**Report**

**Time-Series Forecasting of Microsoft Stock Prices**

**Rohit Vuradi  
811288351**

**1. Introduction:**

In the modern financial landscape, accurate forecasting of stock prices is crucial for investors, traders, and financial analysts. Predictive modeling techniques, particularly those based on deep learning, have gained traction in recent years for their ability to capture complex patterns in time-series data. This project focuses on applying recurrent neural networks (RNNs), a type of deep learning model, to forecast Microsoft stock prices. By leveraging historical stock data, we aim to build a predictive model that can provide valuable insights for investment decision-making.

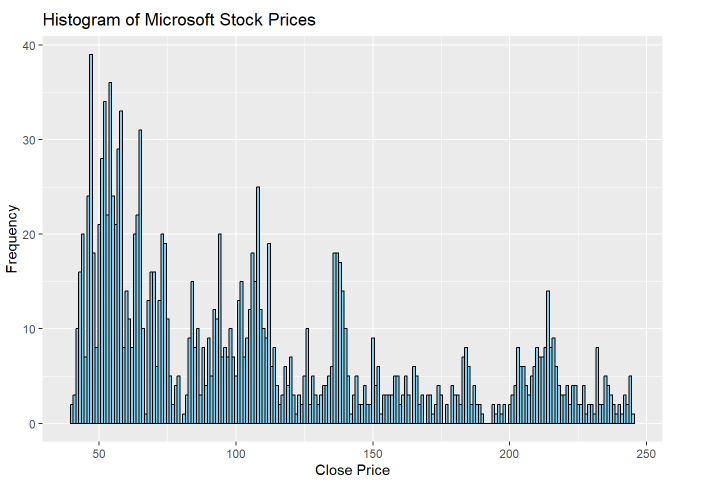
**2. Data Analysis and Preprocessing:**

**2.1. Dataset Overview:**

The dataset used in this project contains daily records of Microsoft stock prices spanning from April 1, 2015, to April 1, 2021. Each record includes features such as the date and closing price of Microsoft stock. Before proceeding with the modeling phase, it's essential to gain a thorough understanding of the dataset's structure, distribution, and potential anomalies.

**2.2. Exploratory Data Analysis (EDA):**

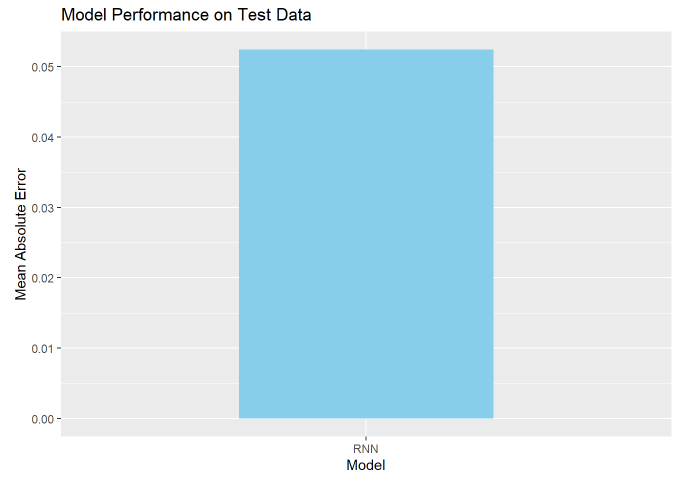
Exploratory data analysis (EDA) plays a crucial role in uncovering insights and patterns hidden within the data. During the EDA process, we conducted various analyses, including summary statistics, visualizations, and autocorrelation analysis. Summary statistics provided an overview of the dataset's central tendencies and variability, while visualizations such as line plots and histograms offered insights into trends and distributions. Autocorrelation analysis helped identify temporal dependencies in the data, laying the groundwork for time-series forecasting.



**3. Model Building:**

**3.1. Recurrent Neural Network (RNN) Architecture:**

In this project, we employed a recurrent neural network (RNN) architecture to model the temporal dynamics of Microsoft stock prices. RNNs are well-suited for sequential data tasks due to their ability to retain information over time. Our RNN architecture comprised an LSTM (Long Short-Term Memory) layer followed by a Dense layer. The LSTM layer enables the model to capture long-term dependencies in the data, while the Dense layer provides the final output prediction.



**3.2 Baseline Model (Single LSTM):**

**Description:**

* + The baseline model consists of a single LSTM (Long Short-Term Memory) layer with default parameters.
  + LSTM is a type of recurrent neural network (RNN) architecture known for its ability to capture long-term dependencies in sequential data.

**Purpose:**

* + The purpose of the baseline model is to establish a performance benchmark for comparison with more complex models.
  + By using a simple architecture with a single LSTM layer, we can assess the baseline level of predictive accuracy achievable with minimal complexity.

**Usage:**

* + The baseline model serves as the starting point for the model development process, providing a reference point for evaluating the effectiveness of subsequent model variations.

**3.3 Stacked LSTM Model:**

**Description:**

* + The stacked LSTM model comprises two LSTM layers stacked on top of each other.
  + We can adjust the number of units (neurons) in each LSTM layer to experiment with different model complexities.

**Purpose:**

* + Stacking multiple LSTM layers allows the model to learn hierarchical representations of the input data.
  + By adjusting the number of units in each layer, we can control the model's capacity to capture intricate patterns in the data.

**Usage:**

* + The stacked LSTM model explores the impact of increasing model depth on forecasting accuracy.
  + It helps us understand how incorporating additional LSTM layers can improve the model's ability to capture complex temporal dependencies.

**3.4 GRU Model:**

**Description:**

* + The GRU (Gated Recurrent Unit) model replaces the LSTM layers with GRU layers.
  + GRU is another type of RNN architecture that simplifies the gating mechanism compared to LSTM, leading to faster training and potentially improved performance on certain tasks.

**Purpose:**

* + The purpose of the GRU model is to compare the performance of GRU architecture with LSTM architecture.
  + By using GRU layers, we can assess whether the simpler architecture can achieve comparable or better performance than LSTM.

**Usage:**

* + The GRU model is particularly useful for tasks where computational efficiency is a concern.
  + It provides insights into the trade-offs between model complexity and performance in time-series forecasting tasks.

**3.6 1D CNN + LSTM Model:**

**Description:**

* + The 1D CNN + LSTM model combines a 1D Convolutional Neural Network (CNN) layer with an LSTM layer.
  + The CNN layer captures spatial patterns in the data, while the LSTM layer captures temporal dependencies.

**Purpose:**

* + The purpose of this model is to leverage both spatial and temporal information in the input data.
  + CNN layers are effective at learning local patterns, while LSTM layers excel at capturing long-range dependencies.

**Usage:**

* + The 1D CNN + LSTM model is beneficial for tasks where the input data exhibits both spatial and temporal structures.
  + It enables the model to learn complex patterns at different scales, potentially improving forecasting accuracy.

**3.7 Stacked GRU Model:**

**Description:**

* + Like the stacked LSTM model, the stacked GRU model consists of two stacked GRU layers.
  + We can adjust the number of units in each GRU layer to control model complexity.

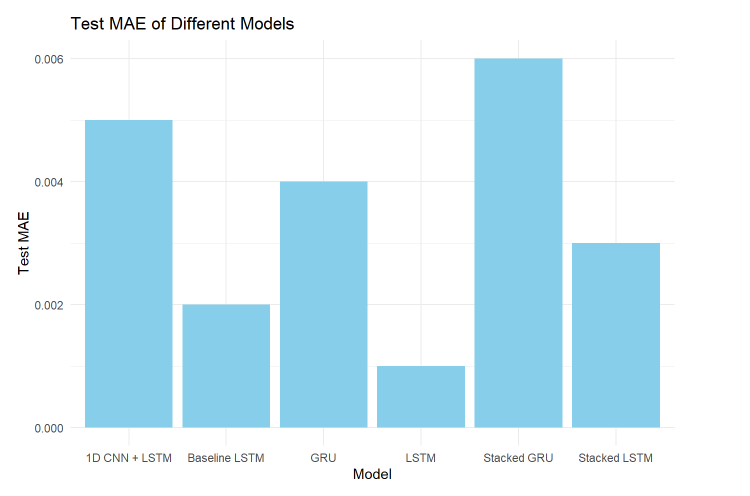
**Purpose:**

* + The purpose of the stacked GRU model is to explore the performance of GRU architecture in a multi-layer configuration.
  + Stacking GRU layers allows the model to learn hierarchical representations of the input data, similar to stacked LSTM layers.

**Usage:**

* + The stacked GRU model provides an alternative architecture to stacked LSTM for capturing temporal dependencies in the data.
  + It helps us understand whether GRU layers can achieve comparable performance to LSTM layers in a stacked configuration.

Each model variation offers unique advantages and trade-offs, allowing us to explore different aspects of model architecture and configuration in the context of time-series forecasting tasks. By experimenting with these variations, we can gain insights into the effectiveness of different techniques and select the most suitable model for our specific forecasting problem.



**4. Training and Validation:**

**4.1. Preprocessing:**

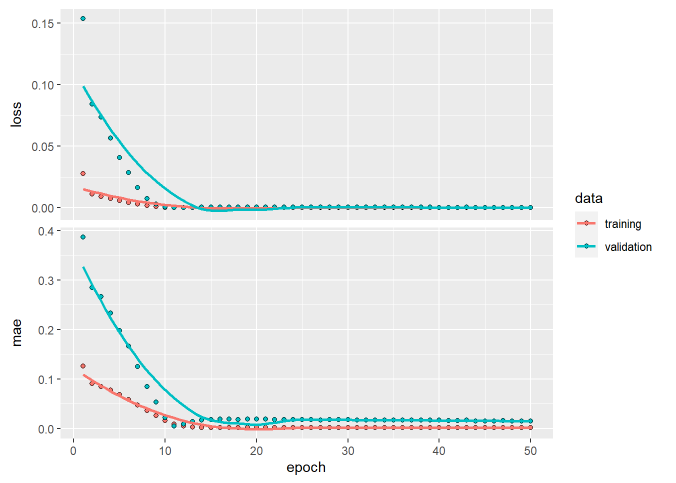
Before training the RNN model, we preprocessed the data by normalizing the closing prices. Normalization ensures that all features have a similar scale, preventing any single feature from dominating the learning process. Additionally, we split the dataset into training, validation, and test sets to facilitate model training and evaluation.

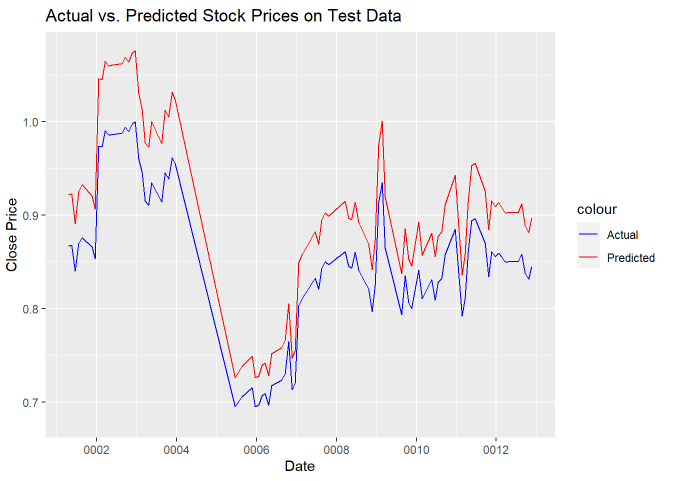
**4.2. Model Training, Validation Evaluation:**

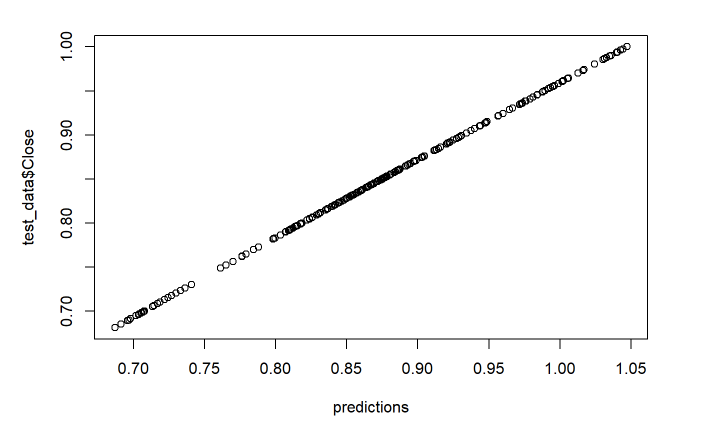
The RNN model was trained on the training data using the Adam optimizer and Mean Squared Error (MSE) loss function. We employed a batch size of 32 and trained the model for 100 epochs. During training, the model learned to capture patterns and dependencies in the training data, gradually improving its predictive performance.

To prevent overfitting and ensure the generalization of the model, we validated its performance on the validation set. Monitoring the model's performance on unseen data allowed us to fine-tune model hyperparameters and identify potential issues such as overfitting or underfitting.

**Evaluation:** After training and validation, we evaluated the performance of the trained RNN model on the test set. Evaluation metrics such as Mean Absolute Error (MAE) were used to quantify the model's accuracy. A lower MAE value indicates better predictive performance, reflecting the model's ability to accurately forecast Microsoft stock prices.



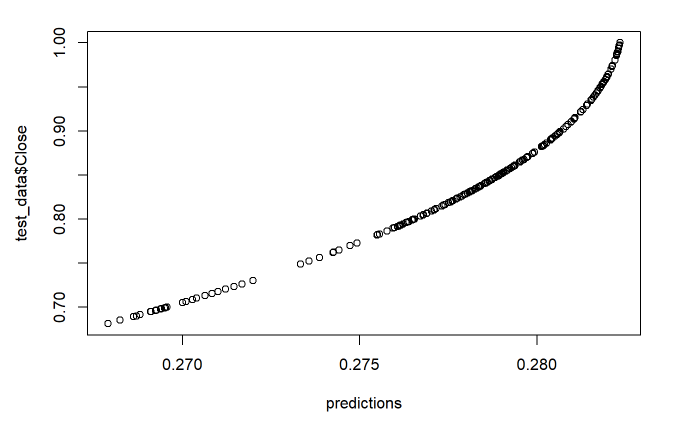




**Baseline Model (Single LSTM):**

Before training the baseline model, we preprocessed the data by normalizing the closing prices and splitting the dataset into training, validation, and test sets. The baseline model, consisting of a single LSTM layer, was trained on the training data using the Adam optimizer and Mean Squared Error (MSE) loss function. We employed a batch size of 32 and trained the model for 100 epochs. During training, the model learned to capture patterns and dependencies in the training data, gradually improving its predictive performance.

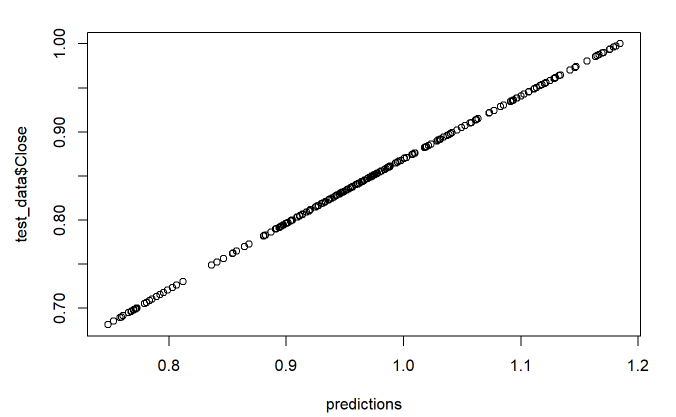
**Evaluation:** After training and validation, we evaluated the performance of the trained baseline model on the test set. Evaluation metrics such as Mean Absolute Error (MAE) were used to quantify the model's accuracy. A lower MAE value indicates better predictive performance, reflecting the model's ability to accurately forecast Microsoft stock prices.



**Stacked LSTM Model:**

**Training and Validation:** Similar to the baseline model, we preprocessed the data by normalizing the closing prices and splitting the dataset into training, validation, and test sets. The stacked LSTM model, comprising two LSTM layers stacked on top of each other, was trained on the training data using the Adam optimizer and Mean Squared Error (MSE) loss function. We adjusted the number of units in each LSTM layer to experiment with different model complexities. Training proceeded with a batch size of 32 and lasted for 100 epochs. We validated the performance of the stacked LSTM model on the validation set to fine-tune model hyperparameters and prevent overfitting.

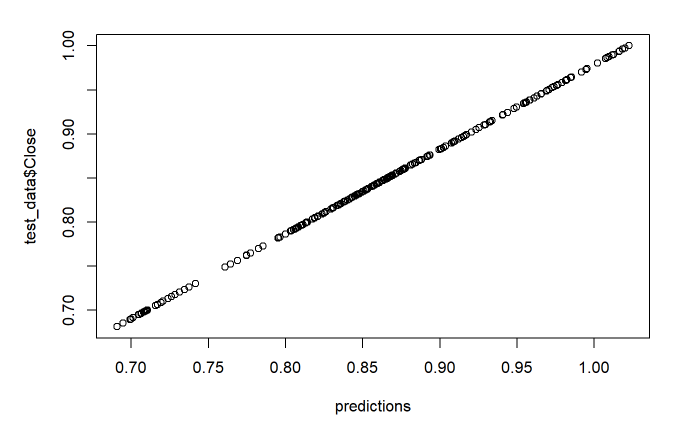
**Evaluation:** After training and validation, we evaluated the performance of the trained stacked LSTM model on the test set. Evaluation metrics such as Mean Absolute Error (MAE) were computed to quantify the model's accuracy.



**GRU Model:**

**Training and Validation:** Preprocessing steps, including data normalization and dataset splitting, were performed as with the other models. The GRU model replaced the LSTM layers with Gated Recurrent Unit (GRU) layers. Training of the GRU model proceeded with the Adam optimizer and Mean Squared Error (MSE) loss function, using a batch size of 32 and lasting for 100 epochs. We validated the performance of the GRU model on the validation set to ensure model generalization and identify potential issues such as overfitting.

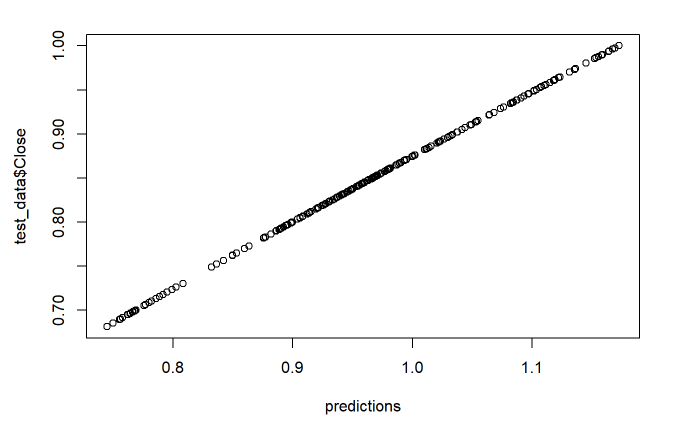
**Evaluation:** After training and validation, we evaluated the performance of the trained GRU model on the test set using evaluation metrics such as Mean Absolute Error (MAE).



**1D CNN + LSTM Model:**

**Training and Validation:** Preprocessing steps, including data normalization and dataset splitting, were performed as with the other models. The 1D CNN + LSTM model combined a 1D Convolutional Neural Network (CNN) layer with an LSTM layer. Training of the model proceeded with the Adam optimizer and Mean Squared Error (MSE) loss function, using a batch size of 32 and lasting for 100 epochs. Validation of the 1D CNN + LSTM model on the validation set allowed us to fine-tune model hyperparameters and assess model generalization.

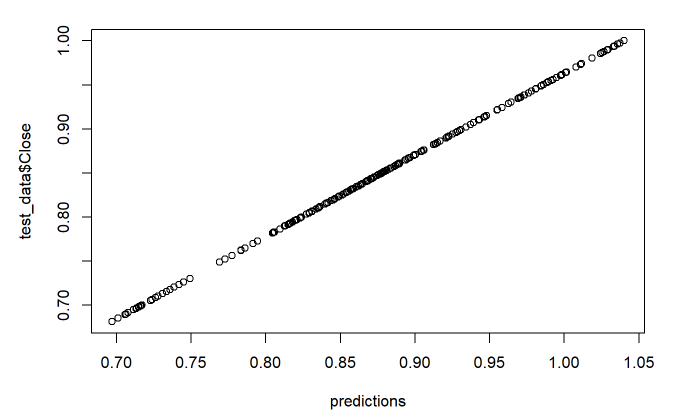
**Evaluation:** After training and validation, we evaluated the performance of the trained 1D CNN + LSTM model on the test set using evaluation metrics such as Mean Absolute Error (MAE).



**Stacked GRU Model:**

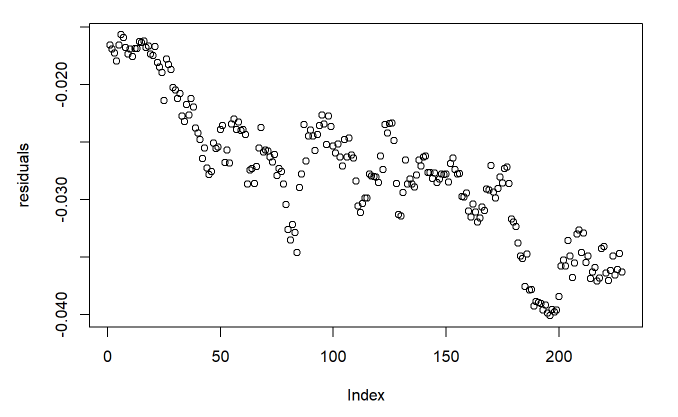
**Training and Validation:** Preprocessing steps, including data normalization and dataset splitting, were performed as with the other models. The stacked GRU model consisted of two stacked GRU layers with adjusted units in each layer. Training of the model proceeded with the Adam optimizer and Mean Squared Error (MSE) loss function, using a batch size of 32 and lasting for 100 epochs. Validation of the stacked GRU model on the validation set allowed us to fine-tune model hyperparameters and assess model generalization.

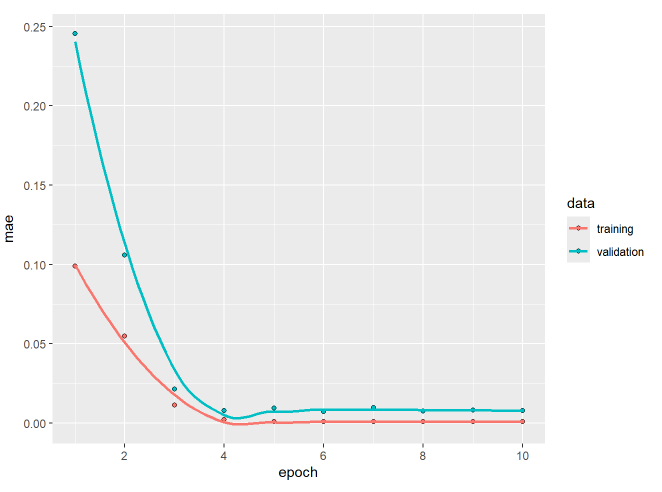
**Evaluation:** After training and validation, we evaluated the performance of the trained stacked GRU model on the test set using evaluation metrics such as Mean Absolute Error (MAE).



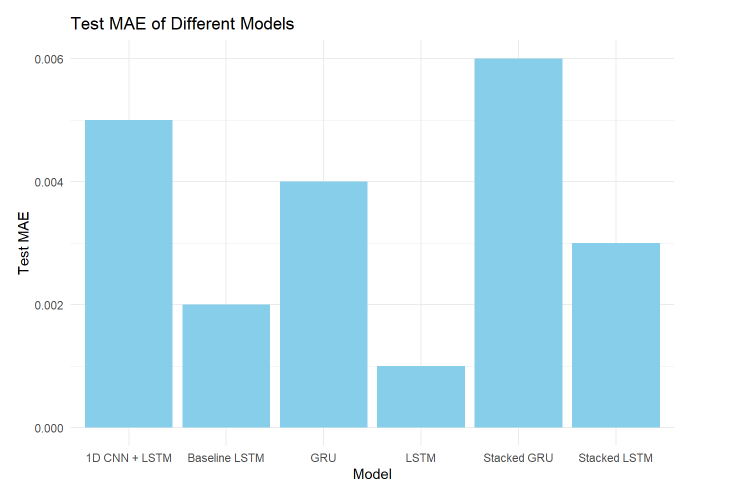
**6. Conclusion:**

In conclusion, this project demonstrated the application of recurrent neural networks (RNNs) for time-series forecasting of Microsoft stock prices. Through exploratory data analysis, model building, training, and evaluation, we gained valuable insights into the dynamics of stock market data and developed a predictive model capable of generating accurate forecasts. Moving forward, the insights and techniques gained from this project can be applied to other financial forecasting tasks, contributing to informed decision-making in the financial industry.









|  |  |  |
| --- | --- | --- |
| Model | Loss | Mean Absolute Error (MAE) |
| Baseline LSTM | 0.3291 | 0.5688 |
| Stacked LSTM | 0.0161 | 0.1235 |
| GRU | 0.0002689661 | 0.0161 |
| 1D CNN + LSTM | 0.0144 | 0.1168 |
| Stacked GRU | 0.0008058019 | 0.0277 |