STOCK PRICE PREDICTION

Phase - 4

Feature Engineering:

1. **Data Collection:** Ensure you have the historical stock price and relevant financial data for the stock(s) you want to analyze. You can collect this data from sources like Yahoo Finance, Alpha Vantage, or other financial data providers.
2. **Data Preprocessing:** Clean and preprocess the data, handling missing values and outliers, and converting date columns to the appropriate datetime format.
3. **Feature Selection:** Carefully select or engineer features that are likely to impact stock prices. These features can include technical indicators, fundamental data, market sentiment, and economic indicators. Common features might include moving averages, volume, volatility, and company-specific metrics like earnings and dividends.
4. **Lag Features:** Create lag features to capture historical trends and patterns in the data. For example, you can create lag features for the closing price of the stock over the past few days or weeks.
5. **Technical Indicators:** Calculate technical indicators like Moving Averages, Relative Strength Index (RSI), Bollinger Bands, and MACD (Moving Average Convergence Divergence).
6. **Market Sentiment Analysis:** Incorporate market sentiment data, such as news sentiment scores, social media sentiment, or sentiment from financial news articles.
7. **Volatility Measures:** Compute measures of volatility like historical volatility and Average True Range (ATR).

Model Training:

1. **Select a Model:** Choose a suitable machine learning or statistical model for stock price prediction. Common models include linear regression, time series models (e.g., ARIMA or LSTM), and machine learning models like decision trees, random forests, or gradient boosting.
2. **Data Split:** Split your data into training and testing sets. You can use historical data for training and reserve a portion for testing to evaluate your model's performance.
3. **Train the Model:** Train the chosen model using the training data. If you're using a machine learning model, you may want to fine-tune hyperparameters for better performance.
4. **Make Predictions:** Use the trained model to make predictions on the testing data or future data.

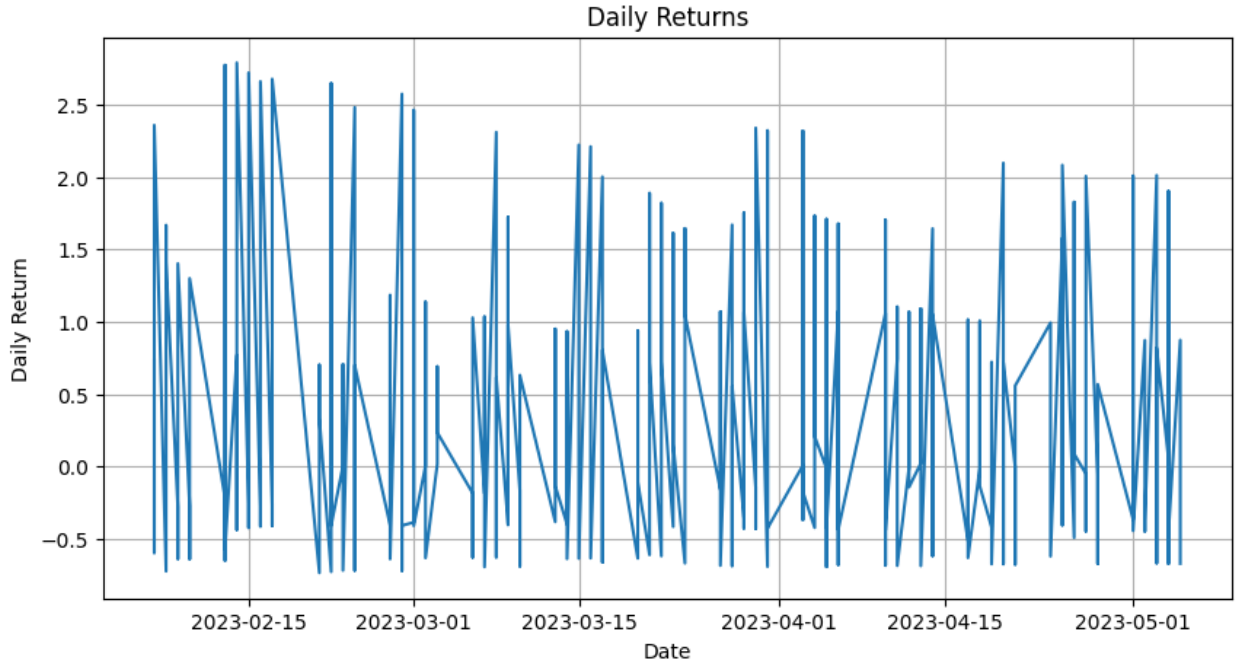
Evaluation:

1. **Evaluation Metrics:** Evaluate the performance of your model using appropriate metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-squared (R^2). Additionally, consider financial metrics like Sharpe Ratio or Annualized Return.
2. **Visualization:** Visualize the model's predictions and compare them to the actual stock prices using line plots or candlestick charts. This helps you assess the model's ability to capture trends and patterns.
3. **Backtesting:** If your analysis includes trading strategies, consider backtesting the strategies using historical data to assess their performance over time.
4. **Cross-Validation:** Perform cross-validation to ensure your model's robustness and avoid overfitting.
5. **Fine-Tuning:** If the model's performance is not satisfactory, consider fine-tuning the model, exploring different features, or trying other models.

Source Code:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
df = pd.read_csv('stocks.csv')
df['Date'] = pd.to_datetime(df['Date'])

df = df.sort_values('Date')

df.set_index('Date', inplace=True)
df['Daily Return'] = df['Close'].pct_change()
mean_daily_return = df['Daily Return'].mean()
std_daily_return = df['Daily Return'].std()
annualized_return = ((1 + mean_daily_return) ** 252) - 1

plt.figure(figsize=(10, 5))
plt.plot(df.index, df['Daily Return'])
plt.title('Daily Returns')
plt.xlabel('Date')
plt.ylabel('Daily Return')
plt.grid()
plt.show()


print(f"Mean Daily Return: {mean_daily_return:.4f}")
print(f"Standard Deviation of Daily Return: {std_daily_return:.4f}")
print(f"Annualized Return: {annualized_return:.4f}")
```

Output:




```
Mean Daily Return: 0.3143
Standard Deviation of Daily Return: 0.9584
Annualized Return: 807093635332932054773647015936.0000
```