A Comparative Simulation Study of Classical and Machine Learning Techniques for Forecasting Time Series Data

https://doi.org/10.3991/ijoe.v19i08.39853

Mbarek Iaousse[1], Youness Jouilil[2(✉)], Mohamed Bouincha[3], Driss Mentagui[2]
[1]C3S Laboratory, Hassan II University of Casablanca, Casablanca, Morocco
[2]Department of Mathematics, Faculty of Sciences, Ibn Tofail University of Kenitra, Kenitra, Morocco
[3]Faculty of Legal, Economic and Social Sciences of Sale, Mohamed V University of Rabat, Rabat, Morocco
y.jouilil@gmail.com

Abstract—This manuscript presents a simulation comparison of statistical classical methods and machine learning algorithms for time series forecasting notably the ARIMA model, K-Nearest Neighbors (KNN), The Support Vector Regression (SVR), and Long-Short Term Memory (LSTM). The performance of the models was evaluated using different metrics especially Mean Squared Error (MSE), Mean Absolute Error (MAE), Median Absolute Error (Median AE), and Root Mean Squared Error (RMSE). The results of the simulations approve that the KNN and LSTM algorithms have better accuracy than the others models' forecasting notably in the medium and long term. Hence, in the medium and long term, ML models are so powerful on big datasets. However, Machine learning architectures outperform ARIMA for shorter-term predictions. Thus, ARIMA is most appropriate in the case of univariate small data sets, where deep learning algorithms are not yet at their best.

Keywords—machine learning, time series forecasting, classical approaches, forecasting

1 Introduction

In the recent decade, time series forecasting and analysis have become an important field, especially in medicine, economy, and industry [1]. This paper provides an in-depth examination of the most efficient and widely utilized machine learning algorithms for forecasting. In fact, we propose to conduct a general approach to time series generation.

The simulation will be performed to investigate the potential of classical and machine learning algorithms to improve the accuracy of forecasting. We design simulations from stationary models where the residual terms follow the standard normal distribution.

Then, we simulate a large time process data set and compute their corresponding features.

The rest of the paper will be structured as follows. We start with the methodology and then the time series generation and data preprocessing. In the third section, we will compare our time series algorithms using different accuracy metrics. In the last, we conclude.

2 Methodology

This section exposes the methodology and a review of classical and deep sequential architectures for forecasting time series. Specifically, we consider the following algorithms: ARIMA, SVR, KNN and LSTM architectures.

2.1 ARIMA algorithm

Auto-Regressive Integrated Moving Average, or ARIMA, models are a class of models that are used to analyze and forecast time series data [2]. The model is defined mathematically as:

$$(1+\phi_1 B+\phi_2 B^2+\cdots+\phi_p B^p)(1-B)^d X_t=(1+\theta_1 B+\theta_2 B^2+\cdots+\theta_q B^q)\eta_t \quad (1)$$

Where X_t the time series data at time t, B is the backshift operator, d is the order of differencing, θ_i and ϕ_i are the parameters of the model, and η_t is the error term.

2.2 K-nearest neighbors algorithm

The K-nearest Neighbors (KNN) algorithm is a non-parametric method used for classification and regression. The algorithm tries to find the K observations in the training dataset that are closest to the observation to be predicted, and it assigns the most common output value among those K observations to the observation to be predicted. Mathematically, the KNN algorithm can be represented as: Given a new observation, xnew, and a training dataset, Xtrain, the KNN algorithm finds the K closest observations in Xtrain to xnew. These K closest observations are represented by XKNN. The algorithm then assigns the most common output value among those K closest observations to xnew.

Algorithm 1: K-nearest Neighbors Algorithm

```
Procedure: KNN(xnew, Xtrain, ytrain, K)
    XKNN ← FindKNN(xnew, Xtrain, K)
    ynew ← MostCommonValue(ytrain[XKNN])
    return ynew
end procedure
```

Where xnew is the new observation, Xtrain is the training dataset, ytrain is the corresponding output values for the training dataset, K is the number of nearest neighbors

to consider, XKNN is the set of K nearest neighbors, and ynew is the predicted output value for the new observation.

2.3 Support vector regression algorithm

Support Vector Regression (SVR) is a supervised machine learning algorithm. It is used for forecasting time series. Typically, it is used to forecast the future values of a given time series on the basis of past values [3]. Therefore, it is a powerful method to predict future results based on historical information.

The idea of SVR is to find a function that can accurately predict future values. The generic SVR can be expressed as follows:

$$f(x_i) = \omega\phi(x_i) + b \tag{2}$$

Where $i \in \{1, 2, \ldots, n\}$ and

$$Min\frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{n}(\zeta_i + \zeta_i^*) \tag{3}$$

Subject to:

$$y_i - f(x_i) \le \epsilon_i + \zeta_i \tag{4}$$

$$f(x_i) - y_i \le \epsilon_i + \zeta_i^* \tag{5}$$

$$C, \zeta_i, \zeta_i^* \ge 0; i \in \{1, 2, \ldots, n\} \tag{6}$$

The Support Vector Regression (SVR) algorithm can be summarized as follows:

Algorithm 2: Support Vector Regression Algorithm
Step 1 : Select kernel function. Step 2 : Select the value of C. Step 3 : Compute the Gram matrix K using kernel function and input data. Step 4 : Solve the quadratic programming problem defined by objective function and constraints. Step 5 : Obtain parameters w and b. Step 6 : Use w and b to make forecasting. Step 7 : Summarize findings.

2.4 Long short term memory algorithm

Long Short-Term Memory (LSTM) is a type of Recurrent Neural Network (RNN) that is able to capture long-term dependencies in sequential data [4]. This is achieved through the use of memory cells, which allow information to flow through the network over multiple time steps. LSTMs are particularly useful for tasks such as speech recognition and natural language processing, where the input sequences can be very long [5]. The mathematical equations for an LSTM unit can be represented as follows:

$$f_t = \sigma(W_{fh} h_{t-1}, W_{fx} x_t, b_f) \tag{7}$$

$$i_t = \sigma(W_{ih} h_{t-1}, W_{ix} x_t, b_i) \tag{8}$$

$$c_t = f_t * c_{t-1} + i_t \odot \tanh(W_{ch} h_{t-1}, W_{cx} x_t, b_x) \tag{9}$$

$$\beta_t = \sigma(W_{\beta h} h_{t-1}, W_{\beta x} x_t, b_\beta) \tag{10}$$

$$h_t = o_t \odot \tanh(c_t) \tag{11}$$

Where i_t, f_t, o_t and g_t are the input, forget, output and cell gates respectively, c_t is the cell state and h_t is the hidden state. W and b are the weight matrices and biases respectively, and σ and tanh are the sigmoid and hyperbolic tangent functions respectively.

3 Data generation & data preprocessing

3.1 Data generation

To assess the underlying algorithms derived in Section 2, we simulated various time processes data sets. Generating stationary Time-Series artificial data will be done by using Python programming language.

First of all, we will transform all the previous algorithms into Python Code. For that purpose, we will begin by fixing the libraries that will be helpful for the whole process notably NumPy, statsmodels, sklearn packages, and notably keras library.

To take in all possible shapes, we will vary the sample size from smallest to largest size. More precisely, the sample size (T) of the simulated time process is set to be:

$$T \in \{10; 15; 30; 100; 1000; 5000; 10000; 100000\}$$

It should be mentioned that with the previous parameter setting, the time process is stationary. For each range of sample size, we have generated 384 replications of integrated time series. Then, we implemented the proposed algorithms in Python language. In particular, a part of the Python code that has been used to run these simulations is given above.

After, we fitted the algorithms on the training set using the appropriate Python functions. Note that the accuracy criterions used for comparison purposes are RMSE. Overall, all the underlying models provide a low RMSE value, which revealed a good forecast accuracy.

The data set related to the work has been created by way of simulation based on random parameters. Now that we have our generated artificial data set, let's see how it compares visually. To do this, we can make a plot for each of the 7 features and show their variation with respect to timestep.

Algorithm 3 : Generation of Stationary AR Processes Coefficients

Input : p ← integer

Output : *AR stationary time process coefficients*

Function *stationary_ar(p)*

```
While True do
    Φ = (ϕ1, ϕ2, ϕ3,..., ϕp)' ~ i.i.d Up(-1,1);
    Solve equation 1-ϕ1Z-ϕ2Z²-ϕ3Z³-...-ϕpZᵖ = 0;
    Roots ← Define roots as the list of all equation's solutions;
    Compute ||Roots||;
    if ||Roots|| > 1 for all roots in Roots then
        Break;
    end
end
return Φ;
```

Algorithm 4 : Generation of Stationary AR Processes Coefficients

Input : Data ← Empty Dictionary, P,D,Q ← integers, Simulated_data ← Empty Directionary, T ← {15, 30, 100, 1000, 5000, 10000}

Output *: Data with P x Q x D x T univariable time processes*

```
for p = 0 to P step 1 do
    Φ ← stationary_ar(p) ;
    for q = 0 to Q step 1 do
        Θ ← (θ0, θ1, θ2,..., θq) ' Arbitrary;
        for d = 0 to D step 1 do
            for t in T do
                Simulated_Data ← ARIMA(Φ, Θ, t, d);
                Data ← Update data with Simulated_data;
            end
        end
    end
end
```

Notes: ML: machine learning; R Squared: Coefficient of determination; MSE: mean square error; MAE: mean average error; SVR: support vector regressor.

3.2 Data preprocessing

Training set vs testing set. The first step to do before running a time series simulation, we have to divide the dataset into two parts: A training set and a testing set. The first part is used to create and generate our algorithms. Then, we apply the models to the testing set to forecast future data points. In Python, we can call the traintest_splitclass from Scikit-learn class.

Feature scaling. In Python programming language, we can do that by implementing Standard-Scaler class from sklearn.preprocessing library.

It should be mentioned that all our simulations and data preprocessing will be done using Python programming language. Many packages will be used for instance scikit-learn library, Pandas, NumPy, tensorflow, Plotly, statsmodels, matplotlib, pmdarima libraries ...etc.

4 Simulation of the datasets

Algorithm 5 : Simulating an ARIMA Time Series
Step 1 : Set the length of the data set (T) ; Step 2 : Create an uncorrelated innovation series from a probability distribution; Step 3 : Make space room by creating an empty vector or a dictionary; Step 4 : Select values for the parameters; Step 5 : Define p, q, and order of integration (d) ; Step 6 : Generate several time series Step 7 : Check how our series looks right

Based on the values of the arguments n, p, q, and d, 384 data series will be raised. In other words, we generated a time process for each model of length n for different orders of p, d and q.

```
params_ma = np. random . uniform (-1, 1, 3)
sample_sizes = [10 , 15, 30, 100 , 1000 , 5000 , 10000 , 100000]
my_dict = {}
for sample_size in sample_sizes :
  for p in ordres :
  for q in ordres :
  for d in (0, 1, 2):
  ar = params_ar [:p]
  ma = params_ma [:q]
  simulated_data = ARIMA (phi = ar , theta = ma , n =
sample_size , d= d)
  my_dict . update ({" size –"+ str( sample_size )+"- ARIMA ("+ str (p)+" ,"+ str(d)+" ,"+ str(q)+")
": list ( simulated_data ) })
```

Source: Authors manipulations' under Python programming language.

We noticed that the generation of the innovations has been done from a Gaussian distribution with a mean of 0 and variance of 1. Thereafter, in order to guarantee that we all get the same findings, we have set the seed to a predetermined value (set.seed (12345)) before generating values for the time processes. We have simulated $4 \times 4 \times 3 \times 8 = 384$ time process variables by combining all possible values of p, q, and d for each sample size. When we plot the simulated ARIMA (p, d, q) processes, they look as follows:

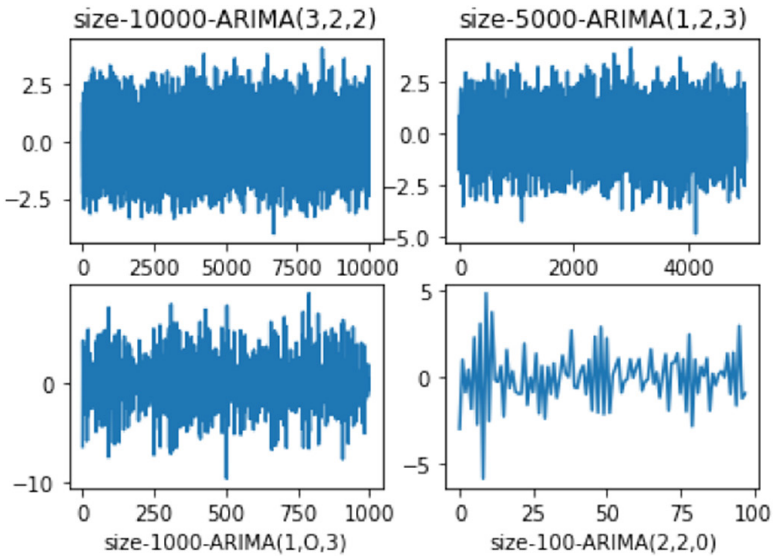

Fig. 1. Overall process

Source: Authors manipulations' under Python programming language.

Above in the Figure 1, we present a set of data sets generated by the previous algorithms.

5 Simulation results

In this section, we will explore the simulation results.

Algorithm 5 : Model Selection Algorithm for Time Series Forecasting
Step 1 : Set T (integer); Step 2 : Generate a time series; Step 3 : Split the series into training and testing period; Step 4 : Train the candidate algorithms; Step 5 : Compute accuracy metrics for each candidate algorithm; Step 6 : Choose the best candidate algorithm based on the accuracy metrics; Step 7 : Establish forecasting for the test part using the adopted model; Step 8 : Repeat steps 2 to 7 T times; Step 9 : Summarize T findings.

We applied machine learning techniques to our generated data sets. The findings obtained by using the accuracy metrics according to the simulating data sets are presented in the table below.

Algorithm 6: Evaluation Metrics for Time Series Forecasting

Input : X a training set time series, model (a trained model for forecasting)

Output : Evaluation metrics for time series forecasting

```
    mape ← 0; mae ← 0; rmse ← 0; X_pred ← model(X);
for i = 1 to length(X) step 1 do
    mape ← mape + abs(Xi −X_predi)/abs(Xi)×length(X));
    mae ← mae + abs(Xi −X_predi)/length(X);
    rmse ← rmse + (Xi −X_predi)2/length(X);
end
rmse ← squart(rmse);
Print(mape);
Print(mae);
Print(rmse);
```

Table 1. Summary of the accuracy metrics results obtained with different samples for $d = 0$

Algorithm	Sample Size							
	10	**15**	**30**	**100**	**1000**	**5000**	**10000**	**100000**
ARIMA	1.052	0.984	1.276	0.947	0.968	0.9716	0.956	0.9808
SVR	0.828	1.196	1.054	0.771	0.7501	0.919	2.12	1.157
LSTM	1.3080	1.095	1.006	0.919	1.068	1.181	0.57	0.907
KNN	1.57	1.76	1.601	0.935	0.217	0.812	0.379	0.418

Table 2. Summary of the accuracy metrics results obtained with different samples for $d = 1$

Algorithm	Sample Size							
	10	**15**	**30**	**100**	**1000**	**5000**	**10000**	**100000**
ARIMA	1.276	1.590	1.384	7.120	10.420	10.938	30.078	146.225
SVR	1.308	2.089	2.993	5.450	20.232	48.3677	100.128	103.131
LSTM	1.281	2.955	2.820	5.040	5.279	6.068	104.545	231.209
KNN	1.623	1.76	2.62	2.631	7.321	2.321	43.721	121.102

Table 3. Summary of the accuracy metrics results obtained with different samples for $d = 2$

Algorithm	Sample Size							
	10	**15**	**30**	**100**	**1000**	**5000**	**10000**	**100000**
ARIMA	2.303	3.390	4.497	417.337	801.588	551.714	4673.724	32013.217
SVR	4.446	15.001	32.806	474.213	17498.301	137187.911	197185.111	239127.112
LSTM	11.582	22.505	33.611	273.175	178.836	6041.017	47040.1	8231.162
KNN	2.871	2.879	6.812	115.862	251.97	323.209	871.62	1383.921

6 Conclusion

Forecasting plays a crucial role in the planning, managing, and monitoring of future movements. Thus, the comparison of time series forecasting based on classical and Machine Learning algorithms showed that ML and classical approaches have different strengths and weaknesses depending on the specific application and the characteristics of the dataset.

The main challenge of classical approaches requires careful not only hyperparameter tuning but also a good understanding of the data set notably the time series stationarity. In fact, ARIMA needs series of parameters (p,q,d) that must be computed based on data, while the majority of ML algorithms do not need to set such parameters [6, 7, 8].

Through the research, the simulation demonstrates that ML techniques gave more accurate and efficient results. The experimental findings (according to Tables 1–3) clearly emphasize that machine learning algorithms work better when we deal with huge amounts of data sets and enough training period is available, while classical approaches are better for smaller data sets. There are some limitations of the classical methods, especially since they are sensitive to missing and outlier values. Also, working with univariate data sets is more challenging to be employed on multivariate data sets. On the other side, machine learning algorithms are not only able to deal with the previous challenges but also can be applied to both univariate data sets and multivariate data sets without being sensitive to outliers.

Forecasting systems based on classical statistical approaches have their own limitations since they require more historical data sets to meet statistical hypotheses, for instance, determining the trends in the data, and statistical attributes like stationarity and causality. Deep learning algorithms provide a novel approach in terms of forecasting. Intuitively, the performance of the LSTM algorithm was expected to be superior to that of the ARIMA one. Investigations of simulations showed that the LSTM model provides greater accuracy compared to ARIMA one.

From the simulations, we can approve that KNN and LSTM algorithms forecasting have better accuracy than the others models' forecasting notably in the medium and long terms. Usually, those two algorithms provide better results in terms of accuracy metrics than the other candidate techniques. Paradoxically, Machine learning architectures outperform ARIMA for shorter-term predictions. Thus, ARIMA is most appropriate in the case of univariate small data sets, where deep learning algorithms are not yet at their best.

7 Funding

The authors report no conflicts of interest. All errors in this document are the responsibility of the authors.

8 References

[1] G. M. Ljung, G. E. Box, "On a measure of lack of fit in time series models," Biometrika 1978, vol. 65, pp. 297–303. https://doi.org/10.1093/biomet/65.2.297

[2] J. Youness and M. Driss, "LSTM Deep Learning vs ARIMA Algorithms for Univariate Time Series Forecasting: A case study," 2022 8th International Conference on Optimization and Applications (ICOA), Genoa, Italy, 2022, pp. 1–4. https://doi.org/10.1109/ICOA55659.2022.9934119

[3] B. Priambodo, A. Ahmad and R. Abdul Kadir, "Prediction of average speed based on relationships between neighbouring roads using K-NN and neural network," Int. J. Onl. Eng., vol. 16, no. 01, pp. 18–33, 2020. https://doi.org/10.3991/ijoe.v16i01.11671

[4] P. Srivastava, "Essentials of deep learning: Introduction to long short term memory," vol. 10, 2017.

[5] I. Nurhaida, "Implementation of deep learning predictor (LSTM) algorithm for human mobility prediction," Int. J. Interact. Mob. Technol., vol. 14, no. 18, pp. 132–144, 2020. https://doi.org/10.3991/ijim.v14i18.16867

[6] Youness Jouilil and Driss Mentagui, "Comparing the forecasting accuracy metrics of support vector regression and ARIMA algorithms for non-stationary time process," Mathematics and Statistics, vol. 11, no. 2, pp. 294–299, 2023. https://doi.org/10.13189/ms.2023.110207

[7] J. Youness and M. Driss, "An ARIMA Model for Modeling and Forecasting the Dynamic of Univariate Time Series: The case of Moroccan Inflation Rate," 2022 International Conference on Intelligent Systems and Computer Vision (ISCV), Fez, Morocco, 2022, pp. 1–5. https://doi.org/10.1109/ISCV54655.2022.9806073

[8] J. Youness and M. Driss, "Mathematical modeling of financial time series volatility: A GARCH model". 2023. In: Y. Farhaoui, A. Rocha, Z. Brahmia, B. Bhushab, (eds) Artificial Intelligence and Smart Environment. ICAISE 2022. Lecture Notes in Networks and Systems, vol. 635. Springer, Cham. https://doi.org/10.1007/978-3-031-26254-8_88

9 Authors

Mbarek Iaousse: Researcher in Applied Mathematics and Computer Sciences, C3S Laboratory, Hassan II University of Casablanca Morocco. iaousse@gmail.com; https://orcid.org/0000-0002-4949-5556

Youness Jouilil: Researcher in Applied Mathematics and Computer Sciences, Faculty of Sciences, Ibn Tofail University of Kenitra, Morocco. His areas of research focus on artificial intelligence, data science, and Econometrics. y.jouilil@gmail.com; https://orcid.org/0000-0002-0394-8329

Mohamed Bouincha: PhD in Applied Economics, Faculty of Legal, Economic and Social Sciences of Sale, Mohamed V University of Rabat, Morocco. m.bouincha@gmail.com

Driss Mentagui: Senior Researcher in Mathematics, Faculty of Sciences, Ibn Tofail University of Kenitra, Morocco. driss.mentagui@uit.ac.ma

Article submitted 2023-03-19. Resubmitted 2023-04-20. Final acceptance 2023-04-21. Final version published as submitted by the authors.