SAPIENZA
UNIVERSITÀ DI ROMA

Short-Rate Forecasting: Unveiling Opportunities to Beat the Random Walk – An Empirical Exploration through Mathematical Finance and Machine Learning Models

Dipartimento di Metodi e Modelli per l'Economia, il Territorio e la Finanza

Corso di Laurea Magistrale in FINASS, curriculum Financial Risk and Data Analysis

Candidate

Yuri Antonelli

ID number 1860118

Thesis Advisor

Prof. Sergio Bianchi

Academic Year 2022/2023

Thesis defended on 19/01/2024
in front of a Board of Examiners composed by:

Prof. Bianchi Sergio (chairman)
Prof. De Angelis Paolo
Prof. Bruno Maria Giuseppina
Prof. San Mauro Cesare
Prof. Sciandrone Marco
Prof. Patrì Stefano
Prof. Scarpitti Mariarita

Short-Rate Forecasting: Unveiling Opportunities to Beat the Random Walk – An Empirical Exploration through Mathematical Finance and Machine Learning Models

Master's thesis. Sapienza – University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: yuriantonelli1999@gmail.com

Abstract

This thesis evaluates interest rate forecasting models drawn from mathematical finance (Vasicek, CIR) and machine learning (Bagging, LSTM). 1-day forecasts reveal intermittent superiority of CIR and Vasicek over the Random Walk, while Bagging and LSTM consistently underperform. For longer forecasting horizons (1 week, 1 month, 3 months), CIR outperforms the Random Walk except for the 3-month horizon.

F-score analysis suggests erratic performance for short-term forecasts but discernible patterns for longer-term ones. A potential avenue for future research involves exploring the connection between empirical methods and the Efficient Market Hypothesis to investigate potential improvements in forecasting by taking market efficiency into account.

Contents

Chapter 1

Introduction

There exists a substantial body of literature dedicated to the field of interest rate forecasting, as the ability to make accurate predictions regarding interest rates carries considerable significance for various economic agents:

- From the view of financial market practitioners, gaining insights into the future trajectory of interest rates holds the potential to perform better investment decisions
- In the realm of risk management, it assumes paramount importance enabling proactive measures to mitigate market-related risks
- Government agencies overseeing bond issuance are highly concerned about predicting how interest rates will change, as they need to assess and plan for the future costs of borrowing money

This challenge can be approached from multiple perspectives, each with its unique characteristics. Indeed, one has the flexibility to forecast various types of interest rates, considering different time horizons and employing diverse methodologies.
These methods range from pure statistical approaches, which for example utilize commonly accepted time series models, to more mathematical techniques, involving the calibration of models specifically designed to explain the dynamics of interest rates.

One potential approach involves the endeavor to fit a model to the entire yield curve and then forecast future changes in the interest rate structure.
An interesting empirical contribute has been carried out by Molenaars et al. (2013), where the whole US Treasuries yield curve has been analyzed, with maturity spanning from 3 to 120 months. The authors use a dynamic Nelson-Siegel model, designed to characterize the shape of the yield curve, to compare its performance with the ones of a simple Random Walk (RW) and an AR(1) model. Their findings reveal that the Random Walk model consistently outperforms both alternative models across most of the time series, regardless of the forecasting horizon considered.
Another valuable contribution to this field has been made by den Butter and Jansen (2013), who concentrate on forecasting long-term interest rate while employing the RW as a benchmark. Their research involves a comprehensive evaluation of diverse models and various combinations of them. Of notable interest is their focus on forecasts that incorporate expert opinions alongside model-based predictions. The outcomes of their study indicate that combinations of structural macro models and expert opinions tend to surpass the performance of the RW model when predicting

long-term interest rates, specifically for a horizon of 12 months. However, these combined approaches exhibit subpar performance for short-term forecasts.
An impactful addition to the methodology has been introduced by Orlando et al. (2020), where the authors propose a partition of the data sample using Normal and non-central Chi-square distribution to improve the Vasicek and CIR models calibration. This permits to account for jumps, and the distributions are chosen given the conditional distributions of the two short-term models.

1.1 Proposed Methodology

In this study, my focus revolves around the analysis of *13 Weeks Treasury Bill Rates*, recorded daily.
This selection is motivated by the applicability of two models I intend to utilize, Vasicek and CIR, which have been deployed to describe short-rate dynamics in continuous time, and such a dataset can be considered a good proxy for them.
Below is a visual representation of the time series:

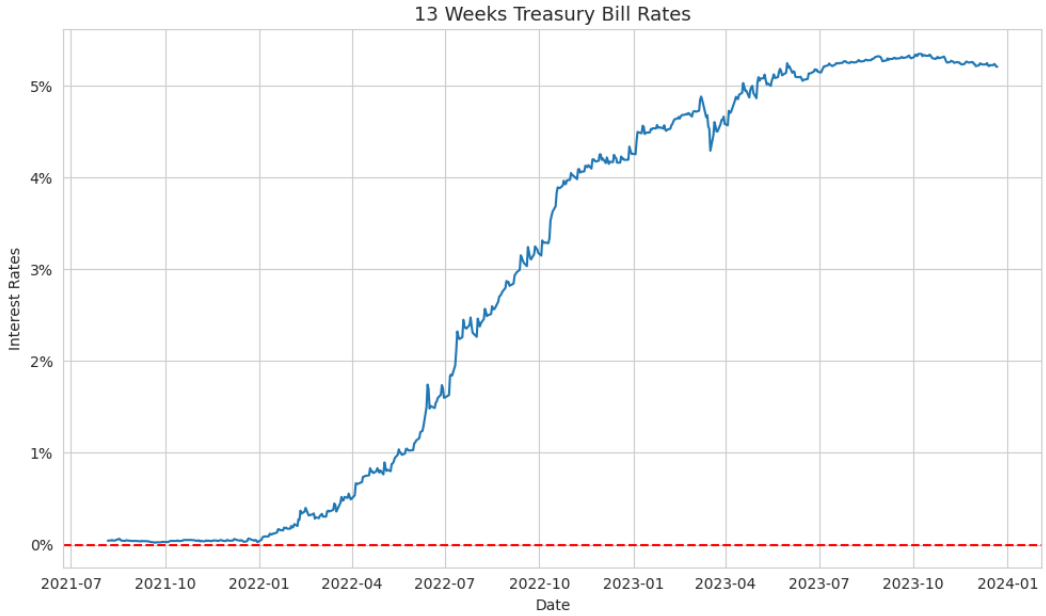


Figure 1.1. Source: https://finance.yahoo.com/quote/%5EIRX?p=%5EIRX

Two different classes of models are employed in this paper:

- The first one, drawn from the **Mathematical Finance** literature, comprises the Vasicek model (Vasicek (1977)) and the CIR model (Cox et al. (2005)). These are primarily designed for short-rate dynamics and find utility in both forecasting and pricing applications
- The second class of models belong to the **Machine Learning** category and encompasses two statistical models that can be used either for general applications (Bagging) or for time series forecasting (Long-Short-Term-Memory)

The calibration framework I utilize is a rolling window approach, commencing with an initial 3-months dataset.
However, the forecasting procedures diverge between the two categories of models. Specifically, while all models generate a 1-step ahead forecast, only the Vasicek and CIR models compute forecasts for 1/7/30/90 days ahead.

The rationale behind this disparity lies in the requirement that each forecast remains viable with real-time data, drawn from a window spanning just 90 days in the past. This feasibility is achievable for all models when restricted to 1-step ahead forecasts. However, for longer forecasting horizons, it's only viable for the Vasicek and CIR models, which solely rely on past observations for calibration. In contrast, the Random Forest and LSTM models would necessitate future data to be trained, constrained to the most recent 90 observations exclusively.

Below, I provide a pseudo-code of the general algorithm:

Algorithm 1: Pseudo-code for Rolling Window

```
// Initialize variables and vectors to store predictions
y = data at time t
X = data at time t-1

ts_length = length(y)
n_obs = 90
max_time_step = 90

pred1_vector, pred7_vector, pred30_vector, pred90_vector = [ ], [ ], [ ], [ ]

// Rolling Window for loop
for n to (ts_length - n_obs) do
    X_train = X[n : n + n_obs]
    y_train = y[n : n + n_obs]
    prediction_index = y_train.index[-1]

    // Check if the max prediction cannot be backtested
    if (prediction_index + max_time_step) > y.index[-1] then
        break // Break the loop
    end
    // Calibrate your model and compute predictions
    // This example is for the Random Walk so no calibration
       required
    r0 = y_train[-1]
    pred1 = r0
    pred7 = r0
    pred30 = r0
    pred90 = r0

    // Store the predictions
    pred1_vector[n] = pred1
    pred7_vector[n] = pred7
    pred30_vector[n] = pred30
    pred90_vector[n] = pred90
end
```

1.2 Random Walk as a Benchmark

To evaluate the performance of the different models, I employ Random Walk (RW) forecasts as a benchmark.
This approach proves particularly valuable due to the prominent role the RW model plays in the financial literature. The RW model is closely linked to the Efficient Market Hypothesis, and empirical results that favor this straightforward model can be interpreted as supporting the notion of market efficiency in a specific context.
From a mathematical point of view:

$$i_{t+\delta t} = i_t + \epsilon \tag{1.1}$$

where:

- $i_{t+\delta t}$ is the interest rate at a certain period in the future
- i_t is the interest rate at time t
- ϵ is the disturbance term, having $\mathbb{E}(\epsilon) = 0$

This leads to:

$$\mathbb{E}(i_{t+\delta t}) = i_t \tag{1.2}$$

Namely, the best forecast at time $t+1$ is the value at time t.
In essence, this means that there is no valuable information in past data that can be exploited to enhance our forecasts. All the stochastic processes having this property are also called Martingale.

In order to assess the performance of our models, I use a similar approach to the one used in the already cited work Molenaars et al. (2013).
Initially, I compute the mean-squared-error (MSE) for the out-of-sample observations. This measure quantifies the accuracy of our model's predictions by assessing the average squared differences between the predicted values and the actual observations.

Following this, I calculate the relative forecast performance index, denoted as F. This index expresses the relative difference in forecast error between our model and the Random Walk (RW) model, which serves as a benchmark.
It provides a valuable metric for gauging how well our model performs in comparison to the simple RW model.

$$F(t)_{model} = \frac{SE_{RW}(t) - SE_{model}(t)}{SE_{RW}(t)} \tag{1.3}$$

By definition, the metric is always zero for the Random Walk (RW) model, as it represents the baseline performance. For other models, the relative forecast performance index is greater than zero only if they outperform the RW model. This makes it a clear and straightforward measure to assess which models are more effective in making accurate predictions compared to the benchmark.

Chapter 2

Modeling

2.1 Mathematical Finance Models

Much of the theoretical framework in interest rate modeling hinges on the assumption of a particular dynamic for the short-rate.
Utilizing the no-arbitrage principle, this assumption readily leads to the establishment of bond prices. Notably, the "First Fundamental Theorem of Asset Pricing" can be effectively employed to value a bond in the following manner:

$$P(t,T) = \mathbb{E}_t^Q\{e^{-\int_t^T r_s\,ds}\} \tag{2.1}$$

Using $\mathbb{E}_t^Q$ to represent the time t-conditional expectation under the risk-free probability measure Q and assuming a payoff at time T equal to 1.

Before delving into particular models, it's essential to lay the foundation with a broad category of models to which the ones I will explore belong.
This category is called "Affine Models". Citing this theorem from Mazzoni (2018):

Theorem 2.1 (Affine term structure)**.** *Suppose the short rate dynamics are governed by a general model of the form*

$$dr = \theta(r,t)dt + \sigma(r,t)dW.$$

This model has an affine term structure, if and only if the squared diffusion and drift functions are of the form

$$\sigma(r,t)^2 = a(t) + \alpha(t)r \quad \textit{and} \quad \theta(r,t) = b(t) + \beta(t)r,$$

for some smooth functions $a(t)$, $b(t)$, $\alpha(t)$, $\beta(t)$, and $A(t,T)$ and $B(t,T)$ satisfy the differential equation system

$$\begin{aligned}\frac{\partial A(t,T)}{\partial t} &= \frac{1}{2}a(t)B(t,T)^2 - b(t)B(t,T),\\ \frac{\partial B(t,T)}{\partial t} &= -\frac{1}{2}\alpha(t)B(t,T)^2 + \beta(t)B(t,T) - 1,\end{aligned}$$

for all $t < T$, with $A(T,T) = B(T,T) = 0$.

While the present work does not focus on bond pricing, it remains crucial to grasp the concept of affine models. This understanding is significant because I will employ models from this category in this chapter. This choice is influenced by the extensive theoretical framework that has been established around affine models.

2.1.1 Vasicek

An initial instance of short-rate dynamics was introduced by Vasicek in his seminal work Vasicek (1977).
After deriving a general form of the term structure of interest rates, he presents an illustrative case within this framework. This particular case later became known as the Vasicek model. It marked the inception of a series of subsequent models aimed at enhancing its efficacy.

The short-rate's dynamic, also referred to as the instantaneous spot rate as denoted by Vasicek in the article, is identified by an Ornstein-Uhlenbeck process characterized by constant coefficients.
This choice of dynamics stands as a distinctive alternative to the conventional Brownian motion. It is grounded in the empirical observations pertaining to interest rates, specifically their tendency to revert towards a mean level.

The SDE (stochastic differential equation) takes the form:

$$dr_t = k[\theta - r_t]dt + \sigma dW_t \tag{2.2}$$

where r_0, k, θ and σ are positive constants. A great feature of this model is the explainability, indeed:

- k can be seen as the average return speed
- θ as the long-term average
- σ as the instantaneous volatility

The affine nature of this model, as explained in the theorem 2.1, readily follows noting that:

$$\begin{aligned} \theta(r,t) &= b(t) + \beta(t)r = k\theta - kr_t \\ \sigma(r,t)^2 &= a(t) + \alpha(t)r = \sqrt{0r_t + \sigma^2} = \sigma \end{aligned} \tag{2.3}$$

After some calculations, it is also possible to write down an explicit solution for the SDE:

$$r_t = r_s e^{-k(t-s)} + \theta\left(1 - e^{-k(t-s)}\right) + \sigma \int_s^t e^{-k(t-u)} dW_u \tag{2.4}$$

with $t > s$.
From this, it becomes evident that the interest rates, under this dynamic, follow a Gaussian distribution.
This outcome holds significant implications, as it implies that interest rates can also assume negative values - a crucial consideration when applying this model to real-world data.

The conditional density of interest rates at time t, given the rates at time s, can be established as follows:

$$f(r_t|r_s, \theta, k, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2(t-s)}} \exp\left(-\frac{[r_t - (r_s + k(\theta - r_s)(t-s))]^2}{2\sigma^2(t-s)}\right) \tag{2.5}$$

Moreover, it is feasible to derive the precise conditional moments of our distribution analytically. Specifically:

$$\mathbb{E}[r_s|r_t] = r_s e^{-k(t-s)} + \theta\left(1 - e^{-k(t-s)}\right) \tag{2.6}$$

$$Var[r_s|r_t] = \frac{\sigma^2}{2k}\left(1 - e^{-2k(t-s)}\right) \tag{2.7}$$

These explicit and straightforward solutions serve as valuable tools for forecasting using the Vasicek model. Their clarity and ease of implementation eliminate the necessity for computationally intensive Monte Carlo methods.

Calibration

Calibrating the Vasicek model to real-world data can be approached through various methods, in this work we are going to use the same approach exposed by Bernal (2016).

The first step involves discretizing the model. One common approach is the Euler-Maruyama Scheme.
Commencing from the SDE (2.2), it leads directly to the ensuing equation:

$$r_{t+\delta t} = r_t + k[\theta - r_t]\delta t + \sigma\sqrt{\delta t}N(0,1) \tag{2.8}$$

Here, δt represents an infinitesimally small time interval. It is possible to leverage the distributional property of the Brownian motion to discretize the stochastic term. Starting from this point, I follow the procedure explained by the already cited work, employing the least squares method.

Least squares method

Referring to equation (2.8), it can be reformulated as:

$$r_{t+\delta t} = r_t(1 - k\delta t) + k\theta\delta t + \sigma\sqrt{\delta t}N(0,1) \tag{2.9}$$

So more generally:

$$r_{t+\delta t} = ar_t + b + \epsilon \tag{2.10}$$

Starting from this linear equation, the estimation of a linear regression using the least squares method is straightforward.
After obtaining the estimations for a, b and ϵ, it is subsequently possible to derive the calibrated parameter of interest in the following way:

$$\begin{aligned} \hat{k} &= \frac{1-a}{\delta t} \\ \hat{\theta} &= \frac{b}{1-a} \\ \hat{\sigma}^2 &= \frac{\hat{Var}(\epsilon)}{\delta t} \end{aligned} \tag{2.11}$$

Maximum Likelihood Estimation

An advanced step in the estimation process involves employing maximum likelihood estimation, utilizing previously estimated parameters as initial values.

Under the Euler scheme, the distribution of $r_{t+\delta t}$, derived from the referenced Vasicek density (2.5), is expressed as:

$$f(r_{t+\delta t}|r_t, \theta, k, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2(\delta t)}} \exp\left(-\frac{[r_{t+\delta t} - (r_t + k(\theta - r_t)\delta t)]^2}{2\sigma^2\delta t}\right) \tag{2.12}$$

Consequently, the log-likelihood function is defined as:

$$L(\theta) = \prod_{i=1}^{n} f(r_{t_{i+1}}|r_{t_i}, \theta) = \sum_{i=1}^{n} \ln f(r_{t_{i+1}}|r_{t_i}, \theta) \tag{2.13}$$

$$= \sum_{i=1}^{n} \ln\left[\frac{1}{\sqrt{2\pi\sigma^2(\delta t)}} \exp\left(-\frac{[r_{t+\delta t} - (r_t + k(\theta - r_t)\delta t)]^2}{2\sigma^2\delta t}\right)\right] \tag{2.14}$$

$$= \sum_{i=1}^{n} \ln \frac{1}{\sqrt{2\pi\sigma^2(\delta t)}} \left(-\frac{[r_{t+\delta t} - (r_t + k(\theta - r_t)\delta t)]^2}{2\sigma^2\delta t}\right) \tag{2.15}$$

$$= ln\left(\sqrt{2\pi\sigma^2(\delta t)}\right)^{-n} - \sum_{i=1}^{n}\left(-\frac{[r_{t+\delta t} - (r_t + k(\theta - r_t)\delta t)]^2}{2\sigma^2\delta t}\right) \tag{2.16}$$

Here, θ represents the entire parameter set rather than solely the long-term average of the Vasicek model. The estimations will be derived from solving the optimization problem:

$$\hat{\theta} = \arg\max_{\theta} \ln L(\theta) \tag{2.17}$$

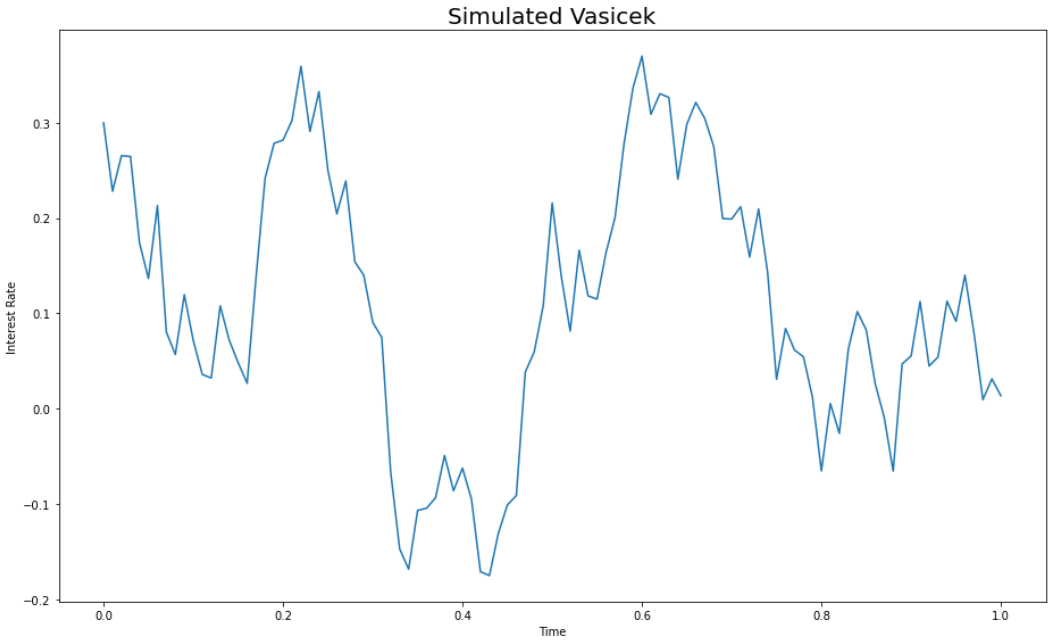
This approach seeks to maximize the log-likelihood function, thereby refining our parameter estimates for the model.

Simulated Results

To demonstrate the efficacy of the estimation method discussed earlier, simulated data will be utilized. Before delving into the simulated results, let's establish the theoretical parameters of the model:

- Mean Reversion Speed (k) = 7
- Long-Term Mean (θ) = 0.05
- Volatility (σ) = 0.5
- Initial Short Rate (r_0) = 0.3

Employing these specified values, the model will generate synthetic data representing interest rate movements. Below a graphical time series representing the simulated path:



On this simulated dataset, the previously explained two-step estimation approach has been utilized. The table below presents estimates obtained through both OLS and MLE methods:

	Real	**OLS**	**MLE**
k	7	9.587	10.078
θ	0.05	0.080	0.080
σ	0.5	0.562	0.591

To maximize the log-likelihood function, I employed the *minimize* function from the *scipy* Python package. Below, various visual representations are displayed, each depicting the log-likelihood with fixed values for 2 or 3 parameters, effectively demonstrating their distinct behaviors.

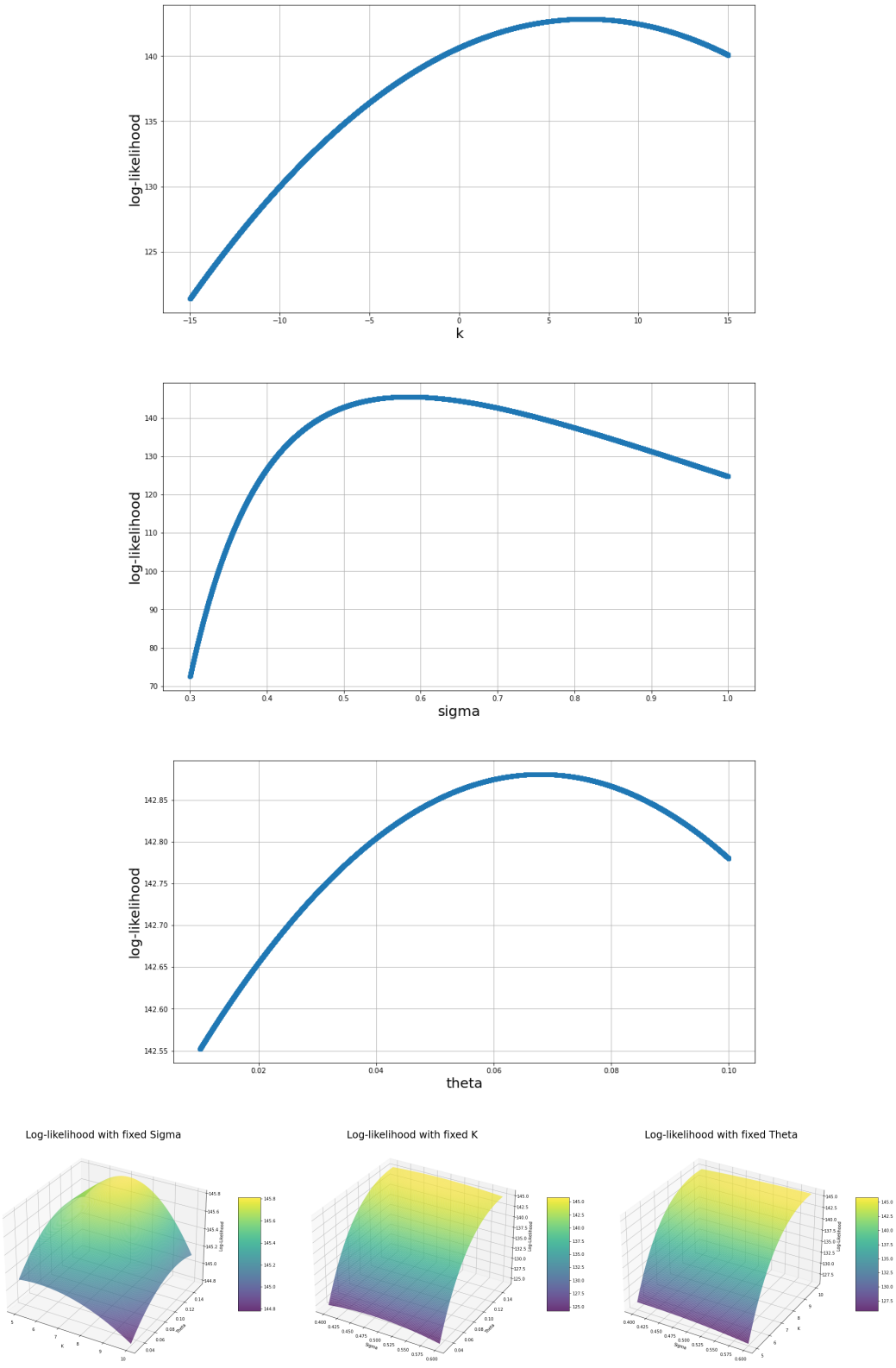


Figure 2.1. Log-Likelihood function plots

2.1.2 Cox, Ingersoll and Ross (CIR)

An important alternative of the Vasicek model is the Cox, Ingersoll and Ross (CIR) model, developed by Cox et al. (2005).

To prevent the short-rate from going negative, a modification is applied to the Vasicek model. In mathematical terms:

$$dr_t = k[\theta - r_t]dt + \sigma\sqrt{r_t}dW_t \tag{2.18}$$

with r_0, k, θ, σ positive constants. Moreover, an additional assumption is considered in order to avoid the interest rate to reach the zero boundary, $k\theta > \sigma^2$.
As for the Vasicek model, also the CIR model exhibits an affine structure, in particular recalling the theorem 2.1:

$$\begin{aligned} \theta(r,t) &= b(t) + \beta(t)r = k\theta - kr_t \\ \sigma(r,t)^2 &= a(t) + \alpha(t)r = \sqrt{\sigma^2 r_t + 0} = \sigma\sqrt{r_t} \end{aligned} \tag{2.19}$$

Starting from the SDE, similarly to what is done for the Vasicek model, it is possible to have an explicit solution or the interest rate, which takes the following form:

$$r_t = r_s e^{-k(t-s)} + k\theta \int_s^t e^{-k(t-u)}du + \sigma \int_s^t e^{-k(t-u)}\sqrt{r_u}dW_u \tag{2.20}$$

Upon reviewing Equation 2.20, it becomes evident that within the stochastic integral, the integrand function incorporates the stochastic process r_u.
This inclusion significantly augments the model's complexity when compared to the Vasicek model.

When working with a Vasicek model, we rely on a normal distribution. However, in the case of the CIR model, research has established a much more complicated closed-form conditional density function, as follows:

$$p(r_s, s, r_t, t) = ce^{-(u+v)}(\frac{v}{u})^{\frac{q}{2}} I_q(2\sqrt{uv}) \tag{2.21}$$

where:

$$\begin{aligned} c &= \frac{2k}{\sigma^2(1 - e^{-k\delta t})} \\ u &= cr_s e^{-k\delta t} \\ v &= cr_t \\ q &= \frac{2k\theta}{\sigma^2} - 1 \\ \delta t &= t - s \end{aligned} \tag{2.22}$$

And $I_q(.)$ denotes the modified Bessel function of the first kind of order q.

Explicit-form solutions for the conditional moments have been derived. As observed from 2.18, it is evident that the key difference between the derived solutions and those of the Vasicek model lies in the variance component, whereas the expected value remains consistent.

$$\begin{aligned} \mathbb{E}[r_t|r_s] &= r_s e^{-k(t-s)} + \theta\left(1 - e^{-k(t-s)}\right) \\ Var[r_t|r_s] &= \frac{\sigma^2}{2k}\left[\theta(1 - e^{-k(t-s)})^2 + 2r_t(e^{-k(t-s)} - e^{-2k(t-s)})\right] \end{aligned} \tag{2.23}$$

Calibration

Calibrating the CIR model encompasses various approaches, ranging from the simplest to the most intricate methods. In this analysis, I adopt the methodology elucidated by Kladívko (2007), which leverages the explicit density described in Equation (2.21) to construct a Maximum Likelihood Estimation (MLE) procedure for model calibration.

As a preliminary step, akin to the Vasicek's approach, I begin by formulating a discretized version of the model. Building upon the SDE detailed in equation (2.18), I readily derive its discretized version:

$$r_{t+\delta t} - r_t = k(\theta - r_t)\delta t + \sigma\sqrt{r_t}N(0, \delta t) \tag{2.24}$$

Although I employ the MLE method, it remains important to establish a discretized form of the model. This is imperative as it necessitates an initial set of parameter values, which will be derived from their OLS estimations. Consequently, let's express equation (2.24) as follows:

$$\frac{r_{t+\delta t} - r_t}{\sqrt{r_t}} = \frac{k\theta\delta t}{\sqrt{r_t}} - k\sqrt{r_t}\delta t + \sigma\sqrt{\delta t}N(0, 1) \tag{2.25}$$

Which can be read as:

$$y_i = \beta_1 z_{1,i} + \beta_2 z_{2,i} + \epsilon_i \tag{2.26}$$

where:

$$\begin{aligned} y_i &= \frac{r_{t+\delta t} - r_t}{\sqrt{r_t}} \\ \beta_1 &= k\theta \\ \beta_2 &= -k \\ z_{1,i} &= \frac{\delta t}{\sqrt{r_t}} \\ z_{2,i} &= \sqrt{r_t}\delta t \\ \epsilon_i &= \sigma\sqrt{\delta t}N(0, 1) \end{aligned} \tag{2.27}$$

Now estimating the parameters of interest becomes trivial:

$$\begin{aligned} \hat{k} &= -\hat{\beta}_2 \\ \hat{\theta} &= \frac{\hat{\beta}_1}{\hat{k}} \\ \hat{\sigma^2} &= \frac{\hat{Var}(\epsilon)}{\delta_t} \end{aligned} \tag{2.28}$$

As shown by Miao (2018), in the case of Vasicek calibration, the OLS estimates closely resemble the MLE estimates. However, this congruence does not hold true for the CIR model. Therefore, these OLS estimates can only serve as an initial value for the forthcoming MLE procedure, which I will elucidate.

Maximum Likelihood Estimation

When dealing with an interest rate time series comprising N observations, the likelihood function can be expressed as follows:

$$L(\theta) = \prod_{i=1}^{n} p(r_{t_{i+1}}|r_{t_i}, \theta) \tag{2.29}$$

Nonetheless, I will work with the logarithmic version as it significantly simplifies computations. This approach is feasible because it constitutes a monotonic transformation, ensuring that the arg max of the function remains unaltered.
It becomes:

$$\ln L(\theta) = \sum_{i=1}^{n} \ln p(r_{t_{i+1}}|r_{t_i}, \theta) \tag{2.30}$$

Subsequently, it is possible to derive the relevant log-likelihood function for the CIR model by employing equation (2.21) as follows:

$$\ln L(\theta) = n \ln c + \sum_{i=1}^{n} (-u_{t_i} - v_{t_{i+1}}) + \frac{q}{2} \frac{v_{t_{i+1}}}{u_{t_i}} + \ln(I_q(2\sqrt{u_{t_i} v_{t_{i+1}}}) \tag{2.31}$$

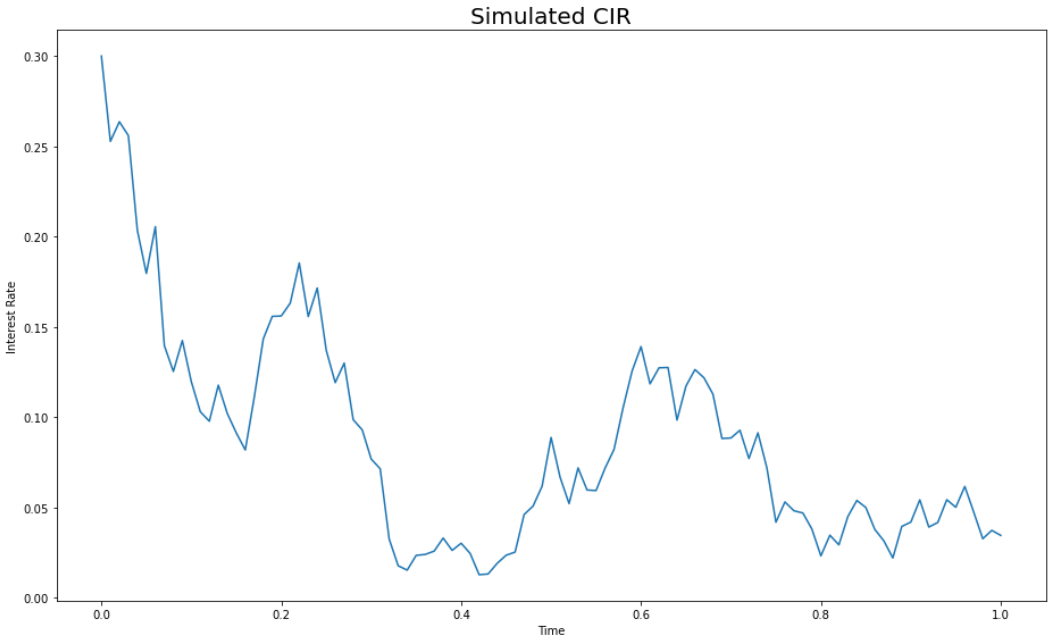
The estimates will be the solutions that satisfy this optimization problem:

$$\hat{\theta} = \arg\max_{\theta} \ln L(\theta) \tag{2.32}$$

Simulated Results

Following the demonstration seen in the Vasicek model, I will now present simulated results of the estimation procedures for the CIR model. The parameters and Gaussian samples used for simulation remain consistent, thereby highlighting the contrast between the two models originating from identical mathematical conditions.

- Mean Reversion Speed (k) = 7
- Long-Term Mean (θ) = 0.05
- Volatility (σ) = 0.5
- Initial Short Rate (r_0) = 0.3



Here the estimates employing both the OLS and the MLE technique.

	Real	**OLS**	**MLE**
k	7	8.965	9.729
θ	0.05	0.057	0.058
σ	0.5	0.562	0.592

In the next page the graphical representations of the log-likelihood:

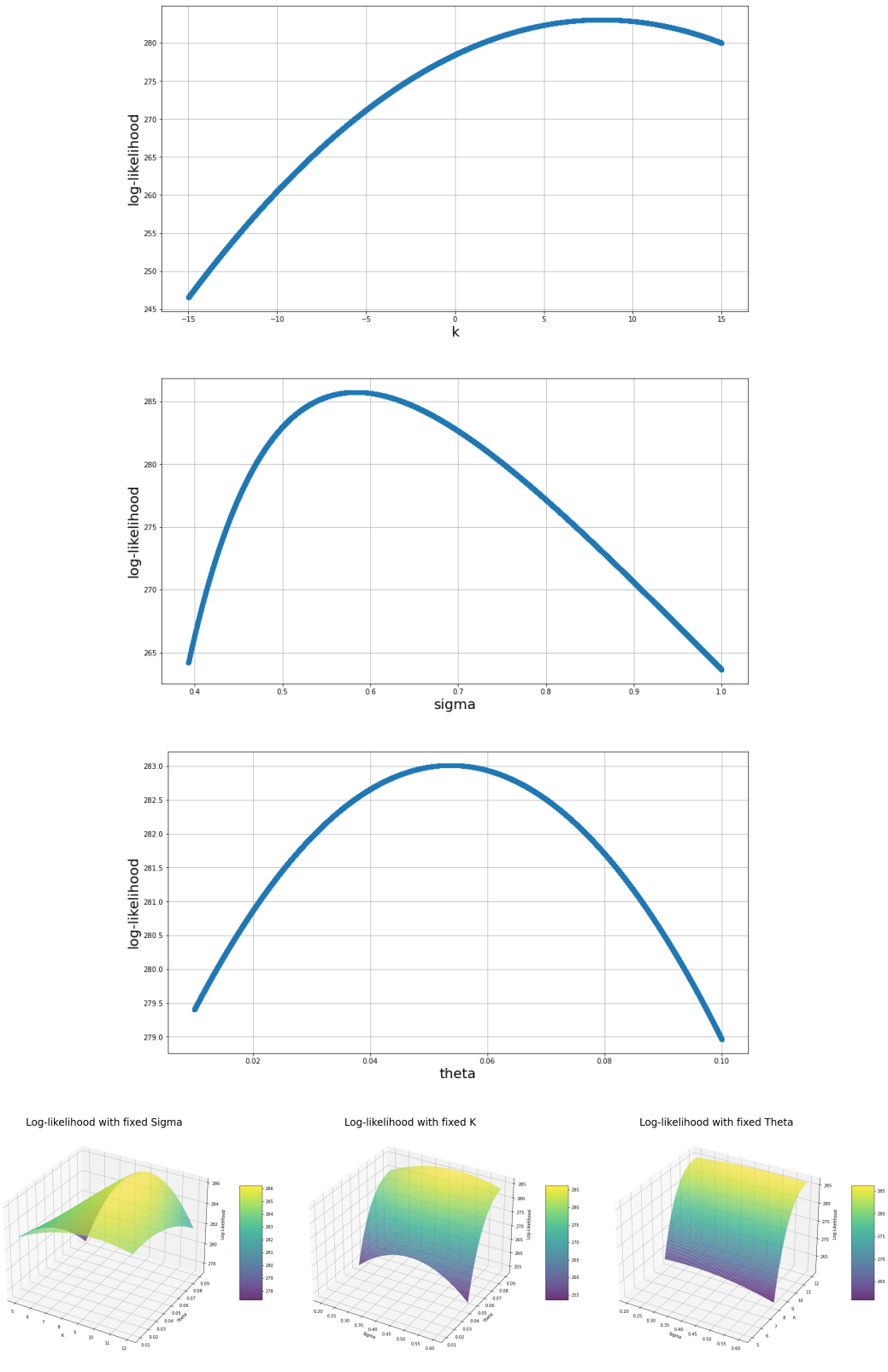


Figure 2.2. Log-Likelihood Plots

2.2 Machine Learning Models

Machine learning models encompass a diverse set of statistical techniques designed to learn patterns from data and make predictions or decisions when presented with new, unseen data. These models serve as powerful tools in extracting insights, recognizing patterns, and predicting outcomes based on historical information. Categorizing machine learning models involves several key distinctions that highlight their functionality and training methodologies.

Machine learning models can be initially categorized based on the nature of the task they perform:

- **Regression algorithms** excel in solving problems involving continuous value prediction. They aim to estimate relationships between variables and make predictions based on continuous numerical outcomes.
- **Classification algorithms** are designed to forecast or categorize data into distinct classes or categories. They excel in scenarios where the outcome needs to be classified into predefined classes or labels.

Moreover, machine learning models can be classified according to their distinct training procedures:

- **Supervised Learning**: These models rely on labeled input and output training data pairs. They learn from the provided labeled dataset to map inputs to outputs accurately. Supervised learning is particularly adept at tasks requiring prediction or classification based on historical data.
- **Unsupervised Learning**: In contrast, unsupervised learning algorithms work with unlabeled or raw data, aiming to identify inherent patterns, structures, or relationships within the data itself. These models explore data without explicit supervision and are instrumental in tasks such as clustering, anomaly detection, and uncovering hidden patterns within the data.

Both models utilized in this study, namely Bagging and LSTM (Long Short-Term Memory), fall within the domain of Supervised Learning algorithms. Below a general representation of a supervised learning problem as explained in Bontempi (2021):

1. A vector of n random input variables $x \in X \subseteq R^n$, whose values are i.i.d. according to an unknown probabilistic distribution $F_X(.)$
2. A target operator which transforms the input values into outputs $y \in Y$ according to an unknown conditional probability distribution $F_Y(y|X = x)$
3. A collection D_N of N input/output data points $< x_i, y_i >$, $i = 1, ..., N$, called the training set and drawn according to the joint input/output density $F_{X,Y}(x, y)$
4. A learning machine or learning algorithm which, on the basis of the training set D_N, returns an estimation (or prediction) of the target for an input x. The input/output function estimated by the learning machine is called hypothesis or model.

2.2.1 Bagging and Random Forests

Decision Trees

The foundation of Bagging rests upon the fundamental concept of decision trees. A decision tree serves to divide the input space into distinct and exclusive regions, with each region being allocated a specific procedure for characterizing its individual data points.
Within a decision tree, nodes can be categorized into two types: internal nodes and terminal nodes. An internal node functions as a decision point, employing a decision function to ascertain the subsequent child node to explore. Conversely, a terminal node, also known as a leaf, lacks child nodes and corresponds to one of the input space partitions.
In classification trees, each terminal node contains a label denoting the class associated with the input region it represents. On the other hand, in regression trees, the terminal node houses a model that defines the mapping between input and output for the relevant input partition.

The approach used for the splitting is known as *recursive binary splitting*. The tree begins by selecting a variable X_j and a corresponding threshold value s such that splitting the regressor space into regions $X|X_j < s$ and $X|X_j > s$ leads to the largest possible reduction in error. In the case of regression trees, the residual sum of squares (RSS) is used, defined as

$$RSS = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2 \tag{2.33}$$

Bagging and Random Forests

The primary drawback of decision trees lies in their high variance, leading to unstable estimations that can significantly fluctuate with minor changes in the training dataset. To counteract this variance, various techniques combine estimators, with two prominent methods being Bagging and Random Forest.

Bagging involves iteratively creating samples and constructing trees from each sample. Subsequently, it aggregates predictions from each tree to derive the final prediction. The algorithm operates as follows:

1. Generate B different bootstrapped samples of size n (with replacement)
2. build a tree on each sample and obtain a prediction for a given x, $\hat{f}_b(x)$
3. Compute the average of all B predictions to get the final bagging prediction

$$\hat{f}_{bagging}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_b(x) \tag{2.34}$$

Random Forest extends the Bagging method by imposing constraints on the number of covariates in each bootstrapped sample. The Random Forest approach involves the following steps:

1. Draw a random sample (with replacement) of size n
2. Randomly select m covariates from the full set of p covariates (where $m \approx \sqrt{p}$)
3. Build a tree using the selected m covariates and obtain a prediction for a given x, $\hat{f}_{b,m}(x)$
4. Repeat steps 1 to 3 for all B bootstrapped samples
5. Compute the average of individual tree predictions in order to obtain the random forest prediction, $\hat{f}_{rf}$

$$\hat{f}_{rf}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_{b,m}(x) \tag{2.35}$$

By restricting the number of covariates considered in each bootstrapped sample, Random Forest aims to reduce the correlation among individual trees, thereby enhancing predictive accuracy and generalization capabilities.

Considering the inherent characteristics of financial data, where proximity in time holds greater relevance for observations, I chose to employ Bagging methodology in this study. Specifically, I engineered my features at each time step by leveraging interest rates from past periods.
Given that I have 90 observations per time step within the rolling window and I sacrifice one observation for each chosen feature, I restricted the data retention to 10 periods before the present time.
A total of 100 bootstrapped trees were selected for this approach.

2.2.2 Long-Short-Term-Memory (LSTM)

Feed-forward Neural Network

The simplest Neural Network architecture is known in literature as Feed-forward Neural Network (FNNs).
FNNs exhibit a layered arrangement, where each layer consists of individual or simpler processing units referred to as "nodes." These nodes are interconnected, with each node in one layer connected to one or more nodes in the subsequent layer through real-valued weights, often termed parameters. However, connections between nodes within the same layer are absent.

Every FNN comprises an input layer, responsible for receiving the initial data, and an output layer, which provides the resulting predictions or outputs. Typically, FNNs are constructed with an additional node known as the bias unit in all layers except the output layer.
Below is an illustrative representation of an FNN structure:

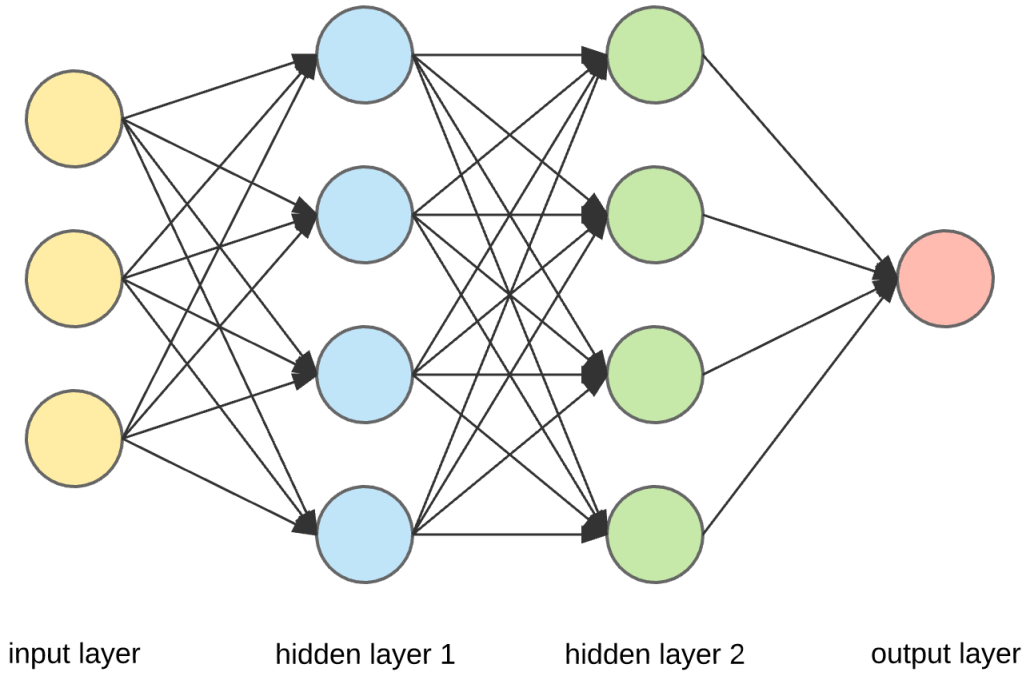


Figure 2.3. FNN's Architecture

Mathematically, considering the following definitions:

- n the number of inputs
- L the number of Layers
- $H^{(n)}$ the number of hidden units of the lth layer $(l = 1, ..., L)$ of the FNN
- $w_{kv}^{(l)}$ denote the weight of the link connecting the kth node in the $l-1$ layer and the vth node in the l layer
- $z_v^{(l)}$, $v = 1, ..., H^l$ the output of the vth hidden node of the lth layer

- $z_0^{(l)}$ denote the bias for the l layer
- Let $H^{(0)} = n$ and $z_v^{(0)}$, $v = 0, ..., n$

For $l \geq 1$, the output of the vth hidden unit ($v = 1, ..., H^l$) in the lth layer is obtained by:

1. Forming a weighted linear combination of the $H^{(l-1)}$ outputs from the preceding layer:

   $a_v^{(l)} = \sum_{k=1}^{H^{(l-1)}} w_{kv}^{(l)} z_k^{(l-1)} + w_{0v}^{(l)} z_0^{(l-1)}$

2. Transforming this sum using an activation function to get the final output of the vth hidden node:

   $z_v^{(l)} = g^{(l)}\left(a_v^{(l)}\right)$

Here, $g^{(l)}(.)$ represents the activation function, typically a non-linear transformation such as the logistic or sigmoid function.

Recurrent Neural Network

Recurrent Neural Networks (RNNs) represent a distinct architecture that addresses two critical limitations of Feedforward Neural Networks (FNNs): handling variable-length sequences and capturing input directionality. These capabilities hold immense significance, particularly in domains such as Natural Language Processing (NLP) and time series forecasting.

RNNs achieve this by incorporating feedback loops among their hidden layers, allowing them to retain information over time and process sequences of varying lengths. This architecture's fundamental design allows it to maintain memory across sequential data.
Below is a visual depiction of the RNN architecture:

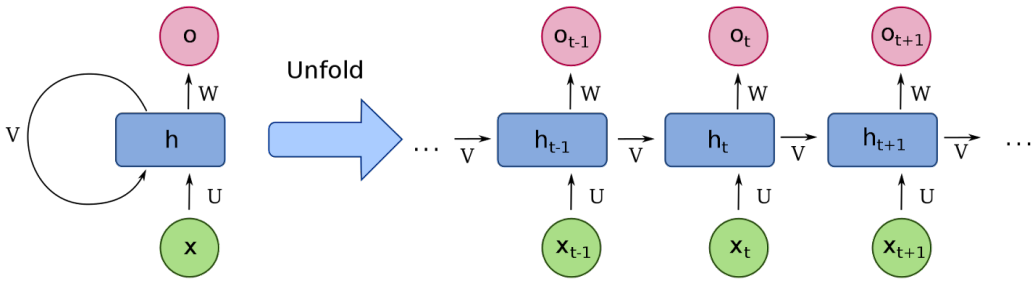


Figure 2.4. RNN's Architecture

In an RNN, each hidden state is a result of a linear combination involving a linear transformation of the current input and the preceding hidden layer. This characteristic enables the model to retain memory and capture temporal dependencies within sequential data.

LSTM

Vanilla RNNs exhibit remarkable potential for various tasks. However, when deployed on real-world data, they often encounter significant challenges, particularly when handling extensive datasets.

Two prevalent issues encountered are known as the vanishing and exploding gradients problems. Delving into the intricacies of these problems involves complex mathematical concepts, which won't be covered in this work. Nonetheless, it is crucial to acknowledge these challenges. These issues lay the groundwork for introducing a more advanced model called Long Short-Term Memory (LSTM), specifically designed to overcome these inherent limitations in vanilla RNNs.
Graphically, the LSTM model can be visualized as follows:

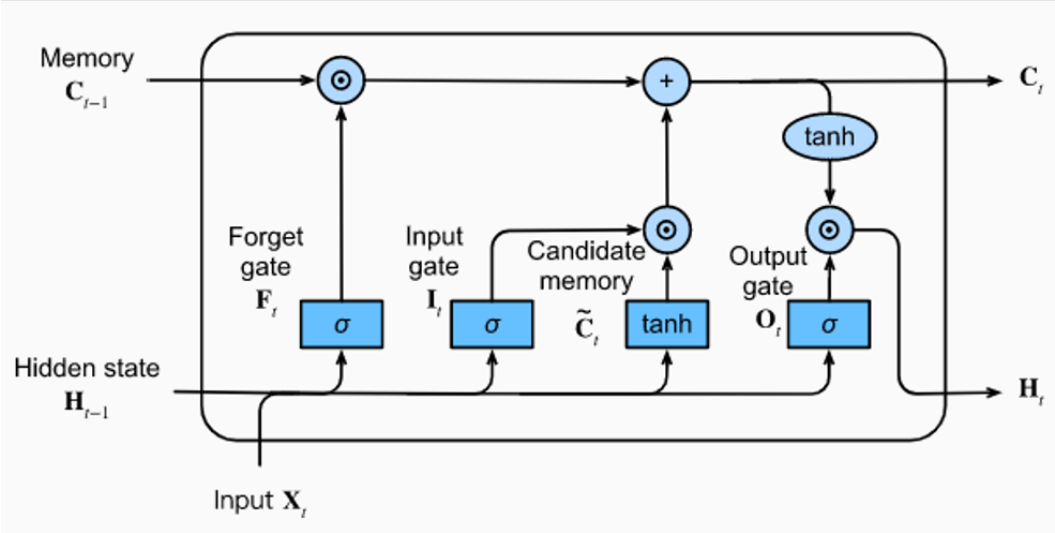


Figure 2.5. LSTM's Architecture

Long Short-Term Memory (LSTM) model consists of several crucial components that enable it to process sequential data effectively:

1. **Cell State (C_t):** The cell state acts as an information highway, allowing data to flow through the entire sequence while selectively adding or removing information. It helps in retaining long-term dependencies.

2. **Input Gate (i_t):** This gate determines which values from the input should be updated and added to the cell state. It controls how much new information is stored in the cell state from the current input.

3. **Forget Gate (f_t):** The forget gate decides what information in the cell state should be discarded or forgotten. It regulates the removal of irrelevant or unnecessary information from the cell state.

4. **Output Gate (o_t):** The output gate controls the information that gets exposed to the next hidden state. It selects the relevant information from the updated cell state to produce the output.

5. **Hidden State (h_t):** The hidden state carries information to the next time step. It's a filtered version of the cell state that focuses on the information that the model decides is important to pass on.

Chapter 3

Calibration & Forecasting

In this section, the empirical results obtained from the application of different methodologies on the 13 Weeks Treasury Bill Rates dataset, comprising 600 observations from 06/08/2021 to 22/12/2023, are presented and discussed.

3.1 Overview

3.1.1 Applied Methods

- **Vasicek and CIR Models**: For the Vasicek and CIR models, predictions were made for four different time steps: 1 day, 1 week, 1 month, and 3 months. These models were fitted using historical data, and predictions were generated for each specified time horizon.
- **Bagging and LSTM Models**: However, for the Bagging and LSTM models, predictions were only computed for a 1-day horizon. Due to the nature of supervised learning, these models were trained using a rolling window approach, utilizing the last 90 observations to predict the subsequent 1-day value. As a result, making predictions for longer horizons using these models was not feasible.

3.1.2 Model Calibration Approach

In Chapter 2, the estimation methodology for the Vasicek and CIR models was elucidated. This method comprises two primary steps: initial estimation via Ordinary Least Squares (OLS) followed by Maximum Likelihood Estimation (MLE). While adopting a more intricate approach is often recommended to ensure more precise estimates, due to computational constraints, I opted for a rolling window analysis solely utilizing OLS estimates.

Conversely, in the case of Random Forest and Long Short-Term Memory (LSTM) models, I leveraged the robust and efficient algorithms readily available in Python. Specifically, I employed *RandomForestRegressor* from the *sklearn* package and *LSTM* from the *Keras* package.

3.1.3 Performance Assessment

The performance assessment begins with a comparison of mean-squared-error (MSE) across each model, enabling the creation of a ranked list that indicates the relative predictive accuracy of these models.
The study then proceeds to explore how these models' performances vary over time for different time steps. This temporal analysis utilizes the F metric introduced in the study Molenaars et al. (2013), defined as:

$$F(t)_{model} = \frac{SE_{RW}(t) - SE_{model}(t)}{SE_{RW}(t)} \tag{3.1}$$

An essential aspect to note about this scoring metric is its behavior: it reaches a maximum value close to infinity when the Random Walk's error approaches zero, and it approaches a value close to one when the model's error approaches zero. Leveraging this characteristic, two statistics are chosen to thoroughly analyze each model's performance.

This analytical approach includes the generation of a series of graphical representations accompanied by two key metrics:

1. the Median F-score for each model
2. an accuracy metric, indicating the percentage of forecasts that outperform the Random Walk, relative to the total number

These graphical representations effectively illustrate instances when a model either surpasses or falls short of outperforming the benchmark (the Random Walk) on specific dates.

The purpose behind utilizing this methodology is to offer a more lucid and detailed portrayal of how the various models compare to the benchmark across different time intervals. This approach facilitates a comprehensive understanding of the relative strengths and weaknesses exhibited by these models within distinct time periods, contributing to a nuanced evaluation of their predictive capabilities.

3.2 Forecasting

3.2.1 Aggregated Results

Below, a sequence of tables presents the empirical results obtained for each model:

Table 3.1. Squared-Error Statistics for Random Walk

	1 day ahead	1 week ahead	1 month ahead	3 months ahead
mean	3.200209e-07	2.578023e-06	2.274667e-05	1.691888e-04
std	8.034475e-07	6.000665e-06	3.139453e-05	1.837303e-04
min	0.000000e+00	0.000000e+00	2.500114e-09	2.499638e-09
max	6.002500e-06	5.183999e-05	1.729225e-04	6.786024e-04

Table 3.2. Squared-Error Statistics for Vasicek

	1 day ahead	1 week ahead	1 month ahead	3 months ahead
mean	3.170212e-07	2.359788e-06	2.342957e-05	3.766314e-01
std	7.508250e-07	5.416998e-06	1.212859e-04	7.428566e+00
min	2.513944e-14	6.968489e-12	1.844309e-11	5.211745e-08
max	6.670697e-06	4.035466e-05	2.194620e-03	1.521947e+02

Table 3.3. Squared-Error Statistics for CIR

	1 day ahead	1 week ahead	1 month ahead	3 months ahead
mean	3.139147e-07	2.203167e-06	1.343410e-05	3.509478e-03
std	7.389320e-07	4.846396e-06	3.060873e-05	4.978649e-02
min	1.052765e-12	1.175202e-12	9.778788e-11	2.182804e-11
max	5.944143e-06	3.983810e-05	3.303571e-04	9.725348e-01

Table 3.4. Squared-Error Statistics for Bagging

	1 day ahead
mean	5.124738e-07
std	1.199837e-06
min	1.207728e-12
max	1.170256e-05

Table 3.5. Squared-Error Statistics for LSTM

	1 day ahead
mean	4.110676e-07
std	9.427976e-07
min	7.052564e-14
max	9.747694e-06

Upon analyzing the data, it becomes evident that the CIR model consistently outperforms all others. However, its superiority over the Random Walk is not significantly pronounced, posing challenges in unequivocally establishing the failure of the

benchmark. Similarly, Vasicek displays a comparable performance but consistently falls short of the CIR model. Notably, both Machine Learning models—Bagging and LSTM—consistently lag behind other models in making accurate predictions for a single step ahead.

3.2.2 Time-Based Results

This section shifts focus from aggregated performance to a time-based analysis, specifically utilizing the 3.1 metric to assess models' performance over time. The graphical representation highlights instances where a model surpasses the Random Walk (RW) in green and where it falls short in red.

The objective is to discern whether superior performance compared to the RW follows any discernible pattern or if it appears sporadic. Through this visual approach, the investigation aims to uncover trends or regularities in the models' outperformance relative to the RW, thereby shedding light on the predictability of their improved performance over time.

1 day ahead

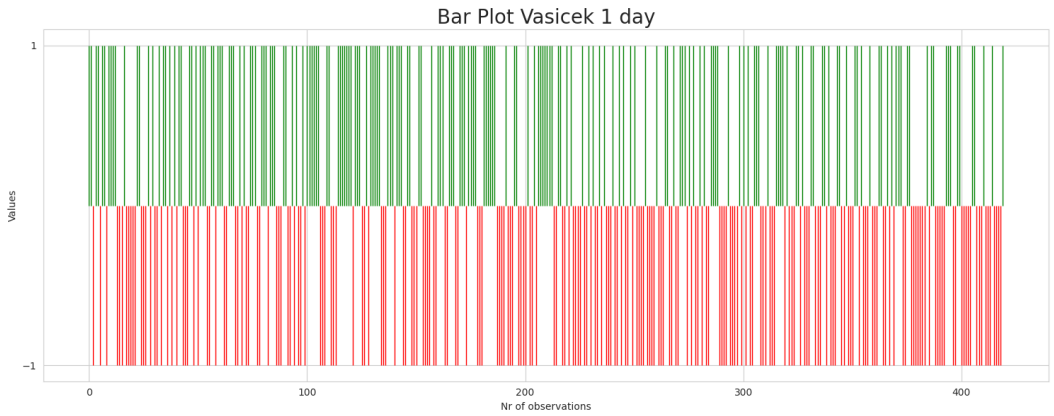


Figure 3.1. Median F-score=-0.11, Accuracy Ratio=47.14%

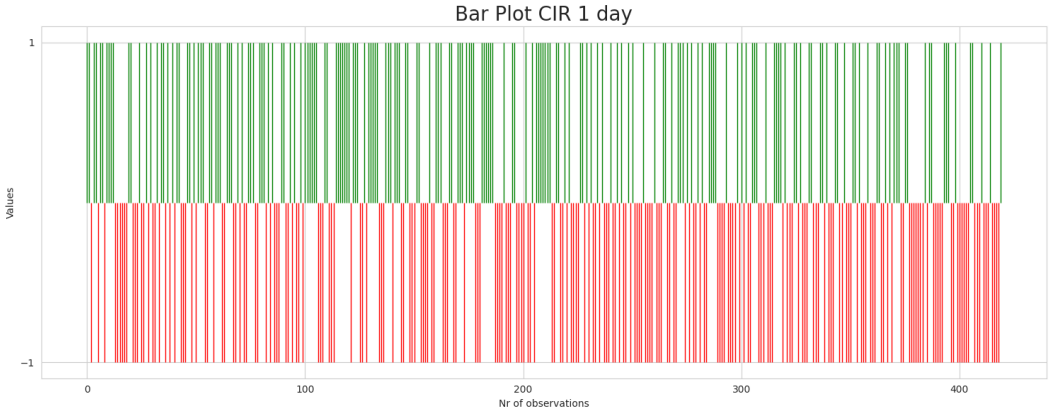


Figure 3.2. Median F-score = -0.12, Accuracy Ratio=46.66%

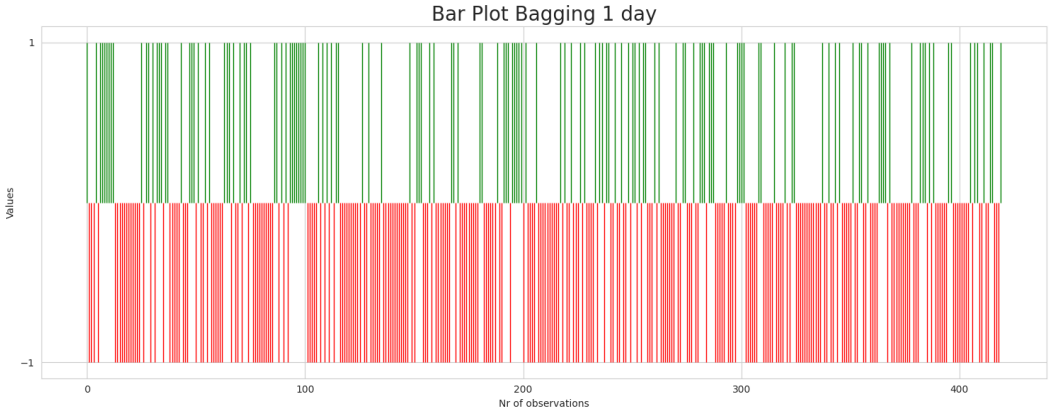


Figure 3.3. Median F-score = -0.74, Accuracy Ratio=34.52%

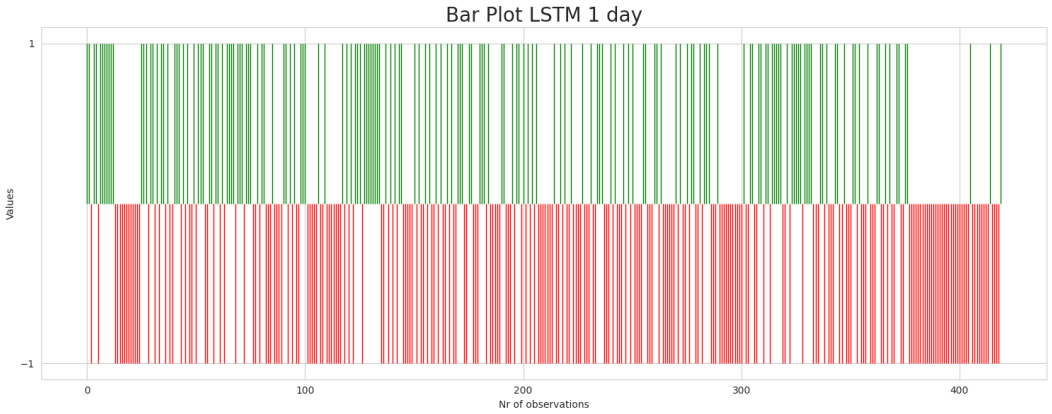


Figure 3.4. Median F-score = -0.29, Accuracy Ratio=41.42%

1 week ahead

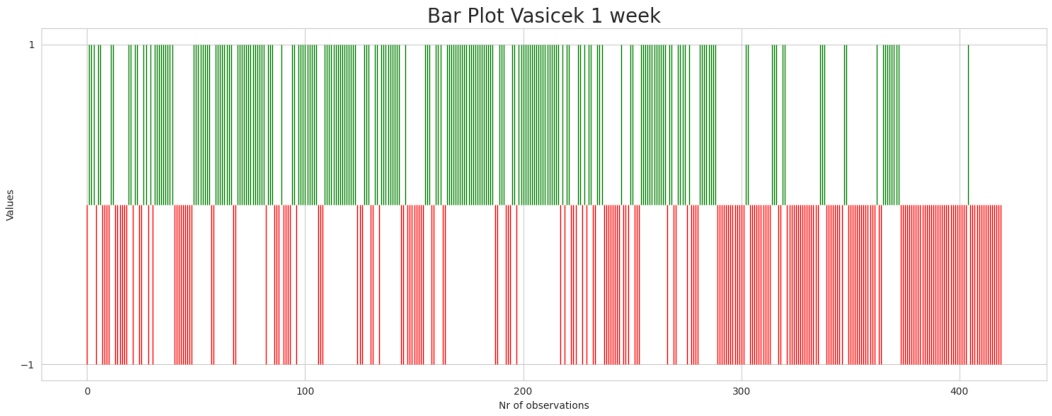


Figure 3.5. Median F-score = 0.07, Accuracy Ratio=50.47%

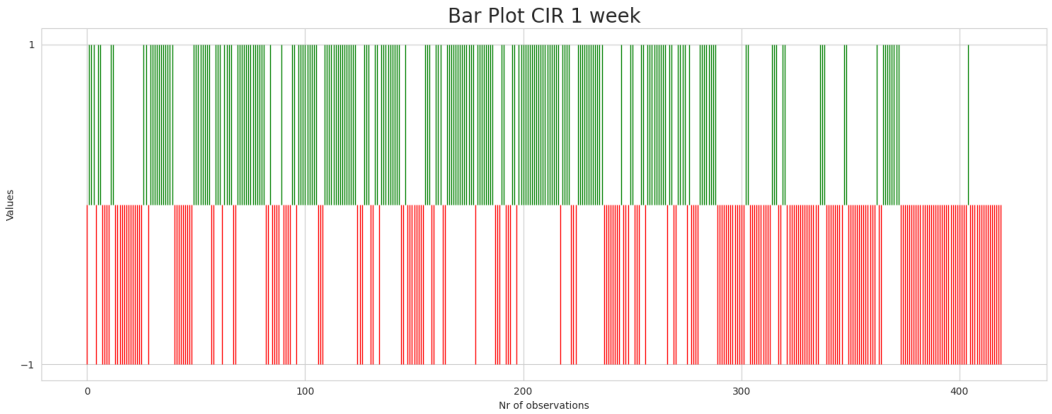


Figure 3.6. Median F-score = -0.03, Accuracy Ratio=49.52%

1 month ahead

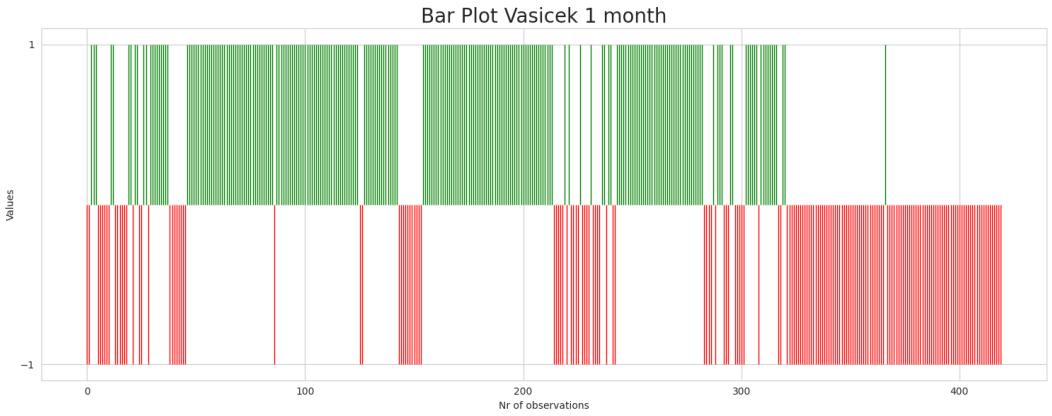


Figure 3.7. Median F-score = 0.35, Accuracy Ratio=58.33%

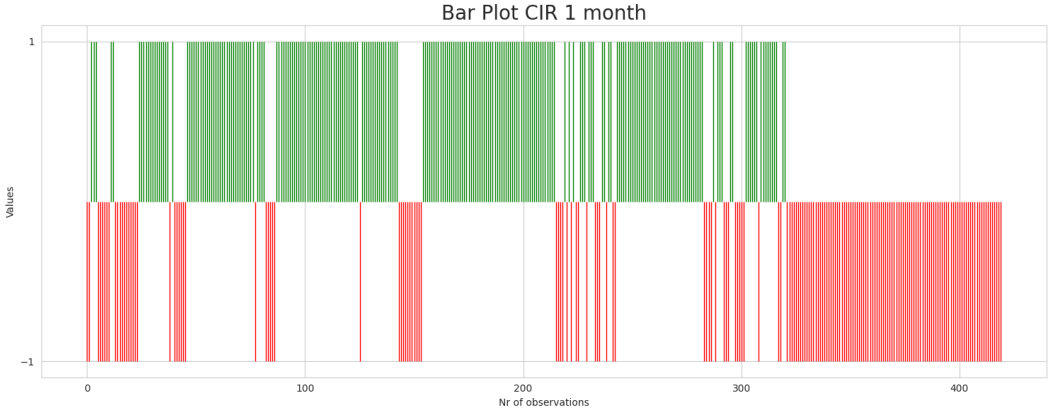


Figure 3.8. Median F-score = 0.42, Accuracy Ratio=58.57%

3 months ahead

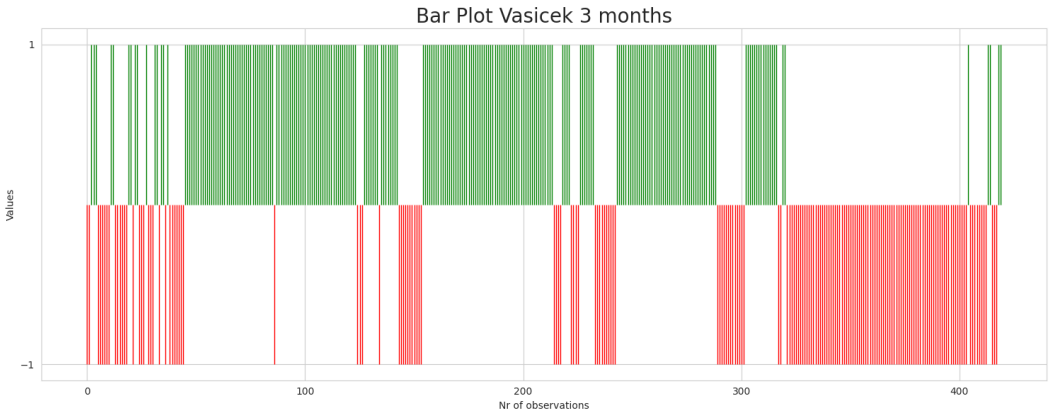


Figure 3.9. Median F-score = 2.991237e-01, Accuracy Ratio=58.80%

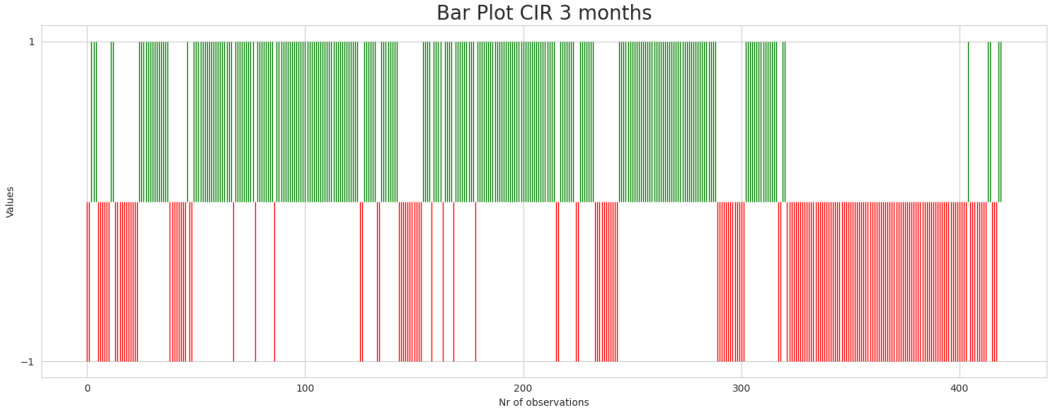


Figure 3.10. Median F-score = 0.34, Accuracy Ratio=58.33%

Comments

The analysis of F-scores reveals an intriguing pattern. While the models exhibit erratic outperformance over the Random Walk for 1-day forecasts, a discernible pattern emerges as the forecast horizon extends. This suggests the potential to exploit predictable patterns for longer-term forecasts.

One plausible approach to further analyze this pattern involves connecting this research with studies related to the **Efficient Market Hypothesis**. This entails comparing the model's performance in two scenarios: forecasting using all available data versus forecasting solely when the market looks inefficient. If the observed patterns are correlated with this hypothesis, the performance delta for longer horizons should significantly surpass that of short-term forecasts.

Chapter 4

Conclusions

The thesis undertakes a comprehensive evaluation of interest rate forecasting, leveraging models drawn from both mathematical finance and machine learning literature. Models such as Vasicek, CIR, Bagging, and LSTM are examined across various forecast horizons spanning 1 day, 1 week, 1 month, and 3 months, employing a rolling window approach of 90 observations at each step.

In the realm of 1-day ahead forecasts, the CIR and Vasicek models exhibit intermittent superiority over the Random Walk. Conversely, the Bagging and LSTM models consistently underperform, with Bagging demonstrating the poorest performance based on Mean Squared Errors, while CIR emerges as the best performer.

For longer-term forecasts, emphasis is placed on the Vasicek and CIR models. CIR outperforms the Random Walk across almost all forecast horizons, except notably for the 3-month horizon where the Random Walk takes the lead. Vasicek mirrors the behavior of CIR but consistently falls short, indicating its inferior adaptation to real data compared to CIR.

An intriguing pattern surfaces upon the analysis of F-scores: while the models display sporadic outperformance over the Random Walk in 1-day forecasts, a discernible pattern emerges as the forecast horizon extends. This suggests the potential to exploit predictable patterns for longer-term forecasts.

Building on this finding, a suggested direction for future research involves linking this empirical methodology with the Efficient Market Hypothesis. This potential study would seek to investigate whether leveraging market inefficiencies can improve forecasting accuracy. By comparing this method with the current study's utilization of all available historical data for modeling and forecasting, the research aims to highlight the performance difference achieved when forecasting only during periods where the market appears inefficient.

This proposed research could offer valuable insights into refining forecasting strategies and understanding the relationship between market efficiency and predictive modeling in financial markets.

Appendix

4.1 Python Code

In this section I illustrate the code related to the calibration of the various models.

```
# Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.ticker as mtick
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import BaggingRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import LSTM, Dense
import yfinance as yf
print('All packages loaded')

# Variables of interest for the calibration
y = df['rates t']
X = df[['rates t-1']]
ts_length = df.shape[0]
n_obs = 90
max_time_step = 90
dt = 1/252

# initiate arrays at each model
pred1_v = np.zeros(ts_length-(n_obs+max_time_step-1))
pred7_v = np.zeros(ts_length-(n_obs+max_time_step-1))
pred30_v = np.zeros(ts_length-(n_obs+max_time_step-1))
pred90_v = np.zeros(ts_length-(n_obs+max_time_step-1))

#-------------#
#-Random Walk-#
#-------------#
"""
create a for loop for the rolling window, but only for those
timeframes where backtesting is possible
```

```
"""
for n in range(ts_length-n_obs):

    # by doing that I keep all the observations from n to (n+n_obs-1)
    X_train = X.iloc[n:n+n_obs, :]
    y_train = y.iloc[n:n+n_obs]
    prediction_index = y_train.index[-1]

    # Need to break the loop if my max prediction cannot be backtested
    if (prediction_index + max_time_step) > y.index[-1]:
        break

    # predictions and exact values at 4 time steps
    r0 = y_train.iloc[-1]

    pred1 = r0
    pred7 = r0
    pred30 = r0
    pred90 = r0

    # store the predictions
    pred1_v[n] = pred1
    pred7_v[n] = pred7
    pred30_v[n] = pred30
    pred90_v[n] = pred90

#---------------------------------

#---------------#
#-Vasicek model-#
#---------------#
for n in range(ts_length-n_obs):

    # linear regression needed for the Vasicek calibration
    model = LinearRegression()

    # by doing that I keep all the observations from n to (n+n_obs-1)
    X_train = X.iloc[n:n+n_obs, :]
    y_train = y.iloc[n:n+n_obs]
    prediction_index = y_train.index[-1]

    # Need to break the loop if my max prediction cannot be backtested
    if (prediction_index + max_time_step) > y.index[-1]:
        break

    # fit the model
    model.fit(X_train, y_train)

    # get parameters of the linear regression
    intercept = model.intercept_
    slope = model.coef_[0]
```

```
    # get the parameter of interest for Vasicek
    k = (1-slope)/dt
    theta = intercept / (1-slope)

    # predictions at 4 time steps
    r0 = y.iloc[prediction_index]

    pred1 = r0 * np.exp(-k*dt) + theta * (1 - np.exp(-k*dt))
    pred7 = r0 * np.exp(-k*7*dt) + theta * (1 - np.exp(-k*7*dt))
    pred30 = r0 * np.exp(-k*30*dt) + theta * (1 - np.exp(-k*30*dt))
    pred90 = r0 * np.exp(-k*90*dt) + theta * (1 - np.exp(-k*90*dt))

    # store the predictions
    pred1_v[n] = pred1
    pred7_v[n] = pred7
    pred30_v[n] = pred30
    pred90_v[n] = pred90
#--------------------------------

#-----------#
#-CIR model-#
#-----------#
for n in range(ts_length-n_obs):

    # linear regression needed for CIR calibration
    model = LinearRegression(fit_intercept=False)

    # by doing that I keep all the observations from n to (n+n_obs-1)
    X_train = X['rates t-1'].iloc[n:n+n_obs]
    y_train = y.iloc[n:n+n_obs]
    prediction_index = y_train.index[-1]

    # Need to break the loop if my max prediction cannot be backtested
    if (prediction_index + max_time_step) > y.index[-1]:
        break

    y_cir = (y_train - X_train) / np.sqrt(X_train)
    z1 = dt / np.sqrt(X_train)
    z2 = dt * np.sqrt(X_train)
    X_cir = np.column_stack((z1, z2))

    model.fit(X_cir, y_cir)

    # Calculate the predicted values (y_hat), residuals and the parameters
    beta1 = model.coef_[0]
    beta2 = model.coef_[1]

    # get the parameter of interest for CIR
    k = -beta2
    theta = beta1/k

    # predictions at 4 time steps
```

```
    r0 = y.iloc[prediction_index]

    pred1 = r0 * np.exp(-k*dt) + theta * (1 - np.exp(-k*dt))
    pred7 = r0 * np.exp(-k*7*dt) + theta * (1 - np.exp(-k*7*dt))
    pred30 = r0 * np.exp(-k*30*dt) + theta * (1 - np.exp(-k*30*dt))
    pred90 = r0 * np.exp(-k*90*dt) + theta * (1 - np.exp(-k*90*dt))

    # store the predictions
    pred1_v[n] = pred1
    pred7_v[n] = pred7
    pred30_v[n] = pred30
    pred90_v[n] = pred90
#--------------------------------

#---------------#
#-Random Forest-#
#---------------#
# Specify the number of trees in the bagging ensemble
n_estimators = 100
# Specify the number of lags
num_lags = 10

for n in range(ts_length - n_obs):
    # Choose the base model for bagging (a decision tree regressor)
    base_model = DecisionTreeRegressor(random_state=0)

    # Create the BaggingRegressor with the base model
    model = BaggingRegressor(estimator=base_model, n_estimators=n_estimators, random_st

    # By doing that I keep all the observations from n to (n+n_obs-1)
    y_train = y[n:n + n_obs]
    prediction_index = y_train.index[-1]

    # Need to break the loop if my max prediction cannot be backtested
    if (prediction_index + max_time_step) > y.index[-1]:
        break

    # Fill the lagged data dataframe
    lagged_data = pd.DataFrame()
    for lag in range(1, num_lags + 1):
        lagged_data[f"Lag_{lag}"] = y_train.shift(lag)

    # Create initial df
    lagged_data = lagged_data.dropna()
    target = pd.DataFrame(y_train[num_lags:])

    # Reset indexes
    lagged_data.reset_index(inplace=True)
    target.reset_index(inplace=True)
    lagged_data = lagged_data.iloc[:, 1:]
    target = pd.DataFrame(target.iloc[:, 1])
```

```
    # Create the lagged data to predict t+1 observation
    X_test = lagged_data.iloc[-1, :]
    new_lag_1 = target.iloc[-1]
    X_test_new = pd.concat([pd.Series(new_lag_1), X_test[:-1]])
    X_test_new.index = X_test.index

    # Train a BaggingRegressor model
    model.fit(lagged_data, np.ravel(target))

    # Store the predictions
    pred1_v[n] = model.predict(X_test_new.to_frame().T)
#---------------------------------

#------#
#-LSTM-#
#------#
counter = 0
for n in range(ts_length-n_obs):

    # by doing that I keep all the observations from n to (n+n_obs-1)
    y_train = y[n:n+n_obs]
    prediction_index = y_train.index[-1]
    interest_rates = y_train.values

    # Need to break the loop if my max prediction cannot be backtested
    if (prediction_index + max_time_step) > y.index[-1]:
        break

    print(f'this is the loop n {counter}')
    counter += 1
    # Normalize the data using MinMaxScaler
    scaler = MinMaxScaler(feature_range=(0, 1))
    interest_rates_normalized = scaler.fit_transform(interest_rates.reshape(-1, 1))

    # Split the data into input (X) and output (y)
    X_lstm = interest_rates_normalized[:-1]
    y_lstm = interest_rates_normalized[1:]

    # Reshape data for LSTM model (samples, time steps, features)
    X_lstm = np.reshape(X_lstm, (X_lstm.shape[0], 1, 1))

    # Build the LSTM model
    model = Sequential()
    model.add(LSTM(units=5, input_shape=(X_lstm.shape[1], X_lstm.shape[2])))
    model.add(Dense(units=1))

    model.compile(optimizer='adam', loss='mean_squared_error')

    # Train the model
    model.fit(X_lstm, y_lstm, epochs=25, batch_size=1, verbose=1)

    # Predict using the trained model the rates at t+1
```

```
#last_observations = interest_rates_normalized.reshape(len(interest_rates_normalize
predicted_next_interests = model.predict(y_lstm)
predicted_next_interests = scaler.inverse_transform(predicted_next_interests)

# store only the last prediction, which has as input the last known rate
pred1_v[n] = predicted_next_interests[-1][0]
```

Acknowledgments

Desidero esprimere la mia sincera gratitudine a tutte le persone che hanno contribuito al completamento di questo percorso accademico e personale.

Innanzitutto, un ringraziamento va al Prof. Sergio Bianchi, il mio relatore, per avermi introdotto in modo eccellente al mondo della Finanza Matematica.
Lavorare a questa tesi durante lo svolgimento di un tirocinio in Banca Centrale Europea non è stato semplice, ma la sua fiducia nella mia capacità di sviluppare questo progetto di tesi è stata fondamentale per lo sviluppo di competenze di ricerca che spero di mettere a frutto in un futuro dottorato. La sua dedizione e la sua passione per l'insegnamento sono state determinanti nel mio percorso accademico nonché di crescita personale.
Lo ringrazio particolarmente per avermi lasciato grande margine decisionale e possibilità di approfondire gli aspetti più confacenti alla ricerca.

Un ringraziamento speciale va alla mia famiglia per il costante sostegno durante questi cinque anni di studio, sotto tutti i punti di vista. Senza il loro incoraggiamento e supporto che mi hanno fornito, non avrei avuto l'opportunità di vivere un'esperienza così formativa come il mio semestre a Bruxelles in Erasmus, periodo che ha arricchito profondamente la mia prospettiva e mi ha permesso di crescere in modo significativo.

Un sentito ringraziamento va anche ai miei compagni di corso, con i quali ho condiviso anni meravigliosi sia dentro che fuori l'aula. Le nostre esperienze condivise hanno reso questo viaggio accademico indimenticabile.

Non posso dimenticare i miei amici di sempre, coloro che mi hanno sostenuto e incoraggiato a dare il massimo. La loro costante presenza nella mia vita è stata un punto fisso che mi ha guidato attraverso le sfide e i successi di questi anni e che spero rimanga per sempre, come farò io per loro.

Grazie di cuore.

Bibliography

Bernal, V. (2016). *Calibration of the Vasicek Model: An step by step guide*. PhD thesis, PhD thesis.

Bontempi, G. (2021). 'statistical foundations of machine learning' handbook.

Brigo, D. and Mercurio, F. (2006). *Interest rate models-theory and practice: with smile, inflation and credit*, volume 2. Springer.

Chan, F. and Mátyás, L. (2022). *Econometrics with machine learning*. Springer.

Cox, J. C., Ingersoll Jr, J. E., and Ross, S. A. (2005). A theory of the term structure of interest rates. In *Theory of valuation*, pages 129–164. World Scientific.

den Butter, F. A. and Jansen, P. W. (2013). Beating the random walk: a performance assessment of long-term interest rate forecasts. *Applied Financial Economics*, 23(9):749–765.

Kladívko, K. (2007). Maximum likelihood estimation of the cox-ingersoll-ross process: the matlab implementation. *Technical Computing Prague*, 7(8):1–8.

Mazzoni, T. (2018). *A First Course in Quantitative Finance*. Cambridge University Press.

Miao, Z. (2018). Cir modeling of interest rates.

Molenaars, T. K., Reinerink, N. H., and Hemminga, M. A. (2013). Forecasting the yield curve-forecast performance of the dynamic nelson-siegel model from 1971 to 2008.

Orlando, G., Mininni, R. M., and Bufalo, M. (2020). Forecasting interest rates through vasicek and cir models: A partitioning approach. *Journal of Forecasting*, 39(4):569–579.

Vasicek, O. (1977). An equilibrium characterization of the term structure. *Journal of financial economics*, 5(2):177–188.