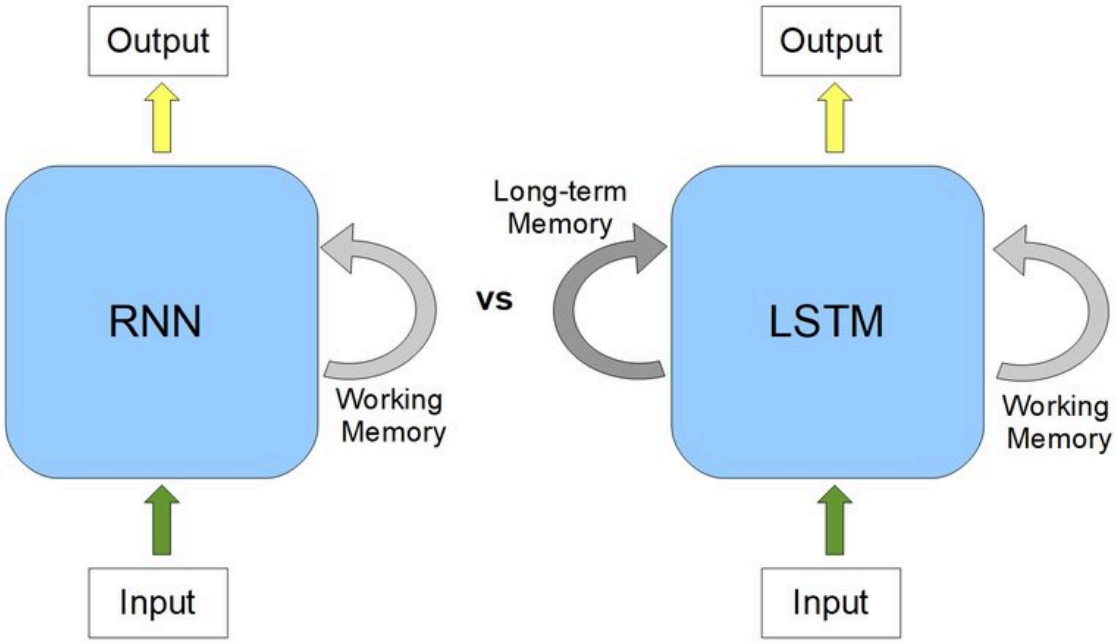
A Comprehensive Review of Recurrent Neural Networks via LSTM on Stock Analysis

Abstract:

The objective of the report is to offer a comprehensive introduction to the fundamental concept of recurrent neural networks and the limitations they face in capturing long-range dependencies. It then dives into the LSTM working on the concept of stock prediction and analysis via RNN usage.
The report highlights the advantages of LSTM over traditional RNNs, emphasizing its ability to model and predict complex sequential patterns. Various applications of LSTM are explored, including natural language processing, speech recognition, time series analysis, and sentiment analysis. The report showcases real-world use cases in each domain, demonstrating LSTM's efficacy in tasks like stock market prediction.

Introduction:

Recurrent Neural Networks (RNN) and their variant, Long Short-Term Memory (LSTM), are powerful models designed to handle sequential data. RNNs process sequential information by maintaining internal states, but face challenges in capturing long-term dependencies. LSTM addresses these challenges by incorporating memory cells and specialized gating mechanisms. The input gate determines what information to store, the forget gate regulates what to discard, and the memory cell retains sequential information. The output gate controls the output from the memory cell. LSTM's ability to capture long-term dependencies has made it invaluable in tasks such as language modeling, speech recognition, sentiment analysis, and time series forecasting. By introducing memory and gating mechanisms, LSTM significantly enhances the modeling of sequential data, surpassing traditional RNNs.
The aim of the LSTM model used for stock prediction is to leverage the sequential nature of historical stock data to accurately forecast future stock prices. By training the LSTM model on historical stock prices, volume, and other relevant features, the objective is to capture complex patterns and dependencies in the data that can aid in predicting future stock movements. The model aims to provide valuable insights to investors and traders by generating reliable predictions, enabling them to make informed decisions regarding buying, selling, or holding stocks.



Problem Statement:

The problem statement in our case is to predict stock with minimal errors and to develop an optimized forecasting model that effectively reduces prediction errors in stock price forecasting. While LSTM models have shown promise in capturing complex patterns, the challenge lies in minimizing the errors and uncertainties associated with stock predictions.
The problem statement is to develop a stock prediction model that utilizes LSTM and time series analysis techniques to compare the model's stock prediction with the actual stock prices.

Objective:

Time series analysis is a field that focuses on analyzing and forecasting data points collected over time. By leveraging LSTM for time series analysis, we can harness the power of its memory cells and gating mechanisms to effectively capture temporal patterns, trends, and relationships in the data.
The precise goals are as follows:

- The objective is to achieve a model that provides more precise and reliable stock price predictions, minimizing the impact of errors and enabling investors to make more informed decisions in the dynamic and volatile stock market environment.
- Train an LSTM-based model using historical stock data to predict future stock prices.
- Apply time series analysis techniques to identify and capture relevant patterns, trends, and dependencies in the stock market data.
- Optimize the model to achieve high accuracy in predicting stock prices and minimize prediction errors, enabling informed decision-making for investors and traders.

Methodology:

RNN as a neural network works well on a time variant type of problems. But due the inclusion of problems like Vanishing Gradient and Exploding Gradient it affects the models overall accuracy and predictive capability. Hence we use LSTM to tackle this problem.

Data Preprocessing:
Collect historical stock data, including prices, volumes, and any relevant features.Since we are using a stock analysis we have used the dataset of “Tata Motors” as the main data. Clean the data by handling missing values, outliers, and noise. Perform necessary transformations such as normalization or scaling to ensure consistent data representation.

These are done in the following steps like:

- Importing libraries like keras,numpy,sklearn.pandas and tensorflow
- Cleaning data by detecting noise.
- Normalization of the data and Transformation

Feature Selection and Engineering:
Identify relevant features that could contribute to predicting stock prices such as technical indicators and market indices. We are using features like the “Open”,“Close”, “Adj close” and “Volume” to go through this methodology.

Splitting the Data:
Divide the dataset into training and testing sets. Since stock market data is time-dependent, it is crucial to preserve the temporal order. Typically, a portion of the latest data is reserved for testing, while the remaining data is used for training.Since most stock dataset come in a singular sheet we have to manually split it into testing and training sets.

```
timesplit= TimeSeriesSplit(n_splits=10)
for train_index, test_index in timesplit.split(feature_transform):
        X_train, X_test = feature_transform[:len(train_index)], feature_transform[len(train_index): (len(train_index)+len(test_in
        y_train, y_test = output_var[:len(train_index)].values.ravel(), output_var[len(train_index): (len(train_index)+len(test_i
```

```
trainX =np.array(X_train)
testX =np.array(X_test)
X_train = trainX.reshape(X_train.shape[0], 1, X_train.shape[1])
X_test = testX.reshape(X_test.shape[0], 1, X_test.shape[1])
```

Splitting data into train and test

Building the LSTM Model:
We then have to design the architecture of the LSTM model, by defining the number of LSTM layers, hidden units, and input or output dimensions.Configure the model with appropriate activation functions, loss functions, and optimization algorithms.
comparison:

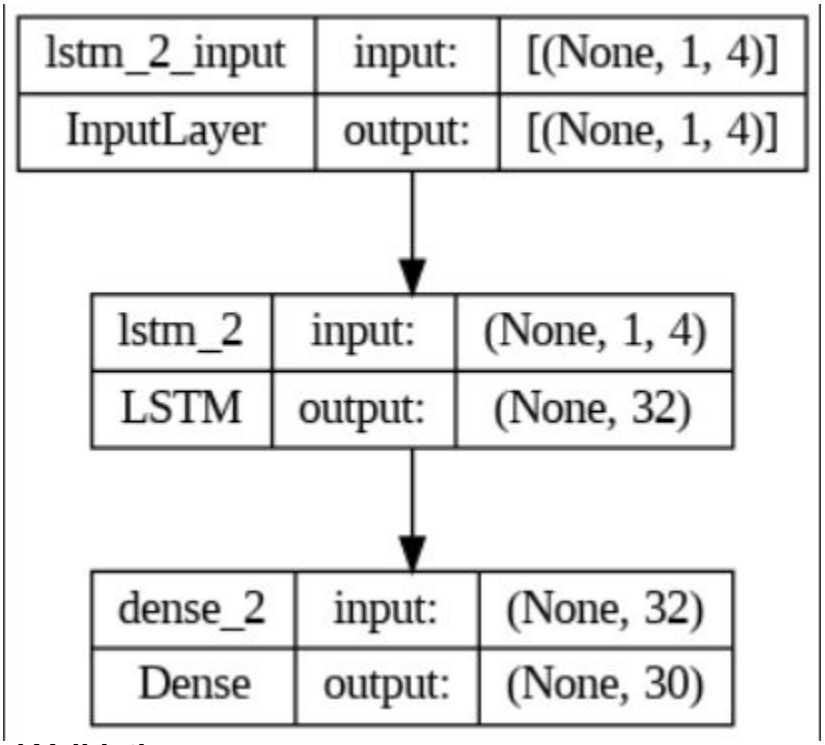
```
lstm = Sequential()
lstm.add(LSTM(32, input_shape=(1, trainX.shape[1]), activation='relu', return_sequences=False))
lstm.add(Dense(1))
lstm.compile(loss='mean_squared_error', optimizer='adam')
plot_model(lstm, show_shapes=True, show_layer_names=True)
```

Figure 1: A singular neural network with only one layer

```
lstm = Sequential()
lstm.add(LSTM(32, input_shape=(1, trainX.shape[1]), activation='relu', return_sequences=False))
lstm.add(Dense(8))
lstm.compile(loss='mean_squared_error', optimizer='adam')
plot_model(lstm, show_shapes=True, show_layer_names=True)
```

Figure 2: Neural Network with increase in nodes

Using lst.add() we can add layer to our neural network inside the add() parameters like type of neural network (LSTM,RNN,Dense,CNN and etc),Activation functions(Step,Sigmoid,tanh,relu and etc),loss functions and size are specified.



Training and Validation:
Train the LSTM model on the training data. Monitor the training process using validation data to prevent overfitting. Experiment with different hyperparameters, such as learning rate, batch size, and number of epochs, to optimize model performance.

Basically saying we evaluate the model in a more optimized way until we get a more desired output.

- Changes in Hyperparameter:
  We can change the hyperparameter size for example we can change from using lstm.add(LSTM(64, activation='relu')) to lstm.add(LSTM(32, activation='relu'))

- Changes of epoch:
  Epoch refers to amount of iteration through which the model goes through.

Example:

history=lstm.fit(X_train, y_train, epochs=100, batch_size=8, verbose=1, shuffle=False)
history=lstm.fit(X_train, y_train, epochs=500, batch_size=8, verbose=1, shuffle=False)

history=lstm.fit(X_train, y_train, epochs=1000, batch_size=8, verbose=1, shuffle=False)

Model Evaluation:
Evaluate the trained LSTM model on the testing data by making predictions of future stock prices. Calculate performance metrics such as mean squared error (MSE), root mean squared error (RMSE), or mean absolute error (MAE) to assess the accuracy and robustness of the model's predictions.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(O_i - F_i)^2}{n}} \qquad MSE = \frac{\sum(y_i - \hat{y}_i)^2}{n}$$

We can evaluate our model by also comparing it to the other models that works on similar models like regression etc.

These can be done by comparing the different models' performance metrics such as mean squared error (MSE), root mean squared error (RMSE), or mean absolute error (MAE) and R-square values.

LSTM:
Mean Squared Error (MSE): 13.951345732437709
Root Mean Squared Error (RMSE): 3.735150028102982
R-squared (R2): 0.9346254249277153

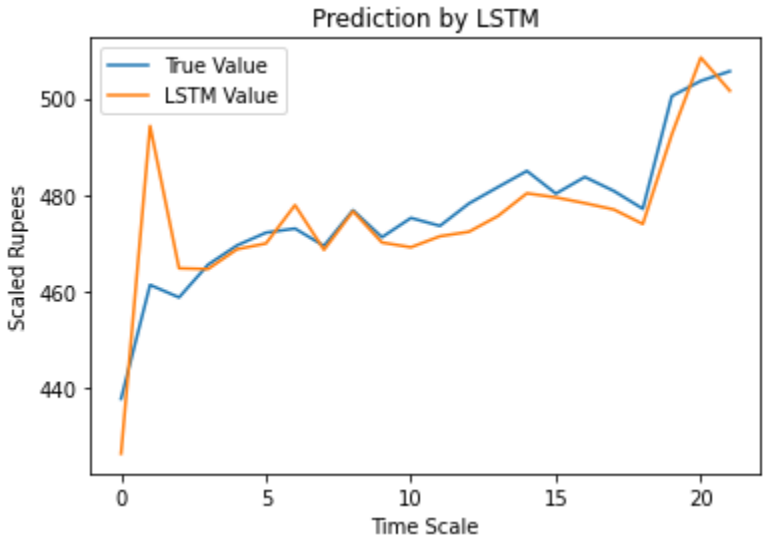
Regression:
Mean Squared Error (MSE): 8.822422337659232
Root Mean Squared Error (RMSE): 2.9702562747445262
R-squared (R2): 0.9586590338671273

The R2 value represents the proportion of the variance in the target variable that is predictable from the input variables. In this case, the LSTM model has a higher R2

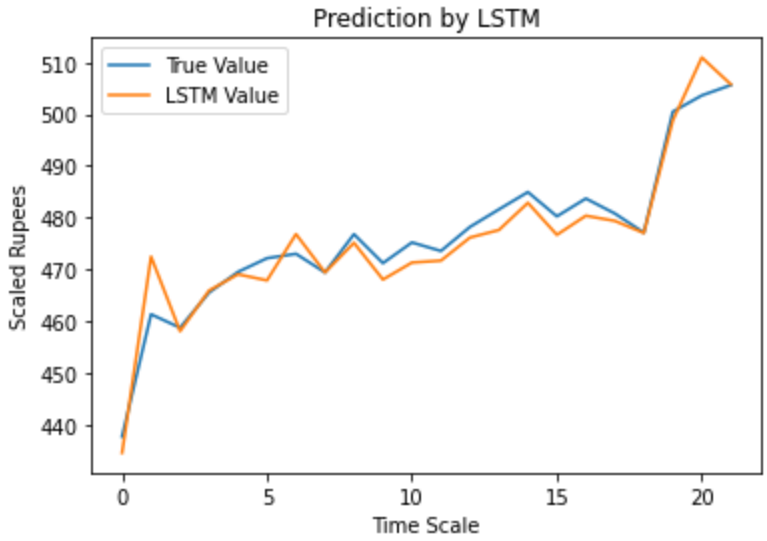
value compared to the regression model, indicating that it explains a higher proportion of the variance in the target variable.

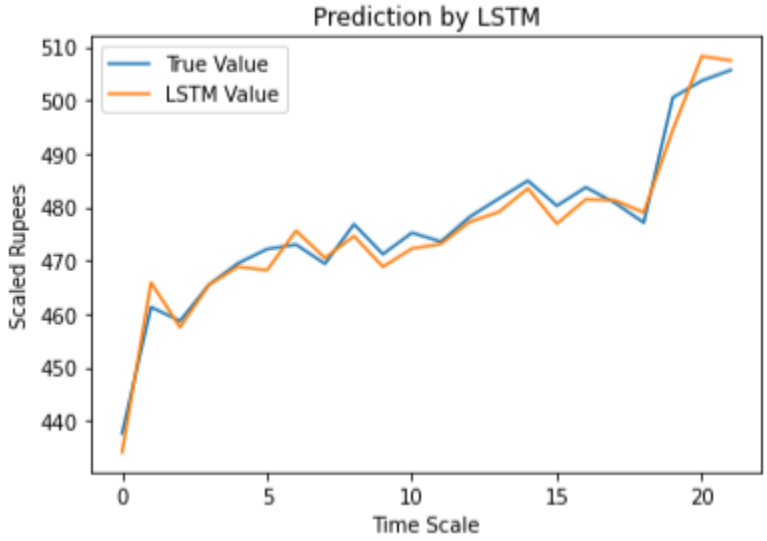
Model Refinement and Iteration:
Analyze the model's performance, identify any shortcomings, and refine the LSTM model accordingly. This may involve adjusting hyperparameters, modifying the model architecture, or incorporating additional features or data sources.

Prediction and Decision Making:
Once satisfied with the model's performance, utilize the trained LSTM model to make predictions on unseen or future stock data. These predictions can assist investors and traders in making informed decisions regarding buying, selling, or holding stocks based on the projected price movements.
We can see various adjustment wise



\\





As we can see from the graphs when we adjust the parameters accordingly to train and fit the model it reduces its error and increases its accuracy.

Limitations:

1. Lack of feature engineering: The model uses only the raw numerical features ('Open', 'High', 'Low', 'Volume') without any additional engineered features. Feature engineering can help capture more meaningful patterns and relationships in the data, potentially improving the model's performance.

2. Limited input features: The model uses a limited number of input features, which may not capture all the relevant information for predicting stock prices accurately. Including additional relevant features, such as technical indicators, market sentiment, or external factors, could potentially enhance the model's performance.

3. Sensitivity to data quality and assumptions: The model's performance heavily relies on the quality of the input data and the assumptions made during the modeling process. Any issues with data quality, such as missing values or outliers, can negatively impact the model's performance.

4. Market unpredictability: Predicting stock prices is inherently challenging due to the unpredictable nature of financial markets. The model's performance may be affected by sudden market shifts, changes in investor sentiment, or unforeseen events that cannot be captured solely by historical data.

Conclusion:

The goal of the model was to predict stock prices using a Long Short-Term Memory (LSTM) neural network. It trained and evaluated the model on historical stock data for TATAMOTORS, aiming to provide insights into future price movements.This model works for any type of dataset of stock given regardless of the country,denomination etc.

Upon analyzing the results, it is evident that the LSTM model shows promise in predicting stock prices. The model's performance, as measured by Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (R2), indicates its ability to capture patterns and trends in the data.

In conclusion, the LSTM model demonstrates potential for predicting stock prices. It can provide valuable insights into future price movements, allowing investors and analysts to make informed decisions. However, it is crucial to continuously refine and improve the model by exploring advanced techniques, incorporating additional features, optimizing hyperparameters, and validating its performance against other models or baselines. It is also important to consider the dynamic and unpredictable nature of financial markets when utilizing the model's predictions.

Reference:

1. Parking Occupancy Prediction Method Based on Multi Factors and Stacked GRU-LSTM CHAO ZENG, CHANGXI MA , KE WANG , AND ZIHAO CUI

2. Research on throughput prediction of 5G network based on LSTM Lanlan Li* and Tao Ye

3. Predicting Chinese Commodity Futures Price: An EEMD-Hurst-LSTM Hybrid Approach HUANG KE1,2, ZHANG ZUOMINYANG2,3, LI QIUMEI 4 , AND LUO YIN

4. BECT Spike Detection Based on Novel EEG Sequence Features and LSTM Algorithms Zhendi Xu, Tianlei Wang , Jiuwen Cao , Senior Member, IEEE, Zihang Bao , Tiejia Jiang, and Feng Gao

5. Recurrent Neural Networks: An Embedded Computing Perspective NESMA M. REZK 1 , MADHURA PURNAPRAJNA 2 , TOMAS NORDSTRÖM 3 , AND ZAIN UL-ABDIN 1

6. Parking Occupancy Prediction Method Based on Multi Factors and Stacked GRU-LSTM CHAO ZENG1,2, CHANGXI MA 3 , KE WANG 3 , AND ZIHAO CUI 1

7. A LSTM Based Model for Personalized Context-Aware Citation Recommendation LIBIN YANG 1 , (Member, IEEE), YU ZHENG2 , XIAOYAN CAI 1 , HANG DAI1 , DEJUN MU 1 , LANTIAN GUO 1 , AND TAO DAI

8. Enhancements of Attention-Based Bidirectional LSTM for Hybrid Automatic Text Summarization JIAWEN JIANG 1 , HAIYANG ZHANG 2 , CHENXU DAI1 , QINGJUAN ZHAO3 , HAO FENG1 , ZHANLIN JI 1,4, (Member, IEEE), AND IVAN GANCHEV 5,6,4, (Senior Member, IEEE)

9. An Extensible Framework for Short-Term Holiday Load Forecasting Combining Dynamic Time Warping and LSTM Network JEFFREY GUNAWAN AND CHIN-YA HUANG , (Member, IEEE)

10. Detecting Dengue/Flu Infections Based on Tweets Using LSTM and Word Embedding SAMINA AMIN, M. IRFAN UDDIN M. ALI ZEB1 , ALA ABDULSALAM ALAROOD 2 , MARWAN MAHMOUD3 , AND MONAGI H. ALKINANI 4