Importing excel sheet(URL)

In [4]:

```
import pandas as pd
# Load URLs from input.xlsx file
input_df = pd.read_excel(r"C:\Users\ajith\Downloads\Input.xlsx")
urls = input_df['URL'].tolist()
```

Extracting article text from URL

In [ ]:

```
pip install readability-lxml


#Readability is a library that can be used for text extraction.
```

In [128]:

```
import requests
from readability import Document
import re

list_of_article_text=[]

for i, url in enumerate(urls):
    # Send a GET request to the URL
    response = requests.get(url)

    # Parse the HTML content using Readability
    doc = Document(response.text)

    # Find the article title and text
    title = doc.title()
    article_text = doc.summary()

    # Remove any HTML tags from the article text
    clean_article_text = re.sub('<.*?>', '', article_text)

    # Remove unwanted notations
    clean_article_text = clean_article_text.replace('\n', '').replace('\r', '').replace('\t', '')

    list_of_article_text.append(title + clean_article_text)

    # Save the article content as a text file with a number as filename
    filename = f"{i+37}.txt"
    with open(f"C:\\Users\\ajith\\OneDrive\\Desktop\\New folder (2)\\{filename}", 'w', encoding='utf-8') as file:
        file.write(title + '\n\n' + clean_article_text)

    print(f'Saved article from {url} as {filename}')
```

```
Saved article from https://insights.blackcoffer.com/ai-in-healthcare-to-improve-patient-outcomes/ (https://i
nsights.blackcoffer.com/ai-in-healthcare-to-improve-patient-outcomes/) as 37.txt
Saved article from https://insights.blackcoffer.com/what-if-the-creation-is-taking-over-the-creator/ (http
s://insights.blackcoffer.com/what-if-the-creation-is-taking-over-the-creator/) as 38.txt
Saved article from https://insights.blackcoffer.com/what-jobs-will-robots-take-from-humans-in-the-future/ (h
ttps://insights.blackcoffer.com/what-jobs-will-robots-take-from-humans-in-the-future/) as 39.txt
Saved article from https://insights.blackcoffer.com/will-machine-replace-the-human-in-the-future-of-work/ (h
ttps://insights.blackcoffer.com/will-machine-replace-the-human-in-the-future-of-work/) as 40.txt
Saved article from https://insights.blackcoffer.com/will-ai-replace-us-or-work-with-us/ (https://insights.bl
ackcoffer.com/will-ai-replace-us-or-work-with-us/) as 41.txt
Saved article from https://insights.blackcoffer.com/man-and-machines-together-machines-are-more-diligent-tha
n-humans-blackcoffe/ (https://insights.blackcoffer.com/man-and-machines-together-machines-are-more-diligent-
than-humans-blackcoffe/) as 42.txt
Saved article from https://insights.blackcoffer.com/in-future-or-in-upcoming-years-humans-and-machines-are-g
oing-to-work-together-in-every-field-of-work/ (https://insights.blackcoffer.com/in-future-or-in-upcoming-yea
rs-humans-and-machines-are-going-to-work-together-in-every-field-of-work/) as 43.txt
Saved article from https://insights.blackcoffer.com/how-neural-networks-can-be-applied-in-various-areas-in-t
he-future/ (https://insights.blackcoffer.com/how-neural-networks-can-be-applied-in-various-areas-in-the-futu
re/) as 44.txt
```

In [131]:

```
list_of_article_text  # This list is in the format ~ [Article-1 text ,Article-2 text ,.......]
```

Out[131]:

```
['AI in healthcare to Improve Patient Outcomes | Blackcoffer InsightsIntroduction“If anything kills over 10
million people in the next few decades, it will be a highly infectious virus rather than a war. Not missiles
but microbes.” Bill Gates’s remarks at a TED conference in 2014, right after the world had avoided the Ebola
outbreak. When the new, unprecedented, invisible virus hit us, it met an overwhelmed and unprepared healthca
re system and oblivious population. This public health emergency demonstrated our lack of scientific conside
ration and underlined the alarming need for robust innovations in our health and medical facilities. For the
past few years, artificial intelligence has proven to be of tangible potential in the healthcare sectors, cl
inical practices, translational medical and biomedical research.After the first case was detected in China o
n December 31st 2019, it was an AI program developed by BlueDot that alerted the world about the pandemic. I
t was quick to realise AI’s ability to analyse large chunks of data could help in detecting patterns and ide
ntifying and tracking the possible carriers of the virus.Many tracing apps use AI to keep tabs on the people
who have been infected and prevent the risk of cross-infection by using AI algorithms that can track pattern
s and extract some features to classify or categorise them.So how does AI do that?IBM Watson, a sophisticate
d AI that works on cloud computing and natural language processing, has prominently contributed to the healt
hcare sector on a global level. Being a conversational AI, since 2013, Watson has helped in recommending tre
atments to patients suffering from cancer to ensure that they get the best treatment at optimum costs.Resear
chers at Google Inc. showed that an AI system can be trained on thousands of images to achieve physician-lev
el sensitivity.By identifying the molecular patterns associated with disease status and its subtypes, gene e
```

In [350]:

```
article_list = list_of_article_text.copy()  #making copy of the list
```

Removing unwanted notations and punctuations

In [351]:

```
for i in range(len(article_list)):
    # Perform multiple text replacements in the current document
    article_list[i] = article_list[i].replace("\n", " ") \
                .replace("\xa0", " ") \
                .replace("“"," ")\
                .replace("”", " ")\
                .replace("’", " ")\
                .replace("‘", " ")
```

Tokenization

In [353]:

```
import string
from nltk.tokenize import word_tokenize

# Punctuation to remove
punctuations = string.punctuation

# List to store the tokenized and normalized sentences
tokenized_list = []

# Loop through each sentence in the list and remove punctuation, then tokenize and normalize
for sentence in article_list:
    sentence = sentence.translate(str.maketrans("", "", string.punctuation))
    tokens = word_tokenize(sentence) # Tokenize
    normalized_tokens = [token.lower() for token in tokens]
    tokenized_list.append(normalized_tokens)

print(tokenized_list)  # This list is in the format ~ [[tokenized words from Article-1],[tokenized words from Article-2],....
```

...

Importing stop words

In [354]:

```
# Create an empty list to store the words
stop_words = []

# List of file names to read
files = [r"C:\Users\ajith\OneDrive\Desktop\stop words\StopWords_Auditor.txt",
         r"C:\Users\ajith\OneDrive\Desktop\stop words\StopWords_Currencies.txt",
         r"C:\Users\ajith\OneDrive\Desktop\stop words\StopWords_DatesandNumbers.txt",
         r"C:\Users\ajith\OneDrive\Desktop\stop words\StopWords_Generic.txt",
         r"C:\Users\ajith\OneDrive\Desktop\stop words\StopWords_GenericLong.txt",
         r"C:\Users\ajith\OneDrive\Desktop\stop words\StopWords_Geographic.txt",
         r"C:\Users\ajith\OneDrive\Desktop\stop words\StopWords_Names.txt"]

for file_name in files:
    with open(file_name, 'r') as file:
        file_words = file.read().split()
        stop_words.extend(file_words)

stop_words = [word.lower() for word in stop_words]  #changing words into lower case

print(stop_words)
```

...

Removing stop words from tokenized list

In [355]:

```
filtered_list = [[word for word in sentence if word not in stop_words] for sentence in tokenized_list]

print(filtered_list)
```

```
[['healthcare', 'improve', 'patient', 'outcomes', 'blackcoffer', 'insightsintroduction', 'kills', '10', 'peo
ple', 'decades', 'highly', 'infectious', 'virus', 'war', 'missiles', 'microbes', 'remarks', 'conference', '2
014', 'world', 'avoided', 'ebola', 'outbreak', 'unprecedented', 'invisible', 'virus', 'hit', 'met', 'overwhe
lmed', 'unprepared', 'healthcare', 'system', 'oblivious', 'population', 'public', 'health', 'emergency', 'de
monstrated', 'lack', 'scientific', 'consideration', 'underlined', 'alarming', 'robust', 'innovations', 'heal
th', 'medical', 'facilities', 'past', 'years', 'artificial', 'intelligence', 'proven', 'tangible', 'potentia
l', 'healthcare', 'sectors', 'clinical', 'practices', 'translational', 'medical', 'biomedical', 'researchaft
er', 'detected', '31st', '2019', 'program', 'developed', 'bluedot', 'alerted', 'world', 'pandemic', 'realis
e', 'ability', 'analyse', 'chunks', 'data', 'detecting', 'patterns', 'identifying', 'tracking', 'carriers',
'virusmany', 'tracing', 'apps', 'tabs', 'people', 'infected', 'prevent', 'risk', 'crossinfection', 'algorith
ms', 'track', 'patterns', 'extract', 'features', 'classify', 'categorise', 'themso', 'thatibm', 'sophisticat
ed', 'works', 'computing', 'natural', 'language', 'processing', 'prominently', 'contributed', 'healthcare',
'sector', 'global', 'level', 'conversational', '2013', 'helped', 'recommending', 'treatments', 'patients',
'suffering', 'cancer', 'ensure', 'treatment', 'optimum', 'costsresearchers', 'google', 'showed', 'system',
'trained', 'thousands', 'images', 'achieve', 'physicianlevel', 'sensitivityby', 'identifying', 'molecular',
'patterns', 'disease', 'status', 'subtypes', 'expression', 'protein', 'abundance', 'levels', 'machine', 'lea
rning', 'methods', 'detect', 'fatal', 'diseases', 'cancer', 'stage', 'machine', 'learning', 'ml', 'technique
s', 'focus', 'analyzing', 'structured', 'data', 'clustering', 'patients', 'traits', 'infer', 'probability',
'disease', 'outcomes', 'patient', 'traits', 'include', 'masses', 'data', 'relating', 'age', 'gender', 'disea
```

In [356]:

```
num_of_sentence = []

for article in article_list:
    # Count the number of sentences
    sentences = nltk.sent_tokenize(article)
    num_sentences = len(sentences)
    num_of_sentence.append(num_sentences)

print(num_of_sentence)  # This list is in the format ~ [Number of sentence in Article-1,Num pf sentence in Art-2,....]
```

...

WORD COUNT

In [357]:

```
word_count=[]

for i, filtered_tokens_list in enumerate(filtered_list):
    word_countt = len(filtered_tokens_list)
    word_count.append(word_countt)
print(word_count)
```

...

AVG NUMBER OF WORDS PER SENTENCE

In [358]:

```
avg_words_per_sentence= []

for word_countt,num_sentences in zip(word_count,num_of_sentence):
    # Calculate average words per sentence
    x = word_countt / num_sentences

    avg_words_per_sentence.append(x)

print(avg_words_per_sentence)
```

...

AVG WORD LENGTH

In [359]:

```
avg_word_length=[]

for article in filtered_list:
    total_chars = 0
    total_words = len(article)
    for word in article:
        total_chars += len(word)
    # Calculate average words length
    avg_word_len = total_chars/total_words

    avg_word_length.append(avg_word_len)

print(avg_word_length)
```

...

COMPLEX WORD COUNT

In [360]:

```
!pip install syllables
```

...

In [361]:

```
import syllables

complex_word_counts = []

def count_syllables(word):
    return syllables.estimate(word)

for article in filtered_list:
    complex_word_count = 0
    for word in article:
        num_syllables = count_syllables(word)
        if num_syllables > 2:
            complex_word_count += 1
    complex_word_counts.append(complex_word_count)

print(complex_word_counts)
```

...

SYLLABLE PER WORD

In [362]:

```
import syllables

syllables_per_word = []

for article in filtered_list:
    total_syllables = 0
    total_words = len(article)
    for word in article:
        syllable_count = syllables.estimate(word)
        total_syllables += syllable_count
        avg_syllables_per_word = total_syllables / total_words
    syllables_per_word.append(avg_syllables_per_word)

print(syllables_per_word)
```

...

PERSONAL PRONOUNS

In [363]:

```
import re

# Define the regex pattern for personal pronouns
pronoun_pattern = r'\b(i|we|my|ours|us)\b'

personal_pronouns_list = []

for article in filtered_list:
    personal_pronouns_count = 0
    for word in article:
        personal_pronouns_count += len(re.findall(pronoun_pattern, word, re.IGNORECASE))
    personal_pronouns_list.append(personal_pronouns_count)


print(personal_pronouns_list)
```

...

AVG SENTENCE LENGTH,PERCENTAGE OF COMPLEX WORDS,FOG INDEX

In [364]:

```
avg_sentence_len = []
pct_of_complex_words = []
fog_index = []

for i in range(len(word_count)):
    num_of_words = word_count[i]
    num_of_sentences = num_of_sentence[i]
    num_of_complex_words = complex_word_counts[i]

    avg_sentence_leng = num_of_words / num_of_sentences
    pct_complex_words = num_of_complex_words / num_of_words
    fog_indexx = 0.4 * (avg_sentence_leng + pct_complex_words)

    avg_sentence_len.append(avg_sentence_leng)
    pct_of_complex_words.append(pct_complex_words)
    fog_index.append(fog_indexx)
```

importing positive and negative words

In [365]:

```
positive_words = []
negative_words = []

with open(r"C:\Users\ajith\OneDrive\Desktop\pos,neg dict\positive-words.txt", 'r') as file:
    # Read the contents of the file and split it into words
    file_words = file.read().split()
    # Append the words to the list
    positive_words.extend(file_words)

with open(r"C:\Users\ajith\OneDrive\Desktop\pos,neg dict\negative-words.txt", 'r') as file:
    # Read the contents of the file and split it into words
    file_words = file.read().split()
    # Append the words to the list
    negative_words.extend(file_words)

positive_words = [word.lower() for word in positive_words]  #changing into lower case
negative_words = [word.lower() for word in negative_words]  #changing into lower case
```

POSITIVE,NEGATIVE,POLARITY AND SUBJECTIVITY SCORES

In [366]:

```
positive_scores = []
negative_scores = []
polarity_scores = []
subjectivity_scores = []

for i, article in enumerate(filtered_list):
    # Calculate positive score
    pos_score = sum([1 if word in positive_words else 0 for word in article])
    positive_scores.append(pos_score)

    # Calculate negative score
    neg_score = sum([1 if word in negative_words else 0 for word in article])
    negative_scores.append(neg_score)

    # Calculate polarity score
    pol_score = (pos_score - neg_score) / ((pos_score + neg_score) + 0.000001)
    polarity_scores.append(pol_score)

    # Calculate subjectivity score
    subj_score = (pos_score + neg_score) / (word_count[i] + 0.000001)
    subjectivity_scores.append(subj_score)
```

In [367]:

```
#posite_score_for_each_article
positive_scores                    #[score of Article-1, Article-2 ,.....]
```

Out[367]:

```
[63,
 55,
 64,
 63,
 56,
 42,
 21,
 0,
 31,
 62,
 47,
 31,
 28,
 52,
 63,
 23,
 75,
 19.
```

In [368]:

```
#negative_score_for_each_article
negative_scores
```

Out[368]:

```
[30,
 35,
 34,
 23,
 25,
 22,
 11,
 0,
 13,
 32,
 53,
 20,
 29,
 23,
 19,
 0,
 39,
 0.
```

In [369]:

```
#polarity_score_for_each_article
polarity_scores
```

Out[369]:

```
[0.35483870586194943,
 0.22222221975308645,
 0.30612244585589343,
 0.4651162736614387,
 0.3827160446578266,
 0.31249999511718757,
 0.3124999902343753,
 0.0,
 0.40909089979338864,
 0.3191489327750114,
 -0.05999999940000001,
 0.21568627028066137,
 -0.017543859341335802,
 0.3866666615111112,
 0.5365853593099347,
 0.999999956521741,
 0.31578947091412746,
 0.9999999473684238.
```

In [370]:

```
#subjectivity_score_for_each_article
subjectivity_scores
```

Out[370]:

```
[0.09810126571930246,
 0.15929203511629728,
 0.1203931202452173,
 0.13479623803323473,
 0.10519480505818857,
 0.11469534029624491,
 0.09609609580751924,
 0.0,
 0.1370716506633282,
 0.10363836813270301,
 0.10857763288970941,
 0.08947368405355494,
 0.0979381441616183,
 0.11520737309491955,
 0.11420612797464327,
 0.09236947754068484,
 0.16497829209120363,
 0.04418604640886966,
```

In [373]:

```
#average_sentence_length_in_each_article
avg_sentence_len
```

Out[373]:

```
[17.555555555555557,
 8.18840579710145,
 11.970588235294118,
 7.975,
 10.266666666666667,
 10.941176470588236,
 7.744186046511628,
 5.0,
 10.35483870967742,
 13.144927536231885,
 9.797872340425531,
 13.902439024390244,
 8.558823529411764,
 18.6,
 13.807692307692308,
 13.833333333333334,
 86.375,
 13.030303030303031,
```

In [372]:

```
#percentage_of_complex_words_for_each_article
pct_of_complex_words
```

Out[372]:

```
[0.5390295358649789,
 0.4353982300884956,
 0.5577395577395577,
 0.45297805642633227,
 0.4714285714285714,
 0.478494623655914,
 0.4954954954954955,
 0.4,
 0.40809968847352024,
 0.48732083792723263,
 0.48859934853420195,
 0.5807017543859649,
 0.44329896907216493,
 0.5253456221198156,
 0.48467966573816157,
 0.5783132530120482,
 0.4558610709117221,
 0.5465116279069767,
```

In [371]:

```
#fog_index_for_each_article
fog_index
```

Out[371]:

```
[7.2378340365682154,
 3.449521610875978,
 5.01133111721347,
 3.3711912225705327,
 4.295238095238096,
 4.56786843769766,
 3.29587261680285,
 2.16,
 4.305175359260376,
 5.452899349663647,
 4.114588675583893,
 5.793256311510484,
 3.600848999393572,
 7.650138248847927,
 5.716948789372188,
 5.764658634538153,
 34.73234442836469,
 5.430725863284003,
```

In [378]:

```
#average_words_per_sentence_in_each_article
avg_words_per_sentence
```

```
 11.970588235294118,
 7.975,
 10.266666666666667,
 10.941176470588236,
 7.744186046511628,
 5.0,
 10.35483870967742,
 13.144927536231885,
 9.797872340425531,
 13.902439024390244,
 8.558823529411764,
 18.6,
 13.807692307692308,
 13.833333333333334,
 86.375,
 13.030303030303031,
 16.61904761904762,
 72.5,
 5.0,
 15.261904761904763,
```

In [376]:

```
#comples_words_counts_in_each_article
complex_word_counts
```

Out[376]:

```
[511,
 246,
 454,
 289,
 363,
 267,
 165,
 2,
 131,
 442,
 450,
 331,
 258,
 342,
 348,
 144,
 315,
 235,
```

In [379]:

```
#word_count_in_each_article
word_count
```

Out[379]:

```
[948,
 565,
 814,
 638,
 770,
 558,
 333,
 5,
 321,
 907,
 921,
 570,
 582,
 651,
 718,
 249,
 691,
 430
```

In [375]:

```
#syllables_per_word_in_each_article
syllables_per_word
```

```
 2.7116279069767444,
 2.8739255014326646,
 2.986206896551724,
 2.2,
 2.497659906396256,
 2.4841772151898733,
 2.596281540504648,
 2.6635220125786163,
 2.2722513089005236,
 2.392523364485981,
 2.8863636363636362,
 2.7733333333333334,
 2.793103448275862,
 2.282674772036474,
 2.433734939759036,
 2.8631051752921537,
 2.409691629955947,
 2.4044444444444446,
 2.6762860727728985,
```

In [374]:

```
#personal_pronouns_in_each_article
personal_pronouns_list
```

Out[374]:

```
[0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 1,
 0,
 0,
 0,
 0,
 0,
 0,
 0
```

In [377]:

```
#average_word_length_in_each_article
avg_word_length
```

Out[377]:

```
[7.910337552742616,
 7.208849557522124,
 7.8452088452088455,
 7.266457680250784,
 7.4,
 7.469534050179211,
 7.213213213213213,
 8.0,
 6.953271028037383,
 7.396912899669239,
 7.309446254071661,
 8.189473684210526,
 7.123711340206185,
 7.743471582181259,
 7.626740947075209,
 7.983935742971887,
 7.164978292329956,
 7.8,
```