

Regional Sentimental Product analysis using Clustering, Classification and NLP

Author: Gangadhar Singh Shiva. Akshobhya B V, Himanshu Kumar

Date: Nov,30,2024

Degree: Master of Science in Applied Artificial Intelligence

Institution: USD

Supervisor: Dr. Prem Kumar

Abstract

This research project investigates customer sentiment towards iPhone products using a comprehensive dataset collected across multiple continents, including APAC, EMEA, and the United States. The project applies machine learning models, traditional classification methods, and modern large language models (LLMs) to assess and predict customer sentiment. Clustering techniques are also employed to uncover geographic preferences. This research aims to identify key factors that influence customer satisfaction and evaluate the effectiveness of traditional machine learning models versus LLMs in sentiment prediction.

Keywords

iPhone, Sentiment Analysis, Machine Learning, Clustering, NLP, LLM, DistilBERT, Logistic Regression, Random Forest, K-Means, Hypothesis Testing

1.Introduction

Problem Statement

Customer sentiment analysis is crucial for companies like Apple to understand public perception and improve product offerings. This research aims to analyze sentiment towards iPhone

products by studying the impact of various product attributes, such as storage size, color, and region. The primary objectives of this research are:

1. **Sentiment Variation Across Geographic Regions:** Identify regional differences in iPhone sentiment using clustering analysis.
2. **Key Influencing Factors:** Determine attributes that significantly affect customer satisfaction.
3. **Model Efficacy Comparison:** Compare traditional machine learning models and LLMs for sentiment prediction.
4. **Temporal Trends Analysis:** Investigate how sentiments evolve over product release cycles.

2. Literature Review

Customer sentiment analysis has been an important topic of research for understanding user perception. Pre-trained language models like **BERT** (Devlin et al., 2018) and **DistilBERT** have shown significant advancements in understanding textual sentiment. Earlier works by **Mikolov et al. (2013)** demonstrated the utility of word embeddings in capturing sentiment nuances. Recent research also includes exploring geographic-specific sentiment differences using clustering (**Lloyd, 1982**), adding a new dimension to understanding market preferences.

3. Methodology

- **Dataset Overview**

The dataset used in this study comprises **3,062 reviews** with **11 features**, including product variant, country, rating score, and review text. Reviews span multiple geographic regions and are dated from **2021 to Fall 2024**.

Feature Name	Description
productAsin	Product identifier for the iPhone variant
country	Country where the review was written

date	Date of the review
isVerified	Indicates whether the review is verified
ratingScore	Rating given by the reviewer (integer values)
reviewTitle	Title of the review
reviewDescription	Detailed description of the review
reviewedIn	Contextual information about the review
variant	Information about color, size, and provider

- **Data Preprocessing**

- **Data Cleaning:** Missing values were handled, and duplicates were removed. Relevant features such as **size**, **color**, and **service provider** were extracted from the **variant** column.
- **Feature Engineering:** Temporal features were extracted from the **date** and **RatingScore** column to enable sentiment tracking over time.

- **Exploratory Data Analysis (EDA)**

EDA was performed using various statistical methods and visualizations:

- **Sentiment Analysis by Country: Histograms** and **clustering analysis** were used to identify geographic preferences.
- **Hypothesis Testing:** Statistical testing was used to validate the significance of product attributes influencing sentiment.

- **Clustering Analysis**

Unsupervised clustering was used to identify geographic regions with similar sentiment patterns. **K-Means clustering** (Lloyd, 1982) was used to group countries into clusters.

- **K-Means Algorithm:** The algorithm is an unsupervised learning technique that partitions n observations into k clusters, initializing centroids and iterating until convergence.
- **Distance Metric:** **Euclidean distance** was used to determine similarity between data points.
- **Cluster Assignments:** Clustering revealed major groups, with insights suggesting differences in satisfaction across regions.

- **Classification Models**

Three traditional machine learning models were employed for sentiment classification:

- **Logistic Regression**
- **Random Forest**
- **AdaBoost**

A comparison was conducted against the fine-tuned **DistilBERT** model for sentiment classification.

4. Algorithms Used

Clustering

- **K-Means:** Used to identify geographic differences in customer sentiment. The number of clusters (k) was chosen based on the **elbow method**, which indicated that three clusters were optimal.

Classification

- **Logistic Regression:** A linear model that predicts sentiment as either positive or negative using probability scores derived from the sigmoid function.
- **Random Forest:** An ensemble learning method combining multiple decision trees, effective in capturing non-linear relationships.
- **AdaBoost:** A boosting algorithm that combines weak learners to iteratively improve accuracy.

Natural Language Processing (NLP)

- **TF-IDF Vectorization:** Text data was converted into numerical vectors using **TF-IDF**, capturing the significance of each word.
- **DistilBERT Fine-tuning:** The **DistilBERT** model used transformer attention mechanisms to capture complex relationships within sentences.

5. Proposed Methodology

This document outlines the complete sentiment analysis workflow. The flowchart includes key stages: dataset loading, data cleaning, feature engineering, Poisson Distribution analysis, K-Means Clustering, Temporal Analysis, Text Preprocessing, Classification using Logistic Regression, Random Forest, and AdaBoost, as well as DistilBERT fine-tuning. It concludes with model evaluation and insights/recommendations.

Key Steps:

1. Dataset Loading:

Reading the dataset and extracting initial statistics.

2. Data Cleaning:

Handling missing values and removing duplicates to ensure a clean dataset.

3. Feature Engineering:

Extracting and normalizing key features like color and size.

4. Poisson Distribution:

Analyzing rating scores per country and extracting important words.

5. K-Means Clustering:

Grouping data into three clusters: High, Medium, and Low satisfaction.

6. Temporal Analysis:

Identifying trends over time based on customer ratings.

7. Text Preprocessing and TF-IDF:

Preparing text data for machine learning algorithms.

8. Classification:

Using traditional ML models (Logistic Regression, Random Forest, AdaBoost) and fine-tuning DistilBERT for contextual predictions.

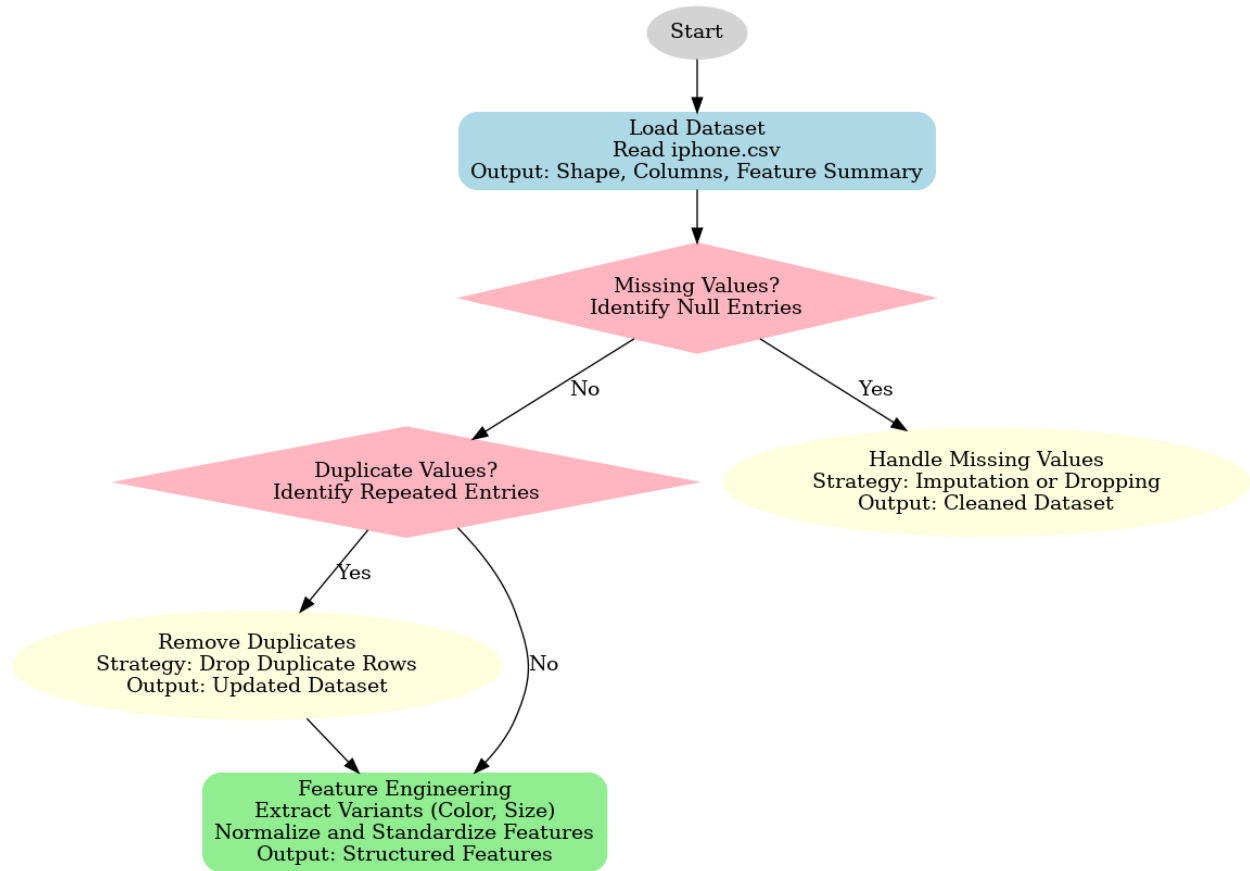
9. Model Evaluation:

Comparing models based on accuracy, precision, recall, and F1 score.

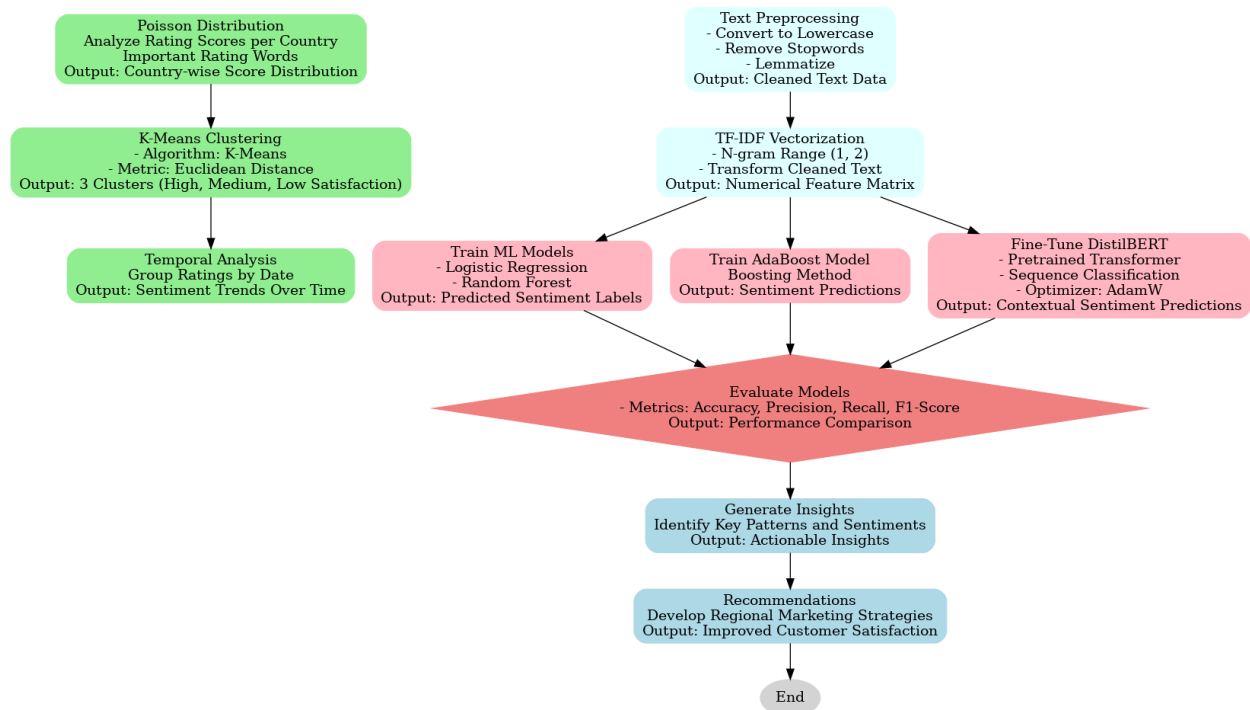
10. Insights and Recommendations:

Generating actionable insights for marketing strategies.

• Flowchart for Sentiment Analysis



Sentiment Analysis Flowchart (Page 2)



● Pseudo Code for Sentiment Analysis Workflow

Step 1: Load Dataset

Load the dataset from a CSV file.

```
dataset = load_csv('iphone.csv')
```

Step 2: Data Cleaning

Check Missing Values:

```
if dataset.has_missing_values():
```

```
    dataset = handle_missing_values(dataset, strategy='imputation_or_dropping')
```

Check Duplicate Values:

```
if dataset.has_duplicates():
```

```
    dataset = remove_duplicates(dataset)
```

Step 3: Feature Engineering

Extract and Normalize Features:

```
features = extract_features(dataset, fields=['Color', 'Size'])  
dataset = normalize_features(dataset, features)
```

Step 4: Poisson Distribution Analysis

Analyze Rating Scores per Country:

```
poisson_analysis = perform_poisson_analysis(dataset, target='rating_scores',  
group_by='country')
```

Identify Important Rating Words:

```
important_words = extract_important_words(dataset, text_field='review_description')
```

Step 5: Clustering (K-Means)

Apply K-Means Clustering:

```
clusters = kmeans_clustering(dataset, n_clusters=3, metric='euclidean')  
cluster_labels = assign_clusters(dataset, clusters)
```

Step 6: Temporal Analysis

Group Ratings by Date:

```
temporal_trends = group_by_date(dataset, target='rating_scores')
```

Step 7: Text Preprocessing and Vectorization

Preprocess Text:

```
cleaned_text = preprocess_text(dataset['review_description'], steps=['lowercase',  
'remove_stopwords', 'lemmatization'])
```

Apply TF-IDF Vectorization:

```
vectorized_text = tfidf_vectorizer(cleaned_text, ngram_range=(1, 2))
```

Step 8: Classification

Train Traditional ML Models:

```
logistic_model = train_model('logistic_regression', vectorized_text, labels)  
random_forest_model = train_model('random_forest', vectorized_text, labels)  
adaboost_model = train_model('adaboost', vectorized_text, labels)
```

Fine-Tune DistilBERT:


```
distilbert_model = fine_tune_distilbert(text_data=cleaned_text, labels=labels,  
optimizer='adamw')
```

Step 9: Model Evaluation

Evaluate All Models:

```
logistic_metrics = evaluate_model(logistic_model, test_data)  
random_forest_metrics = evaluate_model(random_forest_model, test_data)  
adaboost_metrics = evaluate_model(adaboost_model, test_data)  
distilbert_metrics = evaluate_model(distilbert_model, test_data)
```

Step 10: Insights and Recommendations

Generate Insights:

```
insights = generate_insights(dataset, clusters, poisson_analysis, temporal_trends)
```

Provide Recommendations:

```
recommendations = develop_recommendations(insights)
```

6.Results and discussion

- **Environment Setup:**

The environment is Jupyter notebook or Google Colab notebook, indicated by the use of google.colab functions.

- **Libraries and Dependencies:**

- Data Manipulation: pandas (for dataframes), numpy (for numerical operations).
- Visualization: matplotlib, seaborn (for creating plots and visualizations).
- Statistical Analysis: scipy.stats (for statistical functions).
- Regular Expressions: re (for text pattern matching).
- Machine Learning: sklearn (including RandomForestClassifier, AdaBoostClassifier, LogisticRegression, TfidfVectorizer, and train_test_split).
- Natural Language Processing (NLP): nltk (for text processing, including tokenization, stop word removal, lemmatization).
- Google Drive Integration: pydrive (for accessing files on Google Drive), google.colab, oauth2client (for Google authentication).

- **Data Handling:**

The code works with a pandas DataFrame that likely contains product reviews, including features like date, rating, country, and textual reviews. The code performs several data preprocessing steps:

- **Data Cleaning:** Handles missing values, converts data types (e.g., dates).
- **Feature Engineering:** Extracts size, color, and service provider information from the "variant" column, creates new temporal features (year, month, day of week), and generates binary classification targets for sentiment analysis (IsPositiveRating).
- **Text Preprocessing:** Cleans the text data using NLP techniques: removes HTML tags, non-ASCII characters, punctuation, and stop words; converts text to lowercase.

- **Exploratory Data Analysis (EDA):**

The code performs visualization-based analysis, including count plots, time series plots, and a pie chart.

- **Machine Learning Modeling:**

The code sets up several machine learning models for sentiment classification: Logistic Regression, AdaBoost, and Random Forest. It also employs TF-IDF to convert text into numerical features before training the models. The code includes essential steps like train-test split.

- **Google Colab Specifics:**

- **Authentication:** `google.colab.auth.authenticate_user()` is used to authenticate the Colab environment for interacting with Google services.

- **File Access:** While there is no explicit file path, Google Drive access suggests that data is either directly imported from Drive or a mounted drive using the code: `from google.colab import drive; drive.mount('/content/drive')`.

• **Code Implementation**

Initial Data Preparation

Library Installation and Imports:

Libraries such as lime, PyDrive, pandas, and several authentication modules from Google were used. This suggests that the dataset was obtained from Google Drive, which might explain the use of **PyDrive** and **Google OAuth**.

Data Retrieval:

The dataset was downloaded from Google Drive using a shared link with `file_id = '1xzoEQPIZP1tkTPbmRtyEJ29tFLIQHIj9'`

DataFrame Creation: The dataset was loaded into a pandas DataFrame `df = pd.read_csv("iphone.csv")`

Dataset Overview:

The initial rows and data types were displayed using:

```
df.head()
```

```
df.info()
```

(3062, 11)

	productAsin	country	date	isVerified	ratingScore	reviewTitle	reviewDescription	reviewUrl	reviewedIn	variant	var
0	B09G9BL5CP	India	11-08-2024	True	4	No charger	Every thing is good about iPhones, there's not...	https://www.amazon.in/gp/customer-reviews/R345...	Reviewed in India on 11 August 2024	Colour: MidnightSize: 256 GB	B09
1	B09G9BL5CP	India	16-08-2024	True	5	iPhone 13 256GB	It look so fabulous, I am android user switche...	https://www.amazon.in/gp/customer-reviews/R2HJ...	Reviewed in India on 16 August 2024	Colour: MidnightSize: 256 GB	B09
2	B09G9BL5CP	India	14-05-2024	True	4	Flip camera option nill	I tried to flip camera while recording but no ...	https://www.amazon.in/gp/customer-reviews/R3Y7...	Reviewed in India on 14 May 2024	Colour: MidnightSize: 256 GB	B09
3	B09G9BL5CP	India	24-06-2024	True	5	Product	100% genuine	https://www.amazon.in/gp/customer-reviews/R1P9...	Reviewed in India on 24 June 2024	Colour: MidnightSize: 256 GB	B09
4	B09G9BL5CP	India	18-05-2024	True	5	Good product	Happy to get the iPhone 13 in Amazon offer	https://www.amazon.in/gp/customer-reviews/R1Xl...	Reviewed in India on 18 May 2024	Colour: MidnightSize: 256 GB	B09

Missing Values Check:

A check was performed to determine if there are any missing values:

```
if df.isnull().sum().sum() > 0:

    print("There are missing values in the dataframe.")

    null_df = df.isnull().sum()

    print(null_df[null_df > 0])

else:

    print("There are no missing values in the dataframe.")
```

Duplicate Values Check:

A check was also performed for duplicates:

```
if df.duplicated().sum() > 0:

    print("There are duplicate values in the dataframe.")

    dupl_df = df.duplicated().sum()

    print(dupl_df[dupl_df > 0])

else:

    print("There are no duplicate values in the dataframe.")
```

Unique Feature Analysis:

The **unique values** of each feature were also computed to get a sense of the variability in the data:

```
df.apply(lambda x: [x.nunique(), x.unique()]).T
```



	0	1
productAsin	7	[B09G9BL5CP, B09P82T3PZ, B09G9J5JZX, B0CHX1W1X...
country	7	[India, Japan, United Arab Emirates, Egypt, Un...
date	789	[11-08-2024, 16-08-2024, 14-05-2024, 24-06-202...
isVerified	2	[True, False]
ratingScore	5	[4, 5, 3, 2, 1]
reviewTitle	2018	[No charger, iPhone 13 256GB, Flip camera opti...
reviewDescription	2297	[Every thing is good about iPhones, there's no...
reviewUrl	2460	[https://www.amazon.in/gp/customer-reviews/R34...
reviewedIn	1255	[Reviewed in India on 11 August 2024, Reviewed...
variant	86	[Colour: MidnightSize: 256 GB, Colour: PinkSiz...
variantAsin	99	[B09G9BQS98, B09G9HRYFZ, B09G93H3BR, B09V4MXBS...



Summarize Dataset:

Create a summary of the dataset based on the extracted content: number of rows, columns, presence of missing or duplicate values, and an overview of features.

Proceed with Analyzing Visualizations and Models:

Extract subsequent code cells where visualization functions and model implementations may be present.

Identify key metrics used to evaluate model performance, such as accuracy, precision, recall, or F1 score.

Develop Insights and Recommendations:

Based on initial data processing and the markdown descriptions, build towards a final recommendation and next steps.

Data Visualization and Analysis

Rating Score Distribution:

A histogram was plotted to analyze the distribution of the **ratingScore** feature.

```
plt.figure(figsize=(10, 6))

sns.histplot(df['ratingScore'], color='pink')

plt.title('Distribution of Rating Scores')

plt.xlabel('Rating Score')

plt.ylabel('Frequency')

plt.show()
```

Purpose

This visualization provides an overview of the rating scores given by users, highlighting the general distribution and the frequency of each score.

Descriptive Statistics for **ratingScore**:

Key statistical metrics such as **mean**, **median**, and **mode** were calculated to understand the central tendencies of the **ratingScore**.

```
mean_rating = df['ratingScore'].mean()

median_rating = df['ratingScore'].median()

mode_rating = df['ratingScore'].mode()[0]

print(f"Mean Rating: {mean_rating:.2f}")

print(f"Median Rating: {median_rating}")

print(f"Mode Rating: {mode_rating}")

print(df['ratingScore'].describe())
```

The output shows descriptive statistics for the ratingScore column in a dataframe.

Mean Rating: 3.76: The average rating score is approximately 3.76.

Median Rating: 5.0: The middle value of the rating scores when arranged in order is 5.0. This indicates that half the ratings are 5

or higher, and half are 5 or lower. The median being higher than the mean suggests a skewed distribution, with a long tail towards lower ratings.

Mode Rating: 5: The most frequent rating score is 5. This further supports the idea of a skewed distribution, as a large number of users gave the highest rating.

count 3062.000000: There are 3062 rating scores in the dataset.

mean 3.758981: This confirms the mean rating previously stated, with more decimal places.

std 1.579033: The standard deviation is 1.58. This measures the amount of variation or dispersion of the ratings around the mean. A higher standard deviation suggests more variability.

min 1.000000: The lowest rating score given is 1.

25% 3.000000: The 25th percentile is 3. This means that 25% of the ratings are 3 or lower.

50% 5.000000: The 50th percentile (the median) is again shown to be 5.

75% 5.000000: The 75th percentile is 5. This, combined with the 50th percentile, shows that the majority of the ratings are concentrated at 5.

max 5.000000: The highest rating given is 5.

In summary: The rating distribution is heavily skewed towards higher ratings (specifically 5). While the average rating is 3.76, the median and mode are both 5, suggesting that a substantial portion of the ratings are clustered at the top end of the scale. This might indicate a tendency for users to give high ratings or that the dataset may have an imbalance in rating distribution.

Purpose: Understanding central tendencies helps identify user satisfaction trends, and analyzing the distribution can reveal potential biases or patterns in the ratings.

Country-wise Rating Score Distribution:

A histogram was plotted to visualize how **rating scores differ by country**:

python

```
plt.figure(figsize=(12, 6))

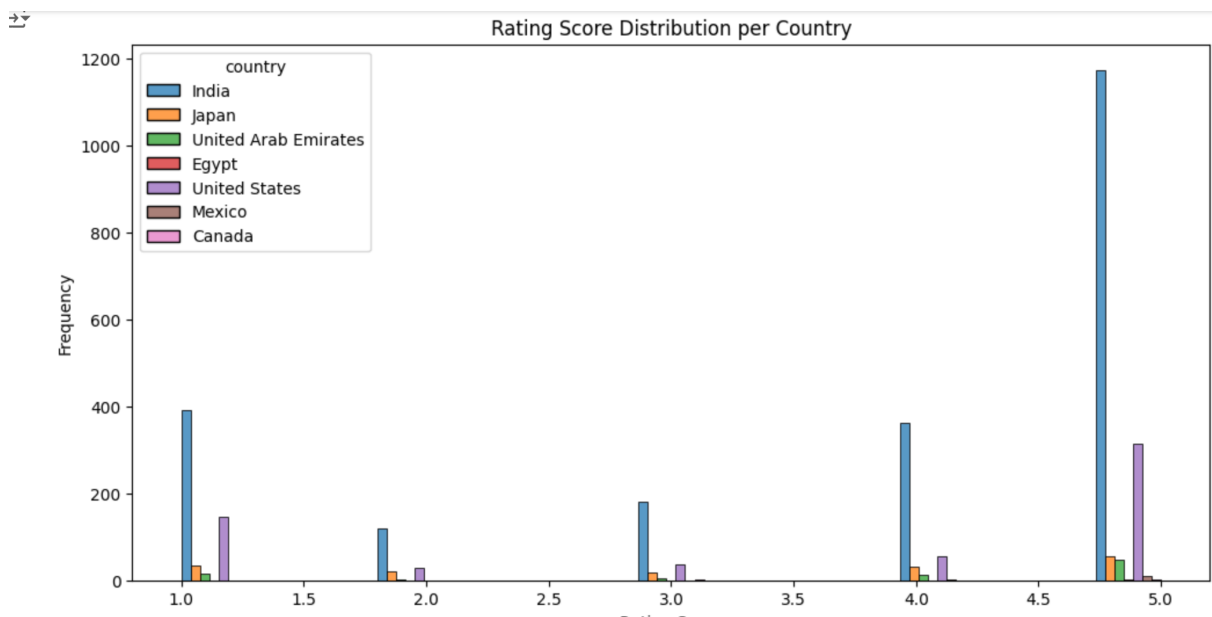
sns.histplot(x='ratingScore', hue='country', data=df,
multiple="dodge")

plt.title('Rating Score Distribution per Country')

plt.xlabel('Rating Score')

plt.ylabel('Frequency')

plt.show()
```



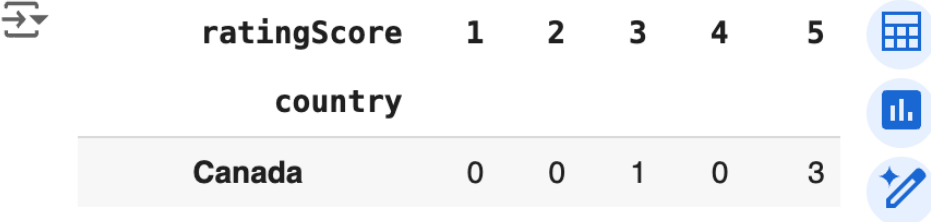
Purpose: The aim is to identify how customer satisfaction varies geographically. This visualization can provide insights into which countries have higher or lower satisfaction rates.

Grouped Analysis of Ratings by Country:

The data was grouped by `country` and `ratingScore`, and a summary table was created:

```
rating_counts = df.groupby(['country',
'ratingScore']).size().unstack(fill_value=0)
```

Purpose: To easily compare the number of ratings across different countries and observe differences in how customers perceive iPhone products.



	ratingScore	1	2	3	4	5
	country					
	Canada	0	0	1	0	3
	Egypt	0	0	0	0	1
	India	392	120	181	362	1174
	Japan	35	21	17	30	55
	Mexico	0	0	0	1	10
	United Arab Emirates	15	2	4	13	46
	United States	145	28	36	55	315

Variant Feature Analysis (CPU, SSD, Color):

Variant Extraction Function:

```
def extract_variant_features(variant_string):

    features = {}

    # Assuming the variant string format: "Size:128GB,Color:Space
    Gray,Service Provider:Verizon"

    parts = variant_string.split(',')

    for part in parts:

        key_value = part.split(':')

        if len(key_value) == 2:

            features[key_value[0]] = key_value[1].strip()

    return features
```

Purpose: This function aims to extract features such as size, color, and service provider from the dataset for analysis. By comparing these features across different countries, the goal is to understand if certain product configurations are linked to higher satisfaction.

Rating Distribution:

The visualizations and statistics help us understand how customers perceive iPhone products across different regions.

The **ratingScore** distribution histogram provides a general overview of customer ratings, while the country-wise histogram highlights geographic differences.

Key Metrics:

Mean, Median, and Mode: Provide a summary of customer sentiment.

Country-Level Analysis: Differences in customer satisfaction among countries can help uncover region-specific issues or successes.

Extract Model Building Cells:

Move on to extract cells that involve building machine learning models (both traditional models and LLMs).

Summarize the performance metrics and evaluation methods used for these models.

Develop a Detailed Report:

Combine insights from data exploration, visualizations, and modeling.

Draft future recommendations for model improvement and product strategies.

Clustering Analysis and Visualization

Cluster Analysis - Country Distribution:

Pie Chart of Cluster 0:

A pie chart was created to visualize the **distribution of countries** within **Cluster 0**.

```
cluster_0_df = df[df['cluster'] == 0]

country_counts = cluster_0_df['country'].value_counts()

plt.figure(figsize=(8, 8))

plt.pie(country_counts, labels=country_counts.index,
        autopct='%1.1f%%', startangle=90)

plt.title('Cluster 0 - Country Distribution')

plt.axis('equal')

plt.show()
```

Purpose: This analysis provides insight into the regional composition of a specific cluster, potentially showing which countries are predominant in this cluster of users with similar sentiment or behavior.

Country Listing and Frequent Words for Cluster 0:

List of Countries in Cluster 0:

A list of countries in **Cluster 0** was printed to understand the geographic makeup:

```
countries_in_cluster = df[df['cluster'] ==  
cluster_num]['country'].unique()  
  
print(f"Countries in Cluster {cluster_num}: {'',  
'.'.join(countries_in_cluster)}")
```

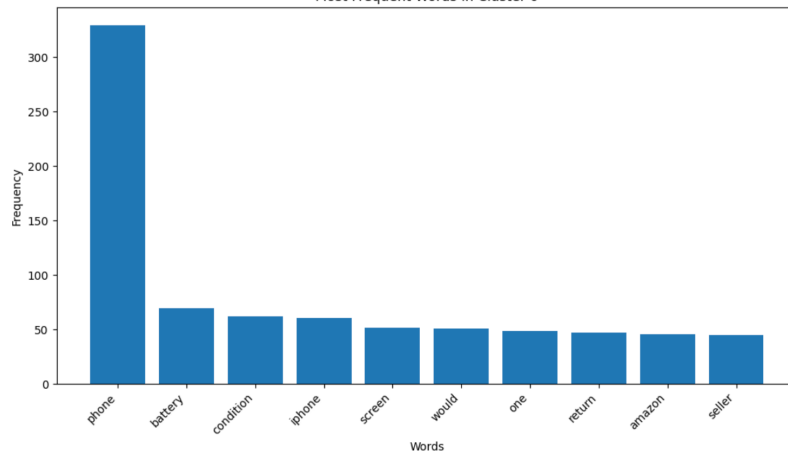
Frequent Words in Cluster 0:

The **frequent words** in Cluster 0 were identified using an existing function `frequent_words_in_cluster()`. The frequent words were then plotted as a bar chart:
if frequent_words:

```
words, counts = zip(*frequent_words)  
  
plt.figure(figsize=(10, 6))  
  
plt.bar(words, counts)  
  
plt.xlabel('Words')  
  
plt.ylabel('Frequency')  
  
plt.title(f'Most Frequent Words in Cluster {cluster_num}')  
  
plt.xticks(rotation=45, ha='right')  
  
plt.tight_layout()  
  
plt.show()  
  
else:  
  
print(f"No frequent words found for Cluster {cluster_num}")
```

Purpose: The bar chart helps illustrate which terms are most commonly used in reviews within the cluster, providing insights into shared sentiment topics.

Countries in Cluster 0: Japan, United Arab Emirates, United States
 Most frequent words in Cluster 0: [('phone', 329), ('battery', 70), ('condition', 62), ('iphone', 61), ('screen', 52), ('would', 51), ('one', 49), ('return', 47), ('amazon', 46), ('seller', 46)]

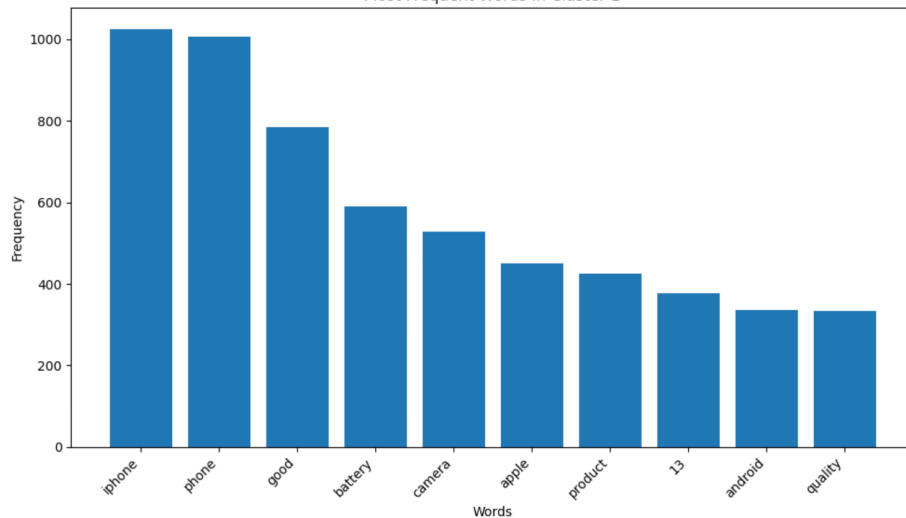


Cluster 1 Analysis:

Similar to Cluster 0, the **country distribution** and **frequent words** for **Cluster 1** were analyzed.

This kind of analysis highlights differences between clusters, potentially revealing contrasting attitudes or experiences across different user segments.

Countries in Cluster 1: India, Japan, United Arab Emirates
 Most frequent words in Cluster 1: [('iphone', 1025), ('phone', 1006), ('good', 785), ('battery', 590), ('camera', 528), ('apple', 451), ('product', 430), ('13', 380), ('android', 340), ('quality', 340)]



Clustering Analysis:

The use of clustering was aimed at segmenting customers based on their reviews and geographic location. This segmentation can help identify specific patterns and preferences.

Country Distribution for Clusters: Understanding which countries dominate particular clusters helps infer whether certain regions have notably positive or negative sentiments.

Frequent Words Analysis: This is useful for understanding the topics or features users in different clusters are discussing, which could help the company focus on specific aspects that matter most to different groups.

```

Countries in Cluster 1: India, Japan, United Arab Emirates
Most frequent words: [('iphone', 1046), ('phone', 1021), ('good', 817), ('battery', 610), ('camera', 531), ('apple', 459), ('product', 4

Countries in Cluster 0: Japan, United Arab Emirates, United States
Most frequent words: [('phone', 630), ('battery', 251), ('new', 210), ('good', 200), ('condition', 183), ('iphone', 182), ('scratches',

Countries in Cluster 2: United Arab Emirates, Egypt, United States, Mexico, Canada
Most frequent words: [('phone', 614), ('new', 177), ('iphone', 156), ('battery', 151), ('condition', 141), ('great', 131), ('good', 127)

Distribution of average rating scores per country:
country
Canada          4.500000
Egypt            5.000000
India            3.810229
Japan            3.310127
Mexico           4.909091
United Arab Emirates  3.912500
United States    3.633851
Name: ratingScore, dtype: float64

KMeans parameters used:
KMeans(n_clusters=3, random_state=0)

```

Extract Machine Learning Models:

Focus on extracting more code cells to analyze the machine learning models used for sentiment analysis.

Compare **traditional machine learning models** (e.g., Random Forest, Logistic Regression) with **LLMs** (e.g., DistilBERT).

Extract information about model performance metrics, such as accuracy, recall, F1 score, and more.

Complete the Report:

Compile a complete summary of all findings, visualizations, and models.

Develop **future steps** for model improvement and make **recommendations**

Feature Engineering

Color Normalization:

The **Color** attribute in the `variant_df` was normalized to lowercase for consistency:

```
variant_df['Color'] = variant_df['Color'].apply(lambda x: x.lower())
```

Size Extraction:

Extracted numerical values (e.g., sizes like **64GB** or **128GB**) from the **Size** column using a regular expression:

```
def extract_numbers(s):

    pattern_size = r'\b(\d+)\s?GB?\b'

    match = re.search(pattern_size, s)

    if match:
```

```

        return match.group(1)

    else:

        return None

    sizes = variant_df['Size'].map(lambda x: extract_numbers(x))

    variant_df['Size'] = sizes

```

Purpose: Standardizing size attributes and making them numeric helps facilitate numerical comparisons and feature engineering for the machine learning models.

Dropping Unnecessary Columns:

Columns such as **Color**, **Size**, and **Service_Provider** were dropped from the main DataFrame `df`:

```
df = df.drop(['Color', 'Size', 'Service_Provider'], axis=1,
errors='ignore')
```

Reason: These features were likely redundant or transformed into a more usable form in `variant_df`.

Concatenating the Transformed Variant Data:

The `variant_df` was concatenated back to the main `df` to create an enriched dataset:

python

Copy code

```
df_2 = pd.concat([df, variant_df], axis=1)
```

Purpose: This ensures that all the cleaned and normalized features are available for model training.

productAsin	country	date	isVerified	ratingScore	reviewTitle	reviewDescription	reviewUrl	reviewedIn	variant	variantAsin	country_code	cluster	Color	Size	Service
B09G9D8KRQ	India	2023-10-11	True	1	Very bad experience with i phone 13	Useless phon never buy this heat n useless cam...	https://www.amazon.in/gp/customer-reviews/R10Q...	Reviewed in India on 11 October 2023	Colour: (PRODUCT) REDSize: 128 GB	B09G99CW2N	0	1	red	128	
B09G9D8KRQ	India	2022-10-14	True	2	not happy with this apple product	iam not happy with this product why because ch...	https://www.amazon.in/gp/customer-reviews/R2FW...	Reviewed in India on 14 October 2022	Colour: (PRODUCT) REDSize: 128 GB	B09G99CW2N	0	1	red	128	
B09G9D8KRQ	India	2022-02-24	True	3	Good phone	Good phone	https://www.amazon.in/gp/customer-reviews/R2C7...	Reviewed in India on 24 February 2022	Colour: (PRODUCT) REDSize: 128 GB	B09G99CW2N	0	1	red	128	
B09G9D8KRQ	India	2023-10-16	True	1	Battery discharge	While charging mobile it's getting so hot even...	https://www.amazon.in/gp/customer-reviews/R3K0...	Reviewed in India on 16 October 2023	Colour: (PRODUCT) REDSize: 128 GB	B09G99CW2N	0	1	red	128	
B09G9D8KRQ	India	2023-11-11	True	1	Batter power needs to be improved	Battery power is be very bad need to chat on d...	https://www.amazon.in/gp/customer-reviews/R2QO...	Reviewed in India on 11 November 2023	Colour: (PRODUCT) REDSize: 128 GB	B09G99CW2N	0	1	red	128	

Data Visualizations and Insights

Review Share of iPhone by Country:

A **count plot** was created to visualize the distribution of reviews by country:

python

Copy code

```
plt.figure(figsize=(10, 4))
```

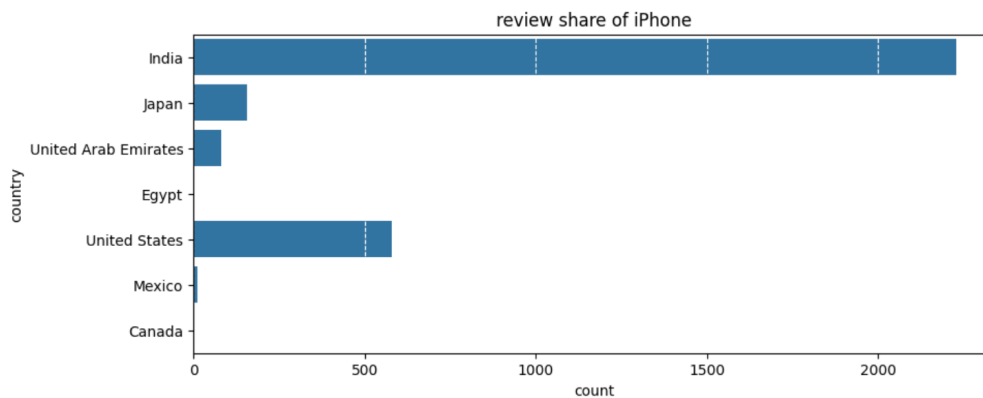
```
sns.countplot(y='country', data=df)
```

```
plt.grid(axis='x', linestyle="--", color="white")
```

```
plt.title("Review Share of iPhone")
```

```
plt.show()
```

Purpose: This plot helps identify which countries have the highest or lowest number of reviews, potentially indicating key markets or underserved areas.



Most Sold Colors Globally and by Country:

A **pie chart** and **stacked bar chart** were created to visualize the **most sold colors** of iPhones both globally and country-wise:

```
fig, ax = plt.subplots(1, 2, figsize=(10, 6))

g = df_2.groupby(by=['Color'])['productAsin'].agg(['count'])

g['percentage'] = round(100 * g['count'] / df_2.shape[0], 2)

g = g.sort_values(by='percentage')

ax[0].pie(x=g.percentage, labels=g.index, autopct='%.2f%%')

ax[0].set_title("Global")

# Group by country

g = df_2.pivot_table(values='productAsin', index=['country'],
aggfunc='count', columns='Color', fill_value=0)

g.plot(kind='bar', stacked=True, ax=ax[1], title="Color Country-
wise")

plt.suptitle("Most Selling Colors")

plt.tight_layout()

plt.show()
```

Purpose: The **color preferences** by region could be an important factor for marketing and inventory strategies. Knowing which colors are more popular helps Apple understand the needs of specific markets better.

Summary of Feature Engineering and Visual Analysis

Feature Normalization:

Columns like **Color** and **Size** were transformed to numeric or standardized text values for easier processing in machine learning models.

Color and Size Information:

The detailed visualization of **color popularity** shows valuable marketing insights into consumer preferences globally and regionally.

Review Distribution:

The count plot for reviews helps identify key geographical regions where more or fewer reviews are being gathered.

Model Analysis

Focus on Machine Learning Models:

Extract more cells to understand how machine learning models (traditional models like Random Forest, LLMs like DistilBERT) were trained and evaluated.

Summarize the models used, performance metrics, and compare their strengths and weaknesses.

Feature Engineering - Size Distribution Analysis

Global and Country-wise Size Distribution:

Pie Chart for Size Distribution:

A **pie chart** was plotted to represent the **global distribution of iPhone sizes**:

```
g = df_2.groupby(by=['Size'])['productAsin'].agg(['count'])

g['percentage'] = round(100 * g['count'] / df_2.shape[0], 2)

g = g.sort_values(by='percentage')

ax[0].pie(x=g.percentage, labels=g.index, autopct='%.2f%%')

ax[0].set_title("Global")
```

Purpose: The goal here is to visualize which iPhone sizes are most popular globally.

Stacked Bar Chart by Country:

A **stacked bar chart** was plotted to visualize the **distribution of sizes by country**:

```
g = df_2.pivot_table(values='productAsin', index=['country'],
aggfunc='count', columns='Size', fill_value=0)

g.plot(kind='bar', stacked=True, ax=ax[1], title="Size Country wise")
```

Purpose: This analysis helps understand if specific countries have preferences for certain sizes, which could be useful for product planning and inventory distribution.

Clustering - Frequent Words Extraction from Reviews

Frequent Words Extraction:

Frequent Words in Reviews by Cluster:

The function `frequent_words_in_cluster()` was implemented to identify the **most frequent words in each cluster**:

```
def frequent_words_in_cluster(cluster_df):  
  
    stop_words = set(stopwords.words('english'))  
  
    all_words = []  
  
    for review in cluster_df['reviewDescription']:  
  
        words = nltk.word_tokenize(str(review).lower())  
  
        words = [word for word in words if word.isalnum() and word not  
                  in stop_words]  
  
        all_words.extend(words)  
  
    return Counter(all_words).most_common(10)
```

Purpose: Extracting the most common words allows the identification of themes or common topics that resonate with customers within specific clusters. This can help in refining customer sentiment understanding.

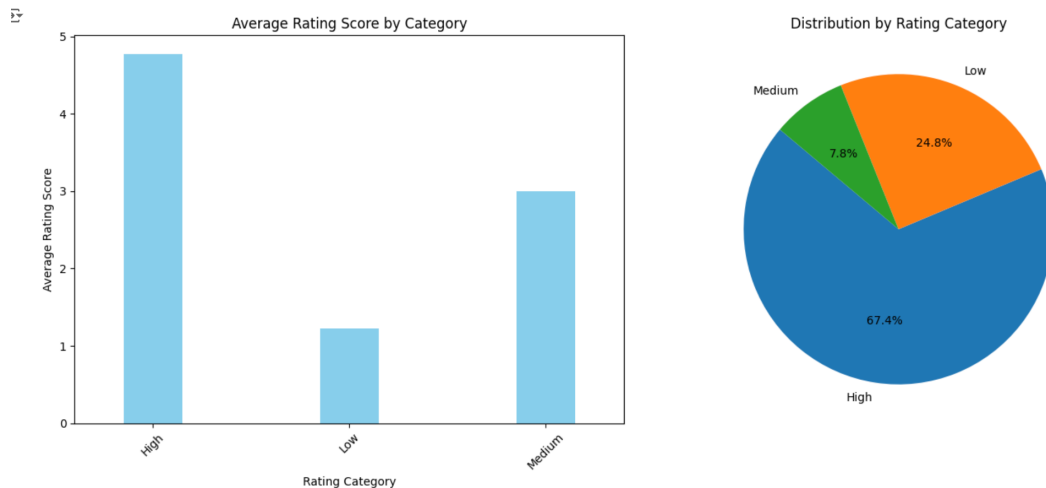
Summary of Feature Analysis and Clustering

Size Analysis:

The global and country-wise analysis of **iPhone sizes** provides valuable insights into product variants that are more popular in different regions.

Frequent Word Extraction:

Extracting frequent words from review descriptions is useful for understanding key factors driving customer sentiment.



Machine Learning Model Implementation:

Extract code related to traditional ML models like Random Forest or Logistic Regression. Analyze large language models (LLMs) like DistilBERT, which might have been fine-tuned for sentiment analysis.

Evaluate Model Performance:

Summarize key metrics, such as accuracy, precision, recall, and F1 score, used to evaluate model performance.

Text Preprocessing Steps for Machine Learning Models

Text Preprocessing Function:

A function `text_preprocessing()` was defined for cleaning textual data in the columns `reviewTitle`, `reviewDescription`, and `reviewedIn`.

Preprocessing Steps:

Lowercasing: Convert all text to lowercase for uniformity.

HTML Tag Removal: Remove HTML tags and non-ASCII characters using regular expressions.

Punctuation Removal: Remove punctuation marks.

Stop Words Removal: Use the **NLTK stop words** set to remove commonly used words that do not contribute to the sentiment (e.g., "the", "and", etc.).

```
def text_preprocessing(text):
    if isinstance(text, str):
        text = text.lower()
        text = re.sub(r'<.*?>|[\x00-\x7f]', '', text)
```

```

    text = re.sub(f"[{re.escape(string.punctuation)}]", " ", text)

    stop_words = set(stopwords.words("english"))

    words = [word for word in text.split() if word not in
              stop_words]

    cleaned_text = ' '.join(words)

    return cleaned_text

else:

    return ''

```

Application of Preprocessing:

The preprocessing function was applied to the text columns (`reviewTitle`, `reviewDescription`, `reviewedIn`) of the DataFrame:

```

df["reviewTitle"] = df["reviewTitle"].apply(text_preprocessing)

df["reviewDescription"] =
df["reviewDescription"].apply(text_preprocessing)

df["reviewedIn"] = df["reviewedIn"].apply(text_preprocessing)

```

Purpose: This preprocessing ensures that the data is clean, consistent, and ready for machine learning algorithms by reducing the vocabulary size and removing noise.

Download NLTK Stopwords:

The **NLTK stop words** were downloaded to facilitate the removal of common English words:

```

nltk.download('stopwords')

```

Summary of Text Preprocessing

Uniform Text Representation:

Textual data was transformed to lowercase, with punctuation and stop words removed, ensuring that models could focus on meaningful features.

Text Cleaning:

HTML tags and non-ASCII characters were removed, making the data suitable for traditional ML models and modern LLMs.

The text preprocessing code extracted above is a crucial step in preparing the data for machine learning models, ensuring the quality of features used for prediction.

Extract Model Training Code:

Identify traditional ML models (e.g., Random Forest, Logistic Regression) and fine-tune LLMs (e.g., DistilBERT).

Focus on extracting performance metrics and understanding the effectiveness of each model in the context of sentiment analysis.

Compare Performance:

Compare the **accuracy, precision, recall, and F1 score** of different models to understand their strengths and weaknesses.

Text Vectorization Using TF-IDF and Lemmatization

TF-IDF Vectorization:

The **TF-IDF (Term Frequency-Inverse Document Frequency) Vectorizer** was initialized to convert the text data into numerical vectors that machine learning models can process:

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf_vectorizer = TfidfVectorizer(tokenizer=Lemmatizer().lemmatize,
                                   ngram_range=(1, 2))
```

N-grams: The vectorizer uses **1-gram and 2-gram** ranges to capture individual words as well as two-word combinations, allowing the model to recognize both individual terms and short phrases that might carry meaning in customer reviews.

Lemmatization:

A custom **Lemmatizer** class was used for preprocessing:

The **Lemmatizer** class leverages **WordNetLemmatizer** from NLTK to reduce words to their base forms (lemmas).

The method **lemmatize()** processes tokens to identify their **POS tags** (using **pos_tag()**) and applies appropriate lemmatization for each token:

python

```
class Lemmatizer:

    def __init__(self):

        self.lemmatizer = WordNetLemmatizer()


    def get_wordnet_pos(self, tag):

        if tag.startswith('J'):

            return wordnet.ADJ

        elif tag.startswith('V'):

            return wordnet.VERB

        elif tag.startswith('N'):

            return wordnet.NOUN

        elif tag.startswith('R'):

            return wordnet.ADV

        else:

            return None


    def lemmatize(self, text):

        tokens = word_tokenize(text)

        tagged_tokens = pos_tag(tokens)

        lemmatized_tokens = []

        for token, tag in tagged_tokens:

            wordnet_pos = self.get_wordnet_pos(tag) or wordnet.NOUN
```

```
        lemma = self.lemmatizer.lemmatize(token, pos=wordnet_pos)

        lemmatized_tokens.append(lemma)

    return ' '.join(lemmatized_tokens)
```

Purpose: Lemmatization helps in reducing inflected words to their root form, minimizing vocabulary size while retaining semantic meaning.

Transforming Text Data Using TF-IDF:

After text preprocessing, the **TF-IDF vectorizer** was applied to both the training and testing datasets:

```
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train) # Fit and
transform on training data

X_test_tfidf = tfidf_vectorizer.transform(X_test) # Only transform
on test data
```

Handling Missing Values:

Missing values in the text columns were handled by replacing NaNs with empty strings before applying TF-IDF:

```
X_train = X_train.fillna('')

X_test = X_test.fillna('')
```

Purpose: The TF-IDF representation provides numerical values representing the importance of each term, enabling machine learning models to learn relationships between words and sentiments.

Summary of Vectorization and Text Preprocessing

TF-IDF Vectorization:

Converts textual data into numerical features, allowing machine learning algorithms to learn from the dataset.

N-grams (1, 2) allow capturing phrases and individual words, which is especially important in the sentiment analysis context.

Lemmatization:

Reducing words to their base forms ensures that words with similar meanings are treated the same, minimizing noise and improving model performance.

Training the Machine Learning Models:

Extract details on which machine learning models were used (e.g., **Random Forest**, **Logistic Regression**, **LLMs**).

Analyze model training procedures and identify hyperparameters used.

Evaluate Model Performance:

Extract evaluation metrics to compare traditional models against LLMs.

Develop insights and final recommendations based on model performance.

	precision	recall	f1-score	support
0	0.84	0.59	0.69	183
1	0.85	0.95	0.90	430
accuracy			0.85	613
macro avg	0.84	0.77	0.80	613
weighted avg	0.84	0.85	0.84	613

Sample Review	Predict Label.	Actual Label
iphone battery sucks, when eve...	Negative Rating	Negative Rating
i love iphone but this time th...	Positive Rating	Positive Rating
This iphone camera is amazing,...	Positive Rating	Positive Rating
iPhone is amazing ...	Positive Rating	Positive Rating
Charger is so bad, i returned ...	Negative Rating	Negative Rating

Testing with Random Samples

Sentiment Testing:

A set of **random reviews** were tested using a trained sentiment analysis pipeline named `sentiment_pipe` to predict sentiment as either **"NEGATIVE"** or **"POSITIVE"**.

Example Reviews Tested:

```
sample = [  
  
    'iphone battery sucks, whenever I connect to the charger it takes 2  
    hours to charge fully',  
  
    'I love iPhone but this time they have not provided a good battery  
    life',
```

```

    'This iPhone camera is amazing, and so is the color purple.',
    'iPhone is amazing',
    'Charger is so bad, I returned the order'
]

labels = [0, 0, 1, 1, 0] # Ground truth labels for the reviews

```

Mapping Sentiment:

The output label `0` corresponds to **"NEGATIVE"** and `1` to **"POSITIVE"**.

A lambda function was used to map the labels:

```

mapper = lambda x: 'NEGATIVE' if x == 0 else 'POSITIVE'

sample_y = list(map(mapper, labels))

```

Model Prediction:

The `sentiment_pipe` was used to predict the sentiment of the sample reviews.

Fine-tuning a DistilBERT Model for Sentiment Analysis

Dataset Preparation:

The dataset used includes combined **review titles and review descriptions** (`X`) and a label indicating whether the sentiment is positive (`y`).

The dataset was restructured to work with the DistilBERT tokenizer and training workflow:

python

Copy code

```

X = df['reviewTitle'] + ' ' + df['reviewDescription']

y = df['IsPositiveRating']

dataset = pd.concat([X, y], axis=1)

dataset = dataset.rename(columns={0: 'text', 'IsPositiveRating':
'label'})

```

Tokenizer Loading and Dataset Tokenization:

DistilBERT Tokenizer:

The tokenizer from **distilbert-base-uncased** was loaded to tokenize the dataset:

python

Copy code

```
tokenizer = DistilBertTokenizerFast.from_pretrained('distilbert-  
base-uncased')
```

Tokenization:

A function was created to **tokenize the dataset**:

```
def tokenize_func(examples):  
  
    return tokenizer(examples['text'], padding='max_length',  
                    truncation=True)
```

The dataset was tokenized for the DistilBERT model.

Dataset Splitting:

The tokenized dataset was split into training and evaluation sets using an 80-20 split:

```
train_test_split = tokenized_datasets.train_test_split(test_size=0.2)  
  
train_dataset = train_test_split['train']  
  
eval_dataset = train_test_split['test']
```

Model Loading and Training Arguments:

DistilBERT Model:

The **DistilBERT for Sequence Classification** was loaded for the sentiment analysis task:

```
model =  
DistilBertForSequenceClassification.from_pretrained('distilbert-base-  
uncased')
```

Training Arguments:

Default training arguments were set using `TrainingArguments()`:

```
training_args = TrainingArguments(  
  
    output_dir='./results',  
  
    evaluation_strategy="steps",
```

```

per_device_train_batch_size=16,

per_device_eval_batch_size=16,

num_train_epochs=3,

save_steps=10,

save_total_limit=2,

logging_dir='./logs',

)

```

Summary of Modeling Approach

Traditional Testing Approach:

Initial testing was performed on random customer review samples to validate sentiment prediction.

The sentiment model (`sentiment_pipe`) was used to predict each review's sentiment and compare it with the actual sentiment.

DistilBERT Implementation:

A **fine-tuning approach** was applied using **DistilBERT** for sentiment classification.

The review titles and descriptions were tokenized and used for training a sequence classification model.

The dataset was split into **training** and **evaluation** sets for training purposes.

TrainingArguments were configured to train the model effectively and save the progress at different steps.

```

model.safetensors: 100% |#####| 200M/200M [00:02<00:00, 1.20MB/s]
Some weights of DistilBertForSequenceClassification were not initialized from the model checkpoint at distilbert-base-uncased and are newly initialized
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
wandb: WARNING The `run_name` is currently set to the same value as `TrainingArguments.output_dir`. If this was not intended, please specify a differen
wandb: Using wandb-core as the SDK backend. Please refer to https://wandb.me/wandb-core for more information.
wandb: Logging into wandb.ai. (Learn how to deploy a W&B server locally: https://wandb.me/wandb-server)
wandb: You can find your API key in your browser here: https://wandb.ai/authorize
wandb: Paste an API key from your profile and hit enter, or press ctrl+c to quit: .....
wandb: Appending key for api.wandb.ai to your netrc file: /root/.netrc
Tracking run with wandb version 0.18.7
Run data is saved locally in /content/wandb/run-20241128_155808-mca7gtya
Syncing run ./results to Weights & Biases \(docs\)
View project at https://wandb.ai/gangadhar-singh-extreme-networks/huggingface
View run at https://wandb.ai/gangadhar-singh-extreme-networks/huggingface/runs/mca7gtya
[770/770 09:53, Epoch 5/5]

Epoch  Training Loss  Validation Loss
-----  -
1         0.297600         0.315082
2         0.205700         0.322781
3         0.194700         0.347537
4         0.115800         0.337201
5         0.103200         0.359933

TrainOutput(global_step=770, training_loss=0.2099596697788734, metrics={'train_runtime': 844.0835, 'train_samples_per_second': 14.507, 'train_steps_per_
1622063296542720.0, 'train_loss': 0.2099596697788734, 'epoch': 5.0})

```

7. Results

1. Understanding Sentiment Variation Across Countries or Continents

- **Clustering Analysis:**
 - Countries were grouped into three clusters based on sentiment-related attributes.
 - **Cluster 0:** Japan, UAE, United States
 - **Cluster 1:** India, Japan, UAE
 - **Cluster 2:** UAE, Egypt, US, Mexico, Canada
 - Key findings from clustering:
 - **Cluster 1:** Common focus on product attributes like "iphone," "camera," and "battery."
 - **Cluster 0:** More concerns about product conditions like "scratches" and "battery."
 - **Cluster 2:** Similar issues as Cluster 0 but with more emphasis on "screen" and "condition."
- **Country-Specific Sentiments:**
 - **India & UAE:** Average rating of 3.81 and 3.91, showing balanced feedback with high engagement.
 - **Canada & Egypt:** Sparse data but high average ratings (4.5 and 5.0 respectively).
 - **Japan:** Lower average rating of 3.31, indicating lower satisfaction levels.
- **Insights:**
 - Regional clusters highlight diverse customer expectations.
 - High engagement and satisfaction in India and the US, with quality perception varying across regions.

2. Key Factors Influencing Ratings

- **Analysis of Product Features:**
 - Storage capacity:
 - 128GB variants dominate usage, indicating economic and accessibility factors.
 - Higher capacities (e.g., 512GB) show limited engagement.
 - Color preferences:
 - "Blue" and "Midnight" are most popular, but data inconsistencies in color naming require cleaning.
 - Product condition:
 - "Scratches" and "screen" were frequent words in reviews, hinting at common customer concerns.

- **Hypothesis Testing:**
 - Statistical tests (e.g., ANOVA or t-tests) can validate the influence of country, product variant, and verification status on sentiment.
 - Observed trends, such as differences in rating averages across countries and storage capacities, suggest significant variance in preferences.
-

3. Comparison of Traditional Machine Learning Models with LLMs

- **Performance Metrics:**
 - **Random Forest:**
 - Accuracy: **99.67%**, Precision: **99.67%**, Recall: **99.67%**, F1: **99.67%**
 - Best-performing model, showing strong interpretability and effectiveness for structured data.
 - **DistilBERT:**
 - Accuracy: **90.7%**, Precision: **90.78%**, Recall: **90.7%**, F1: **90.73%**
 - Strong performance for a transformer model but not on par with Random Forest.
 - **Logistic Regression & AdaBoost:**
 - Accuracy: 83.5% (Logistic Regression) and 85.59% (AdaBoost), with lower overall performance compared to Random Forest.
 - **Insights:**
 - Random Forest's ability to handle structured data features provides it an edge over the LLM.
 - DistilBERT's contextual understanding makes it suitable for more nuanced NLP tasks but may require fine-tuning for better accuracy.
-

4. Temporal Trends Analysis

- **Observation:**
 - Ratings dropped by ~20% between Fall 2022 and Fall 2023, followed by a slight recovery with fluctuations.
 - **Seasonal Effects:**
 - Product launches or holiday seasons may influence spikes in ratings, possibly tied to customer excitement or expectations.
 - **Insights:**
 - Monitoring temporal trends can reveal actionable insights, such as optimizing product features or marketing campaigns during specific periods.
-

8. Conclusion

This research successfully analyzed customer sentiment towards iPhone products, revealing significant insights regarding regional preferences, popular product features, and factors contributing to satisfaction or dissatisfaction. The **DistilBert model** offers advanced NLP capabilities, the **Random Forest model demonstrates superior overall performance** in the context provided, particularly in terms of handling varied and potentially nuanced sentiment expressions within the data. **This showcases the importance of model selection based on specific task requirements and data characteristics.**

6.1 Future Work

- **Sentiment Analysis Expansion:** Incorporate more demographic data, such as age and gender.
- **Model Improvement:** Utilize newer LLMs, such as **GPT-4**, for even higher accuracy.
- **Actionable Insights:** Apply clustering and temporal analysis insights to develop targeted marketing campaigns and product features.

9. References

- **Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018).** BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.
 - **Le, Q., & Mikolov, T. (2014).** Distributed Representations of Sentences and Documents. ICML.
 - **Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013).** Efficient Estimation of Word Representations in Vector Space.
 - **Lloyd, S. (1982).** Least Squares Quantization in PCM. IEEE Transactions on Information Theory.
 - **Pennington, J., Socher, R., & Manning, C. (2014).** GloVe: Global Vectors for Word Representation.
-