**Chapter 4 Examples Part 2 Updated**

The following examples focus on analyzing the data retrieved from Youtube from the older examples in this chapter.

**Example 8:** This example builds on top of Example 7 and demonstrates how to detect sarcasm from the caption of a Youtube video.

Original Tutorial:

<https://thecleverprogrammer.com/2021/08/24/sarcasm-detection-with-machine-learning/>

Printing a List without Quotes and Brackets

<https://www.codeleaks.io/print-list-without-brackets-and-quotes-in-python/>

Requirements:

pip3 install -U scikit-learn

pip3 install nltk

import pandas

import re

import numpy as np

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.model\_selection import train\_test\_split

from sklearn.naive\_bayes import BernoulliNB

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

from nltk.tokenize.treebank import TreebankWordDetokenizer

# This is the training data with already categorized messages

data = pandas.read\_json("Sarcasm\_Headlines\_Dataset.json", lines=True)

#print(data.head())

data["is\_sarcastic"] = data["is\_sarcastic"].map({0: "Not Sarcasm", 1: "Sarcasm"})

#print(data.head())

data = data[["headline", "is\_sarcastic"]]

x = np.array(data["headline"])

y = np.array(data["is\_sarcastic"])

cv = CountVectorizer()

X = cv.fit\_transform(x) # Fit the Data

#test\_size is the number that defines the size of the test set. It's very similar to train\_size .

#Random stats will decide the splitting of data into train and test indices

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20, random\_state=42)

model = BernoulliNB()

model.fit(X\_train, y\_train)

print(model.score(X\_test, y\_test))

#I am creating a dictionary here titled inputdata

inputdata={}

inputdata = pandas.read\_csv('example6results.csv').to\_dict()

# I created a new dictionary here for the text column in my csv file

caption\_text\_dictionary = inputdata.get('text')

# I am converting the caption dictionary to a list so I can analyze the data

caption\_text\_list = list(caption\_text\_dictionary.values())

#convert list to string

caption\_text\_instring = ''

for eachletter in caption\_text\_list:

caption\_text\_instring += eachletter

#I am cleaning the caption data

#1. make all letters lowercase

caption\_text\_instring = caption\_text\_instring.lower()

#2. Remove special characters

caption\_text\_instring = re.sub(r'\W+', ' ', caption\_text\_instring)

#3. Remove numbers

caption\_text\_instring\_no\_numbers = ''.join(c if c not in map(str,range(0,10)) else "" for c in caption\_text\_instring)

# 4. Remove stop words

#We need to convert the string to tokens in order to remove the stop words then convert the tokens back to string format

#A token is a string of contiguous characters between two spaces, or between a space and punctuation marks.

caption\_tokens = word\_tokenize(caption\_text\_instring\_no\_numbers)

cleaned\_caption\_tokens = [word for word in caption\_tokens if not word in stopwords.words()]

input\_data = TreebankWordDetokenizer().detokenize(cleaned\_caption\_tokens)

#Enter cleaned caption data to the model to check sarcasm

data = cv.transform([input\_data]).toarray()

output = model.predict(data)

print(output)

# This seperator helps me to print the result without brackets and quotes

separator = ", "

print(separator.join(output))

**Example 9:** This example demonstrates how to calculate the sarcasm for each Youtube Video comment and save the results into a CSV file.

import pandas

import re

import numpy as np

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.model\_selection import train\_test\_split

from sklearn.naive\_bayes import BernoulliNB

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

from nltk.tokenize.treebank import TreebankWordDetokenizer

# This is the training data with already categorized messages

data = pandas.read\_json("Sarcasm\_Headlines\_Dataset.json", lines=True)

#print(data.head())

data["is\_sarcastic"] = data["is\_sarcastic"].map({0: "Not Sarcasm", 1: "Sarcasm"})

#print(data.head())

data = data[["headline", "is\_sarcastic"]]

x = np.array(data["headline"])

y = np.array(data["is\_sarcastic"])

cv = CountVectorizer()

X = cv.fit\_transform(x) # Fit the Data

#test\_size is the number that defines the size of the test set. It's very similar to train\_size .

#Random stats will decide the splitting of data into train and test indices

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20, random\_state=42)

model = BernoulliNB()

model.fit(X\_train, y\_train)

print(model.score(X\_test, y\_test))

#I am creating a dictionary here titled inputdata

inputdata={}

inputdata = pandas.read\_csv('allvideocomments\_results.csv').to\_dict()

# I created a new dictionary here for the comment column in my csv file

comment\_dictionary = inputdata.get('Comment')

# I am converting the comment dictionary to a list so I can analyze the data

comment\_list = list(comment\_dictionary.values())

sarcasm\_results\_list=[]

for i in range(len(comment\_list)):

# I am cleaning the comments data

# 1. make all letters lowercase

comment\_list[i] = comment\_list[i].lower()

# 2. Remove special characters

comment\_list[i] = re.sub(r'\W+', ' ', comment\_list[i])

# 3. Remove numbers

comment\_list[i] = ''.join(c if c not in map(str, range(0, 10)) else "" for c in comment\_list[i])

# 4. Remove stop words

# We need to convert the string to tokens in order to remove the stop words then convert the tokens back to string format

# A token is a string of contiguous characters between two spaces, or between a space and punctuation marks.

comment\_tokens = word\_tokenize(comment\_list[i])

cleaned\_comment\_tokens = [word for word in comment\_tokens if not word in stopwords.words()]

#I am converying the tokens back into string as an input for the sarcasm detector

input\_data = TreebankWordDetokenizer().detokenize(cleaned\_comment\_tokens)

data = cv.transform([comment\_list[i]]).toarray()

output = model.predict(data)

#print(output)

#This seperator helps me to print the results without brackets and quotes

separator = ", "

sarcasm\_result = {"Is\_Sarcastic":separator.join(output) }

sarcasm\_results\_list.append(sarcasm\_result)

#This is the Sarcasm Analysis Results in Dataframe

sarcasm\_results\_dataframe = pandas.DataFrame(sarcasm\_results\_list)

file = pandas.read\_csv('allvideocomments\_results.csv')

file['Is\_Sarcastic'] = sarcasm\_results\_dataframe

file.to\_csv('example9\_results.csv', index=True)

**Example 10:** This example demonstrates how to calculate the readability or lexical complexity of a Youtube video’s comments and save the results to a CSV file.

Requirement: pip3 install textstat

For the interpretation of the results see the score table in:

<https://pypi.org/project/textstat/>

A negative readability score suggests that comment is very short.

<https://help.siteimprove.com/support/solutions/articles/80000447982-readability-why-are-my-scores-falling-out-of-the-scorecard-range-#:~:text=With%20Flesch%2DKincaid%20if%20the,readability%20score%20can%20be%20negative.&text=In%20the%20example%20above%2C%20we,in%20the%20score%20of%20%2D1.43>.

import pandas

import re

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

from nltk.tokenize.treebank import TreebankWordDetokenizer

import textstat

#I am creating a dictionary here titled inputdata

inputdata={}

inputdata = pandas.read\_csv('example9\_results.csv').to\_dict()

# I created a new dictionary here for the comment column in my csv file

comment\_dictionary = inputdata.get('Comment')

# I am converting the comment dictionary to a list so I can analyze the data

comment\_list = list(comment\_dictionary.values())

readability\_results\_list=[]

for i in range(len(comment\_list)):

# I am cleaning the comments data

# 1. make all letters lowercase

comment\_list[i] = comment\_list[i].lower()

# 2. Remove special characters

comment\_list[i] = re.sub(r'\W+', ' ', comment\_list[i])

# 3. Remove numbers

comment\_list[i] = ''.join(c if c not in map(str, range(0, 10)) else "" for c in comment\_list[i])

# 4. Remove stop words

# We need to convert the string to tokens in order to remove the stop words then convert the tokens back to string format

# A token is a string of contiguous characters between two spaces, or between a space and punctuation marks.

comment\_tokens = word\_tokenize(comment\_list[i])

cleaned\_comment\_tokens = [word for word in comment\_tokens if not word in stopwords.words()]

# I am converying the tokens back into string as an input for the readability detector

input\_data = TreebankWordDetokenizer().detokenize(cleaned\_comment\_tokens)

readability\_result = textstat.flesch\_reading\_ease(input\_data)

readability\_results\_list.append(readability\_result)

#Test the results

#print(readability)

#Save the results to a csv file

#This is the Readability Analysis Results in Dataframe

readability\_results\_dataframe = pandas.DataFrame(readability\_results\_list)

file = pandas.read\_csv('example9\_results.csv')

file['Readability'] = readability\_results\_dataframe

file.to\_csv('example10\_results.csv', index=True)

**Example 11:** This example demonstrates how to categorize the comments via hierarchical cluster analysis (ward method). This method attempts to reduce variance within each cluster.

Tutorial for coding: <https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.linkage.html>

Tutorial for interpreting the diagram:

<https://www.analyticsvidhya.com/blog/2021/08/hierarchical-clustering-algorithm-python/>

**Interpreting the diagram:** Readability scores are represented on the x-axis, and the Euclidean distance between clusters is represented on the y-axis. How do we figure out the best number of clusters based on this diagram? We want to find the longest vertical distance we can without crossing any horizontal lines

Requirements:

pip3 install scipy

pip3 install matplotlib

from scipy.cluster.hierarchy import dendrogram, linkage

from matplotlib import pyplot as plt

import pandas

#I am creating a dictionary here titled inputdata

inputdata={}

inputdata = pandas.read\_csv('example10\_results.csv').to\_dict()

# I created a new dictionary here for the comment column in my csv file

readability\_dictionary = inputdata.get('Readability')

readability\_list = list(readability\_dictionary.values())

#Lets create a dendrogram variable

# linkage is actually the algorithm itself of hierarchical clustering and then in

#linkage we have to specify on which data we apply and engage. This is X dataset

X = [[i] for i in readability\_list]

Z = linkage(X, 'ward')

fig = plt.figure(figsize=(25, 10))

dn = dendrogram(Z)

plt.title("Dendogram")

plt.xlabel('Readability Scores')

plt.ylabel('Euclidean distances')

plt.show()

from sklearn.cluster import AgglomerativeClustering

#Based on the dendogram, there are 4 clusters. Below, we can find each cluster each readability score belongs to

#Change number 4 below based on your interpretation of the dendogram

cluster = AgglomerativeClustering(

n\_clusters=4, affinity='euclidean', linkage='ward')

cluster.fit(X)

labels = cluster.labels\_

#labels

#print the labels to check the cluster each message belongs to

#print(labels)

#This is the Cluster Analysis Results in Dataframe

clusteranalysis\_results\_dataframe = pandas.DataFrame(labels)

file = pandas.read\_csv('example10\_results.csv')

file['Readability\_Cluster'] = clusteranalysis\_results\_dataframe

file.to\_csv('example11\_results.csv', index=True)

**Example 12:**  This example demonstrates how to take into consideration both Sarcasm and readability scores when clustering messages.

When using multiple units of data as input for the cluster analysis, we need to normalize the data so that the scale of each variable is the same. Why is this important? Well, if the scale of the variables is not the same, the model might become biased towards the variables with a higher magnitude

Tutorial:

<https://www.analyticsvidhya.com/blog/2019/05/beginners-guide-hierarchical-clustering/>

from scipy.cluster.hierarchy import dendrogram, linkage

from matplotlib import pyplot as plt

import pandas

import numpy

#I am creating a dictionary here titled inputdata

inputdata={}

inputdata = pandas.read\_csv('example10\_results.csv').to\_dict()

# I created a new dictionary here for the comment column in my csv file

readability\_dictionary = inputdata.get('Readability')

sarcasm\_dictionary = inputdata.get("Is\_Sarcastic")

readability\_list = list(readability\_dictionary.values())

sarcasm\_list = list(sarcasm\_dictionary.values())

updated\_sarcasm\_list = []

for i in sarcasm\_list:

if i == 'Not Sarcasm':

updated\_sarcasm\_list.append(int('0'))

elif i =='Sarcasm':

updated\_sarcasm\_list.append(int('1'))

combined\_list = list(zip(readability\_list,updated\_sarcasm\_list))

from sklearn.preprocessing import normalize

data\_scaled = normalize(combined\_list)

import scipy.cluster.hierarchy as shc

plt.figure(figsize=(10, 7))

plt.title("Dendrograms")

dend = shc.dendrogram(shc.linkage(data\_scaled, method='ward'))

plt.show()

from sklearn.cluster import AgglomerativeClustering

#Based on the dendogram, there are 2 clusters. Below, we can find each cluster each readability score belongs to

#Change number 2 below based on your interpretation of the dendogram

cluster = AgglomerativeClustering(

n\_clusters=2, affinity='euclidean', linkage='ward')

cluster.fit(data\_scaled)

labels = cluster.labels\_

labels

print(labels)

**Example 13**: This example demonstrates how to filter the Youtube comments from a CSV file by using another column from the same CSV file and then drawing wordclouds

1. Filter the comments that are only sarcastic and display a wordcloud of only sarcastic comments

Required library: pip3 install wordcloud

import pandas

import re

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

from nltk.tokenize.treebank import TreebankWordDetokenizer

from wordcloud import WordCloud

import matplotlib.pyplot as plt

#I am creating a dictionary here titled inputdata

inputdata={}

inputdata = pandas.read\_csv('example11\_results.csv').to\_dict()

comment\_dictionary = inputdata.get('Comment')

sarcasm\_dictionary = inputdata.get("Is\_Sarcastic")

# I am converting the comment dictionary to a list so I can analyze the data

comment\_list = list(comment\_dictionary.values())

#I would like to filter comments based on sarcasm values from the csv file. Therefore, I created this sarcasm list, which I will use below

sarcasm\_list = list(sarcasm\_dictionary.values())

#I like to put all sarcastic comments inside the sarcastic comment list

sarcastic\_comment\_list = []

#I like to put ll unsarcastic comments inside the unsarcastic comment list

unsarcastic\_comment\_list =[]

for i in range(len(comment\_list)):

#I am using an if statement to check a row's sarcasm column value from the csv file

if sarcasm\_list[i] == 'Sarcasm':

sarcastic\_comment\_list.append((comment\_list[i]))

elif sarcasm\_list[i] == 'Not Sarcasm':

unsarcastic\_comment\_list.append((comment\_list[i]))

#I will now clean text in sarcastic comments

for i in range(len(sarcastic\_comment\_list)):

# I am cleaning the comments data

# 1. make all letters lowercase

sarcastic\_comment\_list[i] = sarcastic\_comment\_list[i].lower()

# 2. Remove special characters

sarcastic\_comment\_list[i] = re.sub(r'\W+', ' ', sarcastic\_comment\_list[i])

# 3. Remove numbers

sarcastic\_comment\_list[i] = ''.join(c if c not in map(str, range(0, 10)) else "" for c in sarcastic\_comment\_list[i])

# 4. Remove stop words

# We need to convert the string to tokens in order to remove the stop words then convert the tokens back to string format

# A token is a string of contiguous characters between two spaces, or between a space and punctuation marks.

sarcastic\_comment\_tokens = word\_tokenize(sarcastic\_comment\_list[i])

cleaned\_sarcastic\_comment\_tokens = [word for word in sarcastic\_comment\_tokens if not word in stopwords.words()]

# I am converying the tokens back into string as an input for the wordcloud. Wordcloud does not accept token as input

cleaned\_sarcastic\_comments\_string\_data = TreebankWordDetokenizer().detokenize(cleaned\_sarcastic\_comment\_tokens)

#I am drawing a wordcloud for the sarcastic comments

wordcloud = WordCloud(width=500, height=500,

background\_color='white',

min\_font\_size=10).generate(cleaned\_sarcastic\_comments\_string\_data)

plt.figure(figsize=(15, 10), facecolor=None)

plt.imshow(wordcloud)

plt.axis("off")

plt.title("Sarcastic Comments WordCloud")

plt.show()

1. Enhance the previous example to draw wordclouds of both sarcastic and unsarcastic comments

import pandas

import re

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

from nltk.tokenize.treebank import TreebankWordDetokenizer

from wordcloud import WordCloud

import matplotlib.pyplot as plt

#I am creating a dictionary here titled inputdata

inputdata={}

inputdata = pandas.read\_csv('example11\_results.csv').to\_dict()

comment\_dictionary = inputdata.get('Comment')

sarcasm\_dictionary = inputdata.get("Is\_Sarcastic")

# I am converting the comment dictionary to a list so I can analyze the data

comment\_list = list(comment\_dictionary.values())

#I would like to filter comments based on sarcasm values from the csv file. Therefore, I created this sarcasm list, which I will use below

sarcasm\_list = list(sarcasm\_dictionary.values())

#I like to put all sarcastic comments inside the sarcastic comment list

sarcastic\_comment\_list = []

#I like to put ll unsarcastic comments inside the unsarcastic comment list

unsarcastic\_comment\_list =[]

for i in range(len(comment\_list)):

#I am using an if statement to check a row's sarcasm column value from the csv file

if sarcasm\_list[i] == 'Sarcasm':

sarcastic\_comment\_list.append((comment\_list[i]))

elif sarcasm\_list[i] == 'Not Sarcasm':

unsarcastic\_comment\_list.append((comment\_list[i]))

#I will now clean text in sarcastic comments

for i in range(len(sarcastic\_comment\_list)):

# I am cleaning the comments data

# 1. make all letters lowercase

sarcastic\_comment\_list[i] = sarcastic\_comment\_list[i].lower()

# 2. Remove special characters

sarcastic\_comment\_list[i] = re.sub(r'\W+', ' ', sarcastic\_comment\_list[i])

# 3. Remove numbers

sarcastic\_comment\_list[i] = ''.join(c if c not in map(str, range(0, 10)) else "" for c in sarcastic\_comment\_list[i])

# 4. Remove stop words

# We need to convert the string to tokens in order to remove the stop words then convert the tokens back to string format

# A token is a string of contiguous characters between two spaces, or between a space and punctuation marks.

sarcastic\_comment\_tokens = word\_tokenize(sarcastic\_comment\_list[i])

cleaned\_sarcastic\_comment\_tokens = [word for word in sarcastic\_comment\_tokens if not word in stopwords.words()]

# I am converying the tokens back into string as an input for the wordcloud. Wordcloud does not accept token as input

cleaned\_sarcastic\_comments\_string\_data = TreebankWordDetokenizer().detokenize(cleaned\_sarcastic\_comment\_tokens)

#I am drawing a wordcloud for the sarcastic comments

wordcloud = WordCloud(width=500, height=500,

background\_color='white',

min\_font\_size=10).generate(cleaned\_sarcastic\_comments\_string\_data)

plt.figure(figsize=(15, 10), facecolor=None)

plt.imshow(wordcloud)

plt.axis("off")

plt.title("Sarcastic Comments WordCloud")

plt.show()

for i in range(len(unsarcastic\_comment\_list)):

# I am cleaning the comments data

# 1. make all letters lowercase

unsarcastic\_comment\_list[i] = unsarcastic\_comment\_list[i].lower()

# 2. Remove special characters

unsarcastic\_comment\_list[i] = re.sub(r'\W+', ' ', unsarcastic\_comment\_list[i])

# 3. Remove numbers

unsarcastic\_comment\_list[i] = ''.join(c if c not in map(str, range(0, 10)) else "" for c in unsarcastic\_comment\_list[i])

# 4. Remove stop words

# We need to convert the string to tokens in order to remove the stop words then convert the tokens back to string format

# A token is a string of contiguous characters between two spaces, or between a space and punctuation marks.

unsarcastic\_comment\_tokens = word\_tokenize(unsarcastic\_comment\_list[i])

cleaned\_unsarcastic\_comment\_tokens = [word for word in unsarcastic\_comment\_tokens if not word in stopwords.words()]

# I am converying the tokens back into string as an input for the wordcloud. Wordcloud does not accept token as input

cleaned\_unsarcastic\_comments\_string\_data = TreebankWordDetokenizer().detokenize(cleaned\_unsarcastic\_comment\_tokens)

#I am drawing a wordcloud for the sarcastic comments

wordcloud = WordCloud(width=500, height=500,

background\_color='white',

min\_font\_size=10).generate(cleaned\_unsarcastic\_comments\_string\_data)

plt.figure(figsize=(15, 10), facecolor=None)

plt.imshow(wordcloud)

plt.axis("off")

plt.title("UnSarcastic Comments WordCloud")

plt.show()