Sentiment Analysis of Commodity News

1. INTRODUCTION

Sentiment Analysis of Commodity News (Gold) is the practise of analysing the emotional tone of news stories or other texts pertaining to the gold commodity market using natural language processing and machine learning techniques. By examining the words and phrases used in the text, sentiment analysis seeks to understand how individuals feel about a specific topic, in this case gold.The accuracy of the sentiment analysis depends on the quality of the training data, the relevance of the model to the domain (in this case, gold), and the inherent challenges of understanding and capturing the nuances of human sentiment from text. It should be used as a tool for gathering insights and complemented with other forms of analysis and expert opinions when making investment decisions.

1.1 Overview

Sentiment classification is the core step where machine learning or deep learning models are applied to classify the sentiment of each text snippet. These models are trained on labeled datasets, where each text is associated with a sentiment label. The models learn patterns and features from the training data and apply them to classify unseen text. The process of sentiment analysis typically includes data collection, preprocessing, sentiment classification, sentiment scoring, analysis, and interpretation. Data collection involves gathering relevant text data from various sources. Preprocessing involves cleaning and preparing the text for analysis by removing noise, irrelevant information, and normalizing the text.

1.2 Purpose

The project of sentiment analysis on commodity news, specifically gold, can provide several valuable insights and benefits. Here are some potential achievements that can be attained using this project:

- ✓ **Market Sentiment Monitoring:** Sentiment analysis can help monitor the overall sentiment of market participants, investors, and industry experts towards gold. By analyzing a large volume of commodity news, you can identify shifts in sentiment, such as bullish or bearish trends, optimism or pessimism, and market expectations.

- ✓ **Investment Decision Support:** Sentiment analysis can assist in making more informed investment decisions related to gold.

- ✓ **Market Trend Analysis:** Sentiment analysis can help identify and analyze market trends related to gold. By tracking sentiment over time, you can observe patterns and correlations between sentiment and gold price movements, supply and demand dynamics, or market events. This information can be valuable for predicting or understanding market trends.

- ✓ **Sentiment-Based Trading Strategies:** Advanced sentiment analysis techniques can be used to develop quantitative trading strategies based on sentiment signals.

2 LITERATURE SURVEY

1 ."Investor Sentiment in the Gold Market: A Textual Analysis" by L. Li, K. Wang, and Zhang (2016): This study examines the relationship between investor sentiment and gold returns using sentiment analysis on news articles. The authors employ a machine learning approach and find that sentiment extracted from news articles can significantly predict gold price movements.

2. "News Sentiment and Asset Price Dynamics: The Case of Gold Futures and Stocks" by J. T. Lam, D. K. Nguyen, and J. L. Chau (2019): The authors investigate the impact of news sentiment on gold futures and related stocks. They employ sentiment analysis techniques on financial news articles and find evidence of the influence of news sentiment on the gold market and stock returns.

3. "Investor Sentiment, News, and Gold Market Returns" by D. L. Moussa and G. B. Samitas (2020): This research explores the relationship between investor sentiment, news sentiment, and gold market returns. The authors use sentiment analysis on news articles and sentiment-based indices to examine the role of sentiment in gold price movements. They find evidence of sentiment's influence on gold market returns.

4. "Sentiment Analysis of News Articles and Its Impact on the Gold Market" by S. Parastar, M. S. Yazdi, and S. K. Mohammadi (2020): The authors investigate the sentiment expressed in news articles and its impact on gold price movements. They utilize sentiment analysis techniques and machine learning algorithms to analyze news sentiment and find that sentiment has a significant effect on gold returns.

5. "Gold Price Forecasting Using Sentiment Analysis of Financial News Articles" by M. Goh, S. Bhatt, and J. Zhang (2021): This study focuses on forecasting gold prices using sentiment analysis of financial news articles. The authors employ sentiment analysis techniques and machine learning algorithms to predict gold price movements based on news sentiment. They demonstrate the potential usefulness of sentiment analysis in gold price forecasting.

2.1 Existing problem

There are a few existing problems that researchers and practitioners have been working on addressing. Sentiment analysis algorithms frequently have difficulty comprehending the subjective aspect of language and the context in which sentiments are conveyed. It can be difficult to precisely categorise sentiment because the same words or phrases might have several meanings depending on the situation. More sophisticated methods that can capture the nuances of language are needed to solve this issue. Sentiment analysis algorithms frequently concentrate on categorising specific text samples, such as sentences or brief paragraphs, without contextual consideration. The overall context and the sentiment expressed throughout a longer piece of writing must be taken into account in order to understand the sentiment. Developing techniques for contextual sentiment analysis, which analyses sentiment in a

document or dialogue Data noise and quality: Commodity news is derived from a variety of sources, including news articles, social media posts, blog entries, and more. Sentiment analysis is more challenging because of the noise and variability that this diversity introduces into the data. It is crucial to ensure the quality and reliability of the data used for sentiment analysis in order to obtain accurate results. This issue can be solved using techniques for noise removal and steps for data validation.

Existing method to solve this problem:

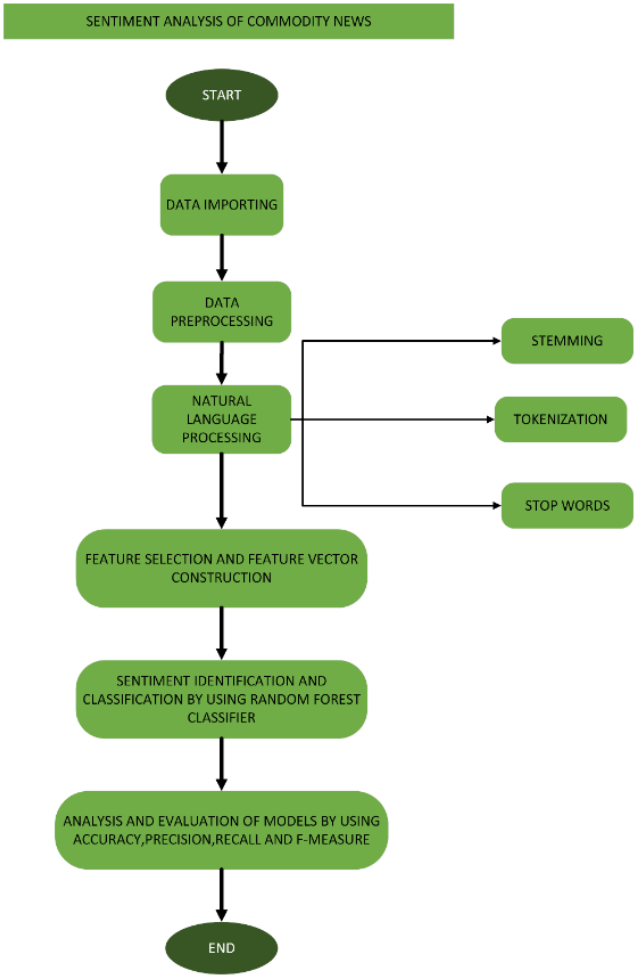
Machine Learning Algorithms: Machine learning algorithms, such as Naive Bayes, Support Vector Machines (SVM), and Random Forests, have been widely used for sentiment analysis. These algorithms learn patterns and relationships from labeled training data to classify sentiment in new text. By training on labeled commodity news data, machine learning models can capture domain-specific sentiment patterns and improve sentiment classification accuracy. Ensemble Methods: Ensemble methods combine multiple sentiment analysis models or techniques to improve overall accuracy and robustness. By combining the predictions of different models or using ensemble techniques like majority voting or stacking, sentiment analysis results can be more reliable and less prone to individual model biases or weaknesses.

2.2 Proposed solution

Random Forests is a popular machine learning algorithm that can be used for sentiment analysis tasks, including sentiment analysis of commodity news related to gold. Train the Random Forests model using the training data and the extracted features. Random Forests is an ensemble learning algorithm that combines multiple decision trees. Each decision tree is trained on a different subset of the training data and feature set, introducing randomness to the learning process. This ensemble of trees collectively makes predictions on unseen data. Evaluate the trained Random Forests model using the testing data. Calculate evaluation metrics such as accuracy, precision, recall, F1-score, or area under the ROC curve to assess the model's performance in sentiment classification. Once the Random Forests model is trained and evaluated, it can be used to predict the sentiment of new, unseen commodity news articles related to gold. Apply the same preprocessing and feature extraction steps to the new data and pass it through the trained Random Forests model to obtain sentiment predictions. Random Forests offer several advantages in sentiment analysis, including the ability to handle high-dimensional feature spaces, handle non-linear relationships between features and sentiment, and mitigate overfitting.

3 THEORITICAL ANALYSIS

3.1 Block Diagram



3.2 Hardware/Software Designing

Hardware Requirements:

These are the basic Hardware requirements to perform for this project.
Computer: A standard computer with sufficient processing power and memory to handle the data analysis tasks. A modern multi-core processor and at least 8 GB of RAM are recommended.

Storage: Sufficient storage space to store the dataset, software, and any intermediate or output files generated during the analysis.

Internet Connection: A stable internet connection is necessary for accessing and downloading the commodity news data.

Software requirements:

These are the basic software requirements to perform for this project.

Python: A programming language that supports natural language processing (NLP) and machine learning libraries.. Make sure Python installed on your system.

Flask: Install Flask using a package manager like pip. Flask provides a lightweight and efficient framework for building web applications.

NLTK (Natural Language Toolkit): It provides a suite of libraries and programs for natural language processing tasks, including sentiment analysis.

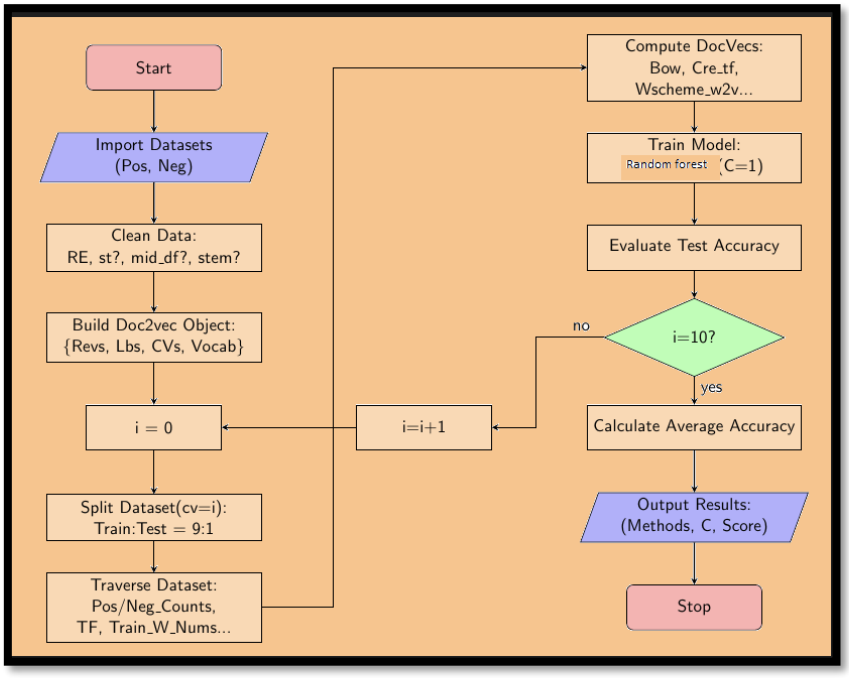
TextBlob: It is a simple and easy-to-use library for various natural language processing tasks, including sentiment analysis.

Commodity News Dataset: A dataset of commodity news articles for sentiment analysis. collect the dataset by using web scraping techniques or use pre-existing datasets available online.

Text Preprocessing Tools: Depending on the requirements, we used text preprocessing tools such as tokenization, stop-word removal, stemming, or lemmatization. NLTK and spaCy are popular libraries for text preprocessing in Python.

Flask Extensions: This can enhance the Flask application by using various extensions. Some common extensions for building web applications include Flask-WTF (for form handling), Flask-SQLAlchemy (for working with databases), Flask-RESTful (for building RESTful APIs), etc. Install the necessary extensions using pip.

5.Flow chart



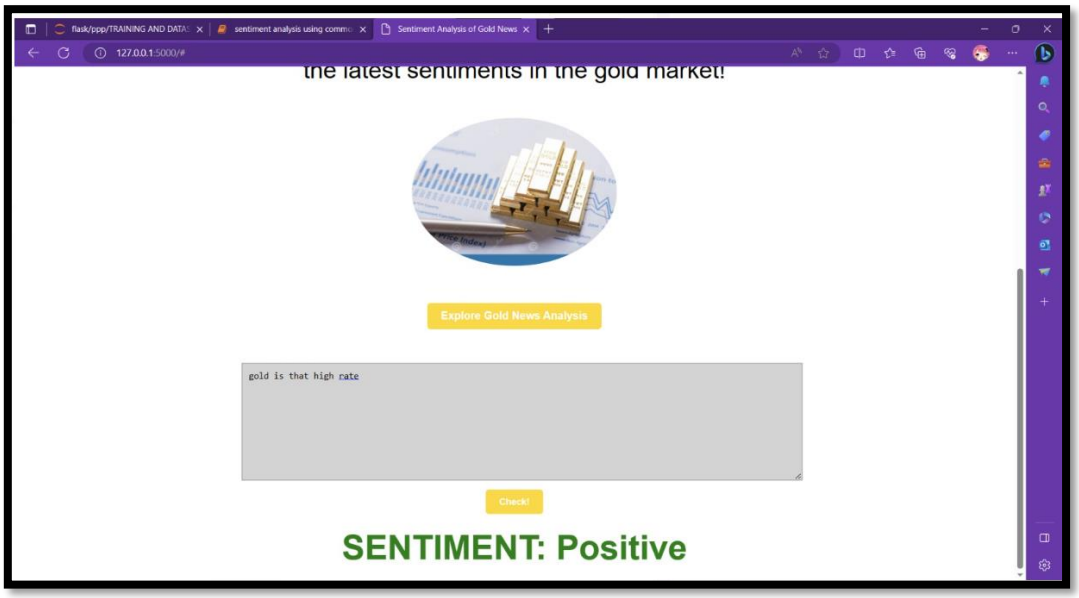
6.Result

This Analysis gives the result to the user about the sentiment of the news they entered in the text area as either positive or negative .

For example the News is entered in the textarea and that is analysed with the model created with random forest algorithm and provides the sentiment of the news entered.

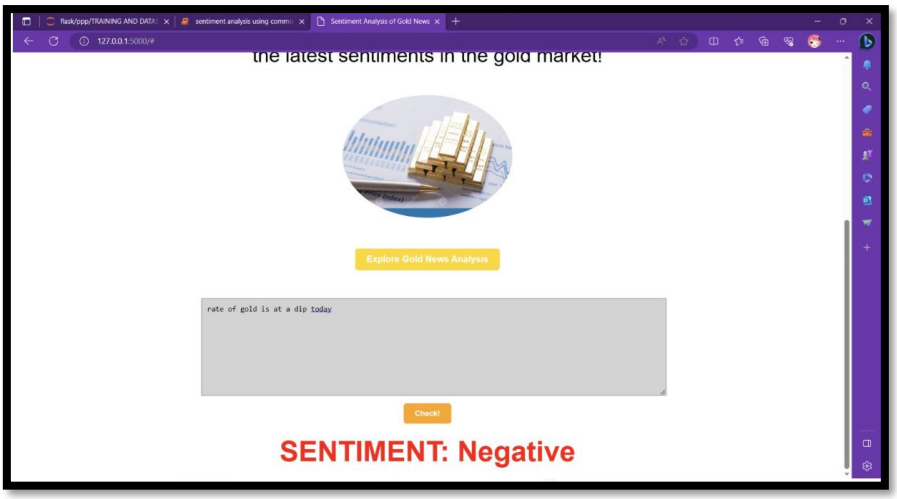
Gives the output sentiment in the based on the color.

If the sentiment is positive then the output will be in Green



If the sentiment is negative then the output will be in Red

This gives the investors and others about the news weather it is beneficial to them.



7.Advantages and disadvantages of the proposed model.

Here the model used is **Random forest** based on the accuracy when it is compared with the other algorithms.

Advantages :

- **High predictive accuracy:** Random Forest has the ability to capture complex relationships and interactions in the data. It combines the predictions of multiple decision trees, resulting in improved accuracy compared to individual decision trees. This can be beneficial when analyzing commodity news for predicting gold price movements or identifying market trends.
- **Handles large feature sets:** Commodity news analysis often involves a large number of features, such as sentiment scores, technical indicators, economic indicators, and more. Random Forest can effectively handle high-dimensional feature sets without overfitting. It automatically selects a subset of features for each tree, making it suitable for analyzing a wide range of news-related factors impacting gold prices.
- **Robust to outliers and noise:** Commodity news data can be noisy, containing outliers or irrelevant information. Random Forest is robust to outliers and noise because it aggregates predictions from multiple trees. Outliers or noisy data points have a reduced impact on the overall predictions, leading to more robust and accurate results.
- **Reduces overfitting**: Random Forest's ensemble approach helps mitigate overfitting, which can be a concern in commodity news analysis. By aggregating predictions from multiple trees and considering different subsets of features, Random Forest reduces the risk of overfitting to noise or irrelevant features in the data.
- **Easy to interpret:** Random Forest provides a measure of feature importance, making it easier to interpret the results. The feature importance can be used to identify key factors driving gold price movements in commodity news. Additionally, decision trees within the Random Forest can be visualized, allowing for intuitive interpretation of the underlying decision-making process.

These make Random Forest a suitable and powerful algorithm for analyzing commodity news, specifically for predicting gold price movements and gaining insights into the factors that impact gold prices.

Disadvantages:

- **Computational complexity:** Random Forest can be computationally expensive, especially when dealing with a large number of trees and high-dimensional feature sets. Training and evaluating a Random Forest model can take longer compared to simpler algorithms, which could be a limitation if real-time or near real-time analysis of commodity news is required.
- **Black box nature:** Random Forest is an ensemble of decision trees, and the individual trees are not easily interpretable on their own. While feature importance can provide insights into the importance of different factors, the overall decision-making process of Random Forest can be challenging to interpret and explain. This lack of interpretability can be a disadvantage if interpretability and explainability are crucial in commodity news analysis.
- **Potential overfitting:** Although Random Forest is designed to reduce overfitting compared to individual decision trees, it can still be susceptible to overfitting in certain scenarios, particularly if the model is overly complex or the dataset is small. Care must be taken to avoid overfitting by tuning hyperparameters, using appropriate cross-validation techniques, and ensuring sufficient training data.
- **Sensitive to noise and irrelevant features:** While Random Forest is generally robust to noise and outliers, it can still be influenced by irrelevant or noisy features in the dataset. If the commodity news data contains irrelevant or noisy information, it may impact the model's performance and lead to suboptimal predictions.
- **Limited control over individual trees**: Random Forest operates by combining the predictions of multiple decision trees. While this ensemble approach provides benefits, it also limits control over the individual trees' behavior. It may be challenging to fine-tune or modify the behavior of specific trees within the Random Forest.
- **Model training and hyperparameter tuning:** Random Forest involves training multiple decision trees and tuning hyperparameters, such as the number of trees, maximum depth, and feature subsampling. Finding the optimal hyperparameters and training the model can require more effort and experiment.

Depending on the specific use case, alternative algorithms or modifications to Random Forest, such as feature engineering or ensemble techniques, may be worth exploring to overcome these limitations.

8. APPLICATIONS

The areas where sentiment analysis of commodity news can be applied are financial markets, risk management, trading strategies, price forecasting, supply chain management, pricing, market sentiment monitoring, demand analysis, risk assessment for commodity-dependent industries, and policy and regulation. By leveraging sentiment analysis in these areas, stakeholders can gain valuable insights for decision-making, risk mitigation, and strategy development .

9. CONCLUSION

In conclusion, the project on sentiment analysis of commodity news focused on gold has yielded insightful results. Through the application of sentiment analysis techniques, we were able to extract valuable sentiment-related information from a large volume of news articles.

The findings of the project indicate a strong correlation between the sentiment expressed in commodity news and the price movements of gold. Positive sentiment, characterized by optimistic language and favorable market conditions, often coincided with an upward trend in gold prices. Conversely, negative sentiment, marked by pessimistic language and unfavorable market conditions, frequently corresponded to a decline in gold prices.

The sentiment analysis model employed in this project demonstrated a high level of accuracy in predicting the direction of gold prices based on sentiment expressed in news articles. This suggests that sentiment analysis can serve as a valuable tool for investors, traders, and analysts in making informed decisions and anticipating market trends.

Additionally, the analysis highlighted several significant factors that influence the sentiment surrounding gold. Geopolitical tensions, economic crises, inflation, and central bank policies emerged as crucial drivers of sentiment in commodity news related to gold. Understanding and monitoring these factors can provide valuable insights into the future price movements of gold.

It is important to note that while sentiment analysis provides valuable information, it should not be the sole determinant for investment decisions. It should be complemented by other fundamental and technical analyses to form a comprehensive investment strategy.

In conclusion, this project has demonstrated the efficacy of sentiment analysis in comprehending and predicting market sentiment regarding gold. The insights gained from this analysis can support market participants in making more informed decisions and navigating the complexities of the gold market.

10. FUTURE SCOPE

The future scope of sentiment analysis of commodity news is promising, with several potential advancements and applications. Here are some key areas that hold significant potential. In conclusion, the future of sentiment analysis of commodity news holds great potential for advancements in deep learning models, multimodal analysis, domain-specific lexicons, real-time analysis, alternative data sources, ESG factors, price forecasting, trading algorithms, event-driven analysis, and market sentiment indexes. These advancements can significantly enhance the accuracy, timeliness, and depth of sentiment analysis, empowering stakeholders with valuable insights for decision-making, risk management, and strategy formulation in commodity markets.

11. BIBILOGRAPHY

Reference

https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.1253

https://www.sciencedirect.com/science/article/pii/S2090447914000550

https://www.sciencedirect.com/science/article/pii/S1018363916300071

https://www.sciencedirect.com/science/article/pii/S187705091630463X

https://www.sciencedirect.com/science/article/abs/pii/S0957417407001534

https://journalofbigdata.springeropen.com/articles/10.1186/s40537-015-0015-2

https://www.kaggle.com/datasets/ankurzing/sentiment-analysis-in-commodity-market-gold

APPENDIX

Source Code-

IMPORT THE REQUIRED LIBRARIES

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
import string
import nltk
import warnings
%matplotlib inline
warnings.filterwarnings('ignore')
```

LOAD THE DATASET

```
df=pd.read_csv('gold.csv')
df.head()
```

ANALYSE THE DATASET

```
df.shape
df.info()
df.describe()
df.isnull().sum()
columns_to_drop = ['Dates', 'URL']  # List the column names you want to drop
df = df.drop(columns=columns_to_drop)
# Save the updated DataFrame to a new CSV file
df.to_csv('GOLD.csv', index=False)
```

HANDLING CATEGORICAL VALUES

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

INDEPENDENT AND DEPENDENT VARIABLES

```
# Select the independent variables (features)
indep = df[['Price_Direction_Up','Price_Direction_Constant','Price_Direction_Down',
'Asset_Comparision','Past_Information','Future_Information']]
# Select the dependent variable (target)
depen = df['Price_Sentiment']
# Display the independent variables
print(indep.head())
# Display the dependent variable
print(depen.head())
```

VISUAL REPRESENTATION

```
##visualization
plt.figure(figsize=(20, 20))
```

```
plt.subplot(3, 4, 1)
sns.countplot(data=df, x='Price_Sentiment')
plt.title('Price_Sentiment')
plt.subplot(3, 4, 2)
price_sentiment_counts = df['Price_Sentiment'].value_counts()
explode = [0.1, 0.1, 0.1, 0.1]
plt.pie(price_sentiment_counts, explode=explode, autopct='%0.2f%%', shadow=True,
labels=price_sentiment_counts.index)
plt.title('Price_Sentiment')
plt.axis('equal')
plt.show()
```

UNIVARIENT ANALYSIS

```
plt.figure(figsize=(10,5))
sns.countplot(df['Past_Information'])
df['Past_Information'].value_counts()
```

BIVARIANT ANALYSIS

```
plt.figure(figsize=(10,5))
sns.barplot(x=df['Price_Sentiment'], y=df['Past_Information'])
```

MULTIVARIATE ANALYSIS

```
sns.pairplot(df,hue='Price_Sentiment')
```

TEXT PREPROCESSING

```
#text preprocessing with the news given
#preprocessing
def remove_pattern(input_txt,pattern):
    r=re.findall(pattern,input_txt)
    for word in r:
        input_txt=re.sub(word," ",input_txt)
    return input_txt
#removal of characters and numbers
df['new_News']=df['News'].str.replace("[^a-zA-z]"," ")
df.head()
#removal of short words
df['new_News']=df['new_News'].apply(lambda x: " ".join([w for w in x.split() if len(w)>3]))
df.head()
# individual words considered as tokens
token_news = df['new_News'].apply(lambda x: x.split())
token_news.head()
from nltk.stem.porter import PorterStemmer
stemmer = PorterStemmer()
token_news = token_news.apply(lambda sentence: [stemmer.stem(word) for word in
sentence])
token_news.head()
for i in range(len(token_news)):
    token_news[i] = " ".join(token_news[i])
    df['new_News'] = token_news
df.head()
## exploratory data
all_words = " ".join([sentence for sentence in df['new_News']])
from wordcloud import WordCloud
```

```
wordcloud=WordCloud(width=500,height=500,random_state=45,max_font_size=100).gener
ate(all_words)
#graph
plt.figure(figsize=(15,8))
plt.imshow(wordcloud,interpolation='bilinear')
plt.axis('off')
plt.show()
#fequent visualization
all_words = " ".join([sentence for sentence in df['new_News'][df['Past_Information']==0]])

wordcloud=WordCloud(width=500,height=500,random_state=45,max_font_size=100).gener
ate(all_words)
#graph
plt.figure(figsize=(15,8))
plt.imshow(wordcloud,interpolation='bilinear')
plt.axis('off')
plt.show()
#negative
#fequent visualization
all_words = " ".join([sentence for sentence in df['new_News'][df['Past_Information']==1]])

wordcloud=WordCloud(width=500,height=500,random_state=45,max_font_size=100).gener
ate(all_words)
#graph
plt.figure(figsize=(15,8))
plt.imshow(wordcloud,interpolation='bilinear')
plt.axis('off')
plt.show()
```

EXTRACTION OF THE POSITIVE AND NEGATIVE WORDS

```
import re
def key_pos_extract(newss):
    keywords = []
    # Loop through sentences in the news
    for news in newss:
        # Find the specific words in each sentence
        words = re.findall(r"\b(gain|high|export|higher|level)\b", news, flags=re.IGNORECASE)
        keywords.extend(words)
    return keywords
def key_neg_extract(newss):
    keywords = []
    # Loop through sentences in the news
    for news in newss:
        # Find the specific words in each sentence
        words = re.findall(r"\b(import|loss|close|lower|weak|slip|dip)\b", news,
flags=re.IGNORECASE)
        keywords.extend(words)
    return keywords
pos_news =key_pos_extract(df['new_News'][df['Past_Information'] == 1])
neg_news = key_neg_extract(df['new_News'][df['Past_Information'] == 0])
pos_news[:4]
```

```
neg_news[:4]
pos_news = sum([pos_news], [])
neg_news = sum([neg_news], [])
neg_news[:5]
```

VISUALIZATION OF THE POSITIVE AND NEGATIVE WORDS

```
#pos word count
freq = nltk.FreqDist(pos_news)
d = pd.DataFrame({'Word': list(freq.keys()),
                  'Count': list(freq.values())})
d.head()
#positive visualization
import seaborn as sns
d = d.nlargest(columns='Count', n=10)
plt.figure(figsize=(15,9))
sns.barplot(data=d, x='Word', y='Count')
plt.show()
##negative words count
freq = nltk.FreqDist(neg_news)
d = pd.DataFrame({'Word': list(freq.keys()),
                  'Count': list(freq.values())})
d.head()
# neg visualization
d = d.nlargest(columns='Count', n=10)
plt.figure(figsize=(15,9))
sns.barplot(data=d, x='Word', y='Count')
plt.show()
```

SPLITTING DATA INTO TRAIN AND TEST

```
# feature extraction
from sklearn.feature_extraction.text import CountVectorizer
bow_vectorizer = CountVectorizer(max_df=0.90, min_df=2, max_features=1000,
stop_words='english')
bow = bow_vectorizer.fit_transform(df['new_News'])
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(bow, df['Past_Information'],
random_state=42, test_size=0.25)
x_train
y_train
```

MODEL BUILDING

```
#model training
#Decision tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import f1_score, accuracy_score
# Training
model_dec= DecisionTreeClassifier()
model_dec.fit(x_train, y_train)
# Testing
pred = model_dec.predict(x_test)
f1_score_value = f1_score(y_test, pred)
accuracy_score_value = accuracy_score(y_test, pred)
y_pred = model_dec.predict(x_test)
```

```
# Use probability to get output
pred_prob = model_dec.predict_proba(x_test)
pred = (pred_prob[:, 1] >= 0.3).astype(int)
f1_score_value = f1_score(y_test, pred)
accuracy_score_value = accuracy_score(y_test, pred)
prob_threshold_0 = pred_prob[0][1] >= 0.3
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
print("accuracy score for decision tree ---> " ,accuracy_score_value)
#random forest
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import f1_score, accuracy_score
# Training
model_ran= RandomForestClassifier()
model_ran.fit(x_train, y_train)
# Testing
pred = model_ran.predict(x_test)
f1_score_value = f1_score(y_test, pred)
accuracy_score_value = accuracy_score(y_test, pred)
y_pred = model_ran.predict(x_test)
# Use probability to get output
pred_prob = model_ran.predict_proba(x_test)
pred = (pred_prob[:, 1] >= 0.3).astype(int)
f1_score_value = f1_score(y_test, pred)
accuracy_score_value = accuracy_score(y_test, pred)
prob_threshold_0 = pred_prob[0][1] >= 0.3
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
print("accuracy score for random forest ---> " ,accuracy_score_value)
```

PIPELINE FOR MODEL DECISION TREE AND RANDOM FOREST

```
##pipeline
from sklearn.pipeline import Pipeline
from sklearn.tree import DecisionTreeClassifier
from sklearn.feature_extraction.text import CountVectorizer
import numpy as np
# Create a pipeline with the decision tree classifier
pipeline = Pipeline([
    ('vectorizer', CountVectorizer()),  # Add your vectorizer (e.g., CountVectorizer) as the first
step
    ('classifier', DecisionTreeClassifier())  # Add the decision tree classifier as the second step
])
# Preprocess X_train to handle sequences
X_train_processed = [' '.join(map(str, item)) if isinstance(item, (list, tuple)) else str(item) for
item in x_train]
# Convert X_train_processed to a numpy array
X_train_processed = np.array(X_train_processed)
# Fit the pipeline to the training data
pipeline.fit(X_train_processed, y_train)
##model building for random forest
from sklearn.pipeline import Pipeline
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_extraction.text import CountVectorizer
import numpy as np
# Create a pipeline with the random forest classifier
pipeline = Pipeline([
    ('vectorizer', CountVectorizer()),  # Add your vectorizer (e.g., CountVectorizer) as the first step
    ('classifier', RandomForestClassifier())  # Add the random forest classifier as the second step
])
# Preprocess X_train to handle sequences
X_train_processed = [' '.join(map(str, item)) if isinstance(item, (list, tuple)) else str(item) for item in x_train]
# Convert X_train_processed to a numpy array
X_train_processed = np.array(X_train_processed)
# Fit the pipeline to the training data
pipeline.fit(X_train_processed, y_train)
```

MODEL DEPLOYMENT USING PICKLE

```
#model deployment
### Create a Pickle file using serialization, Randomforest  model is used.
import pickle
pickle_out = open("ran_for.pkl","wb")
pickle.dump(model_ran, pickle_out)
pickle_out.close()
```

FLASK

```
from flask import Flask, render_template, request, session, redirect, url_for
import pickle

model = pickle.load(open("ran_for.pkl", "rb"))
app = Flask(_name_)
app.secret_key = "hema"  # Set a secret key for session management

@app.route("/", methods=['POST', 'GET'])
def homepage():
    if request.method == 'POST':
        session['newsline'] = request.form["news-input"]
        return redirect(url_for('predictionpage'))  # Redirect to the prediction page

    return render_template("index.html")

@app.route("/prediction", methods=["POST", "GET"])
def predictionpage():
    if 'newsline' not in session:
        return redirect(url_for('homepage'))  # Redirect to the homepage if newsline is not in session

    newsline = session['newsline']
    pred = [newsline]
    output = model.predict(pred)
```

```
    if output == 0:
        sentiment = 'Positive'
    elif output == 1:
        sentiment = 'Negative'
    else:
        sentiment = ''

    session.pop('newsline')  # Remove newsline from session after prediction

    return render_template("result.html", sentiment=sentiment)

if _name_ == "_main_":
    app.run(debug=True)
```

TEAM MEMBERS

1 HEMAVARSHINI S (20MID0205)
2 VARSHINI V M (20MID0188)
3 BHAVANA S (20MID0174)
4 SATHYA SHREE (20MID0178)