Project: Stock Ticker Recommendation and Prediction with Dashboard Data Mining - Aryan Mishra - GH1027140

[Github Link: https://github.com/aryanwise/Stock-Predictions]

> Follow along the README file to install the required libraries to run the Stock Analysis and Recommendation Dashboard

The Problem: High Costs and Limited Access to Personalized Investment Guidance

Nowadays, individual investors often face significant hurdles in managing their portfolios effectively. Traditional investment advisory services and personal brokers who offer their valuable expertise, come with substantial fees that can be a bit expensive, especially for those with smaller portfolios. This financial barrier often leaves many investors without the personalized insights and continuous portfolio monitoring needed to make informed decisions and minimize risk.

Furthermore, readily available investment information can be overwhelming and difficult to interpret without specialized knowledge. Investors often struggle to conduct thorough research, perform complex technical analysis, or get real-time market sentiment, leading to suboptimal investment choices and increased exposure to risk. This lack of accessible, affordable, and actionable guidance prevents many from confidently managing their own investments and achieving their financial goals.

About the Project:

This project addresses the critical need for accessible and affordable personalized investment guidance. Many individual investors are currently priced out of traditional advisory services, leaving them without the crucial insights and risk management strategies needed to confidently navigate the market.

To tackle this, I've developed an AI-powered assistant designed to function like a personal broker, helping investors research independently using technical analysis and receive tailored suggestions based on sentiment data from social media(in this case- twitter).

The goal is to empower users to maintain a balanced portfolio and minimize risk, ultimately saving them the significant fees associated with personal brokers or investment management companies.

My solution is a user-friendly dashboard that provides **technical analysis** for various ticker stocks and offers **recommendations based on sentiment derived from Twitter data.** This

integrated approach offers investors a cost-effective way to gain valuable market insights and make more informed decisions.

About the data:
(https://github.com/aryanwise/Stock-Predictions/tree/main/Datasets)

Historical Price Data:

- I have taken historical data of selected tickers [AAPL, AMZN, NFLX, META, GOOGL] from the NASDAQ website for over 10 years, per day price.
- The data is from 06/15/2015 util 06/13/2025
- Includes column: Date,Close/Last,Volume,Open,High,Low

Dataset Link:

AAPL:
https://www.nasdaq.com/market-activity/stocks/aapl/historical?page=1&rows_per_page=10&timeline=y10

AMZN:
https://www.nasdaq.com/market-activity/stocks/amzn/historical?page=1&rows_per_page=10&timeline=y10

NFLX:
https://www.nasdaq.com/market-activity/stocks/nflx/historical?page=1&rows_per_page=10&timeline=y10

META:
https://www.nasdaq.com/market-activity/stocks/meta/historical?page=1&rows_per_page=10&timeline=y10

GOOGL:
https://www.nasdaq.com/market-activity/stocks/googl/historical?page=1&rows_per_page=10&timeline=y10

Sentiment Data:

- I have taken the Stock-Market Sentiment Dataset from Kaggle (link-> https://www.kaggle.com/datasets/yash612/stockmarket-sentiment-dataset)
- Description: Gathered Stock news from Multiple twitter Handles regarding Economic news dividing into two parts :

- Negative(-1) and Positive(1) .
- Negative count: 2,106
- Positive count: 3,685

How the data will be used:

- Historical price data will be used in calculating technical indicators using python (ta-library….https://technical-analysis-library-in-python.readthedocs.io/en/latest/ta.html)
- Again using the historical price data in training the model using linear regression model.
- Sentiment Data will be used to train the model and to calculate the sentiment score and generate recommendation of top positive stocks.

Project Objective:

Technical Analysis: Using technical analysis for predicting next day/week price using historical price data of tickers.

Training Model: To determine whether incorporating technical indicators into a predictive model can outperform a simple "previous day's closing price" (naive baseline) model in forecasting the next day's stock closing price.

Actionable Investment Recommendations: To provide clear, data-backed recommendations (e.g., 'Buy', 'Sell', 'Hold', 'Strong Buy', 'Strong Sell') by integrating both technical analysis and social media sentiments.

Interactive Dashboard for Insight Visualization: Created a user-friendly and interactive dashboard using streamlit that visualizes key market trends, and the rationale behind recommendations, making complex financial data accessible to diverse audiences, including non-technical stakeholders.

Project Scope:

Core AI Assistant Functionality:

- Provides personalized investment insights and suggestions.
- Aims to help investors balance their portfolios.
- Works towards minimizing investment risk.
- Enables cost savings by reducing reliance on personal brokers or investment management companies.

Dashboard Features:

- Includes technical analysis for various ticker stocks.
- Generates recommendations based on sentiment derived from tweets.

Data Collection:

For an in depth explanation, please check the **data_collection.py(**https://github.com/aryanwise/Stock-Predictions/blob/main/Main/data_collection.py **)** along with **test_data.py(**https://github.com/aryanwise/Stock-Predictions/blob/main/Main/test_data.py **)** to test the data loading and historical tables.

I've implemented a **StockDataCollector** class to efficiently gather and manage historical stock data. Here's how it works:

1. Initialization: I set up the base directory and specify the path to my historical stock data, which is stored in CSV files.
2. Data Collection: I've designed a collect_data method that scans a designated folder for CSV files. It specifically looks for files starting with "HistoricalData_" to identify and extract ticker symbols.
3. Ticker Extraction: A private helper method, _extract_ticker, is used internally to extract the stock ticker from the filename.
4. Data Access: I've provided methods (get_raw_data and get_stock_data) to easily access either the raw CSV data by filename or the processed stock data by ticker symbol.

Data Cleaning:

For an in depth explanation, please check the **data_collection.py(**https://github.com/aryanwise/Stock-Predictions/blob/main/Main/data_collection.py **)** along with **test_data.py(**https://github.com/aryanwise/Stock-Predictions/blob/main/Main/test_data.py **)** to test the data loading and historical tables.

I've developed a **StockDataCleaner** class to ensure the historical stock data is properly formatted and ready for analysis.

Here's how I approach the cleaning process:

1. Individual DataFrame Cleaning: I have a static method, clean_data, that takes a single stock DataFrame. It renames the "Close/Last" column to "Close", removes non-numeric characters from price-related columns ("Close", "Open", "High", "Low") and converts them to float, and finally converts the "Date" column to datetime objects, sorting them as needed.

2. Batch Data Cleaning: To streamline the process, I've implemented another static method, clean_all. This method iterates through all the stock DataFrames collected by the StockDataCollector and applies the clean_data method to each one, returning a dictionary of cleaned DataFrames, indexed by their respective ticker symbols.

Exploratory data analysis:

EDA for Stock Sentiment Analysis:

For an in depth explanation, please check the **stock_sentiment_analysis.ipynb (**https://github.com/aryanwise/Stock-Predictions/blob/main/Notebooks/stock_sentiment_analysis.ipynb **).**

I've conducted a thorough Exploratory Data Analysis (EDA) on the stock sentiment data to understand its characteristics, relationships, and patterns. My analysis involved both statistical techniques and various visualizations.

Here's a breakdown of my findings:

- Data Overview and Preprocessing:
    - I loaded a dataset from `stock_sentiment_data.csv` containing tweet text and a corresponding sentiment label (1 for positive, -1 for negative).
    - I observed that the dataset contains 5,791 records, with a clear imbalance: 3,685 positive sentiment entries and 2,106 negative sentiment entries.
    - My `preprocess_text` function handles cleaning by removing mentions, URLs, and non-alphabetic characters, converting text to lowercase for consistency.
    - I also implemented an `extract_tickers` function using regular expressions to identify stock tickers within the text, filtering out common English words to ensure accuracy.
- Sentiment Distribution and Balance:
    - I created a pie chart to visualize the sentiment distribution in the training data, which clearly showed a 63.6% positive to 36.4% negative split.
    - This imbalance suggests that resampling might be beneficial to prevent bias in the classifier.

- Model Training and Evaluation:
    - I trained a Logistic Regression model on the preprocessed text data, using TF-IDF vectorization to extract features.
    - The model was evaluated on a 20% test set, achieving an overall accuracy of 78.8%.
    - Detailed Class-wise Metrics:
        - For negative sentiment (-1), the precision was 0.83 (correct 83% of the time when predicting negative) and recall was 0.53 (identified 53% of actual negatives).
        - For positive sentiment (1), the precision was 0.77 (correct 77% of the time when predicting positive) and recall was 0.94 (successfully captured 94% of actual positives).
    - Confusion Matrix Analysis: The confusion matrix revealed a higher success rate in correctly identifying positive sentiments (732 True Positives) compared to negative sentiments (427 True Negatives). This confirms the disparity in recall, indicating the model missed a notable portion of genuinely negative signals.
- Feature Importance (Predictive Words):
    - I generated word clouds for both positive and negative sentiments to visually inspect the most frequent words.
    - By analyzing Logistic Regression coefficients, I identified the top predictive words:
        - Positive words such as 'trade', 'stock', 'gain', 'long', and 'good' were strongly correlated with positive sentiment.
        - Negative words like 'short', 'bear', 'fall', 'drop', and 'risk' significantly contributed to negative sentiment predictions. This confirms that the tokenization successfully captures finance-specific sentiment cues.
- Stock Sentiment Analysis and Recommendation Generation:
    - I analyzed the sentiment for each stock ticker found in the dataset, aggregating mentions to calculate average sentiment and total sentiment.
    - My `generate_recommendations` method calculates a 'sentiment score' (normalized between 0 and 1) and a 'confidence' score based on mention count and sentiment consistency.
    - I then generated a list of the top 5 stock recommendations, displaying metrics such as positive/negative mention counts, sentiment score, and confidence.
- Statistical Summary of Recommendations:
    - I calculated and presented key statistics for the top 5 recommendations, including the average sentiment score, median confidence level, total mentions, and the positive-to-negative mention ratio. For example, the average sentiment score was found to be 1.000, and the median confidence was 0.408. The Positive/Negative Ratio was 10.90:1.

- Individual Stock Analysis:
    - For each recommended stock, I provided a detailed breakdown including bullish consensus, sentiment strength, reliability (confidence score), and sample mentions. For instance, I can see the percentage of positive mentions and how far the sentiment score is above neutral.
- Comparative Analysis:
    - I identified stocks with the strongest sentiment, most consistency, and highest controversy based on sentiment scores, confidence levels, and mention ratios. This allows for quick identification of "Strong Buy" candidates and potential "sell" flags.
- Risk Assessment:
    - I presented a risk assessment by visualizing the sentiment distribution among the top picks, showing the percentage of positive versus negative mentions for each ticker.
- Visualizations (Post-Recommendation):
    - Normalized Sentiment Scores (Bar Chart): This chart shows all five top tickers scoring 1.0, indicating a unanimous bullish consensus and "Strong Buy" signals.
    - Confidence Levels (Bar Chart): The confidence levels across the top picks hovered around 0.40-0.41, suggesting that more mentions generally lead to higher trust in the sentiment signal.
    - Positive vs. Negative Mentions (Stacked Bar Chart): This visualization confirmed 100% positive mentions and zero negative mentions for the top recommended stocks, further validating their bullish stance.
    - Stock Sentiment vs. Mention Frequency (Bubble Chart): This chart blended sentiment strength with discussion volume. Large green bubbles on the right indicate hyped "buy" signals.

EDA for Stock Prediction:

For an in depth explanation, please check **stock_prediction_linear_regression.ipynb (**https://github.com/aryanwise/Stock-Predictions/blob/main/Notebooks/stock_prediction_linear_regression.ipynb **).**

I've conducted an Exploratory Data Analysis (EDA) for stock price prediction using Apple (AAPL) historical data. My primary objective was to see if incorporating technical indicators could improve forecasting accuracy over a simple "previous day's closing price" baseline using a Linear Regression model.

Here's how I approached the analysis and what I found:

- Data Acquisition and Preparation:
    - I collected historical daily stock data for AAPL using my `StockDataCollector` and `StockDataCleaner` classes, ensuring the data was clean and standardized.
    - I engineered a `Previous_Close` feature to serve as a direct baseline comparison, being careful to avoid data leakage.
    - I calculated various technical indicators using my `TechnicalIndicators` module. These included: Simple Moving Average (SMA), Exponential Moving Average (EMA), MACD, RSI, Stochastic Oscillators, Bollinger Bands, and ATR (Average True Range).
    - I removed any rows with `NaN` values that were introduced by the shifting and indicator calculations to ensure data integrity for modeling.
- Model Building and Validation (Walk-Forward Validation):
    - I defined a comprehensive set of features for my Linear Regression model, including lagged prices and all the calculated technical indicators.
    - I employed a rigorous walk-forward validation strategy. This method simulates real-world trading by incrementally training the model on a `train_window_size` (approximately one year of trading days) and then predicting for the `test_window_size` (approximately one month) without any data leakage.
    - I collected predictions from both the Linear Regression model and the "previous day's close" baseline, along with the actual prices, for later evaluation.
- Evaluation and Key Findings:
    - Mean Absolute Error (MAE): The Linear Regression model achieved an MAE of 0.419, significantly outperforming the Baseline's MAE of 1.192. This means my model's predictions were, on average, $0.77 closer to the actual closing prices.
    - Mean Squared Error (MSE): My Linear Regression model also showed a much lower MSE of 0.395 compared to the Baseline's 3.649. This indicates that the Linear Regression model is more robust and less prone to large prediction errors.
- Visual Insights:
    - Prediction vs. Actuals Plot: I visualized the actual close price (white line), Linear Regression predictions (lime green line), and Baseline predictions (yellow line). This plot clearly demonstrated that the Linear Regression predictions closely

followed the actual prices, while the baseline lagged significantly during price swings.
    - Daily Mean Absolute Error (MAE) Plot: I plotted the daily MAE for both models. The Linear Regression's daily errors consistently remained lower than those of the baseline, further confirming its superior accuracy.
- Conclusion:
    - My analysis provides strong evidence that incorporating technical indicators into a Linear Regression model significantly outperforms a naive "previous close" strategy for forecasting AAPL's next-day closing prices.
    - The lower MAE and MSE metrics, along with the visual confirmations, validate that technical indicators contain valuable predictive signals when used appropriately.

Stock Prediction Failed Model: LSTM

For an in depth explanation, please check **LSTM_prediction.ipynb** (https://github.com/aryanwise/Stock-Predictions/blob/main/Notebooks/LSTM_prediction.ipynb).

When I initially approached stock prediction, I started with an LSTM (Long Short-Term Memory) model. While it seemed promising, my analysis revealed significant challenges that led me to explore alternative approaches.

Exploratory Data Analysis (EDA) - Stock Prediction (LSTM Model Failure Analysis)

My initial stock prediction model utilized an LSTM neural network, and through extensive analysis, I identified key reasons for its underperformance, particularly in the context of predicting actual price movements.

- Data Preparation for LSTM:
    - I loaded and cleaned historical data for META (Meta Platforms, Inc.) using my `StockDataCollector` and `StockDataCleaner`.
    - I enriched the dataset with various technical indicators, including SMA, EMA, MACD, RSI, Stochastic Oscillators, Bollinger Bands, and ATR, using my `TechnicalIndicators` module.
    - Crucially, I defined my target variable (`Target`) as the difference between the next day's closing price and the current day's opening price. This was a deliberate shift from simply predicting the next day's close, which can lead to an "illusion of accuracy" due to price inertia.

    - I then normalized the data using `MinMaxScaler` to scale all features between 0 and 1, a standard practice for neural networks like LSTMs.
    - The input data (`X`) was structured as sequences of 30 past trading days (`backcandles`), with 15 features per day, and the output (`y`) was the target price difference. I split this data into 80% for training and 20% for testing.
- Core Challenges Identified:
    - Illusion of Accuracy with a Bad Target Variable: My initial attempts at predicting the "next-candle close price" yielded charts that appeared almost perfect. However, this was misleading, as the model was simply mirroring price inertia, offering no real trading advantage. This highlighted that predicting raw price values doesn't capture meaningful market movement.
    - Difficulty Forecasting Price Movement: When I shifted the target to predict the actual "price change" (direction and magnitude), the model's accuracy significantly collapsed. This exposed the LSTM's struggle to capture the inherent stochastic and non-linear nature of financial markets.
- Model Architecture and Training:
    - I constructed an LSTM model with an input layer shaped to receive sequences of `backcandles` and `15` features, followed by an LSTM layer with 150 units, a dense layer, and a linear activation output.
    - The model was compiled with the Adam optimizer and a mean squared error (MSE) loss function, and trained for 30 epochs with a batch size of 15.
- Visual Analysis of Failure:
    - I generated a plot comparing the actual test data (`y_test`) against the LSTM model's predictions (`y_pred`).
    - The "Test" line (black) representing true market price movements was highly volatile with sharp, rapid changes, including a notable spike. This confirmed the stochastic nature of financial markets.
    - In contrast, the "pred" line (green) representing my model's predictions appeared significantly smoother. While it tracked the general trend, it consistently failed to capture sharp fluctuations and entirely missed large spikes or outliers.
    - This visual evidence clearly reinforced the model's limitations: it could follow slow-moving, large-scale directional shifts but exhibited limited responsiveness to short-term volatility and was not robust to sudden price shocks.
- Conclusion on LSTM Failure:
    - The analysis confirmed that while the LSTM model could identify general trends, it was not effective at predicting the precise, volatile, and stochastic nature of stock price movements, especially when the target was price change. This gap between the predicted and actual price movements visually encapsulated the difficulty of predicting chaotic systems with standard LSTM models without further enhancements.

- Improvement Roadmap (Learnings from Failure):
    - To overcome these limitations, my roadmap for improvement included:
        - Target Definition: Focusing on predicting *price movement* (change in close price) instead of raw prices to achieve a more realistic and tradable objective.
        - Feature Engineering: Adding more advanced technical indicators and incorporating alternative data like news and sentiment (which I later implemented in my other model) to enrich the signal space beyond simple OHLC (Open, High, Low, Close) data.
        - Model Architecture: Tuning LSTM depth, units, and dropout to better capture temporal patterns.
        - Look-back Window: Optimizing the sequence length, as too short a window misses trends, and too long introduces noise.
        - Hyperparameter Tuning: Performing comprehensive grid, random, or Bayesian searches for learning rates, batch sizes, and other parameters to find optimal learning.

This detailed analysis of the LSTM model's shortcomings directly informed my decision to explore and ultimately implement the Linear Regression model with technical indicators, as detailed in my successful stock prediction analysis.

Model Selection and Training:

I've carefully selected and trained models for both stock sentiment analysis and stock price prediction, prioritizing effectiveness and interpretability based on my exploratory data analysis and prior model evaluations.

Stock Sentiments:

For an in depth explanation, please check the **stock_sentiment_analysis.ipynb (**https://github.com/aryanwise/Stock-Predictions/blob/main/Notebooks/stock_sentiment_analysis.ipynb **).**

For stock sentiment analysis, I opted for a Logistic Regression classifier.

- Rationale: My initial EDA revealed a clear distinction between positive and negative sentiment words, making Logistic Regression a suitable choice due to its interpretability and efficiency for binary classification tasks. It allows me to understand the contribution of individual words to the sentiment.
- Data Preparation: I preprocessed the tweet text by cleaning it (removing mentions, URLs, non-alphabetic characters) and extracting relevant stock tickers. Features were then extracted using TF-IDF vectorization, which weighs words based on their frequency and importance across the dataset.
- Training and Evaluation:
    - The model was trained on a dataset of stock-related tweets, with sentiment labels (1 for positive, -1 for negative).
    - I split the data into 80% for training and 20% for testing.
    - During training, I monitored the model's performance using accuracy, precision, recall, and F1-score. The model achieved an overall accuracy of 78.8%.
    - A confusion matrix was used to visualize the true positives, true negatives, false positives, and false negatives, providing deeper insight into the model's performance on each sentiment class. This helped confirm its stronger performance in identifying positive sentiments compared to negative ones.

Stock Prediction (Linear Regression)

For an in depth explanation, please check **stock_prediction_linear_regression.ipynb (**https://github.com/aryanwise/Stock-Predictions/blob/main/Notebooks/stock_prediction_linear_regression.ipynb **).**

For stock price prediction, I chose a Linear Regression model, specifically for forecasting the next day's closing price of Apple (AAPL).

- Rationale: My previous attempt with an LSTM model for price change prediction highlighted the challenges of capturing the stochastic nature of stock movements. The Linear Regression model, when augmented with robust features, offers a more stable and interpretable approach for forecasting, allowing me to directly assess the impact of various technical indicators.
- Data Preparation: I collected and cleaned historical daily stock data for AAPL. A crucial step involved feature engineering, where I incorporated a variety of technical indicators such as Simple Moving Average (SMA), Exponential Moving Average (EMA), MACD, RSI, Stochastic Oscillators, Bollinger Bands, and Average True Range (ATR). I also created a `Previous_Close` feature for baseline comparison.
- Training and Validation Strategy:

    - I employed a rigorous walk-forward validation approach to simulate real-world trading conditions and prevent data leakage. This involved incrementally training the model on a `train_window_size` (approximately one year of trading days) and then predicting for the `test_window_size` (approximately one month).
    - I defined a clear target variable: the next day's `Close` price.
- Evaluation: The model's performance was evaluated using Mean Absolute Error (MAE) and Mean Squared Error (MSE). The Linear Regression model significantly outperformed a naive "previous day's closing price" baseline, demonstrating a lower MAE and MSE, indicating greater accuracy and reduced prediction variance.

Evaluation:

My evaluation process for both the sentiment analysis and stock prediction models provided crucial insights into their performance and areas for future refinement.

Stock Sentiment Evaluation Insights

For an in depth explanation, please check the **stock_sentiment_analysis.ipynb (**https://github.com/aryanwise/Stock-Predictions/blob/main/Notebooks/stock_sentiment_analysis.ipynb **).**

The evaluation of the Logistic Regression sentiment model important observations regarding its effectiveness in classifying financial sentiment:

- Overall Strong Performance: The model achieved an accuracy of 78.8%, indicating that it correctly classified the sentiment of tweets nearly 79% of the time on unseen data. This suggests a robust ability to learn general market sentiment from text.
- Asymmetry in Class Performance:
    - High Recall for Positive Sentiment: The model demonstrated a high recall of 0.94 for positive sentiment. This means it was highly effective at identifying genuinely positive signals, capturing 94% of all actual positive sentiments. This is particularly valuable for identifying bullish trends.
    - Lower Recall for Negative Sentiment: Conversely, the recall for negative sentiment was 0.53. While its precision for negative predictions was high (0.83), meaning fewer false negative alarms, it missed a notable portion of truly negative signals. This suggests that the model is more cautious in labeling sentiment as negative, potentially overlooking some bearish indicators.
- Imbalance Impact: The training data had a significant class imbalance (63.6% positive vs. 36.4% negative). This imbalance likely contributed to the model's stronger performance

on the positive class and its tendency to be more conservative with negative predictions. Future improvements could involve resampling techniques to balance the dataset.

- Interpretability of Predictive Features: The analysis of word clouds and Logistic Regression coefficients provided clear insights into which terms strongly correlated with positive (e.g., 'trade', 'stock', 'gain', 'long', 'good') and negative (e.g., 'short', 'bear', 'fall', 'drop', 'risk') sentiments. This interpretability is valuable for understanding the market narrative.

Stock Prediction (Linear Regression) Evaluation Insights

For an in depth explanation, please check **stock_prediction_linear_regression.ipynb (**https://github.com/aryanwise/Stock-Predictions/blob/main/Notebooks/stock_prediction_linear_regression.ipynb **).**

The evaluation of the Linear Regression model for stock price prediction demonstrated a clear advantage over the simple baseline:

- Superior Accuracy over Baseline: The Linear Regression model, incorporating technical indicators, significantly outperformed the naive "previous day's closing price" baseline.
    - It achieved a Mean Absolute Error (MAE) of 0.419, which is substantially lower than the baseline's MAE of 1.192. On average, my model's predictions were about $0.77 closer to the actual closing prices than the baseline.
    - The Mean Squared Error (MSE) of 0.395 for Linear Regression was also much lower than the baseline's 3.649. This indicates that my model is more robust and less prone to large, potentially costly prediction errors.
- Predictive Value of Technical Indicators: The strong performance of the Linear Regression model confirms that the technical indicators I engineered (SMA, EMA, MACD, RSI, etc.) contain significant predictive signals when used appropriately. This contrasts sharply with the earlier LSTM attempt, which struggled without these contextual features.
- Stability and Consistency: Visual analysis showed that the Linear Regression predictions closely followed the actual stock prices, while the baseline consistently lagged, especially during price swings. Furthermore, the daily MAE for Linear Regression remained consistently lower than the baseline, confirming its superior accuracy day-to-day.
- Foundation for Further Development: While a simple Linear Regression model, its effectiveness proves that a well-chosen model combined with meaningful features can deliver tangible predictive value. This forms a solid foundation for exploring more complex models or incorporating additional data sources in the future.

Results:

My project highlights several key findings:

- I've laid the groundwork for an **AI-powered investment assistant**, designed to give users personalized insights, help balance portfolios, and minimize risk, ultimately aiming to save them money on traditional brokerage fees.
- I successfully developed a **robust data pipeline**, including `StockDataCollector` and `StockDataCleaner` modules, for efficiently gathering and thoroughly cleaning historical stock data.
- My **Logistic Regression model for sentiment analysis** proved effective, achieving 78.8% accuracy. It showed a strong ability to identify positive sentiment (94% recall), and I gained insights into the specific words driving sentiment.
- For **stock prediction**, my Linear Regression model, enhanced with technical indicators, significantly **outperformed a naive baseline**. I observed a much lower Mean Absolute Error (0.419 vs. 1.192) and Mean Squared Error (0.395 vs. 3.649), demonstrating superior accuracy.
- A crucial result was learning from the **initial failure of my LSTM model** for stock prediction. This showed me that predicting raw next-candle close prices can create an "illusion of accuracy" and that LSTMs struggled to capture true market volatility. This insight guided my shift to the more effective Linear Regression approach.
- Overall, the project has established a solid **foundation for a dashboard**, capable of providing both technical analysis and sentiment-based stock recommendations.

Future Work:

- Including Real time stock sentiments and stock price data for enhancing the dashboard
- Instead of using a machine learning model like Logistic Regression for Stock Sentiments and Linear Regression for Stock prediction,
- I will use LLM models like Ollama and Neural Network for recommending and predicting stock prices respectively.
- This will create the model more robust, integrating with natural language understanding of sentiments and more advanced technical analysis features for neural network.