PREDICTING FINANCIAL DISTRESS WITH TIME-SERIES AND MARKET SENTIMENT INTEGRATION FOR SOLVING REAL WORLD PROBLEM - CAPITAL A BERHAD

BY

TAN ZHI QI

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology

(Kampar Campus)

February 2025

COPYRIGHT STATEMENT

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisor, Dr Tong Dong Ling who has given me this bright opportunity and guidance to engage in a project related to time series analysis and sentiment analysis on financial distress prediction. Besides, I would also like to appreciate my family's encouragement and support. It is my first step to establish a career in data analytic field. A million thanks to you.

ABSTRACT

During the COVID-19 pandemic, Capital A Berhad, previously known as AirAsia, encountered substantial financial difficulties. This raised concerns about its financial health and PN17 classification status. Hence, this study examines the company's financial distress by combining sentiment analysis with financial data using time-series methods. It investigates how market sentiment, drawn from news outlets and customer reviews, influences financial performance and highlights discrepancies between the Altman Z-score and the government's PN17 classification. Furthermore, the study analyzes the effects of restructuring announcements and market expansion efforts on the company's financial outcomes. Therefore, three forecasting approaches are compared: time-series analysis of market indices, company-specific financial data, and market sentiment analysis. In this stage, advanced techniques such as LSTM networks for financial data and market indices, alongside BERT model for sentiment analysis, are utilized to construct predictive models. The study follows the CRISP-DM framework, with performance assessed through metrics like mean squared error (MSE) and confusion matrices. This is to evaluate the model's accuracy and robustness. Thus, by outlining the strengths and weaknesses of each approach, this research offers valuable insights to internal auditors and decision-makers at Capital A Berhad, supporting enhanced risk management and financial forecasting practices.

Area of Study (Minimum 1 and Maximum 2): Deep Learning-Driven Business Forecasting, Financial Risk Analytics

Keywords (Minimum 5 and Maximum 10): Financial Distress Prediction, Risk Management, Sentiment Analysis, Time Series Forecasting, PN17 Classification, LSTM Networks

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

ζ	Zeta (Altman Z-Score)
A	Working Capital/Total Assets ratio
B	Retained Earnings/Total Assets ratio
C	Earnings Before Interest and Tax/Total Assets ratio
D	Market Value of Equity/Total Liabilities ratio
E	Total Sales/Total Assets ratio
N	Data Points

LIST OF ABBREVIATIONS

PN17	Bursa Malaysia's Practice Note 17/2005
ARIMA	Autoregressive Integrated Moving Average
EEMD	Ensemble Empirical Mode Decomposition
LSTM	Long Short-Term Memory
XAI	Explainable AI
IMF	Intrinsic Mode Functions
NSE	Nash-Sutcliffe Efficiency
MLP	Multi-Layer Perceptron
FYP	Final Year Project
NaN	Not a Number
No.	Number of
KNN	K-nearest neighbors
FCMI	Feature Correlation based Missing Data Imputation
BERT	Bidirectional Encoder Representations from Transformers
RoBERTa	Robustly Optimized BERT Pretraining Approach
FinBERT	Financial Bidirectional Encoder Representations from Transformers
CRISP-DM	Cross-Industry Standard Process for Data Mining
MAE	Mean Absolute Error
MSE	Mean Squared Error
SE	Squared Error
MD&A	Management Discussion & Analysis
MFRS 16	Malaysian Financial Reporting Standard 16
ROU	Right-of-Use

Chapter 1

Introduction

In this chapter, the background and motivation of the research, our contributions to the field, and the outline of the study shall be discussed.

1.1 Problem Statement and Motivation

As highlighted in the abstract, Capital A Berhad, formerly known as AirAsia, faced significant challenges during the COVID-19 pandemic. Hence, this study examines the company's financial distress by combining sentiment analysis with financial data using time-series analysis. The primary goal is to predict financial distress, as reflected in the government's PN17 status, and to analyze the impact of market sentiment—sourced from news outlets and customer reviews—on the company's financial performance.

Although the Altman Z-score consistently identifies Capital A Berhad as being in financial distress, its PN17 status suggests a contrasting perspective. This research aims to investigate this discrepancy by considering not only the company's financial metrics but also how announcements regarding PN17 status, restructuring initiatives, and market expansion efforts during the pandemic influence its performance.

The first motivation for this study stems from the importance of Capital A Berhad as a major player in Malaysia's aviation industry and its significant contribution to the regional economy. Its financial stability directly affects various stakeholders, including investors, employees, and the broader economic ecosystem. The second motivation arises from the observed discrepancy. This underscores the fact that traditional financial metrics often fail to account for non-financial factors. For instance, market sentiment, that can shape financial outcomes. Finally, the third motivation is the need to understand how sentiment correlates with financial performance. Thus, this will offer a more comprehensive view of financial distress.

By quantitatively analyzing changes in market sentiment and their correlation with the company's financial data, this study seeks to develop a predictive model for forecasting financial distress. The findings aim to provide deeper insights into the factors contributing to financial risk and offer a holistic perspective on financial health.

1.2 Objectives

The first two objectives are addressed in final year project 1, while the remaining two are tackled in final year project 2 which is the current project.

The first objective focuses on scraping and cleaning relevant data related to Capital A Berhad. This is to prepare it for subsequent data analysis. In fact, this involves collecting historical financial data, market indices, and sentiment-related textual data from various sources such as customer reviews. Therefore, the goal is to ensure that the dataset is clean, reliable, and structured to facilitate effective analysis using time-series and sentiment analysis techniques.

The second objective aims to investigate the financial risk of Capital A Berhad by employing both time-series analysis and sentiment analysis models. For instance, models like Long Short-Term Memory (LSTM) networks for time-series analysis and TextBlob for sentiment analysis. These models will help evaluate how market sentiment and financial data correlate to predict financial risk.

The third objective involves comparing the models used for time-series and sentiment analysis. This is to determine which approach provides the most accurate predictions of financial distress. Hence, this will include evaluating the performance of the models and their ability to handle different types of data, offering a deeper understanding of their strengths and limitations.

Finally, the fourth objective seeks to combine the results of the time-series and sentiment analyses to interpret the broader business implications of financial distress. Thus, by integrating insights from both models, the study will explore how sentiment and financial data together can offer a more comprehensive view of a company's financial health and its potential for recovery or further distress.

1.3 Project Scope and Direction

The scope of this study encompasses the collection and analysis of three main types of data. The first is market index data, including historical financial information for Capital A Berhad and related market indices. The second involves financial data, such as Capital A Berhad’s financial statements, including balance sheets, income statements, and cash flow statements. At the same time, sentiment data will be gathered. For instance, these sentiment data shall consist of customer reviews, news articles, and other textual content which are related to Capital A Berhad. Next, all data will undergo preprocessing to ensure high-quality, reliable datasets.

Additionally, the study focuses on developing and applying time-series analysis models, such as LSTM. This is to analyze market index and financial data, alongside implementing sentiment analysis techniques like Text Blob for textual data. It is also to be noted that the analysis period spans from 2004 to 2024. Thus, covering the time before, during, and after the pandemic. Next, the processed data from Capital A Berhad will be used with these models to evaluate their effectiveness in predicting past financial distress or stability. In fact, this evaluation will rely on metrics like Mean Squared Error (MSE) and a confusion matrix, classification report to assess the models' performance in forecasting future financial health. Lastly, the study will explore the correlation between sentiment data and economic performance to create a more accurate picture. Furthermore, it is important to note that the scope does not include optimizing the real-time implementation of the developed models or creating a fully automated forecasting system.

1.4 Contributions

This research contributes to understanding the financial health and performance of Capital A Berhad, particularly in the context of the challenges faced during the COVID-19 pandemic. By focusing on this prominent company in Malaysia’s aviation sector, the study provides valuable insights into how external factors, such as market sentiment and public perception, interact with traditional financial data to affect its stability. This is because Capital A Berhad plays a significant role in the regional economy, coupled with its status under financial distress as indicated by the Altman Z-score and its PN17 classification. Thus, this presents a unique case for exploring the limitations of conventional financial metrics and the importance of incorporating non-financial data sources into predictive models.

Besides, this study contributes to the broader body of knowledge regarding Capital A Berhad’s financial strategies and challenges, especially during the pandemic. This includes its efforts in restructuring and expanding into new markets. Therefore, by analyzing news articles, customer reviews, and how these efforts influenced the company’s financial trajectory, the study sheds light on the complex factors that drive a company's performance in times of crisis. This offers insights into how sentiment analysis can complement traditional financial analysis in assessing a company’s risk and stability. Hence, these findings have practical implications for investors, financial analysts, and policymakers who are interested in understanding the multifaceted nature of financial distress and recovery. This is essential in industries significantly impacted by global disruptions, such as the aviation sector.

Lastly, this study contributes to the field of financial distress prediction by performing a comparative analysis of time series analysis using market indices, time series analysis with financial data, and market sentiment analysis. Therefore, by examining these three approaches side by side, financial distress indicators and their significance can be identified. Besides, this comparison also fills a gap in existing research. For example, previous studies either focus on individual methods or compare market index-based predictions across sectors without targeting specific companies.

1.5 Report Organization

The details of this research are organized in the following chapters. In Chapter 1, the problem statement, objectives, scope, and contributions of the study are introduced. In Chapter 2, a literature review, covering Capital A Berhad formerly known Air Asia’s background, financial distress metrics, and past studies on time series and sentiment analysis, along with a comparative analysis of strengths and weaknesses is provided. In Chapter 3, an overview of the system model, including diagrams, equations, and a timeline is provided. This illustrates the integration of time series and sentiment analysis for financial performance evaluation. Next, Chapter 4, focuses on implementing system design, featuring block diagrams, hardware and software specifications, and machine learning workflows for decision trees, LSTM forecasting, and sentiment analysis using TextBlob, FinBERT, and BERT. In Chapter 5, the experimental setup, data extraction, preprocessing, imputation, and analysis, followed by time-series and sentiment modeling, combined results, and implementation challenges are described. In Chapter 6, the system's performance is evaluated, testing results are discussed, and project challenges, and objectives are assessed whether they are met. Finally, Chapter 7 concludes the research, summarizing key findings and providing recommendations for future improvements. Therefore, this structured approach ensures a comprehensive exploration of financial distress prediction through combined analytical methods.

Chapter 2

Literature Review

2.1 Capital A Berhad/Air Asia

2.1.1 Capital A Berhad/Air Asia Overview

Capital A Berhad, formerly AirAsia Group Berhad, is a Malaysian holding company. In fact, the company transitioned into a low-cost carrier in 2001 under Tony Fernandes's leadership. In general, the company primarily invests in and manages a diverse range of travel and lifestyle businesses [1]. It is to be noted that this introduction shall provide some context on the significant events that occurred during the period for a more comprehensive analysis.

During pre-pandemic 2019, AirAsia had established itself as the largest airline in Malaysia by fleet size and destinations with an operation of over 166 routes across 25 countries [1]. Regardless, AirAsia faced challenges in late 2019 due to rising operational costs and increasing competition within the aviation sector. Hence, the company is reported to have suffered net losses of approximately $66 million and have its liabilities exceeded its assets by $430 million [2].

As for during the pandemic in March 2020, the airline had to suspend most of its flights. Thus, it leads to substantial revenue losses. As a result, the company took steps to reduce costs. For instance, implementing layoffs and cutting compensation. Meanwhile, the company also accelerated its shift toward digital transformation and expanded its operations into e-commerce, logistics, and food delivery [2]. Fortunately, by 2023, the airline experienced a strong recovery in passenger traffic, surpassing pre-pandemic load factors. For instance, from January to June 2023, AirAsia Philippines achieved a 92% load factor, demonstrating robust demand for air travel [3].

In the post-pandemic period, as this report is being written, the company is in the process of selling its aviation business to AirAsia X Bhd (AAX) for RM6.8 billion to fortify both businesses and improve their financial situation. In fact, the goal of this action is to enable the business to leave Malaysia's PN17 category and improve it’s balance sheet. The company is reportedly intended to concentrate on its non-aviation endeavors after the sale, including digital services and logistics, with its logistics division, Teleport, already showing notable development [4].

2.1.2 Past Studies on Capital A Berhad

The studies done on Capital A Berhad/Air Asia, can mainly be divided into 2 categories. The first category consists of recent studies that had been done on the financial impact of the COVID-19 pandemic on Capital A, focusing on revenue losses, restructuring efforts, and recovery strategies.

For instance, during the COVID-19 pandemic, a study from Indonesia analyzed AirAsia Group Berhad's business decisions using SWOT (Strengths, Weaknesses, Opportunities, Threats) and PEST (Political, Economic, Social, Technology) frameworks. The study's focus was on the company's strategy to diversify digitally through its "AirAsia Digital" brand. This includes SuperApp, Teleport logistics, and BigPay fintech. In fact, the study explored how the pandemic significantly reduced airline revenues. For instance, AirAsia incurred losses of RM992.89 million in Q2 2020 and RM5.87 billion in Q4 2020. In contrast, by late 2020, AirAsia Digital is said to have contributed 42% to overall revenue. For instance, the AirAsia App reported a 15% increase in revenue, generating RM12 million, while BigPay reduced losses by 41%, and Teleport recorded an EBITDA of RM17 million. This highlights the success story of the company on diversifying its revenue. Unfortunately, while the study highlights AirAsia's ability to adapt and respond to challenges, it does not fully explore the broader effects of restructuring on the company’s overall performance or market confidence [5].

Another analysis by Simply Wall Street in 2024, done after the pandemic, had studied how Capital A Berhad, formerly AirAsia Group Berhad, recently saw its stock rise by 29% in a single month. The study reviewed that the quick jump suggests renewed interest from investors and the impressive 94% revenue increase which uncovers the company’s strong short-term performance. Thus, offering an insightful analysis on the company's current growth. However, it is notable that the stucy lacks a comprehensive evaluation of the company's broader financial health. Instead, the study focuses on short-term revenue improvements and the company's stocks without assessing the sustainability of AirAsia Digital and its competitiveness against established players [6]. The second category of studies done on the company consist of sentiment analysis using surveys which are related to Capital A Berhad's customer service, satisfaction, loyalty and price.

For instance, an Indonesian study provides a more advance survey on the impact of service quality, price, and customer satisfaction on customer loyalty among AirAsia airline customers. Despite surveying a total of 206 respondents using a traditional questionnaire, the data were analyzed using Structural Equation Modeling (SEM) with Lisrel 22 software. In fact, the study found that price was identified as the most influential factor in customer satisfaction. On the other hand, customer satisfaction itself had the most significant effect on customer loyalty. However, the data collected represents a snapshot at a single point in time. At the same time, it also limits insights into long-term customer behavior or changes in loyalty over time. Hence, this study can be further expanded by investigating the impact of all this towards the company's financial performance [7].

Besides, a Malaysian study examined how Capital A Berhad faced significant financial distress due to liabilities exceeding assets, declining flight demand, and border closures. Then, it provides an analysis on customer sentiment based on the survey of 100 Malaysians before and during this challenging period. Thus, highlighting the efforts by the company to secure funding through loans and the public perceptions towards the strategies. However, the study does not offer an in-depth analysis of long-term recovery strategies or how market sentiment might impact the company's financial performance [2].

2.2 Financial Distress Metrics

In this part of the literature review, there will be a brief explanation of the difference between Altman Z-Score and PN17. In general, the Altman Z-Score and PN17 (Practice Note 17) are both tools used to assess the financial health of companies. However, they serve distinctive purposes.

For instance, the Altman Z-Score is a financial formula developed by Edward Altman in 1968. It combines five key financial ratios using a weighted formula to produce a single score that indicates financial health. In terms of interpretation, a Z-Score above 2.99 indicates a low risk of bankruptcy, while a score below 1.81 suggests a high risk [8]. On the other hand, PN17, or Practice Note 17/2005, is a regulatory framework established by Bursa Malaysia. It is used to classify listed companies that are experiencing financial distress. In fact, the companies that fall under this classification may be at risk of delisting if they do not rectify their financial issues. For context, a company may be classified as PN17 if it meets any of the following criteria. The first is if shareholders' equity is less than 25% of total paid-up capital. The next is if Receivers have been appointed to manage the company's assets. The third is if the company has defaulted on loan payments. The fourth is when auditors express adverse opinions on financial statements. Lastly, it happens when the company has ceased operations or has no significant business activities [9].

The following table below shows a more comprehensive comparison of the two financial metrics based on the following reference [8][9].

Table 2.2 Comparison Table Between Altman Z-Score and PN-17 Classification

Aspect	**Altman Z-Score**	**PN17**
Purpose	Predicts bankruptcy risk	Classifies Malaysian companies in financial distress
Methodology	Quantitative score based on 5 key financial ratios	Qualitative criteria set by Bursa Malaysia
Outcome	Provides a score indicating risk level	Triggers regulatory actions and requirements
Applicability	Used broadly across various markets	Specific to companies listed on Bursa Malaysia

In the case of Capital A Berhad, formerly known as AirAsia Group, it was classified as a PN17 company based on three key criteria. Firstly, the company’s shareholders’ equity dropped below 25% of its issued and paid-up capital and was less than RM40 million in 2020. Secondly, Capital A faced payment defaults. This includes the failure to meet loan and financial obligations, which severely affected its liquidity and operational efficiency. Lastly, the company’s external auditor, Ernst & Young PLT, issued an unqualified audit opinion for the financial year ending December 31, 2019, signalling potential insolvency. Hence, all these factors combined led to Capital A’s designation as a PN17 company since July 8, 2020 [10].

2.3 Past Studies on Time Series Analysis

Due to the limited data points, LSTM model and ARIMA model will be applied and studied. This is because LSTM model can leverage its memory cell to retain important information overtime. Thus, allowing said model to learn from fewer observations [11]. On the other hand, ARIMA model can effectively model available data by using its statistical properties while also serving as a benchmark.

For instance, a study proposed predictive models using Ensemble Empirical Mode Decomposition (EEMD), Time series model such as Long Short-Term Memory (LSTM) networks, and Facebook's Prophet algorithm, combined with Explainable AI (XAI) techniques to predict high-risk financial environment much like current study. However, in this study, they use Boruta feature selection algorithm to identify the significance of each technical indicators to predict each financial stress variable. Then, Ensemble Empirical Mode Decomposition (EEMD) is used to decompose complex, nonlinear, and nonstationary time series into simpler subseries or IMFs. These IMFs will simplify the method to capture trends. This could be noted as the figure below shows one of the technical key indicators known as FSI which are undergoing EEMD to simplify the trends [12].

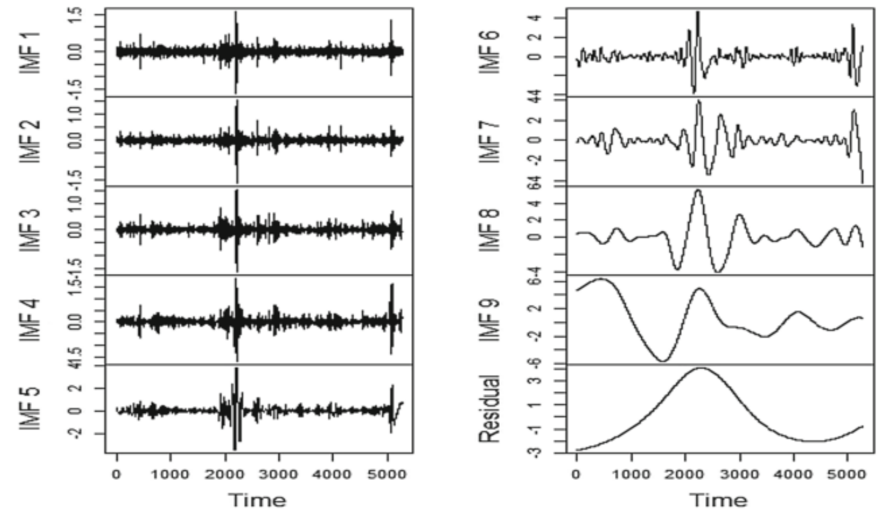


Figure 2.3.1 EEMD Application

Hence, the LSTM model will learn patterns and relationships from historical data to predict future values. In this study, the LSTM model uses memory cells to retain relevant information about past financial stress patterns while forgetting irrelevant data to prevent issues like "vanishing gradient". Finally, the models are evaluated using both static (one-day ahead) and dynamic (multi-step ahead) forecasting based on Nash-Sutcliffe Efficiency (NSE), Index of Agreement (IA) and Theil's Inequality Coefficient (TI). Furthermore, the paper conducts a rigorous comparison between EMD-LSTM and EEMD-Prophet models against traditional predictive models such as ARIMA, SARIMA, Bayesian Structural Time Series Forecasting (BSTSF), and a simple Multi-layer Perceptron (MLP) model with EEMD-LSTM being the best in financial forecasting in original dataset and during pandemic dataset. The comparison could be seen as below [12].

Table 2.3.1 Performance Comparison between Time Series Model for Financial Prediction

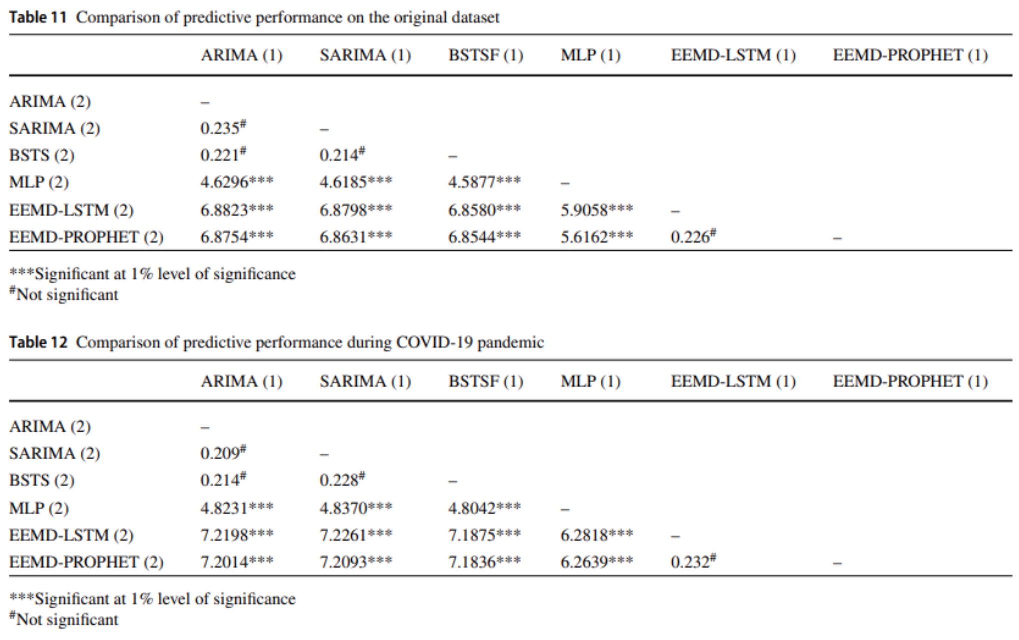


Table 11 Comparison of predictive performance on the original dataset

	ARIMA (1)	SARIMA (1)	BSTSF (1)	MLP (1)	EEMD-LSTM (1)	EEMD-PROPHET (1)
ARIMA (2)	–					
SARIMA (2)	0.235#	–				
BSTS (2)	0.221#	0.214#	–			
MLP (2)	4.6296***	4.6185***	4.5877***	–		
EEMD-LSTM (2)	6.8823***	6.8798***	6.8580***	5.9058***	–	
EEMD-PROPHET (2)	6.8754***	6.8631***	6.8544***	5.6162***	0.226#	–

***Significant at 1% level of significance
#Not significant

Table 12 Comparison of predictive performance during COVID-19 pandemic

	ARIMA (1)	SARIMA (1)	BSTSF (1)	MLP (1)	EEMD-LSTM (1)	EEMD-PROPHET (1)
ARIMA (2)	–					
SARIMA (2)	0.209#	–				
BSTS (2)	0.214#	0.228#	–			
MLP (2)	4.8231***	4.8370***	4.8042***	–		
EEMD-LSTM (2)	7.2198***	7.2261***	7.1875***	6.2818***	–	
EEMD-PROPHET (2)	7.2014***	7.2093***	7.1836***	6.2639***	0.232#	–

***Significant at 1% level of significance
#Not significant

Hence, this study contributed to the combination of two hybrid predictive frameworks. For example, the combination of Ensemble Empirical Mode Decomposition (EEMD) for time series decomposition and Long Short-Term Memory (LSTM) networks. The study had also implemented the use of Explainable AI (XAI), such as Permutation Feature Importance and LIME (Local Interpretable Model-agnostic Explanations) to enhance interpretability for LSTM which is crucial for decision making**.** As mentioned in the paper, this is something that is not widely used. Besides, the study had also validated the robustness of the LSTM model against other time series models making it one of the most comprehensive studies done on financial distress forecasting [12]. In fact, most studies apply LSTM model and ARIMA model for stock forecasting. For instance, a study investigates a hybrid forecasting model combining LSTM model and ARIMA model for algorithmic investment strategies. In fact, the hybrid model, referred to as LSTM-ARIMA, was tested against standalone LSTM and ARIMA models across three major equity indices: S&P 500, FTSE 100, and CAC 40. In fact, the models were evaluated based on their ability to predict the next day's closing price and generate profitable buy/sell signals under various investment strategies. In the end, the study confirms the superior performance of LSTM-ARIMA in most scenarios [13]. Next, another study evaluates the performance of ARIMA and LSTM models in forecasting Apple's stock closing prices over a multi-year period. Once again, the result indicates that the LSTM model is superior. This is because it achieves higher predictive accuracy measured by RMSE. However, it is to be noted that both models lack the ability to incorporate external factors, such as economic or political events which may cause fluctuations. Therefore, the paper highlights the need for hybrid approaches to improve predictions in complex financial markets [14]. Additionally, the superiority of applying LSTM model and ARIMA model in financial forecasting is further cement by the third paper. For instance, this study aims to compare the predictive capabilities of ARIMA, LSTM, and Transformer models in forecasting stock prices of three Moroccan credit companies listed on the Casablanca Stock Exchange known as EQD, LES, SLF. As usual, the LSTM model outperformed ARIMA and Transformer in forecasting accuracy, achieving R^2 values exceeding 0.99 for EQD and LES, and 0.95 for SLF. However, it is worth noting that the study affirms that ARIMA model performed reasonably well for linear relationships but lagged in capturing complex patterns [15]. In conclusion, the three studies here have limited exploration of external economic factors affecting predictions despite their novelties.

2.4 Past Studies on Sentiment Analysis

Due to time constraints, the Text Blob model and BERT model will be applied and studied. This is because the Text Blob model is ideal for its quick and simple analysis which can be used as a benchmark whereas BERT model provides a powerful framework for complex analysis. For context, Text Blob model returns polarity and

subjectivity of a sentence. For instance, the polarity score lies between [-1,1], with -1 defines a negative sentiment and 1 defines a positive sentiment. This is because Text Blob model has semantic labels that help with fine-grained analysis [16]. As for the BERT model, it processes text sequentially, either from left to right or right to left. This means that BERT model looks at all the words in a sentence simultaneously [17].
In fact, most studies had applied BERT model to conduct sentiment analysis for the finance field. For instance, a study explores sentiment analysis in financial markets. Mostly focusing on analyzing textual financial data such as news articles, earnings reports, and market commentary. In fact, it combines traditional lexicon-based sentiment analysis using the Loughran-McDonald dictionary with advanced machine learning techniques using the BERT model. In the end, the study reaffirms the superior performance of BERT for financial sentiment analysis, achieving high accuracy (90%) compared to traditional methods [18]. Finally, another study examines financial forecasting by comparing sentiment analysis (SA) and technical analysis (TA) indicators using a genetic programming (GP) algorithm. The goal is to determine whether sentiment analysis features (derived from the text, titles, and summaries of articles) can outperform technical analysis. In the end, the study shows that SA outperforms TA for some companies but not others. Thus, raising questions about sectoral or contextual influences on the model's performance [19].

2.5 Strength and Weakness Comparison

In conclusion, previous studies focus on either sentiment analysis or time series analysis for financial distress prediction, with most study on Capital A Berhad directed on sentiment analysis or from investor’s perspective. The following table below shows the strength and weakness of each paper mentioned in the literature review in comparison to current study.

Table 2.5 Strength and Weakness Comparison Table Between Papers in Literature Review

	Past Studies on Capital A Berhad/Air Asia	**Past Studies on Time Series Analysis**	**Past Studies on Sentiment Analysis**	**Current Study**

	Category 1 [5] [6]	**Category 2 [7] [2]**	**[12]**	**[13] [14] [15]**	**[18] [19]**	
Specifically Directed at Capital A Berhad	Yes	Yes				Yes
Involved Company Financial Data	Yes		Yes			Yes
Involved Market Index	Yes		Yes	Yes		Yes
Involved Company Stock	Yes		Yes	Yes		Yes
Involved Explainable AI			Yes			No
Involved Customer Reviews		Yes				Yes
Involved News and social media					Yes	No
Span over pre, during and post pandemic		Yes	Yes		Yes	Yes

CHAPTER 3

3.1 System Model Diagram

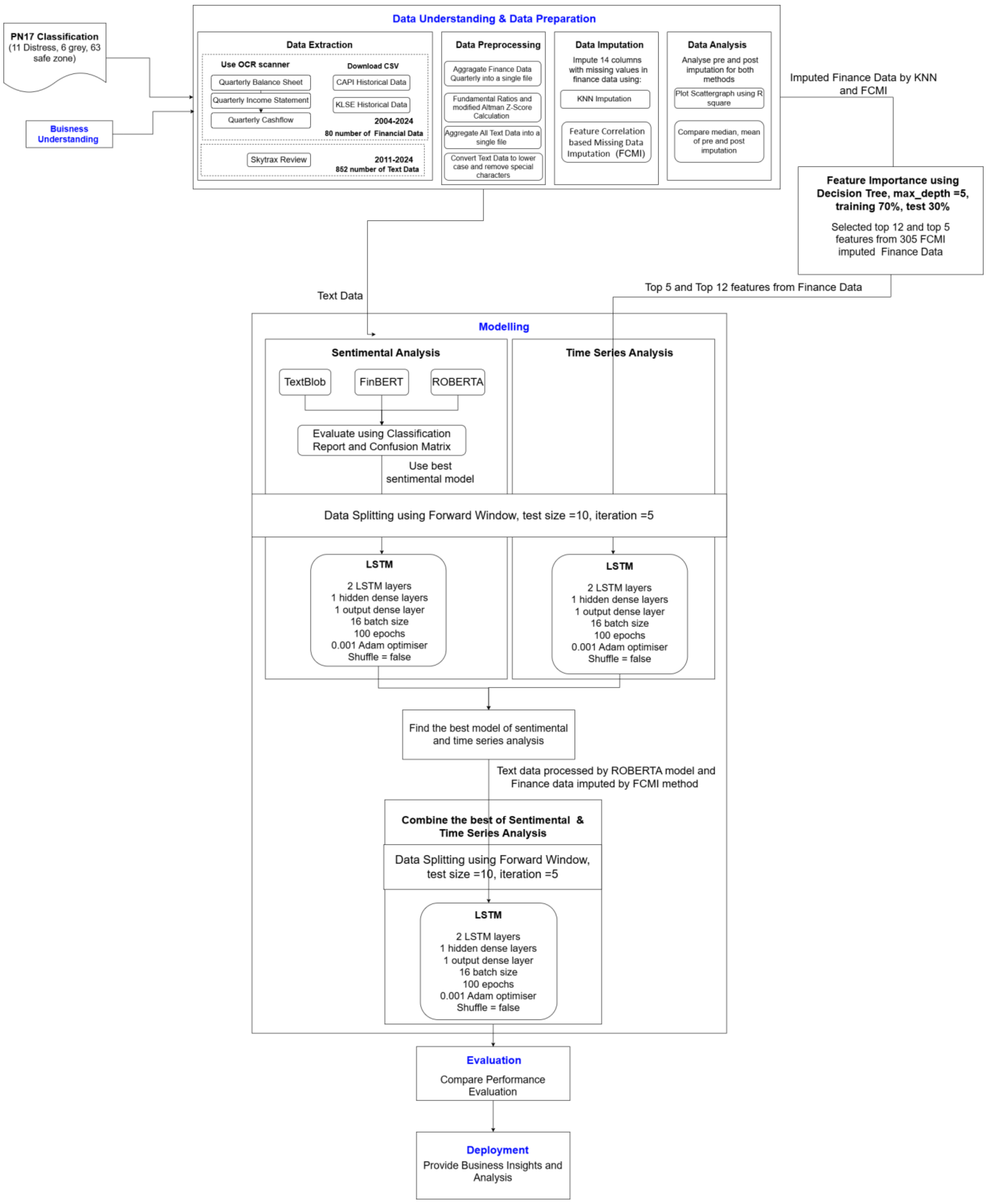

PN17 Classification
(11 Distress, 6 grey, 63 safe zone)
Buisness Understanding
Data Understanding & Data Preparation
Data Extraction
Use OCR scanner
Quarterly Balance Sheet
Quarterly Income Statement
Quarterly Cashflow
Download CSV
CAPI Historical Data
KLSE Historical Data
2004-2024
80 number of Financial Data
Skytrax Review
2011-2024
852 number of Text Data
Data Preprocessing
Aggragate Finance Data Quarterly into a single file
Fundamental Ratios and modified Altman Z-Score Calculation
Aggregate All Text Data into a single file
Convert Text Data to lower case and remove special characters
Data Imputation
Impute 14 columns with missing values in finance data using:
KNN Imputation
Feature Correlation based Missing Data Imputation (FCMI)
Data Analysis
Analyse pre and post imputation for both methods
Plot Scattergraph using R square
Compare median, mean of pre and post imputation
Imputed Finance Data by KNN and FCMI
Feature Importance using Decision Tree, max_depth =5, training 70%, test 30%
Selected top 12 and top 5 features from 305 FCMI imputed Finance Data
Top 5 and Top 12 features from Finance Data
Text Data
Modelling
Sentimental Analysis
TextBlob
FinBERT
ROBERTA
Evaluate using Classification Report and Confusion Matrix
Use best sentimental model
Time Series Analysis
Data Splitting using Forward Window, test size =10, iteration =5
LSTM
2 LSTM layers
1 hidden dense layers
1 output dense layer
16 batch size
100 epochs
0.001 Adam optimiser
Shuffle = false
LSTM
2 LSTM layers
1 hidden dense layers
1 output dense layer
16 batch size
100 epochs
0.001 Adam optimiser
Shuffle = false
Find the best model of sentimental and time series analysis
Text data processed by ROBERTA model and Finance data imputed by FCMI method
Combine the best of Sentimental & Time Series Analysis
Data Splitting using Forward Window, test size =10, iteration =5
LSTM
2 LSTM layers
1 hidden dense layers
1 output dense layer
16 batch size
100 epochs
0.001 Adam optimiser
Shuffle = false
Evaluation
Compare Performance Evaluation
Deployment
Provide Business Insights and Analysis

Figure 3.1.1 System Model Diagram

In this study, the system methodology is based on CRISP-DM. In fact, the first phase is known as the business understanding phase which is used to understand the business and formulate critical questions. This phase ensures that forecasting efforts align with organizational objectives and environment. Therefore, the historical timeline of Capital A Berhad is investigated by going through its official page, reading its financial statement and recent news. All the information collected is recorded and simplified in the literature review. This is crucial to further our understanding of the business operation of the company.

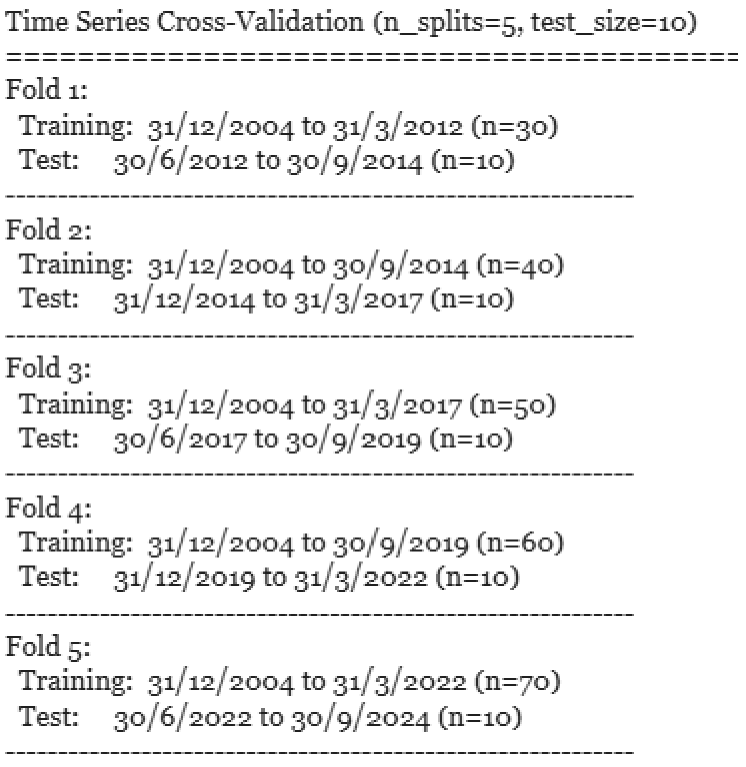
Next, the second and third phase known as data understanding and data preparation which can break down to 4 distinct subsections. The first being the data extraction phase, the pdf file of the finance data such as quarterly income statement, quarterly balance sheet, quarterly cashflow from Capital A Berhad are downloaded from KLSE Screener before using OCR scanners to convert them into csv files. In fact, the market index of the company and the KLSE stock market data is also downloaded from the Bursa Malaysia as csv file. At the same time, relevant text related to Capital A Berhad is collected to conduct sentiment analysis by using BeautifulSoup to convert them into csv files. For example, customer review from Skytrax. This is to ensure that the data collected is relevant to the study.

During the subsection known as data preprocessing phase, the dataset collected will be cleaned to handle the missing values, format the date and conduct data analysis. Data will also be scaled and normalized to ensure that it is suitable for time series analysis. For instance, financial data from 2004 to 2024 are collected quarterly which made up of 80 observation points. However, the finance data exists 14 columns with missing data. Therefore, KNN imputation and FCMI imputation is conducted to find out which method is the best before using data analysis to compare the results of pre- and post-imputation using metrics like R-squared, median, and mean.

Next, feature selection is conducted to reduce the number of features by using Decision tree with max depth = 5 to identify the most important features from 306 variables. The result selects top 5 and top 12 features from the FCMI-imputed dataset for modelling.

On the fourth phase which is known as modelling. In this case, two parallel approaches known as sentiment analysis and time series analysis are employed to forecast financial trends. For sentiment analysis, text data is processed using ROBERTA a type of BERT, FinBERT, and TextBlob, with the results fed into an LSTM model to integrate sentiment insights. Meanwhile, pure time series analysis trains a separate LSTM model exclusively on structured financial data. In fact, both LSTM models share identical architectures (2 LSTM layers → 1 dense hidden layer → 1 output layer) and training configurations (Adam optimizer, 100 epochs, batch size 16), with data split via forward-window cross-validation (test size 10, 5 iterations) to maintain temporal integrity. Finally, the sentiment-only and finance-only LSTM outputs are compared, before employing both sentiment and finance data on LSTM to evaluate the impact of text sentiment on forecasting accuracy. Thus, ensuring robust insights for decision-making.

Figure 3.1.2 Data Splitting Cross Validation Example



```
Time Series Cross-Validation (n_splits=5, test_size=10)
==================================================
Fold 1:
  Training:  31/12/2004 to 31/3/2012 (n=30)
  Test:      30/6/2012 to 30/9/2014 (n=10)
--------------------------------------------------
Fold 2:
  Training:  31/12/2004 to 30/9/2014 (n=40)
  Test:      31/12/2014 to 31/3/2017 (n=10)
--------------------------------------------------
Fold 3:
  Training:  31/12/2004 to 31/3/2017 (n=50)
  Test:      30/6/2017 to 30/9/2019 (n=10)
--------------------------------------------------
Fold 4:
  Training:  31/12/2004 to 30/9/2019 (n=60)
  Test:      31/12/2019 to 31/3/2022 (n=10)
--------------------------------------------------
Fold 5:
  Training:  31/12/2004 to 31/3/2022 (n=70)
  Test:      30/6/2022 to 30/9/2024 (n=10)
--------------------------------------------------
```

The fifth phase is known as Evaluation. In this case, for time series analysis, MSE scores and MAE scores are used to evaluate the LSTM model. For sentiment analysis, confusion matrix and classification report are used to compare between the BERT model, FinBERT model and the Text Blob Model. To compare the best models between the model trained with only text data, the model trained with only finance data and the model trained with both text and finance data, the average MSE and MAE scores are compared for each fold as seen in Figure 3.1.1.

The last phase is known as Deployment. In this case, the delivery is the result taken from the time series analysis and the sentiment analysis using the time series forecasting model. The combination of both results is used to compare against the PN17 classification to determine if sentiment analysis would truly help leverage a more competent financial distress signal using business insight learned in the process.

3.2 Equation

The target for forecasting is the Altman Z-score. Here is the original Altman's Z-score formula and threshold written as follows [20]:

$\boldsymbol{\zeta = 1.2A + 1.4B + 3.3C + 0.6D + 1.0E}$

Where:

- **Zeta** (ζ) is Altman's Z-score
- **A** is the Working Capital/Total Assets ratio
- **B** is the Retained Earnings/Total Assets ratio
- **C** is the Earnings Before Interest and Tax/Total Assets ratio
- **D** is the Market Value of Equity/Total Liabilities ratio
- **E** is the Total Sales/Total Assets ratio

Table 3.2.1 Original Z-score Threshold

Z-score range	Indications
Z-score < 1.8	Distress Zone
1.8 < Z-score < 3.0	Grey Zone
Z-score > 3.0	Safe Zone

However, after certain experimentations that will be further discussed in the paragraphs that followed, it is found out that a modified Altman's Z-score formula and threshold taken from a Malaysian-based financial article works better for financial distress prediction for Capital A Berhad because it's modified Altman Z-scoret is more align with the PN17 classification as it is localized. The modified formula and threshold are written as follows [20]:

$\zeta = 6.56A + 3.26B + 6.72C + 1.05D + 3.25$

Where:

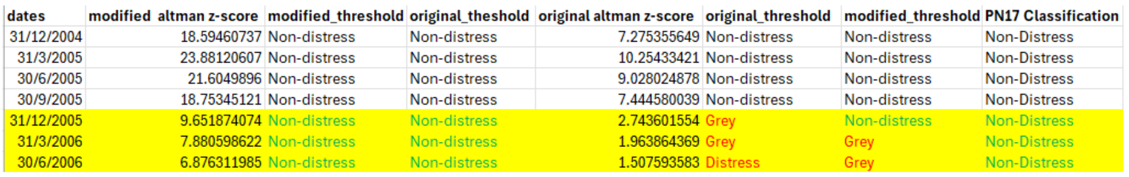
- **Zeta** (ζ) is Altman's Z-score
- **A** is the Working Capital/Total Assets ratio
- **B** is the Retained Earnings/Total Assets ratio
- **C** is the Earnings Before Interest and Tax/Total Assets ratio
- **D** is the Market Value of Equity/Total Liabilities ratio

Table 3.2.2 Modified Z-score Threshold

Z-score range	**Indications**
Z-score < 1.1	Distress Zone
1.1 < Z-score < 2.6	Grey Zone
Z-score > 2.6	Safe Zone

This discrepancy arises because the original Altman Z-score and its thresholds are more conservative. This means it consistently classified the company in the grey or distress zones despite the PN17 classification indicating a more positive outlook. As shown in figure 3.2.1, the original Z-score placed the company in the grey or distress zones from the December 2005 quarter to the June 2006 quarter, contradicting the PN17 classification during that period.

However, pairing the original Altman Z-score calculation with the modified thresholds improves accuracy. Notably, the modified Altman Z-score combined with the modified thresholds aligns closely with the PN17 classification. For context purpose, as illustrated in the figure, green text indicates correct classifications, while red highlights misclassifications:



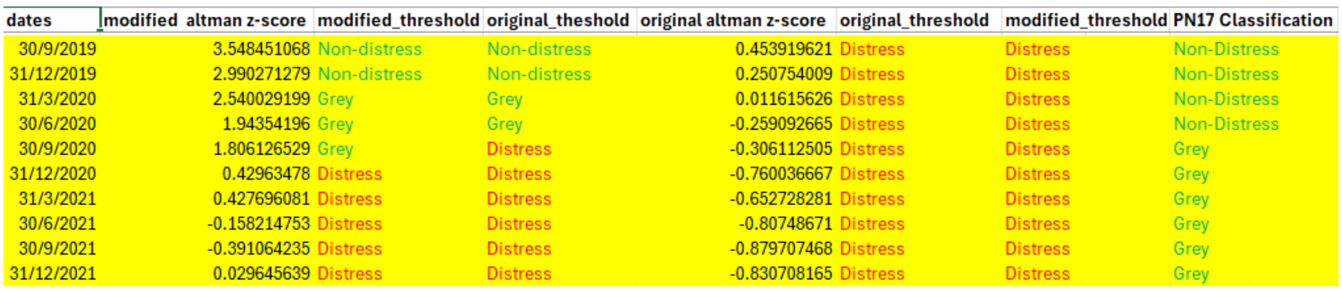
dates	modified altman z-score	modified_threshold	original_theshold	original altman z-score	original_threshold	modified_threshold	PN17 Classification
31/12/2004	18.59460737	Non-distress	Non-distress	7.275355649	Non-distress	Non-distress	Non-Distress
31/3/2005	23.88120607	Non-distress	Non-distress	10.25433421	Non-distress	Non-distress	Non-Distress
30/6/2005	21.6049896	Non-distress	Non-distress	9.028024878	Non-distress	Non-distress	Non-Distress
30/9/2005	18.75345121	Non-distress	Non-distress	7.444580039	Non-distress	Non-distress	Non-Distress
31/12/2005	9.651874074	Non-distress	Non-distress	2.743601554	Grey	Non-distress	Non-Distress
31/3/2006	7.880598622	Non-distress	Non-distress	1.963864369	Grey	Grey	Non-Distress
30/6/2006	6.876311985	Non-distress	Non-distress	1.507593583	Distress	Grey	Non-Distress

Figure 3.2.1 Evidence to justify the modification of Altman Z-score and Threshold

Additionally, the disparity between the original and modified Altman Z-score models becomes particularly evident when examining the period from September 2019 to December 2021. This is because while the original Z-score and thresholds prematurely flagged the company as being in financial distress, the modified version demonstrated greater alignment with the actual PN17 classification timeline.

For context, in 2019, the company reported net losses due to rising maintenance and overhaul costs, the adoption of MFRS16 (Malaysian Financial Reporting Standard 16 which increases reported liabilities and share losses from AirAsia India [34] [35] [36]. However, these challenges alone did not necessarily indicate imminent financial distress. Similarly, in early 2020, the company faced severe revenue declines that plummeted to RM3.13 billion from RM11.86 billion in comparison to the previous year [37]. This is due to COVID-19 travel restrictions. In fact, this was also accompanied by a transition to positive net debt of RM756 million as borrowing increased [37]. And yet, according to PN17 Classifications, the company only entered the grey zone in September 2020. Thus, suggesting that its financial deterioration was more gradual than the original Z-score implied.

Therefore, this discrepancy highlights the original Altman Z-score have a lack of nuance in distinguishing between temporary setbacks and genuine financial distress. In contrast, the modified Z-score and adjusted thresholds proved to be more accurate since it manages to mirror the PN17 classification's gradual progression from the grey zone rather than immediately signaling distress. Once again, this underscores the modified formula and threshold's superior reliability in reflecting true financial health.



dates	modified altman z-score	modified_threshold	original_theshold	original altman z-score	original_threshold	modified_threshold	PN17 Classification
30/9/2019	3.548451068	Non-distress	Non-distress	0.453919621	Distress	Distress	Non-Distress
31/12/2019	2.990271279	Non-distress	Non-distress	0.250754009	Distress	Distress	Non-Distress
31/3/2020	2.540029199	Grey	Grey	0.011615626	Distress	Distress	Non-Distress
30/6/2020	1.94354196	Grey	Grey	-0.259092665	Distress	Distress	Non-Distress
30/9/2020	1.806126529	Grey	Distress	-0.306112505	Distress	Distress	Grey
31/12/2020	0.42963478	Distress	Distress	-0.760036667	Distress	Distress	Grey
31/3/2021	0.427696081	Distress	Distress	-0.652728281	Distress	Distress	Grey
30/6/2021	-0.158214753	Distress	Distress	-0.80748671	Distress	Distress	Grey
30/9/2021	-0.391064235	Distress	Distress	-0.879707468	Distress	Distress	Grey
31/12/2021	0.029645639	Distress	Distress	-0.830708165	Distress	Distress	Grey

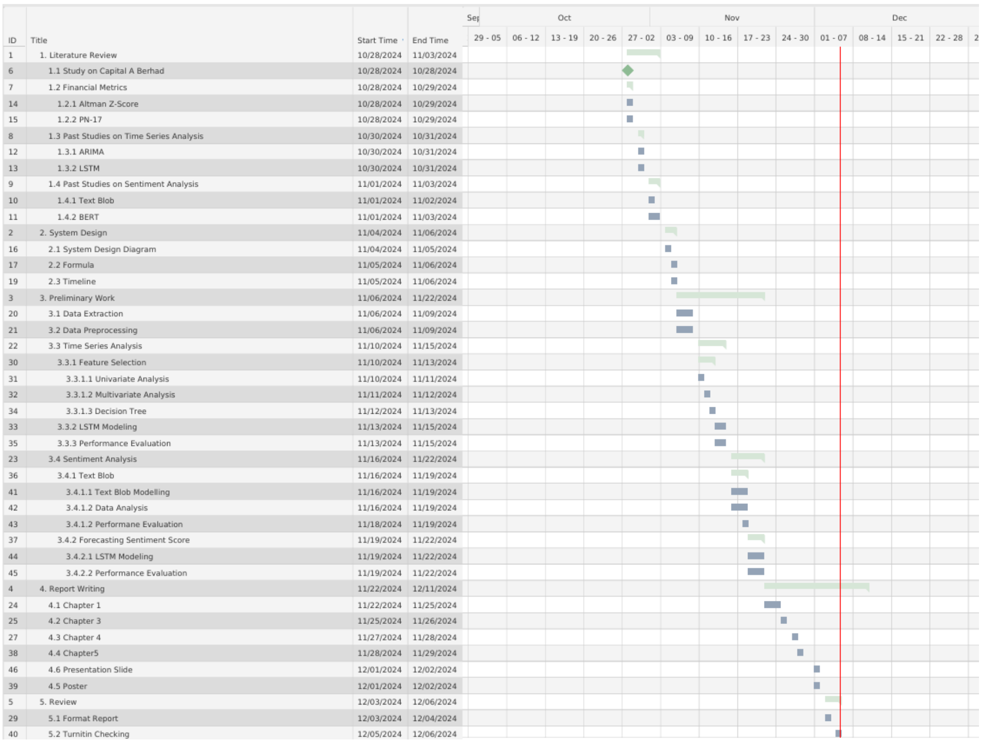
Figure 3.2.2 Second Evidence to justify the modification of Altman Z-score and Threshold

Therefore, as seen in the table below, it is more justified to use the modified Altman Z-score and modified threshold as target for forecasting due to its higher accuracy in classifying a PN17 classification.

Table 3.2.3 Accuracy Comparison Table for PN17 Classification

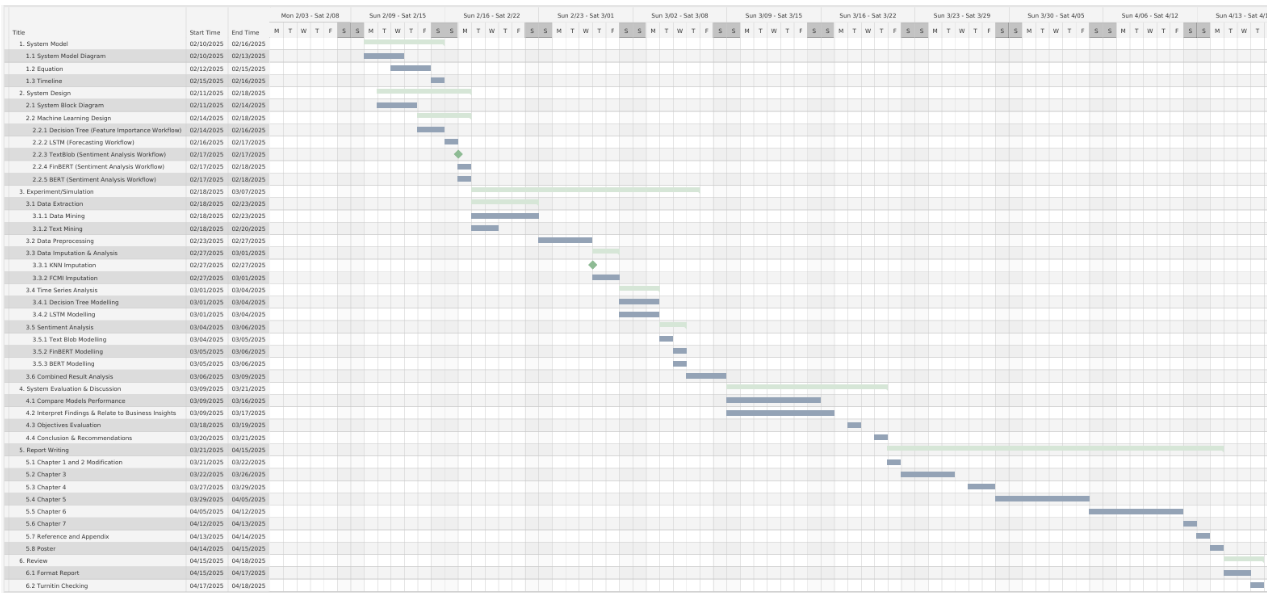
Accuracy against PN17 classification (%)			
Using Modified Altman Z-score Calculation Method and Modified Threshold	**Using Modified Altman Z-score Calculation Method and Original Threshold**	**Using Original Altman Z-score Calculation Method and Original Threshold**	**Using Original Altman Z-score Calculation Method and Modified Threshold**
93.75	92.50	83.75	85.00

3.3 Timeline



ID	Title	Start Time	End Time
1	1. Literature Review	10/28/2024	11/03/2024
6	1.1 Study on Capital A Berhad	10/28/2024	10/28/2024
7	1.2 Financial Metrics	10/28/2024	10/29/2024
14	1.2.1 Altman Z-Score	10/28/2024	10/29/2024
15	1.2.2 PN-17	10/28/2024	10/29/2024
8	1.3 Past Studies on Time Series Analysis	10/30/2024	10/31/2024
12	1.3.1 ARIMA	10/30/2024	10/31/2024
13	1.3.2 LSTM	10/30/2024	10/31/2024
9	1.4 Past Studies on Sentiment Analysis	11/01/2024	11/03/2024
10	1.4.1 Text Blob	11/01/2024	11/02/2024
11	1.4.2 BERT	11/01/2024	11/03/2024
2	2. System Design	11/04/2024	11/06/2024
16	2.1 System Design Diagram	11/04/2024	11/05/2024
17	2.2 Formula	11/05/2024	11/06/2024
19	2.3 Timeline	11/05/2024	11/06/2024
3	3. Preliminary Work	11/06/2024	11/22/2024
20	3.1 Data Extraction	11/06/2024	11/09/2024
21	3.2 Data Preprocessing	11/06/2024	11/09/2024
22	3.3 Time Series Analysis	11/10/2024	11/15/2024
30	3.3.1 Feature Selection	11/10/2024	11/13/2024
31	3.3.1.1 Univariate Analysis	11/10/2024	11/11/2024
32	3.3.1.2 Multivariate Analysis	11/11/2024	11/12/2024
34	3.3.1.3 Decision Tree	11/12/2024	11/13/2024
33	3.3.2 LSTM Modeling	11/13/2024	11/15/2024
35	3.3.3 Performance Evaluation	11/13/2024	11/15/2024
23	3.4 Sentiment Analysis	11/16/2024	11/22/2024
36	3.4.1 Text Blob	11/16/2024	11/19/2024
41	3.4.1.1 Text Blob Modelling	11/16/2024	11/19/2024
42	3.4.1.2 Data Analysis	11/16/2024	11/19/2024
43	3.4.1.2 Performane Evaluation	11/18/2024	11/19/2024
37	3.4.2 Forecasting Sentiment Score	11/19/2024	11/22/2024
44	3.4.2.1 LSTM Modeling	11/19/2024	11/22/2024
45	3.4.2.2 Performance Evaluation	11/19/2024	11/22/2024
4	4. Report Writing	11/22/2024	12/11/2024
24	4.1 Chapter 1	11/22/2024	11/25/2024
25	4.2 Chapter 3	11/25/2024	11/26/2024
27	4.3 Chapter 4	11/27/2024	11/28/2024
38	4.4 Chapter5	11/28/2024	11/29/2024
46	4.6 Presentation Slide	12/01/2024	12/02/2024
39	4.5 Poster	12/01/2024	12/02/2024
5	5. Review	12/03/2024	12/06/2024
29	5.1 Format Report	12/03/2024	12/04/2024
40	5.2 Turnitin Checking	12/05/2024	12/06/2024

Figure 3.3.1 Timeline for FYP1



Title	Start Time	End Time
1. System Model	02/10/2025	02/16/2025
1.1 System Model Diagram	02/10/2025	02/13/2025
1.2 Equation	02/12/2025	02/15/2025
1.3 Timeline	02/15/2025	02/16/2025
2. System Design	02/11/2025	02/18/2025
2.1 System Block Diagram	02/11/2025	02/14/2025
2.2 Machine Learning Design	02/14/2025	02/18/2025
2.2.1 Decision Tree (Feature Importance Workflow)	02/14/2025	02/16/2025
2.2.2 LSTM (Forecasting Workflow)	02/16/2025	02/17/2025
2.2.3 TextBlob (Sentiment Analysis Workflow)	02/17/2025	02/17/2025
2.2.4 FinBERT (Sentiment Analysis Workflow)	02/17/2025	02/18/2025
2.2.5 BERT (Sentiment Analysis Workflow)	02/17/2025	02/18/2025
3. Experiment/Simulation	02/18/2025	03/07/2025
3.1 Data Extraction	02/18/2025	02/23/2025
3.1.1 Data Mining	02/18/2025	02/23/2025
3.1.2 Text Mining	02/18/2025	02/20/2025
3.2 Data Preprocessing	02/23/2025	02/27/2025
3.3 Data Imputation & Analysis	02/27/2025	03/01/2025
3.3.1 KNN Imputation	02/27/2025	02/27/2025
3.3.2 FCMI Imputation	02/27/2025	03/01/2025
3.4 Time Series Analysis	03/01/2025	03/04/2025
3.4.1 Decision Tree Modelling	03/01/2025	03/04/2025
3.4.2 LSTM Modelling	03/01/2025	03/04/2025
3.5 Sentiment Analysis	03/04/2025	03/06/2025
3.5.1 Text Blob Modelling	03/04/2025	03/05/2025
3.5.2 FinBERT Modelling	03/05/2025	03/06/2025
3.5.3 BERT Modelling	03/05/2025	03/06/2025
3.6 Combined Result Analysis	03/06/2025	03/09/2025
4. System Evaluation & Discussion	03/09/2025	03/21/2025
4.1 Compare Models Performance	03/09/2025	03/16/2025
4.2 Interpret Findings & Relate to Business Insights	03/09/2025	03/17/2025
4.3 Objectives Evaluation	03/18/2025	03/19/2025
4.4 Conclusion & Recommendations	03/20/2025	03/21/2025
5. Report Writing	03/21/2025	04/15/2025
5.1 Chapter 1 and 2 Modification	03/21/2025	03/22/2025
5.2 Chapter 3	03/22/2025	03/26/2025
5.3 Chapter 4	03/27/2025	03/29/2025
5.4 Chapter 5	03/29/2025	04/05/2025
5.5 Chapter 6	04/05/2025	04/12/2025
5.6 Chapter 7	04/12/2025	04/13/2025
5.7 Reference and Appendix	04/13/2025	04/14/2025
5.8 Poster	04/14/2025	04/15/2025
6. Review	04/15/2025	04/18/2025
6.1 Format Report	04/15/2025	04/17/2025
6.2 Turnitin Checking	04/17/2025	04/18/2025

Figure 3.3.2 Timeline for FYP2

CHAPTER 4

System Design

4.1 System Block Diagram

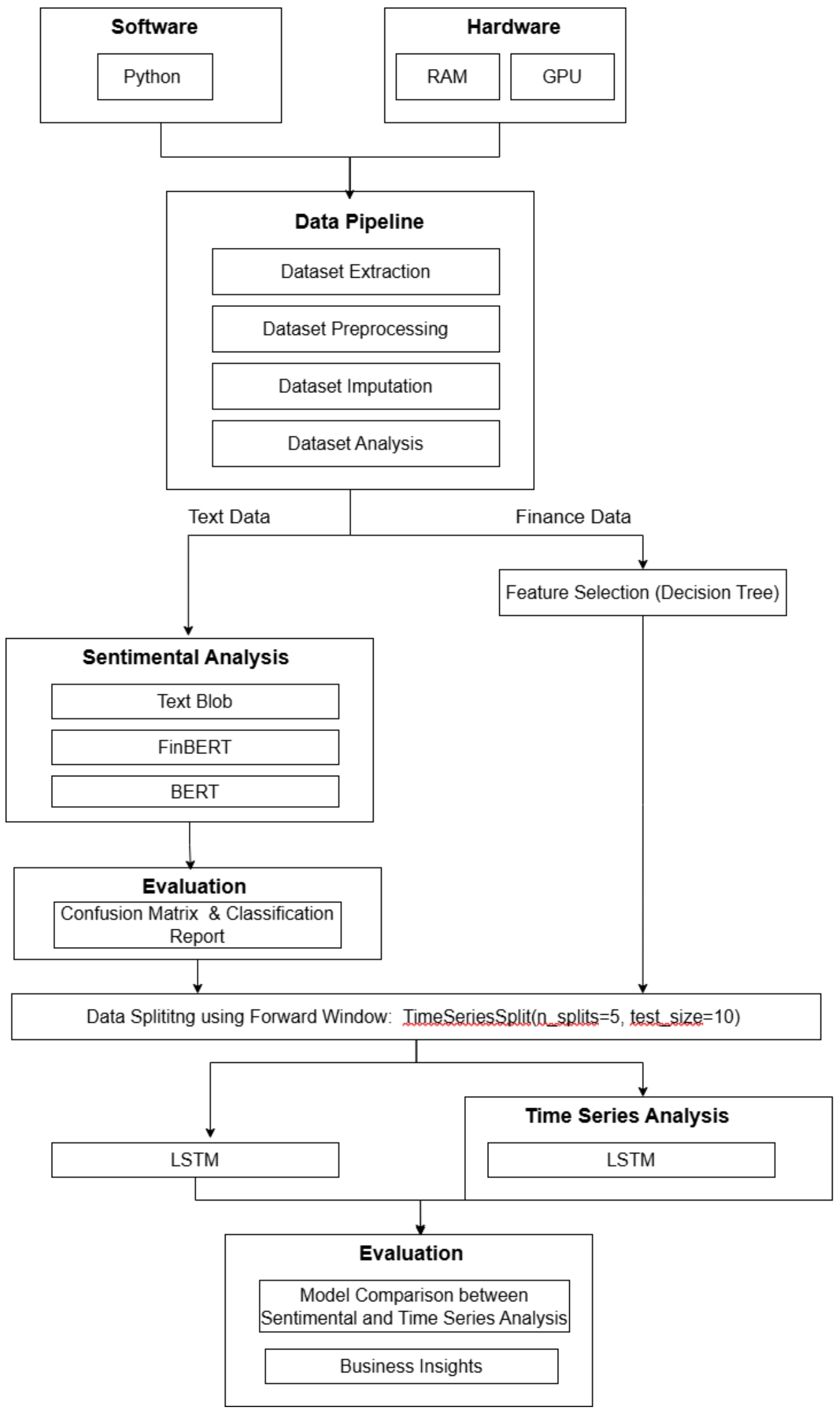


Figure 4.1.1 System Block Diagram

Based on the diagram above, this system is a machine learning pipeline designed for sentiment analysis and time series forecasting. This begins with data extraction followed by preprocessing, imputation, and analysis to clean and explore the data. For sentiment analysis, it employs TextBlob, FinBERT (financial-tuned BERT), and BERT to gauge sentiment from text. Then, decision trees are used to help select key features. The system evaluates sentimental data performance using confusion matrices and classification reports. Simultaneously, an LSTM model analyzes temporal patterns in time series data to predict Altman Z-score. The data splitting is conducted using forward window to spil5 the data into 5 folds with each fold has a test set of 10 consecutive observations. In fact, the training data expands with each fold while the test data move forward in time. The result from the combination of the best from sentimental and time series analysis model is then used to derive business insights, such as correlating sentiment with market trends for financial distress forecasting to support decision-making. Lastly, the entire workflow leverages Python, GPU, and RAM for efficient computation.

4.2 Hardware and Software Specifications

4.2.1 Hardware

The hardware involved in this project is a standard laptop device. The laptop will be used to run the dataset and execute the time series algorithms for financial distress forecasting. Here are the main specifications:

Table 4.2.1.1 Specifications of laptop

Description	Specifications
Model	IdeaPad 3 15ITL6
Processor	11th Gen Intel(R) Core (TM) i5-1135G7 @ 2.40GHz 2.42 GHz
Operating System	Windows 10 64-bit
Graphic	Intel® Iris ® Xe Graphics
RAM	Samsung 12.0 Micron Tech GB 3200MHz
Storage	NVMe SAMSUNG MZALQ256HBJD-00BL2 256GB

4.2.2 Software

Here are the following software applications required for the system:

Google Colab

Google Colab is a free, cloud-based platform for creating and sharing interactive notebooks. It comes pre-installed with widely used data science and machine learning libraries (e.g., TensorFlow, PyTorch, pandas). This software primarily supports Python and enables the leverage of GPUs and TPUs for accelerated computation. While it supports other languages like R, Python will be the primary language used in this case.

Microsoft Excel

Microsoft Excel is powerful spreadsheet software developed by Microsoft that is widely used for data analysis, reporting, and automation of calculations. Excel provides a grid-based interface where users can store, organize, and analyze data. This software will be used to conduct univariate and multivariate analysis.

4.3 Machine Learning Design

4.3.1 Decision Tree (Feature Importance Workflow)

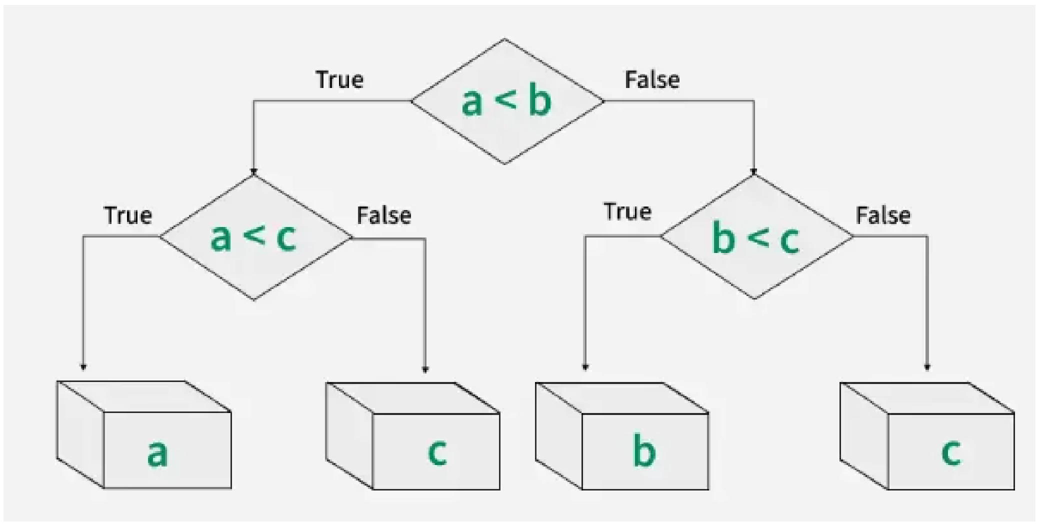


Figure 4.3.1.1 Decision Tree Regressor Diagram

The decision tree algorithm recursively partitions the dataset into subsets by evaluating feature values, aiming to minimize impurity by using decision rules learned from input features. Therefore, the process identifies the most discriminative features by reducing the error in calculating the target variable during each split [38].

In this case, a decision tree regressor is used to identify important financial features for the modified Altman Z-score. Firstly, the tree recursively splits the data based on features that minimize prediction error using MSE criterion. After training, the model calculates feature importance scores using Gini importance to rank variables by their predictive power,

4.3.2 LSTM algorithm (Forecasting Workflow)

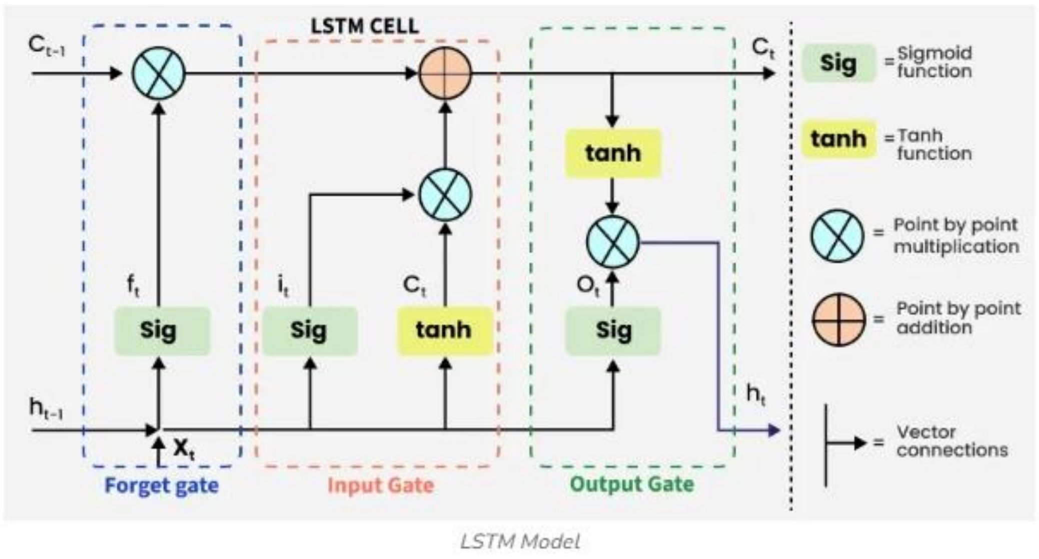


Figure 4.3.2.1 LSTM Diagram

Long Short-Term Memory (LSTM) is a specialized Recurrent Neural Network (RNN) architecture designed to handle sequential data. Thus, making it well-suited for time series forecasting. In this case, predicting future Altman Z-scores. Unlike traditional RNNs, LSTMs use memory cells and three key gates to capture long-term dependencies effectively. In fact, LSTM requires data input in sequences [39]. For instance, the model processes data in sequential windows of past observations. Thus, preserving temporal relationships through careful data splitting.

As seen in Figure 4.3.2, the first component is the forget gate. This is used to determine which information to discard using sigmoid activation. Next, the input gate is used to update the cell state using both sigmoid and tanh. Finally, the output gate is used to filter the updated cell state to produce the hidden state. In short, the cell state maintains long term dependencies while the hidden state handles short term context. These two components work together to capture complex temporal patterns in the time series model [39].

In this project, the LSTM model is built with TensorFlow/Keras. It consists of two LSTM layers (64 and 32 units) followed by dense layers (16 and 1 unit), using ReLU activation and Glorot uniform initialization for weights. Lastly, the model is trained using time-series cross-validation (5 splits) with early stopping to prevent overfitting.

4.3.3 TextBlob (Sentiment Analysis Workflow)

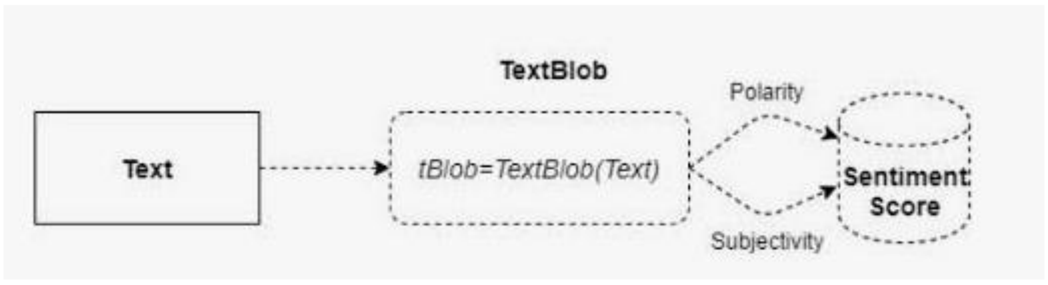


Figure 4.3.3.1 Text Blob Diagram

Text Blob is just a lexicon-driven tool for quick sentiment analysis. In this case, it used a predefined sentiment dictionary for sentimental analysis. Its polarity ranges from -1 which is negative sentiment to +1 which is positive sentiment. Then, for each word in a sentence, a weighted average sentiment score will be applied [40]. However, unlike FinBERT and BERT, Text Blob's rule-based approach lacks contextual nuance which makes it inferior to transformer models.

4.3.4 FinBERT (Sentiment Analysis Workflow)

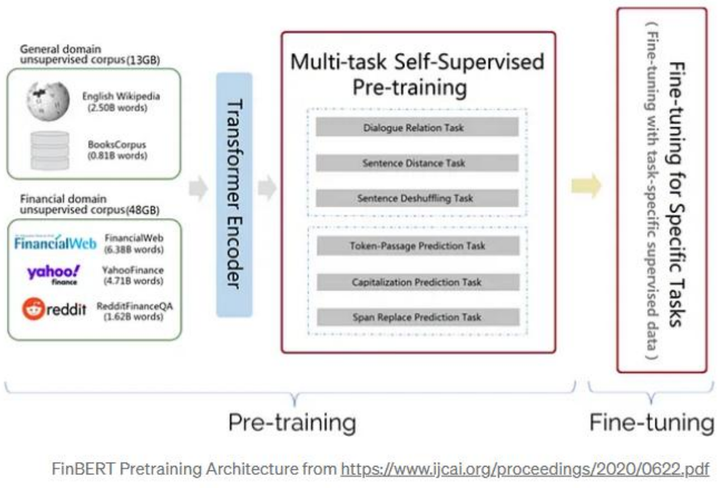

General domain unsupervised corpus(13GB)
English Wikipedia (2.50B words)
BooksCorpus (0.81B words)
Financial domain unsupervised corpus(48GB)
FinancialWeb (6.38B words)
YahooFinance (4.71B words)
RedditFinanceQA (1.62B words)
Transformer Encoder
Multi-task Self-Supervised Pre-training
Dialogue Relation Task
Sentence Distance Task
Sentence Deshuffling Task
Token-Passage Prediction Task
Capitalization Prediction Task
Span Replace Prediction Task
Fine-tuning for Specific Tasks
(Fine-tuning with task-specific supervised data)
Pre-training
Fine-tuning
FinBERT Pretraining Architecture from https://www.ijcai.org/proceedings/2020/0622.pdf

Figure 4.3.4.1 FinBERT Diagram

FinBERT is a financial-domain-specific BERT transformer model fine-tuned for finance domain. This means it retains BERT's multi-layer bidirectional transformer encoder, which is used to generate contextual embeddings for input tokens [41]. However, unlike BERT's original pre-training, FinBERT is pre-trained with six different self-supervised tasks. Therefore, this allows it to capture richer semantic and language knowledge specific to the financial domain [42]. Lastly, it has a SoftMax layer to predict sentiment as such as negative, neutral, positive [43]. In this study, a pretrained model from "yiyanghkust/finbert-tone" will be loaded to perform the sentimental analysis comparison.

4.3.5 BERT (Sentiment Analysis Workflow)

BERT model is a transformer-based language model. It is trained using encoders, which is used to learn the language and generate embeddings [41].

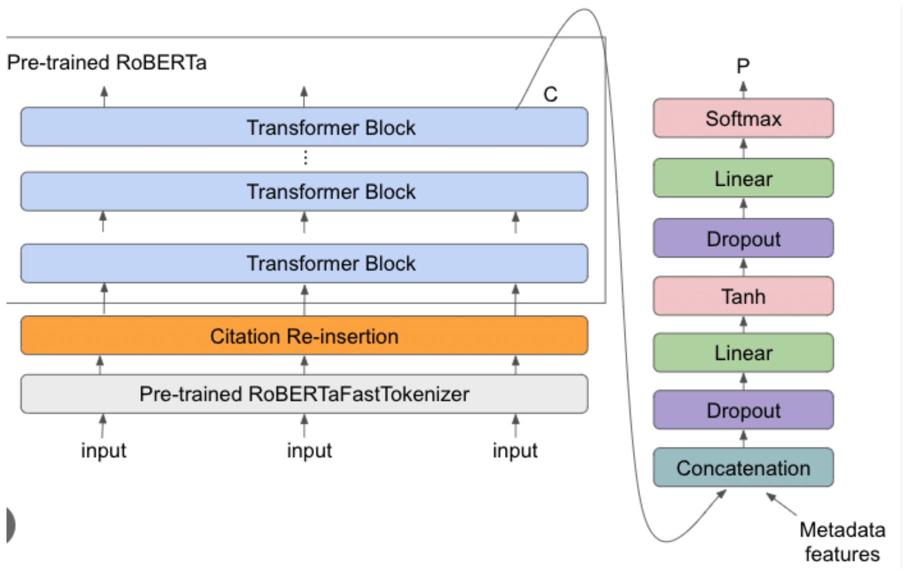


Figure 4.3.5.1 RoBERTa Diagram

In this case RoBERTa which is a type of BERT model, is applied in the project. It uses the uses the standard transformer encoder architecture. This structure consists of multiple layers of self-attention and feed-forward neural networks. The self-attention feature lets the model weigh the importance of each word in the context of the entire input sequence. Thus, capturing nuanced relationships and dependencies. Like BERT, RoBERTa is pre-trained using dynamically changed masking patterns. During training, a random subset of input tokens is masked, and the model learns to predict the original values of these masked tokens using the surrounding context [44]. It is mainly trained on labeled sentiment datasets such as Twitter. This makes it better at contextual understanding. So, it can handle negation and complex phrasing. In this study, a pretrained model from "cardiffnlp/twitter-roberta-base-sentiment" will be used for sentimental analysis comparison.

CHAPTER 5

System Implementation

5.1 Hardware Setup

Table 5.1.1 Computing Resources in Google Colab Cloud Environment

Hardware Accelerator	CPU
RAM	12.67 GB (11.39 GB free)
Disk Space	107.72 GB (36.91 GB used)
Runtime Limitations	Colab disconnects after 30 minutes of inactivity (or ~12 hours for active usage).
Runtime Type	Python 3/ R

Table 5.1.2 Local Hardware

CPU Spes	Intel (1.64 GHz base)
GPU Specs	Intel(R) Iris(R) Xe Graphics
RAM Capacity	11.8 GB (7.9 GB USED)
Storage	SSD (RAID)

5.2 Software Setup

Before starting to develop the project, these are the software needed to be installed and downloaded in my laptop:

1. Google Colab (Python 3.10)
2. Microsoft Excel

Table 5.2.1 Table of Key Libraries

Key libraries related to	Libraries
Deep Learning Models	TensorFlow/Keras
Sentiment Analysis	NLTK, TextBlob, Hugging Face's transformers (e.g., FinBERT), BeautifulSoup
Data Processing	Pandas, NumPy, scikit-learn
Visualization	Matplotlib, Seaborn, Plotly

5.3 Setting and Configuration

1. LSTM Model Configuration:

Table 5.3.1 LSTM Model Configuration

Number of LSTM layers	2
Hidden units per layer	Layer 1: 64, Layer 2: 32
Dense layers	1 hidden Dense (16 units), 1 output Dense (1 unit)
Batch size	16
Epochs	100
Optimiser	Adam (learning_rate=0.001)
Loss function	Mean Squared Error (MSE)
Weight initialization	Seed (42)
Shuffle	False

2. Dataset Configuration:

Data Sources

- **Finance data:** quarterly balance sheet, quarterly cash flow statement, quarterly income statement, historical data for Capital A Berhad and the FTSE Malaysia KLCI index (2004-2024, 80 data points)
- **Text data:** Skytrax review (2011-2024, 852 data points)

3. Google Colab Settings:

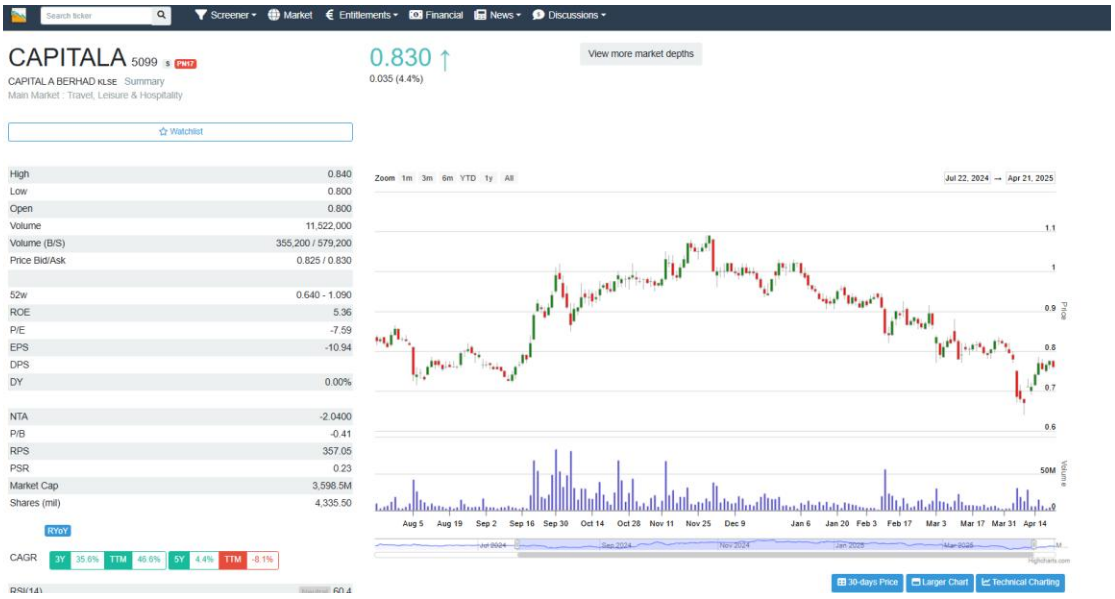
- Mount Google Drive for storing datasets

5.4 System Operation

5.4.1 Data Extraction

5.4.1.1 Data Mining

To support financial distress prediction, quarterly financial reports—including the balance sheet, cash flow statement, and income statement—were collected alongside historical data for Capital A Berhad and the FTSE Malaysia KLCI index from KLSE Screener website and Investing.com Website. These datasets are downloaded from their pdf format before being go through OCR scanning and formatted into csv file. This is to ensure compatibility, ease of processing, and flexibility for tasks like cleaning, normalization, and feature selection. This is because they are essential for evaluating a company's financial health and understanding its market environment. For instance, financial reports allow for the computation of critical metrics, such as the Altman Z-score, liquidity ratios, and profitability indicators, which signal financial stability. On the other hand, Capital A Berhad's historical data provides a detailed view of its performance, while FTSE Malaysia data offers insights into broader economic trends for contextual analysis. The data ranges from 31/12/2004 to 31/9/2024, a total of 80 data points. Hence, this integrated approach combines company-specific insights with market trends to improve the accuracy of financial distress forecasting models.



CAPITALA 5099 s PN17
CAPITAL A BERHAD KLSE Summary
Main Market : Travel, Leisure & Hospitality

0.830 ↑
0.035 (4.4%)

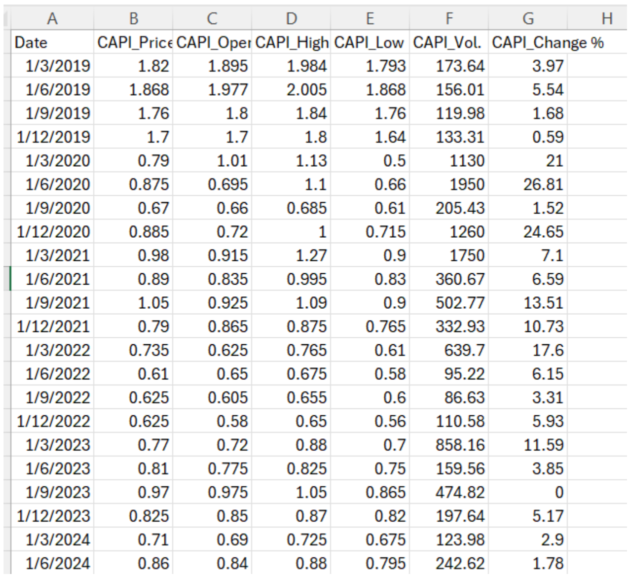
High	0.840
Low	0.800
Open	0.800
Volume	11,522,000
Volume (B/S)	355,200 / 579,200
Price Bid/Ask	0.825 / 0.830
52w	0.640 - 1.090
ROE	5.36
P/E	-7.59
EPS	-10.94
DPS	
DY	0.00%
NTA	-2.0400
P/B	-0.41
RPS	357.05
PSR	0.23
Market Cap	3,598.5M
Shares (mil)	4,335.50

RYoY

CAGR 3Y 35.6% TTM 46.6% 5Y 4.4% TTM -8.1%

RSI(14) 60.4

Figure 5.4.1.1.1 KLSE Screener Website

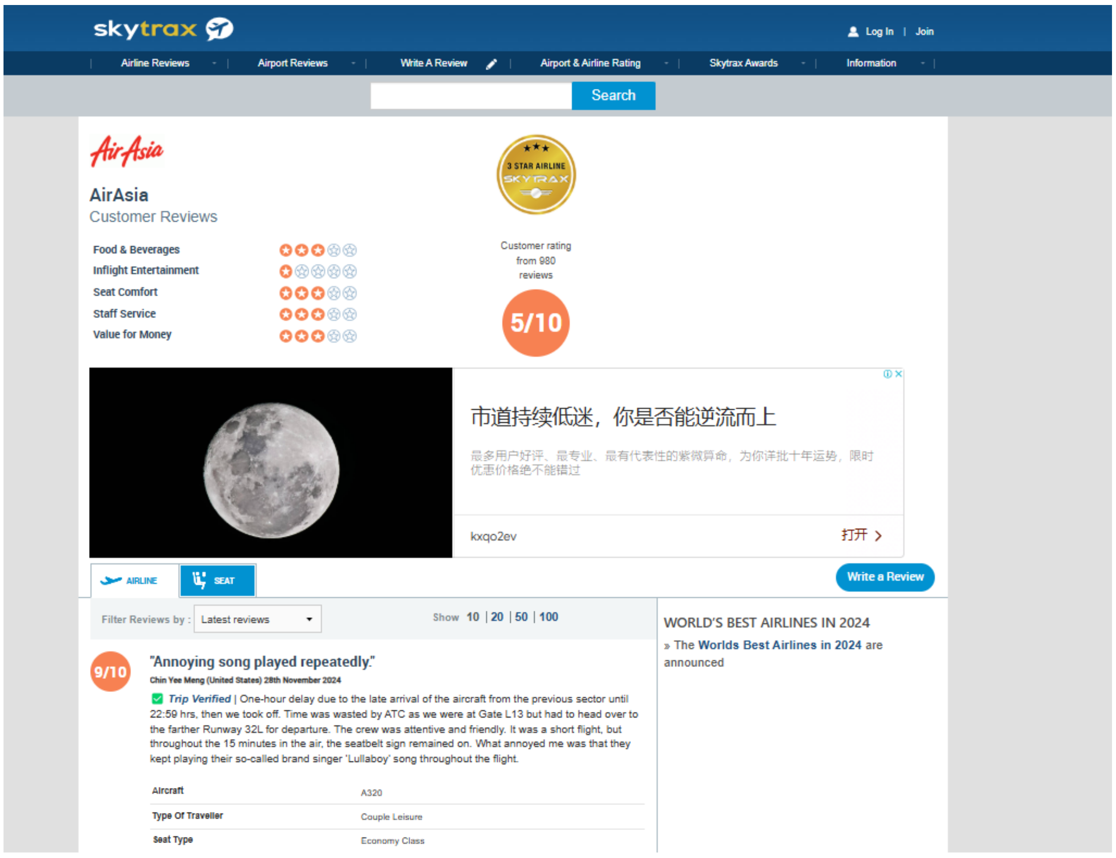


Date	CAPI_Price	CAPI_Open	CAPI_High	CAPI_Low	CAPI_Vol.	CAPI_Change %
1/3/2019	1.82	1.895	1.984	1.793	173.64	3.97
1/6/2019	1.868	1.977	2.005	1.868	156.01	5.54
1/9/2019	1.76	1.8	1.84	1.76	119.98	1.68
1/12/2019	1.7	1.7	1.8	1.64	133.31	0.59
1/3/2020	0.79	1.01	1.13	0.5	1130	21
1/6/2020	0.875	0.695	1.1	0.66	1950	26.81
1/9/2020	0.67	0.66	0.685	0.61	205.43	1.52
1/12/2020	0.885	0.72	1	0.715	1260	24.65
1/3/2021	0.98	0.915	1.27	0.9	1750	7.1
1/6/2021	0.89	0.835	0.995	0.83	360.67	6.59
1/9/2021	1.05	0.925	1.09	0.9	502.77	13.51
1/12/2021	0.79	0.865	0.875	0.765	332.93	10.73
1/3/2022	0.735	0.625	0.765	0.61	639.7	17.6
1/6/2022	0.61	0.65	0.675	0.58	95.22	6.15
1/9/2022	0.625	0.605	0.655	0.6	86.63	3.31
1/12/2022	0.625	0.58	0.65	0.56	110.58	5.93
1/3/2023	0.77	0.72	0.88	0.7	858.16	11.59
1/6/2023	0.81	0.775	0.825	0.75	159.56	3.85
1/9/2023	0.97	0.975	1.05	0.865	474.82	0
1/12/2023	0.825	0.85	0.87	0.82	197.64	5.17
1/3/2024	0.71	0.69	0.725	0.675	123.98	2.9
1/6/2024	0.86	0.84	0.88	0.795	242.62	1.78

Figure 5.4.1.1.2 Excel Sample of Capital A Berhad Historical Data

5.4.1.2 Text Mining

During this phase, customer reviews were exclusively extracted from the Skytrax website. Skytrax, as illustrated in the figures, is an international air transport rating organization based in the UK. A data mining script using BeautifulSoup and HTTP request was employed to directly extract daily reviews from the website, resulting in a total of 853 reviews collected from 2011 to 2024. These reviews were organized by year and stored in a CSV file, as depicted in the figure. The dataset includes columns such as date, header, rating, content, aircraft, traveler type, seat type, route, date flown, seat comfort, cabin staff service, inflight entertainment, ground service, Wi-Fi & connectivity, value for money, recommendation status, and food & beverages. However, for the purpose of data preprocessing, only the date, rating, and content columns were utilized for sentiment analysis.



AirAsia
Customer Reviews

Food & Beverages	★★★☆☆
Inflight Entertainment	★☆☆☆☆
Seat Comfort	★★★☆☆
Staff Service	★★★☆☆
Value for Money	★★★☆☆

Customer rating from 980 reviews

5/10

市道持续低迷，你是否能逆流而上

最多用户好评、最专业、最有代表性的紫微算命，为你详批十年运势，限时优惠价格绝不能错过

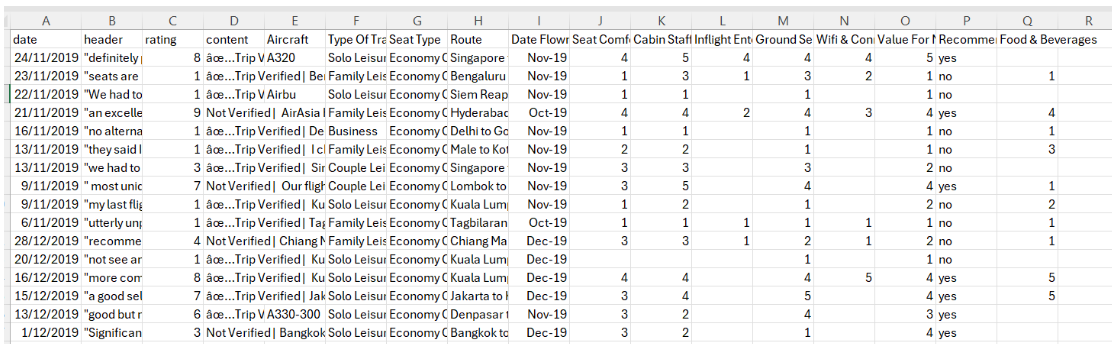
kxqo2ev 打开 >

WORLD'S BEST AIRLINES IN 2024
» The Worlds Best Airlines in 2024 are announced

9/10 "Annoying song played repeatedly."
Chin Yee Meng (United States) 28th November 2024
✓ *Trip Verified* | One-hour delay due to the late arrival of the aircraft from the previous sector until 22:59 hrs, then we took off. Time was wasted by ATC as we were at Gate L13 but had to head over to the farther Runway 32L for departure. The crew was attentive and friendly. It was a short flight, but throughout the 15 minutes in the air, the seatbelt sign remained on. What annoyed me was that they kept playing their so-called brand singer 'Lullaboy' song throughout the flight.

Aircraft	A320
Type Of Traveller	Couple Leisure
Seat Type	Economy Class

Figure 5.4.1.2.1 Skytrax Website



A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
date	header	rating	content	Aircraft	Type Of Tra	Seat Type	Route	Date Flowr	Seat Comf	Cabin Staff	Inflight Ent	Ground Se	Wifi & Con	Value For I	Recomme	Food & Beverages	
24/11/2019	"definitely	8	âœ…Trip V	A320	Solo Leisu	Economy C	Singapore	Nov-19	4	5	4	4	4	5	yes		
23/11/2019	"seats are	1	âœ…Trip Verified | Be		Family Leis	Economy C	Bengaluru	Nov-19	1	3	1	3	2	1	no	1	
22/11/2019	"We had to	1	âœ…Trip V	Airbu	Solo Leisu	Economy C	Siem Reap	Nov-19	1	1		1		1	no		
21/11/2019	"an excelle	9	Not Verified |	AirAsia I	Family Leis	Economy C	Hyderabac	Oct-19	4	4	2	4	3	4	yes	4	
16/11/2019	"no alterna	1	âœ…Trip Verified | De		Business	Economy C	Delhi to Gc	Nov-19	1	1		1		1	no	1	
13/11/2019	"they said I	1	âœ…Trip Verified |	I cl	Family Leis	Economy C	Male to Kot	Nov-19	2	2		1		1	no	3	
13/11/2019	"we had to	3	âœ…Trip Verified |	Sir	Couple Lei	Economy C	Singapore	Nov-19	3	3		3		2	no		
9/11/2019	" most unic	7	Not Verified |	Our fligh	Couple Lei	Economy C	Lombok to	Nov-19	3	5		4		4	yes	1	
9/11/2019	"my last flig	1	âœ…Trip Verified |	Ku	Solo Leisu	Economy C	Kuala Lum	Nov-19	1	2		1		2	no	2	
6/11/2019	"utterly unp	1	âœ…Trip Verified | Tag		Family Leis	Economy C	Tagbilaran	Oct-19	1	1	1	1	1	1	no	1	
28/12/2019	"recomme	4	Not Verified |	Chiang M	Family Leis	Economy C	Chiang Ma	Dec-19	3	3	1	2	1	2	no	1	
20/12/2019	"not see ar	1	âœ…Trip Verified |	Ku	Solo Leisu	Economy C	Kuala Lum	Dec-19				1		1	no		
16/12/2019	"more com	8	âœ…Trip Verified |	Ku	Solo Leisu	Economy C	Kuala Lum	Dec-19	4	4		4	5	4	yes	5	
15/12/2019	"a good sel	7	âœ…Trip Verified | Jak		Solo Leisu	Economy C	Jakarta to I	Dec-19	3	4		5		4	yes	5	
13/12/2019	"good but r	6	âœ…Trip V	A330-300	Solo Leisu	Economy C	Denpasar t	Nov-19	3	2		4		3	yes		
1/12/2019	"Significan	3	Not Verified |	Bangkok	Solo Leisu	Economy C	Bangkok to	Dec-19	3	2		1		4	yes		

Figure 5.4.1.2.2 Excel Sheet Sample of Text Mined

5.4.2 Data Preprocessing

The code begins by utilizing a custom class, DriveDataLoader, to streamline data ingestion. First, files such as quarterly income statements, balance sheets, cash flow statements, and market indices are loaded into Google Colab via google drive using specified file paths.

After loading the datasets, symbols such as commas, percentages, and dashes are removed or replaced with zeros to standardize numeric columns. This ensures compatibility with numerical computations. Next, columns, except for dates, are converted to numeric types, with invalid entries coerced into NaN. This step avoids errors during calculations. Then, the code filters records based on specific date ranges (2004–2024) and focuses on quarterly and monthly intervals relevant to financial reporting cycles. Besides, the date column is standardized into a uniform format, enabling consistent time-series analysis.

The script separates preprocessing logic for quarterly and monthly data due to differences in granularity and reporting:

- **Quarterly Data:** Includes income statements, balance sheets, and cash flows. Columns with all-zero values are dropped, ensuring only meaningful metrics remain.

- **Monthly Data:** Includes stock prices and indices. Special attention is given to scaling values, such as converting billions to millions. Thus, ensuring uniform units across datasets. Data is filtered to specific months to align with quarterly data (e.g. March, June, September, and December) representing key reporting periods.

Lastly, the processed data is prepared for deeper financial analysis through feature engineering. For instance, modified Altman Z-score is calculated to predict financial distress. Components like working capital, retained earnings, and EBIT are derived from the balance sheet and income statement. These are combined using a weighted formula to compute the Z-Score for each date. Aside of that, key Financial Ratios are also calculated. For instance, liquidity, profitability, solvency, and efficiency ratios are computed since they offer insights into a company's operational health.

Table 5.4.2.1 Altman Z-Score Formula Table

Variables	**Acquisition Method**	**Calculation method**
Working Capital	Calculated	total current assets- total current liabilities [21]
Total Assets	Calculated	total non-current assets + total current assets
Retained Earnings	Direct	
Market Value of Equity	Calculated	stock price x shares outstanding [22]
Total Liabilities	Calculated	total non-current liabilities + total current liabilities
Operating Revenue	Direct	

As seen in the table below, on average the modified Altman Z-Score from 2019 to 2024 are all under financial distress due to their lower 1.1 score causing a disparity with the PN17 classification while those during December of 2004 to June of 2019 are under non-distress which is aligned with the PN17 classification.

Table 5.4.2.2 Altman Z-Score Calculated against PN-17 Classification

Date	**Altman Z-Score with Modified Formula**	**Altman Z-Score with Modified Formula and Modified Threshold**	**PN-17 Classification [23][24]**
30/9/2019	3.548451068	Non-distress	Non-Distress
31/12/2019	2.990271279	Non-distress	Non-Distress

31/3/2020	2.540029199	Grey	Non-Distress
30/6/2020	1.94354196	Grey	Non-Distress
30/9/2020	1.806126529	Grey	Grey
31/12/2020	0.42963478	Distress	Grey
31/3/2021	0.427696081	Distress	Grey
30/6/2021	-0.158214753	Distress	Grey
30/9/2021	-0.391064235	Distress	Grey
31/12/2021	0.029645639	Distress	Grey
31/3/2022	-0.762250194	Distress	Distress

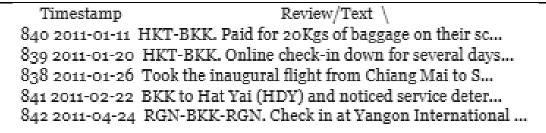
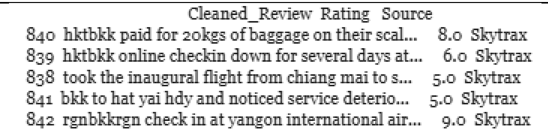
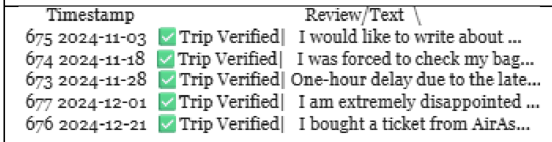
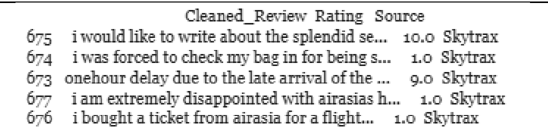
Based on the table below, these financial ratios can be used to apply on the financial data collected from the quarterly report to evaluate financial performance. These financial ratios will be used with other financial data to go through feature selection.

Table 5.4.2.3 Financial Ratios Formula Table

Financial Ratios	**Descriptions [25]**	**Indicators**	**Formula**	**Source**
Liquidity Ratios	The ability of a company to fulfil its short-term financial commitments and manage immediate funding needs.	Current Ratio	total current assets / total current liabilities	[26]
Profitability Ratios	The ability of a company to achieve profitability is relative to its income and available resources.	Operating Profit Margin	operating profit / revenue	[27]
		Net Profit Margin	profit after taxation / revenue	[28]
		Return on Assets (ROA)	net profit / total assets	[29]
		Return on Equity (ROE)	profit after taxation and minority interest / shareholder funds	[29]
		Earnings Per Share (EPS)	profit after taxation / number of shares	[30]
Efficiency Ratio	The capability of a company to meet its long-term debt obligations	Asset Turnover	revenue / total assets	[31]
Solvency Ratios	The capability of a company to meet its long-term debt obligations.	Debt-to-Assets Ratio	total liabilities / total assets	[32]

As for the text data, each text CSV file undergoes preprocessing through a specialized function that standardizes and cleans the data. This process involves removing predefined patterns such as URLs or irrelevant fragments, stripping special characters, and converting all text to lowercase for uniformity. Once cleaned, the processed text is organized into a new data frame and saved as a CSV file named *cleaned_reviews*. The resulting data frame includes columns for the **Timestamp** (date and time of the review), **Review/Text** (cleaned review content), **Source** (origin of the review), and **Rating** (numerical rating given by the reviewer). The figure below shows the differences between the original review and the cleaned review

Table 5.4.2.4 Comparison of Original Text and Cleaned Text in Excel File

Original Text	Cleaned Text
 Timestamp Review/Text \\ 840 2011-01-11 HKT-BKK. Paid for 20Kgs of baggage on their sc... 839 2011-01-20 HKT-BKK. Online check-in down for several days... 838 2011-01-26 Took the inaugural flight from Chiang Mai to S... 841 2011-02-22 BKK to Hat Yai (HDY) and noticed service deter... 842 2011-04-24 RGN-BKK-RGN. Check in at Yangon International ...	 Cleaned_Review Rating Source 840 hktbkk paid for 20kgs of baggage on their scal... 8.0 Skytrax 839 hktbkk online checkin down for several days at... 6.0 Skytrax 838 took the inaugural flight from chiang mai to s... 5.0 Skytrax 841 bkk to hat yai hdy and noticed service deterio... 5.0 Skytrax 842 rgnbkkrgn check in at yangon international air... 9.0 Skytrax
 Timestamp Review/Text \\ 675 2024-11-03 ✅Trip Verified| I would like to write about ... 674 2024-11-18 ✅Trip Verified| I was forced to check my bag... 673 2024-11-28 ✅Trip Verified| One-hour delay due to the late... 677 2024-12-01 ✅Trip Verified| I am extremely disappointed ... 676 2024-12-21 ✅Trip Verified| I bought a ticket from AirAs...	 Cleaned_Review Rating Source 675 i would like to write about the splendid se... 10.0 Skytrax 674 i was forced to check my bag in for being s... 1.0 Skytrax 673 onehour delay due to the late arrival of the ... 9.0 Skytrax 677 i am extremely disappointed with airasias h... 1.0 Skytrax 676 i bought a ticket from airasia for a flight... 1.0 Skytrax

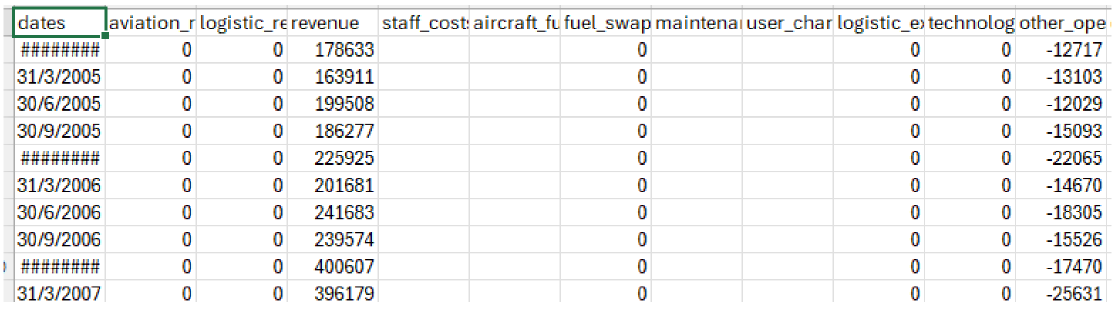
5.4.3 Data Imputation & Analysis

However, for the finance data, there are 14 columns that have missing data that are required to be inputted. The 14 columns are known as:

Table 5.4.3.1 Missing Columns with Missing Rows

	Columns with Missing Rows	**Missing Rows (%)**
1	Average stage length (km)	11.25
2	Unit fuel price (US$/barrel)	15
3	Total equity	6.10
4	Prepayment, deposits and other receivables	15
5	Sales in advance	26.25
6	Reserves	8.75
7	Staff costs	31.25
8	Aircraft fuel expenses	31.25
9	Maintenance, overhaul, user changes and other related expenses	31.25

10	User charges and other related expenses	33.75
11	Finance income	31.25
12	Finance costs	31.25
13	Number of aircraft at the period	3.75
14	Debt-To_Equity	6.10

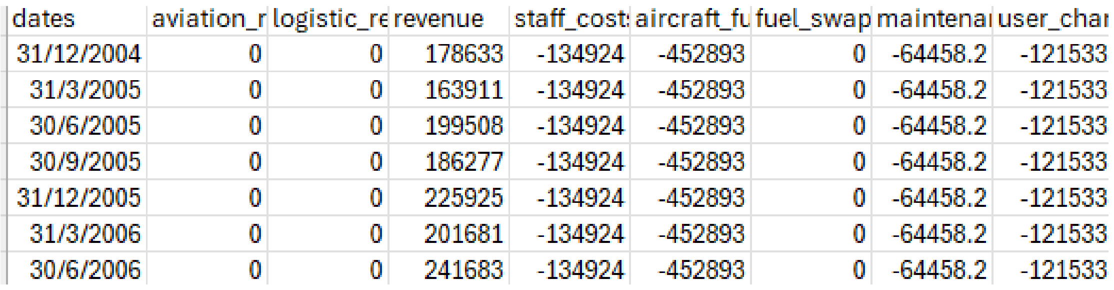


dates	aviation_r	logistic_re	revenue	staff_cost	aircraft_fu	fuel_swap	maintenai	user_char	logistic_e	technolog	other_ope
########	0	0	178633			0			0	0	-12717
31/3/2005	0	0	163911			0			0	0	-13103
30/6/2005	0	0	199508			0			0	0	-12029
30/9/2005	0	0	186277			0			0	0	-15093
########	0	0	225925			0			0	0	-22065
31/3/2006	0	0	201681			0			0	0	-14670
30/6/2006	0	0	241683			0			0	0	-18305
30/9/2006	0	0	239574			0			0	0	-15526
########	0	0	400607			0			0	0	-17470
31/3/2007	0	0	396179			0			0	0	-25631

Figure 5.4.3.1 Evidence of Missing Data

5.4.3.1 KNN Imputation

To handle the missing data, K-Nearest Neighbors (KNN) is conducted. First, the load_data function reads the CSV file, parsing dates as the index and converting all columns to numeric values, while also turning non-numeric entries such as blank cells into NaN. It then drops any columns that are entirely NaN. The knn_impute function then handles missing data by first creating a copy of the DataFrame and identifying missing values. Next, it standardizes the data using StandardScaler before applying KNNImputer, which fills NaN values based on neighboring data points. In this case, the n_neighbours are set to standard 5. This means the missing value will be imputed with the mean of the 5 closest rows' values for that feature. The imputed data is then rescaled back to its original range. Finally, the function returns to a DataFrame where only the originally missing values are replaced, preserving existing valid data.



dates	aviation_r	logistic_re	revenue	staff_cost	aircraft_fu	fuel_swap	maintenai	user_char
31/12/2004	0	0	178633	-134924	-452893	0	-64458.2	-121533
31/3/2005	0	0	163911	-134924	-452893	0	-64458.2	-121533
30/6/2005	0	0	199508	-134924	-452893	0	-64458.2	-121533
30/9/2005	0	0	186277	-134924	-452893	0	-64458.2	-121533
31/12/2005	0	0	225925	-134924	-452893	0	-64458.2	-121533
31/3/2006	0	0	201681	-134924	-452893	0	-64458.2	-121533
30/6/2006	0	0	241683	-134924	-452893	0	-64458.2	-121533

Figure 5.4.3.1.1 Evidence of Imputed Data using KNN imputation

Additionally, a scatter plot is used to compare the R-squared (R²) values of variables before and after imputation, helping assess how imputation affects model performance. In this case, variables with an absolute R² difference (rsq_diff) greater than 0.02 will be labelled. This avoids cluttering the plot with minor changes. Thus, focusing only on variables where imputation had a meaningful impact. Based on the figure below, most data predictably fall along the diagonal line which means the imputation had no effect. Those points that cluster **above the diagonal lines** means the imputed values strengthened the variable's relationship with the target. For instance, the reserves, prepayment_deposits_and_other_receivables and finance_income are the columns that slightly strengthened relationship with the target. At the same time, the points that cluster below the diagonal line means imputed values weakened the variable's relationship with the target, possibly introducing noise or bias. For instance, the sales_in_advance, total_equity, maintenance_overhaul_user_changes_and_other_related_expenses, finance_costs and average_stage_length_km are the columns that weakened relationship with the target.

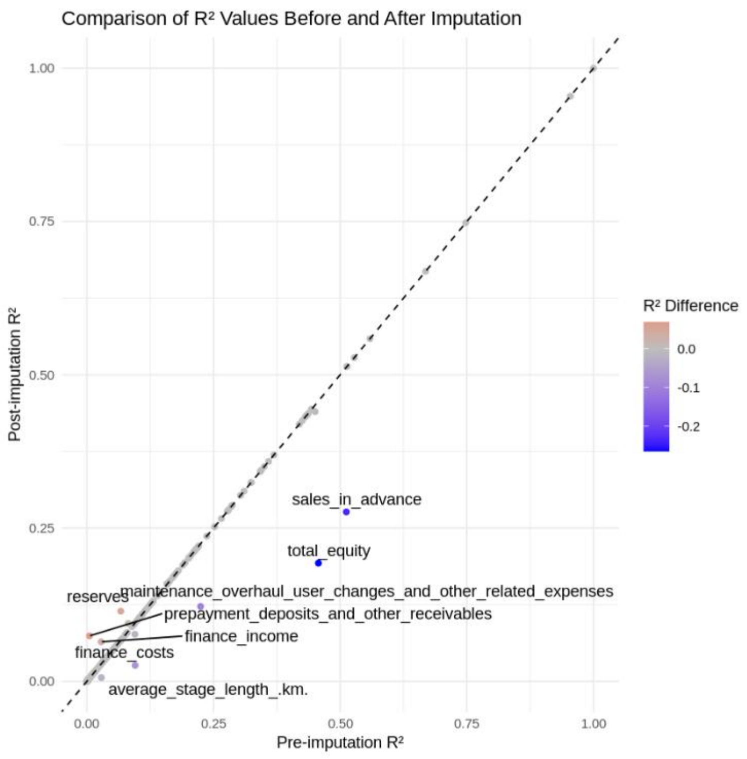
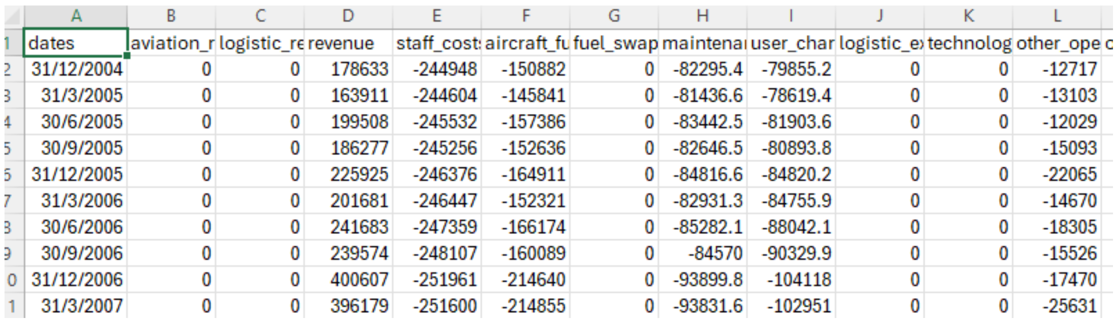


Figure 5.4.3.1.2 Comparison of R Square Values before and after KNN Imputation

5.4.3.2 FCMI Imputation

In this case, the data is imputed by prioritizing features with the strongest relationships to the target variable, which in this case is the column with missing data based on an article [45]. This type of imputation is known as Feature Correlation based Missing Data Imputation.

For instance, the select_top_correlated_features() function identifies the top-k features most correlated with a target column using only complete cases. The core fcmi_imputation() function performs feature-correlation-based imputation: it trains a linear regression model on complete cases using the top-k correlated features to predict missing values in the target column, first imputing missing values in the features themselves using mean imputation. Next, the impute_all_missing_columns() function imputes all columns with missing values in ascending order of missingness to maximize available data for subsequent imputations. This approach combines correlation-based feature selection with regression imputation, making it more sophisticated than simple mean/median imputation while remaining computationally efficient. The sorting by missing rate ensures columns with fewer missing values are imputed first. Thus, creating better anchors for later imputations.



	A	B	C	D	E	F	G	H	I	J	K	L
1	dates	aviation_r	logistic_re	revenue	staff_cost	aircraft_fu	fuel_swap	maintenai	user_char	logistic_e	technolog	other_ope
2	31/12/2004	0	0	178633	-244948	-150882	0	-82295.4	-79855.2	0	0	-12717
3	31/3/2005	0	0	163911	-244604	-145841	0	-81436.6	-78619.4	0	0	-13103
4	30/6/2005	0	0	199508	-245532	-157386	0	-83442.5	-81903.6	0	0	-12029
5	30/9/2005	0	0	186277	-245256	-152636	0	-82646.5	-80893.8	0	0	-15093
6	31/12/2005	0	0	225925	-246376	-164911	0	-84816.6	-84820.2	0	0	-22065
7	31/3/2006	0	0	201681	-246447	-152321	0	-82931.3	-84755.9	0	0	-14670
8	30/6/2006	0	0	241683	-247359	-166174	0	-85282.1	-88042.1	0	0	-18305
9	30/9/2006	0	0	239574	-248107	-160089	0	-84570	-90329.9	0	0	-15526
10	31/12/2006	0	0	400607	-251961	-214640	0	-93899.8	-104118	0	0	-17470
11	31/3/2007	0	0	396179	-251600	-214855	0	-93831.6	-102951	0	0	-25631

Figure 5.4.3.2.1 Evidence of Imputed Data using FCMI imputation

Based on the figure below, most data predictably fall along the diagonal line which means the imputation had no effect. Those points that cluster **above the diagonal lines** means the imputed values strengthened the variable's relationship with the target. For instance, the aircraft_fuel expenses and prepayment_deposits_and_other_receivables are the columns that slightly strengthened relationship with the target. At the same time, the points that cluster below the diagonal line means imputed values weakened the variable’s relationship with the target, possibly introducing noise or bias. For instance, the sales_in_advance, total_equity, maintenance_overhaul_user_changes_and_other_related_expenses, finance_costs and finance income are the columns that weakened relationship with the target.

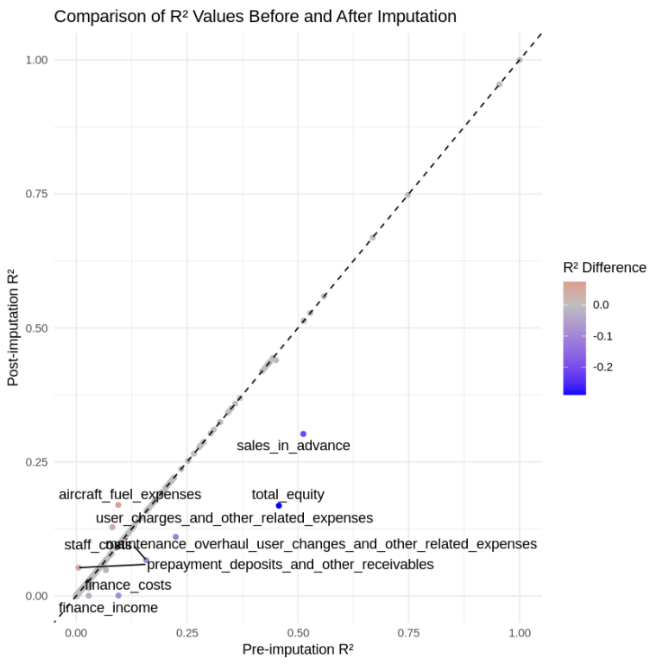


Figure 5.4.3.2.2 Comparison of R Square Values before and after FCMI Imputation

5.4.4 Time Series Analysis

5.4.4.1 Decision Tree Modelling for Feature Selection

In this phase, the python script utilizes the **Decision Tree Regressor** to analyze financial data, aiming to identify and focus on the most significant features that impact the modified Altman Z-Score. First, the script separates datetime columns from the features (X) and assigns the Altman Z-Score as the target variable (y). The data is then split into training and testing sets using a 70-30 ratio. The parameters of the decision tree model are set with a max_depth of 5 to prevent overfitting. The model is then feed with the training data, and extracts feature importance to identify the most influential features. The model is then retrained using only these important features by filtering out zero-importance ones to improve efficiency and interpretability. Finally, it evaluates the model's performance on the test set using Mean Squared Error (MSE) and R^2 score and saves the feature importance rankings to a CSV file.

Next, the two imputed finance data that are implemented using KNN imputation and FCMI (Fully Conditional Specification Multiple Imputation) are then feed to the decision tree model. The results reveal notable differences. Notably, the finance data using both methods have the period between 31/12/2020 to 31/12/2021 removed since they weren't aligned with the PN17 classification. This is to ensure that the decision tree model only select features that follows the PN17 classification by reducing the noise and retaining only relevant finance data.

Table 5.4.4.1.1 Table of Important Features from KNN Imputed Finance Data

	Important Features from KNN Imputed Finance Data	**Importance**
1	depreciation_and_amortisation/_depreciation_of_ property_plant_and_equiptment	0.7885931
2	debt-to-assets_ratio	0.1751300
3	debt_to_equity	0.0181335
4	purchase_of_property_plant_and_equipment	0.0094325
5	current_ratio	0.0038242
6	cash_and_cash_equivalents_at_end_of_the_financial_period	0.0023100
7	intercompany_balances	0.0013784
8	share_of_results_of_jointly_controlled_company_adjustment	0.0003827
9	unrealised_foreign_exchange_plan	0.0003761
10	teleport_(logistics)_non_airline_ebitda	0.0002412
11	current_taxation	0.0001379
12	sectors_flown/_no._of_flights	0.0000581
13	receivables_and_prepayments	0.0000022

Table 5.4.4.1.2 Table of Important Features from FCMI Imputed Finance Data

	Important Features from FCMI Imputed Finance Data	**Importance**
1	depreciation_and_amortisation/_depreciation_of_ property_plant_and_equiptment	0.7885931
2	debt-to-assets_ratio	0.1751300
3	debt_to_equity	0.0181335
4	purchase_of_property_plant_and_equipment	0.0094325
5	current_ratio	0.0038242
6	share_of_results_of_jointly_controlled_company_adjustment	0.0026927
7	intercompany_balances	0.0013784
8	unrealised_foreign_exchange_plan	0.0003761
9	teleport_(logistics)_non_airline_ebitda	0.0002412
10	current_taxation	0.0001379
11	sectors_flown/_no._of_flights	0.0000581
12	receivables_and_prepayments	0.0000022

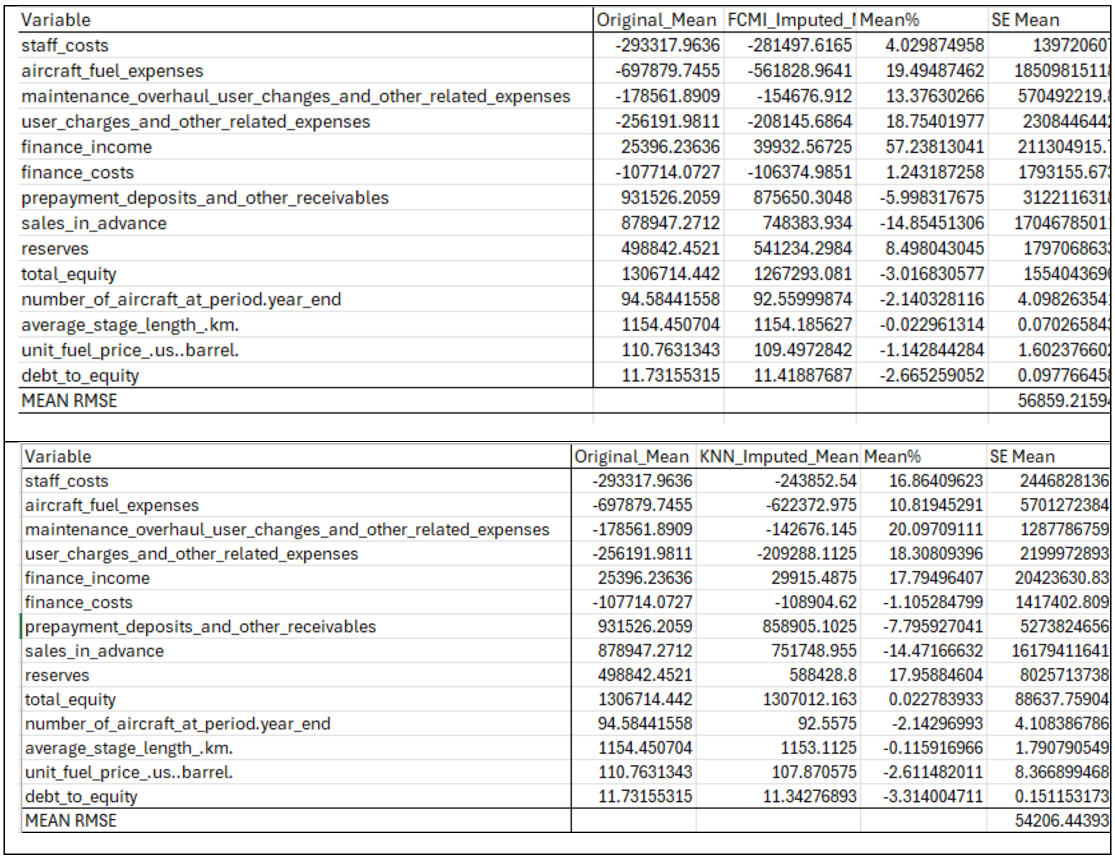
With KNN-imputed data, the model selects 13 important features out of 305, yielding an MSE of 1.2622 and an R^2 of 0.9152. While still strong, these metrics are outperformed by the FCMI-imputed dataset, where only 12 features are deemed important. Thus, achieving a lower MSE of 0.8231 and higher R^2 of 0.9447.

Notably, both methods share 13 identical features with closely aligned importance scores, suggesting these variables are robust predictors regardless of imputation technique. However, the superior performance of FCMI underscores its effectiveness in preserving meaningful data relationships during imputation. Besides, both methods also share the same top five overlapping features with the same R square score of 0.9215 after being further cut from the 13 and 12 features.

Table 5.4.4.1.3 Table of Overlapping Top 5 Important Features from KNN and FCMI Imputed Finance Data

	Overlapping Top 5 Important Features from KNN and FCMI Imputed Finance Data	**Importance**
1	depreciation_and_amortisation/_depreciation_of_ property_plant_and_equiptment	0.7885931
2	debt-to-assets_ratio	0.1751300
3	debt_to_equity	0.0181335
4	purchase_of_property_plant_and_equipment	0.0094325
5	current_ratio	0.0038242

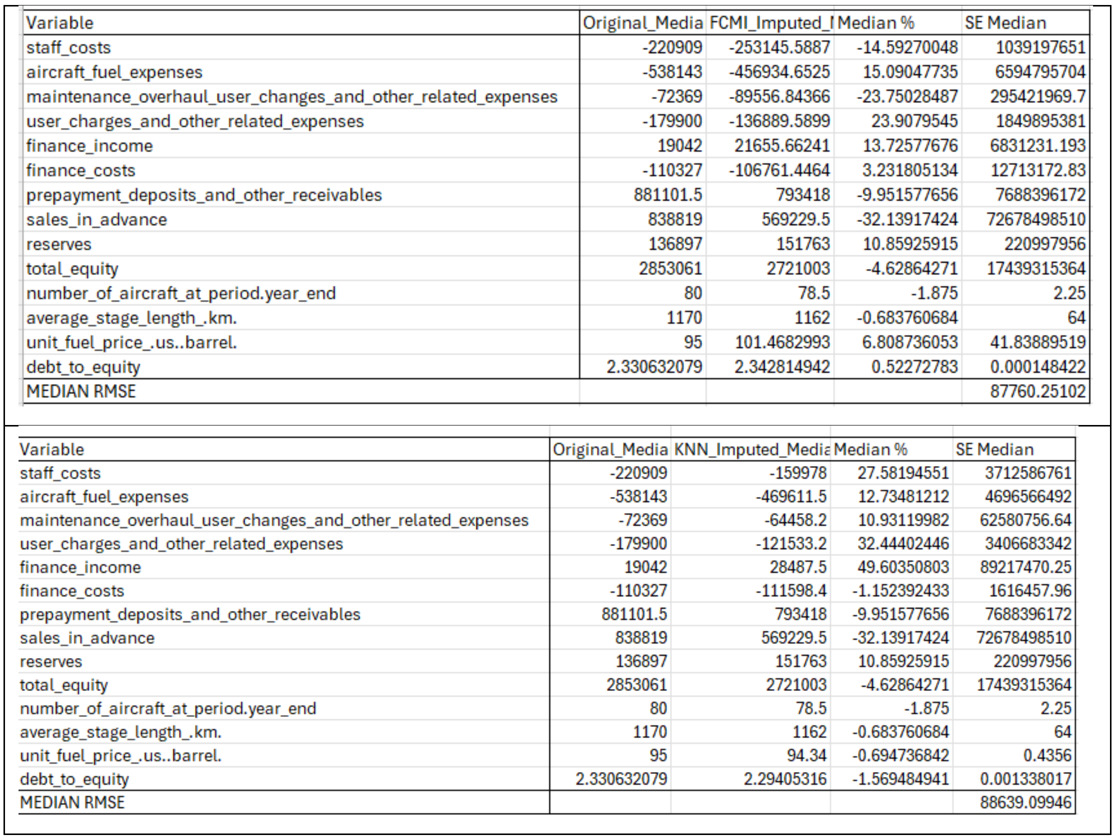
For further explanation in the high overlap in important features and the similar importance scores between KNN and FCMI imputation methods, likely occur because only 14 of the 305 original columns required imputation. Besides, both methods introduced comparable adjustments to missing values, as evidenced by their nearly identical squared errors against the original data's mean (FCMI: 56,859.22 vs KNN: 54,206.44) and median (FCMI: 87,760.25 vs KNN: 88,639.10). This minimal divergence suggests the imputations preserved the dataset's core statistical relationships, causing the decision tree model to identify largely the same predictive features from both versions



Variable	Original_Mean	FCMI_Imputed_I	Mean%	SE Mean
staff_costs	-293317.9636	-281497.6165	4.029874958	13972060
aircraft_fuel_expenses	-697879.7455	-561828.9641	19.49487462	1850981511
maintenance_overhaul_user_changes_and_other_related_expenses	-178561.8909	-154676.912	13.37630266	570492219.
user_charges_and_other_related_expenses	-256191.9811	-208145.6864	18.75401977	230844644
finance_income	25396.23636	39932.56725	57.23813041	211304915.
finance_costs	-107714.0727	-106374.9851	1.243187258	1793155.67
prepayment_deposits_and_other_receivables	931526.2059	875650.3048	-5.998317675	312211631
sales_in_advance	878947.2712	748383.934	-14.85451306	1704678501
reserves	498842.4521	541234.2984	8.498043045	179706863
total_equity	1306714.442	1267293.081	-3.016830577	155404369
number_of_aircraft_at_period.year_end	94.58441558	92.55999874	-2.140328116	4.0982635
average_stage_length_.km.	1154.450704	1154.185627	-0.022961314	0.0702658
unit_fuel_price_.us..barrel.	110.7631343	109.4972842	-1.142844284	1.6023766
debt_to_equity	11.73155315	11.41887687	-2.665259052	0.0977664
MEAN RMSE				56859.2159

Variable	Original_Mean	KNN_Imputed_Mean	Mean%	SE Mean
staff_costs	-293317.9636	-243852.54	16.86409623	2446828136
aircraft_fuel_expenses	-697879.7455	-622372.975	10.81945291	5701272384
maintenance_overhaul_user_changes_and_other_related_expenses	-178561.8909	-142676.145	20.09709111	1287786759
user_charges_and_other_related_expenses	-256191.9811	-209288.1125	18.30809396	2199972893
finance_income	25396.23636	29915.4875	17.79496407	20423630.83
finance_costs	-107714.0727	-108904.62	-1.105284799	1417402.809
prepayment_deposits_and_other_receivables	931526.2059	858905.1025	-7.795927041	5273824656
sales_in_advance	878947.2712	751748.955	-14.47166632	16179411641
reserves	498842.4521	588428.8	17.95884604	8025713738
total_equity	1306714.442	1307012.163	0.022783933	88637.75904
number_of_aircraft_at_period.year_end	94.58441558	92.5575	-2.14296993	4.108386786
average_stage_length_.km.	1154.450704	1153.1125	-0.115916966	1.790790549
unit_fuel_price_.us..barrel.	110.7631343	107.870575	-2.611482011	8.366899468
debt_to_equity	11.73155315	11.34276893	-3.314004711	0.151153173
MEAN RMSE				54206.44393

Table 5.4.4.1.4 Comparison Between the figure of FCMI imputed mean against original mean and the figure of KNN imputed mean against original mean.



Variable	Original_Media	FCMI_Imputed_I	Median %	SE Median
staff_costs	-220909	-253145.5887	-14.59270048	1039197651
aircraft_fuel_expenses	-538143	-456934.6525	15.09047735	6594795704
maintenance_overhaul_user_changes_and_other_related_expenses	-72369	-89556.84366	-23.75028487	295421969.7
user_charges_and_other_related_expenses	-179900	-136889.5899	23.9079545	1849895381
finance_income	19042	21655.66241	13.72577676	6831231.193
finance_costs	-110327	-106761.4464	3.231805134	12713172.83
prepayment_deposits_and_other_receivables	881101.5	793418	-9.951577656	7688396172
sales_in_advance	838819	569229.5	-32.13917424	72678498510
reserves	136897	151763	10.85925915	220997956
total_equity	2853061	2721003	-4.62864271	17439315364
number_of_aircraft_at_period.year_end	80	78.5	-1.875	2.25
average_stage_length_.km.	1170	1162	-0.683760684	64
unit_fuel_price_.us..barrel.	95	101.4682993	6.808736053	41.83889519
debt_to_equity	2.330632079	2.342814942	0.52272783	0.000148422
MEDIAN RMSE				87760.25102

Variable	Original_Media	KNN_Imputed_Media	Median %	SE Median
staff_costs	-220909	-159978	27.58194551	3712586761
aircraft_fuel_expenses	-538143	-469611.5	12.73481212	4696566492
maintenance_overhaul_user_changes_and_other_related_expenses	-72369	-64458.2	10.93119982	62580756.64
user_charges_and_other_related_expenses	-179900	-121533.2	32.44402446	3406683342
finance_income	19042	28487.5	49.60350803	89217470.25
finance_costs	-110327	-111598.4	-1.152392433	1616457.96
prepayment_deposits_and_other_receivables	881101.5	793418	-9.951577656	7688396172
sales_in_advance	838819	569229.5	-32.13917424	72678498510
reserves	136897	151763	10.85925915	220997956
total_equity	2853061	2721003	-4.62864271	17439315364
number_of_aircraft_at_period.year_end	80	78.5	-1.875	2.25
average_stage_length_.km.	1170	1162	-0.683760684	64
unit_fuel_price_.us..barrel.	95	94.34	-0.694736842	0.4356
debt_to_equity	2.330632079	2.29405316	-1.569484941	0.001338017
MEDIAN RMSE				88639.09946

Table 5.4.4.1.5 Comparison Between the figure of FCMI imputed median against original median and the figure of KNN imputed median against original median.

While both methods yielded 13-12 key features with strong agreement, FCMI's slightly better performance (lower MSE of 0.8231 vs 1.2622 and higher R² of 0.9447 vs 0.9152) may reflect its superior handling of multivariate relationships during imputation. The results validate using these overlapping features, particularly from FCMI for subsequent forecasting models. This is because they reliably capture the underlying financial drivers of the Altman Z-Score regardless of imputation technique.

Therefore, the 12 key features from FCMI imputation, along with the top five overlapping features identified by both methods, will each serve as inputs for an LSTM model designed to forecast future Altman Z-Scores.

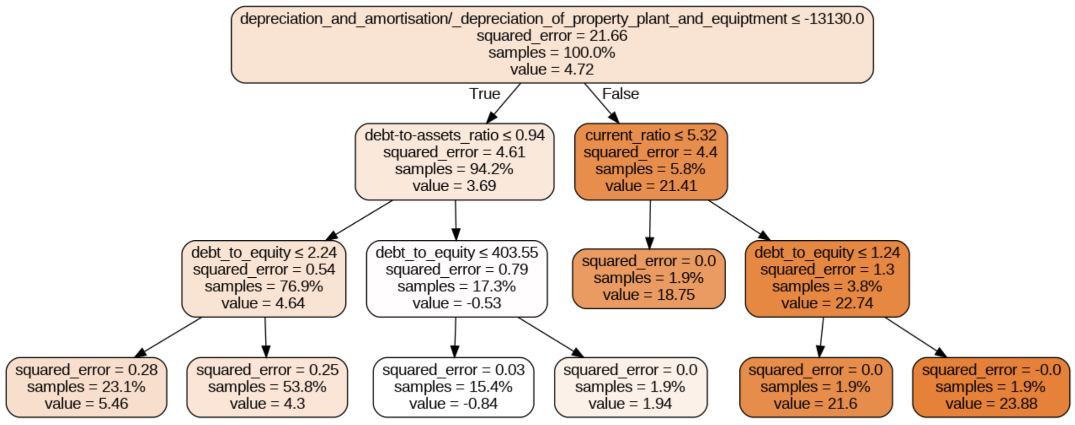


Figure 5.4.4.1.1 Decision Tree Diagram of the Top 5 Overlapping Important Features

5.4.4.2 LSTM Modeling

From the 305 features, the 12 features selected from the FCMI imputed finance data, and the 5 overlapping features selected from the decision tree shall be used to forecast the Altman Z-score in two separate LSTM models before comparison. In this case the code implemented is used to predict quarterly time series data, specifically the Altman Z-Score.

It starts by setting random seeds for reproducibility and loading the dataset from a CSV file. The data is preprocessed using StandardScaler to normalize features and split into sequences with a look-back window of 5-time steps as seen in Table 5.4.4.2.1. The

LSTM model, built with TensorFlow/Keras, consists of two LSTM layers (64 and 32 units) followed by dense layers (16 and 1 unit), using ReLU activation and Glorot uniform initialization for weights. The model is trained using time-series cross-validation (5 splits) with early stopping to prevent overfitting. Predictions are inverse transformed to their original scale. Additionally, the performance metrics (MAE, MSE) are calculated for each fold. Finally, the results (actual vs. predicted Z-Scores) are visualized in a plot.

Here are the figure and table based on the top 12 features using LSTM model:

Table 5.4.4.2.1 MAE and MSE of top 12 Features used in LSTM model

Fold	**Period Trained**	**Period Tested**	**MAE**	**MSE**
1	31/12/2004 - 31/03/2012	30/06/2012 - 30/09/2014	0.217	0.070
2	31/12/2004 - 30/09/2014	31/12/2014 - 31/03/2017	0.452	0.320
3	31/12/2004 - 31/03/2017	30/06/2017 - 30/09/2019	0.501	0.474
4	31/12/2004 - 30/09/2019	31/12/2019 - 31/03/2022	3.280	13.608
5	31/12/2004 - 31/03/2022	30/06/2022 - 30/09/2024	0.734	0.860

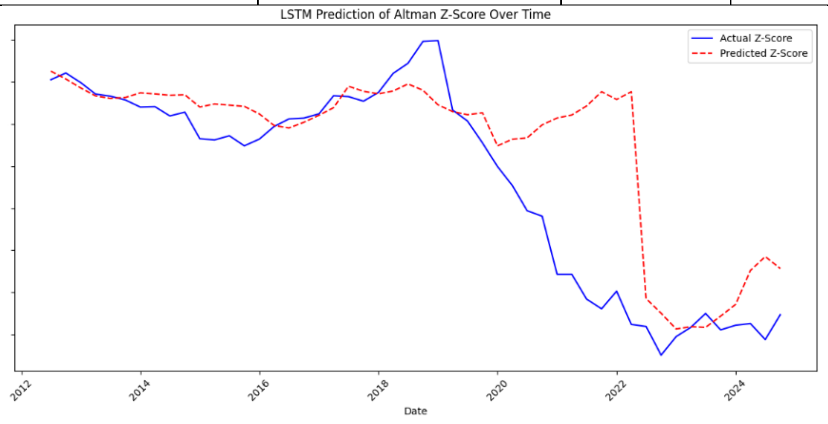


Figure 5.4.4.2.1 LSTM diagram of top 12 features used to forecast modified Altman Z-Score

Here are the figure and table based on the top 5 features using LSTM model:

Table 5.4.4.2.2 MAE and MSE of top 5 Features used in LSTM model

Fold	Period Trained	Period Tested	MAE	MSE
1	31/12/2004 - 31/03/2012	30/06/2012 - 30/09/2014	0.782	0.862
2	31/12/2004 - 30/09/2014	31/12/2014 - 31/03/2017	0.494	0.343

3	31/12/2004 - 31/03/2017	30/06/2017 - 30/09/2019	0.989	1.871
4	31/12/2004 - 30/09/2019	31/12/2019 - 31/03/2022	3.761	16.270
5	31/12/2004 - 31/03/2022	30/06/2022 - 30/09/2024	1.012	1.364

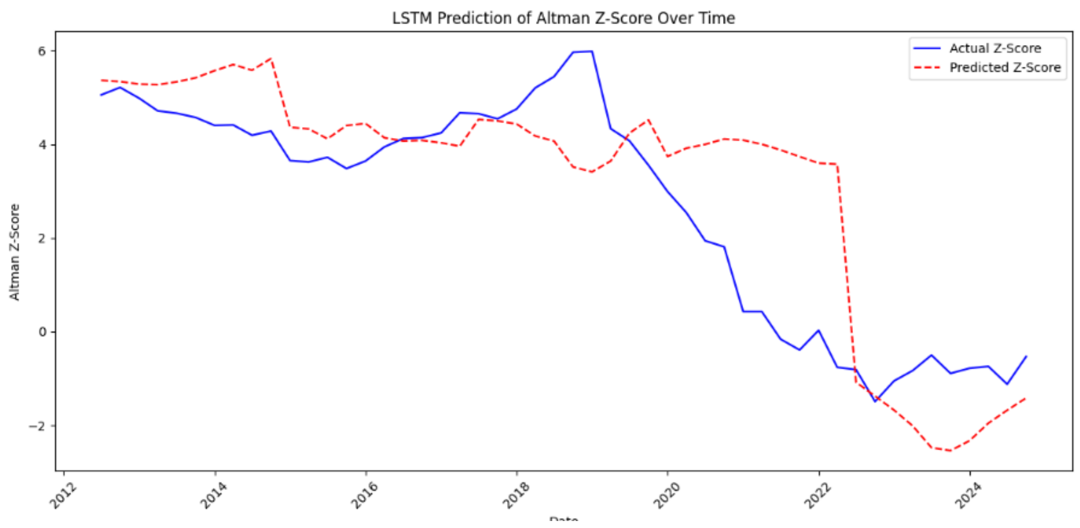


Figure 5.4.4.2.2 LSTM diagram of top 5 features used to forecast modified Altman Z-Score

Based on Table 5.4.4.2.3, the LSTM model utilizing the top 12 important features from FCMI-imputed financial data demonstrates superior performance by achieving average MAE and MSE scores of 1.0367 and 3.0663, respectively. Coincidentally, both models perform terribly on the fourth fold. This decline can be attributed to the training data (31/12/2004 - 30/09/2019) failing to capture the unprecedented patterns present in the test data (31/12/2019 - 31/03/2022). This is because this period coincides with the company's first ever period of financial distress, as classified under PN17, which was caused by the COVID-19 pandemic which is an event with no historical precedent. Hence, the model's inability to adapt to these novel conditions explains the spike in errors. However, the subsequent reduction in MAE and MSE during the fifth fold indicates that the model regained stability once the pandemic's immediate impact subsided.

Table 5.4.4.2.3 Comparison between Average MAE and MSE of top 12 Features and top 5 Features used in LSTM model

	Average MAE	**Average MSE**
Using the top 12 important features	1.0367	3.0663
Using the top 5 important features	1.4076	4.1419

5.4.5 Sentiment Analysis

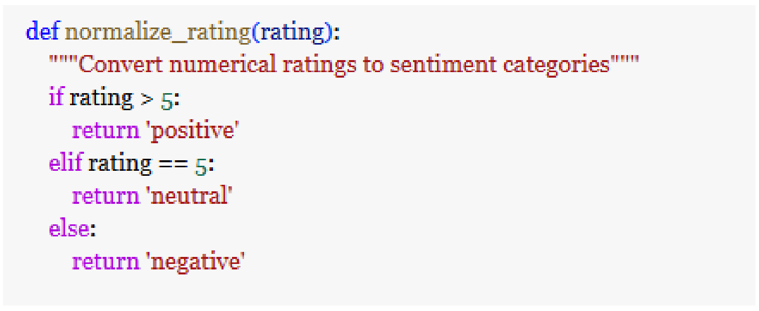
To ensure accurate sentiment analysis, the preprocessed 852 text data from the year 2011 to 2024 will be evaluated using three models: TextBlob, FinBERT, and BERT. Their performance will be assessed through classification reports and confusion matrices, with sentiment scores validated against the actual review ratings. Therefore, the most reliable sentiment scores will then be used as inputs to forecast the Altman Z-Score which is known as a financial distress metric via an LSTM model. Hence, making a more comprehensive financial distress forecasting model for Capital A Berhad.

5.4.5.1 Text Blob Modelling

The preprocessed text data first undergo sentiment analysis by using add_textblob_analysis function. It is used to calculate two numerical metrics known as polarity (ranging from -1 for negative to +1 for positive sentiment) and subjectivity (ranging from 0 for objective to 1 for subjective text) for each entry in the specified text column under the name of 'Cleaned_Review'. Additionally, it assigns a categorical sentiment label known as 'positive', 'negative', or 'neutral' based on the polarity score. Next, The function returns the DataFrame with the original columns (e.g., timestamp, text, rating, source) alongside the new sentiment columns, streamlining further analysis of textual sentiment trends.

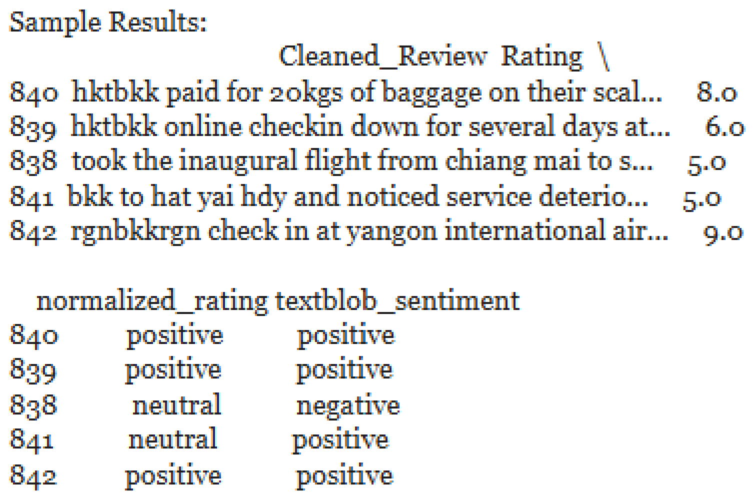
The figure below shows how the reviews are categorized according to the levels of their sentiment polarity.

Figure 5.4.5.1.1 Text Blob Sentiment Polarity Categorizing Scale



```
def normalize_rating(rating):
    """Convert numerical ratings to sentiment categories"""
    if rating > 5:
        return 'positive'
    elif rating == 5:
        return 'neutral'
    else:
        return 'negative'
```

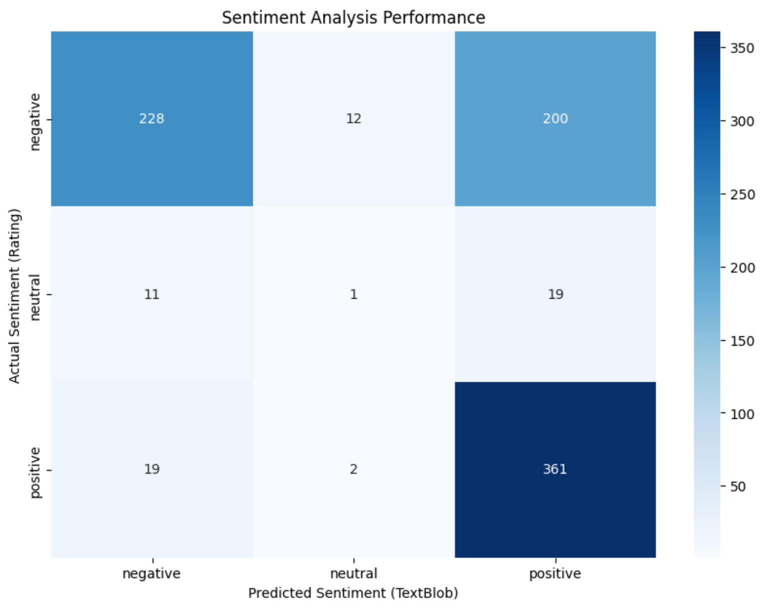
Figure 5.4.5.1.2 Example Output of the Text Blob Sentiment Data after Sentiment Analysis



```
Sample Results:
                                Cleaned_Review  Rating  \
840  hktbkk paid for 20kgs of baggage on their scal...     8.0
839  hktbkk online checkin down for several days at...     6.0
838  took the inaugural flight from chiang mai to s...     5.0
841  bkk to hat yai hdy and noticed service deterio...     5.0
842  rgnbkkrgn check in at yangon international air...     9.0

    normalized_rating textblob_sentiment
840          positive           positive
839          positive           positive
838           neutral           negative
841           neutral           positive
842          positive           positive
```

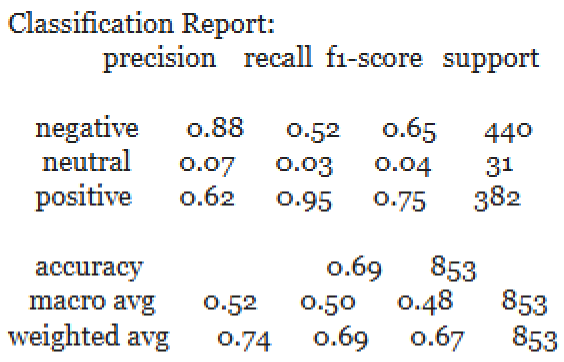
The confusion matrix reveals TextBlob's performance in classifying sentiments against actual ratings. For negative sentiment, the model correctly identified 228 reviews but misclassified 212, which can be further break down into 12 as neutral and 200 as positive. In fact, the neutral sentiment had the weakest performance, with only 1 correct prediction out of 31. This is because most of the neutral sentiments were mislabeled as positive. In contrast, positive sentiment showed strong detection, with 361 correct predictions. However, there are also some which are also falsely labeled. This indicates that while TextBlob excels at identifying positive sentiments. At the same time, it struggles with neutral and negative classifications, particularly in distinguishing negative from positive reviews. This is probably due to class imbalance with majority text data in negative class and positive class instead of neutral class.

Figure 5.4.5.1.3 Text Blob Confusion Matrix



The classification report provides deeper metrics for each sentiment class. Negative sentiment achieved 88% precision but only 52% recall, meaning while predictions were mostly correct, many actual negatives were missed. Neutral sentiment performed poorly across all metrics with 7% precision and 3% recall. Thus, suggesting the model fails to recognize neutral content. At the same time, the positive sentiment had a high recall of 95% but moderate precision of 62%. This means it has over-prediction of positives. In short, Text Blob have an overall accuracy of 69%. However, the macro-average F1-score of 0.48 highlights significant imbalance, with neutral sentiment dragging down performance.

Figure 5.4.5.1.4 Text Blob Classification Report



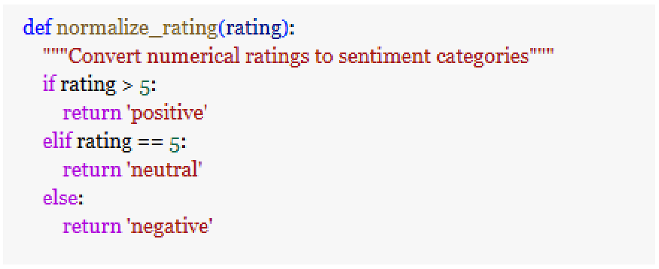
```
Classification Report:
              precision    recall  f1-score   support

    negative       0.88      0.52      0.65       440
     neutral       0.07      0.03      0.04        31
    positive       0.62      0.95      0.75       382

    accuracy                           0.69       853
   macro avg       0.52      0.50      0.48       853
weighted avg       0.74      0.69      0.67       853
```

5.4.5.2 FinBERT Modeling

In this case to access the specialized BERT model trained for financial text, the pre-trained "yiyanghkust/finbert-tone" model is loaded along with its tokenizer. Then, the predict_sentiment() function processes input text by tokenizing it. This is done by using truncation and padding for consistency before feeding it to FinBERT, and extracting the predicted sentiment class. Next, the labeled sentiment data from FinBERT (0=neutral, 1=positive, 2=negative) is mapped to a readable label. The analyze_with_finbert() function then applies this prediction to a DataFrame column adding a new column ("finbert_sentiment") with the results. This enables domain-specific sentiment analysis for financial texts like earnings reports or news.

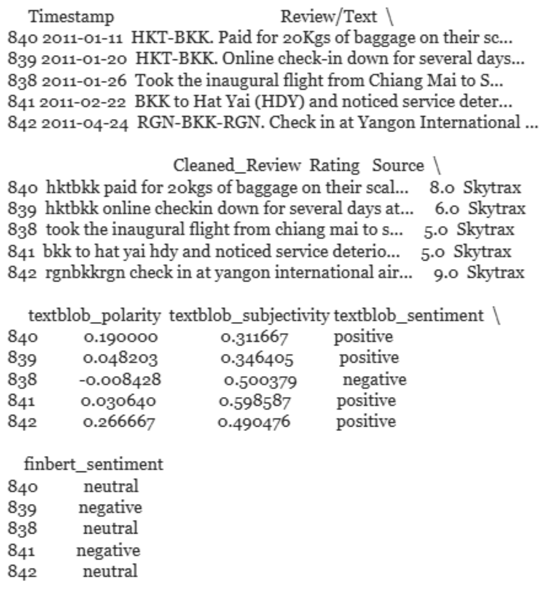
The figure below shows how the reviews are categorized according to the levels of their sentiment polarity.

Figure 5.4.5.2.1 FinBERT Sentiment Polarity Categorizing Scale



```python
def normalize_rating(rating):
  """Convert numerical ratings to sentiment categories"""
  if rating > 5:
    return 'positive'
  elif rating == 5:
    return 'neutral'
  else:
    return 'negative'
```

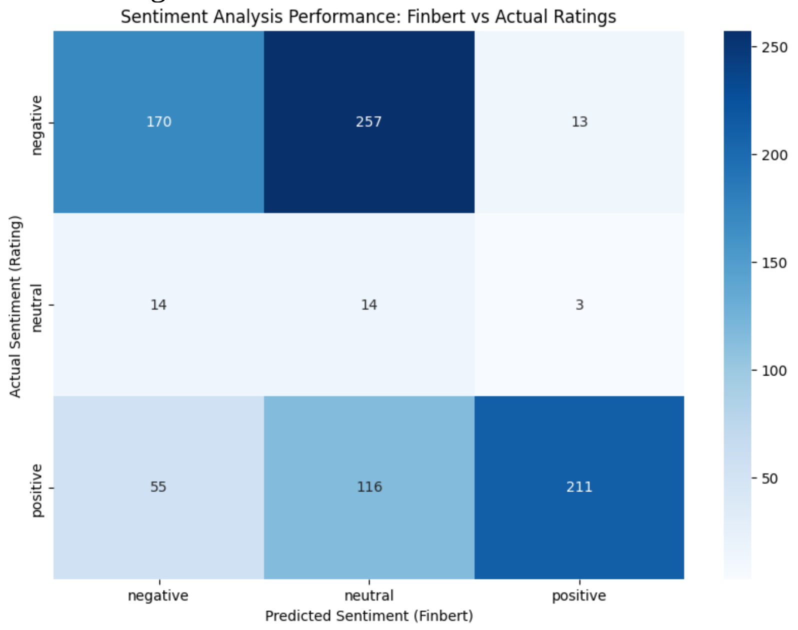
Figure 5.4.5.2.2 Example Output of the FinBERT Sentiment Data after Sentiment Analysis



```
      Timestamp                              Review/Text  \
840 2011-01-11  HKT-BKK. Paid for 20Kgs of baggage on their sc...
839 2011-01-20  HKT-BKK. Online check-in down for several days...
838 2011-01-26  Took the inaugural flight from Chiang Mai to S...
841 2011-02-22  BKK to Hat Yai (HDY) and noticed service deter...
842 2011-04-24  RGN-BKK-RGN. Check in at Yangon International ...

                                    Cleaned_Review  Rating   Source  \
840  hktbkk paid for 20kgs of baggage on their scal...     8.0  Skytrax
839  hktbkk online checkin down for several days at...     6.0  Skytrax
838  took the inaugural flight from chiang mai to s...     5.0  Skytrax
841  bkk to hat yai hdy and noticed service deterio...     5.0  Skytrax
842  rgnbkkrgn check in at yangon international air...     9.0  Skytrax

     textblob_polarity  textblob_subjectivity textblob_sentiment  \
840           0.190000               0.311667           positive
839           0.048203               0.346405           positive
838          -0.008428               0.500379           negative
841           0.030640               0.598587           positive
842           0.266667               0.490476           positive

    finbert_sentiment
840           neutral
839          negative
838           neutral
841          negative
842           neutral
```

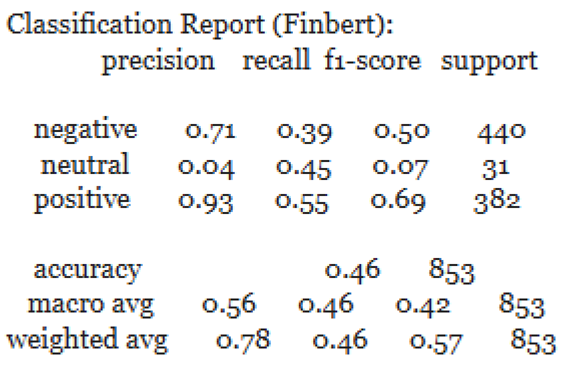
According to the confusion matrix, for negative sentiment, FinBERT correctly identified 170 reviews but misclassified 270, with 257 negative texts incorrectly label as positive and 13 as neutral. This indicates a difficulty in distinguishing negative from positive sentiments. Next, the neutral sentiment showed slightly better performance than TextBlob, with 14 correct predictions, but still suffered from misclassifications with 14 neutral texts incorrectly labeled as negative and 3 as positive. In comparison, positive sentiment had moderate success, with 211 correct predictions, but a significant portion was mislabeled with 55 as negative and 116 as neutral. Therefore, this suggests FinBERT struggles to accurately classify neutral and negative sentiments. This is likely due to its training in financial reports, which often employ cautious or positively skewed language to mitigate stakeholder concerns. When applied to customer reviews which is the text data we used, the text data have a different domain with more direct and varied emotional expressions. Hence, this bias causes misalignment. Specifically, FinBERT's tendency to over-predict negative sentiment which is observed in a high false-negative rate may stem from its financial domain training, where even neutral statements are treated conservatively. This mismatch highlights the need for domain adaptation or hybrid modeling to improve granularity.

Figure 5.4.5.2.3 FinBERT Confusion Matrix



In the classification report, negative sentiment achieved 71% precision but only 39% recall. This means while predictions were often correct, many true negatives were missed. At the same time, the neutral sentiment had abysmal precision of 4% but a higher recall of 45%. Thus, reflecting over-prediction of neutrals at the cost of accuracy. Positive sentiment excelled in precision of 93% but had modest recall of 55%. Thus, highlighting conservative predictions that missed many true positives. In short, the overall accuracy of 46% and macro F1-score of 0.42 underscore poor balance across classes, with neutral sentiment dragging down performance.

Figure 5.4.5.2.4 FinBERT Classification Report

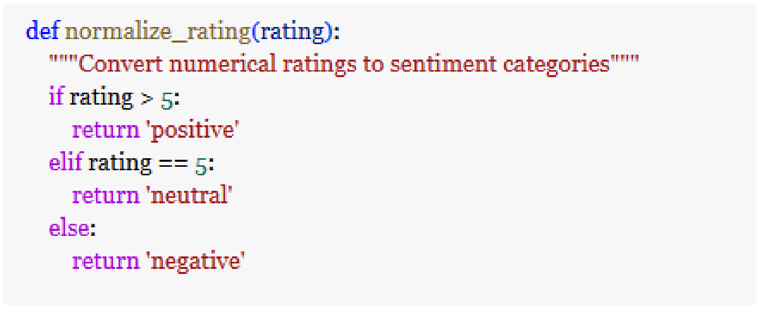


```
Classification Report (Finbert):
              precision    recall  f1-score   support

    negative       0.71      0.39      0.50       440
     neutral       0.04      0.45      0.07        31
    positive       0.93      0.55      0.69       382

    accuracy                           0.46       853
   macro avg       0.56      0.46      0.42       853
weighted avg       0.78      0.46      0.57       853
```

5.4.5.3 BERT Modeling

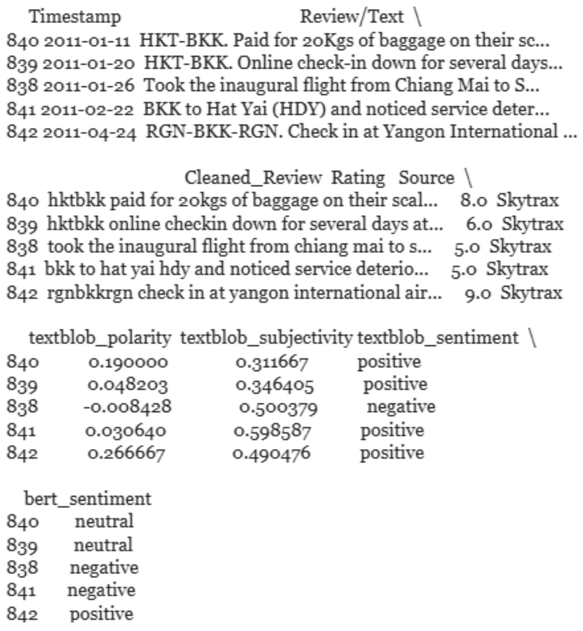
RoBERTa which is a transformer model, is fine-tuned for Twitter sentiment and loaded using "cardiffnlp/twitter-roberta-base-sentiment" along with it's tokenizer. In fact, the tokenizer preprocesses text by truncating/padding inputs to 512 tokens for consistency. The predict_batch function processes text in batches with default size 8 for efficiency. Then it tokens each batch and feeds it to the model without gradient computation to save memory. The model's logits, also known as raw predictions, are converted to class labels known as "positive," "neutral," "negative" using the model's configuration mapping. Next, the analyze_with_bert function applies this batch prediction to a DataFrame adding a new column "bert_sentiment" with the predicted labels.

Figure 5.4.5.3.1 RoBERTa Sentiment Polarity Categorizing Scale



```python
def normalize_rating(rating):
    """Convert numerical ratings to sentiment categories"""
    if rating > 5:
        return 'positive'
    elif rating == 5:
        return 'neutral'
    else:
        return 'negative'
```

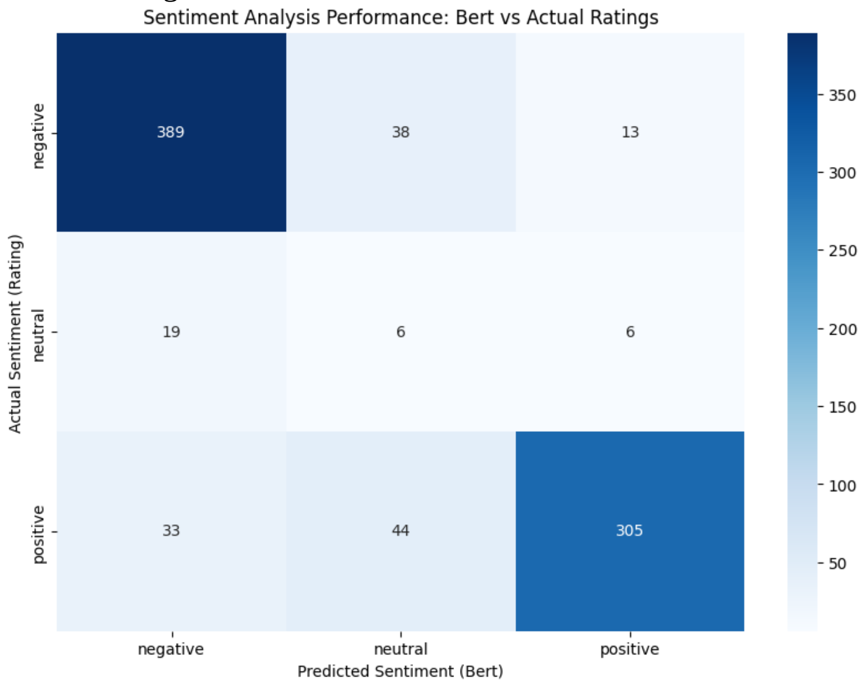
Figure 5.4.5.3.2 Example Output of the RoBERTa Sentiment Data after Sentiment Analysis



```
   Timestamp                       Review/Text  \
840 2011-01-11  HKT-BKK. Paid for 20Kgs of baggage on their sc...
839 2011-01-20  HKT-BKK. Online check-in down for several days...
838 2011-01-26  Took the inaugural flight from Chiang Mai to S...
841 2011-02-22  BKK to Hat Yai (HDY) and noticed service deter...
842 2011-04-24  RGN-BKK-RGN. Check in at Yangon International ...

                             Cleaned_Review  Rating  Source  \
840  hktbkk paid for 20kgs of baggage on their scal...    8.0  Skytrax
839  hktbkk online checkin down for several days at...    6.0  Skytrax
838  took the inaugural flight from chiang mai to s...    5.0  Skytrax
841  bkk to hat yai hdy and noticed service deterio...    5.0  Skytrax
842  rgnbkkrgn check in at yangon international air...    9.0  Skytrax

   textblob_polarity  textblob_subjectivity textblob_sentiment  \
840         0.190000              0.311667           positive
839         0.048203              0.346405           positive
838        -0.008428              0.500379           negative
841         0.030640              0.598587           positive
842         0.266667              0.490476           positive

   bert_sentiment
840       neutral
839       neutral
838      negative
841      negative
842      positive
```

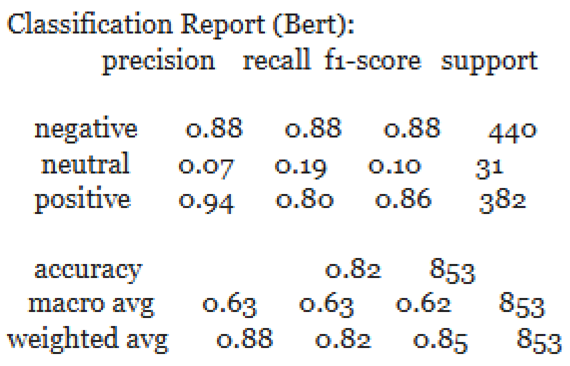
The confusion matrix reveals that RpBERTa excels at classifying negative sentiment by correctly identifying 389 out of 440 negative reviews with an accuracy of 88% with minimal misclassifications. As for neutral sentiment, performance is weak with only 6 correct predictions out of 31 and 19 misclassified as negative and 6 misclassified positive. This indicates difficulty in detecting nuanced neutral language. At the same time, positive sentiment shows strong but imperfect results, with 305 correct predictions and a recall of 80%. However, 33 were mislabeled as negative and 44 as neutral. This suggests RoBERTa’s Twitter-trained model prioritizes clear-cut sentiments such as negative or positive cases over ambiguous neutral cases. Thus, aligning with its social media optimization where neutral tones are rarer.

Figure 5.4.5.3.3 RoBERTa Confusion Matrix



The classification report highlights BERT's high precision of 88% and recall of 88% for negative sentiments and high precision of 94% and high recall of 80% for positive sentiments. Thus, reflecting reliable detection of polarized language. However, neutral sentiment performs poorly with a precision of 7% and a recall of 19%. Thus, underscoring its tendency to over-classify texts as negative or positive. At the same time, 82% overall accuracy and 85% of weighted F1-score demonstrate robust performance for imbalanced datasets, but 62% of macro F1-scores exposes weakness in neutral-class handling. This trade-off is possible due to scarce neutral sentiments.

Figure 5.4.5.3.4 RoBERTa Classification Report



```
Classification Report (Bert):
              precision    recall  f1-score   support

    negative       0.88      0.88      0.88       440
     neutral       0.07      0.19      0.10        31
    positive       0.94      0.80      0.86       382

    accuracy                           0.82       853
   macro avg       0.63      0.63      0.62       853
weighted avg       0.88      0.82      0.85       853
```

Based on the performance of the three models, the RoBERTa model have the best classification. Therefore, it's sentimental column is then used in LSTM model for modified altman-z score prediction.

In the code, the aggregate_sentiment_to_quarterly function converts daily/monthly sentiment data into quarterly aggregates aligned with financial reporting periods. Next, it maps categorical sentiment labels such as negative, neutral, positive to numerical values such as 0, 1, 2. Then, quarterly statistics such as mean sentiment, standard deviation, count of reviews, and ratios of positive/negative sentiments are computed. For periods with missing data, it defaults to neutral values. The output is a DataFrame structured for merging with financial data. But in this case, it won't be used to combined with finance data yet to observe how the forecasting model would fare with only sentimental data.

Next, the prepare_lstm_data_with_sentiment function merges this quarterly sentiment data with modified Altman Z-Scores and scales all features using StandardScaler. It handles cases where only sentiment features are used and creates time-series sequences for LSTM training by sliding a window of t look_back periods. The output includes scaled features (X), targets (y), scalers for inverse transformations, and aligned dates for plotting. The results are as follows:

Table 5.4.5.3.1 MAE and MSE of sentimental data used in LSTM model

Fold	Period Trained	Period Tested	MAE	MSE
1	31/12/2004 - 31/03/2012	30/06/2012 - 30/09/2014	0.831	0.736
2	31/12/2004 - 30/09/2014	31/12/2014 - 31/03/2017	0.670	0.485
3	31/12/2004 - 31/03/2017	30/06/2017 - 30/09/2019	0.769	1.013
4	31/12/2004 - 30/09/2019	31/12/2019 - 31/03/2022	4.045	18.538
5	31/12/2004 - 31/03/2022	30/06/2022 - 30/09/2024	4.656	21.753

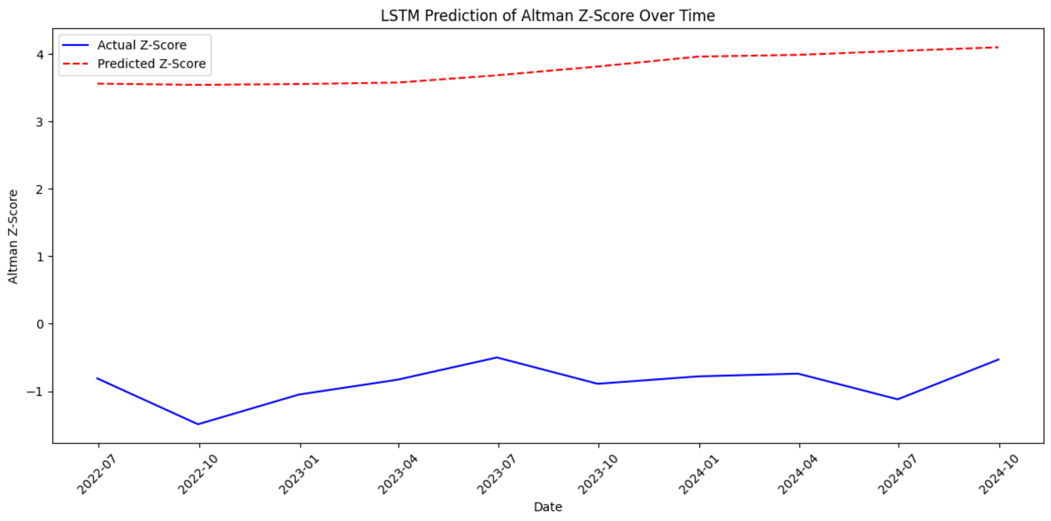


Figure 5.4.5.3.5 LSTM diagram of sentimental data used to forecast modified Altman Z-Score

In comparison to the forecasting result using only financial data, the LSTM model behaves worst on average. However, this model notably performs better the 3 fold with finance model having a MAE of 0.989 and MSE of 1.871 against the sentimental only model with a MAE of 0.769 and MSE of 1.013.

Table 5.4.5.3.2 Average MAE and MSE of sentimental data used in LSTM model

Average MAE	**Average MSE**
2.1942	8.5050

5.4.6 Combined Result Analysis

In the following section, finance data and sentimental data are both used to forecast Altman Z-score. Although, in this case the combined data had lowered the average MAE and Average MSE, it manages to reduce the MAE and MSE during the fourth fold which both models had perform terribly in. This hints that sentimental data had greater influence especially in times of crisis. This happens due to staff turnover, which might lead to lower quality service due to shorthandedness.

Table 5.4.6.1 MAE and MSE of finance and sentimental data used in LSTM model

Fold	Period Trained	Period Tested	MAE	MSE
1	31/12/2004 - 31/03/2012	30/06/2012 - 30/09/2014	0.384	0.181
2	31/12/2004 - 30/09/2014	31/12/2014 - 31/03/2017	0.524	0.352

3	31/12/2004 - 31/03/2017	30/06/2017 - 30/09/2019	0.544	0.623
4	31/12/2004 - 30/09/2019	31/12/2019 - 31/03/2022	3.108	11.741
5	31/12/2004 - 31/03/2022	30/06/2022 - 30/09/2024	1.069	2.039

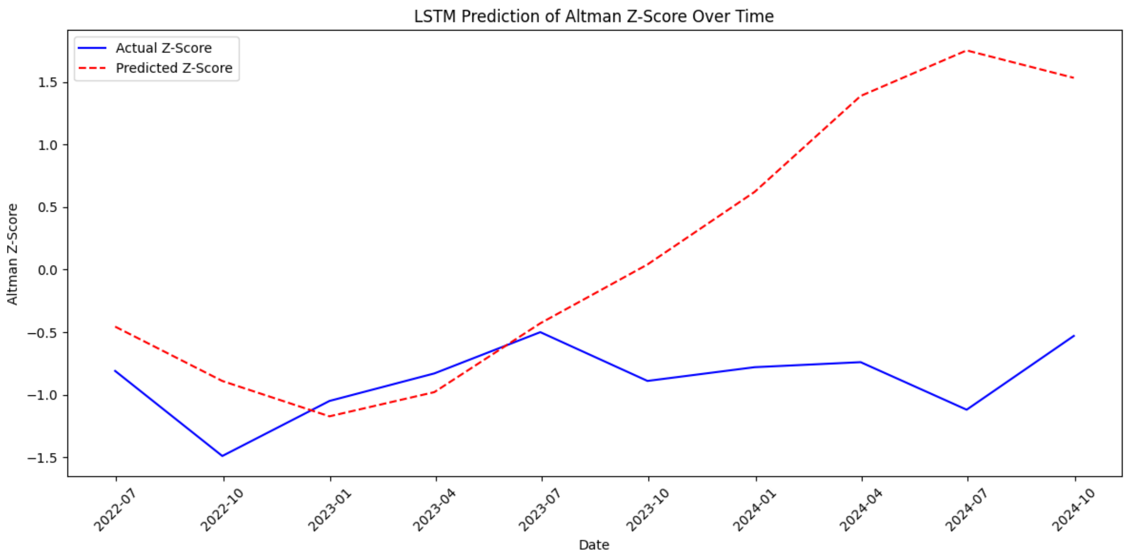


Figure 5.4.6.1 LSTM diagram of finance and sentimental data used to forecast modified Altman Z-Score

Table 5.4.6.2 Average MAE and MSE of finance and sentimental data used in LSTM model

Average MAE	Average MSE
1.1257	2.9872

5.5 Implementation of Issues and Challenges

There exist three types of challenges faced during the implementation. The first is known as technical challenges. This is because there exists environmental limitation because Google Colab's GPU runtime was used, but session timeouts required frequent checkpointing and rerunning when left for 40 minutes. This meant all the code had to be rerun again which waste time.

The second is known as modelling challenges. This is because there exists class imbalance in financial distress labels with most of them congregation in 2021 to 2024. This led to training at the fourth fold for the three models to experience a difficulty since it is the first time the model encounters a period where the modified Altman Z-score dropped below the 1.1 threshold. Though notably, the addition of sentimental data seems to help reduce the MAE score during this unexpected turmoil.

Besides, there exist relatively low sentimental data with neutral statements which causes it to be more easily misclassified. This is because most times people who left review are affected by intense emotions either good or bad which compel them to submit a review on the service provided.

The fourth challenge faced is known as the integration challenge. This is because using numerical financial data with categorical sentiment scores required custom feature engineering. This is because financial data are categorized quarterly while sentimental scores sometimes have a few more in a single day or sometimes none. This means sentimental data needs to be aggregated to quarterly before fusing with finance data to predict the modified Altman Z-score.

5.6 Concluding Remark

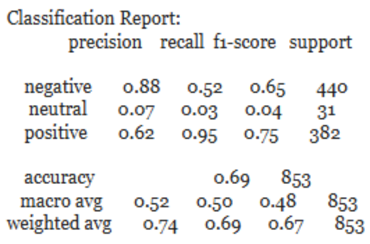
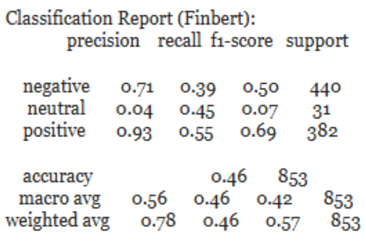
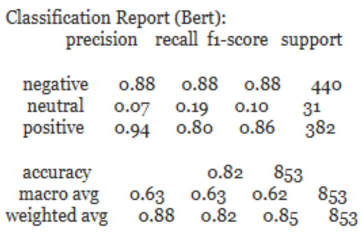
In terms of Time Series Analysis, The LSTM model using only financial data demonstrates that incorporating the top 12 important features yields better forecasting performance than using only the top 5 features, as shown in Table 5.6.1. This is probably because the greater versatility and robustness of the top 12 important features in predicting the modified Altman Z-score, whereas relying on only the top 5 important features may limit the model's predictive capacity.

Table 5.6.1 Comparison between Average MAE and MSE of top 12 Features and top 5 Features used in LSTM model

	Average MAE	**Average MSE**
Using the top 12 important features	1.0367	3.0663
Using the top 5 important features	1.4076	4.1419

In terms of Sentiment Analysis, BERT outperforms TextBlob and FinBERT in classifying customer reviews due to its pretraining on conversational data such as Twitter. In fact, FinBERT, while specializing in financial texts, is better suited for institutional reports or earnings commentary rather than direct customer feedback.

Table 5.6.2 Comparison of Sentiment Analysis Models

Text Blob	**FinBert**	**BERT**
		
• Basic sentiment interpretation without contextual depth	• More negative bias (financial reports often mask turmoil with positive language)	• Best for customer reviews (trained on social media data)

```
Classification Report:
              precision    recall  f1-score   support

    negative       0.88      0.52      0.65       440
     neutral       0.07      0.03      0.04        31
    positive       0.62      0.95      0.75       382

    accuracy                           0.69       853
   macro avg       0.52      0.50      0.48       853
weighted avg       0.74      0.69      0.67       853
```

```
Classification Report (Finbert):
              precision    recall  f1-score   support

    negative       0.71      0.39      0.50       440
     neutral       0.04      0.45      0.07        31
    positive       0.93      0.55      0.69       382

    accuracy                           0.46       853
   macro avg       0.56      0.46      0.42       853
weighted avg       0.78      0.46      0.57       853
```

```
Classification Report (Bert):
              precision    recall  f1-score   support

    negative       0.88      0.88      0.88       440
     neutral       0.07      0.19      0.10        31
    positive       0.94      0.80      0.86       382

    accuracy                           0.82       853
   macro avg       0.63      0.63      0.62       853
weighted avg       0.88      0.82      0.85       853
```

When integrating sentiment scores into the LSTM model using BERT model, it is noted that text data alone performs poorly because customer reviews provide only a single perspective. For instance, like service quality. Therefore, a more holistic approach such as incorporating internal reports, shareholder opinions, or news analysis could probably improve accuracy.

On average, the LSTM model using only financial data achieves the best results, This is followed by the hybrid (finance + text) model, while the text-only model performs worst as seen in Table 5.6.3.

Table 5.6.3 Performance of LSTM Models with Different Inputs

Input Data	**Avg. MAE**	**Avg. MSE**
Finance only	1.0367	3.0663
Text only	2.1942	18.5050
Finance + Text (Hybrid)	1.1257	2.9872

However, during periods of financial turmoil such as the COVID-19 pandemic, the hybrid model excels. This is seen in the 4th fold (test period: 2020–2022), the hybrid model achieved lower errors with the score of MAE being 3.108 and score of MSE being 11.741 compared to finance-only model with the score of MAE being 3.761 and the score of MSE being 16.270 and text-only model with the score of MAE being 4.045 and score of MSE being 18.538 as seen in Table 5.6.5. This suggests that customer sentiment becomes more predictive during crises, as seen with airline companies experiencing service declines and increased complaints prior to financial deterioration.

Table 5.6.4 Model Performance During Crisis (4th Fold)

Input Data	**MAE**	**MSE**
Finance only	3.761	16.270
Text only	4.045	18.538
Finance + Text (Hybrid)	3.108	11.741

CHAPTER 6

System Evaluation and Discussion

6.1 System Testing and Performance Metrics

To evaluate the model's robustness, extensive testing using time-series cross-validation (5 folds) are conducted to ensure the model generalizes across different economic conditions. Therefore, the key performance metrics included:

- Mean Absolute Error (MAE) and Mean Squared Error (MSE) for regression accuracy.
- Feature importance analysis via decision trees to identify the most predictive financial indicators
- Pre post imputation analysis by using the R square, median, mean and square error on the imputed data.
- Sentiment score evaluation using classification report and confusion matrix by comparing the score against the review ratings.

6.2 Testing Setup and Result

For the testing setup, the data is split into five folds. With the 4th fold (2019–2022) including the pandemic-driven distress phase, while earlier folds represented stable periods.

Based on the hybrid model's superior crisis performance in the 4th fold (2019-2022) with its MAE score of 3.108, it suggests firms could monitor customer sentiment to anticipate downturns. This finding is affirmed by Capital A Berhad's experience during 2020. This is because negative sentiment surged due to flight cancellations, refund delays, and poor customer service [46] [47]. Thus, testing brand loyalty. Consequently, these service declines directly correlated with the company's workforce reductions, which included laying off 300+ employees, including 111 cabin crew, 172 pilots, and 50 engineers [48] and implementing 15-75% salary cuts for remaining staff as part of COVID-19 restructuring measures [47]. Unfortunately, the negative sentiment persisted through 2021. During this period, the negative sentiments are primarily focused on digital service quality and ticketing complaints. Fortunately, by 2022, sentiment began improving as the company improve its digital travel platform and gradually resumed normal operations [49]. Therefore, this case demonstrates how workforce disruptions and operational stress during crises can amplify customer dissatisfaction such as when managing service failures like pandemic-related refunds and cancellations. Hence, it highlights the importance of sentimental analysis in financial distress forecasting as it directly impacts the end stakeholders, which in this case are the customers.

Besides, the performance of the top 12 important features from finance data imply the company should prioritize broader financial metrics over a handful of key indicators for distress forecasting results. This is because the latter may not be as robust during financial crisis. For instance, the top 5 important features failed to include critical post-pandemic metrics such as teleport_(logistics)_non_airline_ebitda, which is a newly introduced measure reflecting Capital A Berhad's diversification efforts beyond aviation, and sectors_flown/no._of_flights, which can be used to track the recovery of its airline operations.

Lastly, the result also hinted at the limitations of using only one type of data for decision-making. For instance, as seen in the result at Table 5.6.3, text-only models are unreliable for standalone use but may supplement traditional analysis during volatility. However, this accounts for using only customer reviews data textual data. In fact, studies, such as those by Huang et al. [52] and Gupta et al. [53], emphasize the predictive power of multi-source textual data, including Management Discussion & Analysis (MD&A) sections and audit reports. Furthermore, Huang et al [52]. found that sentiment extracted from audit reports provided more significant incremental improvements in forecasting financial distress than MD&A sections. This is because auditors' conservative tones often reveal hidden risks that optimistic management statements may obscure which make it recommendable to add audit reports as part of the text data for future studies. Similarly, Gupta et al. [53] demonstrated that failed banks exhibited higher positive sentiment in their annual reports. Thus, suggesting potential concealment of financial distress which justify the use of FinBERT to detect early distress signal from text data of annual report, a nuance that customer reviews alone cannot capture. Hence, relying solely on customer reviews overlooks critical insights from structured corporate disclosures for a comprehensive risk assessment.

Therefore, it is vital to integrate multi-source textual data, such as audit reports and MD&A sections, with traditional financial metrics to enhance predictive accuracy, particularly during periods of economic uncertainty or regulatory changes. At the same time, although finance-only data performs better alone but the result from the 4th fold showed it's struggles to forecast during a crisis or when there's a change in the standards of the financial report. This was evident in 2019 when the modified Altman Z-score misclassified Capital A Berhad's financial health due to its adoption of **MFRS 16** (Malaysian Financial Reporting Standard 16). This is because the new standard required operating leases, including aircraft leases, to be recorded on the balance sheet under the **Right-of-Use (ROU)** approach [50]. Therefore, this significantly inflates the reported borrowings and gearing ratios. Hence, this accounting shift increased debt visibility and distorted financial ratios. Thus, affecting investor perception [51].

6.3 Project Challenges

Due to time constraints, the project face challenges such as poor model performance. This could be further break down as subsection as seen below:

Firstly, there exist an over-reliance on Decision Trees for feature selection. This is because the current pipeline uses decision trees exclusively. This may overlook nonlinear relationships captured by other methods. Therefore, to improve feature selection, an ensemble feature selection could be implemented in the future.

Secondly, the LSTM model in general have a poor generalization to distress periods. This is seen when the model struggled in the 4th fold which is also known as the pandemic phase, with finance-only model's MAE spiking to 3.761. This suggests insufficient crisis-era training data or overfitting to stable periods.

Thirdly, the LSTM architecture is suboptimal. This is because the LSTM was not hyperparameter-tuned (e.g., layer depth, dropout rates). Thus, limiting its capacity. In the future, grid search for optimal layers and activation functions can be implemented to optimize the performance of the model.

Lastly, there is limited sentiment data which consist of customer-only perspective. Therefore, the text data with only customer reviews lack insights from stakeholders, employees, or market news. Thus, skewing sentiment analysis. In the future, to improve project challenges, text sources needed to be expanded. For instance, earnings call transcripts, internal and external audit reports and finance news articles.

6.4 Objectives Evaluation

Objective 1: Data Scraping and Data Cleaning

This objective had been fully accomplished in FYP1 through Phase 2 (Data Understanding) and Phase 3 (Data Preparation) of CRISP-DM. For instance, the finance data such as Quarterly statements (income, balance sheet, cash flow) from Capital A Berhad (2004–2024) were extracted from KLSE Screener and converted to CSV. The same goes to the market index data which was sourced from Bursa Malaysia and the text data gathered from customer reviews at Skytrax which were scraped using BeautifulSoup to support sentiment analysis. The 14 missing values from the finance data were addressed using KNN imputation FCMI imputation. In the end with FCMI selected as the superior method post-evaluation using R-squared, median, mean comparisons. The text data are also cleaned from special characters, converted to lower case and normalized and scaled for time series compatibility with the finance data. Hence, this lead to a clean, structured dataset of 306 financial variables and sentiment-labeled text. Thus, enabling robust modeling.

Objective 2: Employ time series analysis and sentiment analysis model for financial distress prediction

This objective was achieved in FYP1 and addressed in Phase 4 Modeling while being further furnished in FYP2. In terms of time series analysis with finance-only LSTM, the important features are then selected via decision tree. Then the LSTM model is used to find the best performance between the top 5 or top 12 financial features. The result is the latter with the lowest average errors (MAE: 1.0367, MSE: 3.0663). Thus, validating financial data's predictive strength. In terms of sentiment analysis, BERT outperformed FinBERT/TextBlob for customer reviews due to its highest accuracy, recall and precision. However, the model with only text data suffers poorly on average in terms of performance. Notably, combined sentiment and financial data in LSTM, shows crisis-period superiority with a MAE of 3.108 in comparison to MAE of. finance-only's model of 3.761. Hence, this fulfilled shows that financial data is primarily for stable-period forecasting whereas sentiment help augment accuracy during turmoil such as the pandemic in this case.

Objective 3: Compare the model performance used for the time series and sentiment analysis

This objective is achieved in FYP2 and addressed in Phase 5 Evaluation using cross validation and metrics. For instance, performance metrics such as MAE and MSE are used to determine the best performing model with finance-only LSTM performing best with an average of MAE 1.0367 and hybrid LSTM showing crisis resilience with an average of MAE 1.1267.

Objective 4: Interpret business implications of results

This objective is achieved in FYP2 and addressed in Phase 5 Deployment. The results provide insights that sentiment trends such as complaint spikes may precede financial decline. Thus, urging firms to monitor reviews during times of volatility. Besides, it also shows the importance of feature prioritization with Top 12 important financial features superseding top 5 important financial features in terms of performance. Thus, suggesting broader metric inclusion in risk assessments. Additionally, the result also urges for expanding sentiment sources such as adding employee and stakeholder feedback to mitigate customer-review bias. Hence, this fulfilled objective demonstrates that the company should integrate sentiment data into quarterly risk audits.

Table 6.4.1 Objective Fulfillment

Objective	**CRISP-DM Phase**	**Key Outcome**
Data Scraping/Cleaning	2–3	Cleaned financial & text datasets
Model Employment	4	Hybrid LSTM validated for crisis forecasting while finance-only for stability.
Performance Comparison	5	Quantified tradeoffs between model types.

Business Implications	6	Crisis detection protocols & multi-source sentiment recommendations.

6.5 Concluding Remark

In short, this study successfully achieved its four key objectives by following the CRISP-DM methodology. Firstly, comprehensive data scraping and cleaning produced a robust dataset of financial and sentiment metrics. Next, the time series and sentiment analysis models were effectively employed, with the hybrid model proving particularly valuable during crises. Thirdly, the model comparisons revealed the dominance of financial data for stable periods and the supplemental role of sentiment in turmoil. Finally, actionable business insights were derived, emphasizing the need for multi-source sentiment integration. Therefore, the results validate the hybrid approach's potential for financial distress prediction while highlighting room for improvement. For instance, the optimisation of LSTM architectures and the need for sentiment data diversity expansion.

CHAPTER 7

Conclusion and Recommendation

7.1 Conclusion

In conclusion, this study successfully integrates financial and sentimental data for financial distress prediction. Thus, demonstrating that although financial data alone is most reliable for financial distress forecasting, but sentiment data adds value especially in times of crisis. Besides, this study also helps identify RoBERTa, a type of BERT model as an optimal sentiment model for customer reviews.

Although the study successfully achieved its objective and provided further insights, there still exists room for improvement for this study. Notably, the sentiment data used in this study is limited by its single perspective scope. Furthermore, the model only uses a decision tree-based feature which may overlook non-linear relationships, and the model also struggles during unprecedented crisis as seen in their performance during the 4th and 5th fold.

7.2 Recommendation

In future studies, model improvement needs to be considered to further enhance the performance. This can be done by introducing naïve bayes as a comparison to the features selected by decision tree. Besides, LSTM architecture can also be optimised by conducting hyperparameter tuning. Secondly, data expansion should also be considered. By incorporating multi-source sentiment data, this can help enrich the model and balance perspective bias. Additionally, explainable AI can also be integrated to interpret the hybrid model decisions for stakeholders

REFERENCES

[1] *About us* [Online]. AirAsia Newsroom. Available: https://newsroom.airasia.com/about-us

[2] D. M. H. Kee et al., "The struggles and the survival of AirAsia during the COVID-19 pandemic," *Adv. Glob. Econ. Bus. J.*, vol. 2, no. 1, pp. 1–12, 2021. [Online]. Available: http://agebj.org/index.php/agebj/article/view/23

[3] C. Carongoy, "AirAsia Philippines reaches pre-pandemic load factor," *AirAsia Newsroom*, Jul. 18, 2023. [Online]. Available: https://newsroom.airasia.com/news/2023/7/18/airasia-philippines-reaches-pre-pandemic-load-factor

[4] A. Aziz and J. Lai, "Diving deep into RM6.8 bil merger of AirAsia X, Capital A's aviation assets," *The Edge Malaysia*, Oct. 8, 2024. [Online]. Available: https://theedgemalaysia.com/node/728601

[5] R. Rahardjoputri, R. Y. Putra, and H. Agustanto, "AirAsia Group Berhad strengthens SuperApp during the COVID-19 pandemic," *Sebelas Maret Bus. Rev.*, vol. 6, no. 1, p. 26, 2021. [Online]. Available: https://doi.org/10.20961/smbr.v6i1.55705

[6] S. W. St, "Capital A Berhad's (KLSE:CAPITALA) 29% price boost is out of tune with revenues," *Simply Wall St News*, Sep. 30, 2024. [Online]. Available: https://simplywall.st/stocks/my/transportation/klse-capitala/capital-a-berhad-shares/news/capital-a-berhads-klsecapitala-29-price-boost-is-out-of-tune

[7] D. C. Hutagaol and R. Erdiansyah, "The effect of service quality, price, customer satisfaction on customer loyalty of AirAsia customers," in *Proc. Tarumanagara Int. Conf. Appl. Soc. Sci. Humanit.*, 2020. [Online]. Available: https://doi.org/10.2991/assehr.k.200515.063

[8] M. Gurung, "Altman Z-Score: Definition, calculation, key ratios, interpretation & more," *Investing.com*. [Online]. Available: https://www.investing.com/academy/analysis/altman-z-score-definition/

[9] *What is PN17? Do you own shares in these PN17 companies?* [Online]. kopiandproperty.com, Mar. 8, 2022. Available: https://kopiandproperty.com/what-is-pn17-do-you-own-shares-in-these-pn17-companies/

[10] *Capital A | Bursa Filings*. [Online]. Available: https://www.capitala.com/news.html/id/2337359

[11] K. Albeladi, B. Zafar, and A. Mueen, "Time series forecasting using LSTM and ARIMA," *Int. J. Adv. Comput. Sci. Appl.*, vol. 14, no. 1, 2023. [Online]. Available: https://doi.org/10.14569/ijacsa.2023.0140133

[12] I. Ghosh and P. Dragan, "Can financial stress be anticipated and explained? Uncovering the hidden pattern using EEMD-LSTM, EEMD-prophet, and XAI methodologies," *Complex Intell. Syst.*, vol. 9, no. 4, pp. 4169–4193, 2022. [Online]. Available: https://doi.org/10.1007/s40747-022-00947-8

[13] K. Kashif and R. Ślepaczuk, "LSTM-ARIMA as a hybrid approach in algorithmic investment strategies," *Working Papers*, 2024. [Online]. Available: https://doi.org/10.33138/2957-0506.2024.7.443

[14] H. Wu, S. Chen, and Y. Ding, "Comparison of ARIMA and LSTM for stock price prediction," *Financ. Eng. Risk Manag.*, vol. 6, no. 1, 2023. [Online]. Available: https://doi.org/10.23977/ferm.2023.060101

[15] K. Lahboub and M. Benali, "Assessing the predictive power of transformers, ARIMA, and LSTM in forecasting stock prices of Moroccan credit companies," *J. Risk Financ. Manag.*, vol. 17, no. 7, p. 293, 2024. [Online]. Available: https://doi.org/10.3390/jrfm17070293

[16] P. Shah, "Sentiment analysis using TextBlob," *Towards Data Science*, Dec. 15, 2021. [Online]. Available: https://towardsdatascience.com/my-absolute-go-to-for-sentiment-analysis-textblob-3ac3a11d524

[17] *Explanation of BERT Model NLP*, GeeksforGeeks, Jan. 10, 2024. [Online]. Available: https://www.geeksforgeeks.org/explanation-of-bert-model-nlp/

[18] S. R and P. K. Aithal, "Millennial’s customer impact of actual purchase to loyalty intention through customer satisfaction and customer engagement on Instagram," *Financ. Eng.*, vol. 2, pp. 162–170, 2024. [Online]. Available: https://doi.org/10.37394/232032.2024.2.15

[19] E. Christodoulaki, M. Kampouridis, and P. Kanellopoulos, "Technical and sentiment analysis in financial forecasting with genetic programming," in *2022 IEEE Symp. Comput. Intell. Financ. Eng. Econ. (CIFEr)*, Helsinki, Finland, 2022, pp. 1–8. [Online]. Available: https://doi.org/10.1109/CIFEr52523.2022.9776186

[20] M. Hashim, K. Muhammad, E. K. Ghani, and Maz, “Financial Distress Analysis of Top 100 Malaysian Public Listed Companies during COVID-19 Pandemic using Altman Z-Score Analysis,” *International journal of economics and financial issues*, vol. 14, no. 4, pp. 200–205, Jul. 2024, doi: https://doi.org/10.32479/ijefi.16545.

[21] CFI Team, “Working Capital Formula,” *Corporate Finance Institute*, Apr. 07, 2020. https://corporatefinanceinstitute.com/resources/financial-modeling/working-capital-formula/(accessed Apr. 25, 2025).

[22] J. Chen, “Market Value Of Equity Definition,” *Investopedia*, Mar. 11, 2020. https://www.investopedia.com/terms/m/market-value-of-equity.asp (accessed Apr. 25, 2025).

[23] “Capital A | Bursa Filings,” *Capitala.com*, 2022. https://www.capitala.com/news.html/id/2368958 (accessed Apr. 25, 2025).

[24] “Capital A | Bursa Filings,” *www.capitala.com*. https://www.capitala.com/news.html/id/2337359 (accessed Apr. 25, 2025).

[25] G. D. Hou, D. L. Tong, S. Y. Liew, and P. Y. Choo, “Improving financial distress prediction using machine learning: A preliminary study,” *ITM Web of Conferences*, vol. 67, p. 01050, 2024, doi: https://doi.org/10.1051/itmconf/20246701050.

[26] J. Fernando, “Current ratio explained with formula and examples,” *Investopedia*, Aug. 16, 2024. https://www.investopedia.com/terms/c/currentratio.asp

[27] A. Hayes, “Operating Margin: What It Is and the Formula for Calculating It, With Examples,” *Investopedia*, Apr. 28, 2024.

https://www.investopedia.com/terms/o/operatingmargin.asp
[28] R. Villaester, “Operating Profit Margin: Definition, Formula and Calculation,” *Wise*, Dec. 10, 2021. https://wise.com/gb/blog/operating-profit-margin

[29] BDC, “Financial ratios: 4 ways to assess your business,” *BDC.ca*, 2020. https://www.bdc.ca/en/articles-tools/money-finance/manage-finances/financial-ratios-4-ways-assess-business

[30] J. Fernando, “Earnings Per Share (EPS): What It Means and How to Calculate It,” *Investopedia*, Jan. 24, 2025. https://www.investopedia.com/terms/e/eps.asp

[31] A. Hayes, “What Is the Asset Turnover Ratio?,” *Investopedia*, Jul. 16, 2024. https://www.investopedia.com/terms/a/assetturnover.asp

[32] Indeed Editorial Team, “How to Calculate and Interpret Your Debt to Asset Ratio,” *Indeed Career Guide*, Jan. 31, 2023. https://www.indeed.com/career-advice/career-development/how-to-calculate-debt-to-asset-ratio

[33] J. Fernando, “Debt-to-Equity (D/E) Ratio Formula and How to Interpret It,” *Investopedia*, Mar. 06, 2024. https://www.investopedia.com/terms/d/debtequityratio.asp

[34] “AirAsia Group Berhad Fourth Quarter and Full Year 2019 Financial Results,” airasia newsroom. https://newsroom.airasia.com/news/airasia-group-berhad-fourth-quarter-full-year-2019-financial-results#gsc.tab=0

[35] “PRESS RELEASE AIRASIA GROUP BERHAD FOURTH QUARTER & FULL YEAR 2019 FINANCIAL RESULTS ROBUST BUSINESS MODEL, EXPERIENCED IN DEALING WITH AVIATION SHOCKS.” Accessed: May 07, 2025. [Online]. Available: https://www.capitala.com/misc/press-release-4Q19_latest.pdf

[36] “MFRS 16 effect keeps AirAsia X in the red in 3Q,” The Edge Malaysia, 2025. https://theedgemalaysia.com/article/mfrs-16-effect-keeps-airasia-x-red-3q-0 (accessed May 07, 2025).

[37] Annual Report 2020, “Annual Report 2020,” Capitala.com, 2020. https://www.capitala.com/misc/FlippingBook/ar2020/56/ (accessed May 07, 2025).

[38] GeeksForGeeks, “Python | Decision Tree Regression using sklearn,” GeeksforGeeks, Oct. 04, 2018. https://www.geeksforgeeks.org/python-decision-tree-regression-using-sklearn/

[39] GeeksforGeeks, “What is LSTM Long Short Term Memory?,” GeeksforGeeks, Jan. 16, 2019. https://www.geeksforgeeks.org/deep-learning-introduction-to-long-short-term-memory

[40] P. Shah, “Sentiment Analysis using TextBlob - TDS Archive - Medium,” Medium, Jun. 27, 2020. https://medium.com/data-science/my-absolute-go-to-for-sentiment-analysis-textblob-3ac3a11d524

[41] S. Kumar and R. Chaturvedi, “Evaluating The Efficacy Of Distilled Transformer Models For Sentiment Analysis In Financial Texts: A Comparative Study,” International Journal of Current Science, vol. 14, no. 2, pp. 2250–1770, 2024, Available: https://www.rjpn.org/ijcspub/papers/IJCSP24B1335.pdf

[42] R. Shinde, “Financial News Sentiment Analysis using FinBERT,” Medium, Apr. 02, 2022. https://medium.com/@ravirajshinde2000/financial-news-sentiment-analysis-using-finbert-25afcc95e65f

[43] A. Huang, H. Wang, and Y. Yang, “FinBERT—A Deep Learning Approach to Extracting Textual Information,” papers.ssrn.com, Jul. 28, 2020. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3910214

[44] https://www.facebook.com/manendra.verma.10, “RoBERTa NLP Model Explained: A Comprehensive Overview -,” quickread, Sep. 11, 2023. https://www.quickread.in/roberta-nlp-model-explained-a-comprehensive-guide/

[45] P. Mishra, K. D. Mani, P. Johri, and D. Arya, “FCMI: Feature Correlation based Missing Data Imputation,” arXiv.org, Jun. 26, 2021. https://arxiv.org/abs/2107.00100 (accessed May 01, 2024).

[46] U. Daniele, “AirAsia faces backlash over delayed pandemic refunds,” Aljazeera, May 23, 2022. https://www.aljazeera.com/economy/2022/5/23/airasia-faces-customer-backlash-over-delayed-pandemic-refunds

[47] D. Mui, Y. R. Kumar, Vanishree Chadaran, S. Karunakaran, Swathy Sugumaran, and Tinisma Binti Talha, “The Struggles and The Survival of Airasia during The Covid-19 Pandemic,” vol. 2, no. 1, pp. 1–12, Jun. 2021.

[48] J. Choong, “Report: AirAsia to Lay off 250 Staff Members following Covid-19 Downturn | Malay Mail,” www.malaymail.com, Jun. 04, 2020. https://www.malaymail.com/news/malaysia/2020/06/04/report-airasia-to-lay-off-250-staff-members-following-covid-19-downturn/1872493

[49] N. H. Md Saad, C. W. San, and Z. Yaacob, “Twitter Sentiment Analysis of the Low-Cost Airline Services After COVID-19 Outbreak: The Case of AirAsia,” Business Systems Research : International journal of the Society for Advancing Innovation and Research in Economy, vol. 14, no. 2, pp. 1–23, Dec. 2023, doi: https://doi.org/10.2478/bsrj-2023-0009.

[50] Annual Report 2019, “Annual Report 2019,” Capitala.com, 2019. https://www.capitala.com/misc/FlippingBook/ar2019/231/ (accessed May 07, 2025).

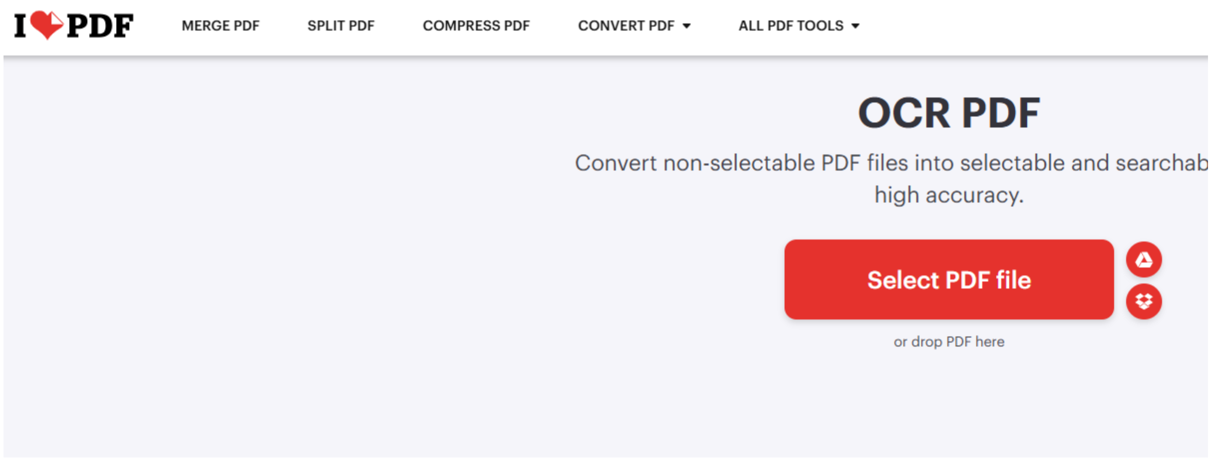
[51] “Uncertainty for corporate earnings as MFRS16 takes effect,” The Edge Malaysia, 2025. https://theedgemalaysia.com/article/uncertainty-corporate-earnings-mfrs16-takes-effect (accessed May 07, 2025).

[52] B. Huang, X. Yao, Y. Luo, and J. Li, “Improving financial distress prediction using textual sentiment of annual reports,” *Annals of Operations Research*, Mar. 2022, doi: https://doi.org/10.1007/s10479-022-04633-3.

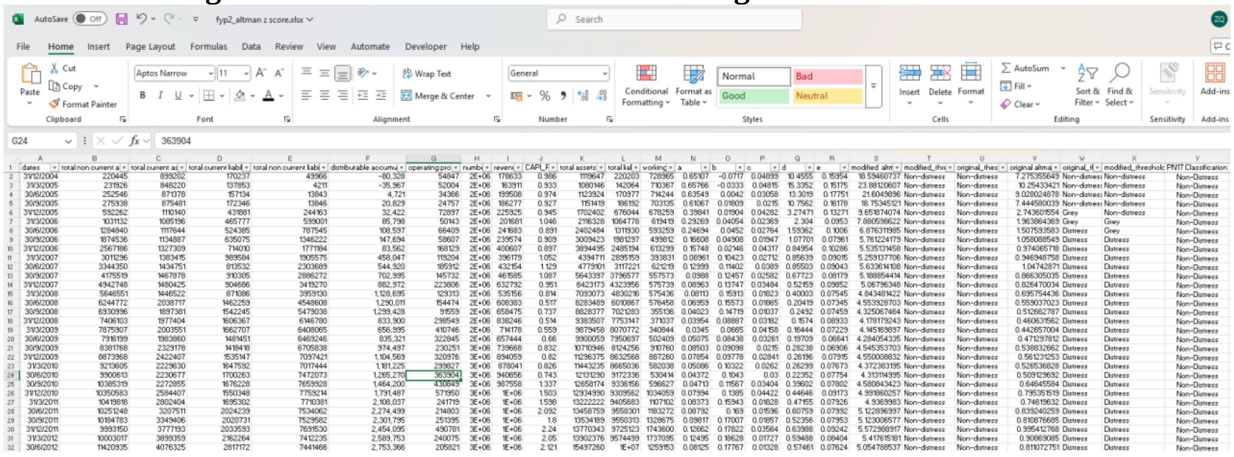
[53] A. Gupta, Majeed Simaan, and M. J. Zaki, “Investigating bank failures using text mining,” *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–8, Dec. 2016, doi: https://doi.org/10.1109/ssci.2016.7850006.

APPENDIX

# OCR website to extract text from pdf file



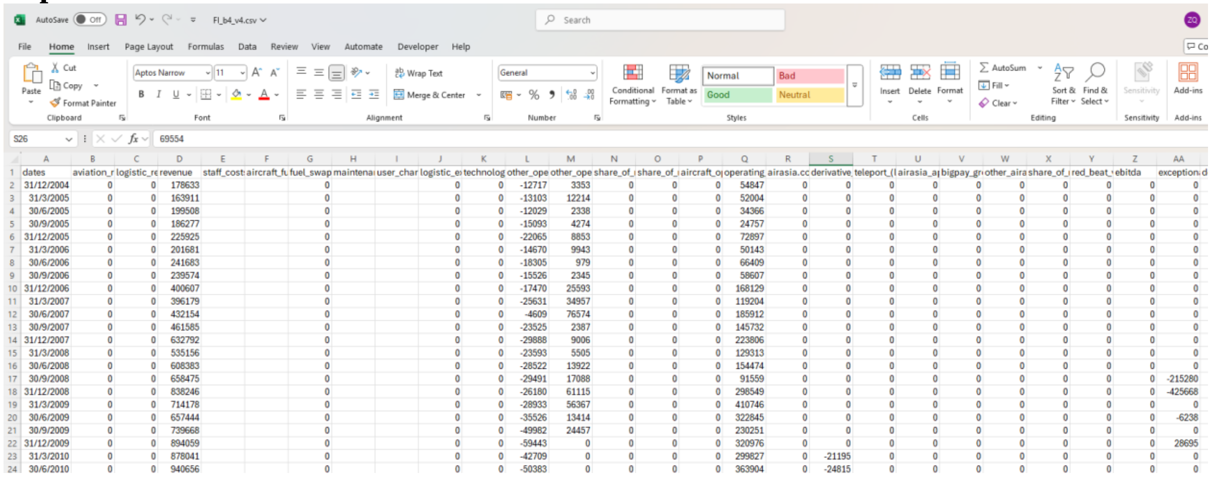
# Calculate original and modified altman z-score using Microsoft excel



G24 363904

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
1	dates	total non current a	total current as	total current liabi	total non current liabi	distributable accumu	operating pro	numbe	revenu	CAPI_F	total assets	total lial	working	a	b	c	d	e	modified altm	modified_thre	original_thes	original altma	original_tl	modified_threshold	PN17 Classification
2	31/12/2004	220445	899202	170237	49966	-80,328	54847	2E+06	178633	0.986	1119647	220203	728965	0.65107	-0.0717	0.04899	10.4555	0.15954	18.59460737	Non-distress	Non-distress	7.275355649	Non-distres	Non-distress	Non-Distress
3	31/3/2005	231926	848220	137853	4211	-35,967	52004	2E+06	163911	0.933	1080146	142064	710367	0.65766	-0.0333	0.04815	15.3352	0.15175	23.88120607	Non-distress	Non-distress	10.25433421	Non-distres	Non-distress	Non-Distress
4	30/6/2005	252546	871378	157134	13843	4,721	34366	2E+06	199508	0.974	1123924	170977	714244	0.63549	0.0042	0.03058	13.3019	0.17751	21.6049896	Non-distress	Non-distress	9.028024878	Non-distres	Non-distress	Non-Distress
5	30/9/2005	275938	875481	172346	13846	20,829	24757	2E+06	186277	0.927	1151419	186192	703135	0.61067	0.01809	0.0215	10.7562	0.16178	18.75345121	Non-distress	Non-distress	7.444580039	Non-distres	Non-distress	Non-Distress
6	31/12/2005	592262	1110140	431881	244163	32,422	72897	2E+06	225925	0.945	1702402	676044	678259	0.39841	0.01904	0.04282	3.27471	0.13271	9.651874074	Non-distress	Non-distress	2.743601554	Grey	Non-distress	Non-Distress
7	31/3/2006	1031132	1085196	465777	599001	85,798	50143	2E+06	201681	1.046	2116328	1064778	619419	0.29269	0.04054	0.02369	2.304	0.0953	7.880598622	Non-distress	Non-distress	1.963864369	Grey	Grey	Non-Distress
8	30/6/2006	1284840	1117644	524385	787545	108,597	66409	2E+06	241683	0.891	2402484	1311930	593259	0.24694	0.0452	0.02764	1.59362	0.1006	6.876311985	Non-distress	Non-distress	1.507593583	Distress	Grey	Non-Distress
9	30/9/2006	1874536	1134887	635075	1346222	147,694	58607	2E+06	239574	0.909	3009423	1981297	499812	0.16608	0.04908	0.01947	1.07701	0.07961	5.761224179	Non-distress	Non-distress	1.058088549	Distress	Distress	Non-Distress
10	31/12/2006	2567186	1327309	714010	1771184	83,562	168129	2E+06	400607	0.897	3894495	2485194	613299	0.15748	0.02146	0.04317	0.84954	0.10286	5.535131458	Non-distress	Non-distress	0.974065718	Distress	Distress	Non-Distress
11	31/3/2007	3011296	1383415	989584	1905575	458,047	119204	2E+06	396179	1.052	4394711	2895159	393831	0.08961	0.10423	0.02712	0.85639	0.09015	5.259137706	Non-distress	Non-distress	0.946948758	Distress	Distress	Non-Distress
12	30/6/2007	3344350	1434751	813532	2303689	544,920	185912	2E+06	432154	1.129	4779101	3117221	621219	0.12999	0.11402	0.0389	0.85503	0.09043	5.633614108	Non-distress	Non-distress	1.04742871	Distress	Distress	Non-Distress
13	30/9/2007	4175519	1467878	910305	2886272	702,995	145732	2E+06	461585	1.087	5643397	3796577	557573	0.0988	0.12457	0.02582	0.67723	0.08179	5.188854414	Non-distress	Non-distress	0.866305035	Distress	Distress	Non-Distress
14	31/12/2007	4942748	1480425	904686	3419270	882,972	223806	2E+06	632792	0.951	6423173	4323956	575739	0.08963	0.13747	0.03484	0.52159	0.09852	5.06796348	Non-distress	Non-distress	0.826470034	Distress	Distress	Non-Distress
15	31/3/2008	5646551	1446522	871086	3959130	1,128,695	129313	2E+06	535156	0.814	7093073	4830216	575436	0.08113	0.15913	0.01823	0.40003	0.07545	4.843481422	Non-distress	Non-distress	0.695754436	Distress	Distress	Non-Distress
16	30/6/2008	6244772	2038717	1462259	4548608	1,290,011	154474	2E+06	608383	0.517	8283489	6010867	576458	0.06959	0.15573	0.01865	0.20419	0.07345	4.553928703	Non-distress	Non-distress	0.559037023	Distress	Distress	Non-Distress
17	30/9/2008	6930996	1897381	1542245	5479038	1,299,428	91559	2E+06	658475	0.737	8828377	7021283	355136	0.04023	0.14719	0.01037	0.2492	0.07459	4.325067464	Non-distress	Non-distress	0.512662787	Distress	Distress	Non-Distress
18	31/12/2008	7406103	1977404	1606367	6146780	833,900	298549	2E+06	838246	0.514	9383507	7753147	371037	0.03954	0.08887	0.03182	0.1574	0.08933	4.178179243	Non-distress	Non-distress	0.460631562	Distress	Distress	Non-Distress
19	31/3/2009	7875907	2003551	1662707	6408065	656,995	410746	2E+06	714178	0.559	9879458	8070772	340844	0.0345	0.0665	0.04158	0.16444	0.07229	4.145169897	Non-distress	Non-distress	0.442657004	Distress	Distress	Non-Distress
20	30/6/2009	7916199	1983860	1481451	6469246	835,321	322845	2E+06	657444	0.66	9900059	7950697	502409	0.05075	0.08438	0.03261	0.19709	0.06641	4.284054335	Non-distress	Non-distress	0.471297812	Distress	Distress	Non-Distress
21	30/9/2009	8381768	2329178	1418418	6705838	974,497	230251	3E+06	739668	0.832	10710946	8124256	910760	0.08503	0.09098	0.0215	0.28238	0.06906	4.545353703	Non-distress	Non-distress	0.538832662	Distress	Distress	Non-Distress
22	31/12/2009	8873968	2422407	1535147	7097421	1,104,569	320976	3E+06	894059	0.82	11296375	8632568	887260	0.07854	0.09778	0.02841	0.26196	0.07915	4.550008832	Non-distress	Non-distress	0.561231253	Distress	Distress	Non-Distress
23	31/3/2010	9213605	2229630	1647592	7017444	1,181,225	299827	3E+06	878041	0.826	11443235	8665036	582038	0.05086	0.10322	0.0262	0.26299	0.07673	4.372383195	Non-distress	Non-distress	0.526536828	Distress	Distress	Non-Distress
24	30/6/2010	9900613	2230677	1700263	7472073	1,265,270	363904	3E+06	940656	0.743	12131290	9172336	530414	0.04372	0.1043	0.03	0.22352	0.07754	4.313114995	Non-distress	Non-distress	0.509129692	Distress	Distress	Non-Distress
25	30/9/2010	10385319	2272855	1676228	7659928	1,464,200	430849	3E+06	987558	1.337	12658174	9336156	596627	0.04713	0.11567	0.03404	0.39602	0.07802	4.580843423	Non-distress	Non-distress	0.64645584	Distress	Distress	Non-Distress
26	31/12/2010	10350583	2584407	1550348	7759214	1,791,487	571950	3E+06	1E+06	1.503	12934990	9309562	1034059	0.07994	0.1385	0.04422	0.44646	0.09173	4.991860257	Non-distress	Non-distress	0.795351519	Distress	Distress	Non-Distress
27	31/3/2011	10419818	2802404	1695302	7710381	2,108,037	241719	3E+06	1E+06	1.598	13222222	9405683	1107102	0.08373	0.15943	0.01828	0.47155	0.07926	4.9369983	Non-distress	Non-distress	0.74619632	Distress	Distress	Non-Distress
28	30/6/2011	10251248	3207511	2024239	7534062	2,274,499	214803	3E+06	1E+06	2.092	13458759	9558301	1183272	0.08792	0.169	0.01596	0.60759	0.07992	5.122896997	Non-distress	Non-distress	0.839240259	Distress	Distress	Non-Distress
29	30/9/2011	10184783	3349406	2020731	7529582	2,301,795	251395	3E+06	1E+06	1.8	13534189	9550313	1328675	0.09817	0.17007	0.01857	0.52356	0.07953	5.123006577	Non-distress	Non-distress	0.810876685	Distress	Distress	Non-Distress
30	31/12/2011	9993150	3777193	2033593	7691530	2,454,095	490781	3E+06	1E+06	2.24	13770343	9725123	1743600	0.12662	0.17822	0.03564	0.63988	0.09242	5.572988917	Non-distress	Non-distress	0.995412768	Distress	Distress	Non-Distress
31	31/3/2012	10003017	3899359	2162264	7412235	2,589,753	240075	3E+06	1E+06	2.05	13902376	9574499	1737095	0.12495	0.18628	0.01727	0.59488	0.08404	5.417615181	Non-distress	Non-distress	0.90869085	Distress	Distress	Non-Distress
32	30/6/2012	11420935	4076325	2817172	7441466	2,753,366	205821	3E+06	1E+06	2.121	15497260	1E+07	1259153	0.08125	0.17767	0.01328	0.57461	0.07624	5.054788537	Non-distress	Non-distress	0.811072751	Distress	Distress	Non-Distress

# Calculate fundamental ratios and add them into the original finance data before imputation

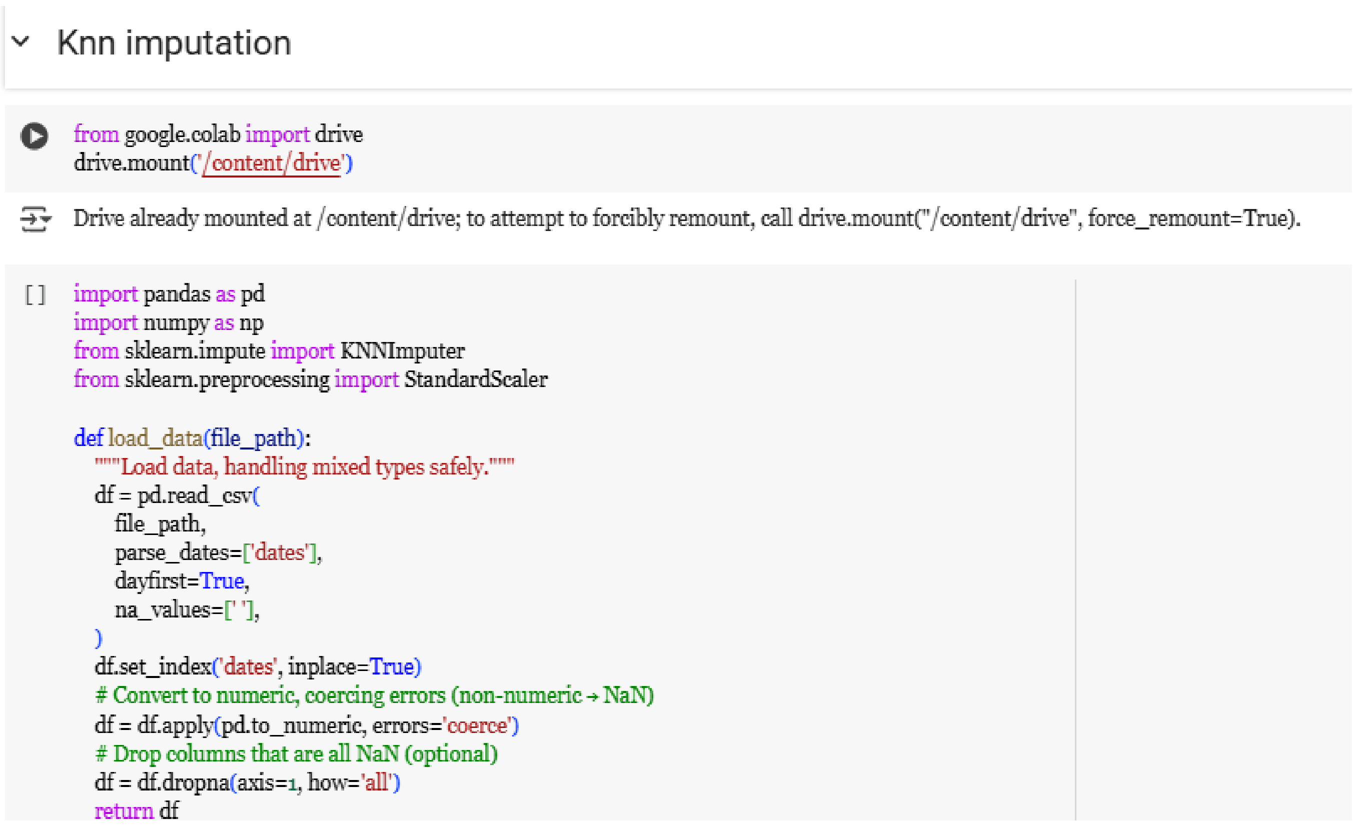


S26 69554

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA
1	dates	aviation_r	logistic_re	revenue	staff_cost	aircraft_fu	fuel_swap	maintenai	user_char	logistic_e	technolog	other_ope	other_ope	share_of_	share_of_	aircraft_o	operating	airasia.cc	derivative	teleport_(	airasia_a	bigpay_gr	other_aira	share_of_	red_beat_	ebitda	exception
2	31/12/2004	0	0	178633			0			0	0	-12717	3353	0	0	0	54847	0	0	0	0	0	0	0	0	0	0
3	31/3/2005	0	0	163911			0			0	0	-13103	12214	0	0	0	52004	0	0	0	0	0	0	0	0	0	0
4	30/6/2005	0	0	199508			0			0	0	-12029	2338	0	0	0	34366	0	0	0	0	0	0	0	0	0	0
5	30/9/2005	0	0	186277			0			0	0	-15093	4274	0	0	0	24757	0	0	0	0	0	0	0	0	0	0
6	31/12/2005	0	0	225925			0			0	0	-22065	8853	0	0	0	72897	0	0	0	0	0	0	0	0	0	0
7	31/3/2006	0	0	201681			0			0	0	-14670	9943	0	0	0	50143	0	0	0	0	0	0	0	0	0	0
8	30/6/2006	0	0	241683			0			0	0	-18305	979	0	0	0	66409	0	0	0	0	0	0	0	0	0	0
9	30/9/2006	0	0	239574			0			0	0	-15526	2345	0	0	0	58607	0	0	0	0	0	0	0	0	0	0
10	31/12/2006	0	0	400607			0			0	0	-17470	25593	0	0	0	168129	0	0	0	0	0	0	0	0	0	0
11	31/3/2007	0	0	396179			0			0	0	-25631	34957	0	0	0	119204	0	0	0	0	0	0	0	0	0	0
12	30/6/2007	0	0	432154			0			0	0	-4609	76574	0	0	0	185912	0	0	0	0	0	0	0	0	0	0
13	30/9/2007	0	0	461585			0			0	0	-23525	2387	0	0	0	145732	0	0	0	0	0	0	0	0	0	0
14	31/12/2007	0	0	632792			0			0	0	-29888	9006	0	0	0	223806	0	0	0	0	0	0	0	0	0	0
15	31/3/2008	0	0	535156			0			0	0	-23593	5505	0	0	0	129313	0	0	0	0	0	0	0	0	0	0
16	30/6/2008	0	0	608383			0			0	0	-28522	13922	0	0	0	154474	0	0	0	0	0	0	0	0	0	0
17	30/9/2008	0	0	658475			0			0	0	-29491	17088	0	0	0	91559	0	0	0	0	0	0	0	0	0	-215280
18	31/12/2008	0	0	838246			0			0	0	-26180	61115	0	0	0	298549	0	0	0	0	0	0	0	0	0	-425668
19	31/3/2009	0	0	714178			0			0	0	-28933	56367	0	0	0	410746	0	0	0	0	0	0	0	0	0	0
20	30/6/2009	0	0	657444			0			0	0	-35526	13414	0	0	0	322845	0	0	0	0	0	0	0	0	0	-6238
21	30/9/2009	0	0	739668			0			0	0	-49982	24457	0	0	0	230251	0	0	0	0	0	0	0	0	0	0
22	31/12/2009	0	0	894059			0			0	0	-59443	0	0	0	0	320976	0	0	0	0	0	0	0	0	0	28695
23	31/3/2010	0	0	878041			0			0	0	-42709	0	0	0	0	299827	0	-21195	0	0	0	0	0	0	0	0
24	30/6/2010	0	0	940656			0			0	0	-50383	0	0	0	0	363904	0	-24815	0	0	0	0	0	0	0	0

# Imputation Codes for finance data

APPENDIX

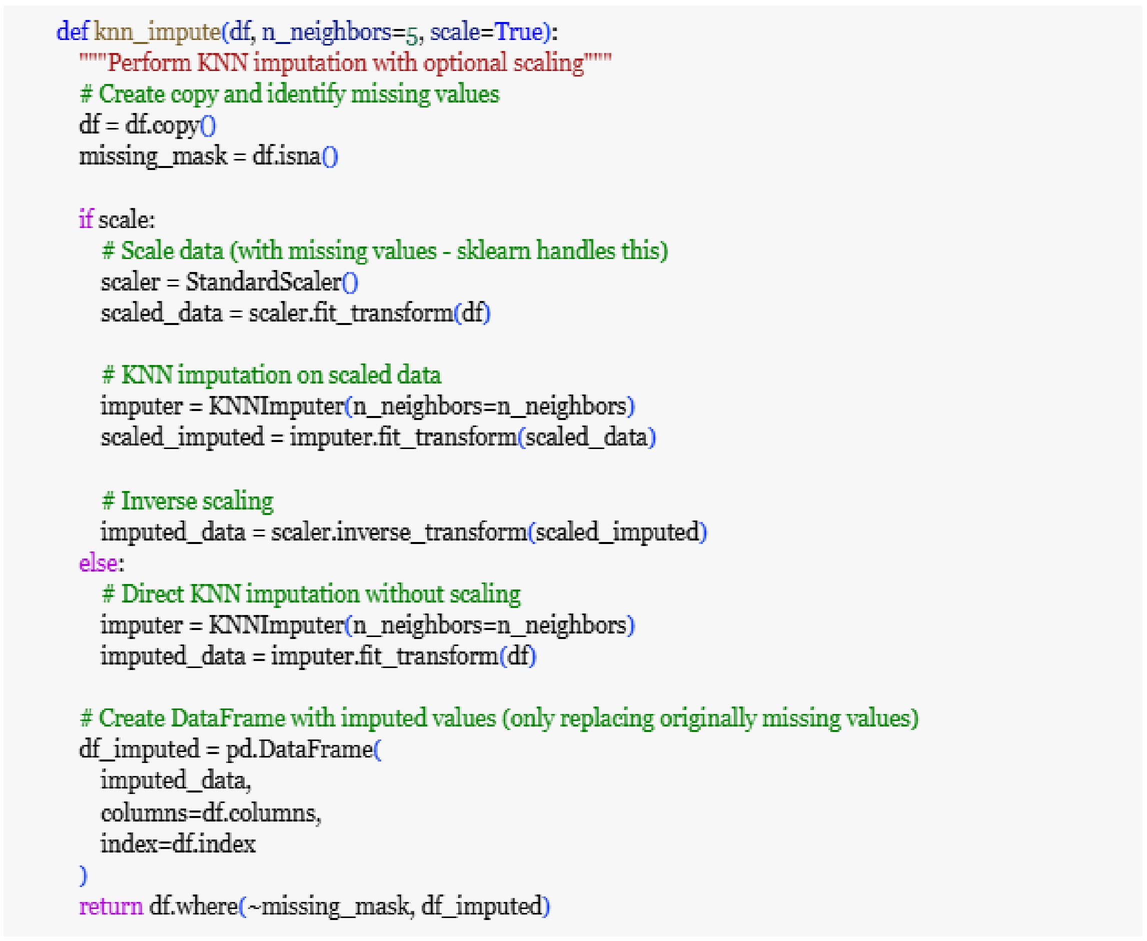


Knn imputation

```
from google.colab import drive
drive.mount('/content/drive')
```

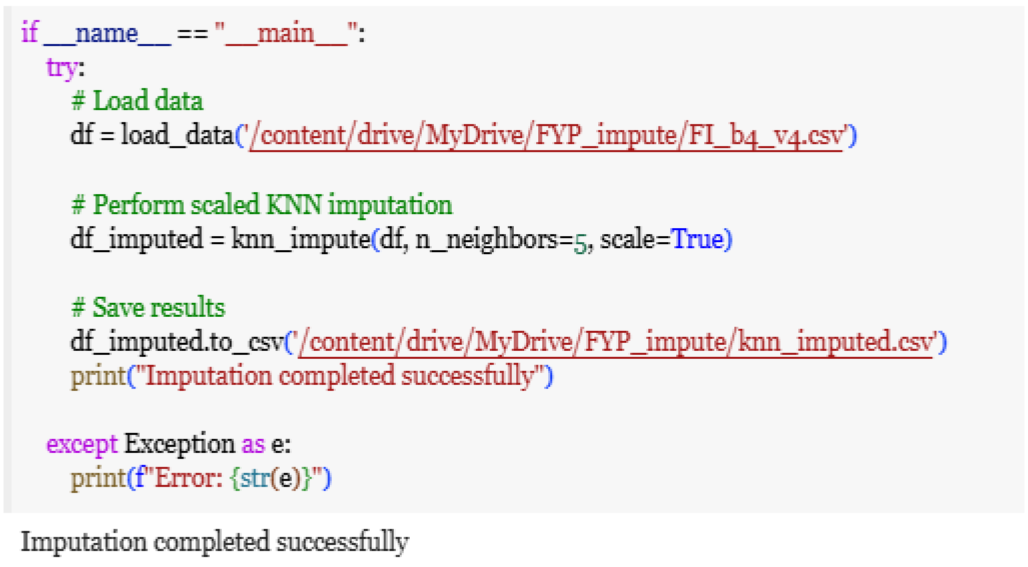
```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
import pandas as pd
import numpy as np
from sklearn.impute import KNNImputer
from sklearn.preprocessing import StandardScaler

def load_data(file_path):
  """Load data, handling mixed types safely."""
  df = pd.read_csv(
    file_path,
    parse_dates=['dates'],
    dayfirst=True,
    na_values=[' '],
  )
  df.set_index('dates', inplace=True)
  # Convert to numeric, coercing errors (non-numeric → NaN)
  df = df.apply(pd.to_numeric, errors='coerce')
  # Drop columns that are all NaN (optional)
  df = df.dropna(axis=1, how='all')
  return df
```



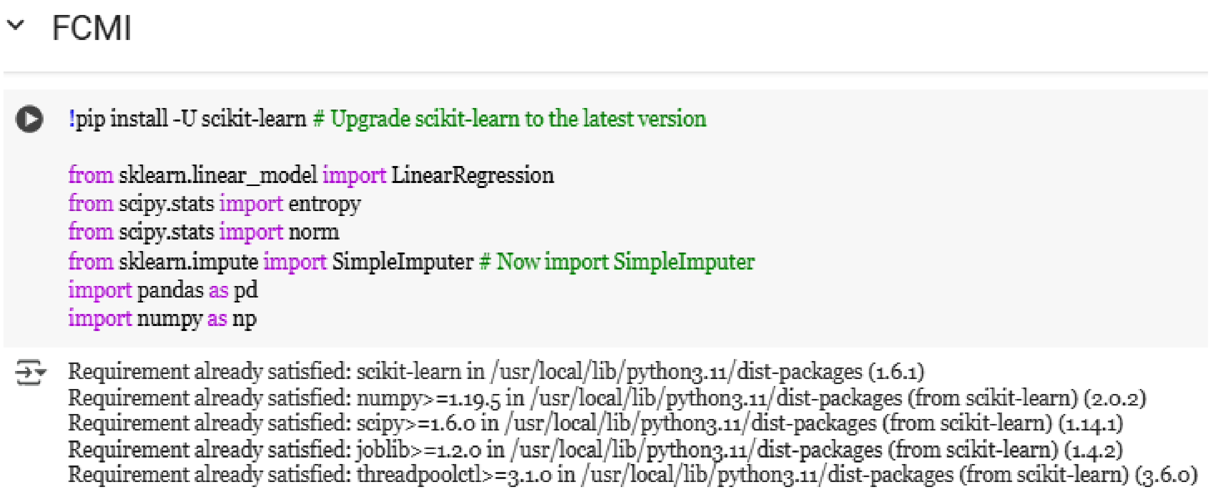
```
def knn_impute(df, n_neighbors=5, scale=True):
  """Perform KNN imputation with optional scaling"""
  # Create copy and identify missing values
  df = df.copy()
  missing_mask = df.isna()

  if scale:
    # Scale data (with missing values - sklearn handles this)
    scaler = StandardScaler()
    scaled_data = scaler.fit_transform(df)

    # KNN imputation on scaled data
    imputer = KNNImputer(n_neighbors=n_neighbors)
    scaled_imputed = imputer.fit_transform(scaled_data)

    # Inverse scaling
    imputed_data = scaler.inverse_transform(scaled_imputed)
  else:
    # Direct KNN imputation without scaling
    imputer = KNNImputer(n_neighbors=n_neighbors)
    imputed_data = imputer.fit_transform(df)

  # Create DataFrame with imputed values (only replacing originally missing values)
  df_imputed = pd.DataFrame(
    imputed_data,
    columns=df.columns,
    index=df.index
  )
  return df.where(~missing_mask, df_imputed)
```

APPENDIX



```
if __name__ == "__main__":
    try:
        # Load data
        df = load_data('/content/drive/MyDrive/FYP_impute/FI_b4_v4.csv')

        # Perform scaled KNN imputation
        df_imputed = knn_impute(df, n_neighbors=5, scale=True)

        # Save results
        df_imputed.to_csv('/content/drive/MyDrive/FYP_impute/knn_imputed.csv')
        print("Imputation completed successfully")

    except Exception as e:
        print(f"Error: {str(e)}")
```

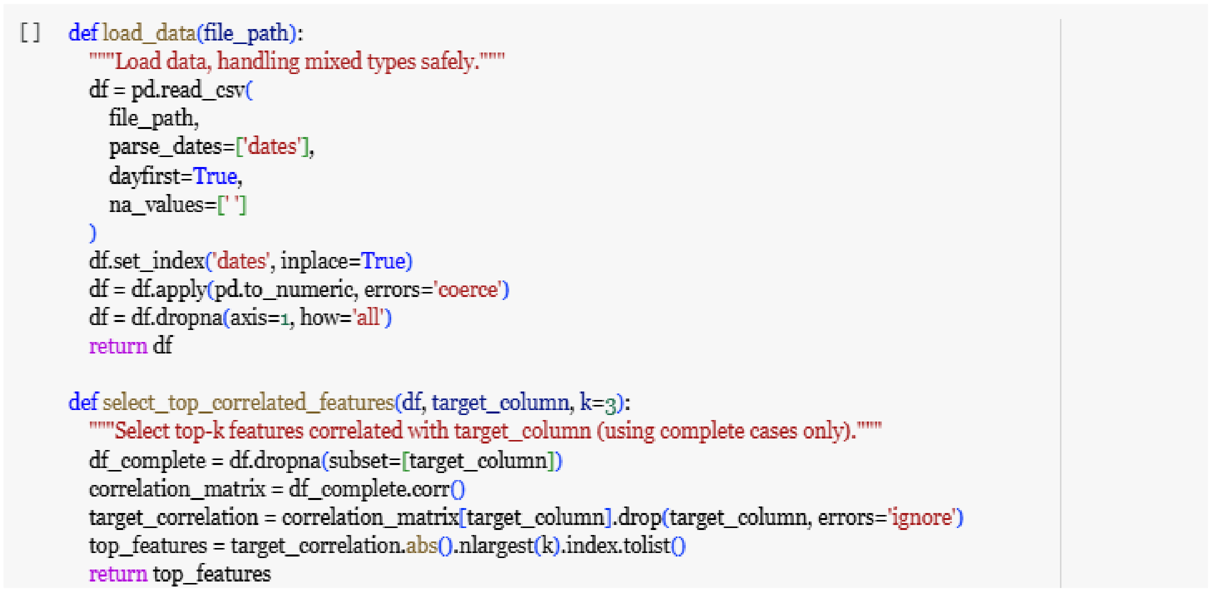
```
Imputation completed successfully
```



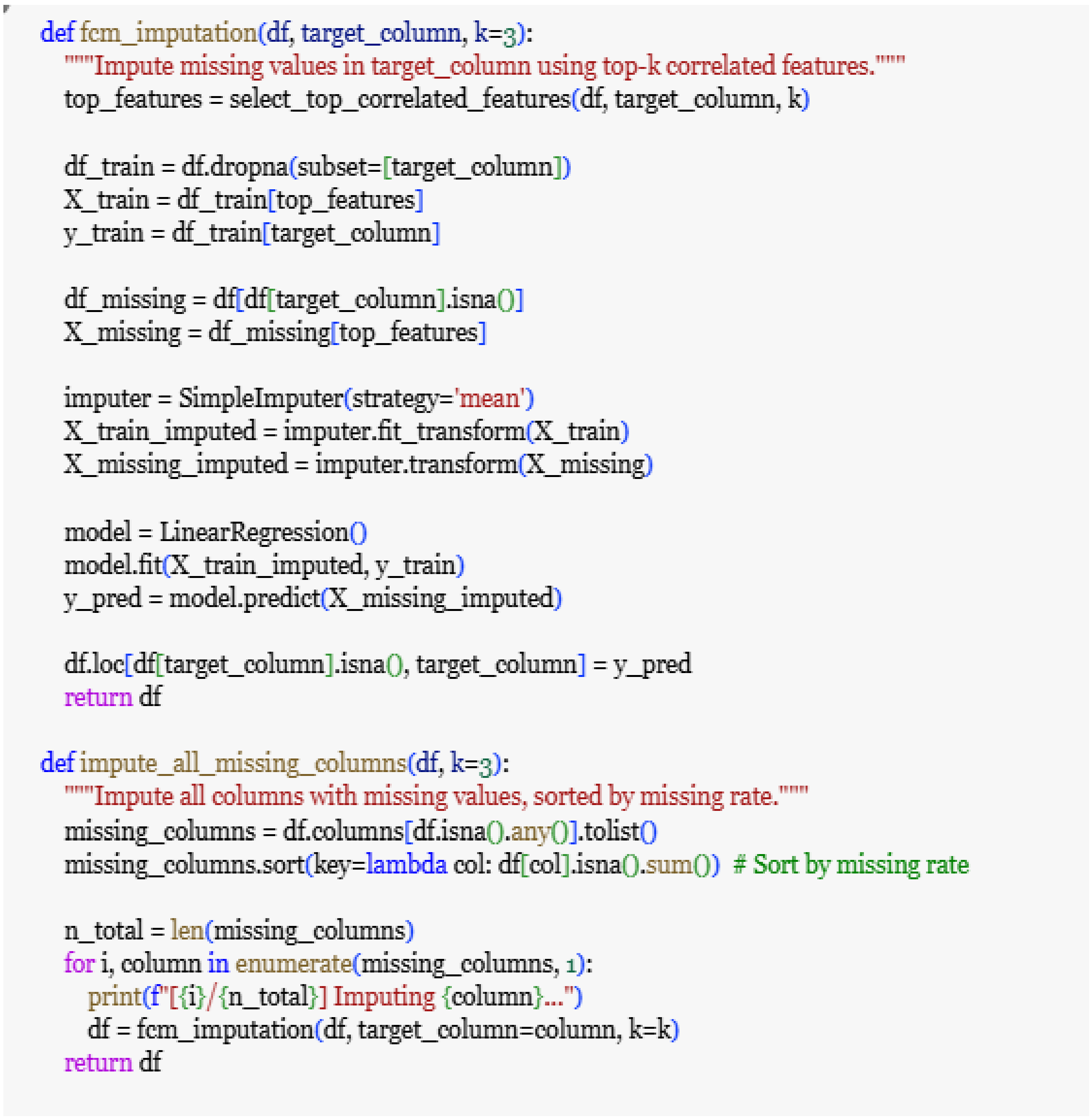
FCMI

```
!pip install -U scikit-learn # Upgrade scikit-learn to the latest version

from sklearn.linear_model import LinearRegression
from scipy.stats import entropy
from scipy.stats import norm
from sklearn.impute import SimpleImputer # Now import SimpleImputer
import pandas as pd
import numpy as np
```

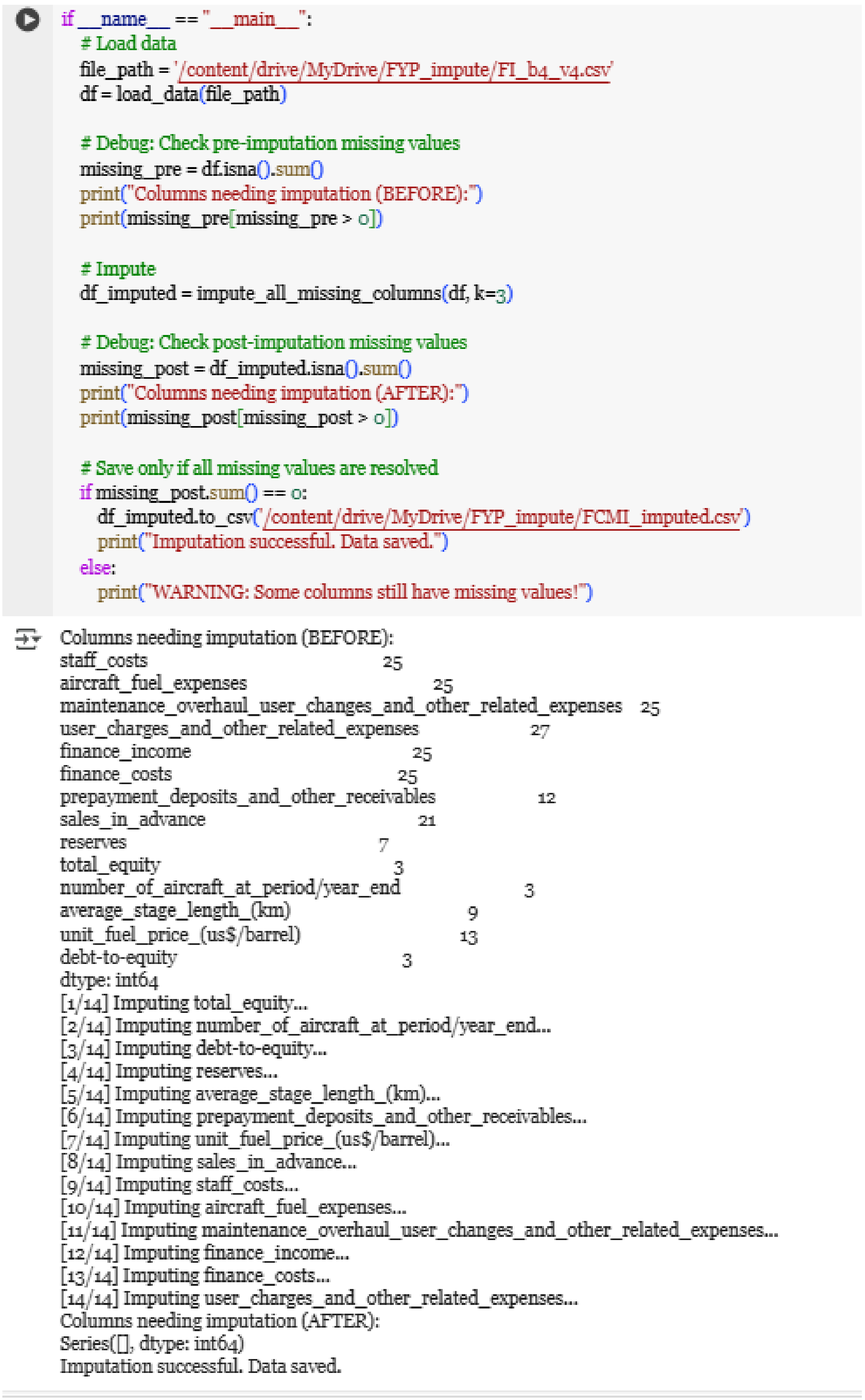
```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (1.6.1)
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (2.0.2)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.14.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (3.6.0)
```



```
[ ]  def load_data(file_path):
         """Load data, handling mixed types safely."""
         df = pd.read_csv(
             file_path,
             parse_dates=['dates'],
             dayfirst=True,
             na_values=[' ']
         )
         df.set_index('dates', inplace=True)
         df = df.apply(pd.to_numeric, errors='coerce')
         df = df.dropna(axis=1, how='all')
         return df

     def select_top_correlated_features(df, target_column, k=3):
         """Select top-k features correlated with target_column (using complete cases only)."""
         df_complete = df.dropna(subset=[target_column])
         correlation_matrix = df_complete.corr()
         target_correlation = correlation_matrix[target_column].drop(target_column, errors='ignore')
         top_features = target_correlation.abs().nlargest(k).index.tolist()
         return top_features
```



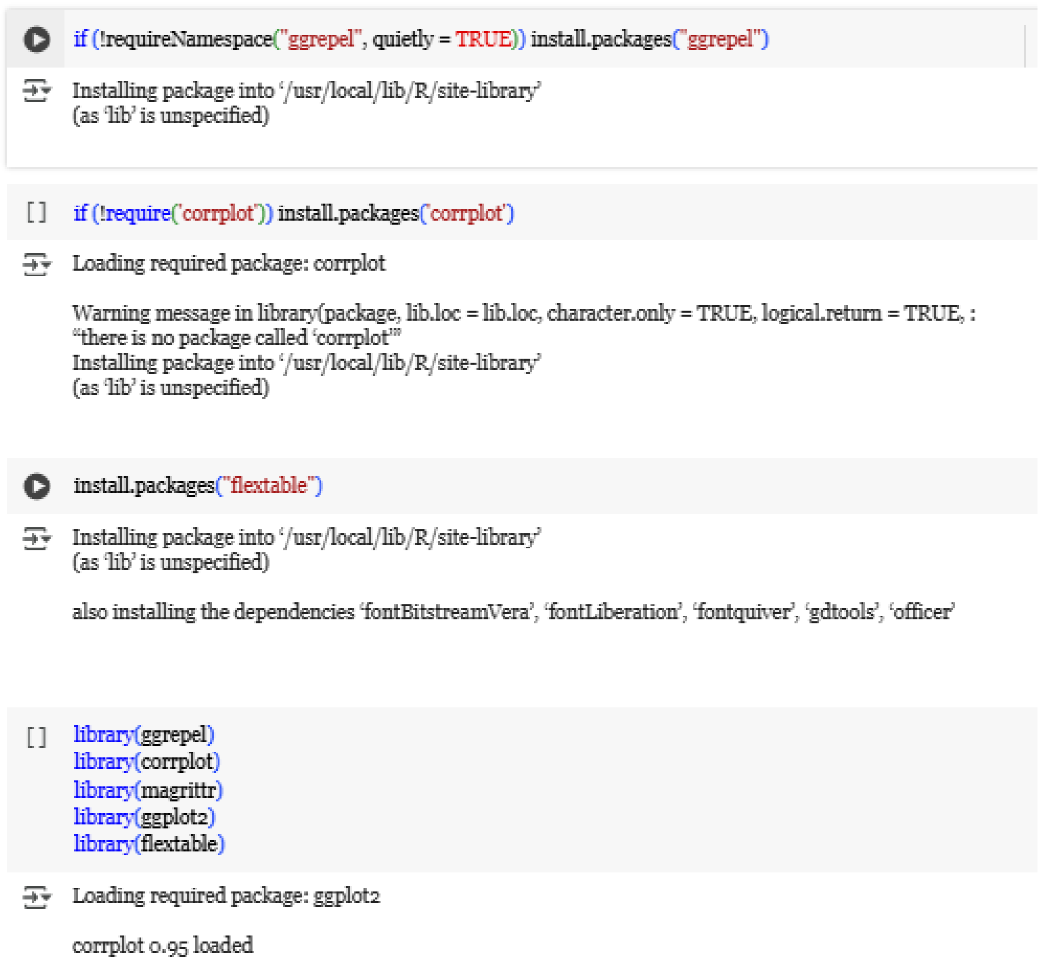
```python
def fcm_imputation(df, target_column, k=3):
    """Impute missing values in target_column using top-k correlated features."""
    top_features = select_top_correlated_features(df, target_column, k)

    df_train = df.dropna(subset=[target_column])
    X_train = df_train[top_features]
    y_train = df_train[target_column]

    df_missing = df[df[target_column].isna()]
    X_missing = df_missing[top_features]

    imputer = SimpleImputer(strategy='mean')
    X_train_imputed = imputer.fit_transform(X_train)
    X_missing_imputed = imputer.transform(X_missing)

    model = LinearRegression()
    model.fit(X_train_imputed, y_train)
    y_pred = model.predict(X_missing_imputed)

    df.loc[df[target_column].isna(), target_column] = y_pred
    return df

def impute_all_missing_columns(df, k=3):
    """Impute all columns with missing values, sorted by missing rate."""
    missing_columns = df.columns[df.isna().any()].tolist()
    missing_columns.sort(key=lambda col: df[col].isna().sum())  # Sort by missing rate

    n_total = len(missing_columns)
    for i, column in enumerate(missing_columns, 1):
        print(f"[{i}/{n_total}] Imputing {column}...")
        df = fcm_imputation(df, target_column=column, k=k)
    return df
```



```
if __name__ == "__main__":
  # Load data
  file_path = '/content/drive/MyDrive/FYP_impute/FI_b4_v4.csv'
  df = load_data(file_path)

  # Debug: Check pre-imputation missing values
  missing_pre = df.isna().sum()
  print("Columns needing imputation (BEFORE):")
  print(missing_pre[missing_pre > 0])

  # Impute
  df_imputed = impute_all_missing_columns(df, k=3)

  # Debug: Check post-imputation missing values
  missing_post = df_imputed.isna().sum()
  print("Columns needing imputation (AFTER):")
  print(missing_post[missing_post > 0])

  # Save only if all missing values are resolved
  if missing_post.sum() == 0:
    df_imputed.to_csv('/content/drive/MyDrive/FYP_impute/FCMI_imputed.csv')
    print("Imputation successful. Data saved.")
  else:
    print("WARNING: Some columns still have missing values!")
```

```
Columns needing imputation (BEFORE):
staff_costs                                                25
aircraft_fuel_expenses                                     25
maintenance_overhaul_user_changes_and_other_related_expenses    25
user_charges_and_other_related_expenses                    27
finance_income                                             25
finance_costs                                              25
prepayment_deposits_and_other_receivables                  12
sales_in_advance                                           21
reserves                                                    7
total_equity                                                3
number_of_aircraft_at_period/year_end                       3
average_stage_length_(km)                                   9
unit_fuel_price_(us$/barrel)                               13
debt-to-equity                                              3
dtype: int64
[1/14] Imputing total_equity...
[2/14] Imputing number_of_aircraft_at_period/year_end...
[3/14] Imputing debt-to-equity...
[4/14] Imputing reserves...
[5/14] Imputing average_stage_length_(km)...
[6/14] Imputing prepayment_deposits_and_other_receivables...
[7/14] Imputing unit_fuel_price_(us$/barrel)...
[8/14] Imputing sales_in_advance...
[9/14] Imputing staff_costs...
[10/14] Imputing aircraft_fuel_expenses...
[11/14] Imputing maintenance_overhaul_user_changes_and_other_related_expenses...
[12/14] Imputing finance_income...
[13/14] Imputing finance_costs...
[14/14] Imputing user_charges_and_other_related_expenses...
Columns needing imputation (AFTER):
Series([], dtype: int64)
Imputation successful. Data saved.
```

# Code for pre-post imputation analysis using R square scatter plot and summary statistics (mean, median, SE Mean)



```
if (!requireNamespace("ggrepel", quietly = TRUE)) install.packages("ggrepel")
```

```
Installing package into '/usr/local/lib/R/site-library'
(as 'lib' is unspecified)
```

```
if (!require('corrplot')) install.packages('corrplot')
```

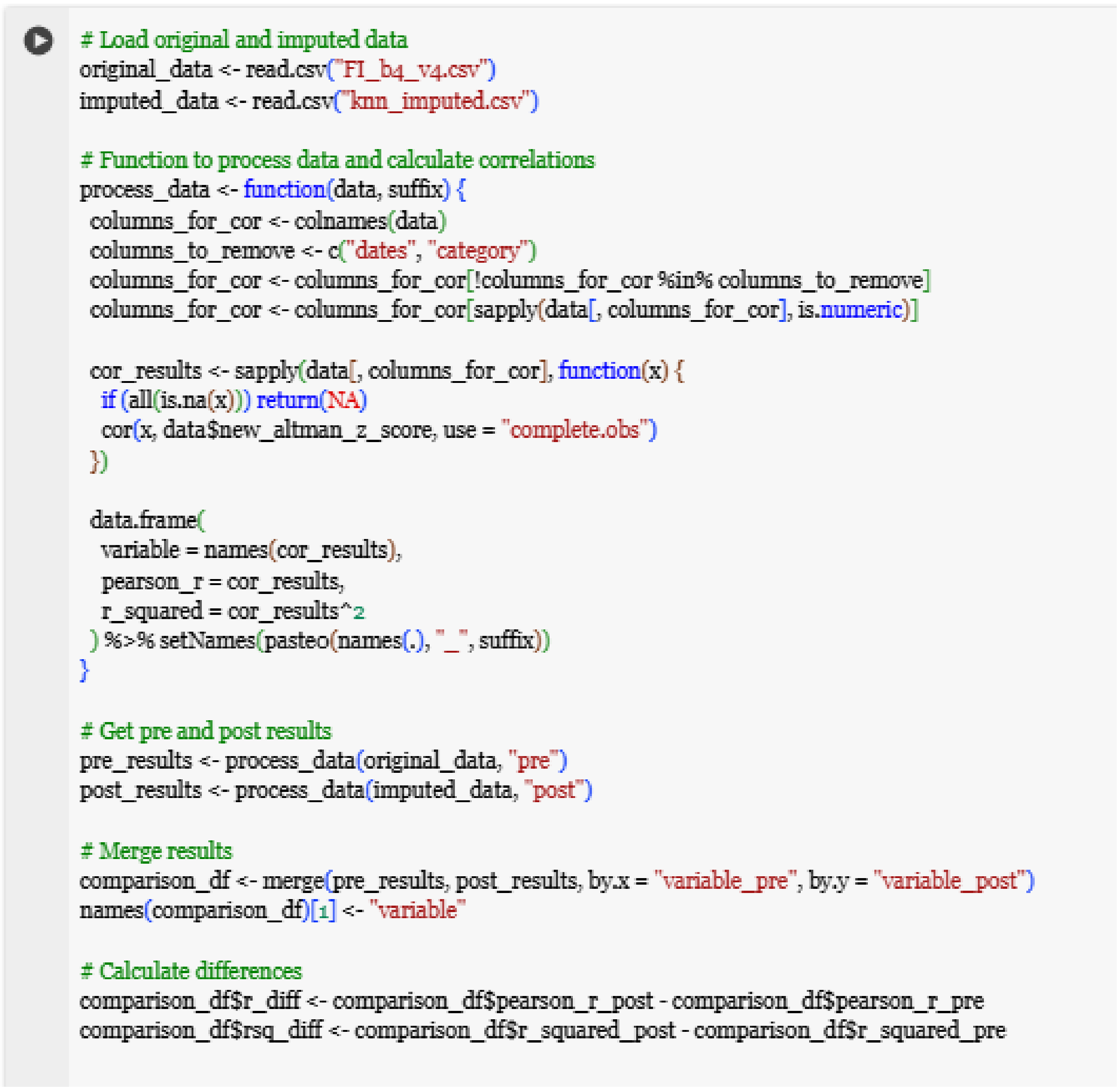
```
Loading required package: corrplot

Warning message in library(package, lib.loc = lib.loc, character.only = TRUE, logical.return = TRUE, :
"there is no package called 'corrplot'"
Installing package into '/usr/local/lib/R/site-library'
(as 'lib' is unspecified)
```

```
install.packages("flextable")
```

```
Installing package into '/usr/local/lib/R/site-library'
(as 'lib' is unspecified)

also installing the dependencies 'fontBitstreamVera', 'fontLiberation', 'fontquiver', 'gdtools', 'officer'
```

```
library(ggrepel)
library(corrplot)
library(magrittr)
library(ggplot2)
library(flextable)
```

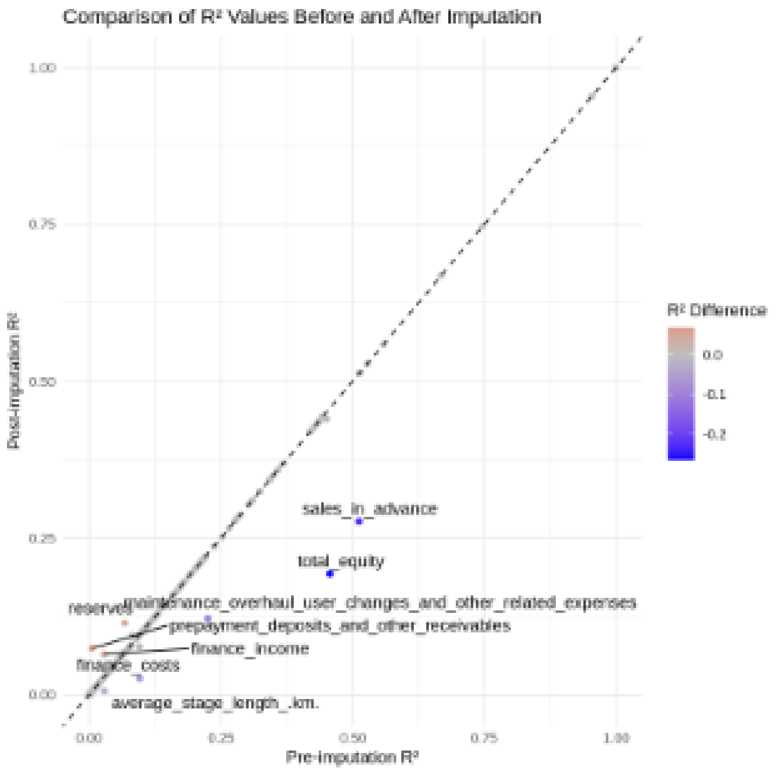
```
Loading required package: ggplot2

corrplot 0.95 loaded
```

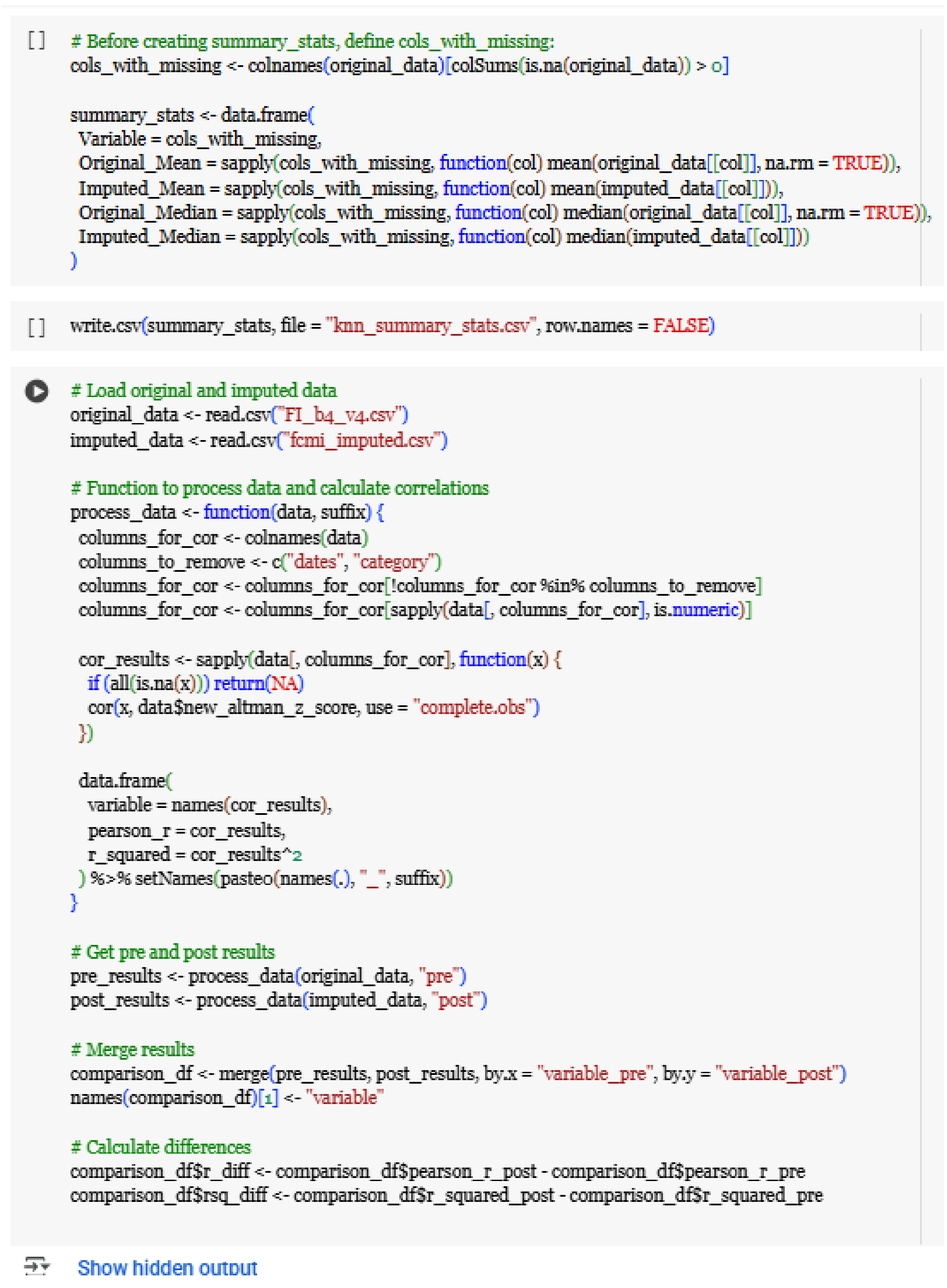


```
# Load original and imputed data
original_data <- read.csv("FI_b4_v4.csv")
imputed_data <- read.csv("knn_imputed.csv")

# Function to process data and calculate correlations
process_data <- function(data, suffix) {
 columns_for_cor <- colnames(data)
 columns_to_remove <- c("dates", "category")
 columns_for_cor <- columns_for_cor[!columns_for_cor %in% columns_to_remove]
 columns_for_cor <- columns_for_cor[sapply(data[, columns_for_cor], is.numeric)]

 cor_results <- sapply(data[, columns_for_cor], function(x) {
  if (all(is.na(x))) return(NA)
  cor(x, data$new_altman_z_score, use = "complete.obs")
 })

 data.frame(
  variable = names(cor_results),
  pearson_r = cor_results,
  r_squared = cor_results^2
 ) %>% setNames(paste0(names(.), "_", suffix))
}

# Get pre and post results
pre_results <- process_data(original_data, "pre")
post_results <- process_data(imputed_data, "post")

# Merge results
comparison_df <- merge(pre_results, post_results, by.x = "variable_pre", by.y = "variable_post")
names(comparison_df)[1] <- "variable"

# Calculate differences
comparison_df$r_diff <- comparison_df$pearson_r_post - comparison_df$pearson_r_pre
comparison_df$rsq_diff <- comparison_df$r_squared_post - comparison_df$r_squared_pre
```

```r
# 1. Create scatter plot of R² values
# Add label only for variables with largest R² differences
threshold <- 0.02
label_data <- comparison_df[abs(comparison_df$rsq_diff) > threshold, ]

# Plot with color representing change in R²
ggplot(comparison_df, aes(x = r_squared_pre, y = r_squared_post)) +
  geom_point(aes(color = rsq_diff)) +
  geom_abline(intercept = 0, slope = 1, linetype = "dashed") +
  geom_text_repel(data = label_data, aes(label = variable)) +
  scale_color_gradient2(low = "blue", mid = "gray", high = "red", midpoint = 0) +
  labs(
    x = "Pre-imputation R²",
    y = "Post-imputation R²",
    color = "R² Difference",
    title = "Comparison of R² Values Before and After Imputation"
  ) +
  theme_minimal()
```

```
Warning message:
“Removed 12 rows containing missing values or values outside the scale range
(`geom_point()`).”
Warning message:
“Removed 12 rows containing missing values or values outside the scale range
(`geom_text_repel()`).”
```





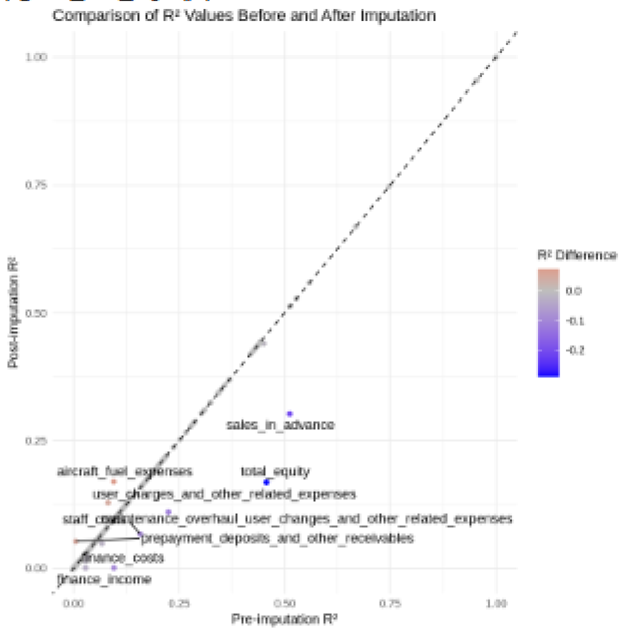
```
[ ] # Before creating summary_stats, define cols_with_missing:
    cols_with_missing <- colnames(original_data)[colSums(is.na(original_data)) > 0]

    summary_stats <- data.frame(
      Variable = cols_with_missing,
      Original_Mean = sapply(cols_with_missing, function(col) mean(original_data[[col]], na.rm = TRUE)),
      Imputed_Mean = sapply(cols_with_missing, function(col) mean(imputed_data[[col]])),
      Original_Median = sapply(cols_with_missing, function(col) median(original_data[[col]], na.rm = TRUE)),
      Imputed_Median = sapply(cols_with_missing, function(col) median(imputed_data[[col]]))
    )
```

```
[ ] write.csv(summary_stats, file = "knn_summary_stats.csv", row.names = FALSE)
```

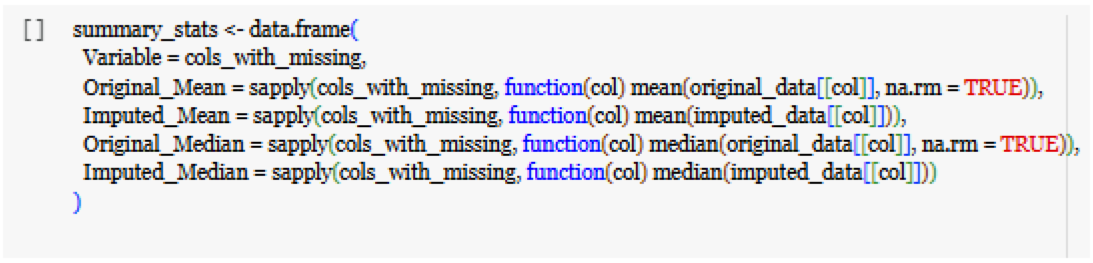
```
# Load original and imputed data
original_data <- read.csv("FI_b4_v4.csv")
imputed_data <- read.csv("fcmi_imputed.csv")

# Function to process data and calculate correlations
process_data <- function(data, suffix) {
  columns_for_cor <- colnames(data)
  columns_to_remove <- c("dates", "category")
  columns_for_cor <- columns_for_cor[!columns_for_cor %in% columns_to_remove]
  columns_for_cor <- columns_for_cor[sapply(data[, columns_for_cor], is.numeric)]

  cor_results <- sapply(data[, columns_for_cor], function(x) {
    if (all(is.na(x))) return(NA)
    cor(x, data$new_altman_z_score, use = "complete.obs")
  })

  data.frame(
    variable = names(cor_results),
    pearson_r = cor_results,
    r_squared = cor_results^2
  ) %>% setNames(paste0(names(.), "_", suffix))
}

# Get pre and post results
pre_results <- process_data(original_data, "pre")
post_results <- process_data(imputed_data, "post")

# Merge results
comparison_df <- merge(pre_results, post_results, by.x = "variable_pre", by.y = "variable_post")
names(comparison_df)[1] <- "variable"

# Calculate differences
comparison_df$r_diff <- comparison_df$pearson_r_post - comparison_df$pearson_r_pre
comparison_df$rsq_diff <- comparison_df$r_squared_post - comparison_df$r_squared_pre
```

```r
# 1. Create scatter plot of R² values
# Add label only for variables with largest R² differences
threshold <- 0.02
label_data <- comparison_df[abs(comparison_df$rsq_diff) > threshold, ]

# Plot with color representing change in R²
ggplot(comparison_df, aes(x = r_squared_pre, y = r_squared_post)) +
 geom_point(aes(color = rsq_diff)) +
 geom_abline(intercept = 0, slope = 1, linetype = "dashed") +
 geom_text_repel(data = label_data, aes(label = variable)) +
 scale_color_gradient2(low = "blue", mid = "gray", high = "red", midpoint = 0) +
 labs(
  x = "Pre-imputation R²",
  y = "Post-imputation R²",
  color = "R² Difference",
  title = "Comparison of R² Values Before and After Imputation"
 ) +
 theme_minimal()
```

```
Warning message:
“Removed 12 rows containing missing values or values outside the scale range
(`geom_point()`).”
Warning message:
“Removed 12 rows containing missing values or values outside the scale range
(`geom_text_repel()`).”
```

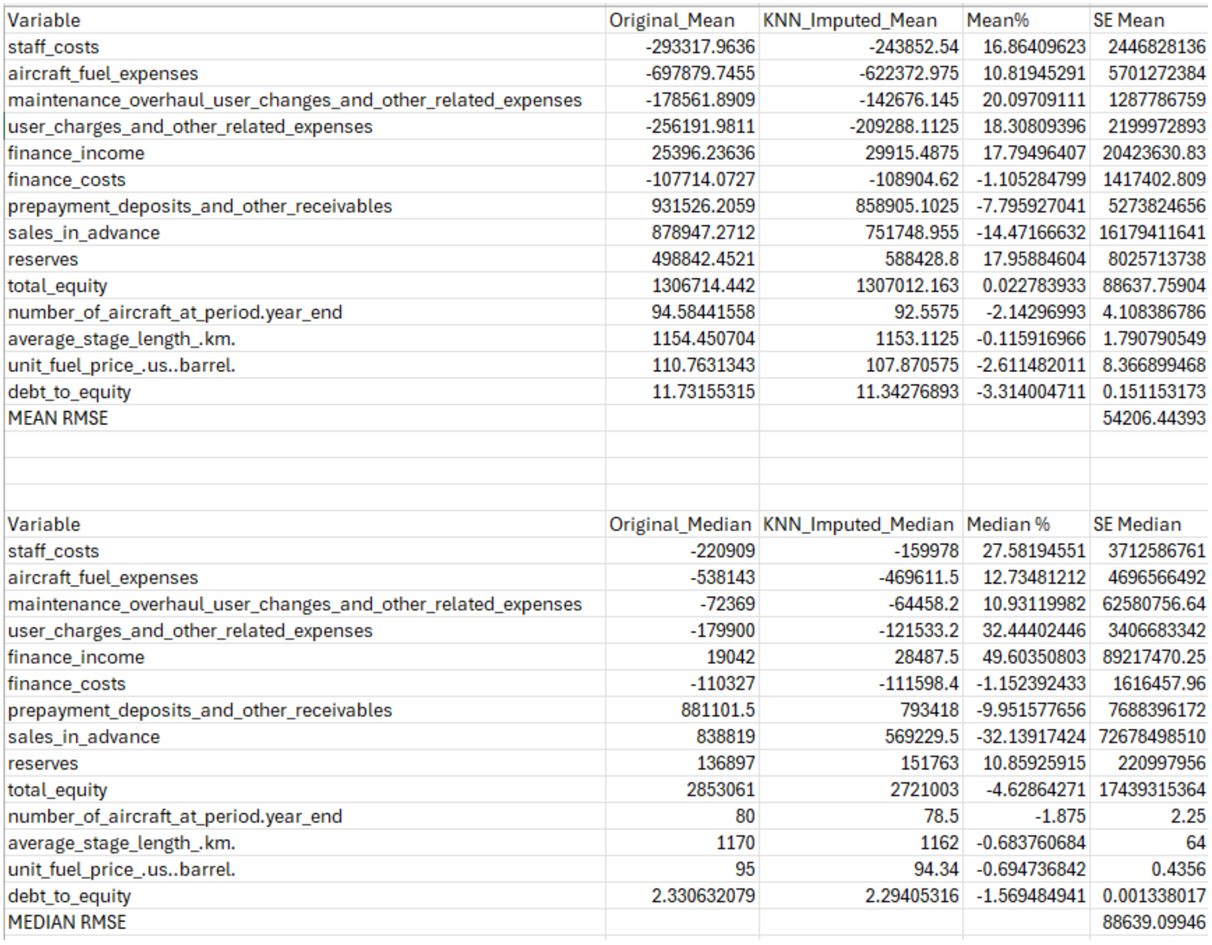


```r
summary_stats <- data.frame(
 Variable = cols_with_missing,
 Original_Mean = sapply(cols_with_missing, function(col) mean(original_data[[col]], na.rm = TRUE)),
 Imputed_Mean = sapply(cols_with_missing, function(col) mean(imputed_data[[col]])),
 Original_Median = sapply(cols_with_missing, function(col) median(original_data[[col]], na.rm = TRUE)),
 Imputed_Median = sapply(cols_with_missing, function(col) median(imputed_data[[col]]))
)
```



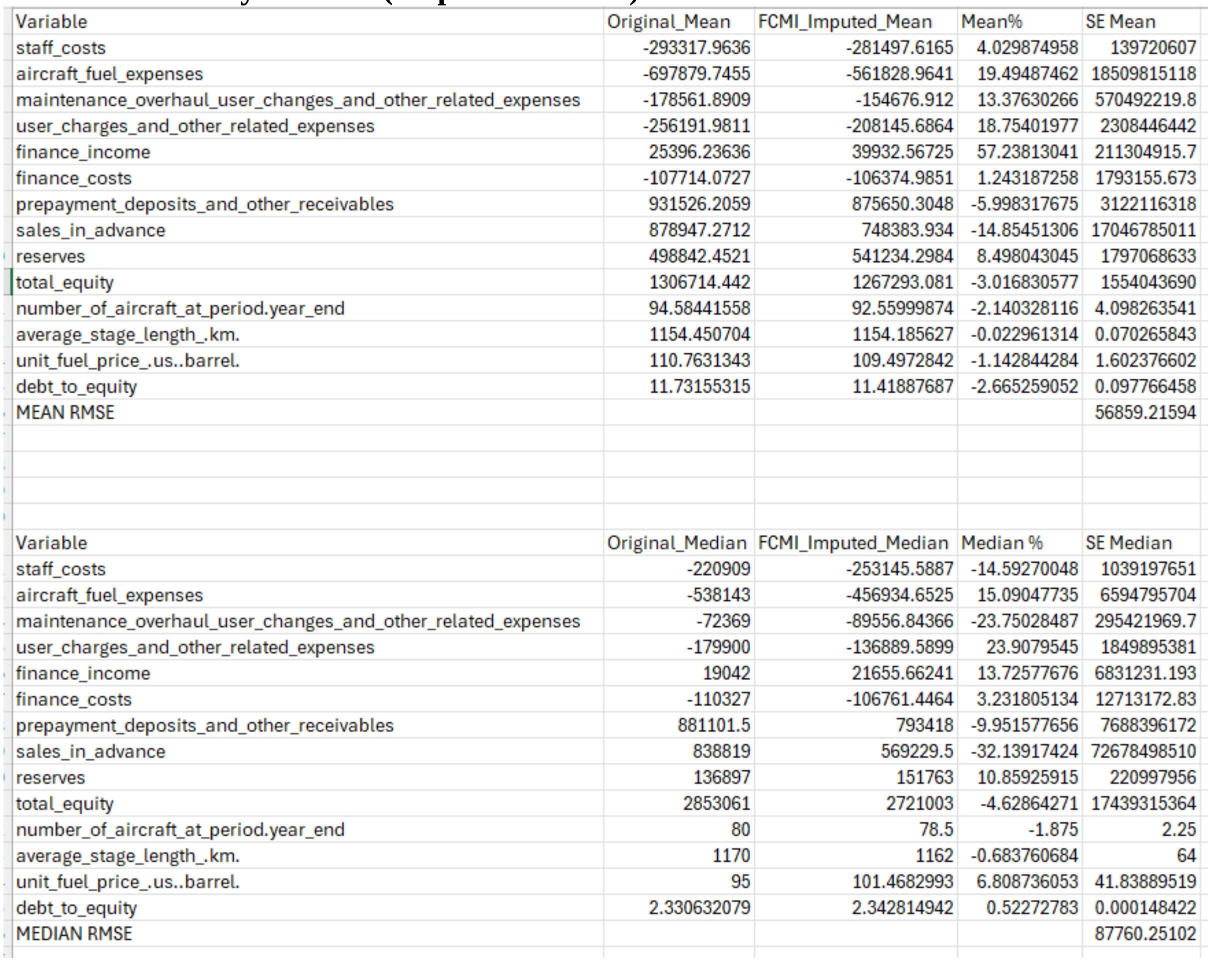
```
[ ]  summary_stats <- data.frame(
      Variable = cols_with_missing,
      Original_Mean = sapply(cols_with_missing, function(col) mean(original_data[[col]], na.rm = TRUE)),
      Imputed_Mean = sapply(cols_with_missing, function(col) mean(imputed_data[[col]])),
      Original_Median = sapply(cols_with_missing, function(col) median(original_data[[col]], na.rm = TRUE)),
      Imputed_Median = sapply(cols_with_missing, function(col) median(imputed_data[[col]]))
     )
```

# KNN Summary Statistic (output from code)



Variable	Original_Mean	KNN_Imputed_Mean	Mean%	SE Mean
staff_costs	-293317.9636	-243852.54	16.86409623	2446828136
aircraft_fuel_expenses	-697879.7455	-622372.975	10.81945291	5701272384
maintenance_overhaul_user_changes_and_other_related_expenses	-178561.8909	-142676.145	20.09709111	1287786759
user_charges_and_other_related_expenses	-256191.9811	-209288.1125	18.30809396	2199972893
finance_income	25396.23636	29915.4875	17.79496407	20423630.83
finance_costs	-107714.0727	-108904.62	-1.105284799	1417402.809
prepayment_deposits_and_other_receivables	931526.2059	858905.1025	-7.795927041	5273824656
sales_in_advance	878947.2712	751748.955	-14.47166632	16179411641
reserves	498842.4521	588428.8	17.95884604	8025713738
total_equity	1306714.442	1307012.163	0.022783933	88637.75904
number_of_aircraft_at_period.year_end	94.58441558	92.5575	-2.14296993	4.108386786
average_stage_length_.km.	1154.450704	1153.1125	-0.115916966	1.790790549
unit_fuel_price_.us..barrel.	110.7631343	107.870575	-2.611482011	8.366899468
debt_to_equity	11.73155315	11.34276893	-3.314004711	0.151153173
MEAN RMSE				54206.44393
Variable	Original_Median	KNN_Imputed_Median	Median %	SE Median
staff_costs	-220909	-159978	27.58194551	3712586761
aircraft_fuel_expenses	-538143	-469611.5	12.73481212	4696566492
maintenance_overhaul_user_changes_and_other_related_expenses	-72369	-64458.2	10.93119982	62580756.64
user_charges_and_other_related_expenses	-179900	-121533.2	32.44402446	3406683342
finance_income	19042	28487.5	49.60350803	89217470.25
finance_costs	-110327	-111598.4	-1.152392433	1616457.96
prepayment_deposits_and_other_receivables	881101.5	793418	-9.951577656	7688396172
sales_in_advance	838819	569229.5	-32.13917424	72678498510
reserves	136897	151763	10.85925915	220997956
total_equity	2853061	2721003	-4.62864271	17439315364
number_of_aircraft_at_period.year_end	80	78.5	-1.875	2.25
average_stage_length_.km.	1170	1162	-0.683760684	64
unit_fuel_price_.us..barrel.	95	94.34	-0.694736842	0.4356
debt_to_equity	2.330632079	2.29405316	-1.569484941	0.001338017
MEDIAN RMSE				88639.09946

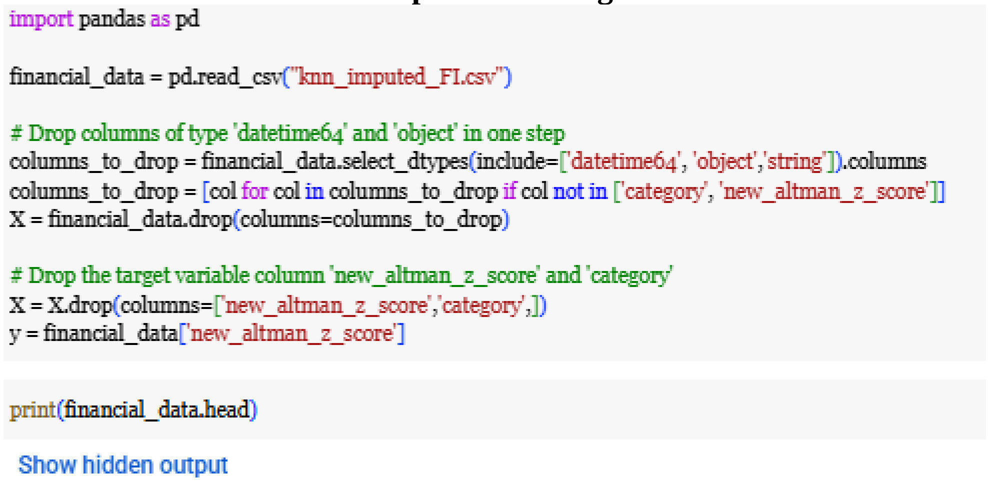
# FCMI Summary Statistic (output from code)



Variable	Original_Mean	FCMI_Imputed_Mean	Mean%	SE Mean
staff_costs	-293317.9636	-281497.6165	4.029874958	139720607
aircraft_fuel_expenses	-697879.7455	-561828.9641	19.49487462	18509815118
maintenance_overhaul_user_changes_and_other_related_expenses	-178561.8909	-154676.912	13.37630266	570492219.8
user_charges_and_other_related_expenses	-256191.9811	-208145.6864	18.75401977	2308446442
finance_income	25396.23636	39932.56725	57.23813041	211304915.7
finance_costs	-107714.0727	-106374.9851	1.243187258	1793155.673
prepayment_deposits_and_other_receivables	931526.2059	875650.3048	-5.998317675	3122116318
sales_in_advance	878947.2712	748383.934	-14.85451306	17046785011
reserves	498842.4521	541234.2984	8.498043045	1797068633
total_equity	1306714.442	1267293.081	-3.016830577	1554043690
number_of_aircraft_at_period.year_end	94.58441558	92.55999874	-2.140328116	4.098263541
average_stage_length_.km.	1154.450704	1154.185627	-0.022961314	0.070265843
unit_fuel_price_.us..barrel.	110.7631343	109.4972842	-1.142844284	1.602376602
debt_to_equity	11.73155315	11.41887687	-2.665259052	0.097766458
MEAN RMSE				56859.21594

Variable	Original_Median	FCMI_Imputed_Median	Median %	SE Median
staff_costs	-220909	-253145.5887	-14.59270048	1039197651
aircraft_fuel_expenses	-538143	-456934.6525	15.09047735	6594795704
maintenance_overhaul_user_changes_and_other_related_expenses	-72369	-89556.84366	-23.75028487	295421969.7
user_charges_and_other_related_expenses	-179900	-136889.5899	23.9079545	1849895381
finance_income	19042	21655.66241	13.72577676	6831231.193
finance_costs	-110327	-106761.4464	3.231805134	12713172.83
prepayment_deposits_and_other_receivables	881101.5	793418	-9.951577656	7688396172
sales_in_advance	838819	569229.5	-32.13917424	72678498510
reserves	136897	151763	10.85925915	220997956
total_equity	2853061	2721003	-4.62864271	17439315364
number_of_aircraft_at_period.year_end	80	78.5	-1.875	2.25
average_stage_length_.km.	1170	1162	-0.683760684	64
unit_fuel_price_.us..barrel.	95	101.4682993	6.808736053	41.83889519
debt_to_equity	2.330632079	2.342814942	0.52272783	0.000148422
MEDIAN RMSE				87760.25102

# Code to calculate feature importance using Decision Tree

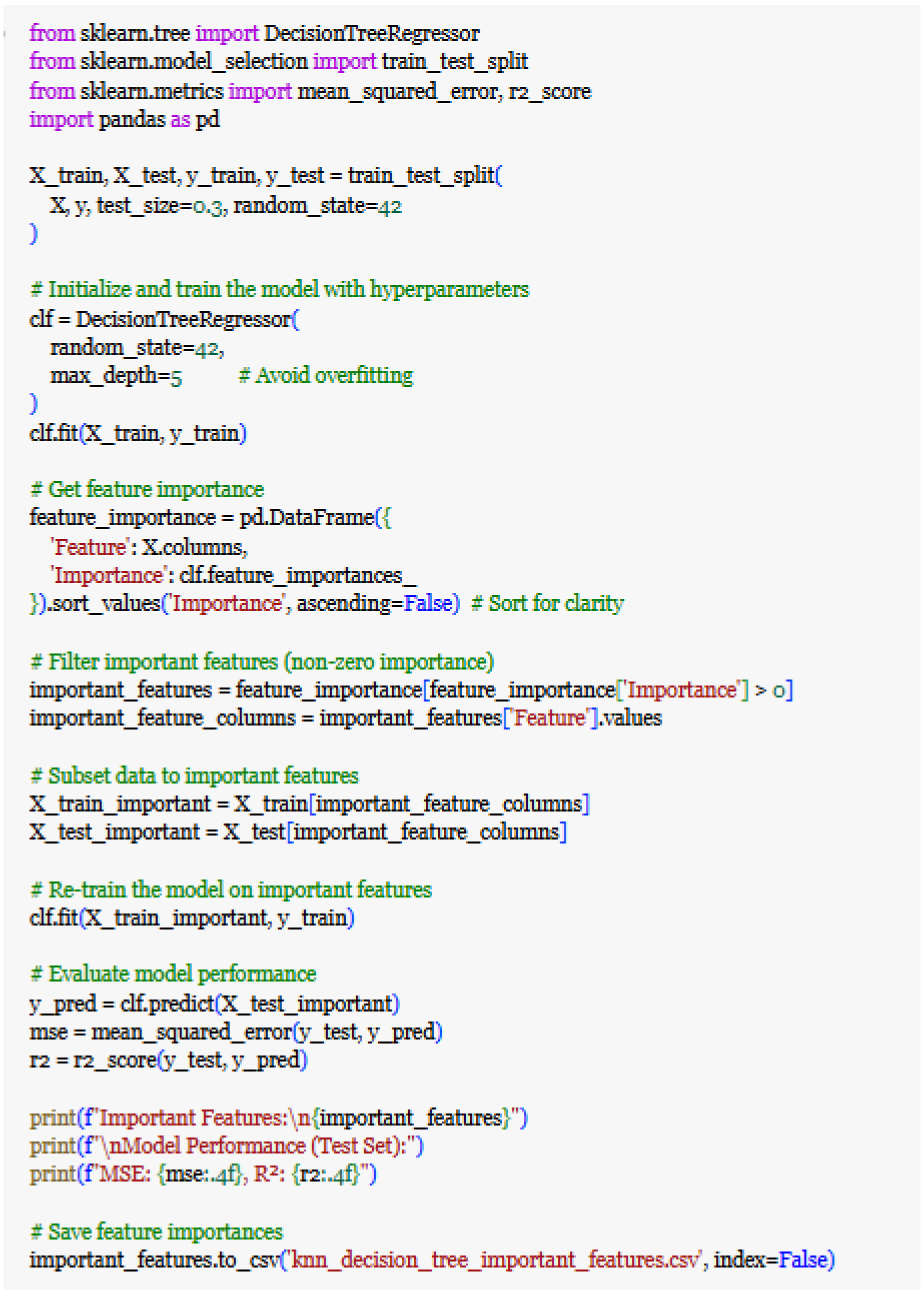


```
import pandas as pd

financial_data = pd.read_csv("knn_imputed_FI.csv")

# Drop columns of type 'datetime64' and 'object' in one step
columns_to_drop = financial_data.select_dtypes(include=['datetime64', 'object','string']).columns
columns_to_drop = [col for col in columns_to_drop if col not in ['category', 'new_altman_z_score']]
X = financial_data.drop(columns=columns_to_drop)

# Drop the target variable column 'new_altman_z_score' and 'category'
X = X.drop(columns=['new_altman_z_score','category',])
y = financial_data['new_altman_z_score']
```

```
print(financial_data.head)
```

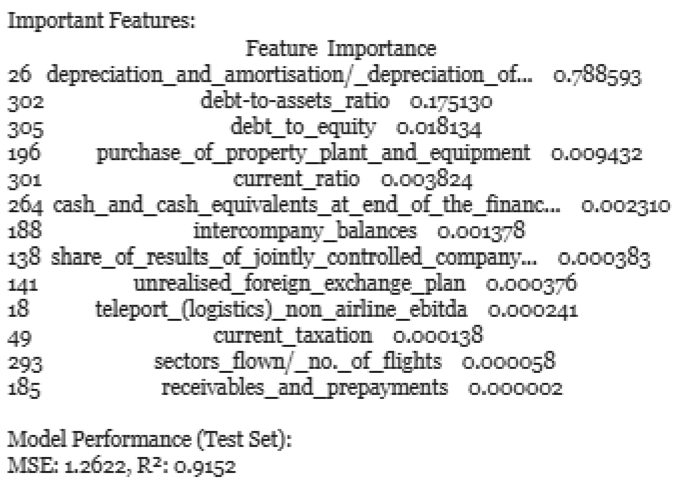
Show hidden output

APPENDIX

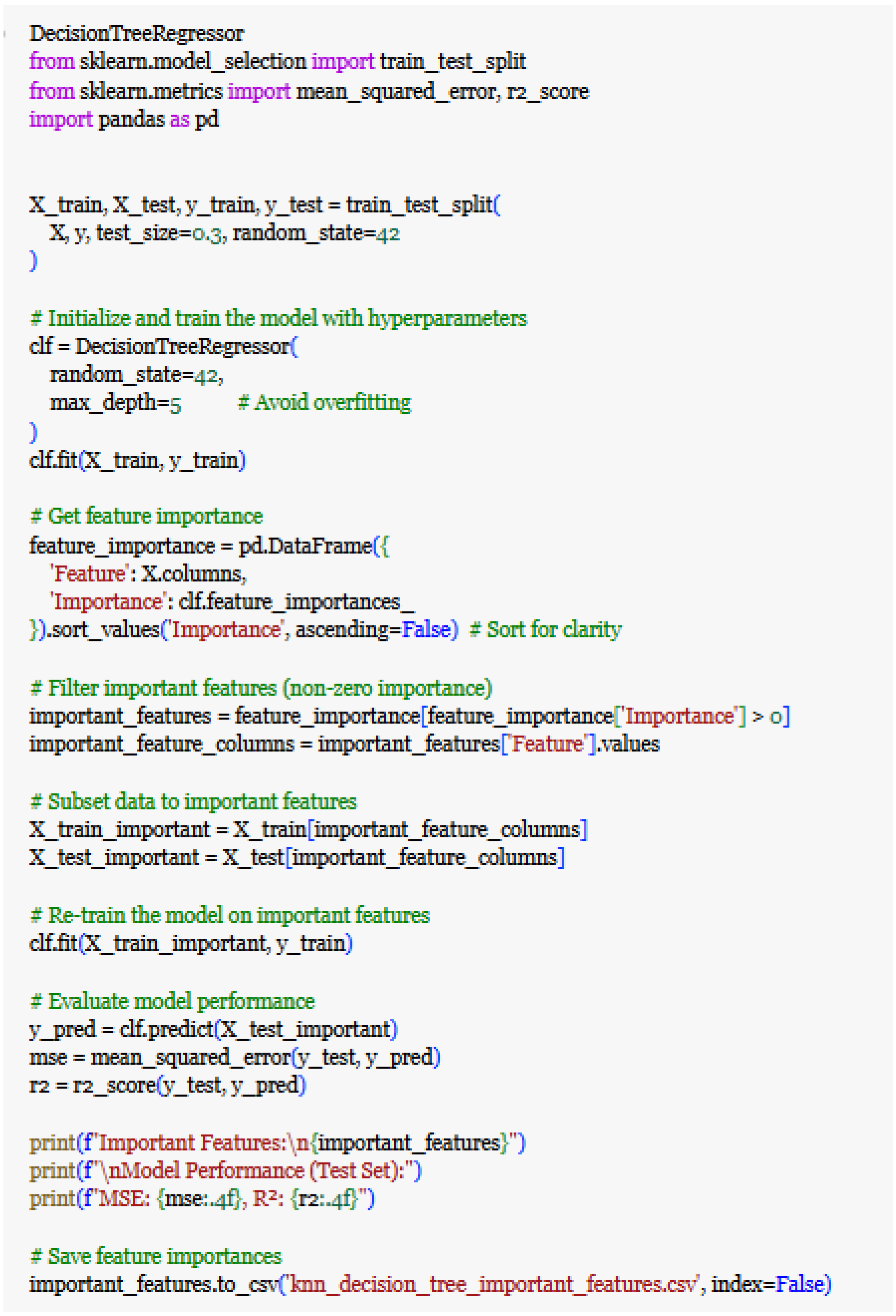


```
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
import pandas as pd

X_train, X_test, y_train, y_test = train_test_split(
   X, y, test_size=0.3, random_state=42
)

# Initialize and train the model with hyperparameters
clf = DecisionTreeRegressor(
   random_state=42,
   max_depth=5        # Avoid overfitting
)
clf.fit(X_train, y_train)

# Get feature importance
feature_importance = pd.DataFrame({
   'Feature': X.columns,
   'Importance': clf.feature_importances_
}).sort_values('Importance', ascending=False)  # Sort for clarity

# Filter important features (non-zero importance)
important_features = feature_importance[feature_importance['Importance'] > 0]
important_feature_columns = important_features['Feature'].values

# Subset data to important features
X_train_important = X_train[important_feature_columns]
X_test_important = X_test[important_feature_columns]

# Re-train the model on important features
clf.fit(X_train_important, y_train)

# Evaluate model performance
y_pred = clf.predict(X_test_important)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Important Features:\n{important_features}")
print(f"\nModel Performance (Test Set):")
print(f"MSE: {mse:.4f}, R²: {r2:.4f}")

# Save feature importances
important_features.to_csv('knn_decision_tree_important_features.csv', index=False)
```

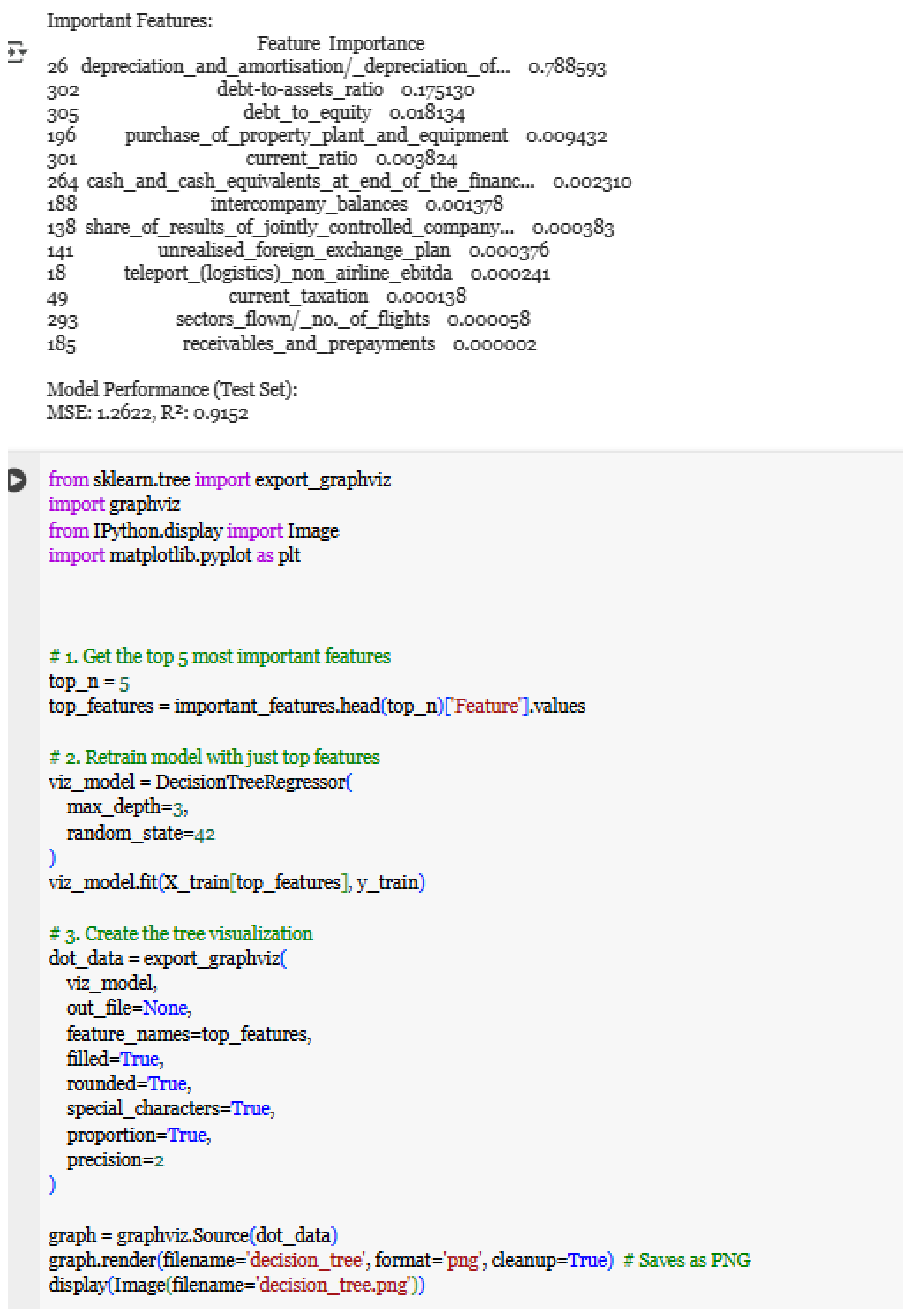
APPENDIX



```
Important Features:
                              Feature  Importance
26   depreciation_and_amortisation/_depreciation_of...    0.788593
302                          debt-to-assets_ratio    0.175130
305                               debt_to_equity    0.018134
196        purchase_of_property_plant_and_equipment    0.009432
301                                current_ratio    0.003824
264  cash_and_cash_equivalents_at_end_of_the_financ...    0.002310
188                         intercompany_balances    0.001378
138  share_of_results_of_jointly_controlled_company...    0.000383
141               unrealised_foreign_exchange_plan    0.000376
18          teleport_(logistics)_non_airline_ebitda    0.000241
49                              current_taxation    0.000138
293                    sectors_flown/_no._of_flights    0.000058
185                    receivables_and_prepayments    0.000002

Model Performance (Test Set):
MSE: 1.2622, R²: 0.9152
```

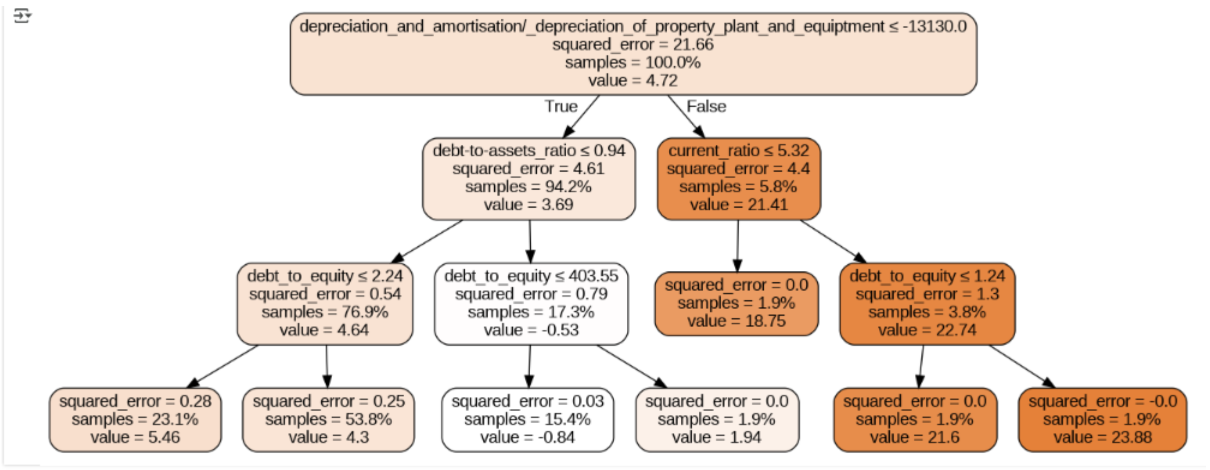


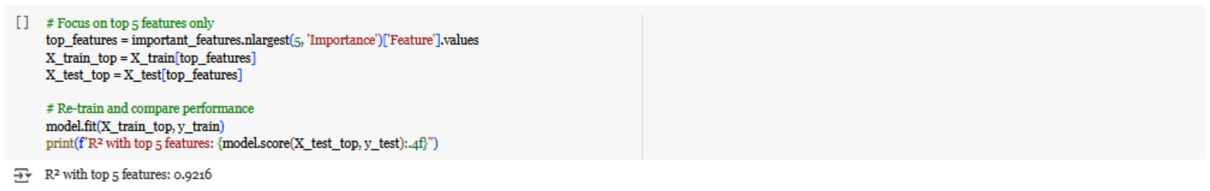
```
DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
import pandas as pd


X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)

# Initialize and train the model with hyperparameters
clf = DecisionTreeRegressor(
    random_state=42,
    max_depth=5         # Avoid overfitting
)
clf.fit(X_train, y_train)

# Get feature importance
feature_importance = pd.DataFrame({
    'Feature': X.columns,
    'Importance': clf.feature_importances_
}).sort_values('Importance', ascending=False)  # Sort for clarity

# Filter important features (non-zero importance)
important_features = feature_importance[feature_importance['Importance'] > 0]
important_feature_columns = important_features['Feature'].values

# Subset data to important features
X_train_important = X_train[important_feature_columns]
X_test_important = X_test[important_feature_columns]

# Re-train the model on important features
clf.fit(X_train_important, y_train)

# Evaluate model performance
y_pred = clf.predict(X_test_important)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Important Features:\n{important_features}")
print(f"\nModel Performance (Test Set):")
print(f"MSE: {mse:.4f}, R²: {r2:.4f}")

# Save feature importances
important_features.to_csv('knn_decision_tree_important_features.csv', index=False)
```



```
Important Features:
                                     Feature  Importance
26  depreciation_and_amortisation/_depreciation_of...    0.788593
302                       debt-to-assets_ratio    0.175130
305                             debt_to_equity    0.018134
196          purchase_of_property_plant_and_equipment    0.009432
301                              current_ratio    0.003824
264 cash_and_cash_equivalents_at_end_of_the_financ...    0.002310
188                      intercompany_balances    0.001378
138 share_of_results_of_jointly_controlled_company...    0.000383
141               unrealised_foreign_exchange_plan    0.000376
18          teleport_(logistics)_non_airline_ebitda    0.000241
49                             current_taxation    0.000138
293                  sectors_flown/_no._of_flights    0.000058
185                    receivables_and_prepayments    0.000002

Model Performance (Test Set):
MSE: 1.2622, R²: 0.9152
```

```python
from sklearn.tree import export_graphviz
import graphviz
from IPython.display import Image
import matplotlib.pyplot as plt



# 1. Get the top 5 most important features
top_n = 5
top_features = important_features.head(top_n)['Feature'].values

# 2. Retrain model with just top features
viz_model = DecisionTreeRegressor(
   max_depth=3,
   random_state=42
)
viz_model.fit(X_train[top_features], y_train)

# 3. Create the tree visualization
dot_data = export_graphviz(
   viz_model,
   out_file=None,
   feature_names=top_features,
   filled=True,
   rounded=True,
   special_characters=True,
   proportion=True,
   precision=2
)

graph = graphviz.Source(dot_data)
graph.render(filename='decision_tree', format='png', cleanup=True)  # Saves as PNG
display(Image(filename='decision_tree.png'))
```

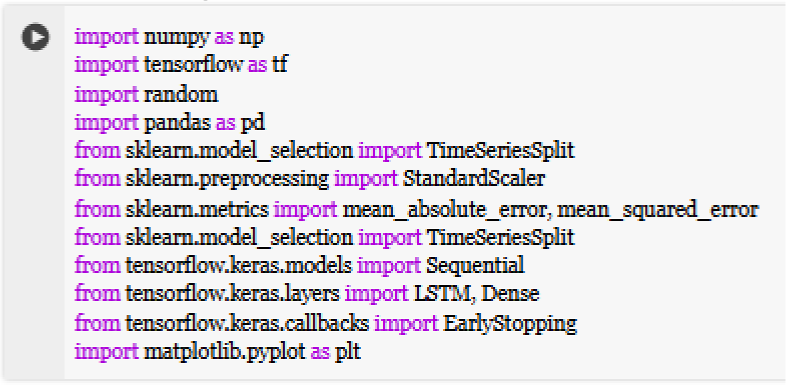




```
# Focus on top 5 features only
top_features = important_features.nlargest(5, 'Importance')['Feature'].values
X_train_top = X_train[top_features]
X_test_top = X_test[top_features]

# Re-train and compare performance
model.fit(X_train_top, y_train)
print(f"R² with top 5 features: {model.score(X_test_top, y_test):.4f}")
```

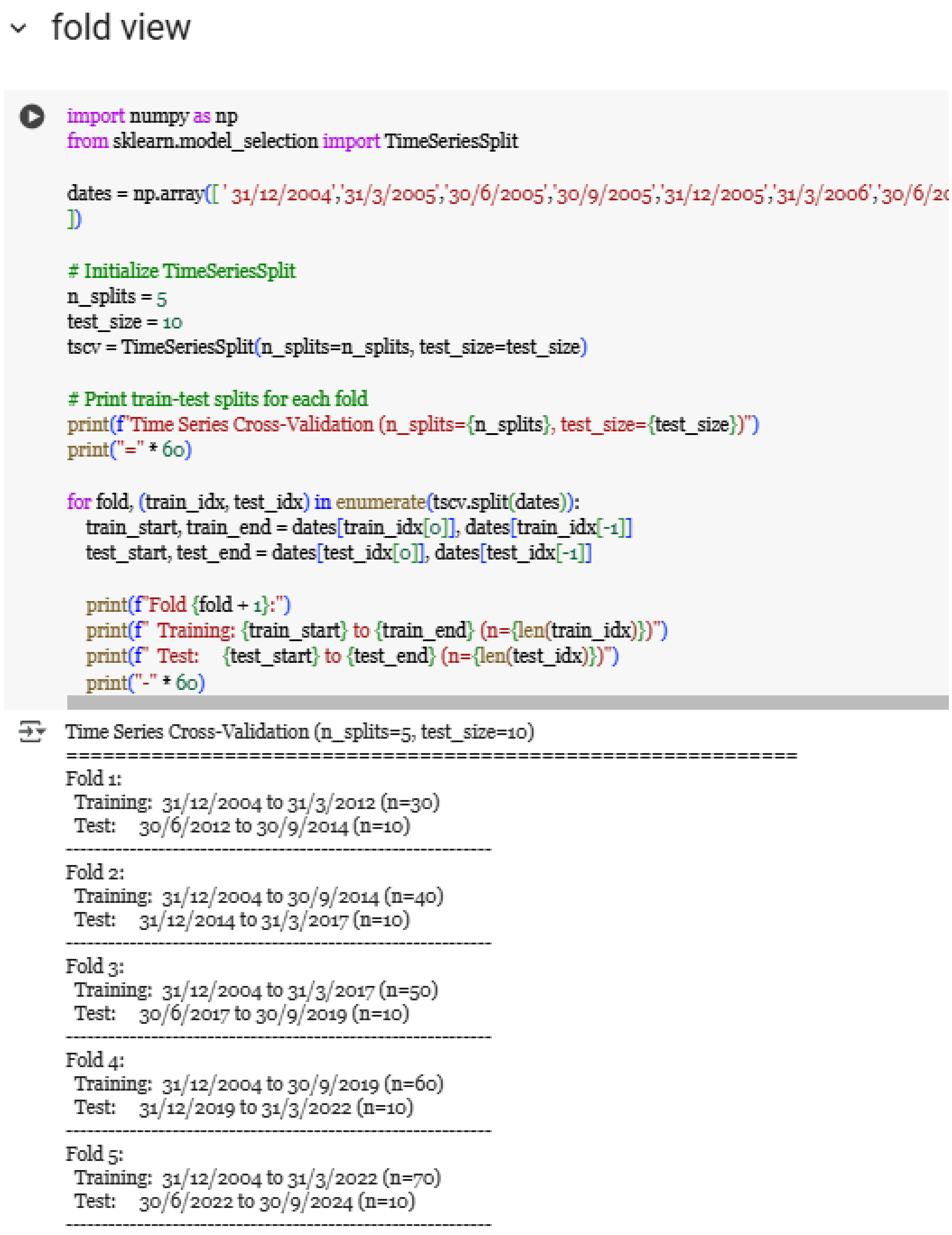
```
R² with top 5 features: 0.9216
```

# Code for using FCMI inputted data as a finance-only LSTM model



```
import numpy as np
import tensorflow as tf
import random
import pandas as pd
from sklearn.model_selection import TimeSeriesSplit
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn.model_selection import TimeSeriesSplit
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from tensorflow.keras.callbacks import EarlyStopping
import matplotlib.pyplot as plt
```

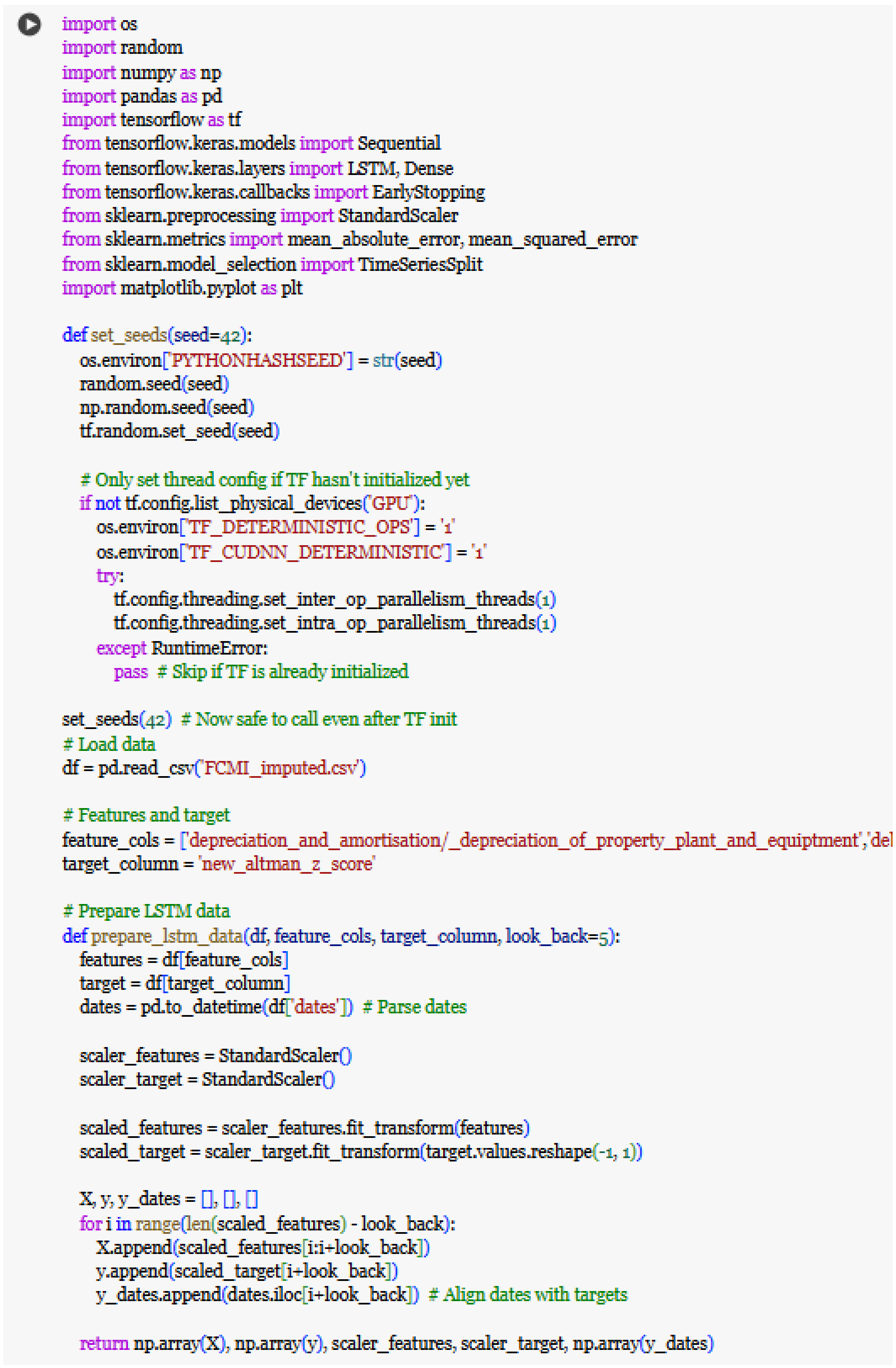
APPENDIX



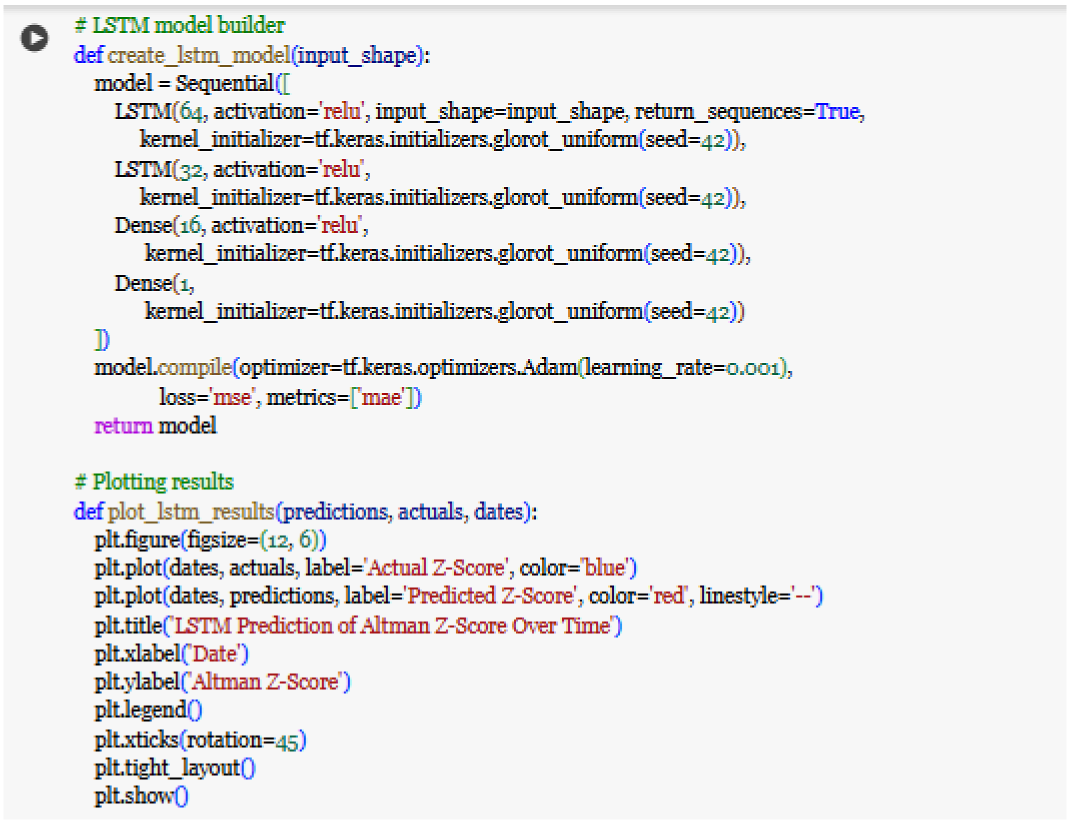
fold view

```
import numpy as np
from sklearn.model_selection import TimeSeriesSplit

dates = np.array([' 31/12/2004','31/3/2005','30/6/2005','30/9/2005','31/12/2005','31/3/2006','30/6/2
])

# Initialize TimeSeriesSplit
n_splits = 5
test_size = 10
tscv = TimeSeriesSplit(n_splits=n_splits, test_size=test_size)

# Print train-test splits for each fold
print(f"Time Series Cross-Validation (n_splits={n_splits}, test_size={test_size})")
print("=" * 60)

for fold, (train_idx, test_idx) in enumerate(tscv.split(dates)):
    train_start, train_end = dates[train_idx[0]], dates[train_idx[-1]]
    test_start, test_end = dates[test_idx[0]], dates[test_idx[-1]]

    print(f"Fold {fold + 1}:")
    print(f"  Training: {train_start} to {train_end} (n={len(train_idx)})")
    print(f"  Test:     {test_start} to {test_end} (n={len(test_idx)})")
    print("-" * 60)
```

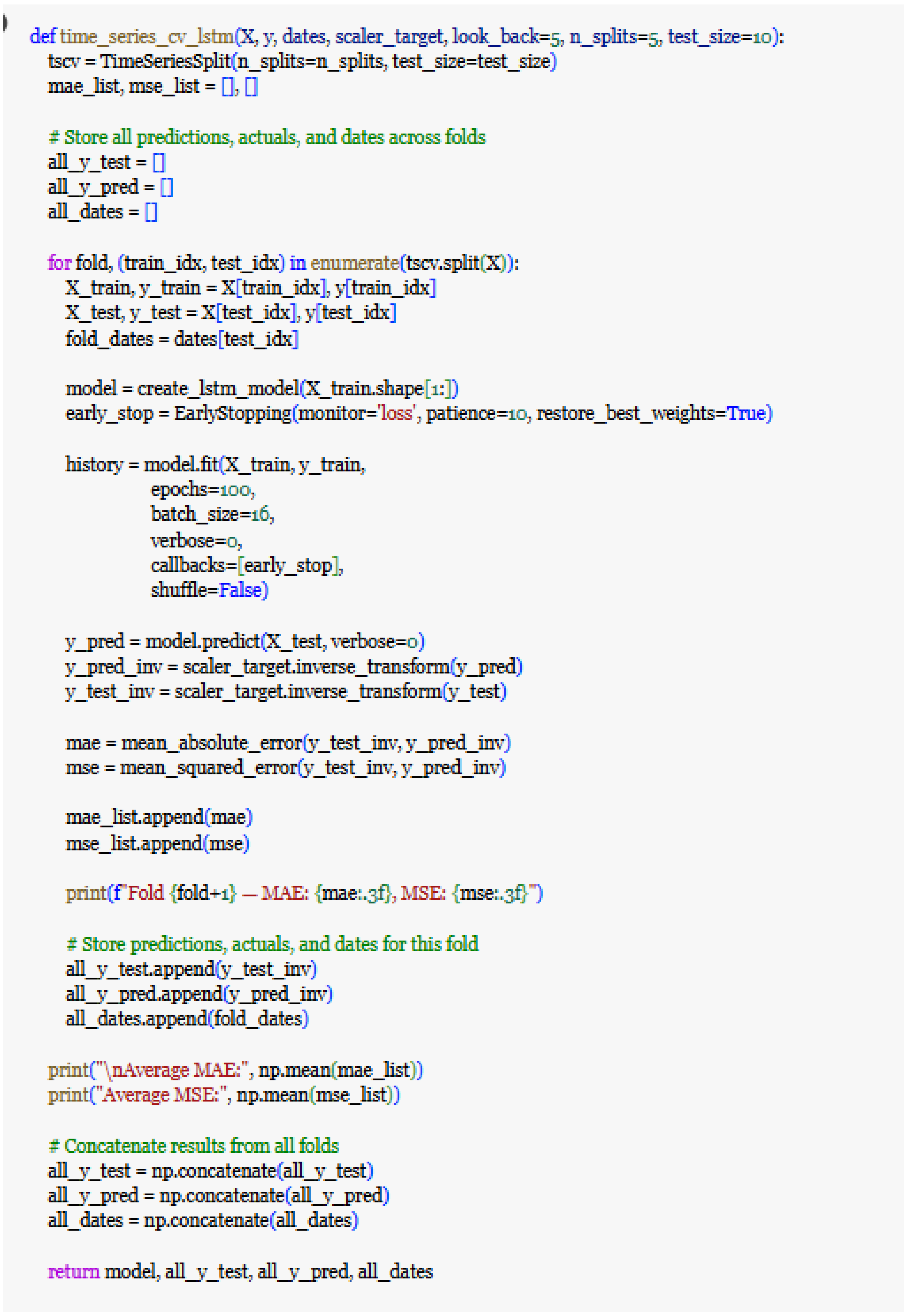
```
Time Series Cross-Validation (n_splits=5, test_size=10)
============================================================
Fold 1:
  Training:  31/12/2004 to 31/3/2012 (n=30)
  Test:     30/6/2012 to 30/9/2014 (n=10)
------------------------------------------------------------
Fold 2:
  Training:  31/12/2004 to 30/9/2014 (n=40)
  Test:     31/12/2014 to 31/3/2017 (n=10)
------------------------------------------------------------
Fold 3:
  Training:  31/12/2004 to 31/3/2017 (n=50)
  Test:     30/6/2017 to 30/9/2019 (n=10)
------------------------------------------------------------
Fold 4:
  Training:  31/12/2004 to 30/9/2019 (n=60)
  Test:     31/12/2019 to 31/3/2022 (n=10)
------------------------------------------------------------
Fold 5:
  Training:  31/12/2004 to 31/3/2022 (n=70)
  Test:     30/6/2022 to 30/9/2024 (n=10)
------------------------------------------------------------
```



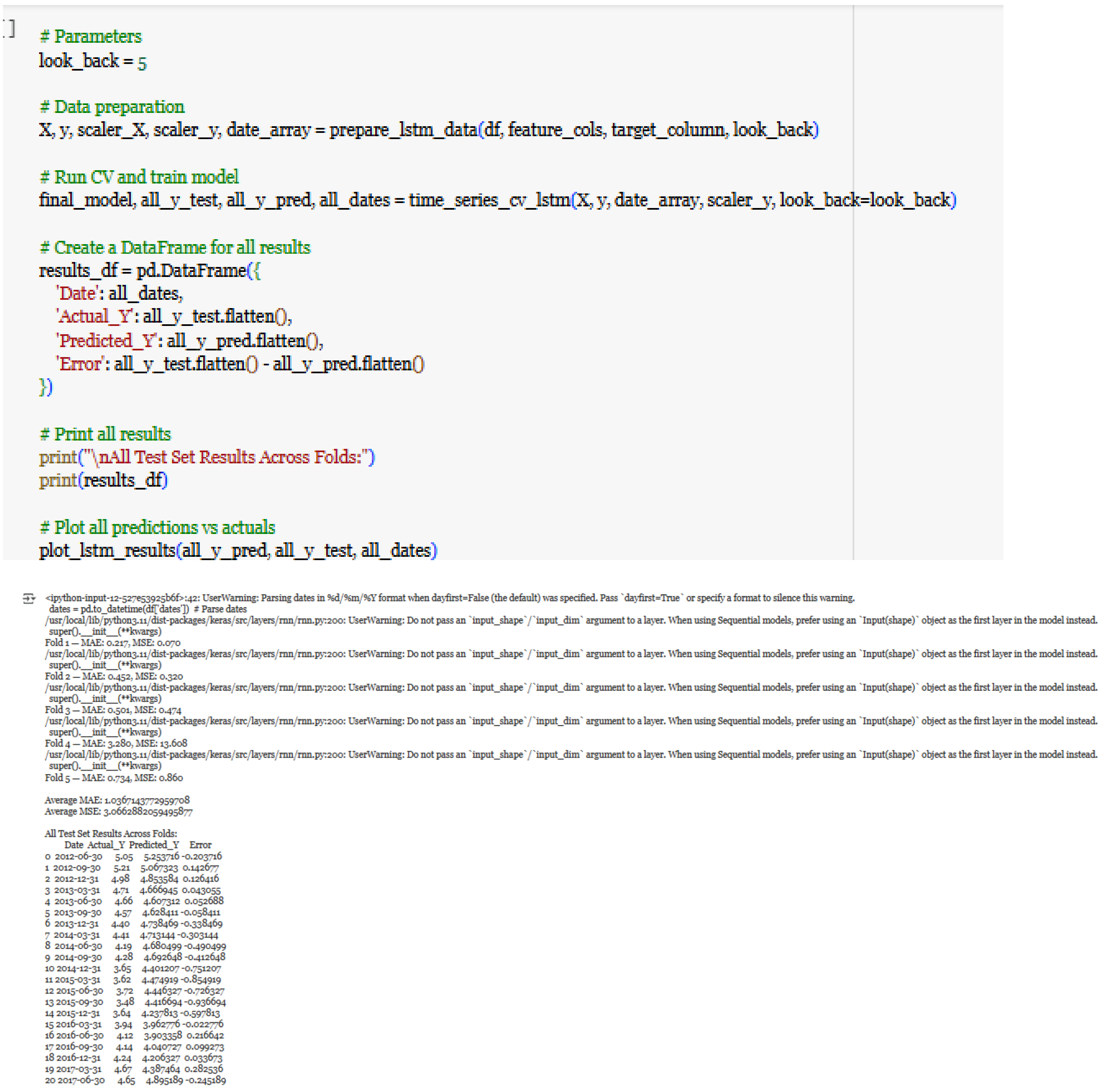
```python
import os
import random
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn.model_selection import TimeSeriesSplit
import matplotlib.pyplot as plt

def set_seeds(seed=42):
    os.environ['PYTHONHASHSEED'] = str(seed)
    random.seed(seed)
    np.random.seed(seed)
    tf.random.set_seed(seed)

    # Only set thread config if TF hasn't initialized yet
    if not tf.config.list_physical_devices('GPU'):
        os.environ['TF_DETERMINISTIC_OPS'] = '1'
        os.environ['TF_CUDNN_DETERMINISTIC'] = '1'
        try:
            tf.config.threading.set_inter_op_parallelism_threads(1)
            tf.config.threading.set_intra_op_parallelism_threads(1)
        except RuntimeError:
            pass  # Skip if TF is already initialized

set_seeds(42)  # Now safe to call even after TF init
# Load data
df = pd.read_csv('FCMI_imputed.csv')

# Features and target
feature_cols = ['depreciation_and_amortisation/_depreciation_of_property_plant_and_equiptment','del
target_column = 'new_altman_z_score'

# Prepare LSTM data
def prepare_lstm_data(df, feature_cols, target_column, look_back=5):
    features = df[feature_cols]
    target = df[target_column]
    dates = pd.to_datetime(df['dates'])  # Parse dates

    scaler_features = StandardScaler()
    scaler_target = StandardScaler()

    scaled_features = scaler_features.fit_transform(features)
    scaled_target = scaler_target.fit_transform(target.values.reshape(-1, 1))

    X, y, y_dates = [], [], []
    for i in range(len(scaled_features) - look_back):
        X.append(scaled_features[i:i+look_back])
        y.append(scaled_target[i+look_back])
        y_dates.append(dates.iloc[i+look_back])  # Align dates with targets

    return np.array(X), np.array(y), scaler_features, scaler_target, np.array(y_dates)
```



```
# LSTM model builder
def create_lstm_model(input_shape):
  model = Sequential([
    LSTM(64, activation='relu', input_shape=input_shape, return_sequences=True,
       kernel_initializer=tf.keras.initializers.glorot_uniform(seed=42)),
    LSTM(32, activation='relu',
       kernel_initializer=tf.keras.initializers.glorot_uniform(seed=42)),
    Dense(16, activation='relu',
        kernel_initializer=tf.keras.initializers.glorot_uniform(seed=42)),
    Dense(1,
        kernel_initializer=tf.keras.initializers.glorot_uniform(seed=42))
  ])
  model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
          loss='mse', metrics=['mae'])
  return model

# Plotting results
def plot_lstm_results(predictions, actuals, dates):
  plt.figure(figsize=(12, 6))
  plt.plot(dates, actuals, label='Actual Z-Score', color='blue')
  plt.plot(dates, predictions, label='Predicted Z-Score', color='red', linestyle='--')
  plt.title('LSTM Prediction of Altman Z-Score Over Time')
  plt.xlabel('Date')
  plt.ylabel('Altman Z-Score')
  plt.legend()
  plt.xticks(rotation=45)
  plt.tight_layout()
  plt.show()
```



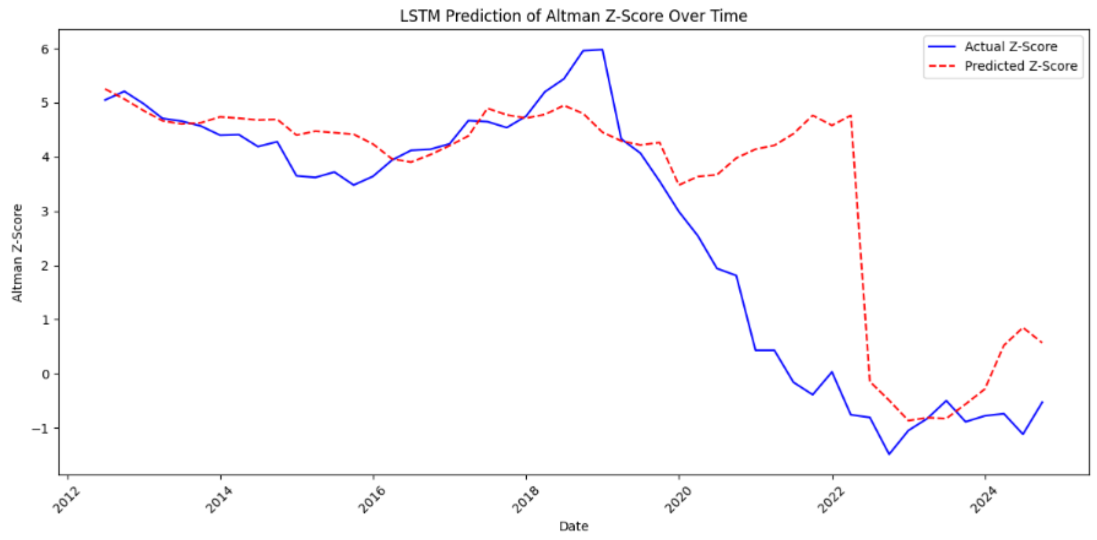
```
def time_series_cv_lstm(X, y, dates, scaler_target, look_back=5, n_splits=5, test_size=10):
    tscv = TimeSeriesSplit(n_splits=n_splits, test_size=test_size)
    mae_list, mse_list = [], []

    # Store all predictions, actuals, and dates across folds
    all_y_test = []
    all_y_pred = []
    all_dates = []

    for fold, (train_idx, test_idx) in enumerate(tscv.split(X)):
        X_train, y_train = X[train_idx], y[train_idx]
        X_test, y_test = X[test_idx], y[test_idx]
        fold_dates = dates[test_idx]

        model = create_lstm_model(X_train.shape[1:])
        early_stop = EarlyStopping(monitor='loss', patience=10, restore_best_weights=True)

        history = model.fit(X_train, y_train,
                            epochs=100,
                            batch_size=16,
                            verbose=0,
                            callbacks=[early_stop],
                            shuffle=False)

        y_pred = model.predict(X_test, verbose=0)
        y_pred_inv = scaler_target.inverse_transform(y_pred)
        y_test_inv = scaler_target.inverse_transform(y_test)

        mae = mean_absolute_error(y_test_inv, y_pred_inv)
        mse = mean_squared_error(y_test_inv, y_pred_inv)

        mae_list.append(mae)
        mse_list.append(mse)

        print(f"Fold {fold+1} — MAE: {mae:.3f}, MSE: {mse:.3f}")

        # Store predictions, actuals, and dates for this fold
        all_y_test.append(y_test_inv)
        all_y_pred.append(y_pred_inv)
        all_dates.append(fold_dates)

    print("\nAverage MAE:", np.mean(mae_list))
    print("Average MSE:", np.mean(mse_list))

    # Concatenate results from all folds
    all_y_test = np.concatenate(all_y_test)
    all_y_pred = np.concatenate(all_y_pred)
    all_dates = np.concatenate(all_dates)

    return model, all_y_test, all_y_pred, all_dates
```



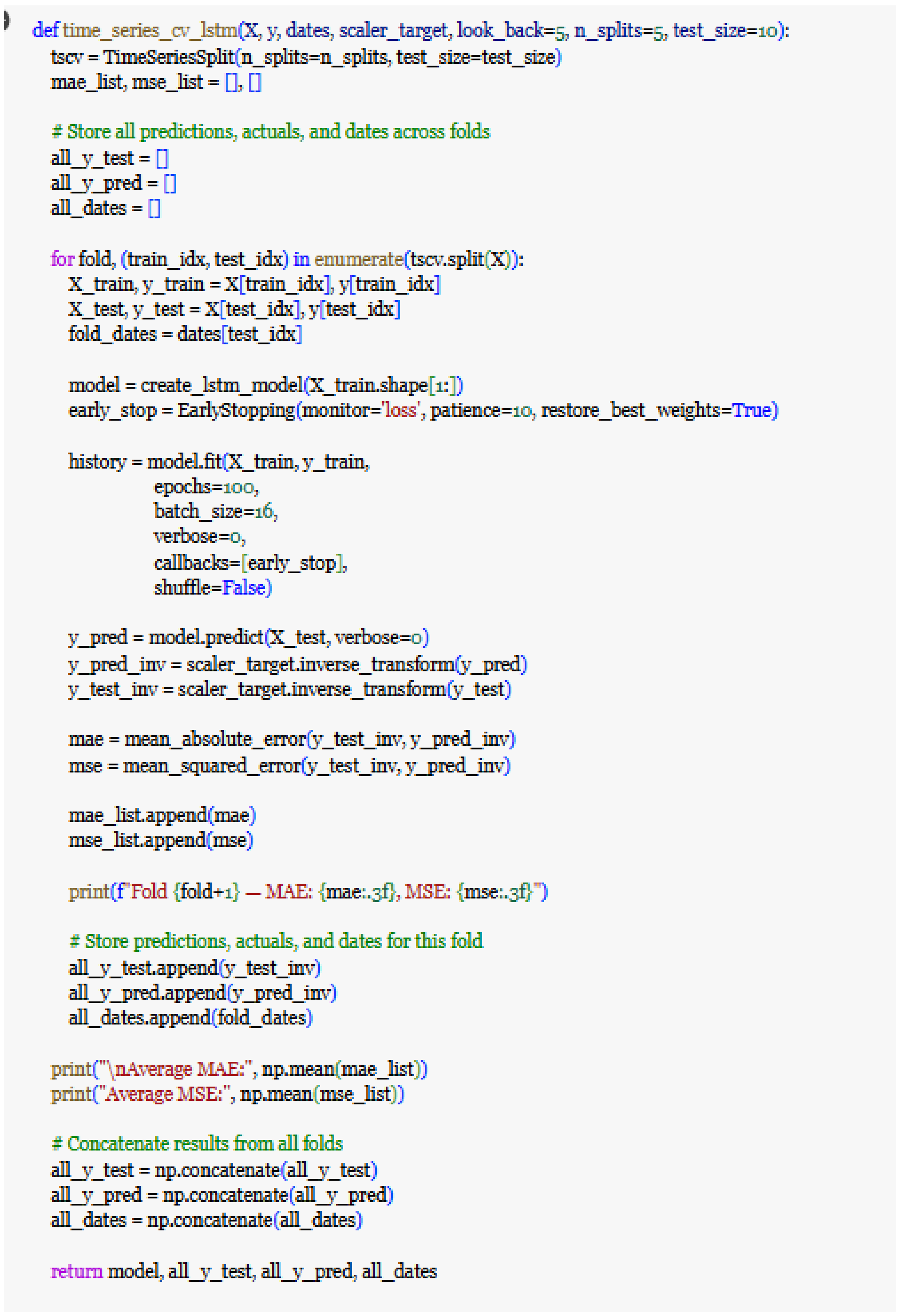
```python
# Parameters
look_back = 5

# Data preparation
X, y, scaler_X, scaler_y, date_array = prepare_lstm_data(df, feature_cols, target_column, look_back)

# Run CV and train model
final_model, all_y_test, all_y_pred, all_dates = time_series_cv_lstm(X, y, date_array, scaler_y, look_back=look_back)

# Create a DataFrame for all results
results_df = pd.DataFrame({
    'Date': all_dates,
    'Actual_Y': all_y_test.flatten(),
    'Predicted_Y': all_y_pred.flatten(),
    'Error': all_y_test.flatten() - all_y_pred.flatten()
})

# Print all results
print("\nAll Test Set Results Across Folds:")
print(results_df)

# Plot all predictions vs actuals
plot_lstm_results(all_y_pred, all_y_test, all_dates)
```

```
<ipython-input-12-527e53925b6f>:42: UserWarning: Parsing dates in %d/%m/%Y format when dayfirst=False (the default) was specified. Pass `dayfirst=True` or specify a format to silence this warning.
  dates = pd.to_datetime(df['dates'])  # Parse dates
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)
Fold 1 — MAE: 0.217, MSE: 0.070
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)
Fold 2 — MAE: 0.452, MSE: 0.320
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)
Fold 3 — MAE: 0.501, MSE: 0.474
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)
Fold 4 — MAE: 3.280, MSE: 13.608
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)
Fold 5 — MAE: 0.734, MSE: 0.860

Average MAE: 1.0367143772959708
Average MSE: 3.0662882059495877

All Test Set Results Across Folds:
         Date  Actual_Y  Predicted_Y     Error
0  2012-06-30      5.05     5.253716 -0.203716
1  2012-09-30      5.21     5.067323  0.142677
2  2012-12-31      4.98     4.853584  0.126416
3  2013-03-31      4.71     4.666945  0.043055
4  2013-06-30      4.66     4.607312  0.052688
5  2013-09-30      4.57     4.628411 -0.058411
6  2013-12-31      4.40     4.738469 -0.338469
7  2014-03-31      4.41     4.713144 -0.303144
8  2014-06-30      4.19     4.680499 -0.490499
9  2014-09-30      4.28     4.692648 -0.412648
10 2014-12-31      3.65     4.401207 -0.751207
11 2015-03-31      3.62     4.474919 -0.854919
12 2015-06-30      3.72     4.446327 -0.726327
13 2015-09-30      3.48     4.416694 -0.936694
14 2015-12-31      3.64     4.237813 -0.597813
15 2016-03-31      3.94     3.962776 -0.022776
16 2016-06-30      4.12     3.903358  0.216642
17 2016-09-30      4.14     4.040727  0.099273
18 2016-12-31      4.24     4.206327  0.033673
19 2017-03-31      4.67     4.387464  0.282536
20 2017-06-30      4.65     4.895189 -0.245189
```

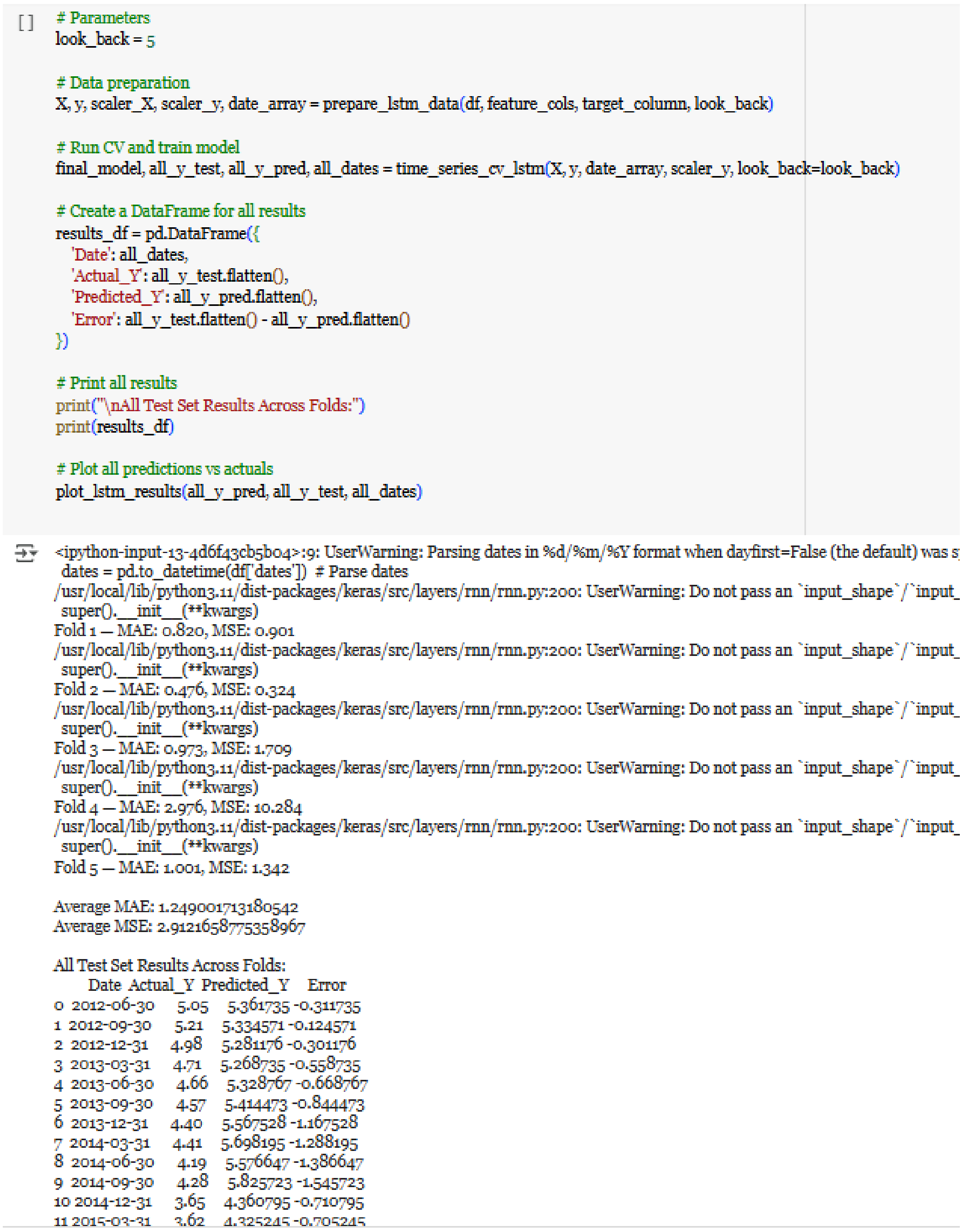
```
48 2024-06-30   -1.12   0.849950 -1.969950
49 2024-09-30   -0.53   0.567583 -1.097583
```



```python
# Features and target
feature_cols =['depreciation_and_amortisation/_depreciation_of_property_plant_and_equiptment','debt-to-assets_ratio','debt_to_equity','purchase_of_property_plant_and_equipment','current_ratio']
target_column = 'new_altman_z_score'

# Prepare LSTM data
def prepare_lstm_data(df, feature_cols, target_column, look_back=5):
  features = df[feature_cols]
  target = df[target_column]
  dates = pd.to_datetime(df['dates'])  # Parse dates

  scaler_features = StandardScaler()
  scaler_target = StandardScaler()

  scaled_features = scaler_features.fit_transform(features)
  scaled_target = scaler_target.fit_transform(target.values.reshape(-1, 1))

  X, y, y_dates = [], [], []
  for i in range(len(scaled_features) - look_back):
    X.append(scaled_features[i:i+look_back])
    y.append(scaled_target[i+look_back])
    y_dates.append(dates.iloc[i+look_back])  # Align dates with targets

  return np.array(X), np.array(y), scaler_features, scaler_target, np.array(y_dates)

# LSTM model builder
def create_lstm_model(input_shape):
  model = Sequential([
    LSTM(64, activation='relu', input_shape=input_shape, return_sequences=True,
       kernel_initializer=tf.keras.initializers.glorot_uniform(seed=42)),
    LSTM(32, activation='relu',
       kernel_initializer=tf.keras.initializers.glorot_uniform(seed=42)),
    Dense(16, activation='relu',
        kernel_initializer=tf.keras.initializers.glorot_uniform(seed=42)),
    Dense(1,
        kernel_initializer=tf.keras.initializers.glorot_uniform(seed=42))
  ])
  model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
          loss='mse', metrics=['mae'])
  return model

# Plotting results
def plot_lstm_results(predictions, actuals, dates):
  plt.figure(figsize=(12, 6))
  plt.plot(dates, actuals, label='Actual Z-Score', color='blue')
  plt.plot(dates, predictions, label='Predicted Z-Score', color='red', linestyle='--')
  plt.title('LSTM Prediction of Altman Z-Score Over Time')
  plt.xlabel('Date')
  plt.ylabel('Altman Z-Score')
  plt.legend()
  plt.xticks(rotation=45)
  plt.tight_layout()
  plt.show()
```

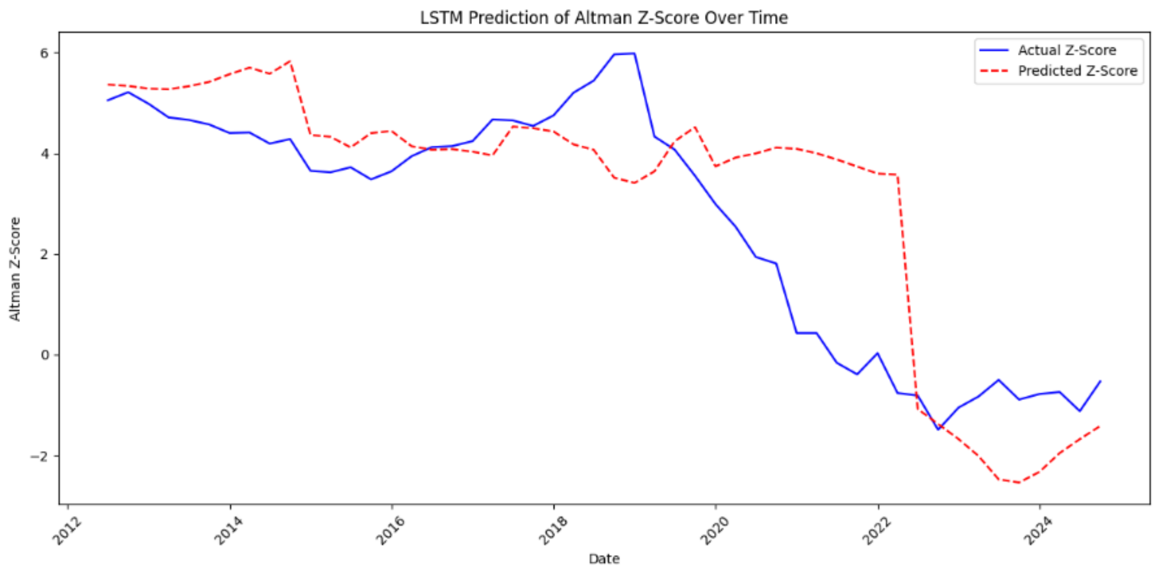


```
def time_series_cv_lstm(X, y, dates, scaler_target, look_back=5, n_splits=5, test_size=10):
    tscv = TimeSeriesSplit(n_splits=n_splits, test_size=test_size)
    mae_list, mse_list = [], []

    # Store all predictions, actuals, and dates across folds
    all_y_test = []
    all_y_pred = []
    all_dates = []

    for fold, (train_idx, test_idx) in enumerate(tscv.split(X)):
        X_train, y_train = X[train_idx], y[train_idx]
        X_test, y_test = X[test_idx], y[test_idx]
        fold_dates = dates[test_idx]

        model = create_lstm_model(X_train.shape[1:])
        early_stop = EarlyStopping(monitor='loss', patience=10, restore_best_weights=True)

        history = model.fit(X_train, y_train,
                            epochs=100,
                            batch_size=16,
                            verbose=0,
                            callbacks=[early_stop],
                            shuffle=False)

        y_pred = model.predict(X_test, verbose=0)
        y_pred_inv = scaler_target.inverse_transform(y_pred)
        y_test_inv = scaler_target.inverse_transform(y_test)

        mae = mean_absolute_error(y_test_inv, y_pred_inv)
        mse = mean_squared_error(y_test_inv, y_pred_inv)

        mae_list.append(mae)
        mse_list.append(mse)

        print(f"Fold {fold+1} — MAE: {mae:.3f}, MSE: {mse:.3f}")

        # Store predictions, actuals, and dates for this fold
        all_y_test.append(y_test_inv)
        all_y_pred.append(y_pred_inv)
        all_dates.append(fold_dates)

    print("\nAverage MAE:", np.mean(mae_list))
    print("Average MSE:", np.mean(mse_list))

    # Concatenate results from all folds
    all_y_test = np.concatenate(all_y_test)
    all_y_pred = np.concatenate(all_y_pred)
    all_dates = np.concatenate(all_dates)

    return model, all_y_test, all_y_pred, all_dates
```

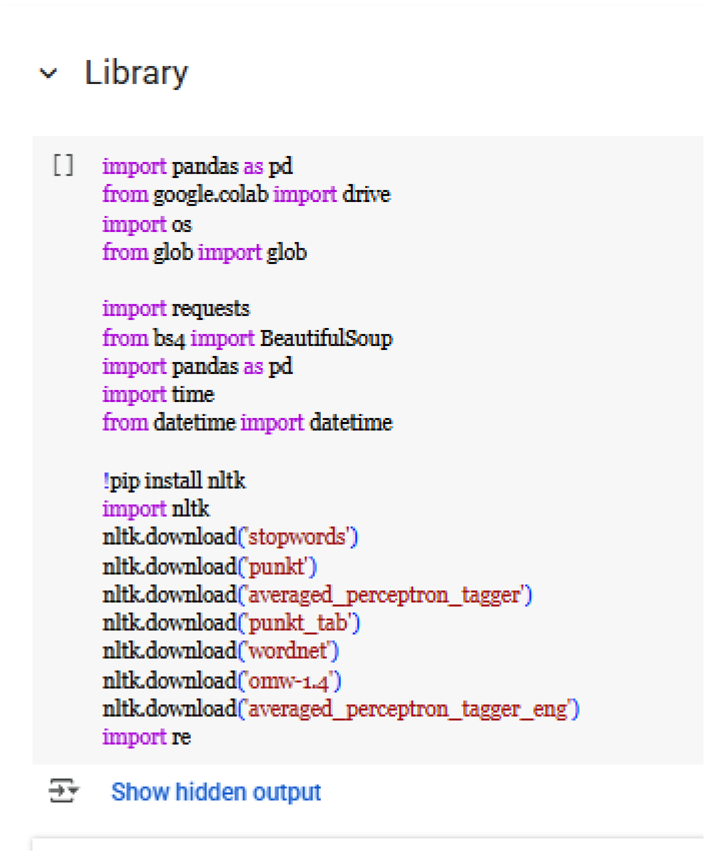


```
[ ] # Parameters
    look_back = 5

    # Data preparation
    X, y, scaler_X, scaler_y, date_array = prepare_lstm_data(df, feature_cols, target_column, look_back)

    # Run CV and train model
    final_model, all_y_test, all_y_pred, all_dates = time_series_cv_lstm(X, y, date_array, scaler_y, look_back=look_back)

    # Create a DataFrame for all results
    results_df = pd.DataFrame({
        'Date': all_dates,
        'Actual_Y': all_y_test.flatten(),
        'Predicted_Y': all_y_pred.flatten(),
        'Error': all_y_test.flatten() - all_y_pred.flatten()
    })

    # Print all results
    print("\nAll Test Set Results Across Folds:")
    print(results_df)

    # Plot all predictions vs actuals
    plot_lstm_results(all_y_pred, all_y_test, all_dates)
```

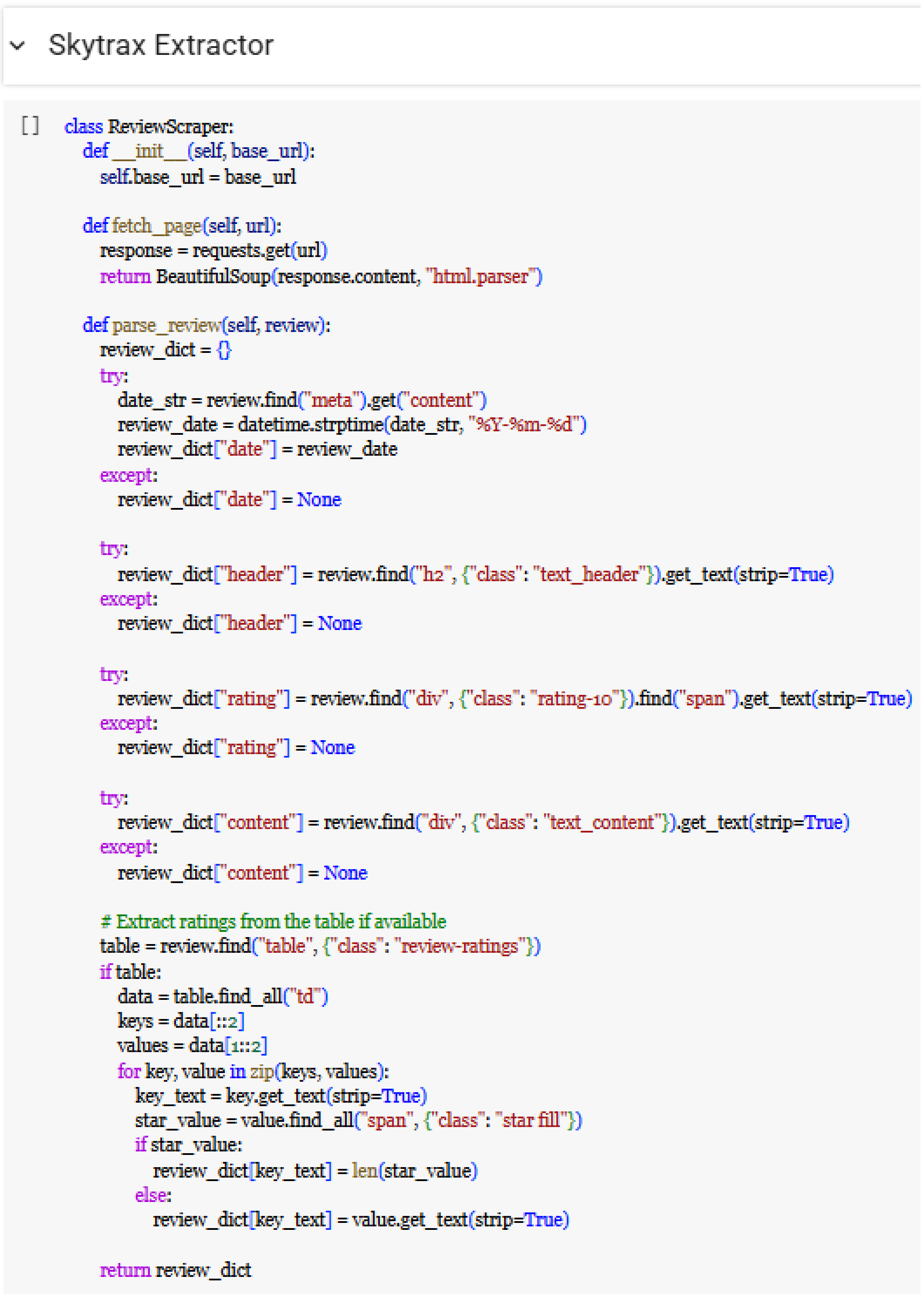
```
<ipython-input-13-4d6f43cb5b04>:9: UserWarning: Parsing dates in %d/%m/%Y format when dayfirst=False (the default) was s
  dates = pd.to_datetime(df['dates'])  # Parse dates
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarning: Do not pass an `input_shape`/`input_
  super().__init__(**kwargs)
Fold 1 — MAE: 0.820, MSE: 0.901
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarning: Do not pass an `input_shape`/`input_
  super().__init__(**kwargs)
Fold 2 — MAE: 0.476, MSE: 0.324
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarning: Do not pass an `input_shape`/`input_
  super().__init__(**kwargs)
Fold 3 — MAE: 0.973, MSE: 1.709
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarning: Do not pass an `input_shape`/`input_
  super().__init__(**kwargs)
Fold 4 — MAE: 2.976, MSE: 10.284
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarning: Do not pass an `input_shape`/`input_
  super().__init__(**kwargs)
Fold 5 — MAE: 1.001, MSE: 1.342

Average MAE: 1.249001713180542
Average MSE: 2.9121658775358967

All Test Set Results Across Folds:
         Date  Actual_Y  Predicted_Y     Error
0  2012-06-30      5.05     5.361735 -0.311735
1  2012-09-30      5.21     5.334571 -0.124571
2  2012-12-31      4.98     5.281176 -0.301176
3  2013-03-31      4.71     5.268735 -0.558735
4  2013-06-30      4.66     5.328767 -0.668767
5  2013-09-30      4.57     5.414473 -0.844473
6  2013-12-31      4.40     5.567528 -1.167528
7  2014-03-31      4.41     5.698195 -1.288195
8  2014-06-30      4.19     5.576647 -1.386647
9  2014-09-30      4.28     5.825723 -1.545723
10 2014-12-31      3.65     4.360795 -0.710795
11 2015-03-31      3.62     4.325245 -0.705245
```



# Code for sentimental analysis to create a text-only LSTM model

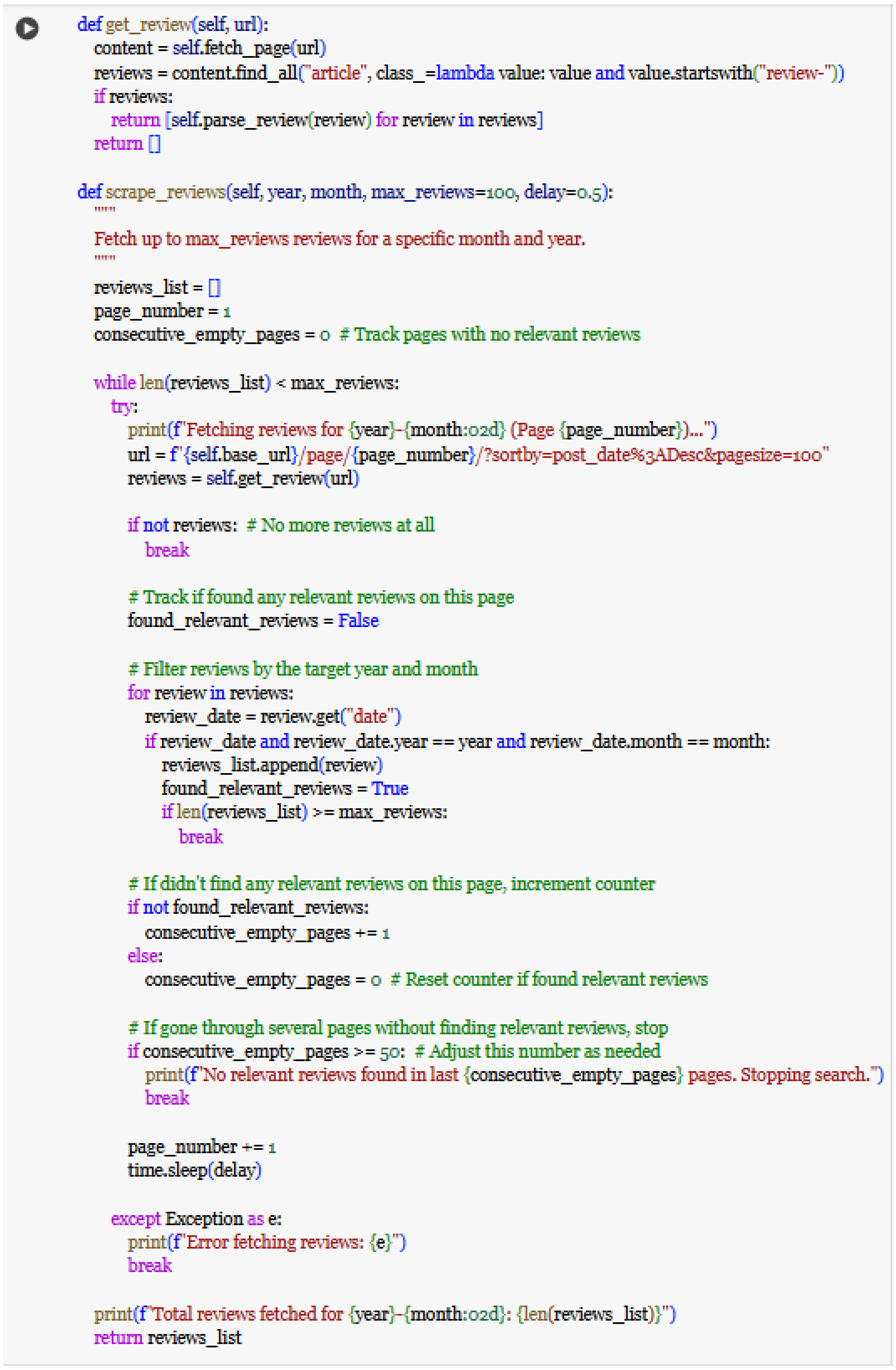


```
import pandas as pd
from google.colab import drive
import os
from glob import glob

import requests
from bs4 import BeautifulSoup
import pandas as pd
import time
from datetime import datetime

!pip install nltk
import nltk
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('punkt_tab')
nltk.download('wordnet')
nltk.download('omw-1.4')
nltk.download('averaged_perceptron_tagger_eng')
import re
```

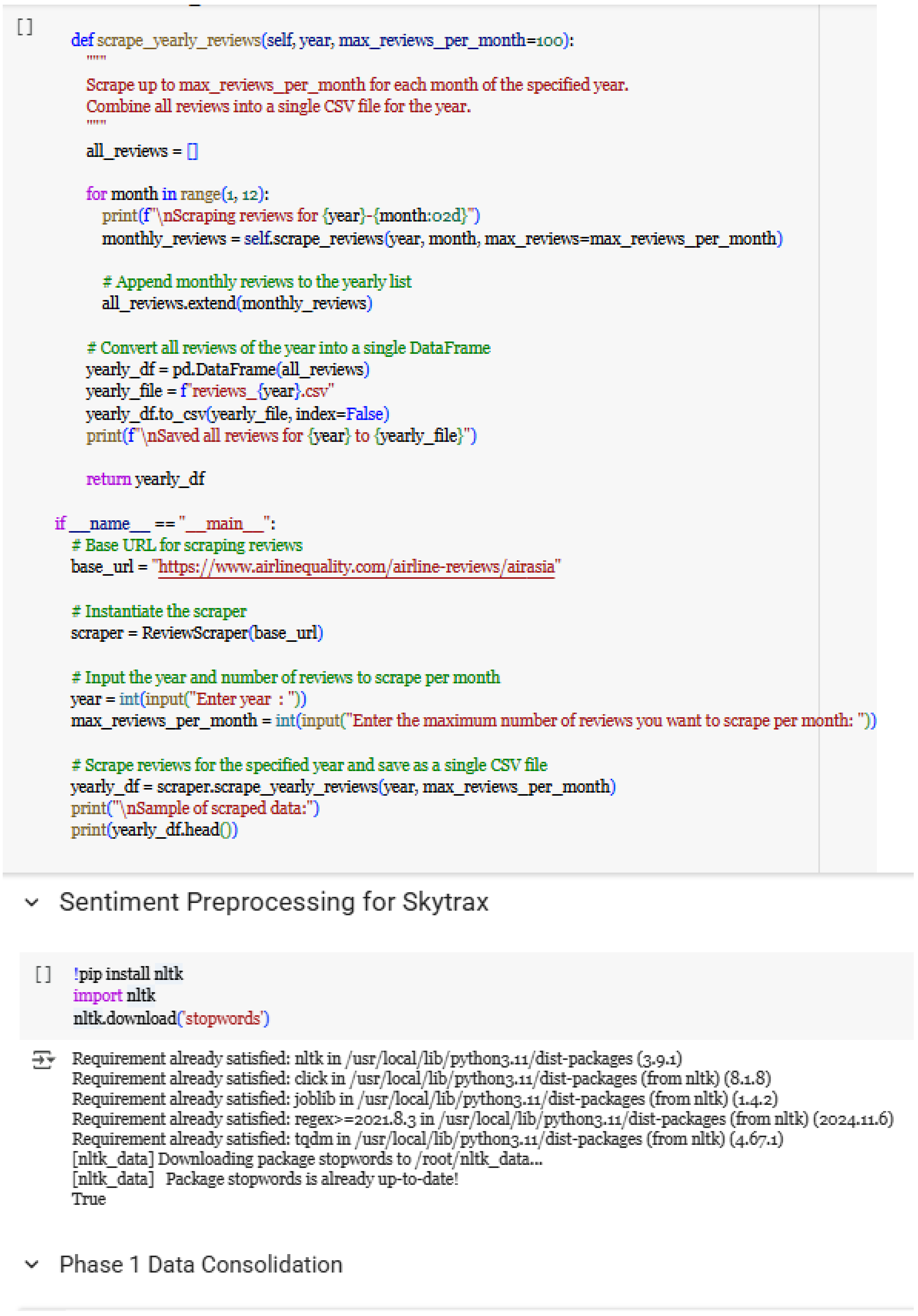


Skytrax Extractor

```
[ ]  class ReviewScraper:
         def __init__(self, base_url):
             self.base_url = base_url

         def fetch_page(self, url):
             response = requests.get(url)
             return BeautifulSoup(response.content, "html.parser")

         def parse_review(self, review):
             review_dict = {}
             try:
                 date_str = review.find("meta").get("content")
                 review_date = datetime.strptime(date_str, "%Y-%m-%d")
                 review_dict["date"] = review_date
             except:
                 review_dict["date"] = None

             try:
                 review_dict["header"] = review.find("h2", {"class": "text_header"}).get_text(strip=True)
             except:
                 review_dict["header"] = None

             try:
                 review_dict["rating"] = review.find("div", {"class": "rating-10"}).find("span").get_text(strip=True)
             except:
                 review_dict["rating"] = None

             try:
                 review_dict["content"] = review.find("div", {"class": "text_content"}).get_text(strip=True)
             except:
                 review_dict["content"] = None

             # Extract ratings from the table if available
             table = review.find("table", {"class": "review-ratings"})
             if table:
                 data = table.find_all("td")
                 keys = data[::2]
                 values = data[1::2]
                 for key, value in zip(keys, values):
                     key_text = key.get_text(strip=True)
                     star_value = value.find_all("span", {"class": "star fill"})
                     if star_value:
                         review_dict[key_text] = len(star_value)
                     else:
                         review_dict[key_text] = value.get_text(strip=True)

             return review_dict
```



```python
def get_review(self, url):
  content = self.fetch_page(url)
  reviews = content.find_all("article", class_=lambda value: value and value.startswith("review-"))
  if reviews:
    return [self.parse_review(review) for review in reviews]
  return []

def scrape_reviews(self, year, month, max_reviews=100, delay=0.5):
  """
  Fetch up to max_reviews reviews for a specific month and year.
  """
  reviews_list = []
  page_number = 1
  consecutive_empty_pages = 0  # Track pages with no relevant reviews

  while len(reviews_list) < max_reviews:
    try:
      print(f"Fetching reviews for {year}-{month:02d} (Page {page_number})...")
      url = f"{self.base_url}/page/{page_number}/?sortby=post_date%3ADesc&pagesize=100"
      reviews = self.get_review(url)

      if not reviews:  # No more reviews at all
        break

      # Track if found any relevant reviews on this page
      found_relevant_reviews = False

      # Filter reviews by the target year and month
      for review in reviews:
        review_date = review.get("date")
        if review_date and review_date.year == year and review_date.month == month:
          reviews_list.append(review)
          found_relevant_reviews = True
          if len(reviews_list) >= max_reviews:
            break

      # If didn't find any relevant reviews on this page, increment counter
      if not found_relevant_reviews:
        consecutive_empty_pages += 1
      else:
        consecutive_empty_pages = 0  # Reset counter if found relevant reviews

      # If gone through several pages without finding relevant reviews, stop
      if consecutive_empty_pages >= 50:  # Adjust this number as needed
        print(f"No relevant reviews found in last {consecutive_empty_pages} pages. Stopping search.")
        break

      page_number += 1
      time.sleep(delay)

    except Exception as e:
      print(f"Error fetching reviews: {e}")
      break

  print(f"Total reviews fetched for {year}-{month:02d}: {len(reviews_list)}")
  return reviews_list
```



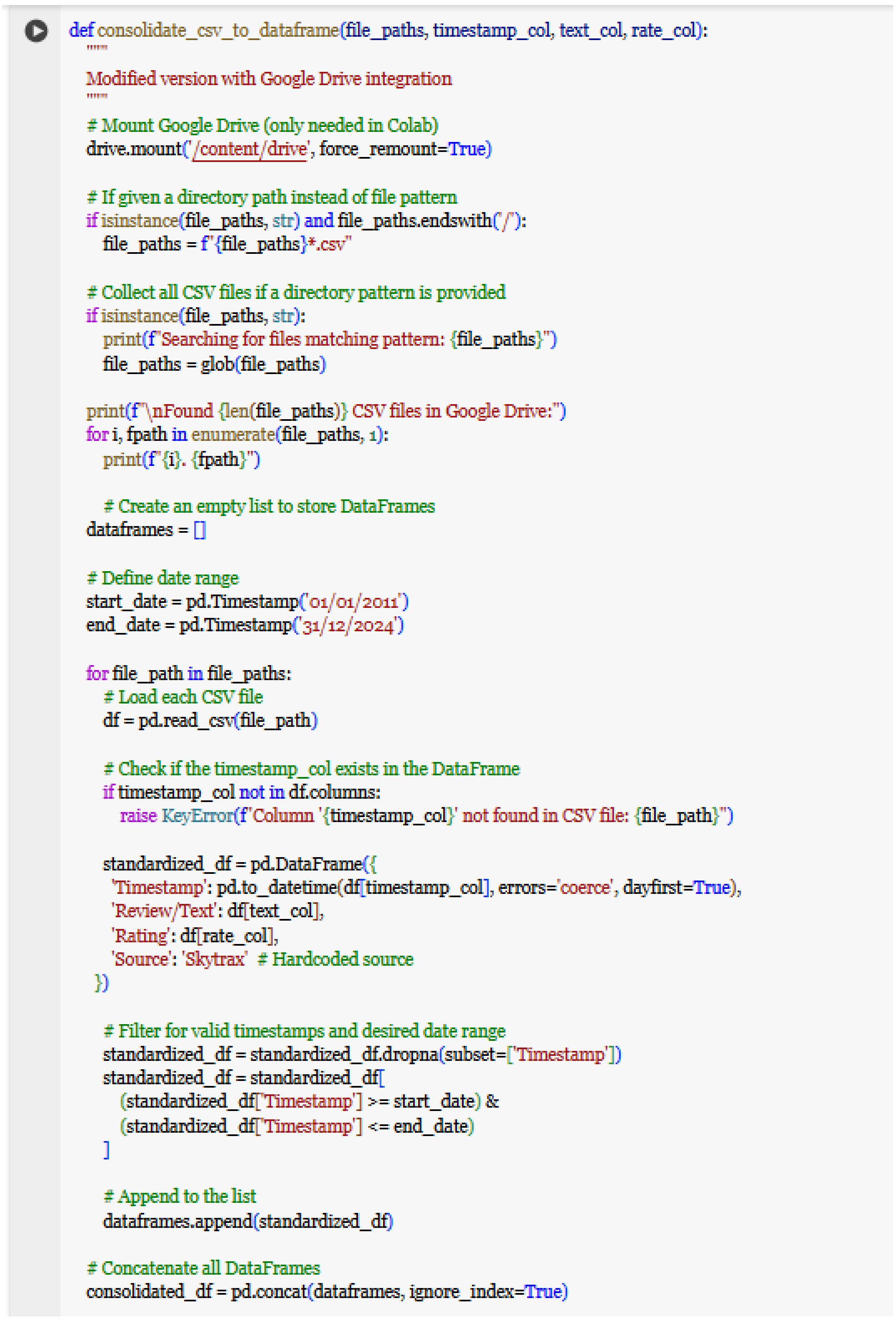
```
[ ]
    def scrape_yearly_reviews(self, year, max_reviews_per_month=100):
        """
        Scrape up to max_reviews_per_month for each month of the specified year.
        Combine all reviews into a single CSV file for the year.
        """
        all_reviews = []

        for month in range(1, 12):
            print(f"\nScraping reviews for {year}-{month:02d}")
            monthly_reviews = self.scrape_reviews(year, month, max_reviews=max_reviews_per_month)

            # Append monthly reviews to the yearly list
            all_reviews.extend(monthly_reviews)

        # Convert all reviews of the year into a single DataFrame
        yearly_df = pd.DataFrame(all_reviews)
        yearly_file = f"reviews_{year}.csv"
        yearly_df.to_csv(yearly_file, index=False)
        print(f"\nSaved all reviews for {year} to {yearly_file}")

        return yearly_df

if __name__ == "__main__":
    # Base URL for scraping reviews
    base_url = "https://www.airlinequality.com/airline-reviews/airasia"

    # Instantiate the scraper
    scraper = ReviewScraper(base_url)

    # Input the year and number of reviews to scrape per month
    year = int(input("Enter year  : "))
    max_reviews_per_month = int(input("Enter the maximum number of reviews you want to scrape per month: "))

    # Scrape reviews for the specified year and save as a single CSV file
    yearly_df = scraper.scrape_yearly_reviews(year, max_reviews_per_month)
    print("\nSample of scraped data:")
    print(yearly_df.head())
```

Sentiment Preprocessing for Skytrax

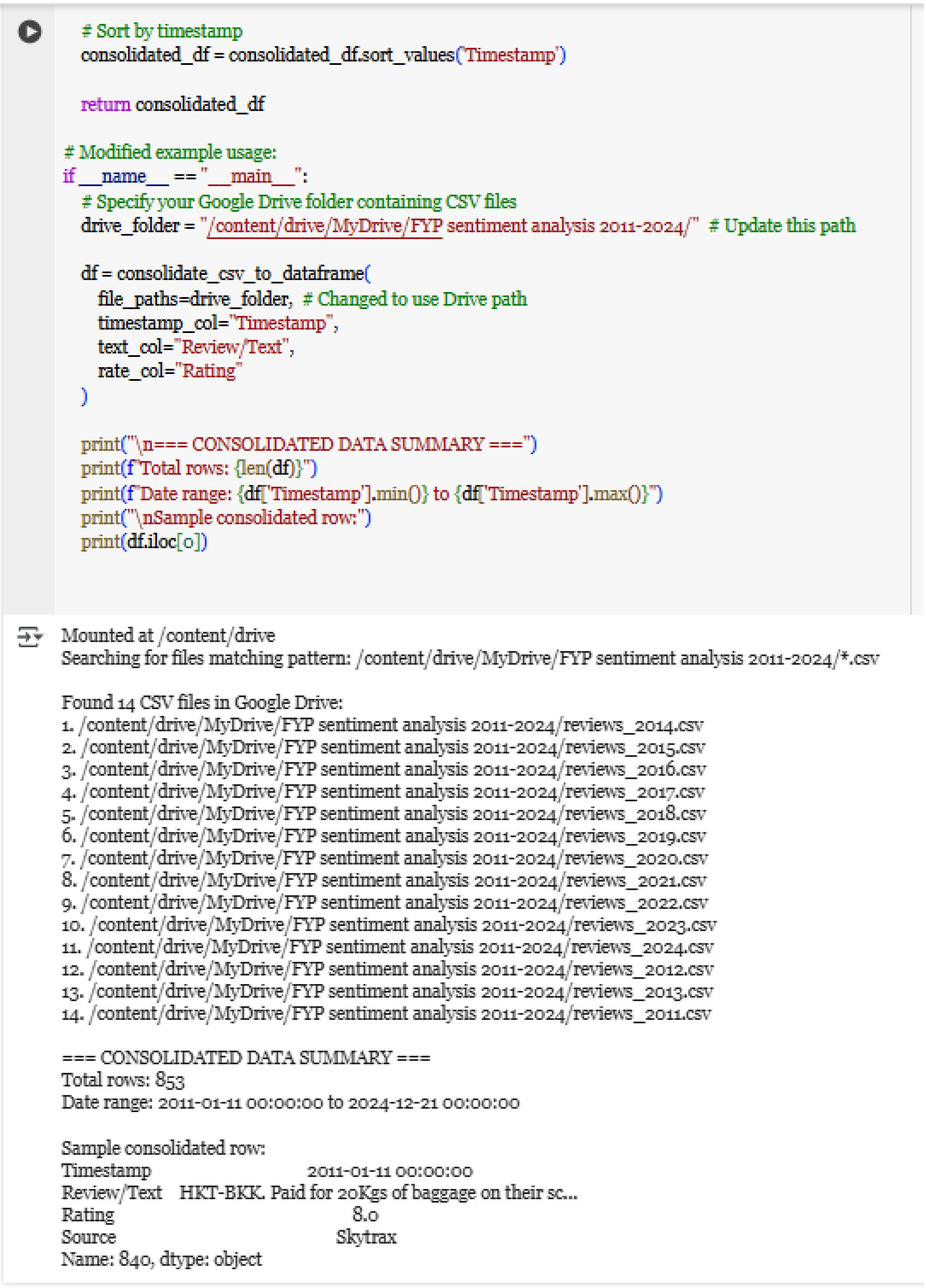
```
[ ] !pip install nltk
    import nltk
    nltk.download('stopwords')
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.11/dist-packages (3.9.1)
Requirement already satisfied: click in /usr/local/lib/python3.11/dist-packages (from nltk) (8.1.8)
Requirement already satisfied: joblib in /usr/local/lib/python3.11/dist-packages (from nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.11/dist-packages (from nltk) (2024.11.6)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from nltk) (4.67.1)
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
True
```

Phase 1 Data Consolidation

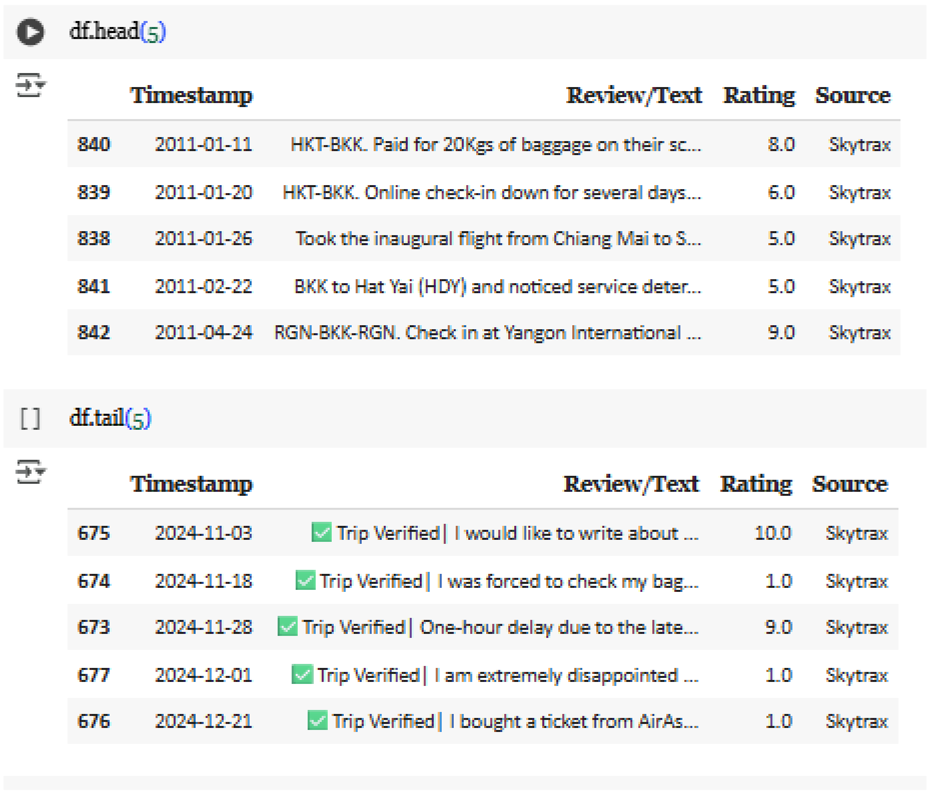


```
def consolidate_csv_to_dataframe(file_paths, timestamp_col, text_col, rate_col):
    """
    Modified version with Google Drive integration
    """
    # Mount Google Drive (only needed in Colab)
    drive.mount('/content/drive', force_remount=True)

    # If given a directory path instead of file pattern
    if isinstance(file_paths, str) and file_paths.endswith('/'):
        file_paths = f"{file_paths}*.csv"

    # Collect all CSV files if a directory pattern is provided
    if isinstance(file_paths, str):
        print(f"Searching for files matching pattern: {file_paths}")
        file_paths = glob(file_paths)

    print(f"\nFound {len(file_paths)} CSV files in Google Drive:")
    for i, fpath in enumerate(file_paths, 1):
        print(f"{i}. {fpath}")

        # Create an empty list to store DataFrames
    dataframes = []

    # Define date range
    start_date = pd.Timestamp('01/01/2011')
    end_date = pd.Timestamp('31/12/2024')

    for file_path in file_paths:
        # Load each CSV file
        df = pd.read_csv(file_path)

        # Check if the timestamp_col exists in the DataFrame
        if timestamp_col not in df.columns:
            raise KeyError(f"Column '{timestamp_col}' not found in CSV file: {file_path}")

        standardized_df = pd.DataFrame({
          'Timestamp': pd.to_datetime(df[timestamp_col], errors='coerce', dayfirst=True),
          'Review/Text': df[text_col],
          'Rating': df[rate_col],
          'Source': 'Skytrax'  # Hardcoded source
      })

        # Filter for valid timestamps and desired date range
        standardized_df = standardized_df.dropna(subset=['Timestamp'])
        standardized_df = standardized_df[
            (standardized_df['Timestamp'] >= start_date) &
            (standardized_df['Timestamp'] <= end_date)
        ]

        # Append to the list
        dataframes.append(standardized_df)

    # Concatenate all DataFrames
    consolidated_df = pd.concat(dataframes, ignore_index=True)
```



```
    # Sort by timestamp
    consolidated_df = consolidated_df.sort_values('Timestamp')

    return consolidated_df

# Modified example usage:
if __name__ == "__main__":
    # Specify your Google Drive folder containing CSV files
    drive_folder = "/content/drive/MyDrive/FYP sentiment analysis 2011-2024/"  # Update this path

    df = consolidate_csv_to_dataframe(
        file_paths=drive_folder,  # Changed to use Drive path
        timestamp_col="Timestamp",
        text_col="Review/Text",
        rate_col="Rating"
    )

    print("\n=== CONSOLIDATED DATA SUMMARY ===")
    print(f"Total rows: {len(df)}")
    print(f"Date range: {df['Timestamp'].min()} to {df['Timestamp'].max()}")
    print("\nSample consolidated row:")
    print(df.iloc[0])
```

```
Mounted at /content/drive
Searching for files matching pattern: /content/drive/MyDrive/FYP sentiment analysis 2011-2024/*.csv

Found 14 CSV files in Google Drive:
1. /content/drive/MyDrive/FYP sentiment analysis 2011-2024/reviews_2014.csv
2. /content/drive/MyDrive/FYP sentiment analysis 2011-2024/reviews_2015.csv
3. /content/drive/MyDrive/FYP sentiment analysis 2011-2024/reviews_2016.csv
4. /content/drive/MyDrive/FYP sentiment analysis 2011-2024/reviews_2017.csv
5. /content/drive/MyDrive/FYP sentiment analysis 2011-2024/reviews_2018.csv
6. /content/drive/MyDrive/FYP sentiment analysis 2011-2024/reviews_2019.csv
7. /content/drive/MyDrive/FYP sentiment analysis 2011-2024/reviews_2020.csv
8. /content/drive/MyDrive/FYP sentiment analysis 2011-2024/reviews_2021.csv
9. /content/drive/MyDrive/FYP sentiment analysis 2011-2024/reviews_2022.csv
10. /content/drive/MyDrive/FYP sentiment analysis 2011-2024/reviews_2023.csv
11. /content/drive/MyDrive/FYP sentiment analysis 2011-2024/reviews_2024.csv
12. /content/drive/MyDrive/FYP sentiment analysis 2011-2024/reviews_2012.csv
13. /content/drive/MyDrive/FYP sentiment analysis 2011-2024/reviews_2013.csv
14. /content/drive/MyDrive/FYP sentiment analysis 2011-2024/reviews_2011.csv

=== CONSOLIDATED DATA SUMMARY ===
Total rows: 853
Date range: 2011-01-11 00:00:00 to 2024-12-21 00:00:00

Sample consolidated row:
Timestamp                      2011-01-11 00:00:00
Review/Text    HKT-BKK. Paid for 20Kgs of baggage on their sc...
Rating                                         8.0
Source                                     Skytrax
Name: 840, dtype: object
```



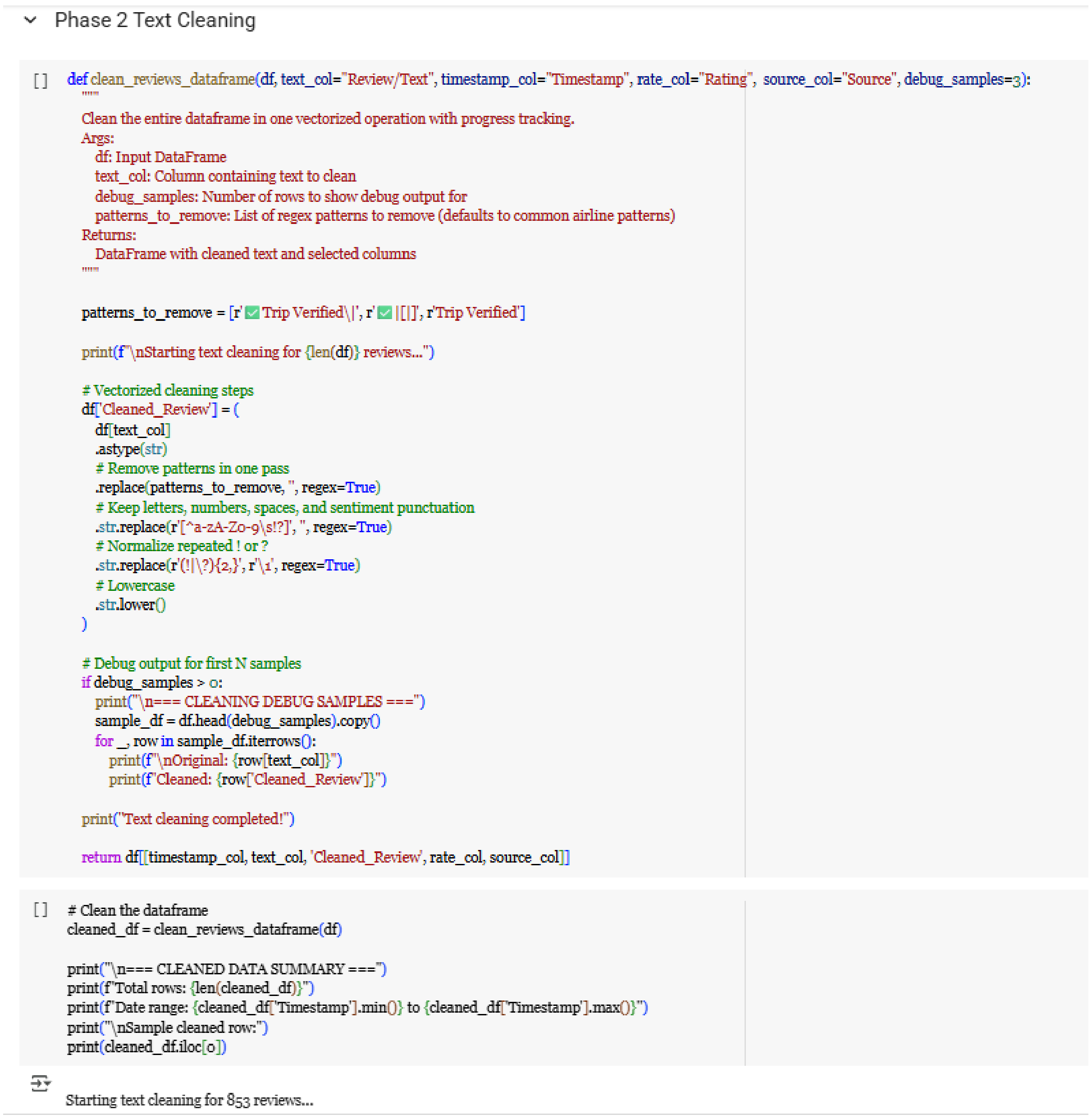
```
df.head(5)
```

	Timestamp	Review/Text	Rating	Source
840	2011-01-11	HKT-BKK. Paid for 20Kgs of baggage on their sc...	8.0	Skytrax
839	2011-01-20	HKT-BKK. Online check-in down for several days...	6.0	Skytrax
838	2011-01-26	Took the inaugural flight from Chiang Mai to S...	5.0	Skytrax
841	2011-02-22	BKK to Hat Yai (HDY) and noticed service deter...	5.0	Skytrax
842	2011-04-24	RGN-BKK-RGN. Check in at Yangon International ...	9.0	Skytrax

```
df.tail(5)
```

	Timestamp	Review/Text	Rating	Source
675	2024-11-03	✅ Trip Verified | I would like to write about ...	10.0	Skytrax
674	2024-11-18	✅ Trip Verified | I was forced to check my bag...	1.0	Skytrax
673	2024-11-28	✅ Trip Verified | One-hour delay due to the late...	9.0	Skytrax
677	2024-12-01	✅ Trip Verified | I am extremely disappointed ...	1.0	Skytrax
676	2024-12-21	✅ Trip Verified | I bought a ticket from AirAs...	1.0	Skytrax

APPENDIX

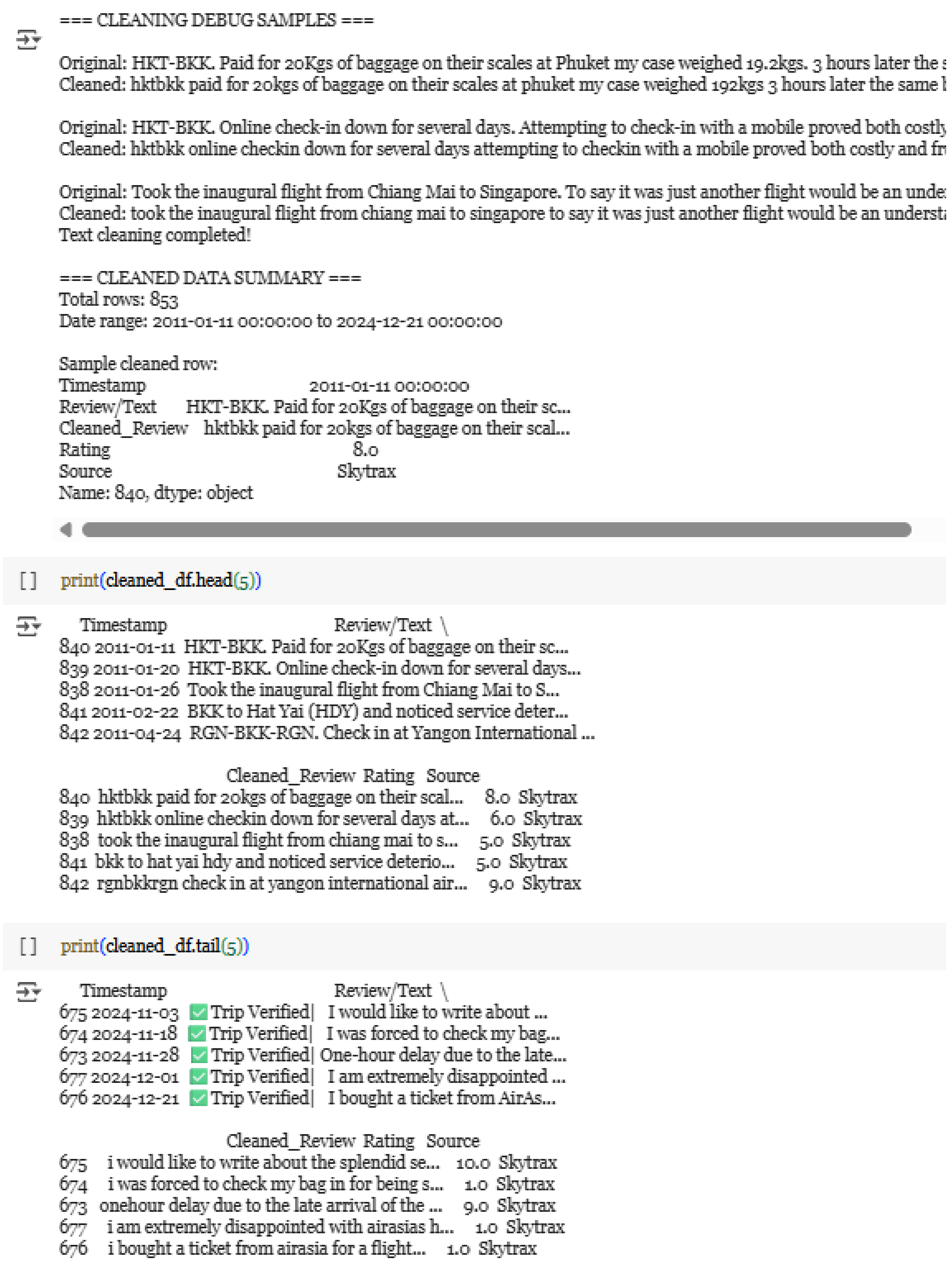


Phase 2 Text Cleaning

```
def clean_reviews_dataframe(df, text_col="Review/Text", timestamp_col="Timestamp", rate_col="Rating", source_col="Source", debug_samples=3):
    """
    Clean the entire dataframe in one vectorized operation with progress tracking.
    Args:
        df: Input DataFrame
        text_col: Column containing text to clean
        debug_samples: Number of rows to show debug output for
        patterns_to_remove: List of regex patterns to remove (defaults to common airline patterns)
    Returns:
        DataFrame with cleaned text and selected columns
    """

    patterns_to_remove = [r'✅ Trip Verified\|', r'✅ |[|]', r'Trip Verified']

    print(f"\nStarting text cleaning for {len(df)} reviews...")

    # Vectorized cleaning steps
    df['Cleaned_Review'] = (
        df[text_col]
        .astype(str)
        # Remove patterns in one pass
        .replace(patterns_to_remove, '', regex=True)
        # Keep letters, numbers, spaces, and sentiment punctuation
        .str.replace(r'[^a-zA-Z0-9\s!?]', '', regex=True)
        # Normalize repeated ! or ?
        .str.replace(r'(!|\?){2,}', r'\1', regex=True)
        # Lowercase
        .str.lower()
    )

    # Debug output for first N samples
    if debug_samples > 0:
        print("\n=== CLEANING DEBUG SAMPLES ===")
        sample_df = df.head(debug_samples).copy()
        for _, row in sample_df.iterrows():
            print(f"\nOriginal: {row[text_col]}")
            print(f'Cleaned: {row['Cleaned_Review']}")

    print("Text cleaning completed!")

    return df[[timestamp_col, text_col, 'Cleaned_Review', rate_col, source_col]]
```

```
# Clean the dataframe
cleaned_df = clean_reviews_dataframe(df)

print("\n=== CLEANED DATA SUMMARY ===")
print(f'Total rows: {len(cleaned_df)}")
print(f'Date range: {cleaned_df['Timestamp'].min()} to {cleaned_df['Timestamp'].max()}")
print("\nSample cleaned row:")
print(cleaned_df.iloc[0])
```

```
Starting text cleaning for 853 reviews...
```



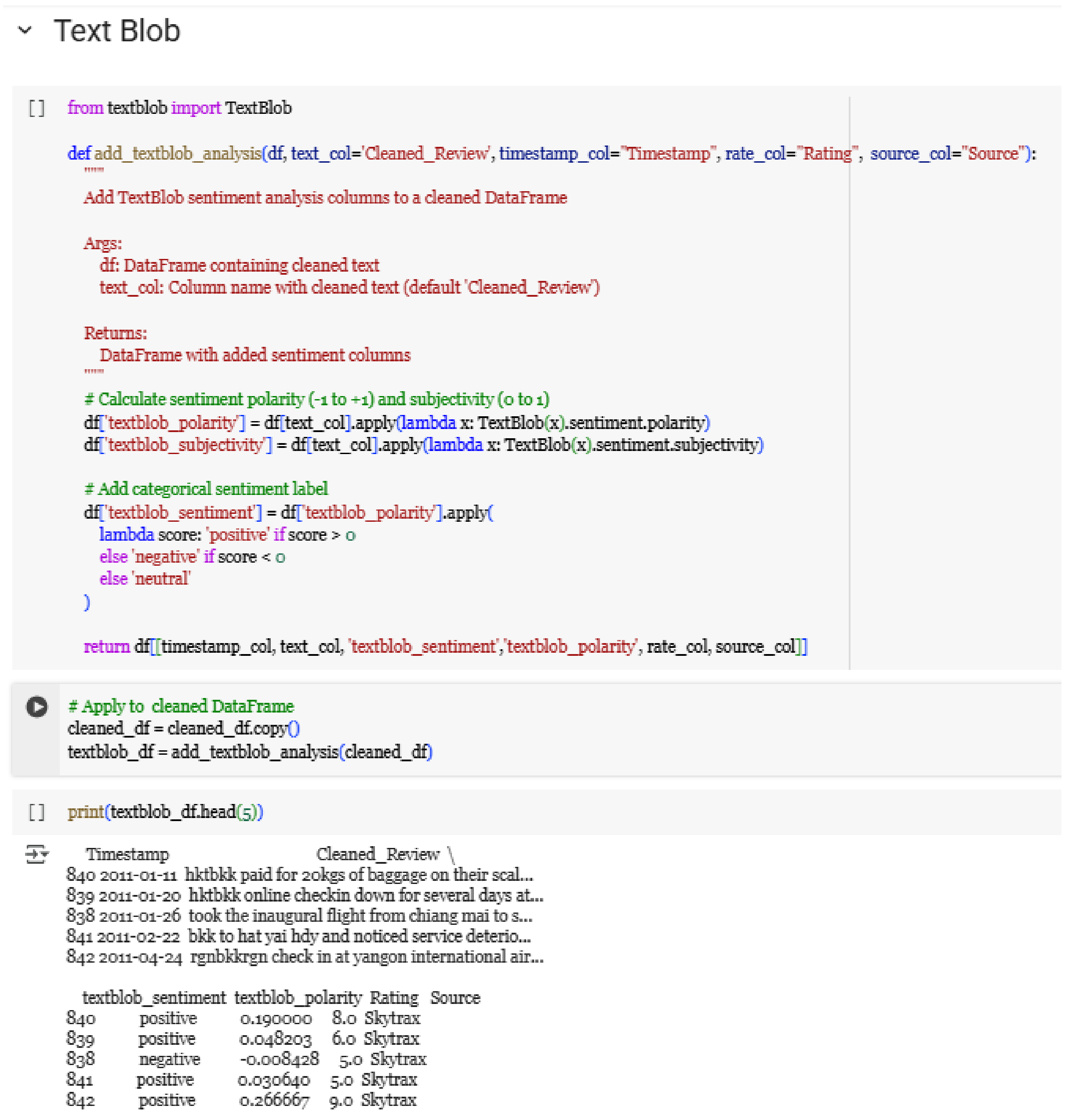
```
=== CLEANING DEBUG SAMPLES ===

Original: HKT-BKK. Paid for 20Kgs of baggage on their scales at Phuket my case weighed 19.2kgs. 3 hours later the s
Cleaned: hktbkk paid for 20kgs of baggage on their scales at phuket my case weighed 192kgs 3 hours later the same b

Original: HKT-BKK. Online check-in down for several days. Attempting to check-in with a mobile proved both costly
Cleaned: hktbkk online checkin down for several days attempting to checkin with a mobile proved both costly and fr

Original: Took the inaugural flight from Chiang Mai to Singapore. To say it was just another flight would be an unde
Cleaned: took the inaugural flight from chiang mai to singapore to say it was just another flight would be an underst
Text cleaning completed!

=== CLEANED DATA SUMMARY ===
Total rows: 853
Date range: 2011-01-11 00:00:00 to 2024-12-21 00:00:00

Sample cleaned row:
Timestamp                        2011-01-11 00:00:00
Review/Text       HKT-BKK. Paid for 20Kgs of baggage on their sc...
Cleaned_Review    hktbkk paid for 20kgs of baggage on their scal...
Rating                                               8.0
Source                                           Skytrax
Name: 840, dtype: object
```

```
[ ] print(cleaned_df.head(5))
```

```
     Timestamp                                    Review/Text  \
840 2011-01-11  HKT-BKK. Paid for 20Kgs of baggage on their sc...
839 2011-01-20  HKT-BKK. Online check-in down for several days...
838 2011-01-26  Took the inaugural flight from Chiang Mai to S...
841 2011-02-22  BKK to Hat Yai (HDY) and noticed service deter...
842 2011-04-24  RGN-BKK-RGN. Check in at Yangon International ...

                                        Cleaned_Review  Rating   Source
840  hktbkk paid for 20kgs of baggage on their scal...     8.0  Skytrax
839  hktbkk online checkin down for several days at...     6.0  Skytrax
838  took the inaugural flight from chiang mai to s...     5.0  Skytrax
841  bkk to hat yai hdy and noticed service deterio...     5.0  Skytrax
842  rgnbkkrgn check in at yangon international air...     9.0  Skytrax
```

```
[ ] print(cleaned_df.tail(5))
```

```
     Timestamp                                    Review/Text  \
675 2024-11-03  ✅ Trip Verified|  I would like to write about ...
674 2024-11-18  ✅ Trip Verified|  I was forced to check my bag...
673 2024-11-28  ✅ Trip Verified| One-hour delay due to the late...
677 2024-12-01  ✅ Trip Verified|  I am extremely disappointed ...
676 2024-12-21  ✅ Trip Verified|  I bought a ticket from AirAs...

                                        Cleaned_Review  Rating   Source
675   i would like to write about the splendid se...    10.0  Skytrax
674   i was forced to check my bag in for being s...     1.0  Skytrax
673  onehour delay due to the late arrival of the ...     9.0  Skytrax
677   i am extremely disappointed with airasias h...     1.0  Skytrax
676   i bought a ticket from airasia for a flight...     1.0  Skytrax
```



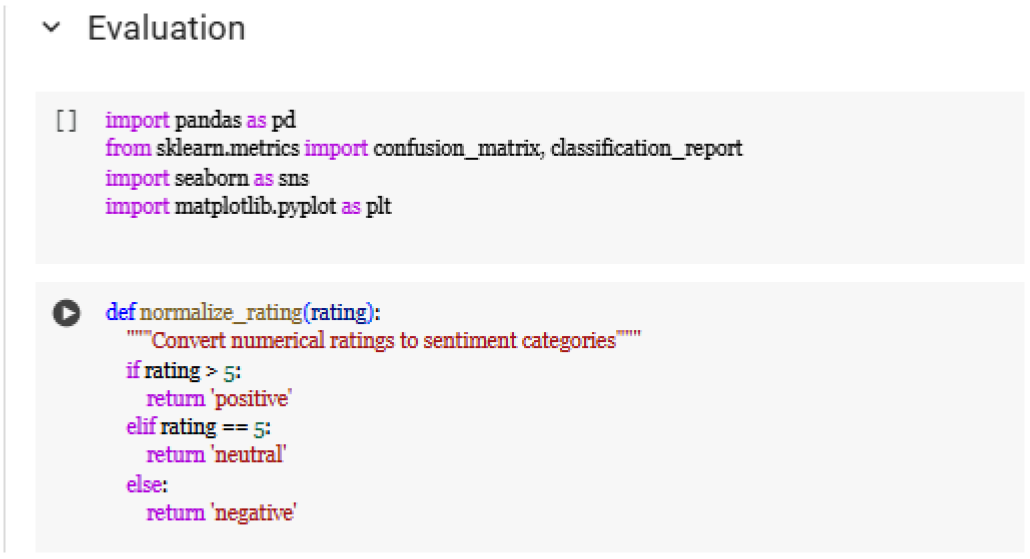
Text Blob

```
from textblob import TextBlob

def add_textblob_analysis(df, text_col='Cleaned_Review', timestamp_col="Timestamp", rate_col="Rating", source_col="Source"):
    """
    Add TextBlob sentiment analysis columns to a cleaned DataFrame

    Args:
        df: DataFrame containing cleaned text
        text_col: Column name with cleaned text (default 'Cleaned_Review')

    Returns:
        DataFrame with added sentiment columns
    """
    # Calculate sentiment polarity (-1 to +1) and subjectivity (0 to 1)
    df['textblob_polarity'] = df[text_col].apply(lambda x: TextBlob(x).sentiment.polarity)
    df['textblob_subjectivity'] = df[text_col].apply(lambda x: TextBlob(x).sentiment.subjectivity)

    # Add categorical sentiment label
    df['textblob_sentiment'] = df['textblob_polarity'].apply(
        lambda score: 'positive' if score > 0
        else 'negative' if score < 0
        else 'neutral'
    )

    return df[[timestamp_col, text_col, 'textblob_sentiment','textblob_polarity', rate_col, source_col]]
```

```
# Apply to  cleaned DataFrame
cleaned_df = cleaned_df.copy()
textblob_df = add_textblob_analysis(cleaned_df)
```

```
print(textblob_df.head(5))
```

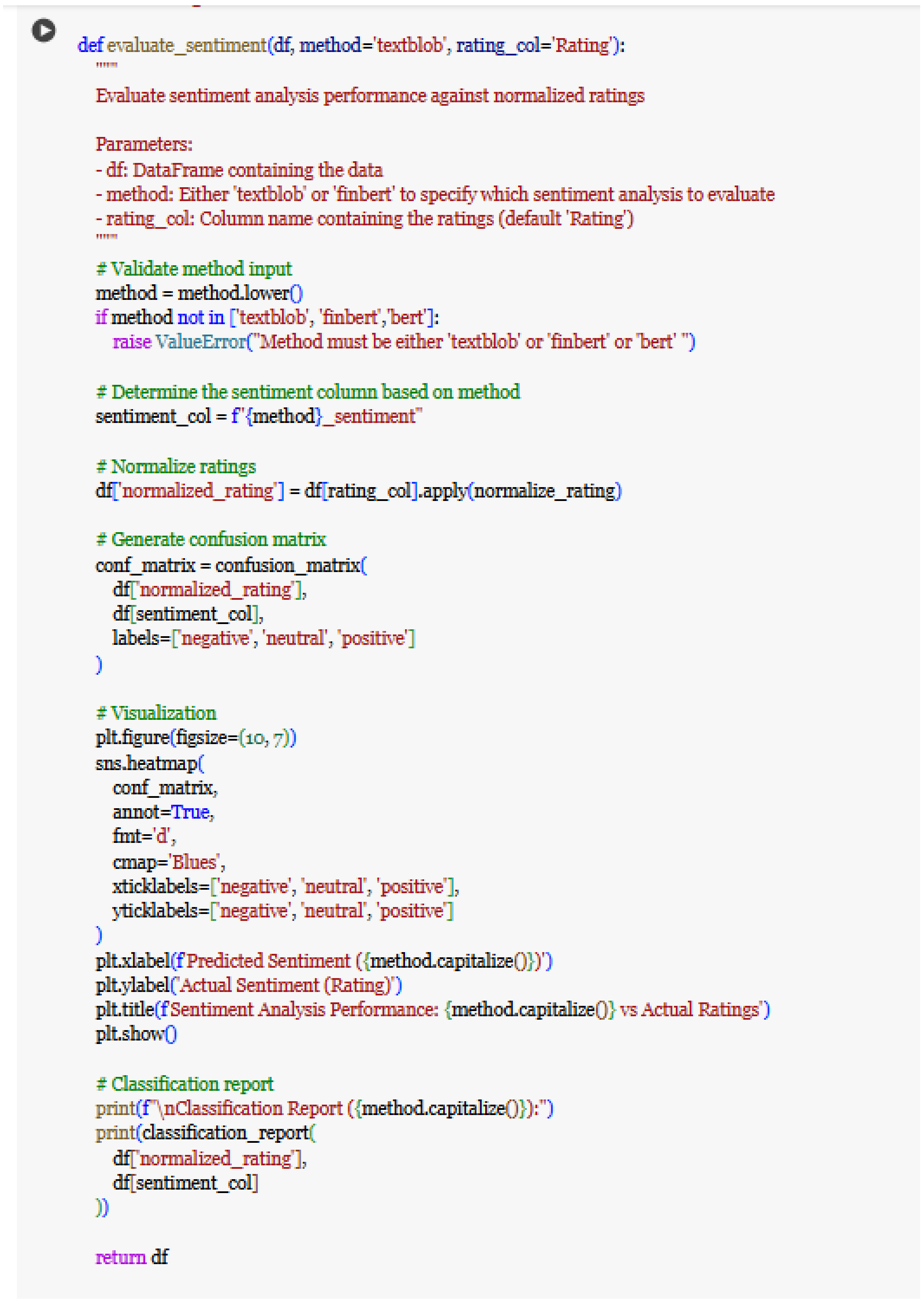
```
     Timestamp                           Cleaned_Review  \
840 2011-01-11  hktbkk paid for 20kgs of baggage on their scal...
839 2011-01-20  hktbkk online checkin down for several days at...
838 2011-01-26  took the inaugural flight from chiang mai to s...
841 2011-02-22  bkk to hat yai hdy and noticed service deterio...
842 2011-04-24  rgnbkkrgn check in at yangon international air...

    textblob_sentiment  textblob_polarity  Rating   Source
840          positive           0.190000     8.0  Skytrax
839          positive           0.048203     6.0  Skytrax
838          negative          -0.008428     5.0  Skytrax
841          positive           0.030640     5.0  Skytrax
842          positive           0.266667     9.0  Skytrax
```



Evaluation

```
import pandas as pd
from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns
import matplotlib.pyplot as plt
```

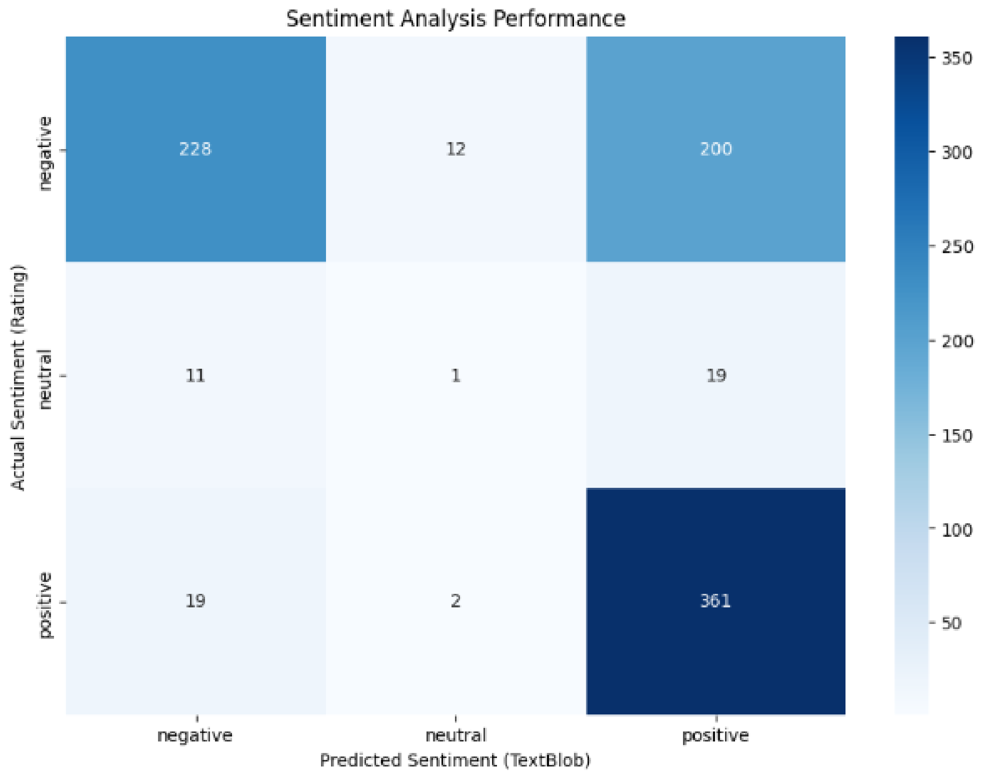
```
def normalize_rating(rating):
    """Convert numerical ratings to sentiment categories"""
    if rating > 5:
        return 'positive'
    elif rating == 5:
        return 'neutral'
    else:
        return 'negative'
```



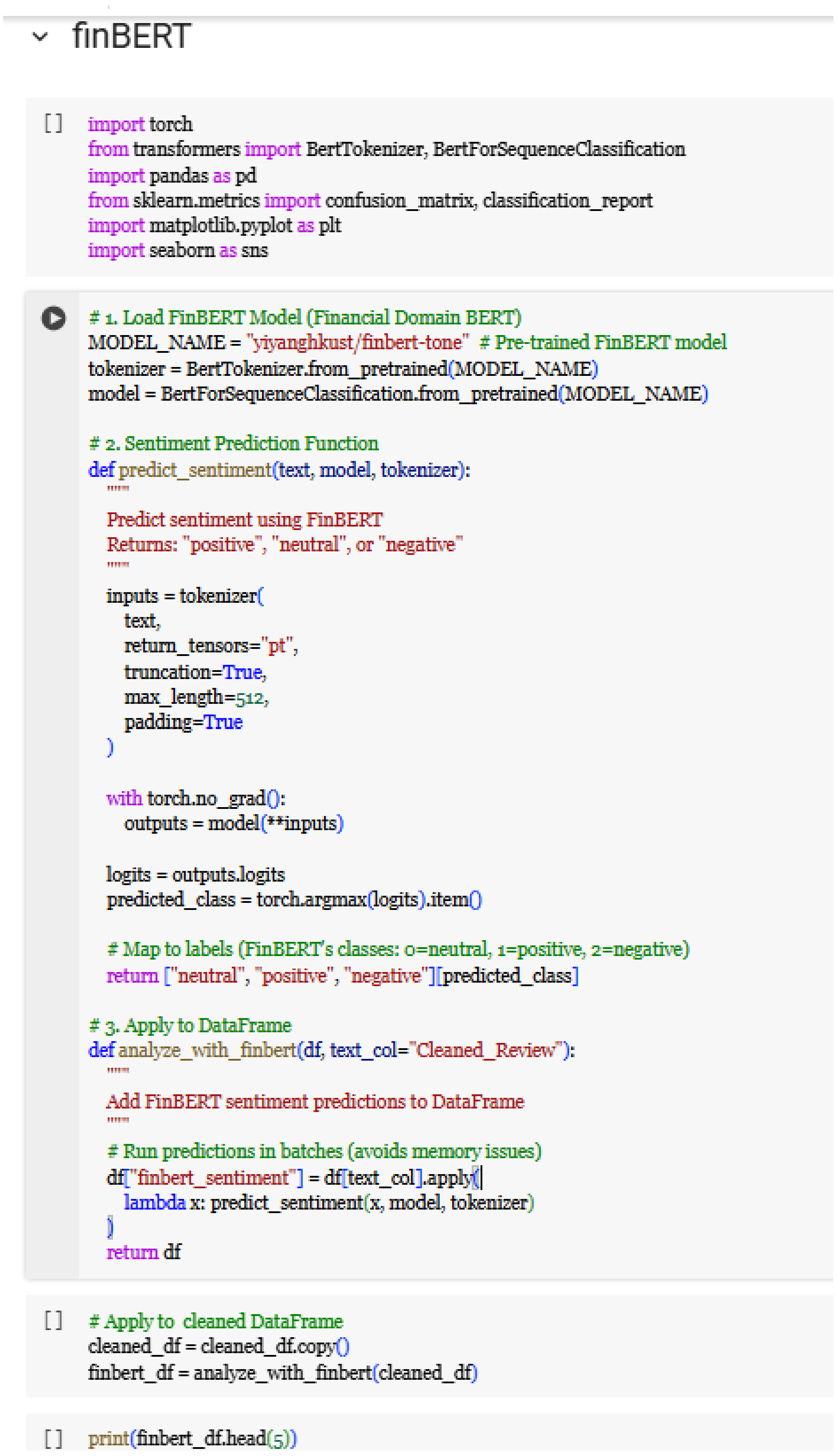
```python
def evaluate_sentiment(df, method='textblob', rating_col='Rating'):
    """
    Evaluate sentiment analysis performance against normalized ratings

    Parameters:
    - df: DataFrame containing the data
    - method: Either 'textblob' or 'finbert' to specify which sentiment analysis to evaluate
    - rating_col: Column name containing the ratings (default 'Rating')
    """
    # Validate method input
    method = method.lower()
    if method not in ['textblob', 'finbert','bert']:
        raise ValueError("Method must be either 'textblob' or 'finbert' or 'bert' ")

    # Determine the sentiment column based on method
    sentiment_col = f"{method}_sentiment"

    # Normalize ratings
    df['normalized_rating'] = df[rating_col].apply(normalize_rating)

    # Generate confusion matrix
    conf_matrix = confusion_matrix(
        df['normalized_rating'],
        df[sentiment_col],
        labels=['negative', 'neutral', 'positive']
    )

    # Visualization
    plt.figure(figsize=(10, 7))
    sns.heatmap(
        conf_matrix,
        annot=True,
        fmt='d',
        cmap='Blues',
        xticklabels=['negative', 'neutral', 'positive'],
        yticklabels=['negative', 'neutral', 'positive']
    )
    plt.xlabel(f'Predicted Sentiment ({method.capitalize()})')
    plt.ylabel('Actual Sentiment (Rating)')
    plt.title(f'Sentiment Analysis Performance: {method.capitalize()} vs Actual Ratings')
    plt.show()

    # Classification report
    print(f"\nClassification Report ({method.capitalize()}):")
    print(classification_report(
        df['normalized_rating'],
        df[sentiment_col]
    ))

    return df
```

```
textblob_df = evaluate_sentiment(textblob_df)

# Show sample results
print("\nSample Results:")
print(textblob_df[['Cleaned_Review', 'Rating', 'normalized_rating', 'textblob_sentiment']].head())
```

```
<ipython-input-123-6250749c25f1>:15: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['normalized_rating'] = df[rating_col].apply(normalize_rating)
```



```
Classification Report:
              precision    recall  f1-score   support

    negative       0.88      0.52      0.65       440
     neutral       0.07      0.03      0.04        31
    positive       0.62      0.95      0.75       382

    accuracy                           0.69       853
   macro avg       0.52      0.50      0.48       853
weighted avg       0.74      0.69      0.67       853


Sample Results:
                                        Cleaned_Review  Rating  \
840  hktbkk paid for 20kgs of baggage on their scal...     8.0
839  hktbkk online checkin down for several days at...     6.0
838  took the inaugural flight from chiang mai to s...     5.0
841  bkk to hat yai hdy and noticed service deterio...     5.0
842  rgnbkkrgn check in at yangon international air...     9.0

    normalized_rating textblob_sentiment
840          positive           positive
839          positive           positive
838           neutral           negative
841           neutral           positive
842          positive           positive
```

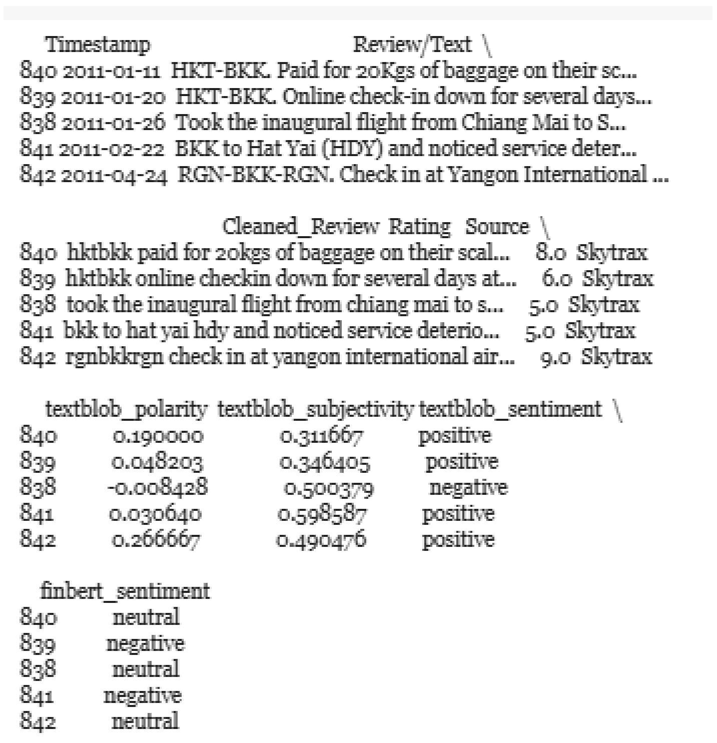


```
import torch
from transformers import BertTokenizer, BertForSequenceClassification
import pandas as pd
from sklearn.metrics import confusion_matrix, classification_report
import matplotlib.pyplot as plt
import seaborn as sns
```

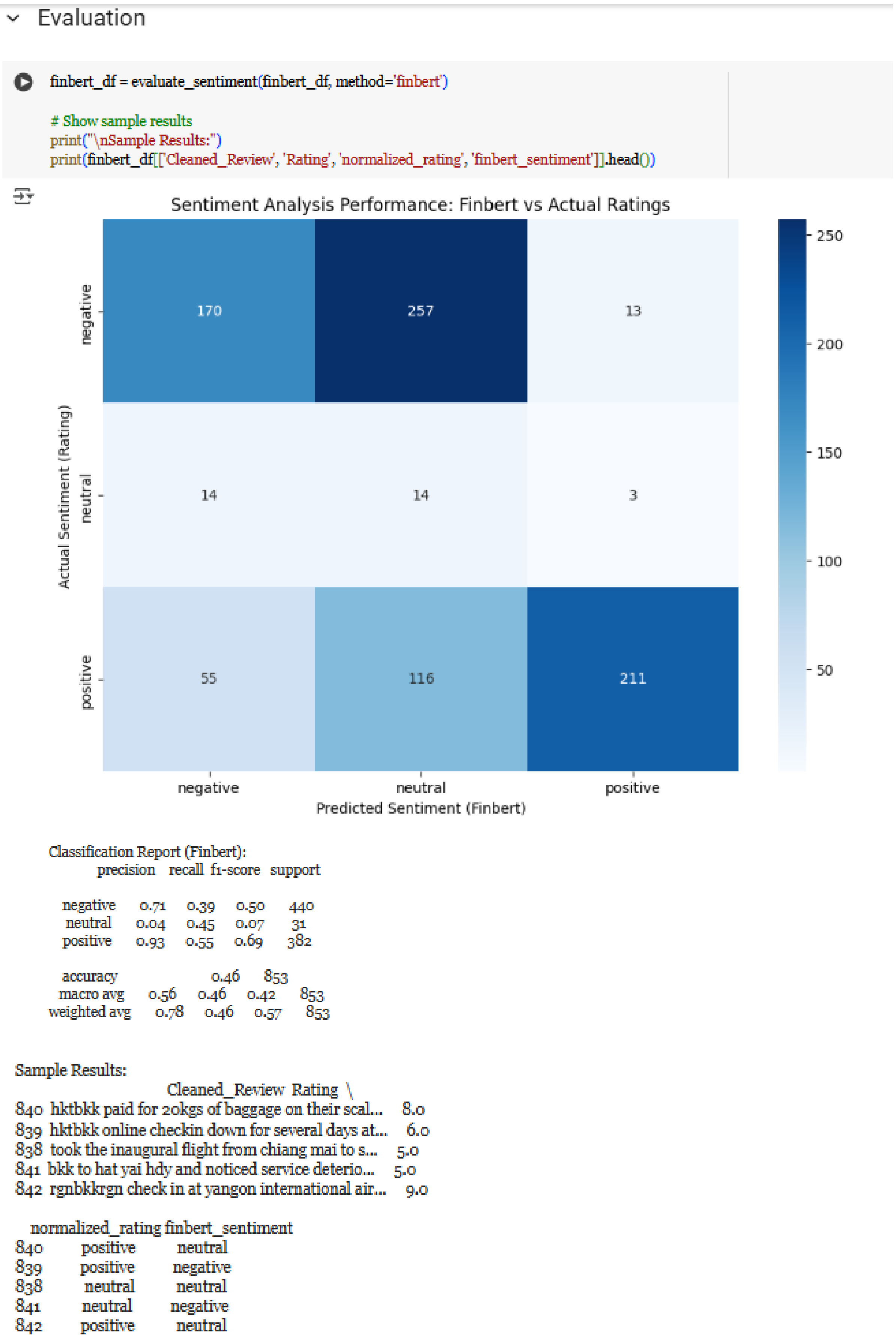
```
# 1. Load FinBERT Model (Financial Domain BERT)
MODEL_NAME = "yiyanghkust/finbert-tone"  # Pre-trained FinBERT model
tokenizer = BertTokenizer.from_pretrained(MODEL_NAME)
model = BertForSequenceClassification.from_pretrained(MODEL_NAME)

# 2. Sentiment Prediction Function
def predict_sentiment(text, model, tokenizer):
    """
    Predict sentiment using FinBERT
    Returns: "positive", "neutral", or "negative"
    """
    inputs = tokenizer(
        text,
        return_tensors="pt",
        truncation=True,
        max_length=512,
        padding=True
    )

    with torch.no_grad():
        outputs = model(**inputs)

    logits = outputs.logits
    predicted_class = torch.argmax(logits).item()

    # Map to labels (FinBERT's classes: 0=neutral, 1=positive, 2=negative)
    return ["neutral", "positive", "negative"][predicted_class]

# 3. Apply to DataFrame
def analyze_with_finbert(df, text_col="Cleaned_Review"):
    """
    Add FinBERT sentiment predictions to DataFrame
    """
    # Run predictions in batches (avoids memory issues)
    df["finbert_sentiment"] = df[text_col].apply(
        lambda x: predict_sentiment(x, model, tokenizer)
    )
    return df
```

```
# Apply to  cleaned DataFrame
cleaned_df = cleaned_df.copy()
finbert_df = analyze_with_finbert(cleaned_df)
```

```
print(finbert_df.head(5))
```

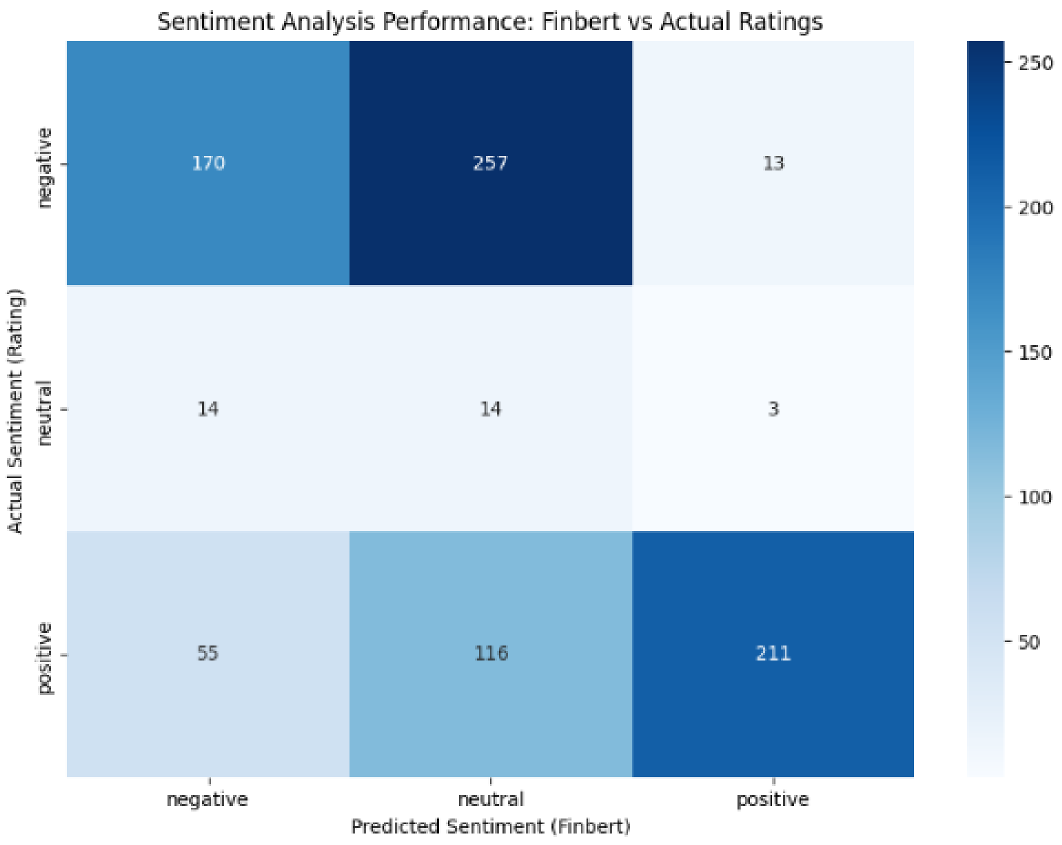


```
    Timestamp                           Review/Text  \
840 2011-01-11  HKT-BKK. Paid for 20Kgs of baggage on their sc...
839 2011-01-20  HKT-BKK. Online check-in down for several days...
838 2011-01-26  Took the inaugural flight from Chiang Mai to S...
841 2011-02-22  BKK to Hat Yai (HDY) and noticed service deter...
842 2011-04-24  RGN-BKK-RGN. Check in at Yangon International ...

                         Cleaned_Review  Rating   Source  \
840  hktbkk paid for 20kgs of baggage on their scal...     8.0  Skytrax
839  hktbkk online checkin down for several days at...     6.0  Skytrax
838  took the inaugural flight from chiang mai to s...     5.0  Skytrax
841  bkk to hat yai hdy and noticed service deterio...     5.0  Skytrax
842  rgnbkkrgn check in at yangon international air...     9.0  Skytrax

     textblob_polarity  textblob_subjectivity textblob_sentiment  \
840          0.190000               0.311667           positive
839          0.048203               0.346405           positive
838         -0.008428               0.500379           negative
841          0.030640               0.598587           positive
842          0.266667               0.490476           positive

    finbert_sentiment
840           neutral
839          negative
838           neutral
841          negative
842           neutral
```



Evaluation

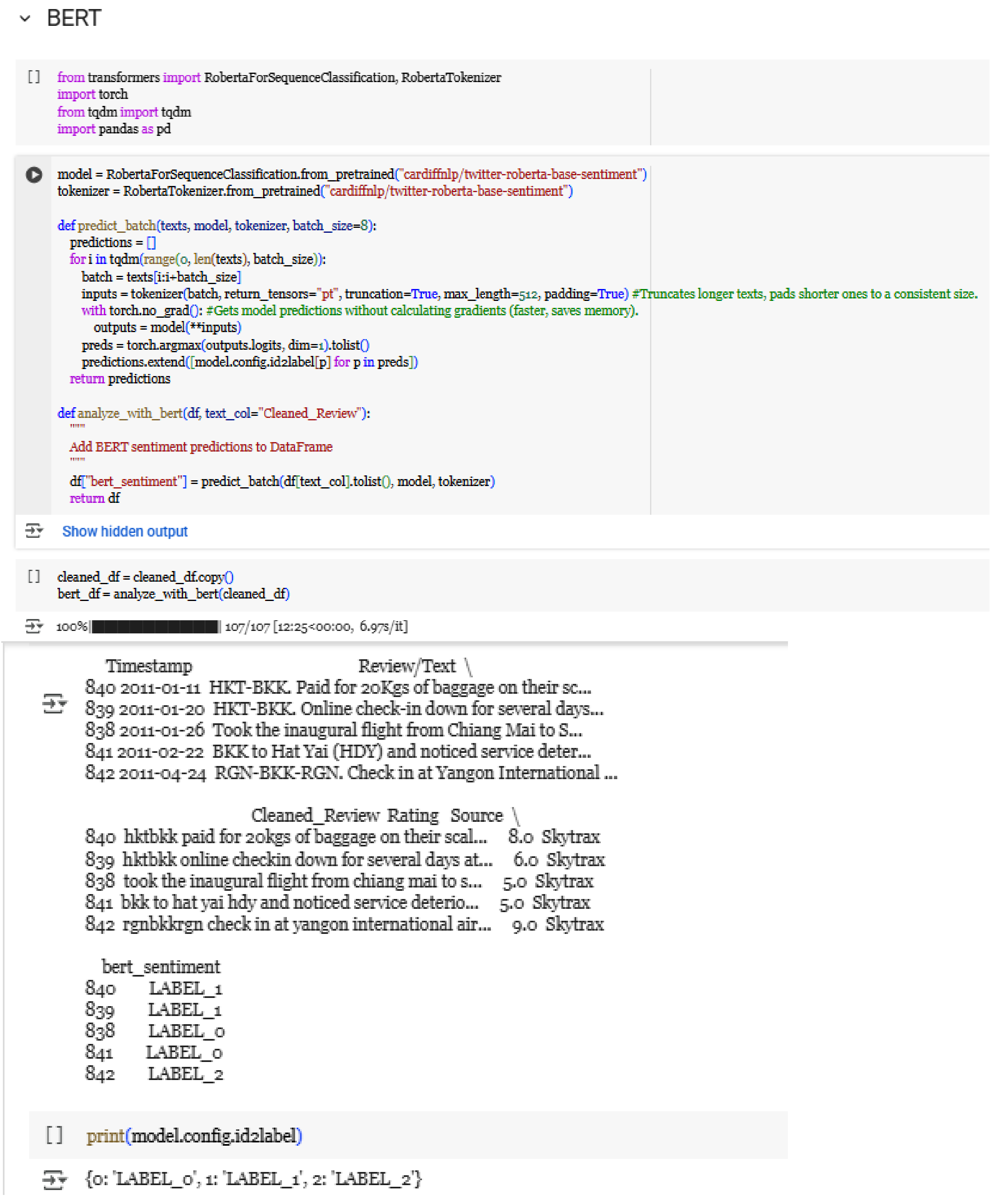
```
finbert_df = evaluate_sentiment(finbert_df, method='finbert')

# Show sample results
print("\nSample Results:")
print(finbert_df[['Cleaned_Review', 'Rating', 'normalized_rating', 'finbert_sentiment']].head())
```



	negative	neutral	positive
negative	170	257	13
neutral	14	14	3
positive	55	116	211

```
Classification Report (Finbert):
              precision    recall  f1-score   support

    negative       0.71      0.39      0.50       440
     neutral       0.04      0.45      0.07        31
    positive       0.93      0.55      0.69       382

    accuracy                           0.46       853
   macro avg       0.56      0.46      0.42       853
weighted avg       0.78      0.46      0.57       853


Sample Results:
                                        Cleaned_Review  Rating  \
840  hktbkk paid for 20kgs of baggage on their scal...     8.0
839  hktbkk online checkin down for several days at...     6.0
838  took the inaugural flight from chiang mai to s...     5.0
841  bkk to hat yai hdy and noticed service deterio...     5.0
842  rgnbkkrgn check in at yangon international air...     9.0

    normalized_rating finbert_sentiment
840          positive           neutral
839          positive          negative
838           neutral           neutral
841           neutral          negative
842          positive           neutral
```

APPENDIX



BERT

```
from transformers import RobertaForSequenceClassification, RobertaTokenizer
import torch
from tqdm import tqdm
import pandas as pd
```

```
model = RobertaForSequenceClassification.from_pretrained("cardiffnlp/twitter-roberta-base-sentiment")
tokenizer = RobertaTokenizer.from_pretrained("cardiffnlp/twitter-roberta-base-sentiment")

def predict_batch(texts, model, tokenizer, batch_size=8):
    predictions = []
    for i in tqdm(range(0, len(texts), batch_size)):
        batch = texts[i:i+batch_size]
        inputs = tokenizer(batch, return_tensors="pt", truncation=True, max_length=512, padding=True) #Truncates longer texts, pads shorter ones to a consistent size.
        with torch.no_grad(): #Gets model predictions without calculating gradients (faster, saves memory).
            outputs = model(**inputs)
        preds = torch.argmax(outputs.logits, dim=1).tolist()
        predictions.extend([model.config.id2label[p] for p in preds])
    return predictions

def analyze_with_bert(df, text_col="Cleaned_Review"):
    """
    Add BERT sentiment predictions to DataFrame
    """
    df["bert_sentiment"] = predict_batch(df[text_col].tolist(), model, tokenizer)
    return df
```

Show hidden output

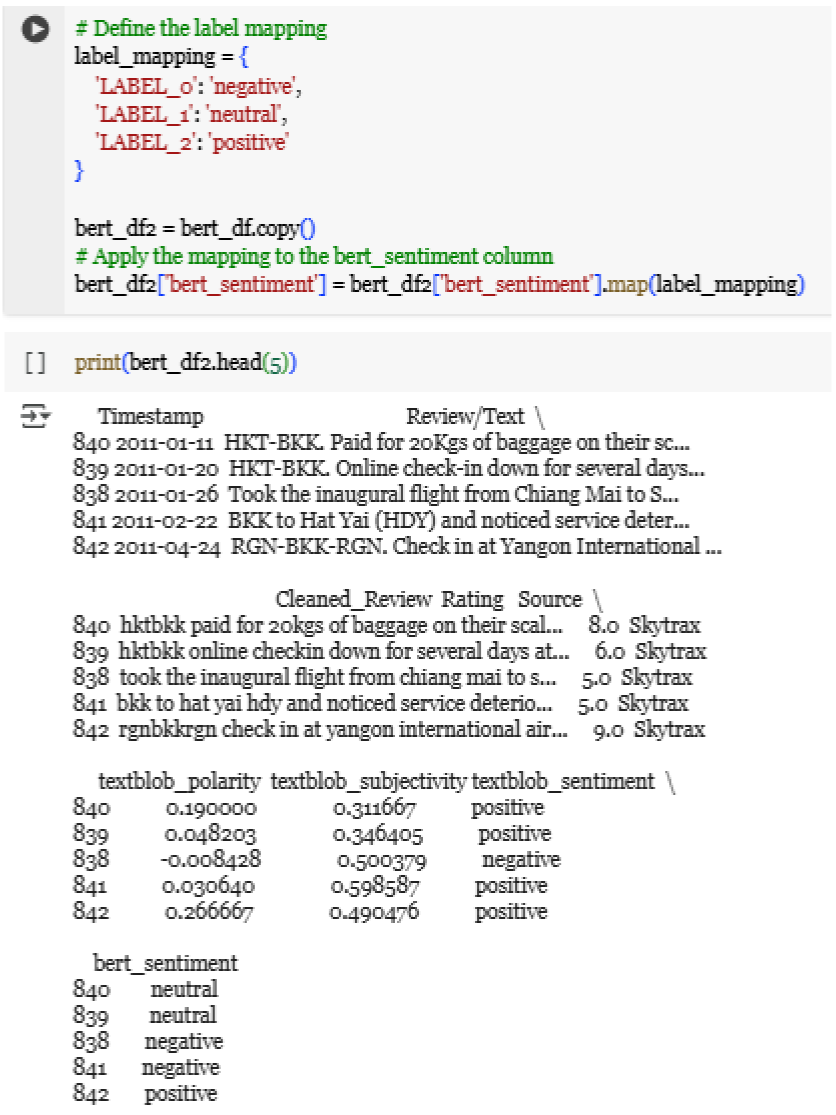
```
cleaned_df = cleaned_df.copy()
bert_df = analyze_with_bert(cleaned_df)
```

```
100%|██████████████| 107/107 [12:25<00:00, 6.97s/it]
```

```
     Timestamp                                        Review/Text  \
840 2011-01-11  HKT-BKK. Paid for 20Kgs of baggage on their sc...
839 2011-01-20  HKT-BKK. Online check-in down for several days...
838 2011-01-26  Took the inaugural flight from Chiang Mai to S...
841 2011-02-22  BKK to Hat Yai (HDY) and noticed service deter...
842 2011-04-24  RGN-BKK-RGN. Check in at Yangon International ...

                                      Cleaned_Review  Rating   Source  \
840  hktbkk paid for 20kgs of baggage on their scal...     8.0  Skytrax
839  hktbkk online checkin down for several days at...     6.0  Skytrax
838  took the inaugural flight from chiang mai to s...     5.0  Skytrax
841  bkk to hat yai hdy and noticed service deterio...     5.0  Skytrax
842  rgnbkkrgn check in at yangon international air...     9.0  Skytrax

    bert_sentiment
840        LABEL_1
839        LABEL_1
838        LABEL_0
841        LABEL_0
842        LABEL_2
```

```
print(model.config.id2label)
```

```
{0: 'LABEL_0', 1: 'LABEL_1', 2: 'LABEL_2'}
```



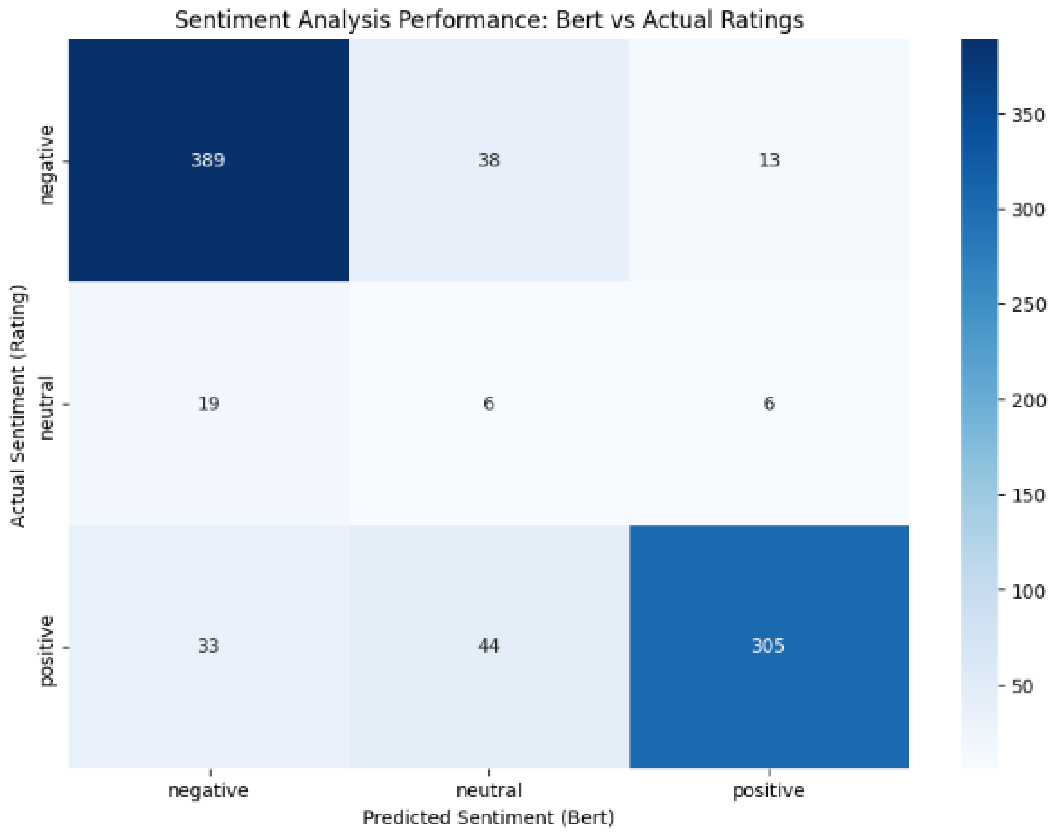
```
# Define the label mapping
label_mapping = {
  'LABEL_0': 'negative',
  'LABEL_1': 'neutral',
  'LABEL_2': 'positive'
}

bert_df2 = bert_df.copy()
# Apply the mapping to the bert_sentiment column
bert_df2['bert_sentiment'] = bert_df2['bert_sentiment'].map(label_mapping)
```

```
[ ] print(bert_df2.head(5))
```

```
     Timestamp                          Review/Text  \
840 2011-01-11  HKT-BKK. Paid for 20Kgs of baggage on their sc...
839 2011-01-20  HKT-BKK. Online check-in down for several days...
838 2011-01-26  Took the inaugural flight from Chiang Mai to S...
841 2011-02-22  BKK to Hat Yai (HDY) and noticed service deter...
842 2011-04-24  RGN-BKK-RGN. Check in at Yangon International ...

                                    Cleaned_Review  Rating  Source  \
840  hktbkk paid for 20kgs of baggage on their scal...     8.0  Skytrax
839  hktbkk online checkin down for several days at...     6.0  Skytrax
838  took the inaugural flight from chiang mai to s...     5.0  Skytrax
841  bkk to hat yai hdy and noticed service deterio...     5.0  Skytrax
842  rgnbkkrgn check in at yangon international air...     9.0  Skytrax

    textblob_polarity  textblob_subjectivity textblob_sentiment  \
840           0.190000               0.311667           positive
839           0.048203               0.346405           positive
838          -0.008428               0.500379           negative
841           0.030640               0.598587           positive
842           0.266667               0.490476           positive

    bert_sentiment
840        neutral
839        neutral
838       negative
841       negative
842       positive
```

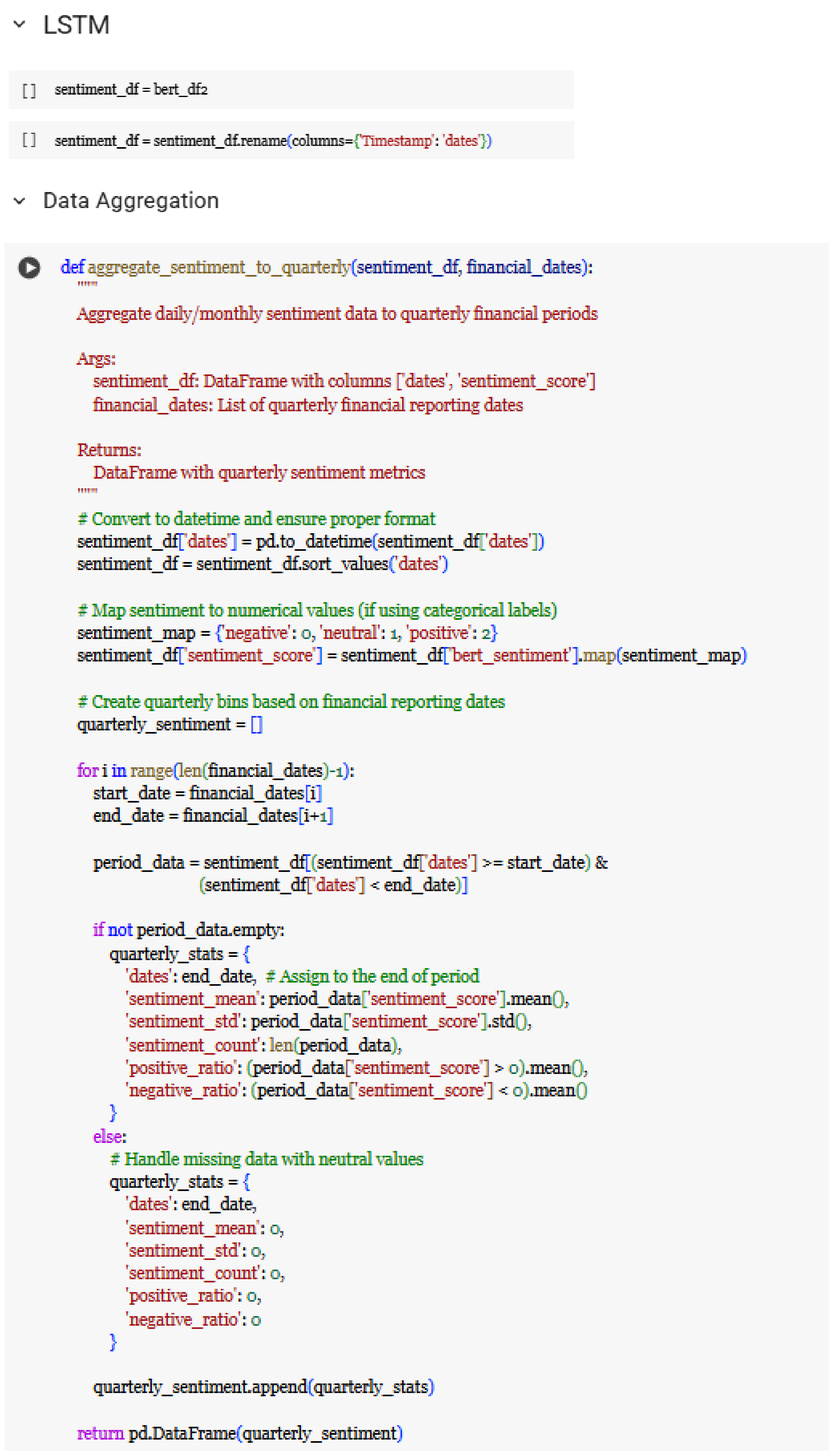
Evaluation

```
bert_df2 = evaluate_sentiment(bert_df2, method='bert')

# Show sample results
print("\nSample Results:")
print(bert_df2[['Cleaned_Review', 'Rating', 'normalized_rating', 'bert_sentiment']].head())
```



```
Classification Report (Bert):
              precision    recall  f1-score   support

    negative       0.88      0.88      0.88       440
     neutral       0.07      0.19      0.10        31
    positive       0.94      0.80      0.86       382

    accuracy                           0.82       853
   macro avg       0.63      0.63      0.62       853
weighted avg       0.88      0.82      0.85       853

Sample Results:
                                      Cleaned_Review  Rating  \
840  hktbkk paid for 20kgs of baggage on their scal...     8.0
839  hktbkk online checkin down for several days at...     6.0
838  took the inaugural flight from chiang mai to s...     5.0
841  bkk to hat yai hdy and noticed service deterio...     5.0
842  rgnbkkrgn check in at yangon international air...     9.0

    normalized_rating bert_sentiment
840          positive        neutral
839          positive        neutral
838           neutral       negative
841           neutral       negative
842          positive       positive
```

APPENDIX



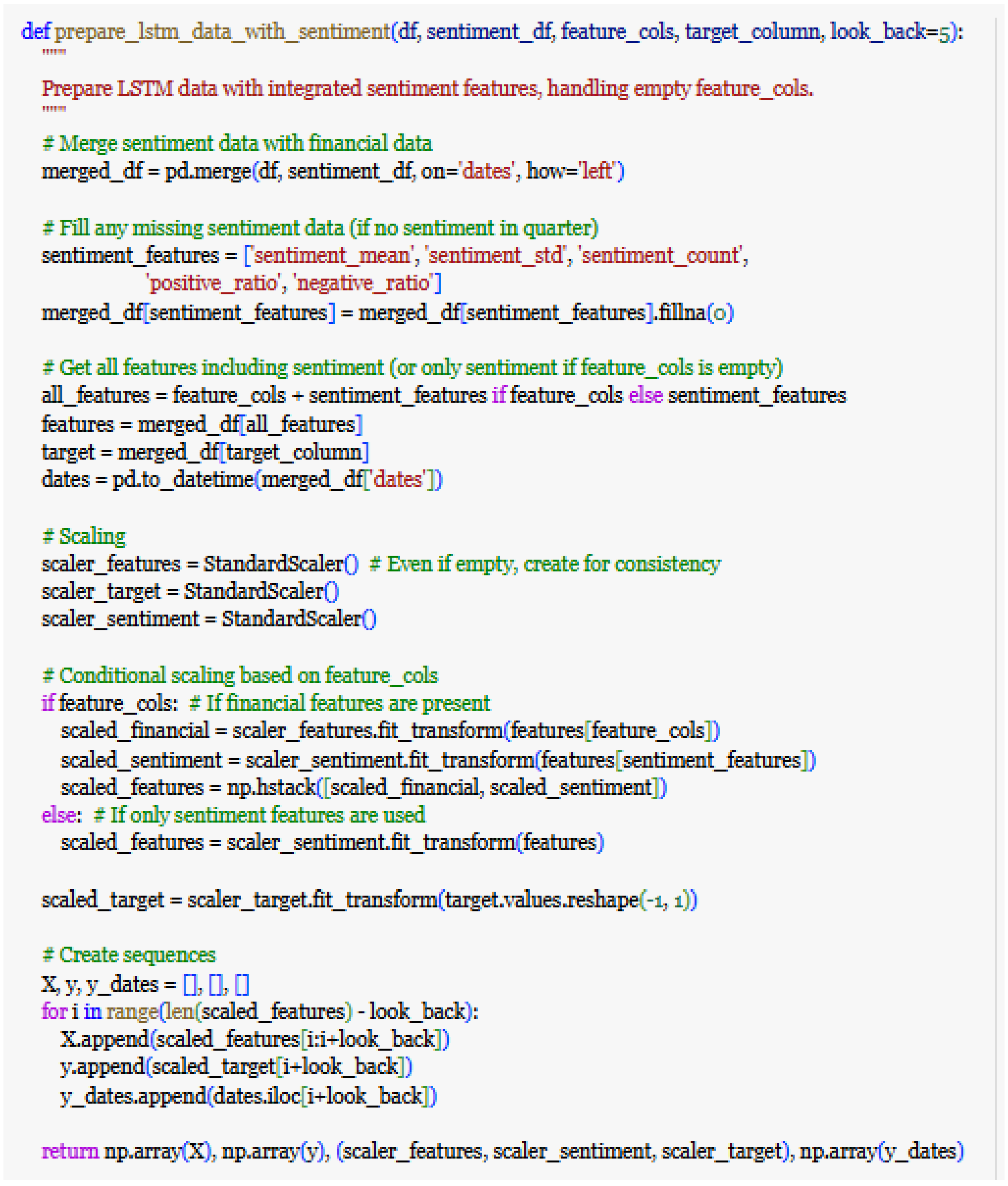
LSTM

```
[ ] sentiment_df = bert_df2
```

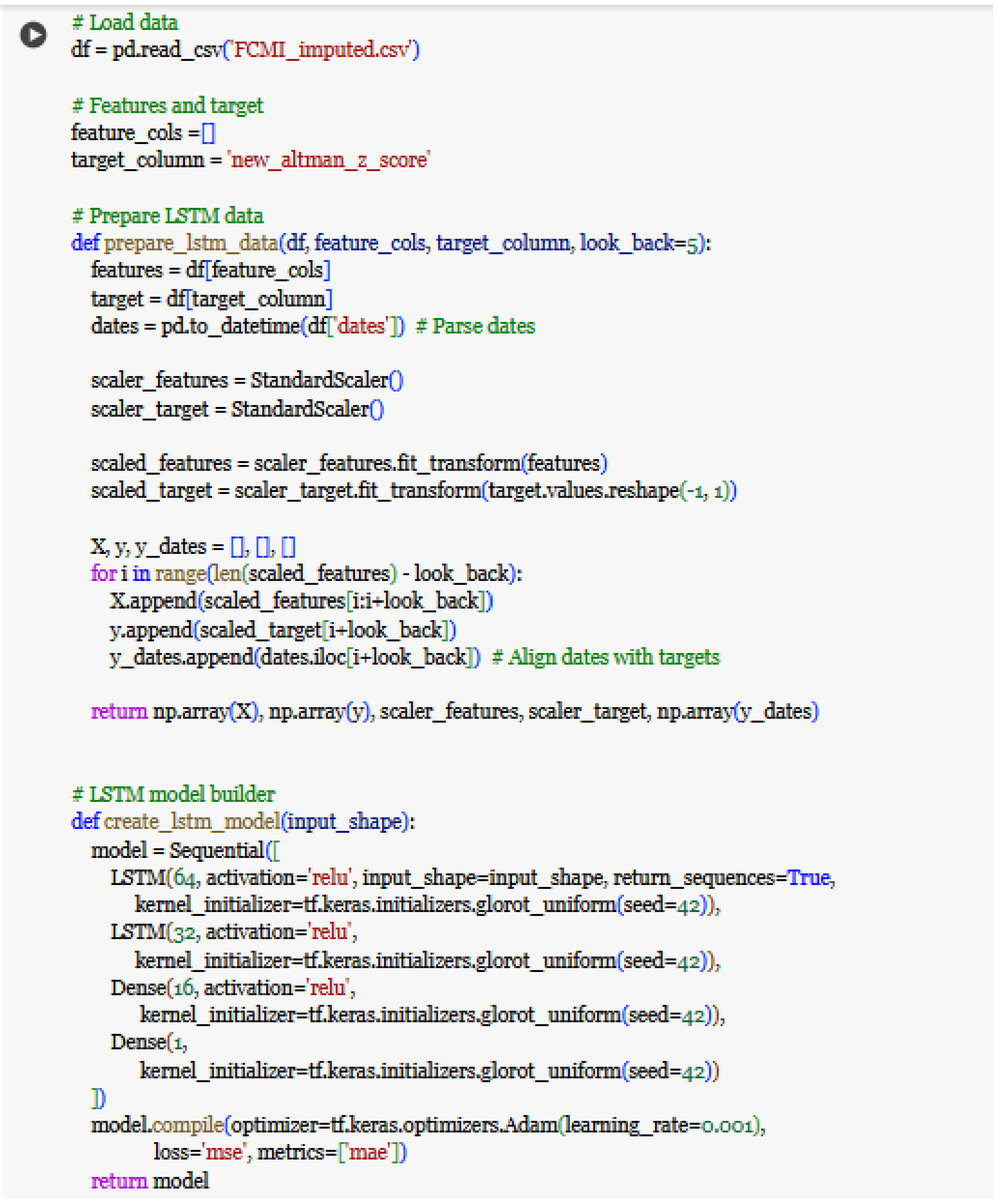
```
[ ] sentiment_df = sentiment_df.rename(columns={'Timestamp': 'dates'})
```

Data Aggregation

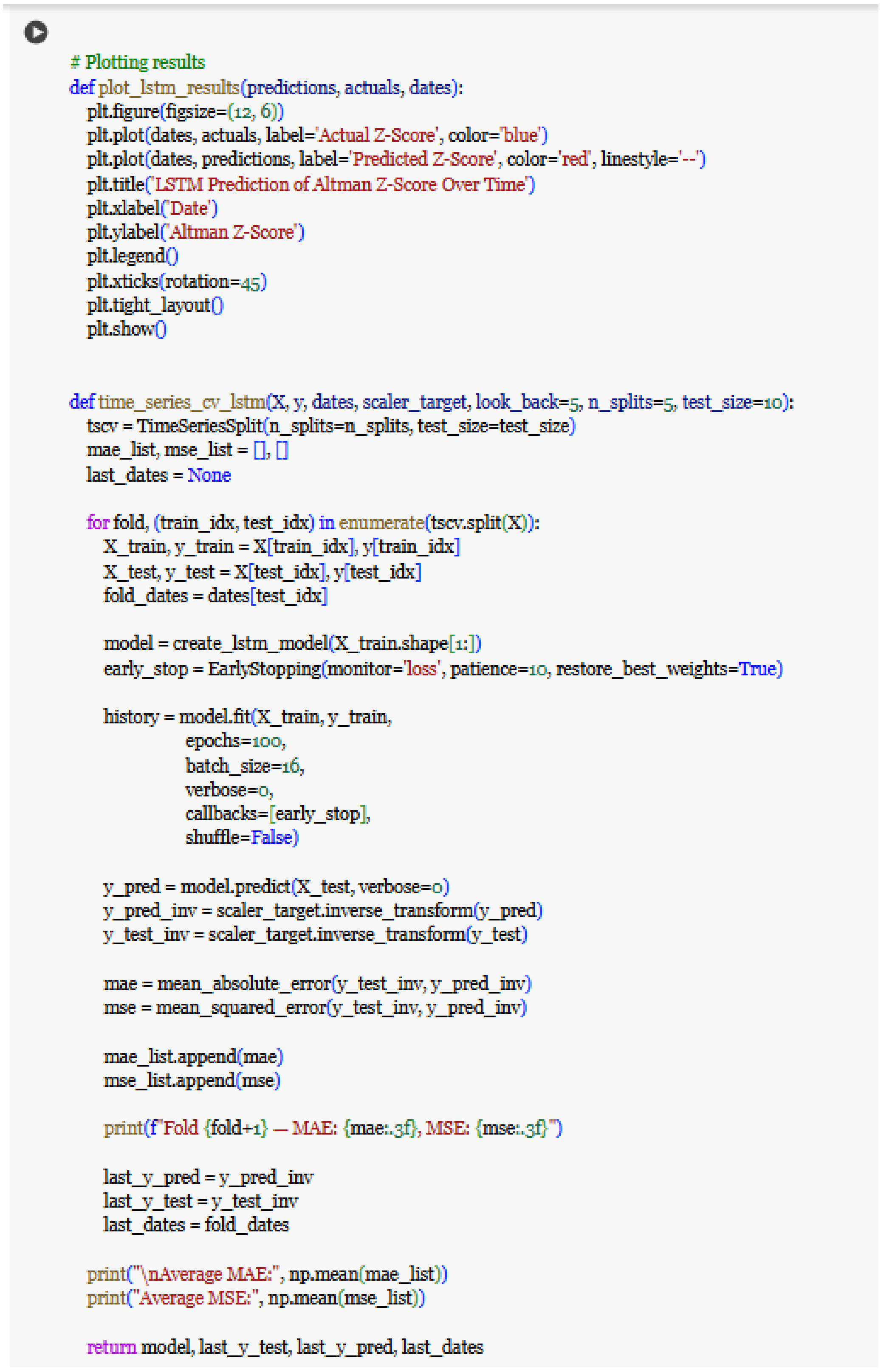
```python
def aggregate_sentiment_to_quarterly(sentiment_df, financial_dates):
    """
    Aggregate daily/monthly sentiment data to quarterly financial periods

    Args:
        sentiment_df: DataFrame with columns ['dates', 'sentiment_score']
        financial_dates: List of quarterly financial reporting dates

    Returns:
        DataFrame with quarterly sentiment metrics
    """
    # Convert to datetime and ensure proper format
    sentiment_df['dates'] = pd.to_datetime(sentiment_df['dates'])
    sentiment_df = sentiment_df.sort_values('dates')

    # Map sentiment to numerical values (if using categorical labels)
    sentiment_map = {'negative': 0, 'neutral': 1, 'positive': 2}
    sentiment_df['sentiment_score'] = sentiment_df['bert_sentiment'].map(sentiment_map)

    # Create quarterly bins based on financial reporting dates
    quarterly_sentiment = []

    for i in range(len(financial_dates)-1):
        start_date = financial_dates[i]
        end_date = financial_dates[i+1]

        period_data = sentiment_df[(sentiment_df['dates'] >= start_date) &
                                   (sentiment_df['dates'] < end_date)]

        if not period_data.empty:
            quarterly_stats = {
                'dates': end_date,  # Assign to the end of period
                'sentiment_mean': period_data['sentiment_score'].mean(),
                'sentiment_std': period_data['sentiment_score'].std(),
                'sentiment_count': len(period_data),
                'positive_ratio': (period_data['sentiment_score'] > 0).mean(),
                'negative_ratio': (period_data['sentiment_score'] < 0).mean()
            }
        else:
            # Handle missing data with neutral values
            quarterly_stats = {
                'dates': end_date,
                'sentiment_mean': 0,
                'sentiment_std': 0,
                'sentiment_count': 0,
                'positive_ratio': 0,
                'negative_ratio': 0
            }

        quarterly_sentiment.append(quarterly_stats)

    return pd.DataFrame(quarterly_sentiment)
```



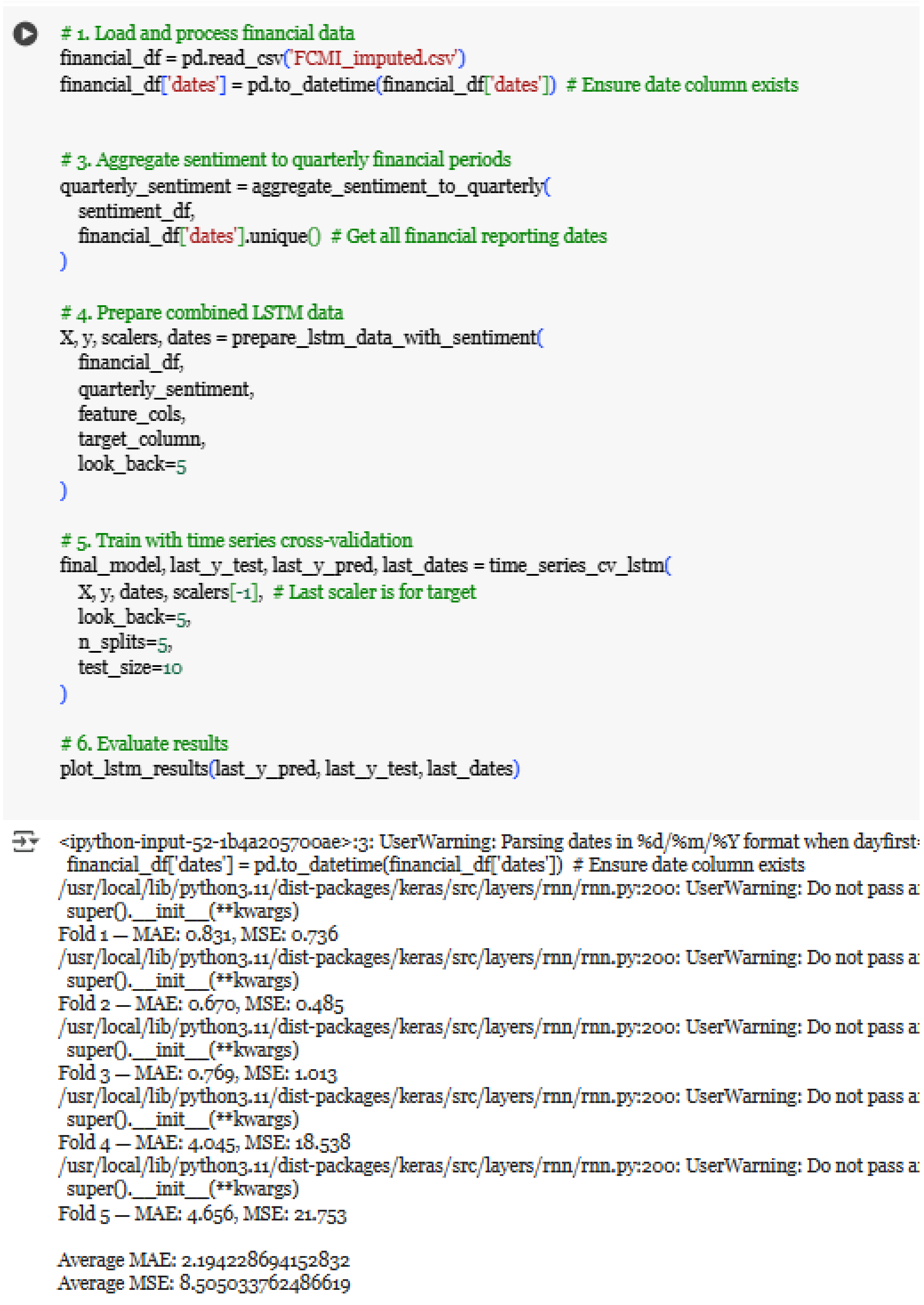
```python
def prepare_lstm_data_with_sentiment(df, sentiment_df, feature_cols, target_column, look_back=5):
    """
    Prepare LSTM data with integrated sentiment features, handling empty feature_cols.
    """
    # Merge sentiment data with financial data
    merged_df = pd.merge(df, sentiment_df, on='dates', how='left')

    # Fill any missing sentiment data (if no sentiment in quarter)
    sentiment_features = ['sentiment_mean', 'sentiment_std', 'sentiment_count',
                    'positive_ratio', 'negative_ratio']
    merged_df[sentiment_features] = merged_df[sentiment_features].fillna(0)

    # Get all features including sentiment (or only sentiment if feature_cols is empty)
    all_features = feature_cols + sentiment_features if feature_cols else sentiment_features
    features = merged_df[all_features]
    target = merged_df[target_column]
    dates = pd.to_datetime(merged_df['dates'])

    # Scaling
    scaler_features = StandardScaler()  # Even if empty, create for consistency
    scaler_target = StandardScaler()
    scaler_sentiment = StandardScaler()

    # Conditional scaling based on feature_cols
    if feature_cols:  # If financial features are present
        scaled_financial = scaler_features.fit_transform(features[feature_cols])
        scaled_sentiment = scaler_sentiment.fit_transform(features[sentiment_features])
        scaled_features = np.hstack([scaled_financial, scaled_sentiment])
    else:  # If only sentiment features are used
        scaled_features = scaler_sentiment.fit_transform(features)

    scaled_target = scaler_target.fit_transform(target.values.reshape(-1, 1))

    # Create sequences
    X, y, y_dates = [], [], []
    for i in range(len(scaled_features) - look_back):
        X.append(scaled_features[i:i+look_back])
        y.append(scaled_target[i+look_back])
        y_dates.append(dates.iloc[i+look_back])

    return np.array(X), np.array(y), (scaler_features, scaler_sentiment, scaler_target), np.array(y_dates)
```



```python
# Load data
df = pd.read_csv('FCMI_imputed.csv')

# Features and target
feature_cols =[]
target_column = 'new_altman_z_score'

# Prepare LSTM data
def prepare_lstm_data(df, feature_cols, target_column, look_back=5):
    features = df[feature_cols]
    target = df[target_column]
    dates = pd.to_datetime(df['dates'])  # Parse dates

    scaler_features = StandardScaler()
    scaler_target = StandardScaler()

    scaled_features = scaler_features.fit_transform(features)
    scaled_target = scaler_target.fit_transform(target.values.reshape(-1, 1))

    X, y, y_dates = [], [], []
    for i in range(len(scaled_features) - look_back):
        X.append(scaled_features[i:i+look_back])
        y.append(scaled_target[i+look_back])
        y_dates.append(dates.iloc[i+look_back])  # Align dates with targets

    return np.array(X), np.array(y), scaler_features, scaler_target, np.array(y_dates)


# LSTM model builder
def create_lstm_model(input_shape):
    model = Sequential([
        LSTM(64, activation='relu', input_shape=input_shape, return_sequences=True,
             kernel_initializer=tf.keras.initializers.glorot_uniform(seed=42)),
        LSTM(32, activation='relu',
             kernel_initializer=tf.keras.initializers.glorot_uniform(seed=42)),
        Dense(16, activation='relu',
              kernel_initializer=tf.keras.initializers.glorot_uniform(seed=42)),
        Dense(1,
              kernel_initializer=tf.keras.initializers.glorot_uniform(seed=42))
    ])
    model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
                  loss='mse', metrics=['mae'])
    return model
```

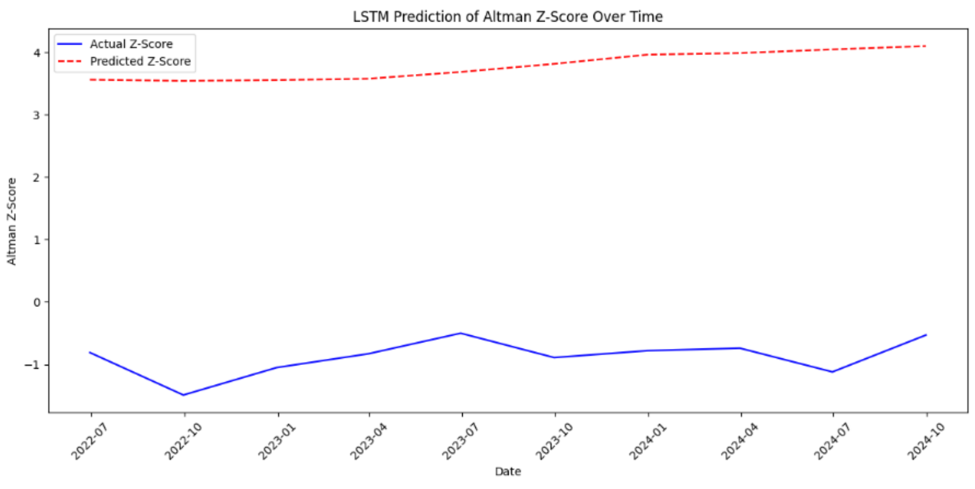


```
# Plotting results
def plot_lstm_results(predictions, actuals, dates):
    plt.figure(figsize=(12, 6))
    plt.plot(dates, actuals, label='Actual Z-Score', color='blue')
    plt.plot(dates, predictions, label='Predicted Z-Score', color='red', linestyle='--')
    plt.title('LSTM Prediction of Altman Z-Score Over Time')
    plt.xlabel('Date')
    plt.ylabel('Altman Z-Score')
    plt.legend()
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()


def time_series_cv_lstm(X, y, dates, scaler_target, look_back=5, n_splits=5, test_size=10):
    tscv = TimeSeriesSplit(n_splits=n_splits, test_size=test_size)
    mae_list, mse_list = [], []
    last_dates = None

    for fold, (train_idx, test_idx) in enumerate(tscv.split(X)):
        X_train, y_train = X[train_idx], y[train_idx]
        X_test, y_test = X[test_idx], y[test_idx]
        fold_dates = dates[test_idx]

        model = create_lstm_model(X_train.shape[1:])
        early_stop = EarlyStopping(monitor='loss', patience=10, restore_best_weights=True)

        history = model.fit(X_train, y_train,
                            epochs=100,
                            batch_size=16,
                            verbose=0,
                            callbacks=[early_stop],
                            shuffle=False)

        y_pred = model.predict(X_test, verbose=0)
        y_pred_inv = scaler_target.inverse_transform(y_pred)
        y_test_inv = scaler_target.inverse_transform(y_test)

        mae = mean_absolute_error(y_test_inv, y_pred_inv)
        mse = mean_squared_error(y_test_inv, y_pred_inv)

        mae_list.append(mae)
        mse_list.append(mse)

        print(f"Fold {fold+1} — MAE: {mae:.3f}, MSE: {mse:.3f}")

        last_y_pred = y_pred_inv
        last_y_test = y_test_inv
        last_dates = fold_dates

    print("\nAverage MAE:", np.mean(mae_list))
    print("Average MSE:", np.mean(mse_list))

    return model, last_y_test, last_y_pred, last_dates
```

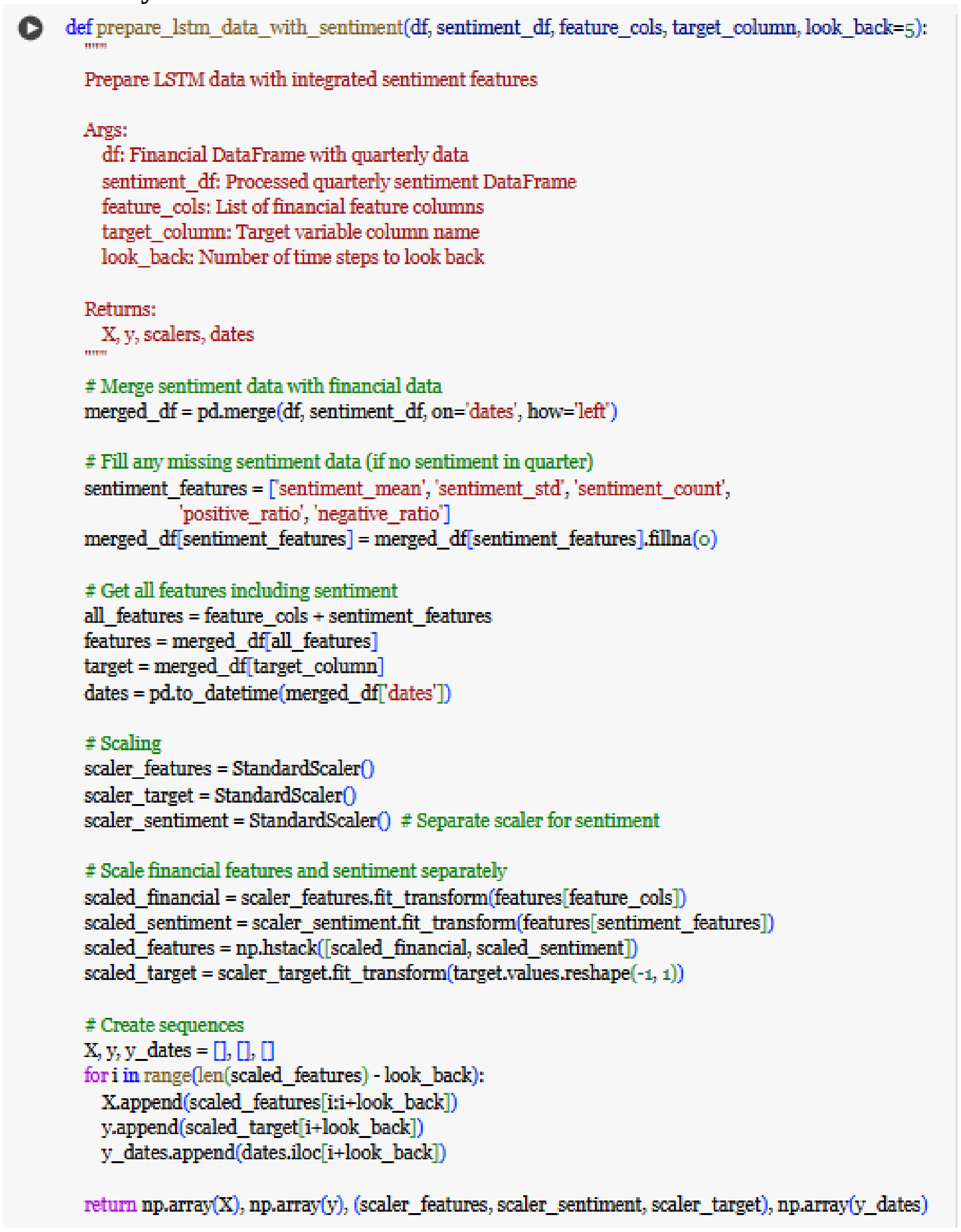


```
# 1. Load and process financial data
financial_df = pd.read_csv('FCMI_imputed.csv')
financial_df['dates'] = pd.to_datetime(financial_df['dates'])  # Ensure date column exists


# 3. Aggregate sentiment to quarterly financial periods
quarterly_sentiment = aggregate_sentiment_to_quarterly(
    sentiment_df,
    financial_df['dates'].unique()  # Get all financial reporting dates
)

# 4. Prepare combined LSTM data
X, y, scalers, dates = prepare_lstm_data_with_sentiment(
    financial_df,
    quarterly_sentiment,
    feature_cols,
    target_column,
    look_back=5
)

# 5. Train with time series cross-validation
final_model, last_y_test, last_y_pred, last_dates = time_series_cv_lstm(
    X, y, dates, scalers[-1],  # Last scaler is for target
    look_back=5,
    n_splits=5,
    test_size=10
)

# 6. Evaluate results
plot_lstm_results(last_y_pred, last_y_test, last_dates)
```

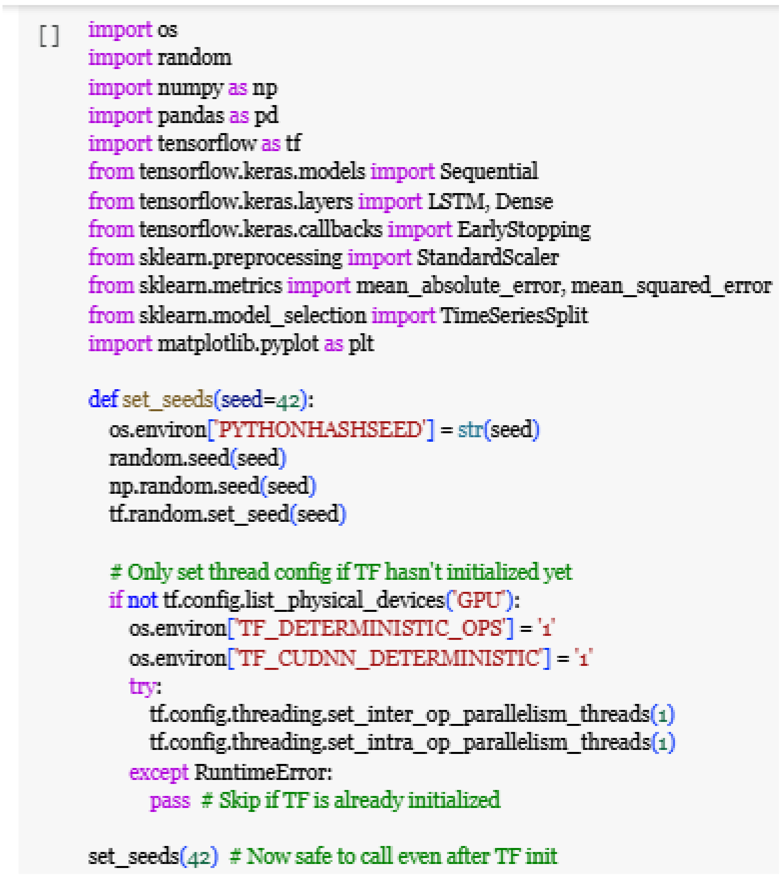
```
<ipython-input-52-1b4a205700ae>:3: UserWarning: Parsing dates in %d/%m/%Y format when dayfirst
  financial_df['dates'] = pd.to_datetime(financial_df['dates'])  # Ensure date column exists
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarning: Do not pass a
  super().__init__(**kwargs)
Fold 1 — MAE: 0.831, MSE: 0.736
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarning: Do not pass a
  super().__init__(**kwargs)
Fold 2 — MAE: 0.670, MSE: 0.485
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarning: Do not pass a
  super().__init__(**kwargs)
Fold 3 — MAE: 0.769, MSE: 1.013
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarning: Do not pass a
  super().__init__(**kwargs)
Fold 4 — MAE: 4.045, MSE: 18.538
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarning: Do not pass a
  super().__init__(**kwargs)
Fold 5 — MAE: 4.656, MSE: 21.753

Average MAE: 2.194228694152832
Average MSE: 8.505033762486619
```



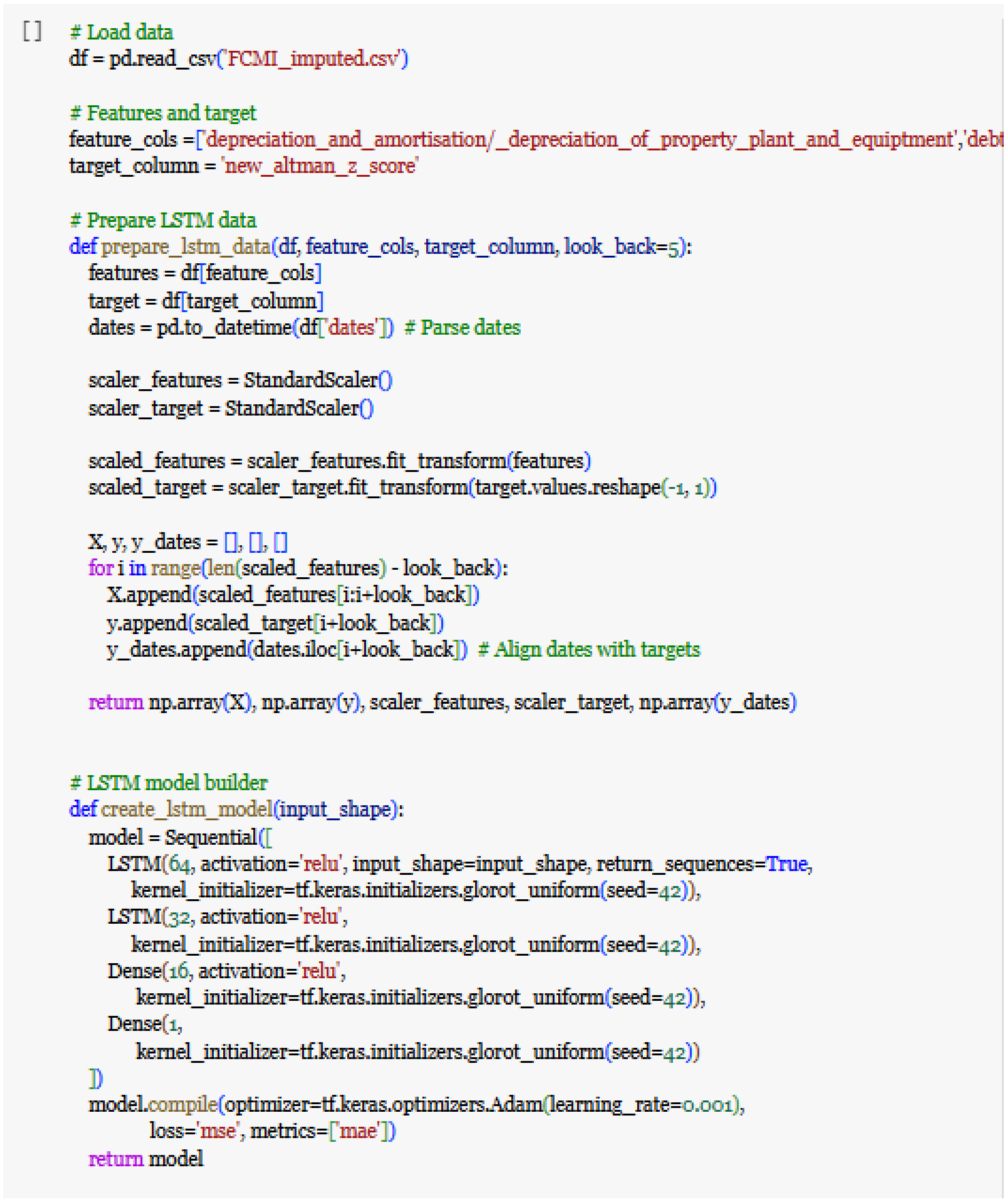
# Code for hybrid LSTM model



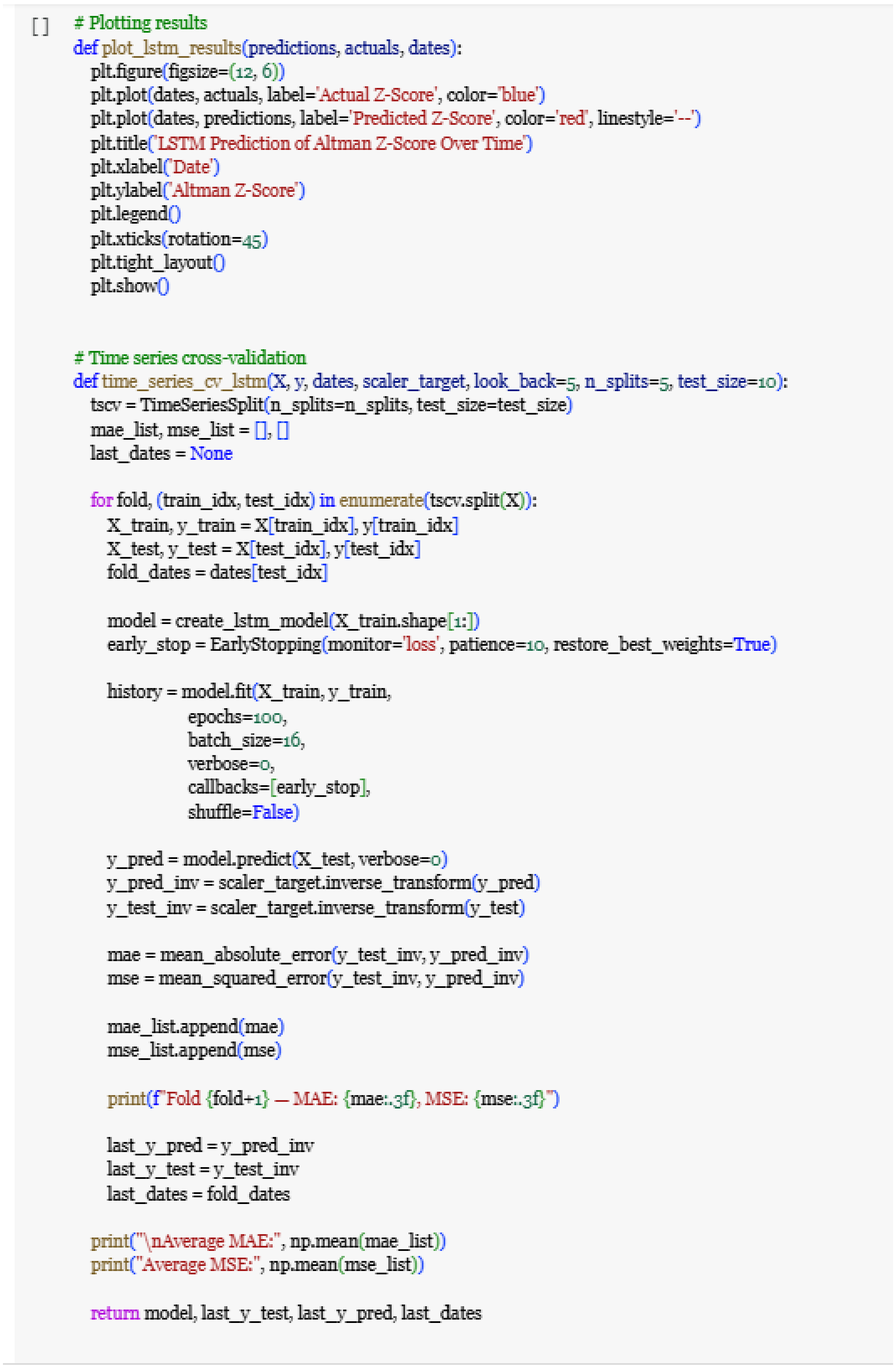
```
def prepare_lstm_data_with_sentiment(df, sentiment_df, feature_cols, target_column, look_back=5):
    """
    Prepare LSTM data with integrated sentiment features

    Args:
      df: Financial DataFrame with quarterly data
      sentiment_df: Processed quarterly sentiment DataFrame
      feature_cols: List of financial feature columns
      target_column: Target variable column name
      look_back: Number of time steps to look back

    Returns:
      X, y, scalers, dates
    """
    # Merge sentiment data with financial data
    merged_df = pd.merge(df, sentiment_df, on='dates', how='left')

    # Fill any missing sentiment data (if no sentiment in quarter)
    sentiment_features = ['sentiment_mean', 'sentiment_std', 'sentiment_count',
                'positive_ratio', 'negative_ratio']
    merged_df[sentiment_features] = merged_df[sentiment_features].fillna(0)

    # Get all features including sentiment
    all_features = feature_cols + sentiment_features
    features = merged_df[all_features]
    target = merged_df[target_column]
    dates = pd.to_datetime(merged_df['dates'])

    # Scaling
    scaler_features = StandardScaler()
    scaler_target = StandardScaler()
    scaler_sentiment = StandardScaler()  # Separate scaler for sentiment

    # Scale financial features and sentiment separately
    scaled_financial = scaler_features.fit_transform(features[feature_cols])
    scaled_sentiment = scaler_sentiment.fit_transform(features[sentiment_features])
    scaled_features = np.hstack([scaled_financial, scaled_sentiment])
    scaled_target = scaler_target.fit_transform(target.values.reshape(-1, 1))

    # Create sequences
    X, y, y_dates = [], [], []
    for i in range(len(scaled_features) - look_back):
      X.append(scaled_features[i:i+look_back])
      y.append(scaled_target[i+look_back])
      y_dates.append(dates.iloc[i+look_back])

    return np.array(X), np.array(y), (scaler_features, scaler_sentiment, scaler_target), np.array(y_dates)
```



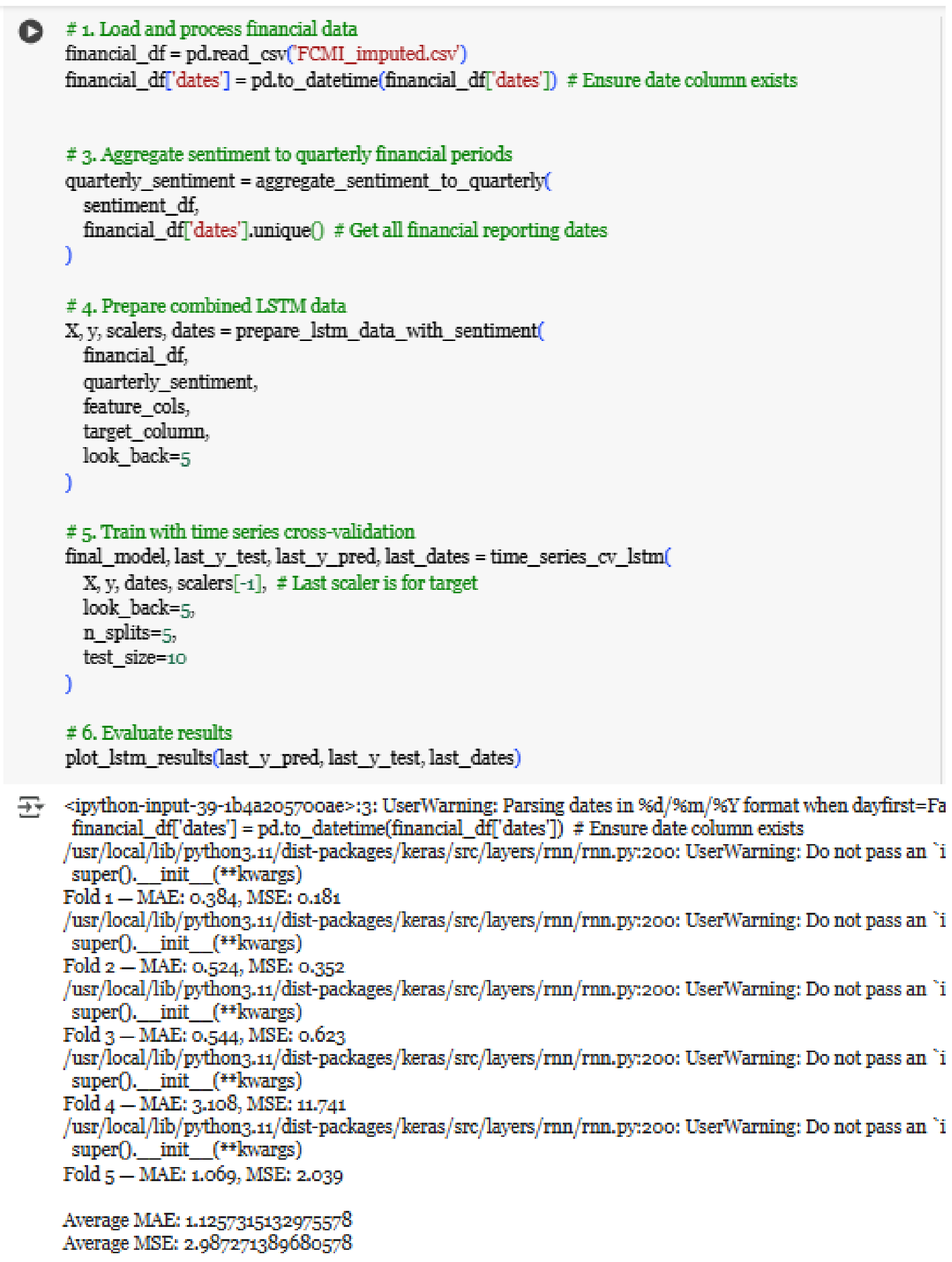
```
[ ]  import os
     import random
     import numpy as np
     import pandas as pd
     import tensorflow as tf
     from tensorflow.keras.models import Sequential
     from tensorflow.keras.layers import LSTM, Dense
     from tensorflow.keras.callbacks import EarlyStopping
     from sklearn.preprocessing import StandardScaler
     from sklearn.metrics import mean_absolute_error, mean_squared_error
     from sklearn.model_selection import TimeSeriesSplit
     import matplotlib.pyplot as plt

     def set_seeds(seed=42):
         os.environ['PYTHONHASHSEED'] = str(seed)
         random.seed(seed)
         np.random.seed(seed)
         tf.random.set_seed(seed)

         # Only set thread config if TF hasn't initialized yet
         if not tf.config.list_physical_devices('GPU'):
             os.environ['TF_DETERMINISTIC_OPS'] = '1'
             os.environ['TF_CUDNN_DETERMINISTIC'] = '1'
             try:
                 tf.config.threading.set_inter_op_parallelism_threads(1)
                 tf.config.threading.set_intra_op_parallelism_threads(1)
             except RuntimeError:
                 pass  # Skip if TF is already initialized

     set_seeds(42)  # Now safe to call even after TF init
```



```
[ ]  # Load data
     df = pd.read_csv('FCMI_imputed.csv')

     # Features and target
     feature_cols =['depreciation_and_amortisation/_depreciation_of_property_plant_and_equiptment','debt
     target_column = 'new_altman_z_score'

     # Prepare LSTM data
     def prepare_lstm_data(df, feature_cols, target_column, look_back=5):
         features = df[feature_cols]
         target = df[target_column]
         dates = pd.to_datetime(df['dates'])  # Parse dates

         scaler_features = StandardScaler()
         scaler_target = StandardScaler()

         scaled_features = scaler_features.fit_transform(features)
         scaled_target = scaler_target.fit_transform(target.values.reshape(-1, 1))

         X, y, y_dates = [], [], []
         for i in range(len(scaled_features) - look_back):
             X.append(scaled_features[i:i+look_back])
             y.append(scaled_target[i+look_back])
             y_dates.append(dates.iloc[i+look_back])  # Align dates with targets

         return np.array(X), np.array(y), scaler_features, scaler_target, np.array(y_dates)


     # LSTM model builder
     def create_lstm_model(input_shape):
         model = Sequential([
             LSTM(64, activation='relu', input_shape=input_shape, return_sequences=True,
                  kernel_initializer=tf.keras.initializers.glorot_uniform(seed=42)),
             LSTM(32, activation='relu',
                  kernel_initializer=tf.keras.initializers.glorot_uniform(seed=42)),
             Dense(16, activation='relu',
                   kernel_initializer=tf.keras.initializers.glorot_uniform(seed=42)),
             Dense(1,
                   kernel_initializer=tf.keras.initializers.glorot_uniform(seed=42))
         ])
         model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
                       loss='mse', metrics=['mae'])
         return model
```

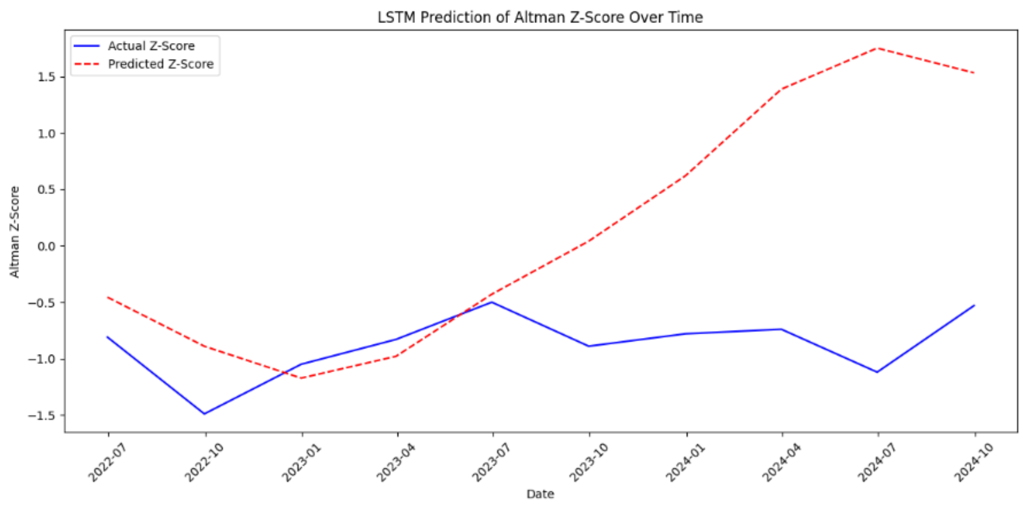


```
# Plotting results
def plot_lstm_results(predictions, actuals, dates):
    plt.figure(figsize=(12, 6))
    plt.plot(dates, actuals, label='Actual Z-Score', color='blue')
    plt.plot(dates, predictions, label='Predicted Z-Score', color='red', linestyle='--')
    plt.title('LSTM Prediction of Altman Z-Score Over Time')
    plt.xlabel('Date')
    plt.ylabel('Altman Z-Score')
    plt.legend()
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()


# Time series cross-validation
def time_series_cv_lstm(X, y, dates, scaler_target, look_back=5, n_splits=5, test_size=10):
    tscv = TimeSeriesSplit(n_splits=n_splits, test_size=test_size)
    mae_list, mse_list = [], []
    last_dates = None

    for fold, (train_idx, test_idx) in enumerate(tscv.split(X)):
        X_train, y_train = X[train_idx], y[train_idx]
        X_test, y_test = X[test_idx], y[test_idx]
        fold_dates = dates[test_idx]

        model = create_lstm_model(X_train.shape[1:])
        early_stop = EarlyStopping(monitor='loss', patience=10, restore_best_weights=True)

        history = model.fit(X_train, y_train,
                            epochs=100,
                            batch_size=16,
                            verbose=0,
                            callbacks=[early_stop],
                            shuffle=False)

        y_pred = model.predict(X_test, verbose=0)
        y_pred_inv = scaler_target.inverse_transform(y_pred)
        y_test_inv = scaler_target.inverse_transform(y_test)

        mae = mean_absolute_error(y_test_inv, y_pred_inv)
        mse = mean_squared_error(y_test_inv, y_pred_inv)

        mae_list.append(mae)
        mse_list.append(mse)

        print(f"Fold {fold+1} — MAE: {mae:.3f}, MSE: {mse:.3f}")

        last_y_pred = y_pred_inv
        last_y_test = y_test_inv
        last_dates = fold_dates

    print("\nAverage MAE:", np.mean(mae_list))
    print("Average MSE:", np.mean(mse_list))

    return model, last_y_test, last_y_pred, last_dates
```

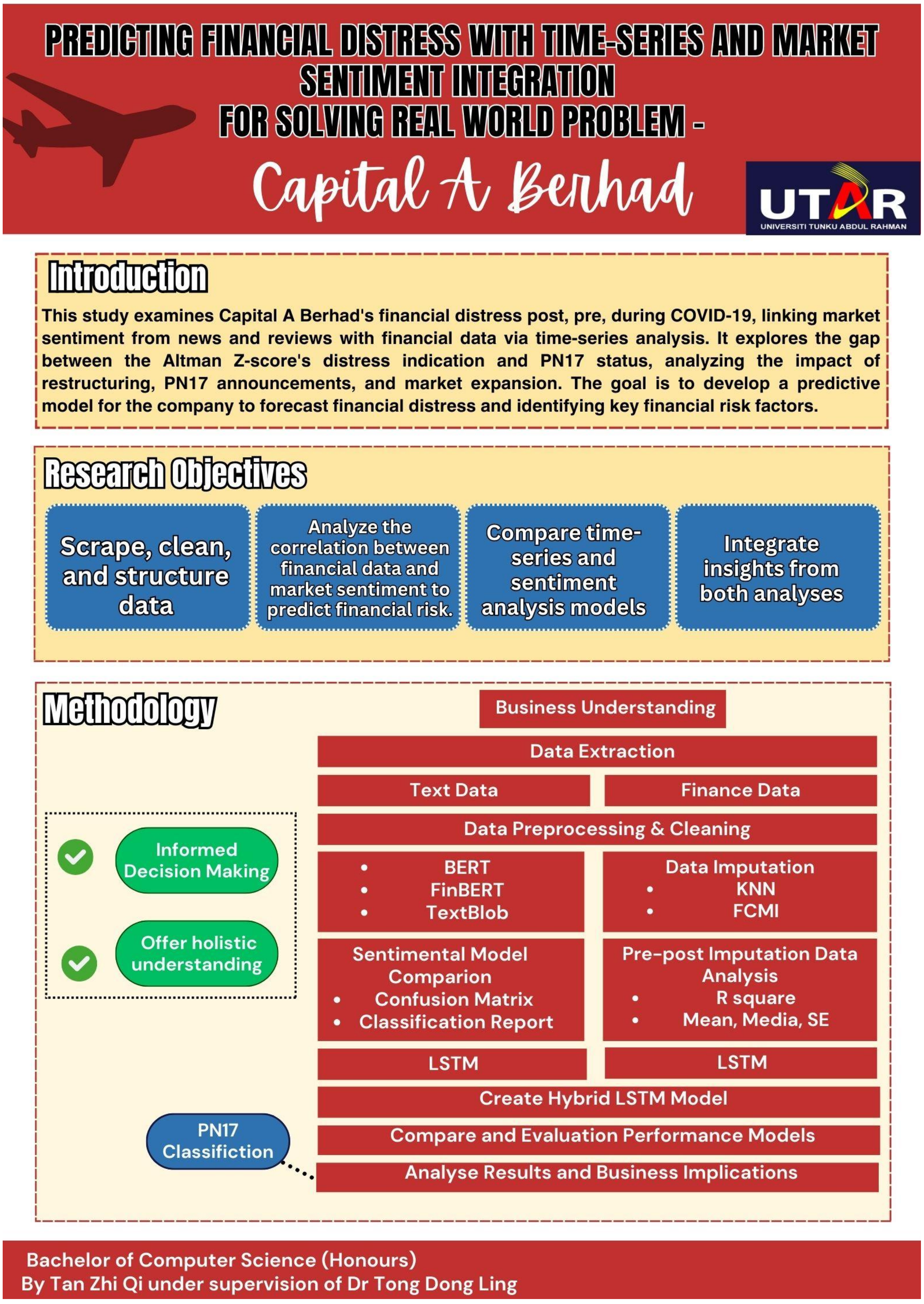


```
# 1. Load and process financial data
financial_df = pd.read_csv('FCMI_imputed.csv')
financial_df['dates'] = pd.to_datetime(financial_df['dates'])  # Ensure date column exists


# 3. Aggregate sentiment to quarterly financial periods
quarterly_sentiment = aggregate_sentiment_to_quarterly(
    sentiment_df,
    financial_df['dates'].unique()  # Get all financial reporting dates
)

# 4. Prepare combined LSTM data
X, y, scalers, dates = prepare_lstm_data_with_sentiment(
    financial_df,
    quarterly_sentiment,
    feature_cols,
    target_column,
    look_back=5
)

# 5. Train with time series cross-validation
final_model, last_y_test, last_y_pred, last_dates = time_series_cv_lstm(
    X, y, dates, scalers[-1],  # Last scaler is for target
    look_back=5,
    n_splits=5,
    test_size=10
)

# 6. Evaluate results
plot_lstm_results(last_y_pred, last_y_test, last_dates)
```

```
<ipython-input-39-1b4a205700ae>:3: UserWarning: Parsing dates in %d/%m/%Y format when dayfirst=Fa
  financial_df['dates'] = pd.to_datetime(financial_df['dates'])  # Ensure date column exists
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarning: Do not pass an `i
  super().__init__(**kwargs)
Fold 1 — MAE: 0.384, MSE: 0.181
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarning: Do not pass an `i
  super().__init__(**kwargs)
Fold 2 — MAE: 0.524, MSE: 0.352
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarning: Do not pass an `i
  super().__init__(**kwargs)
Fold 3 — MAE: 0.544, MSE: 0.623
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarning: Do not pass an `i
  super().__init__(**kwargs)
Fold 4 — MAE: 3.108, MSE: 11.741
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarning: Do not pass an `i
  super().__init__(**kwargs)
Fold 5 — MAE: 1.069, MSE: 2.039

Average MAE: 1.1257315132975578
Average MSE: 2.987271389680578
```



POSTER



PREDICTING FINANCIAL DISTRESS WITH TIME-SERIES AND MARKET SENTIMENT INTEGRATION FOR SOLVING REAL WORLD PROBLEM - Capital A Berhad

UTAR UNIVERSITI TUNKU ABDUL RAHMAN

Introduction

This study examines Capital A Berhad's financial distress post, pre, during COVID-19, linking market sentiment from news and reviews with financial data via time-series analysis. It explores the gap between the Altman Z-score's distress indication and PN17 status, analyzing the impact of restructuring, PN17 announcements, and market expansion. The goal is to develop a predictive model for the company to forecast financial distress and identifying key financial risk factors.

Research Objectives

- Scrape, clean, and structure data
- Analyze the correlation between financial data and market sentiment to predict financial risk.
- Compare time-series and sentiment analysis models
- Integrate insights from both analyses

Methodology

✓ Informed Decision Making

✓ Offer holistic understanding

PN17 Classifiction

Business Understanding

Data Extraction

Text Data | Finance Data

Data Preprocessing & Cleaning

- BERT
- FinBERT
- TextBlob

Data Imputation
- KNN
- FCMI

Sentimental Model Comparion
- Confusion Matrix
- Classification Report

Pre-post Imputation Data Analysis
- R square
- Mean, Media, SE

LSTM | LSTM

Create Hybrid LSTM Model

Compare and Evaluation Performance Models

Analyse Results and Business Implications

Bachelor of Computer Science (Honours)
By Tan Zhi Qi under supervision of Dr Tong Dong Ling