**#Python script for sentiment analysis using TextBlob**

from textblob import TextBlob

#'df' is the DataFrame and 'TEXT' is the column name

text\_column = df['clean\_tokens2']

sentiment\_scores = []

sentiment\_labels = []

for text in text\_column:

blob = TextBlob(text)

polarity = blob.sentiment.polarity

sentiment\_scores.append(polarity)

if polarity > 0:

sentiment\_labels.append('Positive')

elif polarity < 0:

sentiment\_labels.append('Negative')

else:

sentiment\_labels.append('Neutral')

df['sentiment\_score\_tb'] = sentiment\_scores

df['sentiment\_labels\_tb'] = sentiment\_labels

pred\_labels\_tb = [label.capitalize() for label in df["sentiment\_labels\_tb"].tolist()]

accuracy = accuracy\_score(target\_labels, pred\_labels\_tb)

print(f"Accuracy: {accuracy}")

f1 = f1\_score(target\_labels, pred\_labels\_tb, average='macro')

print(f"F1-Score: {f1}")

precision = precision\_score(target\_labels, pred\_labels\_tb, average='macro')

print(f"Precision: {precision}")

recall = recall\_score(target\_labels, pred\_labels\_tb, average='macro')

print(f"Recall: {recall}")

confusion = confusion\_matrix(target\_labels, pred\_labels\_tb)

print("Confusion Matrix:")

print(confusion)

sentiment\_counts = df['sentiment\_labels\_tb'].value\_counts()

print(sentiment\_counts)

import seaborn as sns

# Set the font family and size

font = {'family': 'serif', 'weight': 'normal', 'size': 14}

sns.set(font\_scale=1.2) # Increase font size for Seaborn plots

# Create the histogram using Seaborn

sns.histplot(data=df, x='sentiment\_score\_tb', bins=30, color='orange', edgecolor='black', kde = False)

# Set labels and title

plt.xlabel('Sentiment scores')

plt.ylabel('Frequency')

sns.despine()

plt.show()

#pie plot

# Set the font family and size

font = {'family': 'serif', 'weight': 'normal', 'size': 14}

plt.rc('font', \*\*font)

labels = ["Positive", "Negative", "Neutral"]

sizes = [597, 406, 516]

colors = ['lightgreen', 'red', 'yellow']

plt.pie(sizes, colors = colors , labels = labels, autopct = '%1.1f%%', startangle = 90)

plt.axis('equal')

plt.show()

# Plotting the kernel density plot

# Set the font family and size

font = {'family': 'serif', 'weight': 'normal', 'size': 14}

plt.rc('font', \*\*font)

sns.kdeplot(sentiment\_scores, fill=True)

plt.xlabel('Sentiment Score')

plt.ylabel('Density')

plt.gca().spines['top'].set\_visible(False)

plt.gca().spines['right'].set\_visible(False)

plt.show()

#Subjectivity test

text\_column = df['clean\_tokens2']

subjectivity\_scores = []

subjectivity\_labels = []

for text in text\_column:

blob = TextBlob(text)

subjectivity = blob.subjectivity

subjectivity\_scores.append(subjectivity)

if subjectivity >= 0.5:

subjectivity\_labels.append('Subjective')

elif subjectivity <= 0.5:

subjectivity\_labels.append('Objective')

else:

subjectivity\_labels.append('Neutral')

df['subjectivity\_score\_tb'] = subjectivity\_scores

df['subjectivity\_labels\_tb'] = subjectivity\_labels

# Get subjectivity scores

subjectivity\_scores = df['subjectivity\_score\_tb']

# Set the font family and size

font = {'family': 'serif', 'weight': 'normal', 'size': 14}

plt.rc('font', \*\*font)

# Calculate histogram data

hist, edges = np.histogram(subjectivity\_scores, bins=10)

# Calculate bin centers

bin\_centers = (edges[:-1] + edges[1:]) / 2

# Plotting the histogram

plt.bar(bin\_centers, bin\_counts, width=bin\_width, edgecolor='black', color='teal', alpha=0.7, label='Histogram')

plt.xlabel('Subjectivity Score')

plt.ylabel('Frequency')

plt.gca().spines['top'].set\_visible(False)

plt.gca().spines['right'].set\_visible(False)

plt.show()

# Plotting the sentiment distribution

# Set the font family and size

font = {'family': 'serif', 'weight': 'normal', 'size': 14}

plt.rc('font', \*\*font)

subjectivity\_counts = df['subjectivity\_labels\_tb'].value\_counts()

ax = subjectivity\_counts.plot(kind='bar', color='teal')

plt.xlabel('Semantic orientation label')

plt.ylabel('Frequency')

plt.xticks(rotation='horizontal')

plt.gca().spines['top'].set\_visible(False)

plt.gca().spines['right'].set\_visible(False)

plt.show()

# Plotting the scatter plot

# Set the font family and size

font = {'family': 'serif', 'weight': 'normal', 'size': 14}

plt.rc('font', \*\*font)

# Scatter plot

plt.scatter(subjectivity\_scores, sentiment\_scores, color='darkblue')

# Fit a linear regression line to the data

slope, intercept, r\_value, p\_value, std\_err = linregress(subjectivity\_scores, sentiment\_scores)

trendline\_x = np.array([min(subjectivity\_scores), max(subjectivity\_scores)])

trendline\_y = intercept + slope \* trendline\_x

# Plot the trendline

plt.plot(trendline\_x, trendline\_y, color='red', linestyle='dashed')

# Customize the plot

plt.ylabel('Sentiment Scores')

plt.xlabel('Subjectivity Scores')

#plt.title('The relationship between the sentiment and subjectivity scores')

plt.gca().spines['top'].set\_visible(False)

plt.gca().spines['right'].set\_visible(False)

# Add the trendline equation to the plot

plt.text(0.05, 0.9, f'y = {slope:.2f}x +( {intercept:.2f})', fontsize = 8, color='red', transform=plt.gca().transAxes)

# Calculate the correlation coefficient

correlation\_coefficient = pearsonr(subjectivity\_scores, sentiment\_scores)[0]

# Add the correlation coefficient to the plot

plt.text(0.05, 0.8, f'Correlation coefficient: {correlation\_coefficient:.2f}', color='red',fontsize = 8, transform=plt.gca().transAxes)

plt.show()

#Lowess smoothing

%pip install statsmodels

import statsmodels.api as sm

plt.scatter(subjectivity\_scores, sentiment\_scores, color='darkblue')

lowess = sm.nonparametric.lowess(sentiment\_scores, subjectivity\_scores)

lowess\_x, lowess\_y = lowess.T

# Plot Lowess curve

plt.plot(lowess\_x, lowess\_y, color='red', label='Lowess Smoothing')

plt.ylabel('Sentiment Scores')

plt.xlabel('Subjectivity Scores')

plt.gca().spines['top'].set\_visible(False)

plt.gca().spines['right'].set\_visible(False)

plt.show()

# Plotting the kernel density plot

# Set the font family and size

font = {'family': 'serif', 'weight': 'normal', 'size': 14}

plt.rc('font', \*\*font)

sns.kdeplot(subjectivity\_scores, fill=True)

plt.xlabel('Subjectivity Score')

plt.ylabel('Density')

plt.gca().spines['top'].set\_visible(False)

plt.gca().spines['right'].set\_visible(False)

plt.show()

# Plotting the box plot

# Set the font family and size

font = {'family': 'serif', 'weight': 'normal', 'size': 14}

plt.rc('font', \*\*font)

plt.boxplot(subjectivity\_scores, vert = False)

plt.xlabel('Subjectivity Score')

plt.gca().set\_yticklabels([])

plt.gca().set\_yticks([])

plt.gca().spines['top'].set\_visible(False)

plt.gca().spines['right'].set\_visible(False)

plt.gca().spines['left'].set\_visible(False)

plt.show()

# Plotting the box plot

# Set the font family and size

font = {'family': 'serif', 'weight': 'normal', 'size': 14}

plt.rc('font', \*\*font)

plt.boxplot(df['sentiment\_score\_tb'], vert = False)

plt.xlabel('Sentiment Score')

plt.gca().set\_yticklabels([])

plt.gca().set\_yticks([])

plt.gca().spines['top'].set\_visible(False)

plt.gca().spines['right'].set\_visible(False)

plt.gca().spines['left'].set\_visible(False)

plt.show()

**Python script for sentiment analysis using VADER**

from nltk.sentiment import SentimentIntensityAnalyzer

import seaborn as sns

nltk.download('vader\_lexicon')

text\_column = df['clean\_tokens2']

senti = SentimentIntensityAnalyzer()

sentiment\_scores\_nltk = []

sentiment\_labels\_nltk = []

for text in text\_column:

sentiment\_score\_nltk = senti.polarity\_scores(text)

sentiment\_scores\_nltk.append(sentiment\_score\_nltk)

if sentiment\_score\_nltk['compound'] > 0:

sentiment\_labels\_nltk.append('Positive')

elif sentiment\_score\_nltk['compound'] < 0:

sentiment\_labels\_nltk.append('Negative')

else:

sentiment\_labels\_nltk.append('Neutral')

# Add the sentiment scores to the DataFrame

df['sentiment\_score\_nltk'] = sentiment\_scores\_nltk

df['sentiment\_labels\_nltk'] = sentiment\_labels\_nltk

# Plotting the sentiment distribution

# Count the occurrences of each sentiment label

sentiment\_label\_counts = df['sentiment\_labels\_nltk'].value\_counts()

colors = {'Positive': 'lightgreen', 'Negative': 'red', 'Neutral': 'yellow'}

# Plotting the pie chart with custom colors

font = {'family': 'serif', 'weight': 'normal', 'size': 14}

plt.rc('font', \*\*font)

plt.figure(figsize=(8, 6))

plt.pie(sentiment\_label\_counts, labels=sentiment\_label\_counts.index, autopct='%1.1f%%', startangle=140, colors=[colors[label] for label in sentiment\_label\_counts.index])

plt.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle

plt.show()

# Set the font family and size

font = {'family': 'serif', 'weight': 'normal', 'size': 14}

plt.rc('font', \*\*font)

# Extract compound scores from the dictionaries in 'sentiment\_score\_tb'

df['compound\_scores'] = df['sentiment\_score\_nltk'].apply(lambda x: x['compound'])

# Plot histogram of sentiment scores using Seaborn

plt.figure(figsize=(8, 6))

sns.histplot(data=df, x='compound\_scores', bins=20, color='orange', edgecolor='black', alpha=0.7)

plt.xlabel('Sentiment Scores')

plt.xticks(rotation=45)

# Add custom interval labels to the x-axis

interval\_labels = [-1.0, -0.9, -0.8, -0.7, -0.6, -0.5, -0.4, -0.3, -0.2, -0.1, 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0] # Specify the desired interval labels

plt.xticks(interval\_labels)

plt.ylabel('Frequency')

plt.grid(True)

plt.show()

# Plotting the kernel density plot

sns.kdeplot(sentiment\_score\_nltk, fill=True)

interval\_labels = [-1.8, -1.6, -1.4, -1.2,-1.0, -0.8, -0.6, -0.4, -0.2, 0.0, 0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8] # Specify the desired interval labels

plt.xticks(interval\_labels, rotation = 45)

plt.xlabel('Sentiment score')

plt.ylabel('Density')

plt.gca().spines['top'].set\_visible(False)

plt.gca().spines['right'].set\_visible(False)

plt.show()