**2242-CSE-6363-106**

**ML PROJECT**

**TOPIC**

**STOCK PREDICTION USING MACHINE LEARNING**

**TEAM MEMBERS:**

**DARSHIT SHAH - 1002081551**

**PARIDIHI PATEL - 1002169595**

**MIHIR PANDYA - 1002169201**

**NANDANI RABRA - 1002166086**

**CONTENT:**

1. Introduction
2. RNN

* Explanation
* Types of RNN
* Limitations
* Applications

1. LSTM

* Explanation
* Components of LSTM
* Advantages
* Disadvantages

1. Code

* Explanation
* Output

1. Conclusion
2. References

**Introduction**

Stock price prediction is a critical task in finance, involving the forecasting of future price movements of financial assets based on historical data and other relevant factors. The process typically involves several key components:

Data Preprocessing: This step involves cleaning and transforming the raw historical data into a format suitable for analysis. It may include tasks such as handling missing values, scaling features, and creating input features for the predictive model.

Model Selection: Various machine learning and deep learning models can be used for stock price prediction. In this case, LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit) neural networks are popular choices due to their ability to capture temporal dependencies in sequential data like stock prices.

Hyperparameter Tuning: Fine-tuning the hyperparameters of the chosen model is crucial for optimizing its performance. Hyperparameters such as the number of hidden units in the LSTM/GRU layers, learning rate, and dropout rate can significantly impact the model's ability to generalize to unseen data.

Training: Once the model architecture and hyperparameters are selected, the model is trained on the training dataset. During training, the model learns to map input features (e.g., historical stock prices) to output predictions (e.g., future stock prices) by minimizing a chosen loss function (e.g., mean squared error).

Evaluation: The trained model is evaluated on a separate validation dataset to assess its performance and identify any overfitting or underfitting issues. Metrics such as mean squared error, mean absolute error, and accuracy are commonly used to evaluate the model's predictive accuracy.

In this project, the yfinance library was utilized to access historical stock price data for Microsoft. This library provides a convenient interface for fetching stock data from Yahoo Finance, enabling easy access to a wide range of financial datasets.

The dataset was then divided into three subsets: 80% for training, 10% for validation, and 10% for testing. This splitting strategy ensures that the model is trained on a sufficiently large portion of the data while still reserving unseen data for evaluation.

The benefits of stock price prediction include:

Informed Decision-Making: Predictive models can provide valuable insights to investors, helping them make informed decisions about buying, selling, or holding financial assets.

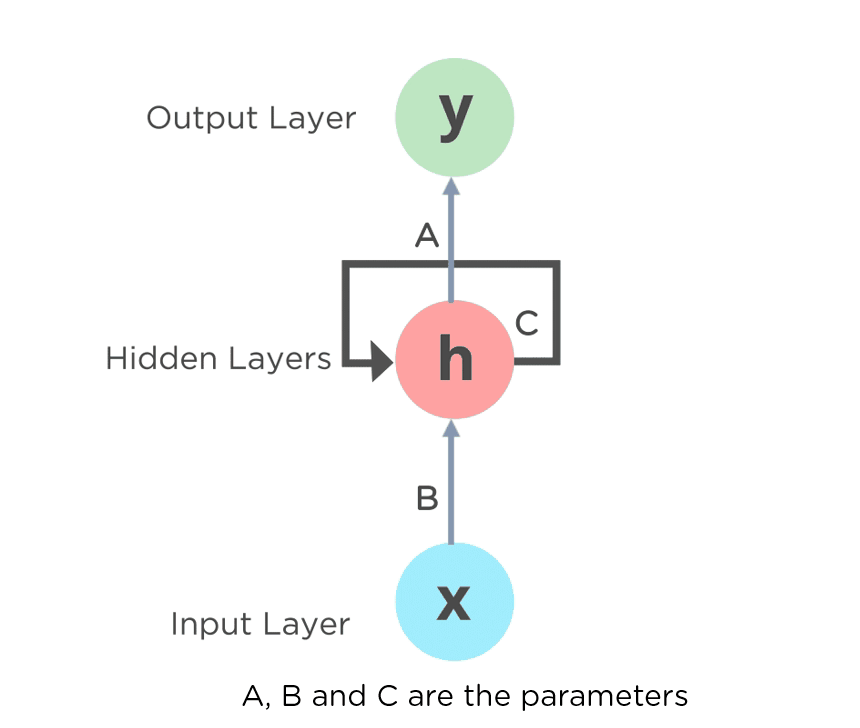
Risk Management: By forecasting future price movements, investors can better manage their risk exposure and hedge against potential losses.

Potential for Higher Returns: Accurate predictions can lead to higher returns by enabling investors to capitalize on profitable trading opportunities and avoid losses during market downturns.

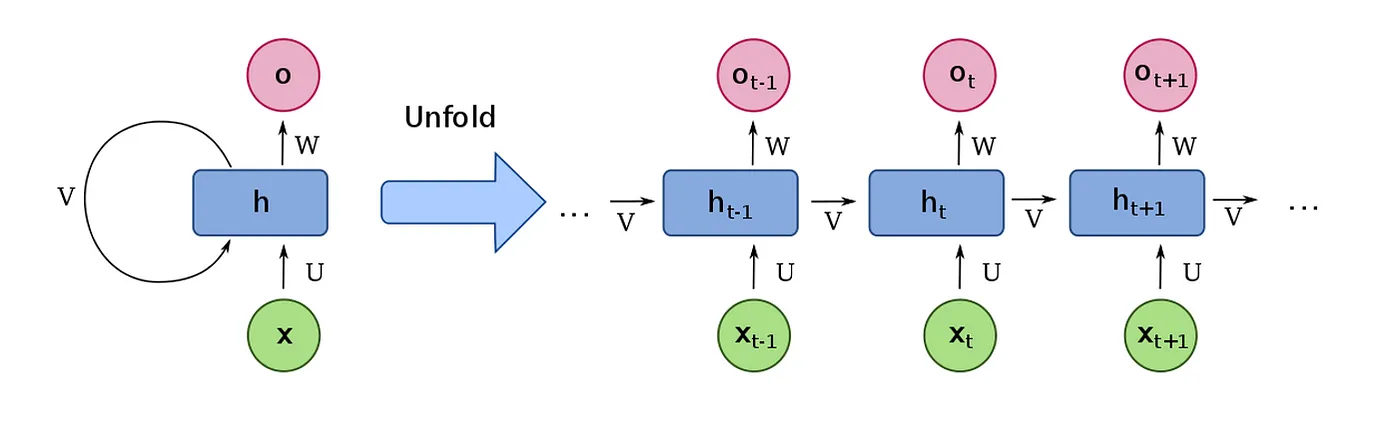
Overall, stock price prediction using machine learning techniques offers a powerful tool for financial analysis and decision-making, with the potential to enhance investment strategies and portfolio management practices.

**RNN**

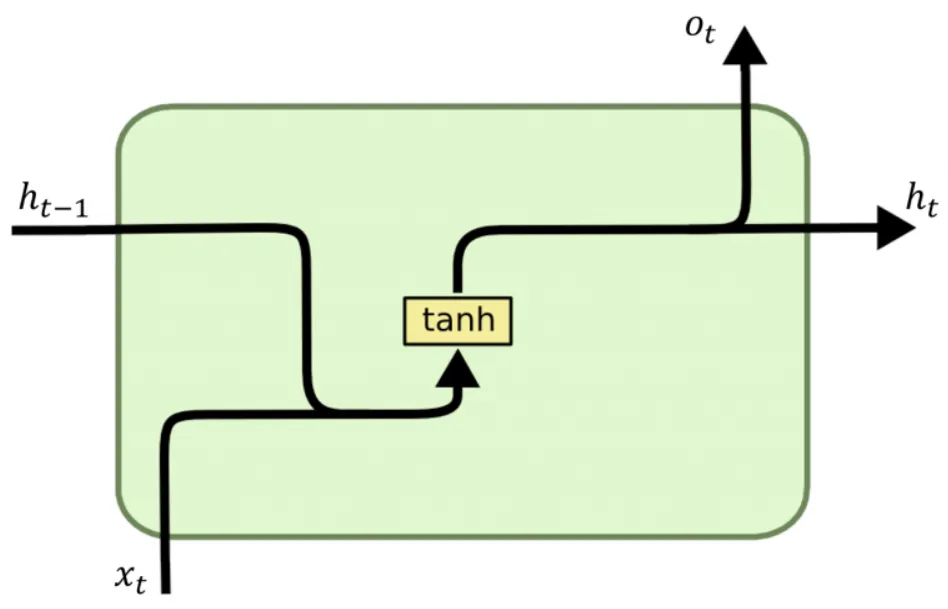
An RNN is a neural network that uses past output to handle new input, enabling it to remember and utilize previous information in sequential data processing.



* Standard NLP techniques like BoW, Word2Vec, or TF-IDF lose sentence structure, impacting model accuracy.
* RNNs address this by utilizing hidden states, retaining and influencing information from previous inputs.
* Feedback connections between hidden and input units enable information retention across time steps.
* RNNs excel in tasks involving sequential data such as NLP, speech recognition, and time series forecasting.
* They've achieved state-of-the-art results in these tasks and are widely used in sequential data ML models.
* RNN architecture comprises input, hidden, and output layers, with multiple hidden layers compressed into a single layer.
* Unfolding an RNN reveals its sequential structure, facilitating understanding of information flow across time steps.

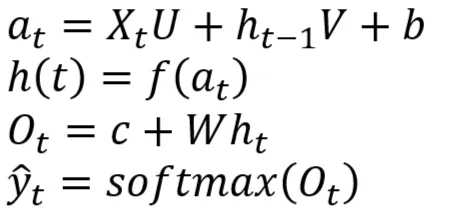


* The output of the previous layer is fed into the next layer for predictions, repeating the structure.
* This characteristic gives rise to the term "Recurrent" Neural Network (RNN).
* X: Input. It can be a word or any sequential data.
* O: Output. It represents the next word or sequence predicted by the network.
* h(t): Represents the hidden state at time t, serving as the network's "memory".
* V: Symbolizes communication from one time-step to another, facilitating information flow.



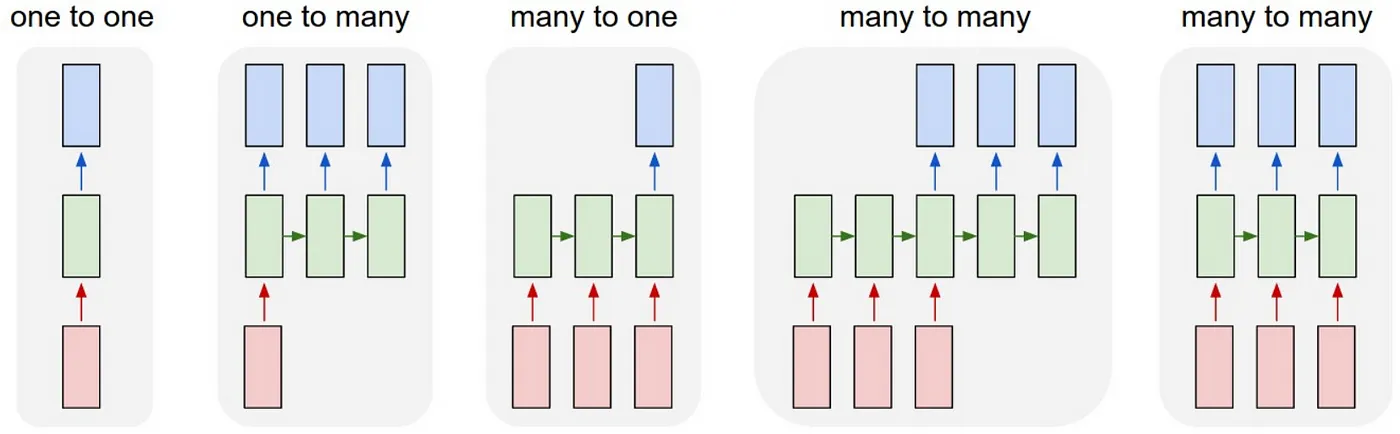
# **Forward Pass**

In the forward pass, the output is calculated as follows:



* h(t) is computed using the current input and the previous time step's hidden state h(t-1)
* The activation function f (e.g., tanh, ReLU) introduces non-linearity into the transformation, such as *f = tanh -> h(t)=tanh(a*ₜ*)*
* U represents the input weight vector corresponding to the input X, while V represents the weight vector corresponding to the memory.
* Backpropagation Through Time (BPTT) calculates gradients and passes them through the network.
* Gradient calculation at time step t depends on all subsequent time steps i.e t+1, t+2,….
* With large sequence lengths, the vanishing gradient problem arises, where gradients diminish over time, hindering the network's ability to remember long sequences.

# **Types of RNNs**



**RNN Modes:**

* One-to-one: Standard processing without RNN, from fixed-sized input to fixed-sized output (e.g., image classification).
* One-to-many: Generates a sequence output (e.g., image captioning from an image input).
* Many-to-one: Processes sequence input (e.g., sentiment analysis classifying sentences).
* Many-to-many: Handles sequence input and sequence output (e.g., machine translation).
* Many-to-many (synced): Manages synced sequence input and output (e.g., video classification).
* Modifications to RNNs:
* Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) address limitations by enhancing the memory and learning capabilities of standard RNNs.

**Limitation of RNN**

Vanishing or Exploding Gradients

* Gradients of the loss function with respect to RNN parameters become extremely small or large during propagation through time.
* Makes training difficult as parameter updates become ineffective due to the small or large gradients.

Training Difficulty:

* RNNs are typically trained using backpropagation, which can be computationally expensive and time-consuming.
* Sensitivity to hyperparameters such as learning rate and number of layers adds to the training challenge.

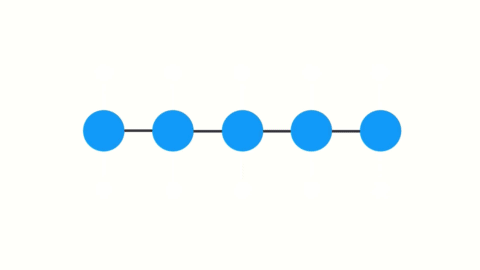
Limitation in Processing Long Sequences:

* Gradients of the loss function can diminish significantly over time, hindering the RNN's ability to learn long-term dependencies.
* Difficulty in capturing and retaining information across lengthy sequences reduces the effectiveness of RNNs in processing long sequences.

## **Applications of RNN’s**

* Text Generation
* Machine Translation
* Visual Search, Face detection, OCR
* Speech recognition
* Semantic Search
* Sentiment Analysis
* Anomaly Detection
* Stock Price Forecasting

**Long Short-Term Memory (LSTM)**



Functionality:

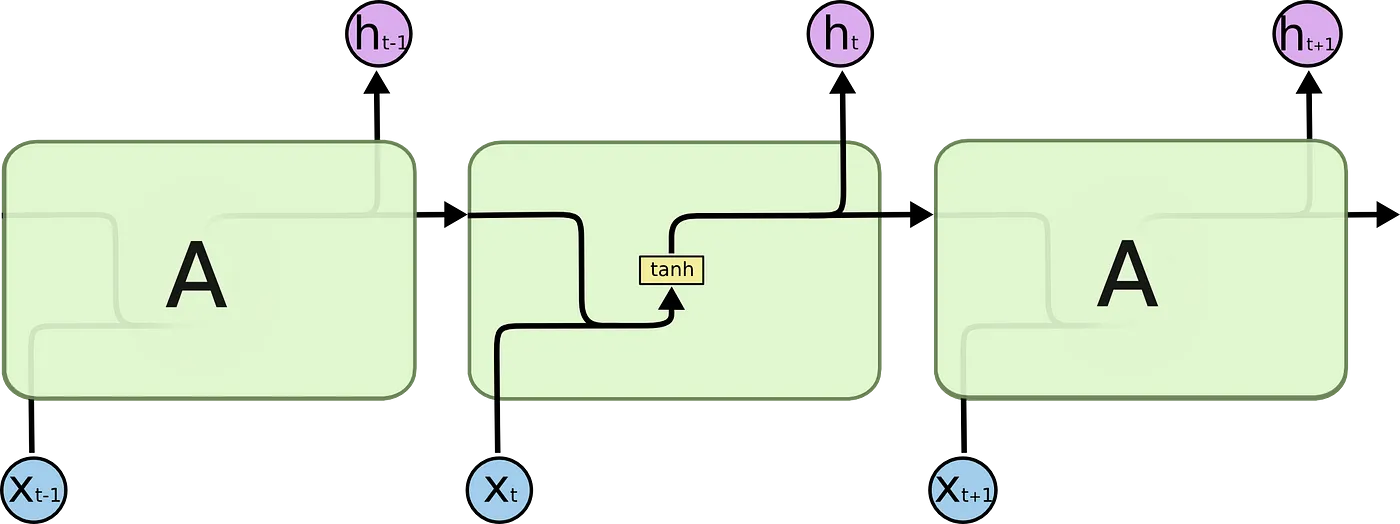
* Solves the vanishing gradient problem by selectively ignoring (forgetting) irrelevant information in the network.
* Determines what information to remember or discard based on the relevance of prior inputs.

LSTM Architecture:

* Contrasts with the simplicity of RNNs by incorporating a more sophisticated structure.

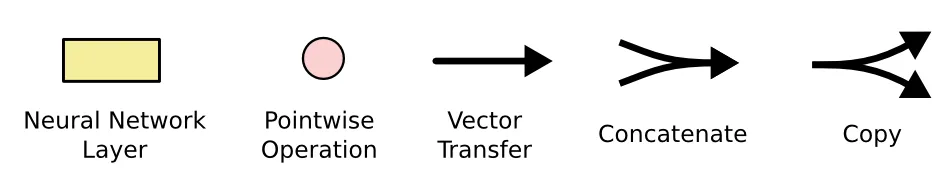
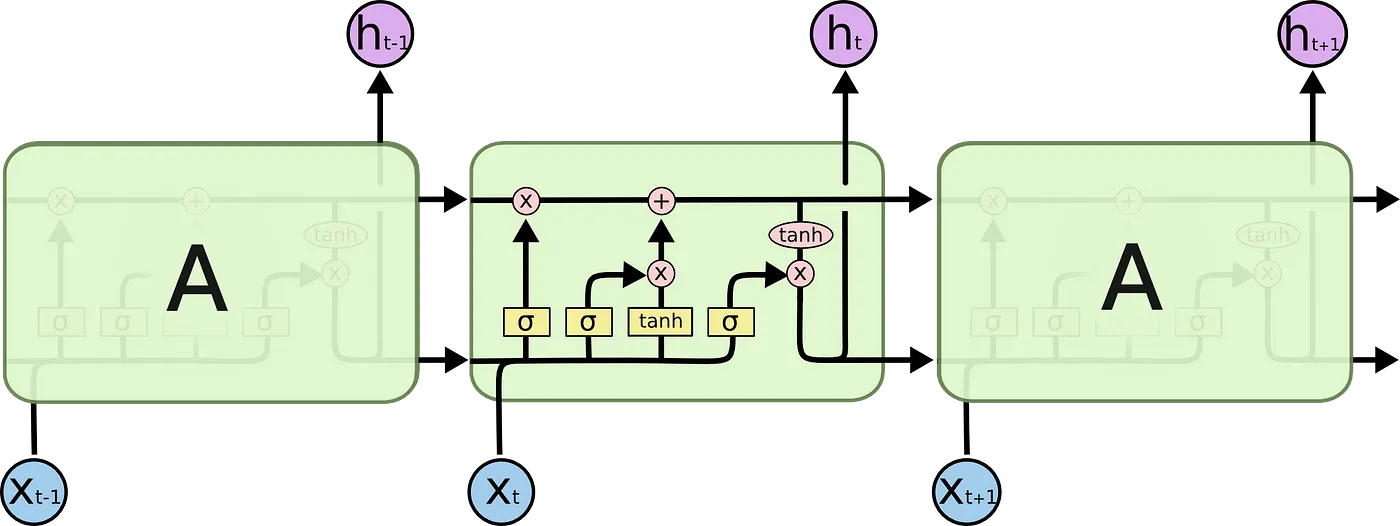
Difference from RNNs:

* RNNs feature a basic structure with a single activation function (usually tanh).



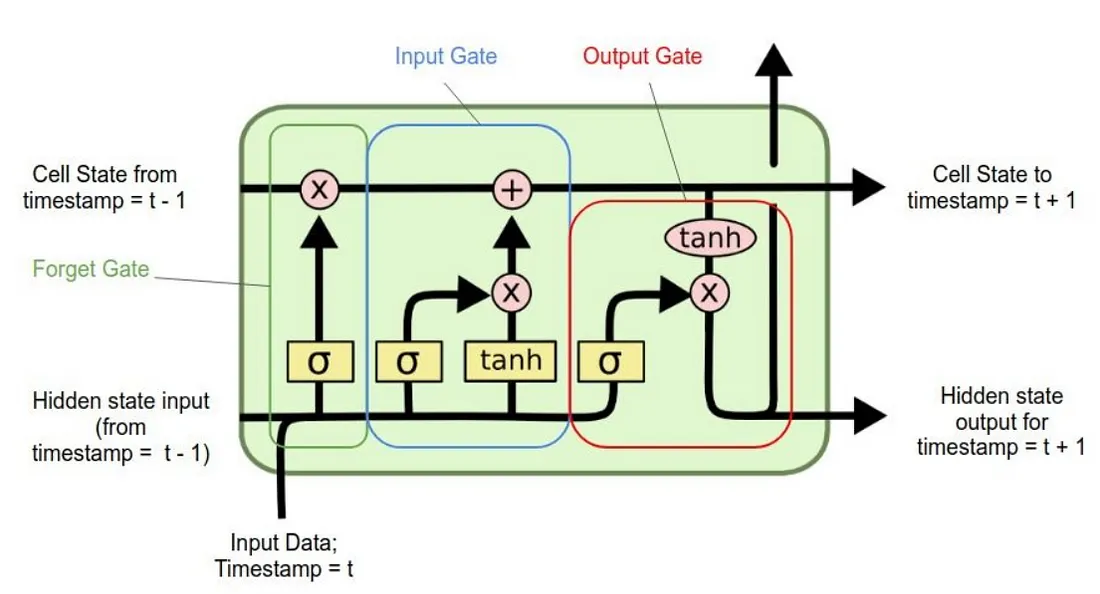
In LSTMs:

* Complex Structure: Instead of a simple network with a single activation function, LSTMs comprise multiple components.
* Enhanced Capabilities: This complexity empowers the network with the ability to both forget and remember information selectively.



LSTMs have 4 different components, namely

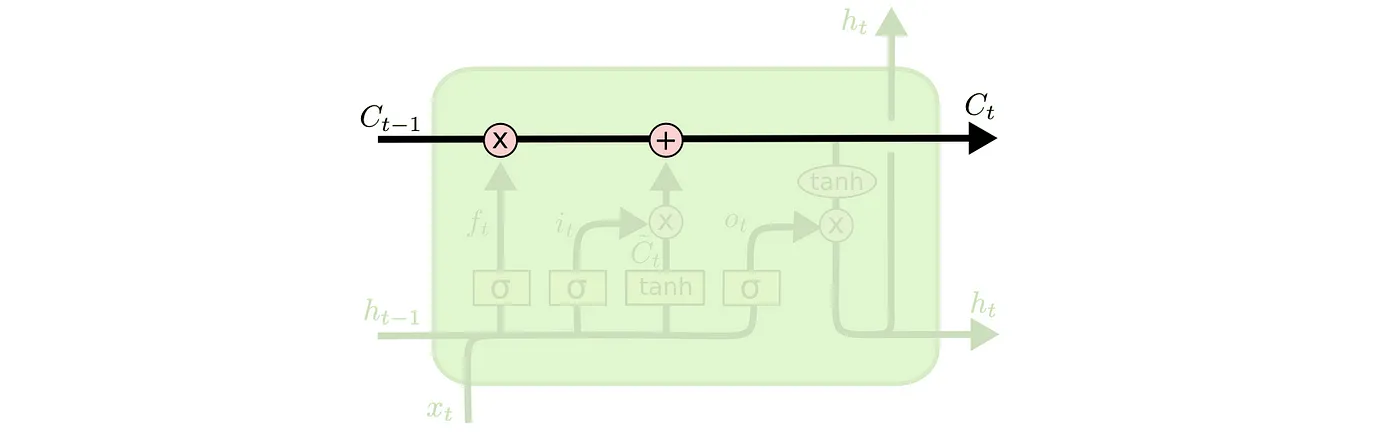
1. Cell state (Memory cell)
2. Forget gate
3. Input gate
4. Output gate



**Components of LSTM:**

1. Cell State (Memory Cell):

* Runs throughout the entire LSTM unit, akin to a conveyor belt.
* Responsible for remembering and forgetting based on the context of the input.
* Implements operations to selectively retain or discard information:
* Forget Gate : Multiplies the cell state by an array of [-1, 0, 1], forgetting information associated with values multiplied by 0.
* Addition (+) :Adds new information to the state, updating the memory accordingly.

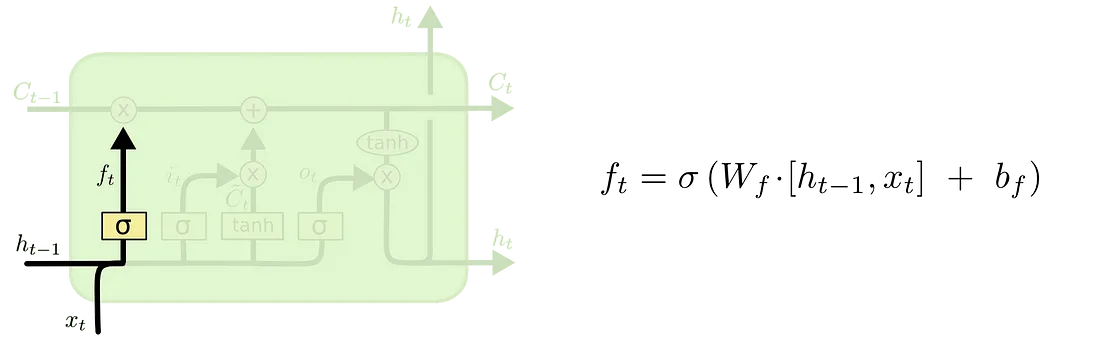


2. Forget Gate:

Function: Determines which information from the previous cell state should be forgotten.

Operation:

* Utilizes a sigmoid layer, known as the "forget gate layer," to make this decision.
* Performs a dot product of *h(t-1)* and *x(t)*, followed by sigmoid activation.
* Outputs values between 0 and 1 for each element in the cell state *C(t-1)*:
* 1 signifies information to be retained.
* 0 indicates information to be completely forgotten.



3. Input Gate:

Function: Provides new information to the LSTM and determines if it should be stored in the cell state.

Components:

1. Sigmoid Layer (Input Gate Layer):

Decides which values to update in the cell state.

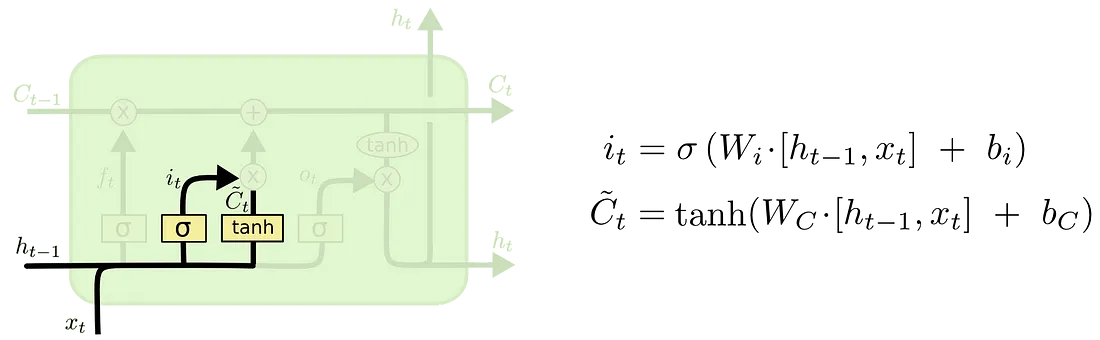
2. Tanh Activation Layer:

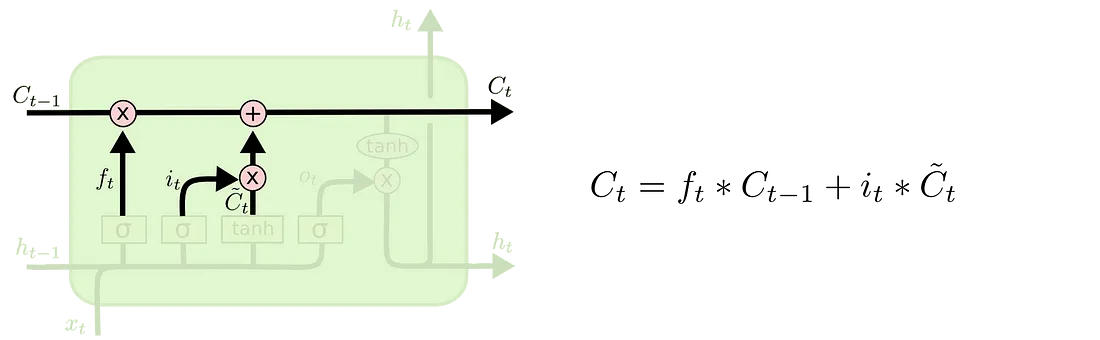
Generates a vector of new candidate values, *Č(t)*, that could be added to the state.

3. Combination and Update:

Multiply the outputs of the sigmoid layer and the tanh layer *i(t) \* Č(t)*.

Update the cell state *C(t)* by adding the output from the forget and input gates.





4. Output Gate:

Function: Determines the output of the LSTM unit based on the new cell state.

Components:

1. Sigmoid Layer:

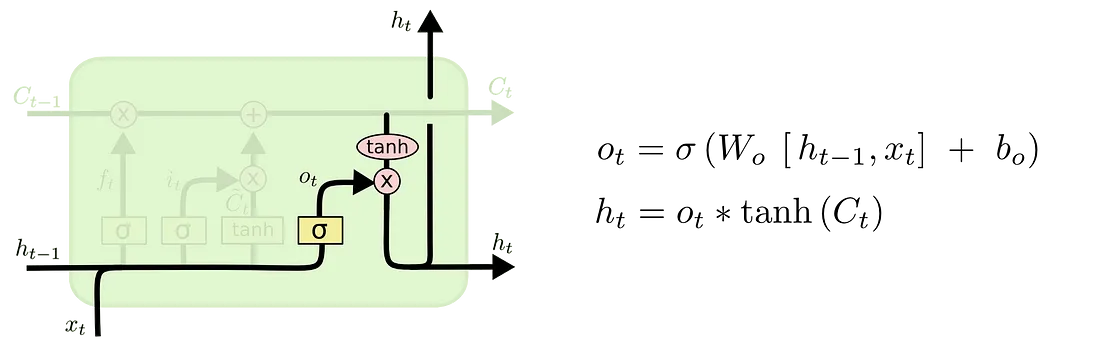
Decides which parts of the cell state to output.

2. Tanh Activation Layer:

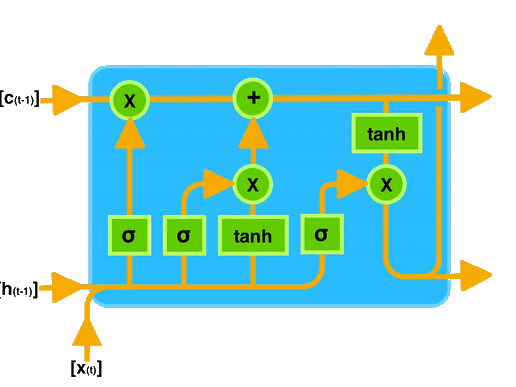
Squashes the values of the cell state between -1 and 1.

3. Combination:

Multiply the output of the tanh layer with the output of the sigmoid gate.



Now that we've grasped the architecture and components of LSTM, let's witness it in action.



**Advantages:**

1. **Handling Long Sequences:** LSTMs are well-suited for processing sequences of data with long-range dependencies. They can capture information from earlier time steps and remember it for a more extended period, making them effective for tasks like natural language processing (NLP) and time series analysis.
2. **Avoiding Vanishing Gradient Problem:** LSTMs address the vanishing gradient problem, which is a common issue in training deep networks, particularly RNNs. The architecture of LSTMs includes gating mechanisms (such as the forget gate) that allow them to control the flow of information and gradients through the network, preventing the gradients from becoming too small during training.
3. **Handling Variable-Length Sequences:** LSTMs can handle variable-length input sequences by dynamically adjusting their internal state. This is useful in many real-world applications where the length of the input data varies.
4. **Memory Cell:** LSTMs have a memory cell that can store and retrieve information over long sequences. This memory cell allows LSTMs to maintain important information while discarding irrelevant information, making them suitable for tasks that involve remembering past context.
5. **Gradient Flow Control:** LSTMs are equipped with mechanisms that allow them to control the flow of gradients during backpropagation. The forget gate, for example, can prevent gradients from vanishing when they need to be propagated back in time. This enables LSTMs to capture information from earlier time steps effectively.

**Disadvantages:**

1. **Computational Complexity:** LSTMs are computationally more intensive compared to other neural network architectures like feedforward networks or simple RNNs. Training LSTMs can be slower and may require more resources.
2. **Overfitting:** Like other deep learning models, LSTMs are susceptible to overfitting when there is insufficient training data. Regularization techniques like dropout can help mitigate this issue.
3. **Hyperparameter Tuning:** LSTMs have several hyperparameters to tune, such as the number of LSTM units, the learning rate, and the sequence length. Finding the right set of hyperparameters for a specific problem can be a challenging and time-consuming process.
4. **Limited Interpretability:** LSTMs are often considered as “black-box” models, making it challenging to interpret how they arrive at a particular decision. This can be a drawback in applications where interpretability is crucial.
5. **Long Training Times:** Training deep LSTM models on large datasets can be time-consuming and may require powerful hardware, such as GPUs or TPUs.

**CODE**

Data Retrieval:

* The script starts by importing necessary libraries including `pandas`, `numpy`, `matplotlib`, `seaborn`, `yfinance`, and `datetime`.
* It defines a function `retrieve\_data(company)` which fetches historical stock price data using the Yahoo Finance API (`yfinance`) from a specified start date (2010-01-01) until the current date. It retrieves only the closing prices and returns a `pandas` DataFrame containing the data.

Data Processing:

* There are two functions defined for data processing: `str\_to\_datetime(s)` and `df\_to\_windowed\_df(dataframe, first\_date\_str, last\_date\_str, n=3)`.
* `str\_to\_datetime(s)` converts a string date (`YYYY-MM-DD`) to a datetime object.
* `df\_to\_windowed\_df(dataframe, first\_date\_str, last\_date\_str, n=3)` takes a DataFrame of stock prices, a start date, an end date, and a window size `n`. It generates a windowed DataFrame where each row contains the closing prices of the previous `n` days (features) and the closing price of the current day (target).

Data Transformation:

* The CODE defines a function `windowed\_df\_to\_date\_X\_y(windowed\_dataframe)` to convert the windowed DataFrame into numpy arrays of dates, features (`X`), and targets (`y`). It reshapes the feature matrix to include a third dimension for compatibility with LSTM.

Model Definition:

* It defines an LSTM neural network model using `Sequential` from TensorFlow Keras. The model architecture consists of an input layer, an LSTM layer with 64 units, two dense layers with ReLU activation functions, and an output layer.

Model Compilation and Training:

* The model is compiled with mean squared error loss, the Adam optimizer with a learning rate of 0.001, and mean absolute error as a metric. Then, it is trained on the training set for 150 epochs.

Model Evaluation:

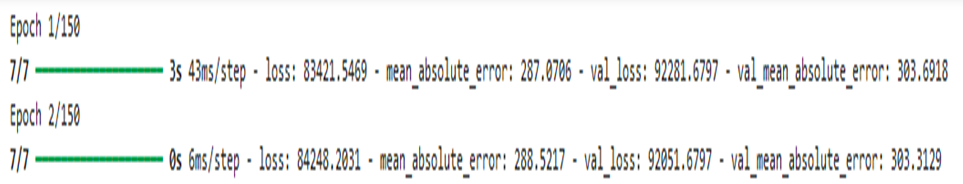
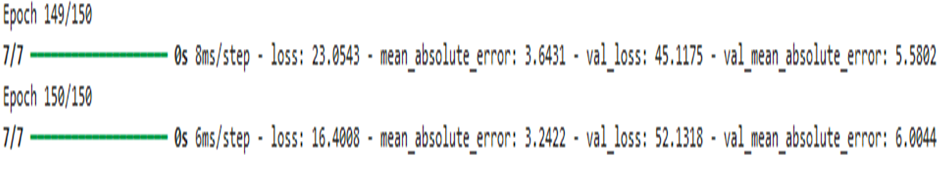
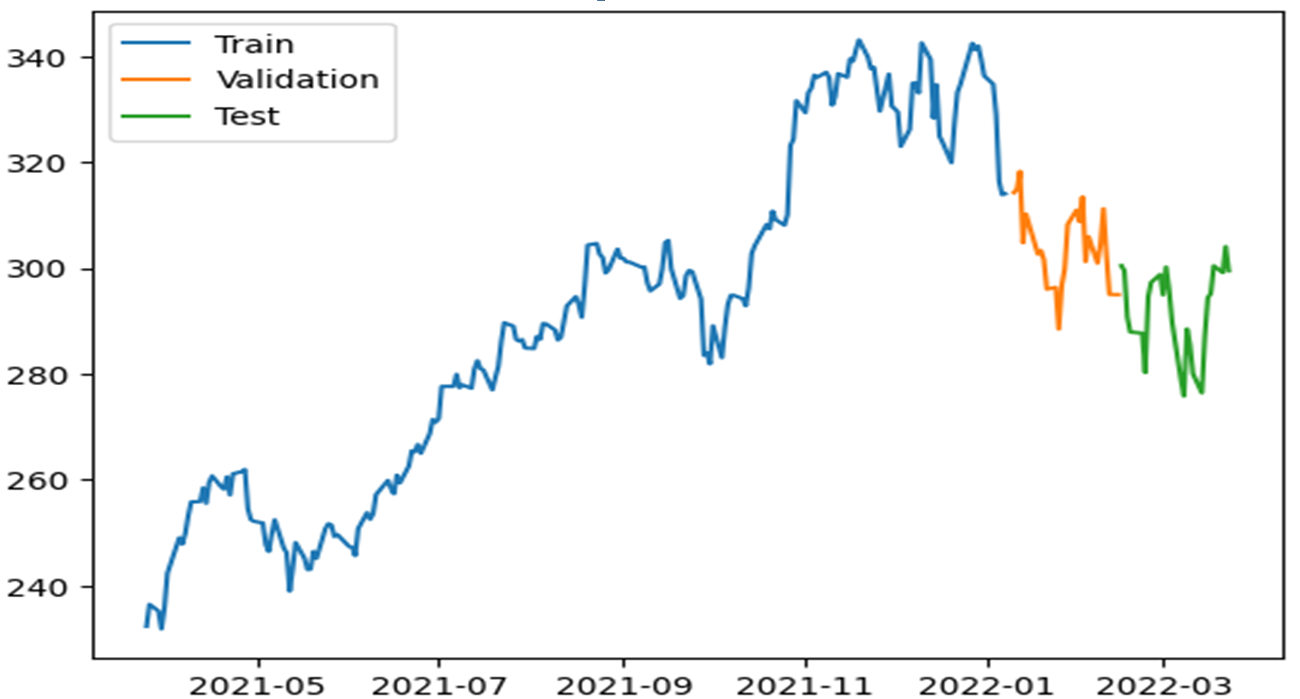
* After training, the model's performance is evaluated on the training, validation, and test sets. Predictions are made on each set and compared with the actual values. Separate plots are generated for each set to visualize the predictions against the actual observations.

Recursive Prediction:

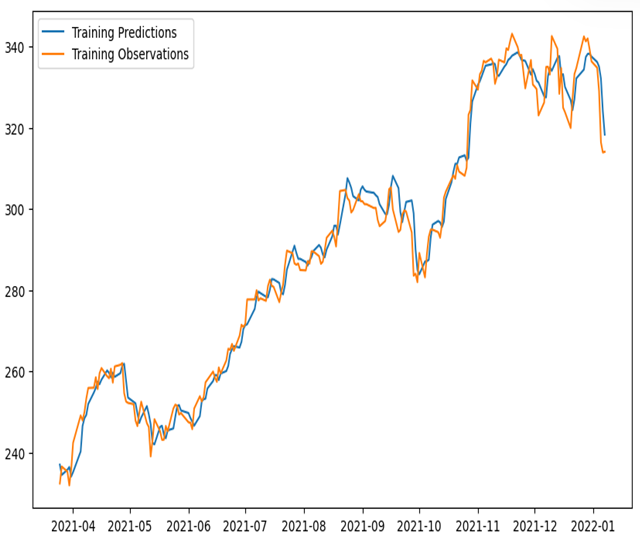
* Finally, the script demonstrates recursive prediction. It iterates over each target date in the validation and test sets, predicts the next value based on the last window of training data, and appends the prediction to a list. This recursive prediction allows for forecasting beyond the test set.

Here, the code provides a structured approach to time series forecasting using LSTM neural networks, from data retrieval to model evaluation and recursive prediction. It's a powerful tool for analyzing and predicting stock prices or any other time series data.

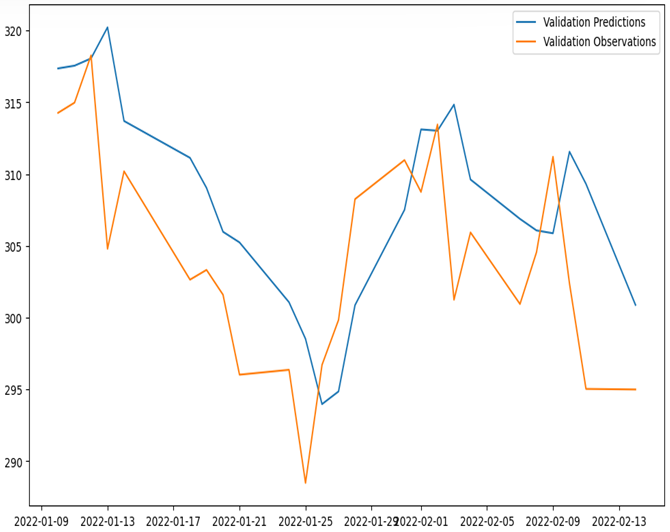
**OUTPUT**

****

**Training**

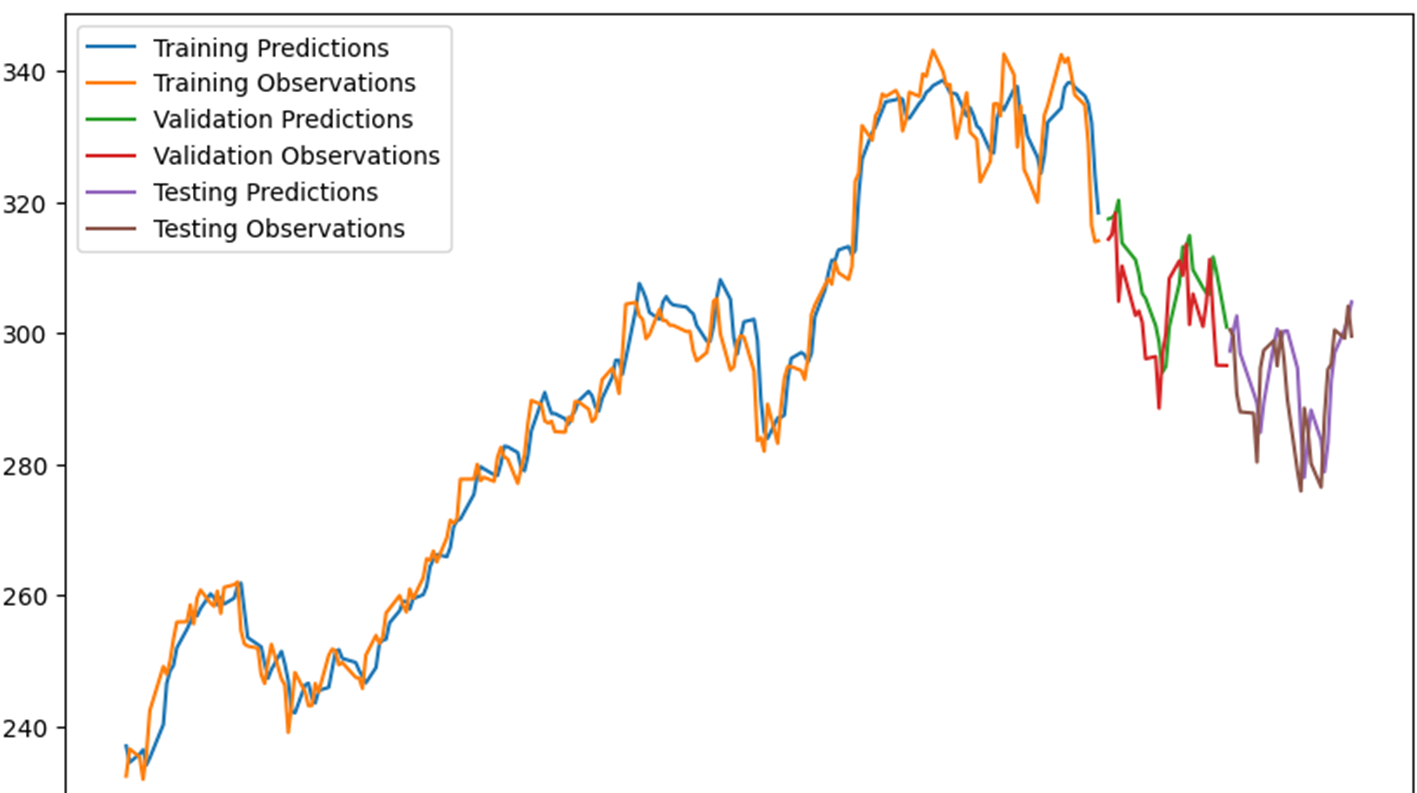
****

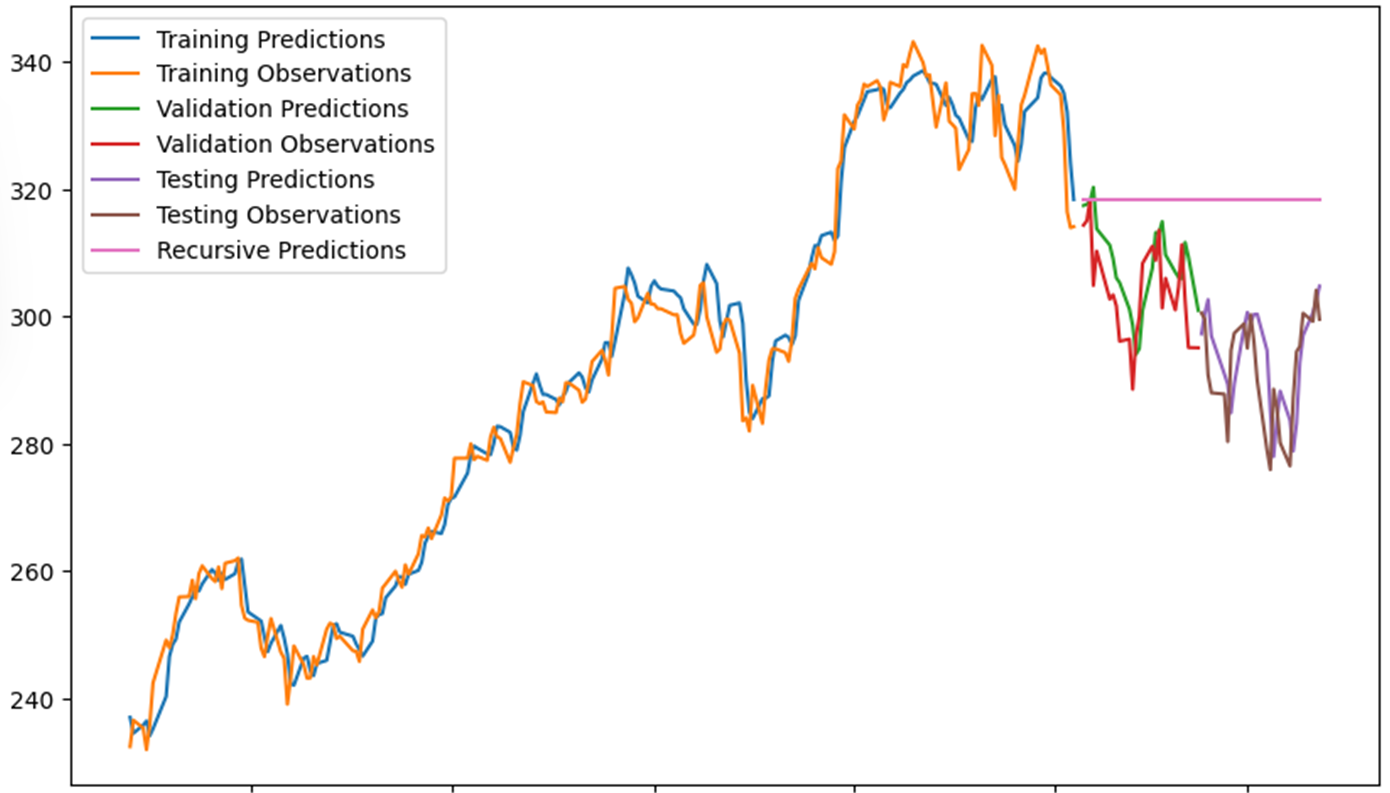
**Validation**

****

**Testing**

****

****

****

**Conclusion:**

* LSTMs excel in retaining information over longer sequences through selective forgetting and remembering.
* The architecture comprises four crucial components: a cell state and three gates (forget, input, and output).
* By addressing the vanishing gradient problem inherent in RNNs, LSTMs demonstrate superiority over vanilla RNNs.
* The functionality and effectiveness of LSTMs lie in their ability to manage and manipulate information flow through the network.
* Through the integration of multiple gates and a memory cell, LSTMs can make informed decisions about which information to retain and which to discard.
* This advanced capability enables LSTMs to perform exceptionally well in tasks involving sequential data processing, such as natural language processing, speech recognition, and time series analysis.
* Understanding the intricacies of LSTM architecture and operation is crucial for effectively leveraging their power in various machine learning applications.

**REFERENCES**

* [**https://medium.com/@prudhviraju.srivatsavaya/lstm-implementation-advantages-and-diadvantages-914a96fa0acb**](https://medium.com/@prudhviraju.srivatsavaya/lstm-implementation-advantages-and-diadvantages-914a96fa0acb)
* <https://github.com/VineetKurapati/Apple-Stock-Price-Prediction-using-Deep-Learning-/blob/main/Apple_Stock_Prediction.ipynb>
* <https://medium.com/analytics-vidhya/introduction-to-long-short-term-memory-lstm-a8052cd0d4cd>