Tech-GC Report

BRUTE VOL1

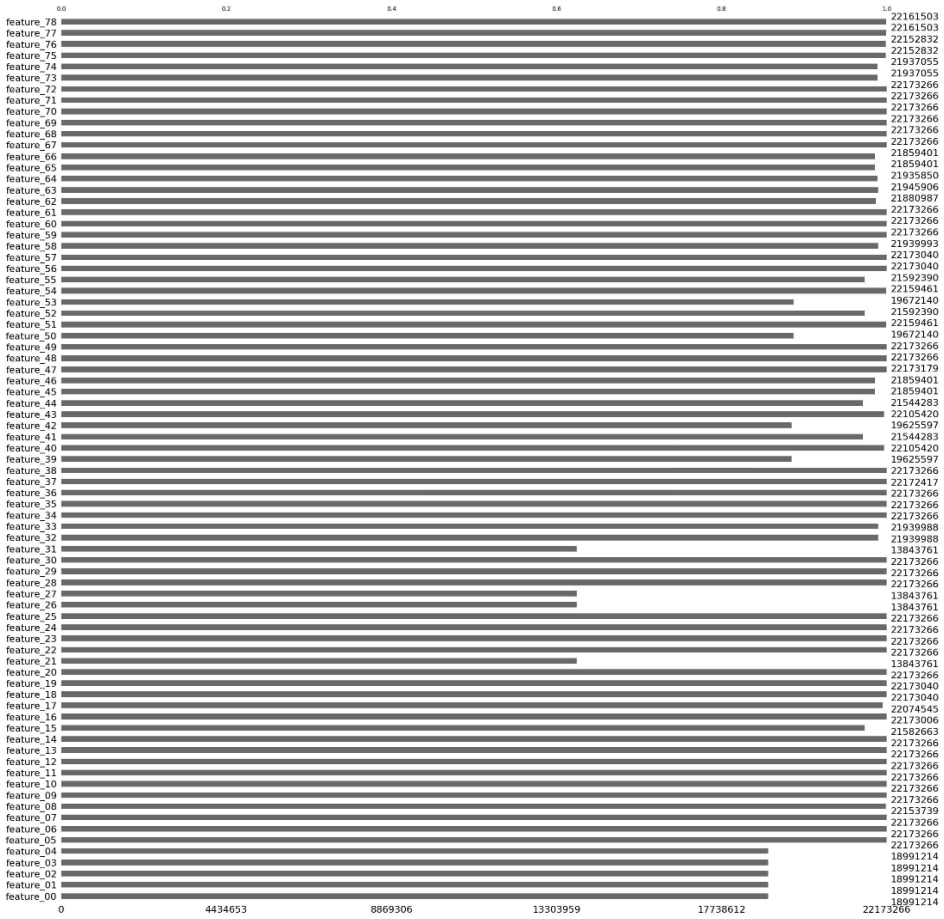
February 2025

1 Introduction

2 Exploratory Data Analysis (EDA)

2.1 Handling Missing Values

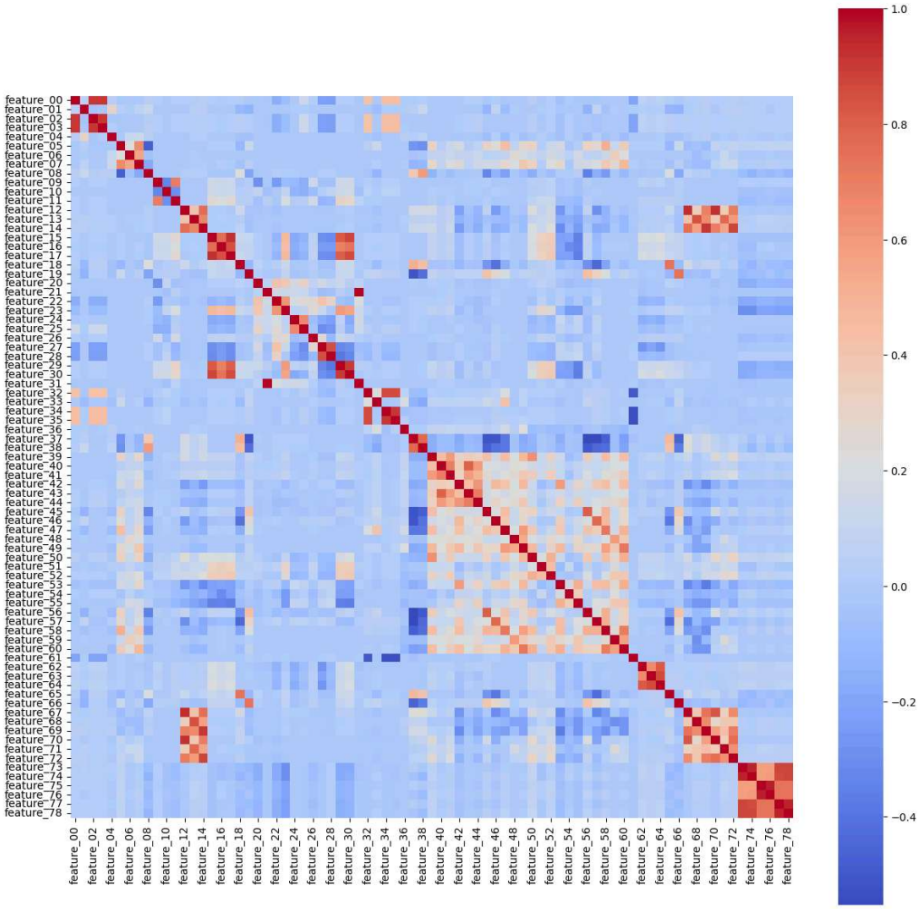
- **Observations from Missing Value Distribution:**
  - **High Missing Percentage (Above 30%)**
    - Features: feature_26, feature_27, feature_31, feature_21 (37.56%)
    - These features have a large proportion of missing values, which may lead to biased results or a significant loss of data if dropped.
    - If these features are crucial to the model, filling missing values is necessary.
  - **Moderate Missing Percentage (10% - 30%)**
    - Features: feature_00, feature_01, feature_02, feature_03, feature_04 (14.35%), feature_42, feature_39 (11.48%), feature_53, feature_50 (11.27%)
    - These features may impact model accuracy, especially if they are highly correlated with target variables.
    - Imputation methods or dropping these features should be carefully considered based on importance.
  - **Low Missing Percentage (Below 10%)**
    - Features: Various features with ¡5% missing values
    - These can be handled using simple imputation techniques such as mean, median, or mode imputation.



2.2 Correlation Analysis Using Heatmap

- **Correlation analysis using the heatmap:**
  - **General Pattern:**
    - The heatmap visualizes pairwise correlations between features.
    - The diagonal is deep red (correlation = 1), indicating each feature's perfect correlation with itself.
  - **High Correlation Regions:**
    - Some feature groups exhibit strong positive correlations (red patches), indicating potential redundancy.
    - There are small clusters where features are highly correlated, which could suggest feature groups with overlapping information.
  - **Low to Moderate Correlation:**
    - The majority of the heatmap is light blue, indicating weak correlations (near 0) between many features.
    - This suggests that most features contribute independently to the model.
  - **Negative Correlations:**
    - Some features show strong negative correlations (dark blue patches), meaning they move in opposite directions.
  - **Implications for Feature Selection:**
    - Features with high correlation (¿0.8 or ¡-0.8) might cause multicollinearity, affecting model interpretability and performance.

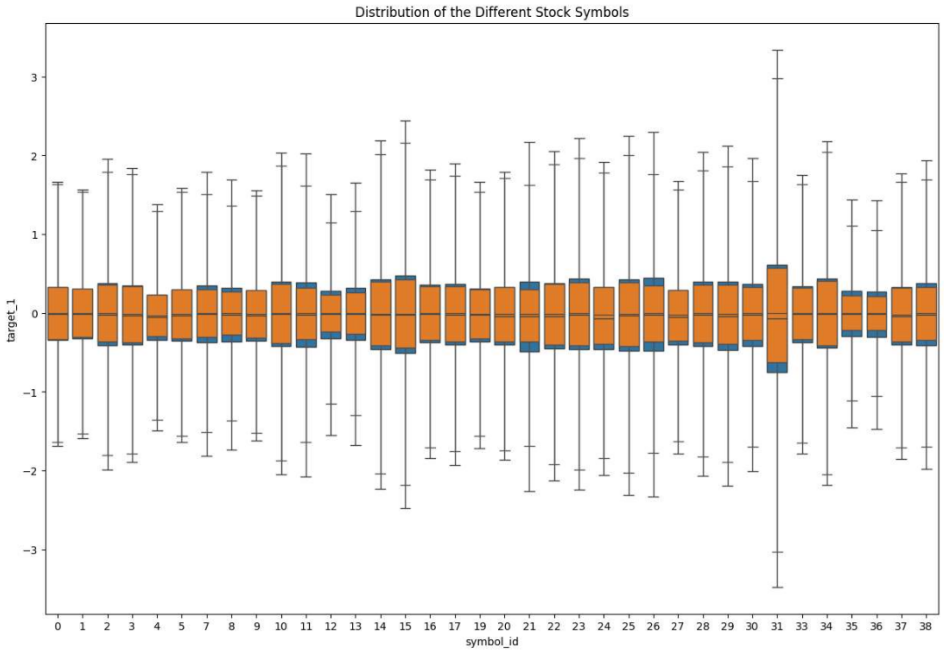
* Consider removing redundant features or using techniques like Principal Component Analysis (PCA) or Variance Inflation Factor (VIF) to handle multicollinearity.



2.3 Target Analysis

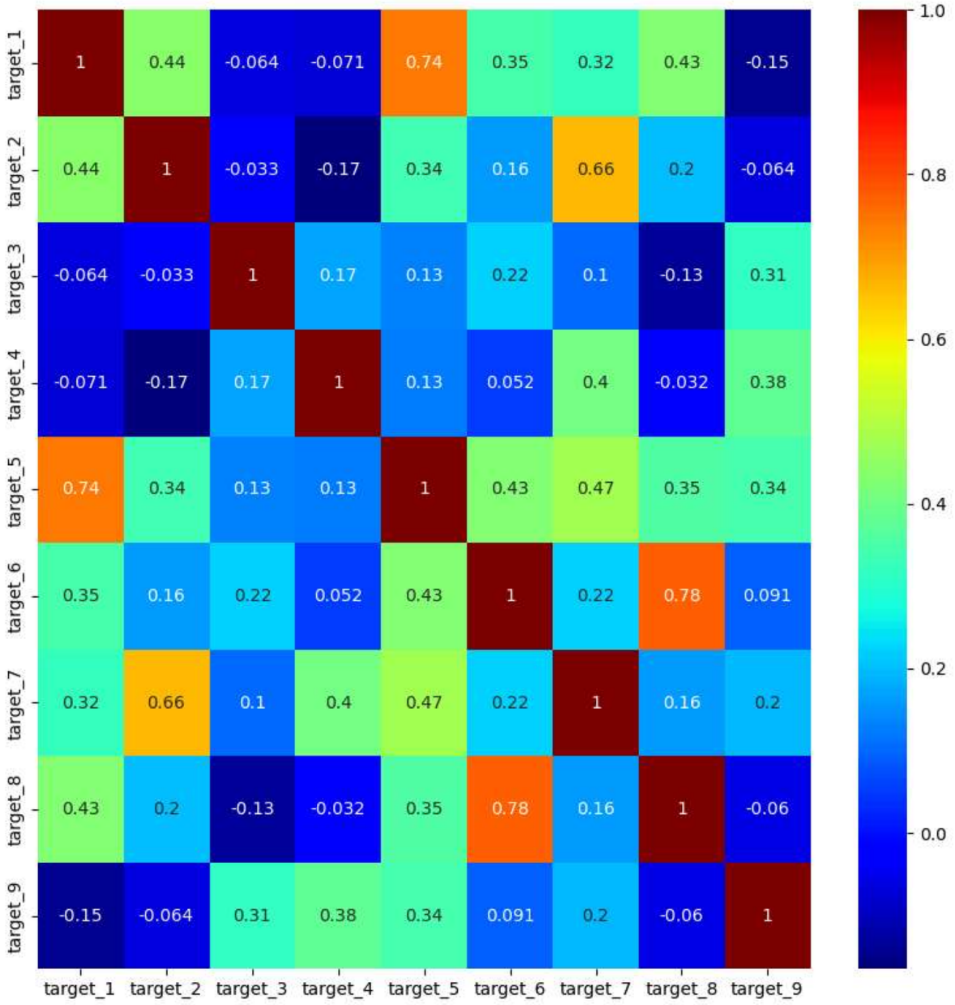
2.3.1 Target Correlation and Distribution

- **Target correlation and Distribution:**
  - **Stock-Specific Variability:**
    * The boxplot visualizes the distribution of the target variable (target_1) across different stock symbols (symbol_id). Each box represents a stock symbol, showing the spread and variability of the target variable.
    * Some symbols have wider IQRs, indicating higher variability in the target variable.
    * Others have narrower boxes, meaning their target values are more stable.
  - **Outliers:**
    * Some symbols have long whiskers and extreme points, suggesting the presence of outliers.
    * This could indicate volatile stocks or rare significant price changes.
  - **Stock-Specific Trends:**
    * Some stocks appear to have consistently higher or lower target values.
    * This could suggest underlying patterns in stock behavior that might be useful for modeling.
  - **Potential Feature Importance:**
    * If the distribution varies significantly across symbols, then symbol_id might be an important feature in predicting target_1.
    * Stocks with a similar median and spread might not provide much predictive power.



2.3.2 Target Correlation Using Heatmap

- **Target correlation using Heatmap:**
  - This heatmap represents the correlation matrix between different target variables (target_1 to target_9). The correlation values range from -1 to 1:
    - 1 (Red) → Perfect positive correlation (two variables move exactly together).
    - 0 (Green/Yellow) → No correlation.
    - -1 (Blue) → Perfect negative correlation (one variable increases while the other decreases).
  - **Key Insights:**
    - **Strong Positive Correlations:**
      - target_1 and target_5 (0.74)
      - target_6 and target_8 (0.78)
      - target_2 and target_7 (0.66)
      - These indicate that these variables tend to move together, meaning if one increases, the other also increases.
    - **Weak or No Correlation:**
      - target_3 and most other targets have correlations close to zero, indicating weak relationships.
      - target_4 has very low correlation with target_1 (-0.071) and target_2 (-0.17), suggesting little to no linear dependency.
    - **Negative Correlations:**
      - target_1 and target_9 (-0.15)
      - target_2 and target_3 (-0.033)
      - These suggest that when one variable increases, the other tends to decrease.
    - **Implications for Modeling:**
      - Highly correlated targets (above 0.6) could be redundant and might need dimensionality reduction (e.g., PCA).
      - Weak correlations suggest independence, meaning separate models for each target may be beneficial.
      - Negative correlations could be useful in hedging strategies in financial applications.

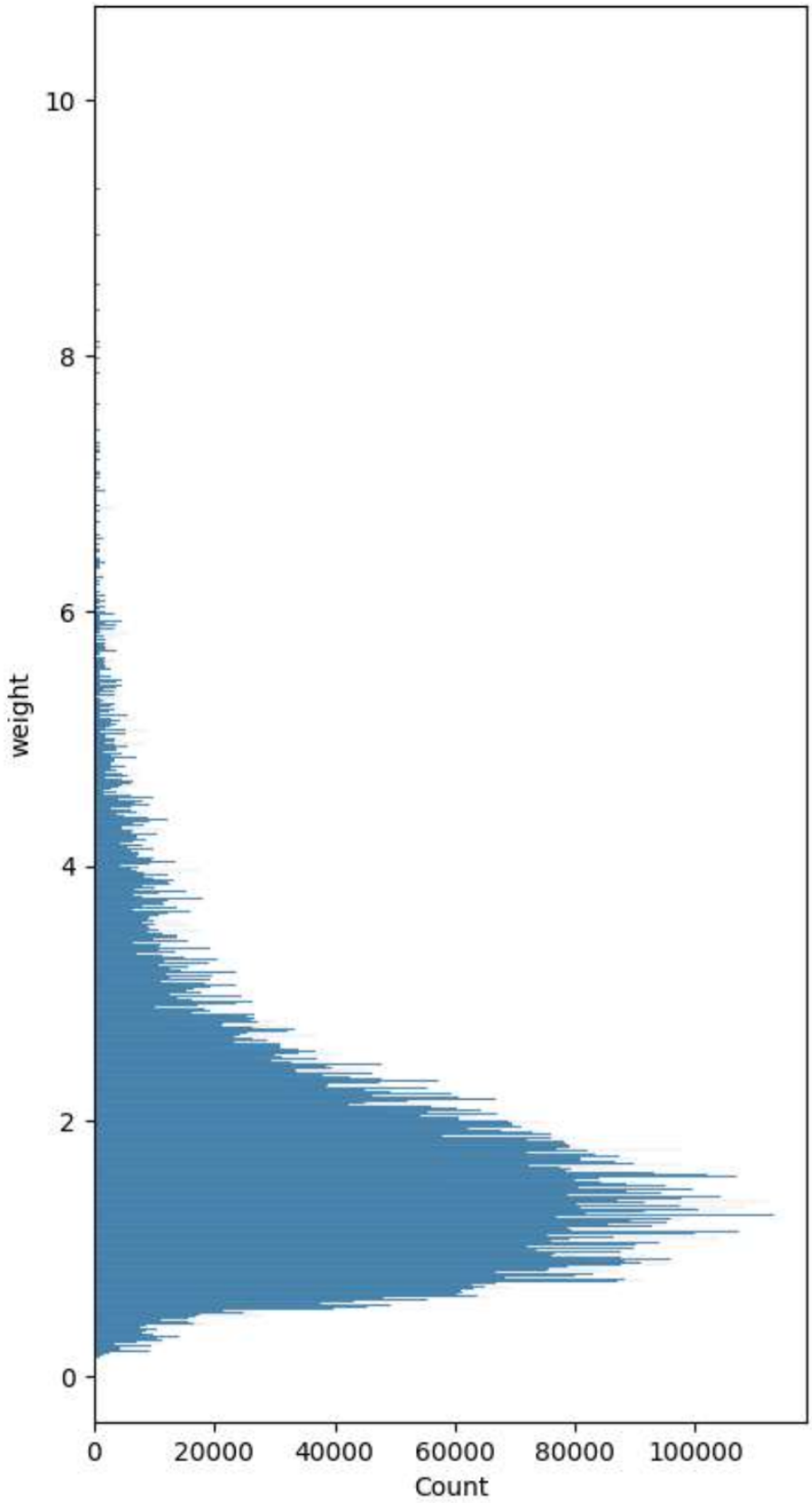


2.4 Feature Analysis

2.4.1 Histogram for Weight Column

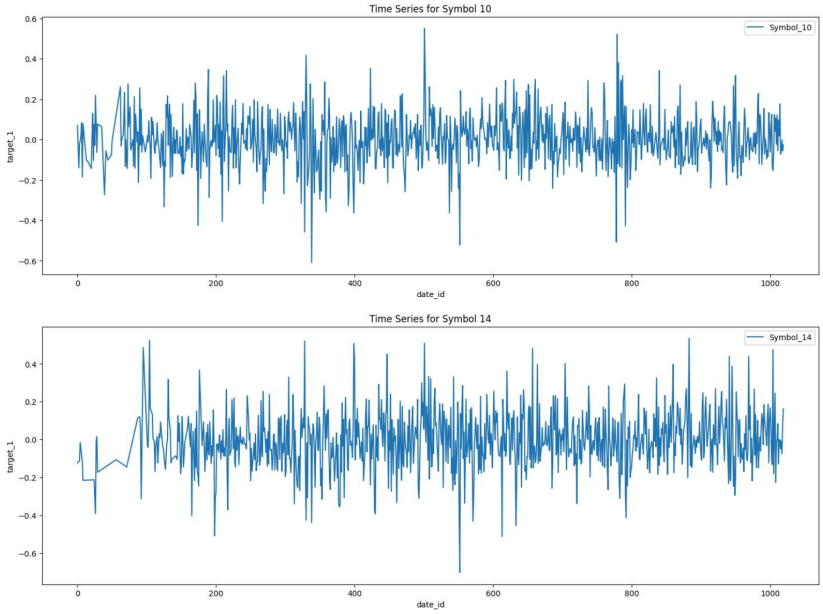
- **Histogram for the weight column:**
  - **Right-Skewed Distribution (Positive Skewness):**
    - The majority of the weight values are concentrated towards the lower end of the scale (left side).
    - As we move towards higher weight values, their frequency decreases gradually, forming a long right tail.
    - This suggests that most observations have low weight, but a few instances have high weight values.
  - **Possible Outliers in Higher Weights:**
    - The presence of high-weight values with lower frequency could indicate potential outliers.
    - These outliers might need further investigation—whether they are legitimate data points or erroneous entries.
  - **Non-Normal Distribution:**
    - Unlike a normal (bell-shaped) distribution, this histogram shows an asymmetric shape.
    - If you're using this data in machine learning models, data transformation (e.g., log transformation) might help normalize it.
  - **Heavy-Tailed Nature:**

* The gradual decline of the histogram suggests that some values are significantly larger than the bulk of the data.
* This might indicate a power-law-like behavior, where few instances contribute disproportionately to the overall weight.



2.5 Time-Series Analysis

- **Time-series analysis:**
  - **Identifying Trends and Patterns**
    - Time-series plots help reveal long-term trends, such as increasing or decreasing movements in the data.
    - They also show seasonality, where patterns repeat at regular intervals (e.g., daily, weekly, or yearly cycles).
    - Detecting these trends helps in forecasting and decision-making.
  - **Detecting Anomalies and Outliers**
    - Visualization of time-series data makes it easy to spot outliers, such as unexpected spikes or drops.
    - This is essential for industries like finance (fraud detection) and manufacturing (quality control monitoring).
    - Understanding anomalies helps in improving data quality and ensuring reliable model performance.
  - **Checking Stationarity for Modeling**
    - Many time-series models (like ARIMA) assume that the data is stationary (i.e., its statistical properties don't change over time).
    - EDA helps in identifying whether detrending or differencing is needed before applying forecasting models.
    - If the time-series is non-stationary, it might require transformations (log, differencing) before further analysis.
  - **Understanding Volatility and Noise**
    - Some time-series datasets exhibit high volatility, meaning the values fluctuate significantly.
    - A noisy dataset may require smoothing techniques, such as moving averages or exponential smoothing, before deriving meaningful insights.
    - This is particularly useful in financial market analysis, stock price predictions, and climate studies.



3 Model Explanation

3.1 GRU Model

- **Learning Rate:** 0.001

- **Optimizer:** Adam
- **Architecture:**

```
GRUModel(
  (gru): GRU(133, 500, batch_first=True)
  (dropout_gru): Dropout(p=0.3, inplace=False)
  (fc): Sequential(
    (0): Linear(in_features=500, out_features=500, bias=True)
    (1): ReLU()
    (2): Dropout(p=0.2, inplace=False)
    (3): Linear(in_features=500, out_features=300, bias=True)
    (4): ReLU()
    (5): Dropout(p=0.1, inplace=False)
    (6): Linear(in_features=300, out_features=1, bias=True)
  )
)
```

- **Why this model?**
  - GRUs are computationally efficient and mitigate the vanishing gradient problem, making them suitable for financial time series prediction.
  - 500 hidden units allow for capturing long-term dependencies in complex data.
  - Gradual dropout (0.3, 0.2, 0.1) helps prevent overfitting while maintaining learning ability.
  - Adam optimizer with LR = 0.001 ensures dynamic learning rate adjustments, leading to efficient convergence.

3.2 LightGBM Model

3.2.1 Model Parameters

The model was initialized with the following parameters:

- n_estimators: 1000, learning_rate: 0.005
- min_child_samples: 100, num_leaves: 50
- subsample: 0.9, colsample_bytree: 0.9
- random_state: 42, device: GPU

3.2.2 Training Process

Training was performed using Dask for parallel processing with early stopping and real-time weighted R^2 evaluation.

3.2.3 Results and Analysis

The LightGBM model yielded negative R^2 values on the validation set, indicating poor predictive performance. Possible reasons include:

- Overfitting to the training data
- Insufficiently informative features
- High data complexity or noise
- Suboptimal hyperparameter choices

3.2.4 Attempts to Improve Performance

We attempted:

- Adjusting learning rate and number of estimators
- Implementing early stopping
- Utilizing GPU acceleration for efficiency

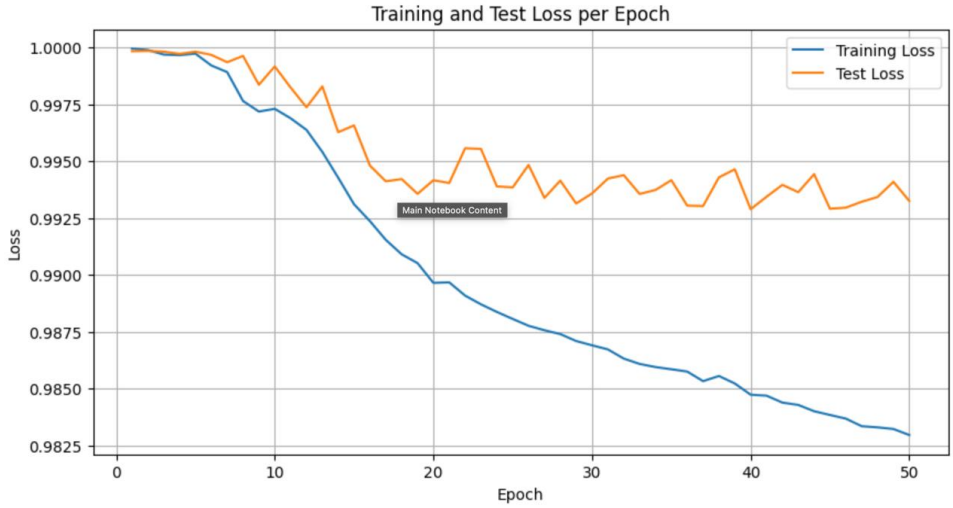
Despite these efforts, performance remained unsatisfactory.

3.3 LSTM Model

- **Dataset:** Using all files with an 80-20 train-validation split.
- **Learning Rate:** 0.001
- **Architecture:**

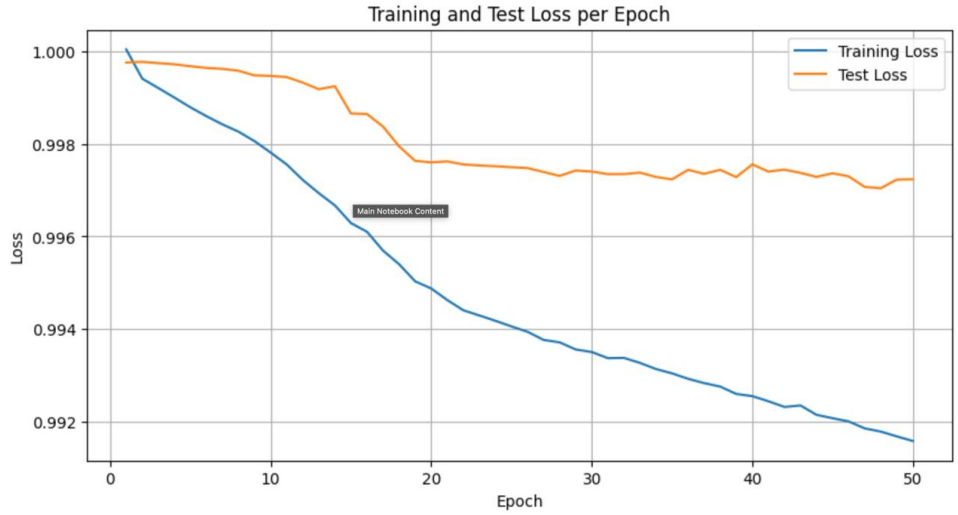
```
StockLSTM(
  (lstm): LSTM(141, 128, num_layers=2, batch_first=True, dropout=0.2)
  (fc): Linear(in_features=128, out_features=1, bias=True)
  (relu): LeakyReLU(negative_slope=0.01)
)
```

- **Why this model?**
  - LSTMs are superior for learning long-term dependencies, crucial for stock price predictions.
  - Two-layer stacked LSTMs allow for hierarchical feature extraction, improving accuracy.
  - 128 hidden units balance model complexity with training efficiency.
  - Dropout of 0.2 prevents overfitting while retaining key information.
  - LeakyReLU avoids dead neurons, ensuring smoother training and preventing vanishing gradients.



3.4 Previous LSTM Model with Lower Learning Rate

- **Learning Rate:** 0.0001
- **Why this model?**
  - Lowering the learning rate helps stabilize training and prevents sudden weight updates that could lead to suboptimal convergence.

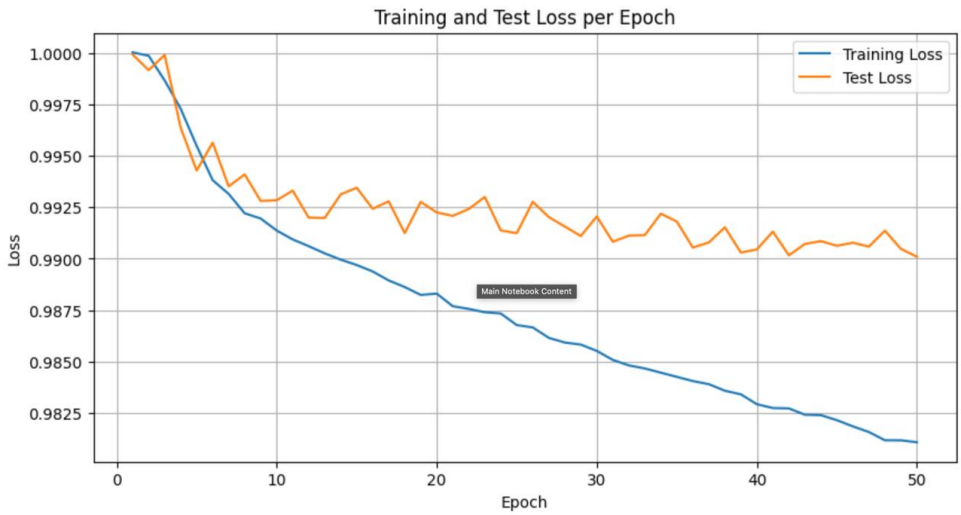


3.5 Effect of Learning Rate on Training

- Increasing the learning rate in the LSTM model caused overshooting of train loss.
- Higher learning rates led to fluctuations in loss, preventing proper convergence.

3.6 Reducing Batch Size for Faster Training

- Reduced batch size from 80,000 to 10,000 in the LSTM model.
- This change improved training speed by making each epoch run faster.
- Smaller batches allow for quicker updates while maintaining model generalization.
- A batch size that is too small can introduce noisy gradients, negatively impacting convergence.



3.7 Model Comparison

Here's a summary of the performance of all models tested:

Model	R^2 Score
GRU	0.007992472
LSTM (LR=0.001)	0.018403157
LSTM (LR=0.0001)	0.002763468
LightGBM	Negative

Table 1: Performance comparison of different models

The negative R^2 scores for the LightGBM model indicate that it performed worse than a simple mean predictor, suggesting that further optimization or different modeling approaches may be necessary for this dataset.

4 Feature Distributions

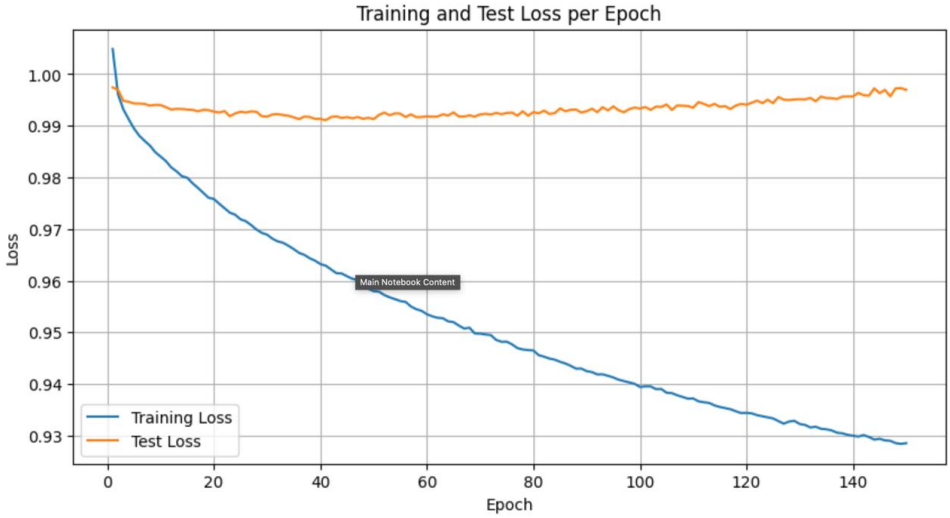
To ensure proper data preprocessing, we categorized features based on their underlying distributions:

- **Normal Distribution:** feature_01, feature_02, feature_03, feature_04, feature_05, feature_06, feature_07, feature_08, feature_18, feature_19, feature_31, feature_32, feature_33, feature_34, feature_35, feature_36, feature_37, feature_38, feature_39, feature_40, feature_41, feature_42, feature_43, feature_44, feature_45, feature_49, feature_50, feature_51, feature_52, feature_53, feature_54, feature_55, feature_56, feature_60, feature_64, feature_65.
- **Log-Normal Distribution:** feature_12, feature_13, feature_14, feature_15, feature_16, feature_17, feature_30, feature_61, feature_62, feature_63.
- **Exponential Distribution:** feature_20, feature_21.
- **Gamma Distribution:** feature_22, feature_23, feature_24, feature_25, feature_66, feature_67, feature_68, feature_69, feature_70.
- **Beta Distribution:** feature_09, feature_10.
- **Uniform Distribution:** feature_26, feature_27, feature_28, feature_29.
- **Gaussian Mixture Model (GMM):** feature_46, feature_47, feature_48, feature_57, feature_58, feature_59.
- **F Distribution:** feature_71, feature_72, feature_73, feature_74, feature_75, feature_76, feature_77.
- **Binomial Distribution:** feature_09, feature_10.
- **Poisson Distribution:** feature_11.

5 Model testing

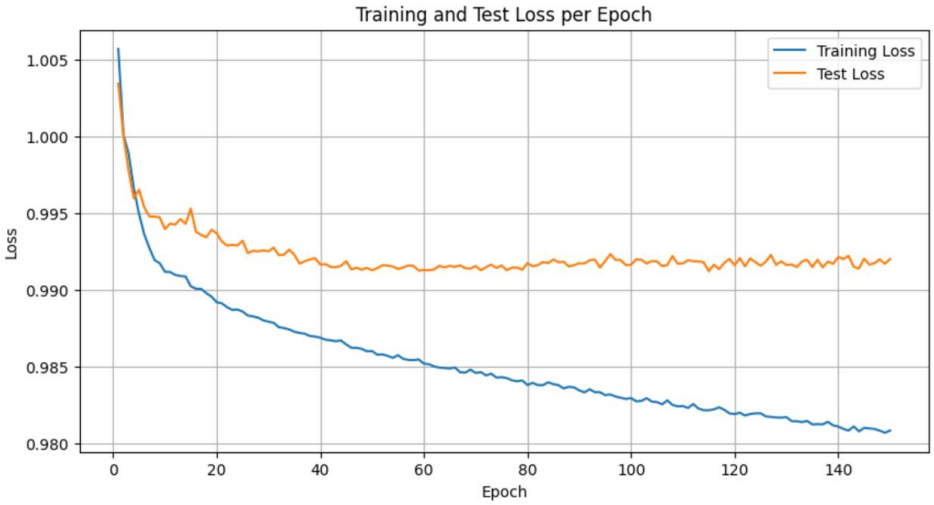
5.1 GRU Model - Using first 6 files

- Train and test loss values when training GRU with first 6 parquet files.



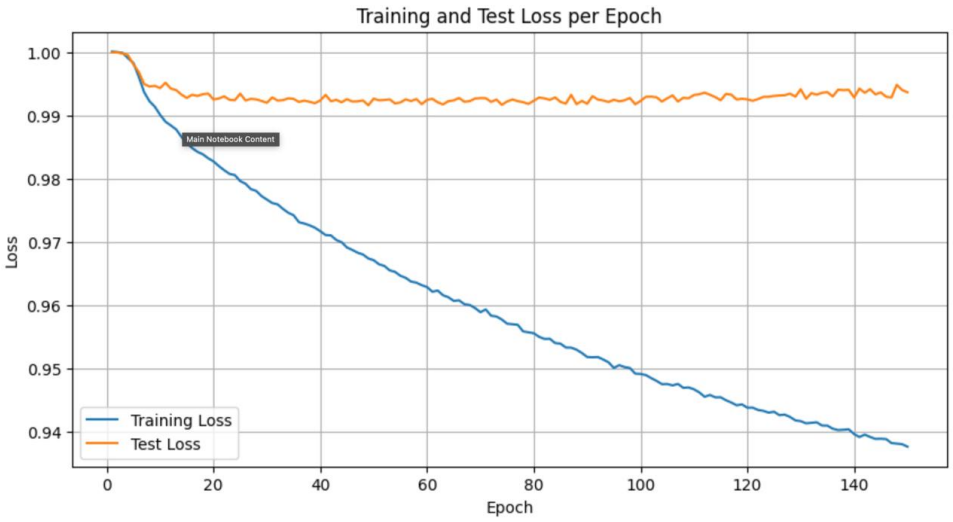
5.2 GRU Model - Using last 6 files

- Train and test loss values when training GRU with last 6 parquet files.



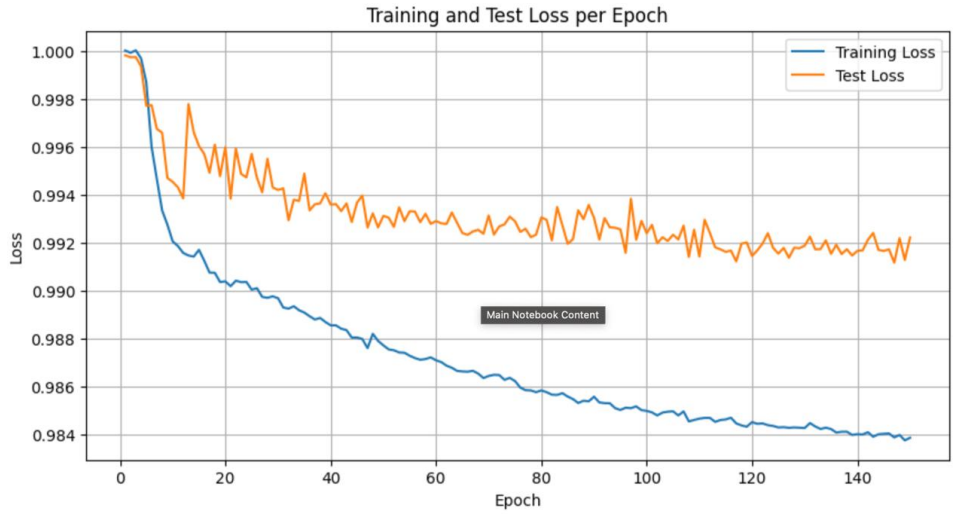
5.2.1 LSTM Model - Using first 6 files

- Train and test loss values when training LSTM with first 6 parquet files.



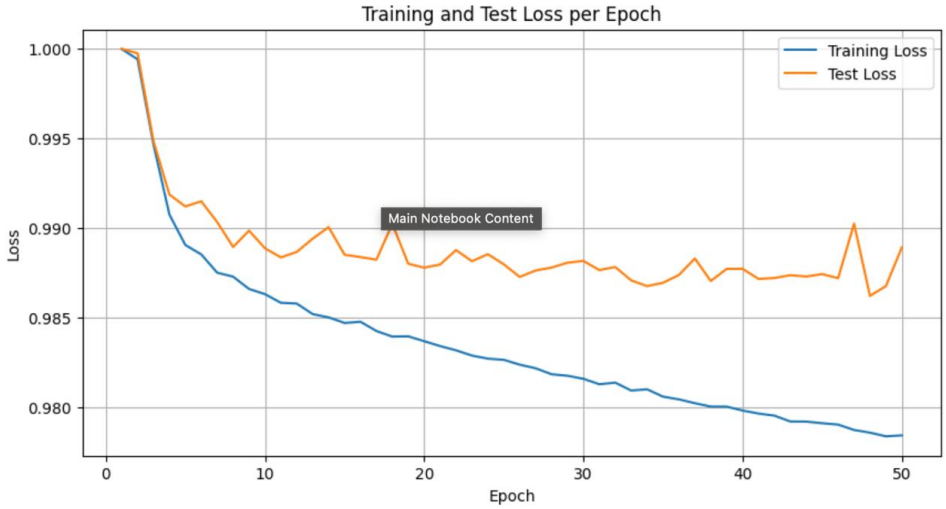
5.2.2 LSTM Model - Using last 6 files

- Train and test loss values when training LSTM with last 6 parquet files.



5.2.3 LSTM Model- Using all 12 files

- Train and test loss values when training LSTM with all the 12 parquet files.



These results indicate that using the entire dataset yields better performance compared to relying only on recent data.

6 Final Model

6.1 Model Improvements

- Applied clipping to predictions in the range of -5 to 5.
- Removed ReLU activation from the last layer.

6.2 Final Model Architecture

```
class StockLSTM(nn.Module):
    def __init__(self, input_size, hidden_size=128, num_layers=2, output_size=2, dropout=0.2):
        super(StockLSTM, self).__init__()
        self.lstm = nn.LSTM(input_size, hidden_size, num_layers, batch_first=True, dropout=dropout)
        self.fc = nn.Linear(hidden_size, output_size)

    def forward(self, x):
        lstm_out, _ = self.lstm(x)
        out = self.fc(lstm_out[:, -1, :])
        return torch.clamp(out, -5, 5)
```

6.3 Final Loss Function

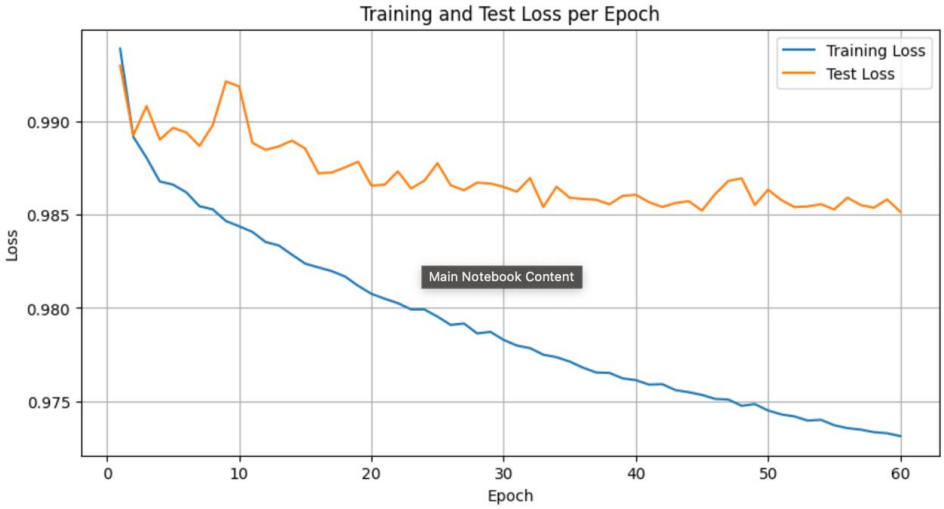
```
class R2Loss(nn.Module):
    def __init__(self):
        super(R2Loss, self).__init__()

    def forward(self, y_pred, y_true, weights):
        ss_total = torch.sum(weights * (y_true ** 2), dim=0)
        ss_residual = torch.sum(weights * ((y_true - y_pred) ** 2), dim=0)
        r2 = 1 - (ss_residual / (ss_total + 1e-8))
        return 1 - torch.mean(r2)
```

6.4 Training Parameters

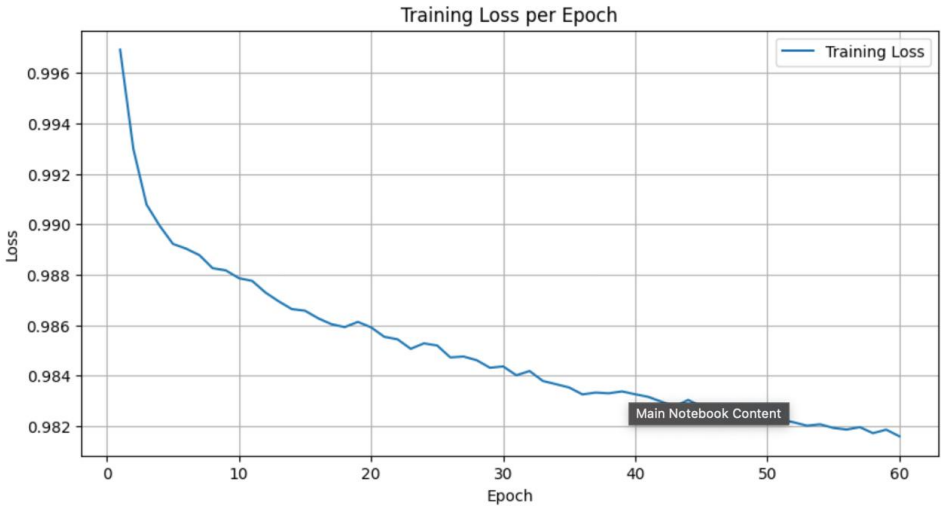
- Number of input features: 141
- Batch size: 10,000

- Optimizer: Adam
- Number of epochs: 60

6.5 Model Performance Visualizations



Train and test loss graph with 80-20 split.



Final model training on full data.