MultinomialNB

March 12, 2023

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

nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
```

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

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

[ ]:
```
True
```

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

train_data_initial = pd.read_csv('/content/gdrive/MyDrive/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
```

[ ]:
```
print(train_x[5])
```

```
Hi there /u/LakotaPride! Welcome to /r/Trump.  [](/sp)

Thank you for posting on r/Trump Please follow all rules and guidelines. Inform
the mods if you have any concerns. [](/sp) Join our live
[discord](https://discord.gg/kh4Wv9DavE) chat to talk to your fellow patriots!
If you have any issues please reach out.


*I am a bot, and this action was performed automatically. Please [contact the
moderators of this subreddit](/message/compose/?to=/r/trump) if you have any
questions or concerns.*
```

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

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

[ ]:
```
#initial training => 88.438
t_start = time.time()

pipe_params = {
}

vectorizer = CountVectorizer()
```

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

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

grid.fit(train_x, train_y)

t_end = time.time()

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

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

```
Fitting 5 folds for each of 1 candidates, totalling 5 fits
The best accuracy is 88.438.
The winning parameters are {}
Run time: 0.5878884792327881 seconds
```

[ ]:
```
#removing punctuation  => not good
t_start = time.time()

pipe_params = {
   'vect__preprocessor': [preprocess_text,remove_punctuation,None],
    "vect__binary": [False,True]
}

vectorizer = CountVectorizer()

pipe = Pipeline(
    [("vect", vectorizer),("clf", MultinomialNB())]
)

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

grid.fit(train_x, train_y)

t_end = time.time()

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

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

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

```
Fitting 5 folds for each of 6 candidates, totalling 30 fits
The best accuracy is 88.438.
The winning parameters are {'vect__binary': False, 'vect__preprocessor': None}
Run time: 4.767054080963135 seconds
```

[ ]:
```
train_x_punc = train_x.copy()

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

[ ]:
```
#initial training,train_x_punc => worse
t_start = time.time()

pipe_params = {
}

vectorizer = CountVectorizer()

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

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

grid.fit(train_x_punc, train_y)

t_end = time.time()

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

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

```
Fitting 5 folds for each of 1 candidates, totalling 5 fits
The best accuracy is 87.185.
The winning parameters are {}
Run time: 0.49113988876342773 seconds
```

[ ]:
```
#stop words => stop_words_library wins 90.809.
t_start = time.time()

pipe_params = {
```

```
    "vect__binary": [False],
    "vect__stop_words":␣
 ↪[list(stop_words_nltk),list(stop_words_sklearn),list(stop_words_library),None]
}

vectorizer = CountVectorizer()

pipe = Pipeline(
    [("vect", vectorizer),("clf", MultinomialNB())]
)

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

grid.fit(train_x_punc, train_y)

t_end = time.time()

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

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


print_best_params(grid)
```

```
Fitting 5 folds for each of 4 candidates, totalling 20 fits
The best accuracy is 90.809.
The winning parameters are {'vect__binary': False, 'vect__stop_words':
['against', 'find', "shan't", 'i', 't', 'whence', 'go', 'ten', 'she',
'somewhere', 'others', 'throughout', "don't", 'serious', 'whereafter', 'own',
'whole', 'should', 'eg', 'his', 'toward', 'whether', 'wherever', 'give', 'its',
'noone', 'is', 'were', "needn't", 'though', 'therein', 'afterwards',
'everywhere', "doesn't", 'ourselves', "you'll", 'found', 'isn', 'into',
"hadn't", 'once', 'are', 'to', 'as', 'down', 'can', 'three', 'don', "wasn't",
'twenty', 'yourselves', 'please', 'often', 'ie', 'an', 'one', 'forty', 'within',
'didn', 'side', 'mightn', 'while', 'sometime', 'hadn', 'all', 'only', "didn't",
'anyone', 'becoming', 'the', 'bottom', 'from', 'almost', 'still', 'describe',
'about', 'anyway', 'd', 'may', 'six', "that'll", 'everything', 'take', 'back',
'for', "isn't", 'mostly', 'eleven', 'whoever', 'whereas', 'moreover', 'why',
'otherwise', 'thus', "she's", 'whose', 'if', 'therefore', 'yet', 'become',
'even', 'five', 'first', 'in', 'something', 'together', 'inc', 'further',
'fill', 'elsewhere', 'very', 'whom', 'each', 'beside', 'some', 'have', 'con',
'latterly', 'themselves', 'hereupon', "it's", 'third', 'upon', 'seems', 'll',
'along', 'itself', 'indeed', 'seem', 'that', 'across', 'will', 'already',
'seemed', 'least', 'becomes', 'show', 'been', "aren't", 'couldnt', 'mill', 'it',
```

```
'except', 'because', 'nowhere', 'by', 'empty', 'out', 'but', 'after',
'beforehand', 'thereby', 'although', 'full', "haven't", 'latter', 'four',
'then', 'hence', 'her', 'see', 'could', 'you', 'these', 'none', 'thereupon',
'hereafter', 'per', 'shouldn', 'how', 'thence', 'was', 'those', 'nothing',
'perhaps', 'mustn', 'hers', 'doesn', 'there', 'nine', 'ma', 'whither', 'this',
'anyhow', 'interest', 'be', 'o', 'too', 'front', 'less', 'due', 'call',
'rather', 'just', 'without', 'name', 'everyone', 'being', 'over', 'when', 'him',
"mightn't", 's', 'amongst', 'amoungst', 'more', 'does', 'formerly', 'de', 'now',
'made', 'hundred', 'below', "you'd", 'through', 'anywhere', 'sincere', 'of',
'meanwhile', 'thin', 'behind', 'whenever', 'wasn', 'nor', 'until', 'among',
'so', 'yours', 'whereby', 'such', "shouldn't", 'sometimes', 'what', 'thru',
'much', 'same', 'must', 'again', 'a', 'am', 'off', 'never', 'ain', 'they',
'herself', 'etc', 'wouldn', 'thereafter', 'few', "you've", 'amount', 'namely',
'get', 'yourself', 'besides', 'my', 'than', 'alone', 'couldn', 'might', 'their',
'two', 'between', "won't", 'most', 'them', "weren't", 'herein', 'and', 'part',
'nevertheless', 'where', 'co', 'another', 'cant', 'bill', 'other', 'fire',
'several', 'did', 'no', 'up', 'cry', "should've", 'do', 'beyond', 'needn',
'neither', 'next', 'always', 'mine', 'put', 'wherein', 'hasn', "couldn't",
'onto', "you're", "hasn't", 'during', 'however', 'aren', 'thick', 'also', 'm',
'move', 'before', 'doing', 'un', 'which', 'with', 'keep', 'whereupon',
'anything', 'cannot', 'system', 'us', 'done', 'both', "wouldn't", 'here',
'ever', 'enough', 've', "mustn't", 'towards', 'having', 'either', 'hasnt',
'who', 'under', 'fifty', 'haven', 'fifteen', 'eight', 'me', 'former', 'he',
'hereby', 'became', 'or', 'top', 'any', 're', 'has', 'we', 'seeming', 'someone',
'ours', 'else', 'myself', 'above', 'since', 'had', 'our', 'your', 'not',
'would', 'many', 'around', 'detail', 'on', 'sixty', 'somehow', 'at', 'nobody',
'via', 'y', 'shan', 'twelve', 'theirs', 'last', 'ltd', 'every', 'himself',
'whatever', 'won', 'well', 'weren']}
Run time: 1.4538230895996094 seconds
        clf: MultinomialNB()
        clf__alpha: 1.0
        clf__class_prior: None
        clf__fit_prior: True
        clf__force_alpha: 'warn'
        memory: None
        steps: [('vect', CountVectorizer(stop_words=['against', 'find',
"shan't", 'i', 't', 'whence',
                                'go', 'ten', 'she', 'somewhere', 'others',
                                'throughout', "don't", 'serious', 'whereafter',
                                'own', 'whole', 'should', 'eg', 'his', 'toward',
                                'whether', 'wherever', 'give', 'its', 'noone', 'is',
                                'were', "needn't", 'though', …])), ('clf',
MultinomialNB())]
        vect: CountVectorizer(stop_words=['against', 'find', "shan't", 'i', 't',
'whence',
                                'go', 'ten', 'she', 'somewhere', 'others',
                                'throughout', "don't", 'serious', 'whereafter',
                                'own', 'whole', 'should', 'eg', 'his', 'toward',
```

```
                            'whether', 'wherever', 'give', 'its', 'noone', 'is',
                            'were', "needn't", 'though', …])
        vect__analyzer: 'word'
        vect__binary: False
        vect__decode_error: 'strict'
        vect__dtype: <class 'numpy.int64'>
        vect__encoding: 'utf-8'
        vect__input: 'content'
        vect__lowercase: True
        vect__max_df: 1.0
        vect__max_features: None
        vect__min_df: 1
        vect__ngram_range: (1, 1)
        vect__preprocessor: None
        vect__stop_words: ['against', 'find', "shan't", 'i', 't', 'whence',
'go', 'ten', 'she', 'somewhere', 'others', 'throughout', "don't", 'serious',
'whereafter', 'own', 'whole', 'should', 'eg', 'his', 'toward', 'whether',
'wherever', 'give', 'its', 'noone', 'is', 'were', "needn't", 'though',
'therein', 'afterwards', 'everywhere', "doesn't", 'ourselves', "you'll",
'found', 'isn', 'into', "hadn't", 'once', 'are', 'to', 'as', 'down', 'can',
'three', 'don', "wasn't", 'twenty', 'yourselves', 'please', 'often', 'ie', 'an',
'one', 'forty', 'within', 'didn', 'side', 'mightn', 'while', 'sometime', 'hadn',
'all', 'only', "didn't", 'anyone', 'becoming', 'the', 'bottom', 'from',
'almost', 'still', 'describe', 'about', 'anyway', 'd', 'may', 'six', "that'll",
'everything', 'take', 'back', 'for', "isn't", 'mostly', 'eleven', 'whoever',
'whereas', 'moreover', 'why', 'otherwise', 'thus', "she's", 'whose', 'if',
'therefore', 'yet', 'become', 'even', 'five', 'first', 'in', 'something',
'together', 'inc', 'further', 'fill', 'elsewhere', 'very', 'whom', 'each',
'beside', 'some', 'have', 'con', 'latterly', 'themselves', 'hereupon', "it's",
'third', 'upon', 'seems', 'll', 'along', 'itself', 'indeed', 'seem', 'that',
'across', 'will', 'already', 'seemed', 'least', 'becomes', 'show', 'been',
"aren't", 'couldnt', 'mill', 'it', 'except', 'because', 'nowhere', 'by',
'empty', 'out', 'but', 'after', 'beforehand', 'thereby', 'although', 'full',
"haven't", 'latter', 'four', 'then', 'hence', 'her', 'see', 'could', 'you',
'these', 'none', 'thereupon', 'hereafter', 'per', 'shouldn', 'how', 'thence',
'was', 'those', 'nothing', 'perhaps', 'mustn', 'hers', 'doesn', 'there', 'nine',
'ma', 'whither', 'this', 'anyhow', 'interest', 'be', 'o', 'too', 'front',
'less', 'due', 'call', 'rather', 'just', 'without', 'name', 'everyone', 'being',
'over', 'when', 'him', "mightn't", 's', 'amongst', 'amoungst', 'more', 'does',
'formerly', 'de', 'now', 'made', 'hundred', 'below', "you'd", 'through',
'anywhere', 'sincere', 'of', 'meanwhile', 'thin', 'behind', 'whenever', 'wasn',
'nor', 'until', 'among', 'so', 'yours', 'whereby', 'such', "shouldn't",
'sometimes', 'what', 'thru', 'much', 'same', 'must', 'again', 'a', 'am', 'off',
'never', 'ain', 'they', 'herself', 'etc', 'wouldn', 'thereafter', 'few',
"you've", 'amount', 'namely', 'get', 'yourself', 'besides', 'my', 'than',
'alone', 'couldn', 'might', 'their', 'two', 'between', "won't", 'most', 'them',
"weren't", 'herein', 'and', 'part', 'nevertheless', 'where', 'co', 'another',
'cant', 'bill', 'other', 'fire', 'several', 'did', 'no', 'up', 'cry',
```

```
"should've", 'do', 'beyond', 'needn', 'neither', 'next', 'always', 'mine',
'put', 'wherein', 'hasn', "couldn't", 'onto', "you're", "hasn't", 'during',
'however', 'aren', 'thick', 'also', 'm', 'move', 'before', 'doing', 'un',
'which', 'with', 'keep', 'whereupon', 'anything', 'cannot', 'system', 'us',
'done', 'both', "wouldn't", 'here', 'ever', 'enough', 've', "mustn't",
'towards', 'having', 'either', 'hasnt', 'who', 'under', 'fifty', 'haven',
'fifteen', 'eight', 'me', 'former', 'he', 'hereby', 'became', 'or', 'top',
'any', 're', 'has', 'we', 'seeming', 'someone', 'ours', 'else', 'myself',
'above', 'since', 'had', 'our', 'your', 'not', 'would', 'many', 'around',
'detail', 'on', 'sixty', 'somehow', 'at', 'nobody', 'via', 'y', 'shan',
'twelve', 'theirs', 'last', 'ltd', 'every', 'himself', 'whatever', 'won',
'well', 'weren']
        vect__strip_accents: None
        vect__token_pattern: '(?u)\\b\\w\\w+\\b'
        vect__tokenizer: None
        vect__vocabulary: None
        verbose: False
```

[ ]:
```python
# test alpha => 92.479. , alpha = 0.1
#selected 3
t_start = time.time()

pipe_params = {
    "vect__binary": [False],
    "vect__stop_words": [list(stop_words_library)],
    "clf__alpha" : [0.001, 0.01, 0.1,0.02,0.5]
}

vectorizer = CountVectorizer()

pipe = Pipeline([("vect", vectorizer),("clf", MultinomialNB())])

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

grid.fit(train_x, train_y)

t_end = time.time()

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

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

```
Fitting 5 folds for each of 5 candidates, totalling 25 fits
The best accuracy is 92.479.
The winning parameters are {'clf__alpha': 0.1, 'vect__binary': False,
```

```
'vect__stop_words': ['against', 'find', "shan't", 'i', 't', 'whence', 'go',
'ten', 'she', 'somewhere', 'others', 'throughout', "don't", 'serious',
'whereafter', 'own', 'whole', 'should', 'eg', 'his', 'toward', 'whether',
'wherever', 'give', 'its', 'noone', 'is', 'were', "needn't", 'though',
'therein', 'afterwards', 'everywhere', "doesn't", 'ourselves', "you'll",
'found', 'isn', 'into', "hadn't", 'once', 'are', 'to', 'as', 'down', 'can',
'three', 'don', "wasn't", 'twenty', 'yourselves', 'please', 'often', 'ie', 'an',
'one', 'forty', 'within', 'didn', 'side', 'mightn', 'while', 'sometime', 'hadn',
'all', 'only', "didn't", 'anyone', 'becoming', 'the', 'bottom', 'from',
'almost', 'still', 'describe', 'about', 'anyway', 'd', 'may', 'six', "that'll",
'everything', 'take', 'back', 'for', "isn't", 'mostly', 'eleven', 'whoever',
'whereas', 'moreover', 'why', 'otherwise', 'thus', "she's", 'whose', 'if',
'therefore', 'yet', 'become', 'even', 'five', 'first', 'in', 'something',
'together', 'inc', 'further', 'fill', 'elsewhere', 'very', 'whom', 'each',
'beside', 'some', 'have', 'con', 'latterly', 'themselves', 'hereupon', "it's",
'third', 'upon', 'seems', 'll', 'along', 'itself', 'indeed', 'seem', 'that',
'across', 'will', 'already', 'seemed', 'least', 'becomes', 'show', 'been',
"aren't", 'couldnt', 'mill', 'it', 'except', 'because', 'nowhere', 'by',
'empty', 'out', 'but', 'after', 'beforehand', 'thereby', 'although', 'full',
"haven't", 'latter', 'four', 'then', 'hence', 'her', 'see', 'could', 'you',
'these', 'none', 'thereupon', 'hereafter', 'per', 'shouldn', 'how', 'thence',
'was', 'those', 'nothing', 'perhaps', 'mustn', 'hers', 'doesn', 'there', 'nine',
'ma', 'whither', 'this', 'anyhow', 'interest', 'be', 'o', 'too', 'front',
'less', 'due', 'call', 'rather', 'just', 'without', 'name', 'everyone', 'being',
'over', 'when', 'him', "mightn't", 's', 'amongst', 'amoungst', 'more', 'does',
'formerly', 'de', 'now', 'made', 'hundred', 'below', "you'd", 'through',
'anywhere', 'sincere', 'of', 'meanwhile', 'thin', 'behind', 'whenever', 'wasn',
'nor', 'until', 'among', 'so', 'yours', 'whereby', 'such', "shouldn't",
'sometimes', 'what', 'thru', 'much', 'same', 'must', 'again', 'a', 'am', 'off',
'never', 'ain', 'they', 'herself', 'etc', 'wouldn', 'thereafter', 'few',
"you've", 'amount', 'namely', 'get', 'yourself', 'besides', 'my', 'than',
'alone', 'couldn', 'might', 'their', 'two', 'between', "won't", 'most', 'them',
"weren't", 'herein', 'and', 'part', 'nevertheless', 'where', 'co', 'another',
'cant', 'bill', 'other', 'fire', 'several', 'did', 'no', 'up', 'cry',
"should've", 'do', 'beyond', 'needn', 'neither', 'next', 'always', 'mine',
'put', 'wherein', 'hasn', "couldn't", 'onto', "you're", "hasn't", 'during',
'however', 'aren', 'thick', 'also', 'm', 'move', 'before', 'doing', 'un',
'which', 'with', 'keep', 'whereupon', 'anything', 'cannot', 'system', 'us',
'done', 'both', "wouldn't", 'here', 'ever', 'enough', 've', "mustn't",
'towards', 'having', 'either', 'hasnt', 'who', 'under', 'fifty', 'haven',
'fifteen', 'eight', 'me', 'former', 'he', 'hereby', 'became', 'or', 'top',
'any', 're', 'has', 'we', 'seeming', 'someone', 'ours', 'else', 'myself',
'above', 'since', 'had', 'our', 'your', 'not', 'would', 'many', 'around',
'detail', 'on', 'sixty', 'somehow', 'at', 'nobody', 'via', 'y', 'shan',
'twelve', 'theirs', 'last', 'ltd', 'every', 'himself', 'whatever', 'won',
'well', 'weren']}
Run time: 2.2182962894439697 seconds
```

[ ]:
```
# test selector = > decreased(90.669.)
t_start = time.time()

pipe_params = {
    "vect__binary": [False],
    "vect__stop_words": [list(stop_words_library)],
    "clf__alpha" : [0.001, 0.01, 0.1,0.02,0.5],
    "selecter__k":[5000,3000,6000]
}

vectorizer = CountVectorizer()
selecter = SelectKBest(chi2)

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

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

grid.fit(train_x_punc, train_y)

t_end = time.time()

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

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

```
Fitting 5 folds for each of 15 candidates, totalling 75 fits
The best accuracy is 90.669.
The winning parameters are {'clf__alpha': 0.5, 'selecter__k': 5000,
'vect__binary': False, 'vect__stop_words': ['against', 'find', "shan't", 'i',
't', 'whence', 'go', 'ten', 'she', 'somewhere', 'others', 'throughout', "don't",
'serious', 'whereafter', 'own', 'whole', 'should', 'eg', 'his', 'toward',
'whether', 'wherever', 'give', 'its', 'noone', 'is', 'were', "needn't",
'though', 'therein', 'afterwards', 'everywhere', "doesn't", 'ourselves',
"you'll", 'found', 'isn', 'into', "hadn't", 'once', 'are', 'to', 'as', 'down',
'can', 'three', 'don', "wasn't", 'twenty', 'yourselves', 'please', 'often',
'ie', 'an', 'one', 'forty', 'within', 'didn', 'side', 'mightn', 'while',
'sometime', 'hadn', 'all', 'only', "didn't", 'anyone', 'becoming', 'the',
'bottom', 'from', 'almost', 'still', 'describe', 'about', 'anyway', 'd', 'may',
'six', "that'll", 'everything', 'take', 'back', 'for', "isn't", 'mostly',
'eleven', 'whoever', 'whereas', 'moreover', 'why', 'otherwise', 'thus', "she's",
'whose', 'if', 'therefore', 'yet', 'become', 'even', 'five', 'first', 'in',
'something', 'together', 'inc', 'further', 'fill', 'elsewhere', 'very', 'whom',
'each', 'beside', 'some', 'have', 'con', 'latterly', 'themselves', 'hereupon',
```

```
"it's", 'third', 'upon', 'seems', 'll', 'along', 'itself', 'indeed', 'seem',
'that', 'across', 'will', 'already', 'seemed', 'least', 'becomes', 'show',
'been', "aren't", 'couldnt', 'mill', 'it', 'except', 'because', 'nowhere', 'by',
'empty', 'out', 'but', 'after', 'beforehand', 'thereby', 'although', 'full',
"haven't", 'latter', 'four', 'then', 'hence', 'her', 'see', 'could', 'you',
'these', 'none', 'thereupon', 'hereafter', 'per', 'shouldn', 'how', 'thence',
'was', 'those', 'nothing', 'perhaps', 'mustn', 'hers', 'doesn', 'there', 'nine',
'ma', 'whither', 'this', 'anyhow', 'interest', 'be', 'o', 'too', 'front',
'less', 'due', 'call', 'rather', 'just', 'without', 'name', 'everyone', 'being',
'over', 'when', 'him', "mightn't", 's', 'amongst', 'amoungst', 'more', 'does',
'formerly', 'de', 'now', 'made', 'hundred', 'below', "you'd", 'through',
'anywhere', 'sincere', 'of', 'meanwhile', 'thin', 'behind', 'whenever', 'wasn',
'nor', 'until', 'among', 'so', 'yours', 'whereby', 'such', "shouldn't",
'sometimes', 'what', 'thru', 'much', 'same', 'must', 'again', 'a', 'am', 'off',
'never', 'ain', 'they', 'herself', 'etc', 'wouldn', 'thereafter', 'few',
"you've", 'amount', 'namely', 'get', 'yourself', 'besides', 'my', 'than',
'alone', 'couldn', 'might', 'their', 'two', 'between', "won't", 'most', 'them',
"weren't", 'herein', 'and', 'part', 'nevertheless', 'where', 'co', 'another',
'cant', 'bill', 'other', 'fire', 'several', 'did', 'no', 'up', 'cry',
"should've", 'do', 'beyond', 'needn', 'neither', 'next', 'always', 'mine',
'put', 'wherein', 'hasn', "couldn't", 'onto', "you're", "hasn't", 'during',
'however', 'aren', 'thick', 'also', 'm', 'move', 'before', 'doing', 'un',
'which', 'with', 'keep', 'whereupon', 'anything', 'cannot', 'system', 'us',
'done', 'both', "wouldn't", 'here', 'ever', 'enough', 've', "mustn't",
'towards', 'having', 'either', 'hasnt', 'who', 'under', 'fifty', 'haven',
'fifteen', 'eight', 'me', 'former', 'he', 'hereby', 'became', 'or', 'top',
'any', 're', 'has', 'we', 'seeming', 'someone', 'ours', 'else', 'myself',
'above', 'since', 'had', 'our', 'your', 'not', 'would', 'many', 'around',
'detail', 'on', 'sixty', 'somehow', 'at', 'nobody', 'via', 'y', 'shan',
'twelve', 'theirs', 'last', 'ltd', 'every', 'himself', 'whatever', 'won',
'well', 'weren']}
Run time: 12.57723093032837 seconds
```

[ ]:
```
#testing normalizer , without : 92.479, with:92.199., 92.34. with norm max =>␣
 ↪no normalizer
t_start = time.time()

pipe_params = {
    "vect__binary": [False],
    "vect__stop_words": [list(stop_words_library)],
    "clf__alpha" : [0.001, 0.01, 0.1,0.02,0.5],
    'normalizer__norm': ['l1','l2','max']
}

vectorizer = CountVectorizer()
normalizer = Normalizer()
```

```
pipe = Pipeline([("vect", vectorizer),("normalizer", normalizer),("clf",␣
 ↪MultinomialNB())])
#pipe = Pipeline([("vect", vectorizer),("clf", MultinomialNB())])


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

grid.fit(train_x, train_y)

t_end = time.time()

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

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

```
Fitting 5 folds for each of 5 candidates, totalling 25 fits
The best accuracy is 91.779.
The winning parameters are {'clf__alpha': 0.01, 'normalizer__norm': 'l1',
'vect__binary': False, 'vect__stop_words': ['thick', 'thru', 'cant', 'below',
'ma', 'becomes', "you've", 'thus', 'fire', 'somewhere', 'latter', 'after',
'much', 'put', 'sometimes', 's', 'see', "aren't", 'seem', 'interest', 'if',
'elsewhere', 'over', 'less', "won't", 'ours', 'ain', 't', 'under', 'anyway',
'whoever', 'ourselves', 'hence', 'not', 'd', 'become', 've', 'should', 'no',
'toward', 'i', 'besides', 'therein', 'something', 'beforehand', 'out', "shan't",
'or', 'through', 'why', 'inc', 'upon', 'last', 'few', 'perhaps', 'one', 'found',
'themselves', 'find', 'again', 'now', 'while', 'same', 'doesn', 'who', 'with',
'formerly', 'eg', 'already', 'side', 'isn', 'don', 'which', "you're", 'give',
'is', 'however', 'couldnt', 'indeed', 'cry', 'nine', 'in', 'someone', 'many',
'whereby', 'before', 'further', 'the', 'whereas', 'often', 'amoungst',
'latterly', "shouldn't", 'they', 'meanwhile', 'our', 'twenty', 'herself',
'once', 'always', 'done', 'namely', 'against', 'wherein', 'still', 'wasn',
'etc', 'his', 'though', 'other', 'all', 'up', 'get', 'herein', 'can', "weren't",
'others', 'because', 'along', 'whole', 'former', 'its', "mightn't", 'keep',
'has', 'as', 'how', 'these', 'shouldn', 'me', 'wouldn', 'what', "you'll",
'seemed', 'within', 'those', 'hasn', 'every', 'hasnt', 'hundred', 'since', 'of',
'didn', "she's", 'via', 'here', 'per', 'otherwise', 'wherever', "you'd",
'whereupon', 'haven', 'never', 'anything', 'empty', 'seems', 'might', 'just',
'next', 'ltd', 'to', 'y', 'couldn', 'hadn', 'by', 'nowhere', 'among', "mustn't",
'seeming', 'it', 'call', 'theirs', 'each', 'behind', 'everything', 'amount',
'de', 'down', 'did', 'alone', "don't", 'about', 'sometime', 'an', 'also',
'will', 'weren', "doesn't", 'sincere', 'whither', 'whenever', 'thence',
'mostly', 'hereby', 'serious', 'twelve', 'doing', 'bill', 'ie', 'made',
'together', 'when', 'eight', 'thereafter', 'third', 'am', 'well', 'll',
```

```
'detail', "couldn't", 'either', 'won', 'where', 'very', 'been', 'she', 'was',
'this', 'front', 'therefore', 'sixty', 'whence', 'beyond', 'were', 'several',
'amongst', 'o', 'three', 'throughout', 're', 'into', 'he', 'shan', 'mustn',
'needn', 'own', 'do', 'anyone', 'first', 'almost', 'due', 'system', 'than',
'con', 'fifteen', 'eleven', 'enough', "needn't", 'mightn', 'most', 'more',
'are', 'everywhere', 'thin', 'that', 'yourselves', 'them', 'fill', 'nothing',
'having', 'at', "didn't", 'may', 'on', 'top', 'became', 'you', 'any', 'take',
'their', 'during', 'only', 'neither', 'whatever', 'us', 'none', 'have', 'both',
'hereupon', 'five', 'cannot', 'mill', 'although', 'co', 'from', 'somehow',
'moreover', 'onto', 'm', 'nevertheless', 'some', 'please', 'too', 'and',
'except', 'even', 'go', 'himself', 'yourself', 'hers', 'bottom', 'un',
'whether', 'another', 'around', "haven't", 'nor', 'such', "hasn't", 'beside',
'whose', 'then', 'two', 'being', 'aren', 'had', 'full', 'whom', 'ten',
'hereafter', 'could', 'there', 'else', 'rather', 'him', 'itself', 'her', 'your',
'thereupon', 'my', 'mine', 'move', 'but', 'ever', 'describe', 'show',
'afterwards', 'noone', 'six', 'thereby', 'we', 'be', "isn't", 'name', 'would',
'a', "it's", 'anywhere', 'anyhow', 'for', 'towards', "wasn't", 'so', 'off',
'yours', 'four', 'without', 'becoming', 'whereafter', "that'll", 'across',
'everyone', 'fifty', 'myself', 'yet', 'until', 'part', 'least', 'nobody',
'must', 'between', "should've", 'above', "hadn't", "wouldn't", 'back', 'does',
'forty']}
Run time: 2.6477572917938232 seconds
```

[ ]:
```
#testing lemma,stemmizer => not working
t_start = time.time()

pipe_params = {
    "vect__binary": [False],
    "vect__stop_words": [list(stop_words_library)],
    "vect__tokenizer": [LemmaTokenizer_word()],
    'selecter__k':[5000,3000],
    "clf__alpha" : [0.001, 0.01, 0.1,0.02,0.5]
}

vectorizer = CountVectorizer()
selecter = SelectKBest(chi2)


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


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

grid.fit(train_x, train_y)

t_end = time.time()
```

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

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

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits

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

The best accuracy is 89.28.
The winning parameters are {'clf__alpha': 0.001, 'selecter__k': 3000,
'vect__binary': False, 'vect__stop_words': ['thick', 'thru', 'cant', 'below',
'ma', 'becomes', "you've", 'thus', 'fire', 'somewhere', 'latter', 'after',
'much', 'put', 'sometimes', 's', 'see', "aren't", 'seem', 'interest', 'if',
'elsewhere', 'over', 'less', "won't", 'ours', 'ain', 't', 'under', 'anyway',
'whoever', 'ourselves', 'hence', 'not', 'd', 'become', 've', 'should', 'no',
'toward', 'i', 'besides', 'therein', 'something', 'beforehand', 'out', "shan't",
'or', 'through', 'why', 'inc', 'upon', 'last', 'few', 'perhaps', 'one', 'found',
'themselves', 'find', 'again', 'now', 'while', 'same', 'doesn', 'who', 'with',
'formerly', 'eg', 'already', 'side', 'isn', 'don', 'which', "you're", 'give',
'is', 'however', 'couldnt', 'indeed', 'cry', 'nine', 'in', 'someone', 'many',
'whereby', 'before', 'further', 'the', 'whereas', 'often', 'amoungst',
'latterly', "shouldn't", 'they', 'meanwhile', 'our', 'twenty', 'herself',
'once', 'always', 'done', 'namely', 'against', 'wherein', 'still', 'wasn',
'etc', 'his', 'though', 'other', 'all', 'up', 'get', 'herein', 'can', "weren't",
'others', 'because', 'along', 'whole', 'former', 'its', "mightn't", 'keep',
'has', 'as', 'how', 'these', 'shouldn', 'me', 'wouldn', 'what', "you'll",
'seemed', 'within', 'those', 'hasn', 'every', 'hasnt', 'hundred', 'since', 'of',
'didn', "she's", 'via', 'here', 'per', 'otherwise', 'wherever', "you'd",
'whereupon', 'haven', 'never', 'anything', 'empty', 'seems', 'might', 'just',
'next', 'ltd', 'to', 'y', 'couldn', 'hadn', 'by', 'nowhere', 'among', "mustn't",
'seeming', 'it', 'call', 'theirs', 'each', 'behind', 'everything', 'amount',
'de', 'down', 'did', 'alone', "don't", 'about', 'sometime', 'an', 'also',
'will', 'weren', "doesn't", 'sincere', 'whither', 'whenever', 'thence',
'mostly', 'hereby', 'serious', 'twelve', 'doing', 'bill', 'ie', 'made',
'together', 'when', 'eight', 'thereafter', 'third', 'am', 'well', 'll',
'detail', "couldn't", 'either', 'won', 'where', 'very', 'been', 'she', 'was',
```

```
'this', 'front', 'therefore', 'sixty', 'whence', 'beyond', 'were', 'several',
'amongst', 'o', 'three', 'throughout', 're', 'into', 'he', 'shan', 'mustn',
'needn', 'own', 'do', 'anyone', 'first', 'almost', 'due', 'system', 'than',
'con', 'fifteen', 'eleven', 'enough', "needn't", 'mightn', 'most', 'more',
'are', 'everywhere', 'thin', 'that', 'yourselves', 'them', 'fill', 'nothing',
'having', 'at', "didn't", 'may', 'on', 'top', 'became', 'you', 'any', 'take',
'their', 'during', 'only', 'neither', 'whatever', 'us', 'none', 'have', 'both',
'hereupon', 'five', 'cannot', 'mill', 'although', 'co', 'from', 'somehow',
'moreover', 'onto', 'm', 'nevertheless', 'some', 'please', 'too', 'and',
'except', 'even', 'go', 'himself', 'yourself', 'hers', 'bottom', 'un',
'whether', 'another', 'around', "haven't", 'nor', 'such', "hasn't", 'beside',
'whose', 'then', 'two', 'being', 'aren', 'had', 'full', 'whom', 'ten',
'hereafter', 'could', 'there', 'else', 'rather', 'him', 'itself', 'her', 'your',
'thereupon', 'my', 'mine', 'move', 'but', 'ever', 'describe', 'show',
'afterwards', 'noone', 'six', 'thereby', 'we', 'be', "isn't", 'name', 'would',
'a', "it's", 'anywhere', 'anyhow', 'for', 'towards', "wasn't", 'so', 'off',
'yours', 'four', 'without', 'becoming', 'whereafter', "that'll", 'across',
'everyone', 'fifty', 'myself', 'yet', 'until', 'part', 'least', 'nobody',
'must', 'between', "should've", 'above', "hadn't", "wouldn't", 'back', 'does',
'forty'], 'vect__tokenizer': <__main__.LemmaTokenizer_word object at
0x7f4c16e05460>}
Run time: 66.79015469551086 seconds
```

[ ]:
```
#test ngram() ,best is 92.47 , 93.176. with ngram(1,2)
#selected 1
t_start = time.time()

pipe_params = {
    "vect__binary": [False],
    "vect__stop_words": [list(stop_words_library)],
    'vect__ngram_range':[(1,1),(1,2),(1,3)],
    "clf__alpha" : [0.001, 0.01, 0.1,0.02,0.5]
}

vectorizer = CountVectorizer()


pipe = Pipeline([("vect", vectorizer),("clf", MultinomialNB())])


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

grid.fit(train_x, train_y)

t_end = time.time()

elapsed_time = t_end-t_start
```

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

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

```
Fitting 5 folds for each of 15 candidates, totalling 75 fits
The best accuracy is 93.176.
The winning parameters are {'clf__alpha': 0.5, 'vect__binary': False,
'vect__ngram_range': (1, 2), 'vect__stop_words': ['against', 'find', "shan't",
'i', 't', 'whence', 'go', 'ten', 'she', 'somewhere', 'others', 'throughout',
"don't", 'serious', 'whereafter', 'own', 'whole', 'should', 'eg', 'his',
'toward', 'whether', 'wherever', 'give', 'its', 'noone', 'is', 'were',
"needn't", 'though', 'therein', 'afterwards', 'everywhere', "doesn't",
'ourselves', "you'll", 'found', 'isn', 'into', "hadn't", 'once', 'are', 'to',
'as', 'down', 'can', 'three', 'don', "wasn't", 'twenty', 'yourselves', 'please',
'often', 'ie', 'an', 'one', 'forty', 'within', 'didn', 'side', 'mightn',
'while', 'sometime', 'hadn', 'all', 'only', "didn't", 'anyone', 'becoming',
'the', 'bottom', 'from', 'almost', 'still', 'describe', 'about', 'anyway', 'd',
'may', 'six', "that'll", 'everything', 'take', 'back', 'for', "isn't", 'mostly',
'eleven', 'whoever', 'whereas', 'moreover', 'why', 'otherwise', 'thus', "she's",
'whose', 'if', 'therefore', 'yet', 'become', 'even', 'five', 'first', 'in',
'something', 'together', 'inc', 'further', 'fill', 'elsewhere', 'very', 'whom',
'each', 'beside', 'some', 'have', 'con', 'latterly', 'themselves', 'hereupon',
"it's", 'third', 'upon', 'seems', 'll', 'along', 'itself', 'indeed', 'seem',
'that', 'across', 'will', 'already', 'seemed', 'least', 'becomes', 'show',
'been', "aren't", 'couldnt', 'mill', 'it', 'except', 'because', 'nowhere', 'by',
'empty', 'out', 'but', 'after', 'beforehand', 'thereby', 'although', 'full',
"haven't", 'latter', 'four', 'then', 'hence', 'her', 'see', 'could', 'you',
'these', 'none', 'thereupon', 'hereafter', 'per', 'shouldn', 'how', 'thence',
'was', 'those', 'nothing', 'perhaps', 'mustn', 'hers', 'doesn', 'there', 'nine',
'ma', 'whither', 'this', 'anyhow', 'interest', 'be', 'o', 'too', 'front',
'less', 'due', 'call', 'rather', 'just', 'without', 'name', 'everyone', 'being',
'over', 'when', 'him', "mightn't", 's', 'amongst', 'amoungst', 'more', 'does',
'formerly', 'de', 'now', 'made', 'hundred', 'below', "you'd", 'through',
'anywhere', 'sincere', 'of', 'meanwhile', 'thin', 'behind', 'whenever', 'wasn',
'nor', 'until', 'among', 'so', 'yours', 'whereby', 'such', "shouldn't",
'sometimes', 'what', 'thru', 'much', 'same', 'must', 'again', 'a', 'am', 'off',
'never', 'ain', 'they', 'herself', 'etc', 'wouldn', 'thereafter', 'few',
"you've", 'amount', 'namely', 'get', 'yourself', 'besides', 'my', 'than',
'alone', 'couldn', 'might', 'their', 'two', 'between', "won't", 'most', 'them',
"weren't", 'herein', 'and', 'part', 'nevertheless', 'where', 'co', 'another',
'cant', 'bill', 'other', 'fire', 'several', 'did', 'no', 'up', 'cry',
"should've", 'do', 'beyond', 'needn', 'neither', 'next', 'always', 'mine',
'put', 'wherein', 'hasn', "couldn't", 'onto', "you're", "hasn't", 'during',
'however', 'aren', 'thick', 'also', 'm', 'move', 'before', 'doing', 'un',
'which', 'with', 'keep', 'whereupon', 'anything', 'cannot', 'system', 'us',
```

```
'done', 'both', "wouldn't", 'here', 'ever', 'enough', 've', "mustn't",
'towards', 'having', 'either', 'hasnt', 'who', 'under', 'fifty', 'haven',
'fifteen', 'eight', 'me', 'former', 'he', 'hereby', 'became', 'or', 'top',
'any', 're', 'has', 'we', 'seeming', 'someone', 'ours', 'else', 'myself',
'above', 'since', 'had', 'our', 'your', 'not', 'would', 'many', 'around',
'detail', 'on', 'sixty', 'somehow', 'at', 'nobody', 'via', 'y', 'shan',
'twelve', 'theirs', 'last', 'ltd', 'every', 'himself', 'whatever', 'won',
'well', 'weren']}
Run time: 17.149862051010132 seconds
```

[ ]:
```
#test CountVectorizer =>93.176
#TfidfVectorizer with (1,1) ngram and selector chi2 =>92.058.
#selected 2
t_start = time.time()

pipe_params = {
    "vect__binary": [False],
    "vect__stop_words": [list(stop_words_library)],
    'vect__ngram_range':[(1,1)],
    "clf__alpha" : [0.01, 0.1,0.02,0.5],
    'selecter__k':[5000,3000]
}

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



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


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

grid.fit(train_x, train_y)

t_end = time.time()

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

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

```
Fitting 5 folds for each of 8 candidates, totalling 40 fits
The best accuracy is 92.058.
```

```
The winning parameters are {'clf__alpha': 0.01, 'selecter__k': 5000,
'vect__binary': False, 'vect__ngram_range': (1, 1), 'vect__stop_words':
['against', 'find', "shan't", 'i', 't', 'whence', 'go', 'ten', 'she',
'somewhere', 'others', 'throughout', "don't", 'serious', 'whereafter', 'own',
'whole', 'should', 'eg', 'his', 'toward', 'whether', 'wherever', 'give', 'its',
'noone', 'is', 'were', "needn't", 'though', 'therein', 'afterwards',
'everywhere', "doesn't", 'ourselves', "you'll", 'found', 'isn', 'into',
"hadn't", 'once', 'are', 'to', 'as', 'down', 'can', 'three', 'don', "wasn't",
'twenty', 'yourselves', 'please', 'often', 'ie', 'an', 'one', 'forty', 'within',
'didn', 'side', 'mightn', 'while', 'sometime', 'hadn', 'all', 'only', "didn't",
'anyone', 'becoming', 'the', 'bottom', 'from', 'almost', 'still', 'describe',
'about', 'anyway', 'd', 'may', 'six', "that'll", 'everything', 'take', 'back',
'for', "isn't", 'mostly', 'eleven', 'whoever', 'whereas', 'moreover', 'why',
'otherwise', 'thus', "she's", 'whose', 'if', 'therefore', 'yet', 'become',
'even', 'five', 'first', 'in', 'something', 'together', 'inc', 'further',
'fill', 'elsewhere', 'very', 'whom', 'each', 'beside', 'some', 'have', 'con',
'latterly', 'themselves', 'hereupon', "it's", 'third', 'upon', 'seems', 'll',
'along', 'itself', 'indeed', 'seem', 'that', 'across', 'will', 'already',
'seemed', 'least', 'becomes', 'show', 'been', "aren't", 'couldnt', 'mill', 'it',
'except', 'because', 'nowhere', 'by', 'empty', 'out', 'but', 'after',
'beforehand', 'thereby', 'although', 'full', "haven't", 'latter', 'four',
'then', 'hence', 'her', 'see', 'could', 'you', 'these', 'none', 'thereupon',
'hereafter', 'per', 'shouldn', 'how', 'thence', 'was', 'those', 'nothing',
'perhaps', 'mustn', 'hers', 'doesn', 'there', 'nine', 'ma', 'whither', 'this',
'anyhow', 'interest', 'be', 'o', 'too', 'front', 'less', 'due', 'call',
'rather', 'just', 'without', 'name', 'everyone', 'being', 'over', 'when', 'him',
"mightn't", 's', 'amongst', 'amoungst', 'more', 'does', 'formerly', 'de', 'now',
'made', 'hundred', 'below', "you'd", 'through', 'anywhere', 'sincere', 'of',
'meanwhile', 'thin', 'behind', 'whenever', 'wasn', 'nor', 'until', 'among',
'so', 'yours', 'whereby', 'such', "shouldn't", 'sometimes', 'what', 'thru',
'much', 'same', 'must', 'again', 'a', 'am', 'off', 'never', 'ain', 'they',
'herself', 'etc', 'wouldn', 'thereafter', 'few', "you've", 'amount', 'namely',
'get', 'yourself', 'besides', 'my', 'than', 'alone', 'couldn', 'might', 'their',
'two', 'between', "won't", 'most', 'them', "weren't", 'herein', 'and', 'part',
'nevertheless', 'where', 'co', 'another', 'cant', 'bill', 'other', 'fire',
'several', 'did', 'no', 'up', 'cry', "should've", 'do', 'beyond', 'needn',
'neither', 'next', 'always', 'mine', 'put', 'wherein', 'hasn', "couldn't",
'onto', "you're", "hasn't", 'during', 'however', 'aren', 'thick', 'also', 'm',
'move', 'before', 'doing', 'un', 'which', 'with', 'keep', 'whereupon',
'anything', 'cannot', 'system', 'us', 'done', 'both', "wouldn't", 'here',
'ever', 'enough', 've', "mustn't", 'towards', 'having', 'either', 'hasnt',
'who', 'under', 'fifty', 'haven', 'fifteen', 'eight', 'me', 'former', 'he',
'hereby', 'became', 'or', 'top', 'any', 're', 'has', 'we', 'seeming', 'someone',
'ours', 'else', 'myself', 'above', 'since', 'had', 'our', 'your', 'not',
'would', 'many', 'around', 'detail', 'on', 'sixty', 'somehow', 'at', 'nobody',
'via', 'y', 'shan', 'twelve', 'theirs', 'last', 'ltd', 'every', 'himself',
'whatever', 'won', 'well', 'weren']}
Run time: 9.161921262741089 seconds
```

[ ]:
```
#confirm 93.1
#same as selected 1
t_start = time.time()

pipe_params = {
    "vect__binary": [False],
    "vect__stop_words": [list(stop_words_library)],
    'vect__ngram_range':[(1,2)],
    "clf__alpha" : [0.5]
}

vectorizer = CountVectorizer()



pipe = Pipeline([("vect", vectorizer),("clf", MultinomialNB())])


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

grid.fit(train_x, train_y)

t_end = time.time()

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

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

```
Fitting 5 folds for each of 1 candidates, totalling 5 fits
The best accuracy is 93.176.
The winning parameters are {'clf__alpha': 0.5, 'vect__binary': False,
'vect__ngram_range': (1, 2), 'vect__stop_words': ['against', 'find', "shan't",
'i', 't', 'whence', 'go', 'ten', 'she', 'somewhere', 'others', 'throughout',
"don't", 'serious', 'whereafter', 'own', 'whole', 'should', 'eg', 'his',
'toward', 'whether', 'wherever', 'give', 'its', 'noone', 'is', 'were',
"needn't", 'though', 'therein', 'afterwards', 'everywhere', "doesn't",
'ourselves', "you'll", 'found', 'isn', 'into', "hadn't", 'once', 'are', 'to',
'as', 'down', 'can', 'three', 'don', "wasn't", 'twenty', 'yourselves', 'please',
'often', 'ie', 'an', 'one', 'forty', 'within', 'didn', 'side', 'mightn',
'while', 'sometime', 'hadn', 'all', 'only', "didn't", 'anyone', 'becoming',
'the', 'bottom', 'from', 'almost', 'still', 'describe', 'about', 'anyway', 'd',
'may', 'six', "that'll", 'everything', 'take', 'back', 'for', "isn't", 'mostly',
'eleven', 'whoever', 'whereas', 'moreover', 'why', 'otherwise', 'thus', "she's",
'whose', 'if', 'therefore', 'yet', 'become', 'even', 'five', 'first', 'in',
```

```
'something', 'together', 'inc', 'further', 'fill', 'elsewhere', 'very', 'whom',
'each', 'beside', 'some', 'have', 'con', 'latterly', 'themselves', 'hereupon',
"it's", 'third', 'upon', 'seems', 'll', 'along', 'itself', 'indeed', 'seem',
'that', 'across', 'will', 'already', 'seemed', 'least', 'becomes', 'show',
'been', "aren't", 'couldnt', 'mill', 'it', 'except', 'because', 'nowhere', 'by',
'empty', 'out', 'but', 'after', 'beforehand', 'thereby', 'although', 'full',
"haven't", 'latter', 'four', 'then', 'hence', 'her', 'see', 'could', 'you',
'these', 'none', 'thereupon', 'hereafter', 'per', 'shouldn', 'how', 'thence',
'was', 'those', 'nothing', 'perhaps', 'mustn', 'hers', 'doesn', 'there', 'nine',
'ma', 'whither', 'this', 'anyhow', 'interest', 'be', 'o', 'too', 'front',
'less', 'due', 'call', 'rather', 'just', 'without', 'name', 'everyone', 'being',
'over', 'when', 'him', "mightn't", 's', 'amongst', 'amoungst', 'more', 'does',
'formerly', 'de', 'now', 'made', 'hundred', 'below', "you'd", 'through',
'anywhere', 'sincere', 'of', 'meanwhile', 'thin', 'behind', 'whenever', 'wasn',
'nor', 'until', 'among', 'so', 'yours', 'whereby', 'such', "shouldn't",
'sometimes', 'what', 'thru', 'much', 'same', 'must', 'again', 'a', 'am', 'off',
'never', 'ain', 'they', 'herself', 'etc', 'wouldn', 'thereafter', 'few',
"you've", 'amount', 'namely', 'get', 'yourself', 'besides', 'my', 'than',
'alone', 'couldn', 'might', 'their', 'two', 'between', "won't", 'most', 'them',
"weren't", 'herein', 'and', 'part', 'nevertheless', 'where', 'co', 'another',
'cant', 'bill', 'other', 'fire', 'several', 'did', 'no', 'up', 'cry',
"should've", 'do', 'beyond', 'needn', 'neither', 'next', 'always', 'mine',
'put', 'wherein', 'hasn', "couldn't", 'onto', "you're", "hasn't", 'during',
'however', 'aren', 'thick', 'also', 'm', 'move', 'before', 'doing', 'un',
'which', 'with', 'keep', 'whereupon', 'anything', 'cannot', 'system', 'us',
'done', 'both', "wouldn't", 'here', 'ever', 'enough', 've', "mustn't",
'towards', 'having', 'either', 'hasnt', 'who', 'under', 'fifty', 'haven',
'fifteen', 'eight', 'me', 'former', 'he', 'hereby', 'became', 'or', 'top',
'any', 're', 'has', 'we', 'seeming', 'someone', 'ours', 'else', 'myself',
'above', 'since', 'had', 'our', 'your', 'not', 'would', 'many', 'around',
'detail', 'on', 'sixty', 'somehow', 'at', 'nobody', 'via', 'y', 'shan',
'twelve', 'theirs', 'last', 'ltd', 'every', 'himself', 'whatever', 'won',
'well', 'weren']}
Run time: 1.3021674156188965 seconds
```

[ ]:
```
#test selector
#[chi2, f_classif, mutual_info_classif, f_regression, mutual_info_regression]
#fclassic : 91.225. chi2: 91.084
t_start = time.time()

pipe_params = {
    "vect__binary": [False],
    "vect__stop_words": [list(stop_words_library)],
    'vect__ngram_range':[(1,2)],
    "selecter__score_func": [mutual_info_classif],
    "selecter__k":[5000,3000],
    "clf__alpha" : [0.01, 0.1,0.02,0.5]
```

```
}

vectorizer = CountVectorizer()
selecter = SelectKBest()


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


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

grid.fit(train_x, train_y)

t_end = time.time()

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

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

```
Fitting 5 folds for each of 8 candidates, totalling 40 fits
The best accuracy is 90.807.
The winning parameters are {'clf__alpha': 0.02, 'selecter__k': 5000,
'selecter__score_func': <function mutual_info_classif at 0x7f4c2cdaf550>,
'vect__binary': False, 'vect__ngram_range': (1, 2), 'vect__stop_words':
['thick', 'thru', 'cant', 'below', 'ma', 'becomes', "you've", 'thus', 'fire',
'somewhere', 'latter', 'after', 'much', 'put', 'sometimes', 's', 'see',
"aren't", 'seem', 'interest', 'if', 'elsewhere', 'over', 'less', "won't",
'ours', 'ain', 't', 'under', 'anyway', 'whoever', 'ourselves', 'hence', 'not',
'd', 'become', 've', 'should', 'no', 'toward', 'i', 'besides', 'therein',
'something', 'beforehand', 'out', "shan't", 'or', 'through', 'why', 'inc',
'upon', 'last', 'few', 'perhaps', 'one', 'found', 'themselves', 'find', 'again',
'now', 'while', 'same', 'doesn', 'who', 'with', 'formerly', 'eg', 'already',
'side', 'isn', 'don', 'which', "you're", 'give', 'is', 'however', 'couldnt',
'indeed', 'cry', 'nine', 'in', 'someone', 'many', 'whereby', 'before',
'further', 'the', 'whereas', 'often', 'amoungst', 'latterly', "shouldn't",
'they', 'meanwhile', 'our', 'twenty', 'herself', 'once', 'always', 'done',
'namely', 'against', 'wherein', 'still', 'wasn', 'etc', 'his', 'though',
'other', 'all', 'up', 'get', 'herein', 'can', "weren't", 'others', 'because',
'along', 'whole', 'former', 'its', "mightn't", 'keep', 'has', 'as', 'how',
'these', 'shouldn', 'me', 'wouldn', 'what', "you'll", 'seemed', 'within',
'those', 'hasn', 'every', 'hasnt', 'hundred', 'since', 'of', 'didn', "she's",
'via', 'here', 'per', 'otherwise', 'wherever', "you'd", 'whereupon', 'haven',
'never', 'anything', 'empty', 'seems', 'might', 'just', 'next', 'ltd', 'to',
```

```
'y', 'couldn', 'hadn', 'by', 'nowhere', 'among', "mustn't", 'seeming', 'it',
'call', 'theirs', 'each', 'behind', 'everything', 'amount', 'de', 'down', 'did',
'alone', "don't", 'about', 'sometime', 'an', 'also', 'will', 'weren', "doesn't",
'sincere', 'whither', 'whenever', 'thence', 'mostly', 'hereby', 'serious',
'twelve', 'doing', 'bill', 'ie', 'made', 'together', 'when', 'eight',
'thereafter', 'third', 'am', 'well', 'll', 'detail', "couldn't", 'either',
'won', 'where', 'very', 'been', 'she', 'was', 'this', 'front', 'therefore',
'sixty', 'whence', 'beyond', 'were', 'several', 'amongst', 'o', 'three',
'throughout', 're', 'into', 'he', 'shan', 'mustn', 'needn', 'own', 'do',
'anyone', 'first', 'almost', 'due', 'system', 'than', 'con', 'fifteen',
'eleven', 'enough', "needn't", 'mightn', 'most', 'more', 'are', 'everywhere',
'thin', 'that', 'yourselves', 'them', 'fill', 'nothing', 'having', 'at',
"didn't", 'may', 'on', 'top', 'became', 'you', 'any', 'take', 'their', 'during',
'only', 'neither', 'whatever', 'us', 'none', 'have', 'both', 'hereupon', 'five',
'cannot', 'mill', 'although', 'co', 'from', 'somehow', 'moreover', 'onto', 'm',
'nevertheless', 'some', 'please', 'too', 'and', 'except', 'even', 'go',
'himself', 'yourself', 'hers', 'bottom', 'un', 'whether', 'another', 'around',
"haven't", 'nor', 'such', "hasn't", 'beside', 'whose', 'then', 'two', 'being',
'aren', 'had', 'full', 'whom', 'ten', 'hereafter', 'could', 'there', 'else',
'rather', 'him', 'itself', 'her', 'your', 'thereupon', 'my', 'mine', 'move',
'but', 'ever', 'describe', 'show', 'afterwards', 'noone', 'six', 'thereby',
'we', 'be', "isn't", 'name', 'would', 'a', "it's", 'anywhere', 'anyhow', 'for',
'towards', "wasn't", 'so', 'off', 'yours', 'four', 'without', 'becoming',
'whereafter', "that'll", 'across', 'everyone', 'fifty', 'myself', 'yet',
'until', 'part', 'least', 'nobody', 'must', 'between', "should've", 'above',
"hadn't", "wouldn't", 'back', 'does', 'forty']}
Run time: 1804.328807592392 seconds
```

[ ]:
```python
#test fit prior => does not improve
t_start = time.time()

pipe_params = {
    "vect__binary": [False],
    "vect__stop_words": [list(stop_words_library)],
    'vect__ngram_range':[(1,1)],
    "clf__alpha" : [0.01],
    "clf__fit_prior" : [True,False],
    'selecter__k':[5000]
}

vectorizer = TfidfVectorizer()
normalizer = Normalizer()
selecter = SelectKBest(chi2)
```

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


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

grid.fit(train_x, train_y)

t_end = time.time()

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

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

```
Fitting 5 folds for each of 2 candidates, totalling 10 fits
The best accuracy is 92.058.
The winning parameters are {'clf__alpha': 0.01, 'clf__fit_prior': True,
'selecter__k': 5000, 'vect__binary': False, 'vect__ngram_range': (1, 1),
'vect__stop_words': ['against', 'find', "shan't", 'i', 't', 'whence', 'go',
'ten', 'she', 'somewhere', 'others', 'throughout', "don't", 'serious',
'whereafter', 'own', 'whole', 'should', 'eg', 'his', 'toward', 'whether',
'wherever', 'give', 'its', 'noone', 'is', 'were', "needn't", 'though',
'therein', 'afterwards', 'everywhere', "doesn't", 'ourselves', "you'll",
'found', 'isn', 'into', "hadn't", 'once', 'are', 'to', 'as', 'down', 'can',
'three', 'don', "wasn't", 'twenty', 'yourselves', 'please', 'often', 'ie', 'an',
'one', 'forty', 'within', 'didn', 'side', 'mightn', 'while', 'sometime', 'hadn',
'all', 'only', "didn't", 'anyone', 'becoming', 'the', 'bottom', 'from',
'almost', 'still', 'describe', 'about', 'anyway', 'd', 'may', 'six', "that'll",
'everything', 'take', 'back', 'for', "isn't", 'mostly', 'eleven', 'whoever',
'whereas', 'moreover', 'why', 'otherwise', 'thus', "she's", 'whose', 'if',
'therefore', 'yet', 'become', 'even', 'five', 'first', 'in', 'something',
'together', 'inc', 'further', 'fill', 'elsewhere', 'very', 'whom', 'each',
'beside', 'some', 'have', 'con', 'latterly', 'themselves', 'hereupon', "it's",
'third', 'upon', 'seems', 'll', 'along', 'itself', 'indeed', 'seem', 'that',
'across', 'will', 'already', 'seemed', 'least', 'becomes', 'show', 'been',
"aren't", 'couldnt', 'mill', 'it', 'except', 'because', 'nowhere', 'by',
'empty', 'out', 'but', 'after', 'beforehand', 'thereby', 'although', 'full',
"haven't", 'latter', 'four', 'then', 'hence', 'her', 'see', 'could', 'you',
'these', 'none', 'thereupon', 'hereafter', 'per', 'shouldn', 'how', 'thence',
'was', 'those', 'nothing', 'perhaps', 'mustn', 'hers', 'doesn', 'there', 'nine',
'ma', 'whither', 'this', 'anyhow', 'interest', 'be', 'o', 'too', 'front',
'less', 'due', 'call', 'rather', 'just', 'without', 'name', 'everyone', 'being',
'over', 'when', 'him', "mightn't", 's', 'amongst', 'amoungst', 'more', 'does',
'formerly', 'de', 'now', 'made', 'hundred', 'below', "you'd", 'through',
```

```
'anywhere', 'sincere', 'of', 'meanwhile', 'thin', 'behind', 'whenever', 'wasn',
'nor', 'until', 'among', 'so', 'yours', 'whereby', 'such', "shouldn't",
'sometimes', 'what', 'thru', 'much', 'same', 'must', 'again', 'a', 'am', 'off',
'never', 'ain', 'they', 'herself', 'etc', 'wouldn', 'thereafter', 'few',
"you've", 'amount', 'namely', 'get', 'yourself', 'besides', 'my', 'than',
'alone', 'couldn', 'might', 'their', 'two', 'between', "won't", 'most', 'them',
"weren't", 'herein', 'and', 'part', 'nevertheless', 'where', 'co', 'another',
'cant', 'bill', 'other', 'fire', 'several', 'did', 'no', 'up', 'cry',
"should've", 'do', 'beyond', 'needn', 'neither', 'next', 'always', 'mine',
'put', 'wherein', 'hasn', "couldn't", 'onto', "you're", "hasn't", 'during',
'however', 'aren', 'thick', 'also', 'm', 'move', 'before', 'doing', 'un',
'which', 'with', 'keep', 'whereupon', 'anything', 'cannot', 'system', 'us',
'done', 'both', "wouldn't", 'here', 'ever', 'enough', 've', "mustn't",
'towards', 'having', 'either', 'hasnt', 'who', 'under', 'fifty', 'haven',
'fifteen', 'eight', 'me', 'former', 'he', 'hereby', 'became', 'or', 'top',
'any', 're', 'has', 'we', 'seeming', 'someone', 'ours', 'else', 'myself',
'above', 'since', 'had', 'our', 'your', 'not', 'would', 'many', 'around',
'detail', 'on', 'sixty', 'somehow', 'at', 'nobody', 'via', 'y', 'shan',
'twelve', 'theirs', 'last', 'ltd', 'every', 'himself', 'whatever', 'won',
'well', 'weren']}
Run time: 3.5223331451416016 seconds
```

[ ]:
```python
#final test before selecting 93.17
t_start = time.time()

pipe_params = {
    "vect__binary": [False],
    "vect__stop_words": [list(stop_words_library)],
    'vect__preprocessor': [preprocess_text,remove_punctuation,None],
    'vect__ngram_range':[(1,2)],
    "clf__alpha" : [0.5]
}

vectorizer = CountVectorizer()



pipe = Pipeline([("vect", vectorizer),("clf", MultinomialNB())])


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

grid.fit(train_x, train_y)

t_end = time.time()

elapsed_time = t_end-t_start
```

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

print(f"The best accuracy is {accuracy}.")
print(f"The winning parameters are {grid.best_params_}")
print(f"Run time: {elapsed_time} seconds")
y_pred = grid.predict(test_x)
create_test_csv(y_pred,"MultinomialNB_93.csv")
```

```
Fitting 5 folds for each of 3 candidates, totalling 15 fits
The best accuracy is 93.176.
The winning parameters are {'clf__alpha': 0.5, 'vect__binary': False,
'vect__ngram_range': (1, 2), 'vect__preprocessor': None, 'vect__stop_words':
['against', 'find', "shan't", 'i', 't', 'whence', 'go', 'ten', 'she',
'somewhere', 'others', 'throughout', "don't", 'serious', 'whereafter', 'own',
'whole', 'should', 'eg', 'his', 'toward', 'whether', 'wherever', 'give', 'its',
'noone', 'is', 'were', "needn't", 'though', 'therein', 'afterwards',
'everywhere', "doesn't", 'ourselves', "you'll", 'found', 'isn', 'into',
"hadn't", 'once', 'are', 'to', 'as', 'down', 'can', 'three', 'don', "wasn't",
'twenty', 'yourselves', 'please', 'often', 'ie', 'an', 'one', 'forty', 'within',
'didn', 'side', 'mightn', 'while', 'sometime', 'hadn', 'all', 'only', "didn't",
'anyone', 'becoming', 'the', 'bottom', 'from', 'almost', 'still', 'describe',
'about', 'anyway', 'd', 'may', 'six', "that'll", 'everything', 'take', 'back',
'for', "isn't", 'mostly', 'eleven', 'whoever', 'whereas', 'moreover', 'why',
'otherwise', 'thus', "she's", 'whose', 'if', 'therefore', 'yet', 'become',
'even', 'five', 'first', 'in', 'something', 'together', 'inc', 'further',
'fill', 'elsewhere', 'very', 'whom', 'each', 'beside', 'some', 'have', 'con',
'latterly', 'themselves', 'hereupon', "it's", 'third', 'upon', 'seems', 'll',
'along', 'itself', 'indeed', 'seem', 'that', 'across', 'will', 'already',
'seemed', 'least', 'becomes', 'show', 'been', "aren't", 'couldnt', 'mill', 'it',
'except', 'because', 'nowhere', 'by', 'empty', 'out', 'but', 'after',
'beforehand', 'thereby', 'although', 'full', "haven't", 'latter', 'four',
'then', 'hence', 'her', 'see', 'could', 'you', 'these', 'none', 'thereupon',
'hereafter', 'per', 'shouldn', 'how', 'thence', 'was', 'those', 'nothing',
'perhaps', 'mustn', 'hers', 'doesn', 'there', 'nine', 'ma', 'whither', 'this',
'anyhow', 'interest', 'be', 'o', 'too', 'front', 'less', 'due', 'call',
'rather', 'just', 'without', 'name', 'everyone', 'being', 'over', 'when', 'him',
"mightn't", 's', 'amongst', 'amoungst', 'more', 'does', 'formerly', 'de', 'now',
'made', 'hundred', 'below', "you'd", 'through', 'anywhere', 'sincere', 'of',
'meanwhile', 'thin', 'behind', 'whenever', 'wasn', 'nor', 'until', 'among',
'so', 'yours', 'whereby', 'such', "shouldn't", 'sometimes', 'what', 'thru',
'much', 'same', 'must', 'again', 'a', 'am', 'off', 'never', 'ain', 'they',
'herself', 'etc', 'wouldn', 'thereafter', 'few', "you've", 'amount', 'namely',
'get', 'yourself', 'besides', 'my', 'than', 'alone', 'couldn', 'might', 'their',
'two', 'between', "won't", 'most', 'them', "weren't", 'herein', 'and', 'part',
'nevertheless', 'where', 'co', 'another', 'cant', 'bill', 'other', 'fire',
'several', 'did', 'no', 'up', 'cry', "should've", 'do', 'beyond', 'needn',
'neither', 'next', 'always', 'mine', 'put', 'wherein', 'hasn', "couldn't",
```

```
'onto', "you're", "hasn't", 'during', 'however', 'aren', 'thick', 'also', 'm',
'move', 'before', 'doing', 'un', 'which', 'with', 'keep', 'whereupon',
'anything', 'cannot', 'system', 'us', 'done', 'both', "wouldn't", 'here',
'ever', 'enough', 've', "mustn't", 'towards', 'having', 'either', 'hasnt',
'who', 'under', 'fifty', 'haven', 'fifteen', 'eight', 'me', 'former', 'he',
'hereby', 'became', 'or', 'top', 'any', 're', 'has', 'we', 'seeming', 'someone',
'ours', 'else', 'myself', 'above', 'since', 'had', 'our', 'your', 'not',
'would', 'many', 'around', 'detail', 'on', 'sixty', 'somehow', 'at', 'nobody',
'via', 'y', 'shan', 'twelve', 'theirs', 'last', 'ltd', 'every', 'himself',
'whatever', 'won', 'well', 'weren']}
Run time: 4.155819654464722 seconds
File saved.
```

[ ]:
```
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
```

[ ]:
```
#test custom dictionary => 94.01
#selected =>4
t_start = time.time()
```

```
pipe_params = {
    "vect__binary": [False],
    "vect__stop_words": [list(stop_words_custom)],
    'vect__preprocessor': [preprocess_text,remove_punctuation,None],
    'vect__ngram_range':[(1,1)],
    "clf__alpha" : [0.5]
}

vectorizer = CountVectorizer()



pipe = Pipeline([("vect", vectorizer),("clf", MultinomialNB())])


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

grid.fit(train_x, train_y)

t_end = time.time()

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

print(f"The best accuracy is {accuracy}.")
print(f"The winning parameters are {grid.best_params_}")
print(f"Run time: {elapsed_time} seconds")
y_pred = grid.predict(test_x)
create_test_csv(y_pred,"MultinomialNB_without.csv")
```

```
Fitting 5 folds for each of 3 candidates, totalling 15 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 94.011.
The winning parameters are {'clf__alpha': 0.5, 'vect__binary': False,
'vect__ngram_range': (1, 1), 'vect__preprocessor': None, 'vect__stop_words':
['i', "i'll", "i'd", "i'm", "i've", 'ive', 'me', 'myself', 'you', "you'll",
"you'd", "you're", "you've", 'yourself', 'he', "he'll", "he'd", "he's", 'him',
'she', "she'll", "she'd", "she's", 'her', 'it', "it'll", "it'd", "it's",
'itself', 'oneself', 'we', "we'll", "we'd", "we're", "we've", 'us', 'ourselves',
'they', "they'll", "they'd", "they're", "they've", 'them', 'themselves',
```

'everyone', "everyone's", 'everybody', "everybody's", 'someone', "someone's",
'somebody', "somebody's", 'nobody', "nobody's", 'anyone', "anyone's",
'everything', "everything's", 'something', "something's", 'nothing',
"nothing's", 'anything', "anything's", 'a', 'an', 'the', 'this', 'that',
"that's", 'these', 'those', 'my', 'your', 'yours', 'his', 'hers', 'its', 'our',
'ours', 'own', 'their', 'theirs', 'few', 'much', 'many', 'lot', 'lots', 'some',
'any', 'enough', 'all', 'both', 'half', 'either', 'neither', 'each', 'every',
'certain', 'other', 'another', 'such', 'several', 'multiple', 'rather', 'quite',
'aboard', 'about', 'above', 'across', 'after', 'against', 'along', 'amid',
'amidst', 'among', 'amongst', 'anti', 'around', 'as', 'at', 'away', 'before',
'behind', 'below', 'beneath', 'beside', 'besides', 'between', 'beyond', 'but',
'by', 'concerning', 'considering', 'despite', 'down', 'during', 'except',
'excepting', 'excluding', 'far', 'following', 'for', 'from', 'here', "here's",
'in', 'inside', 'into', 'left', 'like', 'minus', 'near', 'of', 'off', 'on',
'onto', 'opposite', 'out', 'outside', 'over', 'past', 'per', 'plus',
'regarding', 'right', 'since', 'than', 'there', "there's", 'through', 'to',
'toward', 'towards', 'under', 'underneath', 'unlike', 'until', 'up', 'upon',
'versus', 'via', 'with', 'within', 'without', 'may', 'might', 'will', "won't",
'would', "wouldn't", 'can', "can't", 'cannot', 'could', "couldn't", 'should',
"shouldn't", 'must', "must've", 'be', 'being', 'been', 'am', 'are', "aren't",
"ain't", 'is', "isn't", 'was', "wasn't", 'were', "weren't", 'do', 'doing',
"don't", 'does', "doesn't", 'did', "didn't", 'done', 'have', "haven't",
'having', 'has', "hasn't", 'had', "hadn't", 'get', 'getting', 'gets', 'got',
'gotten', 'go', 'going', 'gonna', 'goes', 'went', 'gone', 'make', 'making',
'makes', 'made', 'take', 'taking', 'takes', 'took', 'taken', 'need', 'needing',
'needs', 'needed', 'use', 'using', 'uses', 'used', 'want', 'wanna', 'wanting',
'wants', 'let', 'lets', 'letting', "let's", 'suppose', 'supposing', 'supposes',
'supposed', 'seem', 'seeming', 'seems', 'seemed', 'say', 'saying', 'says',
'said', 'know', 'knowing', 'knows', 'knew', 'known', 'look', 'looking',
'looked', 'think', 'thinking', 'thinks', 'thought', 'feel', 'feels', 'felt',
'based', 'put', 'puts', 'who', "who's", "who've", "who'd", 'whoever',
"whoever's", 'whom', 'whomever', "whomever's", 'whose', 'whosever',
"whosever's", 'when', 'whenever', 'which', 'whichever', 'where', "where's",
"where'd", 'wherever', 'why', "why's", "why'd", 'whyever', 'what', "what's",
'whatever', 'whence', 'how', "how's", "how'd", 'however', 'whether',
'whatsoever', 'and', 'or', 'not', 'because', 'also', 'always', 'never', 'only',
'really', 'very', 'greatly', 'extremely', 'somewhat', 'no', 'nope', 'nah',
'yes', 'yep', 'yeh', 'yeah', 'maybe', 'perhaps', 'more', 'most', 'less',
'least', 'good', 'great', 'well', 'better', 'best', 'bad', 'worse', 'worst',
'too', 'thru', 'though', 'although', 'yet', 'already', 'then', 'even', 'now',
'sometimes', 'still', 'together', 'altogether', 'entirely', 'fully', 'entire',
'whole', 'completely', 'utterly', 'seemingly', 'apparently', 'clearly',
'obviously', 'actually', 'actual', 'usually', 'usual', 'literally', 'honestly',
'absolutely', 'definitely', 'generally', 'totally', 'finally', 'basically',
'essentially', 'fundamentally', 'automatically', 'immediately', 'necessarily',
'primarily', 'normally', 'perfectly', 'constantly', 'particularly',
'eventually', 'hopefully', 'mainly', 'typically', 'specifically', 'differently',
'appropriately', 'plenty', 'certainly', 'unfortunately', 'ultimately',

```
'unlikely', 'likely', 'potentially', 'fortunately', 'personally', 'directly',
'indirectly', 'nearly', 'closely', 'slightly', 'probably', 'possibly',
'especially', 'frequently', 'often', 'oftentimes', 'seldom', 'rarely', 'sure',
'while', 'whilst', 'able', 'unable', 'else', 'ever', 'once', 'twice', 'thrice',
'almost', 'again', 'instead', 'next', 'previous', 'unless', 'somehow', 'anyhow',
'anywhere', 'somewhere', 'everywhere', 'nowhere', 'further', 'anymore', 'later',
'ago', 'ahead', 'just', 'same', 'different', 'big', 'small', 'little', 'tiny',
'large', 'huge', 'pretty', 'mostly', 'anyway', 'anyways', 'otherwise',
'regardless', 'throughout', 'additionally', 'moreover', 'furthermore',
'meanwhile', 'afterwards', 'thing', "thing's", 'things', 'stuff', "other's",
'others', "another's", 'total', '', 'false', 'none', 'way', 'kind', 'zero',
'zeros', 'zeroes', 'one', 'ones', 'two', 'three', 'four', 'five', 'six',
'seven', 'eight', 'nine', 'ten', 'twenty', 'thirty', 'forty', 'fifty', 'sixty',
'seventy', 'eighty', 'ninety', 'hundred', 'hundreds', 'thousand', 'thousands',
'million', 'millions', 'first', 'last', 'second', 'third', 'fourth', 'fifth',
'sixth', 'seventh', 'eigth', 'ninth', 'tenth', 'firstly', 'secondly', 'thirdly',
'lastly', 'hello', 'hi', 'hey', 'sup', 'yo', 'greetings', 'please', 'okay',
'ok', "y'all", 'lol', 'rofl', 'thank', 'thanks', 'alright', 'kinda', 'dont',
'sorry', 'idk', 'tldr', 'tl', 'dr', 'tbh', 'dude', 'tho', 'aka', 'plz', 'pls',
'bit', 'don', 'www', 'https', 'http', 'com', 'etchtml', 'reddit', 'subreddit',
'subreddits', 'comments', 'reply', 'replies', 'thread', 'threads', 'post',
'posts', 'website', 'websites', 'web site', 'web sites']}
Run time: 5.304453372955322 seconds
File saved.
```

[ ]:
```
#test custom dictionary => 94.01
#selected =>4
t_start = time.time()

pipe_params = {
    "vect__binary": [False],
    "vect__stop_words": [list(stop_words_custom)],
    #'vect__preprocessor': [preprocess_text,remove_punctuation,None],
    'vect__preprocessor': [remove_punctuation],
    'vect__ngram_range':[(1,1)],
    "clf__alpha" : [0.5]
}

vectorizer = CountVectorizer()



pipe = Pipeline([("vect", vectorizer),("clf", MultinomialNB())])


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

```
grid.fit(train_x, train_y)

t_end = time.time()

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

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

```
Fitting 5 folds for each of 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 ['aint', 'anothers', 'anyones',
'anythings', 'arent', 'cant', 'couldnt', 'didnt', 'doesnt', 'everybodys',
'everyones', 'everythings', 'hadnt', 'hasnt', 'havent', 'hed', 'hell', 'heres',
'hes', 'howd', 'hows', 'id', 'ill', 'im', 'isnt', 'itd', 'itll', 'mustve',
'nobodys', 'nothings', 'shed', 'shell', 'shes', 'shouldnt', 'site', 'sites',
'somebodys', 'someones', 'somethings', 'thats', 'theres', 'theyd', 'theyll',
'theyre', 'theyve', 'wasnt', 'web', 'wed', 'werent', 'weve', 'whats', 'whered',
'wheres', 'whod', 'whoevers', 'whomevers', 'whos', 'whosevers', 'whove', 'whyd',
'whys', 'wont', 'wouldnt', 'yall', 'youd', 'youll', 'youre', 'youve'] not in
stop_words.
  warnings.warn(
```

```
The best accuracy is 89.558.
The winning parameters are {'clf__alpha': 0.5, 'vect__binary': False,
'vect__ngram_range': (1, 1), 'vect__preprocessor': <function remove_punctuation
at 0x7f6bd75ee790>, 'vect__stop_words': ['i', "i'll", "i'd", "i'm", "i've",
'ive', 'me', 'myself', 'you', "you'll", "you'd", "you're", "you've", 'yourself',
'he', "he'll", "he'd", "he's", 'him', 'she', "she'll", "she'd", "she's", 'her',
'it', "it'll", "it'd", "it's", 'itself', 'oneself', 'we', "we'll", "we'd",
"we're", "we've", 'us', 'ourselves', 'they', "they'll", "they'd", "they're",
"they've", 'them', 'themselves', 'everyone', "everyone's", 'everybody',
"everybody's", 'someone', "someone's", 'somebody', "somebody's", 'nobody',
"nobody's", 'anyone', "anyone's", 'everything', "everything's", 'something',
"something's", 'nothing', "nothing's", 'anything', "anything's", 'a', 'an',
'the', 'this', 'that', "that's", 'these', 'those', 'my', 'your', 'yours', 'his',
'hers', 'its', 'our', 'ours', 'own', 'their', 'theirs', 'few', 'much', 'many',
'lot', 'lots', 'some', 'any', 'enough', 'all', 'both', 'half', 'either',
'neither', 'each', 'every', 'certain', 'other', 'another', 'such', 'several',
'multiple', 'rather', 'quite', 'aboard', 'about', 'above', 'across', 'after',
'against', 'along', 'amid', 'amidst', 'among', 'amongst', 'anti', 'around',
'as', 'at', 'away', 'before', 'behind', 'below', 'beneath', 'beside', 'besides',
'between', 'beyond', 'but', 'by', 'concerning', 'considering', 'despite',
'down', 'during', 'except', 'excepting', 'excluding', 'far', 'following', 'for',
```

'from', 'here', "here's", 'in', 'inside', 'into', 'left', 'like', 'minus',
'near', 'of', 'off', 'on', 'onto', 'opposite', 'out', 'outside', 'over', 'past',
'per', 'plus', 'regarding', 'right', 'since', 'than', 'there', "there's",
'through', 'to', 'toward', 'towards', 'under', 'underneath', 'unlike', 'until',
'up', 'upon', 'versus', 'via', 'with', 'within', 'without', 'may', 'might',
'will', "won't", 'would', "wouldn't", 'can', "can't", 'cannot', 'could',
"couldn't", 'should', "shouldn't", 'must', "must've", 'be', 'being', 'been',
'am', 'are', "aren't", "ain't", 'is', "isn't", 'was', "wasn't", 'were',
"weren't", 'do', 'doing', "don't", 'does', "doesn't", 'did', "didn't", 'done',
'have', "haven't", 'having', 'has', "hasn't", 'had', "hadn't", 'get', 'getting',
'gets', 'got', 'gotten', 'go', 'going', 'gonna', 'goes', 'went', 'gone', 'make',
'making', 'makes', 'made', 'take', 'taking', 'takes', 'took', 'taken', 'need',
'needing', 'needs', 'needed', 'use', 'using', 'uses', 'used', 'want', 'wanna',
'wanting', 'wants', 'let', 'lets', 'letting', "let's", 'suppose', 'supposing',
'supposes', 'supposed', 'seem', 'seeming', 'seems', 'seemed', 'say', 'saying',
'says', 'said', 'know', 'knowing', 'knows', 'knew', 'known', 'look', 'looking',
'looked', 'think', 'thinking', 'thinks', 'thought', 'feel', 'feels', 'felt',
'based', 'put', 'puts', 'who', "who's", "who've", "who'd", 'whoever',
"whoever's", 'whom', 'whomever', "whomever's", 'whose', 'whosever',
"whosever's", 'when', 'whenever', 'which', 'whichever', 'where', "where's",
"where'd", 'wherever', 'why', "why's", "why'd", 'whyever', 'what', "what's",
'whatever', 'whence', 'how', "how's", "how'd", 'however', 'whether',
'whatsoever', 'and', 'or', 'not', 'because', 'also', 'always', 'never', 'only',
'really', 'very', 'greatly', 'extremely', 'somewhat', 'no', 'nope', 'nah',
'yes', 'yep', 'yeh', 'yeah', 'maybe', 'perhaps', 'more', 'most', 'less',
'least', 'good', 'great', 'well', 'better', 'best', 'bad', 'worse', 'worst',
'too', 'thru', 'though', 'although', 'yet', 'already', 'then', 'even', 'now',
'sometimes', 'still', 'together', 'altogether', 'entirely', 'fully', 'entire',
'whole', 'completely', 'utterly', 'seemingly', 'apparently', 'clearly',
'obviously', 'actually', 'actual', 'usually', 'usual', 'literally', 'honestly',
'absolutely', 'definitely', 'generally', 'totally', 'finally', 'basically',
'essentially', 'fundamentally', 'automatically', 'immediately', 'necessarily',
'primarily', 'normally', 'perfectly', 'constantly', 'particularly',
'eventually', 'hopefully', 'mainly', 'typically', 'specifically', 'differently',
'appropriately', 'plenty', 'certainly', 'unfortunately', 'ultimately',
'unlikely', 'likely', 'potentially', 'fortunately', 'personally', 'directly',
'indirectly', 'nearly', 'closely', 'slightly', 'probably', 'possibly',
'especially', 'frequently', 'often', 'oftentimes', 'seldom', 'rarely', 'sure',
'while', 'whilst', 'able', 'unable', 'else', 'ever', 'once', 'twice', 'thrice',
'almost', 'again', 'instead', 'next', 'previous', 'unless', 'somehow', 'anyhow',
'anywhere', 'somewhere', 'everywhere', 'nowhere', 'further', 'anymore', 'later',
'ago', 'ahead', 'just', 'same', 'different', 'big', 'small', 'little', 'tiny',
'large', 'huge', 'pretty', 'mostly', 'anyway', 'anyways', 'otherwise',
'regardless', 'throughout', 'additionally', 'moreover', 'furthermore',
'meanwhile', 'afterwards', 'thing', "thing's", 'things', 'stuff', "other's",
'others', "another's", 'total', '', 'false', 'none', 'way', 'kind', 'zero',
'zeros', 'zeroes', 'one', 'ones', 'two', 'three', 'four', 'five', 'six',
'seven', 'eight', 'nine', 'ten', 'twenty', 'thirty', 'forty', 'fifty', 'sixty',

```
'seventy', 'eighty', 'ninety', 'hundred', 'hundreds', 'thousand', 'thousands',
'million', 'millions', 'first', 'last', 'second', 'third', 'fourth', 'fifth',
'sixth', 'seventh', 'eigth', 'ninth', 'tenth', 'firstly', 'secondly', 'thirdly',
'lastly', 'hello', 'hi', 'hey', 'sup', 'yo', 'greetings', 'please', 'okay',
'ok', "y'all", 'lol', 'rofl', 'thank', 'thanks', 'alright', 'kinda', 'dont',
'sorry', 'idk', 'tldr', 'tl', 'dr', 'tbh', 'dude', 'tho', 'aka', 'plz', 'pls',
'bit', 'don', 'www', 'https', 'http', 'com', 'etchtml', 'reddit', 'subreddit',
'subreddits', 'comments', 'reply', 'replies', 'thread', 'threads', 'post',
'posts', 'website', 'websites', 'web site', 'web sites']}
Run time: 2.1990275382995605 seconds
```

[ ]:
```
y_pred_new = grid.predict(test_x)
create_test_csv(y_pred_new,"multi_pipeline.csv")
```

```
File saved.
```

[ ]:
```
#test selector => 94.011.
t_start = time.time()

pipe_params = {
    "vect__binary": [False],
    "vect__stop_words": [list(stop_words_custom)],
    'vect__preprocessor': [None],
    'vect__ngram_range':[(1,1)],
    "selecter__k":[5000,3000],
    "clf__alpha" : [0.5,0.1],
   #  "normalizer__norm": ['l2','l1']
}

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




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


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

grid.fit(train_x, train_y)

t_end = time.time()
```

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

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


y_pred = grid.predict(test_x)
create_test_csv(y_pred,"MultinomialNB_S_03032023_01.csv")
```

```
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 94.011.
The winning parameters are {'clf__alpha': 0.5, 'selecter__k': 5000,
'vect__binary': False, 'vect__ngram_range': (1, 1), 'vect__preprocessor': None,
'vect__stop_words': ['i', "i'll", "i'd", "i'm", "i've", 'ive', 'me', 'myself',
'you', "you'll", "you'd", "you're", "you've", 'yourself', 'he', "he'll", "he'd",
"he's", 'him', 'she', "she'll", "she'd", "she's", 'her', 'it', "it'll", "it'd",
"it's", 'itself', 'oneself', 'we', "we'll", "we'd", "we're", "we've", 'us',
'ourselves', 'they', "they'll", "they'd", "they're", "they've", 'them',
'themselves', 'everyone', "everyone's", 'everybody', "everybody's", 'someone',
"someone's", 'somebody', "somebody's", 'nobody', "nobody's", 'anyone',
"anyone's", 'everything', "everything's", 'something', "something's", 'nothing',
"nothing's", 'anything', "anything's", 'a', 'an', 'the', 'this', 'that',
"that's", 'these', 'those', 'my', 'your', 'yours', 'his', 'hers', 'its', 'our',
'ours', 'own', 'their', 'theirs', 'few', 'much', 'many', 'lot', 'lots', 'some',
'any', 'enough', 'all', 'both', 'half', 'either', 'neither', 'each', 'every',
'certain', 'other', 'another', 'such', 'several', 'multiple', 'rather', 'quite',
'aboard', 'about', 'above', 'across', 'after', 'against', 'along', 'amid',
'amidst', 'among', 'amongst', 'anti', 'around', 'as', 'at', 'away', 'before',
'behind', 'below', 'beneath', 'beside', 'besides', 'between', 'beyond', 'but',
'by', 'concerning', 'considering', 'despite', 'down', 'during', 'except',
'excepting', 'excluding', 'far', 'following', 'for', 'from', 'here', "here's",
'in', 'inside', 'into', 'left', 'like', 'minus', 'near', 'of', 'off', 'on',
'onto', 'opposite', 'out', 'outside', 'over', 'past', 'per', 'plus',
'regarding', 'right', 'since', 'than', 'there', "there's", 'through', 'to',
'toward', 'towards', 'under', 'underneath', 'unlike', 'until', 'up', 'upon',
'versus', 'via', 'with', 'within', 'without', 'may', 'might', 'will', "won't",
'would', "wouldn't", 'can', "can't", 'cannot', 'could', "couldn't", 'should',
```

"shouldn't", 'must', "must've", 'be', 'being', 'been', 'am', 'are', "aren't",
"ain't", 'is', "isn't", 'was', "wasn't", 'were', "weren't", 'do', 'doing',
"don't", 'does', "doesn't", 'did', "didn't", 'done', 'have', "haven't",
'having', 'has', "hasn't", 'had', "hadn't", 'get', 'getting', 'gets', 'got',
'gotten', 'go', 'going', 'gonna', 'goes', 'went', 'gone', 'make', 'making',
'makes', 'made', 'take', 'taking', 'takes', 'took', 'taken', 'need', 'needing',
'needs', 'needed', 'use', 'using', 'uses', 'used', 'want', 'wanna', 'wanting',
'wants', 'let', 'lets', 'letting', "let's", 'suppose', 'supposing', 'supposes',
'supposed', 'seem', 'seeming', 'seems', 'seemed', 'say', 'saying', 'says',
'said', 'know', 'knowing', 'knows', 'knew', 'known', 'look', 'looking',
'looked', 'think', 'thinking', 'thinks', 'thought', 'feel', 'feels', 'felt',
'based', 'put', 'puts', 'who', "who's", "who've", "who'd", 'whoever',
"whoever's", 'whom', 'whomever', "whomever's", 'whose', 'whosever',
"whosever's", 'when', 'whenever', 'which', 'whichever', 'where', "where's",
"where'd", 'wherever', 'why', "why's", "why'd", 'whyever', 'what', "what's",
'whatever', 'whence', 'how', "how's", "how'd", 'however', 'whether',
'whatsoever', 'and', 'or', 'not', 'because', 'also', 'always', 'never', 'only',
'really', 'very', 'greatly', 'extremely', 'somewhat', 'no', 'nope', 'nah',
'yes', 'yep', 'yeh', 'yeah', 'maybe', 'perhaps', 'more', 'most', 'less',
'least', 'good', 'great', 'well', 'better', 'best', 'bad', 'worse', 'worst',
'too', 'thru', 'though', 'although', 'yet', 'already', 'then', 'even', 'now',
'sometimes', 'still', 'together', 'altogether', 'entirely', 'fully', 'entire',
'whole', 'completely', 'utterly', 'seemingly', 'apparently', 'clearly',
'obviously', 'actually', 'actual', 'usually', 'usual', 'literally', 'honestly',
'absolutely', 'definitely', 'generally', 'totally', 'finally', 'basically',
'essentially', 'fundamentally', 'automatically', 'immediately', 'necessarily',
'primarily', 'normally', 'perfectly', 'constantly', 'particularly',
'eventually', 'hopefully', 'mainly', 'typically', 'specifically', 'differently',
'appropriately', 'plenty', 'certainly', 'unfortunately', 'ultimately',
'unlikely', 'likely', 'potentially', 'fortunately', 'personally', 'directly',
'indirectly', 'nearly', 'closely', 'slightly', 'probably', 'possibly',
'especially', 'frequently', 'often', 'oftentimes', 'seldom', 'rarely', 'sure',
'while', 'whilst', 'able', 'unable', 'else', 'ever', 'once', 'twice', 'thrice',
'almost', 'again', 'instead', 'next', 'previous', 'unless', 'somehow', 'anyhow',
'anywhere', 'somewhere', 'everywhere', 'nowhere', 'further', 'anymore', 'later',
'ago', 'ahead', 'just', 'same', 'different', 'big', 'small', 'little', 'tiny',
'large', 'huge', 'pretty', 'mostly', 'anyway', 'anyways', 'otherwise',
'regardless', 'throughout', 'additionally', 'moreover', 'furthermore',
'meanwhile', 'afterwards', 'thing', "thing's", 'things', 'stuff', "other's",
'others', "another's", 'total', '', 'false', 'none', 'way', 'kind', 'zero',
'zeros', 'zeroes', 'one', 'ones', 'two', 'three', 'four', 'five', 'six',
'seven', 'eight', 'nine', 'ten', 'twenty', 'thirty', 'forty', 'fifty', 'sixty',
'seventy', 'eighty', 'ninety', 'hundred', 'hundreds', 'thousand', 'thousands',
'million', 'millions', 'first', 'last', 'second', 'third', 'fourth', 'fifth',
'sixth', 'seventh', 'eigth', 'ninth', 'tenth', 'firstly', 'secondly', 'thirdly',
'lastly', 'hello', 'hi', 'hey', 'sup', 'yo', 'greetings', 'please', 'okay',
'ok', "y'all", 'lol', 'rofl', 'thank', 'thanks', 'alright', 'kinda', 'dont',
'sorry', 'idk', 'tldr', 'tl', 'dr', 'tbh', 'dude', 'tho', 'aka', 'plz', 'pls',

```
'bit', 'don', 'www', 'https', 'http', 'com', 'etchtml', 'reddit', 'subreddit',
'subreddits', 'comments', 'reply', 'replies', 'thread', 'threads', 'post',
'posts', 'website', 'websites', 'web site', 'web sites']}
Run time: 4.01872992515564 seconds
File saved.
```

[ ]:
```
#test the final after preprocessing
t_start = time.time()

pipe_params = {
    "vect__binary": [False],
    "vect__stop_words": [list(stop_words_custom)],
    'vect__preprocessor': [],
    'vect__ngram_range':[(1,1)],
    "selecter__k":[5000,3000],
    "clf__alpha" : [0.5,0.1],
   #  "normalizer__norm": ['l2','l1']
}

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




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


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

grid.fit(train_x, train_y)

t_end = time.time()

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

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


y_pred = grid.predict(test_x)
```

```
create_test_csv(y_pred,"MultinomialNB_S_03032023_01.csv")
```

[ ]:
```
#now that the model is finalized , build the final model

#####################################################################do not use this

from sklearn.model_selection import cross_val_score


final_vectorize = CountVectorizer(stop_words = stop_words_custom,␣
 ↪ngram_range=(1,1), binary=False)
vec_x_train = np.asarray(final_vectorize.fit_transform(train_x).todense())
vec_x_test = np.asarray(final_vectorize.transform(test_x).todense())

#skLearnFeatureSelector = SelectKBest(chi2, k=5000)

#selected_x_train = skLearnFeatureSelector.fit_transform(vec_x_train, train_y)
#selected_x_test = skLearnFeatureSelector.transform(vec_x_test)



model = MultinomialNB(alpha=0.5)
model.fit(vec_x_train, train_y)

# Step 4: Evaluate the model using cross-validation
cv_scores = cross_val_score(model, vec_x_train, train_y, cv=5)
mean_cv_accuracy = np.mean(cv_scores)

print(f"The 5-fold cross-validation accuracy is: {mean_cv_accuracy:.5f}")


y_pred = model.predict(vec_x_test)
create_test_csv(y_pred,"final_MultinomialNB.csv")
```

```
/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 5-fold cross-validation accuracy is: 0.92755
File saved.
```

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