ULTRA-LOW

Latency Market Microstructure Modeling

Presented By : TEAM DELTA

ITSOLERA DL PROJECT

BACKGROUND:

Financial markets have evolved significantly with electronic trading, where speed and efficiency are crucial. High-frequency trading (HFT) firms rely on ultra-low latency to execute trades faster than their competitors, gaining a critical edge.
Market microstructure modeling is key to this process. Traditional models are often too slow for HFT needs, which require real-time operation with minimal delay. This project aims to develop and implement models that meet these strict time demands.

PROBLEM STATEMENT

In high-frequency trading and real-time financial decision-making, there's a crucial need for ultra-low latency market microstructure models. Current models often fall short, causing inefficiencies and missed trading opportunities. This project seeks to fill this gap by creating models that operate with ultra-low latency, allowing traders to execute trades quickly and efficiently.

The challenge is to develop and implement market microstructure models that can handle large amounts of market data and make real-time predictions with ultra-low latency. This will enable faster and more accurate trading decisions.

DATA COLLECTION

For this project, data will be sourced from Kaggle, which offers a wide range of high-frequency financial datasets.

DATA PREPROCESSING

In the data cleaning process, we checked the dataset for missing values and irrelevant data points that could impact the model's performance. We removed missing values and outliers to ensure data quality. Scaling was performed to prepare the data for the LSTM model.

The dataset was then split into training, validation, and test sets for effective model training and evaluation. Raw tick data was transformed into time series data suitable for LSTM input. To address the high-frequency nature of the data, techniques like data aggregation and resampling were explored for data augmentation.

MODEL ARCHITECTURE

The model features a two-layer LSTM architecture. The first LSTM layer has 50 units, returns sequences, and takes an input shape of (X_train.shape[1], X_train.shape[2]). A Dropout layer with a rate of 0.2 follows to reduce overfitting.

The second LSTM layer, also with 50 units, does not return sequences. Another Dropout layer with a 0.2 rate is applied before the final Dense layer, which outputs a single value. This setup ensures robust learning and prevents overfitting while maintaining high performance. The sequence-to-sequence output from the first LSTM layer allows for complex time series predictions, enhancing model accuracy.

TRAINING PROCESS

The model uses mean_squared_error as the loss function, which is ideal for regression tasks aiming to minimize the difference between predicted and actual values. The adam optimizer is employed for its efficiency and adaptability in training deep learning models. Training is conducted with a batch size of 32, meaning weights are updated after processing 32 samples.

The model is set to train for 4 epochs, but training may end early if there is no improvement in validation loss. Early stopping is applied with a patience of 10, halting training if validation loss does not improve for 10 consecutive epochs, and restoring the best model weights to prevent overfitting

PERFORMANCE

The model uses mean_squared_error (MSE) for evaluation, but additional metrics such as Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE) can provide deeper insights. RMSE highlights large errors, MAE offers a straightforward interpretation without heavy penalties on large errors, and MAPE indicates error relative to actual values, though it's less effective near zero. Test loss reflects performance on unseen data, with lower values indicating better results. A plot comparing actual versus predicted values helps assess model accuracy, and while the model's latency should be low given its simplicity and short training duration, further optimizations may be needed for ultra-low latency applications. Overall, the model is effective for time series prediction, though additional evaluation and optimization might be required.

MODEL PICTURES

Libararies we used in our model :

```
import pandas as pd
import numpy as np
import os
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.metrics import mean_squared_error
import math
import time
import matplotlib.pyplot as plt
```

Function to load and concatenate all CSV files from a specified folder into a single pandas:

```
def load_data_from_folder(folder_path, file_extension='.csv'):  # Adjust extension as needed
    data_frames = []
    for root, _, files in os.walk(folder_path):
        for file in files:
            if file.endswith(file_extension):  # Adjust the extension as needed
                file_path = os.path.join(root, file)
                print(f"Loading {file_path}")
                df = pd.read_csv(file_path)  # Adjust parameters as needed
                data_frames.append(df)
    if not data_frames:
        raise ValueError(f"No {file_extension} files found in the specified folder.")
    return pd.concat(data_frames, ignore_index=True)

# Specify the folder containing the dataset
dataset_folder = 'high-frequency_dataset'
df = load_data_from_folder(dataset_folder)
```

Dataset after cleaning and preprocessing:

```
X
```

	Time	Open	High	Low	Volume	Amount	Open Interest
0	2015-03-27 20:59:00.005004	1.000000	1.000000	1.000000	2.0	238540.0	610.0
1	2015-03-27 21:00:00.005000	0.999665	0.999665	0.993628	32.0	3802040.0	598.0
2	2015-03-27 21:01:00.004997	0.995221	0.995221	0.994382	16.0	1898500.0	612.0
3	2015-03-27 21:02:00.005003	0.994382	0.996143	0.993879	12.0	1424740.0	616.0
4	2015-03-27 21:03:00.005000	0.993796	0.995556	0.993796	6.0	712020.0	618.0
...	...	...	...	...	...	...	...
25158886	2021-02-01 22:55:00.210	2.939837	2.942166	2.938672	506.0	127716350.0	150510.0
25158887	2021-02-01 22:56:00.209	2.941584	2.943913	2.941584	381.0	96266800.0	150411.0
25158888	2021-02-01 22:57:00.224	2.943331	2.945078	2.942749	326.0	82391800.0	150319.0
25158889	2021-02-01 22:58:00.187	2.945661	2.946825	2.945078	507.0	128235500.0	150109.0
25158890	2021-02-01 22:59:00.188	2.946825	2.946825	2.943913	689.0	174263900.0	149822.0

25138619 rows × 7 columns

MODEL testing and training process:

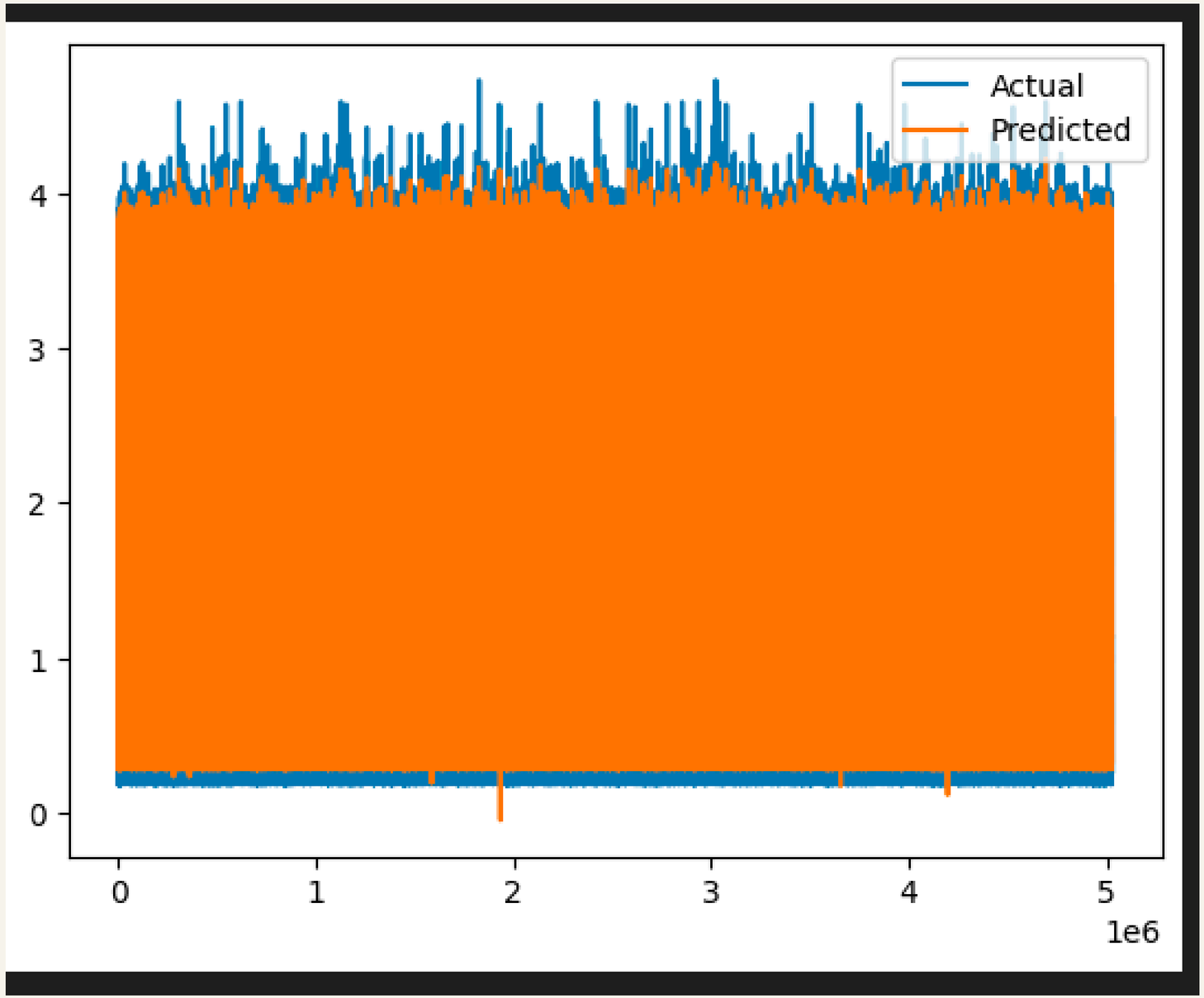
```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X.drop('Time', axis=1))  # Exclude 'Time' column from scaling
y_scaled = scaler.fit_transform(y.values.reshape(-1, 1))
```

```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_scaled, test_size=0.2, random_state=42)
```

```
X_train = X_train.reshape((X_train.shape[0], 1, X_train.shape[1]))
X_test = X_test.reshape((X_test.shape[0], 1, X_test.shape[1]))

# Build the LSTM model
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(Dropout(0.2))
model.add(LSTM(units=50))
model.add(Dropout(0.2))
model.add(Dense(1))
```

Actual and Prediction presentation by Graph:



Final Latency and output of our dataset:

```
Latency for this operation: 4.291534423828125e-06 seconds
Latency for this operation: 4.5299530029296875e-06 seconds
Bought at [0.85845196]
Latency for this operation: 0.00013971328735351562 seconds
Latency for this operation: 5.483627319335937e-06 seconds
Sold at [1.9001871], Profit: [1.0417352]
Latency for this operation: 0.00025248527526855469 seconds
Latency for this operation: 6.198883056640625e-06 seconds
Bought at [0.85845196]
Latency for this operation: 0.00014138221740722656 seconds
Sold at [1.8865491], Profit: [1.0280972]
Latency for this operation: 0.00029730796813964844 seconds
Latency for this operation: 6.914138793945312e-06 seconds
...
Latency for this operation: 7.62939453125e-06 seconds
Latency for this operation: 0.00053548812866210938 seconds
Final Balance: [114432.945]
Profit: [104432.945]
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

THANK YOU