DeepSentiment: AI-Powered Stock Prediction with Historical and Social Insights.

Github - https://github.com/yasir-17/DeepSentiment
Members: Yasir & Asad

Project Overview

1. Objective - The primary goal of this AI project is to predict stock prices by integrating traditional historical market data with real-time financial news and social media sentiment. The project aims to address the challenge of accurately forecasting short-term stock price movements, which are influenced not only by historical trends but also by market sentiment driven by news and social media activity.
    a. a.Expected Outcomes - A system capable of predicting stock prices with improved accuracy by combining historical data and sentiment analysis. Insights into how social sentiment impacts stock prices, providing investors with better decision-making tools.
2. Scope –
    a. Analyze historical stock price data to identify patterns and trends
    b. Generate predictions for future stock prices.
    c. It will cover a selected set of companies or stock indices, with an emphasis on stocks with high market activity and social media presence.
    d. Data Sources - Historical stock price data. Social media posts (e.g.,Twitter, Reddit) related to stocks and financial markets
    e. Limitations - The system will not make trading decisions autonomously. The system will not provide financial advice or investment advice. Predictions are based on available data and may not account for unforeseen events. The system will be limited to a specific set of stocks.
3. AI Technique and Tools –
    a. Time Series Forecasting - To model historical stock price data, methods like Long Short-Term Memory (LSTM) or Gated Recurrent Units (GRU) will be used.
    b. Natural Language Processing - For sentiment analysis, techniques like BERT (Bidirectional Encoder Representations from Transformers) or RoBERTa will be applied to financial news and social media data.
    c. Tools and Frameworks:
        i. TensorFlow or PyTorch for building model
        ii. Hugging Face Transformers for sentiment analysis.
        iii. AWS Sagemaker (Or some other cloud service) for large-scale training, tuning, and model management. Model training may be done with hipergator instead.
        iv. Docker for containerizing the models, ensuring compatibility across environments.

v. Git for version control.

Stakeholders

1. Project Team
    a. Asad – Data Engineer / Data Scientist - Responsible for collecting, cleaning, and managing both historical stock price and social media data. Develop models for stock price prediction.
    b. Yasir – Project Manager / ML Engineer - Oversees the overall progress, sets timelines/ Manager resources. Optimize and Fine tune model and deployment and monitoring.
2. The End user of this models includes
    a. Individual Investors – They would use this system to inform their investment decisions. They would interact through a web or mobile app interface.
    b. Financial Advisors - They would use the system to provide informed advice to their clients. They would interact through a web or mobile app interface. Possibly can include more detailed dashboards with additional analytical tools.
    c. Traders - They would use the system for long/short term trading decisions. They would interact using APIs or web interfaces with real time alerts.
3. Other Stakeholders
    a. Data Providers - Providers of historical stock price data (e.g., Yahoo Finance, Alpha Vantage) and sentiment data from social media (e.g., Twitter API).
    b. Regulatory Bodies (e.g., SEC, FINRA) - Oversee compliance with financial regulations.
    c. Legal teams - Ensures the project complies with relevant laws and regulations. Handles any legal issues related to data usage or system performance
    d. Security team - Ensures the system is secure and protects sensitive financial data
    e. Marketing team - Promotes the AI system to potential users. Gather feedback for system improvement.
    f. External Partners - Collaborators such as third-party AI service providers or cloud platform vendors offering infrastructure or additional tools

Computer Infrastructure Considerations

1. Project Needs Assessment
    a. Primary Objective - The AI system’s main goal is to predict stock prices by combining traditional market data with social media sentiment. The tasks involved include time series forecasting and natural language processing (NLP) for sentiment analysis.

    b. Data Types - Structured historical stock data and unstructured text data from social media.
    c. Performance benchmark - Key metrics includes accuracy in stock price prediction (most important for this project) and latency (for real time prediction)
    d. Deployment Constraints - The project will be deployed in the cloud but may need to integrate with on premise data source.
    e. Resources - https://paperswithcode.com/sota/stock-market-prediction-on-stocknet
2. Hardware Requirements planning
    a. The model will be trained on hipergator with the following specifications - AMD EPYC 75F3 32-Core Processor and NVIDIA A100 GPU. This high performance gpu is needed for training large models like BERT for sentiment analysis. CPU power is needed for preprocessing tasks such as data cleaning.
    b. RAM capacity of 8gb would be sufficient for handling and training data.
3. Software Environment Planning
    a. A Linux based system will be utilized for this project.
    b. Software Stack -
        i. Pandas and NumPy for data manipulation
        ii. HuggingFace Transformers for NLP and sentiment analysis
        iii. TensorFlow/PyTorch for building LSTM for forecasting.
        iv. Scikit-learn for model evaluation and accuracy calculation.
        v. Containerization with Docker for reproducibility and portability
4. Cloud Resources Planning
    a. Use Azure ML or AWS Sagemaker for streamlined model tuning and deployment.
    b. AWS S3 or Azure Blob storage can be used to store both the historical stock data and social media sentiment data.
    c. AWS sagemaker or Azure ML is suitable for deploying, and monitoring the models.
    d. Use Docker for orchestrating containerized model training and inference workloads
5. Scalability and Performance Planning
    a. Use distributed training techniques like data parallelism and model parallelism to train on multi-GPU clusters.
    b. Leverage cluster auto-scaling to dynamically adjust resources based on workload.
    c. For performance optimization use techniques like quantization to reduce model size (specially for large language models for sentiment analysis) and improve inference speed without compromising much prediction accuracy.
    d. Implement real-time monitoring tools like AWS CloudWatch or NVIDIA Nsight Systems to ensure optimal GPU and CPU performance during training and deployment.

Security, Privacy and Ethics Considerations -

1. Problem Definition -
    a. Problem Definition - Perform an Ethical Impact Assessment to identify potential risks related to biased predictions, particularly in market sentiment analysis. For example, certain social media sentiment could disproportionately impact the stock prices of companies in specific industries or regions.
    b. Engage diverse stakeholders including regulators, investors, and consumer advocates to gather feedback on ethical concerns.
2. Data Collection -
    a. Ensure collected stock price and sentiment data is free from selection bias and represents a broad range of companies, industries, and market conditions.
    b. Implement data anonymization techniques to protect privacy of individuals mentioned in social media posts.
3. AI Model Development -
    a. Test models for fairness and bias across different types of stocks, industries, and market conditions. Use techniques like adversarial debiasing to mitigate bias.
    b. Technical Implementation - Use the Fairlearn library to assess model fairness and AIF360 to mitigate biases. Use SHAP and LIME for model interpretability.
4. AI Deployment -
    a. Implement a human-in-the-loop deployment where model predictions are reviewed by experts before being acted upon.
    b. Use secure model serving frameworks like TensorFlow Serving to prevent tampering and protect model integrity.
    c. Provide clear explanations of model predictions and confidence scores to end users.
    d. Technical Tool - Leverage BentoML to ensure the model is deployed with secure APIs and has built-in monitoring and feedback systems. This will help safeguard sensitive financial information and prevent unauthorized access to the deployed model.
5. Monitoring and Maintenance -
    a. Continuously monitor live model performance and compare to expected benchmarks. Set up alerts for deviations and anomalies.
    b. Regularly evaluate model fairness and bias on new real-world data.
    c. Technical Tool - Use tools like Amazon SageMaker Model Monitor to detect data drift and concept drift.

Human Computer Interactions -

1. Understanding Users Requirements -

a. Conduct user interviews and surveys to gather insights from potential users of the stock prediction system, including individual investors and financial advisors. Use tools like Google Forms to conduct structured surveys.
b. Employ semi-structured interviews to explore user needs, goals, and pain points in depth.
c. Some questions that can be used in the survey or interviews are -
    i. What is your current level of experience with investing in the stock market?
    ii. How often do you make investment decisions or trades?
    iii. Do you use automated systems to make decisions?
    iv. What are your primary goals when using a stock prediction or investment research system?

2. Creating personas and Scenarios
    a. To ensure the AI system meets the needs of a diverse user base, we will focus on understanding user goals, behaviors, and challenges through the development of personas and corresponding usage scenarios. These personas will help us visualize how different users interact with the system and ensure that the design aligns with their specific requirements.
    b. We will develop two personas at the extreme end of the spectrum.
        i. Novice Investor “Alex” - Alex is a young investor who has recently started exploring stock markets. He uses basic stock trading platforms but lacks deep financial knowledge. The primary goal of Alex is to find stocks that are going to perform well. The challenge with Alex is that he has limited financial literacy. The scenario of Alex is that he logs into the system and input the stocks in which he is interested in and see the performance prediction of those stocks to make informed decisions.
        ii. Financial Advisor “Priya” - Priya is a professional financial advisor with 10+ years of experience. She works with high-net-worth clients and requires in-depth, reliable market analysis. The primary goal of priya is to obtain deeper insights into historical stock trading patterns and to understand correlations between stock data and social media sentiment trends to inform her advice to clients. The challenges are that she needs customizable data views and advanced analytics to create detailed reports. The scenario of priya is that she logs in to access historical data on a set of stocks her clients have invested in. She generates an AI-driven report combining past trading data, sentiment analysis from news and social media, and potential future performance predictions. Priya customizes the visualization of data, exporting reports to present to her clients.

3. Conducting Task Analysis -
    a. Hierarchical Task Analysis - Break down complex tasks into smaller steps. Use Figma or Lucidchart to map out workflows, like how users move from inputting a stock name to receiving predictions and insights. For financial advisors like Priya

streamlined these processes and included detailed analytics to get better insights. Use tools like Figma to create a task flow.

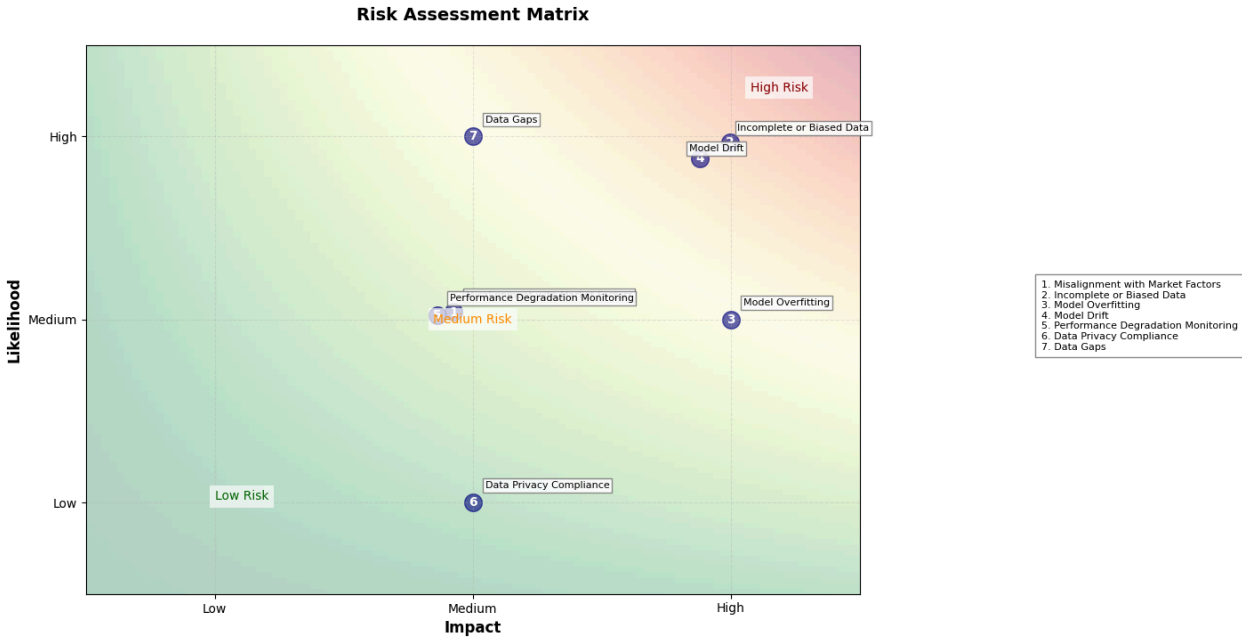
4. Identify Accessibility Requirements -
    a. Ensure the stock prediction system adheres to WCAG 2.1 AA accessibility guidelines. Use tools like WAVE to audit the system for accessibility issues.
    b. Implement features to support people with disabilities such as
        i. Proper color contrast and font sizes for people with visual impairments.
        ii. Compatible with screen readers.
    c. Conduct usability testing with users who have disabilities to validate the accessibility of the system
5. Outline Usability Goal -
    a. Ensure users can complete tasks like generating reports or executing trades in the shortest time possible by optimizing workflows and reducing unnecessary steps.
    b. Maintaining a System Usability Scale (SUS) score above 80 across all user segments.
    c. Example - A key usability goal could be reducing the time it takes for users to generate a stock report from 10 minutes to under 5 minutes by streamlining stock selection and automating parts of the process.

Risk Management Strategy -

1. Problem Definition
    a. Risk: Misalignment between the problem definition and available data (e.g., market movement may depend on additional macroeconomic factors not captured by tweets or stock prices).
    b. Strategy:
        i. Stakeholder collaboration: Ensure alignment between the business and technical teams to define clear goals.
        ii. Residual Risk Assessment: Use a residual risk matrix to continuously assess and re-evaluate any remaining risks after mitigation efforts.
2. Data Collection
    a. Risk: Incomplete, biased, or outdated data (e.g., missing stock data or biased tweets).
    b. Strategy:
        i. Use data validation libraries: Implement Pandas or Great Expectations to validate collected data (check for missing values, duplicates, or inconsistencies).
        ii. Residual Risk: Continuously assess data gaps, and plan for retraining the model as needed.
3. Model Development

    a. Risk: Overfitting the model to historical data that may not generalize well to future events (since markets are highly stochastic).
    b. Strategy:
        i. Model regularization: Use techniques like dropout and L2 regularization to reduce overfitting.
        ii. Cross-validation: Implement time-series cross-validation (e.g., walk-forward validation) to ensure better generalization.
        iii. Technical Implementation: Leverage sklearn.model_selection.TimeSeriesSplit for robust cross-validation.
4. Deployment
    a. Risk: Model drift due to changing market conditions or events not reflected in the original dataset.
    b. Strategy:
        i. Monitoring and Retraining Plan: Monitor the model's performance in production and define triggers for retraining based on drift detection metrics.
        ii. Technical Implementation: Use libraries like Evidently AI or Alibi Detect for drift detection.
5. Monitoring and Maintenance
    a. Risk: Lack of continuous monitoring leads to undetected performance degradation.
    b. Strategy:
        i. Performance Monitoring Dashboard: Set up automated alerts for metrics like accuracy, precision, or recall.
        ii. Residual Risk: Define an SLA (Service-Level Agreement) for the maintenance team to ensure timely responses to issues.
        iii. Use Prometheus or Grafana to implement monitoring dashboards.
6. Graph for likelihood vs Impact for the risk identified

a.



Data Collection Management and Report

1. Data Type:
    a. Tweets: Unstructured text data that may contain personal identifiers (like emails or phone numbers).
    b. Stock prices: Structured numerical data in CSV format.
2. Data Collection Methods:
    a. Tweets: Collected using the Twitter API. Ensure adherence to API rate limits and usage policies.
    b. Stock prices: Retrieved from Yahoo Finance.
3. Compliance with Legal Frameworks:
    a. Twitter Terms of Service: Raw tweets cannot be redistributed; consider releasing only metadata and sentiment scores instead of full text.
    b. GDPR/CCPA Compliance:
        i. Any personal identifiers such as email addresses, phone numbers, or user IDs should be masked or removed to protect privacy.
        ii. Use regular expressions (regex) to detect sensitive information in tweets.
        iii. Example: Replace detected emails/phones with "[MASKED]".
4. Data Ownership:
    a. Twitter: Owns the original tweet data.
    b. Yahoo Finance: Provides public financial data, but redistribution may require additional permissions.
5. Metadata:

    a. Tweets: Metadata includes tweet ID, user ID, timestamp, and preprocessed sentiment (if calculated).
    b. Price Data: Includes stock ticker, date, and adjusted prices.
6. Versioning:
    a. Use Data Version Control (DVC) to track updates to both tweet and stock data. Maintain changelogs for transparency.
7. Data Preprocessing, Augmentation, and Synthesis:
    a. Preprocessing:
        i. Remove personal identifiers using regex masking (\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,7}\b for emails, or \b\d{10}\b for phone numbers).
        ii. Remove non-English tweets and special characters to reduce noise.
    b. Augmentation/Synthesis:
        i. Convert tweets into sentiment scores using VADER or TextBlob for feature extraction.
        ii. Normalize stock prices to account for variations across companies.
8. Report on Risk Management in Data Collection:
    a. Risk:
        i. Privacy Breach: Exposed phone numbers or emails in tweets could lead to compliance violations.
    b. Mitigation Strategy:
        i. Use regex-based masking scripts to automatically detect and mask sensitive information before storing or processing tweets.
        ii. Implement access controls to restrict access to raw tweet data only to authorized personnel.
9. Report on Trustworthiness in Data Collection:
    a. Transparency: Maintain documentation of all data cleaning and masking operations.
    b. Bias Detection: Perform sentiment analysis on tweets to detect any sentiment bias that might skew the model's predictions.
    c. Auditability: Keep logs of when and how masking operations are performed for compliance audits.

Model Development and Evaluation

1. Model Development-
    a. Algorithm Selection – We have used the FinTweetBERT-sentiment model from huggingface which is a fine tuned BERT model for classifying the sentiment of financial tweets. We have then used the LSTM model as our final model for predicting stock price which is an appropriate model for time series data.
    b. Feature Engineering and Selection – Our target variable is Adjusted Close pricing feature which is a closing price of stock after accounting for corporate actions like

stock splits, dividends, and distributions. It provides a more accurate representation of the stock's true value over time. We have created some more features to accurately help predict the prices. They are as follows

i. Price_change – Calculated as (Today's Price - Yesterday's Price) / Yesterday's Price, it helps measure daily price momentum and volatility patterns.
ii. Volumes – This was one of the original features from the dataset and it shows the number of shares traded during the trading period. We have decided to include this as it can indicate trading/activity or market participation of that stock.
iii. Volume_change – It shows the percentage change in the trading volume. It can help identify unusual trading activity and hence we decided to keep this.
iv. Moving averages – These can indicate trends in the market. We have decided to keep 5,10, 20 and 50 days moving averages to help identify short, medium and long term trends in the market.
v. Sentiment – then the final feature is sentiment which was calculated using FinTweetBERT-sentiment model.

c. Feature Importance – Feature importance is added in the LSTM.ipynb file. We calculated the feature importance using two different methods. Even though sentiments have very little importance as concluded by the feature importance they still have some importance nonetheless. Also its important to note that sentiments have different importance in different stocks. For instance, sentiments have more importance for the Google stock as compared to Apple stock and as such incorporating them improved the forecasting accuracy of Google stocks more
    i. Permutation Importance: Measures how much the model's performance decreases when a feature is randomly shuffled.
    ii. Integrated gradients: Uses gradients to attribute the prediction to each input feature.
d. Model Complexity and Architecture – For detailed model architecture please see the following jupyter notebook (Jupyter). Following is the brief summary
    i. 3 LSTM layer with decreasing size.
    ii. Each LSTM layer followed by batch normalization and dropout
    iii. Added dense layer and output layer.
    iv. Total trainable parameter is 328,289

2. Model Training –
    a. Training Process
        i. Data Preparation
            1. Sequence Creation - Historical data is transformed into sequences of length 20 days. After each sequence the next element is the target variable.
            2. Scaling - Features are normalized using MinMaxScaler to range [0,1]

3. Train-Test Split - Data is split chronologically (80-20 split) to maintain temporal order
4. Batch Processing - Data is processed in batches of 32 samples

ii. Training Implementation
1. Forward pass –
   a. Process batch of sequences
   b. Generate Predictions
   c. Calculate loss using Huber Loss
2. Backward pass –
   a. Compute gradients
   b. Update Model parameters
   c. Reset gradients for next iteration
3. Validation Phase
   a. Evaluate model on validation set
   b. Track validation metrics
   c. Saved best model checkpoint

iii. Training Optimization –
1. Loss Function – We used Huber Loss as this loss function is less sensitive to outliers and provides more stable gradients.
2. Learning Rate Management – Initial rate is 0.001 (Adam optimizer default)
3. We used dynamic learning rate adjustments - ReduceLROnPlateau scheduler. Reduces learning rate when validation loss stops decreasing significantly. It can also helps overcome local minima.
4. Regularization Technique
   a. Dropout layers
   b. Batch Normalization
   c. Early stopping

b. Hyperparameter Tuning

i. Key hyperparameter
1. sequence_length = 20
2. hidden_size = 128
3. num_layers = 3
4. dropout_rates = [0.3, 0.3, 0.2]

ii. Training parameters
1. batch_size = 32
2. initial_lr = 0.001
3. epochs = 100
4. early_stopping_patience = 10
5. lr_patience = 5

iii. Dropout rate - Progressive reduction (0.3 → 0.2)

iv. Sequence length – 20

3. Model Evaluation

a. Performance metrics – We used mean square error and root mean squared error as our primary metrics. Additional metrics used are R2 Score which can explain variance in predictions and mean absolute percentage error.
b. Cross validation – We have used walk forward validation which unlike traditional k fold validation respects the temporal order of time series data. It simulates real-world trading conditions where we can only train on past data to predict future values.
    i. Data Splitting Process –
        1. First split - Training: First 20% of data. Validation: Next chunk of data
        2. Second split - Training: First 40% of data. Validation: Next chunk of data
        3. And so on….
    ii. The implementation is present here (Github)
    iii. Result – Although expected, we didn't see significant improvement of accuracy by using these validations compared to traditional validation. One of the reasons might be that the market conditions might be relatively consistent in our data period, reducing the advantage of temporal validation.

Trustworthiness and Risk management in model development

1. Risk management report – As identified in the previous project the risk of our project in the model development was overfitting the model to historical data that may not generalize well to future events. We implemented several strategies to address this risk:
    a. Walk-Forward Cross-Validation: Implemented TimeSeriesSplit for temporal validation instead of traditional random splitting. This approach may sometime better simulates real-world trading conditions by maintaining temporal order, however, in this example it does not improve the result much. However, its important to note that this approach might still outperform when implemented with more and more stocks and on more dataset.
    b. Model Architecture: Implemented multiple dropout layers to prevent overfitting.
    c. Feature Engineering: Selected a focused set of relevant features including:
        i. Technical indicators like moving averages
        ii. Price and volume changes
        iii. Sentiment analysis.
        iv. Applied MinMaxScaler for feature normalization
    d. The risk of overfitting to specific market conditions is minimized as a result of methods used as mentioned above
2. Trustworthiness Report –

a. In the collected data we have anonymised the username, phone and email to maintain privacy of the user.
    i. Implementation detail - For anonymising the users id/name a sha256 hashing function has been used. For phone number and email we used regex to identify and then replaced those instances with MASK value.
b. In the model development stage, the following strategies have been implemented.
    i. Model development documentation - Maintained comprehensive documentation of model architecture, hyperparameters, and training decisions. Tracked all experiments and validation results in version control. Documented rationale for feature selection and engineering choices.
    ii. Fairness and bias mitigation - Ensured model inputs are based on publicly available financial data. Tested model behavior during both bullish and bearish market periods.
    iii. Transparency - Implemented clear logging of model predictions. Used walk-forward validation to demonstrate real-world performance expectations.

Apply HCI Principles in AI Model Development

1. Develop Interactive Prototype: We have created a gradio based interactive prototype which lets user to input current day features and its shows the next data prediction of price. However, the current version is not very effective, asking users to insert current day features. To overcome that in the next version we will create app which will already have current day features and ask user upto how many days in the future to predict the price. To forecast day n+1, use the predicted data from days 1 to n as a input in a cascading model approach. To keep the accuracy of the model good we will only let user input upto 7 days for prediction. We will add intuitive slider for selecting prediction timeframe (1-7 days)
2. Moreover, the current version also shows the graph of the prices of stocks. In the next version, we will do persona-based customization which lets the interface simple, with only the predicted price showing for the individual investors. For financial advisors it will show advanced visualizations, including line charts along with prices predictions.
3. Design transparent interfaces – We used Docker to ensure consistent user experiences by standardizing the app environment. Additionally we will use Prometheus and Grafana which can support real-time monitoring, alerting, and data transparency.
    a. Containerized Environment - Deployed the Gradio app, model, and auxiliary services within Docker containers, ensuring compatibility across different environments. Docker also aids in scaling and isolation, which can benefit your interactive prediction workflows.
4. Feedback mechanisms - Real-time feedback and system monitoring will enhance both user experience and model performance visibility:

    a. Performance feedback: Display real-time feedback on prediction accuracy and provide users with options to rate prediction relevance. This data can help adjust prediction models to better meet user expectations
    b. User feedback integration: Allow users to flag or rate model predictions and insights, feeding these interactions into a feedback loop for continuous model and interface improvements
5. Performance monitoring with Prometheus: Track metrics such as model response time, prediction accuracy over time, and resource usage in real time. These metrics will alert the user to latency or performance bottlenecks, providing better user experience.
6. User feedback Dashboard with Grafana: Set up a Grafana dashboard to visualize user engagement (e.g., number of predictions requested per session, selected prediction ranges).

Performance Metrics

1. Deployment Environment selection -
    a. Local Deployment - The justification is that this project is in the prototyping phase, with limited resource availability and a focus on rapid development and testing. Local deployment using Docker provides simplicity and also avoids the costs and complexities associated with cloud.
2. Deployment Strategy -
    a. Docker based Containerization
        i. Implementation detail - The Gradio app is containerized using Docker, ensuring a consistent runtime environment across development and testing. Docker simplifies dependency management, making it easier to replicate the setup on other systems if needed.
3. Security and Compliance
    a. During data collection and model creation, privacy concerns were addressed by employing techniques like data masking. For deployment, additional measures include:
        i. Minimal base image - Using a lightweight base image minimizes potential vulnerabilities in the deployment environment.
        ii. Local Network Restrictions - The app is restricted to the localhost environment to prevent unauthorized external access during this phase.
        iii. Secure Environment Variables - Sensitive data, such as Github and API keys, is stored securely using environment variables within the Docker container.
4. CI/CD Strategy
    a. Github Actions - Automated testing and container builds will be integrated into the GitHub repository using GitHub Actions. CI/CD will also manage version control and deployment to the local environment when updates are pushed to the repository.

Evaluation, Monitoring and Maintenance

1. System Evaluation and Monitoring -
    a. Prometheus - I have utilized prometheus for metrics collection and tracking system performance.
    b. Grafana - This can be utilized for visualizing metrics and generating actionable insights.
    c. Some metrics tracked - Request Latency is monitored using gradio_request_latency_seconds. Total request is monitored using gradio_request_total to track usage. Error rates are also monitored. Some other metrics tracked are current active requests, model load time and gradio memory usage. It is important to note that for the prototype, I am using validation data for prediction. This data has not been used for training, ensuring that it can provide an unbiased assessment of the predicted values. The main reason for using validation data is that we know the ground truth for this dataset. However, in a real-life scenario, this data would be replaced by new, current data, and the prediction error could be calculated the next day when the ground truth becomes available.
2. Feedback Collection and Continuous Improvement -
    a. No dedicated feedback collection mechanisms have been implemented yet.
    b. Proposed Plan
        i. Add built-in feedback forms in the Gradio app to capture user suggestions and bug reports
        ii. Implement lightweight tools like Google Analytics or Hotjar to understand user interaction patterns.
3. Maintenance and Compliance Audits -
    a. No maintenance or compliance auditing processes are in place.
    b. Proposed Plan -
        i. Scheduled Maintenance:
            1. Manual updates - Retraining the LSTM model quarterly with new historical stock price data and updated social media sentiment data. Updating dependencies and verifying Docker container integrity.
            2. Automated workflows - Monitoring resource usage and model drift via Prometheus with alerts configured in Grafana.
        ii. Compliance Audits:
            1. Manual checks - Quarterly audits of data anonymization processes to ensure compliance with GDPR/CCPA regulations.
            2. Reviewing model predictions to confirm no biases are introduced that could lead to market manipulation.