Anti-Money Laundering Analysis using Large Language Models

Kanya Krishi
KXK220008
M.S Computer Science
University of Texas at Dallas

Kishorekumar Suresh
KXS220007
M.S Computer Science
University of Texas at Dallas

***Abstract*—In the current financial landscape, combating fraud is of utmost importance, as fraudsters continually evolve their tactics in financial crimes. This study delves into the application of Large Language Models (LLMs) to detect fraudulent transactions. Our aim is to leverage LLMs for identifying questionable transactions, by using the IBM Anti-Money Laundering Dataset. The training of the LLM involves extensive datasets encompassing transactional information, including bank account movements and payment patterns. This paper will outline the approach adopted, the challenges encountered, and the transformative potential that LLMs hold in advancing the field of financial fraud detection.**

***Index Terms*—anti-money laundering, LLMs, accuracy score**

I. Introduction

The introduction of digital technologies and the constantly changing landscape of financial crimes are causing a rapid transition in the financial industry. Traditional fraud detection techniques are finding it harder and harder to keep up with the sophistication of fraudulent activity. In order to combat today's financial fraud, new approaches are needed. Using Large Language Models (LLMs), which have demonstrated considerable promise in a number of artificial intelligence and data analysis disciplines, is one such promising approach.

In this study, we focus on harnessing the power of LLMs to detect and prevent fraudulent transactions in the banking sector. The core objective is to employ these models to identify transactions that exhibit characteristics of potential fraud. To achieve this, we leverage the IBM Anti-Money Laundering Dataset, a comprehensive collection of transactional data that provides a realistic and challenging environment for testing and refining our models. This Dataset contains information on transaction details such as the timestamp, sender and receiver bank account, the receiving and payment currency used, the amount transferred, the payment format - Cheque, Credit Card, ACH and Reinvestment and a column for prediction - 'Is Laundering'.

The integration of LLMs in fraud detection represents a paradigm shift from traditional rule-based systems to more dynamic, data-driven approaches. By analyzing patterns in transactional data, LLMs can uncover subtle anomalies and suspicious activities. In this paper, we will be using the following LLMs for analysis:

1) GPT-3.5 Turbo (OpenAI).
2) PaLM (Google).
3) Retrieval Augmented Generation.

This research aims not only to validate the efficacy of each of these LLMs in identifying fraudulent transactions but also to compare their performance with respect to Machine learning Models like:

1) Logistic Regression.
2) K-Nearest Neighbors.
3) Random Forest Classifier.
4) Adaboost.

Furthermore, this paper will detail the methodology employed in training the LLMs, emphasizing the importance of extensive and diverse datasets in achieving high accuracy and reliability. We will also discuss the challenges encountered during the development process, and perform a comparison between all the LLMs used and the machine learning models. Finally, we will present a comprehensive analysis of our findings, highlighting the potential of LLMs and the limitations of LLMs for this particular dataset.

II. System Design

In this section, we will dive into how LLMs and discuss the overall technical approach used for performing our analysis.

A. Large Language Models

An LLM is used for Natural Language processing and generation. They are built using deep learning techniques and several neural networks. Cause of this they are used widely in chatbots, content creation, and language translations. LLMs can:

1) Provide relevant and coherent responses or outputs by grasping the context and nuances in the text.
2) Perform task Adaptability as they can be fine-tuned for specific applications like summarization or sentiment analysis.
3) Enhance efficiency by automating language-based tasks.

Figure 1 depicts the building blocks of an LLM. Let's explore how it operates:

- **Tokenization of Input Text:** The first step in the procedure is to divide the input text into smaller chunks, or tokens. This step is essential so that we can convert

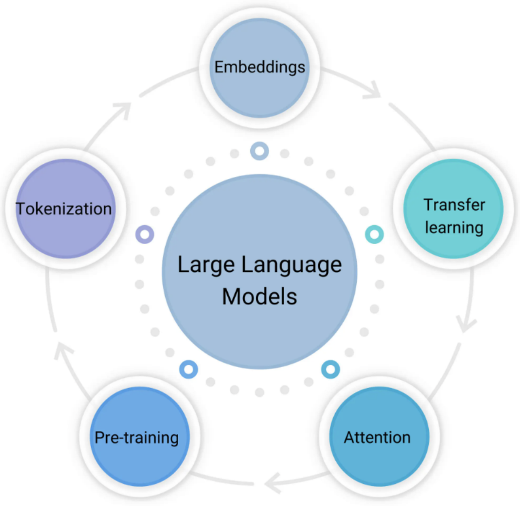

Fig. 1. Working of an LLM.

the unprocessed text into a format that the model can understand.

- **Embedding of Tokens:** Each token is then assigned a numerical value, a process referred to as embedding. These embeddings are essential for representing the tokens in a mathematical space, enabling the model to understand and manipulate them.
- **Transformer Architecture:** At the core of LLMs lies the Transformer architecture, which is specifically designed to handle sequential data. This architecture allows the model to process and generate text effectively.
- **Attention Mechanism:** A key feature of LLMs is the attention mechanism. This mechanism enables the model to focus variably on different parts of the text to understand the context and relationships between words or phrases, enhancing its comprehension capabilities.
- **Layered Processing:** The text undergoes processing through several layers within the model. Each layer contributes to uncovering more complex relationships and patterns in the text, refining the understanding and generation capabilities of the model.
- **Output Generation:** Finally, the processed information culminates in the generation of an output, which could be in the form of text completion, answer generation, or any other format depending on the task at hand.
- **Pre-training and Fine-tuning:** LLMs typically undergo an extensive pre-training phase on large datasets, which equips them with a broad understanding of language and context. For specialized tasks, further fine-tuning is conducted to adapt the model to specific requirements or datasets.

B. Preparation of Test Data

In our analysis, we utilize a test dataset consisting of 100 rows. Due to time constraints, we have opted for a smaller dataset comprising only 100 records for our analysis and comparison with other models.

The original dataset exhibited a significant class imbalance, with values for *is laundering* and *not laundering*. We had 600 thousand records for *not laundering* and only 3000 odd records for *is laundering*. To address this imbalance in our test dataset, we initially divided the data into two subsets: one containing all *is laundering* records and another containing all *not laundering* records. Subsequently, we randomly selected records from these two subsets, effectively creating a dataset of 100 records.

C. Data Preprocessing

Whether working with Large Language Models (LLMs) such as GPT-3 or GPT-4, or any other data analysis or machine learning technique, data preparation is an essential first step. They matter because:

- Well-prepared data ensures that the model learns the right patterns and relationships.
- LLMs require data in a specific format (usually textual). If the data is not in a format that the model can understand and will not be able to process it effectively, leading to poor results or errors.
- Data preparation involves cleaning the data and removing irrelevant features. This step is crucial because irrelevant or noisy data can lead to misleading patterns and decrease the model's performance.
- Well-organized and structured data allows for easier analysis and interpretation of the model's outputs. It helps in understanding the underlying patterns in the data and how they relate to the task at hand.
- Data preparation includes dealing with missing or incomplete data, which is common in real-world datasets. Proper handling of such data ensures that the model is trained on a comprehensive dataset, improving its reliability.

Each record is pre-processed into a format that is understandable by the LLM.

Transaction occurred on [Timestamp] from bank [From Bank] with originating account [Originating Account]. The transaction was directed to bank [To Bank] with receiving account [Receiving Account]. The amount received was [Amount Received] [Receiving Currency], and the amount paid was [Amount Paid] [Payment Currency]. The payment method used was [Payment Format]. This transaction is flagged as [Is Laundering: 'Laundering'/'Not Laundering']

This pre-processed text is used for further analysis.

D. Methods used for analysis

Once the pre-processed dataset was ready we performed the following analysis to each LLM.

1) **Zero-shot prompting**
   For each record in the test data, we present a prompt to the LLM, requesting it to classify the transaction as either related to money laundering or not, without providing any reference examples for the LLM to consult.

Importance - This method assesses the LLM's ability to classify transactions without any prior reference examples. It tests the model's capacity for generalized understanding and decision-making.

2) **One-shot fixed prompting** - For every entry in the test dataset, we supply a prompt to the LLM, asking it to determine whether the transaction is associated with money laundering or not, with a single reference example provided for all records for the LLM's reference.
   Importance - It helps evaluate how well the model can make binary decisions with limited reference data.
3) **One-shot variable prompting** - We provide the LLM with a prompt for each record in the test dataset. The prompt instructs the LLM to classify the transaction as either linked to money laundering or not, while also including a single random reference example each time for the LLM's consideration.
   Importance - This approach introduces variability by supplying a random reference example for each record. It assesses the LLM's adaptability and ability to make classifications when presented with different reference points for different transactions.
4) **Two-shot variable prompting** - For every entry in the test dataset, we supply a prompt to the LLM, asking it to determine whether the transaction is associated with money laundering or not. To aid in this classification, two distinct random reference examples are provided for every record, serving as points of reference for the LLM.
   Importance - Offering two random reference examples for each record adds complexity to the classification task. It evaluates the LLM's capacity to consider multiple references and make nuanced decisions, which can be crucial for handling diverse data.
5) **Four-shot fixed prompting** - For every entry in the test dataset, we supply a prompt to the LLM, asking it to determine whether the transaction is associated with money laundering or not. To aid in this classification, four distinct random reference examples are provided for every record, serving as points of reference for the LLM.
   Importance - Providing four distinct random reference examples for each record further challenges the LLM. It assesses the model's robustness in handling complex scenarios where multiple reference points are available for making decisions.

The next few sections will cover how each of the LLM - GPT-3.5-turbo(OpenAI) and PaLM(Google), performed based on the above methodologies.

III. Implementation and Results

In this section, we will discuss the implementation of the LLMs mentioned along with a comparison of their performance results.

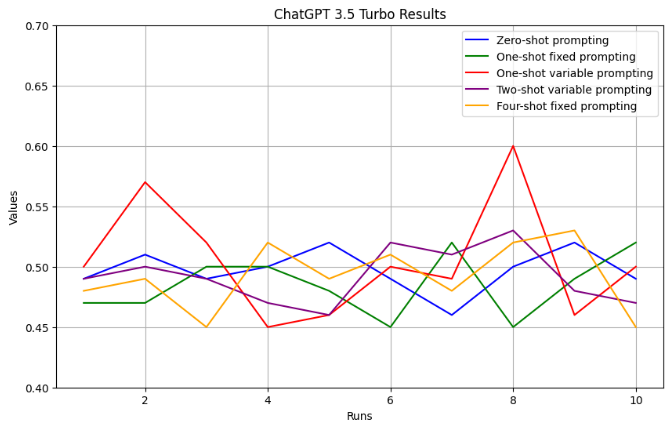

Fig. 2. Accuracy results while using GPT-3.5

A. *GPT-3.5 Turbo*

GPT-3.5-turbo is designed for scalable operations, making it more suitable for commercial use where handling numerous requests at once is crucial. It enables companies to incorporate sophisticated AI language features without major performance constraints. This is a paid API that is used.

We conducted ten trials on the dataset, with the accuracy results displayed in Figure 2. This approach aimed to calculate the average accuracy across the ten trials, ensuring that our findings were not based on a single execution.

The code in Listing 1 was used to execute Zero-shot prompting in case of GPT 3.5 Turbo

```
def classify_chatgpt(text):
    response = client.completions.create(
        model="gpt-3.5-turbo-instruct",
            # Specify the engine
        prompt=f"Based on the provided
            bank transaction details,
            determine whether it should be
             classified as 'laundering' or
             'not_laundering'. Only use
            one of these labels in your
            response.\n\nHere are the Bank
             Transaction Details: {text}",
        max_tokens=100
    )
    return response.choices[0].text.strip
        ()
```

Listing 1. Python code for GPT 3.5

The Python code outlined in Listing 1 serves as a foundational reference, forming the basis for further methodological developments.

In the one-shot fixed prompting approach, we adapted the code to consistently use a single, static example for every prompt.

The one-shot variable prompting variation involves altering the code to incorporate a randomly selected example for each prompt.

With two-shot variable prompting, the modification includes incorporating two unique, randomly chosen examples from the dataset for each prompt. Listing 2 shows how this can be achieved.

```
    # Open file and read all lines
with open('tx.txt', 'r') as file:
    lines = file.readlines()

# Choose one line at random
random_line = random.choice(lines)

def classify_2_shot_chatgpt(text):
    line_chosen = []
    for i in range(2):
      line_chosen.append(random.choice(
         lines))
    response = client.completions.create(
        model="gpt-3.5-turbo-instruct",
        prompt=f"Your task is to classify
             bank transactions as either '
            laundering' or 'not_laundering
            '. Examine each transaction's
            details carefully and provide
            only one label as a response.
            Consider factors such as the
            transaction amount, the banks
            involved, the account numbers,
             the payment method, and any
            other relevant details. Use
            the following examples to
            guide your future predictions
            :\n1.{line_chosen[0]}.\n 2. {
            line_chosen[1]}.\n\n Now,
            based on the criteria
            established in the example,
            classify the following bank
            transaction as 'laundering' or
             'not_laundering':\n{text}",
        max_tokens=100
    )
    return response.choices[0].text.strip
       ()
```

Listing 2. Python code for Two shot variable prompting using GPT 3.5 turbo

Lastly, the four-shot fixed prompting approach involves modifying the code to include four distinct examples from the dataset for each prompt.

The accuracy figures hover around the 50% mark, indicating that the dataset may not be ideally suited for Large Language Model operations. ChatGPT emphasizes that the task of classifying a transaction as 'laundering' or 'not laundering' involves complex decision-making processes. This complexity arises from the need to analyze a multitude of factors that extend well beyond basic transactional data. Such an analysis includes, but is not limited to, a thorough review of the transaction history, which helps in understanding patterns and anomalies in transactional behavior. The account profile data, encompassing the account holder's financial background, transaction frequencies, and associated risk factors, is also critical in this context. Additionally, the consistency of transactions plays a significant role; sudden changes in transactional behavior or amounts, which might deviate from an established norm, could be indicative of suspicious activities.

In summary, the task demands a multidimensional analysis approach, integrating both quantitative and qualitative data insights, to accurately classify financial transactions.

B. PaLM

PaLM, which stands for "Pathways Language Model," is an advanced language model developed by Google. This bears resemblance to the GPT-3.5 API discussed in the preceding section. Additionally, PaLM is available as a free API.

Being a Google product, PaLM potentially integrates well with other Google services and APIs, offering a seamless experience for developers already working within the Google ecosystem. Like many advanced language models, PaLM is expected to offer support for multiple languages, making it versatile for global applications. As a free API, PaLM is accessible to a broad range of developers, from individual hobbyists to large corporations, encouraging innovation and experimentation in the field of AI and language processing.

We conducted ten trials on the dataset, with the accuracy results displayed in Figure 3. This approach aimed to calculate the average accuracy across the ten trials, ensuring that our findings were not based on a single execution.

The below-mentioned code was used to execute Zero-shot prompting using PaLM:

```
def classify_palm(text):
    completion = palm.generate_text(
    model='models/text-bison-001',
    prompt=f"Based on the provided bank
        transaction details, determine
        whether it should be classified as
         'laundering' or 'not_laundering'.
         Only use one of these labels in
        your response.\n\nHere are the
        Bank Transaction Details:  {text}"
        ,
    max_output_tokens=50,
    )
    return completion.result

def rate_limited_call(classify_func, text
    , request_counter, limit=80):
    if request_counter >= limit:
        print("Above limit")
        time.sleep(60)  # Wait for 60
            seconds
        request_counter = 0  # Reset the
            counter
```

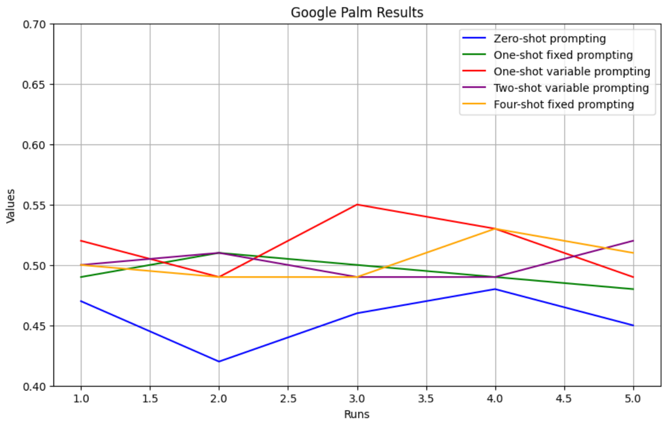

Fig. 3. Accuracy results while using PaLM

```
    return classify_palm(text),
        request_counter + 1
```

Listing 3. Python Code for Palm Classification and Rate-Limited Calls

As PaLM is a freely available API, it imposes a restriction on the number of requests permitted per minute. Should this threshold be exceeded, the program temporarily halts operations for 60 seconds before resuming its request submissions. Hence the method *rate_limited_call* is used.

The additional techniques, including 'One-shot fixed prompting', 'One-shot variable prompting', 'Two-shot variable prompting', and 'Four-shot fixed prompting', are executed similarly to the methods used in Gpt-3.5-turbo.

Figure 3 illustrates the accuracy results from ten trials using the aforementioned methods. It's evident that the accuracy remains relatively low. Large Language Models (LLMs) are known to excel in text classification tasks. For instance, they effectively sort emails into spam and non-spam categories, gauge the sentiment in reviews, and categorize documents into various genres, thanks to their sophisticated grasp of language subtleties.

However, when it comes to classification tasks that involve non-textual data, such as images or numerical datasets, traditional LLMs are not inherently equipped for these. These types of tasks are more efficiently addressed by models tailored for specific data types, like Convolutional Neural Networks (CNNs) for image recognition or dedicated machine learning algorithms for structured data.

In scenarios where the task requires analyzing both textual and non-textual data, a hybrid approach using a combination of different models may be necessary.

C. Retrival Augmented Generation

Retrival Augmented Generation (RAG) is a natural language processing and machine learning approach that brings together the powers of pre-trained language models with external knowledge retrieval. Here's how RAG works :

1) **Combining Language Models with External Knowledge:** RAG harnesses the capabilities of large language models (such as GPT-3) and connects them with an external knowledge source, generally a large-scale database or a corpus of texts. This enables the model to access and use information beyond what it was trained on.
2) **External Data Querying:** When presented with a query or a task, the RAG system creates queries to retrieve pertinent information from an external database. This stage is critical for dealing with issues or subjects that were not included in the language model's training data.
3) **Integrating Contextually:** RAG merges pertinent external data with the context of the query after it has been obtained. This integration is done in such a way that the language model may provide a response using both its pre-trained knowledge and the newly acquired information.
4) **Generating the answer response:** The language model then provides a response based on the information it has gained from its training and the external input. This response is usually more knowledgeable, correct, and up to date, especially when it comes to recent events or specialist expertise.

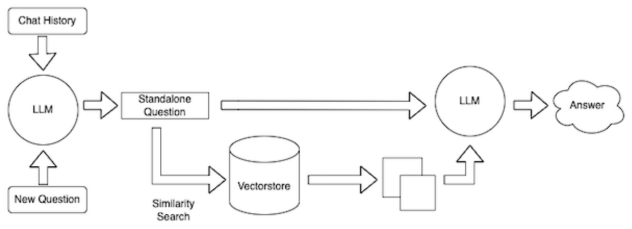

Fig. 4. Retrieval Augmented Generation Pipeline

Figure 4 shows the Pipeline of a RAG. Here are some of the important components used :

1) **Vectorstore:** This external knowledge database holds data points or documents in a structured fashion, frequently as vectors that convey the documents' semantic meaning.
2) **Similarity Search :** To identify the most pertinent documents or data points that match the query vector created from the single question, the system does a similarity search in the Vectorstore. In the vector space, methods like k-nearest neighbors (k-NN) are frequently used for this.
3) **Retrieved Information :** The LLM receives the data that was obtained from the Vectorstore after that. The solo question and the pertinent data that was obtained from the Vectorstore are now both in the LLM. It compiles this data to provide a well-informed response.

How did we use RAG?

We took 100 bank transactions, stored in a .txt file and fed this file to the RAG. This served as the external knowledge database to the RAG. We use the Langchain package to develop this application. Many components of LangChain

```
if PERSIST and os.path.exists("persist"):
    print("Reusing index...\n")
    vectorstore = Chroma(
        persist_directory="persist",
        embedding_function=
        OpenAIEmbeddings())
    index = VectorStoreIndexWrapper(
        vectorstore=vectorstore)
else:
    # TRAINING DATA
    loader = TextLoader("tx-100.txt")
    if PERSIST:
        index = VectorstoreIndexCreator(
            vectorstore_kwargs={"
            persist_directory":"persist"})
            .from_loaders([loader])
    else:
        index = VectorstoreIndexCreator()
            .from_loaders([loader])

chain = ConversationalRetrievalChain.
    from_llm(
    llm=ChatOpenAI(model="gpt-3.5-turbo")
        ,
    retriever=index.vectorstore.
        as_retriever(search_kwargs={"k":
        1}),
)
```

Fig. 5. Python Code Snippet for Retrieval Augmented Generation

are particularly made to facilitate the development of RAG applications. We then provide a query which will take into account the chat history, the external knowledge information and LLM trained knowledge and make a prediction if it is a fraudulent or non-fraudulent transaction. We used ChatGPT's 3.5 Turbo for making the predictions. A sample query or prompt that was fed to the RAG is as follows:

Based on the provided bank transaction details and the information provided to you in tx-100.txt, determine whether the transaction should be classified as 'laundering' or 'not laundering'. Do not give any explanations. Bank Transaction Details: Transaction occurred on 2022/09/07 16:39 from bank 22661 with originating account 8057301D0. The transaction was directed to bank 1 with receiving account 8057302C0. The amount received was 2186.52 US Dollar, and the amount paid was 2186.52 US Dollar. The payment method used was ACH.

D. Machine Learning Models

Machine learning models are computational algorithms that enable computers to learn from and make decisions based on data. They are a subset of artificial intelligence (AI) that focus on the development of systems capable of learning and improving from experience without being explicitly programmed for specific tasks.

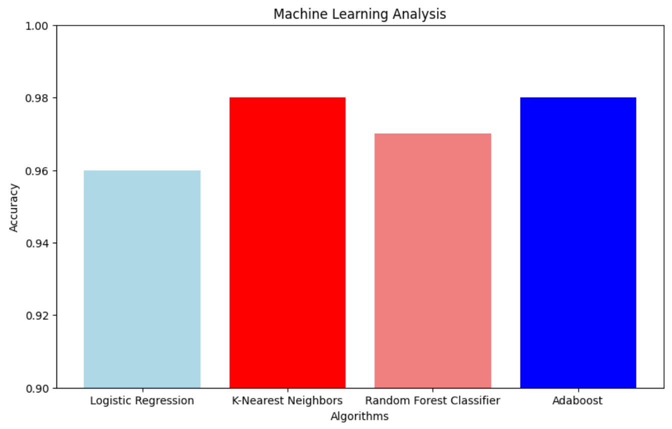

Fig. 6. Accuracy results while using Machine Learning Models

In our study we wanted to compare the performance of LLMs with that of some Machine Learning Models. We have chosen the following models for our analysis.

1) **Logistic Regression:**An approach for classification that uses a logistic function to model the probability of a binary outcome; frequently applied to binary classification problems such as positive/negative medical diagnosis.
2) **K-Nearest Neighbors (KNN):** a straightforward, instance-based learning technique that is frequently used for pattern identification and classifies new data according to the majority label of its 'k' nearest neighbors.
3) **Random Forest Classifier:** a technique for ensemble learning that offers excellent accuracy and resilience against overfitting. It works by building several decision trees during training and producing a class that is the mean of the classes of the individual trees.
4) **Adaboost:** High accuracy and resilience against overfitting are provided by this ensemble learning technique, which builds several decision trees during training and outputs the class that is the mean of the classes of the individual trees.

```
log_reg = LogisticRegression(max_iter
    =200)
knn = KNeighborsClassifier(n_neighbors=3)
rf = RandomForestClassifier(n_estimators
    =75, random_state=42)
ada_boost = AdaBoostClassifier(
    n_estimators=10, random_state=42)
```

Listing 4. Python Code for Machine Learning Models

For our machine learning models, we need to set up both a training and a testing dataset. The Python code to initialize each ML model is presented in Listing 4. After initializing a model, we train it using the training dataset and subsequently make predictions on the test dataset. The accuracy of each model is depicted in Figure 6.

IV. Limitations

Figure 7 shows an overall comparison of the performances of each model. From this, we can see that the Machine learning models have performed better. Machine learning models are particularly effective for classification and prediction tasks involving numerical data due to their inherent ability to process and analyze large quantities of structured, quantitative information. These models excel at identifying complex patterns and correlations in numerical datasets, which is a cornerstone of effective prediction and classification. Their adaptability allows them to learn from the data, adjust their parameters accordingly, and improve their accuracy over time. This is especially valuable in dynamic environments where data patterns may evolve.

Furthermore, the wide range of machine learning algorithms available, such as regression models, decision trees, and neural networks, can be tailored to suit the specific characteristics and requirements of numerical data. This versatility makes them ideal for tasks such as financial forecasting, risk assessment, and resource allocation, where precise quantitative analysis is crucial.

Additionally, the scalability of these models ensures they can handle vast volumes of data efficiently, a key requirement in many modern applications. In summary, the combination of pattern recognition, adaptability, and quantitative analysis capabilities makes machine learning models highly suited for tasks involving numerical data.

When using Large Language Models (LLMs) for specialized tasks like Anti-Money Laundering (AML), the effectiveness of these models can be limited due to their generalist nature of training. Typically, LLMs are trained on extensive datasets encompassing a broad spectrum of topics, but these datasets may lack the depth and specificity required for AML applications. AML tasks demand a nuanced understanding of complex financial patterns, legal intricacies, and compliance requirements that general LLM training might not cover. To address this gap, it's beneficial to supplement the LLM's training with additional, targeted datasets focused on AML. This supplementary training equips the LLM with more relevant examples and context, enhancing its ability to accurately recognize and analyze patterns and behaviors indicative of money laundering. This approach improves the model's precision and reliability in AML-specific applications, ensuring more accurate predictions and classifications in this critical financial security domain.

V. Future Work

In this section, we will look into the future enhancements that can be made to our work.

1) **Boost RAG Configurations with Additional Bank Transactions:** The RAG system's retrieval method may be improved to discover more intricate patterns of fraudulent conduct by integrating a bigger and more varied dataset of bank transactions. This might result in a better degree of accuracy when identifying money laundering operations.

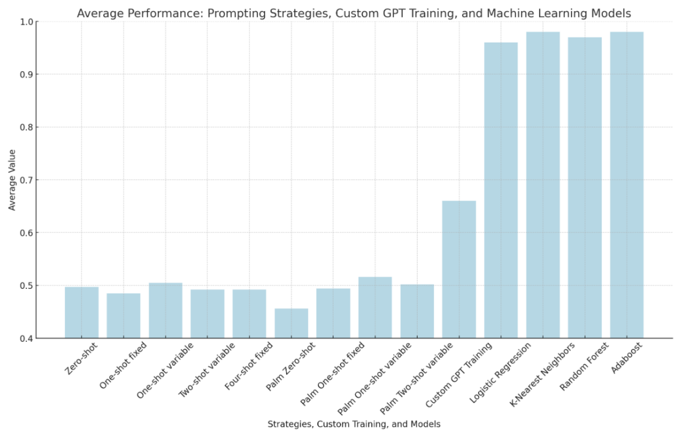

Fig. 7. Overall accuracy results.

2) **Train and Tailor a ChatGPT Model to Financial Situations:** By training a customized version of ChatGPT with bank transaction data explicitly, the model may be able to produce more accurate judgments of transaction legality by better understanding the complexities of financial communication and transactional relationships.
3) **Extend Account Information and Transaction History Data:** A better examination of financial habits will be possible with the acquisition of a more comprehensive dataset that includes specific account information and an extensive transaction history. To increase the model's capacity for prediction, this may entail obtaining private datasets or ethically scraping information from authorized sources.
4) **Create a Hybrid Tool That Combines LLM and ML Techniques:** A reliable method for identifying and forecasting fraudulent transactions with more accuracy may come from developing a hybrid system that combines massive language models for contextual comprehension with machine learning for pattern detection.
5) **Put Continuous Learning Mechanisms Into Practice:** Include continuous learning procedures such that, in order to keep the system effective against changing money laundering strategies and schemes, it changes its models on a regular basis using the most recent transaction data.
6) **Prioritize real-time analysis :** Develop a system that can instantly analyze transactions, send out alarms, and allow for proactive steps to stop money laundering before it happens.

VI. Conclusion

Because of their strong contextual awareness and linguistic sensitivity, LLMs are very successful in understanding and producing natural language, which is useful for tasks like Sentiment Analysis and Email Spam Classification.

Even with a small feature set, traditional machine learning models are highly suited to identify possible money laundering activities by evaluating structured financial data because of their capacity for pattern identification.

When comprehensive details like transaction history and account profiles are included, it may greatly enhance the effectiveness of LLMs in complicated tasks like financial fraud detection by offering a more context-rich analysis.

By constantly acquiring and integrating current, pertinent information, RAG systems can improve the quality of produced replies and, for activities that benefit from such integration of external knowledge, become marginally more successful.

Acknowledgment

We would like to thank Assistant Professor Xinda Wang for their outstanding assistance in setting up the course. Their careful planning of the curriculum gave us a thorough method for evaluating research articles, which greatly increased our comprehension of difficult subjects in the area. Their enlightening advice was crucial in enabling us to read these academic papers critically and with a strong sense of inquiry, which has been extremely beneficial to our academic development.

We also appreciate our peers whose lectures on a variety of topics, such as AI Security and the wider applications of AI, which have enhanced the quality of the learning environment. The range of subjects covered and the in-depth conversations that followed have expanded our horizons and deepened our awareness of the potential applications of AI in the security space. These conversations have fostered a collaborative learning environment that is both instructive and motivating.

In conclusion, the group project has been an incredible educational experience. Our ability to operate in a team environment has improved as a result of this experience, which is an asset in both the professional and academic spheres. In addition to having a stronger understanding of the subject, we have gained stronger interpersonal and collaborative abilities from this project that will help us in our future employment.

VII. Related Work

1) Chen, Zhiyuan, et al. ”Machine learning techniques for anti-money laundering (AML) solutions in suspicious transaction detection: a review.” Knowledge and Information Systems 57 (2018): 245-285.
2) Jullum, Martin, et al. ”Detecting money laundering transactions with machine learning.” Journal of Money Laundering Control 23.1 (2020): 173-186.
3) Kumar, Ashwini, et al. ”Analysis of classifier algorithms to detect anti-money laundering.” Computationally intelligent systems and their applications (2021): 143-152.
4) Colladon, A. F., Remondi, E. (2017). Using social network analysis to prevent money laundering. Expert Systems with Applications, 67, 49-58.

References

[1] https://www.kaggle.com/datasets/ealtman2019/ibm-transactions-for-anti-money-laundering-aml?select=HI-SmallPatterns.txt
[2] https://developers.generativeai.google/products/palm
[3] https://platform.openai.com/docs/api-reference
[4] https://huggingface.co/docs
[5] https://community.openai.com/