

```

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import re
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from collections import Counter
import numpy as np
from sklearn.model_selection import train_test_split

import nltk
nltk.download('stopwords')
print(stopwords.words('english'))

['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you',
"you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself',
'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her',
'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them',
'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom',
'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was',
'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do',
'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or',
'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with',
'about', 'against', 'between', 'into', 'through', 'during', 'before',
'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out',
'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once',
'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both',
'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor',
'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't',
'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now',
'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't",
'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn',
"hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma',
'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan',
"shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren',
"weren't", 'won', "won't", 'wouldn', "wouldn't"]

[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!

#Loading data from the csv file using pandas
X_data=pd.read_csv('/content/X data.csv',encoding='ISO-8859-1')

#Checking number of rows and columns
X_data.shape

(162980, 2)

```

```

#Printing first 5 rows of data
X_data.head()

{"type": "dataframe", "variable_name": "X_data"}

#Naming columns and reading data sets again
COL_NAMES=['TEXT', 'FLAG']
X_data=pd.read_csv('/content/X_data.csv',encoding='ISO-8859-1',names=COL_NAMES)
X_data.head()

{"type": "dataframe", "variable_name": "X_data"}

#counting number of missing values in X_data
X_data.isnull().sum()

TEXT      4
FLAG      7
dtype: int64

#Filling null values with -1
X_data=X_data.fillna(-1)
X_data['TEXT'].replace(to_replace="-1",value="Modi will win")

0                                clean_text
1      when modi promised â€œminimum government maxim...
2      talk all the nonsense and continue all the dra...
3      what did just say vote for modi  welcome bjp t...
4      asking his supporters prefix chowkidar their n...
...
162976  why these 456 crores paid neerav modi not reco...
162977  dear rss terrorist payal gawar what about modi...
162978  did you cover her interaction forum where she ...
162979  there big project came into india modi dream p...
162980  have you ever listen about like gurukul where ...
Name: TEXT, Length: 162981, dtype: object

#checking the distributions of target column
X_data['TEXT'].value_counts()

TEXT
-1
4
2019
2
clean_text
1
should vote modi for cpas after years
1
lok sabha election 2019 live modi has ignored his own constituency
varanasi says priyanka gandhi

```

```

1
..
modi destroying india for personal benefit
1
sreeniwho announced buddha laughing nuclear testjust for sake dont
frowl before knowing the achievementthis not achievement modiits
nations achievementmake your heart bit enlarged
1
modi thunders indiaâ( entry into indiaâ( space club raga calls â(happy
theatreÂ dayâ(
1
back basics jobs\nfarmers\nsmall businesses\ngst reform\neducation\
nhealth\nenvironment\nwater\ninfrastructure\ninvestment revival
national security indiaâ( armed forces are handling that donâ( worry
space narendra modi thanks 1
have you ever listen about like gurukul where discipline are
maintained even narendra modi rss only maintaining the culture indian
more attack politics but someone attack hinduism rss will take action
that proud for 1
Name: count, Length: 162977, dtype: int64

X_data['FLAG'].value_counts()

FLAG
1      72250
0      55213
-1     35510
-1         7
category    1
Name: count, dtype: int64

```

0 -->neutral tweet 1 -->positive tweet -1 --> negative tweet

Stemming is the process of reducing a word to its root form i.e Swimming to swim. We do this using the porter stemmer function

```

port_stem=PorterStemmer()

def stemming(content):
    # Convert content to string to handle non-string values
    content = str(content)
    stemmed_content=re.sub('[^a-zA-Z]', ' ',content)
    stemmed_content=stemmed_content.lower()
    stemmed_content=stemmed_content.split()
    # Correct variable name from stemmed_conted to stemmed_content
    stemmed_content=[port_stem.stem(word) for word in stemmed_content
    if not word in stopwords.words('english')]
    stemmed_content=' '.join(stemmed_content)

```

```

    return stemmed_content

X_data['STEM_TEXT']=X_data['TEXT'].apply(stemming) #about 7mins to
complete this execution!!! S00000 L0000NG!!!!

#Viewing new data
X_data.head()

{"type": "dataframe", "variable_name": "X_data"}

print(X_data['FLAG'])
0          category
1             -1
2              0
3              1
4              1
...
162976          -1
162977          -1
162978           0
162979           0
162980           1
Name: FLAG, Length: 162981, dtype: object

#seperating data and label
X=X_data['STEM_TEXT'].values
Y=X_data['FLAG'].values.astype(str)

print(X)

['clean text'
 'modi promis minimum govern maximum govern expect begin difficult job
 reform state take year get justic state busi exit psu templ'
 'talk nonsens continu drama vote modi' ... 'cover interact forum
 left'
 'big project came india modi dream project happen realiti'
 'ever listen like gurukul disciplin maintain even narendra modi rss
 maintain cultur indian attack polit someon attack hinduism rss take
 action proud']

print(Y)

['category' '-1' '0' ... '0' '0' '1']

# Calculate class distribution
class_distribution = Counter(Y)

# Find classes with only one sample
classes_to_remove = [cls for cls, count in class_distribution.items()
if count < 2]

```

```

# Remove samples belonging to the under-represented classes
mask = ~np.isin(Y, classes_to_remove)
X = X[mask]
Y = Y[mask]

#SPLITTING DATA TO TRAIN AND TEST
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,strat
ify=Y,random_state=2)

print(X.shape,Y_train.shape,X_test.shape)

(162980,) (130384,) (32596,)

print(X_train)

['kaha tha nachoron sarkarscamist parti hai modi ruin countri dont
forget vote'
'chines citizen power right speak'
'missil defenc india strengthen modi govern' ...
'pm narasimha rao communist lack vision like communist includ shastri
moraji desai today due vajpaye narendra modi other lack vision decis'
'smita prakash modi fangirl would modi without ani would ani without
modi'
'lok sabha elect campaign live make sure rahul defeat say prakash
karat financialxpress']

print(X_test)

['modi anti nation lie creat commun divid mislead manipul data'
'jaya pradha join bjp convass goten shorten amount also popular bjp
big rich leader bythat famou cine field enjoy modi also'
'power popular leadership modi one tweet whole countri goe
desper' ... ''
'modi address massiv ralli kurnool andhra pradesh via namo app'
'watch video voic whatsapp section volunt modul narendra modi app']

#Converting textual Data to numerical data
vectorizer=TfidfVectorizer(lowercase=False)
# Fit and transform the original text data 'X_text' (assuming this is
your original text data variable)
X = vectorizer.fit_transform(X_text)
# Now transform the training and testing sets using the fitted
vectorizer
X_train=vectorizer.transform(X_train)
X_test=vectorizer.transform(X_test)

print(X_train)

(0, 5877)      0.23211504791694398
(0, 7538)      0.23548945446948935
(0, 9418)      0.354204268832559

```

```

(0, 10840)    0.2753084460429206
(0, 13783)    0.47083934854041315
(0, 16942)    0.06500689091180703
(0, 19727)    0.25053561795816454
(0, 22855)    0.4318397312580794
(0, 26449)    0.4087484798215461
(0, 28611)    0.2171804773161407
(1, 4756)     0.5918469474568181
(1, 5044)     0.4551164112905416
(1, 20743)    0.3366327029443095
(1, 22610)    0.3759564931415472
(1, 25003)    0.4334989983070124
(2, 6582)     0.5300020214201105
(2, 10358)    0.35074894432798903
(2, 12299)    0.23366050593559784
(2, 16809)    0.43637908746264226
(2, 16942)    0.0916516983467352
(2, 25434)    0.5853623335377406
(3, 405)      0.2523482267005551
(3, 3567)     0.35357562788310454
(3, 8468)     0.2191756447841152
(3, 8638)     0.5081332140365633
:             :
(130381, 21875) 0.23022080815445803
(130381, 24083) 0.25638451452365046
(130381, 26956) 0.13746263513427678
(130381, 28173) 0.20083845408518594
(130381, 28539) 0.3585143380350224
(130382, 1140) 0.5902580758686817
(130382, 8875) 0.37522459976450834
(130382, 16942) 0.12281166280105922
(130382, 20820) 0.31199308145928956
(130382, 24710) 0.3647377075919171
(130382, 29193) 0.3930048745543684
(130382, 29344) 0.33300614024675346
(130383, 4076) 0.22549246814283908
(130383, 6574) 0.2608135213898228
(130383, 8051) 0.16215926088637278
(130383, 9165) 0.43666985999467206
(130383, 13931) 0.4285314472717363
(130383, 15275) 0.21640635333884672
(130383, 15360) 0.25961397199213104
(130383, 15839) 0.18627600953805795
(130383, 20820) 0.38204335439728315
(130383, 21659) 0.18595018555306403
(130383, 22961) 0.2584458163033459
(130383, 23449) 0.16610043871510108
(130383, 25794) 0.23804619772332206

```

```
print(X_test)
```

(0, 1230) 0.26958717265727083
(0, 5360) 0.3121892578044157
(0, 6004) 0.2877293106675747
(0, 6386) 0.3445650741673561
(0, 7407) 0.3619912275820138
(0, 15163) 0.2895043909183832
(0, 16023) 0.44650548079773944
(0, 16789) 0.41099085054222745
(0, 16942) 0.05972087114921265
(0, 18019) 0.20408265440005308
(1, 909) 0.24233964199864133
(1, 1018) 0.1940119208092101
(1, 3142) 0.14963154522697322
(1, 3273) 0.19966985781677263
(1, 4013) 0.3035493266924947
(1, 5018) 0.3035493266924947
(1, 5740) 0.3035493266924947
(1, 8257) 0.18786060786814415
(1, 8859) 0.2158496436901553
(1, 9107) 0.2003429948755047
(1, 10342) 0.3035493266924947
(1, 13257) 0.24283093573712491
(1, 13578) 0.16215342950663475
(1, 14945) 0.1276272236408723
(1, 16942) 0.03186232085438606
:
(32592, 23569) 0.39236097060969066
(32592, 23918) 0.39236097060969066
(32592, 24964) 0.16872467473369665
(32592, 25723) 0.22384292133048414
(32592, 26234) 0.21329268036979332
(32594, 405) 0.2572984313499474
(32594, 1106) 0.3839118348072669
(32594, 1396) 0.31007248344097077
(32594, 14649) 0.45649845172633596
(32594, 16203) 0.3686756673788957
(32594, 16942) 0.05947675840780109
(32594, 17905) 0.28432703479309335
(32594, 20803) 0.3451404640437686
(32594, 21798) 0.28523238717232124
(32594, 28371) 0.24716975759466478
(32595, 1396) 0.27895174324173294
(32595, 16942) 0.053507313051794896
(32595, 17232) 0.4767332578224748
(32595, 17969) 0.1762845855602194
(32595, 23700) 0.3469172423358947
(32595, 28397) 0.27296612439328555
(32595, 28589) 0.3263140040976818
(32595, 28601) 0.39772489655740606

```
(32595, 28827) 0.24696891876614663
(32595, 29002) 0.37476711518967865
```

```
#Training the Machine learning model
```

```
model=LogisticRegression()
model.fit(X_train,Y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

```
n_iter_i = _check_optimize_result(
```

```
#Model Evaluation accurace on traininf data
```

```
X_train_prediction = model.predict(X_train)
training_data_accuracy=accuracy_score(Y_train,X_train_prediction)
```

```
print('Accuracy score on the training Data:',training_data_accuracy)
```

```
Accuracy score on the training Data: 0.8684807951895939
```

Testing Data Accuracy = 86.84%

```
#Model Evaluation accurace on training data
```

```
X_test_prediction = model.predict(X_test)
test_data_accuracy=accuracy_score(Y_test,X_test_prediction)
```

```
print('Accuracy score on the training Data:',test_data_accuracy)
```

```
Accuracy score on the training Data: 0.8392747576389741
```

Model Accuracy= 83.92%

We are saving this model to be used for later to use model directly without having to train multiple times! using the pickle library!!!

```
import pickle
```

```
#Saving the pretrained model to a .sav file using Pickle moduld
```

```
filename='trained_model.sav'
pickle.dump(model,open(filename,'wb'))
```


