DesicionTree

March 12, 2023

```
[ ]: 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
     from sklearn.tree import DecisionTreeClassifier
     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')
     from sklearn.svm import SVC
```

```
[nltk_data] Downloading package omw-1.4 to /root/nltk_data…
[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 wordnet to /root/nltk_data…
[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]   Unzipping corpora/stopwords.zip.
```

[ ]:
```
#import the data
drive.mount('/content/gdrive/', force_remount=True)

train_data_initial = pd.read_csv('/content/gdrive/MyDrive/ecse551-mp2/train.
 ↪csv')
test_data = pd.read_csv('/content/gdrive/MyDrive/ecse551-mp2/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 = train_data["body"]
train_y = train_data["subreddit"]
test_x = test_data["body"]
```

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

[ ]:
```
##########################################################
```

[ ]:
```
#initial training of DecisionTree
t_start = time.time()

pipe_params = {
    'clf__criterion': ['gini', 'entropy'],
    'clf__max_depth': [10, 50, 100, None],
    'clf__min_samples_split': [2, 5, 10],
    'clf__min_samples_leaf': [1, 2, 4]
}

vectorizer = CountVectorizer()
model = DecisionTreeClassifier()

pipe = Pipeline(
    [("vect", vectorizer),("clf",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)
```

```python
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 72 candidates, totalling 360 fits
The best accuracy is 86.072.
The winning parameters are {'clf__criterion': 'entropy', 'clf__max_depth': 50,
'clf__min_samples_leaf': 4, 'clf__min_samples_split': 5}
Run time: 45.189074993133545 seconds
```

[ ]:
```python
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
```

```
[ ]: #testing stop words
t_start = time.time()

pipe_params = {
    'clf__criterion': ['gini', 'entropy'],
    'vect__stop_words': [stop_words_library],
    'clf__max_depth': [10, 50, 100, None],
    'clf__min_samples_split': [2, 5],
    'clf__min_samples_leaf': [1, 2, 4],

}

vectorizer = CountVectorizer()
model = DecisionTreeClassifier()

pipe = Pipeline(
    [("vect", vectorizer),("clf",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 88.305.
The winning parameters are {'clf__criterion': 'entropy', 'clf__max_depth': 100,
'clf__min_samples_leaf': 1, 'clf__min_samples_split': 2, 'vect__stop_words':
['to', 'made', 'his', 'sometime', 'those', 'except', 'no', 'which', 'shan',
'detail', 'when', 'con', 'haven', "needn't", 'take', 'ever', 'would', "shan't",
'side', 'thick', 'thereby', 'hence', 'third', 'ours', 'six', 'she', 'then',
'seeming', 'therein', 'front', 'show', 'below', 'amoungst', 'none', "couldn't",
'keep', 'bottom', 'hereby', 'wherein', 'latterly', 'we', 'upon', 'must', 'once',
'more', 'therefore', 'doesn', 'same', 'seems', 'sincere', 'last', 'whence',
'inc', 'about', "she's", 'needn', 'just', 'somewhere', "don't", 'anything',
'empty', 'via', 'often', 'me', 'whole', 'together', 'call', 'whoever',
'everything', 'former', 'am', 'others', 'elsewhere', 'may', 'thereafter',
"you've", 'seem', "wouldn't", 'himself', 'both', 'ltd', 'because', 'hasnt',
'hereupon', 'even', 'herein', 'could', 'among', 'part', 'should', 'system',
'hadn', 'under', 'yourselves', "wasn't", 'fill', 'who', 'a', 'mill', 'whose',
'whether', 'nothing', 'hereafter', 'any', 'are', 'here', 'nor', 'anyhow',
"didn't", 'ma', 'as', 'before', 'd', 'many', 'cant', 'own', 'whereafter',
'otherwise', 'not', 'my', 'hundred', 'against', 'what', 'two', 'twelve', 'onto',
'eg', 'serious', "you're", 'of', 'well', 'the', 'us', 'your', 'he', 'thereupon',
'along', "hadn't", 'etc', 'whereby', 'been', 'from', 'theirs', 've', 'mustn',
'though', 'ten', 'in', 'never', "you'd", 'least', 'having', 'every', 'with',
'without', 'beforehand', 'i', 'across', 'full', 'couldnt', 'somehow', 'again',
'how', 'becomes', 'you', 'four', 'five', 'until', 'y', 'might', 'll', 'mostly',
'co', 'beside', 'mightn', 'wasn', "mightn't", 'other', 'already', 'her', "it's",
"isn't", 'always', 'by', 'indeed', 'around', 'either', 'noone', 'although',
'beyond', 'or', 'yours', 'is', 'cannot', 'shouldn', 'cry', 'o', 'for', 'only',
'mine', 'out', 'isn', 'less', "mustn't", 'fire', 'fifteen', 'nowhere', 'into',
'nevertheless', 'seemed', 'moreover', 'now', "won't", 'each', 'why', 'eleven',
'at', 'very', 'besides', 'amongst', 'didn', 'don', 'couldn', 'describe',
'ourselves', 'be', 'thin', 'such', 'during', 'someone', "you'll", 'on',
'thence', 'herself', 'rather', 'twenty', 'get', 'toward', 'everyone', 'won',
'too', 'else', 'wouldn', 'if', 'find', 'also', 'eight', 'whereas', "that'll",
'bill', 'itself', 'since', 'sometimes', 'these', 'this', 'off', 'interest',
'where', 'above', 'alone', 'up', "doesn't", 'hers', 'some', 'it', "shouldn't",
'un', 'have', 'anyone', 'please', 'fifty', "weren't", 'all', 'neither', 'they',
'afterwards', "aren't", 'wherever', 'becoming', 'and', 'whereupon', 's', 'name',
'that', 'something', 'has', 'enough', 'further', 'an', 'meanwhile', 'will',
'next', 'thru', 'another', 'perhaps', 'still', 'forty', 'one', 'whatever',
'doing', 'being', 'several', 'sixty', 'everywhere', 'move', 'yourself', 'per',
'whither', 'put', 'give', 'nobody', 'than', 'however', 'aren', 'through',
'namely', 'can', 'almost', 'latter', 'themselves', 'was', 'anywhere', 'there',
'behind', 'formerly', 'its', 'three', 'much', 'nine', 're', 'does', 'back',
'most', 'between', 'within', 'down', 'de', 'over', 'anyway', 'their', 'were',
'amount', 'weren', "should've", 'while', 'towards', 'ain', 'few', 'hasn', 'but',
'become', 'throughout', 't', 'whenever', 'had', 'top', 'ie', 'myself', 'so',
'see', "haven't", 'yet', 'done', 'after', 'whom', 'first', 'them', "hasn't",
'go', 'found', 'due', 'became', 'm', 'did', 'him', 'do', 'thus', 'our']}
Run time: 30.90384078025818 seconds
```

[ ]:
```python
#testing features
t_start = time.time()

pipe_params = {
    'clf__criterion': ['gini', 'entropy'],
    'vect__stop_words': [stop_words_library],
    'clf__max_depth': [100],
    'clf__min_samples_split': [2, 5],
    'clf__min_samples_leaf': [1, 2, 4],
    'selecter__k':[5000,3000]
}

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

pipe = Pipeline(
    [("vect", vectorizer),("selecter", selecter),("clf",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 88.719.
The winning parameters are {'clf__criterion': 'gini', 'clf__max_depth': 100,
'clf__min_samples_leaf': 1, 'clf__min_samples_split': 2, 'selecter__k': 3000,
'vect__stop_words': ['to', 'made', 'his', 'sometime', 'those', 'except', 'no',
'which', 'shan', 'detail', 'when', 'con', 'haven', "needn't", 'take', 'ever',
'would', "shan't", 'side', 'thick', 'thereby', 'hence', 'third', 'ours', 'six',
'she', 'then', 'seeming', 'therein', 'front', 'show', 'below', 'amoungst',
'none', "couldn't", 'keep', 'bottom', 'hereby', 'wherein', 'latterly', 'we',
'upon', 'must', 'once', 'more', 'therefore', 'doesn', 'same', 'seems',
'sincere', 'last', 'whence', 'inc', 'about', "she's", 'needn', 'just',
'somewhere', "don't", 'anything', 'empty', 'via', 'often', 'me', 'whole',
'together', 'call', 'whoever', 'everything', 'former', 'am', 'others',
'elsewhere', 'may', 'thereafter', "you've", 'seem', "wouldn't", 'himself',
'both', 'ltd', 'because', 'hasnt', 'hereupon', 'even', 'herein', 'could',
'among', 'part', 'should', 'system', 'hadn', 'under', 'yourselves', "wasn't",
'fill', 'who', 'a', 'mill', 'whose', 'whether', 'nothing', 'hereafter', 'any',
'are', 'here', 'nor', 'anyhow', "didn't", 'ma', 'as', 'before', 'd', 'many',
'cant', 'own', 'whereafter', 'otherwise', 'not', 'my', 'hundred', 'against',
'what', 'two', 'twelve', 'onto', 'eg', 'serious', "you're", 'of', 'well', 'the',
'us', 'your', 'he', 'thereupon', 'along', "hadn't", 'etc', 'whereby', 'been',
'from', 'theirs', 've', 'mustn', 'though', 'ten', 'in', 'never', "you'd",
'least', 'having', 'every', 'with', 'without', 'beforehand', 'i', 'across',
'full', 'couldnt', 'somehow', 'again', 'how', 'becomes', 'you', 'four', 'five',
'until', 'y', 'might', 'll', 'mostly', 'co', 'beside', 'mightn', 'wasn',
"mightn't", 'other', 'already', 'her', "it's", "isn't", 'always', 'by',
'indeed', 'around', 'either', 'noone', 'although', 'beyond', 'or', 'yours',
'is', 'cannot', 'shouldn', 'cry', 'o', 'for', 'only', 'mine', 'out', 'isn',
'less', "mustn't", 'fire', 'fifteen', 'nowhere', 'into', 'nevertheless',
'seemed', 'moreover', 'now', "won't", 'each', 'why', 'eleven', 'at', 'very',
'besides', 'amongst', 'didn', 'don', 'couldn', 'describe', 'ourselves', 'be',
'thin', 'such', 'during', 'someone', "you'll", 'on', 'thence', 'herself',
'rather', 'twenty', 'get', 'toward', 'everyone', 'won', 'too', 'else', 'wouldn',
'if', 'find', 'also', 'eight', 'whereas', "that'll", 'bill', 'itself', 'since',
'sometimes', 'these', 'this', 'off', 'interest', 'where', 'above', 'alone',
'up', "doesn't", 'hers', 'some', 'it', "shouldn't", 'un', 'have', 'anyone',
'please', 'fifty', "weren't", 'all', 'neither', 'they', 'afterwards', "aren't",
'wherever', 'becoming', 'and', 'whereupon', 's', 'name', 'that', 'something',
'has', 'enough', 'further', 'an', 'meanwhile', 'will', 'next', 'thru',
'another', 'perhaps', 'still', 'forty', 'one', 'whatever', 'doing', 'being',
'several', 'sixty', 'everywhere', 'move', 'yourself', 'per', 'whither', 'put',
'give', 'nobody', 'than', 'however', 'aren', 'through', 'namely', 'can',
'almost', 'latter', 'themselves', 'was', 'anywhere', 'there', 'behind',
'formerly', 'its', 'three', 'much', 'nine', 're', 'does', 'back', 'most',
'between', 'within', 'down', 'de', 'over', 'anyway', 'their', 'were', 'amount',
'weren', "should've", 'while', 'towards', 'ain', 'few', 'hasn', 'but', 'become',
'throughout', 't', 'whenever', 'had', 'top', 'ie', 'myself', 'so', 'see',
"haven't", 'yet', 'done', 'after', 'whom', 'first', 'them', "hasn't", 'go',
'found', 'due', 'became', 'm', 'did', 'him', 'do', 'thus', 'our']}
Run time: 15.338690280914307 seconds
```

[ ]:
```
#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()]
```

```
[ ]: #testing lemma => slight improvement
t_start = time.time()

pipe_params = {
    'clf__criterion': ['entropy'],
    'vect__stop_words': [stop_words_library],
    'vect__tokenizer': [LemmaTokenizer_word()],
    'clf__max_depth': [100],
    'clf__min_samples_split': [2, 5],
    'clf__min_samples_leaf': [1, 2, 4],
    'selecter__k':[5000,3000]
}

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

pipe = Pipeline(
    [("vect", vectorizer),("selecter", selecter),("clf",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 12 candidates, totalling 60 fits
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/feature_extraction/text.py:528:
UserWarning: The parameter 'token_pattern' will not be used since 'tokenizer' is
```

```
not None'
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/feature_extraction/text.py:409:
UserWarning: Your stop_words may be inconsistent with your preprocessing.
Tokenizing the stop words generated tokens ["'d", "'ll", "'re", "'s", "'ve",
'make', "n't", 'need', 'sha', 'win', 'wo'] not in stop_words.
  warnings.warn(

The best accuracy is 86.073.
The winning parameters are {'clf__criterion': 'entropy', 'clf__max_depth': 100,
'clf__min_samples_leaf': 1, 'clf__min_samples_split': 5, 'selecter__k': 5000,
'vect__stop_words': ['to', 'made', 'his', 'sometime', 'those', 'except', 'no',
'which', 'shan', 'detail', 'when', 'con', 'haven', "needn't", 'take', 'ever',
'would', "shan't", 'side', 'thick', 'thereby', 'hence', 'third', 'ours', 'six',
'she', 'then', 'seeming', 'therein', 'front', 'show', 'below', 'amoungst',
'none', "couldn't", 'keep', 'bottom', 'hereby', 'wherein', 'latterly', 'we',
'upon', 'must', 'once', 'more', 'therefore', 'doesn', 'same', 'seems',
'sincere', 'last', 'whence', 'inc', 'about', "she's", 'needn', 'just',
'somewhere', "don't", 'anything', 'empty', 'via', 'often', 'me', 'whole',
'together', 'call', 'whoever', 'everything', 'former', 'am', 'others',
'elsewhere', 'may', 'thereafter', "you've", 'seem', "wouldn't", 'himself',
'both', 'ltd', 'because', 'hasnt', 'hereupon', 'even', 'herein', 'could',
'among', 'part', 'should', 'system', 'hadn', 'under', 'yourselves', "wasn't",
'fill', 'who', 'a', 'mill', 'whose', 'whether', 'nothing', 'hereafter', 'any',
'are', 'here', 'nor', 'anyhow', "didn't", 'ma', 'as', 'before', 'd', 'many',
'cant', 'own', 'whereafter', 'otherwise', 'not', 'my', 'hundred', 'against',
'what', 'two', 'twelve', 'onto', 'eg', 'serious', "you're", 'of', 'well', 'the',
'us', 'your', 'he', 'thereupon', 'along', "hadn't", 'etc', 'whereby', 'been',
'from', 'theirs', 've', 'mustn', 'though', 'ten', 'in', 'never', "you'd",
'least', 'having', 'every', 'with', 'without', 'beforehand', 'i', 'across',
'full', 'couldnt', 'somehow', 'again', 'how', 'becomes', 'you', 'four', 'five',
'until', 'y', 'might', 'll', 'mostly', 'co', 'beside', 'mightn', 'wasn',
"mightn't", 'other', 'already', 'her', "it's", "isn't", 'always', 'by',
'indeed', 'around', 'either', 'noone', 'although', 'beyond', 'or', 'yours',
'is', 'cannot', 'shouldn', 'cry', 'o', 'for', 'only', 'mine', 'out', 'isn',
'less', "mustn't", 'fire', 'fifteen', 'nowhere', 'into', 'nevertheless',
'seemed', 'moreover', 'now', "won't", 'each', 'why', 'eleven', 'at', 'very',
'besides', 'amongst', 'didn', 'don', 'couldn', 'describe', 'ourselves', 'be',
'thin', 'such', 'during', 'someone', "you'll", 'on', 'thence', 'herself',
'rather', 'twenty', 'get', 'toward', 'everyone', 'won', 'too', 'else', 'wouldn',
'if', 'find', 'also', 'eight', 'whereas', "that'll", 'bill', 'itself', 'since',
'sometimes', 'these', 'this', 'off', 'interest', 'where', 'above', 'alone',
'up', "doesn't", 'hers', 'some', 'it', "shouldn't", 'un', 'have', 'anyone',
'please', 'fifty', "weren't", 'all', 'neither', 'they', 'afterwards', "aren't",
'wherever', 'becoming', 'and', 'whereupon', 's', 'name', 'that', 'something',
'has', 'enough', 'further', 'an', 'meanwhile', 'will', 'next', 'thru',
'another', 'perhaps', 'still', 'forty', 'one', 'whatever', 'doing', 'being',
'several', 'sixty', 'everywhere', 'move', 'yourself', 'per', 'whither', 'put',
'give', 'nobody', 'than', 'however', 'aren', 'through', 'namely', 'can',
'almost', 'latter', 'themselves', 'was', 'anywhere', 'there', 'behind',
'formerly', 'its', 'three', 'much', 'nine', 're', 'does', 'back', 'most',
'between', 'within', 'down', 'de', 'over', 'anyway', 'their', 'were', 'amount',
'weren', "should've", 'while', 'towards', 'ain', 'few', 'hasn', 'but', 'become',
'throughout', 't', 'whenever', 'had', 'top', 'ie', 'myself', 'so', 'see',
"haven't", 'yet', 'done', 'after', 'whom', 'first', 'them', "hasn't", 'go',
'found', 'due', 'became', 'm', 'did', 'him', 'do', 'thus', 'our'],
'vect__tokenizer': <__main__.LemmaTokenizer_word object at 0x7f6d73ed0b80>}
Run time: 81.05935955047607 seconds
```

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

```
[ ]: #testing preprocessor for lowering words and removing numeric values => slight␣
      ↪improvement
     t_start = time.time()

     pipe_params = {
         'clf__criterion': ['entropy'],
         'vect__stop_words': [stop_words_library],
         'vect__tokenizer': [LemmaTokenizer_word()],
         'vect__preprocessor': [preprocess_text],
         'clf__max_depth': [100],
         'clf__min_samples_split': [2, 5],
         'clf__min_samples_leaf': [1, 2, 4],
         'selecter__k':[5000,3000]
     }

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

     pipe = Pipeline(
         [("vect", vectorizer),("selecter", selecter),("clf",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)
```

```python
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 12 candidates, totalling 60 fits

/usr/local/lib/python3.8/dist-packages/sklearn/feature_extraction/text.py:528:
UserWarning: The parameter 'token_pattern' will not be used since 'tokenizer' is
not None'
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/feature_extraction/text.py:409:
UserWarning: Your stop_words may be inconsistent with your preprocessing.
Tokenizing the stop words generated tokens ["'d", "'ll", "'re", "'s", "'ve",
'make', "n't", 'need', 'sha', 'win', 'wo'] not in stop_words.
  warnings.warn(

The best accuracy is 86.632.
The winning parameters are {'clf__criterion': 'entropy', 'clf__max_depth': 100,
'clf__min_samples_leaf': 2, 'clf__min_samples_split': 5, 'selecter__k': 3000,
'vect__preprocessor': <function preprocess_text at 0x7f6d6a362670>,
'vect__stop_words': ['to', 'made', 'his', 'sometime', 'those', 'except', 'no',
'which', 'shan', 'detail', 'when', 'con', 'haven', "needn't", 'take', 'ever',
'would', "shan't", 'side', 'thick', 'thereby', 'hence', 'third', 'ours', 'six',
'she', 'then', 'seeming', 'therein', 'front', 'show', 'below', 'amoungst',
'none', "couldn't", 'keep', 'bottom', 'hereby', 'wherein', 'latterly', 'we',
'upon', 'must', 'once', 'more', 'therefore', 'doesn', 'same', 'seems',
'sincere', 'last', 'whence', 'inc', 'about', "she's", 'needn', 'just',
'somewhere', "don't", 'anything', 'empty', 'via', 'often', 'me', 'whole',
'together', 'call', 'whoever', 'everything', 'former', 'am', 'others',
'elsewhere', 'may', 'thereafter', "you've", 'seem', "wouldn't", 'himself',
'both', 'ltd', 'because', 'hasnt', 'hereupon', 'even', 'herein', 'could',
'among', 'part', 'should', 'system', 'hadn', 'under', 'yourselves', "wasn't",
'fill', 'who', 'a', 'mill', 'whose', 'whether', 'nothing', 'hereafter', 'any',
'are', 'here', 'nor', 'anyhow', "didn't", 'ma', 'as', 'before', 'd', 'many',
'cant', 'own', 'whereafter', 'otherwise', 'not', 'my', 'hundred', 'against',
'what', 'two', 'twelve', 'onto', 'eg', 'serious', "you're", 'of', 'well', 'the',
'us', 'your', 'he', 'thereupon', 'along', "hadn't", 'etc', 'whereby', 'been',
'from', 'theirs', 've', 'mustn', 'though', 'ten', 'in', 'never', "you'd",
'least', 'having', 'every', 'with', 'without', 'beforehand', 'i', 'across',
'full', 'couldnt', 'somehow', 'again', 'how', 'becomes', 'you', 'four', 'five',
'until', 'y', 'might', 'll', 'mostly', 'co', 'beside', 'mightn', 'wasn',
"mightn't", 'other', 'already', 'her', "it's", "isn't", 'always', 'by',
'indeed', 'around', 'either', 'noone', 'although', 'beyond', 'or', 'yours',
'is', 'cannot', 'shouldn', 'cry', 'o', 'for', 'only', 'mine', 'out', 'isn',
'less', "mustn't", 'fire', 'fifteen', 'nowhere', 'into', 'nevertheless',
'seemed', 'moreover', 'now', "won't", 'each', 'why', 'eleven', 'at', 'very',
'besides', 'amongst', 'didn', 'don', 'couldn', 'describe', 'ourselves', 'be',
'thin', 'such', 'during', 'someone', "you'll", 'on', 'thence', 'herself',
'rather', 'twenty', 'get', 'toward', 'everyone', 'won', 'too', 'else', 'wouldn',
'if', 'find', 'also', 'eight', 'whereas', "that'll", 'bill', 'itself', 'since',
'sometimes', 'these', 'this', 'off', 'interest', 'where', 'above', 'alone',
'up', "doesn't", 'hers', 'some', 'it', "shouldn't", 'un', 'have', 'anyone',
'please', 'fifty', "weren't", 'all', 'neither', 'they', 'afterwards', "aren't",
'wherever', 'becoming', 'and', 'whereupon', 's', 'name', 'that', 'something',
'has', 'enough', 'further', 'an', 'meanwhile', 'will', 'next', 'thru',
'another', 'perhaps', 'still', 'forty', 'one', 'whatever', 'doing', 'being',
'several', 'sixty', 'everywhere', 'move', 'yourself', 'per', 'whither', 'put',
'give', 'nobody', 'than', 'however', 'aren', 'through', 'namely', 'can',
'almost', 'latter', 'themselves', 'was', 'anywhere', 'there', 'behind',
'formerly', 'its', 'three', 'much', 'nine', 're', 'does', 'back', 'most',
'between', 'within', 'down', 'de', 'over', 'anyway', 'their', 'were', 'amount',
'weren', "should've", 'while', 'towards', 'ain', 'few', 'hasn', 'but', 'become',
'throughout', 't', 'whenever', 'had', 'top', 'ie', 'myself', 'so', 'see',
"haven't", 'yet', 'done', 'after', 'whom', 'first', 'them', "hasn't", 'go',
'found', 'due', 'became', 'm', 'did', 'him', 'do', 'thus', 'our'],
'vect__tokenizer': <__main__.LemmaTokenizer_word object at 0x7f6d73fcd1c0>}
Run time: 81.01828145980835 seconds
```

[ ]:

```
#testing binary in vectorize
t_start = time.time()

pipe_params = {
    'clf__criterion': ['entropy'],
    'vect__stop_words': [stop_words_library],
    'vect__tokenizer': [LemmaTokenizer_word()],
    'vect__binary': [True,False],
    'vect__preprocessor': [preprocess_text],
    'clf__max_depth': [100],
    'clf__min_samples_split': [2, 5],
    'clf__min_samples_leaf': [1, 2, 4],
    'selecter__k':[5000,3000]
}

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

pipe = Pipeline(
    [("vect", vectorizer),("selecter", selecter),("clf",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
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-23-a3178dcbc5cc> in <module>
     24 grid = model_selection.GridSearchCV(pipe, pipe_params, verbose=1,␣
 ↪n_jobs=-1)
     25
---> 26 grid.fit(train_x, train_y)
     27
     28 t_end = time.time()

/usr/local/lib/python3.8/dist-packages/sklearn/model_selection/_search.py in␣
 ↪fit(self, X, y, groups, **fit_params)
    872             return results
    873
--> 874         self._run_search(evaluate_candidates)
    875
    876         # multimetric is determined here because in the case of a␣
 ↪callable

/usr/local/lib/python3.8/dist-packages/sklearn/model_selection/_search.py in␣
 ↪_run_search(self, evaluate_candidates)
   1386     def _run_search(self, evaluate_candidates):
   1387         """Search all candidates in param_grid"""
-> 1388         evaluate_candidates(ParameterGrid(self.param_grid))
   1389
   1390

/usr/local/lib/python3.8/dist-packages/sklearn/model_selection/_search.py in␣
 ↪evaluate_candidates(candidate_params, cv, more_results)
    819                 )
    820
--> 821             out = parallel(

    822                 delayed(_fit_and_score)(
    823                     clone(base_estimator),
/usr/local/lib/python3.8/dist-packages/sklearn/utils/parallel.py in␣
 ↪__call__(self, iterable)
     61             for delayed_func, args, kwargs in iterable
     62         )
---> 63         return super().__call__(iterable_with_config)
     64 
     65 

/usr/local/lib/python3.8/dist-packages/joblib/parallel.py in __call__(self,␣
 ↪iterable)
   1096 
   1097             with self._backend.retrieval_context():
-> 1098                 self.retrieve()
   1099             # Make sure that we get a last message telling us we are do e
   1100             elapsed_time = time.time() - self._start_time

/usr/local/lib/python3.8/dist-packages/joblib/parallel.py in retrieve(self)
    973             try:
    974                 if getattr(self._backend, 'supports_timeout', False):
--> 975                     self._output.extend(job.get(timeout=self.timeout))
    976                 else:
    977                     self._output.extend(job.get())

/usr/local/lib/python3.8/dist-packages/joblib/_parallel_backends.py in␣
 ↪wrap_future_result(future, timeout)
    565         AsyncResults.get from multiprocessing."""
    566         try:
--> 567             return future.result(timeout=timeout)
    568         except CfTimeoutError as e:
    569             raise TimeoutError from e

/usr/lib/python3.8/concurrent/futures/_base.py in result(self, timeout)
    437                     return self.__get_result()
    438 
--> 439                 self._condition.wait(timeout)
    440 
    441                 if self._state in [CANCELLED, CANCELLED_AND_NOTIFIED]:

/usr/lib/python3.8/threading.py in wait(self, timeout)
    300         try:    # restore state no matter what (e.g., KeyboardInterrupt
    301             if timeout is None:
--> 302                 waiter.acquire()
    303                 gotit = True
    304             else:

KeyboardInterrupt:
```

```
[ ]: #testing normalize => not good
     t_start = time.time()

     pipe_params = {
         'clf__criterion': ['entropy'],
         'vect__stop_words': [stop_words_library],
         'vect__tokenizer': [LemmaTokenizer_word()],
         'vect__binary': [False],
         'vect__preprocessor': [preprocess_text],
         'clf__max_depth': [100],
         'clf__min_samples_split': [2, 5],
         'clf__min_samples_leaf': [1, 2, 4],
         'selecter__k':[5000,3000],
         'normalizer__norm': ['l2','l1',None]
     }

     vectorizer = CountVectorizer()
     selecter = SelectKBest(chi2)
     normalizer = Normalizer()
     model = DecisionTreeClassifier()

     pipe = Pipeline(
         [("vect", vectorizer),("normalizer", normalizer),("selecter",␣
       ↪selecter),("clf",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 36 candidates, totalling 180 fits
```

```
/usr/local/lib/python3.8/dist-
packages/sklearn/model_selection/_validation.py:372: FitFailedWarning:
60 fits failed out of a total of 180.
The score on these train-test partitions for these parameters will be set to
nan.
If these failures are not expected, you can try to debug them by setting
error_score='raise'.
```

```
Below are more details about the failures:
--------------------------------------------------------------------------------
60 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.8/dist-
packages/sklearn/model_selection/_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.8/dist-packages/sklearn/pipeline.py", line 390,
in fit
    Xt = self._fit(X, y, **fit_params_steps)
  File "/usr/local/lib/python3.8/dist-packages/sklearn/pipeline.py", line 348,
in _fit
    X, fitted_transformer = fit_transform_one_cached(
  File "/usr/local/lib/python3.8/dist-packages/joblib/memory.py", line 349, in
__call__
    return self.func(*args, **kwargs)
  File "/usr/local/lib/python3.8/dist-packages/sklearn/pipeline.py", line 893,
in _fit_transform_one
    res = transformer.fit_transform(X, y, **fit_params)
  File "/usr/local/lib/python3.8/dist-packages/sklearn/base.py", line 855, in
fit_transform
    return self.fit(X, y, **fit_params).transform(X)
  File "/usr/local/lib/python3.8/dist-packages/sklearn/preprocessing/_data.py",
line 1955, in transform
    return normalize(X, norm=self.norm, axis=1, copy=copy)
  File "/usr/local/lib/python3.8/dist-packages/sklearn/preprocessing/_data.py",
line 1783, in normalize
    raise ValueError("'%s' is not a supported norm" % norm)
ValueError: 'None' is not a supported norm

  warnings.warn(some_fits_failed_message, FitFailedWarning)
/usr/local/lib/python3.8/dist-packages/sklearn/model_selection/_search.py:969:
UserWarning: One or more of the test scores are non-finite: [0.83984071
0.84119075 0.8356352  0.8412296         nan        nan
 0.83706294 0.83565462 0.83424631 0.83981158        nan        nan
 0.84123932 0.83981158 0.84120047 0.82591298        nan        nan
 0.85374903 0.83844211 0.83008936 0.8342366         nan        nan
 0.85654623 0.84958236 0.85516706 0.84260878        nan        nan
 0.86213092 0.85933372 0.85652681 0.84677545        nan        nan]
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/feature_extraction/text.py:396:
UserWarning: Your stop_words may be inconsistent with your preprocessing.
Tokenizing the stop words generated tokens ["'d", "'ll", "'re", "'s", "'ve",
'make', "n't", 'need', 'sha', 'win', 'wo'] not in stop_words.
  warnings.warn(

The best accuracy is 86.213.
The winning parameters are {'clf__criterion': 'entropy', 'clf__max_depth': 100,
'clf__min_samples_leaf': 4, 'clf__min_samples_split': 5, 'normalizer__norm':
'l2', 'selecter__k': 5000, 'vect__binary': False, 'vect__preprocessor':
<function preprocess_text at 0x7f407144cee0>, 'vect__stop_words':
frozenset({'now', 'along', 'empty', 'don', 'yours', 'well', 'll', 'about',
'four', 'top', 'serious', 'yourselves', 'than', 'both', 'due', 'and', 'into',
'her', 'thereby', 've', 'except', 'see', 'i', 'down', 'ourselves', 'as',
'thick', 'must', 'do', 'she', 'my', 'own', 'us', 'thus', 'very', 'of', 'wasn',
'your', 'its', 'he', 'former', 'yet', 'almost', 'wherever', 'any', 'had',
'that', 'an', 'itself', "shan't", 'nine', 'besides', 'some', 'whereafter',
'who', 'haven', 'thence', 'namely', 'would', 'everything', 'others', 'seems',
'ain', 'ma', 'rather', "aren't", 'while', "mightn't", 'needn', "you'll",
'beyond', "wouldn't", 'five', 'them', 'thru', 'several', 'two', 'name',
'bottom', "couldn't", 'per', 'most', "doesn't", 'ltd', 'give', "wasn't",
'without', 'get', 'ten', "don't", 'couldn', 'hasn', 'made', 'or', "weren't",
"hadn't", 'how', 'found', 'anyhow', 'against', 'myself', 'to', 'always',
"won't", 'here', 'has', 'co', 'around', 'does', "you're", 'before', 'sincere',
'anything', "you'd", 'becomes', 'their', 'hereupon', 'hadn', 'inc', 'having',
'whoever', 'until', 'within', 'd', 'because', 'above', 'part', 'we', 'm',
'afterwards', "mustn't", 'hundred', 'perhaps', 'via', 'three', 'mine', 'where',
'nowhere', 'few', 'thereupon', 'upon', 'whole', 'then', 'somewhere', 'less',
'for', 'often', 'ever', 'amount', 'neither', 'front', "it's", 'these', 'onto',
'they', 'meanwhile', 'twelve', 'all', 'being', 'last', 'towards', 'below',
'many', 'six', 'o', 'seeming', 'throughout', 'together', 'again', "that'll",
'may', 'un', 'seemed', 'doesn', 'amoungst', 'con', 'anyone', 'each', 'shan',
'forty', 'am', 'across', 'over', 'everyone', 'this', 'hence', 'herein', 'full',
'fifteen', 'so', 'least', 'only', 'another', 'third', 'please', 'thereafter',
'sometimes', 'there', 'never', 'can', 'nevertheless', 'when', 'whereupon',
'him', 'not', 'such', 'next', 'those', 'why', 'himself', 'could', 'same',
'should', 'shouldn', 'our', 're', "didn't", 'just', 'back', 'first', 'alone',
'since', 'hers', 'still', 'whenever', 'won', 'anywhere', 'further', 'seem',
'during', 'thin', 'might', "should've", 'was', 'even', 'move', 'fire', 'bill',
'been', 's', 'up', 'at', 'whereas', 'will', 'too', 'eleven', 'mill', 'system',
'whom', 'noone', 'out', 'which', 'but', 'hereafter', 'among', 'cant', 'either',
'nobody', "she's", 'eight', 'indeed', "needn't", 'cry', 'a', 'nothing', 'on',
'also', 'ie', 'find', 'keep', 'themselves', "haven't", 'formerly', 'though',
'someone', 'behind', 'twenty', 'everywhere', 'whose', 'wouldn', "you've",
'therefore', 'be', 'cannot', 'were', 'none', 'one', 'aren', 'mustn', 'whereby',
'through', "hasn't", 'enough', 'once', 'mostly', 'much', 'although', 'his',
'me', 'become', 'amongst', 'the', "isn't", 'done', 'latter', 'you', 'nor',
'whence', 'isn', 'if', 'between', 'every', 'couldnt', 'yourself', 'what',
'weren', 'therein', 'de', 'mightn', 'more', 'ours', 'became', 'eg', 'take',
'have', 'latterly', 'go', 'etc', 'already', 'with', 'wherein', 'from', 'other',
'herself', "shouldn't", 'beforehand', 'call', 'off', 'beside', 'whether',
'sixty', 'somehow', 'in', 'fifty', 'otherwise', 'whatever', 'toward', 'did',
'elsewhere', 'didn', 't', 'sometime', 'hereby', 'moreover', 'show', 'detail',
'no', 'hasnt', 'however', 'side', 'anyway', 'theirs', 'is', 'put', 'interest',
'it', 'by', 'else', 'y', 'whither', 'after', 'fill', 'becoming', 'describe',
'are', 'doing', 'something', 'under'}), 'vect__tokenizer':
<__main__.LemmaTokenizer_word object at 0x7f40709b1430>}
Run time: 137.79279041290283 seconds
```

```
[ ]: #testing tfidf  => not good
     t_start = time.time()

     pipe_params = {
         'clf__criterion': ['entropy'],
         'vect__stop_words': [stop_words_library],
         #'vect__tokenizer': [LemmaTokenizer_word()],
         'vect__binary': [False],
         'vect__preprocessor': [preprocess_text],
         'clf__max_depth': [100],
         'clf__min_samples_split': [2, 5],
         'clf__min_samples_leaf': [1, 2, 4],
         'selecter__k':[5000,3000]
     }

     vectorizer = TfidfVectorizer()
     normalizer = Normalizer()
     selecter = SelectKBest(chi2)
     model = DecisionTreeClassifier()

     pipe = Pipeline(
         [("vect", vectorizer),("normalizer", normalizer),("selecter",␣
      ↪selecter),("clf",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 12 candidates, totalling 60 fits
The best accuracy is 87.888.
The winning parameters are {'clf__criterion': 'entropy', 'clf__max_depth': 100,
'clf__min_samples_leaf': 1, 'clf__min_samples_split': 2, 'selecter__k': 3000,
'vect__binary': False, 'vect__preprocessor': <function preprocess_text at
0x7f6d6a362670>, 'vect__stop_words': ['to', 'made', 'his', 'sometime', 'those',
'except', 'no', 'which', 'shan', 'detail', 'when', 'con', 'haven', "needn't",
'take', 'ever', 'would', "shan't", 'side', 'thick', 'thereby', 'hence', 'third',
'ours', 'six', 'she', 'then', 'seeming', 'therein', 'front', 'show', 'below',
'amoungst', 'none', "couldn't", 'keep', 'bottom', 'hereby', 'wherein',
'latterly', 'we', 'upon', 'must', 'once', 'more', 'therefore', 'doesn', 'same',
'seems', 'sincere', 'last', 'whence', 'inc', 'about', "she's", 'needn', 'just',
'somewhere', "don't", 'anything', 'empty', 'via', 'often', 'me', 'whole',
'together', 'call', 'whoever', 'everything', 'former', 'am', 'others',
'elsewhere', 'may', 'thereafter', "you've", 'seem', "wouldn't", 'himself',
'both', 'ltd', 'because', 'hasnt', 'hereupon', 'even', 'herein', 'could',
'among', 'part', 'should', 'system', 'hadn', 'under', 'yourselves', "wasn't",
'fill', 'who', 'a', 'mill', 'whose', 'whether', 'nothing', 'hereafter', 'any',
'are', 'here', 'nor', 'anyhow', "didn't", 'ma', 'as', 'before', 'd', 'many',
'cant', 'own', 'whereafter', 'otherwise', 'not', 'my', 'hundred', 'against',
'what', 'two', 'twelve', 'onto', 'eg', 'serious', "you're", 'of', 'well', 'the',
'us', 'your', 'he', 'thereupon', 'along', "hadn't", 'etc', 'whereby', 'been',
'from', 'theirs', 've', 'mustn', 'though', 'ten', 'in', 'never', "you'd",
'least', 'having', 'every', 'with', 'without', 'beforehand', 'i', 'across',
'full', 'couldnt', 'somehow', 'again', 'how', 'becomes', 'you', 'four', 'five',
'until', 'y', 'might', 'll', 'mostly', 'co', 'beside', 'mightn', 'wasn',
"mightn't", 'other', 'already', 'her', "it's", "isn't", 'always', 'by',
'indeed', 'around', 'either', 'noone', 'although', 'beyond', 'or', 'yours',
'is', 'cannot', 'shouldn', 'cry', 'o', 'for', 'only', 'mine', 'out', 'isn',
'less', "mustn't", 'fire', 'fifteen', 'nowhere', 'into', 'nevertheless',
'seemed', 'moreover', 'now', "won't", 'each', 'why', 'eleven', 'at', 'very',
'besides', 'amongst', 'didn', 'don', 'couldn', 'describe', 'ourselves', 'be',
'thin', 'such', 'during', 'someone', "you'll", 'on', 'thence', 'herself',
'rather', 'twenty', 'get', 'toward', 'everyone', 'won', 'too', 'else', 'wouldn',
'if', 'find', 'also', 'eight', 'whereas', "that'll", 'bill', 'itself', 'since',
'sometimes', 'these', 'this', 'off', 'interest', 'where', 'above', 'alone',
'up', "doesn't", 'hers', 'some', 'it', "shouldn't", 'un', 'have', 'anyone',
'please', 'fifty', "weren't", 'all', 'neither', 'they', 'afterwards', "aren't",
'wherever', 'becoming', 'and', 'whereupon', 's', 'name', 'that', 'something',
'has', 'enough', 'further', 'an', 'meanwhile', 'will', 'next', 'thru',
'another', 'perhaps', 'still', 'forty', 'one', 'whatever', 'doing', 'being',
'several', 'sixty', 'everywhere', 'move', 'yourself', 'per', 'whither', 'put',
'give', 'nobody', 'than', 'however', 'aren', 'through', 'namely', 'can',
'almost', 'latter', 'themselves', 'was', 'anywhere', 'there', 'behind',
'formerly', 'its', 'three', 'much', 'nine', 're', 'does', 'back', 'most',
'between', 'within', 'down', 'de', 'over', 'anyway', 'their', 'were', 'amount',
'weren', "should've", 'while', 'towards', 'ain', 'few', 'hasn', 'but', 'become',
'throughout', 't', 'whenever', 'had', 'top', 'ie', 'myself', 'so', 'see',
"haven't", 'yet', 'done', 'after', 'whom', 'first', 'them', "hasn't", 'go',
'found', 'due', 'became', 'm', 'did', 'him', 'do', 'thus', 'our']}
Run time: 16.844464778900146 seconds
```

```
[ ]: #testing stemmization => does not improve
     t_start = time.time()

     pipe_params = {
         'clf__criterion': ['entropy'],
         'vect__stop_words': [stop_words_library,None],
         'vect__tokenizer': [StemTokenizer()],
         'vect__binary': [False],
         'vect__preprocessor': [preprocess_text],
         'clf__max_depth': [100],
         'clf__min_samples_split': [2, 5],
         'clf__min_samples_leaf': [1, 2, 4],
         'selecter__k':[5000,3000],}

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

     pipe = Pipeline(
         [("vect", vectorizer),("normalizer", normalizer),("selecter",␣
      ↪selecter),("clf",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

/usr/local/lib/python3.8/dist-packages/sklearn/feature_extraction/text.py:396:
UserWarning: Your stop_words may be inconsistent with your preprocessing.
Tokenizing the stop words generated tokens ['abov', 'afterward', 'alon',
'alreadi', 'alway', 'ani', 'anoth', 'anyon', 'anyth', 'anywher', 'becam',
'becaus', 'becom', 'befor', 'besid', 'cri', 'describ', 'doe', 'dure', 'els',
'elsewher', 'empti', 'everi', 'everyon', 'everyth', 'everywher', 'fifti',
'formerli', 'forti', 'ha', 'henc', 'hereaft', 'herebi', 'hi', 'howev', 'hundr',
'inde', 'latterli', 'mani', 'meanwhil', 'moreov', 'mostli', 'need', 'nobodi',
'noon', 'noth', 'nowher', 'onc', 'onli', 'otherwis', 'ourselv', 'perhap',
'pleas', 'seriou', 'sever', 'sha', 'sinc', 'sincer', 'sixti', 'someon',
'someth', 'sometim', 'somewher', 'themselv', 'thenc', 'thereaft', 'therebi',
'therefor', 'thi', 'thu', 'togeth', 'twelv', 'twenti', 'veri', 'wa', 'whatev',
'whenc', 'whenev', 'wherea', 'whereaft', 'wherebi', 'wherev', 'whi', 'wo',
'yourselv'] not in stop_words.
  warnings.warn(

The best accuracy is 86.074.
The winning parameters are {'clf__criterion': 'entropy', 'clf__max_depth': 100,
'clf__min_samples_leaf': 2, 'clf__min_samples_split': 2, 'selecter__k': 3000,
'vect__binary': False, 'vect__preprocessor': <function preprocess_text at
0x7f407144cee0>, 'vect__stop_words': frozenset({'now', 'along', 'empty', 'don',
'yours', 'well', 'll', 'about', 'four', 'top', 'serious', 'yourselves', 'than',
'both', 'due', 'and', 'into', 'her', 'thereby', 've', 'except', 'see', 'i',
'down', 'ourselves', 'as', 'thick', 'must', 'do', 'she', 'my', 'own', 'us',
'thus', 'very', 'of', 'wasn', 'your', 'its', 'he', 'former', 'yet', 'almost',
'wherever', 'any', 'had', 'that', 'an', 'itself', "shan't", 'nine', 'besides',
'some', 'whereafter', 'who', 'haven', 'thence', 'namely', 'would', 'everything',
'others', 'seems', 'ain', 'ma', 'rather', "aren't", 'while', "mightn't",
'needn', "you'll", 'beyond', "wouldn't", 'five', 'them', 'thru', 'several',
'two', 'name', 'bottom', "couldn't", 'per', 'most', "doesn't", 'ltd', 'give',
"wasn't", 'without', 'get', 'ten', "don't", 'couldn', 'hasn', 'made', 'or',
"weren't", "hadn't", 'how', 'found', 'anyhow', 'against', 'myself', 'to',
'always', "won't", 'here', 'has', 'co', 'around', 'does', "you're", 'before',
'sincere', 'anything', "you'd", 'becomes', 'their', 'hereupon', 'hadn', 'inc',
'having', 'whoever', 'until', 'within', 'd', 'because', 'above', 'part', 'we',
'm', 'afterwards', "mustn't", 'hundred', 'perhaps', 'via', 'three', 'mine',
'where', 'nowhere', 'few', 'thereupon', 'upon', 'whole', 'then', 'somewhere',
'less', 'for', 'often', 'ever', 'amount', 'neither', 'front', "it's", 'these',
'onto', 'they', 'meanwhile', 'twelve', 'all', 'being', 'last', 'towards',
'below', 'many', 'six', 'o', 'seeming', 'throughout', 'together', 'again',
"that'll", 'may', 'un', 'seemed', 'doesn', 'amoungst', 'con', 'anyone', 'each',
'shan', 'forty', 'am', 'across', 'over', 'everyone', 'this', 'hence', 'herein',
'full', 'fifteen', 'so', 'least', 'only', 'another', 'third', 'please',
'thereafter', 'sometimes', 'there', 'never', 'can', 'nevertheless', 'when',
'whereupon', 'him', 'not', 'such', 'next', 'those', 'why', 'himself', 'could',
'same', 'should', 'shouldn', 'our', 're', "didn't", 'just', 'back', 'first',
'alone', 'since', 'hers', 'still', 'whenever', 'won', 'anywhere', 'further',
'seem', 'during', 'thin', 'might', "should've", 'was', 'even', 'move', 'fire',
'bill', 'been', 's', 'up', 'at', 'whereas', 'will', 'too', 'eleven', 'mill',
'system', 'whom', 'noone', 'out', 'which', 'but', 'hereafter', 'among', 'cant',
'either', 'nobody', "she's", 'eight', 'indeed', "needn't", 'cry', 'a',
'nothing', 'on', 'also', 'ie', 'find', 'keep', 'themselves', "haven't",
'formerly', 'though', 'someone', 'behind', 'twenty', 'everywhere', 'whose',
'wouldn', "you've", 'therefore', 'be', 'cannot', 'were', 'none', 'one', 'aren',
'mustn', 'whereby', 'through', "hasn't", 'enough', 'once', 'mostly', 'much',
'although', 'his', 'me', 'become', 'amongst', 'the', "isn't", 'done', 'latter',
'you', 'nor', 'whence', 'isn', 'if', 'between', 'every', 'couldnt', 'yourself',
'what', 'weren', 'therein', 'de', 'mightn', 'more', 'ours', 'became', 'eg',
'take', 'have', 'latterly', 'go', 'etc', 'already', 'with', 'wherein', 'from',
'other', 'herself', "shouldn't", 'beforehand', 'call', 'off', 'beside',
'whether', 'sixty', 'somehow', 'in', 'fifty', 'otherwise', 'whatever', 'toward',
'did', 'elsewhere', 'didn', 't', 'sometime', 'hereby', 'moreover', 'show',
'detail', 'no', 'hasnt', 'however', 'side', 'anyway', 'theirs', 'is', 'put',
'interest', 'it', 'by', 'else', 'y', 'whither', 'after', 'fill', 'becoming',
'describe', 'are', 'doing', 'something', 'under'}), 'vect__tokenizer':
<__main__.StemTokenizer object at 0x7f407120e880>}
Run time: 208.76408529281616 seconds
```

[ ]:
```python
#testing custom => 86.351.
t_start = time.time()

pipe_params = {
    'clf__criterion': ['entropy'],
    'vect__stop_words': [stop_words_library],
    'vect__tokenizer': [LemmaTokenizer_word()],
    'vect__binary': [False],
    'vect__preprocessor': [preprocess_text],
    'clf__max_depth': [100],
    'clf__min_samples_split': [2, 5],
    'clf__min_samples_leaf': [1, 2, 4],
    'selecter__k':[5000,3000]
}

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

pipe = Pipeline(
    [("vect", vectorizer),("selecter", selecter),("clf",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 12 candidates, totalling 60 fits
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/feature_extraction/text.py:396:
UserWarning: Your stop_words may be inconsistent with your preprocessing.
Tokenizing the stop words generated tokens ["'d", "'ll", "'re", "'s", "'ve",
'make', "n't", 'need', 'sha', 'win', 'wo'] not in stop_words.
  warnings.warn(

The best accuracy is 86.351.
The winning parameters are {'clf__criterion': 'entropy', 'clf__max_depth': 100,
'clf__min_samples_leaf': 2, 'clf__min_samples_split': 2, 'selecter__k': 3000,
'vect__binary': False, 'vect__preprocessor': <function preprocess_text at
0x7f407144cee0>, 'vect__stop_words': frozenset({'now', 'along', 'empty', 'don',
'yours', 'well', 'll', 'about', 'four', 'top', 'serious', 'yourselves', 'than',
'both', 'due', 'and', 'into', 'her', 'thereby', 've', 'except', 'see', 'i',
'down', 'ourselves', 'as', 'thick', 'must', 'do', 'she', 'my', 'own', 'us',
'thus', 'very', 'of', 'wasn', 'your', 'its', 'he', 'former', 'yet', 'almost',
'wherever', 'any', 'had', 'that', 'an', 'itself', "shan't", 'nine', 'besides',
'some', 'whereafter', 'who', 'haven', 'thence', 'namely', 'would', 'everything',
'others', 'seems', 'ain', 'ma', 'rather', "aren't", 'while', "mightn't",
'needn', "you'll", 'beyond', "wouldn't", 'five', 'them', 'thru', 'several',
'two', 'name', 'bottom', "couldn't", 'per', 'most', "doesn't", 'ltd', 'give',
"wasn't", 'without', 'get', 'ten', "don't", 'couldn', 'hasn', 'made', 'or',
"weren't", "hadn't", 'how', 'found', 'anyhow', 'against', 'myself', 'to',
'always', "won't", 'here', 'has', 'co', 'around', 'does', "you're", 'before',
'sincere', 'anything', "you'd", 'becomes', 'their', 'hereupon', 'hadn', 'inc',
'having', 'whoever', 'until', 'within', 'd', 'because', 'above', 'part', 'we',
'm', 'afterwards', "mustn't", 'hundred', 'perhaps', 'via', 'three', 'mine',
'where', 'nowhere', 'few', 'thereupon', 'upon', 'whole', 'then', 'somewhere',
'less', 'for', 'often', 'ever', 'amount', 'neither', 'front', "it's", 'these',
'onto', 'they', 'meanwhile', 'twelve', 'all', 'being', 'last', 'towards',
'below', 'many', 'six', 'o', 'seeming', 'throughout', 'together', 'again',
"that'll", 'may', 'un', 'seemed', 'doesn', 'amoungst', 'con', 'anyone', 'each',
'shan', 'forty', 'am', 'across', 'over', 'everyone', 'this', 'hence', 'herein',
'full', 'fifteen', 'so', 'least', 'only', 'another', 'third', 'please',
'thereafter', 'sometimes', 'there', 'never', 'can', 'nevertheless', 'when',
'whereupon', 'him', 'not', 'such', 'next', 'those', 'why', 'himself', 'could',
'same', 'should', 'shouldn', 'our', 're', "didn't", 'just', 'back', 'first',
'alone', 'since', 'hers', 'still', 'whenever', 'won', 'anywhere', 'further',
'seem', 'during', 'thin', 'might', "should've", 'was', 'even', 'move', 'fire',
'bill', 'been', 's', 'up', 'at', 'whereas', 'will', 'too', 'eleven', 'mill',
'system', 'whom', 'noone', 'out', 'which', 'but', 'hereafter', 'among', 'cant',
'either', 'nobody', "she's", 'eight', 'indeed', "needn't", 'cry', 'a',
'nothing', 'on', 'also', 'ie', 'find', 'keep', 'themselves', "haven't",
'formerly', 'though', 'someone', 'behind', 'twenty', 'everywhere', 'whose',
'wouldn', "you've", 'therefore', 'be', 'cannot', 'were', 'none', 'one', 'aren',
'mustn', 'whereby', 'through', "hasn't", 'enough', 'once', 'mostly', 'much',
'although', 'his', 'me', 'become', 'amongst', 'the', "isn't", 'done', 'latter',
'you', 'nor', 'whence', 'isn', 'if', 'between', 'every', 'couldnt', 'yourself',
'what', 'weren', 'therein', 'de', 'mightn', 'more', 'ours', 'became', 'eg',
'take', 'have', 'latterly', 'go', 'etc', 'already', 'with', 'wherein', 'from',
'other', 'herself', "shouldn't", 'beforehand', 'call', 'off', 'beside',
'whether', 'sixty', 'somehow', 'in', 'fifty', 'otherwise', 'whatever', 'toward',
'did', 'elsewhere', 'didn', 't', 'sometime', 'hereby', 'moreover', 'show',
'detail', 'no', 'hasnt', 'however', 'side', 'anyway', 'theirs', 'is', 'put',
'interest', 'it', 'by', 'else', 'y', 'whither', 'after', 'fill', 'becoming',
'describe', 'are', 'doing', 'something', 'under'}), 'vect__tokenizer':
<__main__.LemmaTokenizer_word object at 0x7f4071487370>}
Run time: 47.835500717163086 seconds
```

[ ]:
```python
#removing custom preprocessor => 86.21
t_start = time.time()

pipe_params = {
    'clf__criterion': ['entropy'],
    'vect__stop_words': [stop_words_library],
    'vect__tokenizer': [LemmaTokenizer_word()],
    'vect__binary': [False],
    'clf__max_depth': [100],
    'clf__min_samples_split': [2, 5],
    'clf__min_samples_leaf': [1, 2, 4],
    'selecter__k':[5000,3000]
}

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

pipe = Pipeline(
    [("vect", vectorizer),("selecter", selecter),("clf",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 12 candidates, totalling 60 fits

/usr/local/lib/python3.8/dist-packages/sklearn/feature_extraction/text.py:396:
```

```
UserWarning: Your stop_words may be inconsistent with your preprocessing.
Tokenizing the stop words generated tokens ["'d", "'ll", "'re", "'s", "'ve",
'make', "n't", 'need', 'sha', 'win', 'wo'] not in stop_words.
  warnings.warn(

The best accuracy is 86.21.
The winning parameters are {'clf__criterion': 'entropy', 'clf__max_depth': 100,
'clf__min_samples_leaf': 1, 'clf__min_samples_split': 2, 'selecter__k': 5000,
'vect__binary': False, 'vect__stop_words': frozenset({'now', 'along', 'empty',
'don', 'yours', 'well', 'll', 'about', 'four', 'top', 'serious', 'yourselves',
'than', 'both', 'due', 'and', 'into', 'her', 'thereby', 've', 'except', 'see',
'i', 'down', 'ourselves', 'as', 'thick', 'must', 'do', 'she', 'my', 'own', 'us',
'thus', 'very', 'of', 'wasn', 'your', 'its', 'he', 'former', 'yet', 'almost',
'wherever', 'any', 'had', 'that', 'an', 'itself', "shan't", 'nine', 'besides',
'some', 'whereafter', 'who', 'haven', 'thence', 'namely', 'would', 'everything',
'others', 'seems', 'ain', 'ma', 'rather', "aren't", 'while', "mightn't",
'needn', "you'll", 'beyond', "wouldn't", 'five', 'them', 'thru', 'several',
'two', 'name', 'bottom', "couldn't", 'per', 'most', "doesn't", 'ltd', 'give',
"wasn't", 'without', 'get', 'ten', "don't", 'couldn', 'hasn', 'made', 'or',
"weren't", "hadn't", 'how', 'found', 'anyhow', 'against', 'myself', 'to',
'always', "won't", 'here', 'has', 'co', 'around', 'does', "you're", 'before',
'sincere', 'anything', "you'd", 'becomes', 'their', 'hereupon', 'hadn', 'inc',
'having', 'whoever', 'until', 'within', 'd', 'because', 'above', 'part', 'we',
'm', 'afterwards', "mustn't", 'hundred', 'perhaps', 'via', 'three', 'mine',
'where', 'nowhere', 'few', 'thereupon', 'upon', 'whole', 'then', 'somewhere',
'less', 'for', 'often', 'ever', 'amount', 'neither', 'front', "it's", 'these',
'onto', 'they', 'meanwhile', 'twelve', 'all', 'being', 'last', 'towards',
'below', 'many', 'six', 'o', 'seeming', 'throughout', 'together', 'again',
"that'll", 'may', 'un', 'seemed', 'doesn', 'amoungst', 'con', 'anyone', 'each',
'shan', 'forty', 'am', 'across', 'over', 'everyone', 'this', 'hence', 'herein',
'full', 'fifteen', 'so', 'least', 'only', 'another', 'third', 'please',
'thereafter', 'sometimes', 'there', 'never', 'can', 'nevertheless', 'when',
'whereupon', 'him', 'not', 'such', 'next', 'those', 'why', 'himself', 'could',
'same', 'should', 'shouldn', 'our', 're', "didn't", 'just', 'back', 'first',
'alone', 'since', 'hers', 'still', 'whenever', 'won', 'anywhere', 'further',
'seem', 'during', 'thin', 'might', "should've", 'was', 'even', 'move', 'fire',
'bill', 'been', 's', 'up', 'at', 'whereas', 'will', 'too', 'eleven', 'mill',
'system', 'whom', 'noone', 'out', 'which', 'but', 'hereafter', 'among', 'cant',
'either', 'nobody', "she's", 'eight', 'indeed', "needn't", 'cry', 'a',
'nothing', 'on', 'also', 'ie', 'find', 'keep', 'themselves', "haven't",
'formerly', 'though', 'someone', 'behind', 'twenty', 'everywhere', 'whose',
'wouldn', "you've", 'therefore', 'be', 'cannot', 'were', 'none', 'one', 'aren',
'mustn', 'whereby', 'through', "hasn't", 'enough', 'once', 'mostly', 'much',
'although', 'his', 'me', 'become', 'amongst', 'the', "isn't", 'done', 'latter',
'you', 'nor', 'whence', 'isn', 'if', 'between', 'every', 'couldnt', 'yourself',
'what', 'weren', 'therein', 'de', 'mightn', 'more', 'ours', 'became', 'eg',
'take', 'have', 'latterly', 'go', 'etc', 'already', 'with', 'wherein', 'from',
'other', 'herself', "shouldn't", 'beforehand', 'call', 'off', 'beside',
'whether', 'sixty', 'somehow', 'in', 'fifty', 'otherwise', 'whatever', 'toward',
'did', 'elsewhere', 'didn', 't', 'sometime', 'hereby', 'moreover', 'show',
'detail', 'no', 'hasnt', 'however', 'side', 'anyway', 'theirs', 'is', 'put',
'interest', 'it', 'by', 'else', 'y', 'whither', 'after', 'fill', 'becoming',
'describe', 'are', 'doing', 'something', 'under'}), 'vect__tokenizer':
<__main__.LemmaTokenizer_word object at 0x7f406360e400>}
Run time: 46.78005290031433 seconds
```

[ ]:
```
#testing Ngram
t_start = time.time()

pipe_params = {
    'clf__criterion': ['entropy'],
    'vect__stop_words': [list(stop_words_custom)],
    'vect__tokenizer': [LemmaTokenizer_word()],
    'vect__binary': [False],
    'vect__ngram_range':[(1,1)],
    'clf__max_depth': [100],
    'clf__min_samples_split': [2, 5],
    'clf__min_samples_leaf': [1, 2, 4],
    'selecter__k':[5000,3000],
    "normalizer__norm": ['l2','l1']
}

vectorizer = CountVectorizer()
selecter = SelectKBest(chi2)
normalizer = Normalizer()
model = DecisionTreeClassifier()

pipe = Pipeline(
    [("vect", vectorizer),("normalizer",normalizer),("selecter",␣
 ↪selecter),("clf",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
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/feature_extraction/text.py:528:
UserWarning: The parameter 'token_pattern' will not be used since 'tokenizer' is
not None'
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/feature_extraction/text.py:409:
UserWarning: Your stop_words may be inconsistent with your preprocessing.
Tokenizing the stop words generated tokens ["'d", "'ll", "'m", "'re", "'s",
"'ve", 'ai', 'base', 'bite', 'ca', 'comment', 'concern', 'consider', 'exclude',
'follow', 'gon', 'greet', 'leave', "n't", 'na', 'regard', 'sit', 'site', 'wan',
'web', 'wo'] not in stop_words.
  warnings.warn(

The best accuracy is 84.408.
The winning parameters are {'clf__criterion': 'entropy', 'clf__max_depth': 100,
'clf__min_samples_leaf': 4, 'clf__min_samples_split': 2, 'normalizer__norm':
'l1', 'selecter__k': 3000, 'vect__binary': False, 'vect__ngram_range': (1, 1),
'vect__stop_words': ['i', "i'll", "i'd", "i'm", "i've", 'ive', 'me', 'myself',
'you', "you'll", "you'd", "you're", "you've", 'yourself', 'he', "he'll", "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", 'a', 'an', 'the', 'this', 'that',
"that's", 'these', 'those', 'my', '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', 'rather', 'quite',
'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', 'since', 'than', 'there', "there's", 'through', 'to',
'toward', 'towards', 'under', 'underneath', 'unlike', 'until', 'up', 'upon',
'versus', 'via', 'with', 'within', 'without', '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', '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', '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', 'thing', "thing's", 'things', 'stuff', "other's",
'others', "another's", 'total', '', 'false', 'none', 'way', 'kind', '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', 'hello', 'hi', 'hey', 'sup', 'yo', 'greetings', 'please', 'okay',
'ok', "y'all", 'lol', 'rofl', 'thank', 'thanks', 'alright', 'kinda', 'dont',
'sorry', 'idk', 'tldr', 'tl', 'dr', 'tbh', 'dude', 'tho', 'aka', 'plz', 'pls',
'bit', 'don', 'www', 'https', 'http', 'com', 'etchtml', 'reddit', 'subreddit',
'subreddits', 'comments', 'reply', 'replies', 'thread', 'threads', 'post',
'posts', 'website', 'websites', 'web site', 'web sites'], 'vect__tokenizer':
<__main__.LemmaTokenizer_word object at 0x7f6d73fe0dc0>}
Run time: 170.62921714782715 seconds
```

```
[ ]: #testing features
t_start = time.time()

final_pipe_params = {
    'clf__criterion': ['gini', 'entropy'],
    'vect__stop_words': [list(stop_words_custom)],
    'clf__max_depth': [100],
    'clf__min_samples_split': [2, 5],
    'clf__min_samples_leaf': [1, 2, 4],
    'selecter__k':[5000,3000]
}

final_vectorizer = CountVectorizer()
final_selecter = SelectKBest(chi2)
final_model = DecisionTreeClassifier()

pipe = Pipeline(
    [("vect", final_vectorizer),("selecter",␣
 ↪final_selecter),("clf",final_model)]
)

final_grid = model_selection.GridSearchCV(pipe, final_pipe_params, verbose=1,␣
 ↪n_jobs=-1)

final_grid.fit(train_x, train_y)

t_end = time.time()

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

print(f"The best accuracy is {accuracy}.")
print(f"The winning parameters are {final_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 88.444.
The winning parameters are {'clf__criterion': 'gini', 'clf__max_depth': 100,
'clf__min_samples_leaf': 1, 'clf__min_samples_split': 5, 'selecter__k': 5000,
'vect__stop_words': ['i', "i'll", "i'd", "i'm", "i've", 'ive', 'me', 'myself',
'you', "you'll", "you'd", "you're", "you've", 'yourself', 'he', "he'll", "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", 'a', 'an', 'the', 'this', 'that',
"that's", 'these', 'those', 'my', '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', 'rather', 'quite',
'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', 'since', 'than', 'there', "there's", 'through', 'to',
'toward', 'towards', 'under', 'underneath', 'unlike', 'until', 'up', 'upon',
'versus', 'via', 'with', 'within', 'without', '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', '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', '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', 'thing', "thing's", 'things', 'stuff', "other's",
'others', "another's", 'total', '', 'false', 'none', 'way', 'kind', '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', 'hello', 'hi', 'hey', 'sup', 'yo', 'greetings', 'please', 'okay',
'ok', "y'all", 'lol', 'rofl', 'thank', 'thanks', 'alright', 'kinda', 'dont',
'sorry', 'idk', 'tldr', 'tl', 'dr', 'tbh', 'dude', 'tho', 'aka', 'plz', 'pls',
'bit', 'don', 'www', 'https', 'http', 'com', 'etchtml', 'reddit', 'subreddit',
'subreddits', 'comments', 'reply', 'replies', 'thread', 'threads', 'post',
'posts', 'website', 'websites', 'web site', 'web sites']}
Run time: 14.492570400238037 seconds
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/feature_extraction/text.py:409:
UserWarning: Your stop_words may be inconsistent with your preprocessing.
Tokenizing the stop words generated tokens ['ain', 'aren', 'couldn', 'didn',
'doesn', 'hadn', 'hasn', 'haven', 'isn', 'll', 're', 'shouldn', 'site', 'sites',
've', 'wasn', 'web', 'weren', 'won', 'wouldn'] not in stop_words.
  warnings.warn(
```

[ ]:
```
print(round(final_grid.best_score_ * 100,3))
print(f"Run time: {elapsed_time} seconds")
y_pred = final_grid.predict(test_x)
create_test_csv(y_pred,"DesicionTree_04032023_02.csv")
```

```
88.444
Run time: 14.492570400238037 seconds
File saved.
```

[ ]:
```
def print_best_params(grid):
  bestParameters = grid.best_estimator_.get_params()
  # print(bestParameters)
  for paramName in sorted(bestParameters.keys()):
    print("\t%s: %r" % (paramName, bestParameters[paramName]))
```

[ ]:
```
print_best_params(final_grid)
```

```
        clf: DecisionTreeClassifier(max_depth=100, min_samples_split=5)
        clf__ccp_alpha: 0.0
        clf__class_weight: None
        clf__criterion: 'gini'
        clf__max_depth: 100
        clf__max_features: None
        clf__max_leaf_nodes: None
        clf__min_impurity_decrease: 0.0
        clf__min_samples_leaf: 1
        clf__min_samples_split: 5
        clf__min_weight_fraction_leaf: 0.0
        clf__random_state: None
        clf__splitter: 'best'
        memory: None
        selecter: SelectKBest(k=5000, score_func=<function chi2 at
0x7f6d76ec2b80>)
        selecter__k: 5000
        selecter__score_func: <function chi2 at 0x7f6d76ec2b80>
        steps: [('vect', CountVectorizer(stop_words=['i', "i'll", "i'd", "i'm",
"i've", 'ive', 'me',
                            'myself', 'you', "you'll", "you'd", "you're",
                            "you've", 'yourself', 'he', "he'll", "he'd", "he's",
                            'him', 'she', "she'll", "she'd", "she's", 'her',
                            'it', "it'll", "it'd", "it's", 'itself', 'oneself',
…])), ('selecter', SelectKBest(k=5000, score_func=<function chi2 at
0x7f6d76ec2b80>)), ('clf', DecisionTreeClassifier(max_depth=100,
min_samples_split=5))]
        vect: CountVectorizer(stop_words=['i', "i'll", "i'd", "i'm", "i've",
'ive', 'me',
                            'myself', 'you', "you'll", "you'd", "you're",
                            "you've", 'yourself', 'he', "he'll", "he'd", "he's",
                            'him', 'she', "she'll", "she'd", "she's", 'her',
                            'it', "it'll", "it'd", "it's", 'itself', 'oneself',
…])
        vect__analyzer: 'word'
        vect__binary: False
        vect__decode_error: 'strict'
        vect__dtype: <class 'numpy.int64'>
        vect__encoding: 'utf-8'
        vect__input: 'content'
        vect__lowercase: True
        vect__max_df: 1.0
        vect__max_features: None
        vect__min_df: 1
        vect__ngram_range: (1, 1)
        vect__preprocessor: None
        vect__stop_words: ['i', "i'll", "i'd", "i'm", "i've", 'ive', 'me',
'myself', 'you', "you'll", "you'd", "you're", "you've", 'yourself', 'he',
"he'll", "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", 'a', 'an', 'the', 'this',
'that', "that's", 'these', 'those', 'my', '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', 'rather',
'quite', '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', 'since', 'than', 'there', "there's", 'through', 'to',
'toward', 'towards', 'under', 'underneath', 'unlike', 'until', 'up', 'upon',
'versus', 'via', 'with', 'within', 'without', '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', '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', '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', 'thing', "thing's", 'things', 'stuff', "other's",
'others', "another's", 'total', '', 'false', 'none', 'way', 'kind', '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', 'hello', 'hi', 'hey', 'sup', 'yo', 'greetings', 'please', 'okay',
'ok', "y'all", 'lol', 'rofl', 'thank', 'thanks', 'alright', 'kinda', 'dont',
'sorry', 'idk', 'tldr', 'tl', 'dr', 'tbh', 'dude', 'tho', 'aka', 'plz', 'pls',
'bit', 'don', 'www', 'https', 'http', 'com', 'etchtml', 'reddit', 'subreddit',
'subreddits', 'comments', 'reply', 'replies', 'thread', 'threads', 'post',
'posts', 'website', 'websites', 'web site', 'web sites']
        vect__strip_accents: None
        vect__token_pattern: '(?u)\\b\\w\\w+\\b'
        vect__tokenizer: None
        vect__vocabulary: None
        verbose: False
```

[ ]:
```
# Step 5: Make predictions on test data using the trained model
```

[ ]:
```
########################################################## final
```