ho=0.20

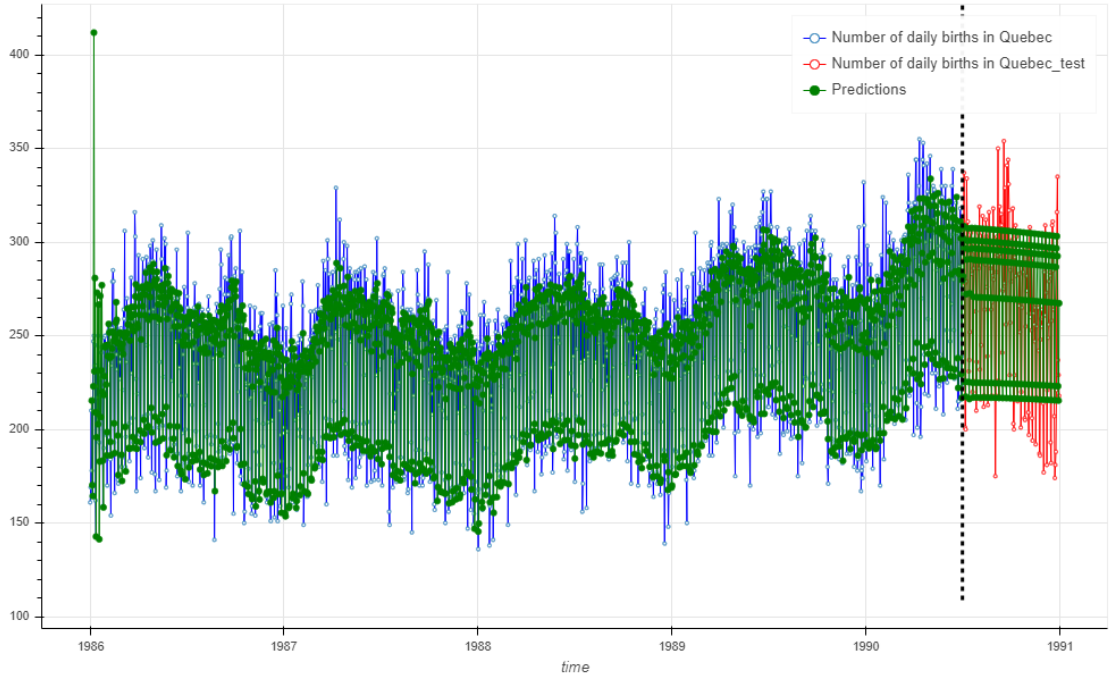


Figure A.12: Number of daily births is Quebec : SARIMA(3,0,1)(3,0,1)7, Ho=0.10

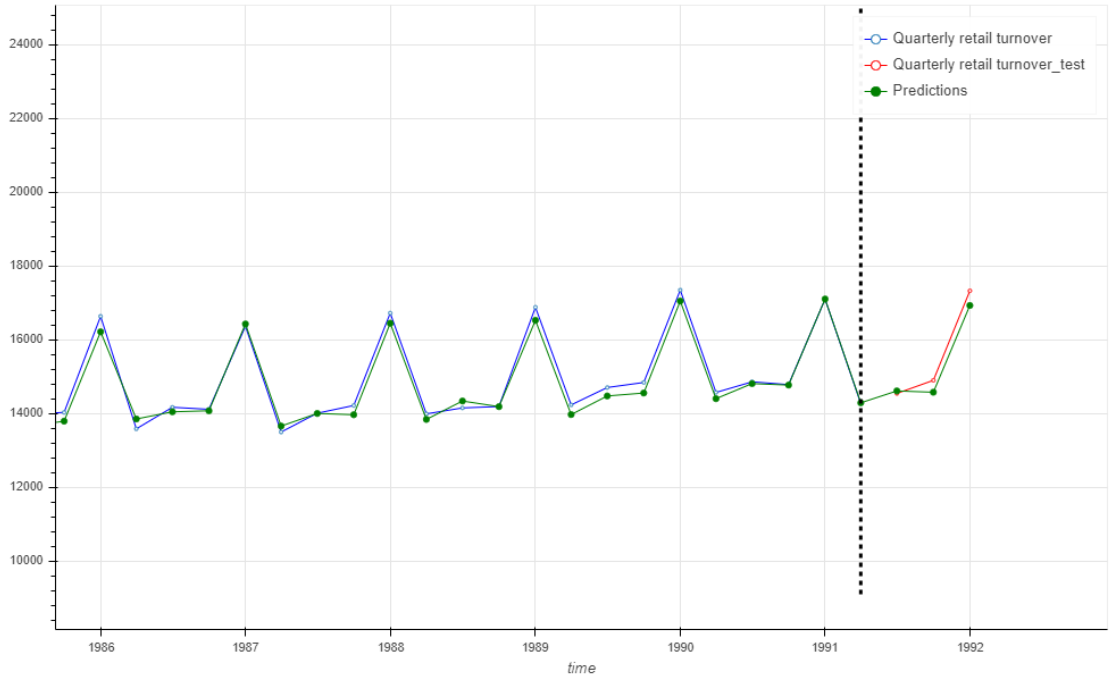


Figure A.13: Quarterly retail turnover : SARIMA(2,0,1)(2,0,1)4, Ho=0.10

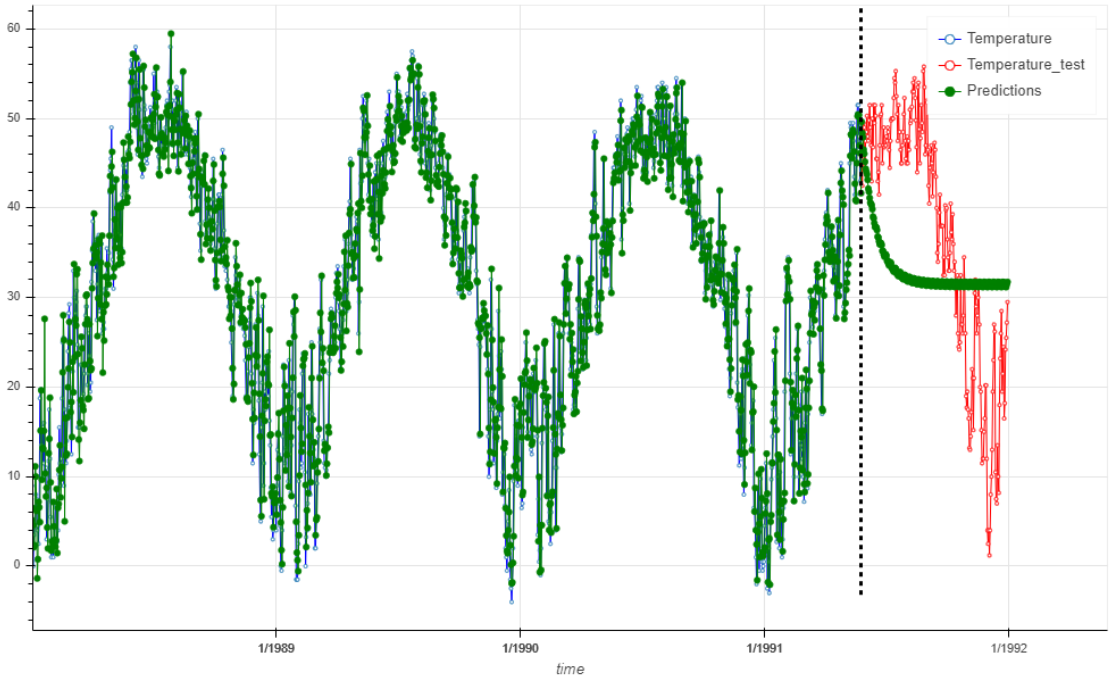


Figure A.14: Temperature : SARIMA(2,0,1)(2,1,1)7, Ho=0.15

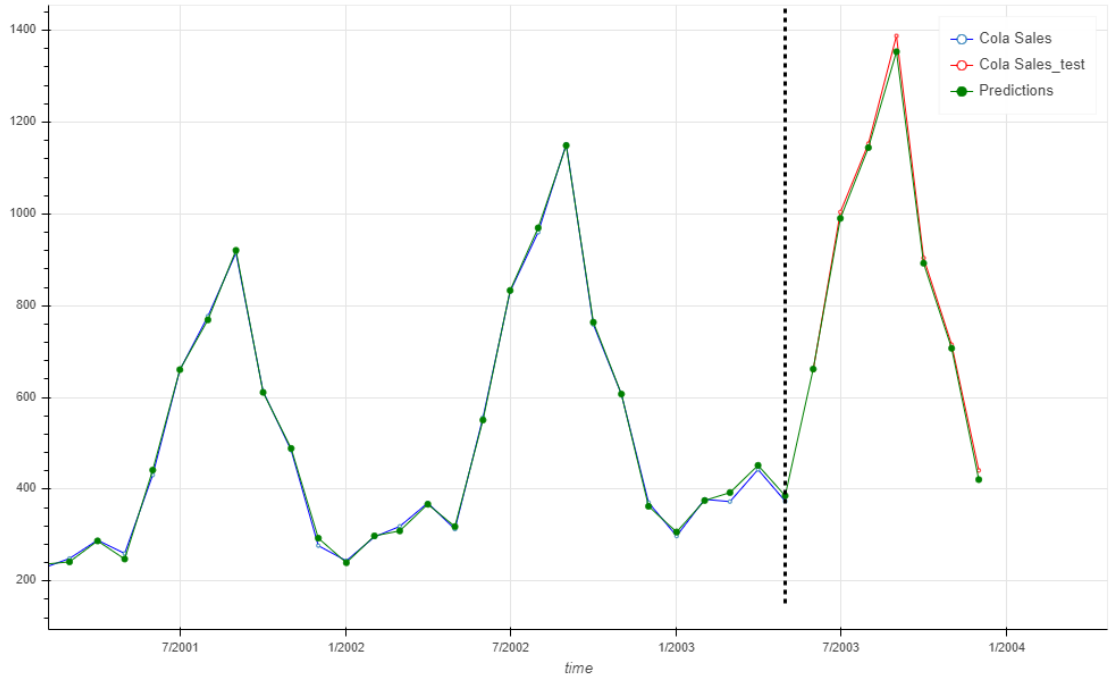


Figure A.15: Cola Sales : TETS, Ho=0.20

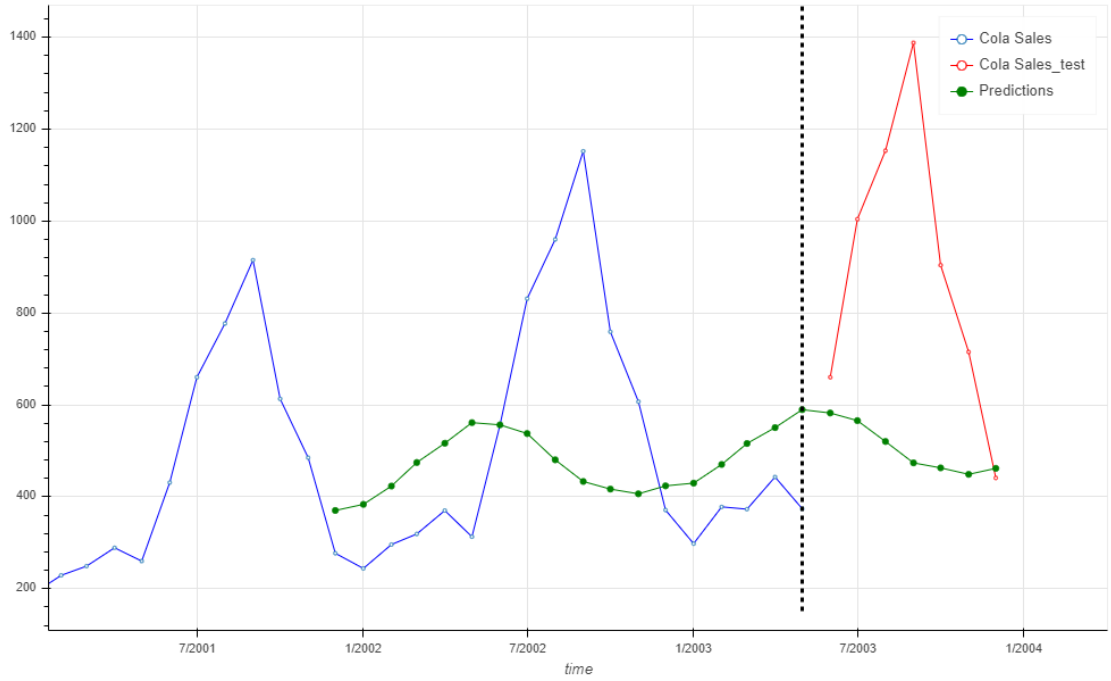


Figure A.16: Cola Sales : LSTM, Ho=0.20

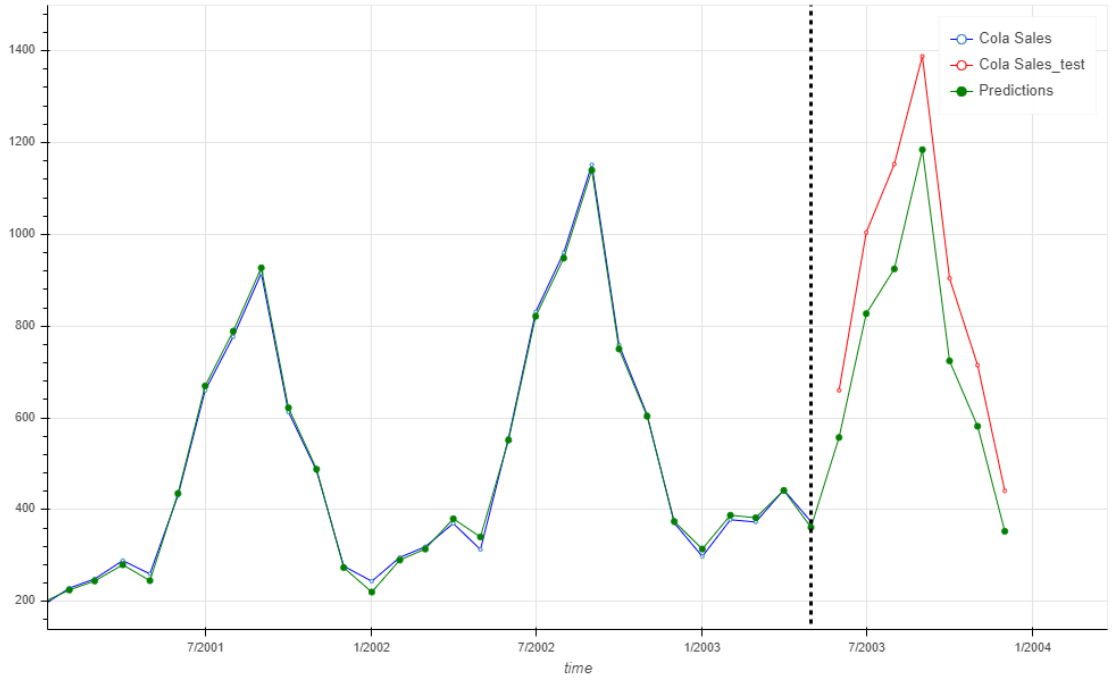


Figure A.17: Cola Sales : Prophet, Ho=0.20

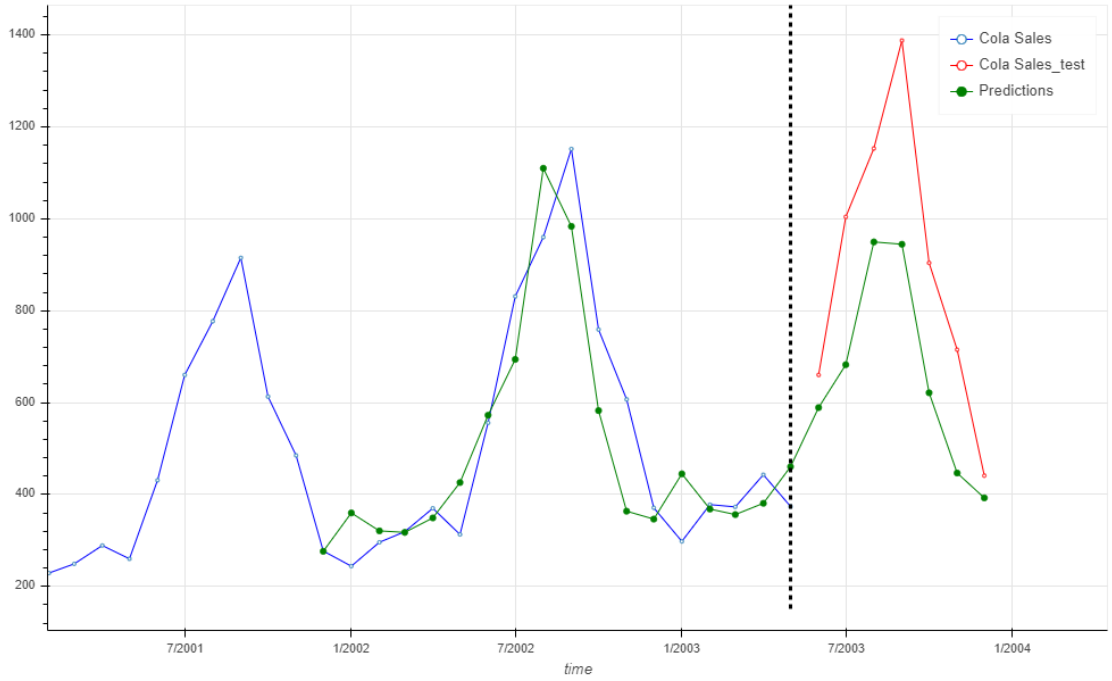


Figure A.18: Cola Sales : GRU, Ho=0.20

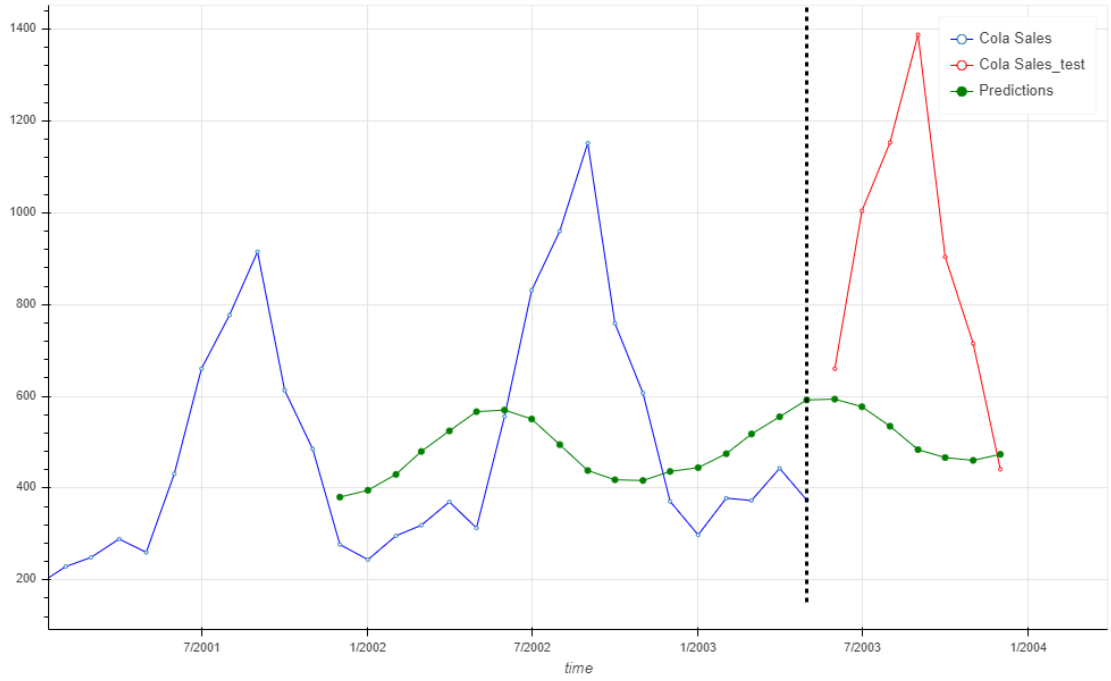


Figure A.19: Cola Sales : sGRU, Ho=0.20

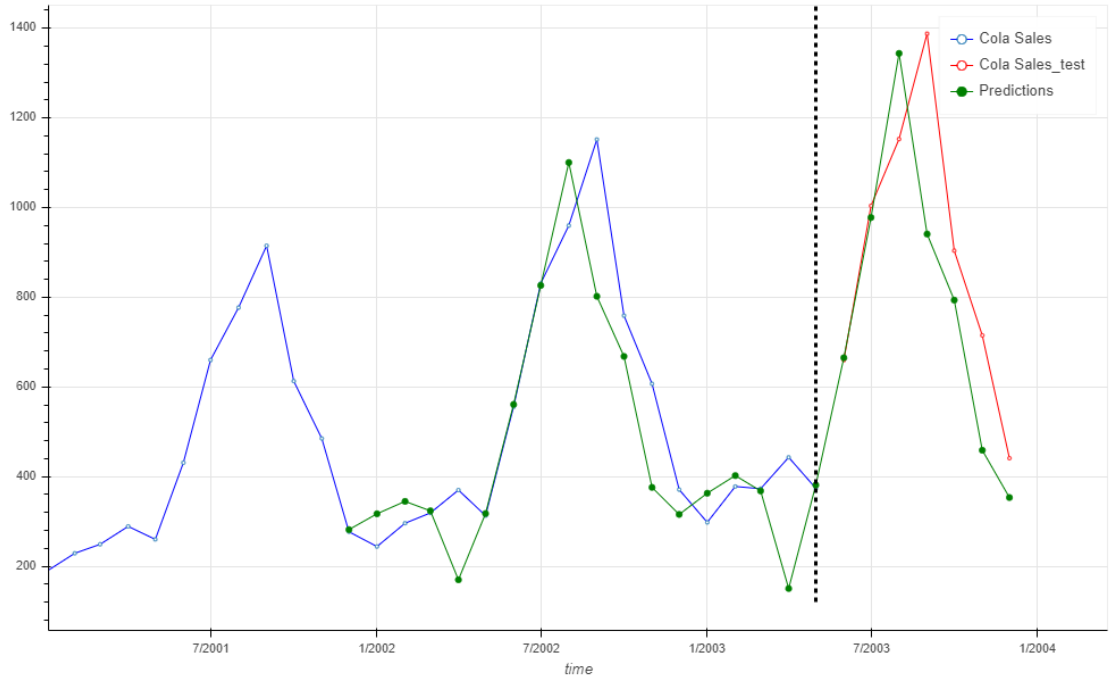


Figure A.20: Cola Sales : SVR, Ho=0.20

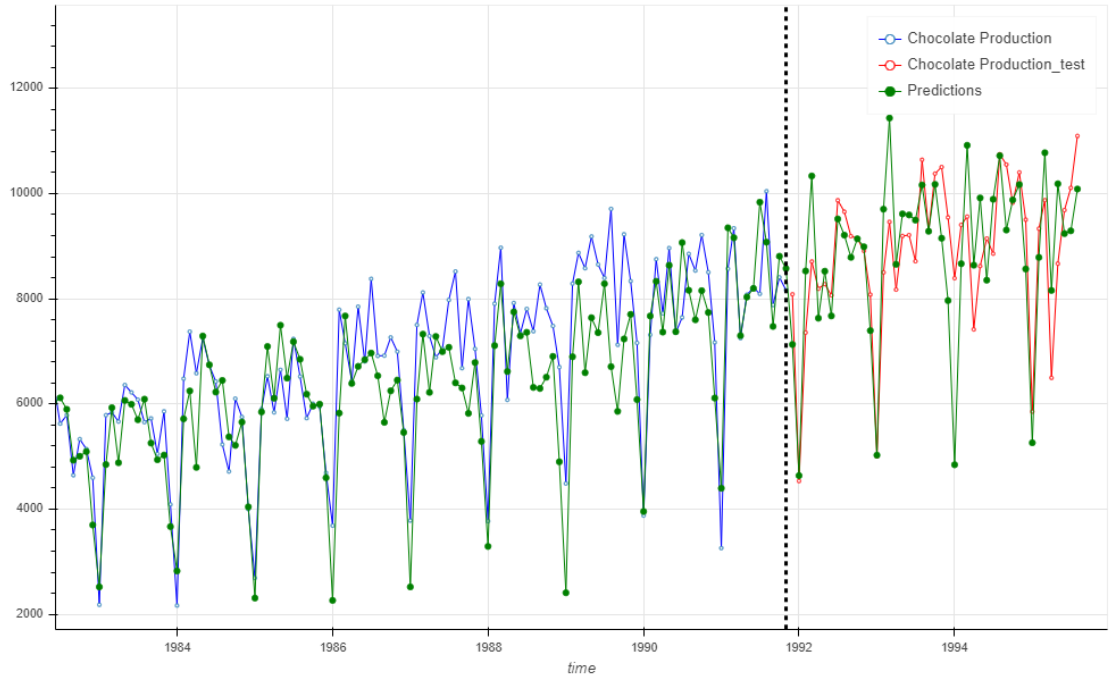


Figure A.21: Chocolate : SVR, Ho=0.10

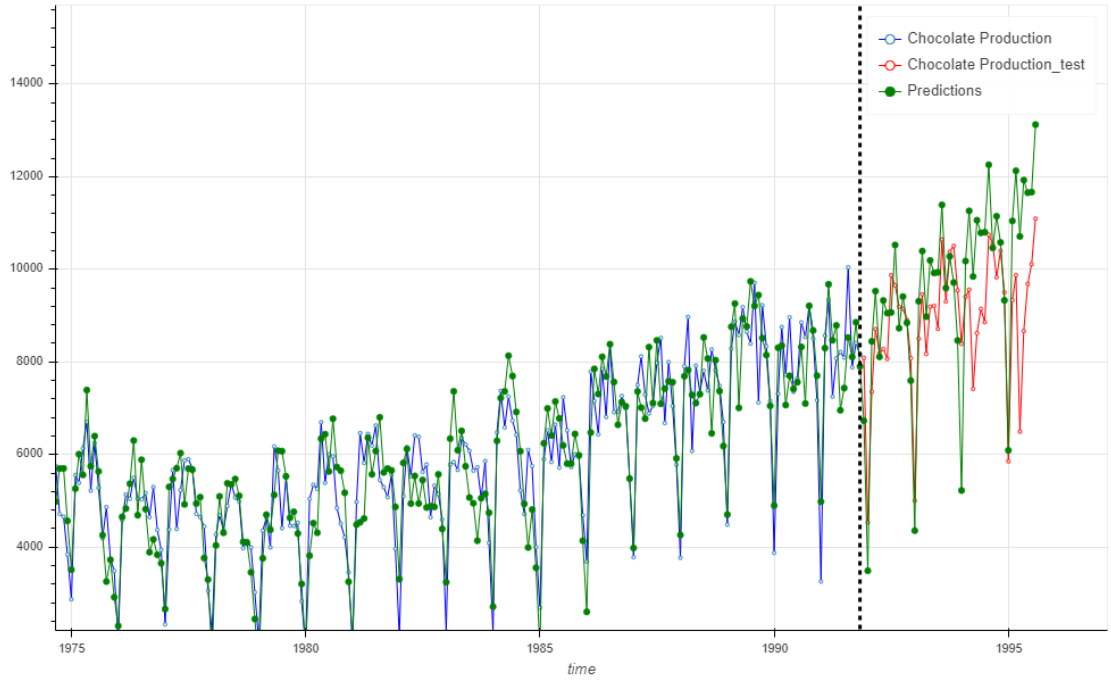


Figure A.22: Chocolate : TETS, Ho=0.10

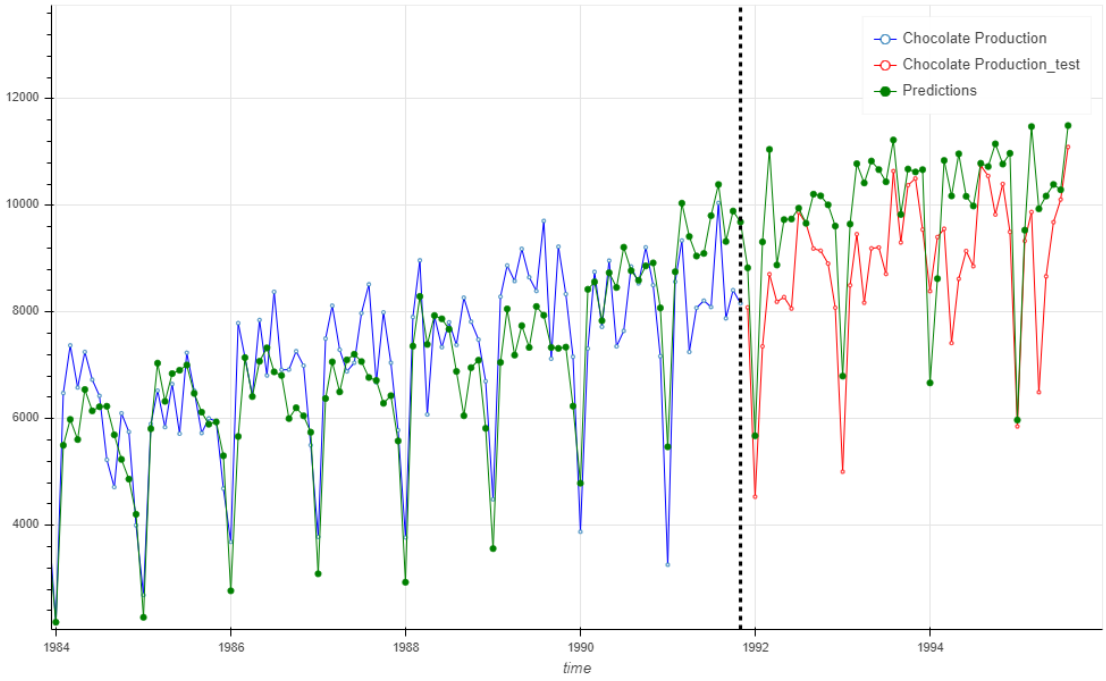


Figure A.23: Chocolate : LSTM, Ho=0.10

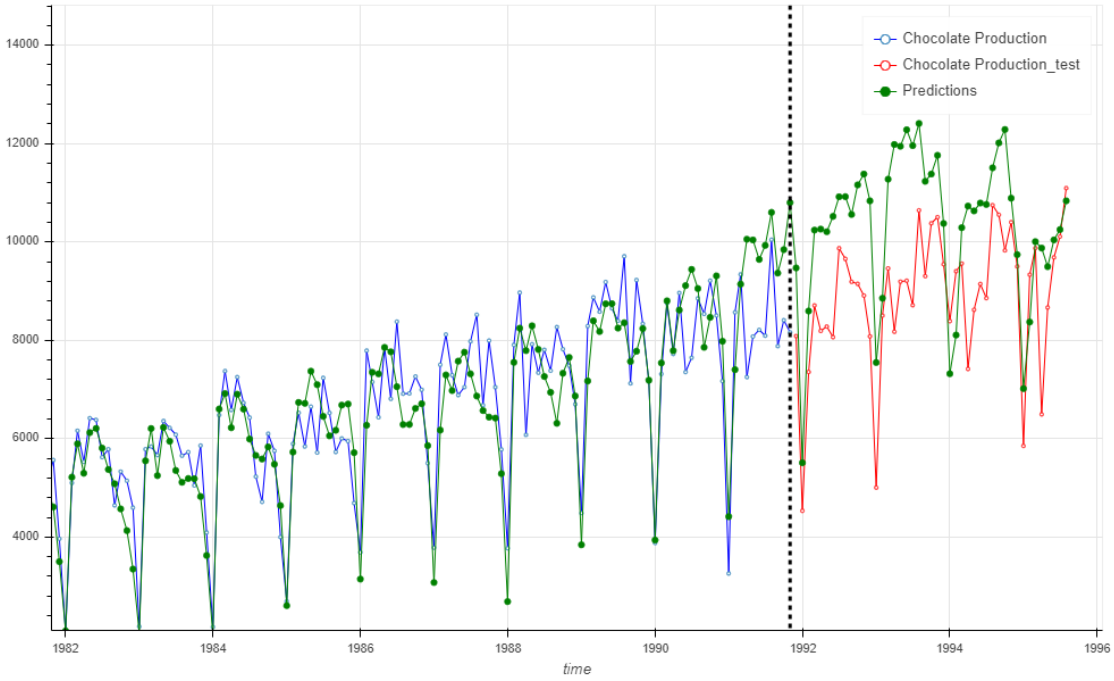


Figure A.24: Chocolate : GRU, Ho=0.10

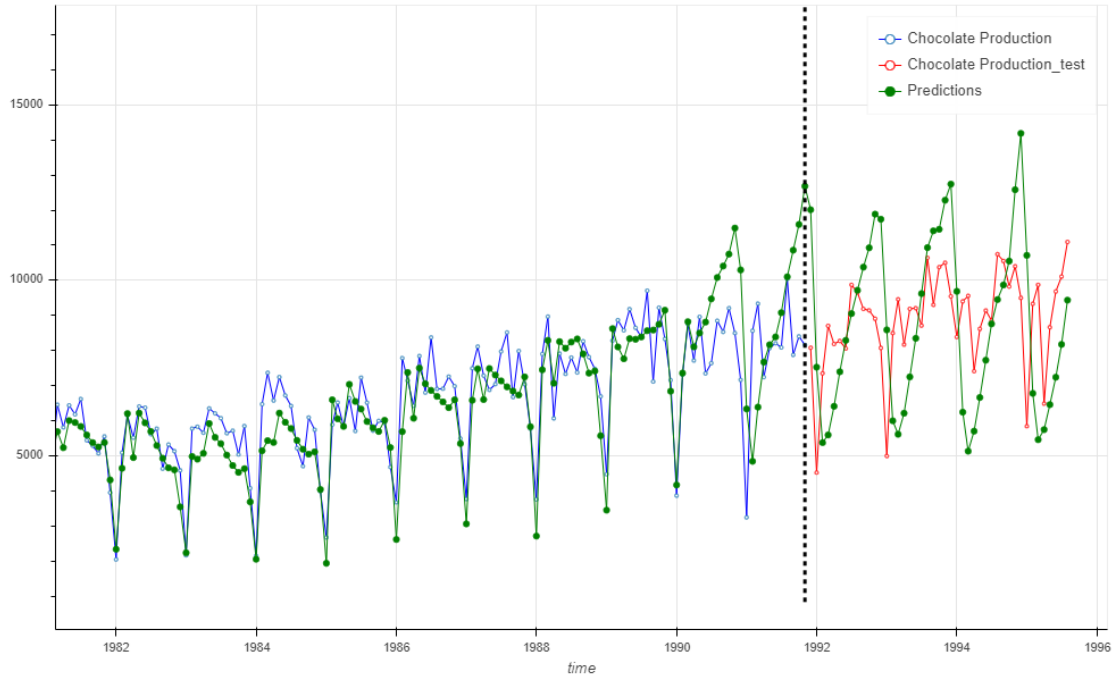


Figure A.25: Chocolate : sLSTM, Ho=0.10

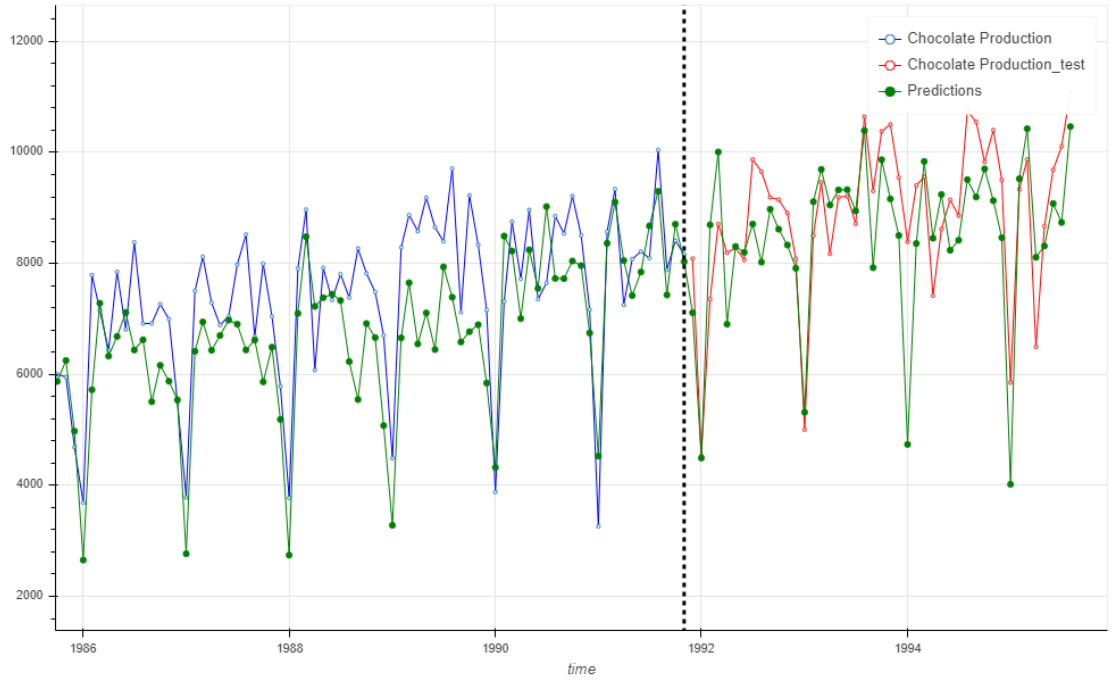


Figure A.26: Chocolate : sGRU, Ho=0.10

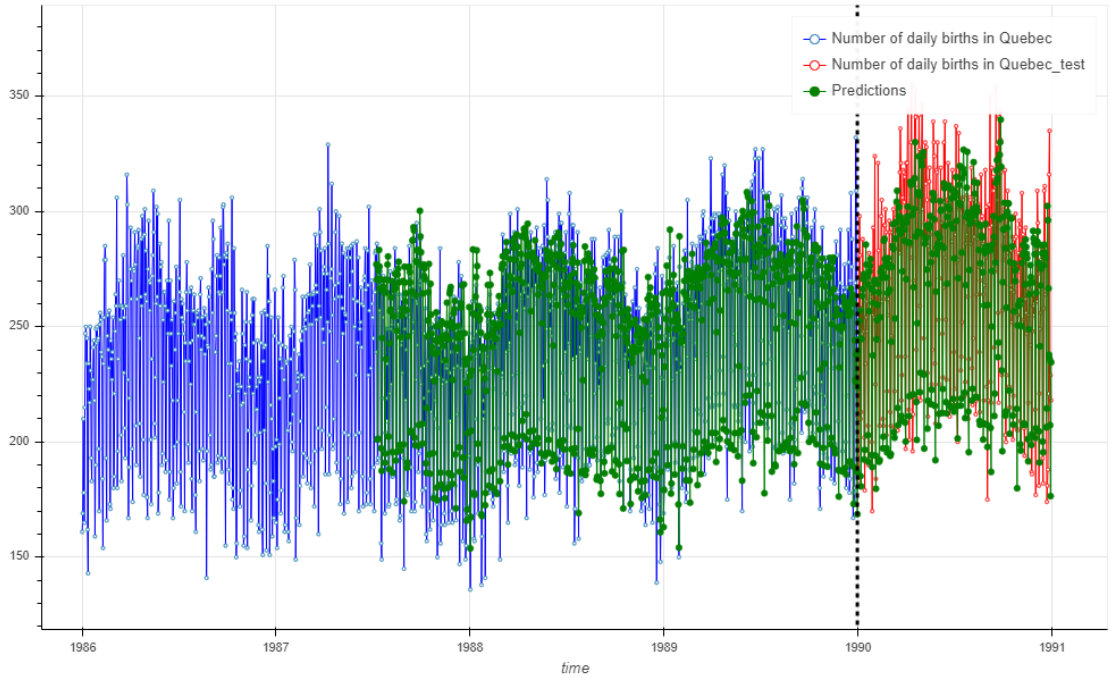


Figure A.27: Number of daily births in Quebec : sGRU, Ho=0.20

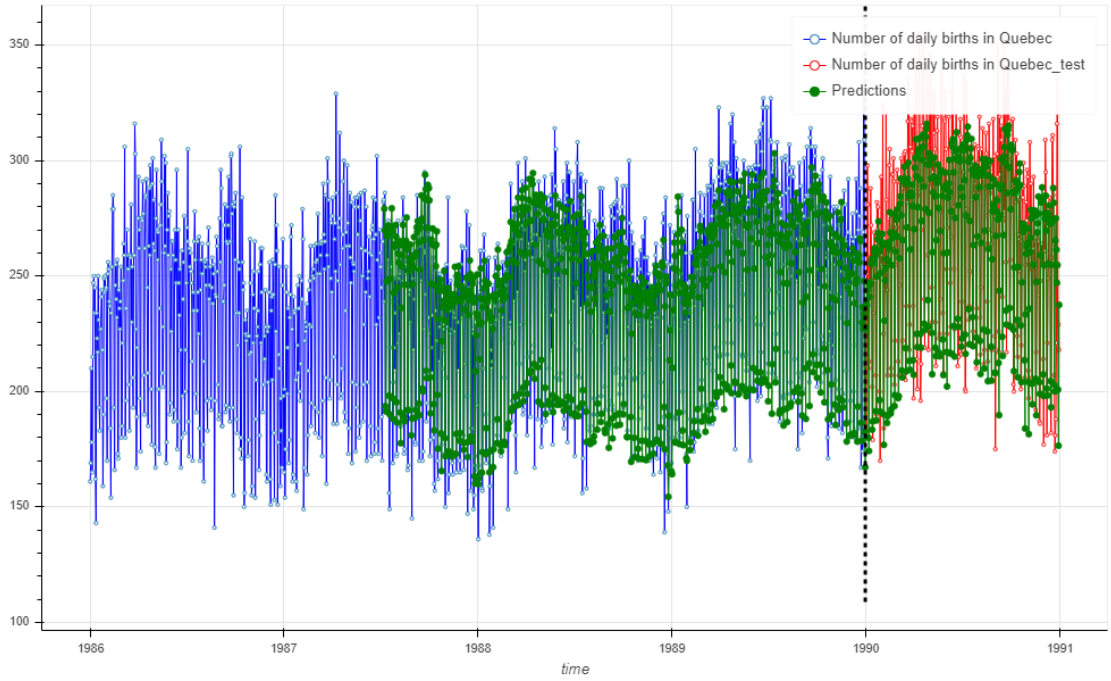


Figure A.28: Number of daily births in Quebec : LSTM, Ho=0.20

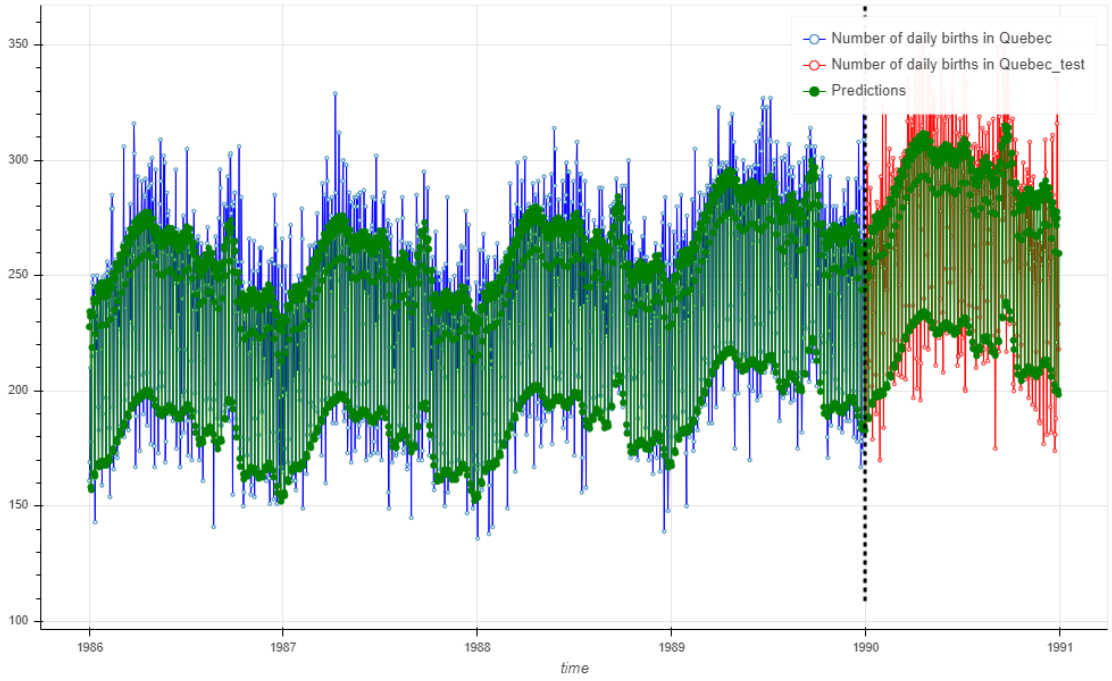


Figure A.29: Number of daily births in Quebec : Prophet, Ho=0.20

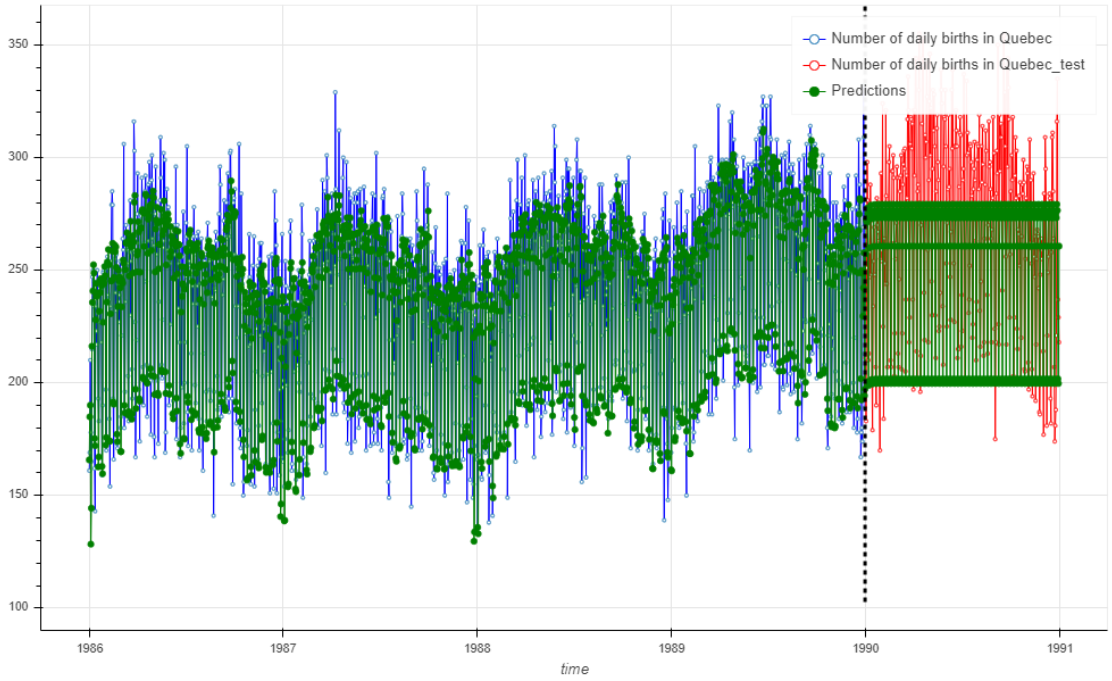


Figure A.30: Number of daily births in Quebec : TETS, Ho=0.20

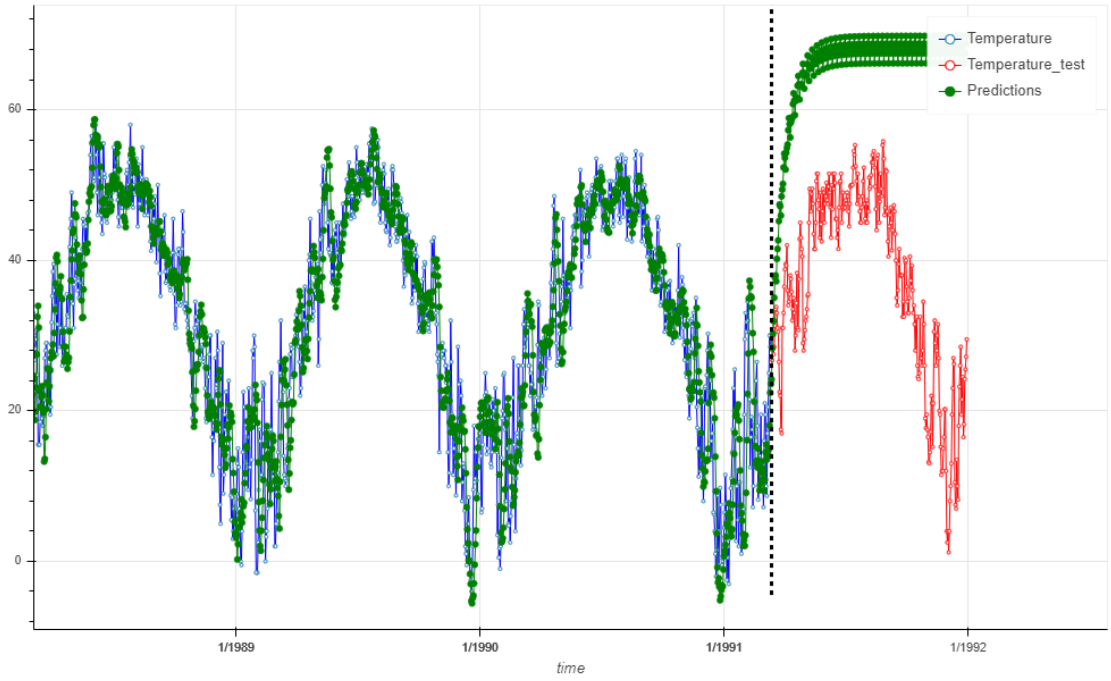


Figure A.31: Temperature : TETS, Ho=0.20

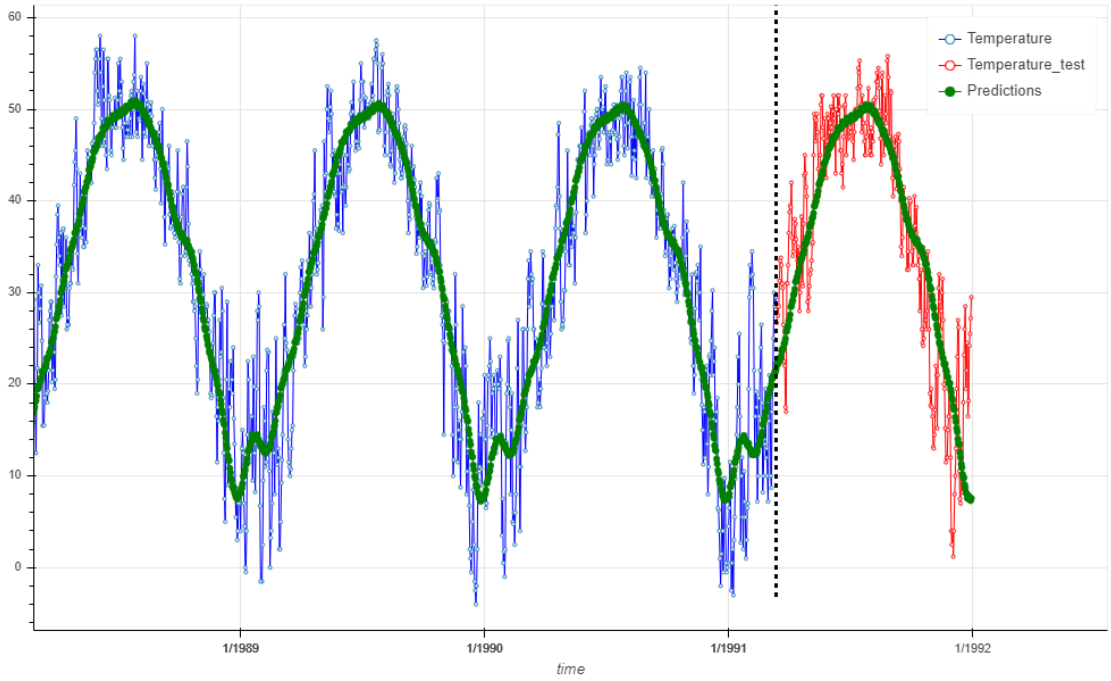


Figure A.32: Temperature : Prophet, Ho=0.20

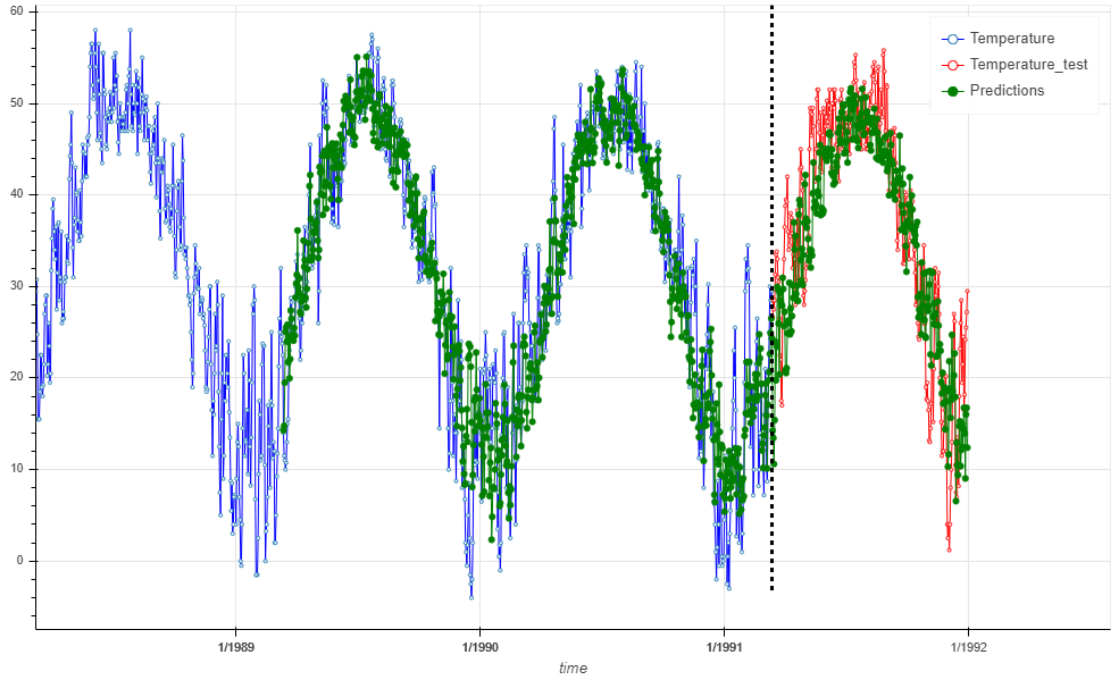


Figure A.33: Temperature : SVR, Ho=0.20

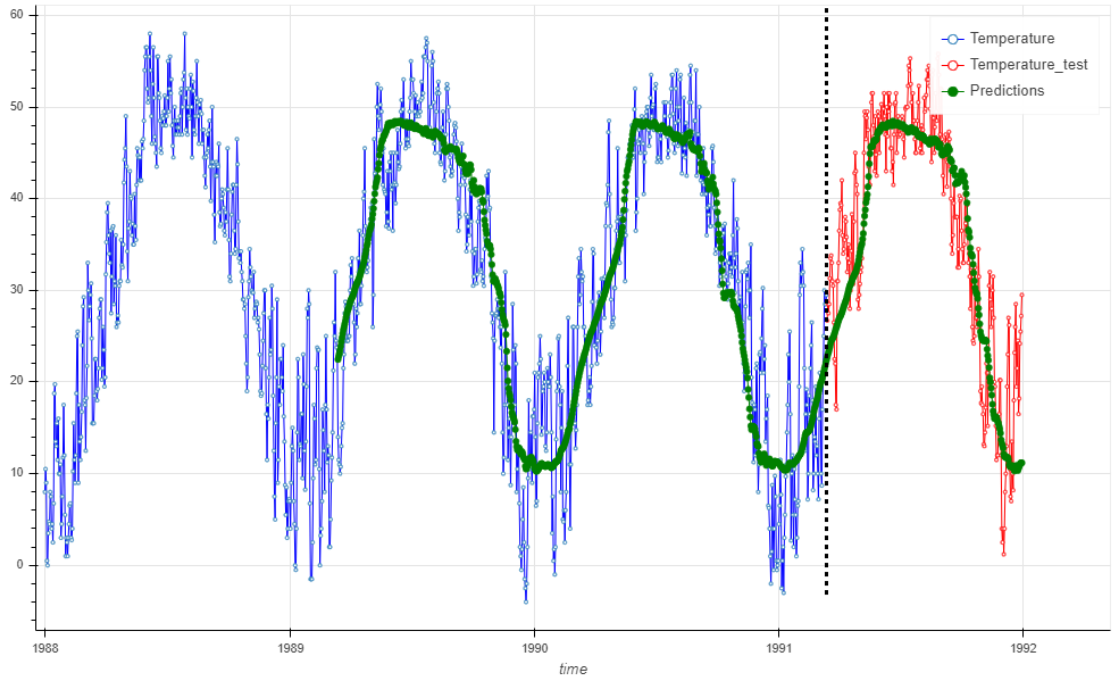


Figure A.34: Temperature : LSTM, Ho=0.20

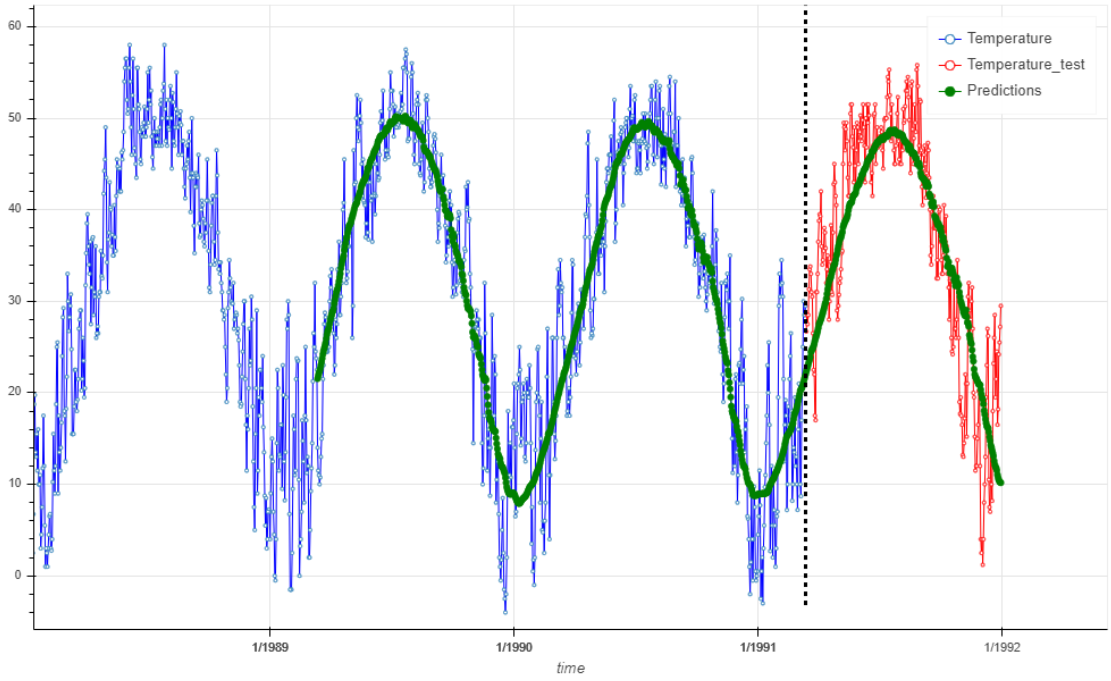


Figure A.35: Temperature : GRU, Ho=0.20

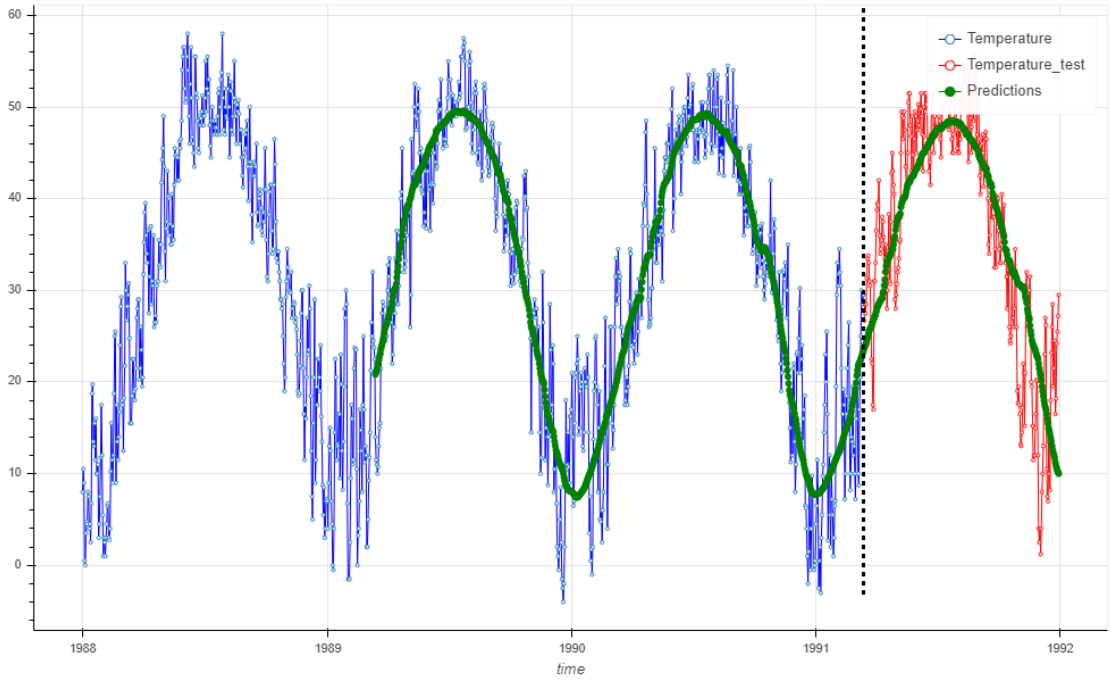


Figure A.36: Temperature : sGRU, Ho=0.20

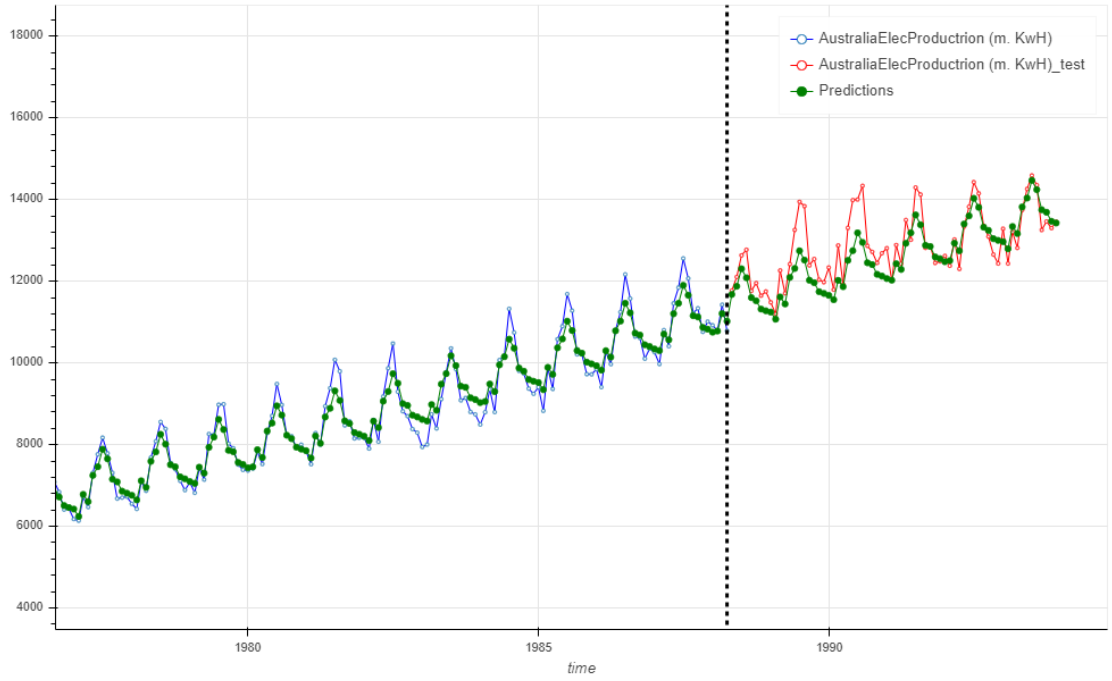


Figure A.37: Electricity Production in Australia : Prophet, Ho=0.15

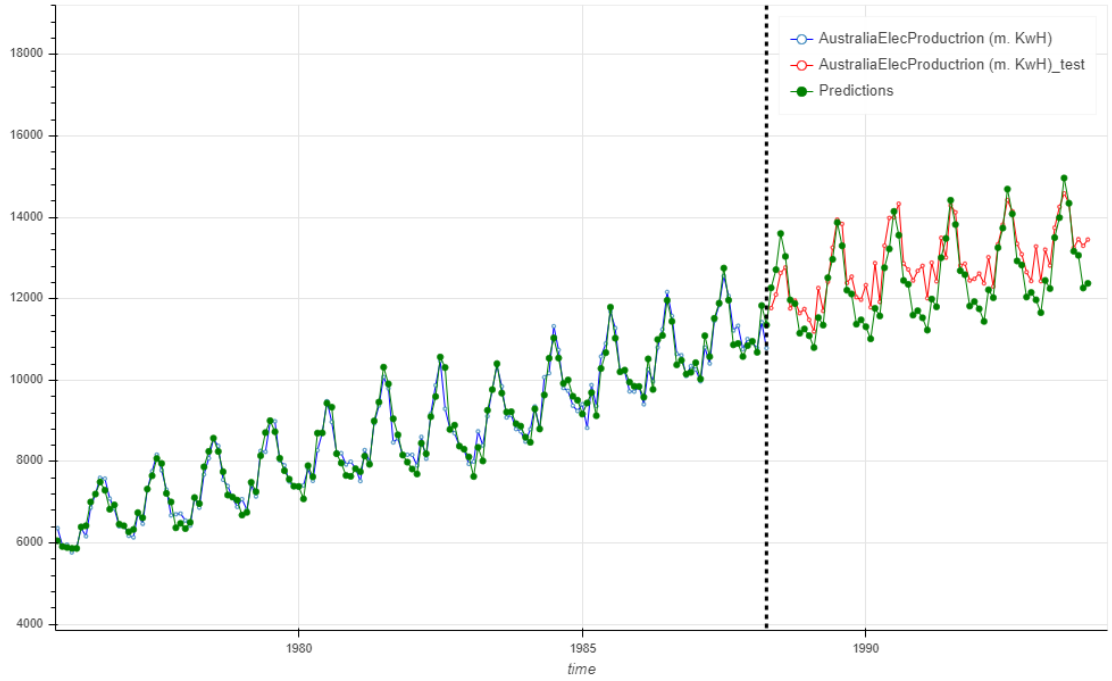


Figure A.38: Electricity Production in Australia : TETS, Ho=0.15

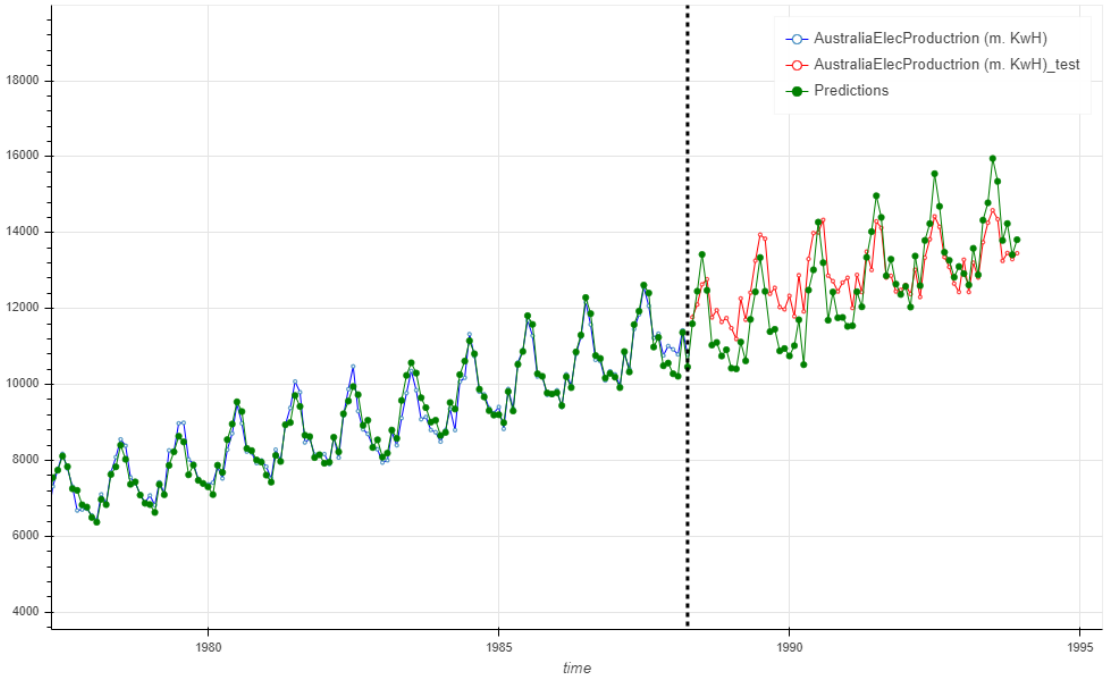


Figure A.39: Electricity Production in Australia : SVR, Ho=0.15

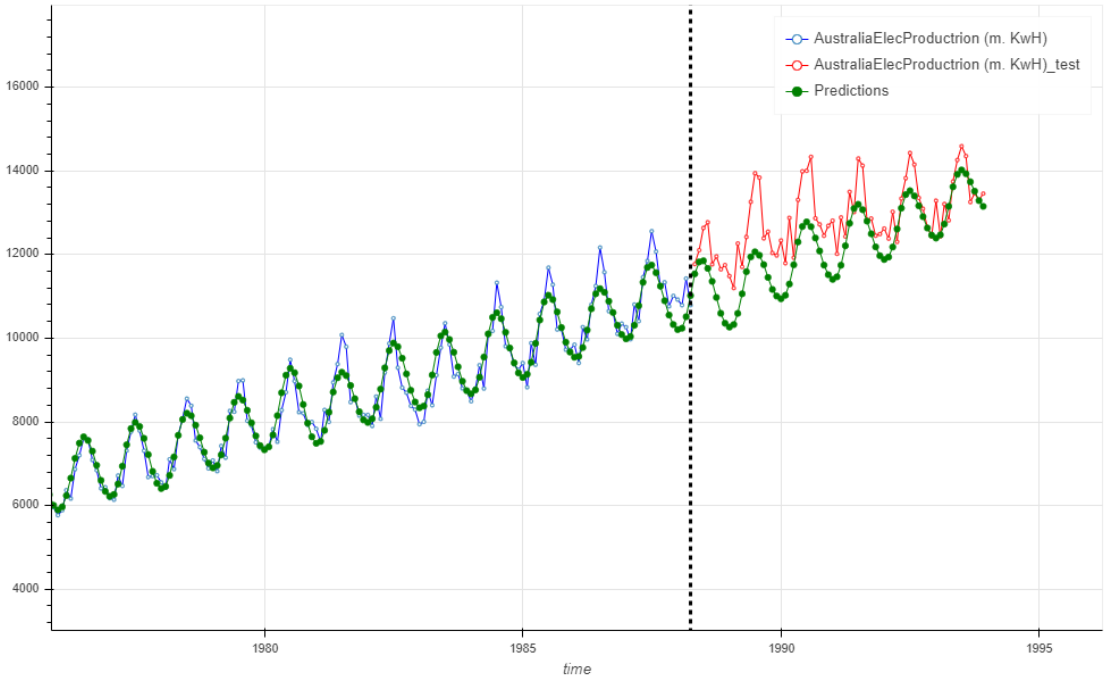


Figure A.40: Electricity Production in Australia : LSTM, Ho=0.15

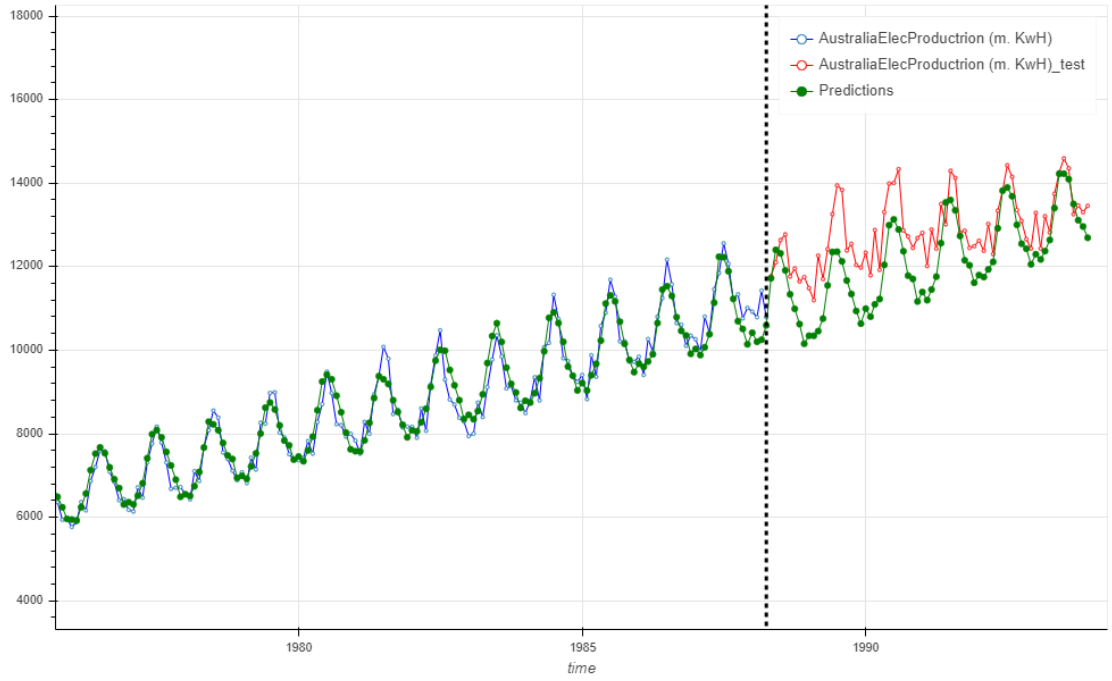


Figure A.41: Electricity Production in Australia : GRU, Ho=0.15

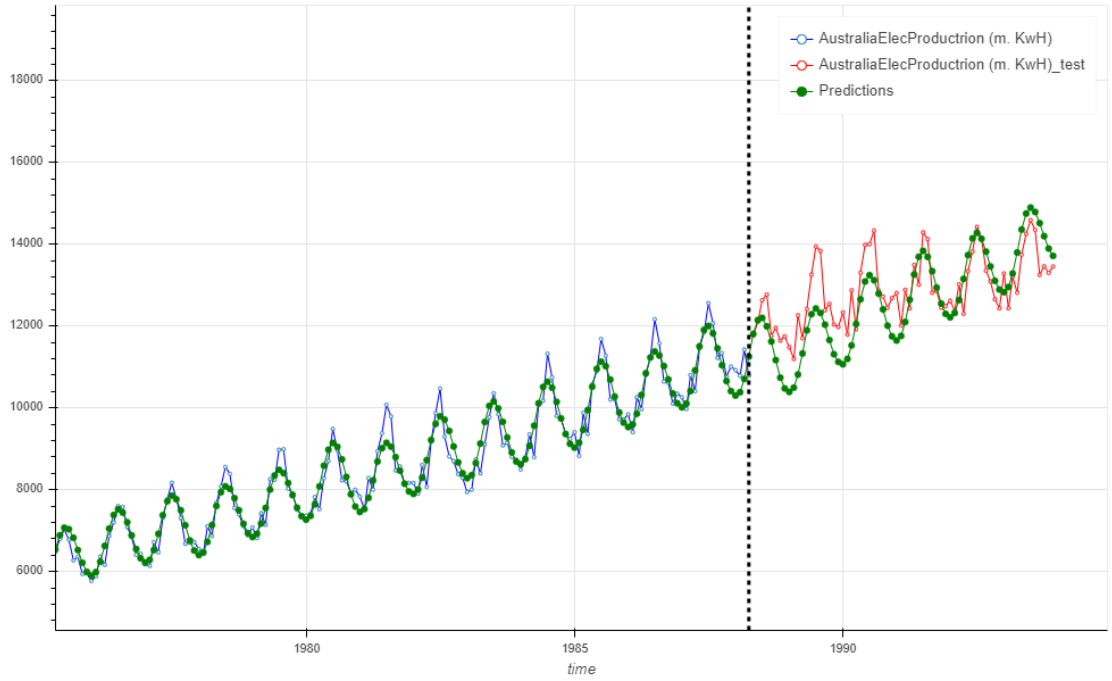


Figure A.42: Electricity Production in Australia : sLSTM, Ho=0.15

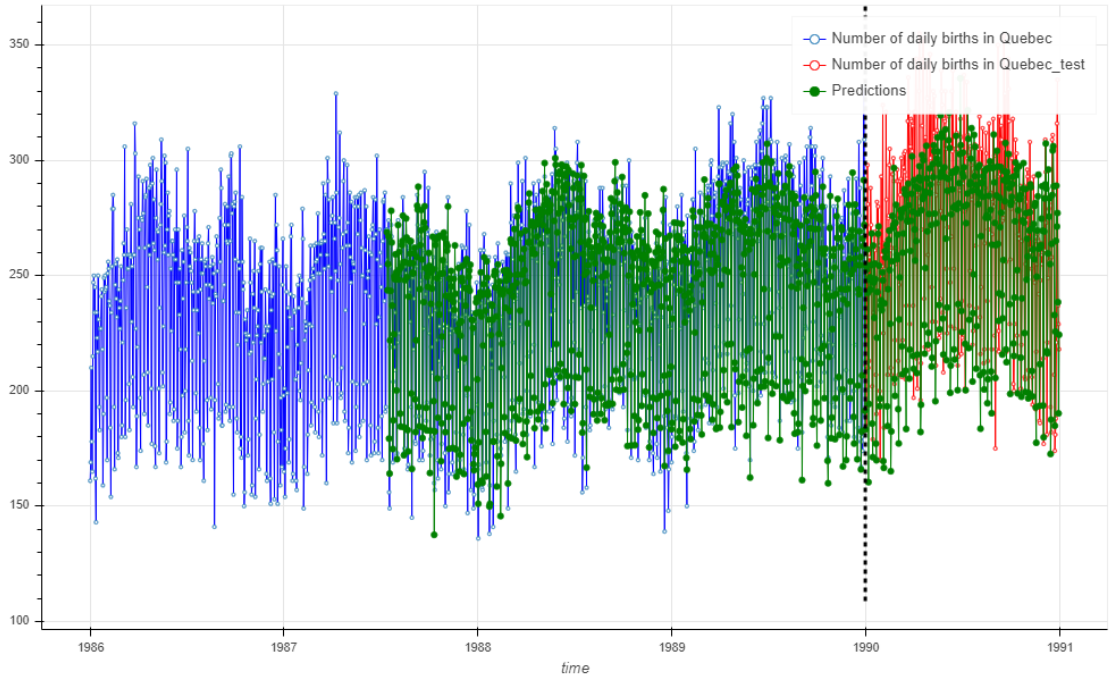


Figure A.43: Number of daily births in Quebec : Wavelets + RNN, Ho=0.20

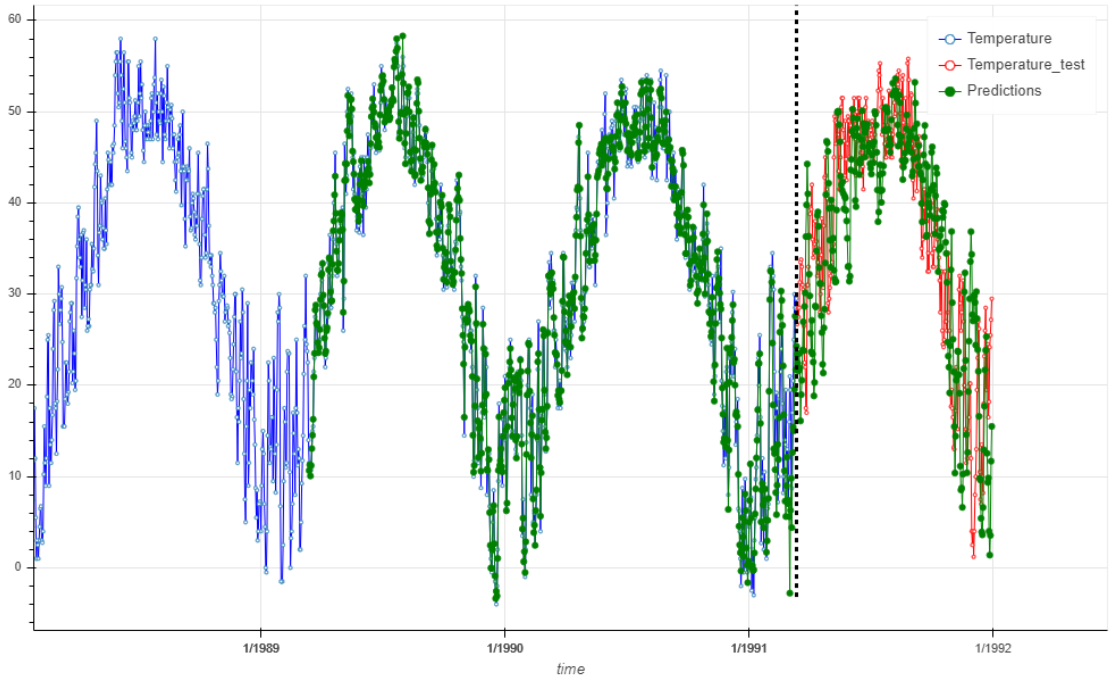


Figure A.44: Temperature : Wavelets + RNN, Ho=0.20