Python Assignment Report

Himanshu Shekhar

April 2024

1 Methodology

1.1 Data Preprocessing

- Dropped the **ID**, **Candidate** and **Constituency** ▽ columns from both the training and test datasets `df.drop()` command. The **ID** and **Candidate** column obviously would not contribute to the **Education**, therefore were removed. The training data contained 2059 rows and the number of unique constituencies were 2037, therefore Constituency of a Candidate is almost unique to him/her, so it would not contribute to the model and was removed.
- Converted **Total Assets** and **Liabilities** to Numerical Values. Upon observation, the Total Assets column had categories *Crore+, Lac+* and *Thou+* and the Liabilities column had categories *Crore+, Lac+, Thou+* and *Hund+*.
- One-Hot Encoded **state** and **Party** column using the `pd.get_dummies()` command, as it led to a higher **f1_score** than what was achieved when these columns were Label Encoded. Also the individual correlation of some states and parties increased than what was achieved originally by state and party on label encoding.
- **LabelEncoded** the Education column using `sklearn.preprocessing.LabelEncoder` to obtain an integer value for each Education label which is easier to handle than the string itself.

1.2 Identifying Outliers

- Plotted a scatterplot for `Total Assets vs Education` and observed that Total Assets is mostly concentrated at values less than $4e9$. Therefore, dropped the rows where the value was greater than $4e9$.

- Plotted a scatterplot for `Liabilities vs Education` and observed that Liabilities is mostly concentrated at values less than $4e9$. Therefore, dropped the rows where the value was greater than $4e9$.
- Plotted a scatterplot for `Criminal Cases vs Education` and observed that Criminal Cases is mostly concentrated at values less than 60. Therefore, dropped the rows where the value was greater than 60.

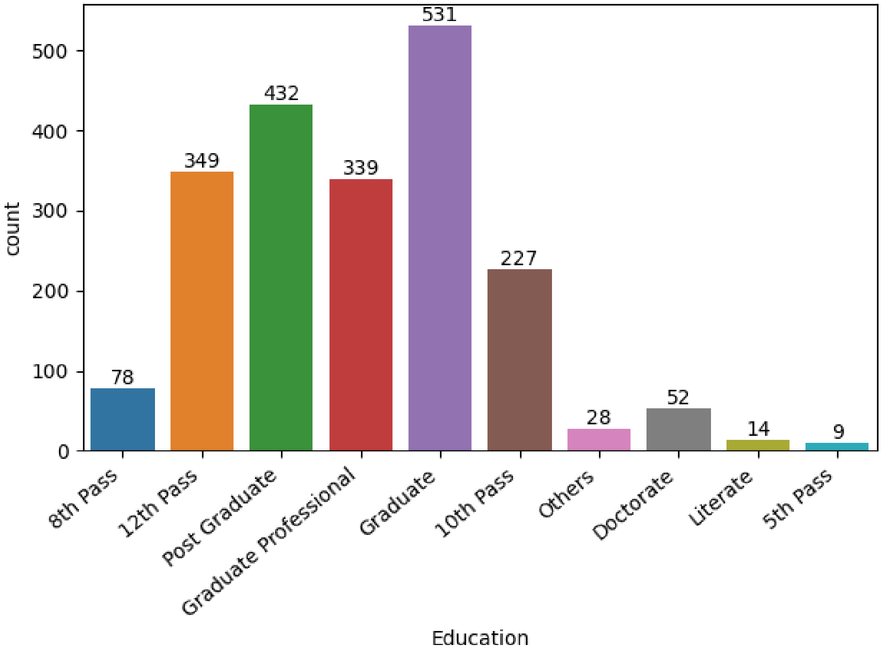
1.3 Transformation

- Used `sklearn.preprocessing.StandardScaler` to standardize the features to have mean 0 and variance 1. This is crucial for algorithms like **KNN** that are sensitive to the scale of the data.

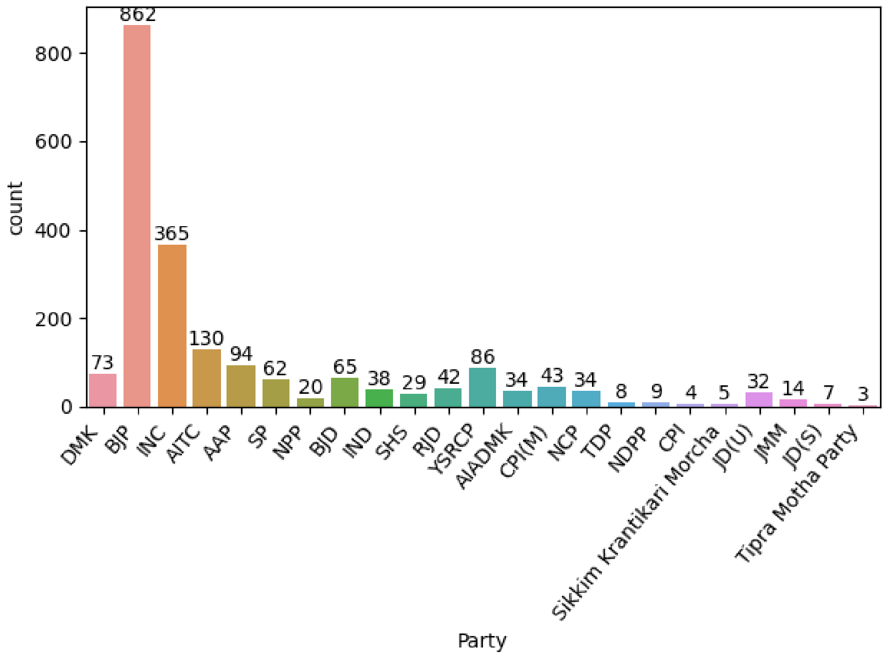
2 Experiment Details

2.1 Data Insights

- Count of Education Labels:



- Count of Party Labels:

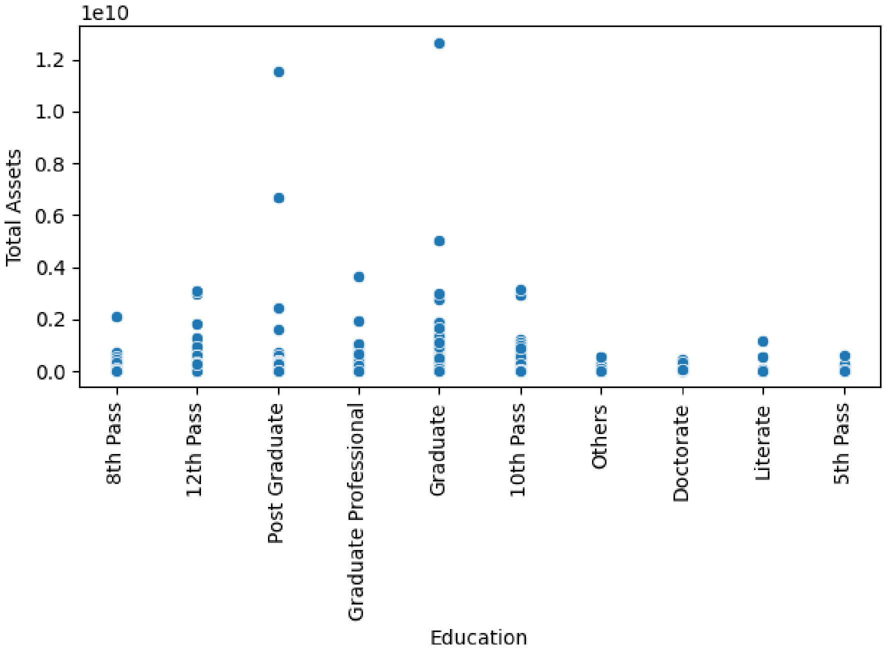


- Count of Constituency ▽ Labels: The number of unique Constituencies were 2037 therefore it was not possible to plot it. I used the `df.value_counts().to_frame()` command to see a description for count and the number of unique Constituencies.

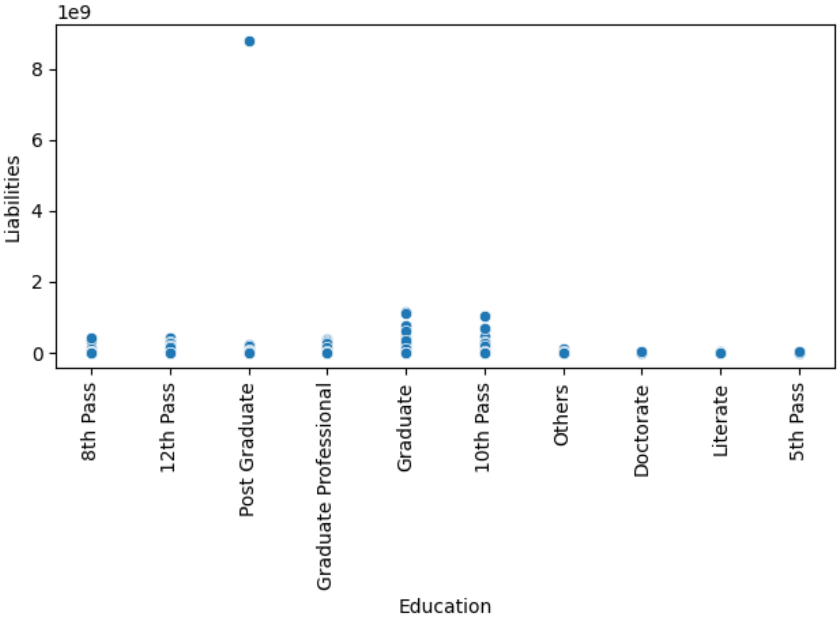
- Checking for **NaN** values: I used the `df.info()` command to check for NaN values and the data types of all the columns. There were **NO** values that were **NaN**.

2.2 Plots

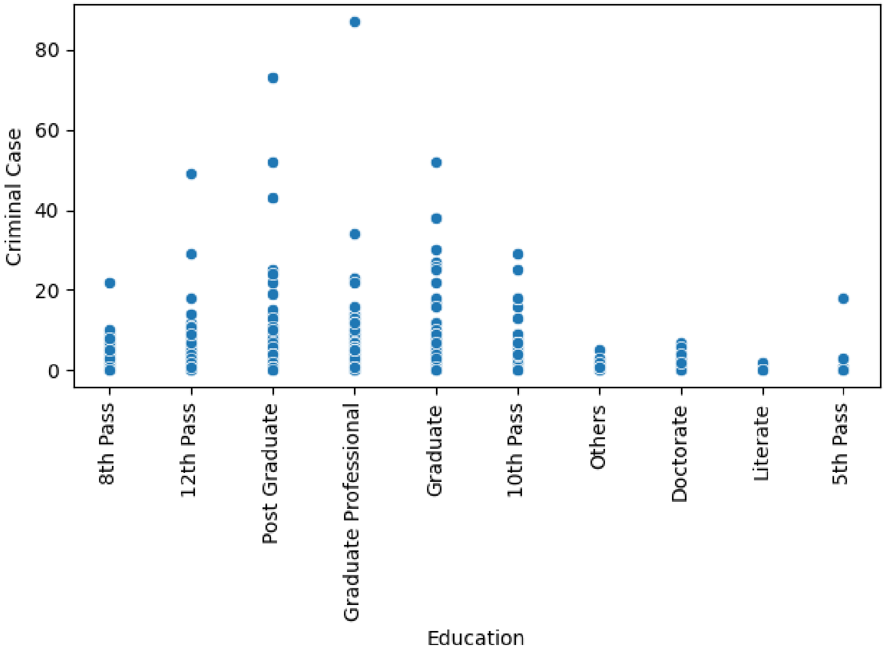
- Total Assets vs Education:



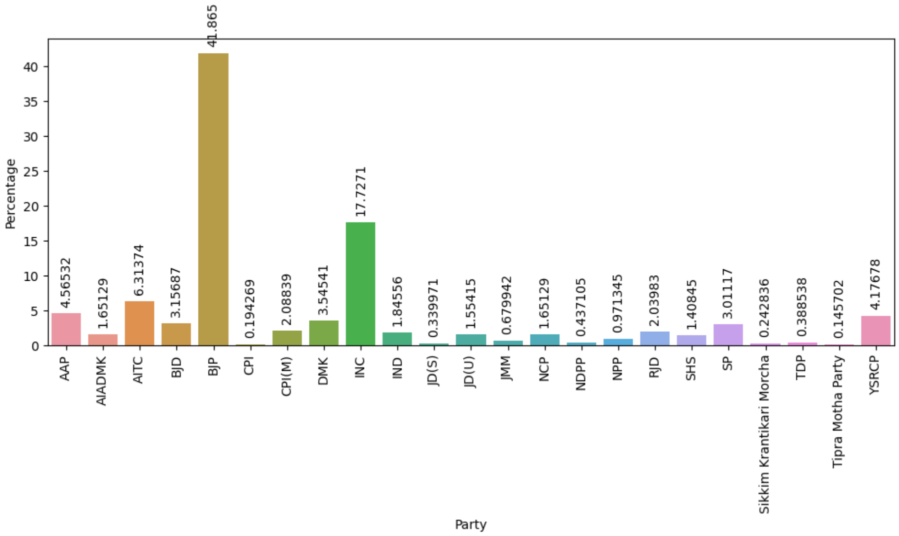
- Liabilities vs Education:



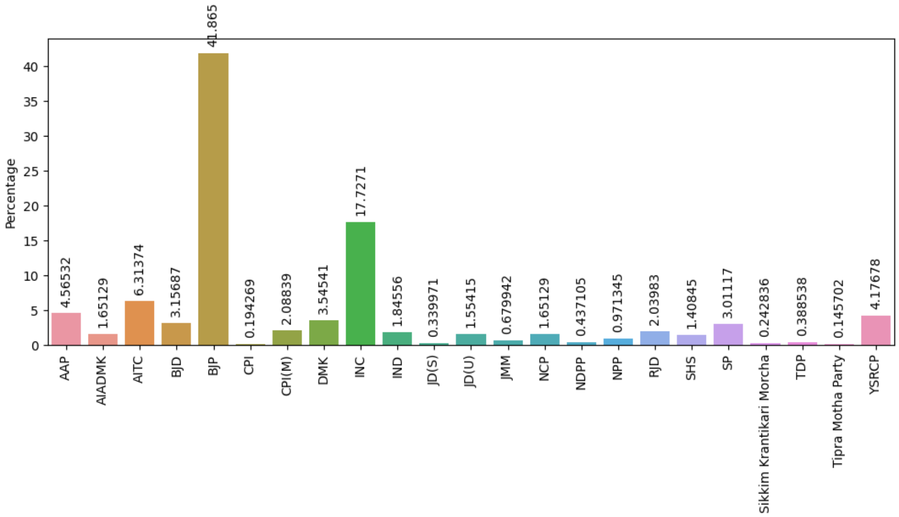
- Criminal Cases vs Education:



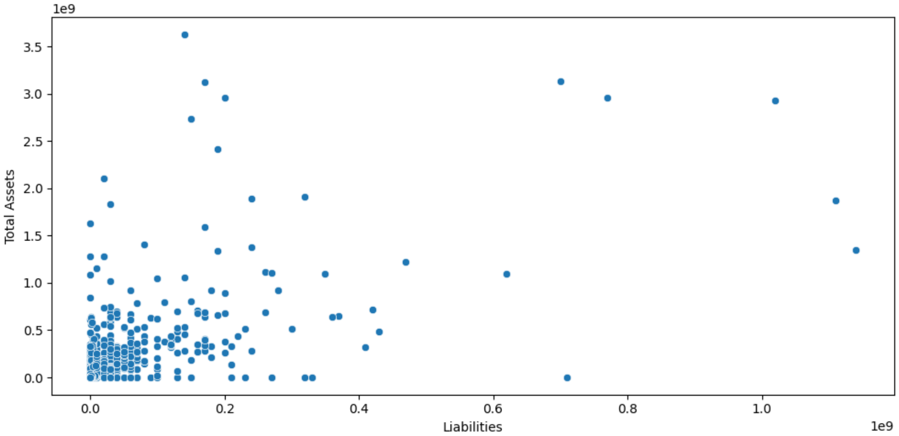
- Percentage Distribution of Parties with Candidates having the most Criminal Records:



- Percentage Distribution of Parties with the most Wealthy Candidates:



- Total Assets vs Liabilities:



2.3 Models

Model	Hyperparameters	Other Details
Train-Test Split	test_size=0.3, random_state=42	Split the data into training and testing sets for cross validation
KNN	n_neighbors=17	Trained on the training set

- The number of n_neighbors was found using a `for` loop where the values were varied from 1 to 29 (inclusive) and the one with the best score was taken.

- I thought of using KNN because on applying the **LazyPredict** library the best `f1_score` was of KNN.
- I tried to find the best hyparameters using the library **Optuna**, but on doing so the final `f1_score` decreased.

3 Results

- **Final F1 Score: 0.13736**
- **Public Leaderboard Rank: 65**
- **Public Leaderboard F1 Score: 0.24898**
- **Private Leaderboard Rank: 103**
- **Private Leaderboard F1 Score: 0.23465**

4 References

1. Precision, Recall, & F1 Score Intuitively Explained: https://www.youtube.com/watch?v=8d3JbbSj-I8
2. Numpy Absolute Beginner's Guide: https://numpy.org/doc/stable/user/absolute_beginners.html
3. Pandas User Guide: https://pandas.pydata.org/docs/user_guide/index.html
4. Matplotlib User Guide: https://matplotlib.org/stable/users/index
5. Seaborn User Guide and Tutorial: https://seaborn.pydata.org/tutorial.html
6. Sklearn Preprocessing Data Library: https://scikit-learn.org/stable/modules/preprocessing.html
7. Sklearn Label Encoder: Click Here!
8. Sklearn Metric and Scoring: https://scikit-learn.org/stable/modules/model_evaluation.html#classificatio metrics
9. Sklearn Model Selection: Click Here!
10. LazyPredict: https://lazypredict.readthedocs.io/en/latest/usage.html
11. KNearestNeighbors Algorithm: https://youtu.be/wTF6vzS9fy4?feature=shared
12. Random Forest Algorithm: https://youtu.be/J4Wdy0Wc_xQ?feature=shared
13. Optuna: https://optuna.org/#code_examples

5 Github Repository

- https://github.com/HimanS1708/CS253_Assignment_3