Task 1: Time Series Forecasting

Zeynep Öykü Erdem, 100002305

June 30, 2025

Contents

1 Time Series Forecasting

Time Series Forecasting is a serious of collected, measured data points at evenly spaced time intervals. According to the past observations, we predict the future values.

2 Dataset

Netflix Stock Price Dataset, consists of 6 columns , as follows Date, Open, High,Low, Close, Adj Close and Volume. Open column specifies the opening value of the stock current day, High and Low columns specify the high and low value of the stock in the current day, Close column specifies the closed value ,and Adj Close column adjusts the Close price for corporate actions such as :

- Stock splits
- Dividends
- Rights offerings

Volume column expresses the number of shares traded.

Dataset consists of 1009 entries : In this project, the aim is to find the adjusted close value

#	Column	Dtype
0	Date	object
1	Open	float64
2	High	float64
3	Low	float64
4	Close	float64
5	Adj Close	float64
6	Volume	int64

Table 1: Data Types

the next day by using a sequence of 60. First from kagglehub I downloaded the csv, then by using pandas library I read the csv file and stored the data in data variable.

Because we are using a large scale of values in different features, I used MinMaxScaler to normalize the data in order to obtain stable range of values,better performance, less memory usage.

```
def create_sequences(features,target,sequence_length):
  X,y=[],[]
  for i in range(len(features) - sequence_length-1):
        X.append(features[i:i+sequence_length])
        y.append(target[i+sequence_length])
  X,y=np.array(X),np.array(y)
  return X,y
```

The create_sequences function takes 2 parameters in order to prepare sequential data for the models LSTM, GRU and bidirectional LSTM , which work with sequences /time series. Creating many small sequences of length sequence_length from my data. For each sequence, I assigned the next target value as the label. This approach is to predict the next value given previous time steps.

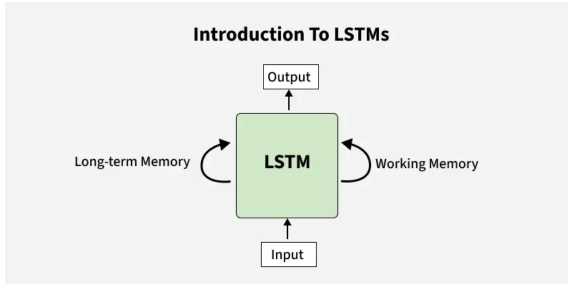

Figure 1: The LSTM Architecture

3 Long Term Dependency Problem

In formal Recurrent Neural Network, long term dependency problem refers to their difficulty in learning and remembering information from many time steps earlier. During training with backpropogation through time, the gradients used to update the RNN's weights tend to vanish (becomes closer to 0 which makes it difficult to determine the direction of gradient descent) or explode (become very large) which makes the learning rate becomes unstable. In order to overcome these problems, specialized architectures like LSTM, GRU and bidirectional LSTM are used to better preserve and regulate information over long sequences.

4 LSTM

Long Short Term Memory Model is enhanced version of the Recurrent Neural Network, which introduce a memory cell that holds information over extended periods addressing the challenge of the long term dependencies where RNN has difficulties to preserve and regulate information over long sequences. LSTM Architectures involves the memory cell which is controlled by three gates:

- **Input Gate**, controls which information is retrained to memory cell
- **Forget Gate** choose what information to remove from the memory cell
- **Output Gate** decides what information to pass on from the memory cell.

sigmoid activation function forces the input between 0 and 1, while the tanh activation function is a 0-centered function that forces the values between -1 and 1, preventing the problem of disappearance of the gradient. In my Dataset, because the output will be regression not classification , so that the sigmoid function will be not efficient. tanh activation function is default function for GRU and LSTM.

5 GRU

The main idea of GRU is to use gate mechanisms to update the hidden state selectively, each time step while retraining important information while discarding irrelevant information. Compared to

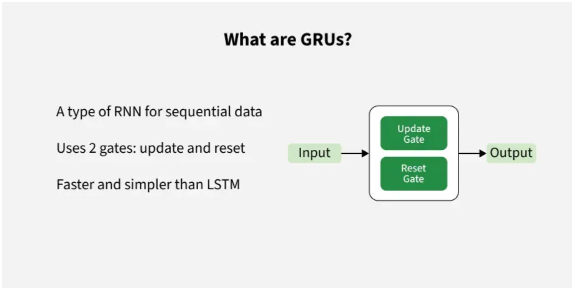

Figure 2: The GRU Architecture

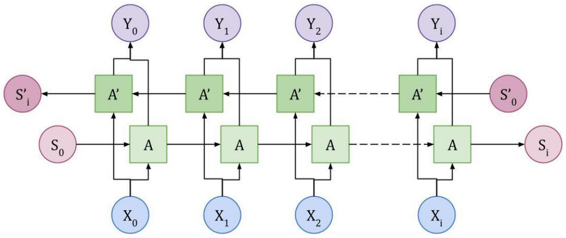

Figure 3: The Bidirectional LSTM Architecture

LSTM, GRU is simpler, faster to train, and performs better especially with shorter sequences. The GRU consists of two main gates:

- **Update Gate**, decides which information form previous hidden state is relevant for the next time step
- **Reset Gate** determines how much should be forgotten of the past hidden state

6 Bidirectional LSTM

Bidirectional Long Short-Term Memory is an extension of traditional LSTM model, which allows two way direction process, enables to capture more contextual information . Forward LSTM captures the previous context information, while Backward LSTM captures future context information. The output of both LSTM's then combined to final output.

7 Evaluation

As we can observe from the graphs, Bidirectional LSTM stands out with the lowest overall loss and the validation loss value through the each epochs. Compared with GRU and LSTM, GRU works better due to its efficiency in computing, better performance on small sequences. The most important reason why Bidirectional LSTM gives a better performance is the two way direction process, which enables to process future and previous values in order to decide which information should pass on, which information should be forgotten or which information should be added. It summarizes more general idea when comparing other models.

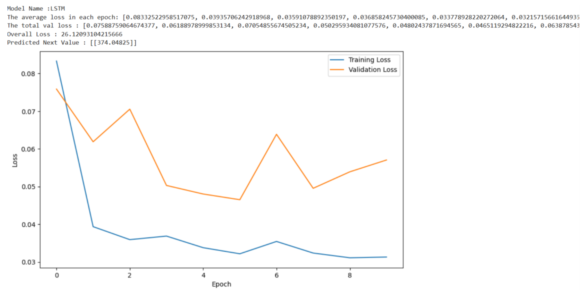

Figure 4: Training Loss and Validation Loss of LSTM

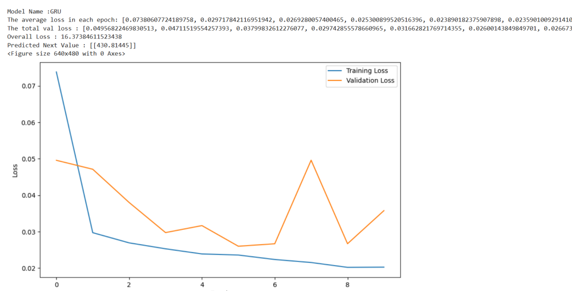

Figure 5: Training Loss and Validation Loss of GRU

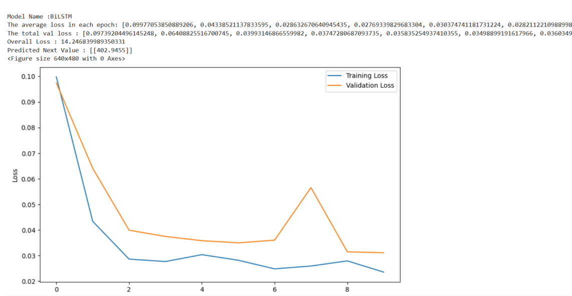

Figure 6: Training Loss and Validation Loss of Bidirectional LSTM

8 References

- Lecture Note: http://gnjatovic.info/machinelearning/
- Dataset: https://www.kaggle.com/datasets/jainilcoder/netflix-stock-price-prediction/data
- Code Snippets: http://gnjatovic.info/machinelearning/
- LSTM Architecture image: https://www.geeksforgeeks.org/deep-learning-introduction-to-long-short-term-memory/
- GRU Architecture image: https://www.geeksforgeeks.org/machine-learning/gated-recurrent-unit-networks/
- Bidirectional LSTM Architecture image: https://www.geeksforgeeks.org/nlp/bidirectional-lstm-in-nlp/