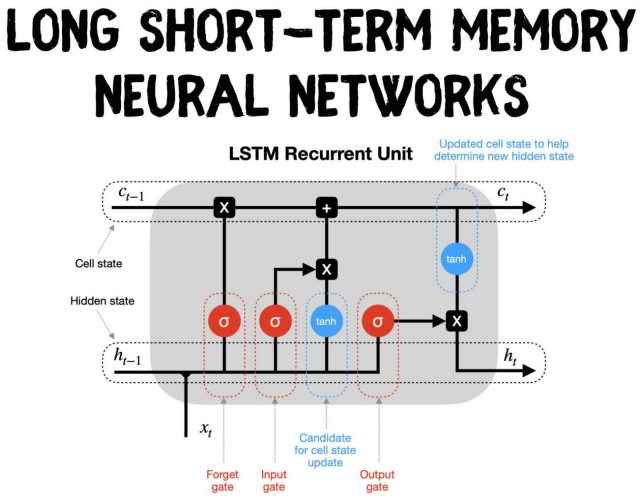
STOCK PREDICTION PROJECT

We decided to do Stock Prediction because we wanted to apply ANN to an extremely fascinating and at the same time unexpected field: the stock world.
As we know, there are different types of neural networks:

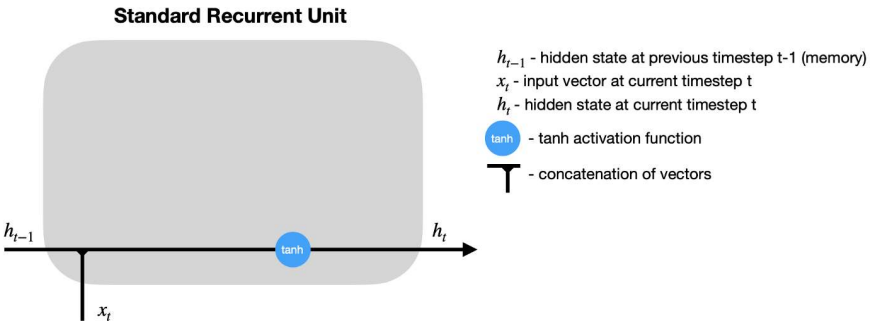
- ANN: Artificial Neural Network
- CNN: Convolutional Neural Network
- RNN: Recurrent Neural Network

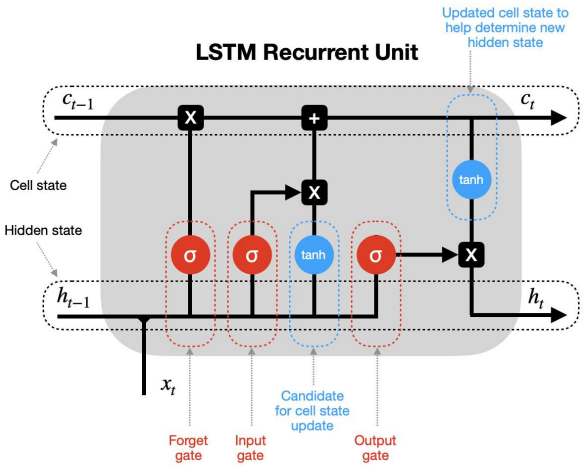
ANNs consist of 3 layers – Input, Hidden and Output.
The input layer accepts the inputs, the hidden layer processes the inputs, and the output layer produces the result. Essentially, each layer tries to learn certain weights.
CNNs are being used across different applications and domains, and they're especially prevalent in image and video processing projects. They essentially use Kernels to extract the relevant features from the input using the convolution operation.
RNNs have a recurrent connection on the hidden state. Therefore, it can capture the sequential information present in the input data, for example dependency between the words in the text while making predictions.
Therefore, to predict the future closing prices of stocks we need to use RNN.
The problem in RNNs is that they suffer from a short-term memory, due to a gradient problem that comes out when analyzing long sequences. Fortunately, we have two more advanced RNNs that can analyze long data sequences: LSTM and GRU.
We decided to stick to the most popular one: the Long Short-Term Memory.



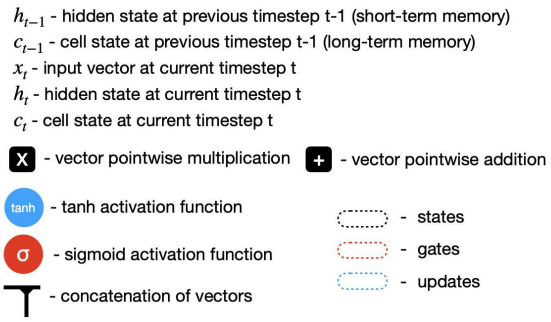
But what is the difference between a normal RNN and a LSTM?
Well, the main difference is how the Recurrent Unit is structured: a normal recurrent unit just combines the previous hidden state with the new input and passes it through the activation function; after the hidden state is calculated at time-step t, it is passed back to the recurrent unit and combined with the input at time-step t+1, to calculate the new hidden state at time-step t+1. This process repeats for t+2, t+3, …, t+n, until the predefined number (n) of time-steps is reached.

Instead, the LSTM Recurrent Unit is much more complex: to make the long story short



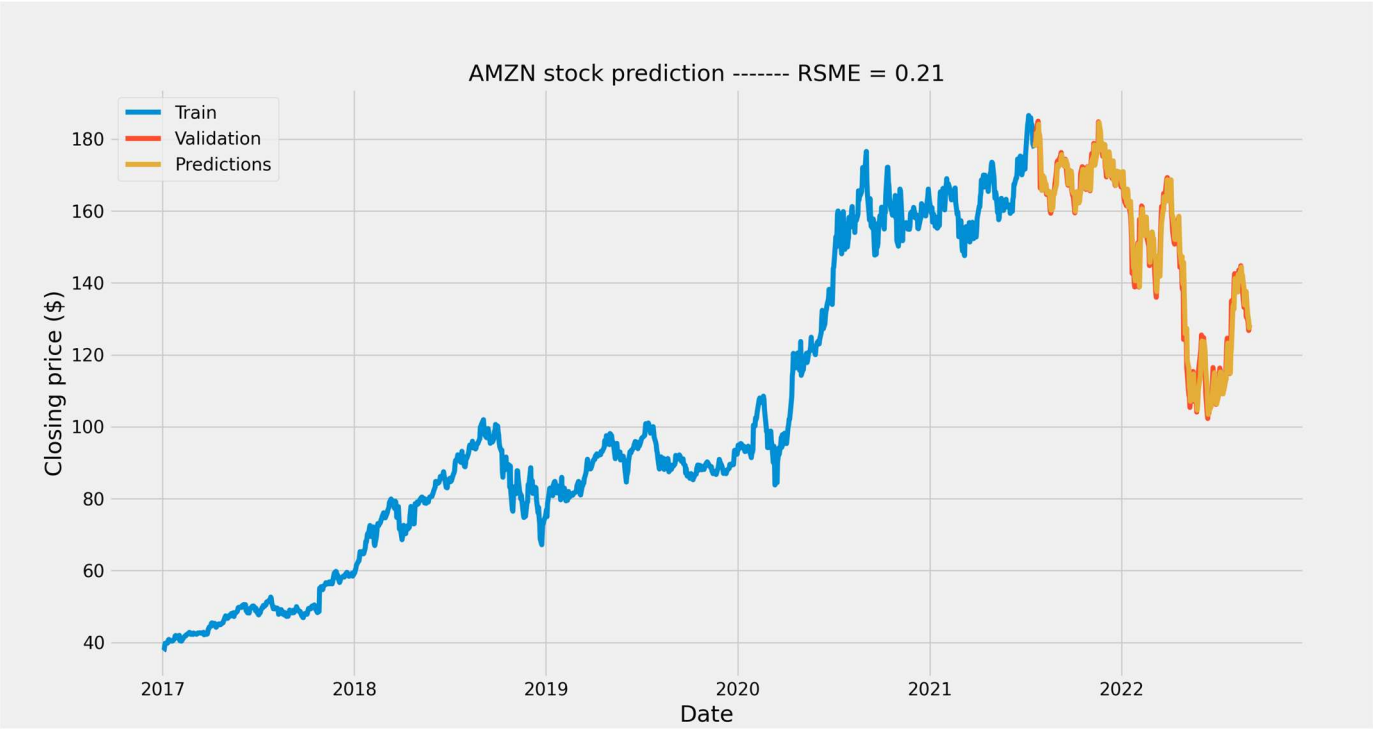


LSTM is able to loop until the end of the sequence.



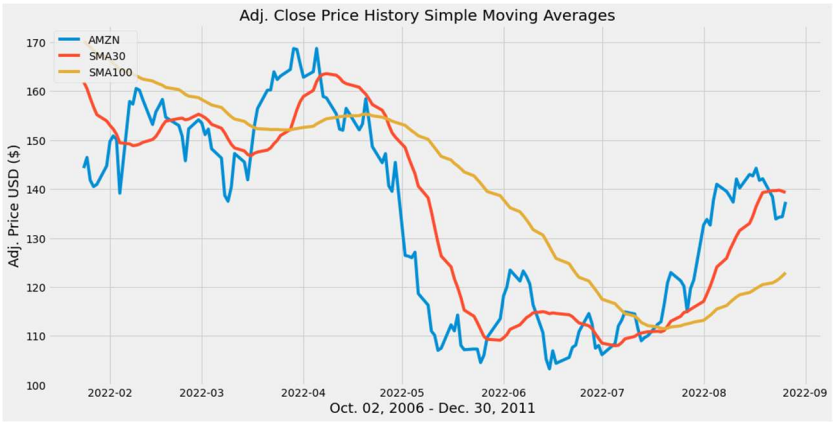
The main problem of this project was how to pass the data to the neural network so that it would be able to analyze it and gives back the prediction it makes. Therefore, we had to do some "work" on the data. First, to predict the next closing price day of the stock we are going to use the previous 60 "closing price day". For this reason, we decided to use the data from Yahoo Finance, that has a great feature that let you download different information about a specific ticker.
Since we're interested only in the closing price, we filter out all the other information and we download the dataset of about 4 and a half years.
We can visualize this graph by plotting it, so we can get an idea of how much the stock gained or lost in this period, and then we're going to split the dataset in two: the train model and the test model. We decided to do an 80-20 % splitting.
Since neuron can take only values between 0 and 1, we needed to normalize our data. Furthermore, we couldn't just pass it as a Pandas dataset, but we needed to pass it as a NumPy array.
Now the hardest part: we want to use the previous 60 days to predict the stock price, so we are going to loop len(train_data)-60—times, and each time we are going to create mini datasets of length=60 where the previous closing price are contained < *x_train* > and we are going to build another dataset < *y_train* > where we are going to put the next day that the neural network should guess.
We then need to reshape our array since LSTM takes only 3-dimensional array, so we are going to add a third dimensionality that's just going to be of length 1.
Finally, we can build our LSTM model. We decided to go with TensorFlow and Keras instead of PyTorch because we had some impressive result and because the implementation is much faster and easier.

To create our model, we need to make it sequential; then we can add out LSTM layer of dimensionality 50, where we also pass the input shape. We need to set *return_sequences=True* because is necessary to stack LSTM layers so the consequent LSTM layer has a three-dimensional sequence input, and input_shape is the shape of the training dataset. Finally, we add a Dense layer of output 1 that is going to be our final prediction result.
To compile our model, we are going to use the Adam Optimizer and to calculate our loss we will use the Mean Squared Error. After we trained the model, we will have to create the testing dataset, in the same way we did before but this time we are not going to create the y_train since it will be the predicted result of our neural network. Now we are able to let the model predicts its results, and since we normalized the dataset, we need to inverse transform the *predictions*. To have a look at the predicted result of how much accurate it is, we will use the round squared mean error (RSME). We can observe that in our case for the Amazon ticker the RSME=0.21, which is actually a really good result.

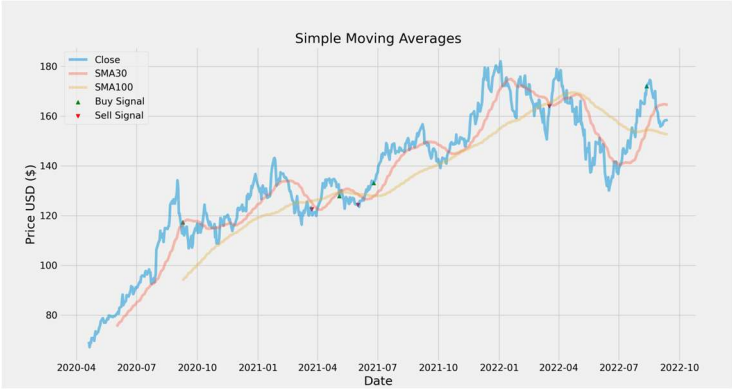


Furthermore, we decided to predict also the next week. The theory is always the same, but this time when we predict the next day, we are going to use that prediction as a true data to predict the day after that. Since this weekly prediction is based on other daily predictions is really a guess and it's probably not reliable as a daily prediction would be.

We tried to take our project to the next level and tried to implement a program that would be able to buy and sell based on the weekly prediction. To do that we used the Simple Moving Average (SMA) to make it automatic. Moving averages are one of the core indicators in technical analysis, and there are a variety of different versions. SMA is the easiest moving average to construct. It is simply the average price over the specified period. The average is called "moving" because it is plotted on the chart bar by bar, forming a line that moves along the chart as the average value changes.



So, we decided to implement a 30 days SMA and a 100 days SMA. This is because SMA Crossing SMA is another common trading signal. When a short period SMA crosses above a long period SMA, you may want to go long. You may want to go short when the short-term SMA crosses back below the long-term SMA.



This is exactly what we did, but unfortunately as we can see this strategy doesn't work always very well. In fact, when the price goes either straight up or straight down, the result is pretty good; if we take for example the graph below, we can see that when there's a "*stall position*" of the stock, it gets crazy and it buys and sells when it should not do it, because of this faltering trend.

So, we think that SMA is not a good way to implement an algorithm that is able to buy and sell by itself, and it is needed much more factor that makes it more efficient.