final_random_forest

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')

from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
```

```
[nltk_data] Downloading package omw-1.4 to /root/nltk_data…
[nltk_data] Downloading package punkt to /root/nltk_data…
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data…
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data…
[nltk_data] Downloading package punkt to /root/nltk_data…
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data…
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package wordnet to /root/nltk_data…
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data…
[nltk_data]   Unzipping corpora/stopwords.zip.
```

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

train_data_initial = pd.read_csv('/content/gdrive/MyDrive/ecse551-mp2/train.
 ↪csv')
test_data = pd.read_csv('/content/gdrive/MyDrive/ecse551-mp2/test.csv')

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

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

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

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

```
  csv.to_csv(outfile_name,index=True, header=True)
  print ("File saved.")
```

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

#train_data = train_data.sample(500).reset_index(drop=True)
#train_data = train_data.head(200)


train_x = train_data["body"]
train_y = train_data["subreddit"]
test_x = test_data["body"]
```

[ ]:
```
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.*
```

[ ]:
```
#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)

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",
    "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",
    # 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: 590
```

[ ]:

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

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

class LemmaTokenizer:
     def __init__(self):
       self.wnl = WordNetLemmatizer()
     def __call__(self, doc):
```

```
        return [self.wnl.lemmatize(t,pos ="v") for t in word_tokenize(doc) if t.
 ↪isalpha()]

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


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

[ ]:
```
# best model
t_start = time.time()

pipe_params = {
    "vect__binary": [False,True],
    "vect__stop_words": [list(stop_words_library)],
    "selecter__k":[5000,3000],
    "normalizer__norm": ['l2','l1','max'],
    'classify__n_estimators':[100]
}

model = RandomForestClassifier()

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

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

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

grid.fit(train_x, train_y)

t_end = time.time()

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

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

```
Fitting 5 folds for each of 12 candidates, totalling 60 fits
The best accuracy is 89.274.
The winning parameters are {'classify__n_estimators': 100, 'normalizer__norm':
'max', 'selecter__k': 3000, 'vect__binary': True, 'vect__stop_words': ['top',
'them', 'however', 'together', 'sixty', 'such', 'elsewhere', 'done', 'for',
'please', "needn't", 'are', 'and', 'the', 'did', 'during', 'as', 'alone',
'though', 'yourselves', 'same', 'bottom', 'should', 'upon', 'themselves', 'at',
'himself', 'll', "haven't", 'ma', 'hasnt', 've', 'always', 'off', 'beforehand',
'i', 'any', 'whose', 'around', 'your', 'along', 'yours', "you're", 'hereby',
'don', 'he', 'mightn', 'yourself', 'ain', 'often', 'some', 'we', 'mustn',
'thereupon', 'per', 'even', 'since', 'will', 'couldnt', 'that', 'someone',
"you'll", 'anything', 'then', 'etc', 'few', 'other', 'those', 'never', 'once',
'ourselves', 'hadn', 'system', 'take', 'now', 'under', 'who', 'seemed', 'had',
'well', 'whatever', 'weren', 'hers', 'nothing', 'next', "don't", 'aren',
'needn', 'its', 'somewhere', 'up', "wasn't", 'side', 'above', 'third', 'full',
'until', 'not', 'fifteen', 'fill', 'hence', 'or', 'rather', 'five', 'put',
"wouldn't", 'seeming', 'only', 'after', 'theirs', 'one', "shan't", 'o', 'just',
'thru', 's', 'being', 'latter', 'amount', 'whereafter', 'front', 'itself',
'fifty', 'within', "she's", 'these', 'me', 'were', "doesn't", 'against', 'mill',
'whoever', 'thereby', 'wherein', "isn't", 'con', 'twenty', 'anyone', 'least',
'via', 'an', 'nine', "it's", 'seem', 'nevertheless', 'she', 'our', 'am', 'ever',
'thin', 'whence', 'how', 'hasn', 'whereupon', 'amoungst', "hadn't", 'was', 'm',
'him', 'while', 'many', 'too', 'into', 'herself', 'three', 'neither', 'can',
'hereupon', 'but', 'get', "should've", 'inc', 'nobody', 'two', 'on', 'further',
'whom', 'whither', 'their', 'afterwards', 'toward', 'ltd', 'everywhere', 'out',
'due', 'whenever', 'might', 'less', 'mine', 'several', 'of', 'four', 'name',
"weren't", 'they', 'with', 'having', 'hereafter', 'forty', 'eleven', 'shan',
'over', 'have', 'herein', 'go', 'formerly', 'my', 'also', 'another', 'where',
'else', "you've", 'anywhere', 'describe', 'yet', 't', 'none', 'there', 'thence',
'without', 'ie', 'whether', 'besides', 'except', 'y', 'every', 'shouldn', 'why',
'indeed', 'otherwise', "mustn't", 'meanwhile', 'before', 'somehow', 'which',
'so', 'more', 'when', 'cant', 'twelve', 'didn', 'if', 'very', 'see', 'beside',
'mostly', "won't", 'doing', 'from', 'again', 'first', 'nor', 'nowhere',
"aren't", 'seems', 'much', "hasn't", 'becoming', 'find', 'ours', 'a', 'eight',
'thus', 'everything', 'this', 'un', 'isn', 'either', 'give', 'may', 'below',
'move', 'about', 'both', "didn't", 'anyway', 'own', 'cry', 'couldn', 'no', 'eg',
'empty', 'must', 'haven', 'be', 'sometimes', 'ten', 'show', 'here', 'interest',
'what', 'co', 'doesn', 'd', 'between', 'de', 'her', 'made', 'namely', 'won',
'almost', 'hundred', 'across', 'fire', 'latterly', 'cannot', 'whole', 'do',
'among', 'his', 're', 'all', 'became', 'been', 'onto', 'than', 'would',
'moreover', 'becomes', 'although', 'still', "mightn't", 'most', 'to', 'beyond',
'former', 'thick', 'each', 'does', "couldn't", 'sometime', 'found', 'could',
'it', 'us', 'in', 'sincere', 'behind', 'everyone', 'last', "you'd", 'bill',
```

```
'whereas', 'detail', 'through', 'thereafter', 'noone', 'wasn', 'therein', 'six',
'towards', 'anyhow', 'has', 'call', 'myself', 'become', 'therefore', 'whereby',
'keep', 'down', 'by', 'something', 'wouldn', 'already', 'amongst', 'is', 'back',
'you', "shouldn't", 'part', 'enough', 'wherever', 'perhaps', "that'll",
'because', 'others', 'serious', 'throughout']}
Run time: 28.590136528015137 seconds
```

[ ]:
```
y_pred = grid.predict(test_x)
create_test_csv(y_pred,"random_forest_06032023_01.csv")
```

```
File saved.
```