IIT KANPUR

CS253
SOFTWARE DEVELOPMENT AND OPERATIONS
Python Programming Assignment

Harsh Agrawal

Roll: 220425

April 13, 2024

Contents

1 Code

The code is present at the following github repository link:
https://github.com/Harsh-Agrawal-425/Multi-Class-Classification-CS253

2 Methodology

2.1 Data Preprocessing Steps

Data preprocessing is a crucial step in machine learning projects, as it involves cleaning, transforming, and organizing raw data into a format suitable for analysis and modeling. In this part, I detail the data preprocessing steps undertaken for the given machine learning assignment

- **Data Type Conversion**
  Several columns containing numerical values were initially of string type due to the presence of special characters and units. We converted these columns to a numeric format to facilitate numerical operations. Specifically, columns such as 'Total Assets' and 'Liabilities' were transformed from string representations of numbers to numeric values.

```
df['Total Assets'] = df['Total
    Assets'].astype(str).str.replace('+', '') \
                        .str.replace(' Crore', 'e+7').str.replace('
                            Lac', 'e+5') \
                        .str.replace(' Thou', 'e+3').str.replace('
                            Hund', 'e+2')
df['Total Assets'] = pd.to_numeric(df['Total Assets'],
    errors='coerce')

df['Liabilities'] = df['Liabilities'].astype(str).str.replace('+',
    '') \
                       .str.replace(' Crore', 'e+7').str.replace('
                           Lac', 'e+5') \
                       .str.replace(' Thou', 'e+3').str.replace('
                           Hund', 'e+2')
df['Liabilities'] = pd.to_numeric(df['Liabilities'],
    errors='coerce')
```

- **One-Hot Encoding**
  Categorical variables such as 'Party' and 'State' were one-hot encoded to represent them as binary vectors. This transformation is crucial for machine learning algorithms to properly interpret categorical data.

```
one_hot_encoded_party = pd.get_dummies(df['Party'])
one_hot_encoded_state = pd.get_dummies(df['State'])
df = pd.concat([df, one_hot_encoded_party, one_hot_encoded_state],
    axis=1)
```

- **Handling Zero Values**
  Zero values in numerical columns like 'Total Assets' and 'Liabilities' were replaced with the mean values of those columns. This ensures that zero values do not skew the data and affect the performance of the model.

```
df['Total Assets'].replace(0, df['Total Assets'].mean(),
    inplace=True)
df['Liabilities'].replace(0, df['Liabilities'].mean(), inplace=True)
```

- **Dropping unnecessary columns**
  In the data pre-processing stage, the column "ID" was removed from the dataset. The "ID" column was deemed irrelevant to the task of predicting education levels, as it does not contain any meaningful information related to the target variable.

- **Conclusion**
  Data preprocessing is a critical step in preparing the dataset for machine learning tasks. By cleaning the data, encoding categorical variables, and transforming features into appropriate formats, we ensure that the dataset is structured and suitable for model training. These preprocessing steps lay the foundation for accurate and robust predictive modeling.

2.2 Feature engineering

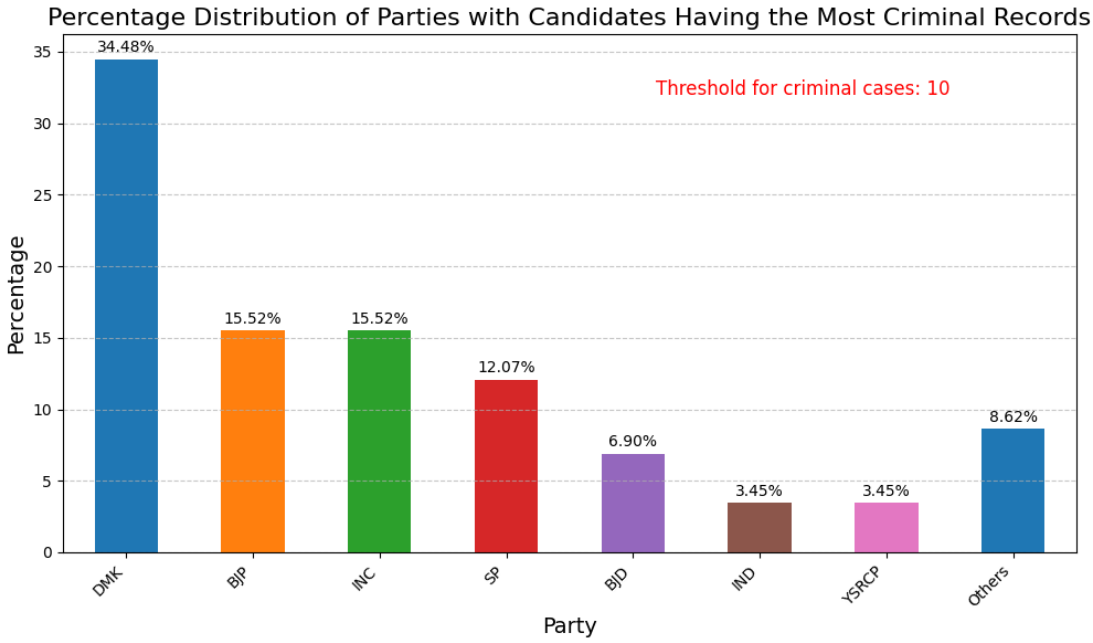
- **Feature Extraction**
  Additional features were extracted from existing columns to capture valuable information. For instance, we extracted the constituency type from the 'Constituency' column and encoded it into numerical values representing different constituency types. Similarly, we identified additional information from candidate names, such as titles like "Dr." or "Adv.", and encoded them into numerical features.

```
# Define function to extract constituency type
def extract_constituency_type(constituency):
    if "(ST)" in constituency:
        return 0
    elif "(SC)" in constituency:
        return 1
    else:
        return 2

# Extract features from 'Constituency' column
df['Constituency_Type'] =
    df['Constituency'].apply(extract_constituency_type)

# Define function to extract extra information
def extract_extra_info(name):
    if "Dr." in name:
        return 1
    elif "Adv." in name:
        return 2
    else:
        return 0

# Extract features from 'Candidate' column
df['Extra'] = df['Candidate'].apply(extract_extra_info)
```

3 Experiment Details

3.1 Graphs and Plots Analysis

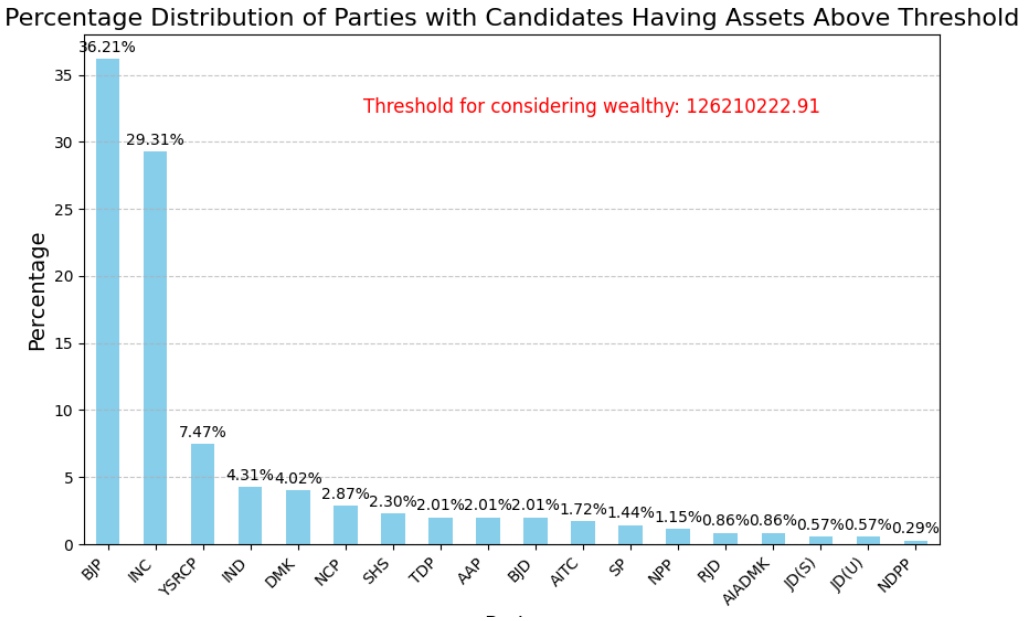
1. **Percentage Distribution of Parties with Candidates Having the Most Criminal Records**
   The first graph illustrates the percentage distribution of parties with candidates having criminal records, with a threshold set at 10 cases. Each bar represents a political party, and its height indicates the percentage of candidates within that party who have criminal records exceeding the threshold. Notably, the top three parties with candidates having the most cases are **DMK, BJP, and INC**, in that order. This visualization offers insights into the prevalence of criminal records among candidates across different political parties.

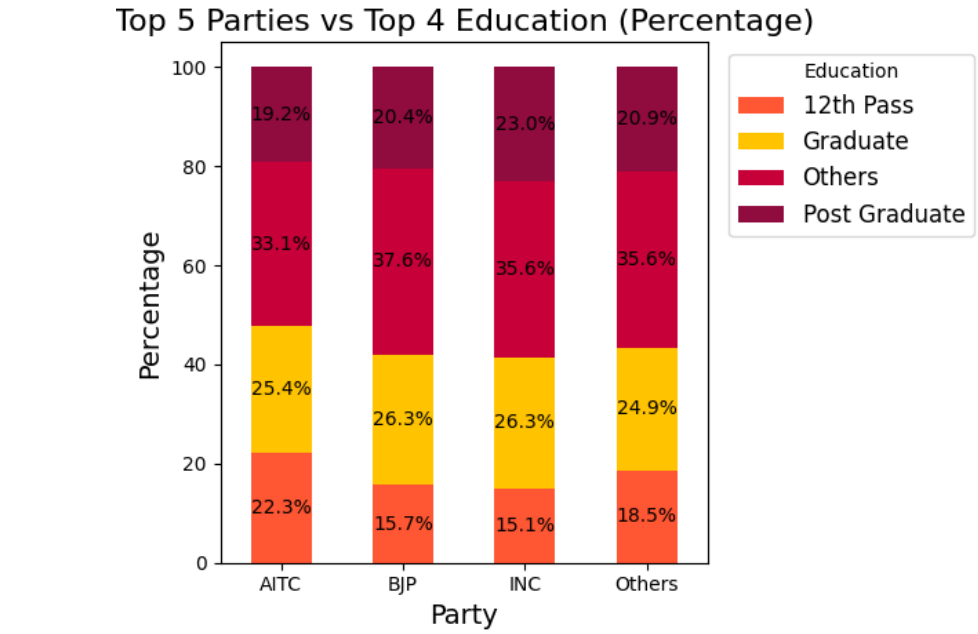


2. **Percentage Distribution of Parties with the Most Wealthy Candidates** The second graph illustrates the percentage distribution of parties with candidates possessing substantial wealth. Like the previous graph, each bar corresponds to a political party, and its height indicates the percentage of candidates within that party with significant financial resources. Notably, the top three parties with the wealthiest candidates are **BJP, INC, and YSRCP.**

3. **Party vs Education** The third graph showcases the relationship between political parties and the education levels of their candidates. Each bar represents a political party, and the stacked bars display the distribution of candidates' education levels within each party. This visualization offers insights into the educational diversity among candidates across different political parties.

4. **Constituency Type vs Education** The fourth graph displays the distribution of education levels among candidates across different constituency types. Each bar represents a


Percentage Distribution of Parties with Candidates Having Assets Above Threshold
Threshold for considering wealthy: 126210222.91
Percentage
36.21%
29.31%
7.47%
4.31%
4.02%
2.87%
2.30%
2.01%
2.01%
2.01%
1.72%
1.44%
1.15%
0.86%
0.86%
0.57%
0.57%
0.29%
BJP
INC
YSRCP
IND
DMK
NCP
SHS
TDP
AAP
BJD
AITC
SP
NPP
RJD
AIADMK
JD(S)
JD(U)
NDPP


Top 5 Parties vs Top 4 Education (Percentage)
Education
12th Pass
Graduate
Others
Post Graduate
Percentage
Party
AITC
BJP
INC
Others
19.2%
33.1%
25.4%
22.3%
20.4%
37.6%
26.3%
15.7%
23.0%
35.6%
26.3%
15.1%
20.9%
35.6%
24.9%
18.5%

constituency type, and the stacked bars depict the distribution of candidates' education levels within each constituency type. This visualization helps to understand candidates' educational background in various constituencies.

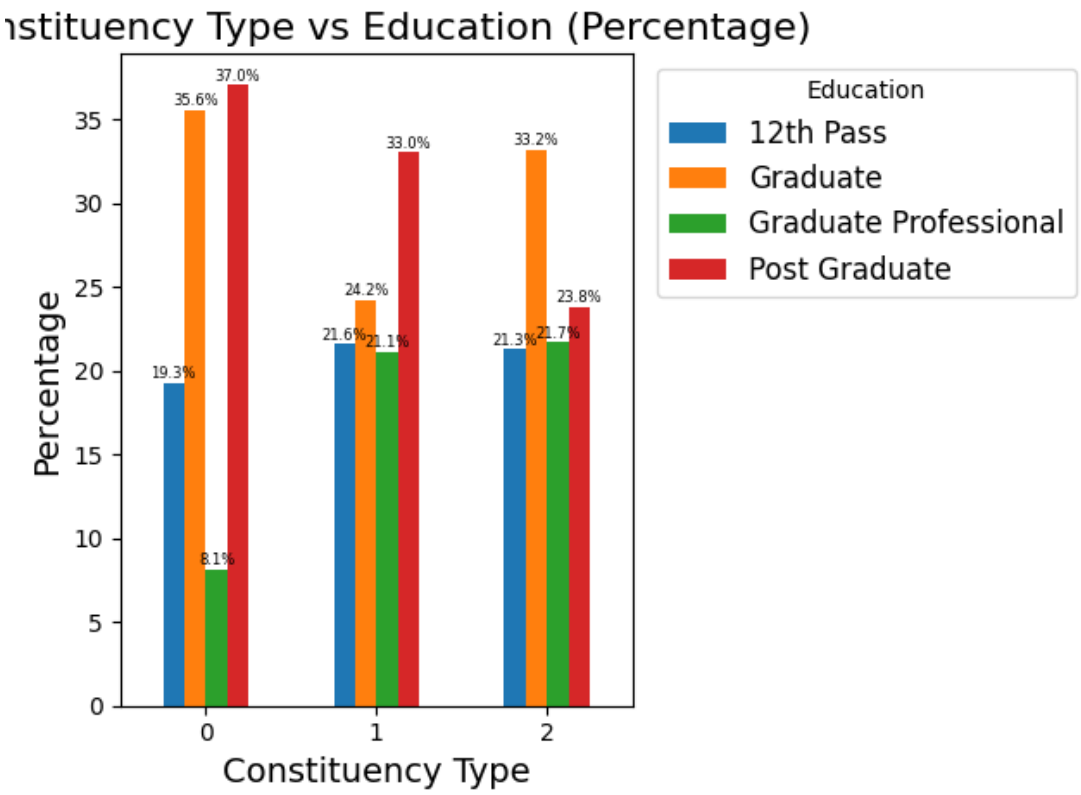


Figure 1: Constituency Type vs Education

3.2 Model Used

I tried using various machine learning models under sci-kit library, including **K-Nearest Neighbors (KNN),random forest, Support Vector Machine and Perceptron**. Each model was evaluated based on its performance, measured using the F1 score metric.
Below is a table summarizing the performance of each model in terms of its F1 score:

Table 1: Model vs F1 Score

Model	**Private F1 Score**
KNN	0.1849
Random Forest	0.23388
SVM	0.10342
Gradient Boosting	0.24652
Perceptron	0.11687

The Final chosen model for this task is the ***Bernoulli Naive Bayes classifier***. Naive Bayes is a probabilistic classification algorithm based on Bayes' theorem with the "naive" assumption of feature independence.
Model Parameters After several trial and error iterations, the following parameters were found to yield optimal performance:

- **Alpha Smoothing Parameter**: The alpha parameter for Laplace smoothing was set to ***0.38***. Laplace smoothing is applied to avoid zero probabilities when encountering unseen features during prediction.
- **Binarization Threshold**: Features were binarized using a threshold of ***0.0***. This threshold determines the cutoff point for binarizing feature values.
- **Fit Prior**: The fit prior parameter was set to ***True***, allowing the model to learn class prior probabilities from the training data.

4 Results

4.1 Final F1 score

After rigorous training and evaluation, my model achieved a final ***Public F1 score of 0.27458 and Private F1 score of 0.28013***

4.2 Public and private Leaderboard Rank

My model secured a ***2nd position*** in the final private leaderboard standings and ***8th position*** in the final public leaderboard standings and

5 References

1. *NumPy.* "NumPy Documentation." : https://numpy.org/doc/
2. *Pandas.* "pandas Documentation." : https://pandas.pydata.org/docs/
3. *Scikit-learn.* "scikit-learn Documentation." : https://scikit-learn.org/stable/documentation.html

4. *Matplotlib*. "Matplotlib Documentation." : https://matplotlib.org/stable/contents.html

5. *CS253 Course Videos* uploaded on HelloIITK platform.