Python Assignment Report

Roshan Sanap

April 2024

1 Methodology

This section details the systematic approach taken to analyze and model the dataset provided in the Kaggle competition "Who is the real winner?". The methodology encompasses several crucial steps, from data preprocessing to exploratory data analysis, each designed to optimize the dataset for effective model training and prediction accuracy.

1.1 Data Preprocessing Steps:

1.1.1 Currency Conversion:

The 'Total Assets' and 'Liabilities' fields were originally presented in a textual format with varying denominations (e.g., Crore, Lac, Thousand). These values were standardized to numerical format in Indian Rupees to facilitate quantitative analysis.

1.1.2 Encoding Categorical Variables:

Categorical variables such as 'Party', 'State', and 'Education' were transformed into numerical codes. This encoding is essential for fitting machine learning models which typically require numerical input.

1.2 Exploratory Data Analysis (EDA)

1.2.1 Distribution and Relationship Analysis:

Initially, the distribution of various features like education levels, criminal cases, and asset values were visualized to understand their individual characteristics and detect any outliers or anomalies.

1.2.2 Correlation Studies:

Relationships between features were explored using scatter plots, correlation matrices, and box plots to discern patterns or trends that could inform the predictive modeling, such as the link between education levels and asset declarations.

1.3 Model Preparation Steps:

1.3.1 Data Splitting

The dataset was split into training and testing sets, ensuring a diverse representation of data points in each set to prevent model overfitting and to validate model performance effectively.

1.3.2 Feature Selection

Key features that are likely to influence the outcome (education level) were identified through domain understanding and iterative testing, which guided the model training phase.

These methodologies provide a structured approach to tackling the problem statement, laying a foundation for the subsequent experimental modeling phase.

2 Experiment Details

This section outlines the specifics of the experimental setup, including the machine learning models tested, their hyperparameters, and other relevant details crucial for understanding the modeling approach.

Model	Hyperparameters	Description
SVM (Support Vector Machine)	Kernel:Linear, C:10	SVM was chosen for its effectiveness in handling high-dimensional data and its capability to model non-linear decision boundaries using various kernels.
KNN (K-Nearest Neighbors)	Number of Neighbors:5, Metric:Minkowski	KNN was utilized to investigate the impact of simple distance-based decision maling in the classification, with a small neighborhood to avoid over-smoothing
Decision Tree	Max Depth: None, Criterion: Gini	Decision Trees provide a straightforward interpretation of decision rules, considered for its transparency in decision making.
Random Forest	Number of Estimators: 100, Criterion: Gini, Max Depth: None	An ensemble method using multiple decision trees to improve prediction stabilty and accuracy, reducing the risk of overfitting inherent in single decision trees:

Additional Details: - Validation Technique:Cross-validation with a 10-fold strategy was used to ensure that the model's performance is consistent across different subsets of the dataset. - Performance Metric: The F1-score, which balances precision and recall, was chosen as the primary metric due to its importance in assessing model performance on imbalanced class distributions prevalent in this dataset.

These models were iteratively trained and tuned, with hyperparameters adjusted based on the validation set performance to find the optimal configuration for each model type.

2.1 Data Insights

This section provides a visual analysis of the dataset, highlighting key trends and distributions that inform the predictive modeling. Below are descriptions of the graphs and plots obtained during the exploratory data analysis, along with interpretations of these visualizations.

2.1.1 Distribution of Education Levels:

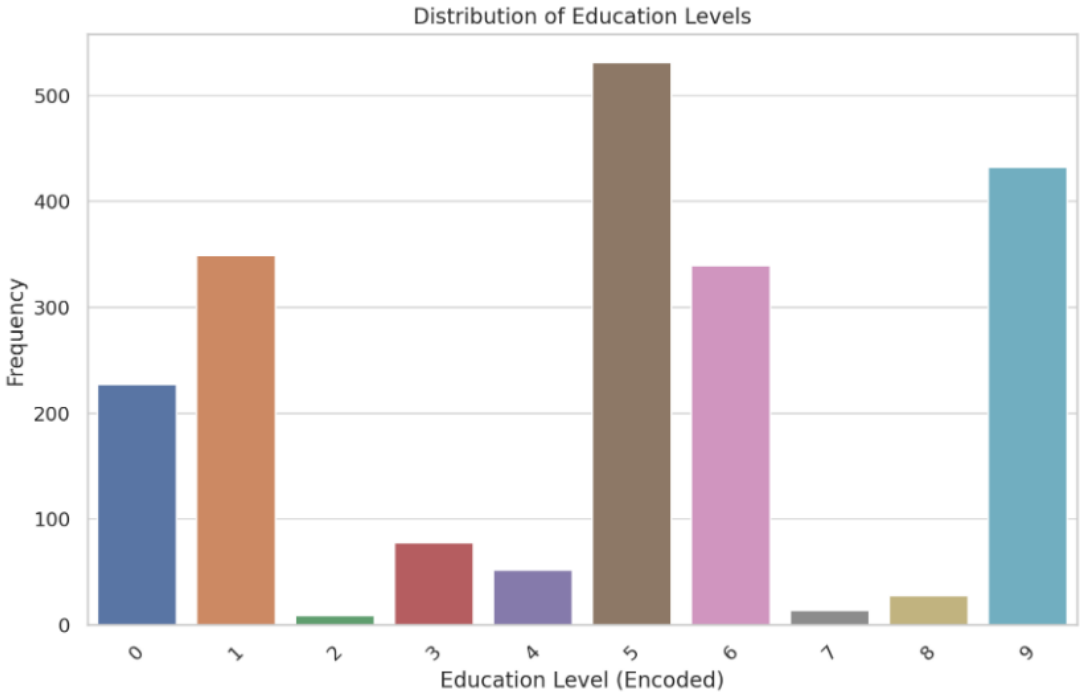

Figure 1: Distribution of Education Levels

Visualization:

Graph Description: The graph displays the frequency of different education levels among election winners.

Insights: The plot shows a predominance of certain education levels, such as 'Graduate' and 'Post Graduate', indicating a higher educational attainment among many candidates. This insight suggests a potential correlation between higher education levels and election success.

2.1.2 Criminal Cases vs. Education Level:

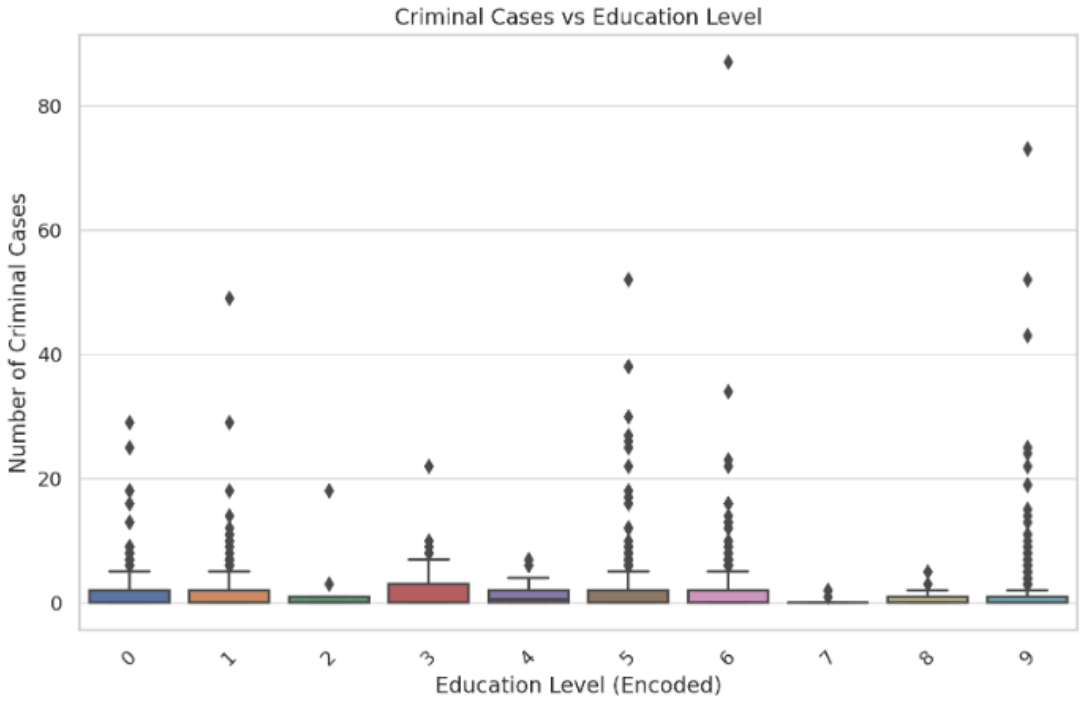

Figure 2: Criminal Cases vs Education Level

Visualization:

Graph Description: This boxplot illustrates the distribution of criminal cases across various education levels.

Insights: Lower education levels tend to have a wider range and higher median of criminal cases, whereas higher education levels generally exhibit fewer criminal cases. This pattern could influence the model's ability to predict education based on the criminal background.

2.1.3 Total Assets vs. Education Level:

Visualization:

Graph Description: A boxplot showing the variation in total assets declared across different education levels.

Insights: There appears to be an increasing trend in assets as education level increases. This observation may be utilized to predict education levels from financial disclosures.

2.1.4 Liabilities vs. Education Level:

Visualization:

Graph Description: The plot compares the liabilities across various education levels.

Insights: Similar to assets, higher education levels often report higher liabilities, which might indicate more complex financial activities or greater access to credit among highly educated candidates.

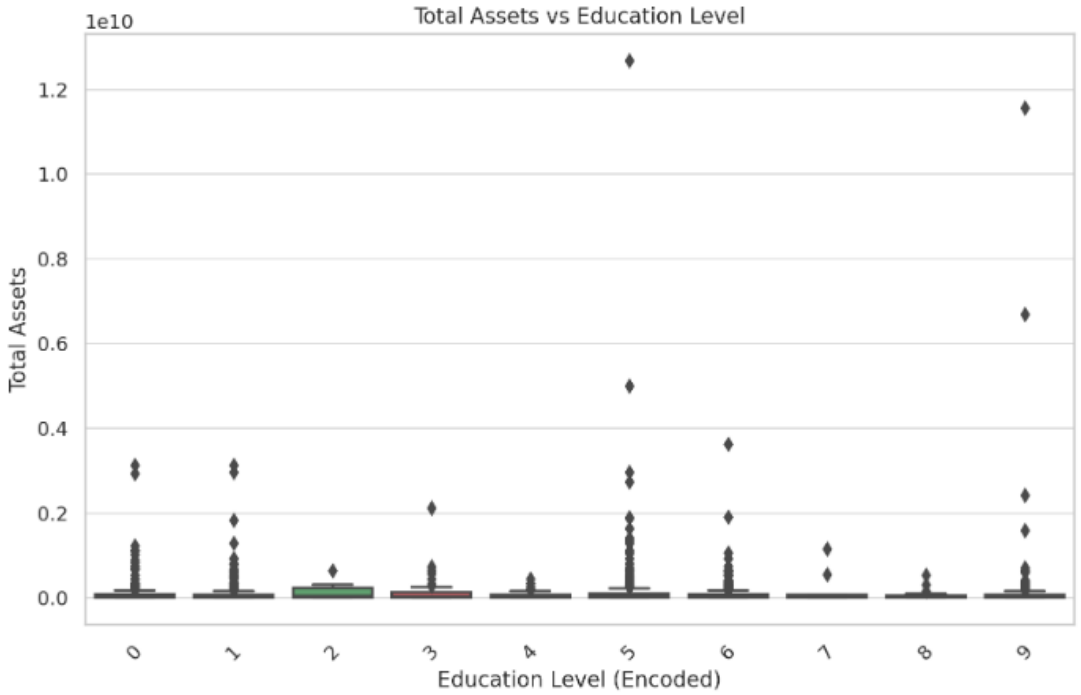

Figure 3: Total Assets vs Education Level

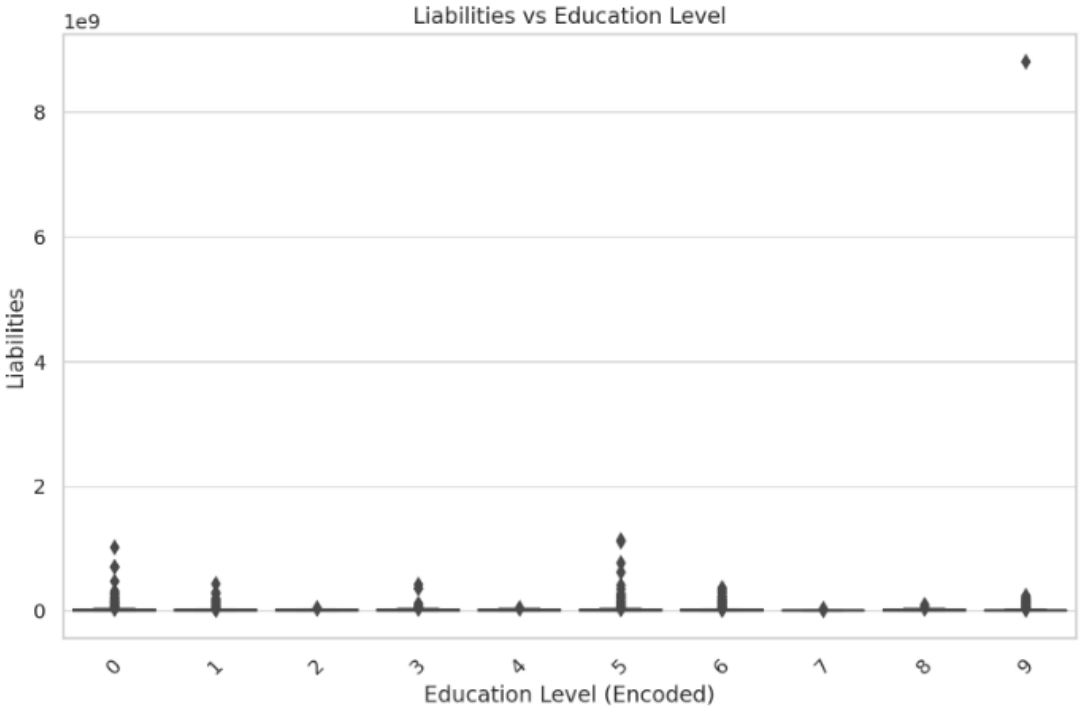

Figure 4: Liabilities vs Education Level

2.1.5 Distribution of Criminal Cases:

Visualization:

Graph Description: A histogram depicting the frequency distribution of the number of criminal cases per candidate.

Insights: Most candidates have few or no criminal cases, with a few exceptions having a significantly higher number. This skewed distribution suggests that criminal cases are an important but variably impacting feature on the dataset.

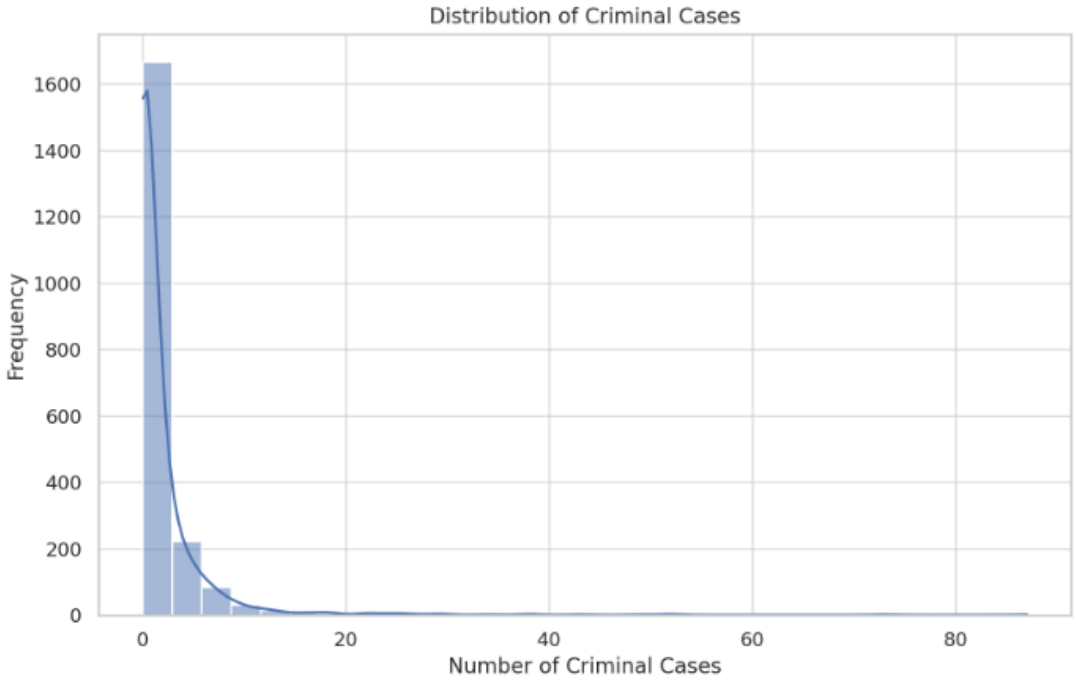

Figure 5: Distribution of Criminal Cases

3 Results

The outcomes of the predictive modeling are quantitatively summarized below, providing an evaluation of performance based on the competition's scoring metrics.

Final F1 Score: Private Score: 0.11648 Public Score: 0.12580 The F1 score, which harmonizes precision and recall, is crucial in this context due to the multi-class nature of the classification and the imbalance in the distribution of classes.

Leaderboard Rank: Unfortunately, a technical error occurred during the final submission process within the stipulated deadline, where the platform inadvertently registered the submission as an Excel file rather than the required CSV format. This necessitated a resubmission past the deadline, which may have affected the leaderboard ranking. Consequently, the final leaderboard rank is not available at the time of reporting and will be updated post-clarification with the competition organizers.

4 References

The analysis and methodologies implemented in this report were informed by a combination of primary dataset sources, machine learning theory, and best practices in data science. Below is the list of references that were instrumental throughout the completion of this assignment:

Election Commission of India. (n.d.). Data on State and UT election winners across India. Available at the official website of the Election Commission of India. Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python, Journal of Machine Learning Research 12, 2825-2830. McKinney, W. et al. (2010). Data Structures for Statistical Computing in Python, Proceedings of the 9th Python in Science Conference. Harris, C.R., Millman, K.J., van der Walt, S.J. et al. (2020). Array programming with NumPy. Nature 585, 357–362. Alammar, J. (2018). A Gentle Visual Intro to Data Analysis in Python Using Pandas. [Blog post]. Retrieved from Jay Alammar's blog. McKinney, W. (2012). Python for Data Analysis. O'Reilly Media, Inc. Hyndman, R. J., and Athanasopoulos, G. (2018). Forecasting: principles and practice, 2nd edition, OTexts: Melbourne, Australia. [Online textbook]. Retrieved from OTexts website. Kaggle. (n.d.). Competition Discussion Forums. [Online forum]. Available at Kaggle website. Each of these sources has contributed to the underlying methodology, the selection and application of statistical models, and the interpretation of data analysis conducted within this report.

5 Code -

```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, f1_score
```

```
from sklearn.preprocessing import LabelEncoder

# Load data
train_data = pd.read_csv('train.csv')
test_data = pd.read_csv('test.csv')

# Assuming 'Education' is the target column and needs to be predicted
X_train = train_data.drop(['Education'], axis=1)
y_train = train_data['Education']

# Encoding categorical variables if any
for column in X_train.columns:
    if X_train[column].dtype == type(object):  # if column is categorical
        le = LabelEncoder()
        X_train[column] = le.fit_transform(X_train[column])
        if column in test_data.columns:
            test_data[column] = le.transform(test_data[column])

# Prepare test data - assuming it has the same structure minus the target
X_test = test_data.drop(['Education'], axis=1)

# Create a RandomForest Classifier
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model
rf_model.fit(X_train, y_train)

# Predict on the training set to see in-sample accuracy
train_predictions = rf_model.predict(X_train)
print("Training F1 Score:", f1_score(y_train, train_predictions, average='weighted'))

# Predict on test set
test_predictions = rf_model.predict(X_test)

# If test labels are available, we can calculate the F1 score
if 'Education' in test_data.columns:
    y_test = test_data['Education']
    print("Test F1 Score:", f1_score(y_test, test_predictions, average='weighted'))

# Classification report
if 'Education' in test_data.columns:
    print(classification_report(y_test, test_predictions))
```