

Stock Price Prediction Using Knowledge Graphs and LLM Reasoning

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology in Computer Science and Engineering

by

Unmukt Singh (20BCE0875)

Pilli Sai Nishanth (20BCE0906)

Vishesh Bhargava (20BCE2506)

Under the guidance of

Dr. Durgesh Kumar

SCOPE

VIT, Vellore.



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

May, 2024

DECLARATION

I hereby declare that the thesis entitled “Stock Price Prediction Using Knowledge Graphs and LLM Reasoning” submitted by me, for the award of the degree of *Bachelor of Technology in Programme* to VIT is a record of bonafide work carried out by me under the supervision of Dr. Durgesh Kumar.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place : Vellore

Date : 9th May, 2024

Signature of the Candidate

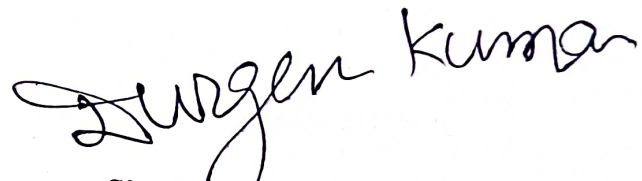
CERTIFICATE

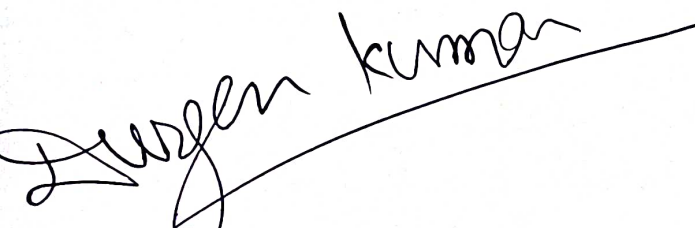
This is to certify that the thesis entitled "Stock Price Prediction Using Knowledge Graphs and LLM Reasoning" submitted by Unmukt Singh (20BCE0875), Pilli Sai Nishanth (20BCE0906) & Vishesh Bhargava (20BCE2506), SCOPE, VIT, for the award of the degree of Bachelor of Technology in Programme, is a record of bonafide work carried out by him / her under my supervision during the period, 10. 01. 2024 to 08.05.2024, as per the VIT code of academic and research ethics.

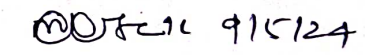
The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfills the requirements and regulations of the University and in my opinion meets the necessary standards for submission.

Place : Vellore

Date : 9th May, 2024


Signature of the Guide


Internal Examiner


External Examiner

Head of the Department
Programme

ACKNOWLEDGEMENTS

I would like to express my thanks to my mentor Dr. Durgesh Kumar for his time and efforts he provided throughout the semester for his contributions to the completion of my project titled Stock Price Prediction using Knowledge Graphs and LLM Reasoning. His useful advice and suggestions were really helpful to me during the project's completion. In this aspect, I am eternally grateful to him.

I would like to express my profound gratitude and special thanks to Mrs Umadevi K.S , HOD of SCOPE department, and Mr. M Anthony Xavier , Dean of Vellore Institute of Technology for their support in completing the project.

I would like to acknowledge that this project was completed entirely by me and my team and not by someone else.

Student Name

EXECUTIVE SUMMARY

Predicting stock prices accurately is a paramount objective for investors, financial analysts, and traders alike, as it enables informed decision-making, risk mitigation, and potential profit maximization. Traditional methods often rely solely on historical data and technical indicators, overlooking the valuable insights embedded in real-time news and social media sentiments. In response, this project introduces an innovative approach leveraging knowledge graphs and Language Model (LLM) reasoning to enhance stock price prediction accuracy. The proposed system employs automated bots equipped with news APIs and social media APIs to gather real-time textual data. These data sources are then subjected to sentiment analysis and LLM Reasoning to gauge market sentiment accurately. Furthermore, the integration of knowledge graphs provides contextual understanding, allowing the LLM to determine the relevance and impact of various events on stock prices. 8 selected tech stocks were used to evaluate the three strategies discussed. LLM KG Reasoning demonstrates the highest performance for 5 out of the 8 stocks however, this strategy also yields negative results for 2 stocks. In conclusion, this project hypothesizes a novel framework for stock price prediction that integrates knowledge graphs, sentiment analysis, and LLM reasoning, offering a more comprehensive approach to forecasting market trends and making informed investment decisions.

TABLE OF CONTENTS

Acknowledgements	4
Executive Summary	5
Table Of Contents	6
List of Figures	8
List of Tables	9
List of Abbreviations	10
1. INTRODUCTION	11
1.1 Background	11
1.2 Objectives of the Project	11
1.3 Motivation	12
2. PROJECT DESCRIPTION AND GOALS	13
2.1 Survey on Existing System	14
2.2 Gaps Identified	19
2.3 Problem Statement	20
3. TECHNICAL SPECIFICATION	20
3.1 REQUIREMENTS	20
3.1.1 Functional	20
3.1.2 Non Functional	21
3.2 Feasibility Study	22
3.2.1 Technical Feasibility	22
3.2.2 Economic Feasibility	22
3.2.3 Social Feasibility	22
3.3 System Specifications	23
3.3.1 Hardware Specifications	23
3.3.2 Software Specifications	24
4. DESIGN APPROACH & DETAILS	25
4.1 Design Approach	25
Module 1 - Backtesting	25
Module 2 - News Extraction Model	25
Module 3 - Sentiment Analysis	26
Module 4 - Knowledge Triplet Extraction	26
Module 5 - LLM reasoning with prompt engineering	26
Module 6 - Embedding Generation and Semantic Search	26
4.2 Codes and Standards	29
4.3 Constraints and Alternatives	30

5. SCHEDULE, TASKS AND MILESTONES	31
5.1 Schedule	31
5.2 Tasks	32
5.3 Milestones	35
6. PROJECT DEMONSTRATION	35
7. COST ANALYSIS / RESULTS & DISCUSSIONS	39
7.1 Cost Analysis	39
7.2 Results and Discussion	39
8. SUMMARY	45
9. REFERENCES: (IEEE format)	46
APPENDIX A - SAMPLE CODE	47

List of Figures

Figure no.	Title	Page no.
4.1	LLM KG Reasoning architecture	25
4.2	Data flow diagram	27
4.3	Class diagram	27
4.4	Sequence diagram	28
4.5	Use-case diagram	29
5.1	Gantt chart	35
6.1	Sentiment Analysis Output	35
6.2	LLM Reasoning Output	36
6.3	Triplet Generation and Extraction Output	36
6.4	Knowledge Graph Visualization	37
6.5	Strategy Backtesting Output	38
7.1	Sentiment Analysis on Tesla news - test 1	39
7.2	Sentiment Analysis on Tesla news - test 2	40
7.3	Sentiment Analysis on Tesla news - test 3	40
7.4	LLM Reasoning on Tesla news - test 1	40
7.5	LLM Reasoning on Tesla news - test 2	41
7.6	LLM Reasoning on Tesla news - test 3	41
7.7	NVDA Performance LLM (left) vs LLM_KG (right)	43
7.8	TSLA Performance LLM (left) vs LLM_KG (right)	43
7.9	AAPL Performance LLM (left) vs LLM_KG (right)	43
7.10	AMD Performance LLM (left) vs LLM_KG (right)	44
7.11	AMZN Performance FinBERT SA(left) vs LLM_KG (right)	44
7.12	GOOG Performance FinBERT SA(left) vs LLM_KG (right)	44

List of Tables

Table no.	Title	Page No.
2.1	Literature Survey	13
7.1	Performance Results	42

List of Abbreviations

KG	Knowledge Graph
LLM	Large Language Model
BERT	Bi-directional Encoder Representations from Transformers
FinBERT	Financial Bi-directional Encoder Representations from Transformers
GPT	Generative Pre-trained Transformer

1. INTRODUCTION

1.1 Background

In the world of stock market trading, the quest for accurate predictions and informed decision-making has long been the holy grail for investors and financial institutions alike. Historically, investment firms have relied heavily on traditional methods grounded in hard data analysis, such as technical indicators and historical price patterns, to guide their trading strategies. While these methods have proven valuable to a certain extent, they often overlook the wealth of information embedded in real-time news, social media sentiments, and the broader context surrounding market events.

As the digital age progresses, the relevance of Natural Language Processing (NLP) techniques in financial markets has steadily gained momentum. NLP enables the extraction of valuable insights from vast amounts of unstructured textual data, empowering traders to augment their decision-making processes with a deeper understanding of market sentiment and dynamics. From sentiment analysis of news articles to tracking social media chatter, NLP has become an invaluable tool in the arsenal of modern traders seeking to gain a competitive edge.

Moreover, the evolution of Language Model technologies, particularly Large Language Models, represents a paradigm shift in how we approach textual data analysis. LLMs, such as OpenAI's GPT series, possess the ability to understand and generate human-like text at an unprecedented scale. This capability opens up new avenues for reasoning over textual data, enabling more nuanced analysis and prediction of market trends.

In this context, the integration of NLP techniques and LLM reasoning into stock market trading strategies presents a compelling opportunity to enhance prediction accuracy and unlock hidden insights. By harnessing the power of real-time news feeds, social media sentiment analysis, and advanced language understanding, traders can gain a more comprehensive view of market dynamics and make more informed investment decisions.

The project "Stock Price Prediction using Knowledge Graphs and LLM Reasoning" suggests that combining recent news summaries with context from dynamically updated KGs can offer insights into market sentiment and fundamental factors impacting stock prices. By employing sentiment analysis techniques and reasoning capabilities of LLMs, the project aims to evaluate investor sentiment and market sentiment towards specific stocks or sectors, thereby improving stock price prediction accuracy.

1.2 Objectives of the Project

This project's primary objective is to develop a reliable framework for predicting stock price movements by utilizing Knowledge Graphs and Large Language Model reasoning, alongside sentiment analysis of news and digital forums. The project seeks to investigate the potential of recent news summaries, integrated with regularly updated KGs, in enhancing comprehension of

market sentiment and fundamental factors affecting stock prices. Through the application of sentiment analysis techniques, the project aims to quantify investor and market sentiment towards specific stocks or sectors, thereby refining predictive models.

This involves the development of various components such as a news extraction unit, backtesting unit, sentiment analysis unit, LLM-reasoning unit, KG triplet generation and extraction unit.

Additionally, the project endeavors to leverage the reasoning capabilities of LLMs, honed through prompt engineering, to extract actionable insights from financial data and facilitate informed decision-making and measure the performance.

Ultimately, the project intends to demonstrate the effectiveness of this integrated approach in improving the accuracy of stock price predictions and providing valuable insights for investors and financial analysts navigating the intricacies of financial markets.

1.3 Motivation

The motivation behind constructing a stock price prediction bot stems from the desire to enhance and automate decision-making processes in financial markets and creating a stock market prediction bot using knowledge graphs and LLM reasoning offers several compelling motivations. Firstly, knowledge graphs provide a structured representation of complex relationships between entities, allowing for more comprehensive analysis of stock market data. By leveraging knowledge graphs, the prediction bot can capture subtle interdependencies between various factors influencing stock prices, leading to more accurate predictions. Additionally, knowledge graphs enable the integration of diverse data sources, including financial news, social media sentiment, and historical market data, providing a holistic view of market dynamics. This comprehensive approach enhances the prediction bot's ability to identify trends and patterns, thereby empowering investors with valuable insights for informed decision-making. Furthermore, building a prediction bot using LLM reasoning fosters innovation and experimentation in the field of artificial intelligence and finance, driving advancements in predictive analytics and algorithmic trading strategies. Overall, the creation of a stock market prediction using knowledge graphs presents an exciting opportunity to harness cutting-edge technology and data-driven methodologies to navigate the complexities of financial markets effectively.

2. PROJECT DESCRIPTION AND GOALS

2.1 Survey on Existing System

Table 2.1 Literature Survey

Title	Authors	Proposed work	Limitations
[1] A Stock Price Prediction Model Based on Investor Sentiment and Optimized Deep Learning	Guangyu Mu; Nan Gao; Yuhang Wang; Li Dai	The proposed method uses the Sparrow Search Algorithm (SSA) to optimize the LSTM model. It does so by conducting a sentiment analysis on the east money stock from yahoo finance forum. Finally, the sentiment index and fundamental trading data are integrated, and LSTM is used to forecast stock prices in the future. Experiments demonstrated that the SSA-LSTM model outperforms the others currently used models and has high universal applicability	1) The study conducted is not refined enough to extract features like fear, anguish, disgust and sadness 2) The predicted model can't handle sudden macroeconomic situations and policy shifts which might drastically influence the stock

<p>[2] Impact of News on the trend of stock Price change: An analysis based on the Deep Bidirectional LSTM Model</p>	<p>Y Ren, F Liao, Y Gong</p>	<p>The paper conducts a comparative study of various machine learning algorithms like SVM , LR , LSTM , DBLSTM to predict the accuracy of the model by using a BIAS calculated by the moving average and closing price of the stock. It conducts its sentiment analysis on Sina Finance and Economics, Oriental Wealth Network, Cathay Tai'an Data Terminal news channels by designing a web crawler to get the information..</p>	<ol style="list-style-type: none"> 1) They use a predetermined BIAS which might not be most suitable due to fast changing trends 2) Python's Jieba Chinese word segmentation library to segment news content is not good at translating it to English leaving important information out.
<p>[3] Using Financial News Sentiment for Stock Price Direction Prediction</p>	<p>B Fazlija, P Harder</p>	<p>The paper fine tunes the use of FinBERT, a pre-trained language model to conduct financial data analysis and predict its accuracy compared to other machine learning models . It mainly has four features i.e stock price prediction,sentiment analysis, entity recognition, relationship extraction.</p>	<ol style="list-style-type: none"> 1) Transaction costs were not taken into consideration 2) Company specific sentiment scores were not considered 3) Past time series information such as a moving value at risk or a moving expected shortfall of the previous days were not added as

			input for improving the quality of the predictions
[4] Improved Stock Price Movement Classification n Using News Articles Based on Embeddings and Label Smoothing	Luis Villamil, Ryan Bausback, Shaeke Salman, Ting L. Liu, Conrad Horn, Xiuwen Liu	The study conducted, trained the model in a dual learning phase on the reuters and Bloomberg dataset as well as conducted sentiment analysis from Yahoo finance forums. They used closing prices and sliding trading windows to calculate the predicted prices and used Label smoothing was used to Reduce Overconfidence, Improve Generalization Performance ,Enhance Calibration.	<p>1) The study conducted is inconsistent with the Efficient Market Hypothesis Once patterns are discovered, they provide an advantage only for a limited time, as rational traders adjust their behavior to compensate for this new information, the pattern will be destroyed.</p> <p>2) The classification model cannot classify using the prediction set and inversely, the prediction model cannot predict using the classification set.</p>

<p>[5] Tree of Thoughts: Deliberate Problem Solving with Large Language Models</p>	<p>Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, Karthik Narasimhan</p>	<p>The authors propose Tree of Thoughts prompting approach which generalizes over the popular Chain of Thoughts approach and produces intermediary steps towards solving a problem. It does this by considering many different reasoning paths and self - evaluating choices to decide the next course of action as well as backtracking and looking ahead to make global choices. Implementation of search heuristic via self - evaluation and deliberation instead of programmed (eg. deepblue) or learned (eg. AlphaGo) search heuristics is novel.</p>	<p>1) This approach is very computationally intensive therefore not suited for everyday tasks.</p> <p>2) The authors are not shy of admitting that they only use tasks such as Game of 24 and Crosswords which require backtracking where the ToT approach has an inherent advantage over the general CoT approach.</p>
<p>[6] Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection</p>	<p>Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, Hannaneh Hajishirzi</p>	<p>Large language models often “hallucinate” or produce information that is factually incorrect due to their reliance on information that they have been trained on. To counter this, an approach known as RAG (Retrieval Augmented Generation) is introduced which is essentially a database from which the model fetches up-to-date information. However, retrieving a fixed number of passages regardless of whether retrieval is necessary, or passages are relevant diminishes LM versatility and can cause unhelpful response generation. The authors propose a new framework called SELF-RAG</p>	<p>1) Since the model has to critique its own output using the critic tokens, it becomes a more computationally intensive process similar to the ToT approach.</p> <p>2) During the training phase, a Retriever, Critic and Generator model need to be trained which may be time consuming.</p>

		which essentially critiques its own response to produce the best possible response.	
[7] Large Language Models Are Reasoning Teachers	Namgyu Ho, Laura Schmid, Se-Young Yun	Authors propose Fine-tune CoT which essentially uses larger LLMs to train smaller LLMs. It provides notable reasoning performance in models while preserving the versatility of prompt-based CoT reasoning. This enables models as small as 0.3B to outperform models as big as 175B in certain tasks.	<p>1) Samples show that output from student models may occasionally be repetitive and digressive as fine tuning on a shorter model can cause the student model to produce shorter rationales.</p> <p>2) The performance of this method can be boosted with a wider array of teachers. Eg : - GPT-4 has shown significant advances in complex reasoning abilities which can improve Fine-tune CoT in difficult datasets such as GSM8K</p>

[8] Lumos: Learning Agents with Unified Data, Modular Design, and Open-Source LLMs	Da Yin, Faeze Brahman, Abhilasha Ravichander, Khyathi Chandu, Kai-Wei Chang, Yejin Choi, Bill Yuchen Lin	LUMOS is a novel framework that employs a unified data format and a modular architecture consisting of a planning, grounding and execution model. The planning model turns the problem into high level sub problems, The grounding model turns these high-level sub problems into low level actions and finally the execution model is responsible for the execution of these low-level actions.	<p>1) Training learnings agents with LUMOS framework requires large amount of data and can be time-consuming and expensive.</p> <p>2) Compared to GPT-based agents, agents trained on LUMOS can struggle with generalizing their knowledge to unseen environments</p>
[9] An Exploratory Study of Stock Price Movements from Earnings Calls	Sourav Medya, Mohammad Rasoolnejad, Yang Yang, Brian Uzzi	<p>The authors proposed that analysis of sentiment in Earning call reports and identify a correlation between EC and temporary stock price movement.</p> <p>They generate a vector by combining vectors generated using Doc2Vec and Gate GNN, which is then used to identify sentiment traits.</p> <p>Sentiment identification of traits is performed using Linguistic Inquiry and Word Count. Traits include: positive, negative, anxiety, sadness</p>	<p>1) The paper restricts itself to classical sentiment analysis as well as delves into Earnings per Share as well as Earning call reports with no emphasis on social media sentiment as well as general news sentiment.</p>

[10] NeMo Guardrails: A Toolkit for Controllable and Safe LLM Applications with Programmable Rails	Traian Rebedea, Razvan Dinu, Makesh Sreedhar, Christopher Parisien, Jonathan Cohen	NeMo Guardrails acts like a proxy between the user and the LLM. It allows developers to define programmatic rails that the LLM should follow in the interaction with the users using Colang, a formal modeling language designed to specify flows of events, including conversations. Colang is interpreted by the Guardrails runtime which applies the user-defined rules or automatically generated rules by the LLM, as described next. These rules implement the guardrails and guide the behavior of the LLM.	1) The three-step CoT prompting approach used by the Guardrails runtime incurs extra costs and extra latency. As these calls are sequentially generated, the calls cannot be batched.
--	--	--	--

2.2 Gaps Identified

Several limitations were identified across the research papers we reviewed. The three-step CoT prompting approach incurs extra costs and latency due to its sequential nature, hindering batch processing. Additionally, the papers primarily focus on classical sentiment analysis and financial reports, neglecting social media and general news sentiment analysis, which could provide valuable insights. Furthermore, training learning agents requires significant data and can be expensive and time-consuming. These agents may also struggle to generalize their knowledge to new situations compared to GPT-based agents. Fine-tuning student models can lead to repetitive and digressive outputs as they prioritize shorter rationales. Finally, the performance of the CoT method could be improved by incorporating a wider range of teachers, with advanced reasoning capabilities like GPT-4, especially for complex datasets. More importantly, it was noticed that stock price prediction had been performed using Sentiment Analysis and LLM Reasoning but none of the architectures incorporate an efficient use of Knowledge Graphs to potentially improve performance.

2.3 Problem Statement

The problem addressed by this project revolves around the inherent difficulty in accurately predicting stock price movements within financial markets. Conventional methodologies often struggle to encapsulate the intricate web of relationships and dependencies among various factors that influence stock prices. One significant aspect that remains challenging is the effective analysis of sentiment expressed in news articles and digital forums, hindering the ability to quantify both investor sentiment and broader market sentiment towards specific stocks or sectors. Moreover, existing approaches lack the agility to adapt to rapidly changing market conditions and incorporate up-to-date information from dynamically updated Knowledge Graphs (KGs). The utilization of Large Language Model (LLM) reasoning techniques to extract actionable insights from financial data is still in its nascent stages. Consequently, there exists a pressing need for a comprehensive predictive framework that seamlessly integrates recent news summaries with KGs and leverages LLM reasoning. Such an integrated approach promises to not only enhance the accuracy of stock price predictions but also provide invaluable insights for investors and financial analysts navigating the complexities of today's financial landscape.

3. TECHNICAL SPECIFICATION

3.1 REQUIREMENTS

3.1.1 Functional

A) Data Acquisition

- The System shall have the ability to retrieve financial data (historical stock prices, company filings, economic indicators) from designated sources using finance and stock market news API such as Marketaux.
- The System shall have the capability to collect publicly accessible news articles, reports, and social media information for LLM analysis.

B) Knowledge Graph Construction

- The System shall have the functionality to build a comprehensive knowledge graph encompassing companies, economic indicators, global events, and their interdependencies.
- The System shall have the feature to define relationships between entities within the knowledge graph.

C) Embeddings and Semantic Search

- The System shall have the functionality to create Knowledge graph embeddings upon construction of knowledge graph triplets.
- The triplets relevant to the news shall be extracted and injected as context for LLM Reasoning

D) Prediction Results

- The System shall have the functionality to generate stock price predictions for a designated time frame.
- The System shall have the feature to present the predictions in a user-friendly format.

3.1.2 Non Functional

A) Performance

- The system demonstrates acceptable processing speed for data acquisition, knowledge graph construction, LLM training, and prediction generation

B) Accuracy

- The stock price predictions generated by the system achieves a desired level of accuracy compared to existing methods.

C) Scalability

- The system architecture is designed in such a way to accommodate future expansion in terms of data sources, companies covered, and prediction timeframe.

D) Data Security

- The system implements appropriate security measures to protect sensitive financial data during acquisition, processing, and storage.

3.2 Feasibility Study

3.2.1 Technical Feasibility

The technical feasibility of this project is bolstered by several strengths. Firstly, the existence of well-developed knowledge graphs (KGs) and advancements in large language models (LLMs) with reasoning capabilities provide a solid foundation. These advancements can be harnessed to create a system that captures complex relationships between entities influencing stock prices and analyzes financial text data with deeper understanding. Secondly, the project can benefit from the extensive landscape of open-source libraries and frameworks. These readily available tools can streamline the development process for knowledge graph construction, LLM training, and machine learning model integration. Finally, cloud computing platforms offer scalable infrastructure for data processing and model training. This scalability is crucial for handling the potentially large datasets involved in financial forecasting.

3.2.2 Economic Feasibility

The economic feasibility of the project hinges on its potential to generate significant value. The system's core strength lies in its ability to improve stock price prediction accuracy compared to existing methods. This enhanced accuracy can empower investors and financial institutions to make more informed investment decisions, potentially leading to better returns. Furthermore, the project has the potential to be commercially viable in the future. By offering the system as a service or financial product, a sustainable revenue stream could be established. The return on investment for stakeholders will depend on the effectiveness of the system in generating superior results.

3.2.3 Social Feasibility

The social feasibility of the project is driven by its potential to positively impact individuals and the financial system as a whole. Improved stock price predictions can empower individuals and institutions to make more informed investment decisions. This can lead to increased participation in the financial markets and potentially fairer outcomes. Additionally, the project can contribute to greater market transparency and efficiency. By providing more accurate insights, the system can foster trust and stability within the financial system. Finally, the project has the potential to drive innovation in financial forecasting techniques. By introducing a novel approach that

leverages KGs and LLMs, the project can pave the way for further advancements in this critical field. However, social considerations regarding bias in LLM outputs and the ethical implications of AI in finance need to be addressed to ensure responsible use of this technology.

3.3 System Specifications

3.3.1 Hardware Specifications

A) Processing Power

- The project will involve data processing tasks such as knowledge graph construction, LLM training, and machine learning model development. These tasks can be computationally intensive, particularly when dealing with large datasets.
- An Intel core i5 / AMD Ryzen 5 processor or above would be adequate for the smooth functionality of this project.

B) Memory (RAM)

- Sufficient RAM is crucial for handling large datasets during training and processing. The exact amount of RAM needed will depend on the size and complexity of your knowledge graph, LLM architecture, and chosen machine learning models. 8GB RAM or above should suffice for the working of this project.

C) Storage

- The project will require storage for various data sources: financial data, news articles, social media information, the constructed knowledge graph, and trained LLM models.
- The suggested storage capacity is a minimum of 2 gigabytes.

D) Graphics Processing Unit (GPU) :

- While not essential for the core functionalities, utilizing GPUs can significantly accelerate LLM training, especially for complex models.

3.3.2 Software Specifications

The entire project has been developed in Python.

A) Knowledge Graph Construction and Embedding

- pandas for optimized data retrieval and storage
- pyvis.network for visualizing the triplets
- google.generativeai for generating and embedding triplets from input news

B) Sentiment Analysis and LLM Reasoning

- transformers for access to finBERT tokenizer and model
- pytorch to perform the sentiment analysis from input news
- google.generativeai for generating analysis from input news
- pandas for efficient retrieval and storage of data

C) Data Processing and Back-Testing

- datetime for uniform date-time management
- marketaux for retrieving stock-wise news
- requests for making API calls to marketaux
- lumibot for trade strategy development and backtesting

D) Project Management and Collaboration

- github for project version control
- figma for architecture development
- vs-code for project development

4. DESIGN APPROACH & DETAILS

4.1 Design Approach

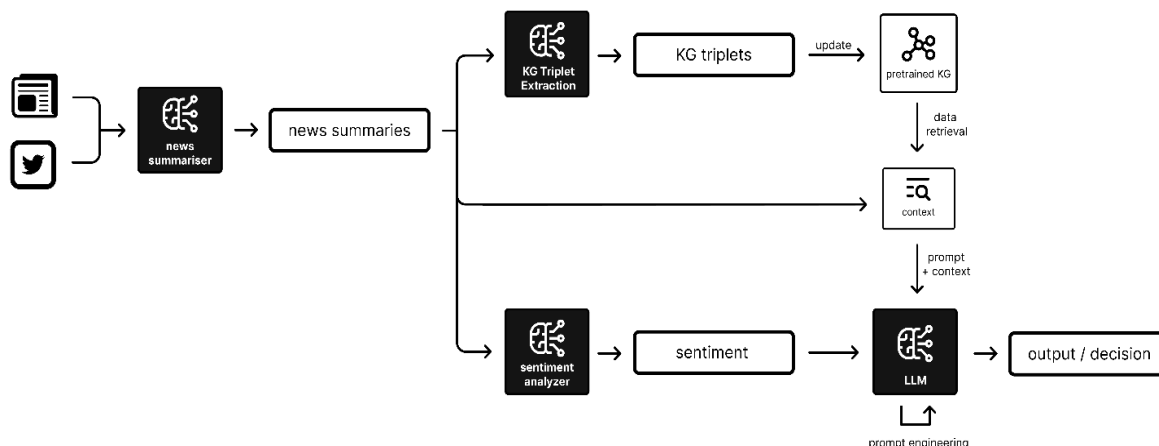


Fig. 4.1 LLM-KG reasoning architecture

The Project's design has been abstracted into several modules to streamline development, enhance modularity, and facilitate scalability. The modules involve stock and strategy movement backtesting, news extraction, sentiment analysis, knowledge triplet generation & extraction, triplet embedding generation, semantic search, and LLM reasoning using single-step prompt reasoning techniques. The following are the modules discussed in detail:

Module 1 - Backtesting

Backtesting serves as the conclusive testing phase for our system. Leveraging the Lumibot Python library, we meticulously evaluate our strategies, integrating insights derived from sentiment analysis and/or LLM prompt engineering. Additionally, we implement advanced quant finance techniques, including but not limited to, take profit prices, stop loss prices, cash risks, and dynamic purchase quantities. This component not only facilitates the selection of specific stocks but also enables a comprehensive performance comparison with the strategies we meticulously develop.

Module 2 - News Extraction Model

The system employs MarketAux APIs to procure stock-specific news, categorized by dates, which are subsequently stored in the database for testing. As the system approaches its final stages, these APIs will be instrumental in accessing real-time news data, thereby enhancing the system's capability to promptly respond to evolving market conditions. Currently we are only restricting the system to Amazon, AMD, Apple, Google, JP Morgan, Microsoft, Nvidia and Tesla related news articles for testing purposes.

Module 3 - Sentiment Analysis

Sentiment analysis using FinBERT harnesses a specialized adaptation of the BERT (Bidirectional Encoder Representations from Transformers) model, tailored to discern sentiment nuances within financial texts. This methodology empowers the classification of sentiments—be it positive, negative, or neutral—embedded in various financial sources such as news articles, social media posts, or corporate reports. This capability renders invaluable insights for traders, investors, and financial analysts, enabling them to make well-informed decisions rooted in the prevailing sentiment landscape.

Module 4 - Knowledge Triplet Extraction

The system utilizes Google's Gemini Pro API to generate KG triplets from unstructured text while reasoning out why the triplets have been deemed to be the most detrimental to the price movement of the specific stock. Parsing techniques are applied to identify and extract relationships between entities, attributes, and values (head, relation and tail) within the text. These triplets provide structured data in the JSON format, facilitating a key task in the creation of this project - knowledge graph construction. Leveraging Google's ongoing advancements in NLP ensures the system remains adept at handling diverse text sources and ever evolving linguistic patterns.

Module 5 - LLM reasoning with prompt engineering

The reasoning method involves utilizing a large language model to provide a confidence score regarding the direction of movement for a financial stock. This process entails formulating a prompt or query tailored to elicit confidence scores from the LLM about the future performance of the stock, while at the same time generating reasons for the confidence scores. The LLM then evaluates various factors such as market trends, company news, and historical data to generate a confidence score indicating the likelihood of the stock's price increasing or decreasing. The LLM used in the system is Google's Gemini Pro API and this LLM has been preferred due to the large amount of free GPU compute units available. The confidence score serves as a quantitative measure of the model's certainty in its prediction, enabling backtesting modules to perform decision making on whether a stock should be bought or sold.

Module 6 - Embedding Generation and Semantic Search

The system uses Google's Embedding-001 API to generate vector embeddings from the extracted triplets. Semantic search is performed over these embeddings to find the K most relevant triplets from the past week and K most relevant triplets in general with respect to the given day's news. These extracted triplets are injected as context into the prompt, this is then used to guide the decision-making and reasoning process of the LLM.

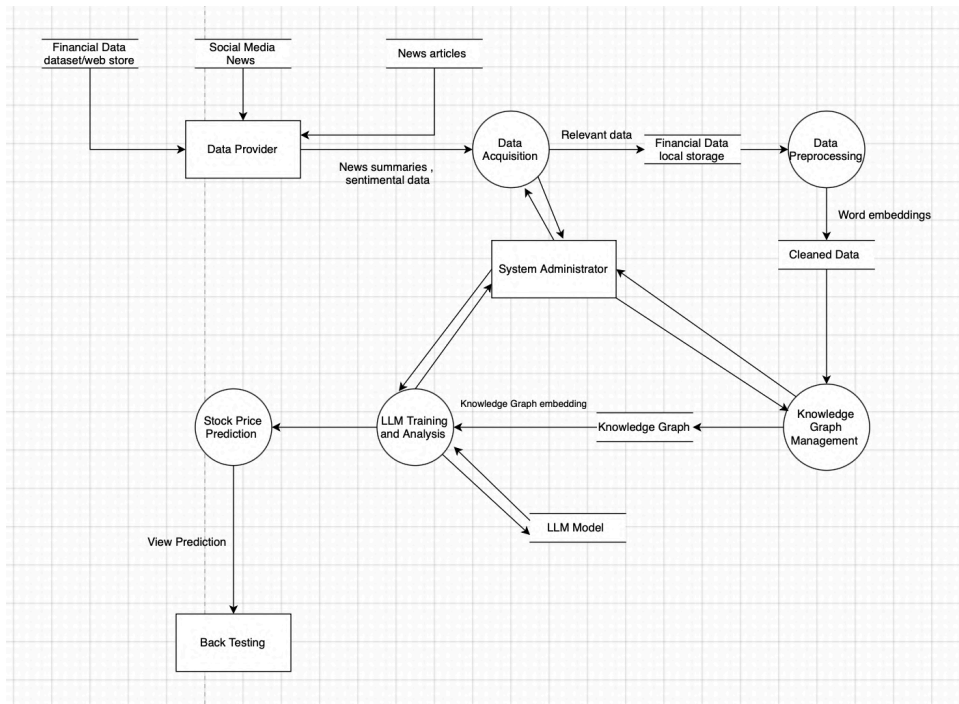


Fig. 4.2 Data Flow diagram

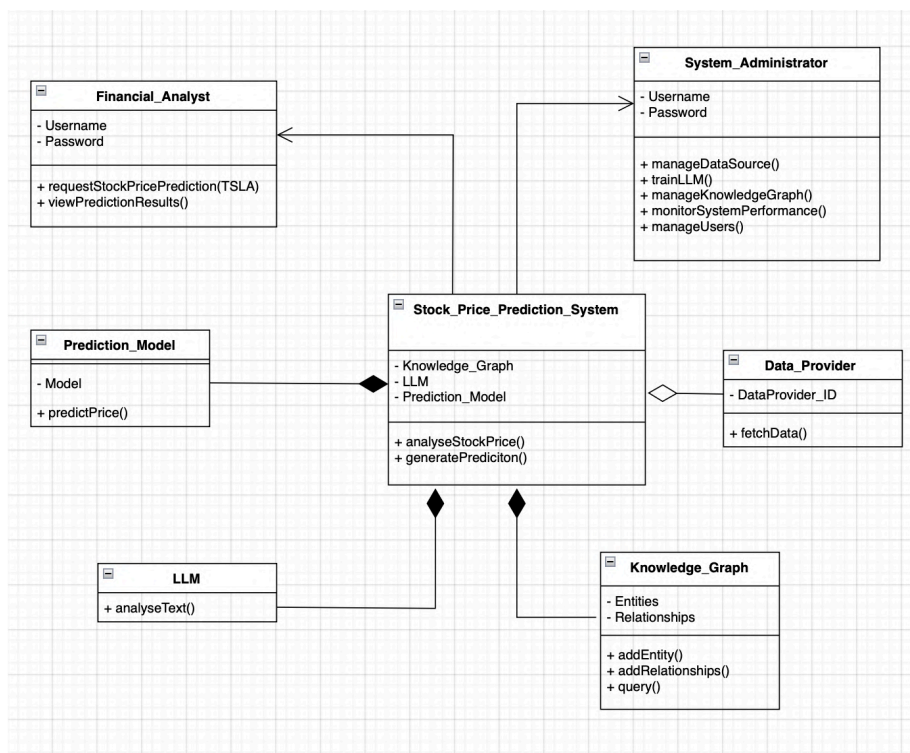


Fig. 4.3 Class diagram

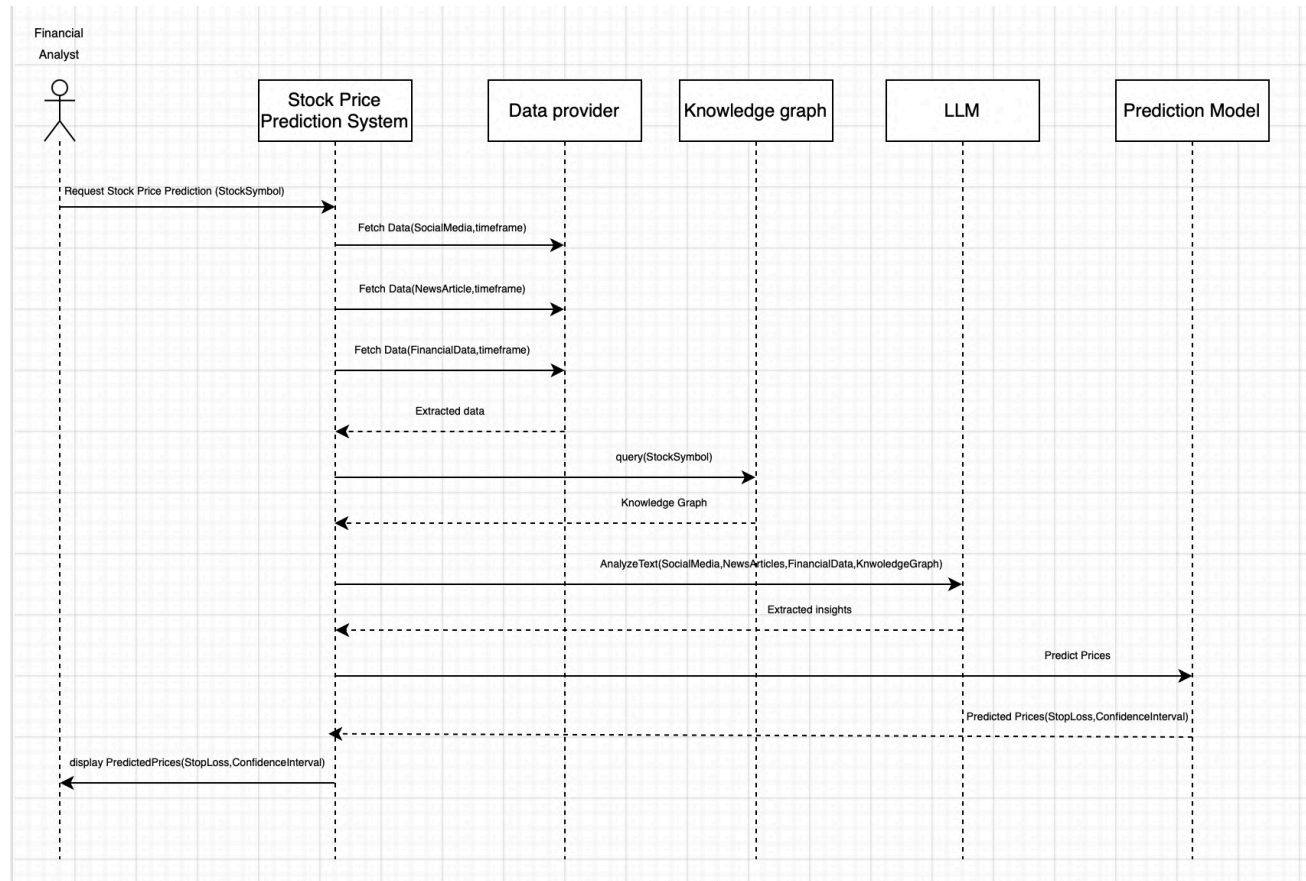


Fig. 4.4 Sequence diagram

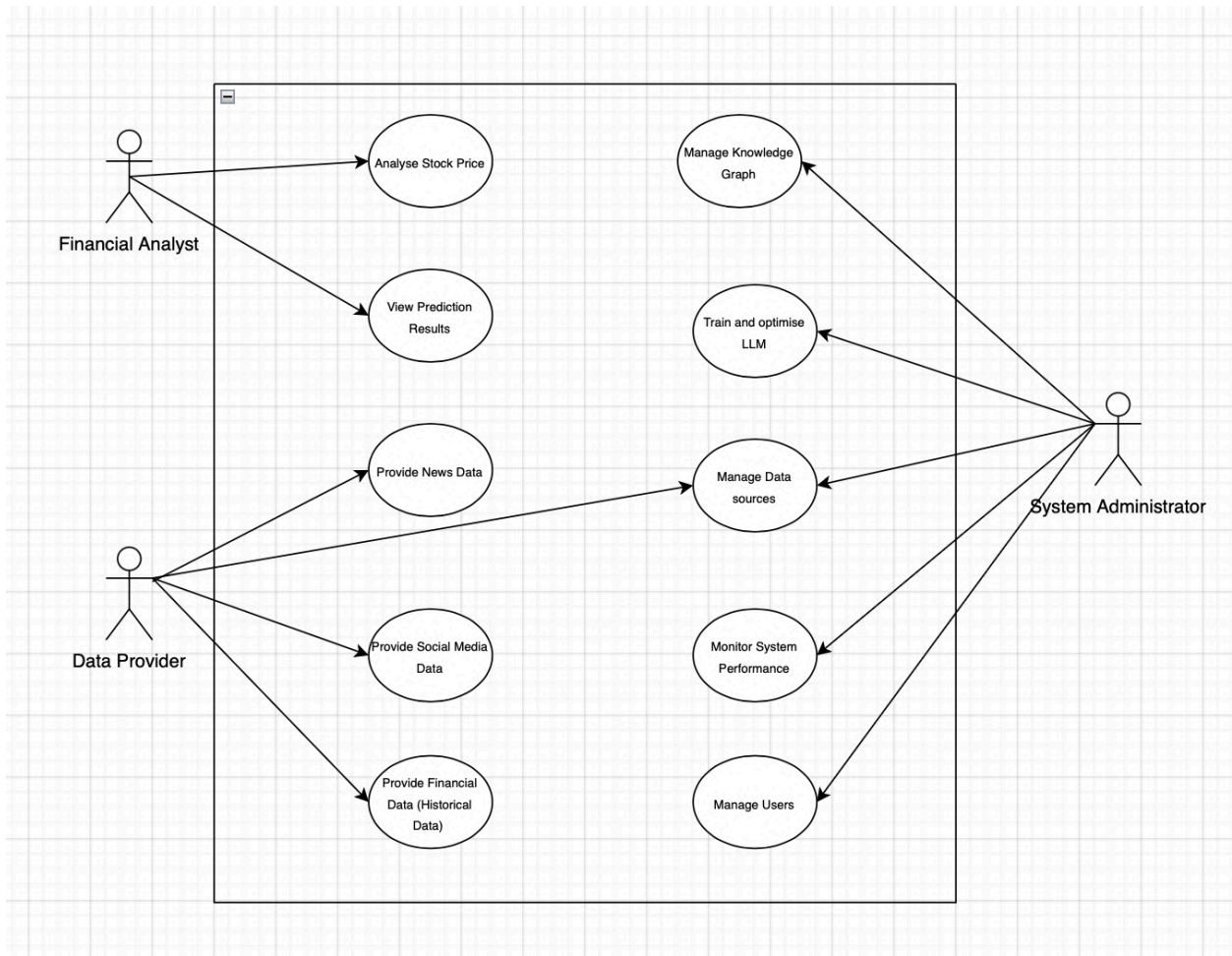


Fig 4.5 Use-case diagram

4.2 Codes and Standards

A) Technical Standards

- **Data Quality**

- Implement data quality checks to ensure the accuracy and consistency of financial data, news articles, and social media information used for training and prediction.
- Consider establishing data quality metrics and procedures for data cleaning and error handling.

- **Model Explainability**

- The system should be designed with explainability in mind. Strive to make the factors influencing the stock price predictions clear and interpretable. This can be

achieved through techniques like feature importance analysis or attention mechanisms in LLMs.

- Documented explanations for model behavior can foster trust in the system's outputs and allow users to understand the rationale behind the predictions.
- **Model Validation and Testing**
 - Utilize rigorous testing and validation procedures to assess the performance of the knowledge graph, LLMs, and the overall prediction model.
 - Employ industry-standard metrics like Mean Squared Error (MSE) or R-squared to evaluate the accuracy of stock price predictions compared to historical data.

B) Ethical Policies

- **Data Privacy**
 - If user data is involved (e.g., social media sentiment analysis), ensure compliance with relevant data privacy regulations like GDPR or CCPA.
 - Implement appropriate data anonymization and security measures to protect user privacy.
- **Algorithmic Bias**
 - Be aware of potential biases within financial data sources, news articles, and LLM training data. Implement techniques like debiasing algorithms or data augmentation to mitigate these biases and ensure fairness in the prediction outcomes.
- **Transparency and Responsible Use**
 - Clearly communicate the limitations and potential risks associated with the system's predictions.
 - Emphasize that the system is a tool to aid decision-making, not a guaranteed predictor of future stock prices.
 - Encourage responsible use of the system by investors and financial institutions, avoiding over-reliance on its outputs.

4.3 Constraints and Alternatives

Constraints

- Google AI Gemini: Only permits a maximum of 60 API calls within a one-minute interval.
- Marketaux: The platform imposes a restriction on account usage, permitting a maximum of 100 API calls. To avail unrestricted access, a subscription purchase is required.

Alternatives

- Google AI Gemini: A suitable alternative to Gemini could be OpenAI GPT models, such as GPT-3. It offers similar capabilities to Gemini in terms of engaging in natural language processing as well as generating KG triplets from text. Other alternatives are Microsoft DialoGPT, BlenderBot etc.
- Marketaux: Several options are available for news retrieval namely Google news API, Yahoo news API, New York Times API, Reuters API etc.

5. SCHEDULE, TASKS AND MILESTONES

5.1 Schedule

The timeline allocated for the completion of this project spans the entire duration of the current academic semester, commencing from January 1st, 2024, to May 9th, 2024. To ensure the attainment of satisfactory outcomes, we delineated this period into three distinct phases, each corresponding to a review milestone.

During the initial review phase, our objective centered on the development of a sentiment analysis bot leveraging the capabilities of FinBERT. Concurrently, we aimed to construct a backtesting module facilitating the comparison of the sentiment analysis bot's predictions against historical stock data.

Subsequently, leading up to the second review, our focus transitioned towards the integration of LLM Reasoning techniques, employing single-step prompt reasoning methodologies. This initiative aimed to refine and augment the predictive capabilities of the sentiment analysis model.

Lastly, as we approached the final review milestone, our objectives encompassed several critical enhancements. These included the expansion of the stock pool for backtesting purposes, the establishment of a comprehensive Knowledge Graph alongside associated embeddings, and the pivotal development of an LLM Reasoning framework seamlessly integrated with Knowledge Graphs—serving as the cornerstone of our project's overarching objectives.

5.2 Tasks

A. RND

Throughout the research and development phase, we diligently assessed the feasibility of our project and conducted an extensive review of existing literature to ascertain the level of prior investigation into the chosen topic. Following thorough examination and analysis of relevant research papers, it became evident that the subject matter is relatively nascent. It was this realization that prompted our decision to embrace the topic for our capstone research endeavor, recognizing the opportunity to contribute to the advancement of knowledge in this emerging field.

B. Requirements analysis

This segment of our project was dedicated to the exploration and examination of pertinent technologies essential for the project's successful execution. We engaged in meticulous research encompassing diverse methodologies of LLM Reasoning, an array of sentiment analysis tools tailored for stock price prediction, and comprehensive investigations into the mechanics of Knowledge Graphs.

C. Guide Suggestions

This phase of our project was specifically allocated to presenting our project to our faculty member Dr. Durgesh Kumar and soliciting his expert advice. Our primary objective during this stage was to seek guidance from a domain expert regarding the feasibility and scope of our project. This collaborative effort aimed to gain invaluable

insights and assessments from seasoned professionals in the field, ultimately enhancing our understanding of the project's viability.

D. Backtesting Module

Our first technical task towards the pursuit of this project involved the utilization of Lumibot for conducting backtesting analyses. Backtesting represents the pivotal concluding phase for our system. Through the utilization of the Lumibot Python library, we conduct thorough assessments of our strategies, incorporating insights gleaned from sentiment analysis and/or LLM prompt engineering. Furthermore, we employ sophisticated quantitative finance methodologies, encompassing aspects such as take profit prices, stop loss prices, cash risks, and dynamic purchase quantities. This integral component not only aids in the meticulous selection of individual stocks but also facilitates a comprehensive performance evaluation vis-à-vis the strategies meticulously devised.

E. News Extraction

This task involves Integration of MarketAux APIs for Stock-Specific News Retrieval. We utilize MarketAux APIs to fetch stock-specific news articles, organized by date, and store them in the project's database for subsequent testing and analysis. This task serves as a foundational step in enhancing the system's capability to respond to market dynamics effectively. For initial testing purposes we restrict the system's focus to news articles related to Tesla but later we expand it to other prominent companies including Amazon, AMD, Apple, Google, JP Morgan, Microsoft and Nvidia.

F. Sentiment Analysis

In this task we utilize FinBERT for sentiment analysis employing a custom adaptation of the BERT (Bidirectional Encoder Representations from Transformers) model, specifically designed to detect subtle sentiment variations within financial texts. This approach enables the classification of sentiments—whether positive, negative, or neutral—present

in diverse financial sources like news articles, social media content, and corporate reports. Such functionality offers significant value to traders, investors, and financial analysts by furnishing them with actionable insights derived from the prevailing sentiment trends, thereby facilitating informed decision-making processes.

G. Knowledge Triplet Extraction

Within this part we utilize Google's Gemini Pro API to derive KG triplets from unstructured text, elucidating the rationale behind why these triplets are considered the most influential factors affecting the price dynamics of a given stock. Utilizing parsing techniques, the system identifies and extracts relationships between entities, attributes, and values (head, relation, and tail) embedded within the text. These extracted triplets are then structured into JSON format, facilitating a crucial aspect of the project—knowledge graph formation. By harnessing Google's continual advancements in Natural Language Processing (NLP), the system ensures its adaptability to diverse textual sources and evolving linguistic intricacies.

H. LLM Reasoning with prompt engineering

Finally, we utilize the reasoning methodology of a large language model (LLM) to assess the likelihood of a financial stock's future movement, providing a confidence score indicative of its direction. This process involves crafting a tailored prompt or query to solicit confidence scores from the LLM regarding the stock's prospective performance, while concurrently generating rationales for these scores. Drawing upon various factors such as market trends, corporate news, and historical data, the LLM evaluates the inputs to determine the probability of the stock's price appreciation or depreciation. The preferred LLM utilized in the system is Google's Gemini Pro API, primarily due to its access to abundant GPU compute resources. The confidence score acts as a quantitative gauge of the model's confidence in its prediction, thereby enabling decision-making within backtesting modules regarding stock purchase or sale strategies. We integrate this with Knowledge graph embeddings to further improve upon the LLM prediction results.

5.3 Milestones

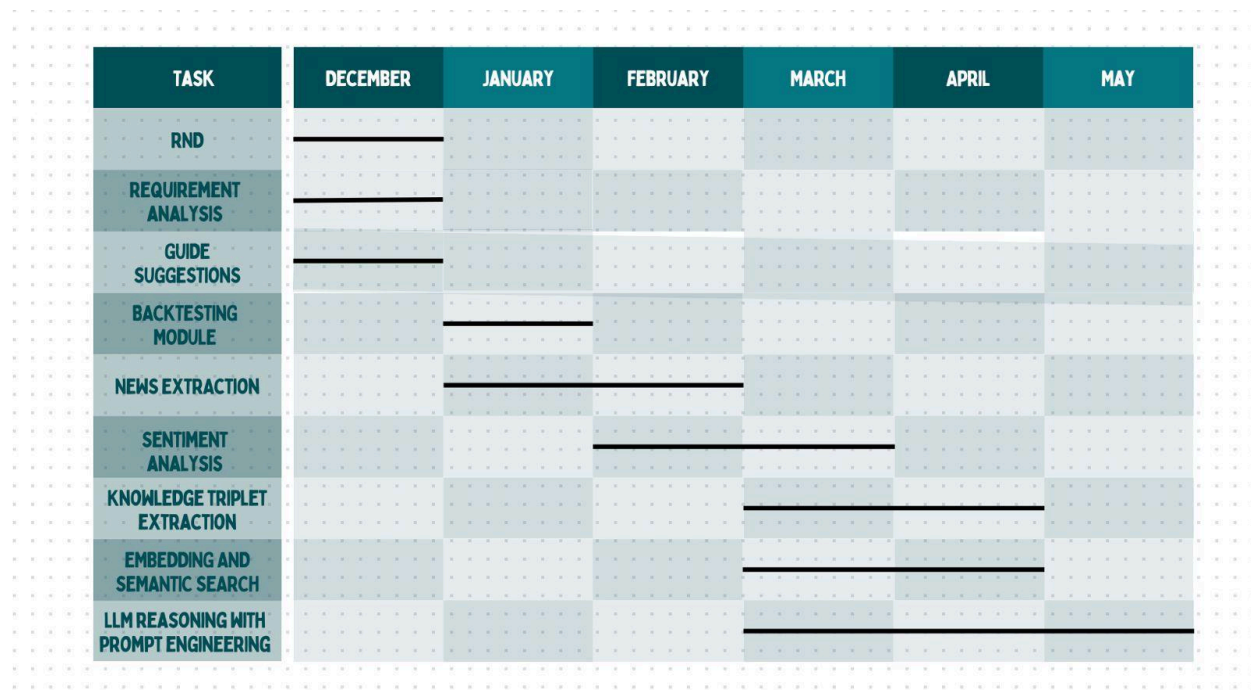


Fig 5.1 Gantt chart

6. PROJECT DEMONSTRATION

```
finbert_utils.py X
finbert_utils.py > ...
Click here to ask Blackbox to help you code faster
1 from transformers import AutoTokenizer, AutoModelForSequenceClassification
2 import torch
3
4 device = "cuda:0" if torch.cuda.is_available() else "cpu"
5
6 tokenizer = AutoTokenizer.from_pretrained("ProsusAI/finbert")
7 model = AutoModelForSequenceClassification.from_pretrained("ProsusAI/finbert").to(device)
8 labels = ["positive", "negative", "neutral"]
9
10 def estimate_sentiment(news):
11     if news:
12         tokens = tokenizer(news, return_tensors="pt", padding=True).to(device)
13
14         result = model(tokens["input_ids"], attention_mask=tokens["attention_mask"])[
15             "logits"
16         ]
17
18     return result
```

PS C:\Users\nisha\Documents\GitHub\trading_bot> python .\finbert_utils.py

C:\Users\nisha\pyenv\pyenv-win\versions\3.11.4\Lib\site-packages\torch_utils.py:776: UserWarning: TypedStorage is deprecated. It will be removed in the future and UntypedStorage will be the only storage class. This should only matter to you if you are using storages directly. To access UntypedStorage directly, use tensor.untyped_storage() instead of tensor.storage().

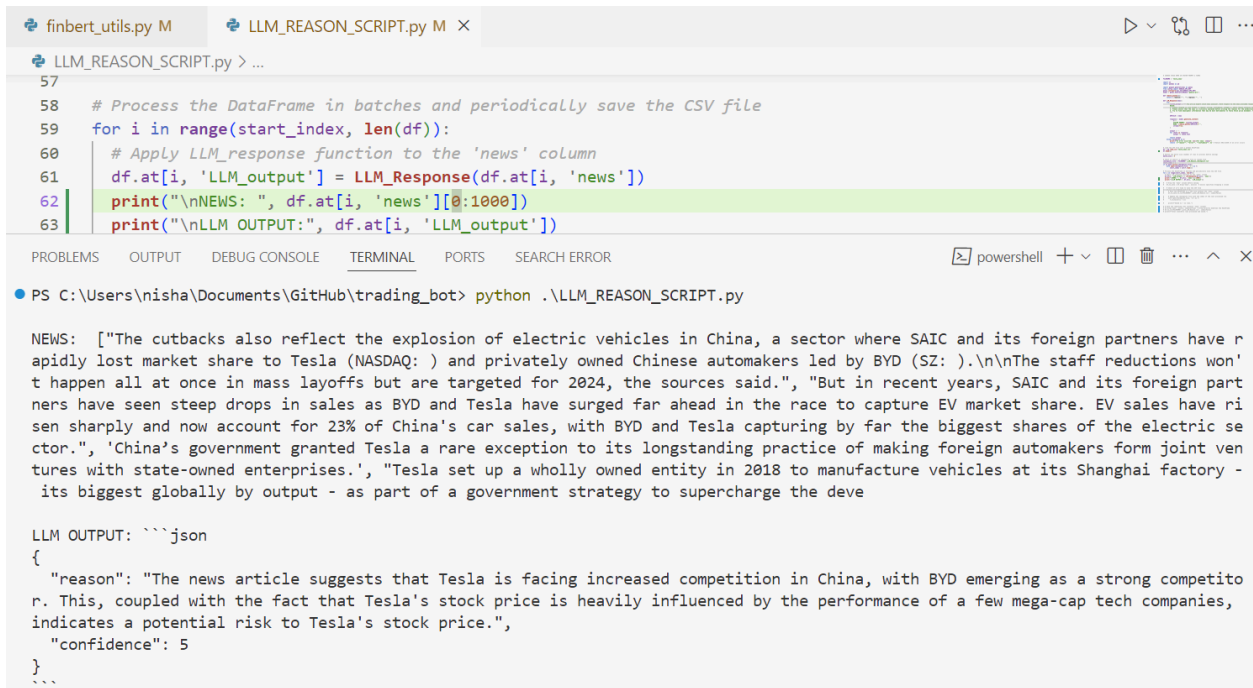
return self.fget.__get__(instance, owner)()

['markets responded negatively to the news!', 'traders were displeased!']

tensor(0.9979, device='cuda:0', grad_fn=<SelectBackward0>) negative

PS C:\Users\nisha\Documents\GitHub\trading_bot>

Fig 6.1 Sentiment Analysis Output




```
finbert_utils.py M LLM_REASON_SCRIPT.py M X
LLM_REASON_SCRIPT.py > ...
57
58 # Process the DataFrame in batches and periodically save the CSV file
59 for i in range(start_index, len(df)):
60     # Apply LLM_response function to the 'news' column
61     df.at[i, 'LLM_output'] = LLM_Response(df.at[i, 'news'])
62     print("\nNEWS: ", df.at[i, 'news'][:1000])
63     print("\nLLM OUTPUT:", df.at[i, 'LLM_output'])
```

PS C:\Users\nisha\Documents\GitHub\trading_bot> python .\LLM_REASON_SCRIPT.py

NEWS: ["The cutbacks also reflect the explosion of electric vehicles in China, a sector where SAIC and its foreign partners have rapidly lost market share to Tesla (NASDAQ:) and privately owned Chinese automakers led by BYD (SZ:).\n\nThe staff reductions won't happen all at once in mass layoffs but are targeted for 2024, the sources said.", "But in recent years, SAIC and its foreign partners have seen steep drops in sales as BYD and Tesla have surged far ahead in the race to capture EV market share. EV sales have risen sharply and now account for 23% of China's car sales, with BYD and Tesla capturing by far the biggest shares of the electric sector.", "China's government granted Tesla a rare exception to its longstanding practice of making foreign automakers form joint ventures with state-owned enterprises.", "Tesla set up a wholly owned entity in 2018 to manufacture vehicles at its Shanghai factory - its biggest globally by output - as part of a government strategy to supercharge the deve

LLM OUTPUT: ``json
{
 "reason": "The news article suggests that Tesla is facing increased competition in China, with BYD emerging as a strong competitor. This, coupled with the fact that Tesla's stock price is heavily influenced by the performance of a few mega-cap tech companies, indicates a potential risk to Tesla's stock price.",
 "confidence": 5
}

Fig 6.2 LLM Reasoning Output



```
TRIPLET_EXTRACTION.py M X
TRIPLET_EXTRACTION.py > ...
14 # Iterate over each row in the DataFrame
15 for index, row in df.iterrows():
16
17     try:
18         output = json.loads(row['LLM_Triplet'])
19         if 'triplets' not in output and 'error' not in output:
20             print(f"Row {index + 1} Date - {row['date']}: LLM_Triplet doesn't contain 'triplets' or 'error'.")
21             warning_flag = True
22         except json.JSONDecodeError:
23             print(f"Date - {row['date']}: LLM_Triplet is not in valid JSON format.")
24             warning_flag = True
```

PS C:\Users\nisha\Documents\GitHub\trading_bot> python Triplet_extraction.py

{'date': '2022-02-27', '\ntriplet': {'head': 'Tesla', 'relation': 'may face competition from', 'tail': 'Porsche'}, '\nhead': 'Tesla', '\nrelation': 'may face competition from', '\ntail': 'Porsche', '\nsentence': 'Tesla may face competition from Porsche'}

{'date': '2022-02-27', '\ntriplet': {'head': 'Elon Musk', 'relation': 'is expanding', 'tail': 'number of games compatible with Tesla cars'}, '\nhead': 'Elon Musk', '\nrelation': 'is expanding', '\ntail': 'number of games compatible with Tesla cars', '\nsentence': 'Elon Musk is expanding number of games compatible with Tesla cars'}

{'date': '2022-02-27', '\ntriplet': {'head': 'Elon Musk', 'relation': 'aims to include', 'tail': 'more than 50,000 video games from Stream on Tesla infotainment system'}, '\nhead': 'Elon Musk', '\nrelation': 'aims to include', '\ntail': 'more than 50,000 video games from Stream on Tesla infotainment system', '\nsentence': 'Elon Musk aims to include more than 50,000 video games from Stream on Tesla infotainment system'}

Fig 6.3 Triplet Generation and Extraction Output

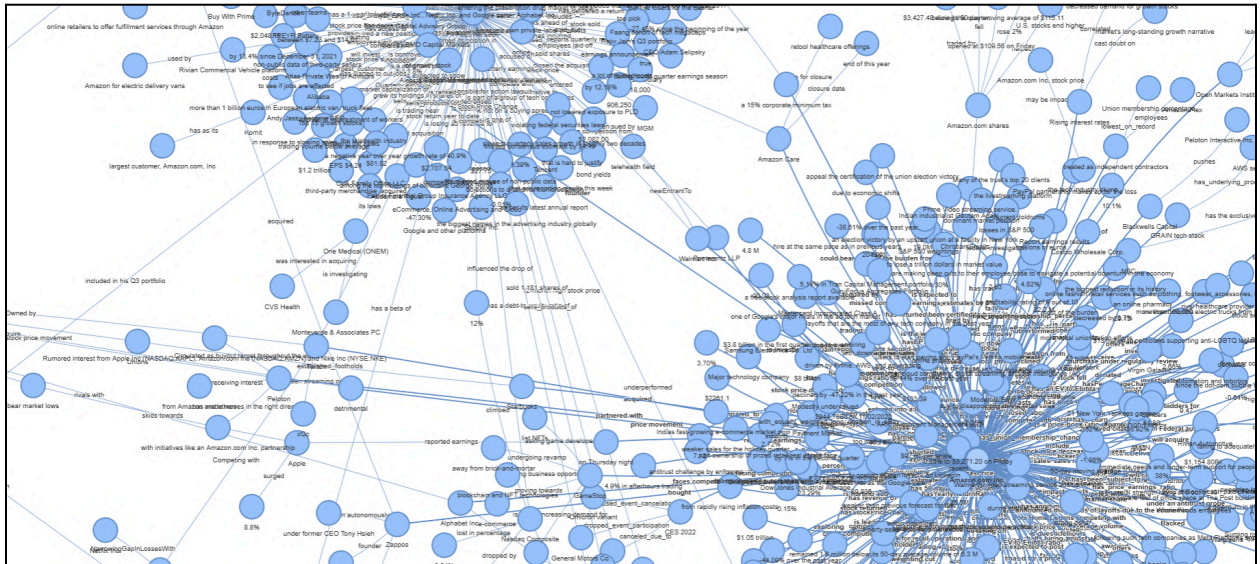
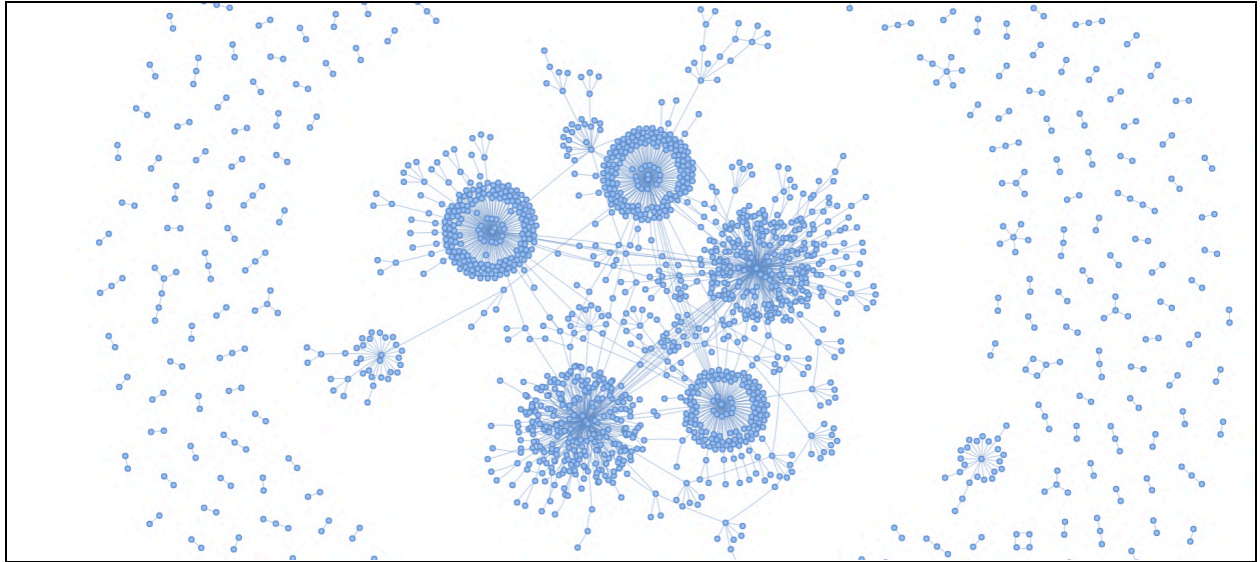
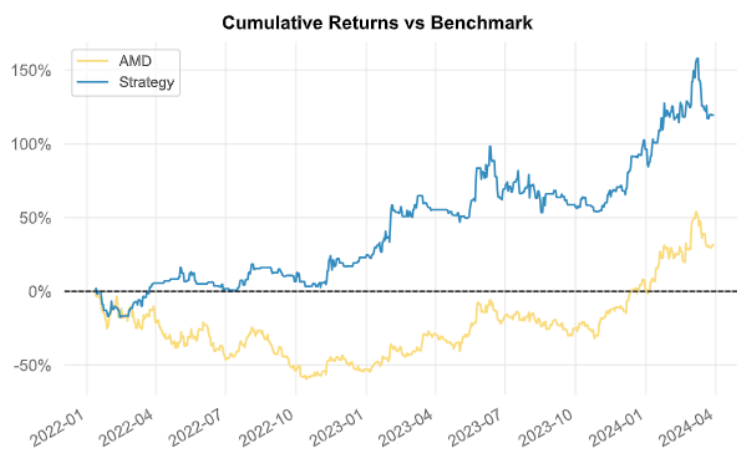
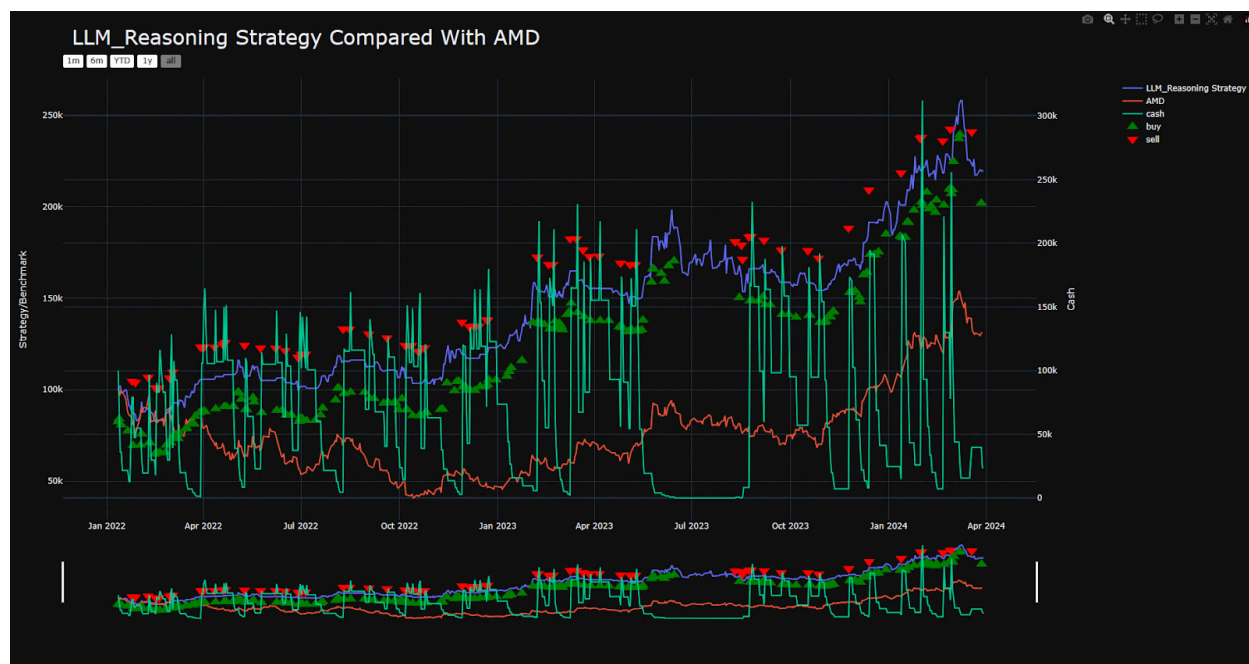


Fig 6.4 Knowledge Graph visualization 2000/31000



Key Performance Metrics

Metric	AMD	Strategy
Risk-Free Rate	5.21%	5.21%
Time in Market	69.0%	47.0%
Total Return	31.45%	119.49%
CAGR% (Annual Return)	13.16%	42.7%
Sharpe	0.4	1.07
ROMaD	0.22	1.89
Corr to Benchmark	1.0	0.39
Prob. Sharpe Ratio	22.85%	63.19%
Smart Sharpe	0.39	1.04
Sortino	0.6	1.66
Smart Sortino	0.59	1.62
Sortino/ $\sqrt{2}$	0.43	1.18
Smart Sortino/ $\sqrt{2}$	0.41	1.15
Omega	1.25	1.25

Fig 6.5 Strategy backtesting output

7. COST ANALYSIS / RESULTS & DISCUSSIONS

7.1 Cost Analysis

For our project, we availed the Marketaux premium basic subscription to obtain access to 2500 daily calls, facilitating the retrieval of ample historical data concerning prominent stocks, including but not limited to Tesla, Apple, JP Morgan, and AMD. This expenditure amounted to \$30 per month or 2504 INR for the monthly subscription fee.

Although not utilized by us, incorporating additional resources such as Gemini AI Advanced or other paid models like GPT-4 would be necessary to refine the module and enhance result accuracy, thereby contributing to the overall project expenditure.

7.2 Results and Discussion

A. Consistency

Before we begin comparing results between Sentiment Strategy, LLM Reasoning Strategy and LLM Reasoning with KG strategy, it is essential to determine whether the models we are using perform consistently. To perform this check, news sentiments using ‘ProsusAI/finbert’ as well as confidence scores using ‘google/geminipro’ (temperature = 0.6) were generated on retrieved Tesla stock news three times each. These sentiments and confidence scores are used to predict Tesla stock prices and the outputs are compared against each other.

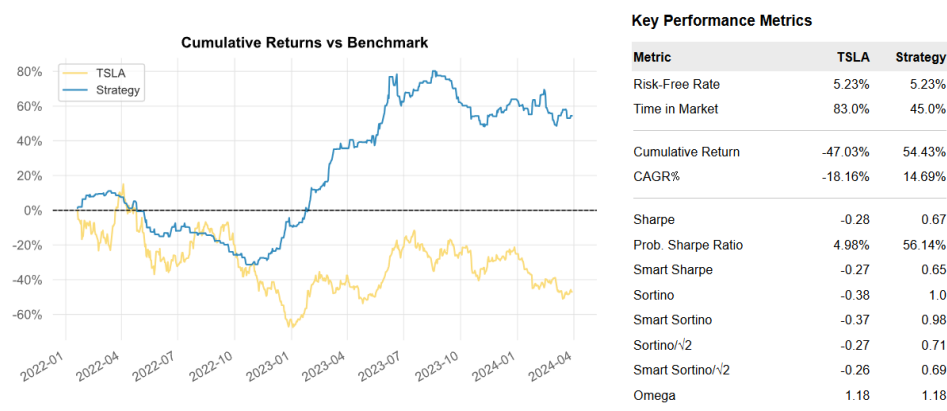
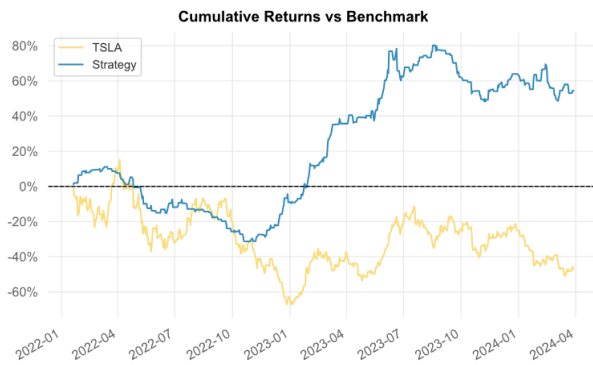


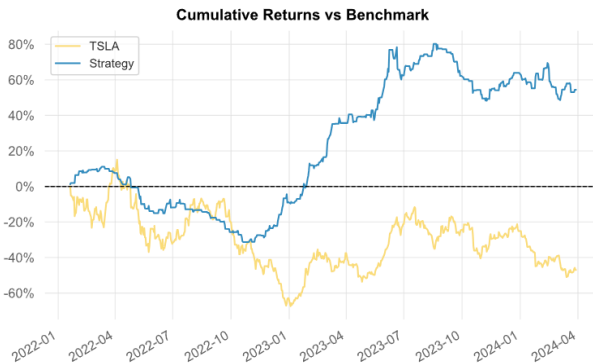
Fig. 7.1 Sentiment Analysis on Tesla news - test 1



Key Performance Metrics

Metric	TSLA	Strategy
Risk-Free Rate	5.23%	5.23%
Time in Market	83.0%	45.0%
Cumulative Return	-47.03%	54.43%
CAGR%	-18.16%	14.69%
Sharpe	-0.28	0.67
Prob. Sharpe Ratio	4.98%	56.14%
Smart Sharpe	-0.27	0.65
Sortino	-0.38	1.0
Smart Sortino	-0.37	0.98
Sortino/ $\sqrt{2}$	-0.27	0.71
Smart Sortino/ $\sqrt{2}$	-0.26	0.69
Omega	1.18	1.18

Fig. 7.2 Sentiment Analysis on Tesla news - test 2

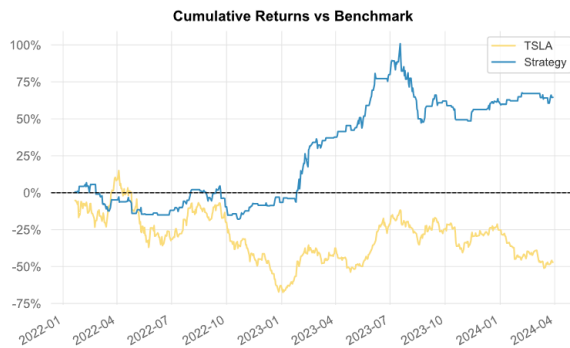


Key Performance Metrics

Metric	TSLA	Strategy
Risk-Free Rate	5.23%	5.23%
Time in Market	83.0%	45.0%
Cumulative Return	-47.03%	54.43%
CAGR%	-18.16%	14.69%
Sharpe	-0.28	0.67
Prob. Sharpe Ratio	4.98%	56.14%
Smart Sharpe	-0.27	0.65
Sortino	-0.38	1.0
Smart Sortino	-0.37	0.98
Sortino/ $\sqrt{2}$	-0.27	0.71
Smart Sortino/ $\sqrt{2}$	-0.26	0.69
Omega	1.18	1.18

Fig. 7.3 Sentiment Analysis on Tesla news - test 3

The sentiment analysis model using FinBERT performs very consistently, resulting in the same output every time the model is run on the same news. However, slight inconsistencies were noticed during price prediction using LLM Reasoning.



Key Performance Metrics

Metric	TSLA	Strategy
Risk-Free Rate	5.24%	5.24%
Time in Market	83.0%	40.0%
Cumulative Return	-47.07%	64.61%
CAGR%	-18.2%	17.04%
Sharpe	-0.28	0.69
Prob. Sharpe Ratio	4.97%	54.17%
Smart Sharpe	-0.27	0.67
Sortino	-0.38	1.0
Smart Sortino	-0.37	0.98
Sortino/ $\sqrt{2}$	-0.27	0.71
Smart Sortino/ $\sqrt{2}$	-0.26	0.69
Omega	1.2	1.2

Fig. 7.4 LLM Reasoning on Tesla news - test 1

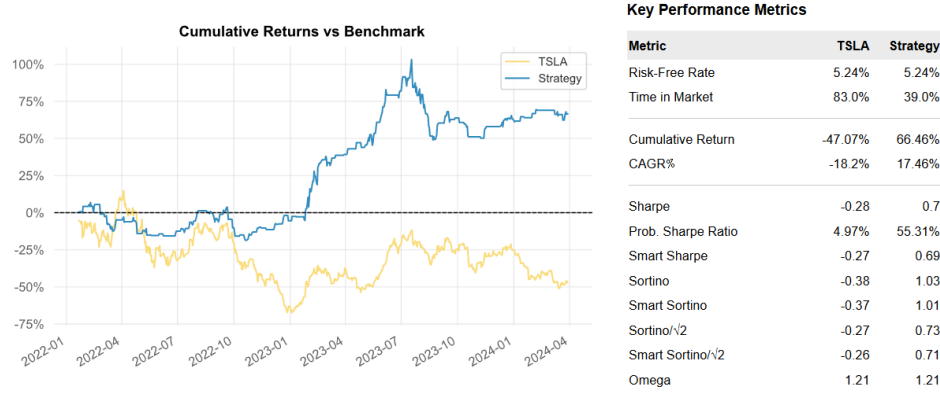


Fig. 7.5 LLM Reasoning on Tesla news - test 2

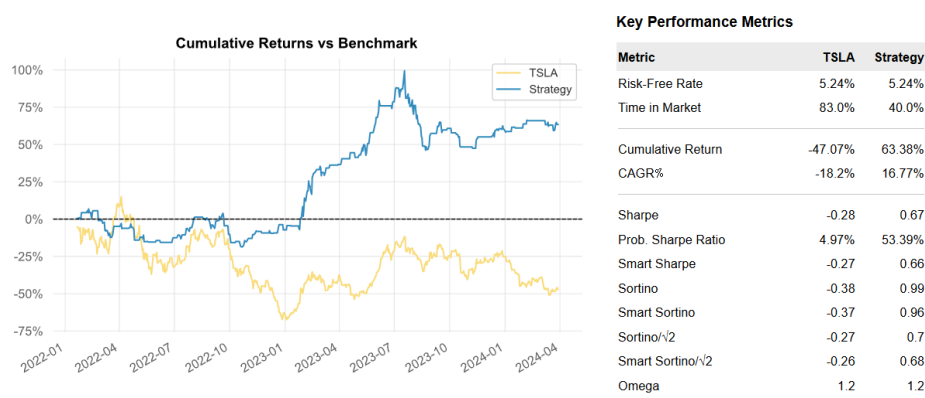


Fig. 7.6 LLM Reasoning on Tesla news - test 3

LLM Reasoning is purely a text-generation task hence being highly probabilistic in nature while Sentiment Analysis is a classification task and hence has a very consistent output. The change in cumulative return has a variation of **3.08%**. Nonetheless, since only three LLM Reasoning outputs have been generated to test consistency of the model, the variation in cumulative return can be approximated to about 10%. This can be considered as a consistent model due to LLM's black box nature and their subjectivity to hallucination. Therefore, both models performed consistently without drastic change. Now, we can compare the performance of the 3 strategies.

B. Performance

Due to the vast amount of finance news available on the top tech stocks, we have considered the following eight stocks for our evaluation: AMZN (Amazon), AMD (Advanced Micro Devices), AAPL (Apple), GOOG (Alphabet Class C), JPM (JPMorgan Chase & Co), MSFT (MicroSoft), NVDA (Nvidia), TSLA (Tesla). The process of evaluation involved collecting the stock-wise news from MarketAux. The collected news is then used to perform sentiment analysis as well as extract LLM reasoning confidence scores. To extract LLM reasoning with KG confidence scores, the news is used to generate triplets and their respective embeddings. Semantic search is

performed on the embeddings using the news per day as the query. Top K triplets are retrieved based on whether the triplet was generated within the past week of the news being analyzed and is injected as Last Week Context. Top K triplets in general are also extracted and injected as General Context. These contexts provide the LLM with necessary information to base its reasoning as to whether one should buy or sell a particular stock, along with its reasoning. The confidence scores generated are the LLM-KG reasoning confidence scores. These confidence scores are used to perform back-testing on the selected stocks and their cumulative return was tested. The table 7.1 encompasses the measured performance results.

Table. 7.1 Performance results

Stock	Single Investment	Sentiment Analysis	LLM Reasoning	LLM_KG Reasoning
AMZN (Amazon)	9.18%	18.67%	-12.29%	-6.11%
AMD (Advanced Micro Devices)	31.45%	86.2%	119.49%	92.93%
AAPL (Apple)	-0.82%	23.04%	18.32%	41.97%
GOOG (Alphabet Class C)	7.49%	44.82%	24.29%	-18.79%
JPM (JPMorgan Chase & Co)	27.17%	29.45%	35.68%	44.11%
MSFT (MicroSoft)	36.3%	31.04%	43.45%	67.62%
NVDA (Nvidia)	238.67%	37.47%	285.86%	409.97%
TSLA (Tesla)	-47.03%	54.43%	66.46%	82.16%

Out of the 8 stocks measured for their performance for various LLM_KG Reasoning performs with the highest cumulative return for 5 stocks namely. Fig. 7.7 compares the performance of the LLM Reasoning, LLM_KG Reasoning and Single-time Investment for Nvidia, a tech stock with an extremely positive return. This displays how LLM Reasoning can further improve the cumulative returns yielded and how adding Financial Market Context improves upon that. Fig 7.8 compares the performance of the LLM Reasoning, LLM_KG Reasoning and Single-time Investment for Tesla, a tech stock with a gradual negative return. LLM Reasoning performs exceptionally well even for a negative performing stock and LLM_KG Reasoning improves upon performance by 21.8%. The improved performance is also demonstrated in AAPL stock where the stock dips for the first half of the measured period of time and rises through the second half. Fig 7.9 contains a graphical comparison of AAPL performance

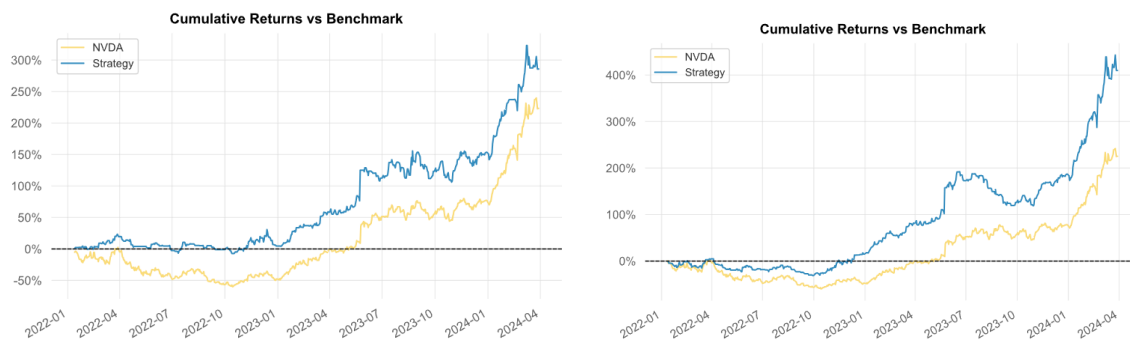


Fig. 7.7 NVDA Performance LLM (left) vs LLM_KG (right)

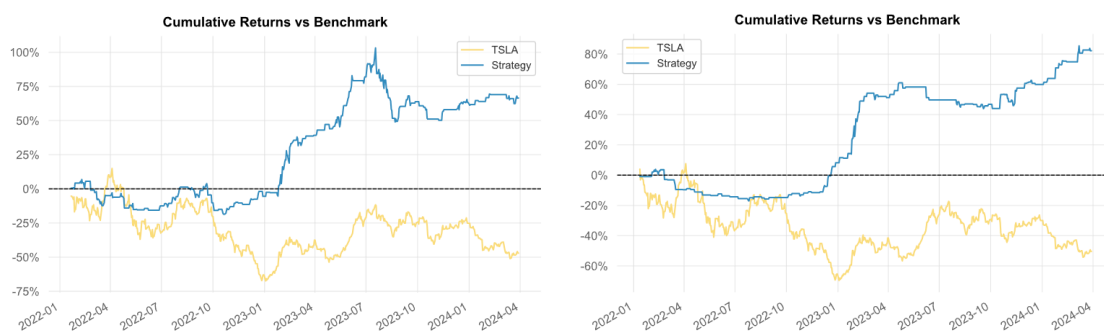


Fig. 7.8 TSLA Performance LLM (left) vs LLM_KG (right)

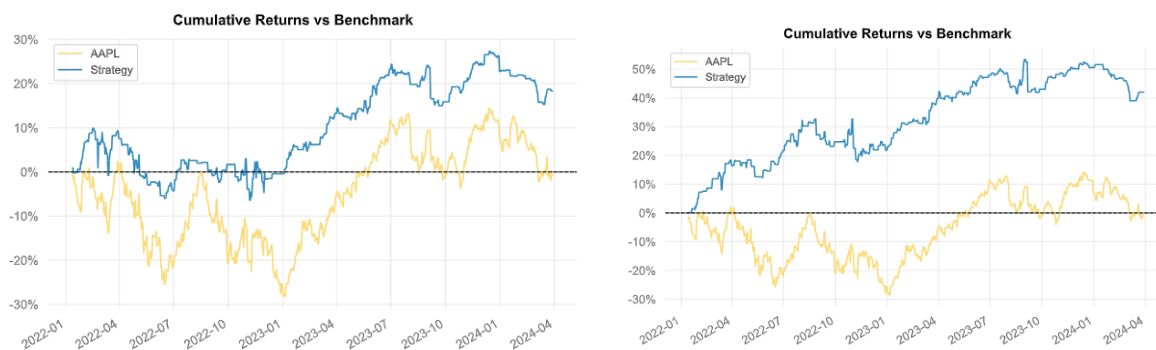


Fig. 7.8 AAPL Performance LLM (left) vs LLM_KG (right)

While LLM Reasoning without added context performs with the highest return for a only a single stock (Fig 7.10), this might not be reflective of its performance as a predictive model as without considering LLM_KG Reasoning, LLM Reasoning would have demonstrated highest performance for 5 out of the 8 stocks measured.

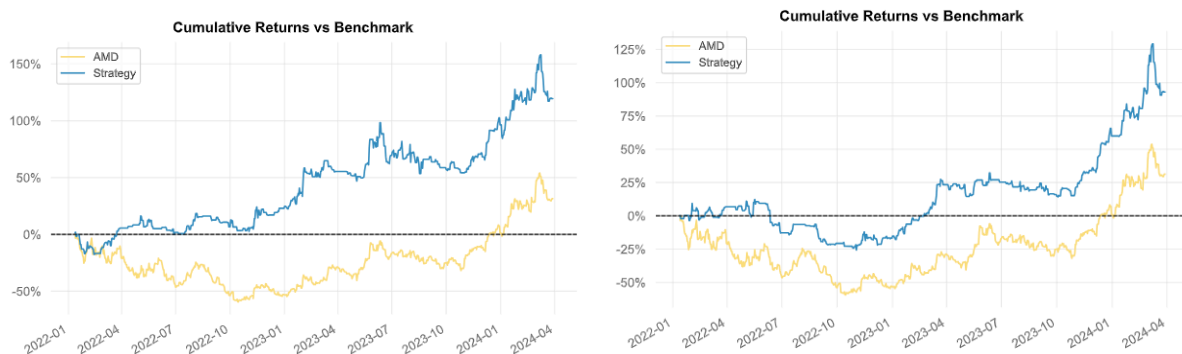


Fig. 7.10 AMD Performance LLM (left) vs LLM_KG (right)

Sentiment Analysis demonstrates highest yield for 2 stocks while a single time investment never exhibits highest performance for any of the stock. However, it is important to note that the improved LLM Reasoning and LLM_KG Reasoning performance does consistently outperform the single time investment performance. For both Amazon and Google, Sentiment Analysis and Single-time investment outperform the LLM Reasoning strategies. In fact, both LLM Reasoning strategies result in extremely negative yield at the end of the back-testing. Fig 7.11 and 7.12 compare the performances of AMZN and GOOG stocks on FinBERT Sentiment Analysis and LLM_KG Reasoning

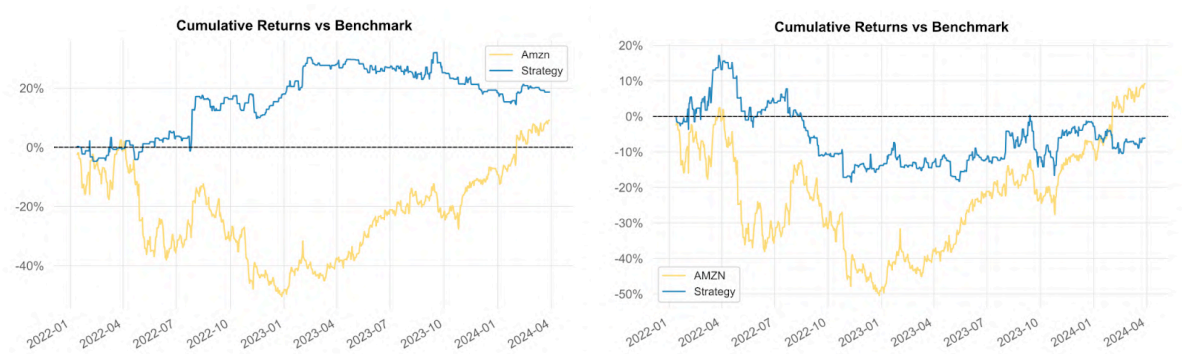


Fig. 7.11 AMZN Performance FinBERT SA (left) vs LLM_KG (right)

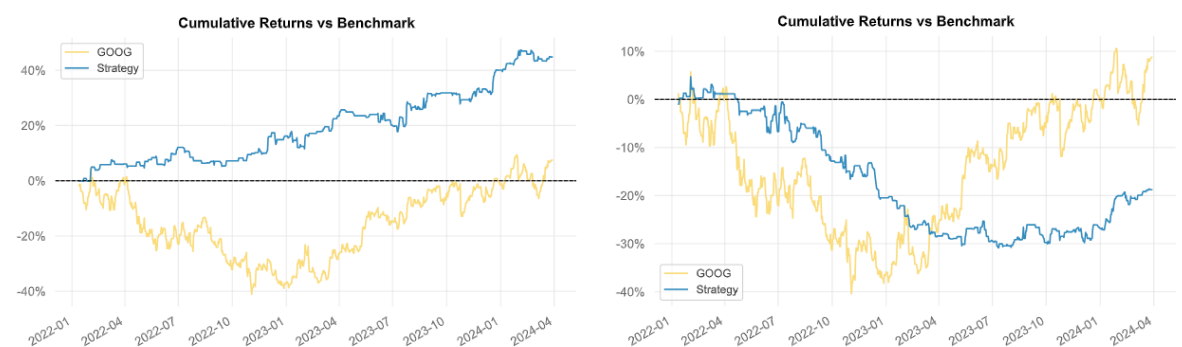


Fig. 7.12 GOOG Performance FinBERT SA (left) vs LLM_KG (right)

C. Limitations

The project heavily depends on the news provided and this is a major bottleneck to this research as the performance depends solely on the news. The news must be verified for its accuracy as well as its popularity must be taken into consideration which this project does not consider as of now. LLM Reasoning as well as triplet generation depend on the quality of the news provided and the news sources taken into consideration, which in-turn affect the decision making process of whether to buy or sell a certain stock.

The outputs of the project are also limited to google's Gemini Pro LLM. Any update, fine-tuning of the LLM or using a different LLM may vastly change the results. LLMs in general are subject to a black-box nature and hence hallucination. Adding financial market news as context helps resolve hallucination to some degree but as mentioned above consistency with LLMs is a challenging drawback.

Moreover, Stock market prediction in general is inherently unpredictable due to the complex interplay of countless variables, including economic indicators, geopolitical events, and investor psychology. The market's inherent volatility amplifies this unpredictability, making it challenging to accurately forecast future trends. The project only accounts for news sources as of now, and does not take into consideration other variables. An integration with hard data, company fundamentals and different sources of news with the LLM KG Strategy may result in further improvements.

8. SUMMARY

In conclusion, the project addresses the formidable challenge of accurately predicting stock price movements in financial markets through the integration of Knowledge Graphs (KGs), Large Language Model (LLM) reasoning, and sentiment analysis of news and digital forums. By leveraging these innovative technologies, the project has demonstrated significant advancements in enhancing predictive accuracy and providing valuable insights for investors and financial analysts. The comprehensive framework developed in this project has shown promise in capturing the complex relationships and dependencies among various factors influencing stock prices, including market sentiment dynamics and fundamental indicators. Moving forward, further research and development in this area hold the potential to revolutionize stock market analysis and empower stakeholders with more informed decision-making capabilities. Ultimately, the integration of KGs, LLM reasoning, and sentiment analysis offers a holistic approach to stock price prediction, contributing to the ongoing evolution of predictive analytics in financial markets.

9. REFERENCES: (IEEE format)

Journal:

- [1] Mu, Guangyu, et al. "A Stock Price Prediction Model Based on Investor Sentiment and Optimized Deep Learning." *IEEE Access* (2023).
- [2] Ren, Yinghao, Fangqing Liao, and Yongjing Gong. "Impact of news on the trend of stock price change: An analysis based on the deep bidirectional LSTM model." *Procedia Computer Science* 174 (2020): 128-140.
- [3] Fazlija, Bledar, and Pedro Harder. "Using financial news sentiment for stock price direction prediction." *Mathematics* 10.13 (2022): 2156.
- [4] Villamil, Luis, et al. "Improved stock price movement classification using news articles based on embeddings and label smoothing." *arXiv preprint arXiv:2301.10458* (2023).
- [5] Yao, Shunyu, et al. "Tree of thoughts: Deliberate problem solving with large language models." *arXiv preprint arXiv:2305.10601* (2023).
- [6] Asai, Akari, et al. "Self-rag: Learning to retrieve, generate, and critique through self-reflection." *arXiv preprint arXiv:2310.11511* (2023).
- [7] Ho, Namgyu, Laura Schmid, and Se-Young Yun. "Large language models are reasoning teachers." *arXiv preprint arXiv:2212.10071* (2022).
- [8] Yin, Da, et al. "Lumos: Learning agents with unified data, modular design, and open-source llms." *arXiv preprint arXiv:2311.05657* (2023).
- [9] Medya, Sourav, et al. "An exploratory study of stock price movements from earnings calls." *Companion Proceedings of the Web Conference 2022*. 2022.
- [10] Rebedea, Traian, et al. "Nemo guardrails: A toolkit for controllable and safe llm applications with programmable rails." *arXiv preprint arXiv:2310.10501* (2023).

Weblinks:

- [11] <https://lumibot.lumiwealth.com> (accessed on February 2024)
- [12] <https://docs.alpaca.markets/docs/getting-started> (accessed on February 2024)
- [13] <https://developer.yahoo.com/api/> (accessed on February 2024)
- [14] <https://seekingalpha.com/earnings/earnings-call-transcripts> (accessed on February 2024)

APPENDIX A - SAMPLE CODE

retrieve_stock_news.py

```
import pandas as pd
import requests
from datetime import datetime, timedelta
from config import MARKETAUX_KEY

api_token = MARKETAUX_KEY
symbols = 'NVDA'

start_date = datetime.strptime('2024-01-01', '%Y-%m-%d')
num_days = 95

def get_news(api_token, symbols, publish_date, filter_entities=True,
             language='en'):
    url =
f"https://api.marketaux.com/v1/news/all?symbols={symbols}&filter_entities=
{filter_entities}&published_on={publish_date}&language={language}&api_token={api_token}"
    response = requests.get(url)

    if response.status_code == 200:
        news_data = response.json()
        highlights = []
        for item in news_data['data']:
            for entity in item['entities']:
                if entity['symbol'] == symbols:
                    for highlight in entity['highlights']:
                        highlights.append(highlight['highlight'].replace('<em>',
                            '').replace('</em>', ''))

        return highlights
    else:
        print(f"Error: {response.status_code}")
        return None

try:
```

```

    df = pd.read_csv('nvidia_news.csv')
except FileNotFoundError:
    df = pd.DataFrame(columns=['date', 'news'])
for i in range(0, num_days):
    publish_date = (start_date + timedelta(days=i)).strftime('%Y-%m-%d')
    summary = str(get_news(api_token, symbols, publish_date))
    new_df = pd.DataFrame({"date": [publish_date], "news": [summary]})
    df = pd.concat([df, new_df], ignore_index=True)
df.to_csv('nvidia_news.csv', index=False)

```

sentiment_analysis.py

```

import os
import pandas as pd
from transformers import AutoTokenizer, AutoModelForSequenceClassification
import torch
device = "cuda:0" if torch.cuda.is_available() else "cpu"

tokenizer = AutoTokenizer.from_pretrained("ProsusAI/finbert")
model =
AutoModelForSequenceClassification.from_pretrained("ProsusAI/finbert").to(
device)
labels = ["positive", "negative", "neutral"]

def estimate_sentiment(news):
    if news:
        tokens = tokenizer(news, return_tensors="pt",
padding=True).to(device)

        result = model(tokens["input_ids"],
attention_mask=tokens["attention_mask"])[
            "logits"
        ]
        result = torch.nn.functional.softmax(torch.sum(result, 0), dim=-1)
        probability = result[torch.argmax(result)]
        sentiment = labels[torch.argmax(result)]
        return probability, sentiment
    else:
        return 0, labels[-1]

```



```

FILE_NAME = 'tesla_news'
df = pd.read_csv(FILE_NAME+'.csv')

batch_size = 10

checkpoint_file = FILE_NAME+'sentiment_checkpoint.txt'
start_index = 0
if os.path.exists(checkpoint_file):
    with open(checkpoint_file, 'r') as f:
        start_index = int(f.read())

for i in range(start_index, len(df)):
    news = df.at[i, 'news']
    prob, sentiment = estimate_sentiment(news[1:1000])
    print(news[1:10], prob.item(), sentiment)
    df.at[i, 'probability'] = prob.item()
    df.at[i, 'sentiment'] = sentiment

    if (i + 1) % batch_size == 0:
        df[:i+1].to_csv(FILE_NAME+'_with_sentiment.csv', index=False)
        with open(checkpoint_file, 'w') as f:
            f.write(str(i + 1))
        print(f"Saved {i + 1} rows.")

df.to_csv(FILE_NAME+'_with_sentiment.csv', index=False)
print(f"Total {len(df)} rows processed and saved.")

```

LLM_reason_generation.py

```
FILENAME = 'tesla_news'

import os
import pandas as pd

import google.generativeai as genai
from config import GOOGLE_API_KEY
genai.configure(api_key=GOOGLE_API_KEY)
model = genai.GenerativeModel('gemini-pro')

def remove_quotes(s):
    return s.replace('"', '').replace("'", '')

def LLM_Response(news):
    try:
        system_prompt = """ You are an expert stock news analyser. With
respect to the news provided respond with a reasons for whether you should
buy or sell Tesla stock and give a confidence ranging from 0 (sell) to 9
(buy). You must ensure the following rules are followed when giving your
analysis:
        RULES:
        1. Please respond in json format: { reason: string, confidence:
integer }, where reason contains the reasons for buying or selling Tesla
stock, and confidence is a integer ranging from 0 (sell) to 9 (buy)
        2. Please ensure that the news is critically analysed before
writing the reasons for the prediction of the stock price
        3. Try to find meaningful information that may be most detrimental
to Tesla Stock price movement. This is very important!
        """

        ARTICLE = news

        response = model.generate_content(
            f'''
            SYSTEM PROMPT: {system_prompt}
            NEWS: {remove_quotes(ARTICLE)} ''',
            stream=True,
```

```

    )

    output = ''
    for chunk in response:
        output += chunk.text

    return output
except Exception as e:
    print(f"An error occurred: {e}\nfor news: {news}")
    return '{"reason": "error", "confidence": 2}'

df = pd.read_csv('tesla_news.csv')

batch_size = 5

checkpoint_file = FILENAME+'_LLM_Reason_checkpoint.txt'
start_index = 0
if os.path.exists(checkpoint_file):
    with open(checkpoint_file, 'r') as f:
        start_index = int(f.read())

for i in range(start_index, len(df)):
    df.at[i, 'LLM_output'] = LLM_Response(df.at[i, 'news'])
    df_to_save = df.drop('news', axis=1)

    if (i + 1) % batch_size == 0:
        df_to_save.to_csv(FILENAME+'_with_LLM_Reason.csv', index=False)

        with open(checkpoint_file, 'w') as f:
            f.write(str(i + 1))
        print(f"Saved {i + 1} rows.")

df.drop('news', axis=1, inplace=True)
df.to_csv(FILENAME+'_with_LLM_Reason.csv', index=False)
print(f"Total {len(df)} rows processed and saved.")

```

KG_triplet_generation.py

```
FILENAME = 'nvidia_news'

import google.generativeai as genai
from config import GOOGLE_API_KEY

genai.configure(api_key=GOOGLE_API_KEY)
model = genai.GenerativeModel('gemini-pro')

import os
import pandas as pd

def remove_quotes(s):
    return s.replace('"', '').replace("'", '')

def LLM_Triplet(news):
    try:
        system_prompt = """ You are an expert in financial news analysis
and knowledge graph generation. You list out important cases or incidents
from the provided news that may be detrimental to Nvidia stock price
movement and generate very specific knowledge-graph triplets based on the
news provided. You must ensure the following rules are followed when
generating the output
        RULES:
        1. Please respond in json format: { impact_cases[...], triplets[
{head, relation, tail}, ... ]}, where impact_cases is an array of strings
containing important snippets from the provided news, and triplets is an
array containing triplets of form {head, relation, tail} and nothing else
as it needs to be used directly as json.
        2. Please generate exactly 5 triplets from the news provided below
        3. Please ensure that the news triplets only contain information
given in the news text and not any additional or irrelevant details.
        4. All elements head relation and tail must contain information.
None of the elements can be None or empty
        5. Please ensure to find meaningful information that may be most
detrimental to Nvidia Stock price movement. This is very important!
        6. Please ensure that the triplets must make sense when read as a
sentence
```

```

"""
response = model.generate_content(
    f'''
SYSTEM PROMPT: {system_prompt}
NEWS: {remove_quotes(news)[:1500]}''' ,
    stream=True)

output = ''
for chunk in response:
    output += chunk.text

return output
except Exception as e:
    print(f"An error occurred: {e}\nfor news: {news}")
    return '{error: news blocked}'

df = pd.read_csv(FILENAME+'.csv')
batch_size = 5

checkpoint_file = FILENAME+'_kg_checkpoint.txt'
start_index = 0
if os.path.exists(checkpoint_file):
    with open(checkpoint_file, 'r') as f:
        start_index = int(f.read())

for i in range(start_index, len(df)):
    df.at[i, 'LLM_Triplet'] = LLM_Triplet(df.at[i, 'news'])

    if (i + 1) % batch_size == 0:
        df[:i+1].to_csv(FILENAME+'_with_triplets.csv', index=False)

        with open(checkpoint_file, 'w') as f:
            f.write(str(i + 1))

        print(f"Saved {i + 1} rows.")

df.to_csv(FILENAME+'_with_triplets.csv', index=False)
print(f"Total {len(df)} rows processed and saved.")

```

```

with open(checkpoint_file, 'w') as f:
    f.write(str(i + 1))

print(f"Saved {i + 1} rows.")

KG_triplet_extraction.py

FILENAME = 'tesla_news'

import pandas as pd
import json

df = pd.read_csv(FILENAME+'_with_triplets.csv')
new_rows = []
warning_flag = False

for index, row in df.iterrows():

    try:
        output = json.loads(row['LLM_Triplet'])
        if 'triplets' not in output and 'error' not in output:
            print(f"Row {index + 1} Date - {row['date']}: LLM_Triplet
doesn't contain 'triplets' or 'error'.")
            warning_flag = True
        except json.JSONDecodeError:
            print(f"Date - {row['date']}: LLM_Triplet is not in valid JSON
format.")
            warning_flag = True

    triplets = json.loads(row['LLM_Triplet'])

    if 'triplets' in triplets:
        for triplet in triplets['triplets']:
            if not triplet['head']:
                print(f"Warning: 'head' not found for DATE {row['date']}")
                warning_flag = True
            if not triplet['tail']:
                print(f"Warning: 'tail' not found for DATE {row['date']}")
                warning_flag = True
            if not triplet['relation']:

```

```

        print(f"Warning: 'relation' not found for DATE
{row['date']}")
        warning_flag = True
    else:
        if 'error' not in triplets:
            print(f"Warning: 'triplets' key not found for DATE
{row['date']}")
            print(f"{row['date']}: {row['LLM_Triplet']}")
            print("\n")
            warning_flag = True

if warning_flag == False:
    for index, row in df.iterrows():
        triplets = json.loads(row['LLM_Triplet'])

        if 'triplets' in triplets:
            for triplet in triplets['triplets']:
                head = str(triplet['head'])
                relation = str(triplet['relation'])
                tail = str(triplet['tail'])
                sentence = head + " " + relation + " " + tail
                new_rows.append({'date': row['date'], 'triplet': triplet,
'head':head, 'relation':relation, 'tail':tail, 'sentence':sentence})
            else:
                if 'error' in triplets:
                    new_rows.append({'date': row['date'], 'triplet': 'error:
news blocked', 'head': '-', 'relation': '-', 'tail': '-'})

    new_df = pd.DataFrame(new_rows)

    new_df.to_csv(FILENAME+'_triplets_for_all_dates.csv', index=False)

```

KG_triplet_extraction.py

```
FILENAME = 'tesla_news'

import pandas as pd
import json

df = pd.read_csv(FILENAME+'_with_triplets.csv')

new_rows = []

warning_flag = False

for index, row in df.iterrows():

    try:

        output = json.loads(row['LLM_Triplet'])

        if 'triplets' not in output and 'error' not in output:

            print(f"Row {index + 1} Date - {row['date']}: LLM_Triplet  
doesn't contain 'triplets' or 'error'.")

            warning_flag = True

    except json.JSONDecodeError:

        print(f"Date - {row['date']}: LLM_Triplet is not in valid JSON  
format.")

        warning_flag = True
```



```

triplets = json.loads(row['LLM_Triplet'])

if 'triplets' in triplets:

    for triplet in triplets['triplets']:

        if not triplet['head']:

            print(f"Warning: 'head' not found for DATE {row['date']}")

            warning_flag = True

        if not triplet['tail']:

            print(f"Warning: 'tail' not found for DATE {row['date']}")

            warning_flag = True

        if not triplet['relation']:

            print(f"Warning: 'relation' not found for DATE
{row['date']}")

            warning_flag = True

    else:

        if 'error' not in triplets:

            print(f"Warning: 'triplets' key not found for DATE
{row['date']}")

            print(f"{row['date']}: {row['LLM_Triplet']}")

            print("\n")

            warning_flag = True

if warning_flag == False:

```

```

for index, row in df.iterrows():

    triplets = json.loads(row['LLM_Triplet'])

    if 'triplets' in triplets:

        for triplet in triplets['triplets']:

            head = str(triplet['head'])

            relation = str(triplet['relation'])

            tail = str(triplet['tail'])

            sentence = head + " " + relation + " " + tail

            new_rows.append({'date': row['date'], 'triplet': triplet,
                              'head':head, 'relation':relation, 'tail':tail, 'sentence':sentence})

        else:

            if 'error' in triplets:

                new_rows.append({'date': row['date'], 'triplet': 'error:
news blocked', 'head': '-', 'relation': '-', 'tail': '-'})

    new_df = pd.DataFrame(new_rows)

    new_df.to_csv(FILENAME+'_triplets_for_all_dates.csv', index=False)

```

KG_triplet_embedding.py

```
FILENAME = 'tesla_news'

from config import GOOGLE_API_KEY

import google.generativeai as genai

import pandas as pd

import pickle

genai.configure(api_key=GOOGLE_API_KEY)

model = 'models/embedding-001'

def embed_fn(text):

    embedding = genai.embed_content(model=model, content=text,
task_type="retrieval_document")["embedding"]

    return embedding

# Read the DataFrame from CSV

df = pd.read_csv(FILENAME+'_triplets_for_all_dates.csv')

# Iterate over DataFrame, generate embeddings, and store in a new column

err_count = 0

df['embeddings'] = [None] * len(df)
```

```
for i in range(0, len(df)):

    try:

        df.at[i, 'embeddings'] = embed_fn(df.at[i, 'sentence'])

        print(f"Embedded {i + 1} rows.")

    except Exception as e:

        print("\nERROR: ", e, "\nROW: ", i+1, "\nDATE: ", df.at[i, 'date'],
              "\n")

        err_count += 1

# Save the DataFrame with pickle

with open(FILENAME+'_embeddings.pkl', 'wb') as f:

    pickle.dump(df, f)

print(f"Total {len(df)} rows processed and saved.")

print("No. of errors", err_count)
```

LLM_KG_Reasoning.py

```
FILENAME = 'nvidia_news'

import time
import numpy as np
import pandas as pd
import pickle
from datetime import datetime, timedelta

import google.generativeai as genai
from config import GOOGLE_API_KEY
genai.configure(api_key=GOOGLE_API_KEY)

def find_top_k_passages_by_week(date, query, dataframe, k=5):

    model = 'models/embedding-001'

    query_date = datetime.strptime(date, '%Y-%m-%d')
    start_date = query_date - timedelta(days=7)
    end_date = query_date - timedelta(days=1)

    dataframe['date'] = pd.to_datetime(dataframe['date'],
format='%d-%m-%Y')
    dataframe_filtered = dataframe[(dataframe['date'] > start_date) &
(dataframe['date'] < end_date)]
    query_embedding = genai.embed_content(model=model, content=query,
task_type="retrieval_query")
    dot_products = np.dot(np.stack(dataframe_filtered['embeddings']),
query_embedding["embedding"])

    dataframe_filtered['score'] = dot_products
    dataframe_sorted = dataframe_filtered.sort_values(by='score',
ascending=False)
    dataframe_sorted = dataframe_sorted.head(k)
    dataframe_sorted = dataframe_sorted.sort_values(by='date',
ascending=True)

    output_strings = []
    for idx, row in dataframe_sorted.iterrows():
        output_strings.append(f"{row['date'].strftime('%Y-%m-%d')}
{row['sentence']}")

    return '\n'.join(output_strings)
```

```

def find_top_k_passages(query, dataframe, k=5):

    model = 'models/embedding-001'

    dataframe['date'] = pd.to_datetime(dataframe['date'],
format='%d-%m-%Y')
    query_embedding = genai.embed_content(model=model, content=query,
task_type="retrieval_query")
    dot_products = np.dot(np.stack(dataframe['embeddings']),
query_embedding["embedding"])

    dataframe['score'] = dot_products
    dataframe_sorted = dataframe.sort_values(by='score', ascending=False)
    dataframe_sorted = dataframe_sorted.head(k)
    dataframe_sorted = dataframe_sorted.sort_values(by='date',
ascending=True)

    output_strings = []
    for idx, row in dataframe_sorted.iterrows():
        output_strings.append(f"{row['date'].strftime('%Y-%m-%d')}
{row['sentence']}")

    return '\n'.join(output_strings)

def remove_quotes(s):
    return s.replace('"', "").replace("'", '')

def LLM_Response(news, week_context, general_context):

    model = genai.GenerativeModel('gemini-pro')

    try:
        system_prompt = """ You are an expert stock news analyser. With
respect to the context news and current news provided respond with a
reasons for whether you should buy or sell Nvidia stock and give a
confidence ranging from 0 (sell) to 9 (buy). You must ensure the following
rules are followed when giving your analysis:
        RULES:
        1. Please respond in json format: { reason: string, confidence:
integer }, where reason contains the reasons for buying or selling Nvidia
stock, and confidence is a integer ranging from 0 (sell) to 9 (buy)
        2. Please ensure that the NEWS is critically analysed before
writing the reasons for the prediction of the stock price

```

3. The LAST WEEK CONTEXT NEWS is relevant news titles from the past week along with dates. Ensure that this LAST WEEK CONTEXT NEWS is also critically analysed before writing the reasons for the prediction of the stock

4. The GENERAL CONTEXT is relevant news titles or information in general that is relevant to the provided news along with the dates. Ensure that this GENERAL CONTEXT NEWS is also critically analysed before writing the reasons for the prediction of the stock

5. Try to find meaningful information that may be most detrimental to Nvidia Stock price movement. This is very important!

```
"""

ARTICLE = news

response = model.generate_content(
    f'''
        SYSTEM PROMPT: {system_prompt}

        GENERAL CONTEXT NEWS: {general_context}

        LAST WEEK CONTEXT NEWS: {week_context}

        NEWS: {remove_quotes(ARTICLE)}''' ,
    stream=True,
)

output = ''
for chunk in response:
    output += chunk.text

return output
except Exception as e:
    print(f"An error occurred: {e}\nfor news: {news}")
    return '{"reason": "error", "confidence": 1}' # Return
ONLY_HIGH if an error occurs

# Load the CSV file into a pandas DataFrame
df = pd.read_csv(FILENAME+'.csv')

with open('EMBEDDINGS.pkl', 'rb') as f:
    triplet_df = pickle.load(f)

# Define the batch size (number of rows to process before saving)
batch_size = 5

# Process the DataFrame in batches and periodically save the CSV file
```

```

for i in range(0, len(df)):
    if i < 9:
        continue

    time.sleep(1)

    try:
        general_context = find_top_k_passages(df.at[i, 'news'], triplet_df,
10)
    except:
        general_context = "No context"

    try:
        week_context = find_top_k_passages_by_week(df.at[i, 'date'], df.at[i,
'news'], triplet_df, 10)
    except:
        week_context = "No context"

    df.at[i, 'LLM_output'] = LLM_Response(df.at[i, 'news'], week_context,
general_context)

    print("DATE: ", df.at[i, 'date'])
    print(df.at[i, 'LLM_output'], "\n\n")

    # Drop the 'news' column before saving
    df_to_save = df.drop('news', axis=1) # axis=1 specifies dropping a
column

    # Check if it's time to save the CSV file
    if (i + 1) % batch_size == 0:
        df_to_save.to_csv(FILENAME+'_with_LLM_KG.csv', index=False)
        print(f"Saved {i + 1} rows.")

    # Save the remaining rows (without 'news' column)
    df.drop('news', axis=1, inplace=True) # Drop directly modifies the
DataFrame
    df.to_csv(FILENAME+'_with_LLM_KG.csv', index=False)
    print(f"Total {len(df)} rows processed and saved.")

```


confidence_extraction.py

```
# CHANGE FILENAME TO LLM REASON OUTPUT FILE
FILENAME = 'tesla_news_2'

import pandas as pd
import json

def check_json_format(df):
    for index, row in df.iterrows():
        try:
            output = json.loads(row['LLM_output'])
            if 'confidence' in output:
                df.at[index, 'confidence'] = output['confidence']
            else:
                print(f"Row {index + 1} Date - {row['date']}: LLM_output
doesn't contain 'confidence'.")
                if 'reason' not in output and 'error' not in output:
                    print(f"Row {index + 1} Date - {row['date']}: LLM_output
doesn't contain 'reason' or 'error'.")
                except json.JSONDecodeError:
                    print(f"Date - {row['date']}: LLM_output is not in valid JSON
format.")

        df.to_csv(FILENAME + '_with_confidence.csv', index=False)

# Call the function to check JSON format
df = pd.read_csv('tesla_news_2_with_LLM_Reason.csv')

df['confidence'] = None
df['LLM_output'] = df['LLM_output'].str.replace("`", '')
df['LLM_output'] = df['LLM_output'].str.replace("json", '')
df['LLM_output'] = df['LLM_output'].str.replace("JSON", '')
df['LLM_output'] = df['LLM_output'].str.replace("reason: error,
confidence: 2", '"reason": "error", "confidence": 2')

check_json_format(df)
```

check_json.py

```
FILENAME = 'tesla_news'

import pandas as pd
import json

def check_json_format(df):
    for index, row in df.iterrows():
        try:
            output = json.loads(row['LLM_output'])
            if 'confidence' not in output and 'error' not in output:
                print(f"Row {index + 1} Date - {row['date']}: LLM_output
doesn't contain 'confidence' or 'error'.")
            except json.JSONDecodeError:
                print(f"Date - {row['date']}: LLM_output is not in valid JSON
format.")
        try:
            output = json.loads(row['LLM_Triplet'])
            if 'triplets' not in output and 'error' not in output:
                print(f"Row {index + 1} Date - {row['date']}: LLM_Triplet
doesn't contain 'confidence' or 'error'.")
            except json.JSONDecodeError:
                print(f"Date - {row['date']}: LLM_Triplet is not in valid JSON
format.")

df = pd.read_csv(FILENAME+'_with_LLM_Reason.csv')

df['LLM_output'] = df['LLM_output'].str.replace("`", '')
df['LLM_output'] = df['LLM_output'].str.replace("json", '')
df['LLM_output'] = df['LLM_output'].str.replace("JSON", '')
df['LLM_output'] = df['LLM_output'].str.replace("reason: error,
confidence: 2", '"reason": "error", "confidence": 2')

check_json_format(df)

check_json_format(df)
```

sentiment_tradebot.py

```
FILENAME = 'tesla_news'
stock = "TSLA"

from datetime import datetime

from lumibot.backtesting import YahooDataBacktesting
from lumibot.strategies import Strategy
import pandas as pd

target_file = FILENAME+'_with_sentiment.csv'
df = pd.read_csv(target_file)

class SentimentAnalysis(Strategy):

    def initialize(self, symbol:str=stock, cash_at_risk:float=.4):
        self.symbol = symbol
        self.sleeptime = "24H"
        self.last_trade = None
        self.cash_at_risk = cash_at_risk

    def position_sizing(self):
        cash = self.get_cash()
        last_price = self.get_last_price(self.symbol)
        if cash < 1000:
            self.cash_at_risk=0.9
        else:
            self.cash_at_risk=0.4
        quantity = round(cash * self.cash_at_risk / last_price,0)
        return cash, last_price, quantity

    def get_sentiment(self):
        today = self.get_datetime().strftime('%Y-%m-%d')

        print("\n", str(today))
        filtered_row = df[df['date'] == str(today)]
        if not filtered_row.empty:
            probability = filtered_row['probability'].values[0]
            sentiment = filtered_row['sentiment'].values[0]
            print("Probability:", probability)
            print("Sentiment:", sentiment)
            return probability, sentiment
        else:
            print("No data found for the given date.")
```

```

        return 0, "neutral"

def on_trading_iteration(self):
    cash, last_price, quantity = self.position_sizing()
    probability, sentiment = self.get_sentiment()

    if cash > last_price:
        if sentiment == "positive" and probability > .7:
            if self.last_trade == "sell":
                self.sell_all()
            order = self.create_order(
                self.symbol,
                quantity,
                "buy",
                type="bracket",
                take_profit_price=last_price*1.5,
                stop_loss_price=last_price*.85
            )
            self.submit_order(order)
            self.last_trade = "buy"
        elif sentiment == "negative" and probability > .7:
            if self.last_trade == "buy":
                self.sell_all()
            order = self.create_order(
                self.symbol,
                quantity,
                "sell",
                type="bracket",
                take_profit_price=last_price*1.5,
                stop_loss_price=last_price*.85
            )
            self.submit_order(order)
            self.last_trade = "sell"

backtesting_start = datetime(2022, 1, 11)
backtesting_end = datetime(2024, 3, 31)

# Run the backtest
SentimentAnalysis.backtest(
    YahooDataBacktesting,
    backtesting_start,
    backtesting_end,
    benchmark_asset=stock
)

```

LLM_reason_tradebot.py

```
FILENAME = "tesla_news"
stock = "TSLA"

from datetime import datetime, timedelta

from lumibot.backtesting import YahooDataBacktesting
from lumibot.strategies import Strategy

import pandas as pd
df = pd.read_csv(FILENAME+'_with_confidence.csv')

class LLM_Reasoning(Strategy):

    def initialize(self, symbol:str=stock, cash_at_risk:float=.4):
        self.symbol = symbol
        self.sleeptime = "24H"
        self.last_trade = None
        self.cash_at_risk = cash_at_risk

    def position_sizing(self):
        cash = self.get_cash()
        last_price = self.get_last_price(self.symbol)
        if cash < 1000:
            self.cash_at_risk=0.8
        else:
            self.cash_at_risk=0.4
        quantity = round(cash * self.cash_at_risk / last_price,0)
        return cash, last_price, quantity

    def get_sentiment(self):
        today = self.get_datetime()
        yesterday = today - timedelta(days=1)
        filtered_row = df[df['date'] == str(today.strftime('%Y-%m-%d'))]
        if not filtered_row.empty:
            confidence = filtered_row['confidence'].values[0]
            print(" confidence:", confidence)
            return confidence
        else:
            print("No data found for the given date.")
            return 5

    def on_trading_iteration(self):
        cash, last_price, quantity = self.position_sizing()
```

```

confidence = self.get_sentiment()

if cash > last_price:
    if confidence > 7:
        if self.last_trade == "sell":
            self.sell_all()
        order = self.create_order(
            self.symbol,
            quantity,
            "buy",
            type="bracket",
            take_profit_price=last_price*1.5,
            stop_loss_price=last_price*.85
        )
        self.submit_order(order)
        self.last_trade = "buy"
    elif confidence < 3:
        if self.last_trade == "buy":
            self.sell_all()
        order = self.create_order(
            self.symbol,
            quantity,
            "sell",
            type="bracket",
            take_profit_price=last_price*1.5,
            stop_loss_price=last_price*.85
        )
        self.submit_order(order)
        self.last_trade = "sell"

backtesting_start = datetime(2022, 1, 11)
backtesting_end = datetime(2024, 3, 31)

LLM_Reasoning.backtest(
    YahooDataBacktesting,
    backtesting_start,
    backtesting_end,
    benchmark_asset=stock
)

```

LLM_KG_reason_tradebot.py

```
FILENAME = "tesla_news"
stock = "TSLA"

from datetime import datetime, timedelta

from lumibot.backtesting import YahooDataBacktesting
from lumibot.strategies import Strategy

import pandas as pd
df = pd.read_csv(FILENAME+'_with_KG_confidence.csv')

class LLM_KG_Reasoning(Strategy):

    def initialize(self, symbol:str=stock, cash_at_risk:float=.4):
        self.symbol = symbol
        self.sleeptime = "24H"
        self.last_trade = None
        self.cash_at_risk = cash_at_risk

    def position_sizing(self):
        cash = self.get_cash()
        last_price = self.get_last_price(self.symbol)
        if cash < 1000:
            self.cash_at_risk=0.8
        else:
            self.cash_at_risk=0.4
        quantity = round(cash * self.cash_at_risk / last_price,0)
        return cash, last_price, quantity

    def get_sentiment(self):
        today = self.get_datetime()
        yesterday = today - timedelta(days=1)
        filtered_row = df[df['date'] == str(today.strftime('%Y-%m-%d'))]
        if not filtered_row.empty:
            confidence = filtered_row['confidence'].values[0]
            print(" confidence:", confidence)
            return confidence
        else:
            print("No data found for the given date.")
            return 5

    def on_trading_iteration(self):
        cash, last_price, quantity = self.position_sizing()
```

```

confidence = self.get_sentiment()

if cash > last_price:
    if confidence > 7:
        if self.last_trade == "sell":
            self.sell_all()
        order = self.create_order(
            self.symbol,
            quantity,
            "buy",
            type="bracket",
            take_profit_price=last_price*1.5,
            stop_loss_price=last_price*.85
        )
        self.submit_order(order)
        self.last_trade = "buy"
    elif confidence < 3:
        if self.last_trade == "buy":
            self.sell_all()
        order = self.create_order(
            self.symbol,
            quantity,
            "sell",
            type="bracket",
            take_profit_price=last_price*1.5,
            stop_loss_price=last_price*.85
        )
        self.submit_order(order)
        self.last_trade = "sell"

backtesting_start = datetime(2022, 1, 11)
backtesting_end = datetime(2024, 3, 31)

LLM_KG_Reasoning.backtest(
    YahooDataBacktesting,
    backtesting_start,
    backtesting_end,
    benchmark_asset=stock
)

```