# PROJECT REPORT

ON

# “DATA PREPROCESSING”

BY

# UMAR AYOUB HAJAM

# (PROJECT LEAD)

Executive Summary

The dataset “Twitter Climate Change Sentiment Dataset” chosen for this project regarding “General Awareness on Climate Change” was downloaded from Kaggle. The Dataset contains texts, comments, replays extracted from the Social networking platform Twitter. The dataset contains three variables/features/columns and around 50 thousand messages/comments/replays/rows/tweets. The dataset does not contain any missing values. Using Feature Engineering two new variables were created “Word\_Count” and “Text Length” containing the number of words per sentence and number of characters per sentence.

Data

1. In this project the data set used Twitter Climate change Sentiment Dataset a csv file.
2. The Dataset Contains Three Features/columns.
3. Sentiment – Numeric Variable, Range (-1: 2) specifying the sentiment a sentence pose.
4. 2(News): the tweet links to factual news about climate change
5. 1(Pro): the tweet supports the belief of man-made climate change
6. 0(Neutral: the tweet neither supports nor refutes the belief of man-made climate change
7. -1(Anti): the tweet does not believe in man-made climate change
8. Message – Character Variable containing the text/comment/replay/message/tweet.
9. Tweeted – Numeric Variable containing the specific key to locate every tweet.
10. Data Source: https://www.kaggle.com/edqian/twitter-climate-change-sentiment-dataset.

Programming Language.

Python 3.8.

Integrated Development Environment.

PyCharm 2021.1 (Community Edition)

Build #PC-211.6693.115, built on April 6, 2021

Runtime version: 11.0.10+9-b1341.35 amd64

VM: Dynamic Code Evolution 64-Bit Server VM by JetBrains s.r.o.

Windows 10 10.0

GC: ParNew, ConcurrentMarkSweep

Memory: 3933M

Cores: 8

Modules/Libraries

The Modules/Libraries used in Data Preprocessing are:

1. Pandas
2. Nltk- Natural Language Text Kit
3. Sklearn
4. Seaborn

Importing the required libraries

import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
from nltk.stem import PorterStemmer  
import regex as re  
from nltk.corpus import stopwords  
from nltk.tokenize import word\_tokenize

Exploratory Data Analysis

Data = pd.read\_csv("twitter\_sentiment\_data.csv")  
Data['word\_counts'] = Data['message'].str.split().str.len()  
Data["Text Length"] = Data["message"].str.len()  
Data.groupby('sentiment')['word\_counts'].mean()  
  
# Exploratory analysis  
Data.describe()

sentiment tweetid word\_counts Text Length

count 43943.000000 4.394300e+04 43943.000000 43943.000000

mean 0.853924 8.367966e+17 17.400792 122.823954

std 0.853543 8.568506e+16 4.621521 24.720780

min -1.000000 5.926334e+17 1.000000 7.000000

25% 0.000000 7.970376e+17 14.000000 111.000000

50% 1.000000 8.402301e+17 18.000000 133.000000

75% 1.000000 9.020003e+17 21.000000 140.000000

max 2.000000 9.667024e+17 97.000000 623.000000

Print the Columns/features in the Dataset

print(Data.columns)

Index(['sentiment', 'message', 'tweetid', 'word\_counts', 'Text Length'], dtype='object')

‘message variable/column/feature is the only feature with non numerical data ’

print(Data["message"])

0 @tiniebeany climate change is an interesting h...

1 RT @NatGeoChannel: Watch #BeforeTheFlood right...

2 Fabulous! Leonardo #DiCaprio's film on #climat...

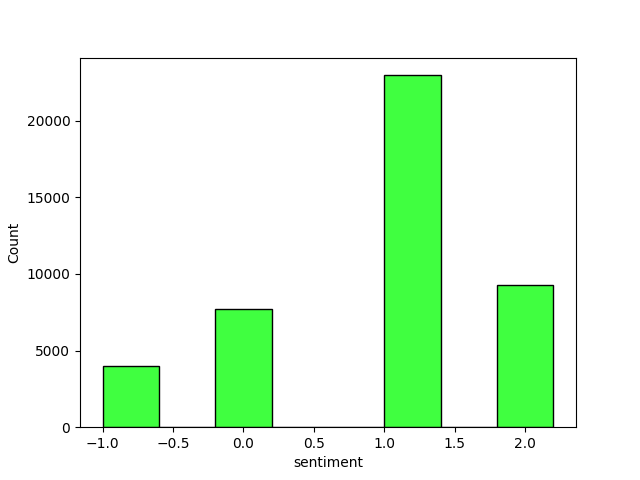
3 RT @Mick\_Fanning: Just watched this amazing do...

4 RT @cnalive: Pranita Biswasi, a Lutheran from ...

...

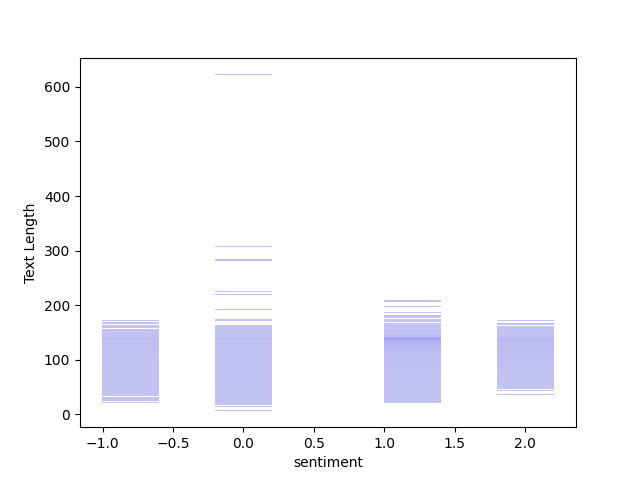
43938 Dear @realDonaldTrump,\nYeah right. Human Medi...

Data Visualization

Plotting seaborn histplot(Histogram) to check the distribution of data 

The histogram show that most of the tweets are pro climate change, and has low number of anti-climate change tweets. This shows that people are aware of the climate change

Plotting a seaborn histplot(Histogram) to check the relation between variables text length and sentiment



The Histogram show that there isn’t much difference in the text length of different classes. The major difference seen is in the neutral class as people tend to write more if they are neutral (0) some tweets going as far as 600 characters.

Checking the dataset for missing values

Data.isna().sum()

sentiment 0

message 0

tweetid 0

word\_counts 0

Text Length 0

dtype: int64

There are no missing values (NA’s) in the dataset.

Natural Language Processing

print(Data["message"])

0 @tiniebeany climate change is an interesting h...

1 RT @NatGeoChannel: Watch #BeforeTheFlood right...

2 Fabulous! Leonardo #DiCaprio's film on #climat...

3 RT @Mick\_Fanning: Just watched this amazing do...

4 RT @cnalive: Pranita Biswasi, a Lutheran from ...

...

43938 Dear @realDonaldTrump,\nYeah right. Human Medi...

43939 What will your respective parties do to preven...

43940 RT @MikkiL: UN Poll Shows Climate Change Is th...

43941 RT @taehbeingextra: i still can$q$t believe th...

43942 @Likeabat77 @zachhaller \n\nThe wealthy + foss...

Processing the main feature of the dataset (message). The message feature is a character variable to process the variable we will have to clean the variable first as their as many unwanted characters in the variable which do not contribute to the meaning of the sentence.

Using Regular Expression to clean the variable message.

def msg\_cleaning(msg):  
 # Removing @abc12  
 msg = re.sub(r'@[A-Za-z0-9]+', '', msg)  
 # Removing Hashtags  
 msg = re.sub(r'#', '', msg)  
 # Removing Chines  
 msg = re.sub(r'[^\x00-\x7F]+', '', msg)  
 # Removing Retweets  
 msg = re.sub(r'RT[\s]+', '', msg)  
 # Removing HyperLinks  
 msg = re.sub(r'https?:\/\/\s+', '', msg)  
 # Removing numeric values  
 msg = re.sub(r'\d+', '', msg)  
 msg = re.sub(r'aa[A-Za-z0-9]+', '', msg)  
 msg = re.sub(r'zz[A-Za-z0-9]+', '', msg)  
 return msg

In the function msg\_cleaning, we are getting rid of

1. Mentions (@tini).
2. Hashtags (#happyworld).
3. Retweets (RT).
4. Hyperlinks (https/http).
5. Numbers (0-9).
6. Chinees words.

Changing the characters of the message feature into lowercase.

Data["message"] = Data["message"].str.lower()

0 climate change is an interesting hustle as it...

1 : watch beforetheflood right here, as travels...

2 fabulous! leonardo dicaprio's film on climate ...

3 \_fanning: just watched this amazing documentar...

4 : pranita biswasi, a lutheran from odisha, giv...

...

43938 dear ,\nyeah right. human mediated climate cha...

43939 what will your respective parties do to preven...

43940 : un poll shows climate change is the lowest o...

43941 : i still can$q$t believe this gif of taehyung...

43942 \n\nthe wealthy + fossil fuel industry know ...

Name: message, Length: 43943, dtype: object

Tokenization

Using Word Tokenization

def identify\_tokens(row):  
 ide\_words = row["message"]  
 tokens = word\_tokenize(ide\_words)  
  
 token\_words = [w for w in tokens if w.isalpha()]  
 return token\_words  
  
  
Data["message"] = Data.apply(identify\_tokens, axis=1)  
print(Data['message'])

0 [climate, change, is, an, interesting, hustle,...

1 [watch, beforetheflood, right, here, as, trave...

2 [fabulous, leonardo, dicaprio, film, on, clima...

3 [just, watched, this, amazing, documentary, by...

4 [pranita, biswasi, a, lutheran, from, odisha, ...

...

43938 [dear, yeah, right, human, mediated, climate, ...

43939 [what, will, your, respective, parties, do, to...

43940 [un, poll, shows, climate, change, is, the, lo...

43941 [i, still, can, q, t, believe, this, gif, of, ...

43942 [the, wealthy, fossil, fuel, industry, know, c...

# Stemming the words

stemming = PorterStemmer()

def stem\_list(row):  
 my\_list = row["message"]  
 stemmed\_list = [stemming.stem(word) for word in my\_list]  
 return (stemmed\_list)  
  
  
Data["message"] = Data.apply(stem\_list, axis=1)  
print(Data["message"])

0 [climat, chang, is, an, interest, hustl, as, i...

1 [watch, beforetheflood, right, here, as, trave...

2 [fabul, leonardo, dicaprio, film, on, climat, ...

3 [just, watch, thi, amaz, documentari, by, leon...

4 [pranita, biswasi, a, lutheran, from, odisha, ...

...

43938 [dear, yeah, right, human, mediat, climat, cha...

43939 [what, will, your, respect, parti, do, to, pre...

43940 [un, poll, show, climat, chang, is, the, lowes...

43941 [i, still, can, q, t, believ, thi, gif, of, ta...

43942 [the, wealthi, fossil, fuel, industri, know, c...

Stop word removal

stops = set(stopwords.words("english"))  
stops.update(["aa", "aaa", "aaaa", "aaaaa", "aaaaaa", "aaaaaaa", "aaaaaaaa", "aaaaaaaaa", "aaaaaaaaaaaaaaaaaaaah"])  
  
  
def remove\_stops(row):  
 my\_list = row["message"]  
 meningful\_words = [w for w in my\_list if not w in stops]  
 return(meningful\_words)

Data["message"] = Data.apply(remove\_stops, axis=1)  
print(Data["message"])

0 [climat, chang, interest, hustl, wa, global, w...

1 [watch, beforetheflood, right, travel, world, ...

2 [fabul, leonardo, dicaprio, film, climat, chan...

3 [watch, thi, amaz, documentari, leonardodicapr...

4 [pranita, biswasi, lutheran, odisha, give, tes...

...

43938 [dear, yeah, right, human, mediat, climat, cha...

43939 [respect, parti, prevent, climat, chang, globa...

43940 [un, poll, show, climat, chang, lowest, global...

43941 [still, q, believ, thi, gif, taehyung, save, h...

43942 [wealthi, fossil, fuel, industri, know, climat...

Name: message, Length: 43943, dtype: object

Saving the file

Data.to\_csv("First\_processed.csv

Part-1 Complete

# Part- 2

# Importing the required libraries

import pandas as pd  
from sklearn.feature\_extraction.text import TfidfVectorizer  
from sklearn.model\_selection import train\_test\_split, RandomizedSearchCV, GridSearchCV, cross\_val\_score  
import numpy as np  
from sklearn.linear\_model import LogisticRegression  
from sklearn import metrics  
from sklearn.metrics import classification\_report, confusion\_matrix, f1\_score, precision\_score, recall\_score, accuracy\_score

# Reading in the data

Data = pd.read\_csv("First\_processed.csv")  
X = Data["message"]  
Y = Data["sentiment"]

# Splitting the data into training set, validation set, test set

# Splitting the data  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.25, random\_state=1103)  
  
# Splitting the data into validation  
X\_test, x\_val, y\_test, y\_val = train\_test\_split(X\_test, y\_test, test\_size=0.5, random\_state=1103)

# Using TF-IDF Vectorizer

# Using Ngrams with N=1:3

tfidf = TfidfVectorizer(ngram\_range=(1, 3), max\_features=12000, use\_idf=True)  
tfidf.fit\_transform(X\_train)  
tfidf.fit\_transform(x\_val)

# Testing the model without the features added using feature engineering

# Algorithm used: Logistic Regression

grid = {"C": np.logspace(-1, -3, 3, 7, 9), "penalty": ["none", "l2"]}# l1 lasso l2 ridge  
logreg = LogisticRegression(n\_jobs=6, max\_iter=2000, verbose=True)  
#logreg\_cv = GridSearchCV(logreg, grid, cv=10, verbose=True)  
# X\_train["word\_count"] = Data["word\_counts"]  
# X\_test["word\_count"] = Data["word\_counts"]  
# X\_train["Text Length"] = Data["Text Length"]  
# X\_test["Text Length"] = Data["Text Length"]  
logreg.fit(tfidf.transform(X\_train), y\_train)  
  
rfc\_predict = logreg.predict(tfidf.transform(x\_val))  
print("ACCURACY SCORE:", metrics.accuracy\_score(y\_val, rfc\_predict))  
print("::::Confusion Matrix::::")  
print(confusion\_matrix(y\_val, rfc\_predict))  
print("\n")  
  
print(":::Classification Report:::")  
print(classification\_report(y\_val, rfc\_predict, target\_names=['Class 1', 'Class 2', 'Class 3', 'Class 4']))  
print("\n")  
  
print(pd.crosstab(y\_val, rfc\_predict, rownames=["Orgnl"], colnames=['Predicted']))

# Results

# 

# Contact Details.

Phone No.: +91 7889911471

Email: [umerayoub54@gmail.com](mailto:umerayoub54@gmail.com)

linkedin: <https://www.linkedin.com/in/umar-ayoub-929831180/> github:<https://github.com/1UmAr1>

personal website: http://www.umerhajam.com