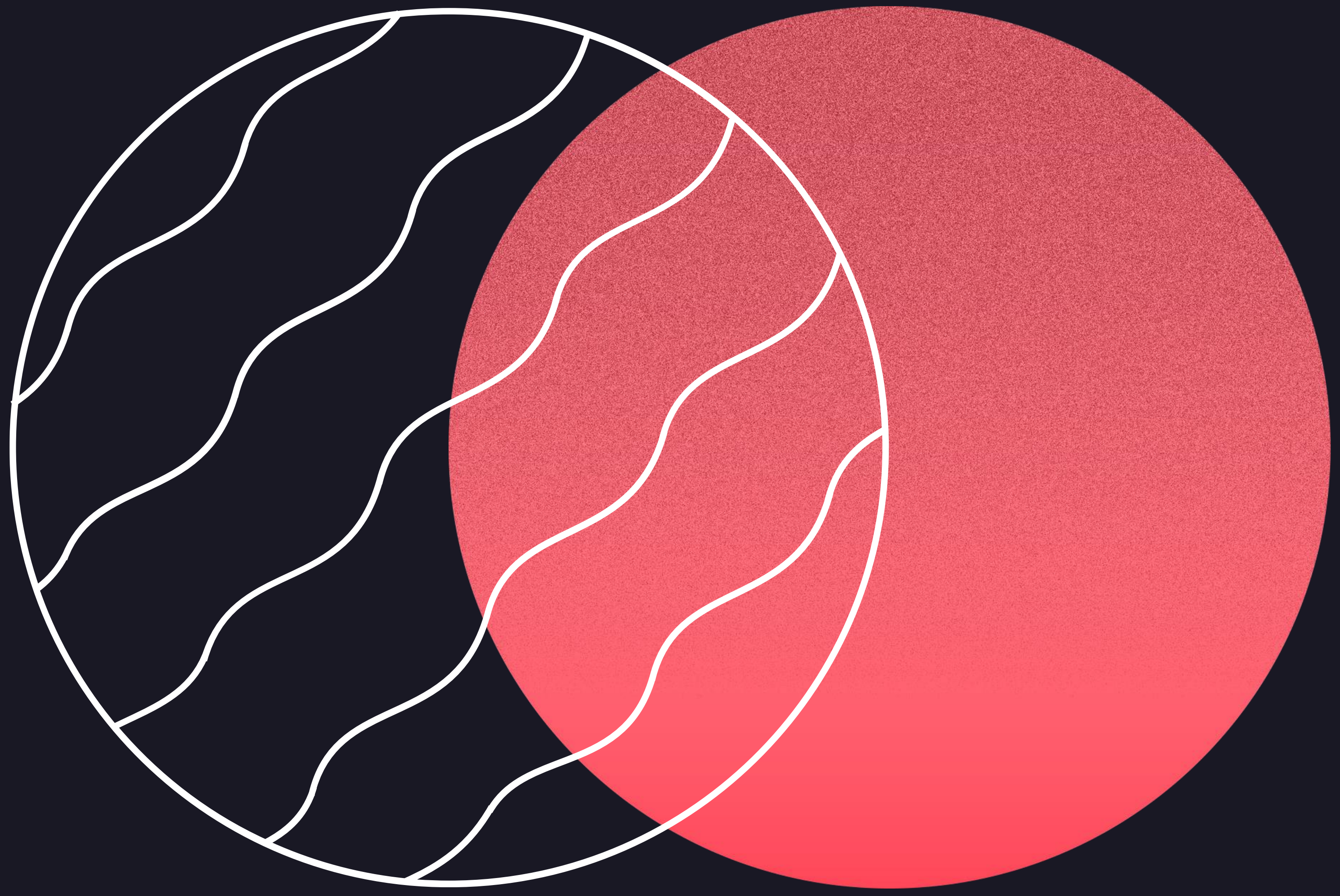
CIS 600 – Social Media and Data Mining



Analyzing Public Perception and Engagement with Climate Change on Reddit

Project Members:

- Atharva Kulkarni
- Nimeesh Bagwe
- Yogesh Patil
- Rahul Pudurkar
- Divit Shetty

Index

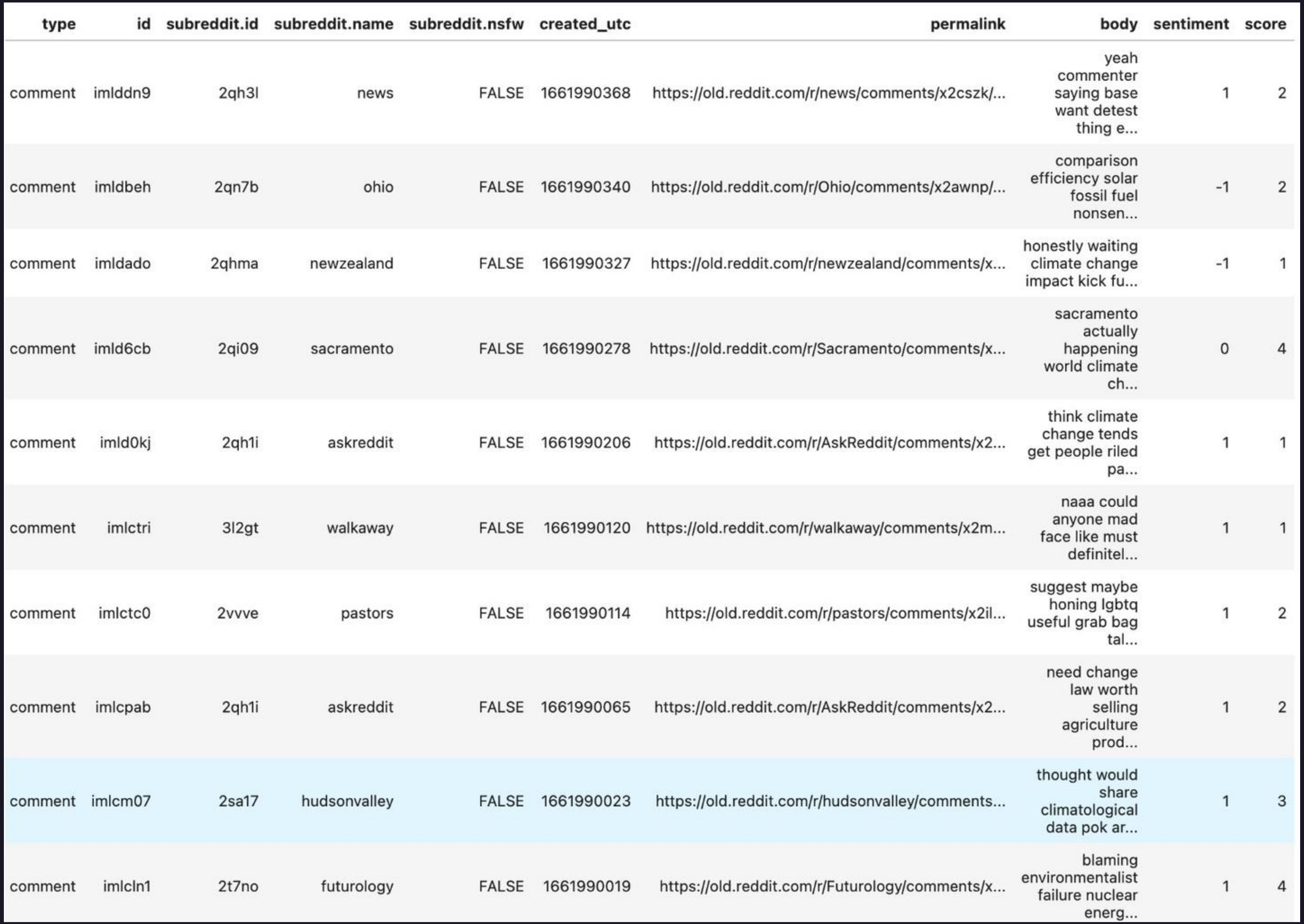
Project Background

Climate change stands as one of the defining challenges of our era, impacting communities globally. To comprehend how the public perceives and engages with this critical issue, our research targets the dynamic landscape of climate change discussions on Reddit. Through advanced Natural Language Processing (NLP), we aim to delve into the sentiments, narratives, and influential voices within these online conversations. As the effects of climate change reverberate worldwide, this project strives to offer nuanced insights into public perspectives, ultimately contributing to data-driven solutions for this pressing global concern. on, addressing critical challenges related to data privacy, real-time processing, and efficiency improvement, while ensuring seamless integration into existing AV systems and reliable operation in diverse automotive environments.

Problem Statement

- Global Challenge: Climate change demands urgent solutions.
- Social Media Role: Key in understanding public views and engagement.
- Research Objective: Analyze Reddit's climate change discussions.
  - Focus Areas
  - Identify key topics and attitudes.
  - Analyze user interactions and engagement.
  - Actionable Insights: Guide effective climate change strategies for policymakers, organizations, and researchers.

DATASET SNIPPET



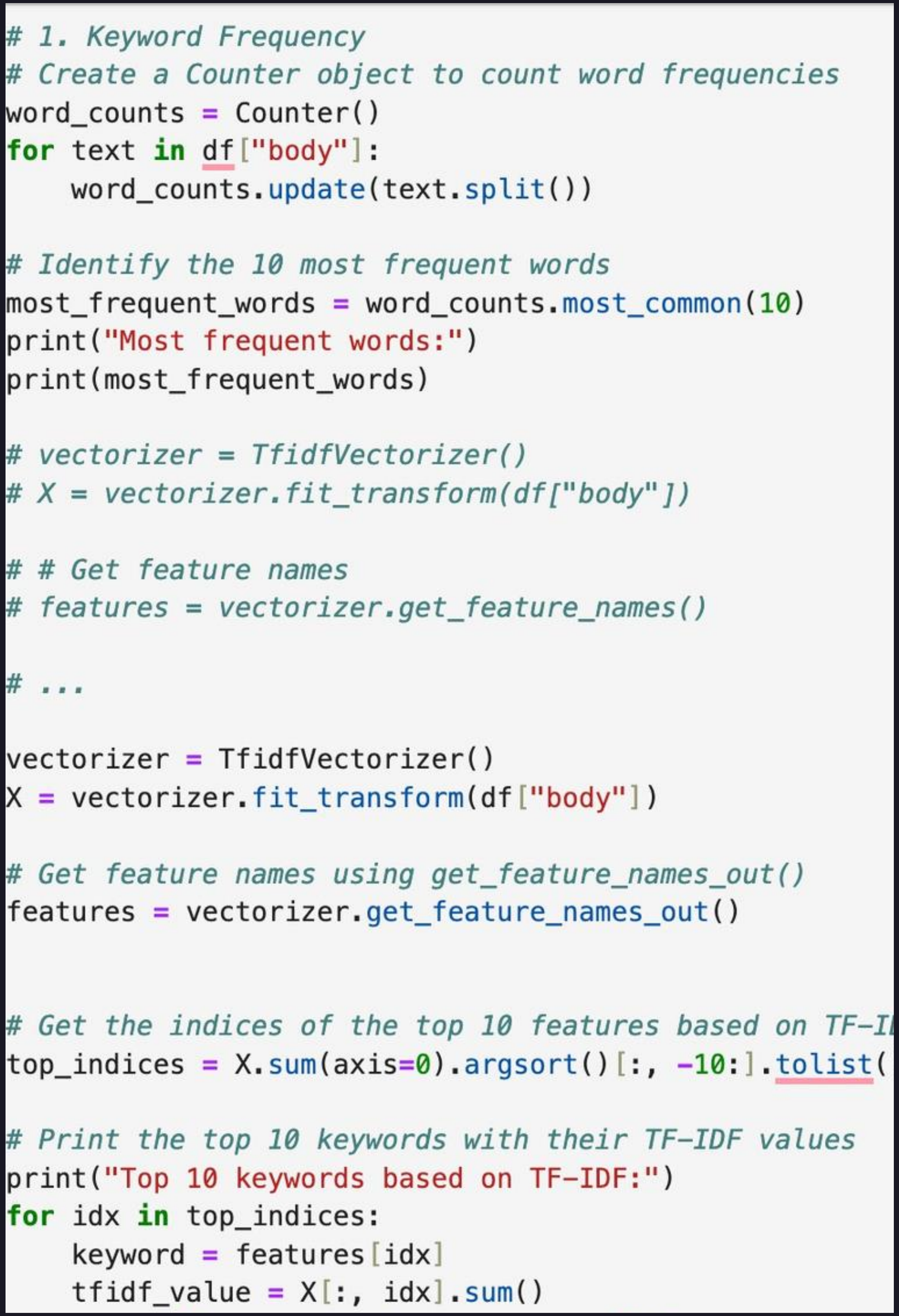
type	id	subreddit.id	subreddit.name	subreddit.nsfw	created_utc	permalink	body	sentiment	score
comment	imlddn9	2qh3l	news	FALSE	1661990368	https://old.reddit.com/r/news/comments/x2cszk/...	yeah commenter saying base want detest thing e...	1	2
comment	imldbeh	2qn7b	ohio	FALSE	1661990340	https://old.reddit.com/r/Ohio/comments/x2awnp/...	comparison efficiency solar fossil fuel nonsen...	-1	2
comment	imldado	2qhma	newzealand	FALSE	1661990327	https://old.reddit.com/r/newzealand/comments/x...	honestly waiting climate change impact kick fu...	-1	1
comment	imld6cb	2qi09	sacramento	FALSE	1661990278	https://old.reddit.com/r/Sacramento/comments/x...	sacramento actually happening world climate ch...	0	4
comment	imld0kj	2qh1i	askreddit	FALSE	1661990206	https://old.reddit.com/r/AskReddit/comments/x2...	think climate change tends get people riled pa...	1	1
comment	imlctri	3l2gt	walkaway	FALSE	1661990120	https://old.reddit.com/r/walkaway/comments/x2m...	naaa could anyone mad face like must definitel...	1	1
comment	imlctc0	2vvve	pastors	FALSE	1661990114	https://old.reddit.com/r/pastors/comments/x2il...	suggest maybe honing lgbtq useful grab bag tal...	1	2
comment	imlcpab	2qh1i	askreddit	FALSE	1661990065	https://old.reddit.com/r/AskReddit/comments/x2...	need change law worth selling agriculture prod...	1	2
comment	imlcm07	2sa17	hudsonvalley	FALSE	1661990023	https://old.reddit.com/r/hudsonvalley/comments...	thought would share climatological data pok ar...	1	3
comment	imlcln1	2t7no	futurology	FALSE	1661990019	https://old.reddit.com/r/Futurology/comments/x...	blaming environmentalist failure nuclear energ...	1	4

Identify Popular Keywords

TF-IDF Overview:(Term Frequency-Inverse Document Frequency.)

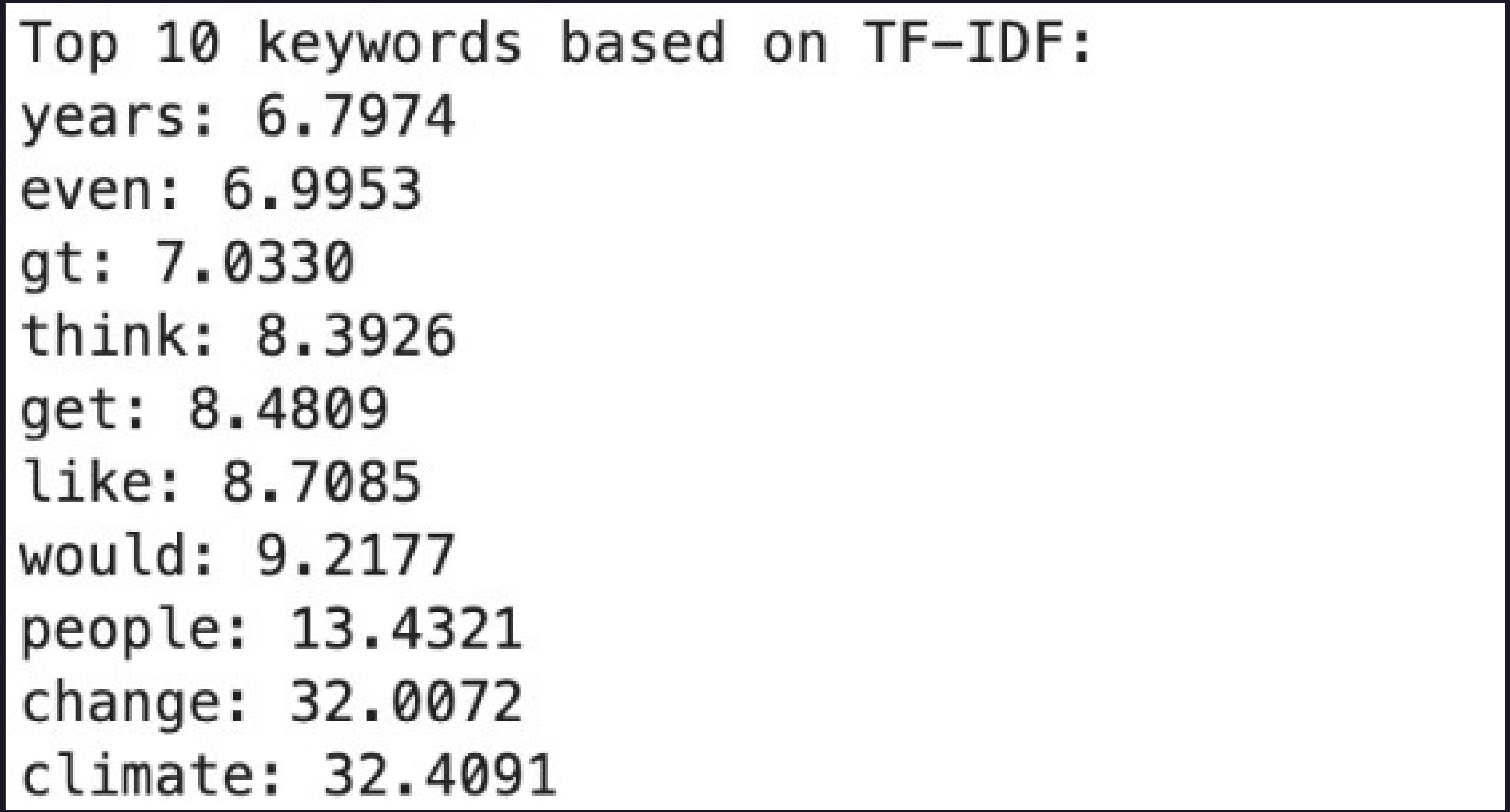
- **Importance** of a word in a document .
- **Conversion of Text to TF-IDF Matrix**:
  - TfidfVectorizer from scikit-learn .
- **Feature Retrieval**:
  - Retrieves the feature names (terms) present in the TF-IDF matrix.
- **Identification of Top Features**:
  - Calculates the sum of TF-IDF values for each term across all documents.
  - Identifies the indices of the top 10 features based on these sums.

Code:



```
# 1. Keyword Frequency
# Create a Counter object to count word frequencies
word_counts = Counter()
for text in df["body"]:
    word_counts.update(text.split())

# Identify the 10 most frequent words
most_frequent_words = word_counts.most_common(10)
print("Most frequent words:")
print(most_frequent_words)

# vectorizer = TfidfVectorizer()
# X = vectorizer.fit_transform(df["body"])

# # Get feature names
# features = vectorizer.get_feature_names()

# ...

vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(df["body"])

# Get feature names using get_feature_names_out()
features = vectorizer.get_feature_names_out()


# Get the indices of the top 10 features based on TF-I
top_indices = X.sum(axis=0).argsort()[:, -10:].tolist(

# Print the top 10 keywords with their TF-IDF values
print("Top 10 keywords based on TF-IDF:")
for idx in top_indices:
    keyword = features[idx]
    tfidf_value = X[:, idx].sum()
```

Output:



```
Top 10 keywords based on TF-IDF:
years: 6.7974
even: 6.9953
gt: 7.0330
think: 8.3926
get: 8.4809
like: 8.7085
would: 9.2177
people: 13.4321
change: 32.0072
climate: 32.4091
```

Key Outcomes

- **Term Importance**:
  - Higher TF-IDF: A higher value indicates that the term is more important in that document relative to the entire corpus.
- **Top Keywords**:
  - Keyword Importance: These top terms are considered the most important keywords in the corpus..
- **Thresholds and Filters**:
  - Setting Thresholds: You can set thresholds to filter out terms with low TF-IDF values, focusing on the most significant terms.
  - Customized Analysis: Helps in customizing the analysis based on your specific criteria and objectives.
- **Application**: Information Retrieval (Search Engines), Content Recommendations.

SENTIMENT ANALYSIS

What is Sentiment Analysis?

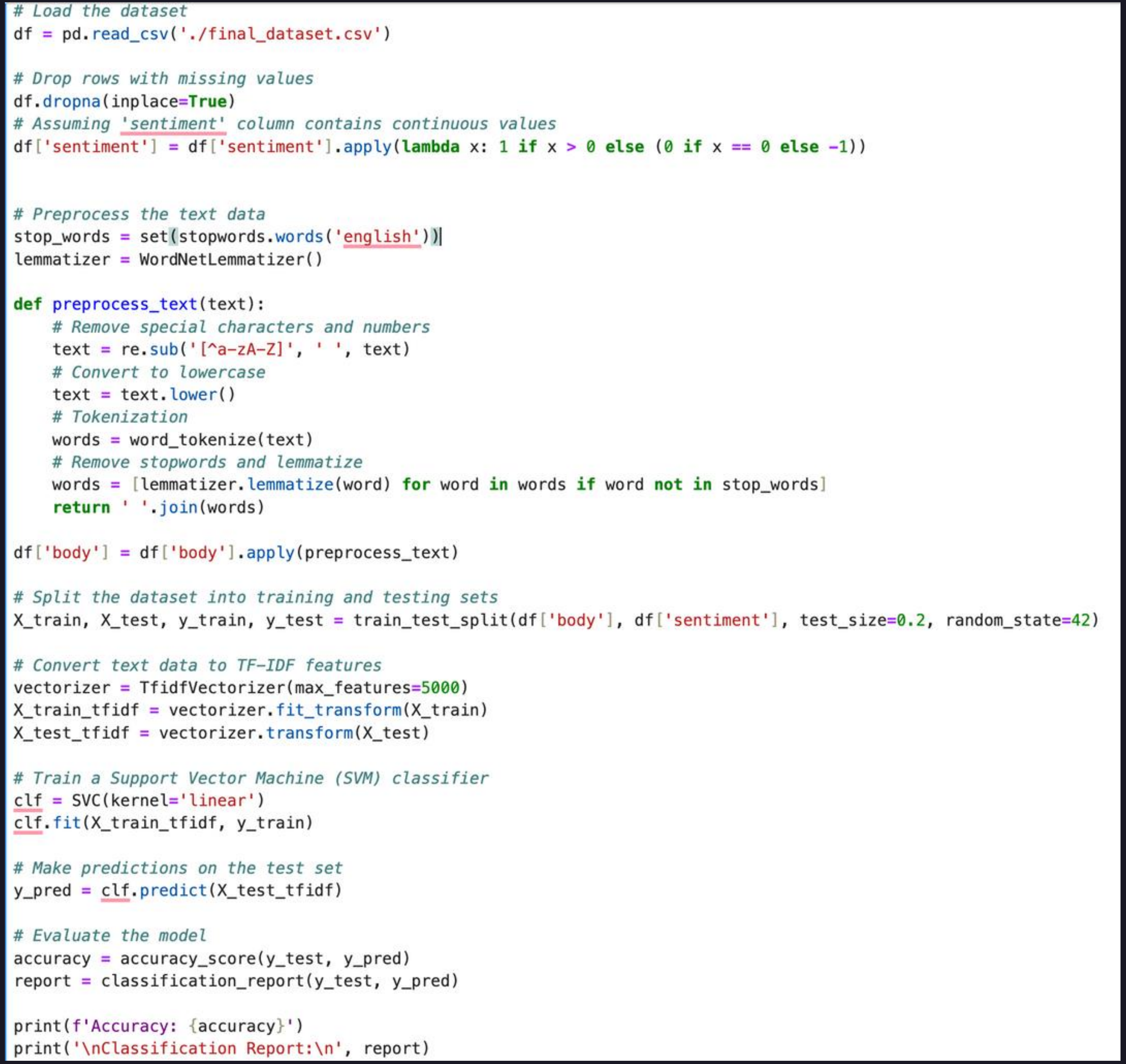
Sentiment analysis is the automated process of gauging opinions and emotions expressed in text to determine if they are positive, negative, or neutral.

Uses of Sentiment Analysis

Opinion Insights
Community Engagement
Sentiment Trends
Decision Support
Brand Perception
Issue Awareness

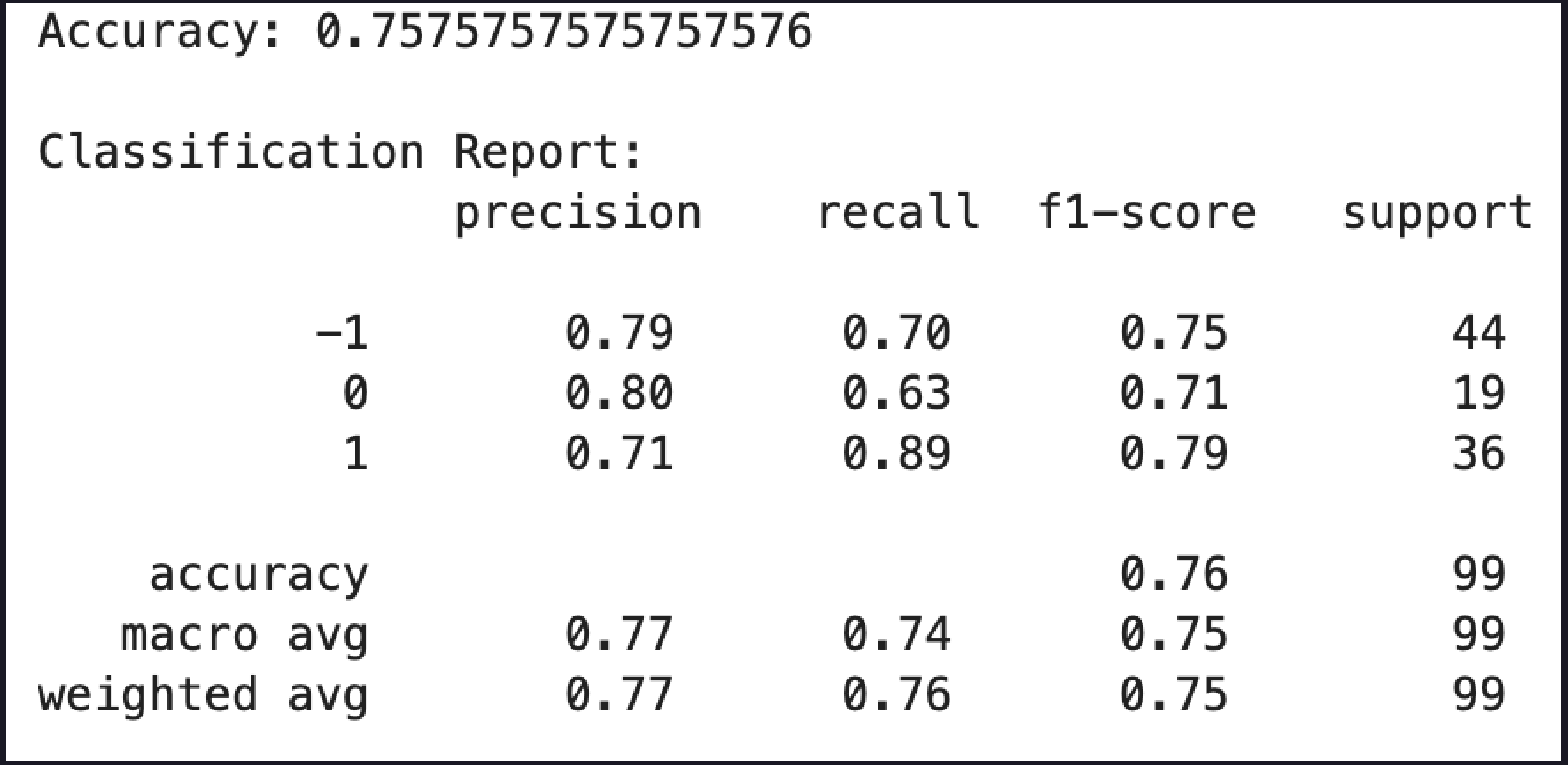
SENTIMENT ANALYSIS - TRAINING

Code:



```
# Load the dataset
df = pd.read_csv('./final_dataset.csv')

# Drop rows with missing values
df.dropna(inplace=True)
# Assuming 'sentiment' column contains continuous values
df['sentiment'] = df['sentiment'].apply(lambda x: 1 if x > 0 else (0 if x == 0 else -1))


# Preprocess the text data
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

def preprocess_text(text):
    # Remove special characters and numbers
    text = re.sub('[^a-zA-Z]', ' ', text)
    # Convert to lowercase
    text = text.lower()
    # Tokenization
    words = word_tokenize(text)
    # Remove stopwords and lemmatize
    words = [lemmatizer.lemmatize(word) for word in words if word not in stop_words]
    return ' '.join(words)

df['body'] = df['body'].apply(preprocess_text)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(df['body'], df['sentiment'], test_size=0.2, random_state=42)

# Convert text data to TF-IDF features
vectorizer = TfidfVectorizer(max_features=5000)
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)

# Train a Support Vector Machine (SVM) classifier
clf = SVC(kernel='linear')
clf.fit(X_train_tfidf, y_train)

# Make predictions on the test set
y_pred = clf.predict(X_test_tfidf)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

print(f'Accuracy: {accuracy}')
print('\nClassification Report:\n', report)
```

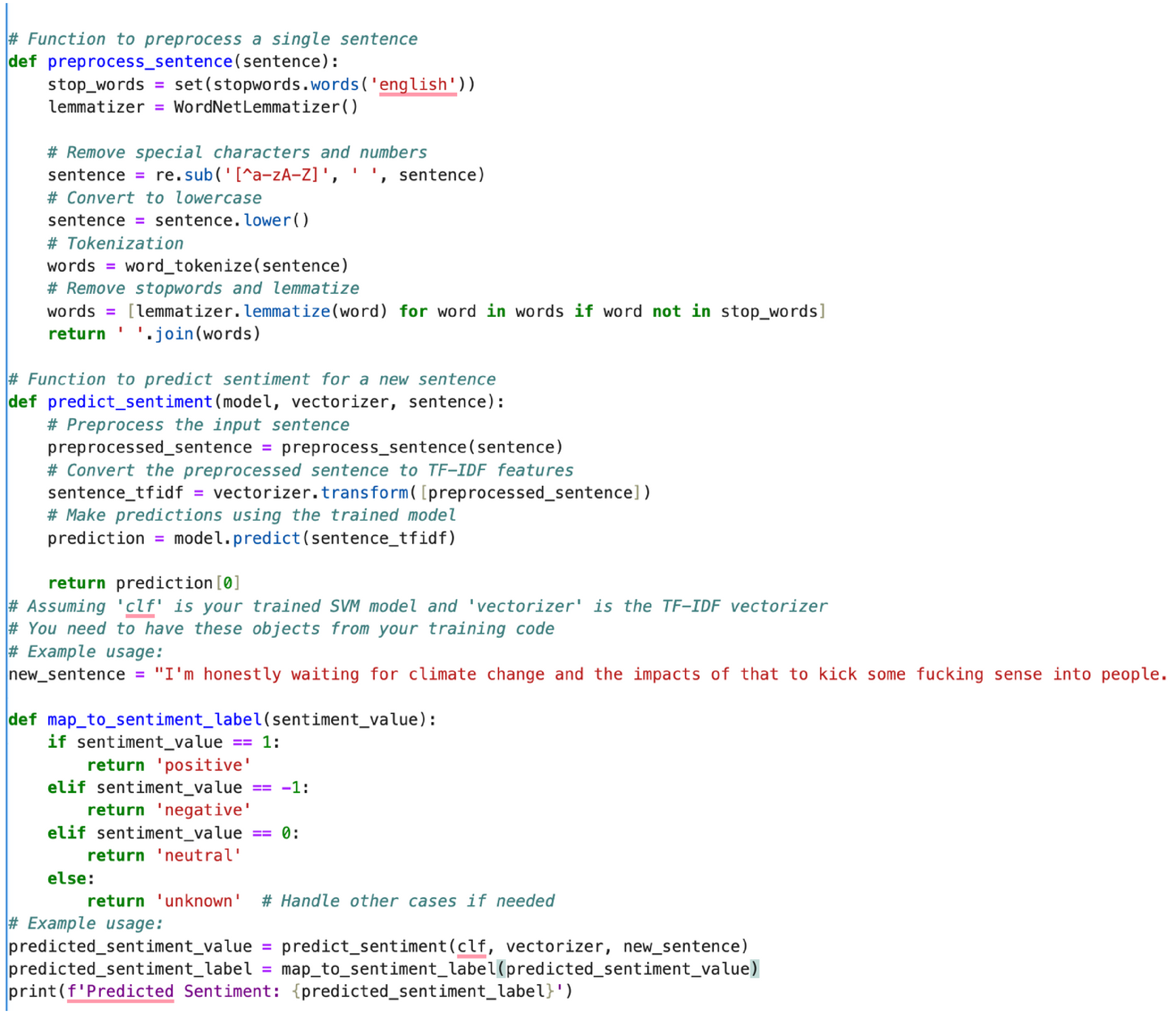
Output:



```
Accuracy: 0.7575757575757576

Classification Report:
               precision    recall  f1-score   support

          -1       0.79      0.70      0.75        44
           0       0.80      0.63      0.71        19
           1       0.71      0.89      0.79        36

    accuracy                           0.76        99
   macro avg       0.77      0.74      0.75        99
weighted avg       0.77      0.76      0.75        99
```

SENTIMENT ANALYSIS PREDICTION

Code:

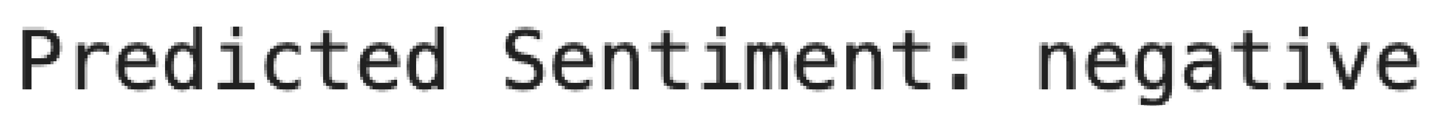


```python
# Function to preprocess a single sentence
def preprocess_sentence(sentence):
    stop_words = set(stopwords.words('english'))
    lemmatizer = WordNetLemmatizer()

    # Remove special characters and numbers
    sentence = re.sub('[^a-zA-Z]', ' ', sentence)
    # Convert to lowercase
    sentence = sentence.lower()
    # Tokenization
    words = word_tokenize(sentence)
    # Remove stopwords and lemmatize
    words = [lemmatizer.lemmatize(word) for word in words if word not in stop_words]
    return ' '.join(words)

# Function to predict sentiment for a new sentence
def predict_sentiment(model, vectorizer, sentence):
    # Preprocess the input sentence
    preprocessed_sentence = preprocess_sentence(sentence)
    # Convert the preprocessed sentence to TF-IDF features
    sentence_tfidf = vectorizer.transform([preprocessed_sentence])
    # Make predictions using the trained model
    prediction = model.predict(sentence_tfidf)

    return prediction[0]
# Assuming 'clf' is your trained SVM model and 'vectorizer' is the TF-IDF vectorizer
# You need to have these objects from your training code
# Example usage:
new_sentence = "I'm honestly waiting for climate change and the impacts of that to kick some fucking sense into people.

def map_to_sentiment_label(sentiment_value):
    if sentiment_value == 1:
        return 'positive'
    elif sentiment_value == -1:
        return 'negative'
    elif sentiment_value == 0:
        return 'neutral'
    else:
        return 'unknown'  # Handle other cases if needed
# Example usage:
predicted_sentiment_value = predict_sentiment(clf, vectorizer, new_sentence)
predicted_sentiment_label = map_to_sentiment_label(predicted_sentiment_value)
print(f'Predicted Sentiment: {predicted_sentiment_label}')
```

Sentence

Sentence - Im honestly waiting for climate change and the impacts of that to kick some fucking sense into people. But who am I kidding it'll still just be more of the poor suffering while the rich claim victim hood for handouts while letting us all starve. Its honestly hard some days to not just give up, and I truly wonder if and when anything will ever actually be done.

Output:



```
Predicted Sentiment: negative
```

MISINFORMATION PREDICTION - USES

Community Moderation:

- Predicting and identifying climate change misinformation in Reddit comments to assist community moderators in maintaining accurate and reliable information within climate-related discussions.

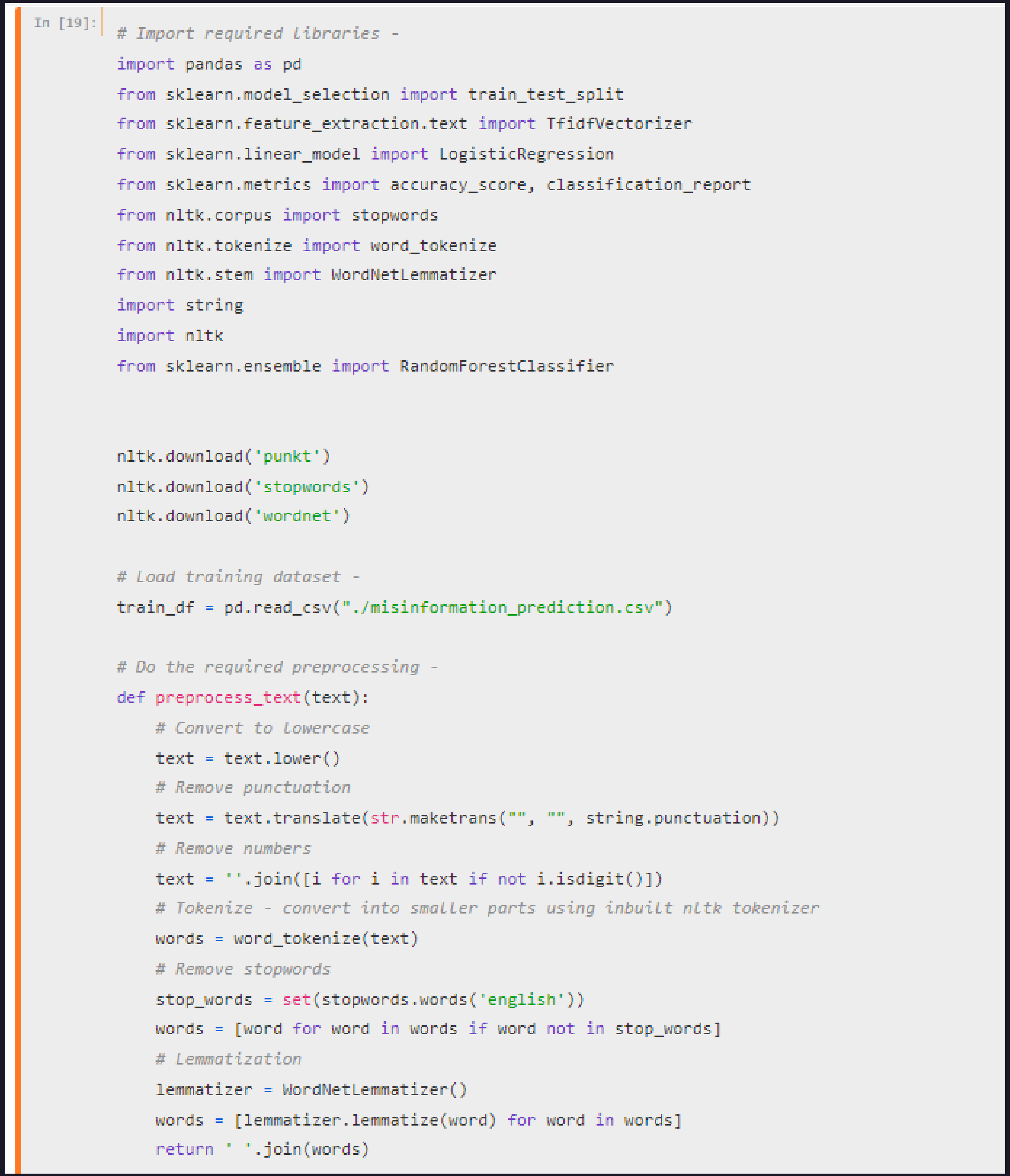
Educational Outreach:

- Utilizing misinformation prediction to target and correct false information in Reddit comments, particularly in educational or informational threads, enhances the overall quality of climate change discussions.

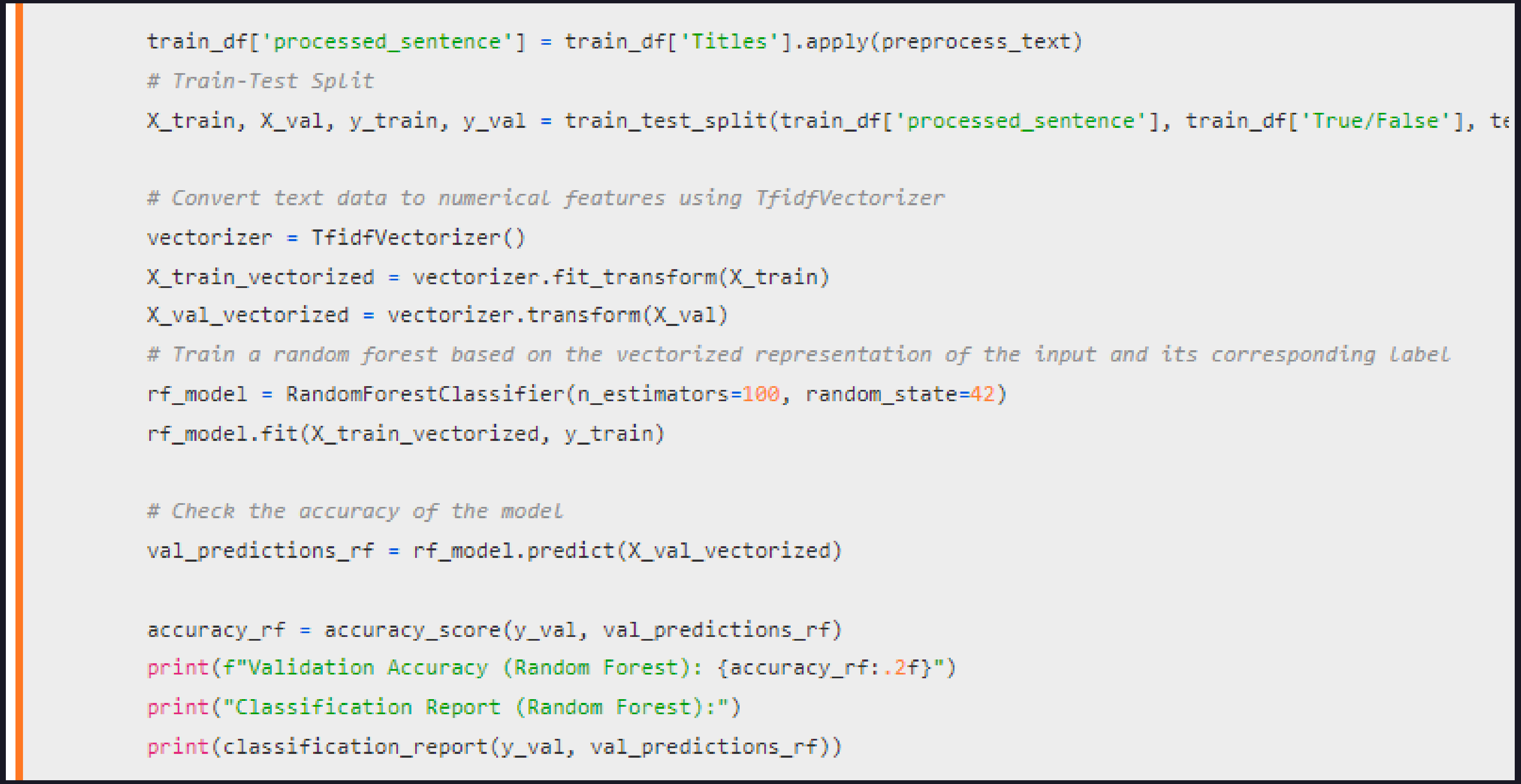
Algorithmic Content Filtering:

- Integrating misinformation prediction into algorithms that filter and rank Reddit comments, helping to reduce the visibility of misleading information and promote more accurate content.

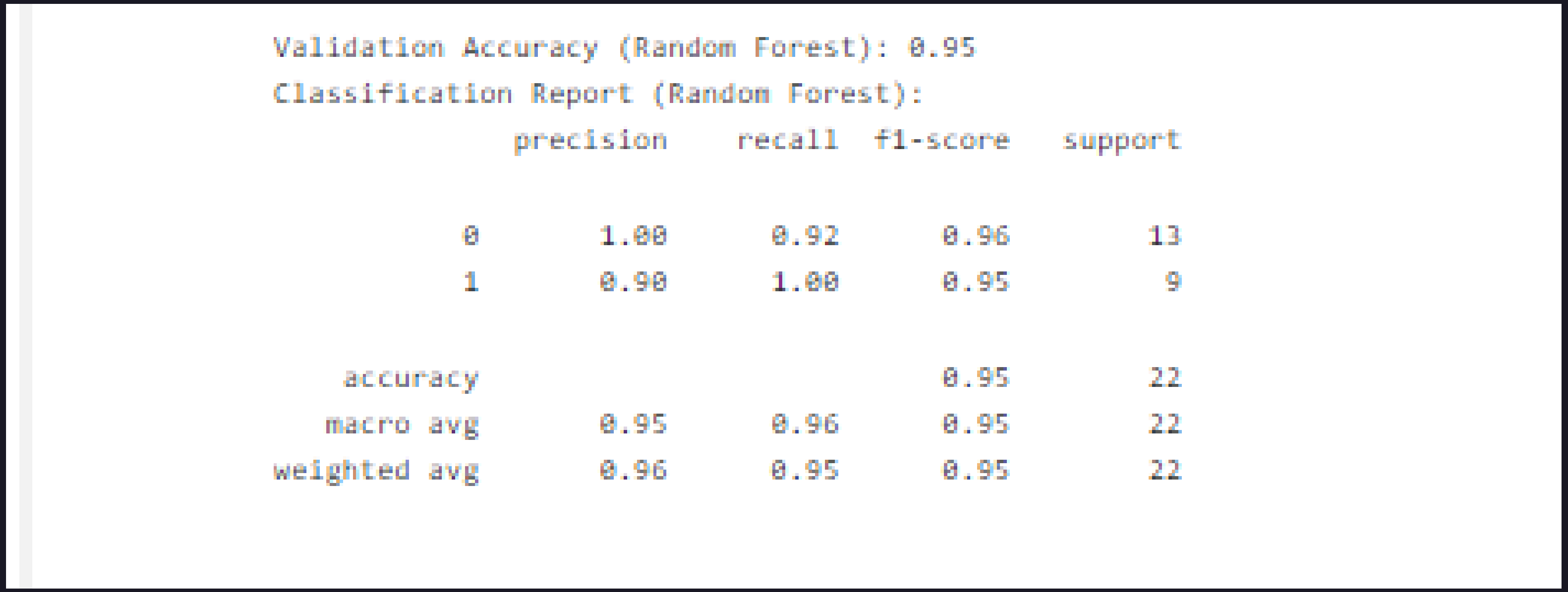
MISINFORMATION PREDICTION - TRAINING



```
In [19]:  # Import required libraries -
          import pandas as pd
          from sklearn.model_selection import train_test_split
          from sklearn.feature_extraction.text import TfidfVectorizer
          from sklearn.linear_model import LogisticRegression
          from sklearn.metrics import accuracy_score, classification_report
          from nltk.corpus import stopwords
          from nltk.tokenize import word_tokenize
          from nltk.stem import WordNetLemmatizer
          import string
          import nltk
          from sklearn.ensemble import RandomForestClassifier


          nltk.download('punkt')
          nltk.download('stopwords')
          nltk.download('wordnet')

          # Load training dataset -
          train_df = pd.read_csv("./misinformation_prediction.csv")

          # Do the required preprocessing -
          def preprocess_text(text):
              # Convert to lowercase
              text = text.lower()
              # Remove punctuation
              text = text.translate(str.maketrans("", "", string.punctuation))
              # Remove numbers
              text = ''.join([i for i in text if not i.isdigit()])
              # Tokenize - convert into smaller parts using inbuilt nltk tokenizer
              words = word_tokenize(text)
              # Remove stopwords
              stop_words = set(stopwords.words('english'))
              words = [word for word in words if word not in stop_words]
              # Lemmatization
              lemmatizer = WordNetLemmatizer()
              words = [lemmatizer.lemmatize(word) for word in words]
              return ' '.join(words)
```

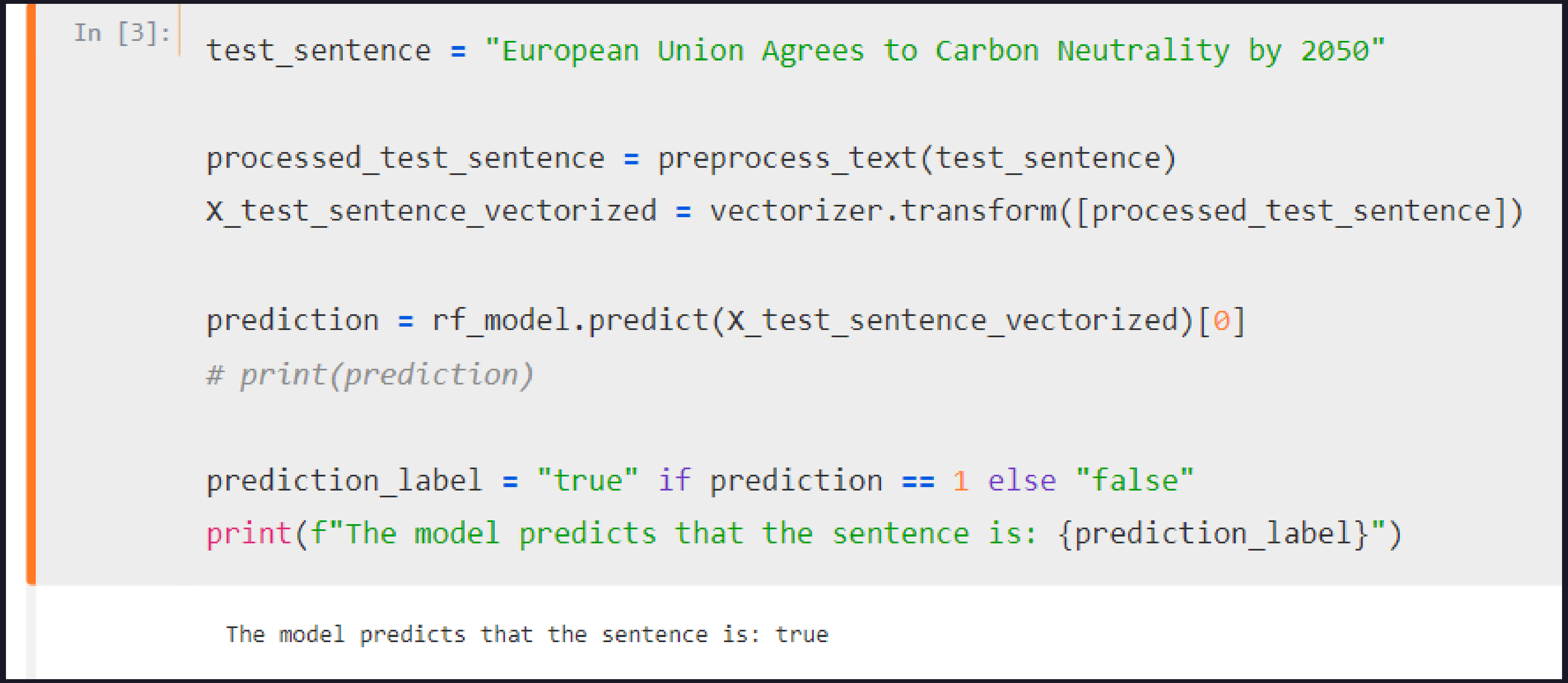


```
train_df['processed_sentence'] = train_df['Titles'].apply(preprocess_text)
# Train-Test Split
X_train, X_val, y_train, y_val = train_test_split(train_df['processed_sentence'], train_df['True/False'], te

# Convert text data to numerical features using TfidfVectorizer
vectorizer = TfidfVectorizer()
X_train_vectorized = vectorizer.fit_transform(X_train)
X_val_vectorized = vectorizer.transform(X_val)
# Train a random forest based on the vectorized representation of the input and its corresponding label
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train_vectorized, y_train)

# Check the accuracy of the model
val_predictions_rf = rf_model.predict(X_val_vectorized)

accuracy_rf = accuracy_score(y_val, val_predictions_rf)
print(f"Validation Accuracy (Random Forest): {accuracy_rf:.2f}")
print("Classification Report (Random Forest):")
print(classification_report(y_val, val_predictions_rf))
```

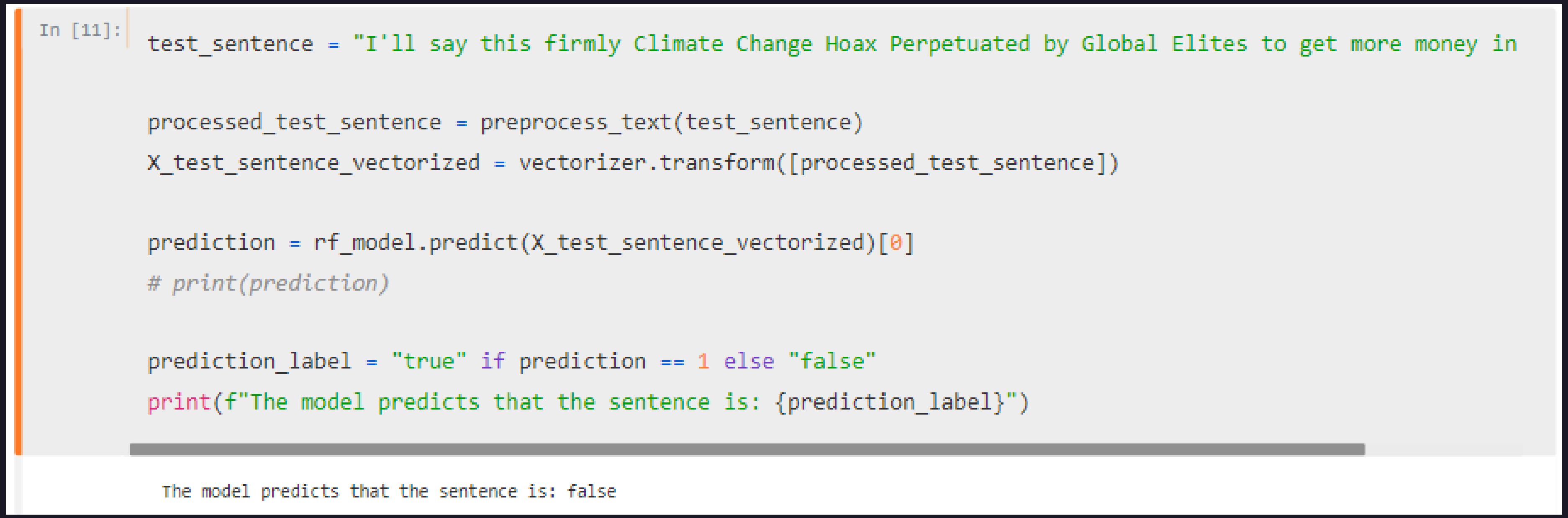


```
Validation Accuracy (Random Forest): 0.95
Classification Report (Random Forest):
              precision    recall  f1-score   support

           0       1.00      0.92      0.96        13
           1       0.90      1.00      0.95         9

    accuracy                           0.95        22
   macro avg       0.95      0.96      0.95        22
weighted avg       0.96      0.95      0.95        22
```

MISINFORMATION PREDICTION - OUTPUT



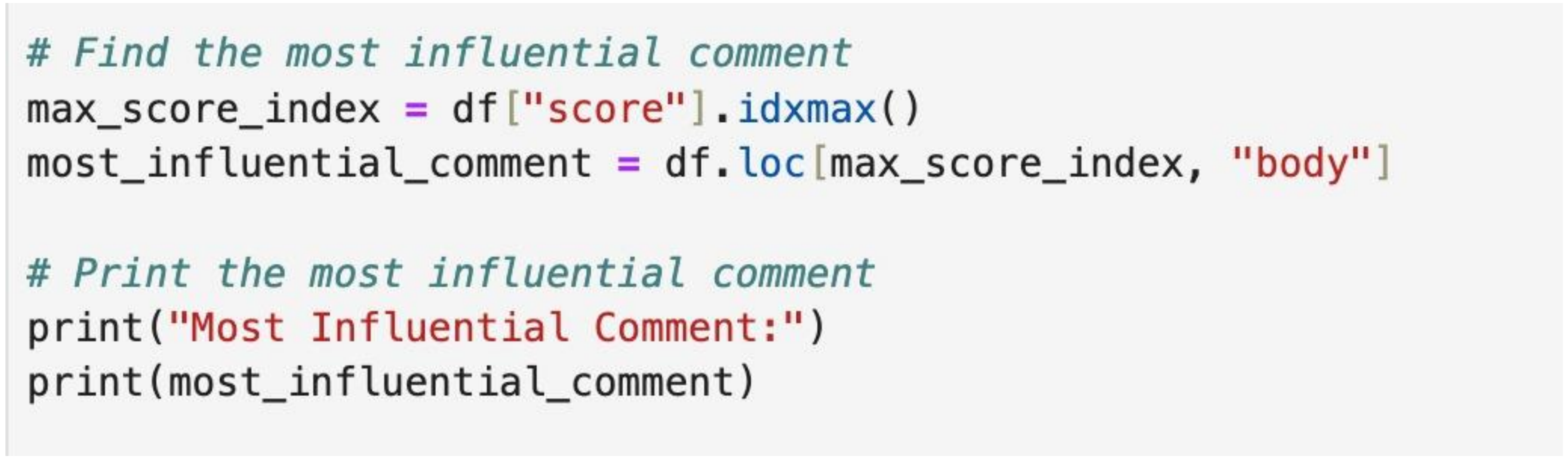
```
In [3]: test_sentence = "European Union Agrees to Carbon Neutrality by 2050"

        processed_test_sentence = preprocess_text(test_sentence)
        X_test_sentence_vectorized = vectorizer.transform([processed_test_sentence])

        prediction = rf_model.predict(X_test_sentence_vectorized)[0]
        # print(prediction)

        prediction_label = "true" if prediction == 1 else "false"
        print(f"The model predicts that the sentence is: {prediction_label}")

        The model predicts that the sentence is: true
```



```
In [11]: test_sentence = "I'll say this firmly Climate Change Hoax Perpetuated by Global Elites to get more money in

         processed_test_sentence = preprocess_text(test_sentence)
         X_test_sentence_vectorized = vectorizer.transform([processed_test_sentence])

         prediction = rf_model.predict(X_test_sentence_vectorized)[0]
         # print(prediction)

         prediction_label = "true" if prediction == 1 else "false"
         print(f"The model predicts that the sentence is: {prediction_label}")

         The model predicts that the sentence is: false
```

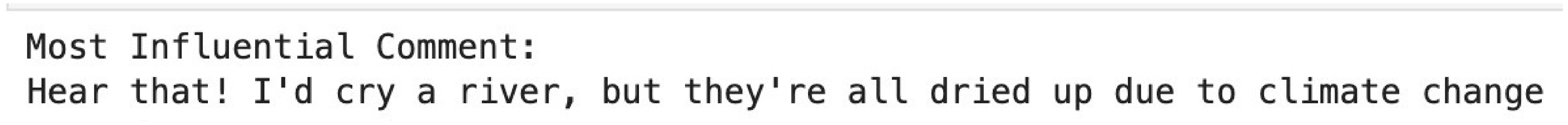
Most Influential Comment

- Stopwords:
  - Stopwords are common, non-informative words (e.g., "the," "and") removed during text analysis to focus on more meaningful content words
- Score Analysis:
  - Score is calculated based on the number of upvotes and downvotes for each comment.
  - Identifies the index of the row with the maximum value in the "score" column



```python
# Find the most influential comment
max_score_index = df["score"].idxmax()
most_influential_comment = df.loc[max_score_index, "body"]

# Print the most influential comment
print("Most Influential Comment:")
print(most_influential_comment)
```

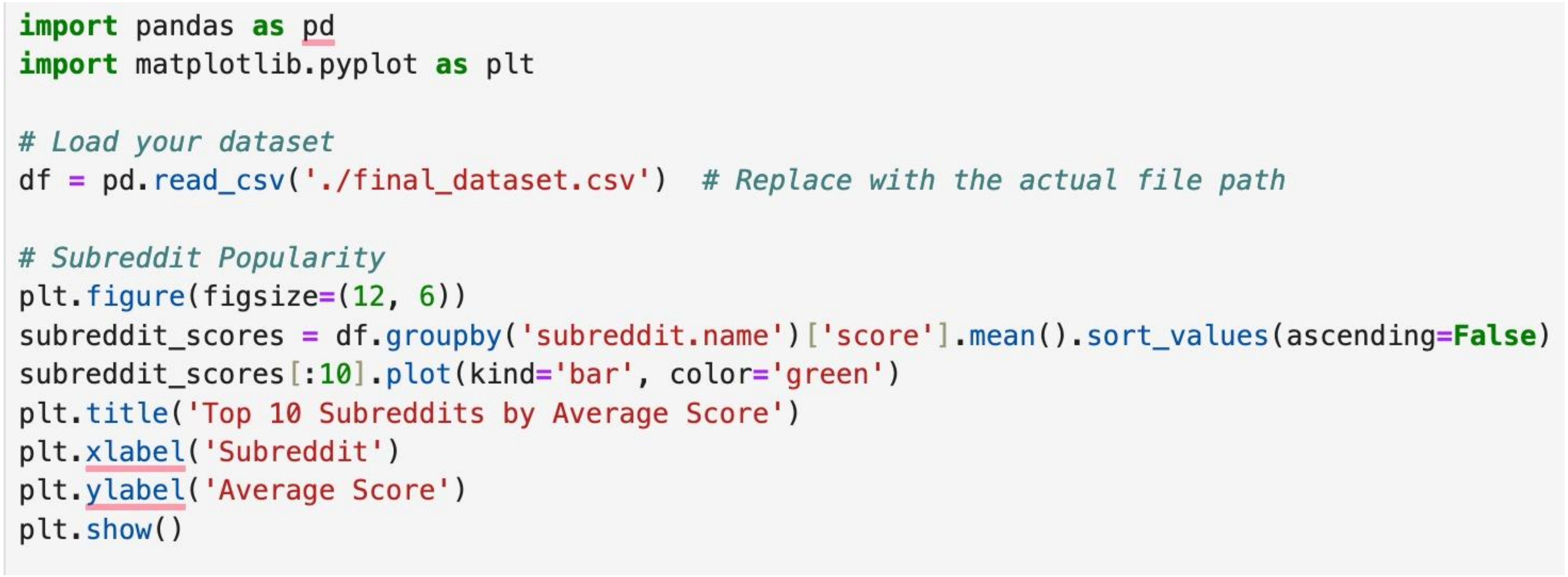
Output:



```
Most Influential Comment:
Hear that! I'd cry a river, but they're all dried up due to climate change
```

Subreddit Popularity Analysis

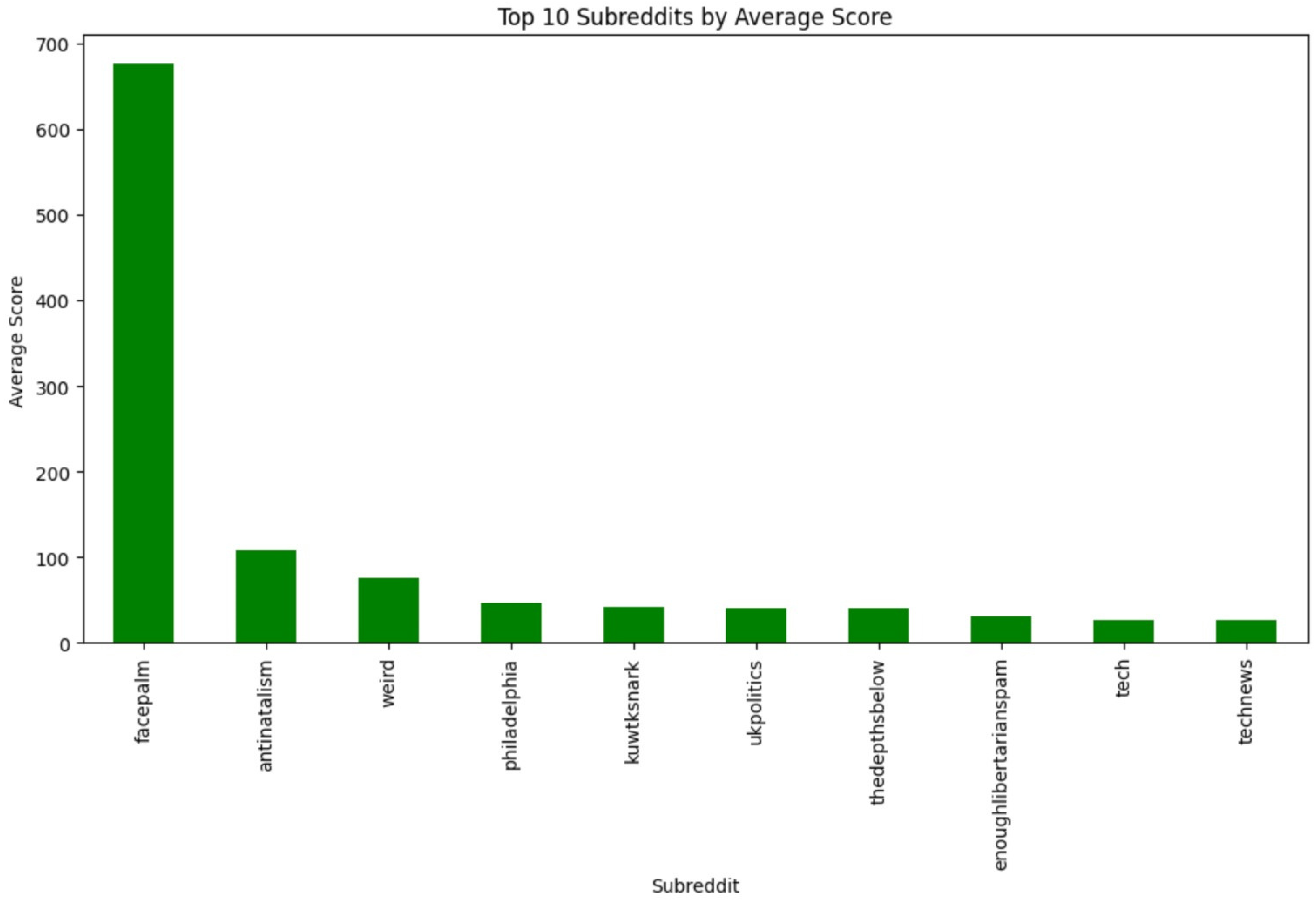
- Created a bar chart to visualize the top 10 subreddits based on the average score of their posts
- Grouped the DataFrame 'df' by the 'subreddit.name' column and calculates the mean score for each subreddit
- Sorted the subreddits in descending order based on the average score
- Selected the top 10 subreddits and plots them as a bar chart in green



```python
import pandas as pd
import matplotlib.pyplot as plt

# Load your dataset
df = pd.read_csv('./final_dataset.csv')  # Replace with the actual file path

# Subreddit Popularity
plt.figure(figsize=(12, 6))
subreddit_scores = df.groupby('subreddit.name')['score'].mean().sort_values(ascending=False)
subreddit_scores[:10].plot(kind='bar', color='green')
plt.title('Top 10 Subreddits by Average Score')
plt.xlabel('Subreddit')
plt.ylabel('Average Score')
plt.show()
```

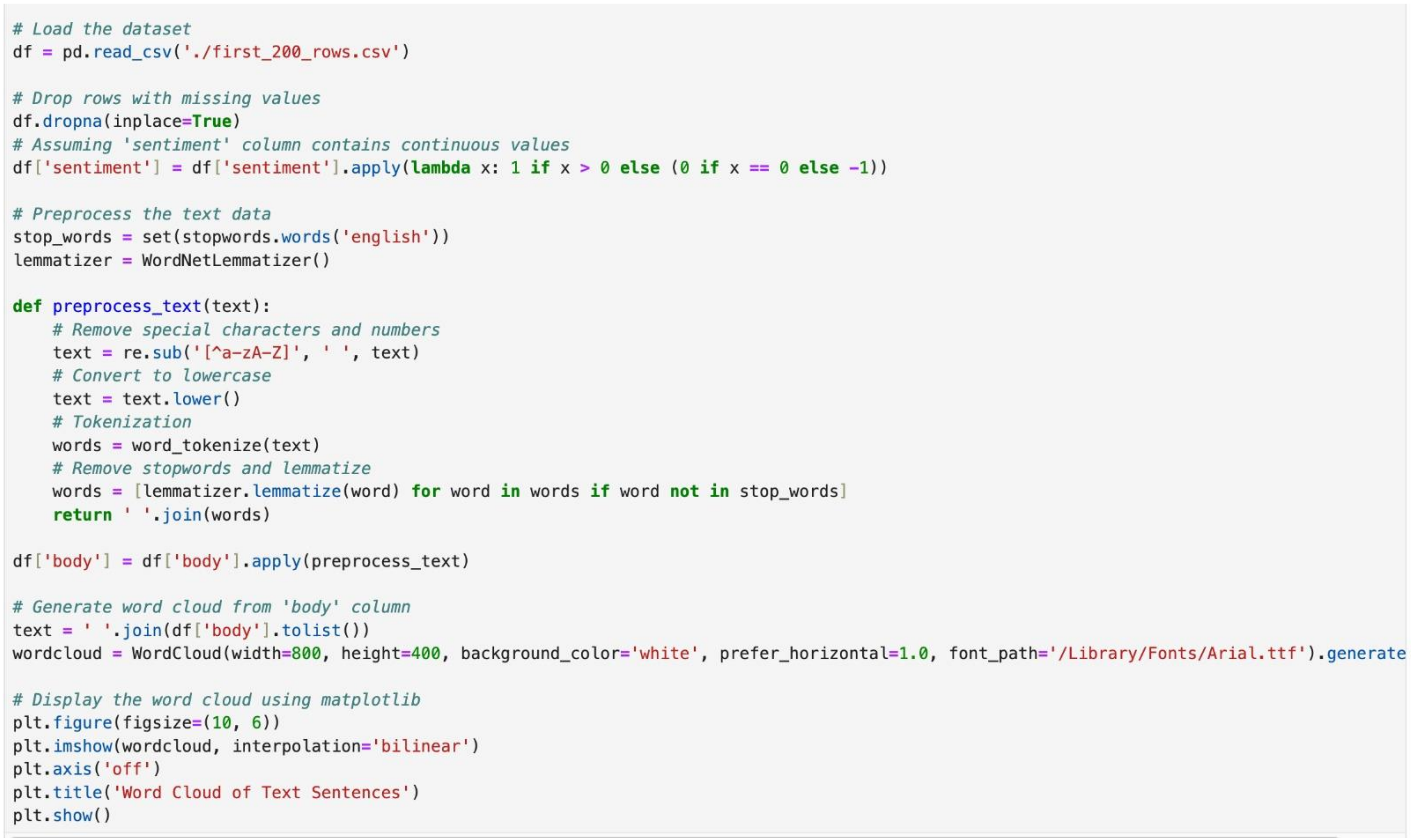
Subreddit Popularity Analysis

Output:



Word Cloud of Text Sentences

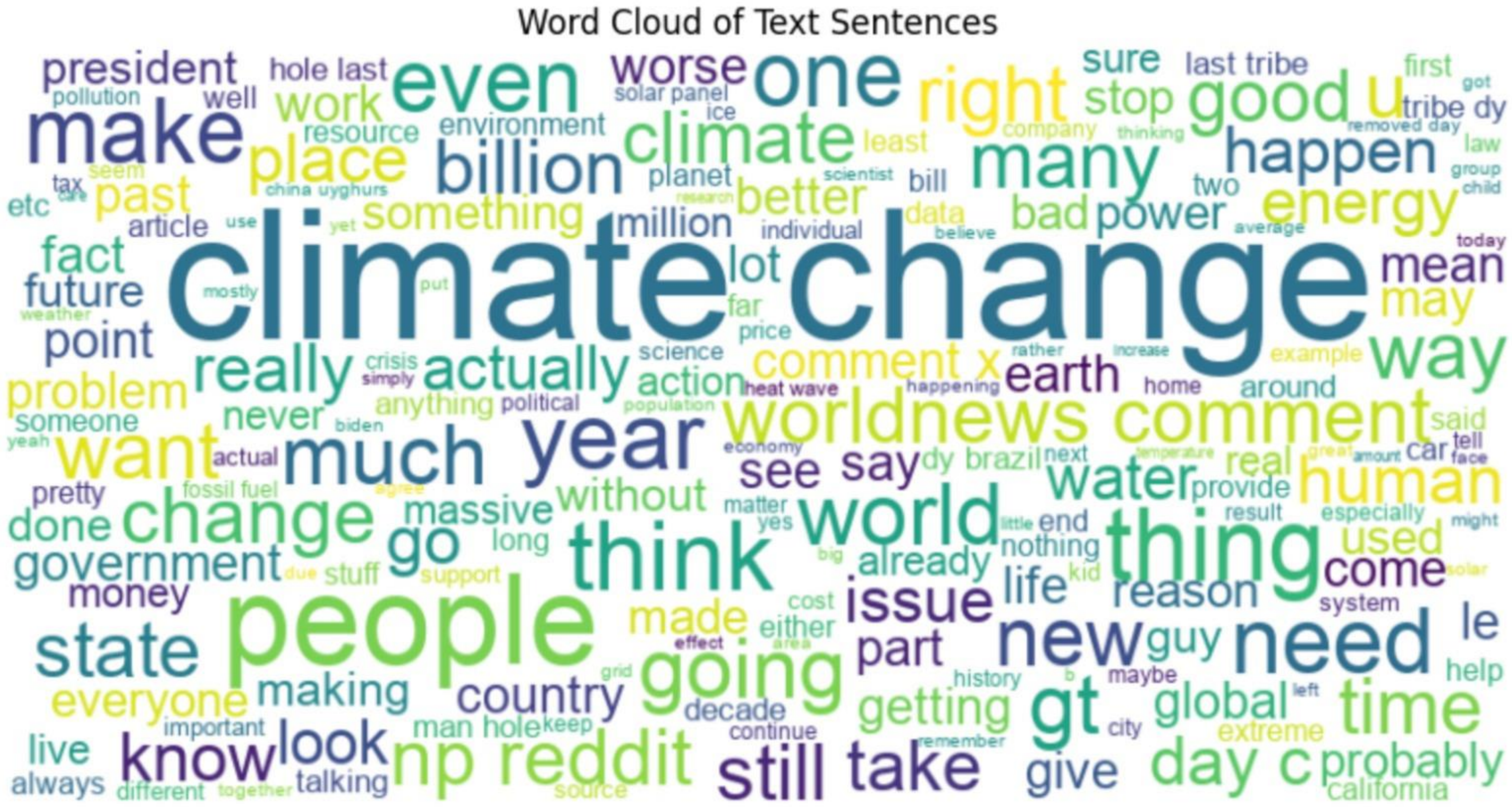
- Word Cloud: Visual representation of text where words are sized based on their frequency
- Word Size: Larger words indicate higher frequency in the given text
- Layout: Words are arranged in an aesthetically pleasing and often random manner
- Color: Words may be displayed in various colors for visual appeal
- Usage: Commonly used in data exploration, sentiment analysis, and summarization of text data



```python
# Load the dataset
df = pd.read_csv('./first_200_rows.csv')

# Drop rows with missing values
df.dropna(inplace=True)
# Assuming 'sentiment' column contains continuous values
df['sentiment'] = df['sentiment'].apply(lambda x: 1 if x > 0 else (0 if x == 0 else -1))

# Preprocess the text data
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

def preprocess_text(text):
    # Remove special characters and numbers
    text = re.sub('[^a-zA-Z]', ' ', text)
    # Convert to lowercase
    text = text.lower()
    # Tokenization
    words = word_tokenize(text)
    # Remove stopwords and lemmatize
    words = [lemmatizer.lemmatize(word) for word in words if word not in stop_words]
    return ' '.join(words)

df['body'] = df['body'].apply(preprocess_text)

# Generate word cloud from 'body' column
text = ' '.join(df['body'].tolist())
wordcloud = WordCloud(width=800, height=400, background_color='white', prefer_horizontal=1.0, font_path='/Library/Fonts/Arial.ttf').generate

# Display the word cloud using matplotlib
plt.figure(figsize=(10, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud of Text Sentences')
plt.show()
```

Word Cloud of Text Sentences

Output:



CONCLUSION

Our project delves into Reddit conversations on climate change, employing advanced tools to grasp public sentiments, tackle misinformation, and empower policymakers and researchers with data-driven strategies—a step toward fostering a more informed and engaged public in the face of the global challenge that is climate change.

Thank You!