Study of Mobile App Rivals: Harnessing Machine Learning to Decode User Reviews in the Mobile App Landscape

Swetha N. K. Sivakumar, Smeet Sheth, Bhavik Patel, and Kashish Thakur

Department of Applied Data Science, San Jose State University

DATA 270: Data Analytics Processes

Dr. Eduardo Chan

December 8, 2023

Abstract

User reviews on the app are of utmost importance as they can be utilized for accurate data-driven decision-making by upcoming users and developers to make their app stand out in the cut-throat market. The existing work majorly focuses on a single domain while analyzing the user sentiment about a particular application with limited attention given to multiple domains. An in-depth analysis of the competitive mobile app landscape across different domains is the need of the hour with the drastic increase in the mobile application market and the consumers that rely on them. The current study aims to suggest an ideal app for new users who are juggling between different Travel and Lodging apps based on their requirements by adequately analyzing the sentiments of the current users about those apps. It involves using different Machine learning algorithms such as Bi-LSTM, BERT, XLNet, and SVM that are used to perform sentiment analysis on the review data collected from 20 different mobile applications from the Google Play Store. App Functionality, Customer Service, User-Friendly, and Payment Experience are the four major topics identified using the LDA technique in Topic Modeling. The models are assessed using performance metrics like Accuracy, F1-Score, Recall, and Precision. Based on the Model Assessment, the BERT model reliably recognizes the user's sentiment and offers insightful app comparisons with 92.32% accuracy. The results achieved can save a lot of downtime for the new users in identifying the app of their choice.

Introduction

Project Background and Executive Summary

Project Background

The market for mobile applications has grown significantly. To capture the interest of the customer, each product has a specific application. According to Jin et al. (2016), the customer has the choice to pick their preferences between multiple apps. Today's activities require mobile applications more than ever before. Every mobile user has downloaded, on average, 80 programs, and people are most inclined to recommend an app that they have enjoyed using. Many people place a lot of weight on customer reviews or ratings before selecting any mobile application. Even though major app stores do provide the most recommended app to their customers based on their requirements, there is still insufficient information for the user to decide before the customer chooses their app. It is very difficult for any mobile app to sustain itself without having a thorough idea of its customer base as suggested by (Guo et al., 2017).

Needs and Importance of the Project

From social networking to business, the market for mobile apps is expanding. Users frequently find it difficult to decide amongst the many apps that have similar objectives. By examining user ratings and feedback, this study simplifies the decision process and directs consumers to the most suitable app for what they are looking for. To track market trends, Ouyang et al. (2019) created an evaluation system for fitness professionals that makes use of machine learning, big data, and visualization. With an accuracy rate of 81.1%, they examined and categorized fitness applications using information from 1,381 apps and more than 100,000 user reviews. Users can choose apps with greater knowledge thanks to their strategy, which makes use

of reviews and ratings. However, future study needs to be improved upon due to potential biases from extreme opinions in the comments.

Target Problem

The problem statement for the proposed research is that given the rapid expansion of mobile applications in the Travel and Lodging industry, it is important to understand consumer choices and views through an extensive examination of user feedback and rating systems.

Motivation and Goal of the Project

With the majority of consumers reading reviews before installing, mobile apps act as brand gateways. Positive criticism improves sales, earnings, and client retention. The objective of the study is to undertake a fair assessment of various competing Travel and Lodging apps. In the project goal, sentiment analysis of app reviews is done using machine learning and deep learning techniques, specifically for booking applications. From unfavorable feedback, the best qualities that indicate areas that need development are picked out. The research results can help users choose apps and assist developers in filling up any gaps and remaining competitive.

Project Approaches and Methods

The Android Play Store's user reviews of various smartphone applications serve as the primary dataset underlying the current study. The Google Play Store's integrated Python Web Scraper (Google Play Scraper) is used to collect the initial data. Sentimental study and other preprocessing techniques in Python are used to preprocess the information gathered to fix any significant errors and narrow down the list of applications with sufficient review data for the study. The Latent Dirichlet Allocation (LDA) technique is used to classify the customer evaluations into the main groups that are frequently seen in the review data that has been

gathered for each App. Utilizing sentiment analysis, each key category of evaluation is divided into those that are favorable, unfavorable, and neutral. Four machine learning techniques, including supervised machine learning models like Support Vector Machine(SVM), a word embedding model like Bi-directional Long Short-Term Memory, and a transformer-based model like Bidirectional Encoder Representations from Transformers (BERT), and Extra Long Transformer (XLNet), are used to analyze and measure the overall Sentiment Score for each Mobile APP that is shortlisted. A holdout validation technique is used to validate the final data. The Model is assessed with the help of different performance metrics such as precision, accuracy, f1 score, confusion matrix, and recall.

Project Contributions and Applications

The Sentiment category that this study yielded is used to determine how users feel about a certain mobile application. Using methods, such as app rankings and net downloads, the model inconsistencies are determined and contrasted with the derived sentiment score. The visualization of data is used to show the differences. Instead of toggling between numerous applications to compare costs and services, customers can quickly identify their needs using such models and move right to the tool that meets those needs.

Project Requirements

Functional Requirements

The project aims to perform sentiment analysis on the mobile application reviews and then implement the competitive analysis between different applications. The major functional requirement is to apply appropriate Machine Learning models to perform an accurate and efficient analysis. The project involves different deep learning and supervised learning

algorithms. Each Machine Learning (ML) model requires proper computational resources to effectively perform modeling. The models require powerful computational power in terms of the Central Processing Unit(CPU), and Graphical Process Unit(GPU) for the proper training and testing model. The appropriate amount of data for the modeling should be preprocessed and cleaned. Any imbalance or discrepancy in data adversely affects the model performance, which is another important functional requirement for the project. The management of the data in terms of storage is necessary, which needs faster storage and sufficient memory space. The data processing needs sufficient Random Access Memory(RAM) capacity to fetch and retrieve the data quickly and efficiently. The raw data needs to be cleaned and preprocessed which needs an appropriate Integrated Development Environment(IDE) environment to execute the necessary code for data transformation and to complete the preparation of data.

AI-Powered Requirements

Natural Language Processing (NLP) has introduced many tools and techniques to analyze user sentiment from text data. The most commonly used techniques like tokenization, and lemmatization help in dissecting the sentences into granular components. Machine Learning and pre-trained transformer-based models are trained to understand the underlying patterns and classify the opinions as positive, negative, or neutral categories. SVM and Bi-LSTM have shown great performance in classification problems. The pretrained models such as BERT and XLNet have demonstrated the state-of-the-art benchmark in Natural Language processing tasks.These models are computationally intense due to their complex architecture. To train and test these models, substantial GPU memory is required. The memory requirements vary with the batch size utilized in the training, validation, and testing phases. To avoid out-of-memory errors, efficient

memory management techniques such as gradient accumulation or gradient checkpoints are employed during the training loops. In terms of computational powers, both the large-scale pre-trained models are resource-consuming and time-consuming. For faster convergence, power CPU and GPUs are required.

The evaluation metrics such as Accuracy, F1-score, precision, confusion matrix, and recall are used to quantify the AI requirements. Accuracy determines the model's effectiveness in predicting the correct sentiment and F1- score can be implemented to evaluate the model's ability to observe the sentiment from the user review information. Precision and Recall are important metrics to notice the imbalance in the distribution of the sentiment categories. Training time also plays a pivotal role in analyzing the model performance. The overall successful implementation of this research depends on the computing resources, optimization techniques, and careful consideration of the memory-efficient strategies to handle the demand.

Data Requirements

The project majorly focuses on the Mobile Application domain and performing Sentimental Analysis and Competitive Analysis on different applications. This requires an appropriate amount of text data which should be cleaned and preprocessed to give optimal results. The cleaning and preprocessing of text data needs different techniques provided by the Natural Language Toolkit (NLTK). This involves techniques such as Tokenization, Stemming, and Lemmatization etc. The text data also needs to be transformed according to the specific model requirements.

For the competitive analysis, there is a requirement for topic modeling approaches, which can identify common themes in the data. Once the data is cleaned and preprocessed, the data

needs to be prepared for model building. This involves the splitting of data into a Train, Test, and Validation set. The split ratio for the different datasets is to be decided which can be either based on the standard 80:10:10 split or the based on the model requirement. The data once prepared is ready for the model development.

Project Deliverables

A Hybrid Agile Approach is used for the project development. The project development follows the Cross-Industry Standard Process for Data Mining(Crisp DM) methodology, which is divided into six development phases- Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, and Deployment. The project consists of listed deliverables that need to be completed for the successful completion of the project:

Project Proposal

The proposal highlights the initial key aspects of the project before starting with it. It includes a brief description of the Proposed Problem, the Goals, and the Motivation behind the Project. The initial sources of data explored and collected for the research followed by the summary of research study materials are provided as a part of the proposal.

Project Planning

The project planning is carried out on the Jira tool, which follows a particular hierarchy for project management i.e., Epics, User Stories, and Tasks, and work is carried out in different Sprints. Each Sprint has a planned number of days. The current project hierarchy consists of six Sprints, where each Sprint lasts for two weeks. The project plan includes six Epics, 32 User Stories, and further sub-tasks. For the Project Scheduling, Work Breakdown Structure, Gantt Chart, and Pert Chart are created using different Jira plugins.

Work Breakdown Structure. It is a project management component that is used to create a hierarchy of the list of tasks to be done in a project. In the project, the Jira plugin named Move and Organize is used to create a Work Breakdown Structure, which consists of the main tasks of the six Epics, and these are further subdivided into respective User Stories and Sub Tasks.

Gantt Chart. It is a bar chart representation of the list of tasks to be performed in the project which depicts the tasks' start date, end date, duration, people assigned to each task, and the dependencies between each task. The project uses WBS Gantt - Chart as the Jira plugin to build the Gantt chart depicting the timelines of the tasks and the resources involved in them.

Pert Chart. It is a tool to visualize the timeline of the whole project, which includes an overview of the high-level tasks and the Critical path for the project, which is a list of high-priority tasks required to complete the project. For the project, a Pert Chart is created in the Draw.io tool where a sequence of high-level tasks present in the project can be visualized along with the identified Critical Path.

Data Collection

Collecting the required data from Google Play Store using the Python Script utilizing web scraping to access app-related information such as name, description, user reviews, and ratings. This involves extraction of 20 Comma Separated Values (CSV) files including the information about the User reviews on the mobile application.

Prototype

A prototype solution is developed for analyzing user sentiments and preferences while using an application. The prototype is built based on machine learning and Natural language processing to perform sentiment analysis and identify and categorize the reviews.

Data Engineering

The extracted raw data is text data of the user reviews. It is cleaned and preprocessed to remove any outliers or missing information. The data cleaning is performed using the Python data frames. The review content is processed using different Natural Language Processing techniques such as Tokenization, Stemming, Stopword Removal, and Lemmatization. The final cleaned data is split into Train, Test, and Validation in 80:10:10 ratio, which is fed to the Machine Learning models.

Model Development & Testing

The four Machine Learning models are supervised ML models like Support Vector Machine(SVM) Algorithm, word embedding models like CNN Bi-directional Long short-term memory (Bi-LSTM); transformer-based models like Bidirectional Encoder Representations from Transformers (BERT), Extra Long Transformer (XLNet), is used to identify the sentiments of the user reviews and most preferred features of the application.

Performance Evaluation

The models created for sentiment analysis are thoroughly assessed during the testing phase. The total accuracy of the model's predictions is gauged by the Accuracy metric. Recall determines how well the model identifies all pertinent entities or feelings in the data. Precision evaluates the model's accuracy in correctly detecting entities or sentiments, hence lowering the

likelihood of false positives. The harmonic mean of precision and recall, or the F1 score, provides a fair evaluation of model performance. Together, these measures make sure that the models are capable of decreasing errors and false identifications, in addition to delivering accurate forecasts.

Optimal Model Selection

The final model is a comprehensive one-stop solution to the sentimental analysis of mobile application reviews. It uses state-of-the-art machine learning and natural language processing methods to detect significant entities in user reviews and ratings as well as customer reviews and ratings.

Individual Research Paper on Model Development

The Individual research paper explores one Machine Learning approach to enhance the accuracy of sentiment classification resulting in precise solutions. As a part of the project deliverable, all four group members have written an Individual Research Paper with around 3000+ words following a proper American Psychological Association(APA) format. Each research paper covers 1-2 different Machine Learning Models that are used for the project. The paper covers all the details as to how the used model aligns with the scope of the project, what were the challenges and outcomes faced while deploying the model and most importantly what is the accuracy of that particular model and what steps needed to modify or improve that as a part of the future scope.

Project Report

The project report includes a summary of the project's objectives, a discussion of the strategies used to achieve them, an analysis of the difficulties encountered, and an analysis of

the results obtained. In addition to giving an overview, it highlights the relevant discoveries. Table 1 below lists the project deliverables and their corresponding due dates.

Table 1

Project Deliverables and the Due Dates

Deliverable	Due date
Project Proposal	09/22/2023
Data Collection	09/20/2023
WBS	09/29/2023
Gantt chart	10/06/2023
Pert Chart	10/13/2023
Data Engineering	10/15/2023
Model Development and Testing	10/19/2023
Performance Evaluation	10/31/2023
Optimal Model Selection	11/08/2023
Individual Research Paper	12/06/2023
Final Report	12/08/2023

Note. The above table shows the list of deliverables for the project along with their timeline.

Technology and Solution Survey

Survey of Current Technologies

The study conducted by Borg and Boldt (2020) focuses on the aspect of customer support given to organizations and has majorly focused on a Swedish Telecom Corporation. The data collected from 168,010 emails consists of 69,900 conversation threads of information. The

fetched data does not provide any information about the Sentiment. A sentiment analyzer tool called Valence Aware Dictionary and Sentiment Reasoner (VADER) is utilized along with a Swedish Sentiment Lexicon to provide the preliminary Labels to the Email data. The study used two Support Vector Machine(SVM) models for sentiment classification, with the email content and the curated sentiment labels as input parameters. It also performs the investigation of predicting the sentiment of future email responses. The results show that the Linear SVM model gives an f1- score value of 0.834 and a mean Area Under Curve(AUC) value of 0.896 for the classification task. With the help of the achieved results, it is possible to predict the sentiment of the email responses that still need to be acquired. The results can also be gainful in providing feedback to the future Customer Sentiment to the corporation's Customer Support.

An investigation on Sentimental Analysis is performed by Al-Amrani et al. (2018) using the techniques of text search using Amazon Product Review data. The product review data has an additional label known as the Sentiment Label. The value of the sentiment label is a binary value that can be either positive or negative feelings based on the different aspects mentioned in the designed query. The study incorporates three supervised ML algorithms: Random Forest algorithm, SVM algorithm, and a hybrid algorithm created using Random Forest and SVM known as RFSVM. The hybrid approach shows the best performance compared to the other ML approach. The RF SVM hybrid model gives an accuracy of 83.4% with the least mode-building time, and RF and SVM show an accuracy of 81% and 82.4%. The enhanced performance of the hybrid approach is due to the utilization of advantages of the two individual ML algorithms, Random Forest and SVM.

Gong et al.(2019) developed a Broad Autoregressive Language Model (BroXLNet) that automatically processes sentiment analysis by integrating the advantages of autoregressive Language Modeling and extracting the deep contextual features using board learning systems. The binary Stanford Sentiment Treebank dataset, which comprises reviews and human remarks on the sentiment, was used to assess the architecture. The framework of BroXLNet starts with Lexicon Encoding the input. The bidirectional context and relative positional encoding schemes are captured using the XLNet model. Finally, the extra features in the input space are extracted using the broad learning system. The experimental results of the BroXLNet were compared with other state-of-the-art methods such as the Continuous Bag-of-Word Model(CBOW), Bidirectional long short-term memory(BiLSTM), GenSen, Embeddings from Language Models (ELMo)and OpenAI Generative Pretrained Transformer (GPT). BroXLNet achieved 94% accuracy on sentiment analysis tasks by extracting more features from the unlabeled corpus proving its excellent classification ability.

Khan et. al(2023) performed research to understand the sentiments of customer reviews using Pre-trained Language Models such as BERT, XLNet, and Electra. In the training and testing phase of pre-trained models, the publicly available Twitter dataset consisting of 1.6 million tweets with positive, negative, and neutral sentiments was utilized. Without fine-tuning the BERT, XLNet, and Electra models achieved 87.6 %, 88.2%, and 87.9% accuracy respectively. After fine-tuning the models to adapt to the sentiment analysis, the models achieved a high accuracy of 89.2% for BERT, 92.3% for XLNet, and 90.1% for Electra in classifying customer reviews.

Bidirectional Representations from Transformers (BERT) was proposed in a paper by

Devlin (2018). Because BERT takes into account both the left and right context in every layer, it is intended to pre-train deep bidirectional representations from unlabeled text. The two primary parts of the model are fine-tuning and pre-training. Two model sizes—BERT-Base (110M parameters) and BERT-Large (340M parameters)—that vary in terms of the number of transformer blocks, hidden size, and self-attention heads are the main subjects of the study. In the pre-training phase, 15% of WordPiece tokens are randomly masked using Masked Language Modeling (MLM), and sentence associations are established using Next Sentence Prediction (NSP). Two datasets—English Wikipedia (2,500M words) and BookCorpus (800M words)—are used to train the model. To distinguish between tokens belonging to distinct phrases, BERT makes use of self-attention, a technique also seen in the Transformer model. To separate the tokens, a sentence's initial token is retained as the classifier token (CLS) and its last token is the separator token (SEP). The model attains an average accuracy of 79.6% on BERT-Base and 82.1% on BERT-Large after fine-tuning with three epochs and choosing the optimal learning rate among 5e-5, 4e-5, 3e-5, and 2e-5 on the development set. The model also achieves high results such as 86.7% on MultiNLI, 93.2% on SQuAD v1.1 question answering Test F1, 83.1% on SQuAD v2.0 Test F1, and 80.5% on the General Language Understanding Evaluation (GLUE) benchmark.

The authors of a study by Pota et al. (2020) suggested a unique method for handling tweeter jargon in sentiment analysis on Twitter. They use language-independent or readily adjustable processes to convert Twitter jargon, including emoticons and emojis, into plain text in the first stage of their method. They then categorize the generated tweets using the BERT language model. The study used the ekphrasis tool from GitHub to normalize the raw tweet data

by validating context in multiple formats, including date, time, email, money, number, percentage, phone, URL, @user, hash-tagged content, emoji-to-word, and uppercase letters. The raw tweet data was obtained using the Twitter API. The following hyperparameters were set on the Twitter dataset for the BERT model used in the study: 12 attention heads, eight batch sizes, five epochs, 16 gradient accumulation steps, 768 hidden sizes, 12 hidden layers, 0.00003 learning rate, 128 maximum sequence length, and 110M parameters. The preprocessing methods were the same as those employed in the work by Devlin et al. (2018). The authors calculated individual classification F1 scores and achieved 73.81% for positive classification, 76.20% for negative classification, and 75% for neutral classification.

Understanding the importance of customer feedback Bhuvaneshwari et al. (2022) published a paper where they introduced a Bi-LSTM Self-attention based Convolutional Neural Network (BAC) for the prediction of the exact polarity of the review. The paper used an Amazon product data set and focused on the opinions of the users and the impact it has on other users while buying a similar product. The proposed model recorded an accuracy of 89% and F1 score of 91%. For future work, they are planning to add stickers and memes in the review as well to check the model's performance in handling text as well as image data.

ANN methodology is utilized by multiple authors, where Sousa et al. (2019) use Bidirectional Encoder Representations (BERT) to analyze the sentiment analysis of the news data and this works for a short collection period of the news. In this study related to sentiment analysis, rule-based machine learning models were frequently used but the development of convolutional neural networks (CNN) and recurrent neural networks (RNNs) showed significant results making them the pioneers of deep learning for sentiment classification. (Habbat et al.,

2021) also used the ANN model in conjunction with XLnet, Multifit, and CamemBERT to represent text data, followed by the classification models CNN, BiLSTM, and GRU. It is discovered that the XLNet works best in word embedding, reaching 96.5% accuracy with the FACR dataset.

Comparison of Solutions

Table 2 below shows the comparison of the research paper explored during the technical summary.

Table 2

Comparison of Solutions

Authors	Dataset	Models	Results
Borg and Boldt (2020)	Swedish Telecom Corporation email threads	VADER, SVM	F1- score: 0.834, AUC: 0.896
Al-Amrani et al. (2018)	Amazon Product Review dataset	Random Forest(RF), SVM, Hybrid Approach RFSVM	Accuracy of RF SVM: 83.4%, RF: 81%, and SVM: 82.4%
Gong et al.(2019)	Standford Sentiment Treebank	BroXLNet, CBOW, ELMo, Bi-LSTM, GenSen, GPT	Accuracies of the models, BroXLNet: 94%, CBOW: 80%, ELMo: 90.4%, Bi-LSTM: 82.8%, GenSen: 83.1%, GPT: 91.3%

Authors	Dataset	Models	Results
Khan et al(2023)	Twitter Dataset	BERT, XLNet, Electra	Accuracies of the model BERT:89.2%, XLNet:92.3%, Electra:90.1%
Devlin (2018)	English Wikipedia (2,500M words) and BookCorpus (800M words)	BERT-Base (110M parameters) and BERT-Large (340M parameters)	BERT-Large gives Accuracy: 82.1% , BERT-Base gives Accuracy: 79.6%
Pota et al. (2020)	Twitter data	BERT	F1-score for positive classification: 73.81%, F1-Score for negative classification: 76.20%, and F1-Score for neutral classification75%
Bhuvaneshwari et al. (2022)	Amazon product data set	Bi-LSTM Self-attention based Convolutional Neural Network (BAC)	Accuracy: 89% and F1-score: 91%
Sousa et al. (2019)	News dataset	XLnet	Accuracy: 96.5%

Note. The table above shows the comparative analysis of the Technical research paper.

Literature Survey of Existing Search

Summary of Literature Survey

A Deep learning analysis approach is proposed by Assi et al. (2021) called FeatCompare. The review data is collected from 2,000 freely downloadable mobile apps on the Google Play Store using a Web Crawler named Akdeniz. In total, they have collected about 14,043,999 customer reviews from different popular Mobile APPs. A comparative study is then conducted using the customer reviews of the identified features between different rival apps. A novel ANN model which is Global Local Sensitive Feature Extractor(GLFE) is used to fetch the highest significance feature from a set of customer reviews available on the mobile apps. The research is conducted on the captured 3.4 million customer reviews gathered from about 196 apps falling into similar categories in a list of 20 different app groups. The Model approach is evaluated against 480 randomly chosen customer reviews within five different app groups. The authors have also conducted about 107 surveys with the mobile developers to analyze how the developers perform the competitive analysis on their end. This survey helps the authors to understand how competitive analysis is carried out in practice. The metrics like Precision, Recall, F1-Score, and Accuracy are measured to assess the performance. According to the Evaluation Criteria, they created three different models named GLFE-25th, GLFE-50th, and GLFE-75th, and for each model, they measured the True Positive, False Positive, and consecutively using these values there have calculated the Precision, Recall, and F1-Score values for each model. It is found that for their GLFE-25th the Precision is 0.81, F1-Score is 0.78, Recall is 0.74, for GLFE-50th, the Precision is 0.70, F1-Score is 0.67, Recall is 0.64, and for GLFE-75th, Precision is 0.32, F1-Score is 0.29, Recall is 0.31. This model has displayed better performance when

compared with the other models. The study is expected to extend by the addition of more different categories of mobile applications which can help improve the proposed model and include more test cases to evaluate their employed approach. It also aims to incorporate and analyze how the different highly significant features advance with each subsequent release.

The main focus of the research article published by Dalpiaz et al. (2019) is user feedback analysis, and the authors have developed a tool called "RE-SWOT" to evaluate customer input based on NLP methodology and forecast the SWOT matrix of requirements engineering. The App Store review data dataset was utilized as the dataset. The information is also visualized to better understand the insights from the result of the NLP algorithm. The developed tool has been demonstrated to the Product Managers for further opinions. The various features of the apps are identified from the user review using tokenization and lemmatization to find the common term. Then the Feature performance score (FPS) of the app is calculated. Subsequently, the generated SWOT matrix relies on the FPS score to classify as positive, negative, and neutral. The visualization on the tool displays the features that are common, unique to the competition, and unique to the app. A detailed overview of the feature can also be requested for future analysis. The tool was assessed utilizing the interview protocol, and the three members of the product management team were shown the product for multiple iterations of analysis. From the analysis, the developers, managers, and CEO read the user reviews once per month or occasionally through the Google Play developer's console. Thus, RE-SWOT has detailed analytics and is easy to use for updating themselves with the latest features of the competing apps. The future scope of these projects leads to the improvement of the feature selection algorithm and additionally, the research can extend to developing analytics tools for developers’ analysis.

In the given research, Gao et al. (2015) have worked on a study where the app developers go through extensive bug-fixing and version update processes in response to customer feedback. This study creates a Prioritizing App Issues for Developers (PAID) framework that app developers may use to prioritize app bugs. They have gathered millions of customer reviews for 37 apps through data crawling. The raw reviews are used in app issues Phase generation which is achieved using True Mutual Information (TMI), Information assurance, and Quality Assurance. They employed the Dynamic Latent Dirichlet Allocation (LDA) modeling technique to capture the timeframes of the user reviews and divided the reviews into subjects for each version issued. The impact score is created considering the semantic aspect (KL-Divergence), and the sentiment aspect for topic interpretation of the phases, and the highest score of the phase is presented to the app developers for corrections in the following release. The transformation of app challenges has been displayed in ThemeRiver for easier comprehension. They evaluated the results of the PAID framework using the official APK4Fun changelog. obtaining an accuracy of 60.3%. A further stage of this research endeavor will examine the discrepancy between the developers' decisions and the user reviews. In the aspect of implementation, applying NLP methods to linguistic rules to extract valuable features from short reviews. To assess the accuracy and precision of the PAID architecture, the framework can be tested by extending the data source to the App Store and Amazon Appstore reviews. The study is mostly concerned with phase research, which could not be successful when utilized with game apps because users often abandon them after a brief period.

In Gao et al. (2017) research paper, the goal of their work is to give fitness operational professionals an evaluation system that integrates machine learning, Big Data, and data

visualization for monitoring a company's market status and adversaries in real-time at a greater resolution and cheaper cost than more typical comparative analysis methods. They collected data from 1,381 mobile applications, 100,892 customer feedback, and 95,705 Google+ user profiles. They also implemented big data technologies to gather structured and unstructured data to analyze content automatically. The Binary Naïve Bayes algorithm was used to differentiate fitness apps and non-fitness apps. After that LDA Topic Modeling, Multidimensional Scaling, and K-Nearest Neighbors Clustering are performed on a fitness app to classify 10 functional categories and four clusters (sports activity, you guide, women's health, motivation exercise video) including viable rivals. They evaluated the model by checking accuracy which is 80.1%, precision is 86.3%, recall is 79.1% and f-score is 82.5%. Future research could include historical data to evaluate a brand's performance over time. The collection of Google Plus suggestions and feedback from users may generate self-selection discrimination, favoring people with solid points of view or consumer expertise while underscoring indifferent or apathetic consumers. This can lead to heated reactions, with extreme viewpoints leading app rankings. Future research could concentrate on sophisticated algorithms to reduce this bias.

The study of competitive analysis as conducted by Jin et al. (2016) proposes the formulation of an Optimization problem. They have incorporated three greedy algorithms to analyze their test cases to find the most successful approach. The study majorly considers the customer’s online product reviews. The main aim is to find a couple of customer reviews characterized into different categories i.e., Representative, Comparative, and diversified. These reviews are then analyzed so that the study can evaluate the proposed performance metrics. They have termed data as “Opinion Data” and are collected from Amazon.com. Firstly, they have

created a framework that is used for extracting the couple of reviews that fulfill the expected criteria. The initial step in the Framework is to perform Part of Speech(POS) tagging, which is done using Stanford POS tagger. It then utilizes two different supervised learning techniques Sentimental analysis and Product Feature Extraction to segregate the customer reviews into different categories(positive, negative, and objective). They have united positive and negative reviews and termed them as subjective reviews. The segregated objective reviews are also expected to provide customer concerns, but these types of reviews are avoided in this research. They have captured the major topics covered by the customer through their reviews with the help of Topic Analysis. Along with the techniques of Sentimental Analysis and Product Extraction to capture the features of the mobile apps, and to determine the polarity levels. Three different performance metrics are used, which are Comparison Metrics which consider the extent of similarity between the selected couple of reviews, and Information representation which considers the extent to which the couple reviews can wrap the data provided in the source. Metrics and information diversification Metrics consider the extent of dissimilarity between the selected couple of reviews. The results are evaluated using three optimization approaches termed C-First, D-First, and R-First Approaches, where "C", "R" and "D" denote the information comparativeness, information representativeness, and information diversity. The D-first strategy was shown to be superior to the remaining two approaches, according to the findings. It can pick a few sentences that cover a range of subjects. The consumer's worries regarding a specific feature of a single product that does not offer the same sort of review for its competition did not constitute the subject of the study in question. This can be considered an aspect for study in further development.

Lee and Raghu (2014) conducted a study to analyze and understand the factors that contribute to an app's success by maintaining its position in the top-grossing chart in Apple's App Store. The dataset was collected weekly for a total of 39 weeks from the Apple App Store (from Dec 2010 – Sept 2011). Apps were categorized into paid, free, and top-grossing. Survival Analysis was calculated on a continuous scale by measuring the entry and exit of the app in the top 300 charts. Apps before and at the end of the study were dropped to avoid any bias. The final dataset consisted of 7579 apps provided by 3882 sellers. A generalized Hierarchical Linear Model(GHLM) was used to know about the relationship between the portfolio management strategy and the overall sales performance of an individual app. The Cox Hazard Model was used to analyze the survival time chart of the app. The Count Regression Model was used to review the results of GHLM and check how one seller's app affects the other apps in the chart. Based on the study, factors such as Sales performance, Price, Category, Regular Updates, and high user reviews were considered impactful for the app's survival rate in the chart. It shows that the proposed method gives an accuracy of 90.20% for a Samsung TV, an LG TV gives an accuracy of 91.90%, and an accuracy of 91.73% for a Sony TV. Further research covers the link between product portfolio decisions and app performance and discusses other alternative factors that might impact the app's rating and its position on the chart.

The study conducted by Y. Lee (2021) compares elements that highlight how products differ from one another using explainable neural networks. When a buyer is thinking about purchasing a product, they are primarily concerned with the feature that sets that product apart from its rivals. Based on a survey, they compared similarities and determined the most helpful product features.: As the main dataset, LG Electronics' survey results and the TV reviews dataset

were used. The customer review is first used to perform word embedding tasks where they have used pre-trained word embedding Amazon review data for identifying linguistic features and used part of speech tags of the Stanford tagger. After the features are extracted, the weights are assigned by variants of the (Gradient-weighted class activation mapping)Grad-CAM algorithm, and synonymous words are formed clusters using the clustering technique followed by supervised convolutional neural network (CNN), Conditional random fields (CRF), and hidden Markov model (HMM). A normalized discounted cumulative gain (NDCG) was used to calculate the effects of the product characteristics. The research established the success of CNN based approach and the combination of CNN-based approaches such as CRF and HMM worked the best. With the evaluation, their proposed method reflects the interest of the customers and identifies the critical element in the product purchase. The future scope of this research work is expanding the neural network and extending the approach for other products in the market.

The research paper published by C. Z. Liu et al. (2014) solves a couple of questions derived from a Freemium model used in the mobile application sector: the way the downloads of a smartphone app's subscription edition differ between its complementary and paid versions, Do the findings from applying the freemium concept in various merchandise markets remain relevant to the market for mobile apps and the last question is what function the ranking serves in the Play Store application market, encompassing both zero-cost and premium apps. To solve the questions, they used longitudinal data collecting for two months which contained 60,147 instances from 1,567 applications. They discovered that providing a mobile app's users with an opportunity to try it for free is positively associated with increasing sales of the app's commercial version with an accuracy of 85.6%. An interesting finding is that sales of the paid version are not

significantly impacted by the ranking of the trial version. Additionally, less popular applications have more impact on review rating compared to more known applications, which suggests that trial versions of applications reduce the significance of rating. The research could be improved by collecting data from similar categories to compare within-category applications.

This comparative study conducted by Li et al. (2017) addresses the booming challenge of solving mobile application comparison, which is limited comparative resources like average rating, and total downloaders. This study helps downloaders to make an informed decision by fetching five million Google reviews from 2,311 Google applications, which are publicly available, of six different categories. Proving in-depth comparisons like privacy, power usage, response time, etc. by making sure why some users prefer the application over other applications from the same category. They performed this study by identifying sentences – from the same categorical application – using the tf-relevance graph model (measures the importance of a word in a document) by which they managed to identify short forms of application used in user reviews to classify the same categorical application. They have also made use of keyword-based analysis to minimize manual labeling. They performed analysis on 10,000 reviews and were able to achieve 93.7% accuracy. 80% accuracy was achieved on 4,000 distinct topics in the same comparative users' review. This study helped the researchers to come up with a current application preference rate of 82.8%. Extending this research by making a review search engine, the researchers can provide suggestions about the application choice based on the preferences and rival application reviews.

The research paper proposed by Ouyang et al. (2019) focuses on building a framework for CompetitiveBike forecasting the popularity of the bike-sharing apps - Mobike and Ofo. They

have achieved the forecast by comparing i) popular features of the bike-sharing apps and ii) Lags between the apps. Data comprises 11 Android app stores in China for Mobike and Ofo for the year 2016-2017. Microblogging data have been collected from Sina Weibo - a commonly used blogging service in China. Android app store and Microblogging data are classified into app features & and statistics Reviews & and comparable microblogs which are then used for feature extraction. The feature extraction is further categorized into coarse-grained and fine-grained competitive features. Decision Tree, AdaBoost, and Random Forest are used to evaluate the popularity of the app, and Support Vector Regression and Random Forest algorithms are used to evaluate the lags between the Mobike and Ofo apps. For Popularity, Accuracy, Precision, Recall, and F-measure are considered, and for lags or comparable intensity, Root Mean Square Deviation is considered. In comparison with the Popularity, and Lags between the two apps, the Random Forest algorithm predicts the highest accuracy of 71.4% precision. The author of the research paper focuses on enhancing economic theories on competitive analysis and improvises the prediction model by coupling the features and identifying the two-way effect.

The research proposed by Su et al. (2019) focuses on the growing market of mobile apps currently among users. With such high demand, developers are keen to develop the apps with a slight to major change but solving an overall same purpose. In such cut-throat competition focusing on other apps' weaknesses and rectifying it can be of much benefit. The way to solve this problem is by user review analysis. The method they used to address this problem is UISAT (User-review mining via topic Identification, Sentiment Analysis, and Topic matching across apps). First, the user reviews of the apps that were similar and solved the same purpose were taken into consideration by Google Play Store and App Store. A particular type of concern or

topic for all the apps was taken and their reviews were compared. After that sentimental analysis was performed on the user reviews using a rule-based model. A sentimental score was assigned for each topic after summing up the sentimental score for each user review. After the cumulative addition of the positive and negative scores, overall satisfaction, and impact of the apps on the users were found. Based on the calculations using UISAT, a Sentimental score of less than 2.5 scores was considered a negative sentiment polarity whereas a score of more than 2.5 was considered a positive sentiment polarity for user's feedback. True positives, false positives, and false negatives were also calculated to avoid the error. The paper mainly focuses on four Android apps i.e., Messenger, Skype, Chrome browser, and Firefox browser. Future research will include the use of the UISAT on other different apps to verify the generalization of UISAT. For the final evaluations, researchers with domain knowledge of Android development were hired to remove any bias.

The research carried out by Xu et al. (2016) found five different human traits (extraversion, neuroticism, agreeableness, conscientiousness, openness to experience) just by analyzing the users' seven different mobile app category data. As well as how the five personality types known as the Big Five can be used to explain how various sorts of mobile apps are adopted. They have collected reviews from 2043 participants and categorized a big-five questionnaire – 44 questions to determine what kind of personality the user has- into one to five where one being fully disagrees and five means to agree. To fetch the user data using APIs, the data had the app's name when they installed the app for the first time, when the app last updated, source of the app – Google Play. They have used a random forest machine learning model and divided it into 70-30 rules for train and text. They have used Statistical Package for the Social

Sciences(SPSS) to perform analyses like statistical, Multivariate analysis of variance(MANOVA), Levene's f test, and Pearson correlation. They found out that machine learning methods are 65% better than random guessing. More mobile app data is to be considered instead of the survey should be collected as it represents the behavioral data of the users. Also, to make a generalized and scalable model, apps from all the categories should be considered.

The research conducted by Zhao et al. (2021) focuses on how digital consumer evaluations, deep learning, as well as the impact of product innovation interact with one another. The authors have collected data from Google reviews as well as a survey of 200 participants asking 14 questions to gain information about their emotional and physical requirements of the application. They have utilized Analysis of Variance(ANOVA) methodology, analyses performed on SPSS software namely: descriptive statistics, correlation test, model summarization, and linear regression on 13 different clusters of topics to test their hypothesis. Through the analyses of different tests, the researchers found out that product innovation (a cluster of reviews) is highly dependent on online feedback. The researchers proposed that the research could be improved by increasing the number of different variables and collecting more data from participants.

These are the summaries of all the research paper studies and explored to identify the target problems and to identify the gaps in the existing research.

Comparison among Relevant Research Papers

Table 3 shows the comparative study between the papers reviewed and studied.

Table 3

Comparison of Literature Review Solutions

Authors	Dataset	Models	Results
Assi et al. (2021)	2,000 mobile apps from Google Play Store	ANN model named Global Local Sensitive Feature Extractor(GLFE)	Precision is 0.81, F1-Score is 0.78, Recall is 0.74,
Dalpiaz et al. (2019)	App Store review data dataset	NLP methodology to	Developed a tool called "RE-SWOT"
Gao et al. (2015)	37 apps through data crawling	Dynamic Latent Dirichlet Allocation (LDA), True Mutual Information (TMI)	Created a Prioritizing App Issues for Developers (PAID) framework, with accuracy of 60.3%
Guo et al. (2017)	1,381 mobile applications from Google Play Store	Binary Naïve Bayes algorithm, K-Nearest Neighbors Clustering	Accuracy: 80.1%, Precision :86.3%, Recall:79.1%, and F1-score :82.5%
Jin et al. (2016)	Amazon Product Reviews	Binary Naive bayes Classifier, and have employed three strategies termed C-First, D-First, and R-First optimized Approaches	D-first strategy termed as information diversity is the most optimal strategy

Authors	Dataset	Models	Results
Lee and Raghu (2014)	Top 300 apps Dataset was collected weekly for a total of 39 weeks from the Apple App Store	Cox Hazard Model to analyze the survival time chart of the app	Factors such as Sales performance, Price, Category, Regular Updates, and high user reviews were found impactful for the app's survival rate
Y. Lee (2021)	LG Electronics' survey results and the TV reviews, Amazon review dataset	Grad-CAM algorithm, supervised convolutional neural network (CNN), Conditional random fields (CRF), and hidden Markov model (HMM)	The Proposed Method gives following Accuracies for Samsung Tv: 90.20%, LG TV: 91.90%, Sony TV: 91.73%
C. Z. Liu et al. (2014)	Longitudinal data containing 60,147 instances from 1,567 applications from Google Play store	Freemium Model	Accuracy: 85.6%
Li et al. (2017)	Google reviews from 2,311 Google applications	tf-relevance graph model	Achieved accuracies of 93.7% and 80% on 10,000 reviews and 4000 distinct topics.

Authors	Dataset	Models	Results
Ouyang et al. (2019)	Mobike and Ofo Android app store in China	Decision Tree, AdaBoost, Random Forest, Support Vector Regression.	The accuracy of the models are Decision Tree: 68.4%, AdaBoost: 67%, Random Forest: 71.4%, Support Vector Machine: 69.8%.
Su et al. (2019)	Google Play Store and APP Store	User-review mining via topic Identification, Sentiment Analysis, and Topic matching across apps (UISAT)	UISAT model achieved Precision:0.9229, Recall: 0.7141, and F-measure: 0.8024
Xu et al. (2016)	Developed Personality Test app to collect	Random Forest	Machine Learning models perform 65% better than a random guess.

Note. The table above shows the comparison between different studies explored during the Technical Survey.

Data Project Management and Plan

Data Management Plan

Data Collection Approach

The data for the project is collected from the Google Play Store with the help of Web Scraping. It is used to gather Customer reviews on the Mobile Apps of different domains. The

Python script is borrowed from (*Python Scraper - Google Play and App Store Reviews*, n.d.) which utilizes the concept of Python Scraping to extract the review data of the Mobile Apps on the Google Play Store. With the help of the Python script, the data is extracted for 20 different Mobile Apps covering five different domains.

The data is extracted as a CSV file using Pandas Dataframes. The four domains that are selected to collect customer reviews of Mobile Apps are Flight Booking, Food Delivery Car Rides, and Vacation Rental. All the legal and privacy-related issues are taken care of during the data collection process. An extensive process of Cleaning, Integration, and Transformation is performed on the data followed by Exploratory Data Analysis and Data Quality Report Generation.

Data Management Methods

The study recognizes the importance of implementing a data management strategy to ensure the reliability and accuracy of the information. It incorporates Google Play Store reviews into the machine-learning models. The data collection begins by obtaining the permissions and adhering to the agreements and terms set by the Google Play Store.

The study's focus is on selecting relevant information that effectively addresses the issue at hand. Then, execute a data cleaning and preprocessing process that involves handling missing values, eliminating duplicates, tokenizing, standardizing text data, and removing words. This procedure is crucial for enhancing the quality of the dataset. The Study adheres to ethical and data safety standards, particularly as the dataset includes user-generated feedback. For example,removing any personal data to preserve user anonymity and adhere to applicable data protection laws, such as General Data Protection Regulation(GDPR), to help people comprehend

and make efficient use of the dataset, thorough documentation that includes metadata and a data dictionary with a cleaned dataset is stored in the GitHub repository.

To manage the project, the study has effectively used the Data Management Plan(DMP) tool to create a comprehensive approach to managing the data. This tool has been very helpful in outlining goals for data collection, processing and analysis. It helps in identifying the types of data we collected including their format, volume and source. This holistic approach greatly aids the data management strategy. Moreover, the DMP tool helps to document the data collection methods consistently and accurately. It also helps to outline how to securely store the data by implementing strategies and access controls to prevent any loss or unauthorized access. Additionally it assists in cleaning and processing the data before analysis. By leveraging the capabilities of the DMP tool, confidence is gained in the quality and integrity of the data, which ultimately contributes to the success of the research project.

Data Storage Methods

Google Shared Drive is used to store the data and Google Cloud Storage is used to train and deploy the Machine Learning models in Google Colab. A shared folder is created on Google Drive to store the dataset that is accessible to the team members. All the extracted data is stored in CSV file formats. The Data Storage folder contains a Raw data folder holding the raw data extracted through the Python scraping, a Transformed Data folder that holds the data after data wrangling, a Training data folder that holds the training data set, and a Testing data folder that holds the testing data set for all four domains with a cumulative list of 20 mobile applications.

Apart from the raw data folder, all the other folders have cumulative data of the four

domains as raw data collection is done individually by each team member. The naming convention is App Name followed by App domain. For example, the data for the Airbnb app is stored as Airbnb_Vacation-Rental.csv. Cleaned data is added to the GitHub Repository with public access so that anyone can use the data for their varied purposes. Table 4 denotes the structure of the dataset and the file size of each dataset.

Table 4

File Storage Structures

Dataset	Folder Structure	Size
Raw Data - 20 Mobile Application Reviews	Raw_Data	~ 400 MB
Transformed Data	Transformed_Data	Varies
Training Data	Training_Data	Varies
Testing Data	Testing_Data	Varies
Archive Data	Archieve_Data	Varies

Note. The figure above shows the file storage structures with the corresponding file sizes.

Data Usage Mechanisms

Different competitive apps are finalized belonging to different domains, followed by the collection of their reviews for comparing the apps to find out which app serves in a better way for the users. Data is collected using a Google Play Scraper which is a built-in Python library. This data is used to predict the sentiment of the user. It is done to know the satisfaction level of the user after using the app and how well the app served its purpose. The extracted data is used

mainly for the analysis purpose to get to know which app has a better reach and has an edge over the other.

Further on, using the holdout method this data is used to train different ML models and later on for testing of those models. All this process is documented in a ReadMe File in a GitHub Repository that is created. Source code used to extract the data and the code used to clean some of the data is also added for all future references. Along with that all the necessary references, links, and other deliverables are added to the GitHub Repository. This documentation is made public so that anyone can use it to compare different apps having the same domain for their analysis. It can also be used by app developers to know the feedback of their apps and make modifications accordingly. Table 5 mentions the roles of all the team members at different stages of the Data Management Plan along with the tools required for each phase.

Table 5

Assignment of Accountability

Role	**Assignee**	**Tools/ Steps conducted**
Data Collection	Kashish, Bhavik, Smeet, Swetha	Python Script
Data Cleaning and processing	Smeet, Swetha	Jupyter Notebook
Data Storage and Sharing	Bhavik	GCP and Google Drive
Data Management & Exploration	Kashish, Bhavik	JIRA, Python,

Role	Assignee	Tools/ Steps conducted
Ethics and legal compliance	Kashish	All the regulatory compliance are be checked, along with the maintenance of anonymity of customer personal data

Note. The table above shows the tasks assigned to individual members for Various Stages of Data Management Plan (DMP).

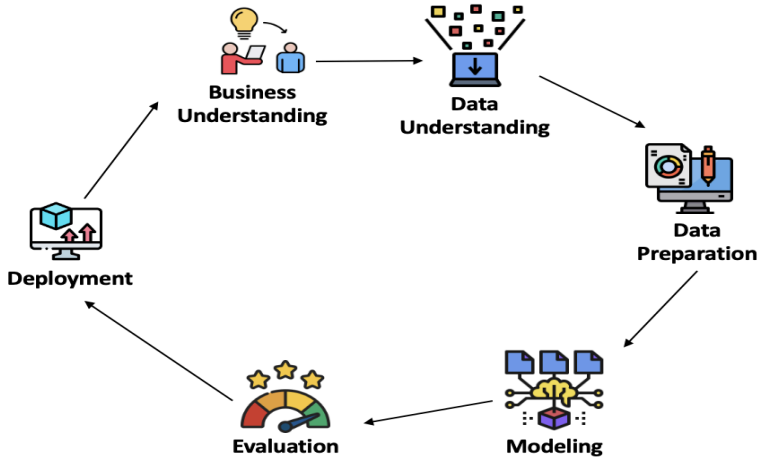
Project Development Methodology

CRISP DM Methodology

The project follows a CRISP DM methodology which is segregated into six different phases shown in Figure 1 which are, Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, and Deployment of the Model. This is the most widely used method of implementing Data Mining and Data Science Projects.

Figure 1

CRISP DM Methodology



Note. This figure demonstrates the different phases in CRISP DM Methodology. The images used in this figure are borrowed from (Flaticon, 2023) which are used to explain the phases.

Project Development Process. The project is similarly divided into six different phases, following the CRISP DM methodology. Each domain is further subdivided into tasks and sub tasks to provide more granularity and simplicity to the implementation of the project. The phases are:

Domain Finalization. This is the first phase in the development of the project. In order to identify the domain of interest, the relevant study material is gathered and explored. With the help of this study, the literature review is completed and proposes a project topic of interest. The feasibility analysis is performed while finalizing the domain in terms of data availability, the project scope and its future scope. Project Proposal is then documented in APA format and submitted to provide a project overview in terms of Project Goal, Motivation, Methodology, Data Sources and the Relevant Work.

Data Collection. The second phase in the project development is the collection of data relevant to the project goal. With the help of Web Scraping, the review data of the Mobile Apps is extracted from the Google Play Store. A python script is borrowed from (*Python Scraper - Google Play and App Store Reviews*, n.d.) which is used to extract the customer review data of multiple apps in different domains. The data is initially collected from four different domains of data which are, Flight Booking, Food Delivery Car Rides, Vacation Rental. The data extracted is converted to CSV files using Pandas Dataframes.

Data Wrangling. This is the third phase of the project development plan where the extracted data goes through a rigorous process of data cleaning and preprocessing. Since the data

from four different domains is collected, it is first integrated into a single data file. The next step is to perform the Data Cleaning, where all the null, duplicate and missing values are handled and resolved. Since the project is dealing with the customer data, maintaining the privacy of the customer is an important constraint which is considered while using it. All the privacy related data is anonymized or removed in order to handle the privacy concern. The normalization of data is performed in order to minimize the modification errors and redundancy in data.The required transformations is done to enhance the performance of the models in terms of efficiency and accuracy.Once done with the Data Cleaning and Preprocessing, Exploratory Data Analysis is conducted to identify the useful insights and outliers in the data. This is followed by Feature Engineering and Feature Scaling. At the end of this phase, a Data Quality Report is prepared to provide the overall data quality information in terms of descriptive statistics for both continuous and categorical features in data.

The next critical step after preprocessing is choosing the Machine learning model that accurately identified the sentiment of the customer review. Developing a machine-learning model for the preprocessed data is the first step in this phase. The models are then evaluated using the testing data.

Model Building. The main objective of the model-building process is to identify the best model for obtaining meaningful sentiment for the mobile application. Support Vector Machine (SVM), Convolutional Neural Network - Bidirectional Long Short Term Memory (CNN-BiLSTM), Bidirectional Encoder Representations from Transformers (BERT), and XLNet are the four different models that are used to provide the accurate sentimental score for each of

the apps. All the models are trained in parallel and fine-tuned by the hyperparameters according to the requirements of the model to enhance the performance.

Model Evaluation. This stage assessed how well the models are designed to predict results based on historical data. Model review is essential to ensure that all the models are accurate and dependable. The first stage in that procedure is choosing the performance metrics that would be used to assess a model. Evaluation metrics like accuracy, precision, recall, and F1 score are calculated to evaluate the model's performance.

Business Evaluation. A machine learning (ML) model's business evaluation entails determining the model's influence and efficacy about a certain organizational objective. It looks at how well the ML model fits the goals of the company and creates value to improve the overall performance of the business. The foundation of the machine learning model is the analysis of customer review sentiment, evaluation of feature requests and enhancement suggestions for improvement, competition analysis to determine market positioning, and measurement of user engagement indicators. To have a comprehensive understanding of a product's functionality and user satisfaction, proper consideration of the App Store ratings is also done. This information helped the study make strategic decisions and adaptations.

Documenting Final Deliverables. *T*he CRISP-DM process model's deployment phase is where all the final deliverables are documented. After reviewing the assessment findings in the test environment, the best machine learning model that outperforms other models constructed for assessing each aspect of the user reviews of the mobile application is chosen. Each team member produced a separate research paper that focuses on their machine learning model; the final report highlights the optimal mode for the business use case.

Project Organization Plan

Work Breakdown Structure

In order to plan the different tasks and activities in the project, a hierarchical structure known as the Work Breakdown Structure is created in the Jira tool using an in-built API in the tool, known as Move and Organize. The hierarchy structure is decided according to the levels of hierarchy utilized in the Jira tool.

In Jira there are three levels of hierarchy, where the top most level is the Epic, followed by User Story and then Tasks. The Work Breakdown Structure follows the CRISP DM methodology where each of the six phases corresponds to the six Epics in Jira Project, which is further divided into User Stories and Sub-Tasks. The six Epics that are created for this study are Domain Finalisation, Data Collection, Data Wrangling, Building and Evaluating the Model, Business Evaluation, and Documenting the Final Deliverables.

The first Epic in the WBS is the Domain Finalization Phase, which consists of four User Stories which is further divided into different sub tasks. This Epics aims to finalize the project area to work upon. It starts with the collection of relevant sources of study and research, which is utilized to complete the Literature Review. The project domain is identified and decided based on the feasibility analysis in terms of Data Availability and the Project Scope. Once the Domain is established, a Project Proposal is documented to provide the overview of the proposed project.

The second Epic in the WBS is the Data Collection Phase that comprises a single User Story divided into a single task. This involves the collection of data which is utilized in the project. The main source of data is Customer Review Data collected from the Mobile Apps on the Google Play Store.

The third Epic in the structure is Data Wrangling Phase, which is divided into seven User Stories and the corresponding sub tasks. This phase starts with the integration of data which is accomplished using Python. The integrated data is then cleaned to remove any missing, null or duplicate values. The necessary reduction and transformation is incorporated in case there are certain summarization levels required for the statistics or there is any invalid data. It is then followed by the Exploratory Data Analysis which is used to find any patterns or insights in the data or any possible outliers which can affect the model's performance. The final step in this phase is the generation of the Data Quality Report, for both Continuous and Categorical columns in the dataset.

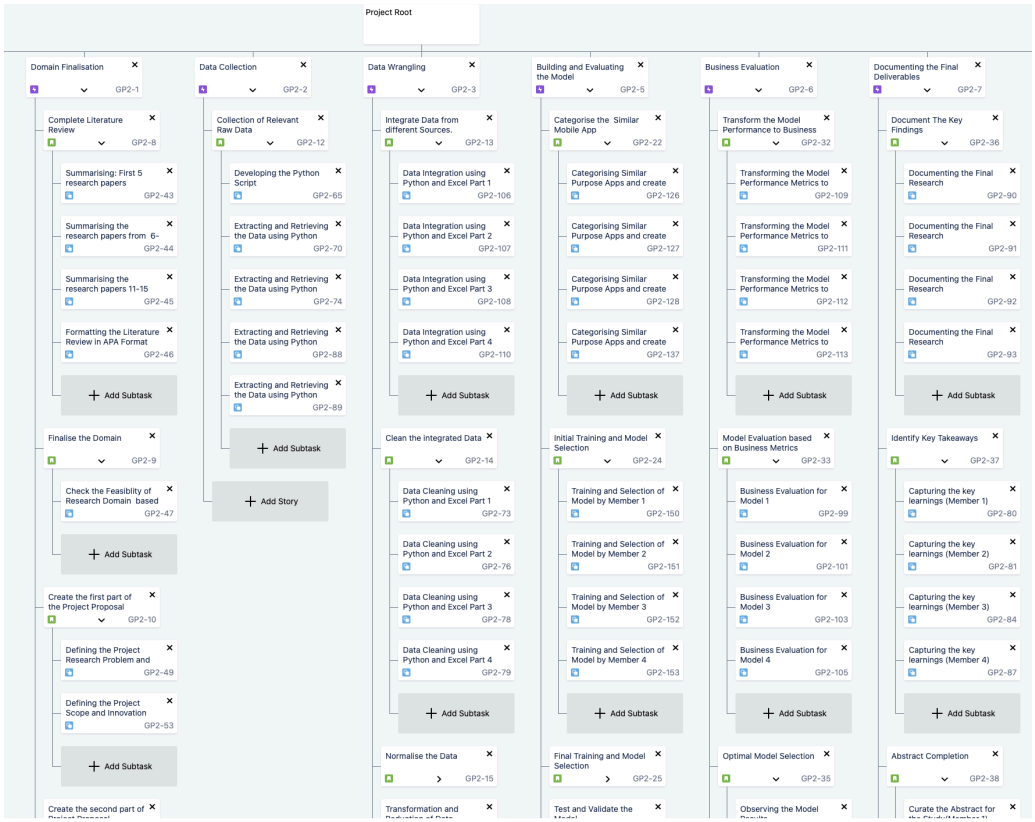
The fourth Epic in the WBS is the Model Building and Evaluation Phase, where there are 9 user stories including the steps that are required to build and evaluate the model. It starts with the categorization of mobile apps in the same domain. The next step is the initial training and selection of the model, in which four different models are selected and trained with the help of training data. The data is divided into 80:10:10 ratio, which is fed to the selected model. The final model training is done by using this split. Sentimental analysis is performed on the individual models along with the calculation of the Sentiment Score. A comparative study is done to compare the sentimental score of the mobile Apps in each domain. In case of any outliers in each model, it is identified and documented. The next step involves the Model Testing and Evaluation based on the different performance metrics like Accuracy, F1- score, and Recall. After evaluating each model, a correlation matrix is created to gauge the performance of each Model.

The fifth Epic is the Business Evaluation Phase, which involves the three user stories where the model performance is evaluated in terms of business perspective. The first step is to transform the model performance metrics into business performance metrics. The transformed metric is then evaluated to select the most optimal Model based on the business perspective.

The sixth Epic is Documentation of Final Deliverable where all the key deliverables are collected and documented. This starts with documenting the key finding in the project, followed by the documentation of the Key Takeaways and Learning. A Project Abstract is curated , which is then included in the Final Project Report. An individual research paper is created, documenting the model building and development. A Model Development video is also created enlisting all the necessary steps taken in building and evaluating the model. The last step is to prepare a Project Presentation which is presented and submitted. Figure 2 shows the Work Breakdown structure.

Figure 2

Work Breakdown Structure



Note. The figure above shows the Work Breakdown Structure explaining the breakdown of tasks.

Project Resource Requirement and Plan

Hardware Requirements

This project aims to predict the better serving app for the users based on the perspectives of different user reviews. For this to be done, there is a need for a high-functional and software-compatible laptop that can take care of Data Storage, Data Cleaning, Data Pre-Processing, Model Building, and Testing. A detailed list of all the Hardware and Software requirements are mentioned in Table 6 and Table 7 respectively.

The raw data consists of around 400 MB which is stored in the primary machine. Also, it is shared in Google Drive which all the group members access and they can also save the files on their laptops. This is done so that every member can perform any tasks on their own on the data and also as a backup in case the primary laptop crashes.

Table 6

Hardware Requirement

Hardware	Memory	Usage
12 Core CPU, 19 Core GPU, 16 Core Neural Engine	Ram: 16 GB, SSD: 1 TB	Data Handling, Training, and Testing ML Models

Note. The table above shows the list of Hardware Requirements for the project.

Software Requirements

Different libraries, packages, API, and coding environments are utilized to develop different machine learning and deep learning models..

Table 7

Software Requirements

Environment/APIs/Libraries/ Packages	Usage	Version
Python	To develop code script for the model	3.10.4
Google-play-scraper	To fetch user review from google play store	1.2.4
Pandas	Used for data manipulation	2.1.1
NumPy	Used for numeric calculation	1.19.2

Environment/APIs/Libraries/ Packages	Usage	Version
scikit-learn	Used to perform traditional machine learning model like, regression, clustering, and classification	1.3.2
PyTorch	Used to design deep learning models like, BERT, XlNet	2.1.0
TensorFlow	Used to develop deep learning model like BiLSTM	2.14.0
NLTK	Used to preprocess the data	3.7
SpaCy	Used to preprocess text data	3.5
Matplotlib	Used for visualization	3.8.0
Seaborn	Used to visualize data	0.13.0

Apart from hardware and software requirements, storing data in a database also plays a vital role. Table 8 describes database requirement

Table 8

Database Requirement.

Database	Usage	Version
Google Cloud Storage	To store and access data on Google Cloud Platform (GCP)	2023 edition

Note. The table above shows the database requirement.

Tools and License Used

Table 9 shows the complete list of the number of tools and licenses utilized in this project. This makes sure all the individuals are aligned. The meticulous selection of these components influences the project's technological viability and scalability as well as compliance, related to the usage of machine learning solutions being built.

Table 9

Tools and Licenses.

Tools	Usage	License
Jupyter Notebook	For executing the model on web-based environment	BSD-3-Clause
Google Colab	To have data pipelined from google cloud storage	Proprietary
Excel	CSV files	Proprietary
Google Suits	To deliver project documentation	Proprietary
GitHub	For hosting code repository and collaboration	Academic Free License v3.0
Zoom	To conduct virtual project meetings	Proprietary
Discord	To conduct virtual project meetings	Proprietary
JIRA	To manage project and keep track tasks	Proprietary
Grammarly Premium	To check grammar score and plagiarism	Proprietary

Project Cost and Justification

Table 10 shows the cost involved in the project of all the resources and tools that are required to complete the implementation of the project.

Table 10

Project Resources and Cost Justification

Resources	Justification	Duration	Cost
Jira	Creation of the WBS, Gantt Chart	4 months	$0.00
Google Colab	For the data storage on Cloud	4 months	$100.0
Zoom, Discord	For the Team Collaboration	4 months	$0.00
Grammarly Premium	For the checking the grammar, spellings and plagiarism	4 months	$140.0
Hardware Cost	The setup required for data processing, wrangling and visualization	4 months	$2700.0

Project Schedule

Gantt Chart

For this project Gantt chart is used which visually represents the complete outline of the activities involved in the project. It resembles a timeline that displays tasks together with their start and end dates and their connections to one another. On the chart, the horizontal bar determines each task, with the task's duration corresponding to the bar's length. Gantt charts

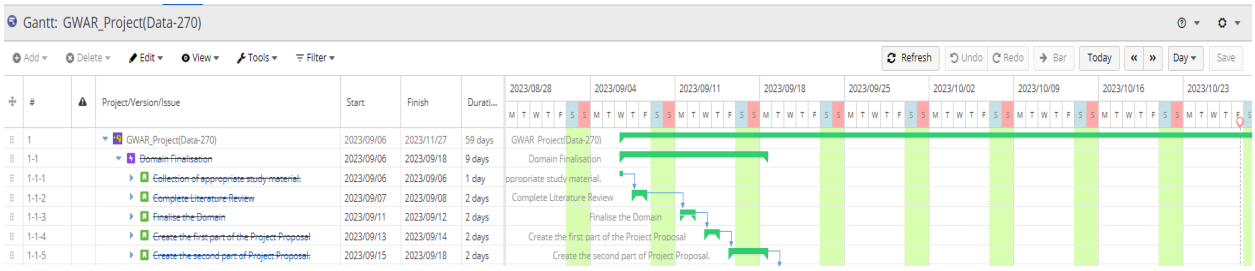
facilitate the visualization of job relationships, order, and completion dates.In a real time environment, Project managers, teams, and stakeholders benefit from this visual clarity in terms of organization, prioritization, and project tracking.

Additionally, Gantt charts are also useful for tracking progress because they allow you to see how the actual timetable compares to the planned schedule. Also, they improve communication by offering a concise, visual depiction of the project's state, facilitating cooperation and decision-making. The complete Gantt Chart can be seen in Appendix A.Figure 3 shows the Gantt chart of Domain Finalization.

The Domain Finalization phase has two main tasks, performing an extensive literature review of the relevant research papers related to the use case and finalizing the domain. This phase is performed from 8th of September, 2023 to 22nd of September, 2023. Figure 3 shows the Gantt chart of the Domain Finalization phase.

Figure 3

Gantt Chart of Domain Finalization

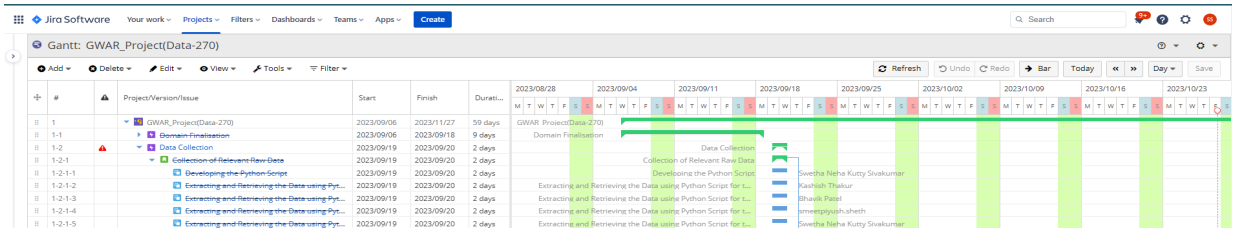


Note. The figure above shows the task and the dependencies for the Domain Finalization.

The Domain Finalization phase has two main tasks, performing an extensive literature review of the relevant research papers related to the use case and finalizing the domain. This phase is performed from 8th of September, 2023 to 22nd of September, 2023.

Figure 4

Gantt Chart of Data Collection



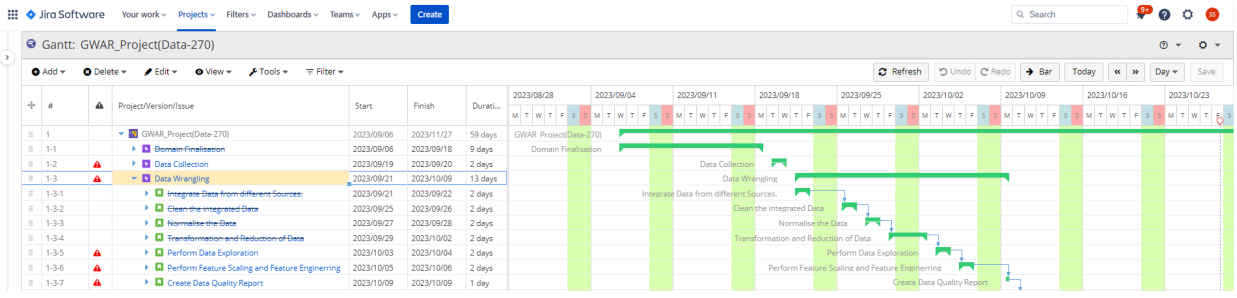
Note. The figure above shows the task and the dependencies for the Data Collection.

The Data collection phase is spread over a period of 10 days starting from 9th september, 2023 to 6th October, 2023 that is shown in Figure 4. The first step is to develop a script to scrap the data from the web followed by extracting the data in the CSV file format.

Phase three of the project is Data Wrangling which is shown in Figure 5. In this phase, the mobile applications review collected during the data collection phase are integrated. After integrating the data, a cleaning process is performed to make sure all the anomalies are handled correctly. This step is handled coordinated in both manually and automatic processes.

Figure 5

Gantt Chart of Data Wrangling

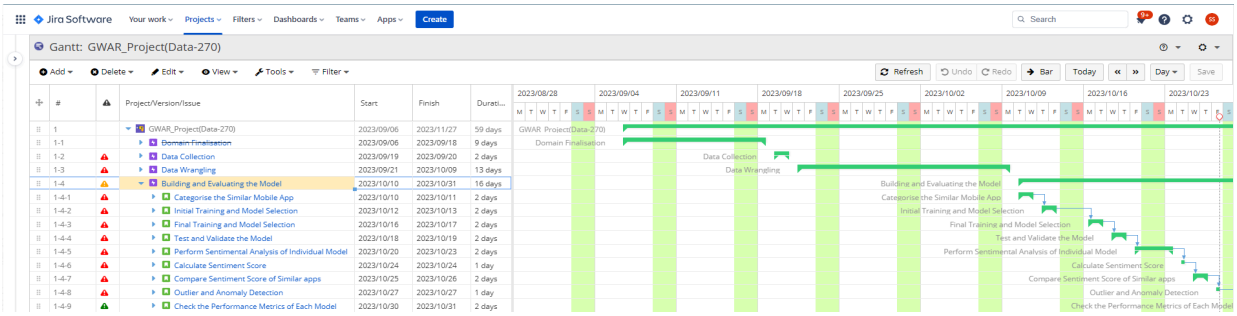


Note. The figure above shows the task and the dependencies for the Data Wrangling.

The Data Normalization is the next step in the data wrangling phase where the data is standardized by using Tokenization,lowercasing the data, removing punctuations, and handling special characters in the dataset. The normalized data is then transformed and the number of features in the dataset are reduced while retaining the important information. In order to make it easier for the Machine learning algorithms to learn the underlying data, feature scaling is performed and to improve the performance of the machine learning model, feature engineering is performed. Both Feature Scaling and Feature Engineering together helped in improving the data quality. The final step in this phase is to generate the data quality report.

Figure 6

Gantt Chart of Model Building and Evaluation



Note. The figure above shows the task and the dependencies for the Model Building

The next critical step in the project is to build models from the normalized data. This is where the normalized data is trained on the model selected. In the initial training phase, Machine Learning models are finalized, which are then tested and validated to evaluate their performance and accuracy.

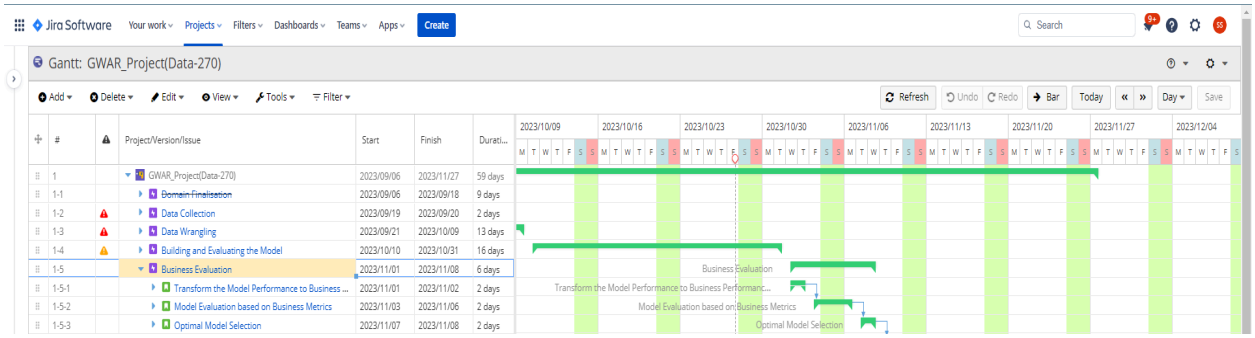
The target of the model building is to understand the sentiment from the user reviews, sentimental analysis is performed to achieve this outcome. Sentiment score quantifies the

emotion in the user reviews. Any further outliers and anomalies are detected and validated against the other models.

By comparing the sentiment score of the trained models the user's attitude towards the application is analyzed. The overall tasks listed above start from 10th October 2023 until 31st October 2023 for a span of 16 days. The list of tasks and dependencies between each task of the Building and Model Evaluation phase of the project are shown in Figure 6.

Figure 7

Gantt Chart of Business Evaluation



Note. The figure above shows the task and the dependencies for the Business Evaluation

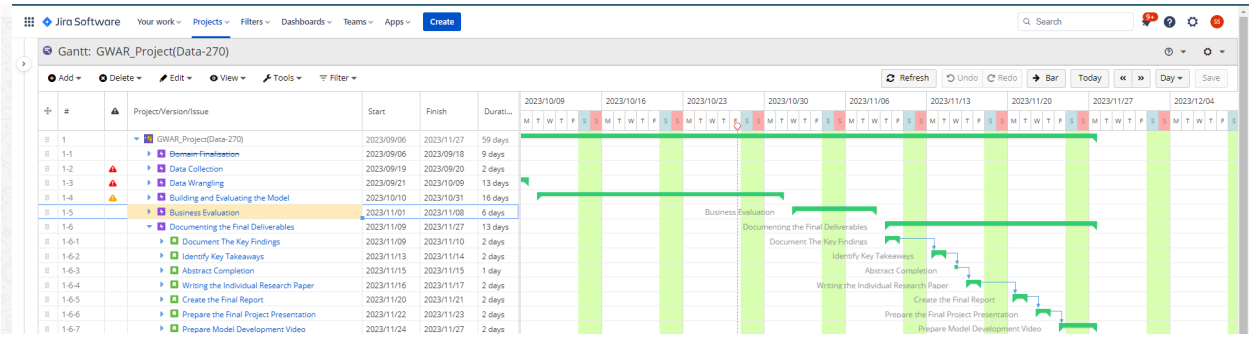
Figure 7 shows the Business Evaluation phase started from the 1st of November, 2023 to the 8th November, 2023 where the machine learning model built is assessed against the business values and potential impact. This assessment goes beyond the technical performance of the models. Finally the optimal model is selected based on the accuracy and performance against the machine learning model built by each team member.

The final phase of the project is Documenting the final deliverables. In this phase, the key findings are captured and takeaways are documented for recommending further improvements to the application. Also, the reports of individual research papers and final reports are created and

presented. Figure 8 shows the tasks and dependencies of documenting the final deliverables phase.

Figure 8

Gantt Chart of Documenting the Final Deliverables



Note. The figure above shows the task list for the Final stage of the project.

PERT Chart

The Critical Path Analysis (CPA) method is used by the Program Evaluation Review Technique (PERT) chart for Text Summarization to comprehend and identify the critical path using a task-based approach. draw.io application's template is used to create the PERT chart for this project. In the project's network graph diagram, the Critical Path Analysis approach assists in identifying the longest series of dependent activities that must be completed within a specific timeframe and are essential to its effective completion. Critical paths are shown using red arrows and dependencies are represented by black arrows.

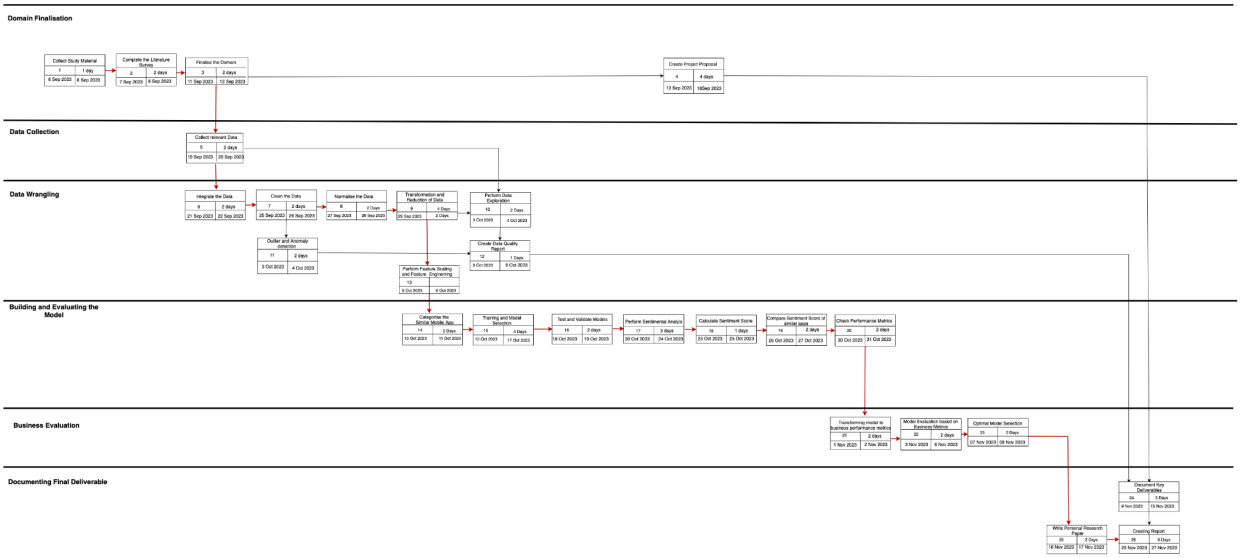
The PERT Chart displays the six phases namely: Domain Finalization, Data Collection, Data Wrangling, Model Building and Evaluating the model, Business evaluation, and Documenting Final Deliverable, in which the whole project is divided. The critical path consisted of 21 tasks and it took 47 days to complete. Figure 9 displays a task-based PERT chart

with the date and duration of individual tasks. The critical path begins with collecting study material to finalize the research domain and decide the final problem that can be solved.

After that, the relevant data of 20 applications are merged together for data cleaning. Standardizing data and data transformation are also a part of the critical path as performing data scaling helped to optimize the model. Once the data is cleaned, model building is done and performance is checked on testing data. Once the model is ready, sentiments are analyzed on different app reviews and different topics to categorize them into good, moderate, and bad. Figure 9 depicts the Pert Chart for the project along with the identified critical path.

Figure 9

PERT chart



Note. A Task-based Pert Chart is illustrated above, where the red arrows depict the Critical path of the project. The tasks along the critical path are all the necessary tasks required for the successful completion of the project.

Data Engineering

Data Process

The objective of the project is to analyze the user reviews to comprehend the sentiment and most recommended features that have received positive feedback from the users. This information helps the users to choose the app best suited to their needs and for programmers to build superior products in the market. To accomplish this goal, the user reviews from 20 different mobile applications in the major travel and leisure domains have been extracted from the Google Play Store using Python and the Google Play Scraper package.

The output of the extraction is 20 CSV files which hold five files each, of individual mobile domains with user review information that has been retrieved. The four major domains for mobile applications are Flight Booking, Vacation Rental, Car Ride, and Food Delivery. After extraction is completed, the twenty files are merged into a single consolidated dataset. The raw dataset is then cleaned and preprocessed to improve the data quality, which is then followed by splitting the transformed, cleaned data into training, testing, and validation datasets. These datasets are used to evaluate the performance of the model.

The collected dataset has been stored in the Google Cloud Platform and Google Drive which can be accessed by individual research team members. The Exploratory Data Analysis (EDA) is performed initially to understand the characteristics of the data, patterns, and significant correlations between the features. This step further helps in further directing the analysis required in the dataset.

The data Pre-processing phase is where all the quality issues in data such as missing values or incomplete data, noisy data, and inconsistent data have been identified and handled.

The first step in this is to handle the missing values by imputing the values by forward or backward filling and in some cases removing the missing records where the record count is less than 10% of the total volume of records followed by removal of null values. The final step is to handle inconsistencies, where the duplicate records have been removed, and inconsistencies in the data format have been resolved.

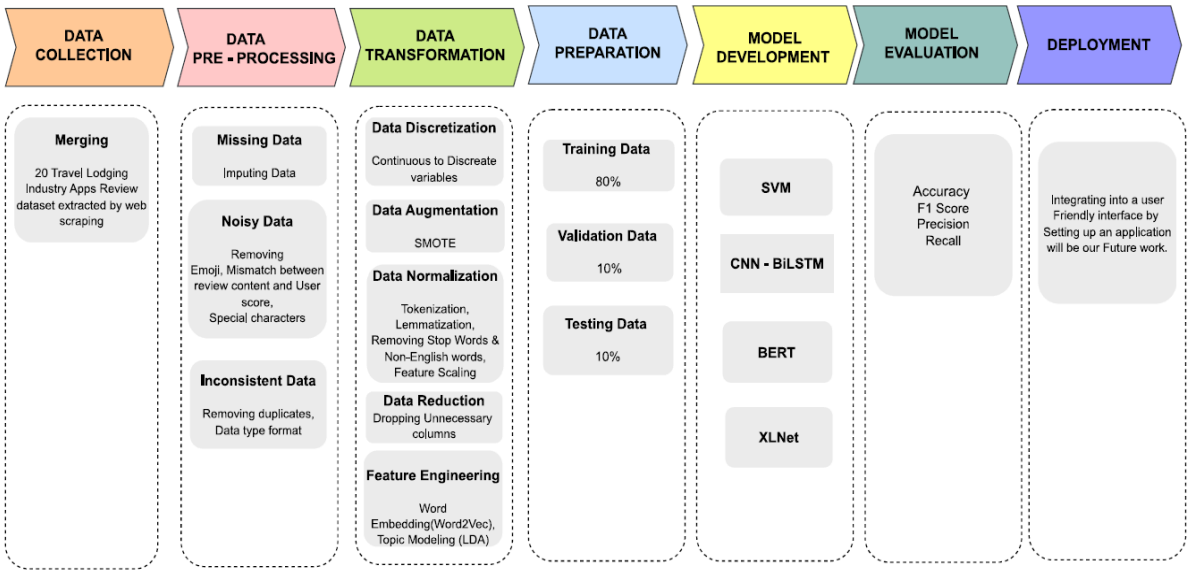
The next consecutive step is to transform the text data to make the data more accessible and understandable for the predictive models. The step-by-step transformation process ensures the accuracy of the model is optimal in the subsequent analysis ultimately maximizing the data-driven decision-making. The several phases involved in data transformation are data Discretization where continuous data have been transformed to discrete followed by Data Normalization and Data Reduction. To normalize the data, Text has been converted to lowercase, Special Characters, punctuations, stop words, and emojis have been removed and Tokens are generated so machines can process the data easily and improve the model's accuracy. In data reduction, the features which have no or minimal relevance to the target features have been removed.

Further Principal Component Analysis (PCA) reduces features by introducing new columns representing the numerical features. The size of the vocabulary is determined. Topic Modeling is performed using the Latent Dirichlet Allocation, where the common themes in the data are identified. Ten different topics are identified using this technique, out of which four are shortlisted for further analysis. The topics identified are Payment Experience, User Friendly, Customer Service, and App Functionality. This is followed by the Word2vec technique on the Review content feature, which is the descriptive feature, and the Category feature, which is the

target feature, which is used for word embedding using the Gensim python library. Data Regularization is the final step in the Data Transformation where Dropout methods are implemented to avoid overfitting. Finally, the transformed data is split into training, testing, and validation datasets in an 80:10:10 ratio. The imbalance in the training data is balanced using the SMOTE technique. Figure 10 below shows the overall flow of the data starting from data collection until model building followed in this project.

Figure 10

Data Process Workflow



Note. The figure above shows the overall Data Process Workflow.

Data Collection

Data Source and Parameters

The project mainly deals with Customer Reviews of Mobile Apps. The Review Data is gathered from the Google Play Store using the concept of Web Scraping. The in-built Python

Library, which is Google Play Scraper is utilized to capture the information about the mobile applications. The data collection is carried out using a Python script which is borrowed from a Youtube website (JiFacts, 2022).

The project mainly consists of 20 different Travel Lodging Industry apps, which are divided into four major domains: Flight and hotel Booking Apps, Vacation Rental Apps, Food Delivery Apps, and Car Riding Booking Apps. Each App domain consist of five CSV files , where the Flight Booking domain consists of a total of 171,576 records of data, the Vacation Rental domain consists of a total of 272,422 records, the Food Delivery domain has a total of 370,969 rows and the Car Ride Booking Domain consists a total of 144,524 rows. The complete merged dataset is about 380MB, with an overall count of 959,491 records of data.

The dataset includes the following features which are reviewId, userName, content, score, userImage, thumbsUpContent, replyContent, replied, reviewCreatedVersion, at, and appVersion. Table 11 below describes different features present in the raw datasets.

Table 11

Feature Description

Feature Name	Description
UnNamed:0	The Index for the Records
reviewId	The unique ID provided to each User
userName	The name of the User giving the Review
userImage	The URL of the User's Image
content	Text representing User's Review
score	User's Rating of Mobile Application

Feature Name	Description
thumbsUpContent	The number of likes on the User's Review
reviewCreatedVersion	The App Version of the Reviewer
at	The Date and Time of the User Review
replyContent	The User's Reply to the Reviews
repliedAt	The Date and Time of the Reply to a User's Review
appVersion	The version of the Mobile Application

Note. The table below shows the number of records fetched for each mobile application.

Table 12 shows the number of records in each application along with the corresponding Application domain.

Table 12

Raw Dataset Records Count For Each Application

Domain	Application	Number of Records
Flight Booking	Cheapflights	141,13
Flight Booking	Priceline	75,633
Flight Booking	CheapOair	5907
Flight Booking	Expedia	72,724
Flight Booking	Momondo	3199
Car Rental Domain	Curb	3537

Domain	Application	Number of Records
Car Rental Domain	Didi Chuxing	918
Car Rental Domain	Easy Taxi	11,763
Car Rental Domain	Gett Taxi	10,397
Car Rental Domain	Lyft	117,909
Vacation Rental	Agoda	99,539
Vacation Rental	Airbnb	106465
Vacation Rental	Hopper	27,599
Vacation Rental	Trivago	26,795
Vacation Rental	Vrbo	12,024
Food Delivery	GoPuff	23,160
Food Delivery	Grubhub	128,753
Food Delivery	Justeat	107,862
Food Delivery	Postmates	55,757
Food Delivery	Talabat	55,437

Note. The above table represents the number of data records in each raw dataset.

Data Collection Plan. All the applications have the same set of features since the data is fetched through a single collection technique of Web Scraping. In the collected dataset, four domains have been identified and included in the Final Merged Dataset, shown in Figure 11.

Figure 11

Data Collection Plan

Data Collection Plan

Project number: 8 | Project title: Study of Mobile App Rivals | Project leader: Kashish Thakur | Date: 11/03/23

Description of the data collection

The data is collected using the concept of Web Scraping. The in-built Python Library, which is Google Play Scraper is utilized to capture the information about the mobile from the Google Play Store.Using the Python Script, the customer review data is fetched of the mobile apps. The extracted data is converted into CSV files using Panda Dataframes, which is then utilized further for the project as the data source. In total we have about 959,491 rows collected from different mobile apps belonging to Travel and Lodging Industry.

What will be done with the data once it has been collected?

	Identify how well the process is meeting **customer** needs		
YES	Obtain **exploratory view** of the process		**Data Cleaning** where the inconsitent, noisy and missing data will be resolved.
	Evaluate the **measurement system**		**Data Transformation**, where Data will be **Normalized**, and **Data Reduction**, **Data Augmentation** and **Discretization** will be carried out
	Check the **stability** of the process (is it in control?)		**Data Preparation,** where data will be split into **Training, Validation** and **Test Data,** followed by **Data Statistics** and **EDA**
	Conduct a **capability** study		

Key Variables - A summary of the chosen input variables (Y's) and/or output variables (X's)

		1	2	3	4	5	6
What?	**Variable title**	ReviewContent	ThumbsUpContent	AppDomain	AppVersion	AppName	ReviewerScore
	Input (X) or output (Y) variable?	X	X	X	X	Y	Y
	Unit of measurement	Text	Integer	Text	Text	Text	Integer
	Data type	Attribute	Continuous	Attribute	Attribute	Attribute	Continuous
	Collection method	Automated	Automated	Automated	Automated	Automated	Automated
	If manual						
MSA	Gauge/instrument	NA	NA	NA	NA	NA	NA
	Location	NA	NA	NA	NA	NA	NA
	Gauge calibrated?	NA	NA	NA	NA	NA	NA
	Measurement system checked?	NA	NA	NA	NA	NA	NA
	Precision (R&R) adequate?	NA	NA	NA	NA	NA	NA
	Accuracy adequate?	NA	NA	NA	NA	NA	NA
Historical data	Historical data exist?	Yes	Yes	Yes	Yes	Yes	Yes
	Source of historical data	Google Play Store					
	Historical data representative/reliable?	Yes	Yes	Yes	Yes	Yes	Yes
	Mean	NA	1.5	NA	NA	NA	3.84
	Upper specification limit	NA	1751	NA	NA	NA	5
	Lower specification limit	NA	0	NA	NA	NA	0
	Standard deviation	NA	13.07	NA	NA	NA	1.63
	Target						
Sampling	Minimum sample size (MSS)	NA	NA	NA	NA	NA	NA
	Sampling frequency	NA	NA	NA	NA	NA	NA
	Sub-grouping needed?	NA	NA	NA	NA	NA	NA
	Sub-group size	NA	NA	NA	NA	NA	NA
	Stratification needed? (time, shift)	NA	NA	NA	NA	NA	NA
Who?	**Data collector**	Smeet for Domain 1	Bhavik for Domain 2	Kashish for Domain 3	Swetha for Domain 4		
	Operational definition exist?	Yes	Yes	Yes	Yes		
	Data collector trained?	Yes	Yes	Yes	Yes		
	Resources available for data collector?	Yes	Yes	Yes	Yes		
When?	**Start date**	19-Sep	19-Sep	19-Sep	19-Sep		
	Due date	21-Sep	21-Sep	21-Sep	21-Sep		
	Duration (in days)	2	2	2	2		

Note. The above Collection Plan describes the detailed information about the Data parameters and the collection information. The plan explains the entire process of data collection where the details about data parameters, type of data, data collector, and the duration of data collection are provided.

Dataset Samples

The information below consists of the data samples of four out of the twenty applications chosen for the Data Collection. Each data sample represents one of the four domains in the Travel and Lodging Industry. The data sample gives an overview of the type of data present in each raw dataset. Since the project focuses on the review data, each dataset consists of different aspects of the Mobile Application and the User's Review. This can be seen in the different data samples shown below.

Figure 12

Airbnb Dataset

	A	B	C	D	E	F	G	H	I	J	K	L
		reviewId	userName	userImage	content	score	thumbsUpCount	reviewCreatedVersi	at	replyContent	repliedAt	appVersion
0		7a-4253-a366-61	hristopher Robbir	i8ocJew8DxO-Esi	Jumping through hoops to make a reservation. You'd think you're visiting a high up government building the way you have to post a photo, then show ID to prove it's you but it won't accept the ID if the lighting isn't just right. The fees and taxes are over half what the entire stay is for 2 nights. It's ridiculous. Air BNB fee alone was over 1/3 of the fee for one nights stay. It would be nice to be able to filter by selected dates instead of looking at a place then finding it's not available.	2	183	23.41	)23-10-16 22:23:5			23.41
1		213-41c1-95d8-0	Rett Anderson	-UjXvv6OYa2jD_	Overall, it works pretty well until you send a property of interest to friends. Once they look at it and do a sample booking on the property, the Airbnb app uses it's algorithm to immediately increase the price before you book. This happened to me today when I waited 5 minutes to book it. They added another $100 to the price. It is incredibly frustrating. I guess you have to make a rash decision right in the moment to get a good deal.	4	698	23.34	)23-08-30 19:08:4			23.34
2		e1-4c3e-8162-83	atrick Christense	LV-UjWBvXEr7m	Every time there's an app update, features disappear and everything gets glitchier. App was totally gutted this year in an endless myriad of ways. The most recent issue is deleting the ability to price long term stays uniquely by any number of days, weeks, or months. I know the length of stay pricing got moved to the calendar tab, but now there's only weekly and monthly rates and no option to customize to specific numbers of days, weeks, or months. Also, can no longer see messages while typing.	1	589	23.37	)23-09-18 11:22:3			23.37
3		acc-4d5b-8a9f-00	Sherry Kersell	ocJc8lEaMHeLO	AirBnB has been fantastic for the most part! I just wish they'd add an option to make additional payments whenever you want. Like I should be able to open my trip and click "make a payment". This way you can pay down the balance as money becomes available, instead is getting hit with the entire amount on the due date! That's really the only complaint I have about this app. Otherwise, everything else is easy to navigate! Please add "make another payment" option.	3	24	23.34	)23-09-27 12:16:1			23.34
4		lca-4cf2-91a8-78	James Seward	ocJbAV3ZFhsI7I	Pretty easy to navigate and use. The interaction to book was pretty easy as well. I do find a little annoying that some rentals appear to be available, but the minute you go to complete the booking, it's all of a sudden unavailable. I don't know if this is a marketing employee or just the system takes too long to update, but it's pretty frustrating to spend all that time searching, you finally find the place you like and are excited about, and immediately it's not available. Just don't show it.	4	402	23.32	)23-08-22 21:50:5			23.32
5		ib2-49b5-b3e8-5	Main Email	ocL0HDmu0f1LF	As a host, it does what it needs to do. But the messaging module is annoying: 1) Text box is too small, displays only 4 lines, so you can't see what you wrote above that (unless you scroll the message). 2) The "send" button is within the text box. It's not difficult to inadvertently hit it when scrolling the text or trying to relocate the cursor to edit. Either move it, or at least allow a confirmation b4 sending. Aug 17 update: messaging module still annoying. (See above)	4	840	23.32	)23-08-17 13:29:(			23.32
6		3cf-4e15-8249-dd	Tyler	3ocKEnYvTB13ye	After attempting to make a reservation, the Airbnb app said I needed to verify my identity. That's all fine and well, except the app didn't allow me to return to the verification page after accidentally closing it. The app stated I would have to complete the verification by following the steps in an email. That email never came. When I canceled and attempted to re-book the reservation, the app said those dates were no longer available. Other than that [not minor] annoyance, the app works great.	3	441	23.36	)23-09-10 20:39:5			23.36
7		0b-4665-912f-e1	E B	8ocKZ9IqkW51r	App is great, but I get more than frastrated by lack of sorting options and the fact that it lies about availability. If you want to stay in 1 place it will break it down, but if you look to book each of the options separately, 9 out if the 10 times your entire stay dates are actually open in both places. So WHY not show full stay options?!?! I wish I could sort out partial stay it drives me crazy and makes searching so much longer and confusing.	3	166	23.35.1	023-09-07 5:29:4			23.35.1
8		laf-4966-9dd7-d4	Justin Valerio	IjWwYQdpRyECt	Very unhappy with the app and website in its entirety. A critical infrastructure bug had prevented me from making reservations on a few different credit cards. Contacted support and they said it's a feature to protect the properties from parties, even when this is my first time making a reservation. Only a few credit cards/accounts would work but not others. It seems to me that Airbnb has this 'feature' confused with a bug they refuse to fix. I'm debating ever using Airbnb again.	1	89	23.31	)23-08-04 15:28:5			23.31

Note. The figure above shows the dataset sample of the Airbnb App.

Figure 12 provides the dataset sample of the Airbnb Vacation Rental dataset which includes the details about the Customer reviews and its related information about the Mobile app. In this particular dataset, about 106,465 records of data are fetched.

Figure 13

Lyft Dataset

	reviewId	userName	userImage	content	score	thumbsUpCount	reviewCreatedVersion	at	replyContent	repliedAt	appVersion
0	d6-4b80-aa60-7	Maggie McMahon	JKwABMQgvhb7U	Time estimates for drivers arriving are WAY off. I had one that said 5 minutes and I waited more than double that. If a driver isn't moving or I need to get to an appointment on time and the actual time is too long, they charge a fee for cancellations. Such a consistently poor experience that I'm done using it and will just stick to Uber.	1	0	15.28.3.1697005859	023-10-26 14:55:			15.28.3.1697005859
1	7df-4bea-9d5b-d3	George Monical	UjWwxnV8DSsE	Driver told me stay where I was then took off. I got charged $6.50. Never had this experience with Uber.	1	0	15.29.3.1697609309	023-10-26 14:33:			15.29.3.1697609309
2	18-4021-976c-c8	Steven Szczepanski	gBocKqnpnKoqw	Great ride	5	0	15.28.3.1697005859	023-10-26 14:23:			15.28.3.1697005859
3	e7-407b-b113-e	Allan Davison	-UjXX6ya3B9Kb2	Ç by	4	0	15.28.3.1697005859	023-10-26 14:19:			15.28.3.1697005859
4	01e-4f6c-b580-1	C.E. Mos	BocI0ahIdlpZQnE	What I want to know is I go to destination(no stops) they quote me a price, I always get walt & save, 30 minutes or less call lyft for return to my home(no stops) & it's higher. Not fair	4	0	15.29.3.1697609309	023-10-26 13:35:			15.29.3.1697609309
5	46-4245-8c99-bt	Cameron	gBocIZRP91-I5JF	Got stranded for half an hour because lyft driver left in the middle of a job. No refund.	1	0	15.22.3.1693375853	023-10-26 13:00:			15.22.3.1693375853
6	d1-47bd-8203-5c	Charles Thorkildson	UjW_rHpBP1tRp	Haili hildermann	5	0		023-10-26 12:53:			
7	d5-425a-a8d1-33	charlie yi	JjVzH-NqMZkBL	Awful experience with both driver and Lyft support. My driver took an unusual route and I ended up being 10 minutes later than the promised time and my driver had a large crack in the windshield, which is a safety concern. I spoke to support about this and the support personnel interrupted me repeatedly, was rude to me, dismissed my concerns, made me wait to see whether or not I could be issued a refund, told me Lyft has a no-refund policy, and cut me off and hung up, never resolving the issue.	1	0	15.29.3.1697609309	023-10-26 12:01:			15.29.3.1697609309
8	50-4cee-a683-a2	bryce wister	9ocKvl4Y66IDvXI	after today i will no longer be using Lyft 20$ to go half a mile are you out of your mind?	1	0		023-10-26 10:51:			
9	lcc-4974-8981-9	Bettie Wylie	V-UjVlj3YqNv7Ol	I don't need a ride 1/2 hour after my appt. That's ridiculous & if I cancel right away, Then it's a $10 charge? I don't understand! Please explain…	1	0	15.29.3.1697609309	023-10-26 10:46:			15.29.3.1697609309
10	1f-4710-9506-ee	Terry Walker	8ocKly8oUq8d0s	You ppl are outta your damn minds $42 to go on a 10 min drive? What kinda drugs are yall doing?	1	1	7.70.3.1672813001	023-10-26 10:41:			7.70.3.1672813001
				Lyft charged me for a ride that I didn't take and the driver said he cancelled. No way to get any			6.93.3.1626246				6.93.3.1626246

Note. The figure above shows the dataset sample of the Lyft App.

Figure 13 depicts the dataset sample of the Lyft Car Ride Booking Dataset, consisting of the review information of the Users. This dataset consists of 117,909 records of user review data.

Figure 14

Cheapflights Dataset

	reviewId	userName	userImage	content	score	thumbsUpCount	reviewCreatedVersi	at	replyContent	repliedAt	appVersion
0	9001-408a-906b-687	Micah Medina	gBocKCAV_37I_AAJC	I have gotten some really good deals with this app before. If you are a frugal person, I suggest having a good idea of when and where you want to go before searching. Airline prices are extremely volatile and you can find an excellent deal, but by the time you click on it, it's gone and the price might double in that minute or so that passed, which I find infuriating. Good app, I have been experiencing lots of errors though. I click on view deal and the page won't load or it will say error.	5	340	136.1	021-08-27 14:42:			136.1
1	f-7af6-4621-ad63-0602	Shreeya Sengupta	LV-UjXiAHpCDPQZR	Cheap flights is very helpful when planning trips. They do everything they can, to make sure you arrive to your destination. Easy to use, nice to look at, good customer support. This app is a great app for flight booking. 5star from my side don't know why so many peoples rated this app very low but this app is amazing and I have found many flights for great prices.	5	80	186	023-08-18 18:59:			186
2	25ff-4a78-9410-5e785	Palei Fonua Liua	VCg8ocKhIqta0Bjn47	I'll look for flights from this app, but as far as booking with it...never! I booked one flight with this app once and it passed me through other venues to book with, it was very confusing and untrustworthy. I'd like an app to be able to do everything for me including flights, hotels, and car rental, and never pass me with my information through the rest of the world, it seemed to me.	1	97	184.1	023-07-14 12:27:			184.1
3	a-0316-4a6a-a5f5-b55d	Anifa bby	g8ocLUAQIKJB8FxF	though i have not yet done ticket booking but,i honestly recommend this app on its accuracy.i asked ticket agent office they gave me the same price in this app. congratulations. am trying to see whether I'll manage to do booking with you soon. thanks	5	0	191.2	023-10-24 9:04:3			191.2
4	1e83-41db-a1dd-88ea4	Jandri du Plessis	2g8ocI0NN0OaKjwOJ	Have tried to book a flights at least 6 times and it has to reload everytime. booking isnt going through amd the numerous redirected pages is bs. this site is supposed to make booking easier and its doing the exact opposite	1	8	189.2	023-10-02 23:12:			189.2
5	1-8740-4a2d-adc8-644f8	Ahmad Zaini	UjWBk5s7ZCELUB4	Convenience and easy to find flight with cheap and best options. Would be great if you could integrate public holiday and school holidays into the calendar when selecting the travel time.	5	1	186	023-08-16 4:10:0			186
6	5-ea05-4749-afeb-f1190	Murad Ghunaim	2g8ocKbqZajKrlTV2	After booking through this application and paying the money...the airline employee told me, "Why do you have a name on the system?"...I suffered from a lot of psychological fatigue and problems when traveling because of it.	1	2	186	023-10-04 18:27:			186
7	a-0ceb-4817-8d27-a1ba	Piu Roy	2g8ocIsJ8dcdrkZzLI2	This app is fantastic! I've found many flights for great prices, that you would never think would be legit. It even searches for flight booking sites that rarely get any internet traffic. The only problem I've had with it is when you save a flight, it doesn't save it the way you'd want it to.	5	24	181	023-06-13 5:58:5			181
8	9-c9c0-4890-a80f-b44b	Abhisek Bhattacharjee	-UjWWolThhBODn7I	Cheap flights is very helpful when planning trips. They do everything they can, to make sure you arrive to your destination. Easy to use, nice to look at, good customer support. This app is a great app for flight booking. 5star from my side don't know why so many peoples rated this app very low but this app is amazing 😍 and I have found many flights for great prices.	5	67	182.3	023-06-26 3:48:3			182.3
9	8925-4c71-b036-35111	tomolal	LV-UjV9nFCP7Dni3Ih	App taking too long to load after updates and sadly no option for Kenya, when setting up the Country of origin, it's not working after it loads. "Something went wrong please try again" is the constant message on the app when I try to log in with my email.	5	12	187.1	023-08-28 15:09:2			187.1
10	7cb4-4723-9d90-85d77	Shayeri Agarwal	LV-UjWoANpCPPhfGI	Awesome app! book flights ticket with cheaper rate than other website. Book ticket for my journey! Easy to use. So convenient. Too many options available in cheper rate.	5	5	186	023-08-18 19:02:3			186
11	4-0416-4f3a-856f-f51e2	Bhojraj Rana	KLV-UjUQRs0aKwjbd	The best deals on flight that you can find on the net. I love this app. Thank you for being around whenever I need to fly at a very reasonable fare.	4	4	186	023-08-31 2:18:2			186
12	c-3d31-4ead-946a-1523	saumitra das	LV-UjX2B4JB6PWYw	Exceptional! Easy to search. Places appear before you even complete the search. One word to describe this App: Figures are realistic. Travel information provided. I am really impressed. I highly recommend this App to all travelers!	5	34	182.3	023-06-26 8:02:5			182.3
13	28ec-484b-bd96-6023	Poulami bhattacharrya	KLV-UjXTI_fKG1GrBS	Cheap flights app is a very useful app for everyone...I'm using it for a long time. It's very easy 2 use. Very easy way to book your tickets & go your destination. I highly recommend it for everyone.	5	26	184.2	023-07-24 1:14:5			184.2
14	9e0c-4445-9ecd-e582c	Fatima Ahmed	-UjWHXZx27vmvfM0	I I always booked my flight with a cheap flight, but this time I have a very bad experience,I have been scammed,cheap flight give me option one website BRAVOFLY showing cheap fare and I book flight from Karachi to Istanbul, dated 1st of September from ETIHAD AIRLINE, I make payment 28th of August, after 10 minute I received a message your flight is not confirm because airline increase the price, SINCE THEN I MESSAGE TO BRAVOFLY THEY DIDNT RETURN MY MONEY,	1	3	187.1	023-09-12 12:14:			187.1

Note. The figure above shows the sample records of Cheap flights Dataset.

Figure 14 depicts a dataset sample of the CheapFlights App in the Flight Booking Domain. The above dataset of the Cheapflights App consists of a total of 14,113 rows of data.

Figure 15

GrubHub Dataset

	A	B	C	D	E	F	G	H	I	J	K	L
		reviewId	userName	userImage	content	score	thumbsUpCount	ReviewCreatedVersi	at	replyContent	repliedAt	appVersion
0		9e-4d86-a7ac-ad	Elizabeth Fuller	:g8ocLsJJbITFpY	Tracking map is extremely annoying: lets you zoom in or out, but quickly resets to show only a few blocks around the car. I always want to see both the car and my place, & before pickup I want the restaurant too. The only point of this feature is for the customer's information--not for GrubHub, or driver, or restaurant. No reason in the world for it to override user's choice. Never seen another tracking app that thinks it knows better than the user. More like a power trip than help.	4	35	2023.38	)23-10-14 16:09:3			2023.38
1		3a4-47ff-ac04-9f1	Rob Youngberg	,V-UjUu7m8kUjLI	I logged in, home address already there, good. I clicked the "Hamburger" category. The chain near my house that my family wanted popped up. I clicked on it, made my order, and submitted. This is when I noticed the order wasn't with the restaurant a few minutes away, it was one 20+ minutes away, 2 cities over. Oops, I didn't check the address, but why would that be the 2nd search result, when there are 5 closer ones? Cold food. Otherwise, the app is great, just that one complaint. Thanks.	3	0	2023.4	)23-10-24 21:07:1			2023.4
2		lb4-48a7-a146-cl	Alec Walker	ocJFGgtF0SGSX	The app is very sloppy. They haven't bothered to make error messages correspond to specific errors, rendering troubleshooting a waste of time. For example, no matter what payment method I enter, I get an error that there is a problem with the payment method and should try again. (Venmo, PayPal, my credit card, friend's credit card, debit card, etc.)	1	33	2023.39	023-10-16 8:47:0			2023.39
3		63-4c71-a94c-7!	Brandy Peightal	UjXS4cEazCVFq	Subpar. The last 5 or so orders that I've placed through this app have either been wrong, canceled, or undelivered. I pay delivery fees and extra fees on top of the regular menu fees to have orders delivered, but no one bothers to check the order or deliver the correct order. And the only thing you can do is get an item or the order refunded. This isn't helpful when you have a house full of hungry people and one or more people don't receive the correct item and/ or anything at all. It just sucks	2	23	2023.4	)23-10-21 16:57:1			2023.4
4		f7-49a2-991b-93	Andrew Lane	ocILKe2qkBB2C	I'd rate the service fairly highly, but the app is quite poorly designed and difficult to use. I don't remember it always being this way. Can't switch between delivery and pick up without starting over. When you search for a restaurant or filter by category, you're stuck with the search result on the restaurant menu, and unable to see all the options. It's very hard to find some restaurants without searching.	3	113	2023.34	023-09-11 9:22:2			2023.34
5		8d-4d62-99c4-72	Carlos Hernande	UjWOeS0IAPCuI	I like the Grubhub app because it is slick and easy to navigate. It has a lot of options for different cuisines and restaurants, and I can filter them by price, rating, delivery time, and more. The app also shows me the estimated delivery time and the status of my order in real time. However, the app is not perfect. Sometimes, I encounter unknown errors when I try to check out. This is very frustrating, especially after I spent time choosing a place and going through all the menu items.	3	10	2023.34	)23-09-11 21:08:1			2023.34
6		5f-4995-bc2f-ad	j d	-UjVkojBHTOxA7	Convenient for when you can't step out. Plus membership is worth it in the long run if you use them frequently. Have had some issues with the app itself with giving me location options that aren't nearby (other cities) so be mindful of that when ordering. Last few updates has an issue where I am unable to empty the cart. It gives an error everytime. Only way around it is by placing something from another place in its place but you're still left with that in your cart if you don't order.	3	145	2023.3	)23-08-09 15:12:(			2023.3

Note. The figure above shows the sample records of the GrubHub Dataset.

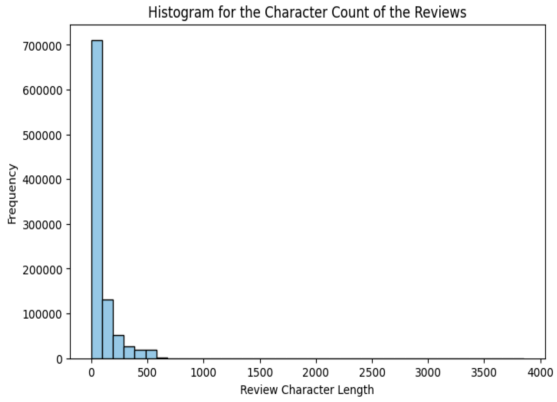
Figure 15 depicts the dataset sample of the Grubhub App used for Food Delivery. The dataset as a whole consists of 128,753 rows of data depicting user's opinion data on the application.

Data Pre-Processing

The raw data from the Flight Booking, Vacation Rental, Food Delivery, and Car Ride booking domains comes to 959491 rows containing the ReviewID, UserImage, UserName, Content, Score, ThumbsUpCount, ReviewCreated Version, at, replied at and appVersion. On average, there are 47975 rows in each file extracted. The content column contains the user reviews, which have an average length of 100 characters and have less than 1% of special characters.

Figure 16

Characters Count in the Customer Reviews

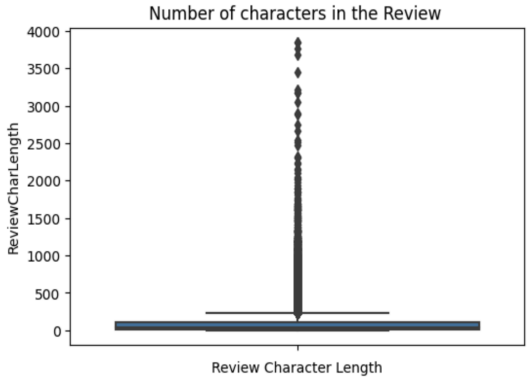


Note. The figure above shows the histogram displaying the character count of the reviews.

Figure 16 displays that the most of the review contents fall in the first bin ranging from zero to 500 characters and very few records greater than 500 characters. The number of reviews having a character count of less than 500 is 957,599 whereas there are just 1892 reviews with a character count greater than 500.

Figure 17

Number of Characters in Customer Reviews

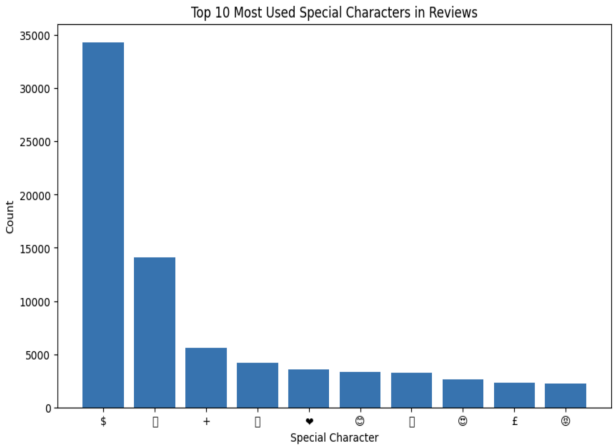


Note: The box plot illustrates the number of characters in the content column.

Figure 17 shows a box plot that displays the number of characters in the reviews where the outliers in the character length can also be seen.In total, there are 1892 records with more than 500 characters in the review data, which can be considered as the outliers in the above plot.

Figure 18

Count of Top 10 Special Characters



Note. The bar graph displays the top 10 special characters in the content column.

Figure 18 shows the count of the most used special characters present in the dataset, where the special character with the highest count is '$' with a count of 34334. All these special characters are dealt with differently when performing the data cleaning and transformation.

Data Cleaning

Handling Missing Data. This step includes the removal of null and missing values in the dataset. In order to handle the null values, the presence of null values is checked in the dataset with the help of the isnull() function. After identifying the null values, depending on the type of column, the null values in each column are handled by imputing the missing values with a forward fill or backward filling approach, removing records where the volume is less than 10%,

and columns that had large volumes of null values are not removed considering the dependencies on the other columns.

Figure 19

Records Count in the APP

```
Number of Records in the APP:  959491
```

Note. The output of the total records in the dataset.

Figure 19 shows the count of the number of records in the merged dataset of the 20 applications, which is 959491.

Figure 20

Check for Presence of NULL Values

```
Are there any null values in the Dataset? :  True
```

Note. The figure above shows the presence of Null values.

Figure 20 depicts the presence of Null values in the dataset and shows the boolean result as true.

Figure 21

Count of Null Values in Dataset

```
The number of null values in the Dataset:
 Unnamed: 0               75633
reviewId                      0
userName                      3
userImage                     0
content                     728
score                         0
thumbsUpCount                 0
reviewCreatedVersion     111123
at                            0
replyContent             821314
repliedAt                821314
appVersion               111123
AppName                       0
App Domain                    0
dtype: int64
```

Note. The figure shows the count of null values in different columns.

The total number of null values in each column of the dataset is displayed in Figure 21 below. The majority of the columns that contain user information are free from null values. Figure 22 shows the removal of null values from the Content and userName column.

Figure 22

Removal of Null Values from Content and UserName Column

```
The number of null values in the Dataset:
 Unnamed: 0               75487
reviewId                      0
userName                      0
userImage                     0
content                       0
score                         0
thumbsUpCount                 0
reviewCreatedVersion     111077
at                            0
replyContent             820595
repliedAt                820595
appVersion               111077
AppName                       0
App Domain                    0
dtype: int64
```

Note. In the above figure, it can be seen that there are null values in both columns.

There is a very small count of null values in these columns, hence these values are dropped using the dropna() function. After the drop, it can be seen that there are no remaining values that are null in these columns. Figure 23 shows the imputation of values in the appVersion column with the help of Forward Filling and Backward Filling.

Figure 23

Handling Null Values in AppVersion Column

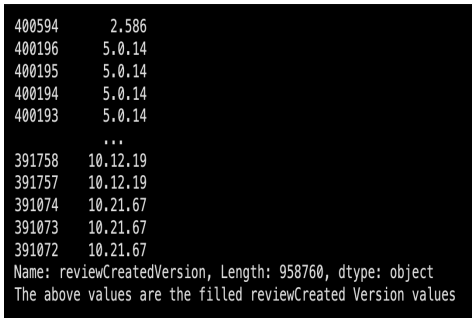
```
400594      2.586
400196     5.0.14
400195     5.0.14
400194     5.0.14
400193     5.0.14
           ...
391758    10.12.19
391757    10.12.19
391074    10.21.67
391073    10.21.67
391072    10.21.67
Name: appVersion, Length: 958760, dtype: object
The above values are the filled App version values
```

Note. The figure above shows the imputed values for all the null values in the App Version.

Figure 24 shows the imputation of values in the reviewcreatedVersion column with the help of Forward Filling and Backward Filling.

Figure 24

Handling Null Values in reviewcreatedVersion column

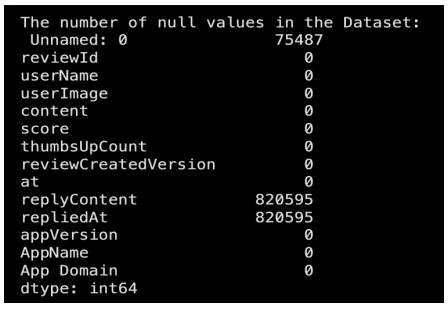


Note. The figure above shows the imputed values for all the null values in the Review Created Version column

The null values in the app version and Review Created Version have been handled by initially sorting the records by the review created column following which the null values are imputed by forward and backward filling the application versions specified in the rows below and rows above. Figure 25 shows the absence of null values in the two columns.

Figure 25

Final Status of Removal Null Values from the appVersion and reviewcreatedVersion Column



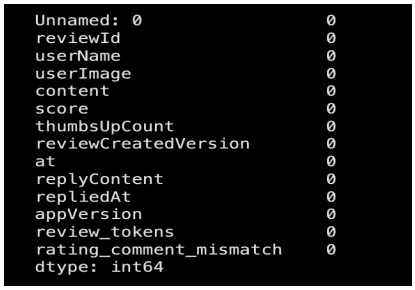
Note. The above figure shows the removed null values from the appVersion Column and reviewcreatedVersion Column.

It can be seen in Figure 25, that there are still null values left in the Unnamed:0, replyContent, and repliedAt columns. The replyContent is expected to have null values because not all comments have a reply comment associated, consequently, the repliedAt also has null values since both columns are interdependent. The null values in the replyContent column and repliedAt column are filled with a default NA value, since the replyContent and repliedAt columns have records with valid review data, it is decided to keep the records with Null values in the two columns. Since, the scope of the project is just to focus on the review data, even though there are null values in the two columns does not cause a big concern. These columns are removed in the Data reduction phase.

Since the Unnamed:0 column mostly consists of floating point numbers which are not useful according to the scope of the project. For this reason, NA has been inserted for the null values for the Unnamed columns to resolve the issue. Figure 26 shows the final status of the presence of null values in the dataset. Figure 27 shows the number of records left after removing the null values.

Figure 26

Final Status of the Null Values in the Dataset



Note. Using the above figure, it can be seen that all the Null values in the dataset have been removed.

Figure 27

Number of Records after removing the Null Values

```
The number of records left after removing all the null values: 958760
```

*Note.*The above figure shows the total number of records after removing the null values.

Handling Noisy Data. In this step, all the noise-related data have handled any of the inaccurate, inconsistent, or irrelevant data in the Raw files extracted. Figure 28 shows the removal of the Emoji data from the content column.

Figure 28

Removing the Emoji Data

```
# Filter the DataFrame to include only rows with emoji
emoji_rows = df1[df1['content'] == True]

# Print the rows with emoji
print(emoji_rows)

Empty DataFrame
Columns: [Unnamed: 0, reviewId, userName, userImage, content, score, thumbsUpCount,
Index: []
```

Note. The figure above shows the records after removing all the emoji data to text data more amenable to modeling.

In order to identify the mismatch between User's Score and Review Content a lambda function is applied to the dataframe that checks if the score value is less than three and the review content mentions "good" keyword then there is a mismatch record. Similarly if the score value is greater than three and the content column has the keyword "worse" then it is identified

as a mismatch record. Figure 29 the number of records that are found as the mismatch between the records.

Figure 29

Mismatch between Score and Content

```
The number of mismatch between the Review content and Reviewer Score: 8469
```

Note. It can be seen that 8469 records have a mismatch between the Review content and the User rating that is provided for the app.

This scenario is possible in cases where the User's review text implies a negative sentiment about the app, but the rating given by the user suggests otherwise or when the User rating is low whereas the User Review indicates a Positive Sentiment about the App. Figure 30 shows the status of Hypertext Markup Language(HTML) tags in terms of boolean value after the removal.

Figure 30

Handling HTML Tags

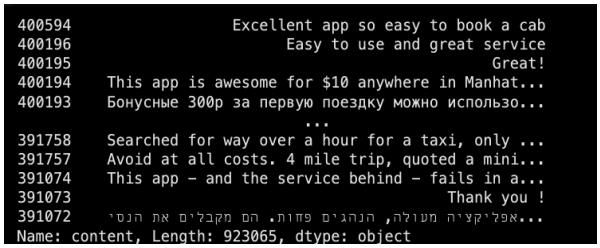
```
Is there any HTML Tags in the Text?:
 400594    False
400196    False
400195    False
400194    False
400192    False
          ...
391759    False
391758    False
391757    False
391074    False
391073    False
```

Note. The above figure shows the removal and handling of HTML tags in the Review Content

The content column also contains special characters, which are removed with the help of a user defined function which identifies all the special characters and replaces it with a blank space. Figure 31 shows the status of the content column after the removal of special characters.

Figure 31

Removing of Special Characters

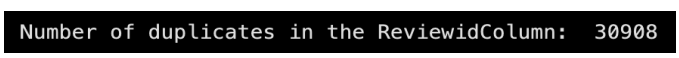


Note. The figure above shows the sample records after removing the special characters from the content column.

Handling Inconsistent Data. This part of the Data Cleaning handles duplicate values and inconsistencies in the data format and values. For the duplicate values, the duplicate values in the content and ReviewId columns are identified, analyzed, and resolved accordingly. The column “at” having the timestamp values has an inconsistent format, which is also resolved. The inconsistent format in the “Content” column consisting of the review text, has mixed case letters which are converted into lower case. This process is already performed while handling Noisy Data and Text Preprocessing. Figure 32 shows the number of duplicates in the review id column.

Figure 32

Handling Duplicates in Review Id



Note. The above figure shows the number of duplicates that are present in the User's Review ID column. There are about 30908 duplicate records in this column. These duplicates are removed with the help of drop_duplicates() function. Figure 33 shows the number of records left after removing the duplicates in the Reviewid Column

Figure 33

Remaining Records in Review Id Column

```
The number of records  after removing duplicates in the review id column:  927852
```

Note. It can be seen that there are 927852 records left after removing duplicates in the ReviewID column.

Figure 34 shows the number of duplicate records in the Review Content Column. It can be possible that more than one User gave the same review like "Good APP" which can explain the presence of duplicates in the Content Column. Hence, the duplicates are not dropped in the Review Content Column, as it represents a valid User Review. Hence the duplicate records in the review content columns are not considered to be removed, as it contains valid user information.

Figure 34

Checking Null Values in Review Content

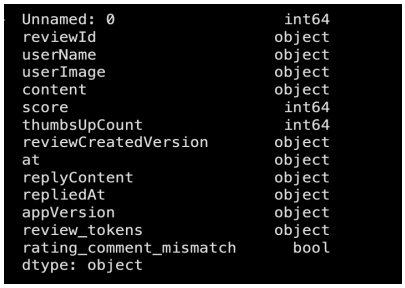
```
Number of duplicates in the Content Column:  256575
```

Note. The above figure shows that there are duplicate values in the Content column.

It can be seen in Figure 35, that the "at" column is of the object data type, which needs to be converted to standard Date time. Figure 36 shows the conversion of the "at" column to the standard Date and Time format.

Figure 35

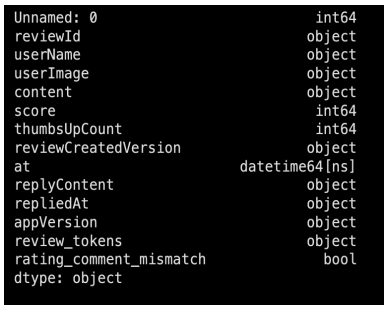
Handling Inconsistent Date Time Format



Note. The column having the date and time value is the “at” column.

Figure 36

Resolved Date Time Format



Note. The figure above shows how the “at” column is converted into standard Date and time format.

The next task is to remove inconsistencies in the text data, which is present in the Review Content column in the dataset. The steps mentioned below are applied to the content column to handle inconsistency in the text data. The output is shown in Figure 37.

Conversion of Text into Lowercase. In this technique, the text data is converted into lowercase which helps in equal treatment of words with different cases.

Removal of Special Characters and Punctuation. With the help of this technique, all the special characters and punctuation are removed from the text. This is useful with a focus on the appropriate text data which is required for the analysis.

Tokenization. This is an important step in Natural Language Processing. In this, the whole text is divided into different tokens or words.

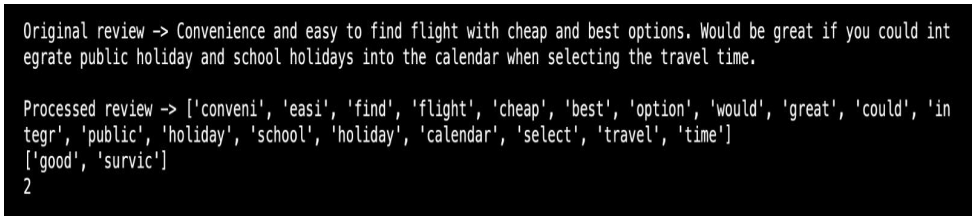
Removal of Stopwords. This involves removing the words that do not add much context to the text or any significant meaning to the text.

Lemmatization/Stemming. This technique is useful while handling variations by converting the words present in the text into their base form

Removal of non-English words. This involves the removal of all the words written in foreign non-English languages from the review content column, which helps in focusing on just English language content while analyzing the review content. The sample code for handling the inconsistencies in the text data is added to Appendix B

Figure 37

Status of Content Data



Note. The figure above shows the one user review content before and after implementing the series of steps involved in data inconsistency.

Data Transformation

Data Transformation is a crucial part of a Data Engineering plan. Once the raw data is cleaned it needs to be transformed properly using various Data Transformation techniques to make the dataset well suitable to train the models. It deals with modifying the cleaned dataset by changing the structures or values of some of the features so that the data makes more sense and it can help in training the model better and accurately.

It seeks to raise the general quality of the data so that the models can provide predictions with more accuracy. Data smoothing, aggregation, standardization, normalization, discretization, regularization, and reduction are some of the most popular and practical data transformation techniques.

Data Discretization

As the name suggests Data Discretization helps in changing the data from Continuous to Discrete. It is done to categorize the data so that it has a limited number of intervals and lesser complexity. The ‘score’ column in the dataset defines a value that is given to the review made by the user for any particular app.

The range of the feature lies from zero to five, with five being the highest score a review can get. Those continuous scores are converted to discrete categories where the score value ‘0‘, 1’ and ‘2’ falls under the category ‘Negative’, the score value ‘3’ falls under the category ‘Neutral’ and the score value ‘4’ and ‘5’ falls under the category ‘Positive’. Figure 38 shows the discretization of the Score column to create a categorical feature which is a Category Column, representing the user sentiment based on the User’s Score.

Figure 38

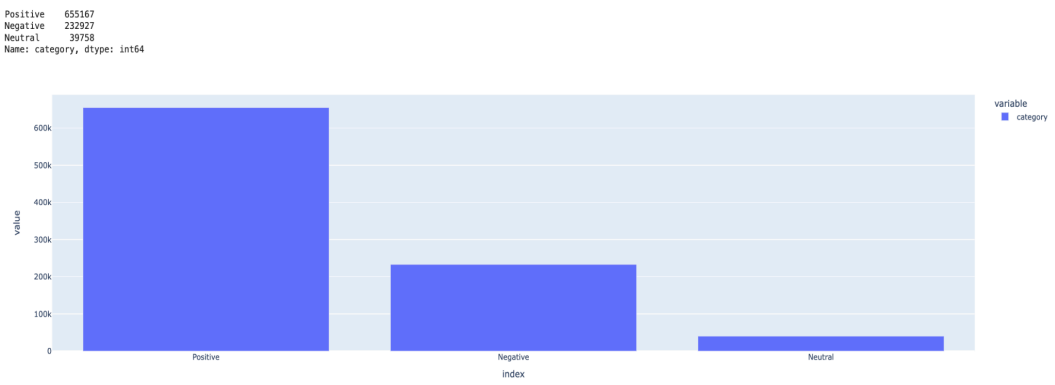
Discretized the Score

unt	reviewCreatedVersion	at	replyContent	repliedAt	appVersion	review_tokens	rating_comment_mismatch	ReviewLength	processed_content	category
0	2.0.602	4/22/14 6:10	NaN	NaN	2.0.602	['love']	False	7	[love]	Positive
3	2.0.603	4/25/14 13:53	NaN	NaN	2.0.603	['fantastically', 'easy', 'says', 'tin', 'anno...	False	198	[fantastically, easy, say, tin, annoyence, cli...	Positive
0	2.0.603	4/27/14 7:11	NaN	NaN	2.0.603	['much', 'expensive', 'booking', 'online']	False	40	[much, expensive, booking, online]	Negative
0	2.0.603	4/30/14 6:29	NaN	NaN	2.0.603	['useful']	False	11	[useful]	Positive
0	2.0.603	5/1/14 4:55	NaN	NaN	2.0.603	['ok']	False	2	[ok]	Neutral

Note. The output shows that a new column ‘category’ is created which differentiates the score into three different categories. Figure 39 distribution of the three categories in the dataset, with the highest number of the positive sentiment, followed by negative sentiment, and the lowest number is the neutral sentiment.

Figure 39

Bar Plot for Category Distribution



Note. The visualization shows that the positive category has the highest frequency suggesting that the majority of the users overall liked the app.

Data Augmentation

As can be observed from Figure 39 the data is highly imbalanced, leaning towards positive reviews. So, to handle the imbalanced data, the Synthetic Minority Oversampling Technique (SMOTE) technique is utilized, which is shown in Figure 40 where around eight lakhs of new synthetic data is generated. The sample code to perform the SMOTE technique on training data is given in Appendix B.

Figure 40

SMOTE Resampling

```
Original Training Set Shape: (723363, 100) (723363,)
SMOTE Resampled Training Set Shape: (1539657, 100) (1539657,)
```

Note. The figure above shows the SMOTE resampling, the resampling is done to handle the imbalance in the training data.

Figure 41

SMOTE Distribution of Data

```
Class distribution in y_train_smote: {0: 513219, 1: 513219, 2: 513219}
```

Note. To validate the distribution, Figure 41 displays that the distribution in the training set for the category column, in which ‘0’ represents negative, ‘1’ represents neutral and ‘2’ represents positive reviews, are in the same proportion (0: 513219, 1: 513219, 2: 513219). This helps the model to perform well in minority classes (here, negative and neutral) and make robust learning.

Data Normalization

Feature Scaling. The technique converts the numerical data columns into a common range of values. In the given dataset, the feature scaling is carried out for the ‘ThumbUpCount’ and ‘ReviewCharLength’ columns, which can be seen in Figure 42.

Figure 42

Scaling the Numerical Features ThumbsUpCount and ReviewCharLength

thumbsUpCount	reviewCreatedVersion	at	replyContent	...	appVersion	AppName	App Domain	ReviewCharLength	special_characters	processed_content	category	word_count	rating_comment_mismatch	has_html_tags
0.000000	2.586	2014-01-01 10:05:00	NA	...	2.586	gett_taxi	Car Ride	0.008833	[]	[excel, app, easi, book, cab]	Positive	8	False	False
0.003998	5.0.14	2015-01-01 01:08:00	NA	...	5.0.14	gett_taxi	Car Ride	0.007275	[]	[easi, use, great, servic]	Positive	6	False	False
0.000000	5.0.14	2015-01-01 10:13:00	NA	...	5.0.14	gett_taxi	Car Ride	0.001299	[]	[great]	Positive	1	False	False
0.000000	5.0.14	2015-01-01 11:03:00	NA	...	5.0.14	gett_taxi	Car Ride	0.032736	[$, $]	[app, awesom, number, anywher, manhattan, also...	Positive	24	False	False
0.000000	5.0.14	2015-01-01 16:28:00	Thank you very much (:	...	5.0.14	gett_taxi	Car Ride	0.002858	[]	[good, survic]	Positive	2	False	False

Note. In the above figure, it can be seen that thumbsUpCount and ReviewCharLength have been converted to a standard form.

Data Reduction

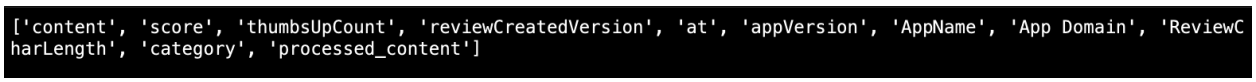
With high dimensional data comes different problems such as overfitting, computational complexity, and the curse of dimensionality. Data Reduction focuses on reducing the unnecessary features, which do not contribute to the prediction of the target variable up to a certain extent. Only the most informative features are kept and the less informative or redundant features are removed.

The important features such as ‘score’, ‘thumbsUpCount’, ‘processed_content’, and ‘appVersion’ have a direct relationship with the target variable and serve an important role in

training and testing the models. There are irrelevant features like Unnamed:0, personal user data such as ‘UserName’, ‘UserId’, ‘UserImage’, ‘reviewId’, and other irrelevant features of user’s reply data such as ‘replyContent' and ‘replied at’. All these features are removed to maintain the user's privacy, as shown in the figure below, and to avoid the complexity while training the model. During the process of Data Cleaning and while performing the initial EDA, new columns are added to the dataset such as special_characters, rating_comment_mismatch, and has_html_tags. All the irrelevant features are removed from the dataset. Figure 43 shows the list of remaining columns after the removal of the irrelevant columns from the dataset.

Figure 43

Resulting Dataset Columns



Note. The figure above shows the resultant dataset after Data Reduction. Informative descriptive features are kept and the rest of them are dropped.

After dropping the unwanted columns Principal Component Analysis is performed to further reduce the dimensionality of data. Figure 44 displays numerical features ‘score’, ‘thumbsUpCount’, and ‘ReviewLength’ are removed from the dataset and two new features PC1 and PC2 are added which represent the original numerical features. The main aim to perform PCA is to reduce the dimensions while keeping the majority of the information as it is.

Figure 44

Performing PCA

	content	reviewCreatedVersion	at	appVersion	AppName	App Domain	processed_content	category	PC1	PC2
400594	Excellent app so easy to book a cab	2.586	2014-01-01 10:05:00	2.586	gett_taxi	Car Ride	[excel, app, easi, book, cab]	Positive	0.178954	-0.001898
400196	Easy to use and great service	5.0.14	2015-01-01 01:08:00	5.0.14	gett_taxi	Car Ride	[easi, use, great, servic]	Positive	0.391247	-0.074373
400195	Great!	5.0.14	2015-01-01 10:13:00	5.0.14	gett_taxi	Car Ride	[great]	Positive	-1.170649	0.171246
400194	This app is awesome for $10 anywhere in Manhat...	5.0.14	2015-01-01 11:03:00	5.0.14	gett_taxi	Car Ride	[app, awesom, number, anywher, manhattan, also...	Positive	2.275503	-0.012608
400192	Good survice	5.0.14	2015-01-01 16:28:00	5.0.14	gett_taxi	Car Ride	[good, survic]	Positive	2.215797	-0.736695
397956	Best	8.3.28	2017-01-01 06:06:00	8.3.28	gett_taxi	Car Ride	[best]	Positive	-1.170649	0.171246
397954	Tried to use it twice. Both times it told me "...	8.3.28	2017-01-01 11:37:00	8.3.28	gett_taxi	Car Ride	[tri, use, twice, time, told, number, minut, w...	Negative	-1.075957	0.159071
397953	It was cold and rainy and my foot hurt. I got ...	8.3.28	2017-01-01 12:18:00	8.3.28	gett_taxi	Car Ride	[cold, raini, foot, hurt, got, phone, got, get...	Positive	-0.927523	0.140065
395956	The best way to get a taxi	9.9.41	2018-01-01 03:29:00	9.9.41	gett_taxi	Car Ride	[best, way, get, taxi]	Positive	-1.120616	0.164817
395955	I don't "gett" this app!! Signed up ot takes m...	9.9.41	2018-01-01 12:50:00	9.9.41	gett_taxi	Car Ride	[gett, app, sign, ot, take, weird, screen, get...	Negative	-1.022215	0.152240
395954	Wish we knew at start of our trip! Credit card...	9.9.44	2018-01-01 21:02:00	9.9.44	gett_taxi	Car Ride	[wish, knew, start, trip, credit, card, cash, ...	Positive	2.133308	0.259327
394709	Can't use a German credit card while being in ...	9.27.41	2019-01-01 01:11:00	9.27.41	gett_taxi	Car Ride	[ca, use, german, credit, card, israel]	Negative	-1.149152	0.168514

Note. Columns PC1 and PC2 are added. They are completely uncorrelated and orthogonal with each other and are important features for modeling as they are linear combinations of the original numerical features.

Data Regularization

Data Regularization is a technique that is used to modify data so that it falls within a given range or structure. It makes sure that the data has similar distributions and takes care that the data with higher magnitude should not be overshadowed so that it can easily train the machine learning models and provide accurate results. In this project, the dropout method is used to handle overfitting, as neural networks are used to build deep learning models, which incorporate dropout methods to perform data regularization.

Feature Engineering

Word Embedding. To convert text into meaningful weightage of the word, a word embedding technique called “Word2vec” is performed on the “Processed_Content” column,

which contains a list of contents of each review. Once the word embedding is performed on the column, the weights of each new feature that are extracted by Word2vec are added to the main dataset.

Several parameters such as vector_size = 100 (every word represented as a 100-dimensional vector), window = 5 (determines the number of neighboring words around the target that can be considered as a context), and min_count = 1 (includes words that appear at least once in the corpus) is passed to achieve the following result that is shown in Figure 45 and Figure 46. The sample code to perform the Word2Vec technique is provided in Appendix B.

Figure 45

Word2vec Result

	vector_0	vector_1	vector_2	vector_3	vector_4	vector_5	vector_6	vector_7	vector_8	vector_9	...	vector_90	vector_91	vector_92	vector_93
0	-0.622932	-0.614061	0.353251	-0.662969	1.302122	-0.662272	1.623627	-0.520460	-0.496635	1.057477	...	-0.577306	-0.178559	-0.434975	-0.936239
1	-0.266236	-0.752602	0.218425	0.729899	1.121586	0.176447	1.332617	0.303797	-0.902290	1.094965	...	-0.427490	-0.815889	-0.196273	-0.943165
2	-2.771326	0.387453	-0.044485	0.686026	1.619579	-0.001892	1.339186	-0.605034	-1.206421	0.520948	...	-0.943030	-0.140151	-0.151959	-1.731853
3	-0.027754	-0.805020	0.324245	1.020163	-0.046569	-0.324235	0.922441	-0.392187	-0.813908	0.350461	...	-0.530611	-0.307605	-0.656667	0.148543
4	-1.082113	0.053775	-0.134807	0.225807	1.001085	0.424071	0.717633	-0.058960	-0.581260	-0.144710	...	-0.190452	-0.146599	-0.094695	-0.486356

5 rows × 100 columns

Note: The figure represents the average of the vectors of the words that each of the reviews contains.

Figure 46

Shape of the Performed Word2vec

```
Shape of X: (904204, 100)
```

Note: This is the shape of the Word2vec result, where 904204 are the total instances, and 100 represents the number of features (100-dimensional vector)

Topic Modeling. Using Latent Dirichlet Allocation (LDA), tokenized text is first preprocessed into a vectorization-ready format and then utilized for sophisticated topic modeling in a text corpus of data. The CountVectorizer effectively records word frequencies and eliminates frequently occurring stop-words from the cleaned text before converting it into a document-term matrix (DTM).

The LDA model can statistically identify latent topics after it has been learned on this matrix, which teaches it the distribution of words across topics and their relative prevalence in each document. After a comprehensive analysis of the model's output, the terms that are most relevant to each topic are determined, producing unique topical insights. After running the code, the dataset is categorized into the top 10 categories from which the project is focused on topics '2', '4', '6', and '9' shown in Table 13.

To connect the study with certain areas of interest, such as "Customer Service", "Payment Experience", "App Functionality" and "User-Friendly" papers are also categorized according to their prominent subjects. The corpus used for LDA is generated by using Dynamic Topic Models(DTM) as the use of LDA approach is to classify each content review in four different categories and to come up with a new column "topic_label", which is shown in Figure 47. Wordtovec is used to generate the vocabulary by its algorithm to train, validate and test models. The sample code for Topic Modeling is added in Appendix B.

Table 13

Top 10 Hidden Topics in Dataset

Topic No.	Content
1	['driver', 'ride', 'lyft', 'uber', 'time', 'friendly', 'user', 'minute', 'better', 'car']
2	['app', 'work', 'super', 'update', 'ca', 'time', 'issue', 'cool', 'use', 'problem']
3	['great', 'love', 'app', 'service', 'amazing', 'deal', 'thanks', 'wonderful', 'experience', 'fantastic']
4	['service', 'customer', 'money', 'refund', 'support', 'worst', 'bad', 'help', 'app', 'company']
5	['delivery', 'app', 'fee', 'restaurant', 'option', 'like', 'need', 'service', 'apps', 'food']
6	['easy', 'use', 'awesome', 'excellent', 'fast', 'quick', 'app', 'simple', 'ok', 'navigate']
7	['hotel', 'app', 'best', 'booking', 'price', 'book', 'flight', 'helpful', 'room', 'agoda']
8	['good', 'app', 'far', 'really', 'airbnb', 'experience', 'like', 'useful', 'host', 'place']
9	['app', 'card', 'nice', 'account', 'use', 'payment', 'ca', 'phone', 'credit', 'code']

Topic No.	Content
10	['order', 'food', 'time', 'delivery', 'driver', 'hour', 'restaurant', 'app', 'ordered', 'grubhub']

Note. The figure above shows the topics identified in the Content Column.

Where Topic ‘2’ is classified as “app functionality” in the dataset. It seems that the usability and technical features of apps are the main focus of this topic. Feedback regarding app updates ('update'), functionality ('work', 'use'), and particular problems or issues ('issue', 'trouble') is probably included. Both positive and negative meanings can be associated with the terms "super" and "cool" (e.g., "super easy" vs. "super slow").

Topic ‘4’ is classified as “customer service”. This topic has a negative bias and is centered on bad customer service encounters. Words like "worst," "bad," "refund," and "money" allude to problems with payments, refunds, or general discontent with the business. "Help" and "support" allude to dealings with customer service divisions.

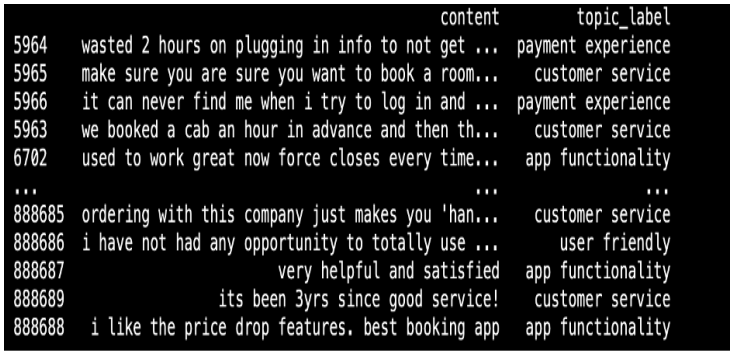
Topic ‘6’ is classified as “user-friendly”. The usability and interface of apps are highlighted in this discussion. It is implied by terms like "easy," "simple," "fast," and "navigate" that consumers believe the software to be effective and user-friendly. Positive generalizations such as "excellent" and "awesome" may be associated with the overall effectiveness of the program.

Topic ‘9’ is classified as “payment experience”. This time, the emphasis is on the apps' ability to conduct financial transactions. 'Card', 'account', 'payment', and 'credit' signify conversations regarding account management and payment options. "Code" might have to do

with security features or promo codes. The sample code for Topic Label Assignment is added in Appendix B.

Figure 47

Topics Labels



	content	topic_label
5964	wasted 2 hours on plugging in info to not get ...	payment experience
5965	make sure you are sure you want to book a room...	customer service
5966	it can never find me when i try to log in and ...	payment experience
5963	we booked a cab an hour in advance and then th...	customer service
6702	used to work great now force closes every time...	app functionality
...	...	...
888685	ordering with this company just makes you 'han...	customer service
888686	i have not had any opportunity to totally use ...	user friendly
888687	very helpful and satisfied	app functionality
888689	its been 3yrs since good service!	customer service
888688	i like the price drop features. best booking app	app functionality

Note. The above figure shows the Topic Labeling of the common themes found in the review content. In total 10 hidden topics are identified, out of which four topics are selected based on the scope of the project.

Data Preparation

Once the dataset is completely Cleaned and Transformed, Data Preparation is carried out. The transformed data set has 905496 records and 10 features which are then split into three different subsets i.e. Training, Validation, and Testing set, which is utilized. The split is done using the train-test split library, maintaining the 80 -10 -10 ratio for the training, validating, and testing respectively. The main purpose of the training set is to train the model so that it understands the data and the relationships among them. A testing set is used to check the accuracy and measure the performance of a model once it has been trained on the training set. Lastly, the validation set is used to overall generalize the model and to check if the model is

predicting values correctly. Figure 48 shows the splitting ratio of training, validation and test set after the split.

Figure 48

Splitting Ratio for Transformed Dataset

```
Training Set Shape (X_train, y_train): (723363, 100) (723363,)
Validation Set Shape (X_val, y_val): (90420, 100) (90420,)
Test Set Shape (X_test, y_test): (90421, 100) (90421,)
```

Note: The above-mentioned diagram shows that there are 723363 instances in the training set and every single feature is represented by a feature vector of length of 100.

In the validation and testing data set, there are 90420 instances with a feature vector length of 100 which fulfills the 80-10-10 split. The absence of the other number indicates that y_train, y_val, and y_test are a single-dimensional array and they only have a single label or are classified into a single category. As there is a data imbalance, SMOTE Analysis is performed on the training set which is mentioned in Figure 39. Figure 40 shows the records counts after performing SMOTE Analysis.The number of records in the training set increased from 723363 to 1539657. Figure 49 shows the sample of the training dataset after performing the Word2vec embedding technique.

Figure 49

Sample of Training Set

```
First 5 instances of X_train:
[[-1.40536976e+00  1.26206708e+00  2.09724545e-01 -1.02606761e+00
   2.21358728e+00  1.99909329e-01 -6.72626555e-01 -3.11761916e-01
   1.21886361e+00 -2.16564581e-01 -9.07086372e-01  4.88561749e-01
   1.27950713e-01 -1.10249531e+00 -1.00109112e+00  1.04161346e+00
  -1.66694924e-01 -6.65995836e-01  5.04570454e-02  2.29065418e-02
  -1.10956110e-01  1.02814949e+00  2.30827704e-01  3.12677115e-01
   1.07912195e+00  8.99489939e-01  1.08713758e+00 -1.92543352e+00
  -2.65231460e-01 -1.22936559e+00 -6.16579652e-01 -6.75708532e-01
  -7.26448715e-01 -1.42187500e+00  7.95624793e-01 -7.89055109e-01
   6.86828494e-02 -7.52020836e-01 -4.77962822e-01  1.11717546e+00
  -3.34484488e-01 -2.15435505e-01 -2.79110581e-01 -3.61343771e-01
   4.34985429e-01  2.28873894e-01 -2.20483318e-01 -1.45096266e+00
   6.25279844e-01 -1.37928450e+00  1.44022334e+00 -2.92056918e-01
  -4.94113207e-01  4.50608850e-01  1.30554020e+00 -4.88687515e-01
   3.56733233e-01  7.02371299e-01  2.38630921e-02 -4.76061016e-01
  -1.02356577e+00  1.41900551e+00 -6.82242393e-01  1.74456522e-01
```

Note: The numbers represent vectors for the words in the sentences of the training set after Word2vec is performed.

The positive, negative,neutral train categories are represented as ‘2’, ‘0’ and ‘1’ respectively. Figure 50, shows the first sentence is positive, the second is negative, the third is positive, the fourth is positive, and the fifth is also positive.

Figure 50

Train Category Column

```
First 5 instances of y_train:
[2 0 2 2 2]
```

Note. The above figure shows the train category for the first five instances of the training set.

The sample records in the validation set is shown in Figure 51.The numbers represent vectors for the words in the sentences of the validation set after Word2vec is performed.

Figure 51

Sample of Validation Set

```
First 5 instances of X_val:
[[ 3.35540175e-02 -8.48125160e-01  4.80289459e-01  6.57782778e-02
  -7.21722007e-01  4.88173395e-01 -7.99344778e-02 -1.08425832e+00
  -3.17739606e-01  7.58968472e-01 -6.07070208e-01  1.81987488e+00
  -1.71889842e-01 -2.49826908e-01 -5.96821904e-01  8.67406487e-01
   5.27670443e-01  1.10587108e+00  5.83196044e-01  9.41613913e-01
   5.14691174e-01 -3.38499278e-01  1.54132277e-01  3.66338998e-01
   7.91521728e-01  3.81664671e-02  2.27449045e-01  1.80737710e+00
   1.06014609e-02  8.72390747e-01  2.59957314e-02  1.75216824e-01
  -2.97018141e-01  5.80798864e-01  6.54570699e-01 -9.95783508e-01
  -9.66345191e-01 -5.02242684e-01  4.30839956e-01  1.32848001e+00
  -4.91568625e-01 -4.82788801e-01  1.08798325e-01 -9.24419403e-01
  -2.24214345e-01  4.61631536e-01  5.22599816e-01 -3.82546633e-02
  -8.10596228e-01 -5.23168385e-01 -5.25579274e-01 -1.16435289e-01
   3.96734983e-01  3.33010256e-01 -3.76419723e-01  1.59066886e-01
   1.30752504e-01  3.32233310e-03  6.03185833e-01  6.08065724e-02
  -9.78782892e-01 -9.65919137e-01 -1.01028383e-02 -5.23970783e-01
```

Note: This figure shows the sample validation set.

In Figure 52, ‘0’ represents Negative, ‘1’ represents Neutral and ‘2’ represents Positive, hence it can be observed that there is a Positive sentiment for the first two reviews and negative sentiment for the last three.

Figure 52

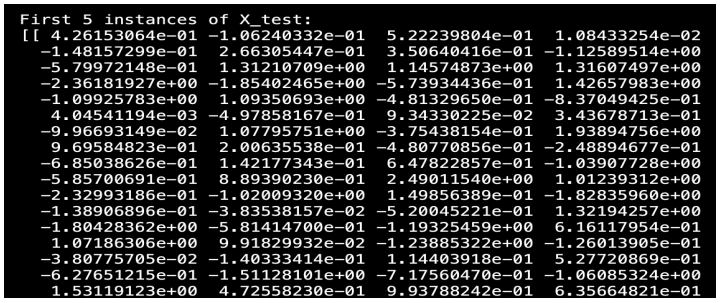
Validation Category Column

Note. This figure shows the positive, negative and neutral category of the first five instances of the validation set.

In Figure 53, the sample testing set is shown and the numbers represent vectors for the words in the sentences of the testing set after Word2vec is performed.

Figure 53

Sample of Testing Set

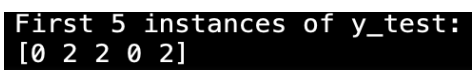


Note. The above figure shows the sample of the training set.

In Figure 54, '0' represents Negative, '1' represents Neutral and '2' represents Positive. Hence it can be seen that there is a negative sentiment for the first review, a positive sentiment for the next two, and negative sentiment again, and a positive sentiment for the last review.

Figure 54

Test Category Column



Note. This figure shows the positive, negative and neutral category of the first five instances of the testing set.

Data Statistics

The main dataset aggregates 20 different CSV files from the four domains. Different Data Cleaning and Transformation tasks are performed on the merged dataset for better and consistent data quality and model development. Once it is complete, the dataset is split into training, testing, and validation sets. Standard splitting i.e. 80-10-10 is done and the instances are randomly selected.

Figure 55 shows the initial raw dataset consists of 959491 records of data and 14 features depicting the different aspects of the Customer Reviews for the 20 different Apps. Since the data is collected using the Web scraping technique for all the applications, the same set of features is fetched for each application.

Figure 55

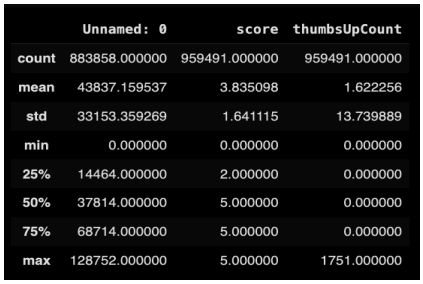
Records in Raw Dataset

Note. The figure above shows the number of records and features in the raw merged dataset for the 20 different mobile applications

The concise summary of the essential characteristics of the raw dataset is shown in Figure 56 which further helps in data analysis and data exploration.

Figure 56

Descriptive Statistics for Raw Dataset



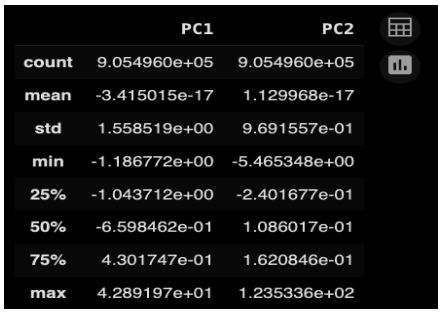
	Unnamed: 0	score	thumbsUpCount
count	883858.000000	959491.000000	959491.000000
mean	43837.159537	3.835098	1.622256
std	33153.359269	1.641115	13.739889
min	0.000000	0.000000	0.000000
25%	14464.000000	2.000000	0.000000
50%	37814.000000	5.000000	0.000000
75%	68714.000000	5.000000	0.000000
max	128752.000000	5.000000	1751.000000

Note. The above figure shows initial descriptive statistics of the raw dataset for the numerical features.

Figure 57 shows the descriptive statistics after performing the data transformation.

Figure 57

Descriptive Statistics after Data Transformation



	PC1	PC2
count	9.054960e+05	9.054960e+05
mean	-3.415015e-17	1.129968e-17
std	1.558519e+00	9.691557e-01
min	-1.186772e+00	-5.465348e+00
25%	-1.043712e+00	-2.401677e-01
50%	-6.598462e-01	1.086017e-01
75%	4.301747e-01	1.620846e-01
max	4.289197e+01	1.235336e+02

Note. The figure above shows the descriptive statistics after performing the data transformation

Table 14

Data Statistics at Different Phases

Stage	Dataset	Instances	Dimensions
Raw Data	Merged Data for 20 Applications	959491	14
Data Cleaning	Valid Dataset	928557	16

Stage	Dataset	Instances	Dimensions
Data Transformation	Normalized Data	905496	21
	Reduced Data	905496	10
	Word2vec Dataset	904204	100
Data Preparation	Training Data (Content Column)	723363	100
	Testing Data (Content Column)	90420	100
	Validation Data (Content Column)	90421	100
	Training Data (Category Column)	723363	-
	Testing Data (Category Column)	90421	-
	Validation Data (Category Column)	90421	-
Data Augmentation	SMOTE Resampled Final Training Data (Content column)	1539657	100
	SMOTE Resampled Final Training Data (Category column)	1539657	-

Multiple alterations and modifications are made to the dataset while performing the Data Cleaning and Transformation processes, such as the removal of duplicate records, and inconsistent data types, handling missing or null values, and removal of special characters or emojis. The instances of every stage are mentioned in Table 14.

The number of duplicates in the content column is 257279 which are not dropped as there can be multiple users who have given the same review for different apps. The Data Cleaning

Process resulted in 928557 records of data and 16 features in the dataset. After the Data Cleaning process is performed, Data Transformation is performed to transform the data and remove the unnecessary features from the dataset. In the Data Transformation process, using data reduction about 8 features are dropped to maintain the user's privacy and for the sake of proper Model Building. During Normalization and Feature Engineering, features like 'ProcessedContent', and 'ScoreCategory' are included. With the help of Topic Modeling, features like 'Topic' are added to properly generalize the dataset. With the help of the Word2vec technique, which is a word embedding technique used to add meaningful context has been added to the text data, where the vector size of the word vectors is set to 100. After the Word Embedding, the dataset consists of 904204 records of data and 100 columns representing the dimensions of the features. After the Feature Engineering, the dataset is split into Training, Validation, and Test subsets, where the split is done in 80:10:10, which is a standard form of splitting the dataset which is used further during Model Training and Evaluation.

With a shape of (723363, 100), X_train indicates that the training set contains 723,363 occurrences, each of which is represented by a feature vector with a length of 100. The Word2vec embeddings calculated for each instance in the dataset are probably these feature vectors. The form of y_train is (723363,), meaning that each instance in the training set includes 723,363 labels. The encoded category of each label corresponds to its corresponding instance. After performing SMOTE Analysis, the training dataset is augmented with another 816294 records resulting in a properly balanced dataset among the three categories.

Validation Set Shape (X_val, y_val): (90420, 100) (90420,) Given that X_val has the shape (90420, 100), the validation set consists of 90,420 instances, each of which has a feature

vector with length 100. The shape of y_val is (90420,), which in the validation set translates to 90,420 labels.

With a shape of (90421, 100), X_test indicates that there are 90,421 instances in the test set, each of which is once more represented by a feature vector with 100 dimensions. The shape of y_test is (90421,), which stands for the 90,421 labels in the test set.

Data Analytics Result

The dataset includes the reviews that the users posted after using the apps from the Google Play Store. Top common and everyday user requirement-based domains are selected to get to know how well the app aligns with the expectations of the users by performing the sentiment analysis on the reviews they placed. Numerous data visualizations are created to know the hidden trends and to point out the major reasons for the overall rating of the app. These visualizations provide a general sense to new and upcoming users so that they can know if the app satisfies their needs or not. Users can save a great deal of time and significantly streamline their processes by utilizing the analysis that is done.

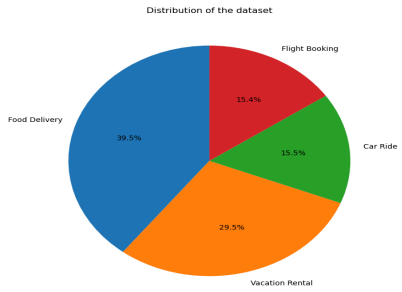
The knowledge gathered from the in-depth analysis provides a solid basis, doing away with the necessity for experimenting with different applications. This maximizes productivity and the advantages received from the accumulated knowledge and conclusions by enabling users to make well-informed decisions quickly.

The visualizations can also be very helpful to the developers as they can know how the users interact with their app. They can take the user reviews and suggestions into consideration and make modifications accordingly making the app a better fit amongst other competitors under the same domain. In Figure 58, the dataset distribution of all four domains is depicted.The

domain for food delivery has the highest data at 39.5%, followed by the domain for vacation rentals at 29.5%, the Car ride domain at 15.5% and the domain for flight booking at 15.4%.

Figure 58

Dataset Distribution

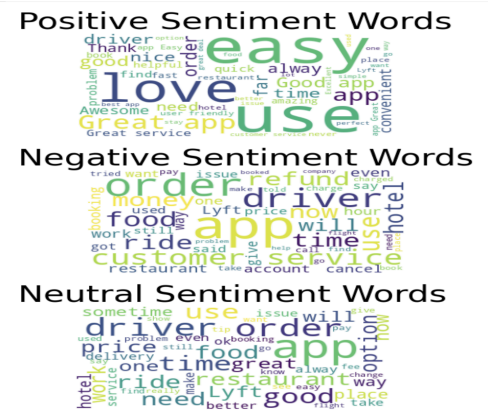


Note. The figure above shows the distribution of the data in the four selected domains.

The positive, negative and neutral review sentiment are presented in the word cloud in Figure 59. This enables quick and easy identification of the frequently used words. Love is the most used positive sentiment word. Similarly, most negative and neutral sentiment words used by the users are order and driver related respectively.

Figure 59

Word Cloud for Positive, Negative, and Neutral Review Sentiment

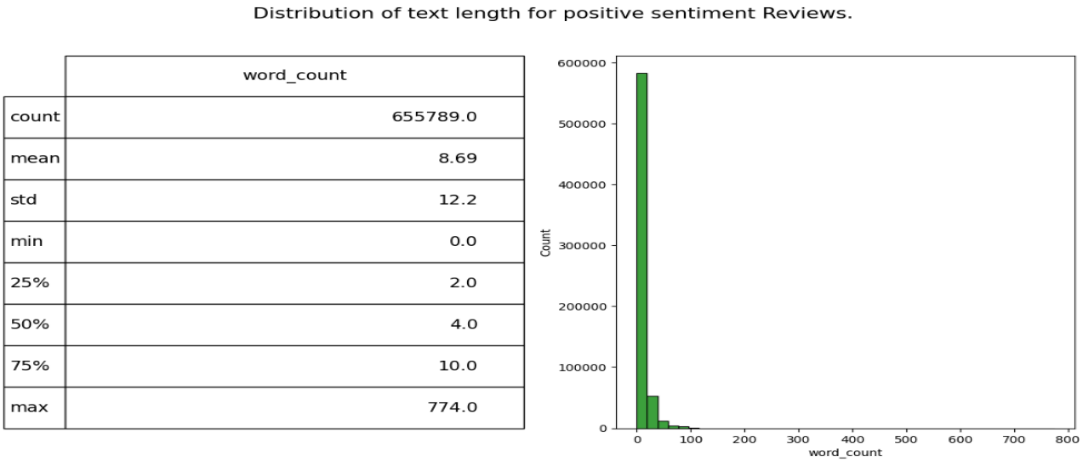


Note. The figure above shows the Word Cloud for Positive words, Negative, and Neutral words in the user reviews. Large fonts emphasize the most common words.

In Figure 60,the statistics for the positive sentiment reviews are represented. This explains that the majority of the positive sentiment reviews have a review length between 10 and 32 words. The mean text length is eight words. Approximately, 25% of the neutral reviews have a text length of 33 words. The longest negative sentiment review has a length of 774 words.

Figure 60

Statistics for the Positive Sentiment Reviews



	word_count
count	655789.0
mean	8.69
std	12.2
min	0.0
25%	2.0
50%	4.0
75%	10.0
max	774.0

Note. The bar chart in the figure shows the distribution of positive sentiment of the user reviews.

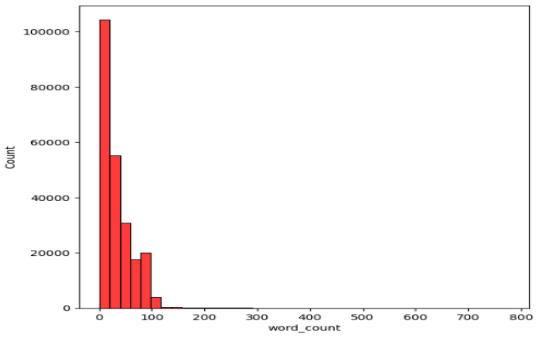
The statistics of the negative sentiment reviews are depicted in Figure 61.The figure shows the distribution of the length of negative sentiment reviews. The majority of negative sentiment reviews have a text length of between 10 and 50 words. The mean text length is 32 words per review. Approximately, 25% of the negative sentiment has a text length of 48 words. The longest negative sentiment review has a length of 777 words.

Figure 61

Statistics for Negative Sentiment Reviews

Distribution of text length for Negative sentiment Reviews.

	word_count
count	232986.0
mean	32.89
std	29.63
min	1.0
25%	10.0
50%	24.0
75%	48.0
max	777.0



Note. The above figure shows the statistical information about the native sentiment in the form of barc graph.

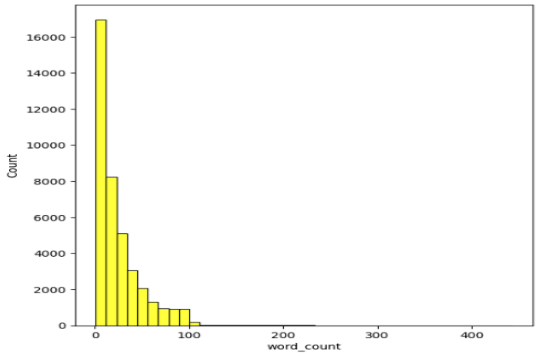
The majority of the neutral sentiment reviews have a review length of between 10 and 32 words as found in the analysis shown in Figure 62. The mean text length is 24 words. Approximately, 25% of the neutral reviews have a text length of 33 words. The longest negative sentiment review has a length of 443 words.

Figure 62

Statistics for Neutral Sentiment Reviews

Distribution of text length for Neutral sentiment Reviews.

	word_count
count	39782.0
mean	24.18
std	25.13
min	1.0
25%	6.0
50%	16.0
75%	33.0
max	443.0

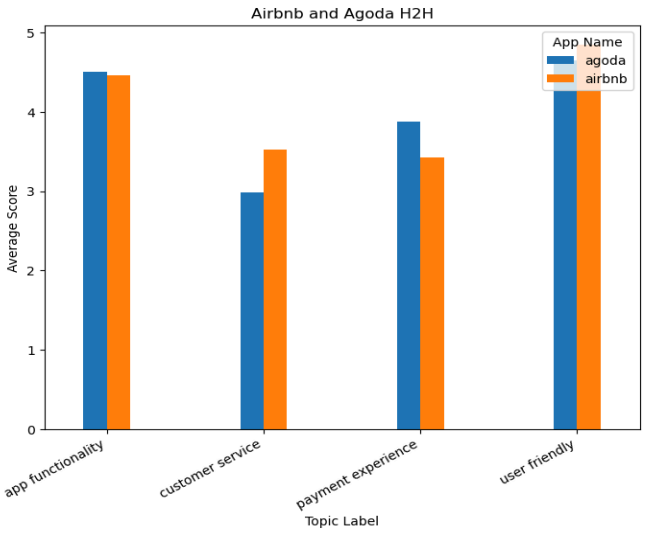


Note. The bar chart in the figure shows the distribution of neutral sentiment of the user reviews.

From Figure 63,the average score for Airbnb is more than that of Agoda for the customer service department whereas Agoda has a better average score for payment experience compared to Airbnb.

Figure 63

Comparative Analysis of Airbnb and Agoda



Note. The above figure displays the comparative analysis of Airbnb and Agoda for different areas of the app.

Model Development

Model Proposals

In the current model building, the Support Vector Machine(SVM), Bi-LSTM, BERT, and XLNet model performs the sentiment analysis of the user review data of 20 different Mobile Applications of the Travel and Lodging industry. The study aims to analyze the user's sentiments and identify the most recommended features based on their opinion. These recommendations

will be helpful to the developers in incorporating the updates based on the actual customer demands.

Support Vector Machine

Support Vector Machine is a supervised ML algorithm, which is used in the project to perform sentiment analysis of the review data of mobile applications and classify the review sentiments in Positive, Negative, and Neutral. It inspects the data and identifies patterns used for Classification and Regression (Manek et al., 2016, as cited in Roushangar & Ghasempour, 2023). The SVM model includes a kernel trick. A kernel function is used to perform this trick that performs a scalar dot product of data points. In this scenario, the kernel function replaces the dot product, which computes the dot product of a higher dimensional feature space(Smola, 1996 as cited in Roushangar & Ghasempour, 2023).

Manek et al. (2016) discussed the Customer's interactions and opinions on the Web, its importance in decision-making, and the usefulness of the information to other customers. It is also beneficial to the businesses in assessing their future in terms of Success and their sustainability. The study incorporates a feature selection method called the Gini Index with a supervised Machine Learning(ML) algorithm called SVM to classify the sentiment for a large movie review dataset. The data about the movie reviews is fetched from different movie-related websites like Bollywoodhungama, Rediff, Times of India, Rottentomatoes, and Mouthshut using a web crawler. The experiment results reveal that the Gini index method performs better in classifying the sentiment in terms of Accuracy and reduced error rate. The SVM classifier performs the classification with an accuracy of 96.96% and an error rate of 3.05%. The Naive

Bayes classifier shows an accuracy of 85.19% with an error rate of 14.81% in classifying the sentiment.

The approach of Sentimental analysis is beneficial for the Media and Network Personnel to gauge the viewer's preference for a particular film based on the available text information and its hidden sentiments data. The study incorporates the data about the film review text to conduct the sentimental analysis using the SVM algorithm. Different types of dictionaries, the Basic Sentiment dictionary, Domain Sentiment dictionary, degree adverb dictionary, and negative word dictionary, are fabricated to perform the classification. An expanded dictionary is created by merging all the constructed dictionaries. The SVM algorithm receives the combined weight of the Sentiment Weight and the User's Score as the input parameter. The sentimental classification performed using the expanded dictionary shows the best performance. It gives an accuracy of 70.4% in classifying the Positive Sentiment and an accuracy of 69.20% in classifying the Negative Sentiment. Whereas if we compare the results of a Basic dictionary, it gives an accuracy of 65.5% in predicting a Positive Sentiment, and in predicting a negative Sentiment, it has an accuracy of 64.8%(Lu & Wu, 2019).

The research performed by Dey et al. (2020) includes a sentimental analysis of Customer Reviews on Amazon Products using two different ML approaches. The aspects of product reviews such as the review content, the product quality, review timestamp, and information about historically older positive reviews are incorporated to generate the overall Product Ranking. The study uses the Naive Bayesian Classifier to determine the Customer's sentiment and the SVM Classifier for the binary categorization. The collected data is preprocessed using TF and IDF (Term Frequency, Inverse Document Frequency). The study aims to perform a comparative

analysis of the two Machine Learning Algorithms, Naive Bayesian and SVM. The SVM model gives an accuracy of 84% and 82.53% precision. The Naive Bayesian model has an accuracy of 82.875% and a precision of 83.990%. The results show that the SVM model performs better than the Naive Bayesian model. Figure 64 shows the steps to execute the SVM algorithm.

Figure 64

SVM Algorithm

Input: I: Input data
Output: V: Support vectors set
Begin
Step 1: Divide the given data set into two set of data items having different class labels assigned to them
Step 2: Add them to support vector set V
Step 3: Loop the divided n data items
Step 4: If a data item is not assigned any of the class labels then add it to set V
Step 5: Break if insufficient data items are found
Step 6: end loop
Step 7: Train using the derived SVM classifier model and test so as to validate over the unlabeled data items
End

Note. The figure above shows the sequence of steps of the algorithm to classify the data using SVM. From "Random Forest and Support Vector Machine based Hybrid Approach to Sentiment Analysis" by Al-Amrani, Y., Lazaar, M., & Kadiri, K. E. E., 2018, *Procedia Computer Science*, *127*, 511–520. https://doi.org/10.1016/j.procs.2018.01.150. Copyright 2008 by Creative Commons CC-BY-NC-ND license

The input parameters for the SVM model as explained by Roushangar & Ghasempour (2023), are the Regularization parameter (C Parameter), Kernel function, degree parameter, and gamma parameter. The Regularization parameter improves the SVM classifier and excludes wrongly classifying the data. A smaller value of parameter C results in a large margin of the decision boundary, and a large value of parameter C implies a smaller margin for the decision

boundary. The kernel function performs the feature transformation, where different versions of the kernel function can decide the kernel value, such as Linear, polynomial, radial basis function(RBF), etc. It enables the SVM to execute in a higher dimensional space without requiring any explicit calculation of Feature space vectors. The degree parameter is an input parameter for the polynomial functions and specifies the degree of the polynomial kernel.

Similarly, Roushangar & Ghasempour (2023) explained about the gamma parameter, which is a very important parameter that decides which data point is selected for the calculation of the decision boundary. For a low gamma value, the data points, which are far distanced are considered for the calculations, and in the case of a high gamma value, the data points close to the decision boundary are considered for the calculation. It affects the dimensions of the decision boundary and depicts the effects which can be seen in the model evaluation. The output of the SVM is the predicted target Labels or classes for the given input parameters.

Linear Classifiers. According to Roushangar & Ghasempour (2023), the SVM technique is typically a linear classifier that is segregated in two distinct classes. In a ML problem that is increasing in knowledge, the data has components with one of the two fixed class labels corresponding to the two classes. In Figure 65, assuming labels with value +1 are representing the positive patterns and value -1 is representing the negative patterns. The variable x is a set of vectors with components x_i. The notation x_i is the i^{th} vector in the dataset $\{(x_i, y_i)\}_{i=1}^{n}$, where y_i is the target label in the dataset. The set X represents the input space of the patterns. Based on a linear discriminant function, Equation (1) represents a linear classifier, where $w^T x = \Sigma_i w_i x_i$ is a scalar product between two vectors, w is the weight vector, and b is the bias.

$$f(x) = w^T x + b \quad (1)$$

Roushangar & Ghasempour, (2023) suggest that, the line separating the positive and negative class regions is the Decision Boundary defined as a Hyperplane. The decision boundary in Figure 65 is the linear SVM since it is linear in the input patterns. The Linear SVM is a classifier with a linear decision boundary (Roushangar et al., 2020, as cited in Roushangar & Ghasempour, 2023).

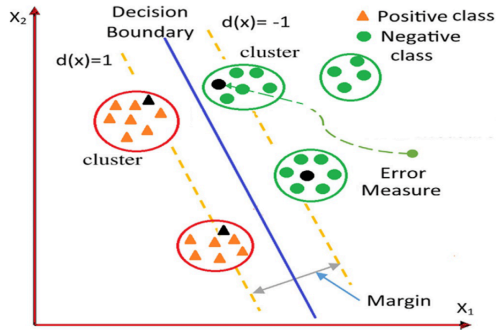
Non-Linear Classifier. According to Roushangar & Ghasempour, (2023),it is an extended version of the Linear SVM, where the decision boundary depends on present data in a non-linear manner. A simple way of creating a non-linear classifier from a linear classifier is by mapping the "data from the input space X to a Feature Space F". The mapping is the task of the Kernel function. Figure 66 depicts the creation of the non-linear classifier, where the non-linear classifier is defined using φ: X→F, where in the space F, Equation (2) represents a non-linear classifier.

$$f(X) = w^T \varphi(x) + b \quad (2)$$

As suggested by Roushangar & Ghasempour,(2023), the decision boundary in a non-linear classifier is dependent on the input data in a non-linear way, and it uses a nonlinear Kernel function such as RBF, polynomial, which allows the SVM model to perform more complex computations on the decision boundary.

Figure 65

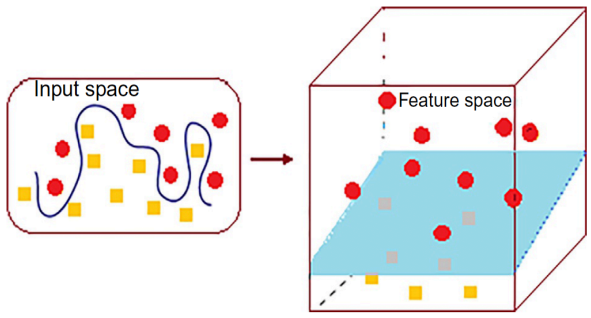
Linear SVM Classifier and its Decision Boundary



Note. The figure illustrated above shows the linear Classification problem depicting the Positive and negative classes and their decision boundary. From "Supporting vector machines" by Roushangar, K., & Ghasempour, R., 2023, In *Elsevier eBooks* (pp. 411–422). https://doi.org/10.1016/b978-0-12-821285-1.00009-9. Copyright 2023 by Elsevier Inc.

Figure 66

Non-Linear SVM Classifier



Note. Figure 66 depicts the Feature Transformation of the SVM from input space to feature space and the non-linear SVM model. From "Supporting vector machines" by Roushangar, K., & Ghasempour, R., 2023, In *Elsevier eBooks* (pp. 411–422). https://doi.org/10.1016/b978-0-12-821285-1.00009-9. Copyright 2023 by Elsevier Inc.

Model Optimization. The initial model training and testing are performed with a Linear Classifier, where parameters given to the model are the kernel function, Regularization parameter, and degree. The kernel function has a value of Linear, the Regularization parameter has a value of 1.0, and the degree has a value of three. The model performance is optimized with the help of Grid Search which identifies the parameters that provide the most optimal performance. A non-linear classifier known as Kernelized(Non-Linear) SVM which is a complex version of the traditional SVM is identified to optimize the model performance. As suggested by Roushangar & Ghasempour, (2023), it allows the SVM model to handle non-linear data without exclusively performing the transformation of its feature vectors by tweaking the kernel function. The kernel function used is RBF which allows the SVM model to run in a higher dimension space. It helps in enhancing the model performance significantly.

CNN Bi-LSTM

XLnet, Multifit, and CamemBERT were used by Habbat et al. (2021) to represent text in the first phase, which was then followed by classifiers CNN, Long Short-Term Memory (LSTM), Bidirectional Long Short-Term Memory (BiLSTM), and Gated Recurrent Unit (GRU). They also present the fusion of various RNN models with CNN. GRU captures the short-term dependencies in sequential data and handles sequences of varying lengths. After analyzing, Habbat et al. (2021) identified that XLNet works best in word embedding achieving 96.5% accuracy with the Fellow of the American College of Radiology(FACR) dataset. Additionally, the CNN model uses the richer original input representations to detect local patterns while the GRU successfully learns time series-based sequential data from XLNet. This combined approach of XLNet and GRU results in the highest accuracy when compared with other neural networks.

A deep learning model was employed by Al-Harbi (2021) where he combined two models CNN and Bi-LSTM for feature extraction and the Support Vector Machine (SVM) for Arabic sentiment classification. Large-Scale Arabic Book Reviews(LABR), (Office Management and Computer Application)OMCCA, and other publicly available dataset was used counting up to a total of 95000 reviews. Pre-trained word embedding that was trained on large Arabic text corpora was used for the generation of the word vectors. Using this modified architecture having CNN and Bi-LSTM layers for sentiment classification, the model outperformed other previously conducted research on the same dataset and recorded the highest precision of 91% and recall of 91.6%.

Convolutional Neural Networks is another type of neural network used in machine learning that can be used for text, audio, and image classification very well. CNN is used to extract high-level features with the help of its multiple convolutional and pooling layers (Hassan & Mahmood, 2018). The output of every layer is connected with the input of the next layer. Multiple layers are required to handle long-term dependencies while classifying the data into given categories. To avoid adding multiple layers and for the sake of lesser complexity BiLSTM layers are added so that bidirectional dependencies are taken care of. Bi-LSTM is a Recurrent Neural Network (RNN) specifically used to determine the sentiment or emotion over a long sequential data or time series. The hidden layers of the network processes input sequences in forward as well as backward direction getting a general context of the content of the review. Mousa et al. (2017). Inputs are first fed to the network layers of the Bi-LSTM model. These inputs are the words embedded vectors for each review. The input sequence is taken by the model's forward layers, which process it from beginning to end before sending it to the following

hidden layer. Analogously, the model's reverse layer operates in the opposite direction as the forward layer, taking the input sequences and processing them from end to beginning. After going through these layers, the input sequence passes through the activation layer of the model where it is categorized into different classes for multi-level classification. The final output is received from the output layer where the sentiment of the review is classified as ‘Positive’, ‘Negative’, or ‘Neutral’. Bi-LSTM works better than standard LSTMs where input is only processed in a single direction. Increasing the number of layers, Bi-LSTM handles the processing more efficiently and provides more accurate results keeping the general context in its memory. On the other hand, the computational load also gets doubled because of the increased number of layers as compared to the LSTM.

BERT

Sousa et al. (2019) have used a pre-trained BERT model to identify sentiments - as BERT can identify polysemy in sentences- from news articles to make better decisions in the stock market. Dataset Dow Jones Industrial Index data, which includes Microsoft, Intel, Boeing, and other stock information as well as articles from Business Insider, Consumer News and Business Channel(CNBC), Forbes, Investopedia, NewYork, Times, Washington Post, Other includes 582 articles. Here, the model is pre-processed the same as a study conducted by Devlin et al. (2018). They have tested that the BERT model performance is 8.6% more accurate than the Convolutional Neural Network and Word Embedding models by achieving an F1 score of 72.50%. In future work, Individual firms' unique news may be extracted, and processing and evaluation of information on the value of the respective companies' shares can be performed.

Munikar et al. (2019) fine-tuned BERT utilizing transfer learning and tested its performance on the Stanford Sentiment Treebank (SST) datasets in their study. On the SST-5 dataset, which comprises 11,855 one-sentence movie reviews taken from Rotten Tomatoes, the researchers used Masked Language Modeling (MSK) and Next Sentence Prediction (NSP) as a transfer learning. Three people labored to meticulously label the final unprocessed dataset, which consisted of around 215,154 statements of varying lengths. The labels were as follows: '0' (very negative), '1' (negative), '2' (neutral), '3' (positive), and '4' (extremely positive). The research added unique tokens, tokenized the data, and canonicalized it. After preprocessing the data, they added BERT embeddings and a 0.1 dropout to stop overfitting. In the final layer, they predicted one of the five classes using a softmax function. On the SST-2 dataset, the study obtained accuracy results of 94.00% and 94.70% for BERT-Base and BERT-Large, respectively. The accuracy values obtained by the authors for BERT-Base and BERT-Large on the SST-5 dataset were 83.90% and 84.20%, respectively.

According to Devlin et al. (2018), the architecture of the BERT model is based on the Transformer's encoder architecture, as it is a multi-layer bidirectional Transformer model. That is why this study first explained Transformer architecture in detail to understand the work of the BERT model. Myagmar et al. (2019) explain in their study how BERT's architecture is based on the encoder portion of the original Transformer model, with twelve identical encoder layers. Each layer consists of a fully connected feed-forward network and two sub-layers with twelve parallel attention heads, allowing the model to focus on different parts of the input sequence simultaneously. In the study conducted by Munikar et al. (2019), unlike traditional models that process sequentially or recurrently, the attention architecture parallels the processing of the entire

input sequence. This allows all input tokens to be simultaneously processed, rather than sequentially or recurrently. The training process of BERT is explained by Devlin et al. (2018) in phases of two unsupervised tasks: the Masked Language Model (MLM) task, in which a portion of the input tokens are randomly masked, and the model predicts these masked tokens; and the Next Sentence Prediction (NSP) task, in which the model learns to determine if a statement B genuinely follows another sentence A, with B being the subsequent sentence 50% of the time and a random sentence otherwise. To be more precise, Myagmar et al. (2019) explained in their study how BERT creates a distorted version of xˆ by randomly assigning 15% of the tokens in x to the special symbol [MASK] given a text sequence x = [x1,..., xT]. Equation (3) shows the reconstructing $\bar{x}$ from $\hat{x}$ is the training goal if the masked tokens are denoted as $\bar{x}$,where H_θ is a Transformer that converts a length-T text sequence x into a series of hidden vectors, $H_\theta(x) = [H_\theta(x)1, H_\theta(x)2,..., H_\theta(x)T]$, and mt = 1 indicates that the token x_t is masked.

$$max log\, p_\theta(\bar{x} \mid \hat{x}) \approx \sum_{t=1}^{T} m_t log\, p_\theta(x_t \mid \hat{x}) = \sum_{t=1}^{T} m_t log \frac{exp(H_\theta(\hat{x})_t^T e(x'))}{\Sigma_{x'} exp(H_\theta(\hat{x})_t^T e(x'))} \quad (3)$$

The structure of BERT-Base model with tokenization steps explained by Devlin et al. (2018) in their study explain that BERT demands the input token sequence to follow a certain pattern, which involves the inclusion of a [CLS] token (classification token) at the starting point of each sequence and a [SEP] token (separation token) after each sentence. For categorizing the whole sequence, the output embedding associated with the [CLS] token serves as the sequence embedding.

XLNet

XLNet is a pre-trained model based on Transformer-XL architecture. The model introduces a novel permutation language modeling method, which considers all possible permutations of the input sequence during training. XLNet understands complex Linguistic relationships by capturing bidirectional context. Built on the transformer based architecture, XLNet demonstrates state-of-the-art benchmarks in Natural Language Understanding.

The research conducted by Adoma(2020) focussed on distinguishing emotions into anger, disgust, sadness, fear, joy, shame, and guilt. The International Survey on Emotion Antecedents and Reactions (ISEAR) dataset is publicly available balanced data with 7666 sentences classified on the seven distinct emotions. This study compares the pre-trained transformer models such as the Bidirectional Encoder Representations from Transformers(BERT), the Transformer-XL network(XLNet), the Robustly optimized BERT approach (RoBERTa), and DistilBERT in identifying emotions in texts. RoBERTa and XLNet achieved the highest accuracy of 0.7431 and 0.7299 respectively. These accuracies were achieved after fine tuning the RoBERTa and XLNet with 768 hidden layers and 12 attention heads.

Another research conducted by Chen and Nguyen(2020) proposes a new solution for enhancing the citation function and citation solution by fine-tuning the BERT, XLNet, and the Universal Language model fine-tuning (ULMFiT) pre-trained models. Citation functions and citation solutions help significantly understand why an author cites another author's work and their opinion regarding the paper cited. The experiment starts with context embedding with Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI) SmartData corpus dataset to further fine-tune the citation context and finally train the sentiment classifier of the citation

function. The XLNet-base model outperformed the other models by achieving 87.2% accuracy for citation function and 91.72% for citation sentiment identification.

To analyze the public's opinion towards the two popular vaccines, Covidshield, and Covaxin, administered in India, Bansal et al. (2022) performed a sentimental analysis of the unlabeled COVID-19 vaccine tweets dataset. The US Airlines Twitter dataset, one of the benchmark datasets for tweet-based sentiment analysis, was used initially to train the XLNet model to categorize the tweets into positive, negative, and neutral opinions. More specifically, the unlabeled tweets about the two COVID-19 vaccinations were examined in the second phase of the study to see if there has been a shift in the public's perception of vaccines in particular. By classifying 85% of the positive tweets and 92% of the negative tweets, the XLNet model outperformed the unsupervised VADER and TextBlob models as well as the supervised learning Bi-LSTM and BERT models for sentiment analysis. The study revealed that positive opinions toward vaccines increased during the second wave in India.

Based on the efficacy and remarkable benchmark in Natural Language Processing tasks, the XLNet model is chosen for the implementation of sentiment analysis of mobile application reviews. The Transformer-XL architecture is the backbone of the XLNet model as indicated by Yang et. al(2019). In each layer of the XLNet transformed encoder, a two-stream self-attention mechanism is applied. Relative positional encoding helps understand the sequence of words in the input data. Equation (4) and (5) shows the positional encoding for inputs where pos is the position of tokens, i is the dimension of individual token embedding and d is the embedding dimension of words to grasp the order. Feed-forward networks understand the relationship between words and generate hidden states for each word to capture the context as well as the

meaning of the words. These feed-forward networks are connected in each layer of the encoder and decoder.

$$PE_{(pos,2i)} = sin(\frac{pos}{10{,}000^{2i/d_{model}}}) \quad (4)$$

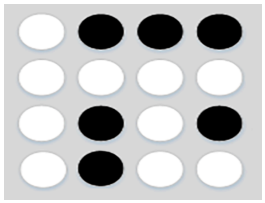
$$PE_{(pos,2i+1)} = cos(\frac{pos}{10{,}000^{2i/d_{model}}}) \quad (5)$$

In the decoder, the attention mask mechanism is used to predict the next word without knowing the future words. XLNet gathers long-range dependencies using Permutation Language Modeling(PLM) which predicts the tokens with all possible permutations with respect to the surrounding words. The word with the highest probability is predicted by the decoder as the next word.

Attention Mask. The token embeddings with the positional information are passed as inputs for probability computation in PLM. The token to be predicted is masked in the transformer to ensure it does not impact the prediction process(Yang et al. 2019). The original sequence arrangement is (1,2,3,4) and the sampling sequence order is (2,4,3,1). When computing the prediction of the first element which is the token two in this instance the model will not have the context of the previous tokens. So the mask for the first element, token two would be [0,0,0,0]. Now proceeding to the second element, token four the mask would be [0,1,0,0] since the model predicted the token two from the previous iteration. Tokens three and four for element three and element four, respectively, will be [0,1,0,1] and [0,1,1,1], respectively. The model considers the pertinent portions of the sequence and disregards the irrelevant portions at each forecast. The order of the (2,4,3,1) Mask Matrix is displayed in Figure 67.

Figure 67

Illustration of Mask Matrix of Sequence (2,4,3,1)



Note. The above figure represents the mask matrix of the sample sequence (2,4,3,1). Here the black circle represents the available data and the white circle represents the masked data. From*,* *"*Text Emotion Recognition based on XLNET-BIGRU-ATT" by Tian, H., Zhang, Z., Ren, M., Dong, C., Jia, X., & Zhuang, Q. (2023), *Electronics*, *12*(12), 2704.

Permutation Language Modeling. Yang et al.(2019) proposed a PLM that captures bidirectional context by considering the probabilities of each word in all possible orders to predict the original order. Thus the model can gather information from words in a sentence in both forward and backward directions. The research proposed PLM objective is represented in Equation (6) where for a sequence x of length T, there are T! different orders to perform valid order. Zt is the 't' element of the permutations. By calculating the sum of the log probabilities the model can predict the next element based on the previous element as it trained on the probabilities of each token in all possible orders of the input context.

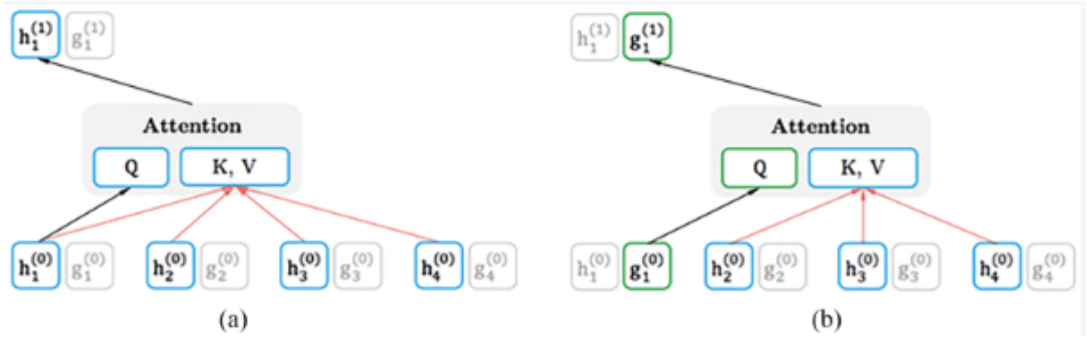
$$max_{\theta}\, E_{z\sim \mathbf{z}_T}[\Sigma_{t=1}^{T} log\, p_{\theta}(x_{z_t} \mid x_{z<t}] \quad (6)$$

Two-Stream Self-Attention for Target-Aware Representations. With XLNet's unique attention mechanism, the model can understand both right-to-left and left-to-right contexts of longer phrases (Yang et al. 2019). This is achieved by introducing two streams, Content stream and Query Stream. The standard content attention shown in Figure 68 shows the attention

mechanism without the positional encoding details and the query stream on the right has details of the content as well the positional information.

Figure 68

Schematic Representation of Content and Query Stream



Note. The figure above shows (a) Content Stream Attention which is the standard stream attention method, and (B) Query Stream Attention, which does not access the information in the predicted word.From, "Text Emotion Recognition based on XLNET-BIGRU-ATT" by Tian, H., Zhang, Z., Ren, M., Dong, C., Jia, X., & Zhuang, Q. (2023), *Electronics*, *12*(12), 2704.

The content stream receives the Query (Q) with the context of the current token as in Figure 69, Key (K), and Value (V) representing all the other tokens in the sequence. Similarly, the Query Stream receives Query(Q) with positional information of the current token computed by positional embeddings and relative encoding. Value(V) denotes the context of other tokens and Key(K) represents the positional information of all other tokens. The attention weights are computed using,

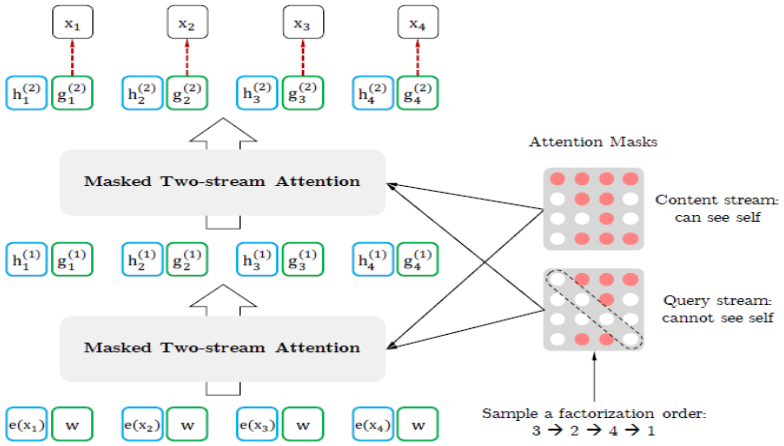
$$\text{Attention}(Q\_i, K, V) = \text{softmax}(Q\_i * K^T / \text{sqrt}(d)) * V \quad (7)$$

In the above formula, each token where Q_i is the query vector of i-token, K is the key vector of all words, V represents the matrix of value vectors of all the words, and d denotes the

dimension of the key and query vector. Here the softmax function normalizes the attention weights to add to 1. By using both streams, XLNet can differentiate between the semantic meaning of the sentence and the respective positioning of the tokens. The attention mechanism combines the query stream and content stream.

Figure 69

Overview of PLM with Two-Stream Attention Architecture



Note. The figure above shows permutation language modeling with two-stream attention for the sequence (x1, x2, x3, x4). The bottom-to-top masked two-stream attention is calculated based on content and query stream. In the attention masks, the content stream of the token itself participates in the prediction; on the other hand, the query stream does not participate in its prediction. From "Xlnet: Generalized autoregressive pre-training for language understanding" by Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., & Le, Q. V. (2019). *Advances in neural information processing systems,* 32.5753–5763.

Table 15 shows the steps involved in the sentiment analysis of the XLNet algorithm.

Table 15

Algorithm for Sentiment Analysis with XLNet

Algorithm Sentimental Analysis with XLNet
Input: Input Text Data
Output: Text Vector Representation of Sentiment Prediction
Step 1: For each text sequence in the dataset:
Step 1.1: Tokenize the text sequence using XLNet Tokenizer with an attention mask.
Step 1.2: Add special tokens ([CLS] - the start of the sequence, [SEP] - separator between two sentences.
Step 1.3: Convert tokens to vector using XLNet Embeddings
Step 2: For each input
Step 2.1: Computer positional encoding for the input sequence using equation (4) and (5)
Step 3: Two-stream Self-attention mechanism
Step 3.1: Calculate the self-attention score of each token using Equation (7)
Step 3.2: Summarize multi-attention as one self-attention before passing through the output layer.
Step 4: Predict the target word sentiment based on the permutation Equation (6) where XLNet computes all possible positions.
Step 5: Use the output encoded token to determine the sentiment class.
Step 6: Assign the sentiment label with the highest probability as the predicted sentiment.
Step 7: Obtain the predicted class probability for the Text classification

Note. The above table lists the steps involved in the sentiment analysis with the XLNet.

Model Optimization. The XLNet's performance is boosted and optimized by the learning rate, sequence length, and optimizer. Learning rate plays an important role in balancing the accuracy and generalization in pre-trained models. It determines the rate at which the model learns from the training data. By sequence length, XLNet is capable of handling sequences of varied lengths. This impacts the training time and memory requirements. Optimizer helps to minimize the loss function and improve model performance for the given task.

Model Support

Environment, Platform, Tools

Table 16 and Table 17 shows the final configuration for the Hardware Requirement ,Software requirements and Tools to fulfill the research end goal.

Table 16

Hardware and Software RequirementRequirements

Hardware & Software	Memory	Usage
12 Core CPU, 19 Core GPU, 16 Core Neural Engine	Ram: 16 GB, SSD: 1 TB	Data Handling, Training, and Testing ML Models
Mac OS Sonoma	-	Operating System

Table 17

Tools Used

	Library	Function	Usage
NLTK	nltk.corpus	stopwords	To remove stopwords.

	Library	Function	Usage
	nltk.stem	WordNetLemmatizer	To perform lemmatization
	nltk.tokenize	word_tokenize	To perform tokenization
	nltk.stem.porter	PorterStemmer	To perform Stemming
Tensorflow	Tensorflow.keras. optimizers	Stochastic Gradient Descent(SGD)	To optimize algorithm
		AdamW	To optimize algorithm
	tensor.keras.model	load_model	To load saved model
Keras	Keras. preprocessing. text	tokenizer	To tokenize sentences
	Keras. preprocessing. sequences	pad_sequences	To pad sentences for consistent input
		Dense	Fully connected layer
		Dropout	To handle overfitting
		lstm	Forward feeding DL model

	Library	Function	Usage
	keras.layers	Embedding	To convert text to numeric form
		Bidirectional	Forward as well as backward feeding DL model
Keras	keras.callbacks	LearningRateScheduler	This helps model to change learning rate according to learning process while training
		EarlyStopping	Helps preventing overfitting
torch	torch.utils.data	DataLoader	To load dataframe data
		random_split	To split data randomly
		TensorDataset	To create tensor dataset after using dataloader
		SequentialSampler	To sample elements in dataset sequentially
	torch.cuda.amp	GradScaler	To perform gradient scaling conveniently

	Library	Function	Usage
torch	torch.cuda.amp	autocast	Allow script to run in mixed precision
	torch.nn	CrossEntropyLoss	To calculate loss while training and validation
transformers	BertTokenizer	from_pretrained	Special tokenizer for BERT
	BertForSequenceClassification	from_pretrained	Is a sequence classification model to train BERT model
	XLNetTokenizer	from_pretrained	XLNet tokenizer for content text
	XLNetForSequenceClassification	from_pretrained	Is a sequence classification model to train XLNet model
Skit-kit Learn	Sklearn.feature_extraction.text	Countvectorizer	To perform Vectorization
	Sklearn.model_selection	train_test_split	To split the data into Train, Test, and Validation set

	Library	Function	Usage
	Sklearn. preprocessing	LabelEncoder	To encode target variable
Skit-kit Learn	sklearn.svm	Support Vector Classification(SVC)	Implementing and optimizing the SVM model
Skit-kit Learn	sklearn.metrics	accuracy_score, precision_score, recall_score, f1_score, confusion_matrix	To evaluate the model with the help of these metrics
	sklearn.utils	Compute_class_ weight	To compute class weight
		SequentialSampler	To sample elements in dataset sequentially
Gensim	gensim.models	Word2Vec	To create vectors of descriptive features and target feature
Matplotlib, seaborn	pyplot		To plot performance metrics

	Library	Function	Usage
imblearn	imblearn. over_sampling	SMOTE	To remove the bias by augmenting the review sentiments
Pandas	Dataframe	Shape, head, drop, read_csv	For feature selection, reading, and manipulating the data frame with changes in modeling
Abstract Classes	collections.abc	Iterable	To check if the tokens are iterable but not a string
Numpy		Mean, Zeros, Unique, Arange	To perform the calculations in count vectorization, to arrange the performance metrics

Note. The above table lists the libraries used in the overall project.

Model Architecture, and Data Flow

Support Vector Machine. The modeling process starts with the data extraction required for the model development. The current study involves the use of Review data collected from 20 mobile applications on Google Play Store. The raw data is cleaned by removing missing, null, or duplicate values, followed by data transformation. Data Transformation involves the preprocessing of review content. The final preprocessed data includes a list of word vectors that

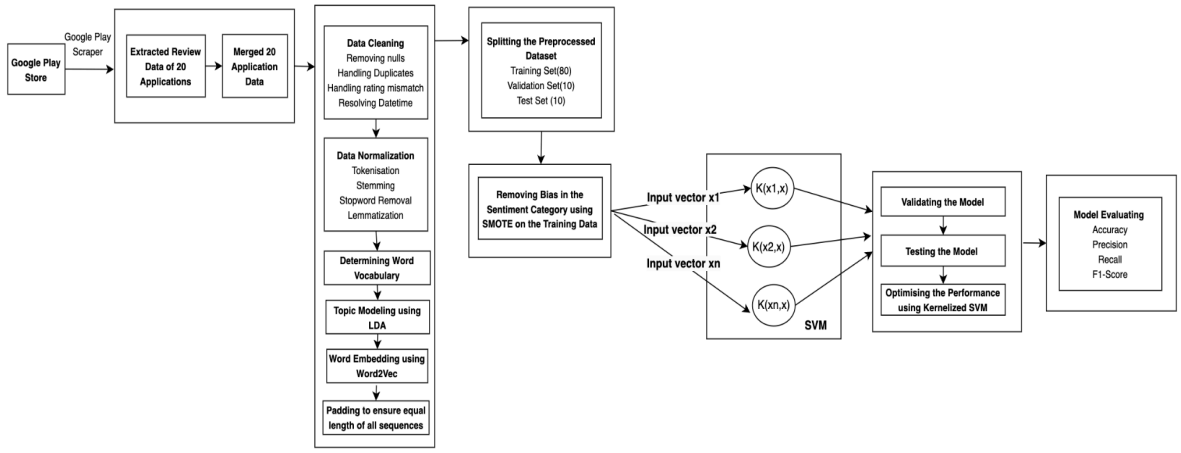
are allocated the same length via Padding. The data is split into Training, Test, and Validation Sets. SMOTE is performed on the training data to balance the dataset and remove the bias. Now the data is ready for the Model development.

The next step is to train the Model. The Model selected is the Support Vector Machine Model, which is a supervised Machine Learning Algorithm used in Classification and Regression problems (Manek et al., 2016, as cited in Roushangar & Ghasempour, 2023). The initial model training is performed using the Linear SVM model. The SVM model is created by setting the input parameters like Kernel function, Regularization Parameter(C), and degree. It is trained on augmented training data. The model is then validated and tested with the help of validation and Test data. The model performance is optimized with the help of Kernelized SVM which is an advanced version of the Linear SVM. It enhances the model performance using an RBF kernel function, which allows linear classifiers to become more complex and handle non-linear data. The Kernelized SVM model is again trained, validated, and tested on the Train, Validation, and Test set. The Kernelized SVM significantly improves the model performance.

The model evaluation is done based on the four performance metrics, Accuracy, F1-Score, Recall, and Precision. Each metric handles key aspects of model performance. The four metrics are employed to incorporate all possible scenarios in sentiment analysis. It also helps in having a comprehensive understanding of the performance. Figure 70 describes the data flow with the different components used in the Model Development. The sample code to preprocess and implement the SVM model is added in Appendix C.

Figure 70

Data Flow



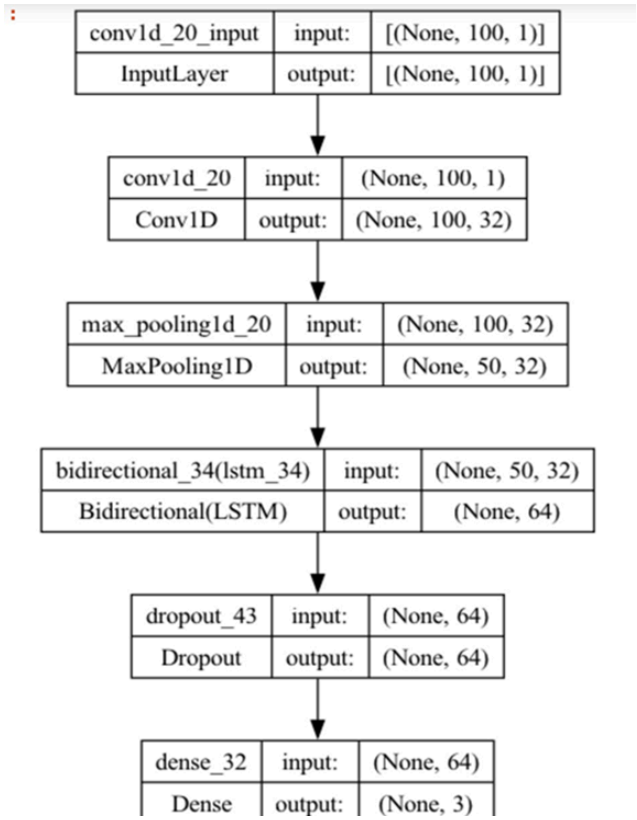
Note. The figure above shows the data flow for the model development. The SVM architecture in the figure above is adapted from "Hybrid Techniques to predict solar radiation using support vector machine and search optimization Algorithms: A review.", by Álvarez-Alvarado, J. M., Ríos-Moreno, J. G., Obregón-Biosca, S. A., Ronquillo-Lomelí, G., Ventura-Ramos, E., & Trejo-Perea, M., 2021, *Applied Sciences*, *11*(3), 1044. https://doi.org/10.3390/app11031044, Copyright 2021 by Creative Commons license.

CNN-BiLSTM. The working structure of the CNN-BiLSTM model is displayed in Figure 71.

Input Layer. The first layer in the table is the input layer where the data after all the cleaning and preprocessing is passed as a training set to the model. Here the batch size is set to none which means any number can be allotted to the batch size, the sequence length is set to 100, and one states the output feature. Every single input represents word-embedded vectors for each of the review content in the training data.

Figure 71

Neural Network Layers of Model



Note. The overall architecture of the model with all layers of the neural network.

1D Convolutional Layer. This layer applies 32 filters to input data and applies convolutional operation filtering the input signal to extract features. It is done to identify hidden patterns. The output is then passed onto the next layer which takes those 32 features as an input.

Max Pooling 1D Layer. Max pooling is performed where the function mainly focuses on reducing the dimensionality curse. Here it takes the sequence length of 100 and reduces it to 50 helping in reducing the computational load on the model. It captures the most important pieces of information.

Bidirectional LSTM Layer. This is a Recurrent Neural Network that takes the input and processes it in forward as well as backward direction. It has two separate recurrent layers that take the consideration of the past content and the future content. Due to that, the number of features increased to 64 from 32.

Dropout Layer. Dropout function is applied on this layer to prevent the model from getting overfitted. For every training iteration, 40% of the input data was automatically set to '0'.

Dense Layer. This is the output layer of the model with a shape of (none, '3'). Softmax Activation function is applied here to get the classifications of the inputs. Over here '3' indicates that the input, which is the user review is classified into three different categories i.e. 'Positive', 'Negative', and 'Neutral'.

Training data must be supplied with suitable parameters for the model to function properly. Following are some of the hyperparameters that were passed for model building.

Optimizer. To improve the effectiveness of the model, Stochastic Gradient Descent (SGD) was used. SGD adjusts the weights of the model during training to minimize the error. Momentum was also passed along with SGD as a booster to accelerate the training process and make sure that the function reached global minima and did not get stuck at local minima.

Learning Rate Scheduler. The Learning Rate is a hyperparameter that is passed while training the model. It is used to update the weights of the neural network after every iteration. The value of the learning rate lies between zero to one. Initially, the learning rate was set to 0.1. A scheduler was set through which the learning rate was reduced by 10% after every five epochs. The main reason to set a learning rate scheduler was to make sure that the model does not get stuck in the local minima and after a certain iteration successfully reaches the global minima.

With the help of this, the model was able to converge faster and had a better impact on its overall performance.

Early Stoppage. Early stopping is a regularization technique that is used to find an optimal point where the model has reached its peak performance or starts to downgrade from that point. It is done to prevent overfitting and to avoid unnecessary training on the model. For this model, a total of 10 epochs were given for the model to be trained. As Early stopping was implemented, the model stopped training at epoch 9 where it reached its max performance.The sample code to preprocess and implement the CNN Bi-LSTM model is added in Appendix C.

BERT. There are two types of BERT models one is BERT-Base which is trained on fewer parameters and the second is BERT-large trained on double then the BERT-Base parameter was trained on. Whereas BERT-Base-Uncased is trained on the lowercase of English words and that is why this model is not case sensitive. The model treats all sentences as lowercase.

BERT-Base-Uncased Preprocessing. According to Devlin et al. (2018) preprocessing text data for BERT begins with loading and using the BERT tokenizer, which was designed specifically for the BERT model. This tokenizer breaks up text into tokens, such as words, subwords, or symbols, and then sets the pieces back together in a way that makes sense for the BERT model. To guarantee that pre-trained models are applied effectively, "BertTokenizer" is used to generate tokens for the written content in a manner that is consistent with the original training of the BERT model. Because of its consistency, the model can analyze and comprehend the input text with accuracy. The CSV file containing the dataset is loaded into the environment. A random sample of 100,000 rows is chosen to guarantee adequate data diversity and efficient management of computational resources. The primary goal is to get the dataset ready for

training. To ensure consistency in processing, the textual content in the 'content' column is transformed into a string format. A LabelEncoder is also used to convert the categorical labels into a numerical format, which is necessary for the model to process the labels in supervised learning tasks like classification. To prepare the dataset for model training, this step is essential.

Figure 72 illustrates the preprocessing steps, a sample text is selected from the dataset and subjected to the following transformations:

1. Tokenization:The text is split into individual tokens, such as words or punctuation marks.
2. Token ID conversion: The tokens are converted into their corresponding numerical representations, which the BERT model can understand.
3. Special token addition: Tokens such as [CLS] and [SEP] are added to the text to provide context and structure.
4. Attention mask creation: An attention mask is generated to help the model focus on relevant parts of the text and ignore padding (non-informative tokens used to equalize the length of text inputs).
5. Final encoding: The encode_plus method integrates all these aspects into a coherent input format for BERT. This encoding method takes into account the tokenization, special token addition, truncation, padding, and maximum length (128) to create a complete input representation for the model.

The preprocessing steps are consistently applied across the training, validation, and test sets to ensure consistency and accuracy in the data format for the BERT model. This comprehensive preprocessing is crucial for effective model training and evaluation, as it aligns

the data format with the inherent design and requirements of the BERT architecture. By uniformly applying these steps, the model can process the data efficiently and accurately, resulting in improved performance and reliability.

Topic Modeling. Using Latent Dirichlet Allocation (LDA), tokenized text is first preprocessed into a vectorization-ready format and then utilized for sophisticated topic modeling in a text corpus of data. The CountVectorizer effectively records word frequencies and eliminates frequently occurring stop-words from the cleaned text before converting it into a document-term matrix (DTM). The LDA model can statistically identify latent topics after it has been learned on this matrix, which teaches it the distribution of words across topics and their relative prevalence in each document. After a comprehensive analysis of the model's output, the terms that are most relevant to each topic are determined, producing unique topical insights. To connect the study with certain areas of interest, such as "Customer Service", "Payment Experience", “App Functionality” and "User-Friendly" papers are also categorized according to their prominent subjects.

Preparing Processed Data For Training, Validation, and Testing. Tensor datasets are created using preprocessed data and labels to start the training process for a BERT model. To enable effective computation and batch processing, the data format must be aligned with PyTorch's requirements, which calls for this conversion. By using GPU acceleration, the training setup is also optimized for hardware efficiency, which is especially helpful for deep learning tasks. Furthermore, data loaders are designed for training, validation, and testing sets. They make data batching and shuffle processing easier, which is essential for training and assessing models. The BERT model is modified for sequence classification by changing the final layers using the

softmax function to correspond with the categories in the dataset, which are positive, negative, and neutral (Geetha & Renuka, 2021). Geetha and Renuka's study gives the softmax formula to classify the target, which is displayed in Equestion (7).

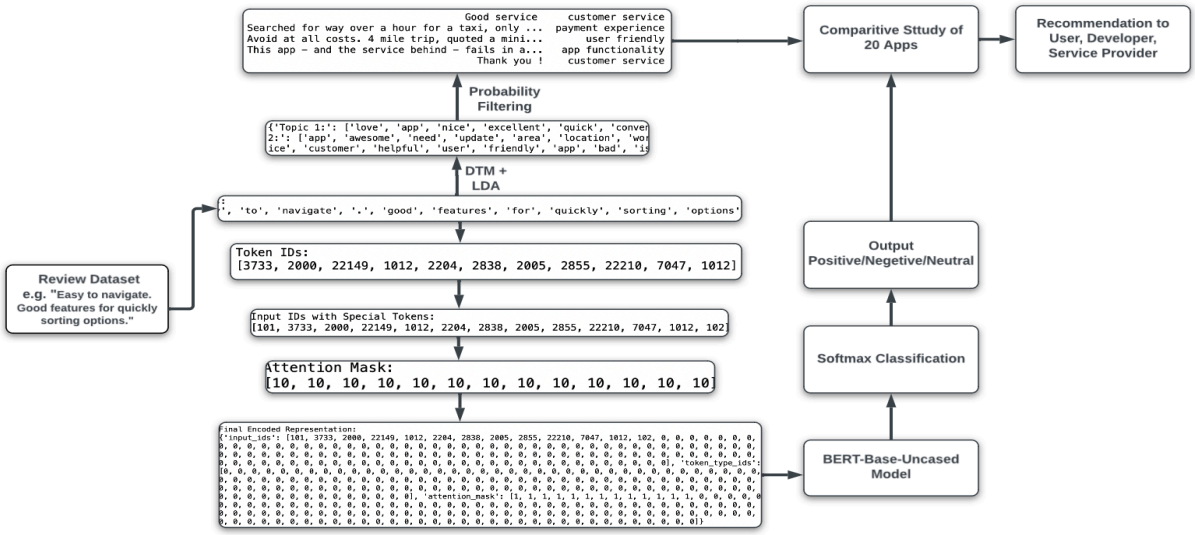
$$S(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \quad (7)$$

where $S(x_i)$ is the softmax function, e^{x_i} represents a standard exponential function for input vector n represents the number of output classes, e^{x_j}represents the standard exponential function for the output vector.

To promote optimal model learning, a study conducted by Myagmar et al. (2019), chose model optimizer AdamW and customized it for this study with specific parameters such as keeping learning rate 2e-5, weight decay is 0.01. Hyperparameter beta1 set to 0.9 and beta2 0.999, and an epsilon set to 1e-9, as proposed in the study done by Vaswani et al. (2017). Table 18 consists of all the hyperparameters that are used for this study. To ensure that the adaptive learning rate does not augment Amsgrad set to False. To balance the complexity of learning against the risk of overfitting, the learning process is organized with four epochs, gradient accumulation steps, and an early stopping mechanism. The dataset is divided into 80-10-10 ratios which represent training, validation, and test sets respectively. The training-validation loop is a cycle in which the model learns from training data and its performance is evaluated on the validation set. Metrics such as accuracy and loss are tracked to evaluate the model's learning progression. The sample code to preprocess and implement the BERT model is added in Appendix C.

Figure 72

Data Flow



Note: The above figure depicts the actual flow of data from raw dataset to sentiment classification to comparative study using LDA.

Table 18

Hyper-Parameters of the Fine-Tuned BERT-Base-Uncased

Hyperparameter	**Value**
Batch Size	16
Epochs	4
Gradient Accumulation Steps	4
Learning Rate	2e-5
Maximum Sequence Length	128
Attention Head	12

Hyperparameter	Value
Hidden Size	768
Hidden Layer	12
Train Parameter	110M

XLNet. The data flow architecture of the XLNet starts with the input layer. The preprocessed data after the transformation phase is split into 80% training data, 10% validation data, and the remaining 10 % for testing purposes. The pre-trained model includes 12 hidden layers of encoder-decoder transformer blocks with 768 hidden layers in each, a feed-forward layer size of 3072, and 12 self-attention heads. The input data is tokenized with positional encoding before training the model. Figure 73 represents the XLNet architecture and the complete data flow.

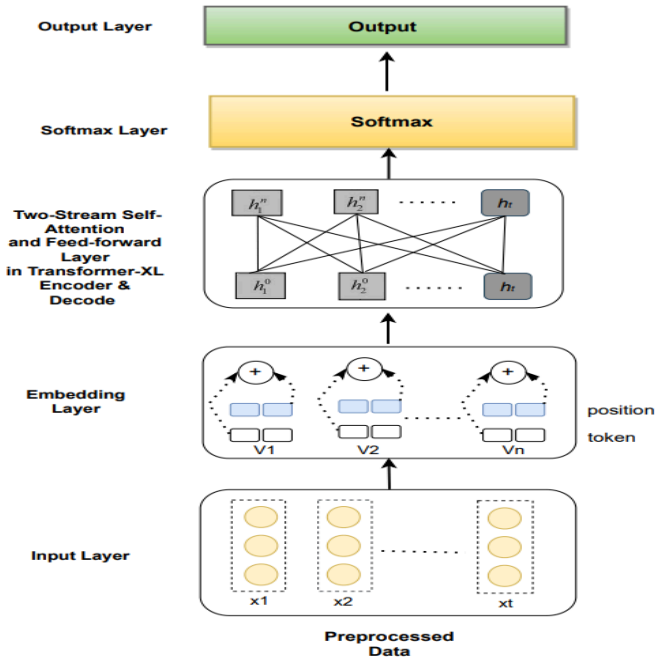
Input Layer. The preprocessed raw data after data transformation is passed to the input layer.

Embedding Layer. The objective of the embedding layer is to encode each token's inherent meaning and relationship with other words. In this study, the embedding layer is set to a size of 128 containing the semantic and contextual information of the input sentence. The model utilizes pre-trained word embeddings to map the tokens to high-dimensional vectors. In this phase, the position of each word is calculated based on the Equation (4) and (5). The embeddings are vectors representing the position of the tokens in the sequence. This helps the model understand the long-range dependencies and context. The token and positional embeddings are concatenated to create a single vector representation for each word. The quality of embedding

significantly impacts the model's performance and pre-trained on large corpus data captures rich semantic information. These embeddings lay the groundwork and transfer to the two-stream self-attention layer.

Figure 73

Architecture and Data Flow of XLnet



Note. The above structure illustrates the XLNet data flow and architecture

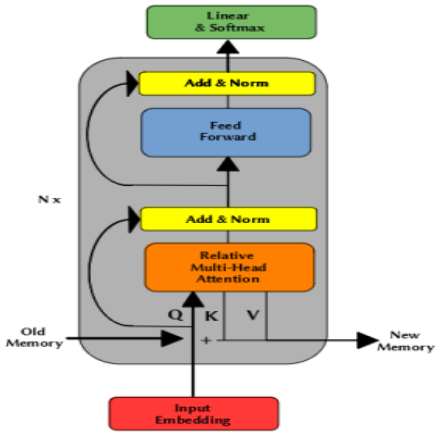
Two-Stream Self-Attention and Feed-Forward Layer. This is a pivotal layer where the XLnet model excels in understanding the bidirectional context. The content layers of the encoder are where the two-stream attention uses standard self-attention mechanisms to understand the meaning of the words. The query layer is where the positions and context of the tokens are captured by computing possible permutations of the input sequence. At this stage, XLNet learns how the change in the order of the token changes the meaning of the sentence. The permutations

of tokens are computed as in equation (6). After integration, the query stream and content are sent to the feed-forward network for further enhancement.

The input of the feed-forward network is the output of the two-stream self-attention mechanism in the previous layer which is transformed into multiple keys (K), value(V), and query(Q) vectors. These transformations involve weights for each token and relative positioning introduces the non-linear positional attention of the tokens. The attention scores are calculated by taking the dot product of modifying the query and key vectors. These scores represent the weight of the complete vector and are added as the output of the attention layers. The old memory stores the attention score of the previous attention layer and the new memory stores the new attention information relevant to the current attention score computation. The add and norm block of the Transformer-XL model adds the output of the previous attention layers to the input of the next attention layer. This feed-forward network of encoders works the same as a standard encoder except that the decoder includes the cross-attention mechanism for focusing on the relevant tokens for the downstream tasks and Masked self-attention where the model tries to predict the future tokens by the masking method. The Transformer-XL model does not explicitly distinguish the encoder and decoder but instead has a recurrence feed-forward network. The output of each attention layer is then concatenated to be passed as input to the Softmax layer. Figure 74 shows the transformer-XL architecture.

Figure 74

Transformer-XL Model



Note. The above figure depicts the transformer-XL architecture starting from the input embedding, feed-forward network, and the final Softmax output layer. From " Language Modeling for Source Code with Transformer-XL" by Dowdell, T., & Zhang, H. (2020).*arXiv preprint arXiv:2007.15813.*

Softmax Layer. The final layer of the XLNet model is the softmax layer which provides the probability distribution of the multiclass sentiment classification task. In this study, the input information is classified into positive, negative, and neutral target classes. The softmax layer produces a vector of probabilities for each class and the class with the highest probability is predicted as the output class. The sample code to preprocess and implement the XLNet model is added in Appendix C.

Model Comparison and Justification

The four different models employed for the study are compared in terms of the Functional and Non Functional characteristics depicting different aspects of each model. Table 19 provides the comparative information of the four models.

Table 19

Model Comparison

Characteristics	SVM	CNN Bi-LSTM	BERT	XLNet
Basic Architecture	Linear,Non Linear with help of Kernel Function	Combination of linear and nonlinear functions	Semi-supervised Transformer based architecture	Unsupervised XL-Transformed-based architecture
Data Type	Suitable for any data type such as images, audio, text.	Works well with sequential data or time series.	Mainly used for text data	Primarily used for Textual data in Natural Language Processing.
Data Size	Works well for small datasets, but underperforms for large datasets	Works well with moderate data size. More computational power required for larger datasets.	Large dataset is required to perform well on training dataset and to tune properly.	Large datasets required for best performance can be fine-tuned for smaller data and lower performance on extremely sparse datasets.

Characteristics	SVM	CNN Bi-LSTM	BERT	XLNet
Complexity	Linear SVM is relatively simple, compared to Non-Linear SVM which is computationally more expensive.	Double complexity as compared to the standard LSTM.	High, due to large number of parameters	High complexity due to the large number of parameters in the Transformer-XL model.
Strengths	Perform well for problems with a large number of features due to its efficiency in working in high dimensional feature space	Captures bidirectional dependencies, giving a better context to a review as a whole.	Outstanding at collecting subtle linguistic context	Built specifically for Natural Language Processing and has demonstrated outstanding results in long-range dependencies and complex sequences.
Limitations	It underperforms in cases of large datasets with high complexity	Underperforms while training on a large dataset with limited resources available.	Risk of overfitting on smaller data, costly in terms of resources	Requires substantial data for training and computational power.

Note. The figure above shows the comparison between the models in terms of different functional and non-functional characteristics.

Model Evaluation Method

Choosing the metrics to be used for evaluating the model performance is very critical. It has an enormous impact on how the models are measured and then compared with the other models. The project is performing the Sentimental analysis of the Review Data, which classifies the User Sentiment into Positive, Negative, and Neutral Sentiment.

The project used five performance metrics which are, Accuracy, Recall, F1-Score, Precision, and Confusion Matrix. Tatsat et al. (2020) explains that the most common terms in these metric calculations are True Positive(TP), False Positive(FP), True Negative(TN), and False Negative(FN). Here True Positive means the Prediction and actual values are both positive, False Positive means the Prediction is positive but the actual is negative, True Negative means both the Predicted and the actual are negative, and False Negative means the Prediction is negative and the actual is Positive.

Accuracy

Tatsat et al. (2020) define accuracy as the ratio of the number of accurate or correct predictions to the total number. It is the most commonly used method in performance evaluation. It depicts how well the model predicts the accurate values. Accuracy can be computed using the formula given in Equation (8). It shows how well the model predicts the User's Sentiment. The higher the accuracy, the better the model performs in classifying the sentiments.

$$\text{Accuracy} = \frac{True\ Positive + True\ Negative}{Total\ number} \quad (8)$$

It is the most suitable metric when there are an equal number of values in each target class. After the model training, the training data is balanced using the SMOTE technique so that each target class has an equal distribution. Hence, the metric is suitable for model evaluation.

Precision

Tatsat et al. (2020) define precision as the percentage of values that are positive out of the total number of values that are predicted positive. The denominator in this metric represents the accurately predicted value out of the given input data. Equation (9) represents the formula to compute the Precision of the model. A high precision value suggests that the model can classify the Review Sentiments with a high value of accuracy and minimal classification error.

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive} \quad (9)$$

Recall

Tatsat et al. (2020) define Recall as the percentage of values that are positive out of the total number of actual positive values. The denominator here is the total of actual positive values which are True Positive and False Negative values. The formula for Recall is given in Equation (10). Since the review sentiments can have varied sentiments, this measure can accurately predict the positive sentiments. It is often used together with Precision. With the increase in the prediction of positive sentiments, the number of false positives might also increase, which is a big tradeoff of this metric.

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad (10)$$

F1-Score

Bonnin (2017) defines F1-Score as the harmonic mean of recall and precision, where a measure F_{β} has the best value which is 1 and the worst score which is 0. When β is equal to 1,

F_β and F_1 are equal. Equation (11) represents the formula for F1-Score. The imbalance in the recall and precision metric is resolved with the help of F1-Score, which computes the harmonic mean of the two metrics. It is a more comprehensive performance measure that considers both false positives and false negatives.

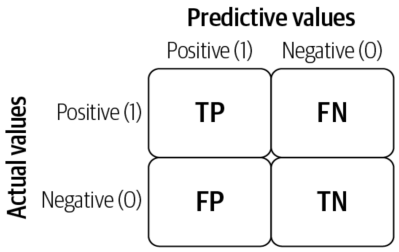
$$\text{F1-Score} = (1 + \beta^2) * \frac{Precision * Recall}{\beta^2 * (Precision + Recall)} \quad (11)$$

Confusion Matrix

Tatsat et al. (2020), define it as a square matrix that provides the value of True Positive, True Negative, False Positive, and False Negative. It provides a precise summary of the model performance. Each row represents the predictions made by the model, and the column represents the accurate or True values. It helps gauge the model's performance in predicting each Sentiment Class(Positive, Negative, and Neutral), as shown in Figure 75.

Figure 75

Confusion Matrix



Note. The figure above shows the Confusion matrix representing the Actual Values and the Predicted values on the y-axis, and the x-axis. Sourced from "Machine learning and data Science blueprints for finance.", by Tatsat, H., Puri, S., & Lookabaugh, B., 2020, O'Reilly Media,

Model Validation and Evaluation Results

Once all the model building is done, Evaluation Metrics for each of them are compared and listed all the metrics in Table 20.

Table 20

Model Comparison based on the Evaluation Metrics

Models	Accuracy	F1-Score	Precision	Recall
SVM	0.7708	0.7698	0.7693	0.7708
CNN Bi-LSTM	0.8602	0.8495	0.8976	0.8163
BERT	0.9232	0.9080	0.9004	0.9232
XLNet	0.9120	0.9013	0.8903	0.9201

Note. The above table depicts the evaluation results of the four models achieved in the Testing Dataset.

Conclusion

The comparison of the overall performance and accuracy of the model provides valuable insights into the model capabilities in this research on the sentimental analysis of mobile application user reviews. The BERT model outperformed the SVM, CNN-BiLSTM, and XLNet models by achieving an accuracy of 92.32% in the sentimental classification of the mobile application reviews. While SVM, CNN-BiLSTM, and XLNet achieved 77.08%, 86.02%, and 91.20% respectively. While all four models exhibit remarkable capabilities, BERT's contextual

embedding and attention mechanism prove instrumental in capturing the intricate user sentiment in the review content.

Limitations

Despite each model's significant and successful outcome, each model has its own set of limitations. The study shows SVM may not be best suited for large-scale intricate datasets. Despite the higher accuracy, CNN-BiLSTM has a possibility of missing crucial information due to its lower recall leading to significant drawbacks in the deployment of the application. On the other hand, both BERT and XLNet have complex architectures, which require high computational resources and restrict the applicability in citations involving resource constraints. Also, in scenarios requiring attention to precision, XLNet won't work the best.

Future Scope

Every model offers potential areas for development as it looks to the future. Future improvements to SVM might involve making it more capable of handling larger and more complex datasets, maybe by creatively designing features or combining them with other modeling approaches. Improvements that increase the CNN Bi-LSTM model's recall without sacrificing its high precision could be made possible by more sophisticated neural network architectures or specially designed algorithms. Because of its high resource requirements, BERT provides an ideal environment for studying optimization strategies that lessen its computational footprint and increase its usability and accessibility for a range of applications. Similarly, using advanced training approaches or model refinement techniques, XLNet might see increases in precision and computational efficiency, enhancing its performance in tasks requiring precision.

References

Al-Amrani, Y., Lazaar, M., & Kadiri, K. E. E. (2018). Random Forest and Support Vector Machine based Hybrid Approach to Sentiment Analysis. *Procedia Computer Science*, *127*, 511–520. https://doi.org/10.1016/j.procs.2018.01.150

Al-Harbi, O. (2021). A Deep Learning Approach Combining CNN and Bi-LSTM with SVM Classifier for Arabic Sentiment Analysis. *International Journal of Advanced Computer Science and Applications*, *12*(6). https://doi.org/10.14569/ijacsa.2021.0120618

Álvarez-Alvarado, J. M., Ríos-Moreno, J. G., Obregón-Biosca, S. A., Ronquillo-Lomelí, G., Ventura-Ramos, E., & Trejo-Perea, M. (2021). Hybrid Techniques to predict solar radiation using support vector machine and search optimization Algorithms: A review. *Applied Sciences*, *11*(3), 1044. https://doi.org/10.3390/app11031044

Assi, M., Hassan, S., Tian, Y., & Zou, Y. (2021). FeatCompare: Feature comparison for competing mobile apps leveraging user reviews. *Empirical Software Engineering*, *26*(5). https://doi.org/10.1007/s10664-021-09988-y

Adoma, A. F., Nunoo‐Mensah, H., & Chen, W. (2020). Comparative Analysis of Bert, Roberta, Distilbert, and Xlnet for Text-Based Emotion Recognition. *17th International Computer Conference on Wavelet Active Media Technology and Information Processing*. https://doi.org/10.1109/iccwamtip51612.2020.9317379

Bansal, A., Susan, S., Choudhry, A., & Sharma, A. (2022, May). COVID-19 Vaccine Sentiment Analysis During Second Wave in India by Transfer Learning Using XLNet. *International Conference on Pattern Recognition and Artificial Intelligence (pp. 443-454)* https://doi.org/10.1007/978-3-031-09282-4_37

Bhuvaneshwari, P., Rao, A. R., Robinson, Y. H., & Thippeswamy, M. N. (2022). Sentiment analysis for user reviews using Bi-LSTM self-attention based CNN model. *Multimedia Toolsand Applications*, *81*(9), 12405–12419. https://doi.org/10.1007/s11042-022-12410-4

Borg, A., & Boldt, M. (2020). Using VADER sentiment and SVM for predicting customer response sentiment. *Expert Systems With Applications*, *162*, 113746. https://doi.org/10.1016/j.eswa.2020.113746

Bonnin, R. (2017). *Machine learning for developers: Uplift your regular applications with the power of statistics, analytics, and machine learning*. Packt Publishing Ltd.

Chen, H., & Nguyen, H. (2020). Fine-tuning Pre-trained Contextual Embeddings for Citation Content Analysis in Scholarly Publication. *arXiv preprint arXiv:2009.05836.*

Dalpiaz, F., & Parente, M. (2019). RE-SWOT: from user feedback to requirements via competitor analysis. In *Lecture Notes in Computer Science* (pp. 55–70). https://doi.org/10.1007/978-3-030-15538-4_4

Devlin, J. (2018, October 11). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv.org. https://arxiv.org/abs/1810.04805

Dey, S., Wasif, S., Tonmoy, D. S., Sultana, S., Sarkar, J., & Dey, M. (2020). A Comparative Study of Support Vector Machine and Naive Bayes Classifier for Sentiment Analysis on Amazon Product Reviews. *2020 International Conference on Contemporary Computing and Applications (IC3A)*. https://doi.org/10.1109/ic3a48958.2020.233300

Dowdell, T., & Zhang, H. (2020). Language modeling for source code with transformer-xl. *arXiv preprint arXiv:2007.15813.*

Flaticon. (2023, December 8). *Analysis Icon - 809497*. https://www.flaticon.com/free-icons/analysis

Gao, C., Wang, B., He, P., Zhu, J., Zhou, Y., & Lyu, M. R. (2015). PAID: Prioritizing app issues for developers by tracking user reviews over versions. *2015 IEEE 26th International Symposium on Software Reliability Engineering*. https://doi.org/10.1109/issre.2015.7381797

Guo, L., Sharma, R., Lei, Y., Lu, R., & Rong, K. (2017). Automated competitor analysis using big data analytics. *Business Process Management Journal*, *23*(3), 735–762. https://doi.org/10.1108/bpmj-05-2015-0065

Geetha, M., & Renuka, D. K. (2021). Improving the performance of aspect based sentiment analysis using fine-tuned Bert Base Uncased model. *International Journal of Intelligent Networks*, *2*, 64–69. https://doi.org/10.1016/j.ijin.2021.06.005

Gong, X., Jin, J., & Zhang, T. (2019). Sentiment Analysis Using Autoregressive Language Modeling and Broad Learning System. *2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. https://doi.org/10.1109/bibm47256.2019.8983025

Hassan, A., & Mahmood, A. (2018). Convolutional Recurrent Deep Learning model for sentence classification. IEEE Access, 6, 13949–13957. https://doi.org/10.1109/access.2018.2814818

Habbat, N., Anoun, H., & Hassouni, L. (2021). A Novel Hybrid Network for Arabic Sentiment Analysis using fine-tuned AraBERT model. *International Journal on Electrical Engineering and Informatics*, 13(4), 801–812. https://doi.org/10.15676/ijeei.2021.13.4.3

Tian, H., Zhang, Z., Ren, M., Dong, C., Jia, X., & Zhuang, Q. (2023). Text Emotion Recognition based on XLNET-BIGRU-ATT. *Electronics*, *12*(12), 2704. https://doi.org/10.3390/electronics12122704

Jin, J., Ji, P., & Gu, R. (2016). Identifying comparative customer requirements from product online reviews for competitor analysis. *Engineering Applications of Artificial Intelligence*, *49*, 61–73. https://doi.org/10.1016/j.engappai.2015.12.005

JiFacts. (2022, June 19). *Python Scraper - Google Play and App Store reviews* [Video]. YouTube. https://www.youtube.com/watch?v=GVwjR6lkS6Q

Khan, S. I., Athawale, S. V., Borawake, M. P., & Naniwadekar, M. Y. (2023). Sentiment Analysis of Customer Reviews using Pre-trained Language Models. *International Journal of Intelligent Systems and Applications in Engineering*, 11(7s), 614-620.

Lee, G., & Raghu, T. S. (2014). Determinants of Mobile Apps' Success: Evidence from the App Store Market. *Journal of Management Information Systems*, *31*(2), 133–170. https://doi.org/10.2753/mis0742-1222310206

Lee, Y. (2021). Extraction of competitive factors in a competitor analysis using an explainable neural network. *Neural Processing Letters*, *53*(3), 1979–1994. https://doi.org/10.1007/s11063-021-10499-6

Li, Y., Jia, B., Guo, Y., & Chen, X. (2017). Mining user reviews for mobile app comparisons. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, *1*(3), 1–15. https://doi.org/10.1145/3130935

Liu, C. Z., Au, Y. A., & Choi, H. (2014). Effects of freemium strategy in the mobile app market: An Empirical Study of Google Play. *Journal of Management Information Systems*, *31*(3), 326–354. https://doi.org/10.1080/07421222.2014.995564

Lu, K., & Wu, J. (2019). Sentiment Analysis of Film Review Texts Based on Sentiment Dictionary and SVM. *3rd International Conference on Innovation in Artificial Intelligence,*73-77. https://doi.org/10.1145/3319921.3319966

Manek, A. S., Shenoy, P. D., Mohan, M., & R, V. K. (2016). Aspect term extraction for sentiment analysis in large movie reviews using Gini Index feature selection method and SVM classifier. *World Wide Web*, *20*(2), 135–154. https://doi.org/10.1007/s11280-015-0381-x

Mousa, A., & Schuller, B. (2017). Contextual bidirectional long short-term memory recurrent neural network language models: A generative approach to sentiment analysis.

Munikar, M., Shakya, S., & Shrestha, A. (2019). Fine-grained Sentiment Classification using BERT. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.1910.03474

Myagmar, B., Li, J., & Kimura, S. (2019). Cross-domain sentiment classification with bidirectional contextualized transformer language models. *IEEE Access*, *7*, 163219–163230. https://doi.org/10.1109/access.2019.2952360

Naseem, U., Razzak, I., Musiał, K., & Imran, M. (2020). Transformer-based Deep Intelligent Contextual Embedding for Twitter sentiment analysis. *Future Generation Computer Systems*, *113*, 58–69. https://doi.org/10.1016/j.future.2020.06.050

Ni, G., Zhang, X., Ni, X., Cheng, X., & Meng, X. (2023). A WOA-CNN-BiLSTM-based multi-feature classification prediction model for smart grid financial markets. *Frontiers in Energy Research*, *11*. https://doi.org/10.3389/fenrg.2023.1198855

Ouyang, Y., Guo, B., Lu, X., Han, Q., Guo, T., & Yu, Z. (2019). CompetitiveBike: Competitive analysis and popularity Prediction of Bike-Sharing apps using Multi-Source data. *IEEE Transactions on Mobile Computing*, *18*(8), 1760–1773. https://doi.org/10.1109/tmc.2018.2868933

Pota, M., Ventura, M., Catelli, R., & Esposito, M. (2020). An effective BERT-Based pipeline for Twitter Sentiment Analysis: A case study in Italian. *Sensors*, *21*(1), 133. https://doi.org/10.3390/s21010133

Roushangar, K., Ghasempour, R., & Biukaghazadeh, S. (2020). Evaluation of the parameters affecting the roughness coefficient of sewer pipes with rigid and loose boundary conditions via kernel-based approaches. *International Journal of Sediment Research*, *35*(2), 171–179. https://doi.org/10.1016/j.ijsrc.2019.08.004

Roushangar, K., & Ghasempour, R. (2023). Supporting vector machines. In *Elsevier eBooks* (pp. 411–422). https://doi.org/10.1016/b978-0-12-821285-1.00009-9

Smola, A.J., (1996). *Regression estimation with support vector learning machines.* [Master's Thesis, Technische Universit€at M€unchen, Germany].

Sousa, M. G., Sakiyama, K., De Souza Rodrigues, L., Moraes, P. H., Fernandes, E. R., & Matsubara, E. T. (2019). BERT for Stock Market Sentiment Analysis. *IEEE 31st International Conference on Tools With Artificial Intelligence (ICTAI)*, 1597–1601. https://doi.org/10.1109/ictai.2019.00231

Su, Y., Wang, Y., & Yang, W. (2019). Mining and Comparing User Reviews across Similar Mobile Apps. *The International Conference on Mobile Ad-Hoc and Sensor Networks (MSN)*, *1*, 338–342. https://doi.org/10.1109/msn48538.2019.00070

Tatsat, H., Puri, S., & Lookabaugh, B. (2020). *Machine learning and data Science blueprints for finance*. O'Reilly Media.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is All You Need. *arXiv (Cornell University)*, *30*, 5998–6008. https://arxiv.org/pdf/1706.03762v5

Xu, R., Frey, R. M., Fleisch, E., & Ilic, A. (2016). Understanding the impact of personality traits on mobile app adoption – Insights from a large-scale field study. *Computers in Human Behavior*, *62*, 244–256. https://doi.org/10.1016/j.chb.2016.04.011

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., & Le, Q. V. (2019). Xlnet: Generalized autoregressive pre training for language understanding. *Advances in neural information processing systems,* 32.5753–5763. https://arxiv.org/pdf/1906.08237

Zhao, H., Qin, Y., & Liu, Z. (2021). Impact of online customer reviews and deep learning on product innovation empirical study on mobile applications. *Business Process Management Journal*, *27*(6), 1912–1925. https://doi.org/10.1108/bpmj-12-2020-0542

Appendix A

Gantt Chart

The figure shows the overall Gantt Chart depicting the project schedule, which includes the list of tasks in each phase. Each task has a dependency on another task.

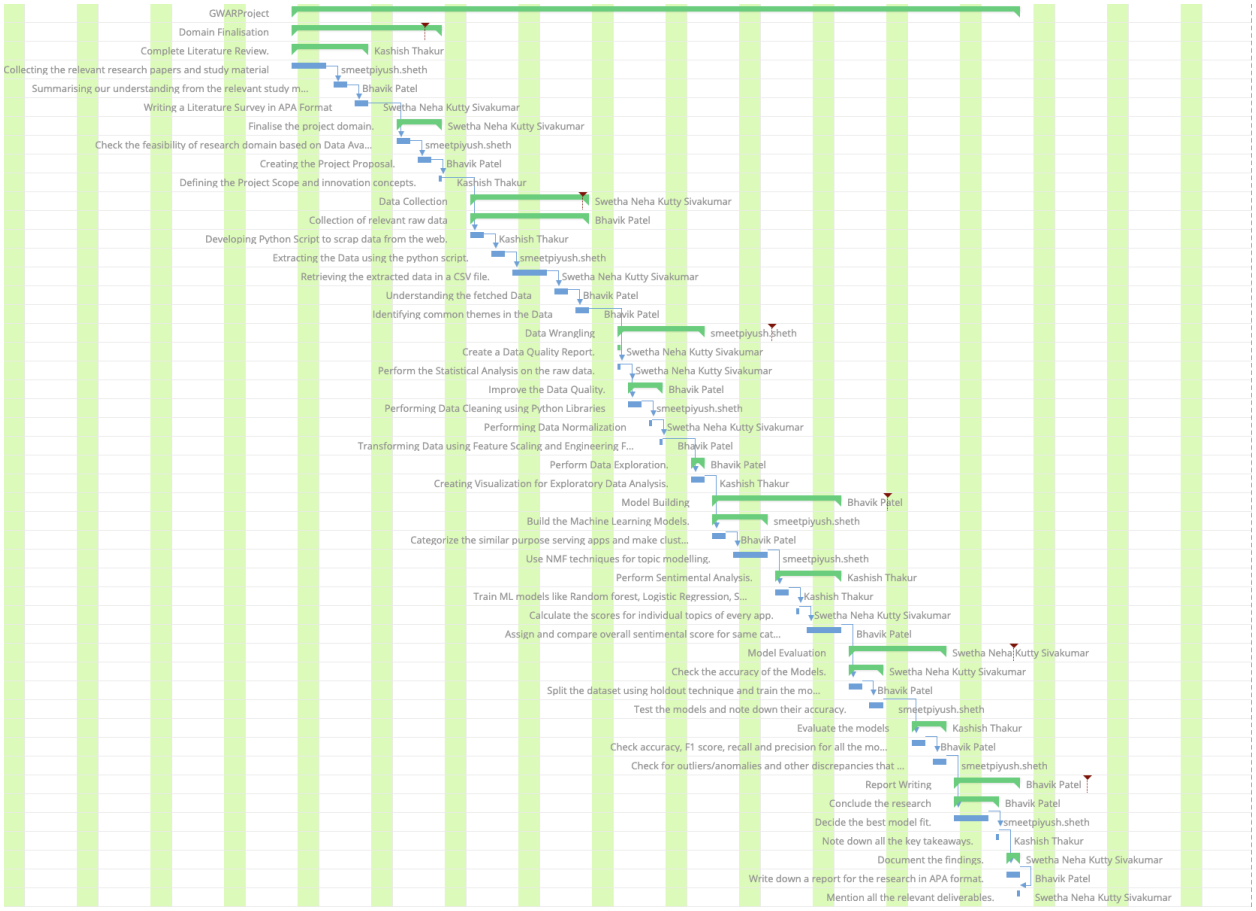


Figure A1. Gantt Chart for the Project displaying the overall Project Schedule

Appendix B

Data Preprocessing

The appendix here shows some of the code snippets that were written for preprocessing data. Figure B1 shows the code where processes such as converting the text to lowercase, tokenization, stopword removal, and lemmatization were performed. Figure B2 shows the topic modeling that was performed on the content and Figure B3 shows the assignment of those topics by using a statistical approach. Figure B4 shows the code for the implementation of Word2vec on the individual tokens. Figure B5 displays the data augmentation process performed on the data to handle the imbalance distribution.

```
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize

# Ensure necessary NLTK data is downloaded
nltk_data = ['punkt', 'wordnet', 'stopwords']
for data in nltk_data:
    nltk.download(data, quiet=True)

def tweet_to_words(tweet):
    if not isinstance(tweet, str):
        return []  # Return an empty list for non-string input

    # Lowercase the text
    text = tweet.lower()

    # Tokenization
    tokens = word_tokenize(text)

    # Remove stopwords
    stop_words = set(stopwords.words('english'))
    tokens = [word for word in tokens if word.lower() not in stop_words and word.isalpha()]

    # Apply lemmatization
    lemmatizer = WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(word) for word in tokens]

    # Return the processed list of words
    return tokens

# Apply the function to the 'content' column
df1['processed_content'] = df1['content'].apply(tweet_to_words)
```

Figure B1. The code illustrates the text preprocessing steps performed on the Review Content Column.

```
import pandas as pd
from collections.abc import Iterable  # For checking if the value is an iterable
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation

# Function to join tokens and handle non-iterable values
def join_tokens(tokens):
    if isinstance(tokens, Iterable) and not isinstance(tokens, str):
        return ' '.join(tokens)
    return ''  # Replace non-iterable values with an empty string

# Apply the function to your DataFrame
reduced_df['processed_text'] = reduced_df['processed_content'].apply(join_tokens)

# Example of further code
count_vectorizer = CountVectorizer(max_df=0.95, min_df=2, stop_words='english')
dtm = count_vectorizer.fit_transform(reduced_df['processed_text'])

num_topics = 10
lda = LatentDirichletAllocation(n_components=num_topics, random_state=0)
lda.fit(dtm)

# Function to display the top words in each topic
def display_topics(model, feature_names, no_top_words):
    topic_dict = {}
    for topic_idx, topic in enumerate(model.components_):
        topic_dict["Topic %d:" % (topic_idx + 1)] = [feature_names[i] for i in topic.argsort()[:-no_top_words - 1:-1
    return topic_dict

# Displaying the top words for each topic
no_top_words = 10
topic_words = display_topics(lda, count_vectorizer.get_feature_names_out(), no_top_words)
print(topic_words)
```

Figure B2. Code to perform Topic Modeling, where different topics in the dataset are identified.

```
topic_distributions = lda.transform(dtm)

# The indices of topics of interest
topics_of_interest = [7, 4, 8, 1]  # these indices are one less than the topic numbers because of 0-indexing

# Filter the topic distributions to only include the four topics of interest
filtered_distributions = topic_distributions[:, topics_of_interest]


# Get the index of the topic with the max probability for each comment
# The index will correspond to the position in topics_of_interest
topic_indices = filtered_distributions.argmax(axis=1)

# Map the index to the class names
class_names = {0: "app functionality", 1: "customer service", 2: "user friendly", 3: "payment experience"}

# Use the mapping to translate indices to class names
reduced_df['topic_label'] = [class_names[index] for index in topic_indices]

# Now reduced_df will have a new column 'topic_label' with class names as labels
```

Figure B3. Code to assign the records to their corresponding Topic Label.

```
# Initialize stopwords and stemmer
stop_words = set(stopwords.words('english'))
stemmer = PorterStemmer()

def vectorize_text(word_list, model):
    # Remove out-of-vocabulary words
    word_list = [word for word in word_list if word in model.wv.key_to_index]

    # If no words are left in the sentence after filtering, return a zero vector
    if len(word_list) == 0:
        return np.zeros(model.vector_size)

    # Compute the average Word2Vec for the remaining words
    word_vectors = [model.wv[word] for word in word_list]
    vectorized_text = np.mean(word_vectors, axis=0)
    return vectorized_text

# Fill NaN values with an empty list in 'processed_content' column
reduced_df['processed_content'] = reduced_df['processed_content'].apply(lambda x: [] if isinstance(x, float) else x)

# Train the Word2Vec model
word2vec_model = Word2Vec(sentences=reduced_df['processed_content'], vector_size=100, window=5, min_count=1, workers

# Apply the vectorization function to each row in the DataFrame
X = np.array([vectorize_text(tokens, word2vec_model) for tokens in reduced_df['processed_content']])

# Label encoding for the target variable
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(reduced_df['category'])

# Splitting the data into train, validation, and test sets with stratification
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.2, stratify=y, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, stratify=y_temp, random_state=42)
```

Figure B4. Code implementation of Stemming, Vectorization, and splitting the train, validation, and testing data.

```
smote = SMOTE(random_state=42)
X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)

# Now, X_train_smote and y_train_smote are your new training sets with balanced classes

# Displaying the shape of the datasets
print("Original Training Set Shape:", X_train.shape, y_train.shape)
print("SMOTE Resampled Training Set Shape:", X_train_smote.shape, y_train_smote.shape)
```

Figure B5. Code Implementation of SMOTE on Training Data

Appendix C

Model Implementation and Prediction

```
from sklearn.preprocessing import StandardScaler
# Standardize the data
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_val_scaled = scaler.transform(X_val)
X_test_scaled = scaler.transform(X_test)

# Initialize the SVM model
svm_model = SVC(kernel='rbf', C=1.0, degree=3, random_state=42,max_iter=100000)

import time
start_time = time.time()

# Train the SVM model on the training set
svm_model.fit(X_train_scaled, y_train)

end_time = time.time()
elapsed_time = end_time - start_time
print(f"Training time: {elapsed_time} seconds")
```

Figure C1. Code implementation of SVM

```
# Model parameters
embedding_size = 100
max_len = 100
epochs = 10
learning_rate = 0.1
momentum = 0.8
# SGD Optimizer
sgd = LegacySGD(learning_rate=learning_rate, momentum=momentum, nesterov=False)
# Learning Rate Scheduler
def scheduler(epoch, lr):
    if epoch % 5 == 0 and epoch != 0:
        return lr * 0.1
    else:
        return lr

lr_scheduler = LearningRateScheduler(scheduler)
# Early Stopping to prevent overfitting
early_stopping = EarlyStopping(monitor='val_loss', patience=5, verbose=1)

# Building the Model with two BiLSTM layers
model = Sequential()
model.add(Conv1D(filters=32, kernel_size=3, padding='same', activation='relu', input_shape=(max_len, 1)))
model.add(MaxPooling1D(pool_size=2))
model.add(Bidirectional(LSTM(32)))
model.add(Dropout(0.4))
model.add(Dense(3, activation='softmax'))  # Adjust the number of units to match the number of classes

# Model Summary
```

Figure C2. Code implementation of CNN_BiLSTM

```python
# Function to preprocess texts for BERT
def preprocess_for_bert(texts, tokenizer, max_len=128):
    encoded_batch = tokenizer.batch_encode_plus(
        texts,
        add_special_tokens=True,
        padding='max_length',
        truncation=True,
        max_length=max_len,
        return_attention_mask=True,
        return_tensors='pt'
    )
    return encoded_batch['input_ids'], encoded_batch['attention_mask']

# Load the BERT tokenizer
print("Loading BERT tokenizer...")
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

# Load your dataset
print("Loading dataset...")
df = pd.read_csv('/Users/bhavikpatel/Desktop/poject MSDA/Data 270/Individual paper/data/BERT_clean_file.csv').sample
df['content'] = df['content'].astype(str)

# Encode labels
label_encoder = LabelEncoder()
labels = label_encoder.fit_transform(df['category'])
```

Figure C3 Preprocessing of BERT

```python
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = BertForSequenceClassification.from_pretrained('bert-base-uncased', num_labels=3)
model.to(device)
optimizer = AdamW(model.parameters(), lr=2e-5, weight_decay=0.01, betas=(0.9, 0.999), eps=1e-9, amsgrad=False)
use_cuda = torch.cuda.is_available()
scaler = GradScaler(enabled=use_cuda)
epochs = 4
accumulation_steps = 4
early_stopping_patience = 3
best_val_loss = float('inf')
epochs_no_improve = 0

train_accuracies = []
val_accuracies = []
train_losses = []
val_losses = []
val_true_labels = []
val_pred_labels = []

for epoch_i in range(epochs):
    total_train_loss = 0
    total_val_loss = 0
    # Training phase with gradient accumulation
    model.train()
    total_loss, total_correct, total = 0, 0, 0
    for step, batch in tqdm(enumerate(train_loader), total=len(train_loader)):
        b_input_ids, b_input_mask, b_labels = tuple(t.to(device) for t in batch)
        model.zero_grad()
        outputs = model(b_input_ids, token_type_ids=None, attention_mask=b_input_mask, labels=b_labels)
        loss = outputs.loss / accumulation_steps
        loss.backward()
        total_train_loss += loss.item()

        if (step + 1) % accumulation_steps == 0:
            optimizer.step()
            optimizer.zero_grad()
```

Figure C4. Code implementation of BERT

```python
# Function to preprocess text for XLNet
def preprocess_for_xlnet(texts, tokenizer, max_len=128):
    encoded_batch = tokenizer.batch_encode_plus(
        texts,
        add_special_tokens=True,
        padding='max_length',
        truncation=True,
        max_length=max_len,
        return_attention_mask=True,
        return_tensors='pt'
    )
    return encoded_batch['input_ids'], encoded_batch['attention_mask']
tokenizer = XLNetTokenizer.from_pretrained('xlnet-base-cased')
```

Figure C5. Preprocessing for XLNet

```python
model = XLNetForSequenceClassification.from_pretrained('xlnet-base-cased', num_labels=3)  # 3 labels
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
optimizer = AdamW(model.parameters(), lr=2e-5, weight_decay=1e-2)
class_counts = df['category'].value_counts()
class_weights = 1. / class_counts
weights = class_weights[label_encoder.transform(df['category'].unique())]
weights = torch.tensor(weights, dtype=torch.float).to(device)
# Early Stopping Setup
accumulation_steps = 4
early_stopping_patience = 3
best_val_loss = float('inf')
epochs_no_improve = 0
epochs = 4  # Adjust as needed
train_accuracies = []
val_accuracies = []
train_losses = []
val_losses = []
val_preds = []
val_labels = []
for epoch_i in range(epochs):
    # Training phase with gradient accumulation
    model.train()
    total_loss, total_correct, total = 0, 0, 0
    for step, batch in tqdm(enumerate(train_loader), total=len(train_loader)):
        b_input_ids, b_input_mask, b_labels = tuple(t.to(device) for t in batch)

        optimizer.zero_grad()

        # Automatic mixed precision
        outputs = model(input_ids=b_input_ids, token_type_ids=None, attention_mask=b_input_mask, labels=b_labels)
        loss = outputs.loss / accumulation_steps

        loss.backward()

        if (step + 1) % accumulation_steps == 0:
            optimizer.step()
            optimizer.zero_grad()
```

Figure C6. Code implementation of XLNet

Link to the GitHub Repository:

https://github.com/bxbpel13/Comparative-Study-of-Mobile-App-Rivals