Experiment No.7
Medical Reviews Analysis from social media data
Date of Performance:
Date of Submission:

Aim: To use social media data to analyse medical reviews
Objective: Given a dataset with social media data analyse, and interpret information and feedback about medical products, services, treatments, or healthcare facilities as expressed by individuals on social media platforms.
Theory:
Medical reviews analysis from social media data refers to the process of collecting, analyzing, and interpreting information and feedback about medical products, services, treatments, or healthcare facilities as expressed by individuals on social media platforms. This analysis is often conducted to gain insights into the effectiveness, safety, patient experiences, and public perceptions related to various aspects of healthcare. Here are the key components of medical reviews analysis from social media data:

Data Collection: Data is collected from various social media platforms such as Twitter, Facebook, Reddit, online forums, and healthcare-specific websites. This data can include text, images, videos, and other forms of user-generated content.

Text Mining and Natural Language Processing (NLP): Text mining and NLP techniques are used to extract and process the textual information within the collected data. This involves sentiment analysis, named entity recognition, and other linguistic and semantic analyses to understand the content and context of medical reviews.

Sentiment Analysis: Sentiment analysis is a critical component of this analysis. It helps categorize the sentiment of reviews as positive, negative, or neutral. Sentiment analysis can reveal the overall satisfaction of patients and the areas where improvements are needed.

Topic Modeling: Topic modeling techniques are used to identify and categorize common themes or topics discussed in the medical reviews. This can help healthcare organizations understand what aspects of their services or products are frequently mentioned and which areas require attention.

Adverse Event Detection: In the case of medical products or treatments, adverse event detection is crucial. By analyzing social media data, it's possible to identify potential adverse events associated with certain drugs or medical interventions.

Patient Experience Analysis: Analysis of social media data can provide insights into the patient experience, including factors like wait times, staff behavior, facility cleanliness, and the overall quality of care provided by healthcare facilities.

Public Perceptions and Trends: Social media data can reveal public perceptions about healthcare issues, emerging trends in medical treatments, and the popularity of different healthcare providers or services.

Insights for Healthcare Providers and Regulators: The findings from social media data analysis can be used to improve the quality of healthcare services and products, identify areas for improvement, enhance patient satisfaction, and ensure compliance with regulatory standards.

Data Privacy and Ethics: It's important to handle social media data with sensitivity to privacy and ethical considerations. Anonymization and consent are crucial aspects when collecting and analyzing such data.

Reporting and Decision-Making: The results of the analysis are typically reported to healthcare providers, pharmaceutical companies, regulatory agencies, and other stakeholders to inform decision-making and improve the healthcare system.

Overall, medical reviews analysis from social media data is a valuable tool for gaining insights into the real-world experiences and perceptions of patients, consumers, and the general public regarding various aspects of healthcare, which can ultimately drive improvements in healthcare quality and patient care.

Code: -

```
!pip install pandas numpy matplotlib seaborn nltk sklearn gensim
import pandas as pd
```

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation as LDA
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report, confusion_matrix
from wordcloud import WordCloud
import gensim
from gensim import corpora
import re

nltk.download('punkt')
nltk.download('stopwords')
from google.colab import files
file_path = 'social_media_reviews.csv'
df = pd.read_csv(file_path)
df.head()
print(df.isnull().sum())
df.dropna(inplace=True)
def preprocess_text(text):
```

```
    text = text.lower()  # Lowercase the text
    text = re.sub(r'\d+', '', text)  # Remove numbers
    text = re.sub(r'[^\w\s]', '', text)  # Remove punctuation
    tokens = word_tokenize(text)  # Tokenize the text
    tokens = [word for word in tokens if word not in stopwords.words('english')]
# Remove stopwords
    return ' '.join(tokens)
df['cleaned_reviews'] = df['review_text'].apply(preprocess_text)



# Display the cleaned reviews
df[['review_text', 'cleaned_reviews']].head()
X = df['cleaned_reviews']
vectorizer = CountVectorizer(max_features=1000)
X_vectorized = vectorizer.fit_transform(X)
import re
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
def preprocess_text(text):
    # Remove URLs and special characters
    text = re.sub(r'http\S+|www\S+|https\S+', '', text, flags=re.MULTILINE)
    text = re.sub(r'\@w+|\#', '', text)

    # Tokenize and remove stopwords
    tokens = word_tokenize(text)
```

```
    tokens = [word.lower() for word in tokens if word.isalpha()]
    filtered_words = [word for word in tokens if word not in stopwords.words('english')]

    return ' '.join(filtered_words)


# Apply preprocessing to the 'review_text' column
df['cleaned_reviews'] = df['review_text'].apply(preprocess_text)


# Inspect cleaned data
df[['review_text', 'cleaned_reviews']].head()
!pip install vaderSentiment
!pip install sklearn
!pip install seaborn


from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer


# Initialize VADER sentiment analyzer
analyzer = SentimentIntensityAnalyzer()


# Function to determine sentiment
```

```
def get_sentiment(text):
    score = analyzer.polarity_scores(text)
    if score['compound'] >= 0.05:
        return 'positive'
    elif score['compound'] <= -0.05:
        return 'negative'
    else:
        return 'neutral'


# Apply sentiment analysis to the cleaned reviews
df['sentiment'] = df['cleaned_reviews'].apply(get_sentiment)


# Inspect results
print(df[['cleaned_reviews', 'sentiment']].head())
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation


# Vectorize the cleaned reviews
vectorizer = CountVectorizer(max_df=0.9, min_df=2, stop_words='english')
X = vectorizer.fit_transform(df['cleaned_reviews'])
```

```
# Initialize LDA model
lda_model = LatentDirichletAllocation(n_components=5, random_state=42)
lda_model.fit(X)


# Display top words in each topic
terms = vectorizer.get_feature_names_out()
for index, topic in enumerate(lda_model.components_):
    print(f'Topic {index}:')
    print([terms[i] for i in topic.argsort()[-10:]])
import matplotlib.pyplot as plt
import seaborn as sns


# Sentiment distribution
sns.countplot(df['sentiment'])
plt.title('Sentiment Distribution')
plt.show()


# Visualize top terms in each topic
for index, topic in enumerate(lda_model.components_):
    top_words = [terms[i] for i in topic.argsort()[-10:]]
    plt.barh(top_words, topic.argsort()[-10:])
    plt.title(f'Topic {index}')
```

plt.show()

Output:

```
!pip install pandas numpy matplotlib seaborn nltk sklearn gensim

Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (2.2.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (1.26.4)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (3.7.1)
Requirement already satisfied: seaborn in /usr/local/lib/python3.10/dist-packages (0.13.1)
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.8.1)
Collecting sklearn
  Downloading sklearn-0.0.post12.tar.gz (2.6 kB)
  error: subprocess-exited-with-error

  × python setup.py egg_info did not run successfully.
  │ exit code: 1
  ╰─> See above for output.

  note: This error originates from a subprocess, and is likely not a problem with pip.
  Preparing metadata (setup.py) ... error
error: metadata-generation-failed

× Encountered error while generating package metadata.
╰─> See above for output.

note: This is an issue with the package mentioned above, not pip.
hint: See above for details.
```

```
[2] import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    import seaborn as sns
    import nltk
```

```
[2] import nltk
    from nltk.corpus import stopwords
    from nltk.tokenize import word_tokenize
    from sklearn.feature_extraction.text import CountVectorizer
    from sklearn.decomposition import LatentDirichletAllocation as LDA
    from sklearn.model_selection import train_test_split
    from sklearn.naive_bayes import MultinomialNB
    from sklearn.metrics import classification_report, confusion_matrix
    from wordcloud import WordCloud
    import gensim
    from gensim import corpora
    import re
```

```
[3] nltk.download('punkt')
    nltk.download('stopwords')

[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

```
[9] from google.colab import files
```

```
[ ] Start coding or generate with AI.
```

```
[10] file_path = 'social_media_reviews.csv'
     df = pd.read_csv(file_path)
```

```
[10] file_path = 'social_media_reviews.csv'
     df = pd.read_csv(file_path)
```

```
[11] df.head()
```

	review_id	user_name	review_text	date
0	1	user048	The staff greeted me warmly.	2024-01-01
1	2	user086	The cleanliness of the facility stood out.	2024-01-02
2	3	user079	I received a follow-up call to check on my pro...	2024-01-03
3	4	user082	The appointment was well worth my time.	2024-01-04
4	5	user001	I will recommend this clinic to my friends.	2024-01-05

```
[12] print(df.isnull().sum())

review_id      0
user_name      0
review_text    0
date           0
dtype: int64
```

```
[13] df.dropna(inplace=True)
```

```
[14] def preprocess_text(text):
         text = text.lower()  # Lowercase the text
         text = re.sub(r'\d+', '', text)  # Remove numbers
```

```
[14]    text = text.lower()  # Lowercase the text
        text = re.sub(r'\d+', '', text)  # Remove numbers
        text = re.sub(r'[^\w\s]', '', text)  # Remove punctuation
        tokens = word_tokenize(text)  # Tokenize the text
        tokens = [word for word in tokens if word not in stopwords.words('english')]  # Remove stopwords
        return ' '.join(tokens)
```

```
[18] df['cleaned_reviews'] = df['review_text'].apply(preprocess_text)

     # Display the cleaned reviews
     df[['review_text', 'cleaned_reviews']].head()
```

	review_text	cleaned_reviews
0	The staff greeted me warmly.	staff greeted warmly
1	The cleanliness of the facility stood out.	cleanliness facility stood
2	I received a follow-up call to check on my pro...	received followup call check progress
3	The appointment was well worth my time.	appointment well worth time
4	I will recommend this clinic to my friends.	recommend clinic friends

```
[20] X = df['cleaned_reviews']
```

```
[21] vectorizer = CountVectorizer(max_features=1000)
     X_vectorized = vectorizer.fit_transform(X)
```

```
[21] X_vectorized = vectorizer.fit_transform(X)
```

```
[24] import re
     import nltk
     from nltk.corpus import stopwords
     from nltk.tokenize import word_tokenize
```

```
[26] nltk.download('stopwords')
     nltk.download('punkt')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
True
```

```
[27] def preprocess_text(text):
         # Remove URLs and special characters
         text = re.sub(r'http\S+|www\S+|https\S+', '', text, flags=re.MULTILINE)
         text = re.sub(r'\@w+|\#', '', text)

         # Tokenize and remove stopwords
         tokens = word_tokenize(text)
         tokens = [word.lower() for word in tokens if word.isalpha()]
         filtered_words = [word for word in tokens if word not in stopwords.words('english')]

         return ' '.join(filtered_words)

     # Apply preprocessing to the 'review_text' column
```

```
     # Apply preprocessing to the 'review_text' column
[27] df['cleaned_reviews'] = df['review_text'].apply(preprocess_text)

     # Inspect cleaned data
     df[['review_text', 'cleaned_reviews']].head()
```

	review_text	cleaned_reviews
0	The staff greeted me warmly.	staff greeted warmly
1	The cleanliness of the facility stood out.	cleanliness facility stood
2	I received a follow-up call to check on my pro...	received call check progress
3	The appointment was well worth my time.	appointment well worth time
4	I will recommend this clinic to my friends.	recommend clinic friends

```
[31] !pip install vaderSentiment
     !pip install sklearn
     !pip install seaborn
```

```
Collecting vaderSentiment
  Downloading vaderSentiment-3.3.2-py2.py3-none-any.whl.metadata (572 bytes)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from vaderSentiment) (2.32.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->vaderSentiment) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->vaderSentiment) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->vaderSentiment) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->vaderSentiment) (2024.8.30)
Downloading vaderSentiment-3.3.2-py2.py3-none-any.whl (125 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 126.0/126.0 kB 7.0 MB/s eta 0:00:00
Installing collected packages: vaderSentiment
Successfully installed vaderSentiment-3.3.2
```

[32]
```python
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

# Initialize VADER sentiment analyzer
analyzer = SentimentIntensityAnalyzer()

# Function to determine sentiment
def get_sentiment(text):
    score = analyzer.polarity_scores(text)
    if score['compound'] >= 0.05:
        return 'positive'
    elif score['compound'] <= -0.05:
        return 'negative'
    else:
        return 'neutral'

# Apply sentiment analysis to the cleaned reviews
df['sentiment'] = df['cleaned_reviews'].apply(get_sentiment)

# Inspect results
print(df[['cleaned_reviews', 'sentiment']].head())
```

```
                   cleaned_reviews sentiment
0             staff greeted warmly  positive
1      cleanliness facility stood   neutral
2  received call check progress  positive
3    appointment well worth time  positive
4         recommend clinic friends  positive
```

[33]
```python
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation
```

[33]
```python
# Vectorize the cleaned reviews
vectorizer = CountVectorizer(max_df=0.9, min_df=2, stop_words='english')
X = vectorizer.fit_transform(df['cleaned_reviews'])

# Initialize LDA model
lda_model = LatentDirichletAllocation(n_components=5, random_state=42)
lda_model.fit(X)

# Display top words in each topic
terms = vectorizer.get_feature_names_out()
for index, topic in enumerate(lda_model.components_):
    print(f'Topic {index}:')
    print([terms[i] for i in topic.argsort()[-10:]])
```

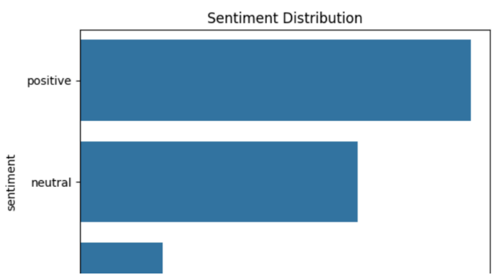
```
Topic 0:
['choose', 'reviews', 'long', 'wait', 'easy', 'place', 'worth', 'appointment', 'facility', 'staff']
Topic 1:
['improved', 'significantly', 'better', 'cautious', 'advise', 'rushed', 'thorough', 'diagnosis', 'visit', 'feel']
Topic 2:
['comfortable', 'hours', 'service', 'convenient', 'office', 'receive', 'charged', 'services', 'staff', 'felt']
Topic 3:
['appreciate', 'definitely', 'listened', 'good', 'attentive', 'caring', 'staff', 'experience', 'come', 'concerns']
Topic 4:
['recommendations', 'options', 'excellent', 'recommend', 'doctor', 'care', 'clinic', 'experienced', 'effects', 'treatment']
```

[34]
```python
import matplotlib.pyplot as plt
import seaborn as sns

# Sentiment distribution
sns.countplot(df['sentiment'])
```

[34]
```python
# Sentiment distribution
sns.countplot(df['sentiment'])
plt.title('Sentiment Distribution')
plt.show()

# Visualize top terms in each topic
for index, topic in enumerate(lda_model.components_):
    top_words = [terms[i] for i in topic.argsort()[-10:]]
    plt.barh(top_words, topic.argsort()[-10:])
    plt.title(f'Topic {index}')
    plt.show()
```



Google Collaboratory Link: -

https://colab.research.google.com/drive/19mJ0cI4oJJZ-SuxzwZxuApPDO51z9Wja

Conclusion: -

Medical review analysis from social media data is crucial for understanding public sentiment and experiences related to healthcare products and services. It enables healthcare providers and organizations to gauge patient satisfaction, identify trends, and address concerns proactively. By analyzing these reviews, stakeholders can improve patient care, tailor services to meet consumer needs, and enhance overall healthcare communication strategies. Additionally, insights gained can inform policy decisions and help in the development of targeted health campaigns.