stacking

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
from sklearn.svm import SVC
import time
import re
import string
from sklearn.naive_bayes import GaussianNB, MultinomialNB
from sklearn import svm
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] 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.
```

[ ]: True

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

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

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

[ ]:
```
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]))
```

[ ]:
```
#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)
```

[ ]:
```
#stemmer 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
```

```
[ ]: #base condition with stacking
from sklearn.pipeline import Pipeline
from sklearn.ensemble import StackingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import GridSearchCV
from sklearn.feature_extraction.text import TfidfVectorizer

# Define the base estimators for the stacking classifier
estimators = [
    ('lr', LogisticRegression(random_state=42)),
    ('mnb', MultinomialNB())
]

# Define the stacking classifier pipeline
stacking_pipeline = Pipeline([
    ('tfidf', TfidfVectorizer()),
    ('stacking', StackingClassifier(estimators=estimators))
])

# Define the grid search parameters
params = {
   # 'tfidf__max_df': [0.5, 0.75, 1.0],
  "tfidf__stop_words": [list(stop_words_library)],
   # 'tfidf__ngram_range': [(1,1), (1,2), (1,3)],
   # 'stacking__final_estimator__penalty': ['l1', 'l2'],
   # 'stacking__final_estimator__C': [0.1, 1.0, 10.0],
   # 'stacking__final_estimator__solver': ['liblinear', 'lbfgs']
}

# Define the grid search object
grid_search = GridSearchCV(stacking_pipeline, params, cv=5,scoring='accuracy')

# Fit the grid search object to the training data
grid_search.fit(train_x, train_y)

#accuracy = round(grid.best_score_ * 100,3)
accuracy = round(grid_search.best_score_ * 100,3)
```

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

```
The best accuracy is 94.568.
The winning parameters are {'tfidf__stop_words': ['even', 'put', 'eleven',
'won', 'didn', 'beforehand', 'toward', 'couldnt', 'mostly', 'eight', 'either',
'enough', 'your', 'while', 'been', 'anyway', 'sincere', 'hasnt', 'others',
'another', 'none', 'itself', 'as', 'formerly', 'often', 'about', 'off', 'just',
'during', 't', 'cannot', 'rather', "aren't", 'too', 'ever', 'wasn', 'less',
'yourself', 'myself', 'do', 'hereafter', "that'll", 'became', 'will', 'back',
"haven't", 'seemed', 'name', 'one', 'never', 'so', 'onto', "wasn't", 'find',
'until', 'if', "won't", 'here', 'elsewhere', 'no', 'those', 'needn', 'hence',
'meanwhile', 'from', 'hereupon', 'for', 'almost', 'did', 'least', 'with', 'she',
'many', 'without', 'noone', 'thereupon', 'not', 'my', 'throughout', 'thick',
'such', 'hadn', 'us', 'all', 'now', 'twenty', 'once', 'at', 'fifty', 'anywhere',
'whereas', 'former', 'else', 'always', 'sometimes', 'please', "mightn't",
'mightn', 'd', 'same', 'other', 'few', 'nobody', 'describe', 'sometime',
'somewhere', 'etc', 'seem', 'seems', "needn't", 'mill', 'which', 'thereafter',
'sixty', 'together', 'therein', 'two', "shan't", 'between', 'he', 'thin',
'already', 'his', 'their', 'hereby', 'doing', 'indeed', 'first', 'latterly',
'still', 'or', 'm', 'nor', 'can', 'neither', "hasn't", 'll', 'next', 'when',
'thru', 'over', 'hers', 'mustn', 'besides', 'could', 'side', 'ten',
'yourselves', 'move', 'nevertheless', 'ours', 'this', 'perhaps', 'fifteen',
"it's", 'well', 'con', 'up', 'un', 'be', 'mine', 'around', 'has', 'whatever',
'wouldn', 'them', 'five', 'last', 'each', "you're", 'nowhere', 'shouldn',
'wherever', 'ie', 'anyone', 'again', 'were', 'via', 'theirs', 'being', 'anyhow',
'it', 'more', 'under', 'have', 'since', 'through', 'having', "you'll", 'four',
'whereby', 'anything', 'front', 'afterwards', 'a', 'does', 's', 'six',
'somehow', 'should', 'shan', 'would', 'its', 'isn', 'any', 'where', 'keep',
'per', 'also', 'among', 'only', 'except', 'must', 'though', 'take', 'amoungst',
'behind', "isn't", 'of', 'done', 'show', 'own', 'by', "shouldn't", "weren't",
'give', 'after', 'twelve', "don't", 'thence', "wouldn't", "you've", 'then',
'these', 'to', 'everything', 'namely', "you'd", 'beside', 'i', 'ltd', 'don',
'me', 'due', "hadn't", 'hasn', 'made', 'whoever', 'above', 'forty',
'themselves', 'both', 'hundred', 're', 'our', 'amongst', 'however', 'moreover',
'out', 'fill', "couldn't", 'down', 'whom', 'become', 'haven', 'weren', 'thus',
'ma', 'below', 'becomes', 'everywhere', 'interest', 'much', 'herein', 'yours',
'seeming', 'is', 'nine', 'full', 'ourselves', 'ain', 'latter', 'across', 'am',
'call', 'whereupon', 'something', "doesn't", 'found', 'why', 'most',
'therefore', 'co', 'thereby', 'someone', 'empty', 'on', 'who', 'towards',
'whereafter', 'go', 'there', 'cry', 'they', 'because', 'beyond', 'bottom',
'that', 'de', 'further', 'y', 'very', 'whole', 'get', 'alone', 'than', 'detail',
'and', 'part', 'whenever', 'top', 'every', 'him', 'but', 'amount', 'everyone',
'herself', 'aren', 'along', 'three', 'fire', 'against', 'we', "she's",
'becoming', 've', 'are', 'bill', 'before', "mustn't", 'within', 'wherein',
'doesn', 'was', 'nothing', 'himself', 'the', 'whence', 'whither', 'otherwise',
'serious', 'eg', 'in', 'inc', 'into', 'o', 'some', 'upon', 'whether', 'yet',
'cant', 'several', 'how', 'had', 'may', 'whose', "should've", 'system',
"didn't", 'an', 'third', 'her', 'see', 'couldn', 'although', 'you', 'might',
'what']}
```

```
[ ]: #base condition with stacking
#=>94.846
from sklearn.pipeline import Pipeline
from sklearn.ensemble import StackingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import GridSearchCV
from sklearn.feature_extraction.text import TfidfVectorizer

# Define the base estimators for the stacking classifier
estimators = [
    ('lr', LogisticRegression(random_state=42)),
    ('mnb', MultinomialNB())
]

# Define the stacking classifier pipeline
stacking_pipeline = Pipeline([
    ('tfidf', TfidfVectorizer()),
    ('stacking', StackingClassifier(estimators=estimators))
])

# Define the grid search parameters
params = {
   # 'tfidf__max_df': [0.5, 0.75, 1.0],
  "tfidf__stop_words": [list(stop_words_library),list(stop_words_custom)],
   # 'tfidf__ngram_range': [(1,1), (1,2), (1,3)],
   # 'stacking__final_estimator__penalty': ['l1', 'l2'],
   # 'stacking__final_estimator__C': [0.1, 1.0, 10.0],
   # 'stacking__final_estimator__solver': ['liblinear', 'lbfgs']
}

# Define the grid search object
grid_search = GridSearchCV(stacking_pipeline, params, cv=5,scoring='accuracy'␣
 ↪,verbose=1, n_jobs=-1)

# Fit the grid search object to the training data
grid_search.fit(train_x, train_y)

#accuracy = round(grid.best_score_ * 100,3)
accuracy = round(grid_search.best_score_ * 100,3)
```

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

```
Fitting 5 folds for each of 2 candidates, totalling 10 fits
```

```
/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(
```

```
The best accuracy is 94.846.
The winning parameters are {'tfidf__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']}
```

[ ]:
```
#base condition with stacking

from sklearn.pipeline import Pipeline
from sklearn.ensemble import StackingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import GridSearchCV
from sklearn.feature_extraction.text import TfidfVectorizer

# Define the base estimators for the stacking classifier
estimators = [
    ('lr', LogisticRegression(random_state=42)),
    ('mnb', MultinomialNB())
]

# Define the stacking classifier pipeline
stacking_pipeline = Pipeline([
    ('cv', TfidfVectorizer()),
    ('stacking', StackingClassifier(estimators=estimators))
])

# Define the grid search parameters
params = {
    # 'tfidf__max_df': [0.5, 0.75, 1.0],
  "cv__stop_words": [list(stop_words_custom)],
    'stacking__mnb__alpha': [0.0001, 0.001, 0.01,0.5],
   # 'tfidf__ngram_range': [(1,1), (1,2), (1,3)],
   #'stacking__final_estimator__penalty': ['l1', 'l2'],
   # 'stacking__final_estimator__C': [0.1, 1.0, 10.0],
   # 'stacking__final_estimator__solver': ['liblinear', 'lbfgs']
}

# Define the grid search object
grid_search = GridSearchCV(stacking_pipeline, params, cv=5,scoring='accuracy'␣
 ↪,verbose=1, n_jobs=-1)

# Fit the grid search object to the training data
grid_search.fit(train_x, train_y)

#accuracy = round(grid.best_score_ * 100,3)
accuracy = round(grid_search.best_score_ * 100,3)

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

#print_best_params(grid_search)
```

```
Fitting 5 folds for each of 4 candidates, totalling 20 fits
```

```
/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(

The best accuracy is 95.123.
The winning parameters are {'cv__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'],
'stacking__mnb__alpha': 0.5}
```

[ ]:
```python
#base condition with stacking

from sklearn.pipeline import Pipeline
from sklearn.ensemble import StackingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import GridSearchCV
from sklearn.feature_extraction.text import TfidfVectorizer

# Define the base estimators for the stacking classifier
estimators = [
    ('lr', LogisticRegression(random_state=42)),
    ('mnb', MultinomialNB())
]

selecter = SelectKBest(chi2)
normalizer = Normalizer()


# Define the stacking classifier pipeline
stacking_pipeline = Pipeline([
    ('cv', TfidfVectorizer()),
    #("selecter", selecter),
    ("normalizer",normalizer),
    ('stacking', StackingClassifier(estimators=estimators))
])

# Define the grid search parameters
params = {
   # 'tfidf__max_df': [0.5, 0.75, 1.0],
    'stacking__mnb__alpha': [0.5],
       # "selecter__k":[5000],
          "cv__stop_words": [list(stop_words_custom)],
            "normalizer__norm": ['l2','l1']
   # 'tfidf__ngram_range': [(1,1), (1,2), (1,3)],
   #'stacking__final_estimator__penalty': ['l1', 'l2'],
   # 'stacking__final_estimator__C': [0.1, 1.0, 10.0],
   # 'stacking__final_estimator__solver': ['liblinear', 'lbfgs']
}

# Define the grid search object
grid_search = GridSearchCV(stacking_pipeline, params, cv=5,scoring='accuracy'␣
 ↪,verbose=1, n_jobs=-1)

# Fit the grid search object to the training data
grid_search.fit(train_x, train_y)

#accuracy = round(grid.best_score_ * 100,3)
accuracy = round(grid_search.best_score_ * 100,3)

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

#print_best_params(grid_search)
```

```
Fitting 5 folds for each of 2 candidates, totalling 10 fits
```

```
/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(
```

```
The best accuracy is 95.123.
The winning parameters are {'cv__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'], 'normalizer__norm':
'l2', 'stacking__mnb__alpha': 0.5}
```

[ ]:
```python
#base condition with stacking

from sklearn.pipeline import Pipeline
from sklearn.ensemble import StackingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import GridSearchCV
from sklearn.feature_extraction.text import TfidfVectorizer
#    'stacking__lr__solver': ['lbfgs', 'liblinear', 'newton-cg',␣
 ↪'newton-cholesky', 'sag', 'saga'],


# Define the base estimators for the stacking classifier
```

```
estimators = [
    ('lr', LogisticRegression(random_state=42)),
    ('mnb', MultinomialNB())
]

selecter = SelectKBest(chi2)
normalizer = Normalizer()


# Define the stacking classifier pipeline
stacking_pipeline = Pipeline([
    ('cv', TfidfVectorizer()),
    #("selecter", selecter),
    ("normalizer",normalizer),
    ('stacking', StackingClassifier(estimators=estimators))
])

# Define the grid search parameters
params = {
   # 'tfidf__max_df': [0.5, 0.75, 1.0],
    'stacking__mnb__alpha': [0.5],
       # "selecter__k":[5000],
    "cv__stop_words": [list(stop_words_custom)],
    "normalizer__norm": ['l2','l1'],
   # 'tfidf__ngram_range': [(1,1), (1,2), (1,3)],
    'stacking__lr__solver': ['sag', 'saga'],
}

# Define the grid search object
grid_search = GridSearchCV(stacking_pipeline, params, cv=5,scoring='accuracy'␣
 ↪,verbose=1, n_jobs=-1)

# Fit the grid search object to the training data
grid_search.fit(train_x, train_y)

#accuracy = round(grid.best_score_ * 100,3)
accuracy = round(grid_search.best_score_ * 100,3)

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

#print_best_params(grid_search)
```

```
Fitting 5 folds for each of 4 candidates, totalling 20 fits
```

```
/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(

The best accuracy is 95.123.
The winning parameters are {'cv__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'], 'normalizer__norm':
'l2', 'stacking__lr__solver': 'sag', 'stacking__mnb__alpha': 0.5}
```

[ ]:
```
#base condition with stacking

from sklearn.pipeline import Pipeline
from sklearn.ensemble import StackingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import GridSearchCV
from sklearn.feature_extraction.text import TfidfVectorizer
#    'stacking__lr__solver': ['lbfgs', 'liblinear', 'newton-cg',␣
↪'newton-cholesky', 'sag', 'saga'],
# Define the base estimators for the stacking classifier
estimators = [
    ('lr', LogisticRegression(random_state=42)),
    ('mnb', MultinomialNB())
]

selecter = SelectKBest(chi2)
normalizer = Normalizer()


# Define the stacking classifier pipeline
stacking_pipeline = Pipeline([
    ('cv', TfidfVectorizer()),
    #("selecter", selecter),
    ("normalizer",normalizer),
    ('stacking', StackingClassifier(estimators=estimators))
])

# Define the grid search parameters
params = {
   # 'tfidf__max_df': [0.5, 0.75, 1.0],
    'stacking__mnb__alpha': [0.5],
       # "selecter__k":[5000],
    "cv__stop_words": [list(stop_words_custom)],
    "normalizer__norm": ['l2','l1'],
   # 'tfidf__ngram_range': [(1,1), (1,2), (1,3)],
    'stacking__lr__solver': ['sag', 'lbfgs'],
}

# Define the grid search object
grid_search = GridSearchCV(stacking_pipeline, params, cv=5,scoring='accuracy'␣
 ↪,verbose=1, n_jobs=-1)

# Fit the grid search object to the training data
grid_search.fit(train_x, train_y)

#accuracy = round(grid.best_score_ * 100,3)
accuracy = round(grid_search.best_score_ * 100,3)

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

#print_best_params(grid_search)
```

```
Fitting 5 folds for each of 4 candidates, totalling 20 fits
/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(

The best accuracy is 95.123.
The winning parameters are {'cv__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'], 'normalizer__norm':
'l2', 'stacking__lr__solver': 'sag', 'stacking__mnb__alpha': 0.5}
```

[ ]:
```
#base condition with stacking

from sklearn.pipeline import Pipeline
from sklearn.ensemble import StackingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import GridSearchCV
from sklearn.feature_extraction.text import TfidfVectorizer
#    'stacking__lr__solver': ['lbfgs', 'liblinear', 'newton-cg',␣
↪'newton-cholesky', 'sag', 'saga'],
# Define the base estimators for the stacking classifier
estimators = [
    ('lr', LogisticRegression(random_state=42)),
    ('mnb', MultinomialNB())
]

selecter = SelectKBest(chi2)
normalizer = Normalizer()


# Define the stacking classifier pipeline
stacking_pipeline = Pipeline([
    ('cv', TfidfVectorizer()),
    ("selecter", selecter),
    ("normalizer",normalizer),
    ('stacking', StackingClassifier(estimators=estimators))
])

# Define the grid search parameters
params = {
    'cv__max_df': [0.5, 0.75],
    'stacking__mnb__alpha': [0.5],
    "selecter__k":[5000],
    "cv__stop_words": [list(stop_words_custom)],
    'cv__preprocessor': [preprocess_text],
    "normalizer__norm": ['l2'],
   'cv__ngram_range': [(1,1)],
    'stacking__lr__solver': ['sag'],
}

grid_search = GridSearchCV(stacking_pipeline, params, cv=5,scoring='accuracy'␣
 ↪,verbose=1, n_jobs=-1)

grid_search.fit(train_x, train_y)

accuracy = round(grid_search.best_score_ * 100,3)

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

#print_best_params(grid_search)
```

```
Fitting 5 folds for each of 2 candidates, totalling 10 fits
```

```
/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(

The best accuracy is 95.123.
The winning parameters are {'cv__max_df': 0.75, 'cv__ngram_range': (1, 1),
'cv__preprocessor': <function preprocess_text at 0x7f6cc558ea60>,
'cv__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'], 'normalizer__norm':
'l2', 'selecter__k': 5000, 'stacking__lr__solver': 'sag',
'stacking__mnb__alpha': 0.5}
```

[ ]:
```
#final

from sklearn.pipeline import Pipeline
from sklearn.ensemble import StackingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import GridSearchCV
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
# Define the base estimators for the stacking classifier
final_estimators = [
    ('lr', LogisticRegression(random_state=42)),
    ('mnb', MultinomialNB())
]

final_selecter = SelectKBest(chi2)
final_normalizer = Normalizer()


# Define the stacking classifier pipeline
final_stacking_pipeline = Pipeline([
    ('cv', TfidfVectorizer()),
    ("selecter", final_selecter),
    ("normalizer",final_normalizer),
    ('stacking', StackingClassifier(estimators=final_estimators))
])

# Define the grid search parameters
final_params = {
    'cv__max_df': [0.5, 0.75],
    'stacking__mnb__alpha': [0.5],
    "selecter__k":[5000],
    "cv__stop_words": [list(stop_words_custom)],
    'cv__preprocessor': [preprocess_text],
    "normalizer__norm": ['l2'],
   'cv__ngram_range': [(1,1)],
    'stacking__lr__solver': ['sag'],
}

final_grid = GridSearchCV(final_stacking_pipeline, final_params,␣
↪cv=5,scoring='accuracy' ,verbose=1, n_jobs=-1)

final_grid.fit(train_x, train_y)

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_best_params(grid_search)
```

```
Fitting 5 folds for each of 2 candidates, totalling 10 fits
```

```
/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(

The best accuracy is 95.123.
The winning parameters are {'cv__max_df': 0.75, 'cv__ngram_range': (1, 1),
'cv__preprocessor': <function preprocess_text at 0x7f6cc558ea60>,
'cv__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'], 'normalizer__norm':
'l2', 'selecter__k': 5000, 'stacking__lr__solver': 'sag',
'stacking__mnb__alpha': 0.5}
```

```
[ ]: y_pred_new = final_grid.predict(test_x)
     create_test_csv(y_pred_new,"Stacking_MultiNB-Logistic-05032023_01.csv")
```

```
File saved.
```

```
[ ]: ############################################################## CNN & Multinomial
```

```
[ ]: 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]))
```

```
[ ]: #new ensemble
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.ensemble import StackingClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score



# Define the base estimators for the stacking classifier
ensemble_estimators = [('nb', MultinomialNB()), ('mlp', MLPClassifier())]

ensemble_pipeline = Pipeline([
    ('cv', TfidfVectorizer()),
    ('stacking', StackingClassifier(estimators=ensemble_estimators))
])

# Define the grid search parameters
ensemble_params = {
    #'cv__max_df': [0.5, 0.75, 1.0],
   # 'nb__alpha': [0.1, 0.5, 1.0],
    "cv__stop_words": [list(stop_words_custom)],
   # 'mlp__hidden_layer_sizes': [(50,), (100,), (200,)],
    'stacking__mlp__alpha': [0.1],
}

ensemble_grid = GridSearchCV(ensemble_pipeline, ensemble_params,␣
 ↪cv=5,scoring='accuracy' ,verbose=1, n_jobs=-1)

ensemble_grid.fit(train_x, train_y)

accuracy = round(ensemble_grid.best_score_ * 100,3)

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

print_best_params(ensemble_grid)
```

```
Fitting 5 folds for each of 1 candidates, totalling 5 fits
```

```
/usr/local/lib/python3.8/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:684:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and
the optimization hasn't converged yet.
```

```
  warnings.warn(
/usr/local/lib/python3.8/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:684:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and
the optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.8/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:684:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and
the optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.8/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:684:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and
the optimization hasn't converged yet.
  warnings.warn(
/usr/local/lib/python3.8/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:684:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and
the optimization hasn't converged yet.
  warnings.warn(
```

```
The best accuracy is 93.729.
The winning parameters are {'stacking__mlp__alpha': 0.1}
        cv: TfidfVectorizer()
        cv__analyzer: 'word'
        cv__binary: False
        cv__decode_error: 'strict'
        cv__dtype: <class 'numpy.float64'>
        cv__encoding: 'utf-8'
        cv__input: 'content'
        cv__lowercase: True
        cv__max_df: 1.0
        cv__max_features: None
        cv__min_df: 1
        cv__ngram_range: (1, 1)
        cv__norm: 'l2'
        cv__preprocessor: None
        cv__smooth_idf: True
        cv__stop_words: None
        cv__strip_accents: None
        cv__sublinear_tf: False
        cv__token_pattern: '(?u)\\b\\w\\w+\\b'
        cv__tokenizer: None
        cv__use_idf: True
        cv__vocabulary: None
        memory: None
        stacking: StackingClassifier(estimators=[('nb', MultinomialNB()),
                              ('mlp', MLPClassifier(alpha=0.1))])
        stacking__cv: None
        stacking__estimators: [('nb', MultinomialNB()), ('mlp',
MLPClassifier(alpha=0.1))]
        stacking__final_estimator: None
        stacking__mlp: MLPClassifier(alpha=0.1)
        stacking__mlp__activation: 'relu'
        stacking__mlp__alpha: 0.1
        stacking__mlp__batch_size: 'auto'
        stacking__mlp__beta_1: 0.9
        stacking__mlp__beta_2: 0.999
        stacking__mlp__early_stopping: False
        stacking__mlp__epsilon: 1e-08
        stacking__mlp__hidden_layer_sizes: (100,)
        stacking__mlp__learning_rate: 'constant'
        stacking__mlp__learning_rate_init: 0.001
        stacking__mlp__max_fun: 15000
        stacking__mlp__max_iter: 200
        stacking__mlp__momentum: 0.9
        stacking__mlp__n_iter_no_change: 10
        stacking__mlp__nesterovs_momentum: True
        stacking__mlp__power_t: 0.5
        stacking__mlp__random_state: None
        stacking__mlp__shuffle: True
        stacking__mlp__solver: 'adam'
        stacking__mlp__tol: 0.0001
        stacking__mlp__validation_fraction: 0.1
        stacking__mlp__verbose: False
        stacking__mlp__warm_start: False
        stacking__n_jobs: None
        stacking__nb: MultinomialNB()
        stacking__nb__alpha: 1.0
        stacking__nb__class_prior: None
        stacking__nb__fit_prior: True
        stacking__nb__force_alpha: 'warn'
        stacking__passthrough: False
        stacking__stack_method: 'auto'
        stacking__verbose: 0
        steps: [('cv', TfidfVectorizer()), ('stacking',
StackingClassifier(estimators=[('nb', MultinomialNB()),
                              ('mlp', MLPClassifier(alpha=0.1))]))]
        verbose: False
```

```
/usr/local/lib/python3.8/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:684:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and
the optimization hasn't converged yet.
  warnings.warn(
[ ]: #new ensemble
     from sklearn.datasets import load_digits
     from sklearn.model_selection import train_test_split, GridSearchCV
     from sklearn.pipeline import Pipeline
     from sklearn.ensemble import StackingClassifier
     from sklearn.neural_network import MLPClassifier
     from sklearn.naive_bayes import MultinomialNB
     from sklearn.metrics import accuracy_score




     # Define the base estimators for the stacking classifier
     ensemble_estimators = [('nb', MultinomialNB()), ('mlp', MLPClassifier())]

     ensemble_pipeline = Pipeline([
         ('cv', TfidfVectorizer()),
         ('stacking', StackingClassifier(estimators=ensemble_estimators))
     ])

     # Define the grid search parameters
     ensemble_params = {
         #'cv__max_df': [0.5, 0.75, 1.0],
        # 'nb__alpha': [0.1, 0.5, 1.0],
         "cv__stop_words": [list(stop_words_custom)],
         'stacking__mlp__solver':["lbfgs"],
         'stacking__mlp__hidden_layer_sizes': [(32,)],
        # 'mlp__hidden_layer_sizes': [(50,), (100,), (200,)],
         'stacking__mlp__alpha': [0.1],
     }

     ensemble_grid = GridSearchCV(ensemble_pipeline, ensemble_params,␣
      ↪cv=5,scoring='accuracy' ,verbose=1, n_jobs=-1)

     ensemble_grid.fit(train_x, train_y)

     accuracy = round(ensemble_grid.best_score_ * 100,3)

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

     print_best_params(ensemble_grid)
```

```
Fitting 5 folds for each of 1 candidates, totalling 5 fits
```

```
/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(

The best accuracy is 94.984.
The winning parameters are {'cv__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'],
'stacking__mlp__alpha': 0.1, 'stacking__mlp__hidden_layer_sizes': (32,),
'stacking__mlp__solver': 'lbfgs'}
        cv: TfidfVectorizer(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',
…])
        cv__analyzer: 'word'
        cv__binary: False
        cv__decode_error: 'strict'
        cv__dtype: <class 'numpy.float64'>
        cv__encoding: 'utf-8'
        cv__input: 'content'
        cv__lowercase: True
        cv__max_df: 1.0
        cv__max_features: None
        cv__min_df: 1
        cv__ngram_range: (1, 1)
        cv__norm: 'l2'
        cv__preprocessor: None
        cv__smooth_idf: True
        cv__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']
        cv__strip_accents: None
        cv__sublinear_tf: False
        cv__token_pattern: '(?u)\\b\\w\\w+\\b'
        cv__tokenizer: None
        cv__use_idf: True
        cv__vocabulary: None
        memory: None
        stacking: StackingClassifier(estimators=[('nb', MultinomialNB()),
                               ('mlp',
                                MLPClassifier(alpha=0.1,
                                              hidden_layer_sizes=(32,),
                                              solver='lbfgs'))])
        stacking__cv: None
        stacking__estimators: [('nb', MultinomialNB()), ('mlp',
MLPClassifier(alpha=0.1, hidden_layer_sizes=(32,), solver='lbfgs'))]
```

```
        stacking__final_estimator: None
        stacking__mlp: MLPClassifier(alpha=0.1, hidden_layer_sizes=(32,),
solver='lbfgs')
        stacking__mlp__activation: 'relu'
        stacking__mlp__alpha: 0.1
        stacking__mlp__batch_size: 'auto'
        stacking__mlp__beta_1: 0.9
        stacking__mlp__beta_2: 0.999
        stacking__mlp__early_stopping: False
        stacking__mlp__epsilon: 1e-08
        stacking__mlp__hidden_layer_sizes: (32,)
        stacking__mlp__learning_rate: 'constant'
        stacking__mlp__learning_rate_init: 0.001
        stacking__mlp__max_fun: 15000
        stacking__mlp__max_iter: 200
        stacking__mlp__momentum: 0.9
        stacking__mlp__n_iter_no_change: 10
        stacking__mlp__nesterovs_momentum: True
        stacking__mlp__power_t: 0.5
        stacking__mlp__random_state: None
        stacking__mlp__shuffle: True
        stacking__mlp__solver: 'lbfgs'
        stacking__mlp__tol: 0.0001
        stacking__mlp__validation_fraction: 0.1
        stacking__mlp__verbose: False
        stacking__mlp__warm_start: False
        stacking__n_jobs: None
        stacking__nb: MultinomialNB()
        stacking__nb__alpha: 1.0
        stacking__nb__class_prior: None
        stacking__nb__fit_prior: True
        stacking__nb__force_alpha: 'warn'
        stacking__passthrough: False
        stacking__stack_method: 'auto'
        stacking__verbose: 0
        steps: [('cv', TfidfVectorizer(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',
…])), ('stacking', StackingClassifier(estimators=[('nb', MultinomialNB()),
                               ('mlp',
                                MLPClassifier(alpha=0.1,
                                              hidden_layer_sizes=(32,),
                                              solver='lbfgs'))]))]
        verbose: False
```

```
[ ]: #new ensemble
     from sklearn.datasets import load_digits
     from sklearn.model_selection import train_test_split, GridSearchCV
     from sklearn.pipeline import Pipeline
     from sklearn.ensemble import StackingClassifier
     from sklearn.neural_network import MLPClassifier
     from sklearn.naive_bayes import MultinomialNB
     from sklearn.metrics import accuracy_score



     # Define the base estimators for the stacking classifier
     ensemble_estimators = [('nb', MultinomialNB()), ('mlp', MLPClassifier())]

     ensemble_selector = SelectKBest(chi2)
     ensemble_normalizer = Normalizer()

     ensemble_pipeline = Pipeline([
         ('cv', TfidfVectorizer()),
         ("normalizer",ensemble_normalizer),
         ("selecter", ensemble_selector),
         ('stacking', StackingClassifier(estimators=ensemble_estimators))
     ])

     # Define the grid search parameters
     ensemble_params = {
         'cv__max_df': [0.75],
         "selecter__k":[5000],
         "cv__stop_words": [list(stop_words_custom)],
         "normalizer__norm": ['l2'],
         'cv__ngram_range': [(1,1)],
         'stacking__mlp__solver':["lbfgs"],
         'stacking__mlp__hidden_layer_sizes': [(32,)],
         'stacking__mlp__alpha': [0.1],
         'stacking__nb__alpha': [0.5],
     }

     ensemble_grid = GridSearchCV(ensemble_pipeline, ensemble_params,␣
      ↪cv=5,scoring='accuracy' ,verbose=1, n_jobs=-1)

     ensemble_grid.fit(train_x, train_y)

     accuracy = round(ensemble_grid.best_score_ * 100,3)

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

```
Fitting 5 folds for each of 1 candidates, totalling 5 fits
```

```
/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(
```

```
The best accuracy is 95.123.
The winning parameters are {'cv__max_df': 0.75, 'cv__ngram_range': (1, 1),
'cv__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'], 'normalizer__norm':
'l2', 'selecter__k': 5000, 'stacking__mlp__alpha': 0.1,
'stacking__mlp__hidden_layer_sizes': (32,), 'stacking__mlp__solver': 'lbfgs',
'stacking__nb__alpha': 0.5}
```

[ ]:
```python
#new ensemble
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.ensemble import StackingClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score



# Define the base estimators for the stacking classifier
ensemble_estimators = [('nb', MultinomialNB()), ('mlp', MLPClassifier())]

ensemble_selector = SelectKBest(chi2)
ensemble_normalizer = Normalizer()

ensemble_pipeline = Pipeline([
    ('cv', TfidfVectorizer()),
    ("normalizer",ensemble_normalizer),
    ("selecter", ensemble_selector),
    ('stacking', StackingClassifier(estimators=ensemble_estimators))
])

# Define the grid search parameters
ensemble_params = {
    'cv__max_df': [0.75],
    "selecter__k":[5000],
    "cv__stop_words": [list(stop_words_library)],
    "normalizer__norm": ['l2'],
    'cv__ngram_range': [(1,1)],
    'stacking__mlp__solver':["lbfgs"],
    'stacking__mlp__hidden_layer_sizes': [(32,)],
    'stacking__mlp__alpha': [0.1],
    'stacking__nb__alpha': [0.5],
}

ensemble_grid = GridSearchCV(ensemble_pipeline, ensemble_params,␣
 ↪cv=5,scoring='accuracy' ,verbose=1, n_jobs=-1)

ensemble_grid.fit(train_x, train_y)

accuracy = round(ensemble_grid.best_score_ * 100,3)

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

```
Fitting 5 folds for each of 1 candidates, totalling 5 fits
The best accuracy is 95.123.
The winning parameters are {'cv__max_df': 0.75, 'cv__ngram_range': (1, 1),
'cv__stop_words': ['even', 'put', 'eleven', 'won', 'didn', 'beforehand',
'toward', 'couldnt', 'mostly', 'eight', 'either', 'enough', 'your', 'while',
'been', 'anyway', 'sincere', 'hasnt', 'others', 'another', 'none', 'itself',
'as', 'formerly', 'often', 'about', 'off', 'just', 'during', 't', 'cannot',
'rather', "aren't", 'too', 'ever', 'wasn', 'less', 'yourself', 'myself', 'do',
'hereafter', "that'll", 'became', 'will', 'back', "haven't", 'seemed', 'name',
'one', 'never', 'so', 'onto', "wasn't", 'find', 'until', 'if', "won't", 'here',
'elsewhere', 'no', 'those', 'needn', 'hence', 'meanwhile', 'from', 'hereupon',
'for', 'almost', 'did', 'least', 'with', 'she', 'many', 'without', 'noone',
'thereupon', 'not', 'my', 'throughout', 'thick', 'such', 'hadn', 'us', 'all',
'now', 'twenty', 'once', 'at', 'fifty', 'anywhere', 'whereas', 'former', 'else',
'always', 'sometimes', 'please', "mightn't", 'mightn', 'd', 'same', 'other',
'few', 'nobody', 'describe', 'sometime', 'somewhere', 'etc', 'seem', 'seems',
"needn't", 'mill', 'which', 'thereafter', 'sixty', 'together', 'therein', 'two',
"shan't", 'between', 'he', 'thin', 'already', 'his', 'their', 'hereby', 'doing',
'indeed', 'first', 'latterly', 'still', 'or', 'm', 'nor', 'can', 'neither',
"hasn't", 'll', 'next', 'when', 'thru', 'over', 'hers', 'mustn', 'besides',
'could', 'side', 'ten', 'yourselves', 'move', 'nevertheless', 'ours', 'this',
'perhaps', 'fifteen', "it's", 'well', 'con', 'up', 'un', 'be', 'mine', 'around',
'has', 'whatever', 'wouldn', 'them', 'five', 'last', 'each', "you're",
'nowhere', 'shouldn', 'wherever', 'ie', 'anyone', 'again', 'were', 'via',
'theirs', 'being', 'anyhow', 'it', 'more', 'under', 'have', 'since', 'through',
'having', "you'll", 'four', 'whereby', 'anything', 'front', 'afterwards', 'a',
'does', 's', 'six', 'somehow', 'should', 'shan', 'would', 'its', 'isn', 'any',
'where', 'keep', 'per', 'also', 'among', 'only', 'except', 'must', 'though',
'take', 'amoungst', 'behind', "isn't", 'of', 'done', 'show', 'own', 'by',
"shouldn't", "weren't", 'give', 'after', 'twelve', "don't", 'thence',
"wouldn't", "you've", 'then', 'these', 'to', 'everything', 'namely', "you'd",
'beside', 'i', 'ltd', 'don', 'me', 'due', "hadn't", 'hasn', 'made', 'whoever',
'above', 'forty', 'themselves', 'both', 'hundred', 're', 'our', 'amongst',
'however', 'moreover', 'out', 'fill', "couldn't", 'down', 'whom', 'become',
'haven', 'weren', 'thus', 'ma', 'below', 'becomes', 'everywhere', 'interest',
'much', 'herein', 'yours', 'seeming', 'is', 'nine', 'full', 'ourselves', 'ain',
'latter', 'across', 'am', 'call', 'whereupon', 'something', "doesn't", 'found',
'why', 'most', 'therefore', 'co', 'thereby', 'someone', 'empty', 'on', 'who',
'towards', 'whereafter', 'go', 'there', 'cry', 'they', 'because', 'beyond',
'bottom', 'that', 'de', 'further', 'y', 'very', 'whole', 'get', 'alone', 'than',
'detail', 'and', 'part', 'whenever', 'top', 'every', 'him', 'but', 'amount',
'everyone', 'herself', 'aren', 'along', 'three', 'fire', 'against', 'we',
"she's", 'becoming', 've', 'are', 'bill', 'before', "mustn't", 'within',
'wherein', 'doesn', 'was', 'nothing', 'himself', 'the', 'whence', 'whither',
'otherwise', 'serious', 'eg', 'in', 'inc', 'into', 'o', 'some', 'upon',
'whether', 'yet', 'cant', 'several', 'how', 'had', 'may', 'whose', "should've",
'system', "didn't", 'an', 'third', 'her', 'see', 'couldn', 'although', 'you',
'might', 'what'], 'normalizer__norm': 'l2', 'selecter__k': 5000,
'stacking__mlp__alpha': 0.1, 'stacking__mlp__hidden_layer_sizes': (32,),
'stacking__mlp__solver': 'lbfgs', 'stacking__nb__alpha': 0.5}
```

[ ]:
```
#new ensemble
from sklearn.datasets import load_digits
```

```python
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.ensemble import StackingClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score



# Define the base estimators for the stacking classifier
ensemble_estimators = [('nb', MultinomialNB()), ('mlp', MLPClassifier())]

ensemble_selector = SelectKBest(chi2)
ensemble_normalizer = Normalizer()

ensemble_pipeline = Pipeline([
    ('cv', TfidfVectorizer()),
    ("normalizer",ensemble_normalizer),
    ("selecter", ensemble_selector),
    ('stacking', StackingClassifier(estimators=ensemble_estimators))
])

# Define the grid search parameters
ensemble_params = {
    'cv__max_df': [1.0],
    "selecter__k":[5000],
    "cv__stop_words": [list(stop_words_custom)],
    'cv__preprocessor': [preprocess_text],
    #    'cv__preprocessor': [preprocess_text,remove_punctuation,None],

    "cv__binary": [False],
    "normalizer__norm": ['l2'],
    'cv__ngram_range': [(1,1)],
    'stacking__mlp__solver':["lbfgs"],
    'stacking__mlp__hidden_layer_sizes': [(32,)],
    'stacking__mlp__alpha': [0.1],
    'stacking__nb__alpha': [0.1],
}

ensemble_grid = GridSearchCV(ensemble_pipeline, ensemble_params,␣
 ↪cv=5,scoring='accuracy' ,verbose=1, n_jobs=-1)

ensemble_grid.fit(train_x, train_y)

accuracy = round(ensemble_grid.best_score_ * 100,3)

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

```
Fitting 5 folds for each of 1 candidates, totalling 5 fits
```

```
/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(
```

```
The best accuracy is 95.402.
The winning parameters are {'cv__binary': False, 'cv__max_df': 1.0,
'cv__ngram_range': (1, 1), 'cv__preprocessor': <function preprocess_text at
0x7f6cc558ea60>, 'cv__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'], 'normalizer__norm':
'l2', 'selecter__k': 5000, 'stacking__mlp__alpha': 0.1,
'stacking__mlp__hidden_layer_sizes': (32,), 'stacking__mlp__solver': 'lbfgs',
'stacking__nb__alpha': 0.1}
```

[ ]:
```
print(f"The best accuracy is {accuracy}.")
```

```
The best accuracy is 95.402.
```

[ ]:
```
y_pred_new = ensemble_grid.predict(test_x)
create_test_csv(y_pred_new,"Stacking_MultiNB-MLP-05032023_02.csv")
```

```
File saved.
```

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