ASSIGNMENT- 2

TEXT MINING AND TWITTER

Name: Moon Karmakar

Course Title: Data Mining

Course Code: MGT7216

Student Id: 40389123

Date: 27th April 2023

Word count: 2051

Contents

1. Introduction and Background: -

Data mining is the process of extracting patterns, correlations, and insights from enormous amounts of data. In order to find trends and patterns in data sets that are not immediately obvious to the human eye, it involves applying statistical and machine learning methodologies.
It is to extract useful information from data so that it may be used to enhance business outcomes, uncover opportunities, and make better decisions. These details may come from customer data, transactional data, website engagements, and social media activity (Chen et al., 1996).

Twitter is a popular platform for social media interaction where users post brief messages called tweets to express their ideas and thoughts. Since the COVID-19 epidemic, distance learning has drawn increased attention, particularly with the expansion of online learning. To understand more about these tweets' features and the responses they receive, we decided to analyse a collection of tweets relating to distance learning(Desai & Mehta, 2017).

In order to understand the audience's perception of distance learning, this study aims to glean meaningful information from the data, including patterns, trends, and insights. Additionally, we will do exploratory data analysis (EDA) to determine the most common terms and phrases used in these tweets as well as to investigate the sentimental state of the tweets that received the most likes and comments. Finally, using the attributes we retrieved from the text, we will create machine learning models to forecast the amount of likes a tweet is likely to get (Bifet & Frank, 2010).

The results of this study will offer insightful information that can be utilised to better understand the target audience, spot any content gaps, and direct content producers in creating user-engagement-boosting methods.

2. Methodology: -

The data mining process typically involves several steps, which are as follows:

Problem Definition: The first stage in the data mining process is to establish the problem or purpose of the investigation. The problem statement should be specific, unambiguous, and quantitative(O'Leary, 2015).

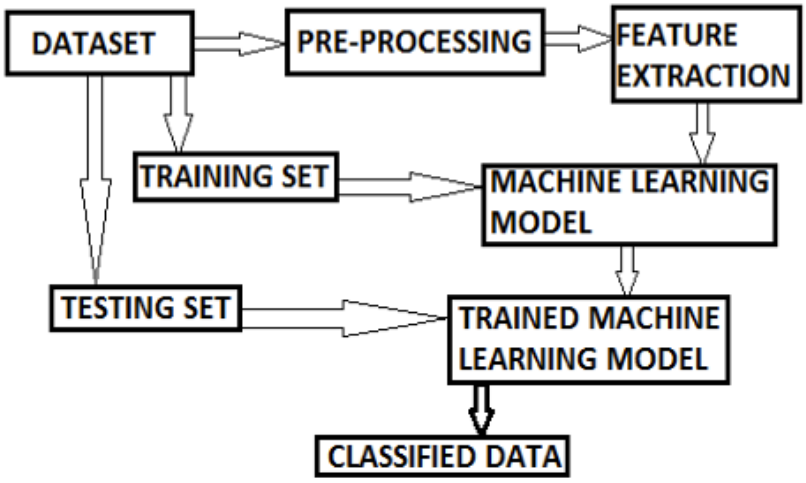
Data Collection: The required data is acquired in this step from numerous sources such as databases, data warehouses, or web sources. The information gathered should be relevant, accurate, and thorough(Jain & Katkar, 2016).

Data Preparation: - include cleaning, processing, and combining the acquired data in order to make it ready for analysis. Inconsistencies, missing values, and errors are identified in the data. The data is transformed using techniques such as data normalisation, data discretization, and feature selection(Jain & Katkar, 2016).

Data Exploration: Various approaches, like as visualisation, clustering, and association rule mining, are employed in this step to examine the data, and uncover patterns, trends, and relationships. This process aids in acquiring insights into the data and selecting significant variables for research(O'Leary, 2015).

Model Building: Various machine learning techniques are used to the data in this step to create predictive or descriptive models. The accuracy, performance, and robustness of the models are assessed(Desai & Mehta, 2017).

Model Evaluation: In this stage, the models created in the preceding step are evaluated. The models are evaluated on new data sets to ensure their correctness and generalizability(O'Leary, 2015).



Sourced : (Gupta et al., 2017)

Data Pre-processing/Featurization:

Data preparation is necessary because high-quality data are crucial and are clearly more significant than high-quality models. As a result, organisations and individuals exert a great deal of effort to clean and prepare data for modelling(Bao et al., 2014). The data in the actual world is noisy, has many quality issues, is inaccurate, and is incomplete. It can be lacking important or specific characteristics, or it might have false or spurious values. Enhancing data quality requires pre-processing. Pre-processing improves the quality of the findings, normalises the data for comparison, and removes duplicates and inconsistencies to help make data consistent(Singh & Kumari, 2016).

Importing all the necessary libraries and dependencies: -

We are using distance-learning dataset,

	Content	Retweet-Count	like	Created at
0	innovate an innovative approach #quoteoftheday...	0	0	02/08/2020 04:56
1	The pandemic is raising concerns about how tee...	0	0	02/08/2020 04:49
2	STI: Staying Education-ready in the New Normal...	0	0	02/08/2020 04:32
3	Digital Learning Through Digital RCRT\n.\n.\nR...	0	0	02/08/2020 04:30
4	Upswing Classroom: Out and Out Virtual School,...	1	0	02/08/2020 04:00

In our dataset we have 4 columns, and we have to perform all the data cleaning and featuring techniques on it.

i. First, we are checking, do we have null values in our dataset

```
Data columns (total 4 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   Content        202645 non-null  object
 1   Retweet-Count  202645 non-null  int64
 2   like           202645 non-null  int64
 3   Created at     202645 non-null  object
```

As result shows, we do not have any null values in our any of the column.

ii. The "Created at" column was transformed into a pandas datetime object using the pd.to_datetime() function to prepare the raw data for analysis. Then, using the.dt.date and.dt.time methods, we retrieved distinct columns for the date and time. Finally, we used the.drop() function with the axis argument set to 1 to eliminate the "Created at" column because it was no longer required(Symeonidis et al., 2018).

```
# convert the "created_at" column to a pandas datetime object
data['Created at'] = pd.to_datetime(data['Created at'])

# extract separate columns for the date and time
data['date'] = data['Created at'].dt.date
data['time'] = data['Created at'].dt.time
```

```
# remove the "likes" and "retweets" columns
data.drop(['Created at'], axis=1, inplace=True)
```

iii. We filtered out tweets with less than 5 words in the "content" column to make sure we have relevant tweets for research. This was accomplished by utilising the 'str.len()' method to count the amount of words in each tweet and the 'str.split()' function to break the "content" column into words. The tweets with less than 5 words in the "content" column were then removed using the boolean mask that was generated. This makes sure that our analysis is based on relevant data by excluding tweets that are probably noise or spam.

```
# filter out tweets with less than 5 words in the "content" column
data = data[data['Content'].str.split().str.len() >= 5]
```

iv. The number of rows in the DataFrame dropped from 202645 to 201117 after the "Content" column's tweets with less than 5 words were eliminated, indicating that 1528 tweets were deleted(Saar-Tsechansky & Provost, 2007).

```
#after removing tweets less than 5

data.shape
```

```
(201117, 5)
```

```
202645-201117

#1528 tweets deleted
```

```
1528
```

v. By utilising the 'datetime.now()' function to get the current date and the 'timedelta()' function to get the date 5 years ago from the current date, we can filter out tweets that are older than 5 years. The "date" column is then converted to a pandas datetime object using the 'pd.to_datetime()' function, and a boolean mask is used to remove tweets older than five years. By doing this, we make sure that the analysis we are conducting is based on more recent data, which is more likely to be relevant and indicative of current trends and viewpoints.

```
# filter out tweets that are not from the last 5 years
now = datetime.now()
five_years_ago = now - timedelta(days=365*5)
data['date'] = pd.to_datetime(data['date'])
data = data[data['date'] >= five_years_ago]
```

vi. The earliest and latest dates in our dataset's "date" column have been determined using the min() and max() procedures. The findings support our earlier step of removing tweets older than five years, which showed that we only have data from one year of the last five years accessible in our dataset.

```
min(data['date']),max(data['date'])

#We have only 1yr data,
```

```
(Timestamp('2020-01-08 00:00:00'), Timestamp('2020-12-08 00:00:00'))
```

vii. Utilising every data cleaning method on our data: -

Remove stop words: Stop words are commonly used English terms that do not add to the sense of a statement. "The," "a," "an," "and," "in," and additional phrases are examples of stop words. By excluding some terms, the data can be reduced in quantity while improving the analysis's quality(Institute of Electrical and Electronics Engineers, n.d.).

Remove punctuation: Commas, periods, slashes, and question marks may all be removed using regular expressions or Python's built-in string module(Institute of Electrical and Electronics Engineers, n.d.).

All content should be written in lowercase. This helps to standardise data and makes analysis easier.

Remove URLs: To get rid of URLs that are commonly included in Twitter data, utilise the Python re package or regular expressions(Jianqiang & Xiaolin, 2017).

Eliminate user mentions: Twitter data commonly includes user mentions, which may be removed using normal expressions or by dividing the tweet and removing any terms that start with the '@' sign(Jianqiang & Xiaolin, 2017).

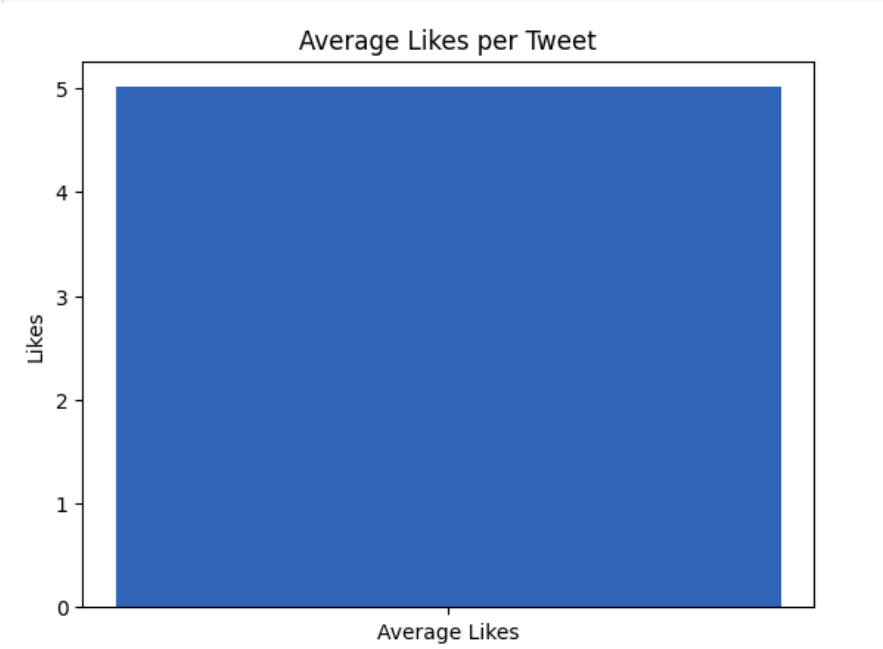
Remove hashtags: Data from Twitter often includes hashtags. These may be removed either via the use of standard expressions or by dividing the tweet and removing any words that start with the symbol "#" (Jianqiang & Xiaolin, 2017)

viii. After completing all data cleaning stages, we are checking for the words that are most frequently used in our tweets:
The tweet text is converted to lowercase, divided into words using the Counter function, and the top 10 most frequent terms are then displayed.

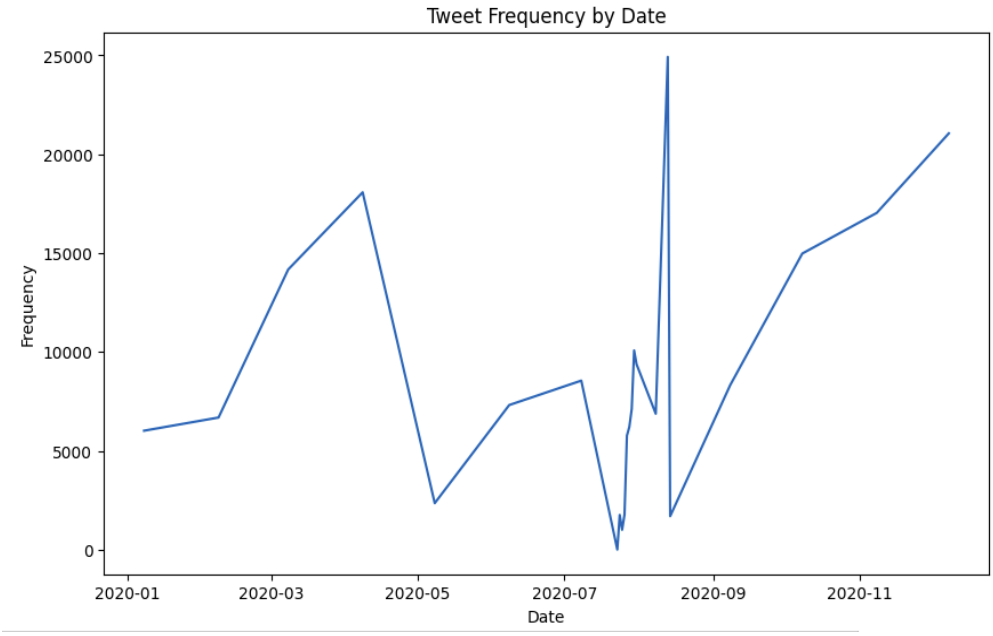
```
# Convert the text column to lowercase and split into words
words = data['Content'].str.lower().str.split()

# Count the frequency of each word
word_counts = Counter(word for tweet in words for word in tweet)

# Print the top 10 most common words
print(word_counts.most_common(10))
```

```
[('online', 126889), ('i', 48797), ('learning', 44256), ('class', 37580), ('course', 35701), ('teaching', 28221), ('education', 23235),
```

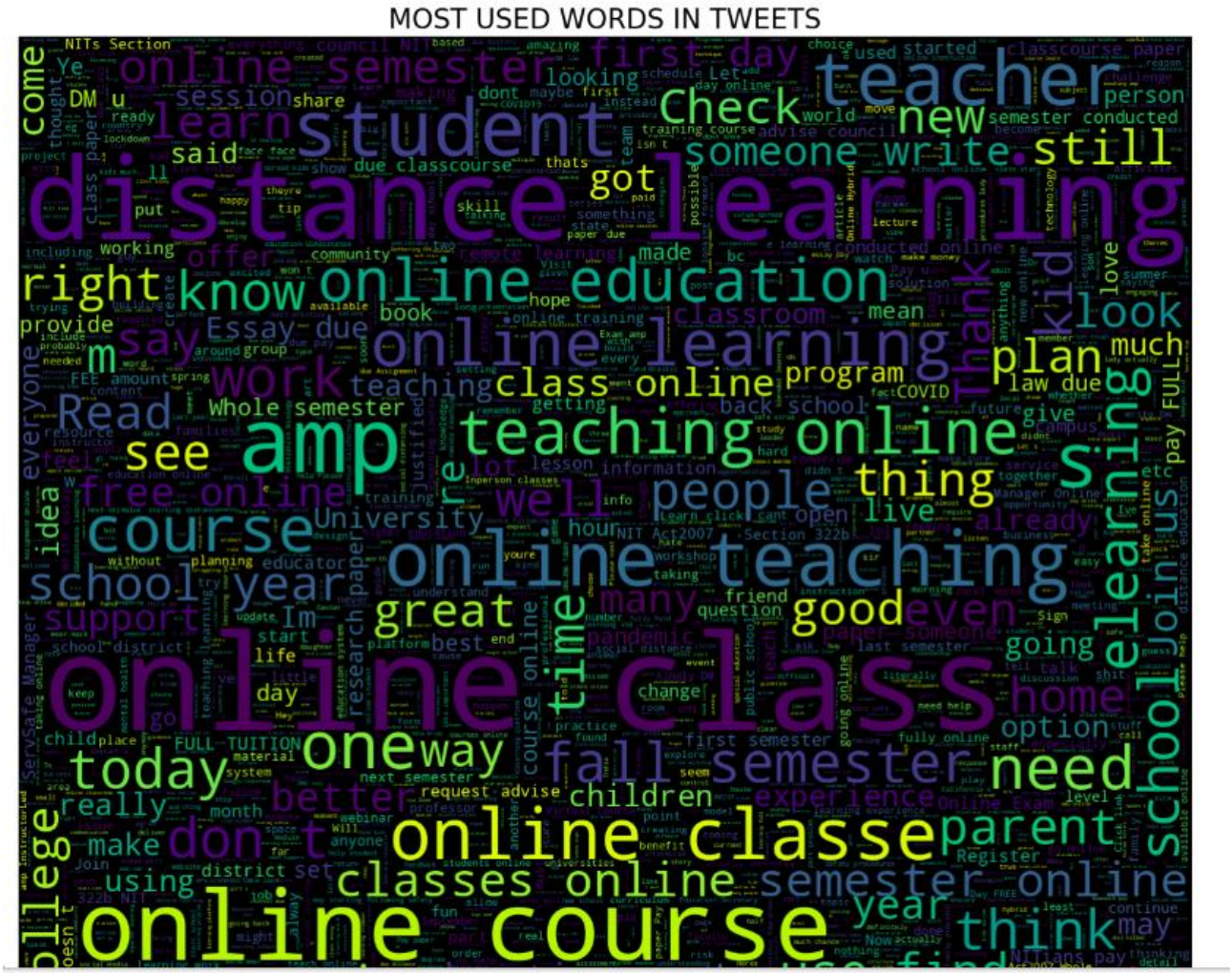
Average Likes per Tweet



Tweet Frequency by Date: -



Exploratory Data Mining Tasks: -



Here we are seeing word cloud, all the most and least used word in the tweets, the most frequent words are all the bold one's, the least word used in the tweets are smaller one's (Shahid et al., 2017).
Word clouds can quickly explain the most frequently used words and phrases in a corpus of text(Kaurav et al., 2020).

Sentimental Modelling

```
# Select 10,000 rows randomly
sampled_data = data.sample(n=1000, random_state=42)
```

For doing sentimental analysis we took 10,000 sample data because 200,000 data will take too much time for computation(Aljedaani et al., 2022).

Top 5 Tweets for Sentiment Analysis

We initially use the sort_values() method to organise the data by the "like" column in chronological sequence in order to identify the top 5 tweets with the most likes. The 'head()' method is then used to choose the top 5 rows.

We employ the VADER (Valence Aware Dictionary and sEntiment Reasoner) lexicon provided by the Natural Language Toolkit (nltk) package to do sentiment analysis on the tweets. Using 'nltk.download()', we first collect the VADER lexicon. Then, using the nltk library, we develop a 'SentimentIntensityAnalyzer()' object that enables us to assess each tweet's emotional content(Zeal Education Society et al., n.d.).

Performing Sentiment Analysis on Top 5 Tweets

```
Tweet:  UNESCO says 85  children  subSaharan Africa   learning But   children attend  religious  private school eg  madrasa  counts    learning And
Sentiment score:  0.0
Sentiment:  positive
Tweet:  kumulo dugo ko sa bagong content ng vincentiments  get  online classes    ideal way  continue  education trust   hate  idea  well pero entirely
Sentiment score:  -0.101
Sentiment:  negative
Tweet:  Folks  12 VAT isnt   Netflix Lazada etc Pati po online trainings elearning facilities webinars payment facilities conduits     12 VAT Instead
Sentiment score:  -0.5574
Sentiment:  negative
Tweet:  Will start teaching online law school classes next week 🤩 Good luck   students 😈😇🤣
Sentiment score:  0.7096
Sentiment:  positive
Tweet:  Q Will   special stages   online concert Dahyun Of course Tzuyu There
Sentiment score:  0.4019
Sentiment:  positive
```

After performing sentimental analysis on top 5 tweets, we are performing sentiment analysis on all data, which are going to be used in our next task.

	Content	Retweet-Count	like	date	time	sentiment_score
154938	COVID caused classes taught online upc...	0	0	2020-04-08	21:57:00	-0.0258
149158	Studentcentered syllabus language	0	0	2020-08-13	15:09:00	0.0000
133971	CONNECTIONS ACADEMY 💻👩‍🏫👨‍🏫👩‍🏫 Online RealWo...	0	0	2020-08-08	22:31:00	0.6486
125280	Edtech firm upGrad Education Pvt Ltd offers o...	0	0	2020-08-13	06:50:00	0.0000
127630	IC gt Purdue University Online personalized le...	0	0	2020-12-08	07:10:00	0.4939

Machine Learning Task

Here we are going to create a pipeline for machine learning to extract tweet attributes and forecast likes.

a. First, we used Stemming:

Stemming is a technique used in natural language processing and information retrieval to reduce words to their base or root form. The goal of stemming is to group together words that have the same root, even if they are inflected or have different suffixes or prefixes. For example, the words "jump", "jumps", "jumped", and "jumping" can all be reduced to the root word "jump" through stemming(M. Arif & Mustapha, 2017).

The main reason to use stemming is to improve the accuracy and efficiency of information retrieval systems. By reducing words to their base form, stemming can help to overcome the problem of variability in natural language and reduce the size of the vocabulary used in a text. It is commonly used in a variety of applications, such as search engines, text classification, and text mining. However, stemming is not always perfect and can sometimes produce false matches or incorrectly group together words with different meanings. To ensure the accuracy and reliability of text analysis, stemming should be used in conjunction with other techniques for natural language processing, such as lemmatization(Gautam & Yadav, 2014).

b. After stemming we have operated vectorisation on Content column

Vectorization is the process of converting raw data into a numerical format that can be used for machine learning and data analysis. This involves representing each data point as a vector, which is a sequence of numbers that encodes the properties or features of the data(Kabra & Nagar, 2023).

```
Shape of vectorized data: (1000, 1000)
```

c. Then we split our dataset into train and test datasets, that going to perform regression modelling(Deshwal & Sharma, 2016).

```
# split the dataset into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Training Models for the dataset: -

Linear Regression model, Decision Tree model, XG Boost model

```
# Training the Models
linr = linr.fit(X_train, y_train)
xg = xg.fit(X_train,y_train)
dt = dt.fit(X_train,y_train)
```

3.Result and Discussion: -

Based on the metrics, we can compare the performance of the three models as follows:

Linear regression model:
Mean Squared Error (MSE): 1.8426985769864717e+18
R-squared Score: -1515639544796388.0
The linear regression model has a very high MSE and a negative R-squared score, indicating that the model performs very poorly and is not a good fit for the data.

Decision Tree model:
Mean Squared Error (MSE): 588.1312037037037
R-squared Score: 0.5162557126842682
The Decision Tree model has a lower MSE than the Linear Regression model, indicating that it is performing better. The R-squared score of 0.51 indicates that the model explains approximately 51% of the variance in the target variable.

XGBoost model:
Mean Squared Error (MSE): 576.7325482953056
R-squared Score: 0.5256312302595456
The XGBoost model has a slightly lower MSE than the Decision Tree model, and a slightly higher R-squared score of 0.53, indicating that it performs slightly better than the Decision Tree model.

In summary, the XGBoost model outperforms both the Linear Regression model and the Decision Tree model, with the lowest MSE and the highest R-squared score. However, the specific metrics and the overall performance of each model may depend on the specific data set and the problem being addressed(Alom et al., 2018).

4.Conclusion

In conclusion, we thoroughly mined and analysed tweets associated to distance learning. Exploratory data analysis was performed to uncover the most frequently used words and phrases as well as the tone of the tweets that earned the most likes and comments after the data had been cleansed and prepped. Finally, we developed a machine learning model to predict how many likes a tweet will probably get(Ullah et al., 2021).

The results of this study may be used to better understand how the audience views distant learning, spot content shortages, and suggest strategies for increasing user engagement. The study's conclusions may be leveraged to create more interesting material, increasing user engagement and social media interactions(Jain & Katkar, 2016).

References: -

Aljedaani, W., Rustam, F., Mkaouer, M. W., Ghallab, A., Rupapara, V., Washington, P. B., Lee, E., & Ashraf, I. (2022). Sentiment analysis on Twitter data integrating TextBlob and deep learning models: The case of US airline industry. *Knowledge-Based Systems*, *255*, 109780. https://doi.org/10.1016/J.KNOSYS.2022.109780

Alom, Z., Carminati, B., & Ferrari, E. (2018). Detecting spam accounts on Twitter. *Proceedings of the 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2018*, 1191–1198. https://doi.org/10.1109/ASONAM.2018.8508495

Bao, Y., Quan, C., Wang, L., & Ren, F. (2014). The role of pre-processing in twitter sentiment analysis. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8589 LNAI*, 615–624. https://doi.org/10.1007/978-3-319-09339-0_62/COVER

Bifet, A., & Frank, E. (2010). Sentiment knowledge discovery in Twitter streaming data. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *6332 LNAI*, 1–15. https://doi.org/10.1007/978-3-642-16184-1_1/COVER

Chen, M. S., Han, J., & Yu, P. S. (1996). Data mining: An overview from a database perspective. *IEEE Transactions on Knowledge and Data Engineering*, *8*(6), 866–883. https://doi.org/10.1109/69.553155

Desai, M., & Mehta, M. A. (2017). Techniques for sentiment analysis of Twitter data: A comprehensive survey. *Proceeding - IEEE International Conference on Computing, Communication and Automation, ICCCA 2016*, 149–154. https://doi.org/10.1109/CCAA.2016.7813707

Deshwal, A., & Sharma, S. K. (2016). Twitter sentiment analysis using various classification algorithms. *2016 5th International Conference on Reliability, Infocom Technologies and Optimization, ICRITO 2016: Trends and Future Directions*, 251–257. https://doi.org/10.1109/ICRITO.2016.7784960

Gautam, G., & Yadav, D. (2014). Sentiment analysis of twitter data using machine learning approaches and semantic analysis. *2014 7th International Conference on Contemporary Computing, IC3 2014*, 437–442. https://doi.org/10.1109/IC3.2014.6897213

Gupta, B., Negi, M., Vishwakarma, K., Rawat, G., & Badhani, P. (2017). Study of Twitter Sentiment Analysis using Machine Learning Algorithms on Python. *International Journal of Computer Applications*, *165*(9), 29–34. https://doi.org/10.5120/ijca2017914022

Institute of Electrical and Electronics Engineers. (n.d.). *2018 International Arab Conference on Information Technology (ACIT).*

Jain, A. P., & Katkar, V. D. (2016). Sentiments analysis of Twitter data using data mining. *Proceedings - IEEE International Conference on Information Processing, ICIP 2015*, 807–810. https://doi.org/10.1109/INFOP.2015.7489492

Jianqiang, Z., & Xiaolin, G. (2017). Comparison research on text pre-processing methods on twitter sentiment analysis. *IEEE Access*, *5*, 2870–2879. https://doi.org/10.1109/ACCESS.2017.2672677

Kabra, B., & Nagar, C. (2023). Convolutional Neural Network based sentiment analysis with TF-IDF based vectorization. *Journal of Integrated Science and Technology*, *11*(3), 503–503. https://pubs.thesciencein.org/journal/index.php/jist/article/view/503

Kaurav, R. P. S., Suresh, K. G., Narula, S., & Baber, R. (2020). NEW EDUCATION POLICY: QUALITATIVE (CONTENTS) ANALYSIS AND TWITTER MINING (SENTIMENT ANALYSIS). *Journal of Content, Community and Communication*, *12*, 4–13. https://doi.org/10.31620/JCCC.12.20/02

M. Arif, S., & Mustapha, M. (2017). The Effect of Noise Elimination and Stemming in Sentiment Analysis for Malay Documents. *Proceedings of the International Conference on Computing,*

Mathematics and Statistics (ICMS 2015), 93–102. https://doi.org/10.1007/978-981-10-2772-7_10/COVER

O'Leary, D. E. (2015). Twitter Mining for Discovery, Prediction and Causality: Applications and Methodologies. *Intelligent Systems in Accounting, Finance and Management*, *22*(3), 227–247. https://doi.org/10.1002/isaf.1376

Saar-Tsechansky, M., & Provost, F. (2007). Handling missing values when applying classification models. *Journal of Machine Learning Research*, *8*, 1625–1657. https://www.jmlr.org/papers/volume8/saar-tsechansky07a/saar-tsechansky07a.pdf

Shahid, N., Ilyas, M. U., Alowibdi, J. S., & Aljohani, N. R. (2017). Word cloud segmentation for simplified exploration of trending topics on Twitter. *IET Software*, *11*(5), 214–220. https://doi.org/10.1049/iet-sen.2016.0307

Singh, T., & Kumari, M. (2016). Role of Text Pre-processing in Twitter Sentiment Analysis. *Procedia Computer Science*, *89*, 549–554. https://doi.org/10.1016/J.PROCS.2016.06.095

Symeonidis, S., Effrosynidis, D., & Arampatzis, A. (2018). A comparative evaluation of pre-processing techniques and their interactions for twitter sentiment analysis. *Expert Systems with Applications*, *110*, 298–310. https://doi.org/10.1016/J.ESWA.2018.06.022

Ullah, H., Ahmad, B., Sana, I., Sattar, A., Khan, A., Akbar, S., Zubair Asghar, M., Khan, I., Pakhtunkhwa, K., & Khan Khyber Pakhtunkhwa, I. (2021). Comparative study for machine learning classifier recommendation to predict political affiliation based on online reviews. *Wiley Online Library*, *6*(3), 251–264. https://doi.org/10.1049/cit2.12046

Zeal Education Society, Zeal Institute of Business Administration, C. A. and R., Institute of Electrical and Electronics Engineers. Pune Section, & Institute of Electrical and Electronics Engineers. (n.d.). *2017 International Conference on Data Management, Analytics and Innovation (ICDMAI) : Zeal Education Society, Pune, India, Feb 24-26, 2017.*

#Appendix: -

```
#Importing all necessary libraries and dependencies

import pandas as pd
from datetime import datetime, timedelta


data = pd.read_csv("/content/drive/MyDrive/ass./distance learning-1.csv")


# convert the "created_at" column to a pandas datetime object
data['Created at'] = pd.to_datetime(data['Created at'])
# extract separate columns for the date and time
data['date'] = data['Created at'].dt.date
data['time'] = data['Created at'].dt.time
# remove the "likes" and "retweets" columns
data.drop(['Created at'], axis=1, inplace=True)

## Task 1
# filter out tweets with less than 5 words in the "content" column
data = data[data['Content'].str.split().str.len() >= 5]


# filter out tweets that are not from the last 5 years
now = datetime.now()
five_years_ago = now - timedelta(days=365*5)
data['date'] = pd.to_datetime(data['date'])
data = data[data['date'] >= five_years_ago]

min(data['date']),max(data['date'])
#We have only 1yr data,




import re
def remove_urls(text):
return re.sub(r'http\S+', '', text)
data['Content']=data['Content'].apply(remove_urls)



def remove_mentions_hashtags(text):
return re.sub(r'@\w+|#\w+', '', text)
data['Content']=data['Content'].apply(remove_mentions_hashtags)

import string
```

```
def remove_punctuation(text):
"""
Removes punctuation from the input text using Python's string library.
"""
translator = str.maketrans('', '', string.punctuation)
return text.translate(translator)

data['Content']=data['Content'].apply(remove_punctuation)



#For removing stopwords
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords


def remove_stopwords(text):
new_text = []
for word in text.split():
if word in stopwords.words('english'):
new_text.append('')
else:
new_text.append(word)
x = new_text[:]
new_text.clear()
return " ".join(x)


data['Content']=data['Content'].apply(remove_stopwords)




# Task 2 statistical analysis:


import matplotlib.pyplot as plt


# Calculate the average number of likes
avg_likes = data['like'].mean()

# Create a bar chart showing the average likes
fig, ax = plt.subplots()
ax.bar(['Average Likes'], [avg_likes])
ax.set(title='Average Likes per Tweet', ylabel='Likes')
plt.show()
```

```
# Count the number of tweets per day
tweet_counts = data['date'].value_counts().sort_index().reset_index()
tweet_counts.columns = ['date', 'count']

# Create a line plot of the tweet frequency by date
fig, ax = plt.subplots(figsize=(10, 6))
ax.plot(tweet_counts['date'], tweet_counts['count'])
ax.set(title='Tweet Frequency by Date', xlabel='Date', ylabel='Frequency')
plt.show()
```

```
#some visualization of reviews
from PIL import Image
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import re
import nltk
# from textblob import Word, TextBlob
import warnings
warnings.filterwarnings('ignore')
```

```
plt.subplots(figsize=(16,13))
wordcloud = WordCloud(
background_color='black',max_words=5000,
width=1200,stopwords=STOPWORDS,
height=1000
).generate(" ".join(data['Content']))
plt.title("MOST USED WORDS IN TWEETS",fontsize=20)
plt.imshow(wordcloud.recolor( colormap= 'viridis'))
plt.axis('off')
plt.show()
```

```
# Select 10,000 rows randomly
sampled_data = data.sample(n=1000, random_state=42)
from nltk.sentiment import SentimentIntensityAnalyzer
# Sort the data based on the like column
sorted_data = data.sort_values(by=['like'], ascending=False)
# Select the top 5 rows
top_5_tweets = sorted_data.head(5)
```

```
import nltk
```

```
nltk.download('vader_lexicon')

# Create a sentiment analyzer object
sia = SentimentIntensityAnalyzer()


# Compute the sentiment score for each of the top 5 tweets
for text in top_5_tweets['Content']:
scores = sia.polarity_scores(text)
sentiment = 'positive' if scores['compound'] >= 0 else 'negative'
print("Tweet: ", text)
print("Sentiment score: ", scores['compound'])
print("Sentiment: ", sentiment)




# Create a sentiment analyzer object
sia = SentimentIntensityAnalyzer()

# Compute the sentiment score for each tweet in the data
sentiment_scores = []
for text in sampled_data['Content']:
scores = sia.polarity_scores(text)
sentiment_scores.append(scores['compound'])

# Add a new column to the DataFrame with the sentiment scores
sampled_data['sentiment_score'] = sentiment_scores

# Compute the overall sentiment of the data
overall_sentiment = sampled_data['sentiment_score'].mean()

print("Overall sentiment score: ", overall_sentiment)



import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.feature_extraction.text import TfidfTransformer


#stemming
from nltk.stem.porter import PorterStemmer
from textblob import Word, TextBlob

import nltk
nltk.download('wordnet')
```

```
ps = PorterStemmer()
sampled_data['Content'] = sampled_data['Content'].apply(lambda x: " ".join([Word(word).lemmatize() for word
in x.split()]))


# creating X and y variables
X = sampled_data['Content']
y = sampled_data['like']


from sklearn.feature_extraction.text import TfidfVectorizer
# Create a TfidfVectorizer object with desired parameters
tfidf_vectorizer = TfidfVectorizer(max_features=1000, stop_words='english')
# Fit and transform the text data
X_vectorized = tfidf_vectorizer.fit_transform(sampled_data['Content'])
# Print the shape of the resulting matrix
print('Shape of vectorized data:', X_vectorized.shape)


# Combine the text data and other features into a single matrix
X = np.hstack([X_vectorized.toarray(), sampled_data[['Retweet-Count', 'sentiment_score']].values])


# Combine the text data and other features into a single matrix
X = np.hstack([X_vectorized.toarray(), sampled_data[['Retweet-Count', 'sentiment_score']].values])


from sklearn.linear_model import LinearRegression
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Ridge
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from sklearn.svm import SVR
from sklearn.metrics import r2_score,mean_squared_error



linr = LinearRegression()
dt = DecisionTreeRegressor()
xg = XGBRegressor()

# Training the Models
linr = linr.fit(X_train, y_train)
xg = xg.fit(X_train,y_train)
dt = dt.fit(X_train,y_train)



# make predictions using the trained regression model
```

```
y_pred = linr.predict(X_test)

# calculate MSE and RMSE
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)

# calculate R-squared score
r2 = r2_score(y_test, y_pred)

# print the results
print("Linear regression model")
print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
print("R-squared Score:", r2)



# make predictions using the trained regression model
y_pred = dt.predict(X_test)

# calculate MSE and RMSE
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)

# calculate R-squared score
r2 = r2_score(y_test, y_pred)

# print the results
print("Decision Tree model")
print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
print("R-squared Score:", r2)



# make predictions using the trained regression model
y_pred = xg.predict(X_test)

# calculate MSE and RMSE
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)

# calculate R-squared score
r2 = r2_score(y_test, y_pred)

# print the results
print("Xg boost model")
print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
print("R-squared Score:", r2)
```