Sentiment Analysis of Tweets and Texts Using Python on Stocks and COVID-19

Yuvraj Jain1* and Vineet Tirth2

1K.R. Mangalam World School, Greater Kailash II, Chittaranjan, New Delhi-110048, India
2College of Engineering, King Khalid University, Abha-61411, Asir, Kingdom of Saudi Arabia

ABSTRACT

In a day and age where billions of users regularly use social media and express their opinions online, there is potentially a lot of data that can be harvested and utilized; therefore, it is crucial to develop a quick way to garner data. This study aimed to develop a program using Python to do the same, trying to understand the sentiments of the authors of the text and tweets as well as other additional information about the top tweets and retweets. The main objective of the Twitter Sentiment Analysis is a query-based analysis of tweets. In simple words, Twitter Sentiment Analysis focuses on analyzing the tweets of a specific/particular topic that the user wants to analyze. An extensive collection of such sentiments could leverage to provide a fair reflection of public sentiment towards a specific topic. There are thousands of tweets that can be quickly processed for the sentimental impact, compared to the amount of time it would take a large team of people to complete the same task manually. There are tons of text documents that can also be processed for sentiments in seconds, much faster than just a team of people manually skimming through the text.

Keywords: Tokenization; Stemming; Lemmatization; Bag of Words; Text Frequency; Inverse Document Frequency.

Context: This is an independent study made during the lockdown period of COVID-19 pandemic. It took about five months to complete this study, summarized in this article.

Motivation: The use of social media and twitter has increasingly become popular in the pandemic period. The effect of the COVID-19 pandemic on the sentiments of the society, mainly due to the tweets and comments of the influential people of the society and the response to their tweets was the primary source of motivation behind this study.

Source code: The source code for the study will be made available in the public domain through the link to the GitHub.
1. INTRODUCTION

During the pandemic of COVID-19, people all over the world have been affected, ranging from ordinary people to organizations, geopolitical entities, and entire countries. People have been expressing their sentiments about the pandemic on Twitter which is an essential outlet for people to discuss their feelings and opinions. This data can then be mined upon to extrapolate useful information that can be beneficial to gain an understanding of the effects caused by COVID-19 to companies, organizations and also to gain an insight into how people are emotionally reacting to the response of their respective governments to the global pandemic. For instance, during COVID-19 countries like Australia, Belgium and India were tweeting with a positive sentiment, whereas the Chinese had expressed negative sentiments for the same [1-2]. Sentiment analysis is the contextual mining of text, which identifies and withdraws subjective information from the source material that can be useful in many scientific and commercial areas, such as event detection, recommender systems, and opinion mining [3].

2. METHODOLOGY

The first step in determining the sentiment of tweets involves extracting the text and processing it first. This is done through Tokenization, which is just a process of breaking up a piece of text into many pieces, such as sentences and words. It works by separating words using spaces and punctuation (Figure 1).

![Figure 1: Tokenization of information.](image)

Here, we obtain a list of individual words which can later be processed to determine the sentiment of the text. In Figure 2, ‘paragraph’ is a variable, which is currently holding a string of a random text to showcase this example of Tokenization.

![Figure 2: An example to determine the text sentiment.](image)
2.1 Problems Faced with Tokenization

Because Tokenization yields a list of every single word in the text, it is practically impossible to operate on the data that will be procured. This problem would just be aggravated by the enormous data set of tweets of millions of users. The data set needs to be reduced to a manageable amount. This is done through Lemmatization, Stemming, and Stop-words.

2.1.1 Removing Stop-words

Stop-words are words that do not contribute to the sentiment of the text, and they do not have much value in a sentiment analysis program. Hence, they are just noise in the text that may be removed [4]. Below are the stop-words, which have been removed.

['he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'only', 'other', 'some', 'such', 'no', 'nor', 'not', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'hadn', "hadn't", 'hasn', "hasn't", 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'few', 'more', 'most', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'a', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't", 'i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yours elf', 'yourselves']

2.1.2 Stemming with 'ntlk' Package

Through stemming, we decrease the inflection in words to their root forms. This is done by mapping a group of words to the same stem even if the root word or the stem itself is not a valid word in the language [5-6]. Through stemming, the data set can be reduced as a lot of similar words would be clubbed into a single group under a common title of their root word. The image below explains how stemming works. A list of words was obtained, where first the stopwords have been stripped, and second the words have been reduced to their root form, as visible in ‘Out[12]’, which represent the output (Figure 4).
2.1.3 Lemmatization with ‘ntlk’ package

Lemmatization, unlike Stemming, reduces the inflected words properly i.e. making sure that the root word is grammatically correct and belongs in the language. In Lemmatization, the root word is called Lemma whereas the root is called stem in Stemming. (Figure 4).

In Figure 5, we obtain a list of words from the sentence. Firstly, the stop-words have been stripped, and secondly, the words have been lemmatized to form meaningful root words. Therefore, after performing these steps, our data set would be reduced to a manageable size that could be passed through a query and would yield results fairly efficiently, making the program run faster and smoother.

2.2 Bag of Words

Natural Language Processing (NLP) doesn’t directly work on text or special symbols. It can only process numbers. Therefore, we cannot directly feed our text into that algorithm. Here, Bag of Words model is useful because it converts the text into a vector, which helps keep a count of the total frequency of most commonly occurring used words. A bag-of-words represents text by giving us information about the most...
frequently word in the document. It involves two things. 1. A vocabulary of known words. 2. A measure of the presence of known words. It is called a “bag” of words, because any irrelevant information about the location and structure of data in the document is discarded by preprocessing the data (Figure 6). The model is only interested in the frequency with which the text appears in the document, and not the location where it does.

2.2.1 Steps to Prepare Bag of Words Model

The steps to prepare Bag of Words model are given hereunder.

**Step 1:** Remove all non-word characters and punctuations. (‘.’, ‘?’ , ‘!’ , ‘1’, ‘$’)
**Step 2:** Convert all the text to lowercase.
**Step 3:** Obtain the bag of words model by obtaining the most frequent words in our text.
**Step 4:** Convert this data into an array. Figure 7 gives the code for creating a Bag of Words model.
2.2.2 Problems associated with the Bag of Words Model

Although the Bag of Words model is useful in finding the most frequent words in the text and arranging it in an array, it cannot assign importance to different words. While working in a sentiment analysis program, certain words are more important than others, and it is crucial to assign this importance. To address this, \textit{TF*IDF} (Text Frequency - Inverse Document Frequency) is utilized (Equation 1 and 2).

2.2.3 About \textit{TF*IDF}

TF*IDF combines the measurement of how regularly a term is used on a page and the measurement of how often that term appears in all pages of a collection to assign the importance of that term to the page. A short example is given to illustrate the benefit of using TF*IDF for assigning importance to words [7-8].

\textbf{Term Frequency (TF):} We can count the number of times each term occurs in each document to distinguish a text document. The number of times a term occurs in a document is called its term frequency, estimated by Equation 1 [9].

\[
\text{tf}(t, d) = \frac{\text{Count of } t \text{ in } d}{\text{Total Word Count of } d} \quad (1)
\]

\(t = \text{Any word in the text}\)
\(d = \text{Text document}\)

\textbf{Inverse Document Frequency (IDF):} Every term is considered equally important while determining the term frequency; however, it is known that specific terms, such as "is", "of", and "that", are present numerous times in our document but have little to no significance in determining the sentiment. Thus, frequently appearing terms are assigned a low score whereas rarely appearing terms are assigned a higher score. IDF helps assign this importance justifiably. When IDF is calculated, it will be very low for the most occurring words such as stop words (because stop words such as "is" is present in most of the documents, and \(N/df\) will give a low value to that word). This finally gives the relative weightage (Equation 2).

\[
idf(t) = \frac{N}{df} \quad (2)
\]

\(N = \text{Number of sentences in the text document}\)
\(df = \text{Count of occurrences of term } t \text{ in the document set } N\)
Now there are few other problems with the IDF, in case of a large corpus, say 100,000,000, the IDF value explodes. To avoid this, we take its logarithm. This is known as log normalization (Equation 3).

\[
idf(t) = \log\left(\frac{N}{df+1}\right)
\]  \hspace{1cm} (3)

**Final Formula Obtained:**
Upon multiplying term frequency and inverse document frequency (Figure 8), we obtain Equation 4.

\[
tf \times idf(t, d) = tf(t, d) \times \log\left(\frac{N}{(df+1)}\right)
\]  \hspace{1cm} (4)

A decimal value is hence obtained for all the different words in the text, giving the importance of the words in the document. The higher is the value; more is the importance assigned to it. This helps solve the problem of the Bag of Words model, which was unable to give importance to different words.

```
In [7]: # Creating the TF-IDF model
from sklearn.feature_extraction.text import TfidfVectorizer
cv = TfidfVectorizer()
X = cv.fit_transform(corpus).toarray()
```

**Figure 8:** Creating a TF*IDF model.

‘TfidVectorizer’ is a function, which is a part of the ‘sklearn’ library is used in creating the TF-IDF model (Figure 8).

For instance, if the corpus contains the following (Figure 9), the output generated with the use of **TF-IDF** would be as shown in Figure 10.

```
In [6]: corpus
Out[6]: ['paper describes weakly supervised system sentiment analysis movie review domain', 'objective classify movie review polarity class positive negative based sentence bearing opinion movie alone leaving irrelevan t text',]
```

**Figure 9:** An example showcasing TD-IDF in action.

### 2.3 Stock Sentiment Analysis

By feeding input and the target output, a model can be trained to predict future outputs whenever a new input is given. This is the utilization of **supervised learning**. Supervised learning is the machine learning task where the labeled data i.e. input and the output have already been provided, and based on this data the machine trains itself to predict future outputs. (Figure 11). It infers to a function from labelled training data consisting of a set of training examples. Using this approach, we can create a model to predict whether the prices of a stock will increase or decrease in the future based on the data fed to it in the past [10].
Steps followed in stock sentiment analysis are given hereunder.

**Step 1: Reading the Data:** Firstly, a labelled set of data is required, which includes information about various stocks at different instances of time using a 'stock_analysis.csv' file that contains this information. This data is read by using the following code (Figure 12).

```python
In [1]: import pandas as pd
In [2]: df = pd.read_csv('P:\datasets\stock_analysis.csv', encoding = 'ISO-8859-1') # 8-bit single-byte coded graphic character sets
```

**Figure 12:** Reading the csv file.

**Step 2: Segregating data b/w Training & Testing data:** Second, the data is segregated into two different groups: a training set and a test set, respectively. The training data is used to train the model for predicting the output. After the model has
been trained, the test data is used to check the accuracy of the model. This code is
divided based on the date when the tweet was published (Figure 13).

```python
In [5]:
train = df[df['Date'] < '20150101']
test = df[df['Date'] > '20141231']
```

**Figure 13:** Segregating the data.

**Step 3: Punctuation and Lower Case:** The punctuation is removed, i.e. anything
which is not a letter such as '#', '123','$' etc., is replaced with a space in the text (Figure
14). Then, the headlines are converted into lowercase (Figure 15) to prevent the
discrepancy caused due to Python being Case sensitive and reading uppercase and
lowercase words differently.

```python
In [8]:
# Removing punctuations
data_train.iloc[:,2:27]
data.replace(’[^a-zA-Z]”, ”,regex=True, inplace=True)

# Renaming column names for ease of access
list1= [i for i in range(25)]
new_Index=[str(i) for i in list1]
data.columns= new_Index
data.head(5)
```

**Figure 14:** Removing punctuations.

```python
In [9]:
# Converting headlines to lower case
for index in new_index:
data[index]=data[index].str.lower()
data.head(5)
```

**Figure 15:** Converting to lowercase.

**Step 4: Implementing Bag of Words and Random Forest Classifier:** Bag of words
model is used to convert text to vectors as mentioned under section 2.2 using Random
Forest Classifier (Figure 16) to train the model to predict the output. The Random Forest
Classifier is a set of decision trees from a randomly selected subset of the training set.
It sums the votes from different decision trees to decide the final class of the test object.
Hence, it can train by providing input and output and obtaining a reasonably accurate
model, which can predict the output of new future input.

**Step 5: Predicting Output of Test data:** After the model has been trained with the
training set, the test data is given as input to the model and let it predict the output
(Figure 17). Then, a comparison is made between the predicted output generated by the
program with the labelled data already provided, determining the accuracy of the
model.
Step 6: Accuracy of the model: Using Sklearn, the accuracy of the model is inferred (Figure 18), which comes out as 86.24%, i.e. it will provide the correct output 86.24% of the time.

```
In [18]: ## Import library to check accuracy
from sklearn.metrics import accuracy_score

In [19]: score = accuracy_score(test['Label'], predictions)
print(score)
0.8624338624338624
```

Figure 18: Accuracy.

2.4 Sentiment Analysis of Tweets on COVID-19

It is necessary to distinguish between positive, negative, and neutral sentiments, or even retrieve scores associated with a given opinion based only on text. There are two main approaches to teach an algorithm to distinguish between positive and negative emotions in writing. 1. Supervised Learning. 2. Unsupervised Learning. Unlike the stock sentiment model, there is no label data available for the tweets on COVID-19. Hence the problem is approached using unsupervised sentiment analysis. The main idea...
behind unsupervised learning is not to give any previous assumptions and definitions to the model about the outcome of the variable fed into it. A pre-processed data is inserted for the model to learn the structure of the data itself. It is instrumental in cases to learn more about the nature of the process being analyzed, without making any previous assumptions about its outcome. The following flow chart shown in Figure 19 describes the approach taken to address this problem. Each process has been further elaborated.

![Flow Chart describing Approach.](image)

**Figure 19:** Flow Chart describing Approach.

The collection of data is firstly pre-processed through removing stop words (2.1.1), stemming (2.1.2), Lemmatization (2.1.3), removing stop words, and cleaning the text by removing punctuation to obtain clean data which contains only meaningful words. Then Word2vec algorithm is applied, which uses a neural network to learn the word associations and relations from a large corpus of text. TF-IDF is used to assign importance, and K-means clustering classifies the data into different groups based on their distance from the nearest mean. Then, each cluster is assigned a value of '1', '0', and '-1'. The steps are given hereunder.

**Step 1: Reading the data from the csv file:** The first step involves extracting the data from the csv files. This is done through the Pandas library (Figure 20). The CSV file used is 'covid19_tweets.csv', which is a file containing a large number of tweets and information about the tweets, including the username, the number of retweets or likes, etc.

```python
In [1]: import pandas as pd
   import numpy as np

In [3]: df = pd.read_csv("P:\datasets\covid19_tweets.csv")
```

![Reading the csv file.](image)

**Figure 20:** Reading the csv file.
Step 2: Pre-processing of Data: This step involves the cleaning of the text to reduce the volume of the text to a manageable amount, decreasing the amount of time taken for the model to run (Figure 21). Pre-processing of data has already been discussed under section 2.1.

```python
In [8]: import re
def clean_tweets(text):
    text = re.sub("RT\@[\w]*:","",text)
    text = re.sub("@[\w]*","",text)
    text = re.sub("https?://[A-Za-z0-9./]*","",text)
    text = re.sub("\n","",text)
    return text

In [9]: df["text"] = df["text"].apply(lambda x : clean_tweets(x))

Figure 21: Cleaning the data.
```

Step 3: Word2Vec: After the pre-processed data has been obtained, the Word2Vec algorithm is used to learn relations between different words from a large document containing text. Once trained, such a model can detect synonymous words or suggest different words for a partial sentence. As the name implies, word2vec represents each specific word with a particular list of numbers called a vector [11].

Step 4: TF-IDF: TF-IDF is used for assigning importance to the different words in the text based on the frequency of their occurrence in the text. TF-IDF (Term Frequency - Inverse Document Frequency) has already been explained in detail under section 2.2.3.

Step 5: K-means Clustering: K-means clustering is a method of vector quantization that groups up all the observations into various different clusters in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid), serving as a prototype of the cluster. Then each of these clusters is assigned a value. In the case of sentiment analysis, tweets can be positive, negative or neutral, and these can be represented by '1', '-1' or '0' respectively.

Vader_Lexicon: VADER (Valence Aware Dictionary and Sentiment Reasoner) is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media, and works well on texts from other domains. (Figure 22). VADER is used to analyze the intensity of the sentiments, which returns a polarity score of the tweets that can have any value as a decimal. The close the value is to ‘1’, the more positive the tweet is considered and vice versa.
3. RESULTS & DISCUSSIONS

3.1 Stock Sentiment Analysis

A model was developed using Python and successfully trained using supervised learning to predict the sentiment of news headlines of stocks with an accuracy of 86.24%. This highly accurate model has the following benefits.

1. Companies can predict which stock prices are going to increase, decrease, or stay the same and make their investments accordingly based on stock news.
2. This model can also be used to check the health of the company.
3. Companies can discover new marketing strategies and improve customer service as they can quickly identify the sentiments of consumers.

The accuracy of the model may be further increased by using other techniques.

3.2 Sentiment Analysis on COVID-19

3.2.1 Who is writing the tweet?

Since Twitter is free media for sharing messages, these tweets can originate from any entity including a person, an organization, a cardinal, a geopolitical entity or maybe some other sources. Finding who is writing tweets can be useful as it can have many significant implications. The coding to assign the entities in a list is given in Figures 23, coding to divide the origin of tweets is given in Figure 24 and to represent the information in the form of a pie chart, coding is given in Figure 25.

The Python Library ‘spacy’, which is an Advanced Natural Language Processing Library in Python is explicitly designed to build applications that process and ‘understand’ large volumes of text. It can be used to process text for deep or information extraction or learning to build natural language understanding systems. In Figure 24, five different entities have been created, namely, organization, person, geopolitical entity, norp (nationality or religious and political groups), and cardinal. The length of the lists of each group, i.e. the number of tweets made by each entity has been calculated. The ‘matplotlib.pyplot’ is used to obtain a pie chart given in Figure 26.
```python
In [16]:
import spacy
nlp = spacy.load('en_core_web_sm')

In [17]:
def['text'].apply(lambda x: [print("tText : ", Entity : {}".format(ent.text, ent.label_)) if not ent.text.startswith('#')
    else """ for ent in nlp(x).ents])
...

In [18]:
def['entities'].apply(lambda x: [(ent.text, ent.label_) if not ent.text.startswith('#')
    else """ for ent in nlp(x).ents])

In [19]:
etentities_graph = []
    for i in df['entities']:
        for j in i:
            entities_graph.append(j)

Figure 23: Assigning entities in a list.

In [21]:
organisation = []
person = []
geopolitical_entity = []
norp = []
cardinal = []
for i in entities_graph:
    for j in i:
        if j=="ORG":
            organisation.append(j)
        elif j=="PERSON":
            person.append(j)
        elif j=="GPE":
geopolitical_entity.append(j)
        elif j=="NORP":
norp.append(j)
        elif j=="CARDINAL":
cardinal.append(j)
else:
    pass

In [22]:
organisation = len(organisation)
person = len(person)
geopolitical_entity = len(geopolitical_entity)
norp = len(norp)
cardinal = len(cardinal)
classes = ["organisation", "person", "geopolitical_entity", "norp", "cardinal"]

Figure 24: Dividing the origin of tweets.

In [23]:
import matplotlib.pyplot as plt

In [24]:
graph = [organisation, person, geopolitical_entity, norp, cardinal]

In [25]:
explode = [0,0,2,0,0]
colors = ['c','b','r','g','y']
textprops = {'fontsize':15}
wedgeprops = {'linewidth': 4, 'width': 1, 'edgecolor': "k"}
plt.figure(figsize=(10,9))
fontdict = {'fontsize':22}
plt.title("Analysis of Entities", fontdict=fontdict)
plt.pie(graph, labels = classes, explode = explode, explode = explode, colors = colors, autopct="%.2f%%",radius = 1,textprops = textprops,
        pctdistance = 0.6,labeldistance = 1.1, wedgeprops = wedgeprops , rotatelabels=False )
plt.legend(loc = 2)
plt.savefig("entities_piechart.png", dpi = 1000, quality = 99)
plt.show

Figure 25: Plotting Pie chart for analysis of entities
Upon running the following code given in Figure 25, analysis of entities are given in Figure 26. The source of the tweets have been categorized into five classes, and their percentage share is represented in Figure 26. Maximum tweets have originated from 'organizations'.

![Figure 26: Analysis of entities in a chart.](image)

### 3.2.2 Sentiment of Tweets

The tweets having a positive, negative, or neutral sentiment have already been determined. Here, the coding is done to display this information pictorially using `explode` library, to make a pie chart to display this data (Figures 27 and 28). The classification of the tweets in the three classes is 43.21% neutral, 38.79% negative, and 18% positive.

```python
In [30]: a = []
for i in df["sentiment"][i].items():
    if value==0.5:
        a.append(key)

In [32]: neutral = a.count("neutral")
negative = a.count("negative")

In [36]: explode = [0,0.1,0]
colors = ["#e74c3c", "#3498db", "#f1c40f", "#2ecc71", "#e74c3c"]

plt.pie(a, explode=explode, colors=colors, autopct="%.2f%%", radius=1,
        wedgeprops=dict(width=0.1, edgecolor="k"))
plt.title("Sentiment analysis of tweets", fontsize=25)
plt.legend(loc=1)
plt.show()
```

![Figure 27: Code for Plotting Sentiment of Tweets.](image)
3.2.3 Sources of Tweets:
In order to obtain information about which device the users are writing their tweets from, codes shown in Figures 29 and 30 has been written. The source used to post tweets have been categorized among four categories, represented in Figure 30. The web platform is the most used source, followed by Android, iPhone, and Tweetdeck.

The sentiments of tweets on COVID-19 using unsupervised learning has been done, which is also able to analyze the intensity of the sentiments of the tweets based on how positive, negative, or neutral they are. This data can have many benefits, such as:

1. Based on which entities (person, company, country, or an organization) the tweets come from, it may be identified that which entities have a problem with the current government of a country by analyzing the sentiments of these entities regarding the handling of COVID-19.
2. By using this data, organizations like WHO (World Health Organization) can come in picture with a helping hand for these entities and also get some help about publishing articles regarding these entities for a country.

3. By analyzing this data, the locations of the users or the organizations who are writing the tweets can be determined. The countries or regions may be analyzed based on the positive or negative sentiments on COVID-19.

![Figure 30: Sources of Tweets.](image)

The information and data processing is the fuel for artificial intelligence. Based on the data analysis, decisions and policies may be framed by governments, agencies, organizations, and companies. The implementation of the coding samples and the analysis of the entities presented in this work is versatile and can be used in diversified applications.

4. **CONCLUSION**

A model has been successfully built with an accuracy of 86.24% for analyzing sentiments of headlines of stocks to predict the increase or decrease in stocks. Demonstration of the model has been successfully done by analyzing the tweets on COVID-19 to gather information, which can have many beneficial uses for companies, businesses, organizations, and even countries. The outcomes of the study have opened new avenues for data analysis using Python, which can fuel the artificial intelligence algorithms.

**Author Contributions:** Conceptualization, methodology, validation, formal analysis, investigation, data curation, Yuvraj Jain; Mentorship, formatting, writing and drafting, review, Vineet Tirth.

**Acknowledgements:** The first author thankfully acknowledges the guidance, motivation and support provided by the teachers of K.R Mangalam World School, New Delhi, India.

**Conflicts of Interest:** The authors declare no conflict of interest.
REFERENCES


