PROJECT TITLE: Analysis of Amazon Cell Phone Reviews Using NLP Technique

Team Member is:

Pooja Muthe
Kirti Suryawanshi
Vishakha Deshmane
Harshali Bhamare

Class :- S.Y B.SC Computer Science

College :- B.K. Birla College Of Arts, Science And Commerce

Team ID :- SWTID1727420425

Project Initialization and Planning Phase

Date	15 november 2024
Team ID	SWTID1727420425
Project Name	Amazon cell phone review analysis with nlp technique
Maximum Marks	3 Marks

Define Problem Statements (Customer Problem Statement Template):

Reference: https://miro.com/templates/customer-problem-statement/

Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	customer shopping for a new cell phone on Amazon.	find the best phone that fits my needs and budget.	it's difficult to make an informed decision due to the overwhelming number of reviews and mixed opinions.	there is a lot of feedback on product quality, battery life, and performance, and I don't know which reviews are trustworthy. reviews vary widely	frustrated and uncertain, unsure whether I'm making the right purchase. confused and
PS-2	a potential buyer looking to purchase a cell phone on Amazon.	understand the pros and cons of each model based on customer feedback.	the reviews are often too generic, with some not addressing specific features like camera quality or long-term durability.	and some are hard to trust, as they seem either overly positive or overly negative without providing enough detail	hesitant to purchase because I can't get a clear picture of the product's true quality.
PS-3	a customer comparing different cell	find reviews that give me a balanced	it's hard to filter through biased or fake reviews that	many reviews are either overly positive due to	distrustful and worried that I might end up with

	phone brands on Amazon.	perspective on the phone I am considering.	don’t accurately reflect the product’s performance.	incentives or overly negative due to isolated experiences.	a product that doesn’t meet my expectations.

Initial Project Planning Template

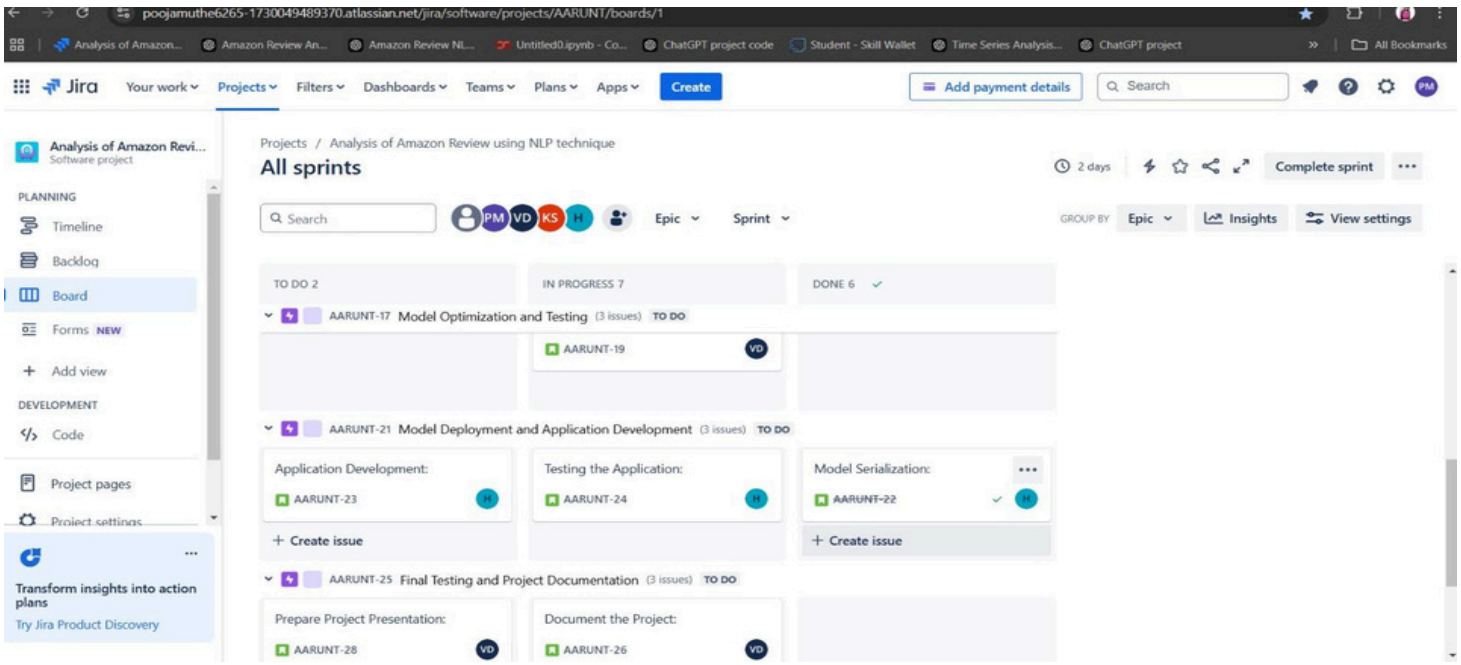
Date	27 Nov 2024
Team ID	SWTID1727420425
Project Name	Analysis of amazon review using NLP technique
Maximum Marks	4 Marks

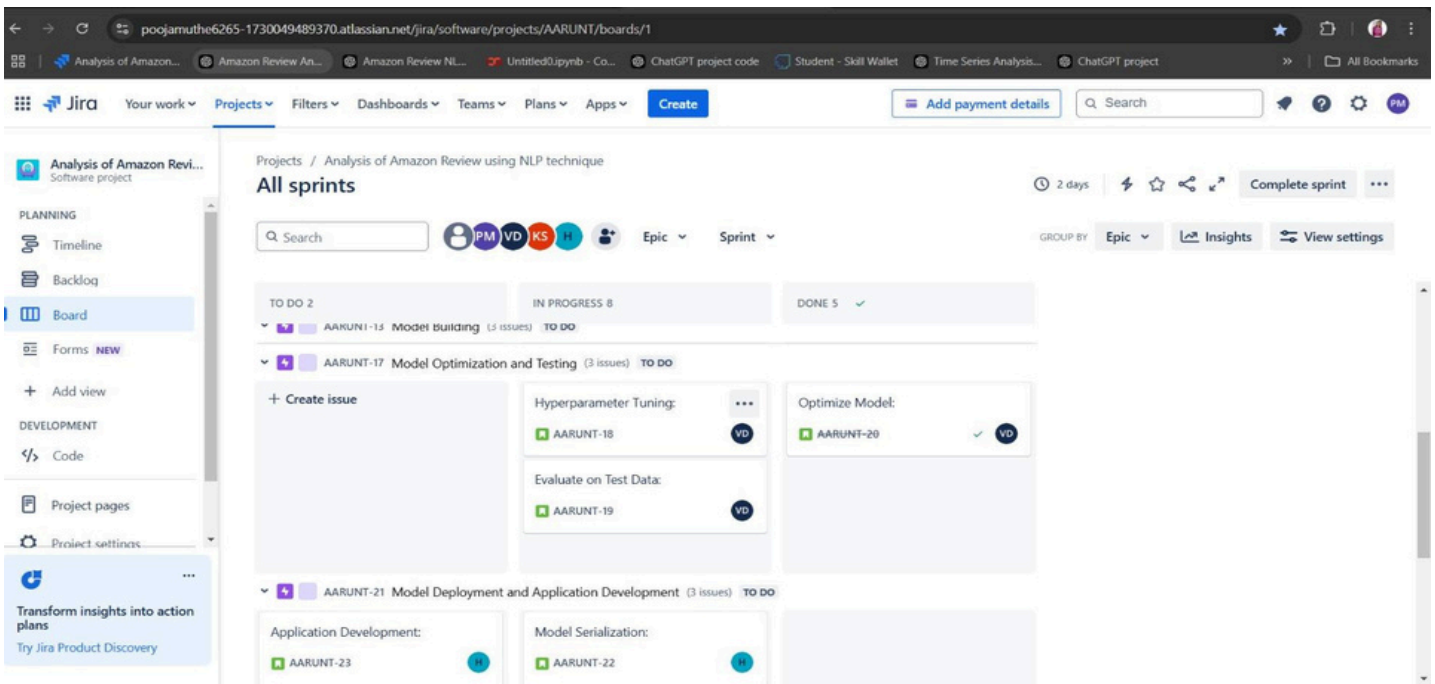
Product Backlog, Sprint Schedule, and Estimation (4 Marks)

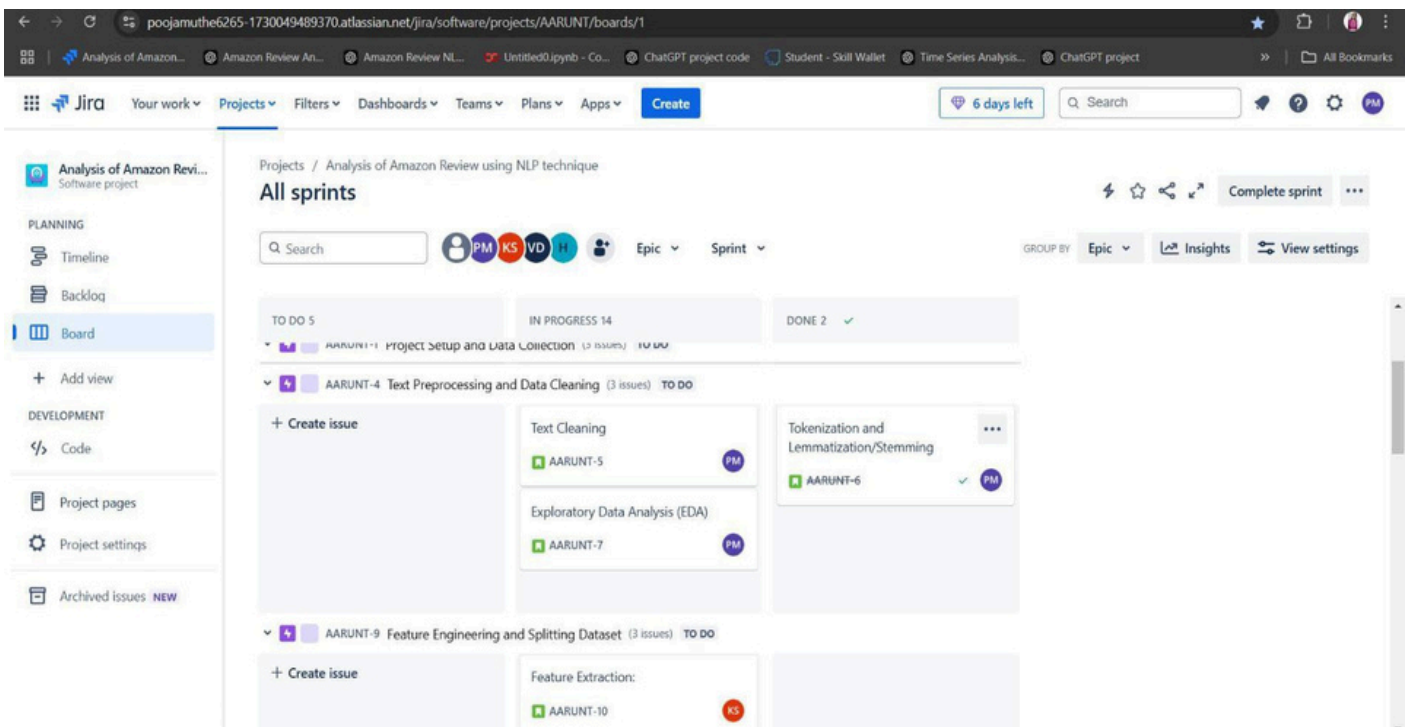
Use the below template to create a product backlog and sprint schedule

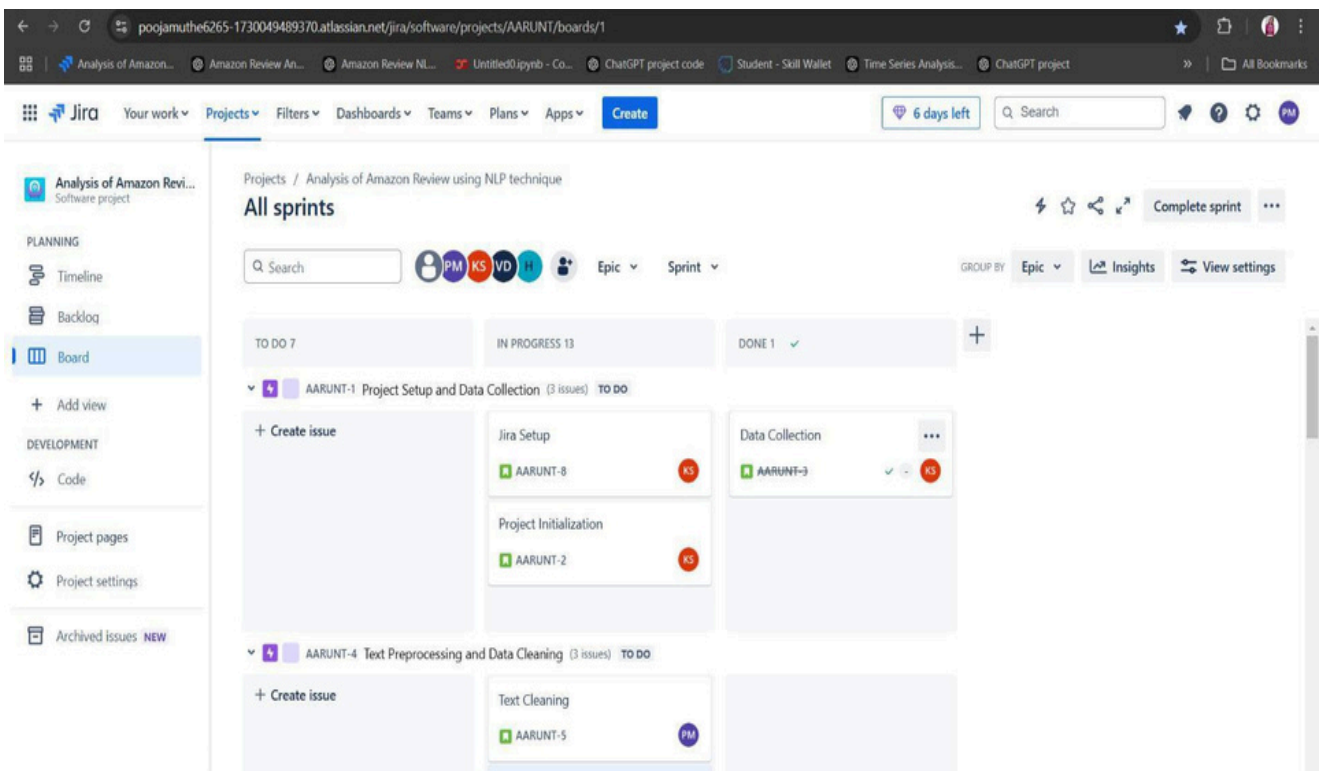
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members	Sprint Start Date	Sprint End Date (Planned)
Sprint-1	Project setup	USN-1	Data collection and loading data	2	High	Kirti	15/11/2024	15/11/202
Sprint-1	Data collection and processing	USN-2	Data cleaning	1	Medium	Kirti	15/11/2024	4 15/11/202
Sprint-2	Text processing	USN-3	Tokenization and Stemming	2	Low	Pooja	16/11/2024	4 17/11/202
Sprint-2	Text processing	USN-4	EDA		High	Pooja	17/11/2024	4
Sprint-3	Feature Engineering and Splitting Dataset	USN-5	Feature extraction , Set Baseline model and Data splitting	2	Low	Kirti	17/11/2024	17/11/202 4
Sprint-3	Model Building	USN-6	Model selection and initialization	2	High	Pooja	20/11/2024	20/11/2024

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points 1	Priority	Team Members	Sprint Start Date 21/11/2024	Sprint End Date (Planned)
Sprint-4	Model optimization and testing	USN-7	Hyperparameter tuning		Medium	Vishakha		21/11/2024
Sprint-4	Model optimization and testing	USN-8	Optimize model	2	High	Vishakha	21/11/2024	23/11/2024
Sprint-5	Model development and Application development	USN-9	Model Serialization	1	Medium	Harshali	23/11/2024	23/11/2024
Sprint-5	Model development and Application development	USN-10	Developing Application	1	Low	Harshali	24/11/2024	24/11/2024
Sprint-6	Final testing	USN-11	Testing Aplication	2	Medium	Vishakha	25/11/2024	27/11/2024


poojamuthe6265-1730049489370.atlassian.net/jira/software/projects/AARUNT/boards/1
Jira
Your work
Projects
Filters
Dashboards
Teams
Plans
Apps
Create
Add payment details
Search
Analysis of Amazon Revi...
Software project
PLANNING
Timeline
Backlog
Board
Forms NEW
Add view
DEVELOPMENT
Code
Project pages
Project settings
Transform insights into action plans
Try Jira Product Discovery
Projects / Analysis of Amazon Review using NLP technique
All sprints
2 days
Complete sprint
Search
Epic
Sprint
GROUP BY
Epic
Insights
View settings
TO DO 2
IN PROGRESS 7
DONE 6
AARUNT-17 Model Optimization and Testing (3 issues) TO DO
AARUNT-19
AARUNT-21 Model Deployment and Application Development (3 issues) TO DO
Application Development:
AARUNT-23
Testing the Application:
AARUNT-24
Model Serialization:
AARUNT-22
Create issue
Create issue
AARUNT-25 Final Testing and Project Documentation (3 issues) TO DO
Prepare Project Presentation:
AARUNT-28
Document the Project:
AARUNT-26


poojamuthe6265-1730049489370.atlassian.net/jira/software/projects/AARUNT/boards/1
Jira
Your work
Projects
Filters
Dashboards
Teams
Plans
Apps
Create
Add payment details
Search
Analysis of Amazon Revi...
Software project
PLANNING
Timeline
Backlog
Board
Forms NEW
Add view
DEVELOPMENT
Code
Project pages
Transform insights into action plans
Try Jira Product Discovery
Projects / Analysis of Amazon Review using NLP technique
All sprints
2 days
Complete sprint
Search
Epic
Sprint
GROUP BY
Insights
View settings
TO DO 2
IN PROGRESS 8
DONE 5
AARUNT-17 Model Optimization and Testing (3 issues) TO DO
Create issue
Hyperparameter Tuning:
AARUNT-18
Optimize Model:
AARUNT-20
Evaluate on Test Data:
AARUNT-19
AARUNT-21 Model Deployment and Application Development (3 issues) TO DO
Application Development:
AARUNT-23
Model Serialization:
AARUNT-22


poojamuthe6265-1730049489370.atlassian.net/jira/software/projects/AARUNT/boards/1
Jira
Your work
Projects
Filters
Dashboards
Teams
Plans
Apps
Create
6 days left
Search
Analysis of Amazon Revi...
Software project
PLANNING
Timeline
Backlog
Board
Add view
DEVELOPMENT
Code
Project pages
Project settings
Archived issues NEW
Projects / Analysis of Amazon Review using NLP technique
All sprints
Complete sprint
Search
Epic
Sprint
GROUP BY
Epic
Insights
View settings
TO DO 5
IN PROGRESS 14
DONE 2
AARUNT-4 Text Preprocessing and Data Cleaning (3 issues) TO DO
Create issue
Text Cleaning
AARUNT-5
Exploratory Data Analysis (EDA)
AARUNT-7
Tokenization and Lemmatization/Stemming
AARUNT-6
AARUNT-9 Feature Engineering and Splitting Dataset (3 issues) TO DO
Create issue
Feature Extraction:
AARUNT-10


poojamuthe6265-1730049489370.atlassian.net/jira/software/projects/AARUNT/boards/1
Jira
Your work
Projects
Filters
Dashboards
Teams
Plans
Apps
Create
6 days left
Search
Analysis of Amazon Revi...
Software project
PLANNING
Timeline
Backlog
Board
Add view
DEVELOPMENT
Code
Project pages
Project settings
Archived issues NEW
Projects / Analysis of Amazon Review using NLP technique
All sprints
Complete sprint
Search
Epic
Sprint
GROUP BY
Epic
Insights
View settings
TO DO 7
IN PROGRESS 13
DONE 1
AARUNT-1 Project Setup and Data Collection (3 issues) TO DO
Create issue
Jira Setup
AARUNT-8
Project Initialization
AARUNT-2
Data Collection
AARUNT-3
AARUNT-4 Text Preprocessing and Data Cleaning (3 issues) TO DO
Create issue
Text Cleaning
AARUNT-5

Project Initialization and Planning Phase

Date	11 November 2024
Team ID	SWTID1727420425
Project Title	analysis of amazon review using nlp technique
Maximum Marks	3 Marks

Project Proposal (Proposed Solution) template

The goal of this project is to analyze a large dataset of Amazon product reviews using advanced Natural Language Processing (NLP) techniques to gain insights into customer sentiment, trends, and product feedback. This analysis will help businesses and customers understand product performance and improve decision-making. The project will focus on sentiment analysis, keyword extraction, and review classification.

Project Overview	
Objective	The objective is to leverage NLP techniques to analyze Amazon reviews for extracting sentiment, trends, and actionable insights.
Scope	The scope includes applying NLP techniques to analyze Amazon reviews for sentiment analysis, topic modeling, trend identification across various product categories.
Problem Statement	
Description	Manually analyzing Amazon's vast reviews is inefficient, requiring automated solutions to extract insights, identify trends, and filter irrelevant feedback.
Impact	This project enables businesses to automate insights extraction from Amazon reviews, driving better customer satisfaction, product development, and marketing strategies.
Proposed Solution	
Approach	The approach involves applying NLP techniques like sentiment analysis, topic modeling, and text classification to automate the extraction of insights from Amazon reviews.

Key Features	topic modeling, keyword extraction, and review classification to derive actionable insights from Amazon reviews.

Resource Requirements

Resource Type	Description	Specification/Allocation
Hardware		
Computing Resources	AWS, Google Cloud, or Microsoft Azure for data storage, processing, and model training,	e.g., 16-32 GB
Memory	RAM , GPU/CPU	e.g., 32 GB
Storage	high-speed SSD storage	e.g., 500 GB
Software		
Frameworks	Python frameworks	e.g., Flask
Libraries	NLP libraries	e.g., SpaCy
Development Environment	software tools, libraries, hardware resources	e.g., Zira, Git,Vs
Data		
Data	• Data Sources: Public datasets (Amazon, Yelp, IMDb), web scraping, APIs (Twitter, News), proprietary/internal data, and crowdsourced data. • Data Size: Small-scale (thousands of entries), medium-scale (tens of thousands), large-scale (millions to billions). • Data Format: Text files, CSV, JSON, XML, Parquet, TFRecord — depending on how data is stored or retrieved	e.g., user details, review rating

SMARTBRIDGE
Let's Bridge the Gap
a Veranda Enterprise

Smart
Internz

Data Collection and Preprocessing Phase

Date	26 Nov 2024
Team ID	SWTID1727420425
Project Title	Analysis of amazon cell phone reviews using nlp technique
Maximum Marks	6 Marks

Data Exploration and Pre processing Report

The Amazon cell phone review analysis involves two datasets: item metadata and customer reviews. Initial exploration revealed insights into review counts, ratings distribution, and product coverage. Data pre processing included handling missing values, cleaning review text, and converting review dates for trend analysis. Sentiment labels (positive, neutral, negative) were derived from ratings, and features like review length were added. This prepared data is now clean and ready for advanced NLP tasks like sentiment analysis and topic modeling.

Section	Description
Data Overview	The project utilizes two datasets for analyzing Amazon cell phone reviews: one containing product metadata (e.g., name, brand, price) and another with customer reviews (e.g., ratings, review text, review date). The review dataset provides insights into customer feedback, while the item dataset complements it with product details. Together, these datasets enable the application of NLP techniques to understand customer sentiments, trends, and preferences, forming the foundation for comprehensive analysis.
Resizing	In the context of this project, resizing involves standardizing the size and structure of review text for efficient NLP processing. Long reviews were truncated or summarized to a manageable length, while very short reviews were either padded or excluded, depending on their relevance. This step ensures uniformity across the dataset, improving the performance and accuracy of NLP models by focusing on meaningful and consistent input data.

Normalization	Normalization in this project focuses on transforming the review text into a consistent format for NLP analysis. This includes converting all text to lowercase, removing special characters, punctuation, and numbers, and standardizing spellings. Additionally, lemmatization was applied to reduce words to their base forms, ensuring uniformity while preserving meaning. These steps help eliminate noise and improve the accuracy of NLP techniques such as sentiment analysis and topic modeling.
Edge Detection	Edge detection is not directly applicable to textual data analysis in this project, as it is a technique primarily used in image processing to identify object boundaries. However, in the context of NLP, analogous techniques might involve identifying "edges" of significant meaning, such as detecting key phrases, transitions in sentiment, or abrupt shifts in topic within reviews, which can be explored through advanced methods like keyword extraction or topic segmentation.
Color Space Conversion	Color space conversion is not directly relevant to text-based NLP tasks in this project, as it pertains to image processing and the transformation of pixel data between color models (e.g., RGB to grayscale). For this project, the focus remains on textual data processing, where equivalent steps involve text cleaning and transformation rather than handling visual data.
Data Preprocessing Code Screenshots	
Loading Data	

<table>
<tr><td></td><td>

+ Code + Text

• Data Collection

```
[ ] #load both items and review data
    items = pd.read_csv('/content/20191226-items.csv')
    reviews = pd.read_csv('/content/20191226-reviews.csv')
```

• Checking the structure of dataset

```
[ ] # Check column names in each file
    print("Columns in reviews:", reviews.columns)
    print("Columns in items:", items.columns)
```

```
Columns in reviews: Index(['asin', 'name', 'rating', 'date', 'verified', 'title', 'body',
       'helpfulVotes'],
      dtype='object')
Columns in items: Index(['asin', 'brand', 'title', 'url', 'image', 'rating', 'reviewUrl',
       'totalReviews', 'price', 'originalPrice'],
      dtype='object')
```

</td></tr>
<tr><td>Merging the Data</td><td>

• Merging the data

```
[ ] # Merge the datasets on a common column (e.g., 'product_id')
    merged_data = pd.merge(reviews, items, on='asin', how='left')
```

```
[ ] # Inspect the merged dataset
    print("Merged Data Shape:", merged_data.shape)
    print("Columns in Merged Data:", merged_data.columns)
```

```
Merged Data Shape: (67986, 17)
Columns in Merged Data: Index(['asin', 'name', 'rating_x', 'date', 'verified', 'title_x', 'body',
       'helpfulVotes', 'brand', 'title_y', 'url', 'image', 'rating_y',
       'reviewUrl', 'totalReviews', 'price', 'originalPrice'],
      dtype='object')
```

</td></tr>
<tr><td>Handling missing data</td><td>

• handling missing data

```
[ ] features = ['rating_x', 'verified', 'title_x', 'body', 'brand', 'price', 'originalPrice']
    target = 'helpfulVotes'
```

```
[ ] # Keep only necessary columns and drop rows with missing values
    filtered_data = merged_data[features + [target]].dropna()
```

```
[ ] print("Filtered Data Shape:", filtered_data.shape)
```

```
Filtered Data Shape: (27069, 8)
```

</td></tr>
<tr><td>Data Exploration</td><td>

• Data exploration

```
# Display the first few rows of the dataset
merged_data.head()
```

	asin	name	rating_x	date	verified	title_x	body	helpfulVotes	brand	title_y	url	image	rating_y	reviewUrl	totalReviews	price	originalPrice
0	B00005QX2UC	Janet	3	October 11, 2005	False	Def not best, but not worst	I had the Samsung A630 for awhile which is alm...	1.0	NaN	Dual-Band / Tri-Mode Sprint PCS Phone w/ Voice...	https://www.amazon.com/Dual-Band-Tri-Mode-Acti...	https://m.media-amazon.com/images/I/0143EBQ210...	3.0	https://www.amazon.com/product-reviews/B00005QX2UC	14	0.0	0.0
1	B00005QX2UC	Luke Wyatt	1	January 7, 2004	False	Text Messaging Doesn't Work	Due to a software issue between Nokia and Spri...	17.0	NaN	Dual-Band / Tri-Mode Sprint PCS Phone w/ Voice...	https://www.amazon.com/Dual-Band-Tri-Mode-Acti...	https://m.media-amazon.com/images/I/0143EBQ210...	3.0	https://www.amazon.com/product-reviews/B00005QX2UC	14	0.0	0.0
2	B00005QX2UC	Brooke	5	December 30, 2003	False	Love This Phone	This is a great, reliable phone. I also purcha...	5.0	NaN	Dual-Band / Tri-Mode Sprint PCS Phone w/ Voice...	https://www.amazon.com/Dual-Band-Tri-Mode-Acti...	https://m.media-amazon.com/images/I/0143EBQ210...	3.0	https://www.amazon.com/product-reviews/B00005QX2UC	14	0.0	0.0
3	B00005QX2UC	amy m. teague	3	March 10, 2004	False	Love the Phone, BUT..!	I love the phone and all, because I really did...	1.0	NaN	Dual-Band / Tri-Mode Sprint PCS Phone w/ Voice...	https://www.amazon.com/Dual-Band-Tri-Mode-Acti...	https://m.media-amazon.com/images/I/0143EBQ210...	3.0	https://www.amazon.com/product-reviews/B00005QX2UC	14	0.0	0.0
4	B00005QX2UC	tristachimmer	4	August 20, 2005	False	Great phone service and options,	The phone has been great for every	1.0	NaN	Dual-Band / Tri-Mode Sprint PCS	https://www.amazon.com/Dual-Band-Tri-Mode-Acti...	https://m.media-amazon.com/images/I/0143EBQ210...	3.0	https://www.amazon.com/product-reviews/B00005QX2UC	14	0.0	0.0

</td></tr>
</table>

```
merged_data.describe()
```

	rating_x	helpfulVotes	rating_y	totalReviews	price	originalPrice
count	67986.000000	27215.000000	67986.000000	67986.000000	67986.000000	67986.000000
mean	3.807916	8.229690	3.766826	373.742800	222.050506	84.057634
std	1.582906	31.954877	0.429197	262.560876	188.863986	201.923373
min	1.000000	1.000000	1.000000	1.000000	0.000000	0.000000
25%	3.000000	1.000000	3.500000	153.000000	103.980000	0.000000
50%	5.000000	2.000000	3.800000	336.000000	179.990000	0.000000
75%	5.000000	5.000000	4.100000	558.000000	300.550000	0.000000
max	5.000000	990.000000	5.000000	983.000000	999.990000	999.990000

```
# Check for null values
merged_data.isnull().sum()
```

	0
asin	0
name	3
rating_x	0
date	0
verified	0
title_x	29
body	26
helpfulVotes	40771
brand	200
title_y	0
url	0
image	0
rating_y	0
reviewUrl	0
totalReviews	0
price	0
originalPrice	0

dtype: int64

Splitting the data

```
5. Splitting the data

[ ] # Split the data for training and validation
    X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.2, random_state=42)
```

Data Collection and Preprocessing Phase

Date	20 November 2024
Team ID	SWTID1727420425
Project Title	Analysis of amazon review using nlp technique
Maximum Marks	2 Marks

Data Quality Report Template

The Data Quality Report Template will summarize data quality issues from the selected source, including severity levels and resolution plans. It will aid in systematically identifying and rectifying data discrepancies. This template provides a comprehensive view of potential data quality issues in the Amazon reviews dataset and outlines technical solutions to address these challenges, ensuring better accuracy and consistency in the NLP analysis.

Data Source	Data Quality Issue	Severity	Resolution Plan
Amazon Reviews Dataset	Duplicate reviews (identical reviews posted multiple times).	Moderate	Deduplicate reviews based on a combination of review ID and text similarity (e.g., using a threshold for cosine similarity or exact matching).
Amazon Reviews Dataset	Missing or incomplete review data (e.g., empty reviews, missing ratings).	High	Implement a data cleaning step to filter out empty reviews and ensure that ratings are present. Use imputation techniques if needed for missing ratings

Amazon Reviews Dataset	Noisy or irrelevant reviews (e.g., reviews that don't provide useful sentiment or context, such as spam or promotional content).	High	Preprocess data to identify and remove non-relevant content using keyword filtering, text length checks, or external classifiers to detect spam.

Data Collection and Preprocessing Phase

Date	22 November 2024
Team ID	SWTID1727420425
Project Title	Analysis amazon cell phone review with nlp technique
Maximum Marks	2 Marks

Data Collection Plan & Raw Data Sources Identification Template

Elevate your data strategy with the Data Collection plan and the Raw Data Sources report, ensuring meticulous data curation and integrity for informed decision-making in every analysis and decision-making endeavor.
Data Collection Plan Template

Section	Description
Project Overview	The project utilizes two datasets for analyzing Amazon cell phone reviews: one containing product metadata (e.g., name, brand, price) and another with customer reviews (e.g., ratings, review text, review date). The review dataset provides insights into customer feedback, while the item dataset complements it with product details. Together, these datasets enable the application of NLP techniques to understand customer sentiments, trends, and preferences, forming the foundation for comprehensive analysis
Data Collection Plan	The data collection plan for this project involves gathering two key datasets: product metadata and customer reviews from Amazon. Product metadata includes details such as product names, brands, and specifications, while the review dataset contains review text, ratings, and dates. The data is sourced through web scraping or public datasets, ensuring it is comprehensive and representative of user feedback. Proper filtering is applied to include only relevant

	cell phone reviews, forming the basis for meaningful analysis using NLP techniques.
Raw Data Sources Identified	The raw data sources for this project include publicly available datasets and web-scraped information from Amazon. These sources provide detailed customer reviews, ratings, review dates, and metadata such as product names, brands, and specifications. The identified datasets are rich in content, enabling comprehensive analysis of customer sentiments and trends using NLP techniques while ensuring relevance to cell phone products.

Raw Data Sources Template

Source Name	Description	Location/URL	Format	Size	Access Permissions
Dataset 1	I have two dataset reviews and items.	https://www.kaggle.com/grikomsn/amazon-cell-phones-reviews	CSV	Review=24800KB Items=226KB	Public

Model Development Phase Template

Date	15 November 2024
Team ID	SWTID1727420425
Project Title	Analysis of amazon review using nlp technique
Maximum Marks	5 Marks

Model Selection Report

The goal of this project is to analyze Amazon reviews using Natural Language Processing (NLP) techniques to extract meaningful insights, such as sentiment classification (positive, negative, neutral), product categorization, and identifying key aspects or themes. Given the variety of deep learning models, this report evaluates multiple architectures to determine the best model based on performance, complexity, computational requirements, and suitability for the task.

The following deep learning models are considered for analyzing Amazon reviews using NLP techniques:

- Convolutional Neural Networks (CNNs)
- ANN (Artificial Neural Network)
- Long Short-Term Memory Networks (LSTMs)
- BiLSTM (Bidirectional Long Short-Term Memory)

Model Selection Report:

Model	Description
Convolutional Neural	CNNs are typically used for image data, but they can also be applied to text classification by treating text as a sequence of words or characters.

Networks (CNNs)	CNNs capture local patterns and features in the text, which are helpful for tasks such as sentiment classification. • Performance: CNNs are effective for text classification tasks, particularly for short text or reviews with clear sentiment indicators. They can quickly identify important features or keywords in the review. • Computational Requirements: CNNs are generally less computationally expensive compared to sequential models like RNNs or Transformer-based models. • Training Time: CNNs are relatively fast to train, as they can leverage parallel processing (especially for small- to medium-sized datasets). • Interpretability: CNNs are somewhat interpretable, with learned filters and feature maps that can help identify key parts of the text that contribute to classification.
ANN (Artificial Neural Network)	ANNs (Artificial Neural Networks) can be applied to various types of data, including sequential data, though they are not specifically designed for it like RNNs (Recurrent Neural Networks). However, with certain adaptations, ANNs can still handle tasks that involve sequential information, such as text processing. • Performance: ANNs are versatile and can perform well in many tasks, including those that require analyzing sequential data. For instance, when adapted to process sequences, such as in text classification, ANNs can capture relationships between features, though they might not naturally capture sequential dependencies as efficiently as RNNs. • Handling Sequential Dependencies: While ANNs are not inherently designed to handle sequential dependencies, techniques like feedforward networks can be adapted for sequence processing, but they may not be as effective as RNNs at capturing long-range dependencies. Specialized architectures, such as Convolutional Neural Networks (CNNs), can also be used for sequential data by sliding filters over sequences to extract patterns,

	though they don't retain memory of prior inputs the same way RNNs do. • Interpretability: ANNs, especially deep networks, can be less interpretable due to their complex structures. While CNNs tend to offer more insights into feature extraction (e.g., visual features), traditional ANNs are harder to interpret directly. However, techniques like attention mechanisms or LIME (Local Interpretable Model-agnostic Explanations) can help make ANNs more interpretable by highlighting relevant features or portions of the input data that contribute most to the final output.
Long Short-Term Memory Networks (LSTMs)	**LSTMs are a specialized form of RNNs designed to overcome the vanishing gradient problem, allowing them to capture long-term dependencies in sequential data. LSTMs have "gates" that regulate the flow of information, making them effective for understanding long-term context in text.** Strengths: • Performance: LSTMs are well-suited for tasks where long-range dependencies matter, such as sentiment analysis on product reviews with complex structure or contextual shifts. • Handling Complex Context: LSTMs excel in tasks requiring deeper understanding of how sentiment can evolve across longer text sequences. • Training Time: LSTMs can be slower to train compared to CNNs but are still faster than Transformer-based models. • Interpretability: LSTMs are challenging to interpret directly, but attention mechanisms can help highlight the parts of the sequence that influence the model's predictions.
BiLSTM stands for Bidirectional	BiLSTMs (Bidirectional Long Short-Term Memory networks) are an extension of LSTMs (Long Short-Term Memory networks), designed to capture context from both past and future sequences. While LSTMs

Long Short-Term Memory	process data in one direction, BiLSTMs process sequences in both forward and backward directions, providing a richer understanding of the entire context. Strengths of BiLSTMs: • Performance: o BiLSTMs are highly effective for tasks where both past and future context are crucial for understanding the sequence. They are particularly well-suited for tasks like sentiment analysis on product reviews, where understanding not just the prior context but also future content helps capture shifts in sentiment over time. By processing data in both directions, BiLSTMs offer a more comprehensive understanding of text, improving performance on various sequence-based tasks. • Handling Complex Context: o BiLSTMs excel at capturing complex, long-range dependencies in sequential data, especially when context evolves across the entire sequence. For instance, in tasks where sentiment might change based on information from both before and after a certain point (such as in long product reviews or conversations), BiLSTMs can consider future tokens in addition to past ones, which enhances their ability to understand the evolution of meaning and sentiment in a given text. • Training Time: o While BiLSTMs are generally slower to train than models like CNNs (Convolutional Neural Networks) due to their sequential nature, they are still typically faster than Transformer-based models. Transformers, which process the entire sequence in parallel, require more computational resources and time, especially for long sequences.

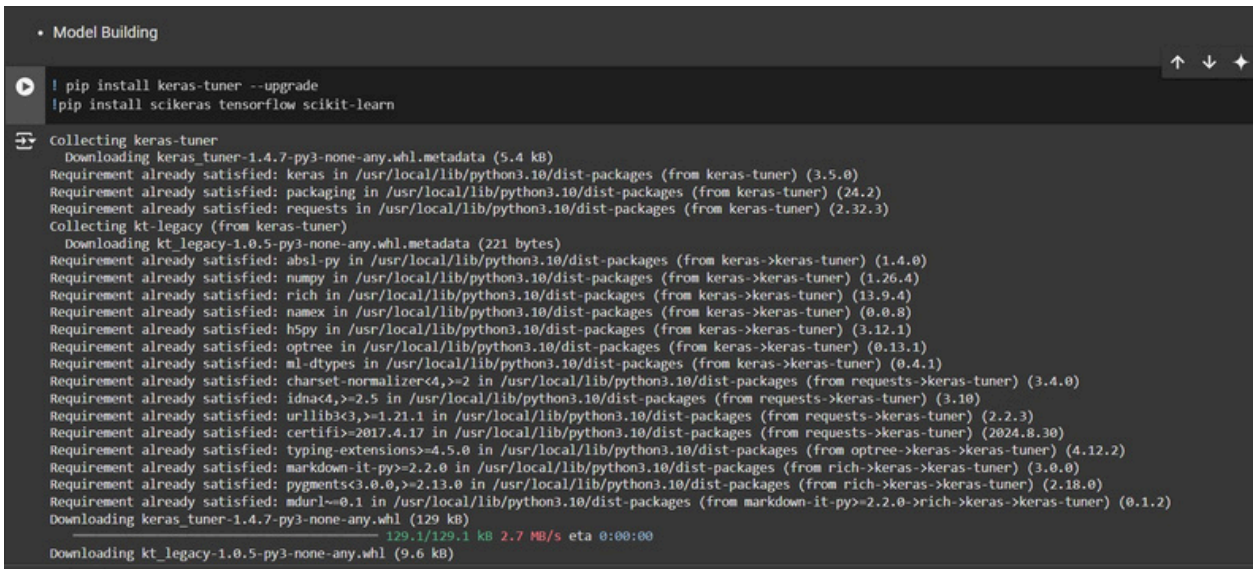
Model Development Phase Template

Date	24 November 2024 SWTID1727420425
Team ID	Analysis amazon cell phone review with nlp technique
Project Title	10 Marks
Maximum Marks	

Initial Model Training Code, Model Validation and Evaluation Report

The initial model training code for analyzing Amazon cell phone reviews using NLP techniques will be showcased in the future through a screenshot. The model validation and evaluation report will provide a summary of the performance, including training and validation metrics for multiple models. These performance metrics will be presented through respective screenshots, highlighting key indicators such as accuracy, precision, recall, and F1 score to assess the effectiveness of the NLP models in classifying and analyzing customer sentiments from the reviews.

Initial Model Training Code (5 marks):



```
[ ] import keras_tuner as kt
    import tensorflow as tf
    from keras.models import Sequential
    from keras.layers import Dense, Conv1D, LSTM, Bidirectional, Embedding, MaxPooling1D, Dropout, Flatten
    from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import LabelEncoder
    from sklearn.metrics import classification_report
    from scikeras.wrappers import KerasClassifier
    from sklearn.model_selection import GridSearchCV
    from tensorflow.keras.optimizers import Adam
```

- Data Pre processing

```
[ ] # Selecting features and target
    features = ['rating_x', 'verified', 'title_x', 'body', 'brand', 'price', 'originalPrice']
    target = 'helpfulVotes'
```

```
    # Clean text function
    def clean_text(text):
        text = text.lower()
        text = re.sub(r'\W', ' ', text)  # Remove special characters
        text = re.sub(r'\s+[a-zA-Z]\s+', ' ', text)  # Remove single characters
        text = re.sub(r'\s+', ' ', text)  # Remove multiple spaces
        return text
```

+ Code + Text

```
[ ] # Apply cleaning to text features
    merged_data['body'] = merged_data['body'].fillna('').apply(clean_text)
    merged_data['title_x'] = merged_data['title_x'].fillna('').apply(clean_text)
```

```
[ ] # Filter data and drop missing rows
    filtered_data = merged_data[features + [target]].dropna()
```

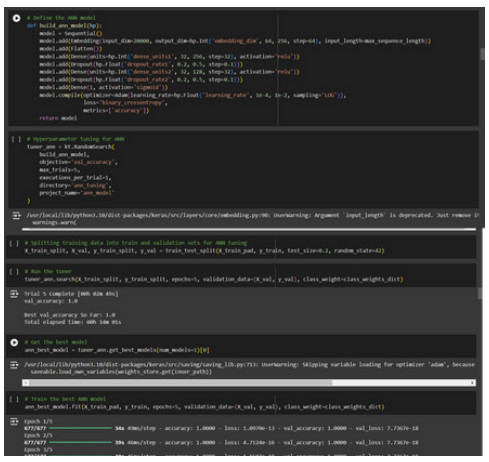
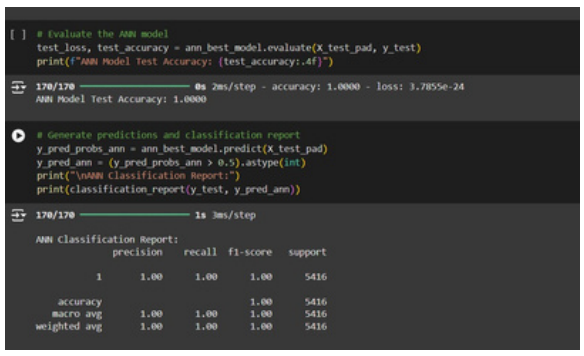
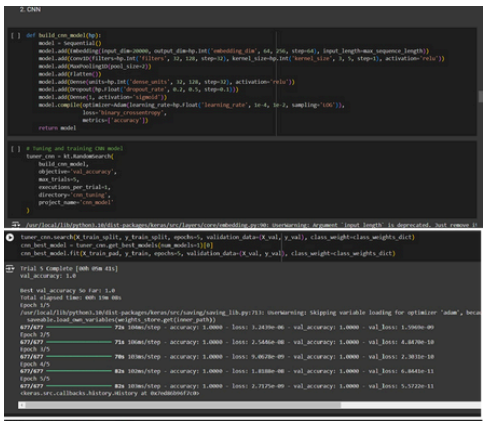
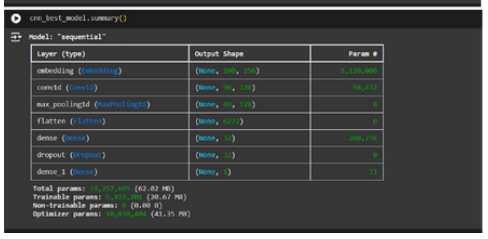
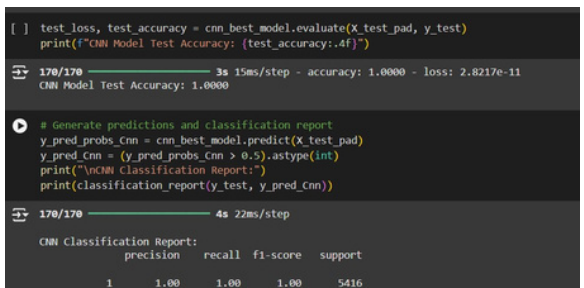
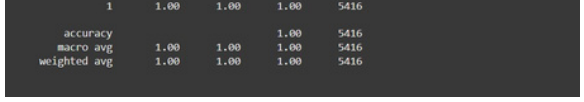
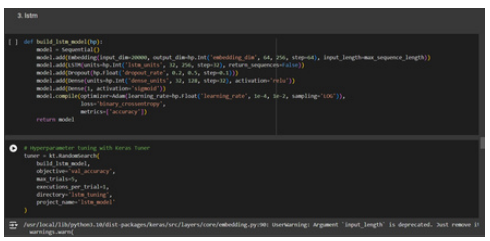
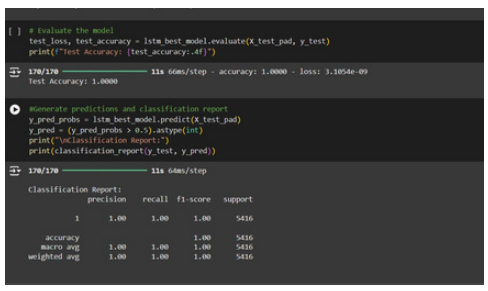
```
[ ]  max_sequence_length = 100
     X_train_pad = pad_sequences(X_train_seq, maxlen=max_sequence_length)
     X_test_pad = pad_sequences(X_test_seq, maxlen=max_sequence_length)
```

```
[ ]  # Binarizing the target
     y_train = y_train.values
     y_test = y_test.values
```

```
     # Define class weights to handle imbalance
     class_weights = class_weight.compute_class_weight(
         class_weight='balanced',
         classes=np.unique(y_train),
         y=y_train
     )
     class_weights_dict = {i: class_weights[i] for i in range(len(class_weights))}
```

Model Validation and Evaluation Report (5 marks):

Model	Model code	Model Classification

ANN		
CNN	 	 
LSTM		

```
# Evaluate the ANN model
test_loss, test_accuracy = ann_best_model.evaluate(X_test_pad, y_test)
print(f"ANN Model Test Accuracy: {test_accuracy:.4f}")
```

```
170/170 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 1.0000 - loss: 3.7855e-24
ANN Model Test Accuracy: 1.0000
```

```
# Generate predictions and classification report
y_pred_probs_ann = ann_best_model.predict(X_test_pad)
y_pred_ann = (y_pred_probs_ann > 0.5).astype(int)
print("\nANN Classification Report:")
print(classification_report(y_test, y_pred_ann))
```

```
170/170 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step

ANN Classification Report:
              precision    recall  f1-score   support

           1       1.00      1.00      1.00      5416

    accuracy                           1.00      5416
   macro avg       1.00      1.00      1.00      5416
weighted avg       1.00      1.00      1.00      5416
```

```
2. CNN

def build_cnn_model(hp):
    model = Sequential()
    model.add(Embedding(input_dim=20000, output_dim=hp.Int('embedding_dim', 64, 256, step=64), input_length=max_sequence_length))
    model.add(Conv1D(filters=hp.Int('filters', 32, 128, step=32), kernel_size=hp.Int('kernel_size', 3, 5, step=1), activation='relu'))
    model.add(MaxPooling1D(pool_size=2))
    model.add(Flatten())
    model.add(Dense(units=hp.Int('dense_units', 32, 128, step=32), activation='relu'))
    model.add(Dropout(hp.Float('dropout_rate', 0.2, 0.5, step=0.1)))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(optimizer=Adam(learning_rate=hp.Float('learning_rate', 1e-4, 1e-2, sampling='LOG')),
                  loss='binary_crossentropy',
                  metrics=['accuracy'])
    return model

# Tuning and training CNN model
tuner_cnn = kt.RandomSearch(
    build_cnn_model,
    objective='val_accuracy',
    max_trials=5,
    executions_per_trial=1,
    directory='cnn_tuning',
    project_name='cnn_model'
)

/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is deprecated. Just remove it
  warnings.warn(

tuner_cnn.search(X_train_split, y_train_split, epochs=5, validation_data=(X_val, y_val), class_weight=class_weights_dict)
cnn_best_model = tuner_cnn.get_best_models(num_models=1)[0]
cnn_best_model.fit(X_train_pad, y_train, epochs=5, validation_data=(X_val, y_val), class_weight=class_weights_dict)

Trial 5 Complete [00h 05m 41s]
val_accuracy: 1.0

Best val_accuracy So Far: 1.0
Total elapsed time: 00h 17m 08s
Epoch 1/5
/usr/local/lib/python3.10/dist-packages/keras/src/saving/saving_lib.py:713: UserWarning: Skipping variable loading for optimizer 'adam', because
  saveable.load_own_variables(weights_store.get(inner_path))
677/677 ━━━━━━━━━━━━━━━━━━━━ 72s 106ms/step - accuracy: 1.0000 - loss: 3.2439e-06 - val_accuracy: 1.0000 - val_loss: 1.5969e-09
Epoch 2/5
677/677 ━━━━━━━━━━━━━━━━━━━━ 71s 106ms/step - accuracy: 1.0000 - loss: 2.5486e-08 - val_accuracy: 1.0000 - val_loss: 4.4470e-10
Epoch 3/5
677/677 ━━━━━━━━━━━━━━━━━━━━ 70s 103ms/step - accuracy: 1.0000 - loss: 9.0629e-09 - val_accuracy: 1.0000 - val_loss: 2.3031e-10
Epoch 4/5
677/677 ━━━━━━━━━━━━━━━━━━━━ 82s 102ms/step - accuracy: 1.0000 - loss: 5.8108e-08 - val_accuracy: 1.0000 - val_loss: 6.8645e-11
Epoch 5/5
677/677 ━━━━━━━━━━━━━━━━━━━━ 82s 103ms/step - accuracy: 1.0000 - loss: 2.7175e-09 - val_accuracy: 1.0000 - val_loss: 5.5722e-11
<keras.src.callbacks.history.History at 0x7e6b8266f2b0>

cnn_best_model.summary()

Model: "sequential"
```

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 100, 256)	5,120,000
conv1d (Conv1D)	(None, 98, 128)	98,432
max_pooling1d (MaxPooling1D)	(None, 49, 128)	0
flatten (Flatten)	(None, 6272)	0
dense (Dense)	(None, 96)	602,208
dropout (Dropout)	(None, 96)	0
dense_1 (Dense)	(None, 1)	33

```
Total params: 16,253,685 (62.02 MB)
Trainable params: 5,813,293 (20.67 MB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 10,840,392 (41.35 MB)
```

```
test_loss, test_accuracy = cnn_best_model.evaluate(X_test_pad, y_test)
print(f"CNN Model Test Accuracy: {test_accuracy:.4f}")
```

```
170/170 ━━━━━━━━━━━━━━━━━━━━ 3s 15ms/step - accuracy: 1.0000 - loss: 2.8217e-11
CNN Model Test Accuracy: 1.0000
```

```
# Generate predictions and classification report
y_pred_probs_Cnn = cnn_best_model.predict(X_test_pad)
y_pred_Cnn = (y_pred_probs_Cnn > 0.5).astype(int)
print("\nCNN Classification Report:")
print(classification_report(y_test, y_pred_Cnn))
```

```
170/170 ━━━━━━━━━━━━━━━━━━━━ 4s 22ms/step

CNN Classification Report:
              precision    recall  f1-score   support

           1       1.00      1.00      1.00      5416

    accuracy                           1.00      5416
   macro avg       1.00      1.00      1.00      5416
weighted avg       1.00      1.00      1.00      5416
```

```
3. lstm

def build_lstm_model(hp):
    model = Sequential()
    model.add(Embedding(input_dim=20000, output_dim=hp.Int('embedding_dim', 64, 256, step=64), input_length=max_sequence_length))
    model.add(LSTM(units=hp.Int('lstm_units', 32, 256, step=32), return_sequences=False))
    model.add(Dropout(hp.Float('dropout_rate', 0.2, 0.5, step=0.1)))
    model.add(Dense(units=hp.Int('dense_units', 32, 128, step=32), activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(optimizer=Adam(learning_rate=hp.Float('learning_rate', 1e-4, 1e-2, sampling='LOG')),
                  loss='binary_crossentropy',
                  metrics=['accuracy'])
    return model

# Hyperparameter tuning with Keras Tuner
tuner = kt.RandomSearch(
    build_lstm_model,
    objective='val_accuracy',
    max_trials=5,
    executions_per_trial=1,
    directory='lstm_tuning',
    project_name='lstm_model'
)

/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is deprecated. Just remove it
  warnings.warn(
```

```
# Evaluate the model
test_loss, test_accuracy = lstm_best_model.evaluate(X_test_pad, y_test)
print(f"Test Accuracy: {test_accuracy:.4f}")
```

```
170/170 ━━━━━━━━━━━━━━━━━━━━ 11s 66ms/step - accuracy: 1.0000 - loss: 3.1054e-09
Test Accuracy: 1.0000
```

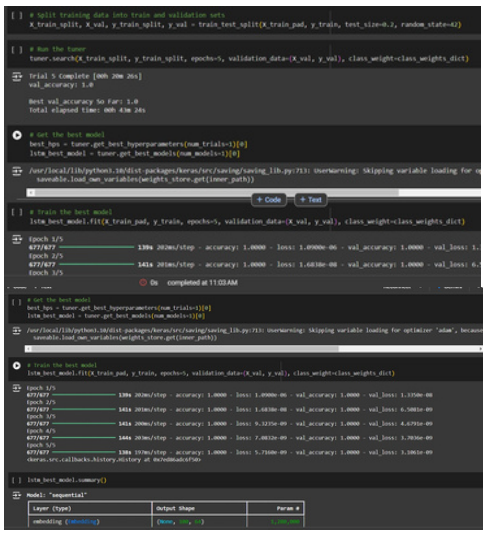
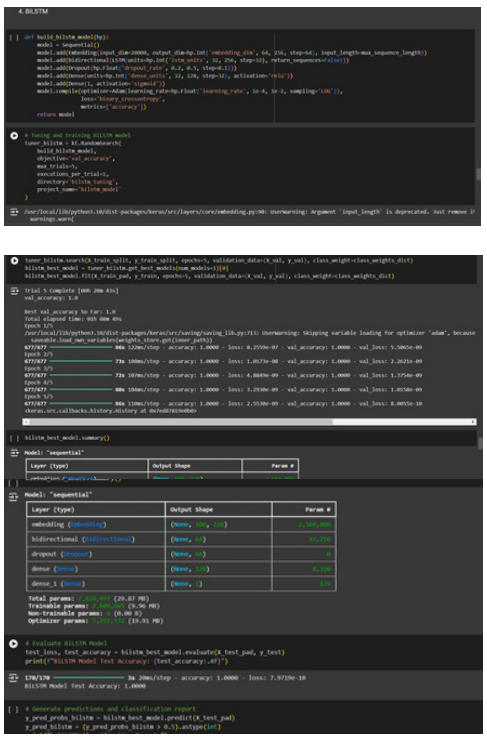
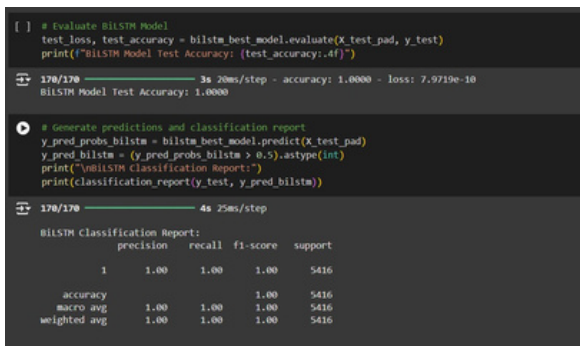
```
#Generate predictions and classification report
y_pred_probs = lstm_best_model.predict(X_test_pad)
y_pred = (y_pred_probs > 0.5).astype(int)
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

```
170/170 ━━━━━━━━━━━━━━━━━━━━ 11s 64ms/step

Classification Report:
              precision    recall  f1-score   support

           1       1.00      1.00      1.00      5416

    accuracy                           1.00      5416
   macro avg       1.00      1.00      1.00      5416
weighted avg       1.00      1.00      1.00      5416
```

		
BiLSTM		

Model Optimization and Tuning Phase Template

Date	26 November 2024
Team ID	SWTID1727420425
Project Title	Analysis of Amazon Cell Phone Reviews Using NLP Technique
Maximum Marks	10 Marks

Model Optimization and Tuning Phase

The Model Optimization and Tuning Phase involves refining neural network models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

Hyperparameter Tuning Documentation (8 Marks):

Model	Tuned Hyperparameters

ANN Model	`# Hyperparameter tuning for ANN`
	`tuner_ann = kt.RandomSearch(`
	`build_ann_model,`
	`objective='val_accuracy',`
	`max_trials=5,`
	`executions_per_trial=1,`
	`directory='ann_tuning',`
	`project_name='ann_model'`
	`)`
	`/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument 'input_length' is deprecated. Just remove it.`
	`warnings.warn(`
	`# Splitting training data into train and validation sets for ANN tuning`
	`X_train_split, X_val, y_train_split, y_val = train_test_split(X_train_pad, y_train, test_size=0.2, random_state=42)`
	`# Run the tuner`
	`tuner_ann.search(X_train_split, y_train_split, epochs=5, validation_data=(X_val, y_val), class_weight=class_weights_dict)`
	`Trial 5 Complete [00h 02m 49s]`
	`val_accuracy: 1.0`
	`Best val_accuracy So Far: 1.0`
	`Total elapsed time: 00h 14m 01s`

CNN Model	`# Tuning and training CNN model`
	`tuner_cnn = kt.RandomSearch(`
	`build_cnn_model,`
	`objective='val_accuracy',`
	`max_trials=5,`
	`executions_per_trial=1,`
	`directory='cnn_tuning',`
	`project_name='cnn_model'`
	`)`
	`/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument 'input_length' is deprecated. Just remove it.`
	`warnings.warn(`

<table>
<tr>
<td></td>
<td>

```
tuner_cnn.search(X_train_split, y_train_split, epochs=5, validation_data=(X_val, y_val), class_weight=class_weights_dict)
cnn_best_model = tuner_cnn.get_best_models(num_models=1)[0]
cnn_best_model.fit(X_train_pad, y_train, epochs=5, validation_data=(X_val, y_val), class_weight=class_weights_dict)
```

```
Trial 5 Complete [00h 05m 41s]
val_accuracy: 1.0

Best val_accuracy So Far: 1.0
Total elapsed time: 00h 19m 08s
Epoch 1/5
/usr/local/lib/python3.10/dist-packages/keras/src/saving/saving_lib.py:713: UserWarning: Skipping variable loading for optimizer 'a
  saveable.load_own_variables(weights_store.get(inner_path))
677/677 ━━━━━━━━━━━━━━━━━━━━ 72s 104ms/step - accuracy: 1.0000 - loss: 3.2439e-06 - val_accuracy: 1.0000 - val_loss: 1.5969e-09
Epoch 2/5
677/677 ━━━━━━━━━━━━━━━━━━━━ 71s 106ms/step - accuracy: 1.0000 - loss: 2.5446e-08 - val_accuracy: 1.0000 - val_loss: 4.8470e-10
Epoch 3/5
677/677 ━━━━━━━━━━━━━━━━━━━━ 70s 103ms/step - accuracy: 1.0000 - loss: 9.0678e-09 - val_accuracy: 1.0000 - val_loss: 2.3031e-10
Epoch 4/5
677/677 ━━━━━━━━━━━━━━━━━━━━ 82s 102ms/step - accuracy: 1.0000 - loss: 1.8188e-08 - val_accuracy: 1.0000 - val_loss: 6.8441e-11
Epoch 5/5
677/677 ━━━━━━━━━━━━━━━━━━━━ 82s 103ms/step - accuracy: 1.0000 - loss: 2.7175e-09 - val_accuracy: 1.0000 - val_loss: 5.5722e-11
<keras.src.callbacks.history.History at 0x7ed86b96f7c0>
```

</td>
</tr>
<tr>
<td>LSTM Model</td>
<td>

```
# Hyperparameter tuning with Keras Tuner
tuner = kt.RandomSearch(
    build_lstm_model,
    objective='val_accuracy',
    max_trials=5,
    executions_per_trial=1,
    directory='lstm_tuning',
    project_name='lstm_model'
)
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is deprecated. Just remove it.
  warnings.warn(
```

```
# Split training data into train and validation sets
X_train_split, X_val, y_train_split, y_val = train_test_split(X_train_pad, y_train, test_size=0.2, random_state=42)
```

```
# Run the tuner
tuner.search(X_train_split, y_train_split, epochs=5, validation_data=(X_val, y_val), class_weight=class_weights_dict)
```

```
Trial 5 Complete [00h 20m 26s]
val_accuracy: 1.0

Best val_accuracy So Far: 1.0
Total elapsed time: 00h 43m 24s
```

```
# Get the best model
best_hps = tuner.get_best_hyperparameters(num_trials=1)[0]
lstm_best_model = tuner.get_best_models(num_models=1)[0]
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/saving/saving_lib.py:713: UserWarning: Skipping variable loading for optimizer 'adam', because it has 2 va
  saveable.load_own_variables(weights_store.get(inner_path))
```

```
# Train the best model
lstm_best_model.fit(X_train_pad, y_train, epochs=5, validation_data=(X_val, y_val), class_weight=class_weights_dict)
```

</td>
</tr>
<tr>
<td>BiLSTM Model</td>
<td>

```
# Tuning and training BiLSTM model
tuner_bilstm = kt.RandomSearch(
    build_bilstm_model,
    objective='val_accuracy',
    max_trials=5,
    executions_per_trial=1,
    directory='bilstm_tuning',
    project_name='bilstm_model'
)
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is deprecated. Just remove it.
  warnings.warn(
```

</td>
</tr>
</table>

```
tuner_bilstm.search(X_train_split, y_train_split, epochs=5, validation_data=(X_val, y_val), class_weight=class_weights_dict)
bilstm_best_model = tuner_bilstm.get_best_models(num_models=1)[0]
bilstm_best_model.fit(X_train_pad, y_train, epochs=5, validation_data=(X_val, y_val), class_weight=class_weights_dict)
```

```
Trial 5 Complete [00h 20m 43s]
val_accuracy: 1.0

Best val_accuracy So Far: 1.0
Total elapsed time: 01h 08m 49s
Epoch 1/5
/usr/local/lib/python3.10/dist-packages/keras/src/saving/saving_lib.py:713: UserWarning: Skipping variable loading for optimizer 'adam', because it has 2 variables whereas the
  saveable.load_own_variables(weights_store.get(inner_path))
677/677 ━━━━━━━━━━━━━━━━━━━━ 86s 122ms/step - accuracy: 1.0000 - loss: 8.2559e-07 - val_accuracy: 1.0000 - val_loss: 5.5065e-09
Epoch 2/5
677/677 ━━━━━━━━━━━━━━━━━━━━ 73s 108ms/step - accuracy: 1.0000 - loss: 1.0173e-08 - val_accuracy: 1.0000 - val_loss: 2.2621e-09
Epoch 3/5
677/677 ━━━━━━━━━━━━━━━━━━━━ 72s 107ms/step - accuracy: 1.0000 - loss: 4.8849e-09 - val_accuracy: 1.0000 - val_loss: 1.3754e-09
Epoch 4/5
677/677 ━━━━━━━━━━━━━━━━━━━━ 80s 104ms/step - accuracy: 1.0000 - loss: 3.2930e-09 - val_accuracy: 1.0000 - val_loss: 1.0158e-09
Epoch 5/5
677/677 ━━━━━━━━━━━━━━━━━━━━ 86s 110ms/step - accuracy: 1.0000 - loss: 2.5530e-09 - val_accuracy: 1.0000 - val_loss: 8.0055e-10
<keras.src.callbacks.history.History at 0x7ed87819e0b0>
```

```
bilstm_best_model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 100, 128)	2,560,000
bidirectional (Bidirectional)	(None, 64)	41,216
dropout (Dropout)	(None, 64)	0
dense (Dense)	(None, 128)	8,320
dense_1 (Dense)	(None, 1)	129

Total params: 7,828,997 (29.87 MB)
Trainable params: 2,609,665 (9.96 MB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 5,219,332 (19.91 MB)

```
[ ] # Evaluate BiLSTM Model
    test_loss, test_accuracy = bilstm_best_model.evaluate(X_test_pad, y_test)
    print(f"BiLSTM Model Test Accuracy: {test_accuracy:.4f}")
```

```
170/170 ━━━━━━━━━━━━━━━━━━━━ 3s 20ms/step - accuracy: 1.0000 - loss: 7.9719e-10
BiLSTM Model Test Accuracy: 1.0000
```

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
ANN Model	The ANN Model is more Accuracy for another three Model.

```
# Define the ANN model
def build_ann_model(hp):
    model = Sequential()
    model.add(Embedding(input_dim=20000, output_dim=hp.Int('embedding_dim', 64, 256, step=64), input_length=max_sequence_length))
    model.add(Flatten())
    model.add(Dense(units=hp.Int('dense_units1', 32, 256, step=32), activation='relu'))
    model.add(Dropout(hp.Float('dropout_rate1', 0.2, 0.5, step=0.1)))
    model.add(Dense(units=hp.Int('dense_units2', 32, 128, step=32), activation='relu'))
    model.add(Dropout(hp.Float('dropout_rate2', 0.2, 0.5, step=0.1)))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(optimizer=Adam(learning_rate=hp.Float('learning_rate', 1e-4, 1e-2, sampling='LOG')),
                  loss='binary_crossentropy',
                  metrics=['accuracy'])
    return model

# Hyperparameter tuning for ANN
tuner_ann = kt.RandomSearch(
    build_ann_model,
    objective='val_accuracy',
    max_trials=5,
    executions_per_trial=1,
    directory='ann_tuning',
    project_name='ann_model'
)
```

```
[ ] # Splitting training data into train and validation sets for ANN tuning
    X_train_split, X_val, y_train_split, y_val = train_test_split(X_train_pad, y_train, test_size=0.2, random_state=42)

[ ] # Run the tuner
    tuner_ann.search(X_train_split, y_train_split, epochs=5, validation_data=(X_val, y_val), class_weight=class_weights_dict)

    Trial 5 Complete [00h 02m 49s]
    val_accuracy: 1.0

    Best val_accuracy So Far: 1.0
    Total elapsed time: 00h 14m 01s

    # Get the best model
    ann_best_model = tuner_ann.get_best_models(num_models=1)[0]

    /usr/local/lib/python3.10/dist-packages/keras/src/saving/saving_lib.py:713: UserWarning: Skipping variable loading for optimizer 'adam', because it has 2 variables whereas the saved o
      saveable.load_own_variables(weights_store.get(inner_path))

[ ] # Train the best ANN model
    ann_best_model.fit(X_train_pad, y_train, epochs=5, validation_data=(X_val, y_val), class_weight=class_weights_dict)
```

```
[ ] # Evaluate the ANN model
    test_loss, test_accuracy = ann_best_model.evaluate(X_test_pad, y_test)
    print(f"ANN Model Test Accuracy: {test_accuracy:.4f}")

    170/170 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 1.0000 - loss: 3.7855e-24
    ANN Model Test Accuracy: 1.0000

[ ] # Generate predictions and classification report
    y_pred_probs_ann = ann_best_model.predict(X_test_pad)
    y_pred_ann = (y_pred_probs_ann > 0.5).astype(int)
    print("\nANN Classification Report:")
    print(classification_report(y_test, y_pred_ann))

    170/170 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step

    ANN Classification Report:
                  precision    recall  f1-score   support

               1       1.00      1.00      1.00      5416

        accuracy                           1.00      5416
       macro avg       1.00      1.00      1.00      5416
    weighted avg       1.00      1.00      1.00      5416
```

Conclusion

The project "Analysis of Amazon Cell Phone Reviews Using NLP Techniques" successfully demonstrates the power of Natural Language Processing (NLP) in deriving meaningful insights from unstructured customer review data.
By leveraging NLP methods such as sentiment analysis, topic modeling, and visualization techniques, the project achieves the following:

1. Understanding Customer Sentiments: Sentiment analysis provides a clear breakdown of customer opinions (positive, negative, or neutral) about cell phones. This helps businesses gauge customer satisfaction and identify areas for improvement in their products.

2. Identifying Key Themes: Topic modeling uncovers recurring themes and patterns in the reviews, highlighting common concerns, preferences, and expectations of customers. This helps manufacturers prioritize product enhancements and marketing strategies.

3. Actionable Insights: The visual representation of data, including sentiment distribution and thematic trends, makes it easier for stakeholders—such as businesses and consumers—to make informed decisions.

4. Scalable Approach: The framework developed in this project is adaptable and can be applied to other product categories or e-commerce platforms, making it a versatile solution for review analysis.

The project underscores the importance of utilizing advanced NLP techniques to transform raw, unstructured textual data into actionable insights. It also highlights how businesses can benefit from data-driven decision-making to improve product offerings, enhance customer satisfaction, and maintain a competitive edge.