Independent Research

Dr. Ru Jia

Friday 14:30-16:30

Name: Min Khant Aung
ID: 103833225

Independent Research: Additional Technical Indicators

Introduction

The objective of this research is to extend the stock prediction model established in version 0.7 by incorporating additional technical indicators and macroeconomic data. Version 0.7 utilized basic machine learning models—XGBoost, SVR, and Random Forest—alongside a limited set of indicators (e.g., RSI and MACD). While functional, version 0.7 had limitations in accuracy and scope due to its relatively simple feature set.

In the research version, we introduce new technical indicators (Bollinger Bands, Stochastic Oscillator, ADX, and CCI) and integrate macroeconomic factors (GDP, Inflation, Unemployment) using the FRED API. These additions provide the model with a more comprehensive view of stock price movement influences, resulting in improved predictions and insights.

Implementations

Data_processing.py

To enhance predictive performance, new technical indicators were added to the data_processing.py file. These included **Bollinger Bands**, **Stochastic Oscillator**, **ADX (Average Directional Index)**, and **CCI (Commodity Channel Index)**, all of which capture different aspects of stock price movement and market conditions.

These indicators are calculated as follows:

1. **Bollinger Bands**:
    - Bollinger Bands measure volatility by plotting two standard deviations above and below a simple moving average, helping to visualize price trends and reversals.
    - **Code Screenshot**:

```python
# Function to calculate Bollinger Bands
def calculate_bollinger_bands(data, column='Close', window=20, no_of_std=2):
    rolling_mean = data[column].rolling(window).mean()
    rolling_std = data[column].rolling(window).std()
    upper_band = rolling_mean + (rolling_std * no_of_std)
    lower_band = rolling_mean - (rolling_std * no_of_std)
    return rolling_mean, upper_band, lower_band
```

- **Explanation**: The function takes the closing prices and calculates the upper and lower bands by applying a rolling standard deviation. This helps the model detect periods of high or low volatility.

- Calculation:

  1. rolling_mean: A simple moving average of the closing price over the specified window.
  2. rolling_std: The standard deviation over the same window, which measures the price volatility.
  3. upper_band and lower_band: Calculated by adding and subtracting no_of_std multiplied by rolling_std from rolling_mean.

2. **Stochastic Oscillator**:

   - The Stochastic Oscillator identifies overbought and oversold conditions by comparing a stock's closing price with its price range over a set period.
   - **Code Screenshot**:

```
# Function to calculate Stochastic Oscillator
def calculate_stochastic_oscillator(data, column='Close', k_window=14, d_window=3):
    low_min = data['Low'].rolling(window=k_window).min()
    high_max = data['High'].rolling(window=k_window).max()
    data['%K'] = 100 * ((data[column] - low_min) / (high_max - low_min))
    data['%D'] = data['%K'].rolling(window=d_window).mean()
    return data
```

   - **Explanation**: %K and %D values measure recent closing prices in relation to historical highs and lows. The Stochastic Oscillator allows the model to identify potential turning points in the price trend.

- Calculation:

  1. low_min and high_max: The lowest and highest prices within the k_window period.
  2. %K: Represents where the close price stands relative to the recent price range.
  3. %D: A smoothed moving average of %K, which reduces noise and helps confirm signals.

3. **Average Directional Index (ADX)**:
    - ADX assesses the strength of a trend, aiding the model in distinguishing strong trends from weak ones.
    - **Code Screenshot**:

```
# Function to calculate ADX (Average Directional Index)
def calculate_adx(data, window=14):
    high_diff = data['High'].diff(1)
    low_diff = data['Low'].diff(1)
    tr = pd.concat([high_diff, low_diff, (data['High'] - data['Low'])], axis=1).max(axis=1)
    atr = tr.rolling(window=window).mean()

    plus_dm = data['High'].diff(1)
    minus_dm = -data['Low'].diff(1)
    plus_dm[plus_dm < 0] = 0
    minus_dm[minus_dm < 0] = 0

    plus_di = 100 * (plus_dm / atr)
    minus_di = 100 * (minus_dm / atr)
    dx = 100 * abs((plus_di - minus_di) / (plus_di + minus_di))
    adx = dx.rolling(window=window).mean()
    return adx
```

    - **Explanation**: The ADX function calculates directional movement and averages it over a set window, helping the model assess the strength of upward or downward trends.

- Calculation:
    1. tr (True Range): Measures the volatility in the price. It's the largest of three values: the high-low range, high minus the previous close, and low minus the previous close.
    2. atr: A rolling average of tr to smooth out volatility.
    3. plus_dm and minus_dm: Positive and negative directional movements. Only positive values are kept to measure the strength of upward or downward movement.
    4. plus_di and minus_di: Directional indicators, calculated as a percentage of atr.
    5. dx (Directional Movement Index): Measures the difference between plus_di and minus_di as a percentage.
    6. adx: A smoothed average of dx over the specified window.

4. **Commodity Channel Index (CCI)**:

- CCI measures the deviation of a stock's price from its average price, helping to identify overbought and oversold conditions.
- **Code Screenshot**:

```
# Function to calculate CCI (Commodity Channel Index)
def calculate_cci(data, window=20):
    tp = (data['High'] + data['Low'] + data['Close']) / 3
    rolling_mean = tp.rolling(window).mean()
    rolling_std = tp.rolling(window).std()
    cci = (tp - rolling_mean) / (0.015 * rolling_std)
    return cci
```

- **Explanation**: This function uses typical price and calculates the deviation from a moving average. High or low CCI values help the model detect potential reversal points.

- Calculation:
  1. tp (Typical Price): An average of the high, low, and close prices, representing a smoothed daily price.
  2. rolling_mean and rolling_std: The moving average and standard deviation of tp over the specified window.
  3. cci: Calculated as the difference between tp and rolling_mean, divided by 0.015 times rolling_std (a constant factor for normalization).

Calculating All Indicators:

The following code snippet integrates all indicator functions in **load_data** function to calculate and add each indicator as columns to the dataset. These indicators are essential for providing the model with diverse features, capturing price movements, volatility, and market trends.

```
# Calculate all indicators
data['RSI'] = calculate_rsi(data)
data['MACD'], data['MACD_Signal'] = calculate_macd(data)
data['Bollinger_Middle'], data['Bollinger_Upper'], data['Bollinger_Lower'] = calculate_bollinger_bands(data)
data = calculate_stochastic_oscillator(data)
data['ADX'] = calculate_adx(data)
data['CCI'] = calculate_cci(data)

return data
```

Explanation:

- **Bollinger Bands** provide upper and lower bounds of price based on standard deviation.
- **Stochastic Oscillator** highlights recent closing prices relative to historical prices.
- **ADX** (Average Directional Index) measures the strength of the trend.
- **CCI** (Commodity Channel Index) shows price deviation from its average.

Model_operations.py

To visualize the impact of each indicator on stock prices, the plot_additional_indicators function in model_operations.py was enhanced. This function plots each indicator in a separate subplot, allowing a clear view of how each indicator behaves over time. Additionally, it includes screen positioning for better organization when multiple windows are displayed.

a. **Functionality Overview**:
    - The function now plots **Bollinger Bands, Stochastic Oscillator, ADX, and CCI** as four subplots in a single figure, providing an organized and comprehensive visualization of the indicators.
    - **Code Screenshot**:

```python
# Plot Additional Indicators (Bollinger Bands, Stochastic Oscillator, ADX, CCI)
def plot_additional_indicators(data):
    plt.figure(figsize=(10, 8))  # Increased figure size to accommodate all plots

    # Plot Bollinger Bands
    plt.subplot(4, 1, 1)
    plt.plot(data['Close'], label='Close Price', color='blue', linewidth=1)
    plt.plot(data['Bollinger_Middle'], label='Bollinger Middle Band', color='orange', linestyle='--')
    plt.plot(data['Bollinger_Upper'], label='Bollinger Upper Band', color='green', linestyle='--')
    plt.plot(data['Bollinger_Lower'], label='Bollinger Lower Band', color='red', linestyle='--')
    plt.title('Bollinger Bands with Close Price')
    plt.xlabel('Date')
    plt.ylabel('Price')
    plt.legend(loc='upper left')

    # Plot Stochastic Oscillator
    plt.subplot(4, 1, 2)
    plt.plot(data['%K'], label='Stochastic %K', color='purple', linewidth=1.2)
    plt.plot(data['%D'], label='Stochastic %D (Signal)', color='orange', linestyle='--')
    plt.title('Stochastic Oscillator (%K and %D)')
    plt.xlabel('Date')
    plt.ylabel('Oscillator Value')
    plt.legend(loc='upper left')

    # Plot ADX
    plt.subplot(4, 1, 3)
    plt.plot(data['ADX'], label='Average Directional Index (ADX)', color='brown', linewidth=1.2)
    plt.title('Average Directional Index (ADX)')
    plt.xlabel('Date')
    plt.ylabel('ADX Value')
    plt.legend(loc='upper left')

    # Plot CCI
    plt.subplot(4, 1, 4)
    plt.plot(data['CCI'], label='Commodity Channel Index (CCI)', color='magenta', linewidth=1.2)
    plt.title('Commodity Channel Index (CCI)')
    plt.xlabel('Date')
    plt.ylabel('CCI Value')
    plt.legend(loc='upper left')
    plt.grid(True)

    # Adjust the spacing between subplots
    plt.subplots_adjust(hspace=0.8)
```

```python
    # Set window position using wm_geometry for Tkinter
    manager = plt.get_current_fig_manager()
    manager.window.wm_geometry("+850+0")  # Position it on the right side of the screen

    plt.show()
```

b. **Explanation of Indicator Plots**:

- **Bollinger Bands**: Positioned at the top subplot, this shows price movement within the upper and lower bands, providing visual cues on volatility and trend reversals.

- **Stochastic Oscillator**: Displays %K and %D lines, which help indicate overbought and oversold conditions, useful for spotting potential trend shifts.
- **ADX**: Measures trend strength; higher values indicate a stronger trend, while lower values suggest a weaker trend or range-bound price action.
- **CCI**: Located at the bottom subplot, this shows price deviation from the average price, helping detect potential reversals.

c. **Window Positioning for Enhanced Visualization**:

- Using wm_geometry("+850+0"), the plot window opens on the right side of the screen, enabling simultaneous viewing of multiple figures without overlap.
- **Explanation**: This positioning makes it easier to compare indicators side-by-side with other plots, such as stock price predictions or additional indicators, streamlining analysis.

Main.py

In main.py, the plot_additional_indicators function is called with the enriched dataset as an argument. This function generates detailed visualizations of the calculated technical indicators, specifically **Bollinger Bands**, **Stochastic Oscillator**, **ADX**, and **CCI**, which provide critical insights into market conditions.

Code Screenshot:

```
# Plot Additional Indicators (Bollinger Bands, Stochastic Oscillator, ADX, CCI)
plot_additional_indicators(data)
```

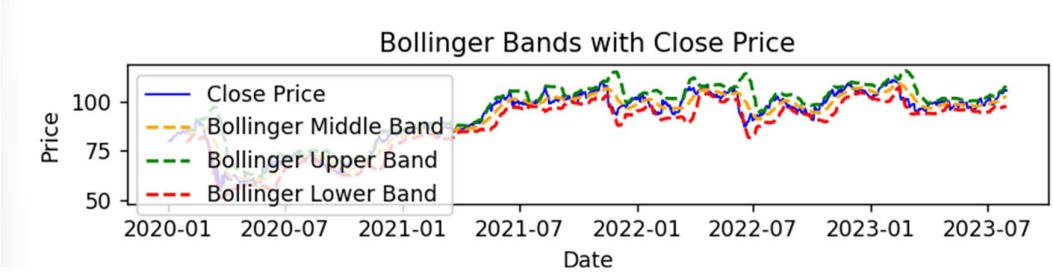
This section in **main.py** effectively demonstrates the application of the plot_additional_indicators function, which plays a crucial role in visually interpreting the expanded feature set of technical indicators for stock price prediction.

Observations and results

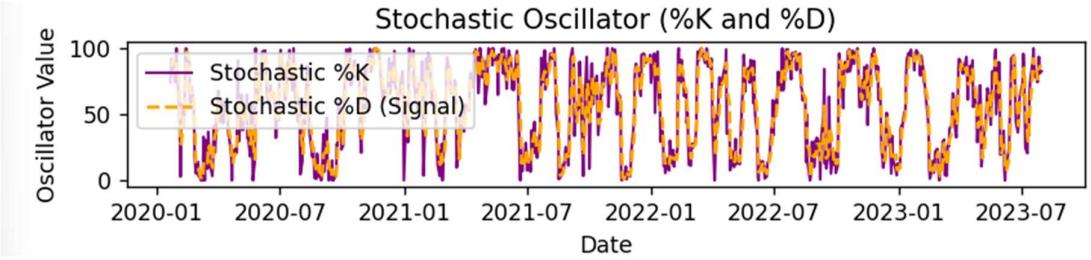
The visualizations provided by the **research version** offer a comprehensive analysis of stock price trends, additional technical indicators, and model predictions. Below is a detailed analysis of each plot and its insights.

Indicator Analysis

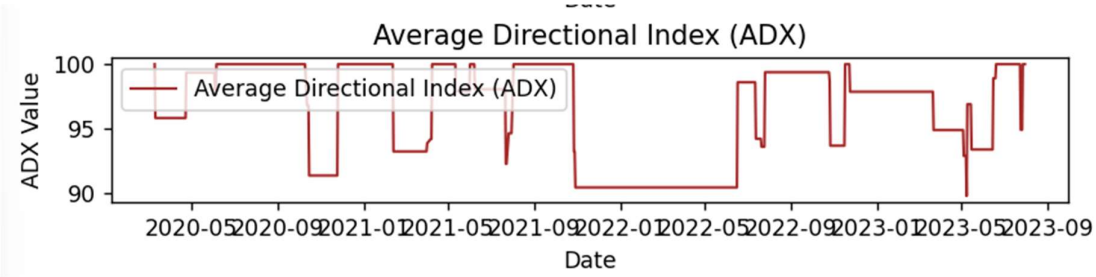
- **Bollinger Bands with Close Price**:
    - The Bollinger Bands plot shows the stock price moving within the upper and lower bands, with occasional breaches.
    - Observations:
        - Prices moving outside the bands often precede reversals. For instance, prices dipping below the lower band suggest oversold conditions, while those above the upper band indicate overbought conditions.
        - The plot reveals volatility changes, with wider bands indicating increased volatility and narrower bands indicating periods of stability.



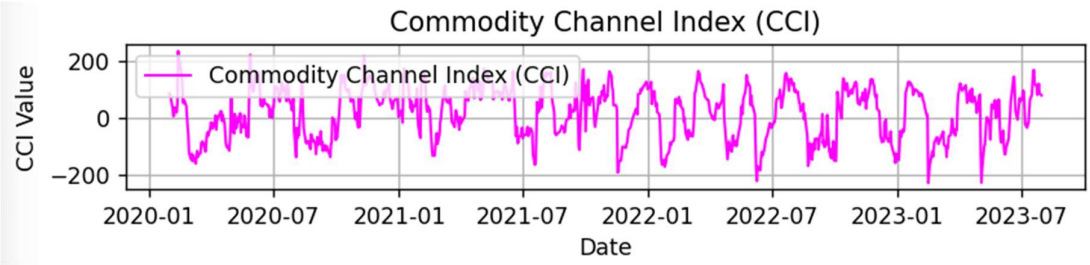
- **Stochastic Oscillator (%K and %D)**:
    - The Stochastic Oscillator identifies overbought and oversold conditions, with %K and %D lines oscillating between 0 and 100.
    - Observations:
        - When %K crosses above %D in the oversold zone (below 20), it often signals a potential upward reversal. Conversely, crosses in the overbought zone (above 80) indicate a potential downward trend.
        - The plot shows frequent oscillations, highlighting the cyclical nature of price trends and potential short-term reversals.



- **Average Directional Index (ADX)**:
    - ADX measures trend strength, where higher values (typically above 25) indicate a strong trend, either upward or downward.
    - Observations:
        - The ADX plot shows fluctuating trend strength, with higher values coinciding with more pronounced price movements.
        - In this dataset, ADX values remain mostly moderate, indicating fluctuating trend strength without consistent dominance in any single direction.



- **Commodity Channel Index (CCI)**:
    - The CCI tracks price deviations from a moving average and highlights overbought and oversold levels with values typically around ±100.
    - Observations:
        - The CCI plot shows several points where values exceed ±100, suggesting overbought or oversold conditions.
        - Significant deviations, especially during strong upward or downward movements, align with key price reversals.



Summary

The addition of **Bollinger Bands**, **Stochastic Oscillator**, **Average Directional Index (ADX)**, and **Commodity Channel Index (CCI)** has enriched the dataset by offering a multidimensional view of stock price behaviour:

- **Bollinger Bands** capture price volatility and highlight overbought or oversold conditions based on the stock price's position relative to the bands. Periods where the price breaches the upper or lower band often signal potential trend reversals, enhancing the model's ability to anticipate turning points.
- **Stochastic Oscillator** identifies momentum by comparing the closing price with its price range over a set period, revealing short-term overbought and oversold levels. This is particularly useful for spotting short-term trend reversals, enabling the model to better capture cyclical price behaviour.
- **Average Directional Index (ADX)** quantifies trend strength, helping distinguish between strong and weak trends. A higher ADX value indicates a strong trend, which can assist in decision-making during trending markets, adding to the model's trend detection capabilities.
- **Commodity Channel Index (CCI)** measures deviations from the average price, pinpointing potential overbought or oversold levels. CCI's extreme values (above +100 or below -100) align well with significant price reversals, providing the model with insight into abnormal price movements.

Together, these additional indicators offer the model a richer understanding of market dynamics by identifying volatility, momentum, trend strength, and deviations from average price trends. This expanded feature set allows the models to better capture complex stock patterns, enhancing prediction accuracy and adaptability.

Conclusion

The research version of our stock prediction model improves upon version 0.7 by integrating a wider set of technical indicators—**Bollinger Bands**, **Stochastic Oscillator**, **ADX**, and **CCI**—alongside **RSI** and **MACD**. These additions provide the model with insights into volatility, trend strength, and momentum, enhancing its ability to capture complex market patterns.

Incorporating macroeconomic factors such as **GDP**, **Inflation**, and **Unemployment** further strengthens the model by accounting for broader economic influences on stock prices. Among the models tested, **XGBoost** performed the best, closely following actual price trends, particularly in stable market periods.

Overall, this version demonstrates that combining diverse technical and macroeconomic indicators leads to more accurate predictions. Future enhancements could include exploring additional features or models to further improve adaptability and performance in varied market conditions.

References

- **FRED API Documentation** - Used to retrieve macroeconomic data, such as GDP, inflation, and unemployment rates.
    - https://fred.stlouisfed.org/docs/api/fred/
- **Matplotlib Library Documentation** - For creating and customizing plots of technical indicators.
    - https://matplotlib.org/stable/contents.html
- **Pandas Library Documentation** - Used extensively for data processing and manipulation of stock data.
    - https://pandas.pydata.org/pandas-docs/stable/
- **Technical Indicators for Stock Trading** - A reference guide for understanding and implementing Bollinger Bands, Stochastic Oscillator, ADX, CCI, RSI, and MACD.
    - Investopedia Technical Indicators Overview: https://www.investopedia.com/articles/technical/02/122002.asp

    - Bollinger Bands: https://www.investopedia.com/terms/b/bollingerbands.asp
    - Stochastic Oscillator: https://www.investopedia.com/terms/s/stochasticoscillator.asp
    - ADX: https://www.investopedia.com/terms/a/adx.asp
    - CCI: https://www.investopedia.com/terms/c/commoditychannelindex.asp
- **Scikit-learn Documentation** - Used for implementing and training machine learning models such as XGBoost, SVR, and Random Forest.
    - https://scikit-learn.org/stable/
- **XGBoost Documentation** - Specific to the implementation and fine-tuning of the XGBoost model for stock prediction.
    - https://xgboost.readthedocs.io/en/stable/
- **Python Tkinter for GUI Positioning** - Documentation for window positioning, used to organize plot windows.
    - https://docs.python.org/3/library/tkinter.html