EE626 Stock Price Prediction Group Project

Nishant Bhat – 180106032

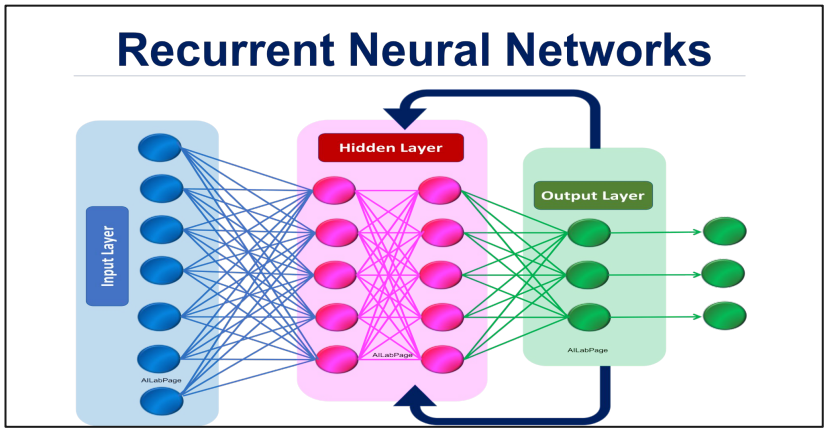
Tahir D'Mello – 180106055

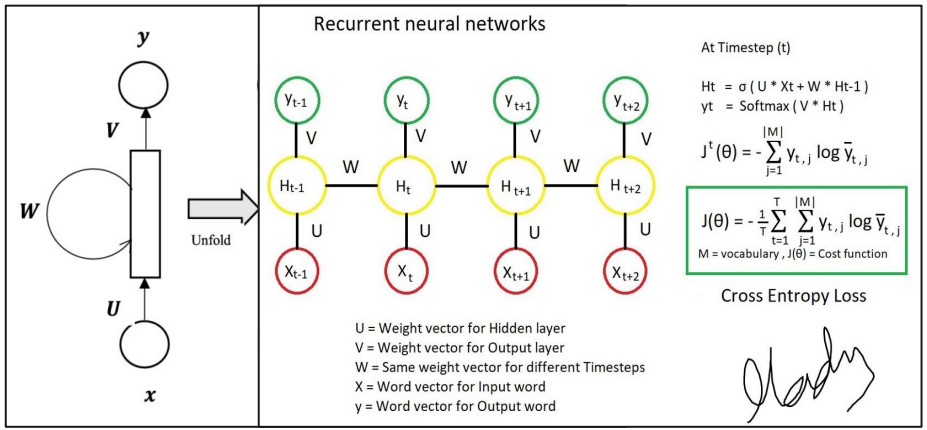
Siddhant Jagtap – 180106050

Aditi Madkaikar – 180106007

Recurrent Neural Network (RNN) Basics

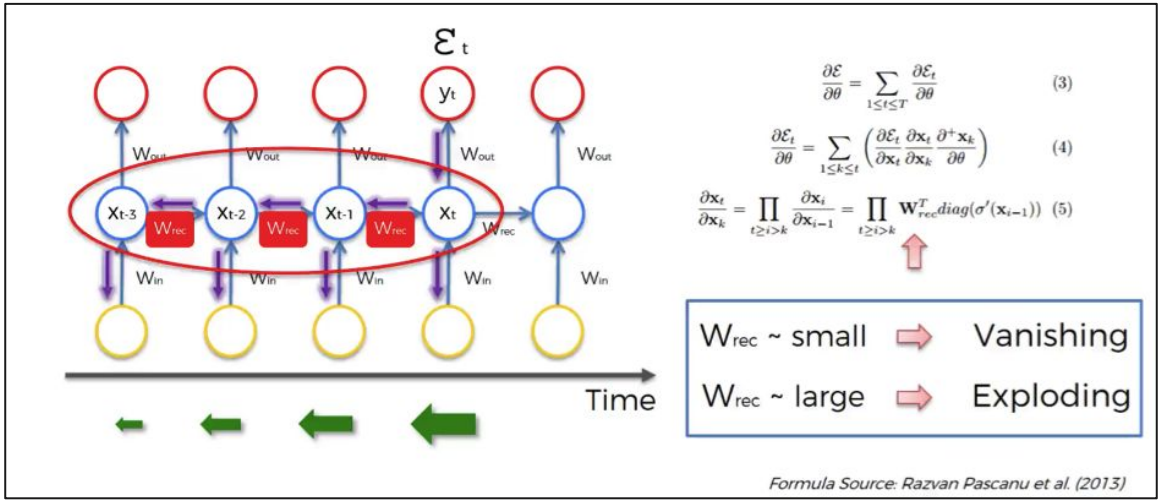
- Recurrent neural network (RNN) is a type of **artificial neural networks** where connections between nodes form a **directed graph** along a **temporal sequence.**
- This allows it to exhibit **temporal dynamic behavior**.
- While RNNs learn similarly while training they also remember things **learnt from prior input(s)** while generating output(s).





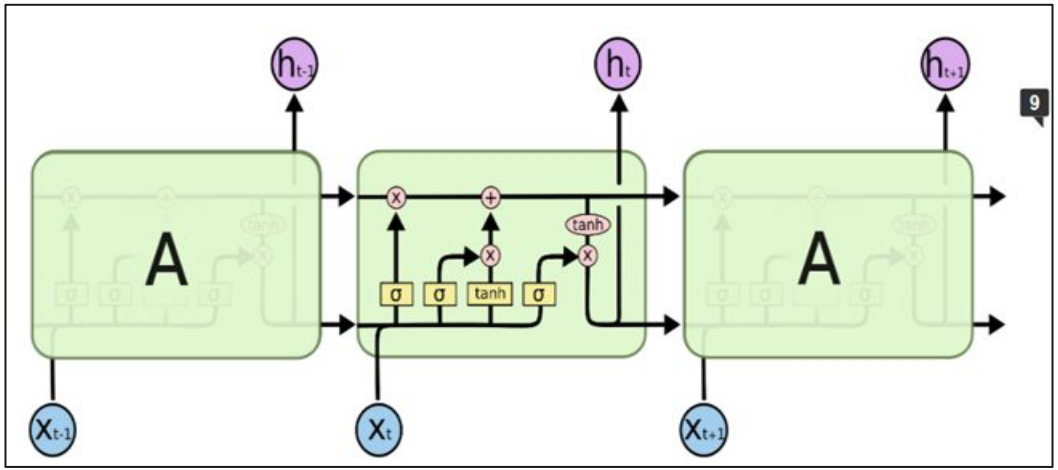
RNN: Vanishing And Exploding Gradients

- An RNN is unable to learn to connect the information in large gaps due to one of the following reasons:
  - Vanishing gradient : If **Wrec is too small**, gradients coming from the deeper layers **shrink exponentially** due to the large distance travelled until they vanish and make it **impossible** for the model **to learn**
  - Exploding gradients: If **Wrec is too large**, gradients coming from the deeper layers get **larger exponentially** due to the large distance travelled and eventually **blow up** and **crash** the model



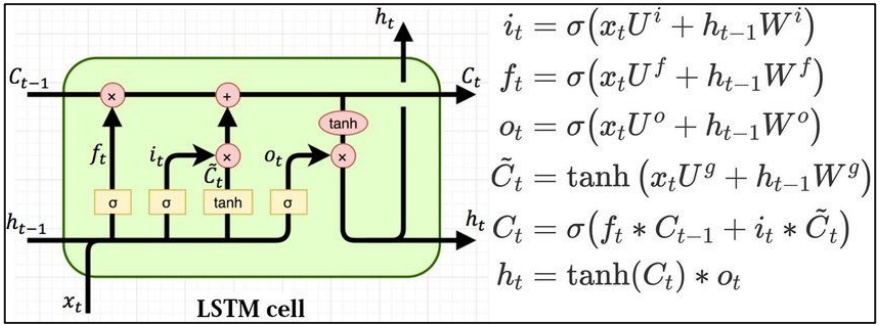
Long Short Term Memory RNNs

- In order to solve this Long Short Term Memory networks create a **special kind of RNN**, capable of **learning long-term dependencies**
- To avoid vanishing/exploding gradients, the **cell state runs straight** down the entire chain, with **only minor linear interactions**.
- It is thus very easy for information to just flow along it unchanged.



Long Short Term Memory RNNs: Equations

- Apart from the normal RNN equations, the LSTM does have some gates composed out of a sigmoid neural net layer and a pointwise multiplication operation which optionally let information through.
- LSTM has three gates, to protect and control the cell state:
  - **Forget Layer (ft)** - Sigmoid input of ht−1 and xt. Outputs a number between 0 (completely discard) and 1 (completely retain) for each number in cell state Ct−1.
  - **Input Layer (it and C*t)** - Sigmoid layer (it) decides which values to be updated. tanh layer (C*t) creates a vector of new candidate values to be added to the state Ct-1. These are multiplied to give complete input.
  - **Output Layer (ht)** - Sigmoid layer which decides what parts of the ht-1 and xt to output to ht. Multiplied by tanh(Ct-1) to output the parts decided in a [-1,1] range.
- These three gates are all combined by linear operations st. **Ct = sigma(ft*Ct-1 +it*C*t)** and **ht = tanh(Ct)*ot** to give the next set of final cell outputs.



LSTM Advantages for Stock Prices

Advantages of using an LSTM for stock price prediction are as follows:

1. LSTM due to their **non parametric nature** are generally considered superior to the traditional statistical autoregressive models (ARIMA, SARIMAX) which are parametric and require **a lot of fine tuning.**
2. LSTM can predict prices of stocks comparatively well as it **does not have to consider the stationarity factor** which is important in time series predictions by ARIMA models.
3. LSTM are very powerful in sequence prediction problems because they're able to **store past information**. Stock Prices are reflections of their previous lagged versions. Today's prices are generally dependent on recent past days at max.

1. Data Collection - Source, Details and Description

Stock data (closing prices) for Apple Inc. [AAPL] was acquired from **Yahoo Finance** for 3814 days from **January 1, 2006 to February 28, 2021**. [Stored in 'apple.csv']

	Opening Price	Closing Price	Volume
Mean	26.985750	26.991520	4.430121e+08
Standard Deviation	26.804558	26.804442	3.960121e+08
Minimum	1.847500	1.809643	4.544800e+07
Maximum	143.600006	143.160004	3.372970e+09

Opening Price - First price paid for a share of stock during the business hours of the exchange
Closing Price - Last price paid for a share of stock during the business hours of the exchange
Volume - Number of shares traded in a stock on a trading day.

Closing Price is the value being **predicted** and the **prediction series**.
Opening Price and Volume are used as **features** for prediction.

2. Data Preprocessing - Cleaning and Normalization

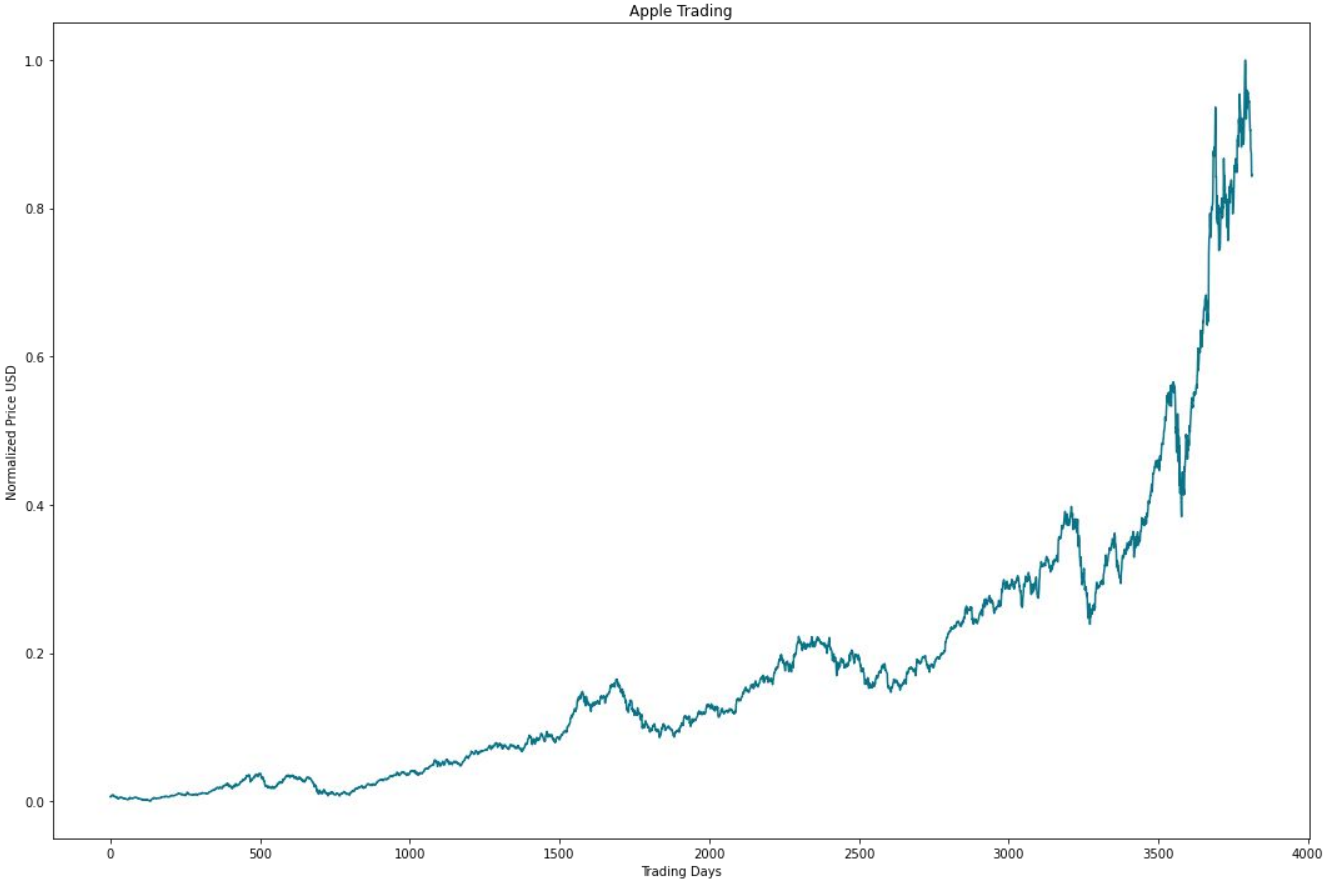
Unneeded data columns were dropped from the downloaded data.
Only Opening Price, Closing Price and Volume was retained.
All entries were indexed and formatted as integers.

The Keras MinMaxScaler function was used to scale and translate each feature individually such that it is in the given range on the training set

X_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))

X_scaled = X_std * (max - min) + min

This was done as LSTMs are sensitive to data fluctuation. Thus, they must be normalized to a [0,1] range.
Data stored in ‘apple_preprocessed.csv’.



Normalized Apple Stock Price USD vs Trading Days

3. Basic LSTM - Data Splitting and Unrolling

The total data was split into:

- Testing Set - 450 trading days (closing prices) [~12%]
- Training Set - Remaining ~88% was used for training the model.

LSTM in Keras requires a specific format in which it has to be passed for the model to get trained,
Conversion of the normal stock data into this format is called the unrolling.

Unrolling in this context is expanding and choosing the **window of past data** to be used to **predict** the next needed values.
In this case, unroll length chosen was 50, implying the use of a 50 day window to predict the closing price for the next 5 days.

The shape of input training and testing dataset required for the LSTM model should be of 3 dimensions column wise :

[Number of Samples, Unrolling Factor (or) Time steps, Number of Features]

Closing Price is chosen as the prediction series and the value being predicted which contributes to number of samples and unrolling factor.
Opening Price and Volume contribute to the number of features chosen. Therefore, we have 3 as the third column dimension.

The LSTM model will repeatedly take in an input of 50 days and give us predictions for the next 5 days.

3. Basic LSTM - Model Details and Compilation

Basic model details are as follows:

1. 3 Layers were chosen consisting of **2 LSTM** and **1 Dense Layer** with **linear activation function** (since it is a regression task)

2. Layer wise number of nodes chosen for basic LSTM model are as follows:
   i. LSTM (first layer) : 50
   ii. LSTM (second layer) : 100
   iii. Dense Layer : 1

3. Optimizer function used is **Adam** as it combines **RMS Propagation** and **Momentum Based Gradient Descent** both of which are useful for stock market data since they use the concept of a moving average.

4. **Batch Size** refers to the number of training examples used in one iteration.
   For the basic LSTM model, Keras default batch size of 32 was chosen.

5. **Number of Epochs** defines the number of times the learning algorithm will work through the entire training dataset.
   For the basic LSTM model, number of epochs was set to one.

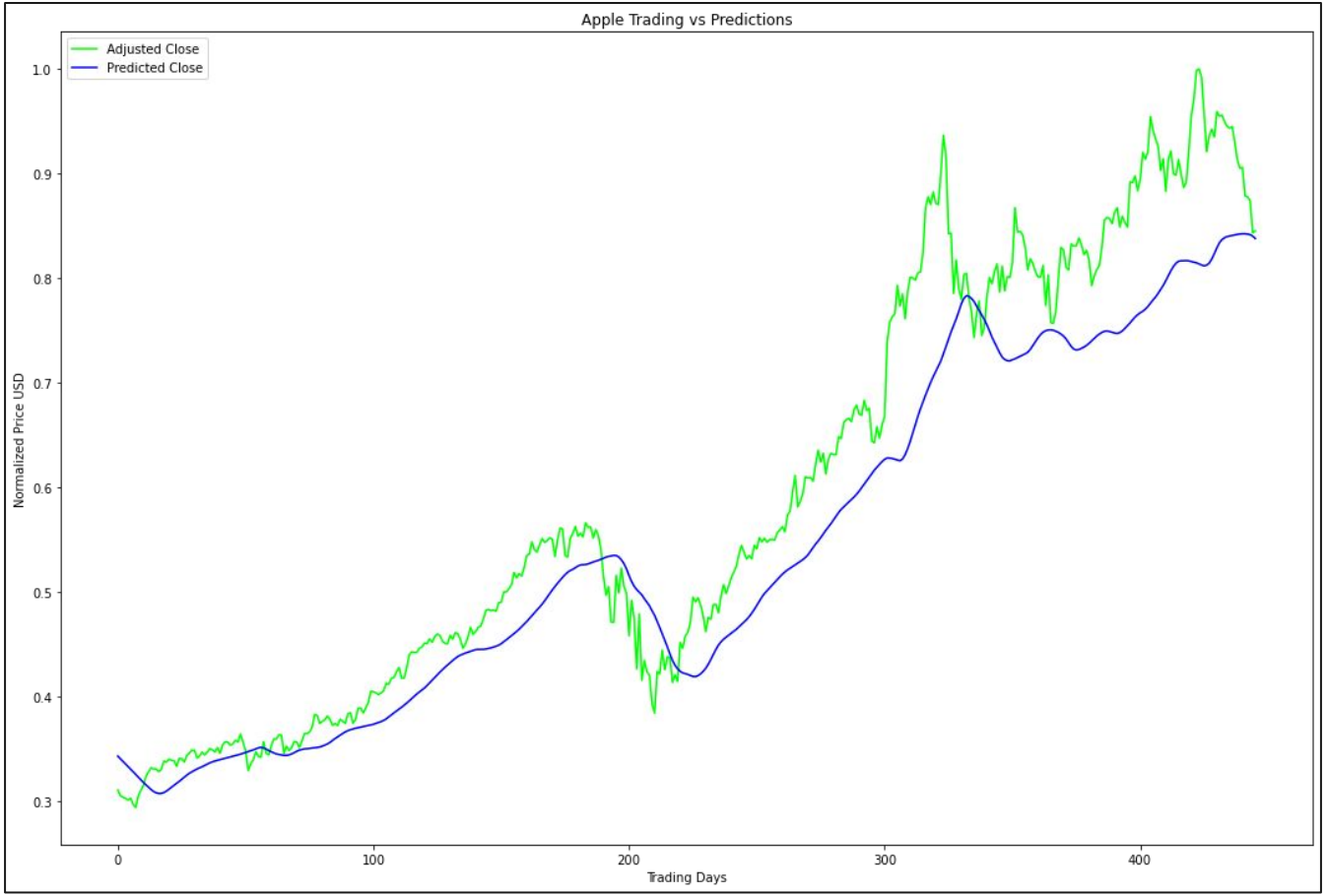
3. Basic LSTM - Training, Testing and Evaluation

Training Data : Model was trained on 3258 trading days which was presented as normalized unrolled data to the sequential LSTM network.
Loss metric used was the **Root Mean Squared Error** and optimizer was **Adam.**

RMSE for training data (Basic LSTM) : 0.00883852

Testing Data : Model was tested on 446 trading days which was presented as normalized unrolled closing prices to the Sequential trained LSTM network.

RMSE for testing data (Basic LSTM) : 0.07076352



Basic LSTM Predictions

4. Improved LSTM Model - Hyperparameters

The basic parameters (number of layers, type of layers, etc.) of the model were kept the same.

Changes were mainly performed in hyperparameter selection:

1. **Hidden Nodes**: Increased the number of Hidden Nodes from 100 to 128. This was done in order to fit the model better and reduce the RMSE.
2. **Dropouts**: Added dropouts of 0.2. Dropouts will basically tend to reduce the overfitting of LSTM to the training data.
3. **Batch Size**: Kept a batch size of 100 instead of training full data to reduce the memory usage.
4. **Number of Epochs**: Increased epochs from 1 to 5 as 1 epoch was not sufficient to update the weights properly for training data.

The goal was to improve RMSE score and reduce overfitting.

3. Improved LSTM - Training, Testing and Evaluation

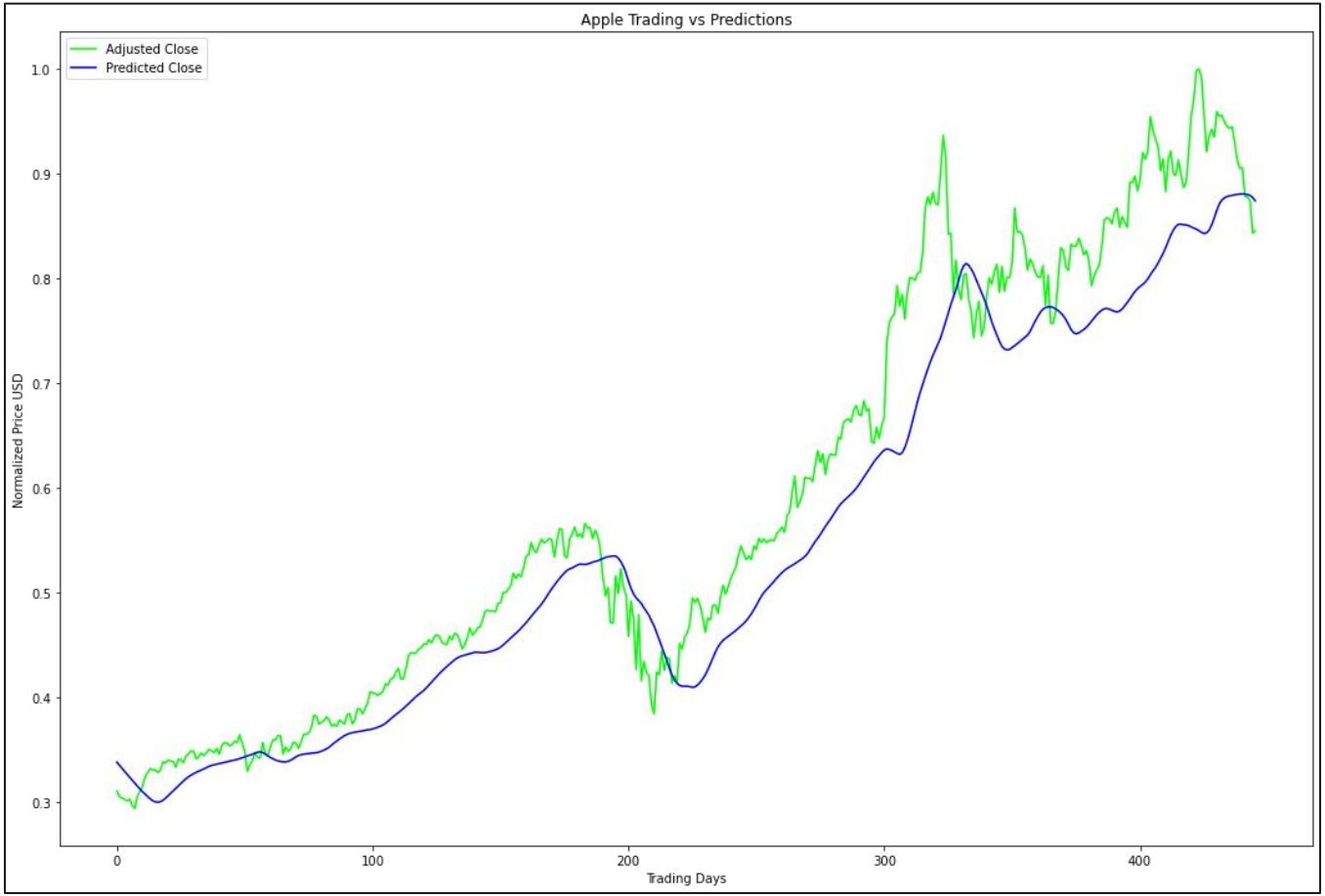
Training Data : Model was trained on 3258 trading days which was presented as normalized unrolled data to the Sequential LSTM network.
Metric used was the **Root Mean Squared Error** and optimizer was **Adam.**

RMSE for training data (Improved LSTM) : 0.00896253

Testing Data : Model was tested on 446 trading days which was presented as normalized unrolled data to the Sequential trained LSTM network.

RMSE for testing data (Improved LSTM) : 0.06134217

The RMSE for training data remained the same approximately. (0.00896253 & 0.00883852)
There was however a reduction in RMSE for the test dataset of improved LSTM. (0.06134217 < 0.07076352)
This implies significant reduction of **overfitting** factor present in the basic LSTM model.



Improved LSTM Predictions

Summary

This was the LSTM RNN model we built to predict stock closing prices using stock price data.

It observed the behaviour of the last 50 days of the stock price to predict the closing price for the next 5 days.

This model can be improved in many ways but this can function as a basic tool for exploring the stock market.