Logistic_Regression

March 12, 2023

[ ]:
```
from sklearn.linear_model import LogisticRegression
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import norm
from google.colab import drive
from sklearn.feature_extraction import text
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
import random
import time
import re
import string
from sklearn.naive_bayes import GaussianNB, MultinomialNB
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.feature_selection import SelectKBest, chi2,␣
↪f_classif,mutual_info_classif,f_regression
from sklearn.preprocessing import Normalizer
from sklearn import model_selection
from sklearn import svm
import nltk
from nltk import word_tokenize
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet
from nltk import word_tokenize
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet
from nltk.tokenize.treebank import TreebankWordDetokenizer
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords
nltk.download('omw-1.4')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
```

```
nltk.download('wordnet')
nltk.download('stopwords')
```

```
[nltk_data] Downloading package omw-1.4 to /root/nltk_data…
[nltk_data]   Package omw-1.4 is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data…
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data…
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package wordnet to /root/nltk_data…
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data…
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data…
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package wordnet to /root/nltk_data…
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
```

[ ]:
```
True
```

[ ]:
```
#import the data


drive.mount('/content/gdrive/', force_remount=True)

train_data_initial = pd.read_csv('/content/gdrive/MyDrive/train.csv')
test_data = pd.read_csv('/content/gdrive/MyDrive/test.csv')

print('shape train:',train_data_initial.shape)
print('shape test:',test_data.shape)
```

```
Mounted at /content/gdrive/
shape train: (718, 2)
shape test: (279, 2)
```

[ ]:
```
def shuffle_data(df):
    random.seed(0)  # Use a fixed seed for the random number generator
    df = df.sample(frac=1, random_state=0).reset_index(drop=True)
    return df
```

[ ]:
```
#function for creating the test csv file to upload to kaggle
def create_test_csv(data, outfile_name):
  rawdata= {'subreddit':data}
  csv = pd.DataFrame(rawdata, columns = ['subreddit'])
  csv.to_csv(outfile_name,index=True, header=True)
  print ("File saved.")
```

[ ]:
```
#shuffle the data and split the features from the label
train_data = shuffle_data(train_data_initial)



train_x_initial = train_data["body"]
train_y = train_data["subreddit"]
test_x_initial = test_data["body"]
```

[ ]:
```
#remove punctuation from train and test
train_x = train_x_initial.copy()

for i in range(train_x.shape[0]):
  train_x[i]= train_x[i].translate(str.maketrans('', '', string.punctuation))

test_x = test_x_initial.copy()

for i in range(test_x.shape[0]):
  test_x[i]= test_x[i].translate(str.maketrans('', '', string.punctuation))
```

[ ]:
```
print(test_x[5])
#print(test_x_initial[5])
```

```
I like cars with screensas long as the UI is intuitive and phonelike Ive never
driven a new Edge nor have I driven a Ford with Sync 3

As far as I can tell it looks good and concise I like it
```

[ ]:
```
def preprocess_text(text):
    text = text.lower()
    text = re.sub(r'\d+', '', text)
    return text
```

[ ]:
```
def remove_punctuation(text):
   translator = str.maketrans('', '', string.punctuation)
   text = text.translate(translator)
   return text
```

[ ]:
```
#create a dictionary of stop words
stop_words_nltk = set(stopwords.words('english'))
```

```
stop_words_sklearn = text.ENGLISH_STOP_WORDS
stop_words_library = stop_words_sklearn.union(stop_words_nltk)
```

[ ]:
```
#initial training without removing parameters

t_start = time.time()

pipe_params = {
    'classify__penalty': ['l1', 'l2'],   #'classify__penalty': ['l1', 'l2'],
    'classify__C': [0.01, 0.1, 1.0, 10.0],     #'classify__C': [0.01, 0.1, 1.0,␣
 ↪10.0],
    'classify__solver': ['liblinear'],   #  #'classify__solver': ['liblinear',␣
 ↪'newton-cg', 'lbfgs', 'sag', 'saga'],
    'classify__max_iter': [100, 500, 1000],
    'classify__class_weight': [None, 'balanced']
}

vectorizer = CountVectorizer()
model = LogisticRegression()

pipe = Pipeline(
    [("vect", vectorizer),("classify",model)]
)

grid = model_selection.GridSearchCV(pipe, pipe_params, verbose=1, n_jobs=-1)

grid.fit(train_x, train_y)

t_end = time.time()

elapsed_time = t_end-t_start
accuracy = round(grid.best_score_ * 100,3)

print(f"The best accuracy is {accuracy}.")
print(f"The winning parameters are {grid.best_params_}")
print(f"Run time: {elapsed_time} seconds")
```

```
Fitting 5 folds for each of 48 candidates, totalling 240 fits
The best accuracy is 89.411.
The winning parameters are {'classify__C': 10.0, 'classify__class_weight': None,
'classify__max_iter': 100, 'classify__penalty': 'l2', 'classify__solver':
'liblinear'}
Run time: 32.65085744857788 seconds
```

[ ]:
```
#initial training with stop words  93.037

t_start = time.time()
```

```
pipe_params = {
    'classify__penalty': ['l2'],   #'classify__penalty': ['l1', 'l2'],
    'classify__C': [10.0],     #'classify__C': [0.01, 0.1, 1.0, 10.0],
    'classify__solver': ['sag'],   #'classify__solver': ['liblinear',␣
 ↪'newton-cg', 'lbfgs', 'sag', 'saga'],
    'classify__max_iter': [1000], # 'classify__max_iter': [100, 500, 1000],
    #'classify__class_weight': ['balanced'],     #'classify__class_weight':␣
 ↪[None, 'balanced'],
    "vect__stop_words": [list(stop_words_nltk), list(stop_words_sklearn),␣
 ↪list(stop_words_library)],
    "selecter__k":[5000]
}




#stop_words_nltk
#stop_words_sklearn


vectorizer = CountVectorizer()
selecter = SelectKBest(chi2)
model = LogisticRegression()



pipe = Pipeline(
     [("vect", vectorizer),("selecter", selecter),("classify",model)]
)

grid = model_selection.GridSearchCV(pipe, pipe_params, verbose=1, n_jobs=-1)

grid.fit(train_x, train_y)

t_end = time.time()

elapsed_time = t_end-t_start
accuracy = round(grid.best_score_ * 100,3)

print(f"The best accuracy is {accuracy}.")
print(f"The winning parameters are {grid.best_params_}")
print(f"Run time: {elapsed_time} seconds")
```

```
Fitting 5 folds for each of 3 candidates, totalling 15 fits
The best accuracy is 93.037.
The winning parameters are {'classify__C': 10.0, 'classify__max_iter': 1000,
```

'classify__penalty': 'l2', 'classify__solver': 'sag', 'selecter__k': 5000,
'vect__stop_words': ['thereupon', 'latter', 'up', 'aren', 'almost', "that'll",
'among', 'their', 'both', 'never', 'been', 'further', 'along', 'whoever', 'inc',
"you'd", 'again', 'eleven', 'another', 'his', 'empty', 'always', 'though',
'within', 'bill', 'put', 'needn', 'behind', 'could', 'all', 'four',
'beforehand', 'otherwise', 'well', 'upon', 'often', "won't", 'under', 'its',
'perhaps', 'fire', 'un', "don't", 'already', 'from', 'six', 'still', 'get',
'go', 'as', 'during', 'toward', 'sixty', 'not', 'isn', 'amongst', 'had',
"didn't", 'see', 'first', 'nowhere', "wouldn't", 'mostly', 'thin', 'detail',
'hence', 'ours', 'against', 'being', 'seem', 'they', 'keep', 'none', 'less',
'become', 'where', 'sometimes', 'move', 'mill', 'nine', 'hereafter', 'yours',
'few', 'i', 'much', 'twenty', 'has', 'last', 'or', 'while', 'anything', 'don',
'top', 'whereas', 'something', 'hasn', 'such', 'between', 'down', 'm', 'off',
'even', 'others', 'we', 'system', 'after', 'therein', 'thereafter', 'this',
'weren', 'cant', 'two', 'which', 'thus', 'wouldn', 'fifteen', 'can', 'wasn',
'will', 'made', 'are', 'once', 'several', 'third', 'whenever', 'must',
'themselves', "shan't", 'front', 'about', 'a', 'amoungst', 'some', "hadn't",
'show', 'whether', 'formerly', 'sincere', 'the', 'with', 'more', 'side',
'there', "you're", 'mine', 'however', 'whereafter', 'give', 'ain', 'mustn',
'here', 'became', 'did', 'now', "isn't", 'shan', 'he', "hasn't", 'any', 'ma',
'nobody', 'didn', 'theirs', 'until', 'you', 'it', 'hereupon', 'above', 'noone',
'least', 'becomes', 'hasnt', 'other', 'them', 'were', 'someone', 'eg', "she's",
'con', 'take', 'haven', 'serious', 'when', 'alone', 'my', 'anyone', "aren't",
'fifty', 'was', 'please', "should've", 'nevertheless', 'de', 'doesn', 'since',
"shouldn't", 'latterly', 'although', 'name', 'cry', 'ourselves', 'too', 'is',
'and', 'herein', 'an', 'full', 'y', 'very', 'forty', 'itself', 'me', 'before',
'd', 'do', 'moreover', 'back', 'eight', 'most', 'so', 'therefore', 'via', 'who',
'thereby', 'through', 'shouldn', 'co', 'many', 'hadn', 'whom', "it's",
'yourselves', 'ltd', 'due', 'somewhere', 's', 'ie', 'done', 'afterwards',
'himself', 'onto', 'call', 'beyond', 'below', 'yourself', 'everyone', "haven't",
'him', 'just', 'seeming', 'does', 'may', 'per', 'find', 'because', 'if',
'would', 'whence', 'cannot', 'enough', 'on', 'five', 'ever', 'across',
'herself', 'to', "couldn't", 'us', "doesn't", 'wherein', 'thence', 'in',
'describe', 'whole', 'whatever', 'everything', 'elsewhere', "you've", "mustn't",
'fill', 'your', 're', 'might', 'twelve', 'having', 'besides', 'am', 'nor', 've',
"needn't", 'else', 'hers', 'what', 'hundred', 'beside', 'thick', 'o', 'either',
'throughout', 'only', 'anywhere', 'couldn', "weren't", 'those', 'interest',
'amount', 'neither', 'each', 'doing', 'bottom', 'sometime', 'next', 'without',
'meanwhile', 'seemed', 'except', 'why', 'for', 'over', "you'll", 'whereby',
'same', "wasn't", 'former', 'wherever', 't', 'around', 'mightn', 'one', 'she',
'no', 'part', 'anyhow', 'couldnt', 'own', 'three', 'rather', 'won', 'our',
"mightn't", 'that', 'yet', 'etc', 'by', 'indeed', 'into', 'at', 'also', 'thru',
'how', 'than', 'towards', 'ten', 'myself', 'of', 'then', 'll', 'anyway',
'hereby', 'her', 'should', 'whither', 'be', 'seems', 'have', 'nothing',
'everywhere', 'whereupon', 'these', 'together', 'becoming', 'but', 'out',
'every', 'namely', 'somehow', 'found', 'whose']}
Run time: 11.839037895202637 seconds

```
/usr/local/lib/python3.8/dist-packages/sklearn/linear_model/_sag.py:350:
ConvergenceWarning: The max_iter was reached which means the coef_ did not
converge
  warnings.warn(
```

[ ]:
```
#stem lemmatizer
def get_wordnet_pos(word):
    """Map POS tag to first character lemmatize() accepts"""
    tag = nltk.pos_tag([word])[0][1][0].upper()
    tag_dict = {"J": wordnet.ADJ,
                "N": wordnet.NOUN,
                "V": wordnet.VERB,
                "R": wordnet.ADV}
    return tag_dict.get(tag, wordnet.NOUN)

class LemmaTokenizer_Pos:
     def __init__(self):
       self.wnl = WordNetLemmatizer()
     def __call__(self, doc):
       return [self.wnl.lemmatize(t,pos =get_wordnet_pos(t)) for t in␣
 ↪word_tokenize(doc) if t.isalpha()]

class LemmaTokenizer:
     def __init__(self):
       self.wnl = WordNetLemmatizer()
     def __call__(self, doc):
       return [self.wnl.lemmatize(t,pos ="v") for t in word_tokenize(doc) if t.
 ↪isalpha()]

class LemmaTokenizer_word:
     def __init__(self):
       self.wnl = WordNetLemmatizer()
     def __call__(self, doc):
       return [self.wnl.lemmatize(t,pos ="v") for t in word_tokenize(doc) ]


class StemTokenizer:
     def __init__(self):
       self.wnl =PorterStemmer()
     def __call__(self, doc):
       return [self.wnl.stem(t) for t in word_tokenize(doc) if t.isalpha()]
```

[ ]:
```
stop_words_custom = [
# All pronouns and associated words
"i","i'll","i'd","i'm","i've","ive","me","myself","you","you'll","you'd","you're","you've","yo
"he'd",
"he's",
```

```
"him",
"she",
"she'll",
"she'd",
"she's",
"her",
"it",
"it'll",
"it'd",
"it's",
"itself",
"oneself",
"we",
"we'll",
"we'd",
"we're",
"we've",
"us",
"ourselves",
"they",
"they'll",
"they'd",
"they're",
"they've",
"them",
"themselves",
"everyone",
"everyone's",
"everybody",
"everybody's",
"someone",
"someone's",
"somebody",
"somebody's",
"nobody",
"nobody's",
"anyone",
"anyone's",
"everything",
"everything's",
"something",
"something's",
"nothing",
"nothing's",
"anything",
"anything's",
# All determiners and associated words
```

```
"a",
"an",
"the",
"this",
"that",
"that's",
"these",
"those",
"my",
#"mine",   #Omitted since mine can refer to something else
"your",
"yours",
"his",
"hers",
"its",
"our",
"ours",
"own",
"their",
"theirs",
"few",
"much",
"many",
"lot",
"lots",
"some",
"any",
"enough",
"all",
"both",
"half",
"either",
"neither",
"each",
"every",
"certain",
"other",
"another",
"such",
"several",
"multiple",
# "what",#Dealt with later on
"rather",
"quite",
# All prepositions
"aboard",
"about",
```

```
"above",
"across",
"after",
"against",
"along",
"amid",
"amidst",
"among",
"amongst",
"anti",
"around",
"as",
"at",
"away",
"before",
"behind",
"below",
"beneath",
"beside",
"besides",
"between",
"beyond",
"but",
"by",
"concerning",
"considering",
"despite",
"down",
"during",
"except",
"excepting",
"excluding",
"far",
"following",
"for",
"from",
"here",
"here's",
"in",
"inside",
"into",
"left",
"like",
"minus",
"near",
"of",
"off",
```

```
"on",
"onto",
"opposite",
"out",
"outside",
"over",
"past",
"per",
"plus",
"regarding",
"right",
#"round",   #Omitted
#"save",#Omitted
"since",
"than",
"there",
"there's",
"through",
"to",
"toward",
"towards",
"under",
"underneath",
"unlike",
"until",
"up",
"upon",
"versus",
"via",
"with",
"within",
"without",
# Irrelevant verbs
"may",
"might",
"will",
"won't",
"would",
"wouldn't",
"can",
"can't",
"cannot",
"could",
"couldn't",
"should",
"shouldn't",
"must",
```

```
"must've",
"be",
"being",
"been",
"am",
"are",
"aren't",
"ain't",
"is",
"isn't",
"was",
"wasn't",
"were",
"weren't",
"do",
"doing",
"don't",
"does",
"doesn't",
"did",
"didn't",
"done",
"have",
"haven't",
"having",
"has",
"hasn't",
"had",
"hadn't",
"get",
"getting",
"gets",
"got",
"gotten",
"go",
"going",
"gonna",
"goes",
"went",
"gone",
"make",
"making",
"makes",
"made",
"take",
"taking",
"takes",
```

```
"took",
"taken",
"need",
"needing",
"needs",
"needed",
"use",
"using",
"uses",
"used",
"want",
"wanna",
"wanting",
"wants",
"let",
"lets",
"letting",
"let's",
"suppose",
"supposing",
"supposes",
"supposed",
"seem",
"seeming",
"seems",
"seemed",
"say",
"saying",
"says",
"said",
"know",
"knowing",
"knows",
"knew",
"known",
"look",
"looking",
"looked",
"think",
"thinking",
"thinks",
"thought",
"feel",
"feels",
"felt",
"based",
"put",
```

```
"puts",
#"wanted"   #Omitted since the advective is relevant
# Question words and associated words
"who",
"who's",
"who've",
"who'd",
"whoever",
"whoever's",
"whom",
"whomever",
"whomever's",
"whose",
"whosever",
"whosever's",
"when",
"whenever",
"which",
"whichever",
"where",
"where's",
"where'd",
"wherever",
"why",
"why's",
"why'd",
"whyever",
"what",
"what's",
"whatever",
"whence",
"how",
"how's",
"how'd",
"however",
"whether",
"whatsoever",
# Connector words and irrelevant adverbs
"and",
"or",
"not",
"because",
"also",
"always",
"never",
"only",
"really",
```

```
"very",
"greatly",
"extremely",
"somewhat",
"no",
"nope",
"nah",
"yes",
"yep",
"yeh",
"yeah",
"maybe",
"perhaps",
"more",
"most",
"less",
"least",
"good",
"great",
"well",
"better",
"best",
"bad",
"worse",
"worst",
"too",
"thru",
"though",
"although",
"yet",
"already",
"then",
"even",
"now",
"sometimes",
"still",
"together",
"altogether",
"entirely",
"fully",
"entire",
"whole",
"completely",
"utterly",
"seemingly",
"apparently",
"clearly",
```

```
"obviously",
"actually",
"actual",
"usually",
"usual",
"literally",
"honestly",
"absolutely",
"definitely",
"generally",
"totally",
"finally",
"basically",
"essentially",
"fundamentally",
"automatically",
"immediately",
"necessarily",
"primarily",
"normally",
"perfectly",
"constantly",
"particularly",
"eventually",
"hopefully",
"mainly",
"typically",
"specifically",
"differently",
"appropriately",
"plenty",
"certainly",
"unfortunately",
"ultimately",
"unlikely",
"likely",
"potentially",
"fortunately",
"personally",
"directly",
"indirectly",
"nearly",
"closely",
"slightly",
"probably",
"possibly",
"especially",
```

```
"frequently",
"often",
"oftentimes",
"seldom",
"rarely",
"sure",
"while",
"whilst",
"able",
"unable",
"else",
"ever",
"once",
"twice",
"thrice",
"almost",
"again",
"instead",
"next",
"previous",
"unless",
"somehow",
"anyhow",
"anywhere",
"somewhere",
"everywhere",
"nowhere",
"further",
"anymore",
"later",
"ago",
"ahead",
"just",
"same",
"different",
"big",
"small",
"little",
"tiny",
"large",
"huge",
"pretty",
"mostly",
"anyway",
"anyways",
"otherwise",
"regardless",
```

```
"throughout",
"additionally",
"moreover",
"furthermore",
"meanwhile",
"afterwards",
# Irrelevant nouns
"thing",
"thing's",
"things",
"stuff",
"other's",
"others",
"another's",
"total",
"",
"false",
"none",
"way",
"kind",
# Lettered numbers and order
"zero",
"zeros",
"zeroes",
"one",
"ones",
"two",
"three",
"four",
"five",
"six",
"seven",
"eight",
"nine",
"ten",
"twenty",
"thirty",
"forty",
"fifty",
"sixty",
"seventy",
"eighty",
"ninety",
"hundred",
"hundreds",
"thousand",
"thousands",
```

```
"million",
"millions",
"first",
"last",
"second",
"third",
"fourth",
"fifth",
"sixth",
"seventh",
"eigth",
"ninth",
"tenth",
"firstly",
"secondly",
"thirdly",
"lastly",
# Greetings and slang
"hello",
"hi",
"hey",
"sup",
"yo",
"greetings",
"please",
"okay",
"ok",
"y'all",
"lol",
"rofl",
"thank",
"thanks",
"alright",
"kinda",
"dont",
"sorry",
"idk",
"tldr",
"tl",
"dr",  #This means that dr (doctor) is a bad feature because of tl;dr
"tbh",
"dude",
"tho",
"aka",
"plz",
"pls",
"bit",
```

```
"don",
# Miscellaneous
"www",
"https",
"http",
"com",
"etc"
"html",
"reddit",
"subreddit",
"subreddits",
"comments",
"reply",
"replies",
"thread",
"threads",
"post",
"posts",
"website",
"websites",
"web site",
"web sites"]
print('length custom:',len(stop_words_custom))
```

```
length custom: 589
```

[ ]:
```
print(len(stop_words_custom))
```

```
589
```

[ ]:
```
#function for creating the test csv file to upload to kaggle
def create_test_csv(data, outfile_name):
  rawdata= {'subreddit':data}
  csv = pd.DataFrame(rawdata, columns = ['subreddit'])
  csv.to_csv(outfile_name,index=True, header=True)
  print ("File saved.")
```

[ ]:
```
#initial training with stop words. LemmaTokenizer_word

t_start = time.time()

pipe_params = {
    'classify__penalty': ['l2'],   #'classify__penalty': ['l1', 'l2'],
    'classify__C': [10.0],     #'classify__C': [0.01, 0.1, 1.0, 10.0],
    'classify__solver': ['sag'],   #'classify__solver': ['liblinear',␣
 ↪'newton-cg', 'lbfgs', 'sag', 'saga'],
    'classify__max_iter': [1000], # 'classify__max_iter': [100, 500, 1000],
```

```
    #'classify__class_weight': [None],     #'classify__class_weight': [None,␣
 ↪'balanced'],
    "vect__stop_words": [list(stop_words_nltk), list(stop_words_sklearn),␣
 ↪list(stop_words_library)],
    "selecter__k":[5000],
    #"vect__tokenizer": [LemmaTokenizer_word()]
}




#stop_words_nltk
#stop_words_sklearn


vectorizer = CountVectorizer()
selecter = SelectKBest(chi2)
model = LogisticRegression()



pipe = Pipeline(
     [("vect", vectorizer),("selecter", selecter),("classify",model)]
)

grid = model_selection.GridSearchCV(pipe, pipe_params, verbose=1, n_jobs=-1)

grid.fit(train_x, train_y)

t_end = time.time()

elapsed_time = t_end-t_start
accuracy = round(grid.best_score_ * 100,3)

print(f"The best accuracy is {accuracy}.")
print(f"The winning parameters are {grid.best_params_}")
print(f"Run time: {elapsed_time} seconds")
```

```
Fitting 5 folds for each of 3 candidates, totalling 15 fits
The best accuracy is 93.037.
The winning parameters are {'classify__C': 10.0, 'classify__max_iter': 1000,
'classify__penalty': 'l2', 'classify__solver': 'sag', 'selecter__k': 5000,
'vect__stop_words': ['most', 'through', 'everything', 'had', 'have', 'these',
'did', 'un', 'still', 'anyone', 'her', 'almost', 'mine', "hadn't", 'its', 'one',
"shouldn't", 'thence', 'never', 'your', 'doing', 'out', 'three', 'some', 'due',
'below', 'although', 'wasn', 'made', 'very', 'other', 'what', 'bill', 'am',
'as', 'see', 'cant', 'whose', 'fifty', 'wherein', 'amount', 'twenty', 'nobody',
```

```
'somewhere', "you're", 'hereafter', 'along', 've', 'hence', 'against', 'hadn',
'often', 'noone', 'more', 'fifteen', 'becomes', 'seem', 'mustn', 'ltd', 'upon',
'two', "haven't", 'won', 'among', 'something', "aren't", 'them', 'do', 'then',
'yourselves', 'give', 'onto', "needn't", 'whither', 'under', 'last', "mightn't",
'seems', 'shan', "won't", 'becoming', 'therefore', 'after', 'done', 'i',
'couldnt', 'another', 'put', 'towards', 'myself', "you'd", 'yet', "shan't",
'all', 'be', 'back', 'hers', 'you', 'from', 'on', "wouldn't", 'wherever', 'not',
'y', 'if', 'because', 'become', 'such', 'so', 'an', 'co', 'once', 'move',
'several', 'ourselves', 'even', 'nowhere', 'ours', 'himself', 'toward', 're',
'hasn', 'whence', 'him', 'must', 'meanwhile', 'there', 'four', 'behind',
"doesn't", 'ain', 'whereupon', 'needn', 'anything', 'where', 'together', 'well',
'everyone', 'else', 'none', 'don', 'couldn', 'take', 'should', 'than', 'anyhow',
'might', 'further', 'whatever', 'someone', 'mightn', 'who', 'thereupon',
'across', 'full', 'least', 'throughout', 'twelve', 'haven', 'being', 'namely',
'call', 'isn', 'ever', 'until', 'yours', 'will', 'inc', "hasn't", 'm',
"weren't", 'whoever', 'my', 'down', 'at', 'sometime', 'she', 't', 'herein',
'itself', 'part', 'sixty', 'here', "couldn't", 'he', 'theirs', 'whereas',
'otherwise', 'yourself', 'that', 'again', 'forty', 's', 'always', 'which',
'bottom', 'how', 'can', 'go', 'hereupon', 'since', 'just', 'latterly', 'could',
'hereby', 'll', 'mostly', "you've", 'much', 'seemed', "mustn't", 'was', 'our',
'without', 'beforehand', 'serious', 'via', 'me', 'formerly', 'why', 'enough',
"should've", 'whereafter', 'perhaps', 'sincere', 'five', 'many', 'now',
'thereafter', 'about', 'detail', 'and', 'wouldn', 'cannot', 'having', "didn't",
'it', 'eleven', 'nor', 'cry', 'either', 'thin', 'sometimes', 'seeming', 'we',
'd', 'con', 'same', 'to', 'per', 'his', 'the', 'fire', 'found', 'describe',
'already', 'within', 'whether', 'doesn', 'latter', 'has', 'therein', 'rather',
'of', 'anywhere', 'amongst', 'ten', 'o', 'would', 'front', 'de', 'alone',
'system', 'elsewhere', 'those', 'for', 'thick', 'etc', 'a', 'are', 'find',
'though', 'neither', 'whereby', 'own', 'over', 'only', 'thereby', "don't", 'no',
'whenever', 'themselves', 'also', 'beside', 'nothing', 'thus', 'ie', 'third',
'aren', 'too', 'during', 'off', 'became', 'didn', 'fill', 'indeed', 'please',
'in', 'hasnt', 'hundred', 'afterwards', 'mill', 'name', 'their', 'former',
'but', 'moreover', 'thru', 'however', 'whole', 'been', 'next', 'besides', 'eg',
'side', "you'll", 'first', 'keep', 'somehow', 'weren', 'each', 'nevertheless',
'up', 'is', 'they', 'amoungst', 'any', 'everywhere', 'around', 'empty', "isn't",
'anyway', 'shouldn', "that'll", 'nine', 'beyond', 'while', 'whom', 'were',
'top', "she's", 'interest', 'show', 'get', 'ma', 'less', 'between', 'by',
'does', 'herself', 'few', 'above', 'into', 'with', 'six', 'may', 'except',
'eight', "wasn't", 'others', "it's", 'us', 'both', 'every', 'this', 'when',
'or', 'before']}
Run time: 15.71203327178955 seconds

/usr/local/lib/python3.8/dist-packages/sklearn/linear_model/_sag.py:350:
ConvergenceWarning: The max_iter was reached which means the coef_ did not
converge
  warnings.warn(
```

[ ]:
```
#initial training with stop words

t_start = time.time()

pipe_params = {
    'classify__penalty': ['l2'],   #'classify__penalty': ['l1', 'l2'],
    'classify__C': [10.0],     #'classify__C': [0.01, 0.1, 1.0, 10.0],
    'classify__solver': ['sag'],   #'classify__solver': ['liblinear',␣
 ↪'newton-cg', 'lbfgs', 'sag', 'saga'],
    'classify__max_iter': [1000], # 'classify__max_iter': [100, 500, 1000],
    #'classify__class_weight': ['balanced'],     #'classify__class_weight':␣
 ↪[None, 'balanced'],
    "vect__stop_words": [list(stop_words_nltk), list(stop_words_sklearn),␣
 ↪list(stop_words_library)],
    "selecter__k":[5000,3000],
    "vect__ngram_range":[(1,1)]
}



vectorizer = CountVectorizer()
selecter = SelectKBest(chi2)
model = LogisticRegression()



pipe = Pipeline(
     [("vect", vectorizer),("selecter",␣
 ↪selecter),("normalizer",normalizer),("classify",model)]
)

grid = model_selection.GridSearchCV(pipe, pipe_params, verbose=1, n_jobs=-1)

grid.fit(train_x, train_y)

t_end = time.time()

elapsed_time = t_end-t_start
accuracy = round(grid.best_score_ * 100,3)

print(f"The best accuracy is {accuracy}.")
print(f"The winning parameters are {grid.best_params_}")
print(f"Run time: {elapsed_time} seconds")
```

```
Fitting 5 folds for each of 6 candidates, totalling 30 fits
The best accuracy is 92.339.
The winning parameters are {'classify__C': 10.0, 'classify__max_iter': 1000,
```

'classify__penalty': 'l2', 'classify__solver': 'sag', 'selecter__k': 5000,
'vect__ngram_range': (1, 1), 'vect__stop_words': ['most', 'through',
'everything', 'had', 'have', 'these', 'did', 'un', 'still', 'anyone', 'her',
'almost', 'mine', "hadn't", 'its', 'one', "shouldn't", 'thence', 'never',
'your', 'doing', 'out', 'three', 'some', 'due', 'below', 'although', 'wasn',
'made', 'very', 'other', 'what', 'bill', 'am', 'as', 'see', 'cant', 'whose',
'fifty', 'wherein', 'amount', 'twenty', 'nobody', 'somewhere', "you're",
'hereafter', 'along', 've', 'hence', 'against', 'hadn', 'often', 'noone',
'more', 'fifteen', 'becomes', 'seem', 'mustn', 'ltd', 'upon', 'two', "haven't",
'won', 'among', 'something', "aren't", 'them', 'do', 'then', 'yourselves',
'give', 'onto', "needn't", 'whither', 'under', 'last', "mightn't", 'seems',
'shan', "won't", 'becoming', 'therefore', 'after', 'done', 'i', 'couldnt',
'another', 'put', 'towards', 'myself', "you'd", 'yet', "shan't", 'all', 'be',
'back', 'hers', 'you', 'from', 'on', "wouldn't", 'wherever', 'not', 'y', 'if',
'because', 'become', 'such', 'so', 'an', 'co', 'once', 'move', 'several',
'ourselves', 'even', 'nowhere', 'ours', 'himself', 'toward', 're', 'hasn',
'whence', 'him', 'must', 'meanwhile', 'there', 'four', 'behind', "doesn't",
'ain', 'whereupon', 'needn', 'anything', 'where', 'together', 'well',
'everyone', 'else', 'none', 'don', 'couldn', 'take', 'should', 'than', 'anyhow',
'might', 'further', 'whatever', 'someone', 'mightn', 'who', 'thereupon',
'across', 'full', 'least', 'throughout', 'twelve', 'haven', 'being', 'namely',
'call', 'isn', 'ever', 'until', 'yours', 'will', 'inc', "hasn't", 'm',
"weren't", 'whoever', 'my', 'down', 'at', 'sometime', 'she', 't', 'herein',
'itself', 'part', 'sixty', 'here', "couldn't", 'he', 'theirs', 'whereas',
'otherwise', 'yourself', 'that', 'again', 'forty', 's', 'always', 'which',
'bottom', 'how', 'can', 'go', 'hereupon', 'since', 'just', 'latterly', 'could',
'hereby', 'll', 'mostly', "you've", 'much', 'seemed', "mustn't", 'was', 'our',
'without', 'beforehand', 'serious', 'via', 'me', 'formerly', 'why', 'enough',
"should've", 'whereafter', 'perhaps', 'sincere', 'five', 'many', 'now',
'thereafter', 'about', 'detail', 'and', 'wouldn', 'cannot', 'having', "didn't",
'it', 'eleven', 'nor', 'cry', 'either', 'thin', 'sometimes', 'seeming', 'we',
'd', 'con', 'same', 'to', 'per', 'his', 'the', 'fire', 'found', 'describe',
'already', 'within', 'whether', 'doesn', 'latter', 'has', 'therein', 'rather',
'of', 'anywhere', 'amongst', 'ten', 'o', 'would', 'front', 'de', 'alone',
'system', 'elsewhere', 'those', 'for', 'thick', 'etc', 'a', 'are', 'find',
'though', 'neither', 'whereby', 'own', 'over', 'only', 'thereby', "don't", 'no',
'whenever', 'themselves', 'also', 'beside', 'nothing', 'thus', 'ie', 'third',
'aren', 'too', 'during', 'off', 'became', 'didn', 'fill', 'indeed', 'please',
'in', 'hasnt', 'hundred', 'afterwards', 'mill', 'name', 'their', 'former',
'but', 'moreover', 'thru', 'however', 'whole', 'been', 'next', 'besides', 'eg',
'side', "you'll", 'first', 'keep', 'somehow', 'weren', 'each', 'nevertheless',
'up', 'is', 'they', 'amoungst', 'any', 'everywhere', 'around', 'empty', "isn't",
'anyway', 'shouldn', "that'll", 'nine', 'beyond', 'while', 'whom', 'were',
'top', "she's", 'interest', 'show', 'get', 'ma', 'less', 'between', 'by',
'does', 'herself', 'few', 'above', 'into', 'with', 'six', 'may', 'except',
'eight', "wasn't", 'others', "it's", 'us', 'both', 'every', 'this', 'when',
'or', 'before']}
Run time: 9.341354131698608 seconds

[ ]:
```
#initial training with stop words. 93.038

t_start = time.time()

pipe_params = {
    'classify__penalty': ['l2'],   #'classify__penalty': ['l1', 'l2'],
    'classify__C': [10.0],     #'classify__C': [0.01, 0.1, 1.0, 10.0],
    'classify__solver': ['sag'],   #'classify__solver': ['liblinear',␣
 ↪'newton-cg', 'lbfgs', 'sag', 'saga'],
    'classify__max_iter': [1000], # 'classify__max_iter': [100, 500, 1000],
    'classify__class_weight': [None, 'balanced'],     #'classify__class_weight':
 ↪ [None, 'balanced'],
    "vect__stop_words": [list(stop_words_library)],  ##[list(stop_words_nltk),␣
 ↪list(stop_words_sklearn), list(stop_words_library)]
    "selecter__k":[5000],
    "vect__ngram_range":[(1,1)]

}




#stop_words_nltk
#stop_words_sklearn


vectorizer = CountVectorizer()
selecter = SelectKBest(chi2)
model = LogisticRegression()



pipe = Pipeline(
     [("vect", vectorizer),("selecter", selecter),("classify",model)]
)

grid = model_selection.GridSearchCV(pipe, pipe_params, verbose=1, n_jobs=-1)

grid.fit(train_x, train_y)

t_end = time.time()

elapsed_time = t_end-t_start
accuracy = round(grid.best_score_ * 100,3)

print(f"The best accuracy is {accuracy}.")
print(f"The winning parameters are {grid.best_params_}")
```

```
print(f"Run time: {elapsed_time} seconds")
```

```
Fitting 5 folds for each of 2 candidates, totalling 10 fits
The best accuracy is 92.899.
The winning parameters are {'classify__C': 10.0, 'classify__class_weight': None,
'classify__max_iter': 1000, 'classify__penalty': 'l2', 'classify__solver':
'sag', 'selecter__k': 5000, 'vect__ngram_range': (1, 1), 'vect__stop_words':
['most', 'through', 'everything', 'had', 'have', 'these', 'did', 'un', 'still',
'anyone', 'her', 'almost', 'mine', "hadn't", 'its', 'one', "shouldn't",
'thence', 'never', 'your', 'doing', 'out', 'three', 'some', 'due', 'below',
'although', 'wasn', 'made', 'very', 'other', 'what', 'bill', 'am', 'as', 'see',
'cant', 'whose', 'fifty', 'wherein', 'amount', 'twenty', 'nobody', 'somewhere',
"you're", 'hereafter', 'along', 've', 'hence', 'against', 'hadn', 'often',
'noone', 'more', 'fifteen', 'becomes', 'seem', 'mustn', 'ltd', 'upon', 'two',
"haven't", 'won', 'among', 'something', "aren't", 'them', 'do', 'then',
'yourselves', 'give', 'onto', "needn't", 'whither', 'under', 'last', "mightn't",
'seems', 'shan', "won't", 'becoming', 'therefore', 'after', 'done', 'i',
'couldnt', 'another', 'put', 'towards', 'myself', "you'd", 'yet', "shan't",
'all', 'be', 'back', 'hers', 'you', 'from', 'on', "wouldn't", 'wherever', 'not',
'y', 'if', 'because', 'become', 'such', 'so', 'an', 'co', 'once', 'move',
'several', 'ourselves', 'even', 'nowhere', 'ours', 'himself', 'toward', 're',
'hasn', 'whence', 'him', 'must', 'meanwhile', 'there', 'four', 'behind',
"doesn't", 'ain', 'whereupon', 'needn', 'anything', 'where', 'together', 'well',
'everyone', 'else', 'none', 'don', 'couldn', 'take', 'should', 'than', 'anyhow',
'might', 'further', 'whatever', 'someone', 'mightn', 'who', 'thereupon',
'across', 'full', 'least', 'throughout', 'twelve', 'haven', 'being', 'namely',
'call', 'isn', 'ever', 'until', 'yours', 'will', 'inc', "hasn't", 'm',
"weren't", 'whoever', 'my', 'down', 'at', 'sometime', 'she', 't', 'herein',
'itself', 'part', 'sixty', 'here', "couldn't", 'he', 'theirs', 'whereas',
'otherwise', 'yourself', 'that', 'again', 'forty', 's', 'always', 'which',
'bottom', 'how', 'can', 'go', 'hereupon', 'since', 'just', 'latterly', 'could',
'hereby', 'll', 'mostly', "you've", 'much', 'seemed', "mustn't", 'was', 'our',
'without', 'beforehand', 'serious', 'via', 'me', 'formerly', 'why', 'enough',
"should've", 'whereafter', 'perhaps', 'sincere', 'five', 'many', 'now',
'thereafter', 'about', 'detail', 'and', 'wouldn', 'cannot', 'having', "didn't",
'it', 'eleven', 'nor', 'cry', 'either', 'thin', 'sometimes', 'seeming', 'we',
'd', 'con', 'same', 'to', 'per', 'his', 'the', 'fire', 'found', 'describe',
'already', 'within', 'whether', 'doesn', 'latter', 'has', 'therein', 'rather',
'of', 'anywhere', 'amongst', 'ten', 'o', 'would', 'front', 'de', 'alone',
'system', 'elsewhere', 'those', 'for', 'thick', 'etc', 'a', 'are', 'find',
'though', 'neither', 'whereby', 'own', 'over', 'only', 'thereby', "don't", 'no',
'whenever', 'themselves', 'also', 'beside', 'nothing', 'thus', 'ie', 'third',
'aren', 'too', 'during', 'off', 'became', 'didn', 'fill', 'indeed', 'please',
'in', 'hasnt', 'hundred', 'afterwards', 'mill', 'name', 'their', 'former',
'but', 'moreover', 'thru', 'however', 'whole', 'been', 'next', 'besides', 'eg',
'side', "you'll", 'first', 'keep', 'somehow', 'weren', 'each', 'nevertheless',
'up', 'is', 'they', 'amoungst', 'any', 'everywhere', 'around', 'empty', "isn't",
```

```
'anyway', 'shouldn', "that'll", 'nine', 'beyond', 'while', 'whom', 'were',
'top', "she's", 'interest', 'show', 'get', 'ma', 'less', 'between', 'by',
'does', 'herself', 'few', 'above', 'into', 'with', 'six', 'may', 'except',
'eight', "wasn't", 'others', "it's", 'us', 'both', 'every', 'this', 'when',
'or', 'before']}
Run time: 9.679741621017456 seconds
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/linear_model/_sag.py:350:
ConvergenceWarning: The max_iter was reached which means the coef_ did not
converge
  warnings.warn(
```

[ ]:
```
#initial training with stop words.

t_start = time.time()

pipe_params = {
    'classify__penalty': ['l2'],   #'classify__penalty': ['l1', 'l2'],
    'classify__C': [10.0],     #'classify__C': [0.01, 0.1, 1.0, 10.0],
    'classify__solver': ['sag'],   #'classify__solver': ['liblinear',␣
 ↪'newton-cg', 'lbfgs', 'sag', 'saga'],
    'classify__max_iter': [1000], # 'classify__max_iter': [100, 500, 1000],
    'classify__class_weight': [None, 'balanced'],     #'classify__class_weight':
 ↪ [None, 'balanced'],
    "vect__stop_words": [list(stop_words_nltk), list(stop_words_sklearn),␣
 ↪list(stop_words_library)],  ##[list(stop_words_nltk),␣
 ↪list(stop_words_sklearn), list(stop_words_library)]
    "selecter__k":[5000],
    "vect__ngram_range":[(1,1)],
    # "vect__binary": [False]
    #"vect__preprocessor": [preprocess_text,remove_punctuation,None]
    #"vect__binary": [False]

}



vectorizer = CountVectorizer()
selecter = SelectKBest(chi2)
model = LogisticRegression()
#normalizer = Normalizer()


pipe = Pipeline(
     [("vect", vectorizer),("selecter", selecter),("classify",model)]
)

grid = model_selection.GridSearchCV(pipe, pipe_params, verbose=1, n_jobs=-1)
```

```
grid.fit(train_x, train_y)

t_end = time.time()

elapsed_time = t_end-t_start
accuracy = round(grid.best_score_ * 100,3)

print(f"The best accuracy is {accuracy}.")
print(f"The winning parameters are {grid.best_params_}")
print(f"Run time: {elapsed_time} seconds")
```

```
Fitting 5 folds for each of 6 candidates, totalling 30 fits
The best accuracy is 93.038.
The winning parameters are {'classify__C': 10.0, 'classify__class_weight': None,
'classify__max_iter': 1000, 'classify__penalty': 'l2', 'classify__solver':
'sag', 'selecter__k': 5000, 'vect__ngram_range': (1, 1), 'vect__stop_words':
['most', 'through', 'everything', 'had', 'have', 'these', 'did', 'un', 'still',
'anyone', 'her', 'almost', 'mine', "hadn't", 'its', 'one', "shouldn't",
'thence', 'never', 'your', 'doing', 'out', 'three', 'some', 'due', 'below',
'although', 'wasn', 'made', 'very', 'other', 'what', 'bill', 'am', 'as', 'see',
'cant', 'whose', 'fifty', 'wherein', 'amount', 'twenty', 'nobody', 'somewhere',
"you're", 'hereafter', 'along', 've', 'hence', 'against', 'hadn', 'often',
'noone', 'more', 'fifteen', 'becomes', 'seem', 'mustn', 'ltd', 'upon', 'two',
"haven't", 'won', 'among', 'something', "aren't", 'them', 'do', 'then',
'yourselves', 'give', 'onto', "needn't", 'whither', 'under', 'last', "mightn't",
'seems', 'shan', "won't", 'becoming', 'therefore', 'after', 'done', 'i',
'couldnt', 'another', 'put', 'towards', 'myself', "you'd", 'yet', "shan't",
'all', 'be', 'back', 'hers', 'you', 'from', 'on', "wouldn't", 'wherever', 'not',
'y', 'if', 'because', 'become', 'such', 'so', 'an', 'co', 'once', 'move',
'several', 'ourselves', 'even', 'nowhere', 'ours', 'himself', 'toward', 're',
'hasn', 'whence', 'him', 'must', 'meanwhile', 'there', 'four', 'behind',
"doesn't", 'ain', 'whereupon', 'needn', 'anything', 'where', 'together', 'well',
'everyone', 'else', 'none', 'don', 'couldn', 'take', 'should', 'than', 'anyhow',
'might', 'further', 'whatever', 'someone', 'mightn', 'who', 'thereupon',
'across', 'full', 'least', 'throughout', 'twelve', 'haven', 'being', 'namely',
'call', 'isn', 'ever', 'until', 'yours', 'will', 'inc', "hasn't", 'm',
"weren't", 'whoever', 'my', 'down', 'at', 'sometime', 'she', 't', 'herein',
'itself', 'part', 'sixty', 'here', "couldn't", 'he', 'theirs', 'whereas',
'otherwise', 'yourself', 'that', 'again', 'forty', 's', 'always', 'which',
'bottom', 'how', 'can', 'go', 'hereupon', 'since', 'just', 'latterly', 'could',
'hereby', 'll', 'mostly', "you've", 'much', 'seemed', "mustn't", 'was', 'our',
'without', 'beforehand', 'serious', 'via', 'me', 'formerly', 'why', 'enough',
"should've", 'whereafter', 'perhaps', 'sincere', 'five', 'many', 'now',
'thereafter', 'about', 'detail', 'and', 'wouldn', 'cannot', 'having', "didn't",
'it', 'eleven', 'nor', 'cry', 'either', 'thin', 'sometimes', 'seeming', 'we',
'd', 'con', 'same', 'to', 'per', 'his', 'the', 'fire', 'found', 'describe',
```

```
'already', 'within', 'whether', 'doesn', 'latter', 'has', 'therein', 'rather',
'of', 'anywhere', 'amongst', 'ten', 'o', 'would', 'front', 'de', 'alone',
'system', 'elsewhere', 'those', 'for', 'thick', 'etc', 'a', 'are', 'find',
'though', 'neither', 'whereby', 'own', 'over', 'only', 'thereby', "don't", 'no',
'whenever', 'themselves', 'also', 'beside', 'nothing', 'thus', 'ie', 'third',
'aren', 'too', 'during', 'off', 'became', 'didn', 'fill', 'indeed', 'please',
'in', 'hasnt', 'hundred', 'afterwards', 'mill', 'name', 'their', 'former',
'but', 'moreover', 'thru', 'however', 'whole', 'been', 'next', 'besides', 'eg',
'side', "you'll", 'first', 'keep', 'somehow', 'weren', 'each', 'nevertheless',
'up', 'is', 'they', 'amoungst', 'any', 'everywhere', 'around', 'empty', "isn't",
'anyway', 'shouldn', "that'll", 'nine', 'beyond', 'while', 'whom', 'were',
'top', "she's", 'interest', 'show', 'get', 'ma', 'less', 'between', 'by',
'does', 'herself', 'few', 'above', 'into', 'with', 'six', 'may', 'except',
'eight', "wasn't", 'others', "it's", 'us', 'both', 'every', 'this', 'when',
'or', 'before']}
Run time: 30.26263689994812 seconds
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/linear_model/_sag.py:350:
ConvergenceWarning: The max_iter was reached which means the coef_ did not
converge
  warnings.warn(
```

[ ]:
```
#initial training with stop words. 93.038

t_start = time.time()

pipe_params = {
    'classify__penalty': ['l2'],   #'classify__penalty': ['l1', 'l2'],
    'classify__C': [10.0],     #'classify__C': [0.01, 0.1, 1.0, 10.0],
    'classify__solver': ['sag'],   #'classify__solver': ['liblinear',␣
 ↪'newton-cg', 'lbfgs', 'sag', 'saga'],
    'classify__max_iter': [1000], # 'classify__max_iter': [100, 500, 1000],
    'classify__class_weight': [None, 'balanced'],     #'classify__class_weight':
 ↪ [None, 'balanced'],
    "vect__stop_words": [list(stop_words_nltk), list(stop_words_sklearn),␣
 ↪list(stop_words_library),list(stop_words_library)], ␣
 ↪##[list(stop_words_nltk), list(stop_words_sklearn), list(stop_words_library)]
    "selecter__k":[5000],
    "vect__ngram_range":[(1,1)],
    # "vect__binary": [False]
    "vect__preprocessor": [preprocess_text,remove_punctuation,None]
    #"vect__binary": [False]

}



vectorizer = CountVectorizer()
```

```
selecter = SelectKBest(chi2)
model = LogisticRegression()
#normalizer = Normalizer()


pipe = Pipeline(
     [("vect", vectorizer),("selecter", selecter),("classify",model)]
)

grid = model_selection.GridSearchCV(pipe, pipe_params, verbose=1, n_jobs=-1)

grid.fit(train_x, train_y)

t_end = time.time()

elapsed_time = t_end-t_start
accuracy = round(grid.best_score_ * 100,3)

print(f"The best accuracy is {accuracy}.")
print(f"The winning parameters are {grid.best_params_}")
print(f"Run time: {elapsed_time} seconds")
```

```
Fitting 5 folds for each of 24 candidates, totalling 120 fits
The best accuracy is 93.038.
The winning parameters are {'classify__C': 10.0, 'classify__class_weight': None,
'classify__max_iter': 1000, 'classify__penalty': 'l2', 'classify__solver':
'sag', 'selecter__k': 5000, 'vect__ngram_range': (1, 1), 'vect__preprocessor':
None, 'vect__stop_words': ['most', 'through', 'everything', 'had', 'have',
'these', 'did', 'un', 'still', 'anyone', 'her', 'almost', 'mine', "hadn't",
'its', 'one', "shouldn't", 'thence', 'never', 'your', 'doing', 'out', 'three',
'some', 'due', 'below', 'although', 'wasn', 'made', 'very', 'other', 'what',
'bill', 'am', 'as', 'see', 'cant', 'whose', 'fifty', 'wherein', 'amount',
'twenty', 'nobody', 'somewhere', "you're", 'hereafter', 'along', 've', 'hence',
'against', 'hadn', 'often', 'noone', 'more', 'fifteen', 'becomes', 'seem',
'mustn', 'ltd', 'upon', 'two', "haven't", 'won', 'among', 'something', "aren't",
'them', 'do', 'then', 'yourselves', 'give', 'onto', "needn't", 'whither',
'under', 'last', "mightn't", 'seems', 'shan', "won't", 'becoming', 'therefore',
'after', 'done', 'i', 'couldnt', 'another', 'put', 'towards', 'myself', "you'd",
'yet', "shan't", 'all', 'be', 'back', 'hers', 'you', 'from', 'on', "wouldn't",
'wherever', 'not', 'y', 'if', 'because', 'become', 'such', 'so', 'an', 'co',
'once', 'move', 'several', 'ourselves', 'even', 'nowhere', 'ours', 'himself',
'toward', 're', 'hasn', 'whence', 'him', 'must', 'meanwhile', 'there', 'four',
'behind', "doesn't", 'ain', 'whereupon', 'needn', 'anything', 'where',
'together', 'well', 'everyone', 'else', 'none', 'don', 'couldn', 'take',
'should', 'than', 'anyhow', 'might', 'further', 'whatever', 'someone', 'mightn',
'who', 'thereupon', 'across', 'full', 'least', 'throughout', 'twelve', 'haven',
'being', 'namely', 'call', 'isn', 'ever', 'until', 'yours', 'will', 'inc',
```

```
"hasn't", 'm', "weren't", 'whoever', 'my', 'down', 'at', 'sometime', 'she', 't',
'herein', 'itself', 'part', 'sixty', 'here', "couldn't", 'he', 'theirs',
'whereas', 'otherwise', 'yourself', 'that', 'again', 'forty', 's', 'always',
'which', 'bottom', 'how', 'can', 'go', 'hereupon', 'since', 'just', 'latterly',
'could', 'hereby', 'll', 'mostly', "you've", 'much', 'seemed', "mustn't", 'was',
'our', 'without', 'beforehand', 'serious', 'via', 'me', 'formerly', 'why',
'enough', "should've", 'whereafter', 'perhaps', 'sincere', 'five', 'many',
'now', 'thereafter', 'about', 'detail', 'and', 'wouldn', 'cannot', 'having',
"didn't", 'it', 'eleven', 'nor', 'cry', 'either', 'thin', 'sometimes',
'seeming', 'we', 'd', 'con', 'same', 'to', 'per', 'his', 'the', 'fire', 'found',
'describe', 'already', 'within', 'whether', 'doesn', 'latter', 'has', 'therein',
'rather', 'of', 'anywhere', 'amongst', 'ten', 'o', 'would', 'front', 'de',
'alone', 'system', 'elsewhere', 'those', 'for', 'thick', 'etc', 'a', 'are',
'find', 'though', 'neither', 'whereby', 'own', 'over', 'only', 'thereby',
"don't", 'no', 'whenever', 'themselves', 'also', 'beside', 'nothing', 'thus',
'ie', 'third', 'aren', 'too', 'during', 'off', 'became', 'didn', 'fill',
'indeed', 'please', 'in', 'hasnt', 'hundred', 'afterwards', 'mill', 'name',
'their', 'former', 'but', 'moreover', 'thru', 'however', 'whole', 'been',
'next', 'besides', 'eg', 'side', "you'll", 'first', 'keep', 'somehow', 'weren',
'each', 'nevertheless', 'up', 'is', 'they', 'amoungst', 'any', 'everywhere',
'around', 'empty', "isn't", 'anyway', 'shouldn', "that'll", 'nine', 'beyond',
'while', 'whom', 'were', 'top', "she's", 'interest', 'show', 'get', 'ma',
'less', 'between', 'by', 'does', 'herself', 'few', 'above', 'into', 'with',
'six', 'may', 'except', 'eight', "wasn't", 'others', "it's", 'us', 'both',
'every', 'this', 'when', 'or', 'before']}
Run time: 114.15120077133179 seconds
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/linear_model/_sag.py:350:
ConvergenceWarning: The max_iter was reached which means the coef_ did not
converge
  warnings.warn(
```

[ ]:
```python
#initial training with stop words. 93.038

t_start = time.time()

pipe_params = {
    'classify__penalty': ['l2'],   #'classify__penalty': ['l1', 'l2'],
    'classify__C': [10.0],     #'classify__C': [0.01, 0.1, 1.0, 10.0],
    'classify__solver': ['sag'],   #'classify__solver': ['liblinear',␣
 ↪'newton-cg', 'lbfgs', 'sag', 'saga'],
    'classify__max_iter': [1000], # 'classify__max_iter': [100, 500, 1000],
    'classify__class_weight': [None, 'balanced'],     #'classify__class_weight':
 ↪ [None, 'balanced'],
    "vect__stop_words": [list(stop_words_nltk), list(stop_words_sklearn),␣
 ↪list(stop_words_library),list(stop_words_library)], ␣
 ↪##[list(stop_words_nltk), list(stop_words_sklearn), list(stop_words_library)]
    "selecter__k":[5000],
```

```
    "vect__ngram_range":[(1,1)],
    # "vect__binary": [False]
    "vect__preprocessor": [preprocess_text,remove_punctuation,None]

}



vectorizer = CountVectorizer()
selecter = SelectKBest(chi2)
model = LogisticRegression()
#normalizer = Normalizer()


pipe = Pipeline(
     [("vect", vectorizer),("selecter", selecter),("classify",model)]
)

grid = model_selection.GridSearchCV(pipe, pipe_params, verbose=1, n_jobs=-1)

grid.fit(train_x, train_y)

t_end = time.time()

elapsed_time = t_end-t_start
accuracy = round(grid.best_score_ * 100,3)

print(f"The best accuracy is {accuracy}.")
print(f"The winning parameters are {grid.best_params_}")
print(f"Run time: {elapsed_time} seconds")



y_pred = grid.predict(test_x)
create_test_csv(y_pred,"LogisticReg.csv")
```

```
Fitting 5 folds for each of 24 candidates, totalling 120 fits
The best accuracy is 93.037.
The winning parameters are {'classify__C': 10.0, 'classify__class_weight':
'balanced', 'classify__max_iter': 1000, 'classify__penalty': 'l2',
'classify__solver': 'sag', 'selecter__k': 5000, 'vect__ngram_range': (1, 1),
'vect__preprocessor': None, 'vect__stop_words': ['see', 'fifty', 'several',
'much', 'yet', 'often', "isn't", "shan't", 'further', 'of', 'together', 'and',
'd', "needn't", 'cannot', "aren't", 'eight', 'across', 'anything', "hadn't",
'con', 'theirs', 'once', 'anyhow', 'twelve', 'those', 'full', 'itself', 'only',
'o', 've', 'in', 'why', 'haven', 'same', 'them', 'give', 'sometime', 'behind',
'enough', 'couldnt', 'becoming', 'already', 'everywhere', 'third', 'hereupon',
```

```
'interest', 'just', 'through', 'without', 'except', 'un', 'another', 'but',
'least', 'somewhere', 're', 'perhaps', 'made', 'co', 'hasn', 'mightn', 'didn',
'onto', 'should', 'cant', 'into', 'whatever', 'from', 'since', 'six',
'wherever', 'having', 'everything', 'ltd', 'as', 'because', 'under', 'or',
'hence', 'meanwhile', 'yourself', 'bottom', 'can', 'nine', 'anyway', "mightn't",
'him', 'wasn', 'everyone', 'rather', "it's", 'becomes', 'cry', 'do', 'ever',
'hundred', 'become', 'on', 'anyone', 'then', 'most', "you've", 'will', 'keep',
'else', "haven't", 'whoever', 'being', 'during', "that'll", "she's", 'yours',
'they', 'five', 'whenever', 'seemed', 'did', 'therefore', 'get', 'call', 'up',
'ten', 'your', 'last', 'to', 'seeming', 'every', 'along', 'is', 'be', 'the',
'all', 'either', 'myself', 'never', "you'd", 'doesn', 'who', "won't",
'amoungst', 'thereby', "don't", 'whereafter', 'beyond', 'are', 'thence', 'show',
'although', 'latter', 'thereupon', 'twenty', 'something', 'his', 'side', 'had',
'somehow', 'their', 'nowhere', 'whereupon', 'ie', 'fifteen', 's', "shouldn't",
'over', 'after', 'out', 'sincere', 'someone', 'fire', 'each', 't', 'beside',
'etc', 'some', 'nobody', 'shan', "should've", 'other', 'about', 'two', 'have',
'done', 'we', 'put', 'one', 'move', 'nothing', 'more', 'yourselves', 'others',
'll', 'among', 'whereby', 'three', 'toward', 'whose', 'an', 'herself',
'towards', "you'll", 'might', 'whom', 'isn', 'these', 'though', 'whether', 'no',
'back', 'ain', 'even', 'herein', 'both', 'hereafter', 'am', 'whence', 'whereas',
'bill', 'name', 'part', 'such', 'it', 'wouldn', 'down', 'thereafter', 'if',
'she', 'don', "didn't", 'now', 'won', 'besides', 'me', 'own', 'her', 'a',
"wouldn't", 'hasnt', 'nevertheless', 'nor', 'ours', 'fill', 'he', 'does',
'there', 'between', 'take', 'again', 'not', 'please', 'four', 'almost', 'thick',
'while', 'us', 'alone', 'serious', "couldn't", 'throughout', 'top', 'could',
'therein', 'noone', 'forty', 'than', 'first', 'de', 'mine', 'latterly', 'any',
'himself', 'also', 'go', 'amount', 'wherein', 'namely', 'were', 'neither',
'find', 'has', 'before', 'at', 'less', 'may', 'elsewhere', 'couldn', 'above',
'per', 'seems', 'many', 'whole', 'still', 'been', 'so', 'around', "mustn't",
'themselves', 'here', 'hereby', 'few', 'off', 'formerly', 'thru', 'sometimes',
'was', 'i', 'eg', 'via', 'well', 'ma', 'empty', 'describe', 'mostly', 'by',
'within', 'with', 'whither', "wasn't", 'my', 'doing', 'eleven', 'for', 'upon',
'became', 'moreover', 'thin', 'would', 'below', 'always', 'former', 'mill',
'afterwards', 'too', 'seem', 'amongst', 'anywhere', 'front', 'hadn', 'needn',
'due', 'detail', 'what', 'which', 'y', 'against', 'next', 'otherwise', "you're",
'hers', 'very', 'aren', "hasn't", 'that', 'm', 'however', 'weren', 'sixty',
"weren't", 'when', 'beforehand', 'ourselves', 'where', 'you', 'indeed',
'system', "doesn't", 'inc', 'shouldn', 'thus', 'until', 'how', 'its', 'mustn',
'found', 'this', 'none', 'our', 'must']}
Run time: 110.83977627754211 seconds
File saved.

/usr/local/lib/python3.8/dist-packages/sklearn/linear_model/_sag.py:350:
ConvergenceWarning: The max_iter was reached which means the coef_ did not
converge
  warnings.warn(
```

[ ]:
```
#initial training with stop words. 93.038

t_start = time.time()

pipe_params = {
    'classify__penalty': ['l2'],   #'classify__penalty': ['l1', 'l2'],
    'classify__C': [10.0],     #'classify__C': [0.01, 0.1, 1.0, 10.0],
    'classify__solver': ['sag'],   #'classify__solver': ['liblinear',␣
 ↪'newton-cg', 'lbfgs', 'sag', 'saga'],
    'classify__max_iter': [10000], # 'classify__max_iter': [100, 500, 1000],
    'classify__class_weight': [None, 'balanced'],     #'classify__class_weight':
 ↪ [None, 'balanced'],
    "vect__stop_words": [list(stop_words_nltk), list(stop_words_sklearn),␣
 ↪list(stop_words_library),list(stop_words_library)], ␣
 ↪##[list(stop_words_nltk), list(stop_words_sklearn), list(stop_words_library)]
    "selecter__k":[5000],
    "vect__ngram_range":[(1,1)],
    "vect__binary": [False],


}



vectorizer = CountVectorizer()
selecter = SelectKBest(chi2)
model = LogisticRegression()
#normalizer = Normalizer()


pipe = Pipeline(
     [("vect", vectorizer),("selecter", selecter),("classify",model)]
)

grid = model_selection.GridSearchCV(pipe, pipe_params, verbose=1, n_jobs=-1)

grid.fit(train_x, train_y)

t_end = time.time()

elapsed_time = t_end-t_start
accuracy = round(grid.best_score_ * 100,3)

print(f"The best accuracy is {accuracy}.")
print(f"The winning parameters are {grid.best_params_}")
print(f"Run time: {elapsed_time} seconds")
```

```
#y_pred = grid.predict(test_x)
#create_test_csv(y_pred,"LogisticReg.csv")
```

```
Fitting 5 folds for each of 8 candidates, totalling 40 fits
The best accuracy is 92.481.
The winning parameters are {'classify__C': 10.0, 'classify__class_weight': None,
'classify__max_iter': 10000, 'classify__penalty': 'l2', 'classify__solver':
'sag', 'selecter__k': 5000, 'vect__binary': False, 'vect__ngram_range': (1, 1),
'vect__stop_words': ['see', 'fifty', 'several', 'much', 'yet', 'often', "isn't",
"shan't", 'further', 'of', 'together', 'and', 'd', "needn't", 'cannot',
"aren't", 'eight', 'across', 'anything', "hadn't", 'con', 'theirs', 'once',
'anyhow', 'twelve', 'those', 'full', 'itself', 'only', 'o', 've', 'in', 'why',
'haven', 'same', 'them', 'give', 'sometime', 'behind', 'enough', 'couldnt',
'becoming', 'already', 'everywhere', 'third', 'hereupon', 'interest', 'just',
'through', 'without', 'except', 'un', 'another', 'but', 'least', 'somewhere',
're', 'perhaps', 'made', 'co', 'hasn', 'mightn', 'didn', 'onto', 'should',
'cant', 'into', 'whatever', 'from', 'since', 'six', 'wherever', 'having',
'everything', 'ltd', 'as', 'because', 'under', 'or', 'hence', 'meanwhile',
'yourself', 'bottom', 'can', 'nine', 'anyway', "mightn't", 'him', 'wasn',
'everyone', 'rather', "it's", 'becomes', 'cry', 'do', 'ever', 'hundred',
'become', 'on', 'anyone', 'then', 'most', "you've", 'will', 'keep', 'else',
"haven't", 'whoever', 'being', 'during', "that'll", "she's", 'yours', 'they',
'five', 'whenever', 'seemed', 'did', 'therefore', 'get', 'call', 'up', 'ten',
'your', 'last', 'to', 'seeming', 'every', 'along', 'is', 'be', 'the', 'all',
'either', 'myself', 'never', "you'd", 'doesn', 'who', "won't", 'amoungst',
'thereby', "don't", 'whereafter', 'beyond', 'are', 'thence', 'show', 'although',
'latter', 'thereupon', 'twenty', 'something', 'his', 'side', 'had', 'somehow',
'their', 'nowhere', 'whereupon', 'ie', 'fifteen', 's', "shouldn't", 'over',
'after', 'out', 'sincere', 'someone', 'fire', 'each', 't', 'beside', 'etc',
'some', 'nobody', 'shan', "should've", 'other', 'about', 'two', 'have', 'done',
'we', 'put', 'one', 'move', 'nothing', 'more', 'yourselves', 'others', 'll',
'among', 'whereby', 'three', 'toward', 'whose', 'an', 'herself', 'towards',
"you'll", 'might', 'whom', 'isn', 'these', 'though', 'whether', 'no', 'back',
'ain', 'even', 'herein', 'both', 'hereafter', 'am', 'whence', 'whereas', 'bill',
'name', 'part', 'such', 'it', 'wouldn', 'down', 'thereafter', 'if', 'she',
'don', "didn't", 'now', 'won', 'besides', 'me', 'own', 'her', 'a', "wouldn't",
'hasnt', 'nevertheless', 'nor', 'ours', 'fill', 'he', 'does', 'there',
'between', 'take', 'again', 'not', 'please', 'four', 'almost', 'thick', 'while',
'us', 'alone', 'serious', "couldn't", 'throughout', 'top', 'could', 'therein',
'noone', 'forty', 'than', 'first', 'de', 'mine', 'latterly', 'any', 'himself',
'also', 'go', 'amount', 'wherein', 'namely', 'were', 'neither', 'find', 'has',
'before', 'at', 'less', 'may', 'elsewhere', 'couldn', 'above', 'per', 'seems',
'many', 'whole', 'still', 'been', 'so', 'around', "mustn't", 'themselves',
```

```
'here', 'hereby', 'few', 'off', 'formerly', 'thru', 'sometimes', 'was', 'i',
'eg', 'via', 'well', 'ma', 'empty', 'describe', 'mostly', 'by', 'within',
'with', 'whither', "wasn't", 'my', 'doing', 'eleven', 'for', 'upon', 'became',
'moreover', 'thin', 'would', 'below', 'always', 'former', 'mill', 'afterwards',
'too', 'seem', 'amongst', 'anywhere', 'front', 'hadn', 'needn', 'due', 'detail',
'what', 'which', 'y', 'against', 'next', 'otherwise', "you're", 'hers', 'very',
'aren', "hasn't", 'that', 'm', 'however', 'weren', 'sixty', "weren't", 'when',
'beforehand', 'ourselves', 'where', 'you', 'indeed', 'system', "doesn't", 'inc',
'shouldn', 'thus', 'until', 'how', 'its', 'mustn', 'found', 'this', 'none',
'our', 'must']}
Run time: 67.39258456230164 seconds
```