Subject – Machine Learning

Final Project

Indian National Election 2024 Prediction

Date: 04-27-2022

Name: Anantha Lakshmi Sathvik Tirukkovalluri

Email: Tirukkovalluria@duq.edu

Content

1. Introduction

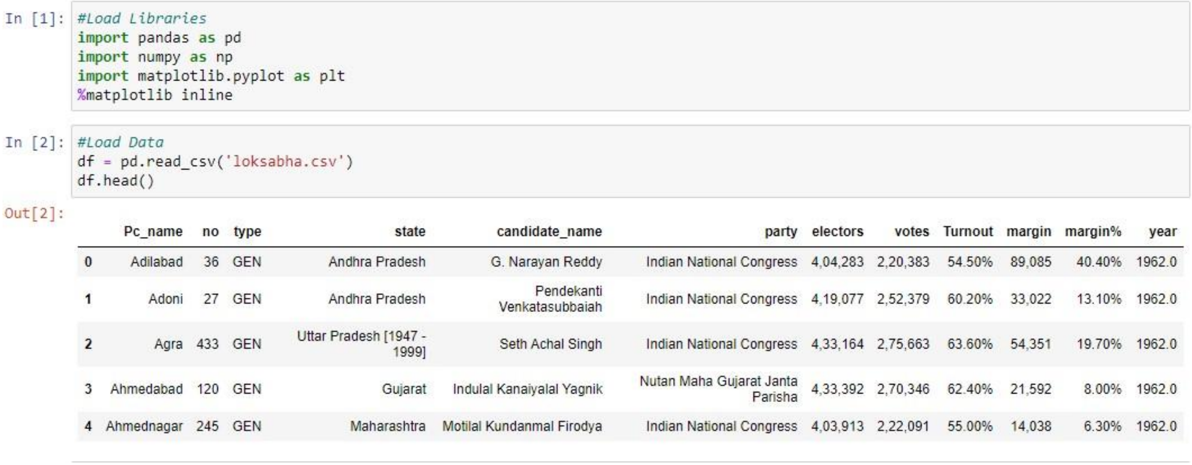
India being the world’s second-largest populated country it’s always been a piece of news to people that who is going to rule India. After getting independence in 1947 India has a parliamentary system as defined by its constitution, elections in India are conducted every 5 years with power distributed between the central government and the states. There are more than 500 parliament constituencies in India there were two majorly political parties Indian National Congress and Bharatiya Janata Party. The primary goal is to predict which party wins the 2024 election on a national level. The dataset consists of Indian elections data from 1962 to 2019. This has many attributes like the constituency, the party, state, year, electors, votes, turnout, margin, margin%.

2. Back Ground

Indian National Congress and Bharatiya Janata Party being the largest democratic parties it is always one of the evaluative ways that every citizen can influence government decision making through voting. But not everyone knows the importance of voting using this project we can study individual histories of both the parties, learn about the insights of all the parties involving in the elections and learn what is the ratio of wining and based on their history. What could be the prediction for 2024 elections.

3. Material & Methods

This project is performed by using the dataset based on the Lok Sabha 2019 in India. We use Jupyter Notebook to run the code, loading data set, importing libraries and using models.

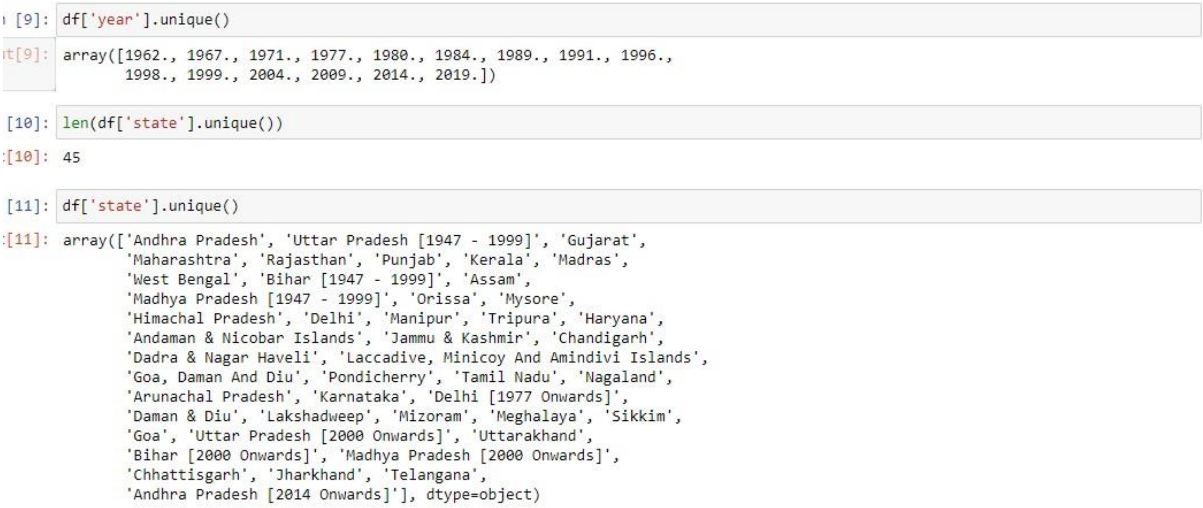


```
In [1]: #Load Libraries
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        %matplotlib inline
```

```
In [2]: #Load Data
        df = pd.read_csv('loksabha.csv')
        df.head()
```

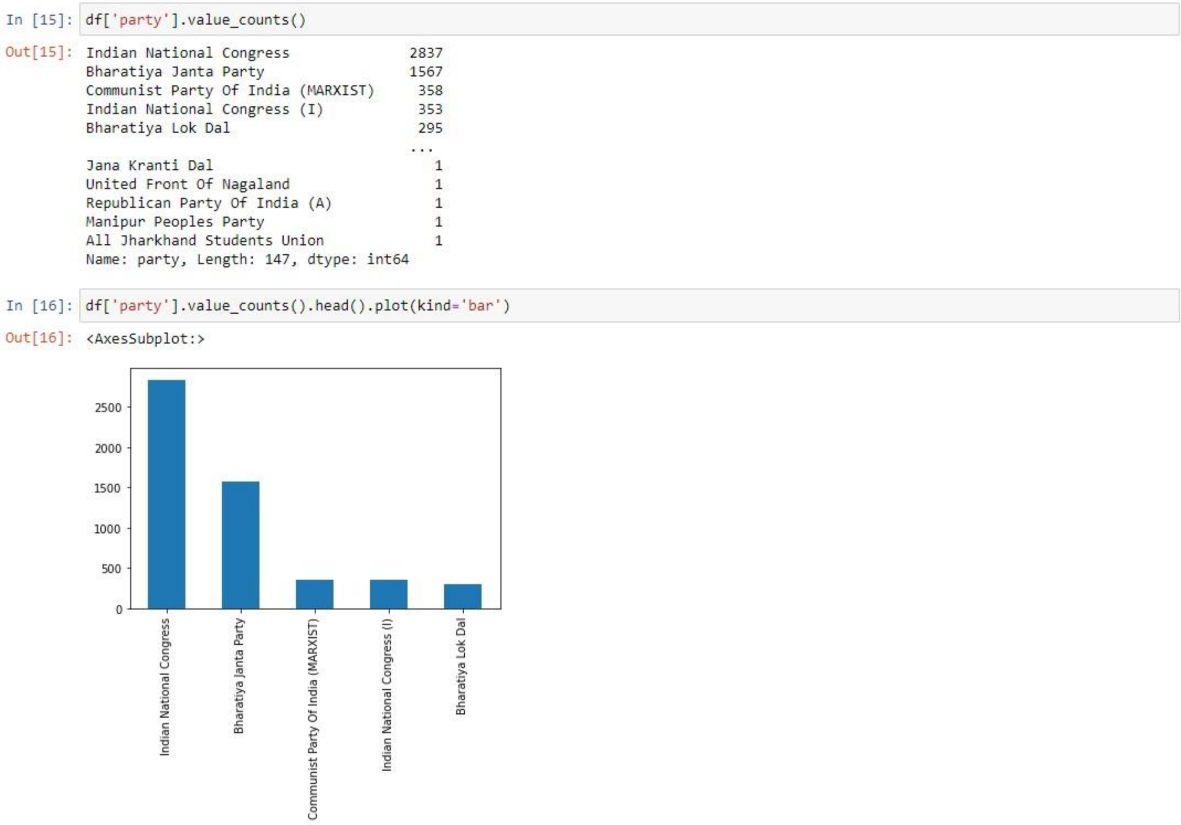
Out[2]:

	Pc_name	no	type	state	candidate_name	party	electors	votes	Turnout	margin	margin%	year
0	Adilabad	36	GEN	Andhra Pradesh	G. Narayan Reddy	Indian National Congress	4,04,283	2,20,383	54.50%	89,085	40.40%	1962.0
1	Adoni	27	GEN	Andhra Pradesh	Pendekanti Venkatasubbaiah	Indian National Congress	4,19,077	2,52,379	60.20%	33,022	13.10%	1962.0
2	Agra	433	GEN	Uttar Pradesh [1947 - 1999]	Seth Achal Singh	Indian National Congress	4,33,164	2,75,663	63.60%	54,351	19.70%	1962.0
3	Ahmedabad	120	GEN	Gujarat	Indulal Kanaiyalal Yagnik	Nutan Maha Gujarat Janta Parisha	4,33,392	2,70,346	62.40%	21,592	8.00%	1962.0
4	Ahmednagar	245	GEN	Maharashtra	Motilal Kundanmal Firodya	Indian National Congress	4,03,913	2,22,091	55.00%	14,038	6.30%	1962.0



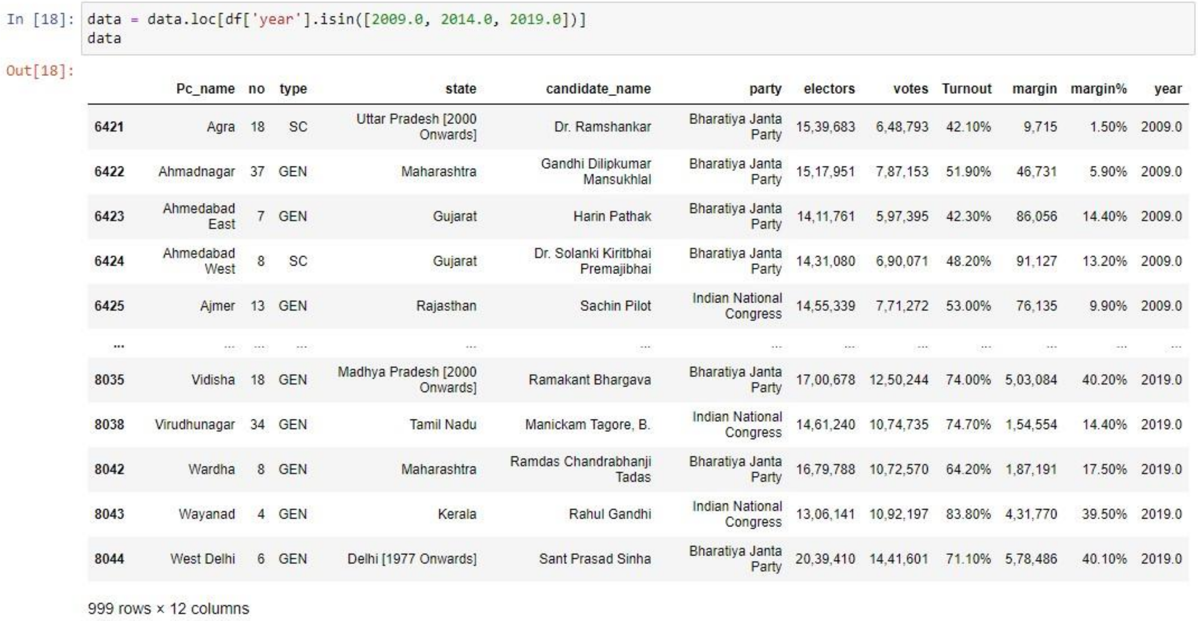
```
ı [9]: df['year'].unique()

ıt[9]: array([1962., 1967., 1971., 1977., 1980., 1984., 1989., 1991., 1996.,
              1998., 1999., 2004., 2009., 2014., 2019.])

 [10]: len(df['state'].unique())

:[10]: 45

 [11]: df['state'].unique()

:[11]: array(['Andhra Pradesh', 'Uttar Pradesh [1947 - 1999]', 'Gujarat',
              'Maharashtra', 'Rajasthan', 'Punjab', 'Kerala', 'Madras',
              'West Bengal', 'Bihar [1947 - 1999]', 'Assam',
              'Madhya Pradesh [1947 - 1999]', 'Orissa', 'Mysore',
              'Himachal Pradesh', 'Delhi', 'Manipur', 'Tripura', 'Haryana',
              'Andaman & Nicobar Islands', 'Jammu & Kashmir', 'Chandigarh',
              'Dadra & Nagar Haveli', 'Laccadive, Minicoy And Amindivi Islands',
              'Goa, Daman And Diu', 'Pondicherry', 'Tamil Nadu', 'Nagaland',
              'Arunachal Pradesh', 'Karnataka', 'Delhi [1977 Onwards]',
              'Daman & Diu', 'Lakshadweep', 'Mizoram', 'Meghalaya', 'Sikkim',
              'Goa', 'Uttar Pradesh [2000 Onwards]', 'Uttarakhand',
              'Bihar [2000 Onwards]', 'Madhya Pradesh [2000 Onwards]',
              'Chhattisgarh', 'Jharkhand', 'Telangana',
              'Andhra Pradesh [2014 Onwards]'], dtype=object)
```

The year is ranging from 1962 to 2019 and there are 45 states in India.



```
In [15]: df['party'].value_counts()

Out[15]: Indian National Congress             2837
         Bharatiya Janta Party                1567
         Communist Party Of India (MARXIST)    358
         Indian National Congress (I)          353
         Bharatiya Lok Dal                     295
                                              ... 
         Jana Kranti Dal                         1
         United Front Of Nagaland                1
         Republican Party Of India (A)           1
         Manipur Peoples Party                   1
         All Jharkhand Students Union            1
         Name: party, Length: 147, dtype: int64

In [16]: df['party'].value_counts().head().plot(kind='bar')

Out[16]: <AxesSubplot:>
```

As we can clearly see that Indian National Congress and Bharatiya Janata Party are most dominating parties from the history of India. So by considering these two parties and their past three election years we will perform our classification and prediction.



In [18]:

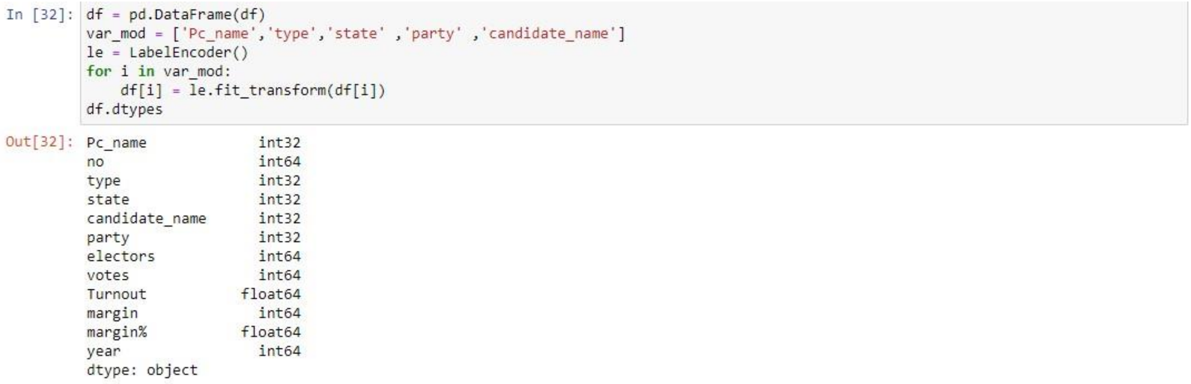
```
data = data.loc[df['year'].isin([2009.0, 2014.0, 2019.0])]
data
```

Out[18]:

	Pc_name	no	type	state	candidate_name	party	electors	votes	Turnout	margin	margin%	year
6421	Agra	18	SC	Uttar Pradesh [2000 Onwards]	Dr. Ramshankar	Bharatiya Janta Party	15,39,683	6,48,793	42.10%	9,715	1.50%	2009.0
6422	Ahmadnagar	37	GEN	Maharashtra	Gandhi Dilipkumar Mansukhlal	Bharatiya Janta Party	15,17,951	7,87,153	51.90%	46,731	5.90%	2009.0
6423	Ahmedabad East	7	GEN	Gujarat	Harin Pathak	Bharatiya Janta Party	14,11,761	5,97,395	42.30%	86,056	14.40%	2009.0
6424	Ahmedabad West	8	SC	Gujarat	Dr. Solanki Kiritbhai Premajibhai	Bharatiya Janta Party	14,31,080	6,90,071	48.20%	91,127	13.20%	2009.0
6425	Ajmer	13	GEN	Rajasthan	Sachin Pilot	Indian National Congress	14,55,339	7,71,272	53.00%	76,135	9.90%	2009.0
...	...	...	...	...	...	...	...	...	...	...	...	...
8035	Vidisha	18	GEN	Madhya Pradesh [2000 Onwards]	Ramakant Bhargava	Bharatiya Janta Party	17,00,678	12,50,244	74.00%	5,03,084	40.20%	2019.0
8038	Virudhunagar	34	GEN	Tamil Nadu	Manickam Tagore, B.	Indian National Congress	14,61,240	10,74,735	74.70%	1,54,554	14.40%	2019.0
8042	Wardha	8	GEN	Maharashtra	Ramdas Chandrabhanji Tadas	Bharatiya Janta Party	16,79,788	10,72,570	64.20%	1,87,191	17.50%	2019.0
8043	Wayanad	4	GEN	Kerala	Rahul Gandhi	Indian National Congress	13,06,141	10,92,197	83.80%	4,31,770	39.50%	2019.0
8044	West Delhi	6	GEN	Delhi [1977 Onwards]	Sant Prasad Sinha	Bharatiya Janta Party	20,39,410	14,41,601	71.10%	5,78,486	40.10%	2019.0

999 rows × 12 columns

By considering the years 2009 to 2019 for INC and BJP we have 999rows and 12 columns.

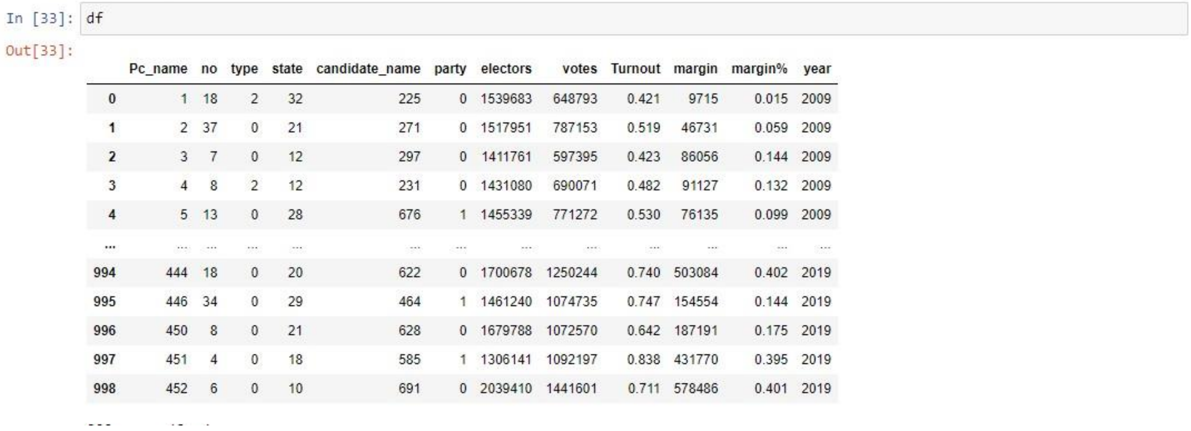


In [32]:

```
df = pd.DataFrame(df)
var_mod = ['Pc_name','type','state' ,'party' ,'candidate_name']
le = LabelEncoder()
for i in var_mod:
    df[i] = le.fit_transform(df[i])
df.dtypes
```

Out[32]:

```
Pc_name             int32
no                  int64
type                int32
state               int32
candidate_name      int32
party               int32
electors            int64
votes               int64
Turnout           float64
margin              int64
margin%           float64
year                int64
dtype: object
```



In [33]:

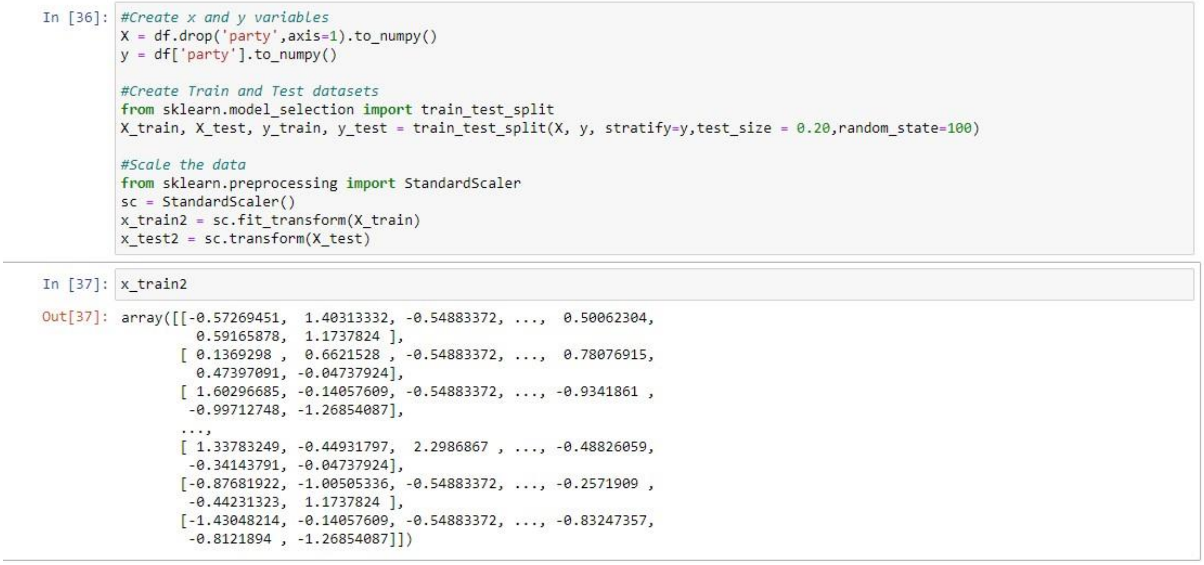
```
df
```

Out[33]:

	Pc_name	no	type	state	candidate_name	party	electors	votes	Turnout	margin	margin%	year
0	1	18	2	32	225	0	1539683	648793	0.421	9715	0.015	2009
1	2	37	0	21	271	0	1517951	787153	0.519	46731	0.059	2009
2	3	7	0	12	297	0	1411761	597395	0.423	86056	0.144	2009
3	4	8	2	12	231	0	1431080	690071	0.482	91127	0.132	2009
4	5	13	0	28	676	1	1455339	771272	0.530	76135	0.099	2009
...	...	...	...	...	...	...	...	...	...	...	...	...
994	444	18	0	20	622	0	1700678	1250244	0.740	503084	0.402	2019
995	446	34	0	29	464	1	1461240	1074735	0.747	154554	0.144	2019
996	450	8	0	21	628	0	1679788	1072570	0.642	187191	0.175	2019
997	451	4	0	18	585	1	1306141	1092197	0.838	431770	0.395	2019
998	452	6	0	10	691	0	2039410	1441601	0.711	578486	0.401	2019

As the data are not in machine readable form, we use LabelEncoder to convert the values into Integers so that our machine can understand.

Where in party variable “0” represents Bharatiya Janata party and “1” represents Indian National Congress.

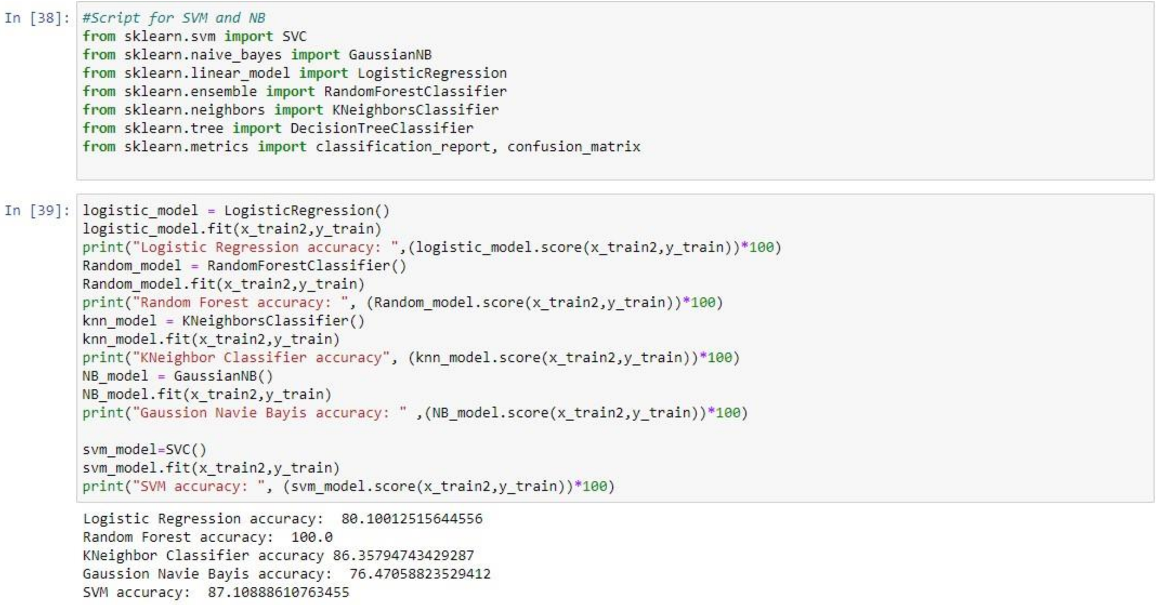


```
In [36]: #Create x and y variables
         X = df.drop('party',axis=1).to_numpy()
         y = df['party'].to_numpy()

         #Create Train and Test datasets
         from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y,test_size = 0.20,random_state=100)

         #Scale the data
         from sklearn.preprocessing import StandardScaler
         sc = StandardScaler()
         x_train2 = sc.fit_transform(X_train)
         x_test2 = sc.transform(X_test)

In [37]: x_train2

Out[37]: array([[-0.57269451,  1.40313332, -0.54883372, ...,  0.50062304,
                  0.59165878,  1.1737824 ],
                [ 0.1369298 ,  0.6621528 , -0.54883372, ...,  0.78076915,
                  0.47397091, -0.04737924],
                [ 1.60296685, -0.14057609, -0.54883372, ..., -0.9341861 ,
                 -0.99712748, -1.26854087],
                ...,
                [ 1.33783249, -0.44931797,  2.2986867 , ..., -0.48826059,
                 -0.34143791, -0.04737924],
                [-0.87681922, -1.00505336, -0.54883372, ..., -0.2571909 ,
                 -0.44231323,  1.1737824 ],
                [-1.43048214, -0.14057609, -0.54883372, ..., -0.83247357,
                 -0.8121894 , -1.26854087]])
```

Now we create two variables x and y where in “x” has the dependent variables and “y” has the independent variable.

We spilt the data into train case and test case taking the train size = 0.8 and test size =0.2.

And to standardize the range functionality of input dataset we use Standard Scalar.



```
In [38]: #Script for SVM and NB
         from sklearn.svm import SVC
         from sklearn.naive_bayes import GaussianNB
         from sklearn.linear_model import LogisticRegression
         from sklearn.ensemble import RandomForestClassifier
         from sklearn.neighbors import KNeighborsClassifier
         from sklearn.tree import DecisionTreeClassifier
         from sklearn.metrics import classification_report, confusion_matrix

In [39]: logistic_model = LogisticRegression()
         logistic_model.fit(x_train2,y_train)
         print("Logistic Regression accuracy: ",(logistic_model.score(x_train2,y_train))*100)
         Random_model = RandomForestClassifier()
         Random_model.fit(x_train2,y_train)
         print("Random Forest accuracy: ", (Random_model.score(x_train2,y_train))*100)
         knn_model = KNeighborsClassifier()
         knn_model.fit(x_train2,y_train)
         print("KNeighbor Classifier accuracy", (knn_model.score(x_train2,y_train))*100)
         NB_model = GaussianNB()
         NB_model.fit(x_train2,y_train)
         print("Gaussion Navie Bayis accuracy: " ,(NB_model.score(x_train2,y_train))*100)

         svm_model=SVC()
         svm_model.fit(x_train2,y_train)
         print("SVM accuracy: ", (svm_model.score(x_train2,y_train))*100)

         Logistic Regression accuracy:  80.10012515644556
         Random Forest accuracy:  100.0
         KNeighbor Classifier accuracy 86.35794743429287
         Gaussion Navie Bayis accuracy:  76.47058823529412
         SVM accuracy:  87.10888610763455
```

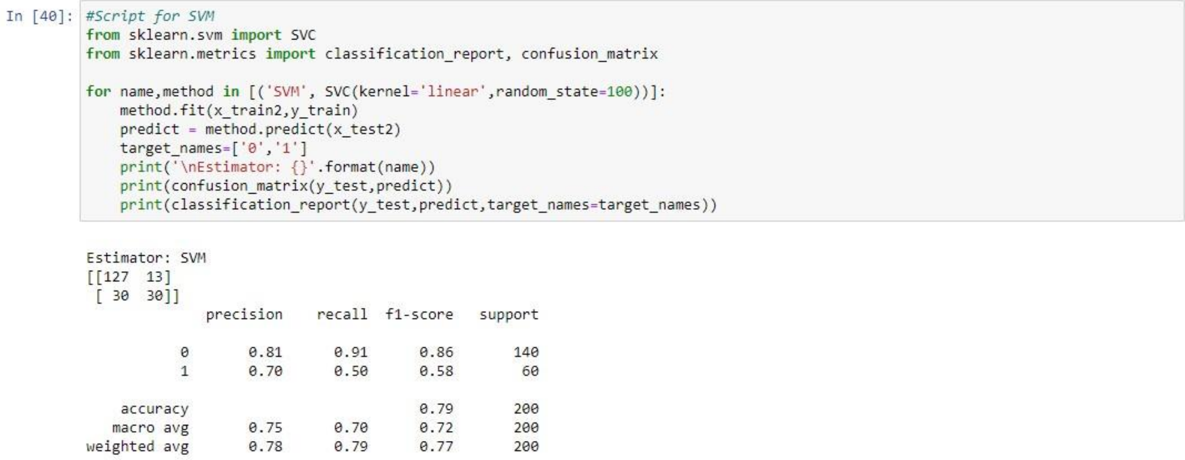
<u>Logistic Regression</u>: is supervised machine learning classification used for predictive analysis. We use this to understand the relationship between independent variables and dependent variables. It is primarily used when the independent variable is categorical. In our case it is “0”(Bharatiya Janata Party)and “1” (Indian National Congress). It is used to predict the likelihood of specific outcome. Logistic Regression utilizes a sophisticated cost function, which is known as the “Sigmoid function” or “logistic function”

Random Forest is a supervised machine learning algorithm used for classification models. It gives better predictive performance based on the decision tress. It predicts by considering the mean or average of the output.

KNN classifier is a supervised machine learning model used to classification problems. Due to its simplicity, it is widely used classification algorithm. Where new data point is classified based on similarity in the specific group of neighboring data points. It trains the data point to understand what is a PEN (for an example) how pen looks like by giving the training set like size of the pen, color of the pen etc.

Navie bayes classification can be used because it can handle categorical variables and can have many variables where the results does not get effected. Naïve bayes learns the data fast the more the data the more the accuracy.

Support vector machine (SVM) is a supervised machine learning model that uses classification algorithms for two-group classification problems. After giving an SVM model sets of labeled training data for each category, they’re able to categorize new input.

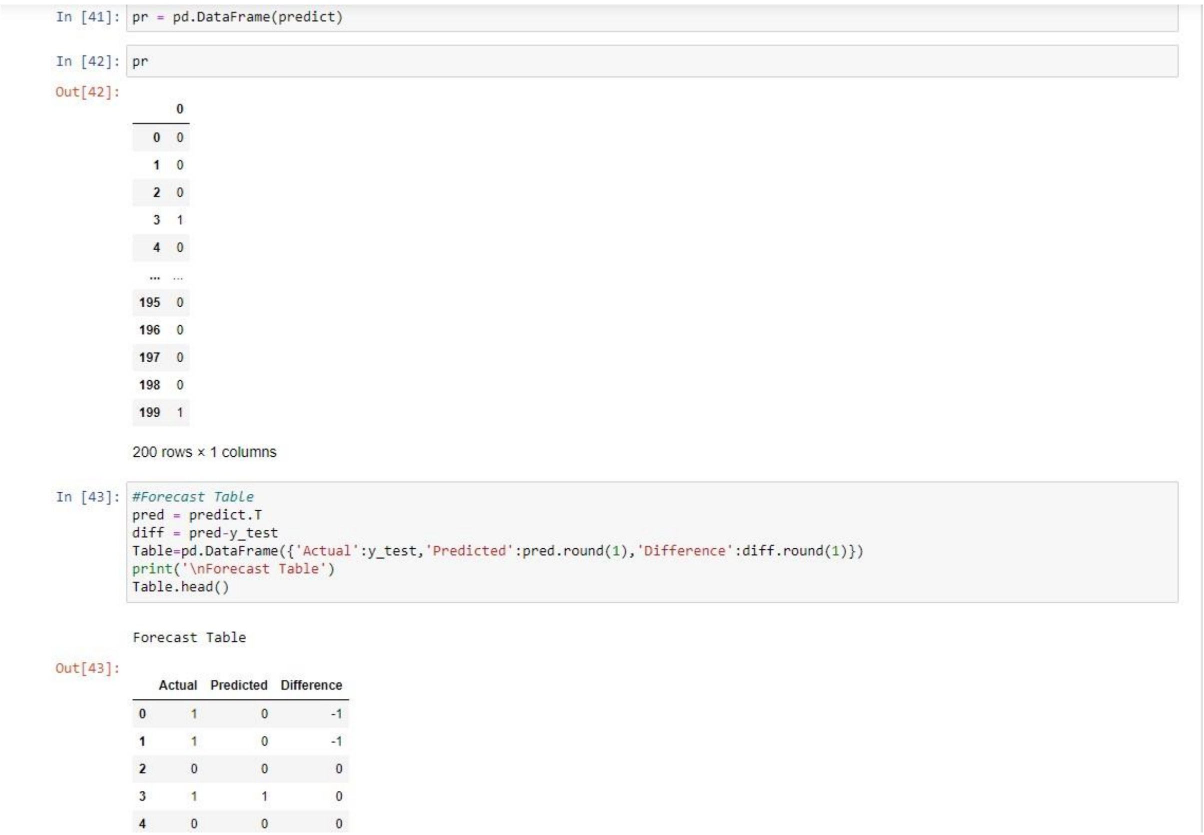


```
In [40]: #Script for SVM
         from sklearn.svm import SVC
         from sklearn.metrics import classification_report, confusion_matrix

         for name,method in [('SVM', SVC(kernel='linear',random_state=100))]:
             method.fit(x_train2,y_train)
             predict = method.predict(x_test2)
             target_names=['0','1']
             print('\nEstimator: {}'.format(name))
             print(confusion_matrix(y_test,predict))
             print(classification_report(y_test,predict,target_names=target_names))
```

```
Estimator: SVM
[[127  13]
 [ 30  30]]
              precision    recall  f1-score   support

           0       0.81      0.91      0.86       140
           1       0.70      0.50      0.58        60

    accuracy                           0.79       200
   macro avg       0.75      0.70      0.72       200
weighted avg       0.78      0.79      0.77       200
```

SVM model is selected for the final prediction. The script elaborates on the metrics obtained when SVM model is used on test data. It shows the entire classification report and a confusion matrix.

We can observe that the accuracy of the model on test data is 79% with 127 “0”(BJP) observations predicted correctly out of 140 and 30 “1”(INC) observations predicted correctly out 60.



```
In [41]: pr = pd.DataFrame(predict)
```

```
In [42]: pr
```

Out[42]:

	0
0	0
1	0
2	0
3	1
4	0
...	...
195	0
196	0
197	0
198	0
199	1

200 rows × 1 columns

```
In [43]: #Forecast Table
         pred = predict.T
         diff = pred-y_test
         Table=pd.DataFrame({'Actual':y_test,'Predicted':pred.round(1),'Difference':diff.round(1)})
         print('\nForecast Table')
         Table.head()
```

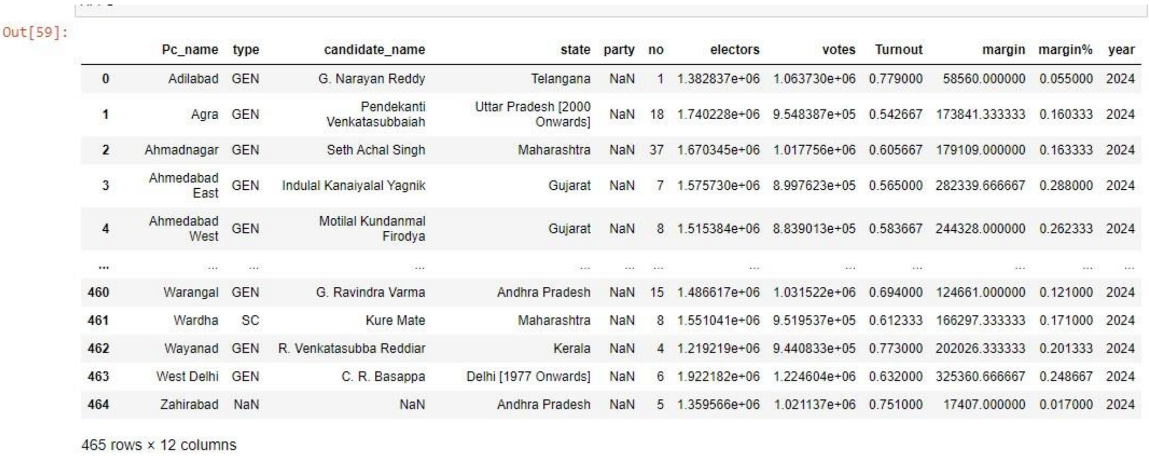
```
Forecast Table
```

Out[43]:

	Actual	Predicted	Difference
0	1	0	-1
1	1	0	-1
2	0	0	0
3	1	1	0
4	0	0	0

The above forecast table shows the actual value and predicted value by the model.

4. Results

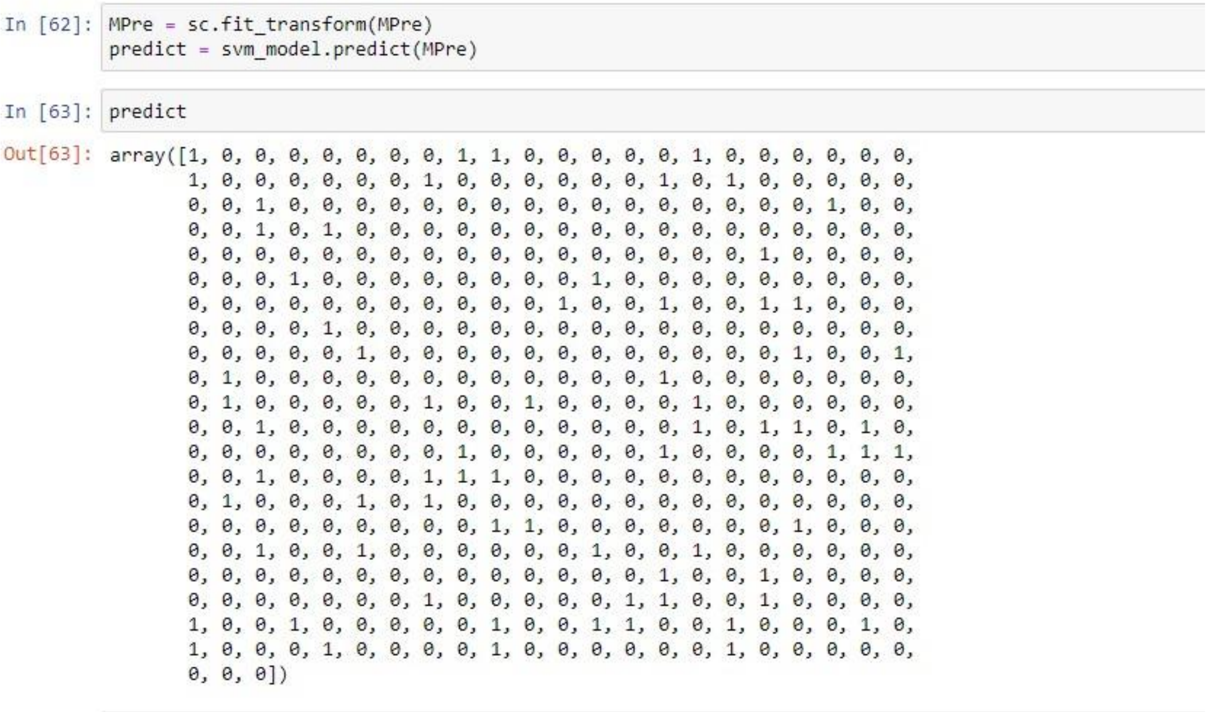


Out[59]:

	Pc_name	type	candidate_name	state	party	no	electors	votes	Turnout	margin	margin%	year
0	Adilabad	GEN	G. Narayan Reddy	Telangana	NaN	1	1.382837e+06	1.063730e+06	0.779000	58560.000000	0.055000	2024
1	Agra	GEN	Pendekanti Venkatasubbaiah	Uttar Pradesh [2000 Onwards]	NaN	18	1.740228e+06	9.548387e+05	0.542667	173841.333333	0.160333	2024
2	Ahmadnagar	GEN	Seth Achal Singh	Maharashtra	NaN	37	1.670345e+06	1.017756e+06	0.605667	179109.000000	0.163333	2024
3	Ahmedabad East	GEN	Indulal Kanaiyalal Yagnik	Gujarat	NaN	7	1.575730e+06	8.997623e+05	0.565000	282339.666667	0.288000	2024
4	Ahmedabad West	GEN	Motilal Kundanmal Firodya	Gujarat	NaN	8	1.515384e+06	8.839013e+05	0.583667	244328.000000	0.262333	2024
...	...	...	...	...	...	...	...	...	...	...	...	...
460	Warangal	GEN	G. Ravindra Varma	Andhra Pradesh	NaN	15	1.486617e+06	1.031522e+06	0.694000	124661.000000	0.121000	2024
461	Wardha	SC	Kure Mate	Maharashtra	NaN	8	1.551041e+06	9.519537e+05	0.612333	166297.333333	0.171000	2024
462	Wayanad	GEN	R. Venkatasubba Reddiar	Kerala	NaN	4	1.219219e+06	9.440833e+05	0.773000	202026.333333	0.201333	2024
463	West Delhi	GEN	C. R. Basappa	Delhi [1977 Onwards]	NaN	6	1.922182e+06	1.224604e+06	0.632000	325360.666667	0.248667	2024
464	Zahirabad	NaN	NaN	Andhra Pradesh	NaN	5	1.359566e+06	1.021137e+06	0.751000	17407.000000	0.017000	2024

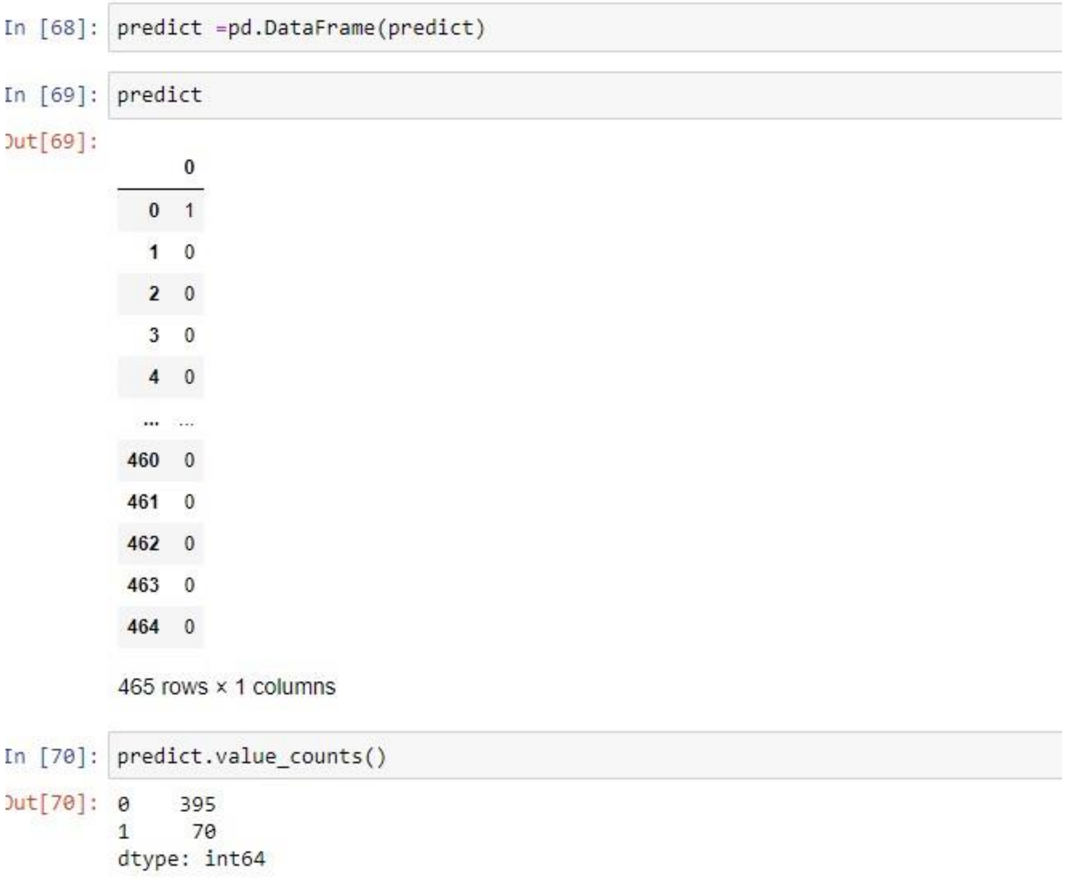
465 rows × 12 columns

So now to predict main output that which party is going win 2024 election. By taking the average of data of constituencies in previous three elections and SVM model for prediction we get the results as below.

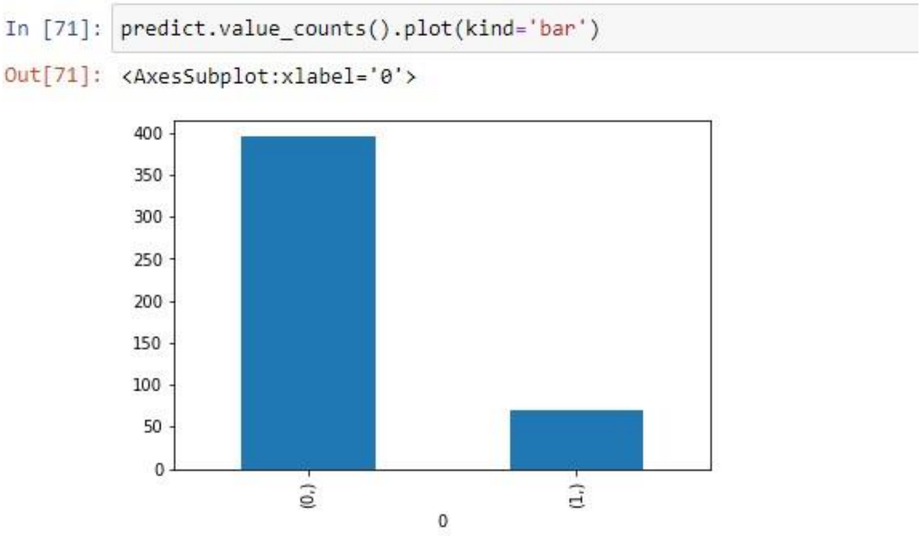


```
In [62]: MPre = sc.fit_transform(MPre)
         predict = svm_model.predict(MPre)

In [63]: predict

Out[63]: array([1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
                1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0,
                0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
                0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
                0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0,
                0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
                0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
                0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
                0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1,
                0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
                0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0,
                1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0,
                1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
                0, 0, 0])
```

We used the predict function on SVM model and predicted the above values for the variable party.



```
In [68]: predict =pd.DataFrame(predict)

In [69]: predict

Out[69]:
               0
           0   1
           1   0
           2   0
           3   0
           4   0
         ...  ...
         460   0
         461   0
         462   0
         463   0
         464   0

         465 rows × 1 columns

In [70]: predict.value_counts()

Out[70]: 0    395
         1     70
         dtype: int64
```



```
In [71]: predict.value_counts().plot(kind='bar')
Out[71]: <AxesSubplot:xlabel='0'>
```

We have made prediction where the results of our prediction are “0”(BJP) = 395 and “1”(INC) = 70. We can say by this bharatiya Janata party has more chances of winning in 2024.

5. Discussion

- Since, Random forest model gave the accuracy of 100%. It might be due to overfitting. Thus being a negative impact on scope.
- During the years 1979 and 1980, the party Indian National Congress had formed different sub parties INC(I) and INC(U) but later they dissolved. However this does not effect our data analysis as our model was trained and tested on data between years 2009 to 2019.

6. Conclusion

Using this prediction model we have predicted that Bharatiya Janata Party has a high probability of winning the next election 2024. Where Indian National Congress has a low probability of winning the election 2024.

References:

https://www.ibm.com/topics/logistic-regression

https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

https://www.geeksforgeeks.org/naive-bayes-classifiers/

https://www.analyticsvidhya.com/blog/2021/10/building-an-end-to-end-logistic-regression-model/

https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/

https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/

https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/