**Sentiment Analysis on Social Media Posts**

**Dataset:**

The Social Media Sentiments Analysis Dataset captures a vibrant tapestry of emotions, trends, and interactions across various social media platforms. This dataset provides a snapshot of user-generated content, encompassing text, timestamps, hashtags, countries, likes, and retweets. Each entry unveils unique stories—moments of surprise, excitement, admiration, thrill, contentment, and more—shared by individuals worldwide.

**1. Loading and Previewing Data**

df = pd.read\_csv('/content/sentimentdataset.csv')

df.head()

* The dataset is loaded into a DataFrame (df) using pd.read\_csv().
* df.head() shows the first five rows of the dataset to preview the data structure.

**2. Dropping Unnecessary Columns**

df.drop(['Unnamed: 0.1', 'Unnamed: 0', 'Timestamp', 'User', 'Month', 'Day', 'Hour'], axis=1, inplace=True)

df.head()

* This drops unnecessary columns (Unnamed: 0.1, Unnamed: 0, Timestamp, etc.) that are not needed for analysis.
* axis=1 specifies column-wise removal, and inplace=True makes the changes directly to df.

**3. Renaming Columns**

df.rename(columns={'Retweets': 'Reposts'}, inplace=True)

df.head()

* This renames the column Retweets to Reposts for clarity, reflecting a change in terminology.
* Again, inplace=True ensures that changes are made directly to the DataFrame.

**4. Data Overview**

df.shape

df.describe()

* df.shape gives the dimensions of the DataFrame (number of rows and columns).
* df.describe() provides summary statistics of the numeric columns (like mean, standard deviation, min, max, etc.).

**5. Cleaning String Columns**

df['Text']= df['Text'].str.strip()

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

df['Platform']= df['Platform'].str.strip()

df['Hashtags']= df['Hashtags'].str.strip()

df['Country']= df['Country'].str.strip()

* The .str.strip() function is used to remove any leading and trailing whitespace from string columns such as Text, Sentiment, Platform, Hashtags, and Country.

**6. Visualizing the Data**

**Plotting Top 10 Sentiments**

df['Sentiment'].value\_counts().nlargest(10).plot(kind='bar')

plt.title('Top 10 Sentiments based on Text')

plt.xlabel('Sentiment')

plt.ylabel('Count')

plt.show()

* This plots the top 10 most frequent sentiments based on the Sentiment column.
* The value\_counts() method counts occurrences of each sentiment, and nlargest(10) selects the top 10.
* The data is visualized as a bar chart.

**Plotting Platform Percentages (Pie Chart)**

df['Platform'].value\_counts().plot(kind='pie', autopct='%1.1f%%')

plt.title('Percentages of Platforms')

plt.legend()

plt.show()

* This creates a pie chart showing the distribution of the data across different platforms.
* autopct='%1.1f%%' formats the percentage display on the pie chart.

**Plotting Top 10 Countries**

df['Country'].value\_counts().nlargest(10).plot(kind='bar')

plt.title('Top 10 Country')

plt.legend()

plt.show()

* This plots a bar chart of the top 10 countries with the highest occurrence in the dataset.

**Creating and Printing a Hashtag Dictionary**

hashtags = {}

for i in df['Hashtags']:

for j in i.split():

if j in hashtags:

hashtags[j] += 1

else:

hashtags[j] = 1

* This part of the code loops through the Hashtags column, splits each hashtag by spaces, and counts the occurrence of each hashtag in the dataset. The results are stored in the hashtags dictionary.

hashtags = pd.DataFrame(hashtags.items(), columns=['Hashtag', 'Count'])

hashtags = hashtags.sort\_values(by='Count', ascending=False)

hashtags.head(10)

* The hashtags dictionary is converted into a DataFrame, sorted by count in descending order, and the top 10 hashtags are displayed.

**Printing Minimum and Maximum Values of Numeric Columns**

numerical\_columns = df[['Year', 'Likes', 'Reposts']]

for col in numerical\_columns.columns:

print(f"Minimum {col}: {df[col].min()} | Maximum {col}: {df[col].max()}")

* This code prints the minimum and maximum values for the numeric columns Year, Likes, and Reposts.

**Top Platforms by Total Likes**

top\_likes\_platform = df.groupby('Platform')['Likes'].sum().nlargest(10)

top\_likes\_platform.plot(kind='bar')

plt.title('Top Platforms by Total Likes')

plt.xlabel('Platform')

plt.ylabel('Total Likes')

plt.show()

* This groups the data by Platform and sums the Likes for each platform.
* It plots a bar chart for the top 10 platforms with the highest total likes.

**Top Platforms by Total Reposts**

top\_retweets\_platform = df.groupby('Platform')['Reposts'].sum().nlargest(10)

top\_retweets\_platform.plot(kind='bar')

plt.title('Top Platforms by Total Reposts')

plt.xlabel('Platform')

plt.ylabel('Total Reposts')

plt.show()

* Similar to the previous step, this code groups by Platform and sums the Reposts, then plots the top 10 platforms with the highest repost counts.

**Top 10 Countries by Total Likes**

top\_country\_likes=df.groupby('Country')['Likes'].sum().nlargest(10)

top\_country\_likes.plot(kind='bar')

plt.title('Country with the most likes')

plt.xlabel('Country')

plt.ylabel('count')

plt.show()

* This groups the data by Country and sums the Likes for each country, plotting the top 10 countries with the highest like counts.

**Likes and Reposts per Year (Line Plot)**

df.groupby('Year').sum()[['Likes', 'Reposts']].plot(kind='line')

plt.title('Likes and Reposts per Year')

plt.xlabel('Year')

plt.ylabel('Count')

plt.show()

* Finally, the data is grouped by Year, and the total Likes and Reposts per year are Finally, the data is grouped by Year, and the total Likes and Reposts per year are plotted on a line chart to show trends over time.

**7. Loading Pre-trained Tokenizer and Model**

tokenizer = DistilBertTokenizer.from\_pretrained('distilbert-base-uncased-finetuned-sst-2-english')

model = DistilBertForSequenceClassification.from\_pretrained('distilbert-base-uncased-finetuned-sst-2-english')

* The DistilBertTokenizer and DistilBertForSequenceClassification are loaded using the pre-trained model distilbert-base-uncased-finetuned-sst-2-english.
* This particular model is fine-tuned on the SST-2 dataset for binary sentiment classification (positive/negative).

**8. Setting Up Sentiment Analysis Pipeline**

nlp = pipeline('sentiment-analysis', model=model, tokenizer=tokenizer)

* The pipeline() function from the transformers library creates a sentiment analysis pipeline using the specified model and tokenizer.
* This pipeline allows easy application of sentiment analysis to any text input.

**9. Performing Sentiment Analysis on Text Data**

texts = list(df['Text'])

results = nlp(texts)

* The code extracts the Text column from the DataFrame df and converts it into a list texts.
* results = nlp(texts) applies the sentiment analysis model to each text, returning a list of results, where each result contains the sentiment label (POSITIVE or NEGATIVE) and the confidence score.

**10. Displaying Texts and Sentiment Results**

for text, result in zip(texts, results):

print(f"Text: {text}\nSentiment: {result['label']}, Score: {result['score']}\n")

* The for loop iterates over both texts and results simultaneously using zip().
* It prints each text along with its corresponding sentiment label and confidence score.

**11. Storing Sentiment Results in DataFrame**

df['Sentiment'] = [result['label'] for result in results]

df.head(10)

* This code adds a new column Sentiment to the DataFrame, storing the sentiment label (POSITIVE or NEGATIVE) from each result.
* df.head(10) shows the first 10 rows of the updated DataFrame.

**12. Some more Visualizations**

**Plotting Pie Chart: Positive vs Negative Sentiments**

df['Sentiment'].value\_counts().plot(kind='pie', autopct='%1.1f%%')

plt.title('Percentages of Sentiments')

plt.legend()

plt.show()

* The value\_counts() method counts the occurrences of each sentiment (POSITIVE and NEGATIVE).
* This data is plotted as a pie chart, showing the percentage breakdown of positive and negative sentiments.

**Plotting Bar Chart: Sentiments per Year**

df.groupby('Year')['Sentiment'].value\_counts().unstack().plot(kind='bar')

plt.title('Negative and Positive Sentiments per Year')

plt.xlabel('Year')

plt.ylabel('Count')

plt.show()

* The data is grouped by the Year column, and within each year, the sentiment counts are calculated.
* unstack() is used to separate the counts into individual columns for POSITIVE and NEGATIVE sentiments.
* A bar chart is plotted to compare the counts of positive and negative sentiments for each year.

**Plotting Bar Chart: Likes and Reposts in Positive vs Negative Posts**

df.groupby('Sentiment')[['Likes', 'Reposts']].sum().plot(kind='bar')

plt.title('Likes and Reposts in Positive vs Negative Posts')

plt.xlabel('Sentiment')

plt.ylabel('Count')

plt.show()

* The data is grouped by the Sentiment column (positive or negative), and the total Likes and Reposts for each sentiment category are summed.
* A bar chart is plotted to show the total number of likes and reposts in positive vs. negative posts.