STOCK PREDICTION USING THE GIVEN DATASETS

- Load the CSV file as a DataFrame using Pandas. Since the data is indexed by date (each row represents data from a different date), we can also index our DataFrame by the date column
- Plotting the High and Low points of Netflix stock
- It will be challenging for a model in the stock prediction using machine learning project to correctly estimate the rapid changes that we can see.
- Similarly, plotting the Open and Close value of the stock for each day gives equivalent observations.
- We use matplotlib to plot the DataFrame columns directly against the Date index column. To make things flexible while plotting against dates, lines 6-8 convert our date strings into datetime format and plot them cleanly and legibly. The interval parameter in line 7 defines the interval in days between each tick on the date axis.
- **Importing the Libraries for Stock Price Prediction Project**
- We will be building our LSTM models using Tensorflow Keras and preprocessing our stock prediction machine learning data using scikit-learn. These imports are used in different steps of the entire process, but it is good to club these statements together.
- **Data Preprocessing for Stock Market Prediction using Machine Learning**
- As with any other machine learning model, it is always good to normalize or rescale the data within a fixed range when dealing with real data. This will avoid features with larger numeric values to unjustly interfere and bias the model and help achieve rapid convergence in the machine learning stock prediction project.
- First, we define the features and the target
- Next, we use a StandardScaler to rescale our values between -1 and 1.
- Scikit-learn also provides a popular MinMaxScaler preprocessing module. However, considering the context, stock prices might max out or minimise on different days, and using those values to influence others might not be great. The change in values from using either of these methods would not be much, so we stick to StandardScaler.
- So, the next step would be to split it into training and testing sets. As explained above, the training of an LSTM model requires a window or a timestep of data in each training step. For instance, the LSTM will take 10 data samples to predict the 10th one by weighing the first nine input samples in one step. So, we need a different approach than the train_test_split provided by scikit-learn.
- Let's define a splitting function called lstm_split() which will make windows of size "n_steps" starting from the first sample of data and ending at n_steps'th sample (if n_steps=10, then the 10th sample) from the end. We understand the latter part because, for each time step, LSTM will take n_steps-1 samples for training and predict the last sample. Loss calculation is done based on the error in this prediction. So if n_steps=10, you cannot use the last 9 samples to predict anything because the "10th" data point for the current step does not exist in the dataset.
- The function below takes the entity.
- Given the simplicity of the model and the data, we note that the loss reduction stagnates after only 20 epochs. You can observe this by plotting the training loss against the number of epochs, and LSTM does not learn much after 10-20 epochs.