Stock price prediction

Burak Efe
Tim Oldörp

Table of Content

Background

In the world of finance, predicting stock prices is crucial for investors and analysts. Machine learning techniques, particularly those involving neural networks, have gained popularity due to their ability to model complex patterns in time series data.

The problem we want to solve

Traditional methods of stock price prediction often struggle to capture the intricate relationships in financial data. The aim of this project is to explore the effectiveness of **Long Short-Term Memory** (LSTM) networks, a type of **recurrent neural network** (RNN), for predicting stock prices using historical data and technical indicators.

Project Goal

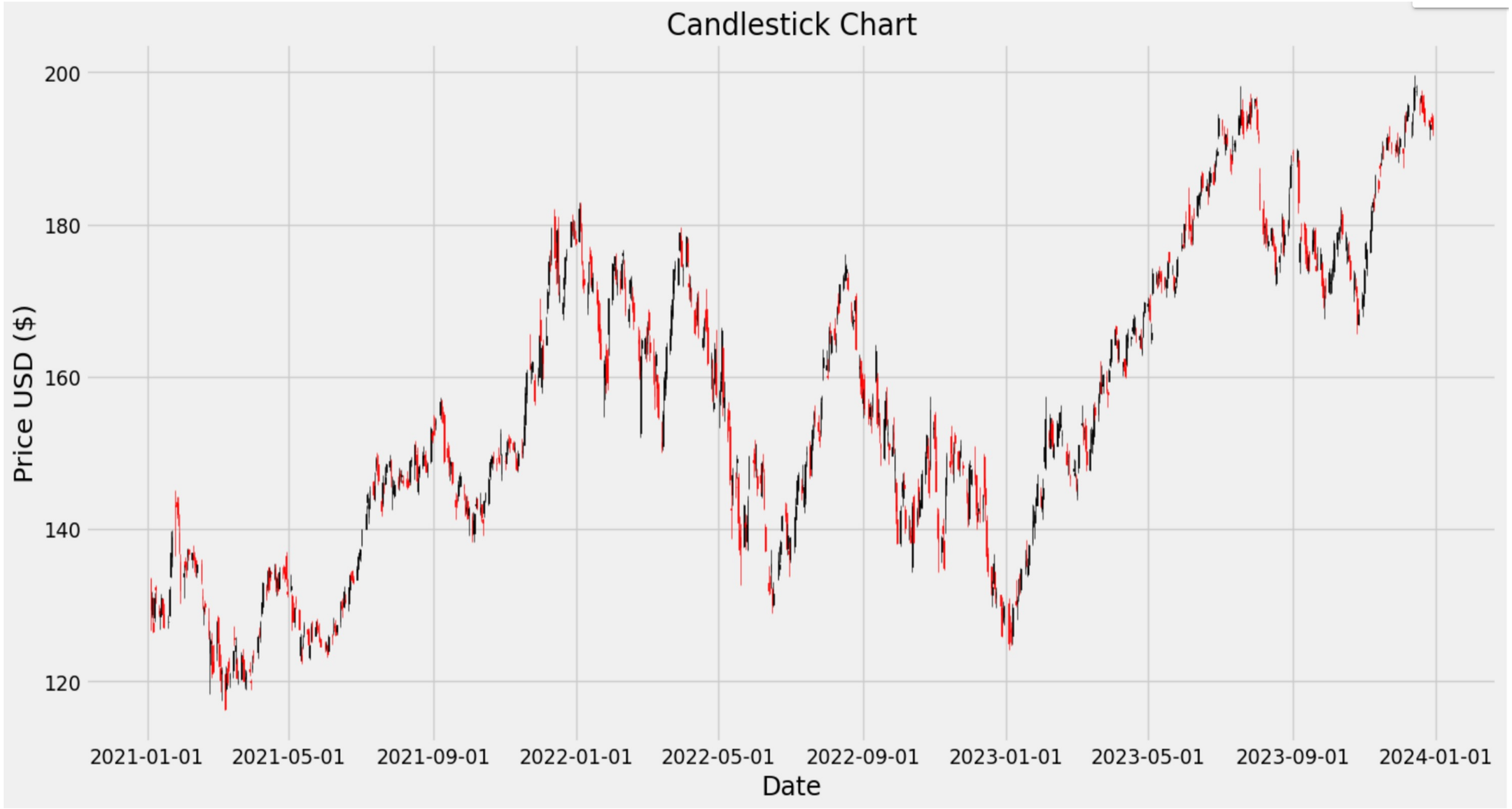
Primary Objective

- To develop an **LSTM-based** model to predict future stock prices of a given ticker symbol using **historical price data** and **technical indicators**.

Secondary Objectives

- To evaluate the model's performance using metrics such as **Mean Absolute Error** (MAE) and **Directional Accuracy** (DA).
- To optimize the model's hyperparameters using **Optuna** for improved prediction accuracy.

DATA


Candlestick Chart
Price USD ($)
200
180
160
140
120
2021-01-01
2021-05-01
2021-09-01
2022-01-01
2022-05-01
2022-09-01
2023-01-01
2023-05-01
2023-09-01
2024-01-01
Date

DATA

Data Source:

- Yahoo Finance: The stock data is downloaded using the yfinance library, which provides historical market data.

Data Range:

- Time Period: January 1, 2021, to January 1, 2024.
- Ticker Symbol: User-specified (e.g., AAPL).

Data Features:

- Raw Features: Open, High, Low, Close, Volume.
- Technical Indicators: **Simple Moving Average** (SMA), **Exponential Moving Average** (EMA), **Relative Strength Index** (RSI), **Moving Average Convergence Divergence** (MACD), **Bollinger Bands** (High and Low).

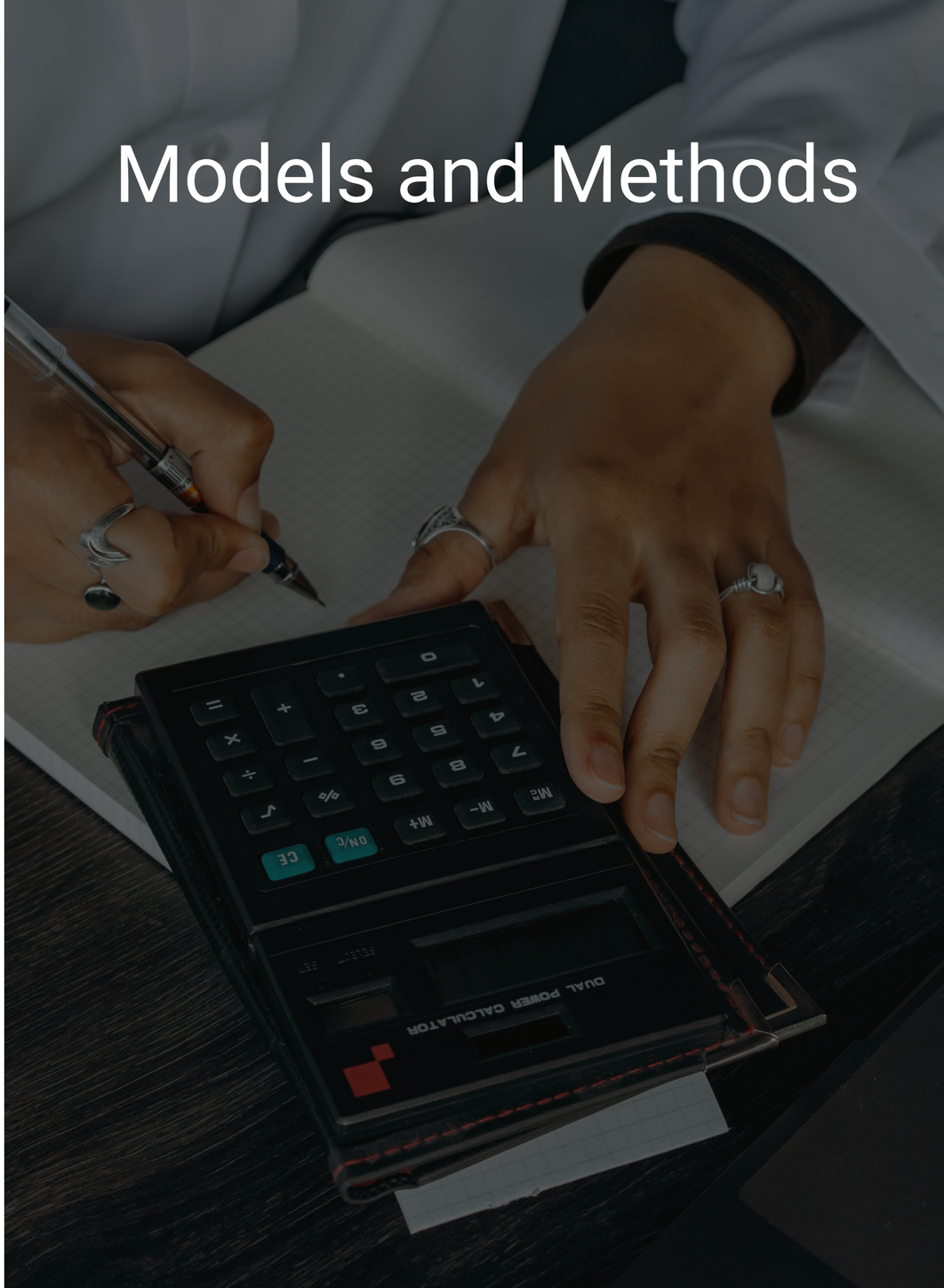
Models and Methods

Model:

- Model Architecture: An **LSTM** network with multiple layers is constructed.
- Hyperparameter Optimization: **Optuna** is used to tune **hyperparameters** like the number of units in LSTM **layers**, **dropout rate**, **learning rate**, and **batch size**.

```
# Function to build LSTM model
def build_lstm_model(window_size, num_features, best_params):
    units = best_params['units']
    dropout_rate = best_params['dropout_rate']
    learning_rate = best_params['learning_rate']
    batch_size = best_params['batch_size']

    print("Building LSTM model...")
    model = Sequential()
    model.add(LSTM(units=units, return_sequences=True, input_shape=(window_size, num_features)))
    model.add(Dropout(dropout_rate))
    model.add(LSTM(units=units, return_sequences=True))
    model.add(Dropout(dropout_rate))
    model.add(LSTM(units=units))
    model.add(Dropout(dropout_rate))
    model.add(Dense(units=1))
    model.compile(optimizer=Adam(learning_rate=learning_rate), loss='mean_squared_error')
    print("Model built successfully!")
    return model
```



Models and Methods

Model:

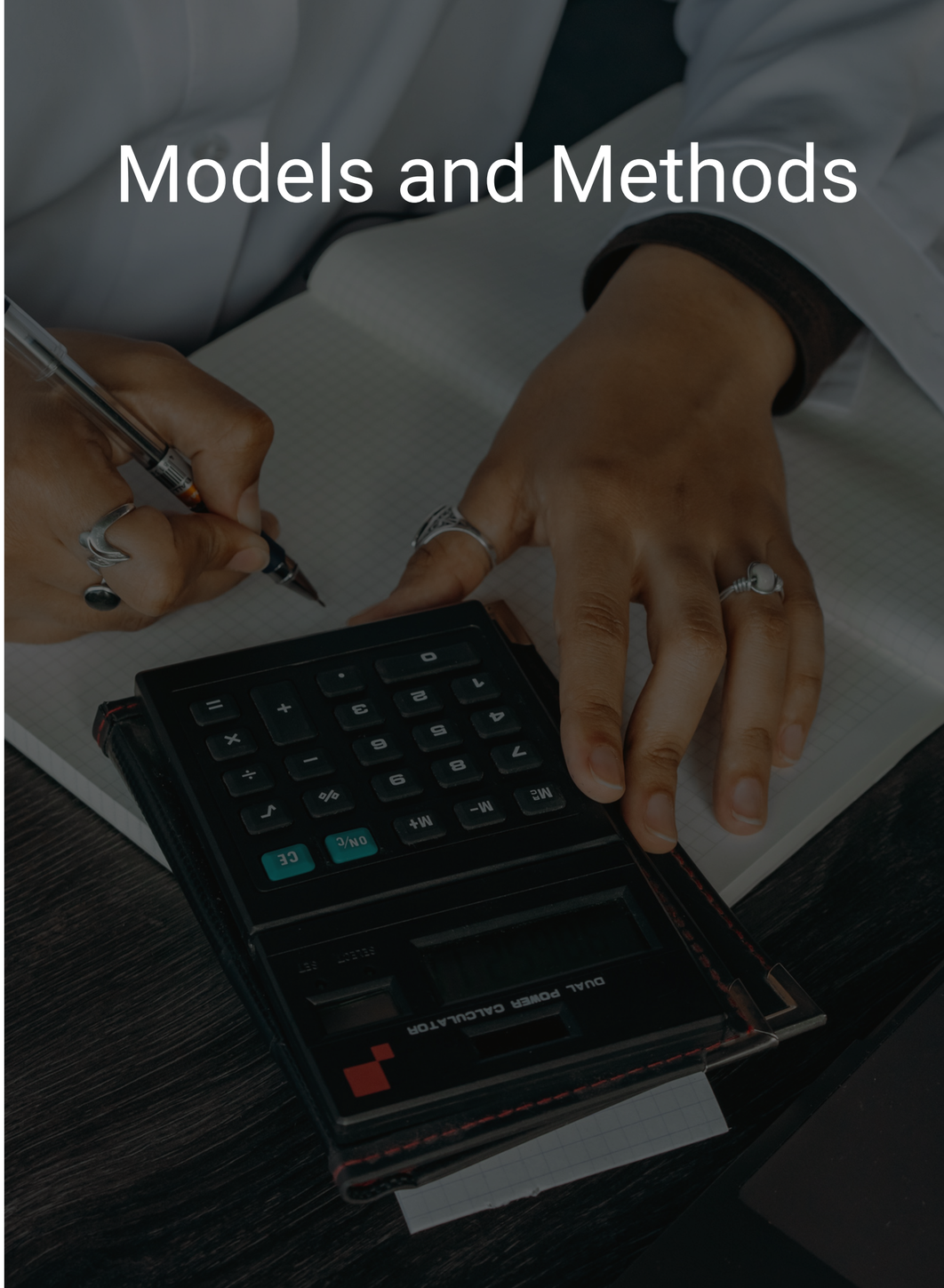
```python
def objective(trial, window_size, num_features, x_train, y_train, close_values, scaler, close_scaler, close_data):
    units = trial.suggest_int('units', 50, 200)
    dropout_rate = trial.suggest_float('dropout_rate', 0.1, 0.5)
    learning_rate = trial.suggest_float('learning_rate', 1e-5, 1e-2)
    batch_size = trial.suggest_categorical('batch_size', [16, 32, 64])

    model = Sequential()
    model.add(LSTM(units=units, return_sequences=True, input_shape=(window_size, num_features)))
    model.add(Dropout(dropout_rate))
    model.add(LSTM(units=units, return_sequences=True))
    model.add(Dropout(dropout_rate))
    model.add(LSTM(units=units))
    model.add(Dropout(dropout_rate))
    model.add(Dense(units=1))
    model.compile(optimizer=Adam(learning_rate=learning_rate), loss='mean_squared_error')

    early_stopping = EarlyStopping(monitor='loss', patience=10, restore_best_weights=True)
    reduce_lr = ReduceLROnPlateau(monitor='loss', factor=0.2, patience=5, min_lr=0.001)
    history = model.fit(x_train, y_train, epochs=100, batch_size=batch_size, verbose=0, callbacks=[early_stopping, reduce_lr])

    x_test = prepare_test_data(close_values, scaler, window_size)
    predictions = make_predictions(model, x_test, close_scaler)

    mae = calculate_mae(predictions, close_data['Close'].values[window_size:])
    return mae
```

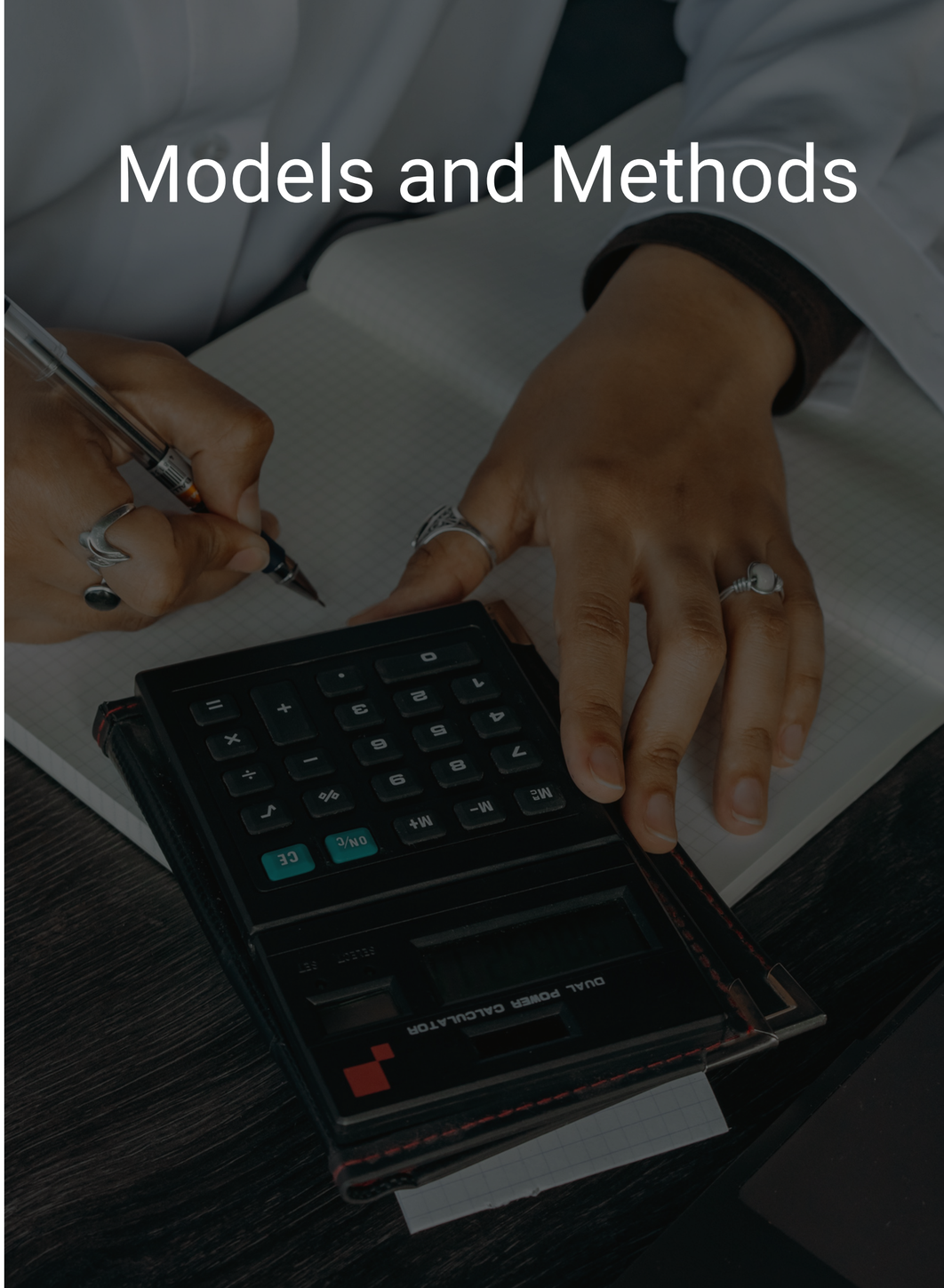


Models and Methods

Training:

- Training Data Preparation: The data is segmented into training windows.
- Model Training: The LSTM model is trained with **early stopping** and **learning rate reduction** techniques to prevent overfitting and enhance convergence.

```python
# Function to train LSTM model
def train_lstm_model(model, x_train, y_train, epochs, batch_size):
    print("Training LSTM model...")
    early_stopping = EarlyStopping(monitor='loss', patience=10, restore_best_weights=True)
    reduce_lr = ReduceLROnPlateau(monitor='loss', factor=0.2, patience=5, min_lr=0.001)
    history = model.fit(x_train, y_train, epochs=epochs, batch_size=batch_size, verbose=1, callbacks=[early_stopping, reduce_lr])
    print("Model trained successfully!")
    return history
```

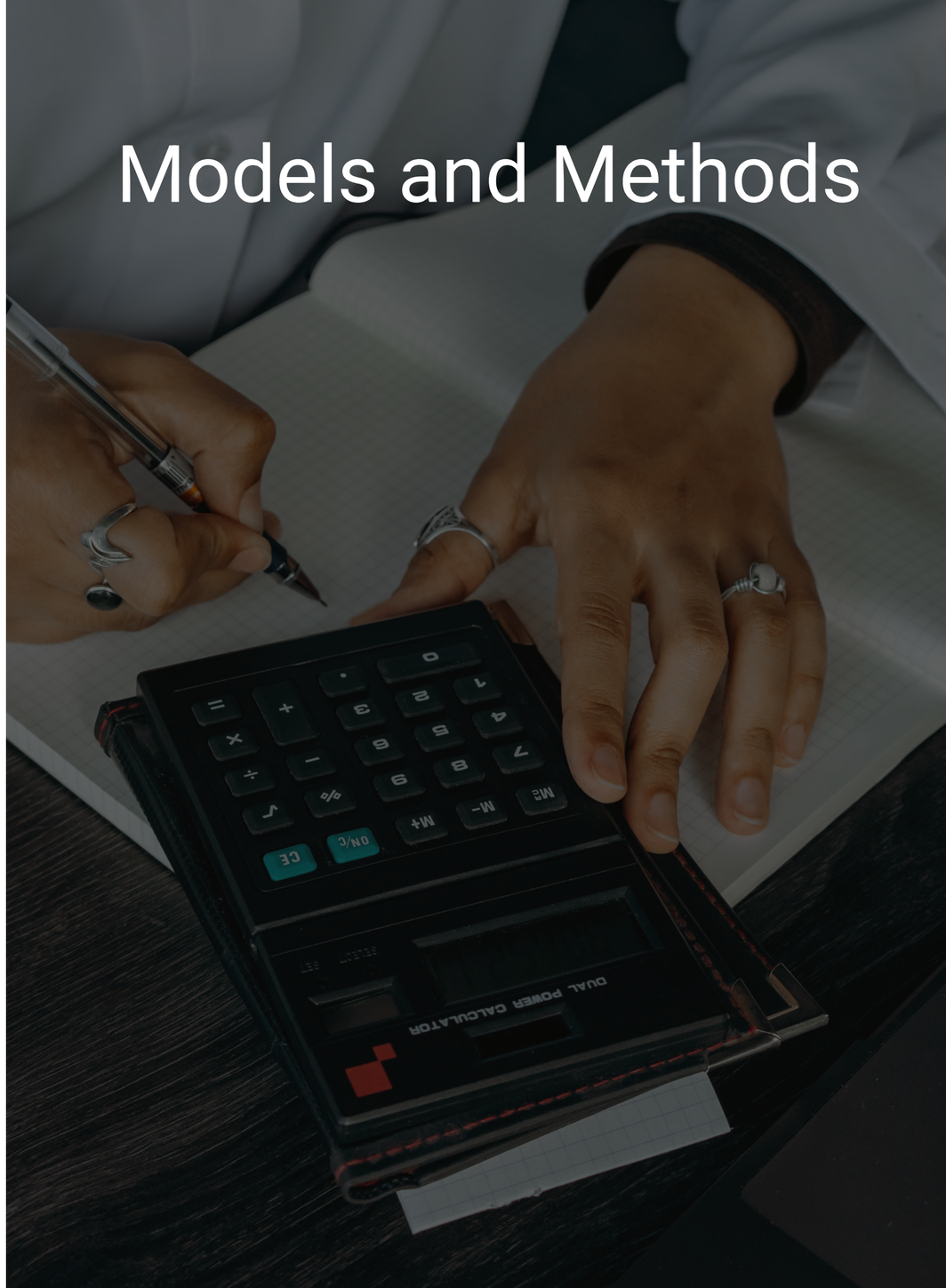


Models and Methods

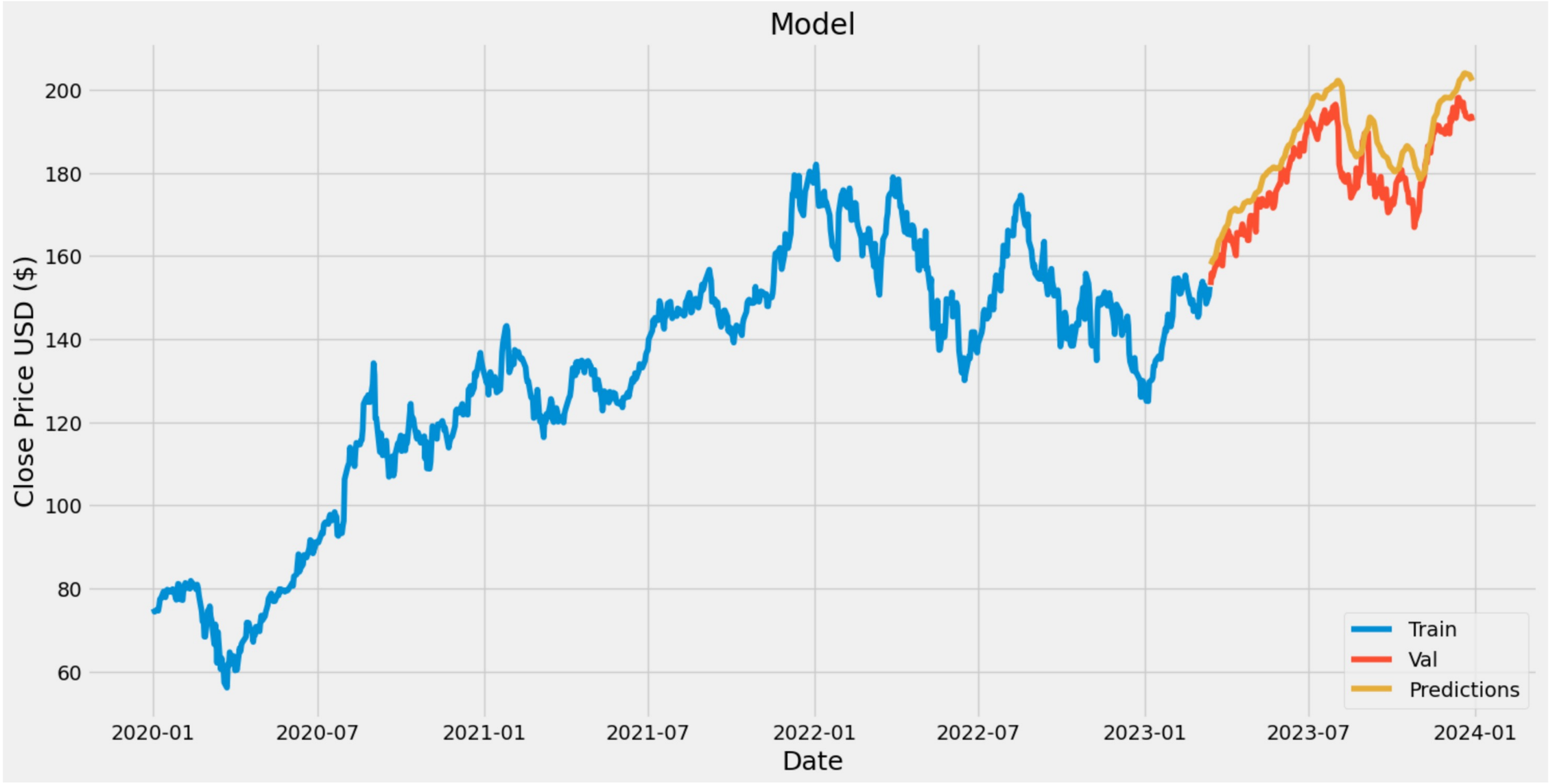
Evaluation Metrics:

- **Directional Accuracy** (DA): Measures the accuracy of the model in predicting the direction of price movement.

- **Mean Absolute Error** (MAE): Measures the average magnitude of the errors in the predictions



Results


Model
Close Price USD ($)
200
180
160
140
120
100
80
60
2020-01
2020-07
2021-01
2021-07
2022-01
2022-07
2023-01
2023-07
2024-01
Date
Train
Val
Predictions

Before tuning with optuna Hyperparameters

```
Predictions made successfully!
Directional Accuracy (DA): 0.6025641025641025
Mean Absolute Error (MAE): 1.686729972198278
Final predicted price: 191.61329697433575
```

After tuning with optuna Hyperparameters

```
Predictions made successfully!
Directional Accuracy (DA): 0.7243589743589743
Mean Absolute Error (MAE): 0.7342709294075369
Final predicted price: 193.47181102293143
```

Before tuning with optuna Hyperparameters

```
Predictions made successfully!
Directional Accuracy (DA): 0.4992412746585736
Mean Absolute Error (MAE): 2.4779755358452555
Final predicted price: 191.66836928045086
```

After tuning with optuna Hyperparameters

```
Predictions made successfully!
Directional Accuracy (DA): 0.5174506828528073
Mean Absolute Error (MAE): 1.9620724769238849
Final predicted price: 191.91752876960163
```

Outlook

Possible improvements for our model:

- Data Preprocessing and Feature Engineering
- Model Architecture
- Training Process
- Regularization Techniques
- Ensemble Methods
- Advanced Techniques
- Model Evaluation and Validation

Thank You

For your attention