**Artificial Intelligence**

**and Machine Learning**

Project Report

Semester-IV (Batch-2022)

Social Media Sentiment Analysis

A red and white sign

Description automatically generated with low confidence

**Supervised By: Submitted By:**

Dr. Karan Deep Singh Pearl, 2210990648

Professor Prachi Anand, 2210990660

Department of Computer Science Prachi Malik, 2210990661

& Engineering

**Department of Computer Science and Engineering**

## Chitkara University Institute of Engineering & Technology,

## Chitkara University, Punjab

**Abstract**

Imagine computers reading text and figuring out how people feel about things. That's what sentiment analysis does. This paper looks at how we teach computers to do this better. We talk about different ways computers learn, like old-school methods and fancy new ones using deep learning. We also look at where sentiment analysis is used, like on social media, in product reviews, and even in politics and healthcare. But it's not all easy—there are tricky parts, like understanding different languages and slang. By studying this, we can get better at helping computers understand our emotions in text, which is super useful in lots of areas.

Sentiment analysis, a subfield of natural language processing, aims to computationally identify and extract subjective information from textual data, discerning the emotional tone and opinion expressed within. This paper provides a comprehensive overview of recent advancements in sentiment analysis techniques, ranging from traditional machine learning approaches to state-of-the-art deep learning models. It discusses various applications of sentiment analysis across diverse domains such as social media, customer reviews, political discourse, and healthcare. Furthermore, the paper addresses the challenges and limitations associated with sentiment analysis, including context ambiguity, linguistic nuances, and cultural differences. Through an analysis of current research trends and methodologies, this paper contributes to a deeper understanding of sentiment analysis and its evolving role in shaping decision-making processes in both academic and commercial domains.

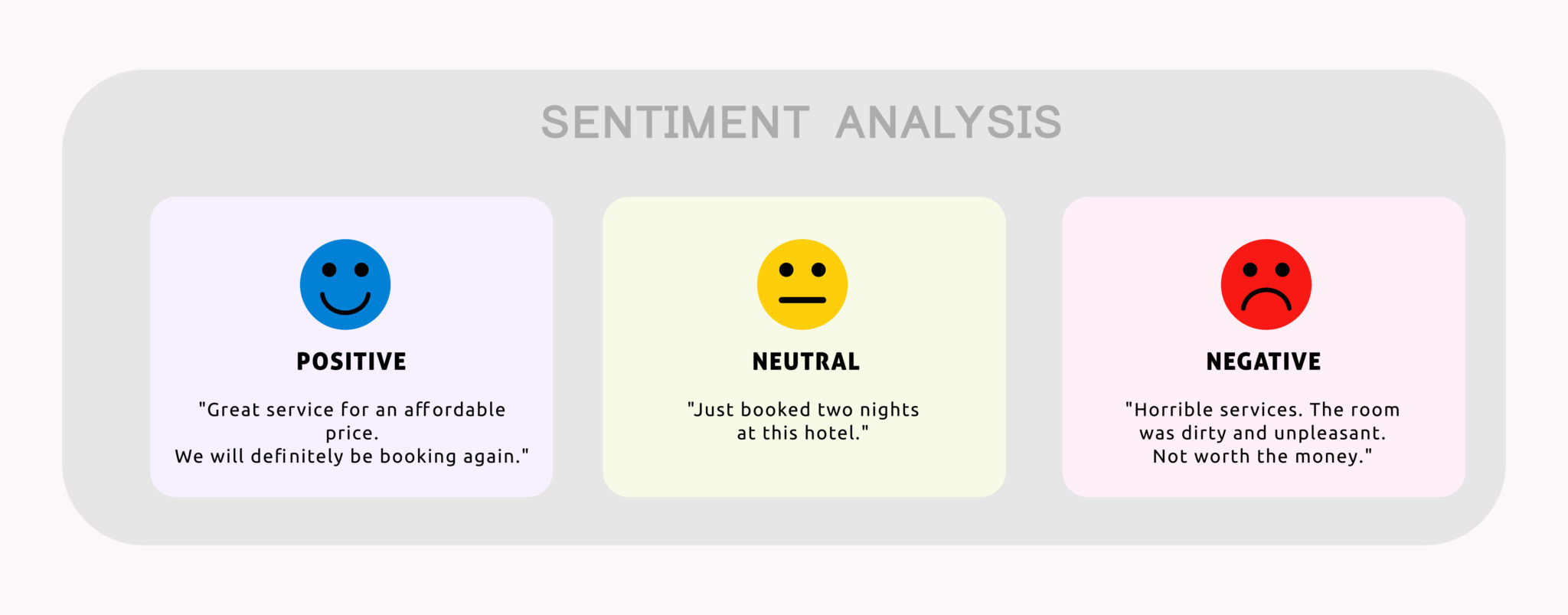
**Table of Content**

|  |  |  |
| --- | --- | --- |
| S No. | Topics | Page No. |
| 1 | **Introduction** | **1-3** |
| 2 | **Problem Definition & Background** | **3-6** |
| 3 | **Proposed Design / Methodology** | **6-12** |
| 4 | **Results** | **12-21** |
| 5 | **References** | **22** |

**1. Introduction**

* 1. **Background**

**What is Sentiment Analysis?**

Sentiment analysis is a natural language processing (NLP) technique used to determine the sentiment or emotional tone expressed within a piece of text. Sentiment analysis is like teaching a computer to understand how people feel when they write something. Imagine you have a magic machine that reads a message and tells you whether the person who wrote it is happy, sad, angry, or something else. That's what sentiment analysis does, but instead of magic, it uses algorithms and data to figure out the emotions behind words. The process typically involves several steps, including text preprocessing, feature extraction, and sentiment classification. During text preprocessing, the raw text is cleaned and transformed into a format suitable for analysis, which may involve tasks like tokenization, removing stop words, and stemming. Feature extraction involves identifying relevant features or attributes of the text that can be used to determine sentiment, such as keywords or linguistic patterns.

It's pretty handy for businesses to understand what customers are saying about their products or for analyzing public opinion on social media. It involves analyzing textual data to identify and categorize opinions, attitudes, and emotions as positive, negative, or neutral.

**Why it is Important?**

Sentiment analysis is important because it helps us understand how people feel about things. Imagine if you had a big business and lots of people were talking about your products online. Sentiment analysis could tell you whether they love your products, are unhappy with them, or just feel okay about them. This helps businesses know what their customers like or don't like, so they can improve their products or services. It's like having a superpower to read people's minds through their words!

**Brief History**

Sentiment analysis has been around for quite some time, but it really started to gain attention in the late 20th century with the rise of the internet. As more and more people began expressing their thoughts and opinions online through blogs, forums, and social media, researchers and businesses saw an opportunity to understand public sentiment.

In the early days, sentiment analysis was quite basic. It mainly focused on counting the frequency of positive and negative words in a piece of text to determine overall sentiment. This approach, however, often lacked nuance and couldn't capture the complexities of human language.

Over time, researchers started developing more sophisticated techniques, drawing from fields like natural language processing (NLP) and machine learning. These techniques allowed sentiment analysis algorithms to not only detect sentiment but also understand context, sarcasm, and emotions expressed in text.

With the arrival of big data and advancements in machine learning algorithms, sentiment analysis became even more powerful. Businesses began using it to analyze customer feedback, monitor brand reputation, and gain insights into market trends.

Today, sentiment analysis is a widely used tool across various industries, helping businesses make data-driven decisions, improve customer experiences, and stay ahead of the curve in an increasingly digital world.

* 1. **Objective**

The objective of this project is to develop and evaluate a sentiment analysis model capable of accurately categorizing the sentiment expressed in textual data into positive, negative, or neutral categories and overall generating a compound score (aggregate of all 3 values) which tells how negative, positive or neutral a text is. Through comprehensive experimentation and analysis, we aim to compare different sentiment analysis techniques, including traditional machine learning algorithms and advanced deep learning models, to determine their effectiveness in handling various types of text data. Additionally, we seek to address specific challenges in sentiment analysis, such as dealing with language ambiguity, sarcasm, and cultural nuances, by exploring novel features, preprocessing techniques, and optimization strategies. By improving the accuracy and performance of sentiment analysis models, we aim to facilitate their practical applications in domains such as social media monitoring, customer feedback analysis, and political discourse analysis. Ultimately, this project strives to contribute to the advancement of sentiment analysis research and provide valuable insights into human emotions and opinions expressed in textual data, thereby enabling informed decision-making and enhancing various applications across different domains.

The significance of sentiment analysis lies in its ability to extract valuable insights from textual data, empowering decision-makers across diverse domains. In business and marketing, it aids in understanding customer sentiments, shaping product development, and enhancing brand reputation. In politics, sentiment analysis helps gauge public opinion, informing policy decisions and political campaigns. Financial markets benefit from sentiment analysis by predicting market trends and managing investment risks. Additionally, sentiment analysis contributes to healthcare by analyzing patient feedback and improving service quality. Overall, sentiment analysis serves as a powerful tool for making data-driven decisions, improving customer experiences, and adapting strategies to meet evolving needs and preferences.

**2. Problem Definition & Requirements**

**Problem Statement**

In today's digital age, understanding and accurately interpreting human emotions and opinions expressed in text data poses significant challenges. Existing sentiment analysis models often struggle with context and sarcasm, leading to unreliable results. Additionally, the diverse range of data sources complicates sentiment analysis tasks further. Hence, there's a critical need to develop more robust models capable of overcoming these challenges and providing accurate insights into public sentiment. This project seeks to address these issues by exploring innovative techniques and enhancing the performance of sentiment analysis models, with the ultimate goal of improving decision-making processes across different domains.

**Software Requirements:**

For a sentiment analysis project, you'll need several software tools and libraries to preprocess text data, build and train sentiment analysis models, and evaluate their performance. Here's a list of essential software requirements:

**Programming Language:** Python

**IDE:** Jupyter Notebook, VS Code or PyCharm

**Python Libraries:**

* **Pandas:** Pandas is a powerful library for data manipulation and analysis in Python. It provides data structures like DataFrame, which is particularly useful for handling structured data such as text data from social media or customer reviews. You can use Pandas to load, clean, and preprocess your text data efficiently.
* **NumPy:** NumPy is a fundamental library for numerical computing in Python. It provides support for multidimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. NumPy is often used in conjunction with Pandas for data manipulation and processing.
* **Seaborn:** Seaborn is a statistical data visualization library based on Matplotlib. It provides a high-level interface for creating informative and attractive statistical graphics. Seaborn's functions are optimized for working with Pandas DataFrame objects, making it easy to visualize relationships between variables in your text data.
* **Matplotlib (Pyplot):** Matplotlib is a versatile plotting library for creating static, interactive, and animated visualizations in Python. Pyplot is a module within Matplotlib that provides a MATLAB-like interface for generating plots. You can use Matplotlib and Pyplot to create customized visualizations of your sentiment analysis results, such as histograms, scatter plots, and bar charts.
* **Transformers:** Transformers is a state-of-the-art library for natural language understanding (NLU) and natural language generation (NLG) tasks. It provides pre-trained models like BERT, GPT, and RoBERTa, which can be fine-tuned for sentiment analysis tasks. With Transformers, you can leverage powerful deep learning models to extract contextualized representations of text data and improve the accuracy of your sentiment analysis model.
* **NLTK (Natural Language Toolkit):** NLTK is a comprehensive library for natural language processing (NLP) tasks in Python. It provides modules and tools for tasks such as tokenization, stemming, lemmatization, part-of-speech tagging, and named entity recognition. NLTK is invaluable for preprocessing text data and extracting linguistic features relevant to sentiment analysis.
* **Tweepy:** Tweepy is a Python library that simplifies the process of accessing the Twitter API. It provides a convenient interface for interacting with Twitter's functionalities, such as reading and posting tweets, accessing user information, managing followers, and more. Tweepy abstracts away much of the complexity of working with the Twitter API, handling authentication, rate limits, and data parsing, allowing developers to focus on building Twitter-based applications or performing Twitter data analysis more efficiently. With Tweepy, developers can create bots, monitor tweets, gather data for analysis, and build custom applications that leverage Twitter's vast repository of information.

**Hardware Requirements:**

For a sentiment analysis project in Python, the hardware requirements are generally modest, as most tasks can be performed on standard computing hardware. Here are the recommended hardware specifications:

**Computer**: Standard desktop or laptop running Windows, Linux or Macintosh

**Processor**: Multi-core CPU recommended.

**RAM**: Minimum 4GB, preferably 8GB or higher.

**Storage**: Adequate space for datasets and code files. Minimum 1 GB.

**GPU (Optional):** NVIDIA GPU with CUDA support for accelerated deep learning tasks.

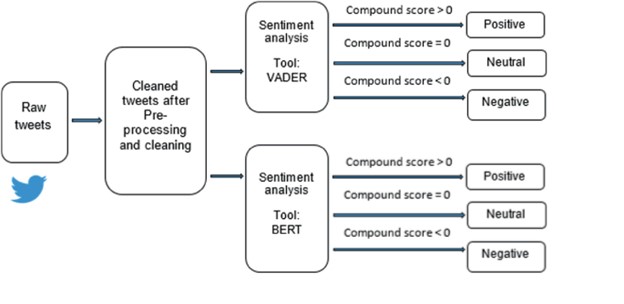
**Internet Connection:** Stable connection for accessing resources.

**Data Sets:**

For a sentiment analysis project, selecting appropriate datasets is crucial for training and evaluating your models. We used Twitter Sentiment Analysis, aiming to create Daily analysis of Trends and Tweets on Twitter.

**3. Proposed Design / Methodology**

Our project initiation involved the selection and import of crucial libraries such as pandas, numpy, seaborn, matplotlib, plotly, and nltk. These libraries were very important for conducting a variety of tasks including data manipulation, visualization, and natural language processing (NLP). Once the dataset was imported using pandas, we embarked on an in-depth initial exploration to fully grasp its structure and key features. This step was crucial in understanding the nature of the data we were dealing with.



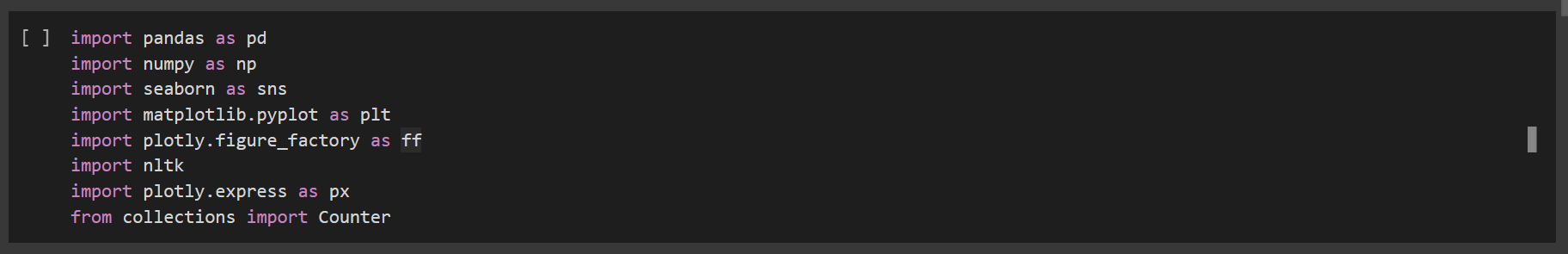
Subsequently, we explored data preprocessing, an essential step in any data analysis project. Here, our focus was primarily on the text data, which underwent rigorous cleaning to eliminate special characters, URLs ,stop words and non-alphanumeric characters. Additionally, we calculated various metrics such as the length of text and selected text, providing valuable insights into the nature of the textual data.

During the exploratory data analysis (EDA) phase, we deep dived into the distribution of sentiment categories within the dataset. Through sophisticated visualizations, we depicted the detailed and complex distribution of the number of words in both text and selected text, further enhancing our understanding. Additionally, we explored the difference in the number of words between text and selected text, uncovering nuanced patterns and trends.

For the sentiment analysis component, we utilized the powerful VADER (Valence Aware Dictionary and Sentiment Reasoner) tool, a widely used sentiment analysis tool in the field of NLP. VADER allowed us to determine the sentiment of textual data with a high degree of accuracy. Additionally, we integrated a state-of-the-art pre-trained Roberta model for more advanced sentiment analysis tasks as Roberta focuses on context of the text more as compared to VADER. Leveraging its capabilities, we were able to extract deeper semantic meaning from the text, enhancing the accuracy of our sentiment analysis.

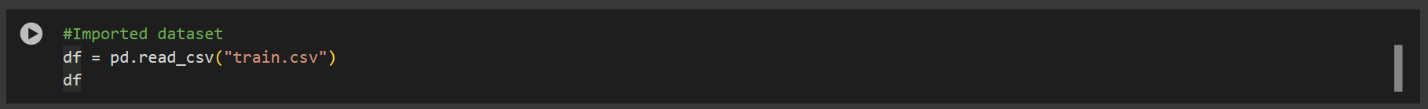
In the results visualization phase, we visualized the compound scores (aggregate of how positive, negative and neutral a text is) and individual sentiment scores (positive, neutral, negative) obtained from VADER. This provided us with valuable insights into the sentiment distribution within our dataset. Furthermore, we conducted an in-depth comparison of sentiment scores from both VADER and the Roberta model, allowing us to gain a deeper understanding of their respective performances and capabilities.

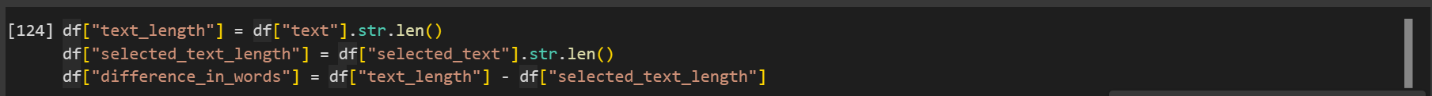
To ensure the comprehensiveness of our report, let's add detailed subsections summarizing each major step:

1) Data Import and Initial Exploration:

We imported necessary libraries such as pandas, numpy, seaborn, matplotlib, plotly, and nltk for data manipulation, visualization, and natural language processing tasks

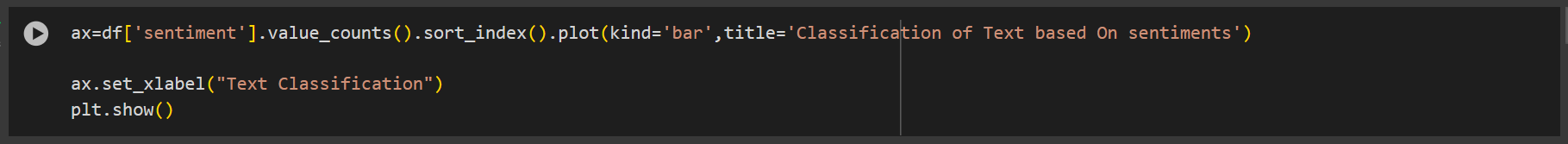
We imported the dataset using pandas and performed initial exploration to understand its structure.

2) Data Pre-Processing

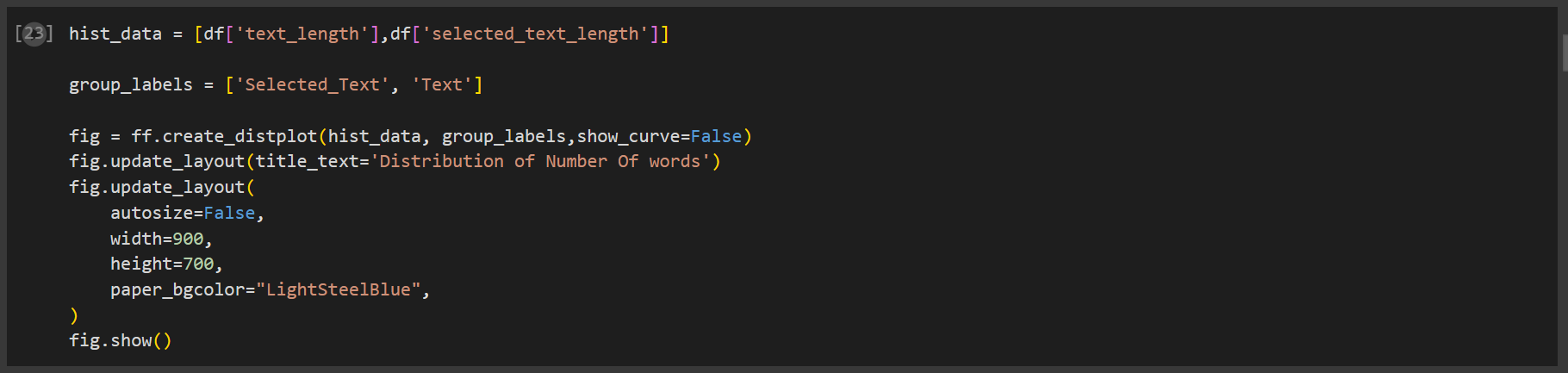


Text and selected text lengths were calculated along with the difference in the number of words between them.

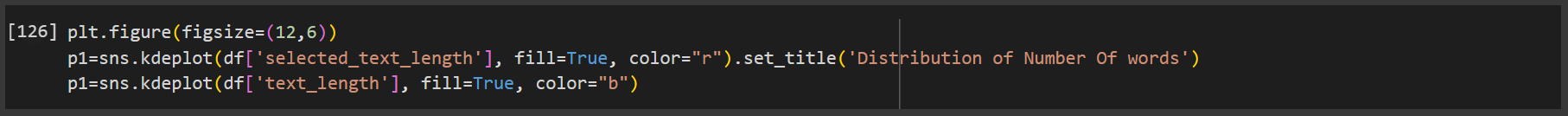
3) EDA Exploratory Data Analysis

* Sentiment Distribution Analysis:

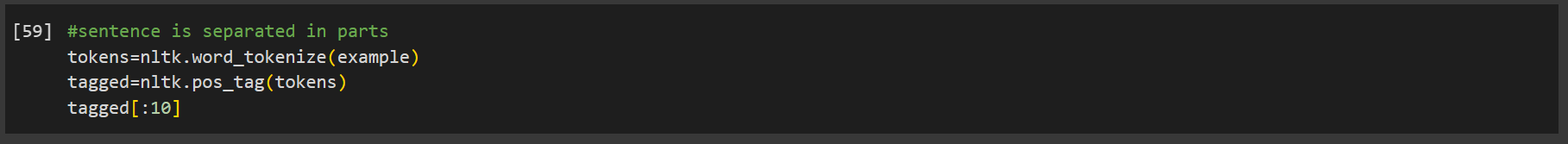
We conducted an analysis to understand the distribution of sentiment categories in the dataset.

* Text Length Distribution

Visualizations were created to depict the distribution of the number of words in both text and selected text.

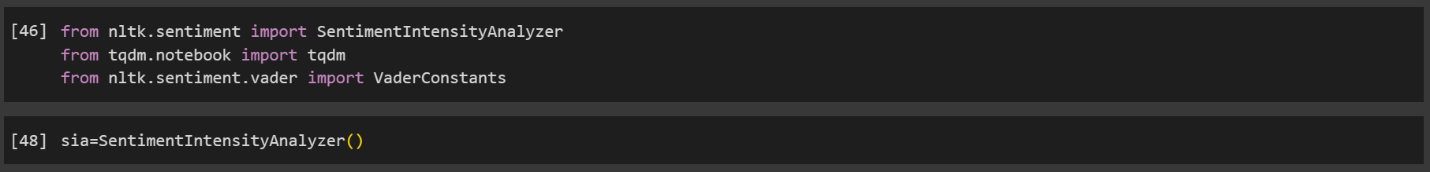
* Difference in Text Lengths:

We explored the distribution of the difference in the number of words between text and selected text.

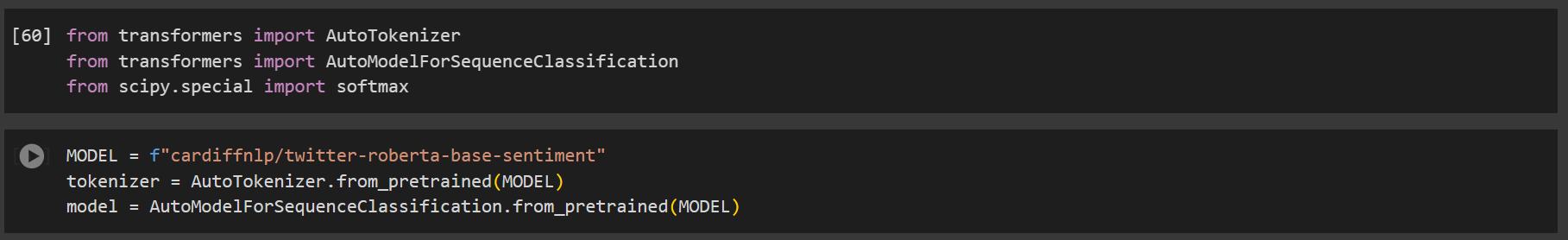
4) Tokenization

Then We Tokenized the Texts, So that we can feed that into the Model. And Model can give scores to each token. And Give us the Overall Scores as Output. Thus, Helping to find the Overall Sentiment of the Text.

5) Sentiment Analysis:

* VADER (Valence Aware Dictionary and Sentiment Reasoner):

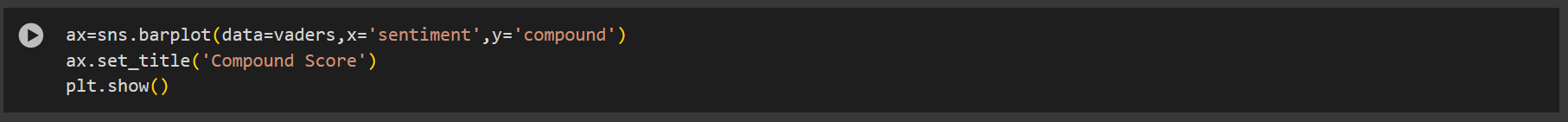
Utilizing the VADER sentiment analysis tool, we determined the sentiment of textual data.

* + Roberta Model Integration:

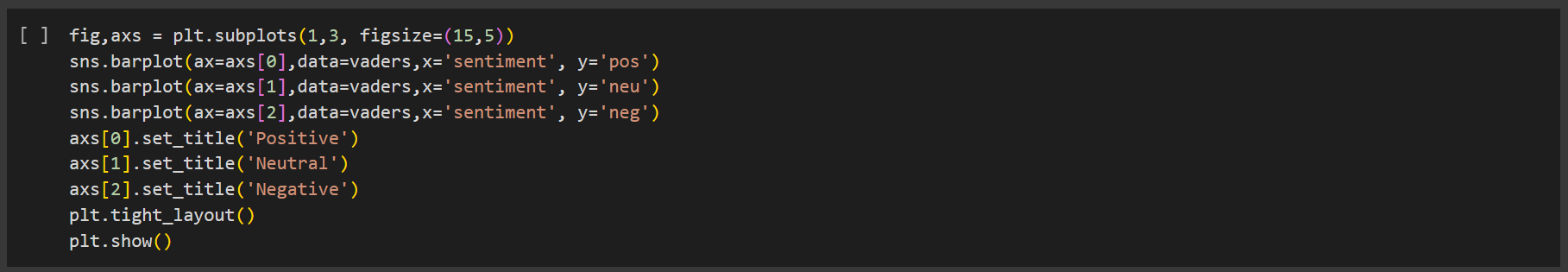
A pre-trained Roberta model was incorporated for sentiment analysis tasks.

6) Results Visualization:

* VADER Scores Visualization:



Compound score and individual sentiment scores (positive, neutral, negative) obtained from VADER were visualized.

* Comparison of Sentiment Scores:

Sentiment scores from both VADER and the Roberta model were compared through visualizations.

**VADER (Valence Aware Dictionary and sEntiment Reasoner):**

VADER (Valence Aware Dictionary and sEntiment Reasoner) is like a smart tool that helps computers understand the emotions behind words in a sentence. It has a big list of words and phrases, each with a score that shows how positive, neutral, or negative it is. When you give VADER a sentence, it looks at the words in that sentence and adds up their scores to figure out the overall sentiment (compound score) – whether it's positive, negative, or neutral. It's a handy tool for sentiment analysis because it can quickly analyze text and give a sense of how people feel about something without needing a lot of training or complex algorithms.

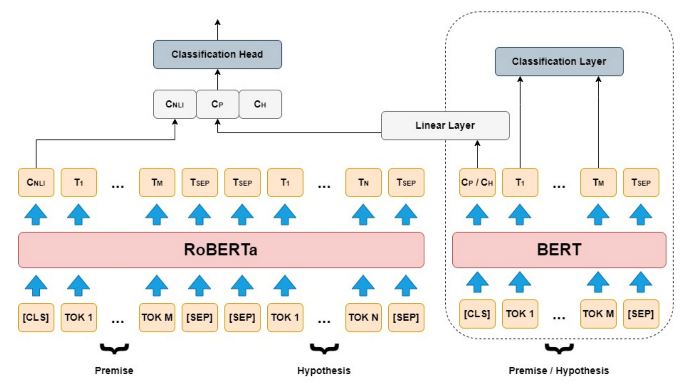
It derives scores empirically and calibrates them to capture the intensity and valence of sentiments in a phrase. Additionally, it also has rules and regulations to deal with linguistic features such as negation, capitalization, punctuation, and emotions which are there in social text but can’t be analyzed via traditional sentiment analysis methods.

It has the ability to capture both the polarity and intensity of sentiment expressed in text as one of its major key advantages. While many sentiment analysis tools solely focus on classifying text as positive, negative, or neutral, VADER goes a step further by providing a quantitative measure of sentiment intensity. This nuanced approach enables VADER to differentiate between subtle variations in sentiment expression, such as distinguishing between mild positivity and strong positivity.

Moreover, VADER is designed to be fast, accurate, and easily adaptable to various domains and languages. Its simplicity and efficiency make it a popular choice for sentiment analysis tasks in social media monitoring, customer feedback analysis, market research, and more. Despite its rule-based nature, VADER demonstrates competitive performance compared to more complex machine learning models, especially in scenarios where training data may be limited or noisy.

Overall, in the era of social media, VADER stands as a valuable tool in the sentiment analysis toolkit, offering researchers and practitioners a straightforward and effective solution for extracting meaningful insights from textual data and beyond.

RoBERTa, short for the Robustly optimized BERT approach, stands as a cornerstone in the landscape of natural language processing (NLP), indicating a significant advancement in the field since its introduction by Facebook AI in 2019. Rooted in the transformative architecture of BERT (Bidirectional Encoder Representations from Transformers), RoBERTa refines and amplifies the capabilities of its predecessor through a meticulous revamp of key hyperparameters and training methodologies.

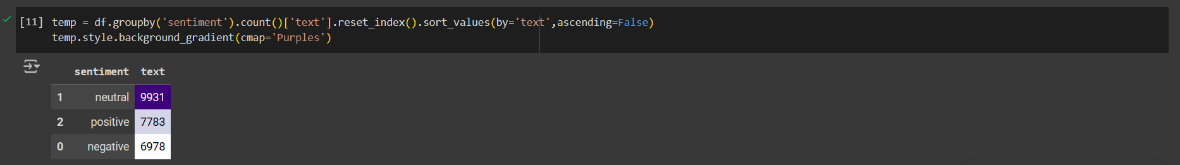
**RoBERTa Model**

At its core, RoBERTa capitalizes on the bidirectional nature of the transformer architecture, enabling it to grasp intricate contextual shades within textual inputs. This bidirectional understanding empowers the model to discover the meanings of words within the broader context of sentences, paragraphs, or documents, thereby enhancing its comprehension and representation capabilities. As it is built upon BERT’s architecture, it can further fortify robustness and efficacy. These modifications involve optimizations in training procedures, encompassing elongated training sequences, dynamic masking strategies, and augmented batch sizes. Such enhancements promote a more profound understanding of text data and strengthen the model's adaptability across a spectrum of NLP tasks.

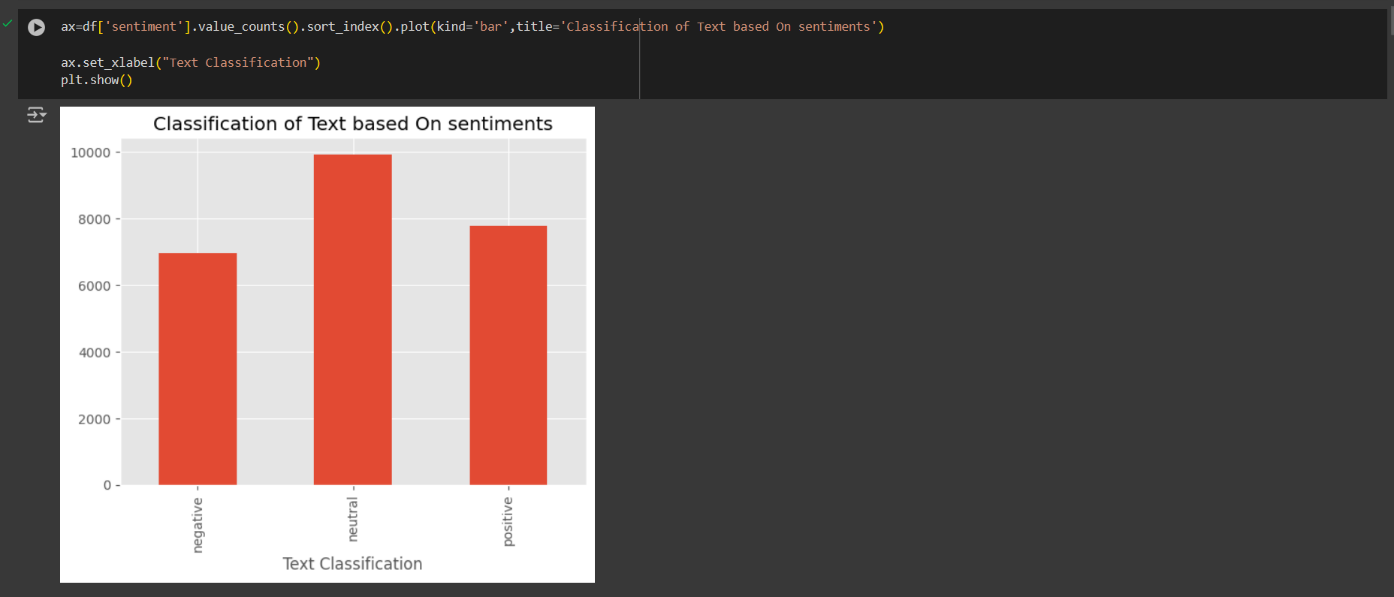
Notably, it diverges from BERT by avoiding the next sentence prediction (NSP) task during pre-training, thereby affording the model more focus on contextual learning and representation. By omitting NSP, it sidesteps the constraints associated with predicting sentence relationships and can allocate more resources toward capturing subtle linguistic features. It is better in understanding and responding because it has been trained on a diverse set of examples.

It has a variety of applications in NLP domains such as text classification, named entity recognition to question answering, and text generation. As RoBERTa is a contextual model, it can efficiently discern sentiments, identify named entities, and generate coherent text. Its adaptability makes it efficient in machine translation systems, elevating the overall quality and fluency of translated text.

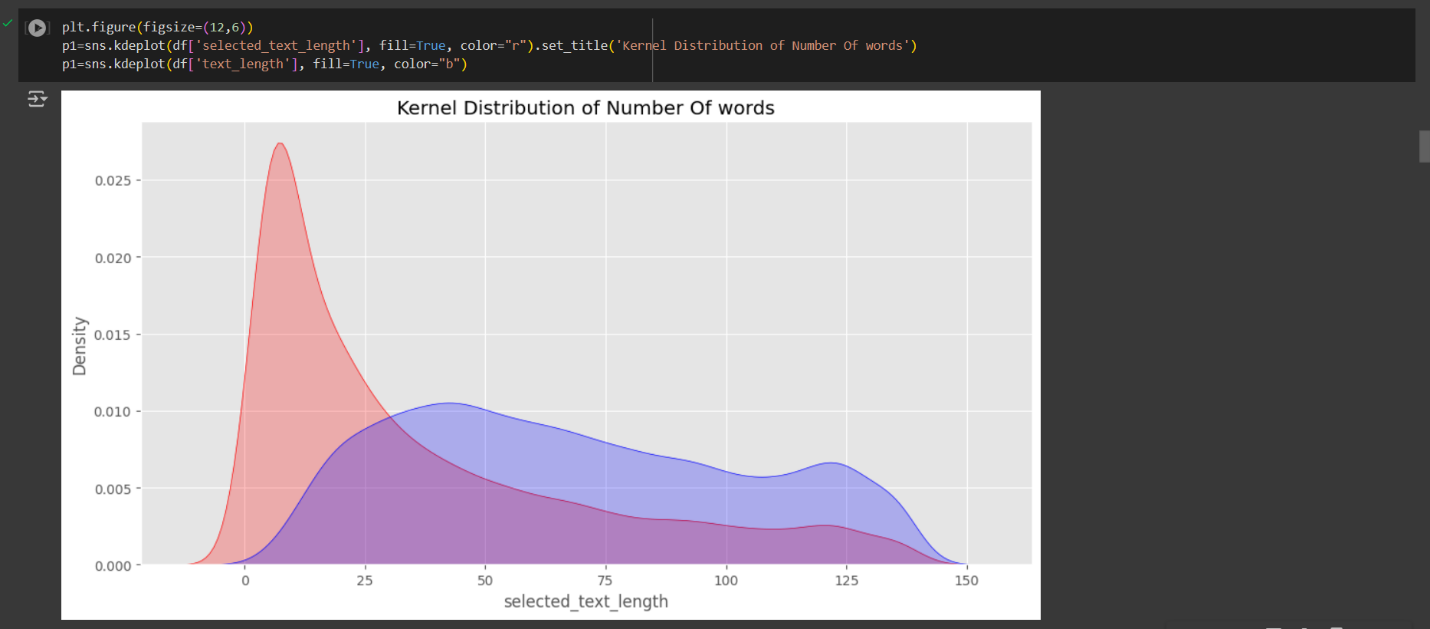
In conclusion, it stands as a testament to the continuous evolution of NLP, incorporating the fusion of cutting-edge research and practical innovation. Its robust architecture, refined training procedures, and versatile applications underscore its prominence as a cornerstone model for natural language understanding. As researchers and practitioners continue exploring its potential, it promises to spearhead transformative advancements in NLP, catalyzing progress across diverse applications and industries.

**4. Results:**

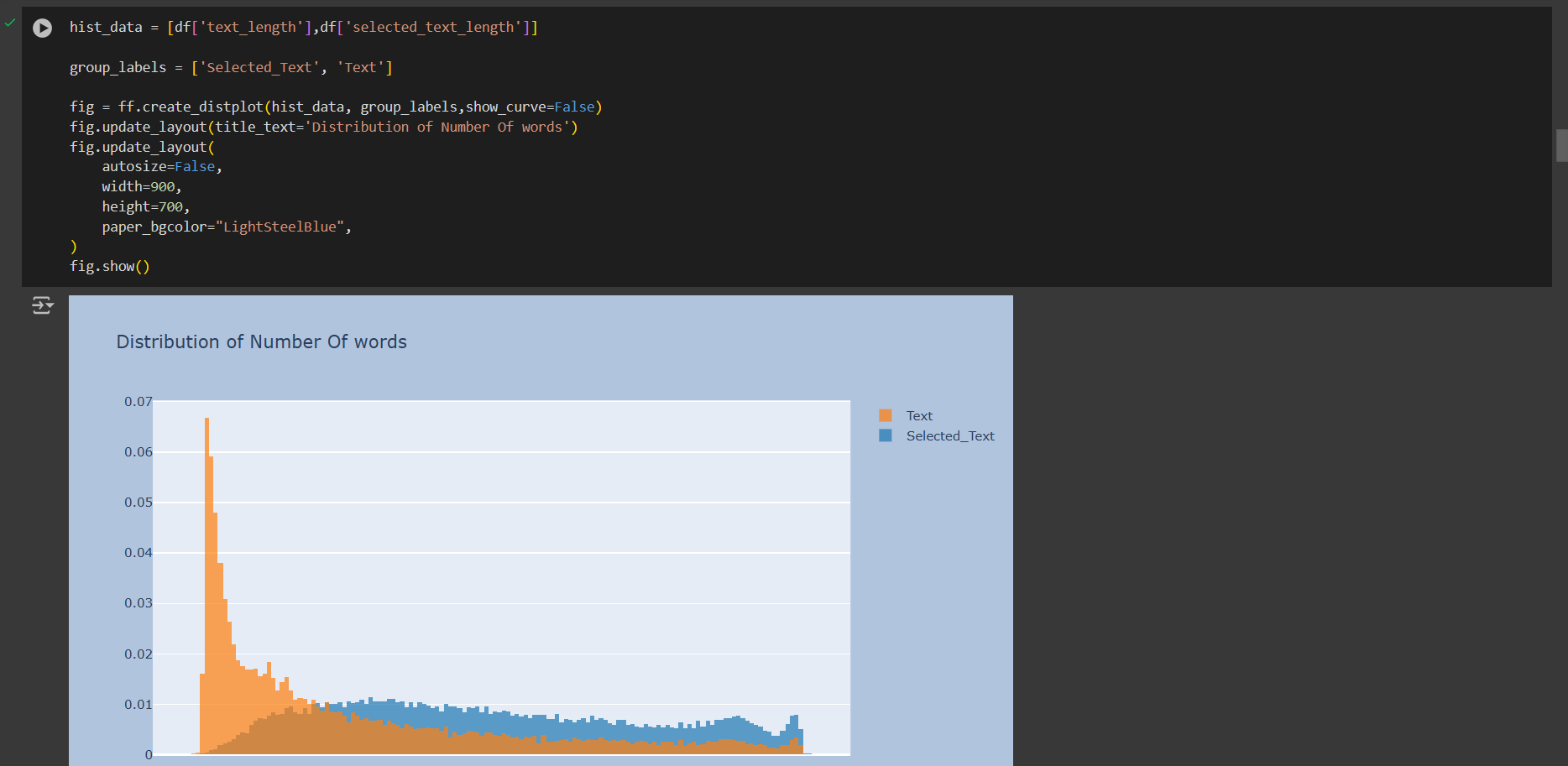
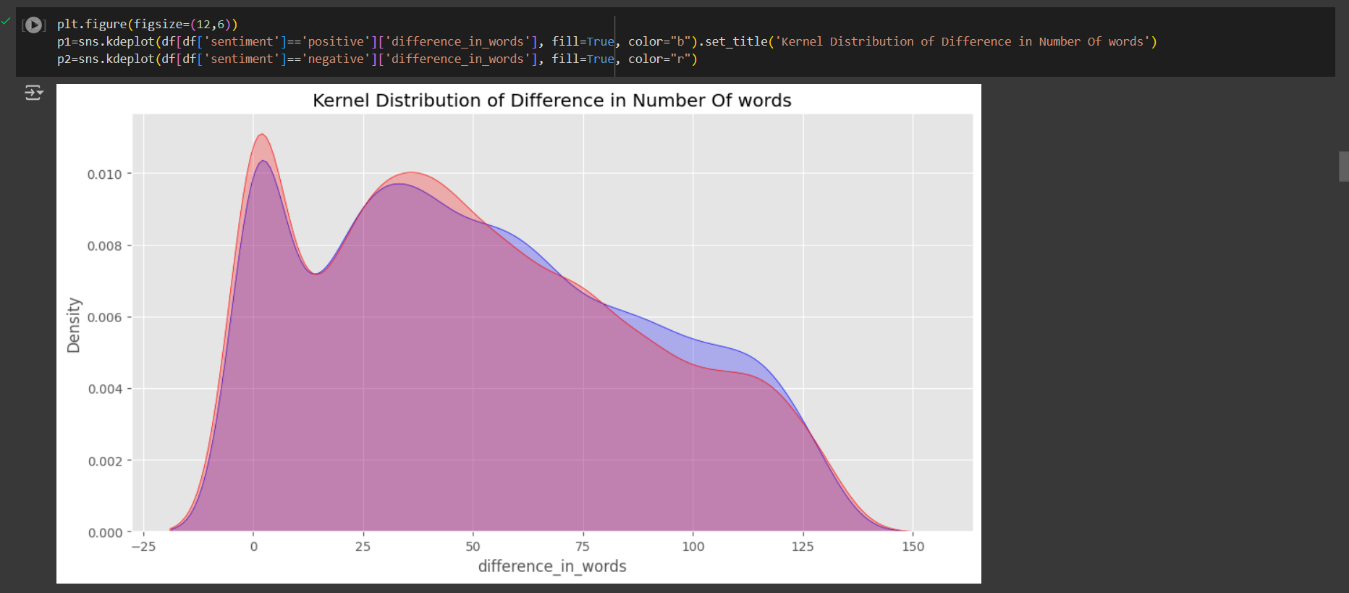
The given graph provides a concise summary of insights of sentiment distribution within our dataset helping us to better understand patterns and trends. It sorts count of sentiment by frequency and aggregate them in DataFrame column to visually represents the data using a purple gradient. Results show 11,117 instances of neutral sentiment, 8,582 positive, and 7,781 negative. The darker shades of gradient denotes higher counts.



In the above graph, a bar plot is shown displaying the distribution of text classifications of sentiment column. Sentiment categories are represented on x-axis, and the count of occurences is represented on y-axis. A clear visualization of sentiment categories is given enabling easy comparison between categories. It gives a concise summary of sentiment frequencies to better understand sentiment composition.

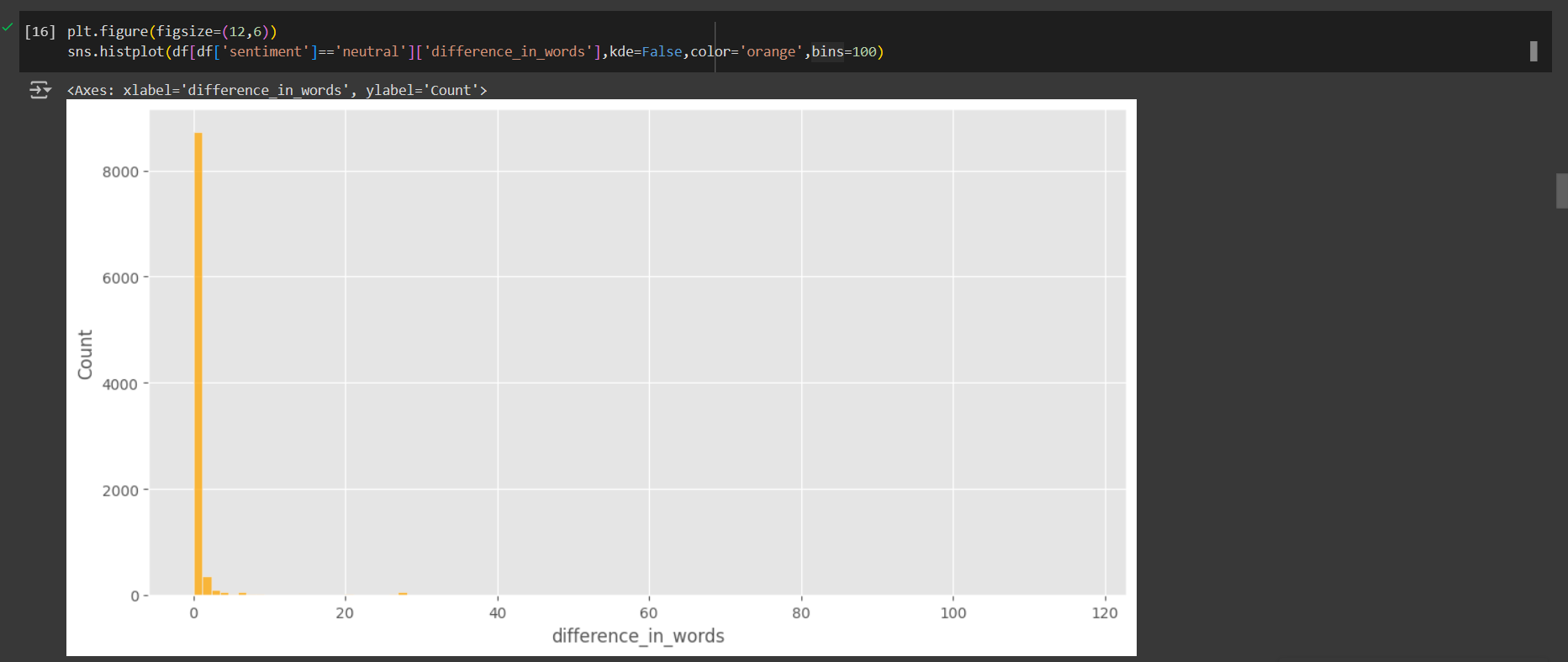


It generates a distribution plot depicting the number of words in both the original text and the selected text columns. This visualization helps us understand the range and distribution of text lengths, providing crucial insights into the nature of the textual data.

The plot illustrates the distribution of the number of words in both the original text and the selected text. It helps us understand whether there's a significant difference in length between the two, which can be crucial for our sentiment analysis tasks. For instance, a large difference in length might indicate more subjective or summarized text in the selected text column.

The following graph tells Kernel Density Estimation (KDE) plots to visualize the distribution of the number of words in both the original text and the selected text. This visualization provides insights into the distribution of the number of words in both the original text (blue) and the selected text (red). Kernel Density Estimation plots represent the probability density function of a continuous random variable. Here, the peaks in the KDE plots indicate regions of higher density, showing where most of the data lies.

Understanding the distribution of word counts is crucial for analyzing the characteristics of text data. In this case, we can observe whether the length of the selected text differs significantly from the original text, which is valuable for tasks such as sentiment analysis or summarization. For instance, a wider spread in the selected text length might suggest variability in the amount of information captured in the selected text compared to the original text.

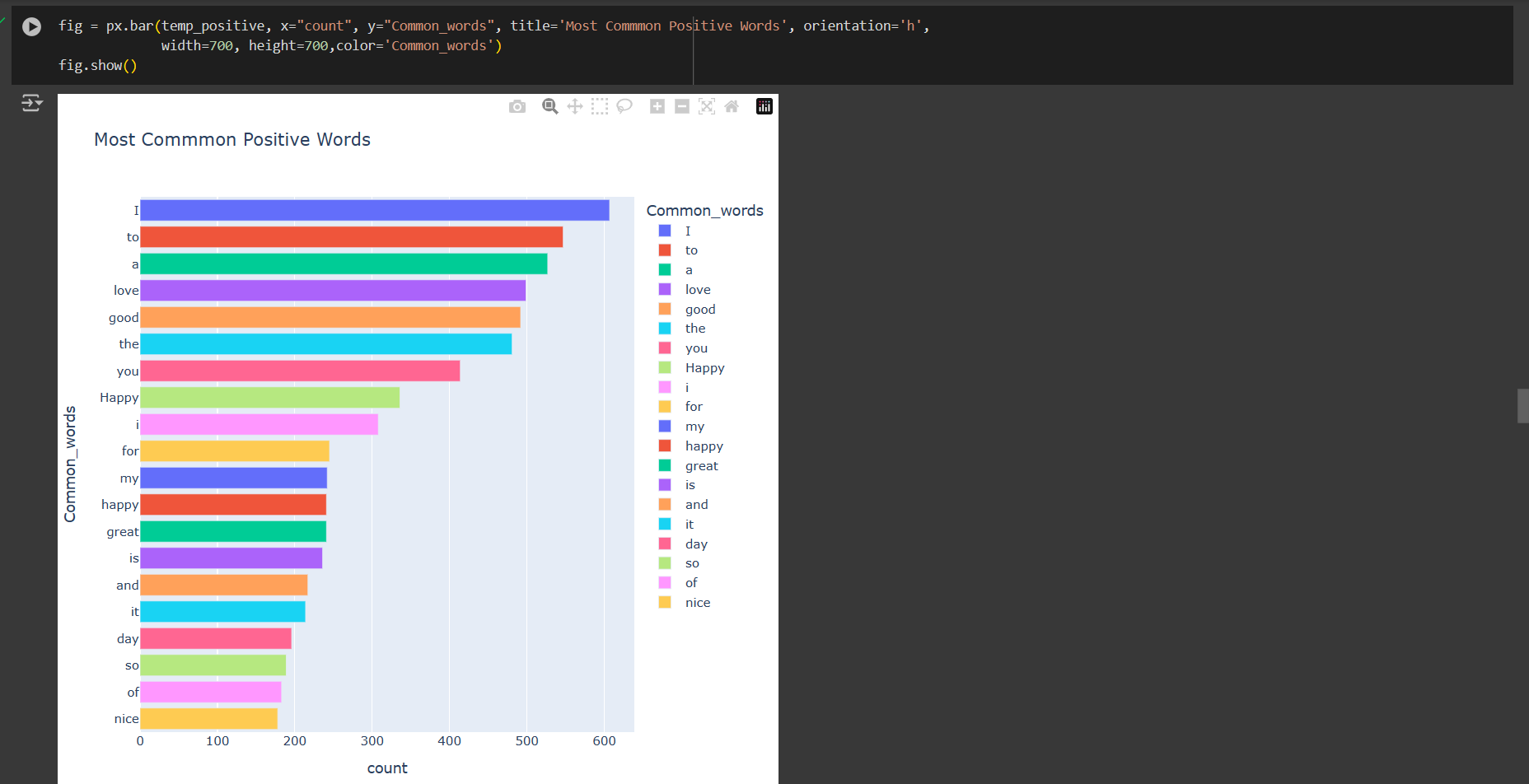
The blue KDE plot represents the distribution of the difference in the number of words for text with positive sentiment, while the red KDE plot represents the distribution for text with negative sentiment.

Analyzing the difference in word counts based on sentiment provides insights into how sentiments are expressed and summarized within the selected text. For example, a wider spread in the positive sentiment distribution might indicate variability in the expression of positive sentiments across different text segments.



The following code generates a histogram to visualize the distribution of the difference in the number of words between the original text and the selected text for instances where the sentiment is labeled as 'neutral'. The plot aids in understanding how the difference in word counts varies specifically for neutral sentiment text segments. The orange color scheme enhances visibility, and the division of data into 100 bins provides granularity in the analysis.

The code snippet utilizes Plotly's treemap visualization to present the distribution of the most common words. Each block within the treemap represents a word, and the size of the block corresponds to the frequency of the word's occurrence. The hierarchical structure allows for easy navigation and comprehension of word frequencies. This visualization offers a clear overview of the most prevalent words in the dataset, aiding in identifying key themes or topics.

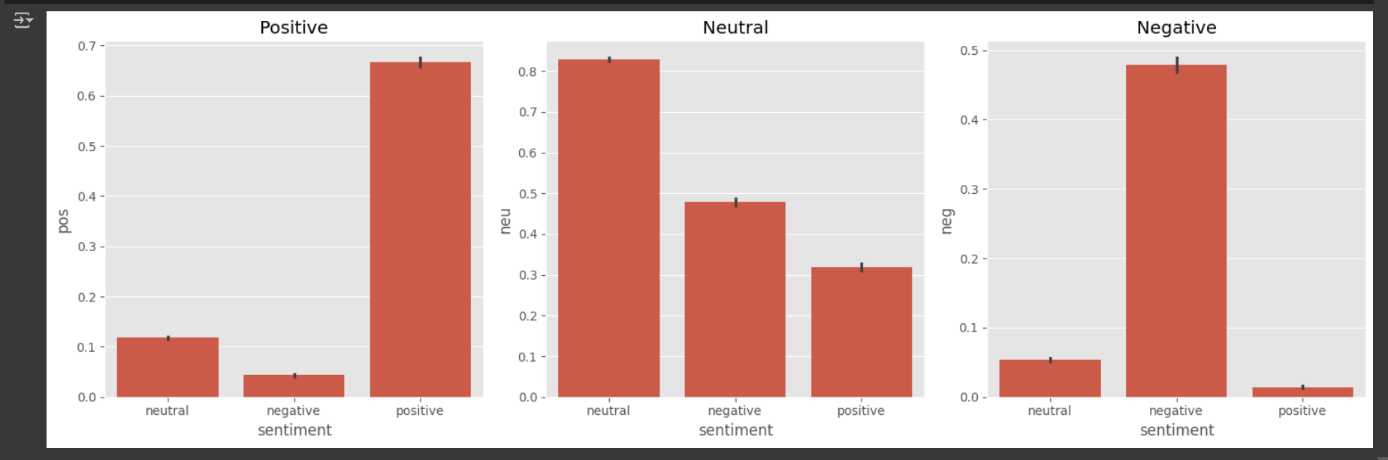


The code snippet employs Plotly's horizontal bar chart to illustrate the distribution of the most common positive words. Each bar represents a positive word, and its length corresponds to the frequency of occurrence. The color variation aids in distinguishing between different positive words. This visualization offers a comprehensive overview of the prevalent positive terms in the dataset, facilitating the identification of key sentiments and themes associated with positivity.



The provided code generates a treemap visualization representing the distribution of the most common neutral words within the dataset. Each block within the treemap corresponds to a neutral word, with the size of the block proportional to the frequency of occurrence. The hierarchical structure of the treemap facilitates the visualization of word frequencies and enables the identification of prevalent neutral terms. This visualization offers valuable insights into the neutral aspects of the dataset, aiding in understanding the underlying themes and topics

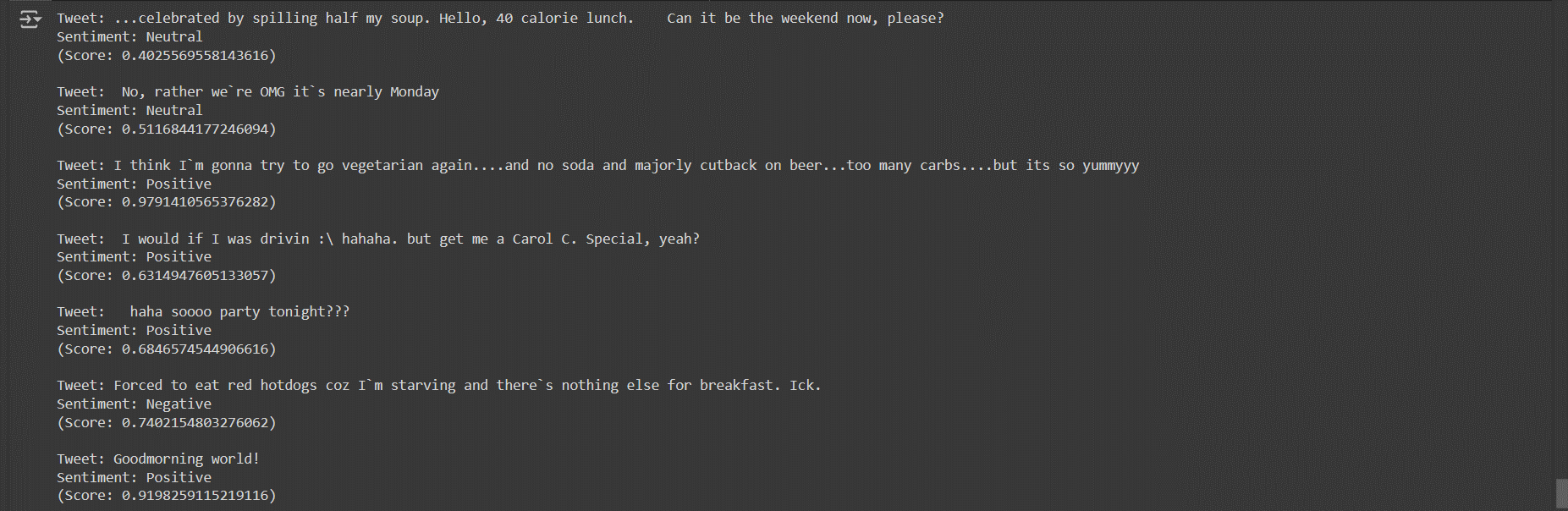
The code snippet creates a treemap visualization illustrating the distribution of the most common negative words within the dataset. Each block in the treemap represents a negative word, with its size proportional to the frequency of occurrence. The hierarchical layout aids in visualizing word frequencies and facilitates the identification of prevalent negative terms. This visualization provides valuable insights into the negative aspects of the dataset, assisting in understanding the underlying sentiments and themes.

The provided code generates a bar plot illustrating the compound score for each sentiment category. Each bar represents a sentiment category, and its height corresponds to the average compound score. The visualization provides an overview of the sentiment polarity within the dataset, with positive scores indicating positive sentiment, negative scores indicating negative sentiment, and scores closer to zero representing neutral sentiment. This summary aids in understanding the overall sentiment distribution and polarity within the text data.

The provided code generates three subplots, each depicting the proportion of positive, neutral, and negative sentiments within the dataset. The bar plots display the distribution of sentiment scores for each sentiment category—positive, neutral, and negative. The first subplot represents the proportion of positive sentiment, the second subplot represents neutral sentiment, and the third subplot represents negative sentiment. This visualization offers insights into the distribution of sentiment categories and their respective proportions within the dataset, facilitating a nuanced understanding of sentiment polarity.

**Final Result:**





At last, We have integrated RoBERTa, a leading transformer-based model known for its robust language understanding capabilities, with the Twitter API to develop an insightful sentiment analysis report. This integration enables us to harness the power of RoBERTa in analyzing Twitter data, providing a comprehensive understanding of sentiment trends within the platform's discourse.

Through the utilization of Tweepy, a Python library facilitating access to Twitter's API, our system efficiently retrieves tweets based on user-defined query terms. Subsequently, each tweet undergoes sentiment analysis via RoBERTa. This process involves tokenization of the tweet text and inference through the RoBERTa model to derive sentiment scores. These scores are then interpreted probabilistically to categorize tweets into Positive, Negative, or Neutral sentiments.

The resultant sentiment analysis report meticulously presents the tweet content alongside their corresponding sentiment classifications and associated confidence scores. This approach not only offers valuable insights into prevailing sentiment patterns across diverse topics but also underscores the efficacy of advanced natural language processing techniques in distilling meaningful insights from social media discourse.

In summary, our integration of RoBERTa with the Twitter API facilitates a systematic and insightful analysis of sentiment trends, thereby providing a strategic framework for extracting actionable intelligence from Twitter data.

**References:**

Geeks for Geeks: <https://www.geeksforgeeks.org>

Wikipedia: <https://www.wikipedia.com>

Medium: <https://medium.com>

Twitter API: <https://developer.twitter.com/en/portal/dashboard>