Rick Hsu, Duraid Syed

Data Management for Data Science

Professor Guna

12/9/24

# Predictive Stock Market Analysis and Ranking System

The stock market is a dynamic and complex system that attracts both researchers and investors, driven by the idea of predicting trends and identifying financial opportunities. With the increasing usage of data science and machine learning, the financial domain has experienced a revolution in its approach to analyzing historical data and making predictions. This project seeks to explore the patterns of stock market behavior using historical data from the S&P 500 index, with five years of daily stock prices for all currently listed companies.

## Project Overview

The core problem this project addresses is the challenge of getting actionable insights from the volatile nature of stock market data. With five years of historical data for all companies currently listed on the S&P 500 index, the objective is to analyze trends, identify patterns, and predict future stock prices. These tasks require combining data science concepts, machine learning algorithms, and database management skills.

Key aspects include data collection and cleaning to ensure appropriate inputs, using SQL for querying and manipulation of the large dataset, and utilizing visualization libraries to communicate trends effectively. Machine learning models such as regression and time series forecasting will form the foundation for predictive analysis. This approach aligns directly with material taught throughout the course, particularly those focusing on data preprocessing, and the application of machine learning to solve real-world problems.

## Novelty and Importance

This project is important because it addresses a domain where actionable insights can lead to financial gains while minimizing risks. The novelty lies in the fact that most existing studies and tools focus on isolated aspects of financial data analysis, such as modeling volatility or single-stock prediction. In contrast, this project explores a less specific approach, allowing for broader insights.

On a personal level, we are excited about this project because it merges our interests in data science and finance. Working with a rich dataset that includes attributes such as opening and

closing prices, trading volumes, and stock volatility provides a unique opportunity to apply advanced machine learning techniques. It also presents an opportunity to explore time series analysis and predictive modeling—skills that are invaluable in both academic and professional contexts.

This project includes several important stages, each contributing to the goal of building a data-driven stock price prediction and analysis model. The contributions span from data preprocessing to predictive modeling and advanced analysis, all leading in insights that can benefit stakeholders such as traders, investors, or data scientists. Below is a breakdown of the work completed:

**Data Source and Preparation:**
The dataset used in this project includes stock market data for all companies in the S&P 500 index, spanning multiple years. This raw data contains key features such as the opening, closing, high, low prices, and volume traded for each stock, as well as the stock ticker and date of the trading day. The dataset required preprocessing to ensure its usability in further analysis.

One of the first tasks was dealing with missing data. In stock market data, missing values are common due to incomplete daily records or inconsistencies in data sources. The missing values were fixed using forward filling, which is a common method for time-series data. This method ensures that each missing entry in the series is filled with the most recent valid entry, preserving the continuity of stock prices.

Additionally, the *date* column was converted to a *datetime* object to allow time-based indexing and filtering. The *stock_id* column was used to uniquely identify each stock, with each ticker mapped to a corresponding identifier, facilitating later data aggregation and analysis by stock.

Duplicate records were identified based on the combination of *stock_id* and *date*. Any duplicate entries were removed to prevent skewed analysis and ensure that each stock's daily data was represented only once.

```
# Fill missing values using forward-fill (previous row's value)
df['open'] = df['open'].ffill()
df['high'] = df['high'].ffill()
df['low'] = df['low'].ffill()

# Verify no missing values remain
print("Missing Values After Cleaning:\n", df.isnull().sum())

# Convert 'date' column to datetime format
df['date'] = pd.to_datetime(df['date'])

# Verify the conversion
print(df.dtypes)

df.rename(columns={
    'Name': 'ticker',
    'date': 'date',
    'open': 'open_price',
    'high': 'high_price',
    'low': 'low_price',
    'close': 'close_price',
    'volume': 'trade_volume'
}, inplace=True)

print("Columns After Renaming:", df.columns)


# Check for duplicates
duplicate_rows = df.duplicated().sum()
print(f"Number of duplicate rows: {duplicate_rows}")
```

**Exploratory Analysis:**
Before beginning model development, we began by analyzing the data. The primary goal was to understand the data's structure, identify trends and patterns, and look for any outliers or anomalies that could impact model accuracy.

**Visualizing Stock Performance:**
Key visualizations were generated to give an intuitive sense of how stocks perform: In order to measure the top-performing stocks in terms of trading volume, we used a bar chart which helped identify the most actively traded stocks in the market. The bar chart showed us that AMD was the best stock in the market based on its average trading volume. We were also able to create a framework for generating time-series line plots of stock prices over time for specific stocks, showing trends in daily closing prices. F This allowed for the identification of cyclical patterns and abrupt changes that might require special handling in the modeling phase.

**Summary Statistics:**
Basic statistical analysis was performed to summarize the dataset, this is what we learned from them:

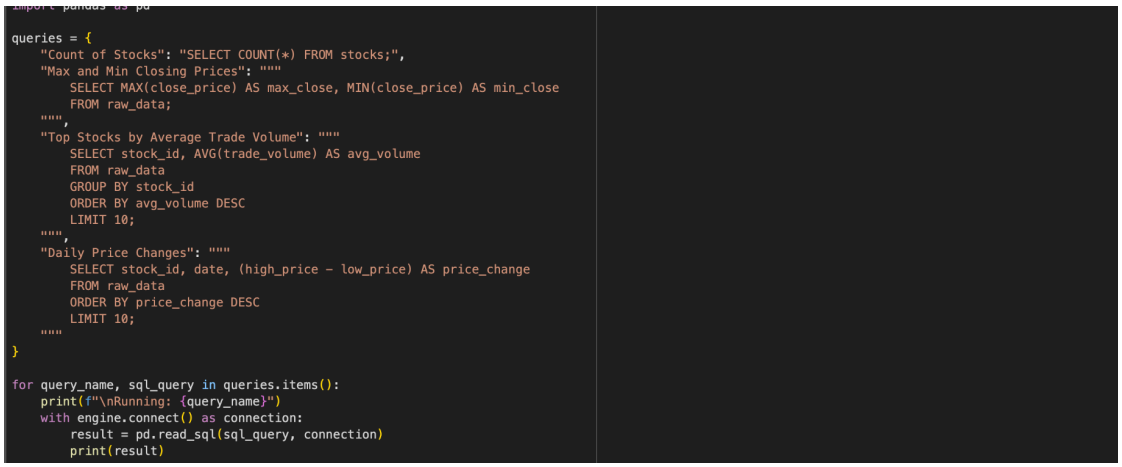
The dataset consists of 619,029 records across various financial metrics, representing daily trading data for S&P 500 companies over several years. The average opening price of stocks was $83.02, with a median of $62.59. Prices ranged from as low as $1.62 to as high as $2,044.00, demonstrating the dataset's inclusion of both low-cap and high-cap stocks.

For the daily closing prices, the mean was $83.04, with a median value of $62.62. The highest recorded closing price was $2,049.00, while the minimum was $1.59, indicating substantial variability in stock performance. The standard deviation of closing prices, at $97.39, further highlights this variability.

Trading volumes also exhibited a wide range, with an average volume of 4.32 million shares per day and a median of 2.08 million shares. The most actively traded day reached a volume of 618.24 million shares, while some days recorded zero trading volume, likely due to holidays or other anomalies.

**Database Usage**

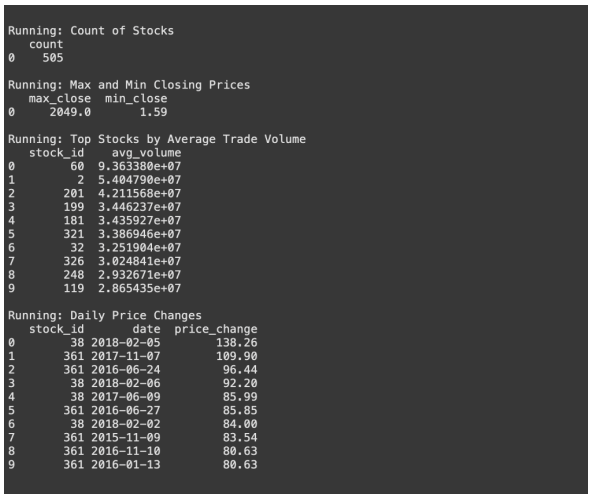
In order to analyze specific financial statistics of the dataset, we loaded all of the data into a database hosted on postgresql and queried the data accordingly. Here is an example of queries being made to find the amount of stocks, the highest and lowest closing prices of a stock, a ranking of top 10 best stocks in terms of average trade volume, and top 10 daily price changes of stocks.



```
import pandas as pd

queries = {
    "Count of Stocks": "SELECT COUNT(*) FROM stocks;",
    "Max and Min Closing Prices": """
        SELECT MAX(close_price) AS max_close, MIN(close_price) AS min_close
        FROM raw_data;
    """,
    "Top Stocks by Average Trade Volume": """
        SELECT stock_id, AVG(trade_volume) AS avg_volume
        FROM raw_data
        GROUP BY stock_id
        ORDER BY avg_volume DESC
        LIMIT 10;
    """,
    "Daily Price Changes": """
        SELECT stock_id, date, (high_price - low_price) AS price_change
        FROM raw_data
        ORDER BY price_change DESC
        LIMIT 10;
    """
}

for query_name, sql_query in queries.items():
    print(f"\nRunning: {query_name}")
    with engine.connect() as connection:
        result = pd.read_sql(sql_query, connection)
        print(result)
```



```
Running: Count of Stocks
   count
0    505

Running: Max and Min Closing Prices
   max_close  min_close
0     2049.0       1.59

Running: Top Stocks by Average Trade Volume
   stock_id    avg_volume
0        60  9.363380e+07
1         2  5.404790e+07
2       201  4.211568e+07
3       199  3.446237e+07
4       181  3.435927e+07
5       321  3.386946e+07
6        32  3.251904e+07
7       326  3.024841e+07
8       248  2.932671e+07
9       119  2.865435e+07

Running: Daily Price Changes
   stock_id       date  price_change
0        38 2018-02-05        138.26
1       361 2017-11-07        109.90
2       361 2016-06-24         96.44
3        38 2018-02-06         92.20
4        38 2017-06-09         85.99
5       361 2016-06-27         85.85
6        38 2018-02-02         84.00
7       361 2015-11-09         83.54
8       361 2016-11-10         80.63
9       361 2016-01-13         80.63
```

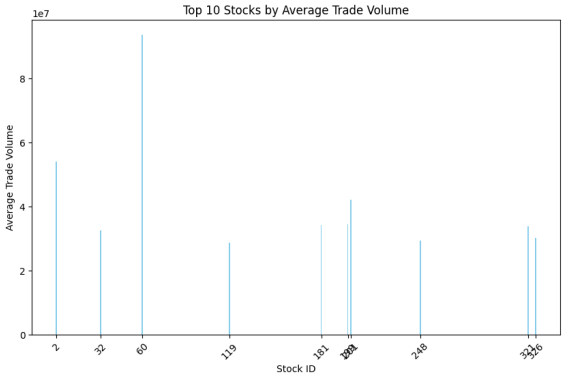
On the left is the output from the queries previously described

The database also allowed for certain visualizations to be created, such as the following graph of the mentioned top 10 stock ranking:

```
query = """
    SELECT stock_id, AVG(trade_volume) AS avg_volume
    FROM raw_data
    GROUP BY stock_id
    ORDER BY avg_volume DESC
    LIMIT 10;
"""

with engine.connect() as connection:
    top_trade_volume = pd.read_sql(query, connection)

# Bar chart
plt.figure(figsize=(10, 6))
plt.bar(top_trade_volume['stock_id'], top_trade_volume['avg_volume'],
plt.title('Top 10 Stocks by Average Trade Volume')
plt.xlabel('Stock ID')
plt.ylabel('Average Trade Volume')
plt.xticks(top_trade_volume['stock_id'], rotation=45)
plt.show()
```




**Feature Engineering and Transformation**

To capture trends and stock behaviors, several indicators were created as features:

- **Moving Averages:** 20-day and 50-day simple moving averages were calculated for each stock. These moving averages are widely used in stock market analysis to smooth out price fluctuations and identify trends.
- **Daily Returns:** Daily returns, calculated as the percentage change in the stock price from one day to the next, were included as a feature to capture short-term market movements.
- **Volatility:** The rolling volatility was computed for each stock by calculating the standard deviation of daily returns over a 20-day window. Volatility is a key measure of risk, as higher volatility generally indicates greater uncertainty in a stock's future price movements.

**Predictive Modeling**

For predictive modeling, a Linear Regression model was chosen due to its simplicity and interpretability. The goal was to predict the next day’s closing price (next_close_price) based on historical stock performance indicators.

```python
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error

# Define features and target
feature_columns = ['daily_return', 'ma_20', 'ma_50', 'volatility']
X = features_df[feature_columns]
y = features_df['next_close_price']

# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Scale the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Train the model
model = LinearRegression()

# Scale the target variable
scaler_target = StandardScaler()
y_train_scaled = scaler_target.fit_transform(y_train.values.reshape(-1, 1)).flatten()

# Train the LinearRegression model with scaled target
model.fit(X_train_scaled, y_train_scaled)

# Evaluate the model
y_pred = model.predict(X_test_scaled)
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
```

The dataset was split into training and testing sets using an 80-20 split. The features were then scaled using the StandardScaler, and the target variable (y, representing the next day's closing price) was also scaled using a separate StandardScaler to avoid bias.

The Linear Regression model was trained on the scaled training data. After fitting the model, predictions were made on the test set, and the model's performance was evaluated using the Mean Squared Error and Mean Absolute Error.

The initial performance metrics indicated that while the model demonstrated some predictive power, there was significant room for improvement. Specifically, the scaled Mean Squared Error (MSE) was 22.751, and the scaled Mean Absolute Error (MAE) was 2.23, both of which are relatively small when compared to the average target value of 84.05. These metrics suggest that, on average, the model's predictions deviated from the actual values by a large margin. However, these values in this context may not be as large as they actually seem. These values are also scaled, with the original values being significantly lower. Additionally, an analysis of feature coefficients revealed that the 20-day moving average (ma_20) had the strongest positive impact on predictions (1.29), while the 50-day moving average (ma_50) contributed negatively (-0.29). Other features, such as daily return (0.009) and volatility (0.00019), had relatively minor effects. These results underscore the need for further optimization, such as incorporating additional features, tuning the model, or testing alternative algorithms to improve predictive accuracy.

Using the predicted closing prices, a new column was added to the dataset that calculated the predicted growth as a percentage of the difference between the predicted closing price and the actual closing price. We were able to create a ranking of the top 10 stocks based on this predicted growth, providing valuable insights into which stocks were expected to perform well in the short term.

To further understand market risk, rolling volatility over a 20-day window was used to rank stocks by their relative risk. The 10 most volatile stocks were identified, highlighting those that investors might want to approach with caution or possibly use for short-term trading strategies.

The **Sharpe ratio**, a measure of risk-adjusted return, was calculated by dividing the predicted growth by the volatility. Stocks with the highest Sharpe ratios were identified as having the most attractive risk-to-reward profiles. Similarly, momentum was measured by the cumulative daily return over a 10-day rolling window, identifying stocks with strong recent performance.

**Results and Insights**

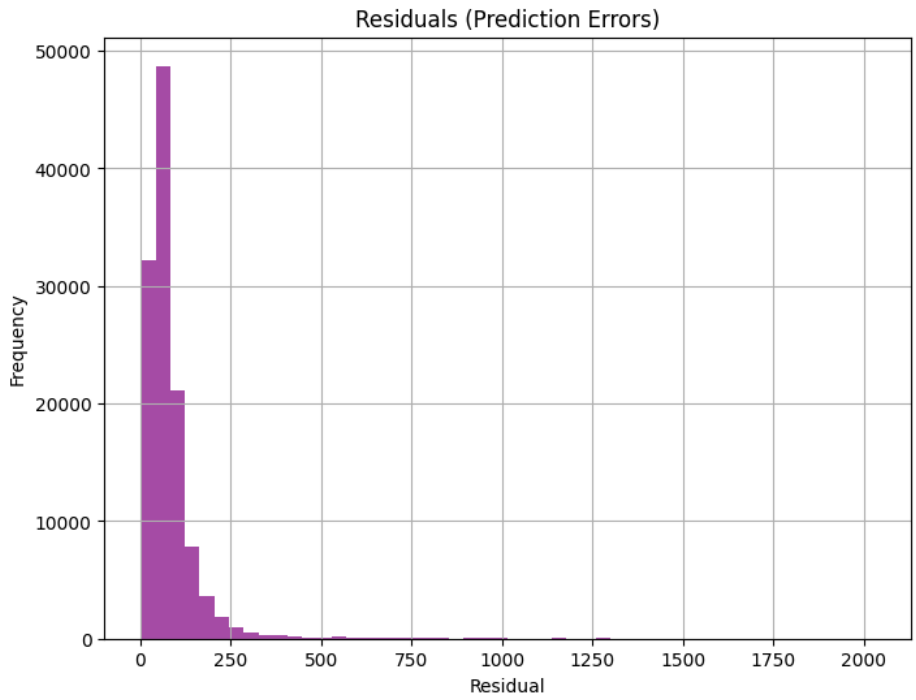
The analysis identified the top 10 stocks with the highest predicted growth, helping investors focus on high-potential opportunities.High momentum stocks and low-volatility stocks were also flagged, providing different strategies for short-term and long-term investments.

By calculating the Sharpe ratio and volatility, the project provided a framework for balancing risk and reward. Investors could use this analysis to make more informed decisions by considering both the potential returns and the risk associated with each stock. The most consistent stocks, characterized by low volatility, were also identified. These stocks are more stable and can be appealing to investors seeking safer, long-term investments.

**Final Visualizations and Reporting**

To communicate findings, a couple of visualizations were created:


The first graph, titled "Actual vs. Predicted Closing Prices," is a scatterplot that compares the actual closing prices on the x-axis with the predicted closing prices on the y-axis. The points closely follow a diagonal line, indicating a strong alignment between the actual and predicted values. This suggests that the model performs well for the majority of predictions. However, as the stock prices increase (particularly beyond 1000), the variance in predictions becomes more noticeable, with some predictions deviating further from the actual values. Despite this, the overall trend demonstrates a high correlation between the actual and predicted prices.




The second graph, "Residuals (Prediction Errors)," is a histogram showing the distribution of residuals, which are the differences between actual and predicted prices. Most residuals are clustered near zero, indicating that the model makes accurate predictions for the majority of stocks. However, the histogram exhibits a positive skew, with a few larger residuals extending to the right. This suggests occasional over-predictions for certain stocks. The frequency of residuals drops sharply beyond 250, indicating that extreme errors are relatively rare.

**Contribution to the Field**

This project contributes to the field of stock market analysis and prediction by integrating traditional financial metrics with modern machine learning techniques. The ability to predict stock prices and assess their potential growth, volatility, and risk provides investors with powerful tools for making informed decisions.

The project's pipeline—from data cleaning to feature engineering, predictive modeling, and advanced analysis—demonstrates how data science and machine learning can be used to analyze and forecast financial markets. By using popular and streamlined tools like scikit-learn, matplotlib, and pandas, the project offers a replicable framework for future financial data analysis.