Department of Computer Science

MSc Data Science and Analytics

Academic Year 2022-2023

Enhancing Financial Fraud Detection: A Comparative Analysis of Large Language Models and Traditional Machine Learning and Deep Learning Approaches

Amit Shushil Kedia

A report submitted in partial fulfilment of the requirement for the degree of Master of Science

Brunel University
Department of Computer Science
Uxbridge, Middlesex UB8 3PH
United Kingdom
Tel: +44 (0) 1895 203397
Fax: +44 (0) 1895 251686

ABSTRACT

The dissertation focuses on the field of financial fraud detection, specifically examining the effectiveness of machine learning models, deep learning models, and Large Language Models (LLMs) in identifying fraudulent activities in financial reports and statements. Financial fraud poses a significant threat to the integrity of financial markets and institutions. Traditional methods of fraud detection are increasingly insufficient due to the complexity and sophistication of fraudulent schemes. This research aims to assess the effectiveness of models in detecting financial fraud, a topic of growing relevance and importance in both academics and industries. The study employed a diverse set of models, including traditional machine learning models like Logistic Regression and Random Forest, deep learning models like Support Vector Machine (SVM) and Hierarchical Attention Network (HAN), and LLMs like FinBERT and GPT-2. These models were trained and fine-tuned on a dataset comprising financial filings from various companies submitted to the SEC. The dataset was manually prepared to train the models. Performance metrics such as accuracy, precision, recall, and F1-score were used for evaluation. The research found that traditional machine learning models like Random Forest and SVM demonstrated superior performance in detecting financial fraud, while some state-of-the-art models like HAN underperformed. Among the LLMs, FinBERT showed balanced performance, making it a viable option for real-world applications.

The study contributes to the existing body of knowledge by introducing and evaluating the performance of LLMs in financial fraud detection for the first time. It also provides a comprehensive evaluation framework, thereby offering a multi-faceted view of each model's effectiveness. The key implication of this research is the critical need for careful model selection in specialized tasks like financial fraud detection. While technology has advanced significantly, the study underscores that not all models, even those considered state-of-the-art, are suitable for every application. This has both academic and practical outcomes, emphasizes the need for future research to focus on model adaptability and effectiveness in specialized domains.

ACKNOWLEDGEMENTS

I am grateful for the support I received from my supervisor, professors, and colleagues.

I certify that the work presented in the dissertation is my own unless referenced.

Signature: Amit Shushil Kedia

Date: 09 - 09 - 2023

TOTAL NUMBER OF WORDS:

Table of Contents

CHAPTER 1: INTRODUCTION

1.1 Financial Fraud

Financial fraud is a pervasive issue that has far-reaching consequences for businesses, stakeholders, and economies worldwide. According to estimates, the global financial implications of fraudulent activities over the past two decades amount to approximately $5.127 trillion, with losses increasing by 56% in the last decade (Gee, Button and Brooks, 2019). These figures only capture the direct financial losses and do not account for the indirect costs such as reputational damage to stakeholders like investors, creditors, and employees. In extreme cases, financial fraud can even lead to the bankruptcy of organizations (Dorris, 2020).

The scope of financial fraud is broad, encompassing corruption, asset misappropriation, and fraudulent financial statements. This research focuses on financial statement fraud, which involves, according to Hajek and Henriques (2017), intentional omissions or misrepresentations in financial reporting. While asset misappropriation and corruption occur more frequently, the impact of financial statement fraud is significantly more severe, with median losses amounting to $954,000 (Dorris, 2020). Such fraudulent activities have led to a decline in stock prices, delisting from stock exchanges, and even bankruptcy in some cases (Beasley, Carcello and Hermanson, 1999).

The challenge of detecting financial fraud is exacerbated by its complexity and the variety of methods employed, such as falsifying revenues or understating liabilities (Huang, Tsaih and Yu, 2014). Traditional methods of detection, often reliant on external auditors, have proven to be less effective, time-consuming, and costly (Dyck, Morse and Zingales, 2010; West and Bhattacharya, 2016). Moreover, these traditional methods sometimes face ethical challenges, such as conflicts of interest when auditors are financially tied to the firms, they are auditing, as well as practical limitations in detecting sophisticated fraud schemes (Simnett, Vanstraelen, and Chua, 2019).

The urgency to address financial fraud is not just a matter of recovering financial losses but also of restoring trust in financial systems and protecting stakeholders. The development of advanced detection systems is crucial for investors, audit firms, and regulatory bodies alike (Abbasi et al., 2012; Albrecht, Albrecht and Albrecht, 2008). Therefore, tackling financial fraud is an imperative challenge that requires immediate and effective solutions.

By synthesizing these insights, it can be said that financial fraud is a critical issue that demands prompt attention, not only for its economic implications but also for its impact on the credibility and functioning of financial systems.

1.2 Natural Language Processing

Natural Language Processing (NLP) has emerged as a transformative technology that bridges the gap between human language and computer understanding. This interdisciplinary field has seen rapid advancements, particularly in the last decade, and its applications are now widespread across multiple sectors (Chen, Xie and Tao, 2022). In finance sector, the technology is particularly adept at extracting actionable insights from vast amounts of unstructured financial text data, thereby aiding in decision-making and risk management (Osterrieder, 2023). However, despite these advancements, challenges remain. Understanding the nuances and contextual meanings in human language is still a hurdle for NLP technologies. Large Language Models (LLMs) like BERT (Bidirectional Encoder Representations from Transformers), GPT (Generative Pretrained Transformers model) have made significant progress in this area. These models are trained on vast datasets and utilize advanced machine learning algorithms, including neural networks, to interpret and generate human-like text. This enables them

to outperform traditional NLP models in tasks such as sentiment analysis, summarization, and even some forms of reasoning (Hadi et al., 2023).

It's crucial to note, however, that while LLMs offer advanced capabilities, they are not without limitations. They can sometimes generate incorrect or nonsensical responses, and their understanding of context may be shallower than human cognition (Hadi et al., 2023).

1.3 FinBERT (Large Language Model)

FinBERT, Financial Bidirectional Encoder Representations from Transformers, is a specialized language model designed to tackle Natural Language Processing (NLP) tasks in the financial domain. Developed as an extension of the BERT model, FinBERT aims to address the unique challenges posed by financial texts, which often employ specialized language and vague expressions. The model has shown significant improvements in every measured metric on current state-of-the-art results for financial sentiment analysis datasets (Huang, Wang and Yang, 2020).

While considering FinBERT's performance, it's essential to critically examine its limitations, especially when considering its application in financial fraud detection. One of the primary constraints is that FinBERT was initially designed for sentiment analysis (Araci, 2019). This focus could potentially limit its effectiveness in identifying fraudulent activities, which often require a more nuanced understanding of financial statements and transactions. Sentiment analysis and fraud detection are fundamentally different tasks; the former generally involves classifying the tone of a text, while the latter often requires the identification of subtle, deceptive practices that may not necessarily be reflected in the text's sentiment.

Moreover, FinBERT uses transfer learning methods and is pre-trained on a large corpus before being fine-tuned on domain-specific financial texts (Araci, 2019). However, in the context of financial fraud detection, the capabilities of FinBERT could be invaluable. Its advanced understanding of financial language may allow for more accurate analysis of financial reports and statements, potentially aiding in the identification of fraudulent activities. While FinBERT was initially designed for sentiment analysis, its architecture and training methodology make it a promising candidate for broader applications in financial fraud detection.

1.4 Research aim and objectives

The aim of this dissertation is to assess the effectiveness of Large Language Models (LLMs) such as FinBERT and GPT-2 (Generative Pre-Trained Transformer) in detecting fraudulent activities in financial reports and statements. The research will compare the performance of LLMs with that of traditional machine learning and deep learning models, aiming to contribute to the field of fraud detection.

The objectives of the dissertation are multifold. First, it aims to provide comprehensive background of research done on the current state-of-the-art models in detecting financial frauds from the financial reports and statements, as well as the other machine learning and deep learning models. Second, it seeks to gain a deeper understanding of Large Language Models (LLMs), their architecture, and their capabilities, with the special focus on open-source models like FinBERT Model trained on financial corpus of data, and GPT-2. The third objective is to implement and fine-tune these Large Language Models using transfer learning techniques, i.e., fine tuning to evaluate their performance in detecting financial frauds. The fourth objective is to compare the performance of these Large Language Models with traditional machine learning and NLP models in the context of financial fraud detection. Lastly, the dissertation aims to analyse the results and understand the strengths and limitations of these models, and to identify areas for future research and improvement.

RQ1. How effective are Large Language Models (LLMs) like FinBERT and GPT-2 in detecting fraudulent activities in financial reports and statements?

RQ2: Can FinBERT model outperform the existing state-of-the-art model like HAN in detecting financial fraud from the annual reports?

RQ3: How do the performances of LLMs compare with traditional machine learning models (e.g., Logistic Regression, Random Forest) and deep learning models (e.g., Hierarchical Attention Network) in detecting financial fraud?

1.5 Research approach

The research approach of this dissertation is as follows:

1. Literature Review:
An extensive review of existing research materials is conducted. This includes academic papers and journals to understand the current techniques in financial fraud detection.
2. Data Collection and Preprocessing:
Relevant financial reports and statements are collected from public sources like the Securities and Exchange Commission (SEC) website of USA and HuggingFace website. The data will be cleaned, preprocessed, and prepared for training the ML and DL models and fine-tuning LLMs.
3. Result Discussions:
The results will contain the matrices like accuracy, recall, F1 score and precision to describe the performance of all the models implemented in this research.
5. Critical Analysis on results:
The results will interpret to understand why different models perform as they do. This includes looking at what makes them strong or weak and what their practical uses might be. It also includes detailed analysis and comparison of models implemented in this research.
6. Discussion:
Based on the findings and limitations, discussion for the limitations and the areas for future research will identify. Then it will conclude with the insights got from this research.

1.6 Dissertation outline

1. Critical Review of the Literature: The literature review section of the dissertation will encompass several key areas. It will begin by exploring traditional machine learning methods used in financial fraud detection, providing an overview of the techniques, their applications, and limitations. This exploration aligns with the objective to compare these methods with Large Language Models (LLMs). Next, the focus will shift to deep learning models, delving into their capabilities in detecting fraudulent activities in financial reports, aligning with the aim to understand and compare various modelling techniques. Finally, the section will critically review Large Language Models like FinBERT and GPT-2 specifically in the context of Natural Language Processing (NLP) for financial fraud detection. This part will cover their capabilities, and special focus on FinBERT, to directly relating to the dissertation's objectives to implement and evaluate LLMs.

2. Methodology: The methodology section of the dissertation will detail the approach taken to address the research aim and achieve the stated objectives. It will begin by outlining the data collection process, including the selection, preprocessing, and labelling of financial reports and statements for both genuine and fraudulent examples. Following this, the section will describe the implementation of traditional machine learning methods, deep learning models, and LLMs. The process of fine-tuning these models will be explained. Moreover, a systematic comparison between

LLMs and traditional models will be conducted, utilizing appropriate evaluation metrics such as accuracy, precision, F1 score, and recall.

3. Results & Evaluation: The results section of the dissertation will present the findings derived from the implementation and evaluation of traditional machine learning methods: Logistic Regression, Support Vector Machine, etc., deep learning models, and Large Language Models: FinBERT and GPT-2, in the context of financial fraud detection. It will provide a detailed account of the performance of these models, highlighting their effectiveness in identifying fraudulent activities within financial reports and statements. Quantitative results, including accuracy, precision, and recall, will be presented concisely to understand the performance difference among the models used in this paper. A critical evaluation will accompany the findings, analysing the strengths and weaknesses of each model, and interpreting the results considering the existing literature and the objectives of this research. The comparison between LLMs and traditional models will be emphasized, shedding light on their relative capabilities and limitations. This section will not only convey the empirical outcomes of the research but also provide insightful analysis.

5. Discussion: The discussion section will interpret the research findings, extending the existing studies in the field. It will analyse the implications of the results, emphasizing the unique contributions to financial fraud detection. The section will explore the comparative performance of the models used, their strengths and weaknesses, and how the findings align with the dissertation's objectives. Additionally, it will provide a reflective analysis, synthesizing the outcomes with existing knowledge, and offering insights into potential future research directions in the subject matter.

6. Conclusion: A summary of the key finding's conclusions drawn from the research

7. References: A comprehensive list of all the sources cited in the dissertation, formatted according to the required citation style.

8. Appendices: Supplementary materials such as data sets, code snippets, and additional analyses that support the main text of the dissertation will be in this section

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

Financial fraud detection has garnered significant attention due to its critical impact on stakeholders and regulatory bodies. Traditional machine learning models like Logistic Regression, SVM, Random Forest, as well as deep learning models like Hierarchical Attention Networks (HAN), have been extensively studied for their efficacy in this domain. However, the emergence of Large Language Models (LLMs) such as FinBERT and GPT-2 offers a new paradigm for detecting fraudulent activities, particularly in the Management Discussion and Analysis (MD&A) and financial statement sections of annual reports. This literature review aims to provide a comprehensive overview of the use of ML and DL models in only financial ratios to the state-of-the-art techniques in financial fraud detection through MD&A section, focusing on these specific sections of annual reports.

The objectives of this review are to introduce to the existing research on the traditional ML and DL models, identifying the research gaps and introducing the use of LLMs in fraud detection. By examining existing literature, this review will set the stage for this research, aiming to contribute to the field by filling identified gaps and offering new insights. After establishing the significance of financial fraud detection, further it will delve into the traditional machine learning models that have been foundational in this field.

2.2 Financial Fraud Detection Overview

The evolution of financial fraud detection has seen a shift from manual methods to more sophisticated analytical techniques. Early research in the field often relied on financial ratios as a key method for identifying fraudulent activities (Lenard and Alam, 2009; Ravisankar et al., 2011; Spathis, Doumpos and Zopounidis, 2002). These ratios were categorized into various groups, such as debt levels and liquidity measures, to assess the likelihood of fraudulent financial statements. For instance, higher levels of debt were considered a potential motivator for financial fraud (Kirkos, Spathis and Manolopolus, 2007; Ravisankar et al., 2011). Similarly, lower liquidity levels were also identified as a possible incentive for fraudulent activities (Lenard & Alam, 2009; Ravisankar et al., 2011). Over time, researchers expanded the scope of financial ratios used for fraud detection to include activity, profitability, and asset composition ratios (Song et al., 2014). While financial ratios were initially effective, their static nature made them less adaptable to the evolving complexities of financial fraud, necessitating the development of machine learning models.

2.3 Traditional Machine Learning Models in Financial Fraud Detection

Logistic Regression

Logistic regression has been identified as a predominant method in the realm of financial fraud detection, accounting for 13% of the data mining techniques applied between 2004 and 2015 (Albashrawi, 2021). Additionally, research by Yue et al. explored the use of LR for identifying fraudulent financial statements within a dataset of 174 Chinese companies. They employed 21 financial ratios as potential indicators and found that LR outperformed the earlier models like "Spathis," "Perons," and "Chen" in terms of accuracy rates for detecting fraudulent financial statements by approximately 10% in predictive accuracy (Yue et al., 2009). However, the study indicated that the initial selection of too many financial ratios could increase the complexity of the model. Additionally, the model serves as an indicator rather than conclusive evidence of fraud,

necessitating further investigation for confirmation. Therefore, for logistic model, it is difficult to capture the complex relation among the variables. Moreover, as this research is focused on both linguistic and financial variables, the LR underperformed many other complex models (Hajek and Henriques, 2017). As the LR has been a staple in financial fraud detection, its linear nature makes it less effective in capturing complex relationships between variables. This limitation is particularly evident when dealing with both linguistic and financial variables (Hajek and Henriques, 2017).

Support Vector Machines (SVM)

Support Vector Machines (SVMs) have been increasingly utilized due to their flexibility in handling both linear and non-linear classification problems. A study by Rizki, Surjandari and Wayasti focuses on Indonesian public companies and employs SVM to detect financial fraud. The study also delves into the impact of feature selection, revealing that this process improves SVM's accuracy to 88.37%. Similarly, Li and Ying studied the Lib-SVM algorithm of RBF (Radial Basis Function) kernel and linear kernel in detecting financial fraud from the financial statements, in which they got 87.5% accuracy from RBF kernel and 86.612% accuracy from linear kernel. But they got 80% and 83.036% accuracy respectively on LR. Therefore, in this study SVM outperformed LR. However, in both the studies they used financial ratios only which lack the uses of textual data of MD&A discussion. Therefore, Goel et al., explore the use of SVMs for fraud detection in the MD&A sections of annual reports. Utilizing the Waikato Environment for Knowledge Analysis (WEKA) for training, the study achieved better results with SVM compared to the Naïve Bayes classifier. However, the dataset required conversion to Attribute Relation File Format (ARFF) and consisted of a large feature set of 261,110 words, which could pose computational challenges. Extending this study, Humpherys et al., employed a 10-variable model focusing on various linguistic cues such as active language and syntactic complexity. The study achieved an accuracy rate of 65.8% using a 10-fold cross-validation but reported a high false positive rate. The dataset used contained 136 instances correctly classified and 66 incorrectly, raising questions about the model's reliability in practical applications. Similar but unique approach taken by Purda and Skillicorn. They trained SVMs on a language-based method using the MD&A sections of both annual and interim reports. The model achieved a high classification accuracy rate of up to 82%. The study also introduced temporal aspects and suggested that language deviations could be additional indicators for fraud detection. However, the study's reliance on SEC's AAER bulletins for training could limit the applicability of its results.

SVMs offer flexibility in handling both linear and non-linear problems, but their requirement for feature selection and computational intensity can be limiting factors, especially with large datasets.

Random Forest

Few studies implemented RF model in detecting frauds from financial statements. Hajek and Henriques highlighted the use of RF in detecting fraudulent transactions, emphasizing its capability to manage imbalanced datasets through techniques like SMOTE (Synthetic Minority Oversampling Technique). Patel et al., further elaborated on the application of Random Forest in classifying companies based on their financial statements. The study utilized a dataset of 178 companies, categorized into Fraud in Financial Statements (FFS) and Non-FFS, based on auditor opinions. A total of 31 financial ratios were initially considered, which were then narrowed down to 10 significant ratios through t-tests. Both achieved higher accuracy in respective studies. Hajek and Henriques reported an accuracy of 99.4% in transaction-based fraud detection, while Patel et al. achieved an accuracy range of 80% to 91% in company classification, with further improvements noted through

hyperparameter tuning. However, there is a need for a more holistic approach that integrates various types of financial data and textual data like MD&A sections of the reports.

Random Forest excels in managing imbalanced datasets through techniques like SMOTE. However, the model's performance can be compromised if it's not carefully tuned, leading to overfitting.

2.4 Deep Learning Models in Financial Fraud Detection

While traditional machine learning models have their merits, they also have limitations that newer deep learning models aim to address.

Artificial Neural Network (ANN)

The application of Artificial Neural Networks (ANNs) in financial fraud detection has shown varying degrees of success across different datasets. In a U.S.-based study, the dataset comprised 208 fraudulent and 7,341 non-fraudulent cases, sourced from companies' annual financial reports and the Compustat database (Craja, Kim and Lessmann, 2020). The ANN model in this study achieved a total accuracy rate of 67.9% on the testing dataset (Craja, Kim and Lessmann, 2020). However, its performance improved significantly to an accuracy rate of 89.1% when both financial and text data (FIN+TXT) were combined. On the other hand, a Taiwan-based study utilized a dataset covering the period from 1998 to 2010, which included 129 fraud companies and 447 non-fraud companies. The study considers 32 risk factors divided into three dimensions: Pressure/Incentive, Opportunity, and Attitude/Rationalization (Lin et al., 2015). These factors include financial metrics like debt/equity ratio, return on assets, and return on equity, as well as non-financial metrics like the percentage of sales-related party transactions and the frequency of CEO turnover. The ANN model in this context achieved a significantly higher total accuracy rate of 92.8% on the testing dataset (Lin et al., 2015). The reason for good performance of ANN is due to the Back-Propagation technique used during the training of the neural network and at some level it is good at understanding the textual data (Zinovyeva, Härdle and Lessmann, 2020). ANNs have shown promising results, especially when combining financial and text data. However, they are prone to overfitting and may not generalize well to new, unseen data (Zinovyeva, Härdle and Lessmann, 2020).

Hierarchical Attention Network (HAN)

The HAN has emerged as a state-of-the-art model in the realm of financial fraud detection, particularly when dealing with both textual and financial data. This model is developed by Craja, Kim and Lessmann in 2020, to address the hierarchical patterns of language, HAN has shown promise in capturing the nuances of financial reports and statements with the accuracy of **92.64 %**. The model also incorporates financial ratios, making it one of the few deep learning models to use both textual and quantitative features for fraud detection (Craja, Kim and Lessmann, 2020). While HAN has shown to be effective, it is essential to note that most comparisons have been made with traditional machine learning models and older deep learning architectures. HANs have emerged as a state-of-the-art model, particularly effective in capturing the hierarchical nature of language. However, their performance against newer, specialized models like FinBERT remains largely unexplored.

2.5 LLMs in Financial Fraud Detection

Language models like GTP-2 model is used by Craja, Kim and Lessmann in their paper: “Deep learning for detecting financial statement fraud”. However, it was not able to outperform the existing state-of-the-art model. Though the model is used with the attention mechanism, it was able to give around

77.7% AUC score on both textual data and combination of financial data and text data (Craja, Kim and Lessmann, 2020). However, the FinBERT model is not included in the study. Current literature primarily focuses on traditional machine learning models and lacks extensive research on the capabilities of Large Language Models like FinBERT in financial fraud detection. This gap is significant because LLMs have shown promise in other domains and could revolutionize fraud detection methods.

Therefore, this research aims to implement the FinBERT model using the Fine-tuning technique to train the model and will be tested rigorously on test and validation data and comparing with the other traditional machine learning and deep learning models.

CHAPTER 3: METHODOLOGY

3.1 Introduction

The methodology section will discuss about the self-attention mechanism, Transformer architecture, Fine-tuning the LLMs, description about the dataset used, the method of data collection, labelling and preparation to fine-tune the LLMs and traditional Machine Learning models, and the implementation and explanation of models trained on the dataset collected.

3.2 Self-Attention

Self-attention is a critical component in transformer models that helps in understanding the context of words within a sentence. Imagine a word like "bank" that can have different meanings depending on the surrounding words. The self-attention mechanism helps the model to differentiate these meanings by considering the relationships between words (Vaswani et al., 2017).

Words are represented as vectors, and the model calculates how much attention each word should pay to others. For example, in the sentences "The bank of the river" and "Money in the bank," the word "bank" has different meanings. The model aligns "bank" with "river" or "money" based on context. It then creates new vectors that capture these relationships, even considering multiple aspects through a process called Multi-Head Attention. Essentially, self-attention enables the model to interpret words in their specific context, much like how humans pay attention to cues in a conversation. This has made transformers highly effective in text understanding and generation (Serrano, n.d.).

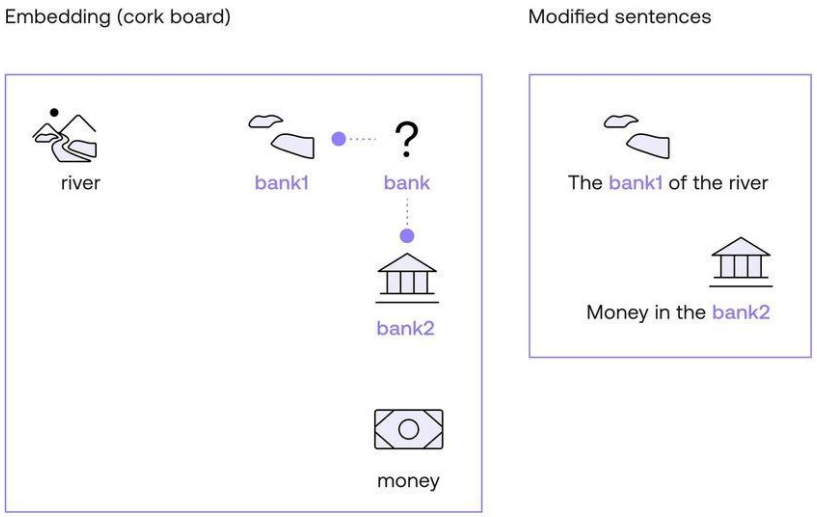


Fig. 1 Explaining the Self-Attention (Serrano, n.d.)

The Fig. 1 shows how Transformer's architecture stores the same word with the different context. It checks the whole sentence in which it will try to find the word "river" or "money" based on that it can predict the context of "bank" in that particular sentence.'

3.3 Transformer

Transformers is innovative architecture in natural language processing, introduced by Vaswani et al., –in 2017 in the paper "Attention Is All You Need." It created impact on the NLP field. They consist of several key components, including tokenization, embedding, positional encoding, transformer blocks, attention mechanisms, and a Softmax layer. The tokenization and Embedding blocks in the architecture is responsible to convert words into numerical vectors, while maintaining sequence of the words.

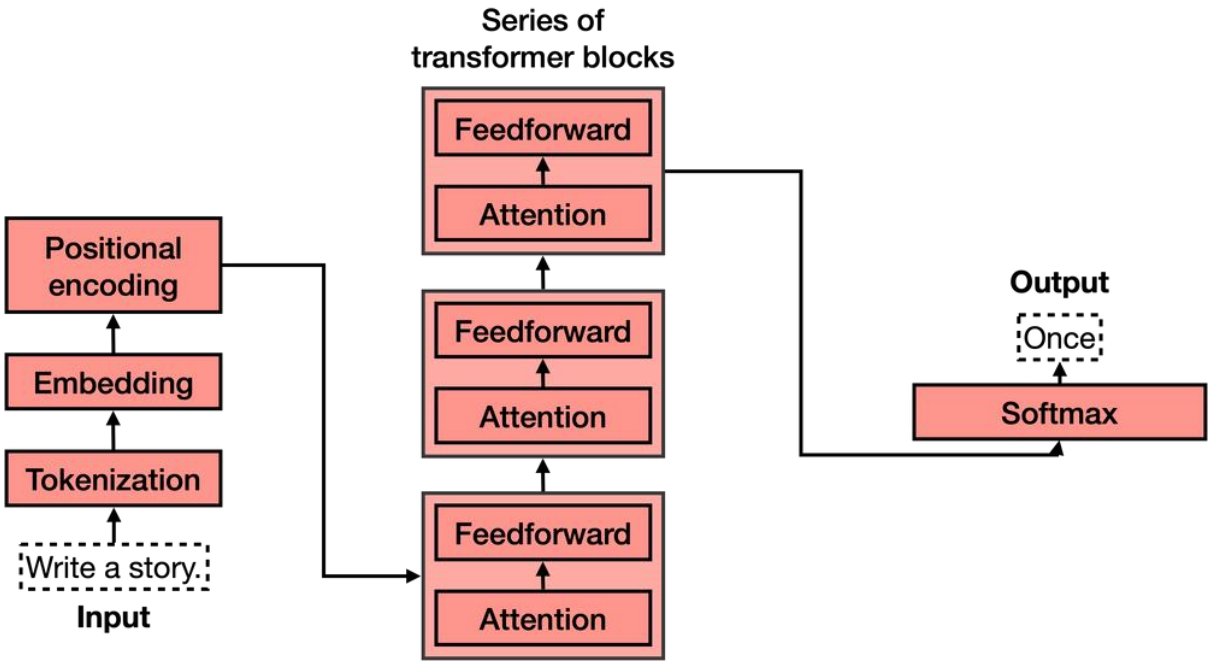


Fig. 2, Simplified Transformer's Arcitecture (Serrano, n.d).

As shown in the Figure 2, Transformer blocks, which contain attention and feedforward components, are concatenated to form the core of the model. Attention mechanisms, including multi-head attention, add context to each word based on the other words in the sentence. Finally, the Softmax layer turns scores into probabilities to predict the next word in the text.

The Transformer model begins its process with a tokenizer, which converts words into distinct tokens. These tokens are then transformed into numerical vectors through an embedding layer, representing them in a vector form. To maintain the sequence of words in the text, positional encoding is applied, adding the order to the words. The core of the model is the transformer block, responsible for predicting the next word. This block is composed of two main parts: an attention mechanism that infuses context into the text, and a feedforward neural network that aids in guessing the subsequent word. Finally, a Softmax layer is utilized to translate the scores into probabilities, facilitating the selection of the next word in the sequence. These repetition of steps makes the text coherent and because of high number of parameters it enables models to capture wider context of the given text.

3.4 Fine-tuning

The fine-tuning is the process which uses the linguistic information captured during the pre-training of the model on large corpus of textual data, allows the model to generalize better on the targeted task. Here, the dataset is prepared to fine-tune the model for the specific task to detect financial fraud from the 10K Fillings (textual data). The LLM models used in this experiment are FinBERT, pre-trained model on financial data, and GPT-2, pre-trained on general large corpus of textual data. These models are fine-tuned on our dataset. There are multiple types of fine-tuning the pre-trained models. This experiment uses Full Network Fine-Tuning method and Task-Specific Fine-Tuning method to train the FinBERT and GPT-2 model. In Full Network Fine-Tuning, the layers of the pre-trained FinBERT and GPT-2 model are being fine-tuned simultaneously on the new task (Fig. 3). Where the entire model's parameters are updated using the Adam optimizer. In Task-Specific Fine-Tuning, the model is fine-tuned for a binary classification task (detecting fraud). For example, the FinBERT model is used with a sequence classification head (BertForSequenceClassification), making it suitable for this specific task.

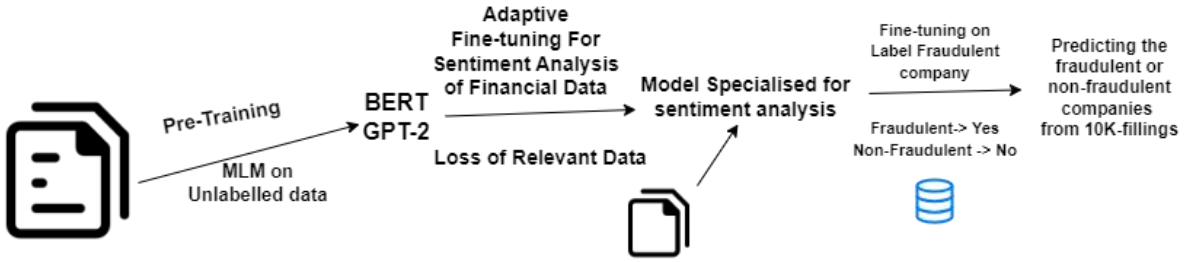


Fig. 3, Fine-tuning LLM models

3.5 Datasets

3.5.1 Data Collection & Preparation

Source and Structure: The dataset for this research is a collection of financial filings from various companies submitted to the U.S. Securities and Exchange Commission (SEC). Each row of the dataset represents a unique filing, with columns including the filename, Central Index Key (CIK), year of filing, sections containing textual information, company name, and a categorical variable indicating fraudulent activity. The data is collected from the secondary source.

Selection Criteria: The dataset consists of 85 companies involved in fraudulent cases and an equal number of companies not involved in any fraudulent activities. The fraudulent companies were extracted from the SEC's Enforcement Division's list of FCPA (Foreign Corrupt Practices Act) (www.sec.gov, 2BC), while the CIK numbers were obtained from the SEC's archives (Sec.gov, 2023). The reason for selecting data from the SEC is because it gives the universal format of reporting by companies which maintains the consistent format throughout the report. Moreover, it provides reliable 10-K fillings that are mandated by law. These filings are particularly valuable for detecting fraudulent activities.

Dataset Composition: The final dataset includes information such as the company's MD&A, and financial statement over the years the company stated on the SEC website. The reason to select MD&A and financial statement for our Fine-Tuning the LLM models is that the most important information present, where management discusses the company's financial performance and

condition, significant business trends, risks, and the outlook (Craja, Kim and Lessmann, 2020). Also, the limitations of tokens accepted by LLMs emphasised on reduce the number of columns. The data was extracted using a Python script (Reference to data extraction script) from the Hugging Face's 10K-Fillings dataset library (Loukas et al., 2021, pp.13--18) and is structured with specific columns like section_7 and Section_8 which contains the aforementioned information. The machine learning, deep learning and LLMs requires one column with the text and another one with the label as fraud and no fraud.

Dataset Preparation: The initial list of fraudulent companies is extracted from the Accounting and Auditing Enforcement Releases (AAER) website. This source is chosen for its high degree of trust and robust evidence of manipulation (Craja, Kim and Lessmann, 2020), as described in the literature. Each fraudulent company is matched with its corresponding Central Index Key (CIK) number. CIK numbers serve as unique identifiers for companies, ensuring precise identification. The relevant sections from the annual reports of fraudulent companies are appended to the list and labelled as "Yes" in a new "Fraud" column. This labelling is consistent with established practices that consider a report as fraudulent if the company that filed it was convicted. Since the appended list contains multiple years of statements placed in different rows, they are merged into one row. This step is crucial to prevent confusion in the model, as treating each row as a different company could adversely affect performance.

From the HF dataset, which contains more than 50,000 companies, a subset of 85 non-fraudulent companies is selected. This number corresponds to the number of fraudulent companies successfully extracted from the SEC. The selection is done using a simple sampling technique to overcome selection bias and avoid creating a highly imbalanced dataset. The newly appended non-fraudulent data is labelled as "No" in the "Fraud" column, clearly distinguishing it from fraudulent data. The dataset includes annual reports filed between 1993 and 2019, with specific focus on litigations issued during the period from 1999 to 2019. The final dataset consists of 85 fraud and 85 non-fraud filings. Therefore, total 170 rows in the dataset.

3.5.2 Data Preprocessing

Data preprocessing for this research involved several steps to ensure the dataset was suitable for training sophisticated language models like FinBERT and GPT-2 and the traditional Machine Learning (ML) and Deep Learning (DL) models. The process began with text cleaning, where irrelevant characters, symbols, and punctuations were removed, and the text was standardized to a consistent case. Tokenization was then applied to break down the text into tokens, a necessary step for converting the text into numerical vectors. Any rows with null values were deleted to avoid errors during model training. As the data is in the text form, therefore it is not possible to impute the missing values.

The text was further transformed through word embeddings, translating the textual information into a machine-readable format. Sequence padding was applied, as models requires the length of each input should same, therefore adjusted the length of text sequences to a fixed size, either by truncating longer sequences or padding shorter ones. which made all the input data in a uniform shape. Class balancing was maintained by selecting same number of both class companies to prevent model bias, and only the most relevant features, such as MD&A and financial statements, were retained.

Special attention was given to language model-specific preprocessing, including the addition of special tokens and adherence to specific formatting required by FinBERT and GPT-2. The reason for

giving special attention to capture context as well as the content of the managerial comments (Craja, Kim and Lessmann, 2020). However, there were some limitations of preprocessing due to which it could impact the performance of the model, which will be discussed in the further sections.

3.5.3 Libraries Used

Model	Libraries Used
Random Forest	pandas, scikit-learn (RandomForestClassifier, train_test_split, metrics)
SVM	pandas, scikit-learn (SVC, train_test_split, metrics), TfidfVectorizer
ANN	pandas, scikit-learn (train_test_split, LabelEncoder, metrics), tensorflow, keras
HAN (Hierarchical Attention Network)	pandas, numpy, tensorflow, nltk, scikit-learn (train_test_split, metrics)
GPT-2	pandas, transformers, scikit-learn (train_test_split, LabelEncoder, metrics), torch, time
BERT	pandas, transformers, scikit-learn (train_test_split, LabelEncoder, metrics), torch, time

Tabel 1. Libraries Used in the Methodology

Library	Purpose
pandas	Data manipulation and analysis. Used for handling CSV files and DataFrames.
scikit-learn	Machine learning algorithms. Provides classifiers, metrics, and data splitting utilities.
TfidfVectorizer	Text vectorization. Converts text data into numerical format.
tensorflow	Deep learning framework. Used for creating neural networks.
keras	High-level neural networks API. Used for creating deep learning models.
numpy	Numerical operations. Used for random sampling and other numerical tasks.
nltk	Text processing. Provides tools for text manipulation.

transformers	NLP models. Provides pre-trained models and tokenizers for natural language processing.
torch	Tensor computation. Provides a multi-dimensional array and the ability to perform operations.
time	Timing utilities. Used for tracking the time taken for model training.

Tabel 2. The Purpose of all the Libraries Used

The table 1 and table 2 are summarizing the libraries used and its purpose in the code.

3.6 Modelling

This section will explain about the implementation of different ML and DL models on the dataset prepared.

3.6.1 Model Selection

The central focus of the research is to understand the performance of LLMs in detecting financial fraud from the textual data. The transformer-based models are particularly effective in NLP task because the attention mechanism enables the model to focus on specific parts of the text that are most relevant to the task at hand. For example, when analysing an ambiguous term in a financial statement, the attention mechanism may distribute more focus on the term itself rather than the surrounding context, as found in the research on word sense disambiguation in neural machine translation (NMT) models by Tang, Sennrich and Joakim Nivre, 2018.

The objective of the research is to evaluate and compare the effectiveness of traditional Machine Learning models with Language Models in detecting financial fraud. Models such as Logistic Regression (LR), Support Vector Machine (SVM), XGBoost (XGB), Random Forest (RF), Artificial Neural Network (ANN), Hierarchical Attention Network (HAN), GPT-2 with attention, and Fin-BERT with attention were selected for training on a custom dataset.

These particular ML models were chosen because they have been successfully used by previous researchers in the field of financial fraud detection. However, for the purposes of this study, it was necessary to re-train all the models. This decision was made due to the unique nature of the dataset used in this research, which was manually collected and labelled, as opposed to datasets purchased from providers like Compustat or gathered from various other sources by other studies mentioned in the literature review.

The re-training ensured that the data remained consistent across the study, reflecting the specific structure and sources of the collected information. The selection of these models was also influenced by their consistent performance in similar tasks.

In addition to the traditional models, Language Models like FinBERT and GPT-2 were included in the study. FinBERT is considered a state-of-the-art model that has been pre-trained on financial data, making it highly relevant for this research. GPT-2 excels in general-purpose text processing.

The inclusion of these Language Models allows for an insightful comparison, exploring whether pre-training on financial data leads to significant improvements, especially in the detection of fraud within the MD&A and financial sections of annual reports.

3.6.2 Training and Validation

The dataset was divided into three distinct sets: training, validation, and test sets. The splitting was done using a 60-20-20 ratio, where 60% of the data was allocated for training, and the remaining 40% was equally divided between validation and test sets. This split ratio was chosen to ensure that the model had enough data to learn from while also having separate sets to tune hyperparameters and evaluate performance. Moreover, the split is done randomly to reduce the bias in training and test data (to maintain the reproducibility the random state is set to 42. So, if the code is re-run than also it will split the data in the similar fashion). The label encoder is used to convert the labels into 0s and 1s. Here, the 0 means non-fraud and 1 means fraud. In the transformer's model, the length of the input token is set to maximum length (512) as this is the limitations of the model. The parameter padding is used to fill the sequence with zeros at the end if the length of the tokens would be less than 512 and truncation parameter is used to reduce the length if it surpasses the maximum length.

FinBERT

FinBERT is a pre-trained language model specifically tailored for financial sentiment analysis. The reason for selecting this model is because it is trained on financial text data, which means it has an inherent understanding of financial terminology, jargon, and context. This makes it highly suitable for tasks related to financial analysis, such as fraud detection or sentiment analysis in financial reports.

By focusing on the financial domain, FinBERT can provide more accurate and nuanced interpretations of financial texts compared to general-purpose language models. This can lead to better performance in tasks that require a deep understanding of financial concepts. As the research is focused on detecting fraud from the statements, therefore the contextual understanding plays an important role to understand the reports thoroughly which will be advantage for this language model over other ML and DL models.

There were different types of FinBERT models are available on Hugging Face (HF). The FinBERT selected for this study is FinBERT-Pretrain (Huang, Wang and Yang, 2022; Yang, Anthony and Huang, 2020) because this model does not contain the last dense layers, therefore it can be designed for specific purpose. Moreover, there were hardly to no models which are specifically designed for financial fraud detection.

As the research focus is to classify the financial reports into fraudulent or non-fraudulent. The model was trained using the Adam optimization algorithm with a learning rate of 5e-5. The loss function used was the default loss provided by the BERT model for sequence classification. The training was conducted over three epochs, with a batch size of 32. The training loop also included attention masks, which help the model focus on the relevant parts of the input.

The validation set was used to tune the model and prevent overfitting. After each epoch of training, the model was evaluated on the validation set to check its performance on unseen data. The step helps to find the suitable model configuration.

GPT-2 with Attention

GPT-2 is known for its generative capabilities and ability to produce coherent and contextually relevant text but for general purposes. Therefore, it would be interesting to understand the performance of general-purpose models in financial context, if it is fine-tuned on the financial data.

Like FinBERT, the GPT-2 model was also loaded for sequence classification with two output labels. The model's token embeddings were resized to accommodate the additional padding token, and the

Adam optimizer was used with a learning rate of 5e-5. The reason to use this optimizer is that it is widely used for the classification problems.

The training loop was run for four epochs, with each batch containing eight samples. The loss *was* computed using the cross-entropy loss function, attention masks were utilized during training and the gradients were updated using the backpropagation algorithm. This way models learns to adapts to the mistakes it is making while predictions.

Like FinBERT model, the model is evaluated on validation set after training epochs.

Hierarchical Attention Network (HAN)

The HAN model was introduced by Craja, Kim and Lessmann in 2020. It is the state-of-the-art model in the financial fraud detection specially using the linguistic plus financial data. Like this research, HAN's architecture is also more focused on the MD&A and financial section of the annual reports. Therefore, it worth comparing the performance of the HAN with the existing transformer's model and ML models on this dataset.

The HAN has two distinct features which makes it the state-of-the-art. They are structured hierarchy of documents and attention mechanisms. Unlike other algorithms, HAN reflects the structured hierarchy of documents, capturing the content and context of managerial comments. Moreover, the model uses two different attention mechanisms at the word and sentence level, allowing content to be differentiated in terms of its importance in constructing the document representation (Craja, Kim and Lessmann, 2020).

As shown in the Figure 4 the architecture of the HAN is a sophisticated deep learning architecture designed to process hierarchical structured data. It begins by representing a document as a sequence of sentences, and each sentence as a sequence of words (Craja, Kim and Lessmann, 2020). These words are then converted into vectors using word embeddings. At the word level, an encoder like LSTM captures the dependencies between words, and an attention mechanism assigns importance to each word. These weighted words are combined to form a sentence vector. This process is repeated at the sentence level, where the sentence vectors are encoded and weighted to form a document vector. The document vector is then passed through fully connected layers and an activation function to generate the final output. The model is trained using a suitable loss function and optimization algorithm, iterating through the dataset multiple times. By applying attention mechanisms at both word and sentence levels, HAN focuses on the most informative parts of the document, leading to accurate and interpretable predictions (Craja, Kim and Lessmann, 2020). This way HAN successfully captures the context of whole document.

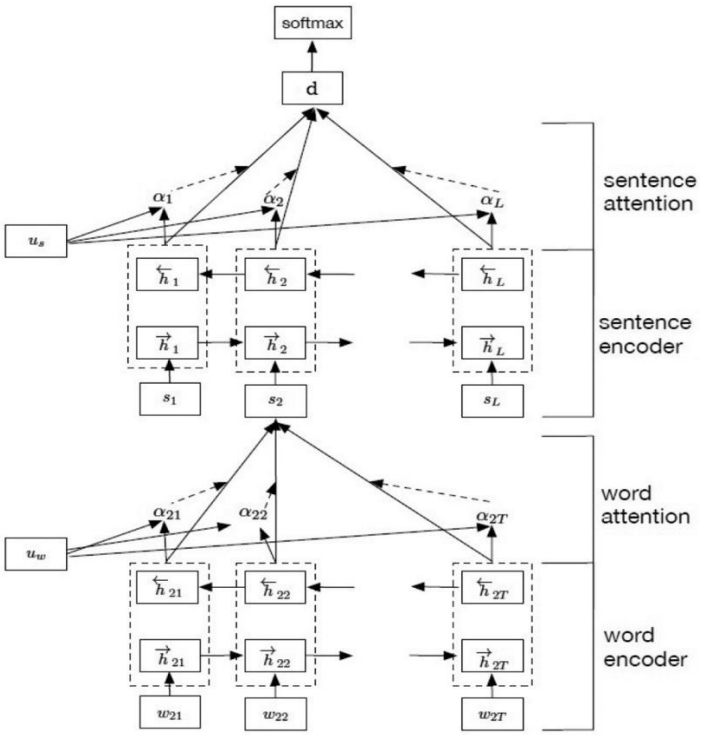


Fig. 4, HAN Architecture (Craja, Kim and Lessmann, 2020)

In this study, the HAN model is implemented with similar parameters used by Craja, Kim and Lessmann in their paper: "Deep learning for detecting financial statement fraud".

Logistic Regression (LR)

LR is a widely used statistical method for binary classification. In the context of this study, it serves as a baseline model to compare against more complex algorithms. Its simplicity, interpretability, and efficiency make it a suitable choice for initial exploration and analysis (Peng, Lee and Ingersoll, 2002).

$$\pi = \text{Probability}\,(Y = \textit{outcome of interest} \mid X_1 = x_1, X_2 = x_2 = \frac{e^{\alpha+\beta_1 X_1+\beta_2 X_2}}{1+e^{\alpha+\beta_1 X_1+\beta_2 X_2}}, \quad (4)$$

Where π is probability of event α is the Y intercept, βs are regression coefficients, and Xs are a set of predictors. α and βs are typically estimated by the maximum likelihood of the events.

The LR model was trained in this research using the TF-IDF vectorization of the text data. TF-IDF, short for Term Frequency-Inverse Document Frequency, is a numerical representation of text that combines the frequency of a term within a document (TF) with its rarity across a set of documents (IDF). TF measures how often a word appears in a specific document, while IDF adjusts this by penalizing words that are common across multiple documents. Together, they create a vector that captures the unique importance of words in a text corpus (Ramadhan, 2021).

The reason for using TF-IDF over other vectorization techniques is that it could provide a balance between complexity and performance, emphasizing relevant features in the text data while maintaining computational efficiency and interpretability. The optimization algorithm and loss

function were the default settings for logistic regression in the utilized library. The model was evaluated on both validation and test sets.

Random Forest (RF)

The Random Forest algorithm is an ensemble learning method that employs multiple decision trees during training. It operates on the principle of bagging, where different subsets of the dataset are used to train individual trees. The final prediction is determined by majority voting in classification problems or by averaging in regression tasks (Saini, 2021).

$$fi_i = \frac{\sum_{j:node\ j\ splits\ on\ feature\ i} ni_j}{\sum_{k \in all\ nodes} ni_k}$$

$$n_i = \frac{N_t}{N}[impurity\ -\ (\frac{N_{t(right)}}{N_t} * right\ impurity) - (\frac{N_{t(left)}}{N_t} * left\ impurity)]$$

$where\ N_t\ is\ number\ of\ rows\ that\ particular\ node\ has$

$N\ is\ the\ total\ number\ of\ rows\ present\ in\ data$

$Impurity\ is\ our\ gini\ index\ value$

$N_{t(right)}\ is\ number\ of\ nodes\ in\ right\ node$

$N_{t(left)}\ is\ number\ of\ nodes\ in\ left\ node$

Fig. 4. Mathematical Representation of RF model

Based on the Gini index, the RF model select the important features and divide into branches

In the context of financial fraud detection, it offers the advantage of handling high-dimensional data and capturing complex patterns. By utilizing TF-IDF vectorization, the model can recognize the significance of specific terms and phrases in the financial statements, contributing to the detection of fraudulent activities.

Like the LR, the RF model was trained with 100 trees using the TF-IDF vectorization of the text data with a maximum of 5000 features. The model was evaluated on validation and test sets, and the default settings for random forest classification were used.

Support Vector Machine (SVM)

SVM is a powerful classification algorithm that works by finding the optimal hyperplane that separates the classes. In the context of financial fraud detection, the linear kernel was chosen which makes it suitable for binary classification task (Ajay Yadav, 2018).

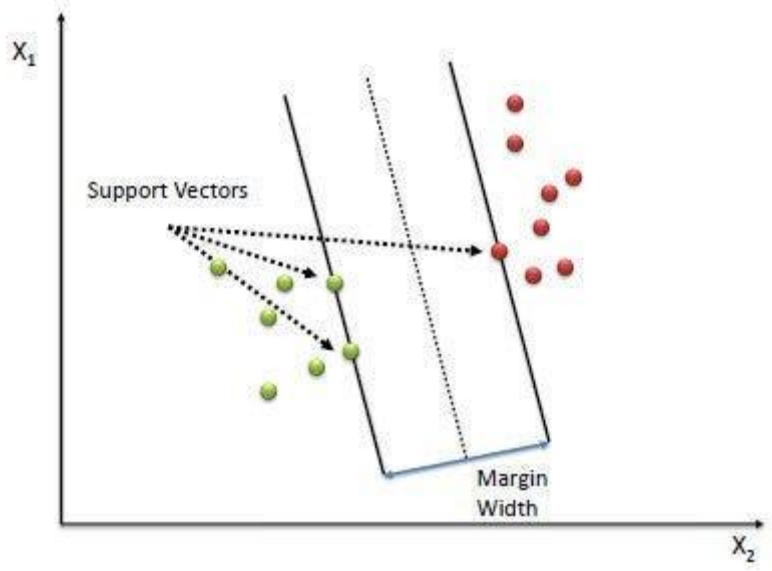


Fig. 5, Working of SVM

Like RF, utilizing TF-IDF vectorization, the model can recognize the significance of specific terms and phrases in the financial statements, contributing to the detection of fraudulent activities.

The model was evaluated on both validation and test sets, and the default settings for SVM classification were used.

Artificial Neural Network (ANN)

Artificial Neural Networks (ANNs) are computational systems inspired by the human brain's functioning, designed to adapt their internal structures to achieve specific objectives. They excel in handling nonlinear problems by dynamically adjusting the connections between their nodes, also known as processing elements (PEs). ANNs learn from data to uncover underlying patterns, making them particularly useful in complex problem-solving where the relationships between data points are not explicitly known (Grossi and Buscema, 2007).

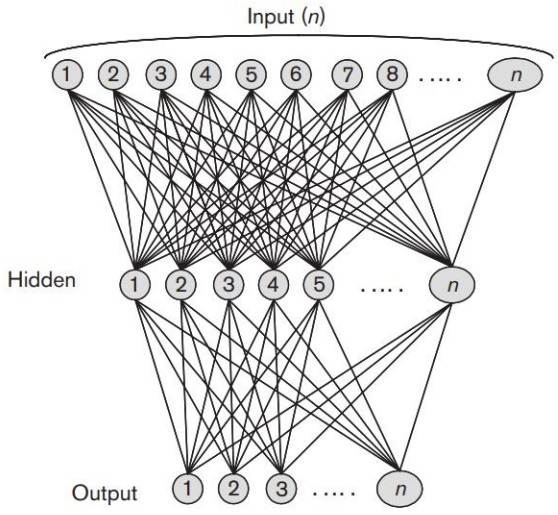


Fig. 6, Back Propagation of ANN

The mathematical representation of a basic artificial neural network's forward propagation. Here's a simplified formula for a single-layer ANN:

$$y = f(\sum_{i=1}^{n} w_i \cdot x_i + b)$$

Where, y is the output of the neuron, f is the activation function (e.g., Sigmoid, ReLU), w_i are the weights for each input x_i , b is the bias term and n is the number of inputs.

The ANN model was chosen for its ability to model complex relationships in the data. The 'tanh' and 'ReLU' (Rectified Linear Unit) activation functions were used in the hidden layers to introduce non-linearity. While 'tanh' provides a smooth gradient, 'ReLU' helps the model to overcome the vanishing gradient problem, allowing for faster learning.

For this study, it was trained using different architectures, including ReLU and tanh activation functions. The RMSprop optimizer was used with the binary cross-entropy loss function. The model was trained for 20 epochs with a batch size of 32, and early stopping was applied based on validation loss.

3.7 Evaluation Metrics:

The evaluation of model performance in the context of financial statement fraud detection revolves around a binary classification scenario, resulting in four potential outcomes. These outcomes comprise: True Positive (TP), which signifies the accurate classification of a fraudulent case. False Negative (FN), indicating the erroneous classification of a fraudulent case as non-fraudulent. True Negative (TN), demonstrating the accurate classification of a non-fraudulent case. False Positive (FP), representing the inaccurate classification of a non-fraudulent case as fraudulent.

To gauge the predictive capabilities of the models, a combination of assessment metrics is employed to quantify their efficacy. Prior research has explored metrics such as accuracy, sensitivity (recall), specificity, precision, and F1-score (Craja, Kim and Lessmann, 2020). This study assesses the model performance based on following matrices:

Area Under the Curve (AUC): The AUC serves as a metric to gauge the model's aptitude in correctly ranking both fraudulent and non-fraudulent cases in their correct order. A higher AUC corresponds to an enhanced ability to distinguish between the two distinct classes. The AUC is recognized for its robustness in the presence of imbalanced class distributions, rendering it particularly pertinent to fraud detection tasks (Craja, Kim and Lessmann, 2020).

$$\mathrm{AUC} = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{\mathrm{TP}_i}{\mathrm{TP}_i + \mathrm{FN}_i} \right)$$

Where, n is the number of thresholds used to calculate the ROC, TP_i is the number of true positives at the i^{th} threshold, FN_i is the number of false negatives at the i^{th} threshold.

Sensitivity: Also referred to as Recall or the True Positive Rate, sensitivity quantifies the proportion of actual fraudulent instances that the model correctly identifies. This metric serves as an indicator of the model's capability to effectively capture fraudulent occurrences.

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Specificity: Termed as the True Negative Rate, specificity gauges the proportion of actual non-fraudulent instances that the model accurately identifies. It offers insights into the model's precision in correctly classifying non-fraudulent cases.

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

Precision: Precision entails the ratio of accurately classified fraudulent instances to the total instances classified as fraudulent. This metric serves as a measure of the model's precision in labelling fraud cases.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

F1-Score: The F1-Score stands as the harmonic mean of precision and sensitivity. It provides a balanced assessment of the model's performance in accurately classifying both fraudulent and non-fraudulent instances. However, it treats precision and sensitivity with equal weight.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Within the realm of financial statement fraud detection, where the cost of failing to detect fraud often surpasses that of misclassifying non-fraudulent cases, metrics like sensitivity, precision, and AUC emerge as pivotal determinants of model effectiveness. Therefore, this study will be more focused on aforementioned matrices.

3.8 Implementation Details

The software used to implement all the model is Visual Studio Code on Windows 10. The libraries and the code used to run the models used in this study is included in the Appendix and all the references is present in each section to navigate the code for the particular model.

The models in this study were executed on a system equipped with a 13th Generation Intel Core i9-13900K processor, operating at a clock speed of 3.00 GHz, complemented by 64 GB of RAM. The system also featured a 12 GB Nvidia RTX A2000 Graphics card and ran on a 64-Bit Operating System.

3.9 Summary

The machine learning and deep learning models were generally used in previous research however, language models like GPT-2 and FinBERT were hardly used to detect financial fraud detection. Though there were limitations in this research as well. The limitations are with the selection of

model's parameters, models, dataset collection method and data preprocessing. The limitations will be discussed in the further sections of this thesis.

For all the algorithms used in this study, metrics used to evaluate the performance of the models are accuracy, precision, recall and F1-score. Because for the classification problems these are the key metrics used by existing research to evaluate the performance.

Table 3 gives the summary of the hyperparameters used in training all the models.

Model	Optimization Algorithm	Loss Function	Number of Epochs	Learning Rate	Batch Size	Other Hyperparameters
GPT-2 with Attention	Adam	Default (Sequence Classification)	4	5.00E-05	8	Attention Masks
Hierarchical Attention Network (HAN)	Adam	Binary Cross-Entropy	10	Default	50	LSTM Layers
Logistic Regression	Default (LR)	Default (LR)	N/A	Default	N/A	TF-IDF Vectorization
Random Forest (RF)	Default (RF)	Default (RF)	N/A	Default	N/A	100 Estimators, TF-IDF Vectorization
Support Vector Machine (SVM)	Linear Kernel	Default (SVM)	N/A	Default	N/A	TF-IDF Vectorization
Artificial Neural Network (ANN)	RMSprop	Binary Cross-Entropy	20	Default	32	ReLU/tanh Activation, Early Stopping
FinBERT with Attention	AdamW	Cross-Entropy	4	5.00E-05	32	12/24 Transformer Layers, 768/1024 Hidden Units

Tabel 3. Hyperparameters Used

CHAPTER 4: RESULTS AND EVALUATIONS

4.1. Introduction

The Results and Evaluation section is dedicated to presenting and interpreting the findings of the research conducted to detect financial fraud from 10K Fillings using various machine learning, deep learning, and language models.

The methodology employed a diverse set of models. These models were trained and fine-tuned on a dataset comprising financial filings from various companies submitted to the SEC. The dataset was thoroughly prepared, pre-processed, and divided into training, validation, and test sets to ensure robustness in the evaluation.

This section will provide a detailed account of the results obtained from each model, including performance metrics such as accuracy, precision, recall, and F1-score. A comparative analysis will be conducted to understand the relative strengths and weaknesses of the models, and the findings will be interpreted in light of the existing literature and theoretical framework.

The insights gained from this research will not only contribute to the academic understanding of financial fraud detection using advanced computational models but also have practical implications for the financial industry.

The following sub-sections will systematically present the results, conduct a comparative analysis, and evaluate the findings.

4.2 Results

4.2.1 Descriptive Statistics

This section will be focused on the descriptive analysis of the dataset that is being prepared from the SEC website. This will give an overview of how the dataset looks like.

Total number of companies present in the dataset is 170. In which 50% of them are fraudulent and another 50% are non-fraudulent companies.

Fraudulent Filings:

- **Mean:** 257,810 words
- **Median:** 269,696 words
- **Standard Deviation:** 162,822 words

Non-Fraudulent Filings:

- **Mean:** 62,937 words
- **Median:** 10,483 words
- **Standard Deviation:** 128,504 words

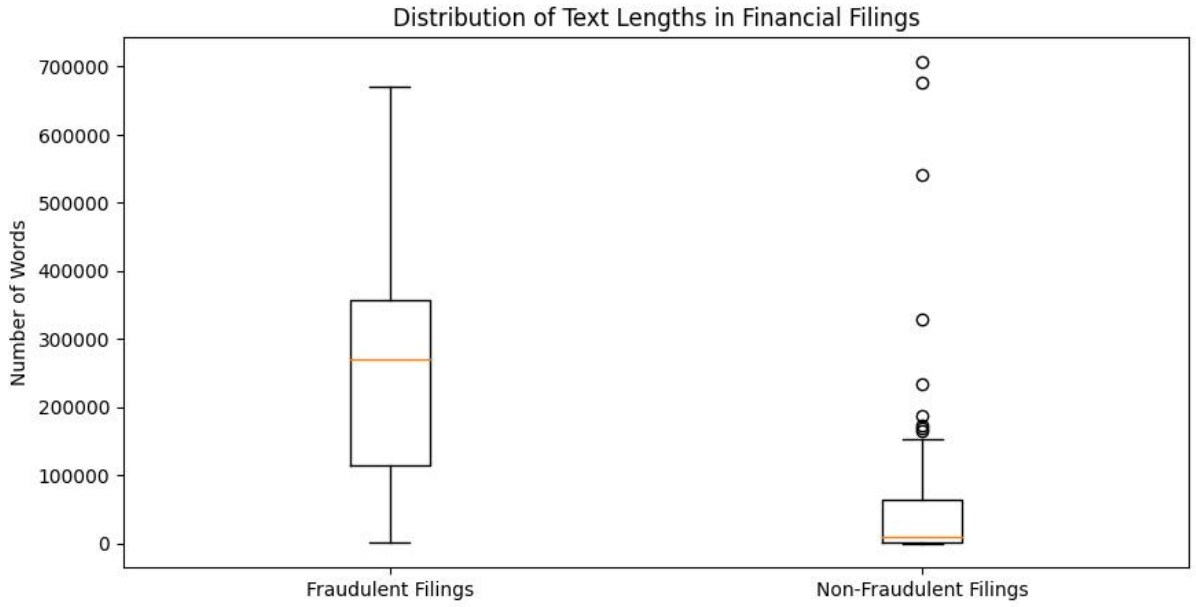


Fig. 7, Box Plot for Distribution of Text lengths

The Figure 7 shows the distributions of the words in fraudulent and non-fraudulent fillings. There is a significant difference in the distribution of words can be seen which indicates that generally fraudulent fillings will have a greater number of words. However, it cannot necessarily be a reliable indicator. Therefore, more in depth analysis is required to distinguish fraudulent with non-fraudulent.

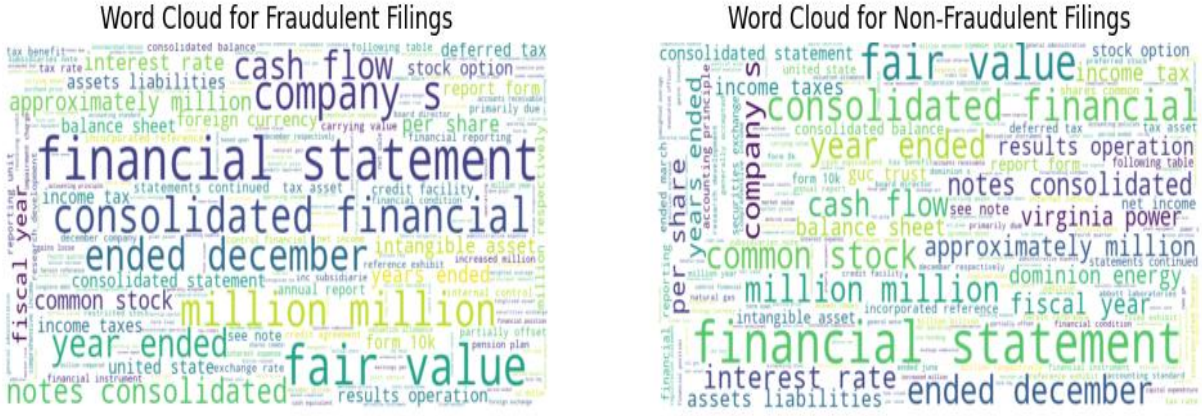


Fig. 8, Wordcloud for most frequent words in both the categories

The Figure 8 describes about the most commonly used words in the fraudulent and non-fraudulent fillings. It is difficult to differentiate the fraudulent and non-fraudulent companies with the most common words.

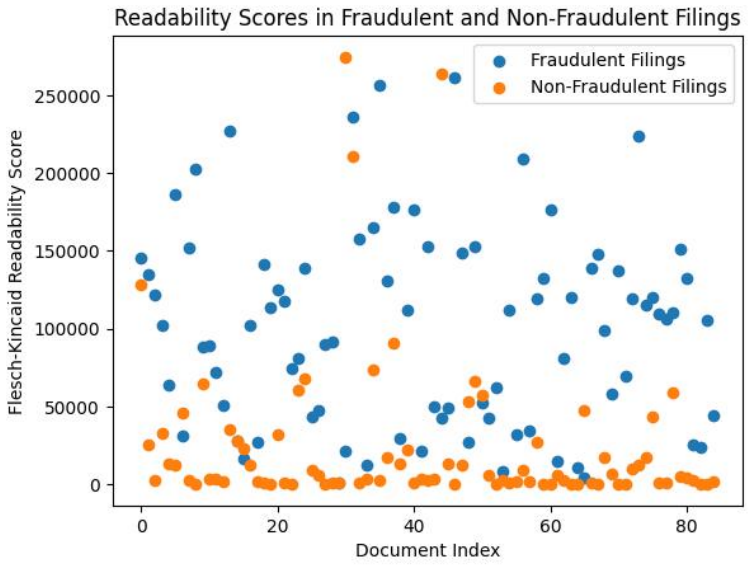


Fig. 9, Scatterplot for Readability score

The scatterplot plots the score of Flesch-Kincaid Grade Level for each filing in the dataset. The Flesch-Kincaid Grade Level is a readability test designed to indicate how difficult a reading passage is to understand. It's often used to assess the readability of English text, and the score represents the U.S. grade level of education required to understand the text.

Score= (0.39 × ASL) + (11.8 × ASW) − 15.59

It can be observed from the graph that fraudulent fillings are consistently showing the Flesch-Kincaid Grade Level score higher compared to the non-fraudulent fillings which indicates that fraudulent companies are using more complex language, possibly to obscure information or make fraudulent activities harder to detect. Conversely, non-fraudulent filings have lower scores, this suggests that the text is more accessible and can be understood by a wider audience. This could be seen as a sign of transparency, as non-fraudulent companies are communicating their financial information in a way that is more easily understood.

4.2.2 Model-Specific Results

In the context of detecting fraud, the goal is to minimize false positives as it could cost billions of dollars if fraudulent companies will be predicted as non-fraudulent. Therefore, the focus of this section will be to compare precision among the models.

Logistic Regression (LR)

Fig. 10, LR Metrics

The Logistic Regression model achieved a training accuracy of 79.41%, with precision, recall, and F1-score of 77.27%, 89.47%, and 82.93%, respectively. On the test set, the model maintained similar performance with an accuracy of 79.41%, precision of 76.47%, recall of 81.25%, and F1-score of 78.79%. There difference between accuracy of training and test matrix is insignificant therefore this model will perform similar on unseen data and less prone to overfit. However, the high recall in the training set at 89.47%, sees a decrease to 81.25% in the test set (Fig. 10). This indicate that while the model is good at identifying positive cases, the ability to generalize this to unseen data may need improvement. Precision is consistent but not exceptionally high.

Random Forest (RF)

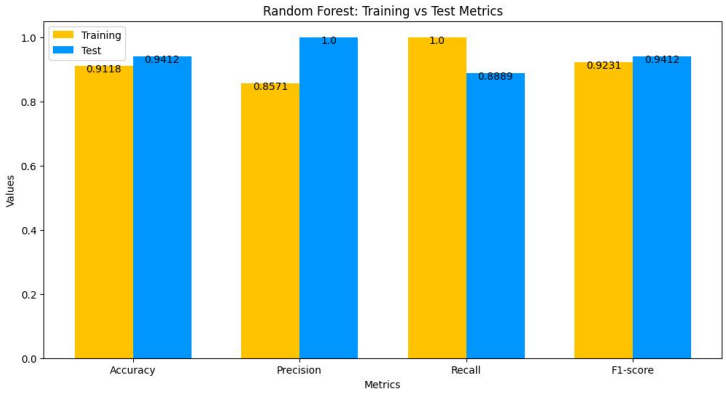


Fig. 11, RF Metrics

Random Forest demonstrated strong performance with a training accuracy of 91.18%, precision of 85.71%, recall of 100%, and F1-score of 92.31%. The test results were even more promising, with an

accuracy of 94.12%, precision of 100%, recall of 88.89%, and F1-score of 94.12% (Fig. 11). This suggests that the model is very effective in correctly identifying fraud cases while not falsely flagging legitimate cases.

Support Vector Machine (SVM)

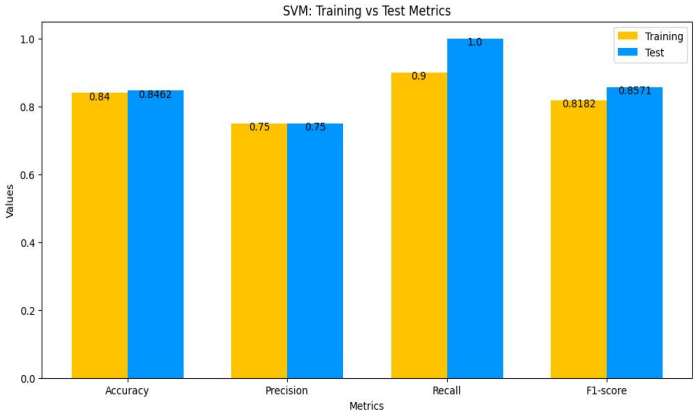


Fig. 12, SVM Metrics

The SVM model achieved a training accuracy of 84%, precision of 75%, recall of 90%, and F1-score of 81.82%. The test results were consistent with an accuracy of 84.62%, precision of 75%, recall of 100%, and F1-score of 85.71%. The precision of 75% confirms that the model is excellent at identifying fraudulent activities but may flag some false positives. With an F1-score of 81.82% in the training set, the model shows a balanced performance between precision and recall and test F1-score is slightly higher at 85.71%, reinforcing the balanced performance of the model (Fig. 12). Consistent performance metrics between training and test sets indicate that the model is not overfitting and is likely to generalize well to new data.

Artificial Neural Network (ANN)

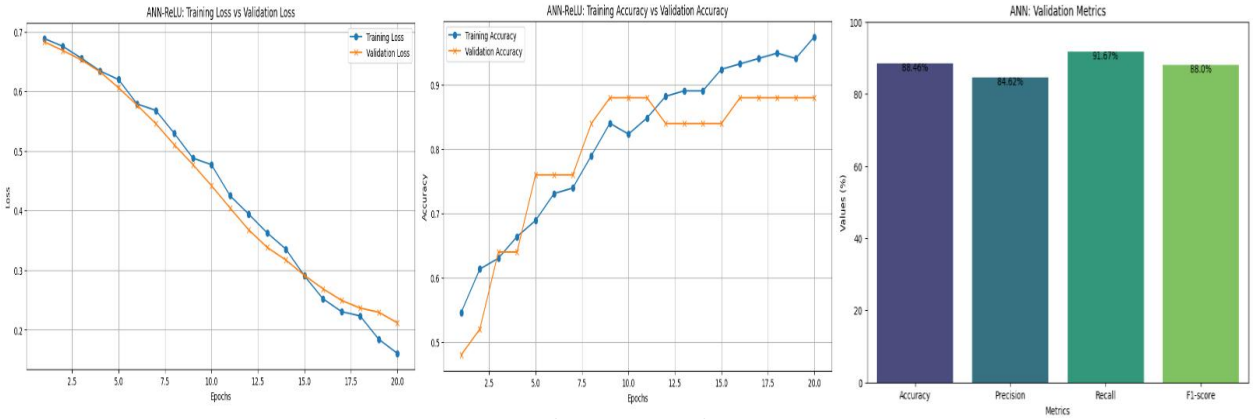


Fig. 13, ANN Metrics

Two different architectures were used for the ANN model. The first architecture, with ReLU activation functions, achieved a validation accuracy of 88%, precision of 81.82%, recall of 90%, and F1-score of

85.71%. The test results were similar, with an accuracy of 92.31%, precision of 85.71%, recall of 100%, and F1-score of 92.31%. The second architecture, with tanh activation functions, maintained similar performance metrics. Given the high validation and test accuracies, the ANN models appear reliable for detecting financial fraud. The similar performance between the two architectures suggests that the model is robust to the choice of activation function. The high recall rates indicate strong capabilities in fraud identification, but there could be a trade-off in terms of false positives, given the slightly lower precision. As shown in Figure 13, the training and validation loss is decreasing over the number of epochs and the accuracy is increasing.

Hierarchical Attention Network (HAN)

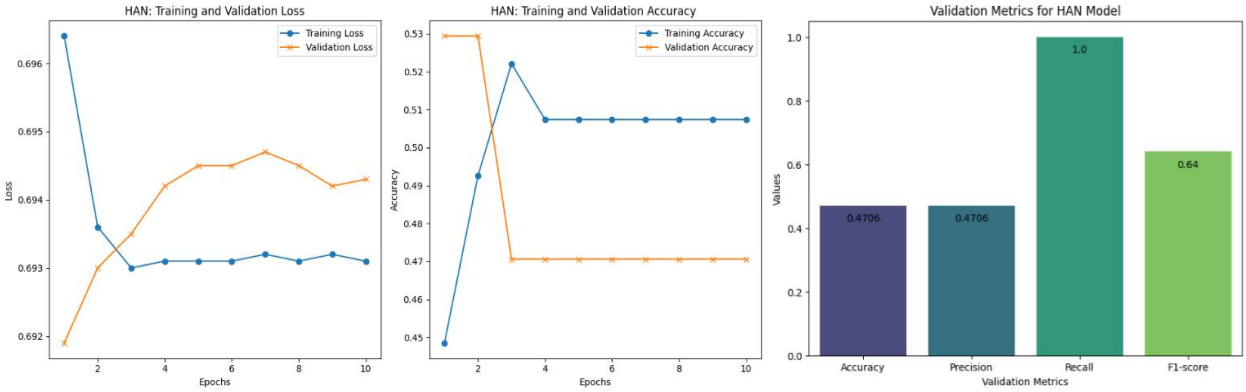


Fig. 14, HAN Metrics

The HAN model's performance was notably lower, with a validation accuracy of 47.06%, precision of 47.06%, recall of 100%, and F1-score of 64%. The perfect recall value model scores, suggesting that it is capable of identifying all true positives. However, given the low precision and accuracy, this comes at the cost of a high number of false positives. The low validation scores already raise concerns about its generalizability which means the model is not able to capture the given dataset well. The low precision could result in substantial costs for investigating false positives, which is a crucial consideration in fraud detection applications. As this model is the state-of-the-art model in the previous research, but due to the limitations of the dataset and preprocessing it may not perform well, which will discussed in further sections.

GPT-2 with Attention

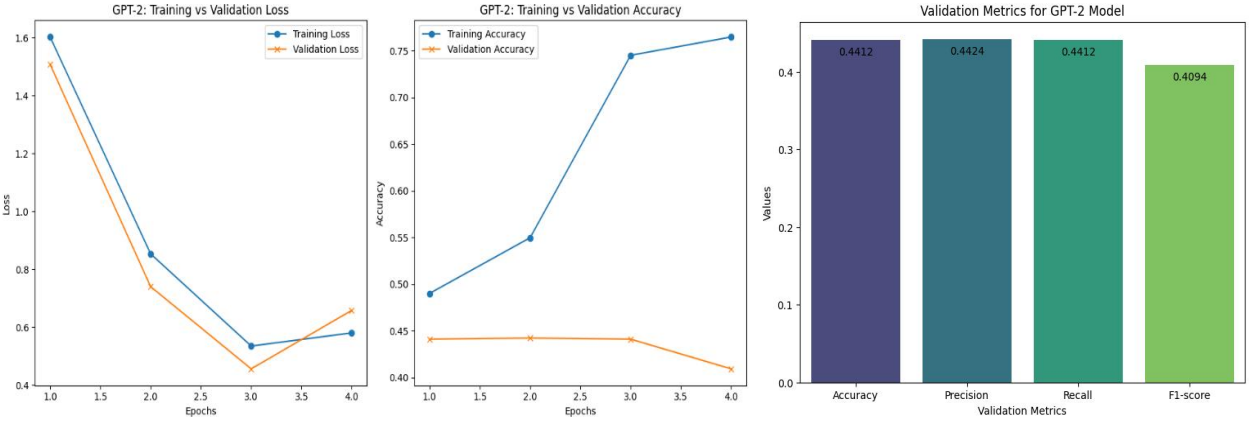


Fig. 15, GPT-2 Metrics

The GPT-2 model underwent four training epochs, with the final epoch achieving a training accuracy of 76.47%, precision of 71.19%, recall of 85.71%, and F1-score of 87.78%. The overall test results were less promising, with an accuracy of 44.12%, precision of 44.24%, recall of 44.12%, and F1-score of 40.94%. The model showed a strong training recall of 85.71%, indicating good sensitivity to identifying true positives. However, it falls dramatically to 44.12% in the test set, suggesting poor performance in identifying actual fraud cases in new data. The above figure shows that until epoch 3 the training and validation loss was decreasing, but it increased in the next epoch. On the other hand, the accuracy is increasing significantly over the epochs in the training accuracy but on validation the performance remained constant and decreased in the last epoch. This also suggests that generalizability would be difficult for this on the unseen data.

FinBERT with Attention

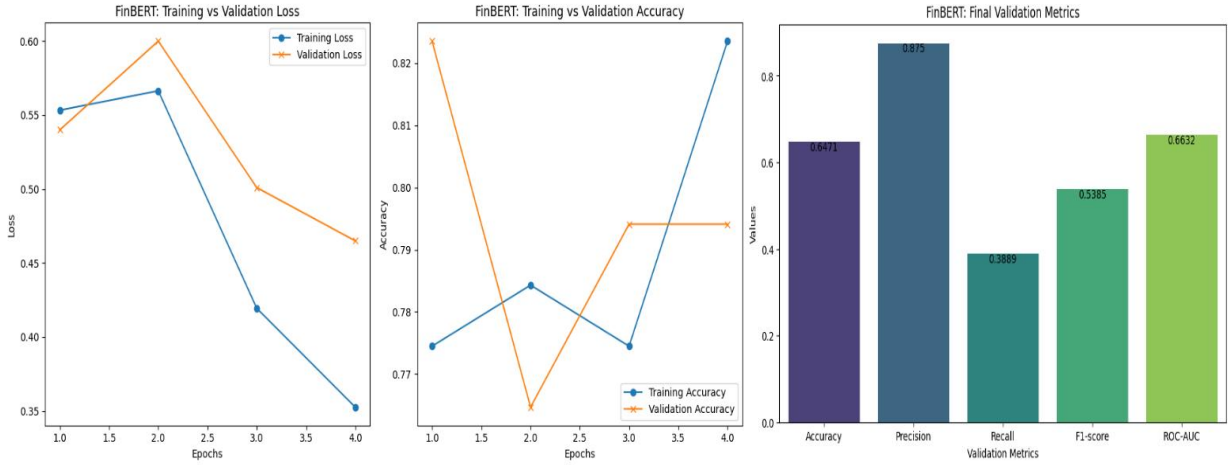


Fig. 16, FinBERT Metrics

The training accuracy stands at 82.35% and precision at 84.44%, which are strong indicators of the model's fit to the training data. The test metrics are relatively close to the training metrics, with an accuracy of 79.41% and precision of 78.95%. This consistency suggests good generalizability to new data. The training recall is 77.55%, indicating that the model is relatively good at identifying true positives. the test set, the recall is slightly better at 83.33%, suggesting that the model performs well in identifying actual cases of financial fraud in unseen data. An F1-score of 80.85% in the training set indicates a balanced performance between precision and recall. The test F1-score is very close to the training score, standing at 81.08%, further reinforcing the model's balanced performance. In the Figure 16, the training and validation loss is decreased after 2nd epoch significantly. In training and validation accuracy, after the 3rd epoch, the accuracy increased significantly but remained constant during the validation which concerns the generalizability on the unseen data. The bar shows, the balanced and consistent F1-scores suggest that FinBERT could be effectively used in a real-world setting without incurring excessive costs for false positives or missing too many actual fraud cases.

Model	Validation Accuracy	Validation Precision	Validation Recall	Validation F1-score	Test Accuracy	Test Precision	Test Recall	Test F1-score
Logistic Regression (LR)	79.41%	77.27%	89.47%	82.93%	79.41%	76.47%	81.25%	78.79%
Random Forest (RF)	91.18%	85.71%	100%	92.31%	94.12%	100%	88.89%	94.12%
Support Vector Machine (SVM)	84%	75%	90%	81.82%	84.62%	75%	100%	85.71%
Artificial Neural Network (ANN) - ReLU	88%	81.82%	90%	85.71%	92.31%	85.71%	100%	92.31%
Artificial Neural Network (ANN) - tanh	92%	90%	90%	90%	92.31%	85.71%	100%	92.31%
Hierarchical Attention Network (HAN)	47.06%	47.06%	100%	64%	47%	48%	100%	64%
GPT-2 with Attention	43.50%	43%	43%	40.94%	44.12%	44.24%	44.12%	40.94%
FinBERT with Attention	79.41%	78.95%	83.33%	81.08%	64.71%	87.50%	38.89%	53.85%

Table 4, Summary of All the Model Matrices

4.3 Comparative Analysis

Model	Validation Accuracy	Test Accuracy	Precision	Recall	F1-Score
Logistic Regression	79.41%	79.41%	77.27%	81.25%	78.79%
Random Forest	**91.18%**	**94.12%**	85.71%	88.89%	**94.12%**
SVM	84.00%	84.62%	75.00%	**100%**	85.71%
Artificial Neural Network (ReLU)	88.00%	92.31%	**81.82%**	**100%**	92.31%

HAN	47.06%	47%	47.06%	**100%**	64.00%
GPT-2	76.47% (Training)	44.12%	71.19%	44.12%	40.94%
FinBERT	**82.35%**	**79.41%**	**78.95%**	**83.33%**	**81.08%**

Table 5, Comparison of Model Performance

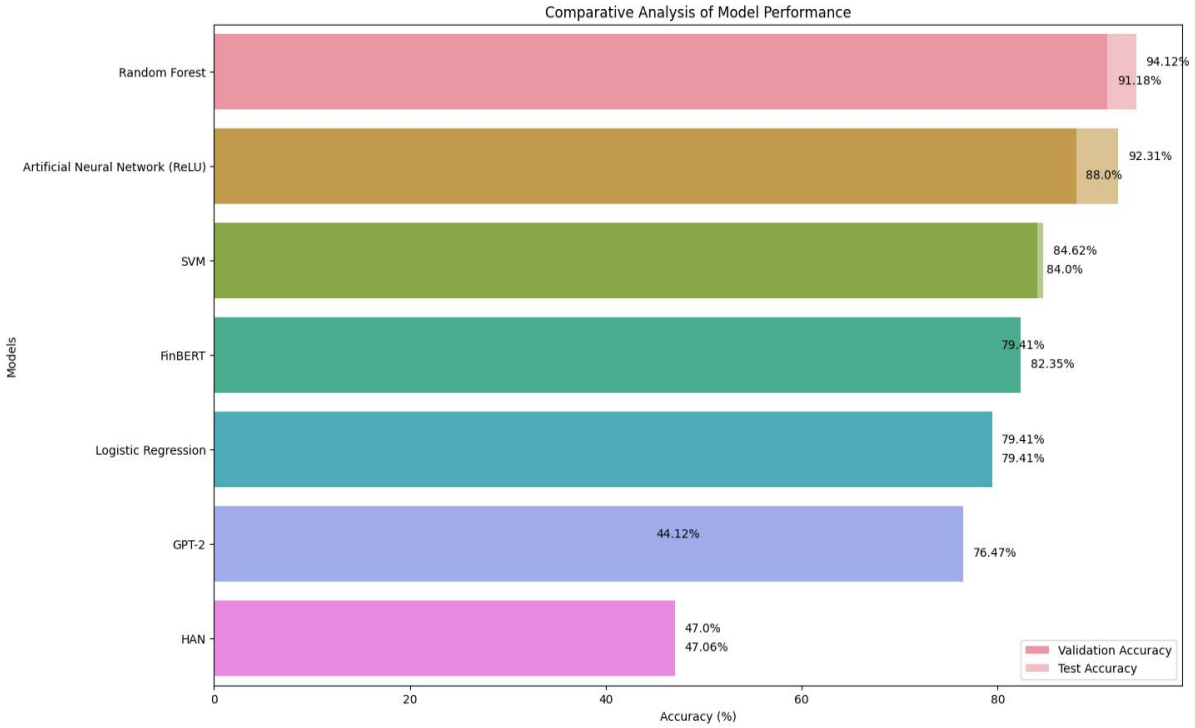


Fig. 17, Bar Graph for comparing the test and validation accuracy of models

Superior Performance of Random Forest and SVM Consistency (RQ3): The evaluation of various models in detecting fraudulent activities within financial reports has yielded compelling insights. In line with the findings of Hajek and Henriques, this study also found Random Forest to be a strong performer in financial fraud detection. Achieving a validation accuracy of 91.18%, this model excels in managing imbalanced datasets, corroborating the observations made by Patel et al. However, there are high chances of Random Forest to get overfit. Moreover, the Support Vector Machine (SVM) model exhibited consistent performance across the validation and test sets. With a validation accuracy of 84% and a test accuracy of 84.62%, coupled with stable precision, recall, and F1-score metrics, the SVM model emerged as a reliable contender for fraud detection tasks, which resonates with the studies conducted by Rizki, Surjandari, and Wayasti, as well as Li and Ying. However, similar to the challenges noted by Humpherys et al., feature selection and computational intensity remain as limitations, especially when dealing with large datasets.

Unforeseen Underperformance of HAN and GPT-2's Limited Applicability (RQ1, RQ2): Contrary to the high accuracy rates reported by Craja, Kim, and Lessmann, the HAN model underperformed with a validation accuracy of 47.06% and F1-score of 64% were unexpectedly low. This raises questions about its applicability in specialized tasks like financial fraud detection, emphasizing the need for further research to understand its limitations in this context. On the other hand, the GPT-2 model's performance was in line with the findings of Craja, Kim, and Lessmann, who reported an AUC score

of around 77.7%. Despite its potential, GPT-2 showed a test accuracy of 44.12%, questioning its suitability for specialized tasks like financial fraud detection. These outcomes underscore the significance of tailored model selection and careful consideration of model capabilities in alignment with the task at hand.

Balanced Performance of FinBERT for Fraud Detection (RQ1, RQ2): Among the models tested, As the study aims to fill the research gap identified in the literature regarding the capabilities of Large Language Models like FinBERT. Achieving a test accuracy of 79.41%, FinBERT presents itself as a balanced and reliable approach for financial fraud detection, especially when compared to traditional machine learning and deep learning models. With a test accuracy of 79.41% and an F1-score of 81.08%, FinBERT struck a harmonious trade-off between precision and recall. This equilibrium is vital in minimizing both false positives and false negatives, thus enhancing the model's applicability in real-world fraud detection settings. Its interpretability and credible performance make FinBERT a compelling option for financial statement fraud detection. However, it is not able to out-perform the model discussed in this research.

4.4 Evaluation and Interpretation

Interpretation of Findings: The Random Forest model's high performance aligns with existing literature on ensemble methods' effectiveness in complex tasks like financial fraud detection. The SVM model's consistent performance reaffirms its reliability, particularly where stable classification is crucial.

Model Strengths and Weaknesses:

Random Forest: Handles complex relationships but has high computational demands.

SVM: Consistent but vulnerable to performance degradation in imbalanced datasets.

HAN: Underperformed, indicating limitations in handling financial text data.

GPT-2: Showed promise in training but not in testing, suggesting limited applicability.

FinBERT: Balanced performance, making it suitable for real-world applications.

Implications: The implications drawn from our findings hold both practical and theoretical significance. The performance disparities across models underscore the importance of carefully selecting and fine-tuning models that resonate with the domain's characteristics. This extends to understanding the strengths and limitations of each model in the context of limited availability of data, a prevalent issue in financial fraud detection. Furthermore, this study's outcomes highlight the application of language models in financial fraud detection can be an innovative solution that harness the potential of emerging technologies. The findings also emphasize the need for further research into adapting and enhancing models to address the unique challenges posed by financial fraud detection tasks, contributing to the advancement of the field's methodology and knowledge base.

To conclude, the evaluation and interpretation of model performance provide a comprehensive understanding of their applicability and effectiveness in detecting fraudulent activities within financial reports. By relating these findings to existing literature and theoretical frameworks, the

detection of fraudulent activities from the reports is practically possible through the language models.

CHAPTER 5: DISCUSSION

In the context of financial fraud detection, the current research aimed to assess the effectiveness of Large Language Models (LLMs) in comparison to traditional machine learning and deep learning models. As highlighted in the literature review, while there have been studies on financial fraud detection using traditional models, the exploration of LLMs, particularly FinBERT and GPT-2, in this domain is not done yet. Therefore, this research is focused on the performance of LLMs and compare the performance of all the existing frequently used ML and DL models.

The Random Forest model emerged as a standout performer, aligning with existing literature that emphasizes the robustness of ensemble techniques in complex classification tasks. Its high accuracy and precision make it a strong contender for real-world financial fraud detection applications. On the other hand, the Support Vector Machine (SVM) model's consistent performance across datasets reaffirms its reliability, especially when consistency is paramount. However, the underperformance of models like GPT-2 and the Hierarchical Attention Network (HAN) underscores the challenges in adapting general-purpose models to specialized tasks like financial fraud detection.

The limitations posed by tokenization constraints, particularly for LLMs, were evident in the research outcomes. The restriction to 512 tokens for models like FinBERT and GPT-2 might have led to a loss of crucial information, affecting their performance. This highlights the need for models that can handle larger token sizes, especially when analysing extensive financial reports. The random selection of non-fraudulent companies is another limitation that could influence the generalizability of the findings. Moreover, this study did not explore the performance of big model like LLMA-2, Flacon, etc., which can be implemented in future research. Moreover, the state-of-the-art model BloombergGPT in the finance domain is not explored as it is a closed-source model (Wu et al., 2023). But it would be interesting to see how this model performs on detecting frauds from the annual reports. On top of that, the limitations of dataset and preprocessing technique may hamper the performance of existing DL models. The dataset size used in this research could be small to train the deep learning model or not removing the stop words and lemmatization during the preprocessing the dataset could limit the performance of the model. So, in the future research, it would be more insightful analysis to compare both the performance with and without preprocessing steps as aforementioned.

CHAPTER 6: CONCLUSION

6.1 Summary of the dissertation

As the aim of this research was to assess the effectiveness of Large Language Models (LLMs) like FinBERT and GPT-2 in detecting fraudulent activities in financial reports and statements, in comparison to traditional machine learning and deep learning models. The study found that Random Forest and Support Vector Machine (SVM) models demonstrated superior performance in terms of accuracy, precision, and F1-score. On the other hand, the Hierarchical Attention Network (HAN) and GPT-2 models underperformed, raising questions about their applicability in specialized tasks like financial fraud detection. FinBERT showed balanced performance, making it a viable option for real-world applications.

6.2 Research Contributions

The findings of this research contribute to the existing body of knowledge by introducing and evaluating the performance of LLMs in the context of financial fraud detection. The study underscores the importance of model selection, as not all models, including some state-of-the-art ones, are suitable for specialized tasks. The research has practical implications for the financial industry, offering insights into the strengths and weaknesses of various models, thereby aiding in the selection of the most effective tools for fraud detection.

This research contributes to the field by introducing and testing new models like FinBERT in financial fraud detection. It also provides a comprehensive evaluation framework that includes a variety of performance metrics, thereby offering a multi-faceted view of each model's effectiveness. The study fills a gap in the literature by comparing the performance of LLMs with traditional machine learning and deep learning models in this specific context. However, the study acknowledges several limitations, including the sample size and the scope of the research. The tokenization constraints for LLMs like FinBERT and GPT-2 may have led to a loss of crucial information. Additionally, the random selection of non-fraudulent companies could influence the generalizability of the findings.

6.3 Future Research and Development

Future research could focus on overcoming the limitations of token size in LLMs and exploring the performance of other state-of-the-art models like BloombergGPT. There is also a need to investigate the impact of data preprocessing techniques on model performance. Further studies could also delve into the ethical considerations of using machine learning models for financial fraud detection.

The research provides a comprehensive understanding of the applicability and effectiveness of various models in detecting fraudulent activities within financial reports. It highlights the need for careful model selection and fine-tuning, especially in specialized tasks like financial fraud detection. The study serves as a foundational work for future research in this area, aiming to contribute both to academic understanding and practical applications in the financial industry.

By reflecting on the outcomes, it's evident that while technology has advanced significantly, the task of financial fraud detection remains complex and demands a new approach. The study has been an enlightening journey into the capabilities and limitations of current technologies, offering valuable insights that could shape future research and practical applications in the field.

REFERENCES

Abbasi, A., Albrecht, C., Vance, A. and Hansen, J. (2012). MetaFraud: A Meta-Learning Framework for Detecting Financial Fraud. *MIS Quarterly*, 36(4), p.1293. doi:https://doi.org/10.2307/41703508.

Ajay Yadav (2018). *SUPPORT VECTOR MACHINES(SVM)*. [online] Medium. Available at: https://towardsdatascience.com/support-vector-machines-svm-c9ef22815589.

Albashrawi, M. (2021). Detecting Financial Fraud Using Data Mining Techniques: A Decade Review from 2004 to 2015. *Journal of Data Science*, 14(3), pp.553–570. doi:https://doi.org/10.6339/jds.201607_14(3).0010.

Albrecht, W.S., Albrecht, C. and Albrecht, C.C. (2008). Current Trends in Fraud and its Detection. *Information Security Journal: A Global Perspective*, 17(1), pp.2–12. doi:https://doi.org/10.1080/19393550801934331.

Araci, D. (2019). FinBERT: Financial Sentiment Analysis with Pre-trained Language Models. *arXiv:1908.10063 [cs]*. [online] Available at: https://arxiv.org/abs/1908.10063.

Beasley, M., Carcello, J. and Hermanson, D. (1999). *Fraudulent financial reporting: 1987-1997 : an analysis of U.S. public companies : research report*. [online] p.249. Available at: https://egrove.olemiss.edu/cgi/viewcontent.cgi?article=1330&context=aicpa_assoc.

Chen, X., Xie, H. and Tao, X. (2022). Vision, status, and research topics of Natural Language Processing. *Natural Language Processing Journal*, 1, p.100001. doi:https://doi.org/10.1016/j.nlp.2022.100001.

Craja, P., Kim, A. and Lessmann, S. (2020). Deep learning for detecting financial statement fraud. *Decision Support Systems*, p.113421. doi:https://doi.org/10.1016/j.dss.2020.113421.

Dorris, B. (2020). *President and CEO, Association of Certified Fraud Examiners President and CEO, Association of Certified Fraud Examiners FOREWORD Foreword Report to the Nations*. [online] Available at: https://acfepublic.s3-us-west-2.amazonaws.com/2020-Report-to-the-Nations.pdf.

DYCK, A., MORSE, A. and ZINGALES, L. (2010). Who Blows the Whistle on Corporate Fraud? *The Journal of Finance*, 65(6), pp.2213–2253. doi:https://doi.org/10.1111/j.1540-6261.2010.01614.x.

Gee, J., Button, M. and Brooks, G. (2019). *the financial cost of fraud what data from around the world shows*. [online] Available at: http://www.pkflb.com/pdf/The-Financial-Cost-of-Fraud.pdf [Accessed 1 Sep. 2023].

Goel, S., Gangolly, J., Faerman, S.R. and Uzuner, O. (2010). Can Linguistic Predictors Detect Fraudulent Financial Filings? *Journal of Emerging Technologies in Accounting*, 7(1), pp.25–46. doi:https://doi.org/10.2308/jeta.2010.7.1.25.

Grossi, E. and Buscema, M. (2007). Introduction to artificial neural networks. *European Journal of Gastroenterology & Hepatology*, 19(12), pp.1046–1054. doi:https://doi.org/10.1097/meg.0b013e3282f198a0.

Hadi, M., Al-Tashi, Q., Qureshi, R., Shah, A., Muneer, A., Irfan, M., Zafar, A., Shaikh, M., Akhtar, N., Wu, J. and Mirjalili, S. (2023). *A Survey on Large Language Models: Applications, Challenges, Limitations, and Practical Usage*.

Hajek, P. and Henriques, R. (2017). Mining corporate annual reports for intelligent detection of financial statement fraud – A comparative study of machine learning methods. *Knowledge-Based Systems*, 128, pp.139–152. doi:https://doi.org/10.1016/j.knosys.2017.05.001.

Huang, A., Wang, H. and Yang, Y. (2020). FinBERT—A Deep Learning Approach to Extracting Textual Information. *SSRN Electronic Journal*. doi:https://doi.org/10.2139/ssrn.3910214.

Huang, A.H., Wang, H. and Yang, Y. (2022). FinBERT : A Large Language Model for Extracting Information from Financial Text†. *Contemporary Accounting Research*. doi:https://doi.org/10.1111/1911-3846.12832.

Huang, S.-Y., Tsaih, R.-H. and Yu, F. (2014). Topological pattern discovery and feature extraction for fraudulent financial reporting. *Expert Systems with Applications*, 41(9), pp.4360–4372. doi:https://doi.org/10.1016/j.eswa.2014.01.012.

Humpherys, S.L., Moffitt, K.C., Burns, M.B., Burgoon, J.K. and Felix, W.F. (2011). Identification of fraudulent financial statements using linguistic credibility analysis. *Decision Support Systems*, 50(3), pp.585–594. doi:https://doi.org/10.1016/j.dss.2010.08.009.

Kanapickienė, R. and Grundienė, Ž. (2015). The Model of Fraud Detection in Financial Statements by Means of Financial Ratios. *Procedia - Social and Behavioral Sciences*, 213, pp.321–327. doi:https://doi.org/10.1016/j.sbspro.2015.11.545.

Khurana, D., Koli, A., Khatter, K. and Singh, S. (2022). Natural language processing: state of the art, current trends and challenges. *Multimedia Tools and Applications*, 82, pp.3713–3744. doi:https://doi.org/10.1007/s11042-022-13428-4.

KIRKOS, E., SPATHIS, C. and MANOLOPOULOS, Y. (2007). Data Mining techniques for the detection of fraudulent financial statements. *Expert Systems with Applications*, 32(4), pp.995–1003. doi:https://doi.org/10.1016/j.eswa.2006.02.016.

Lenard, M. and Alam, P. (2009). *An Historical Perspective on Fraud Detection: From Bankruptcy Models to Most Effective Indicators of Fraud in Recent Incidents Predictiing banking failures using RT programming View project Predicting bank failures View project*.

Li, X. and Ying, S. (2010). *Lib-SVMs Detection Model of Regulating-Profits Financial Statement Fraud Using Data of Chinese Listed Companies*. [online] IEEE Xplore. doi:https://doi.org/10.1109/ICEEE.2010.5660371.

Lin, C.-C., Chiu, A.-A., Huang, S.Y. and Yen, D.C. (2015). Detecting the financial statement fraud: The analysis of the differences between data mining techniques and experts' judgments. *Knowledge-Based Systems*, 89, pp.459–470. doi:https://doi.org/10.1016/j.knosys.2015.08.011.

Loukas, L., Manos , F., Ion , A. and Prodromos, M. (2021). *Proceedings of the Third Workshop on Economics and Natural Language Processing*. [online] *https://aclanthology.org/2021.econlp-1.2*, Association for Computational Linguistics, pp.13--18. Available at: https://aclanthology.org/2021.econlp-1.2/ [Accessed 7 Aug. 2023].

Osterrieder, J. (2023). A Primer on Natural Language Processing for Finance. *SSRN Electronic Journal*. doi:https://doi.org/10.2139/ssrn.4317320.

Patel, H., Parikh, S., Patel, A. and Parikh, A. (2018). An Application of Ensemble Random Forest Classifier for Detecting Financial Statement Manipulation of Indian Listed Companies. *Advances in Intelligent Systems and Computing*, pp.349–360. doi:https://doi.org/10.1007/978-981-13-1280-9_33.

Peng, C.-Y.J., Lee, K.L. and Ingersoll, G.M. (2002). An Introduction to Logistic Regression Analysis and Reporting. *The Journal of Educational Research*, 96(1), pp.3–14. doi:https://doi.org/10.1080/00220670209598786.

Purda, L. and Skillicorn, D. (2014). Accounting Variables, Deception, and a Bag of Words: Assessing the Tools of Fraud Detection. *Contemporary Accounting Research*, 32(3), pp.1193–1223. doi:https://doi.org/10.1111/1911-3846.12089.

Ramadhan, L. (2021). *TF-IDF Simplified*. [online] Medium. Available at: https://towardsdatascience.com/tf-idf-simplified-aba19d5f5530.

Ravisankar, P., Ravi, V., Raghava Rao, G. and Bose, I. (2011). Detection of financial statement fraud and feature selection using data mining techniques. *Decision Support Systems*, 50(2), pp.491–500. doi:https://doi.org/10.1016/j.dss.2010.11.006.

Rizki, A.A., Surjandari, I. and Wayasti, R.A. (2017). *Data mining application to detect financial fraud in Indonesia's public companies*. [online] IEEE Xplore. doi:https://doi.org/10.1109/ICSITech.2017.8257111.

Saini, A. (2021). *An Introduction to Random Forest Algorithm for beginners*. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2021/10/an-introduction-to-random-forest-algorithm-for-beginners/#:~:text=Random%20forest%20randomly%20selects%20observations.

Sec.gov. (2023). *cik-lookup-data*. [online] Available at: https://www.sec.gov/Archives/edgar/cik-lookup-data.txt [Accessed 7 Sep. 2023].

Serrano, L. (n.d.). *The Attention Mechanism*. [online] Cohere AI. Available at: https://docs.cohere.com/docs/the-attention-mechanism [Accessed 30 Aug. 2023].

Serrano, L. (n.d.). *Transformer Models*. [online] Cohere AI. Available at: https://docs.cohere.com/docs/transformer-models [Accessed 7 Sep. 2023].

Sharma, A. and Kumar Panigrahi, P. (2012). A Review of Financial Accounting Fraud Detection based on Data Mining Techniques. *International Journal of Computer Applications*, [online] 39(1), pp.37–47. doi:https://doi.org/10.5120/4787-7016.

Simnett, R., Vanstraelen, A. and Chua, W.F. (2007). *Assurance on Sustainability Reports: An International Comparison*. [online] Social Science Research Network. Available at: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=1025467 [Accessed 3 Sep. 2023].

Song, X.-P., Hu, Z.-H., Du, J.-G. and Sheng, Z.-H. (2014). Application of Machine Learning Methods to Risk Assessment of Financial Statement Fraud: Evidence from China. *Journal of Forecasting*, 33(8), pp.611–626. doi:https://doi.org/10.1002/for.2294.

Spathis, Ch., Doumpos, M. and Zopounidis, C. (2002). Detecting falsified financial statements: a comparative study using multicriteria analysis and multivariate statistical techniques. *European Accounting Review*, 11(3), pp.509–535. doi:https://doi.org/10.1080/0963818022000000966.

Tang, G., Sennrich, R. and Joakim Nivre (2018). An Analysis of Attention Mechanisms: The Case of Word Sense Disambiguation in Neural Machine Translation. doi:https://doi.org/10.18653/v1/w18-6304.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, Ł. and Polosukhin, I. (2017). *Attention Is All You Need*.

West, J. and Bhattacharya, M. (2016). Intelligent financial fraud detection: A comprehensive review. *Computers & Security*, [online] 57, pp.47–66. doi:https://doi.org/10.1016/j.cose.2015.09.005.

Wu, S., Irsoy, O., Lu, S., Dabravolski, V., Dredze, M., Gehrmann, S., Kambadur, P., Rosenberg, D. and Mann, G. (2023). BloombergGPT: A Large Language Model for Finance. *arXiv:2303.17564 [cs, q-fin]*. [online] Available at: https://arxiv.org/abs/2303.17564.

www.sec.gov. (2BC). *SEC Enforcement Actions: FCPA Cases*. [online] Available at: https://www.sec.gov/enforce/sec-enforcement-actions-fcpa-cases [Accessed 7 Aug. 2023].

Yang, Y., Anthony, M. and Huang, A. (2020). FinBERT: A Pretrained Language Model for Financial Communications. *arXiv (Cornell University)*. doi:https://doi.org/10.48550/arxiv.2006.08097.

Yue, D., Wu, X., Shen, N. and Chu, C.-H. (2009). Logistic Regression for Detecting Fraudulent Financial Statement of Listed Companies in China. *2009 International Conference on Artificial Intelligence and Computational Intelligence*. doi:https://doi.org/10.1109/aici.2009.421.

Zinovyeva, E., Härdle, W.K. and Lessmann, S. (2020). Antisocial online behavior detection using deep learning. *Decision Support Systems*, p.113362. doi:https://doi.org/10.1016/j.dss.2020.113362.

APPENDIX A

This section contains the Ethical Approval Letter for the dissertation on financial fraud detection

College of Engineering, Design and Physical Sciences Research Ethics Committee
Brunel University London
Kingston Lane
Uxbridge
UB8 3PH
United Kingdom

www.brunel.ac.uk

10 July 2023

LETTER OF CONFIRMATION

Applicant: Mr. Amit Shushil Kedia

Project Title: Enhancing Financial Fraud Detection: A Comparative Study of State-of-the-Art Natural Language Processing Models and Traditional Deep Learning Approaches

Reference: 43787-NER-Jun/2023- 45615-1

Dear Mr. Amit Shushil Kedia,

The Research Ethics Committee has considered the above application recently submitted by you.

This letter is to confirm that, according to the information provided in your BREO application, your project does not require full ethical review. You may proceed with your research as set out in your submitted BREO application, using secondary data sources only. You may not use any data sources for which you have not sought approval.

Please note that:

- **You are not permitted to conduct research involving human participants, their tissue and/or their data. If you wish to conduct such research (including surveys, questionnaires, interviews etc.), you must contact the Research Ethics Committee to seek approval prior to engaging with any participants or working with data for which you do not have approval.**
- The Research Ethics Committee reserves the right to sample and review documentation relevant to the study.
- If during the course of the study, you would like to carry out research activities that concern a human participant, their tissue and/or their data, you must submit a new BREO application and await approval before proceeding. Research activity includes the recruitment of participants, undertaking consent procedures and collection of data. Breach of this requirement constitutes research misconduct and is a disciplinary offence.

Good luck with your research!

Kind regards,

Professor Simon Taylor

Chair of the College of Engineering, Design and Physical Sciences Research Ethics Committee

Brunel University London

Page 1 of 1

APPENDIX B

APPENDIX B1 (Data Preparation)

Step 1: Matching Company Names and Appending CIK Numbers (Code 1)

Libraries to Install

- **pandas**: Install using **pip install pandas**.
- **fuzzywuzzy**: Install using **pip install fuzzywuzzy**.
- **datasets**: Install using **pip install datasets**.

Detailed Steps

1. **Load Dataset**:
    - The Edgar corpus dataset is loaded using **load_dataset** from the **datasets** library. This dataset contains financial reports of companies.
    - Three splits (train, validation, test) are converted to pandas DataFrames for easier manipulation.
2. **Fuzzy Matching**:
    - The **fuzz** and **process** modules from **fuzzywuzzy** are used to perform text matching.
    - The function **match_name** takes a company name and a list of names to match against, returning the CIK number if a match is found with a minimum score of 70.
    - This is crucial because company names can have slight variations, and fuzzy matching helps in capturing these nuances.
3. **Save to CSV**:
    - The DataFrame containing the matched company names and their CIK numbers is saved to a CSV file.
    - This is important for traceability and for use in subsequent steps.

```
# This file contains the loading of the dataset from the edgar
corpus in hugging face dataset.
# Matching the existing data of company names with the name of the
company from the edgar corpus using the fuzzy logic.
# Appending the CIK number to the company name.

import pandas as pd
from fuzzywuzzy import fuzz, process
from datasets import load_dataset
dataset = load_dataset('c3po-ai/edgar-corpus')
```

Code 1:

```
# Convert the training split to a DataFrame
df_train = pd.DataFrame(dataset['train'])
# Convert the validation split to a DataFrame
df_validation = pd.DataFrame(dataset['validation'])

# Convert the test split to a DataFrame
df_test = pd.DataFrame(dataset['test'])
file_path = "C://DissData/sec.gov_Archives_edgar_cik-lookup-
data.txt"

# Read the file line by line
with open(file_path, 'r') as file:
    lines = file.readlines()

fraud_companies =
pd.read_excel("C://DissData//Fraud_Companies_dataset.xlsx")

data_list = [line.rsplit(":", 2)[:2] for line in lines]

# Create a DataFrame
df = pd.DataFrame(data_list, columns=["Company Name", "CIK
Number"])

cik_dict = pd.Series(df["CIK Number"].values, index=df["Company
Name"]).to_dict()

# Function to match company names and return CIK number
def match_name(name, list_names, min_score=0):
    # -1 score incase we don't get any match
    max_score = -1
    # Returning empty name for no match as well
    max_name = ""
    # Iternating over all names in the other
    for name2 in list_names:
        #Finding fuzzy match score
        score = process.extractOne(name, [name2],
score_cutoff=min_score)
        # Checking if we are above our threshold and have a
better score
        if score and score[1] > max_score:
            max_name = name2
            max_score = score[1]
    return cik_dict.get(max_name)

# Use the function to match the company names and get the CIK
number
fraud_companies["CIK Number"] = fraud_companies["Company
Name"].apply(lambda x: match_name(x, cik_dict.keys(), 70))

fraud_companies_to_save = fraud_companies[["Company Name", "CIK
Number"]]
```

Step 2: Data Collection and Preparation (Code 2)

Libraries to Install

- **torch, torchdata**: Install using **pip install torch torchdata**.
- **transformers, datasets**: Install using **pip install transformers datasets**.

Detailed Steps

1. **Data Conversion**:
    - The Edgar corpus is loaded in shards to manage memory usage effectively.
    - Each shard is converted to a pandas DataFrame, and all are concatenated to form a single DataFrame.
2. **Data Labeling**:
    - A new column named **Fraud** is added to label companies as fraud or non-fraud.
    - This is essential for supervised learning models to learn the patterns associated with fraudulent activities.
3. **Data Aggregation**:
    - The data is grouped by CIK numbers, and text from different sections is aggregated.
    - This ensures that each company has a single, comprehensive record.
4. **Data Cleaning**:
    - Text cleaning is performed to remove special characters, convert all text to lowercase, and remove stop words. This is crucial for NLP tasks as clean data improves model performance.

Code 2:

```
# Data Collection & Preparation

from datasets import load_dataset
from transformers import AutoModelForSeq2SeqLM
from transformers import AutoTokenizer
from transformers import GenerationConfig
from datasets import load_dataset
import pandas as pd

# Load the dataset
dataset = load_dataset('c3po-ai/edgar-corpus')

# Number of shards
num_shards = 10

# Initialize an empty list to hold the DataFrames
dataframes = []
```

```
# Convert each shard to a pandas DataFrame
for i in range(num_shards):
    shard = dataset['validation'].shard(num_shards, i)
    df_shard = shard.to_pandas()
    dataframes.append(df_shard)

# Concatenate all the DataFrames
df_validation = pd.concat(dataframes, ignore_index=True)
# Number of shards
num_shards = 10

# Initialize an empty list to hold the DataFrames
dataframes = []

# Convert each shard to a pandas DataFrame
for i in range(num_shards):
    shard = dataset['test'].shard(num_shards, i)
    df_shard = shard.to_pandas()
    dataframes.append(df_shard)

# Concatenate all the DataFrames
df_test = pd.concat(dataframes, ignore_index=True)
# Concatenate all splits
all_data = pd.concat([df_train, df_validation, df_test])
all_data.head()
fraud_companies_df = pd.read_csv('C:/DissData/Dissertation-
Brunel/Fraud_Comapnies_only.csv')

fraud_companies_df.head()

# Convert CIK Numbers to integers in both datasets for
consistency
fraud_companies_df['CIK Number'] = fraud_companies_df['CIK
Number'].astype(int)
all_data['cik'] = all_data['cik'].astype(int)

# Find the intersection of the CIK numbers in both datasets
common_ciks = set(fraud_companies_df['CIK
Number']).intersection(set(all_data['cik']))

# Print the number of common CIK numbers
print(f"Number of common CIKs: {len(common_ciks)}")

common_rows = all_data[all_data['cik'].isin(common_ciks)]

# Now, common_rows contains only the rows from all_data where the
CIK number is also present in fraud_companies_df
```

```
# Create a list of CIK numbers of all the companies in the
fraud_companies_df
fraud_cik_list = fraud_companies_df['CIK Number'].tolist()
# Convert CIK numbers in all_data to integers
all_data['cik'] = all_data['cik'].astype(int)

# Create a new DataFrame for non-fraud companies
non_fraud_df = all_data[~all_data['cik'].isin(fraud_cik_list)]

non_fraud_df.head()
# Get the list of CIK numbers in non_fraud_df
non_fraud_cik_list = non_fraud_df['cik'].unique().tolist()

# Find the intersection of the CIK numbers in both datasets
common_ciks =
set(non_fraud_cik_list).intersection(set(fraud_cik_list))

# Print the common CIK numbers
print(f"Common CIKs: {common_ciks}")

# No common CIKs are present in the non-fraud companies dataset
# Label the non fraud comapnies as "No"
non_fraud_df['Fraud'] = 'No'
non_fraud_df['Fraud'].value_counts()
print(non_fraud_df.columns)
# Convert 'year' column to string type
non_fraud_df = non_fraud_df.copy()
non_fraud_df['year'] = non_fraud_df['year'].astype(str)

# Define a custom function for joining strings
def join_strings(series):
    return ' '.join([str(item) for item in series])

# Group by 'cik' and aggregate
merged_non_fraud_df = non_fraud_df.groupby('cik').agg({
    'year': lambda x: ', '.join(x.unique()),  # join unique years
    'filename': join_strings,  # join strings
    'section_1': join_strings,
    'section_1A': join_strings,
    'section_1B': join_strings,
    'section_2': join_strings,
    'section_3': join_strings,
    'section_4': join_strings,
    'section_5': join_strings,
    'section_6': join_strings,
    'section_7': join_strings,
    'section_7A': join_strings,
    'section_8': join_strings,
    'section_9': join_strings,
    'section_9A': join_strings,
    'section_9B': join_strings,
    'section_10': join_strings,
```

```
    'section_11': join_strings,
    'section_12': join_strings,
    'section_13': join_strings,
    'section_14': join_strings,
    'section_15': join_strings,
})

# Reset the index to make 'cik' a normal column
merged_non_fraud_df.reset_index(inplace=True)
# Now label the dataframe with No in the fraud column and merge
into fraud companies dataset
merged_non_fraud_df['Fraud'] = 'No'
merged_non_fraud_df['Fraud'].value_counts()
# Randomly sample 85 rows from merged_non_fraud_df
sampled_non_fraud_df = merged_non_fraud_df.sample(n=85,
random_state=1)

# Display the first few rows of the sampled DataFrame
sampled_non_fraud_df.head()
sampled_non_fraud_df['Fraud'].value_counts()
# Concatenate fraud_companies_df and sampled_non_fraud_df
merged_df = pd.concat([fraud_companies_df, sampled_non_fraud_df])

# Shuffle the rows of merged_df
merged_df = merged_df.sample(frac=1, random_state=1)


pd.set_option('display.max_columns', None)  # None means
unlimited

# Now, when you display the DataFrame, all columns should be
shown
print(merged_df.head())
merged_df.to_csv("C:/DissData/Dissertation-
Brunel/Final_Dataset.csv")
merged_df['Fraud'].value_counts()
merged_df.shape
# here total 170 companies are thier in which 85 are frauds and
other 85 are non-fraud companies

# Data Cleaning
### Removing Unnecessary Columns
# Create a new DataFrame with only the specified columns
cleaned_df = merged_df[["section_14", "section_7", "section_7A",
"section_8", "section_15", "Fraud"]]
```

```
# Display the first few rows of the cleaned DataFrame
print(cleaned_df.head())
### Removing non-textual content
# Replace 'NaN' with an empty string
cleaned_df = cleaned_df.replace('NaN', '', regex=True)
### Lowercasing
# Convert all text to lowercase
cleaned_df = cleaned_df.applymap(lambda s: s.lower() if type(s)
== str else s)
cleaned_df['section_15'][0]
### Remove special characters, punctuation, newline characters &
newline characters
import pandas as pd
import re
import string
from nltk.corpus import stopwords

stop = stopwords.words('english')

# Define a function to clean the text
def clean_text(text):
    # Remove special characters and punctuation
    text = text.translate(str.maketrans('', '',
string.punctuation))

    # Remove newline characters
    text = text.replace('\n', ' ')

    # Remove extra white spaces
    text = re.sub(' +', ' ', text)

    # Remove stop words
    stop = stopwords.words('english')
    text = ' '.join([word for word in text.split() if word not in
stop])

    # Lowercasing
    text = text.lower()

    return text

# Apply the clean_text function to every text column
text_columns = ['section_14', 'section_7', 'section_7A',
'section_8', 'section_15']  # replace with your actual column
names
for col in text_columns:
    cleaned_df[col] = cleaned_df[col].apply(clean_text)

# Data preprocessing
cleaned_df.head()
cleaned_df['Fillings'] = cleaned_df[cleaned_df.columns[:-
```

```
# Keep only the 'Fillings' and 'Fraud' columns
cleaned_df = cleaned_df[['Fillings', 'Fraud']]
cleaned_df.head()
import pandas as pd
from collections import OrderedDict

# Function to remove redundant sentences
def remove_redundant_sentences(text):
    sentences = text.split('. ')
    unique_sentences = OrderedDict.fromkeys(sentences)
    cleaned_text = '. '.join(unique_sentences.keys())
    return cleaned_text

# Apply the function to the 'Fillings' column
cleaned_df['Fillings'] =
cleaned_df['Fillings'].apply(remove_redundant_sentences)
import re

def remove_redundant_words(text):
    # Use a regular expression to replace repeated words with a
single occurrence
    cleaned_text = re.sub(r'\b(\w+)( \1\b)+', r'\1', text)

    return cleaned_text

cleaned_df['Fillings'] =
cleaned_df['Fillings'].apply(remove_redundant_words)

print(cleaned_df)

#cleaned_df.to_csv("C:/DissData/Dissertation-
Brunel/Final_Dataset.csv", index= False)
```

Step 3: Final Data Preparation (Code 3)

1. **Data Matching**:
    - The CIK numbers from the fraud list are matched with those in the Edgar corpus.
    - This is done using pandas' **isin** function, which filters the DataFrame based on a condition.
2. **Random Sampling**:
    - 85 non-fraud companies are randomly selected using **numpy**'s **random.choice** function.
    - This is done to balance the dataset, ensuring that the machine learning model is not biased.

3. **Final DataFrame**:
    - The fraud and non-fraud companies are appended to create a final DataFrame.
    - This DataFrame is balanced and can be used for training machine learning models.

```
import pandas as pd
from datasets import load_dataset
import numpy as np

# Load the dataset from the hugging face dataset
dataset = load_dataset('c3po-ai/edgar-corpus')

# Convert the splits to DataFrames
df_train = pd.DataFrame(dataset['train'])
df_validation = pd.DataFrame(dataset['validation'])
df_test = pd.DataFrame(dataset['test'])

# Concatenate all splits
all_data = pd.concat([df_train, df_validation, df_test])

# Convert 'cik' to string
all_data['cik'] = all_data['cik'].astype(str)

# Load the fraud companies
fraud_companies = pd.read_csv("C:/DissData/Dissertation-
Brunel/fraud_companies_with_cik.csv")
fraud_companies['CIK Number'] = fraud_companies['CIK
Number'].astype(str)

df1 = all_data

df2 = fraud_companies

matching_cik = df1['cik'].isin(df2['CIK Number'])

# Matching ciks with the our dataset with the hugging face
dataset

df1_with_matching_cik = df1[matching_cik]
df1_with_non_matching_cik = df1[~matching_cik]
```

Code 3:

```
df1_with_matching_cik['cik'].nunique()     # We got 85 unique
ciks
df1_with_non_matching_cik['cik'].nunique() # We got 37924 unique
ciks

# Filter df2 to only those rows where 'CIK Number' is in 'cik' of
df1
df2_matching_with_df1 = df2[df2['CIK Number'].isin(df1['cik'])]

df2 = df2_matching_with_df1 # df2 is now the same as
df1_with_matching_cik

# Checking If there are duplicat cik in our final dataset
duplicates_df2 = df2.duplicated(subset=['CIK Number'], keep =
False)
num_duplicates_df2 = duplicates_df2.sum()

duplicates_df2 = df2.duplicated(subset=['CIK Number'],
keep=False)
duplicate_rows_df2 = df2[duplicates_df2]
print(duplicate_rows_df2) # we got some duplicate ciks

df2.drop([124,81,206,215,98,195], inplace=True) # dropping the
(124,81,206,215, 98 and 195)th rows as they were redundunt

#df2.to_csv("Final_company_with_cik.csv", index= False)

merged_df = df1.merge(df2, how='inner', left_on='cik',
right_on='CIK Number')

merged_df['Fraud'] = 'Yes'

# Find all unique cik in df1 that are not in df2
cik_df1 = set(df1['cik'].unique())
cik_df2 = set(df2['CIK Number'].unique())
cik_df1_not_in_df2 = list(cik_df1 - cik_df2)

# Randomly select 85 unique companies
random_cik = np.random.choice(cik_df1_not_in_df2, 85,
replace=False)

# Select all rows from df1 that belong to these companies
new_rows = df1[df1['cik'].isin(random_cik)]

# Add 'Fraud' column and set it to 'No'
new_rows = new_rows.assign(Fraud='No')

# Append these rows to the final DataFrame
final_df = merged_df.append(new_rows, ignore_index=True)
```

APPENDIX B2 (Data Description)

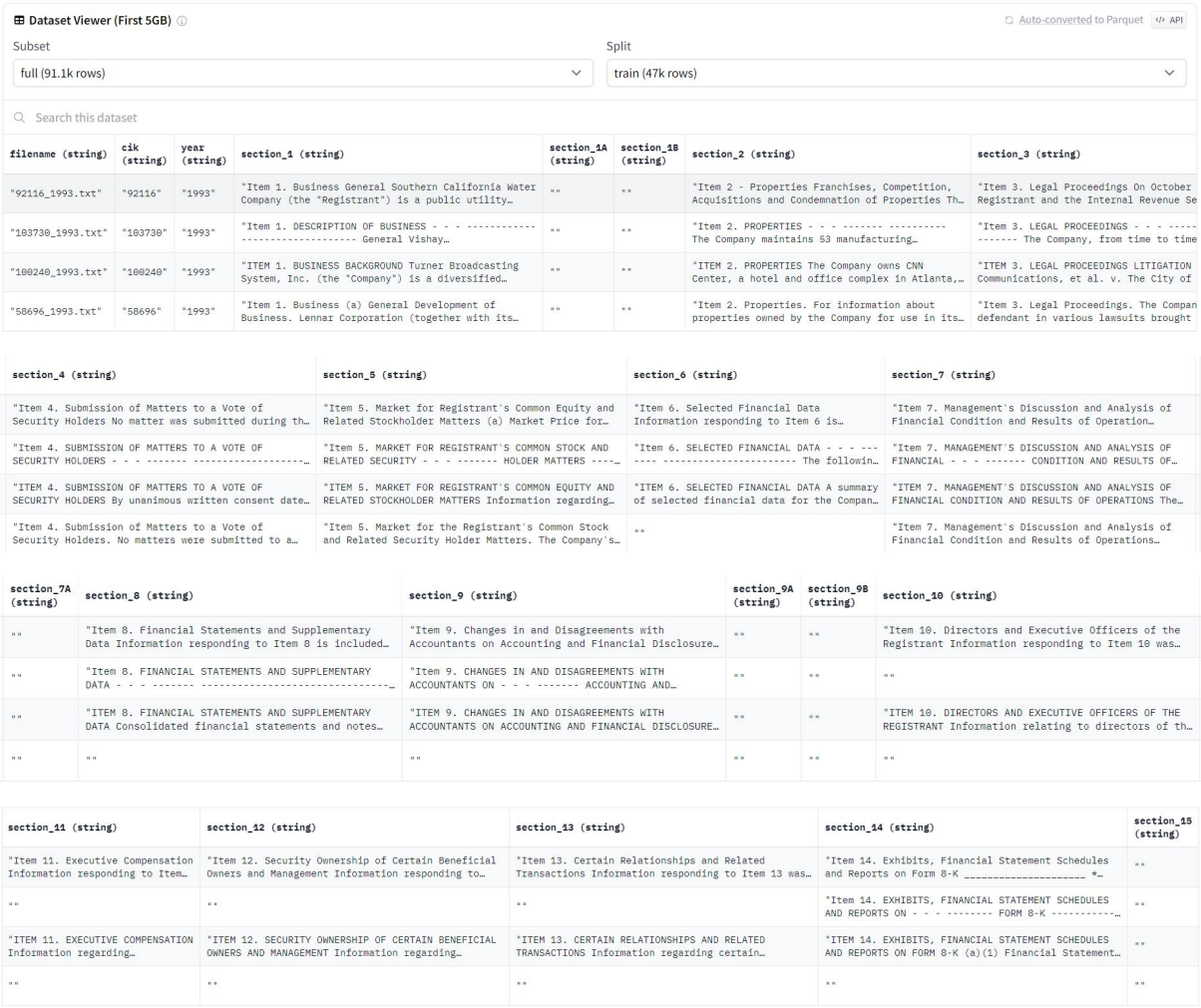
Fraud_companies_with_cik.csv data file would look like this:

This file is made to match the fraud companies' data that we got from the SEC website to the CIK numbers because the company was not consistent with the EDGAR database, we took from HuggingFace Website. Therefore, there was a need to establish a unique and consistent number that could represent each company uniquely so that we can match it directly with the EDGAR database and will be able to extract the 10-K fillings from it.

	A	B
1	**Company Name**	**CIK Number**
2	Gartner, Inc	749251
3	Koninklijke Philips N.V.	313216
4	Frankâ€™s International N.V.	1575828
5	Flutter Entertainment plc, as successor-in-interest to The Stars Group, Inc. -	1635327
6	Rio Tinto plc	863064
7	Honeywell International Inc.	773840
8	ABB	1381406
9	Oracle Corporation	1162378
10	Gol Intelligent Airlines Inc.	1291733
11	Tenaris	1190723
12	Stericycle, Inc.	861878
13	KT Corporation	1162378
14	Credit Suisse	824468
15	WPP plc	806968
16	Amec Foster Wheeler Ltd.	1130385
17	Asante Berko	1501393
18	Deutsche Bank AG	931622
19	Goldman Sachs Group, Inc.	886982
20	J&F Investimentos, S.A.	1438823
21	Herbalife Nutrition, Ltd.	1180262
22	World Acceptance Corp.	108385
23	Alexion Pharmaceuticals	899866

Fig 1, Fraud Companies list with CIK

To prepare the dataset to fine-tune models for this research is taken from open-source platform which contains all the companies 10-K fillings of all the years as well as CIK number which is the unique identifier. The dataset described in the figure 2 contains first four rows of the dataset and all the columns present in the dataset.



Dataset Viewer (First 5GB) ⓘ — Auto-converted to Parquet — API

Subset: full (91.1k rows) — Split: train (47k rows)

Search this dataset

filename (string)	cik (string)	year (string)	section_1 (string)	section_1A (string)	section_1B (string)	section_2 (string)	section_3 (string)
"92116_1993.txt"	"92116"	"1993"	"Item 1. Business General Southern California Water Company (the "Registrant") is a public utility…	""	""	"Item 2 - Properties Franchises, Competition, Acquisitions and Condemnation of Properties Th…	"Item 3. Legal Proceedings On October Registrant and the Internal Revenue Se
"103730_1993.txt"	"103730"	"1993"	"Item 1. DESCRIPTION OF BUSINESS - - - ---------- ---------- General Vishay…	""	""	"Item 2. PROPERTIES - - - ------- ---------- The Company maintains 53 manufacturing…	"Item 3. LEGAL PROCEEDINGS - - - ------ ------- The Company, from time to time
"100240_1993.txt"	"100240"	"1993"	"ITEM 1. BUSINESS BACKGROUND Turner Broadcasting System, Inc. (the "Company") is a diversified…	""	""	"ITEM 2. PROPERTIES The Company owns CNN Center, a hotel and office complex in Atlanta,…	"ITEM 3. LEGAL PROCEEDINGS LITIGATION Communications, et al. v. The City of
"58696_1993.txt"	"58696"	"1993"	"Item 1. Business (a) General Development of Business. Lennar Corporation (together with its…	""	""	"Item 2. Properties. For information about properties owned by the Company for use in its…	"Item 3. Legal Proceedings. The Compan defendant in various lawsuits brought

section_4 (string)	section_5 (string)	section_6 (string)	section_7 (string)
"Item 4. Submission of Matters to a Vote of Security Holders No matter was submitted during th…	"Item 5. Market for Registrant's Common Equity and Related Stockholder Matters (a) Market Price for…	"Item 6. Selected Financial Data Information responding to Item 6 is…	"Item 7. Management's Discussion and Analysis of Financial Condition and Results of Operation…
"Item 4. SUBMISSION OF MATTERS TO A VOTE OF SECURITY HOLDERS - - - ------- ----------…	"Item 5. MARKET FOR REGISTRANT'S COMMON STOCK AND RELATED SECURITY - - - ------- HOLDER MATTERS ----…	"Item 6. SELECTED FINANCIAL DATA - - - --- ---- ---------- The followin…	"Item 7. MANAGEMENT'S DISCUSSION AND ANALYSIS OF FINANCIAL - - - ------- CONDITION AND RESULTS OF…
"ITEM 4. SUBMISSION OF MATTERS TO A VOTE OF SECURITY HOLDERS By unanimous written consent date…	"ITEM 5. MARKET FOR REGISTRANT'S COMMON EQUITY AND RELATED STOCKHOLDER MATTERS Information regarding…	"ITEM 6. SELECTED FINANCIAL DATA A summary of selected financial data for the Compan…	"ITEM 7. MANAGEMENT'S DISCUSSION AND ANALYSIS OF FINANCIAL CONDITION AND RESULTS OF OPERATIONS The…
"Item 4. Submission of Matters to a Vote of Security Holders. No matters were submitted to a…	"Item 5. Market for the Registrant's Common Stock and Related Security Holder Matters. The Company's…	""	"Item 7. Management's Discussion and Analysis of Financial Condition and Results of Operations…

section_7A (string)	section_8 (string)	section_9 (string)	section_9A (string)	section_9B (string)	section_10 (string)
""	"Item 8. Financial Statements and Supplementary Data Information responding to Item 8 is included…	"Item 9. Changes in and Disagreements with Accountants on Accounting and Financial Disclosure…	""	""	"Item 10. Directors and Executive Officers of the Registrant Information responding to Item 10 was…
""	"Item 8. FINANCIAL STATEMENTS AND SUPPLEMENTARY DATA - - - ------- ----------…	"Item 9. CHANGES IN AND DISAGREEMENTS WITH ACCOUNTANTS ON - - - ------- ACCOUNTING AND…	""	""	""
""	"ITEM 8. FINANCIAL STATEMENTS AND SUPPLEMENTARY DATA Consolidated financial statements and notes…	"ITEM 9. CHANGES IN AND DISAGREEMENTS WITH ACCOUNTANTS ON ACCOUNTING AND FINANCIAL DISCLOSURE…	""	""	"ITEM 10. DIRECTORS AND EXECUTIVE OFFICERS OF THE REGISTRANT Information relating to directors of th…
""	""	""	""	""	""

section_11 (string)	section_12 (string)	section_13 (string)	section_14 (string)	section_15 (string)
"Item 11. Executive Compensation Information responding to Item…	"Item 12. Security Ownership of Certain Beneficial Owners and Management Information responding to…	"Item 13. Certain Relationships and Related Transactions Information responding to Item 13 was…	"Item 14. Exhibits, Financial Statement Schedules and Reports on Form 8-K _____ *…	""
""	""	""	"Item 14. EXHIBITS, FINANCIAL STATEMENT SCHEDULES AND REPORTS ON - - - -------- FORM 8-K ----------…	""
"ITEM 11. EXECUTIVE COMPENSATION Information regarding…	"ITEM 12. SECURITY OWNERSHIP OF CERTAIN BENEFICIAL OWNERS AND MANAGEMENT Information regarding…	"ITEM 13. CERTAIN RELATIONSHIPS AND RELATED TRANSACTIONS Information regarding certain…	"ITEM 14. EXHIBITS, FINANCIAL STATEMENT SCHEDULES AND REPORTS ON FORM 8-K (a)(1) Financial Statement…	""
""	""	""	""	""

Fig 2, Edgar Dataset Corpus

The dataset is structured as a series of annual reports or filings for various companies. Each row of the dataset represents a unique filing, with the following columns:

1. **filename**: The name of the file from which the data for the row is taken.
2. **cik**: A unique identifier for the company making the filing, known as the Central Index Key (CIK).
3. **year**: The year in which the filing was made.
4. **section_1** to **section_15**: These columns appear to contain text data from different sections of the filing. The content of these sections can vary, but they typically contain information about the company's business, properties, legal proceedings, directors, and financial information, among other things.

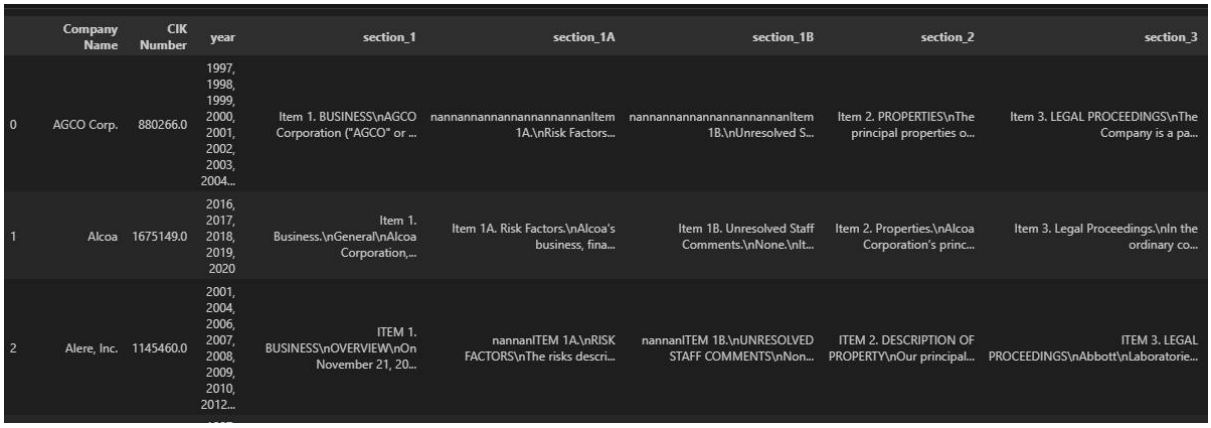
5. **Company Name**: The name of the company making the filing.
6. **CIK Number**: This seems to be a duplicate of the **cik** column.

Here's a brief description of what each section in a typical 10-K filing usually contains:

1. **section_1 (Business):** This section provides an overview of the company's main operations, its products or services, and its strategy. It may also include a history of the company and a list of its significant properties.
2. **section_1A (Risk Factors):** Here, the company outlines the most significant risks that could affect its business. These risks could be specific to the company or its industry, or they could be more general, such as risks related to the economy or legal environment.
3. **section_1B (Unresolved Staff Comments):** This section includes any comments received from the SEC about the company's filings that have not been resolved.
4. **section_2 (Properties):** This section provides details about the significant properties the company owns.
5. **section_3 (Legal Proceedings):** Here, the company describes any significant legal proceedings it is involved in.
6. **section_4 (Mine Safety Disclosures):** This section is for mining companies to disclose information about mine safety.
7. **section_5 (Market for Registrant's Common Equity, Related Stockholder Matters and Issuer Purchases of Equity Securities):** This section provides information about the company's equity securities, such as its common stock.
8. **section_6, 7 (Management's Discussion and Analysis of Financial Condition and Results of Operations - MD&A):** This is one of the most important sections, where management discusses the company's financial performance and condition, significant business trends, risks, and the outlook.
9. **section_7A (Quantitative and Qualitative Disclosures about Market Risk):** In this section, the company discusses the risks or uncertainties it faces in the markets in which it operates.
10. **section_8 (Financial Statements and Supplementary Data):** This section contains the company's audited financial statements, including the income statement, balance sheet, and cash flow statement.
11. **section_9, 9A, 9B (Changes in and Disagreements with Accountants on Accounting and Financial Disclosure):** This section contains any changes in the company's accountants, disagreements on accounting and financial disclosures, and management's assessment of the company's internal control over financial reporting.
12. **section_10 (Directors, Executive Officers and Corporate Governance):** This section provides information about the company's directors and executive officers.
13. **section_11 (Executive Compensation):** Here, the company provides a detailed breakdown of its executive compensation.

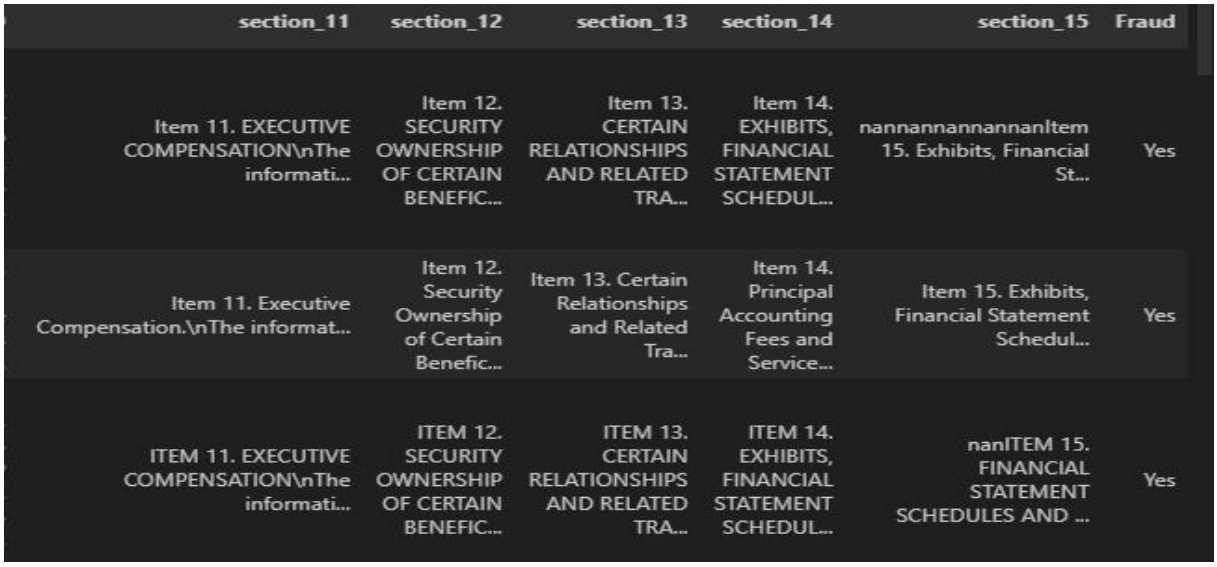
14. **section_12 (Security Ownership of Certain Beneficial Owners and Management and Related Stockholder Matters):** This section contains information about the ownership of the company's securities by certain beneficial owners and management.
15. **section_13 (Certain Relationships and Related Transactions, and Director Independence):** This section provides information about certain relationships and related transactions.
16. **section_14 (Principal Accountant Fees and Services):** This section provides information about the fees paid to the company's principal accountant for various services.
17. **section_15 (Exhibits, Financial Statement Schedules):** This section includes additional financial schedules and exhibits that provide more detailed information to support the main filing.

Fraud_companies_only.csv contains all the fillings of **fraudulent companies** with the years, CIK numbers and all the sections in the 10-K fillings which is labelled with the "Yes" because all the companies have done some financial frauds in the past. The dataset is looked like given in the figure 3.



	Company Name	CIK Number	year	section_1	section_1A	section_1B	section_2	section_3
0	AGCO Corp.	880266.0	1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004...	Item 1. BUSINESS\nAGCO Corporation ("AGCO" or ...	nannannannannannannannanItem 1A.\nRisk Factors...	nannannannannannannannanItem 1B.\nUnresolved S...	Item 2. PROPERTIES\nThe principal properties o...	Item 3. LEGAL PROCEEDINGS\nThe Company is a pa...
1	Alcoa	1675149.0	2016, 2017, 2018, 2019, 2020	Item 1. Business.\nGeneral\nAlcoa Corporation,...	Item 1A. Risk Factors.\nAlcoa's business, fina...	Item 1B. Unresolved Staff Comments.\nNone.\nIt...	Item 2. Properties.\nAlcoa Corporation's princ...	Item 3. Legal Proceedings.\nIn the ordinary co...
2	Alere, Inc.	1145460.0	2001, 2004, 2006, 2007, 2008, 2009, 2010, 2012...	ITEM 1. BUSINESS\nOVERVIEW\nOn November 21, 20...	nannanITEM 1A.\nRISK FACTORS\nThe risks descri...	nannanITEM 1B.\nUNRESOLVED STAFF COMMENTS\nNon...	ITEM 2. DESCRIPTION OF PROPERTY\nOur principal...	ITEM 3. LEGAL PROCEEDINGS\nAbbott\nLaboratorie...

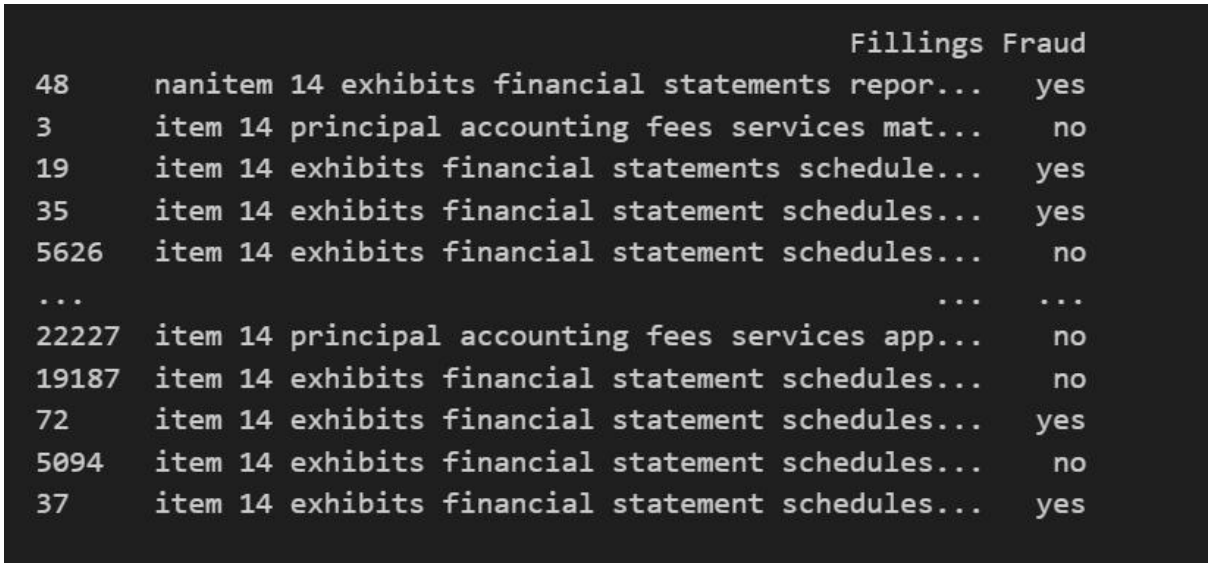
	section_3	section_4	section_5	...	section_9	section_9A	section_9B	section_10
Item 3. LEGAL PROCEEDINGS\nThe Company is a pa...	Item 4. SUBMISSION OF MATTERS TO A VOTE OF SEC...	Item 5. MARKET FOR REGISTRANT'S COMMON EQUITY ...	...	Item 9. CHANGES IN AND DISAGREEMENTS WITH ACCO...	nannannannannannannanItem 9A.\nControls and Proce...	nannannannannannannannanItem 9B. Other Informatio...	Item 10. DIRECTORS AND EXECUTIVE OFFICERS OF R...
Item 3. Legal Proceedings.\nIn the ordinary co...	Item 4. Mine Safety Disclosures.\nThe informat...	Item 5. Market for Registrant's Common Equity,...	...	Item 9. Changes in and Disagreements with Acco...	Item 9A. Controls and Procedures.\n(a) Evaluat...	Item 9B. Other Information.\nNone.\nPART III\n...	Item 10. Directors, Executive Officers and Cor...
ITEM 3. LEGAL ROCEEDINGS\nAbbott\nLaboratorie...	ITEM 4. SUBMISSION OF MATTERS TO A VOTE OF SEC...	ITEM 5. MARKET FOR REGISTRANT'S COMMON EQUITY ...	...	ITEM 9. CHANGES IN AND DISAGREEMENTS WITH ACCO...	nanITEM 9A. CONTROLS AND PROCEDURES\nManagemen...	nanITEM 9B. OTHER INFORMATION\nNot applicable....	ITEM 10. DIRECTORS AND EXECUTIVE OFFICERS OF T...



	section_11	section_12	section_13	section_14	section_15	Fraud
	Item 11. EXECUTIVE COMPENSATION\nThe informati...	Item 12. SECURITY OWNERSHIP OF CERTAIN BENEFIC...	Item 13. CERTAIN RELATIONSHIPS AND RELATED TRA...	Item 14. EXHIBITS, FINANCIAL STATEMENT SCHEDUL...	nannannannannanItem 15. Exhibits, Financial St...	Yes
	Item 11. Executive Compensation.\nThe informat...	Item 12. Security Ownership of Certain Benefic...	Item 13. Certain Relationships and Related Tra...	Item 14. Principal Accounting Fees and Service...	Item 15. Exhibits, Financial Statement Schedul...	Yes
	ITEM 11. EXECUTIVE COMPENSATION\nThe informati...	ITEM 12. SECURITY OWNERSHIP OF CERTAIN BENEFIC...	ITEM 13. CERTAIN RELATIONSHIPS AND RELATED TRA...	ITEM 14. EXHIBITS, FINANCIAL STATEMENT SCHEDUL...	nanITEM 15. FINANCIAL STATEMENT SCHEDULES AND ...	Yes

Fig 3, Edgar Dataset Corpus

Final_Dataset.csv is the final dataset after selecting all the required section based on the requirements (Explained in the Methodology section of this dissertation) and the data cleaning and preprocessing is done in the final step. Some of the preprocessing steps will be done in the accordance with requirements of each algorithm used in this research. Few rows of the final dataset are presented in the figure 4.



	Fillings	Fraud
48	nanitem 14 exhibits financial statements repor...	yes
3	item 14 principal accounting fees services mat...	no
19	item 14 exhibits financial statements schedule...	yes
35	item 14 exhibits financial statement schedules...	yes
5626	item 14 exhibits financial statement schedules...	no
...	...	...
22227	item 14 principal accounting fees services app...	no
19187	item 14 exhibits financial statement schedules...	no
72	item 14 exhibits financial statement schedules...	yes
5094	item 14 exhibits financial statement schedules...	no
37	item 14 exhibits financial statement schedules...	yes

Fig 4, Edgar Dataset Corpus

Final Dataset description:

1. **Fillings:** Contains the 2 sections of 10-K fillings of all the years from the 1990 to 2020. The 2 sections are MD&A section and financial statement section of all the companies in which 85 are fraudulent and other 85 are non-fraudulent.
2. **Fraud:** The “Yes” represents the fraudulent company and “No” represents the non-fraudulent company.

APPENDIX B3 (Model Codes with Outputs)

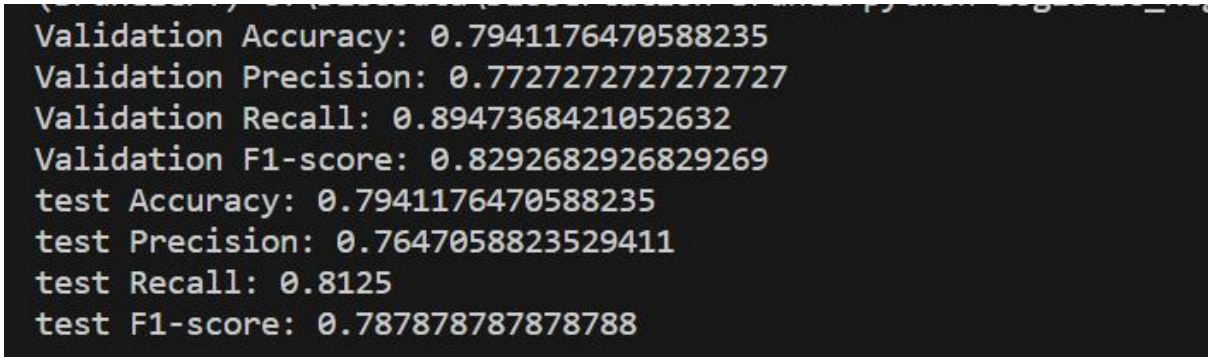
Logistic Regression

The code is a Python script that uses Logistic Regression to identify fraudulent activities based on text data. It first imports necessary libraries and loads a dataset from a CSV file. The dataset is split into training, validation, and test sets. The text data is then transformed into numerical form using TF-IDF Vectorization. It is trained on the training set and evaluated on both the validation and test sets using various metrics like accuracy, precision, recall, and F1-score. Finally, the evaluation results are printed to the console.

```
# Import necessary libraries
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, roc_auc_score
import pandas as pd
from sklearn.preprocessing import LabelEncoder

# Load the dataset from a CSV file
file_path = 'Final_Dataset.csv'
dataset = pd.read_csv(file_path)

# Extract the text data and labels from the dataset
text_data = dataset['Fillings']
labels = dataset['Fraud']

# Encode the labels using LabelEncoder
label_encoder = LabelEncoder()
labels = label_encoder.fit_transform(labels)
positive_label = label_encoder.transform(['yes'])[0]

# Split the data into training, validation, and test sets
X_temp, X_test, y_temp, y_test = train_test_split(text_data, labels,
test_size=0.2, random_state=42)
X_train, X_val, y_train, y_val = train_test_split(X_temp, y_temp,
test_size=0.25, random_state=42)

# Preprocess the text data using TF-IDF Vectorizer
tfidf_vectorizer = TfidfVectorizer()
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_val_tfidf = tfidf_vectorizer.transform(X_val)
X_test_tfidf = tfidf_vectorizer.transform(X_test)

# Train the Logistic Regression model
logistic_model = LogisticRegression()
logistic_model.fit(X_train_tfidf, y_train)
```

```
# Evaluate the model on the validation set
y_val_pred = logistic_model.predict(X_val_tfidf)
val_accuracy = accuracy_score(y_val, y_val_pred)
val_precision = precision_score(y_val, y_val_pred,
pos_label=positive_label)
val_recall = recall_score(y_val, y_val_pred, pos_label=positive_label)
val_f1 = f1_score(y_val, y_val_pred, pos_label=positive_label)


# Evaluate the model on the test set
y_test_pred = logistic_model.predict(X_test_tfidf)
test_accuracy = accuracy_score(y_test, y_test_pred)
test_precision = precision_score(y_test, y_test_pred,
pos_label=positive_label)
test_recall = recall_score(y_test, y_test_pred,
pos_label=positive_label)
test_f1 = f1_score(y_test, y_test_pred, pos_label=positive_label)

# Print evaluation metrics
print("Validation Accuracy:", val_accuracy)
print("Validation Precision:", val_precision)
print("Validation Recall:", val_recall)
print("Validation F1-score:", val_f1)
print("Test Accuracy:", test_accuracy)
print("Test Precision:", test_precision)
print("Test Recall:", test_recall)
```

OUTPUT:



Random Forest

The Python script used the Random Forest Classifier from the scikit-learn library to classify companies as fraudulent or non-fraudulent based on their financial filings. The script starts by loading a dataset from a CSV file using pandas, where the 'Fillings' column contains text data, and the 'Fraud' column contains labels. These labels are converted to numerical form (1 for 'yes' and 0 for 'no'). The dataset is then divided into training, validation, and test sets, allocating 60% for training, 20% for validation, and 20% for testing. The text data is vectorized using the Term Frequency-Inverse Document Frequency (TF-IDF) technique, limiting the number of features to 5000 for computational efficiency. A Random Forest model with 100 estimators is trained on the vectorized training set. The model's performance is evaluated on both the validation and test sets using metrics like accuracy, precision, recall, and F1-score, and the results are printed out.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score
from sklearn.model_selection import train_test_split
import pandas as pd

# Load Dataset

file_path = "Final_Dataset.csv"
dataset = pd.read_csv(file_path)
text_data = dataset['Fillings']
labels = dataset['Fraud'].replace({'yes': 1, 'no': 0})

# Split Data into Train, Validation, and Test Sets

X_train, X_temp, y_train, y_temp = train_test_split(text_data,
labels, test_size=0.4, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp,
test_size=0.5, random_state=42)

# Text Vectorization using TF-IDF

tfidf_vectorizer = TfidfVectorizer(max_features=5000)
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_val_tfidf = tfidf_vectorizer.transform(X_val)
X_test_tfidf = tfidf_vectorizer.transform(X_test)

# Create and Train Random Forest Classifier
random_forest_model = RandomForestClassifier(n_estimators=100,
random_state=42)
random_forest_model.fit(X_train_tfidf, y_train)
```

```
# Evaluate on Validation Data
y_val_pred = random_forest_model.predict(X_val_tfidf)
val_accuracy = accuracy_score(y_val, y_val_pred)
val_precision = precision_score(y_val, y_val_pred)
val_recall = recall_score(y_val, y_val_pred)
val_f1_score = f1_score(y_val, y_val_pred)

# Evaluate on Test Data

y_test_pred = random_forest_model.predict(X_test_tfidf)
test_accuracy = accuracy_score(y_test, y_test_pred)
test_precision = precision_score(y_test, y_test_pred)
test_recall = recall_score(y_test, y_test_pred)
test_f1_score = f1_score(y_test, y_test_pred)

# Print Results

print("Validation Accuracy:", val_accuracy)
print("Validation Precision:", val_precision)
print("Validation Recall:", val_recall)
print("Validation F1-score:", val_f1_score)
print("Test Accuracy:", test_accuracy)
print("Test Precision:", test_precision)
print("Test Recall:", test_recall)
print("Test F1-score:", test_f1_score)
```

OUTPUT:

```
Validation Accuracy: 0.9117647058823529
Validation Precision: 0.8571428571428571
Validation Recall: 1.0
Validation F1-score: 0.923076923076923
Test Accuracy: 0.9411764705882353
Test Precision: 1.0
Test Recall: 0.8888888888888888
Test F1-score: 0.9411764705882353
```

Support Vector Machine (SVM)

The Python script uses the Support Vector Machine (SVM) classifier from the scikit-learn library to categorize companies as either fraudulent or non-fraudulent based on their financial filings. Initially, the script reads a dataset from a CSV file using the panda's library, where the 'Fillings' column contains the text data, and the 'Fraud' column contains the labels. The dataset is then partitioned into training, validation, and test sets, with 70% allocated for training, 15% for validation, and 15% for testing. The text data undergoes vectorization using the Term Frequency-Inverse Document Frequency (TF-IDF) technique. An SVM model with a linear kernel is trained on the vectorized training set. The model's performance is subsequently evaluated on both the validation and test sets using various metrics such as accuracy, precision, recall, and F1-score. The evaluation results are printed to the console for further analysis.

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score
from sklearn.model_selection import train_test_split
import pandas as pd

# Load your CSV dataset
file_path = 'Final_Dataset.csv'
dataset = pd.read_csv(file_path)

# Split the data into features and labels
X = dataset['Fillings']
y = dataset['Fraud']

# Split the data into training, validation, and test sets
X_train, X_temp, y_train, y_temp = train_test_split(X, y,
test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp,
test_size=0.5, random_state=42)

# Transform the text data using TF-IDF vectorizer
tfidf_vectorizer = TfidfVectorizer()
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_val_tfidf = tfidf_vectorizer.transform(X_val)
X_test_tfidf = tfidf_vectorizer.transform(X_test)

# Create and train the SVM model
svm_model = SVC(kernel='linear')
svm_model.fit(X_train_tfidf, y_train)
```

```
# Validate the model
y_val_pred = svm_model.predict(X_val_tfidf)
val_accuracy = accuracy_score(y_val, y_val_pred)
val_precision = precision_score(y_val, y_val_pred,
pos_label='yes')
val_recall = recall_score(y_val, y_val_pred, pos_label='yes')
val_f1 = f1_score(y_val, y_val_pred, pos_label='yes')

# Test the model
y_test_pred = svm_model.predict(X_test_tfidf)
test_accuracy = accuracy_score(y_test, y_test_pred)
test_precision = precision_score(y_test, y_test_pred,
pos_label='yes')
test_recall = recall_score(y_test, y_test_pred, pos_label='yes')
test_f1 = f1_score(y_test, y_test_pred, pos_label='yes')

print("Validation Accuracy:", val_accuracy)
print("Validation Precision:", val_precision)
print("Validation Recall:", val_recall)
print("Validation F1-score:", val_f1)
print("Test Accuracy:", test_accuracy)
print("Test Precision:", test_precision)
print("Test Recall:", test_recall)
print("Test F1-score:", test_f1)
```

OUTPUT:

```
Validation Accuracy: 0.84
Validation Precision: 0.75
Validation Recall: 0.9
Validation F1-score: 0.8181818181818182
Test Accuracy: 0.8461538461538461
Test Precision: 0.75
Test Recall: 1.0
Test F1-score: 0.8571428571428571
```

Artificial Neural Network (ANN)

The script employs an ANN using TensorFlow's Keras library to classify companies as fraudulent or non-fraudulent based on their financial filings. The dataset is read using pandas and the labels are binarized. The data is then split into training, validation, and test sets. Text data is vectorized using the TF-IDF technique. The ANN model consists of an input layer, two hidden layers with 'tanh' activation functions, and an output layer with a 'sigmoid' activation function. The model is compiled using the RMSprop optimizer and binary cross-entropy loss function. Training is performed with early stopping to prevent overfitting. The model's performance is evaluated on both the validation and test sets using metrics like accuracy, precision, recall, and F1-score, and the results are printed for

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping
from keras.regularizers import l2
from keras.models import Sequential
from keras.layers import Dense, Dropout


# Assuming you've loaded the data into df
df = pd.read_csv('Final_Dataset.csv')

# Convert labels to binary
le = LabelEncoder()
df['Fraud'] = le.fit_transform(df['Fraud'])

# Split data into train, validation, and test sets
train, temp = train_test_split(df, test_size=0.3,
random_state=42)
val, test = train_test_split(temp, test_size=0.5,
random_state=42)

# TF-IDF vectorization
vectorizer = TfidfVectorizer(max_features=5000)
X_train = vectorizer.fit_transform(train['Fillings']).toarray()
X_val = vectorizer.transform(val['Fillings']).toarray()
X_test = vectorizer.transform(test['Fillings']).toarray()

# Define the ANN Model:
# model = Sequential()
```

further analysis.

```
# model.add(Dense(128, activation='relu',
input_shape=(X_train.shape[1],)))
# model.add(Dropout(0.5))
# model.add(Dense(64, activation='relu'))
# model.add(Dropout(0.5))
# model.add(Dense(1, activation='sigmoid'))

# model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])

# Hyperparameter Change

# model = Sequential()

# # Input Layer
# model.add(Dense(512, activation='relu',
input_shape=(X_train.shape[1],)))
# model.add(Dropout(0.5))

# # Hidden Layer 1
# model.add(Dense(256, activation='relu'))
# model.add(Dropout(0.5))

# # Hidden Layer 2
# model.add(Dense(128, activation='relu'))
# model.add(Dropout(0.5))

# # Output Layer
# model.add(Dense(1, activation='sigmoid'))

# model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])


model = Sequential()

# Input Layer
model.add(Dense(256, activation='tanh',
input_shape=(X_train.shape[1],)))

# Hidden Layer 1
model.add(Dense(128, activation='tanh'))

# Hidden Layer 2
model.add(Dense(64, activation='tanh'))

# Output Layer
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='rmsprop', loss='binary_crossentropy',
metrics=['accuracy'])
```

```
# Train the Model:

early_stop = EarlyStopping(monitor='val_loss', patience=3)

history = model.fit(
    X_train, y_train,
    epochs=20,
    batch_size=32,
    validation_data=(X_val, y_val),
    callbacks=[early_stop]
)

# Evaluate the Model:

# Validation performance
y_val_pred = (model.predict(X_val) > 0.5).astype("int32")
val_accuracy = accuracy_score(y_val, y_val_pred)
val_precision = precision_score(y_val, y_val_pred)
val_recall = recall_score(y_val, y_val_pred)
val_f1 = f1_score(y_val, y_val_pred)

# Test performance
y_test_pred = (model.predict(X_test) > 0.5).astype("int32")
test_accuracy = accuracy_score(y_test, y_test_pred)
test_precision = precision_score(y_test, y_test_pred)
test_recall = recall_score(y_test, y_test_pred)
test_f1 = f1_score(y_test, y_test_pred)

print(f"Validation Accuracy: {val_accuracy:.4f}")
print(f"Validation Precision: {val_precision:.4f}")
print(f"Validation Recall: {val_recall:.4f}")
print(f"Validation F1-score: {val_f1:.4f}")
print(f"Test Accuracy: {test_accuracy:.4f}")
print(f"Test Precision: {test_precision:.4f}")
print(f"Test Recall: {test_recall:.4f}")
print(f"Test F1-score: {test_f1:.4f}")
```

OUTPUT: (ReLU)

```
Validation Accuracy: 0.8800
Validation Precision: 0.8182
Validation Recall: 0.9000
Validation F1-score: 0.8571
Test Accuracy: 0.8846
Test Precision: 0.8462
Test Recall: 0.9167
Test F1-score: 0.8800
```

OUTPUT: (Tanh)

```
Validation Accuracy: 0.8800
Validation Precision: 0.8182
Validation Recall: 0.9000
Validation F1-score: 0.8571
Test Accuracy: 0.9231
Test Precision: 0.8571
Test Recall: 1.0000
Test F1-score: 0.9231
```

Hierarchical Attention Network (HAN)

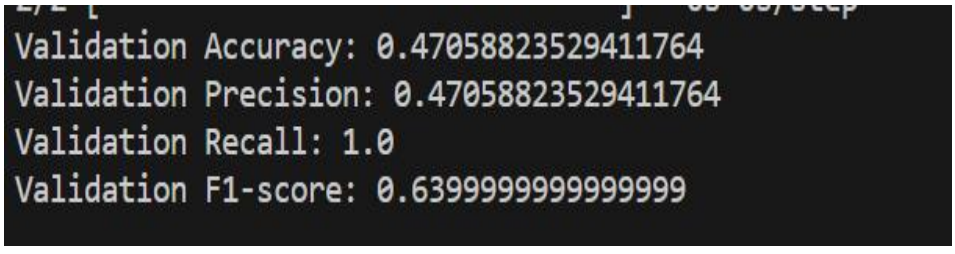
The script employs a Hierarchical Attention Network (HAN) to classify financial filings as either fraudulent or non-fraudulent. The data is read using pandas and tokenized at both the sentence and word levels using the Natural Language Toolkit (NLTK). The tokenized text is then padded to fit a predefined shape. The model consists of two levels of attention mechanisms: word-level and sentence-level. At the word-level, an LSTM layer followed by a Global Average Pooling layer is used to capture the importance of each word within a sentence. At the sentence-level, another LSTM layer followed by Global Average Pooling captures the importance of each sentence within a document. The model is compiled using the Adam optimizer and binary cross-entropy loss function. It is trained on a split of the data, and its performance is evaluated using metrics such as accuracy, precision, recall, F1-score, and AUC (Area Under the Curve). These metrics are printed out for further analysis.

```
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from nltk.tokenize import sent_tokenize, word_tokenize
from tensorflow.keras.layers import Embedding, Input, LSTM,
Dense, Attention, Flatten
from tensorflow.keras.models import Model
from sklearn.model_selection import train_test_split
from tensorflow.keras.layers import TimeDistributed
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, roc_auc_score
from keras.layers import GlobalAveragePooling1D
from nltk.tokenize import sent_tokenize
```

```
# Constants
MAX_NB_WORDS = 20000
EMBEDDING_DIM = 100
MAX_SENT_LENGTH = 50
MAX_SENTS = 15

# Reading the data
data = pd.read_csv('Final_Dataset.csv')
texts = data['Fillings'].tolist()
data['Fraud'] = data['Fraud'].map({'yes': 1, 'no': 0})
labels = data['Fraud'].astype(int).tolist()


# Tokenize sentences and words
texts = [[sent[:MAX_SENT_LENGTH] for sent in
sent_tokenize(text)[:MAX_SENTS]] for text in texts]

# Word Tokenization and Padding
tokenizer = Tokenizer(num_words=MAX_NB_WORDS)
tokenizer.fit_on_texts([word for sent in doc for word in sent]
for doc in texts)
data = np.zeros((len(texts), MAX_SENTS, MAX_SENT_LENGTH),
dtype='int32')

for i, sentences in enumerate(texts):
    for j, sent in enumerate(sentences):
        if j < MAX_SENTS:
            wordTokens = tokenizer.texts_to_sequences([sent])[0]
            k = 0
            for _, word in enumerate(wordTokens):
                if k < MAX_SENT_LENGTH:
                    data[i, j, k] = word
                    k = k + 1

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(data, labels,
test_size=0.2, random_state=42)

# Convert labels to arrays
y_train = np.array(y_train, dtype=np.int32)
y_test = np.array(y_test, dtype=np.int32)


# Word-level attention
word_input = Input(shape=(MAX_SENT_LENGTH,), dtype='int32')
word_sequences = Embedding(MAX_NB_WORDS,
EMBEDDING_DIM)(word_input)
word_lstm = LSTM(128, return_sequences=True)(word_sequences)
word_attention = GlobalAveragePooling1D()(word_lstm)
word_encoder = Model(word_input, word_attention
```

```
# Sentence-level attention
sent_input = Input(shape=(MAX_SENTS, MAX_SENT_LENGTH),
dtype='int32')
sent_encoder = TimeDistributed(word_encoder)(sent_input)
sent_lstm = LSTM(128, return_sequences=True)(sent_encoder)
sent_attention = GlobalAveragePooling1D()(sent_lstm)
preds = Dense(1, activation='sigmoid')(sent_attention)
model = Model(sent_input, preds)

# Compile and train the model
model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
model.fit(X_train, y_train, validation_data=(X_test, y_test),
epochs=10, batch_size=50)

# Predicting the classes for the test set
y_pred = model.predict(X_test)
y_pred_classes = (y_pred > 0.5).astype(int).flatten()
y_pred_probs = model.predict(X_test).flatten()

# Calculating the metrics
accuracy = accuracy_score(y_test, y_pred_classes)
precision = precision_score(y_test, y_pred_classes, pos_label=1)
recall = recall_score(y_test, y_pred_classes, pos_label=1)
f1 = f1_score(y_test, y_pred_classes, pos_label=1)
auc = roc_auc_score(y_test, y_pred_probs)

print("Validation Accuracy:", accuracy)
print("Validation Precision:", precision)
print("Validation Recall:", recall)
print("Validation F1-score:", f1)
print("Validation AUC:", auc)
```

OUTPUT:



GPT-2 with Attention

The script utilizes the GPT-2 model for sequence classification to identify fraudulent financial filings. It starts by reading a CSV dataset using pandas and encoding the labels. The GPT-2 tokenizer is then employed to tokenize and pad the text data. The script also includes attention masks to help the model focus on the relevant parts of the input. The data is split into training, validation, and test sets, and PyTorch's DataLoader is used to batch the data. The GPT-2 model is fine-tuned for sequence classification with two output labels. The Adam optimizer and a learning rate of 5e-5 are used for training. The model is trained for four epochs, and both training and validation metrics such as accuracy, precision, recall, and F1-score are calculated and printed. The script also includes a function to evaluate the model on any given dataset, returning the aforementioned metrics. The total time taken for training is also displayed, providing an idea of the computational cost.

```
import pandas as pd
from transformers import GPT2Tokenizer,
GPT2ForSequenceClassification
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score
import torch
from torch.utils.data import DataLoader, TensorDataset
import torch.nn.functional as F
import time

file_path = 'Final_Dataset.csv'
dataset = pd.read_csv(file_path)

## Preprocessing the data

# Encode Labels
label_encoder = LabelEncoder()
dataset['Fraud'] = label_encoder.fit_transform(dataset['Fraud'])

tokenizer = GPT2Tokenizer.from_pretrained('gpt2')
# Check if padding token is set, if not, set it
if tokenizer.pad_token is None:
    tokenizer.add_special_tokens({'pad_token': '[PAD]'})

#tokenizer.pad_token = tokenizer.eos_token
text_data = dataset['Fillings'].tolist()
encoded_inputs = tokenizer(text_data, padding='max_length',
truncation=True, max_length=512, return_tensors="pt")
attention_masks = encoded_inputs['attention_mask']
```

```
# Split into Training, Validation, and Test Sets
train_inputs, temp_inputs, train_labels, temp_labels,
train_masks, temp_masks = train_test_split(
    encoded_inputs['input_ids'], dataset['Fraud'],
attention_masks, test_size=0.4, random_state=42
)

val_inputs, test_inputs, val_labels, test_labels, val_masks,
test_masks = train_test_split(
    temp_inputs, temp_labels, temp_masks, test_size=0.5,
random_state=42
)

# Create DataLoader with attention masks
train_dataset = TensorDataset(train_inputs, train_masks,
torch.tensor(train_labels.values).long())
train_dataloader = DataLoader(train_dataset, batch_size=8,
shuffle=True)

val_dataset = TensorDataset(val_inputs, val_masks,
torch.tensor(val_labels.values).long())
val_dataloader = DataLoader(val_dataset, batch_size=8)

test_dataset = TensorDataset(test_inputs, test_masks,
torch.tensor(test_labels.values).long())
test_dataloader = DataLoader(test_dataset, batch_size=8)

# Load GPT-2 Model for Sequence Classification
model = GPT2ForSequenceClassification.from_pretrained('gpt2',
num_labels=2)
model.resize_token_embeddings(len(tokenizer) + 1)
model.config.pad_token_id = tokenizer.pad_token_id
optimizer = torch.optim.Adam(model.parameters(), lr=5e-5)

# Training Loop with attention masks
num_epochs = 4
total_start_time = time.time()

for epoch in range(num_epochs):
    start_time = time.time()
    model.train()
    train_loss = 0
    train_predictions = []
    train_true_labels = []

    for batch in train_dataloader:
        inputs, masks, labels = batch
        optimizer.zero_grad()
        outputs = model(inputs, attention_mask=masks,
labels=labels)
```

```
        loss = outputs.loss
        loss.backward()
        optimizer.step()
        train_loss += loss.item()
        logits = outputs.logits

        predictions = torch.argmax(F.softmax(logits, dim=1),
dim=1)
        train_predictions.extend(predictions.tolist())
        train_true_labels.extend(labels.tolist())

    # Training Metrics
    train_accuracy = accuracy_score(train_true_labels,
train_predictions)
    train_precision = precision_score(train_true_labels,
train_predictions)
    train_recall = recall_score(train_true_labels,
train_predictions)
    train_f1 = f1_score(train_true_labels, train_predictions)

    print(f"Epoch {epoch+1}/{num_epochs}")
    print(f"Training Loss: {train_loss/len(train_dataloader)}")
    print(f"Training Accuracy: {train_accuracy}")
    print(f"Training Precision: {train_precision}")
    print(f"Training Recall: {train_recall}")
    print(f"Training F1-score: {train_f1}")

    # Validation Loop
    model.eval()
    val_predictions = []
    val_true_labels = []
    with torch.no_grad():
        for batch in val_dataloader:
            inputs, masks, labels = batch
            outputs = model(inputs, attention_mask=masks)
            logits = outputs.logits
            predictions = torch.argmax(F.softmax(logits, dim=1),
dim=1)
            val_predictions.extend(predictions.tolist())
            val_true_labels.extend(labels.tolist())

    end_time = time.time()
    epoch_time = end_time - start_time
    print(f"Time taken for epoch {epoch+1}: {epoch_time:.2f}
seconds")
```

```
total_end_time = time.time()
total_time = total_end_time - total_start_time
print(f"Total training time: {total_time:.2f} seconds")

def evaluate(model, dataloader):
    model.eval()
    predictions, true_labels = [], []
with torch.no_grad():
        for batch in dataloader:
            inputs, masks, labels = batch
            outputs = model(inputs, attention_mask=masks)
            logits = outputs.logits
            preds = torch.argmax(logits, dim=1)
            predictions.extend(preds.tolist())
            true_labels.extend(labels.tolist())

    accuracy = accuracy_score(true_labels, predictions)
    precision = precision_score(true_labels, predictions,
average='weighted')
    recall = recall_score(true_labels, predictions,
average='weighted')
    f1 = f1_score(true_labels, predictions, average='weighted')

    print(f'Accuracy: {accuracy}')
    print(f'Precision: {precision}')
    print(f'Recall: {recall}')
    print(f'F1 Score: {f1}')

    return accuracy, precision, recall, f1
```

FinBERT

The script employs a pre-trained FinBERT model for sequence classification to detect fraudulent activities in financial filings. It starts by loading a CSV dataset and encoding the labels using scikit-learn's LabelEncoder. The FinBERT tokenizer is then used to tokenize and pad the text data, and attention masks are generated to help the model focus on the relevant parts of the input. The data is divided into training, validation, and test sets, and PyTorch's DataLoader is used to batch the data. The BERT model is fine-tuned for sequence classification with two output labels, using the Adam optimizer with a learning rate of 5e-5. The model is trained for three epochs initially, and the time taken for each epoch is displayed. The script also includes a second training loop for four epochs, calculating and displaying training and validation metrics such as accuracy, precision, recall, and F1-score. Finally, the model is evaluated on the test set, and performance metrics including ROC-AUC are printed. The total time taken for training is also reported, offering insights into the computational cost.

```
# Import Libraries
import pandas as pd
from transformers import BertTokenizer,
BertForSequenceClassification
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, roc_auc_score
import torch
from torch.utils.data import DataLoader, TensorDataset
import torch.nn.functional as F
import time

# Load Dataset
file_path = 'Final_Dataset.csv'
dataset = pd.read_csv(file_path)

# Encode Labels
label_encoder = LabelEncoder()
dataset['Fraud'] = label_encoder.fit_transform(dataset['Fraud'])

# Tokenize the Text Data
tokenizer = BertTokenizer.from_pretrained('yiyanghkust/finbert-
pretrain')
text_data = dataset['Fillings'].tolist()
max_length = 512
encoded_inputs = tokenizer(text_data, padding=True,
truncation=True, max_length=max_length, return_tensors="pt")
attention_masks = encoded_inputs['attention_mask']

# Split the Dataset into Training, Validation, and Test Sets
train_inputs, temp_inputs, train_labels, temp_labels,
train_masks, temp_masks = train_test_split(
    encoded_inputs['input_ids'], dataset['Fraud'],
attention_masks, test_size=0.4, random_state=42
)

val_inputs, test_inputs, val_labels, test_labels, val_masks,
test_masks = train_test_split(
    temp_inputs, temp_labels, temp_masks, test_size=0.5,
random_state=42
)

# Create DataLoader with attention masks
train_dataset = TensorDataset(train_inputs, train_masks,
torch.tensor(train_labels.values).long())
train_dataloader = DataLoader(train_dataset, batch_size=32,
shuffle=True)
```

```
val_dataset = TensorDataset(val_inputs, val_masks,
torch.tensor(val_labels.values).long())
val_dataloader = DataLoader(val_dataset, batch_size=32)
test_dataset = TensorDataset(test_inputs, test_masks,
torch.tensor(test_labels.values).long())
test_dataloader = DataLoader(test_dataset, batch_size=32)

# Load Pre-trained BERT Model for Sequence Classification
model =
BertForSequenceClassification.from_pretrained('yiyanghkust/finber
t-pretrain', num_labels=2)
optimizer = torch.optim.Adam(model.parameters(), lr=5e-5)

# Training Loop
model.train()
for epoch in range(3):
    start_time = time.time()   # track start time
    for batch in train_dataloader:
        inputs, masks, labels = batch
        optimizer.zero_grad()
        outputs = model(inputs, attention_mask=masks,
labels=labels)
        loss = outputs.loss
        loss.backward()
        optimizer.step()
        end_time = time.time()   #track end time
        epoch_time = end_time - start_time   # Time taken to
train the model

        print(f"Time taken for epoch {epoch+1}: {epoch_time:.2f}
seconds")

total_end_time = time.time()
total_time = total_end_time - start_time

print(f"Total training time: {total_time:.2f} seconds")
# Evaluation
model.eval()
test_predictions = []
test_true_labels = []
with torch.no_grad():
    for batch in test_dataloader:
        inputs, masks, labels = batch
        outputs = model(inputs, attention_mask=masks)
        logits = outputs.logits
        predictions = torch.argmax(F.softmax(logits, dim=1),
dim=1)
        test_predictions.extend(predictions.tolist())
        test_true_labels.extend(labels.tolist())

)
```

```
# Metrics for Test Set
accuracy = accuracy_score(test_true_labels, test_predictions)
precision = precision_score(test_true_labels, test_predictions)
recall = recall_score(test_true_labels, test_predictions)
f1 = f1_score(test_true_labels, test_predictions)
roc_auc = roc_auc_score(test_true_labels, test_predictions)

print("Test Accuracy:", accuracy)
print("Test Precision:", precision)
print("Test Recall:", recall)
print("Test F1-score:", f1)
print("Test ROC-AUC:", roc_auc)

# Training Loop with attention masks
num_epochs = 4
for epoch in range(num_epochs):
    model.train()
    train_loss = 0
    train_predictions = []
    train_true_labels = []
    for batch in train_dataloader:
        inputs, masks, labels = batch
        optimizer.zero_grad()
        outputs = model(inputs, attention_mask=masks,
labels=labels)
        loss = outputs.loss
        loss.backward()
        optimizer.step()
        train_loss += loss.item()
        logits = outputs.logits
        predictions = torch.argmax(F.softmax(logits, dim=1),
dim=1)
        train_predictions.extend(predictions.tolist())
        train_true_labels.extend(labels.tolist())

    # Training Metrics
    train_accuracy = accuracy_score(train_true_labels,
train_predictions)
    train_precision = precision_score(train_true_labels,
train_predictions)
    train_recall = recall_score(train_true_labels,
train_predictions)
    train_f1 = f1_score(train_true_labels, train_predictions)

    print(f"Epoch {epoch+1}/{num_epochs}")
    print(f"Training Loss: {train_loss/len(train_dataloader)}")
    print(f"Training Accuracy: {train_accuracy}")
    print(f"Training Precision: {train_precision}")
    print(f"Training Recall: {train_recall}")
    print(f"Training F1-score: {train_f1}")
```