IBM Specialized Models: Time Series and Survival Analysis

Course Final Project: Stock Price Forecasting with Yahoo Finance, Facebook Prophet and ARIMA

By Javier A. Jaime-Serrano

August 4th, 2021

Abstract

The time series models as seen on the course, are capable to predict future values based on previous seen values, and commonly used in Finance with mixed results. And as Yogi Berra quote of Mark Twain:

“It's tough to make predictions, especially about the future.”

In this Project we will attempt to build a tool that can predict stock prices, merging the Yahoo Finance library with the Facebook Prophet procedure and compare it to the ARIMA models, as seen on the course.

Disclaimer: The information and tool provided in this project is for education purposes only, and not intended as financial or investment advice.

Source and Requirements

Yahoo! Finance market data downloader. Since Yahoo Finance decommissioned their data API, we found this reliable Python library to download market data [1]. We can access historical data with the Ticker module, marked with UTC daytimes.

Prophet is a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. Prophet was released as open source software by Facebook [2].

Autoregressive integrated moving average (ARIMA) models are, in theory, the most general class of models for forecasting a time series which can be made to be “stationary” by differencing, and other nonlinear transformation can also be applied.

Implementation

Problem: We want to predict, in a relatively short term, prices of stock using a combination of tools (described in previous slide).
First we need to import a data set, and we required the Stock ticker as an input:

```
Enter the stock ticker:
```

After enter the stock ticker at the prompt, we get a full data set (figures 1 & 2).
Date: timestamp in UTC daytimes.
Open, High, Low & Close: Stock Prices in US Dollars.
Volume: Daily shares traded.
Dividends: If any, also in US Dollars.
Stock Splits: If any, a ratio of stock prices.

Data Exploration

Figure 1.

```
Enter the stock ticker: TOU.TO
You entered TOU.TO, here is the data from the last 10 days:
```

	Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
742	2021-07-20	33.400002	34.340000	33.240002	33.650002	896700	0.0	0
743	2021-07-21	34.080002	34.770000	33.709999	34.299999	1374900	0.0	0
744	2021-07-22	34.400002	34.470001	33.840000	34.110001	691800	0.0	0
745	2021-07-23	34.220001	34.840000	33.869999	34.770000	704900	0.0	0
746	2021-07-26	35.000000	36.009998	34.900002	35.049999	778500	0.0	0
747	2021-07-27	35.000000	35.080002	33.160000	33.680000	1809700	0.0	0
748	2021-07-28	33.820000	33.990002	33.060001	33.880001	876700	0.0	0
749	2021-07-29	34.610001	35.360001	33.770000	34.740002	1851500	0.0	0
750	2021-07-30	34.549999	34.700001	32.730000	34.060001	1607500	0.0	0
751	2021-08-03	33.540001	34.590000	33.439999	34.220001	1877828	0.0	0

Data Exploration

Figure 2.

These are the basic stats:

	Open	High	Low	Close	Volume	Dividends	Stock Splits
count	753.000000	753.000000	753.000000	753.000000	7.530000e+02	753.000000	753.0
mean	17.424663	17.745975	17.100788	17.414396	1.285277e+06	0.001965	0.0
std	5.614929	5.685222	5.562863	5.624662	1.055763e+06	0.015655	0.0
min	6.882140	7.200222	6.486947	6.544780	1.218000e+05	0.000000	0.0
25%	13.137756	13.378726	12.886359	13.140881	7.665000e+05	0.000000	0.0
50%	16.725880	17.019788	16.461013	16.733917	1.072900e+06	0.000000	0.0
75%	19.562654	19.834048	19.280356	19.480240	1.490200e+06	0.000000	0.0
max	35.990002	36.180000	35.330002	36.139999	1.593540e+07	0.160000	0.0

Data Exploration

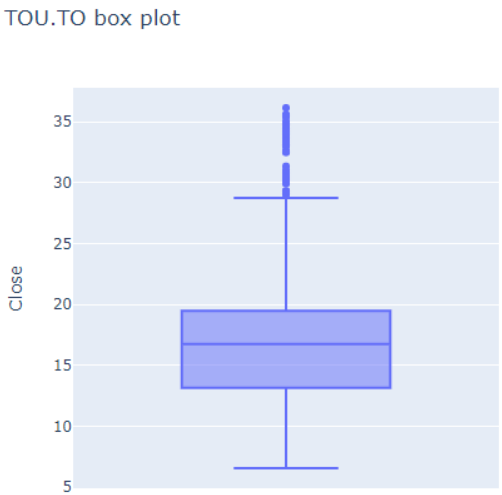
We used plotly express to visualize the data.

- The box plot is used to visualize the distribution of the closing prices (figure 3).

- The line graph is used to visualize the stock performance, with daily closing prices (figure 4).

- The bars are used to visualize the stock trading volume per day (figure 5).

Figure 3.

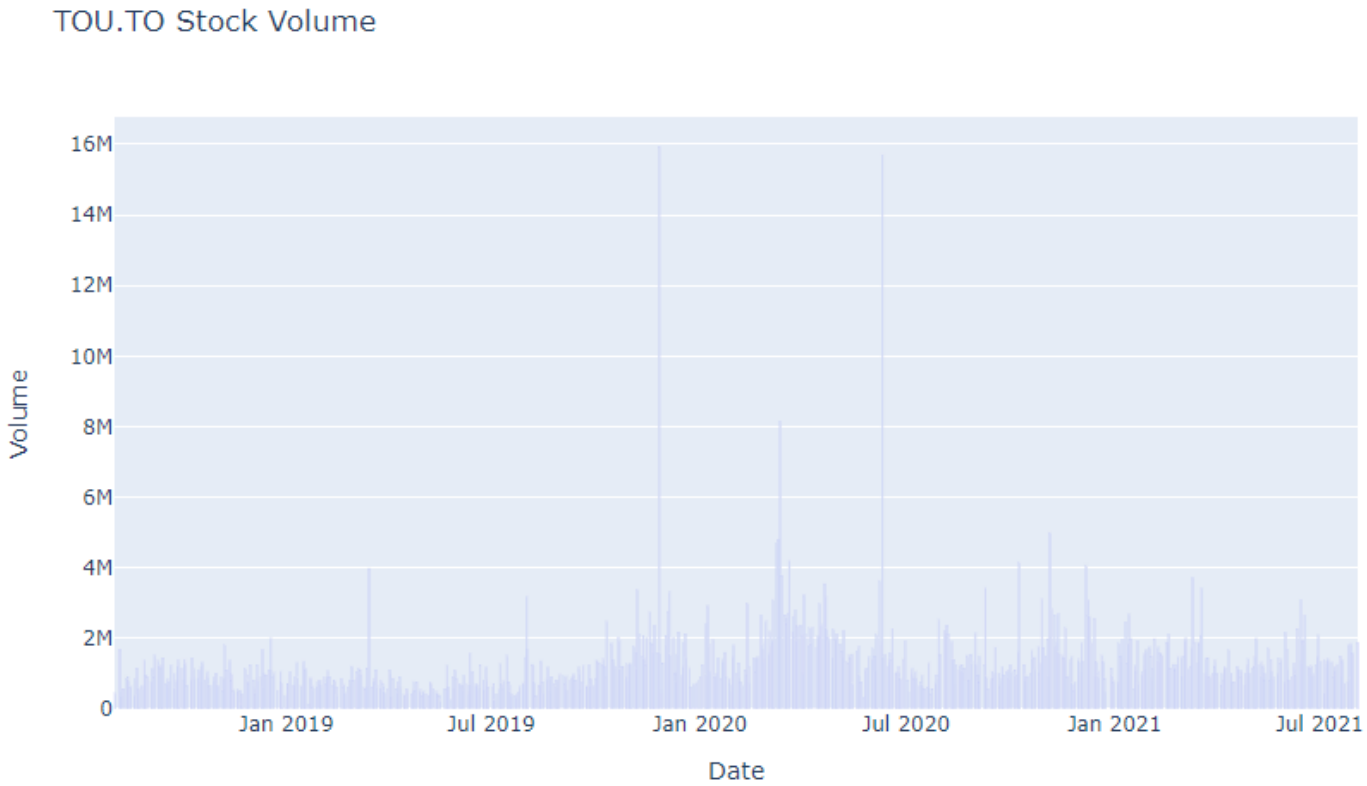


Data Exploration

Figure 4.

Data Exploration

Figure 5.



Findings (Prophet)

First we needed to prepare the data for Facebook Prophet:

- extracted the Date and Close Prices
- rename the Columns: Date to ds (datestamp) and Close to y (target).

Then we created a Prophet model with yearly seasonality (optional).

And made a forecast for a 30 days period (figure 6 table, figure 7 line graph).

Also we had the time series decomposition of the data into trend, weekly seasonality and yearly seasonality (figure 8).

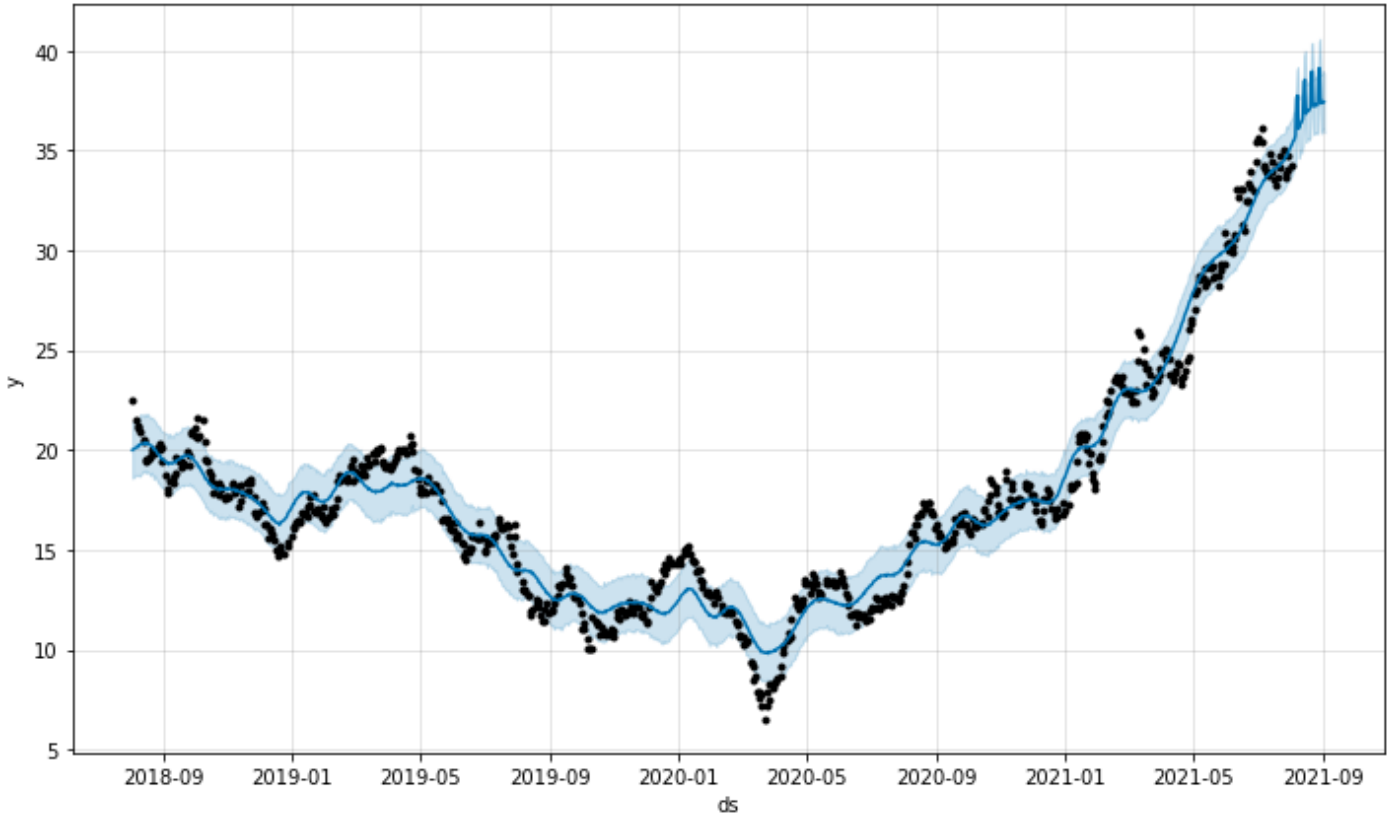
Findings (Prophet)

Figure 6.

	ds	trend	yhat_lower	yhat_upper	trend_lower	trend_upper	additive_terms	additive_terms_lower	additive_terms_upper	weekly	weekly_lower	weekly_upper	yearly	yearly_lower	yearly_upper
0	2018-08-03	19.847255	18.591368	21.425434	19.847255	19.847255	0.156650	0.156650	0.156650	-0.473247	-0.473247	-0.473247	0.629896	0.629896	0.629896
1	2018-08-07	19.781615	18.771442	21.483943	19.781615	19.781615	0.358859	0.358859	0.358859	-0.499441	-0.499441	-0.499441	0.858301	0.858301	0.858301
2	2018-08-08	19.765205	18.763471	21.598602	19.765205	19.765205	0.439659	0.439659	0.439659	-0.477167	-0.477167	-0.477167	0.916827	0.916827	0.916827
3	2018-08-09	19.748795	18.699178	21.467110	19.748795	19.748795	0.413680	0.413680	0.413680	-0.559723	-0.559723	-0.559723	0.973404	0.973404	0.973404
4	2018-08-10	19.732385	18.705401	21.798831	19.732385	19.732385	0.553543	0.553543	0.553543	-0.473247	-0.473247	-0.473247	1.026790	1.026790	1.026790
5	2018-08-13	19.683155	18.947881	21.769077	19.683155	19.683155	0.629233	0.629233	0.629233	-0.527232	-0.527232	-0.527232	1.156465	1.156465	1.156465
6	2018-08-14	19.666745	18.892185	21.678722	19.666745	19.666745	0.686873	0.686873	0.686873	-0.499441	-0.499441	-0.499441	1.186314	1.186314	1.186314
7	2018-08-15	19.650335	18.996764	21.837273	19.650335	19.650335	0.731072	0.731072	0.731072	-0.477167	-0.477167	-0.477167	1.208239	1.208239	1.208239
8	2018-08-16	19.633925	18.854319	21.723719	19.633925	19.633925	0.662041	0.662041	0.662041	-0.559723	-0.559723	-0.559723	1.221764	1.221764	1.221764
9	2018-08-17	19.617515	18.942443	21.833097	19.617515	19.617515	0.753365	0.753365	0.753365	-0.473247	-0.473247	-0.473247	1.226611	1.226611	1.226611

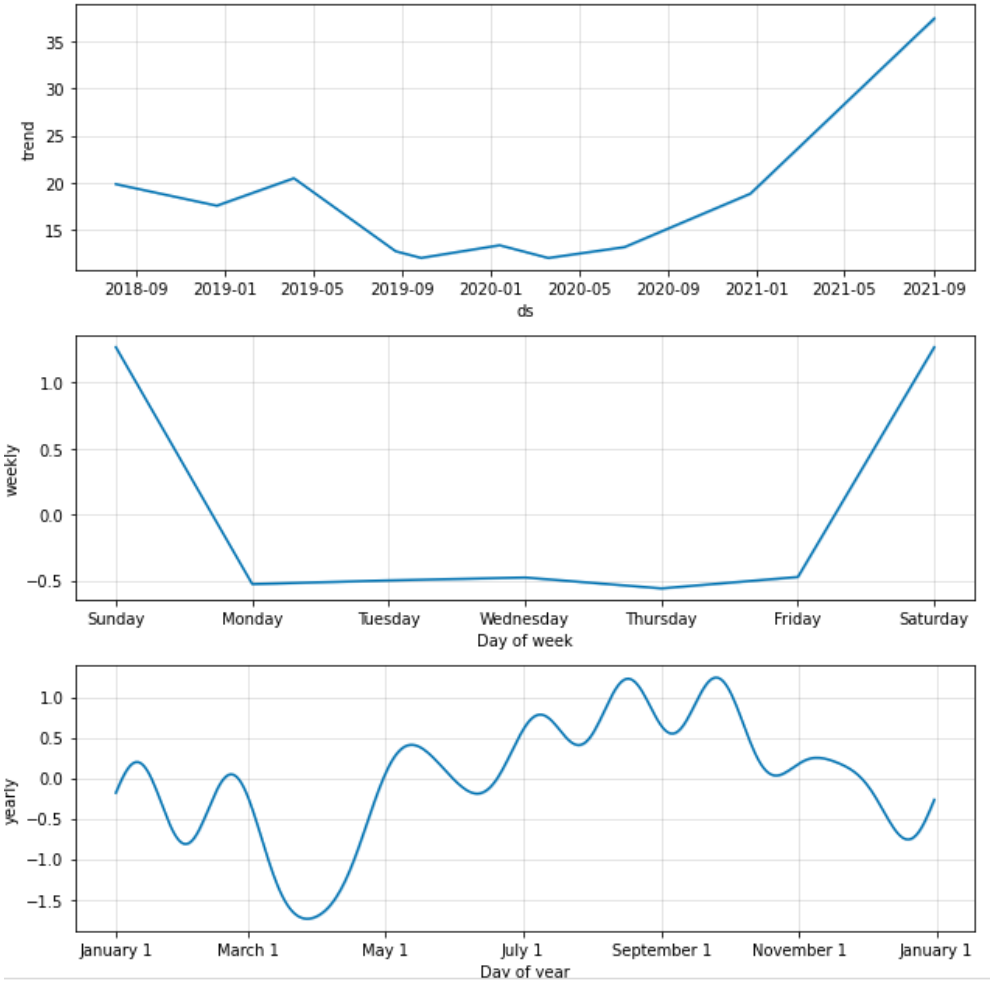
Findings (Prophet)

Figure 7.



Findings (Prophet)

Figure 8.



Results (Prophet)

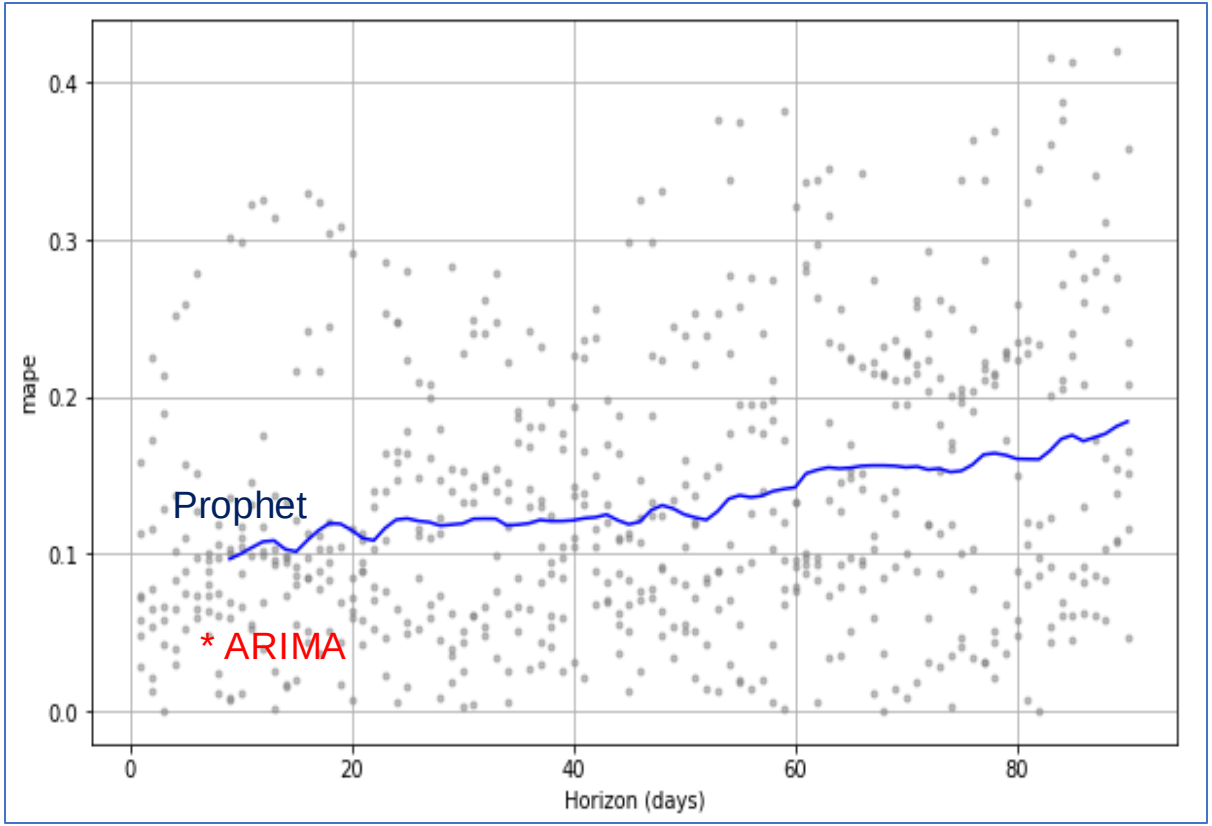
We used the cross validation function from the prophet diagnostics to make 10 forecasts, with an initial timeframe of 2 years, a 30 days forecasting period, and an horizon of 90 days, and with the performance metrics obtained these results:

Figure 9.

	horizon	mse	rmse	mae	mape	mdape	coverage
0	9 days	4.833546	2.198533	1.872137	0.096944	0.077706	0.398126
1	10 days	5.188486	2.277825	1.934192	0.100223	0.089272	0.385246
2	11 days	5.627902	2.372320	2.015032	0.104193	0.089376	0.367681
3	12 days	6.160428	2.482021	2.118181	0.107935	0.096797	0.337237
4	13 days	6.439791	2.537674	2.150961	0.108609	0.097135	0.325137

Results (Prophet)

We chose Mean Absolute Percentage Error (MAPE) as the cv metric and plot it against the horizon. Figure 10.



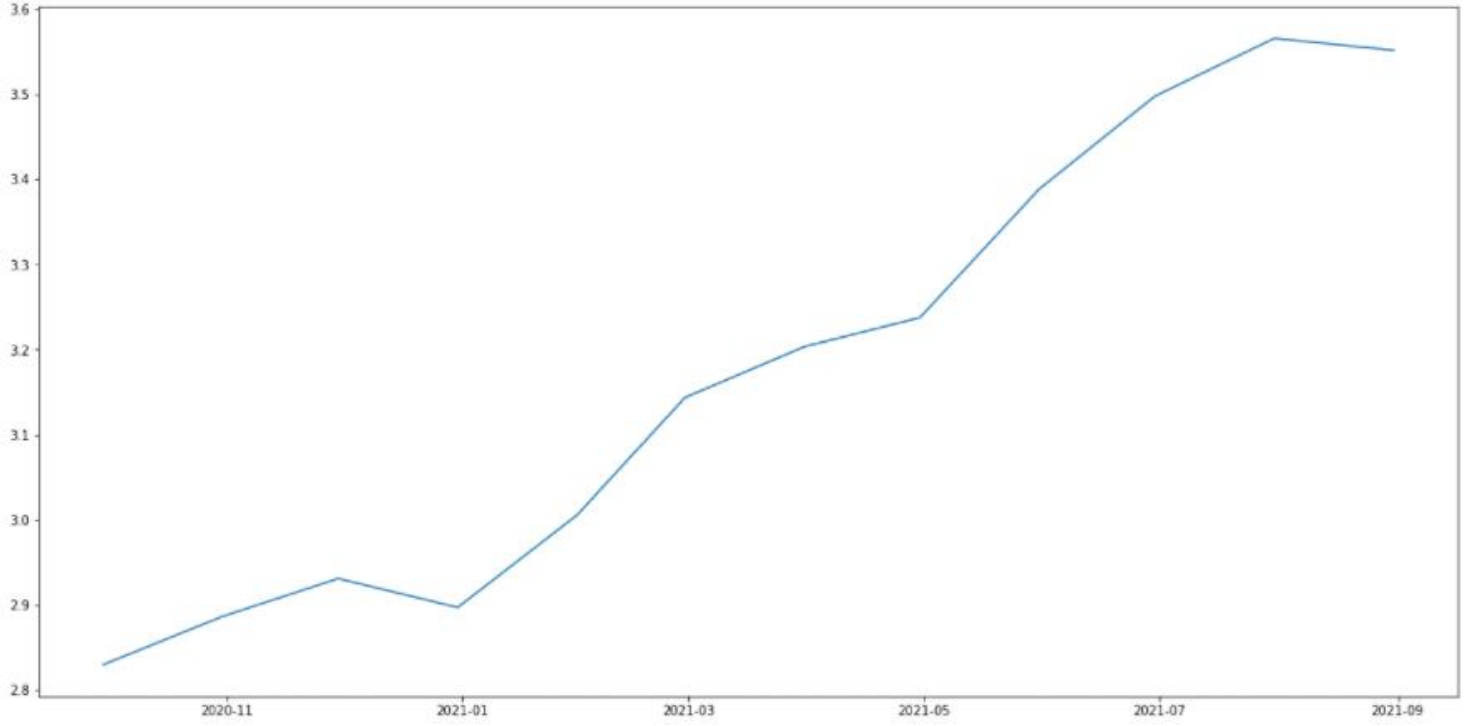
Findings (ARIMA)

First we needed to prepare the data for Arima:
- We extracted the Date and Close Prices, no need to rename the columns.
- In order to reduce the difference in variances we took the log of the Close price.
- We resample the data to months and assign the mean values (figure 11).
- We plotted the seasonal decomposition from statmodels.tsa [2] with 12 months frequency (figure 12), the autocorrelation and partial autocorrelation (figure 13).
- We extracted the year to year difference and ran a Dickey-Fuller test (figure 14).
- We plotted year difference autocorrelation and partial autocorrelation (figure 15).

And then we created the ARIMA model with weekly seasonality, because we failed to reject the null of non-stationary in the Dickey-Fuller test (figure 16).

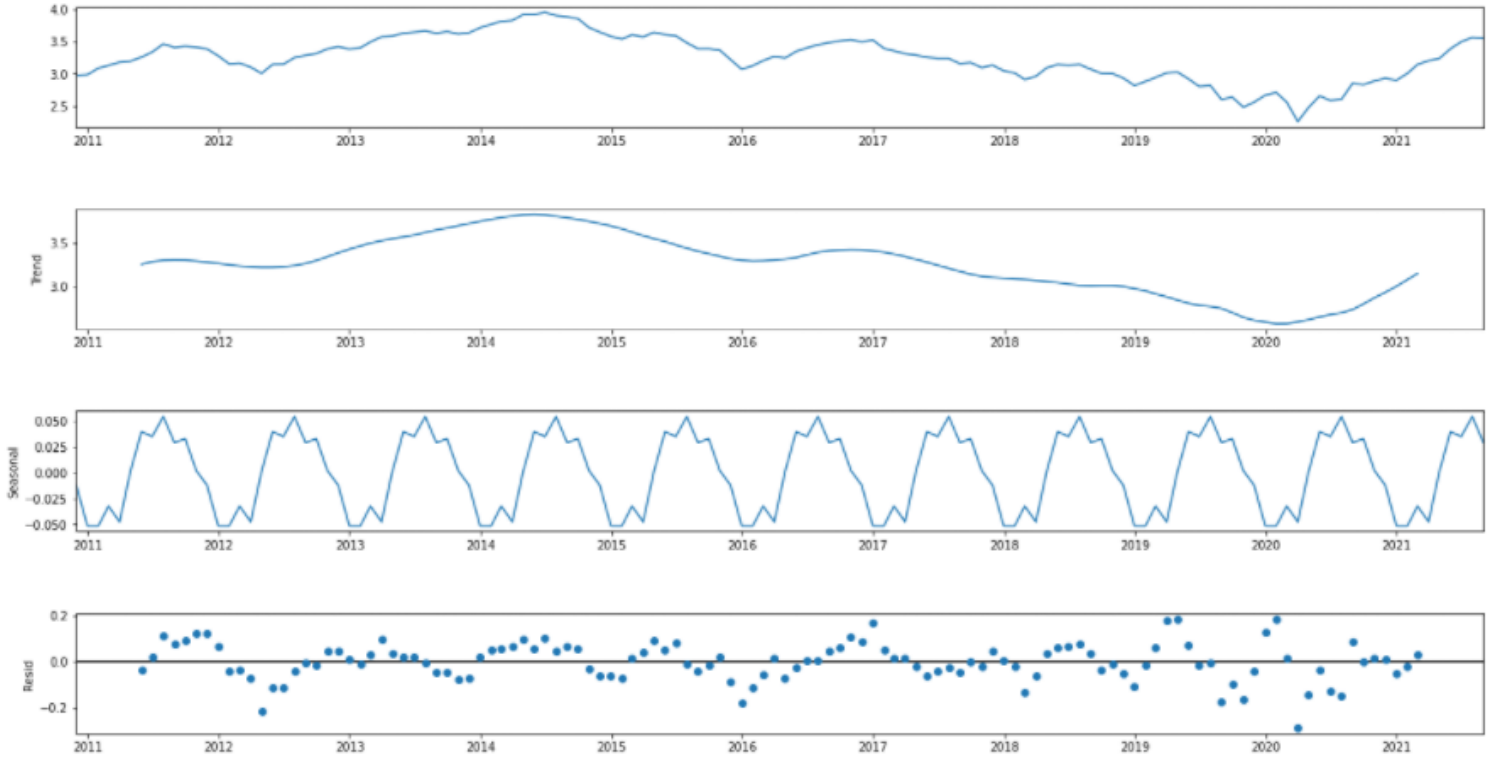
Findings (ARIMA)

Figure 11.



Findings (ARIMA)

Figure 12.



Findings (ARIMA)

Figure 13.

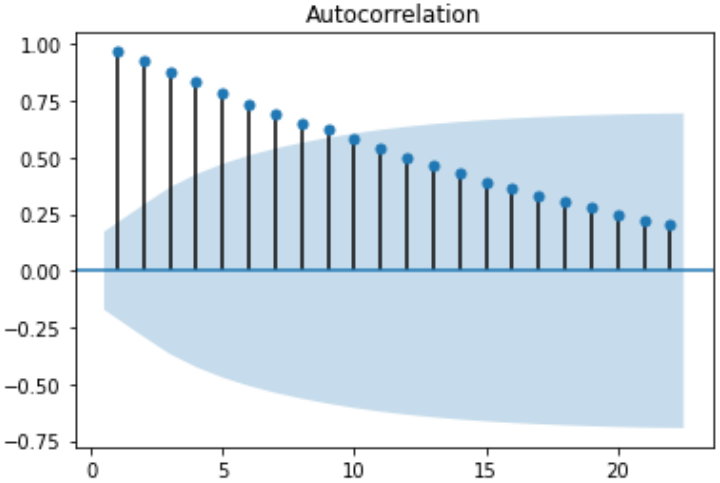
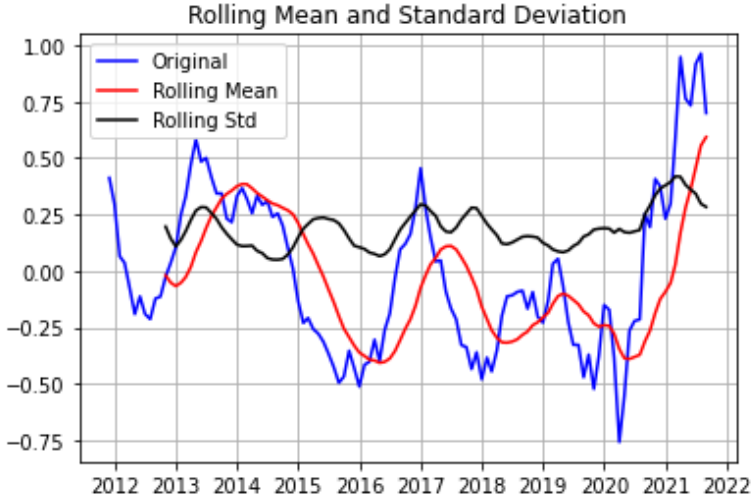
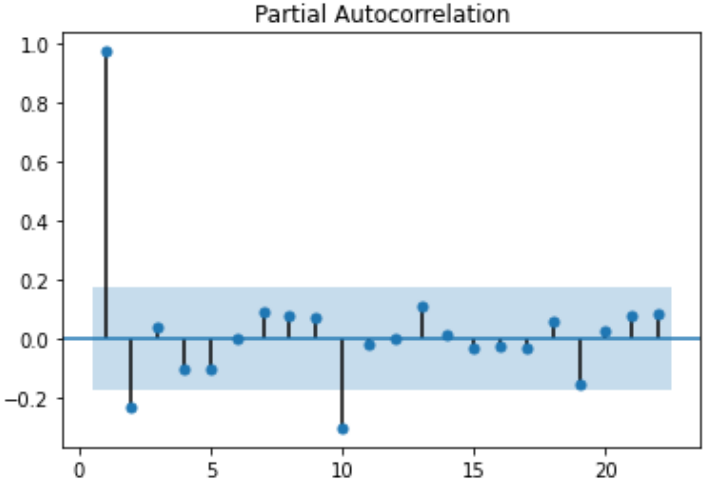


Figure 14.

```
Test Statistic           -1.464259
p-value                   0.551112
Lags Used                12.000000
Observations Used       105.000000
Critical Value (1%)      -3.494220
Critical Value (5%)      -2.889485
Critical Value (10%)     -2.581676
dtype: float64
```





Findings (ARIMA)

Figure 15.

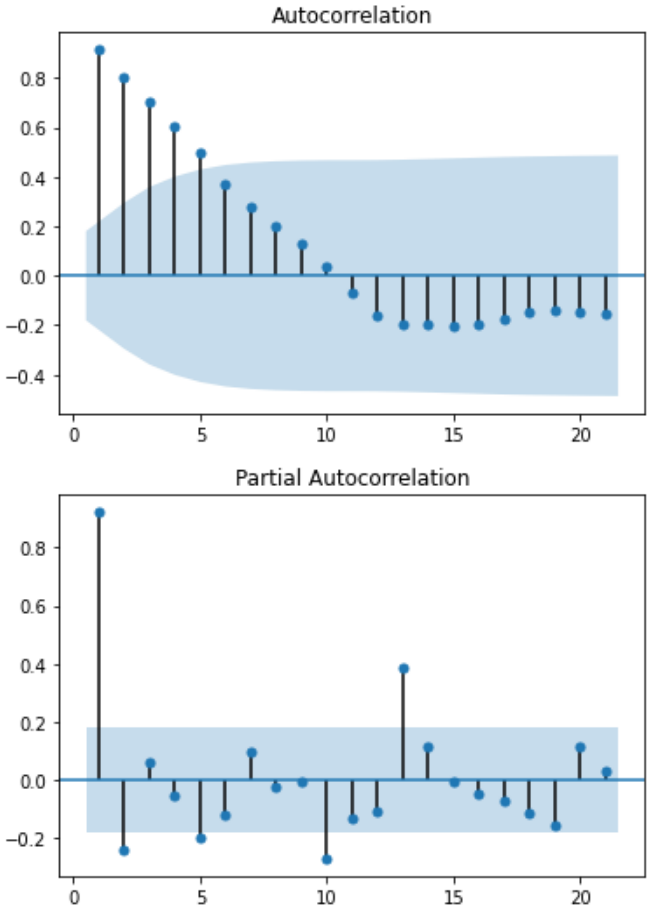


Figure 16.

SARIMAX Results

Dep. Variable:	log_close	No. Observations:	2684
Model:	SARIMAX(0, 1, 0, 7)	Log Likelihood	3789.800
Date:	Wed, 04 Aug 2021	AIC	-7575.600
Time:	19:04:15	BIC	-7563.815
Sample:	11-23-2010	HQIC	-7571.337
	- 08-04-2021		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	0.0015	0.001	1.349	0.177	-0.001	0.004
sigma2	0.0035	7.42e-05	46.474	0.000	0.003	0.004

Ljung-Box (L1) (Q):	2010.90	Jarque-Bera (JB):	168.31
Prob(Q):	0.00	Prob(JB):	0.00
Heteroskedasticity (H):	2.20	Skew:	-0.02
Prob(H) (two-sided):	0.00	Kurtosis:	4.23

Results (ARIMA)

We ran the diagnostics of the model, no lags (figure 17), not perfect, but close.

And then we created and Auto Model to find the best Model parameters with weekly seasonality, and ran the diagnostics again with the new model.

```
Best model:  ARIMA(3,0,0)(1,1,1)[7] intercept      order:  (3, 0, 0)
Total fit time: 378.945 seconds                    seasonal order:  (1, 1, 1, 7)
```

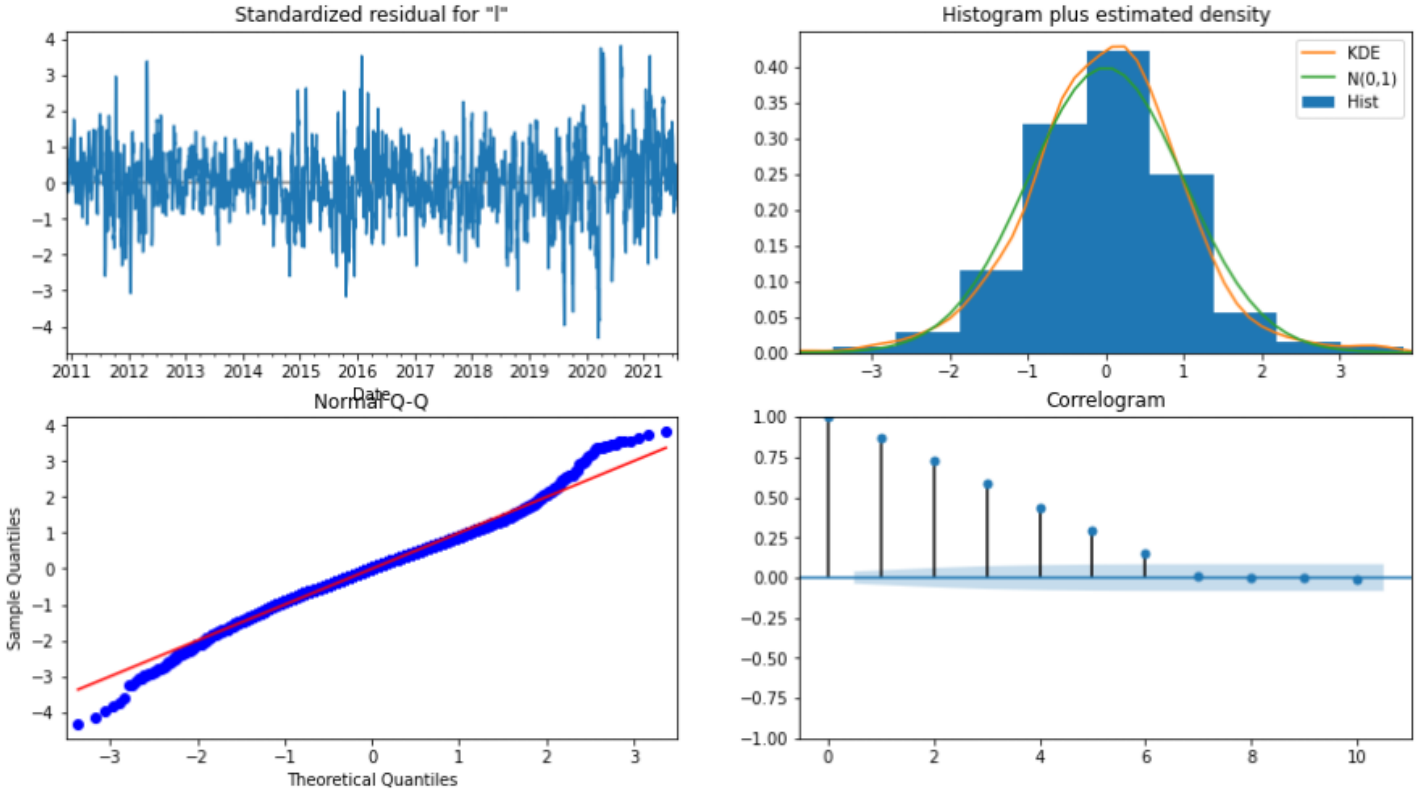
With the help of a function we cross validate and ran it with the log close series and weekly seasonality, with the start at 75% percent of the maximum time.

We apply the np Exp function to reverse the Log and plot the results (figure 18).

The resulting MAPE of 0.058 is slightly better than the Prophet MAPE (figure 10), but this is only for a 7 days horizon and it took hours to run.

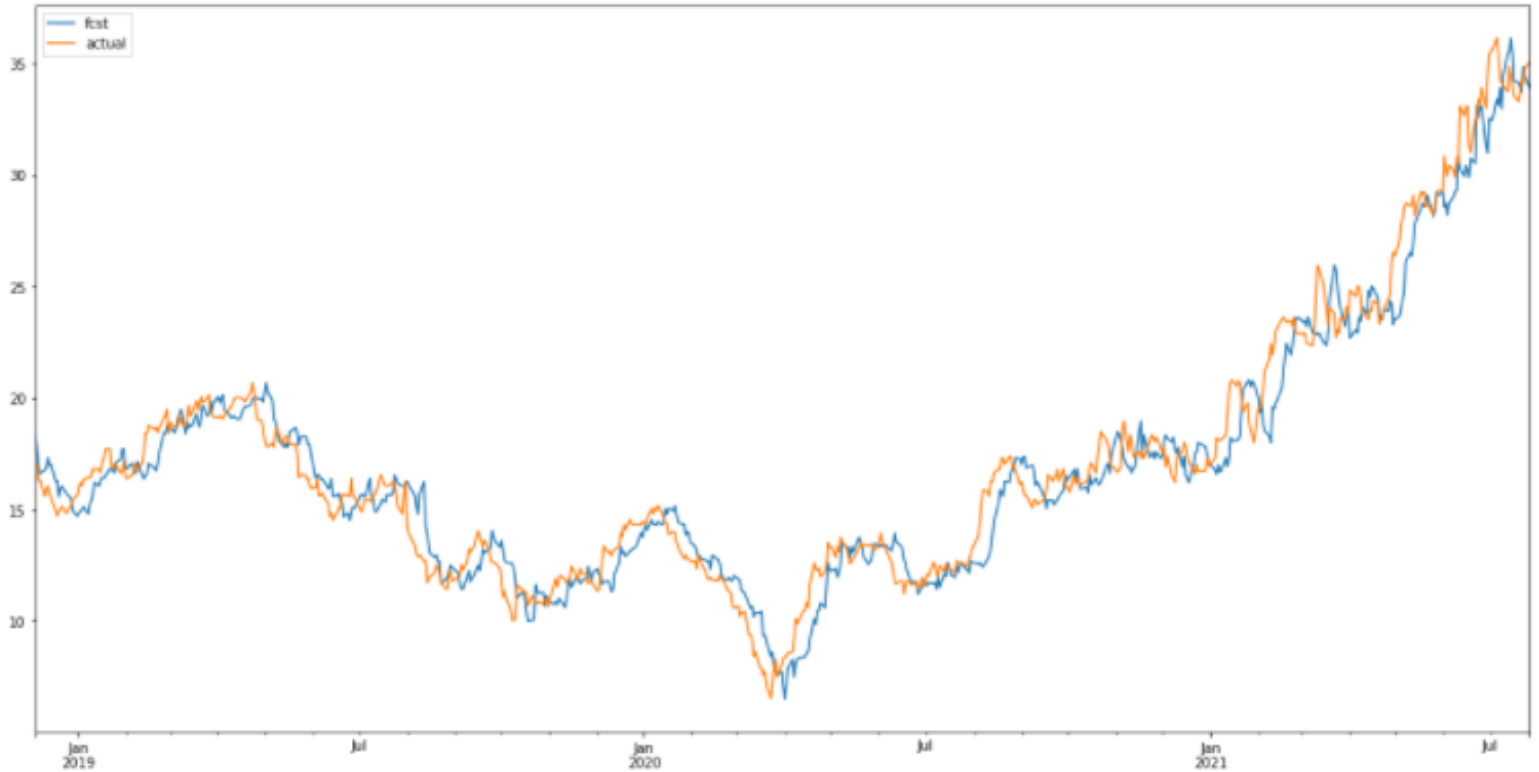
Findings (ARIMA)

Figure 17.



Findings (ARIMA)

Figure 18.



Conclusions

In this Project we built a tool that can predict stock prices (for education purposes) merging the Yahoo Finance and Facebook Prophet libraries we got a good model.

With a minimum 3 years of data we were able to do a seasonal decomposition and extract a valid trend, forecasting short term periods with Mean Absolute Percentage Errors (MAPE) averaging 0.1 for the stock ticker selected, in seconds!

The ARIMA model resulted being more complex, we needed to find to test the seasonality, find the best parameters and cross validate to get a similar MAPE.

Instead of spending hours finding the right parameters, we prefer to play with the periods and the forecasting horizons, and get a better interpretation of the results.

References

[1] Yahoo! Finance market data downloader: https://pypi.org/project/yfinance/

[2] Facebook Prophet. Forecasting at scale: https://facebook.github.io/prophet/

[3] Plotly Express in Python: https://plotly.com/python/plotly-express/

[4] Introduction to ARIMA (from Duke): https://people.duke.edu/~rnau/411arim.htm

[5] Statsmodels time series analysis: https://www.statsmodels.org/devel/tsa.html

[6] Jupiter Notebook: https://github.com/javier-jaime/Stock_Price_Forecasting