# Predicting the Stock Price of REC Silicon (:RECSI) Using Machine Learning

Kasper Bruland, 18.11.22

# THE PROBLEM

I will attempt to develop a machine learning model for predicting the price of the stock REC Silicon (ticker: RECSI) listed on the Oslo Stock Exchange. This model will hopefully be able to predict the price of any stock, but I choose to develop it around RECSI because I am invested in it myself. By creating a model that works, users of the model would be able to make financial profit by investing using the model's predictions.

## About RECSI

REC Silicon produces among other things polysilicon used for production of solar panels. RECSI is widely considered a turbulent stock among investors, suitable for swing trading, swing trading would imply trading the swings in the price. This is one reason to why I want to develop this model, as the potential gain for financial profit would be enormous. The turbulent prices will also mean that the risk of making a poor model will increase. Since its time of listing in 2006 the price has dropped by -98%, and in the last three years the price has increased +518%. There are obvious reasons for the decline of the stock. And that is that by nature, solar energy is not cost effective. I will not dig into the details of this, but the lack of demand for silicon products over the years have contributed to the downfall of the stock. However, this is changing now that governments have started to subsidize green energy. And investors have caught up to that fact, which is why the stock has increased so much the last years. RECSI has announced the reopening of their production facility in Moses Lake, Washington. And recently the American government passed a bill allocating $369 billion to climate change prevention and green energy. Being biased in my own prediction, this would indicate a bullish trend for the stock.

Trying to predict stock prices without computer algorithms would require a person to do either

- Technical analysis
- Mass psychology analysis
- Closely monitor world news and announcements from the board

or all of them at the same time. Doing this requires time and a certain skill. Yet, professional stockbrokers and financial institutions often fail at predicting correct, even with advanced trading tools.

This model will only consider the price history, and no other environmental factors. There are a lot of trading algorithms and models that already exist. My mission is to make one directly aimed at RECSI.
It would be possible to download weekly or monthly data and use the same model for predicting the next weeks or months if the user wants to predict more generally. I believe that a monthly prediction of 1 is more accurate than a daily prediction of 30.

The model will need to be updated with the most recent price changes to be considered accurate.

It would be impossible to speculate on what would happen if my model were “accurate”. I think that is where mass psychology comes into play. Unless I played my cards close to my chest, the masses would affect the price by using the model predictions as a metric for trading, more than the model could anticipate, therefore rendering the model useless. With this assumption, every stock trader would be a stakeholder by rolling out an accurate model.

Unfortunately, the current model will not be finalized. As you will see, the model does not perform well enough to be used for financial decisions. Many people have tried to solve this problem in one way or another, and as far as I know, none of them have succeeded by any substantial amount. Maybe I will get lucky and produce something worthwhile, but I’m afraid many people with expertise in the fields surrounding this problem will be required to solve the problem.

## METRICS

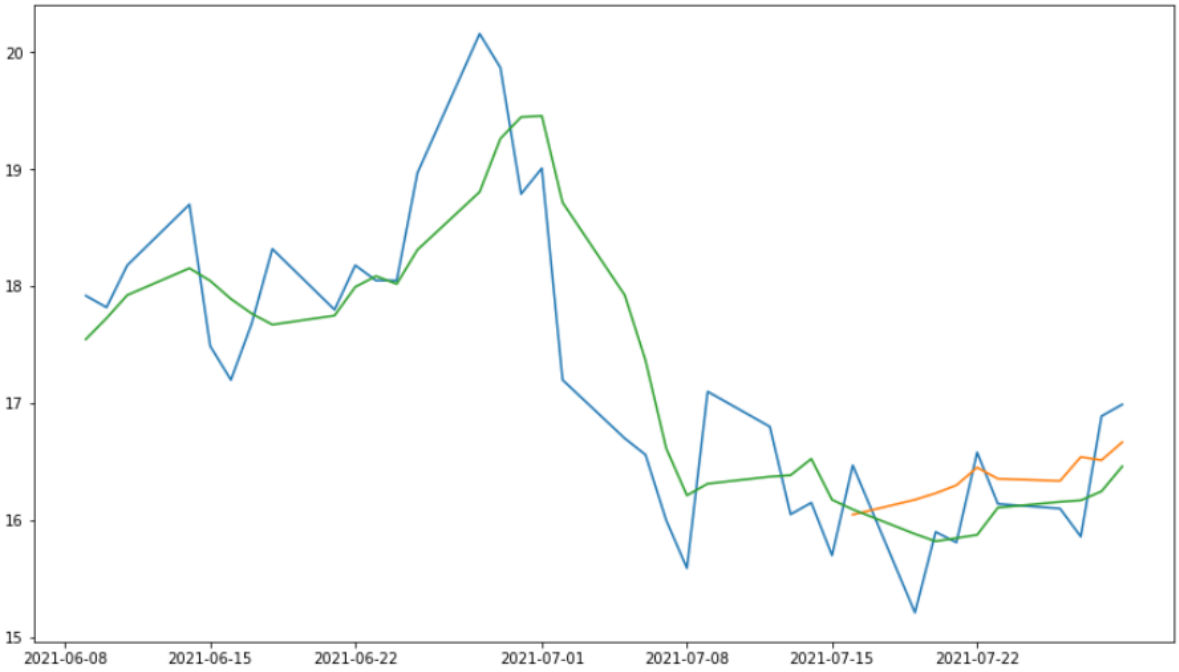
It is important that I make my intentions clear. When searching for solutions for forecasting problems online, most of the examples and solutions involving machine learning and deep learning were not trying to predict unseen data multiple steps in the future. To put it into perspective, they would use a test set (t0,t1,t2,t3,t4) and an input size of 3. Then the model would take (t0,t1,t2) and make the prediction p3, for the next sequence, they would take (t1,t2,t3) and predict p4, and so on. That way they would get ridiculously accurate predictions. This is NOT what I am trying to do, as they are only predicting 1 step ahead.
The model will use an interval of the test data, and make multiple predictions in one go.

What determines if my model is successful? Because this is a regression task my predictions would have to fall within a bound to be considered accurate. If a user wants the prediction after 10 days, then it is possible to take the tenth day and see if it falls within the bounds of the actual price. Or to have a more complete testing, we will measure all the predictions.

This can be done by computing the moving average of the test price, and computing the mean squared error from the test price to the mean average. This will give us a reasonable margin of error. We would have to pass in a value for the computation which is the window size which will be used to calculate the moving average. The smaller the window, the closer to the actual price the moving average will fall. Having computed the moving average, we can find a baseline for the MSE by calculating the MSE for the test values and moving average. We then calculate the MSE for the predicted values and the test values, and compare it with the moving average MSE. If the predicted MSE is lower than the moving average MSE, we will assume we have made an accurate prediction.
There are problems with this approach as the moving average does not capture large price movements very well, because it lags behind if the window size is larger.

For predicting the future prices we would obviously have to wait and see if it indeed was accurate.

With a model input size of 30, output size of 10 and a moving average window of 4, the MSE for moving average values is 0.36284, and the MSE for the predicted values is 0.22874. The model did a good job in this prediction.




## DATA

Using pandas_datareader we can fetch the data directly from Yahoo Finance.
Because of the steep decline and the recent resurgence of RECSI the data will be spanning from 2013 until today. The graph as plotted in Yahoo Finance looks like this, with each candle stick representing one month.

Different datasets will be available. One set for the full 10 years of data, one model spanning from 2017 until today. I chose the staring year 2017 because the stock hit a stable floor around that time. And lastly a dataset for the last two years only.

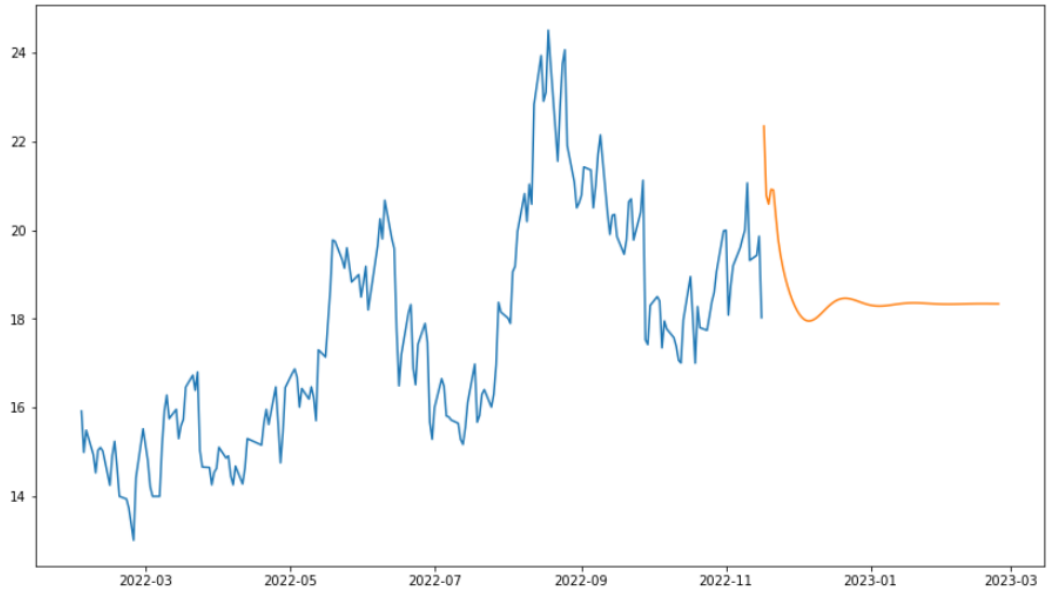
The data is a data-point table consisting of the following features: the date of the trading day, highest price of the day, lowest price of the day, opening price of the day, closing price of the day, the total volume of stocks traded that day, and the adjusted closing price. I had planned to keep the volume feature and create a new feature that is the percentage of change from the previous day, but I was forced to use only the closing price in my implementation as I will go on to discuss later. There are no missing values.
The datapoints are of one day frequency, with exception of the days where the exchange was closed. My wish was to predict intraday as well, but I found no free resources to download intraday stock history.
The data will, as I will go into detail of later, be scaled using sklearns MinMaxScaler.

It is the closing price that I intend to predict.

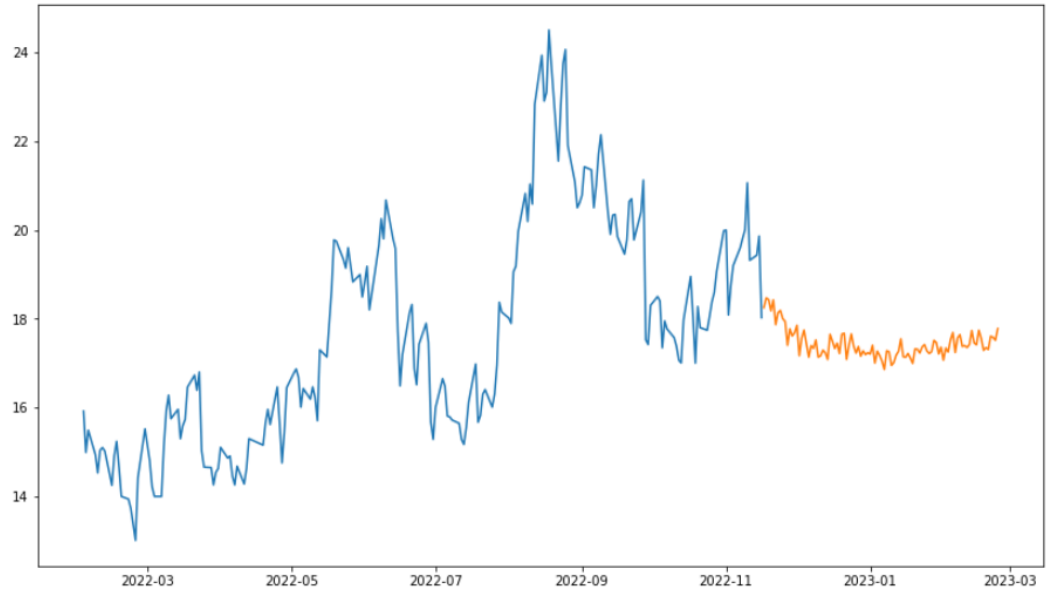
## MODELING

The model will predict the future values based on time and previous values. There are several models which can do this in different manners as seen in this blog https://machinelearningmastery.com/time-series-forecasting-methods-in-python-cheat-sheet/. For example, a simple linear regression model could find the mean trend in the data and thereby predict within a reasonable amount where the next datapoint would fall. Another way would be to use a preexisting technical analysis metric, like 'moving average', and predict the next step. Using regression, you could make the model predict the following steps with the newly predicted step.

The model I found the most interesting was LSTM, short for "long short-term memory", which strictly speaking is a deep learning model. LSTM can be used for many things, but here it will be used for multi-step forecasting. It is gathered from the Keras library in TensorFlow. For my purpose, there were two different model variants. Vector Output Model and Encoder-decoder Model. I tried them both and found the Vector Output Model to give the most "realistic" output. The Encoder-decoder Model would predict data resulting in a large wave form that would converge, as shown below.




The Vector Output Model would look more like this.




Trying to explain how LSTM works is a difficult task for me as I've only explored the surface of it and have no previous knowledge of Recurrent Neural Networks which LSTM is based upon. But I will try to give a simple explanation of parts of it.
LSTM uses a cell with different three gates, input, output and forget gate. The cell has different functions in it and uses them to compute some input and its previous long- and short-term memory to determine if it wants to forget the value or output it for the next sequence of computation. Hence the name, LSTM utilizes both its short- and long-term memory to make a prediction.
By chaining these cells together, the model can make a prediction on input data.

I chose to scale my data with the MinMaxScaler, though not necessary, it generally yields better results. A flaw with LSTM is that by training a model with values ranging from 1 to 10, LSTM will

have difficulty predicting values exceeding these limits, because it has learned with the training data that 1 and 10 are the lowest and highest values you can get.
LSTM requires a lot of resources to train the model, so it may not be the best model if you compare it with more simpler models that can perform well in solving the problem.

As mentioned, I wanted to utilize multiple features in my model, but was forced to use only the Closing price. Multivariate forecasting proved too difficult for my experience in this subject.

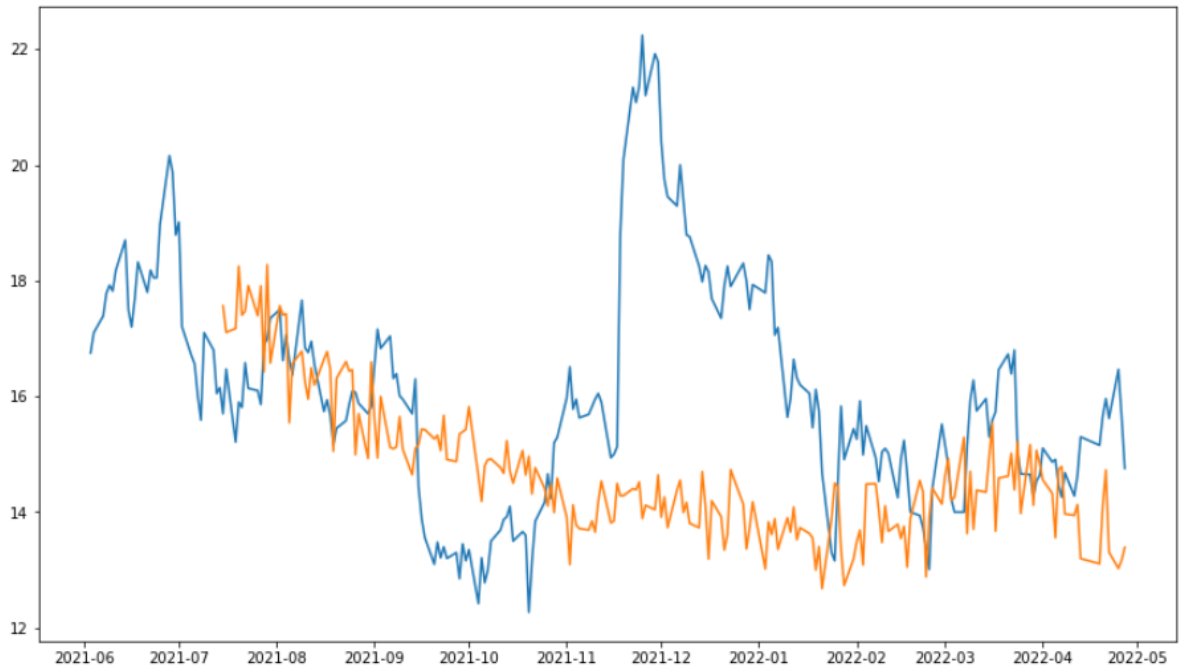
How to implement LSTM. You train the model by giving it a section of the data ($t_{i-4}$,$t_{i-3}$,$t_{i-2}$,$t_{i-1}$), and the next value ($t_i$) to predict $p_i$. Then you give it i+1 to predict $p_{i+1}$, and punish it with a loss function. To do this you need a function that transforms the data into samples of training data with length n and validation data of length m. n is the length of how input steps. m is the length of output steps. The model can only use data in a 3D format, so we must reshape it to this format: (samples, time steps, features).
I have borrowed the splitting function and the model implementation from this blog https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/ at section "Multi-Step LSTM Models".
The model is built up by layers. The first being an input layer specifying the shape of the input. Secondly come the hidden layers of which you can stack multiple of them. Lastly is the output layer where we specify how many output neurons we will output.

The hyperparameters have not been explored properly, so these will be left alone.

The best results came from using the largest data set. I should have compared the LSTM model with other models to see the difference, but I did not get that far. I did not have time to experiment a lot, but the observations I made were that the longer the prediction steps were, the less accurate the model performed, as expected. Ultimately, I have yet to find a sweet spot for the combination of input and output lengths.

It is worth noting that my predictions for the future does not have the correct date indexes, this is because non trading days have not been filtered out. This means we must count from today's date and skip over non trading days to find the real date the prediction will land on.

The model's predictions were too "reserved" for my liking. I had wished they were more loosely bounded like the price movements of RECSI really are. It seems the predictions always fall within a bound of the previous predictions. This is something that probably can be model engineered.




## DEPLOYMENT

There is no plan for rolling out the model just yet, as it's not finished or proven to be working. But with the assumption that it is, it is possible to think it would have to be launched on a cloud-based server, seeing as the LSTM model demands heavy recourse usage. From an interface standpoint, a user would have to input the length of days he wants to predict. Optimally, the model will always be updated with the most recent data, and will have the best input size calibrations matching the output size that the user suggests. The moving average window should also be matched according to the length of the in- and out-put to get the best approximation.

## REFERENCES

Jason Brownlee, How to Develop LSTM Models for Time Series Forecasting, Multi-Step LSTM Models, Machine Learning Mastery, https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/, 18.11.22.

StatQuest with Josh Starmer, Long Short-Term Memory (LSTM), Clearly Explained https://www.youtube.com/watch?v=YCzL96nL7j0 (this helped me understand key parts of LSTM)

Jason Brownlee, Multistep Time Series Forecasting with LSTMs in Python, Machine Learning Mastery, https://machinelearningmastery.com/multi-step-time-series-forecasting-long-short-term-memory-networks-python/ 18.11.22.