Python Assignment Report

Dhruv Gupta (220361)

April 2024

1 Introduction

We were provided with a dataset having following columns -

- ID - Serial ID for the candidate.
- Candidate - Name of the winning candidate.
- Constituency ∇ - Constituency from where the candidate won.
- Party - Political Party to which the candidate belongs.
- Criminal Case - Total number of criminal cases on the candidate.
- Total Assets - Total assets declared by the candidate.
- Liabilities - Liabilities declared by the candidate.
- Education - Education Level of the candidate.

We were required to analyse the data and develop a machine learning model to predict 'Education' of the candidates based on the rest of the data.
Link to github repository containg my code: https://github.com/dhruvgupta22/WITRW.git

2 Methodology

2.1 Libraries Used

- **NumPy** for mathematical functions and vector operations.
- **Pandas** for loading, manipulating and analysing the dataset.
- **Scikit Learn** for model selection and validation, evaluation metrics, feature selection, scaling and hyperparameter tuning.
- **Imbalanced Learn** for oversampling.
- **Matplotlib** and **Seaborn** for making plots and data visualization.

2.2 Data Preprocessing

In this section, we detail the steps taken to preprocess and prepare the dataset. Firstly, the dataset was inspected manually to figure out what kind of preprocessing on data was required. Following steps were taken after inspection -

- ***Total Assets*** and ***Liabilities***: These are supposed to be numeric data but were provided as strings in the dataset, for eg., 21 Crore+, 10 Lac+, etc. These were **converted to integer values**. For eg., 21 Crore+ was converted to 210000000, 10 Lac+ was converted to 1000000, and so on.

- ***Criminal Case***: Inspection showed that approximately half of the entries are 0. Therefore, this data was **converted to binary**, i.e., 1 if non zero number of criminal cases, 0 otherwise.

- ***Party*** and ***state***: These are categorical features and hence they need to be converted into numerical data. For this **one hot encoding** was used so that each category gets equal importance. This led to increase in number of dimensions which was later handled during dimensionality reduction in feature selection.

  Note: **pandas.get_dummies()** method was used to create one hot encoded columns/features.

- ***ID***: Since name of the candidate and ID are different for each data point, *Education* is clearly independent of these. Therefore they were **removed from the dataset**.

- ***Education***: This is also categorical data. But unlike *Party* and *state*, *Education* can be compared and ordered. Therefore, **numerical label encoding** was used for *Education*.
  Table 1 shows the numeric label given to each category in *Education*.

- ***Candidate*** and ***Constituency*** ∇: This data was **used in feature engineering** to make new features, but it was **removed later** from the dataset.

Educational Qualification	**Numerical Label**
Others	0
Literate	1
5th Pass	2
8th Pass	3
10th Pass	4
12th Pass	5
Graduate	6
Graduate Professional	7
Post Graduate	8
Doctorate	9

Table 1: Mapping of Educational Qualifications to Numerical Values

2.3 Feature Engineering

2.3.1 Making New Features

New features were made using existing features and concatenated with the preprocessed dataset.

- Two new features, ***ALRatio*** and ***Net Assets*** were made using *Total Assets* and *Liabilities*.

$$ALRatio = \frac{Total\ Assets}{Liabilities + 1} \tag{1}$$

$$Net\ Assets = Total\ Assets - Liabilities \tag{2}$$

  ALRatio refers to the ratio of *Total Assets* and *Liabilities*. However, in equation (1), 1 was added in the denominator to handle the 0 values in *Liabilities*.

- **Scaling** of numerical features, *Total Assets*, *Liabilities*, and *Net Assets*) was done to create corresponding 'Scaled' features, i.e., *Scaled Total Assets*, *Scaled Liabilities*, and *Scaled Net Assets*.

- **Logarithm** of numerical features, *Total Assets*, *Liabilities*, and *ALRatio* was taken to create corresponding 'Log' features, i.e., *Log Total Assets*, *Log Liabilities*, and *Log ALRatio*. *Log ALRatio* was calculated using equation (3).

$$Log\ ALRatio = Log\ Total\ Assets - Log\ Liabilities \tag{3}$$

  $log(0)$ was taken as 0 while taking logarithm.

  Note: For taking logarithm, **numpy.log()** method was used.

- **Grouping of states** into six regions was done to create six new columns in the dataset namely, *North*, *South*, *East*, *West*, *North East*, and *Central*. All the data was one hot encoded w.r.t these categories.
  Table 2 shows the mapping between states and regions.

Region	**States**
South	Tamil Nadu, Karnataka, Kerala, Andhra Pradesh, Puducherry
North	Uttar Pradesh, Punjab, Haryana, Delhi, Himachal Pradesh, Rajasthan, Uttarakhand
West	Maharashtra, Gujarat, Goa
East	West Bengal, Odisha, Jharkhand, Bihar
North East	Meghalaya, Manipur, Nagaland, Sikkim, Tripura, Assam, Arunachal Pradesh
Central	Madhya Pradesh, Chhattisgarh

Table 2: Regions and Corresponding States

- **Grouping of parties** into six categories based on there region was done to create six new columns namely, *North Parties*, *South Parties*, *East Parties*, *West Parties*, *North East Parties*, and *Nationwide Parties*. All the data was one hot encoded w.r.t these categories.
  Table 3 shows the mapping between parties and their region.

Party Region	Parties
North Parties	AAP, JD(U), SHS, TDP
South Parties	AIADMK, DMK, JD(S), YSRCP, Sikkim Krantikari Morcha, Tipra Motha Party, SP
East Parties	AITC, BJD, CPI, CPI(M), RJD
West Parties	NCP, NDPP
North East Parties	IND, JMM, NPP
Nationwide Parties	BJP, INC

Table 3: Party Regions and Corresponding Parties

Category	Number
SC	299
ST	214
General	1546

Table 4: Number of Candidates by Category

- **Grouping of constituencies** based on reservation category was done. Three categories, SC, ST, and General were created and data was one hot encoded. Once the grouping was done, *Constituency* ∇ was removed from the dataset.
  Table 4 shows the number of SC, ST, and General candidates.

- **Extracting designations from *Candidate***: Some of the candidates had designations 'Adv.' and 'Dr.' in their name. These were encoded as binary data in *Advocate* and *Dr* columns respectively. *Candidate* was then removed from the dataset.

2.3.2 Analysing Correlations

For feature selection, correlations of all the columns with the *Education* column were found using **pandas.DataFrame.corr()** method. **Table 5** shows the correlation of the numerical features with *Education*.

Feature	Correlation
Total Assets	-0.002278
Liabilities	0.007187
Net Assets	-0.006409
ALRatio	-0.004215
Scaled Total Assets	-0.002278
Scaled Liabilities	0.007187
Scaled Net Assets	-0.006409
Log Total Assets	0.021701
Log Liabilities	-0.005026
Log AL Ratio	0.018312

Table 5: Correlation of Numerical Features

Note: Correlations with categorical features were not included in report for brevity.

2.3.3 Feature Selection

From the correlations in Table 5, we can note that scaling the features does not affect their correlations but it will help in reducing training time. Therefore, unscaled features were removed from the dataset.

Feature Selection was done using multiple options -

1. Remove individual states and only keep grouped states.
2. Remove individual parties and only keep grouped parties.
3. Remove state and parties with a very high or very less number of entries in the dataset.
4. Use sklearn.feature_selection to select k best features

Combinations of these options were tried with various models. For option 4, values of k ranging from 20 to 'all' features were tried. Feature selection was done due to two reasons - to decrease the number of dimensions so as to reduce training time, and to discard irrelevant features and select only the relevant features.

2.4 Dimensionality Reduction

Dimensionality reduction was done as a part of feature selection, where less significant features were removed in order to reduce the dimensions. **SelecKBest** method from the **Scikit Learn** library was used for selecting the most relevant features.

2.5 Normalization

Normalization (or Scaling) of numerical features, *Total Assets*, *Liabilities*, and *Criminal Case* was done as a part of feature engineering. Two different scalers - **StandardScaler** and **MinMaxScaler** from the **Scikit Learn** library were used (one at a time) for normalising the numerical data.

2.6 Oversampling

Oversampling was done using **SMOTE** from **imblearn** library as an attempt to balance the number of entries for different categories by creating synthetic data, since imbalanced data may lead to overfit and low f1 scores. Some variations with slight imbalances were also tried, for e.g., leaving the categories with very low number of entries imbalanced.

Two modes were tried - First, replacing the original training data completely with the synthetic data. Second, concatenating the synthetic data with the original data.

2.7 Hyperparameter tuning

GridSearchCV from **sklearn** library was utilized to automate hyperparameter tuning for all models. A classifier list containing all the models and their param grids was created manually to search for best parameters for each model. Data was co-validated 5 times (cv = 5) to find parameters which give maximum *weighted f1 score*.

Then data was once again validated before creating submission files, for all the models using the 'best params' found using GridSearchCV. Various values of validation-training split ratio was tried during this step.

3 Experiment Details

3.1 Models Used

Table 6 shows all the models from **sklearn** which were tried along with their parameters which were tuned.

Classifier	**Parameters Tuned**
BernoulliNB	alpha, force_alpha, fit_prior, class_prior
RandomForestClassifier	n_estimators, max_depth, criterion, min_samples_split, min_samples_leaf
DecisionTreeClassifier	max_depth, criterion, splitter, min_samples_split, min_samples_leaf
KNeighborsClassifier	n_neighbors, weights, algorithm
SVC	C, kernel, degree, gamma

Table 6: Classifiers and Parameters Tuned

General Observations:
In many cases, RandomForestClassifier, DecisionTreeClassifier and KNeighborClassifier led to overfitting. SVC has a large train time. BernoulliNB turns out to be the optimum choice, since it gives almost similar results on validation and test datasets and has a low train time.
The final submission file was made using BernoulliNB.

3.2 Data Visualization

This section contains informative plots for data analysis.

- **Figure 1** shows the percentage of candidates with more than 2 criminal records for each party. It is evident that almost all the parties have at least 10% candidates with more than 2 criminal cases. Further more, regional parties like DMK, RJD, SP, etc., have a relatively higher percentage as compared to National parties like BJP and INC.

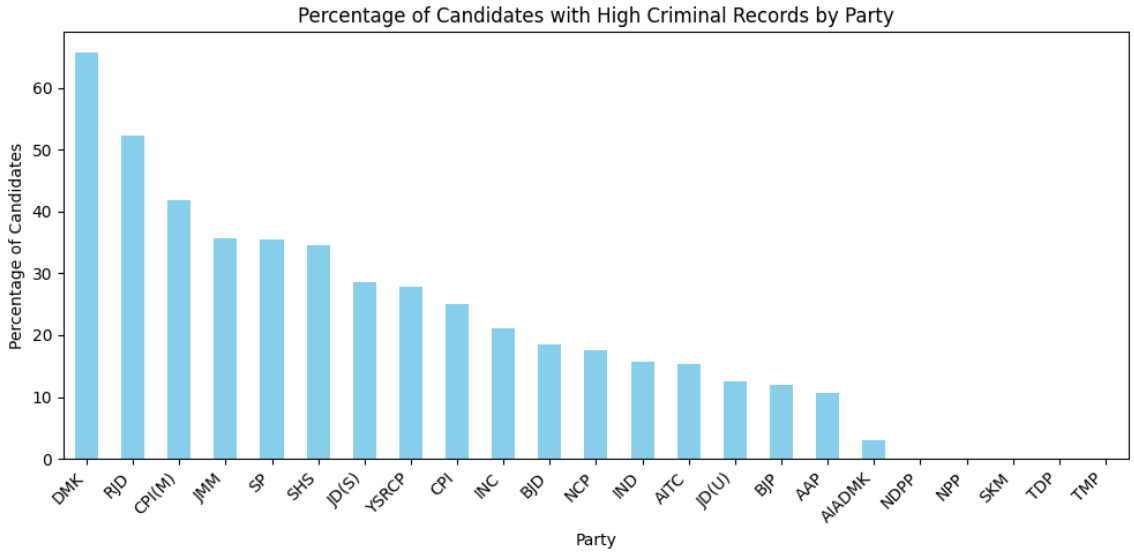

Figure 1: Percentage Distribution of Parties with Candidates with Most Criminal Records

- **Figure 2** shows the average number of criminal cases per candidate for each party.
  It can be observed that this bar graph almost overlaps with the bar graph in Figure 1, that is parties with higher percentage of candidates with high criminal record also tend to have higher number of cases

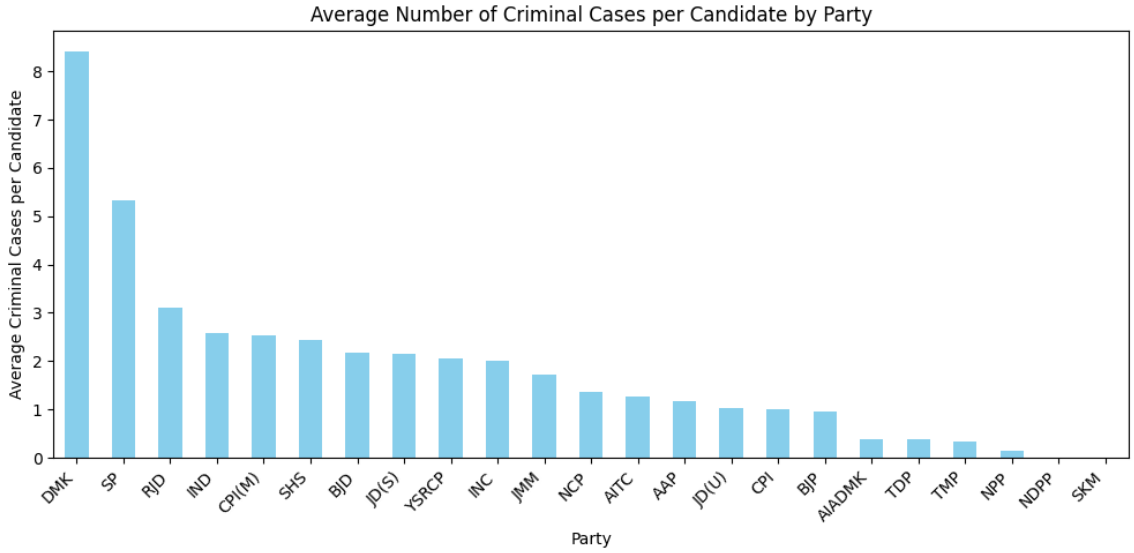

Figure 2: Average Number of Criminal Cases per Candidate for each Party]

- **Figure 3** shows percentage of candidates from each party with Total Assets more than 1 Crore

and Net Assets more than 1 Crore. It can be seen that a large proportion of candidates in each party have huge amounts of wealth. Furthermore, Net Assets and Total Assets graphs almost coincide, implying really less number of Liabilities.

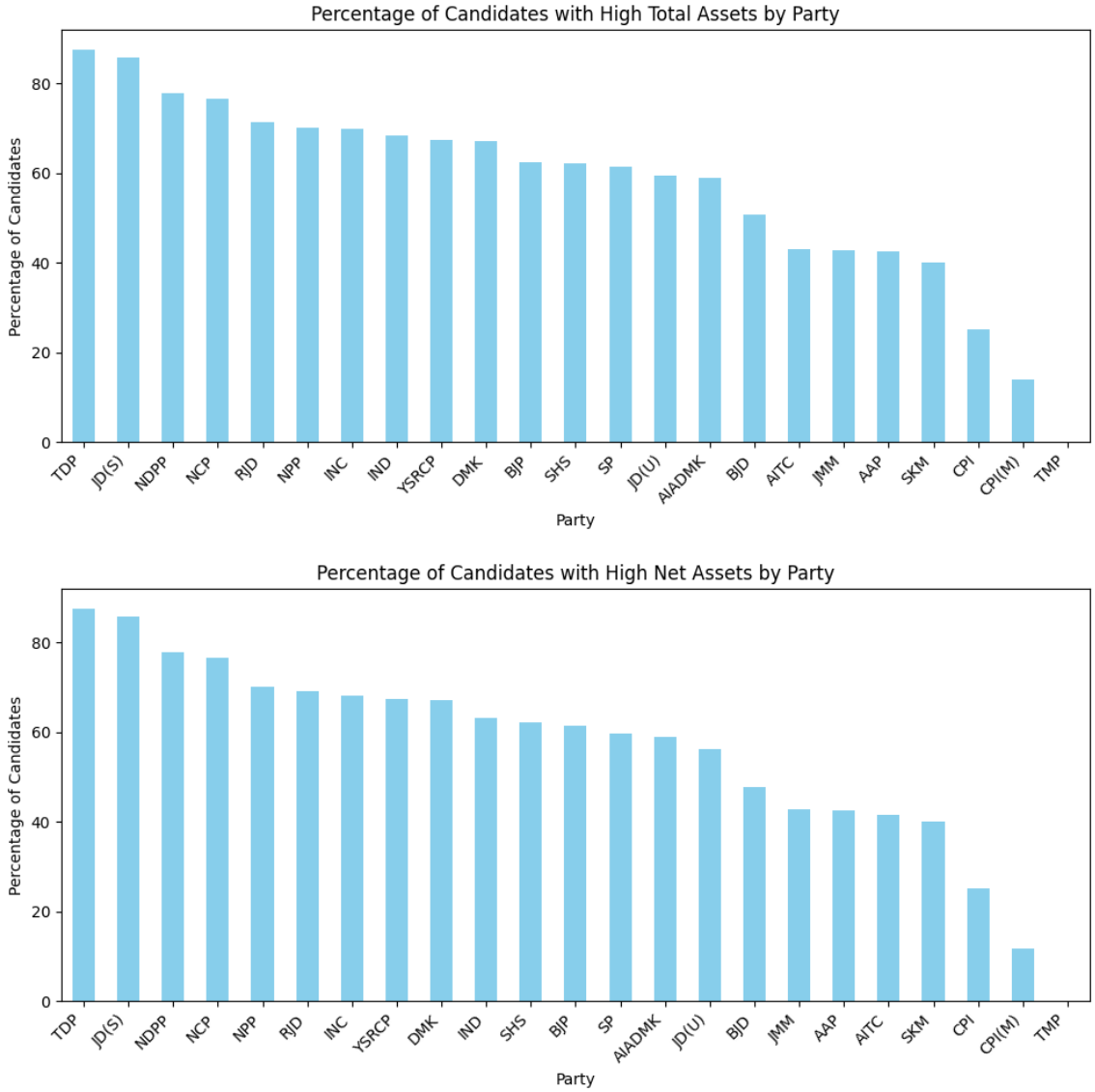

Figure 3: Percentage Distribution of Parties with Most Wealthy Candidates

- **Figure 4** shows the Average Total Assets and Average Net Assets for candidates of each party. Data is plotted using a logarithmic scale because of large variations in wealth.
  It must be noted that that average wealth of almost all the parties cross the 1 Crore mark, and many even cross the 10 Crore mark. However, this graph deviates from the data in Figure 3, as the ranking of parties has changed. For eg., IND and INC have lower percentages but

higher averages than NDPP and NPP. This suggests a skew in wealth, i.e., a small portion of people in INC and IND have a larger proportion of wealth.

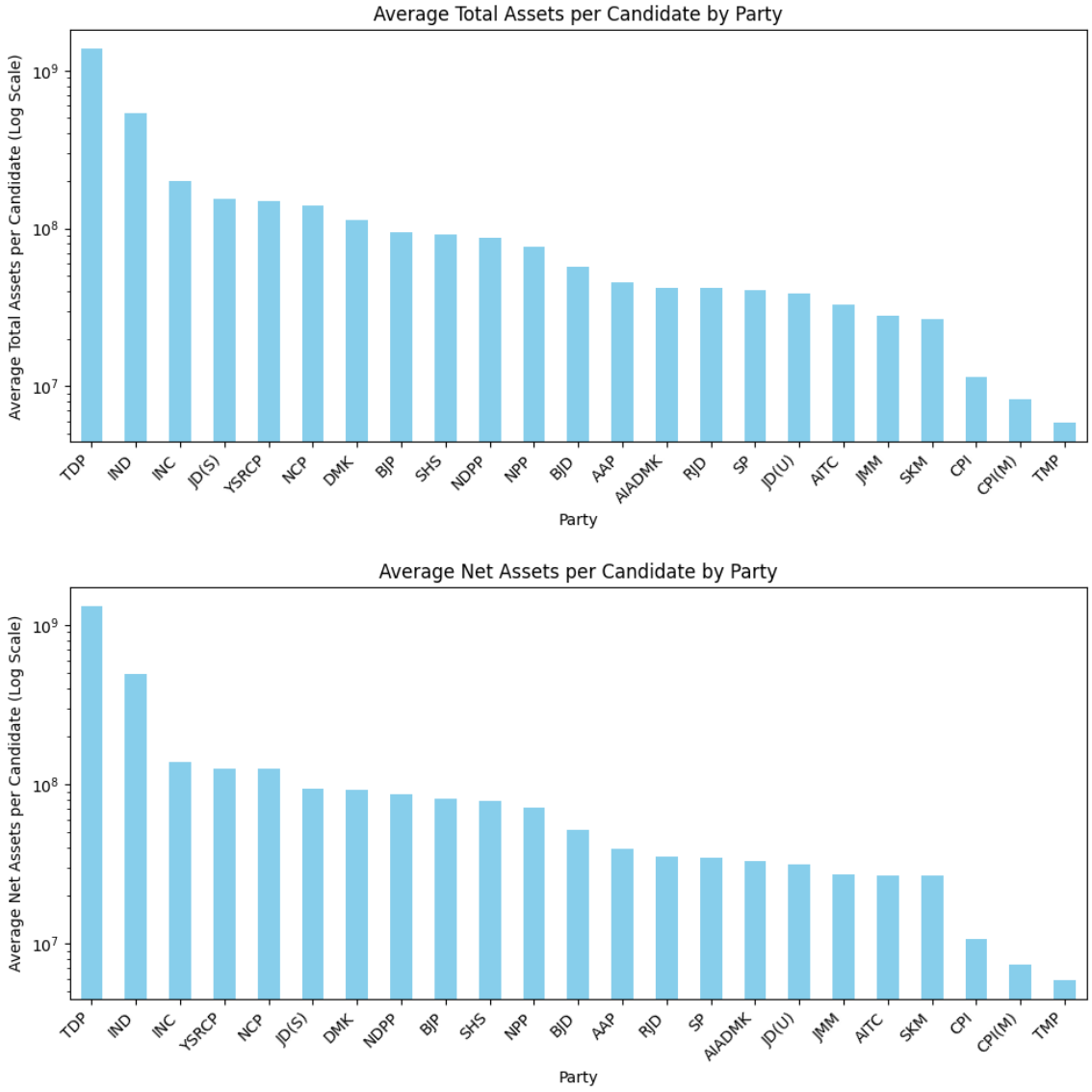

Figure 4: Average Wealth of Candidates for each Party

- **Figure 5** shows the total number of criminal cases of all the candidates of each state. It can be seen that this data is in accordance with Figure 1 and Figure 2. DMK is a regional party of Tamil Nadu, SP is a regional party of Uttar Pradesh, RJD is a regional party of Bihar, and the positions of states and parties are similar among others. Similar observations can be made for other regional parties and their states.

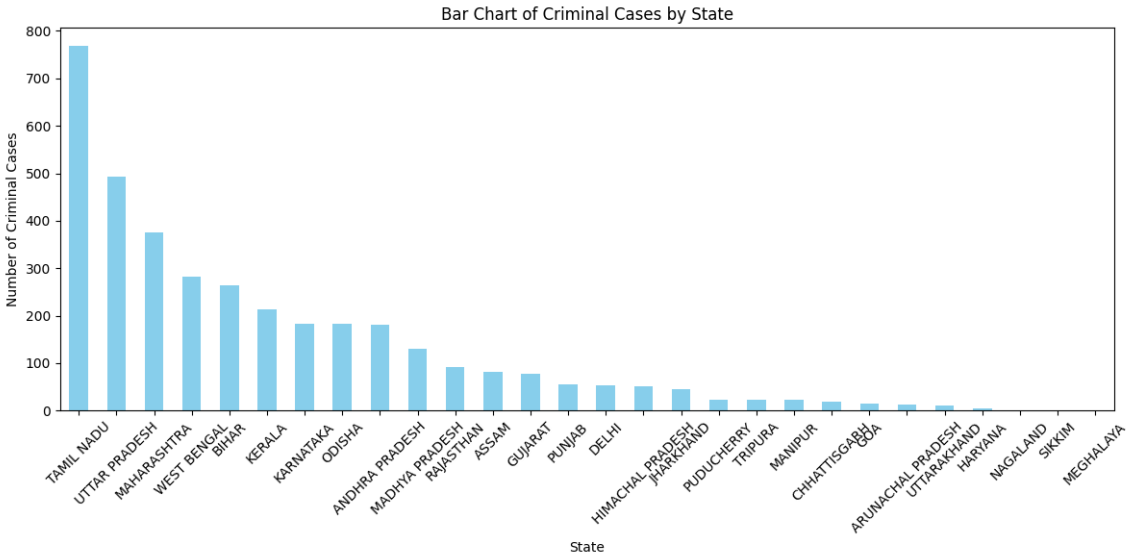

Figure 5: Total Criminal Cases for each State

- **Figure 6** depicts the proportion of number of candidates of each party. It can be clearly seen that national parties like BJP and INC have more than half of the candidates while regional parties have a really smaller share.

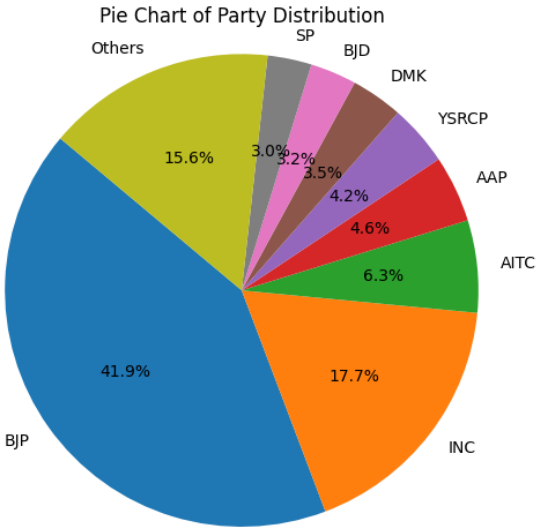

Figure 6: Percentage of Candidates of each Party

- **Figure 7** depicts the number of candidates for each education level for each region. Regions are as formed by grouping of states described in Table 2.

It can be observed that number of Graduates, Post Graduates, and Graduate Professionals are relatively higher across all regions. On the other hand, lower education levels have less proportions. This is a good sign that a lot of candidates are educated.

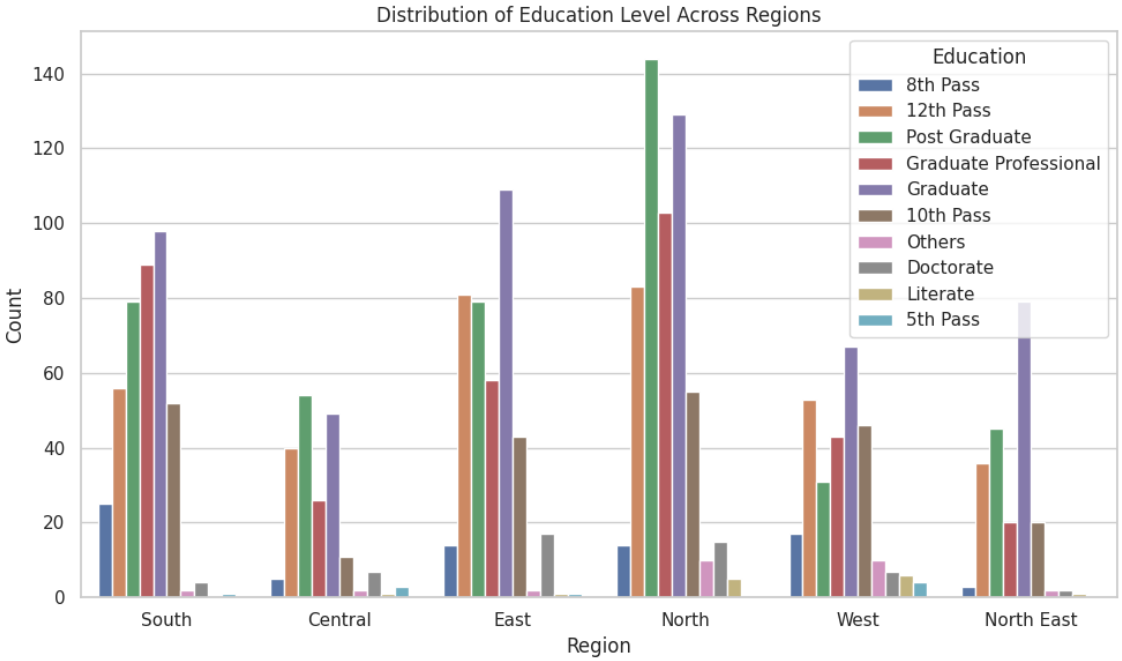

Figure 7: Education levels in each Region

- **Figure 8** depicts the number of candidates for each education level for regional and nationwide parties. Regional Parties refers to all the parties other than Nationwide parties (mentioned in Table 3).
  It can be observed that each education level has a similar proportion in each category.

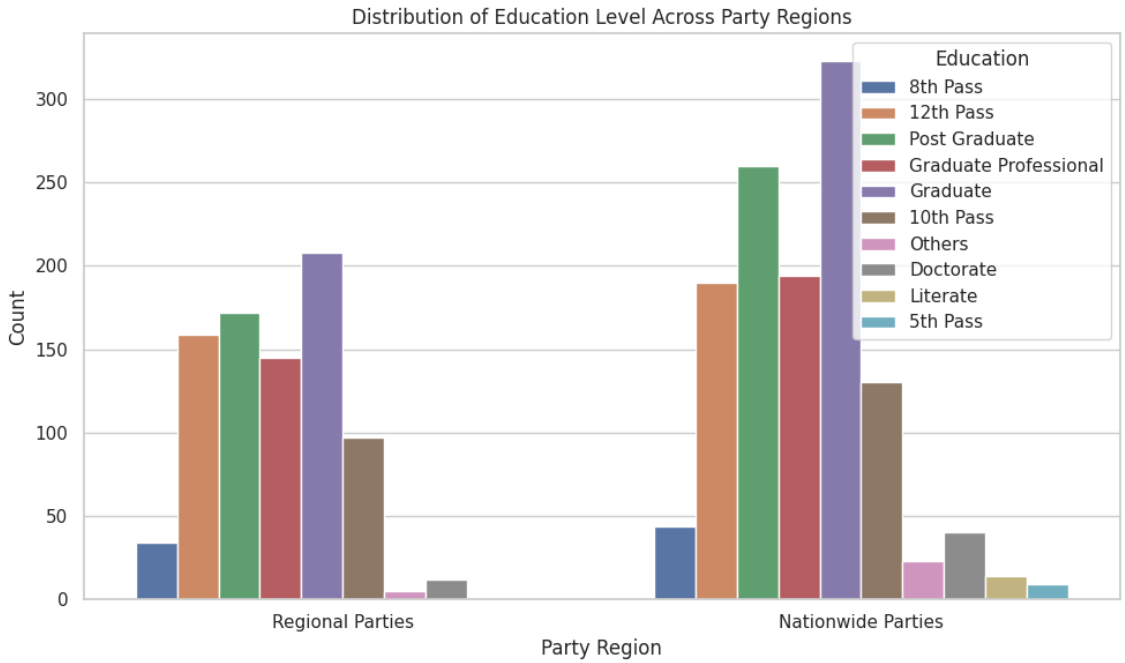

Figure 8: Education levels in Regional and Nationwide Parties

- **Figure 9** depicts the proportion of candidates for each education level for each Reservation Category. Reservation Categories are made as mentioned in grouping of constituencies (refer to Section 2.3.1).
  It can be observed that each education level has a similar proportion in each category.

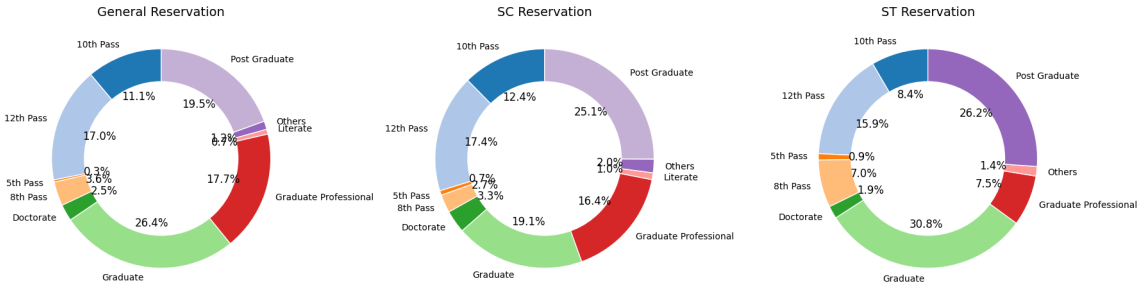

Figure 9: Education level percentages in each Reservation Category

3.3 Key Insights

1. It can be noted that a large proportion of candidates are either Graduates, Post Graduates, or Graduate Professionals. This results in a large number of entries being predicted among these leading to overfitting and a lower f1 score.

2. It should be noted that distribution of education levels is almost similar across (state) regions (refer to Figure 7). This explains the low correlations with respect to states and regions formed after grouping.

3. Similarly, for parties, it can be noted from Figure 8, that distribution is similar for both Nationwide and Regional Parties. This explains the low correlations with respect to parties and the groups formed based on region.
   Same is the case with Reservation Categories extracted from Constituencies.

4. Scatter plots of Education vs Total Assets, Education vs Criminal cases were made. No particular relation could be observed. These plots were therefore omitted from the report. However, it explains low correlations of the numerical features with education.

5. Oversampling of data results in balancing of number of entries for each education level. This results in much higher f1 scores on validation dataset with Decision Trees, KNeighborClassifier and RandomForestClassifier. However, score reduces for test data. This is a clear indication of overfitting.
   However, with BernoulliNB, oversampling causes a slight improvement in f1 scores for both validation and test datasets, i.e., no overfitting is observed. However, if we keep increasing the oversampled data, it leads to overfitting in this case too.

6. Dropping some features like, parties with very large or very less number of candidates, states with very large or very less number of candidates, helped to improve f1 score with BernoulliNB. However, dropping a large number of features lead to lower scores.

4 Results

- Final F1 Score:
- Public Leaderboard Rank:
- Private Leaderboard Rank:

5 References

- numpy: `https://numpy.org/doc/stable/`
- pandas: `https://pandas.pydata.org/docs/`
- **Feature Engineering:**
  - MinMaxScaler: `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html`
  - StandardScaler: `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html`

  - SelectKBest: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html
  - f_classif: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.f_classif.html
  - SMOTE: https://imbalanced-learn.org/stable/references/index.html
- **Models:**
  - RandomForestClassifier: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
  - DecisionTreeClassifier: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
  - KNeighborsClassifier: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
  - BernoulliNB: https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.BernoulliNB.html
  - SVC: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
- **Evaluation and Validation:**
  - test_train_split: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
  - f1_score: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html
- **Plotting:**
  - matplotlib: https://matplotlib.org/stable/contents.html
  - seaborn: https://seaborn.pydata.org/