Python Assignment Report

Ruthvik Tunuguntla

April 2024

1 Final Code

The final code can be found in the following GitHub repository: `https://github.com/Ruthvik16/Education-Level-Predictor-of-Electoral-Candidates.git`

2 Methodology

2.1 Data Preprocessing Steps

Data preprocessing was essential to ensure the dataset was clean, formatted correctly, and suitable for machine learning model training:

- **Dataset Loading**:
  - Loaded the training and test datasets (`train.csv` and `test.csv`) into Pandas DataFrames.
- **Column Removal**:
  - Removed irrelevant columns ('ID', 'Candidate', 'Constituency ∇') from both datasets to focus on relevant features.
- **String to Numeric Conversion**:
  - Developed a custom function `preprocess_amount` to convert string-based monetary values ('Total Assets', 'Liabilities') into numerical values, considering units like 'Crore', 'Lac', 'Thou', 'Hund'.
  - Applied this function to preprocess the 'Total Assets' and 'Liabilities' columns in both training and test datasets.

2.2 Feature Engineering

Feature engineering aimed to create new features or transform existing ones to improve model performance and interpretability:

- **Categorical Encoding**:
  - Used one-hot encoding to convert categorical features ('Party', 'state') into numerical format for model compatibility.
- **Feature Scaling**:
  - Employed RobustScaler to scale selected numerical features ('Total Assets', 'Liabilities', 'Criminal Case') to handle outliers and ensure uniformity in feature scales across the dataset.
- **Statistical Binarization**:
  - Calculated statistical properties (mean, standard deviation) of key features ('Criminal Case', 'Total Assets', 'Liabilities') in the training dataset.
  - Created binary indicator features to capture different value ranges (< mean-std, mean-std to mean, mean to mean+std, > mean+std) for each feature.
  - I achieved a higher score using the Bernoulli Naive Bayes Model. To further increase the score, as this model works well on binary feature vectors, I took the step of binarizing the numerical data based on the range they lie in.

2.3 Feature Scaling and Outlier Handling

Feature scaling and outlier handling are crucial steps to prepare the data for machine learning models. In this methodology, I utilized the RobustScaler technique to address outliers and ensure consistent feature scales across the dataset.

- **RobustScaler for Feature Scaling**:
  - RobustScaler was employed to scale selected numerical features ('Total Assets', 'Liabilities', 'Criminal Case'). This scaler is robust to outliers and uses the median and interquartile range (IQR) for normalization.
  - RobustScaler transforms the data by removing the median and scaling the data according to the IQR, making it effective for datasets with outliers.
  - The scaled features ensure that outliers do not disproportionately impact model training and improve the overall robustness of the machine learning algorithm.

By leveraging RobustScaler for feature scaling and outlier handling, I prepared the dataset effectively for model training, enhancing the performance and reliability of the subsequent machine learning tasks.

2.4 Model Used

- Utilized Bernoulli Naive Bayes (BNB) classifier for training.
- Conducted hyperparameter tuning using GridSearchCV with 15-fold cross-validation to optimize BNB's performance.

3 Experiment Details

The table below summarizes the experiment details for different models used in the analysis along with their best hyperparameters and corresponding Public F1 scores on Kaggle:

Table 1: Summary of Model Performances on Kaggle's Public Score

Model Name	**Optimal Hyperparameters**	**Kaggle F1 Score (Public)**
Logistic Regression	C = 0.01 max_iter = 100 penalty = 'l1' tol = 0.001	0.14152
Gaussian Naive Bayes	var_smoothing = 1e-10	0.05750
Decision Tree	criterion = 'gini' max_depth = None min_samples_leaf = 1 min_samples_split = 2	0.19460
Random Forest	max_depth = None min_samples_leaf = 1 min_samples_split = 2 n_estimators = 100	0.20615
K-Nearest Neighbors (KNN)	metric = 'euclidean' n_neighbors = 1000 weights = 'uniform'	0.18599
Bernoulli Naive Bayes	alpha = 5.0 binarize = 0.0 class_prior = None fit_prior = False	0.25470

4 Data Insights

Insights from data analysis and visualization:

- **Graphs and Plots Obtained**:

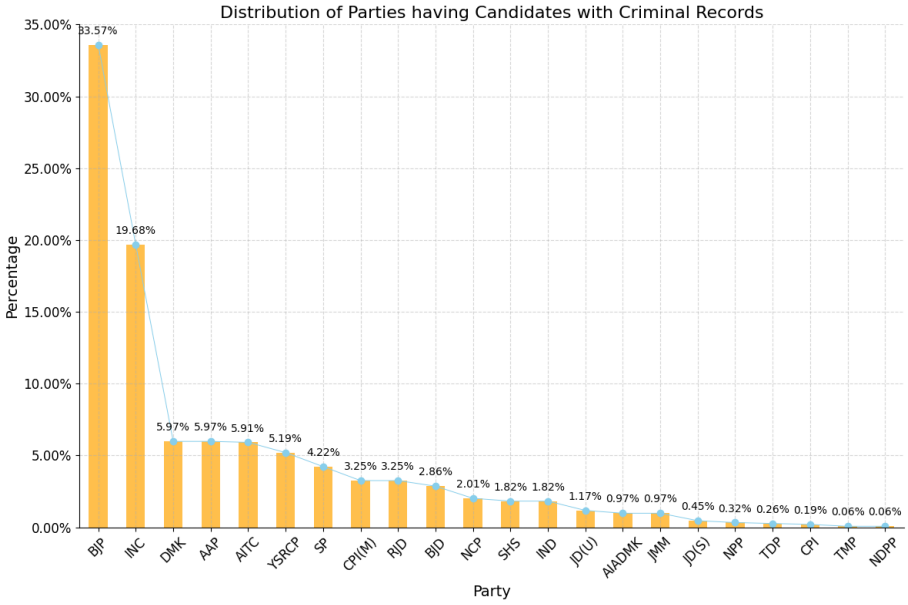

Figure 1: Plots the number of candidates with criminal cases in a Party/ Total number of candidates with criminal cases

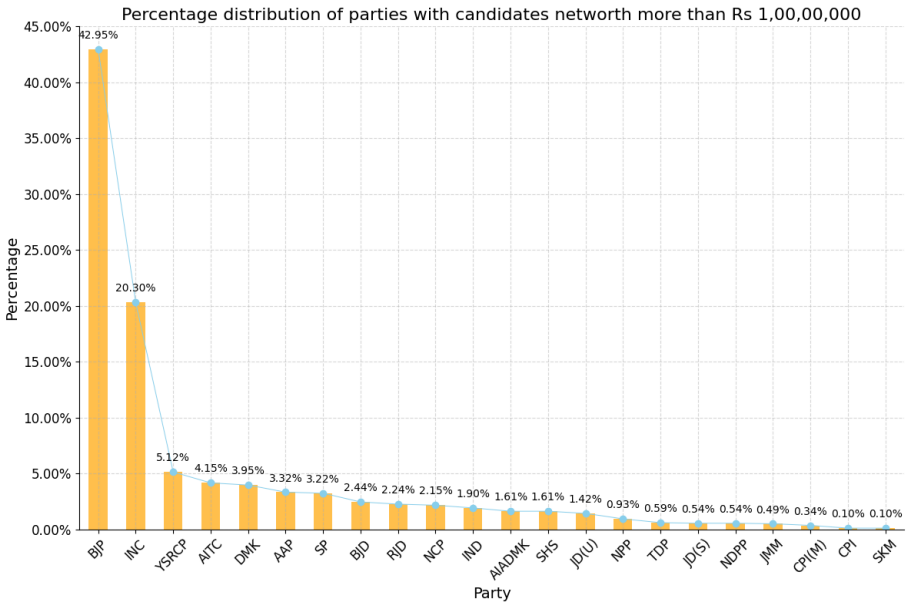

Figure 2: Plots the number of candidates with Net worth (Assets-Liabilities) greater than Rs 1,0,00,0000/ Total number of candidates with Net worth greater than Rs 1,0,00,0000

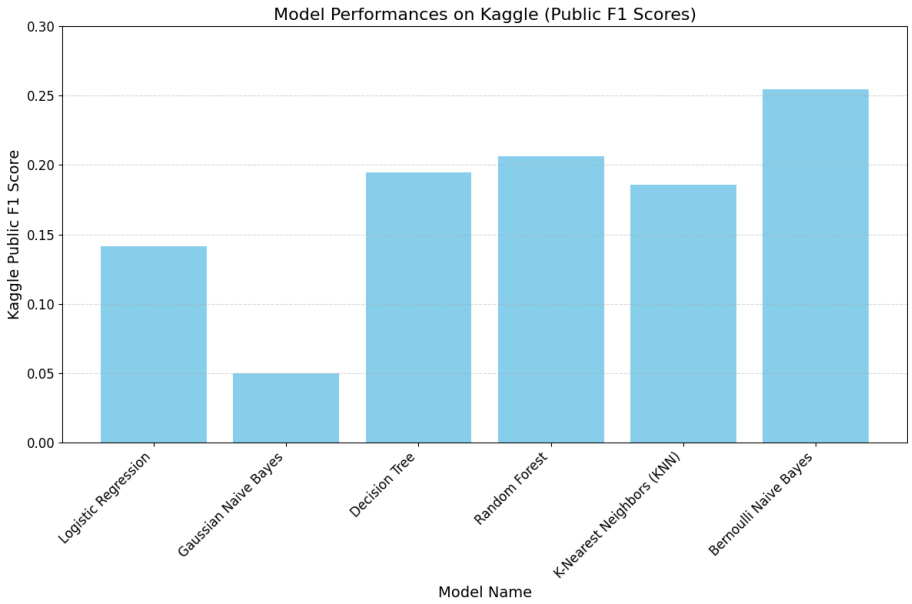

Figure 3: Model vs Public F1 Score on Kaggle

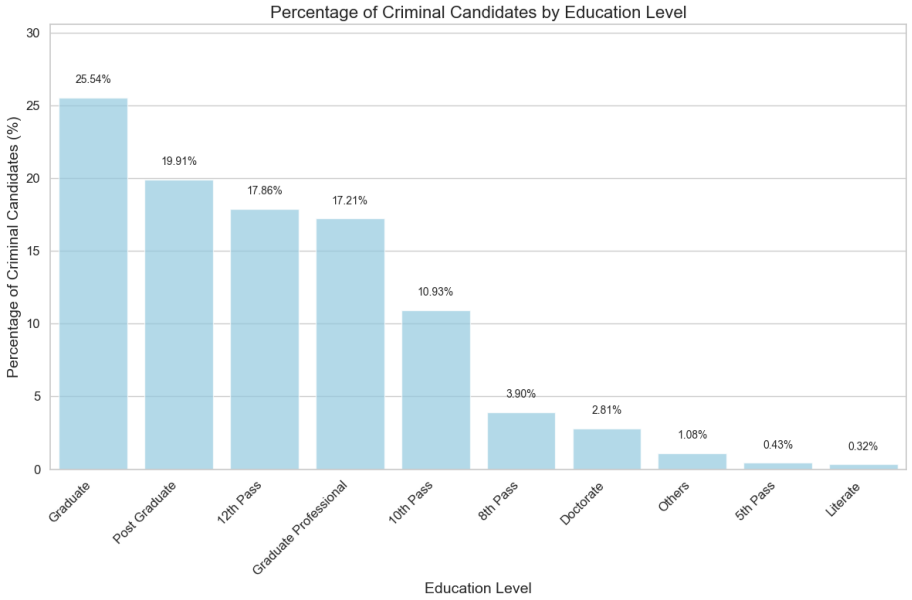

Figure 4: It is surprising to see that the trend in the number of criminal candidates and their education levels

- **Relevant Insights**:
  - Impact of Feature Engineering and Preprocessing on Model Performance:
    - Feature Engineering: Creating new features or transforming existing ones improved model accuracy and interpretability.
    - Preprocessing: Handling missing data and normalizing features optimized dataset quality for stable model predictions.
  - Importance of Scaling and Binarizing Features for Effective Machine Learning:
    - Scaling Features: Scaling the features created uniformity in the dataframe.
    - Binarizing Features: As Bernoulli Naive Bayes model works best on binary feature vectors, Binarizing the data drastically improved the model's performance.

5 Results

- **Final F1 Score**: 0.25470
- **Public Leaderboard Rank**: 47(Includeing the deleted accounts)
- **Private Leaderboard Rank**: NA

6 References

1. Machine Learning Tutorial in Python Playlist by codebasics YouTube Channel
2. Refered for implementation of code for various methods in the below repository `https://github.com/codebasics/py.git`