```
pip install textstat
```

```
Collecting textstat
  Downloading textstat-0.7.3-py3-none-any.whl (105 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 105.1/105.1 kB 1.2 MB/s eta 0:00:00
Collecting pyphen (from textstat)
  Downloading pyphen-0.15.0-py3-none-any.whl (2.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 8.6 MB/s eta 0:00:00
Installing collected packages: pyphen, textstat
Successfully installed pyphen-0.15.0 textstat-0.7.3
```

```
import os
import chardet
# import openpyxl
import pandas as pd
import numpy as np
from bs4 import BeautifulSoup
import requests
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import nltk
from nltk.sentiment import SentimentIntensityAnalyzer
from textstat import flesch_reading_ease, gunning_fog
from nltk.tokenize import sent_tokenize, word_tokenize
# import syllable_count
```

Mounting Google Drive

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
# pip install syllable-count
```

Download required resources for NLTK

```
# Download required resources for NLTK
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('cmudict')
nltk.download('vader_lexicon')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
[nltk_data] Downloading package cmudict to /root/nltk_data...
[nltk_data]   Unzipping corpora/cmudict.zip.
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
True
```

```
os.makedirs("extracted_texts", exist_ok=True)
```

Loading input data

```
input_data=pd.read_excel("/content/drive/MyDrive/Task/Input.xlsx")
input_data
```

	URL_ID	URL
0	blackassign0001	https://insights.blackcoffer.com/rising-it-cit...
1	blackassign0002	https://insights.blackcoffer.com/rising-it-cit...
2	blackassign0003	https://insights.blackcoffer.com/internet-dema...
3	blackassign0004	https://insights.blackcoffer.com/rise-of-cyber...
4	blackassign0005	https://insights.blackcoffer.com/ott-platform-...
...	...	...
95	blackassign0096	https://insights.blackcoffer.com/what-is-the-r...
96	blackassign0097	https://insights.blackcoffer.com/impact-of-cov...
97	blackassign0098	https://insights.blackcoffer.com/contribution-...
98	blackassign0099	https://insights.blackcoffer.com/how-covid-19-...
99	blackassign0100	https://insights.blackcoffer.com/how-will-covi...

100 rows × 2 columns

Next steps: Generate code with input_data View recommended plots

Function to extract article text from url

```
def url_text_extraction(url, encoding='ISO-8859-1'):
    response = requests.get(url)

    if response.status_code == 200:
        try:
            # Attempt to detect encoding if not specified
            if not encoding:
                encoding = chardet.detect(response.content)['encoding']

            # Decode with the specified or detected encoding
            return response.content.decode(encoding)
        except UnicodeDecodeError:
            print(f"Failed to decode content from URL {url} using encoding {encoding}.")
            return None
    else:
        print(f"Failed to fetch content from URL: {url}")
        return None
```

Function to read negative words file

```
with open('/content/drive/MyDrive/Task/negative-words.txt', 'rb') as f:
    rawdata = f.read()
    result = chardet.detect(rawdata)
    print(result['encoding'])

def read_negative_words(file_path, encoding='ISO-8859-1'):  # Replace with actual encoding
    with open(file_path, 'r', encoding=encoding) as file:
        return set(file.read().split())
```

```
ISO-8859-1
```

Function to read positive words file

```
with open('/content/drive/MyDrive/Task/positive-words.txt', 'rb') as f:
    rawdata = f.read()
    result = chardet.detect(rawdata)
    print(result['encoding'])

def read_positive_words(file_path, encoding='ascii'):
    with open(file_path, 'r', encoding=encoding) as file:
        return set(file.read().split())
```

```
ascii
```

Function to read stop words from a combined stop word file

```
with open('/content/drive/MyDrive/Task/Combined_stopwords.txt', 'rb') as f:
    rawdata = f.read()
    result = chardet.detect(rawdata)
    print(result['encoding'])

def read_stop_words(file_path, encoding='utf-8'):  # Use a valid encoding
    with open(file_path, 'r', encoding=encoding) as file:
        return set(file.read().split())
```

```
utf-8
```

```
def save_extracted_text(url_id, url, article_text):
    file_path = os.path.join("extracted_texts", f"{url_id}.txt")
    with open(file_path, "w", encoding="utf-8") as file:
        file.write(article_text)
```

```
def analyze_text(text, positive_words_path, negative_words_path, stop_words_path, excel_file_path):
    # Tokenize the text
    sentences = sent_tokenize(text)
    words = word_tokenize(text)

    # Calculate various metrics
    avg_sentence_length = len(words) / len(sentences)

    # Sentiment Analysis
    sid = SentimentIntensityAnalyzer()
    sentiment_scores = sid.polarity_scores(text)

    # Cleaning using Stop Words Lists
    stop_words = read_stop_words(stop_words_path)
    cleaned_words = [word.lower() for word in words if word.isalpha() and word.lower() not in stop_words]

    # Creating dictionary of Positive and Negative words
    positive_words = read_positive_words(positive_words_path)
    negative_words = read_negative_words(negative_words_path)

    positive_count = sum(1 for word in cleaned_words if word in positive_words)
    negative_count = sum(1 for word in cleaned_words if word in negative_words)

    # Extracting Derived variables
    polarity_score = sentiment_scores['compound']

    # Calculate subjectivity score based on the proportion of subjective words
    subjective_words = [word for word, pos in nltk.pos_tag(cleaned_words) if pos in ['JJ', 'JJR', 'JJS', 'RB', 'RBR', 'RBS']]
    subjectivity_score = len(subjective_words) / len(cleaned_words)

    # # Additional Variables

    word_count = len(cleaned_words)

    # Define syllable_count function before using it
    def syllable_count(word):
      vowels = "aeiouAEIOU"
      count = 0
      for char in word:
          if char in vowels:
              count += 1
      # Handle trailing e exceptions
      if word.endswith(('e', 'es', 'ed')) and not word.endswith(('le', 'les', 'les')):
          count -= 1
      # Handle double vowels as a single syllable
      if count > 1:
          for i in range(1, len(word) - 1):
              if word[i] in vowels and word[i - 1] in vowels:
                  count -= 1
      return count

    complex_word_count = sum(1 for word in cleaned_words if len(word) > 2 and syllable_count(word) > 2)

    syllable_per_word = sum(syllable_count(word) for word in cleaned_words) / len(cleaned_words)

    percentage_of_complex_words = complex_word_count / word_count * 100

    fog_index = 0.4 * (avg_sentence_length + percentage_of_complex_words)  # Using Gunning Fog Index formula

    personal_pronoun_list = ['I', 'me', 'my', 'mine', 'myself', 'you', 'your', 'yours', 'yourself', 'he', 'him', 'his', 'himself', 'she
    personal_pronouns = sum(1 for word in cleaned_words if word.lower() in personal_pronoun_list)

    average_word_length = sum(len(word) for word in cleaned_words) / len(cleaned_words)

    # Additional variables:
    avg_words_per_sentence = avg_sentence_length  # Same as word count divided by sentence count

    return {
        'average_sentence_length': avg_sentence_length,
        'positive_sentiment_score': sentiment_scores['pos'],
        'negative_sentiment_score': sentiment_scores['neg'],
        'overall_polarity_score': polarity_score,
        'subjectivity_score': subjectivity_score,
        'complex_word_count': complex_word_count,
        'word_count': word_count,
        'syllables_per_word': syllable_per_word,
        'personal_pronouns_count': personal_pronouns,
        'average_word_length': average_word_length,
        'average_words_per_sentence': avg_words_per_sentence,
        'percentage_of_complex_words': percentage_of_complex_words,
        'fog_index': fog_index,
    }
```

```
# pip install xlsxwriter


output_data = []

for index, row in input_data.iterrows():
    url_id = row['URL_ID']
    url = row['URL']

    article_text = url_text_extraction(url)

    if article_text:
        analysis_results = analyze_text(article_text, '/content/drive/MyDrive/Task/positive-words.txt', '/content/drive/MyDrive/Task/ne

        # analysis_results['URL_ID'], analysis_results['URL'] = url_id, url

        # Save extracted text
        save_extracted_text(url_id, url, article_text)

        output_data.append(analysis_results)
    else:
        print(f"Failed to fetch content from URL: {url}")

print(analysis_results)
print(output_data)

# Create output Dataframe and save to Excel
output_df = pd.DataFrame(output_data)
output_df.to_excel('/content/drive/MyDrive/Copy of Output Data Structure.xlsx')

# # Creating output Dataframe and save to Input.xlsx Excel
# output_df = pd.DataFrame(output_data)
# output_df.to_excel("Input.xlsx")
```

```
Failed to fetch content from URL: https://insights.blackcoffer.com/how-neural-networks-can-be-applied-in-various-areas-in-the-futur
Failed to fetch content from URL: https://insights.blackcoffer.com/how-neural-networks-can-be-applied-in-various-areas-in-the-futur
Failed to fetch content from URL: https://insights.blackcoffer.com/covid-19-environmental-impact-for-the-future/
Failed to fetch content from URL: https://insights.blackcoffer.com/covid-19-environmental-impact-for-the-future/
{'average_sentence_length': 892.0106382978723, 'positive_sentiment_score': 0.026, 'negative_sentiment_score': 0.028, 'overall_polar
[{'average_sentence_length': 561.3089430894308, 'positive_sentiment_score': 0.027, 'negative_sentiment_score': 0.017, 'overall_pola
```

```
df=pd.read_excel("/content/drive/MyDrive/Task/Output Data Structure.xlsx")
df.head()
```

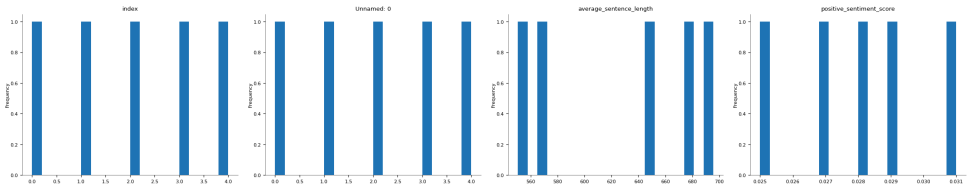
1 to 5 of 5 entries Filter

onal_pronouns_count	average_word_length	average_words_per_sentence	percentage_of_complex_words	fog_index	URL_ID	
0	5.729094076655052	572.0661157024794	20.64459930313589	237.0842860022461	blackassign0001	https://insi cities-and- environme by-the-yea
0	5.830335439961108	550.5396825396825	21.99805542051531	229.0150951840792	blackassign0002	https://insi cities-and- economy- and-city-lif
0	5.864163652238992	648.4245283018868	22.61444430585007	268.4155890430947	blackassign0003	https://insi demands- impact-an pathways/
0	5.838552713661884	680.4851485148515	22.02121023081722	281.0025434982675	blackassign0004	https://insi cybercrim future/
0	5.746312684365781	695.6326530612245	20.94395280235988	286.6306423454337	blackassign0005	https://insi platform-a entertainm

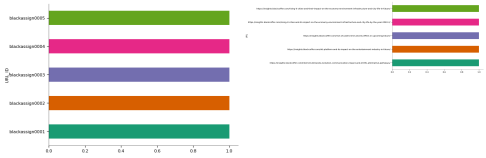
Show 25 per page

Like what you see? Visit the data table notebook to learn more about interactive tables.

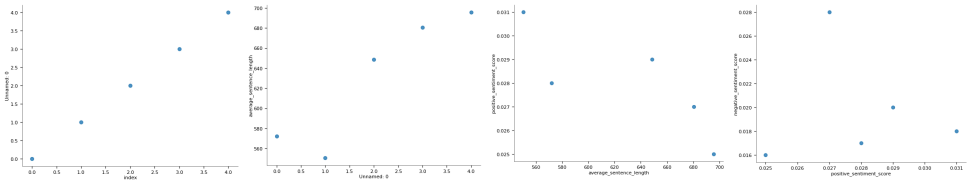
Distributions



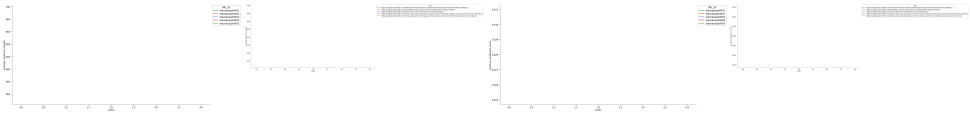
Categorical distributions



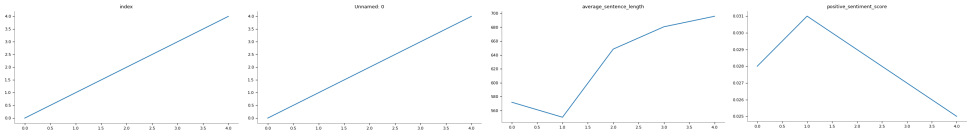
2-d distributions



Time series



Values



2-d categorical distributions