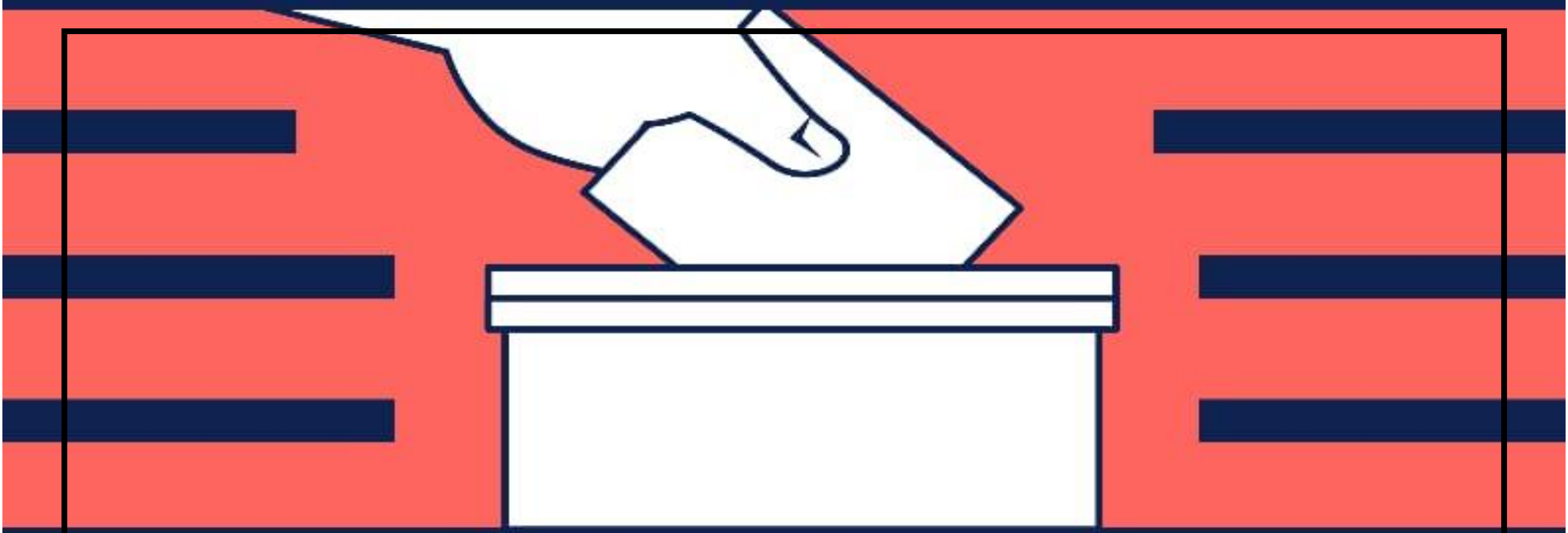


ONE DECISION CAN CHANGE THE FUTURE

LOKSABHA ELECTION ANALYSIS

Public Sentiments Matter

GROUP NO 1

INTRODUCTION

1. We live in a democratic country which follows parliamentary democracy.So president is mainly the head of the country and the entire power is vested in parliament leaded by the Prime Minister.

2.The central power is divided into two houses,Upper house called as Rajya Sabha and the lower house called as Lok Sabha.

3.The people who are elected as MP from a particular constituency represent in Lok Sabha.

4.The Lok Sabha elections are held every 5 years.

5.Here we have analyzed different patterns associated with Voting

Data Sets Used :

LS_3yr_data.xlsx : contains three years loksabha data
Source : kaggle

Twitter_data.csv : contains twitter data of election season
Source : kaggle

LS_2.0.csv : contains 2019 election result data
Sourcr : Kaggle

Data Cleaning

Initially we clean and merge our data
REQUIRED FEATURES
For our project we consider candidates who stood for the last three Lok Sabha Elections. We are interested in these characteristics of a candidate:
State,Total number of votes,Winner,Criminal,Cases,Education,Assets,Liabilities,Year,Constituency,Name,Gender ,Category,Age,Party

#deleting unnecessary columns

```
df09_1.drop(columns =['ST_CODE','Month','PC Type','PC Number'],inplace=True)
#renaming columns
df09_1.rename(columns={'State name':'STATE','Year':'YEAR','PC name':'CONSTITUENCY','Candidate Name':'NAME','Candidate Sex':'GENDER', 'Candidate Category':'CATEGORY', 'Candidate Age':'AGE','Party Abbreviation':'PARTY', 'Total Votes, Polled':'TOTAL_VOTES','Position':'WINNER'},inplace=True)
#modifying winner column to display 1 for winner, and 0 non-winner
for j in range(df09_1.shape[0]):
  if(df09_1.iat[j,9] != 1):
    df09_1.iat[j,9] = 0
```

#deleting unnecessary columns

```
df09_2.drop(columns =['Party','Education','Age','Constituency','Winner','Gender'],inplace=True)
#renaming columns
df09_2.rename(columns={'Candidate':'NAME','Criminal Cases':'CRIMINAL_CASES', 'Total Assets':'ASSETS','Liabilities':'LIABILITIES'},inplace=True)

#Merging the two datasets on left join and dropping duplicate values
df09_2['NAME'] = df09_2['NAME'].str.upper()
df1  = pd.merge(df09_1,df09_2,on= 'NAME',how= 'left')
df1.drop_duplicates(['NAME'],keep ='first',inplace=True)
df3 =pd.read_csv("/content/kaggle_2019.csv")
#delete unnecessary column
df3.drop(columns =['GENERAL\nVOTES', 'POSTAL\nVOTES','OVER TOTAL ELECTORS \nIN
CONSTITUENCY', 'OVER TOTAL VOTES POLLED \nIN CONSTITUENCY','TOTAL
ELECTORS'],inplace=True)
```

# rename invalid column names

```
df3.rename(columns={'CRIMINAL\nCASES': 'CRIMINAL_CASES',
'GENERAL\nVOTES':'GENERAL_VOTES','POSTAL\nVOTES': 'POSTAL_VOTES',
'TOTAL\nVOTES': 'TOTAL_VOTES','OVER TOTAL ELECTORS \nIN CONSTITUENCY':
'OVER_TOTAL_ELECTORS_IN_CONSTITUENCY', 'OVER TOTAL VOTES POLLED \nIN
CONSTITUENCY': 'OVER_TOTAL_VOTES_POLLED_IN_CONSTITUENCY', 'TOTAL
ELECTORS': 'TOTAL_ELECTORS'},inplace=True)

df3['YEAR']=2019
df3=df3.reindex(columns=['STATE','YEAR','CONSTITUENCY','NAME','GENDER','CATEGORY','AG
E','PARTY','TOTAL_VOTES','WINNER','CRIMINAL_CASES','ASSETS','LIABILITIES'])
def value_cleaner(x):
   try:
    str_temp = (x.split('Rs')[1].split('\n')[0].strip())
    str_temp_2 = ''
    for i in str_temp.split(","):
       str_temp_2 = str_temp_2+i
    return str_temp_2
   except:
    x = 0
    return x
df3['ASSETS'] = df3['ASSETS'].apply((value_cleaner))
df3['LIABILITIES'] = df3['LIABILITIES'].apply((value_cleaner))
```

Combining Datasets

```
df3['YEAR']=2019
df3=df3.reindex(columns=['STATE','YEAR','CONSTITUENCY','NAME','GENDER','CATEGORY','AG
E','PARTY','TOTAL_VOTES','WINNER','CRIMINAL_CASES','ASSETS','LIABILITIES'])
def value_cleaner(x):
   try:
    str_temp = (x.split('Rs')[1].split('\n')[0].strip())
    str_temp_2 = ''
    for i in str_temp.split(","):
       str_temp_2 = str_temp_2+i
    return str_temp_2
```

```
 except:
        x = 0
        return x
df3['ASSETS'] = df3['ASSETS'].apply((value_cleaner))
df3['LIABILITIES'] = df3['LIABILITIES'].apply((value_cleaner))
#Filling education, criminal cases, assets, liabilities with modal value
dataset['EDUCATION'].fillna('Graduate',inplace=True)
dataset['CRIMINAL_CASES'].fillna('0',inplace=True)
dataset['ASSETS'].fillna('0',inplace=True)
dataset['LIABILITIES'].fillna('0',inplace=True)
dataset['YEAR'] = dataset['YEAR'].astype(str)

dataset = dataset.dropna()
votes_col = []
series =dataset.groupby('CONSTITUENCY').sum().TOTAL_VOTES
for row in dataset.itertuples():
  cons = row.CONSTITUENCY
  votes_col.append(series[cons])
print(len(votes_col))
dataset['TOTAL_ELECTORS']= votes_col
dataset.info()
```

DATA VISUALIZATION

Introduction

1. Data visualization is the process of translating large data sets and metrics into charts, graphs and other visuals. This representation makes it easier to identify outliers, real-time trends and insights about the information represented in the data.

2. The dataset used has three year(i.e 2009,2014 & 2019) information of Lok sabha elections.

3. For visualizations, year-wise data is considered to observe the trends in the characteristics of the dataset.

Visualization

- The dataset has data of years 2009,2014 and 2019
- For visualization, we have considered some parameters which may play a role in winning of elections.
- All the parameters were tested for year-wise data to draw meaningful interpretations.

Importing packages:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
```

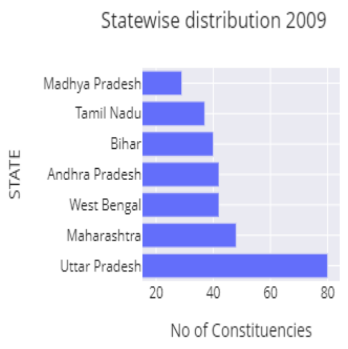
First the dataset was divided year-wise.

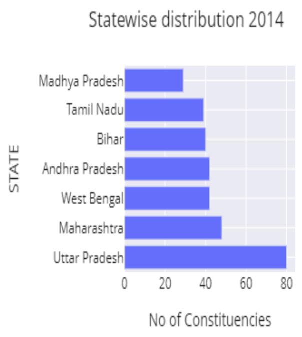
```
df9 = d[d['YEAR'] == 2009]
df14 = d[d['YEAR']== 2014]
df19 = d[d['YEAR']== 2019]
```

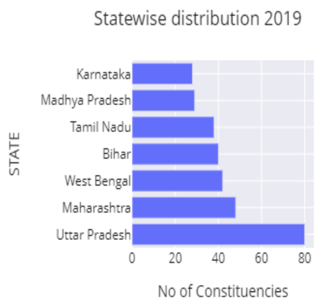
1) State-wise distribution of constituencies

The first objective was to find out the statewise distribution of seats. A bar chart of the top 7 states with maximum number of constituencies is plotted.

```
statecon1=df9.groupby('STATE').apply(lambda
x:x['CONSTITUENCY'].nunique()).reset_index(name='No of Constituencies')
statecon1.sort_values(by='No of Constituencies',ascending=False,inplace=True)
fig1 = px.bar(statecon1.head(7), x='No of Constituencies', y='STATE',orientation='h',height=300,
width=400)
fig1.update_layout(title_text='Statewise distribution 2009',template='seaborn')
fig1.show()
```







Interpretations:

1)In 2009, Uttar Pradesh had the maximum number of constituencies followed by Maharashtra and West Bengal.

2)Uttar Pradesh still had the most number of constituencies. Two more constituencies were added in Tamil Nadu in 2014.

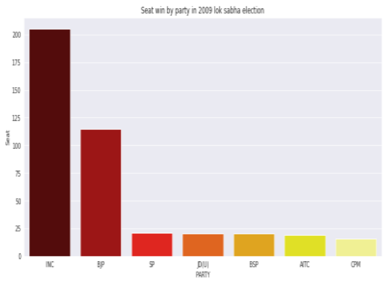
3)The number of constituencies in Andhra Pradesh has fallen from 2014 to 2019.

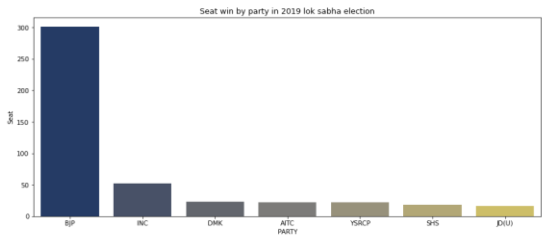
2) Party-wise distribution of seats

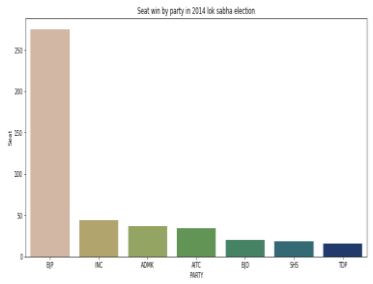
The next objective is to get the seats won by each party. A bar chart was plotted to observe the trends year-wise.

Code:

```
winning_candidates_per_party_9 =
df9.groupby(['PARTY'])['WINNER'].sum().reset_index().sort_values('WINNER',ascending =
False)
winning_candidates_per_party_9 =
winning_candidates_per_party_9[winning_candidates_per_party_9['WINNER'] > 0]
winning_2009 = winning_candidates_per_party_9.head(10)
sns.set_style('darkgrid')
plt.figure(figsize = (14,6))
sns.barplot(data = winning_2009, x = 'PARTY', y = 'WINNER', palette = 'hot')
plt.title('Seat win by party in 2009 lok sabha election')
plt.ylabel('Seat');
```







Interpretations:

1):In the 2009 election INC(congress) won the maximum seat, followed by BJP and SP.

2) In 2014, BJP(Bharatiya Janata Party) won the maximum seat and also was able to cross the majority mark of 274 seats.

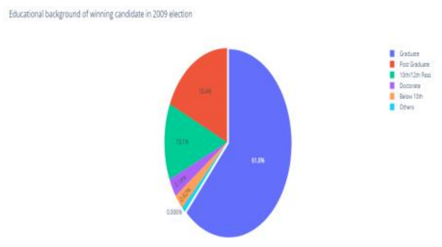
3) In 2019 also BJP(Bharatiya Janata Party) won the maximum seat, followed by INC and DMK.

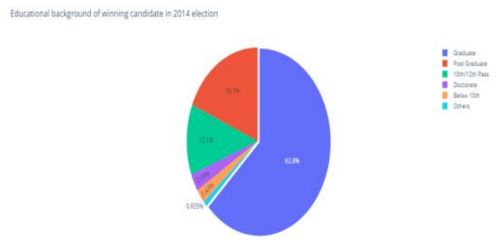
3) Educational qualifications of Leaders.

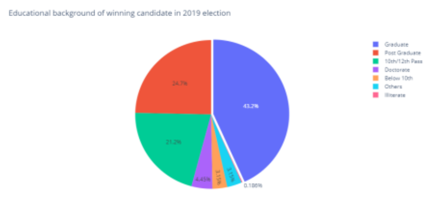
A pie chart is plotted to get the educational qualifications of our leaders. We wanted to see how much percent of our leaders were educated.

Code:

```
d['EDUCATION'].value_counts()
d1 = d.copy()
d1.loc[d1['EDUCATION'] == '8th Pass', 'EDUCATION'] = 'Below 10th'
d1.loc[d1['EDUCATION'] == '5th Pass', 'EDUCATION'] = 'Below 10th'
d1.loc[d1['EDUCATION'] == 'Literate', 'EDUCATION'] = 'Below 10th'
d1.loc[d1['EDUCATION'] == 'Graduate Professional', 'EDUCATION'] = 'Graduate'
d1.loc[d1['EDUCATION'] == '10th Pass', 'EDUCATION'] = '10th/12th Pass'
d1.loc[d1['EDUCATION'] == '12th Pass', 'EDUCATION'] = '10th/12th Pass'
d1 = d1[d1['EDUCATION']!= 'Not Available']
d_9 = d1[d1['YEAR'] == 2009]
d_4 = d1[d1['YEAR']== 2014]
dd_9 = d_9.EDUCATION.value_counts().rename_axis('EDUCATION').reset_index(name =
'COUNTS')
dd_4 = d_4.EDUCATION.value_counts().rename_axis('EDUCATION').reset_index(name =
'COUNTS')
d_99 = d_9.copy()
d_99 = d_99[d_99['WINNER'] == 1]
f_9 = d_99.EDUCATION.value_counts().rename_axis('EDUCATION').reset_index(name =
'COUNTS')
fig8 = go.Figure(data = [go.Pie(labels= f_9.EDUCATION , values = f_9.COUNTS, pull =
[0.03,0,0,0,0,0,0])])
fig8.update_layout(title_text=('Educational background of winning candidate in 2009 election'))
```







Interpretation:

1)In 2009,62% of winning candidate are graduate and only 2.62% are below matric pass

2) 63% of winning candidate are graduate and only 3% of have doctorate degree in 2014.

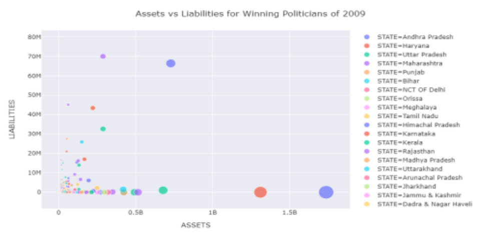
3) 43% of winning candidates are graduates and 25% are post graduates and just 0.3% were illiterate in 2019.

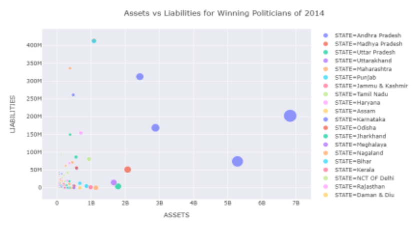
4) Assets and liabilities of winning politicians.

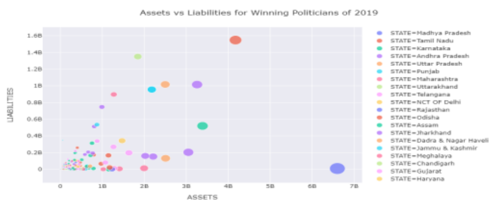
The next objective is to get the assets and liabilities of winning politicians. We made a scatter plot to get all the details of candidates have large amounts of assets and liabilities.

Code:

```
assetsliab1=df9[['NAME','PARTY','ASSETS','LIABILITIES','STATE','CONSTITUENCY','WI
NNER']]
assetsliab1=assetsliab1[assetsliab1['WINNER']==1]
assetsliab1.sort_values(by='ASSETS',ascending=False,inplace=True)
fig1 = px.scatter(assetsliab1, x='ASSETS', y='LIABILITIES',
              color='STATE',size='ASSETS',
              hover_data=(['NAME','PARTY','CONSTITUENCY','STATE']))
fig1.update_layout(title_text='Assets vs Liabilities for Winning Politicians of
2009',template='seaborn')
fig1.show()
```







Interpretation:

1)In 2009, a TDP (Telugu Desam Party) leader had assets around 1.8 billion with 0 liabilities. The highest liability of a leader recorded that year was 70 million.

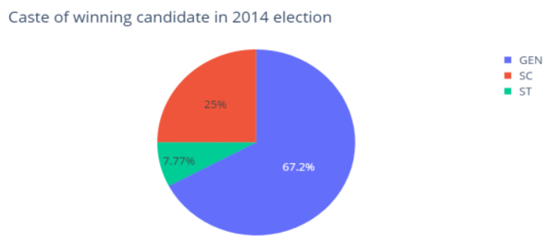
2) In 2014, a TDP (Telugu Desam Party) leader had assets around 7 billion.. The highest liability of a leader recorded that year was 415 million. Both assets and liabilities of leaders has

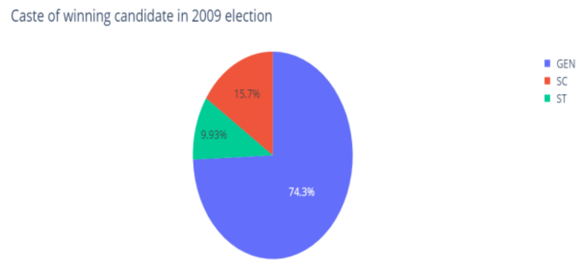
5)Caste of winning candidate

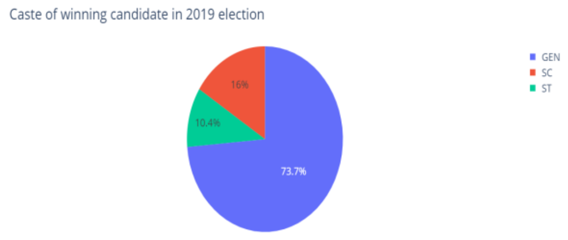
This objective focuses on the caste of the leaders. A pie chart is used to get the percentage of leaders belonging to each category.

Code:

```
caste_9 = d_99['CATEGORY'].value_counts().rename_axis('CATEGORY').reset_index(name =
'SEATS_WIN')
fig10 = go.Figure(data = [go.Pie(labels= caste_9.CATEGORY , values =
caste_9.SEATS_WIN)])
fig10.update_layout(title_text=('Caste of winning candidate in 2009 election'), font = dict(size =
20))
```







Interpretation:

1)74.3% of winning candidate were from general category and 15.7%, 9.93% were from SC, ST category respectively in 2009.

2) In 2014,67.2% of winning candidate were from general category and 25%, 7.77% were from SC, ST category respectively

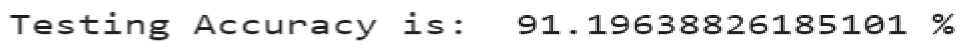
Prediction Models

First Model: Using KNeighborsClassifier

K-NEAREST NEIGHBORS (KNN) model

KNN is one of the simplest forms of machine learning algorithms mostly used for classification. It classifies the data point on how its neighbor is classified. KNN classifies the new data points based on the similarity measure of the earlier stored data points.
With the help of KNN Model we align the scattered and new data points into the dataset and thus increase our accuracy.

```
# separate train features and label
y = dataset["WINNER"]

X = dataset.drop(labels=["WINNER"], axis=1)

# split dataset into train and test data

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1, stratify=y)

# train and test knn model

knn = KNeighborsClassifier()

knn.fit(X_train, y_train)

pred = knn.predict(X_test)

print("Testing Accuracy is: ", knn.score(X_test, y_test)*100, "%")
```



```
Testing Accuracy is:  91.19638826185101 %
```

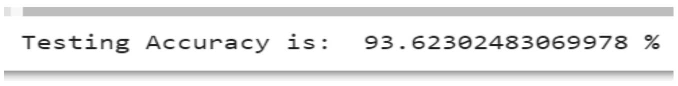
Second

Model: Using MinMaxScaler

USING MinMaxScaler

MinMaxScaler transforms feature by scaling each category to a given range.This estimator scales and translates each feature individually such that it is in the given range on the training set.For example:- between range 0 to 1.By applying the function we see a rise in the accuracy of data.

```
# scaling values into 0-1 range
scaler = MinMaxScaler(feature_range=(0, 1))
features = [
   'STATE', 'CONSTITUENCY', 'NAME', 'PARTY', 'GENDER', 'CRIMINAL_CASES', 'AGE',
'CATEGORY', 'EDUCATION', 'ASSETS',
'LIABILITIES','TOTAL_VOTES','TOTAL_ELECTORS']
dataset2[features] = scaler.fit_transform(dataset[features])
# separate train features and label
y = dataset2["WINNER"]
X = dataset2.drop(labels=["WINNER"], axis=1)
# split dataset into train and test data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1, stratify=y)
# train and test knn model
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
knn.predict(X_test)
print("Testing Accuracy is: ", knn.score(X_test, y_test)*100, "%
```



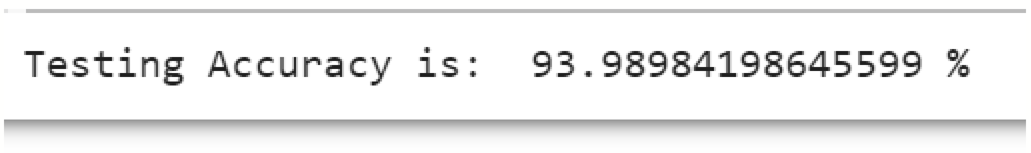
```
Testing Accuracy is:  93.62302483069978 %
```

Third Model

Modifying existing features

- We encode “EDUCATION” column values in numerical form for prediction.
- In order to reduce the errors we then replace the name of the given PARTY as “others”.In this way we remove the unusual variables.
- We then apply the previous model to our new formed dataset and check the accuracy of the model.

```
encoded_edu = []
# iterate through each row in the dataset
for row in dataset1.itertuples():
    education = row.EDUCATION
    if education == "Illiterate":
        encoded_edu.append(0)
    elif education == "Literate":
        encoded_edu.append(1)
    elif education == "5th Pass":
        encoded_edu.append(2)
    elif education == "8th Pass":
        encoded_edu.append(3)
    elif education == "10th Pass":
        encoded_edu.append(4)
    elif education == "12th Pass":
        encoded_edu.append(7)
    elif education == "Graduate":
        encoded_edu.append(8)
    elif education == "Post Graduate":
        encoded_edu.append(9)
    elif education == "Graduate Professional":
        encoded_edu.append(10)
    elif education == "Doctorate":
        encoded_edu.append(11)
    else:
        encoded_edu.append(5)
dataset1['EDUCATION'] = encoded_edu
dataset1['EDUCATION']
```



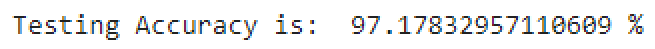
```
Testing Accuracy is:  93.98984198645599 %
```

Fourth Model: Adding New Features

For the given Model, we add new features and group according to the features added like we group for state ,party and constituency with respect to other columns and consequently we iterate through each column and thus compute our accuracy which comes out to be better than previously calculated.

```
# Preparing feature values
cons_per_state = {}
voters_per_state = {}
party_winningSeats = {}
party_criminal = {}
party_education = {}
party_totalCandidates_per_cons = {}
party_winningSeats_per_cons = {}
party_criminal_per_cons = {}
party_education_per_cons = {}

# group by state
subset = dataset[['STATE', 'CONSTITUENCY', 'TOTAL_ELECTORS']]
gk = subset.groupby('STATE')
# for each state
for name,group in gk:
    # total constituencies per state
    cons_per_state[name] = len(group)

    # total voters per state
    voters_per_state[name] = group['TOTAL_ELECTORS'].sum()
# group by party
subset = dataset[['PARTY', 'CONSTITUENCY', 'CRIMINAL_CASES', 'EDUCATION',
'WINNER']]
gk = subset.groupby('PARTY')
# for each party
for name,group in gk:
    # winning seats by party
    party_winningSeats[name] = group[group['WINNER'] == 1.0].shape[0]
    # criminal cases by party
    party_criminal[name] = group['CRIMINAL_CASES'].sum()

    # education qualification by party (sum of candidates)
    party_education[name] = group['EDUCATION'].sum()

    # group by constituency
    gk2 = group.groupby('CONSTITUENCY')
    # for each constituency
    for name2, group2 in gk2:
        key = str(name2)+'_'+str(name)    # cons_party

        # total candidates by party in constituency
        party_totalCandidates_per_cons[key] = len(group2)

        # party winning seats in the constituency
        party_winningSeats_per_cons[key] = group2[group2['WINNER'] == 1.0].shape[0]

        # criminal cases by party in the constituency
        party_criminal_per_cons[key] = group2['CRIMINAL_CASES'].sum()
# education qualification by party in constituency (sum of candidates)
        party_education_per_cons[key] = group2['EDUCATION'].sum()

# Applying feature values
# new feature columns
total_cons_per_state = []
total_voters_per_state = []
total_voters_per_cons = []
winning_seats_by_party = []
criminal_by_party = []
education_by_party = []
total_candidates_by_party_per_cons = []
winning_seats_by_party_per_cons = []
criminal_by_party_per_cons = []
education_by_party_per_cons = []
# iterate through each row in the dataset
```

```
for row in dataset.itertuples():
    subkey = str(row.CONSTITUENCY)+'_'+str(row.PARTY)
    total_cons_per_state.append(cons_per_state.get(row.STATE))
    total_voters_per_state.append(voters_per_state.get(row.STATE))
    winning_seats_by_party.append(party_winningSeats.get(row.PARTY))
    criminal_by_party.append(party_criminal.get(row.PARTY))
    education_by_party.append(party_education.get(row.PARTY))
    total_candidates_by_party_per_cons.append(party_totalCandidates_per_cons.get(subkey))
    winning_seats_by_party_per_cons.append(party_winningSeats_per_cons.get(subkey))
    criminal_by_party_per_cons.append(party_criminal_per_cons.get(subkey))
    education_by_party_per_cons.append(party_education_per_cons.get(subkey))
# append columns to dataset
dataset['total_cons_per_state'] = total_cons_per_state
dataset['total_voters_per_state'] = total_voters_per_state
dataset['winning_seats_by_party'] = winning_seats_by_party
dataset['criminal_by_party'] = criminal_by_party
dataset['education_by_party'] = education_by_party
dataset['total_candidates_by_party_per_cons'] = total_candidates_by_party_per_cons
dataset['winning_seats_by_party_per_cons'] = winning_seats_by_party_per_cons
dataset['criminal_by_party_per_cons'] = criminal_by_party_per_cons
dataset['education_by_party_per_cons'] = education_by_party_per_cons
dataset
```



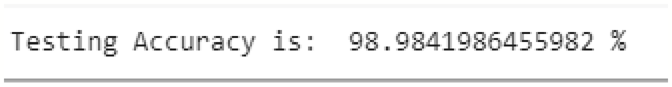
```
Testing Accuracy is:  97.17832957110609 %
```

Fifth Model: Identifying features with more weightage

In this final model to increase the accuracy of our prediction ,we decide our features which significantly affect the result and likewise we take consideration of those and bring out our final prediction.At last we conclude at the final model accuracy to be 98.98%

```
# apply SelectKBest class to extract top most features
bestfeatures = SelectKBest(score_func=chi2, k="all")
fit = bestfeatures.fit(X, y)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(X.columns)
# concat two dataframes for better visualization
featureScores = pd.concat([dfcolumns, dfscores], axis=1)
featureScores.columns = ['Specs', 'Score']
print(featureScores.sort_values('Score',ascending=False))

X.drop(labels=["CRIMINAL_CASES", "TOTAL_ELECTORS", "total_cons_per_state",
"ASSETS","total_voters_per_state", "CATEGORY",
"GENDER","NAME","CONSTITUENCY" ,"criminal_by_party_per_cons"], axis=1,
inplace=True
```

Applying KNN model:



```
Testing Accuracy is:  98.9841986455982 %
```

Sentiment Analysis:

Introduction:

1.Sentimental analysis is an approach to ML that identifies the emotional tone behind a body of text.

2.Almost 400 million tweets were made during the Lok Sabha Elections held in 2019. Tweets were made in English, Hindi as well as other languages

3. Our project aims to analyse all the 23rd hour tweets made between March and May and draw conclusions based on social media activity.

Importing Essential Packages:

Packages play a very important role in python.Package contains a large amount of classes and there are various methods assigned to these classes.

```
import re,csv
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import string
import nltk
from textblob import TextBlob
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KneighborsClassifier
```

Data Preprocessing:

- This is the first step in building a machine learning model. Data pre-processing refers to the transformation of data before feeding it into the model.
- It deals with the techniques that are used to convert unusable raw data into clean reliable data.

Cleaning Data:

Removing non-essential columns from our data:

```
data.drop(columns=['user_id', 'status_id','reply_to_status_id',
'reply_to_user_id','reply_to_screen_name','symbols','urls_url', 'urls_t.co',
       'urls_expanded_url', 'media_url', 'media_t.co', 'media_expanded_url','media_type',
'ext_media_url', 'ext_media_t.co','ext_media_expanded_url', 'ext_media_type','profile_url',
'profile_expanded_url', 'account_lang',
       'profile_banner_url', 'profile_background_url', 'profile_image_url'],inplace=True)
```

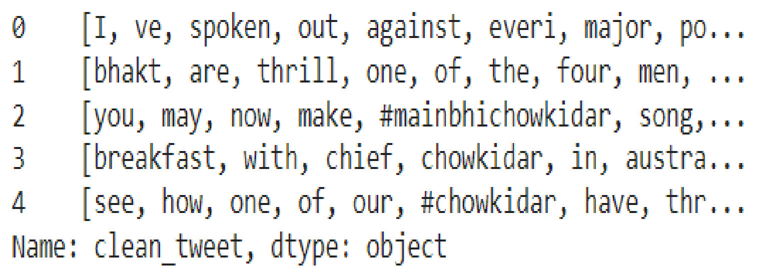
Removing special characters and creating a new column clean_tweet:

```
def remove_symbol(input_text, symbol):
    r = re.findall(symbol, input_text)
    for i in r:
        input_text = re.sub(i, '', input_text)

   return input_text
data['clean_tweet'] = np.vectorize(remove_symbol)(data['text'], "@[\w]*")
data["clean_tweet"] = data['text'].str.replace("[^a-zA-Z#&]", " ")
```

Tokenizing the tweet for creating lenient data:

1.The process of tokenizing basically involves breaking large statements into an array of words.

2.As the large statement contains a sequence of words and in order to analyze the sentiments of these words we need to separate these tweets into an array



```
0    [I, ve, spoken, out, against, everi, major, po...
1    [bhakt, are, thrill, one, of, the, four, men, ...
2    [you, may, now, make, #mainbhichowkidar, song,...
3    [breakfast, with, chief, chowkidar, in, austra...
4    [see, how, one, of, our, #chowkidar, have, thr...
Name: clean_tweet, dtype: object
```

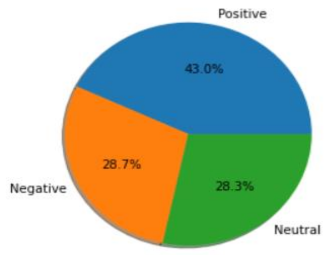
Analyzing the current tweets and bifurcating them according to their polarity

Sentiment analysis Using TextBlob:

Textblob is a library available in python which can be used for breaking down sentiments from a given statement.It basically uses Natural language Processing(NLP).

Code:

```
def analyze_sentiment(tweet):
    analysis = TextBlob(tweet)
    if analysis.sentiment.polarity > 0:
        return 1
    elif analysis.sentiment.polarity==0 :
        return 0
    elif analysis.sentiment.polarity<0:
        return -1
data_new['sentiment'] = np.array([ analyze_sentiment(tweet) for tweet in
data_new['clean_tweet']])
positive=data.loc[data_new.sentiment==1,'clean_tweet'].count()
negative=data.loc[data_new.sentiment==-1,'clean_tweet'].count()
neutral=daPlotting the sentiments:
labels='Positive','Negative','Neutral'
sizes=[positive,negative,neutral]
explode=None
fig1, ax1 = plt.subplots()
ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=0)
ax1.axis('equal')
plt.show()
```



Performing sentiment analysis on the basis of polarity of available tweets using Normalized Values.

As we have obtained the polarity in the previous part we shuffle them and normalize them.

Formula: (Value - Mean of Values)/Standard Deviation
Code:

```
def analyze_sentiment(i):
    if i >=0.05:
        return 1
    elif i<0.05 and i>-0.05 :
        return 0
    elif i<-0.05:
        return -1
data_new['sentiment1'] = np.array([ analyze_sentiment(i) for i in data_new['Compound']])
data_new1=data_new[['screen_name','text','clean_tweet','sentiment1']].copy()
Data_new1
```

Conversion to numeric for ease of model building:

We vectorize our categorical variables in our table so that they can be fed in our machine learning model.Vectorization plays an important role in preparing model specific data.

Code:

```
from sklearn.feature_extraction.text import CountVectorizer
bow_vectorizer = CountVectorizer(max_df=0.90, min_df=2, max_features=1000,
stop_words='english')  # bow = bag of words
# bag-of-words feature matrix
bow = bow_vectorizer.fit_transform(data_new['clean_tweet'])
```

Model Training

For accurately predicting data we have used three machine learning algorithms.
They are as follows:

1. Random Forest Classifier
2.Decision Tree Classifier
3.Gaussian Naive Bayes

Now we will see them one by one :

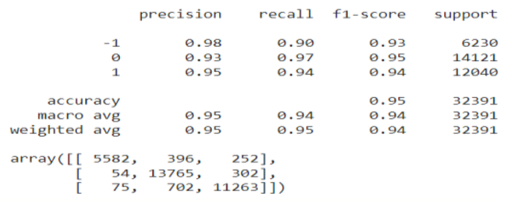
Random Forest Classifier:

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

Training Model - Random Forest Classifier:

Code:

```
X=bow.toarray()
Y=data_new['sentiment']
Y1=data_new1['sentiment1']
# splitting data into training and validation set
xtrain_bow,xtest_bow,ytrain_bow,ytest_bow = train_test_split(X,Y, random_state=42,
test_size=0.3)
xtrain_bow, xval_bow, ytrain_bow, yval_bow= train_test_split(xtrain_bow, ytrain_bow,
test_size=0.25, random_state=42)
rf= RandomForestClassifier()
rf.fit(xtrain_bow, ytrain_bow) # training the model
ypred_bow=rf.predict(xtest_bow)
print(classification_report(ytest_bow,ypred_bow))
Evaluating model Random Forest Classifier(Training set)
confusion_matrix(ytest_bow,ypred_bow
```

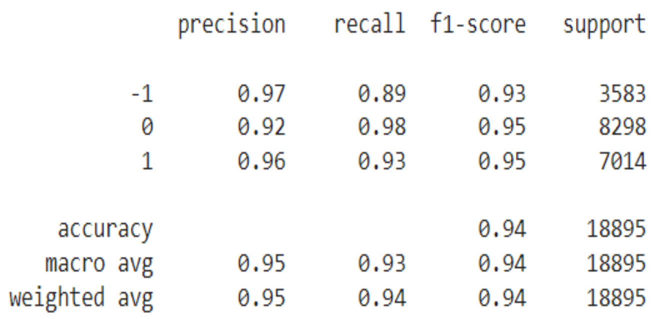


```
              precision    recall  f1-score   support

          -1       0.98      0.90      0.93      6230
           0       0.93      0.97      0.95     14121
           1       0.95      0.94      0.94     12040

    accuracy                           0.95     32391
   macro avg       0.95      0.94      0.94     32391
weighted avg       0.95      0.95      0.94     32391

array([[ 5582,   396,   252],
       [   54, 13765,   302],
       [   75,   702, 11263]])
```

Predicting Test values from model(Random forest Classifier):

Code:

```
ypredv_bow=rf.predict(xval_bow)
confusion_matrix(yval_bow,ypredv_bow)
print(classification_report(yval_bow,ypredv_bow))
```



```
              precision    recall  f1-score   support

          -1       0.97      0.89      0.93      3583
           0       0.92      0.98      0.95      8298
           1       0.96      0.93      0.95      7014

    accuracy                           0.94     18895
   macro avg       0.95      0.93      0.94     18895
weighted avg       0.95      0.94      0.94     18895
```

Decision Tree Classifier:

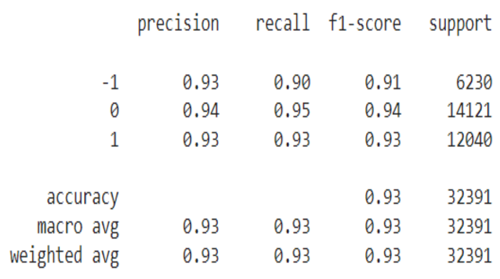
1.Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression.
2.The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.
Training model(Decision Tree classifier):
Code:

```
dt=DecisionTreeClassifier()
dt.fit(xtrain_bow, ytrain_bow)
ypred_bowdt=dt.predict(xtest_bow)
print(classification_report(ytest_bow,ypred_bowdt))
```

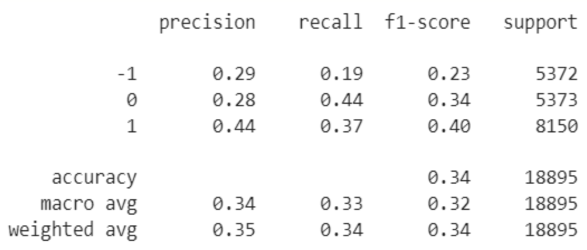
Testing model (Decision Tree Classifier):

Code :

```
ypredv_bowdt=dt.predict(xval_bow)
print(classification_report(yval_bow1,ypredv_bowdt))
```



```
              precision    recall  f1-score   support

          -1       0.93      0.90      0.91      6230
           0       0.94      0.95      0.94     14121
           1       0.93      0.93      0.93     12040

    accuracy                           0.93     32391
   macro avg       0.93      0.93      0.93     32391
weighted avg       0.93      0.93      0.93     32391
```

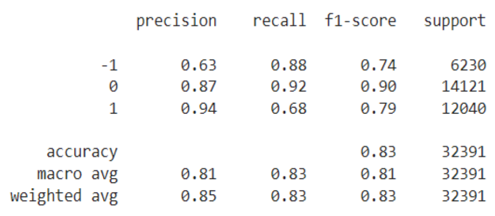


```
              precision    recall  f1-score   support

          -1       0.29      0.19      0.23      5372
           0       0.28      0.44      0.34      5373
           1       0.44      0.37      0.40      8150

    accuracy                           0.34     18895
   macro avg       0.34      0.33      0.32     18895
weighted avg       0.35      0.34      0.34     18895
```

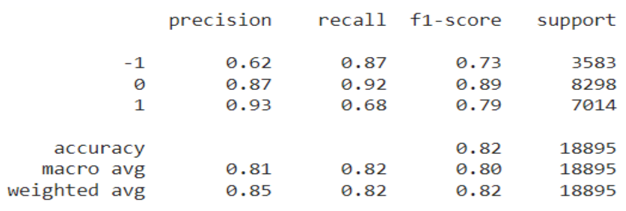
Gaussian Naive Bayes:

1.Gaussian Naive Bayes is a variant of Naive Bayes that follows Gaussian normal distribution and supports continuous data.
2.Naive Bayes are a group of supervised machine learning classification algorithms based on the **Bayes theorem.(training testing)**
Code:

```
nb=GaussianNB()
logistic=LogisticRegression()
svm=SVC()
knn=KNeighborsClassifier()
nb.fit(xtrain_bow, ytrain_bow)
ypred_bownb=nb.predict(xtest_bow)
print(classification_report(ytest_bow,ypred_bownb))
ypredv_bownb=nb.predict(xval_bow)
print(classification_report(yval_bow,ypredv_bownb))
```



```
              precision    recall  f1-score   support

          -1       0.63      0.88      0.74      6230
           0       0.87      0.92      0.90     14121
           1       0.94      0.68      0.79     12040

    accuracy                           0.83     32391
   macro avg       0.81      0.83      0.81     32391
weighted avg       0.85      0.83      0.83     32391
```



```
              precision    recall  f1-score   support

          -1       0.62      0.87      0.73      3583
           0       0.87      0.92      0.89      8298
           1       0.93      0.68      0.79      7014

    accuracy                           0.82     18895
   macro avg       0.81      0.82      0.80     18895
weighted avg       0.85      0.82      0.82     18895
```

Model Deployment:

- The whole process of machine learning does not just stop with model building and prediction.
- It also involves making use of the model to build an application with the new data.

We are doing standard scaling now:

Here we perform standard scaling on each of our categorical variables.

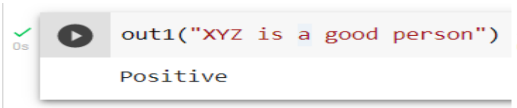
Code:

```
from sklearn.metrics import
classification_report,confusion_matrix,accuracy_score,precision_score,recall_score,roc_auc_sco
re,f1_score
sc = StandardScaler()
Xtrain_sc = sc.fit_transform(xtrain_bow)
Xtest_sc = sc.transform(xtest_bow)
```

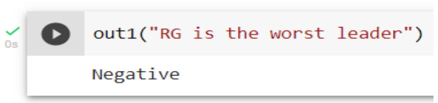
Driver Code:

```
def out1(txt):
    words = txt.split()
    table = str.maketrans('', '', string.punctuation)
    stripped = [w.translate(table) for w in words]
    str1 = " "
    tweet = (str1.join(stripped))
    analysis = TextBlob(tweet)
    if analysis.sentiment.polarity > 0:
        print("Positive")
    elif analysis.sentiment.polarity==0 :
        print("Neutral")
    elif analysis.sentiment.polarity<0:
        print("Negative")
```

Outputs :



```
out1("XYZ is a good person")
Positive
```



```
out1("RG is the worst leader")
Negative
```

<u>Conclusion:</u>

<u>Deriving from Data Visualisation :</u>

- Uttar Pradesh has the maximum number of constituencies. This has remained the same throughout the years. West Bengal and Maharashtra also have a high number of constituencies.
- BJP tremendously won the elections in 2014 and 2019 with a big lead in the number of seats. They managed to increase their seats from 2014 to 2019.
- Around 60% of our leaders were Graduates in the years 2009 and 2014, however this has fallen in the year 2019 by 20%.
- The assets and liabilities of our leaders have risen in this 10 year span.
- In all the parties the average age is 45 to 65 and there is little deviation in every election.

<u>Deriving from Model Prediction :</u>

- We apply the KNN model to our data to predict whether a candidate wins the election, which is displayed by the target variable- WINNER.
- We make some modifications at every stage of the model building, which improves the accuracy of the model.
- Table below summarises the accuracy of the model at each stage.

Model version	**Model- I**	**Model- II**	**Model- III**	**Model- IV**	**Model- V**
Accuracy	**91.2%**	**93.62%**	**93.99%**	**97.18%**	**98.98%**

Deriving from Sentimental analysis :

- In our module project we have successfully analyzed the twitter data during the election season of Lok Sabha 2019.
- We successfully trained,tested our model with three different types of ML algorithms.
- We successfully predicted the values and analyzed our results using a confusion matrix.
- By using different models we were able to observe variations between the three used classifiers.
- Finally, we analyzed the statements given by the user and categorized them as positive,negative or neutral using the driver code.

References:

- https://towardsdatascience.com/feature-engineering-for-election-result-prediction-python-943589d89414
- https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning
- https://scikit-learn.org/stable/modules/preprocessing.html
- https://blog.hubspot.com/marketing/types-of-graphs-for-data-visualization
- 7 Types of Classification Algorithms (analyticsindiamag.com)
- Decision Tree vs. Random Forest - Which Algorithm Should you Use? (analyticsvidhya.com)
- https://www.tableau.com/learn/articles/data-visualization
- https://www.kaggle.com/themlphdstudent/lok-sabha-election-candidate-list-2004-to-2019
- https://www.kaggle.com/c/twitter-sentiment-analysis2