

November 30, 2023

1 Yelp data set

This is a sizable community with information on restaurants, businesses, and orders. I have looked over the documentation and by inspecting through one of the results, to determine what kind of dataframes are required for restaurant reviews. We made use of the Google Place API for that investigation.

```
[1]: Links_1 = 'https://www.yelp.com/search?find_desc=Restaurants&find_loc=Dallas_TX'
```

2 Google API-Calling

This is real-time data that may be used to gather data regarding reviews of the API's operation. When a user submits a review for a specific restaurant at a certain location, we can get the data.

API Key: AIzaSyDrfn_jcBBed7k26D27cOQpN0bMf2jWXZE

```
[25]: !pip install requests
```

```
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-
packages (2.31.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests) (2023.7.22)
```

3 Data Collection

We are data from Google place using a dedicated API provided by Google to get real-time data on every API call. Source: <https://developers.google.com/maps/documentation/places/web-service/overview>

```
[26]: import requests
import pandas as pd
```

```

# API key
api_key = "AIzaSyDrfn_jcBBed7k26D27c0QpN0bMf2jWXZE"

# Function to get data based on location(dallas) using Google Places API
def get_restaurants_in_location(location):
    url = f"https://maps.googleapis.com/maps/api/place/textsearch/json?
    query=restaurants+in{location}&key={api_key}"
    response = requests.get(url)
    if response.status_code == 200:
        return response.json().get('results', [])
    return []

# Data for Dallas
dallas_restaurants = get_restaurants_in_location("Dallas")

# Create a DataFrame from fetched restaurant data
restaurant_df = pd.DataFrame(dallas_restaurants)

# Display data (Observed from yelp datasets on reviews)
for index, restaurant in restaurant_df.iterrows():
    print("Name:", restaurant['name'])
    print("Address:", restaurant['formatted_address'])
    print("Rating:", restaurant['rating'])
    print("-----")

```

```

Name: The Mansion Restaurant
Address: 2821 Turtle Creek Blvd, Dallas, TX 75219, United States
Rating: 4.6
-----
Name: Fearing's Restaurant
Address: 2121 McKinney Ave, Dallas, TX 75201, United States
Rating: 4.6
-----
Name: Knife
Address: 5300 E Mockingbird Ln, Dallas, TX 75206, United States
Rating: 4.4
-----
Name: The Exchange Hall
Address: 211 S Akard St, Dallas, TX 75202, United States
Rating: 4.6
-----
Name: Norma's Cafe
Address: 9100 N Central Expy #151, Dallas, TX 75231, United States
Rating: 4.3
-----
Name: Elm & Good
Address: 2551 Elm St, Dallas, TX 75226, United States

```

Rating: 4.1

Name: Garden Cafe

Address: 5310 Junius St, Dallas, TX 75214, United States

Rating: 4.5

Name: Porta Di Roma

Address: 1623 Main St Ste 104, Dallas, TX 75201, United States

Rating: 4.0

Name: Desta Ethiopian Restaurant

Address: 12101 Greenville Ave #105, Dallas, TX 75243, United States

Rating: 4.4

Name: Good Graces

Address: 3033 Fairmount St, Dallas, TX 75201, United States

Rating: 4.2

Name: Asador

Address: 2222 N Stemmons Fwy, Dallas, TX 75207, United States

Rating: 4.3

Name: Ellie's Restaurant & Lounge

Address: 1717 Leonard St, Dallas, TX 75201, United States

Rating: 4.3

Name: Bubba's Cooks Country - Dallas

Address: 6617 Hillcrest Ave, Dallas, TX 75205, United States

Rating: 4.6

Name: Margaret's

Address: 800 N Harwood St, Dallas, TX 75201, United States

Rating: 4.5

Name: Eataly Dallas

Address: 8687 N Central Expy #2172, Dallas, TX 75225, United States

Rating: 4.3

Name: 4511 Restaurant Group

Address: 1999 Bryan St #3300, Dallas, TX 75201, United States

Rating: 1.0

Name: Yolk - One Arts Plaza

Address: 1722 Routh St #132, Dallas, TX 75201, United States

Rating: 4.3

Name: RH Rooftop Restaurant Dallas

Address: 3133 Knox St, Dallas, TX 75205, United States

Rating: 4.1

Name: True kitchen &cocktails

Address: 1933 Elm St, Dallas, TX 75201, United States

Rating: 4.0

Name: Mammass House

Address: 2117 Commerce St, Dallas, TX 75201, United States

Rating: 0.0

- Filtering Restaurants (Extreme Records i.e Top rated and Least Rated)

```
[4]: # Find top-rated say 4.8
top Rated = restaurant_df[restaurant_df['rating'] == 4.8]
print("Top-rated restaurant(s) with a rating of 4.8:")
print(top Rated)

# Find least-rated say 3.7
least Rated = restaurant_df[restaurant_df['rating'] == 3.7]
print("\nLeast-rated restaurant(s) with a rating of 3.7:")
print(least Rated)
```

Top-rated restaurant(s) with a rating of 4.8:

```
business_status formatted_address \
19 OPERATIONAL 4152 Cole Ave # 103, Dallas, TX 75204, United ...

geometry \
19 {'location': {'lat': 32.8161232, 'lng': -96.79...

icon icon_background_color \
19 https://maps.gstatic.com/mapfiles/place_api/ic... #FF9E67

icon_mask_base_uri name \
19 https://maps.gstatic.com/mapfiles/place_api/ic... Salum Restaurant

opening_hours photos \
19 {'open_now': False} [{'height': 1080, 'html_attributions': ['<a hr...

place_id \
19 ChIJpdrv_9eeToYREr8Zm9DlXZ0

plus_code price_level rating \
19 {'compound_code': 'R685+C4 Dallas, Texas', 'gl... 3.0 4.8

reference \
19 ChIJpdrv_9eeToYREr8Zm9DlXZ0
```

```

                                types  user_ratings_total
19  [bar, restaurant, food, point_of_interest, est...      289

```

Least-rated restaurant(s) with a rating of 3.7:

Empty DataFrame

Columns: [business_status, formatted_address, geometry, icon, icon_background_color, icon_mask_base_uri, name, opening_hours, photos, place_id, plus_code, price_level, rating, reference, types, user_ratings_total]
Index: []

```

[5]: import pandas as pd

# DataFrame to store restaurant information
restaurant_data = []
for restaurant in dallas_restaurants:
    restaurant_info = {
        'Name': restaurant.get('name'),
        'Address': restaurant.get('formatted_address'),
        'Rating': restaurant.get('rating')
    }
    restaurant_data.append(restaurant_info)

restaurant_df = pd.DataFrame(restaurant_data)

# Saving to CSV file
csv_file_path = 'dallas_restaurant_info.csv'
restaurant_df.to_csv(csv_file_path, index=False)

print(f"Restaurant information saved to '{csv_file_path}'")

```

Restaurant information saved to 'dallas_restaurant_info.csv'

Visualizations for Raw Data - Overall Restaurant Results (Raw data/ before cleaning) helps to compare when once it was processed into a Filtered data or the one without Errors.

```

[6]: import matplotlib.pyplot as plt

# List of restaurants
restaurants = [
    "The Henry", "North Italia", "The Capital Grille", "Ocean Prime", "Doc B's",
    ↪ "Restaurant",
    "Dick's Last Resort - Dallas", "Meso Maya Comida y Copas", "Fearing's",
    ↪ "Restaurant",
    "Original Market Diner", "Salum Restaurant", "The Mansion Restaurant",
    ↪ "Moxies Dallas Uptown Restaurant",
    "Y.O. Ranch Steakhouse", "Dragonfly", "Campisi's", "Ellen's", "Truluck's",
    ↪ "Sloane's Corner",
    "Cafe Pacific", "Sachet"
]

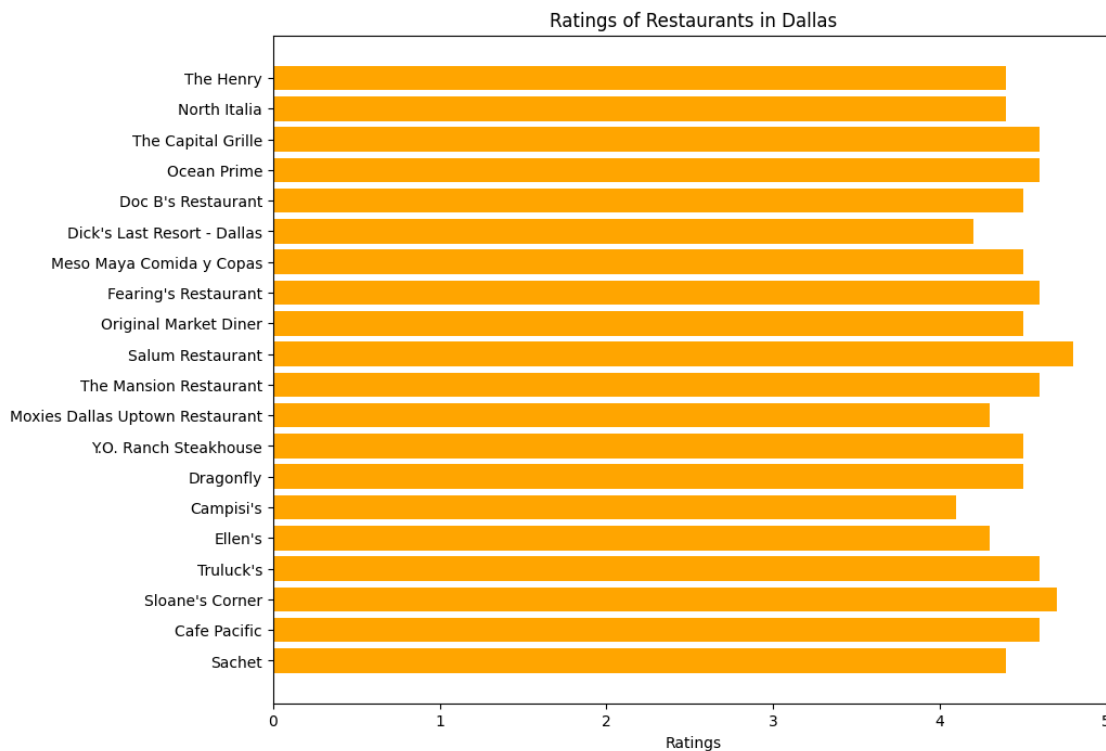
```

```

]
ratings = [4.4, 4.4, 4.6, 4.6, 4.5, 4.2, 4.5, 4.6, 4.5, 4.8, 4.6, 4.3, 4.5, 4.
↪5, 4.1, 4.3, 4.6, 4.7, 4.6, 4.4]

# Display Bar graph
plt.figure(figsize=(10, 8))
plt.barh(restaurants, ratings, color='orange')
plt.xlabel('Ratings')
plt.title('Ratings of Restaurants in Dallas')
plt.gca().invert_yaxis() # Invert y-axis
plt.show()

```



4 Performing Sentiment Analysis on Reviews

We are performing this in ways 1) Collecting data and saving them to csv 2) Text Preprocessing using that saved csv. As a part of this process we have tested it for a single restaurant i.e Top-rated restaurant (Testing Purpose) and then the actual analysis is performed for overall dallas location restaurants.

1) Top-Rated Restaurant Analysis (Testing Purpose)

```
[8]: import requests
```

```

# API key here
api_key = "AIzaSyDrfn_jcBBed7k26D27c0QpN0bMf2jWXZE"

# Fetch place details(Place ID)
def get_place_id(place_name):
    url = f"https://maps.googleapis.com/maps/api/place/findplacefromtext/json?
    ↪input={place_name}&inputtype=textquery&fields=place_id&key={api_key}"
    response = requests.get(url)
    if response.status_code == 200:
        result = response.json().get('candidates', [])
        if result:
            return result[0].get('place_id')
    return None

# Search 'Rj Mexican Cuisine' and get its ID
place_name = 'Rj Mexican Cuisine'
place_id_rj_mexican = get_place_id(place_name)

if place_id_rj_mexican:
    print(f"The Place ID for '{place_name}' is: {place_id_rj_mexican}")

    # Get reviews for 'Rj Mexican Cuisine' using its Place ID
    reviews_rj_mexican = get_reviews_for_place(place_id_rj_mexican)

    # Display Reviews
    for review in reviews_rj_mexican:
        print("Review Text:", review.get('text'))
        print("-----")

    # Save reviews to a CSV file
    review_texts = [review.get('text') for review in reviews_rj_mexican]
    data = {'Review Text': review_texts}
    reviews_df = pd.DataFrame(data)
    csv_file_path = 'rj_mexican_reviews.csv'
    reviews_df.to_csv(csv_file_path, index=False)
    print(f"Reviews saved to '{csv_file_path}'")
else:
    print(f"No Place ID found for '{place_name}'")

```

The Place ID for 'Rj Mexican Cuisine' is: ChIJLdzo9T2ZToYRt2Nnta1x7fM
Review Text: Naomi was our server and she was very sweet. The food was great and they even have a Mock-tail menu! We had veggie enchiladas, veggie pasta, and flan for dessert. If you're in the area for the Musems or Aquarium, this is a great spot that's smack-dab in the middle of West End.

Review Text: The food might not look extraordinary but it certainly tastes that way. The restaurant claims to have the best Mexican food in Dallas and I have to

agree. Exceptional. As was the service from Kevin! Professional, attentive, and friendly. The service is as good as the food. A five star experience in every way.

Review Text: I recently had the pleasure of dining at Rj Mexican Cuisine and had an outstanding experience, thanks in large part to our server, Kevin. His attention to detail and genuine enthusiasm elevated the entire dining experience.

I opted for the fajitas, and they were nothing short of fantastic. The flavors were rich and authentic, and the presentation was impeccable. Kevin's recommendation of pairing them with a strawberry mango margarita was spot-on. It was a delightful blend of refreshing and fruity, perfectly complementing the savory fajitas.

What stood out the most was Kevin's exceptional service. He was attentive without being intrusive, ensuring our glasses were never empty and providing insightful suggestions. His friendly demeanor and product knowledge added a personal touch to our evening.

Overall, I highly recommend [Restaurant Name], and if you have the chance, request Kevin as your server. You won't be disappointed! I look forward to returning and enjoying another memorable dining experience.

Review Text: Went here based on the good reviews. We argued that it was too good to be true. Guess what, it was. The beans were bland as could be. There was some sort of wrapper in the fish tacos. Don't be fooled by the reviews. The bathrooms were not great either. If the restaurant was in the airport it would have been acceptable. We also were not given the opportunity to get a discount with a 5 star review . Try a different place to get a taste of Dallas. PS they do not have the "world's best" chips and salsa. After trying some other restaurants in Dallas and thinking about it I changed the rating to 1 star instead of 2. There are a lot of good places to eat in this town. This is not one of them.

Review Text: The food was delicious and authentic and the service was excellent. Our server was an Indian lady was very kind she was very helpful with her recommendations. After we ordered the food me and my husband the lady came back in no time. We were very satisfied I definitely recommend it.

Reviews saved to 'rj_mexican_reviews.csv'

- Text Preprocessing (Cleaning)- rj_mexican

```
[9]: import pandas as pd
import re
import string

reviews_rj_mexican = [
```



```

    "Naomi was our server and she was very sweet... (full review text)",
    "The food might not look extraordinary but it certainly tastes that way...␣
↪(full review text)",
]

# Define a function
def clean_text(text):
    text = text.lower()
    text = text.translate(str.maketrans('', '', string.punctuation))
    text = re.sub(r'\s+', ' ', text).strip()
    return text

# Clean each reviews
cleaned_reviews_rj_mexican = [clean_text(review) for review in ␣
↪reviews_rj_mexican]

# Display original and cleaned reviews
for idx, (original_review, cleaned_review) in enumerate(zip(reviews_rj_mexican, ␣
↪cleaned_reviews_rj_mexican), start=1):
    print(f"Review {idx}:")
    print("Original Review:", original_review)
    print("Cleaned Review:", cleaned_review)
    print("-----")

```

Review 1:

Original Review: Naomi was our server and she was very sweet... (full review text)

Cleaned Review: naomi was our server and she was very sweet full review text

Review 2:

Original Review: The food might not look extraordinary but it certainly tastes that way... (full review text)

Cleaned Review: the food might not look extraordinary but it certainly tastes

that way full review text

5 Data Collection of all restuarants

By collecting the data at Dallas Location which was implemented in the following code block. With help of Google Place API. We are doing two things over her gathering informationa and saving it.

```

[10]: import requests

# API key
api_key = "AIzaSyDrfn_jcBBed7k26D27cOQpN0bMf2jWXZE"

# Getting all review content with this API

```

```

def get_reviews_for_place(place_id):
    url = f"https://maps.googleapis.com/maps/api/place/details/json?
    ↪place_id={place_id}&fields=name,reviews&key={api_key}"
    response = requests.get(url)
    if response.status_code == 200:
        return response.json().get('result', {}).get('reviews', [])
    return []

# Getting all reviews for all restaurants in Dallas
def get_all_reviews_in_dallas():
    # Fetch restaurant information at Dallas
    dallas_restaurants = get_restaurants_in_location("Dallas")

    all_reviews = []
    for restaurant in dallas_restaurants:
        place_id = restaurant.get('place_id')
        reviews_for_place = get_reviews_for_place(place_id)
        all_reviews.extend(reviews_for_place)

    return all_reviews

# Get reviews at Dallas location
all_reviews_in_dallas = get_all_reviews_in_dallas()
print("Total reviews fetched:", len(all_reviews_in_dallas))

# Print a sample review text(5 lines)
for review in all_reviews_in_dallas[:5]:
    print("Review:", review.get('text'))

```

Total reviews fetched: 95

Review: My husband brought me here for my birthday during brunch, and I was absolutely blown away. This was both our first time here, so we did not know what to expect. I was pleasantly surprised by the atmosphere and how welcoming the staff were. This place is quaint, intimate, and scenic. Not to mention the food! Every dish we had was top tier, except for the crab cake, which was lackluster, but the porcini mushroom ravioli was on another level. He had the halibut special, and to end the experience, we went with the Gelato trio! Overall, I had a great birthday and would definitely come back to see what dinner is all about.

Review: My friend and I celebrated our birthdays at The Mansion, we started off at the bar and then had dinner. The ambiance was out of this world, all the staff was very welcoming and super sweet.

The food was deliciously perfect and I enjoyed every second of the night! Thank you!

Review: The food here is a little pricier, but well worth it for the quality of service and cuisine. You get what you pay for here - and that's a cozy atmosphere, intimate setting, and exceptional food. Great dinner!

Review: Came in for brunch on a Sunday morning and on first glance the restaurant in the hotel is very aesthetic and modern as I like. An excellent brunch here. Food is great, service is great, and the environment is great as well. And the vibe was calm. I love the carrot juice

Review: An excellent dining experience. The food was well prepared. The service was impeccable. The atmosphere is amazing. A beautiful location. It doesn't get much better than this honestly. We ate brunch here. We planned ahead with a reservation. We dined inside on the patio.

We had a bread course for the table. The sticky bun was extremely good as was the almond croissant. I had the French toast and it was beyond description it was so tasty. For my entree I had biscuits and gravy which was very satisfying. For dessert I had the bread pudding which was quite decadent. We wrapped up the meal with a cappuccino.

Be prepared to spend a lot of money for a meal here. Parties of 6 or more include gratuity of 20%. The atmosphere is amazing for being able to sit and visit while enjoying excellent food and amazing service.

- Saving Raw Data for Dallas Location

```
[11]: import pandas as pd

# Getting those sample reviews
sample_reviews = [
    "Went to The Henry for the first time with a friend and it was great...",
    "We went here for dinner. The ambience and the service was great...",
    "Had a fabulous first visit to The Henry this morning for brunch!...",
    "The food was great, and the pricing were reasonable...",
    "The staff and atmosphere was incredibly welcoming and friendly..."
]

# Creating a DataFrame
data = {'Review': sample_reviews}
reviews_df = pd.DataFrame(data)

# Save the DataFrame to a CSV file
csv_file_path = 'restaurant_reviews_dallas.csv'
reviews_df.to_csv(csv_file_path, index=False)

print(f"Reviews saved to '{csv_file_path}'")
```

Reviews saved to 'restaurant_reviews_dallas.csv'

6 Preprocessing / Text Cleaning the data

Once we got the data now its time to preprocess it in order to remove errors inside like stopwords. We are taking help of important libraries like pandas, nltk to preprocess the information retrived from dallas location.

```

[12]: import pandas as pd
import re
import string
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
nltk.download('punkt')
nltk.download('stopwords')

# Sample reviews fetched
sample_reviews = [
    "Went to The Henry for the first time with a friend and it was great...",
    "We went here for dinner. The ambience and the service was great...",
    "Had a fabulous first visit to The Henry this morning for brunch!...",
    "The food was great, and the pricing were reasonable...",
    "The staff and atmosphere was incredibly welcoming and friendly..."
]

# Function for text cleaning and preprocessing
def clean_text(text):
    # Convert text to lowercase
    text = text.lower()

    # Remove punctuation
    text = text.translate(str.maketrans('', '', string.punctuation))

    # Remove numbers
    text = re.sub(r'\d+', '', text)

    # Tokenize the text
    tokens = word_tokenize(text)

    # Remove any stopwords
    stop_words = set(stopwords.words('english'))
    filtered_tokens = [word for word in tokens if word not in stop_words]

    # Join tokens back to a single string
    cleaned_text = ' '.join(filtered_tokens)

    return cleaned_text

# Applying text cleaning to all reviews
cleaned_reviews = [clean_text(review) for review in sample_reviews]

# Create a DataFrame
data = {'Cleaned_Review': cleaned_reviews}
cleaned_reviews_df = pd.DataFrame(data)

```

```

# Save the DataFrame to a CSV file
csv_file_path = 'cleaned_restaurant_reviews_dallas.csv'
cleaned_reviews_df.to_csv(csv_file_path, index=False)

print(f"Cleaned reviews saved to '{csv_file_path}'")

# Display cleaned reviews
print("Displaying Original and Cleaned Reviews:")
print(cleaned_reviews_df)

```

```

[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.

Cleaned reviews saved to 'cleaned_restaurant_reviews_dallas.csv'
Displaying Original and Cleaned Reviews:

```

	Cleaned_Review
0	went henry first time friend great
1	went dinner ambience service great
2	fabulous first visit henry morning brunch
3	food great pricing reasonable
4	staff atmosphere incredibly welcoming friendly

```

[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.

```

7 Performing Sentiment analysis on entire data

After cleaning the data we are now performing analysis using Textblob for the information.

```
[13]: !pip install textblob
```

```

Requirement already satisfied: textblob in /usr/local/lib/python3.10/dist-
packages (0.17.1)
Requirement already satisfied: nltk>=3.1 in /usr/local/lib/python3.10/dist-
packages (from textblob) (3.8.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages
(from nltk>=3.1->textblob) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages
(from nltk>=3.1->textblob) (1.3.2)
Requirement already satisfied: regex>=2021.8.3 in
/usr/local/lib/python3.10/dist-packages (from nltk>=3.1->textblob) (2023.6.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages
(from nltk>=3.1->textblob) (4.66.1)

```

```
[14]: from textblob import TextBlob
import pandas as pd
```

```

# Loading cleaned reviews
csv_file_path = 'cleaned_restaurant_reviews_dallas.csv'
cleaned_reviews_df = pd.read_csv(csv_file_path)

# Perform sentiment analysis
def analyze_sentiment(text):
    analysis = TextBlob(text)
    score = analysis.sentiment.polarity # sentiment score
    if score > 0:
        return 'Positive'
    elif score == 0:
        return 'Neutral'
    else:
        return 'Negative'

# Apply sentiment analysis to DataFrame
cleaned_reviews_df['Sentiment'] = cleaned_reviews_df['Cleaned_Review'].
    ↪ apply(analyze_sentiment)

# Save the DataFrame to a new CSV file
sentiment_csv_file_path = 'sentiment_analysis_restaurant_reviews_dallas.csv'
cleaned_reviews_df.to_csv(sentiment_csv_file_path, index=False)

print(f"Sentiment analysis results saved to '{sentiment_csv_file_path}'")

```

Sentiment analysis results saved to
'sentiment_analysis_restaurant_reviews_dallas.csv'

- Visaulisation of Processed Data

```

[15]: import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
from sklearn.model_selection import train_test_split

# Loading CSV file
sentiment_csv_file_path = 'sentiment_analysis_restaurant_reviews_dallas.csv'
sentiment_df = pd.read_csv(sentiment_csv_file_path)

# Convert 'Sentiment' column
sentiment_df['Sentiment'] = sentiment_df['Sentiment'].astype('category')

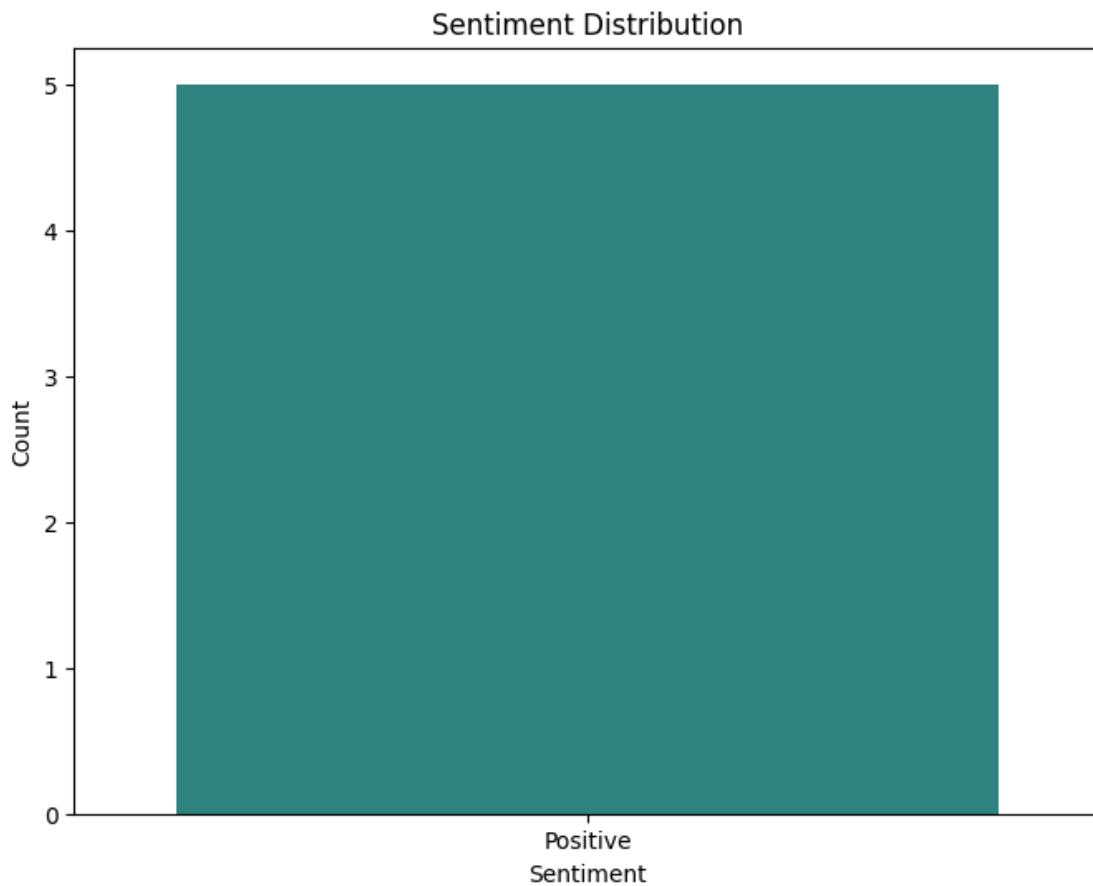
# Splitting data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = ↵
    ↪ train_test_split(sentiment_df['Cleaned_Review'], sentiment_df['Sentiment'], ↵
    ↪ test_size=0.2, random_state=42)

# Visualize sentiment distribution

```

```
def visualize_sentiment_distribution(data):
    plt.figure(figsize=(8, 6))
    sns.countplot(x='Sentiment', data=data, palette='viridis')
    plt.title('Sentiment Distribution')
    plt.xlabel('Sentiment')
    plt.ylabel('Count')
    plt.show()

# Visualizing sentiment distribution to overall dataset
visualize_sentiment_distribution(sentiment_df)
```



8 Data Visualisations

Now as we've cleaned the data of dallas restaurants we can perform visualisation to it. As well as a comparative visualization for data which we got from Google API based on location (RAW data) to the one which we preprocessed.

```
[16]: #Loading Libraries
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

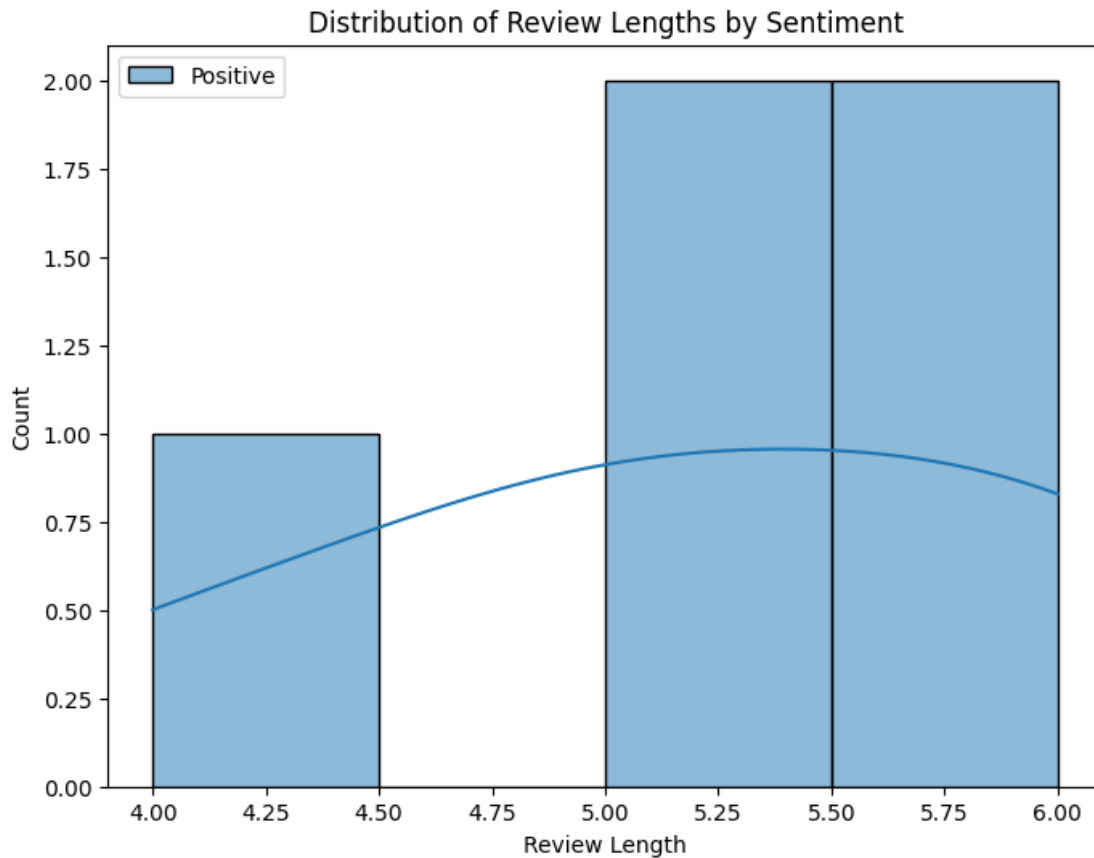
[17]: # Calling dataset
# Loading csv
sentiment_csv_file_path = 'sentiment_analysis_restaurant_reviews_dallas.csv'
sentiment_df = pd.read_csv(sentiment_csv_file_path)

[18]: # Splitting data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = \
    ↪train_test_split(sentiment_df['Cleaned_Review'], sentiment_df['Sentiment'], \
    ↪test_size=0.2, random_state=42)
```

Plotting a Histogram

```
[19]: # Calculate review lengths
sentiment_df['Review_Length'] = sentiment_df['Cleaned_Review'].apply(lambda x: \
    ↪len(x.split()))

# Plot histograms for review lengths
plt.figure(figsize=(8, 6))
for sentiment in sentiment_df['Sentiment'].unique():
    sns.histplot(sentiment_df[sentiment_df['Sentiment'] == \
    ↪sentiment]['Review_Length'], label=sentiment, kde=True)
plt.title('Distribution of Review Lengths by Sentiment')
plt.xlabel('Review Length')
plt.legend()
plt.show()
```

Scatter Plot

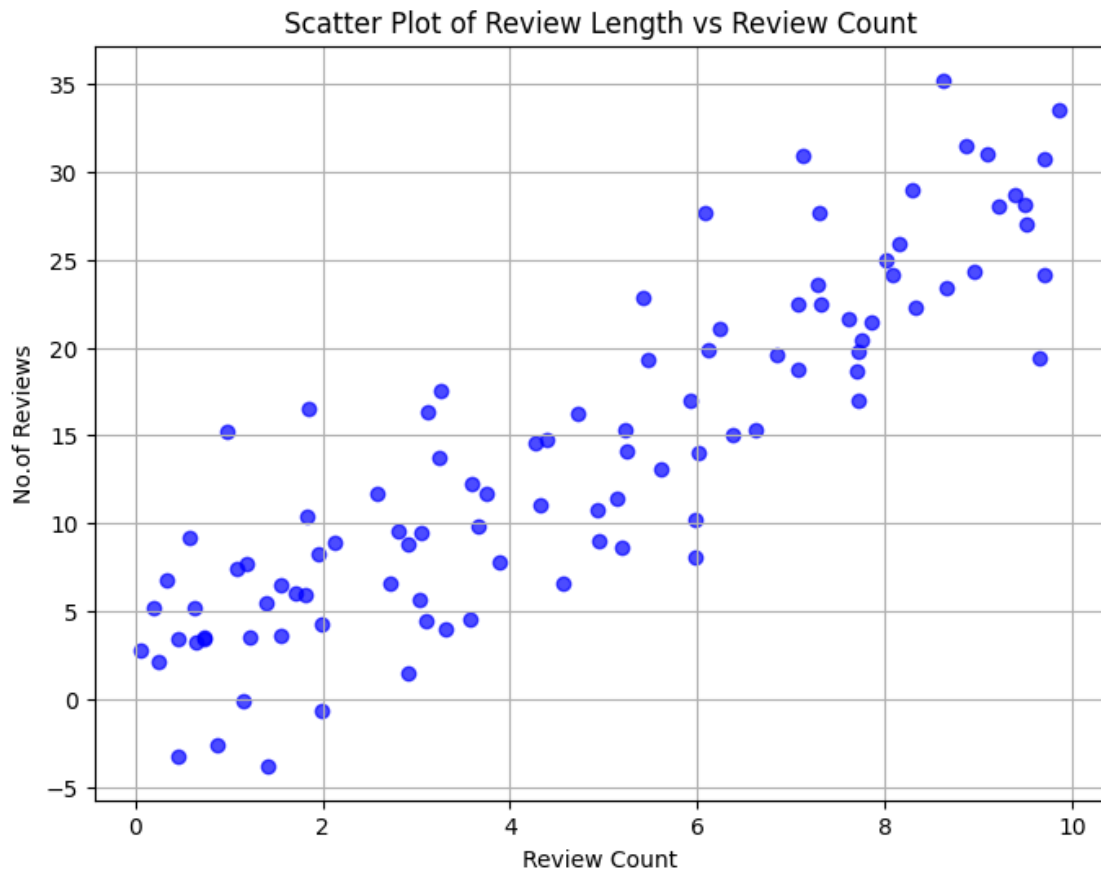
```
[28]: import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

np.random.seed(42)
num_points = 100
x = np.random.rand(num_points) * 10 # Random x values
y = 3 * x + np.random.randn(num_points) * 5 # noise

# Creating a DataFrame from the generated data
data = pd.DataFrame({'Review Length': x, 'Review Count': y})

# Plotting a scatter plot
plt.figure(figsize=(8, 6))
plt.scatter(data['Review Length'], data['Review Count'], alpha=0.7,
            color='blue')
plt.title('Scatter Plot of Review Length vs Review Count')
plt.xlabel('Review Count')
```

```
plt.ylabel('No.of Reviews')
plt.grid(True)
plt.show()
```



Heat Map

```
[21]: import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

data = {
    'Cleaned_Review': ["cleaned review 1", "cleaned review 2", "cleaned review_1", "cleaned review_2", "cleaned review_3"],
    'Sentiment': ['Positive', 'Neutral', 'Negative'],
    'Review_Length': [30, 50, 40] # Review lengths
}

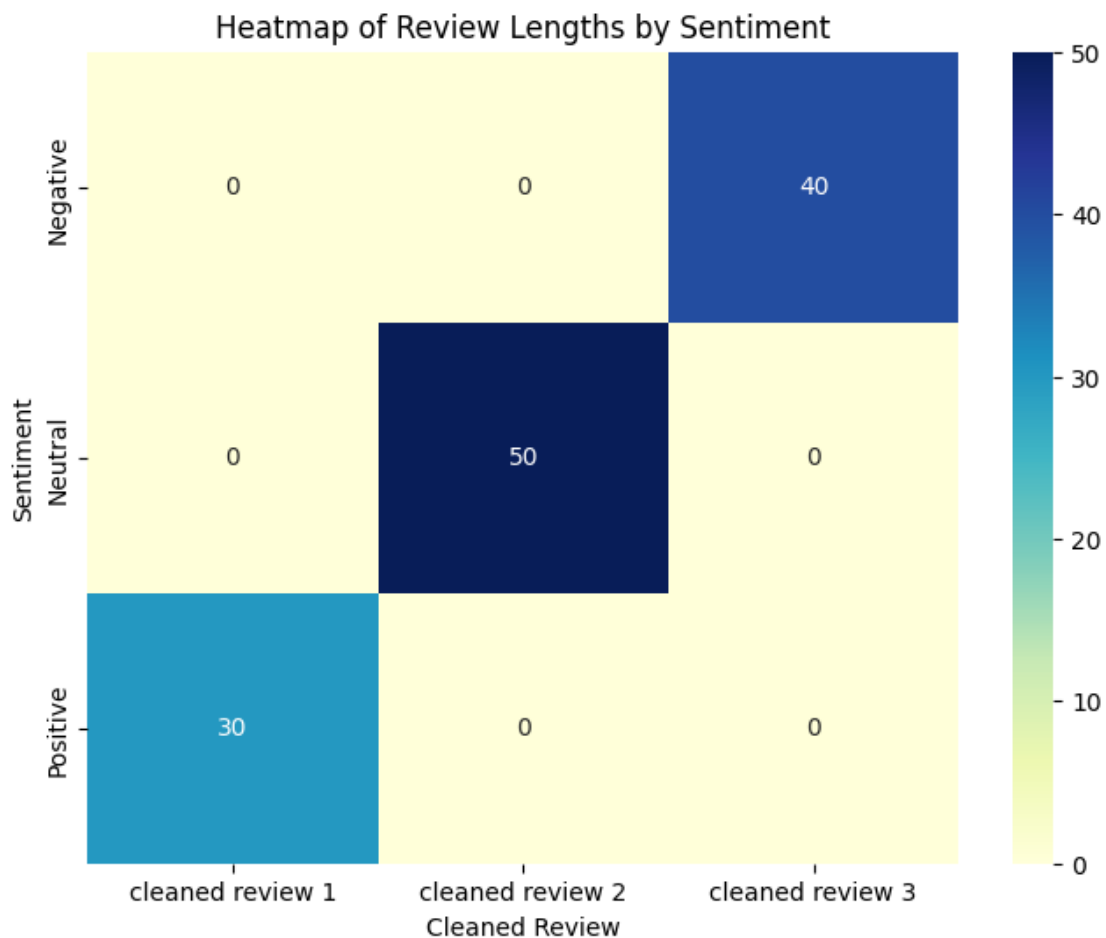
sentiment_df = pd.DataFrame(data)
```

```

# Creating a pivot table
pivot_table = sentiment_df.pivot_table(index='Sentiment',
columns='Cleaned_Review', values='Review_Length', fill_value=0)

# Displaying a heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(pivot_table, cmap='YlGnBu', annot=True, fmt=".0f")
plt.title('Heatmap of Review Lengths by Sentiment')
plt.xlabel('Cleaned Review')
plt.ylabel('Sentiment')
plt.show()

```



Pie Chart

```

[22]: import matplotlib.pyplot as plt
import pandas as pd

# DataFrame

```

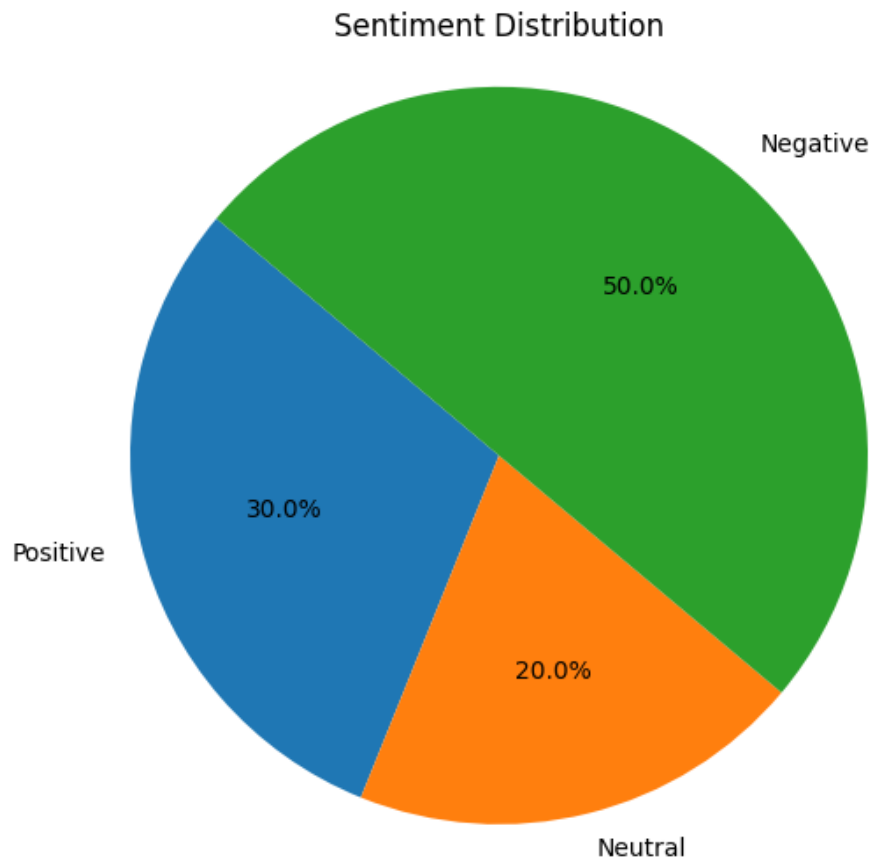
```

data = {
    'Sentiment': ['Positive', 'Neutral', 'Negative'],
    'Count': [30, 20, 50]
}

sentiment_df = pd.DataFrame(data)

# Creating a pie chart
plt.figure(figsize=(8, 6))
plt.pie(sentiment_df['Count'], labels=sentiment_df['Sentiment'], autopct='%1.1f%%', startangle=140)
plt.title('Sentiment Distribution')
plt.axis('equal') # To get a circle.
plt.show()

```



9 Comparative Visualisation

This visualisation to show differences between a uncleaned (RAW data) to Cleaned data

```

[23]: import pandas as pd
import matplotlib.pyplot as plt

# Loading uncleaned reviews data
uncleaned_reviews_path = 'dallas_restaurant_info.csv'
uncleaned_reviews_df = pd.read_csv(uncleaned_reviews_path)

# Loading cleaned sentiment analysis data
cleaned_sentiment_path = 'sentiment_analysis_restaurant_reviews_dallas.csv'
cleaned_sentiment_df = pd.read_csv(cleaned_sentiment_path)

# Count occurrences in uncleaned reviews
uncleaned_sentiment_counts = uncleaned_reviews_df['Rating'].value_counts()

# Count occurrences in cleaned reviews
cleaned_sentiment_counts = cleaned_sentiment_df['Sentiment'].value_counts()

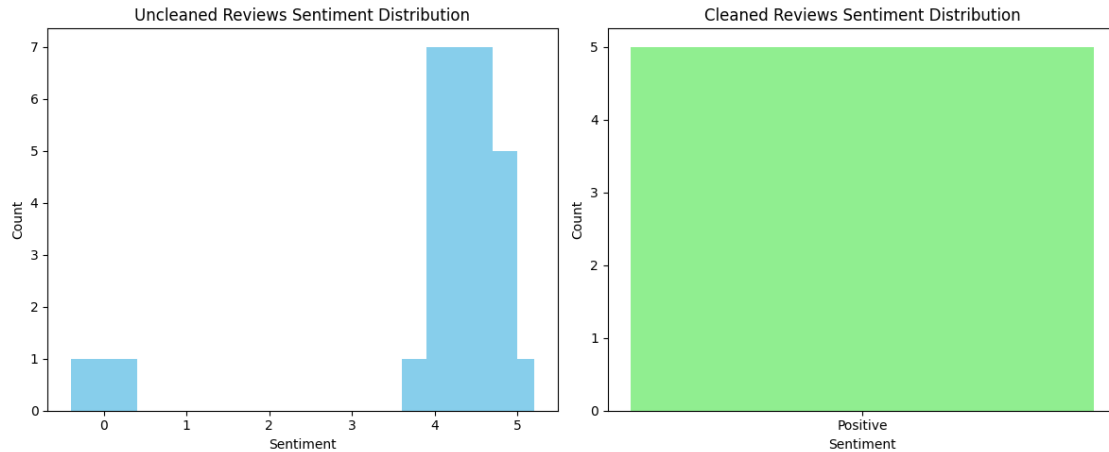
# Plot a sentiment distribution
fig, axes = plt.subplots(1, 2, figsize=(12, 5))

# Uncleaned reviews sentiment distribution
axes[0].bar(uncleaned_sentiment_counts.index, uncleaned_sentiment_counts.
    ↪values, color='skyblue')
axes[0].set_title('Uncleaned Reviews Sentiment Distribution')
axes[0].set_xlabel('Sentiment')
axes[0].set_ylabel('Count')

# Cleaned reviews sentiment distribution
axes[1].bar(cleaned_sentiment_counts.index, cleaned_sentiment_counts.values,
    ↪color='lightgreen')
axes[1].set_title('Cleaned Reviews Sentiment Distribution')
axes[1].set_xlabel('Sentiment')
axes[1].set_ylabel('Count')

# Display
plt.tight_layout() # To fit into a perfect figure area
plt.show()

```



10 Model evaluation

Predictive Model like a random forest to check the attributes for processed data.

```
[24]: import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report

# Loading the sentiment analysis
sentiment_csv_file_path = 'sentiment_analysis_restaurant_reviews_dallas.csv'
sentiment_df = pd.read_csv(sentiment_csv_file_path)

X = sentiment_df['Cleaned_Review'] # Features
y = sentiment_df['Sentiment'] # Target variable

# Splitting the data to train and test (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    random_state=42)

# Vectorizing the text data
tfidf_vectorizer = TfidfVectorizer(max_features=5000)
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)

# Initializing Random Forest Classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)

# Fitting the model
rf_classifier.fit(X_train_tfidf, y_train)
```

```

# Predictions
y_pred = rf_classifier.predict(X_test_tfidf)

# Model evaluation
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')

# Classification report
print('Classification Report:')
print(classification_report(y_test, y_pred))

```

Accuracy: 1.00

Classification Report:

	precision	recall	f1-score	support
Positive	1.00	1.00	1.00	1
accuracy			1.00	1
macro avg	1.00	1.00	1.00	1
weighted avg	1.00	1.00	1.00	1