VIETNAM GENERAL CONFEDERATION OF LABOUR

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**MANAGEMENT OF INFORMATION SYSTEMS**

**Final**

**Machine Learning**

*Instructor*: **MR. LÊ ANH CƯỜNG**

*Performer*: **BÙI ĐỨC DŨNG - 518H0611**

**ĐỖ MAI HƯƠNG - 518H0506**

**LÊ TUẤN SANG -519H0039**

Class: **18H50301**

School year: **22**

**HO CHI MINH CITY, 2022**

VIETNAM GENERAL CONFEDERATION OF LABOUR

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**MANAGEMENT OF INFORMATION SYSTEMS**

**Final**

**Machine Learning**

*Instructor*: **MR. LÊ ANH CƯỜNG**

*Performer*: **BÙI ĐỨC DŨNG - 518H0611**

**ĐỖ MAI HƯƠNG - 518H0506**

**LÊ TUẤN SANG -519H0039**

Class: **18H50301**

School year: **22**

**HO CHI MINH CITY, 2022**

**GRATITUDE**

Sincerely thank you for all the things you taught us and all your helps. We are very appreciate your dedication. We have learned a lot from you and received many knowledge from you. You are always willing to clear up all our queries. We really respect you. Hope to study with you in another course in future.

**PROJECT COMPLETED AT TON DUC THONG UNIVERSITY**

I hereby declare that this is my own project and is under the guidance of MR.LÊ ANH CƯỜNG. The research contents and results in this topic are honest and have not been published in any form before. The data in the tables for analysis, comments and evaluation are collected by the author himself from different sources, clearly stated in the reference section.

In addition, the project also uses a number of comments, assessments as well as data of other authors, other agencies and organizations, with citations and source annotations.

**If I detect any fraud, I will take full responsibility for the content of my project.** Ton Duc Thang University is not related to copyright and copyright violations caused by me during the implementation process (if any).

*Ho Chi Minh City, october 16th 2022*

*Author*

*(signed)*

*Dung*

*Bùi Đức Dũng*

*Huong*

*Đỗ Mai Hươn*

*Sang*

*Lê Tuấn Sang*

**The evaluation part of the teacher who marks the test**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ho Chi Minh City, month day year

(sign and write full name)

# TABLE OF CONTENT

[**TABLE OF CONTENT**](#_heading=h.1fob9te) **1**

[**Problem**](#_heading=h.5yvx4nrvo5x5) **2**

[**Dataset**](#_heading=h.915hu1ii6ocz) **2**

[**Preprocess**](#_heading=h.62ojlwy3fh0t) **2**

[**Correlation**](#_heading=h.x9kgmi7d5at4) **15**

[**Normalizing the dataset**](#_heading=h.1elko2rvyfw5) **16**

[**Model and validate data**](#_heading=h.su721erow3r9) **20**

[**RNN**](#_heading=h.x9dv9iqbndnp) **20**

**CNN 37**

**LSTM 49**

# Problem

## Dataset

We use the stock dataset of Amazon, Facebook and Netflix from 2015-2019 from kaggle. We will compare the stock price of these three companies and predict the stock price of each company.

Each dataset will have seven columns. Each column contain the following features:

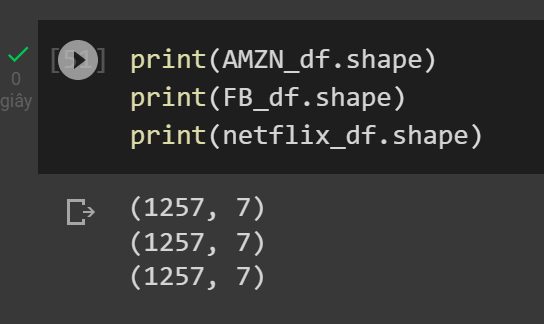
1. Date: Trading day.
2. Open: "open" means the "starting price" of the day.
3. High: "high" refers to the most expensive price.
4. Low: "low" refers to the "cheapest price".
5. Close: "close" means the "ending price" of the day.
6. Adj Close: Adjusted close is the closing price after adjustments for all applicable splits and dividend distributions.
7. Volume: Volume is the number of shares traded.

## Preprocess

First we will import the dataset of three companies.



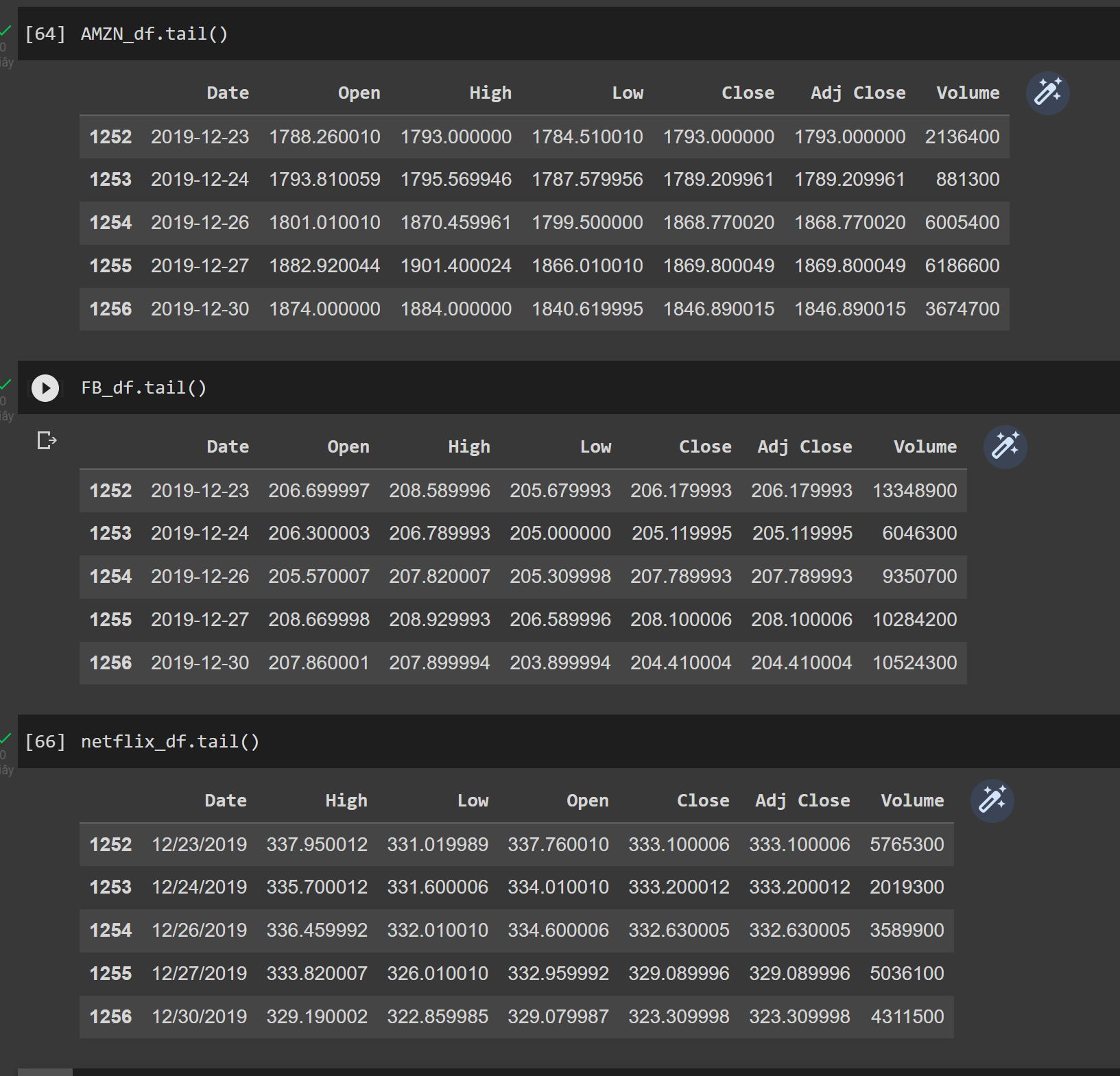
Next, we check the dimensions of three companies. Look at the data, each dataset has seven features columns as we mentioned before and 1257 rows of data.



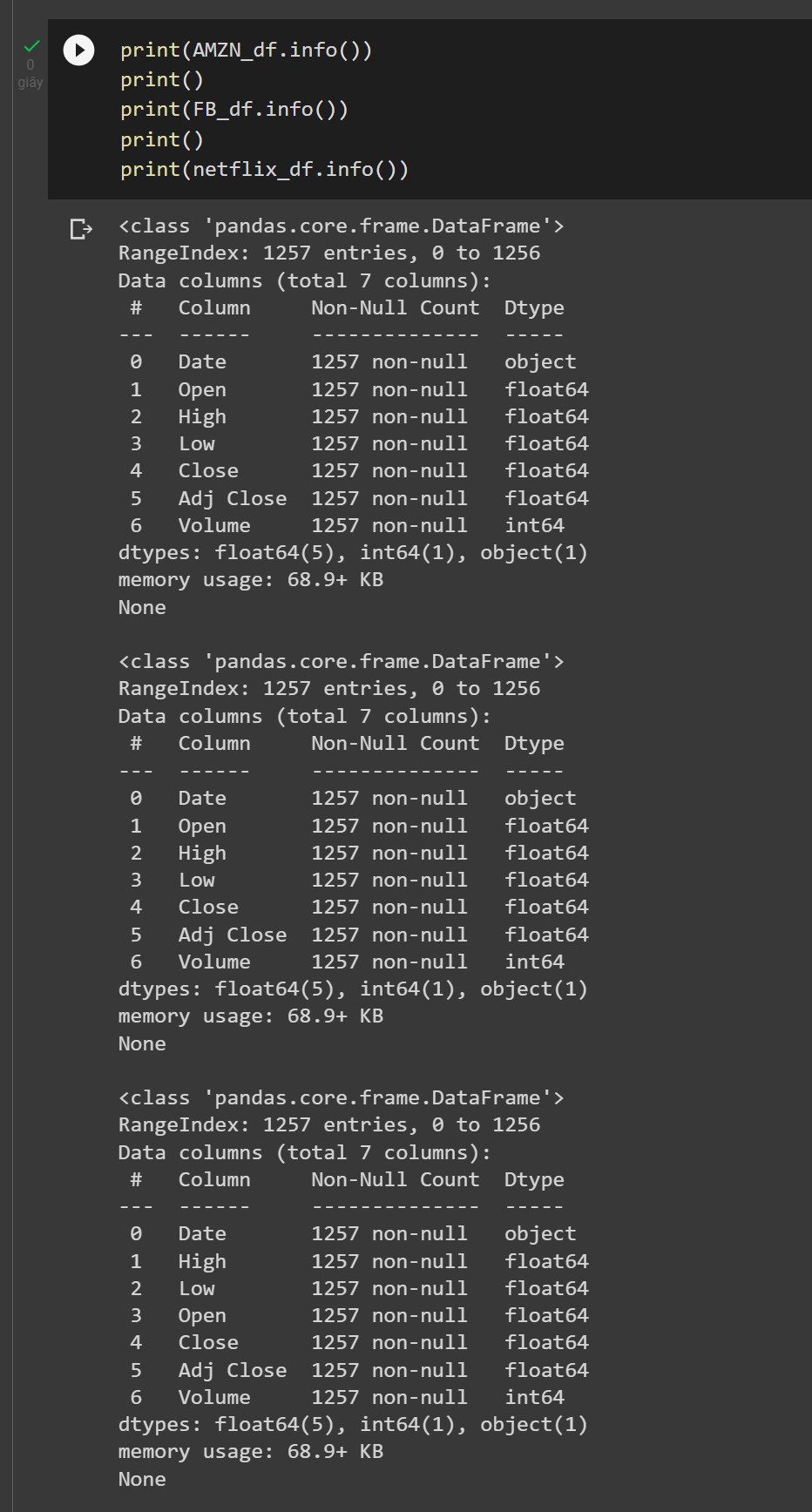
Each dataset trading date starts at 2/1/2015.



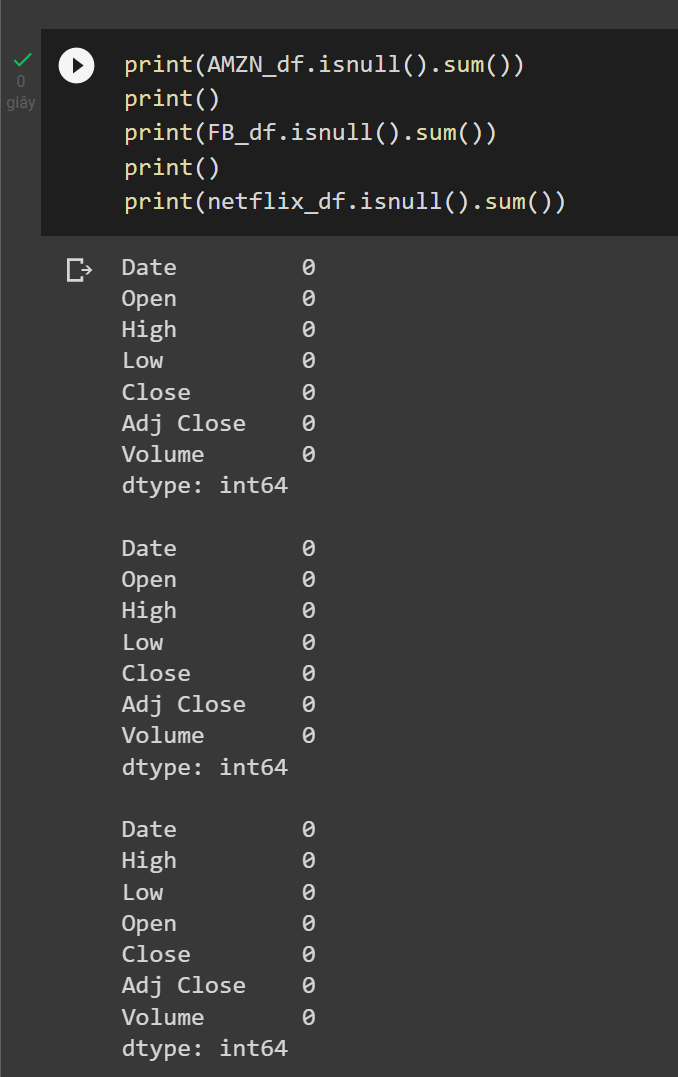
Each dataset ends on 30/12/2019.



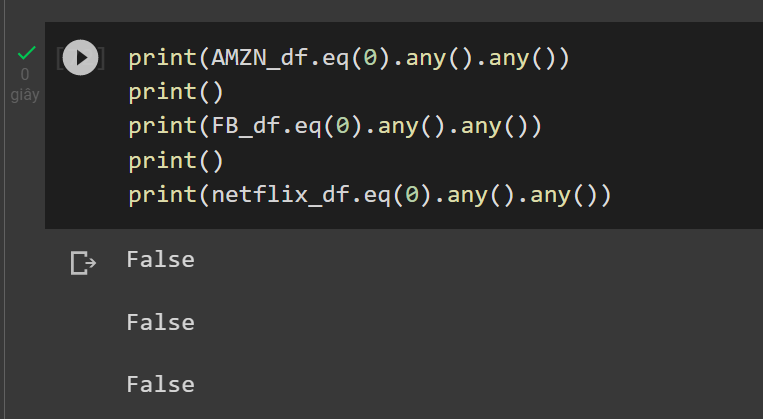
Next we check the missing value.In the picture below, all columns have 1257 rows, so no missing value.



Next we check for NaN.

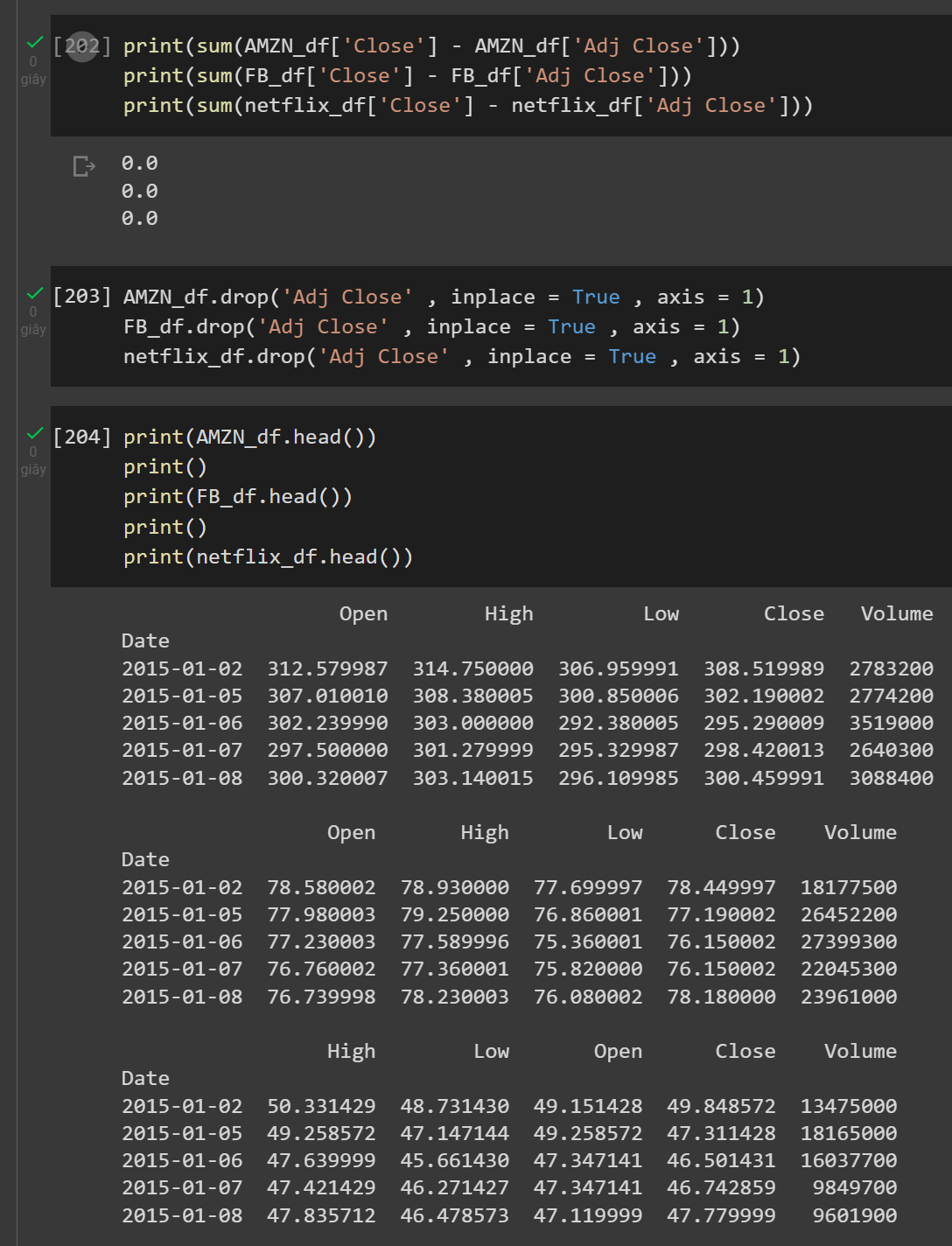


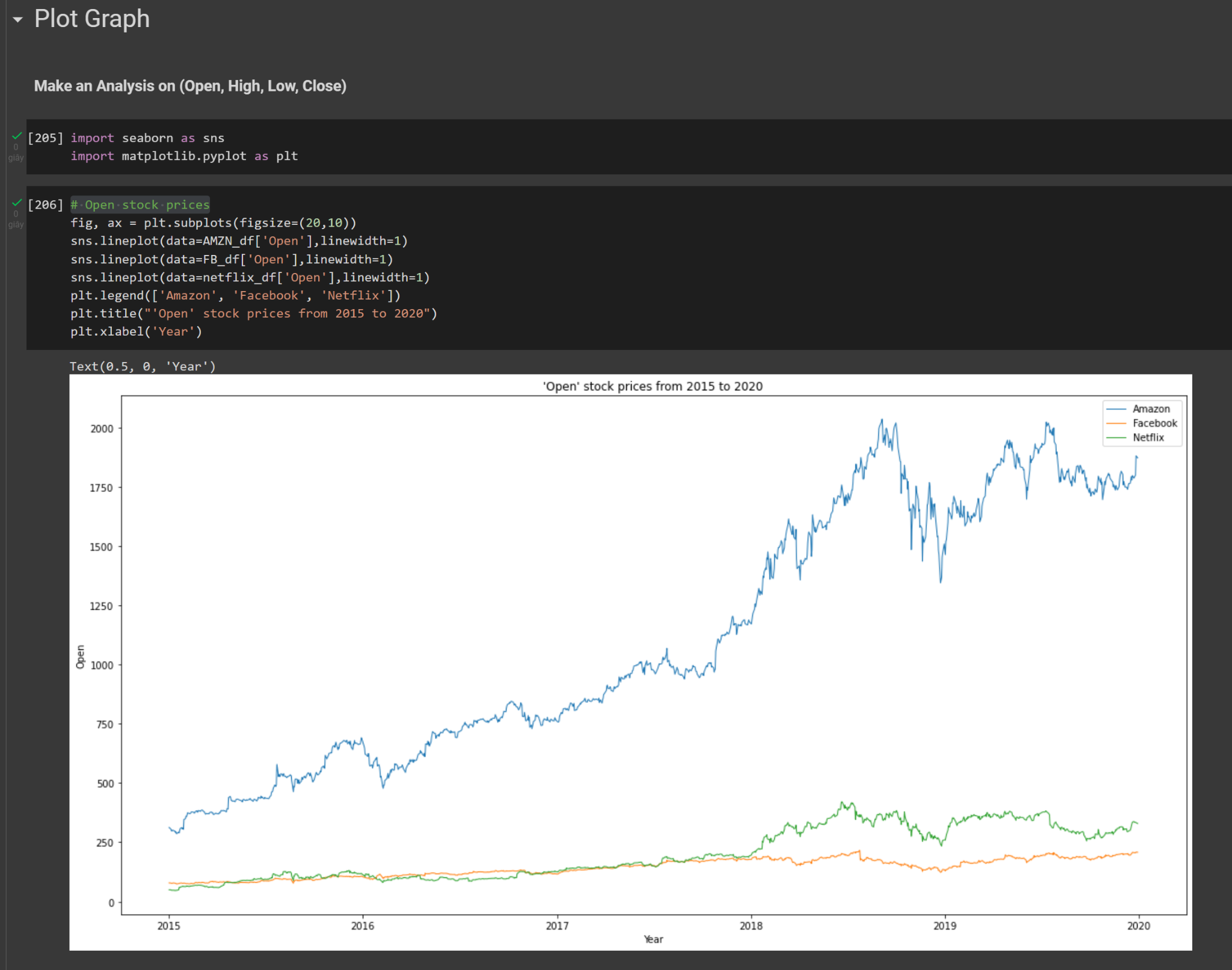
Check if at least one element is 0.



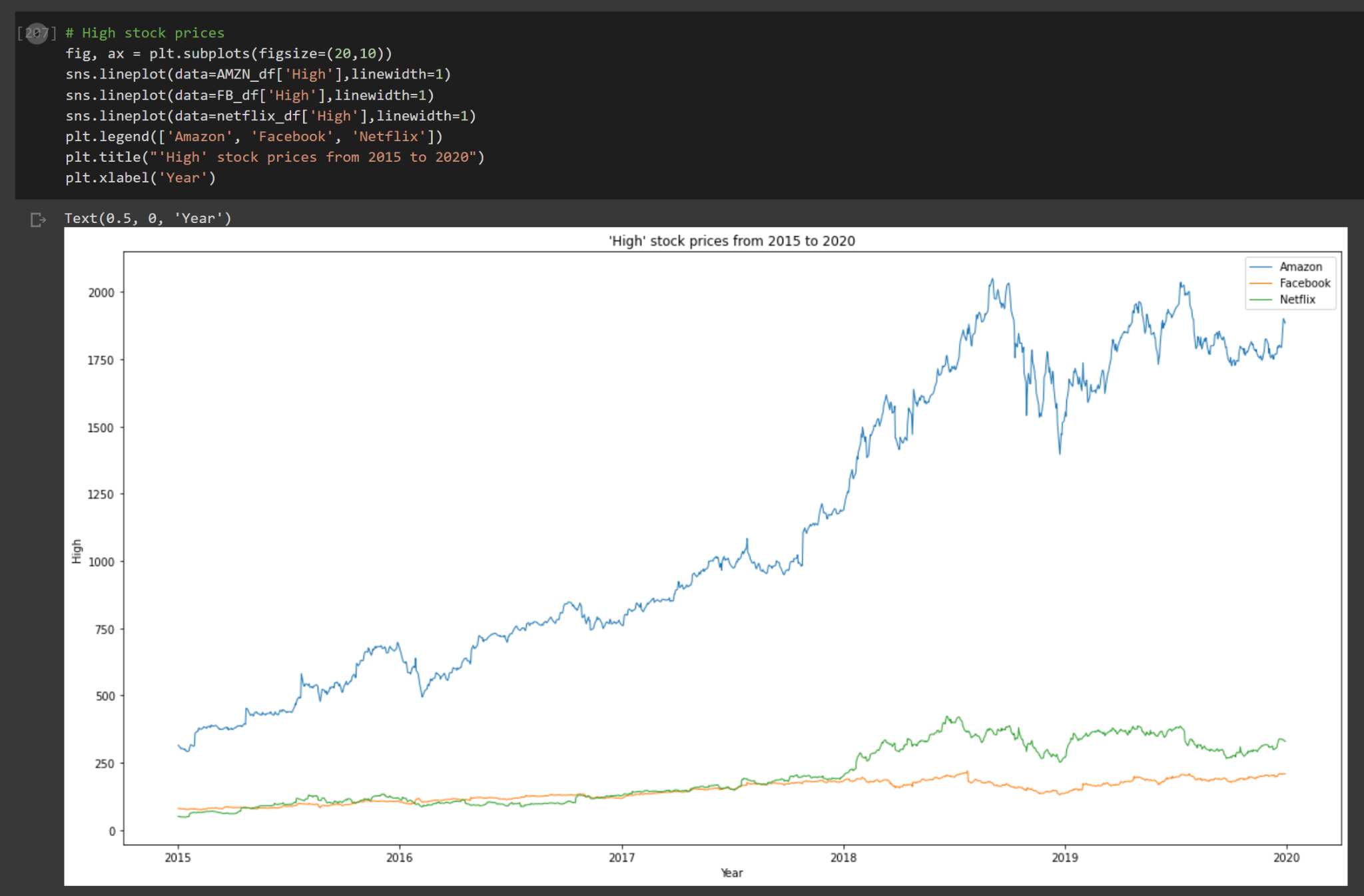
In general, the **adjusted closing** price is considered to be a **more technically** accurate reflection of the true value of the stock. [Adjusted Closing Price vs. Closing Price](http://finance.zacks.com/adjusted-closing-price-vs-closing-price-9991.html)

But here the two columns are equal, so we drop the Adj Close column.

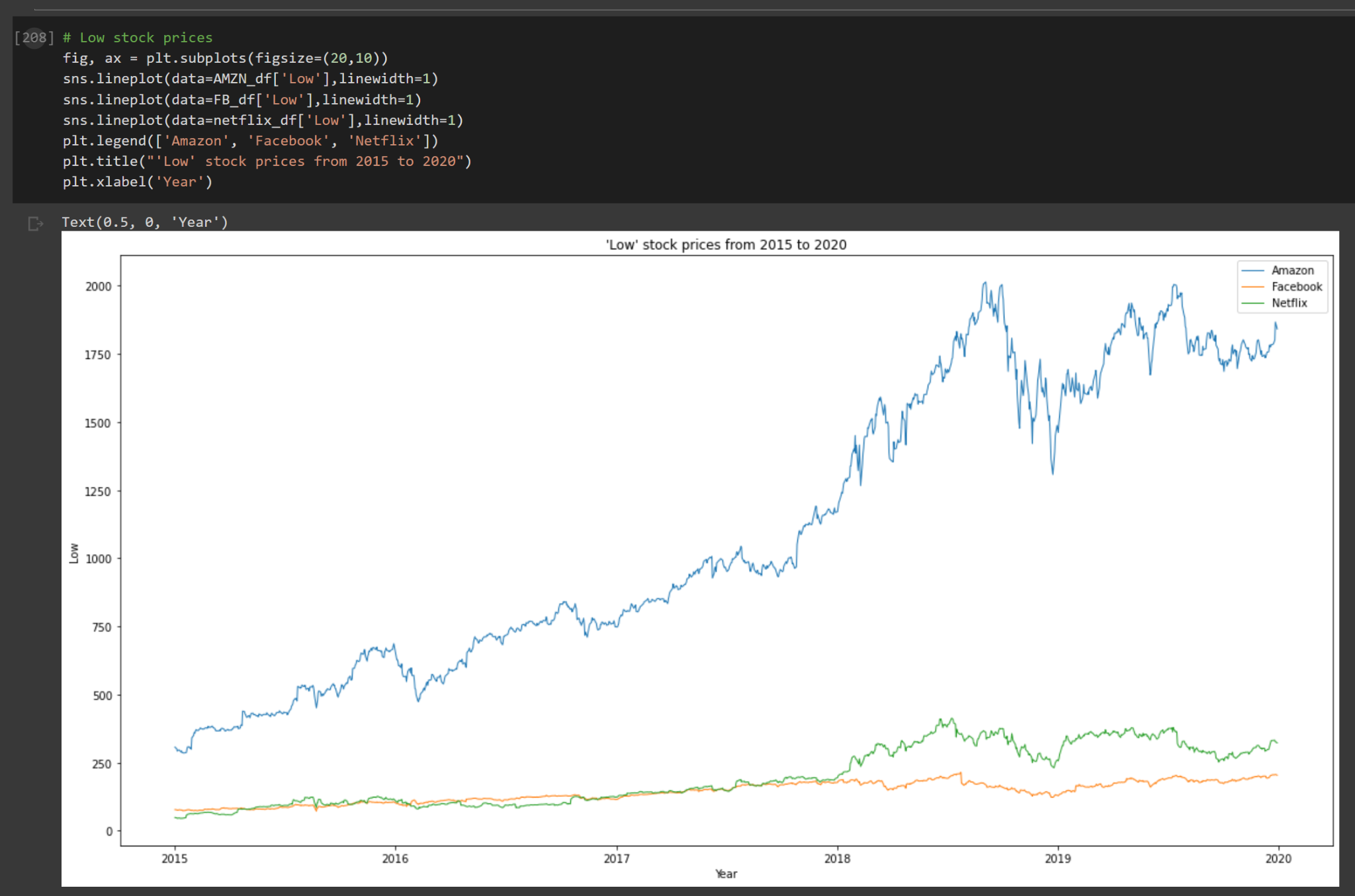


For an overview, we will draw graphs for each feature column. First graph is the ‘Open’ stock prices from 2015 to early 2020.

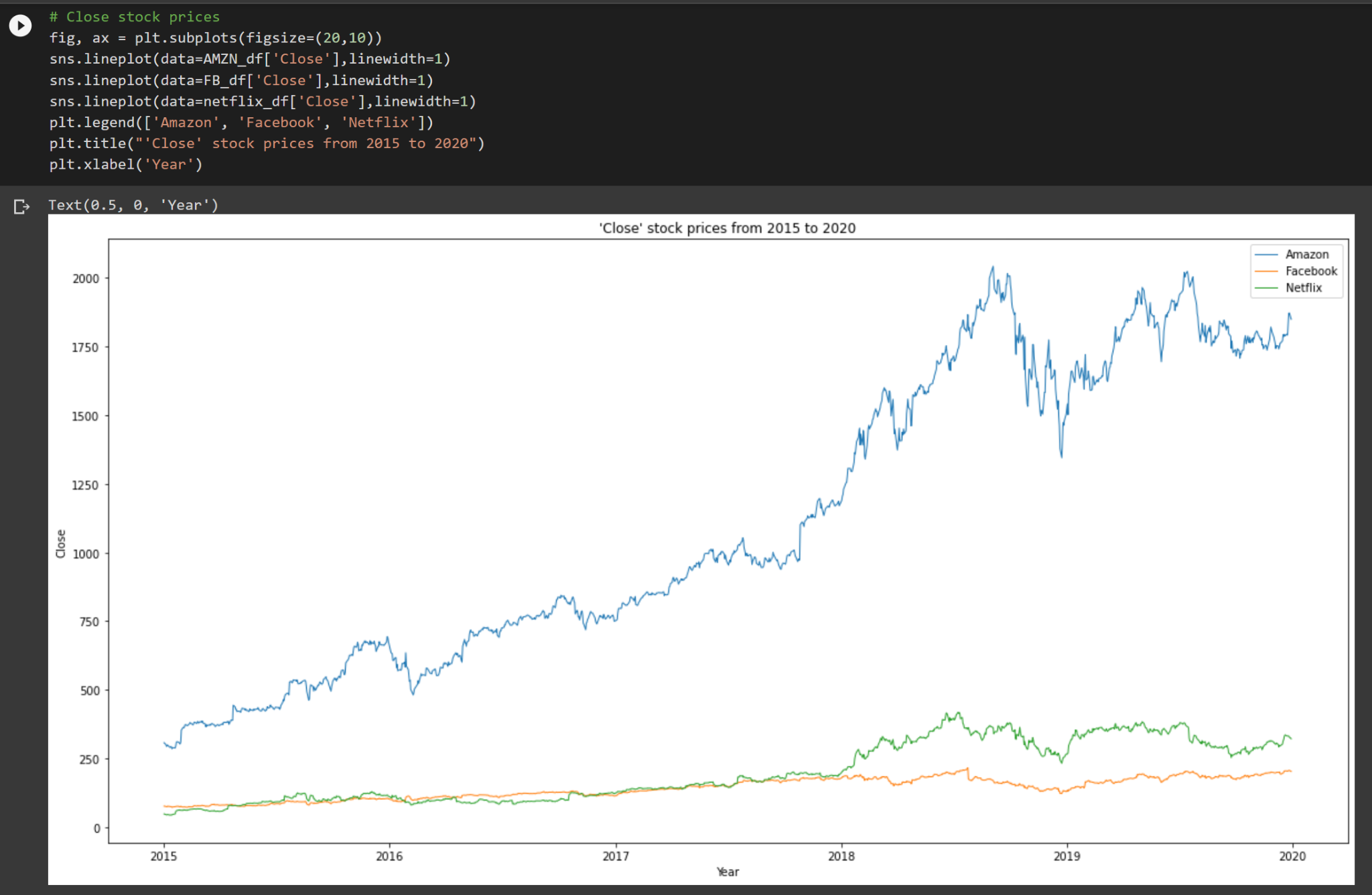
Second graph is the ‘High’ stock prices from 2015 to early 2020.



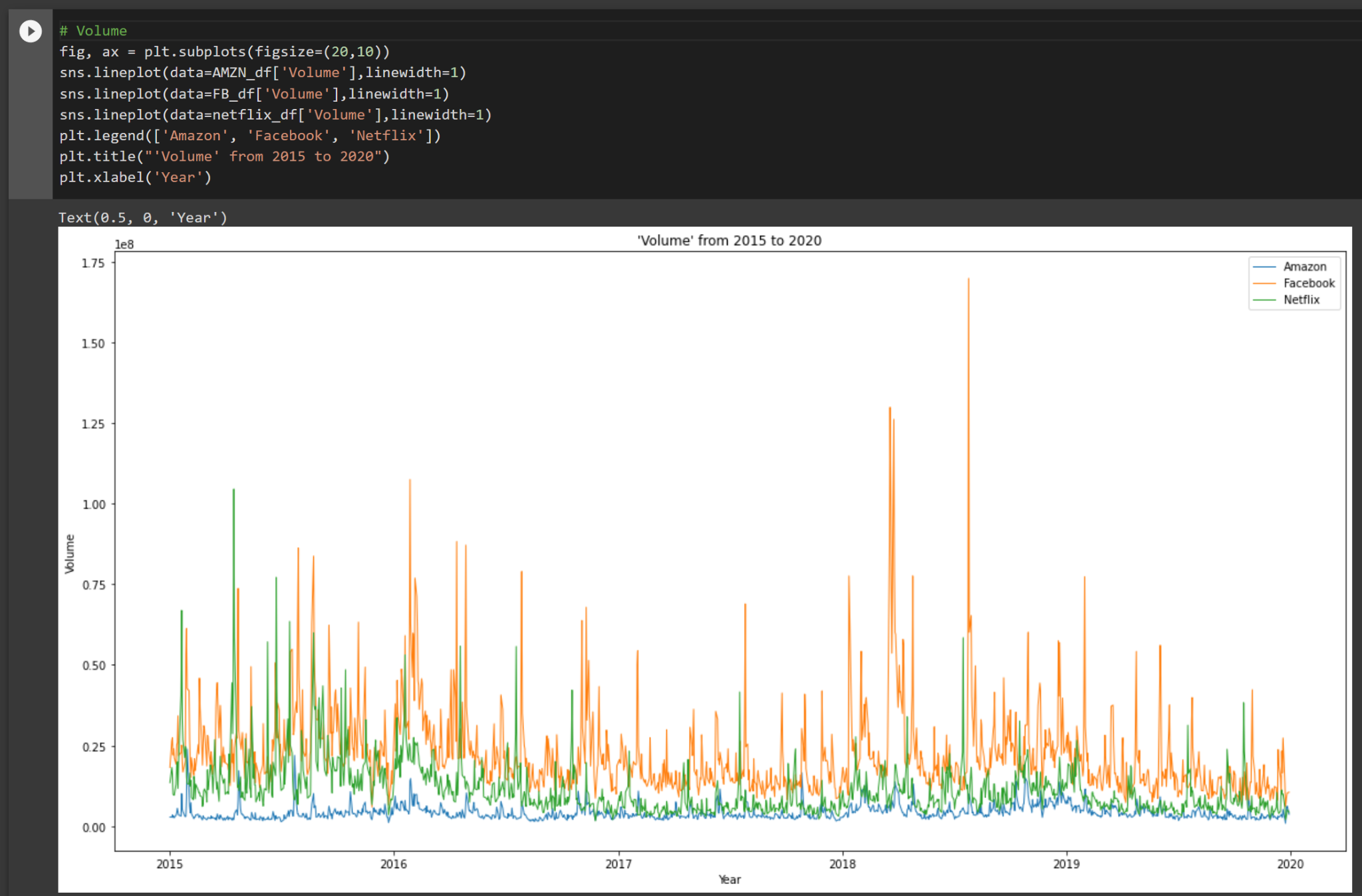
Third graph is the ‘Low’ stock prices from 2015 to early 2020.



Fourth is the ‘Close’ stock prices from 2015 to early 2020.

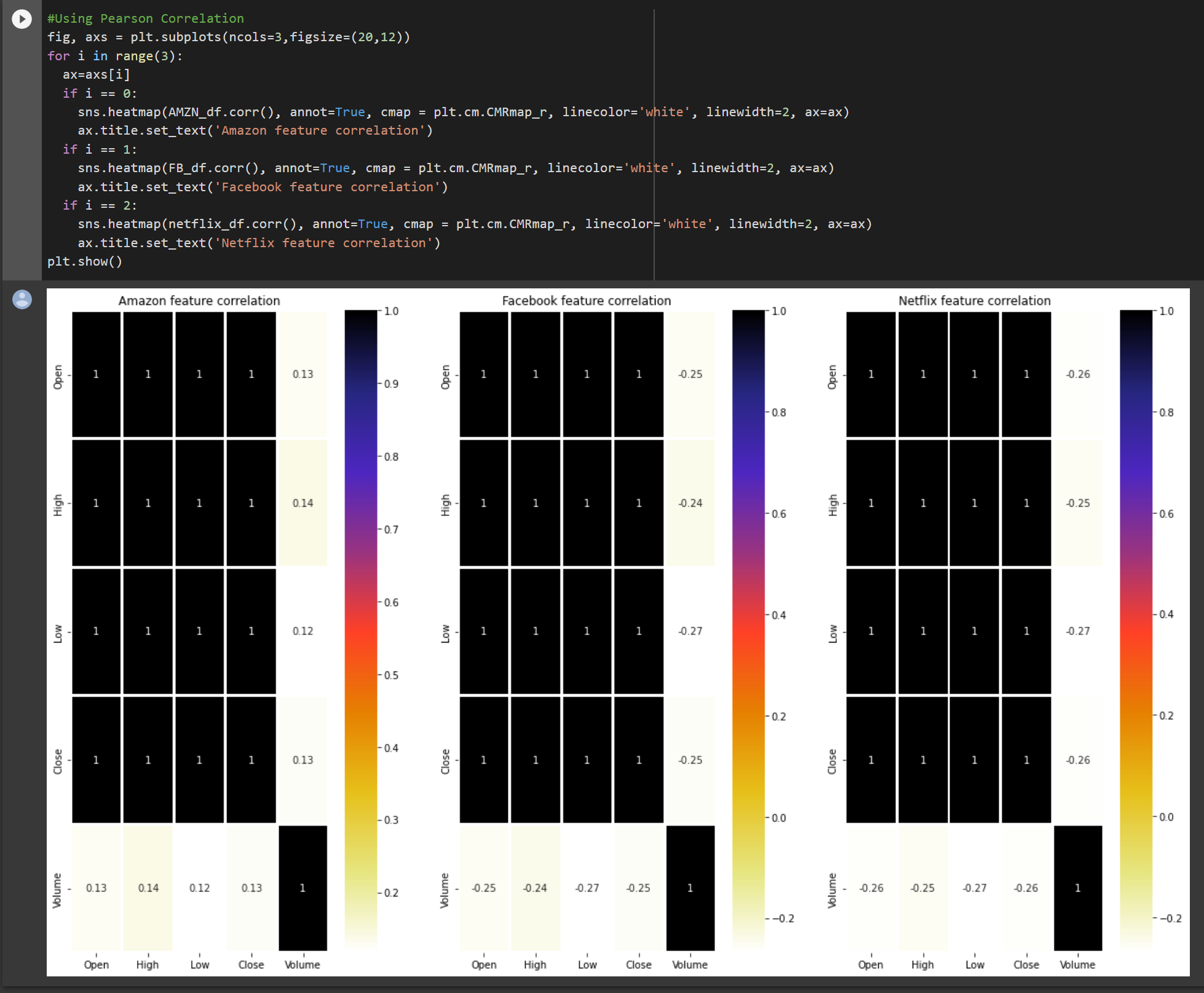


And the last graph is the volume



### Correlation

Then we check the correlation between all the column features.



**Feature Open, high, low are highly correlated to Target feature Close. we can use either one of the features for prediction to avoid multicollinearity.**

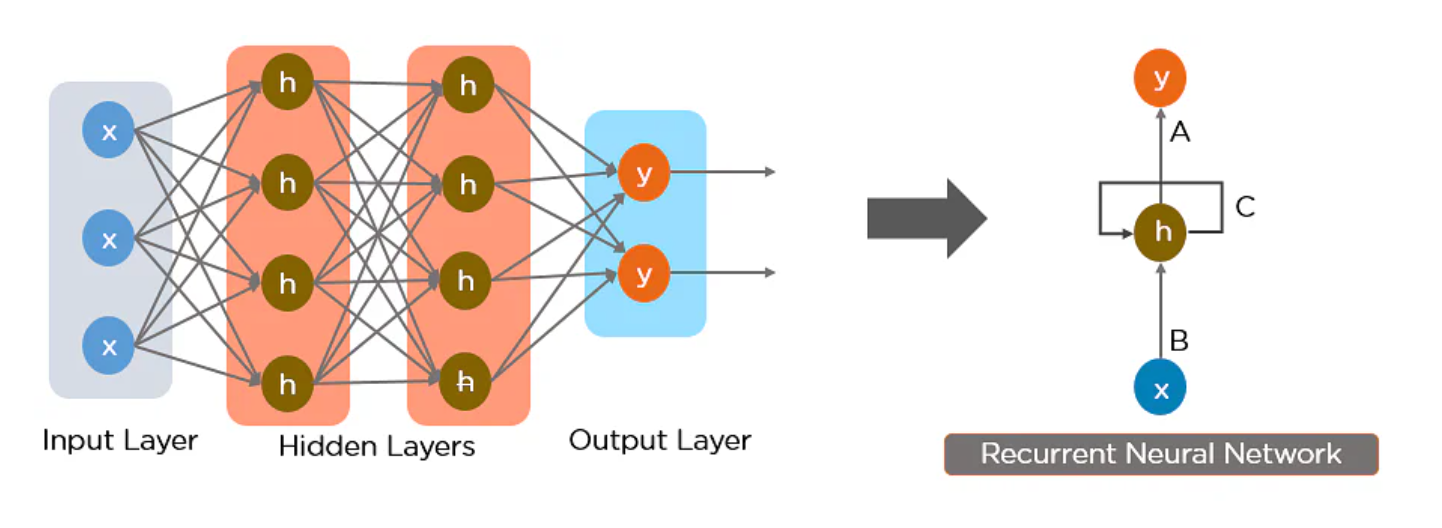
## Normalizing the dataset

Because feature Open, high, low are highly correlated to feature Close. So we will use the Close column to predict the stock price. In the real world, we also do not know the number of shares traded in a day so we also do not need that column.

The first model we will use is Rnn(Recursive neural network).

**Recurrent Neural Network(RNN)** is a type of Neural Network where the **output from the previous step is fed as input to the current step**. In traditional neural networks, all the inputs and outputs are independent of each other, but in cases like when it is required to predict the next word of a sentence, the previous words are required and hence there is a need to remember the previous words. Thus RNN came into existence, which solved this issue with the help of a Hidden Layer. The main and most important feature of RNN is the Hidden **state**, which remembers some information about a sequence.

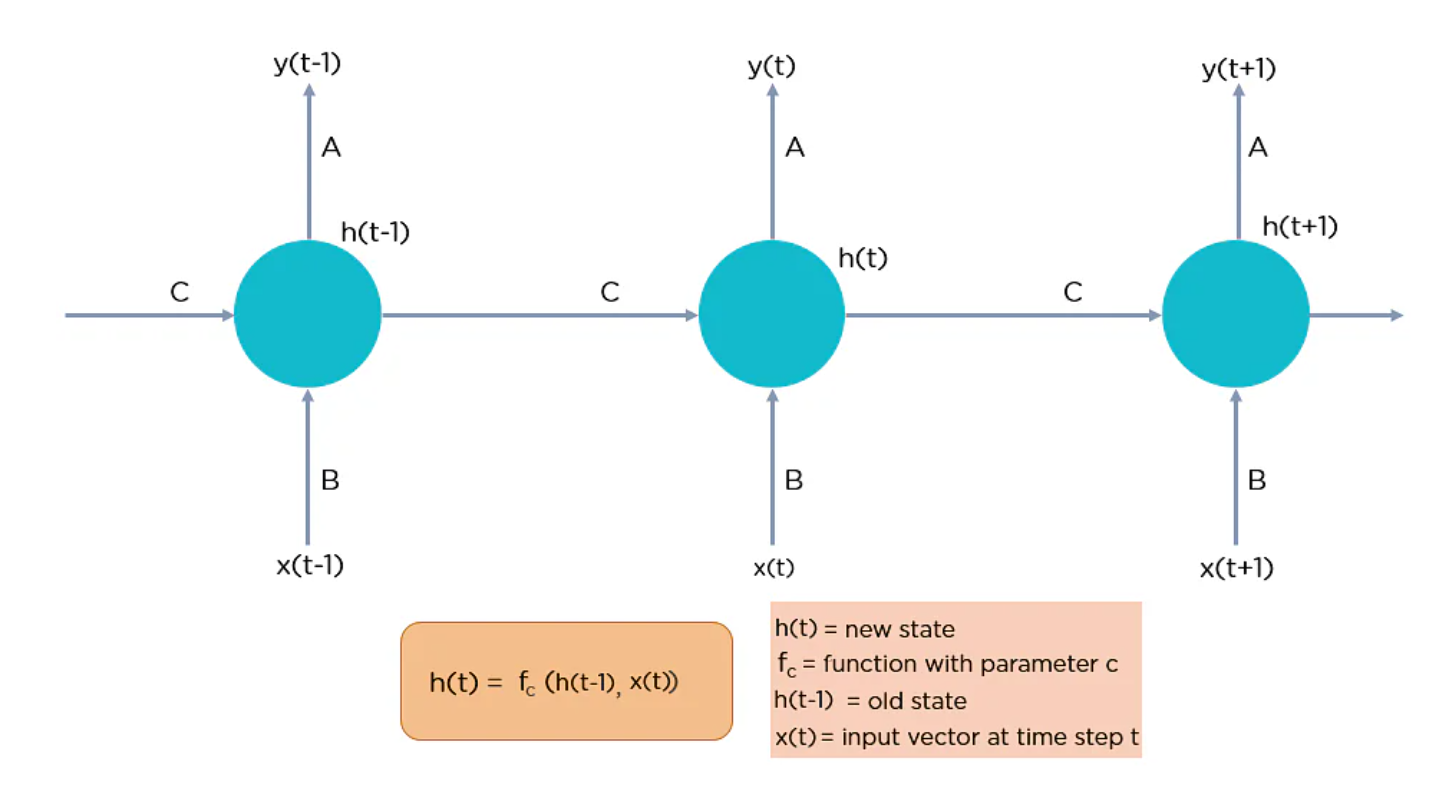
Below is how you can convert a Feed-Forward Neural Network into a Recurrent Neural Network:



RNN has a **“memory”** which remembers all information about what has been calculated. It uses the same parameters for each input as it performs the same task on all the inputs or hidden layers to produce the output. This reduces the complexity of parameters, unlike other neural networks.

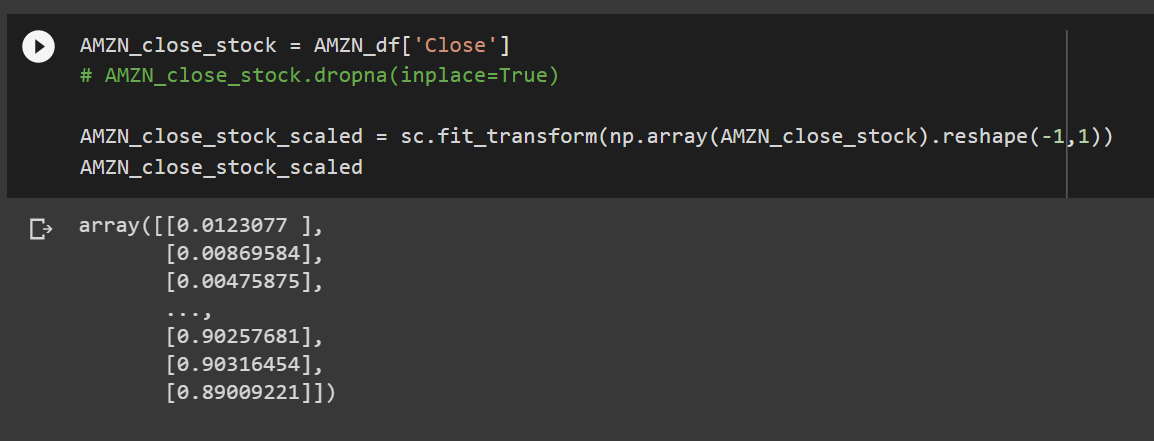
The nodes in different layers of the neural network are compressed to form a single layer of recurrent neural networks. A, B, and C are the parameters of the network.

Here, “x” is the input layer, “h” is the hidden layer, and “y” is the output layer. A, B, and C are the network parameters used to improve the output of the model. At any given time t, the current input is a combination of input at x(t) and x(t-1). The output at any given time is fetched back to the network to improve on the output.

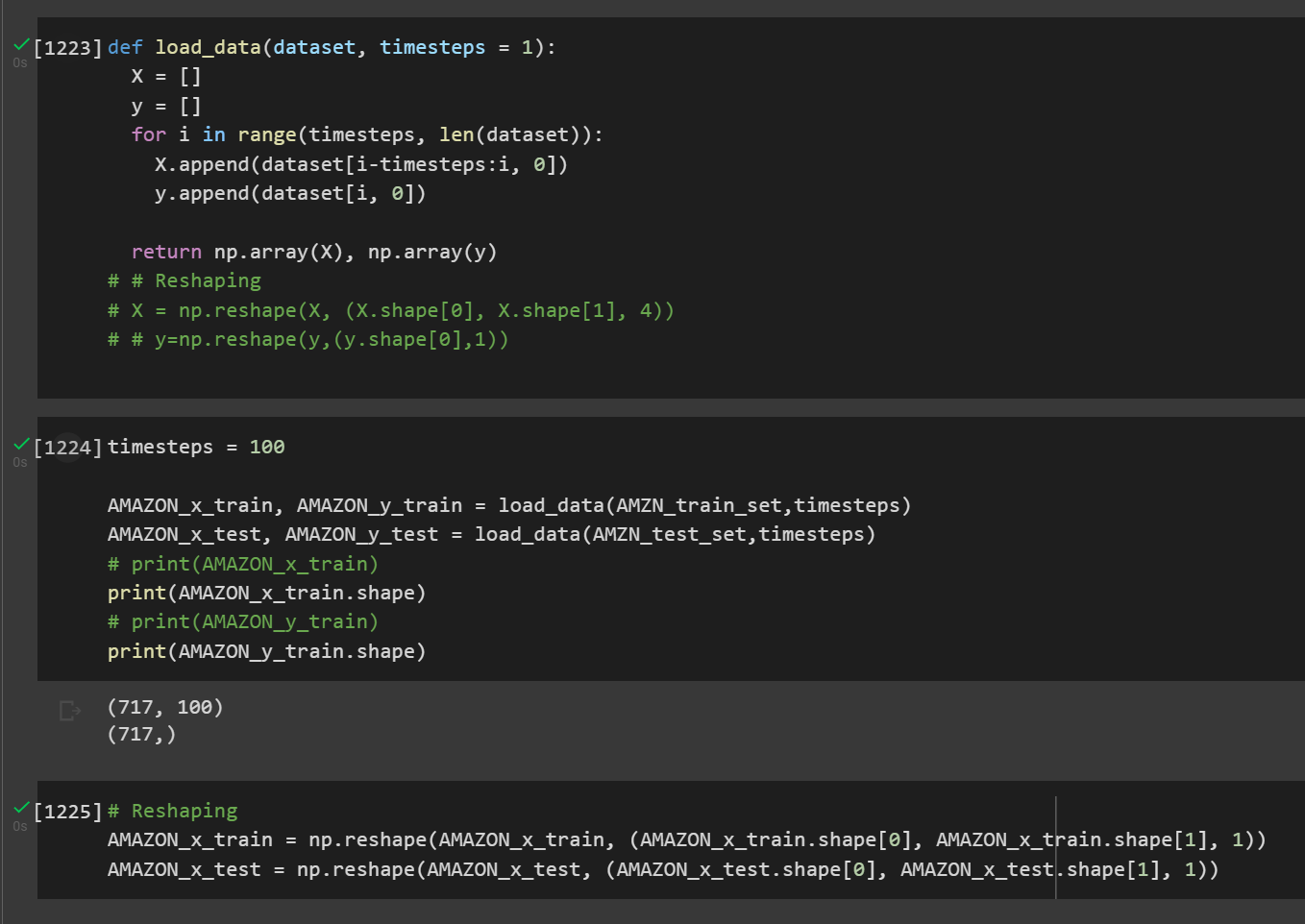


Now that you understand what a recurrent neural network is let’s look at the different types of recurrent neural networks.

Back to our project first we chose the ‘Close’ column then we need to scale it values using ***MinMaxScaler.***

******

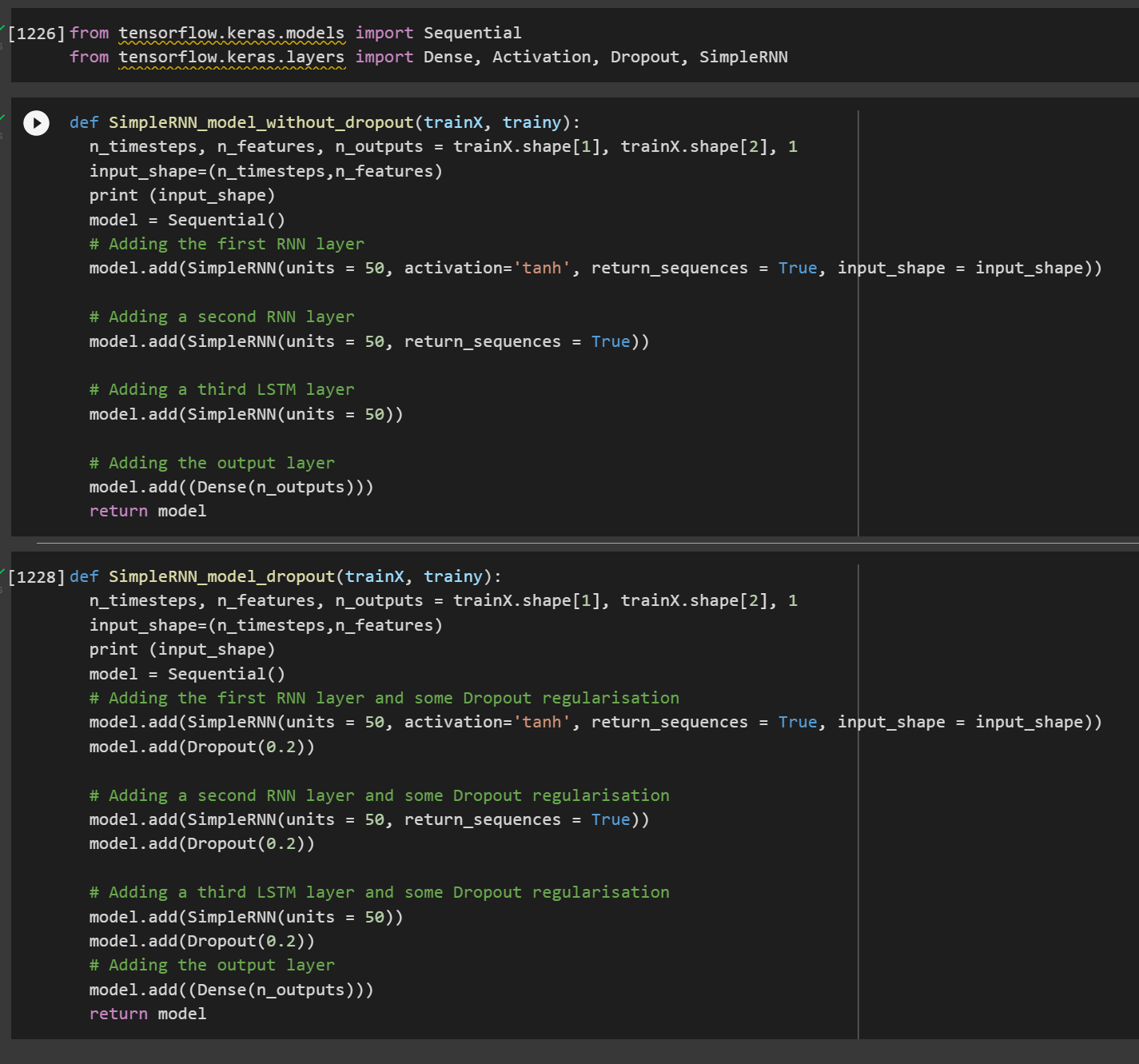
Then we create train and test data. We use the timesteps of 100 days to predict the one day after day 100. Then we reshape the data



## Model and validate data

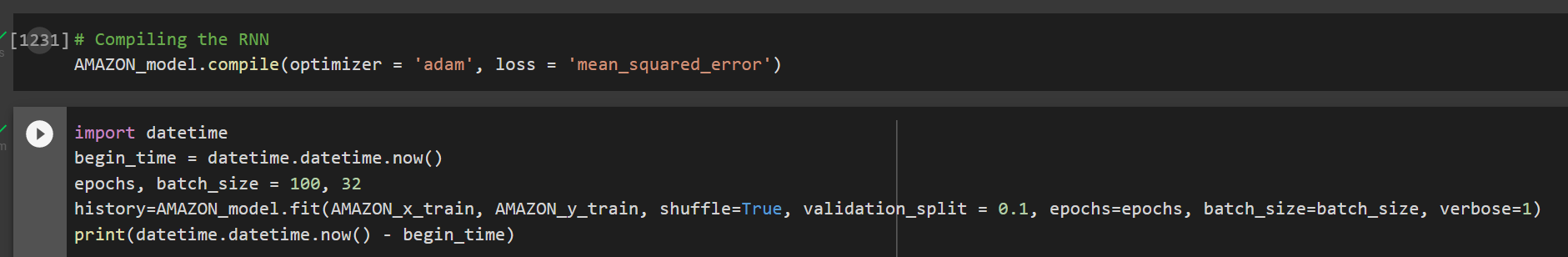
### RNN

Next step is, Model and validate data. We create two functions, first is using SimpleRNN but without Regularization method**.** The second function is using SimpleRNN with the Dropout Regularization method



We define the model and look into its summary.

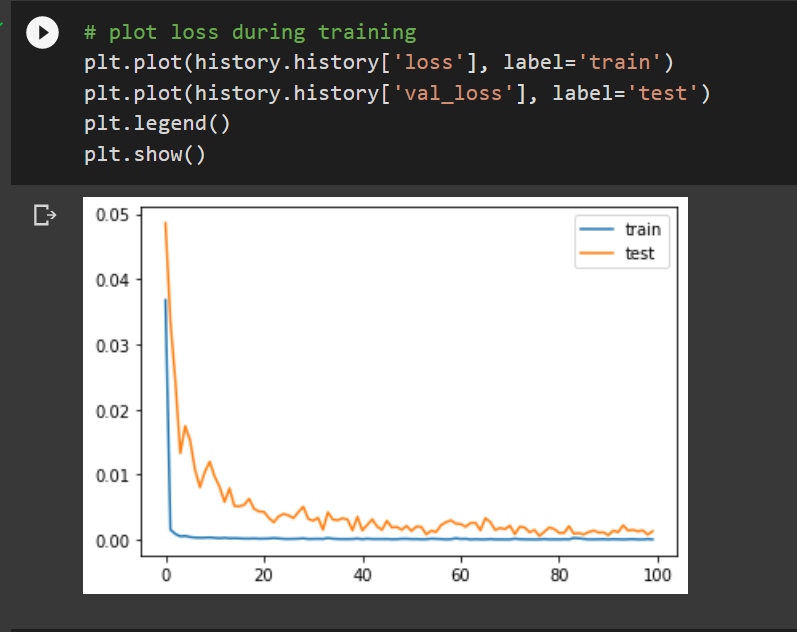
Next we compile the model. In model fit() we use validation split about 10%



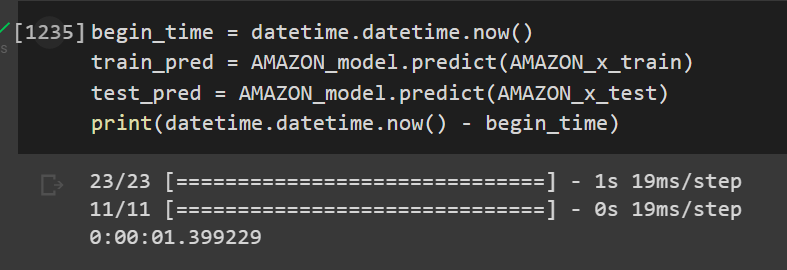
This model trains for about 2 minutes 30 seconds.



This is the Loss during training

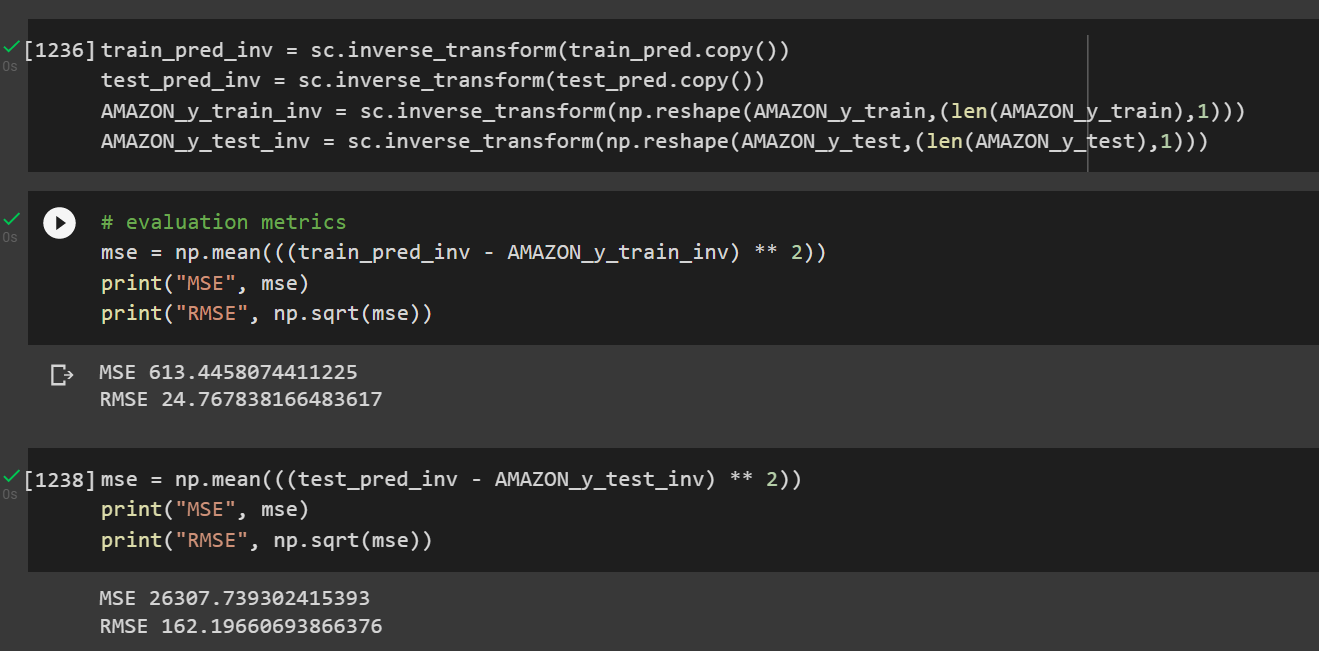


The time of testing is about 1 to 2 seconds.

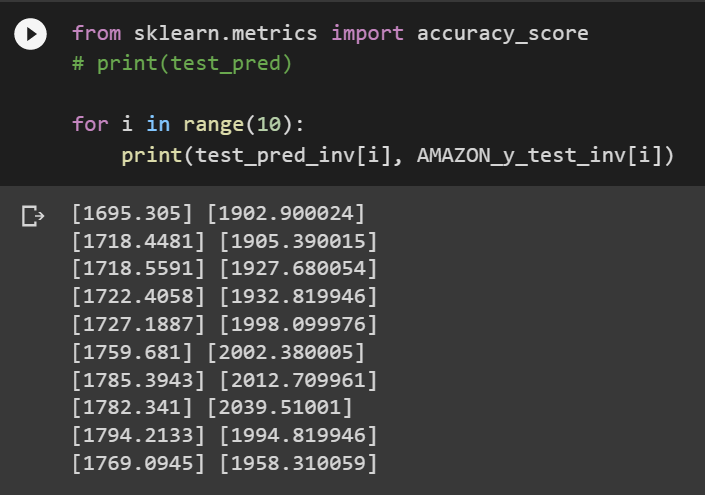


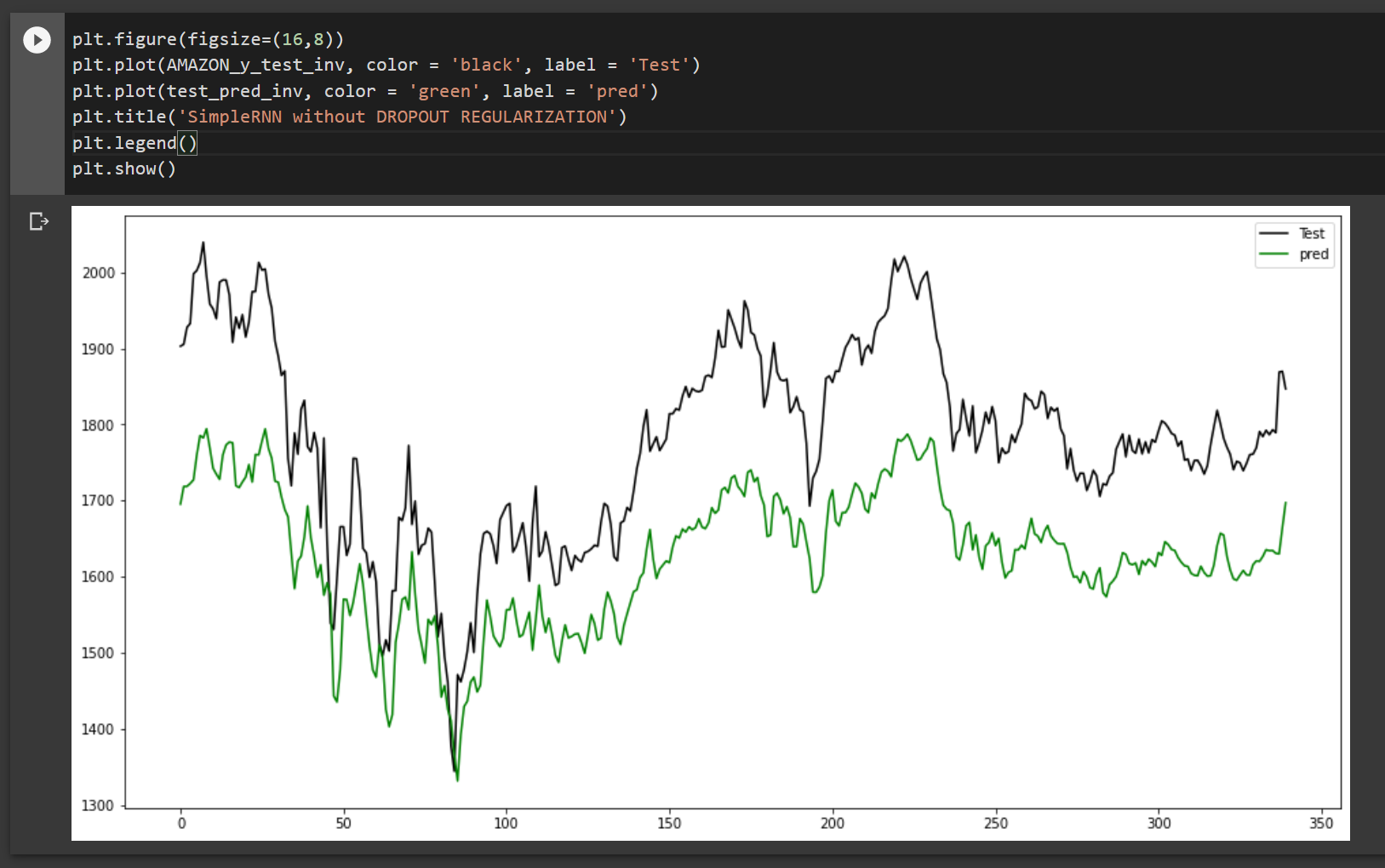
This model we have the MSE is 613.46 and the RMSE is 24.76 for the real train data with prediction train data.

For the real test data with prediction test data MSE is 26307.73 and the RMSE is 162.19



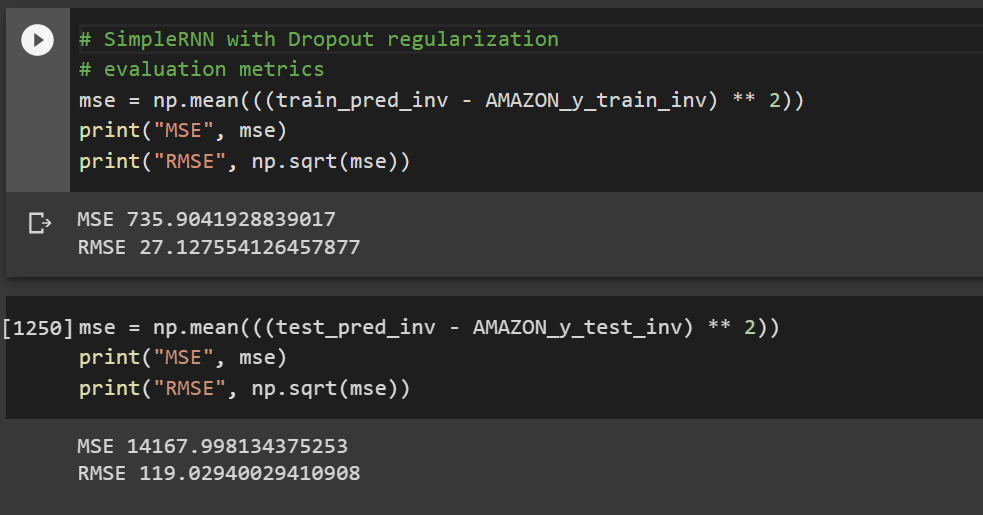
The result is quite good.



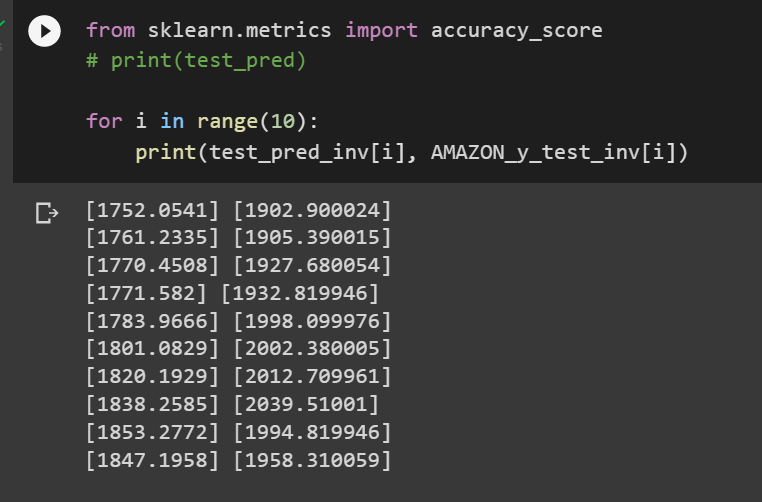


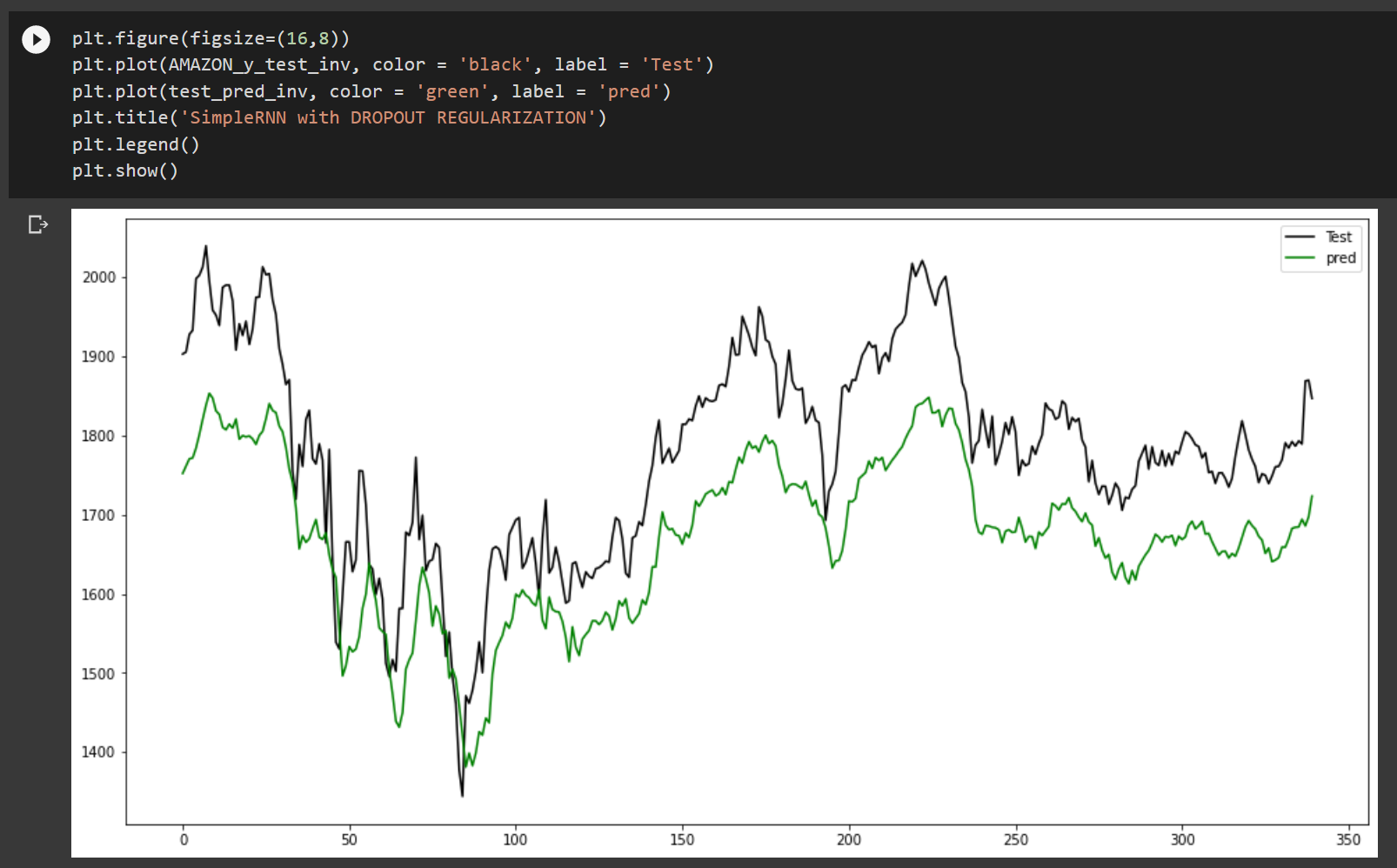
We will compare with the MSE, RMSE of the SimpleRNN using Dropout regularization.

The MSE, RMSE of the train data is a bit higher compared to the above model but for the test data it performs better.



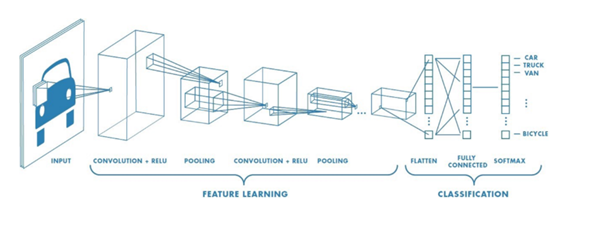
The result is a bit better compare to above model





### CNN

A general architecture of Convolutional Neural Networks



**What is Convolutional Neural Network ?**

• Convolutional Neural Networks (CNNs) are simply neural networks that use Convolution Operation which is a specialized kind of linear operation.

• CNNs are regularized versions of multilayer perceptrons.

• CNNs take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns.

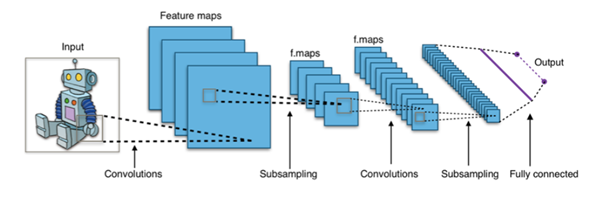
**CNN network structure**

A CNN network is a collection of Convolution layers that overlap and use nonlinear activation functions like ReLU and tanh to activate the weights in the nodes. Each class after passing activation functions will generate more abstract information for the next classes.

Each class after passing activation functions will generate more abstract information for the next classes. In the feedforward neural network model, each input neuron gives each output neuron in subsequent layers.

This model is called fully connected layer or fully connected network. In the CNNs model, the opposite is true. The layers are linked together through the convolution mechanism.

The next layer is the convolution result from the previous layer, so we have local connections. Thus, each neuron in the next layer is generated from the result of a filter applied to a local image region of the previous neuron.



In the CNN model, there are two aspects that need attention: invariance (Location Invariance) and coherence (Compositionality). With the same object, if this object is projected in different degrees (translation, rotation, scaling), the accuracy of the algorithm will be significantly affected.

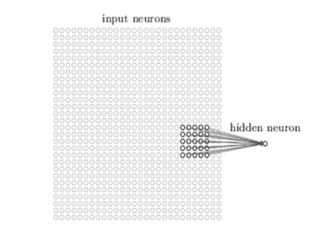
Pooling layer will give you invariant to translation, rotation, and scaling. Local associativity gives us lower-to-higher and more abstract levels of information representation through convolution from filters.

That is why CNNs produce models with very high accuracy. Just like how humans perceive objects in nature.

**Local receptive field**

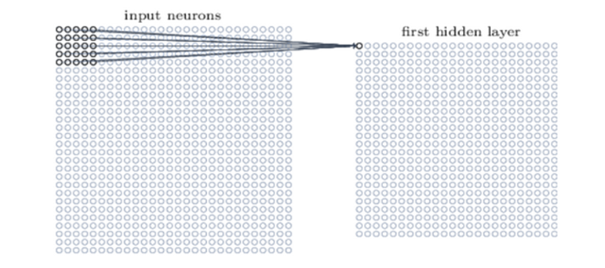
The input to the CNN network is an image. For example, for an image of size 28×28, the corresponding input is a matrix of 28×28 and the value of each pixel is a cell in the matrix. In the traditional ANN model, we will connect the input neurons to the image layer.

However, in CNN we do not do that, we only connect in a small area of ​​input neurons as a filter with size 5×5 corresponding (28-5 + 1) 24 input pixels. Each connection will learn a weight and each hidden neuron will learn a bias. Each of these 5×5 areas is called a local receiving field.

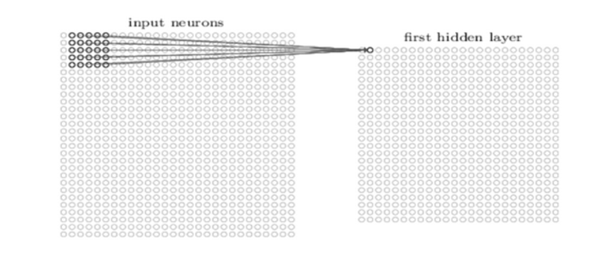


In general, we can summarize the steps of creating a hidden layer in the following ways:

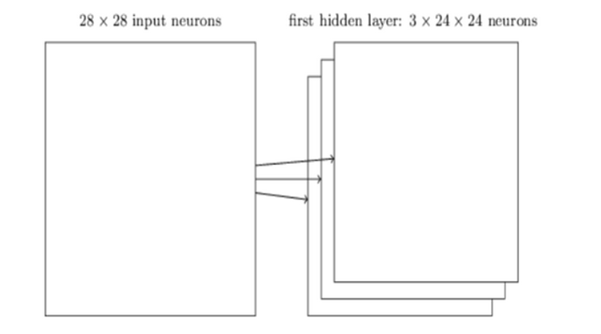
1. Generate the first hidden neuron in the hidden layer 1



1. Shifting the filter to the right one column will create a second hidden neuron



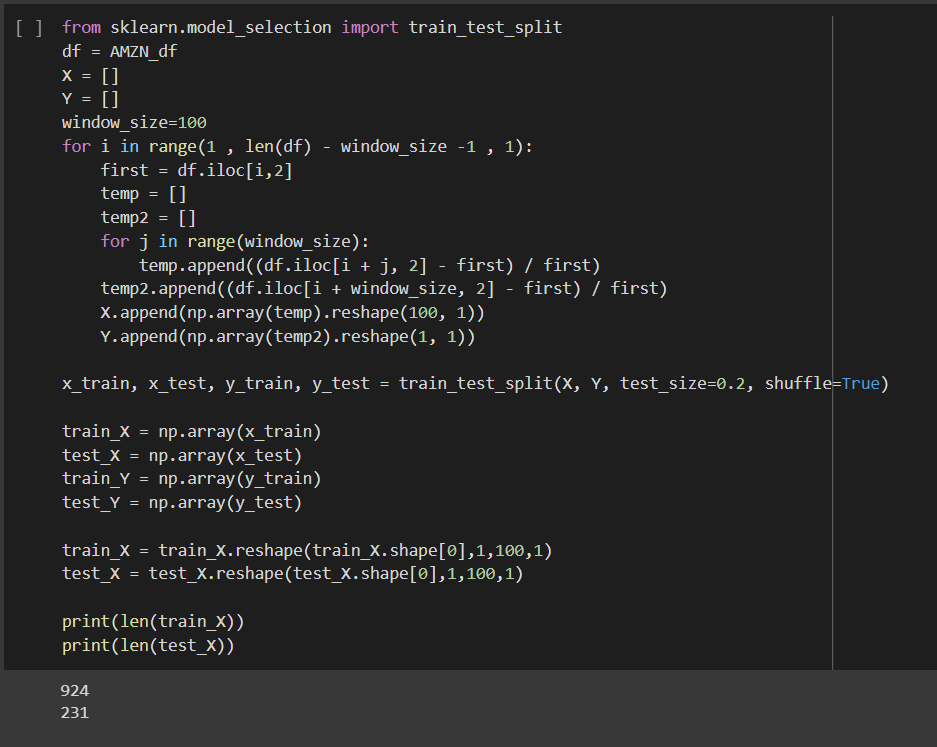
For image recognition problems, people often call the input layer matrix a feature map, the weight determining the features is shared weight and the deviation defining a feature map is the shared bias. So the simplest is that through the above steps we only have 1 feature map. However in image recognition we need more than one feature map



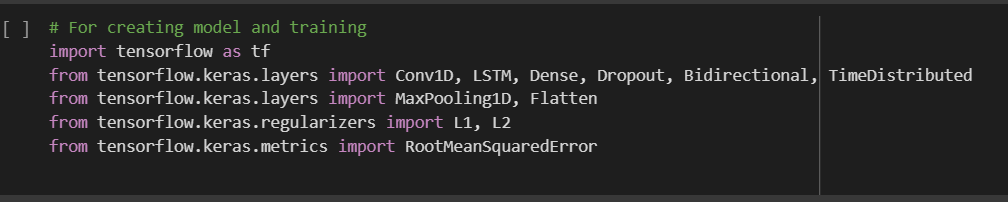
Thus, the local receptive field is suitable for image data analysis, helping to select the most valuable image regions for classification evaluation.

**Code**

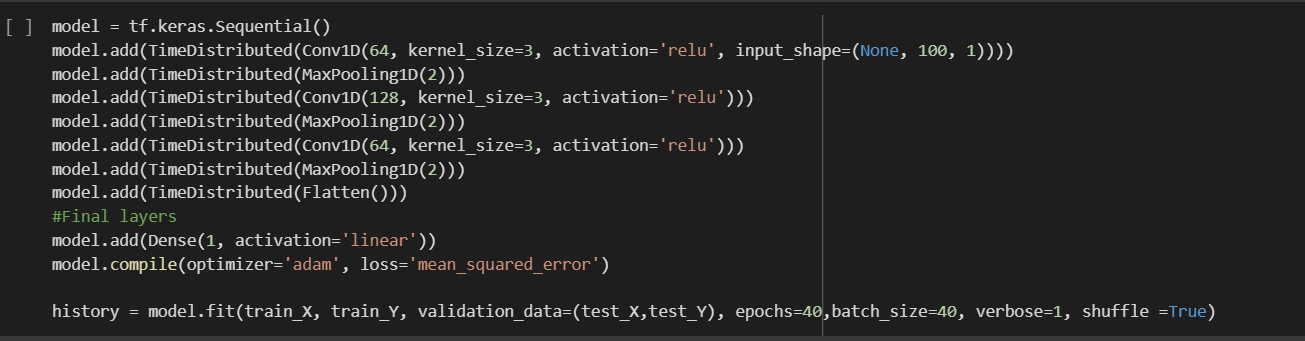
The data has been analyzed but it must be converted into data of shape [100,1] to make it easier for CNN to train on... Else it won't select necessary features and the model will fail

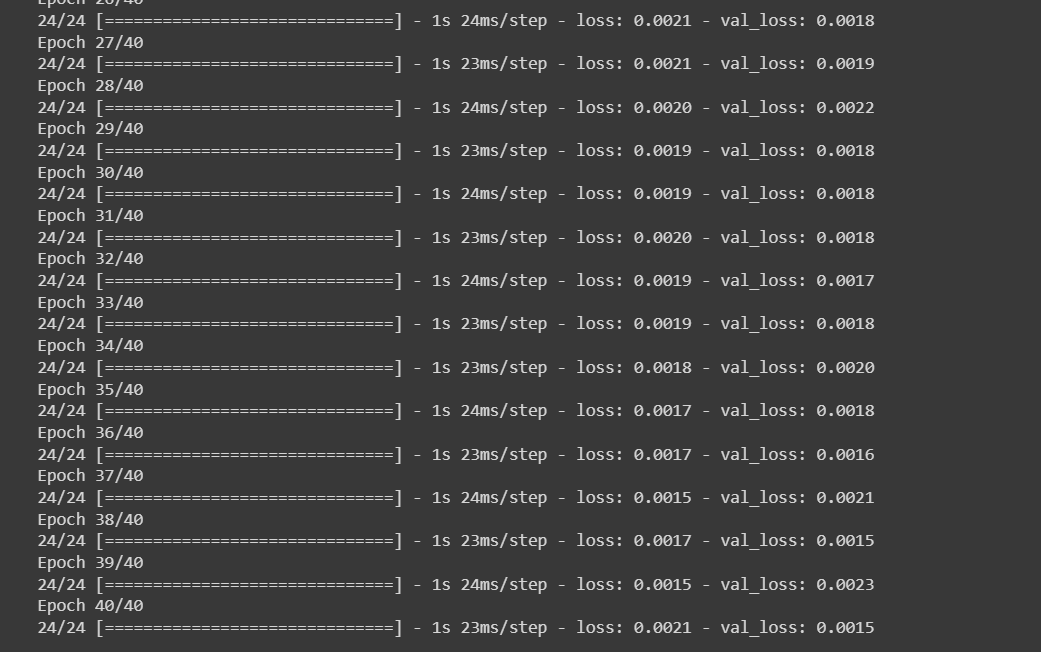


Creating models and training

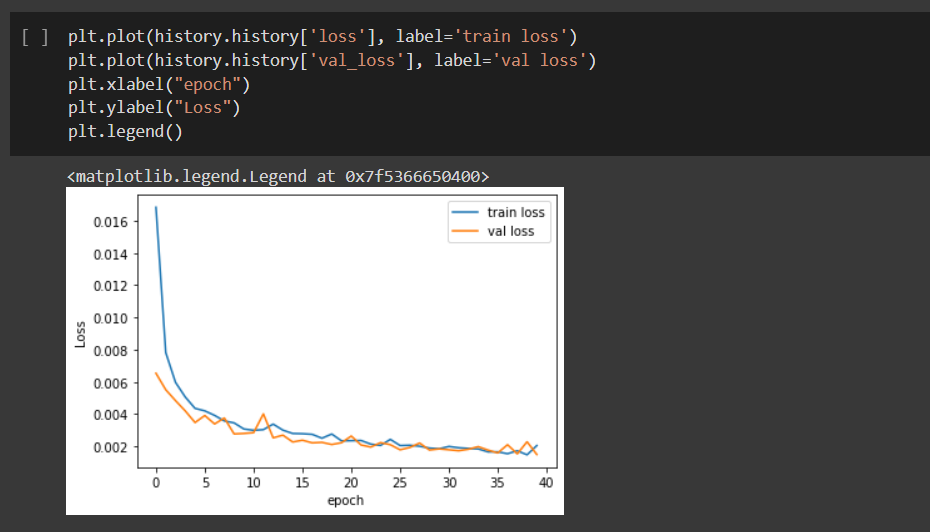


Training models

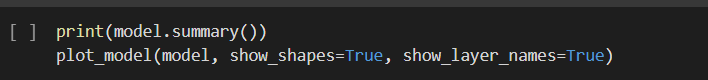


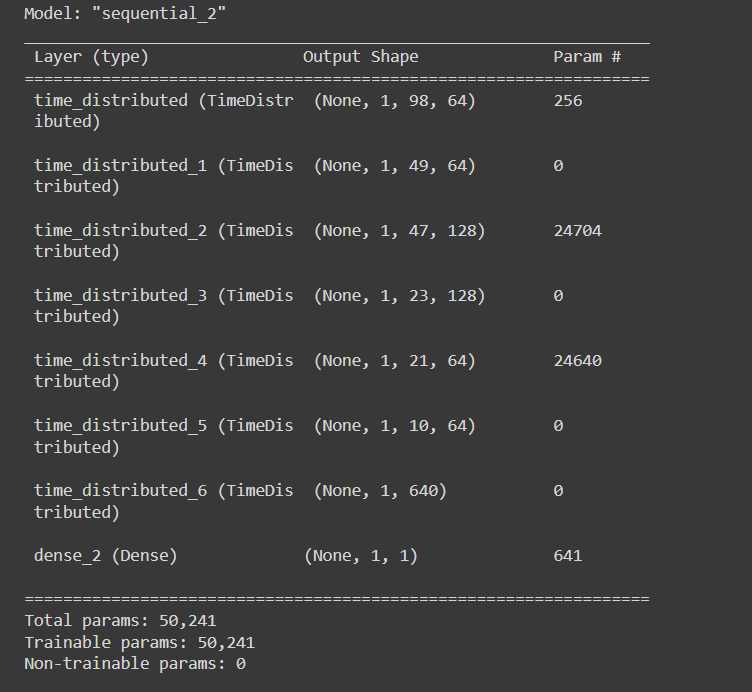


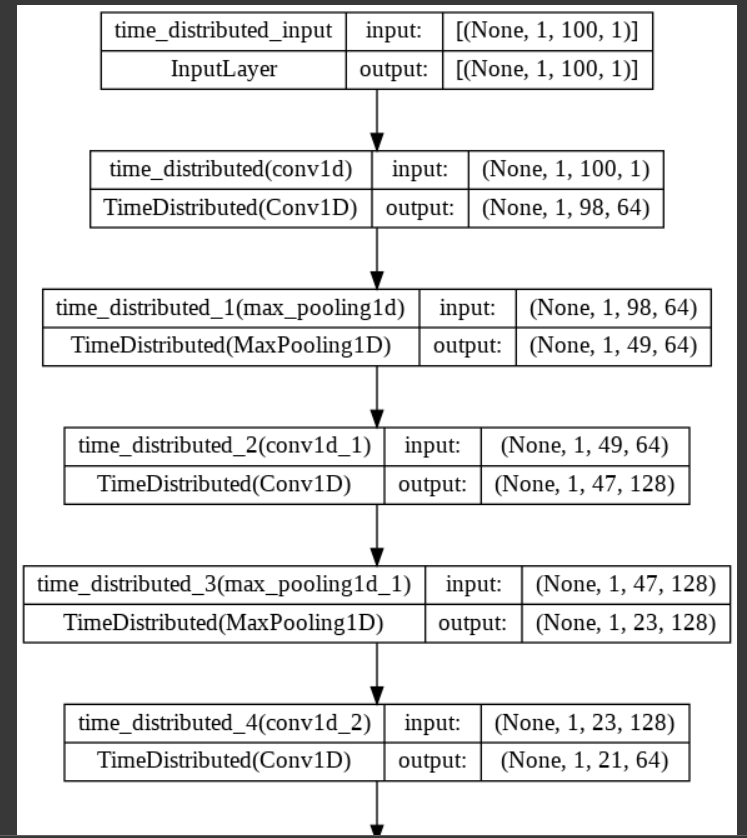
Result

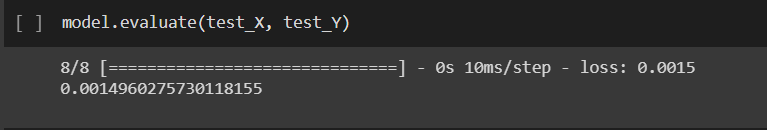


After the model has been constructed, we'll summarise it



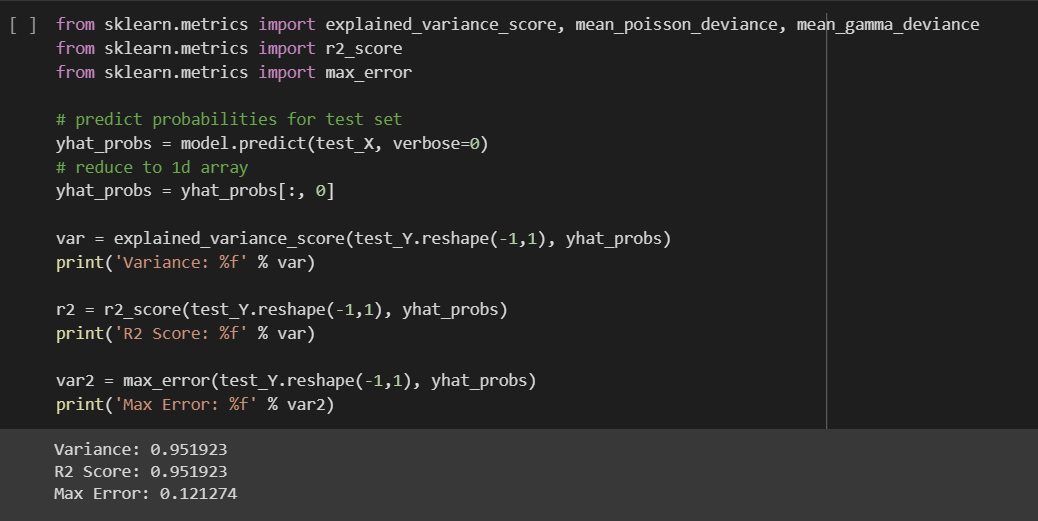




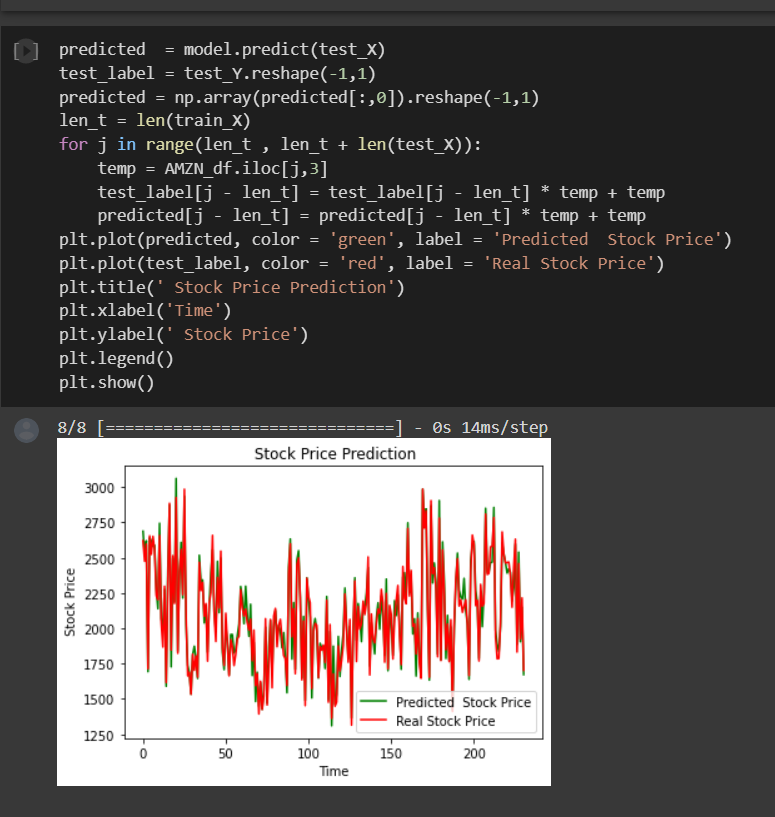


Prediction for models

Predict probabilities for test set



Result for the prediction models



It models is the good, stock real is not higher than stock predicted

### LSTM

### LSTM network model

### The long short-term memory (LSTM) technique largely addresses the backpropagation's vanishing gradient issue. A gating mechanism is used by LSTMs to regulate the memoizing procedure. Through gates that open and close, information may be read, written, or stored in LSTMs. These gates implement element-wise multiplication by sigmoid ranges between 0 and 1, storing the memory in analog format. Because it is differentiable, analog is a good choice for backpropagation.

### Recurrent neural networks have the drawback of having a short-term memory that allows earlier information to be stored in the present cell. For longer sequences, this skill, however, rapidly declines. The LSTM models were developed to address this issue and have even longer memory retention.

### Recurrent neural networks have the drawback of just storing the prior data in its "short-term memory." The longest preserved information is simply deleted and replaced with fresh data when the memory in it runs out. The LSTM model makes an effort to get around this issue by only keeping a few pieces of information in short-term memory. The so-called Cell State is where this short-term memory is kept. There is also the hidden state, which is something we are already familiar with from regular neural networks.

# Bidirectional LSTM

### LSTM is a Gated Recurrent Neural Network, and bidirectional LSTM is just an extension to that model. The key feature is that those networks can store information that can be used for future cell processing. We can think of LSTM as an RNN with some memory pool that has two key vectors:

### · (1) Short-term state: keeps the output at the current time step.

### · (2) Long-term state: stores, reads, and rejects items meant for the long-term while passing through the network.

### In bidirectional LSTM, we train two models rather than just one. The first model picks up the input's sequence, while the second model picks up the opposite of that sequence.

### 

### Since we have two trained models, we must create a system to integrate them. It is commonly known as the Merge stage. One of the following functions is merging:

### · Sum

### · Multiplication

### · Averaging

### · Concatenation (default)

### 

### 1. Which recent and past information is retained and which is deleted is chosen at the so-called Forget Gate. Both the concealed status from the prior run and the present status are included. The sigmoid function receives these variables and can only output values between 0 and 1. A number of 0 indicates that all prior knowledge is lost, whereas a value of 1 indicates that all prior knowledge is retained. The results of this are multiplied by the current Cell State, which causes knowledge that is no longer necessary to retain to be lost out since it is multiplied by 0.

### 2. The value of the current input to complete the task is determined at the input gate. The weight matrix from the previous run is multiplied by the hidden state and the current input for this purpose. The new Cell State is created by adding all information that appears to be significant from the Input Gate to the Cell State. The following run will make use of this new Cell State, which is now the short-term memory's current state.

### 3. The output of the LSTM model is then computed in the Hidden State at the Output Gate. Depending on the context, it may be, for instance, a term that enhances the sentence's meaning. To do this, the cell state is multiplied after being activated with the tanh function, and the sigmoid function determines what information can pass through the output gate.

### Tanh

### Tanh is an activation function that is not linear. It controls the values moving across the network, keeping them within the range of -1 and 1. A function whose second derivative may endure for a longer period of time is required to prevent information fading. It's possible that certain values will grow gigantic, further making other values irrelevant. You can see how the function keeps the value 5 within the bounds.

### Sigmoid

### The family of non-linear activation functions includes the sigmoid. The gate is what keeps it inside. Sigmoid, in contrast to tanh, preserves values between 0 and 1. The network benefits when the data is updated or forgotten. Information is deemed lost if the multiplication yields a value of 0. Similarly, if the value is 1, the data is retained.

### This will aid the network in learning which data can be lost and which should be kept.

## To Sum up!

### Since LSTM networks can accomplish whatever RNNs might be able of with much more refinement, they are in fact superior to RNNs. Although they might be frightening, LSTMs do offer superior outcomes and represent a significant advancement in deep learning. You can anticipate more precise predictions and a clearer understanding of your options as these technologies proliferate.

### 

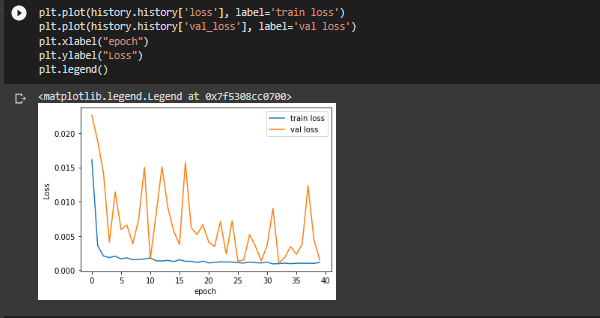
### Code:

### 

Create layers of LSTM after creating and training model.

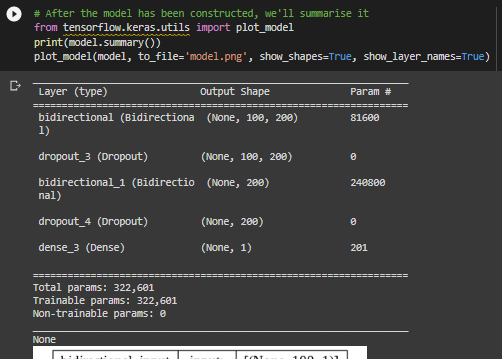
Result:



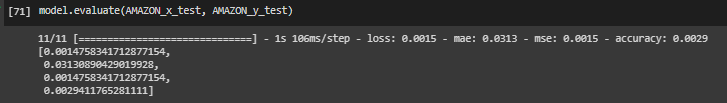


The train loss value is quite different from the real loss value

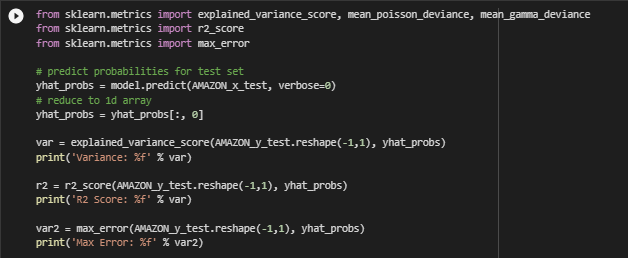




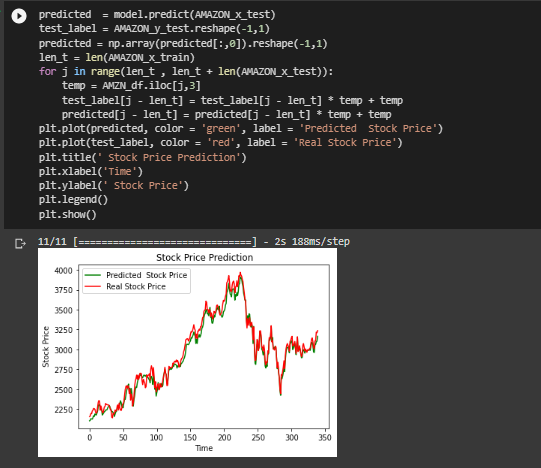
The result after we summarized the model.



Evaluate model and receive the result with loss: 0.0015, mse: 0.0015 and accuracy: 0.0029



Predict model.



The result of this model is quite good since the predicted stock prices are so close to the real ones.

## References:

**RNN**

1. [**https://www.simplilearn.com/tutorials/machine-learning-tutorial/stock-price-prediction-using-machine-learning**](https://www.simplilearn.com/tutorials/machine-learning-tutorial/stock-price-prediction-using-machine-learning)
2. [**https://www.kaggle.com/code/ahmedelrawy/amazon-stock-price-action-with-rnn**](https://www.kaggle.com/code/ahmedelrawy/amazon-stock-price-action-with-rnn)
3. [**https://www.youtube.com/watch?v=H6du\_pfuznE&t=397s**](https://www.youtube.com/watch?v=H6du_pfuznE&t=397s)
4. [**https://www.kaggle.com/code/gcmadhan/amazon-stock-prediction-nn-fbprophet**](https://www.kaggle.com/code/gcmadhan/amazon-stock-prediction-nn-fbprophet)
5. **https://www.kaggle.com/code/raoulma/ny-stock-price-prediction-rnn-lstm-gru/notebook#3.-Manipulate-data-**

**LSTM**

### [Long Short-Term Memory Networks (LSTM)- simply explained! | Data Basecamp](https://databasecamp.de/en/ml/lstms#:~:text=LSTM%20models%20are%20a%20subtype,term%20memory%20or%20discard%20it).

### [Introduction to LSTM Units in RNN | Pluralsight](https://www.pluralsight.com/guides/introduction-to-lstm-units-in-rnn)