Aspect Based Sentiment Analysis On Achieving Sustainable Work Environments Through Recommendation Model Using Job Review

By

JASON YITRO SETIADI

TP062295

APD3F2311CS(DA)

A report submitted in partial fulfillment of the requirements for the degree of

B.Sc. (Hons) Computer Science with a specialism in Data Analytics

at Asia Pacific University of Technology and Innovation.

Supervised by Ts. Dr. LAW FOONG LI

2nd Marker: Ms. HEMA LATHA KRISHNA NAIR

31-JULY-2024

DECLARATION OF THESIS CONFIDENTIALITY

Author's full name: **JASON YITRO SETIADI**

IC No./Passport No.: **X1114982**

Thesis/Project title: **Aspect Based Sentiment Analysis On Achieving Sustainable Work Environments Through Recommendation Model Using Job Review**

I declare that this thesis is classified as:

☐ CONFIDENTIAL

☐ RESTRICTED

☒ OPEN ACCESS

I acknowledged that Asia Pacific University of Technology & Innovation (APU) reserves the right as follows:

1. The thesis is the property of Asia Pacific University of Technology & Innovation (APU).
2. The Library of Asia Pacific University of Technology & Innovation (APU) has the right to make copies for the purpose of research only.
3. The Library has the right to make copies of the thesis for academic exchange.

Author's Signature: 

Date: 20 June 2024

Supervisor's Name: Ts. Dr. Law Foong Li

Date: 25 July 2024

Signature: 

LIBRARY FORM

First Name: Jason
Middle Name (only if applicable): Yitro
Last Name: Setiadi
Title of the Final Year Project / Dissertation / Thesis: Aspect Based Sentiment Analysis On Achieving Sustainable Work Environments Through Recommendation Model Using Job Review
Abstract: In the quest to find the best companies and address HR challenges, employees often turn to job reviews for guidance. This project aimed to enhance workplace environments through aspect-based sentiment analysis of job reviews, utilizing the CRISP-DM methodology. Multiple machine learning models, including BERT, SVM, Logistic Regression, Random Forest, and Naive Bayes, were developed and evaluated to determine the most effective approach for sentiment analysis. BERT emerged at the top, achieving a remarkable accuracy of 97% with undersampling. However, its prolonged training and usage times make it less practical compared to faster traditional models, which reached approximately 96% accuracy with oversampling techniques like SMOTE. Extensive hyperparameter tuning further optimized model performance. These models were deployed using a user-friendly Streamlit app, enabling interactive exploration and analysis of sentiment data. The app provides actionable recommendations for companies to improve their HR practices and foster better workplace environments. By leveraging advanced sentiment analysis techniques and practical applications, this project contributes to achieving the Sustainable Development Goal 8, which promotes sustainable and inclusive economic growth, full and productive employment, and decent work for all. The integration of machine learning models with real-world data underscores the project's potential to drive positive change in workplace environments.
A few keywords associated with the work: Sentiment Analysis, Job Review, Machine Learning, Interpretable AI, Recommendation System
General Subject: Computer Software, Data Analytics, Natural Language Processing
Date of Submission: 31 July 2024

ACKNOWLEDGEMENT

I would like to express my deepest gratitude to all those who have played a pivotal role in the successful completion of my final year project on " Aspect Based Sentiment Analysis On Achieving Sustainable Work Environments Through Recommendation Model Using Job Review." This endeavor would not have been possible without the invaluable support and contributions of numerous individuals and resources.

First, I extend my sincere thanks to my project advisor Ts. Dr. Law Foong Li whose guidance and expertise have been instrumental in shaping the trajectory of this research. The insightful feedback, encouragement, and unwavering support provided have significantly enriched the quality of this project.

Additionally, I want to acknowledge the support from my family and friends who provided constructive feedback, engaged in insightful discussions, and offered encouragement during the various stages of this project. In conclusion, this project has been a collective effort, and I am profoundly grateful to everyone who has contributed to various capacities. Their support has made a lasting impact on the depth and quality of this research, and I am sincerely appreciative of everyone who has been part of this journey.

ABSTRACT

In the quest to find the best companies and address HR challenges, employees often turn to job reviews for guidance. This project aimed to enhance workplace environments through aspect-based sentiment analysis of job reviews, utilizing the CRISP-DM methodology. Multiple machine learning models, including BERT, SVM, Logistic Regression, Random Forest, and Naive Bayes, were developed and evaluated to determine the most effective approach for sentiment analysis. BERT emerged at the top, achieving a remarkable accuracy of 97% with undersampling. However, its prolonged training and usage times make it less practical compared to faster traditional models, which reached approximately 96% accuracy with oversampling techniques like SMOTE. Extensive hyperparameter tuning further optimized model performance. These models were deployed using a user-friendly Streamlit app, enabling interactive exploration and analysis of sentiment data. The app provides actionable recommendations for companies to improve their HR practices and foster better workplace environments. By leveraging advanced sentiment analysis techniques and practical applications, this project contributes to achieving the Sustainable Development Goal 8, which promotes sustainable and inclusive economic growth, full and productive employment, and decent work for all. The integration of machine learning models with real-world data underscores the project's potential to drive positive change in workplace environments.

Keywords: Sentiment Analysis, Job Review, Machine Learning Model, Interpretable AI, Recommendation System

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

1.1 Introduction

The job searching landscape has widened with the rapid development of technologies as many websites, applications, and even softwares that provide such services are coming up, for example LinkedIn, Indeed and many more to look for jobs, or. LinkedIn itself has more than 1 billion members in more than 200 countries and territories worldwide. However, there seems to be an imbalance as we no longer seek a higher salary or payout nowadays, however for a better work environment. There are no other people who know about one's inner company environment the best other than the ones who are in the said company itself. Hence, job reviews are becoming increasingly important. This is where Glassdoor shines as it allows people to find reviews of certain roles and companies.

Nowadays, employees pay more attention to the company's image and would spend more time doing their job review on third party websites rather than their annual performance review as some people felt that there are no direct benefits and/or advantages. Therefore, companies would rather spend most of their resources to appease their employees in hope they leave kinder reviews to allow more job seeker to apply in their companies.

Although their attention is good, some employees would view this as something false as they would not solve the problem to the root as they just rather hide the wound for a while. This is because the internal reviews done by HR departments are not analyzed very well. The process might look easier said than done as there are hundreds of thousands or even millions of reviews. Reviews of the annual performance of the job and company, specifically the role they have, have grown faster and bigger as these days, companies hire employees that are working on-site and remotely. Coming out of the pandemic, there are some people who changed their job from one industry to another and this impacted how HR review their review of job performance.

With size that has grown big, there is a need for automated process that could assess the sentiment of the review. The technology is called Sentiment Analysis and it used Natural Language Processing to remove all the redundancies and human involvement.

1.2 Problem Background

The process of annual performance reviews, for both HR and the employees, is tedious and long. This affects the review given by each employee as the question might be long and has no substantial. In average, it takes about two hundred and ten hours for managers to spend their time on annual performance reviews every year. That is about more than *five weeks* of productivity lost to the dreaded annual performance reviews for most companies.

Nowadays, annual performance reviews have a bad prejudice or label as they are more than just time-consuming, but also ineffective. With that in mind, 9 in 10 managers do not approve of their companies' annual performance review processes. Despite the hours and effort spent conducting performance reviews, they are sadly inaccurate as found by the management research firm Corporation Executive Board (CEB). Up to a whopping 77% of HR executives do not have the belief and confidence of the performance reviews accurately reflect employee contributions (Li, 2020).

This is a problem that has festered in all the industry as it is an integral part of work culture. Today, we are in an unprecedented time as not only have we just got out of a global pandemic of COVID-19, but also an economic crisis and racial tensions at an all-time high.

As job reviews are text that contain feeling and sentiments, it would be easy to overlook by the reviewers because they might be lazy to investigate or there are just too many of them that it would be a chore and a hassle to look in depth one by one. To tackle reviews that have grown sentiments and are harder to read, sentiment analysis is one way of solving this problem. This could also be solved by using the keyword-based recommendation approach by looking at the job review's overall topic trend, which could help simplify the analysis of the reviews.

The problem mentioned above concerns with the 8th goal of Sustainable Development Goal (SDG) which titled "Decent Work and Economic Growth." When thinking about jobs and the workforce, they often think that the fault is with the employees; however we would like to try solving the problem by looking at another way, which is the company and the environment they work. This can be done using text mining on the reviews left by the employees on company's environment and at the same time using the insights extracted from them, we can build a recommendations model that is able to analyze which factor is the negative and the positive to enable us to take the next step as an improvement.

1.3 Project Aim

The aim of the project is to analyze sentiment patterns in Glassdoor job reviews to gain insights into workplace experiences and contribute to fostering better work environments regarding HR impact and societal implications using Text Mining with Aspect Based Sentiment Analysis approach.

1.4 Objectives

The objectives of the investigation report are:

1. To investigate the impact of HR practices on the sustainability of work environments and the relationship between HR initiatives and employee sentiment and trends through job reviews.
2. To conduct a comparative and temporal investigation analysis of job reviews across different companies/industries to understand variances in sustainability practices and sentiment analysis model from different works.
3. To develop actionable recommendations model for HR departments to enhance workplace sustainability based on findings derived from approaches done using text mining through visualization using python libraries.
4. To evaluate the performance of the text mining machine learning models on sentiment analysis through job reviews from Glassdoor used in the Research Objective 3.

1.5 Scopes

1.5.1 Dataset

The dataset utilized for this project was obtained from the online Kaggle database, which lists job descriptions and ranks them according to several factors like culture, money, work-life balance, and so on. The dataset is excellent for multidimensional sentiment analysis because it also discusses the various industries, particularly in the United Kingdom. The review date, the job name, the job location, the reviewers' status, and the review itself are all included in the dataset's columns. The subcategories of Career Opportunities, Compensation & Benefits, Culture & Values, Senior Management, and Work/Life Balance comprise the reviews. Employee recommendations about the company, the CEO, and the forecast are also welcome.

1.5.2 Deliverables

Developing a sentiment analysis and a recommendations model for job review is a multi-step process that begins with data collection and ends with machine learning classification.

1. Data Understanding & Data Preprocessing:
    a) Data Collection:
        - Gather job review data specifically from Glassdoor.
        - Extract both the review text and any associated ratings or sentiment labels.
    b) Data Cleaning:
        - Remove irrelevant or noisy data (e.g., incomplete reviews, non-textual content).
        - Handle missing values, duplicates, and outliers.
        - Ensure consistent formatting across all reviews.
    c) Text Preprocessing:
        - Tokenization: Divide reviews up into discrete terms or concepts.
        - Lowercasing: For consistency, change all text to lowercase.
        - Stop Word Removal: Get rid of words that are frequently used but do not have any meaning, such as "the," "and" and "is".
        - Lemmatization and stemming involve reducing words to their most basic form (e.g., "running" to "run").
        - Eliminate all punctuation and special characters: Eliminate punctuation, emojis, and symbols from your writing.

2. Apply Text Mining and Natural Language Processing Techniques:
    a) Feature Extraction:
        - Make a matrix that shows the frequency of each word in the reviews using the simple **Counter** function.
        - Term Frequency-Inverse Document Frequency, or **TF-IDF** which transforms text into a meaningful representation of numbers which is used to fit machine learning algorithms for prediction.
    b) Data Labelling:
        - A simple and straightforward keyword extraction approach that uses BERT embeddings to generate keywords and keyphrases that are the most related to a document.
    c) Machine Learning Models:
        - Naive Bayes, Logistic Regression, or Random Forest: Train models using labeled data (reviews with sentiment labels) to predict sentiment.
        - Deep Learning Models (e.g., BERT): Utilize neural networks for more complex sentiment analysis.
3. Model Training:
    a) Split Data: Separate the training and validation sets from the dataset.
    b) Train the Sentiment Analysis Model: Fit the chosen model on the training data.
    c) Hyperparameter Tuning: Optimize model parameters using cross-validation.
    d) Recommendation Model:
        - If you have additional features (e.g., job category, company size), build a recommendation model to suggest relevant jobs based on sentiment scores.
        - Collaborative filtering or content-based filtering can be used for recommendations.
4. Model Evaluation:
    a) Validation/Test Set: Evaluate the sentiment analysis model on the validation/test set.
    b) Confusion Matrix, Accuracy, Precision, Recall, F1 scores.
    c) Assess model performance.
    d) Recommendation Evaluation: Measure the effectiveness of your recommendation model.

5. System Development:
    a) Create a user-friendly interface using Streamlit.
    b) Allow users to input user's chosen firm and/or company's name(s).
    c) Display sentiment analysis results (good/bad), job recommendations and more.

1.5.3 Constraint & Project Boundaries

Constraints

1. Time Constraints:
    a) Break down the project into manageable phases (data collection, preprocessing, model training, etc.) and carefully allocate time to each, ensuring completion within the overall timeline.
2. Resource Constraints:
    a) Factor in computational resources (CPU, memory, storage) required for training and deploying models within the allotted time to avoid costly delays or performance bottlenecks.
    b) Ascertain budget limitations and choose cloud services or hardware accordingly.
3. Ethical Constraints:
    a) Ensure ethical handling of customer feedback data, such as de-identifying personal information and adhering to data privacy regulations.
    b) Minimize biases in sentiment analysis of customer feedback by employing fairness-aware algorithms and regularly monitoring for demographic or other biases.

What Will Be Done as Part of the Project

1. Data Collection:
    a) Gather job reviews from Glassdoor.
    b) Extract review text and associated ratings.
2. Data Preprocessing: Clean and preprocess the text data (tokenization, lowercasing, stop word removal, etc.).
3. Sentiment Analysis Model and Analysis: Train a sentiment analysis model using natural language processing.
4. Recommendation Model: Develop a recommendation system based on sentiment scores and additional features (e.g., job category, company size).

5. Visualization and Interpretation of Result:
    a) Create a user-friendly visualization interface using Streamlit.
    b) Display sentiment analysis results and job recommendations with breakdowns with comparison like companies and/or industries.

What Will Not Be Done as Part of the Project

1. Advanced Deep Learning Architectures: While we will use basic deep learning models, we will not explore complex architectures (e.g., advanced transformers) due to time constraints.
2. Custom Lexicon Creation: We will not create a domain-specific sentiment lexicon from scratch.
3. User Authentication and Security: User authentication and data security will not be covered in this project.
4. Large-Scale Deployment: We will not optimize for high traffic or scalability initially.

1.6 Potential Benefits

1.6.1 Tangible benefits

Tangible benefit refers to measurable outcomes or advantages that can be directly observed or quantified. The defining factors of this project include:

a) Improved Hiring Decisions: Analyzing job reviews can gain insights into candidate experiences, helping them make more informed hiring decisions.
b) Enhanced Employee Satisfaction: Identifying positive sentiments in reviews can highlight aspects of the workplace that contribute to employee satisfaction.
c) Optimized Recruitment Strategies: Quantifying sentiments can guide recruitment efforts by identifying areas for improvement.
d) Increased Productivity: Addressing negative sentiments can lead to a more productive workforce.

1.6.2 Intangible benefits

Intangible benefits refer to non-measurable outcomes or advantages that impact feelings, perceptions, or values, but cannot be easily quantified, which include:

a) Employee Morale: Positive reviews contribute to a positive work environment and boost employee morale.
b) Company Reputation: Monitoring sentiments helps maintain a positive brand image and reputation.

c) Cultural Insights: Understanding intangible aspects of company culture from reviews can guide organizational improvements.

1.6.3 Target users

The target users for this sentiment analysis project would include:

a) Human Resources (HR) Professionals: They can use the sentiment analysis model to enhance recruitment processes and improve employee experiences.

b) Managers and Team Leads: Insights from sentiment analysis can guide team management and foster a positive work environment.

c) Executives and Decision-Makers: Recommendations based on sentiment analysis can inform strategic decisions related to employee satisfaction and retention.

1.7 Overview of the IR

The opening chapter serves as the cornerstone of the report, elucidating the project's core and anticipated outcomes. It commences with an introduction, proceeds to pinpoint the problem and objectives of the project, and delineates measurable goals while defining the project's boundaries and enumerating its advantages. Moreover, it discusses the intended users and culminates with the project's roadmap.

Chapter 2, titled "Literature Review," conducts an extensive examination of pertinent literature to underpin the project. It starts with an introduction, proceeds with domain research covering broad subjects to specific theories and technologies, evaluates similar systems highlighting their features and effectiveness, and conducts technical research detailing potential system requirements and technical tools. The chapter concludes with a succinct summary encapsulating its major findings.

Chapter 3, labeled "Methodology," elucidates the research and system development approach adopted for the project. It kicks off with an introduction followed by an in-depth exploration of the selected methodology, then progresses to discuss data collection, understanding, and preprocessing. The chapter wraps up with a summary providing a cohesive understanding of the project's methodological framework.

The last chapter, "Conclusion," brings closure to the initial phase of the project. It begins by discussing the achievements of the project's initial part, evaluates the depth and sufficiency of the research and investigation concerning the project's objectives, and addresses potential shortcomings in the research and design domains while suggesting areas for further exploration and improvement. This chapter acts as a reflective checkpoint, assessing progress and charting future directions.

1.8 Project Plan

Table 1 Project Plan of the Final Year Project

Name of Tasks	Duration	Start Date	End Date	Status
DECLARATION OF THESIS CONFIDENTIALITY	1 day	Mon 7/22/24	Mon 7/22/24	Completed
LIBRARY FORM	2 days	Sun 7/21/24	Mon 7/22/24	Completed
ACKNOWLEDGMENT	1 day	Sun 7/21/24	Sun 7/21/24	**Completed**
ABSTRACT	**1 day**	**Wed 3/13/24**	**Wed 3/13/24**	**Completed**
CHAPTER 1: INTRODUCTION	**43 days**	**Sun 1/14/24**	**Tue 3/12/24**	**Completed**
1.1 Introduction	17 days	Sun 1/14/24	Mon 2/5/24	Completed
1.2 Problem Background	17 days	Sun 1/14/24	Mon 2/5/24	Completed
1.3 Project Aim	17 days	Sun 1/14/24	Mon 2/5/24	Completed
1.4 Objectives	17 days	Sun 1/14/24	Mon 2/5/24	Completed
1.5 Scope	21 days	Sun 1/14/24	Fri 2/9/24	Completed
1.5.1 Dataset	7 days	Sun 1/14/24	Mon 1/22/24	Completed
1.5.2 Deliverables	7 days	Tue 1/23/24	Wed 1/31/24	Completed
1.5.3 Constraint & Project Boundaries	7 days	Thu 2/1/24	Fri 2/9/24	Completed
1.6 Potential Benefits	21 days	Sun 1/14/24	Fri 2/9/24	Completed
1.6.1 Tangible Benefits	7 days	Sun 1/14/24	Mon 1/22/24	Completed
1.6.2 Intangible Benefits	7 days	Tue 1/23/24	Wed 1/31/24	Completed
1.6.3 Target Users	7 days	Thu 2/1/24	Fri 2/9/24	Completed
1.7 Overview of the IR	1 day	Tue 3/12/24	Tue 3/12/24	Completed
1.8 Project Plan	39 days	Sun 1/14/24	Wed 3/6/24	Completed
CHAPTER 2: LITERATURE REVIEW	**122 days**	**Sat 2/3/24**	**Sun 7/21/24**	**Completed**
2.1 Introduction	6 days	Sat 2/3/24	Fri 2/9/24	Completed
2.2 Domain Research	12 days	Sun 2/4/24	Sun 2/18/24	Completed
2.2.1 Sentiment Analysis in Text Analytics on Job Review	1 day	Sun 2/4/24	Sun 2/4/24	Completed
2.2.1.1 Document-Level Sentiment Analysis	1 day	Sun 2/4/24	Sun 2/4/24	Completed
2.2.1.2 Sentence-Level Sentiment Analysis	1 day	Sun 2/4/24	Sun 2/4/24	Completed
2.2.1.3 Aspect-Based Sentiment Analysis	1 day	Sun 2/4/24	Sun 2/4/24	Completed
2.2.2 Machine Learning for Sentiment Analysis	1 day	Mon 2/5/24	Mon 2/5/24	Completed
2.2.3 Interpretable AI-Text Recommendation System	1 day	Tue 2/6/24	Tue 2/6/24	Completed
2.3 Similar Works	12 days	Sun 2/4/24	Sun 2/18/24	Completed
2.4 Technical Research	111 days	Mon 2/19/24	Sun 7/21/24	Completed
2.4.1 Interactive Development Environment (IDE)	1 day	Mon 2/19/24	Mon 2/19/24	Completed
2.4.2 Libraries and/or Tools	**110 days**	**Tue 2/20/24**	**Sun 7/21/24**	**Completed**
2.4.2.1 Libraries	53 days	Thu 5/9/24	Sun 7/21/24	Completed
2.4.2.2 Tools	52 days	Fri 5/10/24	Sun 7/21/24	Completed
2.5 Summary	1 day	Wed 2/21/24	Wed 2/21/24	Completed
CHAPTER 3: METHODOLOGY	**15 days**	**Wed 2/21/24**	**Tue 3/12/24**	**Completed**
3.1 Introduction	1 day	Wed 2/21/24	Wed 2/21/24	Completed
3.2 System Development Methodology	3 days	Wed 3/6/24	Fri 3/8/24	Completed
3.2.1 Introduction of CRISP-DM Methodology	1 day	Wed 3/6/24	Wed 3/6/24	Completed

3.2.2 Methodology Choice and Justification	1 day	Thu 3/7/24	Thu 3/7/24	Completed
3.2.3 Activities and process in each phase toward CRISP-DM in detail	1 day	Fri 3/8/24	Fri 3/8/24	Completed
3.3 Summary	1 day	Tue 3/12/24	Tue 3/12/24	Completed
CHAPTER 4: DESIGN AND IMPLEMENTATION	**98 days?**	**Wed 3/13/24**	**Fri 7/26/24**	**Completed**
4.1 Introduction	1 day	Wed 3/13/24	Wed 3/13/24	Completed
4.2 Data Collection	1 day	Wed 3/6/24	Wed 3/6/24	Completed
4.3 Data Understanding	**53 days**	**Thu 3/7/24**	**Sat 5/18/24**	**Completed**
4.2.1.1 Basic Data Understanding	28 days	Thu 3/7/24	Mon 4/15/24	Completed
4.2.1.2 Advanced Data Understanding	25 days	Tue 4/16/24	Sat 5/18/24	Completed
4.4 Data Pre-Processing	**98 days?**	**Sat 3/9/24**	**Tue 7/23/24**	**Completed**
4.4.1 Missing Values	2 days	Sat 3/9/24	Mon 3/11/24	Completed
4.4.2 Replace Values	4 days	Tue 3/12/24	Fri 3/15/24	Completed
4.4.3 Noise and NaN Values	3 days	Sat 3/16/24	Tue 3/19/24	Completed
4.4.4 Data Transformation	4 days	Wed 3/20/24	Sun 3/24/24	Completed
4.4.5 Data Normalization - Lemmatization	3 days	Mon 6/3/24	Wed 6/5/24	Completed
4.4.6 Data Labelling Bert Model	1 day	Tue 7/23/24	Tue 7/23/24	Completed
4.4.7 Data Post-Processing	1 day	Tue 7/23/24	Tue 7/23/24	Completed
4.4.8 Data Selection (Deletion)	33 days	Thu 6/6/24	Sun 7/21/24	Completed
4.4.9 Data Balancing	**23 days**	**Wed 6/19/24**	**Sun 7/21/24**	Completed
4.4.9.1 Data Balancing using Python Script	4 days	Wed 6/19/24	Sat 6/22/24	Completed
4.4.9.2 Data Balancing using SMOTE	24 days	Wed 6/19/24	Sun 7/21/24	Completed
4.5 Model Buildings	**19 days**	**Tue 7/2/24**	**Fri 7/26/24**	**Completed**
4.5.1 BERT Classification Model	27 days	Sat 5/25/24	Mon 7/1/24	Completed
4.5.2 SVM Classification Model	16 days	Tue 7/2/24	Tue 7/23/24	Completed
4.5.3 Logistic Regression Classification Model	16 days	Tue 7/2/24	Tue 7/23/24	Completed
4.5.4 Random Forest Classification Model	16 days	Tue 7/2/24	Tue 7/23/24	Completed
4.5.5 Naïve Bayes Classification Model	19 days	Tue 7/2/24	Fri 7/26/24	Completed
4.6 Summary	2 days	Thu 7/25/24	Fri 7/26/24	Completed
CHAPTER 5: RESULTS AND DISCUSSIONS	**11 days?**	**Sat 7/13/24**	**Mon 7/29/24**	**Completed**
5.1 Introduction	1 day	Sun 7/21/24	Sun 7/21/24	Completed
5.2 Model Evaluations and Discussions	**11 days?**	**Sat 7/13/24**	**Fri 7/26/24**	Completed
5.2.1 BERT Hyperparameter	3 days	Sat 7/13/24	Tue 7/16/24	Completed
5.2.2 SVM Hyperparameter	2 days	Sat 7/13/24	Sun 7/14/24	Completed
5.2.3 Logistic Regression Hyperparameter	2 days	Sat 7/13/24	Sun 7/14/24	Completed
5.2.4 Random Forest Hyperparameter	2 days	Sat 7/13/24	Sun 7/14/24	Completed
5.2.5 Naïve Bayes Hyperparameter	2 days	Sat 7/13/24	Sun 7/14/24	Completed
5.3 Model Deployment	**11 days**	**Mon 7/15/24**	**Mon 7/29/24**	**Completed**
5.3.1 Pickles for the Model Deployment	2 days	Mon 7/15/24	Tue 7/16/24	Completed
5.3.2 StreamLit Deployment	11 days	Mon 7/15/24	Mon 7/29/24	Completed
5.3.3 LIME Explanation Result Comparison	11 days	Mon 7/15/24	Mon 7/29/24	Completed
5.4 Summary	1 day	Fri 7/26/24	Fri 7/26/24	Completed
CHAPTER 6: CONCLUSION	**2 days**	**Fri 7/26/24**	**Mon 7/29/24**	Completed
6.1 Critical Evaluation	**2 days**	**Fri 7/26/24**	**Mon 7/29/24**	**Completed**
6.1.1 Overall Project Achievement	1 day	Fri 7/26/24	Fri 7/26/24	Completed

6.1.2 Contribution of the Project Towards Communities and/or Industries	1 day	Fri 7/26/24	Fri 7/26/24	Completed
6.1.3 Strength of the Project	1 day	Fri 7/26/24	Fri 7/26/24	Completed
6.2 Limitations	1 day	Fri 7/26/24	Fri 7/26/24	Completed
6.3 Recommendations	2 days	Fri 7/26/24	Mon 7/29/24	Completed
REFERENCES	127 days	Thu 2/1/24	Fri 7/26/24	Completed
APPENDICES	**173 days**	**Mon 12/4/23**	**Wed 7/31/24**	**Completed**
Appendix A: PPF	15 days	Mon 12/4/23	Fri 12/22/23	Completed
Appendix B: Ethics Forms (Fast Track)	5 days	Tue 2/20/24	Mon 2/26/24	Completed
Appendix C: Log Sheets	130 days	Wed 1/31/24	Tue 7/30/24	Completed
Appendix D: Poster	2 days	Sat 7/27/24	Mon 7/29/24	Completed
Appendix E: Gantt Chart	101 days	Fri 3/8/24	Fri 7/26/24	Completed
Appendix F: Sample Code Implementation	1 day	Fri 7/26/24	Fri 7/26/24	Completed
Appendix G: Respondence Demographic Profile	3 days	Fri 7/26/24	Tue 7/30/24	Completed

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

Before going deeper into Literature Review, research questions and gaps written below are needed to understand what the student will explain in deeper details:

Research Questions:

1. To investigate how an employee's job reviews as well as ratings and/or sentiment can be used for sentiment analysis toward ranking companies based on their reputations.
2. To analyze different models of sentiment analysis algorithms and their accuracy based on similar works journals.
3. To implement the analyzed and chosen sentiment analysis model on the chosen dataset.
4. To evaluate the effectiveness of the given dataset toward the sentiment analysis model and recommendations model.

Research Gaps:

1. A few data analysis projects on job review are based on the company's performance review ratings rather than using the comments made by the employees themselves using text analytics. No journals have been researching on correlation between questions such as ratings from 1 to 5 and text comments on job review and solely on one source to see their satisfaction.
2. A thorough investigation and complete layout mapping of the available text analytics' model used for sentiment analysis with each of their benefits and disadvantages.
3. Analysis on job performance review is always done with its focus for the company, however, there is little to no analysis done for the employees' benefits. This could be tackled using the available predictive sentiment analysis with the new recommendations model.
4. Through recommendations model, companies can make actionable improvements on their programs based on employees' reviews.

For businesses and employees alike, job happiness is critical to establishing the ideal environment in which everyone benefits and advances together (Heimerl et al., 2020). Employee satisfaction is more than simply a nice-to-have in the workplace. Increased job satisfaction predicts better financial results for businesses, employee retention and consumer

loyalty. Another study revealed that higher levels of job satisfaction are associated with higher levels of dedication, loyalty, ownership, effectiveness, efficiency, and productivity(B. Vuong et al., 2021). The two studies provided above show a powerful information that job satisfaction is that important for both company and employee, however sadly, job satisfaction level is different from company to company or even countries.

Table 2 2020 Survey on Job Satisfaction by Ranstad (https://shorturl.at/enuL8)

Country	% of very satisfied employees	Country	% of very dissatisfied employees
India	89	Japan	21
Mexico	85	Hong Kong	11
Türkiye	80	Singapore	10
Norway	79	Hungary	10
USA	78	Czech Republic	8
Denmark	78	Greece	8
Spain	77	Sweden	10
Argentina	75	France	12
Brazil	74	Portugal	14
China	74	New Zealand	10

Second and third-world countries' workforces are satisfied, however a striking difference can be seen from Table 2 that most of the dissatisfied employees are from first-world countries. This is because the way companies shape the environment, and employees' priorities are sometimes different and conflicting starting from where the company is located to what characteristics the employees have.

A set of data obtained from 1043 questionnaires—administered to participants with an average age of 35.24 years—showed that just 30% of a well-educated population is content with his employment. Additionally, of all the studied samples, 12% experience intimidation from their superior occasionally, and 23% find it difficult to get out of bed and go to work (Montuori et al., 2022). The job satisfaction might be high in developing countries might be caused by the lack of job satisfaction surveys on companies and the concept is new and foreign, hence the low high percentage of satisfaction as it is caused by an imbalance data (Anh et al., 2018).

However, one thing that is the same across different countries which is employees' sentiments, whether it is positive or negative, poured toward the company and this trend happens all around the world until now which is a problem that keeps persisting despite valiant and various efforts from companies.

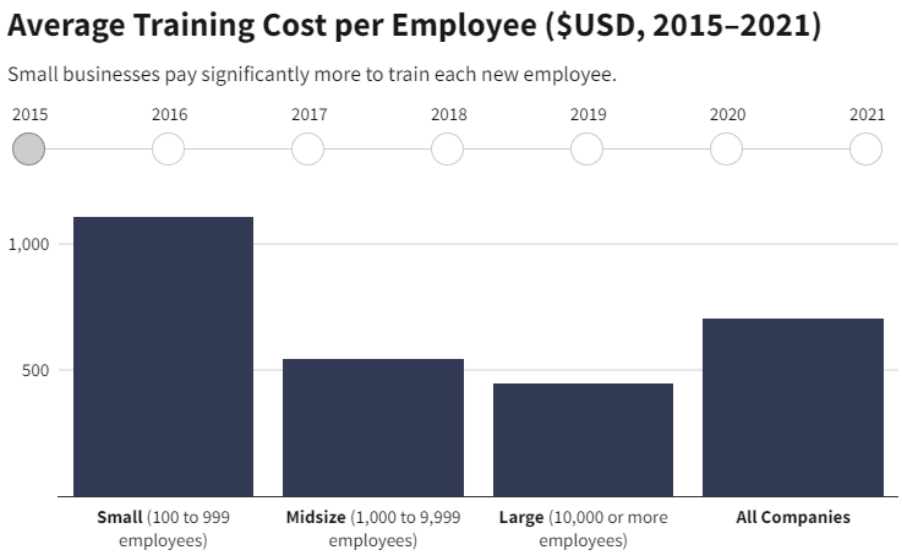


Figure 1 Average Training Cost per Employee (https://shorturl.at/jwG18)

Most companies have put more than hundreds of millions of dollars to further better their employee system. Figure 1 shows the cost of training exceeds more than $92.3 billion in 2020 to 2021. The money spent on training and hiring is specifically intended to boost workers' happiness in their new workplaces and settings. They even go as far as to implement various job training programs based on the demands of professions to improve the wellbeing of their workforce (Shin et al., 2020). It can take a company up to six months or longer to break even on its investment in a recruit, thus integrating a new hire into the team can also be time- and money-consuming (Mueller, 2022).

To make sure employees' job satisfaction is high and at the same time, lowering companies' investments on job training, a job review is often done every half a year to 1-year. An employee's strengths and shortcomings should be noted in a job review to inspire them to do better. For this reason, it ought to provide future-focused input to the business and its staff (Gnepp et al., 2020). Employee inspiration and performance evaluation are positively correlated, as proved by employee performance measurement. Plans for performance evaluations must be created with the goals of the employees and their fellow teammates in mind (T. D. N. Vuong & Nguyen, 2022). This is why companies should create an excellent job review for employees that they can take advantage of as data is valuable for future predictions and prescriptive improvements.

Thus, the student is interested in finding out what influences workers' contentment. Factors including age, gender, education, occupation, commute time, and difficulty, as well as low income, appear to be associated with job satisfaction since they tend to affect people's

expectations and preferences when it comes to how they view their working conditions which is why satisfied employees are more valuable than anything (Judge et al., 2020). The characteristics mentioned above are related to the job review the employees send to the company as they would like for them to satisfy their requirements. Employees' characteristics are connected to job satisfaction, and it is represented through job review.

Job reviews revealed many factors that impacted employees, therefore there is a need to analyze job satisfaction by extracting sentiments through job review in an automated and quick manner. After Machine Learning (ML) has been integrated and used in text form, it has branched into Natural Language Processing (NLP) and gave birth to Text Mining which is the perfect technique when it comes to sentiment analysis. But how can this apply to job review use case?

As the sentiments of the job reviews have been predicted, a solution should also be prescribed and recommended to the problem found which is gaining insight into the dataset inputted. A recommendation model nowadays is needed as a mandatory requirement as the new age of machine learning has come with Artificial Intelligence (AI). Not only predicting, but also recommending what actions companies should proceed with to fix what is wrong by breaking down the positive and applying it on the negative.

Through the above explanations, several questions come to mind when doing sentiment analysis on job reviews which the student will explain deeper one by one through this chapter two (2) of the Final Year Project.

1. What categories and/or keywords should the sentiment analysis model look for and what sentiment score should it be categorized to. Does the keyword reflect a company's strengths or weaknesses? After categorizing all the reviews based on its topic, how accurate is the score/weight compared to the ratings given by employees for each Topic.
2. Which text mining methods can be applied to support both predictive and prescriptive sentiment analysis model.
3. How are the results of the analysis being implemented and applied to a real-world problem.
4. What kind of results should be expected after applying the sentiment analysis and recommendations model.

2.2 Domain Research

2.2.1 Sentiment Analysis in Text Analytics on Job Review

The domain of Text Analytics is preferred by people who would like to analyze their text data. It is tremendously beneficial to apply the concept on huge scale text data where it might take months for humans to do the same thing. This is because most data are in text form, for example books, articles, news and so on (Mehta & Pandya, 2020). There are three famous approaches to Sentiment Analysis with their own pros and cons.

Approaches	Classification	Advantages	Disadvantages
Machine Learning Approach	• Supervised and Unsupervised learning.	• Dictionary is not necessary. • Demonstrate the high accuracy of classification.	• Classifier trained on the texts in one domain in most cases does not work with other domains.
Rule Based Approach	• Supervised and Unsupervised learning.	• Performance accuracy of 91% at the review level and 86% at the sentence level. • Sentence level sentiment classification performs better than the word level.	• Efficiency and accuracy depend the defining rules.
Lexicon Based Approach	• Unsupervised learning.	• Labelled data and the procedure of learning is not required.	• Requires powerful linguistic resources which is not always available.

Figure 2 Comparisons of Three Approaches (Devika et al., 2016)

To measure employee and/or job satisfaction through performance and/or job review, this satisfaction or sentiment could be described using numbers usually starting from one (1) to five (5) or ten (10), as well as using sentences which can be as long as a couple of sentences. To understand hundreds of thousands of those data, a sentiment analysis model will be needed that could classify them in high accuracy while having a lot of data fed into it because those numbers will represent a lot of topics and sentiments.

Typically, job reviews aim to assess four major areas that can be further subdivided into smaller aspects. The first is job satisfaction, which encompasses reputation of the organization, working hours, and circumstances. Coworkers are involved in employee relationships, which comes second. The third is pay, benefits, and company culture. These comprise income, incentives, advancements, chances for growth, and a positive work environment. The final one is employee loyalty to the business since contented workers have a propensity to remain devoted to their employer.

However, as work satisfaction indicates an employee's propensity to stick with a company, it can impact their decision to leave. To tackle this problem, Sentiment Analysis is widely

used to monitor reviews, especially trends, like employee and elections' sentiments and so on. SA itself comes in various levels of analysis:

2.2.1.1 Document-Level Sentiment Analysis

Document-level sentiment analysis is typically done using supervised learning techniques. Existing supervised approaches, such as Naive Bayes and Support Vector Machines, can be easily adapted to sentiment categorization. The first effort on automatic sentiment categorization at the document level divided movie reviews from IMDB into two categories: positive and negative. Classification features include keywords and their frequency, part of speech tags, opinion words, grammatical dependency, and negation.

In addition to binary sentiment prediction, models were developed to predict Yelp review rating ratings. Because the rating ratings are ordinal, the problem was transformed into a regression problem and addressed using SVM regression. An unsupervised method is suggested for sentiment classification, based on established syntactic patterns that are likely to be used by other reviewers (Lyu et al., 2020).

2.2.1.2 Sentence-Level Sentiment Analysis

At the sentence level, each sentence in the document is evaluated and labeled as positive or bad. The methods resemble document-level sentiment analysis. Sentence-level sentiment analysis can employ rules based on the clauses in a sentence. Sentiment classification does not attempt to identify concrete features that were commented on as a result, the granularity of analysis differs from that of aspect-based sentiment analysis. Improving on a single-sentence classification, by applying an auxiliary approach using BERT, pair-sentence classification is produced solves the performance issue of BERT in text classification tasks (Lin & Moh, 2021).

2.2.1.3 Aspect-Based Sentiment Analysis

Opinions based on a single sentence or the paper itself sometimes lack detailed specifics about the sentiment. Certain applications require feedback on specific qualities or components of the object (e.g., what people liked and disliked). These specifications are required for such applications.

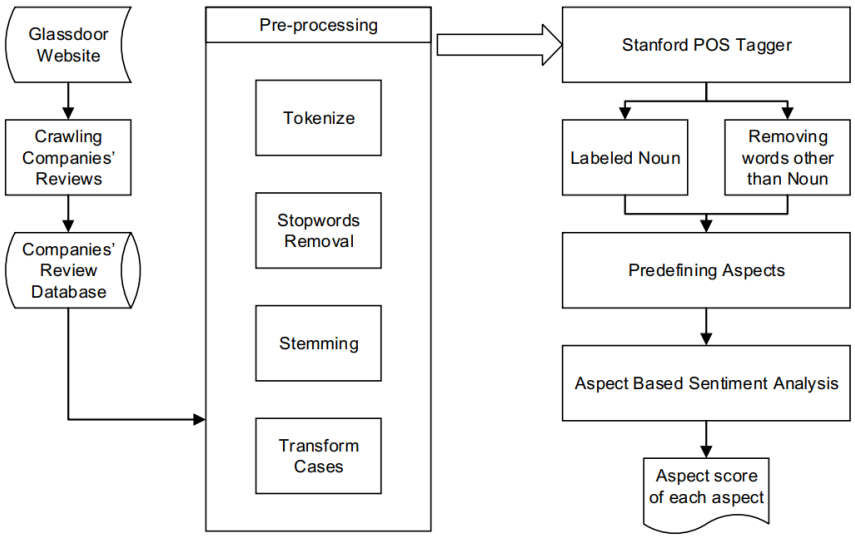


Figure 3 Sentiment Analysis Research Procedure Example (Dina & Juniarta, 2020)

Aspect-level sentiment analysis provides a fine-grained analysis. It is based on the idea that an opinion consists of a target of view and a sentiment, whether positive or negative (Dina & Juniarta, 2020). Conflicting opinions, such as "I love this restaurant even though the service isn't that great," can be resolved by better understanding the sentiment issue. The tone of this statement is positive, yet the service element is negative.

2.2.2 Machine Learning Models for Sentiment Analysis

To train an algorithm on a training dataset before applying it to the real dataset, the machine learning approach—the subject of this study—depends on the training dataset. Two datasets—the training and testing datasets—are used in the machine learning categorization of SA. These datasets are used by the classification system to test its efficiency and acquire new dataset knowledge. While the training dataset is designed to be utilized primarily for dataset learning, the testing dataset serves as a means of verifying that the approach is performing as planned.

Machine learning (ML) tackles sentiment classification in two ways: supervised learning and unsupervised learning. A training dataset including the input label and score is used in the supervised learning technique. By employing classification techniques, it makes it possible for the classification model to learn. It is also applied to forecast the value of fresh inputs. However, the labelled dataset is not used by the unsupervised learning model. Datasets with a variety of inputs are used to train it (Harfoushi et al., 2018).

Table 3 Common Machine Learning Models for Sentiment Analysis (Kaur & Sharma, 2022)

No.	ML Model	Explanation	Accuracy	
1.	Logistic Regression	An approach that is frequently used for binary classification jobs is logistic regression. It forecasts whether a certain text conveys a positive or negative sentiment in sentiment analysis. It makes use of a logistic function to model the likelihood of the result.	Logistic regression can achieve high accuracy, especially when the relationship between the input features and the target variable is relatively linear and the data is well-structured. - **High**	Supervised
2.	Support Vector Machine (SVM)	SVM is an effective method for jobs involving regression and classification. It creates a hyperplane in sentiment analysis to distinguish between positive and negative feelings. Maximizing the margin between classes is the goal of SVM.	SVM can achieve high accuracy when finely tuned. It works well with both linear and non-linear data. However, SVM's performance heavily depends on the choice of kernel functions and hyperparameters. - **High**	Supervised
3.	Naive Bayes	The probabilistic algorithm Naive Bayes assumes that features, given the class label, are conditionally independent. It performs very well for text categorization tasks, despite its "naive" premise.	Naive Bayes is computationally efficient and performs well for sentiment analysis. It is particularly useful when dealing with large text datasets. - **Moderate**	Supervised
4.	Random Forest	Multiple decision trees are combined in the ensemble learning technique known as Random Forest. The ultimate forecast is decided by the majority vote cast by each tree regarding the class label.	Random Forest can handle non-linear relationships and noisy data. It provides robustness against overfitting. Its accuracy depends on the number of trees and feature selection. – **Moderate**	Supervised
5.	Deep Learning Models (e.g., LSTM, CNN)	Sentiment analysis has made deep learning models like Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks prominent. CNNs extract local features, while LSTMs collect sequential dependencies.	Deep learning models excel when trained on large labeled datasets. They can learn complex patterns but require substantial computational resources. - **High**	Supervised and Unsupervised
6.	FastText	FastText is an extension of Word2Vec that considers subword information (character n-grams). It is efficient and performs well for text classification tasks.	FastText is particularly useful for sentiment analysis on social media data. It achieved high accuracy in Twitter sentiment analysis. - **High**	Supervised and Unsupervised

Table 3 above is a summarization of the most common machine learning models used in Sentiment Analysis which are mostly supervised learning models. This is because a sentiment analysis model classifies the text into positive or negative (and sometimes neutral) sentiments in its most basic form. Therefore, naturally the most successful approaches are using supervised models that need a fair amount of labelled data to be trained. Providing such data is an expensive and time-consuming process that is not possible or readily accessible in many cases. Additionally, the output of such models is a number implying how similar the text is to the positive examples provided during the training and does not consider nuances such as sentiment complexity of the text (Devika et al., 2016). For further details with sentiment analysis works will be explained in 2.3 Similar Works.

2.2.3 Interpretable AI-Text Recommendation System

The next step in developing a prescriptive sentiment analysis model is to implement Interpretable AI, which not only provides accurate sentiment predictions but also provides insights into how and why those predictions were made. This is especially significant for applications where understanding the reasons behind the model's judgments is critical, such as in the legal, medical, or financial realms, where accountability and transparency are important (Pavitha et al., 2023).

Biases from training data are implicitly inherited by machine learning models which may lead to biased machine learning algorithms that discriminate against certain groups. For example, covid-19 symptoms, which are frequently seen in a variety of illnesses, or the automatic acceptance or denial of credit applications may be used to target a minority population. When this happens, interpretability serves as a tool for amendment because the user can visualize the underlying process and model (Jawale & Sawarkar, 2020). Therefore, there are things that should be paid attention to when building an Interpretable Sentiment Analysis model:

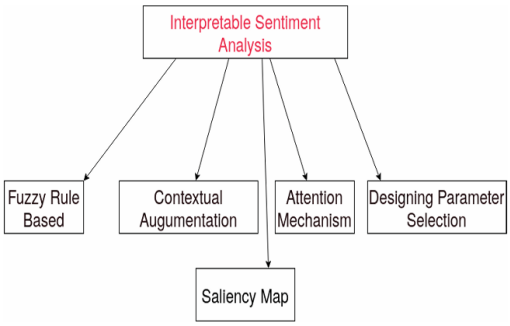


Figure 4 Interpretable Sentiment Analysis (Jawale & Sawarkar, 2020)

1. Feature Importance: Techniques such as feature importance analysis can help identify which words or features contribute the most to the sentiment prediction. This could involve methods like calculating TF-IDF (Term Frequency-Inverse Document Frequency) scores or using techniques like SHAP (SHapley Additive exPlanations) values.

2. Interpretable Models: Instead of using complex models like deep neural networks, simpler models like decision trees or logistic regression can be employed. These models are more transparent and easier to interpret, as they directly show how input features lead to predictions.

3. Attention Mechanisms: In the context of neural networks, attention mechanisms can highlight which parts of the input text are most influential in making the sentiment prediction. This provides insights into which words or phrases the model is focusing on.

4. Rule-based Systems: High interpretability can be achieved by using rule-based systems where sentiment is assessed using pre-established rules. Sentiment scores can be assigned to words, for instance, using sentiment lexicons or dictionaries. Based on the scores of the words that make up a text, the overall sentiment of the text can be ascertained.

5. Visualization: Visualizations such as word clouds, bar charts showing the frequency of sentiment-bearing words, or heatmaps illustrating the attention weights of different words can aid in understanding how the model is making its predictions.

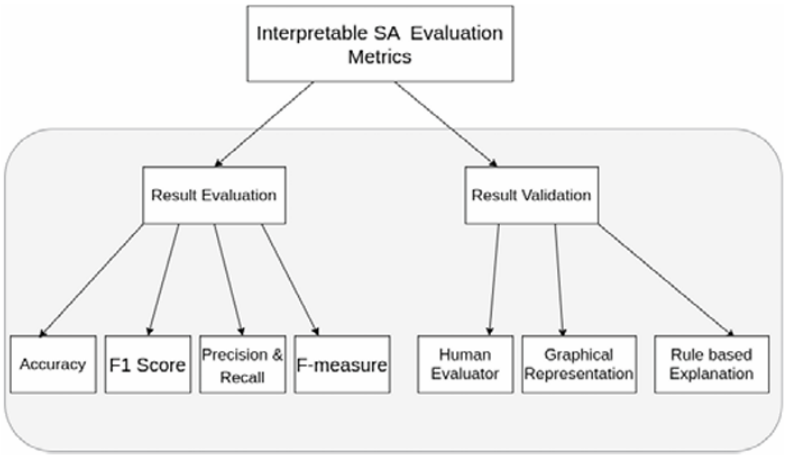


Figure 5 Interpretable SA Evaluation Metrics (Jawale & Sawarkar, 2020)

6. Human-in-the-loop Approaches: Integrating human judgment or feedback into the model training and prediction process can enhance interpretability. For instance, models can be designed to provide explanations along with predictions, allowing users to understand and potentially correct misclassifications.

By employing these approaches, developers can create sentiment analysis models that not only offer accurate predictions but also provide explanations and insights that are understandable and actionable for end-users (Chun & Elkins, 2023).

2.3 Similar Works

Table 4 Similar Systems / Works: Text Analytics Model Comparison Table

Authors	Description	Dataset Used	Analysis Techniques	Evaluation Techniques	Outcomes	Limitations
(Puspitasari, 2023)	Key areas of employee satisfaction and unhappiness are revealed by analyzing both positive and negative opinions using sophisticated data analytics technologies. Following that, the results underwent opportunity mapping, which identified certain areas in need of organizational development.	9,000 unidentified Accenture employee reviews are available on Glassdoor.	Sentiment Analysis: VADER Topic Modeling: LDA	Opportunity Mapping	According to Accenture, the most popular subjects were management, learning opportunities, rewards, and work-life balance. Accenture discovered that while learning opportunities were quite rewarding, aspects pertaining to work-life balance, management, and incentives were not very satisfactory, based on the results of opportunity mapping.	Further investigations can examine the influence of additional categorical variables (such department, gender, and age) through the mapping of sentiment analysis and topic modeling findings. This inquiry's results will clarify how these factors affect worker satisfaction.
(Xu & Li, 2016)	We use a text mining technique called latent semantic analysis to examine online customer reviews of hotels. Our research gives hotel owners guidance on how to improve customer happiness and reduce dissatisfaction among their clientele by meeting their needs and meeting their expectations for the various kinds of hotels they operate.	According to the most recent US Census Bureau population estimate, reviews from 580 hotels in the 100 largest U.S. cities were gathered.	Latent Semantic Analysis (LSA) using RapidMiner	Factor Analysis	Because LSA is based on mathematics, it is a more objective method than other methods for reviewing reviews and doing summaries. The ability of LSA to function as a theory and a technique for deriving and expressing the meaning of human language is one of its main advantages. The study contrasts the factors that influence patron happiness and discontent from the viewpoint of several hotel kinds.	Future research might compare the factors that influence customer happiness and discontent from the standpoint of the demographics of the customers. Secondly, the client demographic category that the model affects. Lastly, to strike a balance in the development between costs and benefits.
(Alduayj & Smith, 2019)	To predict the candidates' feelings following a job interview, machine learning and neural network models—such as logistic regression, support vector machines, Naïve Bayes, and long short-term memory (LSTM)—were developed.	3,983 Glassdoor responses summarizing the general Amazon hiring process interview experience. There are 1,983 negative and 2,000 positive reviews in the sample.	SVM (Support Vector Machines) Logistic Regression Naïve Bayes Random Forest KNN	Five-fold cross-validation.	Consequently, an F1-measure of 0.814 was obtained when training logistic regression using TF–IDF and unigram word representation. Bigram use reduces classifier performance; on the other hand, unigram, trigram, and bigram all improve classifier performance. It was discovered that performance is impacted by the feature weighting selection, and TF-IDF performed exceptionally well for the classification tasks.	-
(Mouli et al., 2023)	An investigation was carried out on social media to find out how satisfied employees were and related issues.	The majority of the dataset's sources were IT companies. There were 156,428 records collected prior to under-sampling.	Binary classification models: GloVe Embedding Layer and Bi-GRU Classifier. Multiclass classification models: BERT Embedding Layer, Relu, SoftMax, and Dropout.	A variety of assessment techniques, including F1-Score, Precision, and Recall, will be applied.	Using the Bert model for multiclass datasets, which has an accuracy of 95%, and Bi-GRU for binary class datasets, which has an overall accuracy of 97%, is the optimal method to arrive at the conclusion. Because Bi-GRU is bidirectional, meaning it can predict both past and future contexts, it performs better than other models. However, BERT can handle	There is no detailed explanation of the factors that affect the accuracy or how it is determined. This study's machine learning models underperformed because they were not optimized.

					long-term dependencies in sequences and understand text context.	
(Miranda & Sassi, 2014)	To suggest a method that will help a Brazilian online job search company evaluate customer satisfaction by using sentiment analysis and Python software.	Real data were collected from a Brazilian online job search company.	Classification method using Repustate API.	Evaluation done using Pearson's correlation coefficient.	The negligible connection between the customer's score and the sentiment score indicates how crucial it is to examine the customer's perception of the services they received. It is interesting to note that both methods of measuring customer satisfaction are complementary and do not preclude one another.	Encountered difficulty locating Sentiment Analysis tools for analyzing Portuguese texts. This adds another translation stage as well. To increase the usability and value of sentiment analysis within the organization, more research should be done to mine this company's database at the aspect level.
(Costa & Veloso, 2015)	Introducing sentiment analysis approaches that are based on learning vector representations for employee communications in an unsupervised way, and then these representations are given as input to a supervised regression algorithm which finally maps text/sentiment to employee factors	Reviews in social media platforms like Indeed and LoveMondays.	SVR Regression Algorithm SVM Classification Algorithm	Standard Root Mean Squared Error (RMSE) measure for Regression. And standard accuracy and F1 measures for classification tasks	Our experiments showed that both approaches are significantly superior to the standard BoW approach.	-
(Bajpai et al., 2019)	Generating aspect-sentiment based embedding for the companies by looking into reliable employee reviews of them.	Comprehensive dataset of company reviews from Glassdoor.	SVM ELM (Extreme Learning Machine)	Frequency and Macro F1-score	We addressed the overall employee sentiment on different aspect granularities, e.g., salary, location, work life, etc. This study offers businesses a helpful tool to handle employee complaints and boost morale. However, job searchers can also use this study more to identify the top employers in their field of interest.	Subsequent research endeavors will primarily concentrate on incorporating user ratings into the creation of a user-product-sentiment model. A major component of this future study is also a more thorough aspect-level sentiment analysis.
(Loke & Lam-Lion, 2021)	This paper applies sentiment analysis in combination with semantic search as a suitable technique to explore how employees perceive organizations. Organizations may meet the expectations of stakeholders and adjust to changes in the market by utilizing our toolset. It can also be used to educate companies that are ignorant of bad internet reviews.	The dataset was scraped from the Indeed website.	Language detection Python libraries, like SpaCy and LangDetect.	-	An overall observation from this graph is that most organizations are negatively reviewed. A satisfied employee is likely to tell three close friends about a company's positive experience, whereas a dissatisfied employee could feel the need to let more people know about a terrible experience in an organization.	The only observation is from the graph available in the paper.
(Gupta et al., 2017)	In this paper, we aim to review some papers regarding research in sentiment analysis on Twitter, describing the methodologies adopted and models applied, along with describing a	The dataset is obtained through gathering relevant tweets about the area of interest.	Deep learning models are used like Bayesian Logistic Regression, Naïve Bayes, SVM, Artificial Neural	-	This research topic has evolved during the last decade with models reaching the efficiency of almost 85%-90%.	The topic still lacks the dimension of diversity in the data. Many analyzers do not perform well when the number of classes increases. Also, it is still not tested how accurate the model will be for

	generalized Python based approach.		Network (ANN), Case Base Reasoning, Maximum Entropy Classifier, Ensemble Classifier.			topics other than the one in consideration.
(Jung & Suh, 2019)	Latent Dirichlet Allocation (LDA) is used to identify factors related to job satisfaction. After that, we did several analyses using the factors. Using DA and CA, we assessed each work satisfaction factor's sentiment and importance at the industry, business, group, and chronological levels.	Data collection done from Jobplanet website which include 204,659 reviews from 2014 to 2017.	Techniques used are LDA (Latent Dirichlet Allocation) through POS Tagger,	Dominance Analysis, and Correspondence Analysis.	To help their employees be more satisfied with their jobs, companies must first focus on the 30 fine-grained job satisfaction indicators that were obtained from LDA and the accompanying keywords. It should be noted that five previously unrecognized job satisfaction factors—project, software development, inter-firm relationships, marketing, and overseas business—were discovered. There is a valid reason some of them are included.	Even while LDA's reliability test results indicate acceptable levels of agreement, its levels of agreement are still not as high as those of humans. Although LDA is an unsupervised learning algorithm and hence cannot fully understand real languages, it does not require human intervention.
(Luo et al., 2016)	For textual analysis, we extract anonymous employee reviews to determine the relationship between worker satisfaction and business performance. Our methodology, which draws on categories from corporate value studies, not only offers a "bird's eye view," but also identifies the facets of employee happiness that are driving these relationships.	The dataset obtained through web scraping from Glassdoor website which consists of 257,454 employee reviews.	Analysis done through Regression Analysis, and Industry Radar Chart Analysis.	Simple comparison of OLS Regression between models.	The study gave examples of the kinds of analysis that can be performed with a text mining approach. Following the extraction of important categories, the dataset was subjected to regression and exploratory analysis to identify intriguing relationships and trends. Furthermore, the favorable relationship between overall employee happiness and business success was validated by our correlation study. Further investigation into the employee satisfaction category revealed unfavorable relationships between Integrity, Safety, and Communication. This result agrees with several earlier research findings.	Although each factor and/or topic is explained in detail, there is no example of review that represents the factor.
(ÖzdemiR et al., 2022)	In this study, the effectiveness of the suggested sample selection strategy was assessed on a few fundamental classifiers by performing a basic literature analysis on the application of topic modeling techniques and considering employee online assessments to identify and examine the elements that influence job satisfaction.	An evaluation was conducted on 67,529 comments that were gathered from employees of Google, Amazon, Netflix, Facebook, Apple, and Microsoft.	The LDA algorithm is used for Topic Modelling as it is both generative and probabilistic. Additionally, five fundamental classification methods are assessed: Random Forest (RF), Logistic Regression (LR), KNN, SVM, NB, and Random Forest (KNN).	Evaluated using accuracy, precision, recall, and F-measure values.	It was highlighted in this work that machine learning techniques can perform better in classification and can be used efficiently and scalable to vast amounts of data. When it comes to basic classifiers utilized in experimental research, the Random Forest method performs the best, scoring an 85.05. Random Forest is used to generate the Random Space method, which has the best performance among ensemble learning architectures.	The study lacks explanation of the accuracy as it does not show any factors that made the result.

Based on Table 4's similar works and its explanations, there are some projects that have corresponding features and benefits to this final year project's research topic:

1. Integration of Recommendations Feature:
    - Among the studies listed, only a few projects have implemented recommendation features in their sentiment analysis models. For instance, a journal mentioned generating aspect-sentiment based embedding for companies from Glassdoor reviews, which could potentially be utilized for recommendation purposes (Bajpai et al., 2019).
    - Recommendation features can enhance the practical utility of sentiment analysis models by providing actionable insights for businesses. For example, identifying specific aspects or attributes of products or services that drive positive or negative sentiment can inform targeted recommendations for improvement.
    - Therefore, further research could explore the integration of recommendation features into sentiment analysis models across various domains and employee satisfaction, to evaluate their effectiveness in facilitating decision-making and enhancing user experiences.
2. Dataset Size and Scope:
    - Most models in the studies have been tested with datasets containing even less than 10 thousand rows of data, which could be considered small-scale in the context of sentiment analysis, especially considering the vast amount of user-generated content available online. For example, a paper collected approximately 156,428 records from social media platforms primarily from IT firms (Mouli et al., 2023).
    - The use of smaller datasets may limit the generalizability and robustness of sentiment analysis models, as they may not capture the full diversity and variability of sentiments expressed across different contexts and populations. For instance, a study evaluated their method using a total of 67,529 comments collected from employees at various tech companies (ÖzdemiR et al., 2022).
    - Therefore, future research could explore the scalability and performance of sentiment analysis models with larger datasets, potentially leveraging techniques such as distributed computing and parallel processing to handle the computational demands of analyzing big data.

3. Integration of Numerical Ratings and Textual Reviews:
    - Only a few studies have integrated ratings and reviews into their datasets and utilized them in their sentiment analysis models. For example, a journal mentioned using company reviews from Glassdoor, which include both textual reviews and accompanying ratings (Bajpai et al., 2019).
    - Incorporating ratings alongside textual reviews can provide additional context and granularity to sentiment analysis, enabling more nuanced interpretations of sentiment polarity and strength. This approach was also hinted at by a journal in their aspect-sentiment based embedding generation (Bajpai et al., 2019).
    - Therefore, further research could investigate the impact of integrating ratings and reviews on the performance and interpretability of sentiment analysis models, as well as explore techniques for effectively combining textual and numerical data in sentiment analysis tasks.

2.4 Technical Research

2.4.1 IDE (Interactive Development Environment)

Leveraging Jupyter Notebook within Visual Studio Code offers a robust environment for researchers, particularly when collaborative work is not a requirement. Jupyter Notebook's support for Markdown and Code cells allows for interactive code execution and immediate visualization feedback, enhancing the programming experience. Furthermore, running Jupyter locally through Visual Studio Code provides greater control over computational resources, enabling efficient handling of resource-intensive tasks without reliance on cloud-based platforms like Google Colab. This setup ensures that the development process can be tailored to individual workflow preferences, optimizing productivity and flexibility.

2.4.2 Libraries and/or Tools

2.4.2.1 Libraries

Python has emerged as the top choice for machine learning and natural language processing (NLP). This is due to Python's simple syntax, which allows developers to concentrate on logic and algorithms rather than arcane syntax difficulties. Python's ability for sentiment analysis on job reviews is mostly due to its extensive library base for machine learning and natural language processing (NLP).

Machine Learning Libraries:

- TensorFlow and PyTorch: These libraries provide flexible machine learning models with little programming effort, allowing you to create sophisticated models that understand the intricacies of human language. They also make model sharing easier by creating specific repositories for pre-trained models, which encourages collaborative research discovery.

NLP Libraries:

- NLTK: This library provides essential tools for text processing and linguistic analysis, crucial for interpreting and classifying job review texts. It includes pre-trained models trained on extensive text corpora, which can be fine-tuned on domain-specific data to enhance sentiment analysis accuracy.
- Transformers: The Transformers library by Hugging Face offers state-of-the-art pre-trained models for various NLP tasks, including sentiment analysis, making it easier to implement and fine-tune models for job review texts.

Traditional Machine Learning:

- Scikit-learn, which is widely used for implementing classical machine learning algorithms, is effective in feature engineering and classification tasks related to sentiment analysis on job reviews. It has features for partitioning datasets, selecting features, preprocessing, building models, and evaluating performance, making it easier to create trained models.

Data Handling and Visualization:

- Pandas and NumPy are essential tools for preparing and handling huge job review datasets, which are typical in sentiment analysis jobs. The interoperability of data manipulation and machine learning libraries simplifies the workflow from data gathering to model deployment in sentiment analysis systems.
- Seaborn, Plotly, and Matplotlib: Essential for data visualization, these libraries help in understanding data distributions, correlations, and trends within job reviews.
- Word Cloud: This visualization technique aids in highlighting significant words or themes within job reviews, offering researchers intuitive insights into the sentiment expressed.

Model Interpretation and Deployment:

- Lime: Lime is important for interpretable AI, providing intuitive explanations for individual predictions, shedding light on the specific features of a job review that contribute to the sentiment classification.
- Pickle: Used for compiling and exporting models, Pickle allows for the serialization and deserialization of Python objects, making it easier to save and load trained models.

Web Application Development:

- Streamlit: Streamlit is a powerful tool that simplifies the development of interactive web applications for data science projects. Leveraging Streamlit alongside Lime offers an insightful approach to sentiment analysis on job reviews. With Streamlit, developers can effortlessly create user-friendly interfaces that allow users to input job review data and visualize the sentiment analysis results in real-time.

- Bing Web Search: The Web Search API allows you to submit a search query to Bing and receive search results that include connections to webpages, photos, and more. This section describes in technical depth the query parameters and headers used to obtain web search results, as well as the JSON response objects that contain them.

By harnessing these Python libraries, researchers can efficiently explore various machine learning approaches for sentiment analysis on job reviews, ranging from supervised to unsupervised learning paradigms. This versatility is instrumental in deciphering and understanding sentiments conveyed in job review texts, contributing to enhanced decision-making processes in employment contexts.

2.4.2.2 Tools

Microsoft Excel remains a powerful tool for data analysis and visualization, often utilized in sentiment analysis of job reviews due to its accessibility and rich feature set. Despite not being a traditional choice for advanced machine learning and NLP tasks, Excel's extensive functionalities, combined with its ease of use, make it an asset in this domain.

Data Handling and Manipulation:

- Data Import and Export**:** Excel supports various data formats, including CSV, making it easy to import and export job review datasets.
- Data Cleaning**:** Excel offers a suite of tools for data cleaning, such as find and replace, and filters, which are crucial for preparing job review texts for analysis.

2.5 Summary

The literature review offers an extensive examination of sentiment analysis applied to job reviews, covering domain exploration, existing systems, and technical methodologies. The domain exploration section delves into text analysis techniques, emphasizing their importance in sentiment analysis through Natural Language Processing (NLP) and feature extraction. It explores various machine learning models, types of sentiment analysis, and interpretable AI.

Text categorization, sentiment analysis on job reviews and annotation, natural language processing and feature extraction, and machine learning models are among the key subjects discussed. The text analysis section highlights the importance of text classification in NLP, including both rule-based and machine learning-based approaches. The sentiment analysis on

job reviews and annotation section focuses on categorizing and evaluating information using machine learning, natural language processing, and deep learning approaches. Furthermore, the NLP and feature extraction section delves into the process of converting raw text input into numerical features for machine learning methods, including Counter and TF-IDF techniques.

Additionally, the chapter delves into the technical aspects of sentiment analysis on job reviews, highlighting the adoption of Python as the programming language due to its simplicity and the availability of libraries such as SciKit-Learn and NLTK. These libraries facilitate the development of models capable of capturing the intricacies of human language and are crucial for text processing and linguistic analysis. Furthermore, the chapter discusses the utilization of Jupyter Notebook within Visual Studio Code as the development environment, underscoring its advantages for code editing and interactivity.

Finally, the chapter presents a comparative analysis of analogous systems and works, evaluating their efficacy, strengths, and weaknesses in sentiment analysis on job reviews. With this, similar works are compared and find a conclusion where research gaps still exist despite much research that has been done toward the topic, like lack of format variety of job review making it hard to provide a prescriptive improvement.

CHAPTER 3: METHODOLOGY

3.1 Introduction

This chapter introduces the methodology utilized for conducting the data analysis project, which adheres to the CRISP-DM framework. Employing a structured methodology is crucial for ensuring a systematic flow throughout the project. CRISP-DM offers a well-defined procedure for executing data analysis projects.

Subsequently, the project progresses through stages encompassing data collection, comprehension, and preprocessing, in accordance with the established methodological guidelines. The data understanding phase involves the loading and exploration of the collected datasets, followed by a discerning process to select, and retain only the relevant data for further processing. The data preprocessing phase entails a sequence of steps aimed at refining the text data, preparing it for the subsequent feature extraction process.

To conclude the processes undertaken in this phase of the project, the refined dataset emerges as a pivotal output. It represents a meticulously cleaned, organized, and meaningful compilation of data, specifically tailored to propel the subsequent tasks outlined in Chapter 4.

3.2 System Development Methodology

3.2.1 Introduction of CRISP-DM Methodology

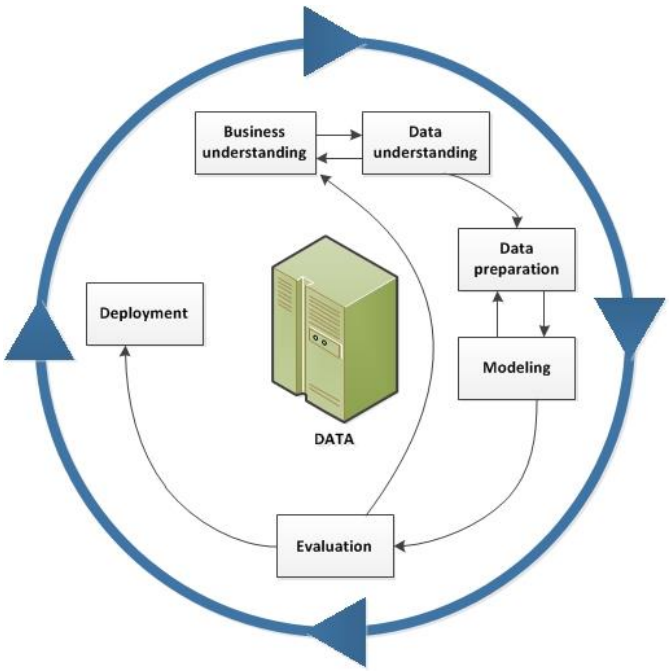


Figure 6 CRISP-DM Life Cycle (https://shorturl.at/kGL58)

The Cross Industry Standard Process for Data Mining (CRISP-DM) is an industry-standard method for arranging data mining projects. It offers a rigorous approach to planning, organizing, and carrying out data mining initiatives across a variety of businesses. This methodology promotes cyclical advances in data analysis through iterative rounds. CRISP-DM comprises six iterative steps: deployment, business understanding, data understanding, data preparation, modeling, and evaluation.

3.2.2 Methodology Choice and Justification

The CRISP-DM methodology was chosen for this project due to its structured nature, offering a clear, step-by-step process. It guides users seamlessly from understanding business problems to deploying the solution, ensuring a systematic approach to data mining. Despite its linear progression, CRISP-DM's cyclical design underscores iterative refinement, allowing for continuous improvement based on feedback and results. This ensures that the project remains aligned with business objectives and can adapt to new insights or changes as well as deployments at the end of the cycle (Schröer et al., 2021).

In contrast, SEMMA lacks a development stage, making it unsuitable for this project. Moreover, its dependency on SAS tools limits its applicability for attaining optimal results. Additionally, Knowledge Discovery in Databases (KDD), focuses more on the technical aspects of knowledge discovery and offers less emphasis on the business context compared to CRISP-DM.

3.2.3 Activities and process in each phase toward CRISP-DM in detail

Process 1: Business Understanding

The initial phase involves understanding the objectives of sentiment analysis on job reviews, highlighting factors that could affect the project's success. It is essential to define the scope of the analysis and understand potential constraints, such as the types of sentiments to be analyzed and the target audience.

Process 2: Data Understanding

This stage focuses on gathering and exploring job review data to ensure data quality as it involves understanding the sources of job reviews, exploring the distribution of sentiments, and identifying any patterns or trends in the dataset.

Process 3: Data Preparation

Tasks in this stage include preparing the data for sentiment analysis as this involves cleaning the data to remove noise or irrelevant information, preprocessing the text to standardize formatting, and labeling the data with sentiment labels (e.g., positive or negative).

Process 4: Modelling

Sentiment analysis models are trained, validated, and tested to help anticipate job review sentiments. The tagged job review data is modeled using supervised learning algorithms (e.g., logistic regression, support vector machines) and deep learning structures (e.g., recurrent neural networks, transformer models). The models are assessed on their ability to reliably anticipate sentiments.

Process 5: Evaluation

During this step, the performance of the sentiment analysis models is assessed using predefined criteria such as accuracy, precision, recall, and F1-score. Models that fulfill the specified performance standards are chosen for further evaluation. In addition, a thorough review of the sentiment analysis method is performed to identify any flaws or opportunities for improvement.

Process 6: Deployment

The deployment phase involves integrating the sentiment analysis models into applications or platforms where job reviews are collected or analyzed. This may include generating sentiment analysis reports for stakeholders or incorporating sentiment analysis features into job search platforms. Careful planning and documentation are essential to ensure the successful deployment of sentiment analysis solutions for job reviews.

3.3 Summary

Chapter 3 of this investigation report explains the methodology employed for the data analysis project, centering on the CRISP-DM framework. This framework is selected for its structured, systematic approach, which proves beneficial in guiding the project from understanding the business problem to implementing the solution. The methodology is iterative, allowing for continual refinement and adjustment based on new insights. The six CRISP-DM phases—Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, and Deployment—each with goals and duties are explained in detailed in this chapter.

CHAPTER 4: DESIGN AND IMPLEMENTATION

4.1 Introduction

Understanding and preprocessing the dataset including job evaluations from Glassdoor supplied on Kaggle is covered in the fourth chapter of the Final Year Project. To prepare for a machine learning model that is expected to perform in accordance with the deliverables, the dataset will undergo Exploratory Data Analysis to obtain a deeper grasp and knowledge of the features and characteristics. It also includes cleaning the data, eliminating noise, and ensuring that the dataset is in ideal shape for building effective models. Not only that, but the student will create a data labelling model using BERT to analyze the sentiment of each review. This is an important stage since the dataset from the next step will be used to train and test the final model. The model's values will be used to classify data using other typical machine learning models, as well as BERT for a basic transformer-based model.

4.2 Data Collection

The “Glassdoor Job Reviews” dataset contains valuable information about job experiences and company ratings across various industries in the UK. This dataset is particularly useful for conducting multidimensional sentiment analysis to gain insights into different aspects of job satisfaction and workplace culture. The dataset is sourced from Kaggle and includes job titles and rankings based on various criteria such as work-life balance, senior management, diversity inclusion, etc. The dataset itself is scraped from Glassdoor website and uploaded to Kaggle in a single csv-formatted file.

Table 5 Dataset's Variables

No.	*Variable/Column*	*Explanation*
1.	firm	The name of the firm.
2.	date_review	The date of the review.
3.	job_title	The title of the job.
4.	current	The current status of the employee.
5.	location	The location of the job.
6.	overall_rating	The overall rating given by the employee for the job.
7.	work_life_balance	The rating given for work life balance.
8.	culture_values	The rating given for culture values.
9.	diversity_inclusion	The rating given for diversity inclusion.
10.	career_opp	The rating given for career opportunities in the job.
11.	comp_benefits	The rating given for company benefits.
12.	senior_mgmt	The rating given for senior management that exists.
13.	recommend	The rating given for recommendation of the job.
14.	ceo_approv	The approval of CEO for the review.
15.	outlook	The firm’s outlook.
16.	headline	The job review’s headline.
17.	pros	The pros of the job.
18.	cons	the cons of the job.

The above Table 5 shows the dataset contains several key columns that provide detailed information about each review:

- Date of the Review: The date when the review was posted.
- Job Name: The title or name of the job being reviewed.
- Job Location: The geographical location of the job.
- Status of the Reviewers: Information about the reviewer's status, such as current or former employee.
- Review Content: The actual review text, including the headline and both positive (pros) and negative (cons) aspects.
- Sub-Categories: Reviews are divided into sub-categories for more granular analysis:
    - Work/Life Balance
    - Compensation & Benefits
    - Career Opportunities
    - Senior Management
    - Culture & Values
- Ratings: Numerical ratings provided by the reviewers, typically on a scale (e.g., 1 to 5).
- Recommendations: Employees can add recommendations about the firm, CEO approval, and the company's outlook.
- Recommendation Categories:
    - v: Positive
    - r: Mild/Neutral
    - x: Negative
    - o: No opinion

Here is one review example from the dataset:

MCDONALD'S:

- Review: "I don't like working here, don't work here."
- Pros: "Some people are nice, some free food, some of the managers are nice about 95% of the time."
- Cons: "95% of people are mean to employees/customers, it's not a clean place, people barely clean their hands of what I see, managers are mean, I got a stress rash because of this I can't get rid of it, they don't give me a little raise even though I do a lot of crap there for them."
- Rating: 1.0

4.3 Data Understanding

4.3.1 Basic Data Understanding

The beginning of doing Exploratory Data Analysis (EDA) starts with importing python libraries and loading the raw dataset.

```
import pandas as pd

df_raw = pd.read_csv('C:/Users/setia/Downloads/glassdoor_reviews_copy.csv')
```

Figure 7 Source Code of Loading the Raw Dataset

```
df_raw.info()
✓ 0.2s

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 838566 entries, 0 to 838565
Data columns (total 18 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   firm                 838566 non-null  object
 1   date_review          838566 non-null  object
 2   job_title            838566 non-null  object
 3   current              838566 non-null  object
 4   location             541223 non-null  object
 5   overall_rating       838566 non-null  int64
 6   work_life_balance    688672 non-null  float64
 7   culture_values       647193 non-null  float64
 8   diversity_inclusion  136066 non-null  float64
 9   career_opp           691065 non-null  float64
 10  comp_benefits        688484 non-null  float64
 11  senior_mgmt          682690 non-null  float64
 12  recommend            838566 non-null  object
 13  ceo_approv           838566 non-null  object
 14  outlook              838566 non-null  object
 15  headline             835976 non-null  object
 16  pros                 838564 non-null  object
 17  cons                 838553 non-null  object
dtypes: float64(6), int64(1), object(11)
memory usage: 115.2+ MB
```

Figure 8 Info of the Raw Dataset

Figure 7 shows the loading of the Glassdoor job review dataset which was obtained through the Kaggle website. The process of loading was through using pandas python library to read the dataset in csv format. The dataset was backup-ed, then the copy will be loaded. After it was loaded, Figure 8 shows the info of the raw dataset, and it can be seen there are 18 columns in total with 838,566 entries. The data types contained in the dataset are float which are numbers with commas included, integer without commas included, and object which is mostly text data.

To understand better each of the columns or variables and how it impacts the dataset as well as reflect the real-life situation, the value count of each variable will be displayed for discussion and shown from Figure 9 to 27.

```
df_raw.firm.value_counts()
✓ 0.0s

firm
IBM                          60436
McDonald-s                   49450
Deloitte                     46995
EY                           34050
PwC                          33227
                             ...
Health-Protection-Agency         4
i-Net-Solution                   4
The-Survey-Association           3
NPSA                             2
UKCIL                            1
Name: count, Length: 428, dtype: int64
```

Figure 9 Count of the Firm in the Raw Dataset

Now, let's check if the raw dataset has any null values. There's no need to drop any null values now because we just want to understand our dataset without any data pre-processing.

```
df_raw.isna().sum()
✓ 0.2s

firm                      0
date_review               0
job_title                 0
current                   0
location             297343
overall_rating            0
work_life_balance    149894
culture_values       191373
diversity_inclusion  702500
career_opp           147501
comp_benefits        150082
senior_mgmt          155876
recommend                 0
ceo_approv                0
outlook                   0
headline               2590
pros                      2
cons                     13
dtype: int64
```

Figure 10 Null Values of the Raw Dataset

```
import matplotlib.pyplot as plt
df_raw['firm'].value_counts()[:10].plot(kind = 'pie', autopct = '%1.1f%%',
                                        shadow = True, explode = [0.1,0,0,0,0,0,0,0,0,0])
plt.title("Top reviewed firms at glassdoor")
fig = plt.gcf()
fig.set_size_inches(7,7)
plt.show()
```

✓ 0.2s

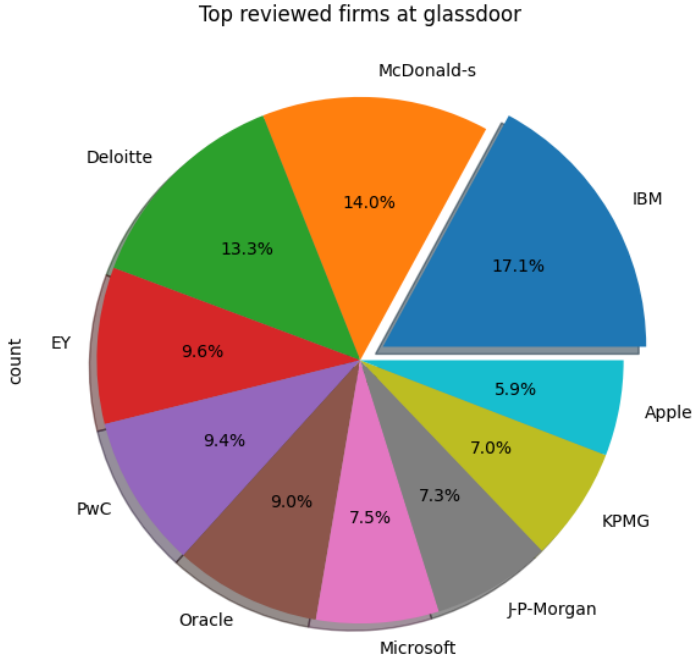


Figure 11 Pie Chart of Top Reviewed Firms

Figures 9 and 11 show the same information but through different visualizations. It is shown that IBM has the highest review in the dataset and followed by McDonald's and Deloitte. Looking at the pie chart, it seems the dataset has many reviews from insurance consulting and technologies companies with McDonald's as an exception. This is a helpful insight to know if there is any bias within the dataset which might affect later recommendations model.

```
df_raw.date_review.value_counts()
```

✓ 0.0s

```
date_review
2021-01-13    1761
2021-01-11    1713
2021-01-12    1629
2021-01-19    1613
2021-01-14    1581
              ...
2008-05-25       1
2008-06-06       1
2008-03-18       1
2008-04-08       1
2021-06-08       1
Name: count, Length: 4813, dtype: int64
```

Figure 12 Value Count of Date Review

```
df_raw.job_title.value_counts()
✓ 0.0s

job_title
Anonymous Employee              162649
                                 79065
Manager                          14906
Consultant                       12559
Software Engineer                10797
                                 ...
Consumer Service Specialist          1
Senior Talent Consultant             1
Manager, Talent Acquisition          1
LBR                                  1
Energy Administrator                 1
Name: count, Length: 62275, dtype: int64
```

Figure 13 Value Count of the Job Title

Figure 12 shows the dataset mostly contains data from the year 2021 with 2008 data as the lowest. As for Figure 13, it shows that almost 20% of the job reviews in the dataset are mostly submitted by anonymous employees which could create difficulty in the recommendations stage as the researcher would not know which industry the review is categorized to. Another problem would be the job title is not as precise and concise because some employees would just put their job place but not their role.

```
df_raw.location.value_counts()
✓ 0.0s

location
London, England, England                                  58665
New York, NY                                              31172
Bangalore                                                 28102
Hyderābād                                                 11458
Mumbai                                                     9543
                                                          ...
Ban Nagnang, Vientiane Prefecture, Vientiane Prefecture       1
Patos de Minas                                                1
Itajaí, Santa Catarina, Santa Catarina                        1
Araucária                                                     1
Olonne-sur-Mer                                                1
Name: count, Length: 14486, dtype: int64
```

Figure 14 Value Count of the Location

```
df_raw.current.value_counts()
✓ 0.0s

current
Current Employee                          209599
Former Employee                           146133
Current Employee, more than 1 year         82749
Current Employee, more than 3 years        66471
Former Employee, more than 1 year          65687
Current Employee, less than 1 year         49603
Former Employee, more than 3 years         43614
Former Employee, less than 1 year          41874
Current Employee, more than 5 years        40155
Current Employee, more than 10 years       25029
Former Employee, more than 5 years         23017
Current Employee, more than 8 years        18506
Former Employee, more than 10 years        15411
Former Employee, more than 8 years         10686
Former Contractor, less than 1 year            6
Former Intern, less than 1 year                5
Current Contractor, less than 1 year           3
Current Contractor, more than 1 year           3
Former Contractor                              2
Former Intern                                  2
Current Contractor                             2
KEY NOT FOUND: jobLine.per_diem-former         2
Former Intern, more than 1 year                1
Former Contractor, more than 1 year            1
...
Former Contractor, more than 8 years           1
Former Temporary Employee                      1
Current Freelancer, more than 3 years          1
KEY NOT FOUND: jobLine.temporary-former        1
Name: count, dtype: int64
```

Figure 15 Value Count of Current Job

Figure 14 shows that the dataset is not in fact provides job reviews particularly located in the United Kingdom, but all over the world. However, the U.K. location still holds the most job reviews. Figure 15 shows employees' job status, whether they have quit or are still working the job, we also can see that there are key errors in the "current" column.

```
df_raw.overall_rating.value_counts()
✓ 0.0s

overall_rating
4    278277
5    232255
3    194267
2     74809
1     58958
Name: count, dtype: int64
```

Figure 16 Value Count of Overall Rating

```
df_raw.work_life_balance.value_counts()
✓ 0.0s

work_life_balance
4.0    176147
5.0    169187
3.0    169056
2.0     92801
1.0     81481
Name: count, dtype: int64
```

Figure 17 Value Count of Work Life Balance

Figure 16 shows the overall rating distribution of the job reviews with 4 stars at the top followed by 5 and 3 stars. The same goes for Work Life Balance rating's distribution at Figure 17.

If we look at Figures 18 to 20, they have the same pattern of ranking for the rating stars as culture values, diversity inclusion, and career opportunities are all important to the overall rating as those factors are felt the most by the employees. However, ratings 3 to 5 have higher combined total to 1 to 3, which means the dataset is imbalanced. This should be noted for data pre-processing.

On the other hand, Figure 21 shows the company benefits rating, and it does not follow other patterns as it starts with 4, followed by 3 then 5 stars. This is strange as other columns show 5 as the top then followed by 4, however it's probably cause by the poor benefits provided by the company despite good environments.

```
df_raw.culture_values.value_counts()
✓ 0.0s
```

```
culture_values
5.0    210028
4.0    169424
3.0    131024
1.0     70813
2.0     65904
Name: count, dtype: int64
```

Figure 18 Value Count of the Culture Values

```
df_raw.diversity_inclusion.value_counts()
✓ 0.0s
```

```
diversity_inclusion
5.0    60422
4.0    36101
3.0    22574
2.0     8516
1.0     8453
Name: count, dtype: int64
```

Figure 19 Value Count of Diversity Inclusion

```
df_raw.career_opp.value_counts()
✓ 0.0s

career_opp
4.0    189882
5.0    177286
3.0    168571
2.0     85349
1.0     69977
Name: count, dtype: int64
```

Figure 20 Value Count of Career Opportunities

```
df_raw.comp_benefits.value_counts()
✓ 0.0s

comp_benefits
4.0    194437
3.0    189896
5.0    148714
2.0     93101
1.0     62336
Name: count, dtype: int64
```

Figure 21 Value Counts of Company Benefits

```
df_raw.senior_mgmt.value_counts()
✓ 0.0s

senior_mgmt
4.0    172629
3.0    171941
5.0    131464
1.0    108354
2.0     98302
Name: count, dtype: int64
```

Figure 22 Value Count of Senior Management

Figures 23 to 25 below show different kinds of records with “o,” “v,” “r” and “x” as their values. “o” means No Opinion, “v” means Positive, “x” means Negative, and “r” mild as in close to neutral. **Recommendations, CEO Approval**, and **Company Outlook** are all based on the experiences of the reviewers which can be represented in essence from the ratings, therefore, these columns should be important as they can be used to narrow down the dataset later by removing all rows that have no opinion or neutral on the above three columns as the student would like to provide recommendations in a concise and accurate manner.

```
df_raw.recommend.value_counts()
```
✓ 0.0s

```
recommend
v    427865
o    234248
x    176453
Name: count, dtype: int64
```

Figure 23 Value Count of Recommendations

```
df_raw.ceo_approv.value_counts()
```
✓ 0.0s

```
ceo_approv
o    311433
v    286695
r    176930
x     63508
Name: count, dtype: int64
```

Figure 24 Value Count of CEO Approval

```
df_raw.outlook.value_counts()
```
✓ 0.0s

```
outlook
v    301413
o    294548
r    154948
x     87657
Name: count, dtype: int64
```

Figure 25 Value Count of Outlook

```
df_raw.headline.value_counts()
```
✓ 0.3s

```
headline
Good                                    14300
Great place to work                     11298
Great                                    6284
Great company                            6163
Good place to work                       6043
                                        ...
Friendly staff members at LEGO              1
Just an awesome company to work for!!!      1
working at lego                             1
not interested in growing their people      1
Work harder and harder for less money       1
Name: count, Length: 390454, dtype: int64
```

Figure 26 Value Count of Headline

```
df_raw.pros.value_counts()
```
✓ 0.8s

```
pros
Great company to work for
Good company to work for
Great people to work with
Work life balance is good
Good people to work with

Friendly colleagues, caring decision-makers, and employee discount. In general it's really fun workplace.
Colleagues and management were really friendly. Nice discounts and some free goodies.
Have a close knit community.
My time working at the LEGO Brand Retail of my local mall was possibly the best retail job I could ever ex
I jumped from a bankassurance world into the world of IFA and was very careful about selecting AFH as I in
Name: count, Length: 778559, dtype: int64
```

Figure 27 Value Count of Pros Text

```
df_raw.cons.value_counts()
```
✓ 0.2s

```
cons
None that I can think of                                                                                  1156
None that I can think of.                                                                                  443
None I can think of                                                                                        340
Nothing I can think of                                                                                     243
Long hours during busy season                                                                              221
                                                                                                          ...
Sometimes it takes a while to improve system/reflect feedbacks as the decision-making process is quite long    1
Low pay and lots of standing, but that's the usual issues for retail.                                          1
Not enough hours for part time workers.                                                                        1
While it wasn't relevant to myself, there didn't appear to be a clear path towards promotion at the time I worked there.    1
Many things are centralized in Denmark and relocation is often for certain jobs is usually must.              1
Name: count, Length: 777133, dtype: int64
```

Figure 28 Value Count of Cons Text

Figures 26 to 28 shows an example of headline, pros, and cons of the job review submitted by the employees.

4.3.2 Advanced Data Understanding

To further progress our data understanding, Figures 29 and 30 show the process of checking the NULL value count in each variable of the dataset. It shows that Diversity Inclusion has the most NULL values followed by location and culture values. This is probably because of the reason mentioned previously as people are not as willing to engage with the company's environment and social circles.

```
df_raw.isna().sum()
✓ 0.2s
```

```
firm                      0
date_review               0
job_title                 0
current                   0
location             297343
overall_rating            0
work_life_balance    149894
culture_values       191373
diversity_inclusion  702500
career_opp           147501
comp_benefits        150082
senior_mgmt          155876
recommend                 0
ceo_approv                0
outlook                   0
headline              2590
pros                     2
cons                    13
dtype: int64
```

Figure 29 Number of NULL Values

```
missing_percent = df_raw.isna().sum().sort_values(ascending=False) / len(df_raw)
missing_percent
missing_percent[missing_percent != 0].plot(kind = 'barh')
✓ 0.9s
```

```
<Axes: >
```

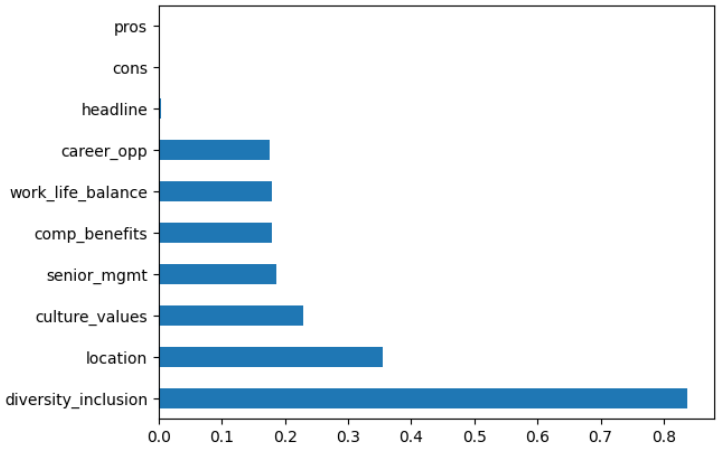


Figure 30 Missing Values in Percentage

Figure 31 and so on will show a more in-depth EDA by understanding what kind of companies exist in the record. Figure 3 shows bank and hotel companies exist in the dataset which expands the industries category.

```
no_jobs = df_raw['firm'].unique()
len(no_jobs)
✓ 0.0s
```

```
428
```

Figure 31 Total Firms in the Dataset

```
jobs_by_reviews = df_raw['firm'].value_counts()
j1 = jobs_by_reviews[:20]
j1
```
✓ 0.0s

```
firm
IBM                    60436
McDonald-s             49450
Deloitte               46995
EY                     34050
PwC                    33227
Oracle                 31941
Microsoft              26675
J-P-Morgan             25814
KPMG                   24815
Apple                  20797
Citi                   18726
Google                 15995
SAP                    14344
HSBC-Holdings          13893
Tesco                  12149
Marriott-International 10409
Barclays                9710
Thomson-Reuters         9553
American-Express        9349
Morgan-Stanley          9093
Name: count, dtype: int64
```

Figure 32 Total Reviews by Firm

```
df1 = df_raw[df_raw['overall_rating'] == 5]
jobs_by_fivestar = df1['firm'].value_counts()
j2 = jobs_by_fivestar[:20]
j2
```
✓ 0.0s

```
firm
IBM                    12468
Deloitte               12184
McDonald-s              9909
Microsoft               9702
Google                  9531
Apple                   8734
PwC                     8157
EY                      8058
SAP                     7461
J-P-Morgan              6847
KPMG                    5264
Oracle                  5233
Salesforce              4676
Marriott-International  3830
Citi                    3587
American-Express        3113
Tesco                   2950
HSBC-Holdings           2714
Goldman-Sachs           2710
McKinsey-and-Company    2679
Name: count, dtype: int64
```

Figure 33 Total Reviews by 5 Star

```
#top companies with the highest percentage of five star rating
(j2 / j1).sort_values(ascending = False)[:20]
```
✓ 0.0s

```
firm
Google                 0.595874
SAP                    0.520148
Apple                  0.419964
Marriott-International 0.367951
Microsoft              0.363711
American-Express       0.332977
J-P-Morgan             0.265244
Deloitte               0.259262
PwC                    0.245493
Tesco                  0.242818
EY                     0.236652
KPMG                   0.212130
IBM                    0.206301
McDonald-s             0.200384
HSBC-Holdings          0.195350
Citi                   0.191552
Oracle                 0.163833
Barclays                    NaN
Goldman-Sachs               NaN
McKinsey-and-Company        NaN
Name: count, dtype: float64
```

Figure 34 Jobs by the Five Star Rating Percentage

Figures 33 and 34 show that despite IBM's highest overall rating, its five-star rating percentage compared to other ratings is still low with 0.206 or 20%. This proves that the number of 5 stars rating does not correlate to its percentage and hence, this problem must be tackled from other perspectives.

```
VODAFONE
Target oriented
Headline:  Target oriented
Pros: Interact with people from diverse background
Cons: Very sales oriented doesn't care about employees
Rating: 1
```

Figure 35 Rating 1 Review Example

```
J-P-MORGAN
The Truth
Headline:  The Truth
Pros: Great Benefits, and vacation time
Cons: You are just a number
Rating: 2

APPLE
Sales Representative
Headline:  Sales Representative
Pros: Great atmosphere, great work for students.
Cons: Strict inside policies, lack of career opportunities.
Rating: 3

TESCO
it was ok
Headline:  it was ok
Pros: got my premises licence
learnt a lot
made a lot of life long friends
Cons: Managers pretty left the shift managers to sort everthing out
Rating: 4

AMERICAN-EXPRESS
By far the the Best Employer
Headline:  By far the the Best Employer
Pros: Excellent & challenging work environment
Cons: Can be too consensus driven at time
Rating: 5
```

Figure 36 Rating 2 to 5 Review Examples

Figures 35 and 36 show the example of reviews from rating 1 to 5 to give an idea of the reviews in the dataset.

```
import seaborn as sns
sns.set_style('darkgrid')

sns.regplot(data = df_raw, x = 'career_opp', y = 'overall_rating')
```

✓ 1m 4.0s

```
<Axes: xlabel='career_opp', ylabel='overall_rating'>
```

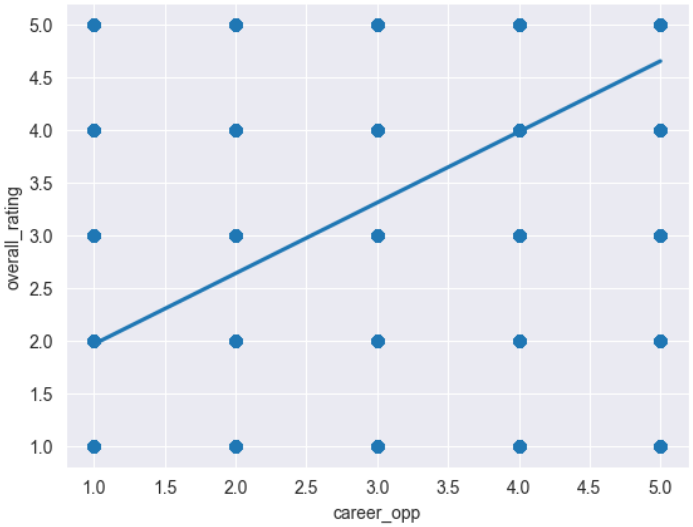


Figure 37 Regression Plot of Overall Rating and Career Opportunities

```
sns.regplot(data = df_raw, x = 'work_life_balance', y = 'overall_rating')
✓ 1m 5.0s
<Axes: xlabel='work_life_balance', ylabel='overall_rating'>
```

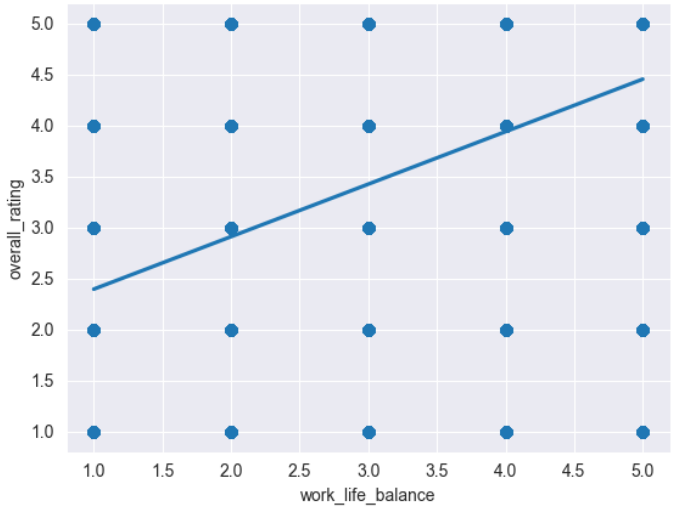


Figure 38 Regression Plot of Overall Rating and Work Life Balance

Figures 37 and 38 show the regressions plots between overall rating and career opportunities as well as work life balance. It shows a correlation with higher overall rating, the higher career opportunities and work life balance ratings are. The same can be seen at Figures 39 and 40 with company benefits and senior management. However, the aggressive hike of overall rating can be seen effected by career opportunities, company benefits, and senior management which shows that company has many things that can effect on to the employees.

```
sns.regplot(data = df_raw, x = 'comp_benefits', y = 'overall_rating')
✓ 1m 2.0s
<Axes: xlabel='comp_benefits', ylabel='overall_rating'>
```

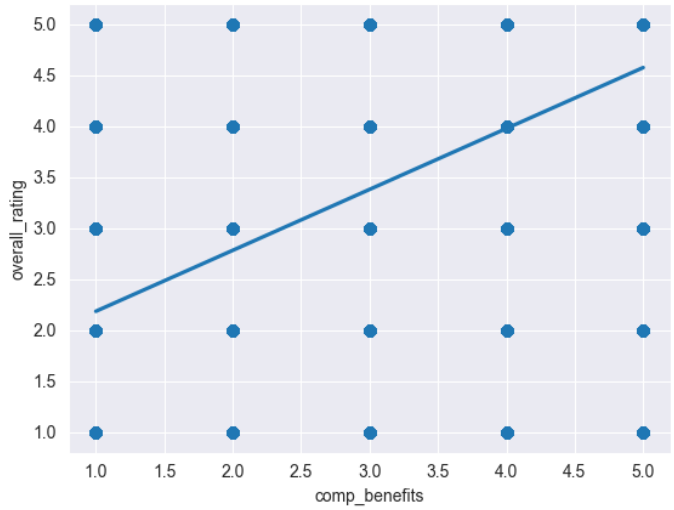


Figure 39 Regression Plot of Overall Rating and Company Benefits

```
sns.regplot(data = df_raw, x = 'senior_mgmt', y = 'overall_rating')
✓ 1m 7.8s
<Axes: xlabel='senior_mgmt', ylabel='overall_rating'>
```

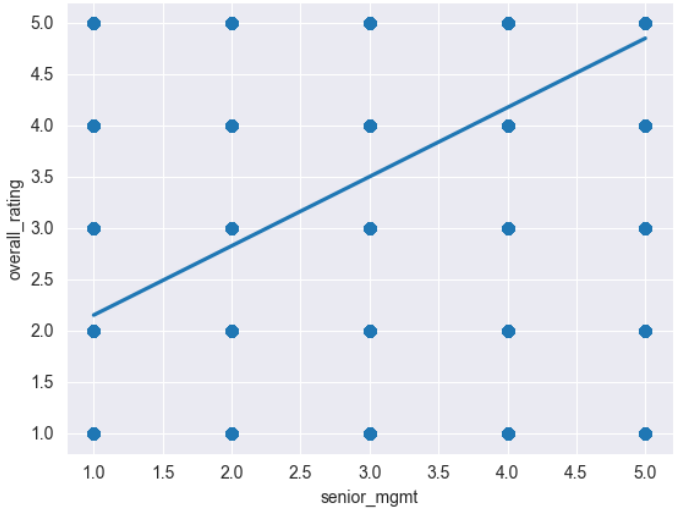


Figure 40 Regression Plot of Overall Rating and Senior Management

It is evident that the career options offered by the organization have the most correlation with the total company grade. The individual gives a higher rating the better the opportunities. Career opportunities have a higher priority for businesses than the other factors.

```
# Calculate and display the length of reviews
review_lengths = df_raw['headline'].str.len()

print(f"Maximum Review Length: {review_lengths.max()}")
print(f"Minimum Review Length: {review_lengths.min()}")
print(f"Average Review Length: {review_lengths.mean():.2f}")
✓ 0.1s
Maximum Review Length: 519.0
Minimum Review Length: 1.0
Average Review Length: 23.34
```

Figure 41 The Maximum, Minimum and Average Length of Reviews

Furthermore, we need to check the maximum and minimum length of the review, specifically headline in the dataset. This might be important for padding purposes in model building.

4.4 Data Pre-Processing

4.4.1 Missing Values

After understanding the characteristics of the dataset, we need to preprocess it. First of all, let's check whether we have any NULL values in any columns.

```
df_raw.isna().sum()
✓ 0.3s
```

```
firm                      0
date_review               0
job_title                 0
current                   0
location                297343
overall_rating            0
work_life_balance       149894
culture_values          191373
diversity_inclusion     702500
career_opp              147501
comp_benefits           150082
senior_mgmt             155876
recommend                 0
ceo_approv                0
outlook                   0
headline                2590
pros                       2
cons                      13
dtype: int64
```

Figure 42 SUM of NULL Values in Each Column Before

```
cleaned_df = df_raw.dropna()
cleaned_df.isna().sum()
✓ 0.2s
```

```
firm                    0
date_review             0
job_title               0
current                 0
location                0
overall_rating          0
work_life_balance       0
culture_values          0
diversity_inclusion     0
career_opp              0
comp_benefits           0
senior_mgmt             0
recommend               0
ceo_approv              0
outlook                 0
headline                0
pros                    0
cons                    0
dtype: int64
```

Figure 43 SUM of NULL Values in Each Column After

```
cleaned_df.info()
✓ 0.0s
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 108544 entries, 50 to 838565
Data columns (total 18 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   firm                 108544 non-null  object
 1   date_review          108544 non-null  object
 2   job_title            108544 non-null  object
 3   current              108544 non-null  object
 4   location             108544 non-null  object
 5   overall_rating       108544 non-null  int64
 6   work_life_balance    108544 non-null  float64
 7   culture_values       108544 non-null  float64
 8   diversity_inclusion  108544 non-null  float64
 9   career_opp           108544 non-null  float64
 10  comp_benefits        108544 non-null  float64
 11  senior_mgmt          108544 non-null  float64
 12  recommend            108544 non-null  object
 13  ceo_approv           108544 non-null  object
 14  outlook              108544 non-null  object
 15  headline             108544 non-null  object
 16  pros                 108544 non-null  object
 17  cons                 108544 non-null  object
dtypes: float64(6), int64(1), object(11)
memory usage: 15.7+ MB
```

Figure 44 Info of Cleaned DataFrame

From Figure 44, we can see that the dataset is balanced in the sense that all the columns have equal amount of data, which is 108544.

4.4.2 Replace Values

Before the DataFrame is exported, let's remove 'r' and 'o' in the recommend, ceo_approv and outlook columns as they mean mild and no opinion respectively, and leave v and x as they mean positive and negative.

```
# Filter out rows where any of the specified columns contain 'o' or 'r'
cols_to_check = ['recommend', 'ceo_approv', 'outlook']
mask = cleaned_df[cols_to_check].isin(['o', 'r']).any(axis=1)
cleaned_df_filtered = cleaned_df[~mask]

print(cleaned_df_filtered.recommend.value_counts())
print(cleaned_df_filtered.ceo_approv.value_counts())
print(cleaned_df_filtered.outlook.value_counts())
```

✓ 0.0s

```
recommend
v    39001
x     6946
Name: count, dtype: int64
ceo_approv
v    40611
x     5336
Name: count, dtype: int64
outlook
v    39937
x     6010
Name: count, dtype: int64
```

Figure 45 Filter Out Any 'o' and 'r' in the Three Columns

The dataset has many types of data types, so the student needs to define several categories:

1. **Primary Variables**: Headlines, Pros, and Cons. These variables contain texts for sentiment analysis purposes.
2. **Secondary Variables**: Work life Balance, Culture Values, Diversity Inclusion, Career Opportunities, Company Benefits, Senior Management
3. **Tertiary Variables**: Firm, Date Review, Job Title, Current (Status of Employment), and Location
4. **Quaternary Variables**: Overall Rating, Recommendation, CEO Approval, and Outlook.

Now, to better enhance our dataset quality, based on the category of the variables, all the data in the quaternary category will be changed to 0 (negative) and 1 (positive) as shown above in Figure 46 below.

```
# Mapping dictionary
mapping = {'v': 1, 'x': 0}

# Apply mapping to specific columns using replace
cleaned_df_filtered.loc[:, ['ceo_approv', 'recommend',
                            'outlook']] = cleaned_df_filtered[['ceo_approv',
                                                               'recommend', 'outlook']].replace(mapping)

print(cleaned_df_filtered.recommend.value_counts())
print(cleaned_df_filtered.ceo_approv.value_counts())
print(cleaned_df_filtered.outlook.value_counts())
```

✓ 0.0s

```
recommend
1    39001
0     6946
Name: count, dtype: int64
ceo_approv
1    40611
0     5336
Name: count, dtype: int64
outlook
1    39937
0     6010
Name: count, dtype: int64
```

Figure 46 Replace 'v' and 'x' With '1' and '0'

Next, the "cleaned_df_filtered" will be exported to a csv format for further pre-processing.

```
# Save to CSV
cleaned_df_filtered.to_csv(r'C:\Users\setia\Downloads\glassdoor_reviews_cleaned_copy_v2.csv', index=False, header=True)
```

Figure 47 Export the cleaned_df_filtered to a csv format

4.4.3 Noise and NaN Values

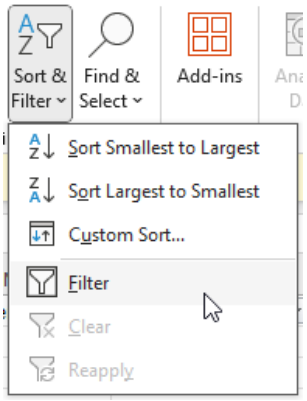


Figure 48 Excel's Filter Function

The next step of pre-preprocessing will be done using Excel as it consists of removing specific data which will be easier using the Microsoft spreadsheet application. After changing the three column's data, next would be removing all rows that have indiscernible and unimportant values because it can cause model performance and accuracy issues.

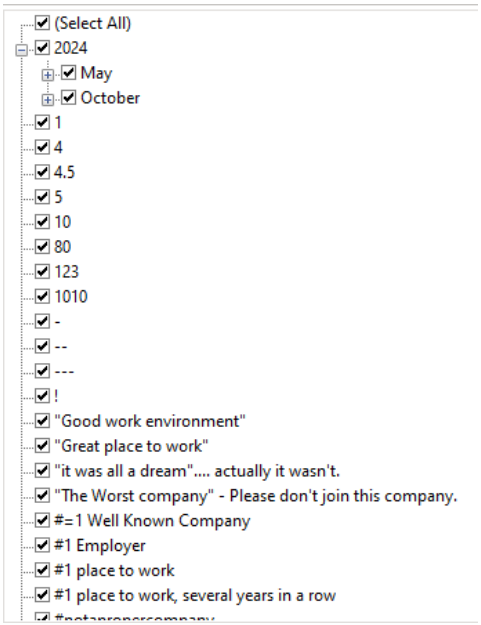


Figure 49 Headline Filter Function Example

The criteria for choosing and deleting useless values would be if it's too short, for example the number one (1) would have no value as it does not have any context, so number only reviews would be the main criteria of deletion. However, a one-word adjective or noun would be fine as it can define a company or firm, but it must be descriptive instead of vague or unclear how it can be represented.

The student starts from headline because in just a sentence, it can represent the whole review of a user from pros and cons, to company benefits and other ratings. Which ensures that headline data is cleaned. At the same time, remove any rows that have no value.

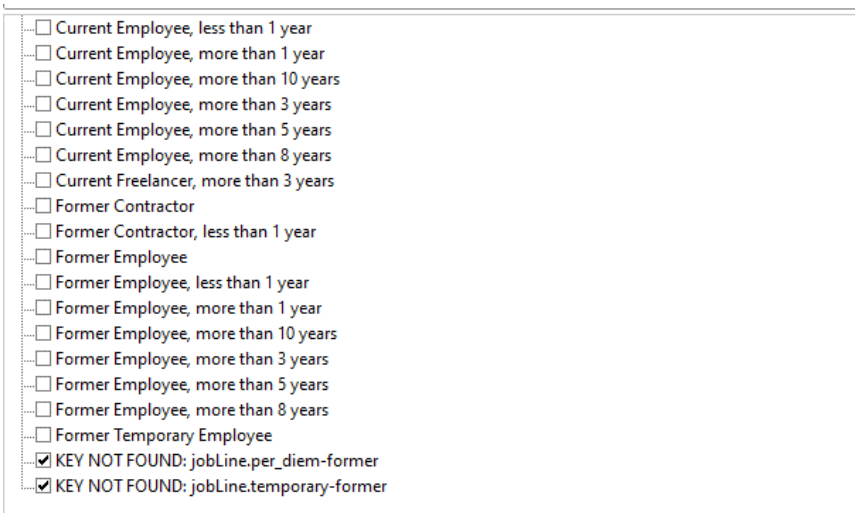


Figure 50 Current

Figure 51 Valueless Location

The Figures below are the examples of the rows that were deleted.

headline	pros	cons
Excellent Company	Good Working Environment. Excellent Management	....
	Market Leader	
	Agile	
	Customer Experience Centric	
	Employee Centric	
Great Place to Worl	Inclusivity and Diversity Leader	.....
Lots to learn	Learning ground, train to be strong	.....
Excellent company	Lots of development opportunities, strong CSR agenc	.....
Great company, Gr	Great company, Great team, Great leadership	.....
Positive	Positive and inclusive approach to work	.....

Figure 52 Troll Inputs

headline	pros	cons
?	Fun working environment, learn a lot	Long hours, high stress levels
*	The team of guys are great	Upper management talks a lot about safety but whe
?	CBRE invests in training for the employees.	Lots of documentation and administration requirec
.....	Open, honest, clear social purpose	Bad previous reputation, not 100% brilliant but tryi
*	itâ€™s close to my home	not fair at all. you need to be a friend of someone (v
*	good start in your career	Canâ€™t balance life and work well
*	good working environment to work in	long hours especially during the lockdown
?	Flexible, supportive, learning, growth, rewarding	The pay could be better.
?	Good pension, nice staff, great to build skills	Can get sacked for having time off cause of mental
?	Itâ€™s a really laid back environment	The pay is low and pay structure is unfair
*	One grows both horizontally and vertically	Nothing as such, great place to work.
?	is gud very nice I guess	Very hard work work condition bad
?	not much to tell on pros	no ot pay, no bonus, no 13 months pay
.....	Medical, MPF and Annual leave	Nothing to learn. Can be better
?	Global so opportunities internally a possibility	Global so decisions made elsewhere
*	None to specify right now	Poor culture and bad management

Figure 53 Undecided Headlines

4.4.4 Data Transformation

Now that the cleaning step with Excel is done, let's move to *Python scripts* for *quicker* preprocessing. First, let's load the cleaned dataset and re-check the contents of it.

```
import pandas as pd

df = pd.read_csv('C:/Users/setia/Downloads/glassdoor_reviews_cleaned_copy_v3.csv')
df.head()
```

Figure 54 Loading the Cleaned Dataset

```
df.info()
✓ 0.0s

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45241 entries, 0 to 45240
Data columns (total 18 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   firm                 45241 non-null  object
 1   date_review          45241 non-null  object
 2   job_title            45241 non-null  object
 3   current              45241 non-null  object
 4   location             45241 non-null  object
 5   overall_rating       45241 non-null  int64
 6   work_life_balance    45241 non-null  int64
 7   culture_values       45241 non-null  int64
 8   diversity_inclusion  45241 non-null  int64
 9   career_opp           45241 non-null  int64
 10  comp_benefits        45241 non-null  int64
 11  senior_mgmt          45241 non-null  int64
 12  recommend            45241 non-null  int64
 13  ceo_approv           45241 non-null  int64
 14  outlook              45241 non-null  int64
 15  headline             45241 non-null  object
 16  pros                 45241 non-null  object
 17  cons                 45241 non-null  object
dtypes: int64(10), object(8)
memory usage: 6.2+ MB
```

Figure 55 Info of df DataFrame

The process of data pre-processing starts with lowercasing the headline column.

```
# Convert 'df' column to lowercase
df['headline'] = df['headline'].str.lower()
df.head()
```

Figure 56 Lowercasing the Headline Column

Then, proceed with removing the extra white spaces for the 'headline' column as shown in Figure 57.

```
# Remove extra white spaces from the 'df' column
df['headline'] = df['headline'].str.strip()
df.head()
```

Figure 57 Removing the White Spaces

```
from bs4 import BeautifulSoup

# Function to remove HTML tags from df
def remove_html_tags(text):
    soup = BeautifulSoup(text, 'html.parser')
    return soup.get_text()

# Remove HTML tags from 'df' column
df['headline'] = df['headline'].apply(remove_html_tags)
```

Figure 58 Removing the HTML Tags

```
import re

# Define a function to remove URLs using regular expressions
def remove_urls(text):
    return re.sub(r'http\S+|www\S+', '', text)

# Apply the function to the 'headline' column
df['headline'] = df['headline'].apply(remove_urls)
```

Figure 59 Remove the URLs using RegEx

```
import string

string.punctuation

# Define the punctuation characters to remove
punctuation = string.punctuation

# Function to remove punctuation from "headline" column
def remove_punctuation(text):
    return text.translate(str.maketrans('', '', punctuation))

# Apply remove_punctuation function to 'headline' column
df['headline'] = df['headline'].apply(remove_punctuation)
```

Figure 60 Remove the Punctuation

Figures 61 to 65 show the steps involving removal of special characters, numeric values emojis, non-alphanumeric characters, and stopwords which are all categorized as noise data.

```
def remove_special_characters(text):
    # Define the pattern to match special characters
    pattern = r'[^a-zA-Z0-9\s]'  # Matches any character that is not alphanumeric or whitespace

    # Replace special characters with an empty string
    clean_text = re.sub(pattern, '', text)

    return clean_text

# Apply the function to the 'headline' column
df['headline'] = df['headline'].apply(remove_special_characters)
```

Figure 61 Remove the Special Characters

```
# Function to remove numeric values from text
def remove_numeric(text):
    return re.sub(r'\d+', '', text)

# Apply the function to the "headline" column
df['headline'] = df['headline'].apply(remove_numeric)
```

Figure 62 Remove Numeric Values

```
import emoji

# Function to remove emojis from text
def remove_emojis(text):
    return emoji.demojize(text)

# Apply remove_emojis function to 'headline' column
df['headline'] = df['headline'].apply(remove_emojis)
```

Figure 63 Remove Emojis

```
# Define a function to remove non-alphanumeric characters
def remove_non_alphanumeric(text):
    return re.sub(r'[^a-zA-Z0-9\s]', '', text)

# Apply the function to the "headline" column
df['headline'] = df['headline'].apply(remove_non_alphanumeric)
```

Figure 64 Remove Non-Alphanumeric Characters

```
import nltk
from nltk.corpus import stopwords

# Download NLTK stopwords corpus
nltk.download('stopwords')

# Get English stopwords from NLTK
stop_words = set(stopwords.words('english'))

# Function to remove stop words from text
def remove_stopwords(text):
    words = text.split()
    filtered_words = [word for word in words if word.lower() not in stop_words]
    return ' '.join(filtered_words)

# Apply remove_stopwords function to 'headline' column
df['headline'] = df['headline'].apply(remove_stopwords)
```

Figure 65 Remove Stopwords

Stop words are commonly used words in any language (such as "the," "is," "in," "for" in English language) that are often disregarded before processing text as they contribute minimal useful information about the text's content. NLTK provides a stop words corpus that aids in filtering out stop words from text (Kalaivani & Marivendan, 2021).

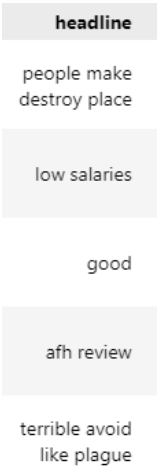


Figure 66 After Data Cleaning and Pre-Processing for Headline Column

4.4.5 Data Normalization - Lemmatization

The next step of data pre-processing would be Data Normalization by using lemmatization method. Previously, the student considered using stemming and found the stemmed headline's result is not up to the student's requirements as it looks like the words been chopped off as the PorterStemmer algorithm remove and replace suffixes. That is why the student chose to use lemmatization as it does not same thing but it's processed with the meaning of the word in mind (Khyani & B S, 2021).

```
# Download NLTK data
nltk.download('wordnet')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')

from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet

# Initialize the WordNetLemmatizer
lemmatizer = WordNetLemmatizer()

# Function to get the part of speech tag for lemmatization
def get_wordnet_pos(word):
    """Map POS tag to first character lemmatize() accepts"""
    tag = nltk.pos_tag([word])[0][1][0].upper()
    tag_dict = {"J": wordnet.ADJ,
                "N": wordnet.NOUN,
                "V": wordnet.VERB,
                "R": wordnet.ADV}
    return tag_dict.get(tag, wordnet.NOUN)

def lemmatize_text(text):
    words = text.split()
    lemmatized_words = [lemmatizer.lemmatize(word, get_wordnet_pos(word)) for word in words]
    return ' '.join(lemmatized_words)

# Apply the lemmatization function to 'headline' column
df['headline'] = df['headline'].apply(lemmatize_text)
```

Figure 67 Lemmatization using NLTK Module

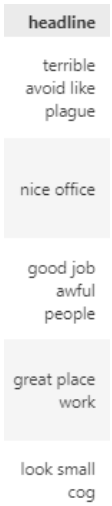


Figure 68 Lemmatization "headline" Result

4.4.6 Data Labelling Bert Model

Since the Kaggle dataset for job review does not have any predicted sentiment for any of the reviews, specifically the headline, it would be a massive chore for the student to manually assign a sentiment for them. Therefore, the student will be using a BERT model to automatically analyze the sentiment of the review (here means "headline") by predicting them. And at the same time, extract keywords from the reviews by using KeyBERT model for deployment purposes (Khan et al., 2022).

```
import torch
import pandas as pd
from torch.utils.data import DataLoader, Dataset
from transformers import BertTokenizer, BertForSequenceClassification
from keybert import KeyBERT
from tqdm import tqdm  # For progress bar

# Load pre-trained BERT model and tokenizer
tokenizer = BertTokenizer.from_pretrained('nlptown/bert-base-multilingual-uncased-sentiment')
model = BertForSequenceClassification.from_pretrained('nlptown/bert-base-multilingual-uncased-sentiment', num_labels=5)

# Initialize KeyBERT with a BERT model
kw_model = KeyBERT(model='paraphrase-MiniLM-L6-v2')
```

Figure 69 Extraction Keywords and Data Labelling using KeyBERT

```
# Custom dataset class
class SentimentDataset(Dataset):
    def __init__(self, texts, tokenizer, max_length=128):
        self.texts = texts
        self.tokenizer = tokenizer
        self.max_length = max_length

    def __len__(self):
        return len(self.texts)

    def __getitem__(self, idx):
        text = self.texts[idx]
        encoding = self.tokenizer.encode_plus(
            text,
            add_special_tokens=True,
            max_length=self.max_length,
            padding='max_length',
            truncation=True,
            return_tensors='pt'
        )
        input_ids = encoding['input_ids'].squeeze()
        attention_mask = encoding['attention_mask'].squeeze()
        return input_ids, attention_mask, text

# Example DataFrame (assuming `df` is your existing DataFrame)
data = df['headline']
df_subset = pd.DataFrame(data)

# Create dataset and dataloader
dataset = SentimentDataset(df_subset['headline'].tolist(), tokenizer)
dataloader = DataLoader(dataset, batch_size=16, shuffle=False)

# Function to extract keywords using KeyBERT
def extract_keywords(text):
    keywords = kw_model.extract_keywords(text, keyphrase_ngram_range=(1, 1), stop_words=None)
    return ', '.join([keyword for keyword, score in keywords])
```

Figure 70 Extraction Keywords and Data Labelling using KeyBERT (Cont'd)

```
# Get attention scores and predicted sentiments
predicted_sentiments = []
labeled_keywords = []

for input_ids, attention_masks, texts in tqdm(dataloader, desc="Processing batches"):
    with torch.no_grad():
        outputs = model(input_ids, attention_mask=attention_masks)
        logits = outputs.logits
        probabilities = torch.softmax(logits, dim=1)
        predicted_labels = torch.argmax(probabilities, dim=1)
        predicted_sentiments.extend(predicted_labels.cpu().numpy())

        for text in texts:
            labeled_keywords.append(extract_keywords(text))

# Add additional columns to df_subset
## df_subset for aggregatedFiltered_dataset
df_subset['firm'] = df['firm']
df_subset['work_life_balance'] = df['work_life_balance']
df_subset['culture_values'] = df['culture_values']
df_subset['diversity_inclusion'] = df['diversity_inclusion']
df_subset['career_opp'] = df['career_opp']
df_subset['comp_benefits'] = df['comp_benefits']
df_subset['senior_mgmt'] = df['senior_mgmt']
df_subset['predicted_sentiment'] = predicted_sentiments
df_subset['labeled_keywords'] = labeled_keywords

## df_reviews for listOfReviews_dataset
df_reviews = df
df_reviews['predicted_sentiment'] = predicted_sentiments

# Display the DataFrame with predicted sentiment and labeled keywords
df_subset.head()
```

Figure 71 Extraction Keywords and Data Labelling using KeyBERT (Cont'd)

predicted_sentiment	labeled_keywords
0	plague, terrible, avoid, like
3	office, nice
0	job, awful, good, people
4	work, place, great
2	cog, small, look

Figure 72 Predicted sentiment and labeled keywords by Lemmatization

The process of executing this particular extraction keywords and data labelling model takes around 100 to 120 minutes, which is about 2 hours, with the size of 45,241 rows. Max_length is fixed to 128 as it is enough for most cases.

After predicted sentiment, let's make a good or bad column

4.4.7 Data Post-Processing

In this step, data post-processing is done to make sure that the headlines are cleaned and there are no nulls for labeled keywords and headline. First, let's save our dataframes to CSV format and load the back to refresh the cache.

Save the Datasets

```
# Checkpoint for df_subset by export to CSV (Middle Part before Combining)
df_subset.to_csv(r'C:\Users\setia\Downloads\df_subset_labeled_lemma.csv', index=False, header=True)
df_reviews.to_csv(r'C:\Users\setia\Downloads\df_reviews_labeled_lemma.csv', index=False, header=True)
```

✓ 0.3s

Figure 73 Saving the Dataframes into CSVs format

```
import pandas as pd
df_subset = pd.read_csv('C:/Users/setia/Downloads/df_subset_labeled_lemma.csv')
df_reviews = pd.read_csv('C:/Users/setia/Downloads/df_reviews_labeled_lemma.csv')
```

Figure 74 Loading Back the Labeled and Lemmatized Datasets

```
df_subset.isna().sum()
```

✓ 0.0s

```
headline              109
firm                    0
work_life_balance       0
culture_values          0
diversity_inclusion     0
career_opp              0
comp_benefits           0
senior_mgmt             0
predicted_sentiment     0
labeled_keywords      139
dtype: int64
```

Figure 75 Check Total of NULL Values in Subset DataFrame

```
df_reviews.isna().sum()
```
✓ 0.0s

```
firm                    0
date_review             0
job_title               0
current                 0
location                0
overall_rating          0
work_life_balance       0
culture_values          0
diversity_inclusion     0
career_opp              0
comp_benefits           0
senior_mgmt             0
recommend               0
ceo_approv              0
outlook                 0
headline              109
pros                    0
cons                    0
predicted_sentiment     0
labeled_keywords      139
dtype: int64
```

Figure 76 Check Total of NULL Values in Reviews DataFrame

As seen Figures above, lets drop any nulls

```
df_subset = df_subset.dropna()
df_subset.isna().sum()
```
✓ 0.0s

```
headline              0
firm                  0
work_life_balance     0
culture_values        0
diversity_inclusion   0
career_opp            0
comp_benefits         0
senior_mgmt           0
predicted_sentiment   0
labeled_keywords      0
dtype: int64
```

Figure 77 Dropping and Checking the NULL Values of Subset DataFrame

```
df_reviews = df_reviews.dropna()
df_reviews.isna().sum()
```

✓ 0.0s

```
firm                  0
date_review           0
job_title             0
current               0
location              0
overall_rating        0
work_life_balance     0
culture_values        0
diversity_inclusion   0
career_opp            0
comp_benefits         0
senior_mgmt           0
recommend             0
ceo_approv            0
outlook               0
headline              0
pros                  0
cons                  0
predicted_sentiment   0
labeled_keywords      0
dtype: int64
```

Figure 78 Dropping and Checking the NULL Values of Reviews DataFrame

As the student has cleaned the datasets, now let's add another column from the 'predicted_sentiment' column with the name of 'label' for better clarity and as a means of preparation for model training as it has values of "good" and "bad."

```
# Create the new column 'label'
def categorize_sentiment(sentiment):
    if sentiment in [0, 1]:
        return 'bad'
    elif sentiment in [3, 4]:
        return 'good'
    else:
        return 'neutral'

# Apply sentiment categorization
df_reviews.loc[:, ['label']] = df_reviews['predicted_sentiment'].apply(categorize_sentiment)
df_subset['label'] = df_subset['predicted_sentiment'].apply(categorize_sentiment)

# Drop rows where 'predicted_sentiment' is 2
df_subset = df_subset[df_subset['predicted_sentiment'] != 2]
df_reviews = df_reviews[df_reviews['predicted_sentiment'] != 2]
```

Figure 79 Replace the Values with Either "Good" or "Bad"

```
df_subset.label.value_counts()
```

✓ 0.0s

```
label
good    38825
bad      3398
Name: count, dtype: int64
```

Figure 80 Labels Distribution in Subset DataFrame

```
df_subset.predicted_sentiment.value_counts()
```
✓ 0.0s

```
predicted_sentiment
4    23236
3    15589
0     2868
1      530
Name: count, dtype: int64
```

Figure 81 Predicted Sentiment Distribution in Subset DataFrame

Since the student needs to test the classification models later, let's make an aggregated list of keywords based on the firm. At the same time, export a csv file an unaggregated version of it to print detailed reviews.

```
## df_subset
# Group by 'firm' and aggregate
df_subset_combined = df_subset.groupby('firm').agg({
    'work_life_balance': 'mean',
    'culture_values': 'mean',
    'diversity_inclusion': 'mean',
    'career_opp': 'mean',
    'comp_benefits': 'mean',
    'senior_mgmt': 'mean',
    'predicted_sentiment': 'mean',  # Average the predicted sentiments
    'labeled_keywords': lambda x: ' '.join(x)  # Combine labeled keywords
}).reset_index()

# Drop the 'headline' column as it's no longer needed
df_subset_combined.drop(columns=['headline'], inplace=True, errors='ignore')

# Save to CSV
df_subset_combined.to_csv(r'C:\Users\setia\Downloads\aggregatedFiltered_Dataset.csv', index=False, header=True)

## df_reviews
# Save to CSV
df_reviews .to_csv(r'C:\Users\setia\Downloads\listOfReviews_Dataset.csv', index=False, header=True)
```

Figure 82 Exporting the DataFrames as CSVs

The exported CSVs are aggregatedFiltered_Dataset and listOfReviews_Dataset which will be further used for deployment purposes.

```
# Display the combined DataFrame
df_subset_combined.head()
```
✓ 0.3s

	firm	work_life_balance	culture_values	diversity_inclusion	career_opp	comp_benefits	senior_mgmt	predicted_sentiment	labeled_keywords
0	AFH-Wealth-Management	1.000000	2.000000	1.000000	1.000000	1.000000	1.000000	0.000000	plague, terrible, avoid, like
1	AJ-Bell	3.500000	3.000000	2.875000	3.250000	2.687500	3.000000	3.250000	office, nice job, awful, good, people work, pl...
2	ALDI	3.396552	3.896552	3.931034	3.810345	3.879310	3.758621	3.310345	family aldi reward, intense aldi, stacker, she...
3	AQA	4.500000	4.250000	4.500000	4.000000	4.000000	4.000000	3.750000	company, great work, place, great flexible wor...
4	ASDA	3.454545	3.602273	3.892045	3.323864	3.460227	3.289773	2.892045	asda, review company, terrible company, terrib...

Figure 83 Example of the Aggregated Filtered Dataset

4.4.8 Data Selection (Deletion)

For the process of data selection/deletion and balancing, the student has three ways to approach this step as it has its pros and cons, such as:

Table 6 Three Approaches to Data Selection & Balancing

No.	Balancing Based On	PROS	CONS
1.	Overall Rating from User	**Reflects User Experience**: The overall rating given by users can be a strong indicator of their sentiment and experience with the company. **Direct Metric**: This is a straightforward approach since the overall rating is a direct metric provided by the users.	**Potential Bias**: If users tend to give extreme ratings (very high or very low), it might not capture nuanced sentiments accurately. **Review Length and Detail**: Users might give an overall rating without detailed reviews, or their review content might not match the rating perfectly.
2.	Predicted Sentiment from BERT Model	**Consistent Classification**: Using a sophisticated model like BERT ensures that the sentiment classification is consistent across reviews. **Handles Nuance**: BERT can capture nuanced sentiments that might not be reflected accurately in the overall rating.	**Model Bias**: If the BERT model has any biases, it will reflect in the balancing. **Dependency on Model Accuracy**: This method relies heavily on the accuracy of the BERT model's predictions. If the model is not well-trained, it could lead to inaccurate balancing.
3.	Based on Both Overall Rating and Predicted Sentiment	**Holistic View**: Combining both metrics can give a more comprehensive view of the sentiment, considering both user-provided and model-predicted insights. **Reduces Bias**: This method can mitigate biases that might arise from using only one of the metrics.	**Complexity**: Implementing a balancing strategy based on two metrics can be more complex and may require more nuanced decision rules. **Inconsistencies**: There might be inconsistencies between the overall rating and the predicted sentiment, which need to be handled carefully.

Based on Table 6, the student has chosen to do the second approach as it correlates with the Final Year Project's title. The data selection and balancing will be based on predicted sentiment of the reviews, which is the main focus of the project.

This step can only be done when the student has labeled the sentiment of each review using BERT model which will be done in section 4.3.1 of this FYP. Therefore, do keep in mind that although deletion and balancing of dataset is included in the Data Pre-Processing, for the sake of this Final Year Project, it will be a standalone process as it will only be executed once the dataset has been labeled.

```
df_subset.predicted_sentiment.value_counts()
✓ 0.0s

predicted_sentiment
4    23236
3    15589
0     2868
1      530
Name: count, dtype: int64
```

Figure 84 Value Count of Predicted Sentiment in Subset DataFrame

As seen on the above Figure 84, the student needs to reduce the "good" data or the total of predicted sentiments that have value above or equal to 3 to be close enough with the "bad" data (values of 1 and 2) so that data balancing can be done.

```
# Step 1: Filter rows where `predicted_sentiment` is >= 3
selected_df = df_subset[df_subset['predicted_sentiment'] >= 3]

# Step 2: Define the random deletion function
def random_delete(df, fraction):
    return df.sample(frac=(1 - fraction), random_state=42).reset_index(drop=True)

# Step 3: Apply random deletion to the filtered subset (50% of the rows)
remaining_selected_df = random_delete(selected_df, 0.5)

# Step 4: Combine the remaining rows from the filtered subset with the rows that didn't meet the criteria
non_selected_df = df_subset[df_subset['predicted_sentiment'] < 3]
df_subset = pd.concat([remaining_selected_df, non_selected_df]).reset_index(drop=True)
```

Figure 85 Data Selection (Deletion)

```
df_subset.predicted_sentiment.value_counts()
✓ 0.0s

predicted_sentiment
4    11655
3     7757
0     2868
1      530
Name: count, dtype: int64
```

Figure 86 After Data Selection Result

After executing the code shown in Figure 85, the student can have a more balanced dataset based on the predicted sentiment.

```
df_subset.info()
✓ 0.0s

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22810 entries, 0 to 22809
Data columns (total 11 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   headline             22810 non-null  object
 1   firm                 22810 non-null  object
 2   work_life_balance    22810 non-null  int64
 3   culture_values       22810 non-null  int64
 4   diversity_inclusion  22810 non-null  int64
 5   career_opp           22810 non-null  int64
 6   comp_benefits        22810 non-null  int64
 7   senior_mgmt          22810 non-null  int64
 8   predicted_sentiment  22810 non-null  int64
 9   labeled_keywords     22810 non-null  object
 10  label                22810 non-null  object
dtypes: int64(7), object(4)
memory usage: 1.9+ MB
```

Figure 87 Info of Subset DataFrame After Data Deletion

4.4.9 Data Balancing

For the Data Balancing step, the student has two ways to ensure that the dataset of job reviews is balanced with ease and without errors. This can be done by using two approaches for balancing dataset, such as:

4.4.9.1 Data Balancing using Python Script

```
from sklearn.utils import resample

# Balancing Data Using Python
def balance_data(df, target_column):
    balanced_df = pd.DataFrame()
    min_class_size = df[target_column].value_counts().min()
    for class_value in df[target_column].unique():
        class_subset = df[df[target_column] == class_value]
        if len(class_subset) > 0:
            balanced_class_subset = resample(class_subset,
                                             replace=True,
                                             n_samples=min_class_size,
                                             random_state=42)
            balanced_df = pd.concat([balanced_df, balanced_class_subset])
    return balanced_df.sample(frac=1).reset_index(drop=True)  # Shuffle the balanced dataframe

# Balance the dataset based on 'label'
df_subset = balance_data(df_subset, 'label')
```

Figure 88 Python Script for Balancing (ASB & PS)

The student use label as the criteria for balancing because the final system will build a classification model for sentiment analysis on “bad” or “good”

	headline	firm	work_life_balance	culture_values	diversity_inclusion	career_opp	comp_benefits	senior_mgmt	predicted_sentiment	labeled_keywords	label
0	sometim frustrat	Compass-Group	4	3	4	3	1	4	1	##tim ##strat fru some	0
1	best place work	McKinsey-and-Company	5	5	5	5	5	5	4	place work best	1
2	sap destroy acquir brand	SAP	1	1	5	2	3	1	0	ac destroy brand ##qui ##r	0
3	good great	Apple	3	5	5	3	4	5	4	great good	1
4	bleed blue learnersof	IBM	3	5	5	3	2	3	0	##ed blue learn ##of ##ers	0

Figure 89 Python Script for Balancing (ASB & PS) Result

Python script is the easiest way to balance the data when dealing with complex models that need certain dataset to be modified in a complex way.

4.4.9.2 Data Balancing using SMOTE

Another way when dealing with imbalanced dataset is by using SMOTE (Synthetic Minority Oversampling Technique) which is a type of Data Augmentation for the minority class. It is done by oversampling the examples in the minority class which balance the class distribution and at the same time , synthesizing new examples (Brownlee, 2020). SMOTE works by selecting nearby samples in the feature space, drawing a line between them, then drawing a new sample at a position along that line.

```
from imblearn.over_sampling import SMOTE
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split

# Define X (features) and y (target)
# X = df_subset['pros'] + ' ' + df_subset['cons']  # Concatenate pros and cons for text input
X = df_subset['headline']
y = df_subset['average_predicted_sentiment']

# Vectorize the text data
vectorizer = TfidfVectorizer(max_features=5000)  # Adjust max_features as needed
X_vec = vectorizer.fit_transform(X)

# Convert string labels to numerical categories
label_mapping = {"bad": 0, "good": 1}
y = y.map(label_mapping)

# Split the data into train and test sets (80-20 split)
X_train, X_test, y_train, y_test = train_test_split(X_vec, y, test_size=0.2, random_state=42)

# Further split train into train and validation sets (80-20 split of the 80%)
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.2, random_state=42)

# Display the shape of each set
print("Train set shape:", X_train.shape, y_train.shape)
print("Validation set shape:", X_val.shape, y_val.shape)
print("Test set shape:", X_test.shape, y_test.shape)
```

Figure 90 SMOTE Code for Balancing

```
Train set shape: (8352, 3263) (8352,)
Validation set shape: (2088, 3263) (2088,)
Test set shape: (2611, 3263) (2611,)
```

Figure 91 SMOTE Code Result

```
from imblearn.over_sampling import SMOTE

# Apply SMOTE for balancing on the training set only
smote = SMOTE(random_state=42)
X_train_balanced, y_train_balanced = smote.fit_resample(X_train, y_train)

# Display the class distribution before and after balancing
print("Class distribution before SMOTE:")
print(y_train.value_counts())

print("\nClass distribution after SMOTE:")
print(pd.Series(y_train_balanced).value_counts())
```

Figure 92 SMOTE Code for Balancing (Cont'd)

```
Class distribution before SMOTE:
label
1    6021
0    2331
Name: count, dtype: int64

Class distribution after SMOTE:
label
0    6021
1    6021
Name: count, dtype: int64
```

Figure 93 Result Before and After SMOTE

Overall, the project pipeline will start from Model Sentiment Labeling & Keywords Extraction → Data Selection → Data Balancing → Model Building.

4.5 Model Buildings

4.5.1 BERT Classification Model

```
import torch
import pandas as pd
from torch.utils.data import DataLoader, Dataset
from sklearn.preprocessing import LabelEncoder
from transformers import BertForSequenceClassification, BertTokenizer, get_linear_schedule_with_warmup
from torch.optim import AdamW

# Load pre-trained BERT model and tokenizer for sequence classification
model = BertForSequenceClassification.from_pretrained('bert-base-uncased', num_labels=2)
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

# Custom dataset class
class SentimentDataset(Dataset):
    def __init__(self, texts, labels, tokenizer, max_length=128):
        self.texts = texts
        self.labels = labels
        self.tokenizer = tokenizer
        self.max_length = max_length
        self.encodings = self.tokenizer(texts, truncation=True, padding=True, max_length=self.max_length, return_tensors="pt")

    def __len__(self):
        return len(self.texts)

    def __getitem__(self, idx):
        item = {key: val[idx] for key, val in self.encodings.items()}
        item['labels'] = torch.tensor(self.labels[idx])
        return item

X = df_subset['headline'].tolist()
y = df_subset['label'].tolist()  # Replace with your target column

# Use LabelEncoder to convert string labels to numerical categories
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)
```

Figure 94 BERT Classification Model Code

Transformer-based Models or Advanced models like BERT (Bidirectional Encoder Representations from Transformers) have shown state-of-the-art performance in various NLP tasks (Qasim et al., 2022). BERT is simple yet powerful as it has its own tokenizer that anybody can import and pretrained already, therefore, the student does not use SMOTE-balanced dataset, instead undersampled dataset was used which is shown in section 4.4.9.2.

The LabelEncoder used by sklearn automatically assigns numerical values to categorical string labels based on alphabetical order. In this case, it will assign the labels bad to 0 and good to 1 because “b” comes before “g” alphabetically.

All the classification models take the headline column and the predicted sentiment from the previous data labelling model, and it will be trained and tested. Later, the model will be used on the raw dataset to classify the reviews of the selected firm by the user whether it is good or bad.

```
# Create the dataset
dataset = SentimentDataset(X, y_encoded, tokenizer)

# Split the dataset into training, validation, and test sets
train_size = int(0.6 * len(dataset))
val_size = int(0.2 * len(dataset))
test_size = len(dataset) - train_size - val_size

train_dataset, val_dataset, test_dataset = torch.utils.data.random_split(dataset, [train_size, val_size, test_size])

# Create dataloaders
train_dataloader = DataLoader(train_dataset, batch_size=16, shuffle=True)
val_dataloader = DataLoader(val_dataset, batch_size=16, shuffle=False)
test_dataloader = DataLoader(test_dataset, batch_size=16, shuffle=False)

# Training setup
epochs = 8
optimizer = AdamW(model.parameters(), lr=2e-5)
total_steps = len(train_dataloader) * epochs
scheduler = get_linear_schedule_with_warmup(optimizer, num_warmup_steps=0, num_training_steps=total_steps)
criterion = torch.nn.CrossEntropyLoss()
```

Figure 95 BERT Classification Model Code (Cont'd)

The split of the dataset into training, validation, and test sets with ratios of 60%, 20%, and 20% is generally reasonable, even for a dataset with 50,000 rows. For more complex models like BERT, having a larger training set can be beneficial to capture the diverse patterns in the data.

```
# Training function with accuracy tracking
def train(model, train_dataloader, val_dataloader, optimizer, scheduler, criterion, epochs):
    model.train()
    train_losses = []
    val_losses = []
    train_accuracies = []
    val_accuracies = []

    for epoch in range(epochs):
        total_loss = 0
        correct_predictions = 0
        total_predictions = 0

        for batch in train_dataloader:
            optimizer.zero_grad()
            input_ids = batch['input_ids'].to(device)
            attention_mask = batch['attention_mask'].to(device)
            labels = batch['labels'].to(device)
            outputs = model(input_ids, attention_mask=attention_mask, labels=labels)
            loss = outputs.loss
            logits = outputs.logits
            loss.backward()
            optimizer.step()
            scheduler.step()
            total_loss += loss.item()

            # Calculate training accuracy
            preds = torch.argmax(logits, dim=1)
            correct_predictions += (preds == labels).sum().item()
            total_predictions += labels.size(0)

        avg_train_loss = total_loss / len(train_dataloader)
        train_accuracy = correct_predictions / total_predictions

        train_losses.append(avg_train_loss)
        train_accuracies.append(train_accuracy)

        # Validation
        model.eval()
        val_loss = 0
        val_correct_predictions = 0
        val_total_predictions = 0
```

Figure 96 BERT Classification Model Code (Cont'd)

```
        with torch.no_grad():
            for batch in val_dataloader:
                input_ids = batch['input_ids'].to(device)
                attention_mask = batch['attention_mask'].to(device)
                labels = batch['labels'].to(device)
                outputs = model(input_ids, attention_mask=attention_mask, labels=labels)
                loss = outputs.loss
                logits = outputs.logits
                val_loss += loss.item()

                preds = torch.argmax(logits, dim=1)
                val_correct_predictions += (preds == labels).sum().item()
                val_total_predictions += labels.size(0)

        avg_val_loss = val_loss / len(val_dataloader)
        val_accuracy = val_correct_predictions / val_total_predictions

        val_losses.append(avg_val_loss)
        val_accuracies.append(val_accuracy)

        print(f"Epoch {epoch + 1}/{epochs}")
        print(f"Train Loss: {avg_train_loss:.4f}, Train Accuracy: {train_accuracy:.4f}")
        print(f"Val Loss: {avg_val_loss:.4f}, Val Accuracy: {val_accuracy:.4f}")

        model.train()

    return train_losses, val_losses, train_accuracies, val_accuracies

# Train the model
device = torch.device('cuda') if torch.cuda.is_available() else torch.device('cpu')
model.to(device)
train_losses, val_losses, train_accuracies, val_accuracies = train(model, train_dataloader, val_dataloader, optimizer, scheduler, criterion, epochs)
```

Figure 97 BERT Classification Model Code (Cont'd)

```
Epoch 1/8 - Training: 100%|██████████| 255/255 [07:14<00:00,  1.70s/it, accuracy=0.924, loss=0.205]
Epoch 1/8 - Validation: 100%|██████████| 85/85 [00:32<00:00,  2.65it/s, accuracy=0.962, loss=0.115]
Epoch 1/8
Train Loss: 0.2046, Train Accuracy: 0.9245
Val Loss: 0.1147, Val Accuracy: 0.9617
Epoch 2/8 - Training: 100%|██████████| 255/255 [08:18<00:00,  1.96s/it, accuracy=0.977, loss=0.0743]
Epoch 2/8 - Validation: 100%|██████████| 85/85 [00:31<00:00,  2.67it/s, accuracy=0.966, loss=0.11]
Epoch 2/8
Train Loss: 0.0743, Train Accuracy: 0.9774
Val Loss: 0.1099, Val Accuracy: 0.9662
Epoch 3/8 - Training: 100%|██████████| 255/255 [08:15<00:00,  1.94s/it, accuracy=0.989, loss=0.0345]
Epoch 3/8 - Validation: 100%|██████████| 85/85 [00:25<00:00,  3.30it/s, accuracy=0.966, loss=0.124]
Epoch 3/8
Train Loss: 0.0345, Train Accuracy: 0.9890
Val Loss: 0.1242, Val Accuracy: 0.9662
Epoch 4/8 - Training: 100%|██████████| 255/255 [07:43<00:00,  1.82s/it, accuracy=0.996, loss=0.0176]
Epoch 4/8 - Validation: 100%|██████████| 85/85 [00:25<00:00,  3.27it/s, accuracy=0.969, loss=0.128]
Epoch 4/8
Train Loss: 0.0176, Train Accuracy: 0.9958
Val Loss: 0.1278, Val Accuracy: 0.9691
Epoch 5/8 - Training: 100%|██████████| 255/255 [06:44<00:00,  1.59s/it, accuracy=0.997, loss=0.0121]
Epoch 5/8 - Validation: 100%|██████████| 85/85 [00:24<00:00,  3.40it/s, accuracy=0.969, loss=0.142]
Epoch 5/8
Train Loss: 0.0121, Train Accuracy: 0.9971
Val Loss: 0.1424, Val Accuracy: 0.9691
Epoch 6/8 - Training: 100%|██████████| 255/255 [05:58<00:00,  1.40s/it, accuracy=0.998, loss=0.00939]
Epoch 6/8 - Validation: 100%|██████████| 85/85 [00:24<00:00,  3.43it/s, accuracy=0.971, loss=0.117]
Epoch 6/8
Train Loss: 0.0094, Train Accuracy: 0.9975
Val Loss: 0.1172, Val Accuracy: 0.9713
Epoch 7/8 - Training: 100%|██████████| 255/255 [05:58<00:00,  1.40s/it, accuracy=0.997, loss=0.00807]
Epoch 7/8 - Validation: 100%|██████████| 85/85 [00:24<00:00,  3.40it/s, accuracy=0.97, loss=0.144]
Epoch 7/8
Train Loss: 0.0081, Train Accuracy: 0.9973
Val Loss: 0.1444, Val Accuracy: 0.9698
Epoch 8/8 - Training: 100%|██████████| 255/255 [05:59<00:00,  1.41s/it, accuracy=0.999, loss=0.00463]
Epoch 8/8 - Validation: 100%|██████████| 85/85 [00:25<00:00,  3.39it/s, accuracy=0.971, loss=0.145]
Epoch 8/8
Train Loss: 0.0046, Train Accuracy: 0.9990
Val Loss: 0.1446, Val Accuracy: 0.9706
```

Figure 98 Result After Lemmatization

```python
import matplotlib.pyplot as plt

# Plotting function
def plot_metrics(train_losses, val_losses, train_accuracies, val_accuracies):
    epochs = range(1, len(train_losses) + 1)

    plt.figure(figsize=(14, 5))

    plt.subplot(1, 2, 1)
    plt.plot(epochs, train_losses, 'b', label='Training loss')
    plt.plot(epochs, val_losses, 'r', label='Validation loss')
    plt.title('Training and validation loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()

    plt.subplot(1, 2, 2)
    plt.plot(epochs, train_accuracies, 'b', label='Training accuracy')
    plt.plot(epochs, val_accuracies, 'r', label='Validation accuracy')
    plt.title('Training and validation accuracy')
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.legend()

    plt.show()

plot_metrics(train_losses, val_losses, train_accuracies, val_accuracies)
```

Figure 99 Generating Graphs and Plots Python Code

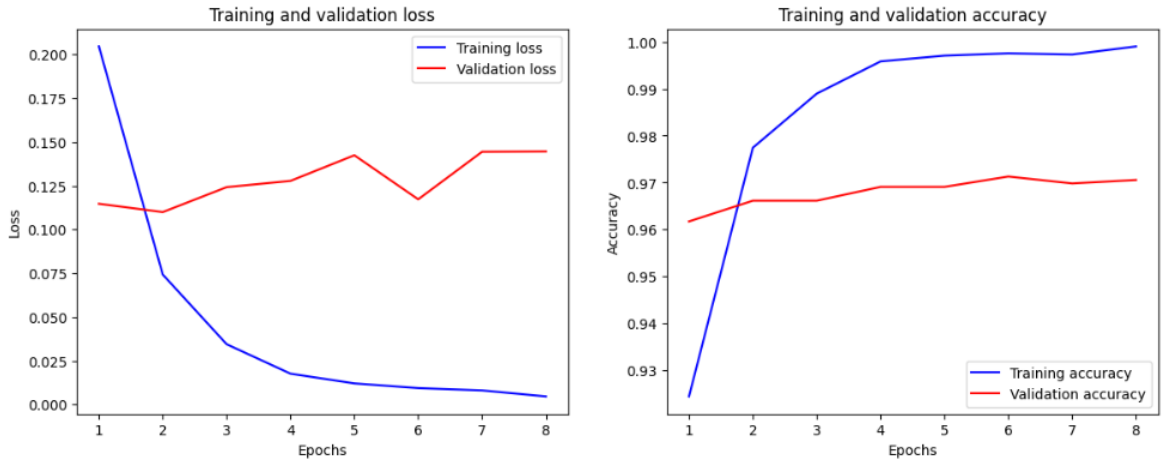


Figure 100 Graphs of Training and Validation's Loss and Accuracy

The training and validation loss graph above shows the training loss decreases steadily with each epoch, indicating that the model is learning and fitting the training data well. However, the validation loss initially decreases but then starts to increase, indicating potential overfitting. After the 3rd epoch, the validation loss does not decrease significantly and even shows some fluctuations, suggesting the model is starting to overfit the training data and not generalizing as well to the validation data.

Overall, the model performs remarkably well on training data but fails to generalize as well on validation data, as indicated by the divergence in training and validation loss and the plateau in validation accuracy.

```python
from sklearn.metrics import classification_report

# Evaluate on the validation set
def evaluate(dataloader):
    model.eval()
    true_labels = []
    predictions = []
    with torch.no_grad():
        for batch in dataloader:
            input_ids = batch['input_ids'].to(device)
            attention_mask = batch['attention_mask'].to(device)
            labels = batch['labels'].to(device)
            outputs = model(input_ids, attention_mask=attention_mask)
            logits = outputs.logits
            preds = torch.argmax(logits, dim=1)
            true_labels.extend(labels.cpu().numpy())
            predictions.extend(preds.cpu().numpy())
    return true_labels, predictions

val_true_labels, val_predictions = evaluate(val_dataloader)
test_true_labels, test_predictions = evaluate(test_dataloader)

# Generate classification report for validation set
val_report = classification_report(val_true_labels, val_predictions, target_names=['Bad', 'Good'])
print("Validation Classification Report:\n", val_report)

# Generate classification report for test set
test_report = classification_report(test_true_labels, test_predictions, target_names=['Bad', 'Good'])
print("Test Classification Report:\n", test_report)
```

Figure 101 Print Validation and Test Classification Report

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

         Bad       0.96      0.98      0.97       664
        Good       0.99      0.96      0.97       695

    accuracy                           0.97      1359
   macro avg       0.97      0.97      0.97      1359
weighted avg       0.97      0.97      0.97      1359

Test Classification Report:
              precision    recall  f1-score   support

         Bad       0.97      0.97      0.97       638
        Good       0.98      0.97      0.97       722

    accuracy                           0.97      1360
   macro avg       0.97      0.97      0.97      1360
weighted avg       0.97      0.97      0.97      1360
```

Figure 102 Validation and Classification Report Undersampling

The model consistently achieves 97% accuracy in both validation and test datasets, indicating it is well-calibrated and generalizes effectively to new data. Both precision and recall are balanced and high, suggesting the model is not biased towards false positives or false negatives. This balance is crucial in applications where both types of errors are costly.

The F1-score consolidates the high performance of the model, confirming that it maintains a good trade-off between precision and recall. The support values indicate that the dataset is balanced, which can contribute to the model's reliable performance across different classes.

4.5.2 SVM Classification Model

Support Vector Machine (SVM) is a supervised learning model used for classification tasks and it works by determining the hyperplane that best splits the data into different classes. In sentiment analysis, SVM is effective because it can handle high-dimensional spaces and is robust to overfitting, especially with text data. SVMs are particularly useful when the number of dimensions (features) exceeds the number of samples, a common scenario in text classification problems (Kristiyanti et al., 2019).

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import classification_report

# Train the SVM model
svm_model = SVC(probability=True, kernel='linear')
svm_model.fit(X_train_balanced, y_train_balanced)

# Evaluate the model on validation set
y_val_pred = svm_model.predict(X_val)
print("Validation Set Evaluation:")
print(classification_report(y_val, y_val_pred, target_names=['Bad', 'Good']))

# Evaluate the model on test set
y_test_pred = svm_model.predict(X_test)
print("Test Set Evaluation:")
print(classification_report(y_test, y_test_pred, target_names=['Bad', 'Good']))
```

Figure 103 SVM Classification Model Code

```
Validation Set Evaluation:
              precision    recall  f1-score   support

         Bad       0.91      0.96      0.93       542
        Good       0.95      0.91      0.93       546

    accuracy                           0.93      1088
   macro avg       0.93      0.93      0.93      1088
weighted avg       0.93      0.93      0.93      1088

Test Set Evaluation:
              precision    recall  f1-score   support

         Bad       0.92      0.96      0.94       679
        Good       0.96      0.92      0.94       681

    accuracy                           0.94      1360
   macro avg       0.94      0.94      0.94      1360
weighted avg       0.94      0.94      0.94      1360
```

Figure 104 Result Using Python Script (Undersampling)

The SVM model performs consistently across both validation and test sets with high precision, recall, and F1-scores. Balanced support across classes helps achieve reliable performance metrics.

```
Validation Set Evaluation:
              precision    recall  f1-score   support

         Bad       0.87      0.86      0.87       532
        Good       0.98      0.98      0.98      3118

    accuracy                           0.96      3650
   macro avg       0.93      0.92      0.92      3650
weighted avg       0.96      0.96      0.96      3650

Test Set Evaluation:
              precision    recall  f1-score   support

         Bad       0.89      0.86      0.87       683
        Good       0.97      0.98      0.98      3879

    accuracy                           0.96      4562
   macro avg       0.93      0.92      0.92      4562
weighted avg       0.96      0.96      0.96      4562
```

Figure 105 Result Using SMOTE Dataset (Oversampling)

The model shows high performance metrics for the "Good" class but slightly lower for the "Bad" class. The high accuracy on both validation and test sets suggests the model's generalization capability is strong, but the class imbalance needs to be addressed.

4.5.3 Logistic Regression Classification Model

Logistic Regression is a statistical model used for binary classification problems. It estimates the probability that a given input belongs to a particular class by fitting a logistic function to the data. In the context of sentiment analysis, logistic regression is widely used due to its simplicity and effectiveness in predicting the likelihood of positive or negative sentiment based on textual features (Jemai et al., 2021). It performs well with large datasets and is easy to interpret, making it a popular choice for sentiment classification.

```python
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report

# Train the Logistic Regression model
logreg_model = LogisticRegression(max_iter=1000)
logreg_model.fit(X_train, y_train)

# Evaluate the model on validation set
y_val_pred = logreg_model.predict(X_val)
print("Validation Set Evaluation:")
print(classification_report(y_val, y_val_pred, target_names=['Bad', 'Good']))

# Evaluate the model on test set
y_test_pred = logreg_model.predict(X_test)
print("Test Set Evaluation:")
print(classification_report(y_test, y_test_pred, target_names=['Bad', 'Good']))
```

Figure 106 Logistic Regression Classification Model

```
Validation Set Evaluation:
              precision    recall  f1-score   support

         Bad       0.88      0.97      0.92       542
        Good       0.97      0.87      0.92       546

    accuracy                           0.92      1088
   macro avg       0.93      0.92      0.92      1088
weighted avg       0.93      0.92      0.92      1088

Test Set Evaluation:
              precision    recall  f1-score   support

         Bad       0.88      0.97      0.92       679
        Good       0.96      0.87      0.91       681

    accuracy                           0.92      1360
   macro avg       0.92      0.92      0.92      1360
weighted avg       0.92      0.92      0.92      1360
```

Figure 107 Undersampling

The Logistic Regression model performs consistently with high precision, recall, and F1-scores across both validation and test sets. Balanced support across classes contributes to reliable performance metrics.

```
Validation Set Evaluation:
              precision    recall  f1-score   support

         Bad       0.85      0.87      0.86       532
        Good       0.98      0.97      0.98      3118

    accuracy                           0.96      3650
   macro avg       0.92      0.92      0.92      3650
weighted avg       0.96      0.96      0.96      3650

Test Set Evaluation:
              precision    recall  f1-score   support

         Bad       0.85      0.88      0.87       683
        Good       0.98      0.97      0.98      3879

    accuracy                           0.96      4562
   macro avg       0.91      0.93      0.92      4562
weighted avg       0.96      0.96      0.96      4562
```

Figure 108 Oversampling

The model shows high performance metrics for the "Good" class but lower for the "Bad" class. Overall accuracy of 0.96 suggests good generalization.

4.5.4 Random Forest Classification Model

Random Forest is an ensemble learning method that builds multiple decision trees and merges them to obtain a more accurate and stable prediction. For sentiment analysis, Random Forest can capture complex interactions between features and improve prediction accuracy by reducing overfitting. It works well with high-dimensional data, such as text, and can handle a large number of input variables, making it a robust choice for classifying sentiments in job reviews (Saputra et al., 2023).

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report

# Train the Random Forest model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train_balanced, y_train_balanced)

# Evaluate the model on validation set
y_val_pred = rf_model.predict(X_val)
print("Validation Set Evaluation:")
print(classification_report(y_val, y_val_pred, target_names=['Bad', 'Good']))

# Evaluate the model on test set
y_test_pred = rf_model.predict(X_test)
print("Test Set Evaluation:")
print(classification_report(y_test, y_test_pred, target_names=['Bad', 'Good']))
```

Figure 109 Random Forest Classification Model

```
Validation Set Evaluation:
              precision    recall  f1-score   support

         Bad       0.87      0.95      0.91       542
        Good       0.95      0.86      0.90       546

    accuracy                           0.90      1088
   macro avg       0.91      0.90      0.90      1088
weighted avg       0.91      0.90      0.90      1088

Test Set Evaluation:
              precision    recall  f1-score   support

         Bad       0.88      0.96      0.92       679
        Good       0.95      0.87      0.91       681

    accuracy                           0.91      1360
   macro avg       0.92      0.91      0.91      1360
weighted avg       0.92      0.91      0.91      1360
```

Figure 110 Undersampling

The Random Forest model performs well with high precision and recall for the "Bad" class but needs improvement for the "Good" class. The overall accuracy is good, but efforts should be made to balance the performance across classes.

```
Validation Set Evaluation:
              precision    recall  f1-score   support

         Bad       0.93      0.81      0.86       532
        Good       0.97      0.99      0.98      3118

    accuracy                           0.96      3650
   macro avg       0.95      0.90      0.92      3650
weighted avg       0.96      0.96      0.96      3650

Test Set Evaluation:
              precision    recall  f1-score   support

         Bad       0.91      0.81      0.86       683
        Good       0.97      0.99      0.98      3879

    accuracy                           0.96      4562
   macro avg       0.94      0.90      0.92      4562
weighted avg       0.96      0.96      0.96      4562
```

Figure 111 Oversampling

The model performs excellently for the "Good" class but shows a drop in performance for the "Bad" class. High overall accuracy indicates strong generalization, but the class imbalance needs addressing.

4.5.5 Naïve Bayes Classification Model

Naive Bayes is a probabilistic classifier based on Bayes' Theorem, assuming independence between predictors. Despite the simplicity of this assumption, Naive Bayes performs well in text classification tasks, including sentiment analysis, because it efficiently handles large vocabularies and sparse data. It is particularly effective for classifying sentiments due to its ability to deal with high-dimensional feature spaces and provide quick, reliable results (Wongkar & Angdresey, 2019).

```python
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report

# Train the Naive Bayes model
nb_model = MultinomialNB()
nb_model.fit(X_train_balanced, y_train_balanced)

# Evaluate the model on validation set
y_val_pred = nb_model.predict(X_val)
print("Validation Set Evaluation:")
print(classification_report(y_val, y_val_pred, target_names=['Bad', 'Good']))

# Evaluate the model on test set
y_test_pred = nb_model.predict(X_test)
print("Test Set Evaluation:")
print(classification_report(y_test, y_test_pred, target_names=['Bad', 'Good']))
```

Figure 112 Naïve Bayes Classification Model

```
Validation Set Evaluation:
              precision    recall  f1-score   support

         Bad       0.92      0.92      0.92       542
        Good       0.92      0.92      0.92       546

    accuracy                           0.92      1088
   macro avg       0.92      0.92      0.92      1088
weighted avg       0.92      0.92      0.92      1088

Test Set Evaluation:
              precision    recall  f1-score   support

         Bad       0.93      0.93      0.93       679
        Good       0.93      0.93      0.93       681

    accuracy                           0.93      1360
   macro avg       0.93      0.93      0.93      1360
weighted avg       0.93      0.93      0.93      1360
```

Figure 113 Undersampling

The Naive Bayes model performs consistently across both classes, with high precision, recall, and F1-scores. The overall accuracy of 0.92 indicates a strong model.

```
Validation Set Evaluation:
              precision    recall  f1-score   support

         Bad       0.70      0.91      0.79       532
        Good       0.98      0.93      0.96      3118

    accuracy                           0.93      3650
   macro avg       0.84      0.92      0.88      3650
weighted avg       0.94      0.93      0.93      3650

Test Set Evaluation:
              precision    recall  f1-score   support

         Bad       0.73      0.93      0.82       683
        Good       0.99      0.94      0.96      3879

    accuracy                           0.94      4562
   macro avg       0.86      0.93      0.89      4562
weighted avg       0.95      0.94      0.94      4562
```

Figure 114 Oversampling

The model shows high precision but struggles with recall for the "Bad" class, leading to lower F1-scores. The overall accuracy of 0.70 on the validation set suggests potential overfitting or class imbalance issues.

4.6 Summary

In Chapter 4, the student collected the data from Kaggle.com and the Data Understanding phase entails loading and exploring to determine their relevance and suitability. This phase involves in-depth analysis and comparison of datasets to identify overlaps and unique characteristics. The initial data preprocessing steps are outlined, focusing on selecting pertinent data, standardizing column names, and cleaning the datasets.

Additionally, the chapter describes the preliminary data preprocessing steps, including data cleansing, punctuation removal, word tokenization, stop word elimination, and word stemming, utilizing tools like NLTK. These steps are vital for preparing the textual data for feature extraction and subsequent analysis in the latter part of the project. The student also has designed and tested the models in which results has been shown below:

Table 7 Accuracy Result of Tested Models

Sampling Method	Model Name	Test Accuracy
Undersampling (Python Script)	BERT Model	0.97
	SVM Model	0.94
	Logistic Regression Model	0.92
	Random Forest Model	0.91
	Naïve Bayes Model	0.93
Oversampling (SMOTE)	BERT Model	-
	SVM Model	0.96
	Logistic Regression Model	0.96
	Random Forest Model	0.96
	Naïve Bayes Model	0.94

The above Table 7 shows the result of the accuracy of the five (5) models on test dataset which is to remind the student of the types of models that were generated through the building and training, as well as all the combinations of different extraction approaches.

Based on Table 7, the **BERT** model outperforms others with undersampling, indicating its robustness to class imbalance without needing synthetic data. Meanwhile, **SVM and Logistic Regression** models perform similarly well with oversampling, suggesting that linear models benefit significantly from balanced datasets. **Random Forest** model shows consistent improvement with SMOTE, reflecting its ability to handle diverse datasets effectively. Lastly, **Naïve Bayes** model performs surprisingly well with undersampling and improves with oversampling, showing that even simpler models can benefit from balanced data.

Based on the observed performance, **BERT** model likely benefits from its deep architecture and pre-training on large text corpora, making it less sensitive to data imbalance. **SVM and Logistic Regression** which are linear models benefit from the balanced class distributions

provided by SMOTE, leading to higher accuracy.: **Random Forest**'s improved performance with SMOTE could be due to model's ensemble nature, which benefits from more diverse and balanced samples. Lastly, **Naïve Bayes** model's increase in accuracy with SMOTE might be attributed to its probabilistic nature, which benefits from a more balanced class distribution. Based on Table 3 which was in section 2.2.2 and Table 4 in section 2.3, the above Table 7 shows an accurate explanation of the reason the last two models have moderate accuracies compared to the first three models.

CHAPTER 5: RESULTS AND DISCUSSIONS

5.1 Introduction

In this chapter, the student presents the results and discussions of our sentiment analysis models, including BERT, SVM, Logistic Regression, Random Forest, and Naïve Bayes. The student also evaluates the performance of each model through detailed analysis of their respective hyperparameters and their impact on model accuracy and loss. Following this, a comprehensive discussion of the results is also provided, highlighting the strengths and limitations of each approach. Additionally, the student also describes the process of model deployment, utilizing pickled models and implementing a user-friendly interface with Streamlit. Finally, the student summarizes the key findings and implications of the study, paving the way for future research and practical applications in sentiment analysis.

5.2 Model Evaluations and Discussions

5.2.1 BERT Hyperparameter

There is no need for BERT hyper tuning and the model is pre-trained specifically for sentiment analysis and/or NLP-related tasks.

5.2.2 SVM Hyperparameter

```
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score

svm_parameter= SVC(random_state=42)

param_grid_svm = {
    'C': [0.1, 1, 10, 100],
    'kernel': ['linear', 'rbf', 'poly', 'sigmoid'],
    'gamma': ['scale', 'auto'],
    'degree': [2, 3, 4, 5]
}

grid_search_svm = GridSearchCV(estimator=svm_parameter, param_grid=param_grid_svm, cv=5, n_jobs=-1, verbose=2)

# Fit the grid search
grid_search_svm.fit(X_train, y_train)

# Best parameters
best_svm = grid_search_svm.best_estimator_

# Predict on the validation set
y_val_pred_svm = best_svm.predict(X_val)

# Calculate accuracy
accuracy_svm = accuracy_score(y_val, y_val_pred_svm)
print(f"SVM best parameters: {grid_search_svm.best_params_}")
print(f"SVM validation accuracy: {accuracy_svm}")
```

Figure 115 SVM Hyperparameter Python Code

```
Fitting 5 folds for each of 128 candidates, totalling 640 fits
SVM best parameters: {'C': 10, 'degree': 2, 'gamma': 'scale', 'kernel': 'rbf'}
SVM validation accuracy: 0.9273897058823529
```

Figure 116 Undersampling

The SVM with an RBF kernel achieved a validation accuracy of approximately 92.7%. The RBF kernel's ability to handle non-linear relationships in the data likely contributed to this high accuracy, even with a reduced dataset from undersampling. The high C value emphasizes the importance of correctly classifying training data points, and the 'scale' gamma allows for flexible adaptation to data variability.

```
Fitting 5 folds for each of 128 candidates, totalling 640 fits
SVM best parameters: {'C': 10, 'degree': 2, 'gamma': 'scale', 'kernel': 'linear'}
SVM validation accuracy: 0.9616438356164384
```

Figure 117 Oversampling

The SVM with a linear kernel achieved a validation accuracy of approximately 96.2%. The improved accuracy with the linear kernel suggests that the oversampled dataset might have been more linearly separable due to the balanced class distribution provided by SMOTE. The same high C value underlines the need to fit the training data accurately. The linear kernel's simplicity and effectiveness for linearly separable data contributed to the high accuracy.

Overall, SVM model's performance varies significantly depending on the sampling method and the kernel used:

- **Undersampling**: The RBF kernel's ability to handle non-linear data helped achieve high accuracy despite reduced data.
- **Oversampling**: The linear kernel performed better with the balanced data provided by SMOTE, achieving even higher accuracy.

5.2.3 Logistic Regression Hyperparameter

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV

lr_parameter = LogisticRegression(random_state=42)

param_grid_lr = {
    'C': [0.01, 0.1, 1, 10, 100],
    'penalty': ['l2'], # Remove elasticnet and l1
    'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga']
}

grid_search_lr = GridSearchCV(estimator=lr_parameter, param_grid=param_grid_lr, cv=5, n_jobs=-1, verbose=2)

# Fit the grid search
grid_search_lr.fit(X_train, y_train)

# Best parameters
best_lr = grid_search_lr.best_estimator_

# Predict on the validation set
y_val_pred_lr = best_lr.predict(X_val)

# Calculate accuracy
accuracy_lr = accuracy_score(y_val, y_val_pred_lr)
print(f"Logistic Regression best parameters: {grid_search_lr.best_params_}")
print(f"Logistic Regression validation accuracy: {accuracy_lr}")
```

Figure 118 Logistic Regression Hyperparameter Python Code

```
Fitting 5 folds for each of 25 candidates, totalling 125 fits
Logistic Regression best parameters: {'C': 100, 'penalty': 'l2', 'solver': 'liblinear'}
Logistic Regression validation accuracy: 0.9292279411764706
```

Figure 119 Undersampling

The Logistic Regression model with L2 regularization and the 'liblinear' solver achieved a validation accuracy of approximately 92.9%. The performance is strong, indicating that Logistic Regression can handle undersampled data effectively. The choice of L2 regularization and the 'liblinear' solver is well-suited for smaller, imbalanced datasets, contributing to the model's robustness.

```
Fitting 5 folds for each of 25 candidates, totalling 125 fits
Logistic Regression best parameters: {'C': 100, 'penalty': 'l2', 'solver': 'lbfgs'}
Logistic Regression validation accuracy: 0.966027397260274
```

Figure 120 Oversampling

The Logistic Regression model with L2 regularization and the 'lbfgs' solver achieved a validation accuracy of approximately 96.6%. The high accuracy with oversampling suggests that Logistic Regression benefits significantly from a balanced dataset, making it easier to find a linear decision boundary. The 'lbfgs' solver is well-suited for handling larger, balanced datasets provided by SMOTE, contributing to improved performance.

Overall, both undersampling and oversampling configurations used L2 regularization, indicating its effectiveness in preventing overfitting across different sampling methods. The choice of **Solver Selection** ('liblinear' for undersampling and 'lbfgs' for oversampling) reflects the adaptation to dataset size and complexity. 'liblinear' is efficient for smaller datasets, while 'lbfgs' is more suitable for larger datasets created by oversampling. Logistic Regression models show improved accuracy with **oversampling**, highlighting the importance of balanced datasets for linear models.

5.2.4 Random Forest Hyperparameter

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score

rf_parameter = RandomForestClassifier(random_state=42)

param_grid_rf = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['sqrt', 'log2', None]  # Removed 'auto'
}

grid_search_rf = GridSearchCV(estimator=rf_parameter, param_grid=param_grid_rf, cv=5, n_jobs=-1, verbose=2)

# Fit the grid search
grid_search_rf.fit(X_train, y_train)

# Best parameters
best_rf = grid_search_rf.best_estimator_

# Predict on the validation set
y_val_pred_rf = best_rf.predict(X_val)

# Calculate accuracy
accuracy_rf = accuracy_score(y_val, y_val_pred_rf)
print(f"Random Forest best parameters: {grid_search_rf.best_params_}")
print(f"Random Forest validation accuracy: {accuracy_rf}")
```

Figure 121 Random Forest Hyperparameter Python Code

```
Fitting 5 folds for each of 324 candidates, totalling 1620 fits
Random Forest best parameters: {'max_depth': None, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 200}
Random Forest validation accuracy: 0.9329044117647058
```

Figure 122 Undersampling

The Random Forest model with the specified hyperparameters achieved a validation accuracy of approximately 93.3%. The use of log2 for max_features help reduce overfitting by limiting the number of features considered at each split. The high number of estimators (200) and the threshold for splitting nodes (min_samples_split = 5) contribute to the robustness and accuracy of the model.

```
Fitting 5 folds for each of 324 candidates, totalling 1620 fits
Random Forest best parameters: {'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100}
Random Forest validation accuracy: 0.9652054794520548
```

Figure 123 Oversampling

The Random Forest model with the specified hyperparameters achieved a validation accuracy of approximately 96.5%. The use of sqrt for max_features is effective in reducing overfitting, especially when combined with SMOTE. The slightly lower number of estimators (100) compared to the undersampling case shows that even with fewer trees, the model can achieve high accuracy when the data is balanced. The lower threshold for splitting nodes (min_samples_split = 2) allows for more detailed splits, contributing to high accuracy.

Overall, Random Forest model's performance varies with the sampling method and chosen hyperparameters: **Undersampling**: Achieved a high accuracy of 93.3%, showing robustness even with a reduced dataset, thanks to the comprehensive hyperparameter tuning. **Oversampling**: Achieved an even higher accuracy of 96.5%, benefiting from the balanced dataset provided by SMOTE and the carefully chosen hyperparameters.

5.2.5 Naïve Bayes Hyperparameter

```python
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import GridSearchCV

nb_parameter = MultinomialNB()

param_grid_nb = {
    'alpha': [0.1, 0.5, 1, 2, 5]
}

grid_search_nb = GridSearchCV(estimator=nb_parameter, param_grid=param_grid_nb, cv=5, n_jobs=-1, verbose=2)

# Fit the grid search
grid_search_nb.fit(X_train, y_train)

# Best parameters
best_nb = grid_search_nb.best_estimator_

# Predict on the validation set
y_val_pred_nb = best_nb.predict(X_val)

# Calculate accuracy
accuracy_nb = accuracy_score(y_val, y_val_pred_nb)
print(f"Naive Bayes best parameters: {grid_search_nb.best_params_}")
print(f"Naive Bayes validation accuracy: {accuracy_nb}")
```

Figure 124 Naïve Bayes Hyperparameter Python Code

```
Fitting 5 folds for each of 5 candidates, totalling 25 fits
Naive Bayes best parameters: {'alpha': 0.5}
Naive Bayes validation accuracy: 0.9237132352941176
```

Figure 125 Undersampling

The Naive Bayes model with an alpha value of 0.5 achieved a validation accuracy of approximately 92.4%. This high accuracy shows that the model is effective even with a reduced dataset, thanks to the smoothing parameter which prevents zero probabilities. The alpha value of 0.5 helps balance the probabilities assigned to features, making the model robust to undersampled data.

```
Fitting 5 folds for each of 5 candidates, totalling 25 fits
Naive Bayes best parameters: {'alpha': 0.1}
Naive Bayes validation accuracy: 0.9443835616438356
```

Figure 126 Oversampling

The Naive Bayes model with an alpha value of 0.1 achieved a validation accuracy of approximately 94.4%. The higher accuracy compared to the undersampled case suggests that the model benefits from the balanced dataset provided by SMOTE. The lower alpha value of 0.1 allows the model to better capture the nuances in the data, which is especially effective when the dataset is balanced through oversampling.

Overall, Naive Bayes model's performance varies depending on the sampling method and chosen hyperparameters:

- **Undersampling**: Achieved a high accuracy of 92.4%, demonstrating the model's robustness with the appropriate smoothing parameter.
- **Oversampling**: Achieved an even higher accuracy of 94.4%, benefiting from the balanced dataset provided by SMOTE and the finely tuned alpha value.

5.2.6 Results Discussion

Table 8 Comparison Between Models' Test and Hyperparameter Accuracies Based on Sampling Methods

Sampling Method	Model Name	Test Accuracy	Hyperparameter Accuracy
Undersampling (Python Script)	BERT Model	0.97	-
	SVM Model	0.94	0.927
	Logistic Regression Model	0.92	0.929
	Random Forest Model	0.91	0.932
	Naïve Bayes Model	0.93	0.923
Oversampling (SMOTE)	BERT Model	-	-
	SVM Model	0.96	0.961
	Logistic Regression Model	0.96	0.966
	Random Forest Model	0.96	0.965
	Naïve Bayes Model	0.94	0.944

The results in Table 8 indicate the performance of different models when trained using two sampling methods: undersampling via a Python script and oversampling using Synthetic

Minority Over-sampling Technique (SMOTE). The discussion of these results involves evaluating the models' test and hyperparameter accuracies, as well as considering potential reasons for the observed performance differences which is shown in Table 9 below.

Table 9 Models Comparison Explanation

NO.	UNDERSAMPLING	OVERSAMPLING
1.	**BERT Model** It achieves a test accuracy of 0.97 could be attributed to the model's ability to understand and contextualize natural language at a deep level.	**BERT Model** Its data is not available for this method as it would take more steps to prepare the data that is able to be inputted to a BERT model which is not the scope of the project, however the accuracy would be similar to its undersampling counterpart.
2.	**SVM Model** It shows a test accuracy of 0.94 and a hyperparameter accuracy of 0.927. The slight difference suggests that the model is well-optimized and performs consistently on unseen data.	**SVM Model** It improves to a test accuracy of 0.96 with a hyperparameter accuracy of 0.961, suggesting that oversampling helps the model better generalize to minority classes.
3.	**Logistic Regression Model** With a test accuracy of 0.92 and hyperparameter accuracy of 0.929, it performs slightly lower than SVM, likely due to its simpler linear nature.	**Logistic Regression Model** Both test and hyperparameter accuracies for the model rise to 0.96 and 0.966, respectively, showing that SMOTE effectively balances the data, enhancing model performance.
4.	**Random Forest Model** The model achieves a test accuracy of 0.91 and a hyperparameter accuracy of 0.932. Its performance is slightly lower, potentially due to the complexity of the model and sensitivity to the undersampled data.	**Random Forest Model** The model achieves a significant improvement with a test accuracy of 0.96 and hyperparameter accuracy of 0.965. This boost is likely due to the model's ability to leverage the additional synthetic data points.
5.	**Naïve Bayes Model** This model shows a test accuracy of 0.93 and a hyperparameter accuracy of 0.923, indicating a reliable performance for simpler models.	**Naïve Bayes Model** The model also benefits from SMOTE, with a test accuracy of 0.94 and hyperparameter accuracy of 0.944, though the gain is less pronounced than for other models.

Performance Consistency The minimal gap between test accuracy and hyperparameter accuracy across models suggests consistent performance. This consistency is crucial as it indicates that the models generalize well from training to testing phases.

Impact of Sampling Methods As seen on Table 8 above, sampling methods do lightly affect the accuracy of the models in this project. **Undersampling** reduces the number of majority class samples to balance the dataset. While effective, it may lead to loss of valuable information, potentially explaining the slightly lower performance of some models like random forest and logistic regression. On the other hand, by generating synthetic minority class samples or **Oversampling**, it helps create a more balanced dataset without losing information which generally leads to improved model performance, as observed with the notable gains in SVM, logistic regression, and random forest models. However, synthetic data might sometimes introduce noise, which needs careful handling (Pradipta et al., 2021).

Model Complexity and Nature:

- BERT Model: The high accuracy of BERT, even without hyperparameter tuning, underscores its robustness in handling text data. BERT's deep learning architecture allows it to capture complex patterns and contextual relationships (Devlin et al., 2019).
- SVM and Logistic Regression: These models benefit significantly from balanced datasets. SVM, with its margin maximization principle, and logistic regression, with its linear decision boundary, both perform better when minority class representation improves.
- Random Forest Model: This model's performance boost with SMOTE suggests that having more diverse data points helps the ensemble of trees make more accurate decisions.
- Naïve Bayes Model: As a simpler probabilistic model, Naïve Bayes shows reliable performance across both sampling methods. Its gains with SMOTE indicate improved handling of minority class predictions.

Hypothetical Reasons for Performance Variations There are many reasons that can widen the gap of accuracies between models, such as:

- Hardware Capabilities: Different models may leverage hardware resources differently. For instance, BERT's deep learning nature might benefit more from advanced GPUs, whereas simpler models like logistic regression might not see such a significant hardware-dependent performance boost.
- Parameter Sensitivity: Some models, particularly complex ones like random forests, are sensitive to hyperparameters. Effective tuning can lead to notable performance improvements.
- Sampling Method Suitability: The inherent characteristics of different models make them somewhat suitable for specific sampling methods. Models that rely heavily on data quantity and diversity (e.g., random forest) tend to perform better with SMOTE, while those that handle imbalanced data inherently well (e.g., Naïve Bayes) show less dramatic improvements (Mohammed et al., 2020).

Overall, the results suggest that while sampling methods can enhance model performance, the extent of improvement varies based on model complexity, sensitivity to data balancing, and the nature of the task at hand.

5.3 Model Deployment

After the student has trained, tested, and tuned the models to be able to reach high accuracies, the next step would be to deploy the models using Streamlit. Before doing that, the models will be saved (or dumped) using Python module called Pickle which is a process whereby a Python object hierarchy is converted into a byte stream shown in section 5.3.1 for all the models.

5.3.1 Pickles for the Model Deployment

BERT

```python
import pickle

# Save the model
model_save_path = 'C:/Users/setia/Downloads/Models/bert_model.pkl'
with open(model_save_path, 'wb') as f:
    pickle.dump(model, f)

# Save the tokenizer
tokenizer_save_path = 'C:/Users/setia/Downloads/Models/bert_tokenizer.pkl'
with open(tokenizer_save_path, 'wb') as f:
    pickle.dump(tokenizer, f)
```

Figure 127 Pickle Python Code for BERT Model Undersampling

SVM

```python
# Save the model
model_save_path = 'C:/Users/setia/Downloads/Models/svm_model_ps.pkl'
with open(model_save_path, 'wb') as f:
    pickle.dump(best_svm, f)

# Save the tokenizer
tokenizer_save_path = 'C:/Users/setia/Downloads/Models/svm_vectorizer_ps.pkl'
with open(tokenizer_save_path, 'wb') as f:
    pickle.dump(vectorizer, f)
```

Figure 128 Pickle Python Code for SVM Model Undersampling

```python
import pickle

# Save the model
model_save_path = 'C:/Users/setia/Downloads/Models/svm_model.pkl'
with open(model_save_path, 'wb') as f:
    pickle.dump(best_svm, f)

# Save the tokenizer
tokenizer_save_path = 'C:/Users/setia/Downloads/Models/svm_vectorizer.pkl'
with open(tokenizer_save_path, 'wb') as f:
    pickle.dump(vectorizer, f)
```

Figure 129 Pickle Python Code for SVM Model Oversampling

Logistic Regression

```
# Save the model
model_save_path = 'C:/Users/setia/Downloads/Models/lr_model_ps.pkl'
with open(model_save_path, 'wb') as f:
    pickle.dump(best_lr, f)

# Save the tokenizer
tokenizer_save_path = 'C:/Users/setia/Downloads/Models/lr_vectorizer_ps.pkl'
with open(tokenizer_save_path, 'wb') as f:
    pickle.dump(vectorizer, f)
```

Figure 130 Pickle Python Code for Logistic Regression Model Undersampling

```
import pickle

# Save the model
model_save_path = 'C:/Users/setia/Downloads/Models/lr_model.pkl'
with open(model_save_path, 'wb') as f:
    pickle.dump(best_lr, f)

# Save the tokenizer
tokenizer_save_path = 'C:/Users/setia/Downloads/Models/lr_vectorizer.pkl'
with open(tokenizer_save_path, 'wb') as f:
    pickle.dump(vectorizer, f)
```

Figure 131 Pickle Python Code for Logistic Regression Model Oversampling

Random Forest

```
# Save the model
model_save_path = 'C:/Users/setia/Downloads/Models/rf_model_ps.pkl'
with open(model_save_path, 'wb') as f:
    pickle.dump(best_rf, f)

# Save the tokenizer
tokenizer_save_path = 'C:/Users/setia/Downloads/Models/rf_vectorizer_ps.pkl'
with open(tokenizer_save_path, 'wb') as f:
    pickle.dump(vectorizer, f)
```

Figure 132 Pickle Python Code for Logistic Regression Model Undersampling

```
import pickle

# Save the model
model_save_path = 'C:/Users/setia/Downloads/Models/rf_model.pkl'
with open(model_save_path, 'wb') as f:
    pickle.dump(best_rf, f)

# Save the tokenizer
tokenizer_save_path = 'C:/Users/setia/Downloads/Models/rf_vectorizer.pkl'
with open(tokenizer_save_path, 'wb') as f:
    pickle.dump(vectorizer, f)
```

Figure 133 Pickle Python Code for Random Forest Model Oversampling

Naïve bayes

```
# Save the model
model_save_path = 'C:/Users/setia/Downloads/Models/nb_model_ps.pkl'
with open(model_save_path, 'wb') as f:
    pickle.dump(best_nb, f)

# Save the tokenizer
tokenizer_save_path = 'C:/Users/setia/Downloads/Models/nb_vectorizer_ps.pkl'
with open(tokenizer_save_path, 'wb') as f:
    pickle.dump(vectorizer, f)
```

Figure 134 Pickle Python Code for Naïve Bayes Model Undersampling

```
import pickle

# Save the model
model_save_path = 'C:/Users/setia/Downloads/Models/nb_model.pkl'
with open(model_save_path, 'wb') as f:
    pickle.dump(best_nb, f)

# Save the tokenizer
tokenizer_save_path = 'C:/Users/setia/Downloads/Models/nb_vectorizer.pkl'
with open(tokenizer_save_path, 'wb') as f:
    pickle.dump(vectorizer, f)
```

Figure 135 Pickle Python Code for Naïve Bayes Model Oversampling

5.3.2 Streamlit Deployment

5.3.2.1 Streamlit Codes

For the deployment of the models, the student will be using Streamlit python module. First, import all the necessary modules that are needed by the function displayed in Figure 136 below.

```
import lime
import torch
import folium
import random
import pickle
import requests
import numpy as np
import pandas as pd
import wikipediaapi
import lime.lime_text
from PIL import Image
import streamlit as st
from io import BytesIO
from datetime import datetime
from bs4 import BeautifulSoup
from wordcloud import WordCloud
import matplotlib.pyplot as plt
from collections import Counter
import matplotlib.ticker as ticker
from matplotlib.colors import to_hex
from streamlit_folium import st_folium
from geopy.geocoders import Nominatim
from googleapiclient.discovery import build
import streamlit.components.v1 as components
from sklearn.metrics.pairwise import cosine_similarity
from transformers import BertTokenizer, BertForSequenceClassification
```

Figure 136 Import Modules for Deployment

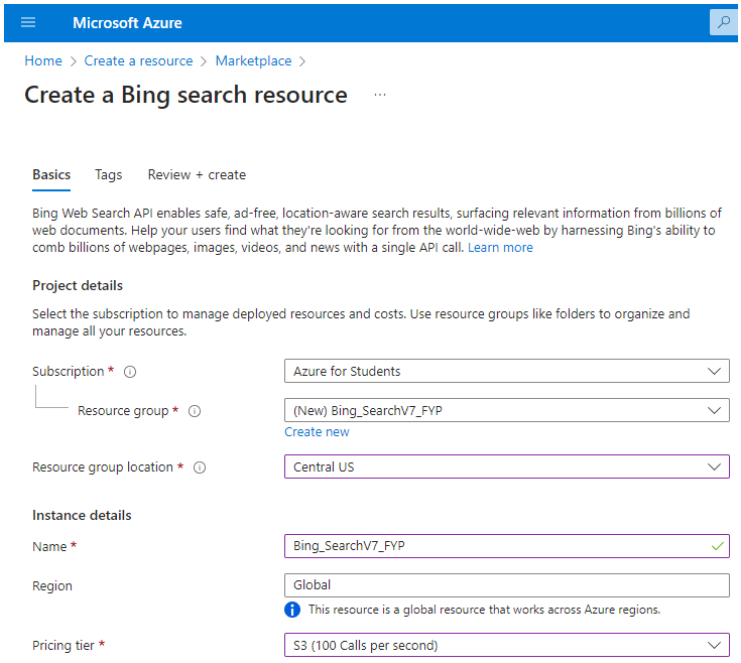


Figure 137 Create a Bing Search V7 Resource

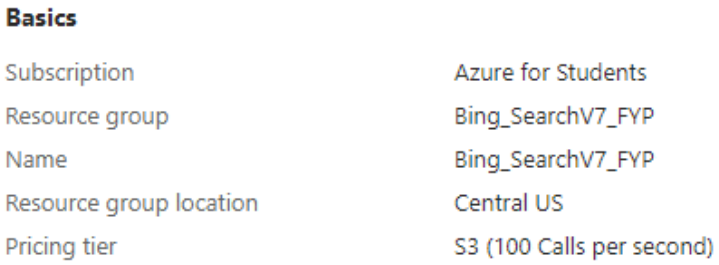


Figure 138 Bing Search V7 Resource

Figures 137 and 138 show the process to get the Bing Search API Key as the student accessed Azure for Students website and created a Bing Search V7 resources.

```
BING_API_KEY = 'dd43a7926f7f41468aefa0e9edc427b7'
BING_SEARCH_URL = 'https://api.bing.microsoft.com/v7.0/images/search'

# Set page configuration
st.set_page_config(layout="wide", page_title="Firm Reputation Classifier")

# Load the dataset
df = pd.read_csv('C:/Users/setia/Downloads/aggregatedFiltered_lemma_Dataset.csv')
reviews_df = pd.read_csv('C:/Users/setia/Downloads/glassdoor_reviews_cleaned_copy_v3.csv')
```

Figure 139 Configure API Key and Load the Datasets

Figure 139 shows the loading and importing of two (2) datasets which will be needed in the process of deployment.

```
# BERT
with open(path + 'bert_model.pkl', 'rb') as f:
    bert_model = pickle.load(f)

with open(path + 'bert_tokenizer.pkl', 'rb') as f:
    bert_tokenizer = pickle.load(f)

# Traditional models
traditional_models = {
    'SVM': {
        'SVM Oversampling': (pickle.load(open(path + 'svm_model.pkl', 'rb')), pickle.load(open(path + 'svm_vectorizer.pkl', 'rb'))),
        'SVM Undersampling': (pickle.load(open(path + 'svm_model_ps.pkl', 'rb')), pickle.load(open(path + 'svm_vectorizer_ps.pkl', 'rb')))
    },
    'Random Forest': {
        'RF Oversampling': (pickle.load(open(path + 'rf_model.pkl', 'rb')), pickle.load(open(path + 'rf_vectorizer.pkl', 'rb'))),
        'RF Undersampling': (pickle.load(open(path + 'rf_model_ps.pkl', 'rb')), pickle.load(open(path + 'rf_vectorizer_ps.pkl', 'rb')))
    },
    'Logistic Regression': {
        'LR Oversampling': (pickle.load(open(path + 'lr_model.pkl', 'rb')), pickle.load(open(path + 'lr_vectorizer.pkl', 'rb'))),
        'LR Undersampling': (pickle.load(open(path + 'lr_model_ps.pkl', 'rb')), pickle.load(open(path + 'lr_vectorizer_ps.pkl', 'rb')))
    },
    'Naive Bayes': {
        'NB Oversampling': (pickle.load(open(path + 'nb_model.pkl', 'rb')), pickle.load(open(path + 'nb_vectorizer.pkl', 'rb'))),
        'NB Undersampling': (pickle.load(open(path + 'nb_model_ps.pkl', 'rb')), pickle.load(open(path + 'nb_vectorizer_ps.pkl', 'rb')))
    }
}
```

Figure 140 Load and Initialize the Datasets

Figure 140 shows the loading and initialization of the models by using Pickle Python module.

```
# Function to predict sentiment
def predict_sentiment(texts, model, tokenizer):
    if isinstance(model, BertForSequenceClassification):
        inputs = tokenizer(texts, return_tensors="pt", truncation=True, padding=True, max_length=128)
        with torch.no_grad():
            outputs = model(**inputs)
            logits = outputs.logits
            probabilities = torch.softmax(logits, dim=1)
            predicted_labels = torch.argmax(probabilities, dim=1)
        return predicted_labels.cpu().numpy(), probabilities.cpu().numpy()
    else:
        texts = tokenizer.transform(texts)
        predicted_labels = model.predict(texts)
        probabilities = model.predict_proba(texts)
        return predicted_labels, probabilities
```

Figure 141 Predict Sentiment Function

```
# Function to get Wikipedia summary
def get_wikipedia_summary(company_name):
    wiki_wiki = wikipediaapi.Wikipedia(
        language='en',
        extract_format=wikipediaapi.ExtractFormat.WIKI,
        user_agent=f'Firm Reputation Classifier/1.0 (https://www.google.com/search?q={company_name})'
    )
    page = wiki_wiki.page(company_name)
    if page.exists():
        # Split the summary into paragraphs and return the first one
        paragraphs = page.summary.split('\n')
        return paragraphs[0] if paragraphs else "No summary available."
    else:
        return "No Wikipedia page found for this company."
```

Figure 142 Display Wikipedia Summary Function

```python
# Function to get Company Image
def get_bing_image_url(query, count=10):
    headers = {"Ocp-Apim-Subscription-Key": BING_API_KEY}
    params = {"q": query, "count": count}
    response = requests.get(BING_SEARCH_URL, headers=headers, params=params)
    search_results = response.json()
    if 'value' in search_results:
        return [img['contentUrl'] for img in search_results['value']]
    else:
        return None
```

Figure 143 Get Company Image Function

```python
# Function to fetch and resize image
def fetch_and_resize_image(image_url, width=400, height=300):
    try:
        response = requests.get(image_url)
        img = Image.open(BytesIO(response.content))
        img = img.resize((width, height), Image.Resampling.LANCZOS)
        return img
    except Exception as e:
        return None
```

Figure 144 Fetch and Resize Company Image

```python
# Function to Color Bar Graph
def color_map(weight):
    """Generate a color based on the weight value."""
    if weight > 0:
        return to_hex([1, 0.5 + weight / 2, 0.1])  # Dynamic orange to yellow
    else:
        # Gradient from dark blue to light blue
        return to_hex([0, 0.5 + weight / 2, 1])  # Dynamic dark blue to light blue
```

Figure 145 Color Bar Graph Function

```python
# Function to Plot Weight for Bar Graph
def plot_weight_bar(weight, color):
    """Generate a bar plot for the weight."""
    fig, ax = plt.subplots(figsize=(3, 0.5))
    ax.barh([''], [weight], color=color)
    ax.set_xlim([-1, 1])
    ax.axis('off')
    return fig
```

Figure 146 Plot Weight for Bar Graph Function

Figures 141 to 146 show the functions that will be called in other functions.

```
# Streamlit interface
st.title("Firm Reputation Classifier")

# Dropdown for firm selection
firm_names = df['firm'].unique()
selected_firms = st.multiselect("Select the firm names:", firm_names)

# Dropdown for model selection
model_type = st.selectbox("Select the type of model:", ['BERT'] + list(traditional_models.keys()))

if model_type == 'BERT':
    model, tokenizer = bert_model, bert_tokenizer
    model_selection_placeholder = None
else:
    model_files = list(traditional_models[model_type].keys())
    selected_model_file = st.selectbox("Select the file for the traditional model:", model_files)
    model, tokenizer = traditional_models[model_type][selected_model_file]
```

Figure 147 Select Firm and Model Function

```
# Use sidebar for vertical tabs
tab = st.sidebar.radio("Select Tab:", ["Company Analysis", "Sentiment Trends & Comparative Analysis", "Review Display",
                                       "Job Titles & Employment Status", "Recommendations, Outlook & CEO Approval",
                                       "Correlation Analysis", "Quick Recommendation System", "Keyword Co-Occurrence","Keyword-Based Recommendation"])

# Create a dictionary to store results for each firm
if 'results' not in st.session_state:
    st.session_state.results = {}
```

Figure 148 Initialize a Sidebar for Vertical Tabs

Figure 147 shows the function for the user to select companies and model they would like to use for prediction. Meanwhile, Figure 148 shows the tabs where the user can select which analysis they want to do and see.

```
# Fetch and update results
if selected_firms:
    for selected_firm in selected_firms:
        if selected_firm not in st.session_state.results:
            # Fetch company images
            image_urls = get_bing_image_url(selected_firm)
            images = []
            if image_urls:
                with st.spinner(f"Fetching image for {selected_firm}..."):
                    for image_url in image_urls:
                        img = fetch_and_resize_image(image_url, width=400, height=300)
                        if img:
                            images.append(img)
                            break
            st.session_state.results[selected_firm] = {
                'images': images,
                'summary': get_wikipedia_summary(selected_firm),
                'reviews': df[df['firm'] == selected_firm]['labeled_keywords'].tolist(),
                'sentiment': None,
                'lime_explanation': None
            }
```

Figure 149 Fetch and Update Results Function

Figure 149 shows the function to fetch all the processed results and store them for future uses if a user wishes to go back to a page that has been opened and executed without waiting again.

```
# Date range filter
available_dates = pd.to_datetime(reviews_df['date_review']).dt.date.unique()
available_dates = sorted(available_dates)

col1, col2 = st.columns(2)
with col1:
    start_date = st.selectbox("Select start date:", available_dates, index=0)
with col2:
    end_date = st.selectbox("Select end date:", available_dates, index=len(available_dates) - 1)

if start_date and end_date:
    filtered_reviews_df = reviews_df[(pd.to_datetime(reviews_df['date_review']).dt.date >= start_date) & (pd.to_datetime(reviews_df['date_review']).dt.date <= end_date)]
```

Figure 150 Date Range Filter

```
if st.button("Predict"):
    for selected_firm in selected_firms:
        firm_reviews = st.session_state.results[selected_firm]['reviews']

        if len(firm_reviews) == 0:
            st.write(f"No reviews found for {selected_firm}.")
        else:
            predicted_labels, probabilities = predict_sentiment(firm_reviews, model, tokenizer)
            avg_sentiment = np.mean(predicted_labels)
            sentiment = 'good' if avg_sentiment > 0.5 else 'bad'
            st.session_state.results[selected_firm]['sentiment'] = sentiment

            # LIME explanation
            explainer = lime.lime_text.LimeTextExplainer(class_names=['bad', 'good'])
            exp = explainer.explain_instance(firm_reviews[0], lambda x: predict_sentiment(x, model, tokenizer)[1], num_features=15)
            st.session_state.results[selected_firm]['lime_explanation'] = exp
```

Figure 151 Predict Button for LIME Function

After selecting the date range filter as shown in Figure 150, the user will need to press the “Predict” button to start the LIME function.

```
if tab == "Company Analysis":
    for selected_firm in selected_firms:
        with st.expander(f"{selected_firm} Analysis", expanded=True):
            # Display Wikipedia summary
            st.subheader("Company Summary")
            st.write(st.session_state.results[selected_firm]['summary'])

            # Display images
            if st.session_state.results[selected_firm]['images']:
                st.subheader("Company Images")
                for img in st.session_state.results[selected_firm]['images']:
                    st.image(img)
            else:
                st.write("No images found for this company.")

            # Display Sentiment
            if st.session_state.results[selected_firm]['sentiment']:
                st.subheader("Predicted Sentiment")
                st.write(f"The predicted sentiment for {selected_firm} is: {st.session_state.results[selected_firm]['sentiment']}")

            # Display LIME Explanation
            if st.session_state.results[selected_firm]['lime_explanation']:
                st.subheader("LIME Explanation")
                exp = st.session_state.results[selected_firm]['lime_explanation']

                # Display textual explanation
                components.html(exp.as_html())

                # Extract and visualize feature weights
                explanation = exp.as_list()
                words, weights = zip(*explanation)

                # Calculate min and max weights
                min_weight = min(weights)
                max_weight = max(weights)

                ## Plot the weights
                fig, ax = plt.subplots()
                bars = ax.barh(words, weights, color=[color_map(w) for w in weights])
                ax.set_xlabel('Weight')
                ax.set_title('LIME Explanation')
                plt.gca().invert_yaxis()
```

Figure 152 Company Analysis Function

```
col1,col2 =st.columns(2)
# Display min and max weights
with col1:
    for bar in bars:
        width = bar.get_width()
        ax.text(width, bar.get_y() + bar.get_height() / 3, f'{width:.4f}', va='center', ha='right' if width < 0 else 'left')

    # Display min and max weights
    plt.text(0, len(words)-1.5, f'Min: {min_weight:.4f}', color='black')
    plt.text(0, len(words)-1.0, f'Max: {max_weight:.4f}', color='black')

    st.pyplot(fig)

with col2:
    ## Create a dataframe for the table
    data = {
        'Feature': words,
        'Weight': weights
    }
    df = pd.DataFrame(data)

    # Display the table
    st.write(df[['Feature', 'Weight']])

# Display Keyword Frequency
st.subheader("Keyword Frequency")
all_keywords = ' '.join(st.session_state.results[selected_firm]['reviews']).split()
keyword_counts = Counter(all_keywords)
top_keywords = keyword_counts.most_common(10)
keyword_df = pd.DataFrame(top_keywords, columns=['Keyword', 'Frequency'])
st.table(keyword_df)
```

Figure 153 Company Analysis Function (Cont'd)

```
# Firm Rankings
st.subheader("Firm Rankings")

# Calculate average ratings for each firm
avg_ratings = reviews_df.groupby('firm')['overall_rating'].mean().reset_index()
avg_ratings.columns = ['Firm', 'Average Rating']

# Add a ranking column
avg_ratings = avg_ratings.sort_values(by='Average Rating', ascending=False).reset_index(drop=True)
avg_ratings['Ranking'] = avg_ratings.index + 1
avg_ratings = avg_ratings[['Firm', 'Average Rating']]

# Highlight selected firms
def highlight_selected_firms(df, selected_firms):···

# Apply the highlighting function
highlighted_avg_ratings = avg_ratings.style.apply(highlight_selected_firms, selected_firms=selected_firms, axis=None)

# Display the dataframe with highlighting
st.dataframe(highlighted_avg_ratings, use_container_width=True)

st.markdown("""
<div style="text-align: center; color: rgba(0, 0, 0, 0.5);">
    Analyze individual companies by examining their reviews, extracting insights from sentiment scores, and identifying common themes or issues.
        This tab provides a deep dive into each company's performance and user feedback.
</div>
""", unsafe_allow_html=True)
```

Figure 154 Company Analysis Function (Cont'd)

Figures 152 to 154 show the function of Company Analysis that will show the introduction of the selected companies.

```
elif tab == "Review Display":
    for selected_firm in selected_firms:
        with st.expander(f"{selected_firm} Reviews", expanded=True):
            st.subheader("Review Rating Statistics")
            firm_reviews = reviews_df[reviews_df['firm'] == selected_firm]
            if not firm_reviews.empty:
                total_reviews = len(firm_reviews)
                rating_distribution = firm_reviews['overall_rating'].value_counts().sort_index()
                st.write(f"Total number of reviews: {total_reviews}")
                rating_df = pd.DataFrame({'Rating': rating_distribution.index, 'Number of Reviews': rating_distribution.values})
                st.write("Number of reviews per rating:")
                st.table(rating_df)

                st.write("Reviews by Rating:")
                # Create a container to hold the reviews
                review_container = st.container()

                # Loop through each rating and display the reviews
                for rating in range(1, 6):
                    with review_container:
                        st.subheader(f"Rating: {rating}")
                        rating_reviews = firm_reviews[firm_reviews['overall_rating'] == rating]
                        if not rating_reviews.empty:
                            st.write(f"Number of reviews with rating {rating}: {len(rating_reviews)}")
                            st.dataframe(rating_reviews[['headline', 'overall_rating', 'pros', 'cons']].head(5))
                        else:
                            st.write(f"No reviews found for rating {rating}.")

    st.markdown("""
    <div style="text-align: center; color: rgba(0, 0, 0, 0.5);">
        View and interact with raw reviews submitted by users.
            This tab allows you to browse through the actual comments and feedback provided for detailed examination.
    </div>
    """, unsafe_allow_html=True)
```

Figure 155 Review Display Function

```
elif tab == "Sentiment Trends & Comparative Analysis":
    if selected_firms:
        for selected_firm in selected_firms:
            st.subheader(f"Sentiment Trends for {selected_firm}")
            sentiment_trends_df = filtered_reviews_df[filtered_reviews_df['firm'] == selected_firm].groupby(pd.to_datetime(filtered_reviews_df['date_review']).dt.date)['overall_rating'].mean().reset_index()
            sentiment_trends_df.columns = ['Date', 'Average Rating']
            st.line_chart(sentiment_trends_df.set_index('Date'))

        feature_columns = ['senior_mgmt', 'comp_benefits', 'work_life_balance', 'culture_values', 'diversity_inclusion', 'career_opp']
        comparative_df = df[df['firm'].isin(selected_firms)]
        comparative_df = comparative_df.groupby('firm')[feature_columns].mean().reset_index()
        comparative_df['Overall Score'] = comparative_df[feature_columns].mean(axis=1)
        comparative_df = comparative_df.sort_values(by='Overall Score', ascending=False)
        st.subheader("Comparative Analysis")
        st.write("Comparative Analysis of Firms:")
        st.dataframe(comparative_df, use_container_width=True)

    st.markdown("""
    <div style="text-align: center; color: rgba(0, 0, 0, 0.5);">
        Explore trends in sentiment over time and compare different companies based on their sentiment scores.
            This tab helps in understanding how sentiment evolves and how companies stack up against each other.
    </div>
    """, unsafe_allow_html=True)
```

Figure 156 Sentiment Trends & Comparative Analysis Function

Figure 155 shows the function to display all the reviews from each rating of the selected companies which enables the user to gain knowledge on what kinds of reviews exist in the system. Meanwhile, Figure 156 shows the function to display the sentiment trends and comparative analysis between selected companies. On this dashboard, the user can compare companies based on their sentiment trends.

```
elif tab == "Job Titles & Employment Status":
    st.subheader("Job Titles and Employment Status")
    for selected_firm in selected_firms:
        with st.expander(f"{selected_firm} Job Titles and Employment Status", expanded=True):
            firm_reviews = reviews_df[reviews_df['firm'] == selected_firm]
            if not firm_reviews.empty:
                col1, col2 = st.columns(2)
                with col1:
                    # Display job title frequency
                    job_title_counts = firm_reviews['job_title'].value_counts()
                    job_title_df = pd.DataFrame({'Job Title': job_title_counts.index, 'Frequency': job_title_counts.values})
                    st.write("Job Title Frequency:")
                    st.table(job_title_df.head(10))  # Show top 10 job titles

                with col2:
                    # Plot job title distribution
                    st.write("Job Title Distribution:")
                    fig, ax = plt.subplots(figsize=(5, 3))
                    job_title_counts.head(10).plot(kind='bar', ax=ax)
                    ax.set_xlabel("Job Title")
                    ax.set_ylabel("Frequency")
                    ax.set_title(f"Top 10 Job Titles in {selected_firm}")
                    ax.set_xticklabels(ax.get_xticklabels(), rotation=45, ha='right')
                    ax.yaxis.set_major_locator(ticker.MaxNLocator(integer=True))

                    # Set y-axis limits (example limits, adjust based on your data)
                    ax.set_ylim(0, job_title_counts.max() + 2)  # Add some padding
                    st.pyplot(fig)

                col3, col4 = st.columns(2)
                with col3:
                    # Display employment status frequency
                    employment_status_counts = firm_reviews['current'].value_counts()
                    employment_status_df = pd.DataFrame({'Employment Status': employment_status_counts.index,
                                                         'Frequency': employment_status_counts.values})
                    st.write("Employment Status Frequency:")
                    st.table(employment_status_df)
```

Figure 157 Job Titles & Employment Status Function

```
                with col4:
                    # Plot employment status distribution
                    st.write("Employment Status Distribution:")
                    fig, ax = plt.subplots(figsize=(5, 3))
                    employment_status_counts.plot(kind='bar', ax=ax)
                    ax.set_xlabel("Employment Status")
                    ax.set_ylabel("Frequency")
                    ax.set_title(f"Employment Status in {selected_firm}")
                    ax.set_xticklabels(ax.get_xticklabels(), rotation=45, ha='right')
                    ax.yaxis.set_major_locator(ticker.MaxNLocator(integer=True))

                    # Set y-axis limits (example limits, adjust based on your data)
                    ax.set_ylim(0, employment_status_counts.max() + 2)  # Add some padding

                    st.pyplot(fig)
            else:
                st.write(f"No reviews found for {selected_firm}.")

    st.markdown("""
    <div style="text-align: center; color: rgba(0, 0, 0, 0.5);">
        Analyze reviews based on job titles and employment status.
        This tab helps identify if there are significant differences in sentiment or feedback
        based on employee roles or their status within the company.
    </div>
    """, unsafe_allow_html=True)
```

Figure 158 Job Titles & Employment Status Function (Cont'd)

Figures 157 and 158 show the function to display all the job titles and employment status of the reviewers which enables the user to know what job roles and status of employment of the reviewers. This could give estimation and expectation of the job role they wanted based on the selected company.

```
elif tab == "Recommendations, Outlook & CEO Approval":
    for selected_firm in selected_firms:
        with st.expander(f"Displaying recommendations, outlook, and CEO approval for {selected_firm}...", expanded=True):
            with st.spinner(f"Loading data for {selected_firm}..."):
                firm_reviews_df = filtered_reviews_df[filtered_reviews_df['firm'] == selected_firm]

                if not firm_reviews_df.empty:
                    recommendations = firm_reviews_df['recommend'].replace({0: 'no', 1: 'yes'}).value_counts()
                    outlook = firm_reviews_df['outlook'].replace({0: 'no', 1: 'yes'}).value_counts()
                    ceo_approval = firm_reviews_df['ceo_approv'].replace({0: 'no', 1: 'yes'}).value_counts()

                    col1, col2, col3 = st.columns(3)
                    with col1:
                        st.write("### Recommendations")
                        st.write(recommendations.to_frame().reset_index().rename(columns={'index': 'Recommend', 'recommend': 'Count'}))
                    with col2:
                        st.write("### Outlook")
                        st.write(outlook.to_frame().reset_index().rename(columns={'index': 'Outlook', 'outlook': 'Count'}))
                    with col3:
                        st.write("### CEO Approval")
                        st.write(ceo_approval.to_frame().reset_index().rename(columns={'index': 'CEO Approval', 'ceo_approv': 'Count'}))

                    fig, ax = plt.subplots(1, 3, figsize=(15, 4))
```

Figure 159 Recommendations, Outlook & CEO Approval Function

```
                    ax[0].bar(recommendations.index, recommendations.values)
                    ax[0].set_title("Recommendations")
                    ax[0].set_xlabel("Recommend")
                    ax[0].set_ylabel("Count")

                    ax[1].bar(outlook.index, outlook.values)
                    ax[1].set_title("Outlook")
                    ax[1].set_xlabel("Outlook")
                    ax[1].set_ylabel("Count")

                    ax[2].bar(ceo_approval.index, ceo_approval.values)
                    ax[2].set_title("CEO Approval")
                    ax[2].set_xlabel("CEO Approval")
                    ax[2].set_ylabel("Count")

                    st.pyplot(fig)
                else:
                    st.write(f"No data available for {selected_firm} within the selected date range.")

    st.markdown("""
    <div style="text-align: center; color: rgba(0, 0, 0, 0.5);">
        Generate actionable recommendations based on review analysis, explore overall outlook on the company, and evaluate CEO approval ratings.
            This tab offers strategic insights and improvement areas.
    </div>
    """, unsafe_allow_html=True)
```

Figure 160 Recommendations, Outlook & CEO Approval Function (Cont'd)

Figures 159 to 160 show the function to display the recommendation, outlook, and CEO approval given by the users. Previously, these three columns had four (4) classes with neutral and no opinion, however the student would like to build a model and system that enables the users to concisely pick a company and find whether that company is good or bad directly.

```
elif tab == "Correlation Analysis":
    if selected_firms:
        st.write("Correlation Analysis")

        # Example of encoding features and calculating correlations
        feature_columns = ['work_life_balance', 'culture_values', 'diversity_inclusion', 'career_opp', 'comp_benefits', 'senior_mgmt']
        feature_df = df[df['firm'].isin(selected_firms)][feature_columns]

        # Replace NaN values with 0 for correlation calculation
        feature_df.fillna(0, inplace=True)

        # Calculate correlations
        correlations = feature_df.corr()

        st.write("Correlation Matrix:")
        st.dataframe(correlations, use_container_width=True)

        col1, col2,col3= st.columns(3)
        with col2:
            # Plot Correlation Heatmap
            fig, ax = plt.subplots(figsize=(10, 5))
            cax = ax.matshow(correlations, cmap='coolwarm', vmin=-1, vmax=1)
            fig.colorbar(cax)
            ax.set_xticks(range(len(correlations.columns)))
            ax.set_yticks(range(len(correlations.columns)))
            ax.set_xticklabels(correlations.columns, rotation=45, ha='left')
            ax.set_yticklabels(correlations.columns)
            ax.set_title("Correlation Heatmap")
            st.pyplot(fig)

    st.markdown("""
    <div style="text-align: center; color: rgba(0, 0, 0, 0.5);">
        Investigate correlations between various factors in the reviews, such as sentiment and keyword frequency.
                This tab helps in identifying relationships and patterns that could impact company performance.
    </div>
    """, unsafe_allow_html=True)
```

Figure 161 Correlation Analysis Function

Figure 161 shows a function to display a correlation heatmap between companies' features like senior management, company benefits and so on. This enables the user to see whether companies have those features (high level of job satisfaction) in mind or as something not worthy of importance.

```
# Recommendation System Tab
if tab == "Quick Recommendation System":
    st.title("Recommendation System")

    # User-based Recommendations
    st.subheader("User Recommendations")

    # Allow users to select criteria for recommendations
    recommendation_criteria = st.selectbox("Select criteria for recommendations:", ["Job Titles", "Sentiment", "Overall Ratings"])

    if recommendation_criteria == "Job Titles":
        st.write("Based on job titles:")
        for selected_firm in selected_firms:
            firm_reviews = reviews_df[reviews_df['firm'] == selected_firm]
            if not firm_reviews.empty:
                job_title_counts = firm_reviews['job_title'].value_counts()
                top_job_titles = job_title_counts.head(5)
                st.write(f"Top job titles for {selected_firm}:")
                st.write(top_job_titles)
            else:
                st.write(f"No reviews found for {selected_firm}.")

    elif recommendation_criteria == "Sentiment":
        st.write("Based on sentiment analysis:")
        for selected_firm in selected_firms:
            firm_reviews = st.session_state.results[selected_firm]['reviews']
            if len(firm_reviews) == 0:
                st.write(f"No reviews found for {selected_firm}.")
            else:
                predicted_labels, _ = predict_sentiment(firm_reviews, model, tokenizer)
                avg_sentiment = np.mean(predicted_labels)
                sentiment = 'good' if avg_sentiment > 0.5 else 'bad'
                st.write(f"The average sentiment for {selected_firm} is: {sentiment}")

    elif recommendation_criteria == "Overall Ratings":
        st.write("Based on overall ratings:")
        for selected_firm in selected_firms:
            firm_reviews = reviews_df[reviews_df['firm'] == selected_firm]
            if not firm_reviews.empty:
                avg_rating = firm_reviews['overall_rating'].mean()
                st.write(f"The average rating for {selected_firm} is: {avg_rating:.2f}")
```

Figure 162 Quick Recommendation System Function

```
# HR-based Recommendations
st.subheader("HR Recommendations")

# Provide actionable insights for HR
hr_insights = st.selectbox("Select HR Insight:", ["Improve Workplace Culture", "Enhance Benefits", "Increase Work-Life Balance"])

if hr_insights == "Improve Workplace Culture":
    st.write("Recommendations for improving workplace culture:")
    for selected_firm in selected_firms:
        firm_reviews = reviews_df[reviews_df['firm'] == selected_firm]
        if not firm_reviews.empty:
            culture_feedback = firm_reviews['culture_values'].value_counts()
            st.write(f"Culture feedback for {selected_firm}:")
            st.write(culture_feedback)

elif hr_insights == "Enhance Benefits":
    st.write("Recommendations for enhancing benefits:")
    for selected_firm in selected_firms:
        firm_reviews = reviews_df[reviews_df['firm'] == selected_firm]
        if not firm_reviews.empty:
            benefits_feedback = firm_reviews['comp_benefits'].value_counts()
            st.write(f"Benefits feedback for {selected_firm}:")
            st.write(benefits_feedback)

elif hr_insights == "Increase Work-Life Balance":
    st.write("Recommendations for improving work-life balance:")
    for selected_firm in selected_firms:
        firm_reviews = reviews_df[reviews_df['firm'] == selected_firm]
        if not firm_reviews.empty:
            work_life_balance_feedback = firm_reviews['work_life_balance'].value_counts()
            st.write(f"Work-life balance feedback for {selected_firm}:")
            st.write(work_life_balance_feedback)

st.markdown("""
<div style="text-align: center; color: rgba(0, 0, 0, 0.5);">
    Investigate correlations between various factors in the reviews, such as sentiment and keyword frequency.
        This tab helps in identifying relationships and patterns that could impact company performance.
</div>
""", unsafe_allow_html=True)
```

Figure 163 Quick Recommendation System Function (Cont'd)

```
if tab == "Keyword-Based Recommendation":
    st.subheader("Keyword-Based Recommendations")
    if selected_firms:
        for selected_firm in selected_firms:
            with st.expander(f"Recommendations for {selected_firm} based on Keywords", expanded=True):
                firm_reviews = df[df['firm'] == selected_firm]

                if not firm_reviews.empty:
                    # Concatenate all labeled_keywords
                    all_keywords = ' '.join(firm_reviews['labeled_keywords'].astype(str))

                    # Count the frequency of each keyword
                    keyword_counts = Counter(all_keywords.split())

                    # Calculate the average frequency
                    avg_frequency = np.mean(list(keyword_counts.values()))

                    # Define the scaling factor
                    scaling_factor = 1.5

                    # Calculate the frequency threshold
                    frequency_threshold = scaling_factor * avg_frequency

                    # Sort by frequency
                    sorted_keywords = sorted(keyword_counts.items(), key=lambda x: x[1], reverse=True)

                    # Convert to DataFrame
                    keyword_df = pd.DataFrame(sorted_keywords, columns=['Keyword', 'Frequency'])

                    col1, col2 = st.columns(2)
                    with col1:
                        # Display the top keywords
                        st.write("Top Keywords:")
                        st.table(keyword_df.head(10))

                    with col2:
                        if selected_firm in st.session_state.results and 'lime_explanation' in st.session_state.results[selected_firm]:
                            exp = st.session_state.results[selected_firm]['lime_explanation']
                            explanation = exp.as_list()

                            # Directly display top 15 LIME weights
                            st.write("Top 15 LIME Weights:")
                            lime_df = pd.DataFrame(explanation, columns=['Keyword', 'Weight'])
                            st.table(lime_df)

                    # Generate recommendations based on keyword frequency and sentiment
                    st.write("Recommendations Based on Keywords:")

                    recommendations = []
```

Figure 164 Keyword-Based Recommendation Function

```
                for keyword, count in sorted_keywords:
                    if count > frequency_threshold:
                        weight = lime_df[lime_df['Keyword'] == keyword]['Weight'].values[0] if keyword in lime_df['Keyword'].values else 0
                        recommendations.append(f"Keyword '{keyword}' appears frequently ({count} times) with a LIME weight of {weight:.2f}. Consider focusing on this aspect.")

                # Sentiment-based recommendations
                sentiment_df = df[['labeled_keywords', 'predicted_sentiment']].copy()
                sentiment_df['labeled_keywords'] = sentiment_df['labeled_keywords'].apply(lambda x: x if isinstance(x, list) else x.split())
                sentiment_df = sentiment_df.explode('labeled_keywords')
                sentiment_counts = sentiment_df.groupby('labeled_keywords')['predicted_sentiment'].mean()

                st.write("Sentiment-Based Recommendations:")

                for keyword, sentiment in sentiment_counts.items():
                    if sentiment < 0:  # Assuming negative sentiment indicates a problem area
                        weight = lime_df[lime_df['Keyword'] == keyword]['Weight'].values[0] if keyword in lime_df['Keyword'].values else 0
                        recommendations.append(f"Improve aspects related to '{keyword}' with a negative sentiment score ({sentiment:.2f}) and a LIME weight of {weight:.2f}.")

                # Display recommendations
                if recommendations:
                    for rec in recommendations:
                        st.write(f"- {rec}")
                else:
                    st.write("No specific recommendations based on keyword analysis.")
            else:
                st.write(f"No keywords found for {selected_firm}.")

st.markdown("""
<div style="text-align: center; color: rgba(0, 0, 0, 0.5);">
    Generate recommendations based on the analysis of frequently occurring keywords in reviews.
            This tab uses keyword frequency and sentiment analysis to suggest focus areas for improvement.
</div>
""", unsafe_allow_html=True)
```

Figure 165 Keyword-Based Recommendation Function (Cont'd)

```
if tab == "Keyword Co-Occurrence":
    for selected_firm in selected_firms:
        with st.expander(f"{selected_firm} Keyword Co-Occurrence Analysis", expanded=True):
            firm_reviews = df[df['firm'] == selected_firm]['labeled_keywords'].tolist()

            # Extract keywords
            all_keywords = set()
            for review in firm_reviews:
                all_keywords.update(review.split())

            # Dropdown for keyword selection
            selected_keywords = st.multiselect("Select keywords for co-occurrence analysis:", list(all_keywords))

            if selected_keywords:
                st.write("Co-Occurrence Recommendations:")

                # Create co-occurrence pairs
                co_occurrence_df = pd.DataFrame({'keywords': firm_reviews})
                co_occurrence_df['keywords'] = co_occurrence_df['keywords'].apply(lambda x: x.split())
                keyword_pairs = []

                for keywords in co_occurrence_df['keywords']:
                    for i in range(len(keywords)):
                        for j in range(i + 1, len(keywords)):
                            if keywords[i] in selected_keywords and keywords[j] in selected_keywords:
                                keyword_pairs.append((keywords[i], keywords[j]))

                pair_counts = Counter(keyword_pairs)

                # Calculate average frequency
                average_frequency = sum(pair_counts.values()) / len(pair_counts)

                # Set frequency threshold
                multiplier = 1.5  # Choose a multiplier based on your needs
                frequency_threshold = average_frequency * multiplier

                recommendations = []
                for (kw1, kw2), count in pair_counts.items():
                    if count > frequency_threshold:
                        recommendations.append(f"Keywords '{kw1}' and '{kw2}' frequently appear together. Try other keywords...")
                    else:
                        recommendations.append(f"Keywords '{kw1}' and '{kw2}' do not frequently appear together. Consider other keywords...")

                st.write("\n".join(recommendations))

    st.markdown("""
    <div style="text-align: center; color: rgba(0, 0, 0, 0.5);">
        Analyze how keywords co-occur in reviews to uncover underlying themes or issues.
            This tab helps in understanding which keywords frequently appear together and their significance.
    </div>
    """, unsafe_allow_html=True)
```

Figure 166 Keyword Co-Occurrence Function

Figures 162 to 165 show multiple kinds of recommendation systems which is one of the aims of this project. The student wishes for users to be able to gain recommendations automatically without spending too much time. This can be enhanced by using co-occurrence tab to display what kind of keywords the user may like to be in the selected company.

5.3.2.2 Streamlit Display Tabs

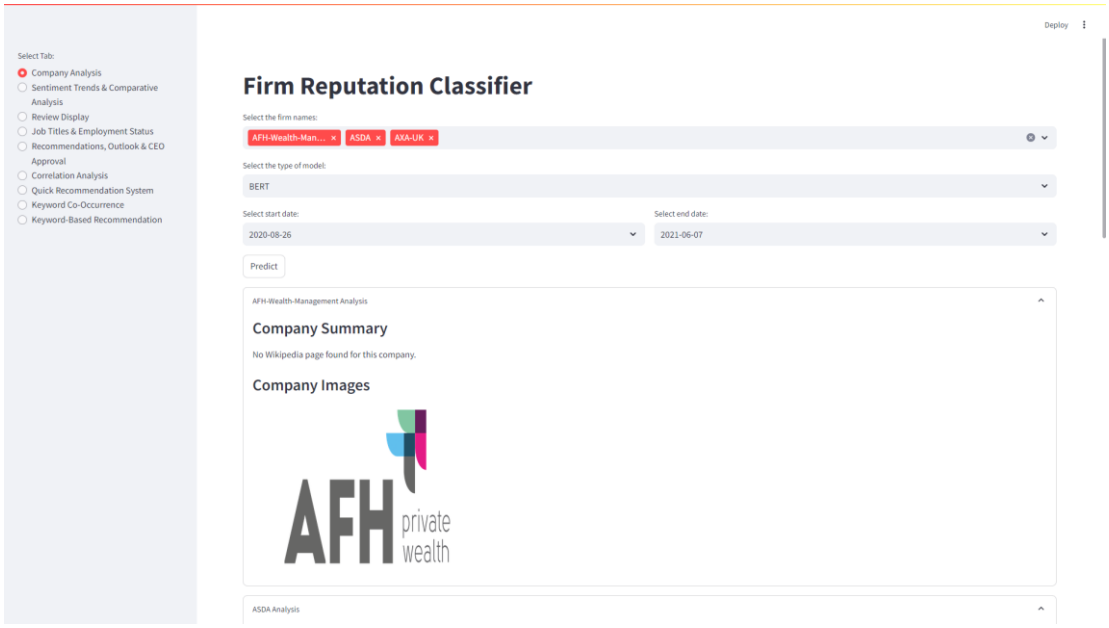


Figure 167 Company Analysis Tab Display

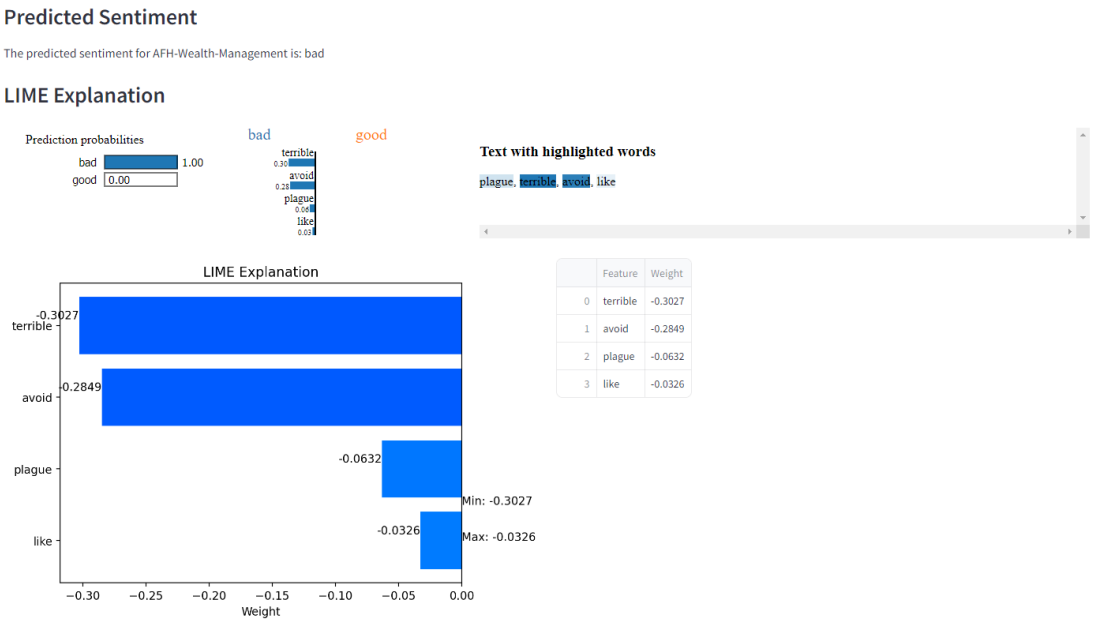


Figure 168 Company Analysis Tab Display (Cont'd)

Figures 167 and 168 show the company summary and images, as well as the LIME explanation with the weights. In this tab, the user can see what keywords are being highlighted that is being recognized by the model and the weights number being given to the keywords.

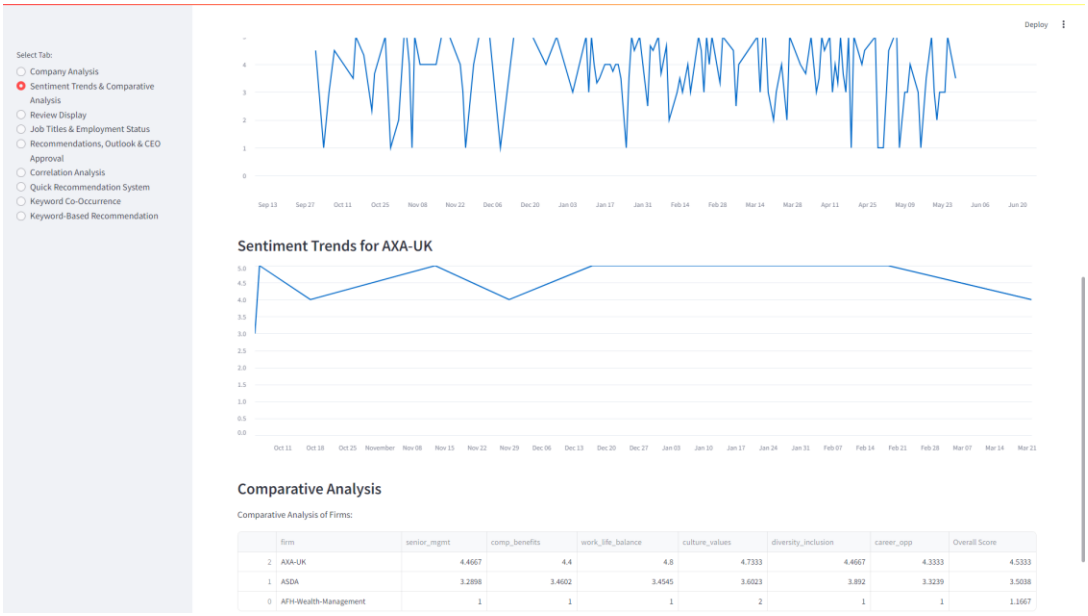


Figure 169 Sentiment Trends & Comparative Analysis Tab Display

Figure 169 shows the temporal and comparative analysis through graphs and a table. In this tab, the user can see the trend of the ratings within the selected range of date. At the same time, they can also compare the selected companies' other variables, like company benefits, senior management, and others.

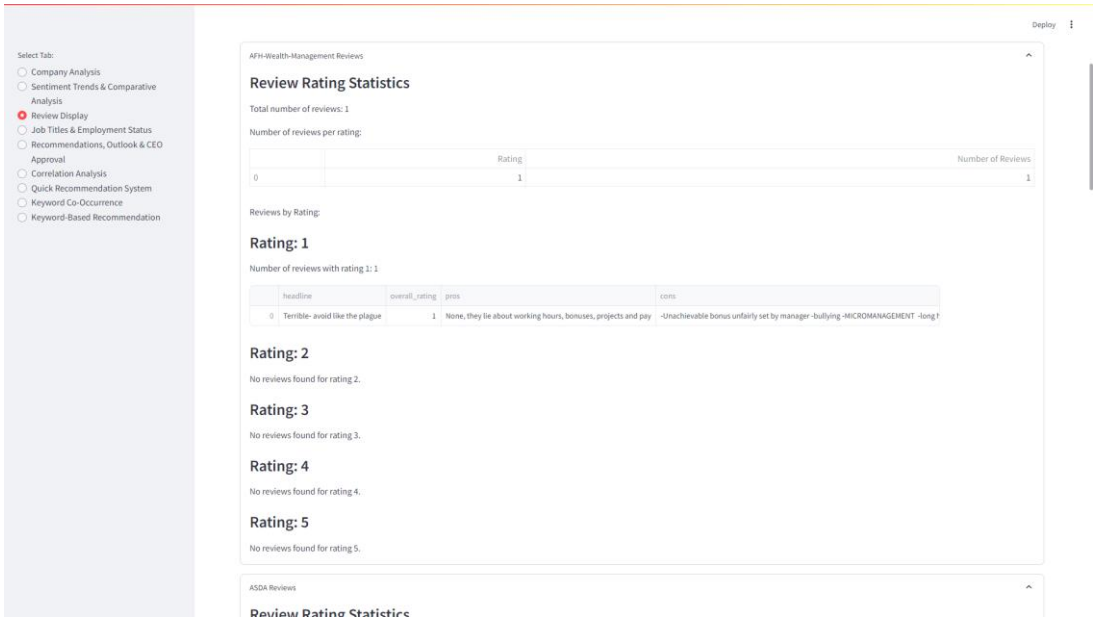


Figure 170 Review Display Tab

Figure 170 shows all the ratings of the selected firms based on its ratings. In this tab, the user can see the pros and cons of the review that are given by the users toward the companies.

Figure 171 Job Titles & Employment Status Tab Display

Figure 171 shows the display for the users to see the job titles and employment status of the reviewers. This display can give insights on the reviews' employment status as this may enable HR executives to know whether any actions that have been taken before in an effort to improve job satisfaction have worked or not.

Figure 172 Recommendations, Outlook & CEO Approval Tab Display

Figure 172 shows the display to show the reviewers' opinion on the company's outlook, CEO approval, and recommendations to go to the selected companies or not. This can be a quick way to assess whether an employee is willing to recommend your company to others.

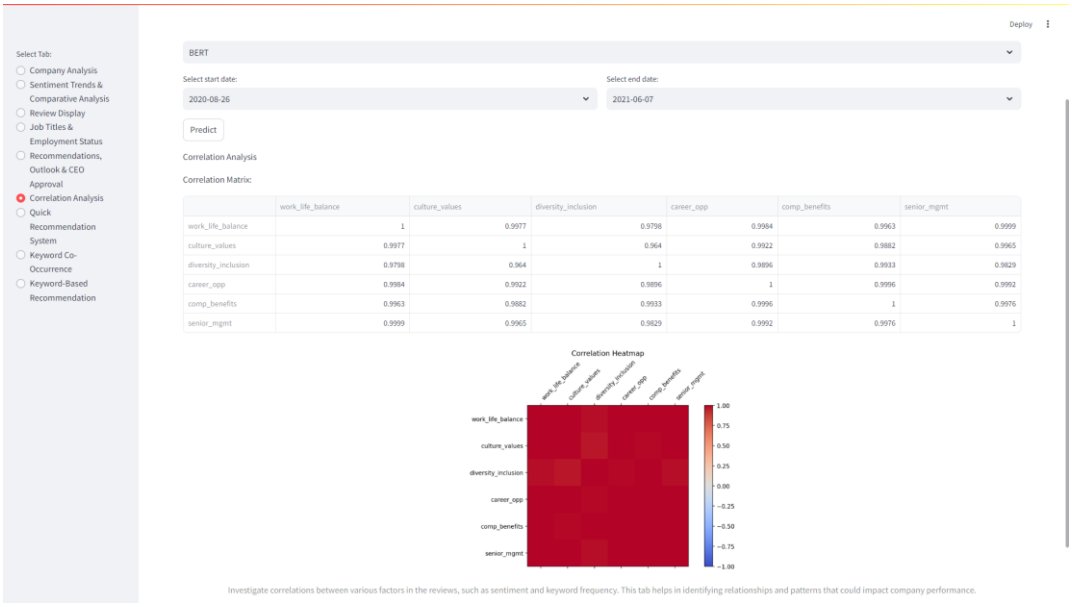


Figure 173 Correlation Analysis Tab Display

Figure 173 shows the display for correlation analysis, and the user can see and check whether the selected companies they chose have any correlations in five (5) variables. This is a valuable feature for HR executives as well because they can monitor competitors closely and make necessary changes.

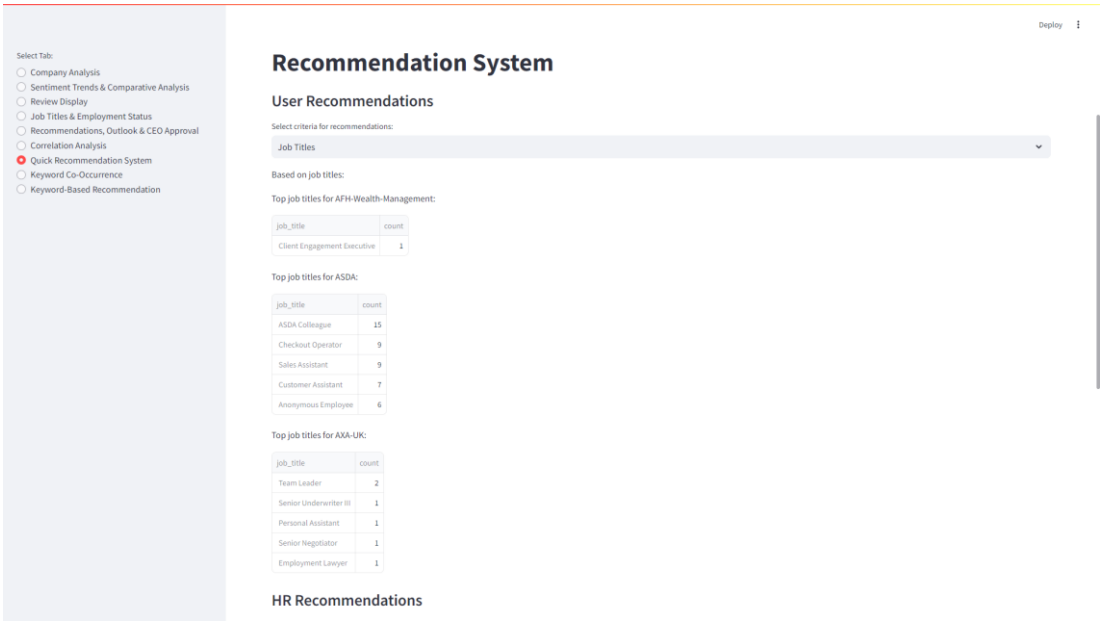


Figure 174 Quick Recommendation System Tab Display

Figure 174 shows a tab where the user and/or HR can get quick recommendations that are mostly required by job seekers on the selected companies. This includes job titles, sentiments, and overall ratings

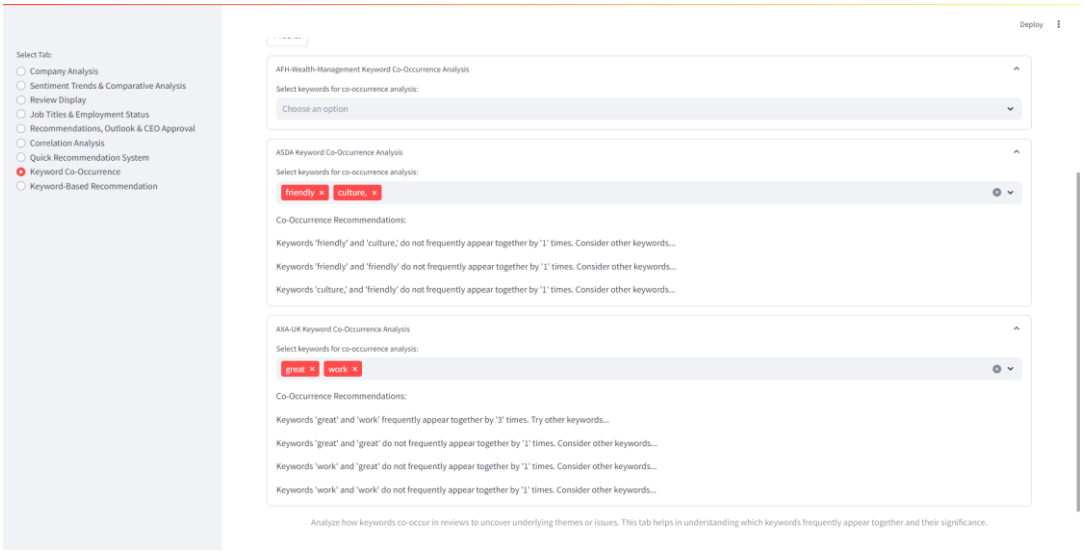


Figure 175 Keyword Co-Occurrence Tab Display

Figure 175 shows the display of the Keyword Co-Occurrence tab that enables users to check what keywords are occurring together in the reviews for the selected companies. This feature is one of the important ones because it allows the users to check whether their desired company meets or exceeds one of many requirements they have. For HR executives, this feature complements the correlation analysis as it allows for much deeper analysis on their competitors.

Figure 176 Keyword-Based Recommendation Tab Display

Figure 176 shows the recommendation system that is based on Keyword and its weights that is obtained through the predicting the reviews' sentiments. To further explain the reason a company is good or bad, keywords' frequencies and weights will be shown.

5.3.3 LIME Explanation Result Comparison

For the LIME Explanation result comparison, only oversampling models will be used for comparison as it has higher accuracy than their undersampling counterparts albeit only a little, and their result is almost similar to each other.

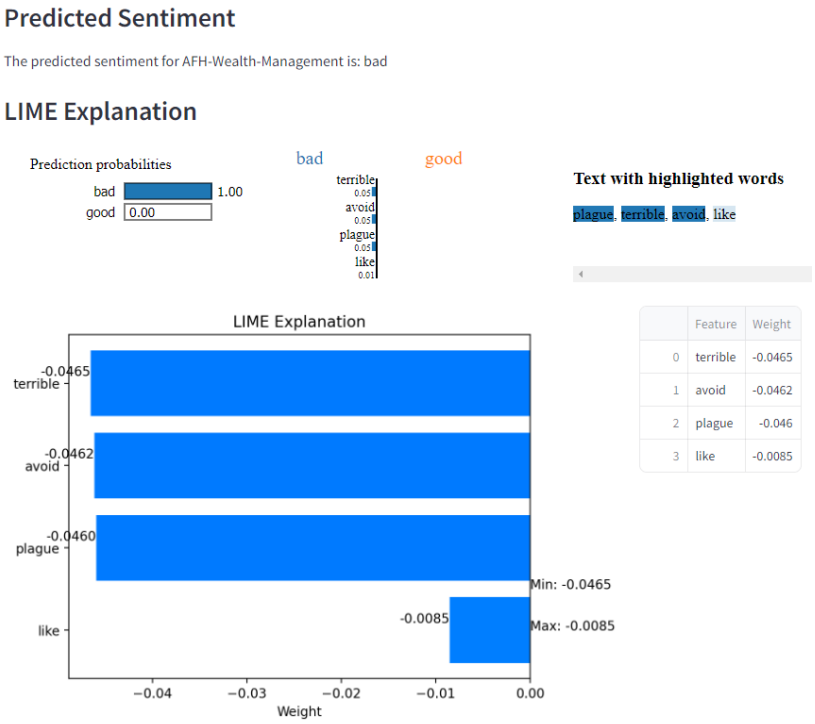


Figure 177 BERT Lime Explanation Result on AFH-Wealth-Management

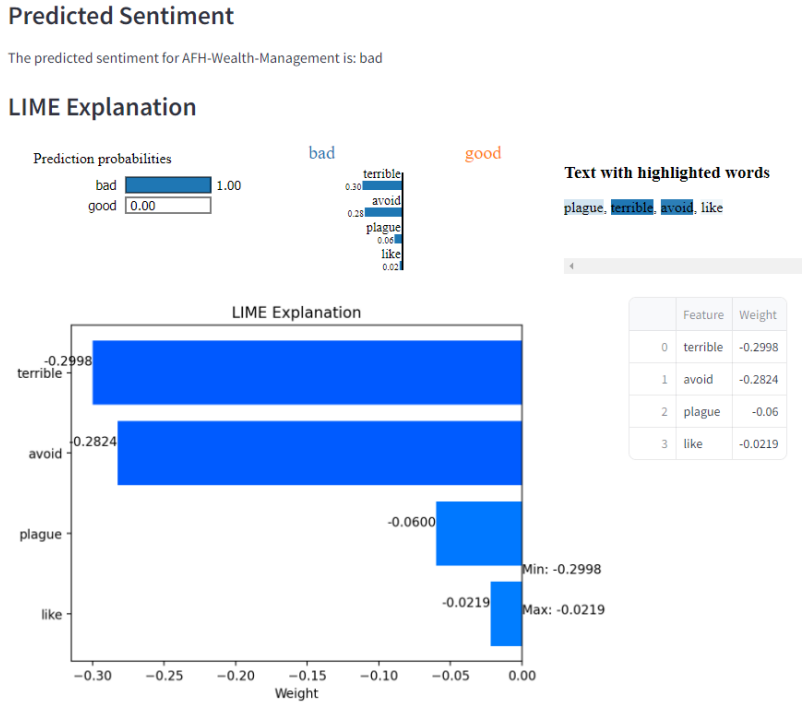


Figure 178 SVM Oversampling Lime Explanation Result on AFH-Wealth-Management

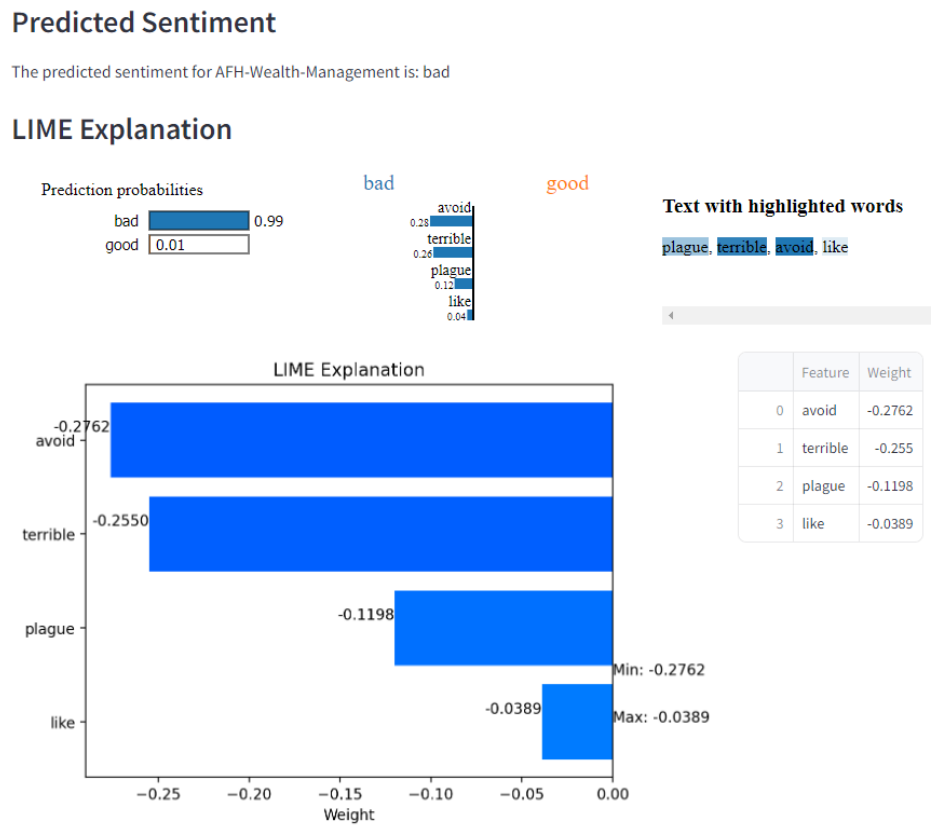


Figure 179 Random Forest Lime Explanation Result on AFH-Wealth-Management

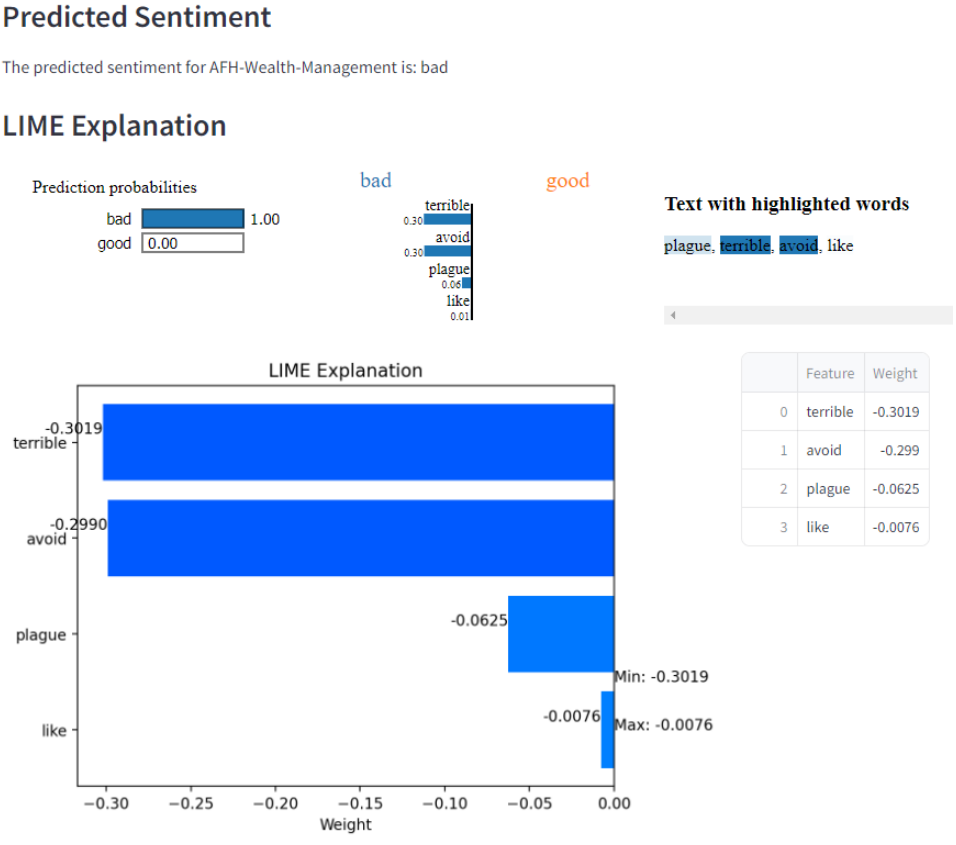


Figure 180 Logistic Regression Lime Explanation Result on AFH-Wealth-Management

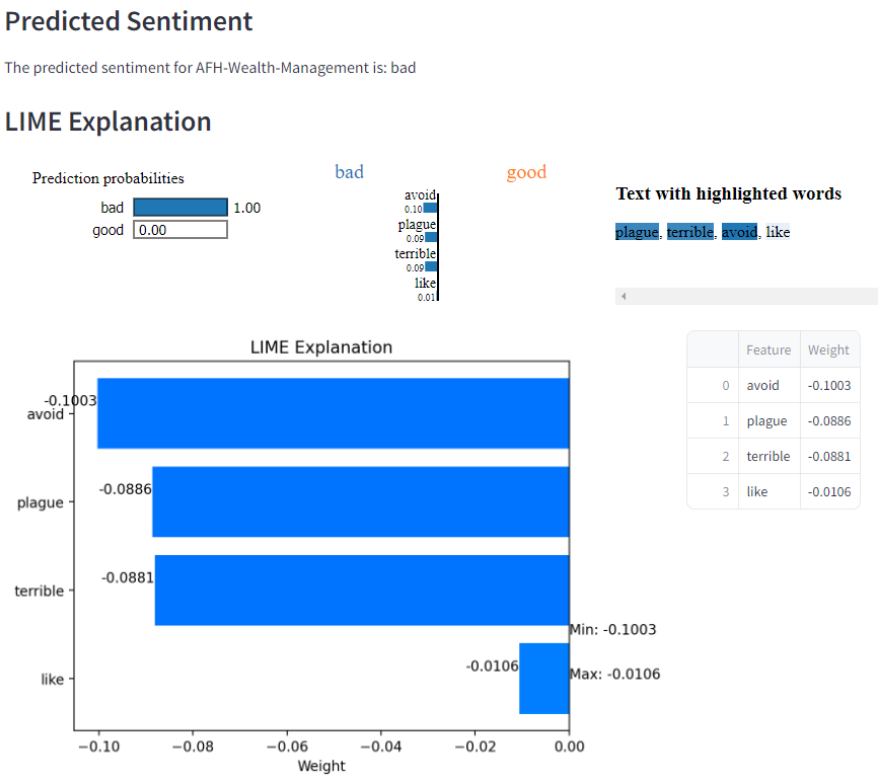


Figure 181 Naïve Bayes Lime Explanation Result on AFH-Wealth-Management

Based on Figures177 to 184, the Lime Explanation result on AFH-Wealth-Management company is basically almost the same with minor differences. BERT and Naïve Bayes model have more similar result than the others as they have 3 bars with similar weights, which are "avoid," "plague" and "terrible." This means they have higher accuracy in terms of finding the negative keywords as the company's sentiment is bad, hence they have the "like" keyword with lower weight. Meanwhile the Random Forest model had "plague" with neither near nor far with the max and min of negative bar. As for SVM and Logistic Regression, they have both "terrible" and "like" with lower negative weight.

The five models' results overall are pretty similar. However, Random Forest would take the win in this case even though it has slightly lower accuracy than BERT, because its running is shorter and therefore, more practical to use.

5.4 Summary

This chapter describes in detail how sentiment analysis models are deployed using a Streamlit application and how the extensive hyperparameter tweaking procedure is carried out. Cross-validation was used in a rigorous, iterative process called hyperparameter tuning to determine the ideal parameters for a variety of models, including BERT, SVM, Random Forest, Logistic Regression, and Naive Bayes. To ensure that the models operated at their best and were as accurate as possible, a method like grid search was used to fine-tune the parameters.

Streamlit was used in the deployment phase to develop an interactive and user-friendly online application that lets users view sentiment analysis results live. All the models were deployed and able to be used, however BERT was running noticeably longer than the other traditional models with neglectable accuracy. The app's key features included real-time data handling, intuitive navigation, vertical tabs for easy access, and functionalities such as company recommendations and outlook analysis. This combination of rigorous model optimization and practical deployment strengthens the project's foundation as well as increase its applicability and usability for HR experts and industrial stakeholders.

CHAPTER 6: CONCLUSION

6.1 Critical Evaluation

6.1.1 Overall Project Achievement

The project achieved significant milestones through the development, exploration, and evaluation of various sentiment analysis models, including SVM, Logistic Regression, Random Forest, Naive Bayes, and BERT. By implementing these models, the project demonstrated high test accuracies across different sampling methods, with BERT notably reaching an accuracy of 97% using undersampling. Despite BERT's slightly higher accuracy, its prolonged training and usage times make it less practical compared to the faster traditional models. The project effectively addressed class imbalances by employing undersampling and oversampling techniques, with oversampling generally resulting in higher model accuracies. Extensive hyperparameter tuning was performed for each model, leading to optimal performance, exemplified by the SVM and Random Forest models achieving validation accuracies of 96.2% and 96.5%, respectively. The models consistently exhibited high validation and test accuracies, underscoring their reliability and generalizability.

In addition to model development, the project included the creation of a Streamlit app for interactive data analysis. This app enables users to explore sentiment analysis results interactively, featuring functionalities like company selection, firm comparisons, and data visualization through charts and graphs. The user-friendly interface, with vertical tabs and dynamic data handling, enhances navigation and usability. Additional features such as company recommendations, outlook, and CEO approval provide comprehensive and extensive insights, while the ability to display example reviews and select different models supports comparative analysis.

6.1.2 Contribution of the Project Towards Communities and/or Industries

The project made significant contributions to both the community and industry by addressing the research gaps and trying to fill it as this FYP offered valuable insights into employee specific sentiments and experiences, helping companies address key issues like work-life balance, management practices, and compensation, potentially fostering healthier work environments. By utilizing publicly available Glassdoor data, the project highlighted the importance of open data for research and practical applications in sentiment analysis.

Industry-wise, it explored and provided actionable recommendations for HR departments in a form of web application using Streamlit Python module that can be used to enhance employee satisfaction and performance benchmarking and demonstrated the applicability of advanced models like BERT for understanding complex text data. Although it is applicable, the BERT model is not practical for this purpose as its running time is noticeably longer than the other traditional models.

The integration of KeyBERT introduces a new way for keyword extraction which enhanced the analysis by identifying significant terms in the reviews, improving the interpretability of traditional models like SVM, Logistic Regression, Random Forest, and Naive Bayes. The Streamlit app further supported decision-making by offering a tool for real-time sentiment analysis and served as an educational resource for demonstrating machine learning applications and aiding further research.

6.1.3 Strength of the Project

The aspect-based sentiment analysis project achieved significant strengths by comprehensively exploring and evaluating multiple traditional and advanced models—BERT, SVM, Logistic Regression, Random Forest, and Naive Bayes—providing a thorough comparison of their performance on the same dataset. This evaluation highlighted each model's strengths and weaknesses, offering valuable insights for future research. The effective use of undersampling and oversampling techniques addressed class imbalances, demonstrating SMOTE's effectiveness in improving accuracy. Extensive hyperparameter tuning ensured optimal model performance and provided detailed documentation of the best parameters for future reference.

The project achieved high validation and test accuracies across models, showcasing the robustness and generalizability of the results. Practical implications were also addressed, with actionable insights and recommendations for HR practices, emphasizing the potential of machine learning and sentiment analysis to positively impact workplace environments. Additionally, the development of a Streamlit app enhanced the project's value by making analysis results accessible and interactive through its dynamic data handling, user-friendly interface, and features like company recommendations and outlook.

6.2 Limitations

The project faces several data limitations, including dataset constraints and class imbalance with the overwhelming positive data compared to minimal negative data. The Glassdoor reviews dataset used for analysis may not represent all industries or geographic regions, limiting the generalizability of the findings. Additionally, being based on self-reported data, it can introduce biases such as overly positive or negative feedback based on individual experiences. Despite employing data balancing techniques, some models still encountered challenges with class imbalances, particularly for less represented sentiment categories.

While models like Naive Bayes and Logistic Regression are useful, they may not detect intricate patterns as effectively as advanced models like BERT. However, BERT is computationally expensive and requires significant hardware resources, making it less practical for resource-constrained environments and unsuitable use cases. The hyperparameter tuning process, though crucial for optimal performance, is time-consuming and computationally intensive, which can be impractical for real-time applications or large datasets. Additionally, the Streamlit app has limitations in real-time data handling and relies on users' familiarity with the interface, which could lead to misinterpretation of data without proper guidance. Lastly, external economic and social factors, such as economic downturns may influence the sentiment captured in reviews, which were not considered in the analysis.

6.3 Recommendations

Future research should consider expanding the dataset by incorporating data from various job review platforms and industry reports to enhance representativeness and generalizability. Addressing class imbalance could involve advanced oversampling techniques like Deep SMOTE and ADASYN. Exploring advanced model architectures such as transformers and hybrid models that are less complex than BERT yet provides the same performance as the traditional models, along with automated hyperparameter tuning methods like Bayesian optimization, could enhance model performance. The Streamlit app could be improved by integrating real-time data updates and offering comprehensive support. Collaborating with organizations and academic institutions could validate findings and incorporate macroeconomic indicators. These recommendations aim to achieve greater accuracy, scalability, and practical relevance, providing valuable insights for improving workplace environments and employee satisfaction.

REFERENCES

Alduayj, S. S., & Smith, P. (2019). Sentiment Classification and Prediction of Job Interview Performance. *2019 2nd International Conference on Computer Applications & Information Security (ICCAIS)*, 1–6. https://doi.org/10.1109/CAIS.2019.8769559

Anh, N., Nguyen, N., Mai, N., & Nguyen, T. (2018). Job Satisfaction in Developing Countries: An Evidence from a Matched Employer-employee Survey in Vietnam. *Journal of Economic Studies*, *46*. https://doi.org/10.1108/JES-04-2017-0096

Bajpai, R., Hazarika, D., Singh, K., Gorantla, S., Cambria, E., & Zimmerman, R. (2019). *Aspect-Sentiment Embeddings for Company Profiling and Employee Opinion Mining* (arXiv:1902.08342). arXiv. http://arxiv.org/abs/1902.08342

Brownlee, J. (2020, January 16). SMOTE for Imbalanced Classification with Python. *MachineLearningMastery.Com*. https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

Chun, J., & Elkins, K. (2023). eXplainable AI with GPT4 for story analysis and generation: A novel framework for diachronic sentiment analysis. *International Journal of Digital Humanities*, *5*(2), 507–532. https://doi.org/10.1007/s42803-023-00069-8

Costa, A., & Veloso, A. (2015). *Employee Analytics through Sentiment Analysis*. https://doi.org/10.13140/RG.2.1.1623.3688

Devika, M. D., Sunitha, C., & Ganesh, A. (2016). Sentiment Analysis: A Comparative Study on Different Approaches. *Procedia Computer Science*, *87*, 44–49. https://doi.org/10.1016/j.procs.2016.05.124

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* (arXiv:1810.04805). arXiv. https://doi.org/10.48550/arXiv.1810.04805

Dina, N. Z., & Juniarta, N. (2020). *Aspect based Sentiment Analysis of Employee's Review Experience*. *6*. https://e-journal.unair.ac.id/JISEBI/article/download/18104/10201

Gnepp, J., Klayman, J., Williamson, I. O., & Barlas, S. (2020). The future of feedback: Motivating performance improvement through future-focused feedback. *PLoS ONE*, *15*(6), e0234444. https://doi.org/10.1371/journal.pone.0234444

Gupta, B., Negi, M., Vishwakarma, K., Rawat, G., & Badhani, P. (2017). Study of Twitter Sentiment Analysis using Machine Learning Algorithms on Python. *International Journal of Computer Applications*, *165*, 29–34. https://doi.org/10.5120/ijca2017914022

Harfoushi, O., Hasan, D., & Obiedat, R. (2018). Sentiment Analysis Algorithms through Azure Machine Learning: Analysis and Comparison. *Modern Applied Science*, *12*(7), Article 7. https://doi.org/10.5539/mas.v12n7p49

Heimerl, P., Haid, M., Benedikt, L., & Scholl-Grissemann, U. (2020). Factors Influencing Job Satisfaction in Hospitality Industry. *Sage Open*, *10*(4), 2158244020982998. https://doi.org/10.1177/2158244020982998

Jawale, S., & Sawarkar, S. D. (2020). Interpretable Sentiment Analysis based on Deep Learning: An overview. *2020 IEEE Pune Section International Conference (PuneCon)*, 65–70. https://doi.org/10.1109/PuneCon50868.2020.9362361

Jemai, F., Hayouni, M., & Baccar, S. (2021). Sentiment Analysis Using Machine Learning Algorithms. *2021 International Wireless Communications and Mobile Computing (IWCMC)*, 775–779. https://doi.org/10.1109/IWCMC51323.2021.9498965

Judge, T. A., Zhang, S. (Carrie), & Glerum, D. R. (2020). Job Satisfaction. In *Essentials of Job Attitudes and Other Workplace Psychological Constructs*. Routledge.

Jung, Y., & Suh, Y. (2019). Mining the voice of employees: A text mining approach to identifying and analyzing job satisfaction factors from online employee reviews. *Decision Support Systems*, *123*, 113074. https://doi.org/10.1016/j.dss.2019.113074

Kalaivani, E. R., & Marivendan, E. R. (2021). The effect of stop word removal and stemming in datapreprocessing. *Annals of the Romanian Society for Cell Biology*, *25*(6), 739–746.

Kaur, G., & Sharma, A. (2022). Comparison of Different Machine Learning Algorithms for Sentiment Analysis. *2022 International Conference on Sustainable Computing and*

Data Communication Systems (ICSCDS), 141–147. https://doi.org/10.1109/ICSCDS53736.2022.9760846

Khan, M. Q., Shahid, A., Uddin, M. I., Roman, M., Alharbi, A., Alosaimi, W., Almalki, J., & Alshahrani, S. M. (2022). Impact analysis of keyword extraction using contextual word embedding. *PeerJ Computer Science*, *8*, e967. https://doi.org/10.7717/peerj-cs.967

Khyani, D., & B S, S. (2021). An Interpretation of Lemmatization and Stemming in Natural Language Processing. *Shanghai Ligong Daxue Xuebao/Journal of University of Shanghai for Science and Technology*, *22*, 350–357.

Kristiyanti, D. A., Normah, & Umam, A. H. (2019). Prediction of Indonesia Presidential Election Results for the 2019-2024 Period Using Twitter Sentiment Analysis. *2019 5th International Conference on New Media Studies (CONMEDIA)*, 36–42. https://doi.org/10.1109/CONMEDIA46929.2019.8981823

Li, L. (2020, August 3). *7 Ways Companies Transformed Their Annual Performance Reviews*. TINYpulse. https://www.tinypulse.com/blog/7-ways-companies-transformed-their-annual-performance-reviews

Lin, H. Y., & Moh, T.-S. (2021). Sentiment analysis on COVID tweets using COVID-Twitter-BERT with auxiliary sentence approach. *Proceedings of the 2021 ACM Southeast Conference*, 234–238. https://doi.org/10.1145/3409334.3452074

Loke, R., & Lam-Lion, R. (2021). A Company's Corporate Reputation through the Eyes of Employees Measured with Sentiment Analysis of Online Reviews: *Proceedings of the 10th International Conference on Data Science, Technology and Applications*, 378–385. https://doi.org/10.5220/0010620603780385

Luo, N., Zhou, Y., & Shon, J. (2016). *Employee Satisfaction and Corporate Performance: Mining Employee Reviews on Glassdoor.com*.

Lyu, C., Foster, J., & Graham, Y. (2020). *Improving Document-Level Sentiment Analysis with User and Product Context* (arXiv:2011.09210). arXiv. https://doi.org/10.48550/arXiv.2011.09210

Mehta, P., & Pandya, D. S. (2020). *A Review On Sentiment Analysis Methodologies, Practices And Applications*. *9*(02).

Miranda, M. D., & Sassi, R. J. (2014). Using Sentiment Analysis to Assess Customer Satisfaction in an Online Job Search Company. In W. Abramowicz & A. Kokkinaki (Eds.), *Business Information Systems Workshops* (Vol. 183, pp. 17–27). Springer International Publishing. https://doi.org/10.1007/978-3-319-11460-6_2

Mohammed, R., Rawashdeh, J., & Abdullah, M. (2020). Machine Learning with Oversampling and Undersampling Techniques: Overview Study and Experimental Results. *2020 11th International Conference on Information and Communication Systems (ICICS)*, 243–248. https://doi.org/10.1109/ICICS49469.2020.239556

Montuori, P., Sorrentino, M., Sarnacchiaro, P., Di Duca, F., Nardo, A., Ferrante, B., D'Angelo, D., Di Sarno, S., Pennino, F., Masucci, A., Triassi, M., & Nardone, A. (2022). Job Satisfaction: Knowledge, Attitudes, and Practices Analysis in a Well-Educated Population. *International Journal of Environmental Research and Public Health*, *19*(21), 14214. https://doi.org/10.3390/ijerph192114214

Mouli, N., Das, P., Bin Muquith, M., Biswas, A., & Kabir Niloy, M. D. (2023). *Sentiment analysis to determine employee job satisfaction using machine learning techniques* [Thesis, Brac University]. http://dspace.bracu.ac.bd/xmlui/handle/10361/19357

Mueller, A. (2022, April 8). *The Cost of Hiring a New Employee*. Investopedia. https://www.investopedia.com/financial-edge/0711/the-cost-of-hiring-a-new-employee.aspx

ÖzdemiR, A., Onan, A., & Çinarli Ergene, V. (2022). Topic Modelling and Artificial Intelligence based Method Using Online Employee Assessments to Analyse Job Satisfaction. *Turkish Journal of Forecasting*, *06*(2), 46–52. https://doi.org/10.34110/forecasting.1173063

Pavitha, N., Ratnaparkhi, P., Uzair, A., More, A., Raj, S., & Yadav, P. (2023). Explainable AI for Sentiment Analysis. In J. Choudrie, P. Mahalle, T. Perumal, & A. Joshi (Eds.), *ICT with Intelligent Applications* (pp. 429–439). Springer Nature. https://doi.org/10.1007/978-981-19-3571-8_41

Pradipta, G. A., Wardoyo, R., Musdholifah, A., Sanjaya, I. N. H., & Ismail, M. (2021). SMOTE for Handling Imbalanced Data Problem: A Review. *2021 Sixth International Conference on Informatics and Computing (ICIC)*, 1–8. https://doi.org/10.1109/ICIC54025.2021.9632912

Puspitasari, F. S. U. P., Ira. (2023, December 22). *Implementation of Sentiment Analysis in HR Management and Development Planning Using Topic Modelling on Employee Review | Kontigensi: Jurnal Ilmiah Manajemen*. https://jurnal.dim-unpas.web.id/index.php/JIMK/article/view/358

Qasim, R., Bangyal, W. H., Alqarni, M. A., & Ali Almazroi, A. (2022). A Fine-Tuned BERT-Based Transfer Learning Approach for Text Classification. *Journal of Healthcare Engineering*, *2022*(1), 3498123. https://doi.org/10.1155/2022/3498123

Saputra, N., Riyadi, A., & Tentua, M. N. (2023). *Sentiment Analysis of Covid Vaccination Policy In Indonesia Using Random Forest*. 205–209. https://doi.org/10.2991/978-94-6463-338-2_31

Schröer, C., Kruse, F., & Gómez, J. M. (2021). A Systematic Literature Review on Applying CRISP-DM Process Model. *Procedia Computer Science*, *181*, 526–534. https://doi.org/10.1016/j.procs.2021.01.199

Shin, M. G., Kim, Y.-K., Kim, S.-Y., & Kang, D. M. (2020). Relationship Between Job Training and Subjective Well-being In Accordance With Work Creativity, Task Variety, and Occupation. *Safety and Health at Work*, *11*(4), 466–478. https://doi.org/10.1016/j.shaw.2020.08.006

Vuong, B., Tung, D., Tushar, H., Quan, T., & Giao, H. (2021). Determinates of factors influencing job satisfaction and organizational loyalty. *Management Science Letters*, *11*(1), 203–212.

Vuong, T. D. N., & Nguyen, L. T. (2022). The Key Strategies for Measuring Employee Performance in Companies: A Systematic Review. *Sustainability*, *14*(21), Article 21. https://doi.org/10.3390/su142114017

Wongkar, M., & Angdresey, A. (2019). Sentiment Analysis Using Naive Bayes Algorithm Of The Data Crawler: Twitter. *2019 Fourth International Conference on Informatics and Computing (ICIC)*, 1–5. https://doi.org/10.1109/ICIC47613.2019.8985884

Xu, X., & Li, Y. (2016). The antecedents of customer satisfaction and dissatisfaction toward various types of hotels: A text mining approach. *International Journal of Hospitality Management*, *55*, 57–69. https://doi.org/10.1016/j.ijhm.2016.03.003

APPENDICES

Appendix A: Turnitin Report

FYP-Jason Yitro Setiadi-TP062295-APD3F2311CS(DA).docx

ORIGINALITY REPORT

15%	10%	8%	11%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Asia Pacific University College of Technology and Innovation (UCTI) Student Paper	5%
2	www.showdays.info Internet Source	<1%
3	Submitted to The University of Manchester Student Paper	<1%
4	Submitted to Southern New Hampshire University - Continuing Education Student Paper	<1%
5	www.mdpi.com Internet Source	<1%
6	Submitted to Liverpool John Moores University Student Paper	<1%
7	localiban.org Internet Source	<1%
8	www.tinypulse.com Internet Source	<1%

9	Submitted to Harrisburg University of Science and Technology Student Paper	<1 %
10	Submitted to Champlain College Student Paper	<1 %
11	Van Doel, Richard M.. "Exploring Stewardship Characteristics in Professional, Scientific, and Technical Services Employee-Owned Companies.", Indiana Wesleyan University, 2017 Publication	<1 %
12	Submitted to Curtin University of Technology Student Paper	<1 %
13	Sarah S. Alduayj, Phillip Smith. "Sentiment Classification and Prediction of Job Interview Performance", 2019 2nd International Conference on Computer Applications & Information Security (ICCAIS), 2019 Publication	<1 %
14	Submitted to Bridgepoint Education Student Paper	<1 %
15	dokumen.pub Internet Source	<1 %
16	Submitted to University of Leiden - EUR Student Paper	<1 %
	www.ijraset.com	

Appendix B: PPF

Office Record	Receipt
Date Received: Received by whom:	Student name: Student number: Received by: Date:

DRAFT PROJECT PROPOSAL FORM

Proposal ID :

Supervisor : 1) Ts. Dr. Law Foong Li (Ms.)
2) Mr. Raheem Mafas (Mr.)
3) Dr. Nowshath Kadhar Batcha (Mr.)
4) Ms. Minnu Helen Joseph (Ms.)
5) Ms. Nurul Izzatie (Ms.)

Student Name : JASON YITRO SETIADI

Student No : TP062295

Email Address : TP062295@mail.apu.edu.my & j4son18th@gmail.com

Programme Name : Bachelor in Computer Science with Specialism in Data Analytics (CSDA)

Title of project : Analyzing Glassdoor Job Reviews For Sustainable Work Environments: HR Impact And Societal Implications Through Sentiment Analysis

Please record which module(s) your topic is related to:
Data Mining and Predictive Modelling (022023-PRE)
Research Methods for Computing and Technology (022023-KBN)
Text Analytics and Sentiment Analysis (122023-RMF)

---

1. Introduction

Assume the reader has very little knowledge of the subject. Introduce the topic, the sector of business/industry concerned and how the project relates to it. Define the context of the problem and identify the research required to solve it.

In the ever-evolving landscape of the modern workforce, understanding the dynamics of the job market and the experiences of employees has become pivotal for all organizations, policymakers, and especially the workers. The popular trend for the current workforce is to change company where they work every one to two years or even months which is caused by *selective attention* to explore more careers and find workplace with fair compensation that give them the highest possible return on their job investments (Tejas, 2023). It is also reported that a millennial employee will only stay at their job for 2.75 years which the millennials deserved the label as the Job-Hopping generation as 21% of millennials surveyed report changing jobs within the past year – more than three times the rate of other generations (Zippia, 2022).

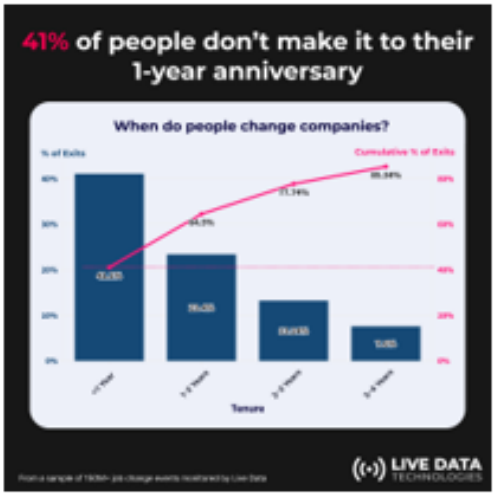


Figure 1 1-Year Anniversary Graph (https://shorturl.at/htxB5)

Figure 1 shows bar chart that represents 41% of people leave before their 1-year anniversary – resulting in $100B of recruiting and turnover waste every year so 2-in-5 hires across the white-collar workforce fail. This happen because most employees that left are millennials and they have characteristics such as: First, great desire to explore careers and industries; Second, millennials want to feel part of something bigger; and lastly, the modern management culture has shifted. This millennial turnover is costly, and it is estimated to cost $30.5 billion yearly. Job hopping proves to be very expensive and costly to the company and not to the employees as it also revealed that half of millennials in the workplace strongly disagree that they will be with their current employer in a year, and 60% are open to a new job opportunity, compared to 45% of non-millennials (Adkins, 2016).

The COVID-19 epidemic may have contributed to the job-hopping culture by encouraging more millennials to change jobs and by encouraging employment changes among employees of all generations. A 2021 IBM study on post-epidemic expectations among employees revealed that the majority of workers across all generations believed their employers did not provide enough assistance to them during the pandemic (IBM, 2021). As a result, many millennials, who make up more than one-third of the American workforce, looked for other jobs. This culture has also been modified for the Gen Z workforce, who frequently experiment with different jobs, industries, and occupations. They can set their own hours and work many jobs (hustles) to earn more money than they would if they were employed by a single business or institution, which would eventually pay less.

Naturally, the industry and company in which the workers are employed may have played a role in creating this culture, since employment in fields that demand long hours with little to no benefits tends to drive out workers. When you are coasting or at ease with what you are doing right now, that is another excellent argument. Nevertheless, those two valid arguments are starting to drive people to shift careers, believing that they would lose money if they remain in their current position even in the absence of an emergency, and that it is improper to engage in "gratitude shaming" (Taylor, 2019). These days, people called that as “Job-Hopping with intention” to increase salary, broaden their career prospects, and improve communication skills set (Pelta, 2020).

Given that organizations have a significant influence on society dynamics, workplace attitudes must be addressed to be in line with the Sustainable Development Goals (SDGs). The "Glassdoor Job Reviews" dataset offers a chance to comprehend and deal with issues pertaining to 8th goal which is Decent Work and Economic Growth as it provides a distinctive perspective on the attitudes and viewpoints of people employed in a variety of industries. Leading workplace insights site Glassdoor gives working employees a forum to openly discuss their experiences with their employers, covering a range of topics like work-life balance, business culture, and career advancement prospects. Through exploring employee sentiments, we hope to provide insights that can support inclusive and diverse work environments that can be used by their employers to improve work opportunities and system, and eventually contribute to the larger objectives of sustainable global development.

2. Problem Statement

Identify past and current work in the subject area. Outline the key references to other people's work, indicate for the most pertinent of these how your proposal relates to the ideas they contain.

Opinion mining, also known as sentiment analysis (SA), is an important field of study that examines people's thoughts, feelings, and attitudes as they are expressed in written language. However, the analysis and classification of text data is a difficult process due to the inherent differences between how humans and computers understand language, as well as the variety of data types, both organized and unstructured. Despite substantial progress in SA, particularly in natural language processing and data mining, the field grapples with persistent challenges in accurately interpreting sentiments and determining suitable sentiment polarity. Current research highlights the existence of different levels of sentiment analysis, each presenting unique challenges that necessitate innovative solutions for contextual interpretation. One specific focus within sentiment analysis pertains to job reviews, where sentiment analysis and emotion detection can offer valuable insights into employees' experiences and satisfaction levels, impacting both the individual and the organizational levels.

Recent studies on sentiment analysis in job reviews reveal a spectrum of limitations that warrant attention. These challenges encompass issues such as incomplete information in online reviews, leading to potential data gaps and loss of specific details critical for a comprehensive analysis. Language diversity within multicultural workplaces poses a significant hurdle, particularly when relying on machine translation for sentiment analysis. The societal impact of job reviews, especially concerning cultural and generational influences on workplace preferences, remains inadequately addressed by current sentiment analysis models. Additionally, there is a discernible lack of consideration for the evolving nature of work trends in the Fourth Industrial Revolution (Industry 4.0), where factors like cultural alignment and societal values play an increasingly significant role in shaping employee preferences and organizational dynamics. These multifaceted challenges underscore the need for advanced sentiment analysis models that account for diverse data types, language nuances, and evolving societal dynamics to ensure a comprehensive understanding of sentiments in the modern workplace.

3. Project Aim and Objectives

Identify the AIM(s) of the project, i.e. what the overall achievement is intended to be, in terms of both academic and commercial/industrial advances. Identify the particular intellectual difficulties posed by the proposal, the problems to be addressed, and explain how these might be solved. Clearly list individual measurable OBJECTIVES which can be related to the workplan and deliverables.

Aims and objectives are subject to approval from supervisor and students are expected to revise them if deemed inappropriate for a Level 3 project.

Aim:

To analyze sentiment patterns in Glassdoor job reviews to gain insights into workplace experiences and contribute to fostering better work environments in regard to HR impact and societal implications.

Objectives:

1. To investigate the impact of HR practices on the sustainability of work environments and the relationship between HR initiatives and employee sentiment in job evaluations.
2. To conduct a comparative analysis of job reviews across different industries/sectors to understand variances in sustainability practices and employee sentiment.
3. To develop actionable recommendations model for HR departments to enhance workplace sustainability based on sentiment analysis findings.
4. To evaluate the performance of the recommendations model developed in Research Objective 3.

4. Literature Review

The world has entered the digital age, and we are seeing a tremendous advancement in technology that makes it quite simple for us to gather and store vast amounts of data—what is commonly referred to as "Big Data." A vast quantity of data can take many different forms, both organized and unstructured, but text-based data, such as that found in books, articles, documents, and even posts on social media, is the sort that we frequently encounter everywhere we go. Any company or even individual can use this data to ascertain their attitudes, whether they are favourable or negative, or whether any more insights can be deduced practically automatically. However, because computers understand numbers rather than letters and/or text the way humans do, they make it more difficult and time-consuming to analyse and categorize data that is based on text. Since computers lack built-in dictionaries, we must start from scratch when teaching language structure and grammar which can be tackled using sentiment analysis.

Sentiment analysis is the study of analysing written language to determine people's opinions, feelings, assessments, attitudes, and emotions. For text analysis, you must find a way to mathematically describe the text and sentence structure without sacrificing informational value; if you're assessing popularity, you might also want to investigate sentiment analysis methods (Liu, 2012). Aiming to identify documents based on their attitudes rather than their topics using machine learning techniques, one of the first sentiment analysis journals was founded in the early 2000s. They discovered that their use of support vector machines (SVM), maximum entropy classification, and Naïve Bayes was underperforming compared to the conventional topic-based categorization. They came to the conclusion that bigrams, parts of speech, position, and feature frequency vs. presence could be contributing variables to those (Pang et al., 2002).

In the meantime, an unsupervised learning system was described in the second publication to categorize reviews into two categories: recommended and not recommended. When using a semantic orientation, they encountered difficulties locating context that differed from the review's headline or title. Combining this with other features in a supervised classification algorithm could solve this (Turney, 2001).

Figure 2 Levels of Sentiment Analysis (Wankhade et al., 2022)

Since it is simpler to store data effectively in this digital era and without concern for capacity size, the amount of data in text form has grown dramatically. However, the difficulty of sentiment analysis is on how the text data is analyzed with regards to their context. Determining the proper feeling polarity and accurately interpreting sentiments are hindered by these difficulties. There are different degrees of sentiment analysis, as illustrated in Figure 2, and these problems are investigated to determine future approaches as each model has a different size and formula (Wankhade et al., 2022).

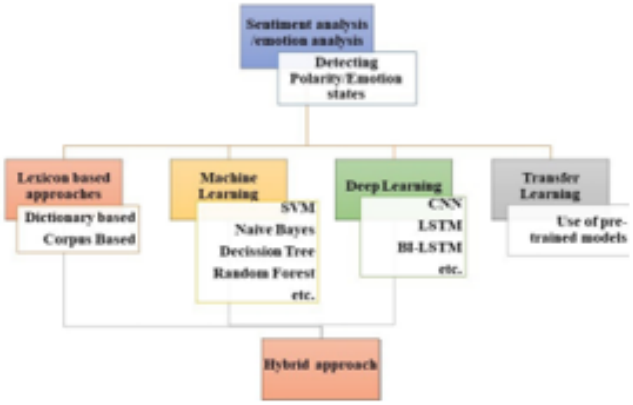


Figure 3 Techniques for Sentiment Analysis and Emotion Detection (Nandwani & Verma, 2021)

Sentiment Analysis itself can be done in many types of text-based data, for example movie reviews, article texts, novels paragraph, etc. The one that will be highlighted in this document is job review. Beside sentiment analysis, the data of job review can be used on emotion detection as shown in Figure 3 (Nandwani & Verma, 2021).

According to a study, there is a way to thoroughly examine employee internet reviews to avoid overlooking any important details. Utilizing the complete distribution of the reviews in the dataset would be another method to avoid it. It is important to address this challenge as it aimed to provide job satisfaction, financial and management between employees' sentiments and firms'

financial outcomes between creating a better workplace culture and short-term financial performance. This is helpful because employees frequently did not fill in their information completely (Feng, 2023). Another journal also uses job evaluations to assist job searchers in locating a company that fully satisfies their needs, and it discovered the same issue: most online reviews lack complete response since some of the company's detailed information is absent. In order to prevent the missing of specific information, the data is therefore analyzed in detail using an aspect-based sentiment analysis generated from user review data, which is grouped into five aspects: work balance, culture value, career opportunities, corporate benefit, and management (Dina & Juniarta, 2020).

In most cases, employees from different nationalities coexist in the same organization and may interact in a variety of languages. This could make it more difficult to analyze job reviews if you are limited to using only one language, such as English, which is typically the primary language in a multicultural workplace. A multilingual job review database will require more steps in the sentiment analysis process because of the need to translate. Additionally, because computer translation is not yet perfected and occasionally requires human review, it may result in mistranslations and prove to be inefficient for large databases. Understanding employees through their comments may not succeed before it even begins (Young & Gavade, 2018). On the other hand, a similar work was done on job review, however the dataset was considerably small with only close to hundred thousand making the accuracy not to flawless, and they had not tried any other models (Mouli et al., 2023).

In order to gain a deeper understanding of the dataset, it may be possible to incorporate hybrid models such as ensemble models, multi-modal models, transfer learning models, hybrid deep learning models, semantic-based models, etc. into sentiment analysis models. Simultaneously, these models can incorporate a wide range of data types, including text, images, and audio, and they can employ different learning approaches, such as traditional machine learning or deep learning.

During my investigation into sentiment analysis-based initiatives, I discovered that there are no publications or projects that discuss the influence of society on job reviews, which might encompass the five factors listed above. With the advent of Industry 4.0, or the Fourth Industrial Revolution, which digitizes every aspect of our lives, including job searching, the nature of labor has drastically changed. Since the journals did not approach the issue from a bird's-eye

perspective, they did not include anything that would lead to a change in firm or occupation as these days millennials look at the company's culture and ideals in addition to salary when looking for a job. The present trend of consumers refusing to buy products that might be at odds with their culture or religion is one illustration of what's going on. Negative evaluations on workplace opinion aggregator websites like Glassdoor have the potential to harm an organization's brand and cause discord, conflict, and instability within and outside the workplace.

How do you employ sentiment analysis and natural language processing (NLP) given the constraints and current understanding found in research papers to ensure that nothing is overlooked? In order to better understand the category and accuracy, the current project intends to map all of the existing sentiment analysis and machine learning models and investigate additional hybrid models, such as Multimodal Sentiment Analysis (MSA) to go beyond a simple positive/negative classification and analyze reviews for specific emotions like frustration, anxiety, joy, or excitement as this can reveal deeper insights into employee experiences. Second, targeted sentiment analysis would be to analyze sentiment towards specific aspects of the job, such as work-life balance, company culture, management, or career growth opportunities.

In addition to the model itself, analysis of the dataset variables will be carried out to generate fresh insights and potentially break down any complex structures that may exist in a multicultural workplace among various departments, companies, industries, and sectors. The comparative sentiment analysis to identify disparities and commonalities in their experiences. The dataset that will be used includes time variable which allows us to do temporal sentiment analysis to analyze how sentiment changes over time, looking for trends or correlations with company events or policy changes.

At the end, a lot of applications can be applied from the result of the analysis, such as:

1. Analyse job applications or interview transcripts to identify promising candidates based on their enthusiasm and fit for the company culture.
2. Use sentiment analysis to predict employee turnover or job satisfaction, allowing companies to take proactive measures to retain talent.
3. Recommend career paths or training opportunities based on an employee's expressed interests and career goals, as identified through sentiment analysis of their feedback.
4. Combine sentiment analysis of job reviews with other data like employee surveys, performance metrics, or social media activity to build a profile of employee sentiment.

5. Deliverables

Provide a clear list of the outputs from the project.

Anyone, notably HR departments, who would like to perform the same action on their dataset of employee reviews with deliverables, can rely on the sentiment analysis model to provide a high-accuracy analysis:

1. Develop a comprehensive research report investigating the impact of HR practices on the sustainability of work environments. This report will encompass findings related to the relationship between HR initiatives and employee sentiment in job evaluations. The analysis will be based on sentiment analysis applied to job reviews, offering insights into how HR practices influence the overall work environment sustainability and employee satisfaction.
2. Conduct a thorough comparative analysis of job reviews across different companies, industries, and sectors. This deliverable will provide insights into the variances in sustainability practices and employee sentiment, shedding light on industry-specific trends and challenges. The comparative analysis will be presented in a structured format, highlighting key findings and patterns discerned from sentiment analysis applied to job reviews in diverse industries.
3. Develop an actionable recommendations model for HR departments based on the sentiment analysis findings from job reviews. This model aims to offer practical suggestions for HR professionals to enhance workplace sustainability. The recommendations will be derived from patterns and insights identified in the sentiment analysis, providing a tailored approach for HR departments to address specific areas that impact employee sentiment and overall workplace sustainability.
4. Evaluate the performance of the recommendations model developed in Research Objective 3. This deliverable involves assessing the effectiveness of the actionable recommendations in influencing HR practices and improving workplace sustainability. The evaluation will be conducted through metrics such as changes in employee sentiment over time, implementation success rates of suggested practices, and potential feedback from HR departments. This assessment will provide valuable insights into the practical applicability and impact of the developed recommendations model.

6. References

Adkins, A. (2016, May 12). *Millennials: The Job-Hopping Generation*. Gallup.Com. https://www.gallup.com/workplace/231587/millennials-job-hopping-generation.aspx

Dina, N. Z., & Juniarta, N. (2020). *Aspect based Sentiment Analysis of Employee's Review Experience*. *6*. https://e-journal.unair.ac.id/JISEBI/article/download/18104/10201

Feng, S. (2023). Job satisfaction, management sentiment, and financial performance: Text analysis with job reviews from indeed.com. *International Journal of Information Management Data Insights*, *3*(1), 100155. https://doi.org/10.1016/j.jjimei.2023.100155

IBM. (2021, February 24). *What employees expect in 2021*. IBM. https://www.ibm.com/thought-leadership/institute-business-value/en-us/report/employee-expectations-2021

Liu, B. (2012). *Sentiment Analysis and Opinion Mining*. Springer International Publishing. https://doi.org/10.1007/978-3-031-02145-9

Mouli, N., Das, P., Bin Muquith, M., Biswas, A., & Kabir Niloy, M. D. (2023). *Sentiment analysis to determine employee job satisfaction using machine learning techniques* [Thesis, Brac University]. http://dspace.bracu.ac.bd/xmlui/handle/10361/19357

Nandwani, P., & Verma, R. (2021). A review on sentiment analysis and emotion detection from text. *Social Network Analysis and Mining*, *11*(1), 81. https://doi.org/10.1007/s13278-021-00776-6

Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up?: Sentiment classification using machine learning techniques. *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - EMNLP '02*, *10*, 79–86. https://doi.org/10.3115/1118693.1118704

Pelta, R. (2020, November 22). *Job Hopping with Intention: Pros, Cons, and Considerations*. FlexJobs Job Search Tips and Blog. https://www.flexjobs.com/blog/post/job-hopping-v2/

Taylor, H. (2019, May 16). *7 reasons to change jobs, even if you don't want to*. Business Insider Nederland. https://www.businessinsider.nl/why-change-jobs-2019-5/

Tejas, V. (2023, October 24). *Why Millennials and Gen Z Change Jobs Often*. Business News Daily. https://www.businessnewsdaily.com/7012-millennial-job-hopping.html

Turney, P. D. (2001). Thumbs up or thumbs down?: Semantic orientation applied to unsupervised classification of reviews. *Proceedings of the 40th Annual Meeting on*

Association for Computational Linguistics - ACL '02, 417. https://doi.org/10.3115/1073083.1073153

Wankhade, M., Rao, A. C. S., & Kulkarni, C. (2022). A survey on sentiment analysis methods, applications, and challenges. *Artificial Intelligence Review*, *55*(7), 5731–5780. https://doi.org/10.1007/s10462-022-10144-1

Young, L. M., & Gavade, S. R. (2018). Translating emotional insights from hospitality employees' comments: Using sentiment analysis to understand job satisfaction. *International Hospitality Review*, *32*(1), 75–92. https://doi.org/10.1108/IHR-08-2018-0007

Zippia. (2022, June 29). *How long do millennials stay at a job?* https://www.zippia.com/answers/how-long-do-millennials-stay-at-a-job/

Appendix C: Ethics Forms (Fast Track)

Office Record	Receipt – Fast-Track Ethical Approval
Date Received:	Student name: Student number: Received by:
Received by whom:	Date:

APU / APIIT FAST-TRACK ETHICAL APPROVAL FORM (STUDENTS)

Tick one box (level of study):
- ☐ POSTGRADUATE (PhD / MPhil / Masters)
- √ UNDERGRADUATE (Bachelors degree)
- ☐ FOUNDATION / DIPLOMA / Other categories

Tick one box (purpose of approval):
- √ Thesis / Dissertation / FYP project
- ☐ Module assignment
- ☐ Other: _____

Title of Programme on which enrolled **B. Sc. (Hons) Computer Science Specialism in Data Analytics**

Tick one box: √ Full-Time Study or ☐ Part-Time Study

Title of project / assignment **Aspect-Based Sentiment Analysis on Achieving Sustainable Work Environments Through Recommendation Model on Job Review**

Name of student researcher **Jason Yitro Setiadi**

Name of supervisor / lecturer **Ts. Dr. Law Foong Li**

you may need to consult when completing this form.

Supervisors/Module Lecturers - please seek guidance from the Chair of the APU Research Ethics Committee if you are uncertain about any ethical issue arising from this application.

		YES	NO	N/A
1	Will you describe the main procedures to participants in advance, so that they are informed about what to expect?			√
2	Will you tell participants that their participation is voluntary?			√
3	Will you obtain written consent for participation?			√
4	If the research is observational, will you ask participants for their consent to being observed?			√
5	Will you tell participants that they may withdraw from the research at any time and for any reason?			√
6	With questionnaires and interviews will you give participants the option of omitting questions they do not want to answer?			√
7	Will you tell participants that their data will be treated with full confidentiality and that, if published, it will not be identifiable as theirs?			√
8	Will you give participants the opportunity to be debriefed i.e. to find out more about the study and its results?			√

If you have ticked **No** to any of Q1-8 you should complete the full Ethics Approval Form.

		YES	NO	N/A
9	Will your project/assignment deliberately mislead participants in any way?		√	
10	Is there any realistic risk of any participants experiencing either physical or psychological distress or discomfort?		√	
11	Is the nature of the research such that contentious or sensitive issues might be involved?		√	

If you have ticked **Yes** to 9, 10 or 11 you should complete the full Ethics Approval Form. In relation to question 10 this should include details of what you will tell participants to do if they should experience any problems (e.g. who they can contact for help). You may also need to consider risk assessment issues.

		YES	NO	N/A

12	Does your project/assignment involve work with animals?			√	
13	Do participants fall into any of the following special groups? **Note that you may also need to obtain satisfactory clearance from the relevant authorities**	Children (under 18 years of age)		√	
		People with communication or learning difficulties			
		Patients			
		People in custody			
		People who could be regarded as vulnerable			
		People engaged in illegal activities (eg drug taking)			
14	Does the project/assignment involve external funding or external collaboration where the funding body or external collaborative partner requires the University to provide evidence that the project/assignment had been subject to ethical scrutiny?			√	

If you have ticked **Yes** to 12, 13 or 14 you should complete the full Ethics Approval Form. There is an obligation on student and supervisor to bring to the attention of the APU Research Ethics Committee any issues with ethical implications not clearly covered by the above checklist.

STUDENT RESEARCHER
Provide in the boxes below (plus any other appended details) information required in support of your application, THEN SIGN THE FORM.

Please Tick Boxes

I consider that this project/assignment has no significant ethical implications requiring a full ethics submission to the APU Research Ethics Committee.	√
Give a brief description of participants and procedure (methods, tests used etc) in up to 150 words. The project is to research on how sentiment analysis can be used to analyze job review in effort to by adopting CRISP-DM methodology. The project will utilize an open dataset source from Kaggle.com with 838,567, and 18 variables. The link of the dataset: https://www.kaggle.com/datasets/davidgauthier/glassdoor-job-reviews. The dataset will then proceed to the data cleaning and pre-processing including handling missing values, data normalization, data transformation which then continued into feature selection and further classification analysis using a machine learning model for accurate sentiment analysis. The models will then be evaluated using a confusion matrix that includes precision, accuracy, and recall ensuring the workings of the model. Not only that, but the model will also be integrated with interpretable AI to interpret the model findings as to make sure its efficacy on solving the research questions. There is no survey or interview to be conducted for the project.	
I also confirm that: i) All key documents e.g. consent form, information sheet, questionnaire/interview are appended to this application. Or	N/A
ii) Any key documents e.g. consent form, information sheet, questionnaire/interview schedules which need to be finalised following initial investigations will be submitted for approval by the project/assignment supervisor/module lecturer before they are used in primary data collection.	

E-signature
(Student Researcher)

Print Name **Jason Yitro Setiadi** Date **20 February 2024**

Please note that any variation to that contained within this document that in any way affects ethical issues of the stated research requires the appending of new ethical details. New ethical consent may need to be sought.
The completed form (and any attachments) should be submitted for consideration by your Supervisor/Module Lecturer

SUPERVISOR/MODULE LECTURER
PLEASE CONFIRM THE FOLLOWING:

	Please Tick Box
I consider that this project/assignment has no significant ethical implications requiring a full ethics submission to the APU Research Ethics Committee	✓
i) I have checked and approved the key documents required for this proposal (e.g. consent form, information sheet, questionnaire, interview schedule) Or	
ii) I have checked and approved draft documents required for this proposal which provide a basis for the preliminary investigations which will inform the main research study. I have informed the student researcher that finalised and additional documents (e.g. consent form, information sheet, questionnaire, interview schedule) must be submitted for approval by me before they are used for primary data collection.	✓

SUPERVISOR AND SECOND ACADEMIC SIGNATORY

STATEMENT OF ETHICAL APPROVAL (please delete as appropriate)

1) **THIS PROJECT/ASSIGNMENT HAS BEEN CONSIDERED USING AGREED APU/SU PROCEDURES AND IS NOW APPROVED**

2) **THIS PROJECT/ASSIGNMENT HAS BEEN APPROVED IN PRINCIPLE AS INVOLVING NO SIGNIFICANT ETHICAL IMPLICATIONS, BUT FINAL APPROVAL FOR DATA COLLECTION IS SUBJECT TO THE SUBMISSION OF KEY DOCUMENTS FOR APPROVAL BY SUPERVISOR (see Appendix A)**

E-signature...  Print Name... Ts. Dr. Law Foong Li Date... 20 Feb 2024
(Supervisor/Lecturer)

E-signature... Print Name... Date...
(Second Academic Signatory)

Office Record	Receipt – Appendix A (Fast-Track Ethics Form)
Date Received:	Student name: Student number: Received by:
Received by whom:	Date:

APPENDIX A
AUTHORISATION FOR USE OF KEY DOCUMENTS

Completion of Appendix A is required when for good reasons key documents are not available when a fast-track application is approved by the supervisor/module lecturer and second academic signatory.

I have now checked and approved all the key documents associated with this proposal e.g. consent form, information sheet, questionnaire, interview schedule

Title of project/assignment... Aspect-Based Sentiment Analysis on Achieving Sustainable Work Environments Through Recommendation Model on Job Review

Name of student researcher ... Jason Yitro Setiadi

Student ID: TP062295 Intake: APD3F2311CS(DA)

E-signature... Print Name... Ts. Dr. Law Foong Li Date... 20 Feb 2024
(Supervisor/Lecturer)

Fast-Track Ethical Approval Form ver. 3.0 (Dec 2015) Page 4 of 4

Appendix D: Log Sheets (6 Log Sheets)

Log Sheet No. 1

(APU: Serial Number)

PLS V1.0

Project Log Sheet – Supervisory | **Session**

Notes on use of the project log sheet:

1. This log sheet is designed for meetings of more than 15 minutes duration, of which there must be at minimum SIX (6) during the course of the project (SIX mandatory supervisory sessions).
2. The student should prepare for the supervisory sessions by deciding which question(s) he or she needs to ask the supervisor and what progress has been made (if any) since the last session and noting these in the relevant sections of the form, effectively forming an agenda for the session.
3. A log sheet is to be brought by the STUDENT to each supervisory session.
4. The actions by the student (and, perhaps the supervisor), which should be carried out before the next session should be noted briefly in the relevant section of the form.
5. The student should leave a copy (after the session) of the Project Log Sheet with the supervisor and to the administrator at the academic counter. A copy is retained by the student to be filed in the project file.
6. It is recommended that students bring along log sheets of previous meetings together with the project file during each supervisory session.
7. The log sheet is an important deliverable for the project and an important record of a student's organisation and learning experience. The student **must** hand in the log sheets as an appendix of the final year documentation, with sheets dated and numbered consecutively.

Student's name: Jason Yitro Setiadi **Date:** Wednesday, 30 January 2024 **Meeting No:** 1

Project title: Analysing Glassdoor Job Reviews for Sustainable Work Environments: HR Impact and Societal Implications through Sentiment Analysis **Intake:** APD3F2311CS(DA)

Supervisor's name: Ts. Dr. Law Foong Li **Supervisor's signature:**

Items for discussion (noted by student before mandatory supervisory meeting):

1. Checking and Changing of Final Year Project's title
2. Investigation Report Chapter 1 Checking
3.
4.

Record of discussion (noted by student during mandatory supervisory meeting):

1. Research Gap and Questions
2. Literature Review
3.
4.

Action List (to be attempted or completed by student by the next mandatory supervisory meeting):

1. Design a concept where an algorithm for sentiment analysis is built for both diagnostic and prescriptive, which able to discern what is wrong and give recommendations toward the problem.
2. Change my FYP title into the recommended one: **Sentiment Analysis Approach on Achieving Sustainable Work Environments Through Recommendation Model on Job Review**
3. Do investigation on algorithms used on sentiment analysis and/or similar works. Also research the features that are available and exist for the sentiment analysis and recommendations model.

Project Log Sheet

4. Accuracy is useless without explanation and/or visualization, therefore we need interpretable AI by using Lime. We need to see the factors that make the negative and positive review. Only by then we can make recommendations model that able to make the negative positive. And at the same time, breakdown the positive review.

5. Make sure that Research Question has the Big Question, and its sub questions corresponds to both Objectives and Research Gap.

6. Research Gap should have both problem and answer to the Questions and Solutions

7. Finish IR before second meeting after CNY.

Note: A student should make an appointment to meet his or her supervisor (via the consultation system) at least ONE (1) week prior to a mandatory supervisor session – please see document on project timelines. In the event a supervisor could not be booked for consultation, the project manager should be informed ONE (1) week prior to the session so that a meeting can be subsequently arranged.

Project Log Sheet

Project Log Sheet No. 2

(APU: Serial Number)

PLS V1.0

Project Log Sheet – Supervisory | **Session**

Notes on use of the project log sheet:

1. This log sheet is designed for meetings of more than 15 minutes duration, of which there must be at minimum SIX (6) during the course of the project (SIX mandatory supervisory sessions).
2. The student should prepare for the supervisory sessions by deciding which question(s) he or she needs to ask the supervisor and what progress has been made (if any) since the last session and noting these in the relevant sections of the form, effectively forming an agenda for the session.
3. A log sheet is to be brought by the STUDENT to each supervisory session.
4. The actions by the student (and, perhaps the supervisor), which should be carried out before the next session should be noted briefly in the relevant section of the form.
5. The student should leave a copy (after the session) of the Project Log Sheet with the supervisor and to the administrator at the academic counter. A copy is retained by the student to be filed in the project file.
6. It is recommended that students bring along log sheets of previous meetings together with the project file during each supervisory session.
7. The log sheet is an important deliverable for the project and an important record of a student's organisation and learning experience. The student **must** hand in the log sheets as an appendix of the final year documentation, with sheets dated and numbered consecutively.

Student's name: Jason Yitro Setiadi **Date:** Monday, 4 March 2024 **Meeting No: 2**

Project title: Aspect-Based Sentiment Analysis on Achieving Sustainable Work Environments Through Recommendation Model on Job Review **Intake:** APD3F2311CS(DA)

Supervisor's name: Ts. Dr. Law Foong Li **Supervisor's signature:**

Items for discussion (noted by student before mandatory supervisory meeting):

1. Revision of Chapter 1 and 2
2.
3.

Record of discussion (noted by student during mandatory supervisory meeting):

1. Chapter 2.4.3 Technical Research
2. Fix all the little things that are marked by lecturer.
3.
4.

Action List (to be attempted or completed by student by the next mandatory supervisory meeting):

1. To do the revision marked by the lecturer.
2. Do Data Cleaning and Pre-Processing

Note: A student should make an appointment to meet his or her supervisor (via the consultation system) at least ONE (1) week prior to a mandatory supervisor session – please see document on project timelines. In the event a supervisor could not be booked for consultation, the project manager should be informed ONE (1) week prior to the session so that a meeting can be subsequently arranged.

Project Log Sheet

Project Log Sheet No.3

(APU: Serial Number)

PLS V1.0

Project Log Sheet – Supervisory **Session**

Notes on use of the project log sheet:

1. This log sheet is designed for meetings of more than 15 minutes duration, of which there must be at minimum SIX (6) during the course of the project (SIX mandatory supervisory sessions).
2. The student should prepare for the supervisory sessions by deciding which question(s) he or she needs to ask the supervisor and what progress has been made (if any) since the last session and noting these in the relevant sections of the form, effectively forming an agenda for the session.
3. A log sheet is to be brought by the STUDENT to each supervisory session.
4. The actions by the student (and, perhaps the supervisor), which should be carried out before the next session should be noted briefly in the relevant section of the form.
5. The student should leave a copy (after the session) of the Project Log Sheet with the supervisor and to the administrator at the academic counter. A copy is retained by the student to be filed in the project file.
6. It is recommended that students bring along log sheets of previous meetings together with the project file during each supervisory session.
7. The log sheet is an important deliverable for the project and an important record of a student's organisation and learning experience. The student **must** hand in the log sheets as an appendix of the final year documentation, with sheets dated and numbered consecutively.

Student's name: Jason Yitro Setiadi **Date:** Tuesday, 12 March 2024 **Meeting No: 3**

Project title: Aspect-Based Sentiment Analysis on Achieving Sustainable Work Environments Through Recommendation Model on Job Review **Intake:** APD3F2311CS(DA)

Supervisor's name: Ts. Dr. Law Foong Li **Supervisor's signature:**

Items for discussion (noted by student before mandatory supervisory meeting):

1. Minor Revision for Overall Document
2. Revision for Chapter 3 and 4
3.

Record of discussion (noted by student during mandatory supervisory meeting):

1. Chapter 1: Abstract and Overview of the IR
2. Chapter 2: Minor Revision from the 2nd Meeting
3. Chapter 3 and 4: Revision on overall explanation

Action List (to be attempted or completed by student by the next mandatory supervisory meeting):

1. Fix all the revision and add some explanations to some part.

Note: A student should make an appointment to meet his or her supervisor (via the consultation system) at least ONE (1) week prior to a mandatory supervisor session – please see document on project timelines. In the event a supervisor could not be booked for consultation, the project manager should be informed ONE (1) week prior to the session so that a meeting can be subsequently arranged.

Project Log Sheet

Project Log Sheet No.4

(APU: Serial Number)

PLS V1.0

Project Log Sheet – Supervisory **Session**

Notes on use of the project log sheet:

8. This log sheet is designed for meetings of more than 15 minutes duration, of which there must be at minimum SIX (6) during the course of the project (SIX mandatory supervisory sessions).
9. The student should prepare for the supervisory sessions by deciding which question(s) he or she needs to ask the supervisor and what progress has been made (if any) since the last session, and noting these in the relevant sections of the form, effectively forming an agenda for the session.
10. A log sheet is to be brought by the STUDENT to each supervisory session.
11. The actions by the student (and, perhaps the supervisor), which should be carried out before the next session should be noted briefly in the relevant section of the form.
12. The student should leave a copy (after the session) of the Project Log Sheet with the supervisor and to the administrator at the academic counter. A copy is retained by the student to be filed in the project file.
13. It is recommended that students bring along log sheets of previous meetings together with the project file during each supervisory session.
14. The log sheet is an important deliverable for the project and an important record of a student's organisation and learning experience. The student **must** hand in the log sheets as an appendix of the final year documentation, with sheets dated and numbered consecutively.

Student's name: Jason Yitro Setiadi **Date:** 20 May 2024 **Meeting No:** 1/**Semester** 2

Project title: Aspect-Based Sentiment Analysis on Achieving Sustainable Work Environments Through Recommendation Model on Job Review **Intake:** APD3F2311CS(DA)

Supervisor's name: Ts. Dr. Law Foong Li **Supervisor's signature:** .....

Items for discussion (noted by student before mandatory supervisory meeting):

5. Declaration of Thesis Confidentiality and Library Form
6. Chapter 4 – Data Understanding and Pre-Processing, and Model Buildings
7.
8.

Record of discussion (noted by student during mandatory supervisory meeting):

1. **Vectorization and Tokenization must be done to ensure correct steps taken.**
2. **Dataset Selection and Balancing must be done after Data Pre-Processing.**
3. **Do Confusion Matrix and Heatmap for Testing and Validation, and graphs for Epochs.**
4.

Action List (to be attempted or completed by student by the next mandatory supervisory meeting):

1. **Make sure to follow NLP basic steps, especially Vectorisation and Tokenisation.**
2. **Complete the BERT model until it is connected with LIME, then continue with Traditional model.**
2.
3.

Note: A student should make an appointment to meet his or her supervisor (via the consultation system) at least ONE (1) week prior to a mandatory supervisor session – please see document on project timelines. In the event a supervisor could not be booked for consultation, the project manager should be informed ONE (1) week prior to the session so that a meeting can be subsequently arranged.

Project Log Sheet

Project Log Sheet No.5

(APU: Serial Number)

PLS V1.0

Project Log Sheet – Supervisory **Session**

Notes on use of the project log sheet:

15. This log sheet is designed for meetings of more than 15 minutes duration, of which there must be at minimum SIX (6) during the course of the project (SIX mandatory supervisory sessions).
16. The student should prepare for the supervisory sessions by deciding which question(s) he or she needs to ask the supervisor and what progress has been made (if any) since the last session, and noting these in the relevant sections of the form, effectively forming an agenda for the session.
17. A log sheet is to be brought by the STUDENT to each supervisory session.
18. The actions by the student (and, perhaps the supervisor), which should be carried out before the next session should be noted briefly in the relevant section of the form.
19. The student should leave a copy (after the session) of the Project Log Sheet with the supervisor and to the administrator at the academic counter. A copy is retained by the student to be filed in the project file.
20. It is recommended that students bring along log sheets of previous meetings together with the project file during each supervisory session.
21. The log sheet is an important deliverable for the project and an important record of a student's organisation and learning experience. The student **must** hand in the log sheets as an appendix of the final year documentation, with sheets dated and numbered consecutively.

Student's name: Jason Yitro Setiadi **Date:** July 12, 2024 **Meeting No:** 2/**Semester** 2

Project title: Aspect Based Sentiment Analysis On Achieving Sustainable Work Environments Through Recommendation Model On Job Review **Intake:** APD3F2311CS(DA)

Supervisor's name: Ts. Dr. Law Foong Li **Supervisor's signature:**

Items for discussion (noted by student before mandatory supervisory meeting):

1. Title of FYP
2. Deployment of the Model
3. Final Result of the System

Record of discussion (noted by student during mandatory supervisory meeting):

1. FYP title needs to be updated as it contains some characters that are not needed in the title.
2. Sentiment Analysis Pipeline needs to be changed.

Action List (to be attempted or completed by student by the next mandatory supervisory meeting):

1. Have an average predicted sentiment for each of the company.
2. Print out Sentiment Keywords for each of the headline, then merged together in one line.
3. Export a new dataset that contains: Firm - Other Features - Positive/Negative (Based on Average Predicted Sentiment) – Labeled Keywords.
4. Create a Classification Model similar to Ham/Spam Detection Model.

Note: A student should make an appointment to meet his or her supervisor (via the consultation system) at least ONE (1) week prior to a mandatory supervisor session – please see document on project timelines. In the event a supervisor could not be booked for consultation, the project manager should be informed ONE (1) week prior to the session so that a meeting can be subsequently arranged.

Project Log Sheet

Project Log Sheet No.6

(APU: Serial Number)

PLS V1.0

Project Log Sheet – Supervisory **Session**

Notes on use of the project log sheet:

22. This log sheet is designed for meetings of more than 15 minutes duration, of which there must be at minimum SIX (6) during the course of the project (SIX mandatory supervisory sessions).
23. The student should prepare for the supervisory sessions by deciding which question(s) he or she needs to ask the supervisor and what progress has been made (if any) since the last session, and noting these in the relevant sections of the form, effectively forming an agenda for the session.
24. A log sheet is to be brought by the STUDENT to each supervisory session.
25. The actions by the student (and, perhaps the supervisor), which should be carried out before the next session should be noted briefly in the relevant section of the form.
26. The student should leave a copy (after the session) of the Project Log Sheet with the supervisor and to the administrator at the academic counter. A copy is retained by the student to be filed in the project file.
27. It is recommended that students bring along log sheets of previous meetings together with the project file during each supervisory session.
28. The log sheet is an important deliverable for the project and an important record of a student's organisation and learning experience. The student **must** hand in the log sheets as an appendix of the final year documentation, with sheets dated and numbered consecutively.

Student's name: Jason Yitro Setiadi **Date:** July 26, 2024 **Meeting No:** 3/**Semester** 2

Project title: Aspect Based Sentiment Analysis On Achieving Sustainable Work Environments Through Recommendation Model On Job Review **Intake:** APD3F2311CS(DA)

Supervisor's name: Ts. Dr. Law Foong Li **Supervisor's signature:**

Items for discussion (noted by student before mandatory supervisory meeting):

1. Format of the FYP
2. Contents of the FYP
3. Deployment Result

Record of discussion (noted by student during mandatory supervisory meeting):

1. Add more features for Streamlit Deployment.
2. Discussion of the models can be about the concept of the models on why it has similar result with small gap of accuracies. This limitation can be from time restriction of training, parameters, and even hardware capabilities.
3. Report should be submitted by weekend for checking.

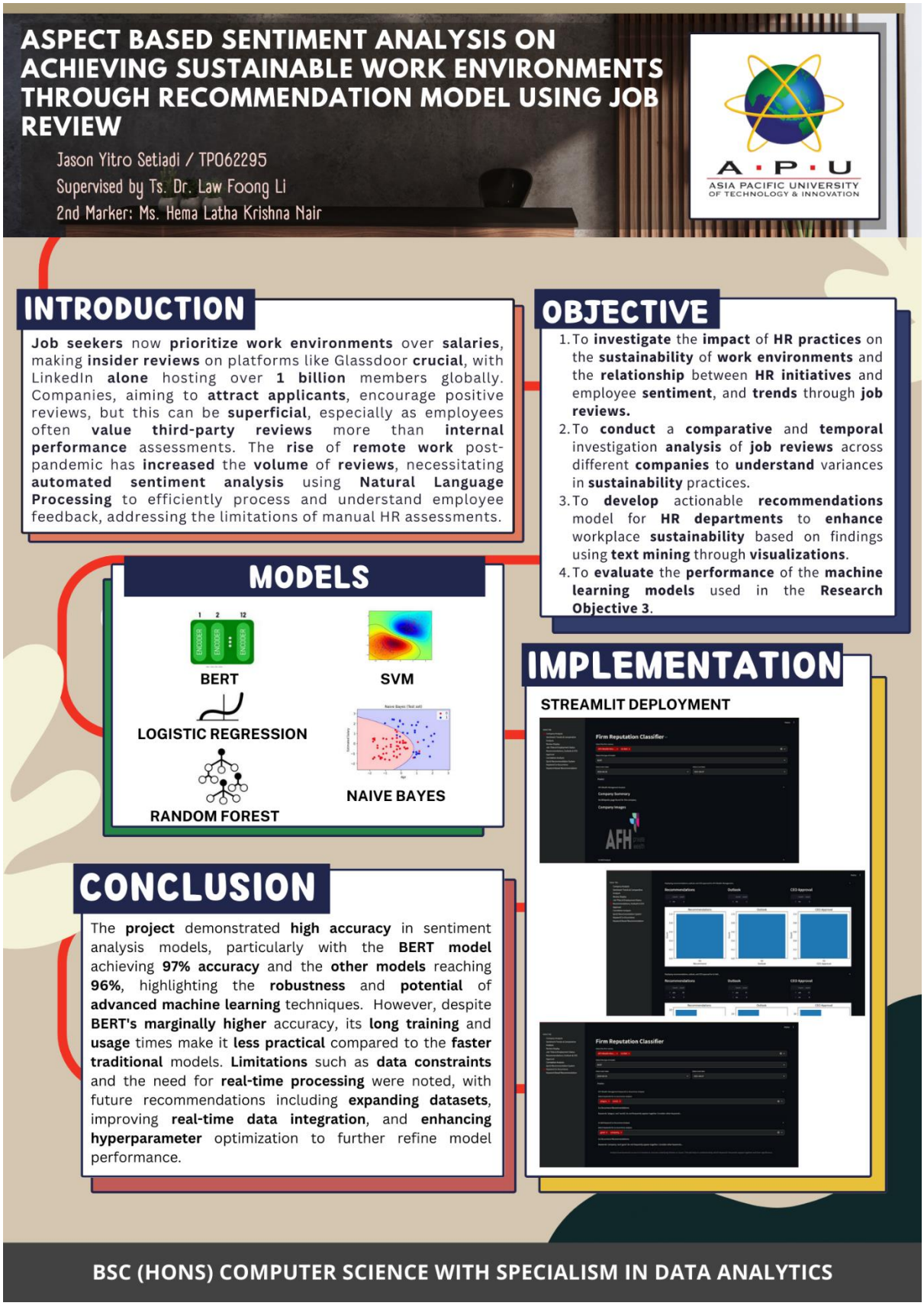
Action List (to be attempted or completed by student by the next mandatory supervisory meeting):

1.
2.

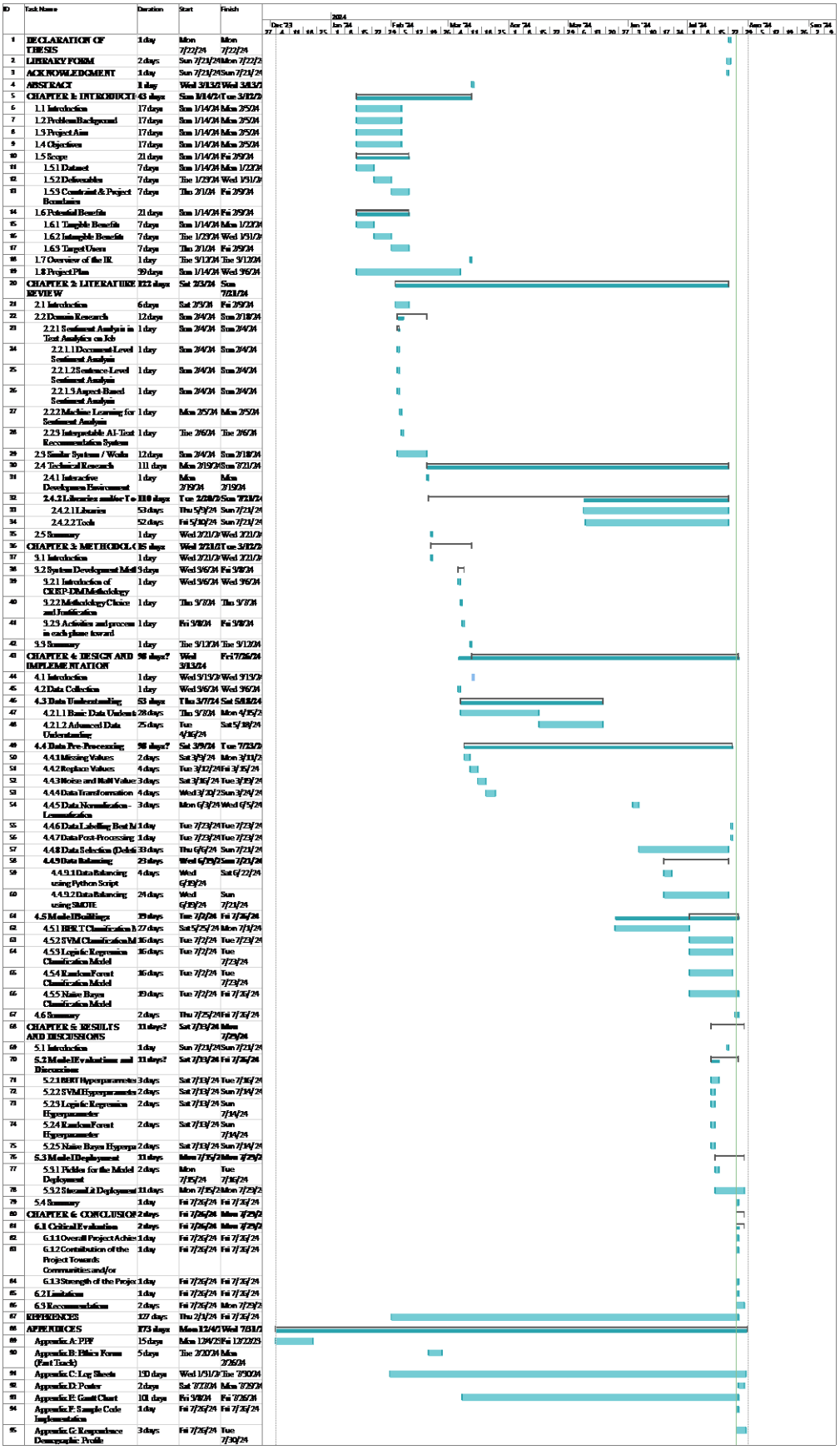
Note: A student should make an appointment to meet his or her supervisor (via the consultation system) at least ONE (1) week prior to a mandatory supervisor session – please see document on project timelines. In the event a supervisor could not be booked for consultation, the project manager should be informed ONE (1) week prior to the session so that a meeting can be subsequently arranged.

Project Log Sheet

Appendix E: Poster



Appendix F: Gantt Chart



Appendix G: Sample Code Implementation

G.1 Sample of Data Transformation

```
# Convert 'df' column to lowercase
df['headline'] = df['headline'].str.lower()
```

G.2 Sample of Lemmatization

```
# Download NLTK data
nltk.download('wordnet')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')

from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet

# Initialize the WordNetLemmatizer
lemmatizer = WordNetLemmatizer()

# Function to get the part of speech tag for lemmatization
def get_wordnet_pos(word):
    """Map POS tag to first character lemmatize() accepts"""
    tag = nltk.pos_tag([word])[0][1][0].upper()
    tag_dict = {"J": wordnet.ADJ,
                "N": wordnet.NOUN,
                "V": wordnet.VERB,
                "R": wordnet.ADV}
    return tag_dict.get(tag, wordnet.NOUN)

def lemmatize_text(text):
    words = text.split()
    lemmatized_words = [lemmatizer.lemmatize(word, get_wordnet_pos(word)) for word in words]
    return ' '.join(lemmatized_words)

# Apply the lemmatization function to 'headline' column
df['headline'] = df['headline'].apply(lemmatize_text)

# Display the DataFrame with lemmatized text
df.head()
```

G.3 Sample of KeyBERT Model

```
import torch
import pandas as pd
from torch.utils.data import DataLoader, Dataset
from transformers import BertTokenizer, BertForSequenceClassification
from keybert import KeyBERT
from tqdm import tqdm  # For progress bar

# Load pre-trained BERT model and tokenizer
tokenizer = BertTokenizer.from_pretrained('nlptown/bert-base-multilingual-uncased-sentiment')
model = BertForSequenceClassification.from_pretrained('nlptown/bert-base-multilingual-uncased-sentiment', num_labels=5)

# Initialize KeyBERT with a BERT model
kw_model = KeyBERT(model='paraphrase-MiniLM-L6-v2')
# kw_model = KeyBERT(model='all-MiniLM-L6-v2')

# Custom dataset class
class SentimentDataset(Dataset):
    def __init__(self, texts, tokenizer, max_length=128):
        self.texts = texts
        self.tokenizer = tokenizer
        self.max_length = max_length

    def __len__(self):
        return len(self.texts)

    def __getitem__(self, idx):
        text = self.texts[idx]
        encoding = self.tokenizer.encode_plus(
            text,
            add_special_tokens=True,
            max_length=self.max_length,
            padding='max_length',
            truncation=True,
            return_tensors='pt'
        )
        input_ids = encoding['input_ids'].squeeze()
        attention_mask = encoding['attention_mask'].squeeze()
        return input_ids, attention_mask, text

# Example DataFrame (assuming `df` is your existing DataFrame)
data = df['headline']
df_subset = pd.DataFrame(data)
```

G.4 Sample of Data Balance Using Undersampling

```
from sklearn.utils import resample

# Balancing Data Using Python
def balance_data(df, target_column):
    balanced_df = pd.DataFrame()
    min_class_size = df[target_column].value_counts().min()
    for class_value in df[target_column].unique():
        class_subset = df[df[target_column] == class_value]
        if len(class_subset) > 0:
            balanced_class_subset = resample(class_subset,
                                             replace=True,
                                             n_samples=min_class_size,
                                             random_state=42)
            balanced_df = pd.concat([balanced_df, balanced_class_subset])
    return balanced_df.sample(frac=1).reset_index(drop=True)  # Shuffle the balanced dataframe

# Balance the dataset based on 'label'
df_subset = balance_data(df_subset, 'label')
```

G.5 Sample of Data Balance using Oversampling

```
from imblearn.over_sampling import SMOTE
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split

# Define X (features) and y (target)
# X = df_subset['pros'] + ' ' + df_subset['cons']  # Concatenate pros and cons for text input
X = df_subset['headline']
y = df_subset['label']

# Vectorize the text data
vectorizer = TfidfVectorizer(max_features=5000)  # Adjust max_features as needed
X_vec = vectorizer.fit_transform(X)

# Convert string labels to numerical categories
label_mapping = {"bad": 0, "good": 1}
y = y.map(label_mapping)

# Split the data into train and test sets (80-20 split)
X_train, X_test, y_train, y_test = train_test_split(X_vec, y, test_size=0.2, random_state=42)

# Further split train into train and validation sets (80-20 split of the 80%)
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.2, random_state=42)

# Display the shape of each set
print("Train set shape:", X_train.shape, y_train.shape)
print("Validation set shape:", X_val.shape, y_val.shape)
print("Test set shape:", X_test.shape, y_test.shape)

# Apply SMOTE for balancing on the training set only
smote = SMOTE(random_state=42)
X_train_balanced, y_train_balanced = smote.fit_resample(X_train, y_train)

# Display the class distribution before and after balancing
print("Class distribution before SMOTE:")
print(y_train.value_counts())

print("\nClass distribution after SMOTE:")
print(pd.Series(y_train_balanced).value_counts())
```

G.6 Sample of BERT Classification Model

```
import torch
import pandas as pd
from torch.utils.data import DataLoader, Dataset
from sklearn.preprocessing import LabelEncoder
from transformers import BertForSequenceClassification, BertTokenizer, get_linear_schedule_with_warmup
from torch.optim import AdamW

# Load pre-trained BERT model and tokenizer for sequence classification
model = BertForSequenceClassification.from_pretrained('bert-base-uncased', num_labels=2)
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

# Custom dataset class
class SentimentDataset(Dataset):
    def __init__(self, texts, labels, tokenizer, max_length=128):
        self.texts = texts
        self.labels = labels
        self.tokenizer = tokenizer
        self.max_length = max_length
        self.encodings = self.tokenizer(texts, truncation=True, padding=True, max_length=self.max_length, return_tensors="pt")

    def __len__(self):
        return len(self.texts)

    def __getitem__(self, idx):
        item = {key: val[idx] for key, val in self.encodings.items()}
        item['labels'] = torch.tensor(self.labels[idx])
        return item

X = df_subset['headline'].tolist()
y = df_subset['label'].tolist()  # Replace with your target column

# Use LabelEncoder to convert string labels to numerical categories
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)
```