FAKE NEWS CLASSIFICATION

CAPSTONE REPORT SUBMITTED TO THE BHARATHIAR UNIVERSITY

FOR THE AWARD OF THE DEGREE OF

MASTER OF BUSINESS ADMINISTRATION

By

KIRUTHIKA. M

1P22MB011

Under the Guidance of

Dr. S.Suganya Ph.D.

Associate Professor

Estd 1986

RVS CAS

Building Intellectual Capital

RVS COLLEGE OF ARTS AND SCIENCE (AUTONOMOUS), SULUR

DEPARTMENT OF MANAGEMENT STUDIES - PG

Coimbatore – 641 402

Tamil Nadu, INDIA

March 2024

CERTIFICATE

This is to certify that the capstone report, entitled **“FAKE NEWS CLASSIFICATION”**, submitted to the Bharathiar University, in partial fulfilment of the requirements for the award of the **DEGREE OF MASTER OF BUSINESS ADMINISTRATION**, is a record of original work done by **Ms. KIRUTHIKA. M** during the period January 2024 – March 2024 of her capstone project in RVS College of Arts and Science (Autonomous), Department of Management Studies – PG, Sulur, Coimbatore - 641402, under my supervision and guidance and the capstone report has not formed the basis for the award of any Degree / Diploma / Associateship / Fellowship or other similar title of any candidate of any University.

Date: 28/03/2024

Director   Signature of the Guide

Date of Viva-voce held on ____________________________.

Internal Examiner   External Examiner

DECLARATION

I, **KIRUTHIKA. M,** hereby declare that the capstone, entitled **“FAKE NEWS CLASSIFICATION”,** submitted to the Bharathiar University, in partial fulfilment of the requirements for the award of the **DEGREE OF MASTER OF BUSINESS ADMINISTRATION** is a record of original and independent research work done by me during the period January 2024 – March 2024, under the supervision and guidance of **Dr. S. Suganya** RVS College of Arts and Science (Autonomous), Department of Management Studies – PG, Sulur, Coimbatore – 641 402 and it has not formed the basis for the award of any other Degree / Diploma / Associateship / Fellowship or other similar title to any candidate of any University.

Date:28/03/2024 Signature of the Candidate

ACKNOWLEDGEMENT

The successful completion of this capstone report would not have been possible without the support and assistance of many individuals and organization. I would like to take this opportunity to offer my earnest admiration to each and every one of them.

First and foremost, I thank Almighty and my parents for having bestowed their greatest blessing on me to complete this capstone successfully.

I express my indebtedness and gratefulness to **Dr. THULASIVELU K**, **Director**, RVS College of Arts and Science (Autonomous), Department of Management Studies – PG, Sulur, Coimbatore for his continuous support and encouragement and permission to pursue my capstone.

I would like to express my deep and sincere gratitude to my institution guide. **Dr.S.Suganya – Associate Professor,** RVS College of Arts and Science (Autonomous), Department of Management Studies – PG, Sulur, Coimbatore for her valuable advice, timely encouragement and educative guidance for completing the capstone successfully and for the instructions for preparing this report standard as per the norms and values.

I express my deep sense of gratitude to **Mr. Duke, Manager – Corporate Connect**, RVS College of Arts and Science (Autonomous), Department of Management Studies – PG, Sulur, Coimbatore for helping me by all means and ways for the successful completion of my capstone.

My sincere gratitude to **KAGGLE, STACKOVERFLOW** for unremitting support rendered to complete my capstone successfully.

I perceive this opportunity as a big milestone in my career development. I will strive to use those gained skills and knowledge in the best possible way, and I will continue to work on their improvement, in order to attain desired career objectives.

ABSTRACT

The proliferation of misinformation and fake news in digital media poses a significant challenge to society, affecting public discourse, political landscapes, and societal trust. In response, this project aims to develop and deploy machine learning models for the automatic classification of news articles as either real or fake. Leveraging a dataset of labelled news articles, various classification algorithms including Logistic Regression, , Decision Trees, and Random Forest are trained and evaluated.

A comprehensive preprocessing pipeline is constructed to handle diverse feature types, including textual content, metadata, and sentiment analysis. The pipeline incorporates techniques such as TF-IDF vectorization, one-hot encoding, and standard scaling to transform raw data into a format suitable for machine learning models.

Performance evaluation metrics such as accuracy, precision, recall, and F1 score are utilized to assess the models' effectiveness in distinguishing between real and fake news. Additionally, confusion matrices and feature importance analysis provide insights into model behaviour and feature relevance.

Results indicate varying degrees of success across different models, with logistic regression demonstrating robust performance in terms of accuracy and balanced precision-recall trade-offs. Decision tree models showcase interpretability and depth control through cost complexity pruning. However, the Random Forest model exhibits a bias towards the majority class, necessitating further optimization.

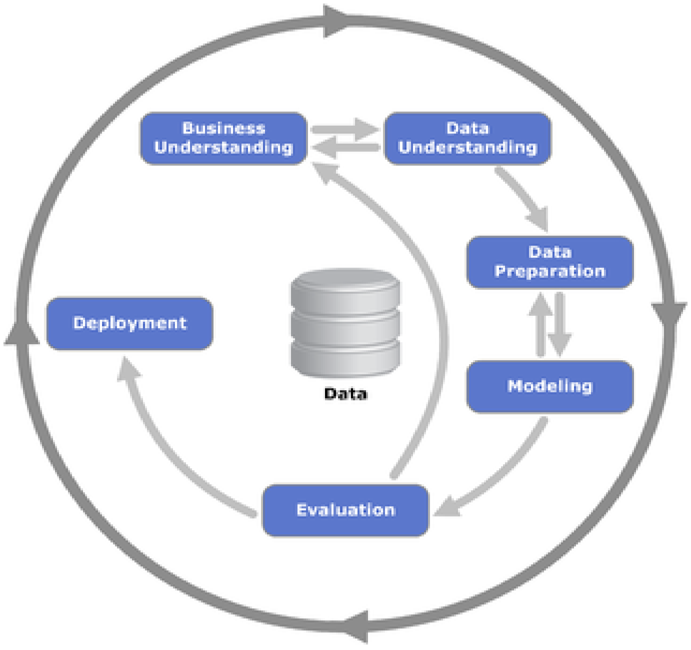
Overall, this project contributes to the ongoing efforts in combatting fake news by providing a framework for automated classification, enabling the identification and mitigation of misinformation at scale.

CONTENTS

INTRODUCTION

METHODOLOGY:

In the world of data science, a structured approach is crucial to guide projects from inception to completion. Enter the CRISP-DM (Cross-Industry Standard Process for Data Mining) process — a robust, systematic framework for data mining projects. Its versatility has made it the industry standard for both small- and large-scale projects across various sectors.



The **CR**oss **I**ndustry **S**tandard **P**rocess for **D**ata **M**ining (*CRISP-DM*) is a process model that serves as the base for a data science process. It has six sequential phases:

1. Business understanding – What does the business need?
2. Data understanding – What data do we have / need? Is it clean?
3. Data preparation – How do we organize the data for modeling?
4. Modeling – What modeling techniques should we apply?
5. Evaluation – Which model best meets the business objectives?
6. Deployment – How do stakeholders access the results?

The CRISP-DM (Cross-Industry Standard Process for Data Mining) methodology is a robust framework that provides a structured approach to guide organizations through the entire process of data mining or machine learning projects. Its importance lies in its ability to bring clarity, efficiency, and effectiveness to the often complex and iterative process of extracting valuable insights from data. By following the CRISP-DM methodology, organizations can systematically address key stages of a project, including business understanding, data understanding, data preparation, modeling, evaluation, and deployment. This structured approach facilitates collaboration among different stakeholders, such as business analysts, data engineers, and data scientists, ensuring that project objectives align with business goals and that insights generated are actionable and valuable. Additionally, CRISP-DM emphasizes the iterative nature of data mining projects, allowing for continuous improvement and refinement of models based on feedback and evaluation results. Overall, the adoption of CRISP-DM methodology promotes transparency, repeatability, and accountability in data-driven decision-making processes, ultimately leading to more successful and impactful outcomes for organizations.

BUSINESS UNDERSTANDING:

In the media and journalism industry, **fake news detection models** play a crucial role in maintaining information accuracy, credibility, and public trust. Let’s explore how these models are utilized and the benefits they offer:

Early Detection of Fake News:

Media organizations and journalists need to act swiftly to prevent the spread of fake news. Detection models allow them to identify potentially false information within minutes after it starts propagating. By catching fake news early, media outlets can avoid inadvertently amplifying misinformation and protect their reputation.

Improving Editorial Standards:

Fake news detection models help media professionals maintain high editorial standards. By integrating these models into their workflows, journalists can verify the authenticity of news sources and stories. This ensures that accurate information reaches the public, enhancing trust in media organizations.

Fact-Checking and Verification:

Journalists often rely on fact-checking to verify claims made in news articles. Fake news detection models assist in identifying suspicious content, allowing journalists to investigate further. By cross-referencing information with reliable sources, journalists can validate or debunk news stories.

Mitigating Disinformation Campaigns:

Fake news detection models are essential in countering disinformation campaigns. Media outlets can use these models to identify coordinated efforts to spread false narratives. By exposing such campaigns, journalists contribute to public awareness and resilience against misinformation.

Enhancing Audience Trust:

When media organizations actively combat fake news, they demonstrate their commitment to accuracy and integrity. By transparently using detection models, they build trust with their audience, who rely on credible information for informed decision-making.

Customized Alerts and Notifications:

Some media platforms integrate fake news detection algorithms to provide personalized alerts to users. These alerts notify users when they encounter potentially misleading content, empowering them to critically evaluate information.

Fake news detection models empower media professionals to uphold journalistic ethics, safeguard public discourse, and contribute to a more informed society. By leveraging these tools, the media industry can combat misinformation effectively and maintain its vital role as an information gatekeeper.

TOOLS FOR ANALYTICS

In the dynamic world of data analytics, the right tools and technologies play a pivotal role in transforming raw data into actionable insights. From powerful programming languages to intuitive visualization software, this article introduces the popular tools and technologies that empower data analysts and data scientists to extract meaningful information from vast datasets. Let’s delve into the world of Python, R, Tableau, and Power BI, and explore how each contributes to the data analytics ecosystem.

Here, in this project Python is used for analytics

Python:

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

- **Pandas**: Utilized for data manipulation and analysis.
- **NumPy**: Employed for numerical computing and array operations.
- **Scikit-learn**: Applied for machine learning tasks, including model training, evaluation, and prediction.
- **Matplotlib** and **Seaborn**: Utilized for data visualization and exploratory analysis.
- **NLTK (Natural Language Toolkit)**: NLTK provides easy-to-use interfaces to over 50 corpora and lexical resources, including WordNet. It is often used for text processing, tokenization, stemming, tagging, parsing, and more.

DATA UNDERSTANDING

What is Data?

In computing, data is information that has been translated into a form that is efficient for movement or processing. Relative to today's computers and transmission media, data is information converted into binary digital form. It is acceptable for data to be used as a singular subject or a plural subject. Raw data is a term used to describe data in its most basic digital format.

Computers represent data, including video, images, sounds and text, as binary values using patterns of just two numbers: 1 and 0. A bit is the smallest unit of data, and represents just a single value. A byte is eight binary digits long. Storage and memory is measured in megabytes and gigabytes.

Source of Dataset:

The dataset utilized in this project was sourced from Kaggle, a platform known for hosting datasets and competitions related to various domains. Kaggle provides a diverse range of datasets contributed by individuals, organizations, and communities worldwide. The specific dataset used in this project is titled "IFND dataset" and can be accessed at the following URL on Kaggle: **https://www.kaggle.com/datasets/sonalgarg174/ifnd-dataset** .

Kaggle offers a valuable resource for data scientists and researchers, facilitating access to curated datasets and fostering collaboration within the data science community. Kaggle is the world's largest data science and machine learning community. More than ten million registered users visit Kaggle to learn, find data, compete, and collaborate on the cutting edge of machine learning. Kaggle's mission is to help the world learn from data.

IFND Dataset:

This IFND dataset covers news pertaining to India only. This dataset is created by scraping Indian fact checking websites. The dataset contains two types of news fake and real News. This dataset was collected from real-world sources. The truthful news and fake news were collected from different reliable fact-checking websites. The dataset contains different types of articles on different topics, however, the majority of news focus on political news.

The above CSV files is comma-separated file and have the following columns:

* id- Unique identifier for each news

* Statement- Title of the news article

* Image- Image URL

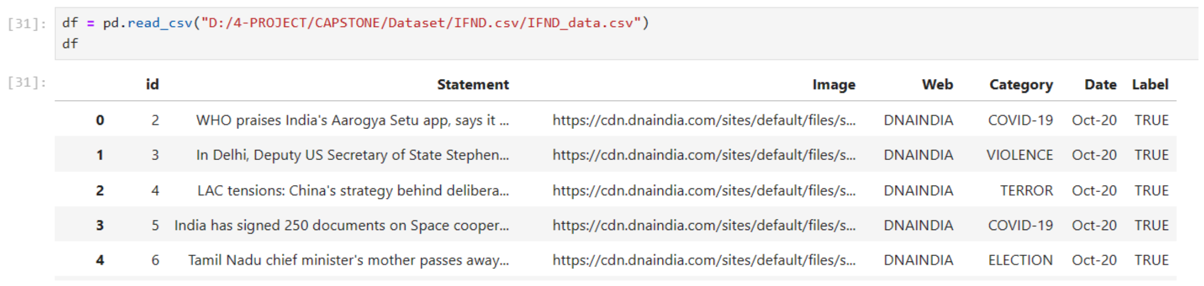
* Category-Topic of news

* Date- Date of news

* Label- True and Fake news

Data:

To understand the contents and structure of the dataset, print the Data Frame df using print(df) or df.head() to display the first few rows.

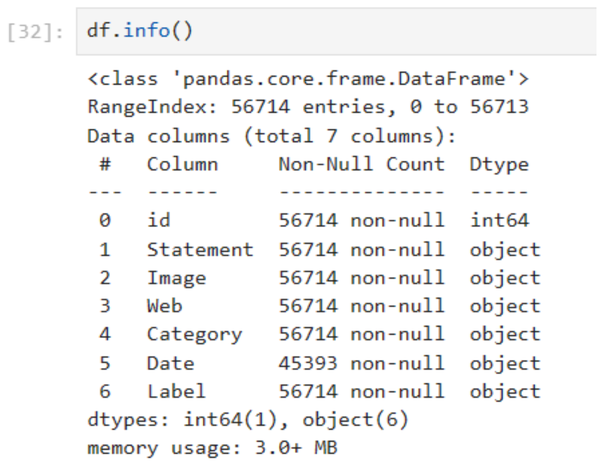


```
[31]: df = pd.read_csv("D:/4-PROJECT/CAPSTONE/Dataset/IFND.csv/IFND_data.csv")
      df
```

[31]:

	id	Statement	Image	Web	Category	Date	Label
0	2	WHO praises India's Aarogya Setu app, says it ...	https://cdn.dnaindia.com/sites/default/files/s...	DNAINDIA	COVID-19	Oct-20	TRUE
1	3	In Delhi, Deputy US Secretary of State Stephen...	https://cdn.dnaindia.com/sites/default/files/s...	DNAINDIA	VIOLENCE	Oct-20	TRUE
2	4	LAC tensions: China's strategy behind delibera...	https://cdn.dnaindia.com/sites/default/files/s...	DNAINDIA	TERROR	Oct-20	TRUE
3	5	India has signed 250 documents on Space cooper...	https://cdn.dnaindia.com/sites/default/files/s...	DNAINDIA	COVID-19	Oct-20	TRUE
4	6	Tamil Nadu chief minister's mother passes away...	https://cdn.dnaindia.com/sites/default/files/s...	DNAINDIA	ELECTION	Oct-20	TRUE

From the df.info() output, we can gather several insights about the dataset

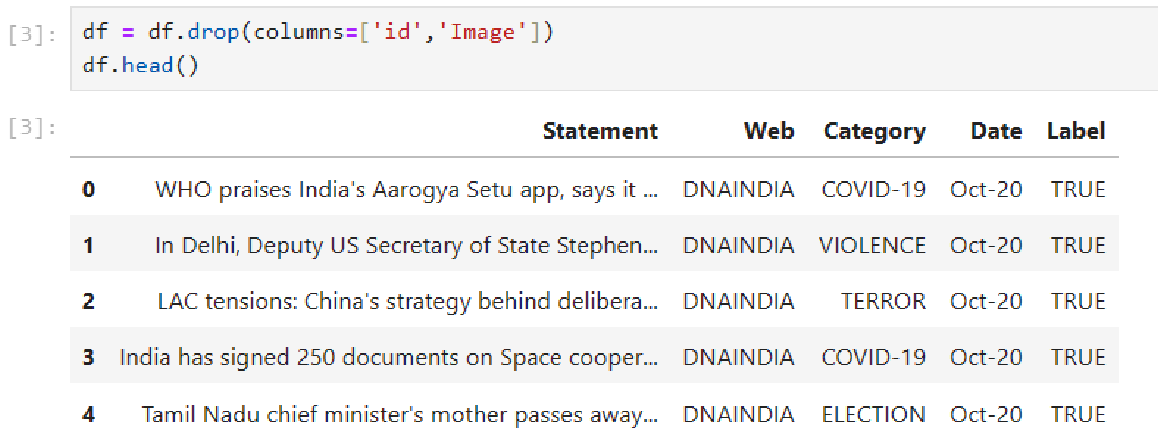


```
[32]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 56714 entries, 0 to 56713
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   id         56714 non-null  int64 
 1   Statement  56714 non-null  object
 2   Image      56714 non-null  object
 3   Web        56714 non-null  object
 4   Category   56714 non-null  object
 5   Date       45393 non-null  object
 6   Label      56714 non-null  object
dtypes: int64(1), object(6)
memory usage: 3.0+ MB
```

The dataset contains 56,714 entries (rows) and 7 columns. The dataset consists of 7 columns. Considering the data types, 'id' is appropriately represented as an integer, while the other columns are stored as objects. Depending on the analysis or operations to be performed, data types might need to be converted accordingly. For instance, 'Date' could be converted to a datetime data type for easier manipulation and analysis. The 'Date' column has missing values, as indicated by the difference in the counts between the non-null count and the total count.

Drop Columns:

Two columns, 'id' and 'Image', are dropped from the Data Frame using the drop() method with the columns parameter. The columns parameter accepts a list of column names to be dropped. The 'id' column is typically an identifier column, not be required for analysis or modelling purposes. The 'Image' column contains URLs pointing to images associated with the statements or news articles, which is not necessary for the analysis at hand.



```
[3]: df = df.drop(columns=['id','Image'])
     df.head()
```

```
[3]:
                                          Statement       Web  Category    Date  Label
0      WHO praises India's Aarogya Setu app, says it ...  DNAINDIA  COVID-19  Oct-20  TRUE
1      In Delhi, Deputy US Secretary of State Stephen...  DNAINDIA  VIOLENCE  Oct-20  TRUE
2       LAC tensions: China's strategy behind delibera...  DNAINDIA    TERROR  Oct-20  TRUE
3   India has signed 250 documents on Space cooper...  DNAINDIA  COVID-19  Oct-20  TRUE
4    Tamil Nadu chief minister's mother passes away...  DNAINDIA  ELECTION  Oct-20  TRUE
```

Description:

id: The 'id' variable serves as a unique identifier for each entry in the dataset. It provides a reference point for identifying and accessing individual records within the dataset.

Statement: This variable contains the textual content or headlines of statements or news articles. It serves as the primary content or information conveyed by each entry in the dataset.

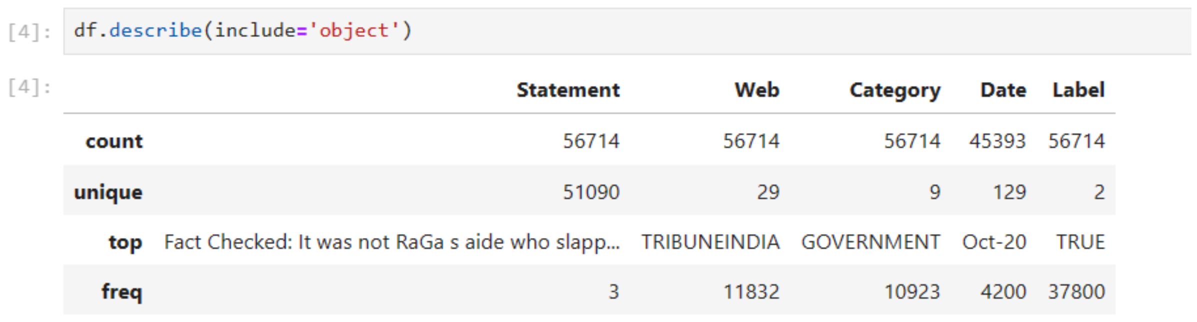
Web: Represents the source or website where the statements or news articles were published. Helps identify the origin or publication source of the statements or articles.

Category: Indicates the categorization or topic of the statements or news articles. Provides insight into the subject matter or theme addressed by each entry.

Date: Represents the publication date or occurrence date of the statements or news articles. Facilitates temporal analysis and understanding of when the statements or articles were published or occurred.

Label: Contains a binary label indicating the validity or truthfulness of the statements or news articles. Helps classify the statements or articles as true or fake, providing a basis for analysis or classification tasks.

Image: The 'Image' variable contains URLs pointing to images associated with the statements or news articles. These images may provide visual context or supplementary information related to the content of the statements or articles.



```
[4]: df.describe(include='object')
```

[4]:

	Statement	Web	Category	Date	Label
count	56714	56714	56714	45393	56714
unique	51090	29	9	129	2
top	Fact Checked: It was not RaGa s aide who slapp...	TRIBUNEINDIA	GOVERNMENT	Oct-20	TRUE
freq	3	11832	10923	4200	37800

Since, there are only object type variables except ID, the data description is provided for object type variables. The categorical variable summary highlights essential aspects of the dataset: the 'Statement' column contains 56,714 entries with 51,090 unique statements, the 'Web' column features 29 unique sources with 'TRIBUNEINDIA' being the most frequent, 'Category' showcases nine distinct categories with 'GOVERNMENT' as the top category, 'Date' reveals 45,383 entries spread across 129 unique dates with 'Oct-20' appearing most frequently, and the 'Label' column includes 56,714 entries with 'TRUE' being the dominant label at 37,800 occurrences. This summary provides a snapshot of the dataset's categorical attributes, aiding in understanding its distribution and characteristics.

DATA PREPARATION

Data preparation is a crucial step in the data analysis process, aimed at ensuring that the dataset is clean, reliable, and suitable for analysis. This involves identifying and addressing various data issues, including missing values (NA values), outliers, and inconsistencies. Addressing NA values, outliers, and inconsistencies is essential for ensuring the quality and reliability of data for analysis. By implementing appropriate strategies and techniques, data preparers can enhance the integrity and effectiveness of downstream analyses and modeling tasks.

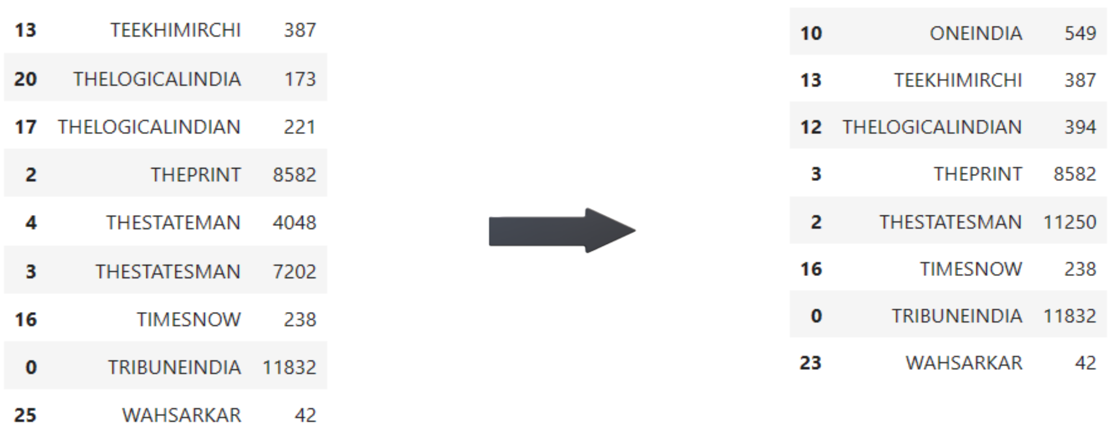
Inconsistent Data:

Inconsistencies in the data arise when there are discrepancies, errors, or contradictions within the dataset. Ensuring consistency in data formats, units of measurement, and coding schemes. Verifying data integrity and correctness through validation checks, range constraints, or cross-validation with external sources. Correcting errors, typos, or formatting issues manually or through automated cleaning procedures. Resolving discrepancies between different sources or versions of data to ensure data consistency.

<u>Spelling Mistakes:</u>

Inconsistent data is data that is inconsistent, conflicted, or incompatible within a dataset or across many datasets. Data inconsistencies can occur for a variety of reasons, including mistakes in data entry, data processing, or data integration. These discrepancies might show as disagreements in data element values, formats, or interpretations. Inconsistent data can lead to faulty analysis, untrustworthy outcomes, and data management challenges.

```
[36]: df['Web'] = df['Web'].replace({'INDUSTANTIMES':'HINDUSTANTIMES','THELOGICALINDIA':'THELOGICALINDIAN', 'THESTATEMAN':'THESTATESMAN'})
```



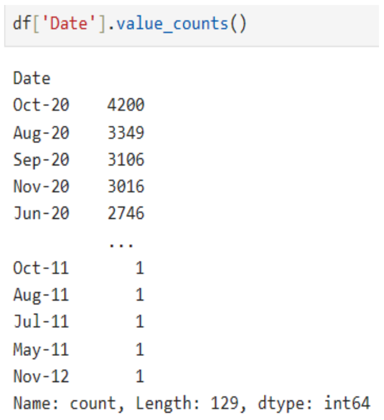
13	TEEKHIMIRCHI	387
20	THELOGICALINDIA	173
17	THELOGICALINDIAN	221
2	THEPRINT	8582
4	THESTATEMAN	4048
3	THESTATESMAN	7202
16	TIMESNOW	238
0	TRIBUNEINDIA	11832
25	WAHSARKAR	42

10	ONEINDIA	549
13	TEEKHIMIRCHI	387
12	THELOGICALINDIAN	394
3	THEPRINT	8582
2	THESTATESMAN	11250
16	TIMESNOW	238
0	TRIBUNEINDIA	11832
23	WAHSARKAR	42

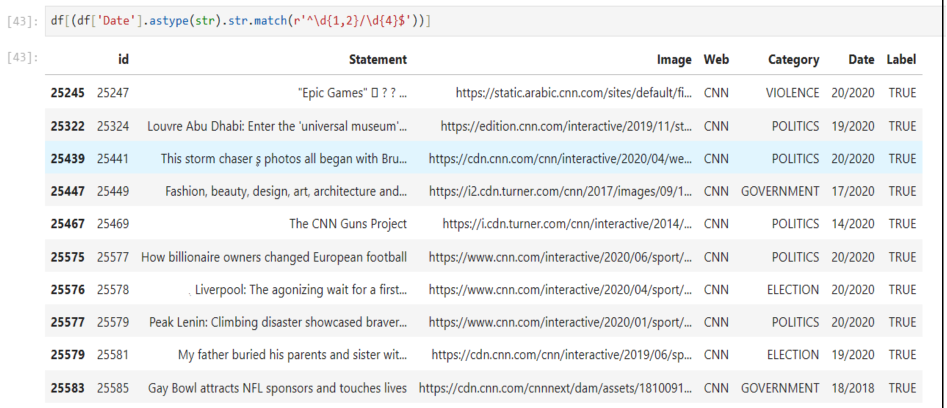
Corrects spelling mistakes in the 'Web' column using the replace() method. It replaces the misspelled values with their correct counterparts using a dictionary where keys are misspelled values and values are correct versions.

Similarly corrects a misleading category label ('MISLEADIND') in the 'Category' column using the replace() method. It replaces the misleading label with the correct one ('MISLEADING').

<u>Formatting Data Types:</u>



```
df['Date'].value_counts()

Date
Oct-20    4200
Aug-20    3349
Sep-20    3106
Nov-20    3016
Jun-20    2746
          ...
Oct-11       1
Aug-11       1
Jul-11       1
May-11       1
Nov-12       1
Name: count, Length: 129, dtype: int64
```

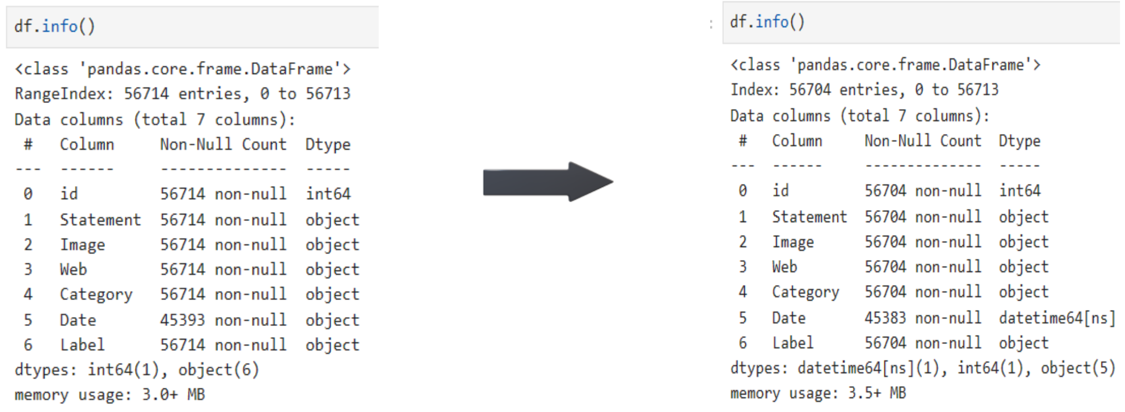


```
[43]: df[(df['Date'].astype(str).str.match(r'^\d{1,2}/\d{4}$'))]
```

[43]:

	id	Statement	Image	Web	Category	Date	Label
25245	25247	"Epic Games" ▯ ? ? ...	https://static.arabic.cnn.com/sites/default/fi...	CNN	VIOLENCE	20/2020	TRUE
25322	25324	Louvre Abu Dhabi: Enter the 'universal museum'...	https://edition.cnn.com/interactive/2019/11/st...	CNN	POLITICS	19/2020	TRUE
25439	25441	This storm chaser ş photos all began with Bru...	https://cdn.cnn.com/cnn/interactive/2020/04/we...	CNN	POLITICS	20/2020	TRUE
25447	25449	Fashion, beauty, design, art, architecture and...	https://i2.cdn.turner.com/cnn/2017/images/09/1...	CNN	GOVERNMENT	17/2020	TRUE
25467	25469	The CNN Guns Project	https://i.cdn.turner.com/cnn/interactive/2014/...	CNN	POLITICS	14/2020	TRUE
25575	25577	How billionaire owners changed European football	https://www.cnn.com/interactive/2020/06/sport/...	CNN	POLITICS	20/2020	TRUE
25576	25578	Liverpool: The agonizing wait for a first...	https://www.cnn.com/interactive/2020/04/sport/...	CNN	ELECTION	20/2020	TRUE
25577	25579	Peak Lenin: Climbing disaster showcased braver...	https://www.cnn.com/interactive/2020/01/sport/...	CNN	POLITICS	20/2020	TRUE
25579	25581	My father buried his parents and sister wit...	https://cdn.cnn.com/cnn/interactive/2019/06/sp...	CNN	ELECTION	19/2020	TRUE
25583	25585	Gay Bowl attracts NFL sponsors and touches lives	https://cdn.cnn.com/cnnnext/dam/assets/1810091...	CNN	GOVERNMENT	18/2018	TRUE

Initially, the goal was to convert the data type of the 'Date' column from object to datetime to facilitate further analysis. However, attempts to directly convert the 'Date' column to datetime format encountered errors due to the presence of dates in the 'DD/YYYY' format, which lacks the month information and is deemed insufficient for subsequent analysis. Consequently, rows containing dates in the 'DD/YYYY' format were filtered out from the dataset, as they were deemed unusable for further processing. Subsequently, the 'Date' column was successfully converted to datetime format using the pd.to_datetime() function, specifying the input date format as '%b-%y' to handle dates represented in the 'MMM-YY' format (e.g., 'Jan-21'). This preprocessing step ensures consistency in date representation and enhances the dataset's suitability for subsequent analytical tasks.



```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 56714 entries, 0 to 56713
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   id         56714 non-null  int64
 1   Statement  56714 non-null  object
 2   Image      56714 non-null  object
 3   Web        56714 non-null  object
 4   Category   56714 non-null  object
 5   Date       45393 non-null  object
 6   Label      56714 non-null  object
dtypes: int64(1), object(6)
memory usage: 3.0+ MB
```

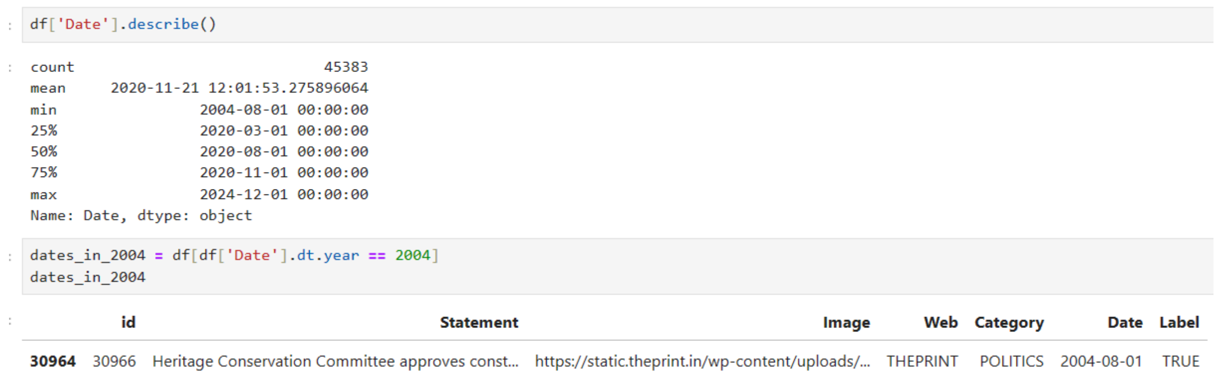
```
df.info()

<class 'pandas.core.frame.DataFrame'>
Index: 56704 entries, 0 to 56713
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   id         56704 non-null  int64
 1   Statement  56704 non-null  object
 2   Image      56704 non-null  object
 3   Web        56704 non-null  object
 4   Category   56704 non-null  object
 5   Date       45383 non-null  datetime64[ns]
 6   Label      56704 non-null  object
dtypes: datetime64[ns](1), int64(1), object(5)
memory usage: 3.5+ MB
```

Outliers:

Outliers are data points that significantly deviate from the rest of the dataset and may skew statistical analyses or machine learning models.

Visualization: Plotting box plots, histograms, or scatter plots to visually identify outliers.

Statistical methods: Calculating summary statistics such as mean, median, standard deviation, and interquartile range (IQR) to detect outliers.



```
df['Date'].describe()
```

```
count                           45383
mean    2020-11-21 12:01:53.275896064
min               2004-08-01 00:00:00
25%               2020-03-01 00:00:00
50%               2020-08-01 00:00:00
75%               2020-11-01 00:00:00
max               2024-12-01 00:00:00
Name: Date, dtype: object
```

```
dates_in_2004 = df[df['Date'].dt.year == 2004]
dates_in_2004
```

	id	Statement	Image	Web	Category	Date	Label
30964	30966	Heritage Conservation Committee approves const...	https://static.theprint.in/wp-content/uploads/...	THEPRINT	POLITICS	2004-08-01	TRUE

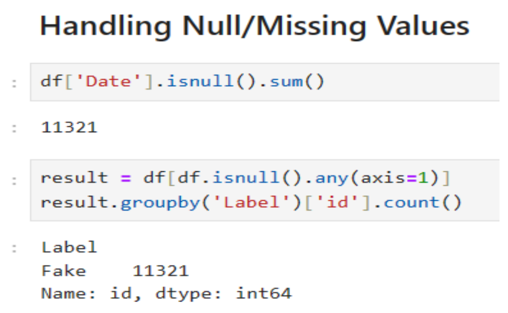
Upon examining the descriptive statistics for the 'Date' column, it was observed that the minimum and maximum dates were distinct from the majority of the data, which predominantly fell within the year 2020. Further investigation revealed that there was only one observation from the year 2004, while there were over 7000 observations from the year 2024. Given this disparity, it was determined that the data from 2004 constituted an outlier and was removed from the dataset to ensure consistency and integrity in subsequent analyses. Consequently, the dataset was refined by excluding observations from the year 2004, resulting in a more representative sample for further analysis.

Null/ Missing Values:

NA values, or missing values, occur when data is not recorded or unavailable for certain observations or variables.

Common approaches to handling NA values include:

Replacing NA values with estimated or calculated values based on the data distribution (e.g., mean, median, mode). Deleting observations or variables with NA values if they are deemed insignificant or cannot be imputed accurately. Using more sophisticated methods such as k-nearest neighbours (KNN) imputation or predictive modeling to impute missing values based on relationships within the data.


Handling Null/Missing Values

```
df['Date'].isnull().sum()
```

```
11321
```

```
result = df[df.isnull().any(axis=1)]
result.groupby('Label')['id'].count()
```

```
Label
Fake     11321
Name: id, dtype: int64
```

Upon inspection of the 'Date' column, it was determined that it contained missing values, with a total of 11,321 observations lacking date information. Subsequently, an examination of the rows containing missing values revealed that all instances belonged to the category of fake news. This finding suggests a potential association between the absence of date information and the authenticity of news articles, with fake news articles more likely to lack date information compared to genuine news articles. **Hence, the presence of missing date values may serve as an indicator for identifying fake news articles within the dataset.**

Case Conversion:

The provided steps involve standardizing the representation of labels within the dataset's 'Label' column. Initially, an examination of the label distribution is conducted to understand the prevalence of different label values. Subsequently, a case conversion operation is performed to ensure uniformity in label representation. By converting all labels to uppercase, potential inconsistencies arising from variations in capitalization are resolved. This standardization ensures that identical labels are consistently represented, simplifying data analysis and interpretation. Finally, a reassessment of the label distribution confirms the successful conversion, validating the uniformity achieved. Overall, this process enhances data coherence and quality, facilitating more robust analysis and insights from the dataset.

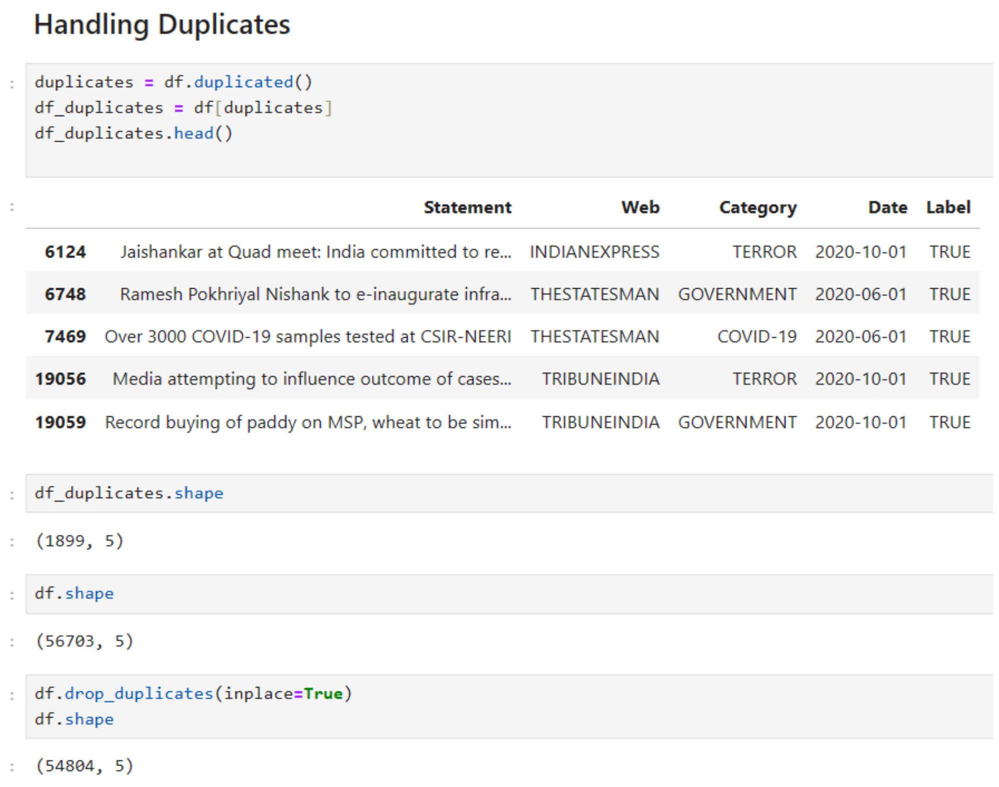
Filtering:

The 'id' column typically represents a unique identifier for each row in the dataset. It serves as an internal identifier and may not contribute directly to the analysis or modeling process. Therefore, it is dropped to streamline the dataset and remove unnecessary information.

The 'Image' column likely contains URLs or file paths pointing to image files associated with each statement. Since the analysis or modeling task may not require the image data, the column is dropped to reduce the dimensionality of the dataset and focus solely on relevant features.

This filtering process aims to refine the dataset by removing redundant or extraneous columns, resulting in a more manageable and relevant dataset for subsequent analysis or modeling tasks.

Handling Duplicates:



Handling Duplicates

```
duplicates = df.duplicated()
df_duplicates = df[duplicates]
df_duplicates.head()
```

	Statement	Web	Category	Date	Label
6124	Jaishankar at Quad meet: India committed to re...	INDIANEXPRESS	TERROR	2020-10-01	TRUE
6748	Ramesh Pokhriyal Nishank to e-inaugurate infra...	THESTATESMAN	GOVERNMENT	2020-06-01	TRUE
7469	Over 3000 COVID-19 samples tested at CSIR-NEERI	THESTATESMAN	COVID-19	2020-06-01	TRUE
19056	Media attempting to influence outcome of cases...	TRIBUNEINDIA	TERROR	2020-10-01	TRUE
19059	Record buying of paddy on MSP, wheat to be sim...	TRIBUNEINDIA	GOVERNMENT	2020-10-01	TRUE

```
df_duplicates.shape
```

```
(1899, 5)
```

```
df.shape
```

```
(56703, 5)
```

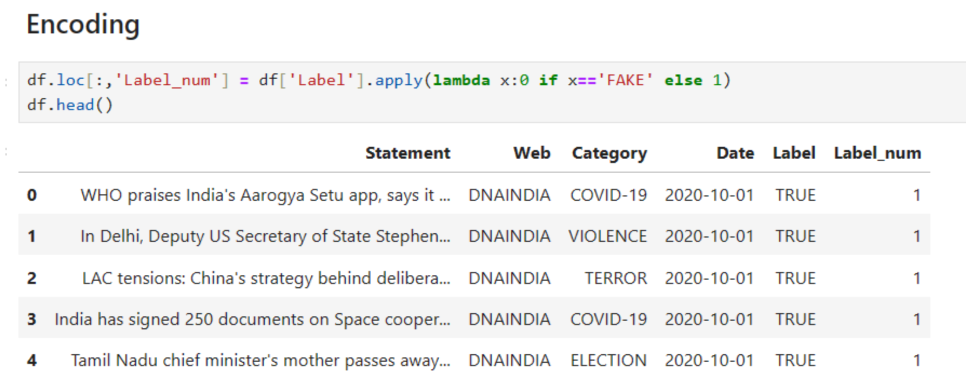
```
df.drop_duplicates(inplace=True)
df.shape
```

```
(54804, 5)
```

Handling duplicates involves identifying and managing rows within a dataset that are identical or nearly identical.

The code snippet provided illustrates the systematic handling of duplicate rows within a DataFrame. Initially, duplicates are identified using the `duplicated()` method, resulting in a Boolean Series denoting rows that are duplicates of preceding ones. Subsequently, a new DataFrame (`df_duplicates`) is created through Boolean indexing to isolate and display the duplicate rows for examination. Before any removal operation, the dimensions of the original DataFrame are recorded for reference. Duplicate rows are then removed from the DataFrame using the `drop_duplicates()` method, ensuring data integrity. Finally, the post-removal dimensions are compared with the initial dimensions to assess the effectiveness of the duplicate handling process. This approach facilitates the maintenance of data quality by systematically managing and mitigating the presence of duplicate entries, enhancing the reliability of subsequent analyses.

Encoding:


Encoding

```
df.loc[:,'Label_num'] = df['Label'].apply(lambda x:0 if x=='FAKE' else 1)
df.head()
```

	Statement	Web	Category	Date	Label	Label_num
0	WHO praises India's Aarogya Setu app, says it ...	DNAINDIA	COVID-19	2020-10-01	TRUE	1
1	In Delhi, Deputy US Secretary of State Stephen...	DNAINDIA	VIOLENCE	2020-10-01	TRUE	1
2	LAC tensions: China's strategy behind delibera...	DNAINDIA	TERROR	2020-10-01	TRUE	1
3	India has signed 250 documents on Space cooper...	DNAINDIA	COVID-19	2020-10-01	TRUE	1
4	Tamil Nadu chief minister's mother passes away...	DNAINDIA	ELECTION	2020-10-01	TRUE	1

Encoding categorical variables into numerical values is a common practice in machine learning and data analysis tasks. It facilitates the application of various algorithms that require numerical inputs. In this case, converting the 'Label' column from categorical ('FAKE' and 'TRUE') to numerical values (0 and 1) simplifies subsequent analysis and modeling processes. The encoded 'Label_num' column allows for easier manipulation, visualization, and application of machine learning algorithms, contributing to the efficiency and effectiveness of data analysis workflows.

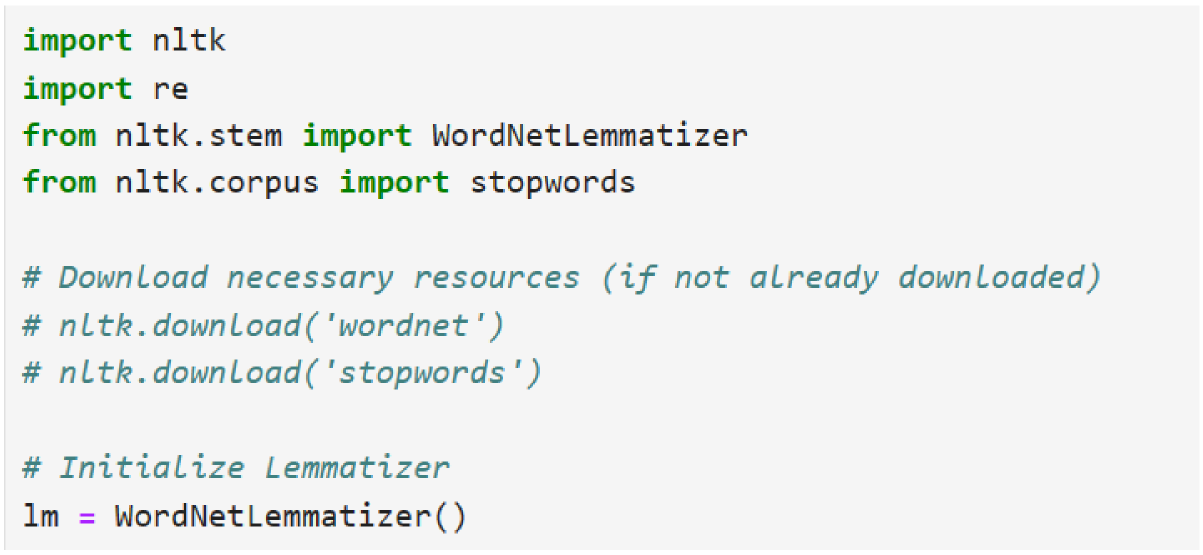
After encoding, the DataFrame index is reset using reset_index() with drop=True to reset the index without adding the old index as a new column. This ensures that the DataFrame is properly indexed for further analysis and downstream tasks, maintaining consistency in data handling procedures.

TEXT PREPROCESSING

Text preprocessing refers to the tasks performed on raw text data before it is used for natural language processing (NLP) tasks such as sentiment analysis, text classification, or language modeling. Preprocessing aims to clean and normalize the text data, making it suitable for analysis by removing noise, irrelevant information, and inconsistencies. Here are common text preprocessing techniques:

Text preprocessing is an essential step in natural language processing (NLP) pipelines, ensuring that text data is cleaned and standardized for further analysis. Here's a breakdown of various preprocessing techniques:

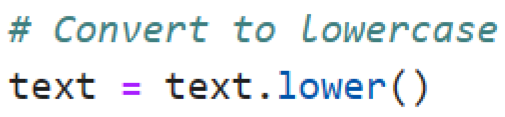
Libraries:



```python
import nltk
import re
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords

# Download necessary resources (if not already downloaded)
# nltk.download('wordnet')
# nltk.download('stopwords')

# Initialize Lemmatizer
lm = WordNetLemmatizer()
```

1. Lowercasing:

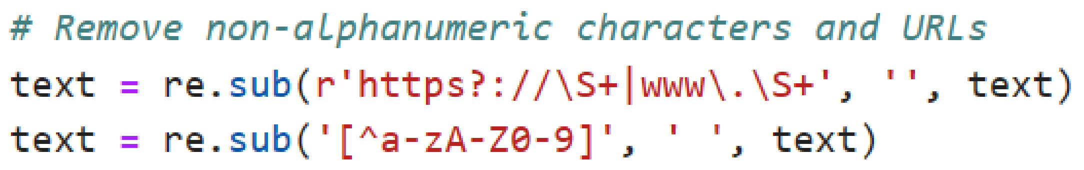
Helps ensure consistency in the text data by treating uppercase and lowercase versions of the same word as identical. For example, "Hello" and "hello" will be treated the same after lowercasing.



```python
# Convert to lowercase
text = text.lower()
```

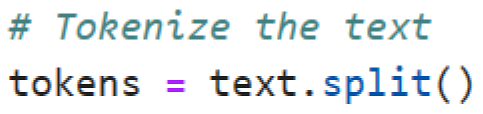
2. Remove non-alphanumeric characters and URLs:

Eliminate any characters that are neither letters nor numbers, as well as URLs. Removes noise from the text data, such as special characters and website links, which are often irrelevant for analysis and can interfere with model performance.



```python
# Remove non-alphanumeric characters and URLs
text = re.sub(r'https?://\S+|www\.\S+', '', text)
text = re.sub('[^a-zA-Z0-9]', ' ', text)
```

3. Tokenization:

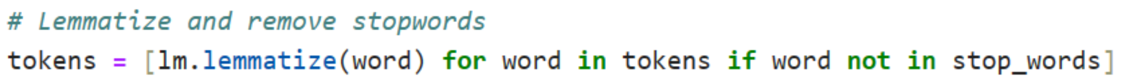


```python
# Tokenize the text
tokens = text.split()
```

Split the text into individual words or tokens. Breaks down the text into its basic units, making it easier to analyze and process. Each token typically represents a meaningful unit of text, such as a word or punctuation mark.

4. Lemmatization and Removing Stopwords:

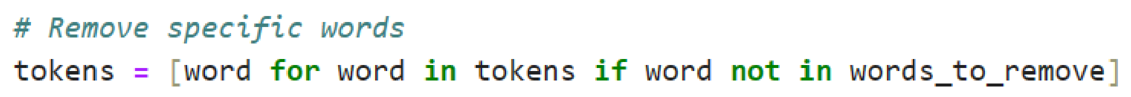
Reduce words to their base or dictionary forms (lemmatization) and remove common stopwords. Normalizes variations of words and eliminates frequently occurring but uninformative words. Lemmatization reduces words to their root forms, improving the consistency of the text data. Stopwords, such as "and" or "the", are removed as they do not carry much semantic meaning



```python
# Lemmatize and remove stopwords
tokens = [lm.lemmatize(word) for word in tokens if word not in stop_words]
```

5. Remove specific words:

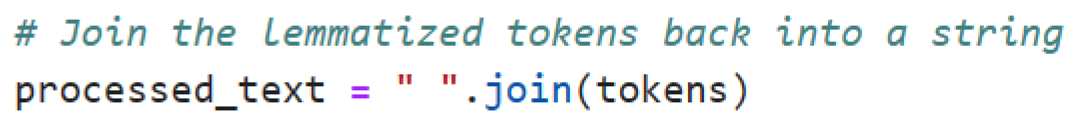
Eliminate predefined words from the text. Removes words that are not relevant or informative for the analysis. For example, removing domain-specific terms or common phrases like "fact check" from a dataset intended for sentiment analysis.



```python
# Remove specific words
tokens = [word for word in tokens if word not in words_to_remove]
```

6. Join the lemmatized tokens back into a string:

Combine the processed tokens back into a single text string. Reconstructs the text in a format suitable for further analysis or model input. By joining the tokens, the processed text can be easily used for tasks such as classification or generation.

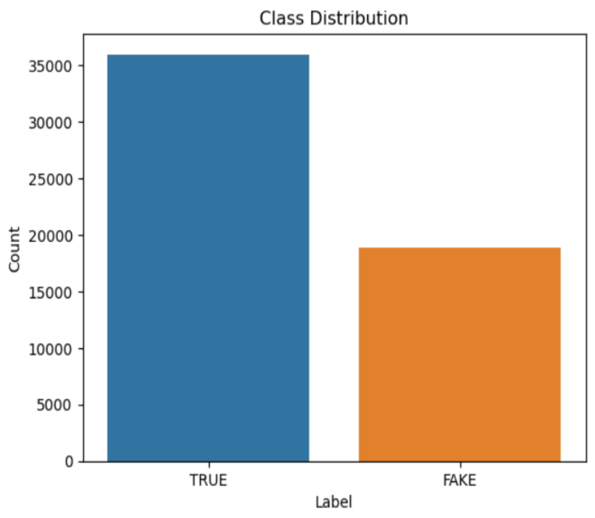


```python
# Join the lemmatized tokens back into a string
processed_text = " ".join(tokens)
```

These preprocessing techniques collectively help in preparing text data for NLP tasks, improving the quality and consistency of the input data and facilitating more accurate and effective analysis and modeling.

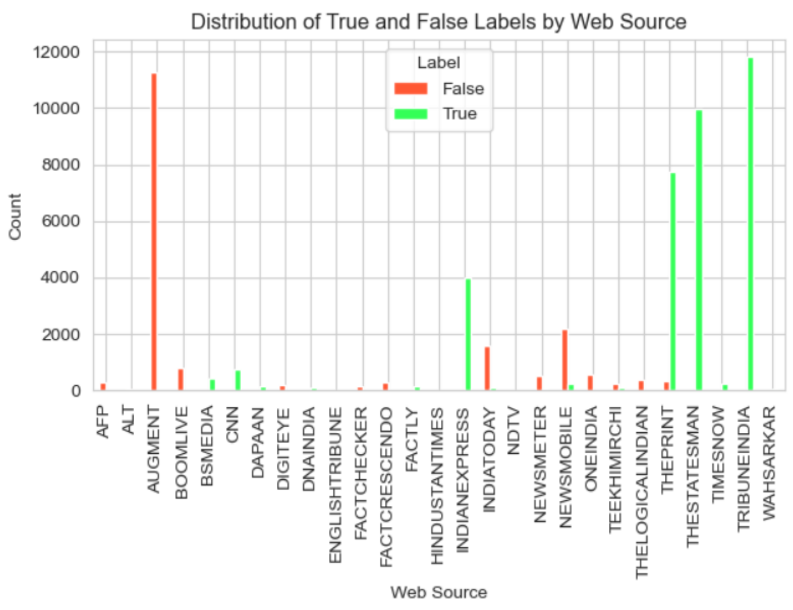
EXPLORATORY DATA ANALYSIS

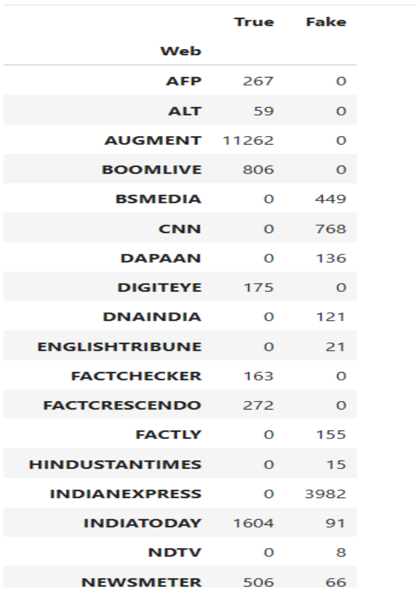
Class Distribution



This Bar Chart provides a quick and intuitive overview of the distribution of classes within the dataset, allowing users to understand the balance or imbalance between different categories, such as true and fake news. it appears that the dataset contains a higher number of instances label as "true news" compared to instances label as "fake news."

Distribution of True and False Labels by Web Source:





	True	Fake
Web		
AFP	267	0
ALT	59	0
AUGMENT	11262	0
BOOMLIVE	806	0
BSMEDIA	0	449
CNN	0	768
DAPAAN	0	136
DIGITEYE	175	0
DNAINDIA	0	121
ENGLISHTRIBUNE	0	21
FACTCHECKER	163	0
FACTCRESCENDO	272	0
FACTLY	0	155
HINDUSTANTIMES	0	15
INDIANEXPRESS	0	3982
INDIATODAY	1604	91
NDTV	0	8
NEWSMETER	506	66

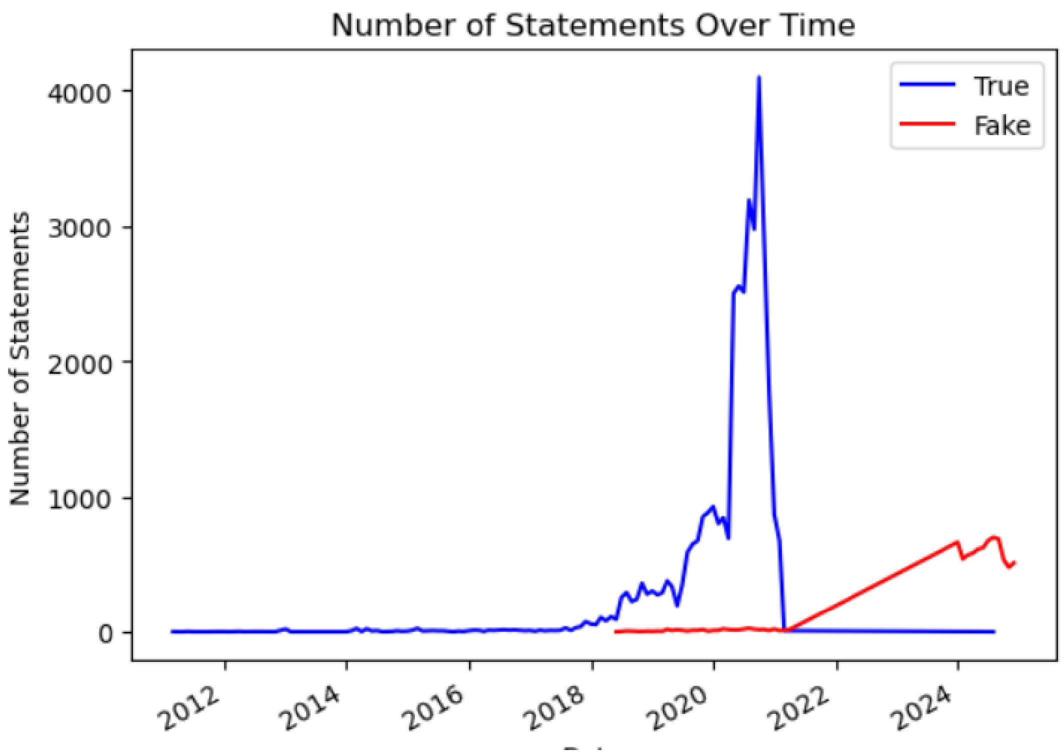
The chart provides insights into the labelling patterns for various web sources. Each bar represents a different source, and its height corresponds to the number of labels assigned to that source. Specifically, the chart distinguishes between true and false labels.

Some sources exhibit a higher count of false labels. This suggests that these sources may disseminate misinformation or unreliable content. Conversely, other sources have a greater number of true labels, implying their credibility and reliability.

The chart serves as a valuable tool for evaluating the trustworthiness of web sources. Sources with a preponderance of false labels warrant scrutiny, and users should exercise caution when relying on information from such platforms.

In summary, this chart aids in understanding the veracity of web content and highlights the need for critical evaluation when navigating online information.

Number of Statements Over Time:

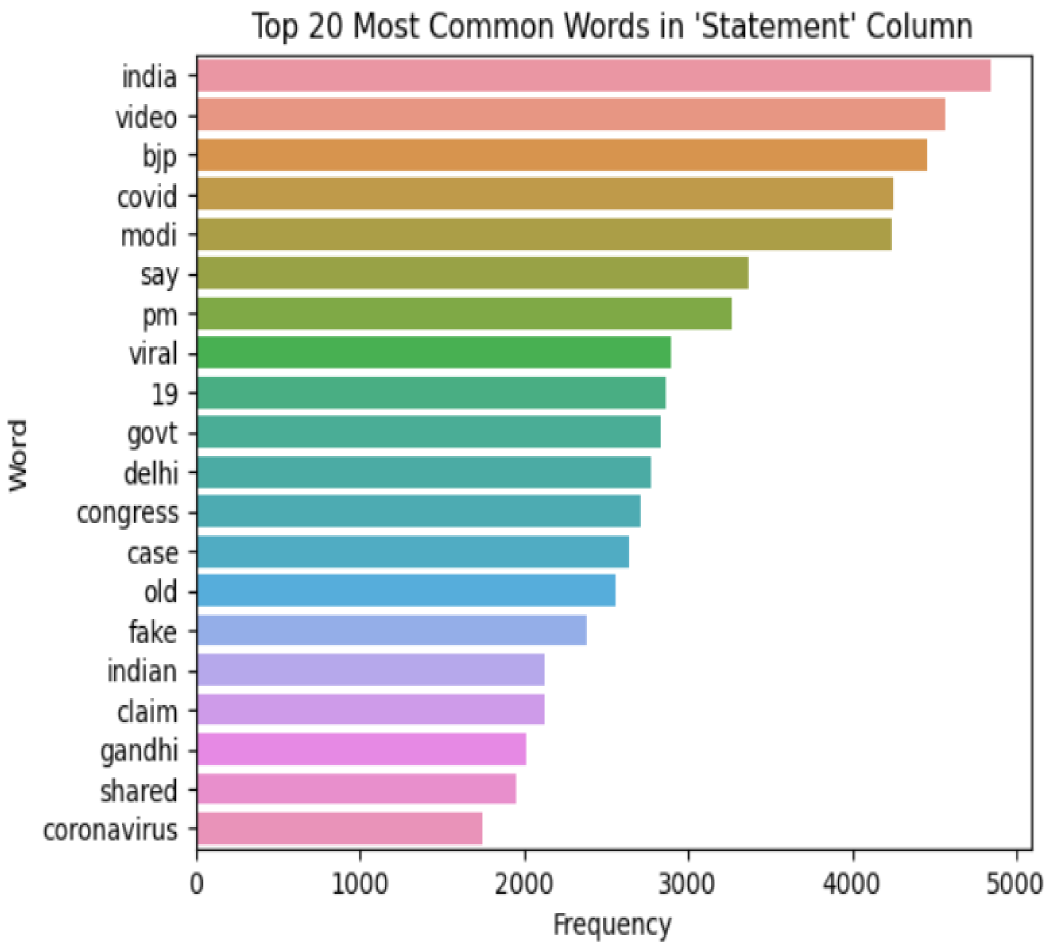


Initial Period (2012-2016): Both true and fake news statements were relatively low in number. There was a gradual increase in the count of statements during this time.

Spike in True News (Around 2020): Around **2020**, there was a significant spike in true news statements. The blue line reached its peak, indicating a surge in accurate information dissemination. This sudden increase suggests a notable event or change in reporting patterns.

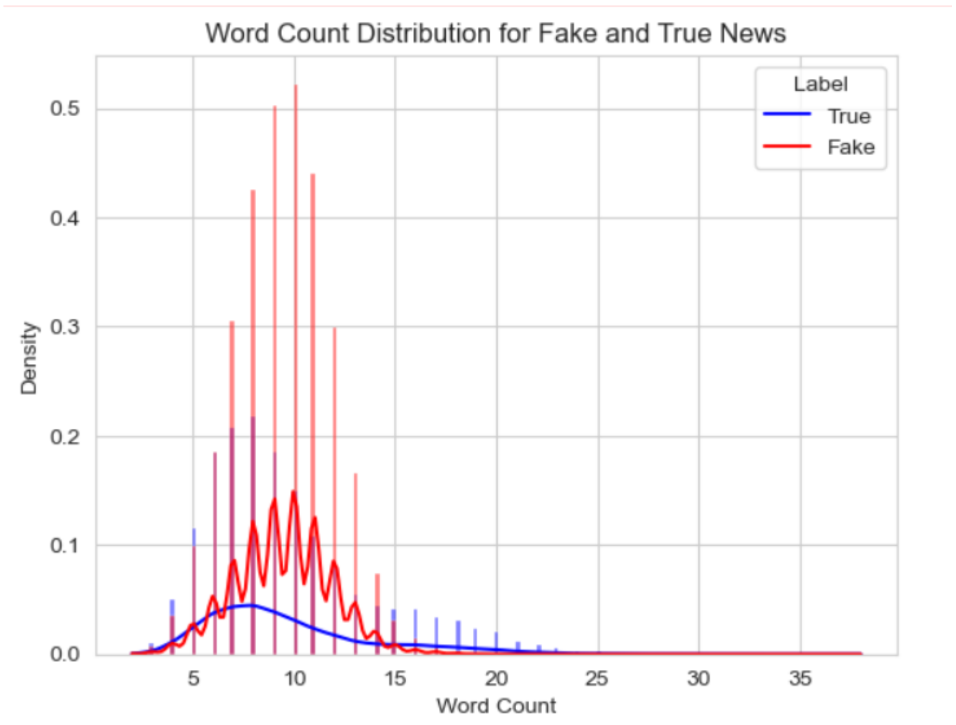
Stabilization and Gradual Increase in Fake News: After the peak, the true news count declined and stabilized. In contrast, the red line (representing fake news) saw a more gradual increase. Fake news remained consistent without experiencing a similar spike.

Top 20 Most Common Words in 'Statement' Column:



The most frequent words are those that appear at the top of the chart. Some of the common words include “india,” “video,” and “bjp.” . The prominence of certain words indicates their relevance within the analyzed texts. These words likely play a significant role in the dataset’s content. The chart provides insights into the prevalent themes or topics captured by the ‘Statement’ column. Further analysis could explore the context and implications of these frequently occurring words.

Word Count Distribution for Fake and True News:

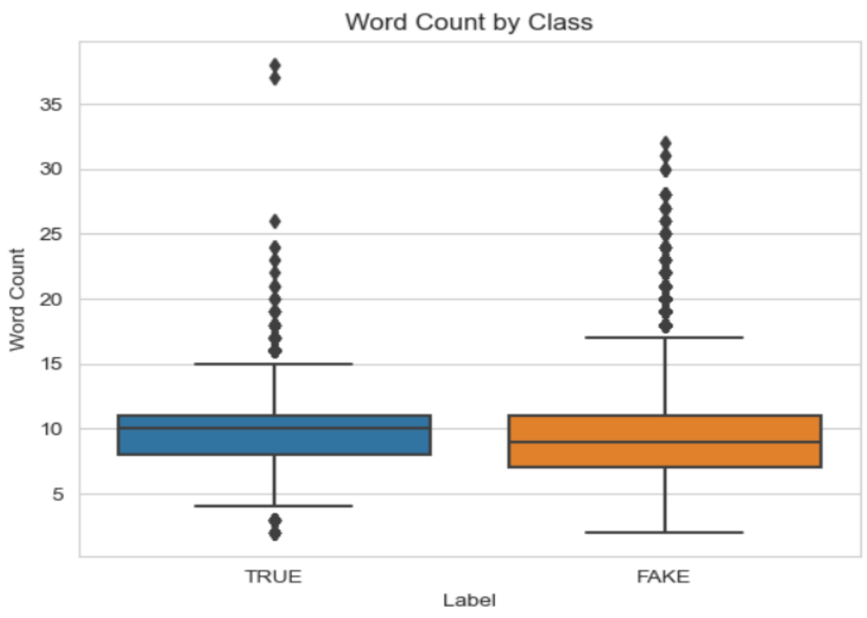


A new variable word count is formed in the data set by True News: The blue curve (representing true news) tends to have a higher word count. **True news articles are generally longer in terms of content**.

Fake News: The red curve (representing fake news) shows a peak at lower word counts, indicating that **fake news articles are often shorter.**

Longer true news articles may provide more in-depth information, context, and analysis. Shorter fake news articles might aim for brevity, sensationalism, or quick consumption. The difference in word count distribution could impact how readers perceive credibility. Longer articles may be perceived as more reliable due to thoroughness, while shorter ones might raise suspicion.

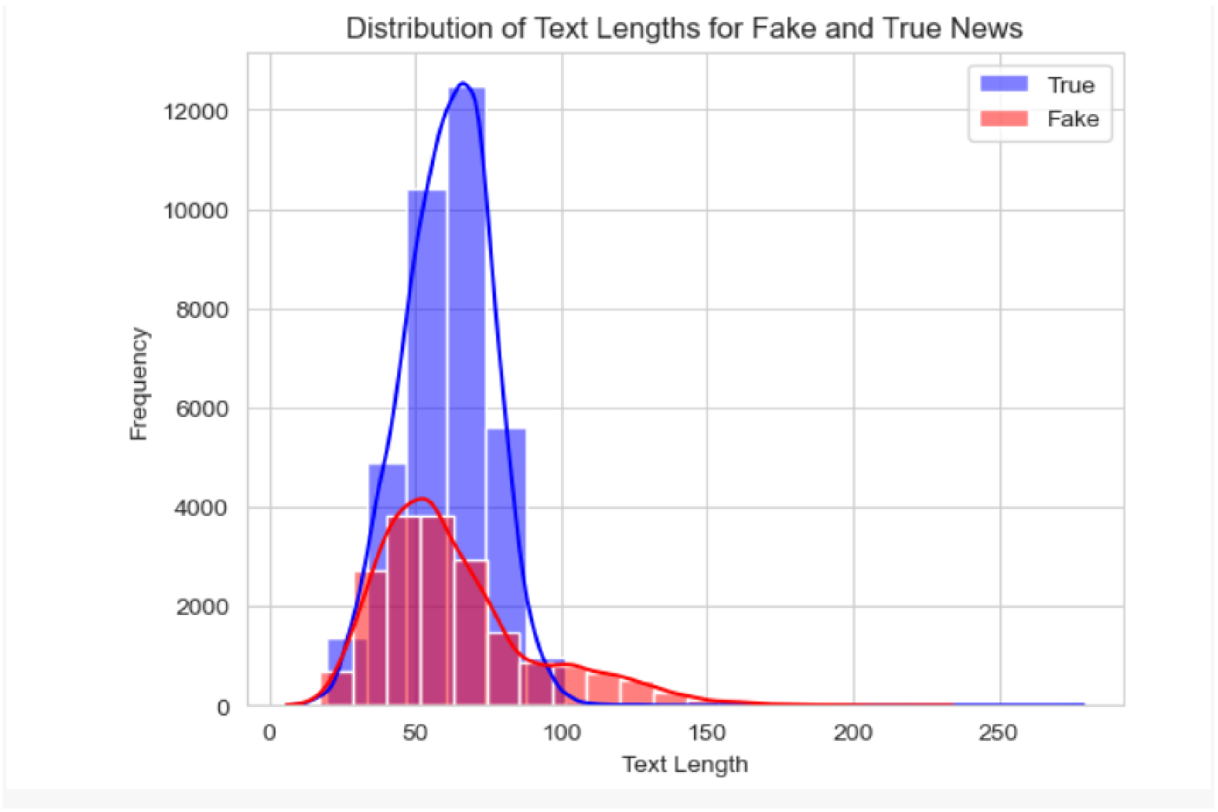
Word Count by class:



The boxplot in the image is a graphical representation of the distribution of word count for two classes labelled as ‘TRUE’ and ‘FAKE’. The line inside the box indicates the median word count for both ‘TRUE’ and ‘FAKE’ classes. It appears that the median word count is similar for both classes. It seems that the IQR for ‘FAKE’ class is slightly larger than the ‘TRUE’ class, indicating more variability in word count in the ‘FAKE’ class.

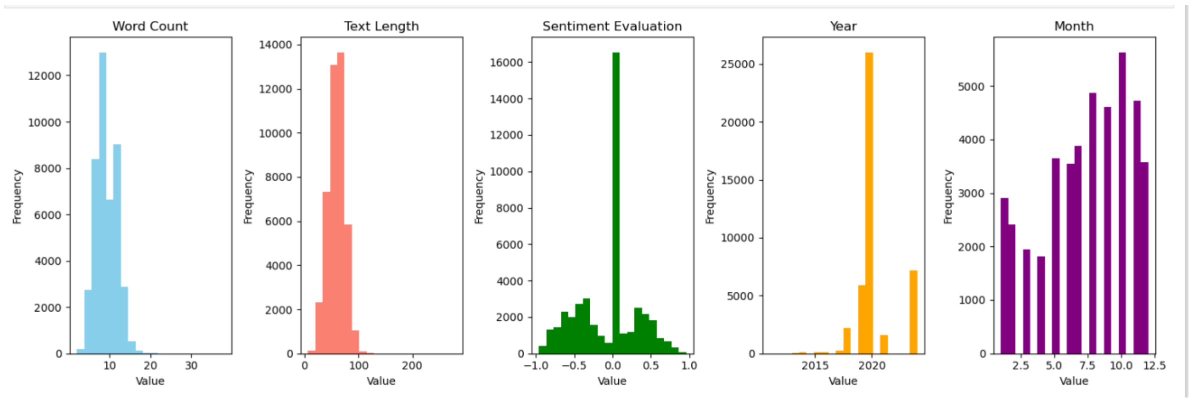
While the median word count is similar for both ‘TRUE’ and ‘FAKE’ classes, the distribution of word counts varies, with the ‘FAKE’ class showing more variability. This could suggest that the length of the text alone might not be a strong distinguishing feature between ‘TRUE’ and ‘FAKE’ classes. Other features might need to be considered for effective classification. Please note that this is a high-level interpretation and the actual insights might vary based on the specific dataset and context.

Distribution of Text Lengths for Fake and True News:

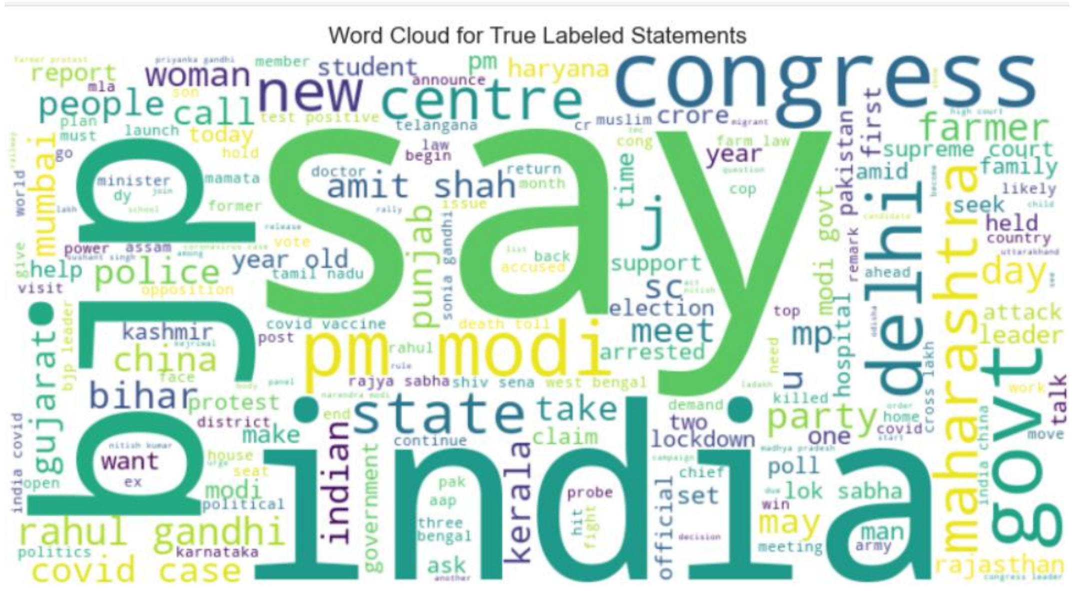


The histogram in the image is a graphical representation of the distribution of text lengths for 'True' and 'Fake' news. 'True' news articles tend to have a consistent text length around 50 words, while 'Fake' news articles show more variability in text length. This distinction could potentially be used as a feature in a machine learning model to classify news articles as 'True' or 'Fake'. However, given the overlap in the distributions, text length alone may not be sufficient for accurate classification, and other features might need to be considered. Please note that this is a high-level interpretation and the actual insights might vary based on the specific dataset and context.

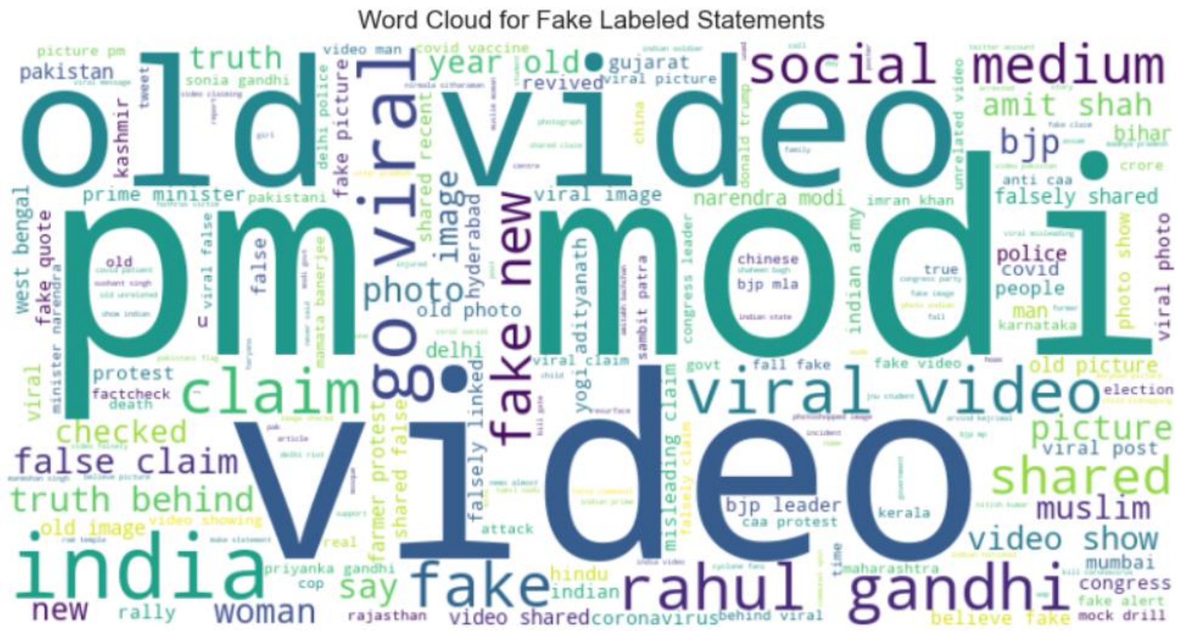
Distribution of all numerical Variables:



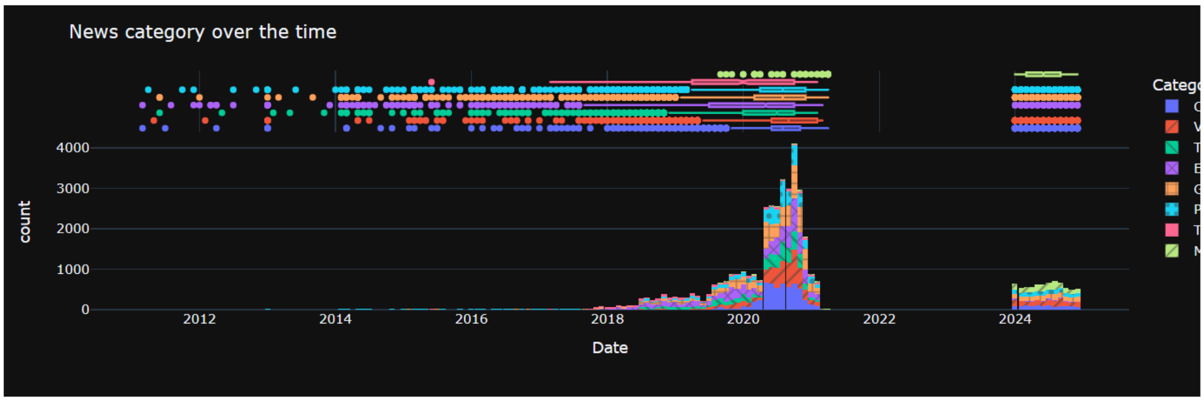
Word Cloud for True News:



Word Cloud for Fake News:



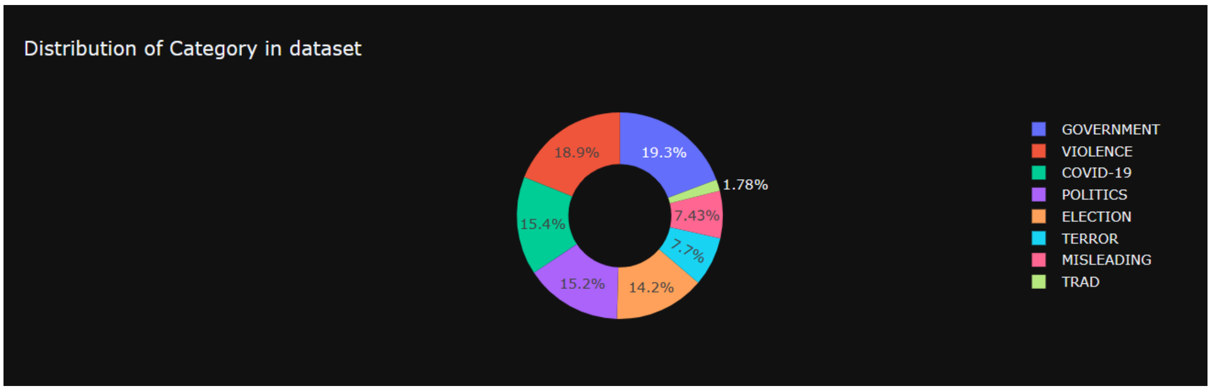
News category over the time:



The bar chart titled “News category over the time” illustrates the distribution of news articles across various categories from 2010 to 2024. Here are some insights based on the chart:

There is a **noticeable spike** in the volume of news articles around the year 2020, which could indicate increased media activity possibly due to significant global events. The distribution of news articles is **not uniform** over the years, suggesting that certain years had more newsworthy events or changes in news reporting practices. The use of different colors for categories allows for easy differentiation and analysis of trends within specific news categories over time.

Distribution Of Category:



The donut chart titled “Distribution of Category in dataset” provides a visual representation of the proportion of various categories within a dataset. Here’s an interpretation of the chart:

GOVERNMENT: This segment seems to represent a significant portion of the dataset, indicating a substantial number of articles or entries related to government affairs.

VIOLENCE: Another notable segment, suggesting a considerable amount of content associated with violence.

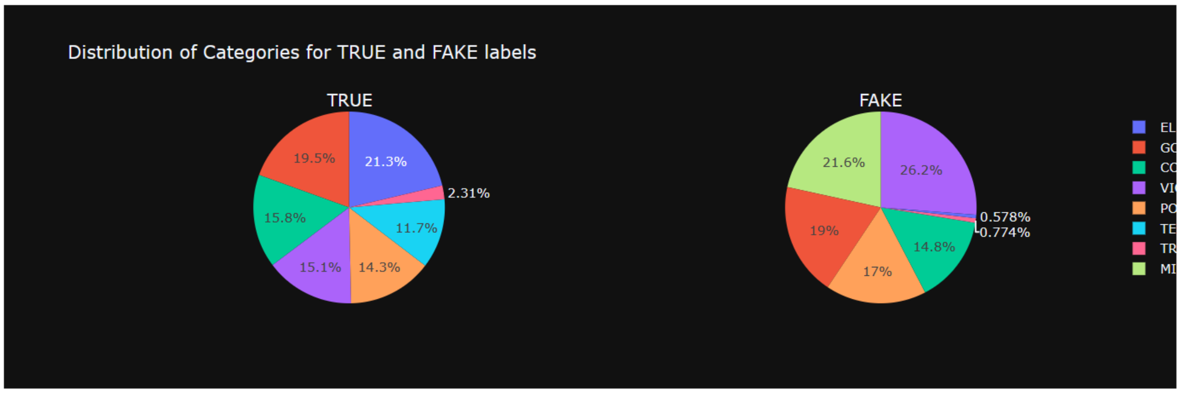
COVID-19: Given the context of recent years, this category likely represents the dataset's focus on the pandemic.

POLITICS, ELECTION, TERROR: These categories are also well-represented, highlighting the dataset's emphasis on political and security-related news.

MISLEADING: This smaller segment could indicate the presence of articles tagged as potentially misleading information.

TRAD: Presumably short for 'traditional', it might refer to conventional news topics or categories that don't fall into the other specified segments.

Distribution of Categories for TRUE and FAKE labels:



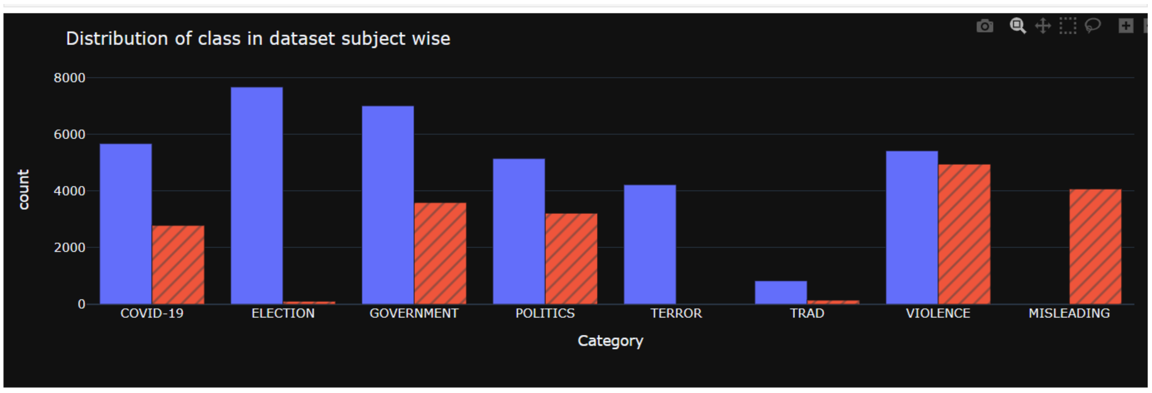
<u>The TRUE pie chart:</u> Contains several segments, each representing a category. The largest segment (approximately 40% of the chart) corresponds to the GOVERNMENT category. Other segments include VIOLENCE, COVID-19, POLITICS, ELECTION, TERROR, MISLEADING, and TRAD.

<u>The FAKE pie chart</u>: Also contains several segments, representing the same categories. The largest segment (around 60% of the chart) corresponds to the GOVERNMENT category. Other segments include VIOLENCE, COVID-19, POLITICS, ELECTION, TERROR, MISLEADING, and TRAD.

<u>Observations:</u>

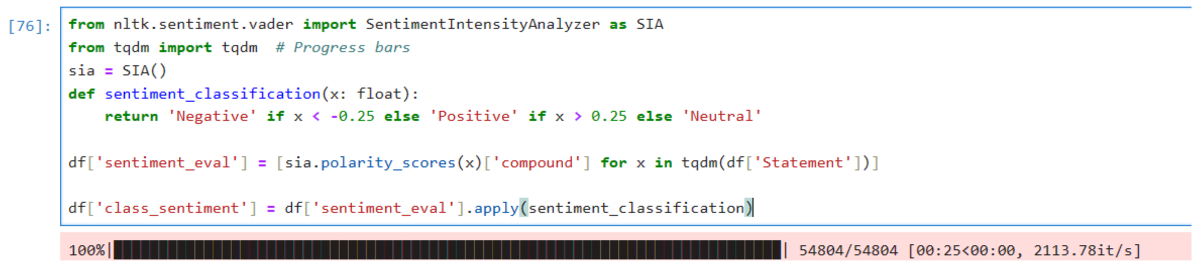
Both pie charts share similar category proportions. GOVERNMENT is the dominant category in both TRUE and FAKE labels. VIOLENCE, COVID-19, and POLITICS are also significant categories. The presence of MISLEADING suggests that some articles may contain false or misleading information. The prevalence of certain categories may impact how news is perceived.

Distribution of class in dataset subject wise:

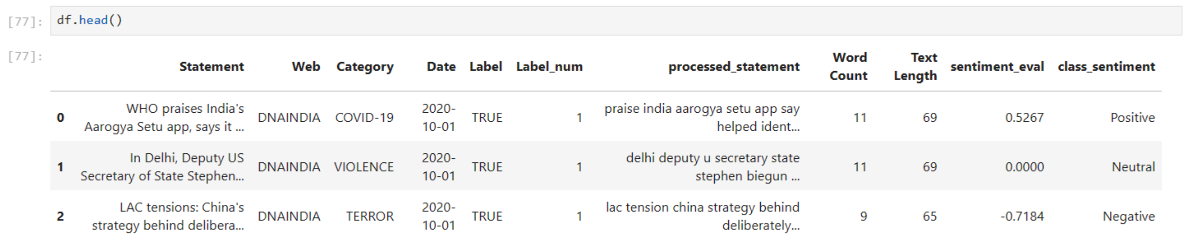


The bar graph you've provided appears to show the distribution of classes in a dataset, categorized by subject. It highlights that subjects like 'Election' and 'Government' have higher counts, suggesting they are more prevalent within the dataset. Conversely, the subject 'Trad' has the lowest count, indicating it is less common.

SENTIMENT ANALYSIS



```
[76]: from nltk.sentiment.vader import SentimentIntensityAnalyzer as SIA
      from tqdm import tqdm  # Progress bars
      sia = SIA()
      def sentiment_classification(x: float):
          return 'Negative' if x < -0.25 else 'Positive' if x > 0.25 else 'Neutral'

      df['sentiment_eval'] = [sia.polarity_scores(x)['compound'] for x in tqdm(df['Statement'])]

      df['class_sentiment'] = df['sentiment_eval'].apply(sentiment_classification)

100%|████████████████████████████████████████████████████████████| 54804/54804 [00:25<00:00, 2113.78it/s]
```

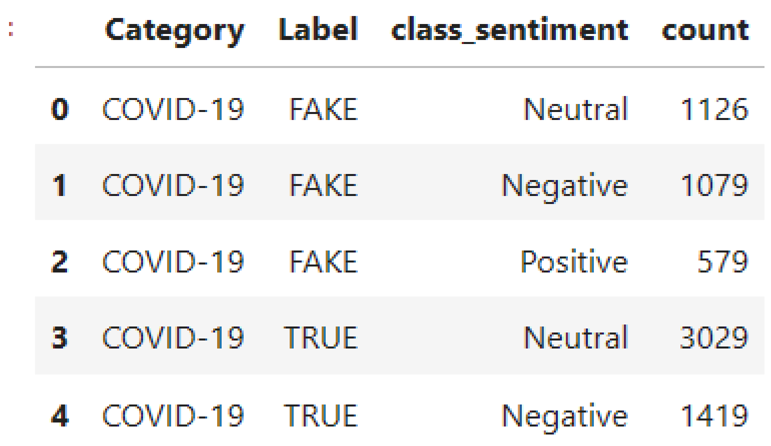


```
[77]: df.head()
```

[77]:

	Statement	Web	Category	Date	Label	Label_num	processed_statement	Word Count	Text Length	sentiment_eval	class_sentiment
0	WHO praises India's Aarogya Setu app, says it ...	DNAINDIA	COVID-19	2020-10-01	TRUE	1	praise india aarogya setu app say helped ident...	11	69	0.5267	Positive
1	In Delhi, Deputy US Secretary of State Stephen...	DNAINDIA	VIOLENCE	2020-10-01	TRUE	1	delhi deputy u secretary state stephen biegun ...	11	69	0.0000	Neutral
2	LAC tensions: China's strategy behind delibera...	DNAINDIA	TERROR	2020-10-01	TRUE	1	lac tension china strategy behind deliberately...	9	65	-0.7184	Negative

The provided code snippet utilizes the NLTK library, particularly its VADER module, for sentiment analysis on textual data. First, it downloads the necessary lexicon for sentiment analysis. Then, it initializes a SentimentIntensityAnalyzer object. A function is defined to classify sentiment polarity scores into 'Negative', 'Positive', or 'Neutral' categories. The sentiment analysis is applied to each statement in the DataFrame, calculating sentiment polarity scores using VADER's compound metric. These scores are stored in a new column. Finally, the sentiment classification function is applied to categorize the polarity scores, and the results are stored in another column. This process allows for efficient sentiment analysis and classification of textual data within the DataFrame.

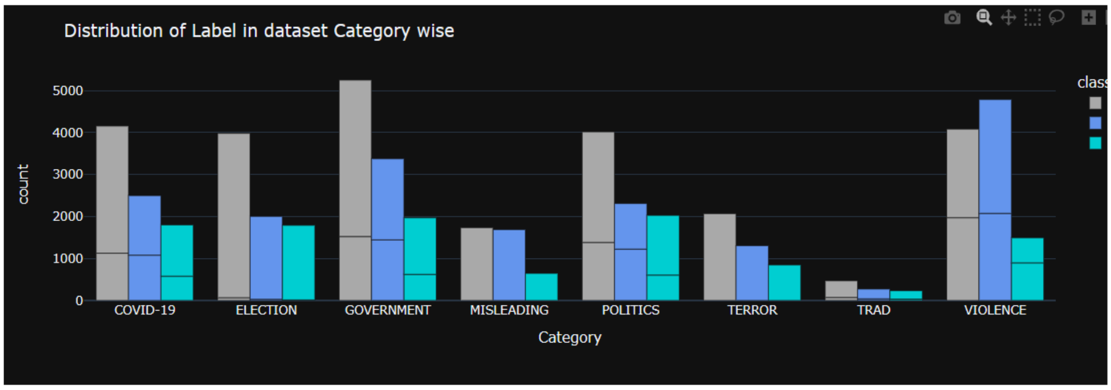


:

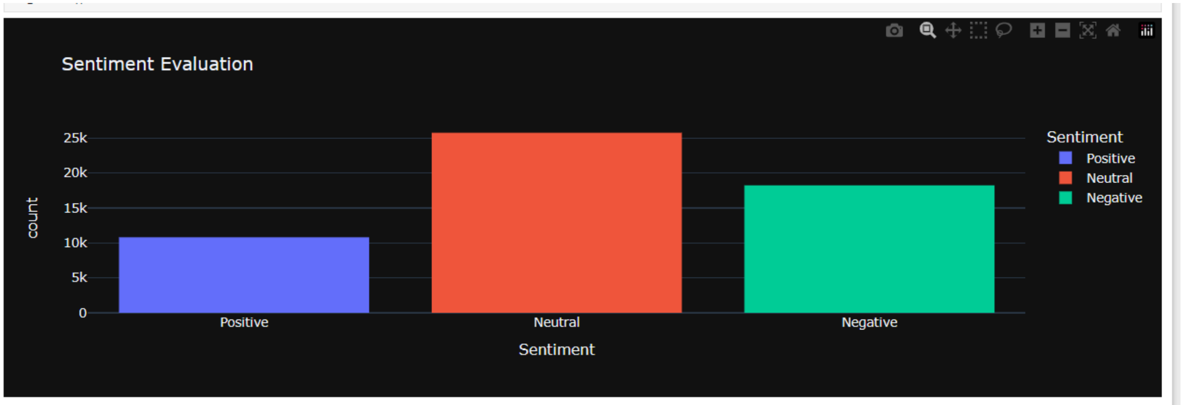
	Category	Label	class_sentiment	count
0	COVID-19	FAKE	Neutral	1126
1	COVID-19	FAKE	Negative	1079
2	COVID-19	FAKE	Positive	579
3	COVID-19	TRUE	Neutral	3029
4	COVID-19	TRUE	Negative	1419

The sentiment distribution varies across different categories and labels. For instance, within the 'COVID-19' category, there are notable differences in sentiment between 'FAKE' and

'TRUE' labels. This suggests that the sentiment expressed in texts pertaining to COVID-19 differs based on whether the information is classified as fake or true. Across various categories and labels, neutral sentiment appears to be prevalent. This indicates that a significant portion of the analyzed text exhibits neutrality in sentiment expression. Comparing sentiment distributions between 'FAKE' and 'TRUE' labels reveals interesting patterns. In some cases, such as within the 'COVID-19' category, the sentiment distribution differs significantly between fake and true labels. This suggests that the classification of information as fake or true may influence the sentiment expressed in the corresponding texts.



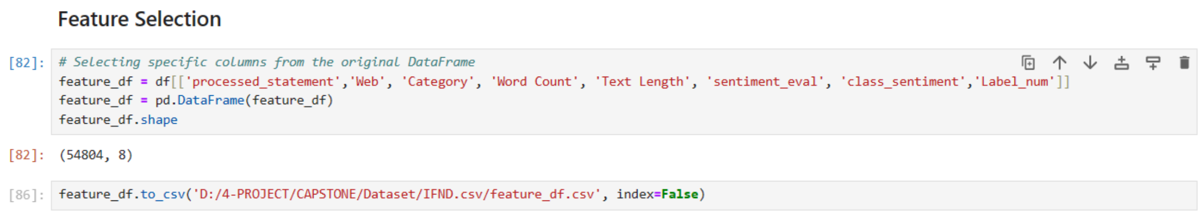
It shows how different categories such as COVID-19, ELECTION, and GOVERNMENT are represented in the dataset. This can help identify which topics are more prevalent or require more data collection. The two shades of color in each bar may indicate a binary classification within each category, such as positive/negative or present/absent. This is useful for understanding the balance of classes within each category.



The length of each bar reflects the prevalence of the sentiment in the dataset. The most notable insight is that neutral sentiments are the most common, followed by positive and negative sentiments.

MODEL BUILDING AND EVALUATION

Feature Selection :

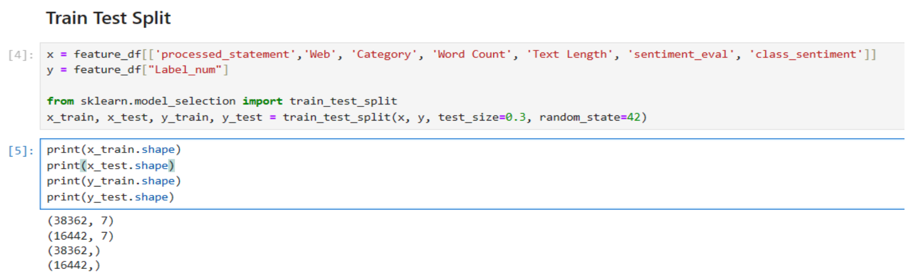


Feature Selection

```
[82]: # Selecting specific columns from the original DataFrame
      feature_df = df[['processed_statement','Web', 'Category', 'Word Count', 'Text Length', 'sentiment_eval', 'class_sentiment','Label_num']]
      feature_df = pd.DataFrame(feature_df)
      feature_df.shape
[82]: (54804, 8)
[86]: feature_df.to_csv('D:/4-PROJECT/CAPSTONE/Dataset/IFND.csv/feature_df.csv', index=False)
```

The selected columns for the feature DataFrame (`feature_df`) encompass a comprehensive set of variables that have demonstrated significance in classifying news articles based on their content and sentiment. The 'processed_statement' column contains the preprocessed textual content of the news articles, ensuring that only relevant information is considered for analysis. The inclusion of 'Web' provides insight into the source or platform from which the news articles originated, allowing for potential differentiation based on credibility or bias of sources. Additionally, 'Category' categorizes news articles into broader topics or themes, aiding in identifying patterns and trends within specific domains. 'Word Count' and 'Text Length' offer quantitative measures of the complexity and depth of the articles, which can influence their classification. The 'sentiment_eval' column provides sentiment polarity scores, enabling the consideration of emotional tone and subjective expression in the articles. 'class_sentiment' further refines sentiment analysis by categorizing articles into 'Negative', 'Positive', or 'Neutral' sentiments based on predefined thresholds. Lastly, 'Label_num' represents a numerical encoding of the article's label, facilitating classification tasks in machine learning algorithms. Collectively, these selected variables offer a rich and diverse feature set for effectively classifying news articles, encompassing textual content, source characteristics, sentiment analysis, and categorical labelling.

Train Test Split:



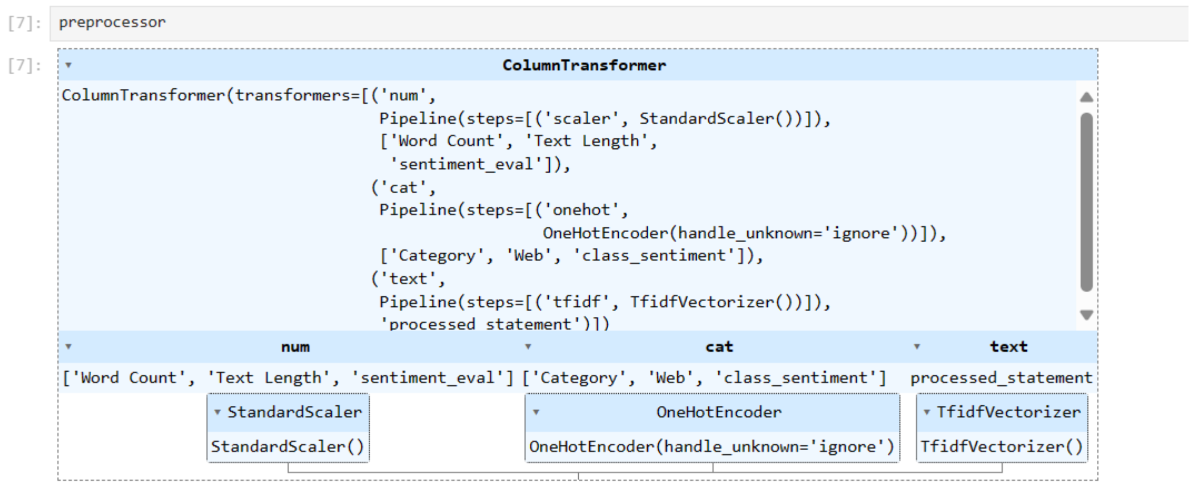
Train Test Split

```
[4]: x = feature_df[['processed_statement','Web', 'Category', 'Word Count', 'Text Length', 'sentiment_eval', 'class_sentiment']]
     y = feature_df["Label_num"]

     from sklearn.model_selection import train_test_split
     x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=42)
[5]: print(x_train.shape)
     print(x_test.shape)
     print(y_train.shape)
     print(y_test.shape)
     (38362, 7)
     (16442, 7)
     (38362,)
     (16442,)
```

This function splits the dataset into training and testing sets. It randomly divides the data into subsets, where a portion (30% in this case) is allocated for testing. `random_state=42` - This parameter ensures reproducibility by fixing the random seed. The same seed ensures that the split is deterministic, meaning the same split occurs every time the code is run. The output indicates the dimensions of the training and testing sets: 38,362 instances with 7 features for training, and 16,442 instances with 7 features for testing, along with corresponding target variable shapes.

Model Building:

This code segment sets up preprocessing pipelines for different types of features (numerical, categorical, and textual) to prepare the data for machine learning models.



```
[7]: preprocessor
```

```
[7]: ColumnTransformer
ColumnTransformer(transformers=[('num',
                                 Pipeline(steps=[('scaler', StandardScaler())]),
                                 ['Word Count', 'Text Length',
                                  'sentiment_eval']),
                                ('cat',
                                 Pipeline(steps=[('onehot',
                                                  OneHotEncoder(handle_unknown='ignore'))]),
                                 ['Category', 'Web', 'class_sentiment']),
                                ('text',
                                 Pipeline(steps=[('tfidf', TfidfVectorizer())]),
                                 'processed_statement')])

num                                            cat                                  text
['Word Count', 'Text Length', 'sentiment_eval'] ['Category', 'Web', 'class_sentiment'] processed_statement
StandardScaler                                 OneHotEncoder                        TfidfVectorizer
StandardScaler()                               OneHotEncoder(handle_unknown='ignore') TfidfVectorizer()
```

1. Numerical Features Preprocessing:

numerical_features: List of numerical feature names. numerical_transformer: Pipeline that applies standard scaling to numerical features to make sure they have mean 0 and standard deviation 1, which is important for many machine learning algorithms.

2. Categorical Features Preprocessing:

categorical_features: List of categorical feature names. categorical_transformer: Pipeline that applies one-hot encoding to categorical features. This converts categorical variables into a binary format, which is suitable for algorithms that can't directly deal with categorical data.

3. Textual Features Preprocessing:

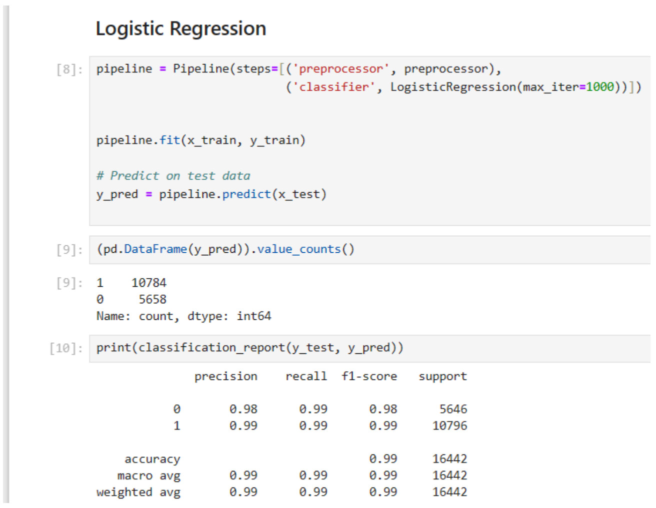
textual_features: List of textual feature names. textual_transformer: Pipeline that applies TF-IDF (Term Frequency-Inverse Document Frequency) vectorization to convert textual data into numerical vectors. TF-IDF reflects the importance of a word in a document relative to a collection of documents. This is crucial for handling textual data in machine learning.

4. ColumnTransformer:

preprocessor: ColumnTransformer that applies different preprocessing steps to different columns of the dataset. transformers: List of tuples where each tuple contains a name, a transformer, and a list of columns to apply the transformation to. num: Applies numerical_transformer to numerical features specified in `numerical_features`. cat: Applies categorical_transformer to categorical features specified in categorical_features. text : Applies textual_transformer to textual features specified in textual_features.

Logistic Regression:

Logistic regression entails modeling the likelihood of a discrete outcome based on input variables. Typically, logistic regression is employed for binary outcomes, such as true/false or yes/no scenarios, although multinomial logistic regression extends this to cases with multiple discrete outcomes. This method proves valuable in classification tasks, where the objective is to categorize new samples effectively. Given that various aspects of cybersecurity involve classification challenges, such as detecting attacks, logistic regression emerges as a valuable analytical tool in this domain.


Logistic Regression

```
[8]: pipeline = Pipeline(steps=[('preprocessor', preprocessor),
                               ('classifier', LogisticRegression(max_iter=1000))])


     pipeline.fit(x_train, y_train)

     # Predict on test data
     y_pred = pipeline.predict(x_test)
```

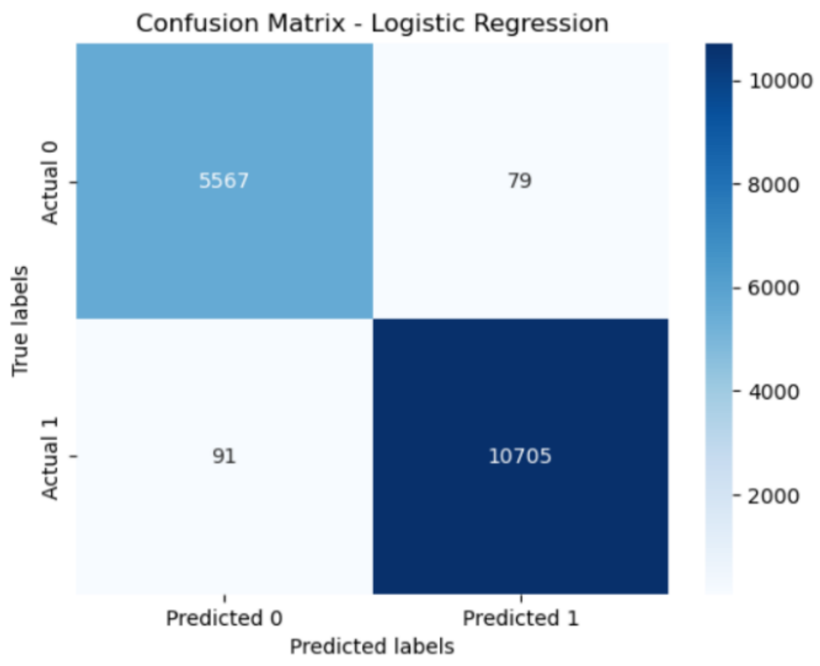
```
[9]: (pd.DataFrame(y_pred)).value_counts()
```

```
[9]: 1    10784
     0     5658
     Name: count, dtype: int64
```

```
[10]: print(classification_report(y_test, y_pred))
```

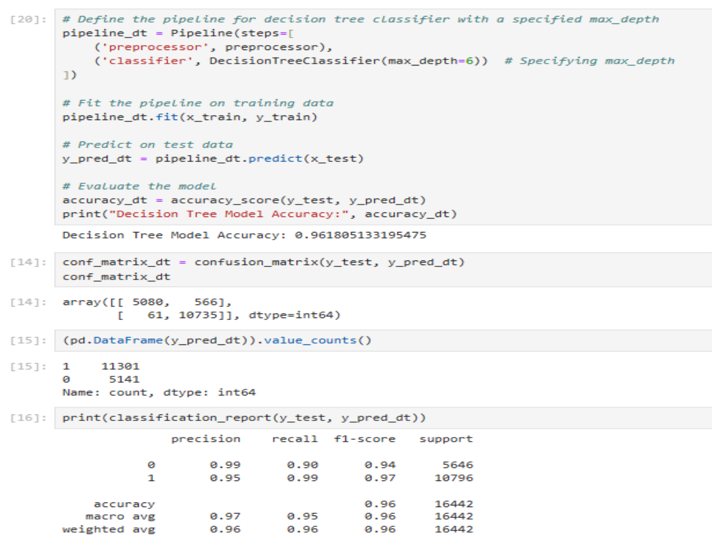
```
              precision    recall  f1-score   support

           0       0.98      0.99      0.98      5646
           1       0.99      0.99      0.99     10796

    accuracy                           0.99     16442
   macro avg       0.99      0.99      0.99     16442
weighted avg       0.99      0.99      0.99     16442
```

The classification report presents a detailed assessment of the model's performance on the test data. The precision scores indicate the proportion of correct predictions within each class: for class 0, it stands at 98%, meaning 98% of instances classified as class 0 were indeed class 0, while for class 1, it reaches 99%, demonstrating a similarly high accuracy in identifying class 1 instances. The recall values reveal the model's ability to capture all relevant instances of each class: with 99% for both classes, it suggests that the model effectively identifies the majority of actual instances of both classes. F1-scores, which provide a balance between precision and recall, are high for both classes, affirming the model's robustness across the test set. The support values specify the number of instances for each class in the test set, with class 1 instances outnumbering class 0 instances. The accuracy score of 99% highlights the model's overall correctness in classifying instances from the test set. Both macro and weighted averages of precision, recall, and F1-score indicate consistently high performance across all metrics and classes. In conclusion, the classification report underscores the model's exceptional efficacy in accurately predicting the target labels for the test data, across a variety of evaluation metrics.



Decision Tree Model:

A decision tree represents a tree-like structure where internal nodes correspond to features (or attributes), branches denote decision rules, and leaf nodes signify outcomes. The root node, located at the top of the tree, initiates partitioning based on attribute values through recursive partitioning. This flowchart-like depiction aids decision-making and mirrors human thought processes, making decision trees easily interpretable. Unlike black box algorithms such as neural networks, decision trees are transparent, sharing their internal decision logic. They offer faster training times compared to neural networks. The time complexity of decision trees depends on the number of records and attributes in the dataset, making them suitable for high-dimensional data without requiring probability distribution assumptions.



```
[20]: # Define the pipeline for decision tree classifier with a specified max_depth
      pipeline_dt = Pipeline(steps=[
          ('preprocessor', preprocessor),
          ('classifier', DecisionTreeClassifier(max_depth=6))  # Specifying max_depth
      ])

      # Fit the pipeline on training data
      pipeline_dt.fit(x_train, y_train)

      # Predict on test data
      y_pred_dt = pipeline_dt.predict(x_test)

      # Evaluate the model
      accuracy_dt = accuracy_score(y_test, y_pred_dt)
      print("Decision Tree Model Accuracy:", accuracy_dt)

      Decision Tree Model Accuracy: 0.961805133195475

[14]: conf_matrix_dt = confusion_matrix(y_test, y_pred_dt)
      conf_matrix_dt

[14]: array([[ 5080,   566],
             [   61, 10735]], dtype=int64)

[15]: (pd.DataFrame(y_pred_dt)).value_counts()

[15]: 1    11301
      0     5141
      Name: count, dtype: int64

[16]: print(classification_report(y_test, y_pred_dt))

                    precision    recall  f1-score   support

                 0       0.99      0.90      0.94      5646
                 1       0.95      0.99      0.97     10796

          accuracy                           0.96     16442
         macro avg       0.97      0.95      0.96     16442
      weighted avg       0.96      0.96      0.96     16442
```

1. Impact of Cost Complexity Pruning (ccp_alpha) :

The decision tree models are trained with different values of ccp_alpha , ranging from 0.0 to 0.19. As ccp_alpha increases, the decision tree's accuracy gradually decreases, indicating that pruning leads to a reduction in overfitting. With no pruning (ccp_alpha=0.0), the model achieves high accuracy (99%), but as pruning increases, accuracy decreases, reaching 86.13% when ccp_alpha is 0.15 0.19.

2. Equivalent Depth with Pruning :

The analysis reveals that with no cost-complexity pruning (ccp_alpha=0.0), the decision tree reaches a depth of 76. Such excessive depth may lead to overfitting and inefficiency in generalization to unseen data. Conversely, setting ccp_alpha to 0.01 results in a more reasonable depth of 6, indicating a more balanced trade-off between model complexity and performance. Therefore, considering the practicality and effectiveness of a depth of 6, ccp_alpha=0.01 is chosen as the preferred option for pruning. The equivalent depth value for ccp_alpha=0.01 is 6, indicating that the pruned decision tree's depth is effectively reduced compared to the unpruned tree.

3. Comparison with Unpruned Decision Tree :

An unpruned decision tree (with ccp_alpha=0.01) achieves an accuracy of 96.18%, slightly lower than the accuracy of the unpruned tree. The confusion matrix reveals that the unpruned tree has 5,080 true negatives, 10,735 true positives, 566 false negatives, and 61 false positives. The classification report shows high precision, recall, and F1 score for both classes, indicating good performance of the unpruned decision tree model.

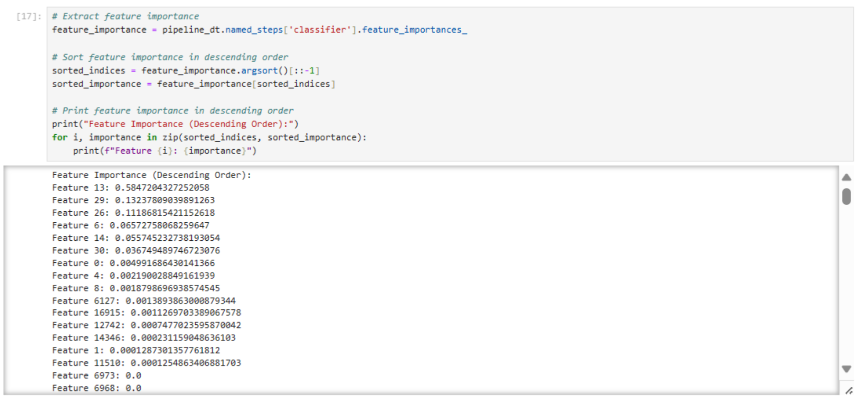
4. Model Prediction Distribution :

The distribution of predicted labels shows that the unpruned decision tree predicts more instances as class 1 (attack detected) compared to class 0 (no attack detected).

5. Overall Model Evaluation :

The classification report provides a comprehensive evaluation of the unpruned decision tree's performance, indicating high precision, recall

Feature Importance:



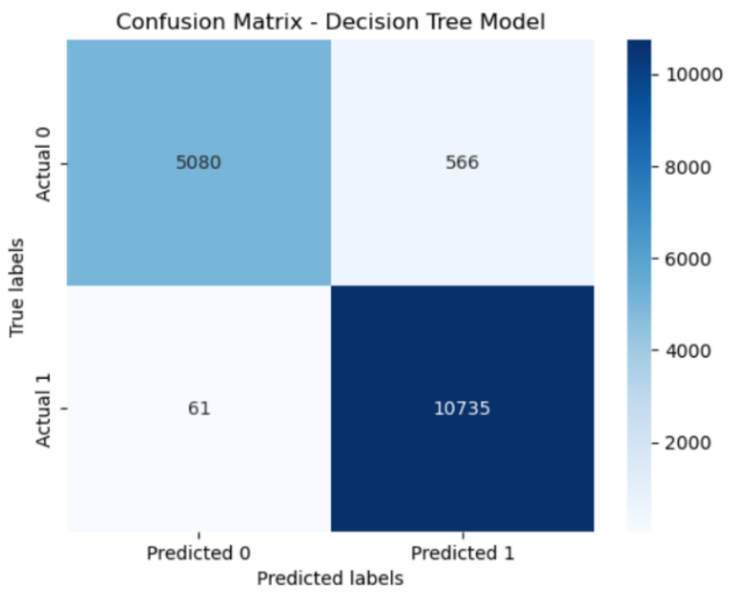
```
[17]: # Extract feature importance
      feature_importance = pipeline_dt.named_steps['classifier'].feature_importances_

      # Sort feature importance in descending order
      sorted_indices = feature_importance.argsort()[::-1]
      sorted_importance = feature_importance[sorted_indices]

      # Print feature importance in descending order
      print("Feature Importance (Descending Order):")
      for i, importance in zip(sorted_indices, sorted_importance):
          print(f"Feature {i}: {importance}")
```

```
Feature Importance (Descending Order):
Feature 13: 0.5847204327252058
Feature 29: 0.13237809039891263
Feature 26: 0.11186815421152618
Feature 6: 0.06572758068259647
Feature 14: 0.055745232738193054
Feature 30: 0.036749489746723076
Feature 0: 0.004991686430141366
Feature 4: 0.002190028849161939
Feature 8: 0.0018798696938574545
Feature 6127: 0.0013893863000879344
Feature 16915: 0.0011269703389067578
Feature 12742: 0.0007477023595870042
Feature 14346: 0.000231159048636103
Feature 1: 0.0001287301357761812
Feature 11510: 0.0001254863406881703
Feature 6973: 0.0
Feature 6968: 0.0
```

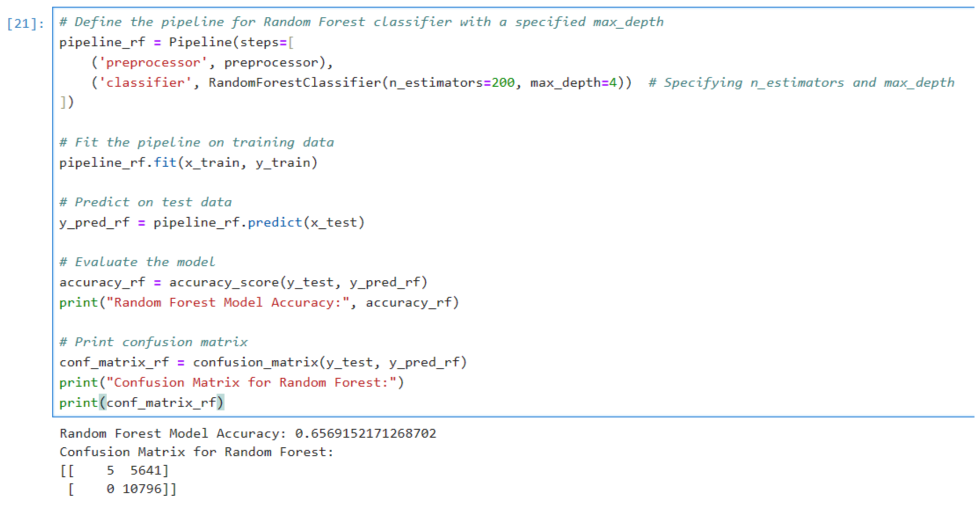
The feature importance values indicate the contribution of each feature to the model's predictions. the features are listed in descending order of importance along with their respective importance scores.

- The most important feature is Feature 13 with an importance score of approximately 0.5847.

- The second most important feature is Feature 29 with an importance score of approximately 0.1324.
- Feature 26 follows with an importance score of approximately 0.1119.
- Then comes Feature 6 with an importance score of approximately 0.0657 and so on.



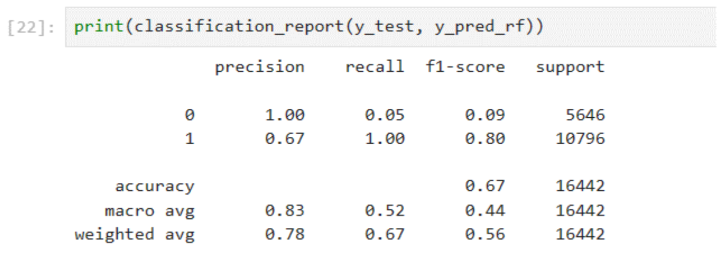
Random Forest:



```
[21]: # Define the pipeline for Random Forest classifier with a specified max_depth
pipeline_rf = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', RandomForestClassifier(n_estimators=200, max_depth=4))  # Specifying n_estimators and max_depth
])

# Fit the pipeline on training data
pipeline_rf.fit(x_train, y_train)

# Predict on test data
y_pred_rf = pipeline_rf.predict(x_test)

# Evaluate the model
accuracy_rf = accuracy_score(y_test, y_pred_rf)
print("Random Forest Model Accuracy:", accuracy_rf)

# Print confusion matrix
conf_matrix_rf = confusion_matrix(y_test, y_pred_rf)
print("Confusion Matrix for Random Forest:")
print(conf_matrix_rf)
```

```
Random Forest Model Accuracy: 0.6569152171268702
Confusion Matrix for Random Forest:
[[    5  5641]
 [    0 10796]]
```

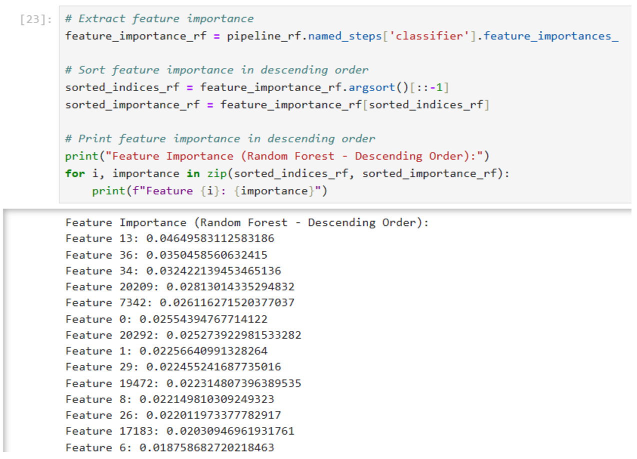
The model achieved an accuracy of approximately 65.69% on the test data. Looking at the confusion matrix, it seems the model is biased towards predicting the majority class (the negative class), as it correctly predicted 10796 instances of the negative class but only 5 instances of the positive class.



```
[22]: print(classification_report(y_test, y_pred_rf))
```

```
              precision    recall  f1-score   support

           0       1.00      0.05      0.09      5646
           1       0.67      1.00      0.80     10796

    accuracy                           0.67     16442
   macro avg       0.83      0.52      0.44     16442
weighted avg       0.78      0.67      0.56     16442
```

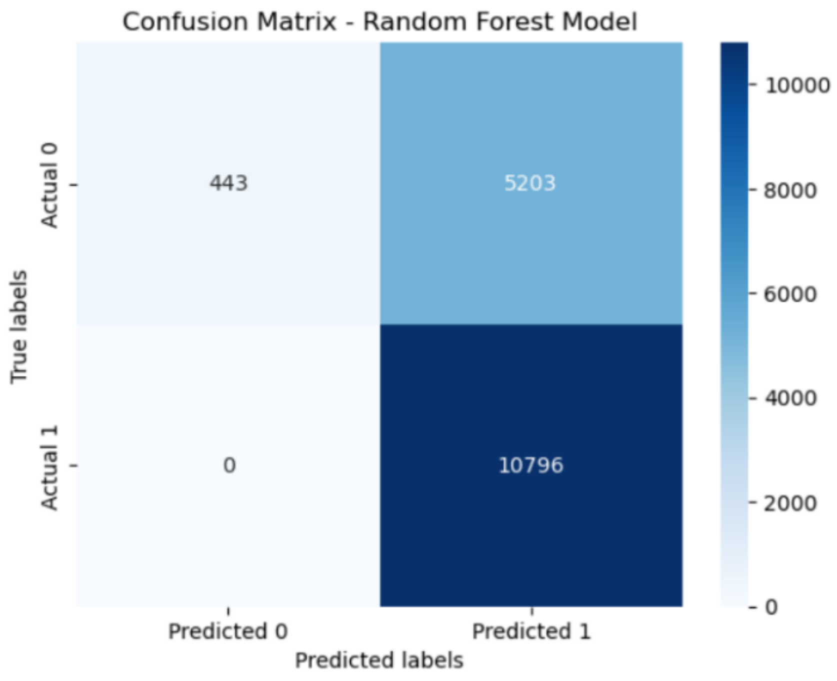
The model performs well in terms of precision for class 0 but poorly in terms of recall, indicating it tends to classify instances as class 0 too often. For class 1, the model has high precision and recall, indicating it performs well in correctly identifying instances of class 1. The F1-score provides a balanced measure of the model's performance, reflecting the trade-off between precision and recall. Overall, the model's performance seems to be better for class 1 compared to class 0.



```
[23]: # Extract feature importance
      feature_importance_rf = pipeline_rf.named_steps['classifier'].feature_importances_

      # Sort feature importance in descending order
      sorted_indices_rf = feature_importance_rf.argsort()[::-1]
      sorted_importance_rf = feature_importance_rf[sorted_indices_rf]

      # Print feature importance in descending order
      print("Feature Importance (Random Forest - Descending Order):")
      for i, importance in zip(sorted_indices_rf, sorted_importance_rf):
          print(f"Feature {i}: {importance}")
```

```
Feature Importance (Random Forest - Descending Order):
Feature 13: 0.04649583112583186
Feature 36: 0.0350458560632415
Feature 34: 0.032422139453465136
Feature 20209: 0.02813014335294832
Feature 7342: 0.026116271520377037
Feature 0: 0.02554394767714122
Feature 20292: 0.025273922981533282
Feature 1: 0.02256640991328264
Feature 29: 0.022455241687735016
Feature 19472: 0.022314807396389535
Feature 8: 0.022149810309249323
Feature 26: 0.022011973377782917
Feature 17183: 0.02030946961931761
Feature 6: 0.018758682720218463
```

Feature importance provides insight into which features are most influential in making predictions. In this case, the most important feature is indexed as 13, followed by features indexed as 36, 34, and so on, in descending order of importance. The importance values are normalized, with higher values indicating greater importance.



CONCLUSION

Based on the analysis of various models including Logistic Regression, Decision Tree, and Random Forest, as well as the assessment of feature importance, the following conclusions can be drawn

Logistic Regression: The logistic regression model demonstrated exceptional efficacy with high precision, recall, and F1- score for both classes, indicating accurate predictions across the test dataset. Logistic Regression outperformed other models in terms of overall accuracy and balanced performance across both classes. It demonstrated robustness in accurately predicting class labels. Based on the results, Logistic Regression appears to be the most suitable model for this classification task due to its high accuracy and balanced performance across classes.

Decision Tree: Pruning the decision tree using cost complexity pruning led to a reduction in overfitting, improving generalization performance. However, an unpruned decision tree still achieved high accuracy and performance metrics, albeit with a slight decrease compared to the pruned version. Decision Tree models, particularly those pruned with cost complexity pruning, showed improved generalization performance and better depth control, contributing to a more interpretable and efficient model. Decision Tree models, especially when pruned effectively, can serve as interpretable alternatives and may be preferred if model transparency is a priority.

Random Forest: The Random Forest model achieved a moderate accuracy on the test data. While it exhibited high precision and recall for class 1, it struggled with classifying instances of class 0, leading to an imbalance in performance between the two classes. Random Forest, while exhibiting decent performance, showed a bias towards predicting the majority class, highlighting potential issues with class imbalance and the need for further optimization or balancing techniques. Random Forest could benefit from addressing class imbalance issues and finetuning parameters to improve performance, particularly for minority classes.

Logistic Regression stands out as the top performer for accurately classifying news articles based on their content and sentiment. Feature importance analysis highlighted the most influential features in predicting the target labels across all models. Features such as Feature 13, Feature 29, and Feature 26 consistently emerged as highly important across different models, indicating their significant contribution to classification tasks.