



CCDS 322 Final Project Report

Customer Feedback

Group members:

Dania Abdelmonem 2112241

Shatha Al-Qarni 2111884

Nrdeen Ali 2110419

Salma Mohammed 2110956

Introduction:

Our dataset contains reviews covering various services and products such as movies, food, and websites. These reviews are categorized into positive/negative sentiments.

Research/Our Approach:

To make it easier for decision makers to be informed about customers' opinions by analyzing their responses and developing the product to satisfy the customer. Our data is from Kaggle.

The Models We Used:

We used Support Vector Machine (SVM) & Random Forest to compare the results and analysis

Our Goal Is To Answer The Following Questions:

1. What are the specific words to see if the comment is positive or negative?
2. Were most comments positive or negative?
3. Is there a comment consisting of a negative/positive part at the same time?
4. Let's say there is a comment that has nothing to do with being positive/negative, how will it be classified?

Analysis and Results: The libraries we used

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
from nltk.corpus import stopwords
stop=set(stopwords.words('english'))
from nltk.util import ngrams
from nltk.tokenize import word_tokenize
from sklearn.feature_extraction.text import CountVectorizer
from collections import defaultdict
from collections import Counter
plt.style.use('ggplot')
import re
import string
import warnings
warnings.filterwarnings('ignore')
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

import seaborn as sns
import matplotlib.pyplot as plt
```

Data Pre-processing:

This is what our data looks like at first.

```
data=pd.read_csv('sentiment-analysis.csv')
data.head()
```

data - DataFrame

Index	text, Source, Date/Time, User ID, Location, Confidence
0	"I love this product!", Positive, Twitter, 202...
1	"The service was terrible.", Negative, Yelp Re...
2	"This movie is amazing!", Positive, IMDb, 2023...
3	"I'm so disappointed with their customer suppo...
4	"Just had the best meal of my life!", Positive...
5	"The quality of this product is subpar.", Nega...
6	"I can't stop listening to this song. It's inc...
7	"Their website is so user-friendly. Love it!",...
8	"I loved the movie! It was fantastic!", Positi...
9	"The customer service was terrible.", Negative...
10	"This book made me feel inspired. Highly recom...
11	"I'm extremely disappointed with their product...
12	"Just had the most amazing vacation! I can't w...
13	"The food at this restaurant was awful. Never ...

Format Resize ☒ Background color ☒ Column min/max

We then divided the data, putting each in a column:

```
#
#loading the data set
data=pd.read_csv('sentiment-analysis.csv')
data.head()
#separating the data into columns
data = data['Text, Source, Date/Time, User ID, Location, Confidence Score'].str.split(' ', expand=True)
data.columns = ['Text', 'Sentiment', 'Source', 'Date/Time', 'User ID', 'Location', 'Confidence Score']
print('There are {} rows and {} columns in data.'.format(data.shape[0], data.shape[1]))

data.head(10)
```

data - DataFrame

	Index	Text	Sentiment	Source	Date/Time	User ID	Location	fidence S	
	0	"I love this product!"	Positive	Twitter	2023-06-15 09:23:14	@user123	New York	0.85	
	1	"The service was terrible."	Negative	Yelp Reviews	2023-06-15 11:45:32	user456	Los Angeles	0.65	
	2	"This movie is amazing!"	Positive	IMDb	2023-06-15 14:10:22	moviefan789	London	0.92	
	3	"I'm so disappointed with their customer suppo..	Negative	Online Forum	2023-06-15 17:35:11	forumuser1	Toronto	0.78	
	4	"Just had the best meal of my life!"	Positive	TripAdvisor	2023-06-16 08:50:59	foodie22	Paris	0.88	
	5	"The quality of this product is subpar."	Negative	Amazon Reviews	2023-06-16 10:15:27	shopper123	San Francisco	0.72	
	6	"I can't stop listening to this song. It's inc..	Positive	Spotify	2023-06-16 13:40:18	musiclover456	Berlin	0.91	
	7	"Their website is so user-friendly. Love it!"	Positive	Website Testimonial	2023-06-16 16:05:36	testimonialuser1	Sydney	0.87	
	8	"I loved the movie! It was fantastic!"	Positive	IMDb	2023-07-02 09:12:34	user123	New York	0.92	
	9	"The customer service was terrible."	Negative	Yelp Reviews	2023-07-02 10:45:21	user456	Los Angeles	0.65	
	10	"This book made me feel inspired. Highly recom..	Positive	Goodreads	2023-07-02 12:34:56	bookworm789	London	0.88	
	11	"I'm extremely disappointed with their product..	Negative	Online Store	2023-07-02 15:21:43	shopper789	San Francisco	0.72	
	12	"Just had the most amazing vacation! I can't w..	Positive	TripAdvisor	2023-07-02 18:01:23	travelenthusiast1	Sydney	0.93	
	13	"The food at this restaurant was awful. Never ..	Negative	Zomato	2023-07-02 20:45:37	foodlover123	Mumbai	0.55	
	14	"I can't stop listening to this song. It's my ..	Positive	Spotify	2023-07-03 09:17:52	musiclover789	Berlin	0.91	
	15	"Their website is so confusing and poorly desi..	Negative	Website Review	2023-07-03 11:59:18	user789	Toronto	0.68	
	16	"I had an incredible experience at the theme p..	Positive	Trip Report	2023-07-03 14:40:05	thrillseeker1	Orlando	0.89	
	17	"The product arrived damaged. Very disappointe..	Negative	Online Store	2023-07-03 17:25:09	buyer123	Chicago	0.76	
	18	"The concert was absolutely breathtaking. Best..	Positive	Event Review	2023-07-03 20:15:33	musicfan456	Paris	0.95	
	19	"I had a terrible experience with their custom..	Negative	Online Chat	2023-07-04 08:32:41	user1234	Sydney	0.61	
	20	"This movie is a masterpiece! I was blown away..	Positive	IMDb	2023-07-04 14:05:27	cinophile789	Los Angeles	0.93	
	21	"The customer service at this store is top-not..	Positive	Yelp Reviews	2023-07-04 16:35:19	shopper456	New York	0.82	
	22	"I'm disappointed with the ending of this book..	Negative	Goodreads	2023-07-04 19:18:53	booklover123	London	0.68	
	23	"The product I received was damaged. Unaccepta..	Negative	Online Store	2023-07-04 21:52:41	buyer789	San Francisco	0.75	
	24	"Just had the worst flight experience. Delayed..	Negative	Airline Review	2023-07-05 09:10:15	traveler456	Sydney	0.57	
	25	"The hotel stay was absolutely amazing! Luxury..	Positive	TripAdvisor	2023-07-05 12:30:08	traveladdict1	Paris	0.95	
	26	"The food at this restaurant was outstanding. ..	Positive	Zomato	2023-07-05 15:45:32	foodie789	Mumbai	0.89	
	27	"This playlist is my go-to for workouts. Energ..	Positive	Spotify	2023-07-05 18:20:17	gymrat123	Berlin	0.91	
	28	"I had a frustrating experience navigating thr..	Negative	Website Review	2023-07-05 21:05:44	user987	Toronto	0.67	
	29	"The roller coaster ride was thrilling! Heart..	Positive	Theme Park Review	2023-07-06 08:48:39	thrillseeker2	Orlando	0.92	
	30	"The product I ordered never arrived. Terrible..	Negative	Online Store	2023-07-06 11:25:56	customer123	Chicago	0.64	

Then we cleaned our data and separated the date&time:

```
#Data Cleaning
data.isnull().sum()
data.dropna(inplace=True)
data.head(10)

#there are some leading and trailing spaces in 'Date/Time' column, so let's trim them
data['Date/Time'] = data['Date/Time'].str.strip()

data[['Date', 'Time']] = data['Date/Time'].str.split(' ', expand=True)

data.drop(columns=['Date/Time'], inplace=True)
data['Date'] = pd.to_datetime(data['Date'])
data['Time'] = pd.to_datetime(data['Time'], format='%H:%M:%S').dt.time
```

data - DataFrame

	Index	Text	Sentiment	Source	User ID	Location	fidence S	Date	Time
	0	"I love this product!"	Positive	Twitter	@user123	New York	0.85	2023-06-15 00:00:00	09:23:14
	1	"The service was terrible."	Negative	Yelp Reviews	user456	Los Angeles	0.65	2023-06-15 00:00:00	11:45:32
	2	"This movie is amazing!"	Positive	IMDb	moviefan789	London	0.92	2023-06-15 00:00:00	14:10:22
	3	"I'm so disappointed with their customer suppo..	Negative	Online Forum	forumuser1	Toronto	0.78	2023-06-15 00:00:00	17:35:11
	4	"Just had the best meal of my life!"	Positive	TripAdvisor	foodie22	Paris	0.88	2023-06-16 00:00:00	08:50:59
	5	"The quality of this product is subpar."	Negative	Amazon Reviews	shopper123	San Francisco	0.72	2023-06-16 00:00:00	10:15:27
	6	"I can't stop listening to this song. It's inc..	Positive	Spotify	musiclover456	Berlin	0.91	2023-06-16 00:00:00	13:40:18
	7	"Their website is so user-friendly. Love it!"	Positive	Website Testimonial	testimonialuser1	Sydney	0.87	2023-06-16 00:00:00	16:05:36
	8	"I loved the movie! It was fantastic!"	Positive	IMDb	user123	New York	0.92	2023-07-02 00:00:00	09:12:34
	9	"The customer service was terrible."	Negative	Yelp Reviews	user456	Los Angeles	0.65	2023-07-02 00:00:00	10:45:21
	10	"This book made me feel inspired. Highly recom..	Positive	Goodreads	bookworm789	London	0.88	2023-07-02 00:00:00	12:34:56
	11	"I'm extremely disappointed with their product..	Negative	Online Store	shopper789	San Francisco	0.72	2023-07-02 00:00:00	15:21:43
	12	"Just had the most amazing vacation! I can't w..	Positive	TripAdvisor	travelenthusiast1	Sydney	0.93	2023-07-02 00:00:00	18:01:23
	13	"The food at this restaurant was awful. Never ..	Negative	Zomato	foodlover123	Mumbai	0.55	2023-07-02 00:00:00	20:45:37
	14	"I can't stop listening to this song. It's my ..	Positive	Spotify	musiclover789	Berlin	0.91	2023-07-03 00:00:00	09:17:52
	15	"Their website is so confusing and poorly desi..	Negative	Website Review	user789	Toronto	0.68	2023-07-03 00:00:00	11:59:18
	16	"I had an incredible experience at the theme p..	Positive	Trip Report	thrillseeker1	Orlando	0.89	2023-07-03 00:00:00	14:40:05
	17	"The product arrived damaged. Very disappointe..	Negative	Online Store	buyer123	Chicago	0.76	2023-07-03 00:00:00	17:25:09
	18	"The concert was absolutely breathtaking. Best..	Positive	Event Review	musicfan456	Paris	0.95	2023-07-03 00:00:00	20:15:33
	19	"I had a terrible experience with their custom..	Negative	Online Chat	user1234	Sydney	0.61	2023-07-04 00:00:00	08:32:41
	20	"This movie is a masterpiece! I was blown away..	Positive	IMDb	cinophile789	Los Angeles	0.93	2023-07-04 00:00:00	14:05:27
	21	"The customer service at this store is top-not..	Positive	Yelp Reviews	shopper456	New York	0.82	2023-07-04 00:00:00	16:35:19
	22	"I'm disappointed with the ending of this book..	Negative	Goodreads	booklover123	London	0.68	2023-07-04 00:00:00	19:18:53
	23	"The product I received was damaged. Unaccepta..	Negative	Online Store	buyer789	San Francisco	0.75	2023-07-04 00:00:00	21:52:41
	24	"Just had the worst flight experience. Delayed..	Negative	Airline Review	traveler456	Sydney	0.57	2023-07-05 00:00:00	09:10:15
	25	"The hotel stay was absolutely amazing! Luxury..	Positive	TripAdvisor	traveladdict1	Paris	0.95	2023-07-05 00:00:00	12:30:08

We then distributed the data, comparing the positive and negative opinions:

```
#Data Distribution
sns.countplot(x='Sentiment', data=data, palette= ['lightgreen', 'red'])
plt.xlabel('Sentiment')
plt.ylabel('Count')
plt.title('Count of Positive and Negative Reviews')
plt.show()
data['Sentiment'] = data['Sentiment'].str.strip()
data['Sentiment'] = data['Sentiment'].str.lower()
fig, axes = plt.subplots(2, 2, figsize=(12, 8))
```



We used 20% of the data randomly as a test set:

```
# Preparing the data
X = data['Text']
y = data['Sentiment']

# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Feature extraction using TF-IDF vectorizer
tfidf_vectorizer = TfidfVectorizer(max_features=5000)
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)
```


X_test - Series		y_test - Series	
Index	Text	Index	Sentiment
80	"The service was terrible."	80	Negative
77	"This song always puts me in a nostalgic mood..."	77	Positive
73	"The customer service at this store is outstan..."	73	Positive
94	"Their website is so confusing and poorly desi..."	94	Negative
33	"This restaurant has the most delicious food. ..."	33	Positive
79	"I love this product!"	79	Positive
69	"The website navigation is smooth and intuitiv..."	69	Positive
42	"The roller coaster ride was exhilarating! Pur..."	42	Positive
0	"I love this product!"	0	Positive
10	"This book made me feel inspired. Highly recom..."	10	Positive
64	"The roller coaster at this theme park is a th..."	64	Positive
30	"The product I ordered never arrived. Terrible..."	30	Negative
18	"The concert was absolutely breathtaking. Best..."	18	Positive
4	"Just had the best meal of my life!"	4	Positive

Our SVM Model (kernel is linear):

```
# Support Vector Machine (SVM)
svm_classifier = SVC(kernel='Linear', C=1)
svm_classifier.fit(X_train_tfidf, y_train)

# Predictions
svm_predictions = svm_classifier.predict(X_test_tfidf)

# Evaluating SVM
svm_accuracy = accuracy_score(y_test, svm_predictions)
print("SVM Accuracy:", svm_accuracy)
print("\nClassification Report for SVM:\n", classification_report(y_test, svm_predictions))
print("Confusion Matrix for SVM:\n", confusion_matrix(y_test, svm_predictions))
```

```
Code )
There are 98 rows and 7 columns in data.
SVM Accuracy: 0.9

Classification Report for SVM:
              precision    recall  f1-score   support

   Negative       0.71         1.00      0.83         5
   Positive       1.00         0.87      0.93        15

   accuracy                0.90         20
  macro avg       0.86         0.93      0.88         20
 weighted avg       0.93         0.90      0.90         20

Confusion Matrix for SVM:
[[ 5  0]
 [ 2 13]]
```

Our Random Forest Model (determined the `n_estimators` to be 100):

```
# Random Forest
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
rf_classifier.fit(X_train_tfidf, y_train)

# Predictions
rf_predictions = rf_classifier.predict(X_test_tfidf)

# Evaluating Random Forest
rf_accuracy = accuracy_score(y_test, rf_predictions)
print("\nRandom Forest Accuracy:", rf_accuracy)
print("\nClassification Report for Random Forest:\n", classification_report(y_test, rf_predictions))
print("Confusion Matrix for Random Forest:\n", confusion_matrix(y_test, rf_predictions))
```

Random Forest Accuracy: 0.85

Classification Report for Random Forest:

	precision	recall	f1-score	support
Negative	0.62	1.00	0.77	5
Positive	1.00	0.80	0.89	15
accuracy			0.85	20
macro avg	0.81	0.90	0.83	20
weighted avg	0.91	0.85	0.86	20

Confusion Matrix for Random Forest:

```
[[ 5  0]
 [ 3 12]]
```

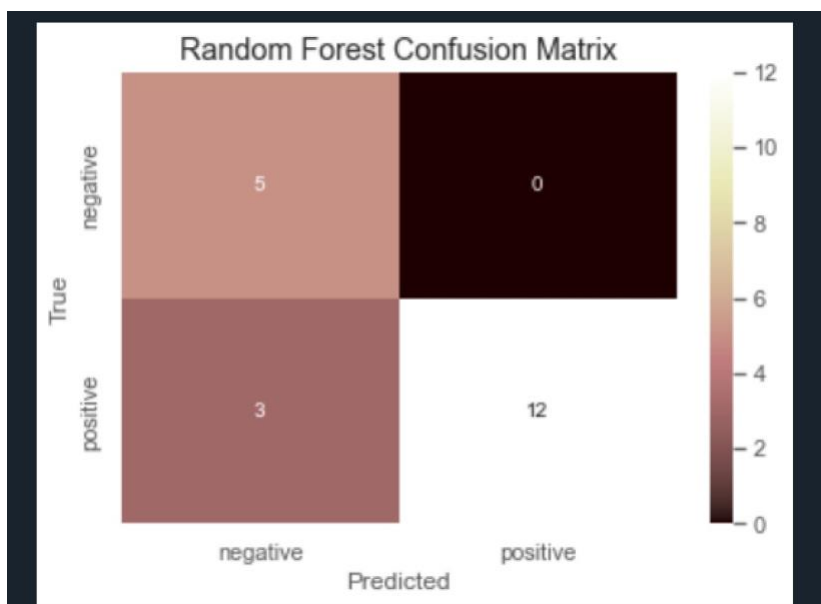
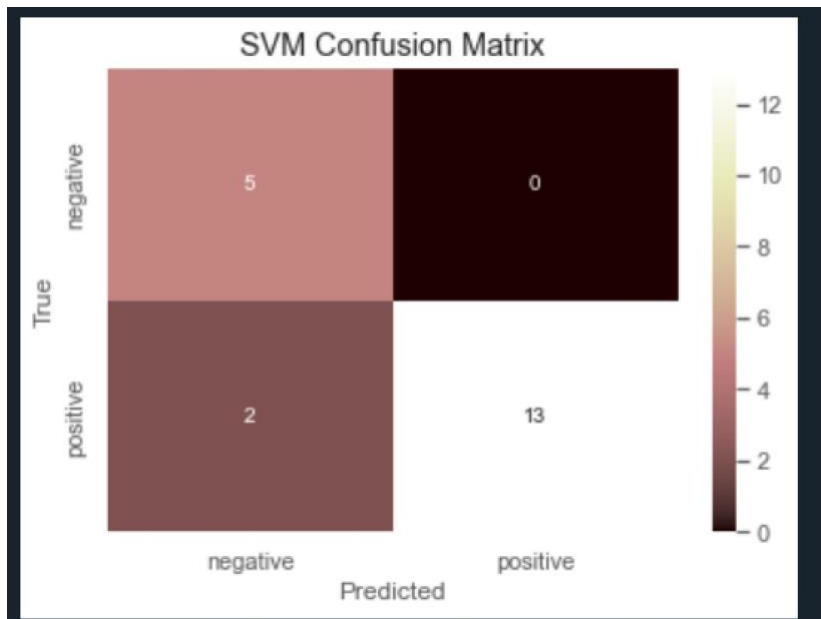
Confusion Matrix, Correlation Heatmap:

```
# Function to plot confusion matrix
def plot_confusion_matrix(y_true, y_pred, title, labels):
    cm = confusion_matrix(y_true, y_pred)
    plt.figure(figsize=(6, 4))
    sns.heatmap(cm, annot=True, fmt='d', cmap='pink', xticklabels=labels, yticklabels=labels)
    plt.title(title)
    plt.xlabel('Predicted')
    plt.ylabel('True')
    plt.show()

# Plotting Confusion Matrix for SVM
plot_confusion_matrix(y_test, svm_predictions, 'SVM Confusion Matrix', ['negative', 'positive'])

# Plotting Confusion Matrix for Random Forest
plot_confusion_matrix(y_test, rf_predictions, 'Random Forest Confusion Matrix', ['negative', 'positive'])

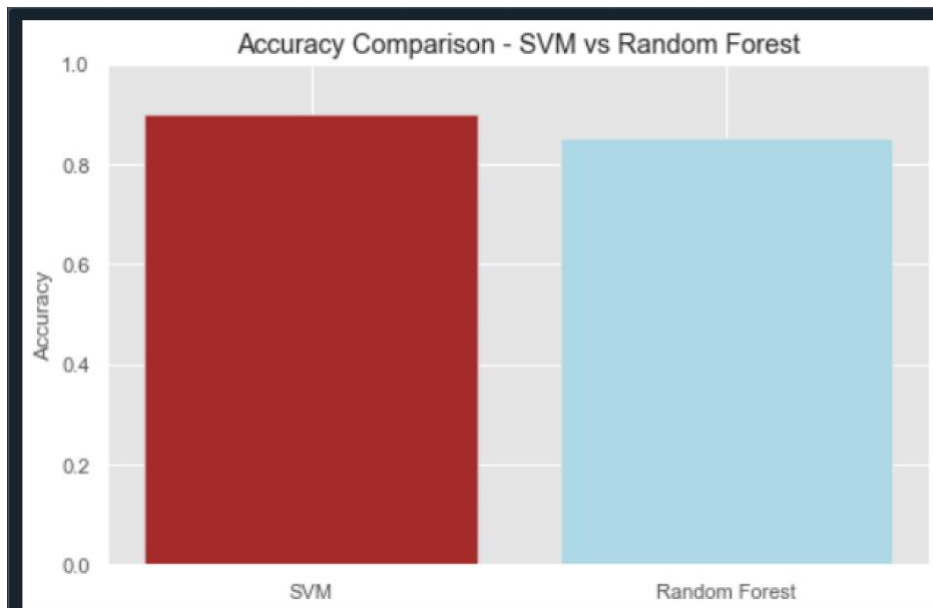
# Plotting Accuracy Comparison
models = ['SVM', 'Random Forest']
accuracies = [svm_accuracy, rf_accuracy]
```



Model Accuracy Comparison:

```
# Plotting Accuracy Comparison
models = ['SVM', 'Random Forest']
accuracies = [svm_accuracy, rf_accuracy]

plt.figure(figsize=(8, 5))
plt.bar(models, accuracies, color=['brown', 'lightblue'])
plt.ylabel('Accuracy')
plt.title('Accuracy Comparison - SVM vs Random Forest')
plt.ylim(0, 1)
plt.show()
```

Comparison between Training/Test Set:

```
# Function to plot confusion matrix
def plot_confusion_matrix(y_true, y_pred, title, labels):
    cm = confusion_matrix(y_true, y_pred)
    plt.figure(figsize=(6, 4))
    sns.heatmap(cm, annot=True, fmt='d', cmap='tab20', xticklabels=labels, yticklabels=labels)
    plt.title(title)
    plt.xlabel('Predicted')
    plt.ylabel('True')
    plt.show()

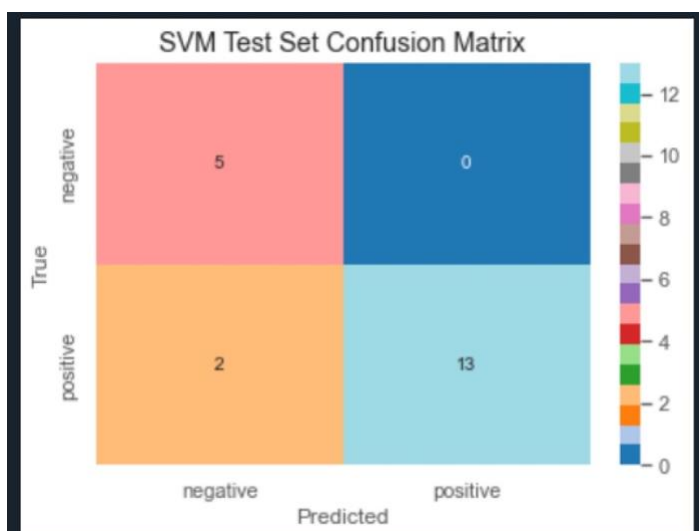
# Plotting Confusion Matrix for SVM (Test Set)
plot_confusion_matrix(y_test, svm_predictions, 'SVM Test Set Confusion Matrix', ['negative', 'positive'])

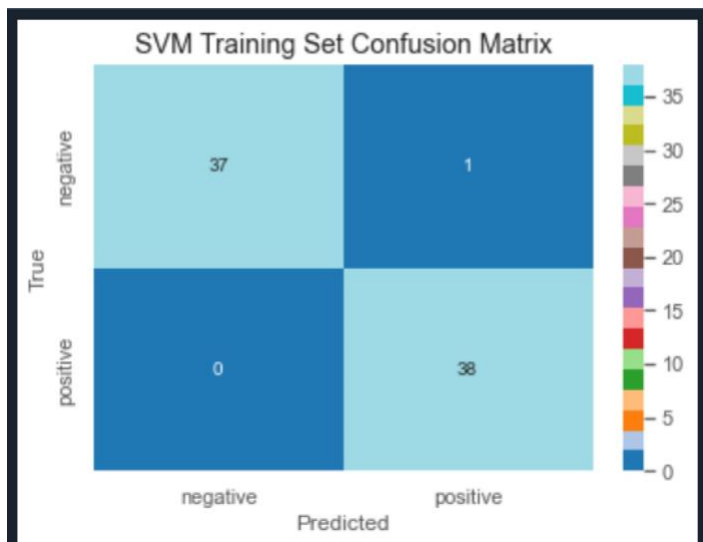
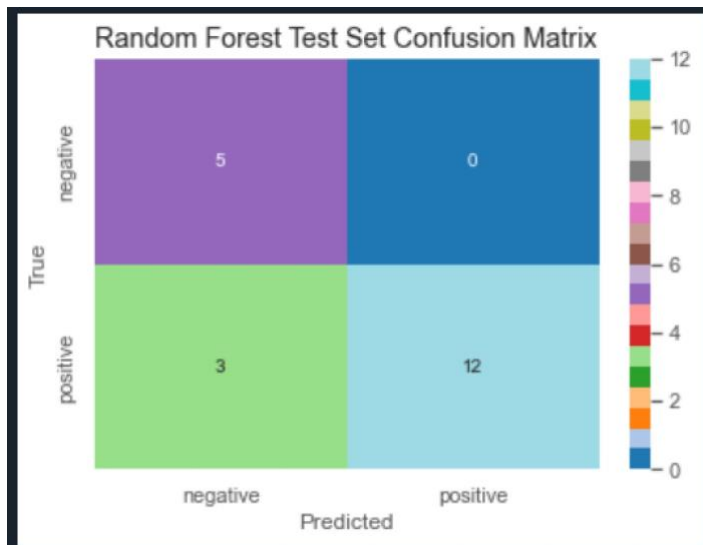
# Plotting Confusion Matrix for Random Forest (Test Set)
plot_confusion_matrix(y_test, rf_predictions, 'Random Forest Test Set Confusion Matrix', ['negative', 'positive'])

# Plotting Confusion Matrix for SVM (Training Set)
svm_train_predictions = svm_classifier.predict(X_train_tfidf)
plot_confusion_matrix(y_train, svm_train_predictions, 'SVM Training Set Confusion Matrix', ['negative', 'positive'])

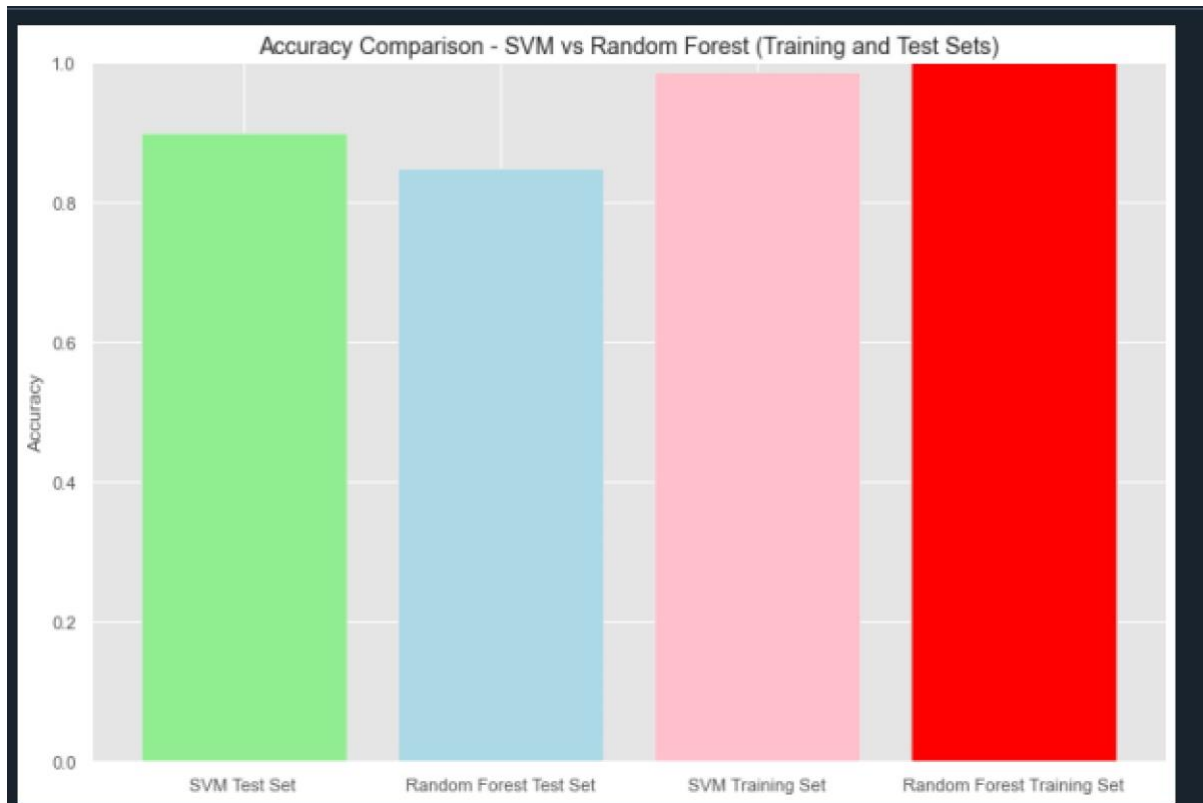
# Plotting Confusion Matrix for Random Forest (Training Set)
rf_train_predictions = rf_classifier.predict(X_train_tfidf)
plot_confusion_matrix(y_train, rf_train_predictions, 'Random Forest Training Set Confusion Matrix', ['negative', 'positive'])

# Plotting Accuracy Comparison
models = ['SVM Test Set', 'Random Forest Test Set', 'SVM Training Set', 'Random Forest Training Set']
accuracies = [accuracy_score(y_test, svm_predictions), accuracy_score(y_test, rf_predictions),
               accuracy_score(y_train, svm_train_predictions), accuracy_score(y_train, rf_train_predictions)]
```





```
plt.figure(figsize=(12, 8))
plt.bar(models, accuracies, color=['lightgreen', 'lightblue', 'pink', 'red'])
plt.ylabel('Accuracy')
plt.title('Accuracy Comparison - SVM vs Random Forest (Training and Test Sets)')
plt.ylim(0, 1)
plt.show()
```



How many times does each user review? Is it positive or negative?

```
#Data Distribution
sns.countplot(x='Sentiment', data=data, palette= ['lightgreen', 'red'])
plt.xlabel('Sentiment')
plt.ylabel('Count')
plt.title('Count of Positive and Negative Reviews')
plt.show()

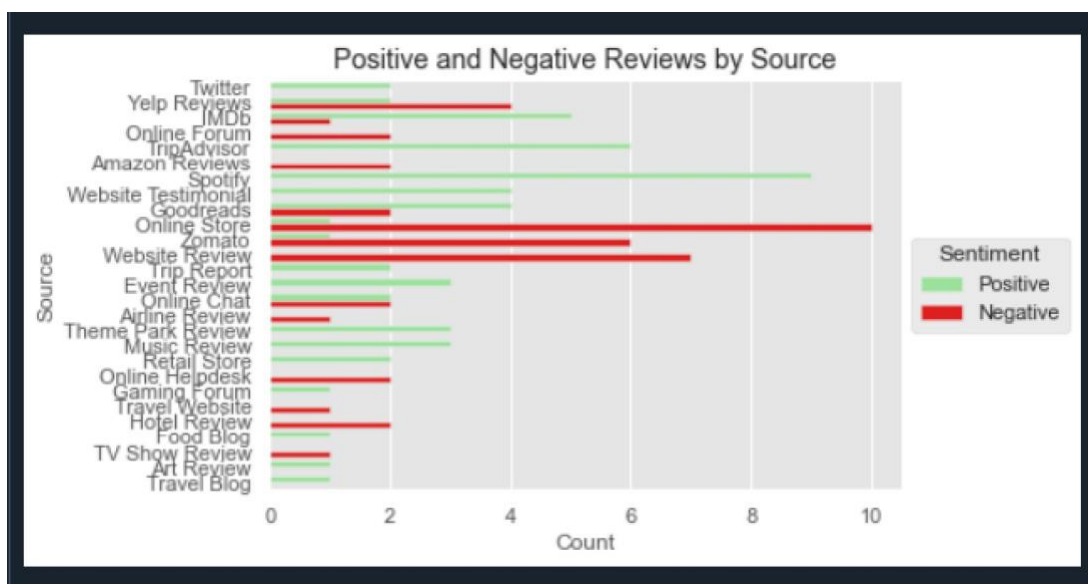
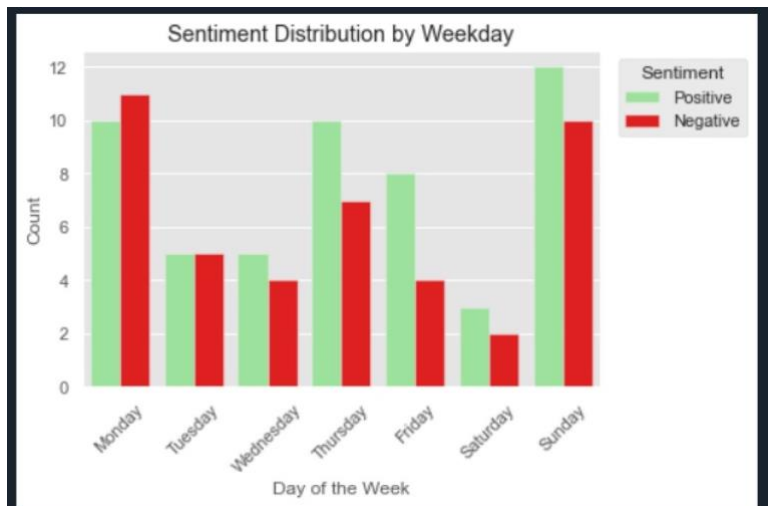
data['Sentiment'] = data['Sentiment'].str.strip()
data['Sentiment'] = data['Sentiment'].str.lower()
data['DayOfWeek'] = data['Date'].dt.day_name()
custom_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
data['DayOfWeek'] = pd.Categorical(data['DayOfWeek'], categories=custom_order, ordered=True)

sns.countplot(x='DayOfWeek', hue='Sentiment', data=data, palette= ['lightgreen', 'red'])

plt.xlabel('Day of the Week')
plt.ylabel('Count')
plt.title('Sentiment Distribution by Weekday')
plt.legend(title='Sentiment', labels=['Positive', 'Negative'], bbox_to_anchor=(1.02, 1), loc='upper left')
plt.xticks(rotation=45)
plt.show()

sns.countplot(y='Source', hue='Sentiment', data=data, palette= ['lightgreen', 'red'])
plt.xlabel('Count')
plt.ylabel('Source')
plt.title('Positive and Negative Reviews by Source')
plt.legend(title='Sentiment', loc='center left', bbox_to_anchor=(1, 0.5), labels=['Positive', 'Negative'])
plt.show()

sns.countplot(y='Location', hue='Sentiment', data=data, palette= ['lightgreen', 'red'])
plt.xlabel('Count')
plt.ylabel('Source')
plt.title('Positive and Negative Reviews by Location')
plt.legend(title='Sentiment', loc='center left', bbox_to_anchor=(1, 0.5), labels=['Positive', 'Negative'])
plt.show()
```



```

#User Distribution
data.describe()

data['User ID'] = data['User ID'].str.strip()

user_reviews = data.groupby('User ID').agg(
    ReviewCount=('Text', 'count'),
    SentimentDistribution=('Sentiment', lambda x: dict(x.value_counts())))
).reset_index()

user_reviews = user_reviews[user_reviews['ReviewCount'] >= 2]

sentiment_data = pd.DataFrame(user_reviews['SentimentDistribution'].to_list())
sentiment_data.index = user_reviews['User ID']
sentiment_data
data[data['User ID'] == 'user456'][['Text', 'Source']]

```

```

tgreen', 'red'])

```

user_reviews - DataFrame

Index	User ID	ReviewCount	SentimentDistribution
0	@user123	2	{'positive': 2}
6	bookworm789	2	{'positive': 2}
17	foodie22	2	{'positive': 2}
20	foodlover123	2	{'negative': 2}
22	foodlover2468	2	{'negative': 2}
23	forumuser1	2	{'negative': 2}
28	moviefan789	2	{'positive': 2}
34	musiclover456	2	{'positive': 2}
35	musiclover789	2	{'positive': 2}
38	nostalgiacat123	2	{'positive': 2}
41	shopper123	2	{'negative': 2}
46	shopper789	2	{'negative': 2}
48	testimonialuser1	2	{'positive': 2}
50	thrillseeker1	2	{'positive': 2}

Format

Resize

☒ Background color

☒ Column min/max

Our Variable Explorer:

custom_order - List (7 elements)

Index	Type	Size	Value
0	str	1	Monday
1	str	1	Tuesday
2	str	1	Wednesday
3	str	1	Thursday
4	str	1	Friday
5	str	1	Saturday
6	str	1	Sunday

stop - Set (179 elements)

Type	Size	Value
str	1	hasn
str	1	how
str	1	was
str	1	in
str	1	ve
str	1	my
str	1	can
str	1	what
str	1	this
str	1	both
str	1	at
str	1	nor
str	1	doesn

X - Series

Index	Text
0	"I love this product!"
1	"The service was terrible."
2	"This movie is amazing!"
3	"I'm so disappointed with their customer suppo...
4	"Just had the best meal of my life!"
5	"The quality of this product is subpar."
6	"I can't stop listening to this song. It's inc...
7	"Their website is so user-friendly. Love it!"
8	"I loved the movie! It was fantastic!"
9	"The customer service was terrible."
10	"This book made me feel inspired. Highly recom...
11	"I'm extremely disappointed with their product...
12	"Just had the most amazing vacation! I can't w...
13	"The food at this restaurant was awful. Never ...
14	"I can't stop listening to this song. It's my ...
15	"Their website is so confusing and poorly desi...
16	"I had an incredible experience at the theme p...
17	"The product arrived damaged. Very disappointe...
18	"The concert was absolutely breathtaking. Best...
19	"I had a terrible experience with their custom...
20	"This movie is a masterpiece! I was blown away...
21	"The customer service at this store is top-not...
22	"I'm disappointed with the ending of this book...
23	"The product I received was damaged. Unaccepta...
24	"Just had the worst flight experience. Delayed...
25	"The hotel stay was absolutely amazing! Luxury...
26	"The food at this restaurant was outstanding. ...
27	"This playlist is my go-to for workouts. Energ...
28	"I had a frustrating experience navigating thr...
29	"The roller coaster ride was thrilling! Heart...
30	"The product I ordered never arrived. Terrible...

y - Series

Index	Sentiment
0	Positive
1	Negative
2	Positive
3	Negative
4	Positive
5	Negative
6	Positive
7	Positive
8	Positive
9	Negative
10	Positive
11	Negative
12	Positive
13	Negative

Format Resize ☒ Background color ☒ Column

to_anchor=(1, 0.5), labels=['Positive', 'Negat

Conclusion:

The accuracy using SVM model was 0.9 (90%) and using Random Forest gave us 0.85 (85%). Thus, we can conclude that SVM is the better option for answering our questions.

Our Code:

```
8
9 import numpy as np
10 import pandas as pd
11 import matplotlib.pyplot as plt
12 import seaborn as sns
13 sns.set()
14 from nltk.corpus import stopwords
15 stop=set(stopwords.words('english'))
16 from nltk.util import ngrams
17 from nltk.tokenize import word_tokenize
18 from sklearn.feature_extraction.text import CountVectorizer
19 from collections import defaultdict
20 from collections import Counter
21 plt.style.use('ggplot')
22 import re
23 import string
24 import warnings
25 warnings.filterwarnings('ignore')
26 from sklearn.model_selection import train_test_split
27 from sklearn.feature_extraction.text import TfidfVectorizer
28 from sklearn.svm import SVC
29 from sklearn.ensemble import RandomForestClassifier
30 from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
31
32 import seaborn as sns
33 import matplotlib.pyplot as plt
34 #
35 #loading the data set
36 data=pd.read_csv('sentiment-analysis.csv')
37 data.head()
38 #separating the data into columns
39 data = data['Text', 'Sentiment', 'Source', 'Date/Time', 'User ID', 'Location', 'Confidence Score'].str.split(' ', expand=True)
40 data.columns = ['Text', 'Sentiment', 'Source', 'Date/Time', 'User ID', 'Location', 'Confidence Score']
41 print('There are {} rows and {} columns in data.'.format(data.shape[0], data.shape[1]))
42
43 data.head(10)
44 #Data Cleaning
45 data.isnull().sum()
46 data.dropna(inplace=True)
47 data.head(10)
48
49 #there are some leading and trailing spaces in 'Date/Time' column, so let's trim them
```

```
49 #there are some leading and trailing spaces in 'Date/Time' column, so let's trim them
50 data['Date/Time'] = data['Date/Time'].str.strip()
51
52 data[['Date', 'Time']] = data['Date/Time'].str.split(' ', expand=True)
53
54 data.drop(columns=['Date/Time'], inplace=True)
55 data['Date'] = pd.to_datetime(data['Date'])
56 data['Time'] = pd.to_datetime(data['Time'], format='%H:%M:%S').dt.time
57 #User Distribution
58 data.describe()
59
60 data['User ID'] = data['User ID'].str.strip()
61
62 user_reviews = data.groupby('User ID').agg(
63     ReviewCount=('Text', 'count'),
64     SentimentDistribution=('Sentiment', lambda x: dict(x.value_counts()))
65 ).reset_index()
66
67 user_reviews = user_reviews[user_reviews['ReviewCount'] >= 2]
68
69 sentiment_data = pd.DataFrame(user_reviews['SentimentDistribution'].to_list())
70 sentiment_data.index = user_reviews['User ID']
71 sentiment_data
72 data[data['User ID'] == 'user456']['Text', 'Source']
73 #ALGORITHMS
74
75 # Preparing the data
76 X = data['Text']
77 y = data['Sentiment']
78
79 # Splitting the data into training and testing sets
80 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
81
82 # Feature extraction using TF-IDF vectorizer
83 tfidf_vectorizer = TfidfVectorizer(max_features=5000)
84 X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
85 X_test_tfidf = tfidf_vectorizer.transform(X_test)
86
87 # Support Vector Machine (SVM)
88 svm_classifier = SVC(kernel='linear', C=1)
89 svm_classifier.fit(X_train_tfidf, y_train)
90
91 # Predictions
92 svm_predictions = svm_classifier.predict(X_test_tfidf)
93
94 # Evaluating SVM
95 svm_accuracy = accuracy_score(y_test, svm_predictions)
96 print("SVM Accuracy:", svm_accuracy)
97 print("\nClassification Report for SVM:\n", classification_report(y_test, svm_predictions))
98 print("\nConfusion Matrix for SVM:\n", confusion_matrix(y_test, svm_predictions))
```

conda base (Python 3.11.4) > > Completions: conda (base)

```

# Evaluating SVM
svm_accuracy = accuracy_score(y_test, svm_predictions)
print("SVM Accuracy:", svm_accuracy)
print("\nClassification Report for SVM:\n", classification_report(y_test, svm_predictions))
print("Confusion Matrix for SVM:\n", confusion_matrix(y_test, svm_predictions))

# Random Forest
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
rf_classifier.fit(X_train_tfidf, y_train)

# Predictions
rf_predictions = rf_classifier.predict(X_test_tfidf)

# Evaluating Random Forest
rf_accuracy = accuracy_score(y_test, rf_predictions)
print("\nRandom Forest Accuracy:", rf_accuracy)
print("\nClassification Report for Random Forest:\n", classification_report(y_test, rf_predictions))
print("Confusion Matrix for Random Forest:\n", confusion_matrix(y_test, rf_predictions))

# ----- plots -----

# Function to plot confusion matrix
def plot_confusion_matrix(y_true, y_pred, title, labels):
    cm = confusion_matrix(y_true, y_pred)
    plt.figure(figsize=(6, 4))
    sns.heatmap(cm, annot=True, fmt='d', cmap='pink', xticklabels=labels, yticklabels=labels)
    plt.title(title)
    plt.xlabel('Predicted')
    plt.ylabel('True')
    plt.show()

# Plotting Confusion Matrix for SVM
plot_confusion_matrix(y_test, svm_predictions, 'SVM Confusion Matrix', ['negative', 'positive'])

# Plotting Confusion Matrix for Random Forest
plot_confusion_matrix(y_test, rf_predictions, 'Random Forest Confusion Matrix', ['negative', 'positive'])

# Plotting Accuracy Comparison
models = ['SVM', 'Random Forest']
accuracies = [svm_accuracy, rf_accuracy]

plt.figure(figsize=(8, 5))
plt.bar(models, accuracies, color=['brown', 'lightblue'])
plt.ylabel('Accuracy')
plt.title('Accuracy Comparison - SVM vs Random Forest')
plt.ylim(0, 1)

```

```

138 plt.figure(figsize=(8, 5))
139 plt.bar(models, accuracies, color=['brown', 'lightblue'])
140 plt.ylabel('Accuracy')
141 plt.title('Accuracy Comparison - SVM vs Random Forest')
142 plt.ylim(0, 1)
143 plt.show()
144 # Function to plot confusion matrix
145 def plot_confusion_matrix(y_true, y_pred, title, labels):
146     cm = confusion_matrix(y_true, y_pred)
147     plt.figure(figsize=(6, 4))
148     sns.heatmap(cm, annot=True, fmt='d', cmap='tab20', xticklabels=labels, yticklabels=labels)
149     plt.title(title)
150     plt.xlabel('Predicted')
151     plt.ylabel('True')
152     plt.show()
153
154 # Plotting Confusion Matrix for SVM (Test Set)
155 plot_confusion_matrix(y_test, svm_predictions, 'SVM Test Set Confusion Matrix', ['negative', 'positive'])
156
157 # Plotting Confusion Matrix for Random Forest (Test Set)
158 plot_confusion_matrix(y_test, rf_predictions, 'Random Forest Test Set Confusion Matrix', ['negative', 'positive'])
159
160 # Plotting Confusion Matrix for SVM (Training Set)
161 svm_train_predictions = svm_classifier.predict(X_train_tfidf)
162 plot_confusion_matrix(y_train, svm_train_predictions, 'SVM Training Set Confusion Matrix', ['negative', 'positive'])
163
164 # Plotting Confusion Matrix for Random Forest (Training Set)
165 rf_train_predictions = rf_classifier.predict(X_train_tfidf)
166 plot_confusion_matrix(y_train, rf_train_predictions, 'Random Forest Training Set Confusion Matrix', ['negative', 'positive'])
167
168 # Plotting Accuracy Comparison
169 models = ['SVM Test Set', 'Random Forest Test Set', 'SVM Training Set', 'Random Forest Training Set']
170 accuracies = [accuracy_score(y_test, svm_predictions), accuracy_score(y_test, rf_predictions),
171               accuracy_score(y_train, svm_train_predictions), accuracy_score(y_train, rf_train_predictions)]
172 plt.figure(figsize=(12, 8))
173 plt.bar(models, accuracies, color=['lightgreen', 'lightblue', 'pink', 'red'])
174 plt.ylabel('Accuracy')
175 plt.title('Accuracy Comparison - SVM vs Random Forest (Training and Test Sets)')
176 plt.ylim(0, 1)
177 plt.show()
178
179 # Data Distribution
180 sns.countplot(x='Sentiment', data=data, palette= ['lightgreen', 'red'])
181 plt.xlabel('Sentiment')
182 plt.ylabel('Count')
183 plt.title('Count of Positive and Negative Reviews')
184 plt.show()
185 data['Sentiment'] = data['Sentiment'].str.strip()
186 data['Sentiment'] = data['Sentiment'].str.lower()

```

```

7
8 # Plotting Accuracy Comparison
9 models = ['SVM Test Set', 'Random Forest Test Set', 'SVM Training Set', 'Random Forest Training Set']
0 accuracies = [accuracy_score(y_test, svm_predictions), accuracy_score(y_test, rf_predictions),
1                 accuracy_score(y_train, svm_train_predictions), accuracy_score(y_train, rf_train_predictions)]
2 plt.figure(figsize=(12, 8))
3 plt.bar(models, accuracies, color=['lightgreen', 'lightblue', 'pink', 'red'])
4 plt.ylabel('Accuracy')
5 plt.title('Accuracy Comparison - SVM vs Random Forest (Training and Test Sets)')
6 plt.ylim(0, 1)
7 plt.show()
8
9 #Data Distribution
0 sns.countplot(x='Sentiment', data=data, palette= ['lightgreen', 'red'])
1 plt.xlabel('Sentiment')
2 plt.ylabel('Count')
3 plt.title('Count of Positive and Negative Reviews')
4 plt.show()
5 data['Sentiment'] = data['Sentiment'].str.strip()
6 data['Sentiment'] = data['Sentiment'].str.lower()
7 data['DayOfWeek'] = data['Date'].dt.day_name()
8 custom_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
9 data['DayOfWeek'] = pd.Categorical(data['DayOfWeek'], categories=custom_order, ordered=True)
0
1 sns.countplot(x='DayOfWeek', hue='Sentiment', data=data, palette= ['lightgreen', 'red'])
2
3 plt.xlabel('Day of the Week')
4 plt.ylabel('Count')
5 plt.title('Sentiment Distribution by Weekday')
6 plt.legend(title='Sentiment', labels=['Positive', 'Negative'], bbox_to_anchor=(1.02, 1), loc='upper left')
7 plt.xticks(rotation=45)
8 plt.show()
9 sns.countplot(y='Source', hue='Sentiment', data=data, palette=['lightgreen', 'red'])
0 plt.xlabel('Count')
1 plt.ylabel('Source')
2 plt.title('Positive and Negative Reviews by Source')
3 plt.legend(title='Sentiment', loc='center left', bbox_to_anchor=(1, 0.5), labels=['Positive', 'Negative'])
4 plt.show()
5
6 sns.countplot(y='Location', hue='Sentiment', data=data, palette=['lightgreen', 'red'])
7 plt.xlabel('Count')
8 plt.ylabel('Source')
9 plt.title('Positive and Negative Reviews by Location')
0 plt.legend(title='Sentiment', loc='center left', bbox_to_anchor=(1, 0.5), labels=['Positive', 'Negative'])
1 plt.show()
2
3

```

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

sns.set()

from nltk.corpus import stopwords

stop=set(stopwords.words('english'))

from nltk.util import ngrams

from nltk.tokenize import word_tokenize

from sklearn.feature_extraction.text import CountVectorizer

from collections import defaultdict

from collections import Counter

plt.style.use('ggplot')

import re

import string

import warnings

warnings.filterwarnings('ignore')

from sklearn.model_selection import train_test_split

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.svm import SVC

```
from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score, classification_report, confusion_matrix


import seaborn as sns

import matplotlib.pyplot as plt

# _____

#loading the data set

data=pd.read_csv('sentiment-analysis.csv')

data.head()

#separating the data into columns

data = data['Text, Sentiment, Source, Date/Time, User ID, Location, Confidence Score'].str.split(' ',
expand=True)

data.columns = ['Text', 'Sentiment', 'Source', 'Date/Time', 'User ID', 'Location', 'Confidence Score']

print('There are { } rows and { } columns in data.'.format(data.shape[0], data.shape[1]))


data.head(10)

#Data Cleaning

data.isnull().sum()

data.dropna(inplace=True)

data.head(10)


#there are some leading and trailing spaces in 'Date/Time' column, so let's trim them

data['Date/Time'] = data['Date/Time'].str.strip()


data[['Date', 'Time']] = data['Date/Time'].str.split(' ', expand=True)


data.drop(columns=['Date/Time'], inplace=True)

data['Date'] = pd.to_datetime(data['Date'])

data['Time'] = pd.to_datetime(data['Time'], format='%H:%M:%S').dt.time

#User Distribution

data.describe()


data['User ID'] = data['User ID'].str.strip()
```

```

user_reviews = data.groupby('User ID').agg(
    ReviewCount=('Text', 'count'),
    SentimentDistribution=('Sentiment', lambda x: dict(x.value_counts()))
).reset_index()

user_reviews = user_reviews[user_reviews['ReviewCount'] >= 2]

sentiment_data = pd.DataFrame(user_reviews['SentimentDistribution'].to_list())
sentiment_data.index = user_reviews['User ID']
sentiment_data
data[data['User ID'] == 'user456'][['Text', 'Source']]
#ALGORITMS

# Preparing the data
X = data['Text']
y = data['Sentiment']

# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Feature extraction using TF-IDF vectorizer
tfidf_vectorizer = TfidfVectorizer(max_features=5000)
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)

# Support Vector Machine (SVM)
svm_classifier = SVC(kernel='linear', C=1)
svm_classifier.fit(X_train_tfidf, y_train)

# Predictions
svm_predictions = svm_classifier.predict(X_test_tfidf)

# Evaluating SVM
svm_accuracy = accuracy_score(y_test, svm_predictions)

```

```

print("SVM Accuracy:", svm_accuracy)

print("\nClassification Report for SVM:\n", classification_report(y_test, svm_predictions))

print("Confusion Matrix for SVM:\n", confusion_matrix(y_test, svm_predictions))


# Random Forest

rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)

rf_classifier.fit(X_train_tfidf, y_train)


# Predictions

rf_predictions = rf_classifier.predict(X_test_tfidf)


# Evaluating Random Forest

rf_accuracy = accuracy_score(y_test, rf_predictions)

print("\nRandom Forest Accuracy:", rf_accuracy)

print("\nClassification Report for Random Forest:\n", classification_report(y_test, rf_predictions))

print("Confusion Matrix for Random Forest:\n", confusion_matrix(y_test, rf_predictions))


#_____plots_____


# Function to plot confusion matrix

def plot_confusion_matrix(y_true, y_pred, title, labels):

    cm = confusion_matrix(y_true, y_pred)

    plt.figure(figsize=(6, 4))

    sns.heatmap(cm, annot=True, fmt='d', cmap='pink', xticklabels=labels, yticklabels=labels)

    plt.title(title)

    plt.xlabel('Predicted')

    plt.ylabel('True')

    plt.show()


# Plotting Confusion Matrix for SVM

plot_confusion_matrix(y_test, svm_predictions, 'SVM Confusion Matrix', ['negative', 'positive'])


# Plotting Confusion Matrix for Random Forest

```



```
plot_confusion_matrix(y_test, rf_predictions, 'Random Forest Confusion Matrix', ['negative', 'positive'])
```

```
# Plotting Accuracy Comparison
```

```
models = ['SVM', 'Random Forest']
```

```
accuracies = [svm_accuracy, rf_accuracy]
```

```
plt.figure(figsize=(8, 5))
```

```
plt.bar(models, accuracies, color=['brown', 'lightblue'])
```

```
plt.ylabel('Accuracy')
```

```
plt.title('Accuracy Comparison - SVM vs Random Forest')
```

```
plt.ylim(0, 1)
```

```
plt.show()
```

```
# Function to plot confusion matrix
```

```
def plot_confusion_matrix(y_true, y_pred, title, labels):
```

```
    cm = confusion_matrix(y_true, y_pred)
```

```
    plt.figure(figsize=(6, 4))
```

```
    sns.heatmap(cm, annot=True, fmt='d', cmap='tab20', xticklabels=labels, yticklabels=labels)
```

```
    plt.title(title)
```

```
    plt.xlabel('Predicted')
```

```
    plt.ylabel('True')
```

```
    plt.show()
```

```
# Plotting Confusion Matrix for SVM (Test Set)
```

```
plot_confusion_matrix(y_test, svm_predictions, 'SVM Test Set Confusion Matrix', ['negative', 'positive'])
```

```
# Plotting Confusion Matrix for Random Forest (Test Set)
```

```
plot_confusion_matrix(y_test, rf_predictions, 'Random Forest Test Set Confusion Matrix', ['negative', 'positive'])
```

```
# Plotting Confusion Matrix for SVM (Training Set)
```

```
svm_train_predictions = svm_classifier.predict(X_train_tfidf)
```

```
plot_confusion_matrix(y_train, svm_train_predictions, 'SVM Training Set Confusion Matrix', ['negative', 'positive'])
```

```

# Plotting Confusion Matrix for Random Forest (Training Set)

rf_train_predictions = rf_classifier.predict(X_train_tfidf)

plot_confusion_matrix(y_train, rf_train_predictions, 'Random Forest Training Set Confusion Matrix',
['negative', 'positive'])


# Plotting Accuracy Comparison

models = ['SVM Test Set', 'Random Forest Test Set', 'SVM Training Set', 'Random Forest Training Set']
accuracies = [accuracy_score(y_test, svm_predictions), accuracy_score(y_test, rf_predictions),
               accuracy_score(y_train, svm_train_predictions), accuracy_score(y_train, rf_train_predictions)]

plt.figure(figsize=(12, 8))

plt.bar(models, accuracies, color=['lightgreen', 'lightblue', 'pink', 'red'])

plt.ylabel('Accuracy')

plt.title('Accuracy Comparison - SVM vs Random Forest (Training and Test Sets)')

plt.ylim(0, 1)

plt.show()


#Data Distribution

sns.countplot(x='Sentiment', data=data, palette= ['lightgreen', 'red'])

plt.xlabel('Sentiment')

plt.ylabel('Count')

plt.title('Count of Positive and Negative Reviews')

plt.show()

data['Sentiment'] = data['Sentiment'].str.strip()

data['Sentiment'] = data['Sentiment'].str.lower()

data['DayOfWeek'] = data['Date'].dt.day_name()

custom_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']

data['DayOfWeek'] = pd.Categorical(data['DayOfWeek'], categories=custom_order, ordered=True)


sns.countplot(x='DayOfWeek', hue='Sentiment', data=data, palette= ['lightgreen', 'red'])


plt.xlabel('Day of the Week')

plt.ylabel('Count')

plt.title('Sentiment Distribution by Weekday')

plt.legend(title='Sentiment', labels=['Positive', 'Negative'], bbox_to_anchor=(1.02, 1), loc='upper left')

plt.xticks(rotation=45)

```

```
plt.show()

sns.countplot(y='Source', hue='Sentiment', data=data, palette=['lightgreen', 'red'])

plt.xlabel('Count')

plt.ylabel('Source')

plt.title('Positive and Negative Reviews by Source')

plt.legend(title='Sentiment', loc='center left', bbox_to_anchor=(1, 0.5), labels=['Positive', 'Negative'])

plt.show()
```

```
sns.countplot(y='Location', hue='Sentiment', data=data, palette=['lightgreen', 'red'])

plt.xlabel('Count')

plt.ylabel('Source')

plt.title('Positive and Negative Reviews by Location')

plt.legend(title='Sentiment', loc='center left', bbox_to_anchor=(1, 0.5), labels=['Positive', 'Negative'])

plt.show()
```