TITLE : DECODING EMOTION THROUGH SENTIMENT ANALYSIS OF SOCIAL MEDIA COVERSATION

OUTPUT

Upload the dataset

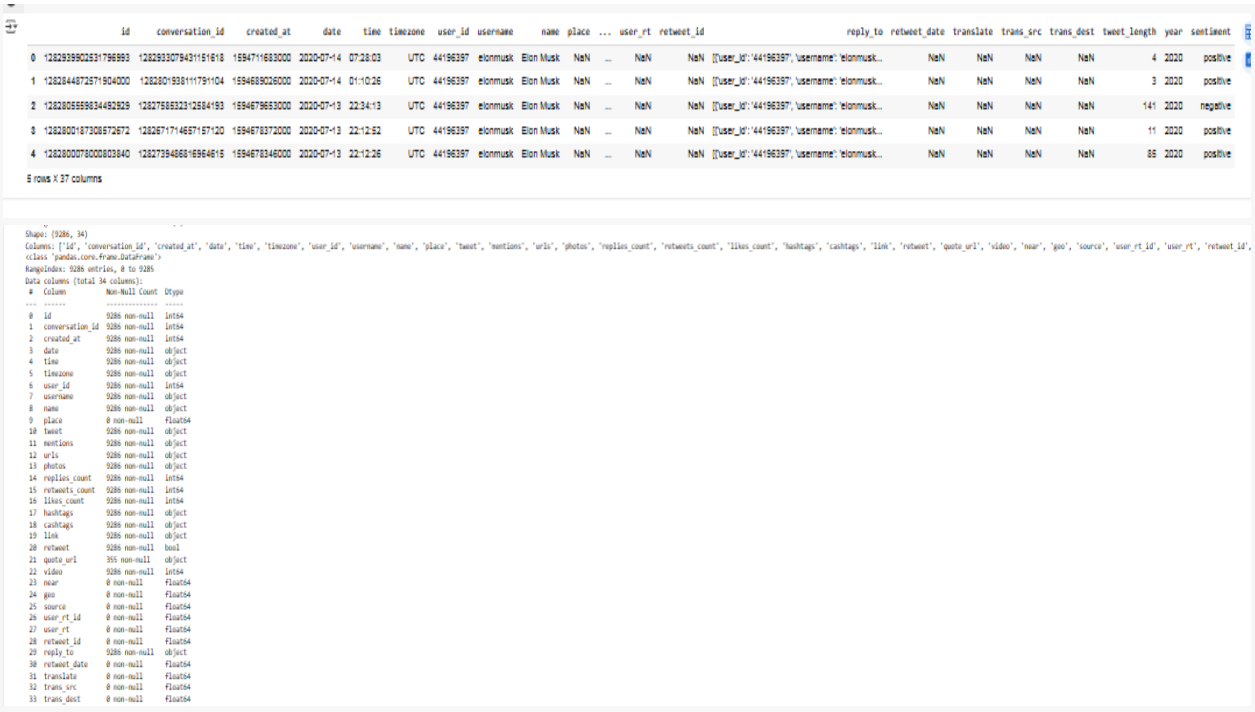
```
# Step 1: Upload
Dataset (Only for Google Colab)
from google.colab import files
uploaded = files.upload()
```

Load the dataset

```
# Step 2: Load the CSV
import pandas as pd
df = pd.read_csv('/content/drive/MyDrive/Elon
musk_dataset.csv/elonmusk_dataset.csv')  # Replace with actual filename if
different
```

Data Exploration

```
# Step 3: Basic Info
df.head()
print("Shape:", df.shape)
print("Columns:", df.columns.tolist())
df.info()
```



	id	conversation_id	created_at	date	time	timezone	user_id	username	name	place	...	user_rt	retweet_id	reply_to	retweet_date	translate	trans_src	trans_dest	tweet_length	year	sentiment
0	1282939902931796993	1282933079431151618	1594711683000	2020-07-14	07:28:03	UTC	44196397	elonmusk	Elon Musk	NaN	...	NaN	NaN	[{'user_id': '44196397', 'username': 'elonmusk...	NaN	NaN	NaN	NaN	4	2020	positive
1	1282844872571904000	1282801938111791104	1594689026000	2020-07-14	01:10:26	UTC	44196397	elonmusk	Elon Musk	NaN	...	NaN	NaN	[{'user_id': '44196397', 'username': 'elonmusk...	NaN	NaN	NaN	NaN	3	2020	positive
2	1282805569834492929	1282758532312584193	1594679653000	2020-07-13	22:34:13	UTC	44196397	elonmusk	Elon Musk	NaN	...	NaN	NaN	[{'user_id': '44196397', 'username': 'elonmusk...	NaN	NaN	NaN	NaN	141	2020	negative
3	1282800187308572672	1282671714687187120	1594678372000	2020-07-13	22:12:52	UTC	44196397	elonmusk	Elon Musk	NaN	...	NaN	NaN	[{'user_id': '44196397', 'username': 'elonmusk...	NaN	NaN	NaN	NaN	11	2020	positive
4	1282800078000803840	1282739486816964615	1594678346000	2020-07-13	22:12:26	UTC	44196397	elonmusk	Elon Musk	NaN	...	NaN	NaN	[{'user_id': '44196397', 'username': 'elonmusk...	NaN	NaN	NaN	NaN	88	2020	positive

5 rows × 37 columns

```
Shape: (9286, 34)
Columns: ['id', 'conversation_id', 'created_at', 'date', 'time', 'timezone', 'user_id', 'username', 'name', 'place', 'tweet', 'mentions', 'urls', 'photos', 'replies_count', 'retweets_count', 'likes_count', 'hashtags', 'cashtags', 'link', 'retweet', 'quote_url', 'video', 'near', 'geo', 'source', 'user_rt_id', 'user_rt', 'retweet_id',
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9286 entries, 0 to 9285
Data columns (total 34 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   id              9286 non-null   int64
 1   conversation_id 9286 non-null   int64
 2   created_at      9286 non-null   int64
 3   date            9286 non-null   object
 4   time            9286 non-null   object
 5   timezone        9286 non-null   object
 6   user_id         9286 non-null   int64
 7   username        9286 non-null   object
 8   name            9286 non-null   object
 9   place           0 non-null      float64
 10  tweet           9286 non-null   object
 11  mentions        9286 non-null   object
 12  urls            9286 non-null   object
 13  photos          9286 non-null   object
 14  replies_count   9286 non-null   int64
 15  retweets_count  9286 non-null   int64
 16  likes_count     9286 non-null   int64
 17  hashtags        9286 non-null   object
 18  cashtags        9286 non-null   object
 19  link            9286 non-null   object
 20  retweet         9286 non-null   bool
 21  quote_url       355 non-null    object
 22  video           9286 non-null   int64
 23  near            0 non-null      float64
 24  geo             0 non-null      float64
 25  source          0 non-null      float64
 26  user_rt_id      0 non-null      float64
 27  user_rt         0 non-null      float64
 28  retweet_id      0 non-null      float64
 29  reply_to        9286 non-null   object
 30  retweet_date    0 non-null      float64
 31  translate       0 non-null      float64
 32  trans_src       0 non-null      float64
 33  trans_dest      0 non-null      float64
```

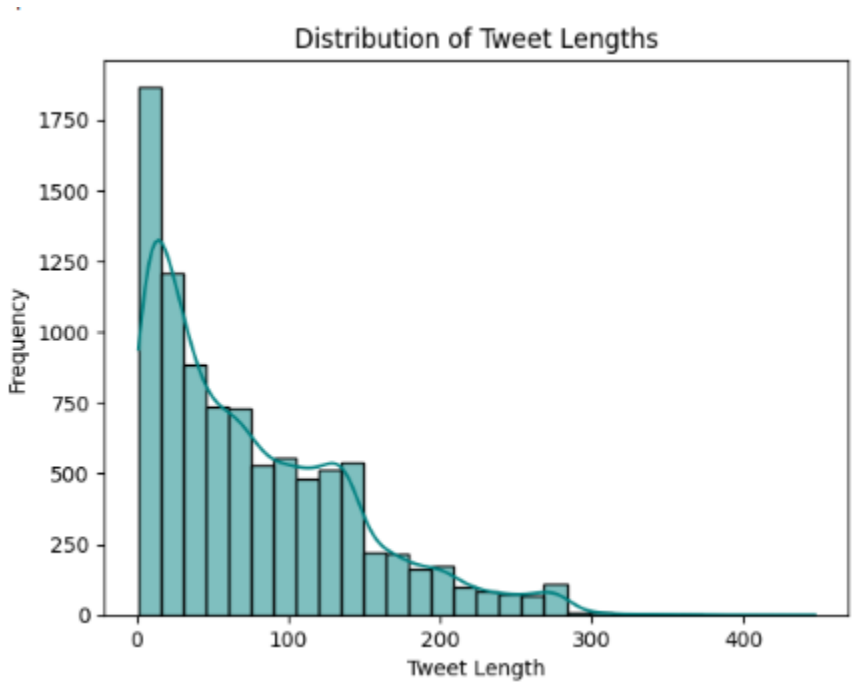
Checking for Missing Values

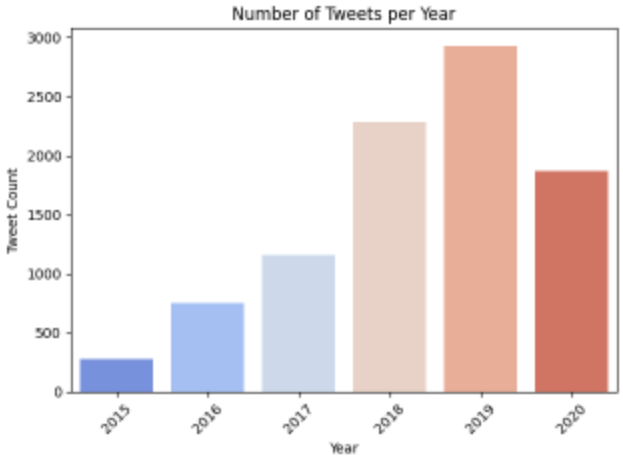
```
# Step 4: Missing values and duplicates
print("Missing values:\n", df.isnull().sum())
print("Duplicate rows:", df.duplicated()
```

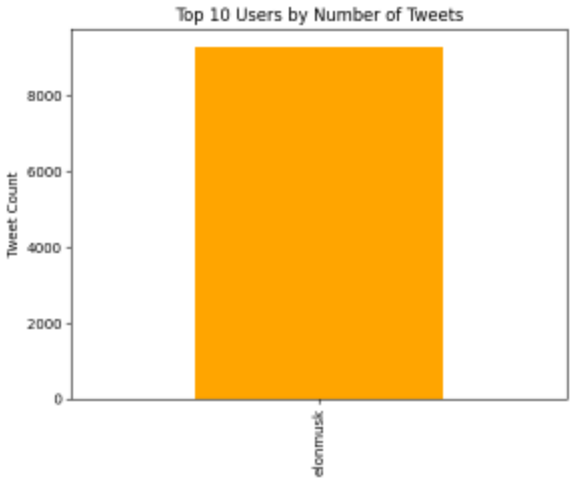
```
Missing values:
 id                  0
conversation_id       0
created_at            0
date                  0
time                  0
timezone              0
user_id               0
username              0
name                  0
place              9286
tweet                 0
mentions              0
urls                  0
photos                0
replies_count         0
retweets_count        0
likes_count           0
hashtags              0
cashtags              0
link                  0
retweet               0
quote_url          8931
video                 0
near               9286
geo                9286
source             9286
user_rt_id         9286
user_rt            9286
retweet_id         9286
reply_to              0
retweet_date       9286
translate          9286
trans_src          9286
trans_dest         9286
dtype: int64
Duplicate rows: 0
```

Visualize a New Features

```
# Step 5: Visualizations
import seaborn as sns
import matplotlib.pyplot as plt
df['tweet_length'] = df['tweet'].astype(str).apply(len)
# Tweet length distribution

sns.histplot(df['tweet_length'], kde=True, bins=30, color='teal')

plt.title('Distribution of Tweet Lengths')
plt.xlabel('Tweet Length')
plt.ylabel('Frequency')
plt.show()


# Tweets per year
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```







Top usernames

```
top_users = df['username'].value_counts().head(10)
top_users.plot(kind='bar', color='orange')
plt.title('Top 10 Users by Number of Tweets')
plt.xlabel('Username')
plt.ylabel('Tweet Count')
plt.show()
```

Sentiment Analysis using TextBlob

```
!pip install -q textblob
from textblob import TextBlob

def get_sentiment(text):
    polarity = TextBlob(str(text)).sentiment.polarity
```

```
    if polarity > 0:
        return 'positive'
    elif polarity == 0:
        return 'neutral'
    else:
        return 'negative'

df['sentiment'] = df['tweet'].apply(get_sentiment)
print(df[['tweet', 'sentiment']].head())
```

Train-Test Split

```
# Step 7: Train-Test Split
from sklearn.model_selection import train_test_split
X = df['tweet'].astype(str)
y = df['sentiment']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
stratify=y, random_state=42)
```

TF-IDF + Logistic Regression Pipeline

```
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix

model = Pipeline([
    ('tfidf', TfidfVectorizer(max_features=5000, stop_words='english')),
    ('clf', LogisticRegression(max_iter=1000))
])

model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

Evaluation

```
# Step 9: Evaluation
print("Accuracy:", accuracy_score(y_test, y_pred))
print ("Classification Report:\n", classification_report(y_test, y_pred))
```

✅ Accuracy: 0.7981700753498385

Test on new tweets

```
new_tweets = [
    "Tesla's new update is amazing!",
    "I'm not happy with Twitter's new algorithm.",
    "SpaceX launch was a huge success!",
    "This is just disappointing."
]
predictions = model.predict(new_tweets)
for tweet, sentiment in zip(new_tweets, predictions):
    print(f"Tweet: {tweet}\nPredicted Sentiment: {sentiment}\n")
```

```
Tweet: Tesla's new update is amazing!
Predicted Sentiment: positive

Tweet: I'm not happy with Twitter's new algorithm.
Predicted Sentiment: positive

Tweet: SpaceX launch was a huge success!
Predicted Sentiment: positive

Tweet: This is just disappointing.
Predicted Sentiment: positive
```

Gradio Web App

```
# Step 11: Gradio Web App
!pip install -q gradio
import gradio as gr

def predict_sentiment(tweet):
    return model.predict([tweet])[0]

iface = gr.Interface(
    fn=predict_sentiment,
    inputs=gr.Textbox(lines=3, placeholder="Enter a tweet..."),
    outputs="text",
    title="Elon Musk Tweet Sentiment Analyzer",
    description="Enter a tweet related to Elon Musk to classify sentiment as
Positive, Neutral, or Negative."
)
iface.launch()
```

```
It looks like you are running Gradio on a hosted a Jupyter notebook. For the Gradio app to work, sharing must be enabled. Automatically setting `share=True` (you can turn this off by setting `share=False` in `launch()` explicitly).

Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://8962889e53d6fe7d9c.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working directory to deploy to Hugging Face Spaces (https://huggingface.co/spaces)
```