**STOCK DOCS**

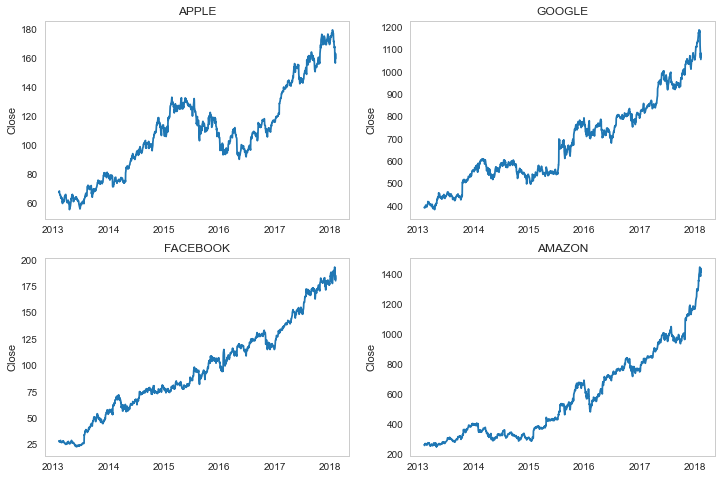
1. **About the Data**
2. **Data Exploration**
3. **Prophet Modeling**
4. **RNN Modeling**
5. **About the data**

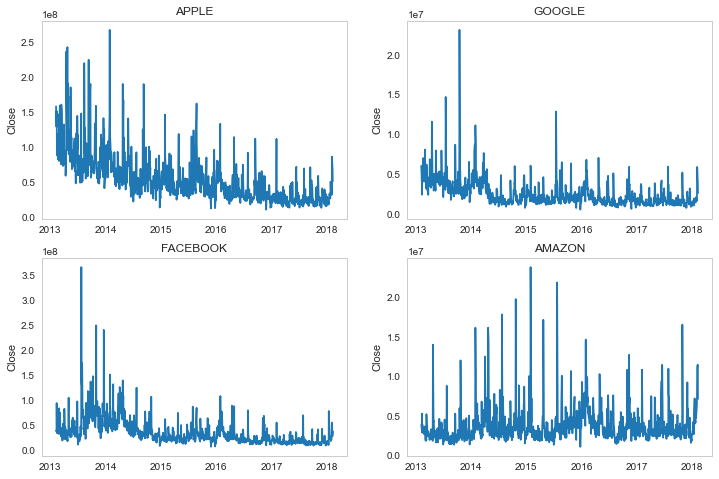
Our data is the historical data of the S&P 500 from Yahoo Finance. This data features stock prices from the year 2013 to 2018. The S and P 500 contains securities of 500 companies and we have decided to use the stock prices of GOOGLE, AMAZON, FACEBOOK and APPLE.

1. **Data Exploration**

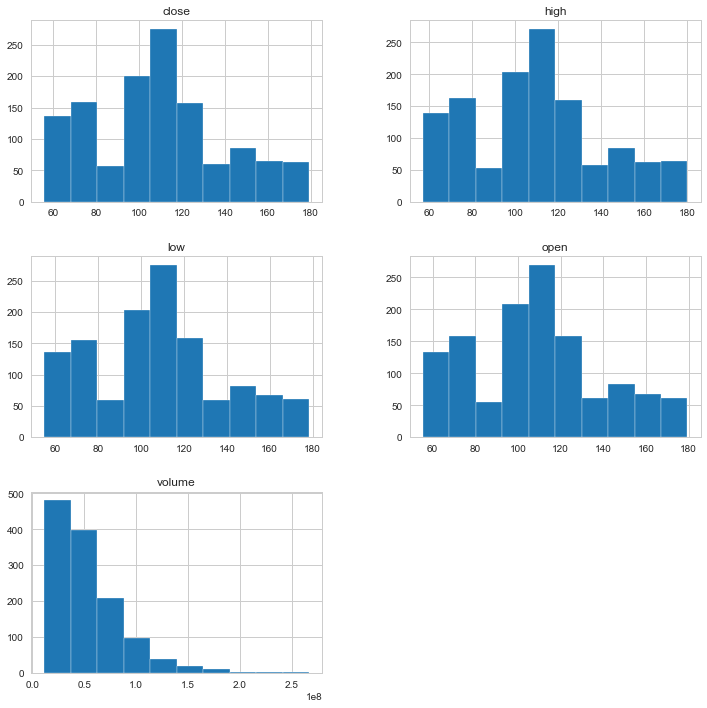
We took a look at the datapoints in the dataset to see the trend in it.

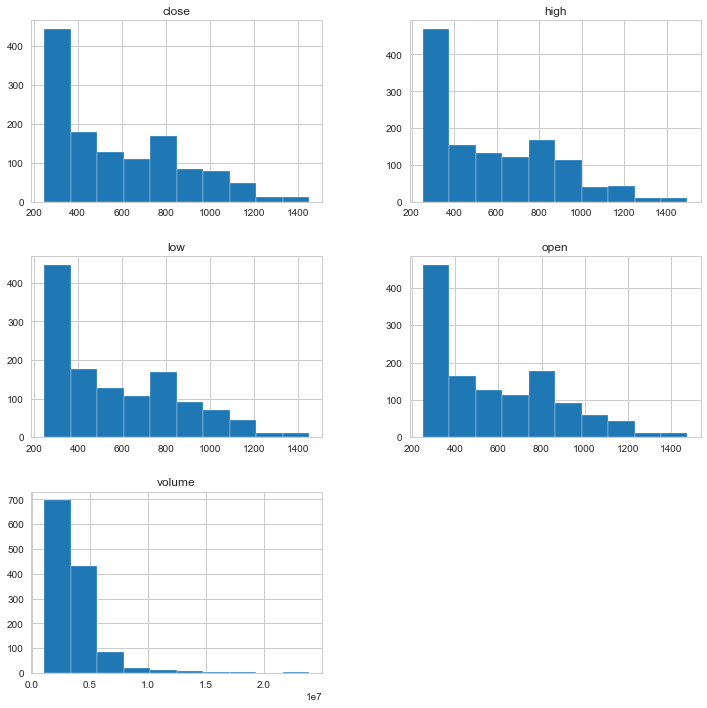
Below are stock trends in each companies’ prices.

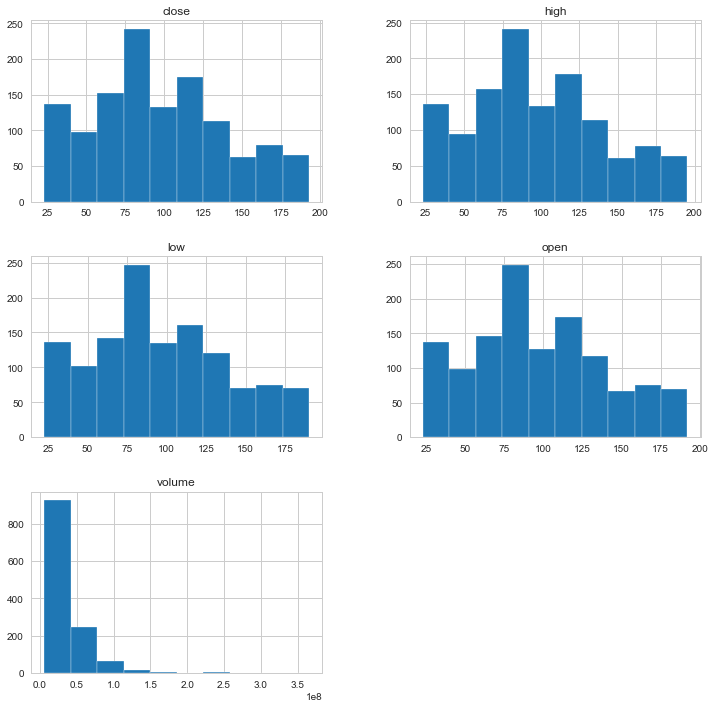
The we take a look at the stock volume from that time.

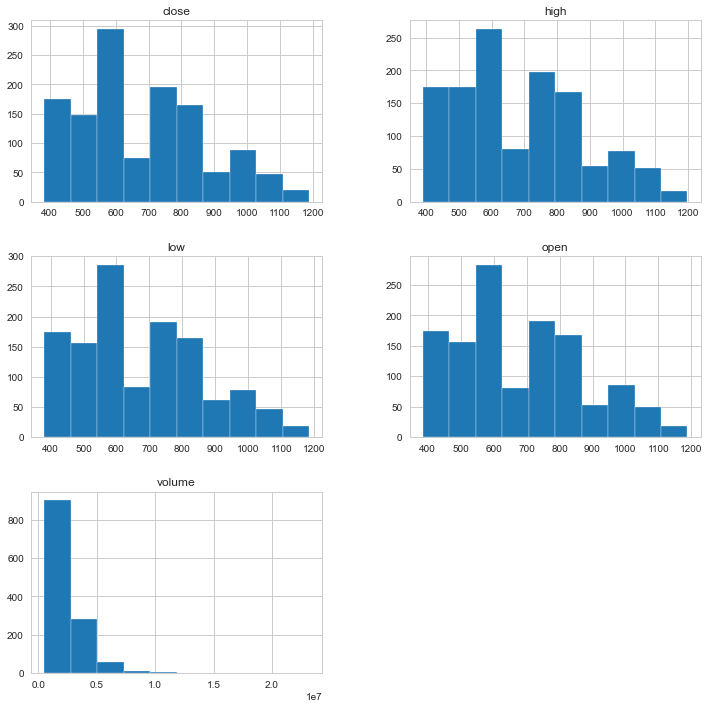
Also, we look at the moving average of the prices. The **moving average** (MA) is a simple technical analysis tool that smooths out price data by creating a constantly updated **average** price. The **average** is taken over a specific period of time, like 10 days, 20 minutes, 30 weeks or any time period the trader choose.

Apple’s Moving Average

Amazon’s Moving Average

Facebook’s moving Average

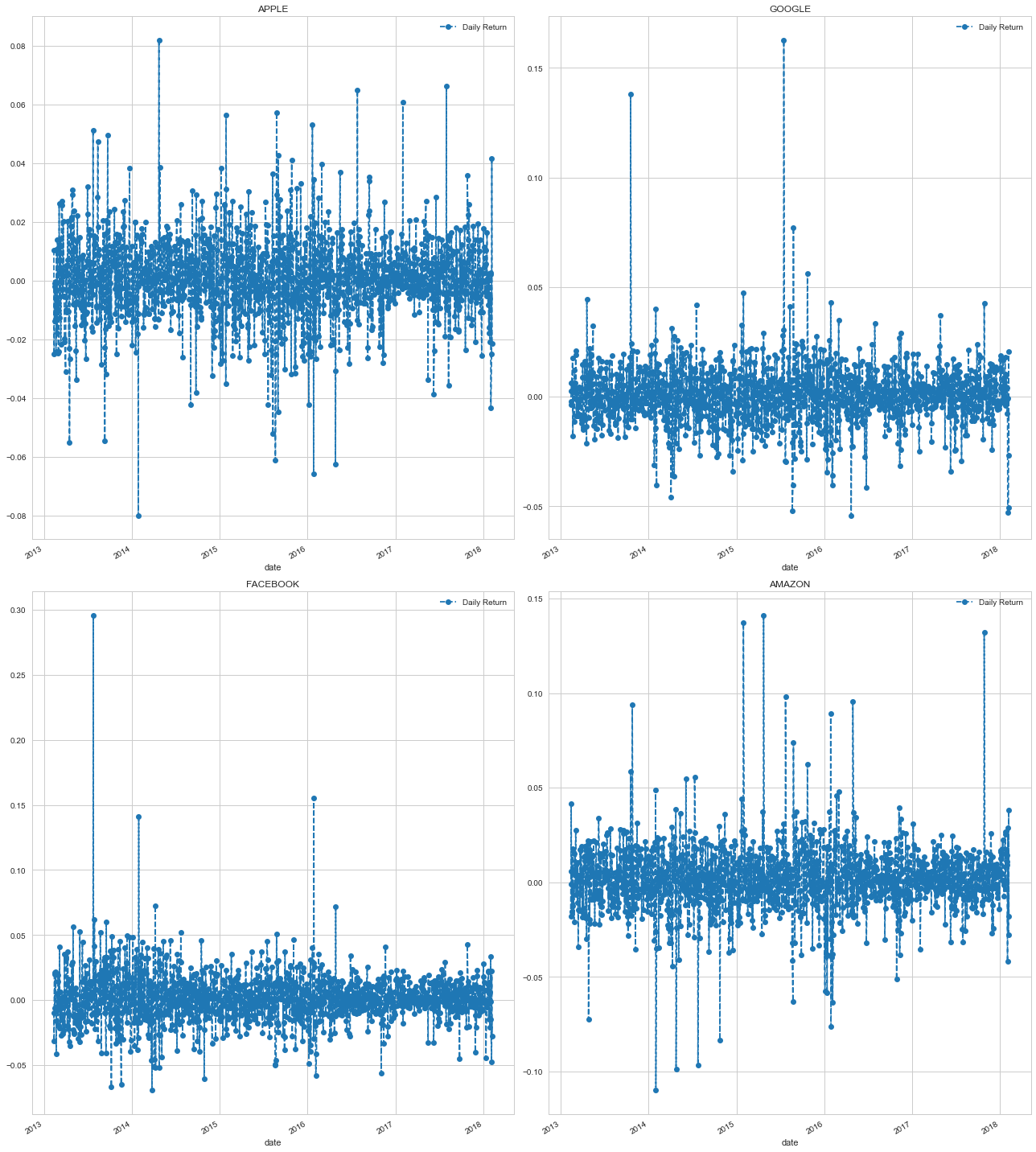
Google’s Moving Average



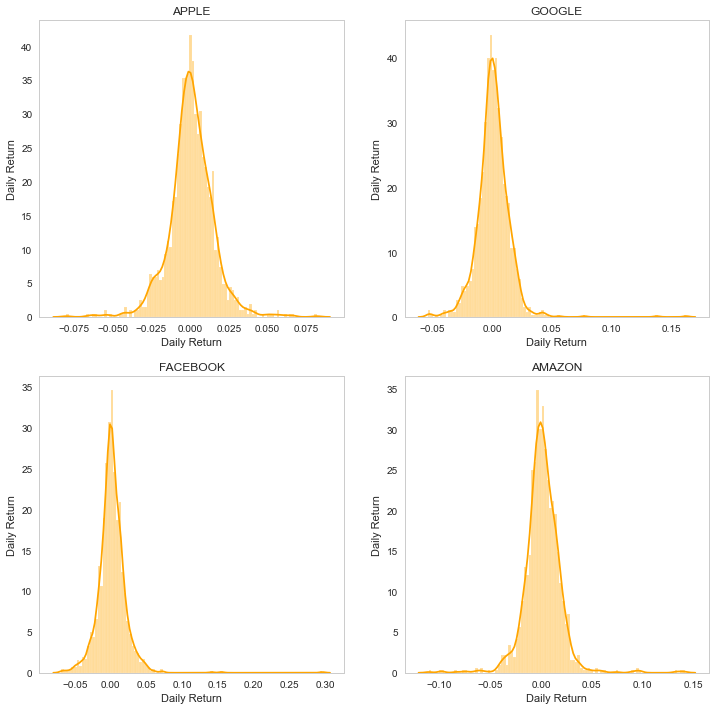
Overall plot of the moving average



We also visualize the risk of these companies stock. This is informed by the daily return on each stocks

****

Distribution of daily return



1. **Modeling with Prophet**

**Prophet Introduction:** Prophet is Facebook's library for time series forecasting. In my opinion, Prophet works best with datasets that are hugely influenced by seasonality (electricity bills, restaurant visitors etc.)

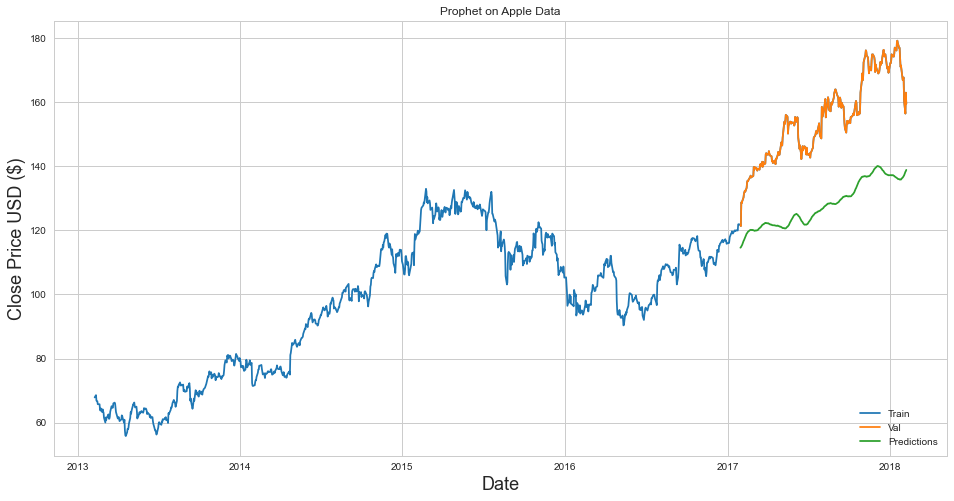
**Steps for using Prophet:**

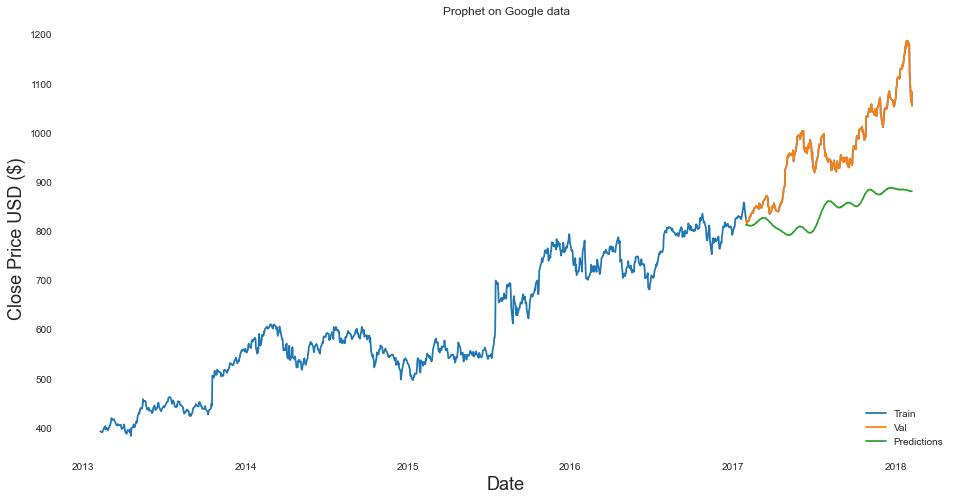
* Make sure you replace closing price for y and date for ds.
* Fit that DataFrame to Prophet in order to detect future patterns.
* Predict the upper and lower prices of the closing price.

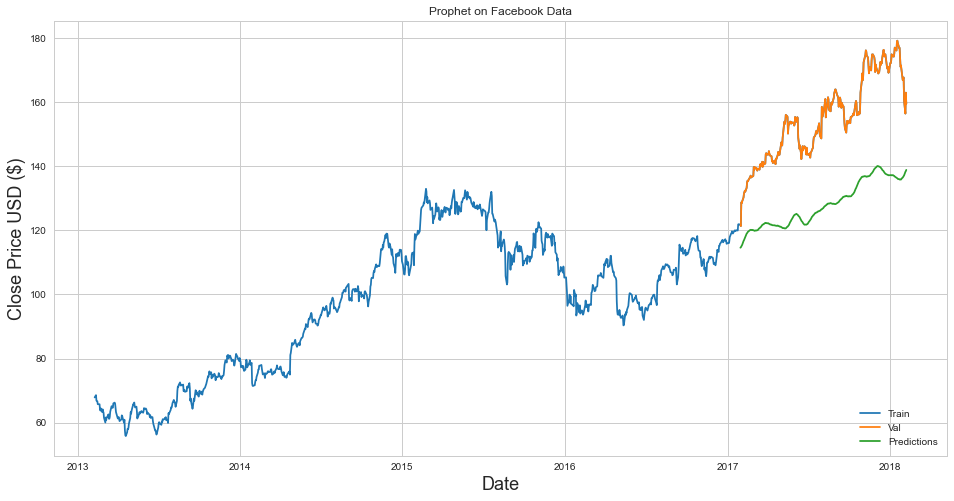
Prophet takes two inputs, the date columns the we renamed as “ds” and the close price which we name “y”.

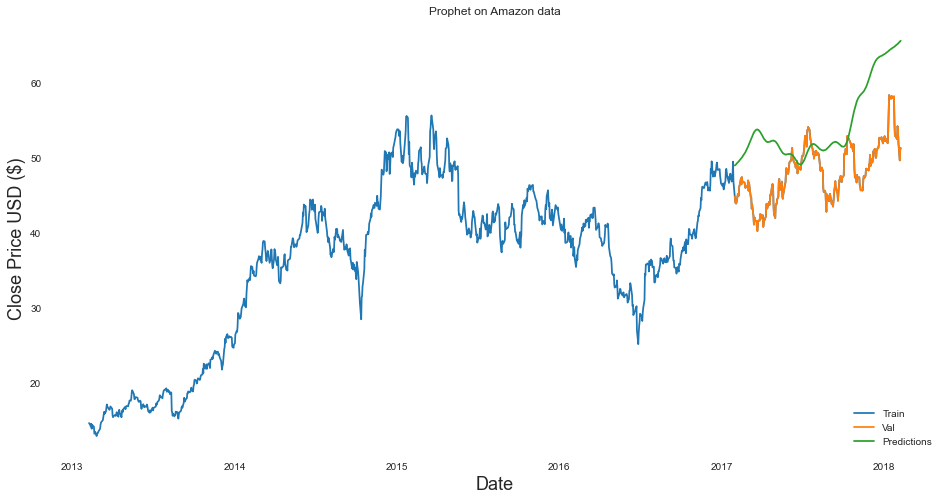
For each stock (apple, google, fb and amazon), we splitted the data into 1000 for training and the rest for test.

The results are as below



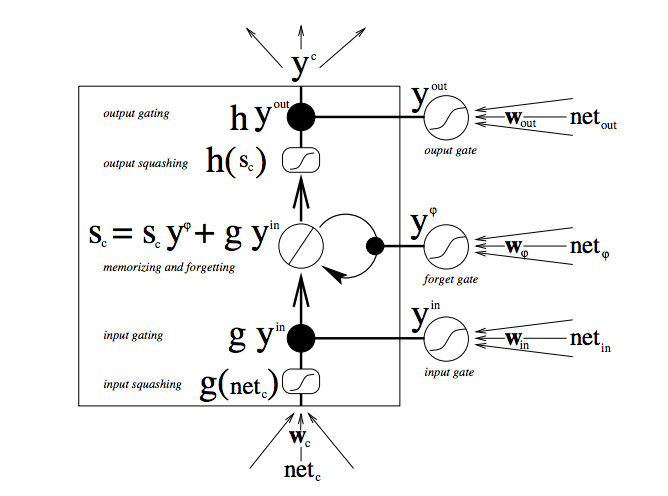




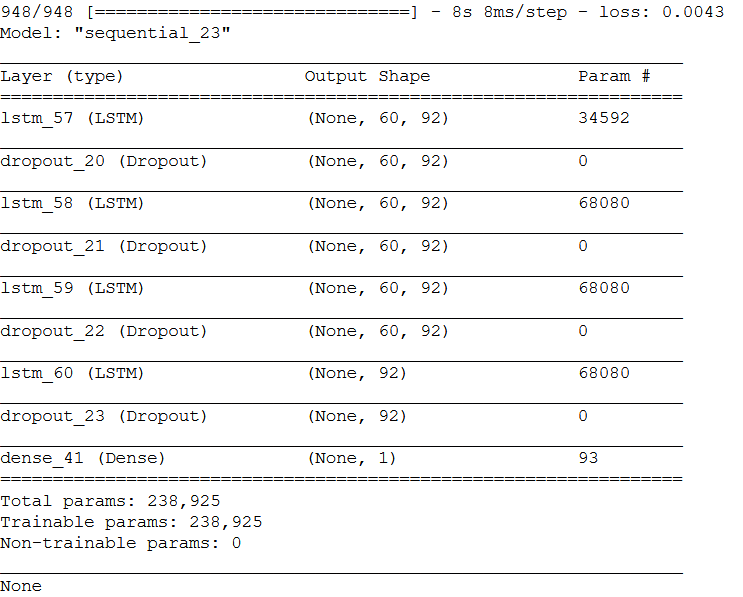


1. **RNN Modeling**

LSTM are the most powerful and well know subset of RNN, they are designed to recognize patterns in sequences of data, such as numerical times series data emanating from sensors, stock markets and government agencies (but also including text, genomes, handwriting and the spoken word). LSTM has 3 gates, the input gate, forget gate and output gate.



The architecture of our network is displayed below.



We trained on the data using different epochs 30, 50 and 100, and observed our metric **mean squared error** under each.

Before fitting the model, LSTM accepts inputs in a format different from that of prophet. The input shape for lstm is (samples, time steps, features)

In this case,

Sample = 1000

Time steps = 60

Features = 1

So, we tried to predict 60-time step into the future.

To do this we need to preprocess our data and make it in the format above.

**Preprocessing**

1. Scaling: Our dataset contains large values ranging from 0 to inf, hence we scale the data into the range of (0,1) such that the lowest value is 0 and the highest value is 1.
2. Reshape: After scaling, we reshape the data into the format above (sample, time-step, features)
3. Splitting: we split the data into train and test in the ratio of 80:20

All these steps are applied on the apple, google, facebook and amazon data

We can then proceed with the modeling. Modeling entails passing the training data through the architecture and predicting on the test data.

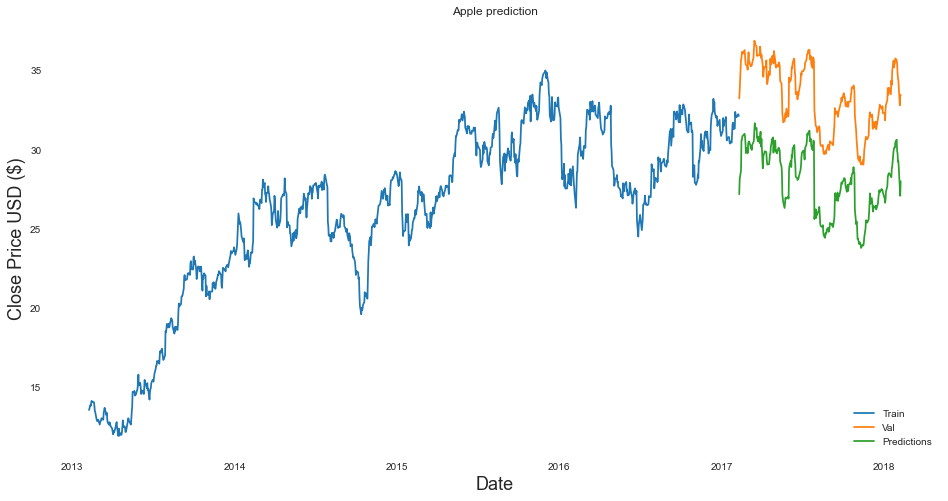
Hyperparameters used in compiling the model are:

Optimizer: **adam optimizer**

Loss: **mean\_squared\_error**

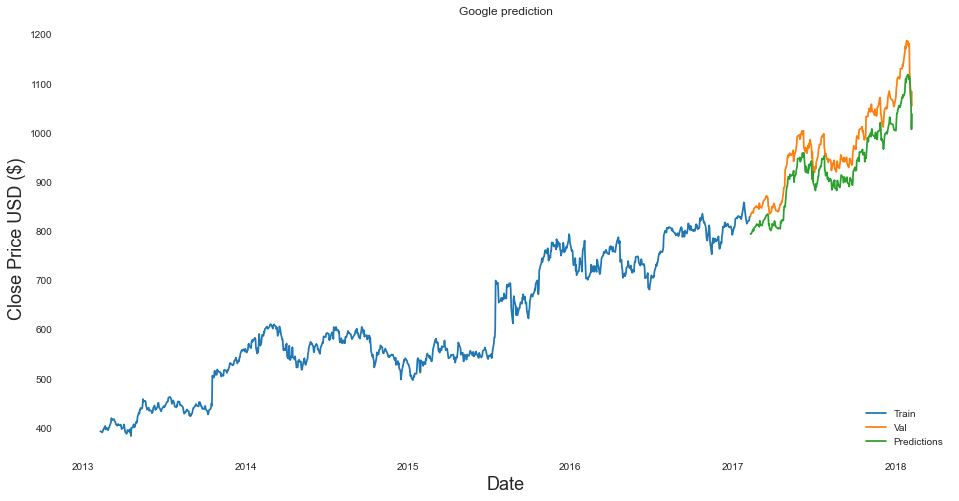
**Result on 30 epochs**

**Apple**

****

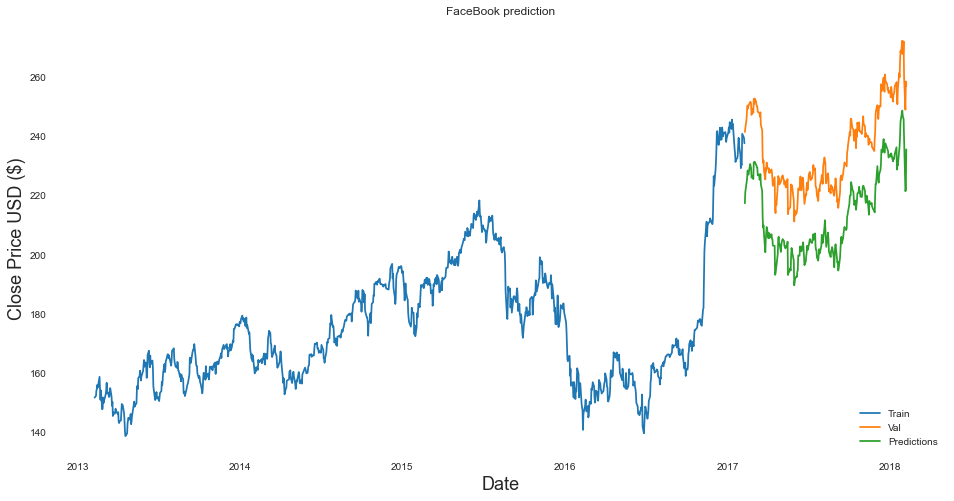
**Rmse: 35.6**

**Google**

****

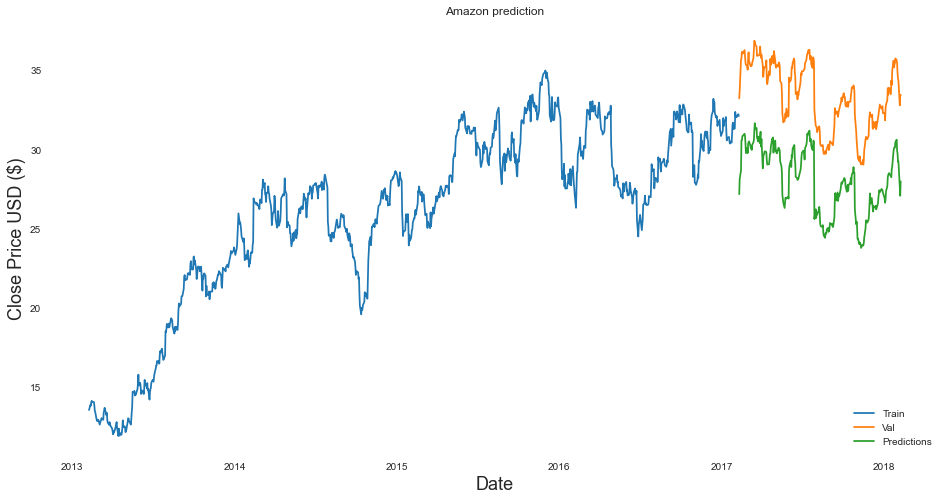
**Rmse: 15.4**

**FaceBook**

****

**Rmse: 25.2**

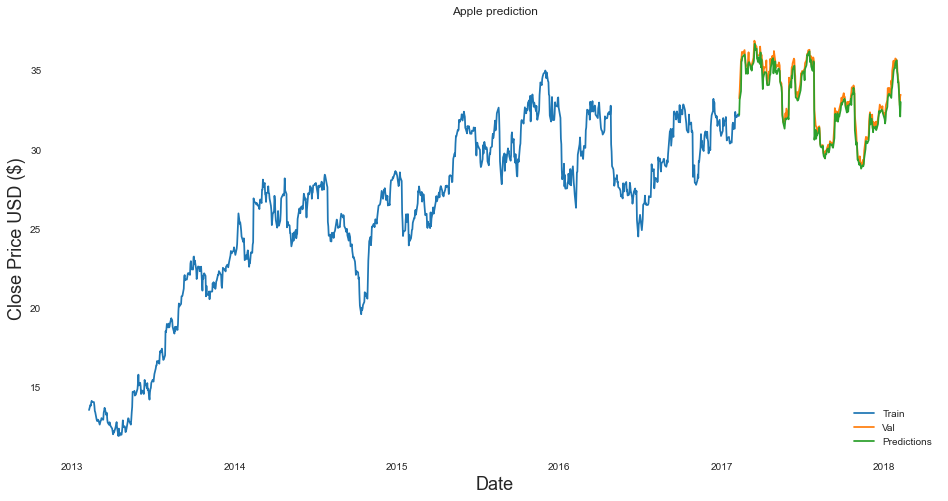
**Amazon**

****

**Rmse: 30.7**

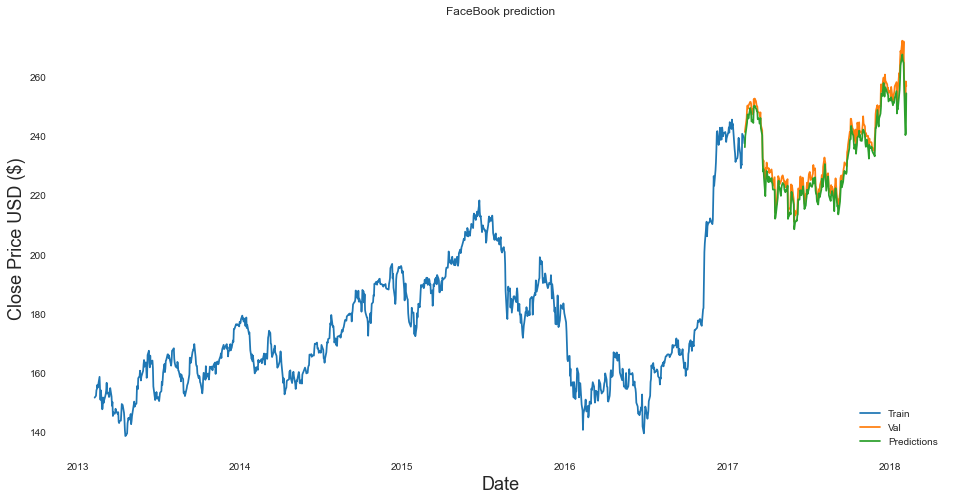
**Result on 100 epochs**

**Apple**

****

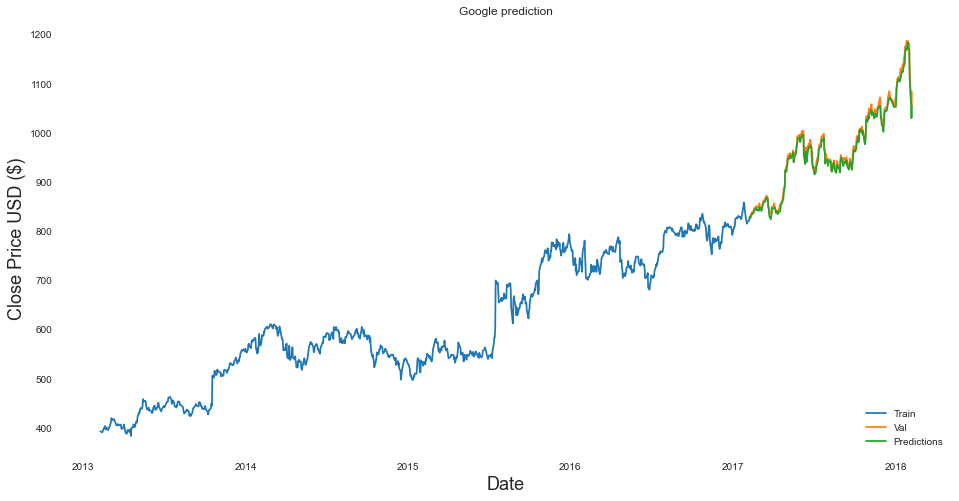
**Rmse: 0.034**

**Facebook**

****

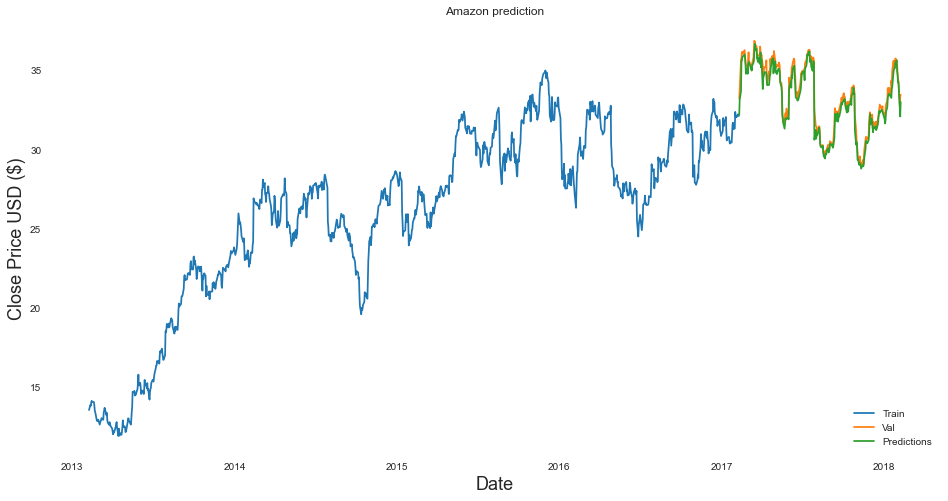
**Rmse: 0.031**

**Google**

****

**Rmse: 0.023**

**Amazon**

****

**Rmse: 0.012**