Machine Learning Engineer Nanodegree
Capstone Project

Pedro Couto
February, 2021

Stock Price Predictions of Brazilian commodity-based companies on AWS SageMaker

Image 1: Stock Price Illustration. - Source: https://unsplash.com/photos/fiXLQXAhCfk

1. Definition

1.1. Overview and Background

The stock price market might seems chaotic at a first glance. There is a lot of information that must be taken into account in order to decide whether to buy or not a stock. Some say that the Fundamental Analysis [1] is the right way to decide whether or not to buy a stock, and some say that the Technical Analysis [2].

Based on this lack of convergence about which method should be used, and once that we can find a giant amount of data, a data science project could be a nice fit to predict the behaviour of the prices in the next day. Therefore we could use these predictions to guide the decisions.

Nowadays, besides the corona crises, the markets of the USA are at their historic maximum. This fact means that there is not so much place to find a share of business with good profits possibilities. Therefore, when we would like to have a bigger profit, we must accept more risk and investments that fit this profile are those in emergent markets, like Brazil, India, China, etc. I'm a chemical engineer from Brazil and to use data science to understand the market behaviour for the next weeks are a topic that interests me a lot and can bring my education, my experience living here and my data science skills together.

As we know, the commodity is an economic good that is often used as input for the production of other goods or services. These goods have full or substantial fungibility, that is, they can be treated as equivalent (or nearly) regardless of who produced them. Some examples are coffee, gold ore, iron ore, oil, water, electric power, etc. Brazil is a land that has a commodity-based economy [3] and to be able to predict the share prices of business that work in some level with commodities, cold be an excellent opportunity for discovering bad and good calls.

Besides that, nowadays there are lots of discussions about a commodity rally in 2021 [4][5][6][7]. That means, understanding the future of these goods and being able to predict the share price of the companies or even the commodity itself like gold or silver, are key points to have success investing in Brazil.

So let's dive into the Brazilian Stock Prices of commodity-based companies, explore them and predict their prices!

1.2. Problem Statement

The problem will be stated in two phases:

a) **Analysis**: of the 5 biggest commodity-based companies in Brazil in order to understand their behaviour, possible future results and the relation between Expected Return and Risk. Finally, we will select one company with the most interesting profile to be used in the next phase.

b) **Forecast Model Building**: in order to predict the stock price for the next 28 days (4 weeks) so we can make important decisions for 2021. For this task should try different models in order to find the best one.

Since this project is part of the Machine Learning Engineer Nanodegree, the problem should be developed using the AWS Platform as well as its workflow.
Another point that must come to light is: this project aims to improve skills on SageMaker and not to build a prediction for each stock. Hence the use of the algorithms is the same regardless of the stock, I would need to train 5 to 7 models spending SageMaker Resources at a cost of repetitions that don't improve any skill on AWS.

1.3. Metrics

Because this type of forecasting works with continuous output that can be a big range of values, the metrics used in the project will be **MSE (mean square error)**, **RMSE (root mean square error)** and if needed **MAE (mean absolute error).** They are standard for approaching this type of problem, so it will fit pretty good this project.

2. Analysis

2.1. Dataset Exploration

The data were gathered using the Yahoo! Finance API. A great tutorial (unfortunately only in Portuguese) can be found in this medium post. The library yahooquery gives us all tools to collect the data we want. Once that this API is developed for the Brazilian Market, all the prices are in Real (R$), a Brazilian Currency that nowadays worths around 20% of a dollar.

We will gather data on the 5 biggest commodity-based companies in Brazil, that is:

- Petrobras (PETR4): largest Brazilian company, Petrobras produces oil;
- Vale (VALE3): the company is among the largest companies in Brazil, and is the largest producer of iron ore in the world;
- CSN (CSNA3): it is the largest steel industry in Brazil and Latin America, and one of the largest in the world;
- JBS (JBSS3): the company is one of the largest producers of animal protein in the world;
- Suzano (SUZB3): the company is considered the largest producer of eucalyptus pulp of the world;

Since the last truly commodity rally was in 2008 and these companies had their maximum prices there.

The data will be divided into 7 features for each day: lowest, highest, open, closed and adjusted close price, as well as volume and ticker.
Later on the EDA an extra feature, daily return, was created in order to explore the risk and expected return of each stock. A sample of the dataset is presented in the figure below.

	symbol	date	open	low	high	volume	close	adjclose	dividends	splits	daily_return
0	PETR4.SA	2007-01-02	25.00	24.879999	25.225000	10244800.0	25.160	18.318081	0.225	0.0	NaN
1	PETR4.SA	2007-01-03	25.08	24.004999	25.200001	19898600.0	24.395	17.761106	0.000	0.0	-0.030406
2	PETR4.SA	2007-01-04	24.25	23.700001	24.375000	21060200.0	23.850	17.364315	0.000	0.0	-0.022340
3	PETR4.SA	2007-01-05	23.60	22.549999	23.995001	24864000.0	23.125	16.836473	0.000	0.0	-0.030398
4	PETR4.SA	2007-01-08	23.25	22.900000	23.570000	19440200.0	23.395	17.033047	0.000	0.0	0.011675

Image 2: Dataset snapshot.

After collecting all this data, the first interesting step is to take a look at some statistic, like mean, std, min, max and quartiles for each stock.

As the feature Adjusted Close Price provides a better representation of the price over time, once that it handles splits, dividends, etc., the statistics mentioned above are calculated for this feature.

	PETR4	VALE3	CSN3	JBSS3	SUZB3
Mean	18.956508	31.596306	10.600504	10.018938	22.399299
Std	6.400043	13.744593	5.333533	6.085572	11.194288
Min	3.896351	7.356939	2.273753	2.501025	13.760171
25%	14.690168	23.487716	6.954032	6.182464	14.714258
50%	18.432398	28.694193	8.978834	7.732455	18.720858
75%	23.465368	37.801821	13.860640	11.096821	19.289434
Max	39.535816	102.320000	38.759998	32.449677	76.879997

Table 1: Statistics for the Adjusted Close Price of each stock.

From these statistics, we can extract already some information like:

- The std was always between 30% and 50% of the mean. This represents very well how an emergent market like Brazil behaves. It may bring higher returns but there is also risk within.
- Looking at the min and max values we see how wide is the price range from 2008 until now. Meanwhile, Brazil has passed through political crises, corruptions scandals and corona crisis. PETR4 is a good representation of it. In its worst moment, the company stock price was around 3.89 and in the best moment it was around 39.53.
- Though the std, mean, min and max values may show a very wide range, the 3 quartiles show that most of the data stand in a much smaller range, meaning that most part of the time, the price stays stable and in some very specific time they step up or dow. This goes directly to the goal of this project, that is to know fo 2021 could be one of these unusual moments due to the commodity rally.

Now let’s move forward to get some insights into the visualizations.

2.2. Exploratory Visualization

There are a lot of ways to visualize stock prices like candlestick graphs which take opening and close prices in one graph among other visualizations. But as we are interested in the long term behaviour, that means, long therm variation, the line graph is a better match for this task.

So let’s start visualizing the adjusted closed prices over time for the five companies.

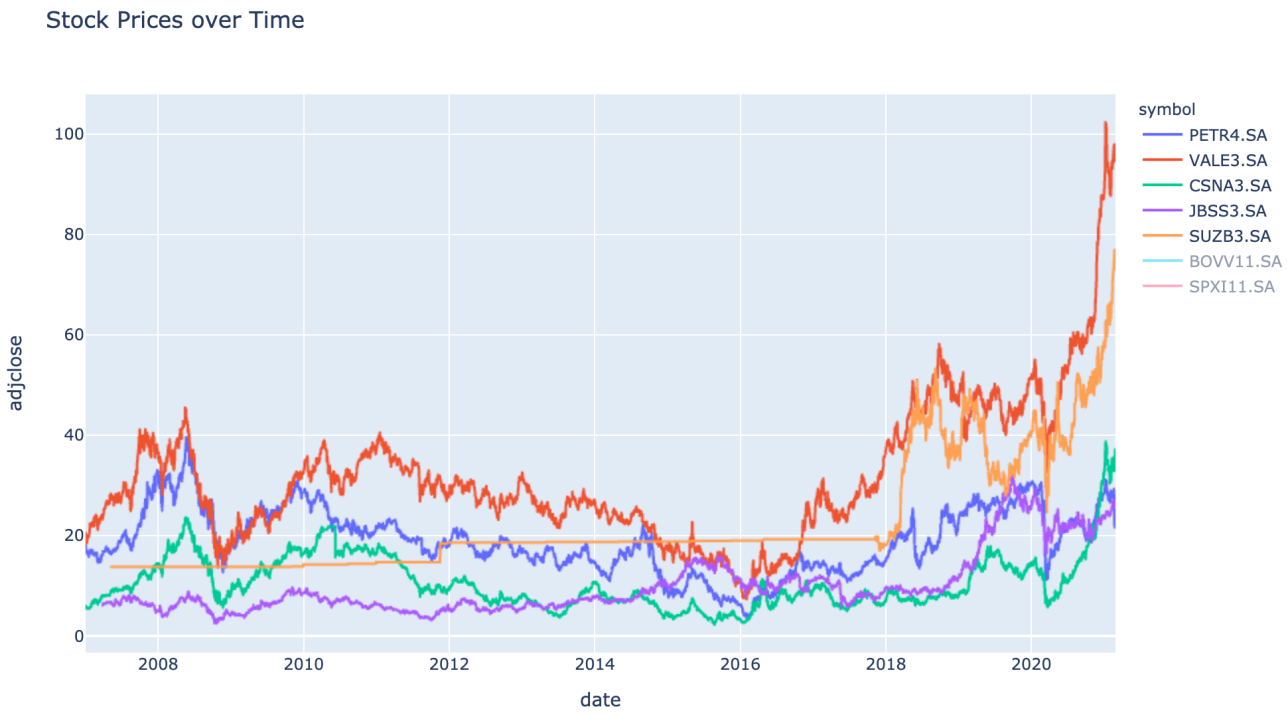

Image 3: Price over time for each stock since 2008.

Analyzing this graph we can conclude the following:

- Among the 5 stocks, VALE3 reached the highest price both in 2008 and also now in 2020/2021;
- A big difference is SUZB3. From 2008 until 2018 the company shows a constant price. It means we can't compare the data before 2018 and it may indicate a failure of registration or even that the company wasn't in the stock market until then;
- After 2018 SUZB3 shows a strong increase in its value, something from around 19.81 to 74.04, increasing 373.75% in 3 years;
- For the other 3 Stocks, we can see that they walk around but did not increase their value at all along all these years;
- Another point to see is these 3 Stocks have today similar prices like 2008 when there was a commodity rally. Though 2020 was atypical because of the corona, we can notice a strong increase in their prices. When we compare 2008 and 2020/2021 we can see similar behaviour;
- PETR4, as a state-owned company, show in this graph all the problems to be managed by the state. We can see its lowest price in 2016 when the impeachment process of President Dilma was running.

Another interesting visualization is the stock trading volume over time. We must understand the trading volume, the flow, as the market fuel. A large flow, a large volume of trading means that a certain directional movement - bullish, for example - is sustained.

In the image above we can visualize this feature for each stock. It might seem a bit chaotic because some stocks volume are so close to each other, but making this graph allowed me to take a deep look one by one and it makes possible to draw the conclusions even for this graph.

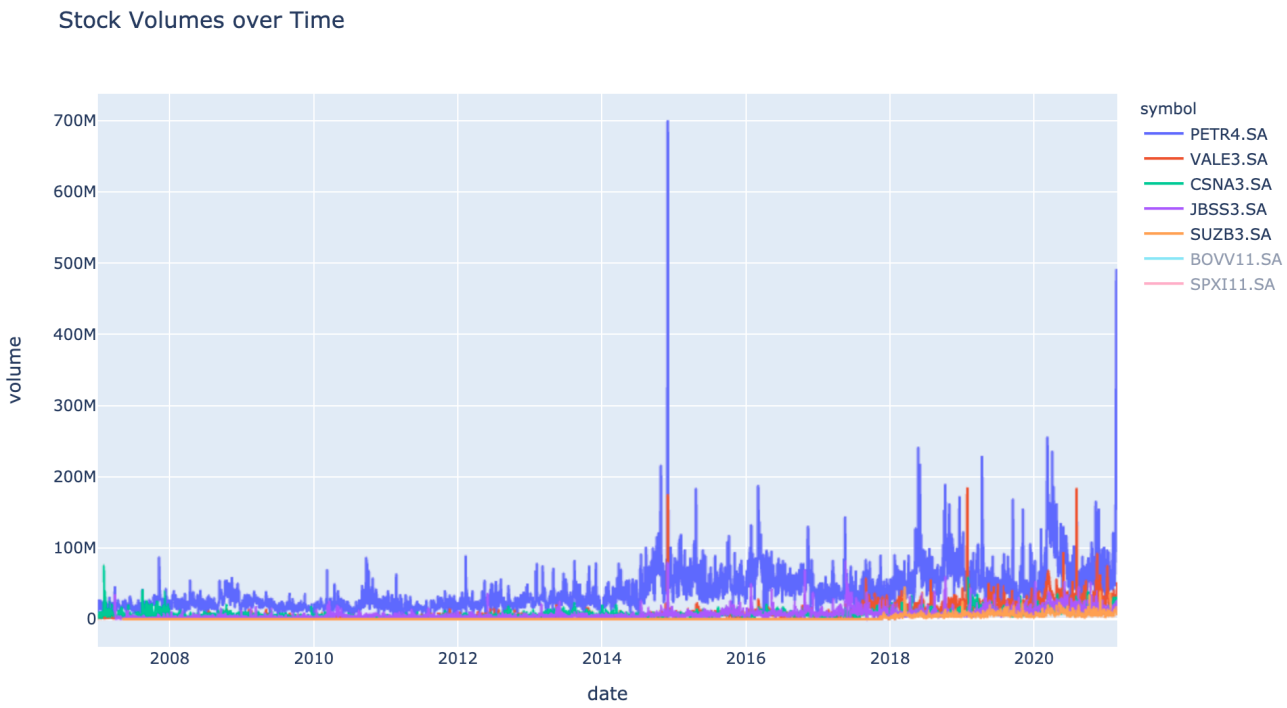

Image 4: Stock volumes over time for each stock since 2008.

Looking stock by stock, separately, we may conclude the following:

- In descending order of volume we have PETR4, VALE3, CSNA3, JBSS3 and SUZB3;
- Though the price is increasing, the volumes are also increasing. It might mean that the investors believe that the companies still at interesting prices;
- Regardless of the share and the date that it started, there is an increase in volume. We can also notice that this behaviour got stronger and for 2020 and 2021 there is a visible growth of the volumes. **It may show that there is really a bullish tendency.**

As all the companies are commodity-based, it might be very interesting to investigate how they are correlated.

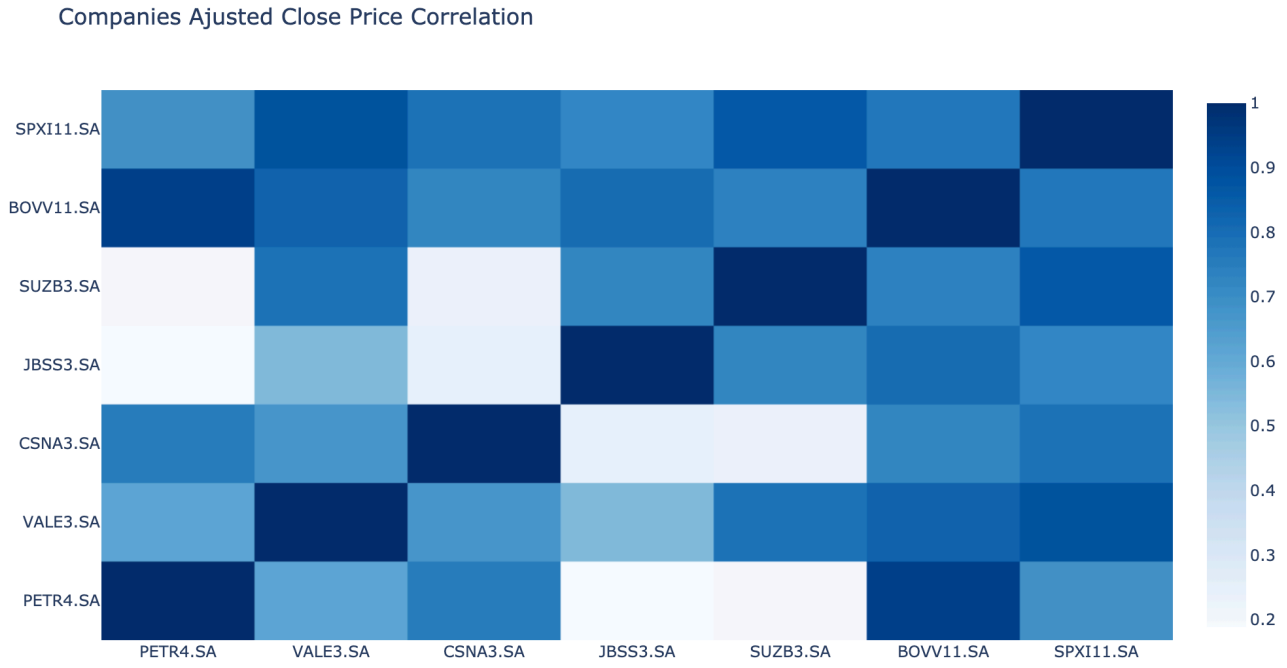

Image 5: Adjusted Close Price Correlation.

In this part, we used also two ETFs, BOVV11 that replicates the Brazilian BOVESPA index and SPXI11 that replicates the S&P500 index. It is pretty useful, once it allows us to see how to import commodity-base companies are in the Brazilian Market and how dependent they are on the American Market as well as correlated to the dollar. We can extract the following information from this image:

- PETR4 and BOVV11 are highly correlated. It is expected because Petrobras is one of the biggest Brazilian companies, so it has a heavyweight on BOVV11;
- VALE3 and SPXI11 are also pretty correlated (0.88). It is very interesting to see how the Brazilian and American market are connected through iron ore performance;
- CSNA3 also has an interesting correlation with SPXI11 (0.78). As it is a steel producer, I expected to see a stronger correlation with VALE3, but the value was just about 0.65. It seems that it is more correlated with oil by PETR4 (0.76) than with steel;
- JBSS3 has its highest correlation with IBOVV11. It shows how Agribusiness play a strong role in the Brazilian economy;
- SUZB3 is also very correlated with SPXI11. It shows also how this kind of business is dependent on the dollar.

Lastly, once the feature “daily return” was created based on the adjusted close price, it is possible to plot the relation between Expected Return and Risk. It is the last piece of this case to decide which company to take in the next steps for modelling and prediction.

Image 6: Risk and Expected Return for each company.

Using this graph, we can select and establish some order in which companies bring more return with less risk within. That means we want companies that lay far in the x-axes and have lower values for y-axes. So as we notice, this visualization is very helpful for making decisions. Through the graph we can state the following:

- The two indexes ETFs have low-risk associate and SPXI11 has the most desired profile: lowest risk and the second-highest expected return;
- PETR4 doesn't seem a good choice at all. High Risk and lowest expected return isn't a good fit for the portfolio;
- **VALE3 was the highest-priced stock in the 2008 rally, it is also the highest 2021, we find it with an interesting position with intermediate values both for expected return and for risk;**

- SUZB3 the second-highest priced stock has both the low expected return and low risk. So it could be a safe paper to have in the portfolio;
- CSNA3 has also both, the highest expected return and the highest risk. It is still interesting to have in the portfolio, once we want to balance SUZB3.
- Of these 5 companies, we should consider, in a real live portfolio, if we want to keep PETR4 and JBSS3. But as we want to create a portfolio for a possible commodity rally, JBSS3 can be easily considered. The only PETR4 due to its low expected return and high risk haven't an interesting profile for the future.

Once that we investigated the stocks and saw that VALE3 brings an interesting combination of expected return and risk, as well as the fact that this company reached the highest value among commodity-based stocks and presents some signs that it will be the same case again, for the rest of this project the data of the company will be used to construct the models and make the predictions for the next 28 days (4 weeks).
Though it could be interesting to replicate the models for the other stocks, it will demand resources from AWS meaning costs that don't need to exist for a capstone project.

2.3. Algorithms and Techniques

This project was done using AWS SageMaker, which means following the AWS pipeline. In practice, it means to follow a different workflow and use models requiring some special configurations. The algorithms, techniques and special configurations used were:

- **Scaling the data**: distance-based algorithms need to receive the inputs in a scaled way in order to produce good results. The models used in this project also takes advantage of the data in this format;
- **Uploading the data to S3**: once the data is ready, it must be uploaded to S3 to be used as input for the AWS training job task. Some algorithms accept the data in CSV format but some demands the input data in JSON format, so a transformation is also required;
- **Feature Engineering**: as it is all about time-series forecasting, feature engineering as Lag, Diff and Rolling Average were used to build a dataset that could be used for the models;
- **Random Forest Regressor**: this simple and well know algorithm was used with all 3 types of features mentioned above in order to produce a fast and robust baseline for the project;
- **AWS DeepAR Forecasting**: algorithm is a supervised learning algorithm for forecasting scalar (one-dimensional) time series using recurrent neural networks (RNN). This means we don't need to do any feature engineering for this algorithm. As it is a native AWS Resource, for the instantiate and using it works pretty good with the overall goal of the project, that is developing solutions using SageMaker.
  - This algorithm works creating, automatically, features based on time series periods. To facilitate learning time-dependent patterns, such as spikes during weekends, DeepAR automatically creates feature time series based on the frequency of the target time series. For example, DeepAR creates two feature time series (day of the month and day of the year) for a weekly time series frequency. It uses these derived feature time series with the custom feature time series that you provide during training and inference. The following figure shows two of these derived time series features: $u_{i,1,t}$ represents the hour of the day and $u_{i,2,t}$ the day of the week. The figure below shows this feature generation process.

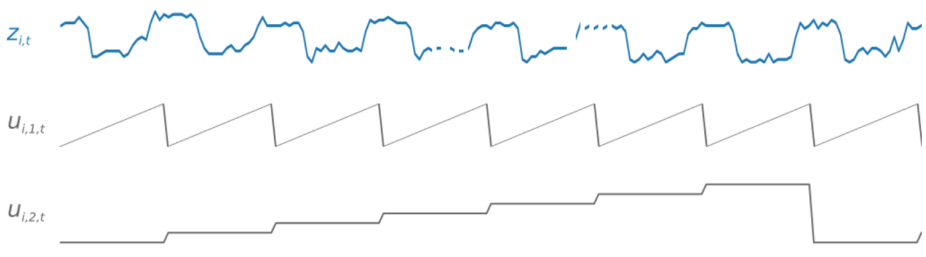

Image 7: DeepAR time-series feature generation. Source: https://docs.aws.amazon.com/sagemaker/latest/dg/

- Mainly because of this feature generating characteristic, this algorithm worth a try in the project. If it shows a good performance, we should end up having an algorithm that doesn't need any feature engineering or something like that. It could just take adjusted close price, context and prediction length and that is all. Simple and easy to work with, which are desired qualities to have in any project that involves machine learning.

- **LSTM Model with TensorFlow**: it is known the power of LSTM in forecasting problems, and this model also gives the opportunity to implement my own training script. The NN itself is not complex, it is compound by 3 LSTM layers with 264, 128 and 64 units each, one Flatten layer and two Dense layers with 32 and 1 units each;
  - To understand a LSTM layer, first we need to take a look at **RNNs cells**. A recurrent neural network looks very much like a feedforward neural network, except it also has connections pointing backward. Since the output of a recurrent neuron at time step *t* is a function of all the inputs from previous time steps, you could say it has a form of memory. A part of a neural network that preserves some state across time steps is called a memory cell (or simply a cell). This allows it to exhibit temporal dynamic behavior. That is why this type of neural network can handle time-series very well;
  - **The Long Short-Term Memory (LSTM)** cell is a special type that derivates from the RNN branch. The following image helps us to make it less abstract and to explain how it works.

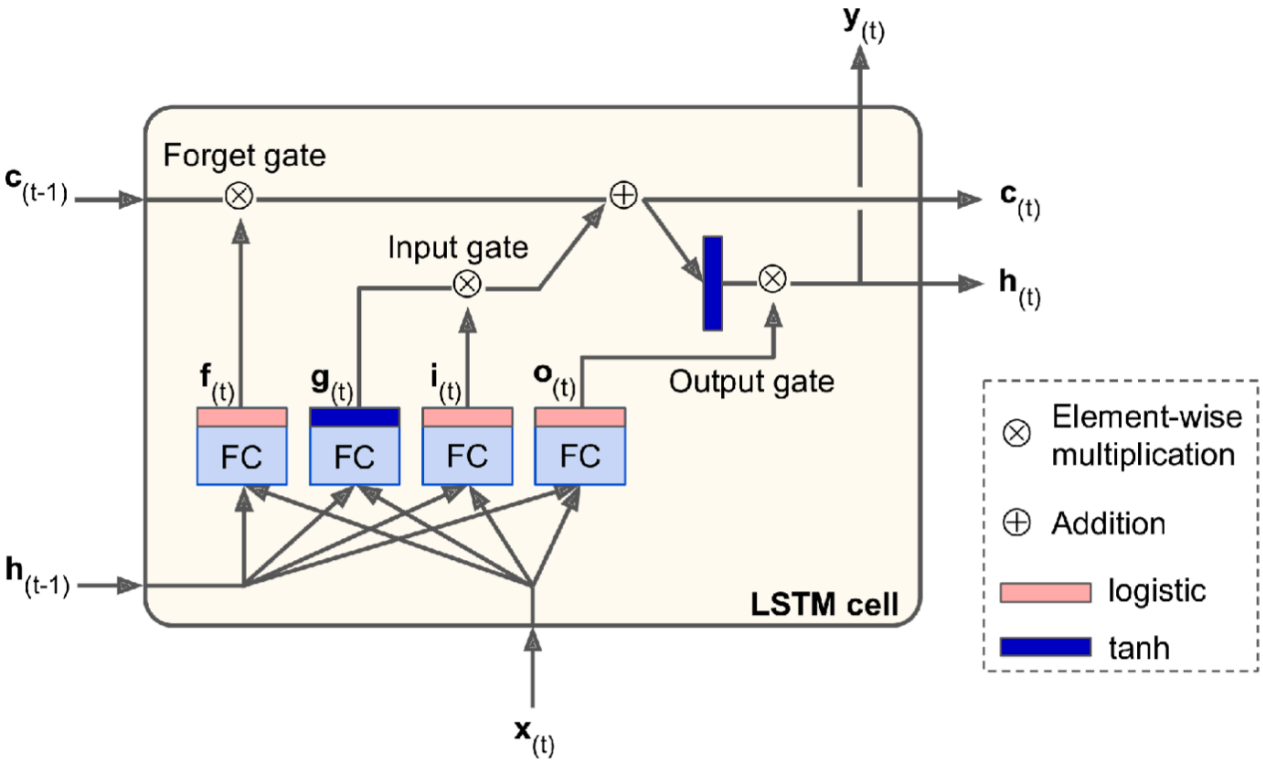

Figure 15-9. LSTM cell

Image 8: LSTM under the hood. Source: Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow- O'reilly (2019).

- The key idea is that the network can learn what to store in the long-term state, what to throw away, and what to read from it. As the long-term state $c_{(t-1)}$ traverses the network from left to right, we can see that it first goes through a forget gate, dropping some memories, and then it adds some new memories via the addition operation (which adds the memories that were selected by an input gate). The result $c_{(t)}$ is sent straight out, without any further transformation. So, at each time step, some memories are dropped and some memories are added. Moreover, after the addition operation, the long-term state is copied and passed through the tanh function, and then the result is filtered by the output gate. This produces the short-term state $h_{(t)}$ (which is equal to the cell's output for this time step, $y_{(t)}$).

  - Another critical point is that LSTM prevents backpropagated errors from vanishing or exploding. Instead, errors can flow backwards through unlimited numbers of virtual layers unfolded in space. That is, LSTM can learn tasks that require memories of events that happened thousands or even millions of discrete time steps earlier.
  - For all the points mentioned above, LSTM is a good match to work with time series forecasting and that is why I would like to work with it in my model.
- **Deployment**: in order to make predictions for the last two models, their deployment was done using the high-level API. After this deployment, their URI could be used for latter applications like a website that could make predictions with the model through a POST Request.

2.4. Benchmark

Since it is a personal project the benchmark can be established from a simple model or even a statistical model. For this case, I will use Random Forest, mentioned above, as a regression algorithm to fit the data and to make predictions for the next days. For a general-purpose, it would generate a simple, fast and strong baseline that guides the work pretty well.

3. Methodology

3.1. Data Preprocessing

The first point is to scale the data so all the models can perform better. It was done using the MinMaxScaler from scikit-learn.

After that, the data for all companies were split into train and test, keeping 28 instances for the test set and the rest for the training set.
As the main goal is to be able to predict the stock price behaviour for the next 28 days (4 weeks), the point is to decide how far in the past should we look. As a first try, the start date for this part of the was 02 January of 2018. As we achieved good results, there was no need to extend this context length.

In the next step, the data were saved on the data folder and just the data for VALE3 were uploaded to S3. The image below shows the code used to upload the data to S3.

```
# Creating specific configuration
prefix = "capstone-project"
data_dir = "./data"
paths = {}

# Addressing the data on the disc
train_key = os.path.join(data_dir, "train_{}.csv".format("vale"))
test_key = os.path.join(data_dir, "test_{}.csv".format("vale"))

# Path where the files will be saved
train_prefix = "{}/{}".format(prefix, "train_{}".format("vale"))
test_prefix = "{}/{}".format(prefix, "test_{}".format("vale"))

# Uploading to S3
paths["train"] = session.upload_data(train_key, bucket = bucket, key_prefix = train_prefix)
paths["test"] = session.upload_data(test_key, bucket = bucket, key_prefix = test_prefix)
```

Image 9: Uploading the data to S3.

In three other moments, data preparations were needed. They were:

- Feature engineering for the Random Forest Model by creating a feature called diff (the difference between the present instance and the previous one), 5 lags features, and one moving average feature.
- The DeepAR model demands the data in JSON format, so functions created were creating for this formatting task and the training data was formatted as the algorithm required and uploaded to the S3.
- Standard practice is to feed LSTM with lags, so the feature engineering required here was just the created a couple of lags.

3.2. Implementation

As instructed, the project was implemented using Jupyter Notebook on AWS SageMaker.
The implementation followed a sequence, first creating and training and evaluating the Random Forest model, as it is the benchmark. After this, the DeepAR model and then the LSTM model was also created, trained and evaluated. Here are the major points for each model.

Random Forest:

- Instantiate a Random Forest Regressor object from scikit-learn packet with just one hyperparameter - n_estimators = 1000 - this is 1000 decision trees. As we want to keep the baseline simple, it doesn't need any extra hyperparameter;
- Training the model;
- Making predictions rescale the data back to the normal price;
- Measure the RMSE and MAE;

DeepAR:

- Formatting and Uploading the train data to S3;
- Instantiate the AWS DeepAR algorithm by assign and URI image;
- Using the AWS Estimator with the image to create a DeepAR model;
- Setting up the hyperparameters. These steps are shown in the figure below;

```
# Setting up the output path
output_path = "s3://{}/{}/output".format(bucket, prefix)

# Instatiating the model
regressor_deepar = sagemaker.estimator.Estimator(image_uri = deepar_image,
                                                 role = role,
                                                 instance_count = 1,
                                                 instance_type = "ml.c4.4xlarge",
                                                 output_path = output_path,
                                                 sagemaker_session = session)
```

```
# Hyperparamenters
freq = 'D'
prediction_length = timespan # Four weeks
context_length = train["VALE3.SA"].shape[0] - prediction_length

hyperparameters = {
    "epochs": str(50),
    "time_freq": freq,
    "prediction_length": str(prediction_length),
    "context_length": str(context_length),
    "num_cells": "50",
    "num_layers": "2",
    "mini_batch_size": "128",
    "learning_rate": "0.001",
    "early_stopping_patience": "10"
}
```

```
regressor_deepar.set_hyperparameters(**hyperparameters)
```

Image 10: DeepAR Setup.

- Training the model with the JSON-formatted data;
- Deploy the model with .deploy() function indicating “instance_type” and “initial_instance_count”;
- Making predictions rescale the data back to the normal price;
- Measure the RMSE and MAE.

LSTM Model:

- Creating a training script (train.py). This script contains my own designed model, functions to load and prepare the data, function to accept and parse input arguments and training and save the model;
- The image above shows how the model was defined;

```
# Defining the model
def model(X_train, y_train, epochs, batch_size, early_stop_patient):
    """Generates the model instanciating the LSTMStockPredictor and makes it read
    for use"""
    model = keras.models.Sequential([
        keras.layers.LSTM(units = 264, return_sequences = True, input_shape= (X_train.shape[1], 1)),
        keras.layers.Dropout(0.2),
        keras.layers.LSTM(units = 128, return_sequences = True),
        keras.layers.Dropout(0.2),
        keras.layers.LSTM(units = 64, return_sequences = False),
        keras.layers.Flatten(),
        keras.layers.Dropout(0.2),
        keras.layers.Dense(units = 32),
        keras.layers.Dropout(0.2),
        keras.layers.Dense(units = 1)])

    model.compile(optimizer = 'adam',
                  loss = 'mse')

    early_stop = keras.callbacks.EarlyStopping(monitor = "loss",
                                               patience = early_stop_patient,
                                               restore_best_weights = True)

    model.fit(X_train, y_train,
              epochs = epochs,
              batch_size = batch_size,
              callbacks = [early_stop])

    #model.evaluate(X_test, y_test)
    return model
```

Image 11: LSTM Model.

- Instantiate the LSTM model with the AWS TensorFlow estimator indicating the train.py in “entry_point” field;
- Training the model. These steps are shown below;

```
# Setting up the output path
output_path = "s3://{}/{}/output".format(bucket, prefix)


regressor_tf = TensorFlow(
    entry_point='train.py',
    #source_dir='source', # directory of your training script
    role=role,
    framework_version='2.3.0',
    model_dir = False,
    py_version='py37',
    instance_type='ml.c4.xlarge',
    instance_count=1,
    output_path=output_path,
    hyperparameters={
        'batch-size':32,
        'epochs':100})
```

```
regressor_tf.fit(paths["train"])
```

Image 12: LSTM Setup.

- Deploy the model with .deploy() function indicating “instance_type” and “initial_instance_count”;
- Making predictions rescale the data back to the normal price;
- Measure the RMSE and MAE.

3.3. Refinement

In this project, there were 2 major refinements. They are:

Creating the training algorithm:

- As it was my first time producing a training algorithm, the whole process was a mixture of development and refinement;
- The greatest refinement in this part was to incorporate the data pre-processing in the loading function in the train.py. It allows the user to just give the adjusted price feature as input and the creation of the lags is done in the training script. Thinking on deployment, it will allow the model to avoid the use of tools like the AWS Lamda function, which we need to pay each time it gets called.

Creating the LSTM model:

- Testing different layers like the Flatten one;
- Testing different units combination;
- Testing different values for the dropout layers;
- Testing different optimizer like Adam or AdamW.

4. Results

4.1. Model Evaluation and Validation

Now that we have our models trained and evaluated on a test set, we can compare the metrics and use visualization to get insights on how good they are and how close they were in the forecast of stock prices for the last 28 days.
The table below shows the RMSE and MAE for each model

	RMSE	MAE
Random Forest	4,777	4,254
DeepAR	61,347	61,181
LSTM	3,930	3,311

Table 2: RMSE and MAE for each model.

There is a couple of things here that we can be discussed about each performance:

- **Random Forest**, though simple, performed pretty good with a very basic feature engineering for it. The two errors were small and though it is just a baseline, it was a good job forecasting the next 28 days.
- **DeepAR**, though a very good algorithm for time-series forecasting, had a mediocre performance. It can be explained by the fact that this algorithm is designed to cover repetitive paters like power consumption, and as stock prices behaviour does not fit in this profile, it is a good reason to explain such failure.
  - Another negative point is, this algorithm took 972 seconds to train and as in AWS, we pay as we use it, this higher time (compared with LSTM) means unnecessary costs.
- **LSTM** did an incredible job, as expected. It demands only basic feature engineering by generating some lags and achieve a very good result in both metrics. Using the Random Forest Model as a baseline, we can say this model bit our baseline by far and can be already be used in the feature.
  - It took 554 seconds to train and as explained above, it will mean also lower costs when compared with DeepAR. It gets even more critical when we think on retrain this algorithm every two weeks. So another advantage for this model.

Now let's visualize this performance difference. First with all 3 models and the real prince and then without DeepAR, once it lays so low down in the graphic that it disrupts the visualization.

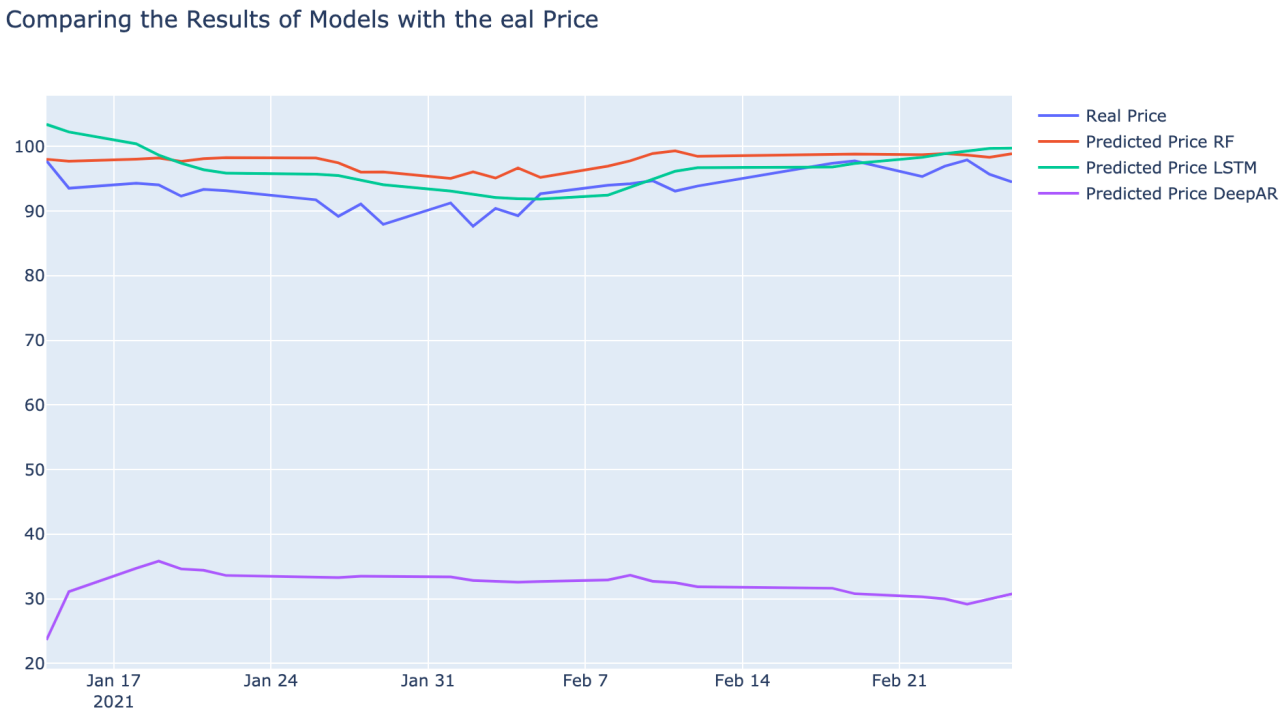

Image 13: Results for each Model compared to the Real Price on the test set.

As expected, here we see how far below lays the predictions of DeepAR. This model was unexpected bad and should no more be considered. Let's focus on the other two.

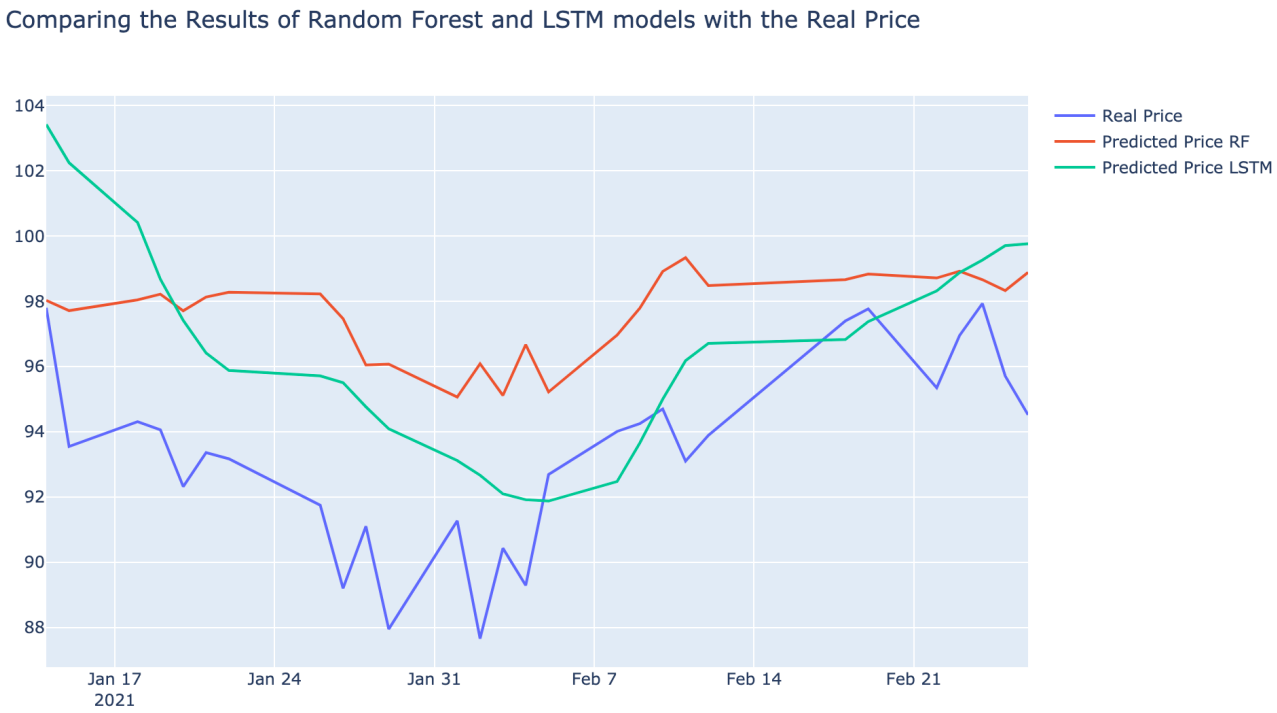

Image 14: Results for Random Forest and LSTM compared to the Real Price on the test set.

4.2. Justification

As mentioned before, the benchmark (Random Forest) produced, with simple feature engineering, a great model reaching an RMSE of 4,777 and MAE of 4,254 and bit it was not trivial as expected.

On one hand, the DeepAR model produced a mediocre model with an RMSE of 61,347 and an MAE of 61,181. Though this model is a better fit when there are repetitive patterns, such high values were really unexpected and in someway a big disappointment.
At least the implementation was straightforward and the data preparation was a great exercise.

On the other hand, my LSTM model did great with such a simple structure and just a few refinements. It scored 3,93 for RMSE and 3,311 for MAE. This was also visible on the graphs showed before where the LSTM model replicates the real price very well.

4.3. Next Steps

There is a lot of space for improvement in this work, like:

- Make use of SageMaker Hyperparameters Tuning to improve the LSTM model;
- Try more and different layers in the LSTM model;
- Try out a model based on Convolutional Networks, they aren't known as great for the forecasting job but they can still do some great job for time-series forecasting;
- Create an application where the user may select the stock, the prediction length and receive the predictions for these days. The model could be made available through the endpoint created on SageMaker and if needed it could use the AWS Lambda for any pre-processing.

5. References

[1] https://en.wikipedia.org/wiki/Fundamental_analysis#The_two_analytical_models
[2] https://en.wikipedia.org/wiki/Technical_analysis
[3] https://www.thebalance.com/brazil-and-commodities-808912
[4] https://www.nasdaq.com/articles/3-reasons-why-commodities-etfs-may-rally-in-2021-2021-01-15
[5] https://www.reuters.com/article/column-russell-commodities-yearahead-idUSL1N2IQ0A2
[6] https://plusmining.com/en/commodities-rally-is-projected-to-2021-the-coronavirus-would-mark-a-milestone-in-the-cycle-potentially-leaving-years-of-weak-prices-behind/
[7] https://www.fxempire.com/forecasts/article/speculators-bet-on-a-continued-commodity-rally-in-2021-690009
[8] https://www.kaggle.com/miracl16/tesla-stock-price-prediction-lstm-vs-gru
[9] https://www.kaggle.com/fatmakursun/tesla-stock-price-prediction
[10] https://www.kaggle.com/akanksha496/stock-price-prediction-lstm
[11] https://www.kaggle.com/raoulma/ny-stock-price-prediction-rnn-lstm-gru
[12] https://www.kaggle.com/biphili/time-series-data-analysis-stock-price-code-12#5.Forecasting-Stock-Price
[13] https://towardsdatascience.com/python-for-finance-stock-portfolio-analyses-6da4c3e61054
[14] Hands-on machine learning with Scikit-Learn, Keras and TensorFlow. A. Géron. O'Reilly Media, Sebastopol, CA, (2019)
[15] https://en.wikipedia.org/wiki/Recurrent_neural_network#Long_short-term_memory