CS253 Python Assignment Submission

Naman Kumar Jaiswal
220687
namankj22@iitk.ac.in

Spring Semester 2024

Abstract

This report presents a solution to the multi-class classification problem "Who-is-the-real-winner?" on Kaggle.We have used Multinomial Naive Bayes Model to classify candidates into different educational backgrounds based on features such as constituency, party affiliation, criminal records, financial assets and Liabilities and finally state. The analysis includes data pre-processing, model selection, and evaluation using the F1 score metric.

Link to the github code: https://github.com/Naman-K-Jaiswal/CS253_ML_Assignment

1 Introduction

Machine learning (ML) techniques have become increasingly prevalent across various domains due to their capability to extract insights and make predictions from data. In this report, we address the task of multi-class classification using a provided training dataset. Our objective is to train a machine learning model to classify candidates into different educational backgrounds based on various features such as their constituency, party affiliation, criminal records, and financial assets. There are two types of data sets provided to you namely train and test where train data allows you to train an ML model and use them to predict test data. Even though there is a 50-50 public private split of data where only the public score is visible for the time period of the tournament and the private score is revealed only after the contest.

The F1 score is a metric that balances precision and recall to evaluate the performance of a classification model. Precision, the positive predictive value, measures the proportion of correct positive predictions out of all positive predictions and reflects how few false positives the model has. Recall, or sensitivity, measures the proportion of actual positive cases correctly identified by the model, indicating how effectively the model detects positives. The F1 score, which ranges from 0 to 1, is the harmonic mean of precision and recall, offering a single measure of the model's accuracy and effectiveness. This metric is especially useful in cases of class imbalance, as it equally considers both false positives and false negatives. In our analysis, the F1 score serves as a crucial performance metric for assessing the model's ability to classify candidates into different educational backgrounds while minimizing misclassifications.

2 Methodology

2.1 Data Preprocessing

1. We started with looking for missing values in the database, luckily there are none so we need not perform filling of those cells.
2. Next we converted Total Assets, Liabilities into Numerical data. These columns made sense as a numerical value and replacing the string with a number which reveals more about its standing wrt to its distribution.

2.2 Feature Engineering

We created four new features to capture additional information from the dataset:

1. Dr: Indicates whether a candidate has a prefix 'Dr.' in their name.
2. Adv: Indicates whether a candidate has a prefix 'Adv.' in their name.
3. SC: Indicates whether a candidate is running on a SC constituency.
4. ST: Indicates whether a candidate is running on a SC constituency.

The features used are:

1. Party: Political party affiliation of the candidate.

2. Criminal Case: Number of criminal cases registered against the candidate.
3. Total Assets: Total assets declared by the candidate.
4. Liabilities: Total liabilities declared by the candidate.
5. State: State where the constituency is located.
6. Dr: Binary feature indicating whether the candidate has a prefix 'Dr' in their name.
7. Adv: Binary feature indicating whether the candidate has a prefix 'Adv' in their name.
8. SC: Binary feature indicating preference for constituencies with special status.
9. ST: Binary feature indicating preference for constituencies with special status

2.3 Dimensionality Reduction

We have dropped the columns 'Candidate' and 'Constituency' from the database because it add little to zero information about the target variable. These columns are almost unique for each entry. We can drop these values only after

2.4 Outlier Detection

Identified outliers using the IQR method. We have dropped the rows which have the numerical columns like 'Liabilities', 'Total Assets', 'Criminal Case' beyond 2.2 times the Inter Quartile Range. These values are way too skewed which drops the accuracy of our model while training. Training the model on the rest of the data improves the accuracy by atleast 0.04.

2.5 Normalization and Standardization

1. We have a big range of Assets and Liabilities which makes the values ending in hundreds under represented as compared to the large values. We ran a map on these columns where each value is mapped to int(3*np.log10(int(value/100)). This forces the values to be on a log scale for best representation.
2. We have used a discretizer to covert the the numerical values into bins which is essential for a multuinomial nb to work.

3 Models and Hyperparameters

For our multi-class classification task, we employed the model, 'MultinomialNB(alpha=0.55, fit_prior=True)', a Multinomial Naive Bayes classifier from the 'scikit-learn' library. It is designed for classification tasks where the features represent discrete counts (such as word frequencies in a document), making it particularly suitable for natural language processing tasks and other situations where the features are non-negative integer counts.

Here are details about the model:

1. alpha: This is the smoothing parameter. In Naive Bayes models, alpha is used to handle zero-frequency problems. A non-zero alpha value (like 0.55 in your case) adds a small constant to the numerator and denominator of probability estimates, preventing any issues from zero probabilities. It also controls the degree of smoothing—higher values lead to more smoothing.
2. fit_prior: This is a boolean flag that indicates whether or not the model should learn the class prior probabilities from the training set. If set to 'True', the model calculates the prior probabilities from the data; otherwise, it assumes uniform prior probabilities across classes.

3.1 Discretizer

A discretizer is a preprocessing tool that converts continuous features into discrete bins. This can be useful for certain machine learning models, particularly those that require categorical inputs. It segments continuous data into a set number of bins or intervals, converting the continuous values into categorical or discrete values.

Hyperparameters of a discretizer:

1. n_bins (5): The number of bins or intervals to divide the data into. This is an important parameter that controls the granularity of the discretization. A higher number of bins can lead to more detailed feature representation but might also increase noise or overfitting.
2. Strategy (uniform): This parameter defines how the bins are computed. Common strategies include 'uniform' (equal width intervals), 'quantile' (equal frequency intervals), and 'kmeans' (bins determined by k-means clustering).
3. encode (encode): This parameter determines how the discretized features are encoded. Common options are 'onehot' for one-hot encoding or 'ordinal' for integer encoding.

3.2 Local Machine Evaluation

We are using KFold cross-validation (KFold(n_splits=5, shuffle=True, random_state=42)) to evaluate our model. This is a method for evaluating your model by splitting the dataset into 5 folds and training/testing the model multiple times (one fold held out each time as a test set). It provides multiple accuracy scores that can be used to calculate mean and standard deviation, giving you insight into the model's performance consistency.

4 Other models applied

We have also applied the following models but they showed poor accuracy results as compared to multinomial.

1. Random Forest: F1 score of 23%
2. Decision Tree: F1 score of 22%
3. Ada Boost: F1 score of 20%
4. Gradient Boost: F1 score of 21%
5. Extreme Gradient Boost: F1 score of 19%
6. Bernoulli NB: F1 score of 25%
7. Gaussian NB: F1 score of 23%

5 Results

Our final model earned the spot of rank 6 with the f1 score of 0.27491 on public data and rank 44 with f1 score of 0.24711 on private data in just 10 tries.

6 References

1. **Numpy Documentation:** https://numpy.org/doc/stable/user/index.html
2. **Pandas Documentation:** https://pandas.pydata.org/docs/reference/frame.html
3. **SciKit Learn Documentation:** https://scikit-learn.org/stable/

4. **Boosting Algorithms:** https://scikit-learn.org/stable/modules/ensemble.html#gradient-boosted-trees
5. **Naive Bayes Algorithm:** https://scikit-learn.org/stable/modules/naive_bayes.html

7 Data Analysis

7.1 Plots

Generated plots such as the distribution of education levels, percentage distribution of parties with criminal records, and percentage distribution of parties with wealthy candidates. Created a correlation heatmap to explore relationships between numerical features.

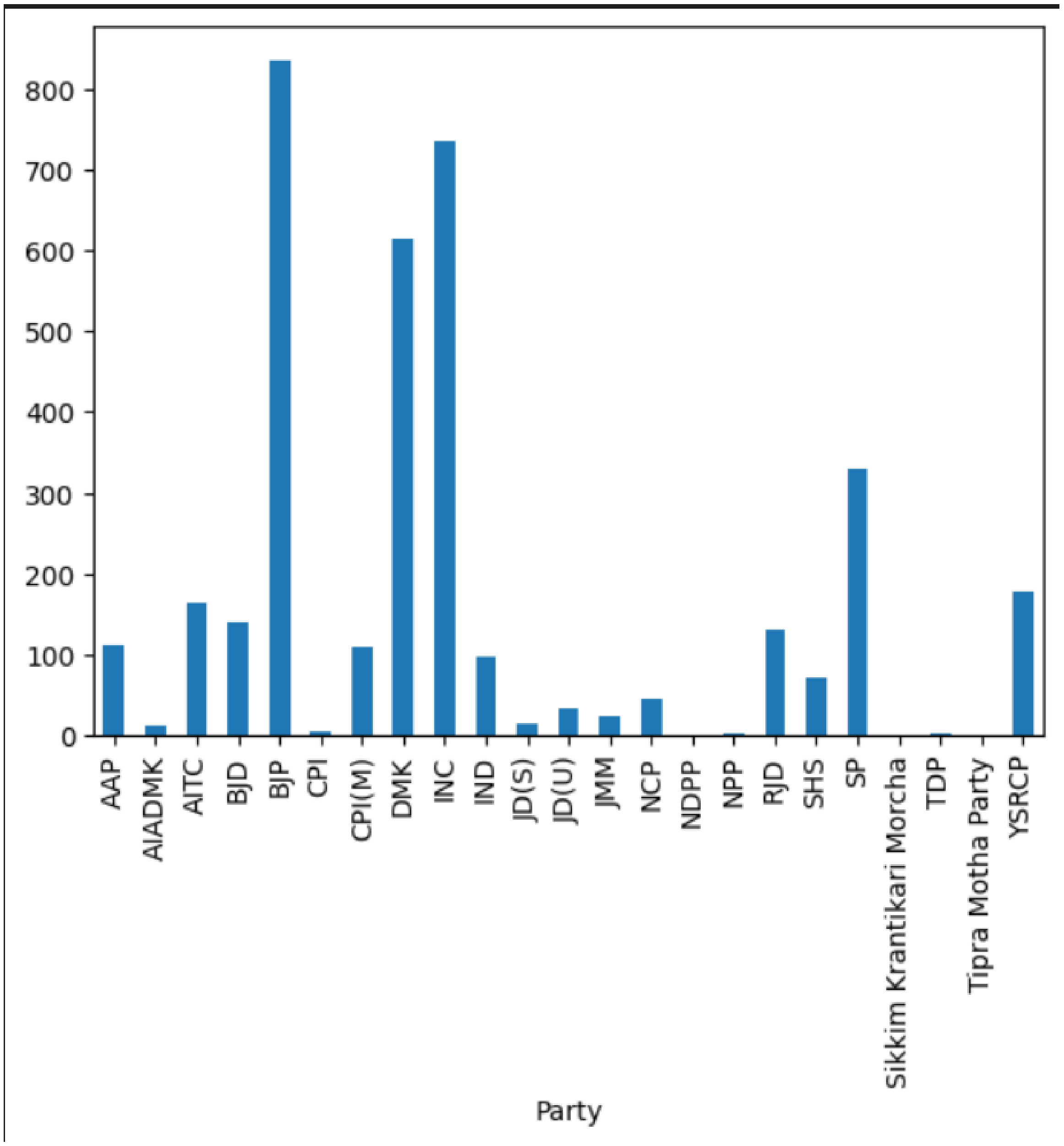

Figure 1: Criminal Cases v Party

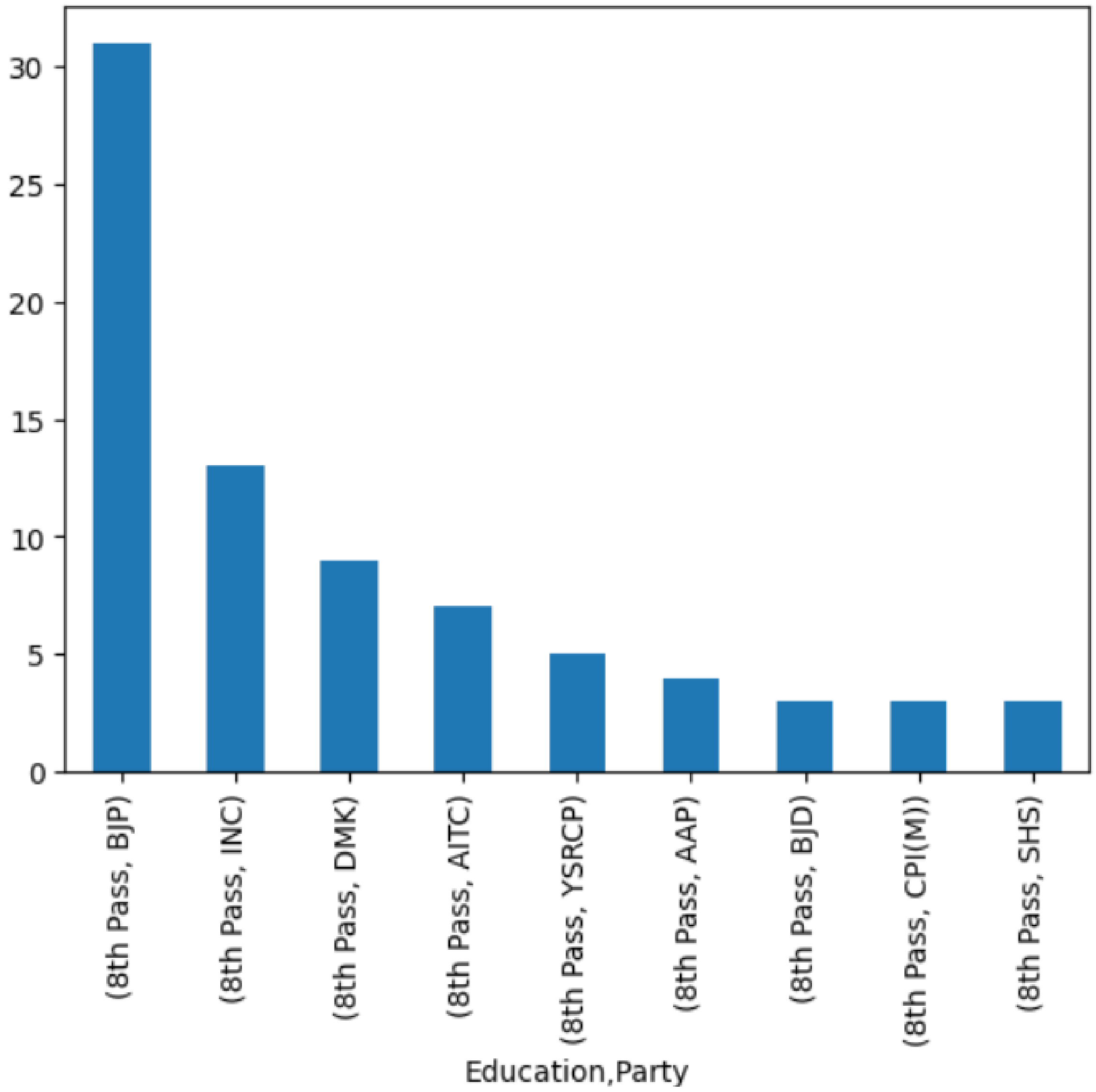

Figure 2: 8th Pass, Party v Count

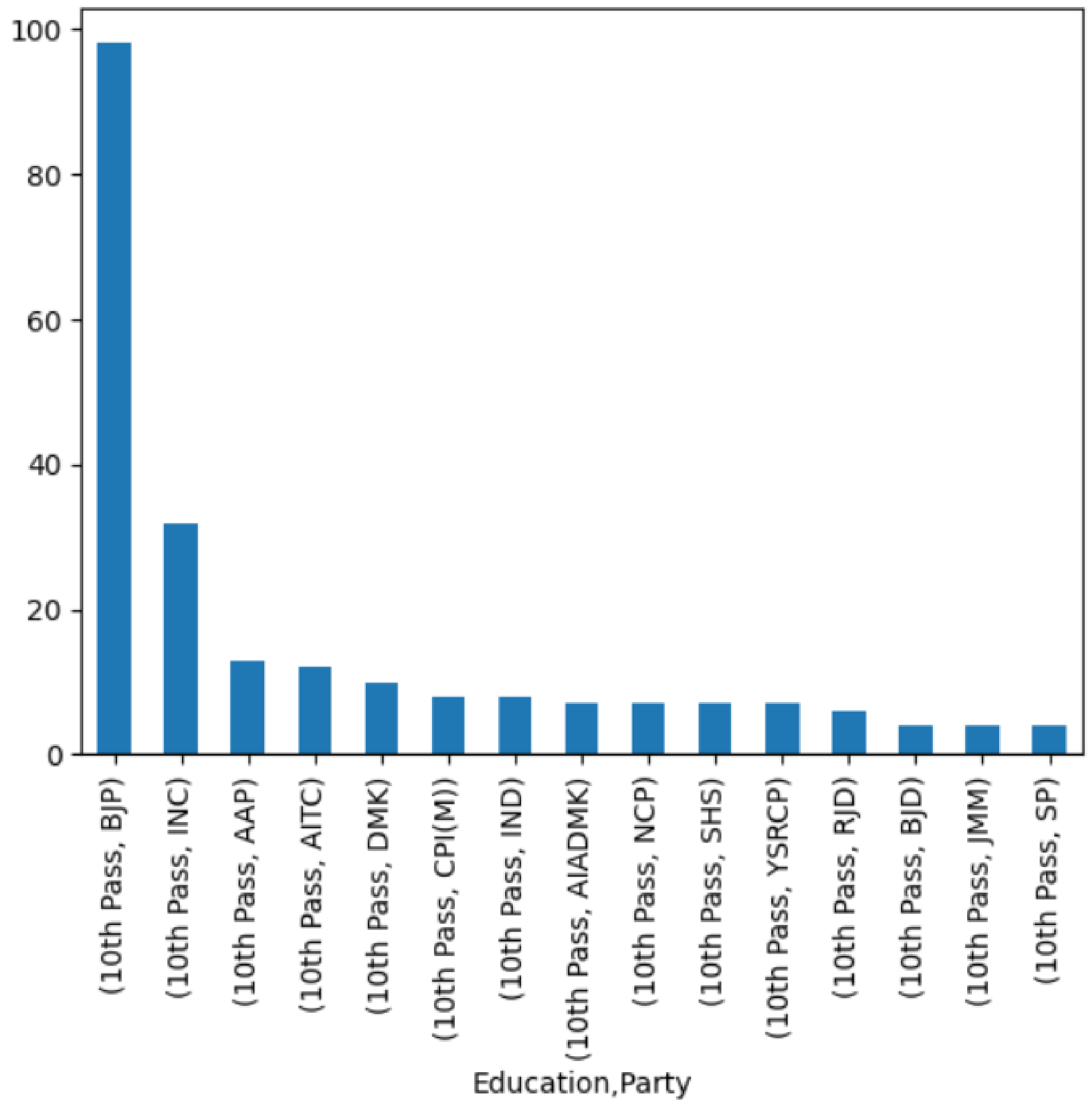

Figure 3: 10th Pass, Party v Count

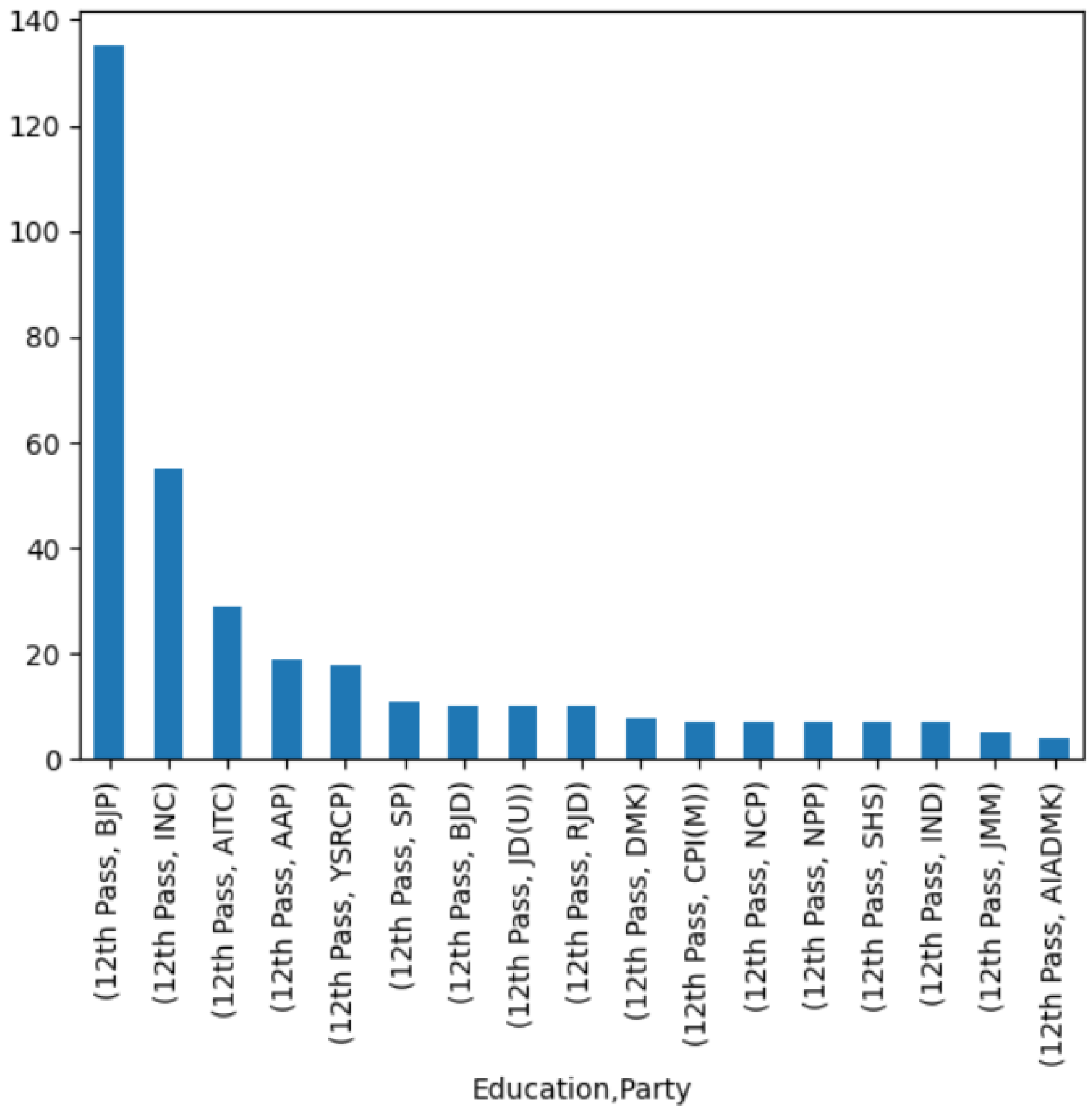

Figure 4: 12th Pass, Party v Count

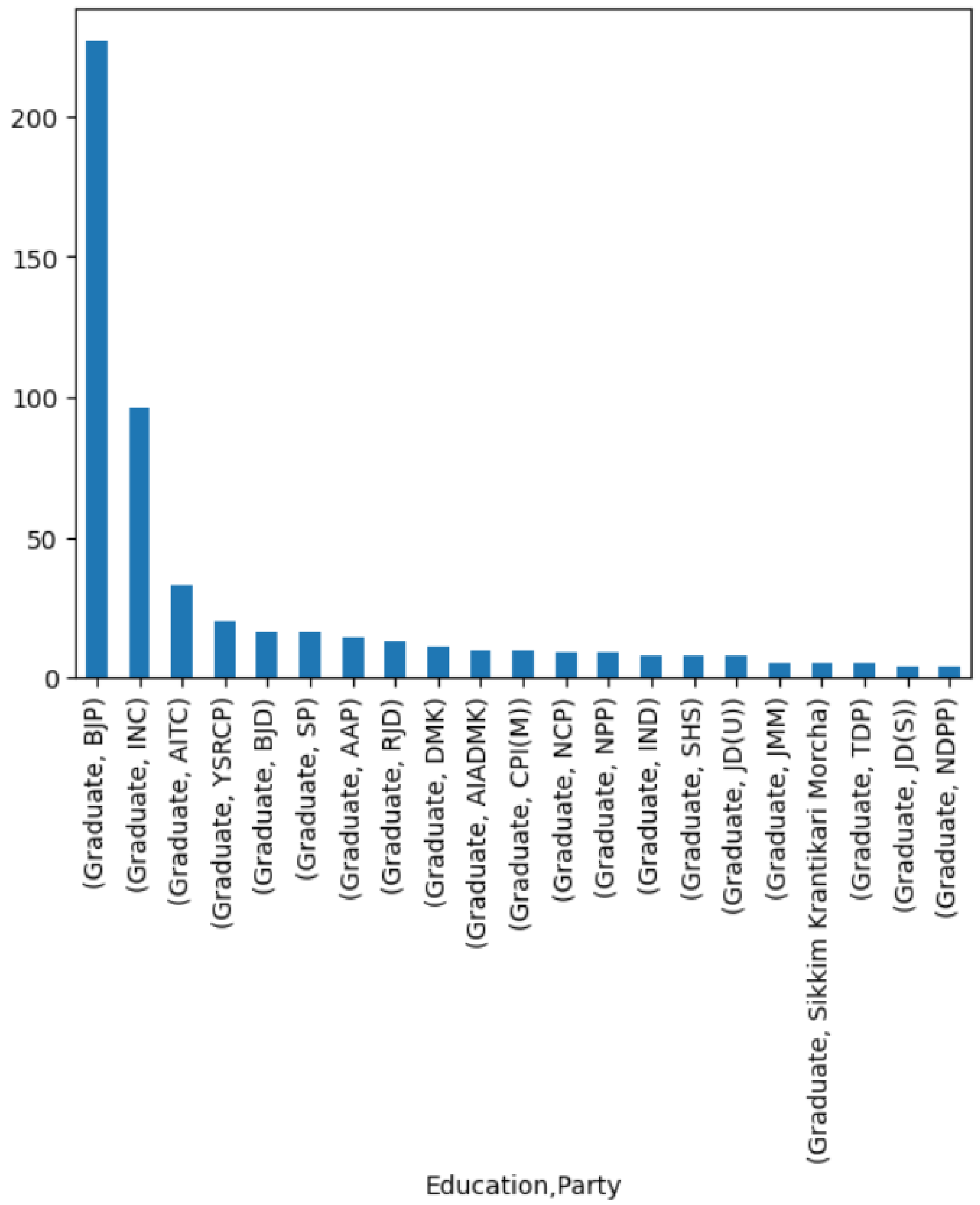

Figure 5: Graduate, Party v Count

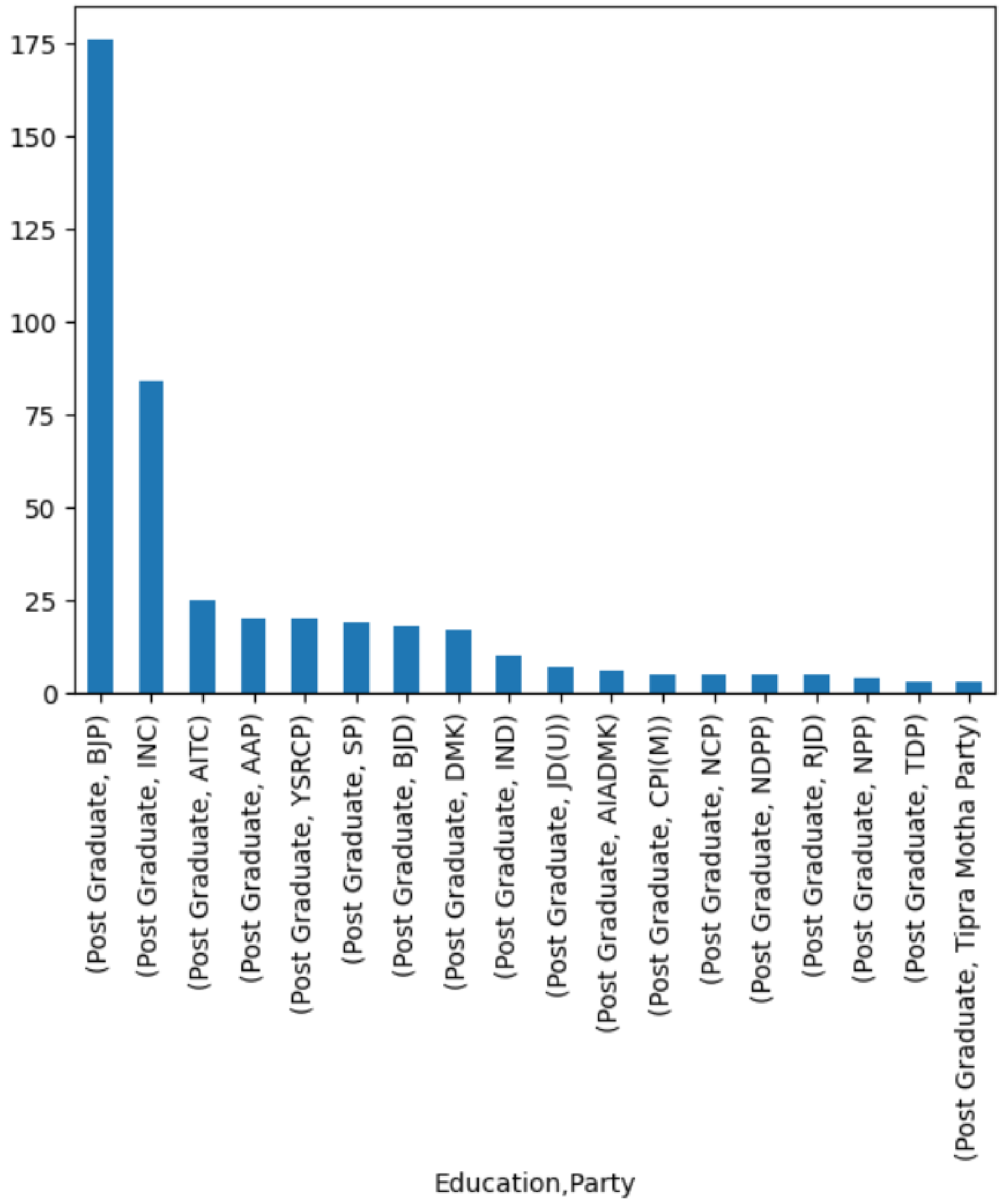

Figure 6: Post Graduate, Party v Count

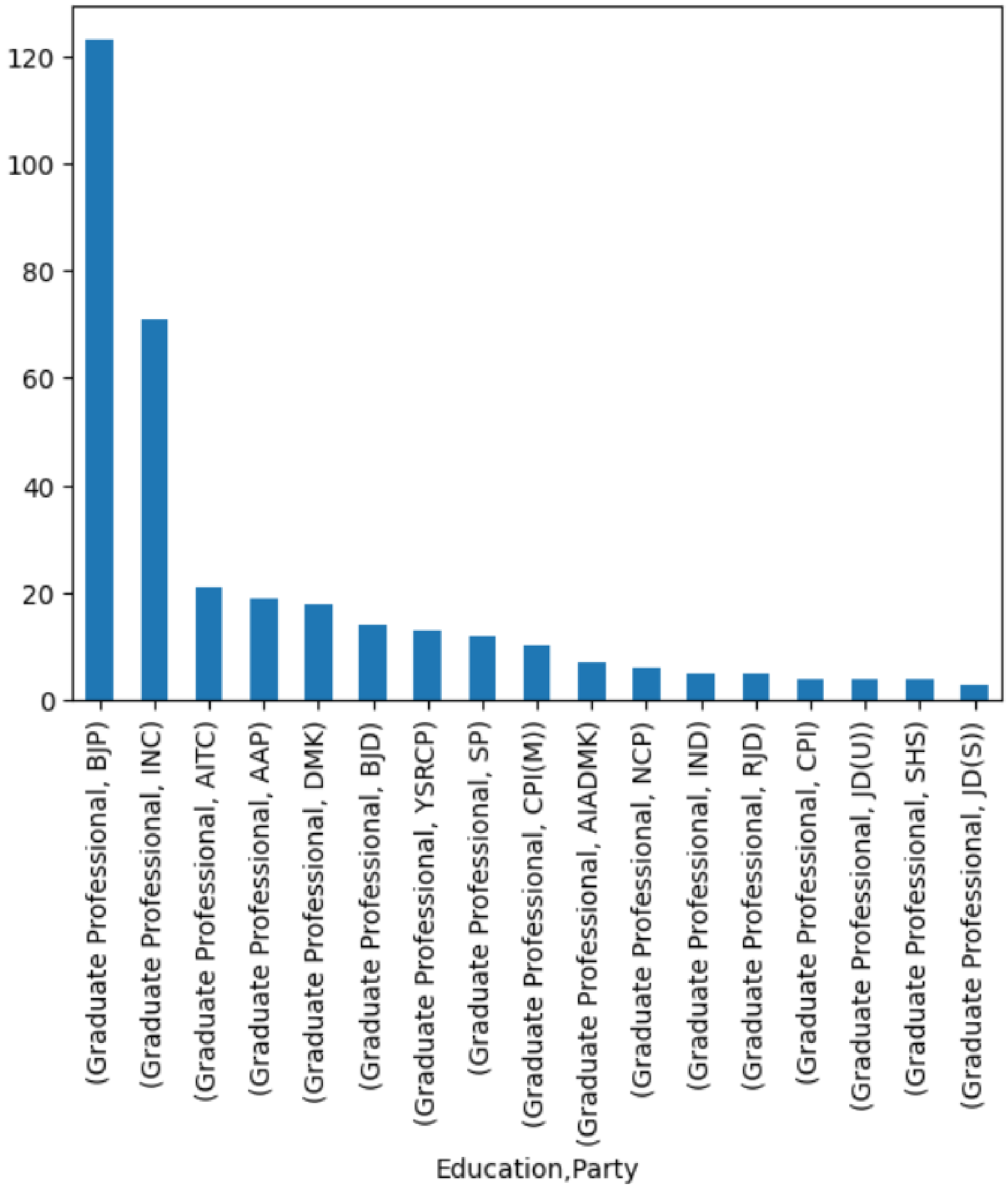

Figure 7: Graduate Professional, Party v Count

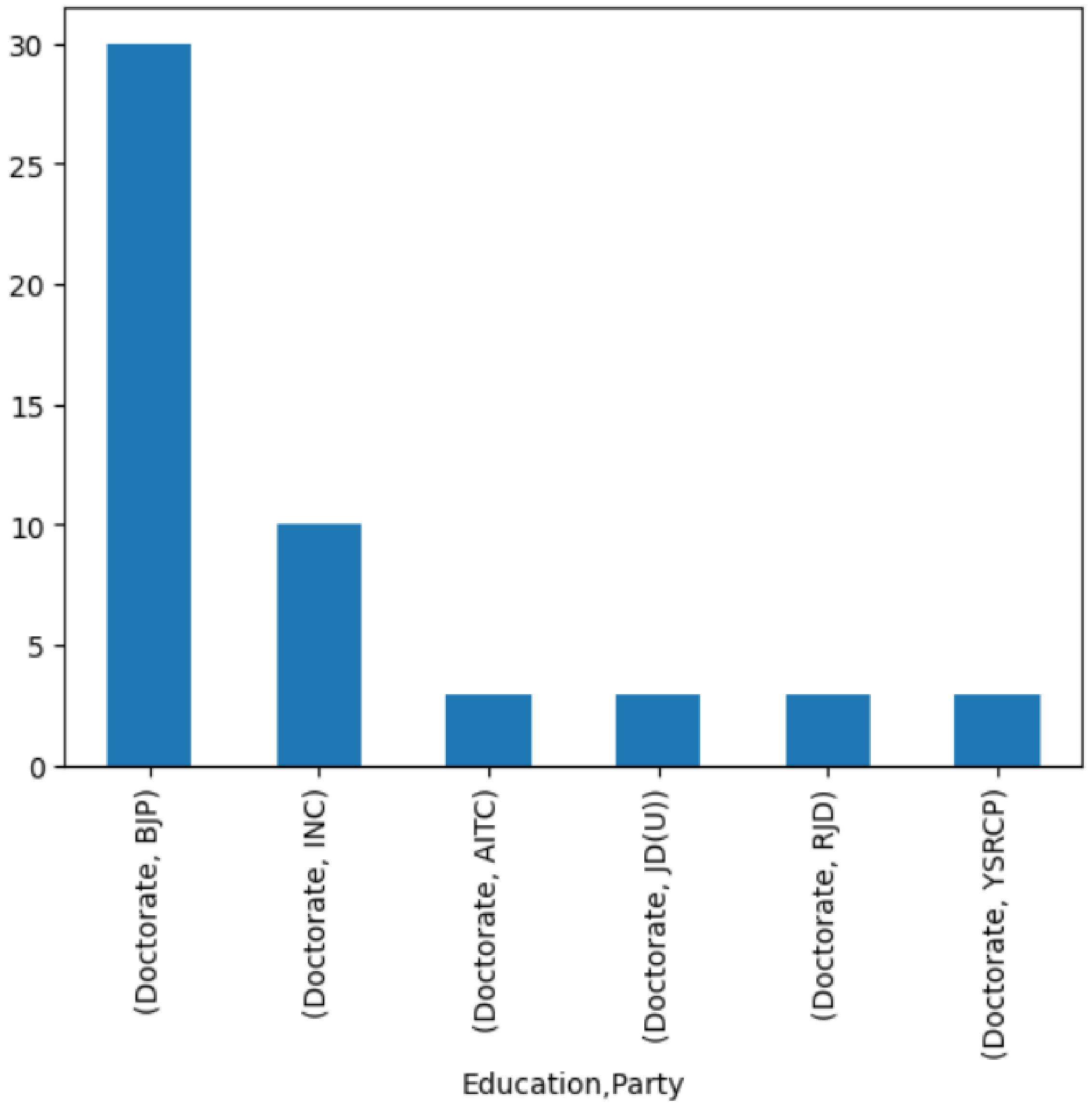

Figure 8: Doctorate, Party v Count

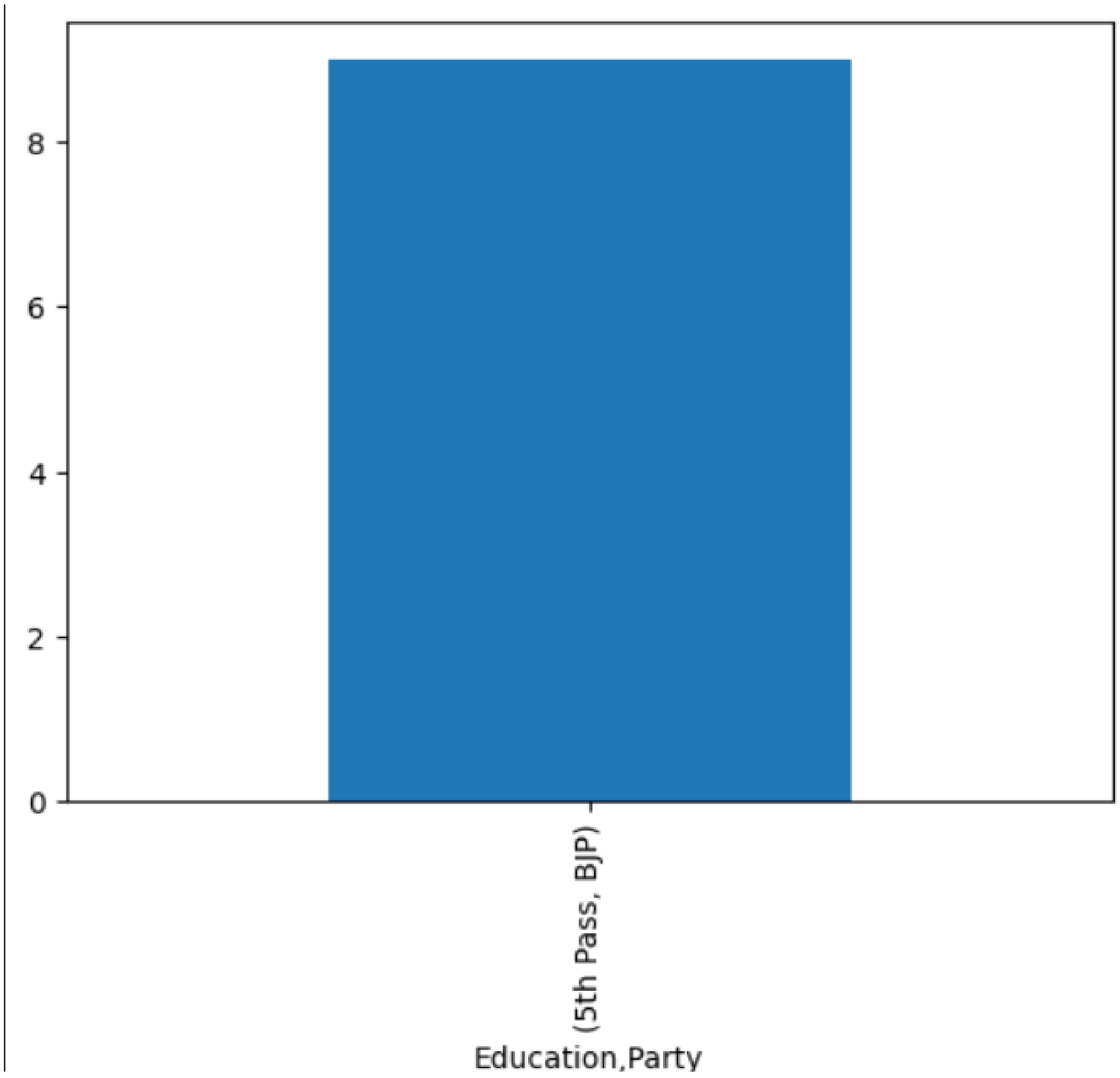

Figure 9: 5th Pass, Party v Count

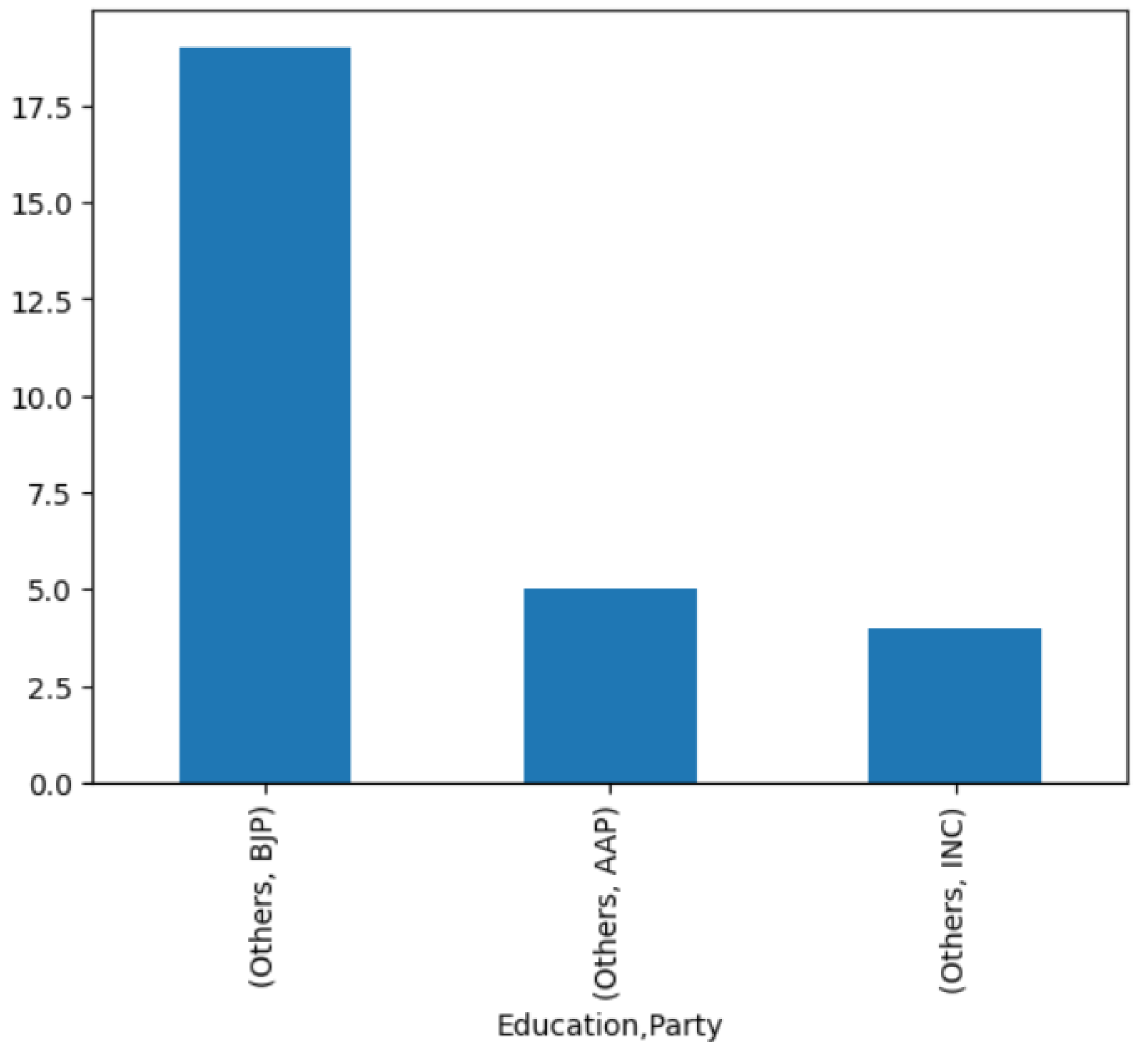

Figure 10: Others, Party v Count

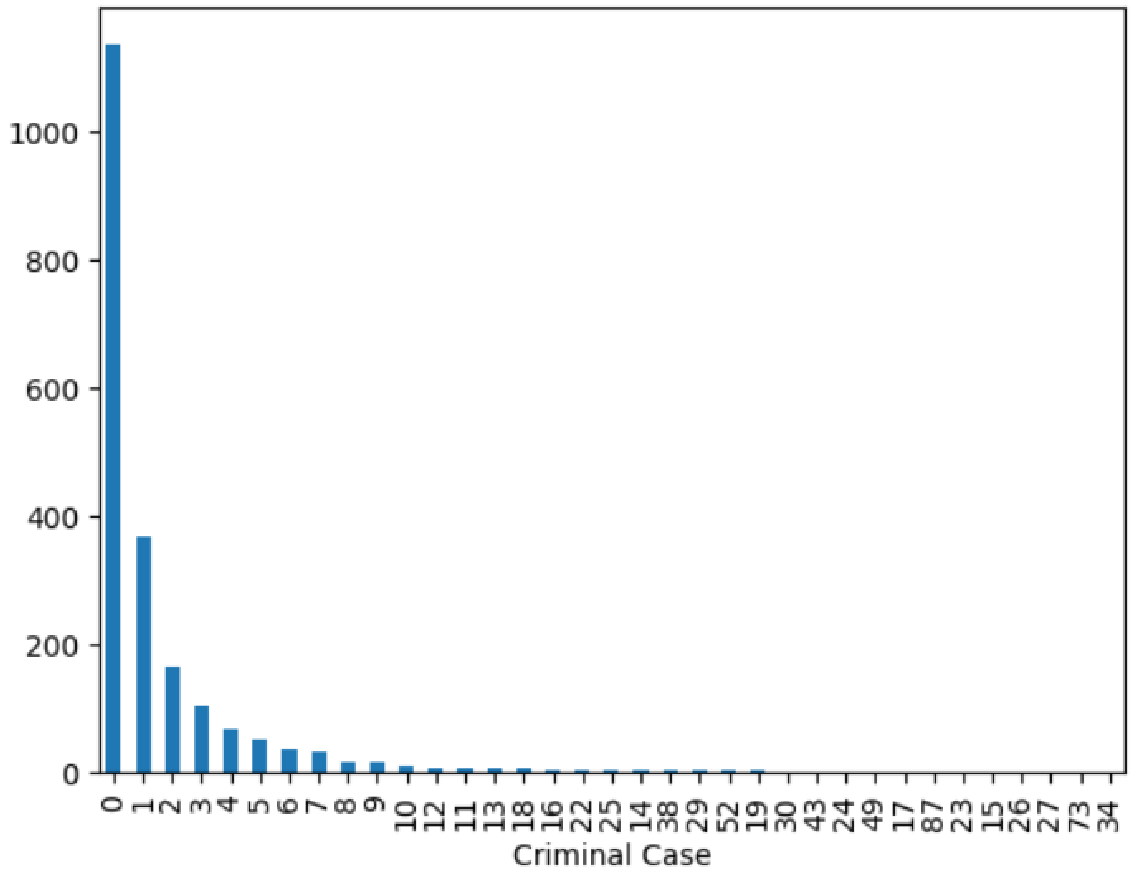

Figure 11: Criminal Cases v Count

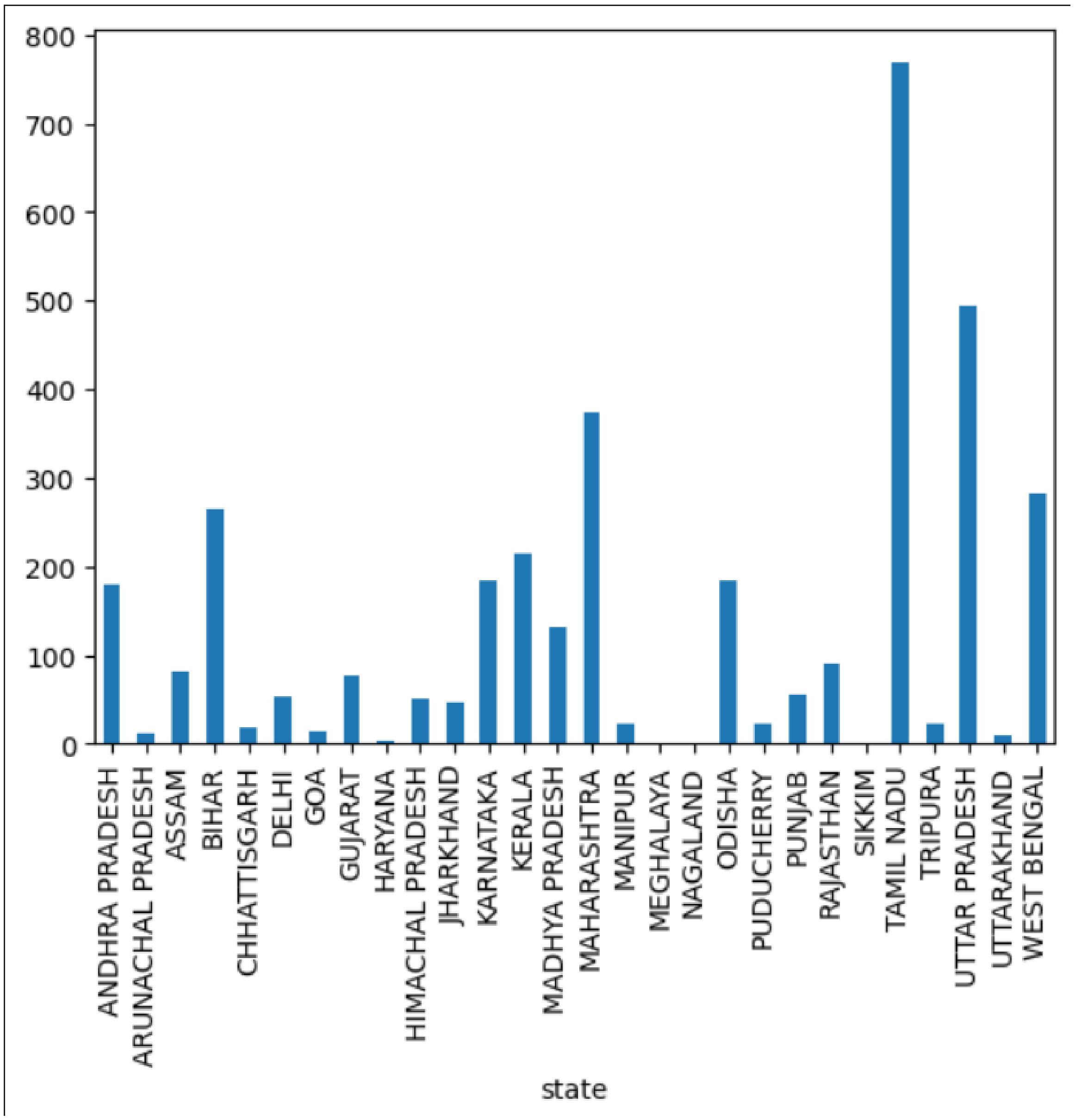

Figure 12: Criminal Cases v State

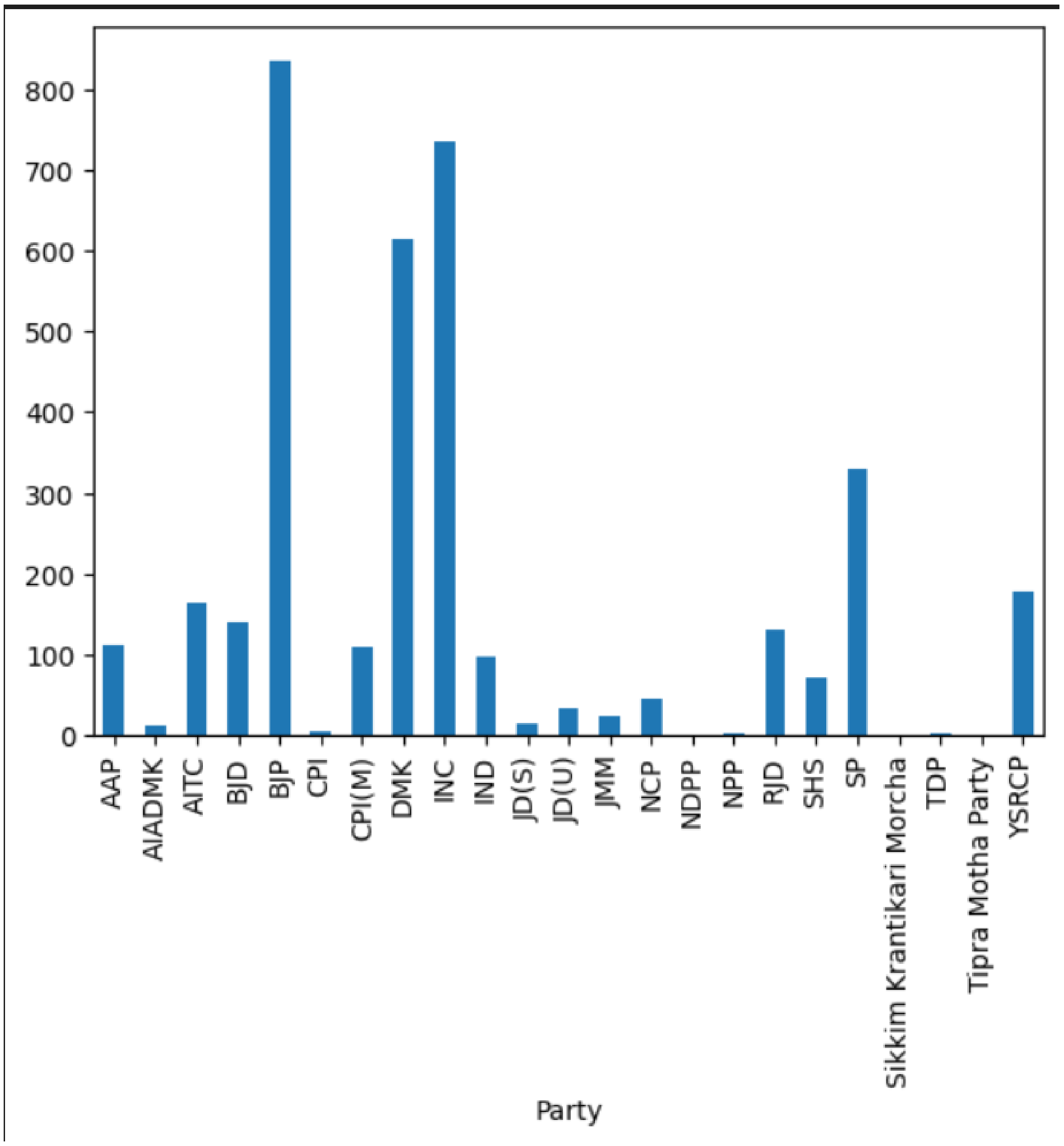

Figure 13: Criminal Cases v Party

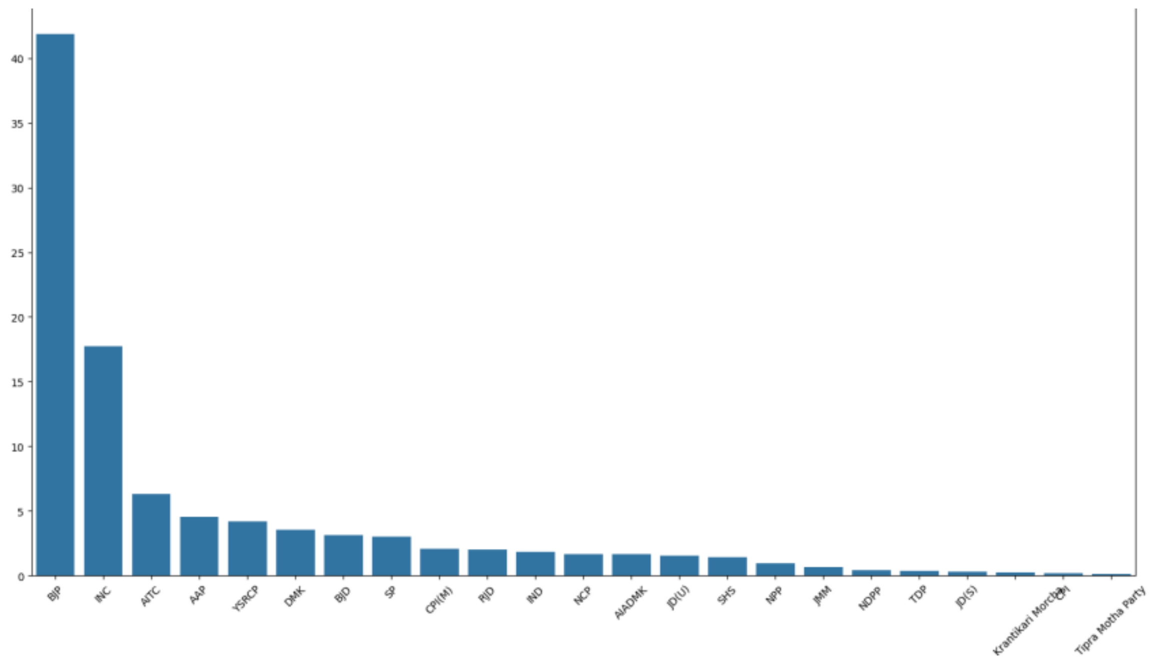

Figure 14: Total Assets v Party