Comprehensive Exploratory Data Analysis Report for Stock Price Prediction

1. Data Overview and Initial Inspection

The dataset contains historical stock price data with the following columns:

- **Date**: Trading date
- **Close**: Closing price
- **High**: Highest price during the trading day
- **Low**: Lowest price during the trading day
- **Open**: Opening price
- **Volume**: Number of shares traded

Initial inspection of the dataset revealed its structure and properties:

```
# Initial inspection of the data
print(f"Dataset shape: {df.shape}")
print(f"Date range: {df['Date'].min()} to {df['Date'].max()}")
print(f"Number of missing values: {df.isnull().sum().sum()}")
```

2. Data Preprocessing Steps and Justification

Our preprocessing approach included several key decisions based on the nature of financial time series data:

Date Formatting and Indexing

Converting the 'Date' column to datetime format and setting it as the index is essential for time series analysis, allowing for proper chronological ordering and time-based operations.

```
# Convert 'Date' to datetime and set as index
df['Date'] = pd.to_datetime(df['Date'])
df = df.set_index('Date')
```

Missing Value Handling

After examining the data, we chose to handle missing values by removing them, as they constituted a small percentage of the dataset and imputation might introduce bias in financial data.

```
# Drop missing values
df = df.dropna()
```

This approach maintains the integrity of the actual market data without introducing artificial values that could misrepresent historical price movements.

3. Visualizations and Key Pattern Analysis

Long-term Price Trends

Visualizing the closing price over time reveals the overall trend and major market phases:

```
plt.figure(figsize=(14, 7))
plt.plot(df['Close'])
plt.title('Stock Closing Price Over Time')
plt.xlabel('Date')
plt.ylabel('Price')
plt.grid(True)
```

Key observations include:

- Overall upward trend with several distinct growth phases
- Notable periods of correction and consolidation
- Increasing price volatility in more recent periods

Return Distribution Analysis

Daily returns visualization helps understand the stock's volatility characteristics:

```
# Calculate daily returns
df['Daily_Return'] = df['Close'].pct_change()

plt.figure(figsize=(10, 6))
sns.histplot(df['Daily_Return'].dropna(), kde=True)
plt.title('Distribution of Daily Returns')
```

The return distribution reveals:

- Approximately normal distribution centered near zero
- Presence of fat tails, indicating more frequent extreme moves than a normal distribution would predict
- Slight negative skewness, suggesting that negative returns tend to be more extreme than positive returns

Correlation Analysis

Understanding relationships between price and volume variables:

```
plt.figure(figsize=(10, 8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
```

Key correlations observed:

- Strong positive correlation between Open, High, Low, and Close prices
- Weaker correlation between Volume and price metrics
- Understanding these relationships helps inform feature selection for modeling

Monthly Price Patterns

Examining monthly aggregated data to identify potential seasonality:

```
# Monthly analysis
monthly_avg = df.resample('M')['Close'].mean()
plt.figure(figsize=(14, 7))
plt.plot(monthly_avg)
plt.title('Monthly Average Closing Price')
plt.grid(True)
```

This analysis helps identify:

- Long-term trend more clearly without daily noise
- Potential cyclical patterns in price movement
- Major market regime changes over time

4. Feature Engineering Implementation and Justification

Based on the EDA insights, we implemented several classes of features known to be effective for stock price prediction:

Price-based Features

Moving averages smooth out short-term fluctuations to highlight longer-term trends and are widely used in technical analysis for identifying support and resistance levels.

```
# Moving averages
df['MA5'] = df['Close'].rolling(window=5).mean()
df['MA20'] = df['Close'].rolling(window=20).mean()
df['MA50'] = df['Close'].rolling(window=50).mean()
```

These specific windows (5, 20, and 50 days) were chosen to capture short, medium, and long-term price trends respectively, providing the model with multiple timeframe perspectives.

Technical Indicators

RSI (Relative Strength Index) measures the speed and magnitude of price movements to identify overbought or oversold conditions:

```
# RSI (Relative Strength Index)
def calculate_rsi(data, period=14):
    delta = data.diff()
    up = delta.clip(lower=0)
    down = -1 * delta.clip(upper=0)
    ema_up = up.ewm(com=period-1, adjust=False).mean()
    ema_down = down.ewm(com=period-1, adjust=False).mean()
    rs = ema_up / ema_down
    return 100 - (100 / (1 + rs))
```

```
df['RSI'] = calculate_rsi(df['Close'])
```

MACD (Moving Average Convergence Divergence) is a trend-following momentum indicator that shows the relationship between two moving averages:

```
# MACD (Moving Average Convergence Divergence)
df['EMA12'] = df['Close'].ewm(span=12, adjust=False).mean()
df['EMA26'] = df['Close'].ewm(span=26, adjust=False).mean()
df['MACD'] = df['EMA12'] - df['EMA26']
df['MACD_Signal'] = df['MACD'].ewm(span=9, adjust=False).mean()
```

Bollinger Bands measure price volatility relative to moving averages, helping identify potential reversal points:

```
# Bollinger Bands
df['20MA'] = df['Close'].rolling(window=20).mean()
df['20SD'] = df['Close'].rolling(window=20).std()
df['Upper_Band'] = df['20MA'] + (df['20SD'] * 2)
df['Lower_Band'] = df['20MA'] - (df['20SD'] * 2)
```

Volatility is a key factor in price movement prediction:

```
# Volatility
df['Volatility'] = df['Close'].rolling(window=20).std()
```

Temporal Features

Calendar-based features capture potential day-of-week and month-of-year effects:

```
# Time-based features
df['DayOfWeek'] = df.index.dayofweek
df['Month'] = df.index.month
df['Year'] = df.index.year
```

These features allow the model to learn potential seasonal patterns like month-end effects, January effect, or day-of-week patterns often observed in financial markets.

Target Variable

For this prediction task, we aim to forecast the closing price 5 days into the future:

```
# Target variable (5-day future price)
df['Future_Close_5'] = df['Close'].shift(-5)
```

The 5-day horizon provides a balance between short-term prediction accuracy and practical utility for trading strategies.

5. Feature Selection for Modeling

Based on the comprehensive EDA and domain knowledge of financial markets, the following features were selected:

```
features = ['Close', 'High', 'Low', 'Open', 'Volume', 'MA5', 'MA20', 'MA50',
'RSI', 'MACD', 'MACD_Signal', 'Volatility', 'DayOfWeek', 'Month']
```

Justification for Selected Features:

1. **Price Variables** (Close, High, Low, Open):
   - Provide the fundamental price information
   - Different price points capture the full range of trading activity
2. **Volume**:
   - Trading volume often precedes price movements
   - Helps identify the strength behind price trends
3. **Technical Indicators** (MA5, MA20, MA50, RSI, MACD, MACD_Signal):
   - Capture different aspects of market dynamics (trend, momentum, overbought/oversold conditions)
   - Proven effectiveness in traditional technical analysis
4. **Volatility**:
   - Critical for understanding potential price ranges
   - Market volatility regimes often persist, making this a valuable predictive feature
5. **Temporal Features** (DayOfWeek, Month):
   - Capture cyclical patterns in market behavior
   - Account for potential seasonality effects

These features represent a comprehensive set of predictors that cover different aspects of market behavior, providing the model with a rich representation of the factors that influence stock prices.