

A Project Report on

Product Recommendation using
Feature-Level Analysis

submitted in partial fulfillment for the award of

Bachelor of Technology

in

Computer Science and Engineering

by

B. Pranavi (Y20ACS535)

R. Likhitha (L21ACS417)

Sk. Abdul Gouse (Y20ACS557)

R. Mahathi (Y20ACS551)



Under the guidance of
Mr. P. Nanda Kishore.
Assistant Professor
Department of Computer Science and Engineering
Bapatla Engineering College
(Autonomous)
(Affiliated to Acharya Nagarjuna University)
BAPATLA – 522 102, Andhra Pradesh, INDIA
2023-2024

**Department of
Computer Science and Engineering**



CERTIFICATE

This is to certify that the project report entitled **Product Recommendation using Feature-Level Analysis** that is being submitted by B. Pranavi (Y20ACS535), R. Likhitha (L21ACS417), Sk. Abdul Gouse (Y20ACS557), R. Mahathi(Y20ACS551) and in partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Science and Engineering to the Acharya Nagarjuna University is a record of bonafide work carried out by them under our guidance and supervision.

Date:

Signature of the Guide
P. Nanda Kishore
Asst. Prof

Signature of the HOD
Dr. M. Rajesh Babu
Associate Professor

DECLARATION

We declare that this project work is composed by ourselves, that the work contained herein is our own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

B. Pranavi (Y20ACS535)

R. Likhitha (L21ACS417)

Sk. Abdul Gouse (Y20ACS557)

R. Mahathi (Y20ACS551)

Acknowledgement

We sincerely thank the following distinguished personalities who have given their advice and support for successful completion of the work.

We are deeply indebted to our most respected guide **P. Nanda Kishore**, Assistant. Prof. Department of CSE, for his valuable and inspiring guidance, comments, suggestions and encouragement.

We extend our sincere thanks to **Dr. M. Rajesh Babu**, Assoc. Prof. & Head of the Dept. for extending his cooperation and providing the required resources.

We would like to thank our beloved Principal **Dr. Nazeer Shaik** for providing the online resources and other facilities to carry out this work.

We would like to express our sincere thanks to our project coordinator **Dr. N. Sudhakar**, Prof. Dept. of CSE for his helpful suggestions in presenting this document.

We extend our sincere thanks to all other teaching faculty and non-teaching staff of the department, who helped directly or indirectly for their cooperation and encouragement.

B. Pranavi (Y20ACS535)

R. Likhitha (L21ACS417)

Sk. Abdul Gouse (Y20ACS557)

R. Mahathi (Y20ACS551)

Table of Contents

List of Figures	vii
List of Tables	viii
Abstract	ix
1 Introduction.....	1
1.1. Problem Statement and objective.....	2
1.2. Technology Background	2
1.2.1. Machine Learning using Python	2
1.2.2. Libraries and Algorithms of project.....	4
1.3. Runtime Environment – Jupyter notebook	11
2 Literature Survey	12
3 Proposed System.....	14
3.1 Various modules in the project.....	14
3.1.1 User Interface and Engagement	14
3.1.2 Feature Selection and Feedback Collection	14
3.1.3 Data Processing and Sentiment Analysis.....	15
3.1.4 Feature Importance Assessment and Categorization	15
3.1.5 Recommendation Generation.....	15
3.2 Dataset.....	15
3.3 Preprocessing	17
3.4 Sentiment Analysis.....	17

3.4.1	Overview	18
3.4.2	Key Components of Sentiment Analysis	18
3.4.3	Applications of Sentiment Analysis.....	19
3.5	VADER algorithm for sentiment analysis	20
3.5.1	Overview	21
3.5.2	Key components of VADER	21
3.5.3	VADER Architecture	22
3.6	K-means clustering.....	23
3.6.1	Overview	23
3.6.2	Working of Kmeans algorithm	23
3.6.3	Kmeans Architecture	25
3.7	Proposed architecture	26
3.8	Advantages of Proposed system.....	27
4	Design	29
4.1	Usecase diagram.....	30
4.2	Class diagram	31
4.3	Activity diagram.....	32
4.4	Statechart diagram.....	33
4.5	Sequence diagram	34
5	Implementation	35
5.1	Credibility assessment.....	35

5.2	Feature selection.....	37
5.3	Feedback collection and Dataset generation	38
5.4	Data preprocessing	39
5.5	Sentiment analysis.....	41
5.6	Feature strength categorization	43
5.7	Recommendation generation.....	44
6	Conclusions and Future Work	46
6.1	Future enhancements.....	46
7	References.....	49

List of Figures

Figure 3.1 VADER Algorithm Workflow	22
Figure 3.2 KMeans Architecture.....	25
Figure 3.3 Proposed Architecture	26
Figure 4.1 Usecase Diagram	30
Figure 4.2 Class Diagram	31
Figure 4.3 Activity Diagram	32
Figure 4.4 Statechart Diagram	33
Figure 4.5 Sequence Diagram.....	34
Figure 5.1 Credibility assessment	36
Figure 5.2 Error.....	36
Figure 5.3 Feature Selection	38
Figure 5.4 Feedback Collection	39
Figure 5.5 Generated Dataset.....	39
Figure 5.6 Handling Missing Data.....	40
Figure 5.7 Preprocessed Dataset	41
Figure 5.8 Generating Sentiment Score	42
Figure 5.9 Generating Final Feature Rating	43
Figure 5.10 Feature Strength Categorization	44
Figure 5.11 Recommendation Generation	45

List of Tables

Table 3.1 Initial Dataset	16
Table 3.2 Generated Dataset After Feedback Collection.....	17

Abstract

In this project, we present a systematic approach to recommending products based on a comprehensive assessment of customer preferences and feedback. The methodology involves a multi-step user interface where customers answer boolean questions to determine their interest, select desired product features, and provide opinions through reviews or ratings. The collected data is organized into a structured dataset, which undergoes preprocessing to address missing values and prepare textual reviews for sentiment analysis using the VADER algorithm.

A novel combined metric is derived, integrating sentiment scores and feature ratings, to classify product features into categories (strong, moderate, weak) using K-means clustering. The final recommendation decision is made based on the balance of strong and moderate features relative to the total selected features. This approach offers a systematic framework for personalized product recommendations informed by customer sentiment and feature preferences.

Keywords: Sentiment analysis, K-means clustering, VADER algorithm, recommendations.

1 Introduction

In today's competitive marketplace, personalized product recommendations play a pivotal role in enhancing customer satisfaction and driving sales. This project aims to develop an intelligent recommendation system that leverages customer feedback and preferences to suggest products tailored to individual needs. The methodology involves a structured interface where customers engage in a series of interactions: from initial assessment of interest through boolean questions, to selection of desired features, and finally, providing opinions via reviews or ratings.

The core of this project lies in the effective utilization of user-generated data. By collecting and organizing customer inputs into a dataset, we enable systematic analysis and processing to derive valuable insights. A key challenge addressed in this project is the handling of missing data and the preprocessing of textual reviews to prepare them for sentiment analysis.

To gauge customer sentiment effectively, the VADER (Valence Aware Dictionary and sEntiment Reasoner) algorithm is employed to assign sentiment scores to textual reviews. These scores, combined with feature ratings, form a novel metric used to categorize product features into distinct groups (strong, moderate, weak) using K-means clustering. This categorization provides a nuanced understanding of feature significance based on customer sentiment and feedback.

Ultimately, the recommendation decision is driven by the balance and strength of features selected by the customer. If the combined count of strong and moderate features surpasses a certain threshold relative to the total features selected, a product is recommended as strong; otherwise, it is categorized as weak.

This project not only showcases a practical application of data-driven decision-making in product recommendations but also highlights the value of customer-centric design in enhancing the overall user experience. It can be included with real-time applications of recommendation systems, as it gives high preference to reviews and ratings collected at feature-level, which in turn helps recommendation systems to be more personalized and accurate.

1.1. Problem Statement and objective

Generating recommendation for products based on a combination of customer reviews and feature ratings presents challenges in data collection, processing, and analysis. Ensuring accurate classification of product strength and providing meaningful recommendations requires effective handling of missing data, sentiment analysis of customer reviews, and feature classification.

1.2. Technology Background

The technical background provides an overview of the foundational technologies and methodologies utilized in the project to achieve desired results. It can be analyzed after finalizing a idea to be implemented. The technical background plays a crucial role as it effects the results to be achieved. Suitable tools need to be used such that accurate results can be achieved. After selecting the domain, various approaches can be checked, such that we finally derive at that one mostly relatable approach for our model.

1.2.1. Machine Learning using Python

Machine learning enables systems to learn from data and make predictions or decisions without being explicitly programmed. Python has emerged as one of the most popular

programming languages for machine learning due to its simplicity, extensive libraries, and vibrant community support.

When working with machine learning in Python, it's essential to understand the fundamental concepts of the field, such as supervised learning, unsupervised learning, and reinforcement learning. Supervised learning involves training a model on labeled data to make predictions or decisions, while unsupervised learning involves finding patterns or relationships in unlabeled data. Reinforcement learning focuses on training agents to make sequential decisions through interactions with an environment.

In Python, several libraries are commonly used for machine learning tasks. One of the most well-known is scikit-learn, which provides a simple and efficient toolkit for data mining and data analysis. Scikit-learn includes a wide range of algorithms for tasks such as classification, regression, clustering, dimensionality reduction, and model selection. Its user-friendly interface makes it an excellent choice for those new to machine learning.

In addition to these libraries, Python also offers powerful tools for data manipulation and analysis, such as pandas and NumPy. Pandas provides data structures and functions that simplify working with structured data, while NumPy offers support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently.

For deep learning tasks, TensorFlow and PyTorch are widely used libraries that offer more flexibility and efficiency for building and training neural networks. TensorFlow, developed by Google, provides a comprehensive ecosystem of tools and resources for machine learning, while PyTorch, developed by Facebook's AI Research

lab (FAIR), offers a dynamic approach that is favored by researchers for its ease of use and flexibility.

Besides these, python has powerful tools for data visualization, which is matplotlib. Matplotlib provides various tools for creating static, interactive, and animated visualizations. The library supports a wide range of plot types, including line plots, scatter plots, bar charts, histograms, pie charts, and more. Matplotlib also offers customization options to control the appearance of plots, such as colors, labels, titles, and axes formatting. Additionally, it integrates well with other libraries like NumPy and pandas, enabling seamless data manipulation and visualization workflows.

Not only these tools, there are many such other tools which make machine learning with python more easy and popular among researchers and students. The tools in python assists us to implement machine learning models effectively. In our project, we dealt with some of the tools in python to implement our proposed idea and to get the desired results. These tools made the tasks easy as they can be understood and implemented easily.

1.2.2. Libraries and Algorithms of project

The libraries and algorithms form the technical backbone of the project, enabling data preprocessing, sentiment analysis, clustering, and user interface development. By leveraging these tools effectively, desired application can be built which lays emphasis on feature-level analysis.

A. Pandas

Pandas is a powerful and popular open-source library in Python designed specifically for data manipulation and analysis. It provides easy-to-use data structures, such as DataFrame and Series, that allow users to efficiently handle and analyze structured data making it easier to clean, transform, and analyze datasets.

The primary data structure in pandas is the DataFrame, which represents tabular data with rows and columns. A DataFrame can be thought of as a two-dimensional labeled data structure with labeled axes (rows and columns). This makes it easy to perform various operations like filtering, grouping, reshaping, and aggregating data. Pandas also offers the Series data structure, which is one-dimensional and represents a single column or row of data within a DataFrame.

One of the key features of pandas is its ability to handle missing or incomplete data gracefully. It provides convenient methods for identifying, removing, or filling missing values, which is essential for data cleaning and preprocessing tasks. Pandas also supports reading and writing data from various file formats, including CSV, Excel, SQL databases, JSON, and more, simplifying the process of importing and exporting data.

Moreover, pandas includes powerful functionalities for data manipulation, such as indexing, slicing, merging, joining, and reshaping datasets. It allows users to perform complex operations on data quickly and efficiently. Additionally, pandas integrates well with other Python libraries like NumPy and scikit-learn, enabling seamless integration into data analysis and machine learning workflows. Overall, pandas is a versatile and essential library for anyone working with data in Python, providing a robust toolkit for data cleaning, exploration, and analysis.

B. Tkinter

Tkinter is a standard GUI (Graphical User Interface) toolkit for Python, included with the standard Python distribution. It provides a set of modules and classes to create simple and interactive desktop applications with graphical interfaces. Tkinter is based on the Tk GUI toolkit, which originated as part of the Tcl (Tool Command Language) scripting language but has since been widely adopted by other programming languages, including Python.

One of the main advantages of Tkinter is its simplicity and ease of use, making it an excellent choice for beginners learning GUI programming in Python. Tkinter allows developers to create windows, buttons, menus, labels, text boxes, canvas, and other GUI components using intuitive and straightforward syntax. This simplicity makes it accessible to programmers who want to quickly prototype or build small-scale applications with graphical interfaces.

Tkinter supports a variety of widgets that can be customized and organized to design user-friendly interfaces. Widgets include buttons for triggering actions, labels for displaying text or images, entry widgets for accepting user input, and canvas widgets for drawing shapes and graphics. Tkinter widgets can be arranged using geometry managers such as pack, grid, or place, which provide flexibility in laying out the components within a window.

Another benefit of Tkinter is its platform independence. Tkinter applications can run on different operating systems, including Windows, macOS, and Linux, without requiring modifications to the code. This cross-platform compatibility makes Tkinter a versatile choice for developing desktop applications that target a broad audience across various environments.

Although Tkinter is primarily used for building simple to moderate complexity GUI applications, it can be extended and customized using third-party libraries and tools. For more advanced GUI development, developers can leverage additional packages and frameworks that build upon Tkinter's foundation, offering enhanced capabilities and features for building sophisticated graphical applications.

In summary, Tkinter is a straightforward and versatile library for creating graphical user interfaces in Python. Its ease of use, cross-platform compatibility, and integration with the Python ecosystem make it a practical choice for developers interested in building desktop applications with graphical interfaces, from basic utilities to interactive tools and prototypes.

C. Scikit-learn (sklearn)

Scikit-learn, often referred to as sklearn, is a powerful and user-friendly open-source library for machine learning in Python. It provides a wide range of tools and algorithms for building and deploying machine learning models efficiently. Scikit-learn is built on top of other popular Python libraries such as NumPy, SciPy, and matplotlib, making it easy to integrate into existing data analysis workflows.

One of the key strengths of scikit-learn is its simplicity and consistency in design. It offers a clean and intuitive interface for implementing various machine learning algorithms, including supervised learning (e.g., classification, regression), unsupervised learning (e.g., clustering, dimensionality reduction), and model selection. Scikit-learn emphasizes ease of use and provides a uniform API across different algorithms, making it accessible to both beginners and experienced practitioners.

Scikit-learn includes a comprehensive set of functionalities for data preprocessing, feature extraction, feature selection, and model evaluation. It provides

tools for handling missing values, scaling and normalizing data, encoding categorical variables, and splitting data into training and testing sets. Additionally, scikit-learn offers a variety of metrics and scoring functions to assess the performance of machine learning models accurately.

Another notable feature of scikit-learn is its extensive collection of machine learning algorithms and techniques. It includes implementations of popular algorithms such as linear and logistic regression, support vector machines (SVM), decision trees, random forests, k-nearest neighbors (KNN), k-means clustering, and more. Scikit-learn also supports advanced techniques like ensemble methods, feature selection, and model validation through cross-validation.

Furthermore, scikit-learn supports model persistence and serialization, allowing users to save trained models to disk and reload them later for deployment or further analysis. The library is actively maintained and regularly updated with new features and improvements, reflecting the vibrant community of developers and researchers contributing to its development. Overall, scikit-learn is an essential tool for anyone interested in applying machine learning techniques to real-world problems using Python. Its simplicity, versatility, and robustness make it a valuable asset in the data science toolbox.

D. NLTK (Natural Language ToolKit)

NLTK (Natural Language Toolkit) is a popular open-source library for natural language processing (NLP) in Python. It provides a suite of libraries and programs for symbolic and statistical NLP tasks, such as tokenization, stemming, tagging, parsing, and semantic reasoning. NLTK is widely used by researchers, educators, and practitioners to explore and work with human language data.

One of the key features of NLTK is its extensive collection of text processing libraries and corpora. NLTK includes tools for tokenization, which involves splitting text into words or sentences, and stemming, which reduces words to their base or root form. It also offers part-of-speech tagging, named entity recognition, and syntactic parsing functionalities, enabling users to analyze and understand the structure of text data.

NLTK provides access to various corpora and lexical resources, including text collections, word lists, and annotated datasets. These resources are invaluable for training and evaluating NLP models, conducting linguistic research, and building applications that require linguistic knowledge. NLTK's built-in support for these resources makes it convenient for users to experiment with different datasets and leverage them for specific NLP tasks.

Moreover, NLTK offers implementations of popular algorithms and models in computational linguistics, such as n-gram language models, machine translation, and sentiment analysis. It also supports integration with other libraries and tools for machine learning and data analysis, such as scikit-learn and TensorFlow, enabling users to combine NLP techniques with broader data science workflows.

Another notable aspect of NLTK is its educational value. It is often used as a teaching tool in NLP and computational linguistics courses due to its comprehensive documentation, tutorials, and examples. NLTK's modular design and ease of use make it accessible to learners at different levels of expertise, from beginners exploring basic NLP concepts to advanced researchers developing state-of-the-art NLP applications.

In summary, NLTK is a versatile and powerful library for natural language processing in Python, offering a rich set of tools, resources, and algorithms for

analyzing and manipulating text data. Whether you are interested in linguistic research, text analysis, or building NLP applications, NLTK provides a solid foundation and practical utilities to support a wide range of NLP tasks.

E. VADER (Valence Aware Dictionary and sEntiment Reasoner)

The VADER (Valence Aware Dictionary and sEntiment Reasoner) algorithm is a lexicon and rule-based sentiment analysis tool designed specifically for analyzing text sentiment. VADER is built upon a pre-existing lexicon of sentiment-related words and phrases, which are scored based on their polarity (positive, negative, or neutral) and intensity. One of the key strengths of the VADER algorithm is its ability to handle sentiment analysis in a context-aware manner, when data is informal, ambiguous, and context-dependent.

VADER is implemented as part of the NLTK (Natural Language Toolkit) library in Python, making it accessible and easy to use for text analysis tasks. It has gained popularity in various applications, including social media monitoring, customer feedback analysis, brand sentiment analysis, and opinion mining. While VADER provides a robust and efficient approach to sentiment analysis, it is important to note that its performance may vary depending on the nature of the text and the specific context of the application.

F. K-means Algorithm

The k-means algorithm is a popular clustering technique used in unsupervised machine learning to partition a set of data points into k clusters based on similarity. The goal of the k-means algorithm is to minimize the variance within clusters by iteratively assigning data points to the nearest cluster centroid and updating the centroids to minimize the total within-cluster variance.

The k-means algorithm aims to optimize the objective function known as the within-cluster sum of squares (WCSS), where the goal is to minimize the sum of squared distances between data points and their respective cluster centroids. Despite its simplicity and efficiency, k-means is sensitive to the initial selection of centroids and may converge to local optima depending on the initial seeds.

Despite its simplicity, k-means is widely used in various applications, including customer segmentation, image compression, anomaly detection, and data preprocessing. It serves as a foundational clustering algorithm and has inspired many extensions and variations, such as k-means++, which improves the initial selection of centroids to achieve better clustering performance. Overall, the k-means algorithm is a versatile and effective tool for exploratory data analysis and pattern discovery in unlabeled datasets.

1.3. Runtime Environment – Jupyter notebook

Jupyter Notebook is an open-source web application that allows you to create and share documents containing live code, equations, visualizations, and narrative text. It supports over 40 programming languages, including Python, R, Julia, and Scala, making it a versatile tool for interactive computing and data analysis.

It is a versatile tool that facilitates interactive computing, data analysis, and collaborative work, making it popular among data scientists, researchers, educators, and developers. Its environment makes easy for developers to clearly understand what they are implementing as it follows a cellular approach. This approach also helps to implement project in a modular fashion, which in turn increases the readability of the code.

2 Literature Survey

One of the earlier work studies how we can obtain feature-level ratings of the mobile products from the customer reviews and review votes to influence decision making, both for new customers and manufacturers. Such a rating system gives a more comprehensive picture of the product than what a product-level rating system offers. While product-level ratings are too generic, feature-level ratings are particular; we exactly know what is good or bad about the product. Different customers are interested in different features. Thus, feature-level ratings can make buying decisions personalized [1].

Another work introduces VADER, a simple rule-based model for general sentiment analysis, and compare its effectiveness to eleven typical state-of-practice benchmarks including LIWC, ANEW, the General Inquirer, SentiWordNet, and machine learning oriented techniques relying on Naive Bayes, Maximum Entropy, and Support Vector Machine (SVM) algorithms. Using a combination of qualitative and quantitative methods, the study first construct and empirically validate a gold standard list of lexical features which are specifically attuned to sentiment in microblog-like contexts. It then combine these lexical features with consideration for five general rules that embody grammatical and syntactical conventions for expressing and emphasizing sentiment intensity [2].

Other work helped us in understanding sentiment analysis. This work presents present a customer feedback reviews on product, where we utilize opinion mining, text mining and sentiments, which has affected the surrounded world by changing their opinion on a specific product. Data used in this study are online product reviews collected from <http://Amazon.com>. They performed a comparative sentiment analysis of retrieved reviews. This research paper provides you with sentimental analysis of various smart phone opinions on smart phones dividing them Positive, Negative and Neutral Behavior [3].

There is another work that helped us in understanding current recommendation systems. This paper presents an overview of the field of recommender systems and describes the present generation of recommendation methods. Recommender systems or recommendation systems (RSs) are a subset of information filtering system and are software tools and techniques providing suggestions to the user according to their need. Many popular Ecommerce sites widely use RSs to recommend news, music, research articles, books, and product items. Recommendation systems use personal, implicit and local information from the Internet. This paper attempts to describe various limitations of recommendation methods and their advantages [4].

This literature survey helped us to better understand few existing works related to our project and gave insightful ideas to enhance the scope of our project. Based on these works, we identify gaps, and establish a foundation for the project's objectives and methodology. This involves identifying key terms, concepts, and variables relevant to the project's subject matter. Finally, it serves as the foundation for a project's research design and implementation.

3 Proposed System

The proposed system is an intelligent product recommendation platform that leverages customer feedback and sentiment analysis to deliver personalized product suggestions. The system comprises several key components designed to engage customers, analyze their preferences, and generate tailored recommendations:

3.1 Various modules in the project

The following modules collectively form the architecture of the project, enabling the implementation of an intelligent product review system that leverages customer feedback and sentiment analysis. Each module handles specific tasks and interactions to deliver desired results.

3.1.1 User Interface and Engagement

Through the proposed system, we design and implement an intuitive interface to collect Boolean - type responses from customers to assess their authenticity. Once the credibility assessment is success, the customer will be redirected to next interface to perform next tasks for feature-level analysis.

3.1.2 Feature Selection and Feedback Collection

Customers will be able to select desired product features from a curated list and then they can provide feedback for selected features through textual reviews or numerical ratings. Once customer provides his feedback the data is replicated into a new dataset for further analysis.

3.1.3 Data Processing and Sentiment Analysis

Robust data preprocessing techniques will be implemented to handle missing data and clean textual inputs (reviews) .Sentiment analysis using the VADER algorithm will be employed to analyze textual reviews and derive sentiment scores that reflect customer sentiment towards specific product features

3.1.4 Feature Importance Assessment and Categorization

Sentiment scores from reviews will be integrated with feature ratings to derive a combined metric that quantifies feature importance based on customer sentiment. Clustering algorithms such as K-means will categorize product features into groups (strong, moderate, weak) based on the combined metric, facilitating data-driven decision-making in recommendations.

3.1.5 Recommendation Generation

The system will generate recommendation utilizing categorized feature groups and customer preferences to generate personalized product recommendations. Decision rules will be implemented to determine recommendation outcomes (e.g., strong recommendation vs. weak recommendation) based on the balance and strength of categorized features selected by the customer.

3.2 Dataset

Initially to enable feature selection through customers, we have formed a dataset which simply contains various features of a product. Here, we have considered around 40 features of a mobile phone to be incorporated in the dataset. These are the most common features that can be observed.

Table 3.1 Initial Dataset

S.no.	Feature	S.no	Feature
1	charger	21	touch
2	camera	22	price
3	battery	23	sim
4	screen	24	permissions
5	apps	25	notifications
6	android	26	updates
7	services	27	software
8	size	28	multitasks
9	appearance	29	brightness
10	calls	30	music
11	sound	31	network
12	picture	32	buttons
13	warranty	33	features
14	ram	34	heat
15	design	35	waterproof
16	storage	36	video
17	speed	37	games
18	hardware	38	recordings
19	compatibility	39	gestures
20	sensors	40	gps

After module 2, we'll be provided with actual dataset that can be used for further analysis. The dataset formed consists of the selected features along with their corresponding reviews or ratings.

Table 3.2 Generated Dataset After Feedback Collection

Feature	Review	Rating
camera		4
battery	takes lot of time to charge. not recommended	
screen		5
services		3
sound	Good audio quality	
picture		5
ram	High gb ram at this cost	
design		5
storage		3
speed	High speed. Best for general use	
hardware	Good hardware is used may not get easily damaged.	
updates		4
software	Better software compatible for many applications	

3.3 Preprocessing

As the dataset comprises of missing data, it need to be handled. To handle the missing data, pandas libraries functions 'fillna' is used accordingly. Also to clean the textual reviews and to prepare them for sentiment analysis, the preprocessing is done with the help of nltk library in python. After preprocessing, the pre-processed reviews are augmented to the dataset.

3.4 Sentiment Analysis

Sentiment analysis can be described as the process of computationally identifying and categorizing opinions, emotions, and attitudes expressed in text data. It involves

analyzing text to determine the sentiment conveyed by the writer. The following section provides a brief study of sentiment analysis by covering its major aspects such as key components and applications.

3.4.1 Overview

Sentiment analysis, also known as opinion mining, is the process of extracting and analyzing sentiment or emotion from text data. The goal of sentiment analysis is to determine the overall sentiment expressed in a piece of text, whether it's positive, negative, or neutral. This analysis is valuable for understanding public opinion, customer feedback, and social media sentiment.

3.4.2 Key Components of Sentiment Analysis

In sentiment analysis, various techniques from natural language processing (NLP) and machine learning are applied to understand the sentiment expressed in textual content, such as product reviews, social media posts, customer feedback, and news articles. The goal of sentiment analysis is to automatically extract subjective information from text and quantify the sentiment polarity (e.g., positive, negative, or neutral) associated with it. The following points help us to understand the key components of sentiment analysis.

A. Text Preprocessing:

Before sentiment analysis, text data undergoes preprocessing steps such as tokenization (breaking text into words or phrases), removing stop words (commonly used words that do not contribute much meaning), and stemming (reducing words to their base or root form).

B. Sentiment Lexicons and Dictionaries:

Sentiment analysis often relies on lexicons or dictionaries containing words annotated with sentiment scores (e.g., positive, negative, neutral). These lexicons assign polarity values to words based on their sentiment. These polarity values help us to extract the sentiment associated with the text (e.g., positive, negative, or neutral).

C. Machine Learning Models:

Machine learning techniques, such as supervised learning (classification) and unsupervised learning (clustering), can be employed for sentiment analysis. Supervised learning uses labeled data to train models to predict sentiment, while unsupervised learning identifies patterns and clusters in text data.

D. Rule-based Approaches:

Rule-based approaches in sentiment analysis refer to methods that rely on predefined rules, patterns, or heuristics to determine the sentiment expressed in text. Unlike machine learning-based approaches that learn sentiment patterns from data, rule-based methods use explicit rules crafted by experts to analyze sentiment. These approaches may include grammatical rules, linguistic patterns, and syntactic analysis.

3.4.3 Applications of Sentiment Analysis

Sentiment analysis is a valuable tool for businesses and organizations to gain insights into customer opinions, market trends, and public perception. By analyzing sentiment in textual data, companies can make data-driven decisions, improve customer satisfaction, monitor brand reputation, and identify emerging issues or sentiments in real-time. Few of its real-time applications are:

A. Social Media Monitoring:

Sentiment analysis is widely used to analyze public opinion and sentiment expressed on social media platforms. It helps businesses understand customer perceptions and brand sentiment. We can observe this today in popular applications such as twitter, facebook to differentiate positive and negative public opinions.

B. Customer Feedback Analysis:

Sentiment analysis is used to analyze customer reviews, surveys, and feedback to gauge customer satisfaction and identify areas for improvement [5]. It can be observed today in most of the e-commerce websites like amazon, flipkart etc., where customer feedback is considered crucial for running businesses.

C. Market Research:

Sentiment analysis provides insights into market trends, consumer preferences, and sentiment towards products or services, aiding in market research and decision-making. As it has its applications across various platforms, sentiment analysis can help in understanding market trends clearly.

3.5 VADER algorithm for sentiment analysis

VADER (Valence Aware Dictionary and sEntiment Reasoner) is a sentiment analysis tool that uses Natural Language Processing (NLP) to identify the sentiment of text. VADER assigns sentiment scores to text based on the weighted sum of intensity scores of individual words and modifiers in the lexicon. This section provides a brief study of sentiment analysis by VADER.

3.5.1 Overview

The VADER algorithm is a lexicon and rule-based approach designed specifically for sentiment analysis of text data. It was developed by C.J. Hutto and Eric Gilbert, VADER is widely used for its simplicity and effectiveness in capturing sentiment nuances in natural language.

3.5.2 Key components of VADER

VADER is implemented as part of the NLTK (Natural Language Toolkit) library in Python, making it accessible and easy to use for text analysis tasks. It has gained popularity in various applications, including social media monitoring, customer feedback analysis, brand sentiment analysis, and opinion mining. The following points helps us to understand the key components of sentiment analysis through VADER.

A. Lexicon-based Approach:

VADER utilizes a sentiment lexicon that contains words annotated with sentiment scores (positive, negative, neutral). Each word in the lexicon is assigned a polarity value based on its sentiment intensity, ranging from -4 (extremely negative) to +4 (extremely positive), with 0 representing neutrality.

B. Rule-based Sentiment Analysis:

In addition to sentiment scores, VADER incorporates grammatical rules and linguistic features to analyze sentiment in text. It considers punctuation marks, capitalization, degree modifiers (e.g., "very", "extremely"), and emoticons to infer sentiment intensity. These rules are crafted by experts to analyze sentiment.

C. Handling Context and Negation:

As VADER has the ability to handle sentiment analysis in a context-aware manner, it is equipped to handle contextual nuances and negation in text. It can recognize changes in sentiment due to modifiers or qualifiers (e.g., "not good") and adjust sentiment scores accordingly, capturing the sentiment shift accurately.

D. Scalability and Real-time Analysis:

One of the key advantages of VADER is its computational efficiency and scalability. It can perform sentiment analysis in real-time on large volumes of text data, making it suitable for applications such as social media monitoring and customer feedback analysis.

3.5.3 VADER Architecture

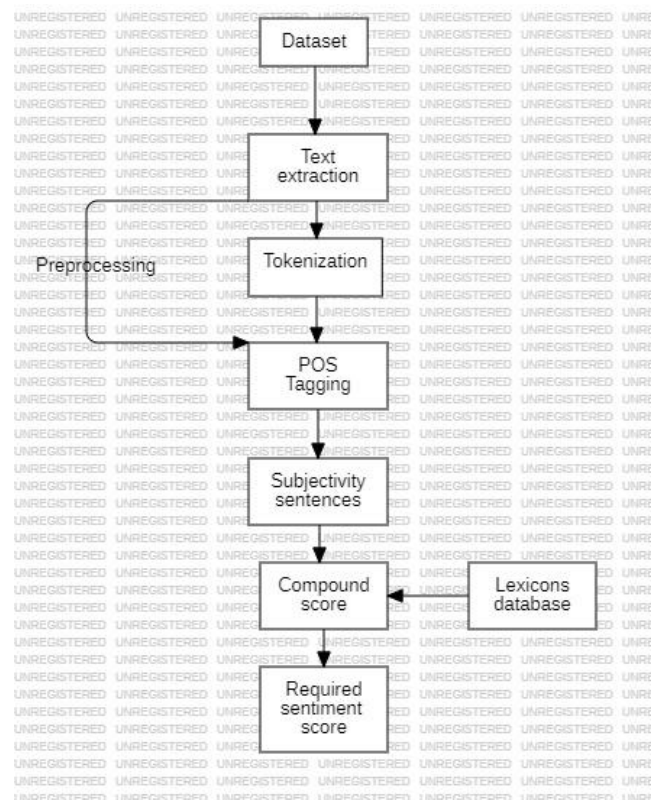


Figure 3.1 VADER Algorithm Workflow

The architecture of VADER reflects a hybrid approach to sentiment analysis, leveraging both lexical knowledge (via sentiment lexicon) and linguistic rules to capture sentiment nuances in text data. The combination of lexicon-based scoring and rule-based heuristics enables VADER to achieve robust sentiment analysis performance across various types of textual content, making it a valuable tool for applications such as social media monitoring, brand sentiment analysis, and opinion mining in NLP tasks.

3.6 K-means clustering

Clustering is the task of dividing the unlabeled data or data points into different clusters such that similar data points fall in the same cluster than those which differ from the others. The following section gives a brief study of one such clustering algorithm kmeans.

3.6.1 Overview

The K-means clustering algorithm is a popular unsupervised machine learning technique used for partitioning the data into K distinct, non-overlapping clusters based on similarity of data points. The algorithm aims to minimize the within-cluster sum of squared distances from each data point to its assigned cluster centroid. It is widely applied in various domains for data segmentation, pattern recognition, and data mining tasks.

3.6.2 Working of Kmeans algorithm

The k-means algorithm aims to optimize the objective function known as the within-cluster sum of squares (WCSS), where the goal is to minimize the sum of squared distances between data points and their respective cluster centroids. It serves as a

foundational clustering algorithm and has inspired many extensions and variations, such as k-means++, which improves the initial selection of centroids to achieve better clustering performance. The working of kmeans can be understood by the following points

A. Initialization:

Choose the number of clusters (K) that the dataset should be divided into. Initialize K cluster centroids either randomly or based on predefined criteria (e.g., randomly selecting K data points as initial centroids). These cluster centroids are typically referred to as seeds. k-means is sensitive to the initial selection of these centroids and may converge to local optima depending on the initial seeds.

B. Assignment of Data Points to Clusters:

Iterate through each data point in the dataset. For each data point, calculate the distance to each of the K cluster centroids using a distance metric such as Euclidean distance. Assign the data point to the cluster whose centroid is closest (i.e., has the minimum distance).

C. Update Cluster Centroids:

After assigning all data points to clusters, recalculate the centroids of the clusters based on the mean (average) of the data points assigned to each cluster. Move each cluster centroid to the new mean location calculated from the data points assigned to its cluster. This updation takes place till the cluster assignments no longer change significantly between iterations.

D. Iterative Refinement:

Repeat the assignment of data points to clusters and centroid updates iteratively until convergence criteria are met. Convergence is typically achieved when the cluster

assignments and centroids stabilize (i.e., minimal change in cluster assignments between iterations).

3.6.3 Kmeans Architecture

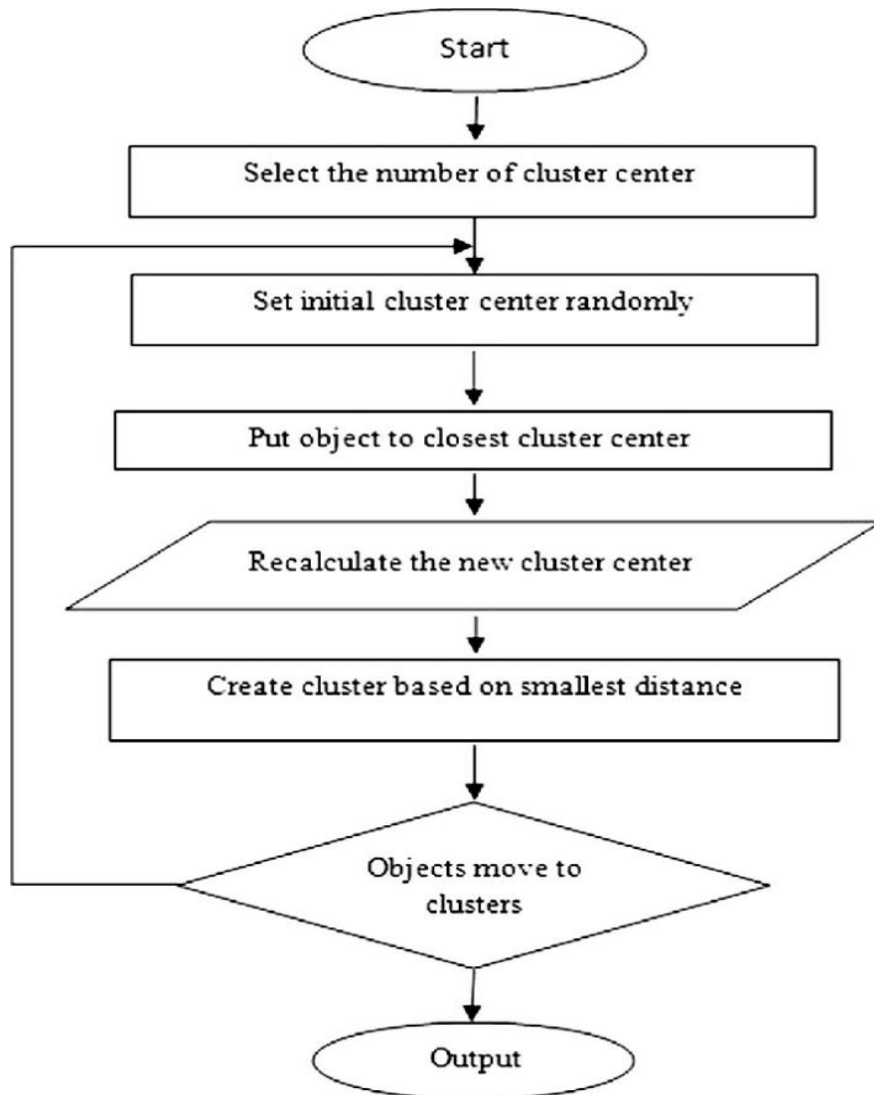


Figure 3.2 KMeans Architecture

The architecture of the k-means algorithm revolves around the iterative process of assigning data points to clusters based on proximity to cluster centroids and updating centroids based on the mean of assigned data points. This iterative approach allows k-means to partition data into clusters efficiently, making it a widely used clustering

technique in various applications, including customer segmentation, image segmentation, and data preprocessing in machine learning workflows.

3.7 Proposed architecture

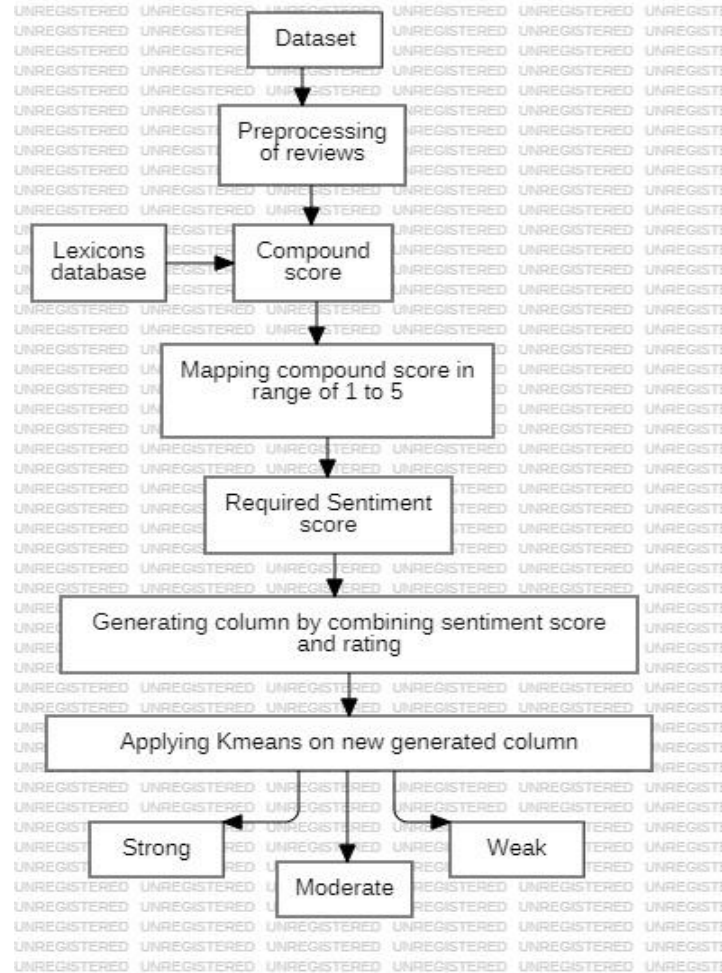


Figure 3.3 Proposed Architecture

The proposed architecture integrates VADER sentiment analysis and K-means clustering to develop specified model for feature-level analysis that leverages customer feedback and sentiment that aids in personalized product recommendations. By combining VADER sentiment analysis with K-means clustering, the proposed architecture enables the extraction of valuable insights from textual data and translates them into effectful reviews to aid recommendations. This approach leverages the power

of NLP techniques and unsupervised learning to deliver personalized experiences, enhance user satisfaction, and drive business outcomes in recommendation systems.

3.8 Advantages of Proposed system

The proposed system offers significant advantages by leveraging VADER sentiment analysis and K-means clustering to enhance the effectiveness and personalization of product recommendations through feature-level analysis. By analyzing customer feedback and sentiments expressed in textual data, the system can deliver highly tailored recommendations that align with individual preferences and sentiments. This personalized approach not only improves user engagement and satisfaction but also drives higher conversion rates and customer loyalty. Here are few advantages of the system.

A. Personalized Recommendations:

The project aims to deliver personalized product recommendations based on customer preferences and sentiment analysis. This personalized approach can enhance user satisfaction and increase engagement by offering tailored suggestions aligned with individual preferences.

B. Improved User Experience:

By incorporating customer feedback and sentiment analysis, the project can enhance the overall user experience. Customers will receive product recommendations that reflect their sentiments and preferences, leading to more meaningful interactions with the platform.

C. Effective Handling of Heterogeneous Data:

The project implements robust data preprocessing techniques to handle heterogeneous data types such as reviews and ratings. This ensures data quality and reliability in the

sentiment analysis process, leading to more accurate insights and recommendations. Here we dealt with heterogeneous data with help of pandas library.

D. Real-time Recommendation Updates:

Leveraging efficient algorithms like VADER for sentiment analysis and K-means for clustering allows the project to perform real-time recommendation updates. This agility enables the system to adapt quickly to changing customer sentiments and preferences aiding in personalized recommendations.

E. Data-driven Decision Making:

By categorizing product features based on sentiment and feedback, the project facilitates data-driven decision making in recommendation outcomes. This approach ensures that recommendations are not only personalized but also aligned with the collective sentiment of customers.

F. Scalability and Adaptability:

The project's architecture, leveraging machine learning techniques and efficient algorithms, supports scalability and adaptability. As the user base grows or evolves, the recommendation system can efficiently handle larger datasets and adapt to emerging trends and patterns.

G. Enhanced Business Insights:

The project generates valuable insights into customer sentiments, product preferences, and market trends through sentiment analysis. These insights can inform business strategies, marketing campaigns, and product development efforts, ultimately leading to better business outcomes.

4 Design

System modeling using UML (Unified Modeling Language) diagrams plays a crucial role in project design by providing a visual representation of the system's structure, behavior, and interactions. UML diagrams are used to capture and communicate the requirements, design decisions, and architecture of a software system in a standardized and systematic manner. By leveraging UML diagrams in project design, software development teams can improve clarity, consistency, and efficiency in the development process, leading to better-designed and more maintainable software systems.

One key aspect of system modeling with UML diagrams is the creation of different types of diagrams to depict various aspects of the system. For example, structural diagrams like class diagrams illustrate the static structure of the system by showing classes, attributes, methods, and relationships among objects. Class diagrams help in understanding the organization of the system's components and their relationships.

Overall, system modeling using UML diagrams facilitates communication and collaboration among stakeholders, developers, and designers by providing a common visual language to describe the system's architecture, functionality, and behavior. UML diagrams serve as blueprints for software development, enabling teams to identify requirements, design components, and validate system behavior before implementation.

4.1 Usecase diagram

Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors. Actors are the external entities that interact with the system. The use cases are represented by either circles or ellipses. The Figure 4.1 shows the use case representation of the system.

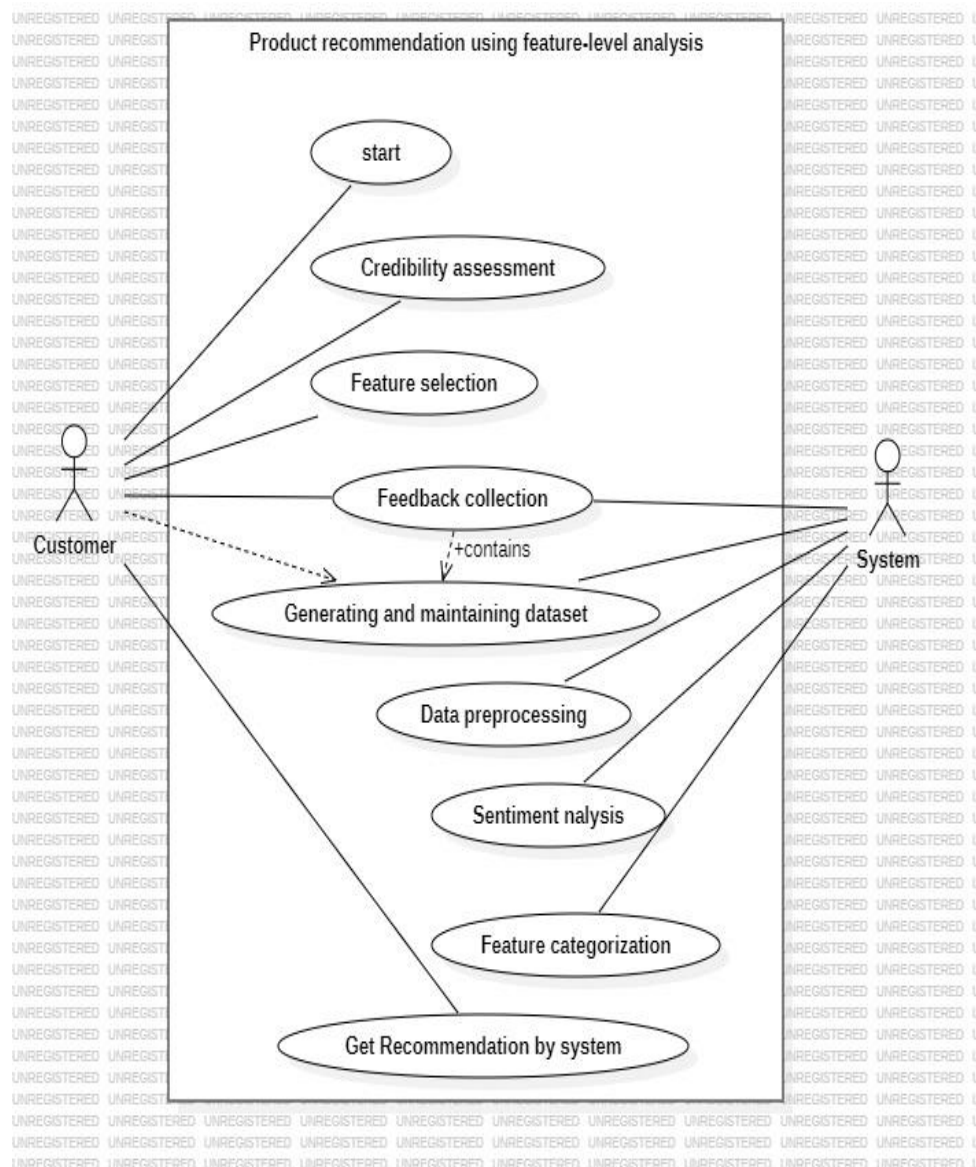


Figure 4.1 Usecase Diagram

4.2 Class diagram

Class diagrams give an overview of a system by showing its classes and the relationships among them. Class diagrams are static – they display what interacts but not what happens when they do interact. In general a class diagram consists of some set of attributes and operations. Operations will be performed on the data values of attributes. The Figure 4.2 shows the class diagram representation of the system.

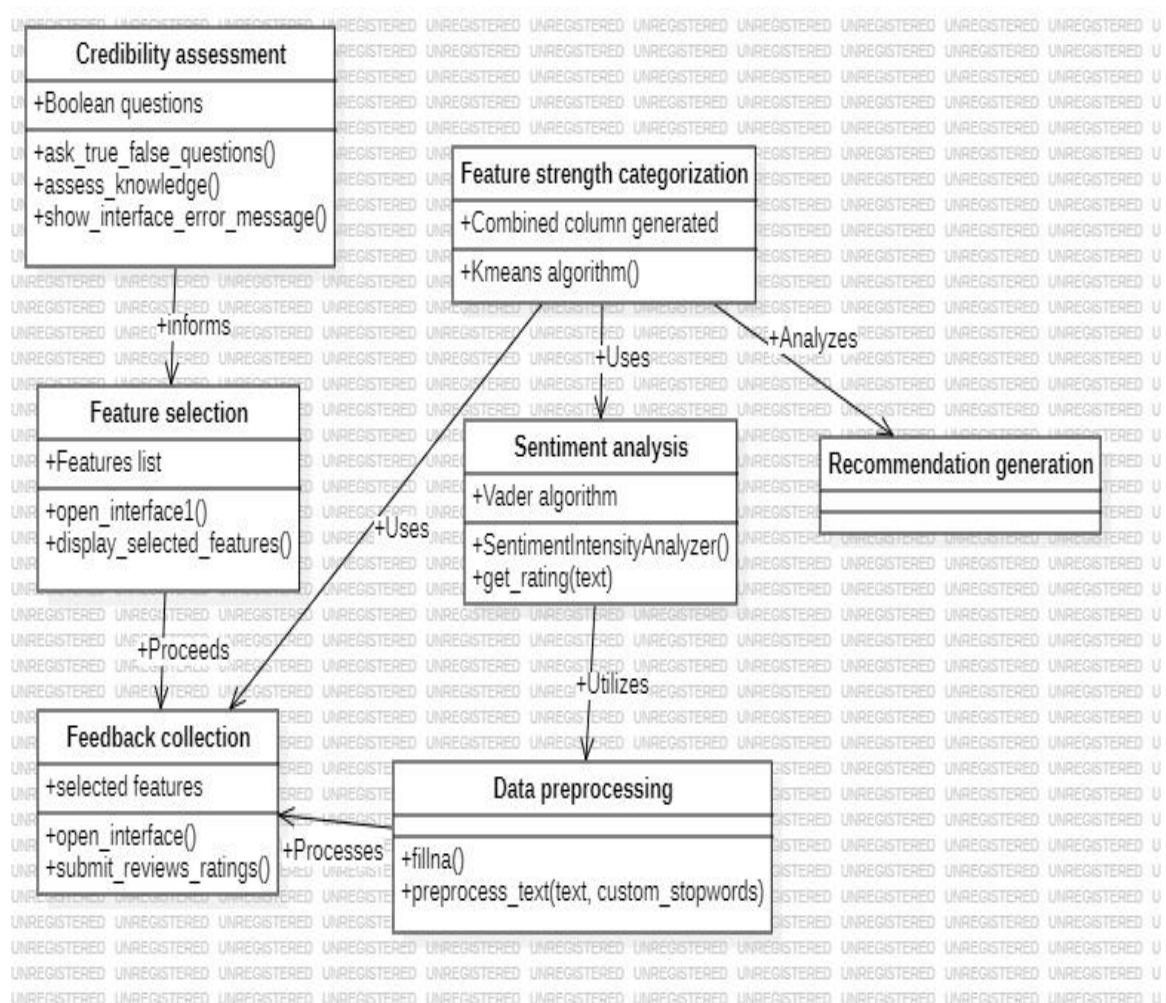


Figure 4.2 Class Diagram

4.3 Activity diagram

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. In UML, an activity diagram provides a view of the behavior of a system by describing the sequence of actions in a process. The Figure 4.3 shows the activity diagram representation of the system.

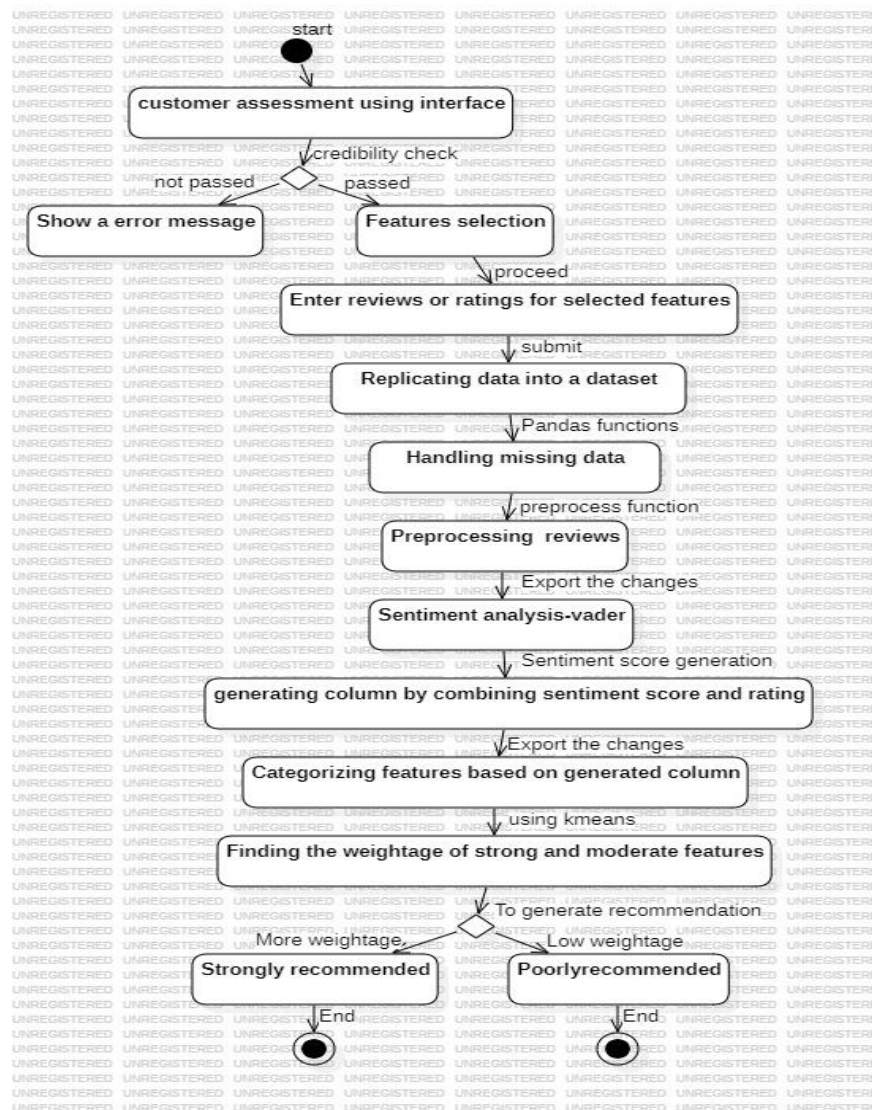


Figure 4.3 Activity Diagram

4.4 Statechart diagram

A state diagram, also known as a state machine diagram or state chart diagram, is an illustration of the states an object can attain as well as the transitions between those states in the Unified Modeling Language (UML). The Figure 4.4 shows the state chart diagram representation of the system.

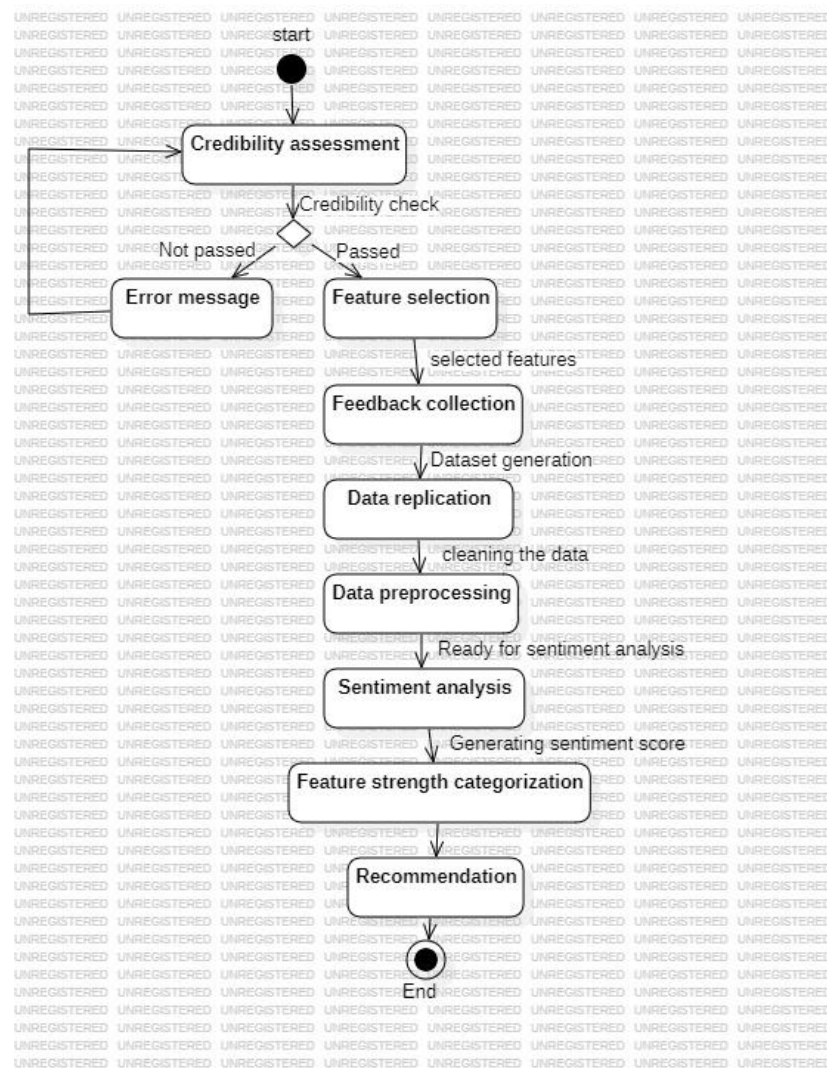


Figure 4.4 Statechart Diagram

4.5 Sequence diagram

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. The Figure 4.5 shows the sequence diagram representation of the system.

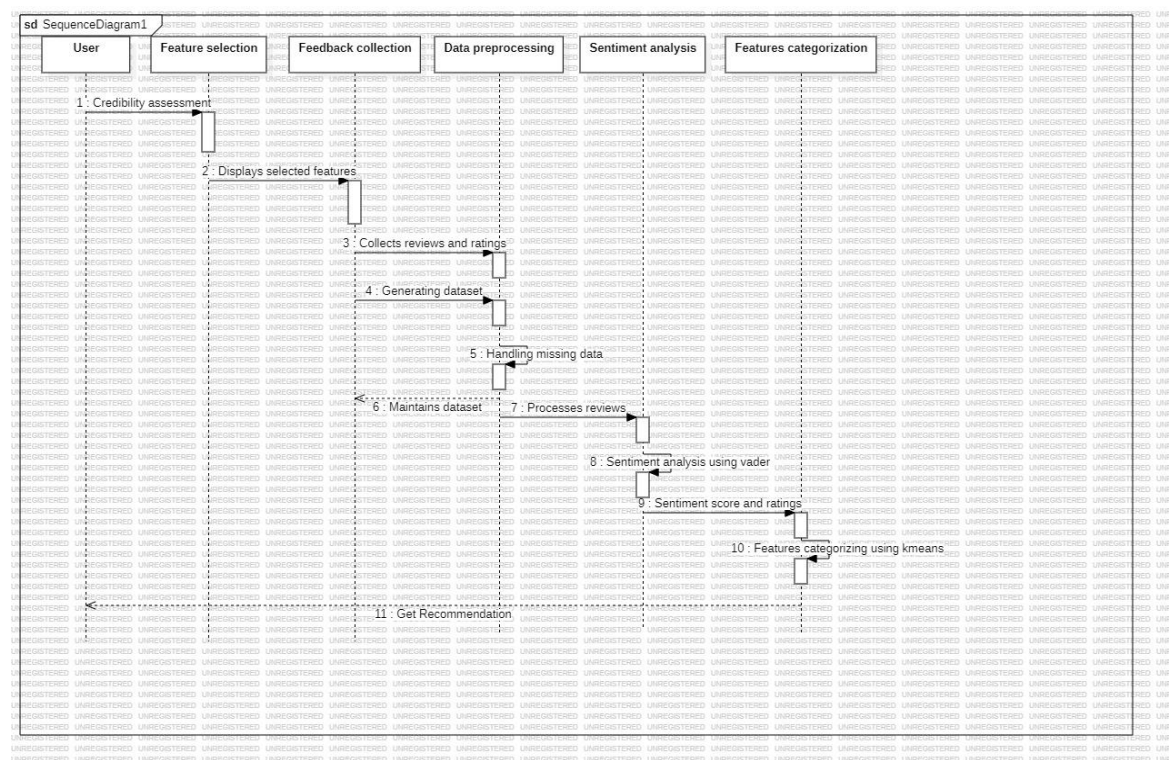


Figure 4.5 Sequence Diagram

5 Implementation

Implementing a project involves translating the design specifications and requirements into functional software or system components. The implementation phase is where developers write code, integrate modules, and build the system according to the outlined design and architecture. This involves writing clean, modular, and well-documented code that adheres to the project's coding conventions and guidelines. The proposed system can be implemented in a modular approach as follows:

5.1 Credibility assessment

In this module, primarily we aim to check the credibility of a user through a simple interface which collects Boolean-type responses from users. To implement this module, we have used tkinter library which is the standard GUI (Graphical User Interface) toolkit for Python. It provides a set of built-in widgets and functions for creating desktop applications with graphical interfaces.

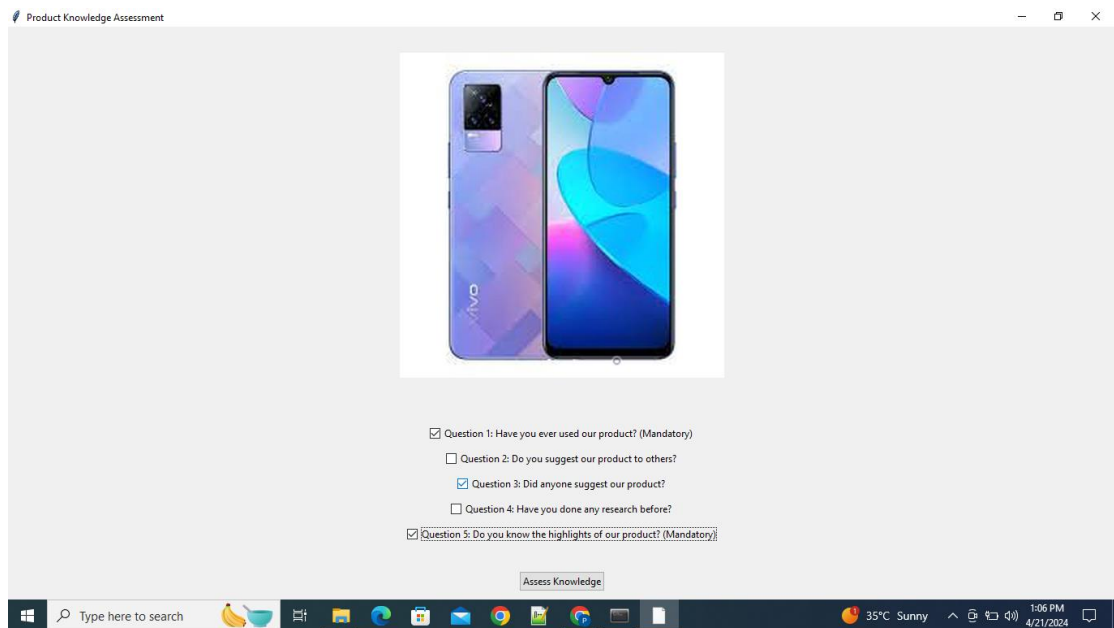
- a) The interface contains few Boolean-type questions to assess credibility of a customer. Among these, few need to be answered mandatorily such that the customer can meet the threshold and can redirect to the next interface.
- b) If customer does not pass the assessment, an error message is displayed.
- c) In our code, the implementation of this module can be observed through these methods:

`ask_true_false_questions()`


`assess_knowledge()`

`show_interface_error_message(message)`

The following images shows the results of this module:



Product Knowledge Assessment



☒ Question 1: Have you ever used our product? (Mandatory)

☐ Question 2: Do you suggest our product to others?

☒ Question 3: Did anyone suggest our product?

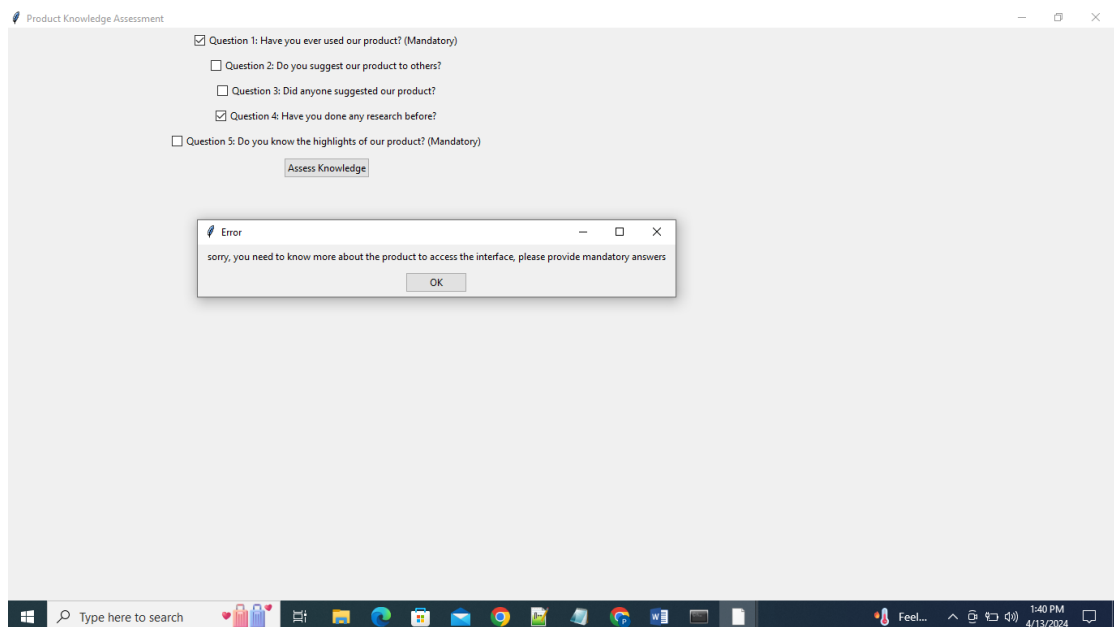
☐ Question 4: Have you done any research before?

☒ Question 5: Do you know the highlights of our product? (Mandatory)

Assess Knowledge

Figure 5.1 Credibility assessment

If customer doesnot meet the threshold, output is as follows:



Product Knowledge Assessment

☒ Question 1: Have you ever used our product? (Mandatory)

☐ Question 2: Do you suggest our product to others?

☐ Question 3: Did anyone suggested our product?

☒ Question 4: Have you done any research before?

☐ Question 5: Do you know the highlights of our product? (Mandatory)

Assess Knowledge

Error
sorry, you need to know more about the product to access the interface, please provide mandatory answers
OK

Figure 5.2 Error

5.2 Feature selection

After credibility assessment, the customer is redirected to the next screen which contains a curated list of features extracted from a sample dataset. Here, we have considered around 40 features of a mobile. Through this interface, the customer can select the features for which he want to give feedback in form of either reviews or ratings. To implement this module, we have used tkinter library.

- a) The input for feature selection is list of features and the output is the screen to provide feedback.
- b) If customer does not select features, it will display a popup message to select at least one feature.
- c) After selection of features the customers will be redirected to a screen where only selected features are provided to enter feedback.
- d) Thus, this module provides a interface which gives importance to the opinion of customers through selection.
- e) In our code, the implementation of feature selection can be observed through these methods:

`open_interface()`

`display_selected_features()`

The following image shows the results of this module:

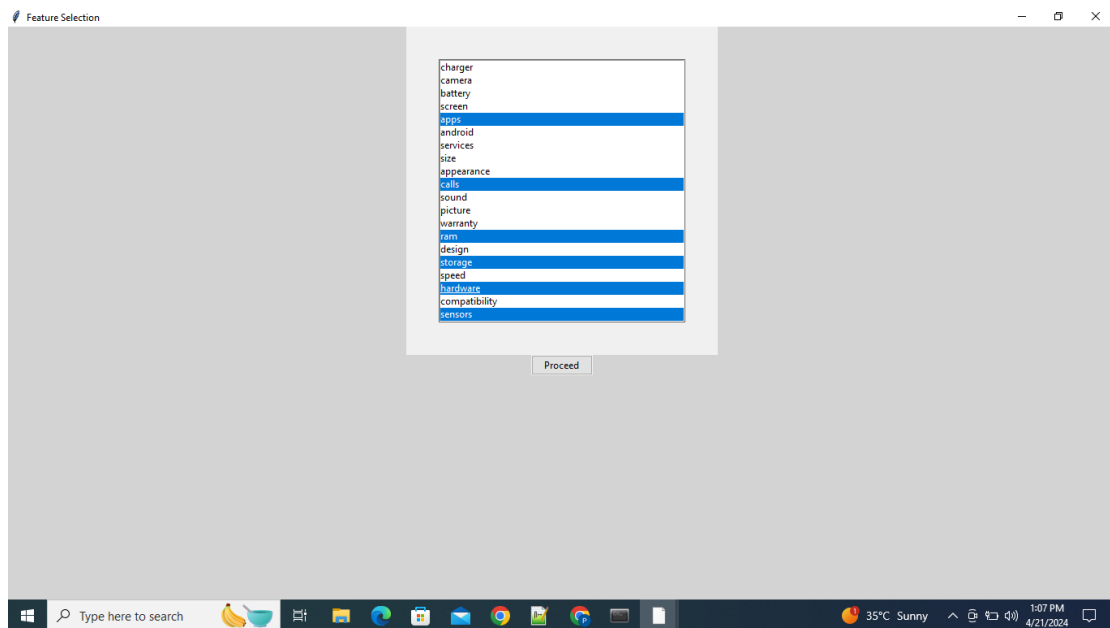


Figure 5.3 Feature Selection

5.3 Feedback collection and Dataset generation

After feature selection, the user can provide his feedback through either reviews or ratings. The interface where customer provides their feedback, consists of selected features. After giving the feedback, the customer can click on submit, after which a dataset is generated through the collected data which helps in further analysis.

- a) The dataset generated thus consists of selected features along with their corresponding reviews and ratings. To implement this module, we have used tkinter library.
- b) In our code, the implementation of feedback collection can be observed through these methods:

`open_interface()`

`submit_reviews_ratings()`

The following images shows the results of this module:

Feature	Review	Rating
camera		4
battery	e. Not recommended	
services		3
sound	Good audio quality.	
picture		5
ram	gh gb ram at this cost.	
design	ign similar to i-phone.	
storage		3

Submit Reviews and Ratings

Figure 5.4 Feedback Collection

Out[14]:

	Feature	Review	Rating
0	camera		4
1	battery	takes lot of time to change. not recommended	
2	screen		5
3	services		3
4	sound	Good audio quality/n	
5	picture		5
6	ram	High gb ram at this cost	
7	design		5
8	storage		3
9	speed	High speed.Best for general use	
10	hardware	Good hardware is used may not get easily damaged.	
11	updates		4
12	software	Better software compatible for many applications	

In [15]: reviews_ratings_df.to_excel("C:\\Users\\User\\Desktop\\reviews_ratings_df.xlsx",index=False)

Figure 5.5 Generated Dataset

5.4 Data preprocessing

As the dataset generated contains missing data it need to be handled, and also the textual reviews need to undergo preprocessing before we continue further with sentiment analysis. Hence, in this module, the dataset generated through feedback collection is

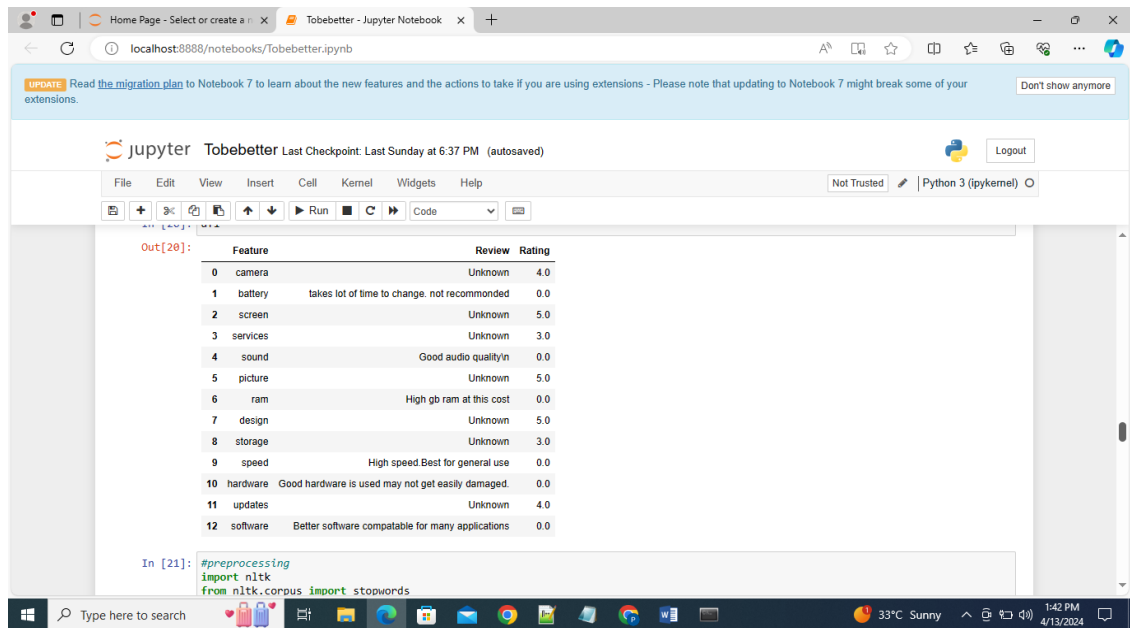
considered as input and on which pandas library functions are applied to handle missing data.

a) The reviews column is considered as input for processing textual reviews which uses nltk library.

b) In our code, the implementation of processing textual reviews, can be observed through this method:

```
preprocess_text(text,custom_stopwords)
```

The result after handling missing data is as follows:



	Feature	Review	Rating
0	camera	Unknown	4.0
1	battery	takes lot of time to change. not recommended	0.0
2	screen	Unknown	5.0
3	services	Unknown	3.0
4	sound	Good audio quality/n	0.0
5	picture	Unknown	5.0
6	ram	High gb ram at this cost	0.0
7	design	Unknown	5.0
8	storage	Unknown	3.0
9	speed	High speed Best for general use	0.0
10	hardware	Good hardware is used may not get easily damaged.	0.0
11	updates	Unknown	4.0
12	software	Better software compatible for many applications	0.0

```
In [21]: #preprocessing
import nltk
from nltk.corpus import stopwords
```

Figure 5.6 Handling Missing Data

The result after processing reviews is as follows:

UPDATE Read the [migration plan](#) to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions. [Don't show anymore](#)

Jupyter Tobebetter Last Checkpoint: Yesterday at 1:46 PM (unsaved changes) [Logout](#)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Out[27]:

	Feature	Review	Rating	preprocessed_review
0	camera	Unknown	4.0	unknown
1	battery	Takes lot of time to charge.. Not recommended	0.0	takes lot time charge not recommended
2	screen	Unknown	5.0	unknown
3	services	Unknown	3.0	unknown
4	sound	Good audio quality.	0.0	good audio quality
5	picture	Unknown	5.0	unknown
6	ram	High gb ram at this cost.	0.0	high gb ram cost
7	design	Unknown	5.0	unknown
8	storage	Unknown	3.0	unknown
9	speed	High speed. Best fir general use	0.0	high speed best fir general use
10	hardware	Good hardware is used may not get easily damaged.	0.0	good hardware used may not get easily damaged
11	updates	Unknown	4.0	unknown
12	software	Better software compatible for many applications.	0.0	better software compatible many applications

In [27]: `from nltk.sentiment import SentimentIntensityAnalyzer`

Figure 5.7 Preprocessed Dataset

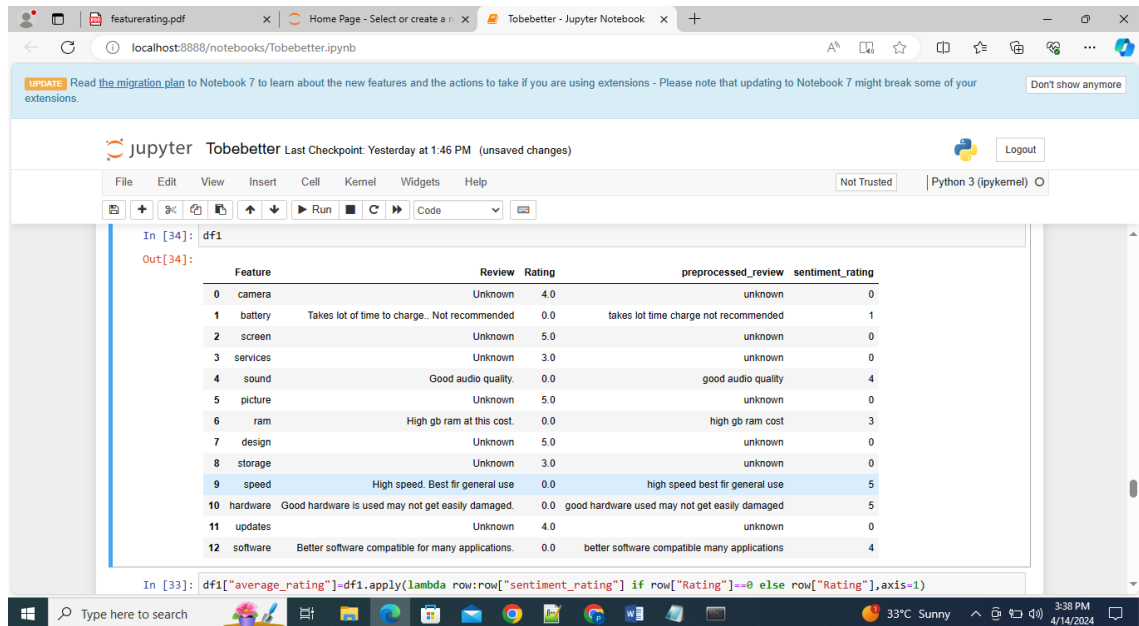
5.5 Sentiment analysis

The key module of the proposed model is sentiment analysis. It helps in better feature-level analysis. Here, we are using VADER algorithm to generate sentiment scores for reviews in the range of 1 to 5. The preprocessed reviews are considered as input and the output is the generated sentiment score.

- a) In our code, the implementation of sentiment analysis to generate sentiment score can be observed through the method:

`generate_rating()`

The result after sentiment analysis is as follows:



UPDATE Read the [migration plan](#) to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions. [Don't show anymore](#)

jupyter Tobebetter Last Checkpoint: Yesterday at 1:46 PM (unsaved changes) [Logout](#)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

In [34]: df1

Out[34]:

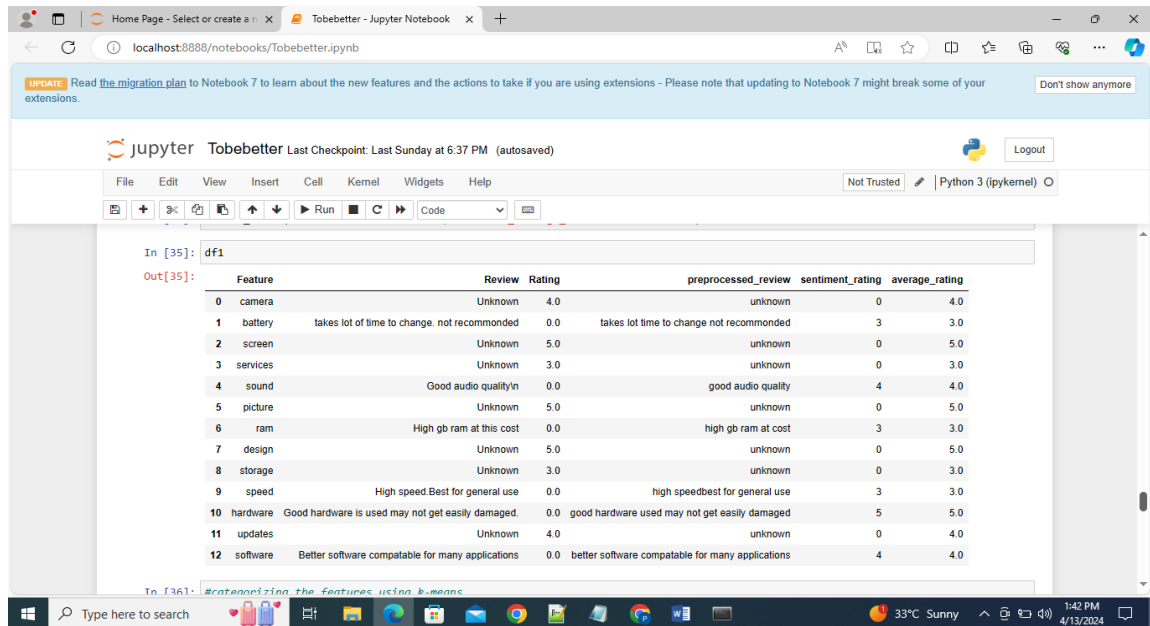
	Feature	Review	Rating	preprocessed_review	sentiment_rating
0	camera	Unknown	4.0	unknown	0
1	battery	Takes lot of time to charge.. Not recommended	0.0	takes lot time charge not recommended	1
2	screen	Unknown	5.0	unknown	0
3	services	Unknown	3.0	unknown	0
4	sound	Good audio quality.	0.0	good audio quality	4
5	picture	Unknown	5.0	unknown	0
6	ram	High gb ram at this cost.	0.0	high gb ram cost	3
7	design	Unknown	5.0	unknown	0
8	storage	Unknown	3.0	unknown	0
9	speed	High speed. Best fir general use	0.0	high speed best fir general use	5
10	hardware	Good hardware is used may not get easily damaged.	0.0	good hardware used may not get easily damaged	5
11	updates	Unknown	4.0	unknown	0
12	software	Better software compatible for many applications.	0.0	better software compatible many applications	4

In [33]: df1["average_rating"]=df1.apply(lambda row:row["sentiment_rating"] if row["Rating"]==0 else row["Rating"],axis=1)

Figure 5.8 Generating Sentiment Score

- b) After generating sentiment score, these scores from reviews will be integrated with feature ratings to derive a combined metric, which is considered as final rating of the corresponding feature.
- c) This metric is derived in such a way that wherever the reviews are unknown the sentiment score resulted is zero and wherever the ratings are not provided it is filled with zero priorly.
- d) Now, with the help of sentiment scores and prior ratings, wherever the sentiment score is zero, feature rating is considered and wherever the ratings are zero, sentiment scores are considered. In tis way, finally we derived at a metric which represents the final rating of features.

The dataset after generating final ratings for features looks as follows:



UPDATE Read the [migration plan](#) to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions. Don't show anymore

Jupyter Tobebetter Last Checkpoint: Last Sunday at 6:37 PM (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (pykernel)

In [35]: df1

Out[35]:

	Feature	Review	Rating	preprocessed_review	sentiment_rating	average_rating
0	camera	Unknown	4.0	unknown	0	4.0
1	battery	takes lot of time to change. not recommended	0.0	takes lot time to change not recommended	3	3.0
2	screen	Unknown	5.0	unknown	0	5.0
3	services	Unknown	3.0	unknown	0	3.0
4	sound	Good audio quality	0.0	good audio quality	4	4.0
5	picture	Unknown	5.0	unknown	0	5.0
6	ram	High gb ram at this cost	0.0	high gb ram at cost	3	3.0
7	design	Unknown	5.0	unknown	0	5.0
8	storage	Unknown	3.0	unknown	0	3.0
9	speed	High speed Best for general use	0.0	high speedbest for general use	3	3.0
10	hardware	Good hardware is used may not get easily damaged.	0.0	good hardware used may not get easily damaged	5	5.0
11	updates	Unknown	4.0	unknown	0	4.0
12	software	Better software compatible for many applications	0.0	better software compatible for many applications	4	4.0

In [36]: #categorizing the features using k-means

Figure 5.9 Generating Final Feature Rating

5.6 Feature strength categorization

In this module, with the help of generated final rating column, features are categorized as strong, moderate and weak through kmeans clustering algorithm. The categorization of features aims at signifying strength of features which assists in feature-level analysis.

- The features falls under different categories which is decided on final rating metric of features.
- This categorization further helps in recommendation generation and provides a more meaningful feature-level analysis.

The result of this module is as follows:

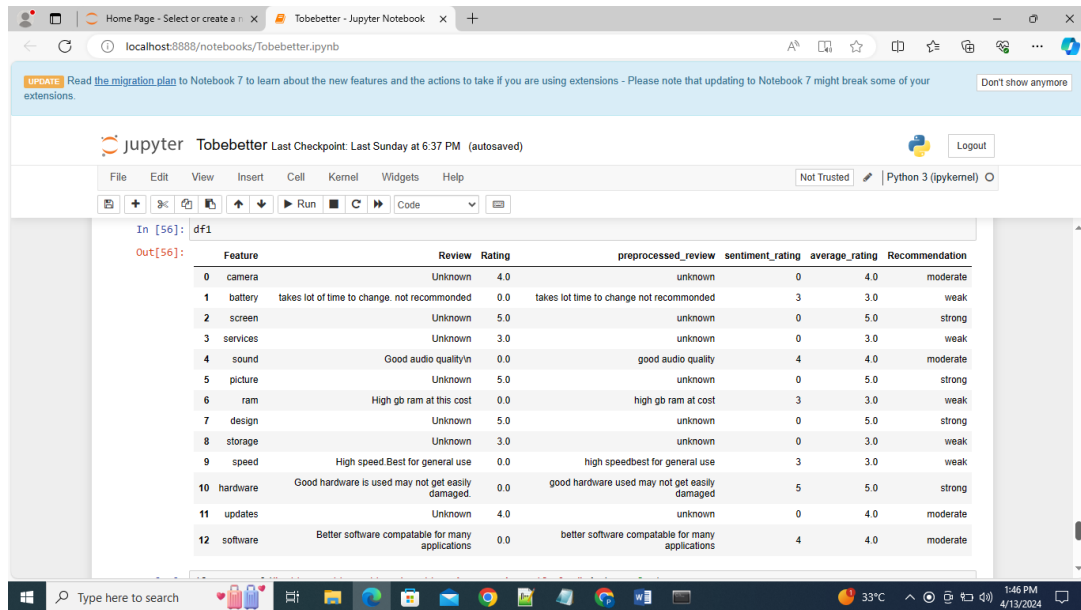


Figure 5.10 Feature Strength Categorization

5.7 Recommendation generation

In this module, with the help of feature categories, the recommendation is generated as strong or weak. Decision rules will be implemented to determine recommendation outcomes (e.g., strong recommendation vs. weak recommendation) based on the balance and strength of categorized features selected by the customer.

- The recommendation is generated automatically as output after the customer submits his feedback in feedback collection module. All the other processes are called internally, once after submitting feedback.
- This model primarily not aims at generating recommendations, but our main aim is to aid the recommendation process through our proposed model.

The result of this module is as follows:

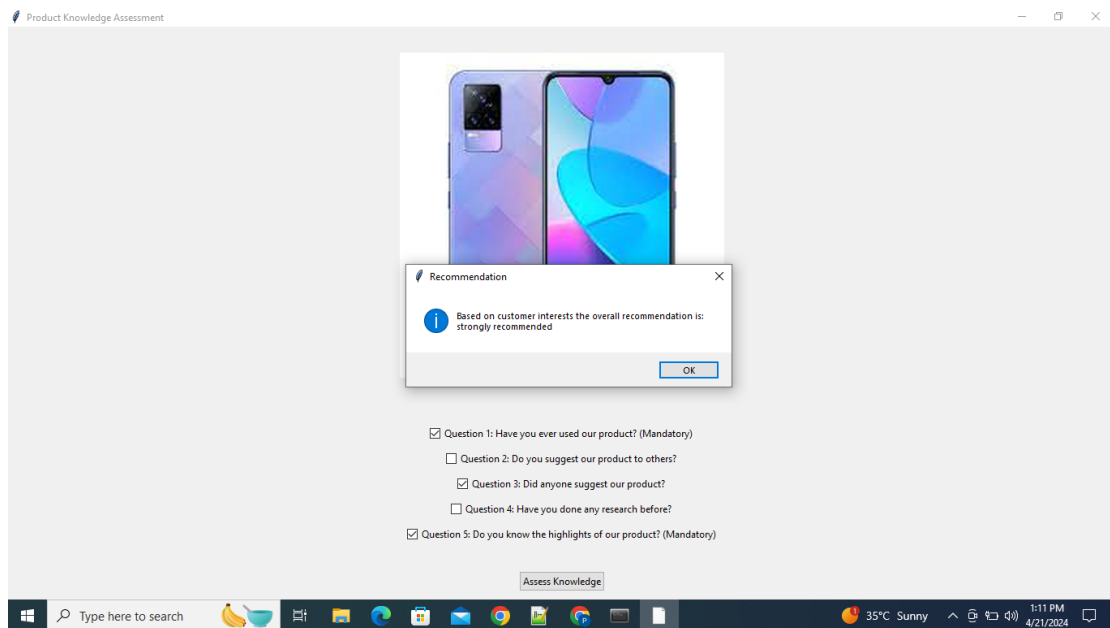


Figure 5.11 Recommendation Generation

Though the proposed model is automated, the dataset images are provided for clean understanding of the various processes involved in the model. Finally, Through the implementation of model and by understanding the results, we can say that the proposed system serves as a part of recommendation systems, though it does not completely implement a recommendation system, it aids in the processes of generating accurate recommendations through feature-level analysis. It can be merged with the actual existing recommendation systems to generate more personalized product suggestions which in turn helps businesses by providing actionable insights such as identifying popular product features, understanding customer sentiment trends, and optimizing product recommendations. The repository link regarding the implementation of project code is as follows: <https://github.com/budatipranavi/Team-C3.git>

6 Conclusions and Future Work

In today's world, we can observe most of the businesses are running through their own websites. So, it is necessary that consumer opinion need to be considered and their requirements are met. This system provides a deep feature-level analysis by incorporating customer feedback and sentiment analysis. It combines credibility assessment, feature selection, feedback collection, sentiment analysis and feature strength categorization which solely relies on features of the product.

The system results in personalized product suggestions that resonate with individual users. This personalized approach enhances user engagement and satisfaction with the platform. It keeps both the manufacturer and the customer well-informed in the decisions to make in improving the product and buying, respectively. Different customers are interested in different features. Thus, feature-level ratings can make buying decisions personalized. The system can find its applications in recommendation systems, customer research etc..

6.1 Future enhancements

Considering future enhancements related to a project involves envisioning potential improvements, features, or expansions that can be implemented to enhance the project's functionality, usability, and value over time. By embracing a forward-thinking approach to project development, organizations can leverage future enhancements to drive value, innovation, and sustainable success. Similarly, there are several avenues for further development and enhancement of this system:

A. Strict credibility assessment

These days, fake reviews are thriving everywhere. This causes hindrance to a product popularity which in turn decreases the sales and adversely affects businesses. Hence, it is necessary to know the credibility of a customer. Rather than simply evaluating credibility using a method, the system can incorporate any cryptographic algorithm through which false reviews can be prevented.

B. Advanced Machine Learning Models

For sentiment analysis, one can opt different kind of algorithms like LSTM, BERT etc.. and one can explore different probabilities of results. Not only for sentiment analysis, but also for other modules, various tools need to be verified such that one can derive at better results. Sometimes, these results may be better, resulting in more optimal performance of the system.

C. User Interface and Interaction Design

The user interface can be designed more engaging and interactive. Incorporate interactive features such as personalized dashboards, recommendation explanations, and proactive feedback mechanisms to encourage user interaction and feedback. As the interface improves better the application reaches to maximum people as it can be easily accessible by users.

D. Dynamic Clustering Techniques

Implement dynamic clustering techniques that adapt to evolving customer sentiments and product trends in real-time. Explore algorithms that automatically adjust cluster boundaries based on changing data distributions, ensuring the recommendation system remains agile and responsive.

E. Incorporating into dynamic recommendation systems

Incorporate the system into current recommendation systems to give tremendous results depicting personalized product suggestions and accurate recommendations through feature-level analysis. As the proposed model is mainly intended in aiding recommendation systems, its performance and its usage can be measured to the fullest, only when it gets incorporated with real-time systems.

7 References

- [1] K. R. Jerripothula, A. R. K. G. and Y. S. Rautela, "Feature-level Rating System using Customer Reviews and Review Votes," *IEEE Transactions on Computational Social Systems*, vol. 7, no. Oct. 2020, pp. 1210-1219, 2020.
- [2] H. C.J. and E. G. , "VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text," in *Eighth International AAAI Conference on Weblogs and Social Media (ICWSM-14)*, Michigan USA, 2014.
- [3] P. P. Pandey, M. and N. S. , "Sentiment Analysis on Customer Feedback Data: Amazon Product Reviews," in *International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, Faridabad, India, 2019.
- [4] K. S. A. Salunke, S. Dongare and K. Antala, "Recommender systems: An overview of different approaches to recommendations," in *International Conference on Innovations in Information, Embedded and Communication Systems (ICIECS)*, Coimbatore, India, 2017.
- [5] F. X and Z. J, "Sentiment analysis using product review data," *Journal of Big Data*, vol. 2, no. June 2015, pp. 1-5, 2015.