**News Analysis Project: Uncover What Matters!!!**

**By- AAYUSH.J.SHAMKUWAR**

**Aim:** 1.) Clean Up Articles

2.) Check the Mood.

3.) Find Connections.

4.) Aspect Analysis (Optional)

**Objective:** Develop an integrated analysis system utilizing three modules - clean-up, mood check, and finding connections - to process news articles.

**Tools Used:** Jupyter Notebook.

**Program Summary:**

Load and Preprocess the data

importing libraries to load the dataset and for preprocessing

pandas

numpy

**pandas:** pandas is an open source library widely used for data analysis. It is used for reading and manipulating data in machine learning. To understand high level overview of data, pandas offer multiple functions and some of them are :-

head(): shows top 5 records of the dataset

tail(): shows bottom 5 records of the dataset

shape: shows the sum of rows and columns in the dataset

columns: it returns the column names of the dataset

info(): it shows the data tyoes of each columns and tells the null values of the dataset.

**Numpy:** NumPy is essential for handling numerical data in Python, providing efficient array operations and serving as the foundation for many other scientific computing libraries. It's a core component of the Python ecosystem for data science and machine learning.

NumPy arrays support advanced indexing and slicing operations, allowing you to access and manipulate specific elements or subsets of an array efficiently.

**1.) Clean Up Articles: We'll remove unnecessary clutter like punctuation and common words to focus on the main ideas.**

For cleaning the data we used various libraries that are-

**re (Regular Expressions):** Regular expressions are a powerful tool for pattern matching and string manipulation. In this project the article text are modified on certain patterns.

**nltk (Natural Language Toolkit):** NLTK provides tools for tasks like tokenization (splitting text into words), stemming (reducing words to their root form), and more.

It helps computers understand and work with human languages by breaking down text into smaller parts like words and sentences, and performing various operations on them.

**stopwords:** Stopwords are common words (like "the", "and", "is") that often don't add much meaning to a sentence. Removing them can help focus on the important words.

Stopwords are like the background noise in a conversation. They're the words we use all the time without really thinking about them. Removing them helps us focus on the important words that carry the real meaning.

After applying stopwords we will preprocess the data by applying function:

It is a common preprocessing pipeline for text data designed to clean and prepare textual data for further sentiment analysis. By removing noise like URLs, special characters, and stopwords, and breaking the text into manageable tokens, we create a cleaner and more focused dataset for downstream tasks.

* **Lowercasing**: Convert text to lowercase to ensure consistency in word representation.
* **Removing URLs:** Eliminate web links from the text, as they are not relevant to the analysis.
* **Removing Special Characters**: Get rid of symbols and punctuation marks, retaining only letters and numbers.
* **Tokenization**: Split the text into individual words to analyze them separately.
* **Filtering Stopwords:** Remove common words like "the", "and", and "is" that don't carry significant meaning.
* **Joining Words Back:** Reassemble the meaningful words into a coherent text after processing.

After preprocessing the data we will do Stemming:

Stemming is the process of reducing words to their root or base form. For example, "writing", "writes", and "wrote" would all be reduced to "write".

You can pass them through the PorterStemmer. The PorterStemmer applies a set of rules to reduce each word to its root form. These rules are based on linguistic research and aim to capture common patterns in English words.

And thus after stemming data is cleaned:

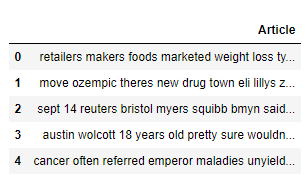


Fig: Preprocessed and cleaned data

**2.) Check the Mood: We'll build a system to figure out if an article is happy, sad, or neutral about a topic.**

For sentiment analysis we will use TextBlob library:

TextBlob is a Python library for processing textual data in a simple and intuitive way. It's built on top of NLTK and other libraries and provides a high-level interface for tasks like part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more.

TextBlob is easy to use, have wide range of NLP functionality including tokenization, POS tagging, noun phrase extraction, sentiment analysis, translation, and language detection.

TextBlob is a handy tool for anyone working with textual data in Python. It simplifies many common NLP tasks and provides a convenient interface for text analysis and processing.

We will use polarity to find the sentiments:

**Polarity Function (polarity):**

This function takes a piece of text as input and calculates its sentiment polarity.

It uses TextBlob, a library for natural language processing, to analyze the sentiment of the text.

The sentiment polarity score is a numeric value ranging from -1 (very negative) to 1 (very positive), with 0 representing neutral sentiment.

The function returns this polarity score.

**Sentiment Classification Function (sentiment):**

This function takes a numerical sentiment polarity score (label) as input.

It interprets the polarity score and classifies it into one of three categories: "Negative", "Neutral", or "Positive".

If the polarity score is less than 0, it's classified as "Negative".

If the polarity score is equal to 0, it's classified as "Neutral".

If the polarity score is greater than 0, it's classified as "Positive".

The function returns the corresponding sentiment label.

These functions provide a way to analyze the sentiment of text data, quantify it as a polarity score, and then classify it into understandable sentiment categories. This can be useful in various applications such as social media monitoring, customer feedback analysis, and opinion mining.

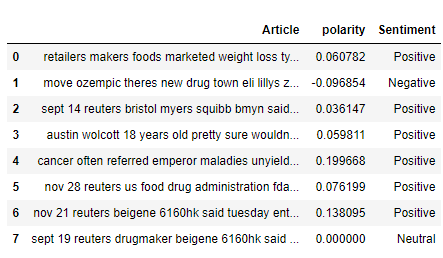


Fig: data with polarity and sentiments

**Visualization of Sentiments**

Now, we will visualize the sentiments got based on the polarity, thus we will use two libraries for visualization are:

**Matplotlib:** Matplotlib is a powerful Python library used for creating static, interactive, and animated visualizations in Python. It's widely used for generating plots, charts, histograms, scatterplots, and more. Think of it as a tool that helps you turn your data into easy-to-understand visual representations like graphs.

**Seaborn:** Seaborn is built on top of Matplotlib and provides a higher-level interface for creating attractive and informative statistical graphics. It's specifically designed for working with structured data and provides a simpler way to create complex visualizations compared to Matplotlib alone. Seaborn comes with built-in themes and color palettes, making it easy to create visually appealing plots with minimal effort.

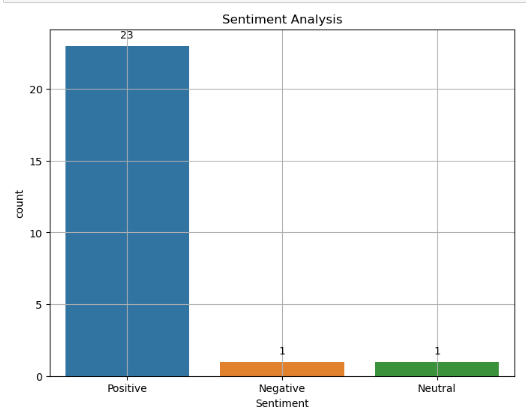


Fig: Distribution of sentiments through bar graph

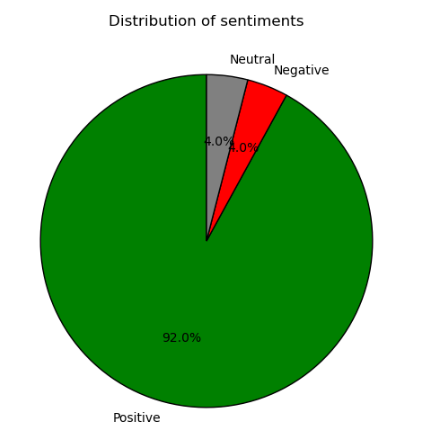


Fig: Distribution of sentiments through pie chart

**Visualization of Sentiments by WordCloud and plotting different logos**

Now we will visualize the data by wordcloud and will give different logos for wordcloud, thus we used three libraries for it are:-

**WordCloud:** This library is used to create word clouds, which are visual representations of text data where the size of each word indicates its frequency or importance. You can customize aspects like font, color, and shape to create unique word clouds.

**PIL (Python Imaging Library) with Image:** PIL is a library for working with images in Python. When combined with the Image module, it allows you to load and manipulate image files. In the context of word clouds, you can use PIL to load custom shapes and use them as masks for your word clouds, so the words appear within the contours of the shape.

**stylecloud:** This library builds on top of WordCloud and adds more customization options and styles to your word clouds. It allows you to create visually stunning word clouds with various color schemes, gradients, fonts, and orientations. Additionally, stylecloud provides integration with Google Fonts, making it easy to use a wide range of fonts in your word clouds.

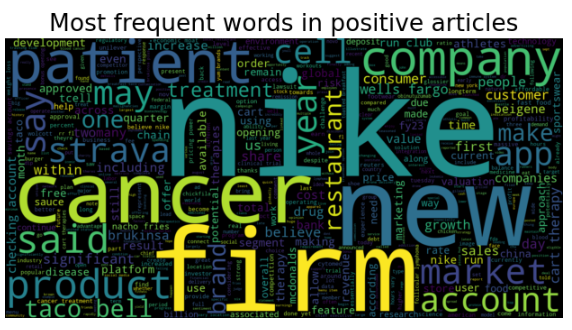


Fig: Most frequent words in positive articles



Fig: Leaf logo for Most frequent words in positive articles

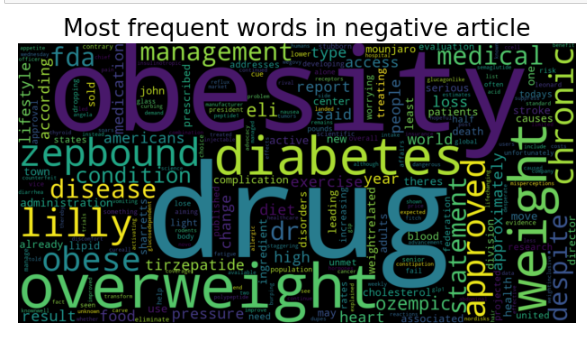


Fig: Most frequent words in negative article

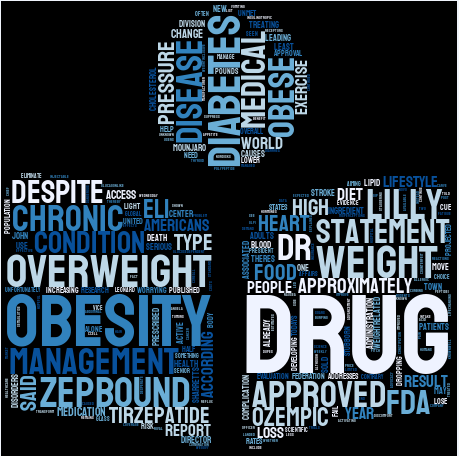


Fig: Book logo for Most frequent words in negative article

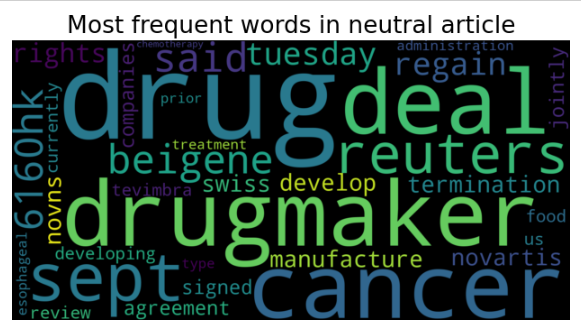


Fig: Most frequent words in neutral article

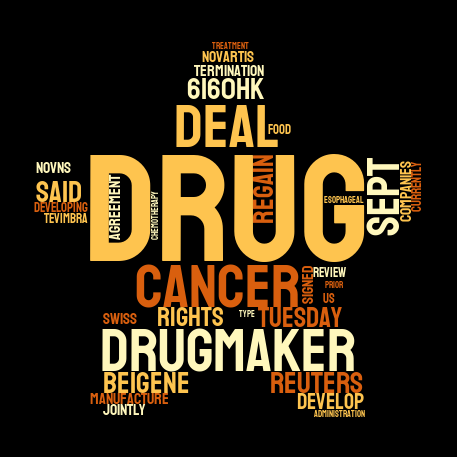


Fig: Star logo for Most frequent words in neutral article

**Model Evaluation**

Now based on polarity and sentiments we will build our model and then evaluate it.

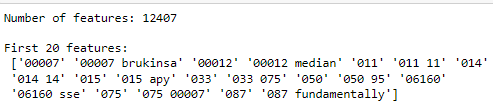
For model building we will extract features. To extract those feature we will use **CountVectorizer** for feature extraction.

After feature extraction we will build our model. Thus we will use **LogisticRegression** and then we will evaluate the models performance.

To evaluate the model we will check for **Accuracy** and display **confusion** **matrix**.

**CountVectorizer:** CountVectorizer is a class within scikit-learn that converts a collection of text documents into numerical vectors (arrays) representing the frequency of each word (or token) in the documents. Each column in the matrix represents a unique word in the corpus, and each row represents a document, with the cell values indicating the frequency of each word in the corresponding document.

CountVectorizer is a tool for converting text data into numerical features that can be used as input for machine learning models, particularly in tasks like text classification, clustering, or information retrieval.



Snapshot: Features extracted by using CountVectorizer

After feature extraction we will Train Test Split the data:-

Train Test Split is divide into two parts, one for training a machine learning model and the other for testing its performance. The training set (x\_train and y\_train) is used to teach the model, and the testing set (x\_test and y\_test) is used to evaluate how well it predicts unseen data.

After train test split we will build our model:-

We used scikit-learn's LogisticRegression class to create a logistic regression model (lr). Then, it trains the model using the training data (x\_train and y\_train) with the fit method. In simple terms, it's building a predictive model that learns from the training data, aiming to predict the target variable (y) based on the input features (x).

And then we will make predictions on the testing data (x\_test). It predicts the target variable (y) for the input features (x\_test).

Now we will evaluate our model:-

To evaluate the model, the metrics module of scikit-learn is used:-

**accuracy\_score:** This function computes the accuracy of the classification model. It compares the predicted labels with the true labels and returns the fraction of correctly classified samples.

**classification\_report:** This function generates a text report showing the main classification metrics, such as precision, recall, F1-score, and support, for each class in the classification model. It provides a detailed summary of the model's performance.

**confusion\_matrix:** This function computes a confusion matrix, which is a table that shows the number of true positive, false positive, true negative, and false negative predictions made by the model. It's useful for evaluating the performance of a classification model, especially when dealing with imbalanced classes.

**ConfusionMatrixDisplay:** This class visualizes the confusion matrix generated by the confusion\_matrix function. It provides a graphical representation of the confusion matrix, making it easier to interpret and analyze.



Fig: Accuracy score of the model

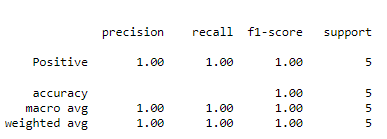


Fig: Classification report of the evaluated model

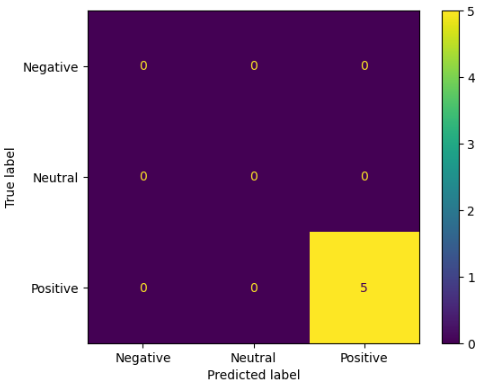


Fig: Confusion matrix

**3.) Find Connections: We'll look for common themes across many articles, like seeing a pattern in tech news.**

**4.) Aspect Analysis (Optional): Gain deeper insights by understanding the sentiment towards different aspects of the news.**

To Analyse the connections and find out the topic/themes and aspects and sentiments we have used **spacy** and **genism** libraries:-

**spacy:** Spacy is a powerful NLP library in Python that's used for various tasks like tokenization, part-of-speech tagging, named entity recognition, and dependency parsing. It provides pre-trained models and efficient processing pipelines for working with text data.

**gensim:** Gensim is a library for unsupervised topic modeling and natural language understanding. It's widely used for tasks like document similarity analysis, document clustering, and topic modeling. The corpora module within gensim provides tools for building and handling corpora, which are collections of text documents used for training topic models.

**To Load SpaCy Model:** **spacy.load('en\_core\_web\_sm')** loads the English language model "en\_core\_web\_sm" provided by SpaCy. This model includes linguistic annotations and trained pipelines for various NLP tasks such as tokenization, part-of-speech tagging, named entity recognition, and syntactic parsing.

**Topic Modeling:-**

For topic modeling we will build a function and do tokenization, create dictionary and corpus, perform LDA and retrieve the topics

* **Tokenization:** Break down the text into individual words, removing unnecessary words like "the", "is", etc., and converting words to their base forms.
* **Create Dictionary and Corpus:** Collect all unique words from the tokenized text and represent the text as a bag of words, a simple way of counting word occurrences.
* **Perform LDA:** Use the bag-of-words representation to uncover underlying topics in the text using Latent Dirichlet Allocation (LDA), a statistical model that groups similar words into topics.
* **Get Topics:** Retrieve and present the identified topics, each represented by a list of words, which helps in understanding the main themes or subjects within the text.

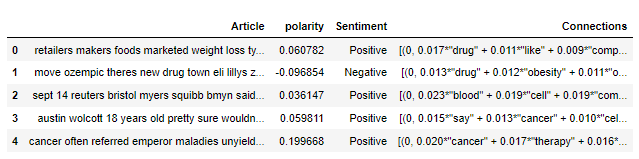


Fig: Retieved the connections from articles

After getting connections we will analyse the data and the provide the topic/themes for each article:-

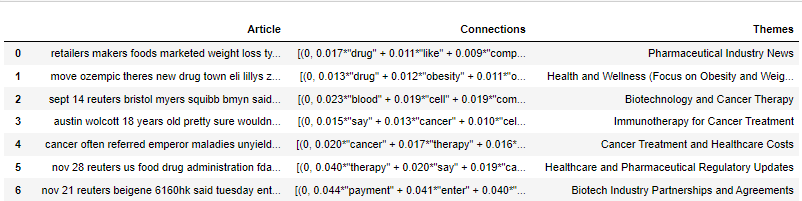


Fig: Provided Themes as per connections

After providing themes we will give aspects and sentiments as per aspects for the themes:-



Fig: As per Themes- Aspects and Sentiments Analysis

**Conclusion:-**

"News Analysis Project: Uncover What Matters!!!" aims to enhance our comprehension of news articles through three primary objectives. Firstly, by cleansing articles by elements such as punctuation and common words. Secondly, it seeks to develop a mood detection system to ascertain whether an article expresses happiness, sadness, or neutrality towards a given topic, providing insights into its tone. Lastly, by identifying recurring themes across numerous articles, particularly in areas like Medical and pharma industry, Food Industry, Airline industry, etc; the project aims to uncover patterns and connections, offering deeper understanding and valuable insights into current events and trends. The models accuracy is 100% when it comes to figuring out the mood of articles and picking out the main points. And we've made it easy to understand with cool visuals like bar graphs and word clouds.

Overall, these efforts underscore the project's commitment to extracting meaningful information from news articles and facilitating a more nuanced understanding of the world around us.