Financial Statement Fraud Detection Using Large Language Models

Ryan Huang

Abstract

We propose a novel financial statement fraud detection model leveraging recent advances in Large Language Models (LLMs), specifically LLaMA2. By integrating deep learning techniques with domain-specific financial expertise, we demonstrate significant enhancements in the detection of fraudulent activities. Unlike prior studies, our approach utilizes LLMs to systematically analyze a comprehensive combination of numerical and textual data extracted from annual 10-K financial statements. Our findings highlight the potential of LLMs to substantially improve predictive accuracy, offering practical value to auditors, regulators, and investors.

Keywords: Fraud Detection, Large Language Models, Machine Learning

1 Introduction

Fraud is a global issue that spans various industries, and when left unchecked, it can cause serious harm to stakeholders, employees, and other affiliates. Financial fraudulent activities worldwide have amounted to $5.127 trillion, with most of the increase in the past ten years (Gee and Button 2019). However, with the rise of artificial intelligence usage in businesses in the past five years, that number is almost certain to rise further and cause more damage. Financial statement fraud, however, is challenging to detect and prevent. Typically, there are 40-50 cases a year and even those that are detected end up being detected years after the fraud had been committed (Dyck et al. 2010). Standard machine learning models have proven to have good accuracy but need to catch up in correctly predicting the right firms to be fraudulent. With recent technological advances in large language models, more effective methods would be invaluable to regulators, auditors, and investors.

Our study aims to develop a new dataset and new financial statement fraud detection model using available information from the SEC financial statements and the Accounting and Auditing Enforcement Releases (AAERs). We differ from Bao et al. in that we do not limit ourselves to only financial data. We will model our model after Chen, Kelly, and Xiu (Chen et al. 2022). They leverage the LLMs such as GPT and LLaMA to analyze all news data to predict stock returns. In the scope of this study, there is little existing literature on using LLMs to analyze financial text, voice, and facial expression data. We provide an implementation using LLMs to improve on existing machine learning results. Similarly to Bao et al., we seek to explain fraud out-of-sample, essentially making this a prediction problem, rather than the standard causal inference problem such as in Dechow et al. (Dechow et al. 2011).

We use five types of fraud prediction models from previous literature as benchmarks: logistic regression, random forest, SVM, XGBoost, and RUSBoost. We use similar performance evaluation metrics as Bao et al. (Bao et al. 2020) to assess the performance of these models. These will be discussed in section IV of the paper. First, we use the area under the Receiver Operating Characteristics (ROC) curve (AUC). We also use the Normalized Discounted Cumulative Gain at the position k (NDCGk). Thirdly, we modify our measures of sensitivity and precision to reflect the top 1% of predicted fraudulent companies probability.

An exciting question is whether adding more features to our models will improve the performance. Once the LLM is implemented, we must re-evaluate existing machine learning models with the new data. We will also perform a more in-depth analysis of the most important financial variables and ratios based on more current literature. With only 28 financial variables and 14 financial ratios, we do not see the performance increasing significantly unless more information is given about the companies (i.e., MD&A, earnings call data, etc.). Our study will join an extremely small area of financial literature incorporating LLMs to predict financial statement fraud out of sample. Studies like Chen, Kelly, Xiu (Chen et al. 2022) have paved the way for using LLMs in financial research. Some of the preliminary results raise the possibility that the LLMs could further improve the precision of machine learning models in fraud detection in financial statements.

1.1 Differences from Bao et al. (2020)

Our paper builds upon and extends the foundational methodology introduced by Bao et al. (Bao et al. 2020), yet differs distinctly in several critical aspects. Bao et al. focus exclusively on structured financial data (e.g., accounting ratios and numerical indicators) and employ classical machine learning techniques, notably Random Under-Sampling Boosting (RUSBoost), to predict fraudulent financial statements. In contrast, our approach integrates recent advancements in large language models (LLMs), specifically LLaMA2, enabling us to leverage unstructured textual and linguistic information alongside traditional financial metrics extracted directly from annual 10-K reports. By exploiting the rich narrative disclosures contained in these financial statements—particularly the Management Discussion and Analysis (MD&A) sections—we aim to capture subtle, previously overlooked linguistic indicators of fraudulent reporting. Thus, while we initially reproduce Bao et al.'s framework as a baseline, our main methodological innovation is the incorporation of multimodal analysis through advanced transformer-based language models, significantly expanding the scope and capability of fraud detection beyond purely numerical approaches.

2 Background

One of the most common accounting principles is the fraud triangle, proposed by Cressey in the 1970s. It consists of three components: perceived financial need, opportunity, and rationalization (Cressey 1973). Most companies expect significant returns while putting in the minimum effort to maximize their scarce resources. Fraudulent activities are often only discovered after whistleblower tips to the SEC. While fraudulent activities may seem one-dimensional on the surface, there are many indirect costs associated with fraud, including loss of reputation, productivity loss, and irreversible financial damage to employees of the firm that is committing fraud.

2.1 Machine Learning Approaches to Financial Fraud Detection

Early fraud detection research primarily relied on statistical methods and traditional classifiers using financial ratios and simple accounting-based risk indicators (Dechow et al. 2011). However, these early approaches struggled with issues such as severe class imbalance and limited predictive accuracy (Cecchini et al. 2010). More recent developments have applied sophisticated machine learning techniques, notably ensemble methods, to enhance detection accuracy.

Bao et al. (Bao et al. 2020) significantly advanced this line of research by using Random Under-Sampling Boosting (RUSBoost) and extensive sets of raw financial statement data, achieving better ranking and detection performance than traditional models. Their approach demonstrated how ensemble classifiers combined with accounting domain expertise could yield substantial improvements in fraud prediction. Similar research efforts have leveraged deep neural networks to automatically capture nonlinear interactions between financial variables, further enhancing predictive power (Chen et al. 2022).

2.2 Large Language Models in Fraud Detection

Recent advances in transformer-based Large Language Models (LLMs), such as GPT models, have attracted attention for their ability to effectively capture subtle linguistic patterns associated with fraudulent activities across multiple domains (Chen et al. 2022). Chen et al. (Chen et al. 2022), for instance, illustrate the potential of LLMs to analyze financial texts comprehensively, extracting latent indicators predictive of firm-level outcomes, including returns and potentially fraudulent reporting.

Other studies using LLMs in broader fraud detection contexts demonstrate significant predictive improvements over traditional models. These improvements stem from transformers' capability to learn nuanced patterns in sequential data, overcoming limitations of previous simpler methods (Fawcett 2006). While Bao et al. (Bao et al. 2020) initially hypothesized the potential benefits of integrating textual data with financial information, recent research explicitly demonstrates these gains empirically, highlighting LLMs' substantial promise in fraud-related tasks.

2.3 Financial Text Analysis Using Large Language Models

The financial disclosure literature increasingly emphasizes the predictive value of textual narratives found in annual reports (10-K filings), earnings calls, and management discussions. Traditional textual analysis methods using dictionary-based approaches or readability scores have proven limited. Recently, however, more sophisticated LLM-based methods have improved fraud prediction by extracting nuanced signals from extensive financial narratives (Loukas et al. 2021).

Research demonstrates that fine-tuning LLMs specifically on financial disclosures can detect subtle cues of misreporting and fraud risk effectively. For instance, leveraging linguistic features from Management's Discussion and Analysis (MD&A) sections has enhanced fraud prediction models significantly compared to purely numeric data approaches **??**. These developments highlight the importance of domain-specific fine-tuning, showing clear methodological advances in financial text analysis capabilities using large language models. The literature thus supports integrating financial texts, processed via LLMs, into comprehensive fraud detection frameworks to improve overall accuracy and reliability.

3 Methodology

We use several machine learning methods outlined in Bao et al. (Bao et al. 2020) and choose other common methods to experiment on fraud datasets. We adjust the hyper-parameters to tune the model performance. We use the public dataset from Bao et al. (Bao et al. 2020) and conduct our baseline analysis on their data. For the initial testing year 2003, we used 1991-2001 as the training set and 2002 as a validation set to tune our models. For each additional year after 2003 up until our last sample test year of 2008, we increase the training and validation sets by one year to increase model performance. For example, for the test year 2008, we use 1991-2006 as the training set, and validate on 2007. Dyck, Morse, and Zingales (Dyck et al. 2010) find it takes, on average, two years for fraudulent activities to be discovered, hence the two-year gap between the training set and test set. Additionally, we set the $random\_state$ to be 42 for ease of recreatibility. We also normalize the features for easier interpretability. Fig. 1 shows the overall methodology for our experiment.

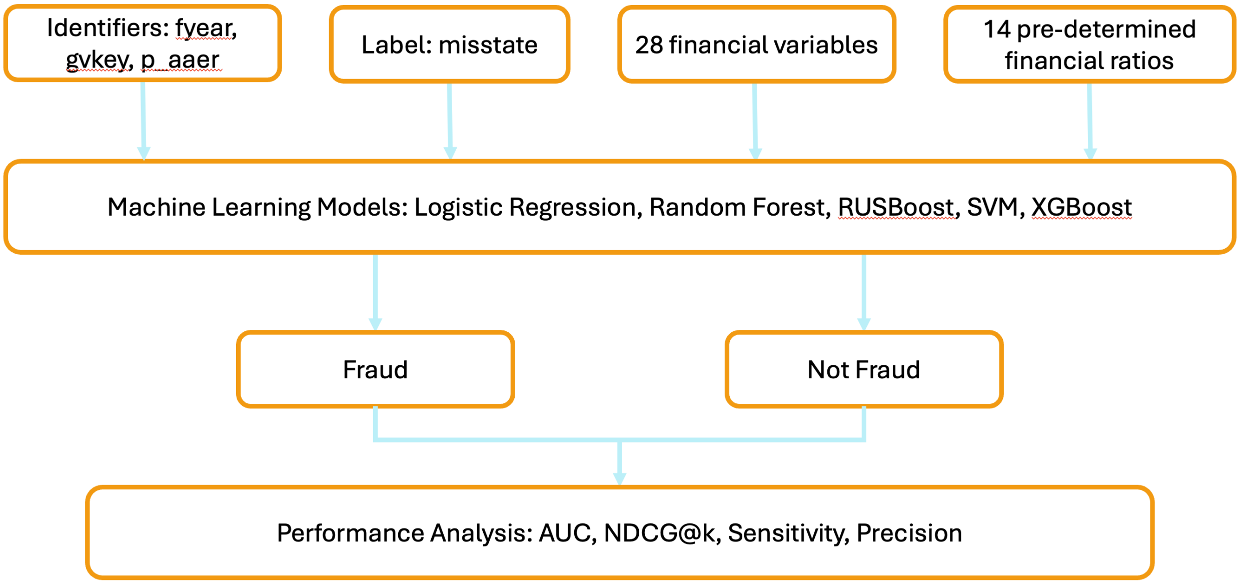

Identifiers: fyear, gvkey, p_aaer
Label: misstate
28 financial variables
14 pre-determined financial ratios
Machine Learning Models: Logistic Regression, Random Forest, RUSBoost, SVM, XGBoost
Fraud
Not Fraud
Performance Analysis: AUC, NDCG@k, Sensitivity, Precision

Figure 1: Figure of Experiment

We describe the parameters of our experiments below:

3.1 Machine Learning Methods

3.1.1 Logistic Regression

We leverage the standard logistic regression model to predict fraud probabilities. We employ a class weight ratio to deal with the imbalance in the dataset, calculated by taking the ratio of fraudulent instances to non-fraudulent instances. We then pick those firms above a threshold and classify them as fraudulent.

3.1.2 Random Forest

We use a standard Random Forest classifier by tuning the following hyper-parameters: $n\_estimators$, $class\_weight$, and max_depth. We use the validation set and GridSearch to find the optimal parameters that maximize AUC. We employ the same class weight ratio from logistic regression to handle imbalanced data.

3.1.3 SVM

We optimize our SVM model by picking a C, which is the cost ratio of misclassifying fraud and non-fraud. The financial kernel from Cecchini et al. (Cecchini et al. 2010) takes advantage of the raw financial data by mapping it into pre-defined ratios, typically intrayear and year-to-year change ratios. We choose not to use it because of the difficulty of implementation and the lack of performance gains over the standard SVM model.

3.1.4 XGBoost

Similar to Random Forest, we perform a GridSearch for the optimal parameters. We optimize for AUC by adjusting max_depth, $learning\_rate$, and $n\_estimators$, where $learning\_rate$ scales the prediction of each weak learner.

3.1.5 RUSBoost

We use the random undersampling package in Python to undersample the majority class so that the input is a matched set of fraud and non-fraudulent observations. We train the matched sample through a weak classifier, which will give a correct or incorrect classification. If the weak classifier is incorrect, the weight of the incorrect observation will be increased and vice versa. Once we iterate over and over, we apply a error rate-based weight on each weak classifier and take a weighted average to build a strong classifier to predict. This is depicted in Fig. 2:

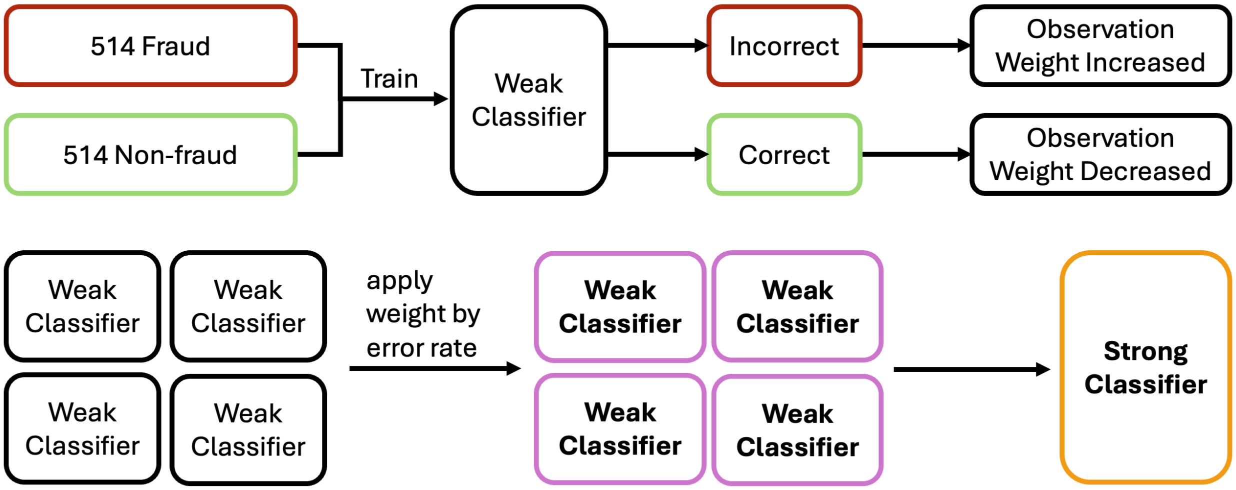


Figure 2: Depiction of RUSBoost model. The top half repeats itself, and the bottom half takes a weighted average of all the weak classifiers from the top.

3.2 Fine Tuning LLaMA2

We input the text in a format that is easily readable to the model. We use the LLaMA2 tokenizer and set the max length of each tokenizer to be 512 initially. We find that setting the max length of each tokenizer higher will result in higher GPU memory usage. Additionally, to reduce computational stress on our system, we decide to sample the training set with 100 positive and 100 negatives cases of fraud. Our goal is to eventually use all or more cases, but to acquire some preliminary results, we test on a small sample first.

3.2.1 Hardware Description

We use 5 x H100 GPUs with 80GiB of memory each. These GPUs are essential for handling large token lengths and batch sizes during the training process. The H100 GPUs are known for their high performance in both computational speed and memory handling, making them suitable for extensive deep learning tasks.

3.2.2 Bitsandbytes Configuration

To optimize the memory usage and computational efficiency, we employ the BitsAndBytes library. We configure it with `load_in_4bit` set to True, enabling 4-bit quantization to reduce the model's memory footprint. We use `bnb_4bit_use_double_quant` set to True for double quantization, `bnb_4bit_quant_type` set to ”nf4”, and `bnb_4bit_compute_dtype` set to `torch.bfloat16`. This configuration helps in achieving a balance between memory usage and computational efficiency, which is critical for handling large models and datasets.

3.2.3 Loading the Model

We load the LLaMA2 model using the HuggingFace `AutoModelForCausalLM` with the specified BitsAndBytes configuration. The model is loaded with `quantization_config` set to the BitsAndBytes configuration and `device_map` set to ”auto” to automatically allocate the model across available GPUs. The tokenizer is loaded from the Meta LLaMA2 official HuggingFace repository, ensuring compatibility with the model. We set the tokenizer's padding token to be the end-of-sequence (EOS) token to handle variable-length inputs during training.

3.2.4 Preprocessing

Preprocessing involves several steps ensuring that the model receives all necessary information to complete the task:

1. **Formatting the Dataset**: Each sample is formatted using the defined prompt structure. The input text is formatted to include an instruction, the input context, and a placeholder for the response. The format is designed to be clear and consistent, ensuring that the model can understand and process the input correctly. The format includes the following sections:

    - **Introduction Blurb**: Provides context that an instruction follows.
    - **Instruction Key**: Specifies the task the model should perform.
    - **Input Key**: Contains the input text to be analyzed.
    - **Response Key**: Placeholder for the model's response.
    - **End Key**: Marks the end of the formatted text.

```
<s> Below is an instruction that describes a task. Write a response that
   appropriately completes the request.

### Instruction:
[Instruction Text]

Input:
[Input Text]

### Response:
["Yes"] or ["No"]

### End
```

Listing 1: Defined Prompt Structure

2. **Tokenization**: The formatted text is tokenized using the LLaMA2 tokenizer, ensuring that each tokenized sample fits within the maximum token length.

3. **Splitting Long Texts**: Long input texts are split into smaller chunks that fit within the token length limit while retaining the instruction and response placeholders in each chunk. We also use overlapping tokens to ensure continuity between the chunks.

4. **Batch Processing**: The tokenized texts are processed in batches to improve efficiency due to our hardware constraints.

5. **Filtering**: Samples that exceed the token length after splitting are truncated, ensuring that all samples fit within the specified limit. In our case, we should not have any samples being filtered out due to splitting, in order to maintain all information.

This preprocessing ensures that the dataset is ready for training, with each sample correctly formatted and tokenized.

3.2.5 PEFT Configuration

We use the Parameter-Efficient Fine-Tuning (PEFT) approach with Low-Rank Adaptation (LoRA) to fine-tune the model efficiently. The PEFT configuration includes:

- **LoRA Rank (`lora_r`)**: Determines the size of the low-rank decomposition.
- **LoRA Alpha (`lora_alpha`)**: Scaling factor for the adaptation.
- **LoRA Dropout (`lora_dropout`)**: Dropout rate to regularize the adaptation process.
- **Bias Term Inclusion**: Option to include bias terms in the adapted layers.

This configuration allows for efficient fine-tuning by adapting a subset of the model parameters, reducing computational overhead while maintaining performance.

3.2.6 Parameters

We define the following function $fine\_tune$ with various parameters:

```
fine\_tune(model, tokenizer, preprocessed\_dataset, lora_r, lora_alpha,
    lora_dropout, bias, task_type, per_device_train_batch_size,
    gradient_accumulation_steps, warmup_steps, num_train_epochs, learning_rate
    , fp16, logging_steps, output_dir, optim)
```

Listing 2: Fine Tuning Function

- **model**: We use the LLaMA2 official model from Meta on the HuggingFace repository.
- **tokenizer**: We use the LLaMA2 tokenizer.json file from the Meta LLaMA2 official HuggingFace repository.
- **preprocessed_dataset**: We input a cleaned dataset, with columns `input`, `output`, and `instruction`. This dataset consists of the 10-K annual reports for firms from 1993-2003 that are matched in the AAER dataset.
- **lora_r**: We set `lora_r` to 16. This value is chosen to control the size of the low-rank decomposition, balancing model complexity and computational efficiency.
- **lora_alpha**: We set `lora_alpha` to 64. This scaling factor adjusts the contribution of the low-rank matrices, impacting the intensity of the adaptation.

- **lora_dropout**: We set `lora_dropout` to 0.1. This dropout rate regularizes the adaptation process and helps prevent overfitting, improving generalization.

- **bias**: We set `bias` to ”none”, meaning we do not include bias terms in the adapted model’s layers to reduce the number of additional parameters.

- **task_type**: We set `task_type` to ”CAUSAL_LM” as we are fine-tuning for the task of predicting the next token following a sequence of tokens.

- **per_device_train_batch_size**: We set this to 1. This batch size is chosen based on the GPU memory capacity to ensure that training can proceed without memory issues.

- **gradient_accumulation_steps**: We set this to 8. This means gradients are accumulated over 8 steps before performing a backward pass, allowing for effectively larger batch sizes without exceeding memory limits.

- **warmup_steps**: We set `warmup_steps` to 2. This number of steps is used to linearly increase the learning rate from zero, stabilizing training in the early stages.

- **num_train_epochs**: We set `num_train_epochs` to 3. This total number of epochs is chosen to balance between sufficient training and the risk of overfitting.

- **learning_rate**: We set `learning_rate` to 2e-4. This initial learning rate is chosen to control the step size during gradient descent, balancing between convergence speed and stability.

- **fp16**: We set `fp16` to True, indicating that we use mixed precision training with 16-bit floating-point numbers to significantly speed up training and reduce memory usage.

- **logging_steps**: We set `logging_steps` to 1. This frequency ensures that training progress is logged regularly, helping to monitor and detect potential issues early on.

- **optim**: We set `optim` to ”paged_adamw_32bit”. The optimizer chosen, AdamW, is efficient and effective in handling sparse gradients and large-scale data.

3.3 Testing Fine-Tuned LLaMA2 Model

After completion of fine tuning, we use our new model to run a simple inference task on the test dataset, which consists of the firms in year 2004. Our goal is to expand the test set further in the future, but for now, we only test year 2004. We take a sample of four firms, two positive and two negative, and input it into the fine-tuned model using the same preprocessing methods used when we fine tuned the model.

4 Data

4.1 AAER Dataset

We use the public dataset compiled from Detecting Accounting Fraud in Publicly Traded U.S. Firms Using a Machine Learning Approach. We will refer to this dataset as the JAR dataset in the rest of the paper. They use financial data from Compustat to create the dataset. The dataset consists of two unique identifier variables for each firm: $fyear$ and $gvkey$. The $gvkey$, or Global Company Key, is a six-digit code assigned to each firm in the Compustat database. For the fraud label, they track the case number from the SEC AAER database, which contains a list of all firms that have been audited or investigated for fraud. This is represented by p_aaer in the public dataset, and for every instance of a case number, the label we are predicting, $misstate$, is equal to one. The main features of the public dataset consist of a pairing of 28 financial variables with 14 financial ratios. As the paper we are basing our study on is from 2019, the dataset only consists of cases up until 2018 and needs to be completed. However, we only used the 1991-2008 sample period for our initial testing. The sample began in 1991 due to the stricter enforcement of accounting fraud. The sample ends in 2008 due to the reduction of accounting fraud enforcement due to the higher probability of accounting fraud cases going undetected post-2008 (Rakoff 2014).

4.2 SEC 10-K Dataset

We use the dataset compiled by (Loukas et al. 2021) with annual 10-K filings from the years 1993-2020. For our study, we use a sample set of years 1993-2008 to match the AAER Dataset. We select ITEM 7 from the 10-K annual report, known better as the Management's Discussion and Analysis. This section is commonly used for sentiment analysis and any other forms of textual analysis. We choose to use it for fraud detection because of the complex nature of large language models and their ability to detect any patterns within the texts. Our full dataset consists of nearly 225,000 entries. We use the SEC 10-K Dataset to merge with the AAER dataset and create a training dataset using matched companies and those only in the years 1993-2003. We find that there are around 35,000 matched firms from the years 1993-2003. For testing, we match the years we test on from 2004 and on, and find that around 90% of the firms in the AAER dataset are present in the SEC 10-K dataset.

4.3 Data Distribution

The total number of fraud instances in the JAR dataset is 964, and the total number of non-fraudulent instances is 145,081, summing up to 146,045 entries total. We also plot the distribution of the number of fraud cases per year and find that it follows a relatively normal distribution with a peak in fraud cases in the 2000-2003 range. When we look at the unique number of fraud cases per year, we find that the shape of the distribution of fraud cases per year is similar, but there are significantly fewer cases in the peak years (2000-2003). Table I displays the most common offenders within the dataset. Figure III shows the number of fraud cases per year with the number of total fraud cases overlapping the number of unique fraud cases per year.

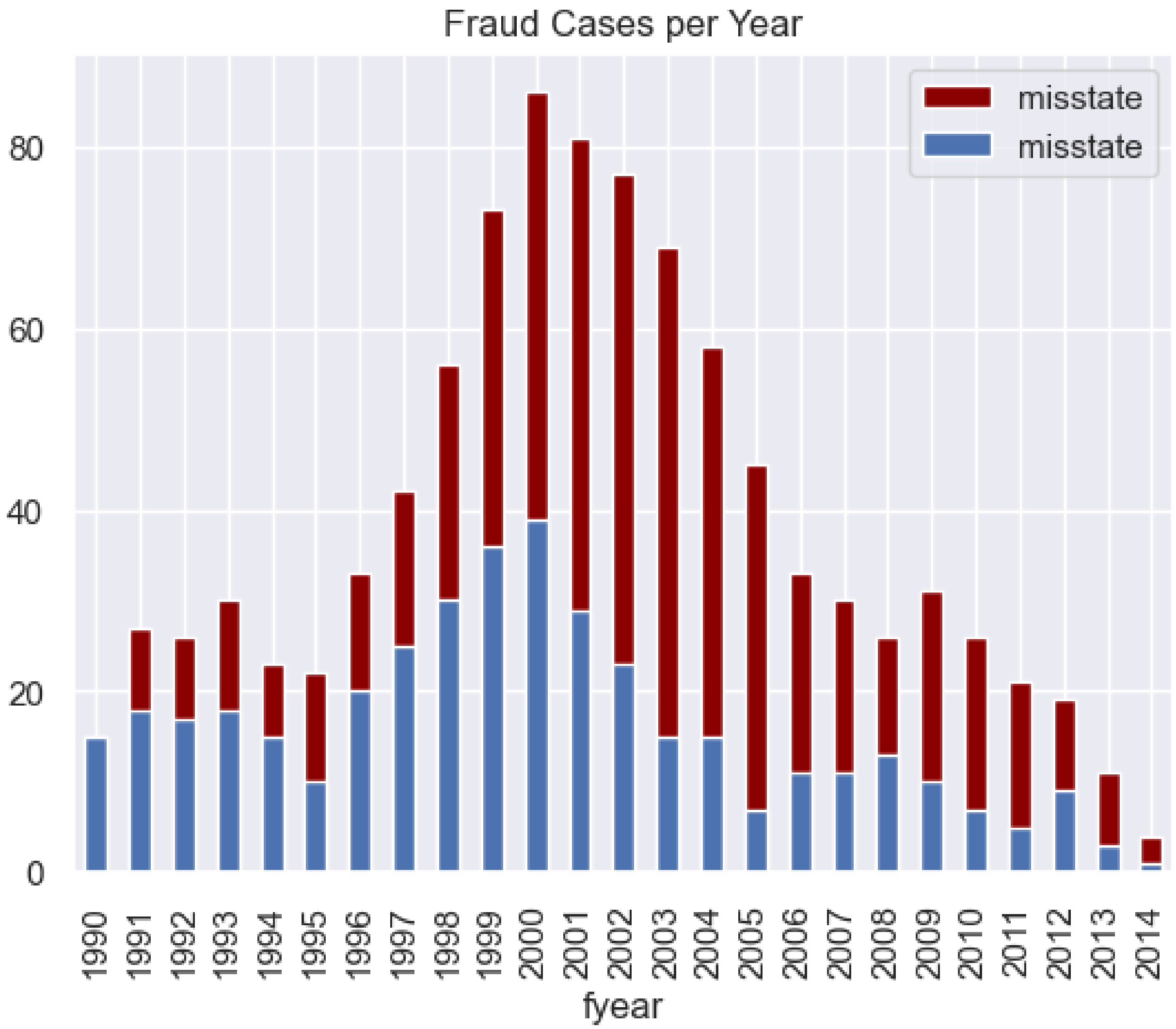


Figure 3: The blue bars are new cases for every year, and the red bars represent the total number of fraud cases, including repeat offenders.

Table 1: Repeat Offenders Frequency

Company Name	**Frequency**
Black Box Corp	13
Overseas Shipholding Group	12
Affiliated Computer Services	12
UnitedHealth Group Inc	11
Integrated Silicon Solution	9
Mercury Interactive Corp	9
Monster Worldwide Inc	8
Broadcom Corp -Cl A	8
Vitesse Semiconductor Corp	8
Comverse Technology Inc	7
Elbit Medical Imaging Ltd	7
Conagra Foods Inc	7
Interpublic Group of Cos	7
NCI Inc	7
Analogic Corp	6
Dow Chemical	6
Take-Two Interactive Sftwr	6
Diebold Inc	6
NIC Inc	6
Engineered Support Systems	6
Cummins Inc	6
UCI Medical Affiliates Inc	6
Brooks Automation Inc	6

4.4 Features

We use a list of raw financial variables selected by Bao et al. (Bao et al. 2020) based on Cecchini et al. (Cecchini et al. 2010) and Dechow et al. (Dechow et al. 2011), compiled from the Compustat database. The final dataset from Bao et al., which we use in this study for a baseline, contains 28 financial variables and 14 financial ratios, which are abbreviated for convenience in Table II and III:

Table 2: 28 Financial Variables

Abbreviation	**Financial Variable**
ACT	Current Assets, Total
AP	Accounts Payable Trade
AT	Assets, Total
CEQ	Common/Ordinary Equity, Total
CHE	Cash and Short-Term Investments
COGS	Cost of Goods Sold
CSHO	Common Shares Outstanding
DLC	Debt in Current Liabilities, Total
DLTIS	Long-Term Debt Insurance
DLTT	Long-Term Debt, Total
DP	Depreciation and Amortization
IB	Income Before Extraordinary Items
INVT	Inventories, Total
IVAO	Investments Advances, Other
IVST	Short Term Investments, Total
LCT	Current Liabilities, Total
LT	Liabilities, Total
NI	Net Income (Loss)
PPEGT	Property, Plant, and Equipment, Total
PSTK	Preferred/Preference Stock (Capital), Total
RE	Retained Earnings
RECT	Receivables, Total
SALE	Sales/Turnover (Net)
SSTK	Sale of Common and Preferred Stock
TXP	Income Taxes Payable
TXT	Income Taxes, Total
XINT	Interest and Related Expense, Total
PRCC_F	Price Close, Annual, Fiscal

Table 3: 14 Financial Ratios

Abbreviation	Financial Variable
DCH_WC	WC Accruals
CH_RSST	RSST Accruals
DCH_REC	Change in Receivables
DCH_INV	Change in Inventory
SOFT_ASSET	% Soft Assets
DPI	Depreciation Index
CH_CS	Change in Cash Sales
CH_CM	Change in Cash Margin
CH_ROA	Change in Return on Assets
CH_FCF	Change in Free Cash Flows
REOA	Retained Earnings Over Total Assets
EBIT	Earnings Before Interest and Taxes Over Total Assets
ISSUE	Actual Issuance
BM	Book-to-Market

5 Performance Evaluation Metrics

Due to the imbalance of our dataset, running standard machine learning methods will result in many false positives. Similarly, if we consider the reality, auditors cannot and will not audit every company. That would take too many resources, and even then, they still may not be accurate in detecting fraud. Therefore, we use the AUC, Precision, Sensitivity, and NCDG@k as defined below from Bao et al.

5.0.1 AUC

The standard performance metric is accuracy which is defined as $\frac{TP+TN}{TP+FN+FP+TN}$, where TP is when the model correctly classifies the firm as a fraud; TN is when the model correctly classifies the firm as non-fraudulent; FP is when the model classifies a firm as fraudulent, but it is not; and FN is when the model does not classify a firm as fraudulent when it should be. On the other hand, AUC relies on the true and false positive rates, such as the ROC curve, which shows a classifier's performance at a certain threshold. In our study, anything below 0.5 should not be feasible since a random guess results in an AUC of 0.5. On the other hand, a perfect AUC is 1.0 and should be very rare. The AUC is the probability that the classifier will rank a true positive higher than a non-fraudulent firm (Fawcett 2006). As shown in Fig. 4, our random forest model performs quite well with an AUC of 0.8037. The optimal ROC curve is when the true positive rate is maximized,

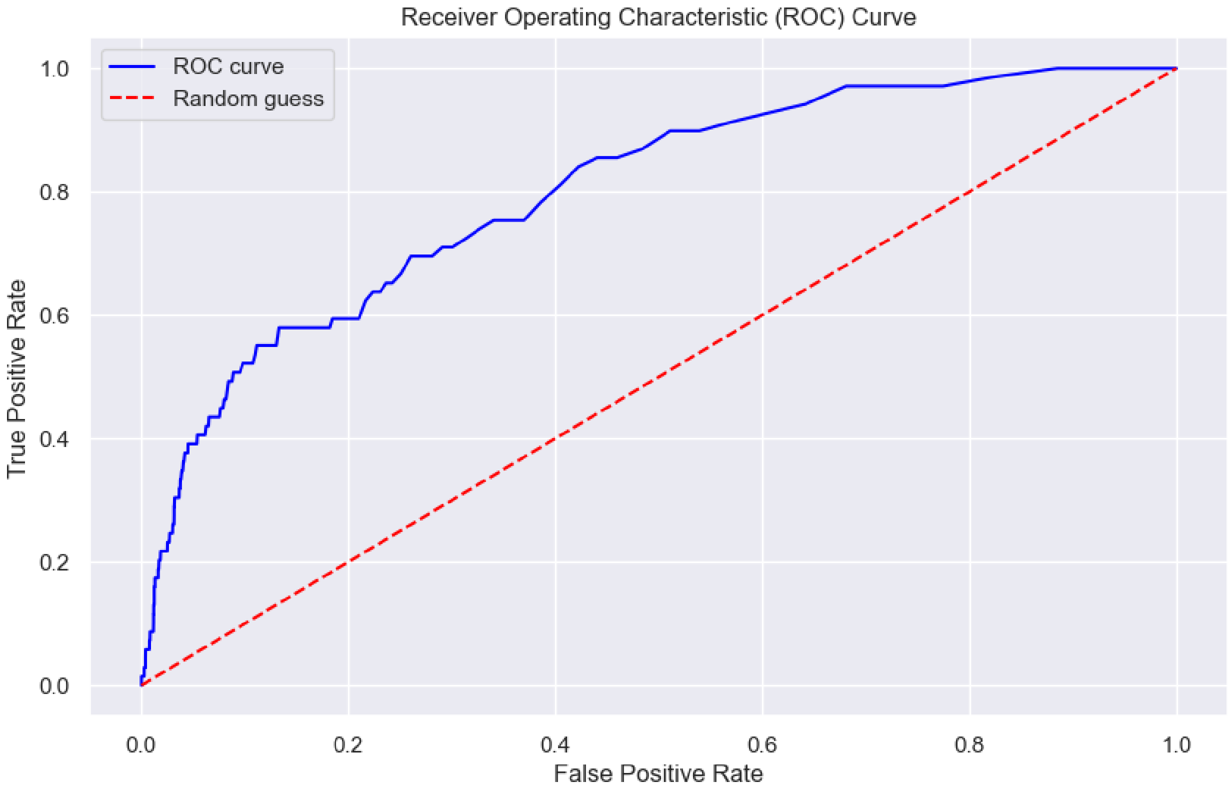
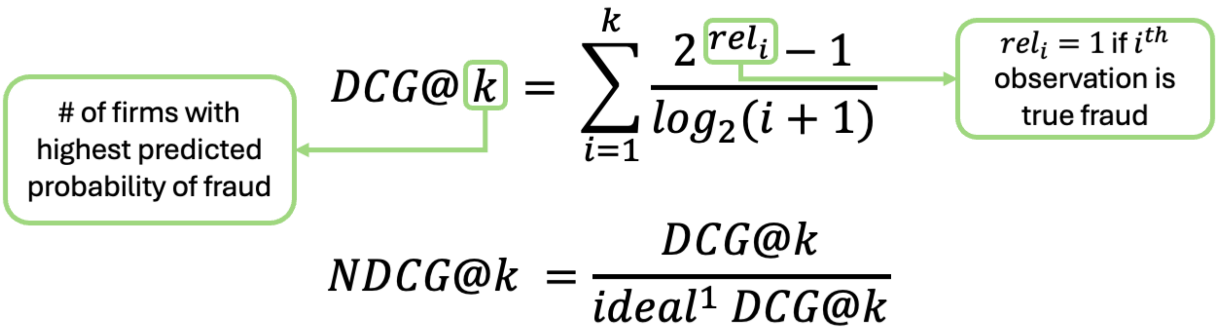


Figure 4: ROC curve for Random Forest, Test Year = 2003

and the false positive rate is minimized.

5.0.2 NDCG@k

Bao et al. considers the fraud prediction task as a ranking problem. They use a discounted cumulative gain function to limit the out-of-sample performance evaluation to only the firms with the highest predicted probability of fraud. The DCG@k scores a fraudulent instance higher than a non-fraudulent instance. The DCG@k is then normalized by the ideal DCG@k, where the ideal is defined by all the true instances of fraud being ranked at the top. Formally, this is:



$$DCG@\boxed{k} = \sum_{i=1}^{k} \frac{2^{\boxed{rel_i}} - 1}{log_2(i+1)}$$

$$NDCG@k = \frac{DCG@k}{ideal^1\ DCG@k}$$

# of firms with highest predicted probability of fraud
$rel_i = 1$ if i^{th} observation is true fraud

Figure 5: Formula for DCG@k and NDCG@k

5.0.3 Sensitivity and Precision

From the NDCG@k, we also modify the sensitivity and precision metrics to follow the same methodology as NDCG@k. Rather than consider all the true positive firms, we consider only the top 1% of firms with the highest predicted probability of committing fraud. This is more accurate for our results because of the imbalance of data. Formally, this is:

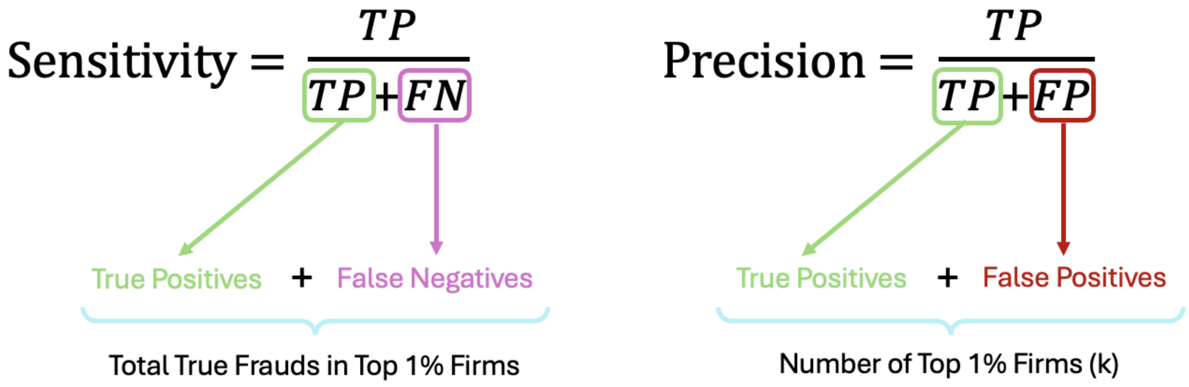

Sensitivity = $\frac{TP}{TP+FN}$
True Positives + False Negatives
Total True Frauds in Top 1% Firms
Precision = $\frac{TP}{TP+FP}$
True Positives + False Positives
Number of Top 1% Firms (k)

Figure 6: Formula for Sensitivity and Precision

5.0.4 Comparing Improved Metrics with Standard Metrics

We compare the accuracy and AUC, sensitivity and modified sensitivity, and recall and modified recall. The figures below are from running the Random Forest model on 28 financial variables. The training set is the firms in the years 1991-2001, the validation set is 2002, and the test set is 2003. In Fig. 7, we find that accuracy is much more inflated than AUC, which naturally makes sense because a 1:150 imbalance has a less than one percent chance of incorrectly classifying a non-fraudulent firm. Once we delve deeper into the data, we can see in Fig. 8 that our number of true positives is relatively low, and our number of false positives is very high. When comparing the improved sensitivity to the standard sensitivity in Fig. 11, our figure shows un-interpretable values at one extreme or the other (0 or 1) and does not align with the confusion matrix (tp, fp, fn, tn). The modified sensitivity is interpretable as it aligns with our confusion matrix. Our precision displays the same behavior in Fig. 10.

When looking at NDCG@k, sensitivity, and precision (improved metrics) in Fig. 9, the shape of the graphs shows a similar behavior.

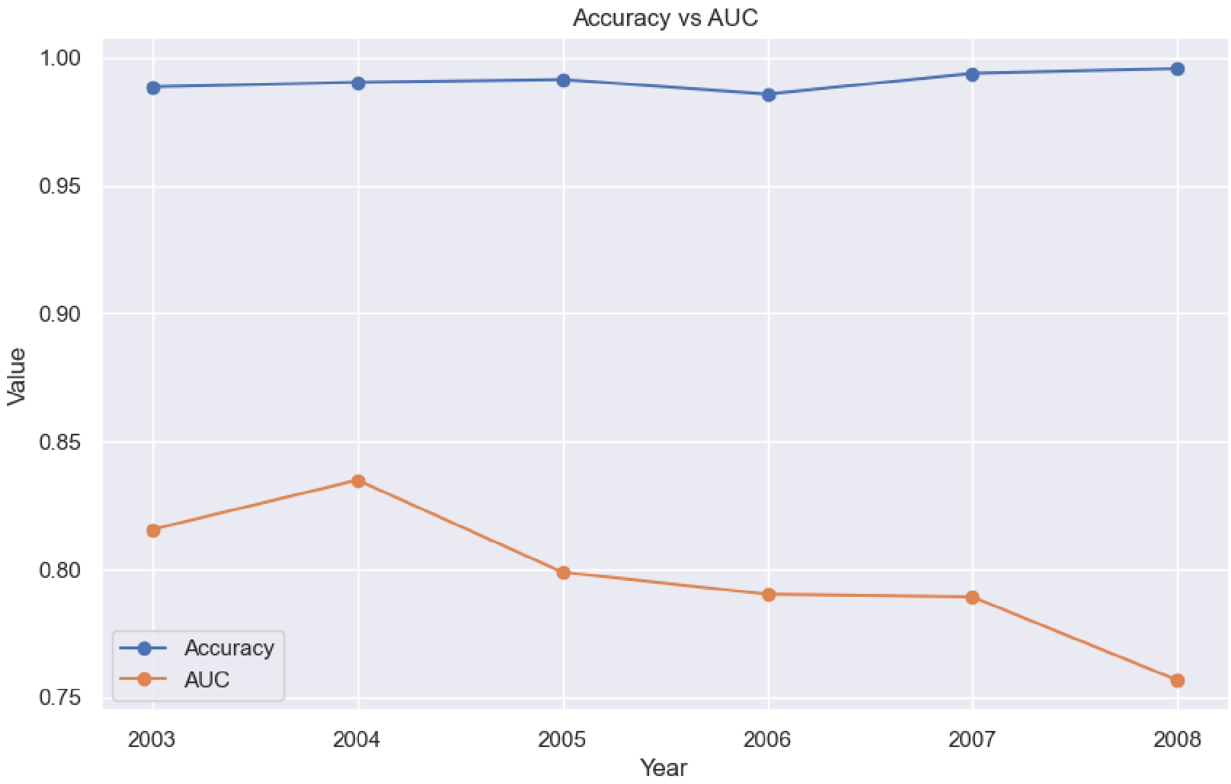


Figure 7: Accuracy vs AUC

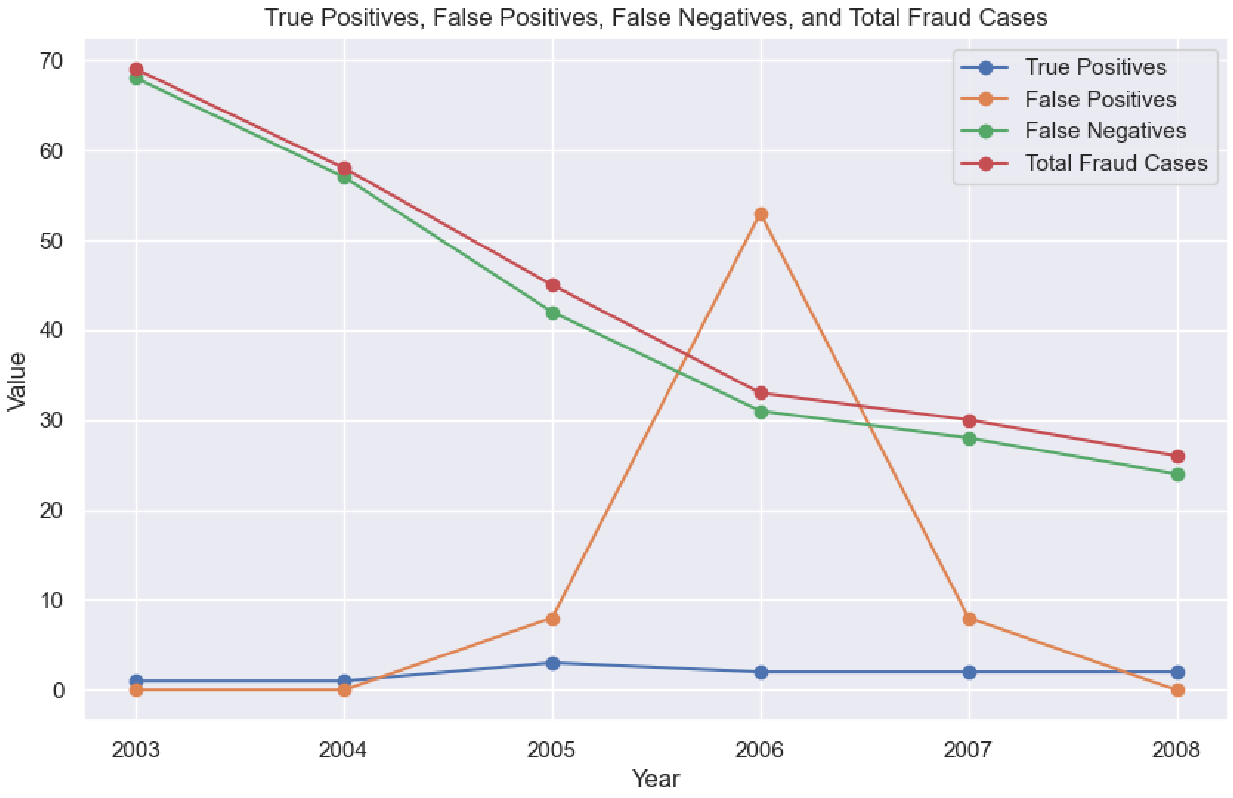


Figure 8: of True Positives, False Positives, False Negatives, Total

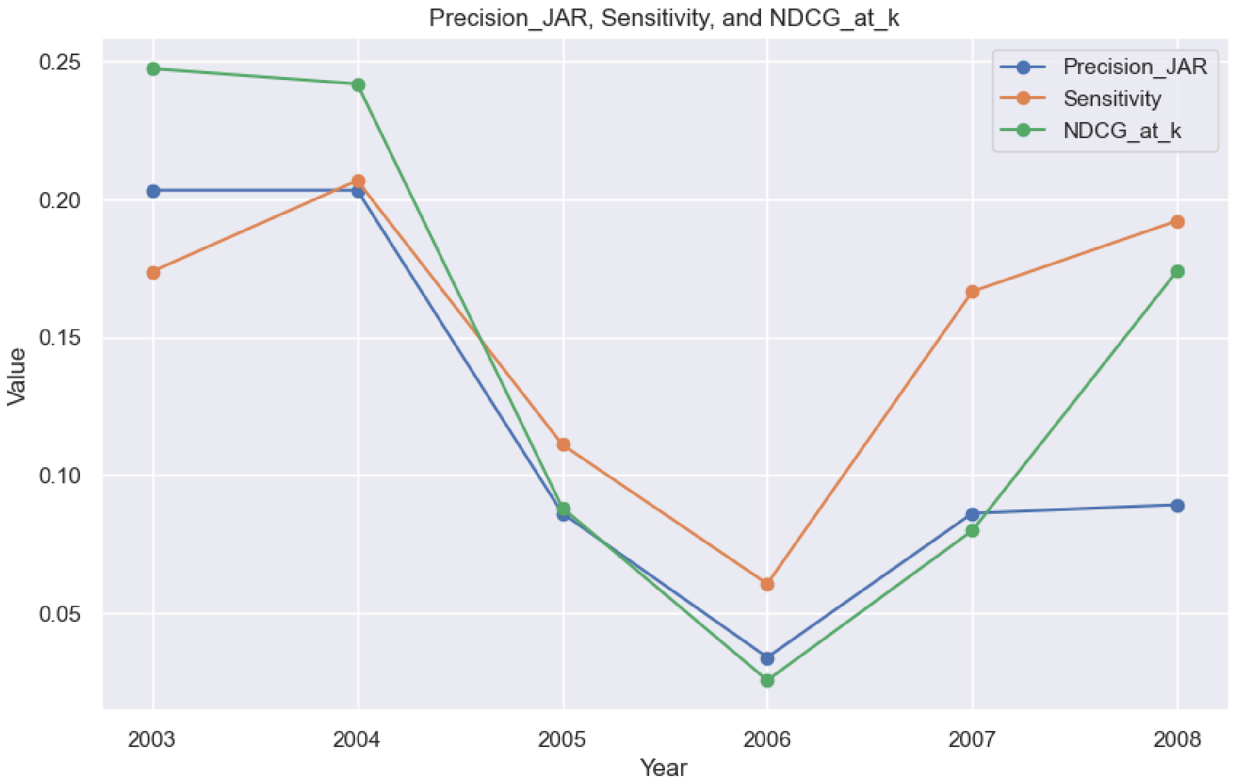


Figure 9: Improved Performance Evaluation Metrics)

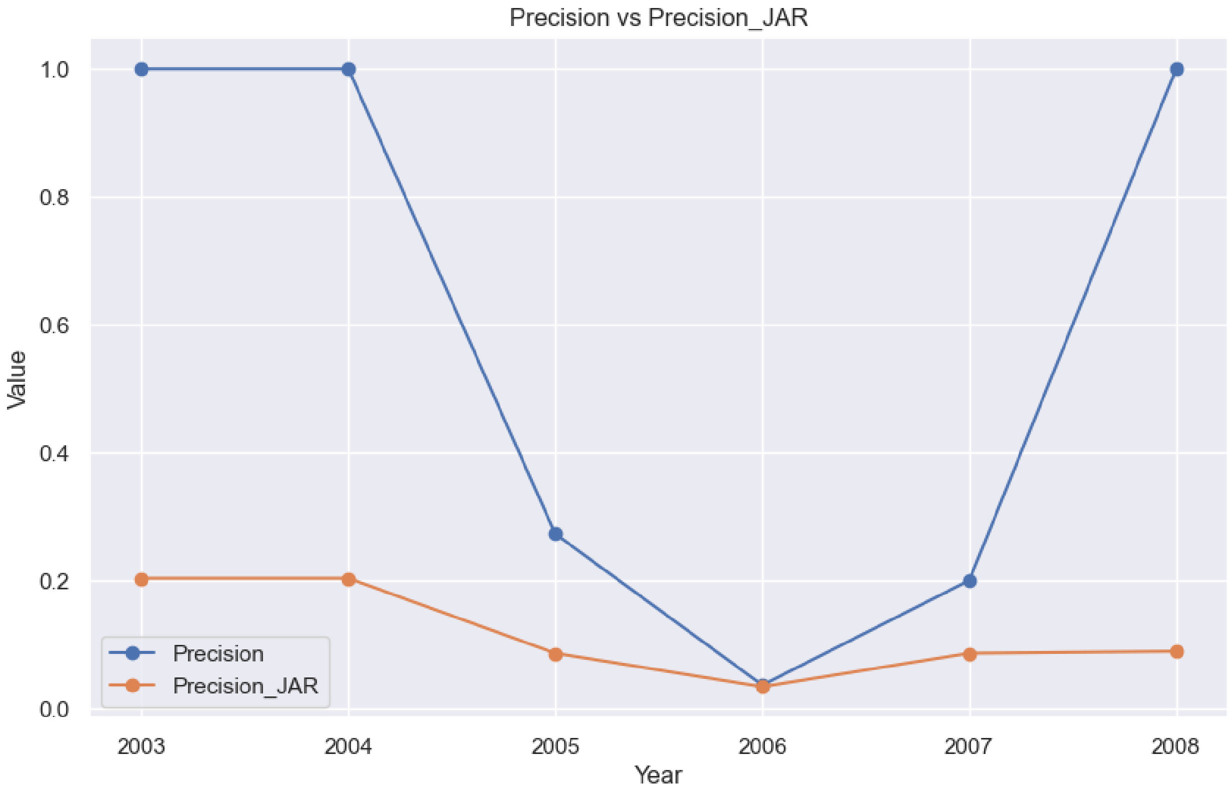


Figure 10: Precision vs Precision (top 1%)

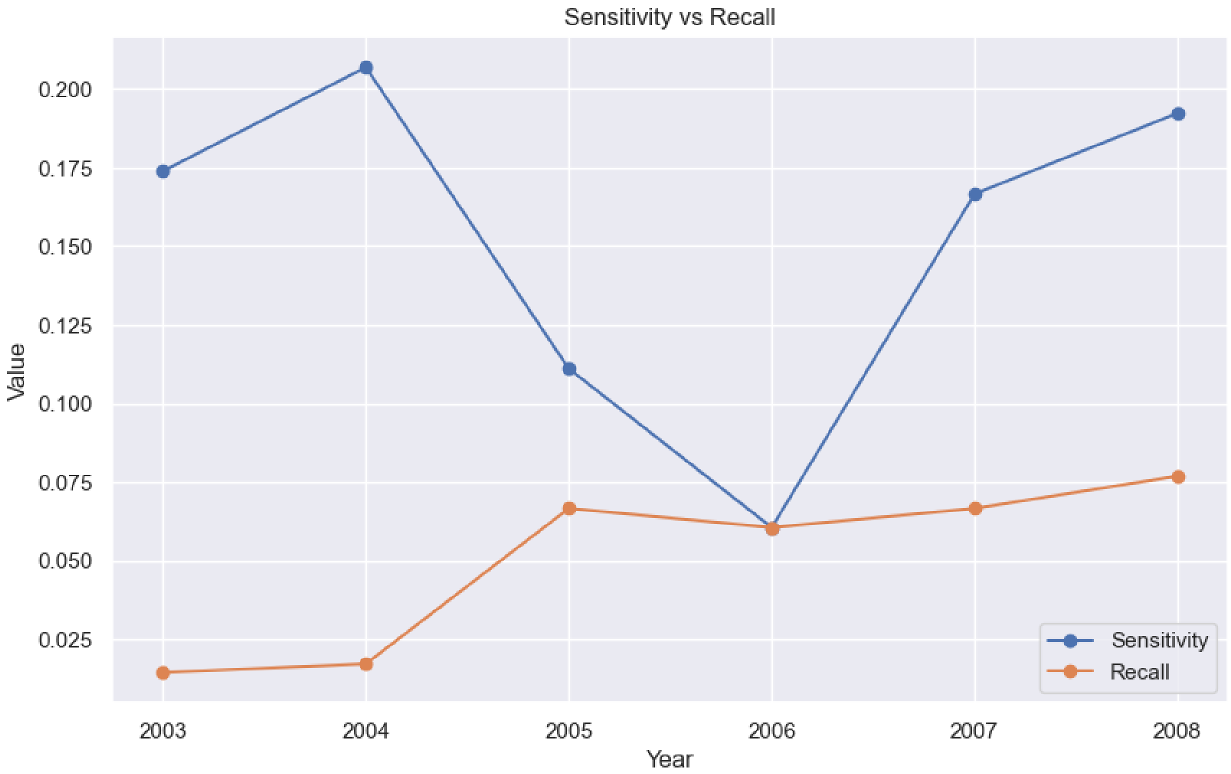


Figure 11: Sensitivity (Recall for Top 1%) vs Recall

6 Machine Learning Results

The preliminary results of our study, as shown in Tables IV-VII, provide an in-depth comparison of various machine learning methods applied to financial prediction tasks using different sets of input variables. By comparing methods across different input sets, we observe notable trends and patterns that shed light on the effectiveness of these techniques and the impact of additional financial ratios. The evaluation metrics considered are AUC (Area Under the Curve), NDCG@k (Normalized Discounted Cumulative Gain at rank k), Sensitivity, and Precision.

6.1 Bao et al. vs. our Results

Initially, the paper evaluates Logit, RUSBoost, and SVM using 28 financial variables. Among these, RUSBoost stands out with the highest AUC (0.725) and substantial improvements in sensitivity and precision. This suggests that RUSBoost, which balances class distributions through under-sampling, is particularly effective in handling the imbalanced nature of financial data, leading to more reliable predictions. Now compare this with our recreation. Table VII shows RUSBoost behaving similarly, with relatively little change in all performance metrics. Overall, we see slight differences in the AUC from our results and the results from Bao et al. However, when it comes to the other three metrics, we see they increase dramatically. We can attribute some of this increase to implementation differences, which we describe in detail how we leverage Logit and SVM to acquire the highest performance in all metrics. Expanding our methodology to include Random Forest and XGBoost in the second set of evaluations, we see significant enhancements in performance metrics:

- Random Forest emerges as the top performer with an AUC of 0.798 and superior sensitivity and precision. This indicates its robustness in capturing complex patterns and interactions within the financial data, making it a highly effective method.
- XGBoost also demonstrates strong performance, though slightly trailing behind Random Forest. Its notable AUC and precision reflect its ability to handle large datasets with high dimensionality effectively

6.2 Change in Input

When additional financial ratios are incorporated (Table VI), the performance of the models exhibits mixed results:

- Random Forest maintains a high AUC but experiences reductions in other metrics. This indicates that while the additional ratios provide new information, they might also introduce noise or redundancy, slightly hindering the model's sensitivity and precision.
- XGBoost, on the other hand, shows a slight improvement in AUC, suggesting it can leverage the extra features to some extent, but like Random Forest, it faces a trade-off with sensitivity and precision.

6.3 Main Takeaways

Overall, our preliminary finds underscore several key points:

- **Model Selection**: The choice of the machine learning method significantly impacts the performance. RUSBoost, Random Forest and XGBoost consistently outperform Logit and SVM, highlighting the benefits of ensemble methods and advanced boosting techniques in financial predictions.
- **Feature Set Composition**: The introduction of additional financial ratios has a nuanced impact. While it can slightly enhance AUC, it often reduces sensitivity and precision. This suggests that while more features can provide more information, they also require careful

selection and preprocessing to avoid diminishing returns or performance degradation due to overfitting or noise.

- **Robustness and Adaptability**: Random Forest demonstrates robust performance across different feature sets, indicating its adaptability and ability to generalize well. In contrast, methods like Logit and SVM show more variability and are less robust to changes in the feature set.
- **Sensitivity to Imbalanced Data**: Methods like RUSBoost, specifically address the class imbalance, show marked improvements in sensitivity and precision, critical for financial prediction tasks where the minority class (e.g., financial distress) is often of most significant interest.

These preliminary results highlight the importance of method selection and feature engineering in financial prediction. Ensemble methods, particularly Random Forest and RUSBoost, demonstrate superior performance and robustness while introducing additional financial ratios, which presents opportunities and challenges. Our future research will focus more on optimizing feature selection and exploring LLMs to introduce new features that may increase the performance of these models.

Table 4: Performance Evaluation Metrics for the Test Period 2003-08
Input: 28 Financial Variables

Method	**AUC**	**NDCG@k**	**Sensitivity**	**Precision**
(1) Logit	0.640	0.027	3.345%	2.845%
(2) Random Forest	0.798	0.143	15.192%	11.706%
(3) RUSBoost	0.770	0.051	4.963%	5.099%
(4) SVM	0.638	0.024	3.057%	2.562%
(5) XGBoost	0.760	0.099	9.634%	7.687%

Table 5: Performance Evaluation Metrics for the Test Period 2003-08
Input: 28 Financial Variables + 14 Financial Ratios

Method	**AUC**	**NDCG@k**	**Sensitivity**	**Precision**
(1) Logit	0.703	0.022	3.103%	2.562%
(2) Random Forest	0.790	0.084	7.689%	6.007%
(3) RUSBoost	0.751	0.029	2.996%	2.835%
(4) SVM	0.708	0.027	3.084%	2.285%
(5) XGBoost	0.713	0.070	8.121%	6.546%

Table 6: Percentage Change From Table IV to Table V

Method	**AUC**	**NDCG@k**	**Sensitivity**	**Precision**
(1) Logit	9.84%	-18.52%	-9.95%	-7.24%
(2) Random Forest	-1.00%	-41.26%	-48.68%	-49.39%
(3) RUSBoost	-2.47%	-43.14%	-44.40%	-39.63%
(4) SVM	10.97%	12.50%	-10.81%	0.88%
(5) XGBoost	-6.18%	-29.29%	-14.84%	-15.71%

7 LLM Results

Our script finds the max token length to be 4096; however, when we try to train our model using the specified token length, the system does not have enough memory. We use 5 x H100 GPUs with 80GiB of memory each. When training with a token length of 4096, we find that the training halts at 2500 steps, whereas at a token length of 1024, we find that the training completes in about three hours. Therefore, we sample the training and test sets to improve efficiency and produce some results. Rather than using the modified performance metrics, we revert back to the standard metrics in the field: accuracy, recall, precision, and F1-score. We find that our fine tuning either has no impact or our testing has some issues, as the performance is no better than a guess, which we referred to as 50/50 in our machine learning models.

Table 7: Model Performance Metrics

Metric	**Value**
Accuracy	0.5000
Precision	0.5000
Recall	1.0000
F1-score	0.6667

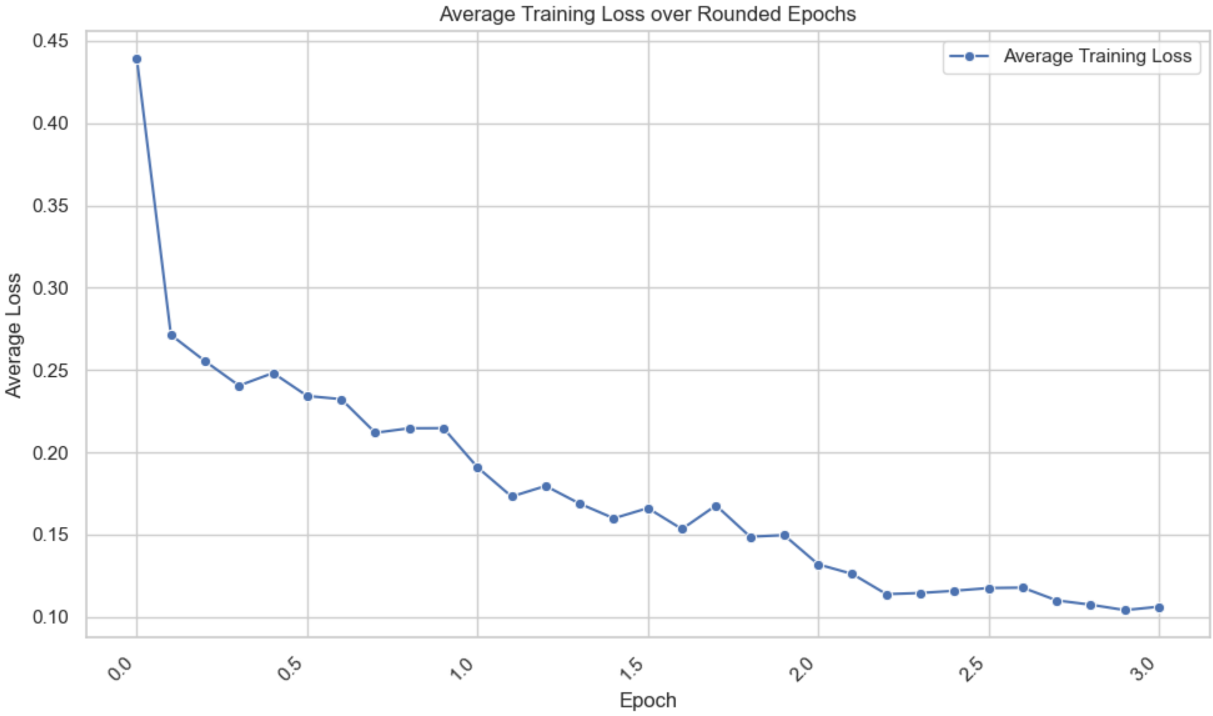


Figure 12: Average Training Loss per Epoch

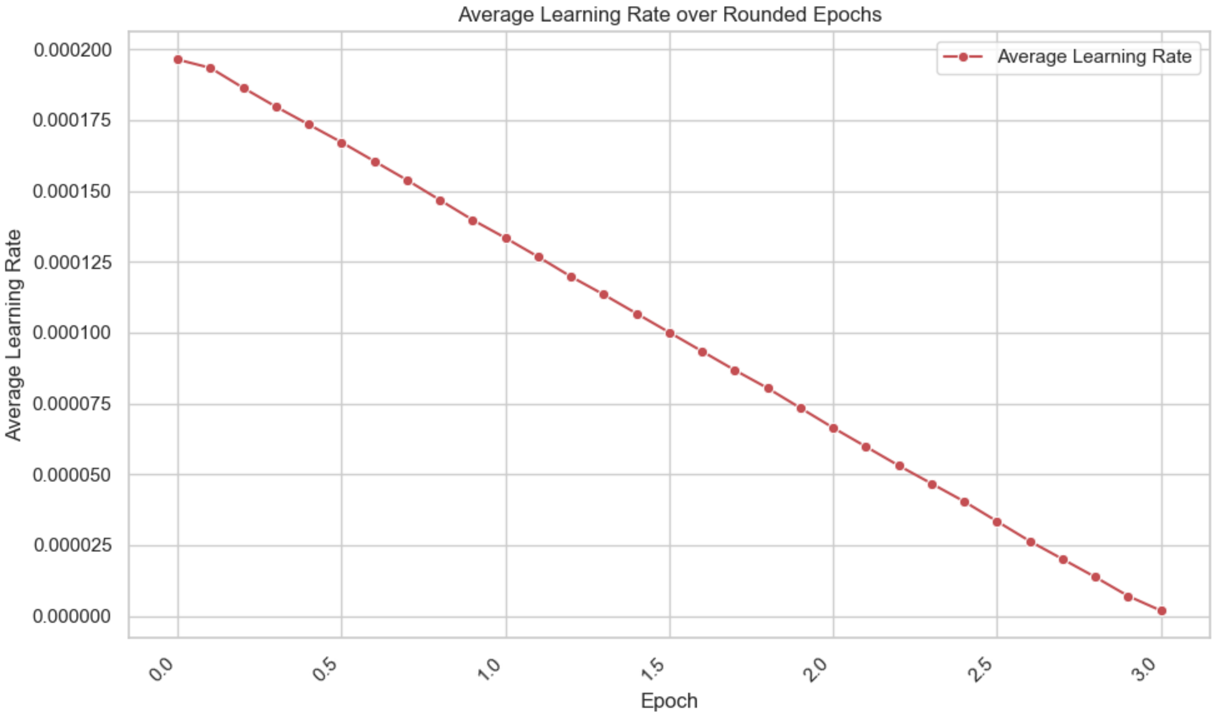


Figure 13: Average Learning Rate per Epoch

8 Economic Implications

Financial statement misstatements carry far-reaching consequences not only for individual firms but also for the broader market and overall economic stability. From an internal perspective, firms that issue misstated financials face the risk of regulatory investigations, legal penalties, and reputational harm. These consequences can lead to a decline in firm value, elevated financing costs, and, in severe cases, delisting or bankruptcy. When misstatements become widespread, market participants lose confidence in the integrity of financial reporting, which can erode trust in capital markets as a whole and ultimately reduce market efficiency.

At a higher level, financial misreporting also distorts the allocation of resources in the economy. Investors rely on accurate statements to gauge a firm's true performance and growth prospects; when these statements are unreliable, capital can be misallocated to unproductive or unsustainable business operations. This misallocation dampens the potential for economic growth and may lead to systemic risks if such practices become pervasive. Moreover, the existence of misstatements often signals deeper governance and internal control weaknesses, raising concerns about the long-term resilience and stability of the affected firm.

The model proposed in our study aims to mitigate these risks by providing a more accurate prediction mechanism for potential misstatements. If regulators and external auditors can pinpoint high-risk companies early, they can allocate investigative resources more efficiently and potentially prevent large-scale misstatements before they occur. This proactive approach reduces both the direct economic harm to individual firms and the broader repercussions for investors, creditors, and other market participants. Additionally, our predictive framework can be integrated into firms' internal oversight procedures—for example, by flagging anomalies for further review or imposing additional audit controls—thereby enhancing corporate governance and reducing the likelihood of erroneous reporting.

By bridging the gap between advanced analytics and practical application, our model underscores the importance of prevention in securing the integrity of financial statements. Enhanced detection and timely intervention not only protect the interests of shareholders and stakeholders but also reinforce public confidence in the capital markets. This, in turn, fosters a healthier investment climate, encourages more efficient capital allocation, and upholds the standards of financial reporting necessary for a stable and prosperous economy.

9 Future Direction

We present an updated visualization to Fig. 1 with the further implementation of LLMs in our next steps.

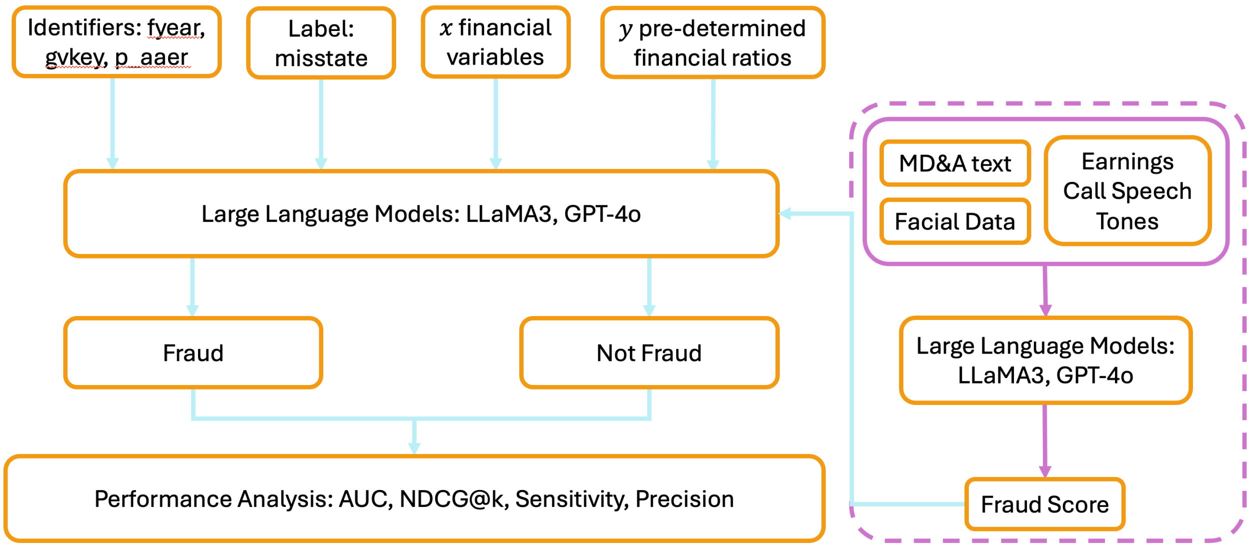

Identifiers: fyear, gvkey, p_aaer
Label: misstate
x financial variables
y pre-determined financial ratios
Large Language Models: LLaMA3, GPT-4o
Fraud
Not Fraud
Performance Analysis: AUC, NDCG@k, Sensitivity, Precision
MD&A text
Facial Data
Earnings Call Speech Tones
Large Language Models: LLaMA3, GPT-4o
Fraud Score

Figure 14: Figure of Future Experiments

9.1 Dataset Expansion

We plan to increase the dataset's scope into the current year of study, 2024. That would bring valuable training data and present an opportunity to test our fraud detection model on previously untested data. This would require lots of manual work by scraping off of the SEC AAERs webpage to update the dataset up until 2024, as the current dataset only goes until 2018. Additionally, we will leverage the Compustat database to fill in the financial variables for firms from 2018-2024. Once this dataset is complete, we hope to re-run the experiment and determine any improvements and hyperparameter tunings that can be made to the models. Below is a description of what new data could be used for:

9.1.1 Facial Data

There is no dataset as of the present that gathers facial expression data from earnings calls. We can extract facial expression data using computer vision packages and use this data as another input into our LLM. Fraud detection is quite difficult, so we plan to use this data to predict stock returns.

9.1.2 Linguistic Data

In addition, there have already been previous research in the field on linguistic data, gathering information from speech tones and word choices from conference calls. We can use these data in various ways, such as stock return or fraud detection. There are existing literature that discuss risk exposure to certain variables such as ESG, China, etc. We want to use LLM and similar methodologies used in this paper to make some predictions about those questions.

9.2 Large-Language Models (LLMs)

We plan to leverage LLMs to give a fraud score when given an input of financial statement text data (MD&A, earnings call transcript). This will become another feature inputted into a baseline's preliminary machine learning methods. We aim to use the LLMs and see if we can achieve performance gains by inputting our financial variables, ratios, and text data. Similarly to our machine learning approach, the model will classify a firm as either fraudulent or non-fraudulent, and then a performance analysis will be conducted to see if the results have improved or worsened or stayed the same. Currently, we only have tried to input text data into the LLM, not a combination of both. We hope to eventually reach this stage and publish as a state of the art model as there are very few studies that do both with the latest advances in deep learning research.

References

Bao, Y., B. Ke, B. Li, J. Yu, and J. Zhang (2020). Detecting accounting fraud in publicly traded u.s. firms using a machine learning approach. Journal of Accounting Research 58(1), 199–235.

Cecchini, M., H. Aytug, G. J. Koehler, and P. Pathak (2010). Detecting management fraud in public companies. Management Science 56(7), 1146–1160.

Chen, Y., B. T. Kelly, and D. Xiu (2022). Expected returns and large language models. https://ssrn.com/abstract=4416687. SSRN Working Paper.

Cressey, D. R. (1973). Other Peoples' Money: A Study in the Social Psychology of Embezzlement. Glencoe: Free Press.

Dechow, P. M., W. Ge, C. R. Larson, and R. G. Sloan (2011). Predicting material accounting misstatements. Contemporary Accounting Research 28(1), 17–82.

Dyck, A., A. Morse, and L. Zingales (2010). Who blows the whistle on corporate fraud? Journal of Finance 65(6), 2213–2253.

Fawcett, T. (2006). An introduction to roc analysis. Pattern Recognition Letters 27(8), 861–874.

Gee, J. and M. Button (2019). The financial cost of fraud 2019. Technical report, Crowe LLP. Technical Report.

Loukas, L., M. Fergadiotis, I. Androutsopoulos, and P. Malakasiotis (2021). Edgar-corpus: Billions of tokens make the world go round. In Proceedings of the Third Workshop on Economics and Natural Language Processing, pp. 13–18. Association for Computational Linguistics.

Rakoff, J. S. (2014, January). The financial crisis: Why have no high-level executives been prosecuted? The New York Review of Books. Jan. 9, 2014.

Appendix

Table 8: Reported Evaluation Metrics for Test Period 2003–08 (28 Financial Variables)

Method	**AUC**	**NDCG@k**	**Sensitivity**	**Precision**
(1) Logit	0.690	0.006	0.73%	0.85%
(2) RUSBoost	0.725	0.049	4.88%	4.48%
(3) SVM	0.680	0.016	1.69%	1.90%

Table 9: Percentage Change From Table VII to Table IV

Method	**AUC**	**NDCG@k**	**Sensitivity**	**Precision**
(1) Logit	-7.25%	350.00%	358.22%	234.71%
(2) RUSBoost	6.21%	4.08%	1.70%	13.82%
(3) SVM	-6.18%	50.00%	80.89%	34.84%