CS253 Python Assignment Report

Pragati Agrawal (220779)

April 2024

1 Methodology

1.1 Introduction

This assignment concerns a multi-class classification task aimed at predicting the education levels of candidates. The dataset provided comprises the 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. (The target variable)

1.2 Libraries Imported

- **re**: Used for pattern matching of strings in data cleaning.
- **scipy.stats**: Used to calculate z-score and indentify outliers.
- **numpy**: Numerical library, used for arrays and mathematical functions.
- **pandas**: Data manipulation and analysis library, used for working with tabular data.
- **matplotlib.pyplot**: Plotting library, to create plots and identify relevant features.
- **seaborn**: Also used for plotting. Offers a variety of plotting techniques.
- **sklearn**: The python library ”Scikit-learn”. It is used to train different models on the given dataset and also to print the accuracy reports of each model. It includes various functions like Label Encoder, Selecting k-Best features and printing confusion matrix.
- **sklearn.model_selection.GridSearchCV**: It is used for hyperparameter tuning. It tries all the given parameters listed out in the ”params_grid” and returns the best set of parameters. The metric used to decide the best set of parameters is ”F1 score weighted”.

1.3 Data Preprocessing

- Dropped the columns: **ID** since it is different for everyone.
- The columns **Total Assets** and **Liabilities** needed to be converted to numbers from strings. I made two functions: one to convert these strings to int values, and the other to convert them to float values. Tried improving accuracy using both of these functions.
- Performed **One-Hot encoding** of the columns **State** and **Party** to have better dependence of target variable (Education) on each columns' values. Replaced these one-hot encoded values: 'False', 'True' by 0/1 respectively for numerical data.
- The **Candidate** name of any candidate can be decisive in predicting the education level if that candidate is an **Advocate** or a **Doctor of Philosophy**. All Advocates (Adv. in Name) must be classified as Graduate Professionals while the Doctorates (Dr. in Name) be classified among Post Graduate, Doctorate and Graduate Professionals.
  For this I created two columns: *Advocate* and *Doctor*, and filled 1 if that field was true.

1.4 Feature Engineering

For this I tried making new features from the existing ones to improve prediction:

- **Net Capital:** It is defined as the difference between the 'Total Assets' and 'Liabilities' of any candidate. It gives us an idea of the actual cash he/she has at hand.
- **Leverage Ratio:** It is defined as the ratio of the 'Total Assets' to the 'Net Capital' of any candidate. Since 'Net Capital' might come out to be 0 and to avoid division by zero, the value of Total Assets' is divided by the median of all 'Net Capital' values in those cases.
- **Criminal Cases:** I created a new column that is 1 iff that candidate has atleast one criminal case against him/her, 0 otherwise.
- **States Grouping:** States had a good correlation with the 'Education' values. So I grouped states into six regions:
  - **North**: Jammu and Kashmir, Himachal Pradesh, Punjab, Haryana, Uttarakhand
  - **South**: Andhra Pradesh, Telangana, Karnataka, Kerala, Tamil Nadu
  - **East**: Bihar, Jharkhand, Odisha, West Bengal, Sikkim
  - **West**: Rajasthan, Gujarat, Goa, Maharashtra
  - **Central**: Madhya Pradesh, Chhattisgarh, Uttar Pradesh, Delhi
  - **North East**: Assam, Meghalaya, Tripura, Nagaland, Manipur, Mizoram, Arunachal Pradesh

  Also performed One-Hot encoding of these regions.
- **Constituencies Grouping:** Constituencies had the type written in brackets: **SC**, **ST** and **General**. So I created three new one-hot encoded columns: to denote *SC_Constituency*, *ST_Constituency* and *General_Constituency*.
- **Parties Grouping:** Parties can also be classified into two: **National** and **Regional**. Created one-hot encoded column for National versus Regional parties:

  - **Regional**: AAP, AIADMK, AITC, BJD, DMK, JD(S), JD(U), JMM, NDPP, NPP, RJD, SHS, SP, Sikkim Krantikari Morcha, TDP, Tipra Motha Party, YSRCP
  - **National**: BJP, CPI, CPI(M), INC, IND, NCP

1.5 Correlation of Education with others

- To find the correlation of Education field with other columns, I label-encoded Education categorical data and found the correlation w.r.t. every other column. The top 15 most correlated columns are:

State/Party	**Corr(%)**
Uttar Pradesh	0.100315
Region Central	0.098486
Maharashtra	0.077860
SHS	0.076779
JMM	0.069881
Gujarat	0.064532
Jharkhand	0.057861

State/Party	**Corr(%)**
NDPP	0.054552
Tipra Motha Party	0.053749
Nagaland	0.050284
Bihar	0.048069
INC	0.044707
Chhattisgarh	0.041963
RJD	0.040645

Table 1: State/Party and Correlation Percentage

1.6 Removing Outliers

To find the outliers, I used the z-score statistic for each feature in the dataset. If the absolute Z-score for any sample exceeds a specified threshold of 3, it is indicated as an outlier.
Formula for z-score:

$$Z = \frac{x-\mu}{\sigma}$$

For threshold value 3, 1905 rows out of 2058 rows came out to be outliers. This indicates that the dataset is too scattered. There is no significant correlation.

1.7 Scaling

Since the values of **Total Assets** and **Liabilities** had a great range of variation from 0 to *crores*, I tried scaling them in 2 different ways:

- **MinMaxScaler**: This ensures that all feature values are scaled to fall within the range of 0 to 1. It preserves the relative relationships between feature values while bringing them to a consistent scale. The *fit* method computes the minimum and maximum values for each feature in the training set, and then the *transform* method scales the feature values accordingly based on these computed minimum and maximum values. The formula it uses is:

$$X_{\text{scaled}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}}$$

  where X is the original feature value, X_{min} is the minimum value of the feature in the dataset, X_{max} is the maximum value of the feature in the dataset, and scaled is the scaled feature value.

- **StandardScaler**: This standardizes features by shifting the **mean to** 0 and scaling to **unit variance**. The *fit* method computes the mean (μ) and standard deviation (σ) for each feature in the training set, and then the *transform* method standardizes the feature values accordingly based on these computed statistics using the formula:

$$X_{\text{scaled}} = \frac{X - \mu}{\sigma}$$

  where X is the original feature value, μ is the mean of the feature in the dataset, σ is the standard deviation of the feature in the dataset, and X_{scaled} is the standardized feature value.

1.8 SMOTE for over sampling

Since we had only 2058 training points, to train the model more accurately, I tried minimising the skewness of the train data set. After several submissions, I observed that submissions leading to higher public scores typically had a higher frequency of individuals categorized as "Graduates" followed by "Post Graduates" and "Graduate Professionals." Next were individuals categorized as "10th" and "12th Pass" with the remaining categories following suit. Consequently, I oversampled these categories proportionally to better train the model.

```
desired_samples = {‘Graduate’: 1500, ‘Post Graduate’: 1300, ‘12th Pass’:1000,
‘Graduate Professional’:1300, ‘10th Pass’:1000, ‘8th Pass’:400, ‘Doctorate’:400}
```

1.9 Dimensionality Reduction

- After creating all these new features, I tried removing many different types of rows, to improve F1 score. I had tried by removed Liabilities, SC/ST/General Constituencies, National/Regional Parties, States Grouping, some Parties which were too small in number like ‘NDPP’,‘TDP’,‘JD(S)’,‘CPI’ etc in various trials.
- I also used sklearn's **SelectKBest** function to select $k = 40$ best features of my train data set, wrt the target variable Education.

1.10 Hyperparameter tuning

For any given model, the task is to find the best set of parameters. For this I used the **GridSearchCV** technique, which takes in input the **param_grid** (the set of values for each parameter of the model) using which the model is trained and finally, the function GridSearchCV returns the best set of parameters for that model and given set of values of each parameter (the unspecified ones remain default ones).
The metric used to find the best set of parameters is **weighted F1 score**. The value of **cv** (cross validation) was set to 5, which means the dataset will be divided into 5 equal parts, and the model will be trained and validated 5 times, each time using a different subset as the validation set and the remaining subsets as the training set.
Using the best set of parameters, I trained upon the cross validation set and test data set. The value of the weighted F1 score on the CV set would be an approximate indicator of its value on the test data set.

2 Experiment Details

The github repo for this assignment can be found here:

https://github.com/apragati22/Who-is-the-real-winner-

2.1 Models Used

The train data was split into 13% **Cross Validation set** with the **random state 48**. The following table provides the list of all the models I used. F1 Score mentioned in the table is the value of the highest weighted F1 score attained using that model on Cross Validation set. Similarly, Public and Private scores are the highest values obtained for that model.

Model Name	**Best Params**	**F1 Score**	**Public Score**	**Private Score**
BernoulliNB Classifier	alpha : 1.0 force_alpha: True fit_prior: True binarize : 0.0	0.24	0.26365	0.26463
Decision Tree Classifier	splitter: 'best' criterion: 'entropy' max_depth: 14	0.20	0.22325	0.24306
Random Forest Classifier	n_estimators: 100 min_samples_split: 9 criterion: 'entropy' max_depth: 13	0.21	0.23780	0.23586
K Nearest Classifier	n_neighbors: 6 weights: 'distance' metric: 'euclidean'	0.21	0.23349	0.22102
SVC Classifier	C: 15 tol: 0.1 kernel: 'rbf'	0.19	0.19964	0.22665
Gradient Boosting Classifier	learning_rate: 0.1 loss: 'log_loss' n_estimators: 100	0.18	0.13675	0.14873

Table 2: Model Evaluation

2.2 Data Visualization

1. Plot of candidates of various parties versus the percentage of criminal cases filed against them: We can infer from the graph below that almost all national parties have less than 50% criminal records on average. The parties **RJD**, **DMK** and **JDS(U)** have the highest percentage of candidates with criminal cases.

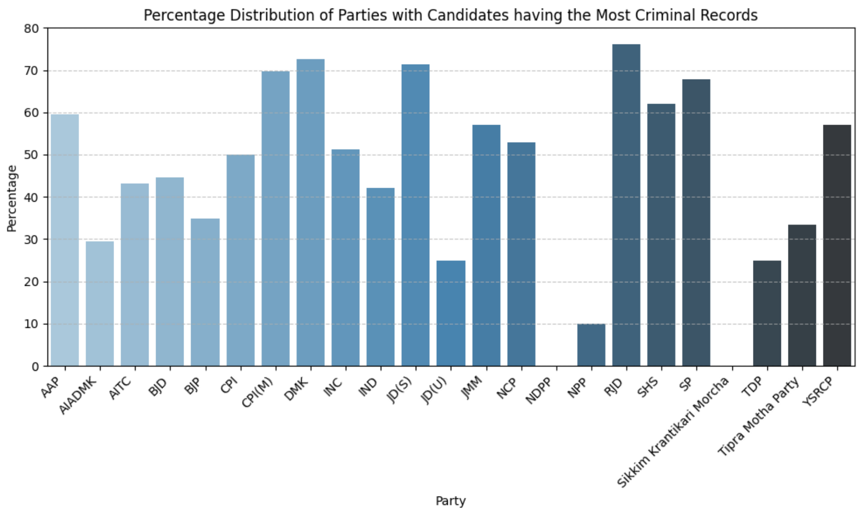

Figure 1: Parties v/s Average number of Criminal Records of their candidates

The following pie-chart shows percentage distribution of candidates who have 10 or more criminal cases against them wrt their party. Clearly 'DMK', 'INC' and 'BJP' have more percentage of such candidates. This may also be because they have more candidates than other parties.

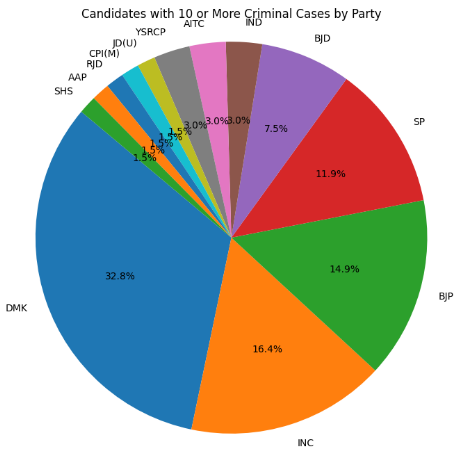

Figure 2: Parties v/s Candidates with atleast 10 criminal cases

2. Plot of the Wealth of Candidates of Various parties
   The plot below shows that almost all parties have an equitable distribution of assets among their candidates. So this feature is not useful to predict education levels.

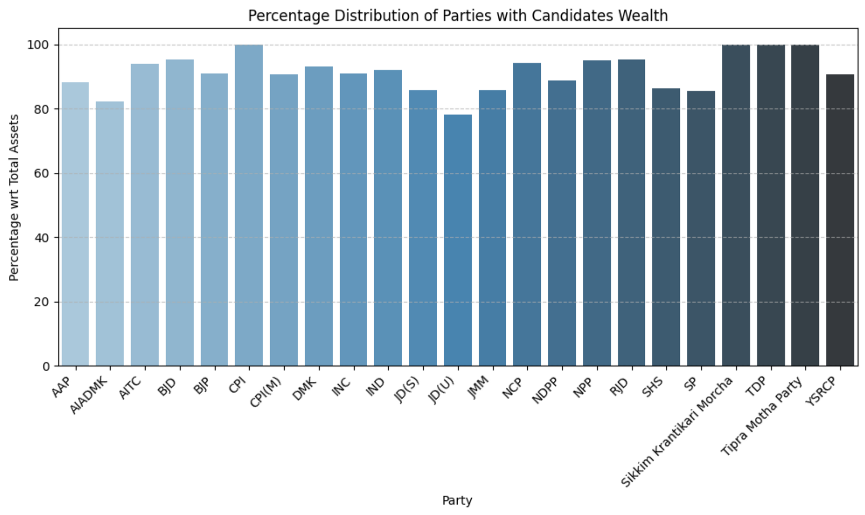

Figure 3: Total Assets of Candidates of various Parties

3. Plot of Education Level distribution according to State
   The graph depicts that for each state, on an average, the **Graduate** level of Eduaction is the most prevelant, followed by **Post Graduate** and **Graduate Professional**.
   States with larger populations, such as Uttar Pradesh, Madhya Pradesh, and Maharashtra, tend to have a higher count of educated candidates due to the larger number of individuals contesting elections in those states.

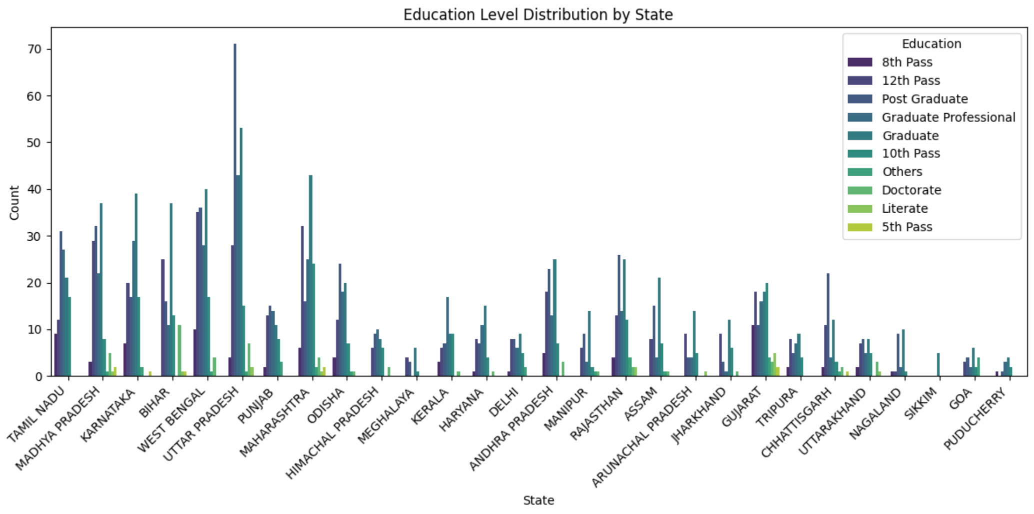

Figure 4: Education Level of candidates of each state

4. Plot of Education Level of candidates wrt type of constituency they are contesting in The pie charts below show that the distribution of education levels in different types of constituencies is approximately the same. So differentiating among these features might not be helpful in improving F1 score.

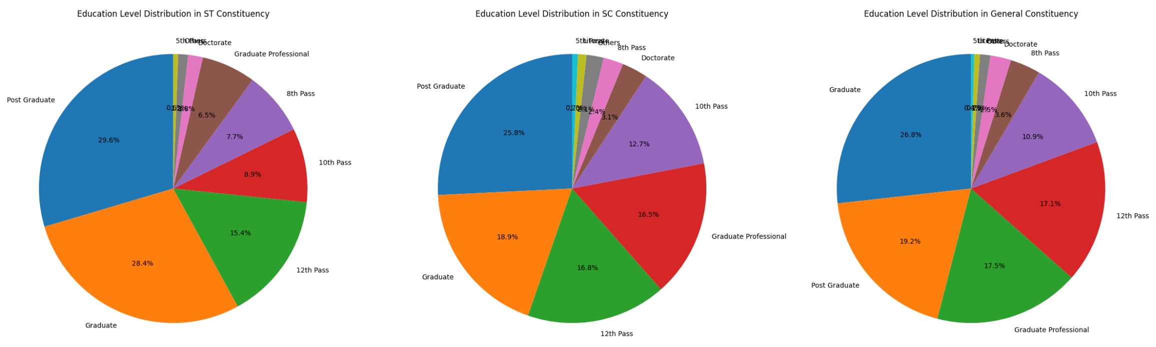

Figure 5: Education Level of candidates with respect to type of constituency: i) ST constituency ii) SC constituency iii) General Constituency

5. Plot of Education Level versus the number of criminal cases against candidate
The plot below shows that the less literate (5th Pass, 8th Pass) candidates have higher number of criminal cases than the more literate ones.

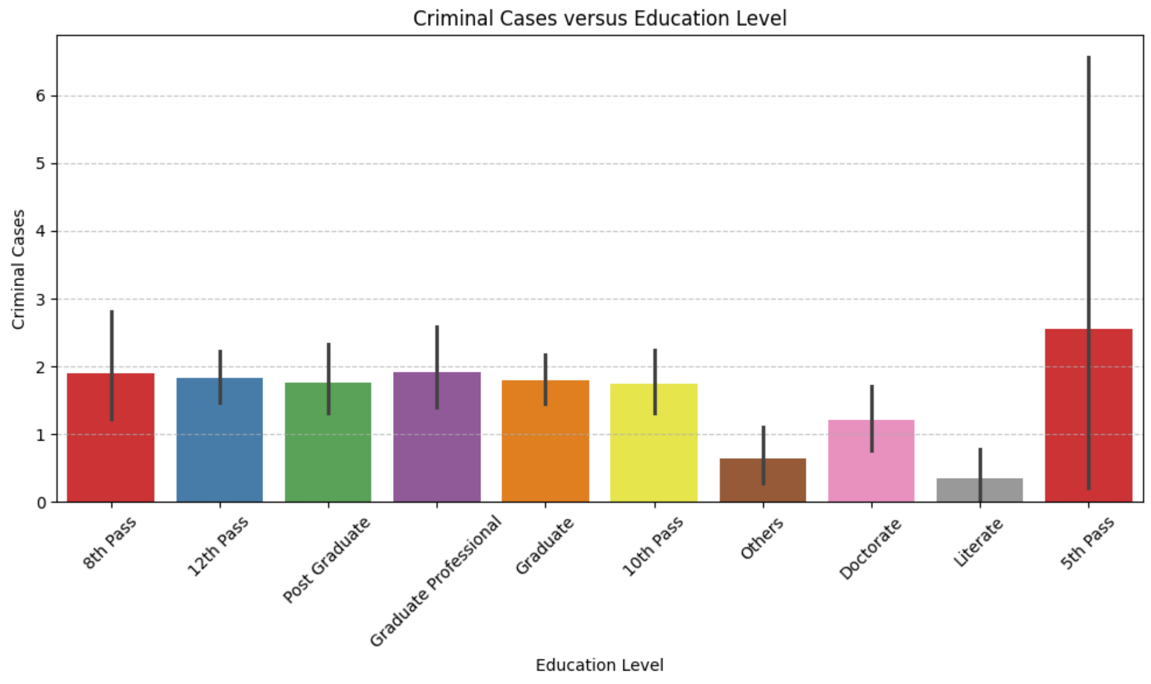

Figure 6: Education Level of candidates with varying number of criminal cases

2.3 Key Insights

- If a candidate's name includes **"Adv." or "Dr."**, we can be more confident about their education level. Advocates mostly had graduate-level education, while doctors typically had postgraduate or higher qualifications. So the columns *'Advoacte'* and *'Doctor'* play a significant role in making predictions.
- The grouping of constituencies into SC, ST and General did not correlate with the education level of those candidates. So this grouping along with the column Constituency were dropped to reduce dimensions.
- Education level had a significant correlation with states. Some states have more people educated than others. So this was an important feature.
- Grouping parties as National or Regional did not make the F1 score better. This could be because a candidate's education level might not be influenced by whether the party they are in is National or Regional.
- Typically the size of training dataset is many time larger than the test data set. Since we had the train set just about 1.5 times bigger than the test set, oversampling using SMOTE in desired proportions helped the models to learn better.
- Out of the various models I had tried, **BernoulliNB** and **DecisionTreeClassifier** resulted in the best performances. The GridSearchCV technique was used to reduce overfitting.

3 Results

1. **My Final F1 Score:** 0.26365 (Public), 0.26463 (Private)
2. **My Public Leaderboard Rank:** 21
3. **My Private Leaderboard Rank:** 7
   (excluding the invalid/deleted accounts)

4 References

- **For Pre-processing and other statistical analysis:**
  - *LabelEncoder*: `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html`
  - *train_test_split*: `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html`
  - *classification_report*: `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html`
  - *confusion_matrix*: `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.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`
  - *scipy.stats*: `https://docs.scipy.org/doc/scipy/reference/stats.html`
  - *MinMaxScaler*: `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html`
  - *StandardScaler*: `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html`

- **numpy**: `https://numpy.org/doc/stable/`
- **pandas**: `https://pandas.pydata.org/docs/`
- **For plotting:**
  - **matplotlib**: `https://matplotlib.org/stable/contents.html`
  - **seaborn**: `https://seaborn.pydata.org/tutorial.html`
- **SMOTE**: `https://imbalanced-learn.org/stable/over_sampling.html`
- **GridSearchCV**: `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html`
- **Models used:**
  - *BernoulliNB Classifier*: `https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.BernoulliNB.html`
  - *Decision Tree Classifier*: `https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html`
  - *Random Forest Classifier*: `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`
  - *K Nearest Neighbors Classifie r(KNN)*: `https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html`
  - *Support Vector Classifier (SVC)*: `https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html`
  - *Gradient Boosting Classifier*: `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html`