Exploratory Data Analysis Report

Mihir Mallick 21CS30031
Shashwat Kumar 21CS30047
Tejas Kumar 21CS30021

September 9, 2024

Abstract

This report details an Exploratory Data Analysis (EDA) performed on a dataset compiled following the shooting incident involving former US President Donald Trump. The analysis explores demographic and professional data from a nationwide sample of individuals, categorized by a probability score indicating their potential involvement in the crime. The study aims to uncover patterns and correlations that assist in distinguishing between likely criminals and innocents. This exploration includes frequency distributions, correlations, and key visualizations to provide insights that may support ongoing investigative efforts.

For further details, you can access the Colab Notebook.

1 Introduction

On a recent Saturday evening, a shocking event unfolded at an outdoor rally in Pennsylvania where former US President Donald Trump was critically injured in a shooting incident. In response to this, an intelligence team was tasked with compiling a comprehensive dataset through a nationwide random sampling of individuals. This dataset includes crucial demographic and professional details such as age, gender, occupation, and notably, a probability score indicating the likelihood of each individual being involved in criminal activities.

The objective of this Exploratory Data Analysis (EDA) is to delve deep into the dataset to extract actionable insights and discern patterns that could potentially pinpoint characteristics common among individuals with a higher probability of being criminals. This analysis incorporates a variety of statistical techniques to explore the underlying structure of the data, assess relationships between different variables, and visualize key distributions and correlations.

The EDA is structured to first address missing values with appropriate imputation strategies, followed by a detailed examination of the distribution of key variables. Multivariate analysis further elucidates the relationships between demographic characteristics and the assigned probability scores. The insights derived from this analysis are critical for enhancing the effectiveness of the ongoing investigation, aiming to isolate potential suspects from innocent sampled individuals.

2 Data Preparation and Analysis

2.1 Handling Missing Values

During the initial data exploration, various strategies were employed to handle missing values. For individuals under the age of 18, the missing values in the `maritalstatus` column were filled with "Never-married". For other missing values in `maritalstatus`, the mode of the column was used to impute values. Similarly, missing values in `race` and `sex` were filled using their respective modes. The column `hoursperweek` was handled based on the `workclass` column: entries labeled "Without-pay" were assigned 0, while missing values for other work classes were filled using the median.

2.2 Univariate Analysis

Univariate analysis was conducted for both categorical and numerical data:

- **Categorical Data:** Frequency distributions for categorical variables such as `workclass`, `maritalstatus`, `race`, `sex`, and others were analyzed using value counts. Visualizations included histograms and pie charts. For instance, a bar chart visualized the distribution of countries in the `native` column, while pie charts were employed for `sex` and `race` distributions.

- **Numerical Data:** Numerical columns like `age`, `educationno`, `capitalgain`, `capitalloss`, and `hoursperweek` were analyzed with histograms. This helped in identifying the distribution, skewness, and spread of these variables.

2.3 Multivariate Analysis

A correlation matrix was created to explore relationships between numerical variables. This matrix was visualized using a heatmap, which provided insights into the strength of correlations between variables such as `age`, `educationno`, `capitalgain`, and others.

2.4 Tools and Libraries Used

- **Pandas:** Used for data manipulation and handling missing values.

- **Matplotlib:** Utilized for creating visualizations such as histograms, bar charts, and pie charts.

- **Seaborn:** Applied for the heatmap visualization of the correlation matrix.

- **NumPy:** Employed for numerical computations, particularly in setting up the angles and positions for pie chart annotations.

3 Naive Bayes Implementation from Scratch

3.1 Data Preprocessing

Before implementing Naive Bayes, the categorical columns such as `workclass`, `education`, `maritalstatus`, `race`, `sex`, and others were converted into numerical codes using the `cat.codes` function in `pandas`. The target variable, `Possibility`, was also transformed into a binary form, where a value greater than 0.5 was labeled as 1, indicating a positive class.

3.2 Naive Bayes Algorithm

We implemented the Naive Bayes classification algorithm from scratch. The key steps involved are:

- **Class Separation:** We separated the data by class labels using a helper function `separate_by_class`.
- **Data Summarization:** For each class, the mean, standard deviation, and count were calculated for every feature using the `summarize_data` function.
- **Probability Calculation:** For each feature of the input, the Gaussian probability density function was used to compute the likelihood based on the mean and standard deviation (The likelihood is assumed to be a Gaussian and the prior is assumed to simply the fraction of the data points in the training dataset having that feature value). This was done using the `calculate_probability` function to ensure smooth computations, even with zero variance by adding a small value.
- **Prediction:** For each input, class probabilities were computed, and the class with the highest probability was selected as the prediction.

3.3 Model Evaluation

The model was trained on 80% of the dataset, and predictions were made on the remaining 20%. The confusion matrix was computed, followed by the calculation of precision, recall, F1 score, and accuracy using custom helper functions. The Naive Bayes classifier achieved the following metrics:

- **Precision:** 0.68
- **Recall:** 0.31
- **F1 Score:** 0.430233
- **Accuracy:** 0.788793

3.4 Comparison with Scikit-Learn Models

The results of our Naive Bayes implementation were compared with other classification models from the `scikit-learn` library, including Support Vector Machines (SVM), Decision Trees, and K-Nearest Neighbors (KNN). The following table summarizes the performance of each model:

Model	**Accuracy**	**F1 Score**
Naive Bayes (Custom)	0.791149	0.444444
Naive Bayes (Sklearn)	0.788793	0.430233
SVM	0.792309	0.400191
Decision Tree	0.815680	0.631055
KNN	0.832587	0.652204

Table 1: Comparison of Model Performance

4 Results

4.1 Histograms, Pie Charts and Scatter Plots

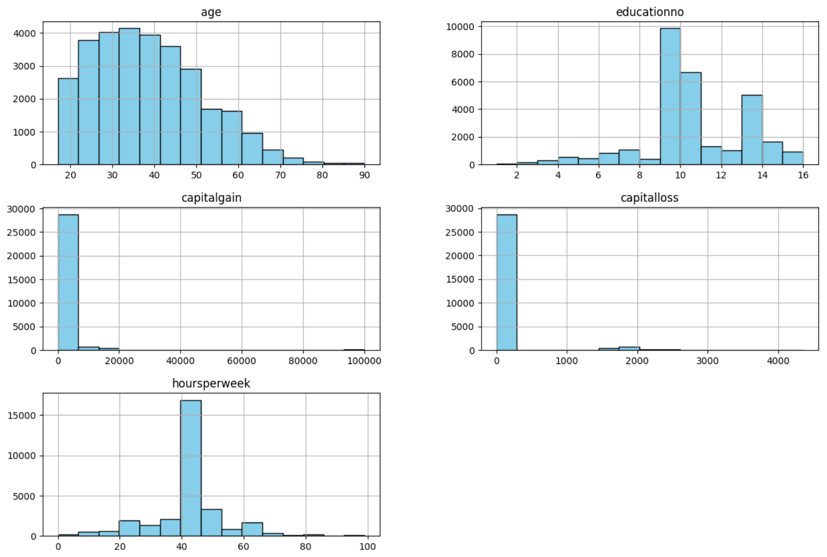

Figure 1: Histogram of Age, Education number, Capital Gain, Capital Loss, and Hours per week

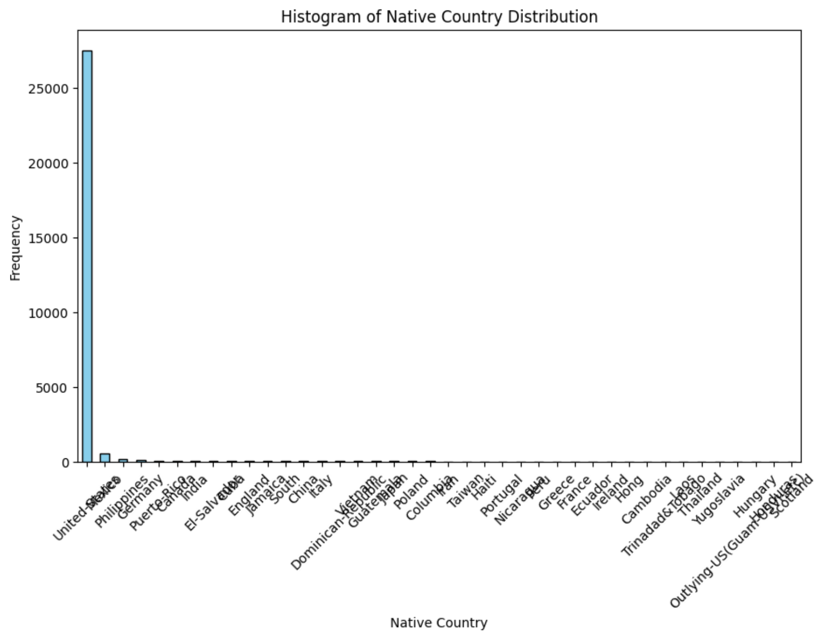

Figure 2: Histogram of Native Country Distribution

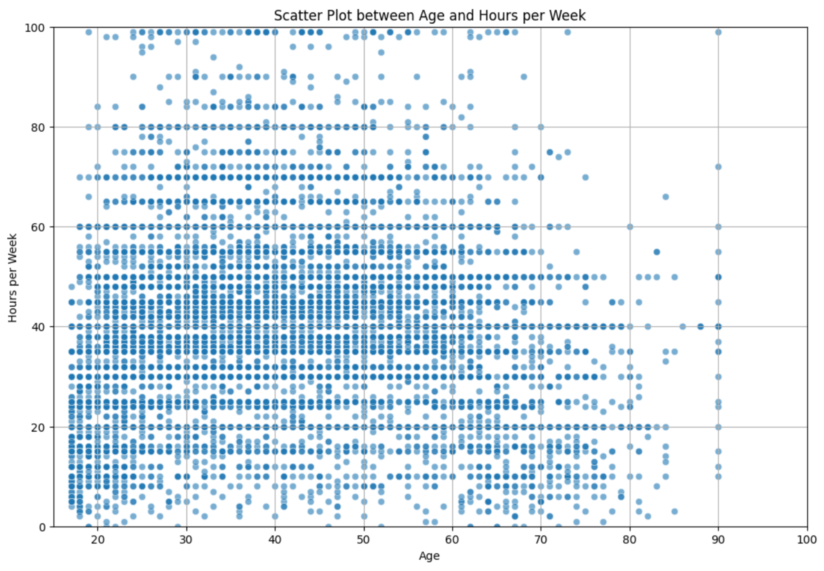

Figure 3: Scatter plot between Age and Hours per Week

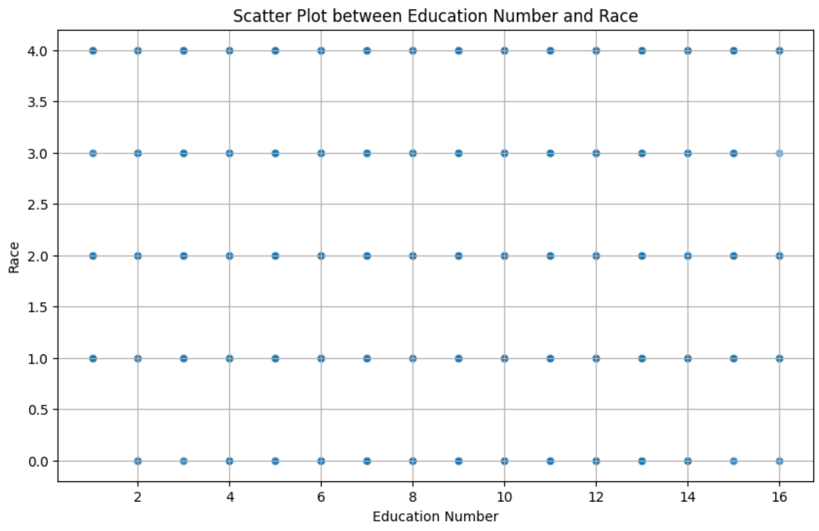

Figure 4: Scatter plot between Education Number and Race (Encoding : 0 = Amer-Indian-Eskimo, 1 = Asian-Pac-Islander, 2 = Black, 3 = Other, 4 = White)

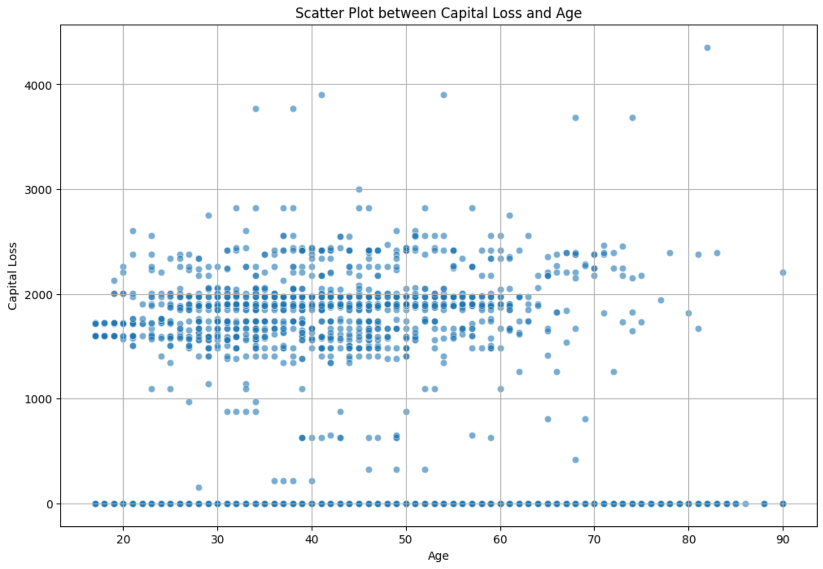

Figure 5: Scatter plot between Capital Loss and Age

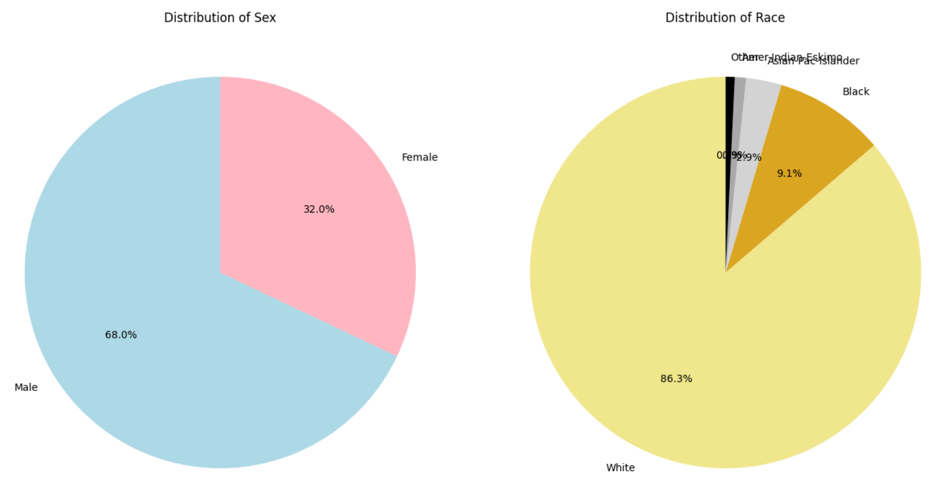

Figure 6: Pie charts showing the distribution of Sex and Race within the dataset

4.2 Correlation Matrix

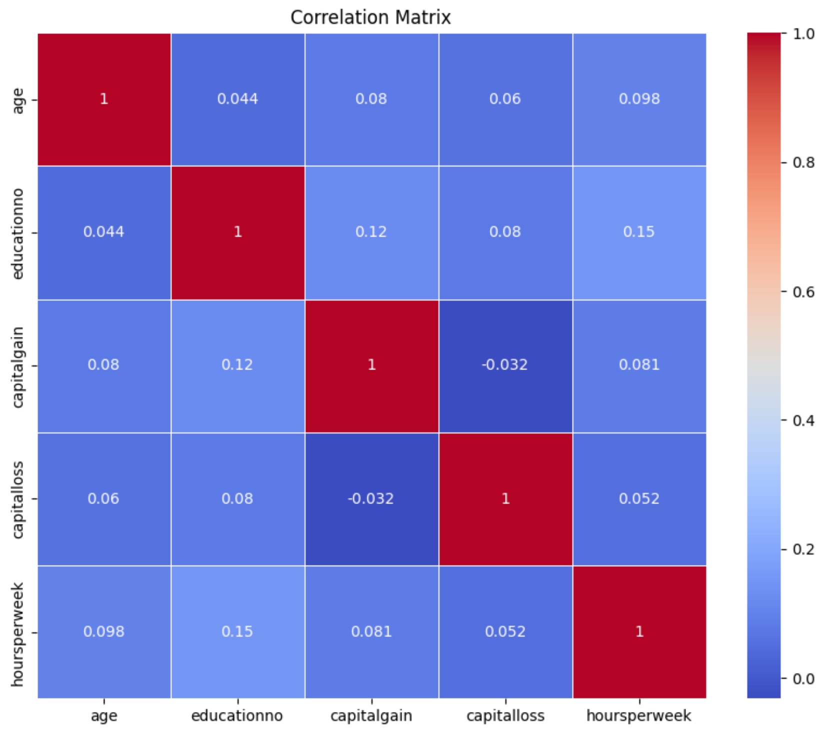

Figure 7: Correlation matrix of numerical variables

4.3 Voting Ensemble Model Overview

The `VotingEnsemble` class implements a basic ensemble learning method that combines the predictions from multiple classifiers using majority voting. The model includes:

- Support Vector Machine (SVM)
- Decision Tree Classifier
- K-Nearest Neighbors (KNN)
- Precomputed Naive Bayes predictions

The ensemble is trained using the `fit` method, and the `predict` method combines the predictions from all classifiers to make a final decision based on majority voting. Additionally, the `plot_accuracies` method plots the accuracy of individual models compared to the ensemble accuracy.

4.4 Ensemble Accuracy Evaluation

The accuracy of the ensemble model is compared against the individual models, and the results are visualized in a bar chart.

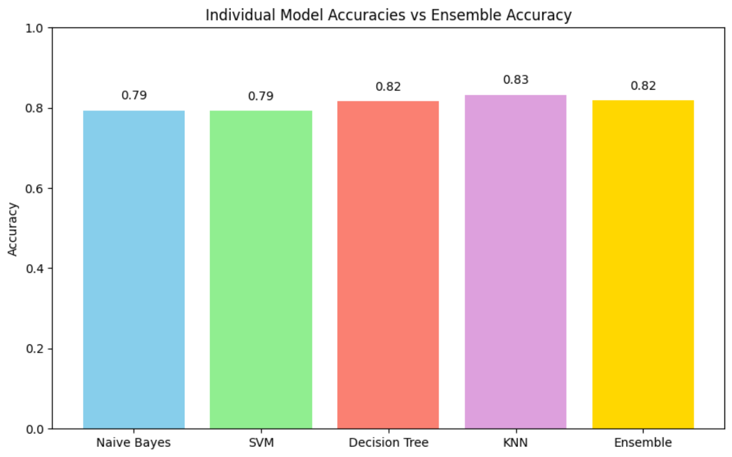

Figure 8: Comparison of Individual Model Accuracies vs Ensemble Model Accuracy

The plot compares the accuracy of Naive Bayes, SVM, Decision Tree, and KNN with the ensemble accuracy.

5 Future Work

We made some assumptions on the hyper parameters like the likelihood for the Naive Bayes to be Gaussian, prior, the train-validation split ratio, imputing the missing values etc.

One could experiment with different values, methods and assumptions of the above and obtain similar or even better results.

More advanced ensemble methods could be explored to potentially improve performance beyond the simple majority voting used in the current implementation. Notable approaches include:

- **Stacking**: This technique involves training a meta-model to combine the predictions of several base models. The base models are first trained on the dataset, and their predictions are then used as inputs to the meta-model, which learns to aggregate these predictions in an optimal manner.

- **Boosting**: Boosting methods, such as AdaBoost and Gradient Boosting, build ensembles by sequentially training models. Each new model attempts to correct the errors made by the previous models. This process helps to improve the overall performance of the ensemble by focusing on hard-to-classify instances.