Data science project

Topic: Social Media Sentiment Analysis

Name: Ahmed Mohsen Ahmed

ID: 202200061

TKH, Coventry University

School of Computing

Submission date: 27/4/2025

Table of Contents

Table of figures:

Abstract:

This project focuses on solving some research questions about the elections of 2024 and we will collect data on users who comment on related posts for the elections and create a classification model that predicts comments sentimental value for positive, neutral, or negative comments that user writes about the election the data will be split to train and test model on and it will tuned to perform at an approximate accuracy of 79% In addition of finding out if length of comments has any effect of the sentiment of a comment and find out, and what words are most used for each category of comments and the model will be deployed on Streamlit so it can be tested on multiple other comments.

Introduction:

Problem Statement (Research Questions):

Our aim to solve all the research Questions with an analytic approach that can prove that our answers have evidence to back it up, some of the challenges that was faced were creating ML model that predicts such sentiment from comment with high accuracy and to keep data less biased as much as possible to get the best results as project goal is to have detailed solution for each question.

So the research Questions are:

1- Can we create an ML model to predict the sentimental values of comments?
2- Is there any correlation between comment length and sentiment values?
3- Are there certain words that are common in positive or negative comments?

Data Description:

The dataset was collected from Kaggle and further data was scraped from Reddit comments from the r/Election2024 by using the Reddit API.

For our data, we have two files Train.csv and Test.csv which are used for our model that is originally split from the original dataset 75% train and 25% test.

Train.csv, test.csv:

Both Train and Test .csv both have the same columns:

Tweet_id: is the id for every tweet and its auto-increment.

User_handle: username of the user who send the comment.

Comment: The comments themselves.

Sentiment (Target): is the value of the sentimental value of comment ranging from -1 to 1 (-1= Negative, 0= Neutral, 1= Positive).

Comment length: is the length of a comment in words.

Scaled_length: is a Scaled version of the comments to be easier on the model.

Length Category: is a Category of length that less than 50 is short and between 50 and 150 is medium and anything above 150 is long.

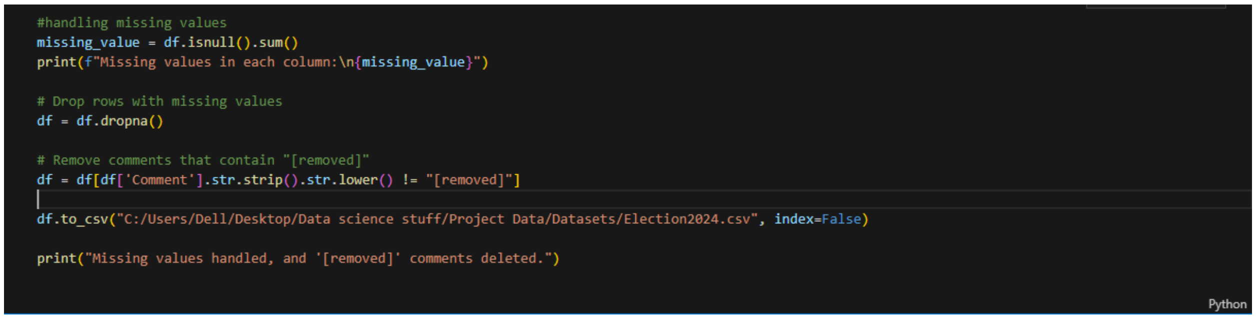
Data Preprocessing:

A couple steps were taken to prepare data before using it in the model:

Data Cleaning:

Handle missing value:

We handle missing values by drops all the missing rows and also removing any comment that was removed by Reddit



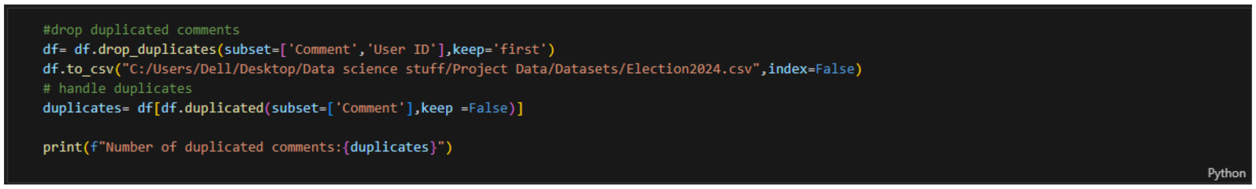
```python
#handling missing values
missing_value = df.isnull().sum()
print(f"Missing values in each column:\n{missing_value}")

# Drop rows with missing values
df = df.dropna()

# Remove comments that contain "[removed]"
df = df[df['Comment'].str.strip().str.lower() != "[removed]"]

df.to_csv("C:/Users/Dell/Desktop/Data science stuff/Project Data/Datasets/Election2024.csv", index=False)

print("Missing values handled, and '[removed]' comments deleted.")
```

Python

Figure 1.0 Screenshot of missing value code

Handle duplicated comments:

By using pandas we just use drop_duplicates



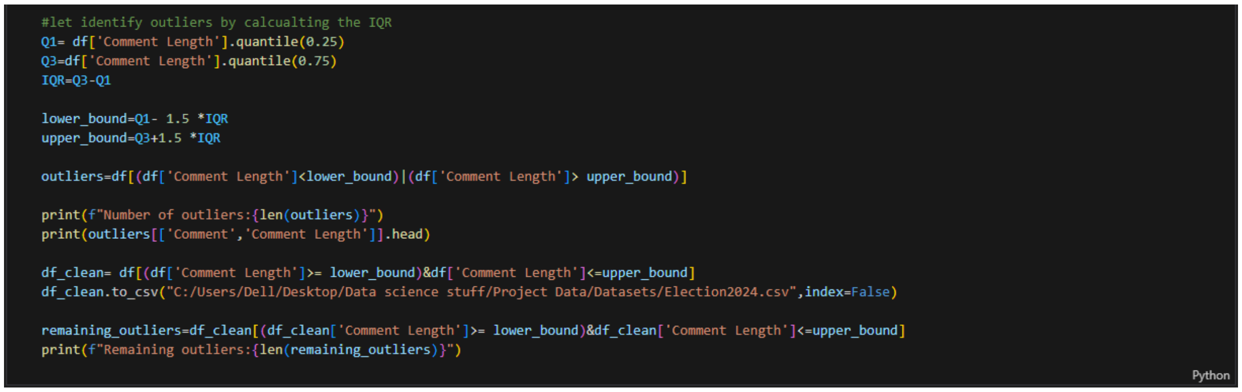
```python
#drop duplicated comments
df= df.drop_duplicates(subset=['Comment','User ID'],keep='first')
df.to_csv("C:/Users/Dell/Desktop/Data science stuff/Project Data/Datasets/Election2024.csv",index=False)
# handle duplicates
duplicates= df[df.duplicated(subset=['Comment'],keep =False)]

print(f"Number of duplicated comments:{duplicates}")
```

Python

Figure 1.1 Screenshot of drop duplicates code

Removing Outliers:

We calculate IQR to find values that are outside of the normal range to remove



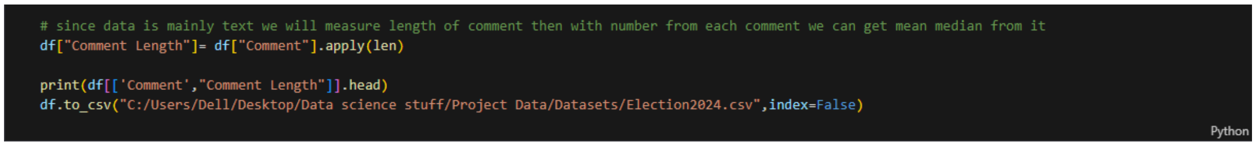
```python
#let identify outliers by calcualting the IQR
Q1= df['Comment Length'].quantile(0.25)
Q3=df['Comment Length'].quantile(0.75)
IQR=Q3-Q1

lower_bound=Q1- 1.5 *IQR
upper_bound=Q3+1.5 *IQR

outliers=df[(df['Comment Length']<lower_bound)|(df['Comment Length']> upper_bound)]

print(f"Number of outliers:{len(outliers)}")
print(outliers[['Comment','Comment Length']].head)

df_clean= df[(df['Comment Length']>= lower_bound)&df['Comment Length']<=upper_bound]
df_clean.to_csv("C:/Users/Dell/Desktop/Data science stuff/Project Data/Datasets/Election2024.csv",index=False)

remaining_outliers=df_clean[(df_clean['Comment Length']>= lower_bound)&df_clean['Comment Length']<=upper_bound]
print(f"Remaining outliers:{len(remaining_outliers)}")
```

Figure 1.2 Screenshot of remove outliers code

Feature Engineering:

Create Comment length column:

We wanted to count the length of each comment so we could use it in answering one of our research questions

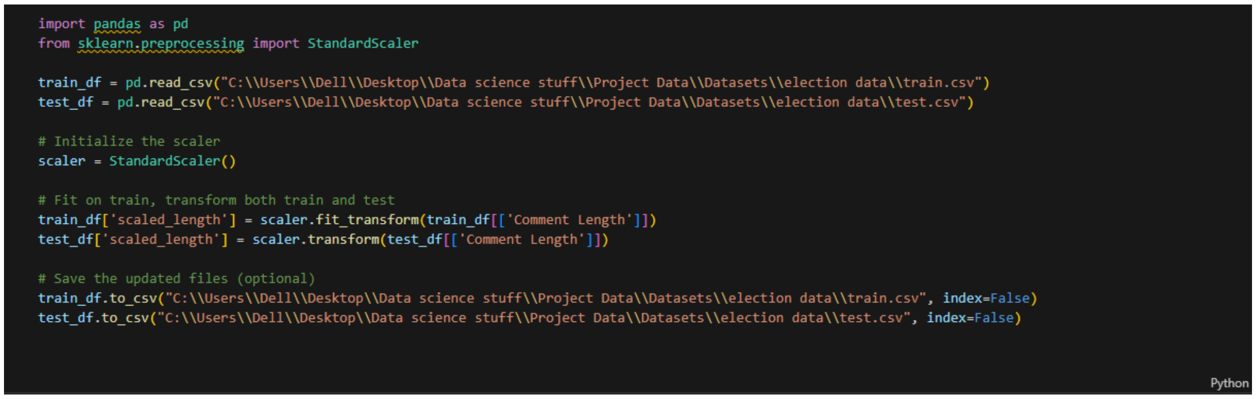


```python
# since data is mainly text we will measure length of comment then with number from each comment we can get mean median from it
df["Comment Length"]= df["Comment"].apply(len)

print(df[['Comment',"Comment Length"]].head)
df.to_csv("C:/Users/Dell/Desktop/Data science stuff/Project Data/Datasets/Election2024.csv",index=False)
```

Figure 1.3 Screenshot of code creating new column Comment length

Scale comment length:

We scale the comment length so we want to use it in our model

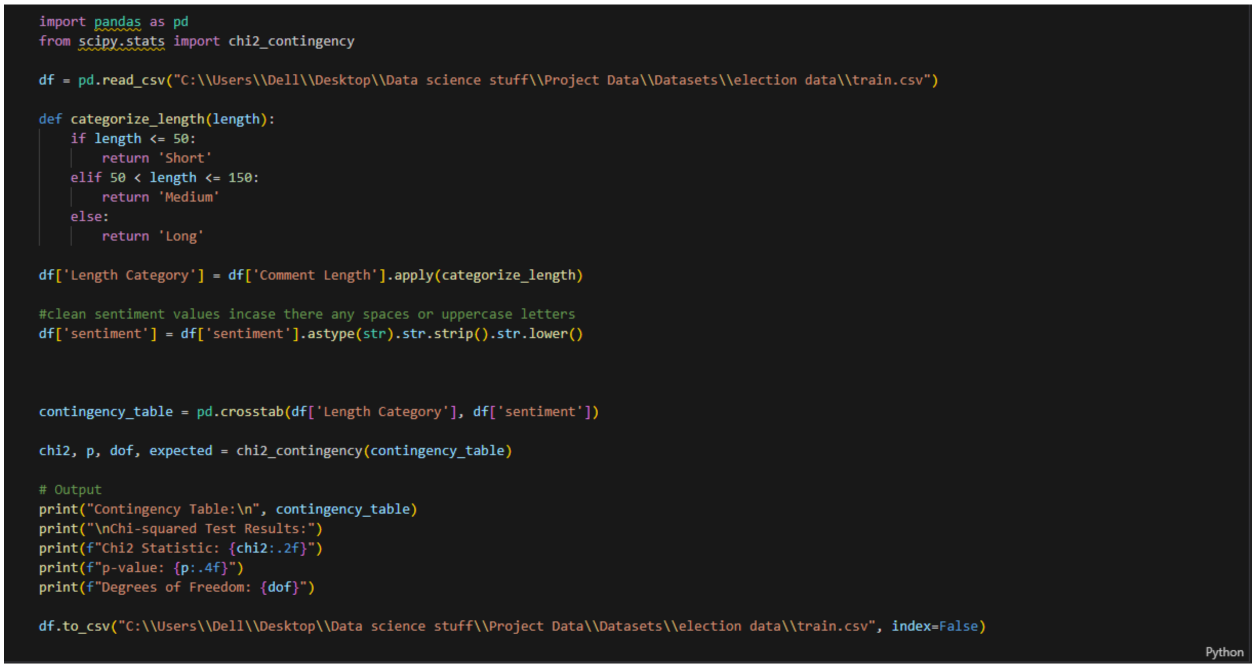


```python
import pandas as pd
from sklearn.preprocessing import StandardScaler

train_df = pd.read_csv("C:\\Users\\Dell\\Desktop\\Data science stuff\\Project Data\\Datasets\\election data\\train.csv")
test_df = pd.read_csv("C:\\Users\\Dell\\Desktop\\Data science stuff\\Project Data\\Datasets\\election data\\test.csv")

# Initialize the scaler
scaler = StandardScaler()

# Fit on train, transform both train and test
train_df['scaled_length'] = scaler.fit_transform(train_df[['Comment Length']])
test_df['scaled_length'] = scaler.transform(test_df[['Comment Length']])

# Save the updated files (optional)
train_df.to_csv("C:\\Users\\Dell\\Desktop\\Data science stuff\\Project Data\\Datasets\\election data\\train.csv", index=False)
test_df.to_csv("C:\\Users\\Dell\\Desktop\\Data science stuff\\Project Data\\Datasets\\election data\\test.csv", index=False)
```

Figure 1.4 Screen shot of code creating a new column for Scaled comment length

Categorize the comment length:

While we are doing the Chi-squared test we categorize the comment length to make it easier when we answer our question:

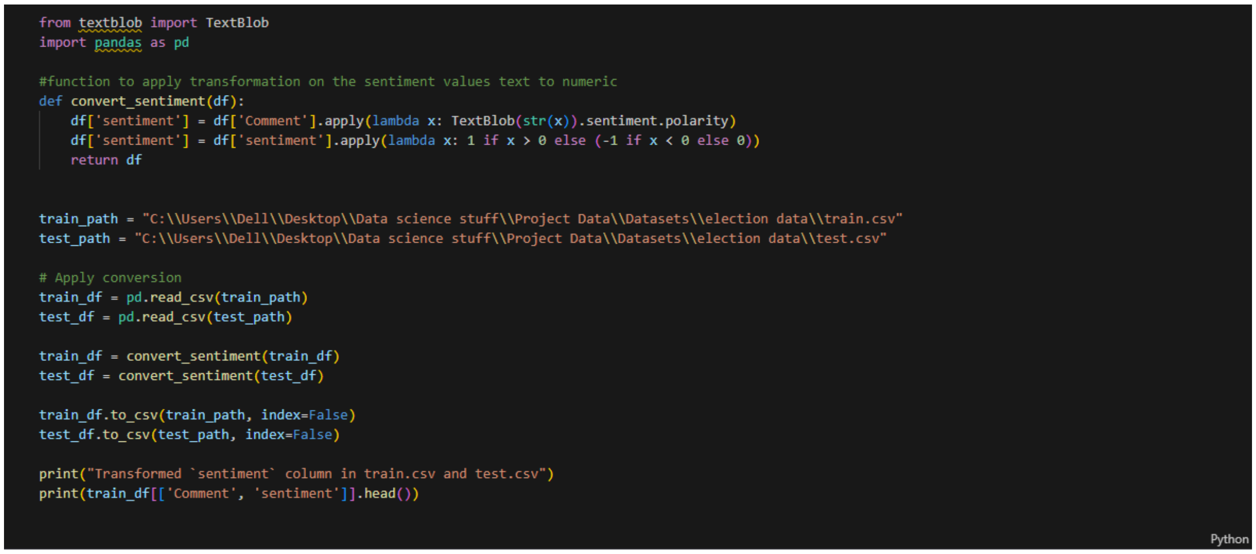


```python
import pandas as pd
from scipy.stats import chi2_contingency

df = pd.read_csv("C:\\Users\\Dell\\Desktop\\Data science stuff\\Project Data\\Datasets\\election data\\train.csv")

def categorize_length(length):
    if length <= 50:
        return 'Short'
    elif 50 < length <= 150:
        return 'Medium'
    else:
        return 'Long'

df['Length Category'] = df['Comment Length'].apply(categorize_length)

#clean sentiment values incase there any spaces or uppercase letters
df['sentiment'] = df['sentiment'].astype(str).str.strip().str.lower()



contingency_table = pd.crosstab(df['Length Category'], df['sentiment'])

chi2, p, dof, expected = chi2_contingency(contingency_table)

# Output
print("Contingency Table:\n", contingency_table)
print("\nChi-squared Test Results:")
print(f"Chi2 Statistic: {chi2:.2f}")
print(f"p-value: {p:.4f}")
print(f"Degrees of Freedom: {dof}")

df.to_csv("C:\\Users\\Dell\\Desktop\\Data science stuff\\Project Data\\Datasets\\election data\\train.csv", index=False)
```

Figure 1.4 Screen shot of code for create a new column for Categorized comments lengths

Transforming the sentimental values:

We transform the sentimental value from text to numeric



```python
from textblob import TextBlob
import pandas as pd

#function to apply transformation on the sentiment values text to numeric
def convert_sentiment(df):
    df['sentiment'] = df['Comment'].apply(lambda x: TextBlob(str(x)).sentiment.polarity)
    df['sentiment'] = df['sentiment'].apply(lambda x: 1 if x > 0 else (-1 if x < 0 else 0))
    return df


train_path = "C:\\Users\\Dell\\Desktop\\Data science stuff\\Project Data\\Datasets\\election data\\train.csv"
test_path = "C:\\Users\\Dell\\Desktop\\Data science stuff\\Project Data\\Datasets\\election data\\test.csv"

# Apply conversion
train_df = pd.read_csv(train_path)
test_df = pd.read_csv(test_path)

train_df = convert_sentiment(train_df)
test_df = convert_sentiment(test_df)

train_df.to_csv(train_path, index=False)
test_df.to_csv(test_path, index=False)

print("Transformed `sentiment` column in train.csv and test.csv")
print(train_df[['Comment', 'sentiment']].head())
```

Figure 1.5 Screen shot for code that transforms the sentimental values to numeric

Analytics approaches:

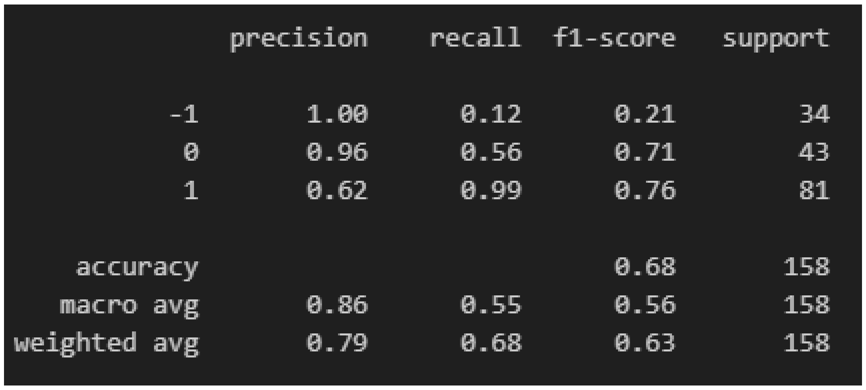
Machine Learning Model (Research Question 1):

For our model, we decided to go for Logistic Regression classifier + TF-IDF (Term frequency-inverse Document Frequency) which is a great model for classification for this problem.

TF-IDF: converts each comment to a numeric vector for how often each word is repeated and common words are given lower weight while the more unique ones are given a bigger weight

Logistic Regression: takes the converted numeric vector and it classifies them into our sentimental categories (Positive, Neutral, and Negative).

We fit the model with the train data and then we give test data x field only so it can predict the target here was the result after predictions:



```
              precision    recall  f1-score   support

          -1       1.00      0.12      0.21        34
           0       0.96      0.56      0.71        43
           1       0.62      0.99      0.76        81

    accuracy                           0.68       158
   macro avg       0.86      0.55      0.56       158
weighted avg       0.79      0.68      0.63       158
```

Figure 2.0 screen shot of Classification model states for predictions

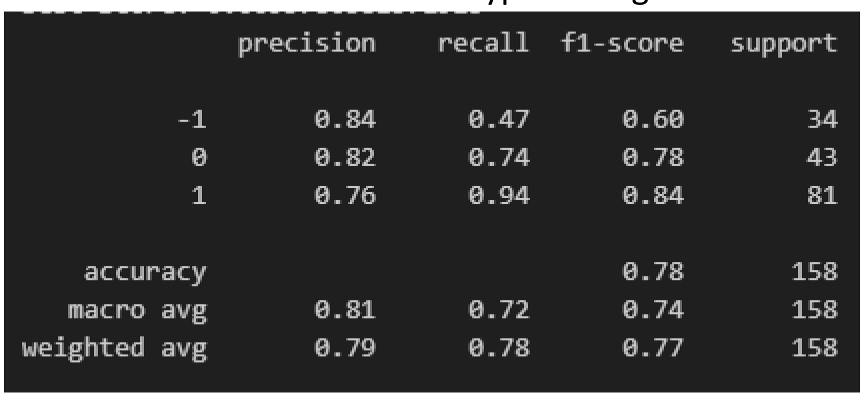
We can see that all predicted negative comments from the model were correct but he did not get all the negative comments right due to low recall meaning it missed some negative comments but that is because most of the data is positive I also predicted most neutral values but didn’t get them all and positive guess some and it catch almost all the positive comment so the model is really good at finding positive comments.

Hyper Tuning:

After hypertuning our model using the Grid search method:

The hypertuning tries many different until it finds the best set to get the best results.

Here result after hypertuning:



```
              precision    recall  f1-score   support

          -1       0.84      0.47      0.60        34
           0       0.82      0.74      0.78        43
           1       0.76      0.94      0.84        81

    accuracy                           0.78       158
   macro avg       0.81      0.72      0.74       158
weighted avg       0.79      0.78      0.77       158
```

Figure 2.1 Screen shot of classification model predictions after hyper tuning

The most noticeable change is the accuracy has increased by 10% while the precision for negative comments got was less the recall got higher which is better and the same for neutral comments the recall on positive got a little worst but the precision got a lot better.

Model performance:

So we were able to create a ML model that can predict the sentiment value of the comment given and here is a visualization of the performance of the model:

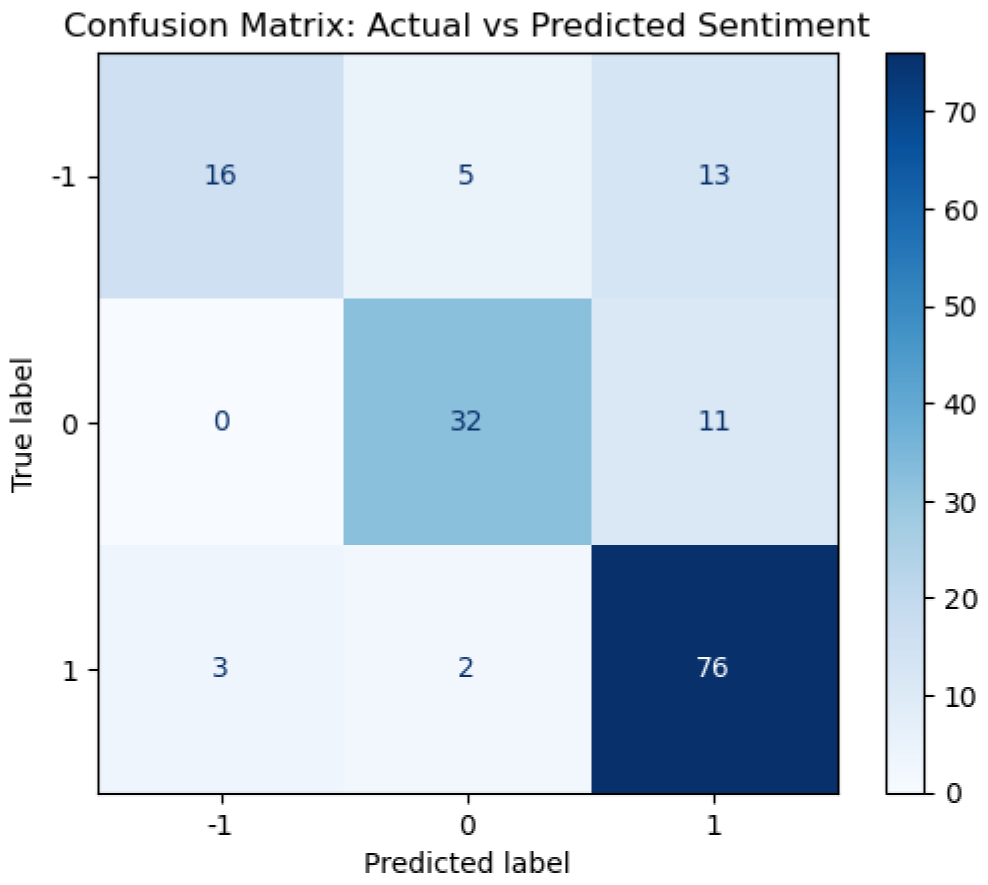


Figure 2.2 visualization graph for the model predictions for the accuracy confusion matrix

This is a confusion matrix that shows the true sentimental value vs what was predicted and tells you how many were right.

Research Question 2:

To solve the question we first categorize the comment lengths to make it easier to tell which comments are short, medium, or long we also make sentiment values numeric to make sure there are no errors in typos, and form this graph:

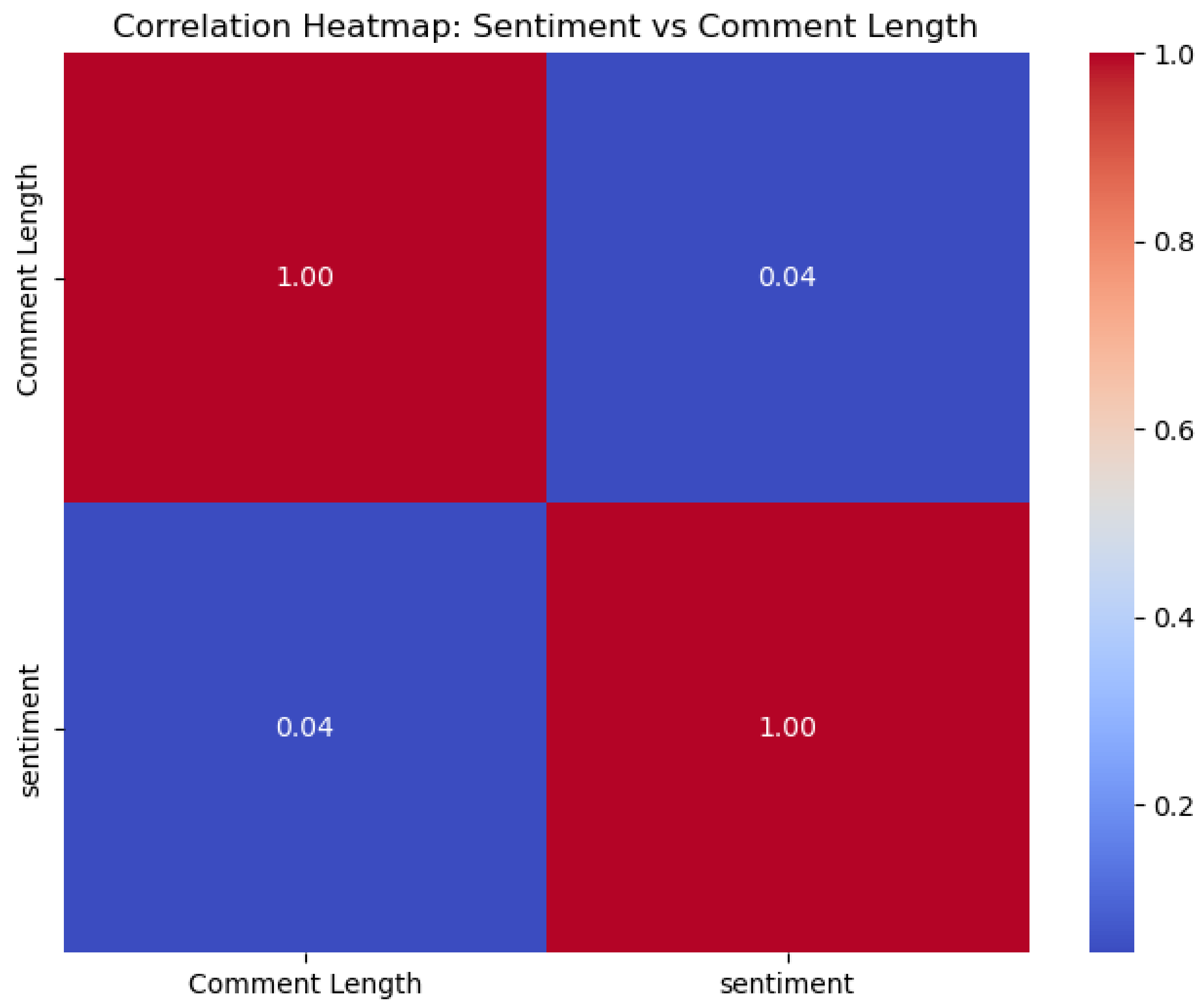
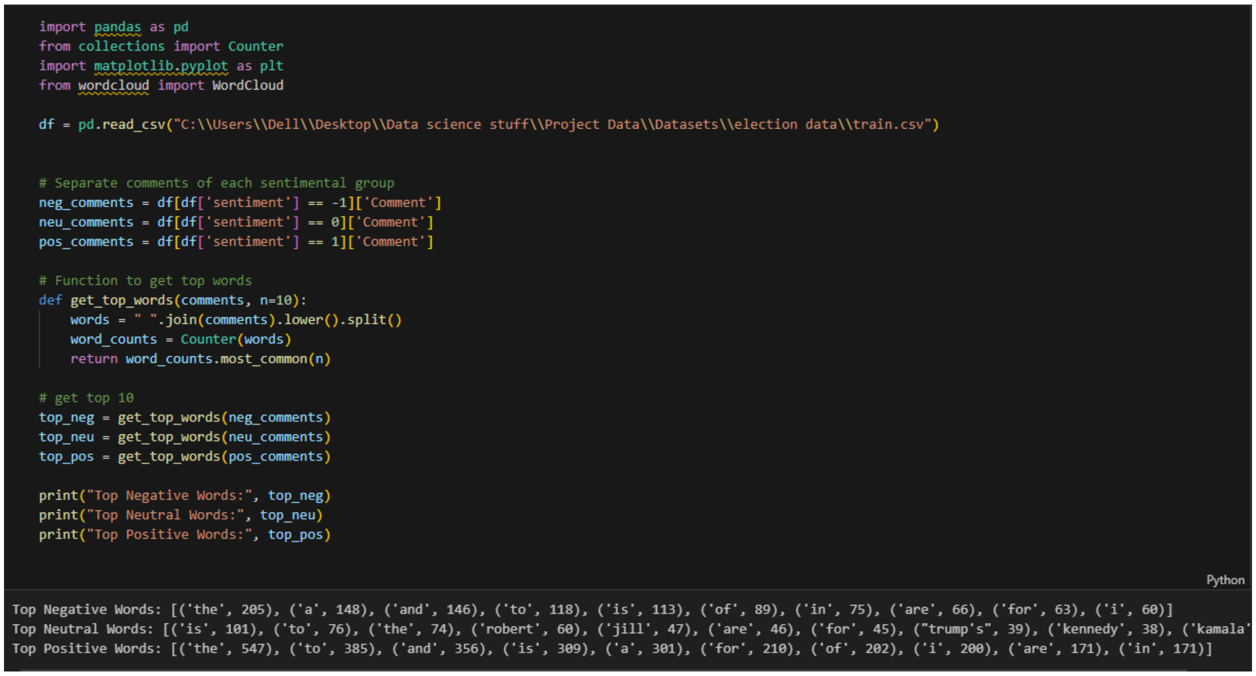


Figure 3.0 heatmap that visualize correlation between Comment length and Sentimental value

We can see that it is 0.04 which is almost 0 which means that Comment Length has no impact on the sentimental values that means if a comment is long or short it cannot tell us if the comment is positive, neutral, or negative and that shows that there not a strong correlation between comment length and sentimental value.

Research Question 3:

To solve such a question, we split comments into different groups to separate the positive, neutral, and negative comments, and then we create functions that get the top 10 words for each group, which means words with the highest counts



```python
import pandas as pd
from collections import Counter
import matplotlib.pyplot as plt
from wordcloud import WordCloud

df = pd.read_csv("C:\\Users\\Dell\\Desktop\\Data science stuff\\Project Data\\Datasets\\election data\\train.csv")


# Separate comments of each sentimental group
neg_comments = df[df['sentiment'] == -1]['Comment']
neu_comments = df[df['sentiment'] == 0]['Comment']
pos_comments = df[df['sentiment'] == 1]['Comment']

# Function to get top words
def get_top_words(comments, n=10):
    words = " ".join(comments).lower().split()
    word_counts = Counter(words)
    return word_counts.most_common(n)

# get top 10
top_neg = get_top_words(neg_comments)
top_neu = get_top_words(neu_comments)
top_pos = get_top_words(pos_comments)

print("Top Negative Words:", top_neg)
print("Top Neutral Words:", top_neu)
print("Top Positive Words:", top_pos)
```

```
Top Negative Words: [('the', 205), ('a', 148), ('and', 146), ('to', 118), ('is', 113), ('of', 89), ('in', 75), ('are', 66), ('for', 63), ('i', 60)]
Top Neutral Words: [('is', 101), ('to', 76), ('the', 74), ('robert', 60), ('jill', 47), ('are', 46), ('for', 45), ("trump's", 39), ('kennedy', 38), ('kamala'
Top Positive Words: [('the', 547), ('to', 385), ('and', 356), ('is', 309), ('a', 301), ('for', 210), ('of', 202), ('i', 200), ('are', 171), ('in', 171)]
```

Figure 4.0 screen shot of code for the splitting of comments to 3 groups for each sentimental value

Here a graph for top 5 words from positive and negative comment (unweighted):

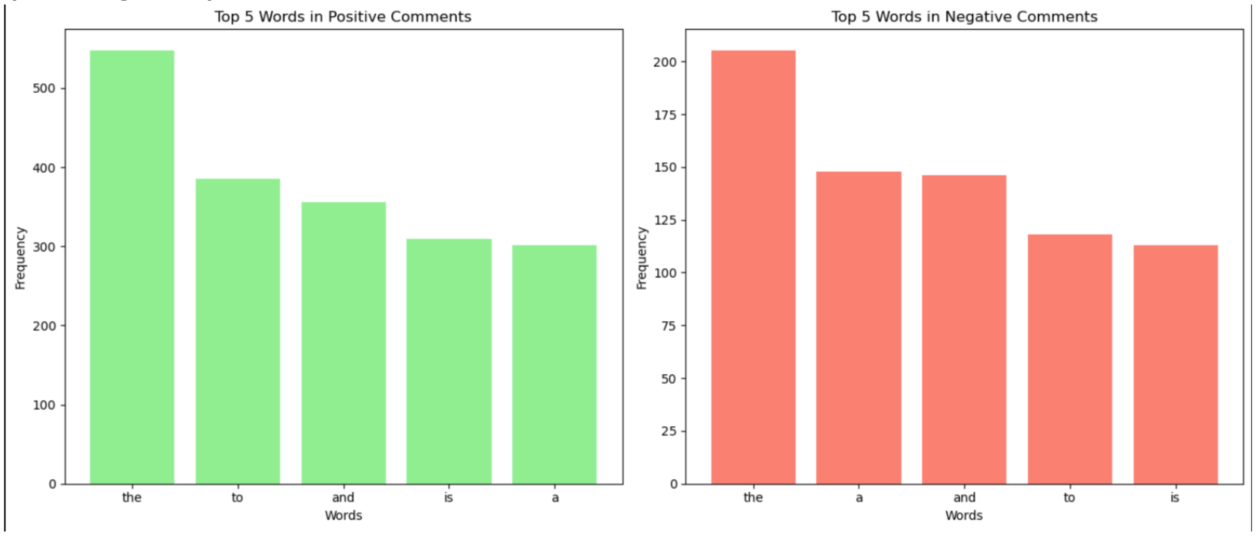


Figure 4.1 graph for most repeated words for positive and negative comments

Challenges to the code:

Empty comments:

The app handles empty comments by showing a warning that text input is empty but if we want to prevent this problem completely we can strip to prevent model from predicting empty inputs.

Very Short or very long comment:

Extremely short comments or long one might confuse our model to counter such problem we can allow model to predict such comment but log warning for the comment being too short or long, or better method is to set minimum and maximum character length.

Incorrect Data input:

Column had incorrect types of string if positive was written wrong for example to avoid such issue we change these values to numbers from -1 to 1 so that there less change for errors to happen for our data.

Conclusion:

In conclusion, we were successful in answering all our research questions with a full detailed description on how we did it and had evidence at the end that proved our points and successfully created our ML classification model which we deployed on streamlit for the public and we managed to create such model with a pretty high accuracy and little chance for errors.

Link to GitHub:

https://github.com/Cookieking2490/Data-science-project

Link to Streamlit app:

https://data-science-project-7txhakkuxlr9ip8rqwzo3o.streamlit.app/

Reference used for the project:

1- *scikit-learn: machine learning in Python — scikit-learn 1.6.1 documentation*. (). https://scikit-learn.org/stable/

2- *pandas documentation — pandas 2.2.3 documentation*. (). https://pandas.pydata.org/docs/