Rob Ferro (rf599)

Stock Database and Share Price Predictor

Just about everything in our world revolves around money. In our personal lives, it gives us the ability to live freely and enhances not only the quality of our life, but the lives of our loved ones as well. In the professional world, finance influences every decision made from small managerial roles to the boardroom. With this in mind, it should come as no surprise that the United States stock market is one of the most important systems within our society and the ability to perform at a high level within it is one of the most sought-after skills to acquire. If you can accurately predict the trajectory of any given stock and know how to act upon this information, you have a license to print money. Hedge funds and other investment firms are constantly leveraging new technologies such as developing powerful trading algorithms to create an edge in the market and increase profitability.

Over the past few years, I have personally dedicated countless hours of my free time to the art of finance and knew the exact topic that I wanted my final project to cover as soon as it was announced. We briefly touched upon time-series data, specifically stocks, during lectures throughout the semester. In addition, the recitation sessions covering learning models, specifically linear regression models to accurately predict the price of real estate property values, really piqued my interest and got me thinking about how to apply it toward my interest in investing.

The problem that my project aims to solve is to clear all the uncertainty of investing in an individual company within the stock market. I'm sure there are seemingly countless instances where an individual with no experience has trusted a company enough to invest with their hard-earned dollars just for that company's valuation to plummet shortly after their share/option purchase and cause a great deal of financial hardship.

My goal was to build a project that obtains and stores valuable trading metrics to train a learning model that could accurately predict the share price of any individual company given a specified time frame. I would be naïve to believe that this is an entirely original idea for a project, but it is the perfect subject matter to incorporate every aspect of data science that we have learned about throughout the semester during implementation. However, my project is unique in the sense that there are so many metrics related to stock performance that allowed me to formulate my own approach to tackling this problem given my own knowledge.

This project is important because it could give any individual the tools they need to make informed investment decisions to financially prosper regardless of knowledge or experience.

Data and Sources

- **Data Description:**
  - Historical stock price data: (open, high, low, close, adjusted close, and volume).
  - Economic indicator data: (CPI and Federal Funds Interest Rate).
- **Data Sources and File Types:**

- **Stock Price Data:** Retrieved using the Yahoo Finance API through the yfinance library. The data spans the lifespan of the stock's status as a publicly traded company and includes the data from every stock market trading day.
    - **Yahoo Finance API Request Data File Type**: Structured Tabular Data

- **Economic Data:** Collected from the Federal Reserve Economic Data (FRED) API for CPI and federal funds interest rates. The data spans several decades and is updated monthly.
    - **FRED API Request Data File Type**: JSON

Data Types:

- **Historical Stock Data:**
    - **Format:** Pandas dataframe, later stored in SQLite database.
    - **Columns:**
        - date (datetime): The date for each trading record.
        - open (float): Opening price of the stock.
        - high (float): Highest intraday trading price reached during the day.
        - low (float): Lowest intraday trading price reached during day.
        - close (float): Closing price of the stock.
        - adj close (float): Adjusted closing price considering any corporate actions such as stock splits and dividends.
        - volume (integer): Total number of shares traded during the day.
- **Technical Indicators:**
    - **Format:** Calculated using the pandas dataframe containing stock price data, later stored in the SQLite Database.
    - **Columns:**
        - date (datetime): corresponding to the stock price date.
        - symbol (text): Stock ticker symbol.
        - EMA 200, EMA 100, EMA 50, EMA 25 (floats): 200-day, 100-day, 50-day, and 25-day exponential moving averages, which considers the most recent data points more heavily to effectively identify emerging trends better than the Simple Moving Average, which considers all data points within a window equally.

    - MACD (float): Moving Average Convergence Divergence value, a technical indicator to help investors identify price trends, measure trend momentum, and identify entry points for buying or selling. The MACD line is calculated by subtracting the 26-period exponential moving average (EMA) from the 12-period EMA.
    - Signal Line (float): a nine-period EMA of the MACD line.
    - RSI (float): 14-day Relative Strength Index, momentum oscillator that measures the speed and change of price movements on a value system from 0 to 100.

- **Historical Economic Data:**
  - **Format:** Pandas DataFrame, later stored in the SQLite database.
  - **Columns:**
    - date (datetime): Month-end date for each economic indicator report.
    - CPI (float): Consumer Price Index, used as a measure of inflation by reflecting the annual percentage change in the cost to the average consumer in the purchase of goods and services.
    - Interest Rate (float): Federal Funds Effective Interest Rate.
- **Database Structure/Schema:**

I used an SQLite database, which contains three primary tables:

1. **'stock_prices' Table**: Stores raw stock price data from corresponding data in the Pandas data frame.
   - **Columns:** date (DATE), symbol (TEXT), open (REAL), high (REAL), low (REAL), close (REAL), adj_close (REAL), volume (INTEGER)
   - **Primary Key:** (date, symbol)
2. **'technical_indicators' Table:** Stores calculated technical indicators for stocks.
   - **Columns:** date (DATE), symbol (TEXT), EMA_200 (REAL), EMA_100 (REAL), EMA_50 (REAL), EMA_25 (REAL), MACD (REAL), Signal_Line (REAL), RSI_14 (REAL)
   - **Primary Key:** (date, symbol)
3. **'economic_indicators' Table**: Stores corresponding economic data from the Pandas dataframe containing data fetched from the Federal Reserve Economic Data (FRED) API.
   - **Columns:** date (DATE), CPI (REAL), interest_rate (REAL)
   - **Primary Key:** date

Processing/Flow Summary

1. Fetch overall economic data and raw stock price data for individual tickers from FRED and Yahoo finance API requests. The raw data from Yahoo for stocks came in a structured tabular format and the raw data from FRED for economic data came in JSON format.
2. Clean and preprocess the data into a Pandas dataframe by converting raw stock data to a standardized format (rename columns, convert dates to datetime format, ensure numeric types for price and volume columns, filter/drop rows with missing or invalid data. Monthly economic data had to be resampled to daily frequency with forward-filling to align with daily stock data).
3. Calculate technical indicators.
4. Store the processed data into an SQLite database.
5. Query and merge data for analysis and visualization.

Models/Techniques/Algorithms

- **Technical Indicators:**
  I calculated multiple well-known technical indicators that I specifically picked to assess stock performance trends:
  - Exponential Moving Averages - I originally proposed to use the Simple Moving Average values in my model but opted to only use EMA values to identify emerging trends more effectively. These were calculated using the Pandas '.ewm()' method.
  - Moving Average Convergence Divergence (MACD) and its Signal Line - MACD is computed as the difference between the 12-day EMA and the 26-day EMA. The the signal line is the 9-day EMA of the MACD. The '.ewm()' method was also used to obtain these values.
  - Relative Strength Index (RSI) - I utilized the 'pandas_ta' library module to calculate RSI.
- **Machine Learning Model:**
  - We were only taught Linear Regression models throughout the course, which is the simplest approach that assumes a linear relationship between the input variables and the output variable, but I opted to experiment with a more comprehensive approach.
  - I used the Random Forest Regressor model for stock price prediction, which combines multiple decision trees to improve prediction accuracy. The standard decision tree models split data into branches based on feature values, reaching a decision at each node. The Random Forest model handles non-linear relationships on large datasets well and averages the results of the numerous decision trees which helps to enhance robustness and reduce the risk of overfitting.

- Features used: adjusted close price, volume, every EMA value (200, 100, 50, 25 day), MACD, MACD signal line, RSI (14 day), and economic indicators (CPI and Fed Funds interest rate).

Experiments Designed

- **Stock Data Analysis:**
  - Conducted analysis of individual stock tickers to understand trends using technical indicators and how well these align with actual market conditions.
- **Machine Learning Back testing:**
  - My initial plan was to back test the model's performance on historical data by comparing its predictions to actual stock price movements given a specified window of past data. I attempted to develop and implement this but couldn't produce a working method which produced any meaningful results as the task proved to be too complex and was excluded from my final project due to its limitations.
- **Visualization:**
  - I used matplotlib to create graphs to help analyze and interpret stock price trends and indicator behaviors over a specified window of data, or all historical data depending on parameters passed.
  - Plotted charts of stock prices and EMA lines for trend analysis.
  - MACD and Signal Line graphs to study momentum and buy/sell signals.
  - RSI graph with thresholds for overbought/oversold conditions (75 and 25).
  - In my proposal, I stated that I wanted to create a dashboard utilizing Flask but I simply did not have the time to develop this feature due to having no prior experience using it.

Key Findings and Results

- **Hypothesis and Conclusion:**
  I originally hypothesized that technical indicators combined with machine learning models could predict stock prices and provide insight into stock trends. The hypothesis was partially supported:
  - My evaluation of the model was conducted using Mean Squared Error (MSE), Mean Absolute Error (MAE), and R^2 score.
  - My model demonstrated strong performance metrics through high R^2 scores and low mean squared error numbers during training and testing phases.

- Visualization highlighted meaningful trends, such as alignment between RSI values and price movements, and the effectiveness of EMA lines in tracking trends.

Advantages and Limitations

- **Advantages:**
  - **Flexible:** The control flow of the program allows you to add as many specific stocks to analyze as SQLite can handle. The database makes it easy to adjust the window of time that you wish to predict and/or visualize for assessment.
  - **Visualization:** Enabled intuitive analysis of stock trends and technical indicators through clear visual representations.
  - **Prediction Accuracy:** The Random Forest model produced high performance metrics.
- **Limitations:**
  - **Lack of back testing:** I was unable to successfully create a method to compare model predictions with actual stock movements over time, which would give more insight into model performance and help to identify areas for improvement.
  - **Overfitting Risk:** The strong performance metrics might indicate overfitting.
  - **Real-World Generalizability:** Unfortunately, fundamental technical analysis and economic data do not account for external market factors. The stock market of the modern age is extremely susceptible to make violent price movements driven by developments within individual companies, geopolitical factors and major news events which are very difficult to account for when developing a predictive learning model.

Screenshots of key outputs:

1: processStockData('PLTR', 'stonks.db')

```
[14]: processStockData('PLTR', 'stonks.db')
```

```
[*********************100%***********************]  1 of 1 completed
PLTR DataFrame Sample:

         date  adj_close  close   high   low   open     volume
0  2020-09-30       9.50   9.50  11.41  9.11  10.00  338584400
1  2020-10-01       9.46   9.46  10.10  9.23   9.69  124297600
2  2020-10-02       9.20   9.20   9.28  8.94   9.06   55018300
3  2020-10-05       9.03   9.03   9.49  8.92   9.43   36316900
4  2020-10-06       9.90   9.90  10.18  8.90   9.04   90864000
            date  adj_close      close       high        low       open  \
1050  2024-12-03  70.959999  70.959999  71.370003  66.150002  66.410004
1051  2024-12-04  69.849998  69.849998  71.180000  67.279999  71.129997
1052  2024-12-05  71.870003  71.870003  72.980003  69.889999  70.110001
1053  2024-12-06  76.339996  76.339996  76.820000  72.279999  72.949997
1054  2024-12-09  72.230003  72.230003  80.910004  71.050003  80.580002

         volume
1050  100751400
1051   86284800
1052   66585800
1053   92566000
1054  131544678


Technical Indicators Sample:

         date   EMA_200   EMA_100    EMA_50    EMA_25      MACD  Signal_Line  \
0  2020-09-30  9.500000  9.500000  9.500000  9.500000  0.000000     0.000000
1  2020-10-01  9.499602  9.499208  9.498431  9.496923 -0.003191    -0.000638
2  2020-10-02  9.496621  9.493283  9.486728  9.474083 -0.026395    -0.005790
3  2020-10-05  9.491978  9.484109  9.468817  9.439923 -0.057836    -0.016199
4  2020-10-06  9.496038  9.492345  9.485726  9.475313 -0.012408    -0.015441

   RSI_14
0     NaN
1     NaN
2     NaN
3     NaN
4     NaN


Data for PLTR successfully stored and cleaned.
Database connection closed.
```

2.) economicData = fetchEconomicData(fredAPIKey)

(fredAPIKey = 'f9ac6d2c51f5d5aeea0b07652fd8a52f')

```
[17]: economicData = fetchEconomicData(fredAPIKey)
```

```
Fetching data for CPI (CPIAUCSL)...
Fetching data for interestRate (FEDFUNDS)...
CPI and Interest Rate data successfully fetched.
Economic DataFrame Sample:

         date    CPI  interestRate
0  1947-01-01  21.48           NaN
1  1947-02-01  21.62           NaN
2  1947-03-01  22.00           NaN
3  1947-04-01  22.00           NaN
4  1947-05-01  21.95           NaN
           date      CPI  interestRate
930  2024-07-01  313.534          5.33
931  2024-08-01  314.121          5.33
932  2024-09-01  314.686          5.13
933  2024-10-01  315.454          4.83
934  2024-11-01      NaN          4.64
```

3: storeEconomicData(economicData, ‘stonks.db’)

[19]:
```python
# Store Economic data in the database
storeEconomicData(economicData, 'stonks.db')
```

```
Economic data successfully cleaned and stored.
Database connection closed.
```

4: validateStockData('PLTR', 'stonks.db')

[21]:
```python
# Make sure stock and economic data were successfully stored in the database
validateStockData('PLTR', 'stonks.db')
```

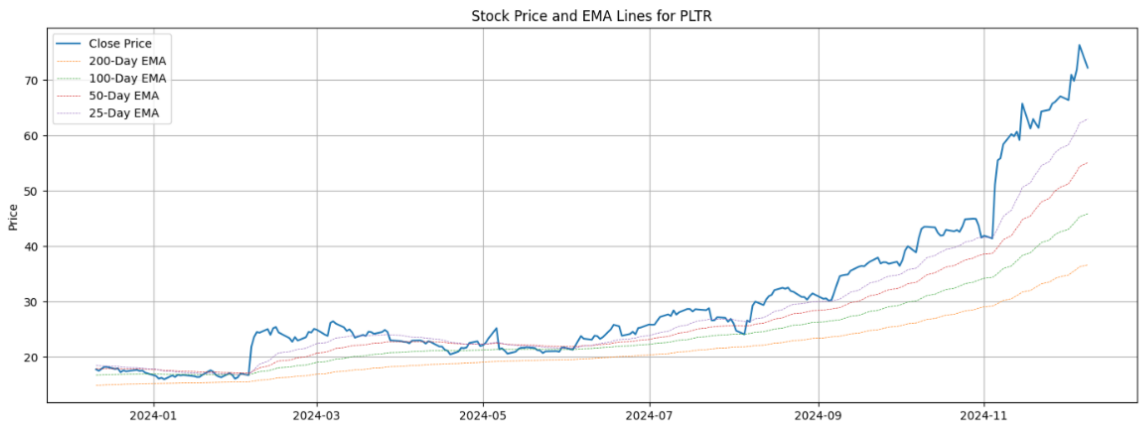
```
Stock Prices Sample for PLTR:
         date symbol   open   high   low  close  adj_close     volume
0  2020-09-30   PLTR  10.00  11.41  9.11   9.50       9.50  338584400
1  2020-10-01   PLTR   9.69  10.10  9.23   9.46       9.46  124297600
2  2020-10-02   PLTR   9.06   9.28  8.94   9.20       9.20   55018300
3  2020-10-05   PLTR   9.43   9.49  8.92   9.03       9.03   36316900
4  2020-10-06   PLTR   9.04  10.18  8.90   9.90       9.90   90864000


Technical Indicators Sample for PLTR:
         date symbol   EMA_200   EMA_100    EMA_50    EMA_25      MACD  \
0  2020-09-30   PLTR  9.500000  9.500000  9.500000  9.500000  0.000000
1  2020-10-01   PLTR  9.499602  9.499208  9.498431  9.496923 -0.003191
2  2020-10-02   PLTR  9.496621  9.493283  9.486728  9.474083 -0.026395
3  2020-10-05   PLTR  9.491978  9.484109  9.468817  9.439923 -0.057836
4  2020-10-06   PLTR  9.496038  9.492345  9.485726  9.475313 -0.012408

   Signal_Line  RSI_14
0     0.000000     NaN
1    -0.000638     NaN
2    -0.005790     NaN
3    -0.016199     NaN
4    -0.015441     NaN



Missing Values in stock_prices Table:
date         0
symbol       0
open         0
high         0
low          0
close        0
adj_close    0
volume       0
dtype: int64



Missing Values in technical_indicators Table:
date            0
symbol          0
EMA_200         0
EMA_100         0
EMA_50          0
EMA_25          0
MACD            0
Signal_Line     0
RSI_14         14
dtype: int64



Duplicate Entries in stock_prices Table:
No duplicates found.

Duplicate Entries in technical_indicators Table:
No duplicates found.
Validation completed and database connection closed.
```

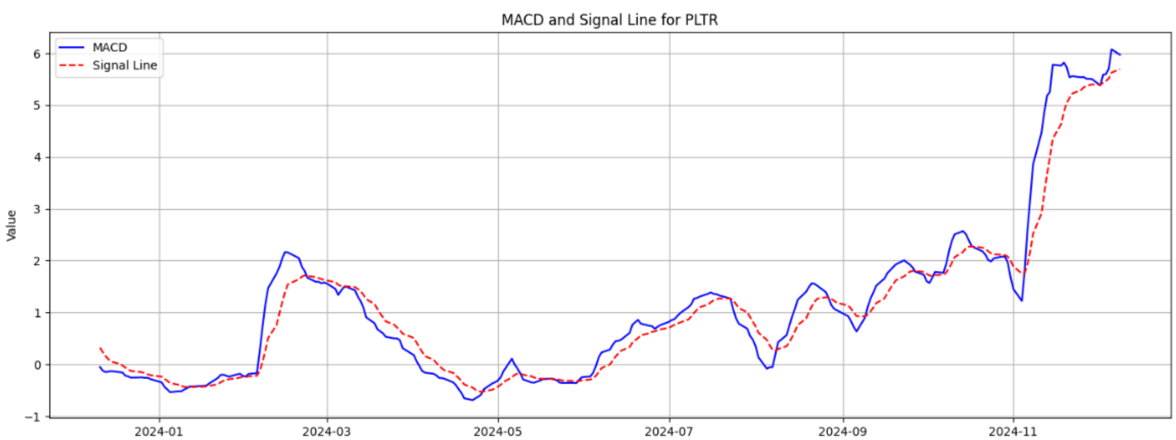
5: predictStockPrice('PLTR', 30, 'stonks.db')

```
[23]: predictStockPrice('PLTR', 30, 'stonks.db')

      DataFrame Columns: Index(['date', 'adj_close', 'open', 'high', 'low', 'close', 'volume',
             'EMA_200', 'EMA_100', 'EMA_50', 'EMA_25', 'MACD', 'Signal_Line',
             'RSI_14', 'symbol'],
            dtype='object')

      Model Performance:
      MSE: 3.09, MAE: 0.91, R2: 0.96
      Predicted price 30 days ahead: $72.18

[23]: (RandomForestRegressor(random_state=17),
       3.087124529546909,
       0.9137005254233758,
       0.9635694245556993,
       72.18250137329102)
```

6: plotStockData('PLTR', 'stonks.db', 365)

```
[25]: plotStockData('PLTR', 'stonks.db', 365)
```





Sources:

Federal Reserve for Economic Data:
[Federal Reserve Economic Data | FRED | St. Louis Fed]
Information on EMA technical indicator:
https://www.investopedia.com/terms/e/ema.asp

Information on MACD and Signal Line:
https://www.investopedia.com/terms/m/macd.asp

Information on RSI technical indicator:
https://www.mssqltips.com/sqlservertip/7079/relative-strength-index-time-series-data-sql-server/

Blog post that helped me consider which would be the best learning model to use:
https://modelsai.org/blog/best-machine-learning-model-for-stock-prediction