Project Report

Opinion mining on Russia-Ukraine conflicts in social media - Twitter

Project Report submitted on the fulfilment of the requirements of Post Graduate Diploma in Big Data Analytics

Authors:		Co-ordinators:
Ms. Sneha Sangale	(230360925034)	Ms. Roopa Panikar
Mr. Sohan Chaudhari	(230360925035)	Ms. Divya Das
Mr. Sumesh Sonawane	(250360925036)	Ms. Soorya M.
Mr. Akshay Sultane	(250360925038)	
Mr. Suraj Ghonsikar	(250360925039)	

STDC

CDAC Thiruvananthapuram

Trivandrum, Kerala 695581

Table of Contents

I. Abstract

Social media websites have emerged as one of the platforms to raise users' opinions and influence the way any business is commercialized. Opinion of people matters a lot to analyse how the propagation of information impacts the lives in a large-scale network like Twitter. Sentiment analysis of the tweets determine the polarity and inclination of vast population towards specific topic, item or entity. These days, the applications of such analysis can be easily observed during public elections, movie promotions, brand endorsements and many other fields.

The Russia – Ukraine war has become a highly contested and polarizing conflict. Generating intense debates and discussion on social media platform (twitter). We read that all debates tweet how to goes on. In this project we predict public sentiment on Russia Ukraine conflicts.

This project focuses on sentiment analysis of Twitter data related to the Russia-Ukraine conflict. We aim to predict public sentiment through various steps, language analysis, hashtag tracking, including tokenization and sentiment classification & by creating a machine learning model. Our approach is informed by relevant literature and aims to highlight the dynamic views expressed on this controversial topic in the digital realm and to predict the sentiments of tweets using machine learning models.

II. Introduction

Social networks are currently one of the most important platforms for population information. Abstract Users of social networks share numerous posts in many domains, including social, economic, and political, which reflect their attitudes and feelings about world events and events. Twitter as a social network is a key source of user’s emotions and perspectives. Users' views can be analysed using Natural Language Processing and data retrieval.

Natural Language Processing (NLP) focuses on the design and analysis of computational. Algorithms and Representations for Natural Human Language Processing. To provide new computational capabilities around human language. indeed, we can extract information from many texts using NLP techniques.

Machine Learning, a subset of artificial intelligence, equips us with the ability to teach computers how to learn from data and make predictions or decisions. This project leverages the power of Machine Learning (ML) to gain insights from this vast dataset. ML is an essential tool that enables us to extract patterns, classify sentiments, and uncover trends in Twitter data related to this contentious issue. Our ML approach involves training models to predict sentiments. By fine-tuning algorithms and models, we aim to accurately capture the nuances of public sentiment, offering a deeper understanding of the Russia-Ukraine conflicts.

Opinion Mining is used for various types of use like social media monitoring and customer feedback. Opinion mining is also called as sentiment analysis. At present, sentiment analysis is used to monitor social media platforms and track public sentiment regarding their products, campaigns, or reputation. This can help in crisis management and marketing strategies. On account that there are a greater number of tweets, their sentiment recognition may be an important task.

For the purpose of our project, we decided to choose the Russia-Ukraine war tweets Dataset. The Russia-Ukraine war dataset contains various columns related to Russia-Ukraine war Tweets around the world. Contents of this dataset include 30,000 raw tweets in total and other columns that used for EDA part.

III. Literature Survey

Proposed the system deals with performing functions dynamically through an online social media i.e., Twitter. Twitter posts of electronic products creates a dataset. Tweets are short messages with slang words and mis-spellings. So, the sentence level sentiment analysis is performed. This can be done in seven phases. In the first phase, input data is given. Here the input data refers to a username or a hashtag. Then, the number of tweets to be analysed are specified, this step is performed before feature extraction. Processing steps include removing URLs, removing stop-words, avoiding mis-spellings and slang words. Mis-spellings square measure avoided by commutation continual characters with two occurrences. Next part is feature extraction. A feature vector is formed mistreatment relevant options.

Proposed that the twitter data analysis has been made on various datasets to mine the sentiments or opinions. They have implemented and compared various techniques on a particular dataset to achieve better accuracy, and they have also found that logistic regression works faster than other techniques. The biggest drawback of supervised learning techniques is that it does not produces the best result when the dataset is not sufficient.

The tweets can be analysed and characterized based on the emotions used by the social users. And classified by using the polarity of the tweet where it is either positive or negative. If the tweet has both positive and negative elements, the more dominant sentiment should be picked as the final label. We use the dataset from Kaggle which was crawled and labelled positive/negative. The data provided comes with emoticons, usernames and hashtags which are required to be processed and converted into a standard form. It also needs to extract useful features from the text such unigrams and bigrams which is a form of representation of the “tweet”.

This million unique English-language tweets from January 2022 to the first week of March 2022 for classifying the tweets. Overall, there were 13.88% neutral tweets, 54.29% negative tweets, and 31.83% good tweets. The most frequently used words were revealed via bag of words and sum vector.

IV. Proposed System

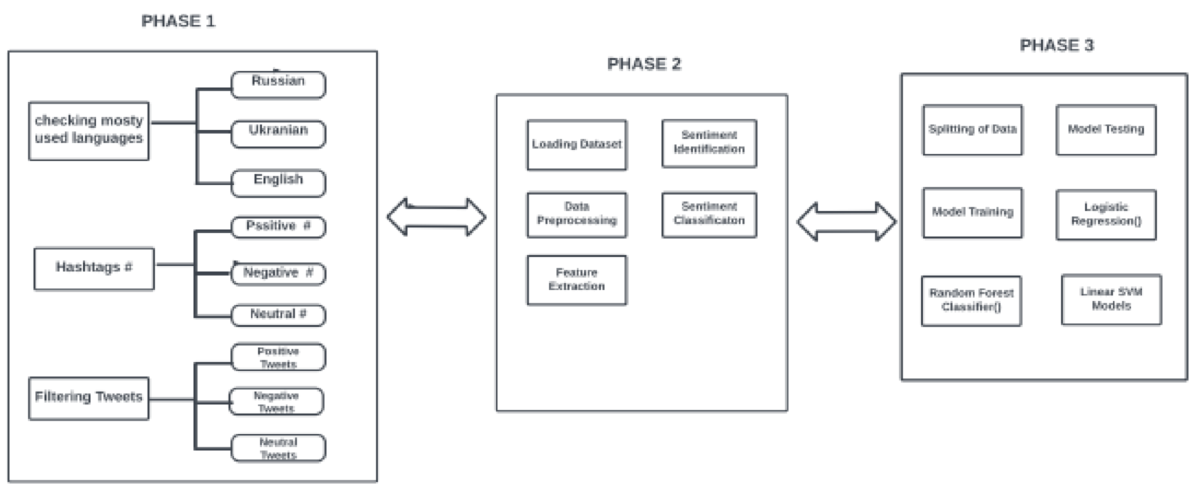
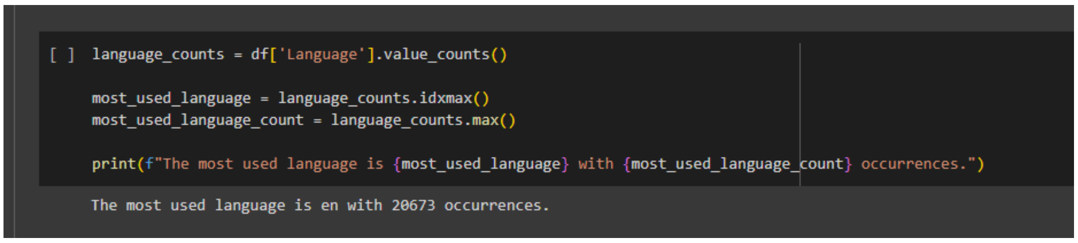


Fig – Block Diagram

Sentiment analysis considers emotions and opinions of masses about an event, idea, or topic from publicly available unstructured data. The development of technology to analyse and explain the process in multiple languages has been the subject of many studies.

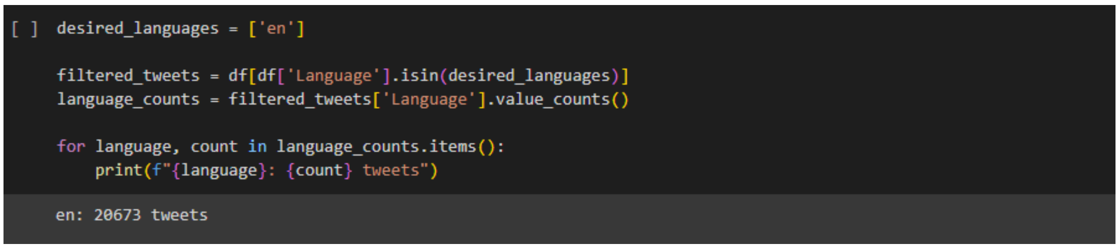
This project used machine learning to present new information about public opinion on economic sanctions based on approximately 1 million social media posts made in 109 countries during the conflict between Russia and Ukraine. It demonstrates the geographic disparity between official positions and popular opinion.

1. Checking Which Languages are mostly used in dataset



```
[ ]  language_counts = df['Language'].value_counts()

     most_used_language = language_counts.idxmax()
     most_used_language_count = language_counts.max()

     print(f"The most used language is {most_used_language} with {most_used_language_count} occurrences.")

     The most used language is en with 20673 occurrences.
```

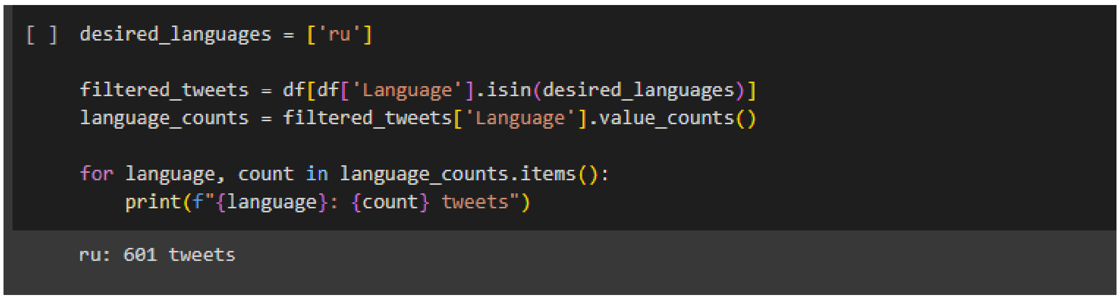
2. Filtering Tweets by English Languages



```
desired_languages = ['en']

filtered_tweets = df[df['Language'].isin(desired_languages)]
language_counts = filtered_tweets['Language'].value_counts()

for language, count in language_counts.items():
    print(f"{language}: {count} tweets")
```

```
en: 20673 tweets
```

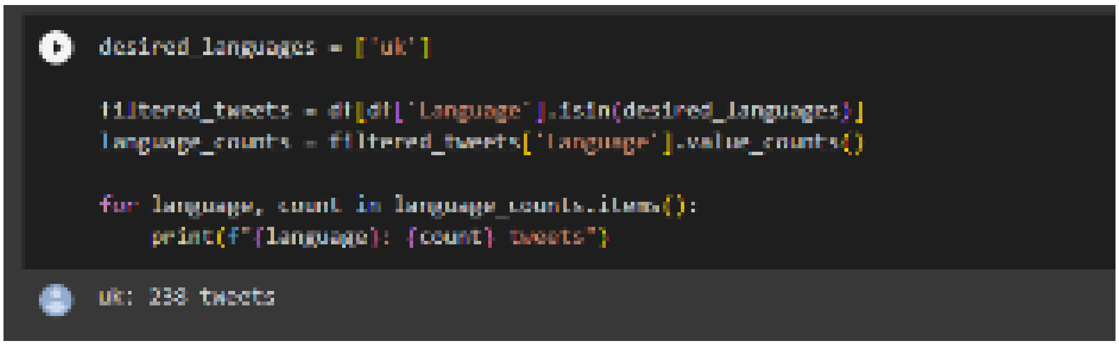
3. Filtering Tweets by Russian Languages



```
desired_languages = ['ru']

filtered_tweets = df[df['Language'].isin(desired_languages)]
language_counts = filtered_tweets['Language'].value_counts()

for language, count in language_counts.items():
    print(f"{language}: {count} tweets")
```

```
ru: 601 tweets
```

4. Filtering Tweets by Ukrainian Languages



```
desired_languages = ['uk']

filtered_tweets = df[df['Language'].isin(desired_languages)]
language_counts = filtered_tweets['Language'].value_counts()

for language, count in language_counts.items():
    print(f"{language}: {count} tweets")
```

```
uk: 238 tweets
```

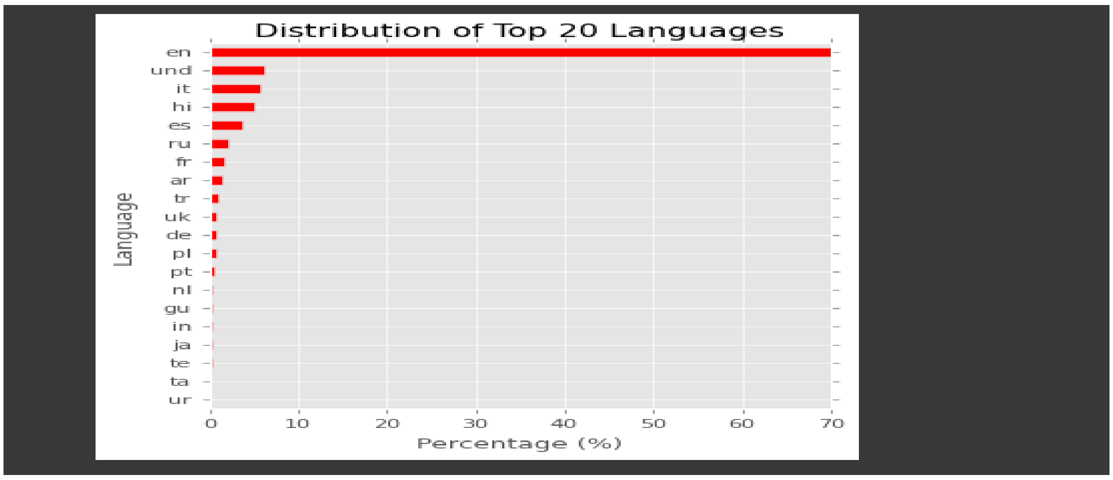
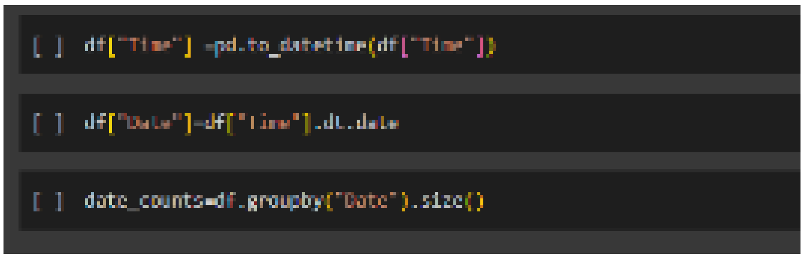


Fig – Plotting of Distribution of Top 20 Languages

5. Tweets Frequency by Date and months



```
[ ] df["Time"] =pd.to_datetime(df["Time"])
```

```
[ ] df["Date"]=df["Time"].dt.date
```

```
[ ] date_counts=df.groupby("Date").size()
```

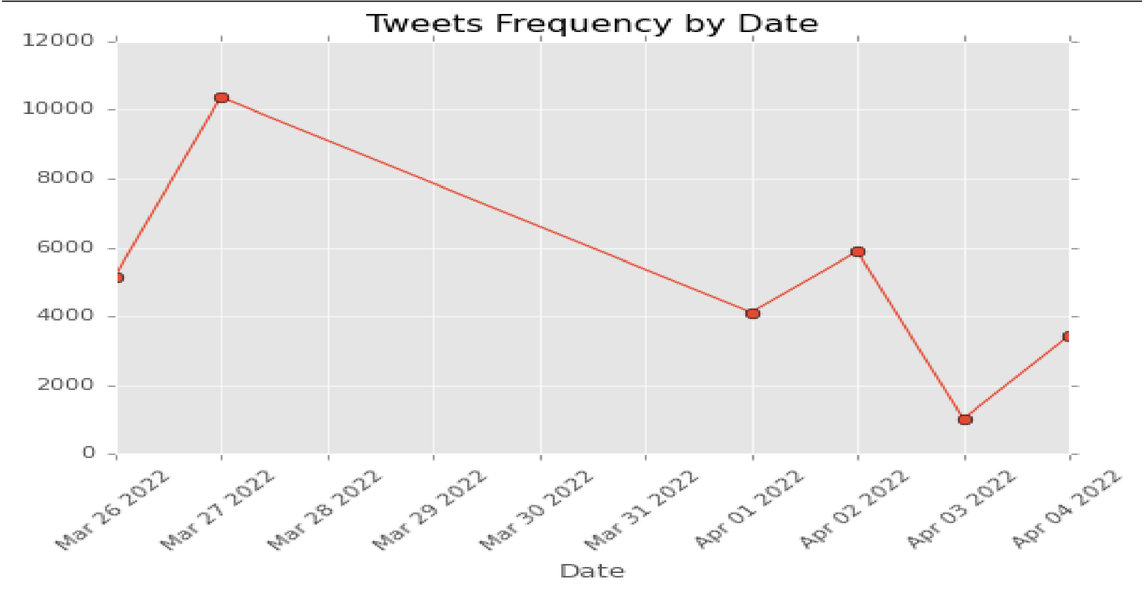
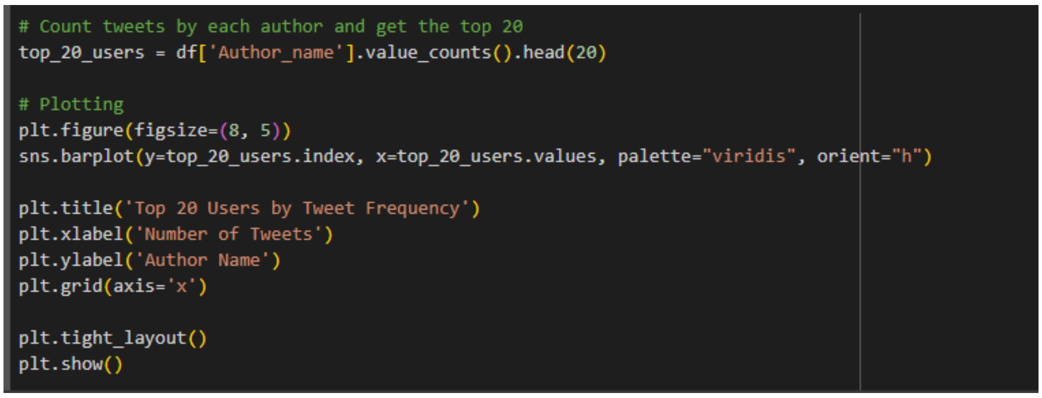


Fig – Plotting of Tweets Frequency by Date

6. Top 20 User by Tweets Frequency



```
# Count tweets by each author and get the top 20
top_20_users = df['Author_name'].value_counts().head(20)

# Plotting
plt.figure(figsize=(8, 5))
sns.barplot(y=top_20_users.index, x=top_20_users.values, palette="viridis", orient="h")

plt.title('Top 20 Users by Tweet Frequency')
plt.xlabel('Number of Tweets')
plt.ylabel('Author Name')
plt.grid(axis='x')

plt.tight_layout()
plt.show()
```

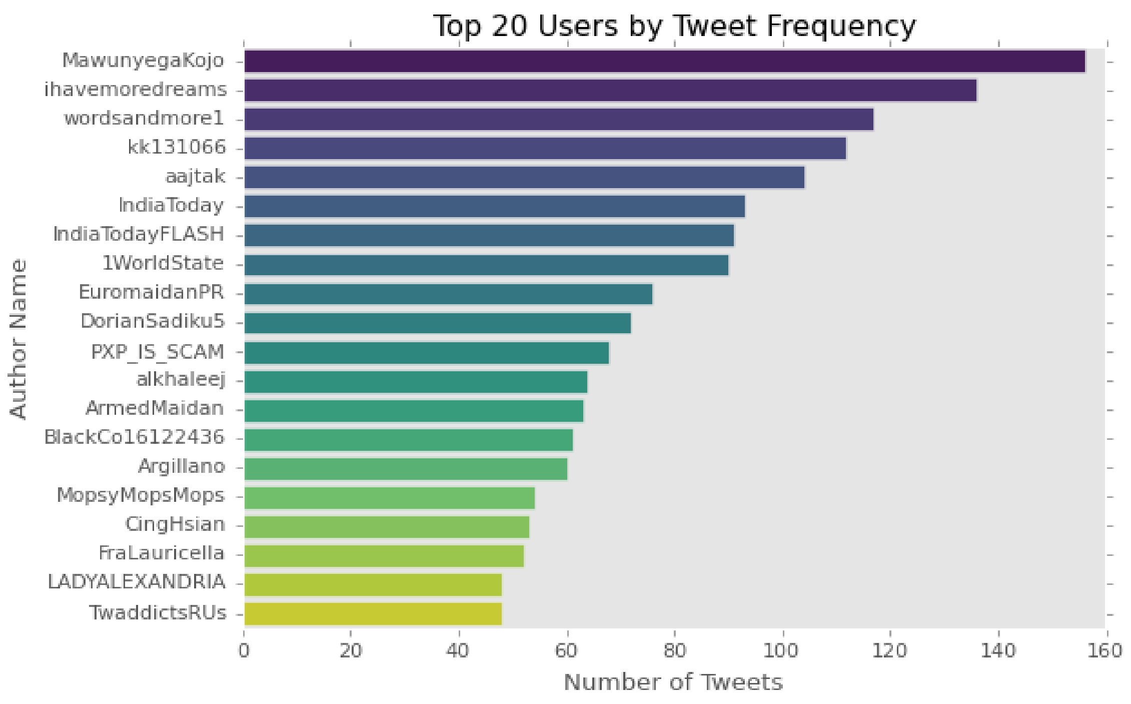
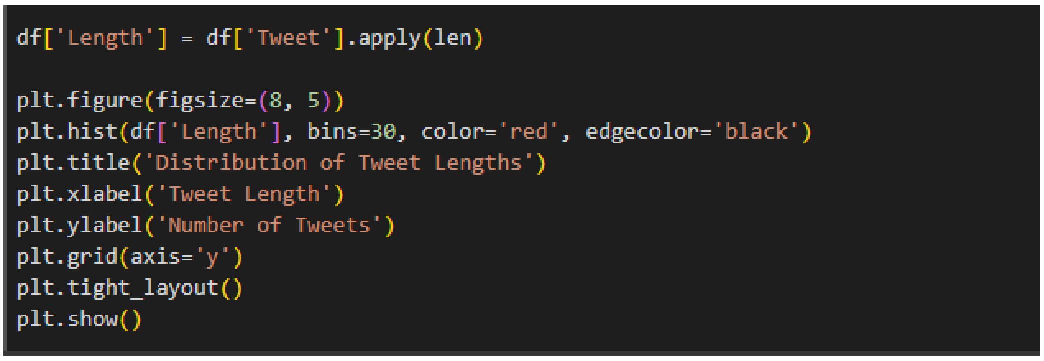


Fig – Plotting Top 20 User by Tweets Frequency

7. Distribution of Tweet Lengths



```
df['Length'] = df['Tweet'].apply(len)

plt.figure(figsize=(8, 5))
plt.hist(df['Length'], bins=30, color='red', edgecolor='black')
plt.title('Distribution of Tweet Lengths')
plt.xlabel('Tweet Length')
plt.ylabel('Number of Tweets')
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```

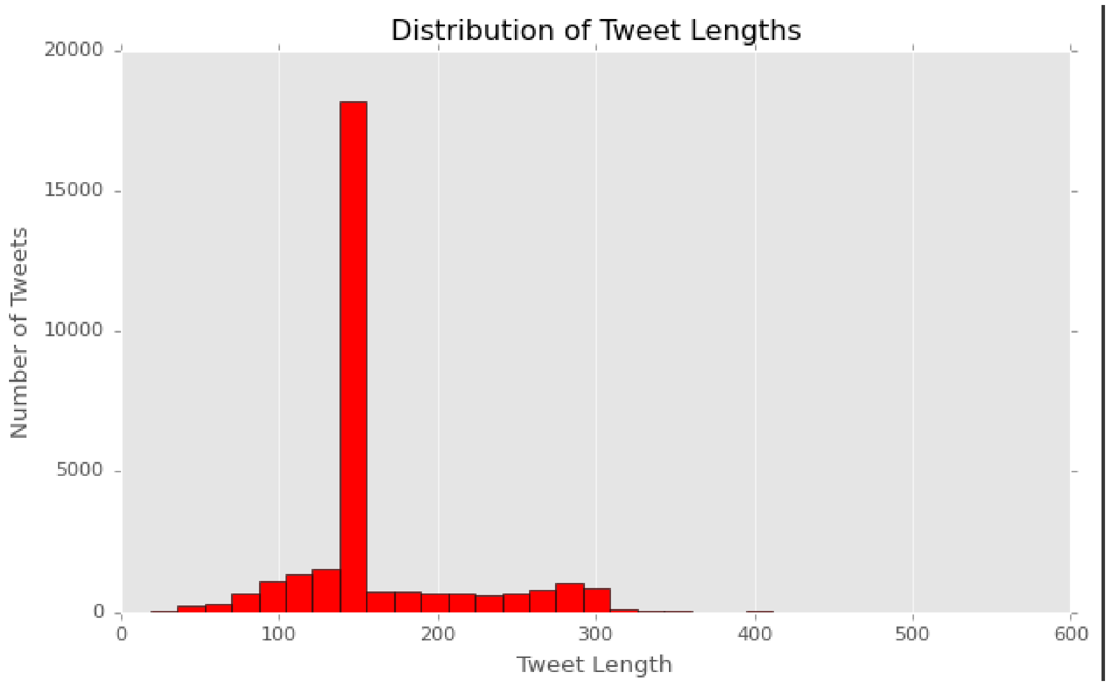


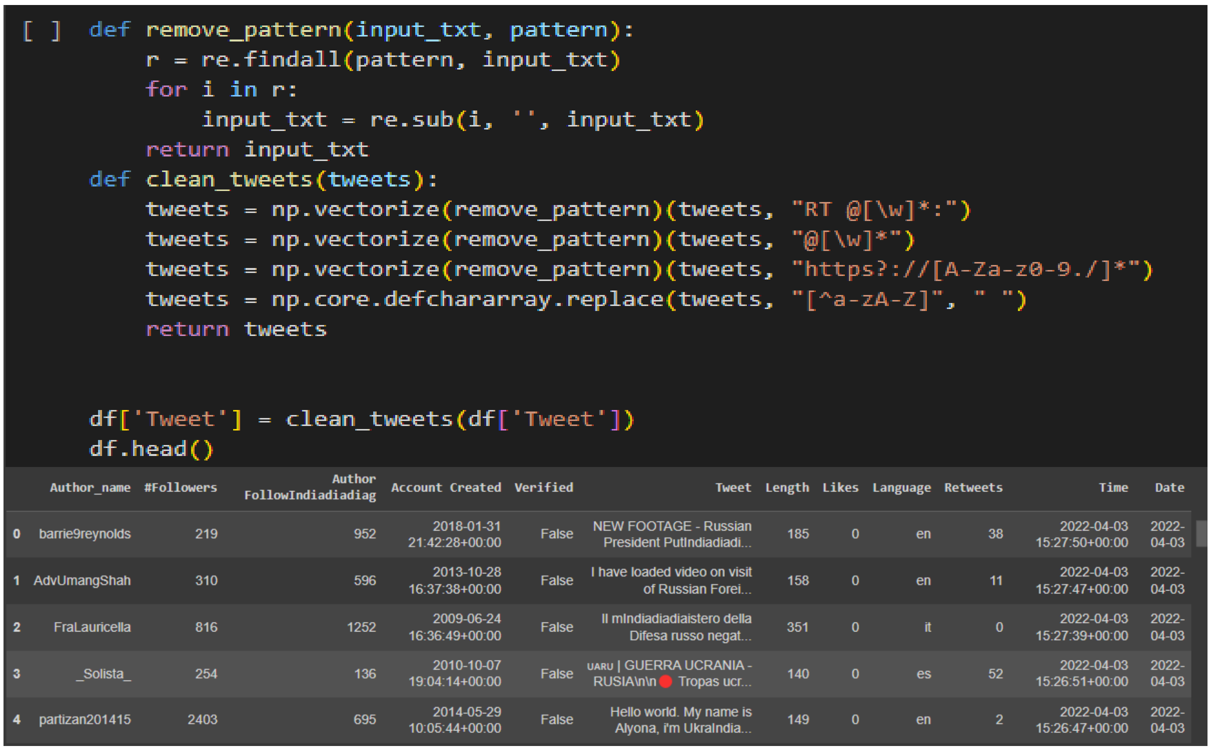
Fig - Plotting Distribution of Tweet Lengths

A. Dataset Preprocessing

Data Preprocessing is a crucial step in the data analysis and machine learning pipeline. It involves cleaning, transforming, and organizing raw data into a format suitable for analysis and modelling.

This process would eliminate any special or garbage characters from the tweets. Removing urls from tweet and also links in tweets.During the cleaning process, all unique characters, digits, and single letter words are removed. We had a clean dataset after these operations, with no unique characters, digits, or single- letter words. To tokenize the dataset, NLTK tokenizers are employed.

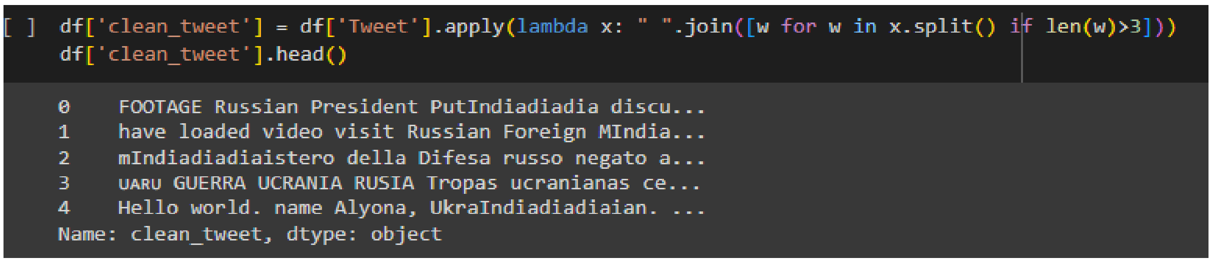
1.) Cleaning Tweets and Remove Patterns



```
[ ]  def remove_pattern(input_txt, pattern):
         r = re.findall(pattern, input_txt)
         for i in r:
             input_txt = re.sub(i, '', input_txt)
         return input_txt
     def clean_tweets(tweets):
         tweets = np.vectorize(remove_pattern)(tweets, "RT @[\w]*:")
         tweets = np.vectorize(remove_pattern)(tweets, "@[\w]*")
         tweets = np.vectorize(remove_pattern)(tweets, "https?://[A-Za-z0-9./]*")
         tweets = np.core.defchararray.replace(tweets, "[^a-zA-Z]", " ")
         return tweets


     df['Tweet'] = clean_tweets(df['Tweet'])
     df.head()
```

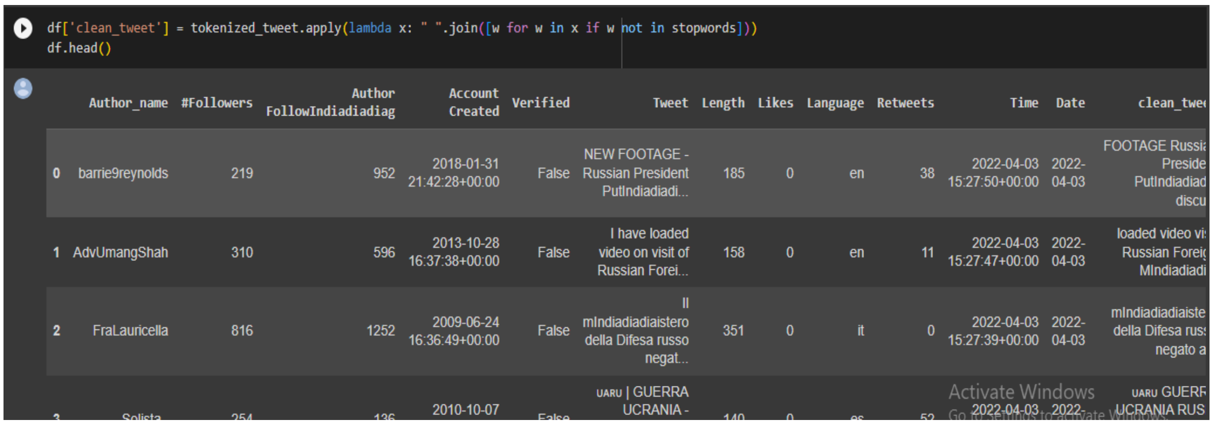
	Author_name	#Followers	Author FollowIndiadiadiag	Account Created	Verified	Tweet	Length	Likes	Language	Retweets	Time	Date
0	barrie9reynolds	219	952	2018-01-31 21:42:28+00:00	False	NEW FOOTAGE - Russian President PutIndiadiadi...	185	0	en	38	2022-04-03 15:27:50+00:00	2022-04-03
1	AdvUmangShah	310	596	2013-10-28 16:37:38+00:00	False	I have loaded video on visit of Russian Forei...	158	0	en	11	2022-04-03 15:27:47+00:00	2022-04-03
2	FraLauricella	816	1252	2009-06-24 16:36:49+00:00	False	Il mIndiadiadiaistero della Difesa russo negat...	351	0	it	0	2022-04-03 15:27:39+00:00	2022-04-03
3	_Solista_	254	136	2010-10-07 19:04:14+00:00	False	UARU | GUERRA UCRANIA - RUSIA\n\n 🔴 Tropas ucr...	140	0	es	52	2022-04-03 15:26:51+00:00	2022-04-03
4	partizan201415	2403	695	2014-05-29 10:05:44+00:00	False	Hello world. My name is Alyona, i'm UkraIndia...	149	0	en	2	2022-04-03 15:26:47+00:00	2022-04-03

2.) Remove Short Words



```
[ ]  df['clean_tweet'] = df['Tweet'].apply(lambda x: " ".join([w for w in x.split() if len(w)>3]))
     df['clean_tweet'].head()

     0    FOOTAGE Russian President PutIndiadiadia discu...
     1    have loaded video visit Russian Foreign MIndia...
     2    mIndiadiadiaistero della Difesa russo negato a...
     3    UARU GUERRA UCRANIA RUSIA Tropas ucranianas ce...
     4    Hello world. name Alyona, UkraIndiadiadiaian. ...
     Name: clean_tweet, dtype: object
```

3.) Remove Stop Words

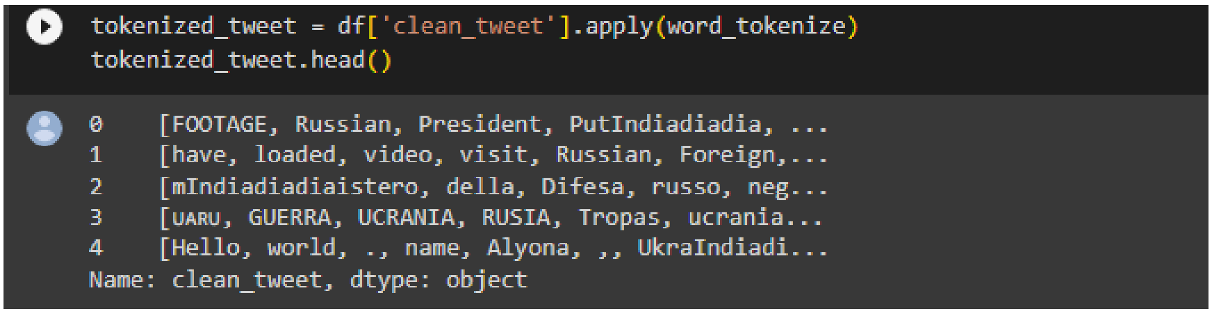


```
df['clean_tweet'] = tokenized_tweet.apply(lambda x: " ".join([w for w in x if w not in stopwords]))
df.head()
```

	Author_name	#Followers	Author FollowIndiadiadiag	Account Created	Verified	Tweet	Length	Likes	Language	Retweets	Time	Date	clean_twe
0	barrie9reynolds	219	952	2018-01-31 21:42:28+00:00	False	NEW FOOTAGE - Russian President PutIndiadiadi...	185	0	en	38	2022-04-03 15:27:50+00:00	2022-04-03	FOOTAGE Russi Preside PutIndiadiad discu
1	AdvUmangShah	310	596	2013-10-28 16:37:38+00:00	False	I have loaded video on visit of Russian Forei...	158	0	en	11	2022-04-03 15:27:47+00:00	2022-04-03	loaded video vi Russian Forei MIndiadiadi
2	FraLauricella	816	1252	2009-06-24 16:36:49+00:00	False	Il mIndiadiadiaistero della Difesa russo negat...	351	0	it	0	2022-04-03 15:27:39+00:00	2022-04-03	mIndiadiadiaiste della Difesa rus negato a
3	Solista	254	136	2010-10-07	False	UARU | GUERRA UCRANIA -	140	0	es	52	2022-04-03	2022-	UARU GUERR UCRANIA RUS

4.) Tokenization

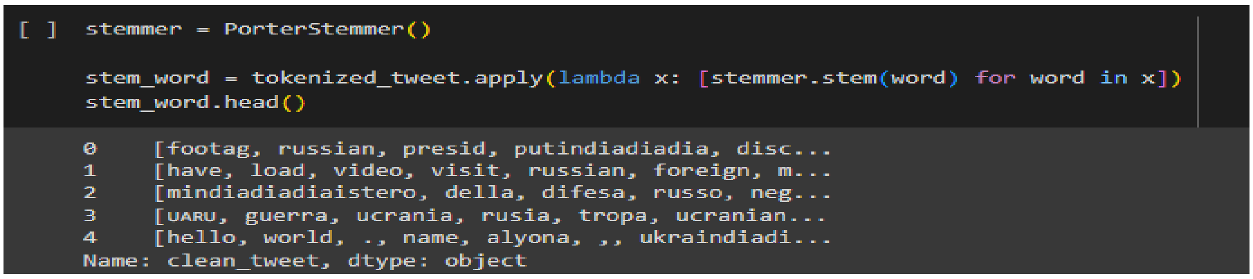
NLTK provides the `word_tokenize` and `sent_tokenize` functions. Tokenization is the process of breaking down text into smaller pieces, typically words or sentences. This is a fundamental step in NLP as it helps prepare the text for further processing.



```
tokenized_tweet = df['clean_tweet'].apply(word_tokenize)
tokenized_tweet.head()

0    [FOOTAGE, Russian, President, PutIndiadiadia, ...
1    [have, loaded, video, visit, Russian, Foreign,...
2    [mIndiadiadiaistero, della, Difesa, russo, neg...
3    [UARU, GUERRA, UCRANIA, RUSIA, Tropas, ucrania...
4    [Hello, world, ., name, Alyona, ,, UkraIndiadi...
Name: clean_tweet, dtype: object
```

5.) Stemming

Using tools like `PorterStemmer` and `WordNetLemmatizer`, NLTK can reduce words to their root form. This can help in consolidating words with similar meanings and improving analysis accuracy.



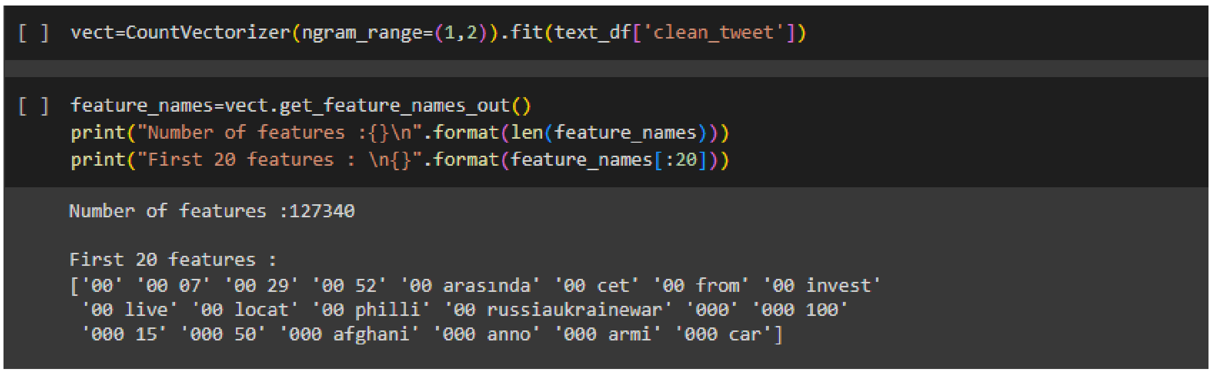
```
[ ]  stemmer = PorterStemmer()

     stem_word = tokenized_tweet.apply(lambda x: [stemmer.stem(word) for word in x])
     stem_word.head()

     0    [footag, russian, presid, putindiadiadia, disc...
     1    [have, load, video, visit, russian, foreign, m...
     2    [mindiadiadiaistero, della, difesa, russo, neg...
     3    [UARU, guerra, ucrania, rusia, tropa, ucranian...
     4    [hello, world, ., name, alyona, ,, ukraindiadi...
     Name: clean_tweet, dtype: object
```

B. Feature Extraction

CountVectorizer is a popular feature extraction technique used in machine learning for converting text data into numerical representations. CountVectorizer plays a vital role in transforming text data into numerical representations suitable for machine learning algorithms. It is a fundamental step in natural language processing tasks such as sentiment analysis, document classification, and topic modelling.

One of the key advantages of using CountVectorizer is that it captures both the presence and frequency of words, providing valuable information about the importance of different terms within documents. This allows machine learning algorithms to understand textual data and make predictions based on it.

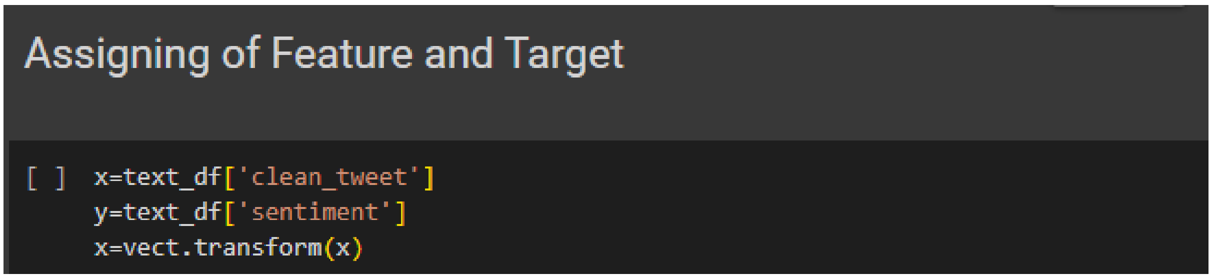
CountVectorizer from scikit-learn in Python to transform text data. p initializes a CountVectorizer with an n-gram range of (1, 2). This means that it will generate both single words (unigrams) and pairs of consecutive words (bigrams) as features for your text data in the code we retrieve the feature names from the fitted CountVectorizer object (vect) and prints the number of features as well as the first 20 feature names.



```
[ ]  vect=CountVectorizer(ngram_range=(1,2)).fit(text_df['clean_tweet'])
```

```
[ ]  feature_names=vect.get_feature_names_out()
     print("Number of features :{}\n".format(len(feature_names)))
     print("First 20 features : \n{}".format(feature_names[:20]))
```

```
Number of features :127340

First 20 features :
['00' '00 07' '00 29' '00 52' '00 arasında' '00 cet' '00 from' '00 invest'
 '00 live' '00 locat' '00 philli' '00 russiaukrainewar' '000' '000 100'
 '000 15' '000 50' '000 afghani' '000 anno' '000 armi' '000 car']
```



Assigning of Feature and Target

```
[ ]  x=text_df['clean_tweet']
     y=text_df['sentiment']
     x=vect.transform(x)
```

C. Model Used

1.) Logistic Regression

Logistic regression is a statistical method commonly used for predicting a numerical outcome based on one or more predictor variables. While it's more often used for regression tasks where the outcome is a continuous numerical value, it can also be adapted for sentiment analysis in a binary or ordinal classification setting, where the goal is to predict sentiment scores or labels based on text features from Twitter data.

Model Training:

Fit a linear regression model to the training data. In this context, you treat it as a classification problem by predicting sentiment scores or probabilities. Each tweet's sentiment score can be interpreted as the likelihood or strength of a positive sentiment.

Model Evaluation:

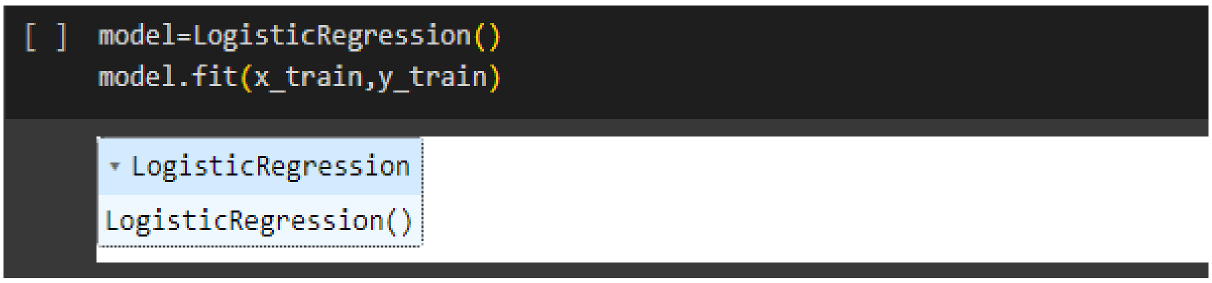
Use a suitable evaluation metric to assess the performance of your linear regression model. Common metrics for sentiment analysis include accuracy, precision, recall, F1-score, and ROC AUC.

Thresholding:

You might choose a threshold (e.g., 0.5) to convert the predicted sentiment scores into binary sentiment labels. For example, if the score is greater than or equal to 0.5, classify it as positive; otherwise, classify it as negative.

Prediction:

Apply the trained model to new, unlabelled Twitter data to predict sentiment labels or scores for those tweets.



```
[ ] model=LogisticRegression()
    model.fit(x_train,y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

2.) Linear SVC

Support Vector Machines (SVMs) and, in particular, the Linear Support Vector Classifier (LinearSVC) algorithm can be effectively used in Twitter sentiment analysis. LinearSVC is a variant of SVM designed for binary and multiclass classification tasks, making it suitable for sentiment analysis where the goal is to classify text into different sentiment categories, such as positive, negative, or neutral. Here's some information about using LinearSVC in Twitter sentiment analysis:

Data Splitting:

Split your dataset into training and testing sets. The training set is used to train the LinearSVC model, while the testing set is used to evaluate its performance.

Model Training:

Initialize and train the LinearSVC model using the training data.

Model Evaluation:

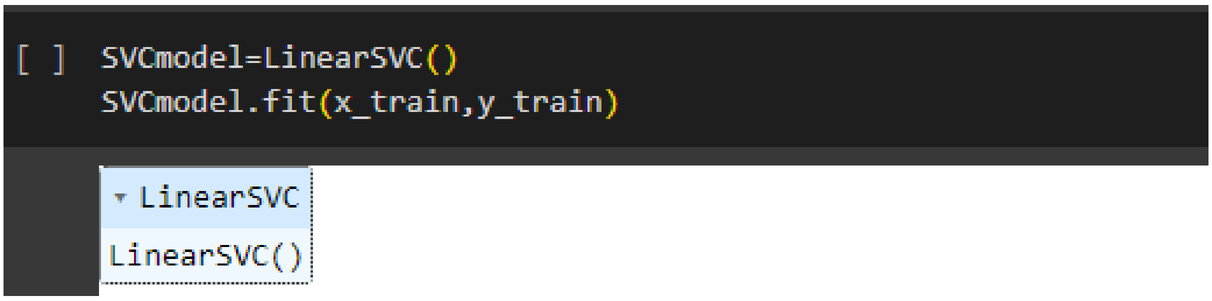
Use evaluation metrics to assess the model's performance on the testing data. Common metrics for sentiment analysis include accuracy, precision, recall, F1-score, and ROC AUC.

Thresholding (if needed):

Depending on your requirements, you may apply a threshold to the model's output scores to classify tweets into specific sentiment categories. For example, if the model outputs a probability score, you can set a threshold (e.g., 0.5) to determine whether a tweet is positive or negative.

Predictions:

Apply the trained LinearSVC model to new, unlabeled Twitter data to predict sentiment labels or scores for those tweets.



```
[ ]  SVCmodel=LinearSVC()
     SVCmodel.fit(x_train,y_train)
```

```
▾ LinearSVC
LinearSVC()
```

3.) Random Forest

The Random Forest algorithm is an ensemble of decision trees. It works by training multiple decision trees on different subsets of the data and then combining their predictions to make a final classification. Here's how it works:

Bootstrapping:

Randomly select subsets of the dataset with replacement. Each subset is used to train a decision tree. This introduces randomness and diversity into the model.

Feature Randomization:

At each split in the decision tree, only a random subset of features is considered. This prevents individual trees from becoming too specialized and overfitting the data.

Voting:

When making predictions, each decision tree in the forest provides a classification. The final prediction is determined by majority voting (for classification tasks) or averaging (for regression tasks) among the individual tree predictions.

Training the Random Forest:

Once the Random Forest is set up, can train it using the preprocessed and feature-extracted data. The algorithm will create a set of decision trees, each optimized to capture different patterns in the data.

Hyperparameter Tuning:

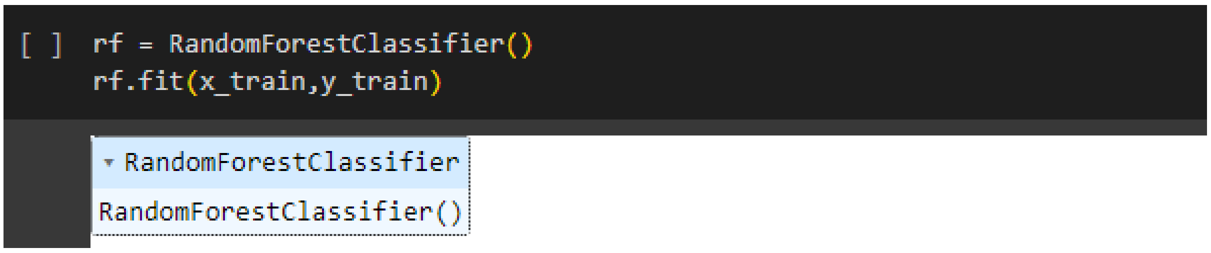
Random Forests have hyperparameters that can tune, such as the number of trees in the forest, the maximum depth of each tree, and the minimum number of samples required to split a node. Hyperparameter tuning is typically done using techniques like cross-validation to find the best configuration for your dataset.

Model Evaluation:

After training, evaluate the Random Forest model using a separate testing dataset. Common evaluation metrics for sentiment analysis include accuracy, precision, recall, F1-score, and confusion matrix.

Prediction:

Once Random Forest model is trained and evaluated, you can use it to predict the sentiment of new, unseen tweets.



```
[ ]  rf = RandomForestClassifier()
     rf.fit(x_train,y_train)
```

```
▾ RandomForestClassifier
RandomForestClassifier()
```

4.) Deep Learning

In deep learning, the Sequential model is a fundamental building block used for creating neural networks, and LSTM (Long Short-Term Memory) and Dense are types of layers commonly used in these networks.

1. **Sequential Model:**
   The Sequential model is a linear stack of layers used to create deep neural networks in Keras, a popular deep learning library.
   It's called "sequential" because you add layers to it in a sequential order, one after another.

2. **LSTM (Long Short-Term Memory):**
   LSTM is a type of recurrent neural network (RNN) layer used for handling sequences and time-series data.
   Unlike standard feedforward layers like Dense, LSTM layers have memory that allows them to capture long-term dependencies in sequences.
   LSTM layers are particularly useful for tasks like natural language processing (NLP) and speech recognition.

3. **Dense Layer:**
   The Dense layer is a fully connected layer where each neuron in the layer is connected to every neuron in the previous and subsequent layers. It's also known as a fully connected layer or a feedforward layer.
   The Dense layer performs a weighted sum of its inputs and applies an activation function to produce the output.
   In the example above, Dense(units=64) creates a layer with 64 neurons, and activation='relu' specifies the Rectified Linear Unit (ReLU) activation function.
   The output layer has a single neuron with a sigmoid activation function, which is commonly used for binary classification to produce a probability between 0 and 1.

V. Discussion and Results

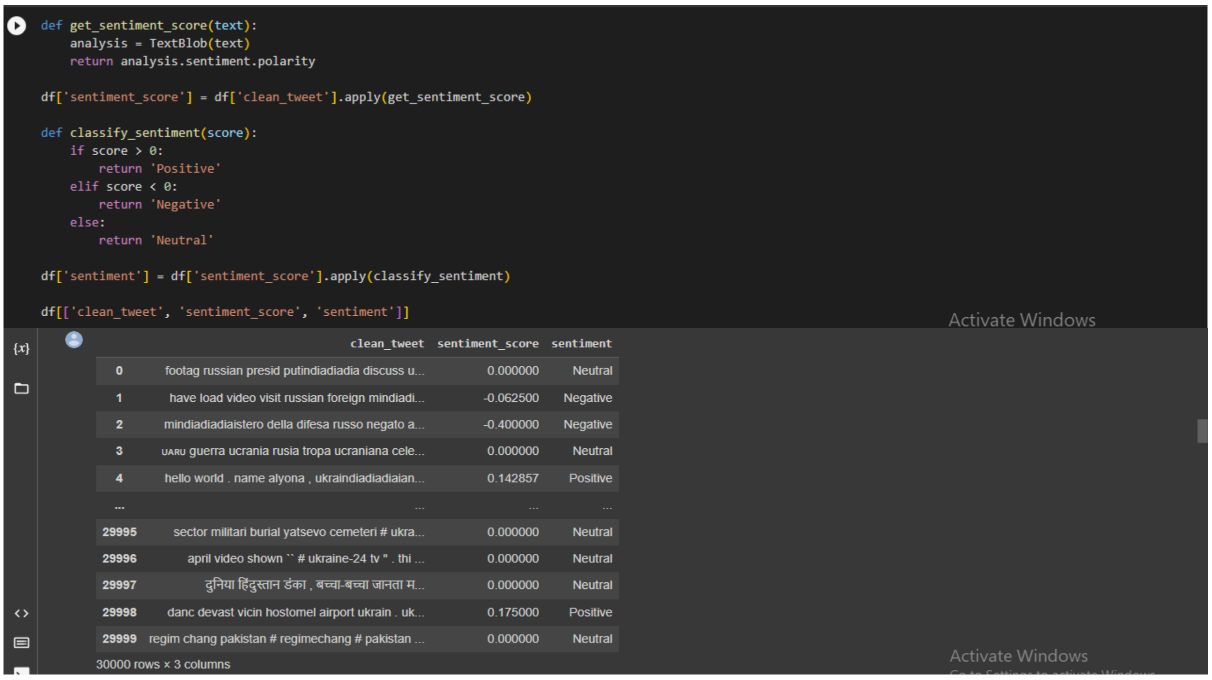
Opinion mining on Russia-Ukraine conflicts in social media" suggests analyzing social media data to extract public sentiment or opinions related to the Russia-Ukraine conflicts. To do this task, the Natural Language Toolkit (NLTK) can be a valuable tool.

A. Output Obtained

1.) Sentiment Score and Classify Tweets

A sentiment score is a numerical representation of the sentiment or emotional tone of a piece of text, often in the context of Natural Language Processing (NLP) and Sentiment Analysis. Sentiment analysis aims to determine whether a given text express positive, negative or neutral sentiment.

Sentiment scores are valuable for various applications, including understanding public opinion, brand monitoring and social media sentiment tracking.

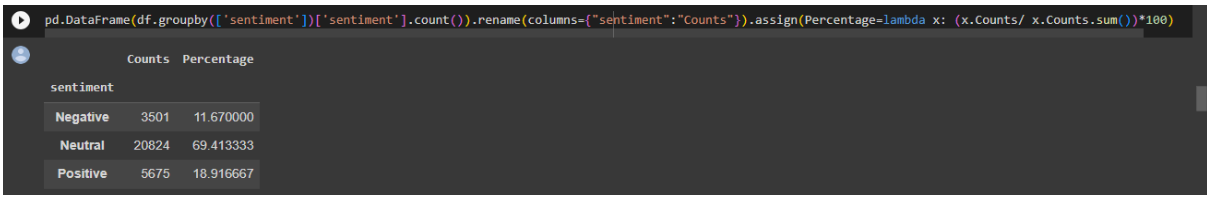


```python
def get_sentiment_score(text):
    analysis = TextBlob(text)
    return analysis.sentiment.polarity

df['sentiment_score'] = df['clean_tweet'].apply(get_sentiment_score)

def classify_sentiment(score):
    if score > 0:
        return 'Positive'
    elif score < 0:
        return 'Negative'
    else:
        return 'Neutral'

df['sentiment'] = df['sentiment_score'].apply(classify_sentiment)

df[['clean_tweet', 'sentiment_score', 'sentiment']]
```

	clean_tweet	sentiment_score	sentiment
0	footag russian presid putindiadiadia discuss u...	0.000000	Neutral
1	have load video visit russian foreign mindiadi...	-0.062500	Negative
2	mindiadiadiaistero della difesa russo negato a...	-0.400000	Negative
3	UARU guerra ucrania rusia tropa ucraniana cele...	0.000000	Neutral
4	hello world . name alyona , ukraindiadiadiaian...	0.142857	Positive
...	...	...	...
29995	sector militari burial yatsevo cemeteri # ukra...	0.000000	Neutral
29996	april video shown `` # ukraine-24 tv " . thi ...	0.000000	Neutral
29997	दुनिया हिंदुस्तान डंका , बच्चा-बच्चा जानता म...	0.000000	Neutral
29998	danc devast vicin hostomel airport ukrain . uk...	0.175000	Positive
29999	regim chang pakistan # regimechang # pakistan ...	0.000000	Neutral

30000 rows × 3 columns

- Counts and Percentage of Tweets by Sentiment



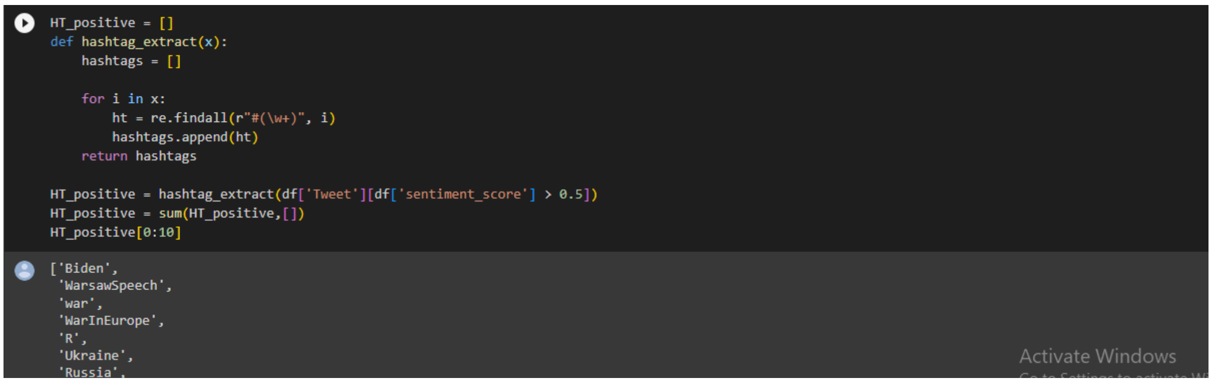
```python
pd.DataFrame(df.groupby(['sentiment'])['sentiment'].count()).rename(columns={"sentiment":"Counts"}).assign(Percentage=lambda x: (x.Counts/ x.Counts.sum())*100)
```

sentiment	Counts	Percentage
Negative	3501	11.670000
Neutral	20824	69.413333
Positive	5675	18.916667

6.) Extracting Hashtags

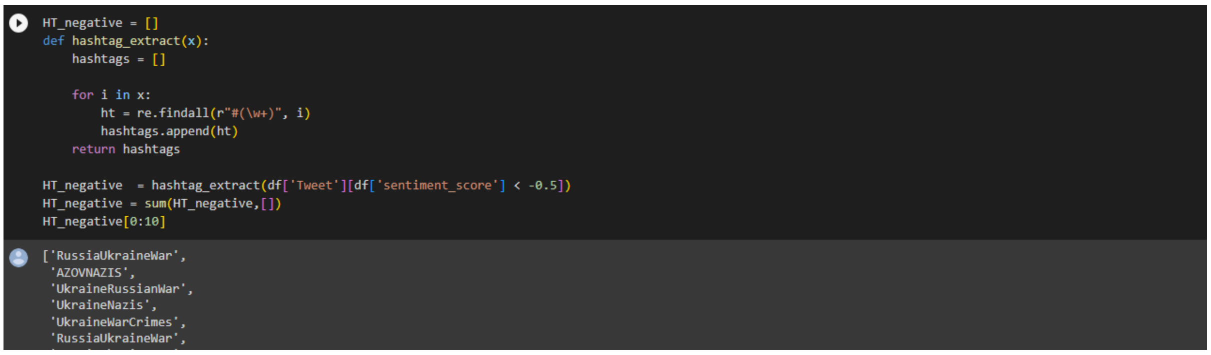
Hashtags play a significant role in categorizing social media content related to this war. Positive hashtags such as #peacefulsolution or #supportUkraine reflect sentiments favouring peace and solidarity with Ukraine. On the other hand, negative hashtags like #Russianaggression or #warcrimes highlight criticism towards Russia's actions in Ukraine. Neutral hashtags such as #RussiaUkraineWar or #conflict provide a more objective perspective without expressing any particular sentiment.

A.) Positive Hashtag and Negative Hashtags



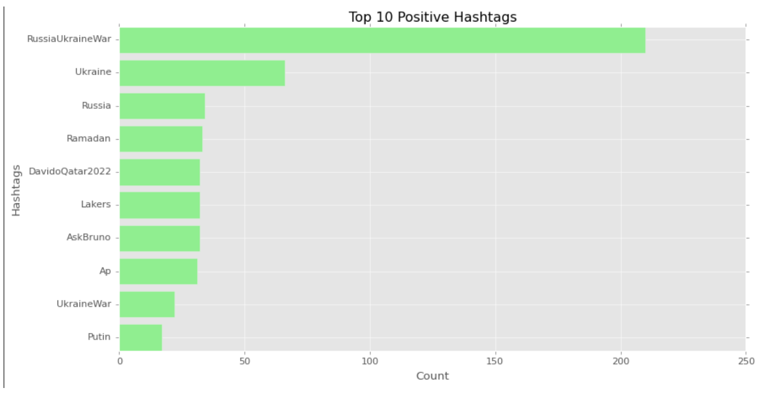
```python
HT_positive = []
def hashtag_extract(x):
    hashtags = []

    for i in x:
        ht = re.findall(r"#(\w+)", i)
        hashtags.append(ht)
    return hashtags

HT_positive = hashtag_extract(df['Tweet'][df['sentiment_score'] > 0.5])
HT_positive = sum(HT_positive,[])
HT_positive[0:10]
```

```
['Biden',
 'WarsawSpeech',
 'war',
 'WarInEurope',
 'R',
 'Ukraine',
 'Russia',
```



```python
HT_negative = []
def hashtag_extract(x):
    hashtags = []

    for i in x:
        ht = re.findall(r"#(\w+)", i)
        hashtags.append(ht)
    return hashtags

HT_negative  = hashtag_extract(df['Tweet'][df['sentiment_score'] < -0.5])
HT_negative = sum(HT_negative,[])
HT_negative[0:10]
```

```
['RussiaUkraineWar',
 'AZOVNAZIS',
 'UkraineRussianWar',
 'UkraineNazis',
 'UkraineWarCrimes',
 'RussiaUkraineWar',
```



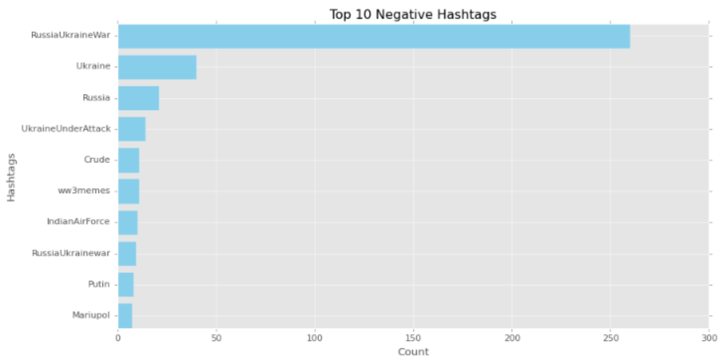
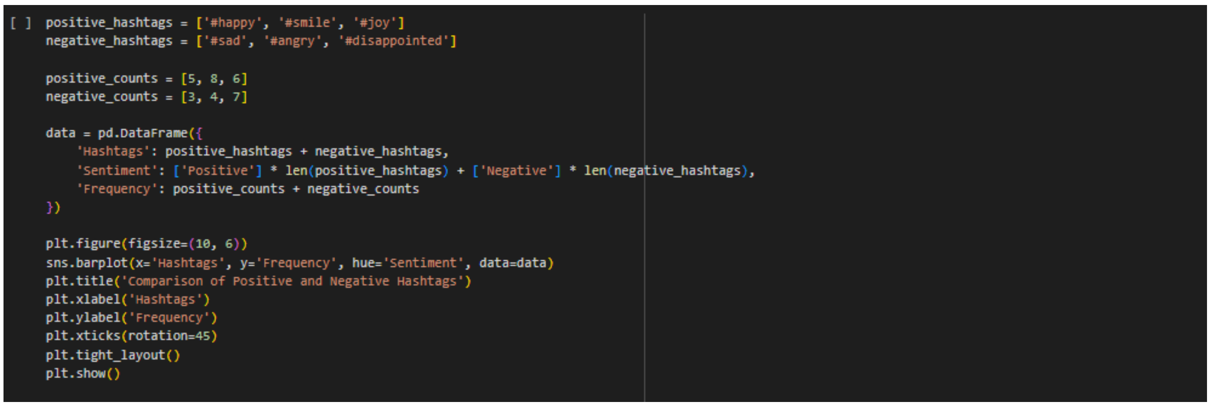
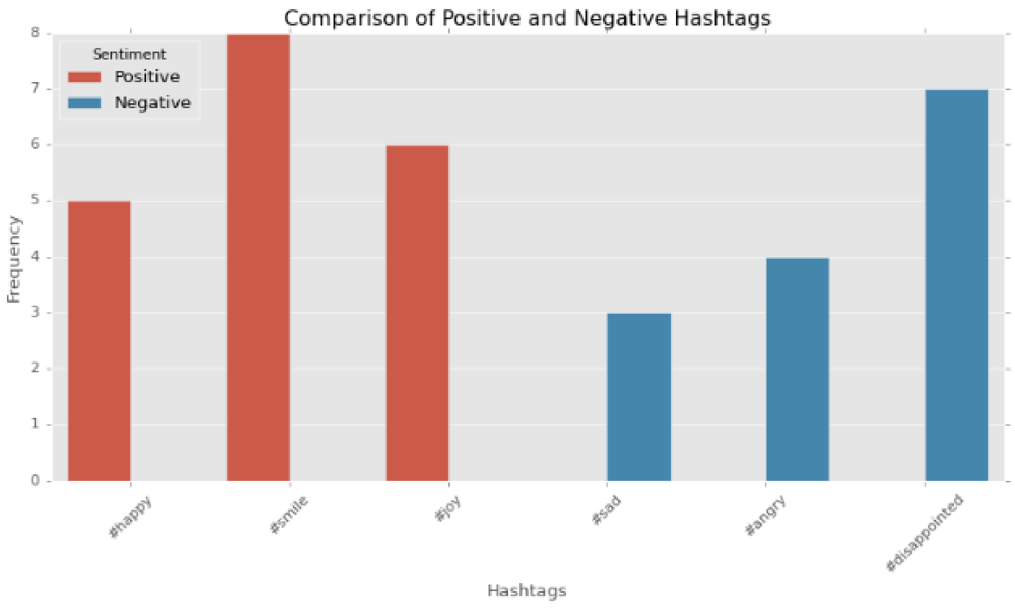


Fig. Top 10 Positive Hashtags #

Fig. Top 10 Negative Hashtags #

B.) Comparison Positive and Negative Hashtags


```
[ ]  positive_hashtags = ['#happy', '#smile', '#joy']
     negative_hashtags = ['#sad', '#angry', '#disappointed']

     positive_counts = [5, 8, 6]
     negative_counts = [3, 4, 7]

     data = pd.DataFrame({
         'Hashtags': positive_hashtags + negative_hashtags,
         'Sentiment': ['Positive'] * len(positive_hashtags) + ['Negative'] * len(negative_hashtags),
         'Frequency': positive_counts + negative_counts
     })

     plt.figure(figsize=(10, 6))
     sns.barplot(x='Hashtags', y='Frequency', hue='Sentiment', data=data)
     plt.title('Comparison of Positive and Negative Hashtags')
     plt.xlabel('Hashtags')
     plt.ylabel('Frequency')
     plt.xticks(rotation=45)
     plt.tight_layout()
     plt.show()
```



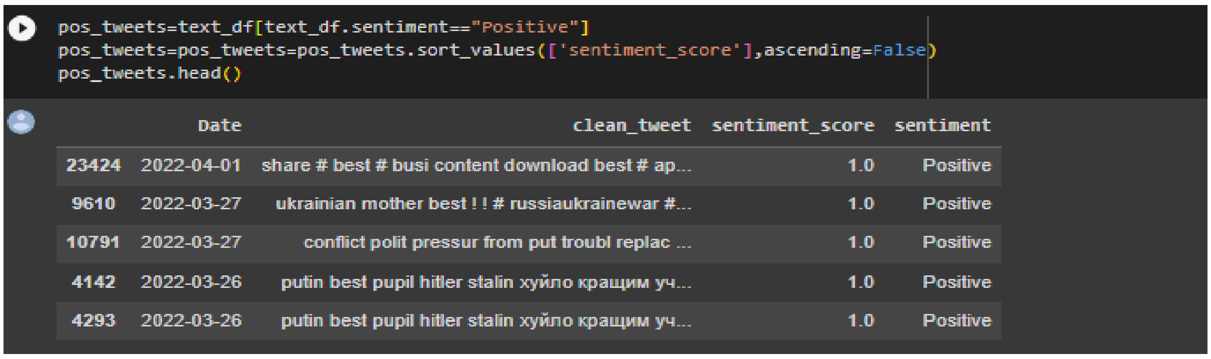
<u>Fig. Bar chart of Comparison of Positive and Negative Hashtags</u>

7.) Filtering Sentiments of Tweets (Positive, Negative and Neutral)

Filtering sentiment in tweets can be a challenging task due to the vast amount of data generated every second. Nevertheless, it is crucial to categorize these sentiments into positive, negative, or neutral to gauge public perception accurately. By employing natural language processing techniques and machine learning algorithms, sentiment analysis tools can effectively identify keywords and phrases that indicate positivity or negativity.

Ukraine-Russian war conflicts is essential for understanding public perceptions accurately. Through advanced technologies like natural language processing and machine learning algorithms, sentiment analysis tools can effectively categorize tweets as positive, negative, or neutral. This analysis provides valuable insights for policymakers and researchers alike when formulating strategies or interventions related to these conflicts.

A.) Filtering Positive Tweets and create Wordcloud on that tweets



```
pos_tweets=text_df[text_df.sentiment=="Positive"]
pos_tweets=pos_tweets=pos_tweets.sort_values(['sentiment_score'],ascending=False)
pos_tweets.head()
```

```
             Date                                   clean_tweet  sentiment_score  sentiment
23424  2022-04-01  share # best # busi content download best # ap...              1.0   Positive
 9610  2022-03-27   ukrainian mother best ! ! # russiaukrainewar #...             1.0   Positive
10791  2022-03-27       conflict polit pressur from put troubl replac ...             1.0   Positive
 4142  2022-03-26    putin best pupil hitler stalin хуйло кращим уч...             1.0   Positive
 4293  2022-03-26    putin best pupil hitler stalin хуйло кращим уч...             1.0   Positive
```

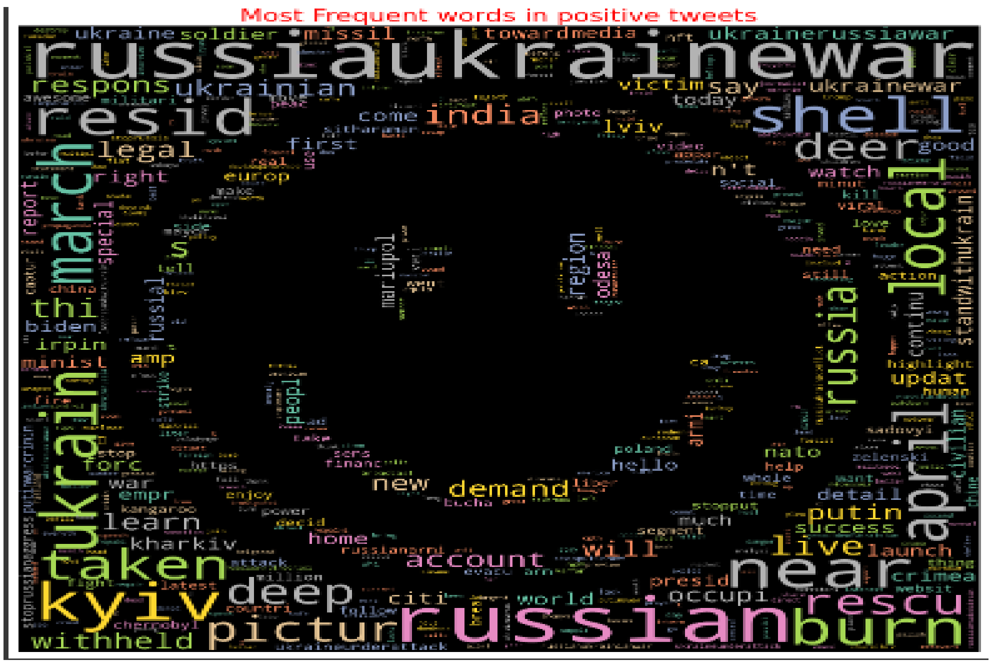
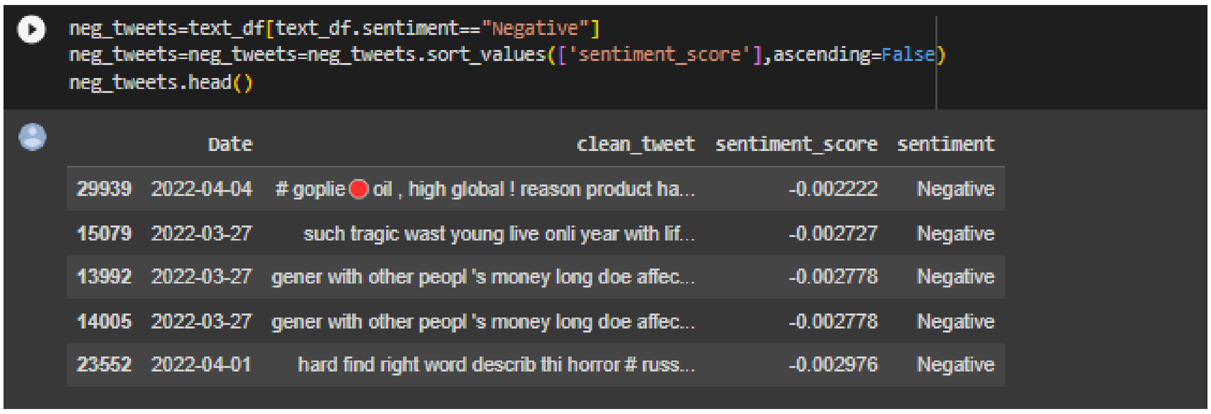


Fig. Wordcloud of Positive Tweets

B.) Filtering Negative Tweets and Create Wordcloud on that Tweets



```
neg_tweets=text_df[text_df.sentiment=="Negative"]
neg_tweets=neg_tweets.sort_values(['sentiment_score'],ascending=False)
neg_tweets.head()
```

```
             Date                                      clean_tweet  sentiment_score  sentiment
29939  2022-04-04   # goplie🔴 oil , high global ! reason product ha...        -0.002222   Negative
15079  2022-03-27      such tragic wast young live onli year with lif...        -0.002727   Negative
13992  2022-03-27   gener with other peopl 's money long doe affec...        -0.002778   Negative
14005  2022-03-27   gener with other peopl 's money long doe affec...        -0.002778   Negative
23552  2022-04-01     hard find right word describ thi horror # russ...        -0.002976   Negative
```

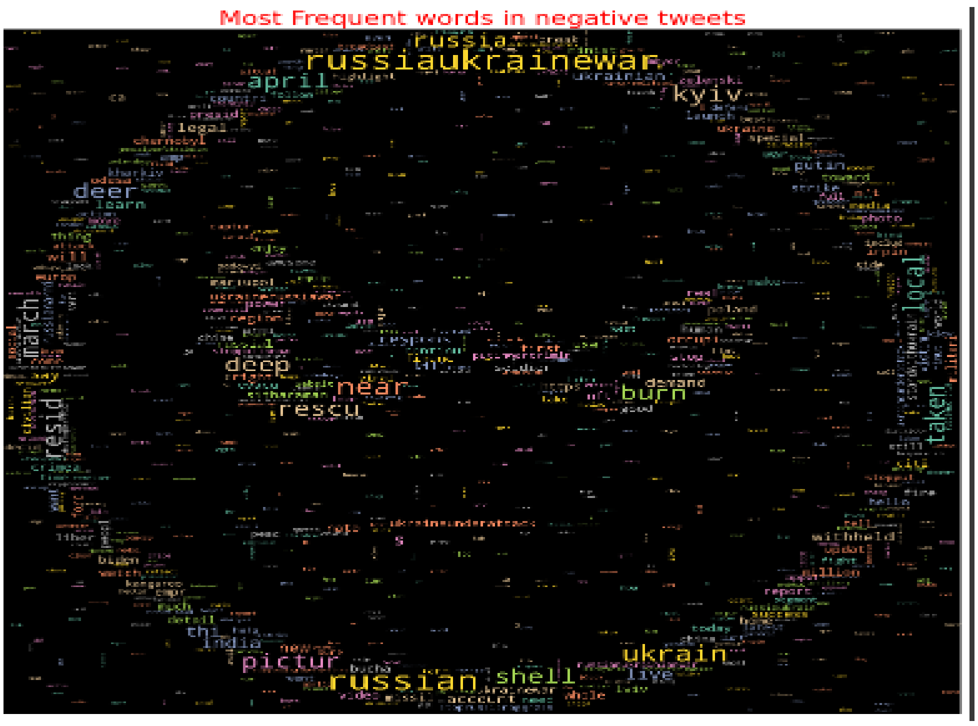
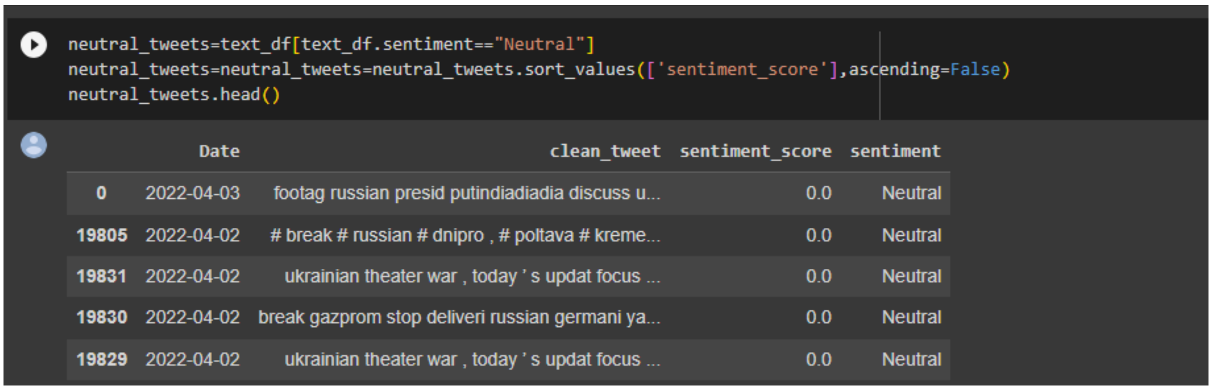


Fig. – Wordcloud of Negative Tweets

C.) Filtering Neutral Tweets and Create Wordcloud on that Tweets



```
neutral_tweets=text_df[text_df.sentiment=="Neutral"]
neutral_tweets=neutral_tweets=neutral_tweets.sort_values(['sentiment_score'],ascending=False)
neutral_tweets.head()
```

	Date	clean_tweet	sentiment_score	sentiment
0	2022-04-03	footag russian presid putindiadiadia discuss u...	0.0	Neutral
19805	2022-04-02	# break # russian # dnipro , # poltava # kreme...	0.0	Neutral
19831	2022-04-02	ukrainian theater war , today ' s updat focus ...	0.0	Neutral
19830	2022-04-02	break gazprom stop deliveri russian germani ya...	0.0	Neutral
19829	2022-04-02	ukrainian theater war , today ' s updat focus ...	0.0	Neutral

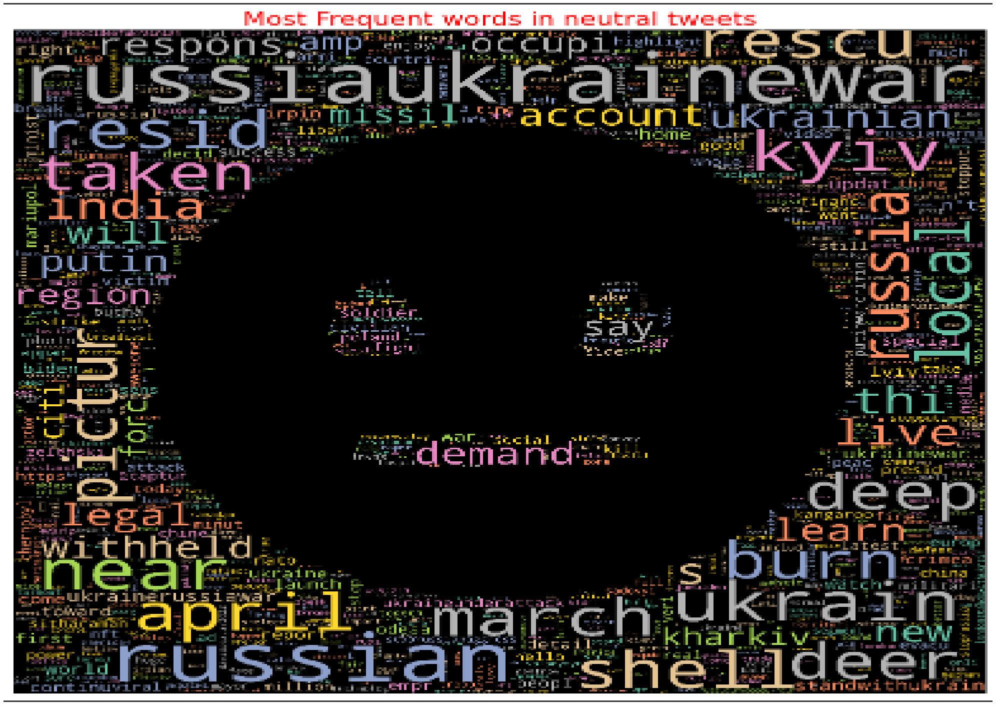
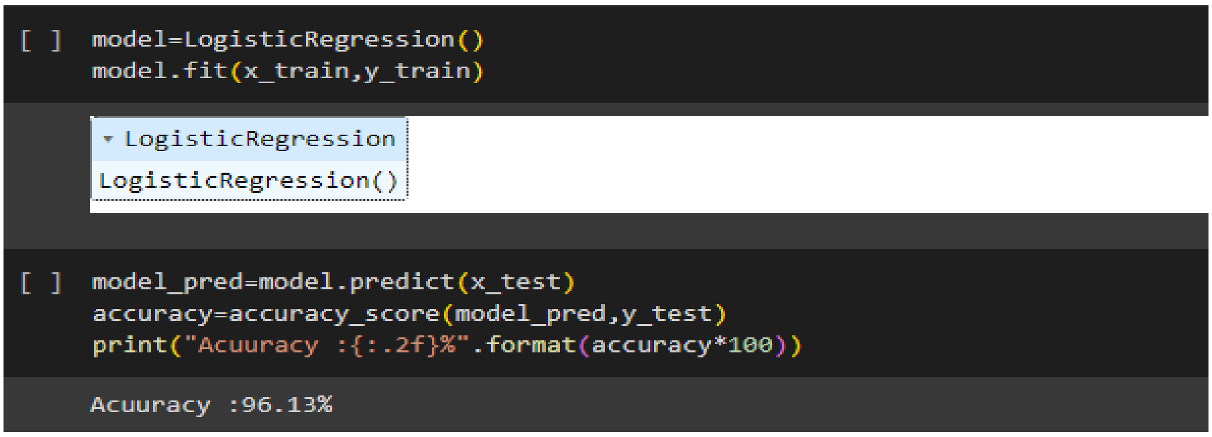


Fig – Wordcloud of Neutral tweets

B. Evaluation Measures Used

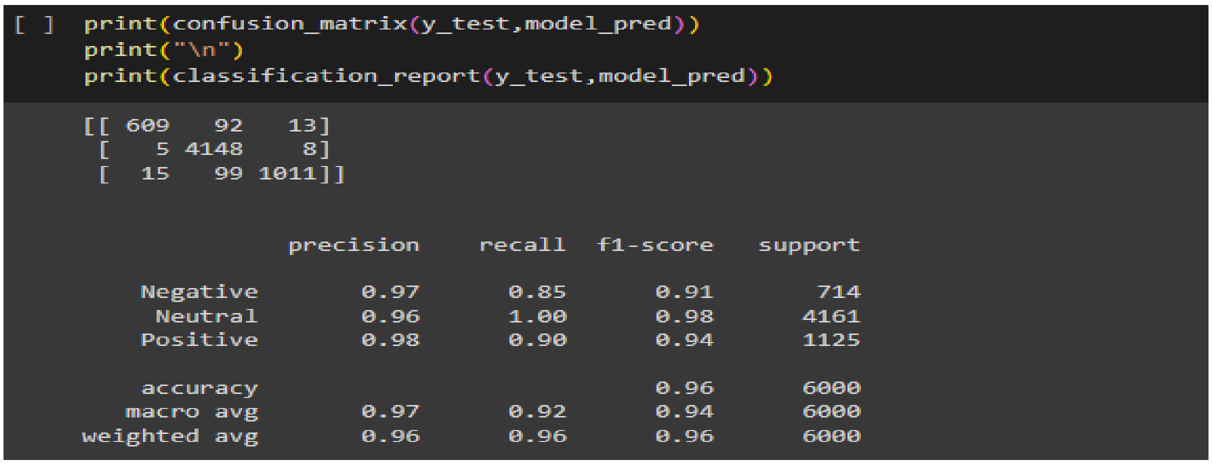
1. Logistic Regression

Logistic Regression is a simple and interpretable algorithm suitable for binary classification tasks. It can be extended to multiclass classification using techniques like one-vs-all (OvA) or softmax regression. Regularization techniques (L1 and L2 regularization) can also be applied to prevent overfitting.

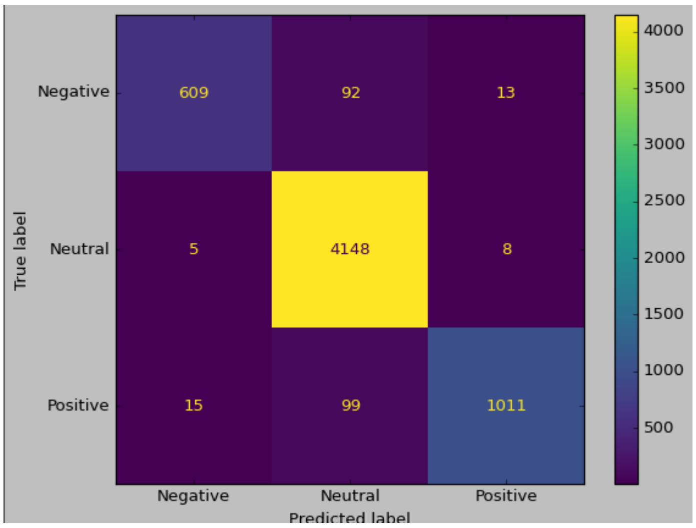


```
[ ] model=LogisticRegression()
    model.fit(x_train,y_train)

    ▾ LogisticRegression
    LogisticRegression()
```

```
[ ] model_pred=model.predict(x_test)
    accuracy=accuracy_score(model_pred,y_test)
    print("Acuuracy :{:.2f}%".format(accuracy*100))

    Acuuracy :96.13%
```

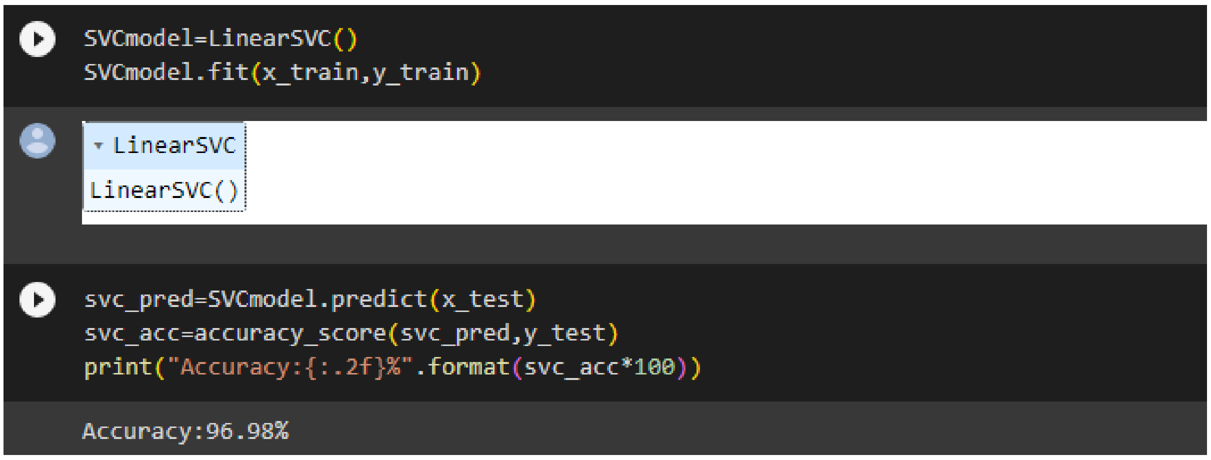


```
[ ] print(confusion_matrix(y_test,model_pred))
    print("\n")
    print(classification_report(y_test,model_pred))

    [[ 609   92   13]
     [   5 4148    8]
     [  15   99 1011]]


                  precision    recall  f1-score   support

        Negative       0.97      0.85      0.91       714
         Neutral       0.96      1.00      0.98      4161
        Positive       0.98      0.90      0.94      1125

        accuracy                           0.96      6000
       macro avg       0.97      0.92      0.94      6000
    weighted avg       0.96      0.96      0.96      6000
```



2. Linear SVC

Linear SVC is a powerful algorithm for linear classification problems and is particularly useful when you have high-dimensional data that can be separated by a linear decision boundary.

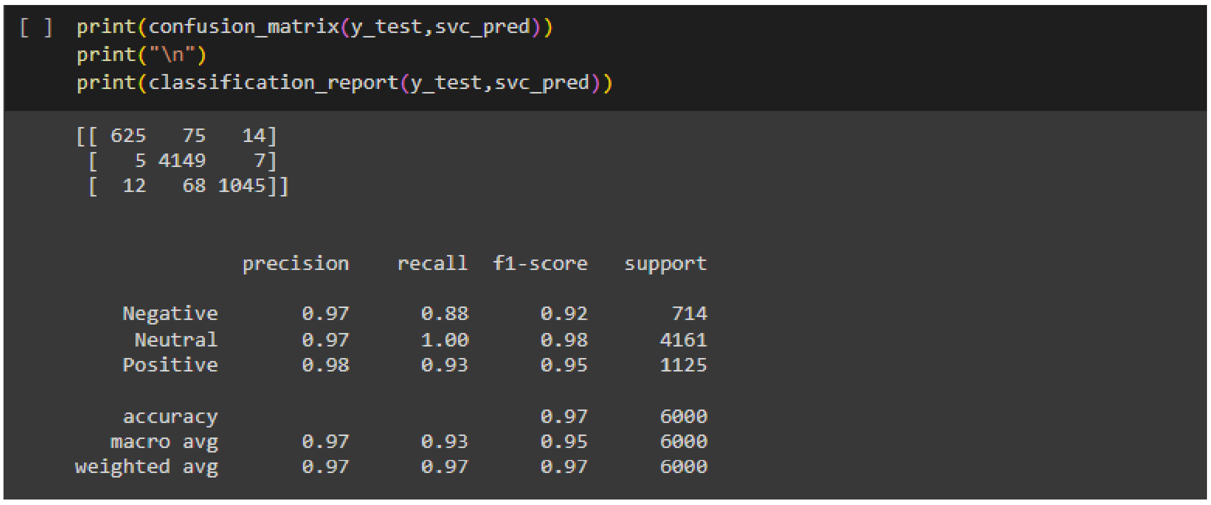


```
SVCmodel=LinearSVC()
SVCmodel.fit(x_train,y_train)
```

```
▾ LinearSVC
LinearSVC()
```

```
svc_pred=SVCmodel.predict(x_test)
svc_acc=accuracy_score(svc_pred,y_test)
print("Accuracy:{:.2f}%".format(svc_acc*100))
```

```
Accuracy:96.98%
```

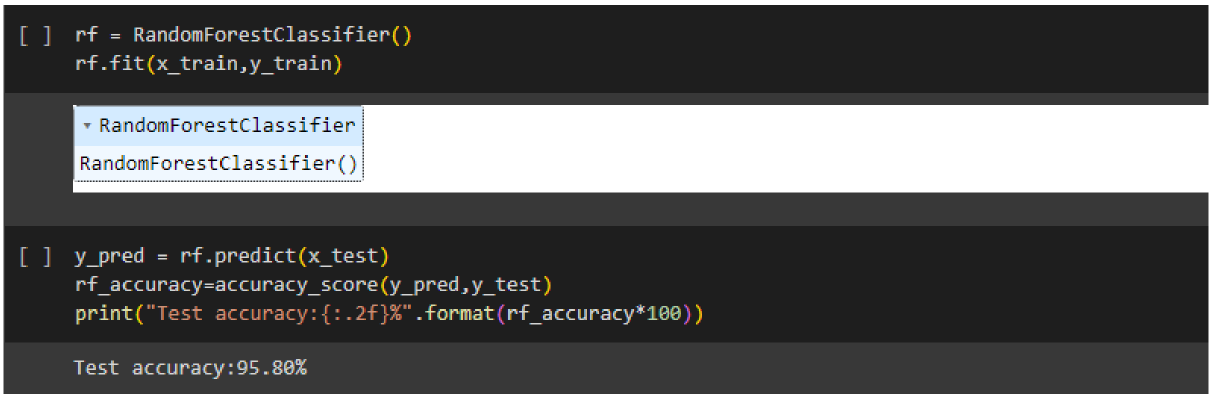


```
[ ]  print(confusion_matrix(y_test,svc_pred))
     print("\n")
     print(classification_report(y_test,svc_pred))
```

```
[[ 625   75   14]
 [   5 4149    7]
 [  12   68 1045]]


              precision    recall  f1-score   support

    Negative       0.97      0.88      0.92       714
     Neutral       0.97      1.00      0.98      4161
    Positive       0.98      0.93      0.95      1125

    accuracy                           0.97      6000
   macro avg       0.97      0.93      0.95      6000
weighted avg       0.97      0.97      0.97      6000
```

3. Random Forest

Random Forest is a powerful and versatile algorithm that is widely used in various applications, including classification, regression, feature selection, and outlier detection. It is known for its robustness and ease of use, making it a popular choice in machine learning projects.

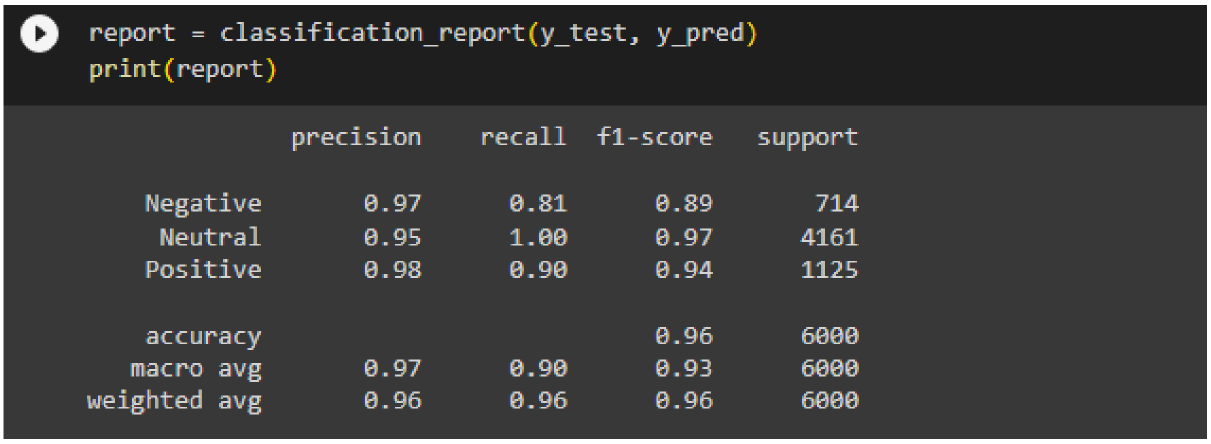


```
[ ] rf = RandomForestClassifier()
    rf.fit(x_train,y_train)
```

```
▾ RandomForestClassifier
RandomForestClassifier()
```

```
[ ] y_pred = rf.predict(x_test)
    rf_accuracy=accuracy_score(y_pred,y_test)
    print("Test accuracy:{:.2f}%".format(rf_accuracy*100))
```

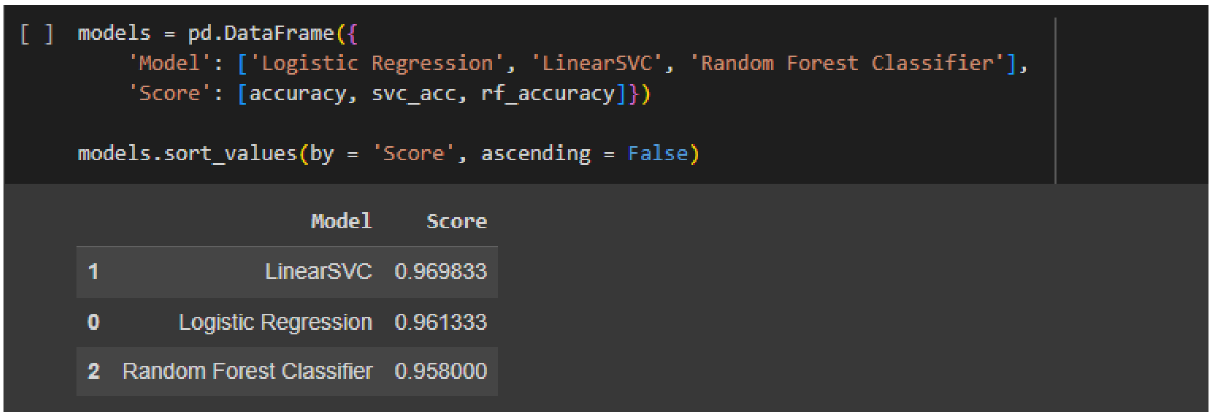
```
Test accuracy:95.80%
```



```
report = classification_report(y_test, y_pred)
print(report)
```

```
              precision    recall  f1-score   support

    Negative       0.97      0.81      0.89       714
     Neutral       0.95      1.00      0.97      4161
    Positive       0.98      0.90      0.94      1125

    accuracy                           0.96      6000
   macro avg       0.97      0.90      0.93      6000
weighted avg       0.96      0.96      0.96      6000
```

4. Comparing Accuracy Score in (Logistic Regression, Linear SVC, Random Forest)



```
[ ]  models = pd.DataFrame({
         'Model': ['Logistic Regression', 'LinearSVC', 'Random Forest Classifier'],
         'Score': [accuracy, svc_acc, rf_accuracy]})

     models.sort_values(by = 'Score', ascending = False)
```

```
                      Model     Score
1                 LinearSVC  0.969833
0       Logistic Regression  0.961333
2  Random Forest Classifier  0.958000
```

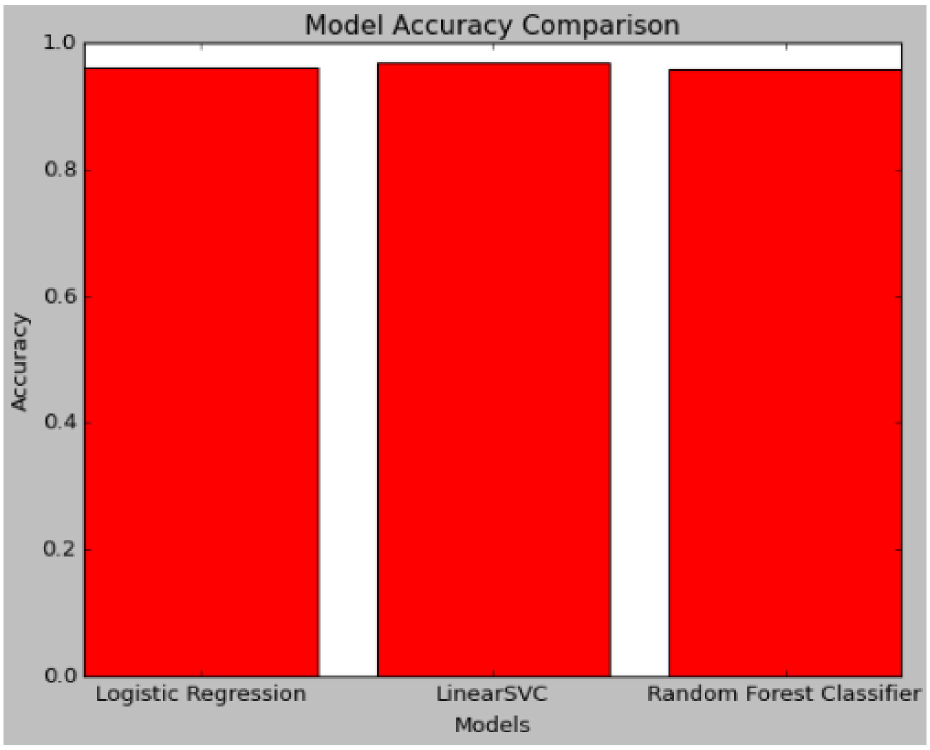


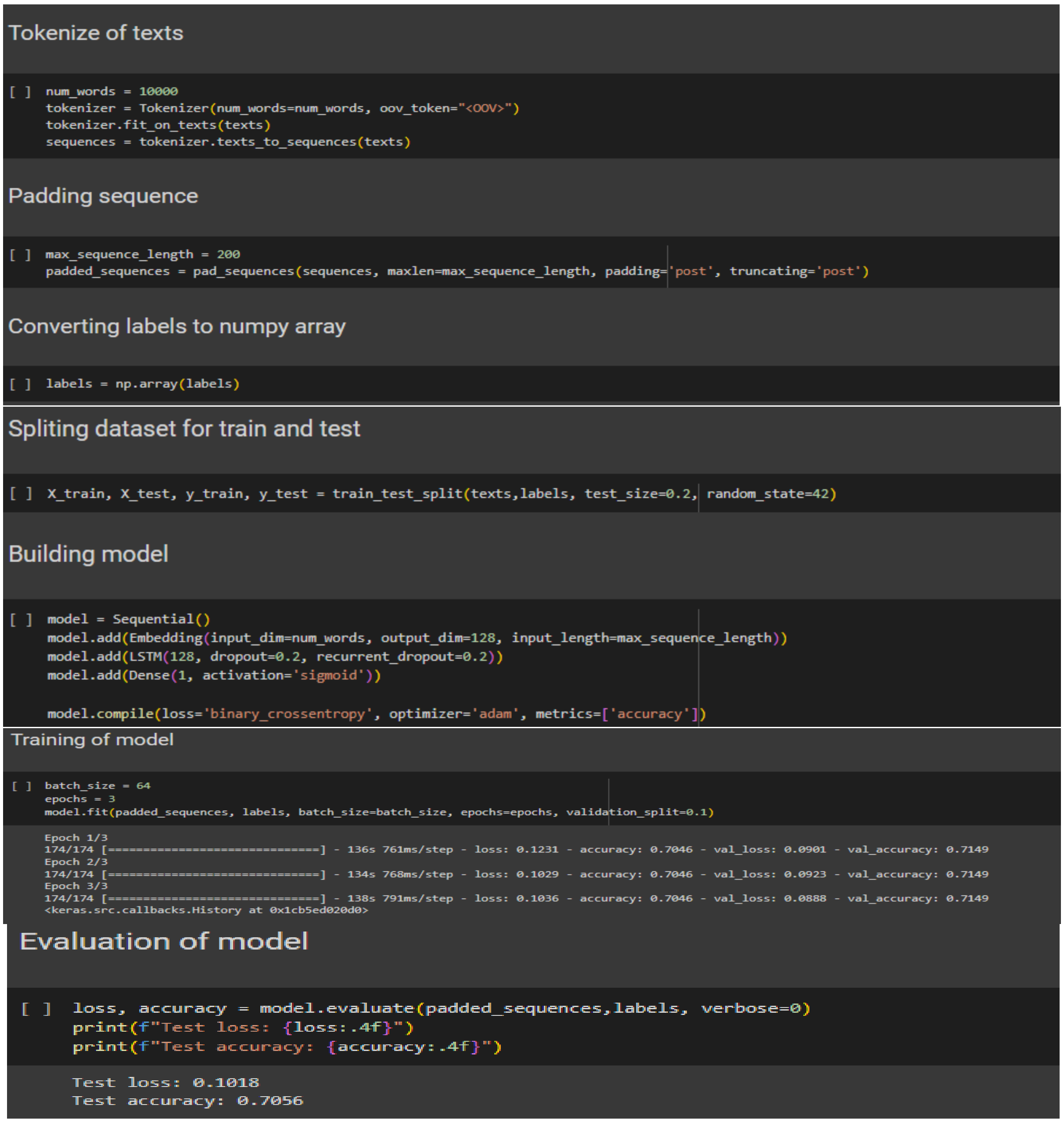
Fig. Bar chart of Comparing of Logistic Regression, Linear SVC, Random Forest

5. Deep Learning

A Sequential model is a linear stack of layers. It is a simple and flexible way to build deep learning models. It is often used for tasks such as image classification, natural language processing, and time series forecasting.

An LSTM (Long Short-Term Memory) is a type of recurrent neural network (RNN) that is specifically designed to handle sequential data. It can learn long-term dependencies, which makes it suitable for tasks such as machine translation, speech recognition, and text summarization.

A dense layer is a type of layer that is fully connected to the previous layer. It is the most common type of layer in deep learning models. It is used to learn the relationship between the features of the input data.



Tokenize of texts

```
[ ] num_words = 10000
    tokenizer = Tokenizer(num_words=num_words, oov_token="<OOV>")
    tokenizer.fit_on_texts(texts)
    sequences = tokenizer.texts_to_sequences(texts)
```

Padding sequence

```
[ ] max_sequence_length = 200
    padded_sequences = pad_sequences(sequences, maxlen=max_sequence_length, padding='post', truncating='post')
```

Converting labels to numpy array

```
[ ] labels = np.array(labels)
```

Spliting dataset for train and test

```
[ ] X_train, X_test, y_train, y_test = train_test_split(texts,labels, test_size=0.2, random_state=42)
```

Building model

```
[ ] model = Sequential()
    model.add(Embedding(input_dim=num_words, output_dim=128, input_length=max_sequence_length))
    model.add(LSTM(128, dropout=0.2, recurrent_dropout=0.2))
    model.add(Dense(1, activation='sigmoid'))

    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Training of model

```
[ ] batch_size = 64
    epochs = 3
    model.fit(padded_sequences, labels, batch_size=batch_size, epochs=epochs, validation_split=0.1)

    Epoch 1/3
    174/174 [==============================] - 136s 761ms/step - loss: 0.1231 - accuracy: 0.7046 - val_loss: 0.0901 - val_accuracy: 0.7149
    Epoch 2/3
    174/174 [==============================] - 134s 768ms/step - loss: 0.1029 - accuracy: 0.7046 - val_loss: 0.0923 - val_accuracy: 0.7149
    Epoch 3/3
    174/174 [==============================] - 138s 791ms/step - loss: 0.1036 - accuracy: 0.7046 - val_loss: 0.0888 - val_accuracy: 0.7149
    <keras.src.callbacks.History at 0x1cb5ed020d0>
```

Evaluation of model

```
[ ] loss, accuracy = model.evaluate(padded_sequences,labels, verbose=0)
    print(f"Test loss: {loss:.4f}")
    print(f"Test accuracy: {accuracy:.4f}")

    Test loss: 0.1018
    Test accuracy: 0.7056
```

VI. Conclusion

Project focused on analyzing sentiment related to the Russia-Ukraine war conflict using data collected from Twitter. analyzed tweets in English, Russian, and Ukrainian languages. The project involved various steps, including data preprocessing, sentiment score generation, hashtag extraction, word cloud creation, and the development of multiple machine learning models.

English was the most commonly used language in the Twitter dataset, followed by Russian and Ukrainian. This reflects the multilingual nature of discussions surrounding the conflict. examined the frequency of tweets over time, which can provide insights into the dynamics and events related to the conflict. identified the top 20 Twitter users based on their tweet frequency. This can help in understanding influential voices in the online discourse about the conflict.

Generated sentiment scores and classified tweets into positive, negative, and neutral categories. This allowed you to gauge public sentiment regarding the conflict.

Extracted positive and negative hashtags, including '#happy', '#smile', '#joy' (positive), and '#sad', '#angry', '#disappointed' (negative). This analysis highlighted the emotional aspects of the tweets.

Created word clouds for positive, negative, and neutral tweets, visualizing the most frequently occurring words in each sentiment category.

Developed several machine learning models, including Logistic Regression, Linear SVC, and Random Forest. These models achieved high accuracy in classifying sentiment in the tweets. models achieved high accuracy, with Linear SVC outperforming the others at 96.98%, closely followed by Logistic Regression (96.13%) and Random Forest (95.80%). also explored deep learning by building a Sequential model with LSTM layers. While the accuracy was comparatively lower, this approach showcased your experimentation with different techniques.

The accurate sentiment classification suggests that machine learning models can effectively categorize sentiment in multilingual social media data.

VII. References

1. Rosenthal, S., Farra, N., &Nakov, P. (2017). SemEval2017 task 4: Sentiment analysis in Twitter. In Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)(p. 502-518)

2. Poria, S., Cambria, E., & Gelbukh, A. (2015). Deep convolutional neural network textual features and multiple kernel learning for utterance-level multimodal sentiment analysis. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing

3. Haddi, E., Liu, X., & Shi, Y. (2013). The role of text preprocessing in sentiment analysis. Procedia Computer Science, 17, 26-32.

4. Riya Suchdev, Pallavi Kotkar, Rahul Ravindran, Sridhar Swamy-Twitter Sentiment Analysis using Machine Learning and Knowledge-based Approach‖ Computer Engineering VES Institute of Technology, University of Mumbai, Mumbai, India.

5. Wang, H., Can, D., Kazemzadeh, A., Bar, F., & Narayanan, S. (2012, July). A system for real-time twitter sentiment analysis of 2012 us presidential election cycle. In Proceedings of the ACL 2012 System Demonstration.

VIII. Data References

30K Tweets with Russia - Ukrainewar hashtag Kaggle, https://www.kaggle.com/code/avinandandutta/twitter-sentiment-analysis-russia-ukraine-conflict/input