Politechnika Wrocławska
Faculty of Management
Department of Operations Research and Business Intelligence

P01: Descriptive Analytics

Maria Aya (277150)
Sebastian Suwada (255314)

TIME - SERIES FORECASTING OF SEASONAL ITEMS SALES USING MACHINE LEARNING

Introduction

In management there are many methods to know and predict the sales environment and make better decisions that favor economic growth, one of the most reliable tools is time-series forecasting, which is why in this report, we will aim to achieve the results obtained in the article entitled "*Time series forecasting of seasonal item sales using machine learning - A comparative analysis*".

As the name implies, the goal is to understand and implement different simple forecasting models, such as *Seasonal Autoregressive Integrated Moving Average* (SARIMA) and triple exponential smoothing, and more complex models, such as *Prophet, long term memory* (LSTM) and *Convolutional Neural Network* (CNN). The mentioned models will be tested on a dataset from sales reports between 2014 and 2017 from the furniture industry, one of the fastest-growing sectors that present trend and seasonality in its behavior, two of the most important components to implement the models mentioned above.

To evaluate the quality of these forecasts, there were computed by the authors several estimators for each model, providing a comprehensive basis for determining the best-performing model. These estimators include the *Mean Squared Error* (MSE), *Root Mean Squared Error* (RMSE), and the *Mean Absolute Percentage Error* (MAPE) which expresses errors as a percentage, helping to gauge the accuracy of the forecasts.

In the article used as guide the authors "*question the ability of neural networks models for the task of seasonal sales forecasting which is widely encountered in numerous applications. In addition, ... compare the performance with various classical and advanced time-series forecasting methods, to find the best method among them*." [6] For this it is just necessary to compare the obtained results and find the most accurate, from the article we know the lead of the Stacked LSTM model over the others, in this report it will be possible to find the methodology used to achieve results close to the obtained in the original article and all the similarities and differences from it.

About the data

The models were trained using a 75% portion of the dataset, while the remaining 25% was dedicated to testing the predictions generated by the selected models. This division allowed for a robust assessment of model performance, with three years of data used for training and the final year reserved for testing and selecting the most suitable model for future predictions. To ensure consistency in the analysis, the data was transformed from a daily frequency to a monthly one, aggregating values to represent the average for each month. The outcomes were then visualized through graphs, enabling a visual comparison with the original dataset. The assessment was further facilitated by the utilization of the three error estimators.

Models used

Used classical models: ARMA ARIMA and SARIMA. For the finding optimal hyperparameters of the ARIMA and SARIMA used in calculating those models, grid search was performed and used estimators like Bayesian Information Criterion (BIC), Akaike Information Criterion (AIC) which can describe the quality of the model. In the case of using Exponential Smoothing methods: there were considered two of them where one is dedicated for forecasting data with both systematic trends and recurring patterns – Triple Exponential Smoothing (TES) known also as Winter-Holt's model. Second one is dedicated to handle trends and time-series data – Double Exponential Smoothing (DES) known also as Holt's linear trend model which is controlled as a TES model using a level of alpha that indicates smoothing factor and has trending parameters. The

difference is that the Holt-Winters model includes all three components: level, trend, and seasonality where Holt's only first two. Next used model is Prophet designed by scientists working in the Data department for Facebook. It's dedicated to time-series data when dealing with seasonality and season effects. Includes four components: trend, seasonal, holiday and error. Artificial Neural Network (ANN) class of machine learning models that is possible to learn from given data, predict relationships and capture patterns using this data. Convolutional Neural Network (CNN) is also based on neural networks but includes more convolutional layers, (CNN) is mostly used for image processing which defines it as having better performance in grid-like data (2D) – matrix, two-dimensional structure. Long Short-Term Memory (LSTM) Networks is based on (RNN) Recurrent Neural Networks also called "vanilla" which is an ANN approach with additional memory for past values. In the case of LSTM, as the name shows, it is about long-range dependencies. In RNN there is a problem of vanishing values so it can have problems with remembering information from the long past. LSTM creates the possibility to remember as much information as needed. In this article is also defined counterparts of LSTM as:

- Vanilla LSTM – single hidden layer of LSTM and output.
- Stacked LSTM – create a stack of layers on top of each other.
- Bidirectional LSTM – learns from forward and using previous layers.

Results of main article

ARMA and ARIMA fail to capture the seasonality pattern present in the data. The least effective model in this dataset is the Double Exponential Smoothing (DES), with a high Mean Absolute Percentage Error (MAPE) exceeding 98%. This result suggests that DES can only discern the late-year growth pattern, limiting its usefulness to predicting the slope.

On the other hand, the top three performing models are as follows:

1. Stacked Long Short-Term Memory (LSTM) with an impressive accuracy of MAPE: 17.34%, ranking first.
2. Vanilla LSTM, securing the second position with MAPE: 18.39%.
3. A model based on Convolutional Neural Networks (CNN), which even outperforms the popular Prophet model, securing the third spot with MAPE: 22.26%.

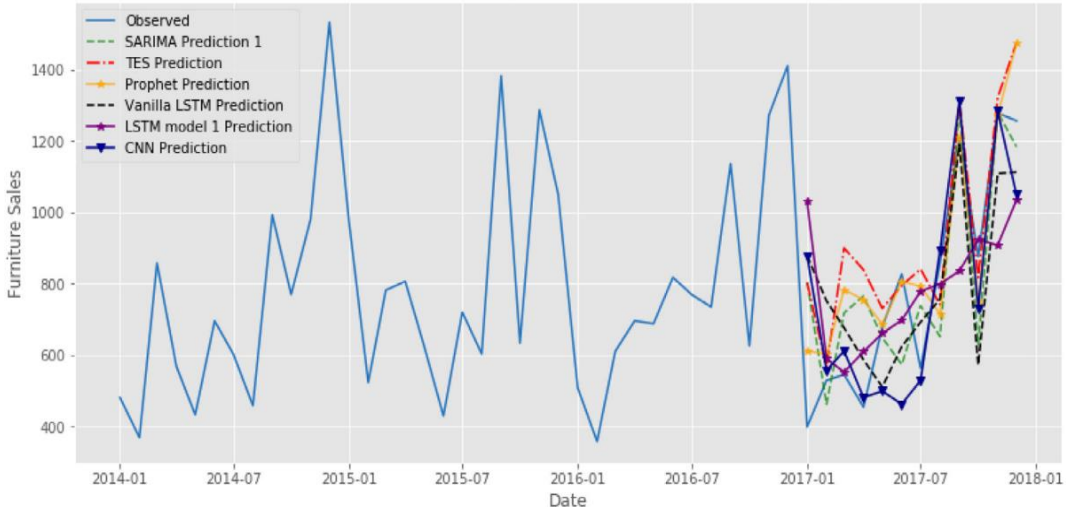
These findings open up new possibilities for leveraging these modern forecasting methods. They hint at the potential of combining LSTM and CNN models or fine-tuning them to suit specific datasets. This not only provides fresh perspectives on predictive modeling but also encourages further exploration in future research. The application of advanced forecasting methods like LSTM and CNN, possibly in conjunction with other models or tailored to the dataset in question, holds promise for delivering exceptional results in future predictive analyses.

Table 1. *Errors for each model original article*

Model	MSE	RMSE	MAPE
ARMA	87,237.01	295.36	33.88
ARIMA	79,804.31	282.50	35.07
SARIMA 1	42,305.37	205.68	28.89
SARIMA 2	55,497.86	235.58	33.50
DES	40,4596.36	636.08	98.79
TES	49,846.48	223.26	30.81
Prophet1	37,992.52	194.92	26.67
Prophet2	27,986.56	167.29	22.62
Vanilla LSTM	18,829.66	137.22	18.39
Stacked LSTM	16,515.49	128.51	17.34
Bidirectional LSTM	53,981.32	232.34	31.40
LSTM 1	68,671.50	262.05	29.40
CNN	39,938.47	199.85	22.26

In **Table 1** is possible to observe all the estimated errors obtained by the authors of the article showing that Stacked LSTM is the model with the best performance, followed by Vanilla LSTM and Prophet. [6]

Figure 1. *Different models comparison*

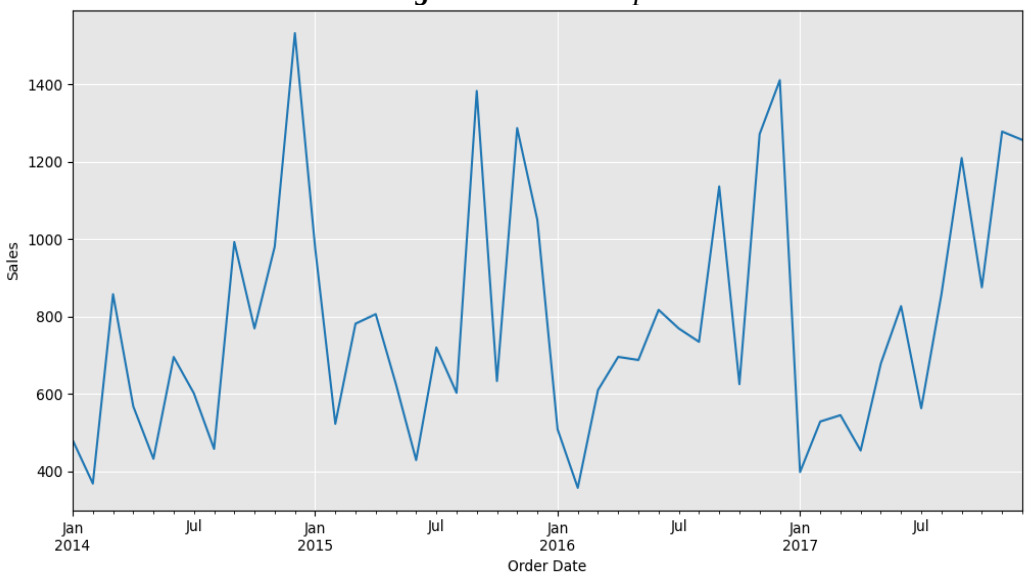


In **Figure No. 1** It is possible to see a graphical comparison of all the obtained results of used forecasted methods by the authors. [6]

Empirical Research

The objective of this project is to replicate the findings of the original article using Python as the primary programming language and the same dataset for all calculations and plots. The initial code utilizes Pandas and Matplotlib libraries to analyze and visualize sales data from the "Sample_Superstore.xls" dataset [12] Once the dataset is loaded, it is filtered to include only the 'Furniture' category. The 'Order Date' and 'Sales' columns are then extracted, with the 'Order Date' being designated as the index. Next, the data is grouped by 'Order Date' and the daily sales are summed. The data is then resampled to a monthly frequency using the mean, resulting in a time series of monthly furniture sales. This initial code serves as a crucial step for conducting further analysis, such as time series forecasting or trend exploration. Additionally, it enables the creation of visualizations that aid in comprehending the sales patterns over time.

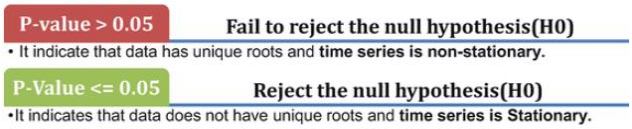
Figure 2. *Initial Graph*



In **Figure No. 2** It is possible to observe a time series plot illustrating the monthly sales trend of furniture products over the provided dataset period.

The Dickey-Fuller test is a statistical test commonly used to determine the stationarity of a time series. Stationarity is a crucial concept in time series analysis, where a stationary time series has constant statistical properties over time, while a non-stationary time series exhibits trends or seasonality and has changing statistical properties. For this test and most of the forecasting models in this project, statsmodels library is used. The results of the Dickey-Fuller test are reported with test statistics, p-value, and critical values, which are used to determine whether to reject the null hypothesis that the time series is non-stationary.

Illustration 1. *P-Value Interpretation*



[18]

Test Statistic: This is a negative number. The more negative it is, the stronger the evidence against the null hypothesis. The obtained value is -5.19.

p-value: This is the probability of observing a test statistic as extreme as the one computed from the data, assuming the null hypothesis is true. A small p-value (usually less than 0.05) indicates that the null hypothesis can be rejected. The p-value obtained is very close to zero (9.16e-06 or approximately 0), suggesting strong evidence against the null hypothesis.

Critical Values: These are values that can be compared with the test statistic to determine whether to reject the null hypothesis. The critical values are specific thresholds for different confidence levels (1%, 5%, and 10%). If the test statistic is more extreme than these critical values, the null hypothesis can be rejected. In the project, the critical values obtained are as follows:

1%: -3.621
5%: -2.943
10%: -2.610

In this case, the test statistic is much smaller (more negative) than the critical values for all confidence levels, and the p-value is very close to zero. Therefore, it is possible to reject the null hypothesis. Rejecting the null hypothesis in this context is good news because it suggests that the time series is likely stationary, which is often a prerequisite for various time series analysis techniques.

The sales data is also analyzed by calculating the rolling mean and rolling standard deviation over a 12-month period. These calculations help identify trends and seasonality in the data, which are then visualized in **Figure 3**

Figure 3. *Rolling Mean & Standard Deviation*

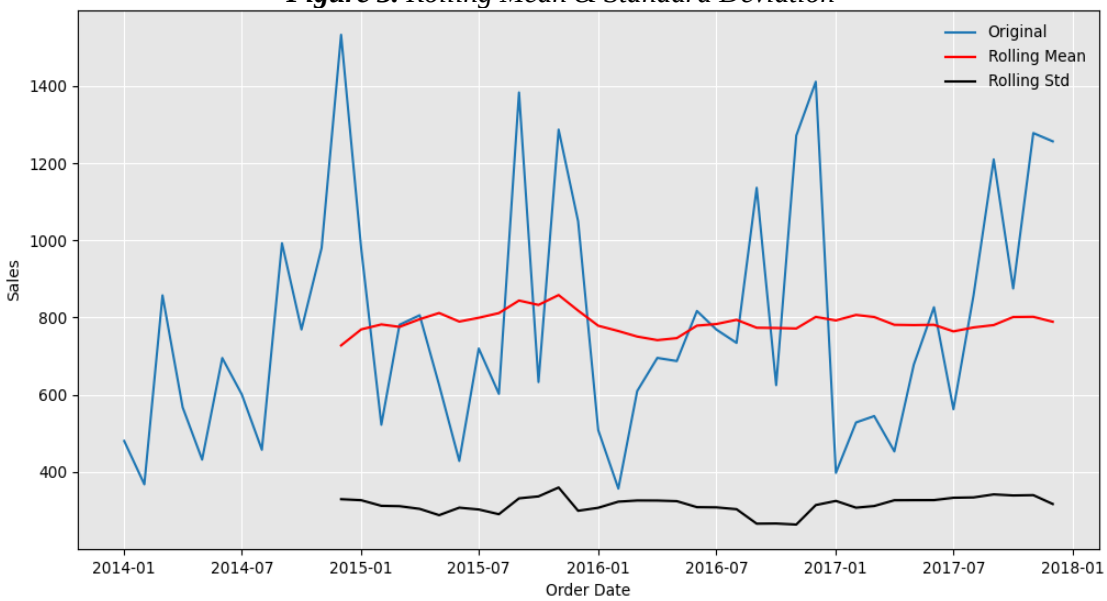


Figure No. 3 Rolling mean and rolling standard deviation are computed over a 12-month window.

ARMA ARIMA & SARIMA

ARIMA, or Autoregressive Integrated Moving Average, stands out as one of the most widely used forecasting methods specifically designed for time series data. ARIMA can be applied to gain insights into predictions within a time series context. The three primary types of ARIMA predictions are:

- ***ARMA***, or Autoregressive Moving Average, is characterized by just two distinct integers: Autoregressive (AR) and Moving Average (MA), denoted as (p, q). In the Python environment, specifically when implementing it with the statsmodel library, the parameter "d" is set to zero. This configuration is crucial for accurately defining ARMA models and their components in a Python environment [14].

- ***ARIMA***, or Autoregressive Integrated Moving Average, is characterized by three essential parameters: the Autoregressive part of the model denoted as "p," the Integrated part as "d," and the Moving Average as "q." To determine the optimal values for these parameters, a Grid Search approach is employed. Through Grid Search, various combinations of (p, d, q) are explored to identify the configuration that yields the lowest error indicators, such as MAPE or MSE. This systematic approach ensures that the ARIMA model is fine-tuned to provide the most accurate and reliable predictions for a given time series dataset [14].

- ***SARIMA***, standing for Seasonal Autoregressive Integrated Moving Average, is denoted as SARIMA (p, d, q) (P, D, Q), where (p, d, q) pertains to the stationary description, and (P, D, Q) represents the seasonal components. Similar to the standard ARIMA model, a Grid Search methodology can be applied in SARIMA for improved parameter estimation. This involves systematically exploring various combinations of seasonal and non-seasonal parameters (P, D, Q) to identify the configuration that minimizes error indicators, such as AIC (Akaike Information Criterion) and BIC (Bayesian Information Criterion). By employing this approach, SARIMA can be fine-tuned to effectively capture both non-seasonal and seasonal patterns in time series data [14].

Figure 4. *Combinations of ARMA, ARIMA and SARIMA*

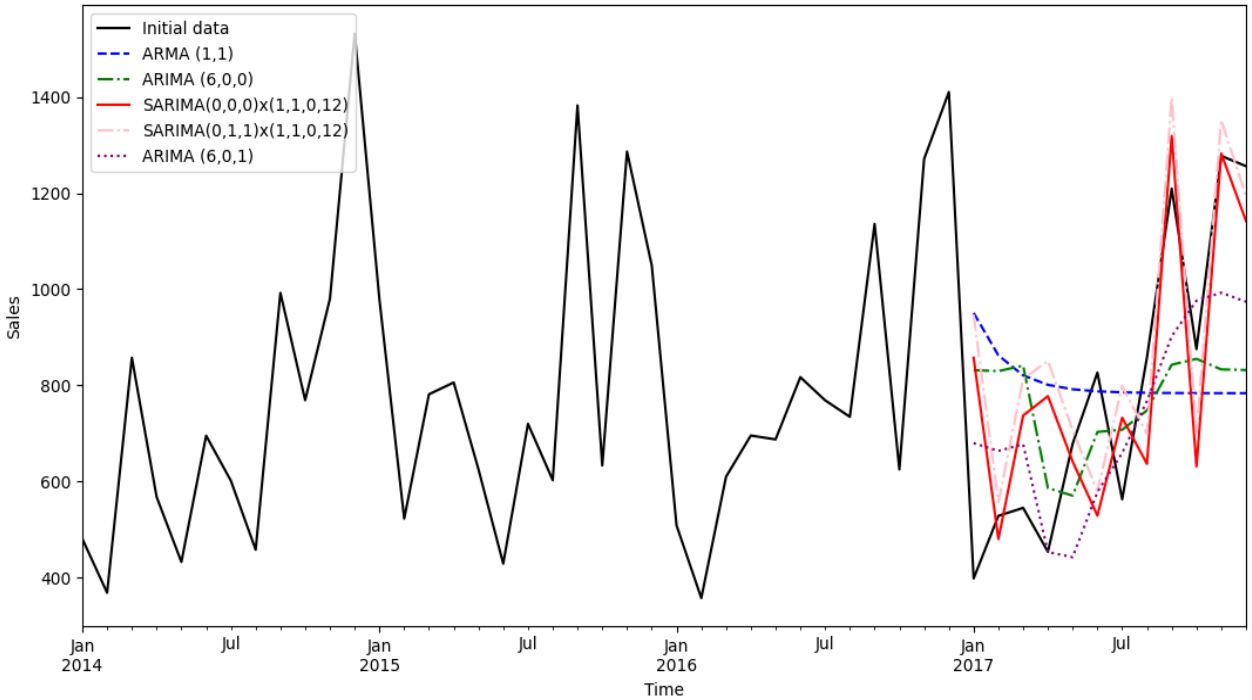


Figure No. 4 Combination of all estimated parameters for ARMA, ARIMA and SARIMA presented on one graph.

Figure 5. *Estimated forecast using SARIMA method.*

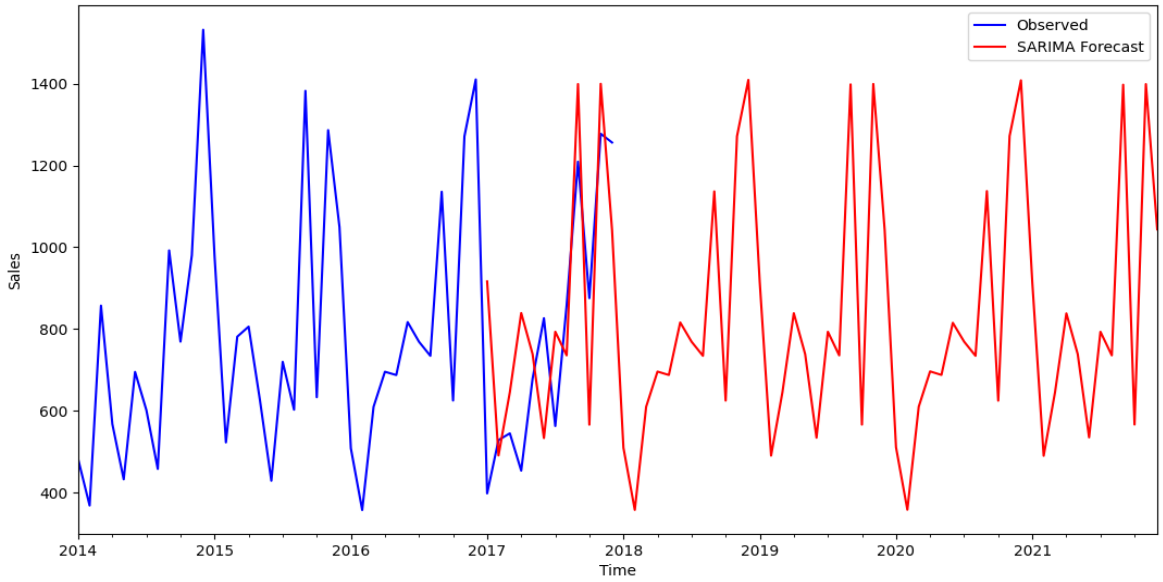


Figure No. 5. Forecast for next four years using SARIMA prediction method.

Simple, Double and Triple Exponential Smoothing (ETS, DES & TES)

Exponential smoothing is a method employed to smooth time-series data by applying an exponential window function. This technique proves valuable in forecasting time-series data that exhibit both seasonality and trend patterns. [1]

This model examines the prediction of future sales for the 'Furniture' category using three different Exponential Smoothing models, Single Exponential Smoothing (SES), which is define by the following equation:

$$\hat{y}_{t+1} = \hat{\alpha}\, y_t + (1-\hat{\alpha})\, \hat{y}_t \qquad (1)$$

Where α is the exponential smoothing factor, it shapes forecasting by determining the weight given to current observations. A lower α prioritizes historical values, while a higher α emphasizes the current value for a more adaptive forecast. This factor controls the balance between integrating new information and preserving stability in the forecasting process. This model is usually applied for forecasting univariate data without a trend or seasonality.

Holt's linear trend model (DES), is an extension of simple exponential smoothing that includes an additional parameter to capture and forecast trends in time series data, this model has two smoothing equations [2]:

$$\hat{y}_{t+1} = \alpha\, y_t + (1-\alpha)\,(\hat{y}_t + b_t) \qquad (2)$$

$$b_{t+1} = \beta(\hat{y}_{t+1} - \hat{y}_t) + (1-\beta) b_t \qquad (3)$$

Where β is the trend smoothing factor, its role is to adjust the influence of the current trend on future predictions. A higher β increases sensitivity to recent trend changes, while a lower β reduces sensitivity.

Finally Triple Exponential Smoothing with a linear seasonality (TES), which allow to capture seasonality with level and trend, the model is define by the following equations [2]:

$$\hat{y}_{t+1} = \alpha\,(y_{t+1} - S_t) + (1-\alpha)\,(\hat{y}_t + b_t) \qquad (4)$$

$$b_{t+1} = \beta(\hat{y}_{t+1} - \hat{y}_t) + (1-\beta)(b_t) \qquad (5)$$

$$S_t = \gamma(\hat{y}_t - \hat{y}_t) + (1-\gamma) S_{t-C} \qquad (6)$$

Where γ is seasonality smoothing factor, controls the smoothing of the seasonal component, helping the model adapt to repeating patterns in the time series data. By utilizing monthly sales data from the "Sample Superstore"dataset, the models are trained, validated, and evaluated using performance metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error

(MAPE). Two different approaches were employed to develop the models: the first approach utilized the 'statsmodel' library in Python, while the second approach involved creating custom functions based on the provided equations, for the TES the function was based on the provided in the book 'Practical Time-Series Analysis'[1], all results are visually presented through a plot that compares the forecasted values with the actual sales, providing a comprehensive assessment of the accuracy of the models. (**Figure 6** & **Figure 7**)

Figure 6. *SES, DES, & TES with library*

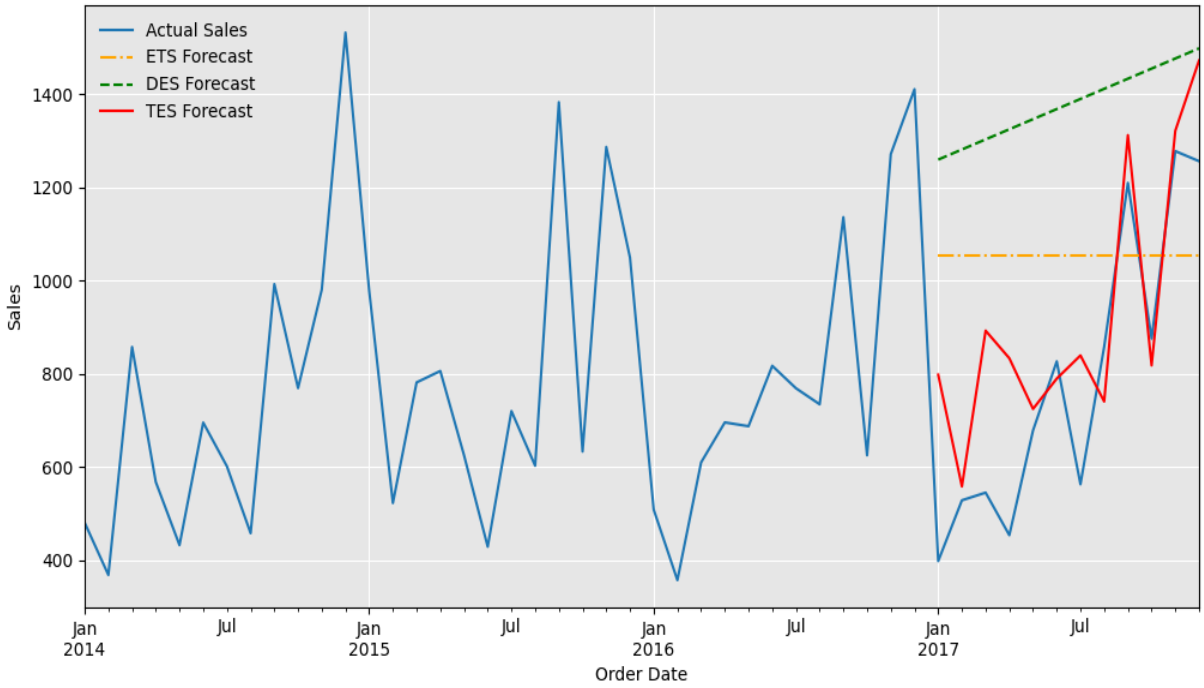


Figure No. 6 is possible to observe the result using 'statsmodels.tsa.holtwinters' module from the Statsmodels library and its classes 'SimpleExpSmoothing' for the SES and 'ExponentialSmoothing' for DES and TES.

Figure 7. *Simple ETS, DES and TES own implementation*

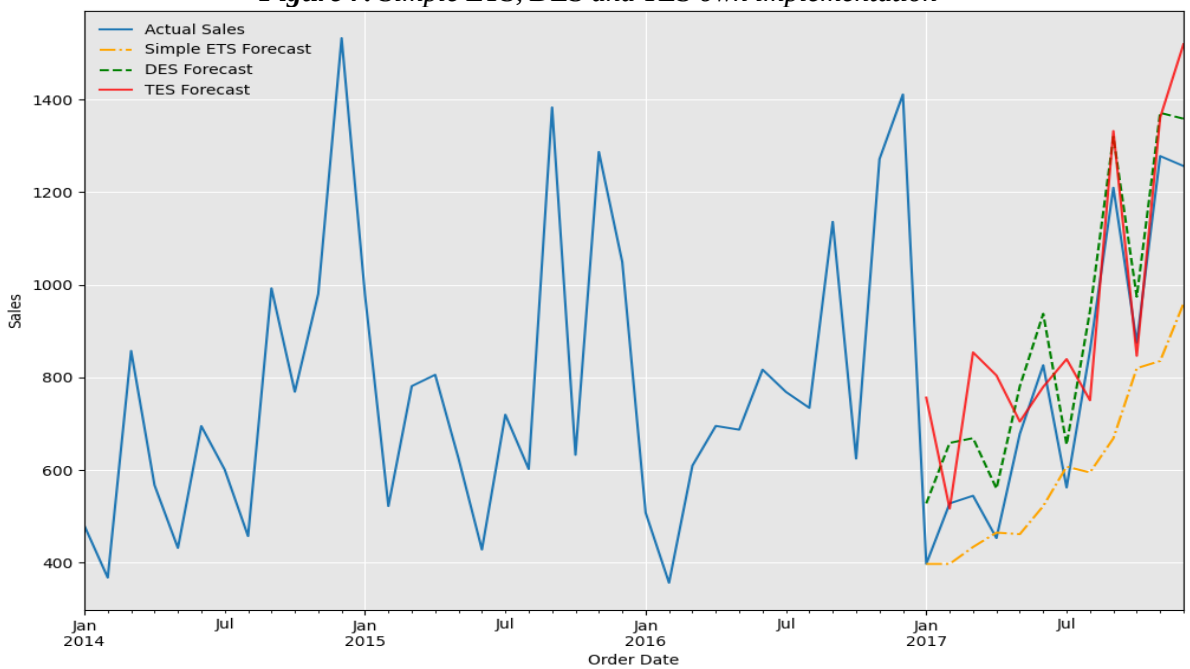


Figure No. 7 The plot illustrating the execution of Simple ETS, DES and TES has been acquired using custom functions derived from the provided formulas.

Meta Prophet Model

The Prophet forecasting model, developed by Facebook's Meta company under the leadership of Sean J. Taylor and Benjamin Letham, is a powerful tool for predicting future values in time series data based on historical patterns. This open-source algorithm is designed to handle various time series data characteristics, providing a flexible and user-friendly approach to forecasting.

Prophet excels in capturing different time-related patterns, incorporating yearly, weekly, and monthly seasonality effects, as well as considering the impact of holidays [5]. Prophet is an open-source algorithm that combines three essential functions to generate accurate forecasts. The first component focuses on modeling the underlying trend through a sophisticated linear regression model. This approach assumes that the overall trend is composed of distinct linear segments, with each segment's slope indicating the direction in which the trend is evolving at specific points in the time series. This trend modeling can be conceptualized as a function [10].

$$y(t) = g(t) + e(t) \quad (18)$$

y(t) - time series value
g(t) - trend component
e(t) - error

In Prophet's Seasonality Modeling, they use a clever mathematical technique called Fourier coefficients within a Bayesian model. This technique, based on the Fourier series, allows the model to effectively capture various recurring patterns—whether they happen weekly, monthly, or yearly. By using Fourier coefficients, the model becomes quite flexible in understanding and predicting different types of seasonal behaviors in the data [10].

Figure 8. *Prophet with including and without holidays.*

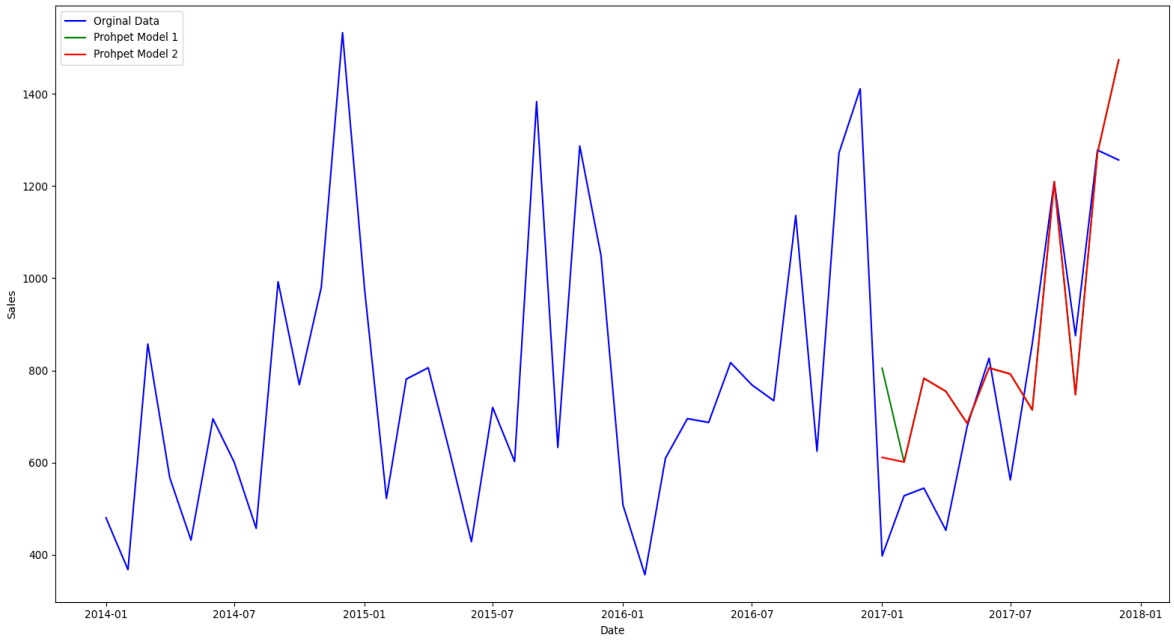


Figure No. 8 the visual representation of the Prophet model showcases the influence of yearly seasonality, with included holidays highlighted in red color.

Long Short-Term Memory Networks (LSTM)

For this part of the project Keras library was used to implement three different LSTM (Long Short-Term Memory) neural network architectures for time series prediction. Once the data is treated and divided into training and test sets, it is preprocessed using Min-Max scaling to transform it into the range [-1, 1]. The TimeseriesGenerator is employed to convert the univariate time series data into a supervised learning problem, preparing it for training the LSTM models.

The Vanilla LSTM model is implemented in a loop (repeated 20 times) to account for the stochastic nature of neural network training. The architecture consists of a single LSTM layer with 50 units, followed by two additional Dense layers. The model is trained using the Adam optimizer and Mean Squared Error (MSE) loss function. Predictions are made iteratively for the next 12 months, and the results are stored in a 2D array. Finally, the mean of the predictions for each month is calculated, and error metrics (MSE, RMSE, MAPE) are computed.

Like the Vanilla LSTM, the Stacked LSTM model is implemented in a loop. However, this architecture includes two LSTM layers with 50 units each, followed by additional Dense layers. The training and prediction process is repeated, and the mean of predictions along with error metrics are calculated as well.

The Bidirectional LSTM model is a variation where the LSTM layer processes the input sequence in both forward and backward directions. The model is defined with one Bidirectional LSTM layer with 20 units and a Dense output layer. Training is conducted using the generator, predictions are made, and error metrics are calculated as with the previous 2 models. The forecast obtained can be visualized in **Figure 9**.

Figure 9. *LSTM Models*

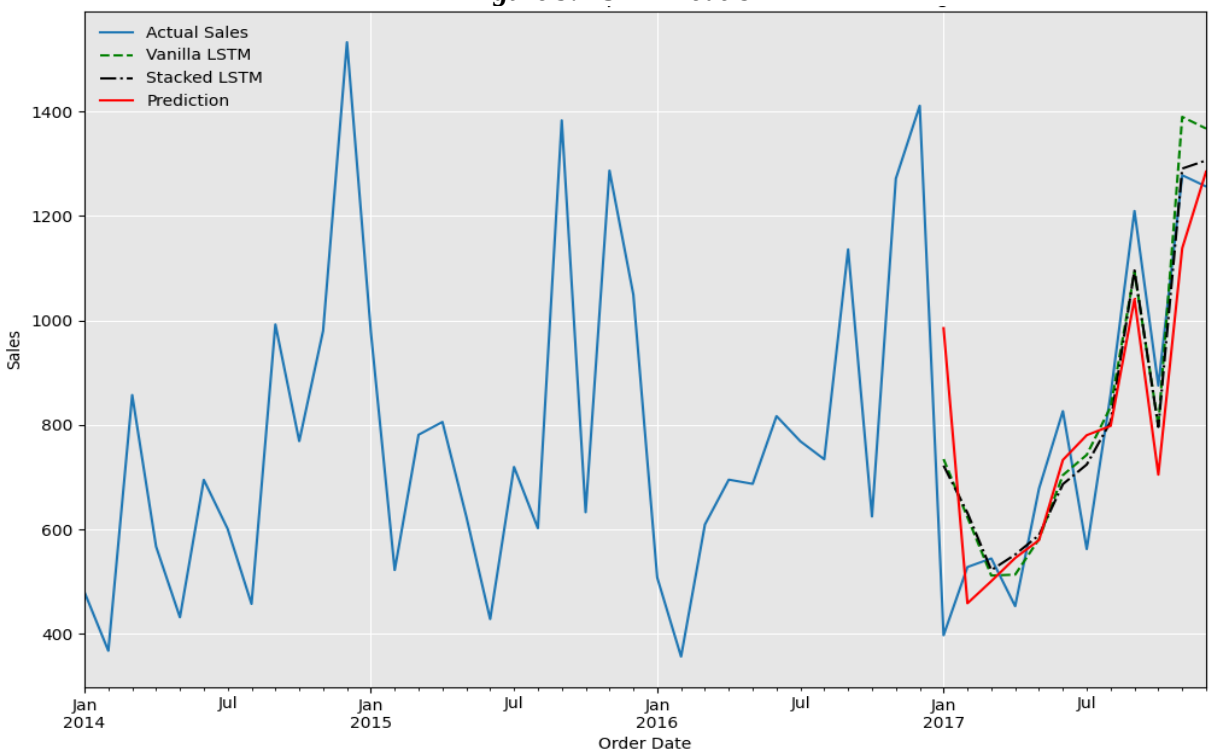


Figure No. 9 illustration of Vanilla, Stacked and Bidirectional LSTM models.

Convolutional Neural Network (CNN)

The Convolutional Neural Network (CNN) finds its primary application in two-dimensional scenarios, specifically for image classification and object recognition. CNNs are trained for these tasks, with 1D and 3D CNNs also available, adapting to the dimensions required for the intended execution and data type. It's essential to note that not all data types are suitable for CNN applications [17]. One-dimensional CNNs, for instance, prove effective when seeking patterns within shorter data segments. Take, for example, an accelerometer and gyroscope dataset, where patterns in 1D graphs can determine the person's movement state—whether stationary, jogging, or ascending stairs [11].

In this article, we employ a 1D CNN network to forecast furniture sales patterns. TensorFlow with the Keras library facilitates the creation of this forecast. Initially, the data is divided into a training set and evaluation data, with 75% assigned to the training set. Metrics such as MAPE, MSE, and RMSE are then calculated. The training data undergoes further segmentation using a rolling window, breaking it into smaller data portions stored in arrays. A sequential model is constructed based on this, featuring three one-dimensional convolutions with 128 filters. Following each convolutional layer, a max-pooling layer identifies the highest value within the window, reducing the number of places in the matrices. After flattening, converting the feature map into a single column, the model is compiled, fitted, and ready for forecasting [13]. The entire forecasting and modeling process is repeated ten times, and the result is the average of all iterations of the created forecasts. The model achieved an accuracy range of 23% to 25%, as measured by the Mean Absolute Percentage Error (MAPE) metric. The graphical representation illustrating the CNN's effectiveness in identifying patterns is presented in the figure below (**Figure 10**).

Figure 10. *CNN Forecasting method*

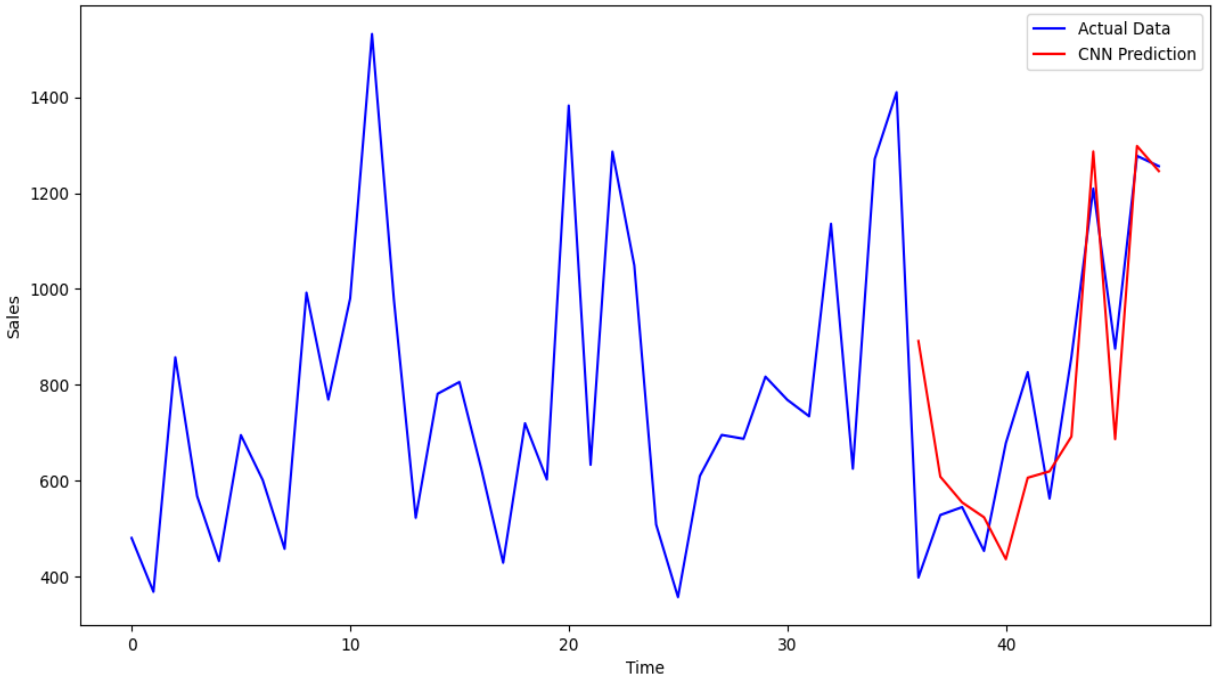


Figure No. 10 illustration of 1D CNN prediction on “X axis” weeks and on “Y axis” Sales

Identifying Optimal Configuration: Grid Search of all models for Minimizing MAPE

In the final phase of this study, a Grid Search algorithm was implemented to systematically iterate through all acquired results. This step aimed to establish an average performance metric across various forecast combinations. The Grid Search determined the optimal result by evaluating the Mean Absolute Percentage Error (MAPE) for each interconnected model, thus pinpointing the most accurate forecasting configuration. The most favorable outcomes were achieved through a customized implementation combining ETS and DES. This strategic amalgamation resulted in a noteworthy reduction of the MAPE to an impressive 9.6%, representing the lowest MAPE value across all implemented forecast models.

Figure 11. *Combined ETS&DES from own implementation*

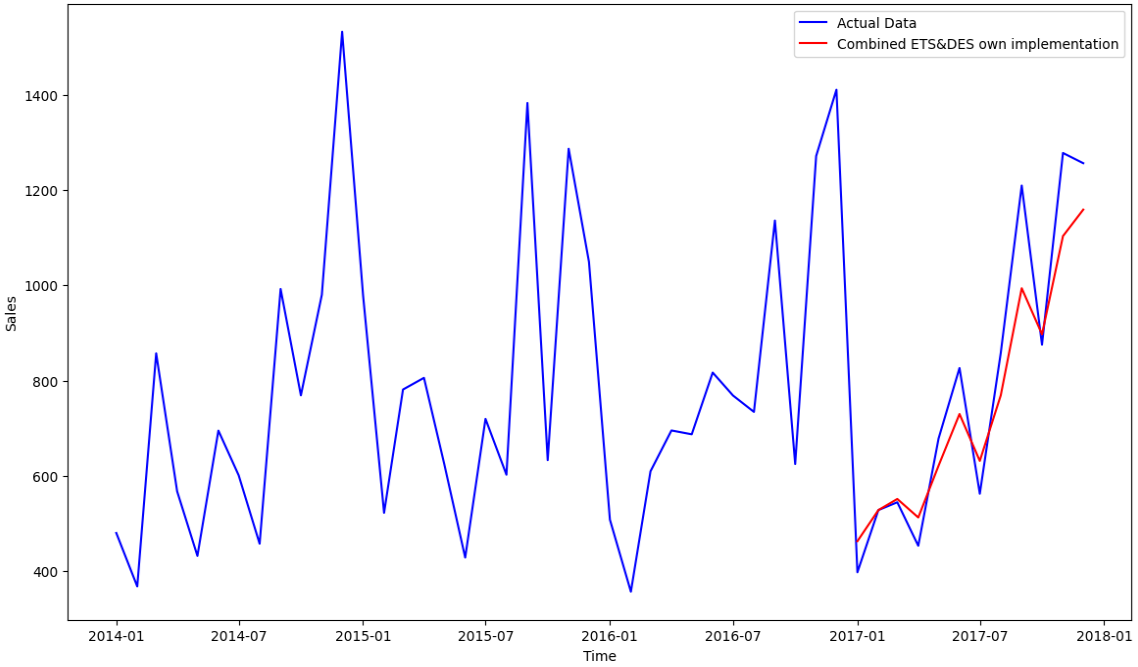


Figure No. 11 Combination of ETS and DES from own implementation. The amalgamation of these two forecasts, achieved through averaging.

RESULTS

ETS, DES & TES

The errors observed in the obtained results are similar to those in the original article, but not identical. This discrepancy may be attributed to variations in implementation, such as the selection of smoothing levels, data preparation techniques, or random factors. Based on the results obtained from the self-constructed functions, it can be concluded that the ETS model exhibits higher values for MSE, RMSE, and MAPE compared to DES and TES. This indicates that ETS is not the most suitable model for seasonality data. Additionally, DES and TES demonstrate similar MSE values, but TES outperforms DES in terms of lower RMSE and MAPE. Therefore, based on these metrics, TES appears to be the preferred model. In all cases, TES demonstrates the best performance for the dataset.

ARMA, ARIMA & SARIMA

For ARMA, although the MAPE results differ, they consistently convey the same directional trend graphically representing the increase in furniture sales. For ARIMA, two sets of results were obtained. The first aligns with the implementation by the original authors, while the second stems from parameter optimization via grid search. This process led to the discovery of different parameters than those proposed by the authors, resulting in a notable reduction in MAPE. Both SARIMA models yield comparable results.

LSTM

The training process of neural network algorithms involves initializing random weights, resulting in different outcomes even when using the same network and data. As a result, replicating the exact results from the original article is not possible, although the errors are close to the values presented in the original article.

Based on the results the Stacked LSTM appears to be the most effective model among the three, providing the lowest error metrics and better capturing the patterns in the data. The Vanilla LSTM also performs reasonably well but is outperformed by the Stacked LSTM. Finally, the Bidirectional LSTM, seems to be less suitable for achieving accurate predictions. This result aligns with the result obtained in the original article.

PROHET

Both Prophet models, one incorporating holidays and the other without, exhibit identical results to those reported by the authors of the article.

CNN

In the case of CNN and LSTM, results may vary in each attempt, making it challenging to precisely replicate values reported by the authors in the article. However, leveraging this understanding of the inherent variability in our CNN model, we can establish a range by observing error metrics such as MSE and MAPE. Our findings indicate that the model's MAPE falls within the range of 22-25%. Notably, we enhanced robustness by creating ten different CNN models, averaging their results, and ultimately obtaining a consistent range of MAPE errors.

COMBINE FORECAST

ETS and DES models capture distinct elements of time series patterns. ETS considers error, trend, and seasonality, whereas DES primarily focuses on the trend. By leveraging the strengths of both models, a more reliable and precise forecast can be achieved. The combination of these two models exhibits the lowest errors, (**Table 2**) making it a promising forecasting approach for the given dataset.

Table 2. *Final errors for the different models*

Model	MSE	RMSE	MAPE
ARMA (1,1)	111780.56	334.34	43.44
ARIMA (6,0,0)	79779.20	282.45	35.04
ARIMA (6,0,1)	43018.80	207.41	24.61
SARIMA 1	50711.00	225.20	31.30
SARIMA 2	62702.21	250.04	35.14
ETS (using library statsmodels)	162533.84	403.15	62.60
ETS (own implementation)	68279.35	261.30	22.02
DES (using library statsmodels)	404988.29	636.39	98.84
DES (own implementation)	11717.68	108.25	16.23
TES (using library statsmodels)	48529.78	220.29	30.31
TES (own implementation)	12352.55	111.14	10.59
Prophet 1	37992.43	194.92	26.67
Prophet 2	27986.66	167.29	22.62
Vanilla LSTM	19091.40	138.17	18.50
Stacked LSTM	17051.48	130.58	17.95
Bidirectional LSTM	42324.52	205.73	25.61
CNN	38612.86	195.40	23.75
Combine Forecast (ETS – DES)	9996.89	99.98	9.60

Bibliography

1. Auffarth, B. (2021). *Machine Learning for Time-Series with Python.* Packt Publishing. Retrieved from https://learning.oreilly.com/library/view/machine-learning-for/9781801819626/
2. Avishek, P., & Prakash, P. (2017). *Practical Time-Series Analysis.* Packt Publishing. Retrieved from https://learning.oreilly.com/library/view/practical-time-series-analysis/9781788290227/
3. Brownlee, J. (2020, April 12). *A Gentle Introduction to Exponential Smoothing for Time Series Forecasting in Python*. Retrieved from Machine Learninig Mastery: https://machinelearningmastery.com/exponential-smoothing-for-time-series-forecasting-in-python/
4. Brownlee, J. (2020, August 28). *How to Develop LSTM Models for Time Series Forecasting*. Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/
5. Duong, C. (2023, October 18). *Prphet Facebook*. Retrieved from Github: https://github.com/facebook/prophet
6. Ensafi, Y., Amin , S. H., Zhang, G., & Shah, B. (2022). Time-series forecasting of seasonal items sales using machine learning – A comparative analysis. *International Journal of Information Management Data Insights*, 16. Retrieved from https://www.sciencedirect.com/science/article/pii/S2667096822000027
7. Jain, A. (2023, May 2). *A Comprehensive Beginner's Guide to Creating a Time Series Forecast (With Codes in Python and R)*. Retrieved from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2016/02/time-series-forecasting-codes-python/
8. Joseph, M. (2022). *Modern Time Series Forecasting with Python.* Packt Publishing. Retrieved from https://learning.oreilly.com/library/view/modern-time-series/9781803246802/
9. Ketkar, N., & Moolayil, J. (2021). *Deep Learning with Python: Learn Best Practices of Deep Learning Models with PyTorch.* Apress. Retrieved from https://learning.oreilly.com/library/view/deep-learning-with/9781484253649/
10. Khare, P. (2023, May 13). *Understanding FB Prophet: A Time Series Forecasting Algorithm*. Retrieved from Medium: https://medium.com/illumination/understanding-fb-prophet-a-time-series-forecasting-algorithm-c998bc52ca10
11. KURŞUN, F. (2022). *Predict Sales(time series with CNN)*. Retrieved from Kaggle: https://www.kaggle.com/code/fatmakursun/predict-sales-time-series-with-cnn/notebook
12. Martin, M. (2017, November 11). *Sample - Superstore*. Retrieved from Tableau Community: https://community.tableau.com/s/question/0D54T00000CWeX8SAL/sample-superstore-sales-excelxls
13. MWITI, D. (n.d.). *How to build CNN in TensorFlow: examples, code and notebooks*. Retrieved from cnvrg.io: https://cnvrg.io/cnn-tensorflow
14. Ocean, D. (2017, March 23). *A Guide to Time Series Forecasting with ARIMA in Python 3*. Retrieved from Thomas Vincent: https://www.digitalocean.com/community/tutorials/a-guide-to-time-series-forecasting-with-arima-in-python-3
15. Purkait, N. (2019). *Hands-On Neural Networks with Keras.* Packt Publishing. Retrieved from https://learning.oreilly.com/library/view/hands-on-neural-networks/9781789536089/
16. Svetunkov, I. (2023). *Forecasting and Analytics with the Augmented Dynamic Adaptive Model*. Retrieved from Open Forecast Org: https://openforecast.org/adam/SES.html
17. Verma, S. (2019, September 20). *Understanding 1D and 3D Convolution Neural Network | Keras*. Retrieved from Medium: https://towardsdatascience.com/understanding-1d-and-3d-convolution-neural-network-keras-9d8f76e29610
18. Vishwas, B. V., & PATEL, A. (2020). *Hands-on Time Series Analysis with Python: From Basics to Bleeding Edge Techniques.* Apress. Retrieved from https://learning.oreilly.com/library/view/hands-on-time-series/9781484259924/