Minor_propose.ipynb

by Shivam Techiesgroup

Submission date: 05-Apr-2023 02:20AM (UTC-0500)
Submission ID: 1883756893
File name: Minor_propose.ipynb_-_Colaboratory.pdf (1.73M)
Word count: 3883
Character count: 21968

```
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
!pip install contractions
```

```
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

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

```
# 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)
```

Dataframe befoe cleaning containing the emoji emoticons , stopword , punctuation , digits , wrong spell , dual spaces , single characters ,

---

Here we are translating the HINDI text into ENGLISH To get better Results

---

```
df2.Tweet.iloc[2]
```

```
'@BabySafeShiba Great super amauzing giveaway 🔥\nI hope both of you brings properity
and Success to each other . Also This is great oppurtunity to those loyal Supporters.
Thanks\n\n@Since1998       @Yogi_han_han @YoditaR_\n\n#BSSI #doge'
```

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

```
!pip3 install googletrans==3.1.0a0

# import the library
import googletrans
from googletrans import Translator
```

Show hidden output

```
# Here we are creating a translators object
translator = Translator()
```

```
df2 = df2.dropna()
df2 = df2.drop_duplicates()
```

```
def trans(sentance):
  sent = translator.translate(sentance).text
  return sent
```

```
%%time
df2['Tweet'] = df2['Tweet'].progress_apply(lambda text: trans(text))
```

```
100%|██████████| 12000/12000 [10:34<00:00, 18.90it/s]CPU times: user 1min 51s, sys:
Wall time: 10min 34s
```

```
'''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
import contractions
'''This is a custom Function to Clean the text'''
def cleaning(sentance):
    re.sub(r"http\S+", "", sentance)                                    # For remov
    sentance.translate(str.maketrans('', '', punctuation_removal))      # For Remov
    sentance = contractions.fix(sentance)                               # for spell
    re.sub(r'\s+[a-zA-Z]\s+', '', sentance)                             # For remov
    re.sub(r'\s+', ' ', sentance, flags=re.I)                           # double spa
    sentance = ''.join([i for i in sentance if not i.isdigit()])        # To remove
    sentance = re.split('\W+', sentance)                                # Tokenisat
    sentance = [word for word in sentance if word not in stopword]      # Remving S
    sentance = [lemmatizer.lemmatize(word) for word in sentance]        # Lemmitizi
```

```
    sentance = [lemmatizer.lemmatize(word) for word in sentance]                    # Lemmitizin
    sentance = ' '.join([str(word) for word in sentance])                           # Detokenis

    return sentance
```

Show hidden output

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

```
100%|██████████| 12000/12000 [00:04<00:00, 2467.96it/s]CPU times: user 4.67 s, sys:
Wall time: 4.88 s
```

Dataframe after Cleaning

```
df2
```

	Unnamed: 0	Date	User	Tweet
0	0	2021-11-01 23:59:49+00:00	MrLiepitz	nicotrijudie machete ummm google ivan rodrigu...
1	1	2021-11-01 23:58:24+00:00	bilal460	yogi adityanath warns afghan leader another ad...
2	2	2021-11-01 23:58:07+00:00	imammud72803050	babysafeshiba great super amauzing giveaway h...
3	3	2021-11-01 23:58:00+00:00	florentosmani1	thing stand well simple fact allowed thought e...
4	4	2021-11-01 23:57:36+00:00	YOGI_2K2	olga_stfu boutta non smoker geeked http co uk...
...	...	...	...	...
11995	11995	2021-12-18 19:53:06+00:00	mike49849353	raleigh_yogi justyntyme_ superssports missed
11996	11996	2021-12-18 19:53:02+00:00	livdose	ganga expressway connect district also heart c...

Here we are removing the rare and frequent words

```
from collections import Counter
cnt = Counter()
for text in df2['Tweet'].values:
    for word in text.split():
        cnt[word] += 1
```

```
print(cnt)
```

```
Counter({'yogi': 6843, 'co': 3457, 'http': 3386, '_yogi_bear_': 1234, 'like': 879, 'be
```

```
words = []
for text in df2['Tweet'].values:
  for word in text.split():
    if(cnt[word] == 1):
      words.append(word)
```

```
len(words) ### Number of words getting removed as they occour only once
```

```
18003
```

```
def remove_rare(text):
  text = re.split('\W+', text)
  text = [word for word in text if word not in words]
  text = ' '.join([word for word in text])
  return text

df2['Tweet'] = df2['Tweet'].progress_apply(lambda text: remove_rare(text))
```

```
100%|██████████| 12000/12000 [01:10<00:00, 169.35it/s]
```

This si the word and its word count That are present in all over the dataframe

---

```
count = Counter()
for text in df2['Tweet'].values:
    for word in text.split():
        count[word] += 1
print(count)
```

```
Counter({'yogi': 6843, 'co': 3457, 'http': 3386, '_yogi_bear_': 1234, 'like': 879, 'be
```

```
FREQWORDS = set([w for (w, wc) in count.most_common(3)])
def remove_freqwords(text):
    """custom function to remove the frequent words"""
    return " ".join([word for word in str(text).split() if word not in FREQWORDS])
df2['Tweet'] = df2['Tweet'].progress_apply(lambda text: remove_freqwords(text))
```

```
100%|██████████| 12000/12000 [00:00<00:00, 172069.30it/s]
```

```
FREQWORDS ### Removing the top 3 frequent words
```

```
{'co', 'http', 'yogi'}
```

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

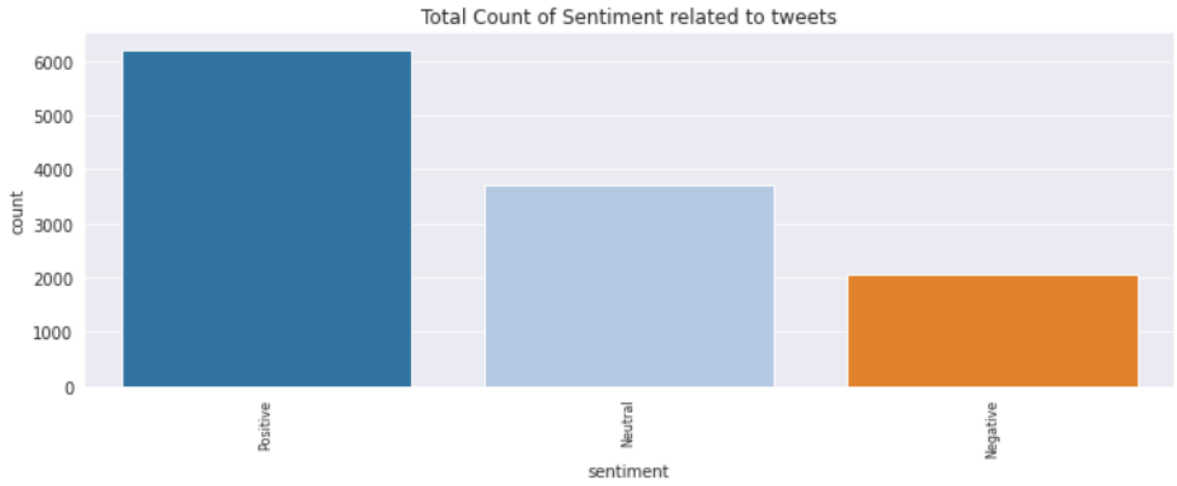
```
## get the polarity of an emotion VADER was used by me to analyse the text.

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

    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 [00:02<00:00, 4829.44it/s]
```

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

```
    Positive    6214
    Neutral     3711
    Negative    2075
    Name: sentiment, dtype: int64
```

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

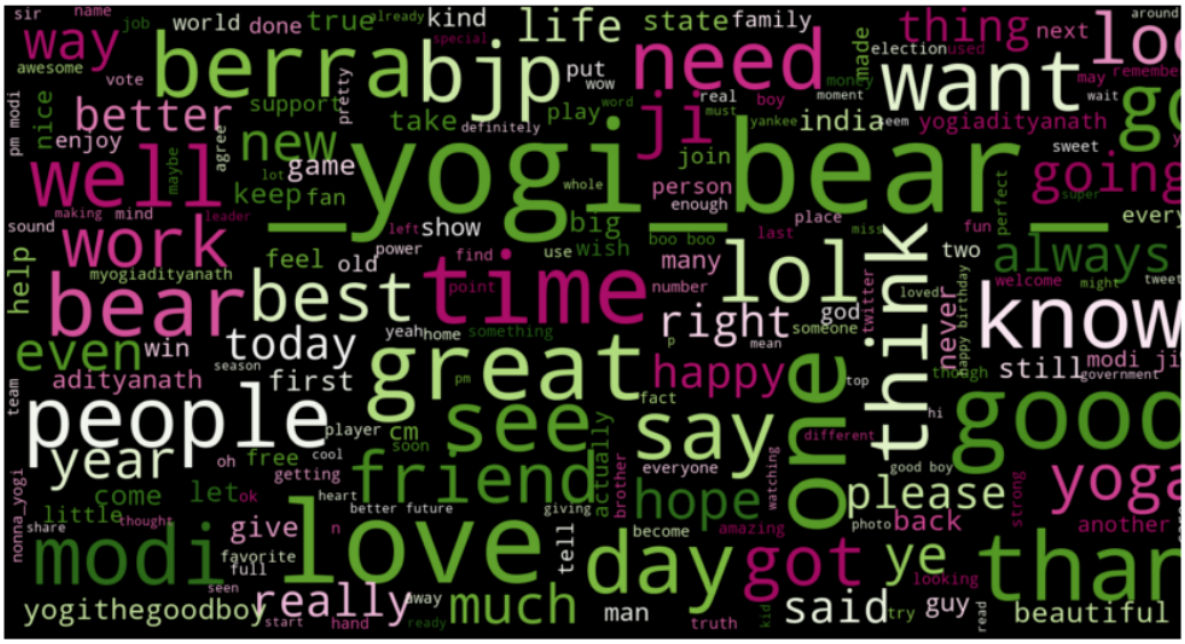


WordClouds

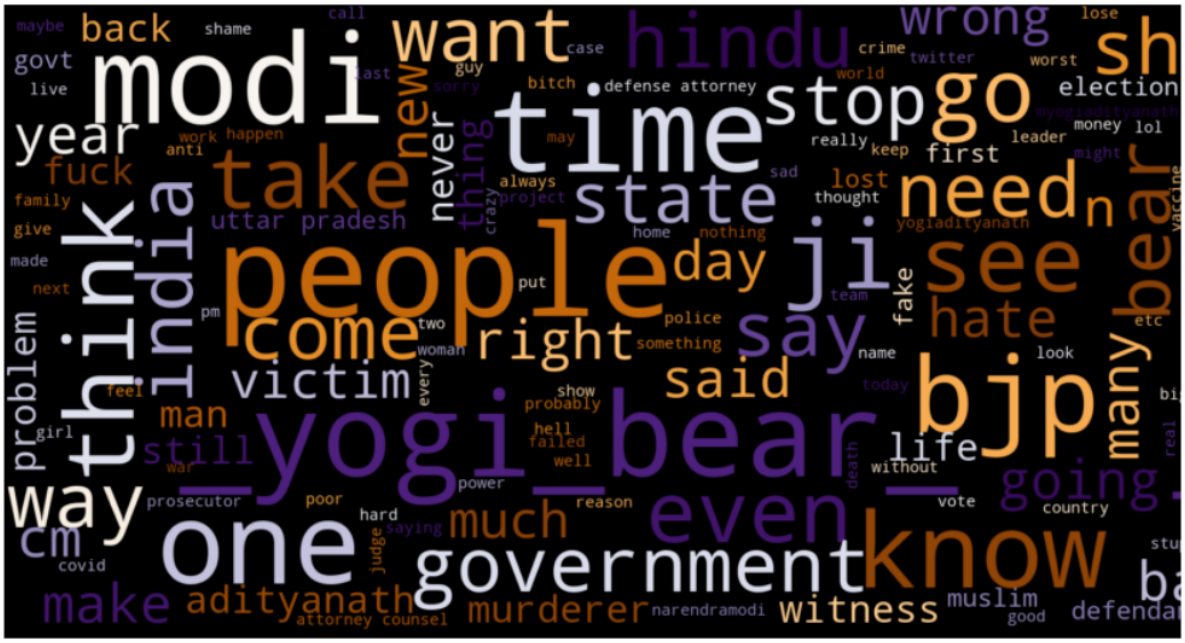
```
positive = df2[df2["sentiment"] == "Positive"]
negative = df2[df2["sentiment"] == "Negative"]
```

```
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 remov
    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']
```

16

```
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,)
```

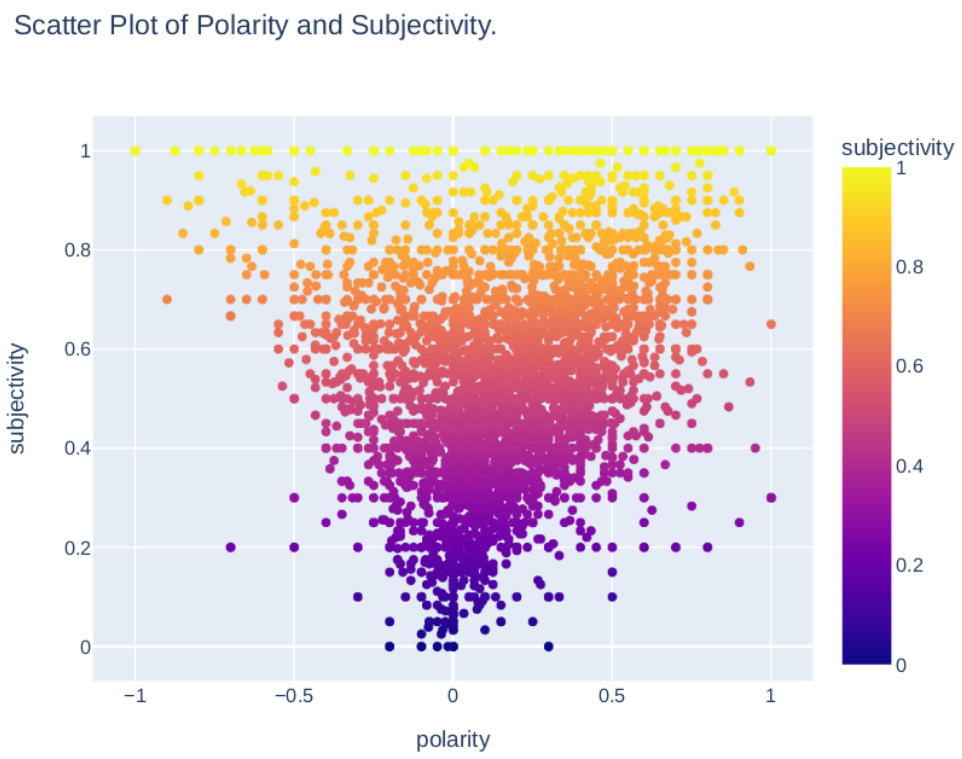
Here we are applying the Textblob for gettting the polarity and subjectivity of the tweets that we have featched

```
from textblob import TextBlob
# Function to assign polarity and subjectivity to the tweets
def blob_fun(text):
  senti = TextBlob(text)
  senti_polarity = senti.sentiment.polarity
  senti_subjectivity = senti.sentiment.subjectivity
  result = {'polarity':senti_polarity,'subjectivity':senti_subjectivity} ### Here we are su
  return result
```

```
df2['result'] = df2['Tweet'].apply(blob_fun)  ### appling the previous function
```

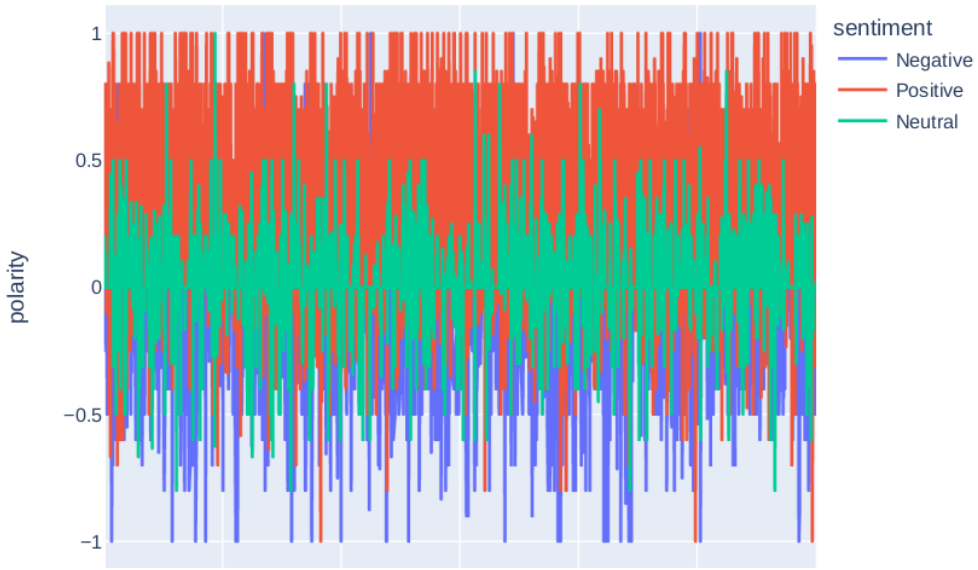
```
df2 = df2.join(pd.json_normalize(data=df2['result']))  ### Normalising the json format to d
```

```
import plotly.express as px

fig = px.scatter(df2, x="polarity", y="subjectivity", color='subjectivity' ,  title="Scatte
fig.show()
```




```
fig = px.line(df2, y="polarity", title='Polarity scale of each tweet in the dataset.' , col
fig.show()
```

Polarity scale of each tweet in the dataset.



Implementing Bag of word

```
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=0.2, random_state=42)
```

```
from sklearn.feature_extraction.text import TfidfVectorizer as tfidf # Applying TFIDF vecto
tfidf_vect = tfidf(use_idf=True , lowercase=False , max_features=8000)


X_trn = tfidf_vect.fit_transform(X_trn) # X_trn is text data
X_tst = tfidf_vect.transform(X_tst) # X_trn is text data
```

```
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}")
```

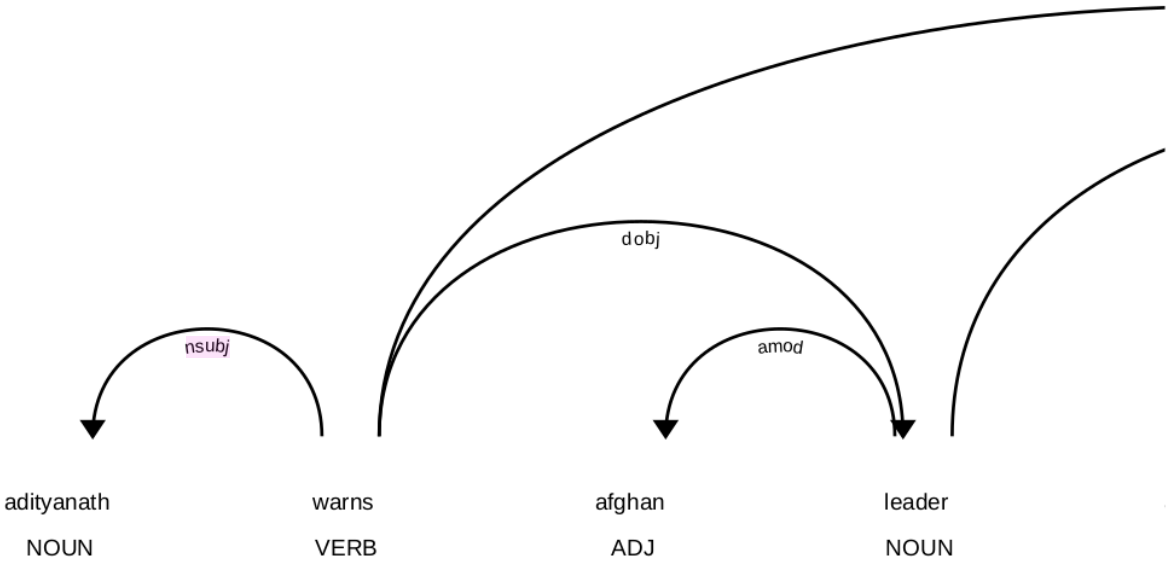
```
The shape of Training data is (9600, 8000) , (9600,)
The shape of Testing data is (2400, 8000) , (2400,)
```

Here we are using the spacy Liberary for identifing the verb , noun , pronouns in the sentance

---

Also here we are searching for Name and Entity

```
from spacy import displacy
import spacy
# Load the installed model "en_core_web_sm"
nlp = spacy.load("en_core_web_sm")

doc = nlp(X[1])
img = displacy.render(doc, style="dep" , jupyter=True)
```




```
doc = nlp(X[1])
# Text and label of named entity span
[(ent.text, ent.label_) for ent in doc.ents]
```

```
[('afghan', 'NORP')]
```

```
doc = nlp(X[1])
displacy.render(doc, style="ent", jupyter=True)
```

adityanath warns afghan NORP leader another adventure future via youtube

This is the code for Generating the report that include Confusion matrix , classification report , accuracy , precession ,recall, F1score .

```
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 is a tailored function for producing reports and obtaining confusion matrices.'

    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("  ")
```

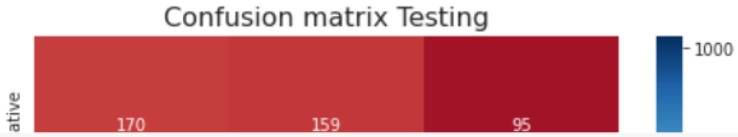
LGBM

```
# build the lightgbm model
import lightgbm as lgb
clf = lgb.LGBMClassifier()
clf.fit(X_trn, y_trn)

# predict the results
y_pred=clf.predict(X_tst)
```

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

	precision	recall	f1-score	support
Negative	0.73	0.40	0.52	424
Neutral	0.71	0.95	0.81	761
Positive	0.90	0.84	0.87	1215
accuracy			0.80	2400
macro avg	0.78	0.73	0.73	2400
weighted avg	0.81	0.80	0.79	2400

```
================================================
Testing Accuracy   : 0.8
Testing precision  : 0.81
Testing recall     : 0.8
Testing f1score    : 0.79
================================================
```




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

	precision	recall	f1-score	support
Negative	0.92	0.56	0.69	1651
Neutral	0.72	0.97	0.83	2950
Positive	0.93	0.86	0.89	4999
accuracy			0.84	9600
macro avg	0.86	0.80	0.81	9600

MultiLayer Perceptron

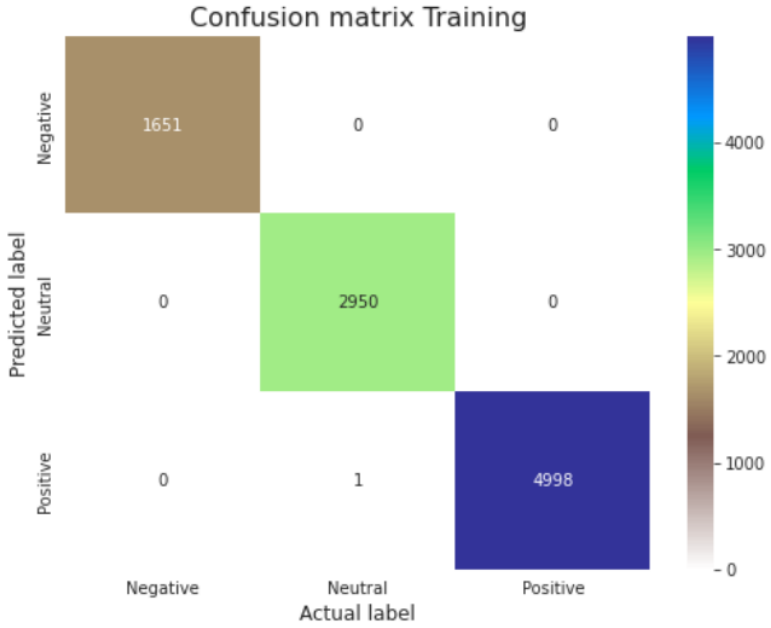
```
from sklearn.neural_network import MLPClassifier

# Neural nets
model = MLPClassifier()
model.fit(X_trn, y_trn)
y_train_hat = model.predict(X_trn)
y_test_hat = model.predict(X_tst)
```

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

```
                precision    recall  f1-score   support

      Negative       0.67      0.62      0.64       424
       Neutral       0.80      0.83      0.82       761
```

```
report_generator(y_trn , y_train_hat , "Training") ### Preciction and the Report
```

```
                precision    recall  f1-score   support

      Negative       1.00      1.00      1.00      1651
       Neutral       1.00      1.00      1.00      2950
      Positive       1.00      1.00      1.00      4999

      accuracy                           1.00      9600
     macro avg       1.00      1.00      1.00      9600
  weighted avg       1.00      1.00      1.00      9600


================================================
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['Text'].iloc[0]
```

```
'@priyankagandhi मेडम वो EVM वाले ट्वीट कितने बजे से करना है ?\nया इस बार Toolkit में कुछ और है ?\n#Ele
sults \n#UPElectionResult2022\n#UttarakhandElections2022 \n#ManipurElections2022 \n#Go
ons2022 \n#PunjabElections2022'
```

As Here we can see that the most of the characters are in Hindi so first of all we will treat them by translating the character into English and then we'll apply further Preprocessing of the text

---

```
goa_df['Text'] = goa_df['Text'].progress_apply(lambda text: trans(text))
```

```
100%|██████████| 21879/21879 [25:24<00:00, 14.35it/s]
```

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

```
100%|██████████| 5495/5495 [05:37<00:00, 16.27it/s]
```

```
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 [00:11<00:00, 1917.86it/s]
100%|██████████| 5495/5495 [00:01<00:00, 3816.88it/s]
100%|██████████| 49434/49434 [00:15<00:00, 3109.31it/s]
100%|██████████| 65016/65016 [00:24<00:00, 2673.22it/s]
100%|██████████| 18599/18599 [00:06<00:00, 2725.65it/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(clf.predict(i))
```

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

```
predictions
Neutral        13044
Positive        7352
Negative        1483
```

```
dtype: int64
====================================
predictions
Neutral        3718
Positive       1507
Negative        270
dtype: int64
====================================
predictions
Neutral        37920
Positive        8552
Negative        2962
dtype: int64
====================================
predictions
Neutral        54898
Positive        7623
Negative        2495
dtype: int64
====================================
predictions
Neutral        16267
Positive        1875
Negative         457
dtype: int64
====================================
```

```
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, 238387.88it/s]
100%|██████████| 5495/5495 [00:00<00:00, 198762.47it/s]
100%|██████████| 49434/49434 [00:00<00:00, 206317.07it/s]
100%|██████████| 65016/65016 [00:00<00:00, 195501.25it/s]
100%|██████████| 18599/18599 [00:00<00:00, 214837.50it/s]
```

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

```
others    34385
```

```
aap        6304
bjp        5805
inc        2940
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 == "inc" 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"]), a
```

```
dfs[1]
```

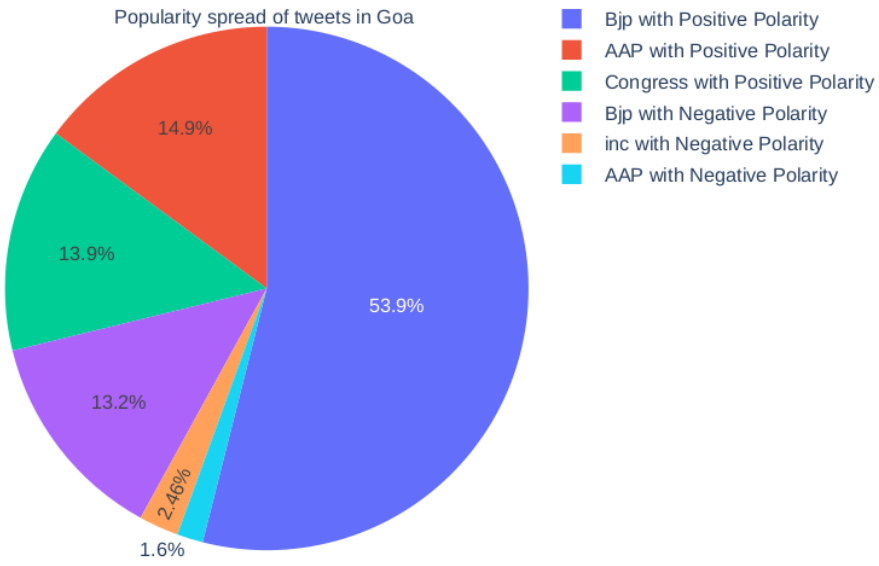
	index	Unnamed: 0	Datetime	Tweet Id	Text	Username	su
2	15	15	2022-03-09 17:45:53+00:00	1501615270607011847	personal prediction manipurelections bjp winni...	iamANKUR_96	bj
4	18	18	2022-03-09 17:34:31+00:00	1501612407298596868	dainik bhaskar special song launched	adityatiwarilive	in

```
import plotly.graph_objects as go

def show_plt(df , state):
  x = df.party_sentiments.value_counts().sort_values()
  x.to_dict()
  labels = x.keys()
  values = x.values
  fig = go.Figure(data=[go.Pie(labels=labels, values=values, title=f"Popularity spread of t
  fig.show()
```

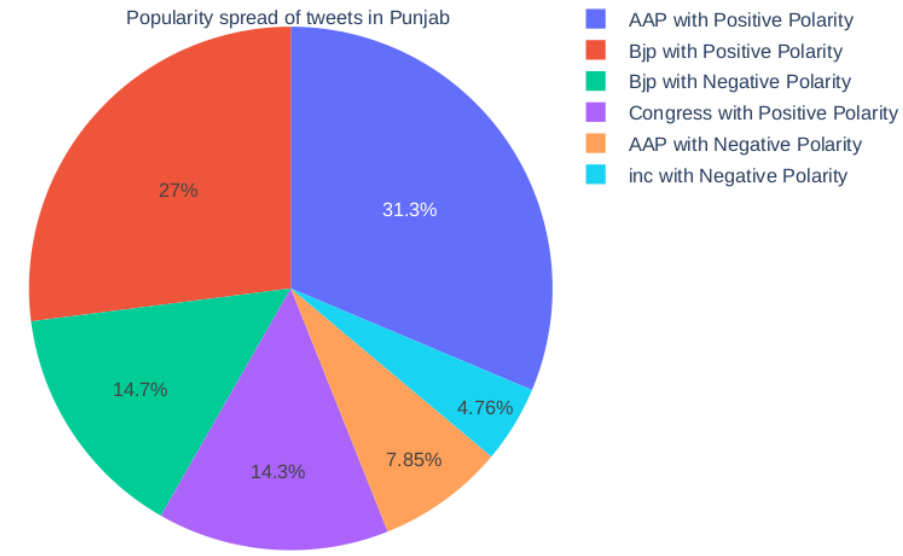
12	32	32	[illegible]	1501587123996028930	[illegible]	iamAiavSinh	bi

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

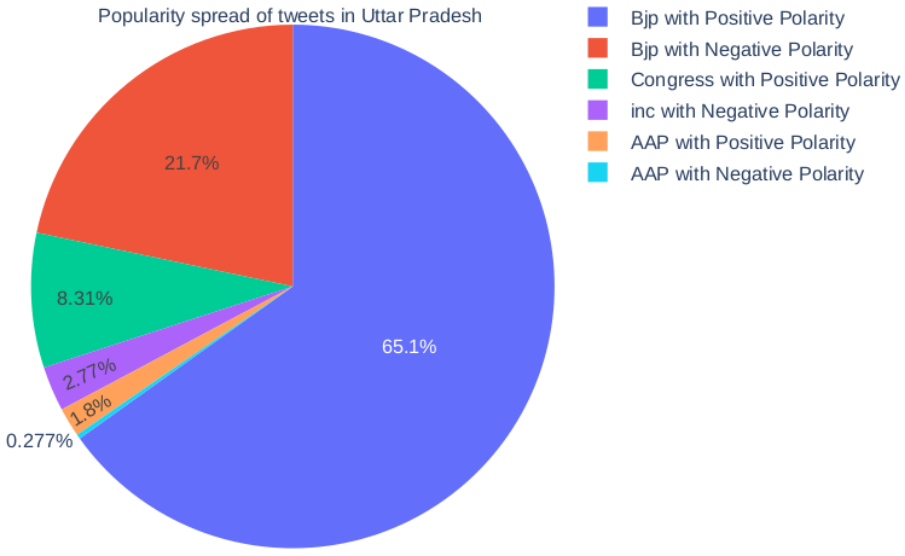
```
show_plt(dfs[0] , state[0])
```



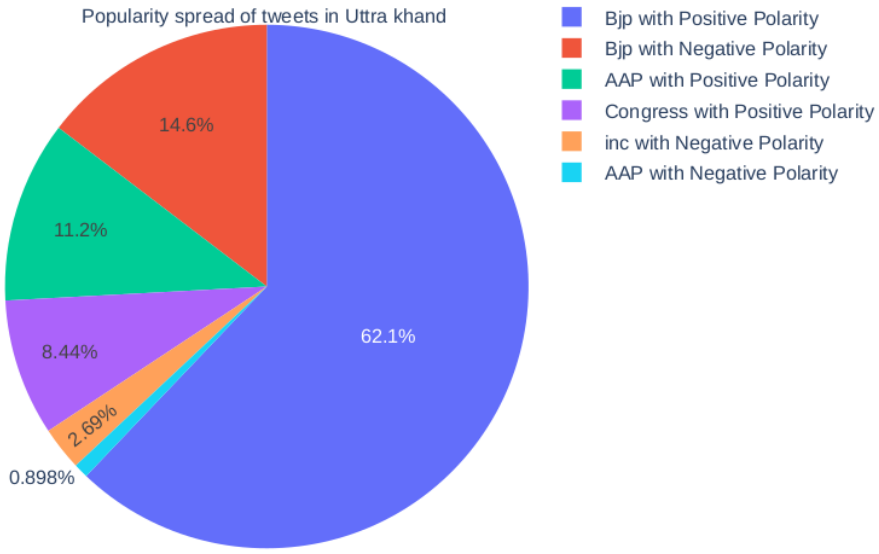

```
show_plt(dfs[2] , state[2])
```




```
show_plt(dfs[3] , state[3])
```



```
show_plt(dfs[4] , state[4])
```




```
show_plt(dfs[1] , state[1])
```

```
for i , j in zip(dfs,state):
    x = i.party_sentiments.value_counts().sort_values().keys()[-1].split()[0]
    print(f'The Predicted Winning party in the state {j} is {x}')
    print(' ')
```

```
The Predicted Winning party in the state Goa is Bjp

The Predicted Winning party in the state Manipur is Bjp

The Predicted Winning party in the state Punjab is AAP

The Predicted Winning party in the state Uttar Pradesh is Bjp

The Predicted Winning party in the state Uttra khand is Bjp
```

Minor_propose.i

ORIGINALITY REPORT

18%	11%	9%	15%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to University of Hertfordshire Student Paper	5%
2	Submitted to University of Surrey Student Paper	2%
3	overfitter.github.io Internet Source	1%
4	Submitted to University of New South Wales Student Paper	1%
5	Submitted to Queen Mary and Westfield College Student Paper	1%
6	Submitted to University of Sydney Student Paper	1%
7	Submitted to Harrisburg University of Science and Technology Student Paper	1%
8	towardsdatascience.com Internet Source	1%
9	Submitted to University of Edinburgh	

	Student Paper	1 %
10	dataanablog.com Internet Source	<1 %
11	affectivetweets.cms.waikato.ac.nz Internet Source	<1 %
12	www.coursehero.com Internet Source	<1 %
13	www.datacamp.com Internet Source	<1 %
14	Submitted to AUT University Student Paper	<1 %
15	Submitted to Hong Kong Baptist University Student Paper	<1 %
16	Submitted to International Business School Student Paper	<1 %
17	Submitted to University of San Francisco Student Paper	<1 %
18	Akshay Kulkarni, Adarsha Shivananda. "Natural Language Processing Recipes", Springer Science and Business Media LLC, 2021 Publication	<1 %
19	Submitted to National University of Ireland, Galway	<1 %

Student Paper

20	Nilesh Kumar, Bharat Mishra, M. Shahid Mukhtar. "A pipeline of integrating transcriptome and interactome to elucidate central nodes in host-pathogens interactions", STAR Protocols, 2022 Publication	<1 %
21	Submitted to University of South Florida Student Paper	<1 %
22	norma.ncirl.ie Internet Source	<1 %
23	Submitted to University of Portsmouth Student Paper	<1 %
24	doku.pub Internet Source	<1 %
25	Arvind Thorat. "APPLICATIONS OF ARTIFICIAL INTELLIGENCE IN CYBER SECURITY", Institute of Electrical and Electronics Engineers (IEEE), 2021 Publication	<1 %
26	Akshay Kulkarni, Adarsha Shivananda, Anoosh Kulkarni. "Natural Language Processing Projects", Springer Science and Business Media LLC, 2022 Publication	<1 %
27	www.kaggle.com Internet Source	

<1%

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		