Final Year Project Progress Report Two

Fake News Detection and Sentiment Analysis

Submitted to Dr Syed Asim Ali

Muhammad Anas Atiq ---- B19102067

Abdul Hannan Shaikh ----- B19102003

Muhammad Anas --------- B19102066

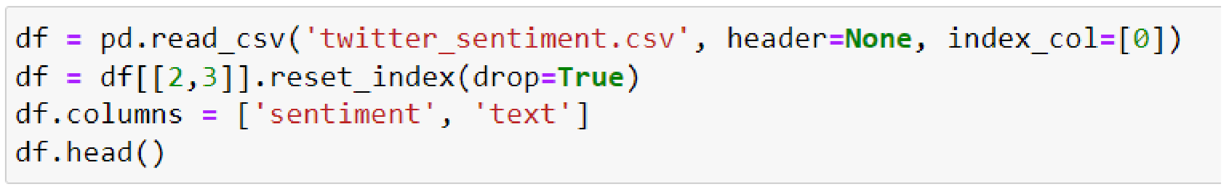
Ameer Hamza Khan ------- B19102016

Introduction

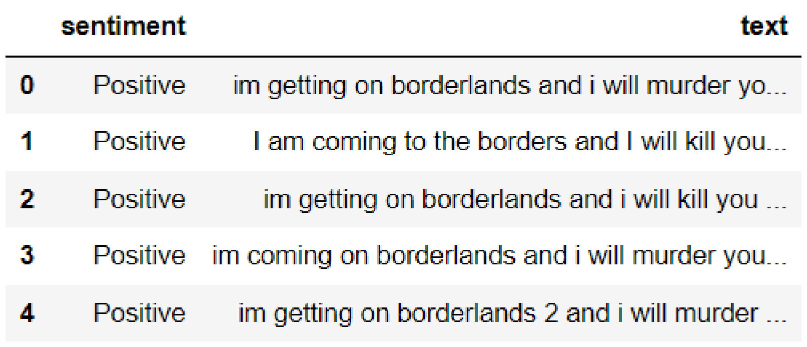
This section of the project focuses on sentiment analysis using a labeled dataset containing Twitter tweets about games. Leveraging Python libraries such as Pandas, Numpy, Spacy, TextBlob, Matplotlib, Seaborn, RE, and Wordcloud, the goal is to analyze the sentiments expressed in the tweets and build a machine learning model for sentiment classification.

Dataset Overview

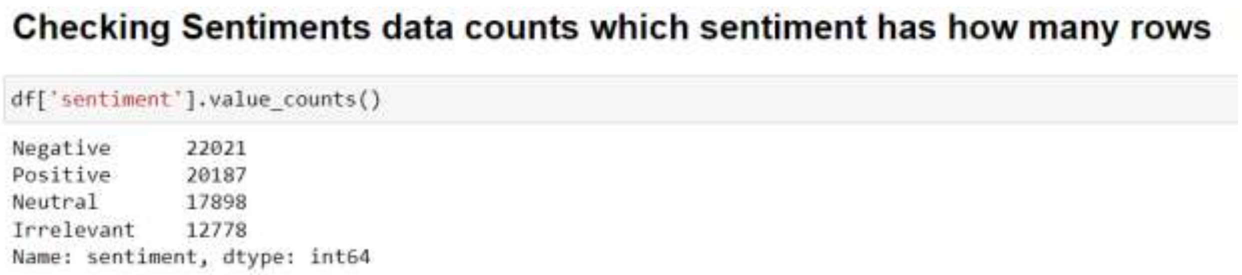
The dataset comprises 75,000 rows, each containing text and its corresponding sentiment label. The sentiment labels include "Positive," "Negative," "Neutral," and "Irrelevant." To ensure data quality, various preprocessing steps were taken.



```python
df = pd.read_csv('twitter_sentiment.csv', header=None, index_col=[0])
df = df[[2,3]].reset_index(drop=True)
df.columns = ['sentiment', 'text']
df.head()
```



	sentiment	text
0	Positive	im getting on borderlands and i will murder yo...
1	Positive	I am coming to the borders and I will kill you...
2	Positive	im getting on borderlands and i will kill you ...
3	Positive	im coming on borderlands and i will murder you...
4	Positive	im getting on borderlands 2 and i will murder ...



Checking Sentiments data counts which sentiment has how many rows

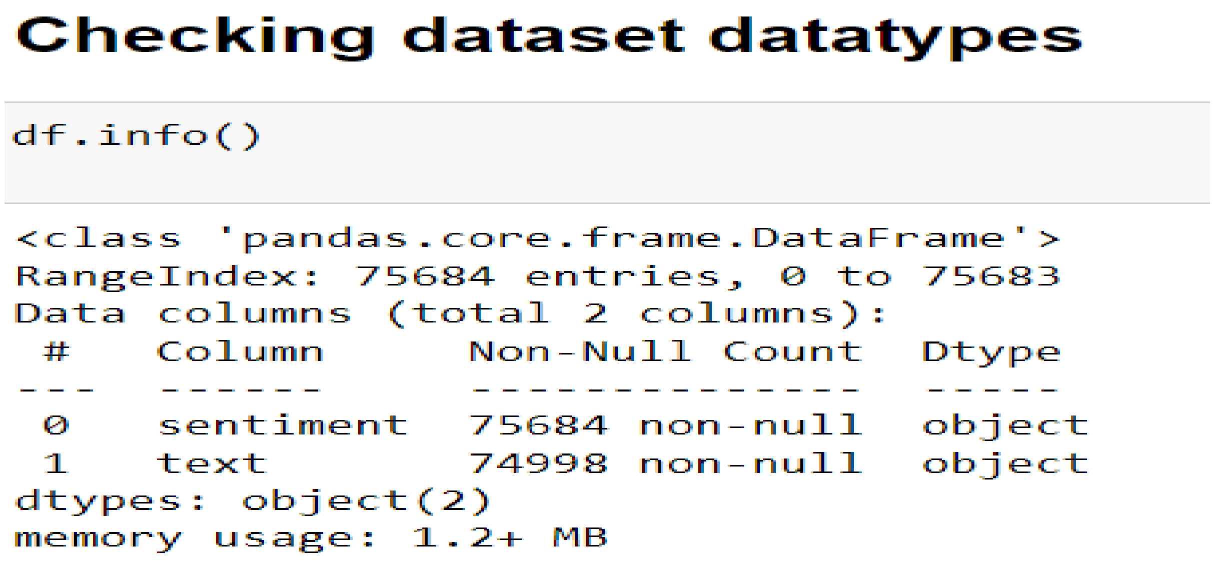
```python
df['sentiment'].value_counts()
```

```
Negative      22021
Positive      20187
Neutral       17898
Irrelevant    12778
Name: sentiment, dtype: int64
```

Data Cleaning and Preprocessing:

1. **Data Type correction and NULL value removal**: The data types were corrected, and null values were removed to maintain data integrity.
2. **Filtering Short Texts**: Texts with a length less than 5 characters were excluded to enhance accuracy in sentiment analysis.

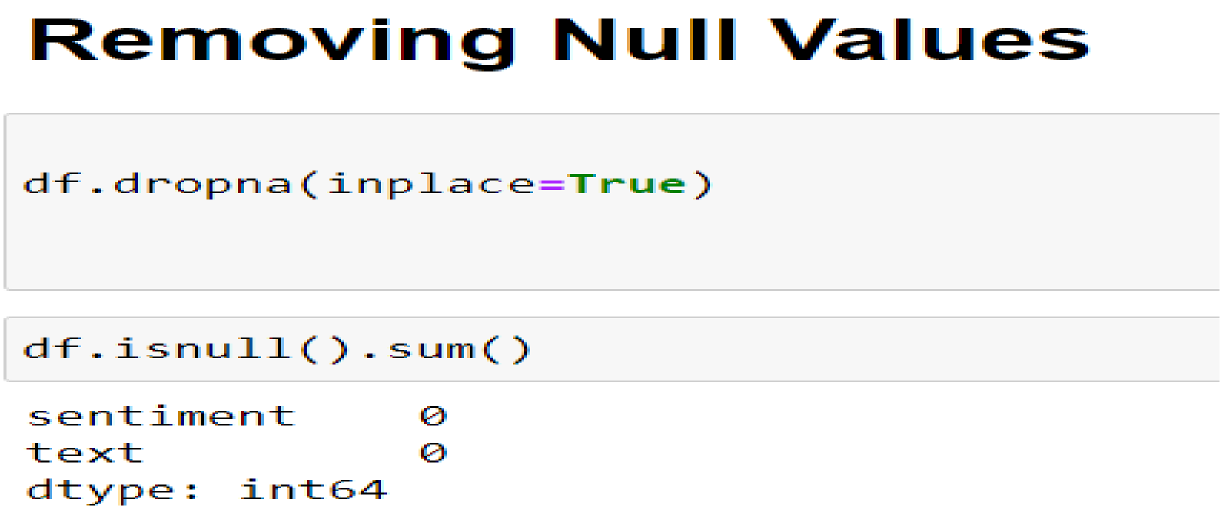
3. **Exploratory Data Analysis (EDA):** Several EDA techniques were applied to better understand the dataset: Character count, Word count, Average word length, Stopwords count, Hashtags count, Mentions count, Digits count, Uppercase counts
4. **Graphical Representation**: The Matplotlib library was utilized to create visualizations for each predefined function concerning sentiment texts. This step provided insights into the distribution of sentiment-related features.



Checking dataset datatypes

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 75684 entries, 0 to 75683
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   sentiment  75684 non-null  object
 1   text       74998 non-null  object
dtypes: object(2)
memory usage: 1.2+ MB
```

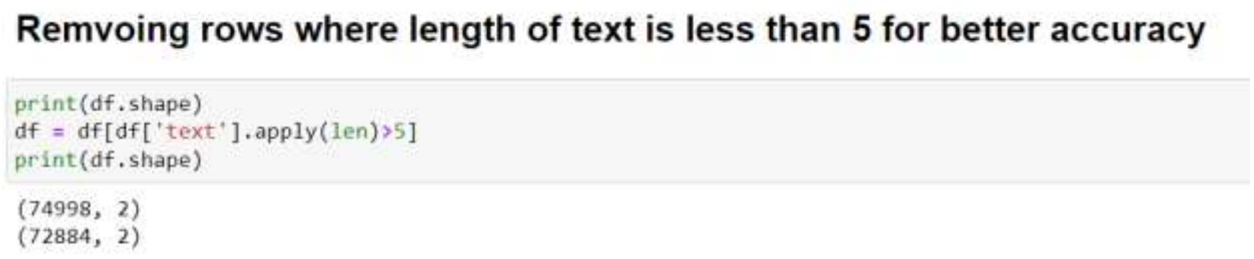


Removing Null Values

```
df.dropna(inplace=True)
```

```
df.isnull().sum()
```

```
sentiment    0
text         0
dtype: int64
```

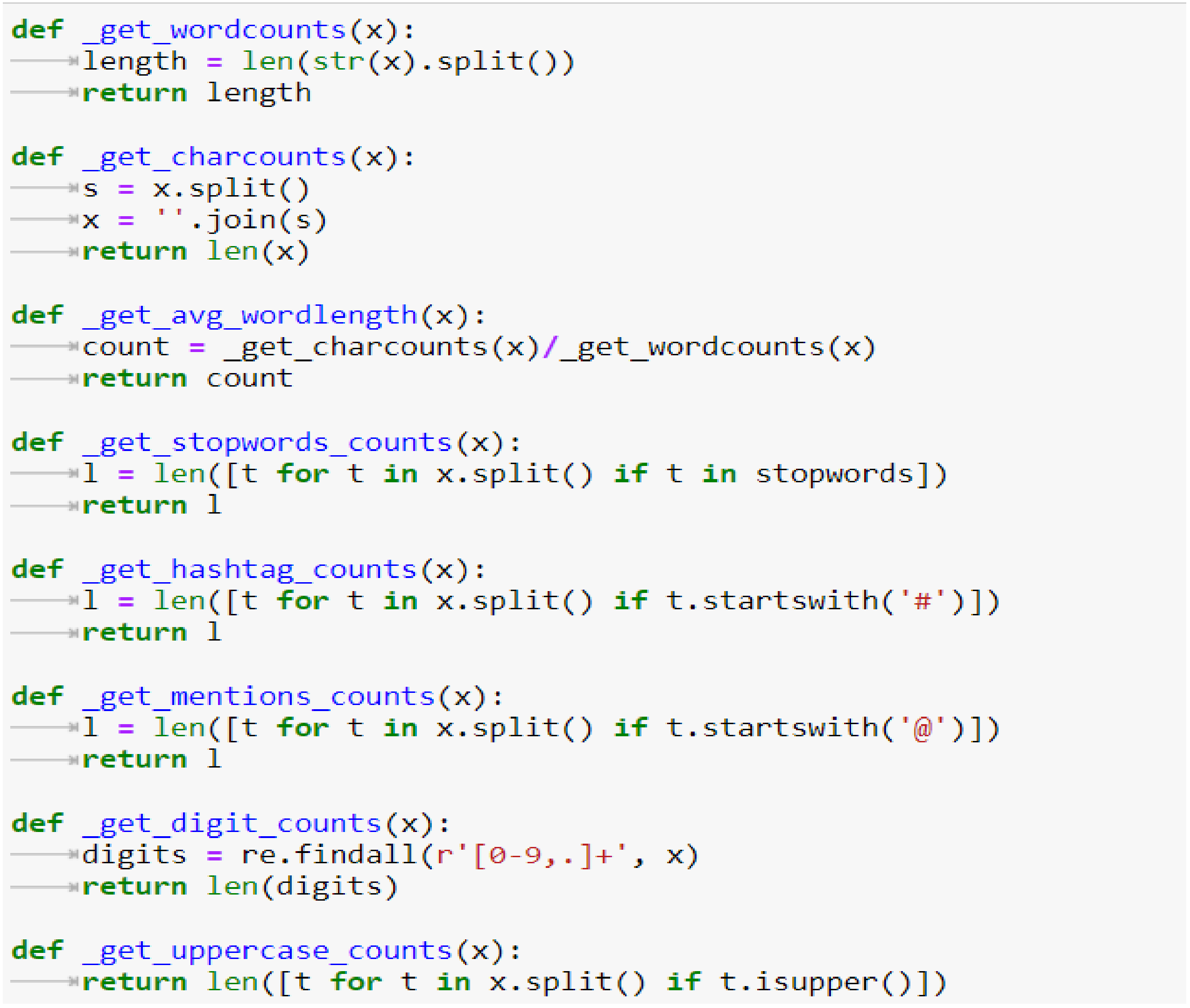


Remvoing rows where length of text is less than 5 for better accuracy

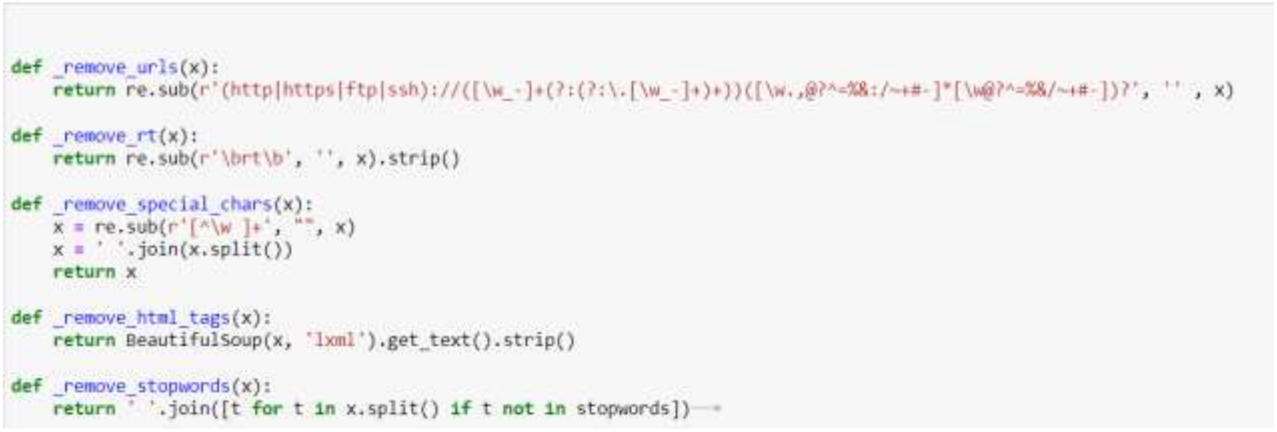
```
print(df.shape)
df = df[df['text'].apply(len)>5]
print(df.shape)
```

```
(74998, 2)
(72884, 2)
```

Doing Exploratry Data Analysis on text data



```python
def _get_wordcounts(x):
    length = len(str(x).split())
    return length

def _get_charcounts(x):
    s = x.split()
    x = ''.join(s)
    return len(x)

def _get_avg_wordlength(x):
    count = _get_charcounts(x)/_get_wordcounts(x)
    return count

def _get_stopwords_counts(x):
    l = len([t for t in x.split() if t in stopwords])
    return l

def _get_hashtag_counts(x):
    l = len([t for t in x.split() if t.startswith('#')])
    return l

def _get_mentions_counts(x):
    l = len([t for t in x.split() if t.startswith('@')])
    return l

def _get_digit_counts(x):
    digits = re.findall(r'[0-9,.]+', x)
    return len(digits)

def _get_uppercase_counts(x):
    return len([t for t in x.split() if t.isupper()])
```



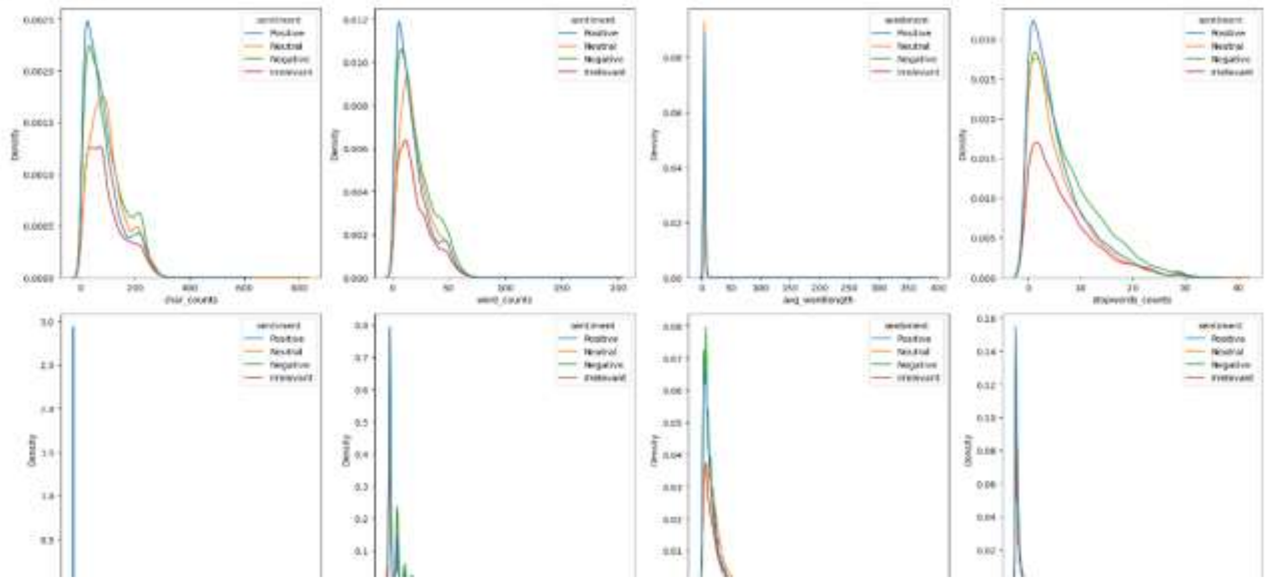
```python
def _remove_urls(x):
    return re.sub(r'(http|https|ftp|ssh)://([\w_-]+(?:(?:\.[\w_-]+)+))([\w.,@?^=%&:/~+#-]*[\w@?^=%&/~+#-])?', '' , x)

def _remove_rt(x):
    return re.sub(r'\brt\b', '', x).strip()

def _remove_special_chars(x):
    x = re.sub(r'[^\w ]+', "", x)
    x = ' '.join(x.split())
    return x

def _remove_html_tags(x):
    return BeautifulSoup(x, 'lxml').get_text().strip()

def _remove_stopwords(x):
    return ' '.join([t for t in x.split() if t not in stopwords])
```

Applying predefined functions on text data for EDA

```
def _get_basic_features(df):
    if type(df) == pd.core.frame.DataFrame:
        df['char_counts'] = df['text'].apply(lambda x: _get_charcounts(x))
        df['word_counts'] = df['text'].apply(lambda x: _get_wordcounts(x))
        df['avg_wordlength'] = df['text'].apply(lambda x: _get_avg_wordlength(x))
        df['stopwords_counts'] = df['text'].apply(lambda x: _get_stopwords_counts(x))
        df['hashtag_counts'] = df['text'].apply(lambda x: _get_hashtag_counts(x))
        df['mentions_counts'] = df['text'].apply(lambda x: _get_mentions_counts(x))
        df['digits_counts'] = df['text'].apply(lambda x: _get_digit_counts(x))
        df['uppercase_counts'] = df['text'].apply(lambda x: _get_uppercase_counts(x))
    else:
        print('ERROR: This function takes only Pandas DataFrame')

    return df
```

Checking each predefined function with respect to Sentiment text

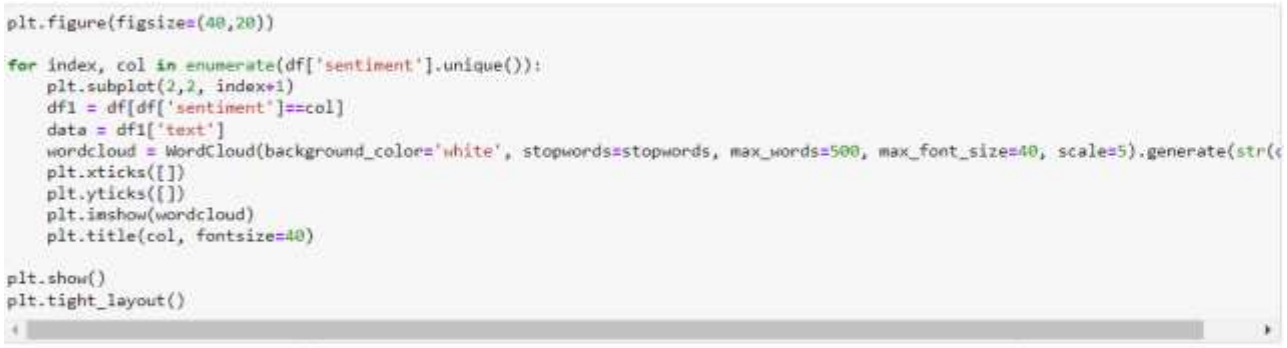
```
plt.figure(figsize=(20,10))

num_cols = df.select_dtypes(include='number').columns

for index,col in enumerate(num_cols):
    plt.subplot(2,4, index+1)
    sns.kdeplot(data=df, x=col, hue='sentiment', fill=False)

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



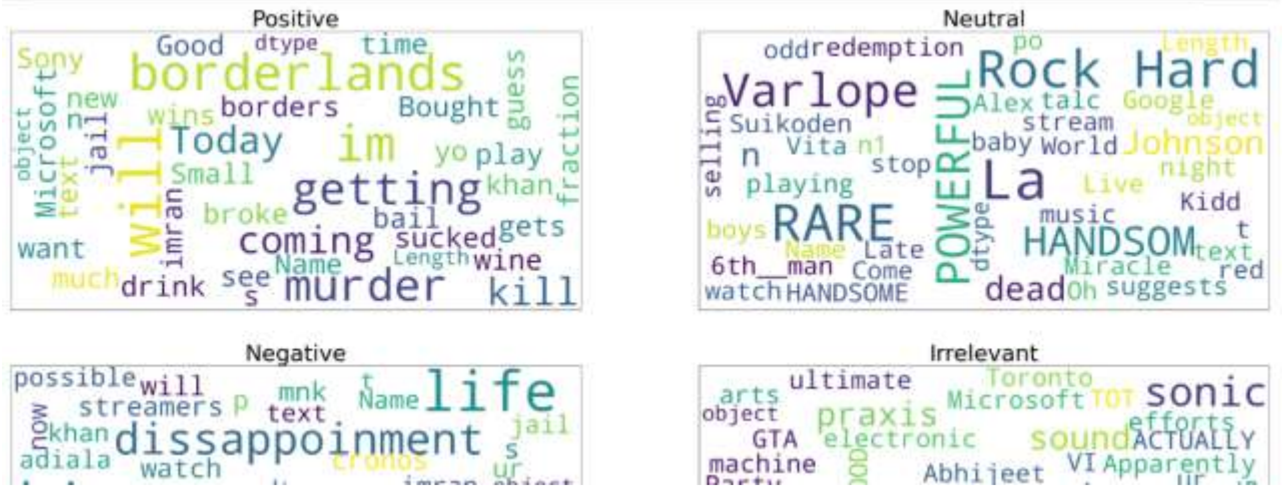
Top Words Analysis

The project delved into identifying the top words associated with each sentiment. This analysis aimed to understand the most frequently used words within each sentiment category.

Checking sentiment wise wordcounts which word comes most often

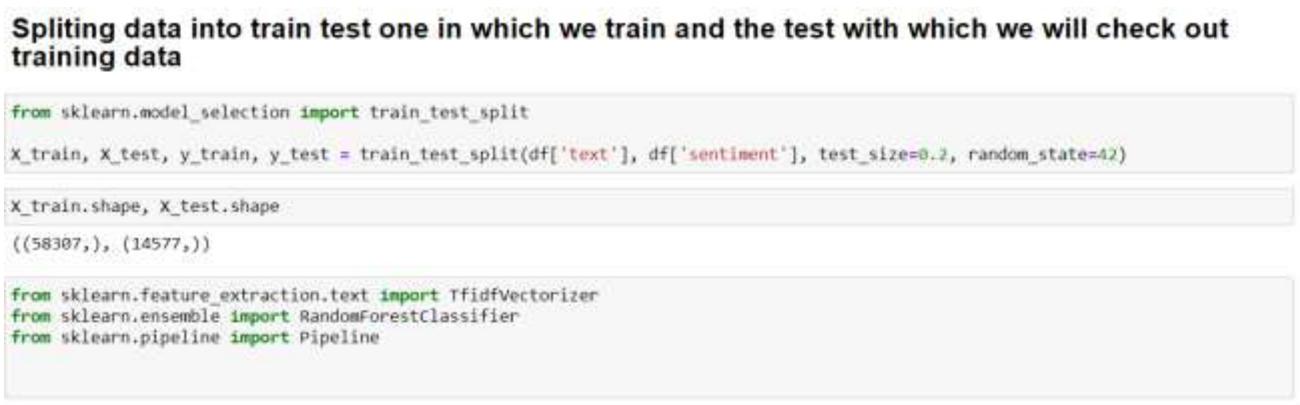


```
plt.figure(figsize=(40,20))

for index, col in enumerate(df['sentiment'].unique()):
    plt.subplot(2,2, index+1)
    df1 = df[df['sentiment']==col]
    data = df1['text']
    wordcloud = WordCloud(background_color='white', stopwords=stopwords, max_words=500, max_font_size=40, scale=5).generate(str(c
    plt.xticks([])
    plt.yticks([])
    plt.imshow(wordcloud)
    plt.title(col, fontsize=40)

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



Data Splitting and Machine Learning Model

1. **Train-Test Split**: The dataset was split into training and testing sets. The training set is utilized to train the machine learning model, while the testing set is reserved for evaluating the model's performance.
2. **Machine Learning Model**: A Random Forest Classifier was chosen for sentiment classification. This algorithm is known for its robustness and efficiency in handling text data.
3. **Pipeline Construction**: To streamline the machine learning process, a pipeline was constructed, encapsulating data preprocessing and model training.
4. **Model Evaluation**: he accuracy of the sentiment classification model was assessed using the testing data. The Random Forest Classifier demonstrated promising results, showcasing its effectiveness in sentiment analysis.



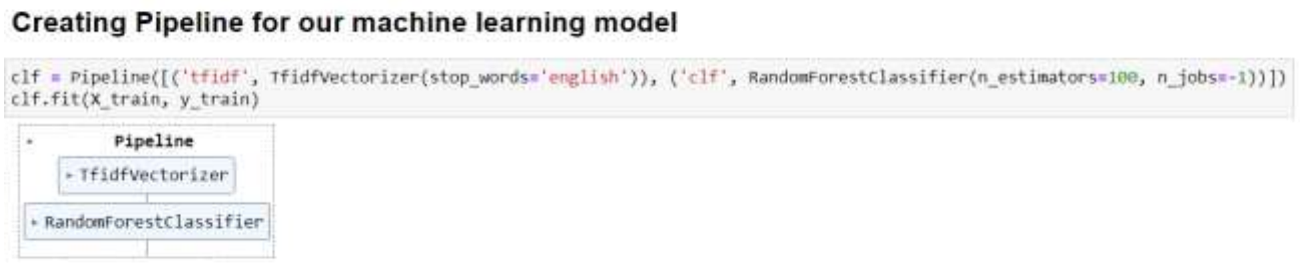
Spliting data into train test one in which we train and the test with which we will check out training data

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(df['text'], df['sentiment'], test_size=0.2, random_state=42)
```

```
X_train.shape, X_test.shape
```

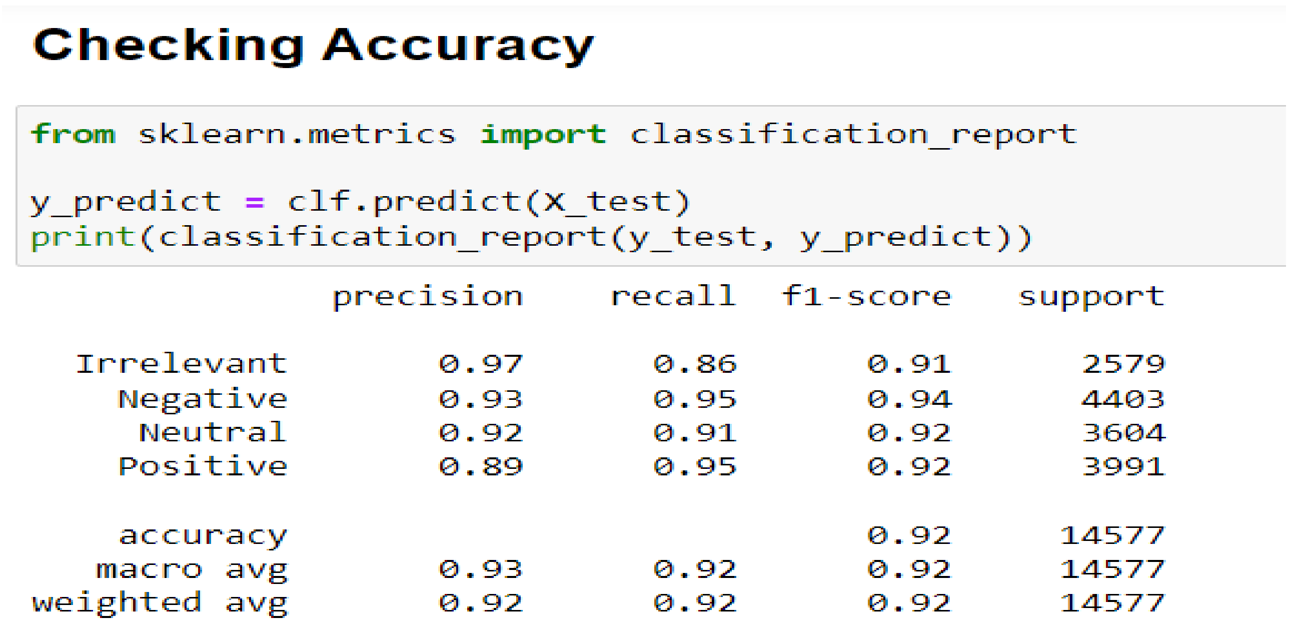
```
((58307,), (14577,))
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import Pipeline
```



Creating Pipeline for our machine learning model

```
clf = Pipeline([('tfidf', TfidfVectorizer(stop_words='english')), ('clf', RandomForestClassifier(n_estimators=100, n_jobs=-1))])
clf.fit(X_train, y_train)
```



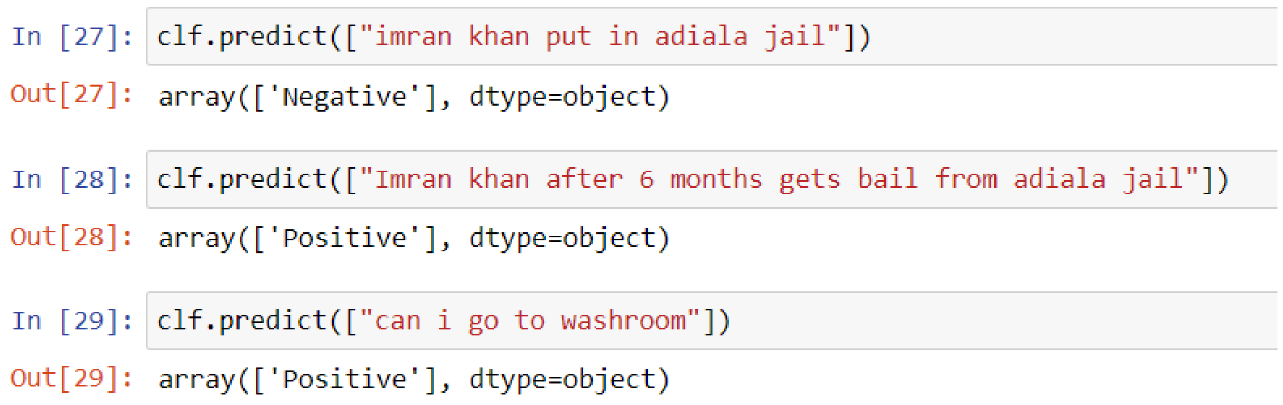
Checking Accuracy

```
from sklearn.metrics import classification_report

y_predict = clf.predict(X_test)
print(classification_report(y_test, y_predict))
```

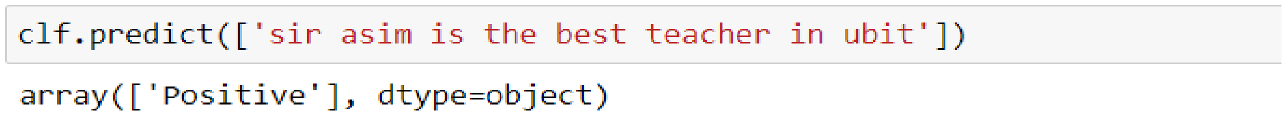
```
              precision    recall  f1-score   support

  Irrelevant       0.97      0.86      0.91      2579
    Negative       0.93      0.95      0.94      4403
     Neutral       0.92      0.91      0.92      3604
    Positive       0.89      0.95      0.92      3991

    accuracy                           0.92     14577
   macro avg       0.93      0.92      0.92     14577
weighted avg       0.92      0.92      0.92     14577
```

Conclusion and Next Steps

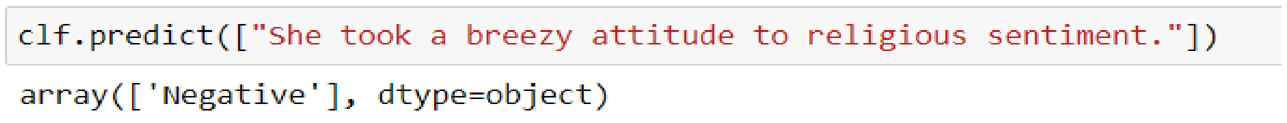
The successful completion of our Fake News Detection and Sentiment Analysis models, utilizing Python libraries such as Pandas, Numpy, Spacy, TextBlob, and more, marks a significant milestone. With 75,000 labeled Twitter data rows, sentiments categorized, and accurate models in place, we're now poised for the next steps:



```
In [27]: clf.predict(["imran khan put in adiala jail"])
Out[27]: array(['Negative'], dtype=object)

In [28]: clf.predict(["Imran khan after 6 months gets bail from adiala jail"])
Out[28]: array(['Positive'], dtype=object)

In [29]: clf.predict(["can i go to washroom"])
Out[29]: array(['Positive'], dtype=object)
```



```
clf.predict(['sir asim is the best teacher in ubit'])
array(['Positive'], dtype=object)
```



```
clf.predict(["She took a breezy attitude to religious sentiment."])
array(['Negative'], dtype=object)
```

Next Steps:

1. **Serialization and Interface**: Serialize models using Pickle for easy retrieval and develop a user-friendly interface for seamless interaction.
2. **Testing and Optimization**: Thoroughly test and optimize the interface for performance.
3. **Deployment and Documentation**: Deploy the interface on the chosen platform and provide clear user documentation for effective utilization.
4. **Integration and Holistic Solution**: Integrate Sentiment Analysis with Fake News Detection for a comprehensive content analysis tool.

These steps aim to make our models practical and accessible, providing users with a powerful solution for navigating and understanding textual content on social media platforms.