# Predicting 15-Minute Price Changes in VN30F Using Deep Learning

## Objective

The task is to build an AI model to predict the 15-minute percentage change in the closing price of the VN30F1M futures contract, using historical OHLC and trading volume data. The percentage change is calculated by comparing the closing price at each 15-minute milestone with that of 15 minutes earlier.

The change is then categorized into three classes:

- **+1 (Increase):** if the change is greater than +0.1%
- **0 (No Change):** if the change is between -0.1% and +0.1%
- **-1 (Decrease):** if the change is less than -0.1%

The final 30 days of the dataset were used as the test set, while the remaining data was used for training and validation.

## Model Selection Rationale

Several models were considered for this classification task, including Random Forest and XGBoost. While both tree-based models produced extremely high accuracy on the test set (100% and 99% respectively), such results suggested strong overfitting and poor generalization. This is further supported by flat accuracy curves and perfect precision/recall scores across all classes—even on unseen data—raising concerns about the test set's adequacy or the models' ability to generalize beyond it.

In contrast, deep learning models like LSTM and CNN are well-suited for time-series data due to their ability to learn temporal patterns. LSTMs can capture both short- and long-term dependencies, while Conv1D layers can extract local patterns and reduce noise. The combination of Conv1D with Bidirectional LSTM enhances both local and contextual learning, allowing the model to leverage information from both past and future time steps. This hybrid architecture is more robust to noise and better suited for high-frequency financial data with complex patterns.

## Data Preprocessing

The raw dataset was cleaned by removing irrelevant columns, and the `Time` column was parsed into datetime format and set as the index. The percentage change in closing price over a 15-minute window was calculated, and the data was labeled into one of the three classes as described above.

## Feature Engineering

To improve model performance and allow it to better learn patterns from the data, several technical indicators were engineered based on domain knowledge from financial analysis. These features help the model understand not just the current market snapshot, but also short-term dynamics such as volatility, trend, and momentum:

- **Moving Averages (5-period and 10-period):** These are simple averages of the closing price over the past 5 and 10 intervals, respectively. They help detect short-term trends and smooth out noise.
- **Rolling Standard Deviation (5-period) — `std_5`:** This measures the volatility of prices in the recent past. Higher values indicate more unstable price movements, which often precede breakouts or reversals.
- **One-period Return — `return_1`:** The percentage change in price between the current and previous interval. It captures immediate momentum and is a direct signal of recent price direction.
- **Volume Change — `volume_change`:** Computed as the relative change in volume compared to the previous interval. Volume surges often accompany price breakouts or shifts in sentiment.
- **Relative Strength Index (RSI, 5-period) — `rsi_5`:** A momentum oscillator that measures the speed and change of price movements. Values above 70 or below 30 typically indicate overbought or oversold conditions, respectively.

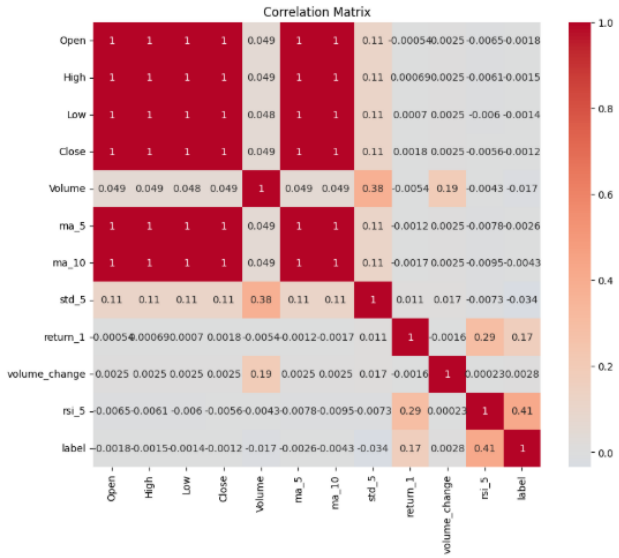


Figure 1: Correlation Matrix

After generating these features, correlation analysis was performed to identify the most informative and non-redundant ones. Based on the results, the following five features were selected for the final model input: `Close`, `std_5`, `return_1`, `volume_change`, and `rsi_5`. These features together provide a rich snapshot of recent price trends, volatility, momentum, and market strength.

## Data Scaling and Sequencing

All selected features were normalized using MinMaxScaler. Input sequences of 15-minute windows were created to match the prediction interval, making each input sample a sequence of 15 time steps.

## Model Architecture

The final deep learning model includes:

- Conv1D layers for local pattern extraction
- MaxPooling and BatchNormalization

- Bidirectional LSTM layers for temporal modeling
- Dense layers for classification
- Softmax activation at the output layer (3 classes)

Hyperparameters including LSTM units, dropout rate, and learning rate were tuned using grid search and validated using TimeSeriesSplit cross-validation.

# Evaluation

The best-performing model was trained on the entire training set and evaluated on the 30-day test set.

## Evaluation Metrics

**Confusion Matrix:**

$$\begin{bmatrix} 763 & 197 & 1 \\ 121 & 2877 & 167 \\ 4 & 137 & 768 \end{bmatrix}$$

**Classification Report:**

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Down | 0.86 | 0.79 | 0.83 | 961 |
| Neutral | 0.90 | 0.91 | 0.90 | 3165 |
| Up | 0.82 | 0.84 | 0.83 | 909 |
| accuracy | | | 0.88 | 5035 |
| macro avg | 0.86 | 0.85 | 0.85 | 5035 |
| weighted avg | 0.88 | 0.88 | 0.88 | 5035 |

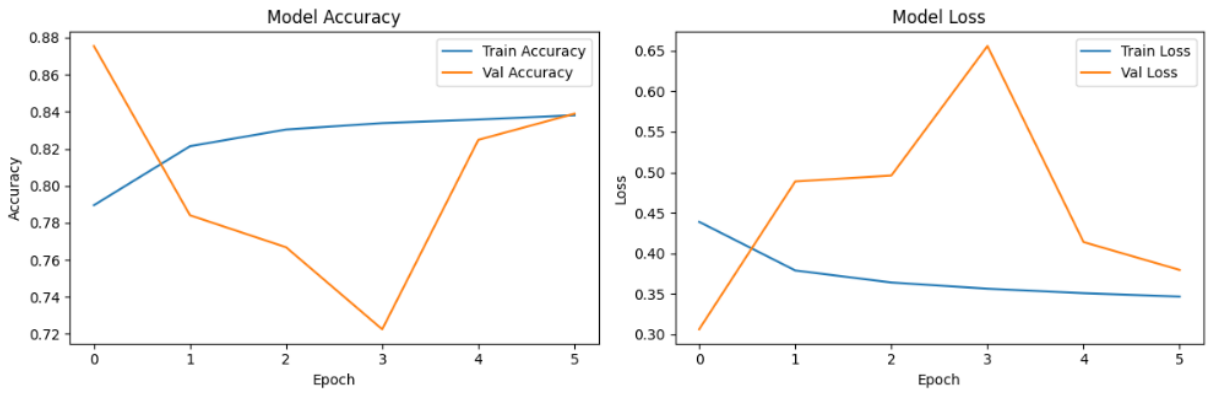


Figure 2: Training and Validation Accuracy / Loss Over Epochs

## Result Interpretation

The hybrid CNN-BiLSTM model achieved an overall accuracy of **88%** on the test set. The model showed strong generalization, with high precision and recall across all classes.

- **Neutral class:** Most frequent class in the dataset and the model handled it with excellent precision (90%) and recall (91%), resulting in a 90% F1-score.

- **Down and Up classes:** Both minority classes were learned reasonably well:
  - Down: 86% precision, 79% recall — suggests more false negatives, i.e., some down moves were misclassified as neutral.
  - Up: 82% precision, 84% recall — indicates slightly better detection of upward movements.

The training and validation curves reveal early signs of overfitting (epoch 3–4) followed by convergence, likely due to the use of regularization techniques like dropout and batch normalization.

## Comparison with Random Forest and XGBoost

**Note on Feature Representation:** The Random Forest and XGBoost models were trained using a separate, simplified feature engineering pipeline. The data was first resampled into 15-minute intervals, and each data point consisted of static snapshot features: one-period return (`return_1`), volume change (`volume_change`), and closing price (`Close`). These models operated on single time-step representations rather than sequences. While this makes training simpler and faster, it also significantly reduces the number of training samples compared to sequence-based deep learning models. In financial time series where patterns are often subtle and noisy, this lower data granularity can hinder generalization.

Despite these limitations, both Random Forest and XGBoost achieved near-perfect performance on the test set (100% and 99% accuracy, respectively). However, such results are almost certainly due to overfitting. The test set was drawn from the same aggregated structure as the training data, without explicit modeling of temporal dependencies. Furthermore, the models showed flat learning curves and perfect precision and recall across all classes, even on unseen data. These signs point to memorization rather than generalization and raise concerns about the reliability of these models in live trading or production settings.

In contrast, the CNN-BiLSTM model was trained on sequences of 15-minute windows using five engineered features and learned temporal relationships directly. While it achieved a slightly lower accuracy (88%), the performance was more balanced and realistic across classes. The deep learning model was better equipped to handle the dynamic nature of the financial market and demonstrated stronger generalization to unseen patterns. This makes it more suitable for real-world deployment, where adaptability and robustness are critical.

## Conclusion

The hybrid CNN-LSTM model demonstrates strong potential for predicting short-term price direction in the VN30F futures market. With an overall accuracy of 88% and well-balanced class performance, it outperforms traditional tree-based models in terms of generalization. Future improvements could involve the use of attention layers, transformer architectures, or the integration of external macroeconomic indicators and order book features.