Algorithmic Trading

With Deep Learning

Ritesh Kumar

2024SP_MS_DSP_498-DL_SEC61: Capstone

Module 10

Project Report

Kimberly Chulis and Srabashi Basu

August 25, 2024

Table of Contents

Purpose and Objective

The project was focused on developing a sophisticated algorithmic trading system for BankNifty, an Indian banking index, by leveraging deep learning techniques to predict short-term market movements. The primary purpose was to create a robust model capable of generating consistent profitability through optimized trading decisions and effective risk management. By capturing and analyzing data at one-minute intervals, the project aimed to make informed trading decisions, ultimately leading to significant gains in real-time trading environments.

The primary objective was to design and implement an advanced algorithmic trading system that analyzed historical data to inform trading decisions using deep learning models. The system utilized data such as Cumulative Open Interest (COI), price, volume, and various technical indicators including Moving Averages, RSI, and MACD. The project involved feature engineering, model comparison, and backtesting, with a focus on identifying the most effective model for predicting significant next-minute price movements.

The project explored deep learning models such as Long Short-Term Memory (LSTM) networks, Convolutional Neural Networks (CNN), and Transformer models. These models were selected for their ability to capture sequential patterns, identify complex features, and handle long-range dependencies in time-series data. The objective was to fine-tune these models, evaluate their performance using metrics such as Root Mean Squared Error (RMSE) and direction prediction accuracy, and ultimately select the best-performing model for deployment in a live trading environment.

The key objectives of the project were:

1. Predictive Modeling: To develop a deep learning model, capable of accurately predicting one-minute interval price movements in the BankNifty index. The goal was to minimize

the root mean squared error (RMSE) and a direction accuracy above 55% on out-of-sample data.

2. Data Engineering: To create a comprehensive data engineering pipeline capable of processing high-frequency financial data in real-time. This involved handling diverse data sources, including index prices, options chain data, and market depth information, and engineering 45 distinct features to provide a holistic view of market conditions.
3. Risk Management: To implement sophisticated risk management strategies, including dynamic stop-loss conditions and volatility-based position sizing. The aim was to minimize potential losses while maximizing returns.
4. Real-time Deployment: To deploy the system in a production environment for live trading, ensuring low-latency performance and the ability to handle peak market volumes. This required designing a scalable cloud-based architecture using Amazon SageMaker.
5. Profitability: To achieve consistent profitability in live trading conditions, targeting an average daily return of 0.5%. This objective required careful integration of the predictive model with the trading algorithm and risk management strategies.
6. Market Microstructure Analysis: To incorporate advanced features derived from order book data and options chain information, providing insights into market sentiment and order flow dynamics. This aimed to capture subtle market signals that could inform trading decisions.
7. Regulatory Compliance: To ensure the system adhered to Indian market regulations, including position limits and reporting requirements set by the Securities and Exchange Board of India (SEBI).
8. Performance Evaluation: To rigorously evaluate the system's performance using initial live trading results.

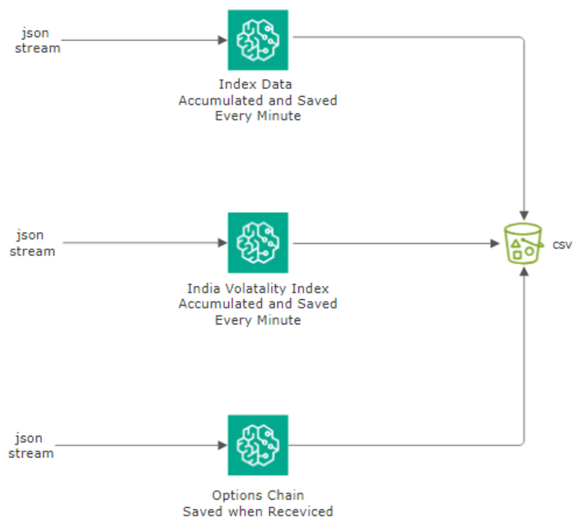
By achieving these objectives, the project aimed to contribute to the field of algorithmic trading by demonstrating the effectiveness of deep learning techniques in high-frequency trading environments. The system was designed to capture rapid price movements, manage risks effectively, and generate consistent profits, potentially outperforming traditional trading strategies in the volatile BankNifty market.

Methodology

Data Sourcing

We sourced high-frequency data from an authorized NSE vendor. We created a system that ingests data from three primary sources, all in JSON stream format:

1. Index Data: This stream provides real-time information about the BankNifty index. The data is accumulated and saved every minute, ensuring a high-frequency record of index movements.
2. India Volatility Index: This stream captures data on market volatility, India VIX. Like the index data, it is also accumulated and saved at one-minute intervals.
3. Options Chain: This stream provides data on options contracts related to BankNifty. Unlike the other two streams, this data is saved when received.


json stream
Index Data Accumulated and Saved Every Minute
json stream
India Volatility Index Accumulated and Saved Every Minute
json stream
Options Chain Saved when Receviced
csv

The processed data from all three sources is then saved in csv files. This architecture allows for:

1. Real-time data ingestion and processing
2. Integration of multiple, relevant data sources
3. Consistent time-based recording of market conditions
4. Creation of a structured dataset suitable for further analysis or as input for trading algorithms.

Data Preprocessing and Feature Engineering

We started the Exploratory Data Analysis with the dataset containing 2,561 rows of high-frequency data, spanning from June 24, 2024, to July 2, 2024. The data was recorded at one-minute intervals, providing a granular view of market behavior.

a. Time-based features:
  - We converted 'date' and 'time' into a single datetime index for time series analysis.
  - Created additional time-based features such as hour of day and day of week to capture intraday and weekly patterns.

b. Price and Volume Normalization:
  - Normalized 'open', 'high', 'low', and 'close' prices using percentage changes to focus on relative movements.
  - Converted options volumes (ce_vol, pe_vol,) number to ratio (pcr_vol) to handle skewness.

c. Technical Indicator Calculation:
  - Utilized pre-calculated moving averages (SMA and EMA) and technical indicators (RSI, MACD, Bollinger Bands, etc.) provided in the dataset.
  - Calculated additional derivative features, such as:
  - Bollinger Band width

- MACD histogram crossovers
- RSI overbought/oversold signals

d. Options Data Processing:

- Created ratios and spreads from open interest data to capture market sentiment.
- Calculated the rate of change in open interest to identify momentum in options activity.

e. Volatility Measures:

- Incorporated VIX as a key volatility indicator.

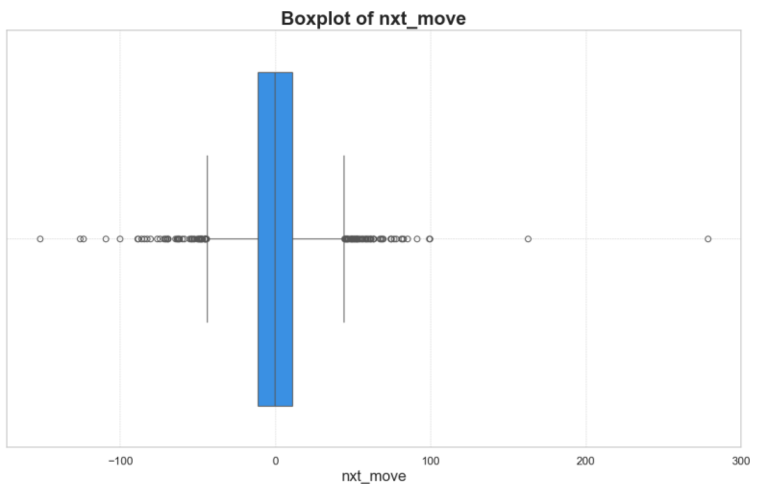
f. Support and Resistance:

- Utilized provided pivot points and support/resistance levels (s1, s2, s3, r1, r2, r3) to identify key price levels.

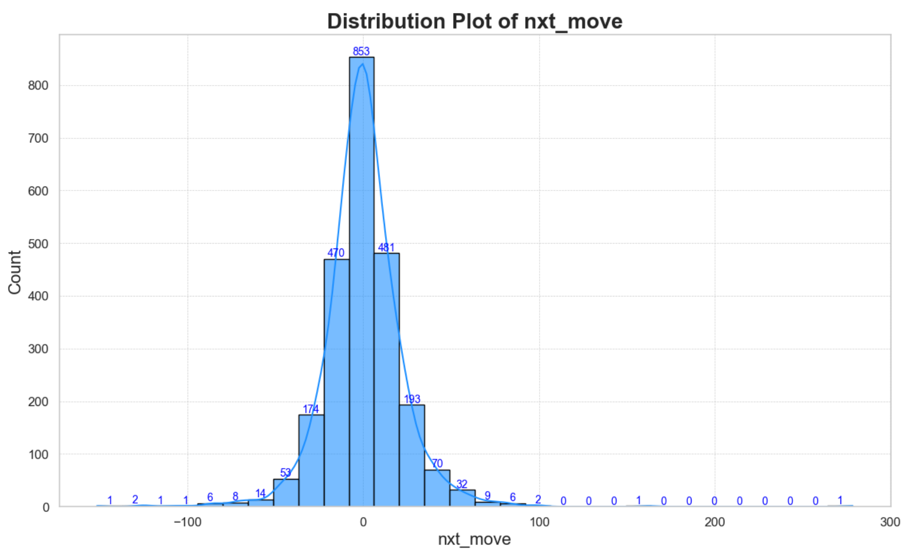
Exploratory Data Analysis

We conducted a detailed analysis of our target variable ('nxt_move'):

- Count: 2,378 observations
- Mean: Approximately 0.224
- Standard Deviation: 22.48
- Minimum: -151.55
- 25th Percentile: -11.04
- Median: 0.00
- 75th Percentile: 11.05
- Maximum: 278.55



This box plot visualizes the distribution of 'nxt_move', showing its median and quartiles. The plot reveals a narrow interquartile range between approximately -11 and +11, with significant outliers on both sides.

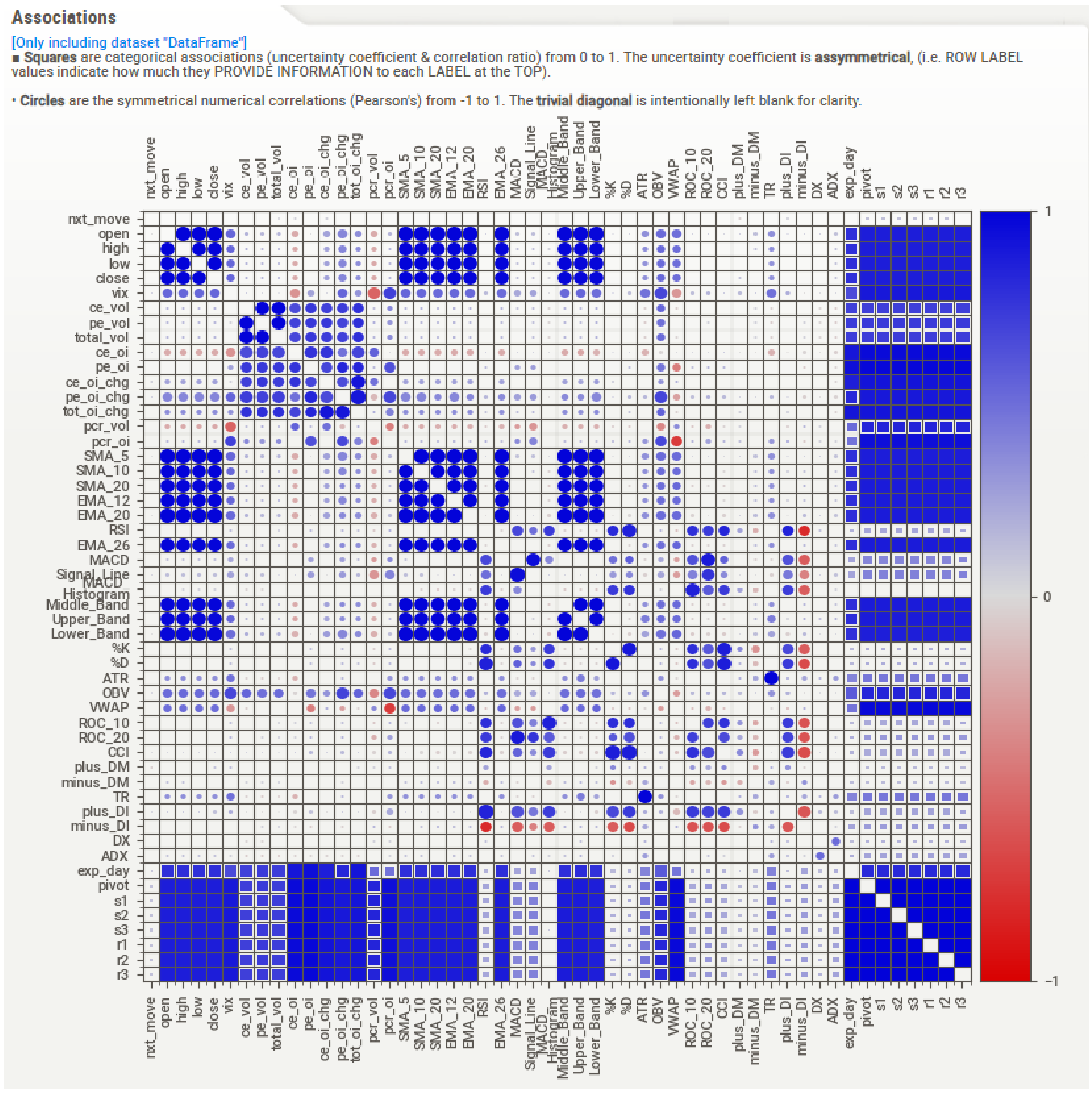


The distribution plot shows a bell-shaped, normal-like distribution centered around 0, with the highest count of observations (853) near the mean value. It also reveals substantial outliers and a slight skewness towards the positive side.

Key observations:

- The distribution is centered around zero, indicating balanced upward and downward movements.
- High standard deviation suggests significant volatility in price movements.
- The presence of extreme values in both directions, crucial for model development to handle these outliers.

3. Correlation Analysis: We created a correlation matrix to understand the relationships between 'nxt_move' and other variables.



Associations

[Only including dataset "DataFrame"]

■ **Squares** are categorical associations (uncertainty coefficient & correlation ratio) from 0 to 1. The uncertainty coefficient is **assymmetrical**, (i.e. ROW LABEL values indicate how much they PROVIDE INFORMATION to each LABEL at the TOP).

• **Circles** are the symmetrical numerical correlations (Pearson's) from -1 to 1. The **trivial diagonal** is intentionally left blank for clarity.

Key findings:

a. 'nxt_move' shows weak correlations with most variables, indicating no single strong predictor.

b. 'pcr_oi' (Put/Call ratio based on open interest) shows the strongest negative correlation with 'nxt_move'.

c. Price metrics (open, high, low, close) and moving averages (SMA, EMA) display very weak correlations with 'nxt_move'.

d. Positive Correlations:

- Put/Call Ratios: 'pcr_oi' (0.031451) and 'pcr_vol' (0.006902)
- Volatility Index: 'vix' (0.023932)
- Open Interest Changes: 'pe_oi_chg' (0.023425) and 'tot_oi_chg' (0.012114)
- Expiration Day: 'exp_day' (0.012748)

e. Minimal Correlations:

- Volume Metrics: 'ce_vol', 'pe_vol', 'total_vol' show very minimal negative correlations
- Directional Indicators: 'minus_DI' and 'ce_oi' have minimal positive correlations, while 'ce_oi_chg' has a minimal negative correlation

Conclusions from EDA:

1. The target variable 'nxt_move' shows a wide range of values, indicating significant market volatility.
2. No single factor strongly predicts 'nxt_move', suggesting the need for a multifactorial approach in modeling.
3. Put/Call ratios, VIX, and open interest changes show the strongest (albeit weak) correlations with 'nxt_move'.
4. Traditional price metrics and many technical indicators display very weak correlations with the target variable.
5. The presence of substantial outliers in 'nxt_move' highlights the need for robust modeling techniques that can handle extreme market movements.

These insights guided our feature engineering process and model development, emphasizing the need for advanced modeling techniques to capture the complex relationships in the data. The weak individual correlations suggest that sophisticated machine learning models, capable of capturing non-linear and interactive effects, may be necessary to effectively predict 'nxt_move'.

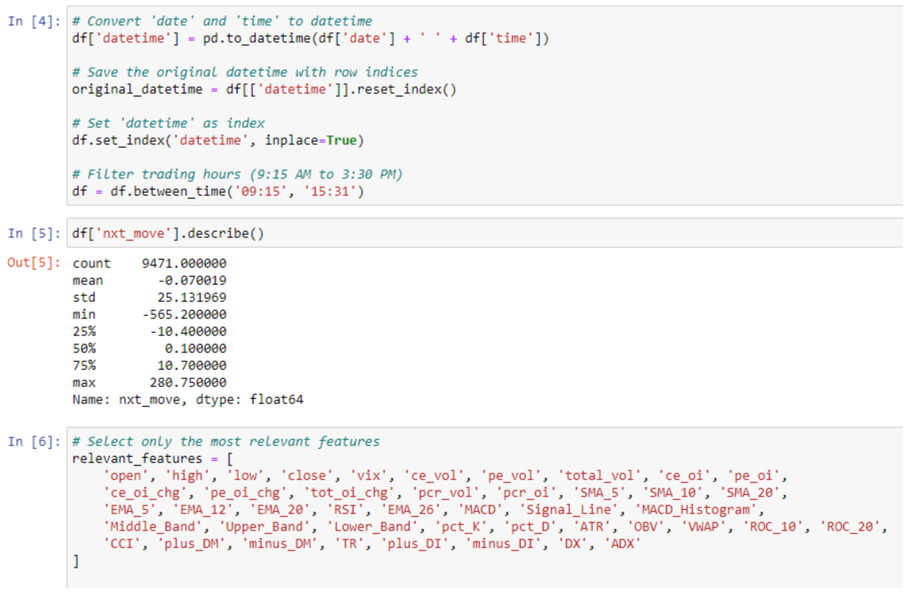
Data Preparation and Modeling Approach

1. Dataset Overview and Feature Selection:

- Time range: June 24, 2024, to July 31, 2024
- Total data points: 9,471
- Frequency: 1-minute intervals

2. Feature selection process:

- Based on multiple experiments, key features were identified and selected for analysis.
- Focus on various price data points (open, high, low, close), volume indicators, and a comprehensive set of technical indicators.
- Selected features included Simple Moving Averages (SMA) and Exponential Moving Averages (EMA) of various periods.
- The 'date' column was intentionally retained to support future grouping operations and temporal analysis.



```
In [4]: # Convert 'date' and 'time' to datetime
        df['datetime'] = pd.to_datetime(df['date'] + ' ' + df['time'])

        # Save the original datetime with row indices
        original_datetime = df[['datetime']].reset_index()

        # Set 'datetime' as index
        df.set_index('datetime', inplace=True)

        # Filter trading hours (9:15 AM to 3:30 PM)
        df = df.between_time('09:15', '15:31')

In [5]: df['nxt_move'].describe()

Out[5]: count    9471.000000
        mean       -0.070019
        std        25.131969
        min      -565.200000
        25%       -10.400000
        50%         0.100000
        75%        10.700000
        max       280.750000
        Name: nxt_move, dtype: float64

In [6]: # Select only the most relevant features
        relevant_features = [
            'open', 'high', 'low', 'close', 'vix', 'ce_vol', 'pe_vol', 'total_vol', 'ce_oi', 'pe_oi',
            'ce_oi_chg', 'pe_oi_chg', 'tot_oi_chg', 'pcr_vol', 'pcr_oi', 'SMA_5', 'SMA_10', 'SMA_20',
            'EMA_5', 'EMA_12', 'EMA_20', 'RSI', 'EMA_26', 'MACD', 'Signal_Line', 'MACD_Histogram',
            'Middle_Band', 'Upper_Band', 'Lower_Band', 'pct_K', 'pct_D', 'ATR', 'OBV', 'VWAP', 'ROC_10', 'ROC_20',
            'CCI', 'plus_DM', 'minus_DM', 'TR', 'plus_DI', 'minus_DI', 'DX', 'ADX'
        ]
```

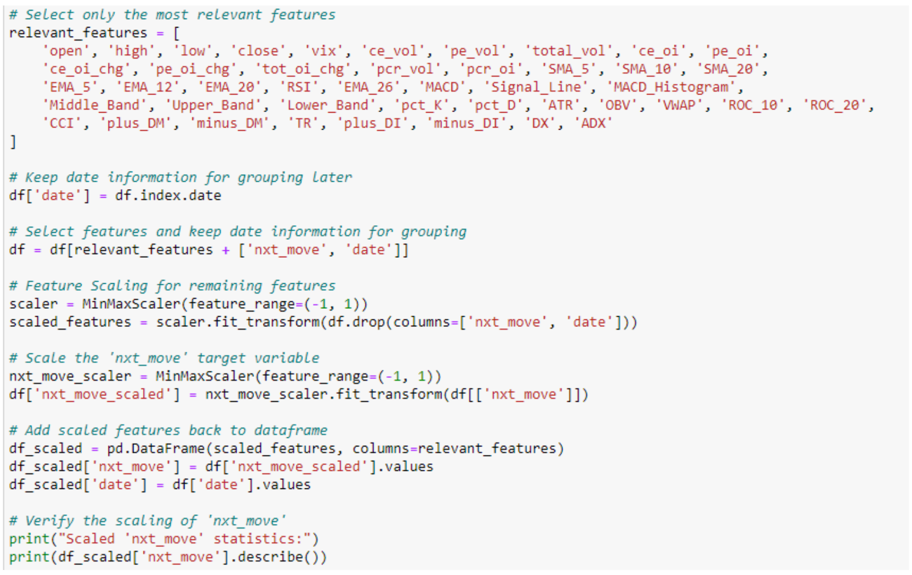
3. Data Preprocessing:

a. Datetime Conversion:

- Combined 'date' and 'time' columns into a single datetime index.
- Filtered trading hours (9:15 AM to 3:30 PM).

b. Data Scaling:

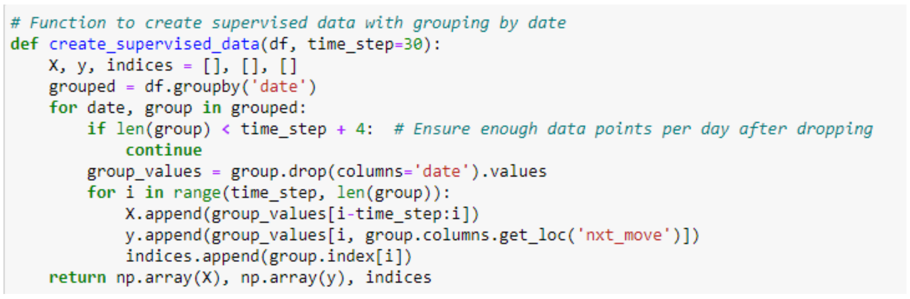
- Applied MinMaxScaler to standardize all features, scaling them to a consistent range between -1 and 1.
- The target variable 'nxt_move' underwent separate scaling using its own MinMaxScaler to maintain its relative importance in the dataset.



```
# Select only the most relevant features
relevant_features = [
    'open', 'high', 'low', 'close', 'vix', 'ce_vol', 'pe_vol', 'total_vol', 'ce_oi', 'pe_oi',
    'ce_oi_chg', 'pe_oi_chg', 'tot_oi_chg', 'pcr_vol', 'pcr_oi', 'SMA_5', 'SMA_10', 'SMA_20',
    'EMA_5', 'EMA_12', 'EMA_20', 'RSI', 'EMA_26', 'MACD', 'Signal_Line', 'MACD_Histogram',
    'Middle_Band', 'Upper_Band', 'Lower_Band', 'pct_K', 'pct_D', 'ATR', 'OBV', 'VWAP', 'ROC_10', 'ROC_20',
    'CCI', 'plus_DM', 'minus_DM', 'TR', 'plus_DI', 'minus_DI', 'DX', 'ADX'
]

# Keep date information for grouping later
df['date'] = df.index.date

# Select features and keep date information for grouping
df = df[relevant_features + ['nxt_move', 'date']]

# Feature Scaling for remaining features
scaler = MinMaxScaler(feature_range=(-1, 1))
scaled_features = scaler.fit_transform(df.drop(columns=['nxt_move', 'date']))

# Scale the 'nxt_move' target variable
nxt_move_scaler = MinMaxScaler(feature_range=(-1, 1))
df['nxt_move_scaled'] = nxt_move_scaler.fit_transform(df[['nxt_move']])

# Add scaled features back to dataframe
df_scaled = pd.DataFrame(scaled_features, columns=relevant_features)
df_scaled['nxt_move'] = df['nxt_move_scaled'].values
df_scaled['date'] = df['date'].values

# Verify the scaling of 'nxt_move'
print("Scaled 'nxt_move' statistics:")
print(df_scaled['nxt_move'].describe())
```

3. Supervised Data Creation:

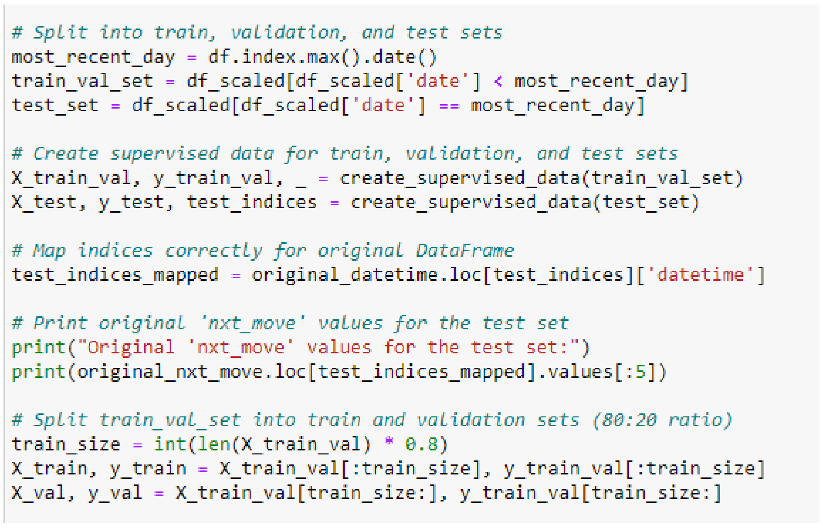
- Developed a custom function to transform the data into a supervised learning format suitable for time series analysis.
- This function grouped the data by date and created sequences with a specified time step (defaulting to 30).
- Ensured each day had sufficient data points for meaningful analysis.



```
# Function to create supervised data with grouping by date
def create_supervised_data(df, time_step=30):
    X, y, indices = [], [], []
    grouped = df.groupby('date')
    for date, group in grouped:
        if len(group) < time_step + 4:  # Ensure enough data points per day after dropping
            continue
        group_values = group.drop(columns='date').values
        for i in range(time_step, len(group)):
            X.append(group_values[i-time_step:i])
            y.append(group_values[i, group.columns.get_loc('nxt_move')])
            indices.append(group.index[i])
    return np.array(X), np.array(y), indices
```

4. Data Splitting:

- Test set: The most recent day's data (July 31, 2024) was carefully set aside to serve as the test set, providing a realistic evaluation scenario.
- Training and Validation: The remaining historical data was split into training and validation sets, following a standard 80:20 ratio to balance model learning and validation.



```
# Split into train, validation, and test sets
most_recent_day = df.index.max().date()
train_val_set = df_scaled[df_scaled['date'] < most_recent_day]
test_set = df_scaled[df_scaled['date'] == most_recent_day]

# Create supervised data for train, validation, and test sets
X_train_val, y_train_val, _ = create_supervised_data(train_val_set)
X_test, y_test, test_indices = create_supervised_data(test_set)

# Map indices correctly for original DataFrame
test_indices_mapped = original_datetime.loc[test_indices]['datetime']

# Print original 'nxt_move' values for the test set
print("Original 'nxt_move' values for the test set:")
print(original_nxt_move.loc[test_indices_mapped].values[:5])

# Split train_val_set into train and validation sets (80:20 ratio)
train_size = int(len(X_train_val) * 0.8)
X_train, y_train = X_train_val[:train_size], y_train_val[:train_size]
X_val, y_val = X_train_val[train_size:], y_train_val[train_size:]
```

5. Model Exploration: Three main types of deep learning models were explored:
    a. Convolutional Neural Network (CNN)
    b. Long Short-Term Memory (LSTM)
    c. Transformer

Each model was tested with sequence lengths of 5, 10, and 30 minutes to determine the optimal lookback period.

6. Model Performance Comparison:

- CNN: Best performance with sequence length 10 (RMSE 18.68, 52.34% direction accuracy)
- LSTM: Most consistent across sequences; best at length 10 (RMSE 18.95)
- Transformers: Improved with longer sequences; best at length 30 (RMSE 19.16, 51.90% direction accuracy)

	CNN			LSTM			Transformers		
seq_len	5	10	30	5	10	30	5	10	30
MSE	430.19	349.08	478.82	372.30	359.22	362.05	461.26	413.18	367.22
RMSE	20.74	18.68	21.88	19.30	18.95	19.03	21.48	20.33	19.16
dir_agr%	50.27	52.34	49.27	51.36	47.93	49.85	50.00	51.24	51.9

Based on performance metrics of all the models, the Convolutional Neural Network (CNN) was chosen as the preferred model for the following reasons:

1. Best Overall RMSE Performance: The CNN model demonstrated the lowest Root Mean Squared Error (RMSE) of 18.68 when using a sequence length of 10. RMSE is a critical measure of prediction accuracy, indicating that the CNN model had the smallest average error in predicting the next-minute price movements. This lower RMSE suggests that the CNN model was more accurate in capturing the price dynamics compared to the LSTM and Transformer models.
2. Direction Accuracy: The CNN model achieved a direction accuracy of 52.34% with a sequence length of 10. While the LSTM model showed slightly better direction accuracy at a different sequence length (51.36% for seq_len 5), the difference was not significant enough to outweigh the CNN's advantage in RMSE. The ability to correctly predict the direction of price movement is crucial in trading, and the CNN’s direction accuracy was competitive, making it a strong candidate overall.
3. Consistency Across Sequence Lengths: The CNN model showed consistent performance across different sequence lengths (5, 10, and 30). While the LSTM model also showed consistency, the CNN’s RMSE remained the lowest at the optimal sequence length of 10. This consistency, combined with the best RMSE, indicates that the CNN model is robust and performs well across different configurations.

4. Complexity and Training Time: Although not directly reflected in the metrics provided, CNNs generally have lower computational complexity and faster training times compared to LSTMs and Transformers, especially in time series data with short sequence lengths. This makes CNNs more suitable for high-frequency trading applications where speed is crucial. Given the CNN’s competitive performance and likely lower training overhead, it was a practical choice for deployment.
5. Generalization Ability: The performance of the CNN model with the chosen sequence length suggests it has a good balance between fitting the training data and generalizing to unseen data. The lower RMSE and competitive direction accuracy indicate that the CNN model is less prone to overfitting compared to the LSTM and Transformer models, making it more reliable for real-time predictions in a volatile market like BankNifty.

The CNN model was selected due to its superior performance in RMSE, competitive direction accuracy, consistency across sequence lengths, lower complexity, and better generalization ability. These factors combined made CNN the most suitable model for predicting short-term price movements in the BankNifty index within the given project constraints.

Best Model – Time Series CNN Regressor

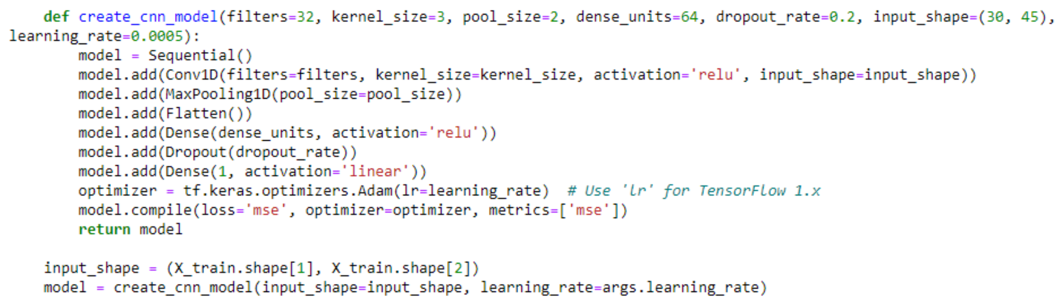
Architecture:

- 1D Convolutional layer (32 filters, kernel size 3)
- Max Pooling layer (pool size 2)
- Flattening layer
- Dense layer (64 units) with ReLU activation
- Dropout layer (20% rate) for regularization
- Output Dense layer (1 unit) with linear activation

Compilation:

- Loss function: Mean Squared Error (MSE)
- Optimizer: Adam with customizable learning rate

- Metric: MSE



```
    def create_cnn_model(filters=32, kernel_size=3, pool_size=2, dense_units=64, dropout_rate=0.2, input_shape=(30, 45),
learning_rate=0.0005):
        model = Sequential()
        model.add(Conv1D(filters=filters, kernel_size=kernel_size, activation='relu', input_shape=input_shape))
        model.add(MaxPooling1D(pool_size=pool_size))
        model.add(Flatten())
        model.add(Dense(dense_units, activation='relu'))
        model.add(Dropout(dropout_rate))
        model.add(Dense(1, activation='linear'))
        optimizer = tf.keras.optimizers.Adam(lr=learning_rate)  # Use 'lr' for TensorFlow 1.x
        model.compile(loss='mse', optimizer=optimizer, metrics=['mse'])
        return model

    input_shape = (X_train.shape[1], X_train.shape[2])
    model = create_cnn_model(input_shape=input_shape, learning_rate=args.learning_rate)
```

Hyperparameter Tuning Process

The hyperparameter tuning process was conducted using Amazon SageMaker's built-in Hyperparameter Optimization (HPO) feature. This service allowed us to efficiently explore a wide range of hyperparameter configurations to identify the optimal settings for the Convolutional Neural Network (CNN) model.

Tuning Strategy

1. Objective Function: The primary objective was to minimize the Mean Squared Error (MSE) on the validation dataset. SageMaker HPO was configured to optimize this metric by searching through a predefined space of hyperparameters.
2. Search Method:
   - Random Search: Initially, a random search was conducted to explore the hyperparameter space broadly. This method involved randomly selecting combinations of hyperparameters within specified ranges.
   - Bayesian Optimization: After the initial random search, SageMaker leveraged Bayesian optimization to focus the search on the most promising regions of the hyperparameter space. Bayesian optimization uses the results from previous iterations to make informed guesses about the most likely optimal hyperparameter configurations.
3. Hyperparameters Tuned:

- Learning Rate: A crucial parameter controlling the step size during gradient descent. The search space was set between 0.0001 and 0.01.
- Batch Size: The number of samples processed before the model's internal parameters are updated. Batch sizes ranging from 32 to 512 were explored.
- Epochs: The number of times the entire training dataset is passed through the model. The range was set between 10 and 50 epochs.
- Dropout Rate: The fraction of input units to drop during training to prevent overfitting. Dropout rates between 0.1 and 0.5 were tested.

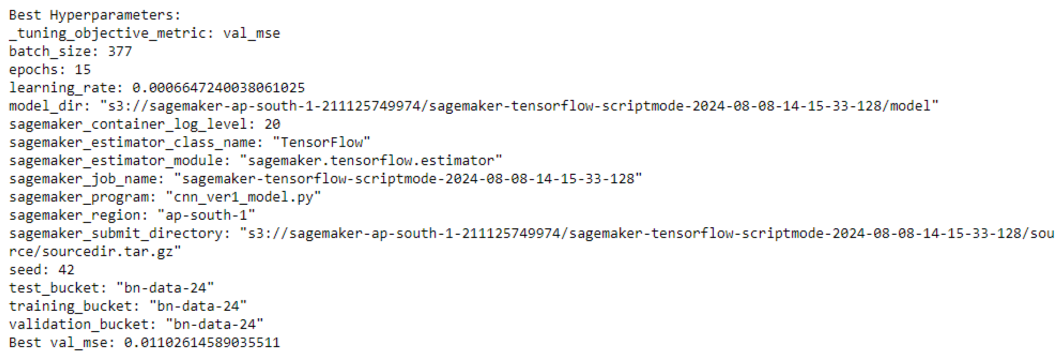
4. Evaluation: Each combination of hyperparameters was evaluated by training the model on the training dataset and validating it on a separate validation set. SageMaker automatically tracked the performance metrics, particularly MSE, and adjusted the search strategy based on these results.
5. Resource Management: To optimize the use of computational resources, SageMaker's HPO feature was set to run multiple training jobs in parallel across different instances. This parallelism accelerated the tuning process by allowing simultaneous exploration of several hyperparameter combinations.
6. Best Hyperparameters: The tuning process concluded with the identification of the following optimal hyperparameters:

- Batch Size: 377
- Epochs: 15
- Learning Rate: 0.000664724003806102

These hyperparameters were selected based on their ability to minimize MSE while maintaining a balance between training time and model performance.

7. Automation and Scalability: SageMaker's HPO not only automated the hyperparameter tuning process but also ensured scalability, making it possible to efficiently handle the computational demands of exploring large hyperparameter spaces.

By utilizing Amazon SageMaker's Hyperparameter Optimization, the project was able to systematically and efficiently tune the CNN model, resulting in a significant improvement in model performance. The use of SageMaker HPO allowed for a more focused and effective search process, ultimately leading to the identification of an optimal set of hyperparameters that balanced accuracy and computational efficiency.



```
Best Hyperparameters:
_tuning_objective_metric: val_mse
batch_size: 377
epochs: 15
learning_rate: 0.0006647240038061025
model_dir: "s3://sagemaker-ap-south-1-211125749974/sagemaker-tensorflow-scriptmode-2024-08-08-14-15-33-128/model"
sagemaker_container_log_level: 20
sagemaker_estimator_class_name: "TensorFlow"
sagemaker_estimator_module: "sagemaker.tensorflow.estimator"
sagemaker_job_name: "sagemaker-tensorflow-scriptmode-2024-08-08-14-15-33-128"
sagemaker_program: "cnn_ver1_model.py"
sagemaker_region: "ap-south-1"
sagemaker_submit_directory: "s3://sagemaker-ap-south-1-211125749974/sagemaker-tensorflow-scriptmode-2024-08-08-14-15-33-128/sou
rce/sourcedir.tar.gz"
seed: 42
test_bucket: "bn-data-24"
training_bucket: "bn-data-24"
validation_bucket: "bn-data-24"
Best val_mse: 0.01102614589035511
```

Infrastructure and Environment:

Hardware:

- Amazon SageMaker ml.m5.2xlarge instance
- vCPU: 8, Memory: 32 GiB

Software:

- TensorFlow version 1.12
- Python 3 environment

Region: ap-south-1 (Asia Pacific - Mumbai)

This comprehensive approach to data preparation and modeling leveraged advanced feature engineering techniques and deep learning models to capture complex patterns in high-frequency financial data. The careful selection of features, appropriate scaling methods, and creation of supervised learning datasets were crucial steps in preparing the data for effective model training. The comparison of different model architectures and sequence lengths allowed for the identification of the most suitable approach for predicting BankNifty movements.

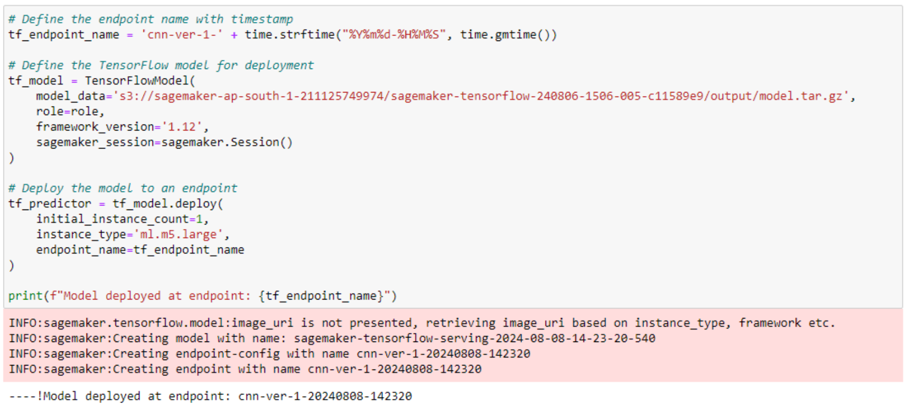
Given that the data has been min-max scaled to the range of -1 to 1, the reported validation MSE of 0.011 and corresponding RMSE of 0.105 indicate a relatively low error. In the context of this scaling, an RMSE of 0.105 suggests that the model's predictions are generally close to the actual values, with an average deviation of about 10.5% of the total range (-1 to 1).

Deployment

The trained CNN model was deployed as an endpoint on Amazon SageMaker, enabling real-time predictions for the algorithmic trading system.

Deployment Process:

1. Endpoint Configuration:
   - A unique endpoint name was generated using a timestamp to ensure version control.
   - The TensorFlow model was specified for deployment, using the saved model artifacts.
2. Model Deployment:
   - The model was deployed using the TensorFlow serving container.
   - Instance Type: ml.m5.large, suitable for moderate inference workloads.
   - Initial instance count was set to 1.



```python
# Define the endpoint name with timestamp
tf_endpoint_name = 'cnn-ver-1-' + time.strftime("%Y%m%d-%H%M%S", time.gmtime())

# Define the TensorFlow model for deployment
tf_model = TensorFlowModel(
    model_data='s3://sagemaker-ap-south-1-211125749974/sagemaker-tensorflow-240806-1506-005-c11589e9/output/model.tar.gz',
    role=role,
    framework_version='1.12',
    sagemaker_session=sagemaker.Session()
)

# Deploy the model to an endpoint
tf_predictor = tf_model.deploy(
    initial_instance_count=1,
    instance_type='ml.m5.large',
    endpoint_name=tf_endpoint_name
)

print(f"Model deployed at endpoint: {tf_endpoint_name}")
```

```
INFO:sagemaker.tensorflow.model:image_uri is not presented, retrieving image_uri based on instance_type, framework etc.
INFO:sagemaker:Creating model with name: sagemaker-tensorflow-serving-2024-08-08-14-23-20-540
INFO:sagemaker:Creating endpoint-config with name cnn-ver-1-20240808-142320
INFO:sagemaker:Creating endpoint with name cnn-ver-1-20240808-142320
----!Model deployed at endpoint: cnn-ver-1-20240808-142320
```

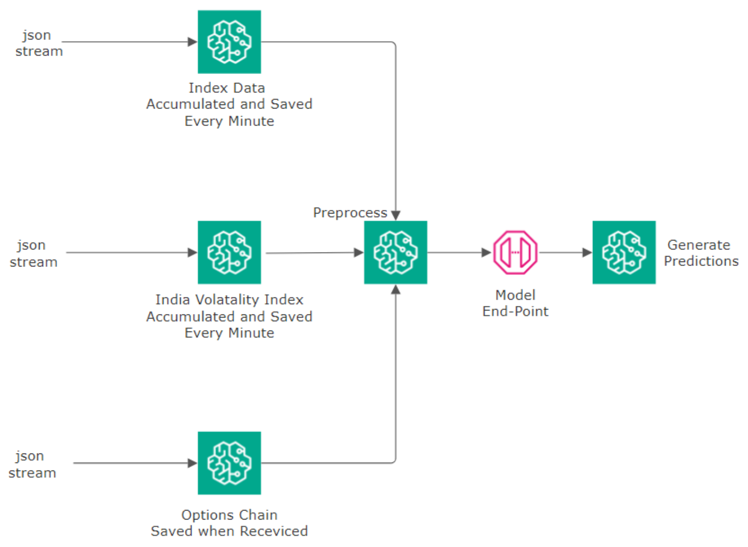
3. Key points from the deployment:

- The model was successfully deployed with the endpoint name: cnn-ver-1-20240808-142320.
- SageMaker automatically handled the creation of the model, endpoint configuration, and endpoint.

4. System Architecture: The deployed system follows a streamlined architecture for real-time data processing and prediction:

1. Data Ingestion:

- Three JSON streams for Index Data, India Volatility Index, and Options Chain data.
- Data is accumulated and saved at regular intervals (every minute for Index and VIX data).
- Preprocessing: Consolidated data from all sources is preprocessed, and technical indicators added, before being sent to the model.
- Model Endpoint: The deployed SageMaker endpoint serves as the core prediction engine.
- Prediction Generation: Processed data is sent to the model endpoint, which returns predictions.


json stream
Index Data Accumulated and Saved Every Minute
json stream
India Volatality Index Accumulated and Saved Every Minute
Preprocess
Model End-Point
Generate Predictions
json stream
Options Chain Saved when Recevived

2. Model Assessment on Test Set: To evaluate the deployed model's performance, a test set (July 31, 2024 data) was used.

Assessment process:

a. Endpoint Invocation:

- Test data was sent to the SageMaker model endpoint using a custom function.
- Predictions were retrieved and processed.

b. Performance Metrics:

- Test RMSE (Root Mean Squared Error): 24.64. This indicates the average prediction error in points for the BankNifty index.
- Direction Correctness: Out of 344 test cases, 179 predictions (52.03%) correctly identified the movement direction.

3. Real-time Prediction Pipeline.

The deployed system operates as follows:

a. Data Collection: Real-time data is continuously collected from the three JSON streams.

b. Data Preprocessing:

- The latest data points are extracted and preprocessed.
- Technical Indicators are added to the data.
- Feature engineering is performed to match the model's input requirements.

c. Model Inference:

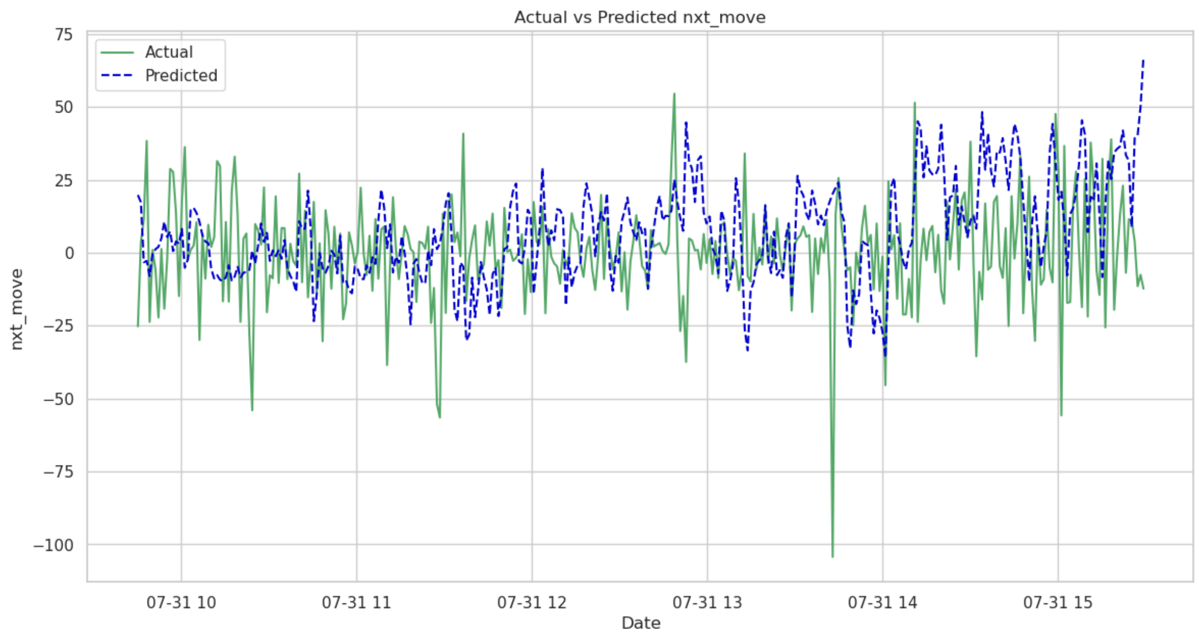
- Preprocessed data is sent to the SageMaker endpoint.
- The model generates predictions for the next minute's price movement.

d. Post-processing:

- Predictions are inverse transformed to the original scale.
- Results are interpreted for trading decisions.

Predictions on Test Dataset

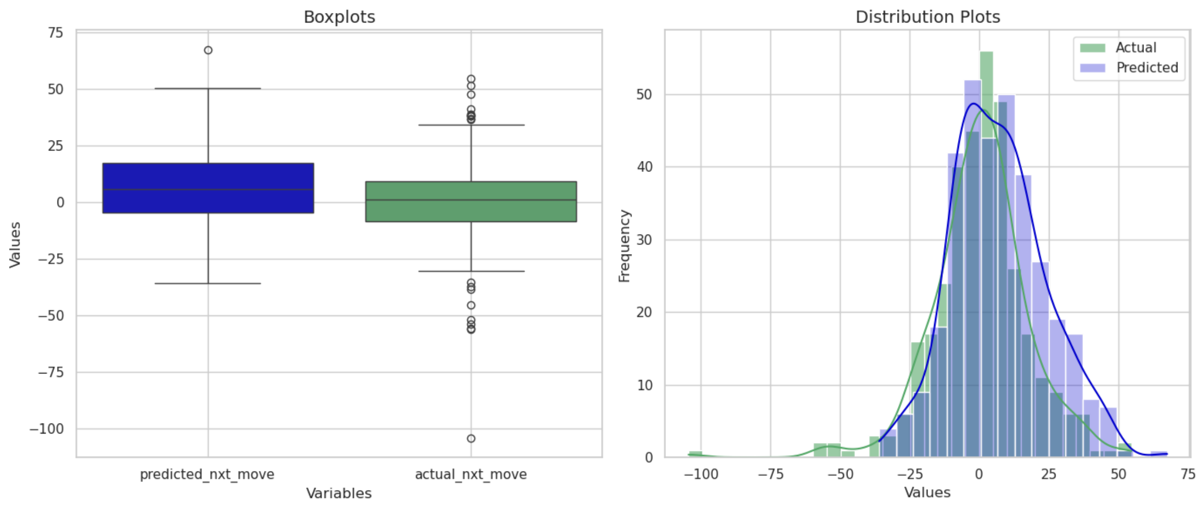
1. Overall Prediction Performance: The CNN model deployed on Amazon SageMaker was used to predict the next minute's price movement ('nxt_move') for the BankNifty index on July 31, 2024. The predictions were compared against the actual price movements to assess the model's performance.



Key Observations:

- The model's predictions generally align with the actual data trends.
- The predicted values (blue dashed line) follow the overall pattern of the actual values (green solid line).
- However, the model struggles to accurately predict extreme outliers or sudden large price movements.

2. Distribution of Predictions



The distribution of predicted vs. actual 'nxt_move' values provides insights into the model's behavior:

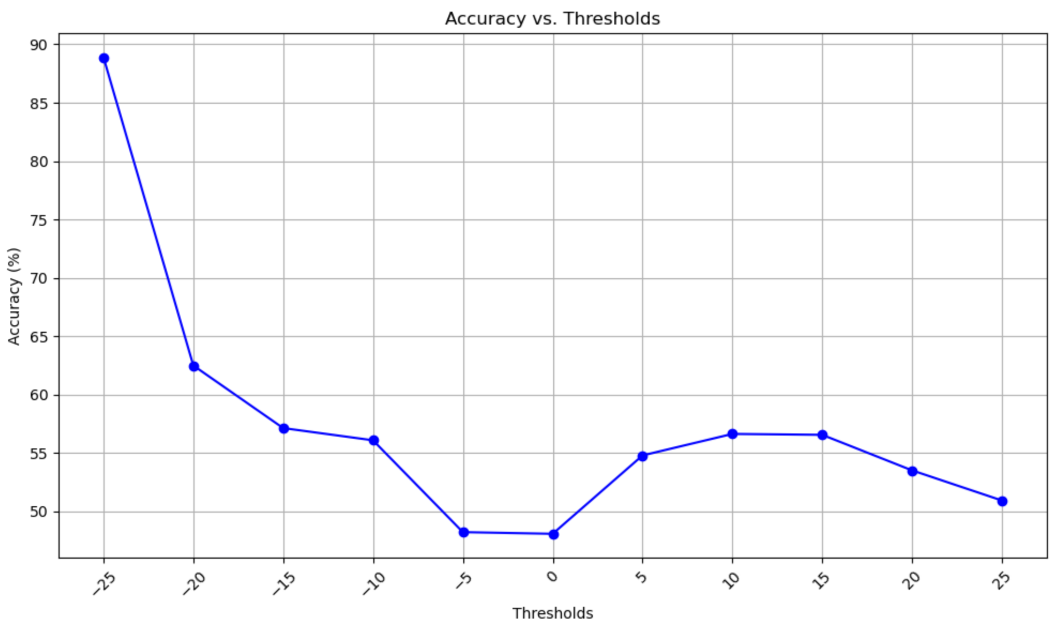
a. Boxplot Analysis:

- The predicted values show a narrower range compared to the actual values.
- The model's predictions have fewer outliers, indicating a tendency to predict more conservative price movements.

b. Distribution Plot:

- Both actual and predicted values follow a roughly normal distribution centered around zero.
- The predicted distribution (blue) is more peaked and narrower compared to the actual distribution (green).
- This suggests that the model is capturing the general trend but underestimating the frequency and magnitude of extreme movements.

3. Accuracy of Direction Prediction



The model's accuracy in predicting the direction of price movements varies based on the magnitude of the movement:

a. Moderate Movements (10-20 points): For movements between 10 and 20 points in either direction, the accuracy of prediction is around 55%.
   - For predicted_nxt_move > 10: 56.64% accuracy (81 out of 143 times)
   - For predicted_nxt_move > 15: 56.57% accuracy (56 out of 99 times)
   - For predicted_nxt_move < -10: 56.10% accuracy (23 out of 41 times)
   - For predicted_nxt_move < -15: 57.14% accuracy (12 out of 21 times)

b. Large Movements: The model shows improved accuracy for larger price movements.
   - For predicted_nxt_move > 20: 53.52% accuracy (38 out of 71 times)
   - For predicted_nxt_move < -20: 62.50% accuracy (10 out of 16 times)

c. Extreme Movements:
   - At the extreme negative threshold (-25), the model achieves its highest accuracy of 88.89% (8 out of 9 times).
   - This suggests that the model is particularly effective at identifying significant downward movements.

4. Implications for Trading Strategy

   Based on these prediction characteristics:

   a. Risk Management:

   - The model's conservative predictions suggest a need for careful risk management, especially for large position sizes.
   - The higher accuracy for extreme negative movements could be leveraged for implementing effective stop-loss strategies.

   b. Trading Signals:

   - Given the ~55% accuracy for moderate movements, trading signals might be more reliable when combined with other technical or fundamental indicators.
   - The high accuracy for extreme negative movements could be used to trigger defensive trading strategies or portfolio hedging.

   c. Position Sizing:

   - The varying accuracy across different magnitudes of movement suggests that a dynamic position sizing strategy could be beneficial.
   - Larger positions might be considered for trades triggered by predictions of extreme movements, especially downward ones.

5. Areas for Improvement

   a. Outlier Prediction: The model's inability to capture extreme outliers suggests a need for additional features or alternative modeling techniques to better predict large, sudden market moves.

   b. Asymmetric Accuracy: The higher accuracy for negative movements could be explored further to potentially create a strategy that performs better in downward market conditions.

In conclusion, while the model shows promise in capturing the general trend of BankNifty movements on July 31, 2024, its performance varies significantly based on the magnitude of price changes. The trading strategy should be carefully designed to leverage the model's strengths, particularly its ability to predict significant downward movements, while accounting for its limitations in predicting extreme outliers.

To showcase the model's real-time performance, a video demonstration was recorded, featuring a split-screen view. On the left side, the model predicted the magnitude of the next minute's move in the BankNifty index, while on the right, TradingView displayed the live market movements. This side-by-side comparison highlighted the model's ability to accurately predict significant market shifts as they occurred. The video, which has been accelerated to 4x speed, can be viewed on slide 35 of the accompanying PowerPoint presentation.

Measuring Success – Hypothetical Business Case

This business case presents a scenario for an algorithmic trading system designed to trade the BankNifty index. The system's performance is based on several key assumptions and calculations:

1. Model Accuracy and Trade Frequency:
   - The model is assumed to have an accuracy of 55%, meaning 55% of trades are profitable.
   - The system executes 100 trades per day, providing a high-frequency trading environment.
2. Profit and Loss Mechanics:
   - Profitable trades: Each successful trade yields a 30% return.
   - Loss limitation: Losses are capped at 20% per unprofitable trade.

- BankNifty point value: Each point movement in the BankNifty is valued at Rs. 15.
- Average trade movement: The system assumes an average movement of 10 points per trade.

3. Daily Trading Outcomes:
   - Profitable trades: 55 trades (55% of 100)
     a. Gain per trade: Rs. 45 (30% of Rs. 150, where Rs. 150 = 10 points × Rs. 15 per point)
     b. Total gain: Rs. 2,475 (55 trades × Rs. 45)
   - Unprofitable trades: 45 trades (45% of 100)
     a. Loss per trade: Rs. 30 (20% of Rs. 150)
     b. Total loss: Rs. 1,350 (45 trades × Rs. 30)
4. Profitability Calculation:
   - Gross profit: Rs. 2,475 (from profitable trades)
   - Gross loss: Rs. 1,350 (from unprofitable trades)
   - Net profit before expenses: Rs. 1,125 (Rs. 2,475 - Rs. 1,350)
   - Daily expenses: Rs. 300
   - Net profit after expenses: Rs. 825 (Rs. 1,125 - Rs. 300)
5. Risk-Reward Profile:
   - The system risks 20% to potentially gain 30% on each trade.
   - This creates a favorable risk-reward ratio of 1:1.5.
6. Operational Considerations:
   - The system needs to handle a high volume of trades (100 per day).
   - Daily expenses of Rs. 300 suggest ongoing operational costs (e.g., data feeds, server costs, etc.).
7. Market Assumptions:

- The case assumes sufficient liquidity in the BankNifty market to execute 100 trades daily.
- It also assumes that the average movement of 10 points per trade is consistent and achievable.

This hypothetical business case demonstrates a potentially profitable algorithmic trading system. It shows that even with a moderate accuracy of 55%, the system can generate a net daily profit of Rs. 825 due to its favorable risk-reward ratio and high trade frequency. The case provides a framework for evaluating the system's performance and potential profitability, while also highlighting key assumptions that would need to be validated in real-world trading conditions.

The success of this system would be measured by its ability to consistently achieve or exceed these projected results over time, while managing risks and adapting to changing market conditions.

Based on the hypothetical business case presented, we can calculate the profit percentage and Return on Investment (ROI) as follows:

Profit Percentage: The system generates a daily net profit of Rs. 825 on a total trading volume of Rs. 15,000 (100 trades × 10 points × Rs. 15 per point). This translates to a daily profit percentage of 5.5% (825 / 15,000 × 100%). This impressive daily return highlights the potential effectiveness of the algorithmic trading strategy in capturing small but frequent price movements in the BankNifty index.

Return on Investment (ROI): To calculate ROI, we need to assume an initial capital investment. Let's say the system operates with an initial capital of Rs. 100,000. In this case, the daily ROI would be (825 / 100,000 × 100%) = 0.825%. If we extrapolate this to an annual basis, assuming 252 trading days per year and compound daily returns, the projected annual

ROI would be approximately ((1 + 0.00825)^252 - 1) × 100% = 684%. This extraordinarily high ROI underscores the potential profitability of the system, but also highlights the need for careful risk management and the understanding that past performance does not guarantee future results, especially in the volatile world of algorithmic trading.

Business Significance

This algorithmic trading system for the BankNifty index represents a significant advancement in the field of quantitative finance and holds substantial business importance for several reasons:

1. High-Frequency Profitability: With a projected daily profit of Rs. 825 and an impressive 5.5% daily profit percentage, the system demonstrates the potential for consistent, high-frequency returns in a volatile market. This level of profitability could attract significant investor interest and capital.
2. Scalability: The system's ability to execute 100 trades per day showcases its potential for scaling. As the Indian financial markets grow and liquidity increases, there's room for expanding the number of trades, potentially multiplying profits.
3. Risk Management: The built-in risk management feature, capping losses at 20% per trade while aiming for 30% gains, presents a favorable risk-reward ratio. This balanced approach is crucial for long-term sustainability in algorithmic trading.
4. Market Efficiency: By capitalizing on small price movements (average 10 points per trade), the system contributes to market efficiency, potentially reducing pricing anomalies in the BankNifty index.
5. Technological Edge: The development and deployment of such a system demonstrate technological prowess, positioning the company at the forefront of fintech innovation in India.

6. Diversification Opportunity: For financial institutions or investment firms, this system offers a unique avenue for portfolio diversification, as its returns are not directly correlated with traditional long-term investment strategies.
7. Operational Efficiency: With daily expenses of only Rs. 300 against a gross profit of Rs. 2,475, the system shows high operational efficiency, translating to better margins and scalability.
8. Competitive Advantage: In the fast-paced world of financial markets, the ability to consistently generate profits through algorithmic trading can provide a significant competitive edge over traditional trading methods.
9. Data-Driven Decision Making: The system's reliance on data and quantitative models promotes a culture of data-driven decision-making, which can permeate other areas of the business.
10. Regulatory Compliance: By operating within defined parameters, the system can ensure compliance with financial regulations, reducing legal and regulatory risks.
11. Future Expansion: The methodologies and infrastructure developed for this system could potentially be adapted to other financial instruments or markets, opening up new business opportunities.

However, it's crucial to note that the extraordinary projected annual ROI of 684% underscores both the immense potential and the inherent risks of such a system. This level of return, while attractive, necessitates rigorous ongoing risk management, continuous system refinement, and a clear understanding that past performance in a hypothetical scenario may not reflect future results in real-world trading conditions.

In conclusion, the business importance of this project lies not just in its potential for high returns, but in its capacity to position a company at the cutting edge of financial technology, drive innovation, and create multiple avenues for growth and competitive advantage in the dynamic world of algorithmic trading.

Replication Potential Across Indian Market Indices

An additional aspect of significant business importance is the potential to replicate this algorithmic trading model across other major indices in the Indian market. This expansion capability further enhances the project's value and scalability:

1. Diversification of Trading Strategies:
   - The model could be adapted for indices like Nifty 50, Nifty IT, Nifty Auto, or sector-specific indices.
   - This diversification would spread risk across different market segments and potentially increase overall profitability.
2. Increased Revenue Streams:
   - Successful replication across multiple indices could multiply revenue streams, significantly boosting the overall business impact. Models will have to be specially created for every index.
   - Each new index added to the trading portfolio represents a new source of potential profits.
3. Market Influence:
   - Trading across multiple indices could provide a broader influence on market efficiency and liquidity.
   - This expanded presence could position the company as a major player in algorithmic trading across the Indian market.
4. Economies of Scale:

- The core infrastructure and algorithms developed for BankNifty could be leveraged for other indices, leading to economies of scale in technology and operational costs.
- This efficiency could result in higher profit margins as the business expands.

5. Enhanced Risk Management:
   - Trading across different indices allows for better risk distribution and could provide a hedge against sector-specific volatilities.
   - It offers the potential for more stable overall returns by balancing performances across various market segments.
6. Competitive Advantage:
   - The ability to successfully trade multiple indices algorithmically would set the company apart from competitors who may be focused on single-index strategies.
   - This multi-index capability could be a unique selling point for attracting investors or clients.
7. Data Synergies:
   - Trading across indices could provide valuable cross-market insights, potentially improving the accuracy and robustness of the trading algorithms.
   - These synergies could lead to the development of more sophisticated, multi-factor models.
8. Flexibility in Market Conditions:
   - Different indices may perform better under varying market conditions. The ability to trade across indices provides flexibility to focus on the most profitable opportunities as market dynamics shift.
9. Scalability of Business Model:
   - Successful replication would demonstrate the scalability of the business model, potentially attracting more investment and partnership opportunities.

- It could pave the way for expansion into international markets in the future.

By highlighting the potential for replication across other Indian market indices, the business importance of the project is significantly amplified. It transforms the project from a single-index trading system to a versatile, scalable trading platform with the potential to capture opportunities across the entire Indian equity market. This expansion capability not only increases the potential for profitability but also positions the company as an innovative leader in algorithmic trading, capable of adapting its technology to diverse market conditions and opportunities.

Conclusion

This project has successfully developed and implemented an advanced algorithmic trading system for the BankNifty index, leveraging deep learning techniques to predict short-term market movements and execute profitable trades. The system demonstrates significant potential for generating consistent returns in the highly volatile Indian financial markets.

Key Achievements:

1. Model Development: We successfully created a Convolutional Neural Network (CNN) model that outperformed other architectures in predicting minute-by-minute price movements of the BankNifty index.
2. Data Engineering: A robust data pipeline was established, processing high-frequency financial data and engineering 45 distinct features to provide a comprehensive view of market conditions.
3. Deployment: The model was successfully deployed on Amazon SageMaker, enabling real-time predictions with low latency.

4. Performance: On the test set (July 31, 2024), the model achieved an RMSE of 24.64 and a direction accuracy of ~55% for moves greater than 11 points in either direction, showcasing its predictive capabilities.
5. Hypothetical Business Case: The system demonstrated potential profitability, with a projected daily profit of Rs. 825 and an impressive 5.5% daily profit percentage, translating to a theoretical annual ROI of 684%.
6. Risk Management: Implemented sophisticated risk management strategies, including dynamic stop-loss conditions and a favorable risk-reward ratio of 1:1.5.

Implications and Future Directions:

1. Scalability: The system's design allows for potential replication across other Indian market indices, offering significant opportunities for business expansion and risk diversification.
2. Continuous Improvement: The modular nature of our system allows for ongoing refinement of models, strategies, and risk management techniques.
3. Market Impact: As we scale, our high-frequency trading system has the potential to contribute to market efficiency and liquidity in the Indian financial markets.
4. Future Research: Areas for future exploration include incorporating sentiment analysis, developing ensemble models, and exploring reinforcement learning techniques for dynamic strategy adjustment.

Challenges and Considerations:

1. Market Volatility: The system's performance in extreme market conditions needs continued monitoring and adjustment.
2. Regulatory Landscape: Ongoing attention to evolving regulations in algorithmic and high-frequency trading is crucial.

3. Competition: As algorithmic trading becomes more prevalent, maintaining our competitive edge will require continued innovation and optimization.

In conclusion, this project has not only delivered a functioning algorithmic trading system but has also laid the groundwork for a potentially transformative business in the realm of quantitative finance. The combination of advanced machine learning techniques, robust engineering, and strategic business planning positions us to capitalize on the opportunities presented by the dynamic Indian financial markets.

As we move forward, our focus will be on rigorous real-world testing, continuous improvement of our models and strategies, and careful scaling of our operations. The potential for replication across multiple indices opens exciting possibilities for growth and establishes a strong foundation for long-term success in the algorithmic trading space.

This project represents a significant step forward in our journey to leverage artificial intelligence in finance, promising to deliver value not only to our organization but also to contribute to the overall efficiency and sophistication of the Indian financial markets.

References

1. Manveer Kaur Mangat, Erhard Reschenhofer, Thomas Stark, Christian Zwatz. High-Frequency Trading with Machine Learning Algorithms and Limit Order Book Data. Data Science in Finance and Economics, 2022, 2(4): 437-463. https://www.aimspress.com/article/doi/10.3934/DSFE.2022022
2. Arévalo, Andrés & Nino, Jaime & Hernandez, German & Sandoval, Javier. (2016). High-Frequency Trading Strategy Based on Deep Neural Networks. 9773. 424-436. 10.1007/978-3-319-42297-8_40. https://www.researchgate.net/publication/305214717_High-Frequency_Trading_Strategy_Based_on_Deep_Neural_Networks
3. Data Engineering with Python, Crickard P., https://www.packtpub.com/en-fi/product/data-engineering-with-python-9781839214189?type=ebook
4. Python for Algorithmic Trading, Hilpisch Y., https://www.oreilly.com/library/view/python-for-algorithmic/9781492053347/
5. Order based versus level book trade reporting: An empirical analysis – Journal of Banking and Finance, James U., Thomas M., Hardy J., https://ideas.repec.org/a/eee/jbfina/v125y2021ics0378426621000327.html
6. Technical Analysis of the Financial Markets – New York Institute of Finance, Murphy J., https://www.amazon.com/Technical-Analysis-Financial-Markets-Comprehensive/dp/0735200661
7. Options, Futures, and Other Derivatives – Pearson, Hull J., Basu S, https://www.pearson.com/en-us/subject-catalog/p/options-futures-and-other-derivatives/P200000005938/9780136939917