Financial Statement Item Prediction with SEC's EDGAR Data Using Machine Learning Models.

Khatansanaa Bandi (Nero)
Bachelor of Business Administration
SolBridge International School of Business
E-mail: kbandi222@student.solbridge.ac.kr
Student ID: 202203031

I. Abstract

Developed a financial statements' quarterly or annual directional prediction machine learning model (ML) using SEC's EDGAR data. Main purpose of the research is to (1) contribute to the literature of financial and accounting prediction, and (2) provide a data mining and utilization technique to work with the mass database of EDGAR.

(1) Prediction: The benchmark study used the same XBRL data and was able to outperform previous annual earnings prediction models significantly and performed with AUC/ROC of 67.52% to 68.66%. Our model, with added feature such as industry classification, feature engineering for time-series forecasting, and choice of quarterly or annual prediction was able to outperform the benchmark research model by significant degree with annual AUC/ROC of 76.4% and quarterly prediction with 82.28% - these are weighed average numbers since different industries have different predictive nature.

(2) Utilization: Although significant amount of study has been conducted on the XBRL Data initiative of SEC including its validity and implications, its application in quantitative analysis seems to lack more attention. Therefore, this paper includes the Python source code for the prediction from data collection to machine learning model development, providing techniques to utilize the public database.

II. Introduction

A. Financial Statement Prediction

Financial statements are a one of the crucial cornerstones of economy and finance industry, as it quantifies companies and their values as close as possible. Therefore, 17,693 publicly traded companies in the U.S. file various types of financial reports (e.g., 10-K, 10-Q, 8-K, etc.) multiple times a year.

With the XBRL initiatives of SEC, all those data are available for the public. From such massive database, numerous priceless insights can be driven, and one of the most valuable analysis would be to predict a given financial item of a company for next the next reporting period such as annually or quarterly.

1) Overview

According to the collected data EDA, on average, a company reports about 540 different financial items that includes from generic items commonly found in many companies such as Assets, Profit, all the way to extremely custom and specific items that is only found in small number of companies such as: Proceeds From Collection Of Other Loans Held For Sale, Premiums Receivable Allowance For Doubtful Accounts Additions For Charges To Expense.

Inside those financial items, there are few generic commonly found items such as Net Income Loss, Earnings Per Share, Gross Profit, Operational Cash Flow, which reflect priceless insights about the company. Therefore, being able to reliable predict those items for the future creates massive competitive advantage for companies to strategize internally or study the industry.

Therefore, this research created a model that can predict a given financial item for a given company for next quarter of annual report. For the sake of simplicity, the example in the paper will show the prediction of Net Income Loss of companies.

2) Importance

First of all, predicting financial items such as Net Income or EPS is a vital part of investment and fintech industry as they make investment decisions based on those data. Therefore, having a reliable method of predicting any given item will create valuable insights.

One of the main parts of business management, analytics, and business intelligence is to study the company, derive insights, make predictions and make decisions based on that prediction. Intracompany decisions such as resource allocation, or investment decide the faith of significant amount of resources and money, making the prediction and insights crucial part of the decision making that they have to be as accurate as possible. The model proposed in this paper was proven to have significant predictive power for this prediction problem.

For a company, it is a vital part of the business to be informed about their competitor's activities as well as the industries'. Having an ability to predict the industries' or competitor companies' future earning or operational cash flow provides valuable insights to the company and provides high degree of competitive advantage that allows them to stay competetive in the market.

3) Target Variable

For this research, used Net Income Loss because it was the most commonly found item to be used as target variable. Also, according to research, Net Income Loss is one of the most valuable items in a company's financial statements that can be utilized in many ways and if predicted, can create numerous values.

Previously, I was considering the Earnings Per Share Basic variable for the target variable. However, because of major missing values and different earnings per share variables that include diluted, basic, basic and diluted, distributed, and undistributed, it wasn't a good target variable, and because it is one of the most important thing in the training, target variable should be as reliable as possible.

B. SEC's EDGAR Database

1) Overview

The SEC's EDGAR Database, or the Electronic Data Gathering, Analysis, and Retrieval system, is the primary system for companies and others to submit documents under various U.S. securities laws. It serves as a comprehensive repository containing millions of filings, which benefits investors, corporations, and the U.S. economy by enhancing the efficiency, transparency, and fairness of the securities markets [15].

2) Data

EDGAR houses a vast array of filings, including annual and quarterly reports, registration statements, and other documents required by the SEC. The database allows for searches by company or fund name, ticker symbol, central index key (CIK), and more, providing access to over 20 years of filing data [16]. It also includes key mutual fund disclosures, mutual fund voting records, and other pertinent financial data [17].

Includes all submissions for all 17,693 companies. The types of data available online are: Submissions, Company Facts, Financial Statement Data (Including tag, sub etc.). The collected dataset includes 17,693 companies including all their reports (e.g., 10-K, 10-Q, 8K etc.) for all reported financial statement items by the company. Chose the top 28 industries (out of 434) that includes the greatest number of companies in the model development, which include 8163 companies out of the total reported in the database which is 17,693.

3) Usage & Purpose

Mostly SEC EDGAR Database is used for qualitative purposes where corporations or individuals need to retrieve certain submission data for a company. The database is freely accessible to the public, enabling research into a public company's financial information and operations through the filings made with the SEC. It is also used for researching information provided by mutual funds, exchange-traded funds (ETFs), variable annuities, and individuals. The EDGAR system processes about 3,000 filings per day and serves up 3,000 terabytes of data to the public annually [15].

4) Utilization in Training Quantitative Models

Research papers have utilized the EDGAR database for training machine learning predictive models. For instance, OpenEDGAR is an open-source Python framework designed to rapidly construct research databases based on EDGAR for use in both academic research and industrial applications [18]. Another study added a new dimension to corporate disclosure analysis by showing that the filing frequency of disclosures has a negative impact on future stock returns, using an AI methodology to process filings [19].

C. Model Overview

1) Collected Dataset

Main dataset comes from converting the 'Company Facts' bulk download which includes 17,693 JSON files each for a single company including all of their filings. The 'Submission' bulk download folder is utilized for deriving industry data for the company fact data.

2) Data Preparation & Model Development

One of the main contributions of the study is to introduce a technique to convert the SEC EDGAR data into machine readable tabular data. The data downloaded from SEC are usually in JSON or TXT file types, and extensive EDA, data preprocessing, and data mining techniques were needed to create a merged final tabular dataset that includes all companies, their submissions, and their tags. Therefore, exact steps in the data processing techniques as well as the EDA are mentioned in the Data section of Methodology.

When it comes to model development, to capture the time-series essence of the dataset, feature engineering techniques were used to reshape the dataset so that the machine learning models can find connection and pattern across historic data. For this research 2 tree based machine learning models (1) Random Forest Classifier, and (2) XGBoost Classifier have been used to predict the next reporting period's Net Income Loss direction.

Another new dimension the research paper contributes to the financial prediction literature is that the model is developed on each industry on the assumption that different industries have different financial accounting and reporting structure, and their predictability varies from industry to industry. This assumption has been proven after the model result came in where we can see clear and significant performance difference between different industries.

Some more regulated and highly structured industries such as 'State Commercial Bank' were most predictable due to similar financial reporting structure among companies which results in less missing data in the converted tabular dataset and more variables used in the prediction model, while other industries such as 'Metal Mining' were less predictable and performed poorer than other industries and the reason could be that because the companies inside the industries have company specific custom financial items which increases the amount of missing values in the dataset significantly.

III. Literature Review

A. Why is Financial Statement Prediction Important?

Financial statement prediction is crucial for multiple stakeholders including investors, analysts, and corporate managers. Accurate prediction of financial statements, particularly earnings, is essential because it directly influences stock prices and investment decisions. Predicting earnings helps in assessing a company's future performance and intrinsic value, enabling investors to make informed decisions that can lead to abnormal returns. As noted by Monahan (2018) [1], earnings are the fundamental determinant of equity and enterprise value, more informative than dividends or cash flows. Consequently, the ability to forecast earnings allows investors to identify mispriced securities, thereby gaining a competitive edge in the market [1][3].

For corporate managers, earnings forecasts are vital for strategic planning and resource allocation. Accurate predictions aid in performance evaluation and decision-making processes, enhancing overall financial management practices. Additionally, regulatory bodies like the SEC emphasize the importance of financial transparency and accurate reporting, which further underscores the need for precise earnings predictions [1][4].

B. *Current Researches on Financial Statement Predictions*

Recent studies have extensively explored financial statement prediction using various data sources and models. Traditional methods primarily relied on financial ratios and statistical techniques, while modern approaches increasingly utilize machine learning (ML) models to improve predictive accuracy.

Monahan (2018) [1] provided a comprehensive analysis of financial statement analysis and earnings forecasting, highlighting the use of financial ratios such as return on assets (ROA), return on equity (ROE), and profit margins to predict future earnings. These traditional models, though effective to some extent, have limitations in handling complex and large datasets.

Baranes and Palas (2019) [4] explored the use of support vector machines (SVM) for earnings movement prediction. Their study demonstrated that SVMs could significantly enhance predictive accuracy compared to traditional statistical methods. They used detailed financial statement data and focused on the application of machine learning techniques in financial prediction [4].

Chen et al. (2022) [2] conducted a seminal study on fundamental analysis using detailed financial data and machine learning approaches. They developed models using financial statement information and various machine learning algorithms, including neural networks and gradient boosting machines. Their results showed significant improvements in predictive performance over traditional methods, highlighting the potential of machine learning in financial forecasting [2][7].

C. *Current Researches that Used SEC EDGAR XBRL Data to Build Machine Learning Predictive Models*

The introduction of XBRL (eXtensible Business Reporting Language) by the SEC has revolutionized the way financial data is analyzed and utilized. XBRL filings are computer-readable and provide a rich source of detailed financial information that can be leveraged for predictive modeling.

Baranes and Palas [3] utilized XBRL data to predict earnings movements, showcasing the benefits of structured and detailed data in enhancing model accuracy. Their study emphasized the efficiency and effectiveness of XBRL data in financial predictions compared to traditional data sources like Compustat [3].

Rogers, Skinner, and Zechman (2017) [9] examined the impact of SEC EDGAR dissemination in a high-frequency trading environment. They highlighted the importance of timely and accurate financial data provided by XBRL filings in improving the speed and accuracy of financial analysis and predictions [9].

Lonare, Patil, and Raut (2021) [10] introduced an R package for retrieving and parsing SEC EDGAR filings, facilitating easier access to XBRL data for financial analysis. Their work underscores the increasing utilization of XBRL data in academic and practical financial research [10].

Williams (2015) [11] assessed the capability of XBRL filings in predicting future earnings, arguing that the structured nature of XBRL data allows for more precise and timely financial analysis, thereby improving the predictive performance of earnings models [11].

D. *Detailed Analysis of Chen et al. (2022)*

Chen et al. (2022) [2] is a benchmark study for this research, providing a foundational approach to using machine learning for financial statement predictions. Their study utilized detailed financial statement data and applied various machine learning algorithms to predict future earnings changes. The main highlights of their study include:

1) *Data and Methodology:*

The study used a comprehensive dataset of financial statements from public companies, incorporating detailed line items from balance sheets, income statements, and cash flow statements.

Machine learning algorithms such as neural networks, gradient boosting machines, and random forests were employed to analyze the data and predict earnings changes [2][7].

2) *Feature Engineering:*

Chen et al. emphasized the importance of feature engineering in improving model performance. They developed features based on financial ratios, industry classifications, and time-series data, which were crucial in capturing the underlying patterns in financial statements [2][7].

3) *Results:*

The models developed by Chen et al. outperformed traditional statistical methods in predicting future earnings changes. They achieved significant improvements in predictive accuracy, with AUC/ROC scores substantially higher than those of previous models [2][7].

The study demonstrated that machine learning models, when properly designed and trained on detailed financial data, could provide more reliable and accurate earnings forecasts [2][7].

4) *Implications:*

This study has profound implications for both academic research and practical applications. It highlights the potential of machine learning in transforming financial analysis and improving the accuracy of earnings predictions.

The use of detailed financial data and advanced modeling techniques can lead to better investment strategies and corporate decision-making processes [2][7].

IV. Methodology

Following are all the steps involved in training machine learning models starting from downloading the data from the SEC EDGAR website. Each steps include (1) approach for the technique, (2) justification of the approach, and (3) source code used to work with the data.

A. *Downloading the bulk folder*

The ‘Companyfacts’ zip file includes folder of 17,693 JSON files each representing a single publicly traded U.S. company including the history of its XBRL data from financial statements (forms 10-Q, 10-K,8-K, 20-F, 40-F, 6-K, and their variants). This ZIP file is updated and republished nightly at approximately 3:00 a.m. ET. Please use following link to access the EDGAR API Documentation and download the file: https://www.sec.gov/edgar/sec-api-documentation.

B. *Dividing by industry*

Before converting the JSON files, it is important to consider which files to convert. Converting all 17,693 files into one data frame is (1) computationally demanding, and (2) not efficient because there are magnitude of different companies with different financial items and structures.

For example, merging a commercial bank's financial statement items with a manufacturing companies' financial statement items, and developing a model across both companies to create a predictive model will make the model fundamentally inefficient because those types of companies are fundamentally different from each other. One way to deal with this problem is to divide the companies based on their industries.

1) *Industries EDA*

In the SEC EDGAR website, there are 2 different bulk download files. The first file is the 'companyfacts' zip file discussed in previous section. However, the other bulk download file is called 'submissions', and it includes all the company information across all companies. Within these submissions files, it shows the SIC number which is a unique number associated with different industries, and their industry name. Using this file, we can segregate the companies based on their industries.

TABLE I. 28 MOST POPULATED INDUSTRIES

SIC	Count	Description
NaN	2008	NaN
2834	1087	Pharmaceutical Preparations
6770	1045	Blank Checks
7372	653	Services-Prepackaged Software
6798	502	Real Estate Investment Trusts
1311	441	Crude Petroleum & Natural Gas
6022	416	State Commercial Banks
7389	400	Services-Business Services, NEC
2836	286	Biological Products, (No Diagnostic Substances)
3841	276	Surgical & Medical Instruments & Apparatus
3674	243	Semiconductors & Related Devices
7374	242	Services-Computer Processing & Data Preparation
1000	236	Metal Mining
6021	220	National Commercial Banks
6500	200	Real Estate
6035	195	Savings Institution, Federally Chartered
1040	189	Gold and Silver Ores
6221	163	Commodity Contracts Brokers & Dealers
7370	161	Services-Computer Programming, Data Processing, Etc.
4911	151	Electric Services
6199	146	Finance Services
8742	126	Services-Management Consulting Services
5812	124	Retail-Eating Places
8200	119	Services-Educational Services
7371	116	Services-Computer Programming Services
7373	111	Services-Computer Integrated Systems Design
6331	108	Fire, Marine & Casualty Insurance
4813	105	Telephone Communications (No Radiotelephone)
2860	102	Industrial Organic Chemicals

[a] Company count included for each industry.

As can be seen from the table, there are 2008 companies with no apparent industry label, and the most popular industries are (1) Pharmaceutical Preparations with 1087 companies and (2) Blank Checks with 1045 companies. As can be seen from the line graph below, there are lots of industries with significantly low number of companies inside (around 1-5), making the line look similar to the line graph of the number of missing values across all companies for unique variables.

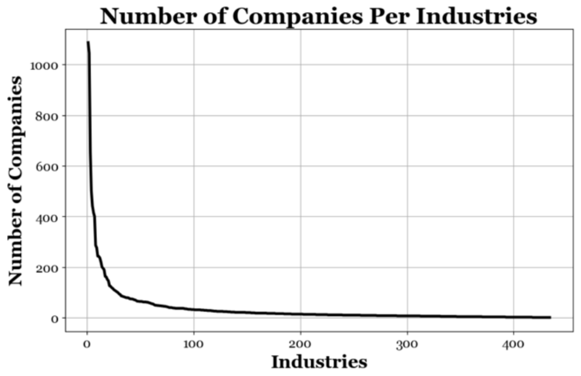

Fig. 1. Shows number of companies included in each industry.

This creates few problems. First, it makes it hard to choose which industries to choose in the model. If we choose only the industries with the most number of companies inside, then the model will be more robust, however, it will not take into account the less populated industries. Therefore, one approach for this could be to merge similar industries with each other. For example, there are multiple different 'furniture' making industries but named differently (e.g, Household Furniture, Retail Home Furniture, and Office Furniture etc.) Therefore, it might be possible to merge those industries together since they are all furniture manufacturing companies that must not be fundamentally different.

2) *Choosing files to convert*

First step in converting the files is to have the list of companies' 'CIK' numbers to convert and merge together. For example, if we want to merge together all the 'State Commercial Banks' industry's companies' financial statements together in one big data frame, the first step is to have all those companies' 'CIK' numbers stored in a list object. To do this, we will need to work with the 'Submissions' folder of the bulk download. The Submission folder is too, a collection of JSON files for all companies. After converting all the JSON files into a CSV file and choosing the necessary columns, following dataset is created:

TABLE II. CONVERTED SUBMISSIONS DATA

CIK	SIC	Name
815097	4400	CARNIVAL CORP
1438569	2834	Grifols SA
13156	7900	Galaxy Gaming, Inc.
14693	2080	BROWN FORMAN CORP
1069308	2834	Acer Therapeutics Inc.
1475115	7370	Eventbrite, Inc.

891478	6029	Banco Santander, S.A.
1086303	3845	Hongchang International Co., Ltd
311094	6021	WESTAMERICA BANCORPORATION
1757898	3842	STERIS plc
1408075	2650	GRAPHIC PACKAGING HOLDING CO
1691936	2000	STRYVE FOODS, INC.
1743971	7389	MOGU Inc.
5981	2870	AMERICAN VANGUARD CORP

[b] The table shows only 3 columns of the bigger converted submissions file. Submissions is originally a collection of mass JSON files in a given folder.

The converted 'Submissions' table includes all 17,693 companies and their information. With this table loaded, simply choose the CIKs of the selected industry and turn it into a list object in Python. In this case, the SIC number of State Commercial Banks is 6022, and there are 416 different companies inside.

After selecting the CIK list, it is necessary to convert them into file paths so that the code can loop through the folder to choose and convert the desired companies' files. For example, a company CIK '944745' must be converted into a file path that looks like: 'companyfacts/CIK0000944745.json'. To do this, a custom function called cik_to_file() has been created:

CODE 1. TURNING CIK TO FILE PATH

```
def cik_to_file(num_list):
    newlist = []
    for i in num_list:
        if len(str(i))==4:

newlist.append('companyfacts/CIK000000' +
str(i)+'.json')
        if len(str(i))==5:

newlist.append('companyfacts/CIK00000' +
str(i)+'.json')
        if len(str(i))==6:

newlist.append('companyfacts/CIK0000' +
str(i)+'.json')
        if len(str(i))==7:
            newlist.append('companyfacts/CIK000'
+ str(i)+'.json')
    return newlist
```

Fig. 2. After specifying industry SIC number, all the companies in the given industry can be retrieved by using their CIK number form the converted submissions dataset. After retreiving the CIK lists for each given industry, it is necessary to turn all of them into list of file paths inside the companyfacts folder to make sure the convertor code can iterate through them.

After running this function for desired industry, all the company's JSON files inside the folder will be saved as a JSON file path name inside the companyfacts folder, and it can be directly used to iterate through the folder and convert the files.

C. *Converting JSON files into tabular dataset*

1) *Choosing a key entry date for scraping*

Now that we have the necessary file paths to convert and merge into a single data frame, the next step would be to loop through the original 'companyfacts' folder to choose and convert the given company files into a tabular dataset and merge them into single data frame. However, significant amount of EDA and time have spent in this step to ensure the quality of the dataset. The main problem was that there were multiple entries for a financial statement item for a single filed time frame. For a given company, a given financial statement item looks like following table (EPS was chosen for example).

TABLE III. ITEMS INSIDE EACH ENTRY

start	end	val	accn	fy	fp	form	filed	frame
2008-01-01	2008-12-31	3.96	0000950123-11-014739	2010	FY	10-K	2011-02-16	CY2008
2009-01-01	2009-06-30	1.3	0000950123-10-067722	2010	Q2	10-Q	2010-07-23	
2009-04-01	2009-06-30	0.7	0000950123-10-067722	2010	Q2	10-Q	2010-07-23	CY2009Q2
2009-01-01	2009-09-30	2.31	0000950123-10-095292	2010	Q3	10-Q	2010-10-22	
2009-07-01	2009-09-30	1.01	0000950123-10-095292	2010	Q3	10-Q	2010-10-22	CY2009Q3
2009-01-01	2009-12-31	3.16	0000950123-11-014739	2010	FY	10-K	2011-02-16	
2009-01-01	2009-12-31	3.16	0001308179-12-000024	2011	FY	10-K	2012-02-15	CY2009
2010-01-01	2010-03-31	0.64	0000950123-11-038128	2011	Q1	10-Q	2011-04-22	
2010-01-01	2010-03-31	0.64	0001308179-12-000024	2011	FY	10-K	2012-02-15	CY2010Q1
2010-01-01	2010-06-30	1.6	0000950123-10-067722	2010	Q2	10-Q	2010-07-23	
2010-01-01	2010-06-30	1.6	0000950123-11-067480	2011	Q2	10-Q	2011-07-22	
2010-04-01	2010-06-30	0.96	0000950123-10-067722	2010	Q2	10-Q	2010-07-23	
2010-04-01	2010-06-30	0.96	0000950123-11-067480	2011	Q2	10-Q	2011-07-22	
2010-04-01	2010-06-30	0.96	0001308179-12-000024	2011	FY	10-K	2012-02-15	CY2010Q2
2010-01-01	2010-09-30	2.79	0000950123-10-095292	2010	Q3	10-Q	2010-10-22	
2010-01-01	2010-09-30	2.79	0000950123-11-091227	2011	Q3	10-Q	2011-10-21	
2010-07-01	2010-09-30	1.19	0000950123-10-095292	2010	Q3	10-Q	2010-10-22	
2010-07-01	2010-09-30	1.19	0000950123-11-091227	2011	Q3	10-Q	2011-10-21	
2010-07-01	2010-09-30	1.19	0001308179-12-000024	2011	FY	10-K	2012-02-15	CY2010Q3
2010-01-01	2010-12-31	3.61	0000950123-11-014739	2010	FY	10-K	2011-02-16	
2010-01-01	2010-12-31	3.61	0001308179-12-000024	2011	FY	10-K	2012-02-15	
2010-01-01	2010-12-31	3.61	0001308179-13-000027	2012	FY	10-K	2013-02-13	CY2010
2010-10-01	2010-12-31	0.82	0001308179-12-000024	2011	FY	10-K	2012-02-15	CY2010Q4
2011-01-01	2011-03-31	0.83	0000950123-11-038128	2011	Q1	10-Q	2011-04-22	

[c] Each entries inside given taxonomy includes these items inside from start to frame. This table shows an example of an EarningsPerShareBasic item for a given company. As can be seen, only small proportion of the entry includes frame with them, and only those entries with specified frames are used as a key date entry for convertor code.

As you can see, it is very messy because it involves all entries from all different reports. There are multiple start, end, filed dates for a same value while there are multiple different values for same dates for example. Also these dates vary greatly from different financial items to others. This is a big problem as there should be only one value per unique reporting date. And every other financial statement items and their value will be represented based on that date.

Therefore, it is important to choose a common key that can be used across all financial statement items. Therefore, the 'Frame' value is selected as the key universal date across all different items as it was the most reliable and unified format. Following is the official description of the Frame value reported in the SEC Website: 'The xbrl/frames API aggregates one fact for each reporting entity that is last filed that most closely fits the calendrical period requested.' As can be seen from the table, not all entries have the Frame label, and after choosing only the entries that include Frame label, the table will look like following:

TABLE IV. ENTRIES WITH SPECIFIED FRAMES

start	end	val	accn	fy	fp	form	filed	frame
2008-01-01	2008-12-31	3.96	0000950123-11-014739	2010	FY	10-K	2011-02-16	CY2008
2009-01-01	2009-12-31	3.16	0001308179-12-000024	2011	FY	10-K	2012-02-15	CY2009
2009-04-01	2009-06-30	0.7	0000950123-10-067722	2010	Q2	10-Q	2010-07-23	CY2009Q2
2009-07-01	2009-09-30	1.01	0000950123-10-095292	2010	Q3	10-Q	2010-10-22	CY2009Q3
2010-01-01	2010-12-31	3.61	0001308179-13-000027	2012	FY	10-K	2013-02-13	CY2010
2010-01-01	2010-03-31	0.64	0001308179-12-000024	2011	FY	10-K	2012-02-15	CY2010Q1
2010-04-01	2010-06-30	0.96	0001308179-12-000024	2011	FY	10-K	2012-02-15	CY2010Q2
2010-07-01	2010-09-30	1.19	0001308179-12-000024	2011	FY	10-K	2012-02-15	CY2010Q3
2010-10-01	2010-12-31	0.82	0001308179-12-000024	2011	FY	10-K	2012-02-15	CY2010Q4
2011-01-01	2011-12-31	4.47	0001308179-14-000038	2013	FY	10-K	2014-02-18	CY2011
2011-01-01	2011-03-31	0.83	0001308179-13-000027	2012	FY	10-K	2013-02-13	CY2011Q1
2011-04-01	2011-06-30	1.08	0001308179-13-000027	2012	FY	10-K	2013-02-13	CY2011Q2
2011-07-01	2011-09-30	1.38	0001308179-13-000027	2012	FY	10-K	2013-02-13	CY2011Q3
2011-10-01	2011-12-31	1.18	0001308179-13-000027	2012	FY	10-K	2013-02-13	CY2011Q4
2012-01-01	2012-12-31	5.05	0001628280-15-000853	2014	FY	10-K	2015-02-19	CY2012
2012-01-01	2012-03-31	1.06	0001308179-14-000038	2013	FY	10-K	2014-02-18	CY2012Q1
2012-04-01	2012-06-30	1.31	0001308179-14-000038	2013	FY	10-K	2014-02-18	CY2012Q2
2012-07-01	2012-09-30	1.47	0001308179-14-000038	2013	FY	10-K	2014-02-18	CY2012Q3

[d] Table shows the specific company's EarningsPerShareBasic item but only after choosing the entries that had specified Frame value.

As can be seen form the table, now there is no more duplicate dates or values, and the order is now chronological. To explain the values of the Frame label, following is the official description from the SEC Website: 'The period format is CY#### for annual data (duration 365 days +/- 30 days), CY####Q# for quarterly data (duration 91 days +/- 30 days), and CY####Q#I for instantaneous data.'

2) *Converting & merging using frame*

Finally, the next step is to convert and merge the JSON files using the Frame key label. To do this, a custom function convertor_multiple() has been created.

CODE 2. JSON TO TABULAR DATA FRAME CONVERTOR

```
def convertor_multiple(file_paths):
    all_dfs = []
    for idx, path in enumerate(file_paths, start=1):
        with open(path, 'r') as json_file:
            data = json.load(json_file)
        columns = set()
        values = {}
        try:
            cik = data['cik']
            entity_name = data['entityName']
            us_gaap_facts = data['facts']['us-gaap']
        except KeyError as e:
            pass
            #print(f"Key {e} missing in file {path}. Skipping this file.")
            continue
        for key, value in us_gaap_facts.items():
            for unit_key in value['units'].keys():
                for entry in value['units'][unit_key]:
                    try:
                        if 'frame' in entry:
                            frame = entry['frame']
                            if frame.endswith('I'):
                                frame = frame[:-1]
                            row_key = (frame, cik)
                            columns.add(key)
                            values.setdefault(row_key, {}).update({
                                'CIK': cik,
                                'EntityName': entity_name,
                                'Form': entry['form'],
                                'FP': entry['fp'],
                                'End': entry['end'],
                                'Filed': entry['filed'],
                                'Frame': frame,
                                key: entry['val']
                            })
                    except KeyError as e:
                        pass
                        #print(f"Key {e} missing in entry for {key}. Skipping this entry.")
        df = pd.DataFrame.from_dict(values, orient='index')
        all_dfs.append(df)
    merged_df = pd.concat(all_dfs, ignore_index=True)
    return merged_df
```

Fig. 3. This code is one of the most important code that was created for the research paper. This code gets the list of CIK file paths for specific industry from the previous function's output, and iterates over the bulk download companyfacts and converts and merges all the given JSON files given on the list into one merged data frame. As can be seen from the code, the 'Frame' key is used as the main date entry seperator.

The function takes a list of file paths as an input (explained in previous section), and converts them into tabular data and merges them in a single data frame. Following table shows an example of how the converted and merged dataset looks like:

TABLE V. CONVERTED JSON FILE

CIK	Frame	Assets	NetIncomeLoss
89800	CY2022Q1	21730400000	370800000
89800	CY2022Q2	22052800000	577900000
89800	CY2022Q3	22245800000	685100000
89800	CY2022Q4	22594000000	
89800	CY2023Q1	23129900000	477400000
89800	CY2023Q2	23166100000	793700000
89800	CY2023Q3	23004500000	761500000
93389	CY2022Q1	1310409000	19446000
93389	CY2022Q2	1326552000	19126000
93389	CY2022Q3	1318726000	8846000
93389	CY2022Q4	1254929000	
93389	CY2023Q1	1320020000	11918000
93389	CY2023Q2	1302145000	9137000
93389	CY2023Q3	1299930000	6659000
106532	CY2022Q1	261786000	4053000
106532	CY2022Q2	268760000	4495000
106532	CY2022Q3	310720000	10770000
106532	CY2022Q4	326620000	
106532	CY2023Q1	307709000	7445000
106532	CY2023Q2	287028000	4864000
106532	CY2023Q3	288860000	9337000

e. The table shows an example of 3 different company's (CIK: 89800, 93389, and 106532) 2 given financial items (Assets, NetIncomeLoss) from the converted data frame.

For the sake of simplicity, table shows only the data for 3 companies with CIK numbers (89800, 93389, and 106532), and only chosen example 4 different financial items. Moreover, the table only shows Frame labels that have been chosen (explained in the next section). As can be seen from the table, because Frame label is found across most of the financial statement items, using one unique frame across all financial items can retrieve good chunk of the data as it is found in most of the entries.

D. *Choosing selected frames*

Following code was used to (1) Choose only the quarterly report data, (2) choose only the frames after 2012, and (3) Drop all the missing values in the EarningsPerShareBasic column.

CODE 3. CHOOSING DESIRED REPORTS

```
def chosen_period(df,period):
    quarters = [ “CY2012Q1", "CY2012Q2",
"CY2012Q3", "CY2012Q4", "CY2013Q1", "CY2013Q2",
"CY2013Q3", "CY2013Q4", “CY2014Q1", "CY2014Q2",
"CY2014Q3", "CY2014Q4", “CY2015Q1", "CY2015Q2",
"CY2015Q3", "CY2015Q4", “CY2016Q1", "CY2016Q2",
"CY2016Q3", "CY2016Q4", "CY2017Q1", "CY2017Q2",
"CY2017Q3", "CY2017Q4", “CY2018Q1", "CY2018Q2",
"CY2018Q3", "CY2018Q4", "CY2019Q1", "CY2019Q2",
"CY2019Q3", "CY2019Q4", "CY2020Q1", "CY2020Q2",
"CY2020Q3", "CY2020Q4", "CY2021Q1", "CY2021Q2",
"CY2021Q3", "CY2021Q4", "CY2022Q1", "CY2022Q2",
"CY2022Q3", "CY2022Q4","CY2023Q1", "CY2023Q2",
"CY2023Q3", "CY2023Q4"]
    years =
["CY2012","CY2013","CY2014","CY2015","CY2016",
"CY2017","CY2018","CY2019","CY2020","CY2021", ”C
Y2022","CY2023"]
    if period == 'years':
        chosen = years
    if period == 'quarters':
        chosen = quarters
    if period == 'all':
        chosen = quarters + years
    df = df[df['Frame'].isin(chosen)]
    df = df.sort_values(['CIK','Frame'])
    return df
```

Fig. 4. This code gets 3 different inputs (1) years, (2) quarters, and (3) all, that retreives the desired reports and dates based on the given frame specified in the code. The frames include from 2012 Quarter 1 to 2023 Quarter 4. Because data quality was poor prior to 2012

To find out if there is correlation of missing values depending on the frame, I pivoted the sample dataset for unique frame values, counted their sum across all columns and plotted.

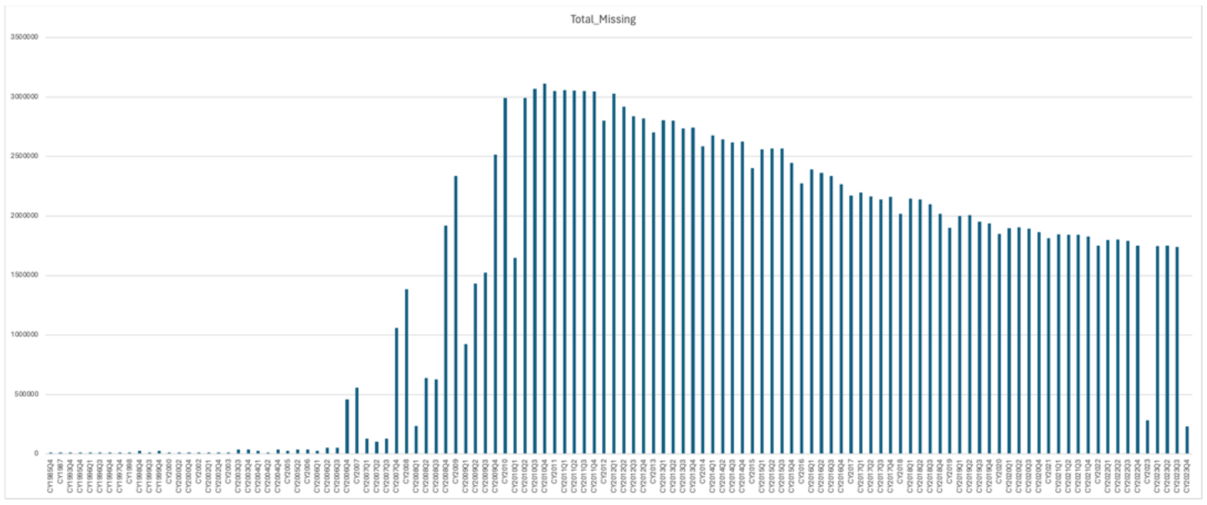

Fig. 5. Missing value frequency plotted against all frames provided in the dataset.

As can be seen from the plot, earlier reports have more missing values and it was decreasing over time. Next step would be to take into consideration the frequency count of unique frames. In following, I calculated the frequency count for each unique frames, and divided the total by the unique count and plotted.

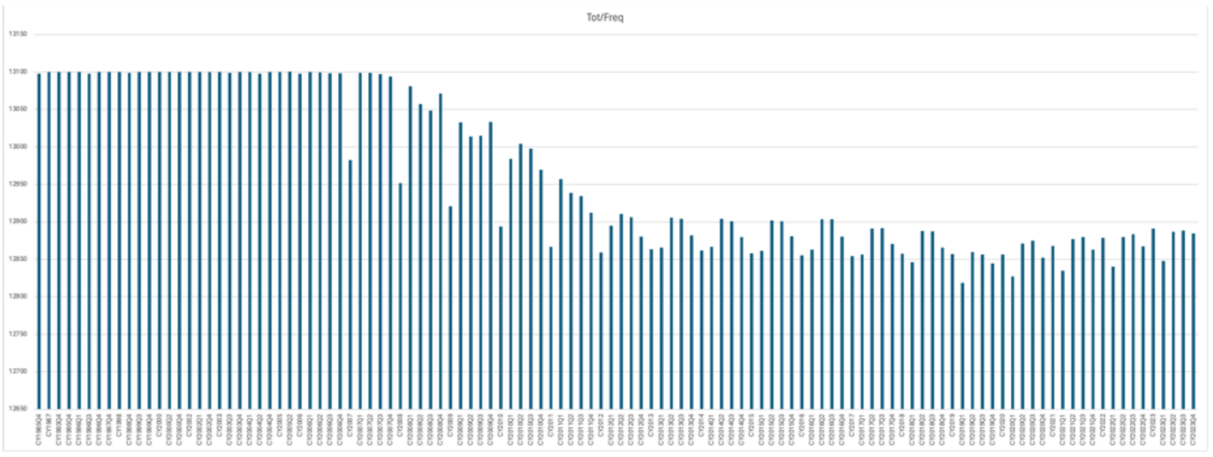

Fig. 6. Missing value frequency count for each unqiue frames divided by the unique count.

A simple conclusion can be drawn from this plot. Until year 2008, only 1 out of several thousand frames have been reported which shows that they are outliers that only one or two companies have reported until then. Therefore, those frames can be dropped.

Another interesting insight is that we can see a dramatic reduction in missing values have occurred from 2008 to 2012 and the missing values have been constant ever since 2012, and clear pattern can be seen: annual reports have the highest missing values followed by Q1 and Q4. Q2 and Q3 have been plotted to be most populated. Therefore, I decided to start the frames from CY2012.

E. Calculate missing target variable

a) Annual Reports (10-Ks)

After choosing only the annual reports, on average, 7 percent of Net Income Loss were missing, and within those missing values, it seems like all the missing values are either at the end or the start of the frames for given company. There are no missing values at the middle of the frames. Also, those missing values at the start or end, they seem to be chronologically ordered, meaning that simply dropping all those missing values wouldn't make inefficiencies in the time-series analysis.

b) Quarterly Reports (10-Qs)

For 10Qs however, depending on the company's fiscal year, one of the quarterly reports includes missing values and the total value of the year is reported in the Annual report. Therefore, a way to solve this is to deduct other 3 quarter's sum of net income loss from the annual report to calculate the missing quarter value for the net income loss. Following Function to calculate the missing quarterly Net Income Loss value.

CODE 4. MISSING QUARTER VALUE CALCULATOR

```
def calculate_missing_quarters(df):
    for cik in df['CIK'].unique():
        years = df['Frame'].str.extract(r'(CY\d{4})')[0].unique()
        for year in years:
            known_quarters_sum = df[(df['CIK'] == cik) & df['Frame'].str.startswith(year) &
                                    (df['Frame'].str.len() > 6) &
(~df['NetIncomeLoss'].isna())]['NetIncomeLoss'].sum()
            annual_frame = year
            annual_row = df[(df['CIK'] == cik) & (df['Frame'] == annual_frame)]
            if not annual_row.empty:
                annual_value = annual_row['NetIncomeLoss'].values[0]
                missing_count = df[(df['CIK'] == cik) & df['Frame'].str.startswith(year) &
                                   (df['Frame'].str.len() > 6) & (df['NetIncomeLoss'].isna())].shape[0]
                if missing_count > 1:
                    continue
                for quarter in ['Q1', 'Q2', 'Q3', 'Q4']:
                    quarter_frame = year + quarter
                    quarter_row = df[(df['CIK'] == cik) & (df['Frame'] == quarter_frame)]
                    if not quarter_row.empty and np.isnan(quarter_row['NetIncomeLoss'].values[0]):
                        missing_value = annual_value - known_quarters_sum
                        df.loc[(df['CIK'] == cik) & (df['Frame'] == quarter_frame), 'NetIncomeLoss'] =
missing_value
    return df
```

[f] When it comes to target variable such as NetIncomeLoss, it is not acceptable to have missing values. However, the dataset usually includes 1 missing value on a given quarter out of all 4 quarters. However, when the annual report value is provided, then it is possible to calculate the missing value for the given quarter. Above code calculates the missing NetIncomeLoss value for a given quarter if only 1 quarter is missing.

TABLE VI. RESULT AFTER CALCULATING MISSING QUARTERLY VALUE

CIK	EntityName	Frame	Original NetIncomeLoss	Calculated NetIncomeLoss
4828	AMERICAN CRYSTAL SUGAR CO /MN/	CY2012	548253000	548253000
4828	AMERICAN CRYSTAL SUGAR CO /MN/	CY2012Q1	212256000	212256000
4828	AMERICAN CRYSTAL SUGAR CO /MN/	CY2012Q2	149485000	149485000
4828	AMERICAN CRYSTAL SUGAR CO /MN/	CY2012Q3		-102967000
4828	AMERICAN CRYSTAL SUGAR CO /MN/	CY2012Q4	289479000	289479000
4828	AMERICAN CRYSTAL SUGAR CO /MN/	CY2013Q1	226001000	226001000
4828	AMERICAN CRYSTAL SUGAR CO /MN/	CY2013Q2	250767000	250767000
8504	AGEAGLE AERIAL SYSTEMS INC.	CY2012	741120	741120
8504	AGEAGLE AERIAL SYSTEMS INC.	CY2012Q1	-1081682	-1081682
8504	AGEAGLE AERIAL SYSTEMS INC.	CY2012Q2	3061726	3061726
8504	AGEAGLE AERIAL SYSTEMS INC.	CY2012Q3	-1296049	-1296049
8504	AGEAGLE AERIAL SYSTEMS INC.	CY2012Q4		57125
8504	AGEAGLE AERIAL SYSTEMS INC.	CY2013	1290453	1290453
8504	AGEAGLE AERIAL SYSTEMS INC.	CY2013Q1	20036	20036
8504	AGEAGLE AERIAL SYSTEMS INC.	CY2013Q2	716585	716585
8504	AGEAGLE AERIAL SYSTEMS INC.	CY2013Q3	-619902	-619902
8504	AGEAGLE AERIAL SYSTEMS INC.	CY2013Q4		1173734
8504	AGEAGLE AERIAL SYSTEMS INC.	CY2014	4573624	4573624
8504	AGEAGLE AERIAL SYSTEMS INC.	CY2014Q1	103634	103634
8504	AGEAGLE AERIAL SYSTEMS INC.	CY2014Q2	-1006805	-1006805
8504	AGEAGLE AERIAL SYSTEMS INC.	CY2014Q3	1787667	1787667

[g] As can be seen from the table across given 2 companies, the NetIncomeLoss value that was missing for every 3rd and 4th quarter have been calculated by subtracting the sum of given 3 quarters value from the annual report value.

In case the data include more than one missing quarterly values, the code simply skips it as they cannot be calculated. Following is an example of how it deals with such situations:

TABLE VII. CASE OF ERROR

CIK	EntityName	Frame	Original NetIncomeLoss	Calculated NetIncomeLoss
26324	CURTISS-WRIGHT CORPORATION	CY2021Q3	69703000	69703000
26324	CURTISS-WRIGHT CORPORATION	CY2021Q4		76579000
26324	CURTISS-WRIGHT CORPORATION	CY2022	294348000	294348000
26324	CURTISS-WRIGHT CORPORATION	CY2022Q1	40685000	40685000
26324	CURTISS-WRIGHT CORPORATION	CY2022Q2	70872000	70872000
26324	CURTISS-WRIGHT CORPORATION	CY2022Q3	73768000	73768000
26324	CURTISS-WRIGHT CORPORATION	CY2022Q4		109023000

26324	CURTISS-WRIGHT CORPORATION	CY2023Q1	56846000	56846000
26324	CURTISS-WRIGHT CORPORATION	CY2023Q2	80999000	80999000
26324	CURTISS-WRIGHT CORPORATION	CY2023Q3	96778000	96778000
34067	DMC GLOBAL INC.	CY2012		
34067	DMC GLOBAL INC.	CY2012Q1	2428000	2428000
34067	DMC GLOBAL INC.	CY2012Q2		
34067	DMC GLOBAL INC.	CY2012Q3		
34067	DMC GLOBAL INC.	CY2012Q4		
34067	DMC GLOBAL INC.	CY2013	6459000	6459000
34067	DMC GLOBAL INC.	CY2013Q1		
34067	DMC GLOBAL INC.	CY2013Q2		
34067	DMC GLOBAL INC.	CY2013Q3		
34067	DMC GLOBAL INC.	CY2013Q4		
34067	DMC GLOBAL INC.	CY2014	2567000	2567000
34067	DMC GLOBAL INC.	CY2014Q1	1487000	1487000
34067	DMC GLOBAL INC.	CY2014Q2	2316000	2316000
34067	DMC GLOBAL INC.	CY2014Q3	2342000	2342000
34067	DMC GLOBAL INC.	CY2014Q4	-3578000	-3578000

[h] In case where there are more than 1 quarters missing, then the quarterly value will be impossible to calculate. Therefore, the code is written in the way that when such situation occurs, the code simply ignores those missing values. Then those sequential missing values are dropped in the coming sections to make sure the sequential nature of the data is not disrupted.

Following is the first ever model built on quarterly data. This time, for the sake of simplicity, the 6022 is used.

TABLE VIII. RESULT OF CALCULATING MISSING QUARTERLY DATA (NETINCOMELOSS)

	Raw	Frame	Calculated
Total Row	4368	3787	3787
Missing Values	1399	1094	569
Missing Percentage	0.3202	0.2504	0.1302

[i] Raw columns is the number of rows from a freshly converted industry's converted dataset. Frame column represents the number of rows included after the dataset has been run through the "chosen_period" after choosing the desired frames. Calculated column refers to the number of rows and missing values for the NetIncomeLoss variable after the dataset has been processed by the calculate quarterly missing values code.

As can be seen, in the example of 6199, the freshly merged dataset includes 32 percent missing values in net income, and when chosen frames from 2012 to 2023, the number goes down to 25 percent. Still, for a target variable, 25 percent is a large amount. Therefore, after running the model, the new dataset with replaced missing values include 13 percent missing value. Almost 50 percent reduction in target missing variable.

F. *Choosing variables*

Across all 17,693 companies, total of 10,838 different financial items are found, and due lack of standardization, most of those items include highly custom items that are specific for given companies meaning that there are mass of financial tags that are available only for small amount of companies creating major missing value problem. Following table shows all 10,838 columns including their frequency count across all companies sorted in descending order.

TABLE IX. TOTAL COUNT OF UNIQUE FINANCIAL STATEMENT ITEMS

#	Key	Count
1	Assets	14753
2	LiabilitiesAndStockholdersEquity	14683
3	NetIncomeLoss	14434
4	NetCashProvidedByUsedInOperatingActivities	14416
5	NetCashProvidedByUsedInFinancingActivities	14344
6	StockholdersEquity	14013
7	CashAndCashEquivalentsAtCarryingValue	13859
8	RetainedEarningsAccumulatedDeficit	13571
9	NetCashProvidedByUsedInInvestingActivities	13531
10	Liabilities	12932
…	…	…
7414	InterimPeriodCostsNotAllocableAmountFirstItem	2
7415	OtherOwnershipInterestsCumulativeNetIncome	2
7416	ClosedBlockDividendObligationPeriodIncreaseD…	2
7417	PublicUtilitiesPropertyPlantAndEquipmentPlant…	2
7418	SharesSubjectToMandatoryRedemptionSettleme…	1
7419	AccumulatedOtherComprehensiveIncomeLossC…	1
7420	IndefiniteLivedTradeSecrets	1
7421	RestrictedInvestmentsNotExemptFromRegistrat…	1
7422	CapitalLeasesFutureMinimumPaymentsDueInR…	1

[j] As can be seen from the table, Assets and Liabilities and Stock Holders' Equity, as well as the Net Income Loss are most commonly and populated financial statement items across all 17,693 companies. Higher the counts are, more generic and mutually found those items are across different companies' financial statements.

There are total of 4061132 counts for these items. However these items are so imbalance that the items with frequency count of top 10 percent accounts for over 70 percent of the total counts of the dataset as whole. Therefore, before proceeding into next step, we have to solve the question of which variables to keep in the model.

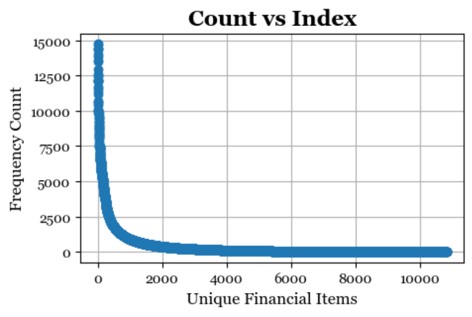

Fig. 7. The total count of unique financial statement items are plotted against their frequency count. The X axis has been sorted in descending manner.

The decision to divide the companies by industries is one way to deal with this problem. The underlying assumption is that companies within same industry might share similarities in their financial report structure and items that those companies might share more similar variables with each other. The datasets created for each industry proves that the assumption holds true as each industry includes on average about 4000 items or columns which proves that dividing companies by industry reduces the data redundancy.

However, this is not enough as within the 4000 variables inside a tabular dataset, still the majority of the dataset is missing values. Therefore, another solution was proposed which is to select only the most populated variables of a given industry of companies' dataset for the model. For example, when training a model for each industry, after converting and merging their data in a tabular dataset, only the given percentage of top populated variables were selected to train the model. To achieve this, following code is used to keep the given percentage of populated variables.

CODE 5. KEEPING MOST POPULATED COLUMNS

```
def na_max(df,threshold):
    missing_values = df.isnull().sum()
    missing_percentage = missing_values /
len(df)
    columns_to_drop =
missing_percentage[missing_percentage >
threshold].index
    df = df.drop(columns=columns_to_drop)
    return df
```

Fig. 8. This code takes a dataframe and a threshold value (0.8 for 80 percent) as input and keeps the columns that include columns that only include missing values of less than the given threshold. For example, putting 0.8 as threshold will keep only the columns from the dataframe that has columns that has missing values of less than 80 percent of them.

The usage of the code is that after converting and merging the industry dataset, we run the na_max() function on the industry dataset with specific percentage of missing values that are accepted. For example, running na_max(df, 0.8) code will keep only the columns that are less than 80 percent missing, and exclude other variables. Across all industries, after keeping the variables that have less than 80 percent missing values, on average, each industry include about 176 variables that are mostly populated.

The assumption is that more populated variables, those most commonly found financial items within the industry across the companies, are more significant in predicting the target variable. This assumption was proven to be valid as the performance of the model significantly increases when small number of more populated variables were chosen than using all the available variables within the industry dataset.

G. Calculating percentage difference

Before moving to next steps, it is essential to standardize the financial data of the companies as the size of those items across companies vary greatly depending on the company. For example, the Net Income Loss of a multi-billion dollar company will be significantly larger than those medium sized companies. Therefore, it is necessary to standardize all variable across the dataset to make sure the size of the company doesn't affect the model performance in unwanted way.

To standardize the variables, the model proposes to calculate the percentage difference of all variables and display all values as percentage difference compared to its previous period. In this way, it conveys much more meaningful information compared to raw reported monetary value.

TABLE X. GENERAL DATA FRAME FORMAT AND SHAPE AFTER CONVERSION AND MERGING

Company	**Year**	**Asset**	**Profit**	**EPS**
A	2010	12	4	0.14
	2011	11	3	0.13
	2012	13	5	0.15
	2013	12	2	0.18
B	2011	14	3	0.2
	2012	13	3	0.18
	2013	11	2	0.17
	2014	9	1	0.14
	2015	8	1	0.12
	2016	12	4	0.11
	2017	14	5	0.21

[k] The table shows an example of how the converted and merged data frame looks like and shaped like after the conversion and merge has been made.

Above table shows the main format of the dataset across different companies. To calculate and display the percentage difference, following code has been created:

CODE 6. PERCENTAGE DIFFERENCE CALCULATOR

```
def percent_diff(df):
    df_2 =
df[df.columns[2:]].pct_change(fill_method=None).
round(3)
    df_1 = df[df.columns[:2]]
    df = pd.concat([df_1,df_2],axis=1)
    pivoted = pd.pivot_table(df, index=['CIK',
'Frame'])
    rows_to_drop =
pivoted.groupby(level=0).head(1).index
    pivoted = pivoted.drop(rows_to_drop)
    pivoted = pivoted.reset_index()
    pivoted = pivoted.fillna(0)
    return pivoted
```

Fig. 9. Above code calculates the percentage difference of a given data frame and shows all of their value as a percentage difference. This code has been created as a way of standardizing the dataset.

After running the code, the result will look as follows. As can be seen from the table, all values are displayed as percentage difference. Consequently, because the code compares all variables with its previous period – the value above the given value – it creates one missing row for each companies. The code uses pivoting technique to divide the companies from each other in calculation to make sure it doesn't compare a company's values with its previous company in the dataset. Therefore, all 1st item in each company group in the pivoted dataset is dropped.

TABLE XI. RESULT OF CODE 6

Company	Year	Asset	Profit	EPS
A	2010			
	2011	-0.08	-0.25	-0.07
	2012	0.18	0.67	0.15
	2013	-0.08	-0.60	0.20
B	2011			

	2012	-0.07	0.00	-0.10
	2013	-0.15	-0.33	-0.06
	2014	-0.18	-0.50	-0.18
	2015	-0.11	0.00	-0.14
	2016	0.50	3.00	-0.08
	2017	0.17	0.25	0.91

[l] After running the percentage difference code, the example Table 10 will be presented like this.

H. *Reshaping the dataset*

The main approach of the model to predict the time-series data is by creating lagged feature. Because the model makes prediction for only the given variables for the given row, the row must include all the necessary information that might be necessary to make the time-series prediction. Therefore, the historical data of the financial items must be included in the row. Therefore, to achieve this, it is necessary to create given number of shifts for each row to make sure those rows include the given number of historic period data with itself.

At this point our dataset after creating the percentage difference is as follows:

TABLE XII. PERCENTAGE DIFFERENCE TABLE AFTER CODE 6

Company	Year	Asset	Profit	EPS
A	2011	-0.08	-0.25	-0.07
	2012	0.18	0.67	0.15
	2013	-0.08	-0.60	0.20
B	2012	-0.07	0.00	-0.10
	2013	-0.15	-0.33	-0.06

	2014	-0.18	-0.50	-0.18
	2015	-0.11	0.00	-0.14
	2016	0.50	3.00	-0.08
	2017	0.17	0.25	0.91

[m] Similar to Table 11.

In order to create the shifts, following code is used:

CODE 7. RESHAPING FUNCTION

```
def pivot_lag(df, level_1, level_2, shift):
    shift_periods = [-i for i in range(1, shift
+ 1)]
    pivoted = pd.pivot_table(df, index=[level_1,
level_2])
    new_columns = {}
    for col in pivoted.columns:
        for period in shift_periods:
            new_col_name =
f'{col}_{abs(period)}'
            new_columns[new_col_name] =
pivoted.groupby(level=level_1)[col].shift(period
s=period)
    pivoted = pd.concat([pivoted,
pd.DataFrame(new_columns, index=pivoted.index)],
axis=1)
    desired_order =
sorted(list(pivoted.columns))
    pivoted = pivoted[desired_order]
    pivoted = pivoted.reset_index()
    return pivoted
```

Fig. 10. This is another one of the most important functions created by the research. This code creates lagged features.

The level_1 input identify the first level of pivoting which in the example dataset is the 'Company' column, and the level_2 is the 'Year' column as they are the defining columns that the pivot must be made on. The 'shift' input takes an integer number and it determines the number of shifted values needed to make. After running the code with shift of 2, the dataset will be reshaped into following form:

TABLE XIII. RESHAPED DATA FRAME

Index	Asset	Asset_1	Asset_2	Profit	Profit_1	Profit_2	EPS	EPS_1	EPS_2
A			-0.08			-0.25			-0.07
		-0.08	0.18		-0.25	0.67		-0.07	0.15
	-0.08	0.18	-0.08	-0.25	0.67	-0.60	-0.07	0.15	0.20
	0.18	-0.08		0.67	-0.60		0.15	0.20	
	-0.08			-0.60			0.20		
B			-0.07			0.00			-0.10
		-0.07	-0.15		0.00	-0.33		-0.10	-0.06
	-0.07	-0.15	-0.18	0.00	-0.33	-0.50	-0.10	-0.06	-0.18
	-0.15	-0.18	-0.11	-0.33	-0.50	0.00	-0.06	-0.18	-0.14
	-0.18	-0.11	0.50	-0.50	0.00	3.00	-0.18	-0.14	-0.08
	-0.11	0.50	0.17	0.00	3.00	0.25	-0.14	-0.08	0.91
	0.50	0.17		3.00	0.25		-0.08	0.91	
	0.17			0.25			0.91		

[n] The table shows the Table 11 after processing by the Code 7.

As can be seen from the table, 2 shifts were made for each variables. Therefore, for each complete row, it includes 3 different years of value for each variables. As the shift creates missing values for each shift, depending on the number of shifts specified, there will be number of rows that include missing variables. To make sure a different company's financial items overlap into another company's, the code uses pivoting technique again to divide the dataset companies. Therefore, after the shift is made, the rows that include missing values are dropped, and the remaining rows are the complete rows that are ready to be trained for the machine learning model. In the table above, the highlighted rows are those complete rows that include all historic values for each variables, and those rows with missing values are dropped.

I. Creating target variable

After the shifts have been made and the dataset is reshaped, the next step will be to create the target binary variable. As the target variable for the given row must be the next period's data, simply shifting the last shift of the target variable one more time would be enough to create the target variable.

TABLE XIV. TARGET VARIABLE

Index	Asset	Asset_1	Asset_2	Profit	Profit_1	Profit_2	EPS	EPS_1	EPS_2	Target
										-0.07
			-0.08			-0.25			-0.07	0.15
A		-0.08	0.18		-0.25	0.67		-0.07	0.15	0.20
	-0.08	0.18	-0.08	-0.25	0.67	-0.60	-0.07	0.15	0.20	
	0.18	-0.08		0.67	-0.60		0.15	0.20		
	-0.08			-0.60			0.20			
										-0.10
			-0.07			0.00			-0.10	-0.06
		-0.07	-0.15		0.00	-0.33		-0.10	-0.06	-0.18
	-0.07	-0.15	-0.18	0.00	-0.33	-0.50	-0.10	-0.06	-0.18	-0.14
B	-0.15	-0.18	-0.11	-0.33	-0.50	0.00	-0.06	-0.18	-0.14	-0.08
	-0.18	-0.11	0.50	-0.50	0.00	3.00	-0.18	-0.14	-0.08	0.91
	-0.11	0.50	0.17	0.00	3.00	0.25	-0.14	-0.08	0.91	
	0.50	0.17		3.00	0.25		-0.08	0.91		
	0.17			0.25			0.91			

o. Similar to Table 13, however, a new 'Target' column has been added.

As can be seen from the table above, let's assume that we want to make EPS as a target variable. After shifting for 2 times, the last shift of EPS will be named EPS_2. Therefore, the code simply creates another shift for given variable, creating the 'Target' column shown in the table. Therefore, one more row for each company starts including missing variable, and those are dropped. Finally, the rows that doesn't include any missing value including the target variables (highlighted in the table) are kept for future steps to train the model on. Then the next step will be to make the newly created target variable binary. Following function is created to make such transformation.

CODE 8. BINARY TARGET & DROPPING MISSING VARIABLE

```
def target_dropna(df,lvl1,lvl2,last):
    df = pd.pivot_table(df, index=[lvl1,lvl2])
    df['target_lag'] = df.groupby(level=lvl1)[last].shift(-1)
    df['target'] = 0
    df.loc[df[last] < df[df.columns[-2]], 'target'] = 1
    df.loc[df[last] > df[df.columns[-2]], 'target'] = 0
    df = df.drop('target_lag',axis=1)
    df = df.reset_index()
    df = df.drop(lvl1,axis=1)
    df = df.drop(lvl2,axis=1)
    df = df.dropna()
    df = df.replace([np.inf, -np.inf, np.nan], 0).round(2)
    return df
```

Fig. 11. This code Pivots the dataset to create a binary target variable and dropping all the missing values that has been created by the process of processing and reshaping the datasets.

If the newly created target column is greater than its previous shift, the target variable will be 1, and it will be 0 if the new shift target column in less than the previous amount. In the given example, the highlighted section of the dataset will look as follows:

TABLE XV. ROWS TO BE PASSED TO TRAIN MACHIN LEARNING MODELS

Index	Asset	Asset_1	Asset_2	Profit	Profit_1	Profit_2	EPS	EPS_1	EPS_2	Target
	-0.07	-0.15	-0.18	0.00	-0.33	-0.50	-0.10	-0.06	-0.18	1
B	-0.15	-0.18	-0.11	-0.33	-0.50	0.00	-0.06	-0.18	-0.14	1
	-0.18	-0.11	0.50	-0.50	0.00	3.00	-0.18	-0.14	-0.08	1

p. Table shows the highlighted rows of the Table 14 as they are the only rows with no missing values due to reshaping. Therefore, these rows are the ones to be passed to training machine learning models.

J. Splitting dataset

1) Shuffling the dataset

For two reasons, the dataset needs to be shuffled. First, from 2012 to 2023, there has been various different business cycles and different pattern of economic performance. Therefore, splitting the dataset into train and test when the dataset is still in chronological order poses a type of overfitting risk. Because we already created shifted variables and accounted for the historical values, having the dataset in chronological order is not necessary.

Secondly, the industry datasets include hundreds to thousands of different companies, and they are sorted in ascending order by their CIK number. To make sure to train the model on all companies proportionally, it needs to be randomly shuffled. Therefore, the 'Sample' function of the SKLearn is used to shuffle the dataset.

2) Splitting the dataset

After the dataset is shuffled, it can be moved on to be split. For the sake of simplicity, test size of 0.3 used for splitting the dataset. The code simply uses the train_test_split() function of SKLearn:

CODE 9. DATASET SPLITTER

```
X_train, X_test, y_train, y_test = 
train_test_split(
        df.drop(target_name, axis=1), 
df[target_name], test_size=0.3, random_state=42
    )
```

Fig. 12. Regular SKLearn library ahs been utilized to split with 0.3 test size.

K. Training models

Now that the dataset is shuffled and split, it can be moved on to train the machine learning models. For the sake of simplicity, 2 tree based models (Random Forest and XGBoost) have been used. Following is the function to train the models:

CODE 10. TRAINING MACHINE LEARNING MODELS

```
def train_model(df, target_name, model_name):
    X_train, X_test, y_train, y_test = 
train_test_split(
        df.drop(target_name, axis=1), 
df[target_name], test_size=0.3, random_state=42
    )
    if model_name == 'RandomForest':
        model = 
RandomForestClassifier(random_state=42)
    elif model_name == 'XGBoost':
        model = XGBClassifier(random_state=42)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    roc_auc = roc_auc_score(y_test, 
model.predict_proba(X_test)[:, 1])
    return model, accuracy, roc_auc
```

Fig. 13. The code trains 2 different machine learning models specified by the input ('RandomForest', or 'XGBoost').

The function returns the accuracy and AUC/ROC performance metrics for both models, and following is the simple function for that.

CODE 11. EVALUATION FUNCTION

```
def evaluate_model(model, df, target_name):
    y_pred = model.predict(df.drop(target_name, 
axis=1))
    accuracy = accuracy_score(df[target_name], 
y_pred)
    roc_auc = roc_auc_score(df[target_name], 
model.predict_proba(df.drop(target_name, 
axis=1))[:, 1])
    return accuracy, roc_auc
```

Fig. 14. The code used for evaluating the result of the code such as creating AUC/ROC and Accuracy performance measurements.

L. Running the model

To run the models more efficiently, 2 functions were created: (1) everything() function including all the previous data preprocessing steps as well as training and evaluating model, and (2) everything_loop() model which loops through folder of datasets for each 28 industries.

1) Everything() function

Following function includes all the previous data preprocessing steps, as well as training and testing the model. One thing worth mentioning is that the function first splits the dataset into 2 different data frames with another 0.3 test size to create another 0.3 size unseen dataset to validate on. The 0.7 percent of the split dataset is further split inside the model training function to train and test to create the initial performances. After that, the function saves the model separately as a file, then loads it back again and trains it on the unseen data to validate the performance of the function.

CODE 12. EVERYTHING FUNCTION

```
def 
everything(df,report,lvl1,lvl2,shift,chunk,thres
hold,target_shift,testsize,model_name):
    if report == 'annual':
        df = chosen_period(df,'years')
    if report == 'quarter':
        df = chosen_period(df,'all')
        df = calculate_missing_quarters(df)
        df = chosen_period(df,'quarters')
    na_maxed = na_max(df,threshold)
    na_maxed = 
na_maxed.dropna(subset=['NetIncomeLoss'])
    na_maxed = na_maxed.drop(['End', 
'EntityName',   'FP',   'Filed', 
'Form'],axis=1)
    df_1 = percent_diff(na_maxed)
    merged = 
process_in_chunks(df_1,lvl1,lvl2,shift,chunk)

df_targeted=target_dropna(merged,lvl1,lvl2,targe
t_shift)
    df_shuffled = df_targeted.sample(frac=1, 
random_state=42)
    train_set, test_set = 
train_test_split(df_shuffled, 
test_size=testsize, random_state=42)
    df = train_set
    df2 = test_set
    rf_model, train_accuracy, train_roc_auc = 
train_model(df, 'target',model_name)
    feature_importances = 
rf_model.feature_importances_
    importances = pd.Series(feature_importances, 
index=df.columns[:-1])
    sorted_importances = 
importances.sort_values(ascending=False)
    save_model(rf_model, 'trained_model.joblib')
    rf_model_loaded = 
load_model('trained_model.joblib')
    unseen_accuracy, unseen_roc_auc = 
evaluate_model(rf_model_loaded, df2, 'target')
    df_shape = df_info(df)
    return df_shape, train_accuracy, 
train_roc_auc, unseen_accuracy, unseen_roc_auc, 
sorted_importances
```

Fig. 15. All the custom function for data processing has been integrated into this single code.

2) Everything_loop() function

The function was created to loop the Everything() function across the folder of dataset that includes 28 datasets for each industries. Moreover, the code creates integrated result for across the industries as well as the feature importances.

CODE 13. EVERYTHING FUNCTION ON LOOP

```
def everything_loop(directory_path, report,
lvl1, lvl2, shifts, chunks, threshold, target,
test_size,model_name):
    results_df = pd.DataFrame(columns=['File',
'Train Accuracy', 'Train ROC AUC', 'Unseen
Accuracy', 'Unseen ROC AUC', 'Shape'])
    feature_importances_df = pd.DataFrame()
    for i in os.listdir(directory_path):
        if i.endswith('.csv'):
            df =
pd.read_csv(os.path.join(directory_path, i))
            df_shape, train_accuracy,
train_roc_auc, unseen_accuracy, unseen_roc_auc,
feature_importances = everything(df, report,
lvl1, lvl2, shifts, chunks, threshold, target,
test_size,model_name)
            temp_results_df = pd.DataFrame({
                'File': [i],
                'Train Accuracy':
[train_accuracy],
                'Train ROC AUC':
[train_roc_auc],
                'Unseen Accuracy':
[unseen_accuracy],
                'Unseen ROC AUC':
[unseen_roc_auc],
                'Shape': [df_shape]
            })
            results_df = pd.concat([results_df,
temp_results_df], ignore_index=True)
            temp_feature_importances_df =
pd.DataFrame(feature_importances)
            temp_feature_importances_df['File']
= i
            feature_importances_df =
pd.concat([feature_importances_df,
temp_feature_importances_df], ignore_index=True)
    return results_df, feature_importances_df
```

Fig. 16. After creating the everything function, it was necessary to run the function on each 28 different industries and create integrated efficient results such as feature importance.

V. RESULTS

The financial statement prediction models developed in this study demonstrate significant improvements over the benchmark study by Chen et al. [2]. Detailed analysis of the results is presented below:

A. Annual Predictions

1) Random Forest Model:

Accuracy: The Random Forest model achieved an accuracy of 73.9% for predicting annual financial statements. This level of accuracy is a substantial improvement over traditional models and indicates a high reliability of the predictions.

AUC/ROC: The model achieved an AUC/ROC of 75%, which significantly outperforms the benchmark study's AUC/ROC range of 67.52% to 68.66%. This indicates that the Random Forest model has a better capability of distinguishing between different financial outcomes than the benchmark models.

2) XGBoost Model:

Accuracy: The XGBoost model demonstrated an accuracy of 74.8%, further indicating the robustness of this model in handling complex financial data.

AUC/ROC: With an AUC/ROC of 76%, the XGBoost model outperformed both the Random Forest model and the benchmark study, showing its superior predictive power. The higher AUC/ROC value suggests that XGBoost is more effective in identifying the true positive and true negative rates.

B. Quarterly Predictions

1) Random Forest Model:

AUC/ROC: The AUC/ROC value for the quarterly Random Forest model was 81%, which is significantly higher than the annual prediction. This suggests that quarterly data provides more granular and timely information, enhancing the model's predictive capability.

2) XGBoost Model:

AUC/ROC: The model's AUC/ROC value was 82%, the highest among all models tested. This substantial increase in predictive performance for quarterly data highlights the advantage of using more frequent reporting periods in financial prediction models.

3) Industry-Specific Models

The decision to divide companies by industry and develop specific models for each sector has proven to be highly effective. The industry-specific models provide insights into how different industries exhibit unique financial patterns and trends, which can be leveraged to improve prediction accuracy.

C. Analysis of Results

The significant outperformance of the quarterly models suggests that quarterly financial reports provide more timely and relevant information for predicting future financial outcomes. This could be due to the inclusion of more recent trends and less noise from seasonal variations compared to annual reports. The industry-specific models indicate that different sectors have unique financial behaviors that can be better captured when models are tailored to these specific patterns.

The overall higher performance of the XGBoost models across both annual and quarterly predictions indicates that this algorithm is particularly well-suited for financial statement prediction tasks. XGBoost's ability to handle sparse data and its robustness to overfitting likely contribute to its superior performance.

Moreover, the substantial improvement in AUC/ROC values from the industry-specific models emphasizes the importance of recognizing and incorporating industry-specific financial dynamics into predictive models.

The detailed insights gained from these tailored models can be invaluable for investors and analysts seeking to make informed decisions based on sector-specific financial health and trends.

TABLE XVI. INTEGRATE RESULT FOR 2 MODELS

SIC	Number of Companies	Industry	RandomForest				XGBoost			
			Annual		Quarter		Annual		Quarter	
			Accuracy	AUC/ROC	Accuracy	AUC/ROC	Accuracy	AUC/ROC	Accuracy	AUC/ROC
2834	1087	**Pharmaceutical Preparations**	74%	**78%**	76%	**83%**	75%	**81%**	76%	**83%**
6770	1045	**Blank Checks**	72%	**83%**	78%	**85%**	67%	**77%**	78%	**85%**
7372	653	**Services-Prepackaged Software**	74%	**75%**	73%	**80%**	68%	**73%**	73%	**81%**
6798	502	**Real Estate Investment Trusts**	65%	**72%**	75%	**81%**	68%	**75%**	74%	**82%**
1311	441	**Crude Petroleum & Natural Gas**	78%	**82%**	73%	**80%**	76%	**83%**	70%	**80%**
6022	416	**State Commercial Banks**	80%	**87%**	77%	**82%**	78%	**86%**	77%	**84%**
7389	400	**Services-Business Services, NEC**	63%	**72%**	74%	**80%**	63%	**70%**	74%	**81%**
2836	286	**Biological Products, (No Diagnostic Substances)**	78%	**78%**	76%	**82%**	74%	**78%**	76%	**84%**
3841	276	**Surgical & Medical Instruments & Apparatus**	69%	**73%**	73%	**79%**	72%	**79%**	74%	**81%**
3674	243	**Semiconductors & Related Devices**	68%	**73%**	71%	**78%**	71%	**76%**	72%	**80%**
7374	242	**Services-Computer Processing & Data Preparation**	68%	**75%**	74%	**80%**	75%	**80%**	74%	**81%**
1000	236	**Metal Mining**	65%	**66%**	71%	**77%**	65%	**66%**	71%	**79%**
6021	220	**National Commercial Banks**	72%	**82%**	77%	**82%**	74%	**83%**	76%	**84%**
6500	200	**Real Estate**	70%	**71%**	73%	**81%**	67%	**76%**	74%	**80%**
6035	195	**Savings Institution, Federally Chartered**	73%	**74%**	76%	**81%**	74%	**76%**	75%	**83%**
1040	189	**Gold and Silver Ores**	67%	**74%**	74%	**82%**	70%	**74%**	75%	**83%**
6221	163	**Commodity Contracts Brokers & Dealers**	78%	**78%**	75%	**84%**	71%	**79%**	75%	**83%**
7370	161	**Services-Computer Programming, Data Processing, Etc.**	63%	**63%**	70%	**75%**	66%	**69%**	71%	**79%**
4911	151	**Electric Services**	65%	**71%**	76%	**83%**	71%	**80%**	77%	**85%**
6199	146	**Finance Services**	69%	**74%**	72%	**79%**	75%	**83%**	71%	**80%**
8742	126	**Services-Management Consulting Services**	71%	**64%**	74%	**80%**	69%	**71%**	76%	**82%**
5812	124	**Retail-Eating Places**	67%	**63%**	73%	**80%**	68%	**63%**	75%	**83%**
8200	119	**Services-Educational Services**	65%	**63%**	71%	**79%**	60%	**65%**	76%	**83%**
7371	116	**Services-Computer Programming Services**	64%	**71%**	70%	**79%**	64%	**70%**	74%	**80%**
7373	111	**Services-Computer Integrated Systems Design**	58%	**64%**	70%	**77%**	73%	**78%**	70%	**78%**
6331	108	**Fire, Marine & Casualty Insurance**	67%	**63%**	72%	**80%**	67%	**64%**	76%	**84%**
4813	105	**Telephone Communications (No Radiotelephone)**	61%	**60%**	72%	**76%**	63%	**65%**	74%	**79%**
2860	102	**Industrial Organic Chemicals**	65%	**74%**	70%	**76%**	60%	**71%**	72%	**80%**
Total	**8163**	**Weighed Averages:**	**71%**	**75%**	**74%**	**81%**	**70%**	**76%**	**75%**	**82%**

q. This is the integrated result created by the both Random Forest and XGBoost models for both annual and quarterly data across all 28 different industries.

D. Conclusion

The study's financial statement prediction models, utilizing Random Forest and XGBoost algorithms, significantly outperform the benchmark study by Chen et al. [2]. The enhanced performance is primarily attributed to the use of quarterly data and the development of industry-specific models, which capture more detailed and timely financial information. These findings highlight the potential of advanced machine learning techniques and structured financial data in improving the accuracy and reliability of financial predictions. The study's approach offers valuable contributions to the literature and provides practical tools for stakeholders in the financial industry to make more informed decisions based on robust predictive models.

VI. Discussion

A. Importance of Quarterly Data Utilization

One of the most notable findings of this study is the superior performance of models trained on quarterly data compared to those trained on annual data. This result underscores the value of quarterly financial reports, which provide more frequent and up-to-date information on a company's financial health and operational trends.

B. Granularity and Timeliness:

Granularity: Quarterly data captures more detailed financial fluctuations within a year, providing insights into short-term trends that annual data might overlook. This granularity allows the models to detect subtle changes and patterns that are critical for accurate predictions.

Timeliness: Quarterly reports reflect more recent financial conditions, reducing the lag between financial events and the availability of data for analysis. This timeliness is crucial for making informed predictions and timely decisions.

C. Higher Predictive Accuracy:

The AUC/ROC values for quarterly predictions (81% for Random Forest and 82% for XGBoost) are significantly higher than those for annual predictions (75% for Random Forest and 76% for XGBoost). This indicates that quarterly data provides a more robust foundation for financial predictions, likely due to its ability to capture more immediate responses to economic and market conditions.

D. Impact of Industry-Specific Models

Another critical factor contributing to the improved performance is the development of industry-specific models. Financial behaviors and trends vary widely across different industries, and capturing these unique characteristics is essential for accurate predictions.

TABLE XVII. Training Datasets Shapes

SIC	Number of Companies	Industry	Annual (Used in Training Model)		Quarterly (Used in Training Model)	
			Rows	Columns	Rows	Columns
2834	1087	**Pharmaceutical Preparations**	2384	268	15144	238
6770	1045	**Blank Checks**	226	100	4231	130
7372	653	**Services-Prepackaged Software**	1162	352	7121	265
6798	502	**Real Estate Investment Trusts**	1194	376	7623	268
1311	441	**Crude Petroleum & Natural Gas**	703	319	4982	262
6022	416	**State Commercial Banks**	1105	577	6813	670
7389	400	**Services-Business Services, NEC**	653	355	4249	274
2836	286	**Biological Products, (No Diagnostic Substances)**	585	283	3905	235
3841	276	**Surgical & Medical Instruments & Apparatus**	669	337	4201	295
3674	243	**Semiconductors & Related Devices**	517	442	2918	370
7374	242	**Services-Computer Processing & Data Preparation**	373	334	2409	262
1000	236	**Metal Mining**	256	133	1715	157
6021	220	**National Commercial Banks**	569	655	3404	670
6500	200	**Real Estate**	270	196	1848	166
6035	195	**Savings Institution, Federally Chartered**	310	547	2282	589
1040	189	**Gold and Silver Ores**	207	190	1320	190
6221	163	**Commodity Contracts Brokers & Dealers**	333	91	2101	82
7370	161	**Services-Computer Programming, Data Processing, Etc.**	244	397	1403	289
4911	151	**Electric Services**	219	385	1338	316
6199	146	**Finance Services**	275	253	1500	169
8742	126	**Services-Management Consulting Services**	235	226	1591	199
5812	124	**Retail-Eating Places**	271	415	1708	298
8200	119	**Services-Educational Services**	214	346	996	247
7371	116	**Services-Computer Programming Services**	200	316	1194	247
7373	111	**Services-Computer Integrated Systems Design**	198	421	1323	316
6331	108	**Fire, Marine & Casualty Insurance**	312	607	1880	484
4813	105	**Telephone Communications (No Radiotelephone)**	161	352	1059	295
2860	102	**Industrial Organic Chemicals**	206	256	1277	247

r. The table shows the rows and columns shape for all 28 industries for both annual and quarterly datasets before getting fed into training the machine learning models. Interesting insights can be observed such as industries with larger number of companies having larger number of columns imply that those industries have large number of commonly shared financial statement items across companies inside the industry.

E. Tailored Models:

Industry Variability: Different industries have distinct financial structures, risk profiles, and operational cycles. By tailoring models to specific industries, this study leverages these unique characteristics, leading to more precise predictions. For example, the pharmaceutical industry (SIC 2834) showed higher AUC/ROC values with both annual and quarterly data compared to other industries.

Enhanced Predictive Power: The industry-specific models provide a deeper understanding of sector-specific financial dynamics, which is crucial for making accurate predictions. For instance, the Blank Checks industry (SIC 6770) exhibited an AUC/ROC of 84.8% for quarterly predictions with Random Forest, highlighting the effectiveness of specialized models.

F. Customized Feature Engineering:

By incorporating industry-specific financial ratios and metrics, the models can better capture relevant factors that drive financial outcomes in each sector. This customized feature engineering is a key component of the enhanced performance observed in the study.

G. Advanced Machine Learning Techniques

The use of advanced machine learning algorithms, such as Random Forest and XGBoost, played a significant role in the improved performance of the prediction models. These algorithms offer several advantages over traditional statistical methods.

1) Random Forest:

Robustness: Random Forest is known for its robustness and ability to handle large, complex datasets with numerous features. It mitigates overfitting by averaging multiple decision trees, leading to more stable and reliable predictions.

Interpretability: Although slightly less interpretable than linear models, Random Forest provides insights into feature importance, helping to identify the most influential financial indicators.

2) XGBoost:

Efficiency: XGBoost is an efficient and scalable implementation of gradient boosting that enhances predictive performance by iteratively improving the model through the minimization of errors. Its ability to handle sparse data and regularization techniques reduces overfitting.

Superior Performance: The higher AUC/ROC values achieved by XGBoost, particularly for quarterly predictions, indicate its superior ability to capture complex relationships within the financial data.

H. Contribution to Literature

This study makes several notable contributions to the literature on financial statement prediction:

Enhanced Predictive Models: The study demonstrates that using more frequent financial data (quarterly reports) and industry-specific models can significantly improve prediction accuracy. These findings provide a strong foundation for future research and practical applications in financial analysis and forecasting.

Utilization of SEC's EDGAR XBRL Data: The study highlights the value of SEC's EDGAR XBRL data in developing robust predictive models. By leveraging detailed financial disclosures, the research showcases how public financial data can be effectively used in machine learning applications.

Methodological Advancements: The inclusion of advanced machine learning techniques, such as Random Forest and XGBoost, and comprehensive feature engineering sets a new standard for financial prediction studies. This approach can be replicated and extended to other areas of financial analysis and forecasting.

Practical Implications: The provision of Python source code for the entire process, from data collection to model development, offers a practical resource for analysts and researchers. This transparency and reproducibility enhance the study's impact and utility.

I. Implications and Future Research

The findings of this study have several important implications for both academia and industry:

For Researchers: Future research can build on these findings by exploring additional machine learning algorithms and feature engineering techniques. Comparative studies across different industries and regions can further validate and extend the results.

For Practitioners: Financial analysts and investors can leverage the insights from this study to improve their predictive models and investment strategies. The use of quarterly data and industry-specific models can enhance the accuracy and reliability of financial forecasts.

For Policymakers: The study underscores the importance of timely and detailed financial disclosures. Policymakers can encourage more frequent and comprehensive reporting standards to facilitate better financial analysis and decision-making.

VII. Conclusion

This study demonstrates substantial advancements in financial statement prediction by leveraging SEC's EDGAR XBRL data and employing advanced machine learning models. The research shows that using quarterly data and developing industry-specific models significantly improves prediction accuracy, achieving AUC/ROC values as high as 82% for quarterly predictions with XGBoost. These findings underscore the critical importance of timely and detailed financial disclosures in capturing short-term trends and financial fluctuations that annual data might overlook.

The research highlights the benefits of tailored predictive models that account for the unique financial dynamics of different industries. By integrating industry-specific features and financial ratios, the models provide more precise and reliable predictions, as evidenced by the superior performance in sectors like pharmaceuticals and blank checks. The advanced machine learning techniques used, particularly XGBoost, further enhance predictive power by effectively capturing complex relationships within the data.

This study contributes to the literature on financial and accounting prediction by demonstrating the practical utility of SEC's EDGAR XBRL data and advanced machine learning methods. The provision of Python source code for the entire process, from data collection to model development, offers a valuable resource for financial analysts and researchers, promoting transparency and reproducibility.

For researchers, this study provides a foundation for further exploration into the use of detailed, high-frequency financial data and sophisticated predictive algorithms. For practitioners, the findings offer actionable insights to improve predictive models and investment strategies. For policymakers, the results highlight the value of encouraging more comprehensive and frequent financial reporting standards to facilitate better financial analysis and decision-making.

In conclusion, the study not only demonstrates significant improvements in financial prediction accuracy but also underscores the broader implications of utilizing advanced machine learning techniques and detailed financial disclosures in the field of financial analysis. This work sets a new standard for future research and practical applications, paving the way for continued innovations in financial prediction and analysis.

VIII. References

[1] Monahan, S. J. (2018). Financial statement analysis and earnings forecasting. Foundations and Trends® in Accounting, 12(2), 105-215.

[2] Chen, X., Cho, Y. H., Dou, Y., & Lev, B. (2022). Fundamental analysis of detailed financial data: A machine learning approach. Journal of Accounting Research, 60(2), 467-515.

[3] Baranes, A., & Palas, R. The Prediction of Earnings Movements Using Accounting Data: Based on XBRL.

[4] Baranes, A., & Palas, R. (2019). Earning movement prediction using machine learning-support vector machines (SVM). Journal of Management Information and Decision Sciences, 22(2), 36-53.

[5] Gordon, M. J. (1959). Dividends, earnings, and stock prices. The review of economics and statistics, 99-105.

[6] Charitou, A., & Charalambous, C. (1996). The prediction of earnings using financial statement information: empirical evidence with logit models and artificial neural networks. Intelligent Systems in Accounting, Finance & Management, 5(4), 199-215.

[7] Chen, X., Cho, Y. H., Dou, Y., & Lev, B. (2022). Predicting future earnings changes using machine learning and detailed financial data. Journal of Accounting Research, 60(2), 467-515.

[8] Gabriel, L. Predicting Stock Price Changes using EDGAR.

[9] Rogers, J. L., Skinner, D. J., & Zechman, S. L. (2017). Run EDGAR run: SEC dissemination in a high-frequency world. Journal of Accounting Research, 55(2), 459-505.

[10] Lonare, G., Patil, B., & Raut, N. (2021). edgar: An R package for the US SEC EDGAR retrieval and parsing of corporate filings. SoftwareX, 16, 100865.

[11] Williams, K. L. (2015). The prediction of future earnings using financial statement information: are XBRL company filings up to the task?.

[12] Fama, E. F. (1965). The behavior of stock-market prices. The journal of Business, 38(1), 34-105.

[13] Minasyan, R., & Erlandsson, P. (2023). Automatic Extraction of Financial Data in Credit Rating Analysis.

[14] Baranes, A., & Palas, R. Using XBRL Data to Predict Earnings Movements

[15] About EDGAR. U.S. Securities and Exchange Commission. (n.d.). Retrieved from https://www.sec.gov/edgar/about

[16] EDGAR-Search and Access. U.S. Securities and Exchange Commission. (n.d.). Retrieved from https://www.sec.gov/edgar/search-and-access

[17] Accessing EDGAR Data. U.S. Securities and Exchange Commission. (n.d.). Retrieved from https://www.sec.gov/os/accessing-edgar-data

[18] Bommarito, M. J., Katz, D. M., & Detterman, E. (2018). *OpenEDGAR: Open source software for SEC EDGAR analysis.* SSRN.

[19] Aldridge, I., & Li, B. (2022). A New Variable in Corporate Disclosure Analysis: An AI Study of the SEC EDGAR Database. *Available at SSRN 4032861.*