CS253 Assignment 3

Saksham Malhotra

1 Introduction

This document outlines the process of training a machine learning model to predict the education level of individuals using a RandomForestClassifier. The dataset includes various features such as party affiliation, criminal cases, total assets, and liabilities. For more detailed information about RandomForestClassifier, see the official documentation here.

2 Implementation Details

2.1 Imports

The Python script utilizes several libraries:

- `pandas` for data manipulation.
- `scikit-learn` for machine learning tools.

2.2 Numeric Conversion Function

```
def convert_to_numeric(value):
    value = str(value)
    if 'Crore+' in value:
        return float(value.replace('Crore+', '')) * 10**7
    elif 'Lac+' in value:
        return float(value.replace('Lac+', '')) * 10**5
    else:
        return pd.to_numeric(value, errors='coerce')
```

This function converts string representations of numbers into floats, handling 'Crore+' and 'Lac+' suffixes appropriately.

2.3 Data Loading

The datasets are loaded from CSV files using:

```
train_df = pd.read_csv('path_to_train.csv')
test_df = pd.read_csv('path_to_test.csv')
```

2.4 Data Preprocessing

The script includes conversion of certain columns to numeric types, filling missing values, and encoding categorical features.

2.5 Feature and Label Split

Features are isolated in `X` and the target variable 'Education' in `y`.

2.6 Hyperparameter Tuning

Using `GridSearchCV` from scikit-learn, the script performs hyperparameter tuning to find the optimal settings for the RandomForestClassifier. The parameters tested include:

- **n_estimators**: Number of trees in the forest. Tested values: 100, 200, 300. More trees can lead to better performance but increase computational cost. Best Parameter: 200
- **max_depth**: Maximum number of levels in each tree. Tested values: None (unlimited), 10, 20. Limiting depth can reduce overfitting. Best Parameter: None
- **min_samples_split**: Minimum number of samples required to split an internal node. Tested values: 2, 5, 10. Higher values prevent the model from learning overly specific patterns, reducing overfitting. Best Parameter: 2
- **min_samples_leaf**: Minimum number of samples required to be at a leaf node. Tested values: 1, 2, 4. Like `min_samples_split`, higher values increase the robustness of the model. Best Parameter: 1

2.7 Model Training

```
model = RandomForestClassifier(**best_params, random_state=42)
model.fit(X, y)
```

The model is trained using the best parameters found from the grid search.

2.8 Prediction and Submission

```
y_pred = model.predict(X1)
test_df['Education'] = le.inverse_transform(y_pred)
test_df[['ID', 'Education']].to_csv('submission_7.csv', index=False)
```

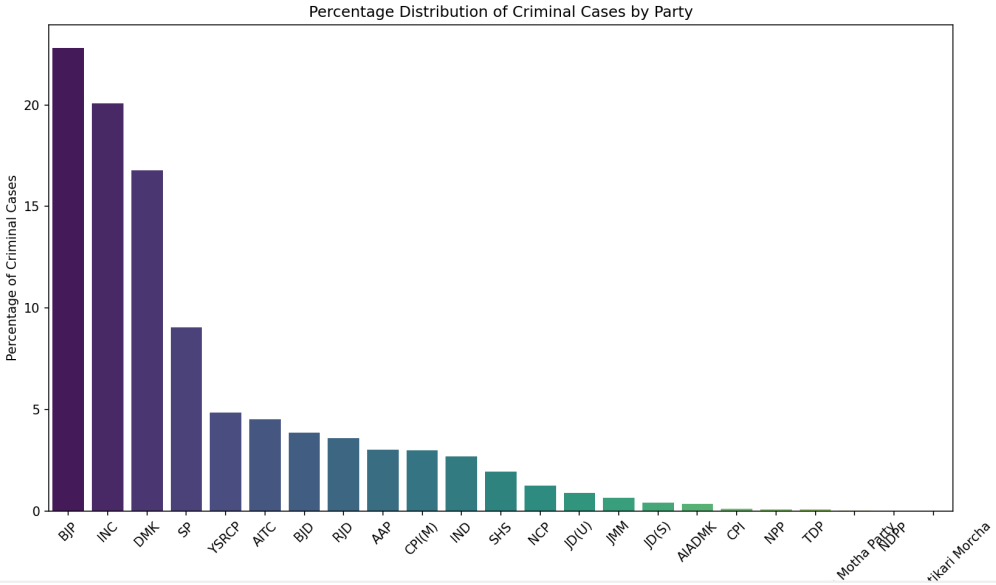
Predictions are made on the test set, transformed back to original labels, and saved for submission.

3 Github Link

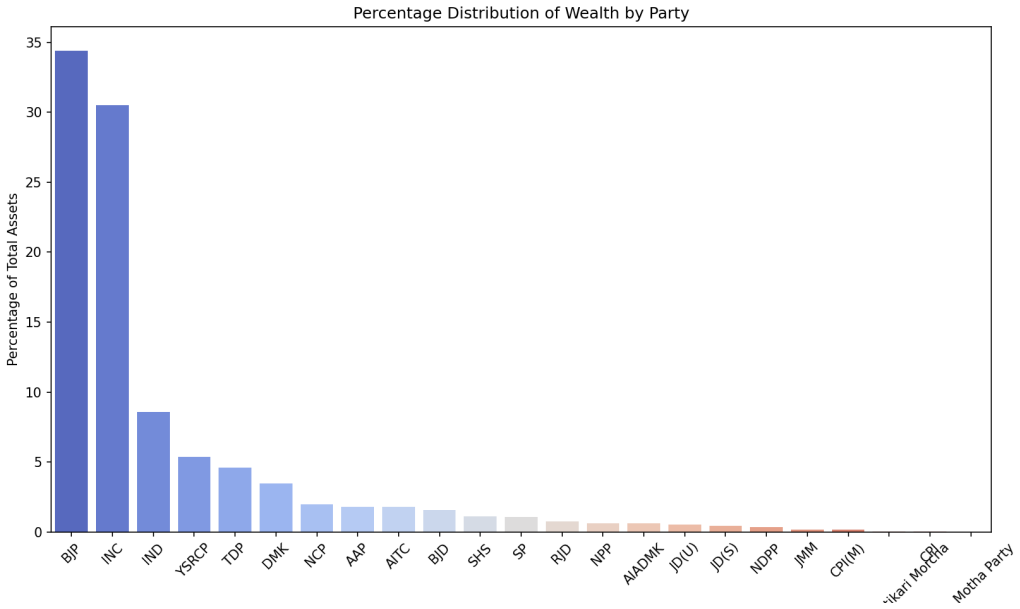
`https://github.com/sakshamm21/CS253-Assignment-3.git`

4 Plots

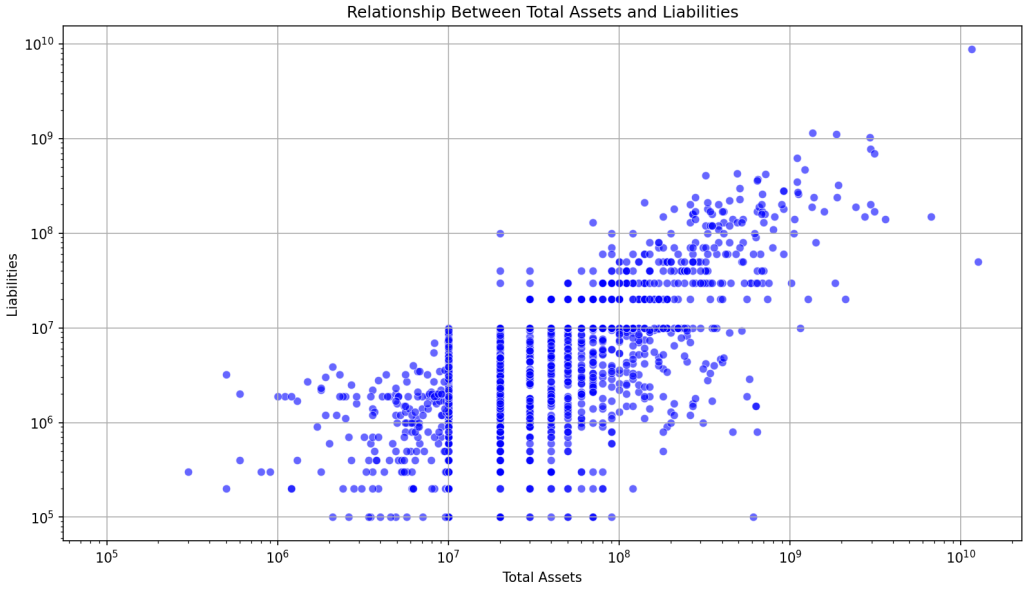
4.1 Percentage distribution of parties with candidates having the most criminal records



4.2 Percentage distribution of parties with the most wealthy candidates



4.3 Bonus Plot: Total Assets vs Liabilities



5 Leaderboard Rank and Score

RANK : 215 (Many deleted submission, this rank in including those also)
SCORE : 0.20071