**SMA – 1:**

|  |
| --- |
| import pandas as pd |
|  |  |
|  | # !pip install pyLDAvis |
|  | import pyLDAvis.gensim\_models |
|  |  |
|  | from gensim import corpora |
|  | from gensim.models import LdaModel |
|  | from gensim.utils import simple\_preprocess |
|  | from gensim.parsing.preprocessing import STOPWORDS |
|  |  |
|  | df = pd.read\_csv('social\_media\_data.csv') |
|  |  |
|  | stop\_words = set(STOPWORDS) |
|  | def preprocess(text): |
|  | result = [] |
|  | for token in simple\_preprocess(text): |
|  | if token not in stop\_words: |
|  | result.append(token) |
|  |  |
|  | return result |
|  |  |
|  |  |
|  | texts = df['text'].apply(preprocess) |
|  | dictionary = corpora.Dictionary(texts) |
|  | corpus = [dictionary.doc2bow(text) for text in texts] |
|  |  |
|  | num\_topics = 5 |
|  | passes = 10 |
|  | lda\_model = LdaModel(corpus=corpus, id2word=dictionary, num\_topics=num\_topics, passes=passes) |
|  |  |
|  | for topic in lda\_model.print\_topics(): |
|  | print(topic) |
|  |  |
|  | vis = pyLDAvis.gensim\_models.prepare(lda\_model, corpus, dictionary) |
|  | pyLDAvis.display(vis) |

**SMA – 2: LOCATION ANALYTICS**

|  |
| --- |
|  |
|  | import numpy as np |
|  | import matplotlib.pyplot as plt |
|  | import seaborn as sns |
|  |  |
|  | # Load the tweet data from a CSV file |
|  | data = pd.read\_csv("tweet\_data.csv") |
|  |  |
|  | # Remove any rows with missing values |
|  | data = data.dropna() |
|  |  |
|  | # Create a new column for the state abbreviation based on the location data |
|  | states = { |
|  | 'AL': 'Alabama', 'AK': 'Alaska', 'AZ': 'Arizona', 'AR': 'Arkansas', 'CA': 'California', |
|  | 'CO': 'Colorado', 'CT': 'Connecticut', 'DE': 'Delaware', 'FL': 'Florida', 'GA': 'Georgia', |
|  | 'HI': 'Hawaii', 'ID': 'Idaho', 'IL': 'Illinois', 'IN': 'Indiana', 'IA': 'Iowa', |
|  | 'KS': 'Kansas', 'KY': 'Kentucky', 'LA': 'Louisiana', 'ME': 'Maine', 'MD': 'Maryland', |
|  | 'MA': 'Massachusetts', 'MI': 'Michigan', 'MN': 'Minnesota', 'MS': 'Mississippi', 'MO': 'Missouri', |
|  | 'MT': 'Montana', 'NE': 'Nebraska', 'NV': 'Nevada', 'NH': 'New Hampshire', 'NJ': 'New Jersey', |
|  | 'NM': 'New Mexico', 'NY': 'New York', 'NC': 'North Carolina', 'ND': 'North Dakota', 'OH': 'Ohio', |
|  | 'OK': 'Oklahoma', 'OR': 'Oregon', 'PA': 'Pennsylvania', 'RI': 'Rhode Island', 'SC': 'South Carolina', |
|  | 'SD': 'South Dakota', 'TN': 'Tennessee', 'TX': 'Texas', 'UT': 'Utah', 'VT': 'Vermont', |
|  | 'VA': 'Virginia', 'WA': 'Washington', 'WV': 'West Virginia', 'WI': 'Wisconsin', 'WY': 'Wyoming' |
|  | } |
|  |  |
|  |  |
|  | def get\_state(location): |
|  | words = location.split() |
|  | for word in words: |
|  | if word.upper() in states: |
|  | return states[word.upper()] |
|  | return None |
|  |  |
|  |  |
|  | data['state'] = data['location'].apply(get\_state) |
|  | print(data.head()) |
|  |  |
|  | # Plot the frequency of each state in the data |
|  | state\_counts = data['state'].value\_counts() |
|  | # print("Count", state\_counts) |
|  | plt.figure(figsize=(12, 6)) |
|  |  |
|  | # name, values |
|  | sns.barplot(x=state\_counts.index, y=state\_counts.values, palette="rocket") |
|  | plt.xticks(rotation=90) |
|  | plt.xlabel('State') |
|  | plt.ylabel('Frequency') |
|  | plt.title('Frequency of States in Tweet Data') |
|  | plt.show() |

**SMA – 3 : TRENDS ANALYSIS**

|  |
| --- |
|  |
|  | import matplotlib.pyplot as plt |
|  |  |
|  | # Read in the CSV file |
|  | df = pd.read\_csv('trend\_analysis.csv') |
|  |  |
|  | # Convert the 'date' column to a datetime data type |
|  | df['date'] = pd.to\_datetime(df['date']) |
|  |  |
|  | print(df.head()) |
|  |  |
|  | # Set the 'date' column as the index of the DataFrame |
|  | df.set\_index('date', inplace=True) |
|  |  |
|  | # Resample the data by day and count the number of entries in each day |
|  | # argument 'D' indicating that we want to resample by day. |
|  | daily\_counts = df.resample('D').count() |
|  | print("Daily Counts: \n", daily\_counts) |
|  |  |
|  |  |
|  | # Plot the daily counts over time |
|  | plt.plot(daily\_counts.index, daily\_counts['id']) |
|  | plt.xlabel('Day') |
|  | plt.ylabel('Number of Entries') |
|  | plt.title('Social Media Trends') |
|  | plt.show() |

**SMA-4 : HASHTAGS ANALYSIS**

|  |
| --- |
|  |
|  | import matplotlib.pyplot as plt |
|  |  |
|  | # Read in the CSV file |
|  | df = pd.read\_csv('hashtag\_analysis.csv') |
|  |  |
|  | # Group the data by hashtags and user groups, and count the occurrences |
|  | hashtags\_by\_group = df.groupby( |
|  | ['user\_group', 'hashtags']).size().reset\_index(name='count') |
|  |  |
|  | # Plot a horizontal bar chart for each user group |
|  | for group in df['user\_group'].unique(): |
|  | group\_data = hashtags\_by\_group[hashtags\_by\_group['user\_group'] == group].sort\_values('count', ascending=False).head(10) |
|  | ax = group\_data.plot(kind='barh', x='hashtags', y='count', |
|  | legend=False, title=f'Top Hashtags for {group}') |
|  | ax.set\_xlabel('Frequency') |
|  | plt.tight\_layout() |
|  | plt.show() |

**SMA 5: SENTIMENT ANALYSIS**

|  |
| --- |
|  |
|  | from textblob import TextBlob |
|  | import seaborn as sns |
|  | import matplotlib.pyplot as plt |
|  |  |
|  | # Read the dataset into a pandas DataFrame |
|  | df = pd.read\_csv("sentiment.csv") |
|  |  |
|  | # Perform sentiment analysis using TextBlob and store the polarity score |
|  | df["polarity"] = df["text"].apply(lambda x: TextBlob(x).sentiment.polarity) |
|  |  |
|  | # Create separate DataFrames for positive and negative tweets |
|  | pos\_tweets = df[df["label"] == "positive"] |
|  | neg\_tweets = df[df["label"] == "negative"] |
|  |  |
|  | # Plot the distribution of polarity scores for positive and negative tweets |
|  | sns.kdeplot(pos\_tweets["polarity"], shade=True, label="Positive") |
|  | sns.kdeplot(neg\_tweets["polarity"], shade=True, label="Negative") |
|  | plt.xlabel("Polarity Score") |
|  | plt.ylabel("Density") |
|  | plt.title("Sentiment Analysis of Tweets") |
|  | plt.legend() |
|  | plt.show() |

**SMA-6: USER ENGAGEMENT**

|  |
| --- |
|  |
|  | import seaborn as sns |
|  | import matplotlib.pyplot as plt |
|  | import nltk |
|  | from nltk.corpus import stopwords |
|  | from textblob import TextBlob |
|  | from wordcloud import WordCloud |
|  |  |
|  | # Read the dataset into a pandas DataFrame |
|  | df = pd.read\_csv("engagement.csv") |
|  |  |
|  | # Remove stop words from the tweet text |
|  | # stop\_words = set(stopwords.words("english")) |
|  | # df["text"] = df["text"].apply(lambda x: " ".join( |
|  | # word for word in x.split() if word.lower() not in stop\_words)) |
|  |  |
|  | # Perform sentiment analysis using TextBlob and store the polarity score |
|  | df["polarity"] = df["text"].apply(lambda x: TextBlob(x).sentiment.polarity) |
|  |  |
|  | # Create a scatter plot of retweets vs. favorites to see how users engage with the content |
|  | sns.scatterplot(x="retweets", y="favorites", data=df) |
|  | plt.xlabel("Retweets") |
|  | plt.ylabel("Favorites") |
|  | plt.title("Engagement Analysis of Tweets") |
|  | plt.show() |
|  |  |
|  | # Create a bar chart of the most common words in the tweet text |
|  | wordcloud = WordCloud(background\_color="white", max\_words=50, |
|  | contour\_width=3, contour\_color="steelblue") |
|  | wordcloud.generate(" ".join(df["text"])) |
|  | plt.imshow(wordcloud, interpolation='bilinear') |
|  | plt.axis("off") |
|  | plt.show() |
|  |  |
|  | # Create a histogram of the polarity scores to see the distribution of sentiment in the dataset |
|  | sns.histplot(data=df, x="polarity", bins=20) |
|  | plt.xlabel("Polarity Score") |
|  | plt.ylabel("Count") |
|  | plt.title("Sentiment Analysis of Tweets") |
|  | plt.show() |

**SMA – 8: BRAND ANALYSIS**

|  |
| --- |
|  |
|  | from nltk.corpus import stopwords |
|  | from nltk.tokenize import word\_tokenize |
|  | from nltk.stem import WordNetLemmatizer |
|  | from textblob import TextBlob |
|  | import string |
|  | import matplotlib.pyplot as plt |
|  | from wordcloud import WordCloud |
|  | import warnings |
|  | warnings.filterwarnings('ignore') |
|  |  |
|  | dataset = pd.read\_csv('reviews.csv') |
|  | stops = stopwords.words('english') |
|  | lem = WordNetLemmatizer() |
|  | dataset['cleaned\_review'] = '' |
|  | dataset['sentiment'] = '' |
|  |  |
|  | corpus = '' |
|  |  |
|  | for i,row in dataset.iterrows(): |
|  | review = [token for token in word\_tokenize(str(row.review\_text).lower())] |
|  | review = [token for token in review if token not in stops] |
|  | review = [lem.lemmatize(token) for token in review if token not in stops] |
|  | review = [token.translate(str.maketrans('','',string.punctuation)) for token in review] |
|  | review = ' '.join(review) |
|  | dataset['cleaned\_review'][i] = review |
|  | senti = TextBlob(review) |
|  | senti = senti.sentiment.polarity |
|  | if senti > 0: |
|  | dataset['sentiment'][i] = 'positive' |
|  | elif senti < 0: |
|  | dataset['sentiment'][i] = 'negative' |
|  | else: |
|  | dataset['sentiment'][i] = 'neutral' |
|  |  |
|  | corpus += review + ' ' |
|  |  |
|  | cloud = WordCloud(background\_color='white',width=800,height=800, max\_words=50).generate(corpus) |
|  | plt.imshow(cloud) |
|  | plt.axis('off') |
|  | plt.title('Word Cloud for what customers are saying about Apple Iphone :-') |
|  | plt.show() |
|  |  |
|  | dataset = dataset[['cleaned\_review','sentiment']] |
|  | dataset['sentiment'].value\_counts().plot(kind='barh') |
|  | plt.xlabel('Count') |
|  | plt.ylabel('Sentiment') |
|  | plt.title('Sentiment of Reviews of IPhone') |
|  | plt.show() |