CS253 - Python Programming

Machine Learning (Ass3)

1 Abstract

This assignment offered a fascinating opportunity to delve into the intricacies of elections within the world's largest democracy. My objective in this report is to forecast the Education level of winners in State and Union Territory (UT) elections across India, leveraging a dataset encompassing recent election outcomes. The main motive was to implement a multi-class classification model in which different factors are used to determine the candidates' educational backgrounds. Alongside meticulous preprocessing and data insights, key components of my methodology include feature engineering techniques, thoughtful model selection, and the utilization of suitable evaluation metrics.

2 Methodology

In this section, we delve into the analysis of the provided dataset to gain insights into the characteristics and patterns of the data provided. We analyse the distribution of various features, identify any missing values or outliers, and assess the correlation between different variables. Moreover, we elucidate the process of data transformation, ensuring alignment with our model's preferences and visualise them through distinct plots. By understanding the underlying structure of the data, we lay the foundation for building effective machine-learning models.

2.1 Data-Preprocessing

- **Handling Categorical Columns:** With only two numeric features present in the dataset, the majority being categorical variables, I prioritized converting as many features into numeric format as possible. This involved transforming the **Total Assets** and **Liabilities** columns into numerical values by removing the **Crore+**, **Lac+**, **Thou+**, and **Hund+** string components from the feature elements. This initial step aimed to enhance compatibility with machine learning models, nonetheless, it's worth noting that certain models, such as RandomForest, demonstrate adeptness in handling both categorical and numerical features.

(2059, 9)

	ID	Candidate	Constituency ∇	Party	Criminal Case	Total Assets	Liabilities	state	Education
0	0	M.K. Mohan	ANNA NAGAR	DMK	4	211 Crore+	2 Crore+	TAMIL NADU	8th Pass
1	1	Khatik Ramesh Prasad	KARERA (SC)	BJP	0	1 Crore+	0	MADHYA PRADESH	12th Pass
2	2	Dr. Mantar Gowda	MADIKERI	INC	0	7 Crore+	22 Lac+	KARNATAKA	Post Graduate
3	3	Kundan Kumar	BEGUSARAI	BJP	0	9 Crore+	24 Lac+	BIHAR	Post Graduate
4	4	Swapan Majumder	BANGAON DAKSHIN (SC)	BJP	2	2 Crore+	61 Lac+	WEST BENGAL	8th Pass

- **Encoding:** I employed two widely-used encoding techniques: Label Encoding for the target variable and One-hot encoding for the **Party** and **state** features. This approach ensured that the 'Education' column contained integer values, facilitating a one-to-one correspondence between education types and newly encoded integers, typically whole numbers. Furthermore, this encoding strategy allowed for straightforward inverse transformations, facilitating conversion back to the original format after model predictions. Conversely, One-hot encoding created distinct columns for each unique value in the encoded feature, representing these values as True/False entries in the newly created columns. This methodology proved particularly advantageous for my final model, which requires input data to be in binary format.

Train Data: (2059, 58) Test Data: (1374, 57)

	ID	Candidate	Constituency ∇	Criminal Case	Total Assets	Liabilities	Party_AAP	Party_AIADMK	Party_AITC	Party_BJD	...	state_OD
0	0	Geeta Bharat Jain	MEERA BHAYANDAR	2	70 Crore+	11 Crore+	False	False	False	False	...	
1	1	Becharam Manna	SINGUR	1	2 Crore+	13 Lac+	False	False	True	False	...	
2	2	Sunil Vijay Tingre	VADGAON SHERI	3	49 Crore+	1 Crore+	False	False	False	False	...	
3	3	Asit Mazumder (Tapan)	CHUNCHURA	1	2 Crore+	0	False	False	True	False	...	
4	4	Hriday Narayan Singh Patel	SAGRI	0	16 Crore+	2 Crore+	False	False	False	False	...	

5 rows × 57 columns

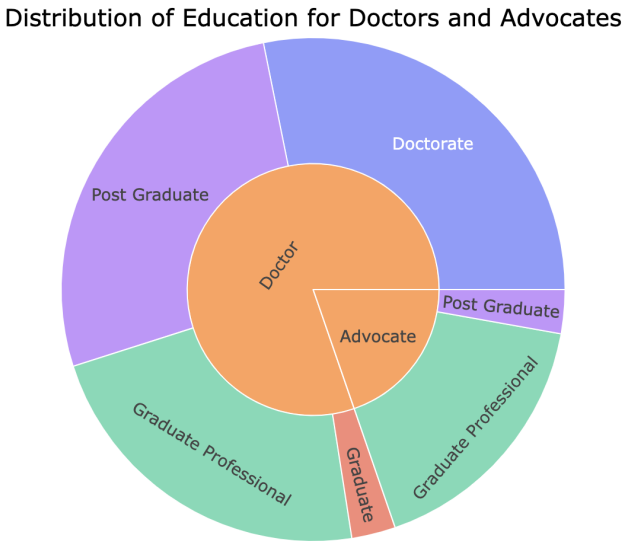
2.2 Feature Engineering and Selection

Firstly, we employ the correlation coefficient to learn information about the relationship between each feature and our target variable. This coefficient, calculated using the following formula, gives us an overall idea of all our features.

$$r(X,Y) = \frac{\sum(X - X_{mean})(Y - Y_{mean})}{\sqrt{\sum(X - X_{mean})^2 \sum(Y - Y_{mean})^2}}$$

- Maximal Correlation: **Total Assets** and **Liabilities**: As depicted in the correlation matrix heatmap, a significant correlation is observed between Total Assets and Liabilities. Consequently, to mitigate the presence of redundant features and reduce the risk of overfitting in any model, I introduce a new column named **Aggregate_Assets** = **Total Assets** - **Liabilities**.

- **Candidate Patterns:** While analyzing the Candidate column, I observed that some names began with **Dr.** or **Adv.**, indicating potential patterns in their education distribution. Subsequently, I created two new features, **Is_Doctor** and **Is_Advocate**, to capture these patterns and incorporate them into the dataset. These new features are Binary features having just 1/0 values.



- **Constituency Patterns:** Perfectly similar to the above pattern further examination of the Constituency column revealed additional patterns, such as the presence of **(SC)** or **(ST)** at the end of some constituency names. To encode these distinctions, I created a new feature called **Constituency_type**, assigning values of 1, 2, and 0 for SC, ST, or other cases, respectively.

2.3 Outliers

These are data points that significantly deviate from the rest of the dataset, often indicating unusual or erroneous observations. In the context of election commission data, outliers may manifest in features like Total Assets, Liabilities, or Criminal Cases, suggesting candidates with extreme financial or legal circumstances. Detecting and removing these is a necessary step to avoid models from being mislead.

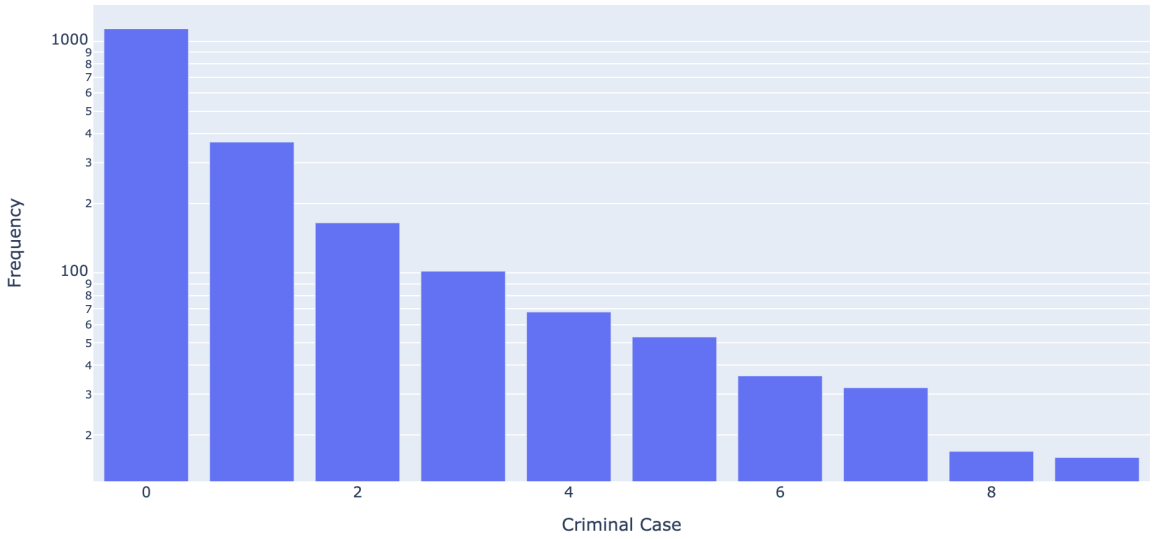


Figure 1: **NOTE**: Here the frequency is plotted in logarithmic scale to visualise the outliers.

2.4 Dimensionality Reduction

After leveraging features like **Candidate**, **Constituency** to derive new, more significant predictors for the education levels of individuals, the columns for Candidate and Constituency hold little value. With each element in these features now possessing unique values, along with the ID column which merely represents indexes, they were deemed redundant and subsequently dropped.

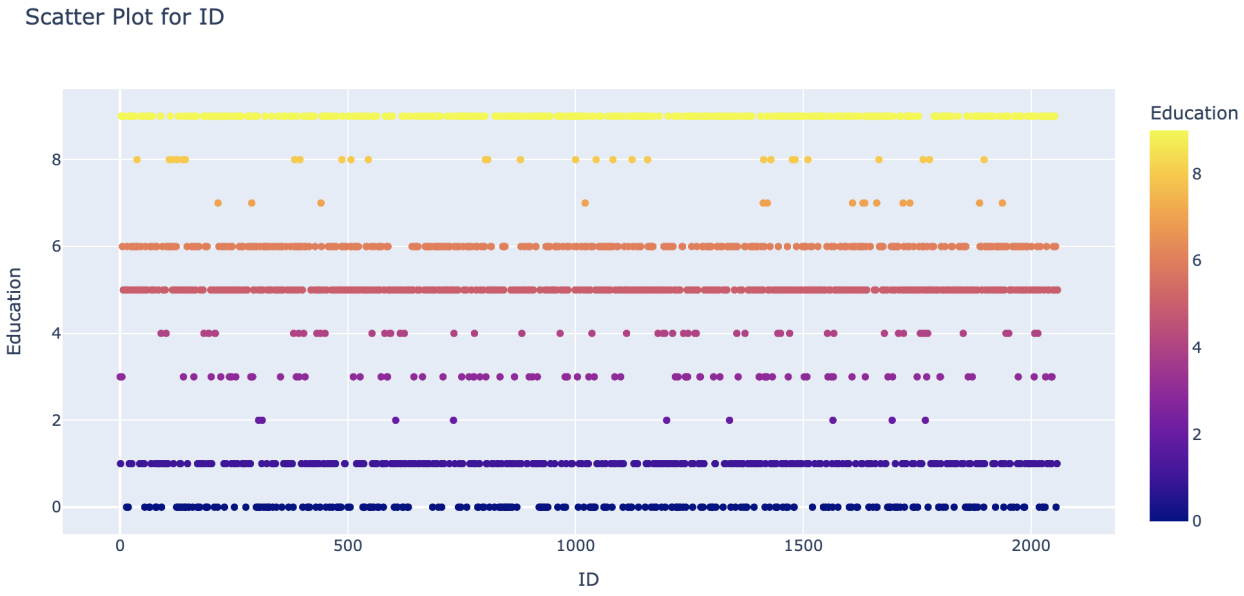


Figure 2: Highlighting the redundancy of the ID feature.

2.5 Normalization & Scaling

- **Experimentation with Scaling Techniques:**
  - Explored **StandardScaler** and **MinMaxScaler** from the `sklearn` library.
  - Primarily implemented during *Random Forests* and *Decision Trees* model development.
- **Limited Impact on Final Model:**
  - Despite scaling attempts, the final model choice, *Bernoulli Naive Bayes (BernoulliNB)* classifier, remained unaffected.
  - Several factors influenced this decision.

  Bernoulli Naive Bayes (BNB) is a classification algorithm that operates on binary features, where each feature represents a binary occurrence (e.g., presence or absence). Unlike some other machine learning algorithms, such as Support Vector Classifier's (SVCs) or K-Nearest Neighbors (KNN), BNB does not require feature scaling. The reason for this lies in how BNB calculates probabilities. BNB estimates probabilities based on counts of feature occurrences in each class. Since BNB only considers whether a feature is **present or absent**, the magnitude of the feature values does not affect the probability calculations. Therefore, scaling the features does not impact the performance of BNB.
- **Reasoning for Omitting Scaling:**
  - **Model Characteristics:** *BernoulliNB* is inherently robust to feature magnitude variations, given its binary feature modeling.
  - **Dataset Attributes:** No significant feature magnitude discrepancies observed that would necessitate scaling.
  - **Feature Nature:** Binary or categorical features might not benefit from scaling, preserving their interpretability.
  - **Model Simplicity:** *BernoulliNB*'s simplicity and efficiency make it less sensitive to feature scaling compared to more complex models.
- **Conclusion:** Considering dataset attributes and model characteristics, scaling was omitted in favor of preserving model interpretability and efficiency.

3 Model Selection

Model	Hyperparameters + Default	Validation Score	Validation Metric/Strategy
SVC	max_depth = 25	14%	f1 score
KNN	leaf_size = 10, n_neighbors = 6, p = 1	18.5%	f1 score
DT	max_depth = 25	21.46%	f1 score
RF	n_estimators=85, max_depth=29, min_samples_split=7	23.99%	f1 score
BNB	alpha=0.5, binarize=0.1	28.05%	f1 score & kFold

Table 1: Models Implemented and shown above can be found in the GitHub Repository mentioned at last.

- Upon analyzing the task, I opted to utilize **Decision Trees** and **Random Forest** models due to their recognized efficiency in handling multi-label classification tasks. These models are renowned for their robustness and adaptability in scenarios involving multiple labels. Some important reasons for using these models:
  - **Flexibility in Handling Non-Linear Relationships:** Decision Trees and Random Forests are capable of capturing non-linear relationships between features and labels, making them suitable for complex classification tasks where the relationships may not be linear.

- **Ensemble Learning for Improved Accuracy:** Random Forests, being an ensemble learning method, combine the predictions of multiple individual decision trees to produce a more accurate and stable prediction. This ensemble approach helps in reducing overfitting and improving generalization performance, especially in multi-label classification scenarios with diverse and overlapping classes.

Reasons for SVC's and KNN's Failure

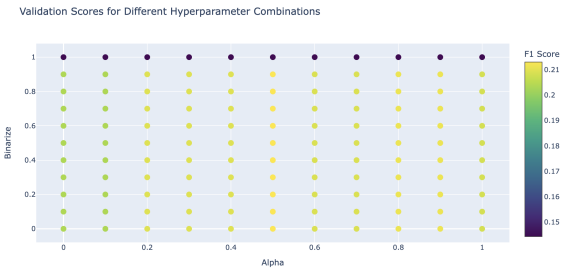
- Furthermore, I experimented with alternative models such as Support Vector Classifiers (SVC) and K-Nearest Neighbors (KNN), but they did not yield successful outcomes compared to my efforts with Random Forests.
  - **Non-Linear Separability:** SVCs and KNNs perform optimally when the data is well-separated and exhibits clear boundaries between classes. However, in our case, the data was highly scattered and lacked distinct class boundaries, making it challenging for these models to effectively separate the classes. Also, had the data been balanced, SVC/KNN would have performed better.
  - **Presence of Noise:** SVCs and KNNs are susceptible to noise and outliers in the data, which can significantly impact their performance. With scattered data and no clear correlation between features, the presence of noise may have interfered with the decision boundaries learned by these models, leading to suboptimal results.
- After realizing the absence of correlation within my dataset, I opted for one of the Naive Bayes (NB) models, known for their efficacy in handling uncorrelated data. However, given the availability of three NB variants – Bernoulli, Multinomial, and Gaussian – I needed to choose the most suitable one. Among them, Bernoulli Naive Bayes emerged as the ideal choice for our dataset, characterized by binary features resulting from pre-processing and feature engineering. Applying this model yielded a higher score compared to previous attempts, indicating its compatibility with our data.

Use Cases of Different NB Models

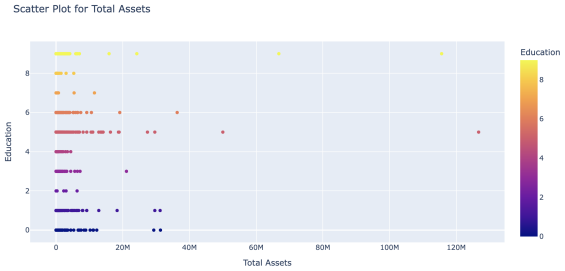
  - **Bernoulli Naive Bayes:**
    - Utilized when the dataset comprises binary data features, making it suitable for scenarios where features are represented as binary variables.
    - Particularly effective in cases where pre-processing and feature engineering result in binary-type data, as it can seamlessly handle such features.
  - **Multinomial Naive Bayes:**
    - Preferred when the focus is on counting occurrences or when dealing with discrete data, making it suitable for datasets where features represent counts or frequencies.
    - Commonly employed in text classification tasks, such as document categorization or sentiment analysis, where the frequency of words or tokens is essential.
  - **Gaussian Naive Bayes:**
    - Applied when the dataset contains continuous-valued features, as it assumes a Gaussian (normal) distribution of the data.
    - Well-suited for datasets with continuous numerical features, such as sensor data, financial metrics, or physical measurements.

Given the nature of the dataset, it is evident that both the train and test data exhibit minimal correlation among themselves. In such scenarios, where the features demonstrate limited correlation and primarily consist of binary values, the Bernoulli Naive Bayes model stands out as the optimal choice for analysis.

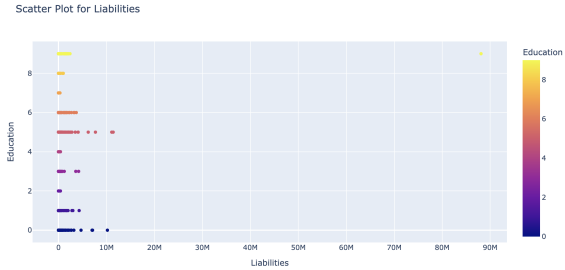
4 Some Important Plots



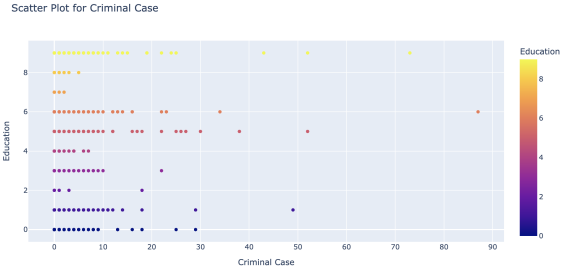
(a) Hyperparameter Variation Scatter Plot



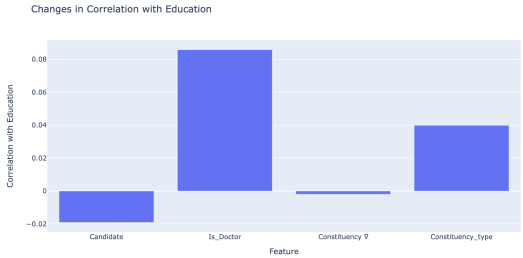
(b) Education v/s Total Assets Scatter Plot



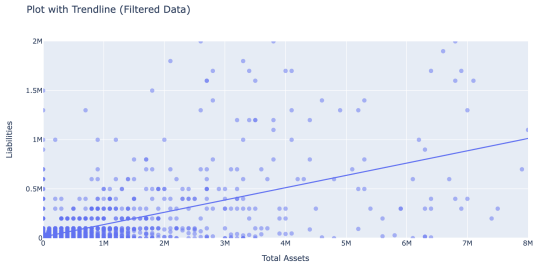
(c) Education v/s Liabilities Scatter Plot



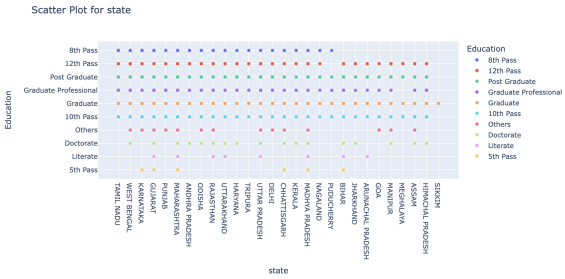
(d) Education v/s Criminal Cases Scatter Plot



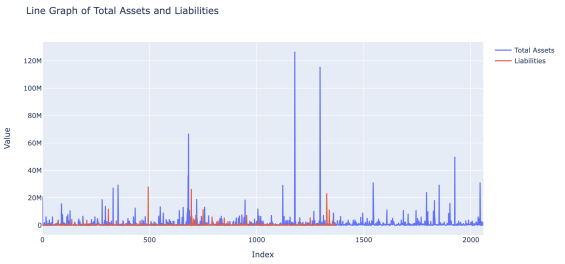
(e) Change in Correlation after Feature Engineering



(f) Liabilities v/s Total Assets Plot

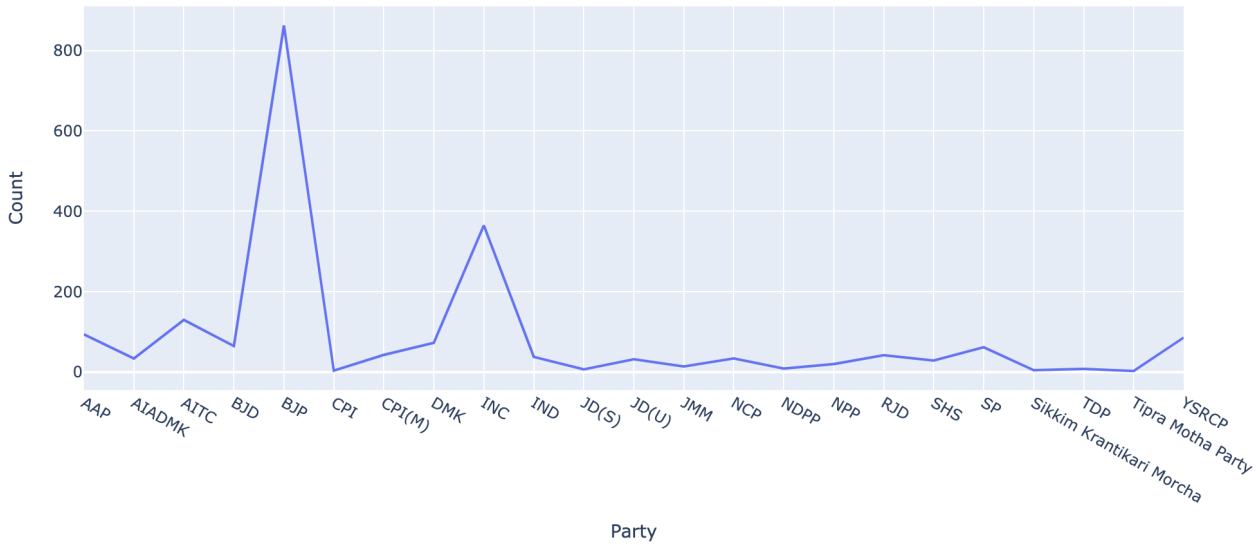


(g) Education Statistics for each State

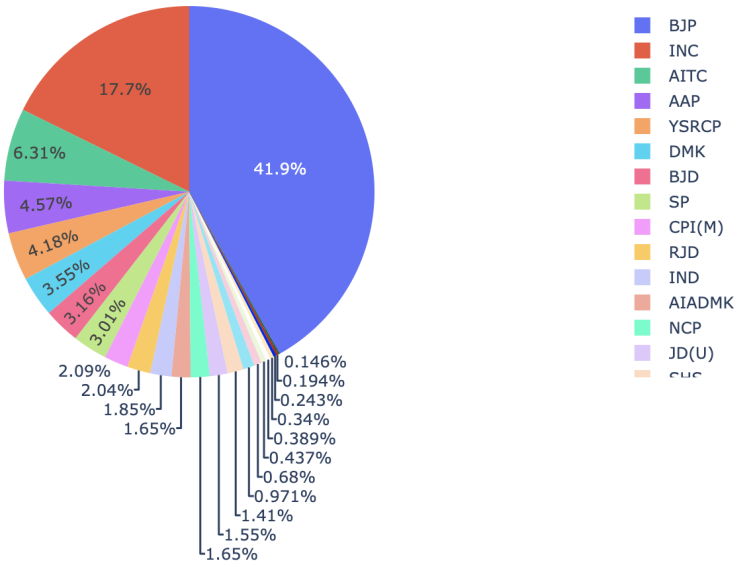


(h) Assets & Liabilities v/s ID Line graph

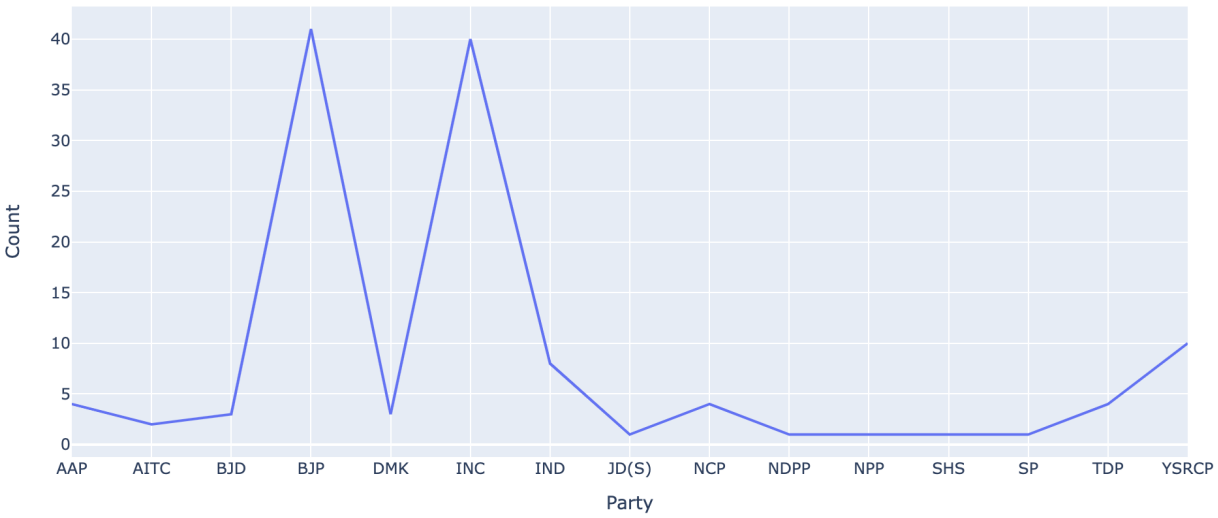
Number of People with More than One Criminal Case per Party



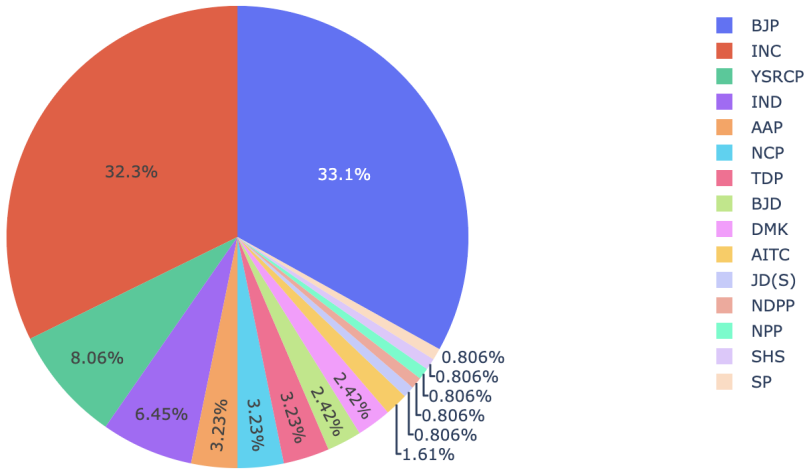
Percentage of People with More than One Case per Party

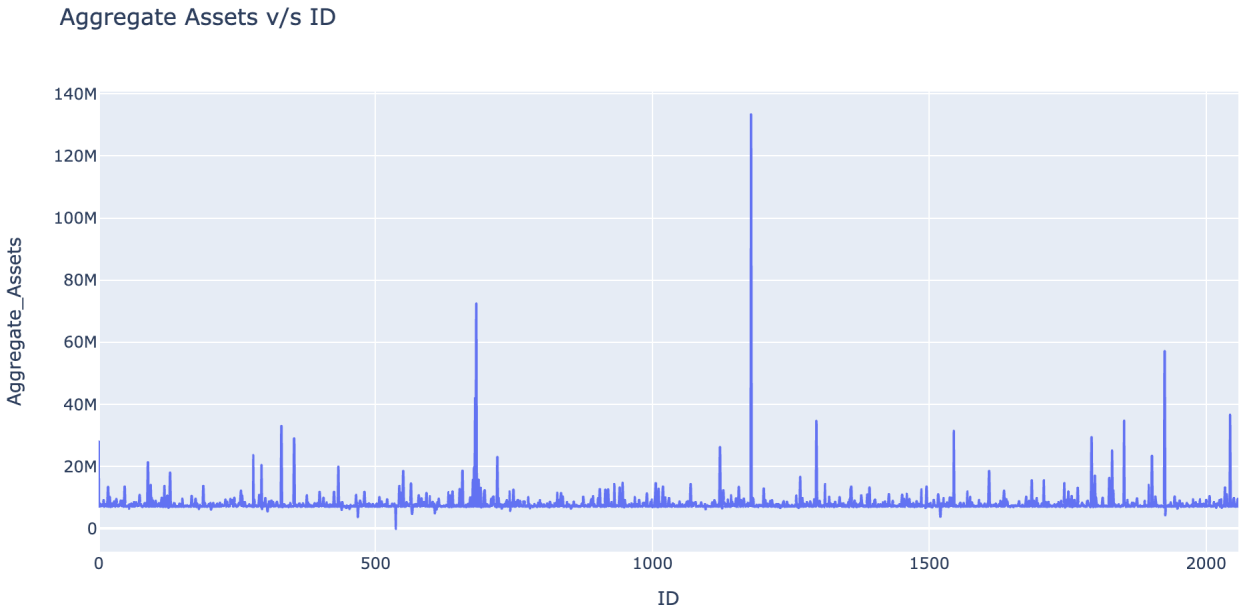


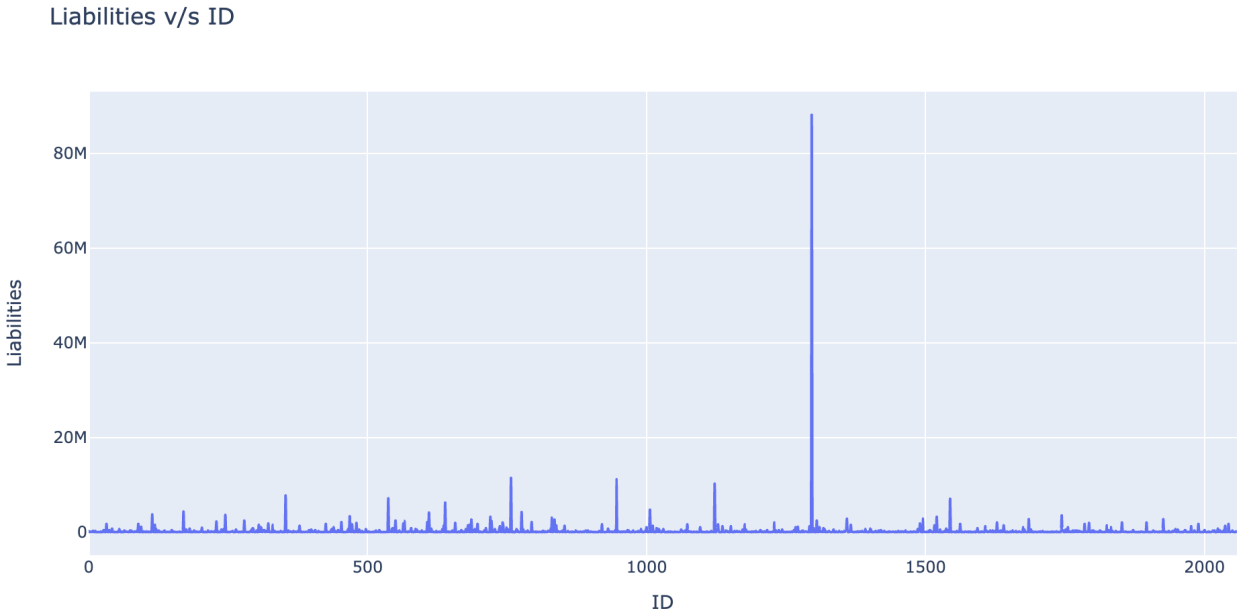
Number of Candidates with More than 10,000,000 Aggregate Assets per Party



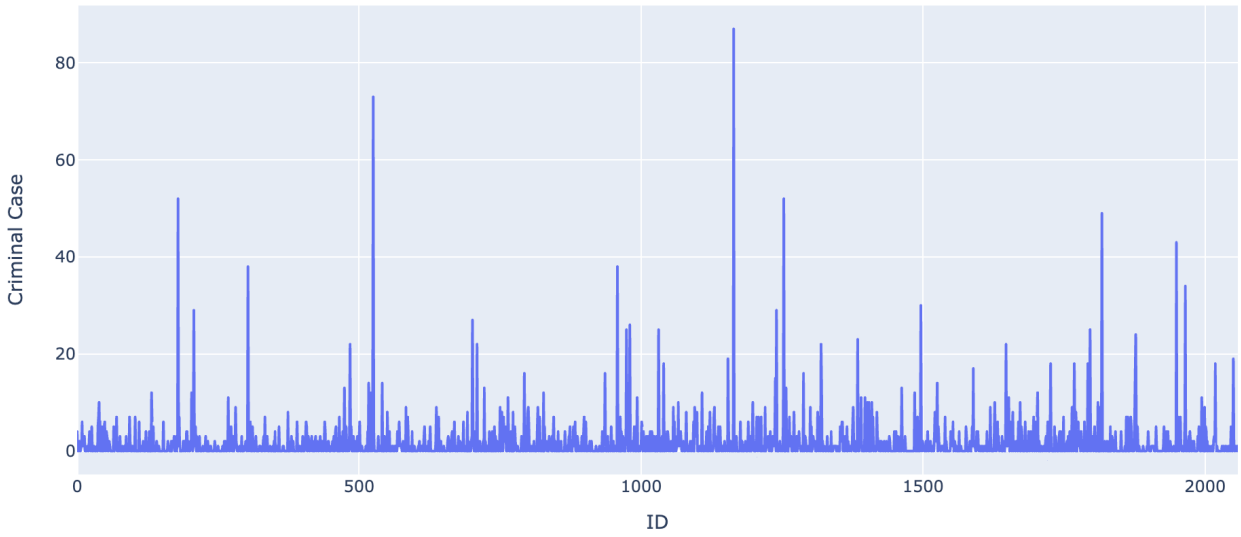
Percentage of Candidates with More than 10,000,000 Aggregate Assets per Party



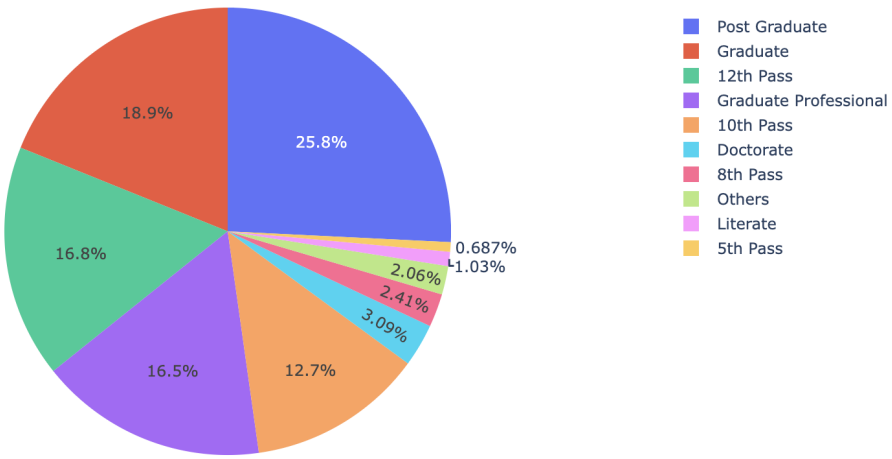

Aggregate Assets v/s ID
140M
120M
100M
80M
60M
40M
20M
0
Aggregate_Assets
0
500
1000
1500
2000
ID


Liabilities v/s ID
80M
60M
40M
20M
0
Liabilities
0
500
1000
1500
2000
ID

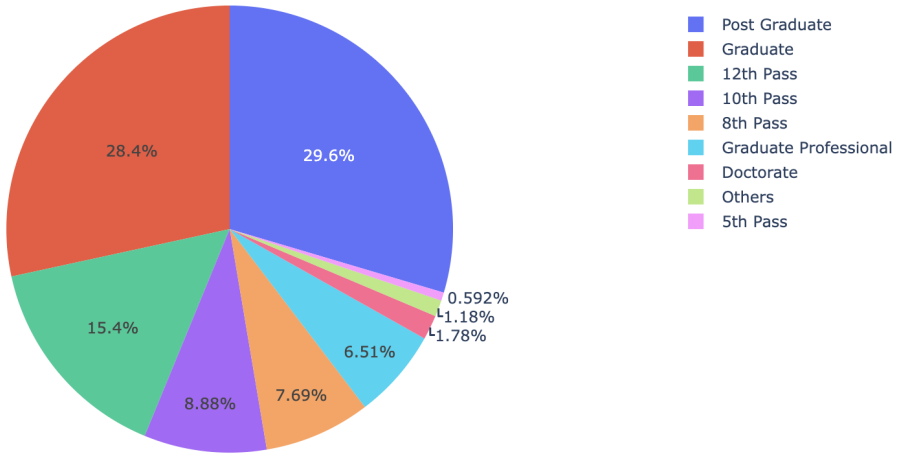
Criminal Cases v/s ID



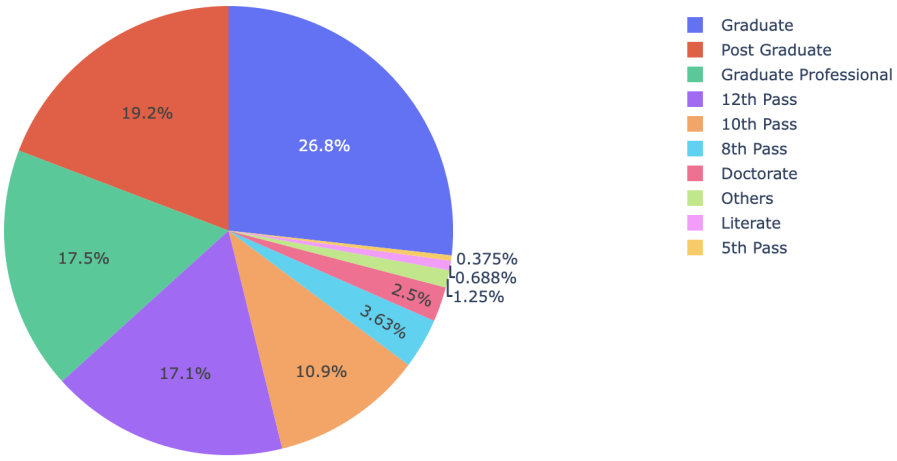
Education Distribution in SC Constituency



Education Distribution in ST Constituency



Education Distribution in Other Constituencies



5 Conclusion & Results

[H] In the final stage of evaluating my model, I employed two distinct approaches. Initially, I utilized the built-in model evaluation method (**bernoulli_nb.score()**) alongside **kFold** cross-validation, yielding an average score of **23.9%**, with the highest score reaching **25.3%**. This metric was directly correlated with the accuracies obtained upon submitting predictions on the real test dataset. Conversely, assessing the **F1 score** on predictions generated from a train data split using an optimal ratio of 0.8:0.2 revealed a maximum F1 score of **21.5%**. Despite utilizing the best hyperparameters, the F1 score proved to be unreliable for this particular case due to the observed decrease in accuracy of models such as DT and RF on the actual test dataset.

Leaderboard Rankings

Maximum Score on Public Leaderboard - **28.049%**

Leaderboard	Aggregate Rank	Actual Rank
Public	**4**	9
Private	**13**	21

Final Code & CSV File

- **Google Colab Code**
- **Best Submission: CSV File**
- Github: PKING1501.

Resources and References

In preparing this assignment, I relied on the following credible sources:

- SkLearn-BernoulliNB
- SkLearn-SVC
- SkLearn-KNN
- SkLearn-DecisionTrees
- SkLearn-RandomForests
- Kaggle-GBClassifier
- Kaggle-XGBoost
- Plotly Express

Contact

Name	Email
Prem Kansagra	premk22@iitk.ac.in