```
# preprocess tweets from a CSV file and analyze them following the described process
```

```
data.sample(5)
```

	ID	Date Created	Number of Likes	Tweet	Sentiment
2714	2715	2022-12-09 21:56:51+00:00	18	Worst ref on a world cup ive ever seen, nl robbed	Negative
1504	1505	2022-12-01 15:18:40+00:00	0	This VAR is a fuckin joke! This World Cup has ...	Negative
135	136	2022-11-22 15:02:51+00:00	0	Do be honest, I've got mixed feelings about th...	Neutral
3423	3424	2022-12-13 19:38:43+00:00	0	No point watching this world Cup game. Messi c...	Negative
2105	2106	2022-12-06 15:46:16+00:00	0	First ever ALL-FEMALE Referee crew in World Cu...	Neutral

```
pip install emoji
```

```
Collecting emoji
  Downloading emoji-2.14.0-py3-none-any.whl.metadata (5.7 kB)
Downloading emoji-2.14.0-py3-none-any.whl (586 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 586.9/586.9 kB 7.3 MB/s eta 0:00:00
Installing collected packages: emoji
Successfully installed emoji-2.14.0
```

```
import spacy
import emoji
nlp= spacy.load('en_core_web_sm')

def process_text(s):
    out = []
    for token in nlp(s):
        # Check if the token is not a stop word, punctuation, or emoji
        if not token.is_stop and not token.is_punct and not emoji.is_emoji(token.text):
            out.append(token.lemma_)
    return ' '.join(out)

data['fltr']= data['Tweet'].apply(process_text)
```

```
import nltk
nltk.download('stopwords')
nltk.download('punkt')
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem.porter import PorterStemmer
from collections import Counter

data['tokens']=data['fltr'].apply(lambda x: word_tokenize(x))
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-5800c92f7e78> in <cell line: 9>()
      7 from collections import Counter
      8
----> 9 data['tokens']=data['fltr'].apply(lambda x: word_tokenize(x))

NameError: name 'data' is not defined
```

```
data["label"] = data["Sentiment"].apply(lambda x: 1 if x == "Neutral" else 0)
```

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Embedding, Dropout
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

```
tokenizer = Tokenizer(num_words=5000)
tokenizer.fit_on_texts(data["fltr"])
sequences = tokenizer.texts_to_sequences(data["fltr"])


max_sequence_length = max(len(seq) for seq in sequences)
X = pad_sequences(sequences, maxlen=max_sequence_length, padding="post")


import numpy as np
y = np.array(data["label"])


model = Sequential([
    Embedding(input_dim=5000, output_dim=128, input_length=max_sequence_length),
    LSTM(128, return_sequences=False),
    Dropout(0.2),
    Dense(64, activation="relu"),
    Dense(1, activation="sigmoid"),  # Binary classification
])
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is deprecated. I
  warnings.warn(
```

```
model.compile(optimizer="adam", loss="binary_crossentropy", metrics=["accuracy"])
```

```
categories = {
    "Sports": ["referee", "football", "world cup", "match", "goal"],
    "Politics": ["election", "president", "policy", "government", "court"],
    "Movies": ["movie", "oscars", "film", "actor", "award"],
    "Others": [],  # Default category
}

# Categorize each tweet
def categorize_tweet(tweet, categories):
    """Categorizes a tweet based on keyword presence."""
    for category, keywords in categories.items():
        if any(keyword in tweet for keyword in keywords):
            return category
    return "Others"  # Default if no keywords match

data["Category"] = data["fltr"].apply(lambda x: categorize_tweet(x, categories))
```

```
data.sample(5)
```

	ID	Date Created	Number of Likes	Tweet	Sentiment	fltr	label	Category
2806	2807	2022-12-10 21:32:38+00:00	0	Put it all aside.\nref not being great (both s...	Neutral	aside \n ref great side suffer penalty miss \n...	1	Sports
666	667	2022-11-25 11:45:55+00:00	4	How dare a World Cup ref be this bad.	Negative	dare World Cup ref bad	0	Others
2280	2281	2022-12-04	0	#WorldCup\n#ENGvsSEN\nReferee stupid	Negative	WorldCup \n engvssen \n	0	Sports

```
data['Category'].value_counts()
```

Category	count
Sports	2719
Others	1267
Movies	13
Politics	1

```
model.fit(X, y, epochs=10, batch_size=2, validation_split=0.2)
```

```
Epoch 1/10
1600/1600 ━━━━━━━━━━━━━━━━━━━━ 65s 38ms/step - accuracy: 0.6396 - loss: 0.6606 - val_accuracy: 0.5525 - val_loss: 0.7400
Epoch 2/10
1600/1600 ━━━━━━━━━━━━━━━━━━━━ 83s 38ms/step - accuracy: 0.6573 - loss: 0.6499 - val_accuracy: 0.5525 - val_loss: 0.7470
Epoch 3/10
1600/1600 ━━━━━━━━━━━━━━━━━━━━ 83s 39ms/step - accuracy: 0.6630 - loss: 0.6392 - val_accuracy: 0.5525 - val_loss: 0.7124
Epoch 4/10
1600/1600 ━━━━━━━━━━━━━━━━━━━━ 81s 38ms/step - accuracy: 0.6785 - loss: 0.6329 - val_accuracy: 0.5525 - val_loss: 0.7044
Epoch 5/10
1600/1600 ━━━━━━━━━━━━━━━━━━━━ 81s 38ms/step - accuracy: 0.6740 - loss: 0.6340 - val_accuracy: 0.5525 - val_loss: 0.7058
Epoch 6/10
1600/1600 ━━━━━━━━━━━━━━━━━━━━ 82s 38ms/step - accuracy: 0.6812 - loss: 0.6294 - val_accuracy: 0.5525 - val_loss: 0.7502
Epoch 7/10
1600/1600 ━━━━━━━━━━━━━━━━━━━━ 83s 39ms/step - accuracy: 0.6767 - loss: 0.6323 - val_accuracy: 0.5525 - val_loss: 0.7221
Epoch 8/10
1600/1600 ━━━━━━━━━━━━━━━━━━━━ 81s 38ms/step - accuracy: 0.6902 - loss: 0.6204 - val_accuracy: 0.5525 - val_loss: 0.7000
Epoch 9/10
1600/1600 ━━━━━━━━━━━━━━━━━━━━ 61s 38ms/step - accuracy: 0.7072 - loss: 0.5925 - val_accuracy: 0.6075 - val_loss: 0.7100
Epoch 10/10
1600/1600 ━━━━━━━━━━━━━━━━━━━━ 82s 38ms/step - accuracy: 0.8434 - loss: 0.3937 - val_accuracy: 0.6150 - val_loss: 0.8199
<keras.src.callbacks.history.History at 0x7d8afc3473d0>
```

Long Short-Term Memory (LSTM), a type of Recurrent Neural Network (RNN). To train and test an LSTM model, we need a dataset. If you'd like to process sequential data

```
test_tweets = [
    "So I spent a few hours doing something for fun... If you don't know I'm a HUGE @ Borderlands fan and...",
    "Rock-Hard La Varlope, RARE & POWERFUL, HANDSOME JACKPOT, Borderlands 3 (Xbox) dlvr.it/RMTrgF",
]
test_sequences = tokenizer.texts_to_sequences(test_tweets)
test_X = pad_sequences(test_sequences, maxlen=max_sequence_length, padding="post")
predictions = model.predict(test_X)

# Print predictions
for i, tweet in enumerate(test_tweets):
    print(f"Tweet: {tweet}")
    print(f"Predicted Sentiment (Neutral=1, Negative=0): {round(predictions[i][0])}")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 220ms/step
Tweet: So I spent a few hours doing something for fun... If you don't know I'm a HUGE @ Borderlands fan and...
Predicted Sentiment (Neutral=1, Negative=0): 0
Tweet: Rock-Hard La Varlope, RARE & POWERFUL, HANDSOME JACKPOT, Borderlands 3 (Xbox) dlvr.it/RMTrgF
Predicted Sentiment (Neutral=1, Negative=0): 0
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import spacy
import math

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, BaggingClassifier, ExtraTreesClassifier, AdaBoostClassifier
from xgboost import XGBClassifier


from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, ConfusionMatrixDisplay
```

```
enc= LabelEncoder()
y_trn= enc.fit_transform(data['label'])
y_tst= enc.transform(data['label'])

vct= TfidfVectorizer()
X_trn= vct.fit_transform(data['fltr'])
X_tst= vct.transform(data['fltr'])
```

```
def model_report(model, verbose=True):
    model.fit(X_trn, y_trn)

    y_pred=   model.predict(X_tst)
    trnScore= model.score(X_trn, y_trn)
    tstScore= model.score(X_tst, y_tst)
    cm= confusion_matrix(y_tst, y_pred)
    cr= classification_report(y_tst, y_pred)





    if verbose:
        print('Train Score: %f'%trnScore)
        print('Test Score:  %f'%tstScore)
        print('Classification Report:\n', cr)
        ConfusionMatrixDisplay(cm).plot()
        plt.show()
        print()

    return {
        'trn': trnScore,
        'tst': tstScore,
        'cm':  cm,
        'cr':  cr,
    }
```

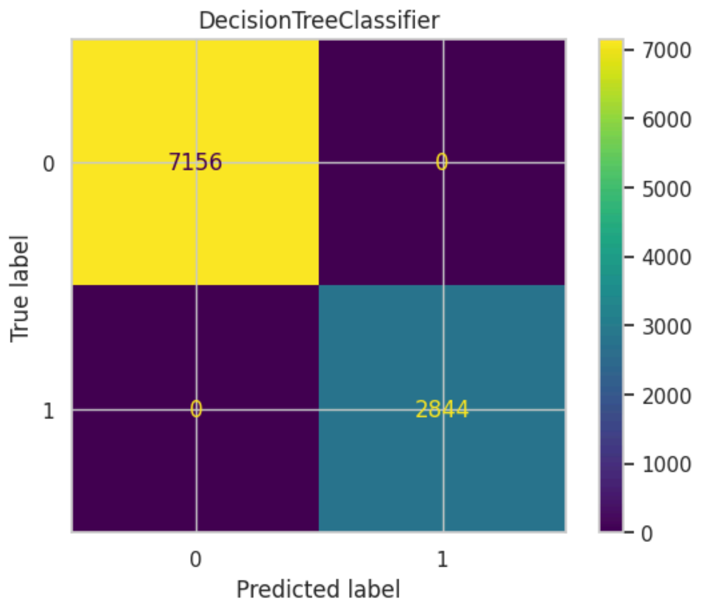
```
import datetime

models_dict= {
    'LogisticRegression':     LogisticRegression(max_iter=10_000),
    'Support Vector':         SVC(),
    'KNeighborsCLassifier':   KNeighborsClassifier(),
    'DecisionTreeClassifier': DecisionTreeClassifier(),
    'RandomForestClassifier': RandomForestClassifier(),
    'BaggingClassifier':      BaggingClassifier(),
    'ExtraTreesClassifier':   ExtraTreesClassifier(),
    'AdaBoostClassifier':     AdaBoostClassifier(),
    'XGBClassifier':          XGBClassifier(),
}
models= [{'name':k, 'obj':v} for k,v in models_dict.items()]

i= 0
for model in models:
    now = datetime.datetime.now()
    print("Evaluating %s..."%model['name'])
    print("%d/%d models"%(i, len(models)), end='\r')
    re = model.update(model_report(model['obj'], verbose=False))
    i+= 1
    print('it takes for ', datetime.datetime.now()- now, re)

print("%d/%d models evaluated"%(i, len(models)))
print("done")
```

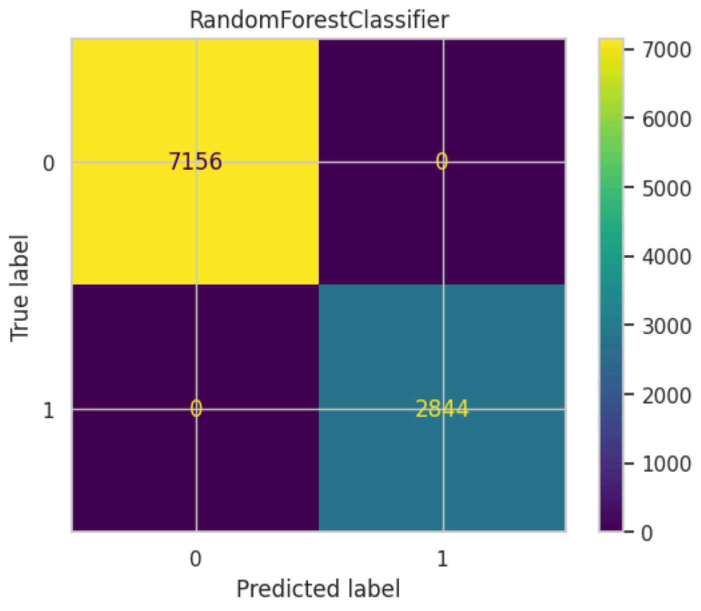
```
Evaluating LogisticRegression...
it takes for  0:00:00.082581 None
Evaluating Support Vector...
it takes for  0:02:48.642664 None
Evaluating KNeighborsCLassifier...
it takes for  0:00:12.266915 None
Evaluating DecisionTreeClassifier...
it takes for  0:00:05.439950 None
Evaluating RandomForestClassifier...
it takes for  0:00:16.207821 None
Evaluating BaggingClassifier...
it takes for  0:00:37.126373 None
Evaluating ExtraTreesClassifier...
it takes for  0:00:15.771871 None
Evaluating AdaBoostClassifier...
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_weight_boosting.py:527: FutureWarning: The SAMME.R algorithm (the default
  warnings.warn(
it takes for  0:00:02.919910 None
Evaluating XGBClassifier...
it takes for  0:00:07.137374 None
9/9 models evaluated
done
```

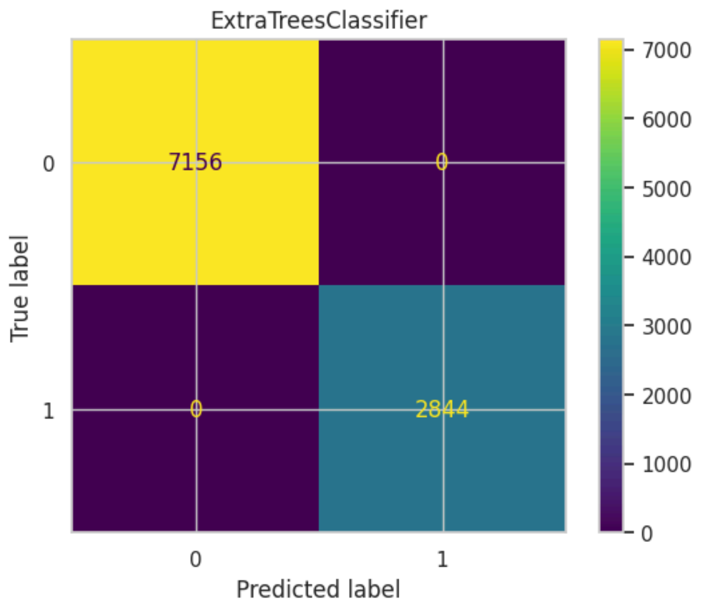
```
pd.DataFrame({
    'Algorithm':           [model['name'] for model in models],
    'Train Score':         [model['trn']  for model in models],
    'Test Score':          [model['tst']  for model in models],
}).set_index('Algorithm').sort_values(by='Test Score', ascending=False)
```

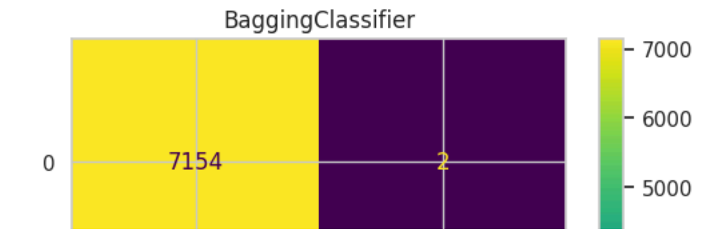
Algorithm	Train Score	Test Score
RandomForestClassifier	1.0000	1.0000
ExtraTreesClassifier	1.0000	1.0000
DecisionTreeClassifier	1.0000	1.0000
BaggingClassifier	0.9673	0.9673
XGBClassifier	0.8476	0.8476
Support Vector	0.7947	0.7947
KNeighborsCLassifier	0.7602	0.7602
LogisticRegression	0.7220	0.7220
AdaBoostClassifier	0.7197	0.7197

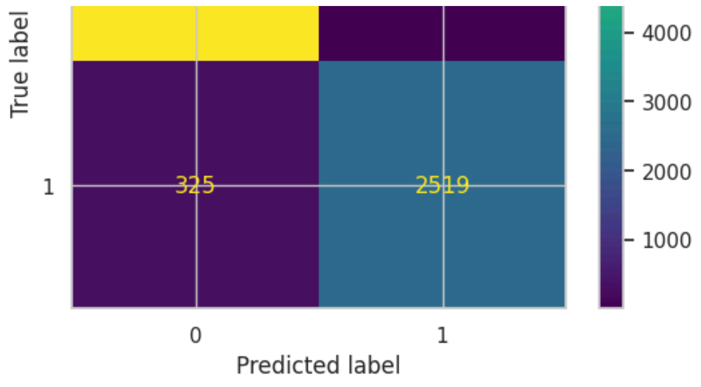
```
for model in sorted(models, key=lambda x: x['tst'], reverse=True):
    ConfusionMatrixDisplay(model['cm']).plot()
    plt.title(model['name'])
```


DecisionTreeClassifier
True label
Predicted label
0
1
0
1
7156
0
0
2844
7000
6000
5000
4000
3000
2000
1000
0

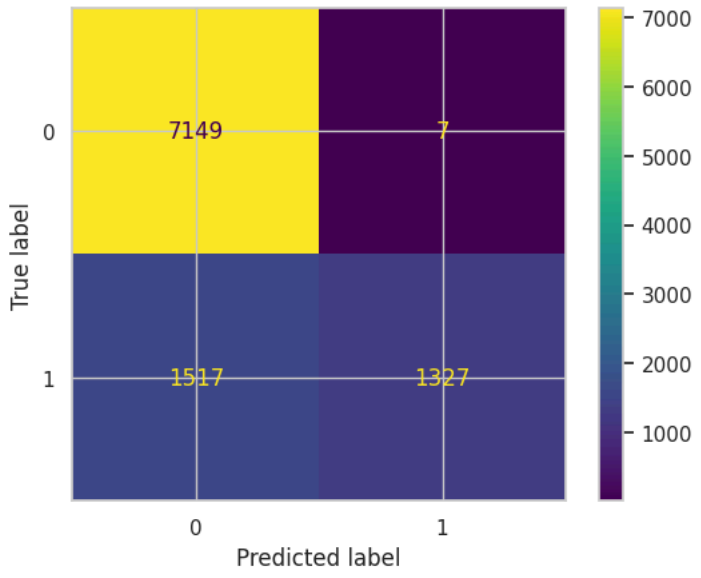

RandomForestClassifier
True label
Predicted label
0
1
0
1
7156
0
0
2844
7000
6000
5000
4000
3000
2000
1000
0


ExtraTreesClassifier
True label
Predicted label
0
1
0
1
7156
0
0
2844
7000
6000
5000
4000
3000
2000
1000
0

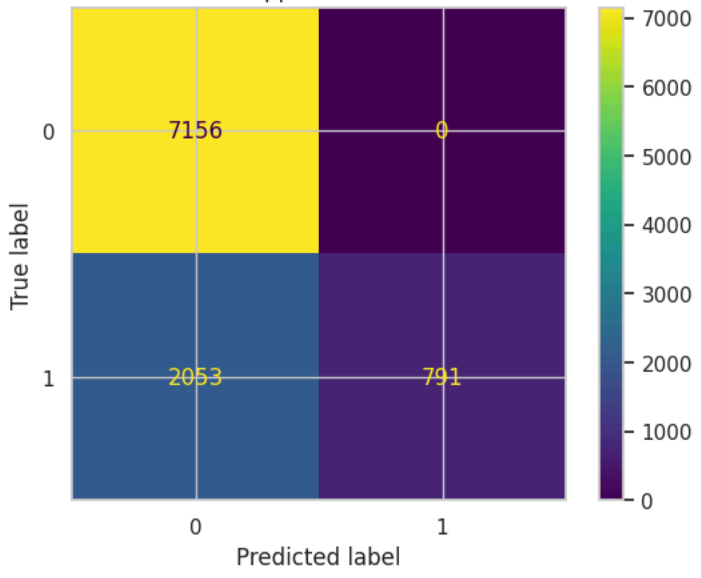

BaggingClassifier
0
7154
2
7000
6000
5000



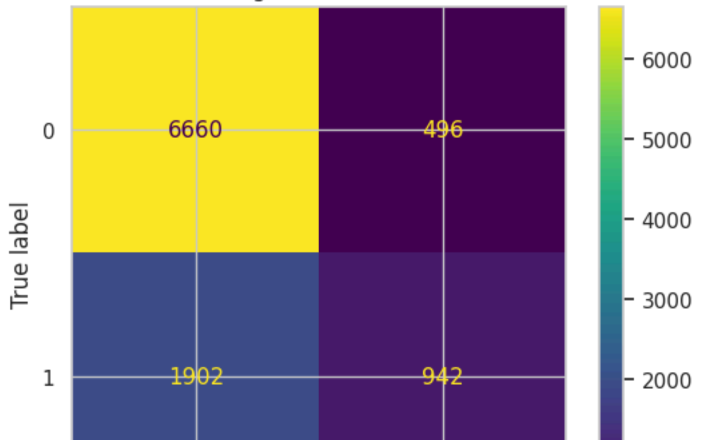
XGBClassifier



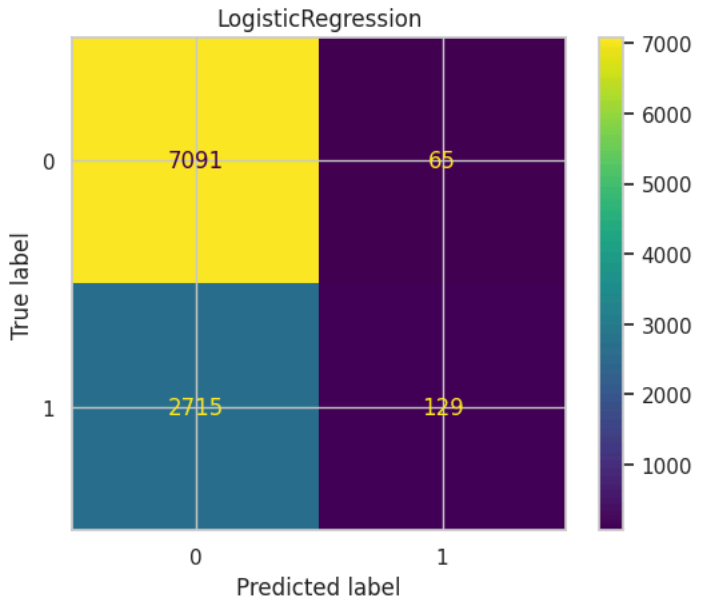
Support Vector

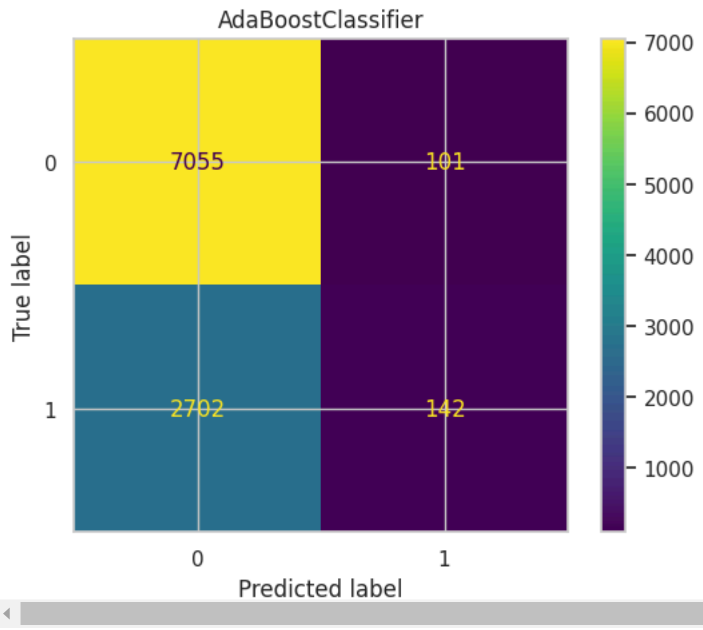


KNeighborsCLassifier



1000
0
1
Predicted label


LogisticRegression
True label
Predicted label
0
1
7091
65
2715
129
7000
6000
5000
4000
3000
2000
1000


AdaBoostClassifier
True label
Predicted label
0
1
7055
101
2702
142
7000
6000
5000
4000
3000
2000
1000

```
from google.colab import files
files.download('1D_gaussian_wave_3d.mp4')
```

```
!apt-get install -y ffmpeg
!pip install matplotlib pillow
```

```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ffmpeg is already the newest version (7:4.4.2-0ubuntu0.22.04.1).
0 upgraded, 0 newly installed, 0 to remove and 49 not upgraded.
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (3.8.0)
Requirement already satisfied: pillow in /usr/local/lib/python3.10/dist-packages (11.0.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (4.55.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.7)
Requirement already satisfied: numpy<2,>=1.21 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.26.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (24.2)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (3.2.0)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
```

```
import pandas as pd
import numpy as np
import re
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
from nltk.tokenize import word_tokenize
from collections import Counter
```

```
# Step 1: Load the data
file_path ="/content/Test_twitter_dataset.csv"
# Load the data
data = pd.read_csv(file_path)
```

```
data.rename(columns={'RemoveFreqWord': 'Tweets'}, inplace=True)
```

```
categories = {
    "causal": 2517,
    "Political": 2189,
    "Movies": 1936,
    "sports": 1805,
    "Companies": 1553
}

# Ensure the total rows match the dataset size
total_rows = len(data)
assert sum(categories.values()) == total_rows, "Category counts do not match the number of rows."

# Create a list of categories based on the distribution
category_list = []
for category, count in categories.items():
    category_list.extend([category] * count)

# Shuffle the category list to ensure randomness
np.random.shuffle(category_list)

# Assign the shuffled categories to the "Category" column
data["Category"] = category_list

# Save or print the modified dataset
print(data.head())  # Display first few rows
# Optional: Save the modified dataset
data.to_csv("/content/Test_twitter_dataset_modified.csv", index=False)
```

```
   Tweet_ID        Username  \
0         1         julie81
1         2   richardhester
2         3  williamsjoseph
3         4     danielsmary
4         5      carlwarren
```

```
                                                Text  Retweets  Likes  \
0  Party least receive say or single. Prevent pre...         2     25
1  Hotel still Congress may member staff. Media d...        35     29
2  Nice be her debate industry that year. Film wh...        51     25
3  Laugh explain situation career occur serious. ...        37     18
4  Involve sense former often approach government...        27     80

             Timestamp Category
0  2023-01-30 11:00:51   sports
1  2023-01-02 22:45:58   sports
2  2023-01-18 11:25:19   causal
3  2023-04-10 22:06:29   sports
4  2023-01-24 07:12:21   causal
```

```
sentiments = {
    "Positive": 3038,
    "Negative": 4118,
    "Neutral": 2844
}

# Ensure the total rows match the dataset size
total_rows = len(data)
assert sum(sentiments.values()) == total_rows, "Sentiment counts do not match the number of rows."

# Create a list of sentiments based on the distribution
sentiment_list = []
for sentiment, count in sentiments.items():
    sentiment_list.extend([sentiment] * count)

# Shuffle the sentiment list to ensure randomness
np.random.shuffle(sentiment_list)

# Assign the shuffled sentiments to the "Sentiment" column
data["Sentiment"] = sentiment_list

# Save or print the modified dataset
print(data.head())  # Display first few rows
# Optional: Save the modified dataset
data.to_csv("/content/Test_twitter_dataset_with_sentiments.csv", index=False)
```

```
   Tweet_ID        Username  \
0         1         julie81
1         2   richardhester
2         3  williamsjoseph
3         4     danielsmary
4         5      carlwarren

                                                Text  Retweets  Likes  \
0  Party least receive say or single. Prevent pre...         2     25
1  Hotel still Congress may member staff. Media d...        35     29
2  Nice be her debate industry that year. Film wh...        51     25
3  Laugh explain situation career occur serious. ...        37     18
4  Involve sense former often approach government...        27     80

             Timestamp Category        Date Day_of_Week  Hour Sentiment
0  2023-01-30 11:00:51   sports  2023-01-30      Monday    11  Negative
1  2023-01-02 22:45:58   sports  2023-01-02      Monday    22  Negative
2  2023-01-18 11:25:19   causal  2023-01-18   Wednesday    11  Positive
3  2023-04-10 22:06:29   sports  2023-04-10      Monday    22  Positive
4  2023-01-24 07:12:21   causal  2023-01-24     Tuesday     7  Positive
```

```
data
```

	Tweet_ID	Username	Text	Retweets	Likes	Timestamp	Category	Date	Day_of_Week	Hour	Sentiment
0	1	julie81	Party least receive say or single. Prevent pre...	2	25	2023-01-30 11:00:51	sports	2023-01-30	Monday	11	Negative
1	2	richardhester	Hotel still Congress may member staff. Media d...	35	29	2023-01-02 22:45:58	sports	2023-01-02	Monday	22	Negative
2	3	williamsjoseph	Nice be her debate industry that year. Film wh...	51	25	2023-01-18 11:25:19	causal	2023-01-18	Wednesday	11	Positive
3	4	danielsmary	Laugh explain situation career occur serious. ...	37	18	2023-04-10 22:06:29	sports	2023-04-10	Monday	22	Positive
4	5	carlwarren	Involve sense former often approach government...	27	80	2023-01-24 07:12:21	causal	2023-01-24	Tuesday	7	Positive
...	...	...	...	...	...	...	...	...	...	...	...
9995	9996	ntate	Agree reflect military box ability ever hold. ...	81	86	2023-01-15 11:46:20	sports	2023-01-15	Sunday	11	Positive

```
data['Category'].value_counts()
```

	count
Category	
causal	2517
Political	2189
Movies	1936
sports	1805
Companies	1553

```
data['Date'] = pd.to_datetime(data['Timestamp']).dt.date
data['Day_of_Week'] = pd.to_datetime(data['Timestamp']).dt.strftime('%A')
data['Hour'] = pd.to_datetime(data['Timestamp']).dt.hour
```

```
data
```

	Tweet_ID	Username	Text	Retweets	Likes	Timestamp	Category	Date	Day_of_Week	Hour
0	1	julie81	Party least receive say or single. Prevent pre...	2	25	2023-01-30 11:00:51	sports	2023-01-30	Monday	11
1	2	richardhester	Hotel still Congress may member staff. Media d...	35	29	2023-01-02 22:45:58	sports	2023-01-02	Monday	22
2	3	williamsjoseph	Nice be her debate industry that year. Film wh...	51	25	2023-01-18 11:25:19	causal	2023-01-18	Wednesday	11
3	4	danielsmary	Laugh explain situation career occur serious. ...	37	18	2023-04-10 22:06:29	sports	2023-04-10	Monday	22
4	5	carlwarren	Involve sense former often approach government...	27	80	2023-01-24 07:12:21	causal	2023-01-24	Tuesday	7
...	...	...	...	...	...	...	...	...	...	...
9995	9996	ntate	Agree reflect military box ability ever hold. ...	81	86	2023-01-15 11:46:20	sports	2023-01-15	Sunday	11
9996	9997	garrisonjoshua	Born which push still. Degree sometimes contro...	73	100	2023-05-06 00:46:54	Political	2023-05-06	Saturday	0

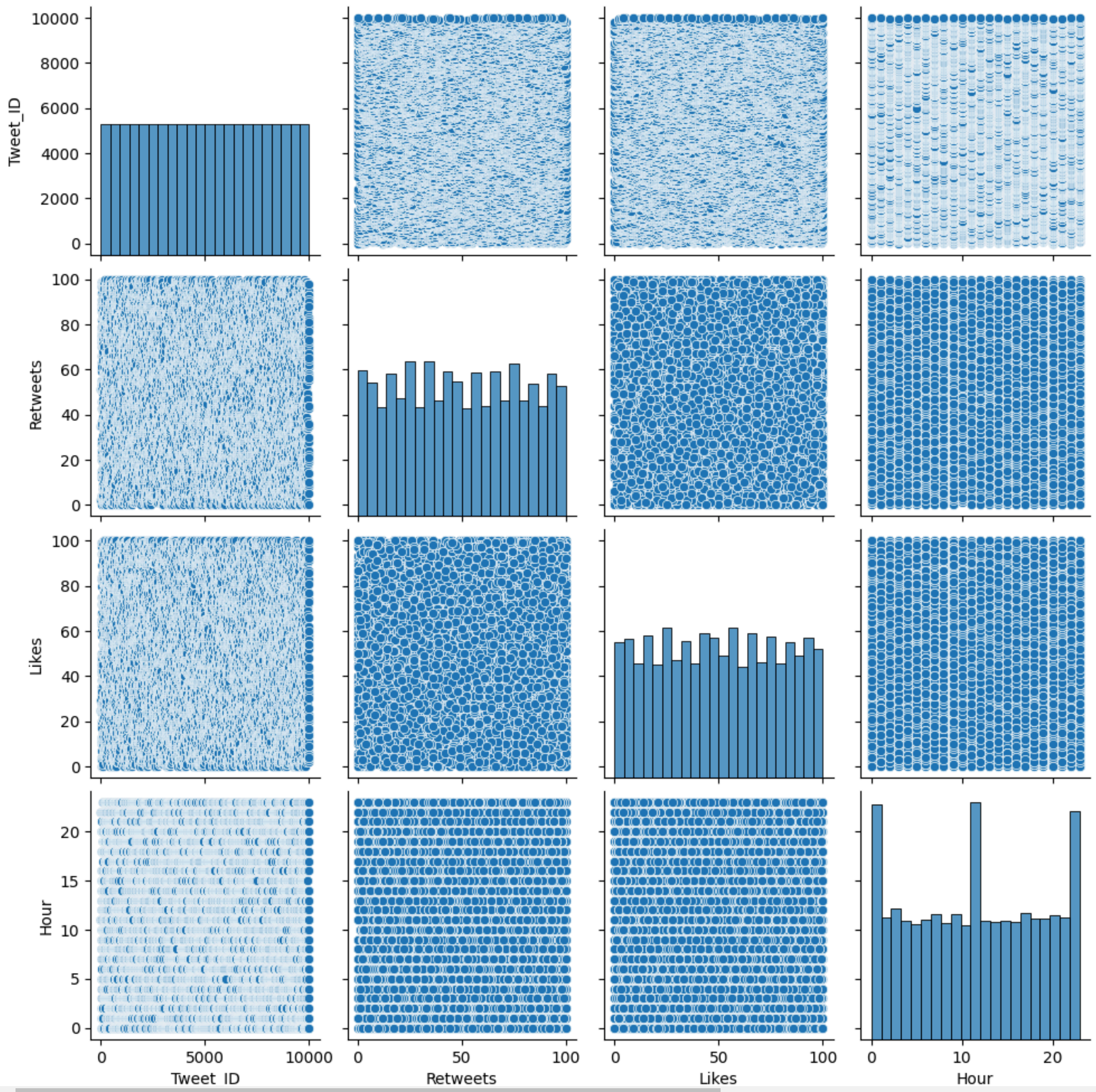
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import spacy
import math

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, BaggingClassifier, ExtraTreesClassifier, AdaBoostClassifier
from xgboost import XGBClassifier


from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, ConfusionMatrixDisplay
```

```
pip install xgboost
```

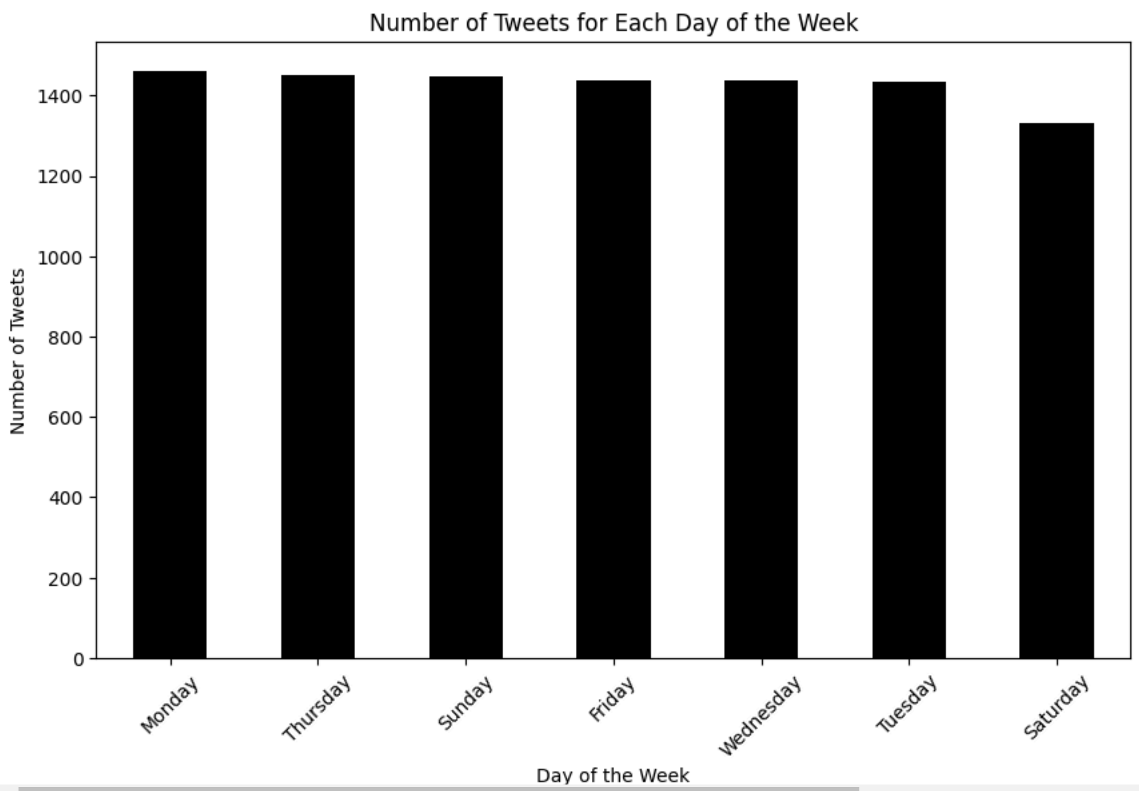
```
Collecting xgboost
  Downloading xgboost-2.1.3-py3-none-manylinux_2_28_x86_64.whl.metadata (2.1 kB)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from xgboost) (1.26.4)
Collecting nvidia-nccl-cu12 (from xgboost)
  Downloading nvidia_nccl_cu12-2.23.4-py3-none-manylinux2014_x86_64.whl.metadata (1.8 kB)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from xgboost) (1.13.1)
Downloading xgboost-2.1.3-py3-none-manylinux_2_28_x86_64.whl (153.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 153.9/153.9 MB 7.1 MB/s eta 0:00:00
Downloading nvidia_nccl_cu12-2.23.4-py3-none-manylinux2014_x86_64.whl (199.0 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 199.0/199.0 MB 5.3 MB/s eta 0:00:00
Installing collected packages: nvidia-nccl-cu12, xgboost
Successfully installed nvidia-nccl-cu12-2.23.4 xgboost-2.1.3
```

```
sns.pairplot(data, kind='scatter')
```

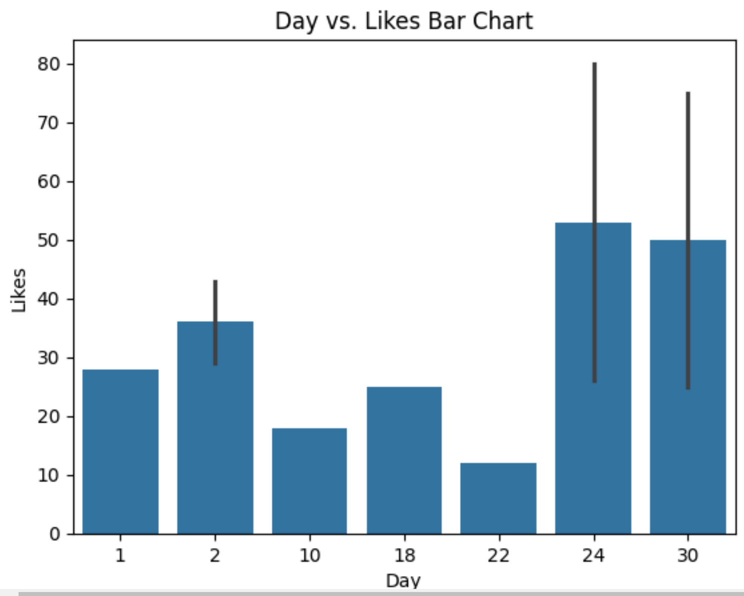
```
<seaborn.axisgrid.PairGrid at 0x7feafbfd93f0>
```



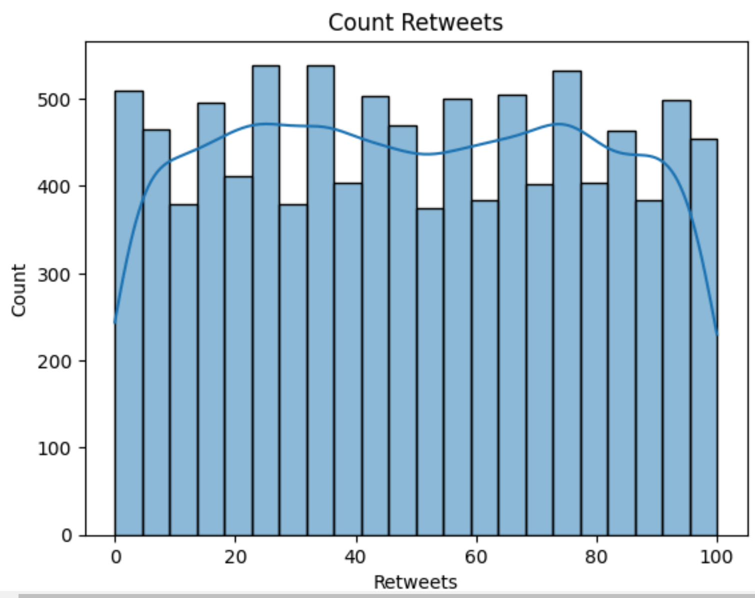
```
data['Month'] = pd.to_datetime(data['Timestamp']).dt.month
data['Day'] = pd.to_datetime(data['Timestamp']).dt.day


tweet_counts = data['Day_of_Week'].value_counts()

# Plot the number of tweets for each day of the week
plt.figure(figsize=(10,6))
tweet_counts.plot(kind='bar', color='black')
plt.title('Number of Tweets for Each Day of the Week')
plt.xlabel('Day of the Week')
plt.ylabel('Number of Tweets')
plt.xticks(rotation=45)
plt.show()
```



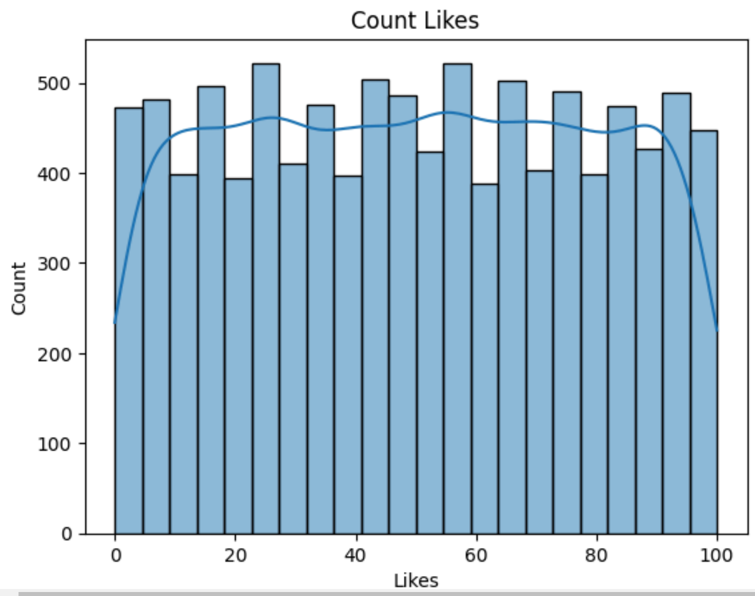
```
sns.barplot(data.head(10), x='Day', y='Likes')
plt.xlabel('Day')
plt.ylabel('Likes')
plt.title('Day vs. Likes Bar Chart')
plt.show()
```



```
sns.histplot(data['Retweets'], kde=True)
plt.xlabel('Retweets')
plt.ylabel('Count')
plt.title('Count Retweets')
plt.show()
```

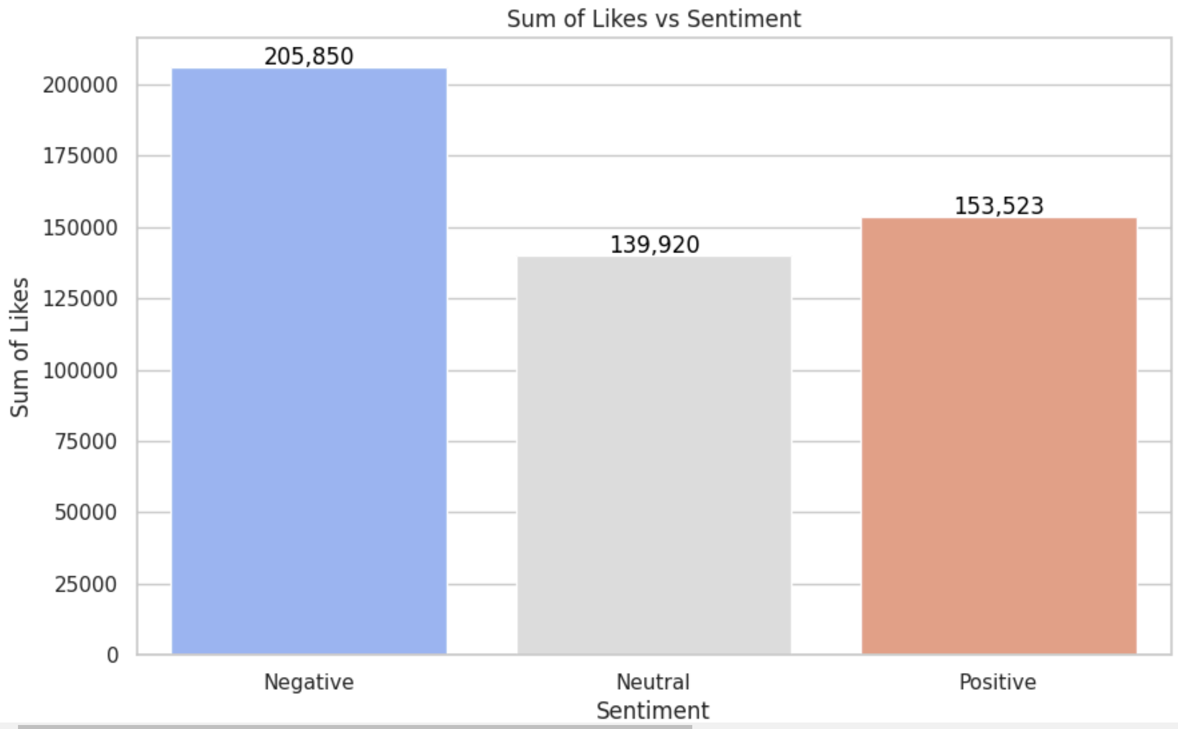


```
sns.histplot(data['Likes'], kde=True)
plt.xlabel('Likes')
plt.ylabel('Count')
plt.title('Count Likes')
plt.show()
```



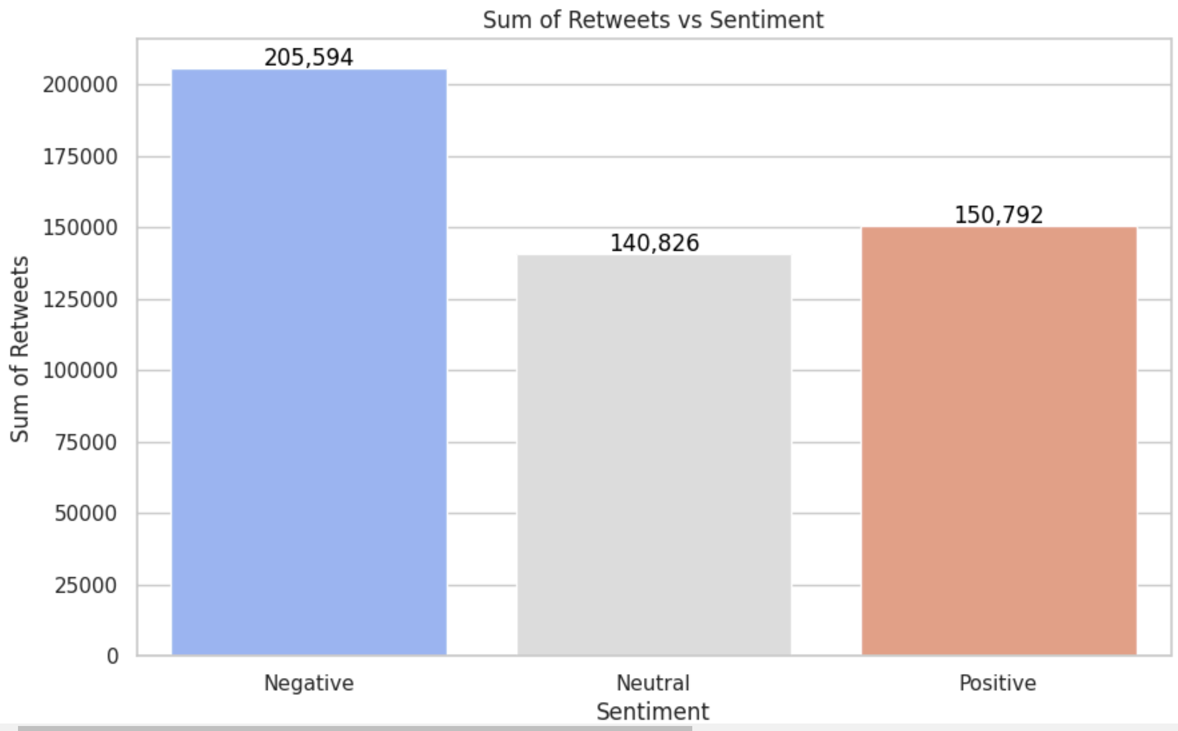
```
sentiment_summary = data.groupby("Sentiment")["Likes"].sum().reset_index()

# Set up the visual style
sns.set(style="whitegrid")

# Plotting the sum of Likes vs Sentiment using a bar plot
plt.figure(figsize=(10, 6))
ax = sns.barplot(x="Sentiment", y="Likes", data=sentiment_summary, palette="coolwarm")
plt.title("Sum of Likes vs Sentiment")
plt.xlabel("Sentiment")
plt.ylabel("Sum of Likes")

# Adding the values on the bars
for p in ax.patches:
    ax.annotate(f'{p.get_height():,.0f}',
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center',
                fontsize=12, color='black',
                xytext=(0, 5), textcoords='offset points')

plt.show()
```

```
<ipython-input-48-58ec458ea2c3>:8: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  ax = sns.barplot(x="Sentiment", y="Likes", data=sentiment_summary, palette="coolwarm")
```



```
sentiment_summary_retweets = data.groupby("Sentiment")["Retweets"].sum().reset_index()

# Plotting the sum of Retweets vs Sentiment using a bar plot
plt.figure(figsize=(10, 6))
ax = sns.barplot(x="Sentiment", y="Retweets", data=sentiment_summary_retweets, palette="coolwarm")
plt.title("Sum of Retweets vs Sentiment")
plt.xlabel("Sentiment")
plt.ylabel("Sum of Retweets")

# Adding the values on the bars
for p in ax.patches:
    ax.annotate(f'{p.get_height():,.0f}',
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center',
                fontsize=12, color='black',
                xytext=(0, 5), textcoords='offset points')

plt.show()
```

```
<ipython-input-49-d9aca34dd88b>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  ax = sns.barplot(x="Sentiment", y="Retweets", data=sentiment_summary_retweets, palette="coolwarm")
```



```
sentiment_summary = data.groupby("Sentiment")["Likes"].sum().reset_index()
sentiment_summary_retweets = data.groupby("Sentiment")["Retweets"].sum().reset_index()

# Display the table
print(sentiment_summary)
```

```
  Sentiment   Likes
0  Negative  205850
1   Neutral  139920
2  Positive  153523
```

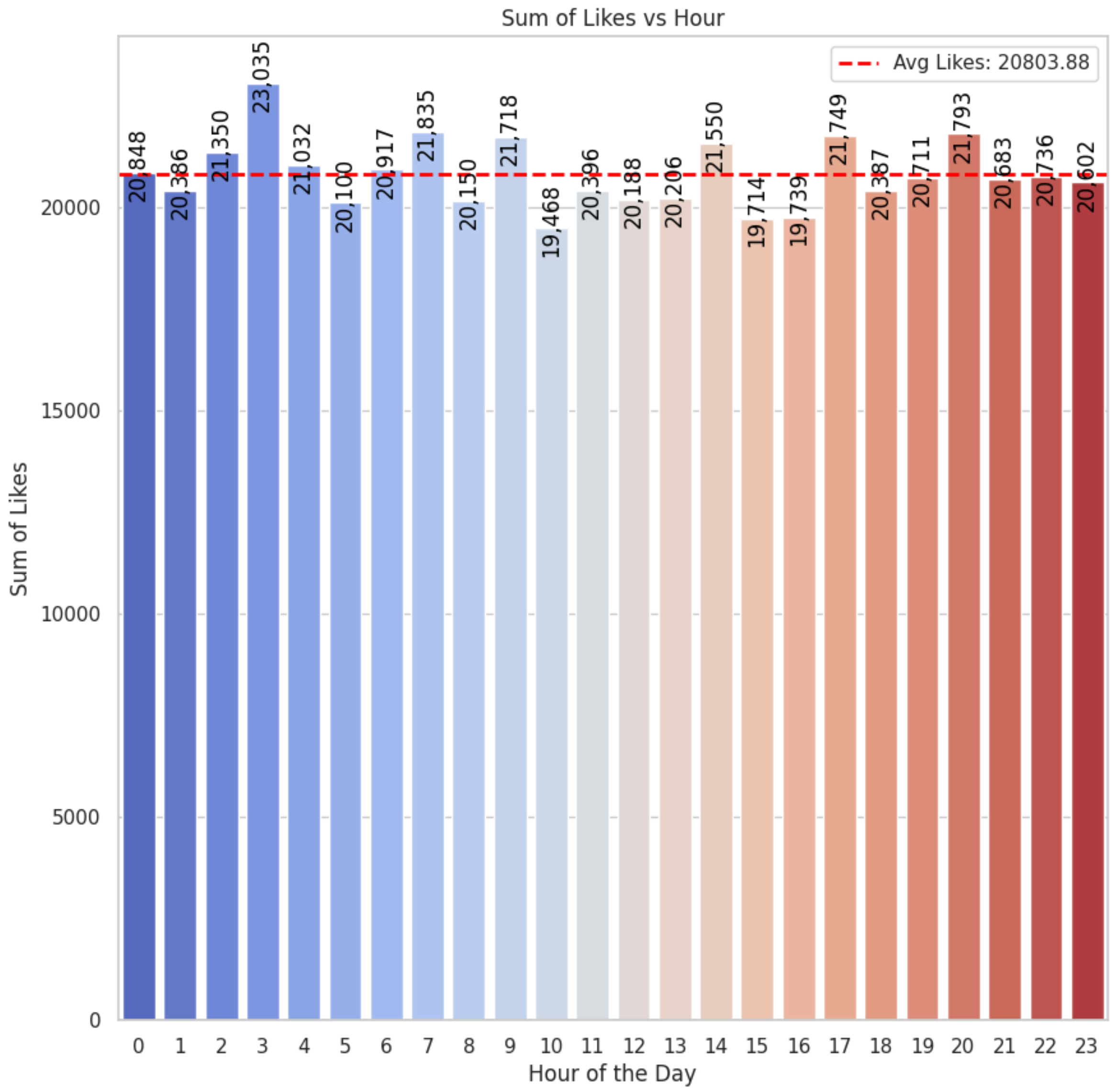
```
print(sentiment_summary_retweets)
```

```
  Sentiment  Retweets
0  Negative    205594
1   Neutral    140826
2  Positive    150792
```

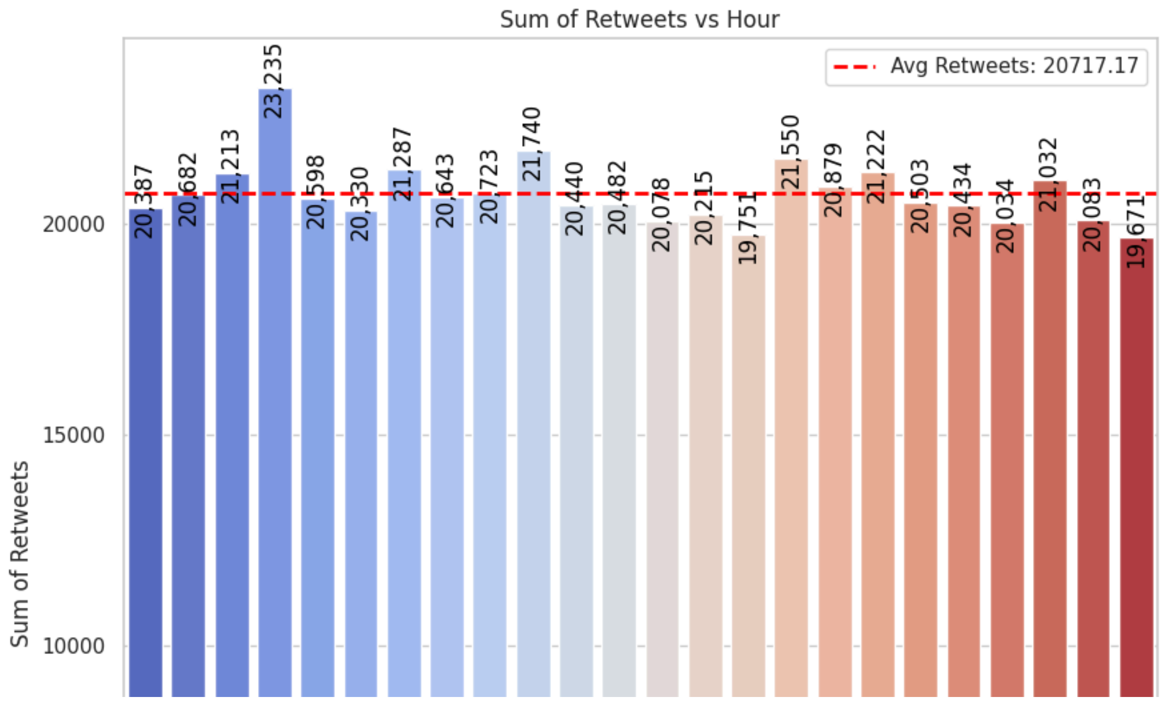
```
hourly_summary = data.groupby("Hour")[["Likes", "Retweets"]].sum().reset_index()

# Set up the visual style
sns.set(style="whitegrid")

# Plotting the sum of Likes vs Hour using a bar plot
plt.figure(figsize=(10, 10))
ax = sns.barplot(x="Hour", y="Likes", data=hourly_summary, palette="coolwarm")
plt.title("Sum of Likes vs Hour")
plt.xlabel("Hour of the Day")
plt.ylabel("Sum of Likes")

# Adding the values on the bars vertically
for p in ax.patches:
    ax.annotate(f'{p.get_height():,.0f}',
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center',
                fontsize=12, color='black',
                xytext=(0, 5), textcoords='offset points', rotation=90)

# Adding the average dotted line for Likes
average_likes = hourly_summary["Likes"].mean()
ax.axhline(average_likes, color='red', linestyle='--', linewidth=2, label=f'Avg Likes: {average_likes:.2f}')

# Adding the legend for the dotted line
ax.legend()

plt.show()
```

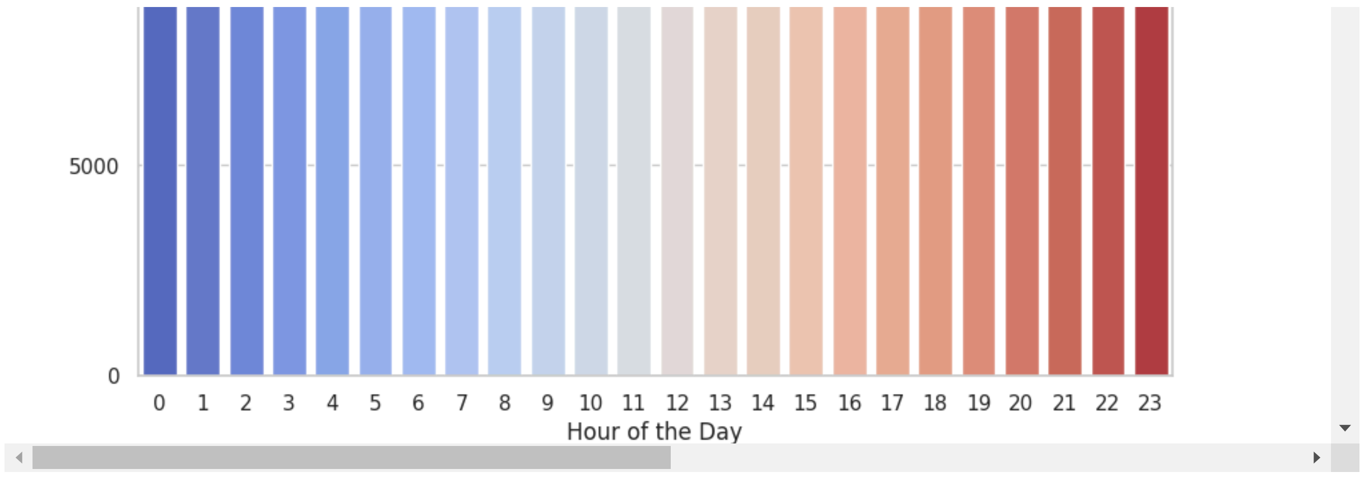
```
# Plotting the sum of Retweets vs Hour using a bar plot
plt.figure(figsize=(10, 10))
ax = sns.barplot(x="Hour", y="Retweets", data=hourly_summary, palette="coolwarm")
plt.title("Sum of Retweets vs Hour")
plt.xlabel("Hour of the Day")
plt.ylabel("Sum of Retweets")

# Adding the values on the bars vertically
for p in ax.patches:
    ax.annotate(f'{p.get_height():,.0f}',
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center',
                fontsize=12, color='black',
                xytext=(0, 5), textcoords='offset points', rotation=90)

# Adding the average dotted line for Retweets
average_retweets = hourly_summary["Retweets"].mean()
ax.axhline(average_retweets, color='red', linestyle='--', linewidth=2, label=f'Avg Retweets: {average_retweets:.2f}')

# Adding the legend for the dotted line
ax.legend()

plt.show()
```

```
<ipython-input-55-9f96763f1045>:8: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set

  ax = sns.barplot(x="Hour", y="Likes", data=hourly_summary, palette="coolwarm")
```



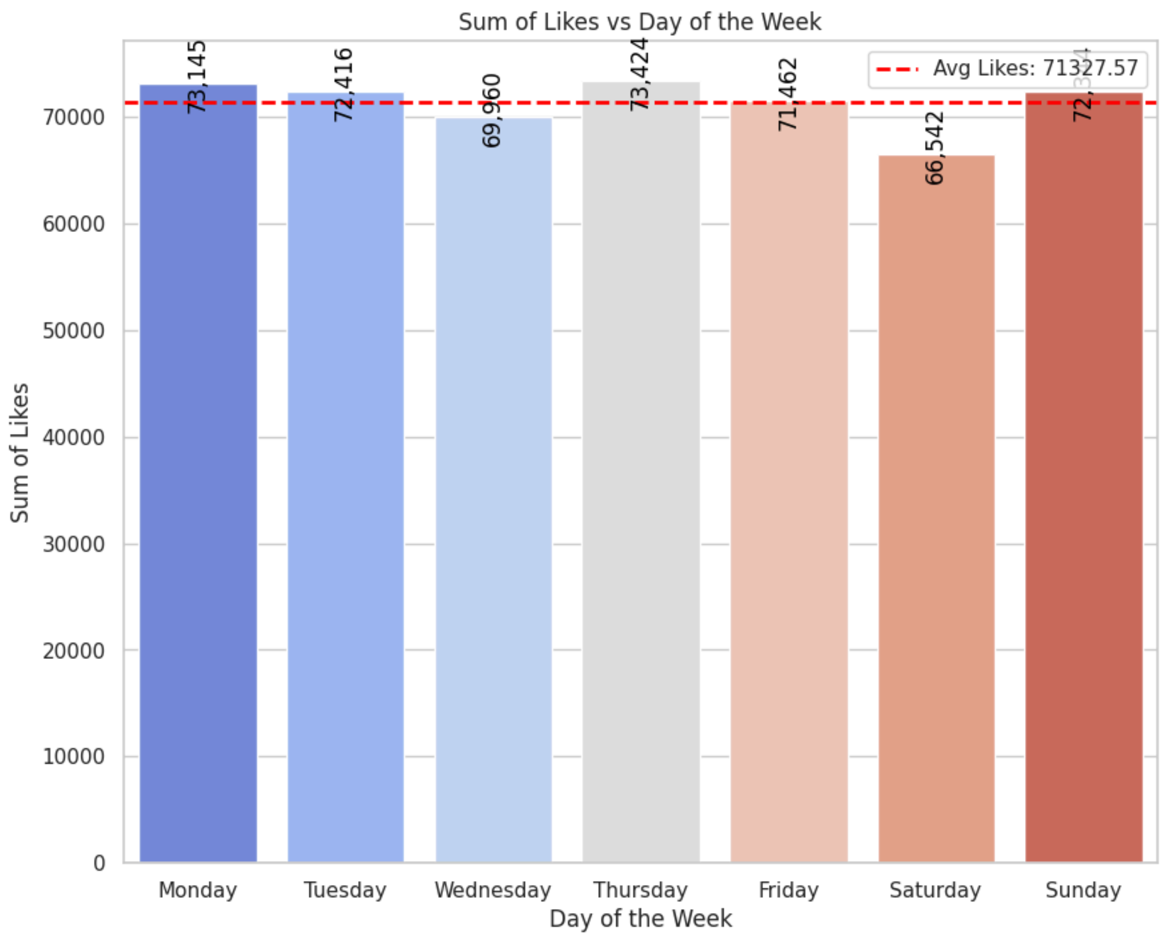
```
<ipython-input-55-9f96763f1045>:32: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set

  ax = sns.barplot(x="Hour", y="Retweets", data=hourly_summary, palette="coolwarm")
```



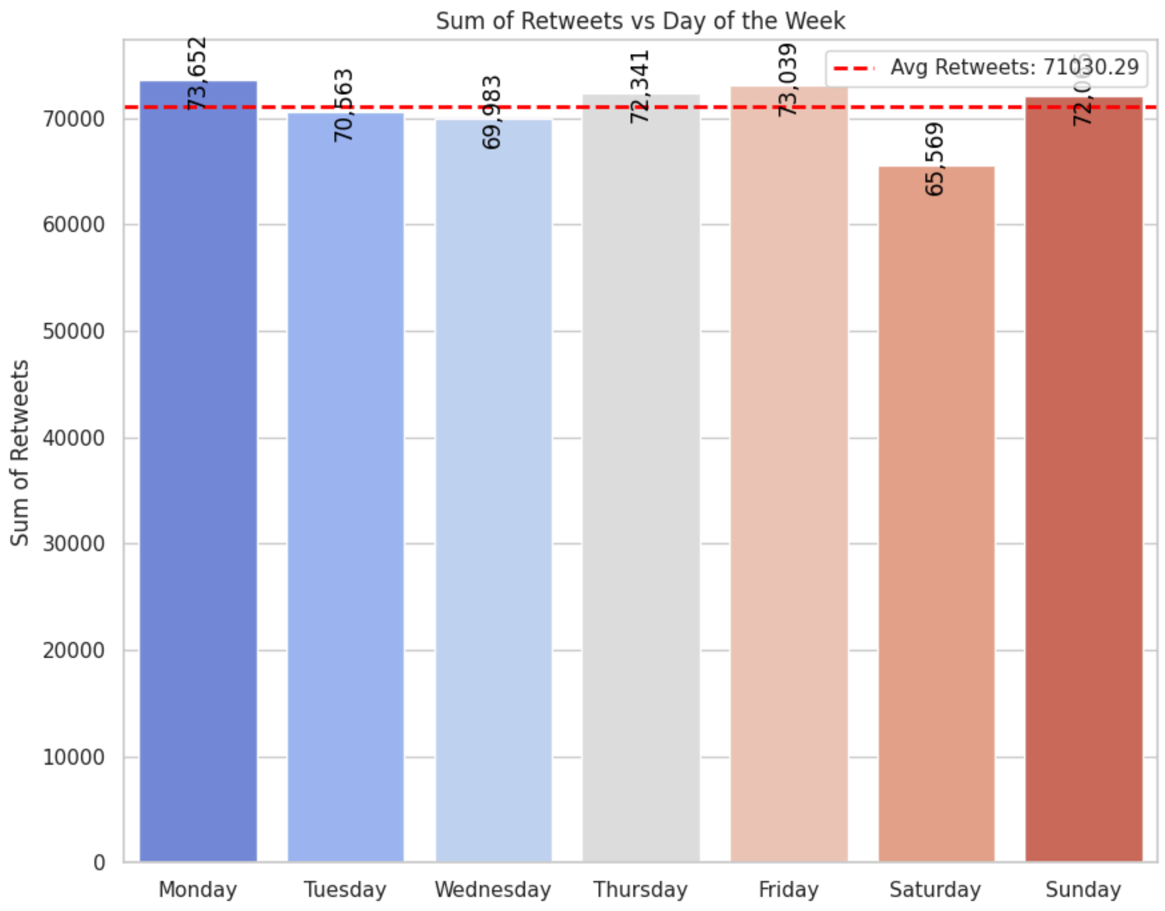


```
day_summary = data.groupby("Day_of_Week")[["Likes", "Retweets"]].sum().reset_index()

# Reorder days to ensure proper order (Monday, Tuesday, ..., Sunday)
ordered_days = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"]
day_summary['Day_of_Week'] = pd.Categorical(day_summary['Day_of_Week'], categories=ordered_days, ordered=True)
day_summary = day_summary.sort_values('Day_of_Week')

# Set up the visual style
sns.set(style="whitegrid")

# Plotting the sum of Likes vs Day of the Week using a bar plot
plt.figure(figsize=(10, 8))
ax = sns.barplot(x="Day_of_Week", y="Likes", data=day_summary, palette="coolwarm")
plt.title("Sum of Likes vs Day of the Week")
plt.xlabel("Day of the Week")
plt.ylabel("Sum of Likes")

# Adding the values on the bars vertically
for p in ax.patches:
    ax.annotate(f'{p.get_height():,.0f}',
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center',
                fontsize=12, color='black',
                xytext=(0, 5), textcoords='offset points', rotation=90)

# Adding the average dotted line for Likes
average_likes = day_summary["Likes"].mean()
ax.axhline(average_likes, color='red', linestyle='--', linewidth=2, label=f'Avg Likes: {average_likes:.2f}')

# Adding the legend for the dotted line
ax.legend()

plt.show()

# Plotting the sum of Retweets vs Day of the Week using a bar plot
plt.figure(figsize=(10, 8))
ax = sns.barplot(x="Day_of_Week", y="Retweets", data=day_summary, palette="coolwarm")
plt.title("Sum of Retweets vs Day of the Week")
plt.xlabel("Day of the Week")
plt.ylabel("Sum of Retweets")

# Adding the values on the bars vertically
for p in ax.patches:
    ax.annotate(f'{p.get_height():,.0f}',
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center',
                fontsize=12, color='black',
                xytext=(0, 5), textcoords='offset points', rotation=90)

# Adding the average dotted line for Retweets
average_retweets = day_summary["Retweets"].mean()
ax.axhline(average_retweets, color='red', linestyle='--', linewidth=2, label=f'Avg Retweets: {average_retweets:.2f}')

# Adding the legend for the dotted line
ax.legend()

plt.show()
```

```
<ipython-input-57-3fd62ac91203>:13: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set

  ax = sns.barplot(x="Day_of_Week", y="Likes", data=day_summary, palette="coolwarm")
```



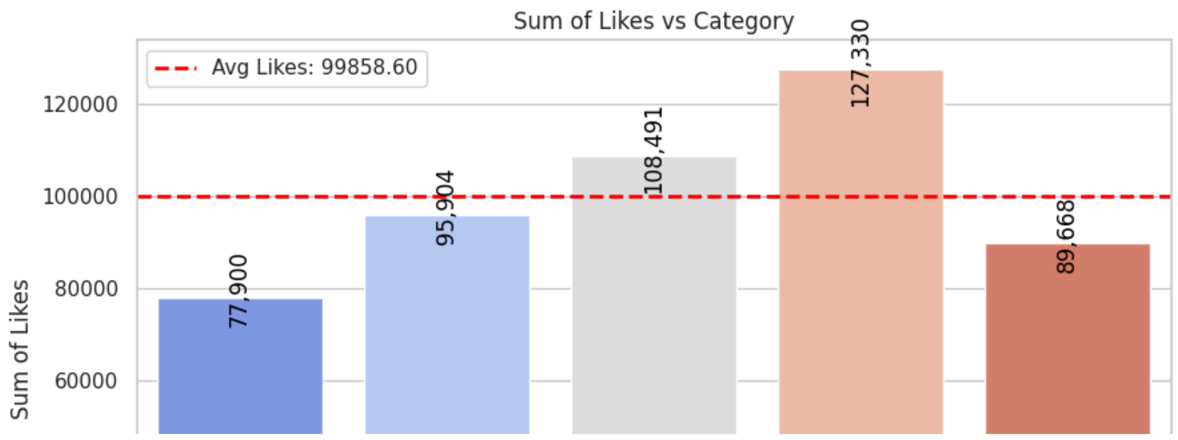
```
<ipython-input-57-3fd62ac91203>:37: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set

  ax = sns.barplot(x="Day_of_Week", y="Retweets", data=day_summary, palette="coolwarm")
```



Day of the Week

```
category_summary = data.groupby("Category")[["Likes", "Retweets"]].sum().reset_index()

# Set up the visual style
sns.set(style="whitegrid")

# Plotting the sum of Likes vs Category using a bar plot
plt.figure(figsize=(10, 6))
ax = sns.barplot(x="Category", y="Likes", data=category_summary, palette="coolwarm")
plt.title("Sum of Likes vs Category")
plt.xlabel("Category")
plt.ylabel("Sum of Likes")

# Adding the values on the bars vertically
for p in ax.patches:
    ax.annotate(f'{p.get_height():,.0f}',
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center',
                fontsize=12, color='black',
                xytext=(0, 5), textcoords='offset points', rotation=90)

# Adding the average dotted line for Likes
average_likes = category_summary["Likes"].mean()
ax.axhline(average_likes, color='red', linestyle='--', linewidth=2, label=f'Avg Likes: {average_likes:.2f}')

# Adding the legend for the dotted line
ax.legend()

plt.show()

# Plotting the sum of Retweets vs Category using a bar plot
plt.figure(figsize=(10, 6))
ax = sns.barplot(x="Category", y="Retweets", data=category_summary, palette="coolwarm")
plt.title("Sum of Retweets vs Category")
plt.xlabel("Category")
plt.ylabel("Sum of Retweets")

# Adding the values on the bars vertically
for p in ax.patches:
    ax.annotate(f'{p.get_height():,.0f}',
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center',
                fontsize=12, color='black',
                xytext=(0, 5), textcoords='offset points', rotation=90)

# Adding the average dotted line for Retweets
average_retweets = category_summary["Retweets"].mean()
ax.axhline(average_retweets, color='red', linestyle='--', linewidth=2, label=f'Avg Retweets: {average_retweets:.2f}')

# Adding the legend for the dotted line
ax.legend()

plt.show()
```

```
<ipython-input-58-e45d537907cf>:8: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set

  ax = sns.barplot(x="Category", y="Likes", data=category_summary, palette="coolwarm")
```


Sum of Likes vs Category
Avg Likes: 99858.60
Sum of Likes