minor_base .

by Shivam Techiesgroup

```
cmap = ['vlag', 'gnuplot_r', 'Purples_r', 'Oranges', 'Greys', 'Spectral_r', 'tab20_r', 'RdY
```

```
!pip install snscrape  # WE'll use snscrape to get the historical Tweets
```

```
from google.colab import drive# loading The Dataset with Google Dreive Link
drive.mount('/content/gdrive')
```

```
import numpy as np# Basic Library --> NUMPY
import pandas as pd# Basic Library --> PANDAS --> For Dataloading
import re # Basic Library
import string# Basic Library --> For String manupulation
import nltk# Importing NLTk
import matplotlib.pyplot as ploting# Basic Visulising Library
ploting.rc('figure',figsize=(17,9))# Setting The Size of PLOT
import seaborn as sns# Setting The Style of PLOT
sns.set_style('darkgrid')
import plotly.express as ex
from plotly.subplots import make_subplots# To show Sublplots of the graphs
nltk.download('vader_lexicon')# Sentiment Analyser
from wordcloud import WordCloud,STOPWORDS, ImageColorGenerator# For showing the most freque
from nltk.corpus import stopwords# Liberary to remove stopword
import datetime# converting date column to date format --> for better visualisation
import warnings# Basic Library
import time
# Stemming
ps = nltk.PorterStemmer()
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
from sklearn.metrics import confusion_matrix# Basic liberary for result showing
from sklearn.metrics import classification_report# Basic liberary for model building
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_au


warnings.filterwarnings("ignore")
print("Library Setup Complete.")# To check If all library imported Succesfully or not
```

```
import sys
import time
import datetime
import pandas as pd
import numpy as np
from os import path
import snscrape.modules.twitter as sntwitter
import calendar
import seaborn as sns
import matplotlib.pyplot as ploting
import re
import warnings
warnings.filterwarnings("ignore")
```

This is the section for getting the tweets from twitter which will only be fetched once

```
# keyword = "UPelection OR Punjabelection OR YogiAdityanath OR BJPelections2022 OR INCPunja
```

```
# since = "2021-11-01"
# until = "2021-12-31"
```

format : YYYY-MM-DD

```
# query = []
# curr = since
# since1 = (str(pd.to_datetime(curr) + pd.DateOffset(days=1))).split(' ')[0]

# while(curr != until):
#   query.append(f'''({keyword}) lang:en until:{since1.split(' ')[0]} since:{curr.split(' '
#   curr = (str(pd.to_datetime(curr) + pd.DateOffset(days=1))).split(' ')[0]
#   since1 = (str(pd.to_datetime(since1) + pd.DateOffset(days=1))).split(' ')[0]
```

```
# tweets = []
# limits = 250
# i = 0
```

```
# while (i < len(query)):
#     twee = []
#     for tweet in sntwitter.TwitterSearchScraper(query[i]).get_items():
#         if len(twee) == limits:
#             break
#         else:
#             twee.append([tweet.date, tweet.username, tweet.content])
#     i = i + 1
#     tweets.append(twee)
```

```
# import itertools
# merged = list(itertools.chain(*tweets))
```

```
# df2 = pd.DataFrame(merged, columns=['Date', 'User', 'Tweet'])
```

```
# df2
```

```
# df2.to_csv('/content/gdrive/MyDrive/Political_Tweets/tweets.csv')
```

Reading the dataset

```
df2 = pd.read_csv('/content/gdrive/MyDrive/Political_Tweets/tweets.csv' , nrows=12000)
```

```
'''Importing NLTK for processing of Natural language --> Necessary for text processing'''
import nltk
nltk.download('wordnet')# Downloading the wordnet
nltk.download('stopwords')# Downloading the list of stopword
nltk.download('omw-1.4')# Downloading the owm-1.4
```

```
stopword = nltk.corpus.stopwords.words('english')# Loading the stopword in english text
punctuation_removal = string.punctuation # Loading the punctuation in english text
'''This is a custom Function to Clean the text'''
def cleaning(sentence):
    re.sub(r"http\S+", "", sentence)                                            # For re
    sentence.translate(str.maketrans('', '', punctuation_removal))              # For Re
    re.sub(r'\s+[a-zA-Z]\s+', '', sentence)                                     # For re
    sentence = re.sub("[^a-zA-Z0-9]+", " ",sentence)
    sentence = re.split('\W+', sentence)                                            # To
    sentence = [word for word in sentence if word not in stopword]                  # Re
    sentence = [lemmatizer.lemmatize(word) for word in sentence]# Lemmitizing the word
    sentence = ' '.join([str(word) for word in sentence])# Detokenisation the sentance for 

    return sentence
```

Show hidden output

```
from tqdm import tqdm
tqdm.pandas()
```

```
%%time
df2['Tweet'] = df2['Tweet'].progress_apply(lambda sentence: cleaning(str(sentence).lower())
```

```
100%|██████████| 12000/12000 [00:09<00:00, 1245.36it/s]CPU times: user 6.36 s, sys: 
Wall time: 9.68 s
```

code for Plotting Graphs

```
import sys , warnings
warnings.filterwarnings('ignore')
import random
def show_count_plot(feature, title, df, size=1, ordered=True):
    ### setting the figure size of the plot
    f, ax = ploting.subplots(1,1, figsize=(4*size,4))
    ### Getting the total lenth of df in float
    total = float(len(df))
    if ordered:
        g = sns.countplot(df[feature], order = df[feature].value_counts().index[:50], palet
    else:
        g = sns.countplot(df[feature], palette=random.choice(cmap))
        ### Setting the tittle of the plot
    g.set_title(title)
    ## If size given is greater than 2 then the labels automatically gets rotated
    if(size > 2):
      ### Here we are setting the X_ticks
        ploting.xticks(rotation=90, size=8)

    ploting.show()
```

Sentiment VADER

```
## To get sentiment polarity I have Used VADER for the analysis of the sentence.
```

```
from nltk.sentiment.vader import SentimentIntensityAnalyzer as SIA# Importing the VADER sen
sia=SIA() # VADER (Valence Aware Dictionary and sEntiment Reasoner)
scores=[]
for i in tqdm(range(len(df2['Tweet']))):

    # It is an emotion analysis tool that concentrates on the feelings that are conveyed in
    # VADER employs a mix of a sentiment dictionary and a lexical feature list.

    score = sia.polarity_scores(df2['Tweet'][i])
    score=score['compound']
    scores.append(score)

    #  VADER not only provides information on our positivity and negativity scores, but als

sentiment=[]
# Giving Values to the sentiment -> if score is less than -0.05 then negative and if more t
for s in scores:
  ### Positive sentiment
    if s>=0.05:
        sentiment.append('Positive')
  ### Negative sentiment
    elif s<=(-0.05):
        sentiment.append('Negative')
  ### Neutral sentiment
    else:
        sentiment.append('Neutral')


df2['sentiment']= pd.Series(np.array(sentiment))
```

```
    100%|██████████| 12000/12000 [00:07<00:00, 1585.43it/s]
```

```
df2['sentiment'].value_counts()
```

```
    Positive    6213
    Neutral     3630
    Negative    2157
    Name: sentiment, dtype: int64
```

```
show_count_plot('sentiment' , "Total Count of Sentiment related to tweets" , df2 , 3)
```

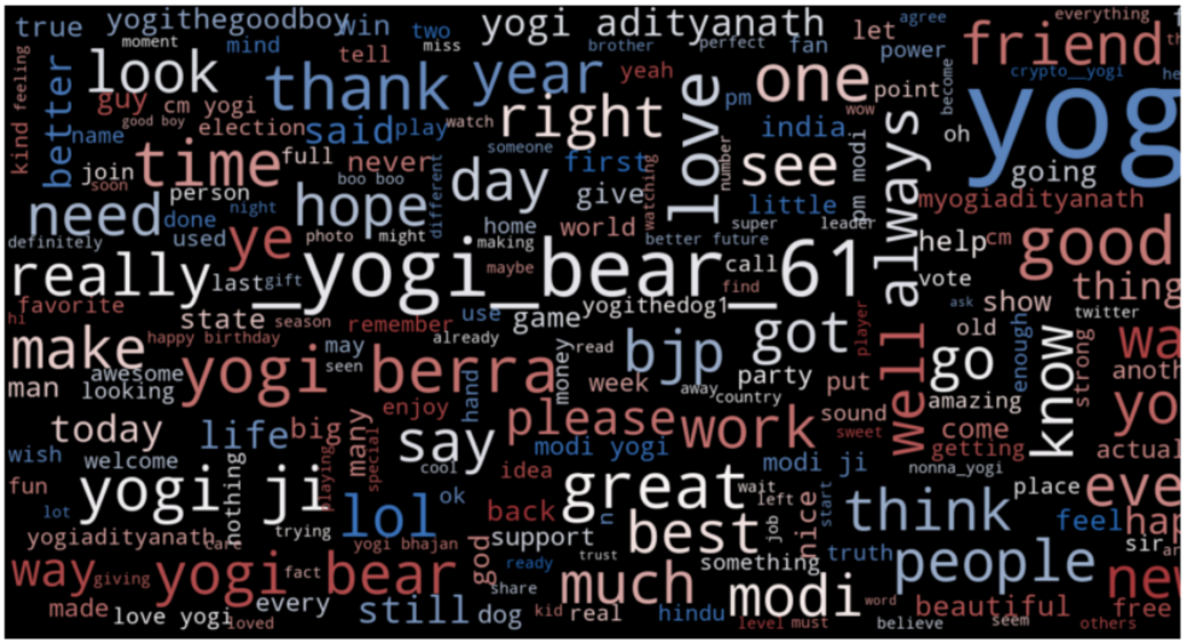
Total Count of Sentiment related to tweets

WordClouds

```
positive = df2[df2["sentiment"] == "Positive"]
negative = df2[df2["sentiment"] == "Negative"]
```

```
from wordcloud import WordCloud, STOPWORDS
import random
def Display_wordcloud(data, title=""):
    text = " ".join(t for t in data.dropna())
    stopwords = set(STOPWORDS)
    stopwords.update(["t", "co", "https", "amp", "U" , "th"])# Updation of stopword To remo
    wordcloud = WordCloud(stopwords=stopwords, scale=4, max_font_size=40, max_words=5000,co
    fig = ploting.figure(1, figsize=(16,16))
    ploting.axis('off')
    fig.suptitle(title, fontsize=20)
    fig.subplots_adjust(top=2.3)
    ploting.imshow(wordcloud, interpolation='bilinear')
    ploting.show()
```

```
# Creating wordclouds for positive, negative, neutral tweets
Display_wordcloud(positive.Tweet, 'Positive')
print(" ")
Display_wordcloud(negative.Tweet, 'Negative')
```



Positive

```
X = df2['Tweet']
Y = df2['sentiment']
```

```
print(f"The shape of X is {X.shape} , The shape of Y is {Y.shape}")
```

```
The shape of X is (12000,) , The shape of Y is (12000,)
```

```
from sklearn.model_selection import train_test_split # For spliting the data into train tes
X_trn, X_tst, y_trn, y_tst = train_test_split(X, Y, test_size=0.25, random_state=30)
```

```
from sklearn.feature_extraction.text import TfidfVectorizer as tfidf # Applying TFIDF vecto
tfidf_vect = tfidf(use_idf=True , lowercase=True , ngram_range=(1,2))
### we have used TFIDF for getting better result


X_trn = tfidf_vect.fit_transform(X_trn)
X_tst = tfidf_vect.transform(X_tst)
```

```
print(f"The shape of Training data is {X_trn.shape} , {y_trn.shape}")
print(f"The shape of Testing data is {X_tst.shape} , {y_tst.shape}")
```

```
The shape of Training data is (9000, 103850) , (9000,)
The shape of Testing data is (3000, 103850) , (3000,)
```

```
from sklearn.metrics import confusion_matrix# Basic liberary for result showing
from sklearn.metrics import classification_report# Basic liberary for model building
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_au
import random
```

```
def report_generator(y_tst , y_pred , title="Testing" , rnd=2):
    '''This a customised function to generate report and getting confusion matrix'''

    sentiment_classes = ['Negative','Neutral','Positive']
    print(" ")
    ploting.figure(figsize=(8,6))
    ### for the figure size we are setting the size
    sns.heatmap(confusion_matrix(y_tst, y_pred) ,cmap=random.choice(cmap), fmt = 'd' ,annot
    ploting.title(f'Confusion matrix {title}', fontsize=16)
    # Setting the tittle of the plot
    ploting.xlabel('Actual label', fontsize=12)
    ### for the plot we are giving the Xlabel
    ploting.ylabel('Predicted label', fontsize=12)
    ### for the plot we are giving the Ylabel
    print(classification_report(y_tst, y_pred))
    ### Generating the classification report
    print(" ")
    ### for the plot we are giving the Accuracy
    accuracy = accuracy_score(y_tst, y_pred)
    ### for the plot we are giving the accuracy
    print("  ")
    print("="*50)
    print(f'{title} Accuracy   : {round(accuracy , rnd)}')
    precision = precision_score(y_tst, y_pred,average='weighted')#estimating precision__sco
    recall = recall_score(y_tst, y_pred,average='weighted')#estimating recall___score
    f1score = f1_score(y_tst, y_pred,average='weighted')#esimaating f1_score
    print(f'{title} precision  : {round(precision , rnd)}')
    print(f'{title} recall     : {round(recall , rnd)}')
    print(f'{title} f1score    : {round(f1score , rnd)}')
    print("="*50)
    print("  ")
```

Linear SVC

```
from sklearn.svm import LinearSVC

linearSVC_Model = LinearSVC(random_state=10, tol=1 , max_iter=1 , C=1.0 , fit_intercept=Fal
linearSVC_Model.fit(X_trn, y_trn)
y_predict = linearSVC_Model.predict(X_tst)
```

```
    [LibLinear]
```

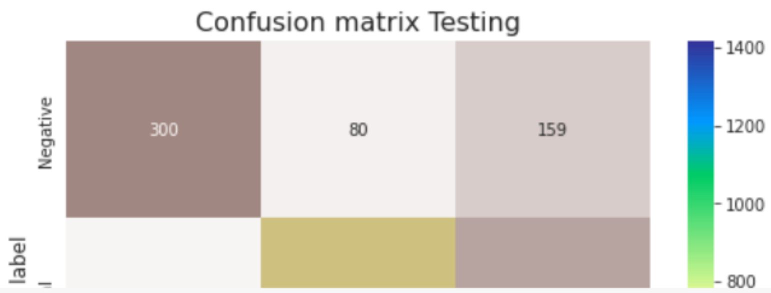
```
report_generator(y_tst , y_predict) ### Preciction and the Report
```

```
              precision    recall  f1-score   support

    Negative       0.72      0.56      0.63       539
     Neutral       0.78      0.66      0.72       911
    Positive       0.78      0.91      0.84      1550

    accuracy                           0.77      3000
   macro avg       0.76      0.71      0.73      3000
weighted avg       0.77      0.77      0.76      3000


==================================================
Testing Accuracy   : 0.77
Testing precision  : 0.77
Testing recall     : 0.77
Testing f1score    : 0.76
==================================================
```



```
# predict the results
y_pred_train=linearSVC_Model.predict(X_trn)
report_generator(y_trn , y_pred_train , "Training"  , 2) ### Preciction and the Report
```

```
              precision    recall  f1-score   support

    Negative       0.92      0.80      0.86      1618
     Neutral       0.92      0.88      0.90      2719
    Positive       0.91      0.97      0.94      4663

    accuracy                           0.91      9000
   macro avg       0.92      0.89      0.90      9000
weighted avg       0.91      0.91      0.91      9000



==================================================
Training f1score    : 0.91
```

Random forest

```python
# import Random Forest classifier

from sklearn.ensemble import RandomForestClassifier

# instantiate the classifier

rfc = RandomForestClassifier(random_state=42 , max_depth = 1000 , n_estimators= 200 , warm_s

# fit the model

rfc.fit(X_trn, y_trn)

# Predict the Test set results

y_pred = rfc.predict(X_tst)
accuracy_rf = report_generator(y_tst , y_pred , "Random Forest Testing") ### Preciction and
```

```
              precision    recall  f1-score   support

    Negative       0.92      0.28      0.43       539
     Neutral       0.61      0.96      0.75       911
    Positive       0.87      0.79      0.83      1550

    accuracy                           0.75      3000
   macro avg       0.80      0.68      0.67      3000
weighted avg       0.80      0.75      0.73      3000



==================================================
Random Forest Testing Accuracy   : 0.75
Random Forest Testing precision  : 0.8
```

```
y_prediction = rfc.predict(X_trn)
report_generator(y_trn , y_prediction , "Random Forest Training") ### Preciction and the Re
```

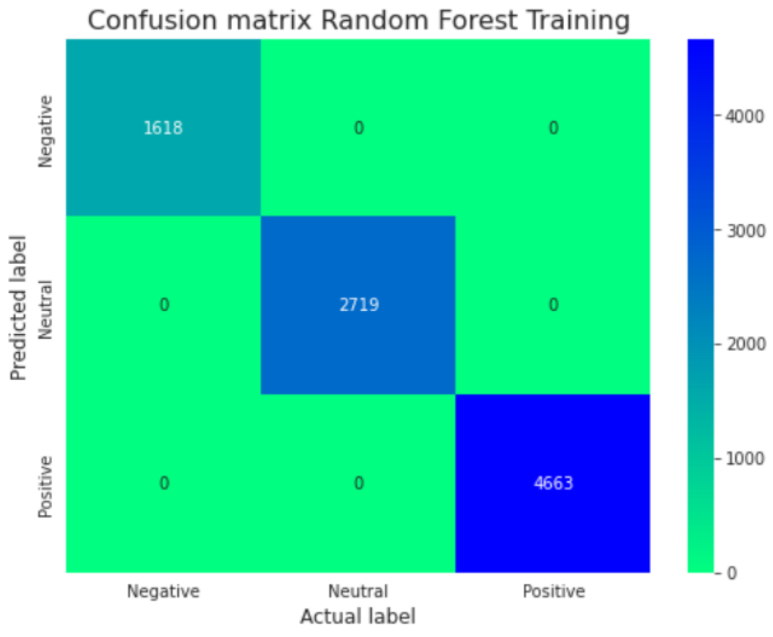
```
              precision    recall  f1-score   support

    Negative       1.00      1.00      1.00      1618
     Neutral       1.00      1.00      1.00      2719
    Positive       1.00      1.00      1.00      4663

    accuracy                           1.00      9000
   macro avg       1.00      1.00      1.00      9000
weighted avg       1.00      1.00      1.00      9000



==================================================
Random Forest Training Accuracy   : 1.0
Random Forest Training precision  : 1.0
Random Forest Training recall     : 1.0
Random Forest Training f1score    : 1.0
==================================================
```



SOFT VOTING CLASSIFIER (SVE)

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report

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

```
from sklearn.ensemble import VotingClassifier
voting_clf_1 = RandomForestClassifier()
voting_clf_2 = SVC(probability=True)
voting_clf_3 = LogisticRegression()

evoting_clf_1 = VotingClassifier(estimators=[('RF', voting_clf_1), ('SVM', voting_clf_2), (
evoting_clf_1.fit(X_trn, y_trn)
predictions = evoting_clf_1.predict(X_tst)
```

```
report_generator(y_tst , predictions)
```

```
              precision    recall  f1-score   support

    2
    Negative       0.75      0.48      0.59       539
     Neutral       0.71      0.83      0.77       911
    Positive       0.84      0.86      0.85      1550
```

```
predict = evoting_clf_1.predict(X_trn)
report_generator(y_trn , predict , "Training")
```

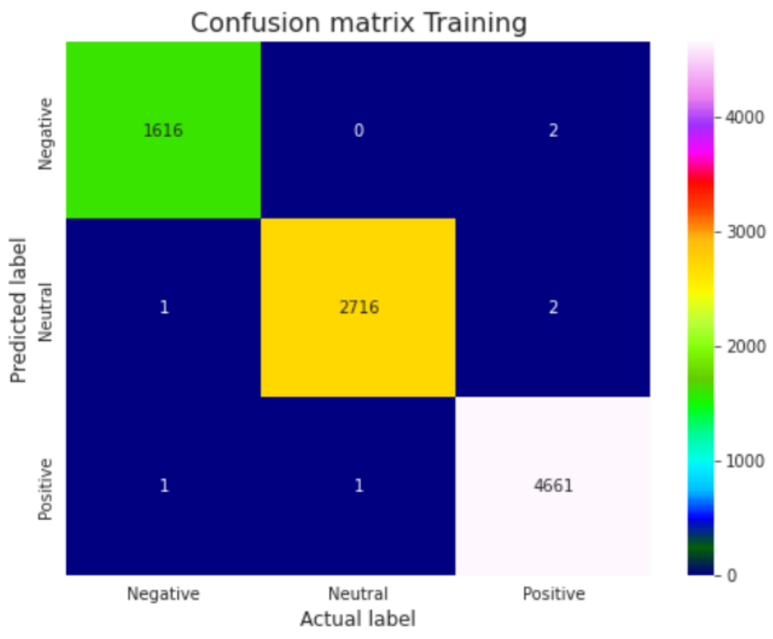
```
              precision    recall  f1-score   support

    Negative       1.00      1.00      1.00      1618
     Neutral       1.00      1.00      1.00      2719
    Positive       1.00      1.00      1.00      4663

    accuracy                           1.00      9000
   macro avg       1.00      1.00      1.00      9000
weighted avg       1.00      1.00      1.00      9000



==================================================
Training Accuracy   : 1.0
Training precision  : 1.0
Training recall     : 1.0
Training f1score    : 1.0
==================================================
```



Lets load the other dataset and Predict the Probability of winning of a Party

```
goa_df = pd.read_csv('/content/gdrive/MyDrive/Political_Tweets/State_tweets/Goa2022.csv')
manipur_df = pd.read_csv('/content/gdrive/MyDrive/Political_Tweets/State_tweets/Manipur2022
punjab_df = pd.read_csv('/content/gdrive/MyDrive/Political_Tweets/State_tweets/PunjabElecti
up_df = pd.read_csv('/content/gdrive/MyDrive/Political_Tweets/State_tweets/UttarPradeshElec
uk_df = pd.read_csv('/content/gdrive/MyDrive/Political_Tweets/State_tweets/Uttarakhand2022.

goa_df
```

	Unnamed: 0	Datetime	Tweet Id	Text	Us
0	0	2022-03-09 23:57:16+00:00	1501708732442746880	@priyankagandhi मेडम वो EVM वाले ट्वीट कितने ब...	n
1	1	2022-03-09 23:56:58+00:00	1501708655527591936	#ElectionResults मतमोजणीचं काऊंटडाऊन सुरू! मतद...	ab
2	2	2022-03-09 23:49:14+00:00	1501706707227254786	बहुत लोगों को कल रात नींद नहीं आई होगी!! फिर भ...	Aro
3	3	2022-03-09 23:39:12+00:00	1501704185691406336	Yesterday @BJP4Goa called meeting of all candi...	siddha
4	4	2022-03-09 23:38:52+00:00	1501704098907041792	कुछ घंटों में 5 राज्यों की सियासत और सियासतदान...	JeetBl
...	...	...	...	...	
21874	21874	2021-11-01 13:11:19+00:00	1455160523704741890	गोवा के दौरे पर पहुंचे दिल्ली के मुख्यमंत्री अ...	apr
21875	21875	2021-11-01 07:00:08+00:00	1455067115967291400	My writeup\n#india #IndianPolitics #Goa\n#GoaE...	

```
goa_df['Text'] = goa_df['Text'].progress_apply(lambda text: cleaning(str(text).lower()))
manipur_df['Text'] = manipur_df['Text'].progress_apply(lambda text: cleaning(str(text).lowe
punjab_df['Text'] = punjab_df['Text'].progress_apply(lambda text: cleaning(str(text).lower(
up_df['Text'] = up_df['Text'].progress_apply(lambda text: cleaning(str(text).lower()))
uk_df['Text'] = uk_df['Text'].progress_apply(lambda text: cleaning(str(text).lower()))
```

```
    100%|██████████| 21879/21879 [00:03<00:00, 5786.31it/s]
    100%|██████████| 5495/5495 [00:00<00:00, 5512.91it/s]
    100%|██████████| 49434/49434 [00:10<00:00, 4671.73it/s]
    100%|██████████| 65016/65016 [00:13<00:00, 4839.46it/s]
    100%|██████████| 18599/18599 [00:02<00:00, 7299.11it/s]
```

```
all_text = [goa_df['Text'] , manipur_df['Text'] , punjab_df['Text'] , up_df['Text'] , uk_df
```

```
X = []
for x in all_text:
    X.append(tfidf_vect.transform(x))
```

```
predictions = []
for i in X:
  predictions.append(linearSVC_Model.predict(i))
```

```
for i in predictions:
  i = pd.DataFrame(i , columns=["predictions"])
  print(i.value_counts())
  print("=="*20)
```

```
    predictions
```

```
Positive      10040
Neutral        9188
Negative       2651
dtype: int64
========================================
predictions
Positive      2681
Neutral       2199
Negative       615
dtype: int64
========================================
predictions
Neutral       24628
Positive      17159
Negative       7647
dtype: int64
========================================
predictions
Neutral       38367
Positive      17571
Negative       9078
dtype: int64
========================================
predictions
Neutral       12283
Positive       4214
Negative       2102
dtype: int64
========================================
```

```
import re
```

```
party = ["bjp" , "inc" , "aap"]
```

```
def subject(text):
  regex = r"bjp"
  match = re.search(regex, text)
  if match != None:
    return "bjp"
  elif match == None:
    reg = r"inc"
    match = re.search(reg, text)
    if match != None:
      return "inc"
    else:
      rex = r"aap"
      match = re.search(rex, text)
      if match != None:
        return "aap"
      else:
        return "others"
```

```
goa_df['sub'] = goa_df['Text'].progress_apply(lambda text: subject(text))
manipur_df['sub'] = manipur_df['Text'].progress_apply(lambda text: subject(text))
punjab_df['sub'] = punjab_df['Text'].progress_apply(lambda text: subject(text))
up_df['sub'] = up_df['Text'].progress_apply(lambda text: subject(text))
uk_df['sub'] = uk_df['Text'].progress_apply(lambda text: subject(text))
```

```
100%|██████████| 21879/21879 [00:00<00:00, 220473.87it/s]
100%|██████████| 5495/5495 [00:00<00:00, 146352.26it/s]
100%|██████████| 49434/49434 [00:00<00:00, 192442.52it/s]
100%|██████████| 65016/65016 [00:00<00:00, 200420.15it/s]
100%|██████████| 18599/18599 [00:00<00:00, 211392.22it/s]
```

```
punjab_df['sub'].value_counts()
```

```
others    34392
aap        6303
bjp        5800
inc        2939
Name: sub, dtype: int64
```

```
all_df = [goa_df , manipur_df , punjab_df , up_df , uk_df]
```

```
goa_df.columns
```

```
Index(['Unnamed: 0', 'Datetime', 'Tweet Id', 'Text', 'Username', 'sub'],
dtype='object')
```

```
dfs = []
for i,j in zip(all_df , predictions):
  j = pd.DataFrame(j)
  res = pd.concat([i,j] , axis=1)
  res = res[res['sub'] != "others"]
  res = res.reset_index(inplace=False)
  res.columns = [*res.columns[:-1], 'predictions']
  res = res[res['predictions'] != "Neutral"]
  dfs.append(res)
```

```
def party_sentiment(party , sentiment):
    party_sen = None
    if(party == "bjp" and sentiment == "Positive"):
        party_sen = "Bjp with Positive Polarity"
    elif(party == "inc" and sentiment == "Negative"):
        party_sen = "inc with Negative Polarity"
    elif(party == "bjp" and sentiment == "Negative"):
        party_sen = "Bjp with Negative Polarity"
    elif(party == "Congress" and sentiment == "Positive"):
        party_sen = "Congress with Positive Polarity"
    elif(party == "aap" and sentiment == "Negative"):
        party_sen = "AAP with Negative Polarity"
    elif(party == "aap" and sentiment == "Positive"):
        party_sen = "AAP with Positive Polarity"
    return party_sen
```

```
for i in dfs:
  i["party_sentiments"] = i.apply(lambda x : party_sentiment(x["sub"], x["predictions"]), ax
```

```
dfs[1]
```

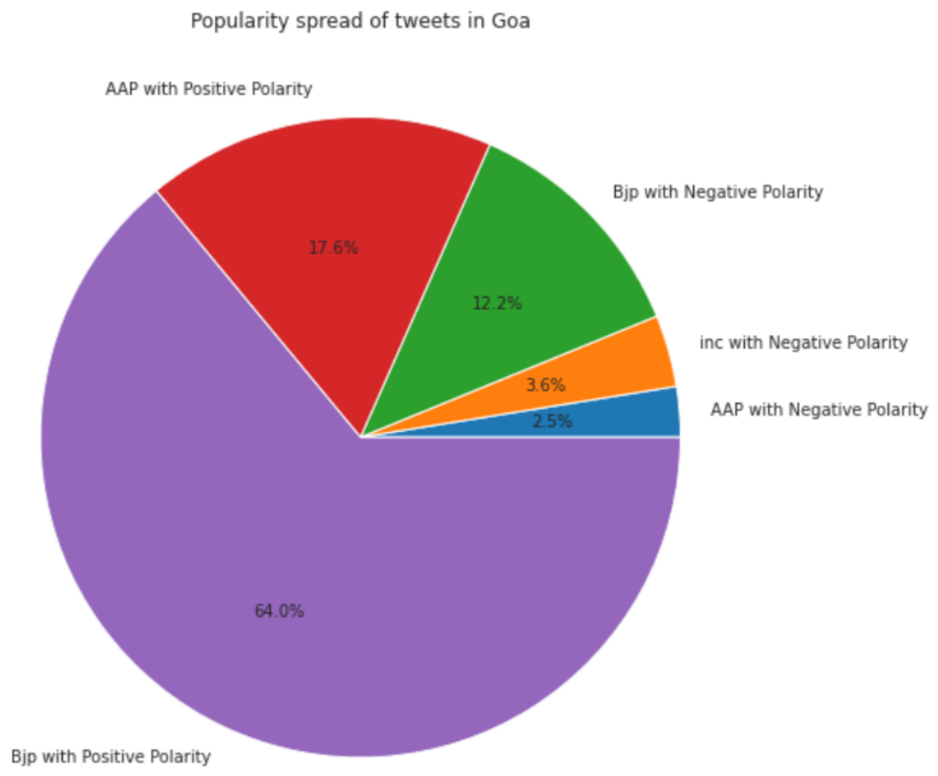
	index	Unnamed: 0	Datetime	Tweet Id	Text	Username
0	2	2	2022-03-09 23:35:19+00:00	1501703205666709507	today bjp losing election 5 state goaelections...	saifuddin_06
1	9	9	2022-03-09 18:29:02+00:00	1501626126178738176	evm bjp goaelections2022 punjabelections2022 ...	AgarwalJiSpeaks
2	15	15	2022-03-09 17:45:53+00:00	1501615270607011847	personal prediction manipurelections2022 bjp w...	iamANKUR_96
3	17	17	2022-03-09 17:41:14+00:00	1501614097330286592	watch medium political sensex 5 state election...	ymediagroup
			[illegible]		manipurelections2022	

```
ploting.figure(figsize=(5, 5))
def show_plt(df , state):
  x = df.party_sentiments.value_counts().sort_values()
  ax = ploting.pie(x = x, labels=x.index, autopct = '%1.1f%%')
  ploting.title(f"Popularity spread of tweets in {state}")
  ploting.show()
  print(" ")
  print("=="*50)
  print(" ")
```

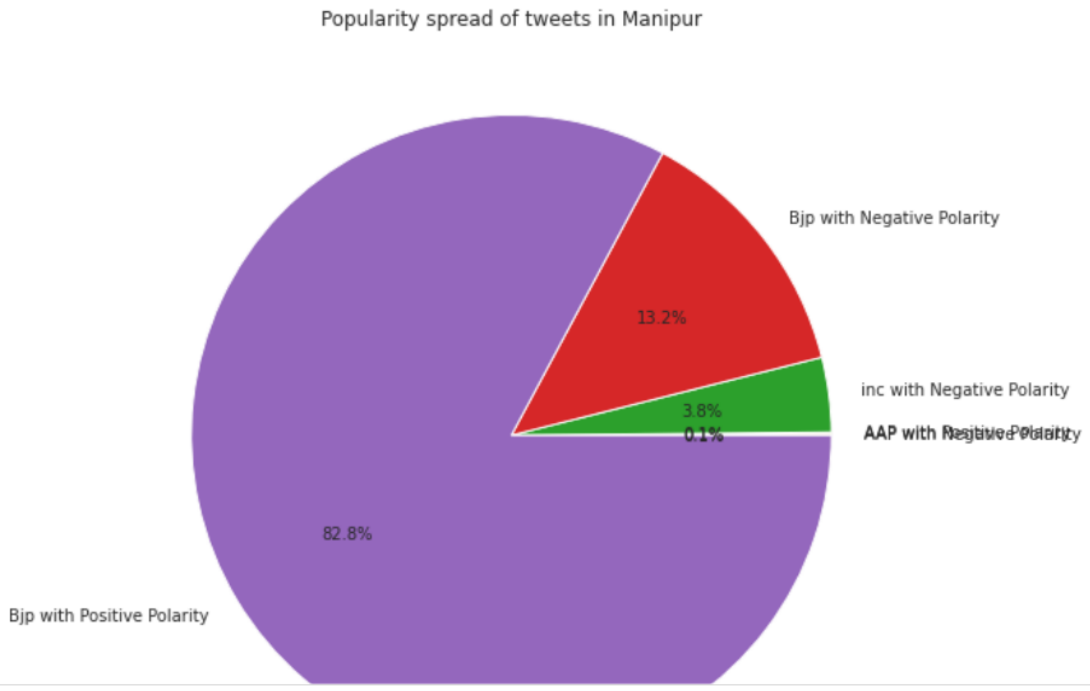
```
<Figure size 360x360 with 0 Axes>
```

```
state = ["Goa" , "Manipur" , "Punjab" , "Uttar Pradesh" , "Uttra khand"]
```

```
for i,j in zip(dfs,state):
  show_plt(i , j)
```


Popularity spread of tweets in Goa
AAP with Positive Polarity
17.6%
Bjp with Negative Polarity
12.2%
inc with Negative Polarity
3.6%
AAP with Negative Polarity
2.5%
64.0%
Bjp with Positive Polarity

```
================================================================================
```


Popularity spread of tweets in Manipur
Bjp with Negative Polarity
13.2%
inc with Negative Polarity
3.8%
0.1%
AAP with Negative Polarity
AAP with Positive Polarity
82.8%
Bjp with Positive Polarity

minor_base .

ORIGINALITY REPORT

22%	12%	10%	19%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to University of Hertfordshire Student Paper	6%
2	Submitted to University of New South Wales Student Paper	2%
3	machinelearningmastery.com Internet Source	1%
4	Akshay Kulkarni, Adarsha Shivananda. "Natural Language Processing Recipes", Springer Science and Business Media LLC, 2021 Publication	1%
5	Submitted to University of Portsmouth Student Paper	1%
6	Submitted to University of Teesside Student Paper	1%
7	niebla.unileon.es Internet Source	1%
8	Submitted to Eastern University Student Paper	1%

9	Submitted to Harrisburg University of Science and Technology Student Paper	1 %
10	mafiadoc.com Internet Source	1 %
11	deepnote.com Internet Source	1 %
12	dataanablog.com Internet Source	1 %
13	deepai.org Internet Source	1 %
14	Submitted to University of Southern California Student Paper	1 %
15	Submitted to University of Sydney Student Paper	<1 %
16	Manohar Swamynathan. "Mastering Machine Learning with Python in Six Steps", Springer Science and Business Media LLC, 2017 Publication	<1 %
17	Submitted to Georgia State University Student Paper	<1 %
18	Stackabuse.Com Internet Source	<1 %
19	norma.ncirl.ie Internet Source	<1 %

20	dfzljdn9uc3pi.cloudfront.net Internet Source	<1 %
21	www.codeprj.com Internet Source	<1 %
22	Puneet Mathur. "Machine Learning Applications Using Python", Springer Science and Business Media LLC, 2019 Publication	<1 %
23	researchrepository.wvu.edu Internet Source	<1 %

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		