Project Report

GitHub URL
https://github.com/nmparikh/UCDPA_NimeshParikh

Abstract
This project is about predicting winner in Indian General Election results. Last general elections were held in 2019 and next one is schedule in 2024. I want to find out some key insights and prepare a model to predict the outcome of next election.

You can find complete code in GitHub URL >> Data_Cleaning_Analysis.ipynb notebook.

Introduction
Since it's less than 2 years now for next general election in India, Political parties wants to understand what happened in 2019 and prepare for the next elections. Data insights will help them understand the likes and dislikes of voters so that parties can offer candidates accordingly. Key python libraries used are numpy, pandas, matplotlib, seaborn, re, plotly and sklearn.

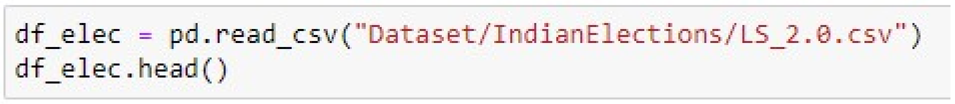
We have seen in past that data-based decisions have helped major country's leaders to win respective elections. This is now new normal.

Dataset
Colleting ground level data is difficult and I used dataset from Kaggle as this was free to use and have good parameters.
(https://www.kaggle.com/datasets/prakrutchauhan/indian-candidates-for-general-election-2019)

Implementation Process
I used Jupyter notebook with python as core language for the analysis. Python has rich set of libraries which make this task easy. You can find all code with output in Git Repo (Link is already provided in first section)

There are different phases of Applications which has been performed as below:

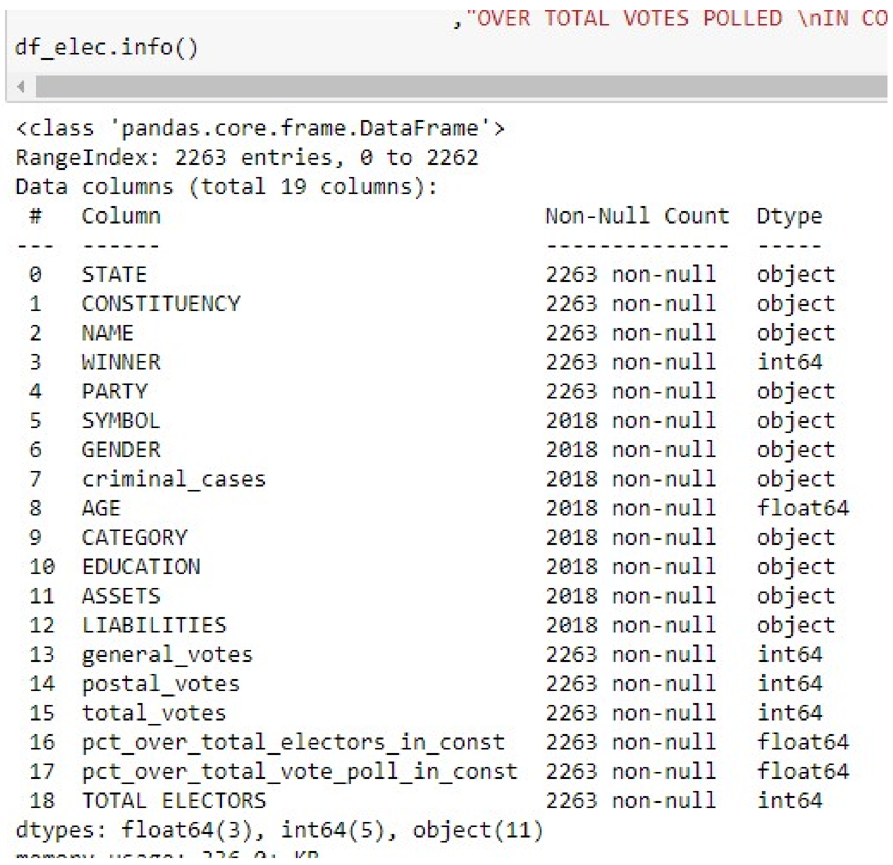
1. **Data On-boarding**
   For On-boarding of data, I have downloaded it from Kaggle as CSV and start reading that data from csv file with below code.

   ```
   df_elec = pd.read_csv("Dataset/IndianElections/LS_2.0.csv")
   df_elec.head()
   ```

   Since this is multi column relation structure, Data is uploaded as Python data frame. Data frame is much suitable over List or Dict. as it has multiple features. Currently we do not have any API based source but if we do, we can download data using Rest API as well. Code is very simple and easy for the same.

2. **Data Understanding**
   Fundamental part of any analysis is to understand your data. What each column means, is all required information available for your analysis, what is the grain of the data and so forth...



```
                              ,"OVER TOTAL VOTES POLLED \nIN CO
df_elec.info()
```

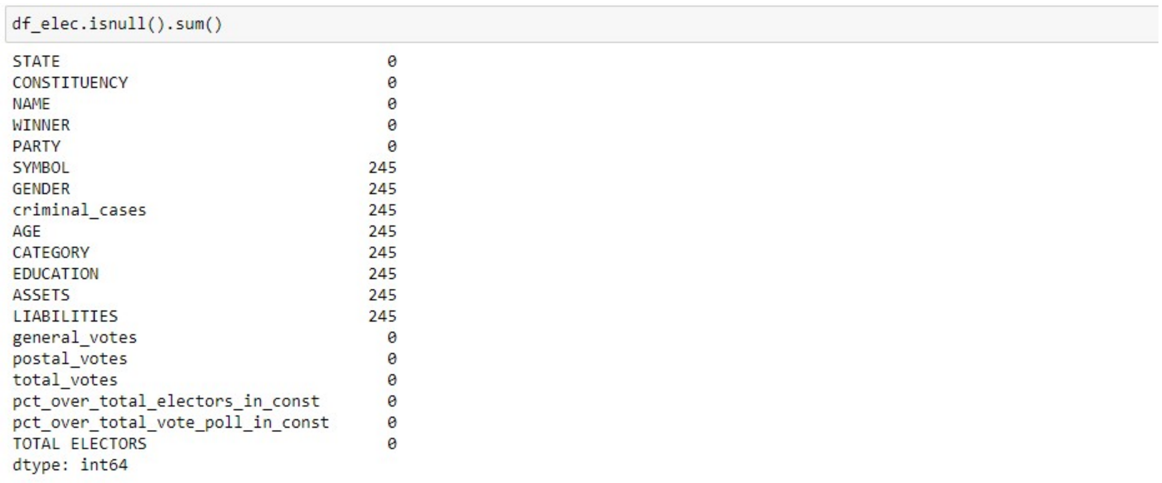
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2263 entries, 0 to 2262
Data columns (total 19 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   STATE                            2263 non-null   object
 1   CONSTITUENCY                     2263 non-null   object
 2   NAME                             2263 non-null   object
 3   WINNER                           2263 non-null   int64
 4   PARTY                            2263 non-null   object
 5   SYMBOL                           2018 non-null   object
 6   GENDER                           2018 non-null   object
 7   criminal_cases                   2018 non-null   object
 8   AGE                              2018 non-null   float64
 9   CATEGORY                         2018 non-null   object
 10  EDUCATION                        2018 non-null   object
 11  ASSETS                           2018 non-null   object
 12  LIABILITIES                      2018 non-null   object
 13  general_votes                    2263 non-null   int64
 14  postal_votes                     2263 non-null   int64
 15  total_votes                      2263 non-null   int64
 16  pct_over_total_electors_in_const 2263 non-null   float64
 17  pct_over_total_vote_poll_in_const 2263 non-null  float64
 18  TOTAL ELECTORS                   2263 non-null   int64
dtypes: float64(3), int64(5), object(11)
```

This shows we have 19 columns and 2263 rows. Some of the columns are defined as Int meaning they are our numeric data and rest are text/categorial data.

This data is at the grain of State+Constituency+Party+Name. Which mean you should have unique data rows based on this key. This dataset also gives you important factors like gender, cast, education, asset, cases and age for each candidate.

3. Data Quality Issues, Fix & Data Analysis

After understanding Data, Next item is to look at if we have any DQ Issues and if so, fix them or avoid those items from analysis.

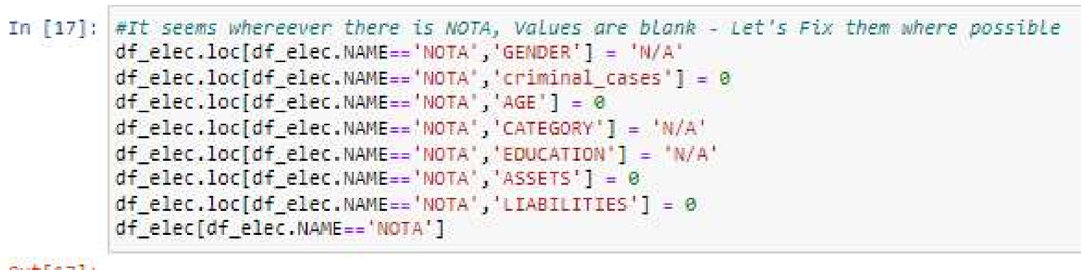


```
df_elec.isnull().sum()
```

```
STATE                                 0
CONSTITUENCY                          0
NAME                                  0
WINNER                                0
PARTY                                 0
SYMBOL                              245
GENDER                              245
criminal_cases                      245
AGE                                 245
CATEGORY                            245
EDUCATION                           245
ASSETS                              245
LIABILITIES                         245
general_votes                         0
postal_votes                          0
total_votes                           0
pct_over_total_electors_in_const      0
pct_over_total_vote_poll_in_const     0
TOTAL ELECTORS                        0
dtype: int64
```

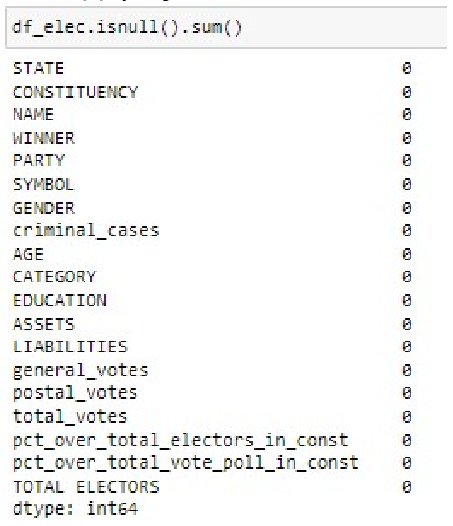
==> After Looking at columns, we can see there about 245 rows has NULL values in 8 columns. We will have to understand if there are any pattern

To start with, Let’s see where we have NULLs and try to understand why they are appearing and see what we can do to fix them. By looking further, I found that this is happening for all NOTA entries. NOTA is what voter can select when they do not like any candidate. We have fix them by applying default value. Alternatively, you can use dropna() method to omit the rows or you can use backfill or forwardfill techniques for populating respective values in continuous variable e.g., Temperature of the day.



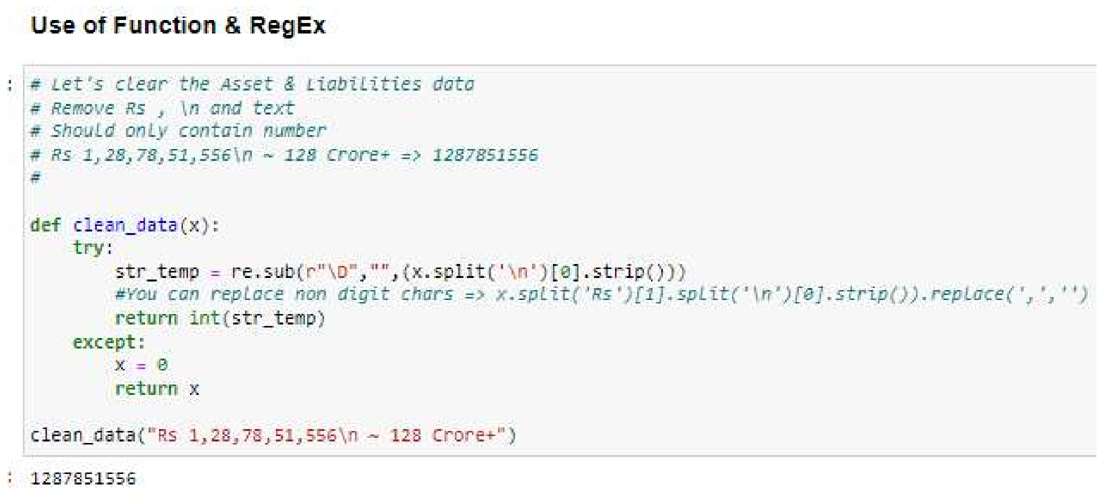
```
In [17]: #It seems whereever there is NOTA, Values are blank - Let's Fix them where possible
         df_elec.loc[df_elec.NAME=='NOTA','GENDER'] = 'N/A'
         df_elec.loc[df_elec.NAME=='NOTA','criminal_cases'] = 0
         df_elec.loc[df_elec.NAME=='NOTA','AGE'] = 0
         df_elec.loc[df_elec.NAME=='NOTA','CATEGORY'] = 'N/A'
         df_elec.loc[df_elec.NAME=='NOTA','EDUCATION'] = 'N/A'
         df_elec.loc[df_elec.NAME=='NOTA','ASSETS'] = 0
         df_elec.loc[df_elec.NAME=='NOTA','LIABILITIES'] = 0
         df_elec[df_elec.NAME=='NOTA']
```

After applying Default values, we can see there are no NULL



```
df_elec.isnull().sum()

STATE                              0
CONSTITUENCY                       0
NAME                               0
WINNER                             0
PARTY                              0
SYMBOL                             0
GENDER                             0
criminal_cases                     0
AGE                                0
CATEGORY                           0
EDUCATION                          0
ASSETS                             0
LIABILITIES                        0
general_votes                      0
postal_votes                       0
total_votes                        0
pct_over_total_electors_in_const   0
pct_over_total_vote_poll_in_const  0
TOTAL ELECTORS                     0
dtype: int64
```

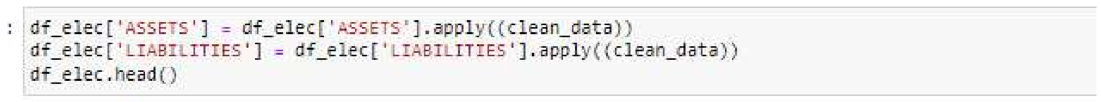
Now I want to clear Asset and Liabilities column values. I want to make sure we have only amount value and no other information. I have created Function and used Regular Expression to make sure we extract amount from current value and remove unwanted characters. I have achieved this with below code



Use of Function & RegEx

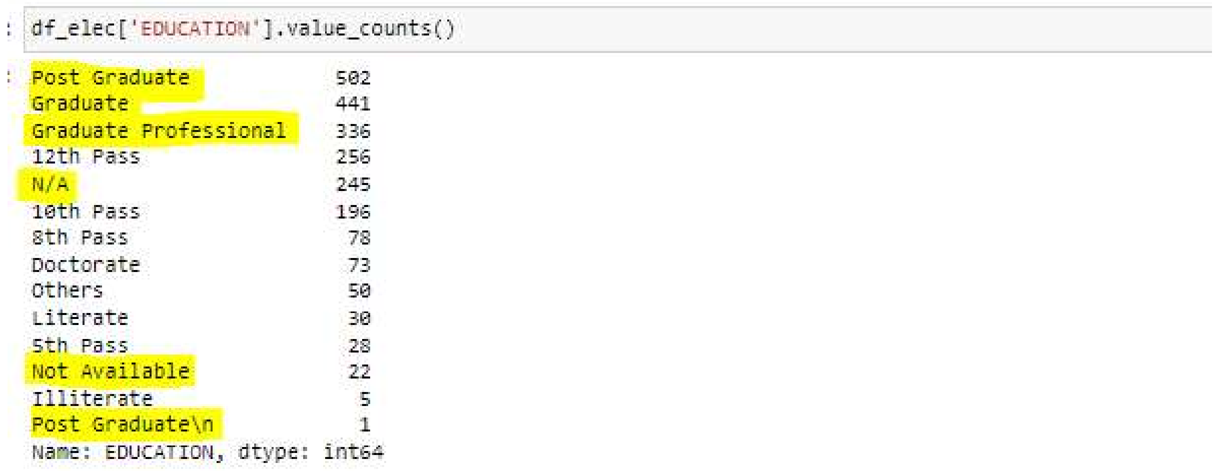
```
# Let's clear the Asset & Liabilities data
# Remove Rs , \n and text
# Should only contain number
# Rs 1,28,78,51,556\n ~ 128 Crore+ => 1287851556
#

def clean_data(x):
    try:
        str_temp = re.sub(r"\D","",(x.split('\n')[0].strip()))
        #You can replace non digit chars => x.split('Rs')[1].split('\n')[0].strip()).replace(',','')
        return int(str_temp)
    except:
        x = 0
        return x

clean_data("Rs 1,28,78,51,556\n ~ 128 Crore+")
```

```
1287851556
```



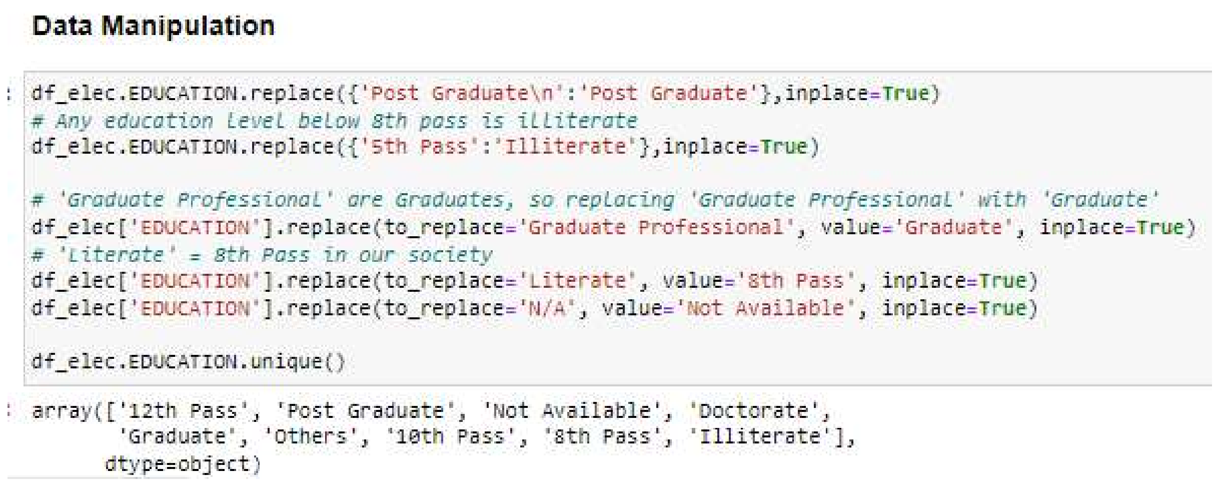
```
df_elec['ASSETS'] = df_elec['ASSETS'].apply((clean_data))
df_elec['LIABILITIES'] = df_elec['LIABILITIES'].apply((clean_data))
df_elec.head()
```

The other DQ issues are like there are Inconsistent values for Education.



```
df_elec['EDUCATION'].value_counts()
```

```
Post Graduate            502
Graduate                 441
Graduate Professional    336
12th Pass                256
N/A                      245
10th Pass                196
8th Pass                  78
Doctorate                 73
Others                    50
Literate                  30
5th Pass                  28
Not Available             22
Illiterate                 5
Post Graduate\n            1
Name: EDUCATION, dtype: int64
```



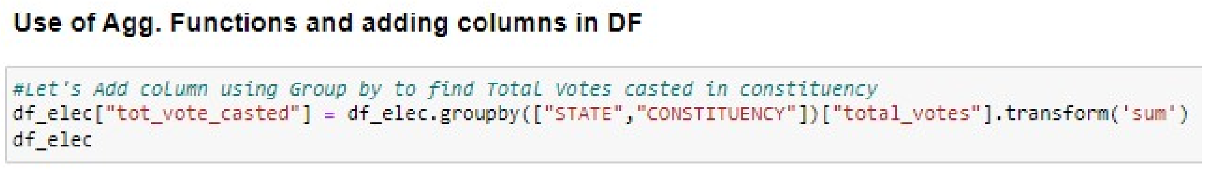
Data Manipulation

```
df_elec.EDUCATION.replace({'Post Graduate\n':'Post Graduate'},inplace=True)
# Any education level below 8th pass is illiterate
df_elec.EDUCATION.replace({'5th Pass':'Illiterate'},inplace=True)

# 'Graduate Professional' are Graduates, so replacing 'Graduate Professional' with 'Graduate'
df_elec['EDUCATION'].replace(to_replace='Graduate Professional', value='Graduate', inplace=True)
# 'Literate' = 8th Pass in our society
df_elec['EDUCATION'].replace(to_replace='Literate', value='8th Pass', inplace=True)
df_elec['EDUCATION'].replace(to_replace='N/A', value='Not Available', inplace=True)

df_elec.EDUCATION.unique()
```

```
array(['12th Pass', 'Post Graduate', 'Not Available', 'Doctorate',
       'Graduate', 'Others', '10th Pass', '8th Pass', 'Illiterate'],
      dtype=object)
```

I have replaced some of the instance and there are few occasion where I have to combine few similar kind of instances. This has been achieved with replace() method.

During further analysis, I found that we may require Total Votes casted per constituency. This has been added with the help of Groupby and using aggregation on Data frame.



Use of Agg. Functions and adding columns in DF

```
#Let's Add column using Group by to find Total Votes casted in constituency
df_elec["tot_vote_casted"] = df_elec.groupby(["STATE","CONSTITUENCY"])["total_votes"].transform('sum')
df_elec
```

We have also check if there are any duplicate based on key during our analysis. We have few interesting analyses which are highlighted in Insight section of the report.

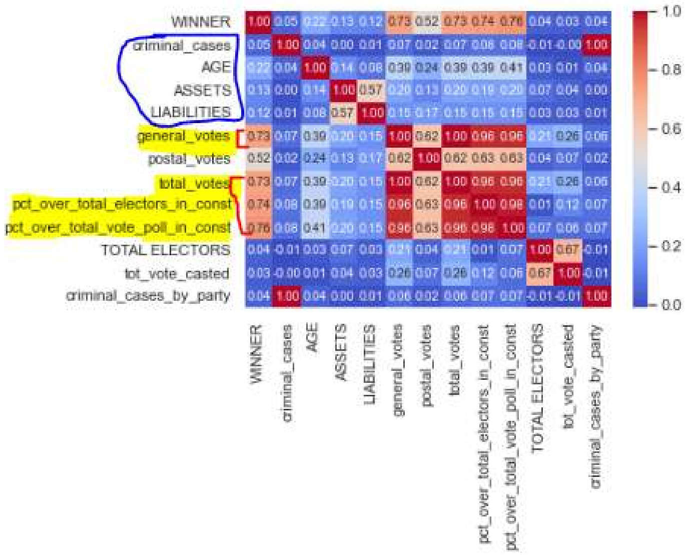
I spent major part of my time here in this section as the more you understand your data and fix DQ issue, you get good quality data for your model.

4. Classification Model

Once we have good quality data, we need to start preparing data for Model.

First let's try to understand if we have co-relations between features

```
#Let's try to understand if there is corelation between different fetures
# plotting correlation heatmap
dataplot = sns.heatmap(df_elec.corr(), cmap='coolwarm',cbar=True, annot=True,  fmt='.2f', annot_kws={'size': 9})

# displaying heatmap
plt.show()
```



Here you can see No. of Votes are related to Winners which is correct as that is the only deciding factor for a winner. Interestingly, No. of criminal cases, Age or Assets are not determining winners. There are no clear corelation between features.

The decision trees in scikit-learn can't work with string values, we need to convert strings to numbers with get_dummies()
For numeric values columns we will have to perform scaling. This will help model understand data better.

This will create few new columns. Basically, it will transpose your data from rows to column for categorial columns.

```
dataset = pd.get_dummies(df1, columns = txt_cols)

standardScaler = StandardScaler()
columns_to_scale = num_cols
dataset[columns_to_scale] = standardScaler.fit_transform(dataset[columns_to_scale])
dataset.head()
```

	WINNER	criminal_cases	AGE	total_votes	TOTAL ELECTORS	ASSETS	LIABILITIES	STATE_Andaman & Nicobar Islands	STATE_Andhra Pradesh	STATE_Arunachal Pradesh	...	CATEGORY_ST
0	1	7.018675	0.272939	0.448506	-0.534986	-0.291905	-0.205143	0	0	0	...	1
1	0	-0.179470	0.374295	0.219690	-0.534986	-0.252610	-0.197873	0	0	0	...	1
2	0	0.235807	0.272939	0.201809	-0.534986	-0.206583	-0.027182	0	0	0	...	1
3	0	-0.179470	-2.362307	-0.975103	-0.534986	-0.299825	-0.207877	0	0	0	...	0
4	1	0.512659	0.577006	1.501550	0.889409	-0.110040	-0.106233	0	0	0	...	0

5 rows × 598 columns

Now I tried doing resampling to make sure we have even spread. This is because we have many rows with losing the election and very few with winning. This is valid scenario as there are multiple candidates for single spot but winner will be only 1 out of them. I took 1724 because there are 1724 rows in majority dataset.

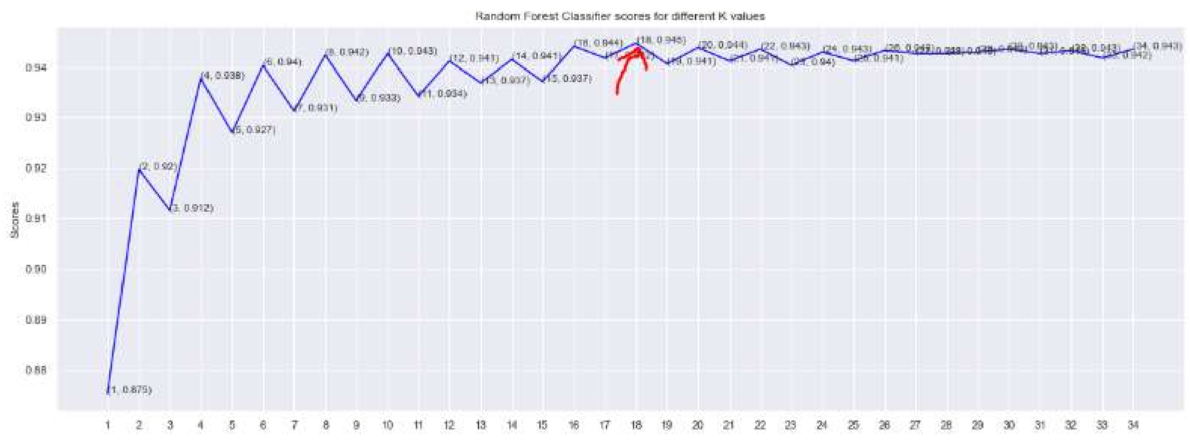
```
#balancing the dataset
df_majority = dataset[dataset.WINNER == 0]
df_minority = dataset[dataset.WINNER == 1]
df_minority_upsampled = resample(df_minority, replace = True,n_samples = 1724, random_state = 0)
df_upsampled = pd.concat([df_majority, df_minority_upsampled])
df_upsampled.WINNER.value_counts()
```

```
0    1724
1    1724
Name: WINNER, dtype: int64
```

Now, that we have balance dataset which can be pass to the model, we will be using Random Forest Classifier to predict the winners of the election. The good thing about random forests is that they’re easy to understand and can be utilized with great effect without any complicated hyperparameter tuning.
In order to know the optimum number of trees required to predict the result with highest accuracy, we will be plotting the accuracy score for various values of k and will be selecting k value that gives highest accuracy.

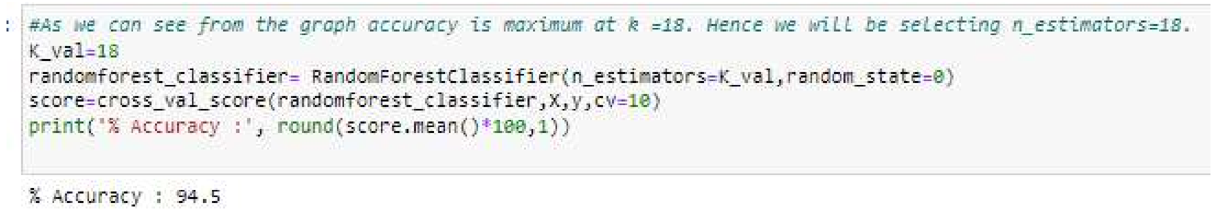
```
y = df_upsampled['WINNER']
X = df_upsampled.drop(['WINNER'], axis = 1)
max_kval = 35
rfc_scores = []
for k in range(1,max_kval):
    randomforest_classifier= RandomForestClassifier(n_estimators=k,random_state=0)
    score=cross_val_score(randomforest_classifier,X,y,cv=10)
    rfc_scores.append(score.mean())
plt.figure(figsize =(20,7))
plt.plot([k for k in range(1, max_kval)], rfc_scores, color = 'blue')
for i in range(1,max_kval):
    plt.text(i, rfc_scores[i-1], (i, round(rfc_scores[i-1],3)))
plt.xticks([i for i in range(1, max_kval)])
plt.xlabel('Number of Estimators (K)')
plt.ylabel('Scores')
plt.title('Random Forest Classifier scores for different K values')
```

```
Text(0.5, 1.0, 'Random Forest Classifier scores for different K values')
```



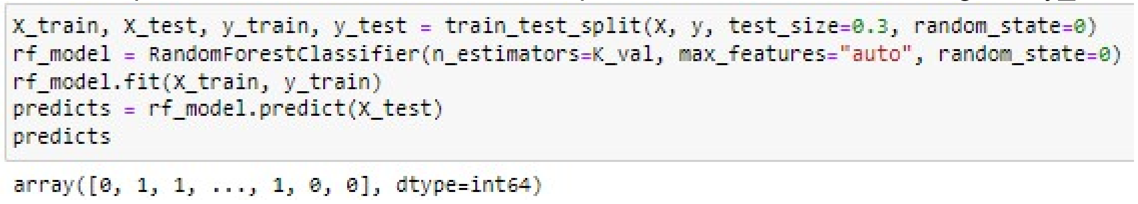
Results

With K value 18, We have achieved accuracy of 94.5%. This can be improved is required with Hyper parameter tuning (topic is I am yet to understand and learn about).



```
#As we can see from the graph accuracy is maximum at k =18. Hence we will be selecting n_estimators=18.
K_val=18
randomforest_classifier= RandomForestClassifier(n_estimators=K_val,random_state=0)
score=cross_val_score(randomforest_classifier,X,y,cv=10)
print('% Accuracy :', round(score.mean()*100,1))

% Accuracy : 94.5
```

I have divided the dataset into train and test and apply fit() method for learning and predict() method to predict the outcome. You can compare and validate the result against y_test.



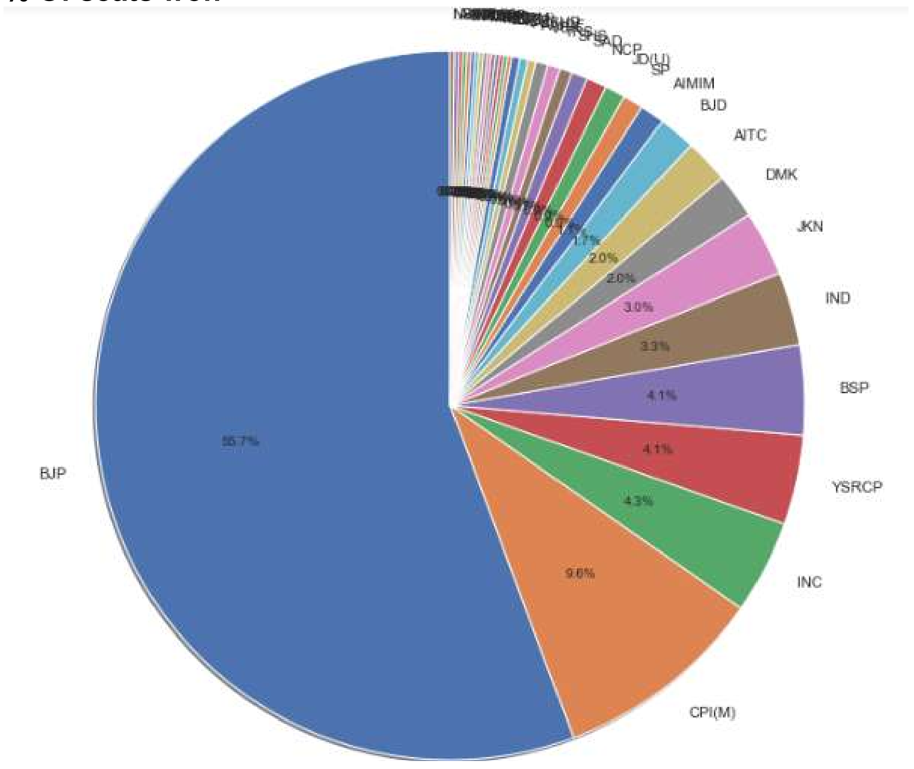
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
rf_model = RandomForestClassifier(n_estimators=K_val, max_features="auto", random_state=0)
rf_model.fit(X_train, y_train)
predicts = rf_model.predict(X_test)
predicts

array([0, 1, 1, ..., 1, 0, 0], dtype=int64)
```

Overall, I have learned the approach about ML project, what is required, things we need to take care of and areas of considerations.

Insights

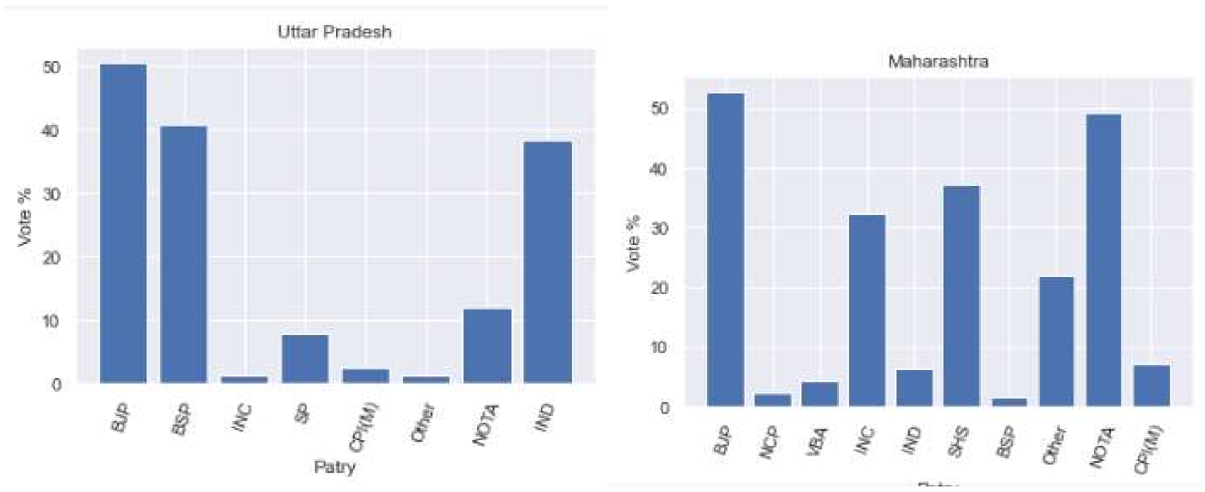
Here are some interesting insights about dataset.

% Of seats won



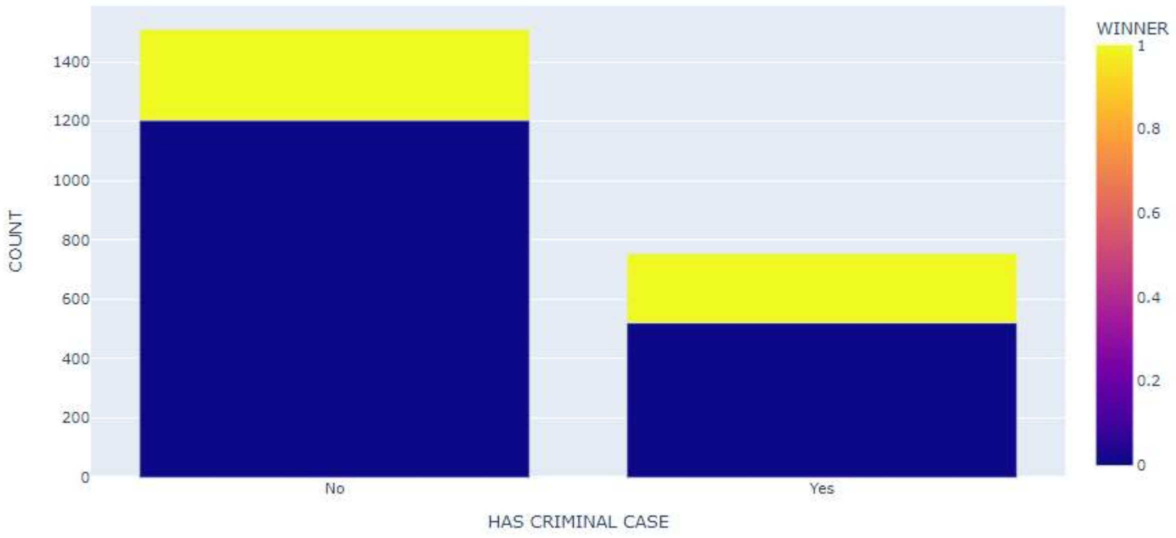
You can see, Party named "BJP" has won more then 50% of the seats. This is something extra ordinary but this makes dataset less balanced. For whatever reason this result seems to be bias towards one party.

Vote % in different states of India



For some key states, you can see vote % are not one sided, they are distributed between few parties, this means winning margin will be relatively less between winner and runner up. This proves that there are followers of different parties and they have come out and vote for their favourite.

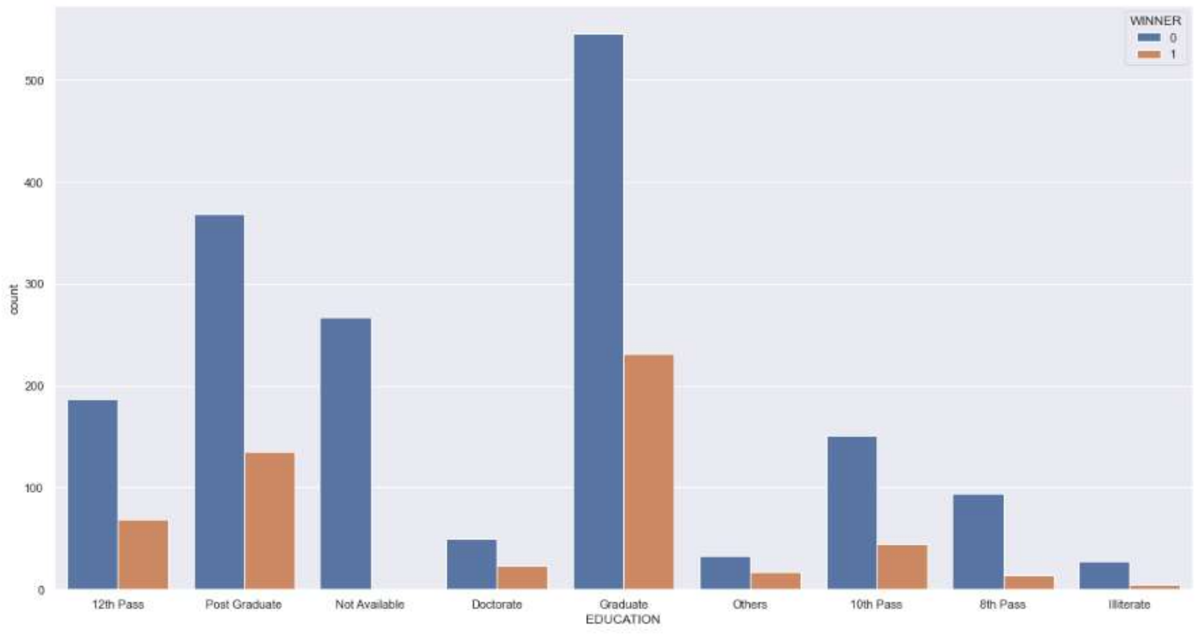
Effect of candidate with criminal cases



Here you can see there is *marginal* impact if candidate having clean image and do not have any open criminal case against them. I think major % of voters do not care about this parameter while voting.

Effect of Educated candidates

This is clearly visible that voters like educated candidate. Those who are educated have higher winning chances against less educated or illiterate.



References

- https://data36.com/random-forest-in-python/
- Different notebook samples from Kaggle and GitHub
- Python Library documentation