Project 3: Time Series Forecasting

Decision support systems

by group 9:

Daniel Christopher Biørrith, 201909298

Jeppe Stjernholm Schildt, 201905520

Aarhus University

Department of Electrical and Computer Engineering

May 15, 2023

Contents

1 Introduction

With the rapidly increasing availability to data, the need for accurate forecasting algorithms is bigger than ever. Time-series forecasting is crucial in many industries, such as finance, retail and healthcare. Being able to accurately predict the price of stocks, or the demand for certain items, results increased profitability and efficiency. Predicting financial trends is the focus point of this report, more specifically the price of the Yahoo stock between 2015 and 2021. Throughout, the implementation and performance of multiple well-known forecasting algorithms is presented, with a focus on simplicity and optimizing accuracy. To ensure a non-biased evaluation of the algorithms, objective metrics are utilized, such as the Mean Absolute Percentage Error (MAPE). By conducting a systematic comparison between these forecasting algorithms, this project aims to provide insight into their relative strengths and weaknesses, as well as ultimately selecting the best method for this particular data-set.

2 Methods and materials

The data-set chosen for this project is the Yahoo Stock price data-set[1], which presents the daily values of the Yahoo stock, from end of 2015 to end of 2020. It has 5 relevant parameters, namely the Opening, Closing, High, Low and Adjacent Closing values. The parameter we wish to predict, is the closing value. In figure 1, this value is presented as a time-series, split into training and validation data at the 1st of January 2020.

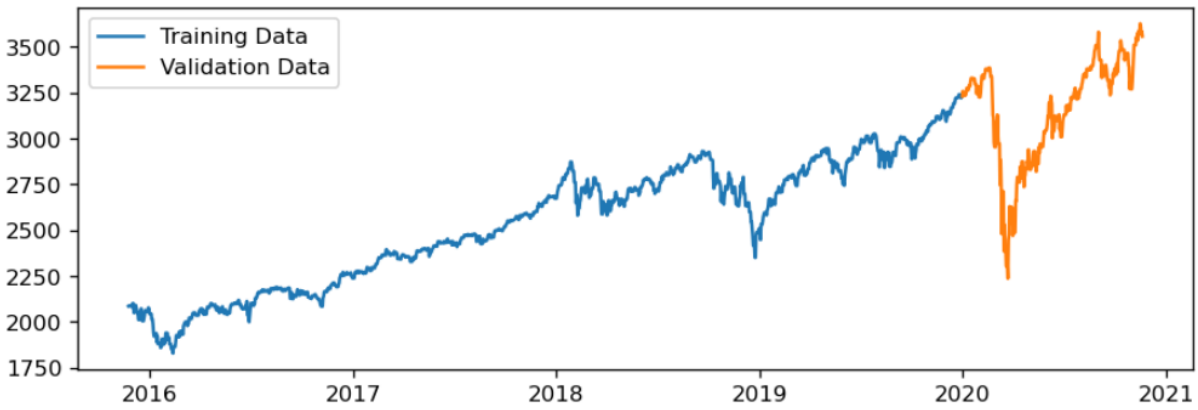


Figure 1: Closing value of Yahoo Stock

2.1 Introductory

The following section will briefly introduce the models selected, and any notable strengths and weaknesses. The implementation will be presented, but results are to be found in section 3.

2.2 ARIMA

The ARIMA model requires careful selection of the p, d, q parameters based on the characteristics of the time-series data. Choosing these values poorly may result in very inaccurate forecast. The parameters are defined as follows[2]:

- p (AR) : the number of auto-regressive terms, in essence incorporating the effect past values of the time series has on the current value.
- d (I) : the number of differences required to make the data stationary, i.e removing trend and seasonality.

- q (MA): the number of moving averages, incorporating the effect of past forecast errors on current value.

2.2.1 Selecting parameters

First, we find our value for d, as that will help us find our other values subsequently. Initially we test for stationarity of the data, using the Augmented Dickey-Fuller test[3]. This is done using the 'adfuller' function from statsmodels[4]. Here we get a p value of 0.79, and thus we cant reject the null-hypothesis (on 0.05 significance level) that non-stationary exists. Thus, our data is related to the time, and in need of differencing.
In figure 2, the closing price and the auto-correlation graph (ACF) are shown for the original series, and 2 differencings.

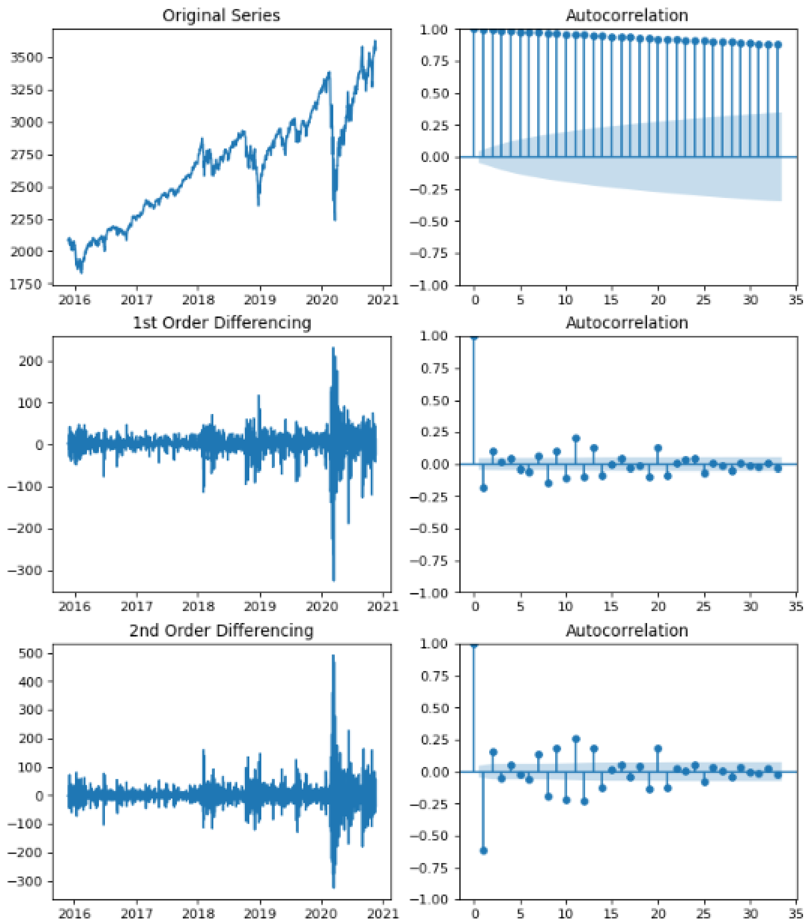


Figure 2: Differencing and corresponding ACF's

It's evident that the data becomes stationary, at least to the naked eye, is there seemingly is no correlation to the x axis. The auto-correlation plot for the 2nd order differencing shows a very negative second lag, which is unwanted and as such we select 1 for the d value.
Now, to find the value of p we inspect the PACF plot for the differenced data. This is seen in figure 3.

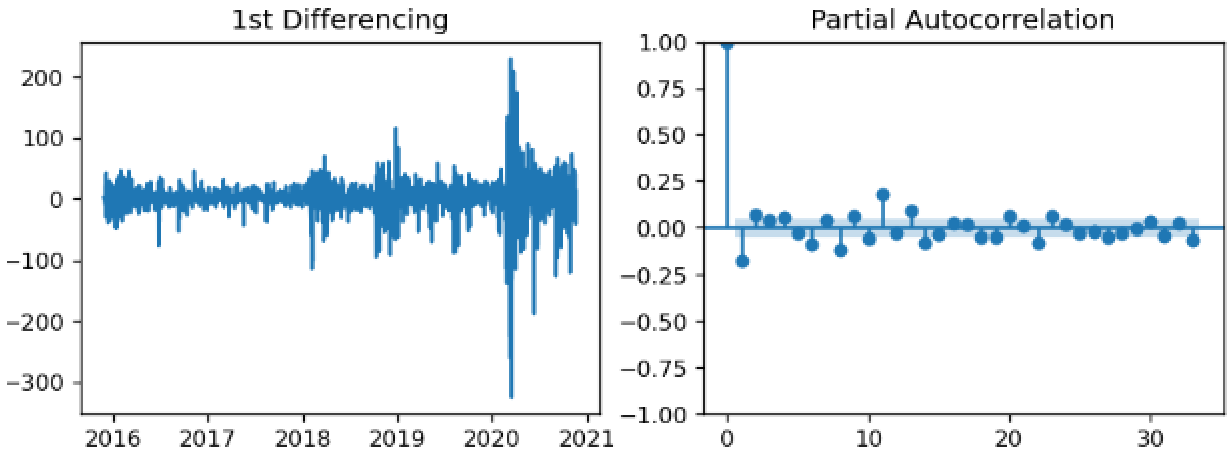


Figure 3: 1st differencing, Series and PACF

Here we see that the first 2 lags are significantly out of the significance limit, and as such we select p as 2.
Lastly, to select q, we once again look at figure 2. Here we see that the auto correlation plot for the 1st order differencing has a lot of similarities with the PACF, and the 2 first lags are also significant outliers, we select q as 2.
This gives us a model order on the form $(p, d, q) = (2, 1, 2)$
To validate our order, we use 'auto arima' from pmdarima[5], which selects the parameters automatically based on the data. This also gives us a model of $(p, d, q) = (2, 1, 2)$, confirming out manual tuning, so we use that for the ARIMA forecasting.

2.3 SARIMA

The core difference between SARIMA, and ARIMA, is that it is able to incorporate seasonality into its predictions, making it more suitable for data which have a seasonal tendency, whether that be daily, weekly, monthly or even yearly. Before implementing this model, the data was examined in an attempt to predict the applicability of that specific algorithm. The seasonal decompose function[6] was applied to a subset, chosen arbitrarily as February 2017, to highlight characteristics of the data.

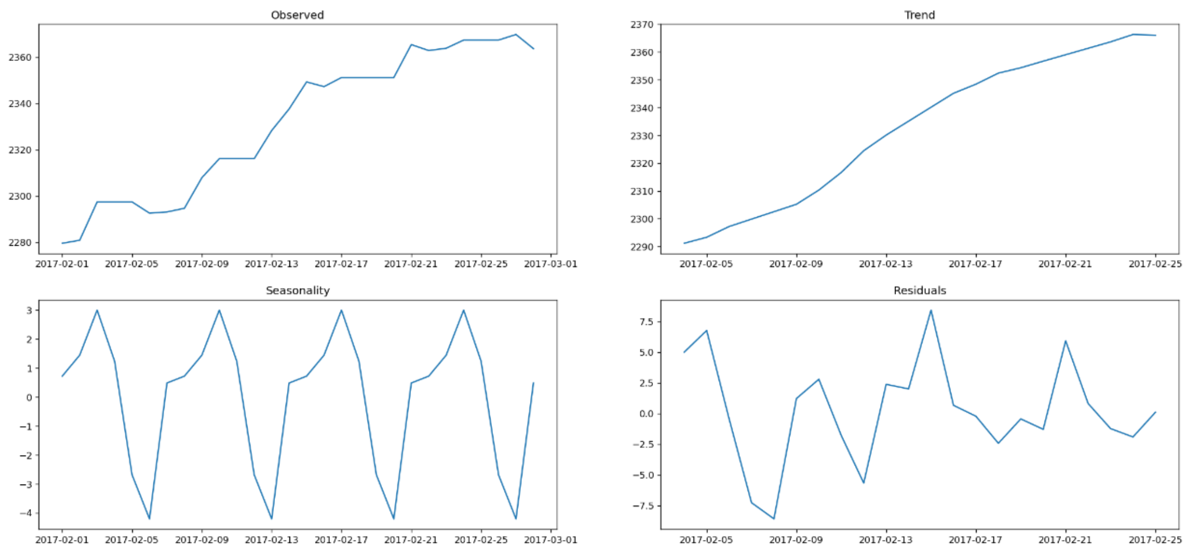


Figure 4: Seasonal decompose of feb. 2017

Here, we can see clear seasonality on a weekly basis, and as such, we select the new parameter for SARIMA, namely the m, as 7, due to the 7 time-steps per period. This gives us a SARIMA model on the form of $(p, d, q)x(P, D, Q)m = (2, 1, 2)x(2, 1, 2)7$

2.4 LSTM

Before creating the LSTM model, we first normalize our timeseries, to ensure a faster converging model and more accurate weights. This is done using the Darts library[7], which is also used to create the LSTM model. This is shown below:

```
lstm_model = BlockRNNModel(model='LSTM',input_chunk_length=28,output_chunk_length=7,
n_epochs=100)
```

Here its intended to use a months values, to predict the next weeks. Different epoch values was attempted but 100 seemed to provide a very good training-time vs performance ratio.

2.5 PROPHET

Prophet, a forecasting algorithm developed by Facebook is an "additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. It works best with time series that have strong seasonal effects and several seasons of historical data."[8]. The algorithm is the easiest to implement of the ones presented in this report, only needing to provide seasonality mode (additive vs multiplicative) and seasonality, which was previously found to be weekly. The implementation is shown below:

```
proph_model = Prophet(seasonality_mode='multiplicative',yearly_seasonality=False,
weekly_seasonality=True,daily_seasonality=False)
```

3 Experiments & Results

This section presents the evaluation of the previously presented algorithms, compared against the validation data. Diagrams of the results will be shown, but objective measurements such as the mean absolute percentage error (MAPE) is also used.

3.1 ARIMA

The graph seen in figure 5 is the prediction made by the ARIMA implementation:

Figure 5: ARIMA forecasting result

Here, it's evident that the prediction is just a linear function, seemingly following the trend of the data. Doesn't seem very accurate, at least from day to day, but could be useful if the perspective was long-term.

3.2 SARIMA

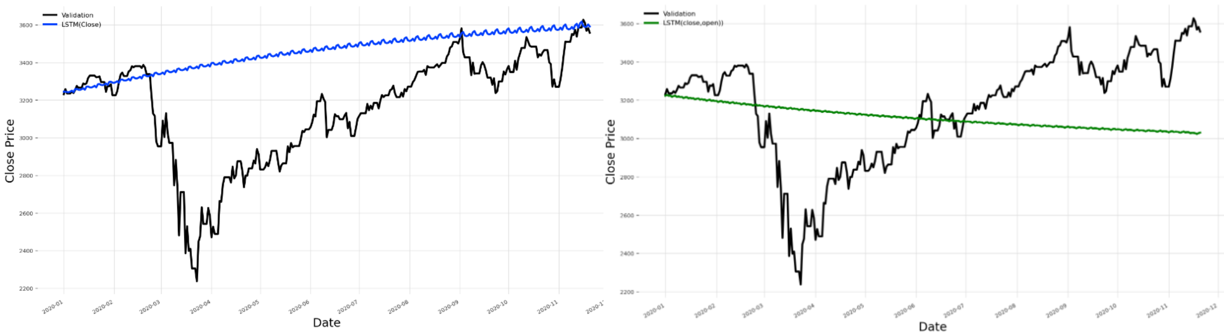
Utilizing the SARIMA model, which incorporates seasonality as previously mentioned, the accuracy should hopefully be better. The result is shown in figure 6:

Figure 6: SARIMAX forecasting result

This model seems to properly incorporate the trend as opposed to the ARIMA model, but its also nearly linear and thus more applicable for long term forecasting than short-term.

3.3 LSTM

Since both ARIMA and SARIMA is uni-variate[9] (and thus only trained on closing value), the testing for LSTM, which can handle multiple variables, is a bit more extensive.
We initially tested the model with purely the 'Close' variable, to ensure proper compatibility with the other models, and then subsequently attempted to also incorporate the 'Open' values in the training. The differences are shown below in figure 7



(a) LSTM trained on 'Close' (b) LSTM trained on 'Close' & 'Open'

Figure 7: LSTM Comparison

This led to the question, does more variables provide better prediction of the closing value, to which an increasing amount of variables was added in the same order as they were introduced in chapter 1, providing the graph(s) found in figure 9

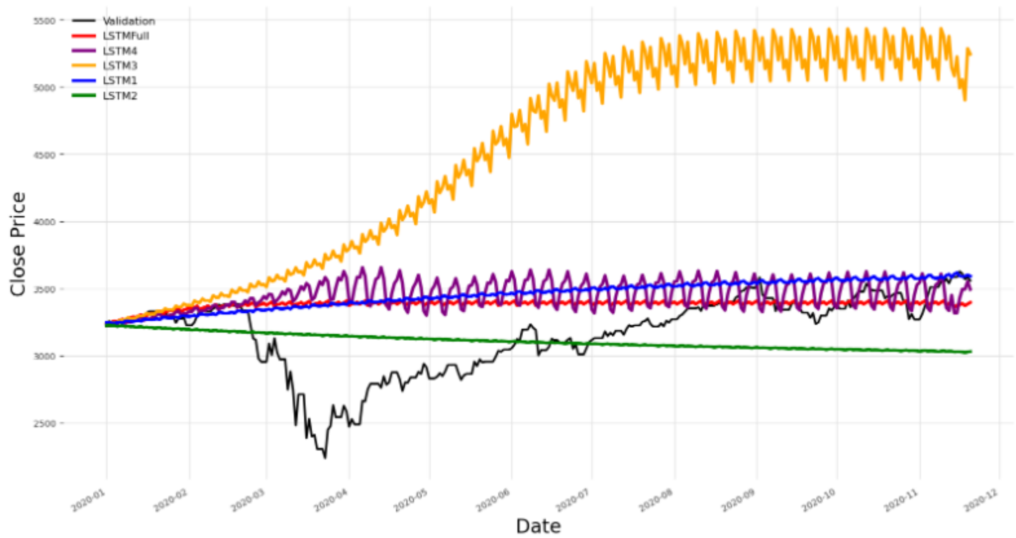


Figure 8: Same LSTM model trained on different variables

While this provides a quite busy picture, it's evident that LSTM trained on 3 values performs very poorly, and the model trained on all models is slightly better than training on only the 'Close' value. Also, the LSTM trained on all values except Adjacent closing, seems to follow the trend of the other 2 good models, but with a much larger seasonality.

3.4 Prophet

Lastly, the performance of the Prophet algorithm, is shown below:

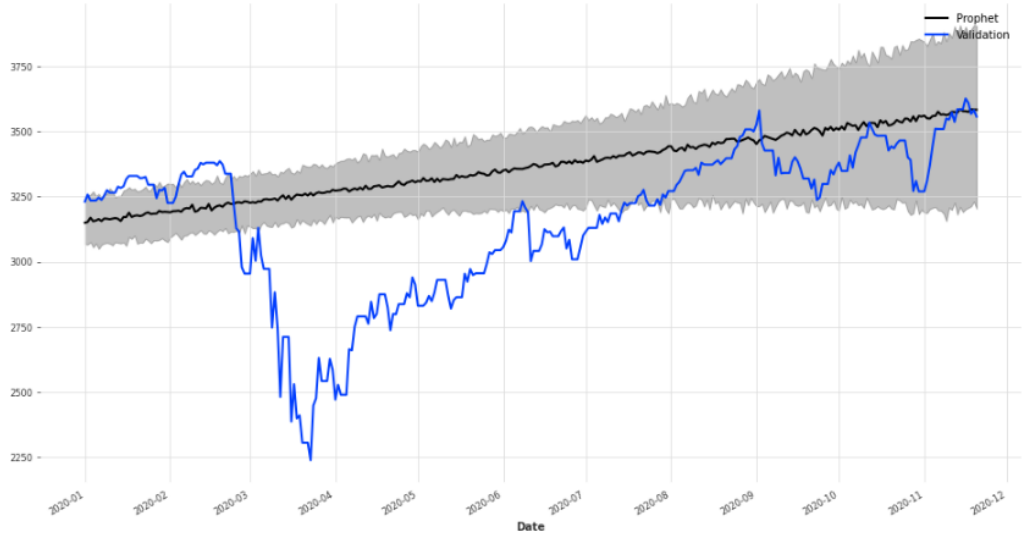


Figure 9: Prophet forecasting accuracy

This shows a model which resembles the SARIMA prediction quite well, which is quite impressive especially considering the ease of implementation, needing no explicitly defined values.

3.5 Metrics

While looking at graphs does provide some information regarding the accuracy of the forecasting, objective ways to qualify the predictions is needed. This report utilizes the MAPE values, calculated using the implementation provided by the earlier-mentioned Darts library.
The values for the different algorithms is presented below:

- ARIMA : 7.60
- SARIMA : 8.12
- LSTM(1) : 10.47
- LSTM(2) : 8.57
- LSTM(3) : 42.73
- LSTM(4) : 11.10
- LSTM(Full) : 9.00
- Prophet : 8.43

4 Discussion

The values found in section 3.5 clearly show a upper limit of the predictability of the Yahoo stock, especially due to the Covid19 crash in 2020 which resulted in a very large stock price drop. The LSTM model made very different predictions based on the training variables, with the one trained on 'Open','Close' and 'High' performing horribly, likely due to the 'High' column influencing the model to predict extremely large values. Furthermore, ARIMA, while not technically suitable for data with seasonality, proved to be the most accurate of all, with a very linear and flat prediction. It would be interesting to see how this model functioned on validation data with a steeper trend than the ones used in this project. Prophet also had great predictions, with no parameter tuning needed, something that was a slight pain point in the implementation of the other algorithms, e.g the model order for the (S)ARIMA models, and the input/output lengths of the LSTM.

5 Conclusion

While none of the models, understandably, predicted the crash, the best versions of all of them proved to be quite decent predictors of the long-term behaviour of the data. ARIMA was surprisingly the best of the tested models, even tho it shouldn't be suitable for data with seasonality. Using any of the models, bar the LSTM(3), could proof beneficial to people speculating in stocks.

Bibliography

[1] Kaggle. "Yahoo stock price dataset." (), [Online]. Available: https://www.kaggle.com/datasets/arashnic/time-series-forecasting-with-yahoo-stock-price.

[2] F. et al., "Forecasting of demand using arima model," 2018. [Online]. Available: https://www.researchgate.net/publication/328633706_Forecasting_of_demand_using_ARIMA_model.

[3] "Augmented dickey–fuller test." (), [Online]. Available: https://en.wikipedia.org/wiki/Augmented_Dickey\%E2\%80\%93Fuller_test.

[4] (), [Online]. Available: https://www.statsmodels.org/dev/generated/statsmodels.tsa.stattools.adfuller.html.

[5] "Pmdarima auto arima." (), [Online]. Available: https://alkaline-ml.com/pmdarima/modules/generated/pmdarima.arima.auto_arima.html.

[6] "Statsmodels.seasonal$_{d}$*ecompose*." (), [Online]. Available: https://www.statsmodels.org/dev/generated/statsmodels.tsa.seasonal.seasonal_decompose.html.

[7] "Darts library." (), [Online]. Available: https://unit8co.github.io/darts/.

[8] Z. et al., "Application of facebook's prophet algorithm for successful sales forecasting based on real-world data," 2020. [Online]. Available: https://www.researchgate.net/publication/341071241_Application_of_Facebook's_Prophet_Algorithm_for_Successful_Sales_Forecasting_Based_on_Real-world_Data.

[9] J. Korstanje, *Advanced Forecasting with Python*. 2021, [pp 115–122].