Project Report

Analyzing Income Inequality using K-Nearest Neighbors

Author:

Syed Faizan

November 8th, 2024

Contents

Introduction

The purpose of this report is to explore income classification among US citizens using census data, focusing on various demographic and occupational attributes such as education, occupation, gender, and race. With the ongoing commitment to ensuring **fair and equal pay** in the workforce, this analysis is intended to build a predictive model capable of distinguishing between low and high-income individuals. By understanding the features that contribute to income level, this model can support policy recommendations aimed at addressing income disparities.

The analysis is divided into two primary parts. **Part 1** outlines the EDA and the data preprocessing steps, ensuring data quality and reliability for modeling purposes. This involves identifying and addressing missing values, encoding categorical features, and standardizing numeric variables. In **Part 2**, we apply the *K-Nearest Neighbors (KNN)* classification algorithm to predict income level. We explore key variables that contribute to income prediction, optimize model parameters, and evaluate model performance using accuracy, ROC curves, and AUC metrics.

Through this study, we aim to highlight significant patterns in income-related factors, providing insights that could inform efforts to enhance **income equality** in the US workforce.

Description of Packages Used

```python
# Import necessary libraries for EDA
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# Load the dataset
data = pd.read_csv(r'C:\Users\sfaiz\OneDrive\Desktop\ALY 6020 Project Module 1\adult.csv')


# Display the first few rows and basic info to understand structure
data.head()
```

Figure 1: Requirements of the python environment

The above figure illustrates the initial steps of an exploratory data analysis (EDA) in Python, showcasing essential procedures to import and prepare a dataset. The script begins by importing necessary libraries, which are fundamental for performing data manipulation, visualization, and numerical analysis in Python. Specifically, `pandas` is imported as `pd` for data manipulation tasks, allowing for structured data handling in DataFrames. The `numpy` library, imported as `np`, facilitates efficient numerical computations, especially when dealing with arrays and mathematical operations that extend beyond native Python capabilities. Additionally, `seaborn`, a statistical data visualization library, is imported as `sns` to generate aesthetically pleasing and informative plots. Finally, `matplotlib.pyplot` is imported as `plt`, providing a flexible plotting interface widely used for creating custom charts and visualizations in Python.

Following the library imports, the script proceeds to load a dataset. This is achieved through the `pd.read_csv()` function, which reads data from a specified CSV file path. In this instance, the file path points to a specific location on the user's device, suggesting that this data file, titled `adult.csv`, is stored within a designated project folder on OneDrive. The use of `r''` before the file path denotes a raw string format in Python, which ensures that backslashes are treated literally rather than as escape characters. This approach is especially relevant for Windows file paths, where backslashes are prevalent.

The last line of the code snippet employs the `data.head()` function, a common command in **pandas** to display the first few rows of a dataset. This function provides a quick preview

of the dataset's structure, helping the analyst understand the nature and type of data they are working with. By examining the initial rows, the analyst can gain insight into key aspects such as column names, data types, and preliminary values, which are essential for making informed decisions about subsequent data preprocessing and analysis steps.

Overall, this segment of code represents foundational steps in data analysis, emphasizing best practices in data handling and preparation. The use of clearly structured imports and essential commands exemplifies a methodical approach, setting a foundation for a systematic and reproducible EDA process. Such initial steps are crucial for ensuring data integrity and enabling deeper analytical insights in the subsequent stages of data exploration and modeling.

Part 1: Exploratory Data Analysis (EDA) and Data Cleansing and Pre-processing

	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
0	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
1	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
2	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
3	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K
4	37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	White	Female	0	0	40	United-States	<=50K

Figure 2: The structure of the Dataset

Overview of the Dataset

The above figure presents a tabular view of a subset of the dataset, which contains demographic and occupational attributes for a sample of individuals. Each row represents an individual, with multiple columns providing detailed characteristics that are useful for data analysis and predictive modeling tasks. The columns are a mix of numerical and categorical data types, highlighting essential variables related to personal demographics, employment, education, and income level.

The first column, labeled `39`, represents the **age** of each individual. This column is numerical, offering direct quantitative data about the age distribution within the dataset. The second column, `State-gov`, contains categorical data representing the **workclass** of each individual, indicating their employment sector or type, such as "Self-emp-not-inc" or

"Private." The next column, `77516`, labeled with the final weight identifier **fnlwgt**, is also numerical and is commonly used in survey data to adjust for unequal sampling probabilities.

Subsequent columns provide information on `education` and `education_num`. The `education` column shows each individual's highest completed level of education in categorical terms, like "Bachelors" or "HS-grad," while `education_num` quantifies these educational levels with corresponding numerical values, enabling a more structured comparison between educational attainments. The column labeled `Never-married` represents **marital status**, capturing various states of personal relationships, such as "Married-civ-spouse" or "Divorced."

Following the marital status column, the table includes `occupation`, detailing the job type for each individual, and `relationship`, specifying the individual's relationship within their household, such as "Husband," "Wife," or "Not-in-family." These categories are crucial for understanding family structures and professional roles, which may correlate with income levels.

The `race` and `sex` columns contain demographic information, with categorical values representing the **racial background** (e.g., "White," "Black") and **gender** ("Male" or "Female") of each individual. This data enables analyses that investigate disparities based on race or gender.

The columns labeled `2174` and `0` provide numerical values for **capital gain** and **capital loss**, respectively. These financial variables are significant in evaluating the economic background of individuals and assessing wealth-related attributes. The column labeled `40` represents **hours per week**, indicating the average weekly working hours of each individual.

The final columns contain `native_country`, which specifies each person's country of origin, and $\leq 50K$, indicating income classification. Here, income is classified into two categories: $\leq 50K$ and $> 50K$, representing lower and higher income groups, respectively. This income attribute serves as the target variable for modeling tasks focused on predicting income levels based on the provided attributes.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48841 entries, 0 to 48840
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   39              48841 non-null  int64
 1   State-gov       48841 non-null  object
 2   77516           48841 non-null  int64
 3   Bachelors       48841 non-null  object
 4   13              48841 non-null  int64
 5   Never-married   48841 non-null  object
 6   Adm-clerical    48841 non-null  object
 7   Not-in-family   48841 non-null  object
 8   White           48841 non-null  object
 9   Male            48841 non-null  object
 10  2174            48841 non-null  int64
 11  0               48841 non-null  int64
 12  40              48841 non-null  int64
 13  United-States   48841 non-null  object
 14  <=50K           48841 non-null  object
dtypes: int64(6), object(9)
memory usage: 5.6+ MB
```

Figure 3: Dataset details

The above figure provides a comprehensive overview of the structure and composition of the dataset, displaying crucial information about each column, including its name, data type, and the count of non-null values. The dataset comprises 48,841 entries across 15 columns, and it utilizes approximately 5.6 MB of memory. This summary aids in understanding the data's format, completeness, and storage requirements, which are essential for effective data preprocessing and analysis.

Each column represents a distinct attribute, with the **Column** field listing the names of each attribute in the dataset. The columns include both numerical and categorical data, reflecting diverse types of information such as demographics, employment, and financial characteristics. The **Non-Null Count** field indicates that all columns contain 48,841 non-null values, implying that the dataset is complete and has no missing data. This completeness

is beneficial for analysis, as it eliminates the need for imputation or data exclusion strategies that could impact the integrity of the dataset.

The **Dtype** field specifies the data type of each column, categorized as either `int64` or `object`. The `int64` data type is associated with numerical variables, including columns labeled `39`, `77516`, `13`, `2174`, `0`, and `40`. These numerical columns represent variables such as age, sampling weight, education number, capital gain, capital loss, and hours per week, respectively. Numerical data types facilitate statistical computations and numerical transformations, which are often necessary in predictive modeling tasks.

The remaining columns are designated as `object` data types, indicating categorical variables. These categorical columns include attributes such as `State-gov` (representing the work sector), `Bachelors` (education level), `Never-married` (marital status), `Adm-clerical` (occupation), `Not-in-family` (relationship), `White` (race), `Male` (gender), and `United-States` (native country). Categorical data types are essential for capturing non-numeric information, allowing for analysis of qualitative attributes that might correlate with other variables, such as income.

The final column, labeled $\leq 50K$, represents the **target variable**, categorizing income levels into binary classes. This column's `object` data type indicates a categorical classification, which is common in datasets aimed at classification tasks. The binary nature of this column makes it suitable for supervised learning models, such as logistic regression or K-nearest neighbors, which aim to predict income class based on the provided attributes.

Data Cleansing

Missing Values:

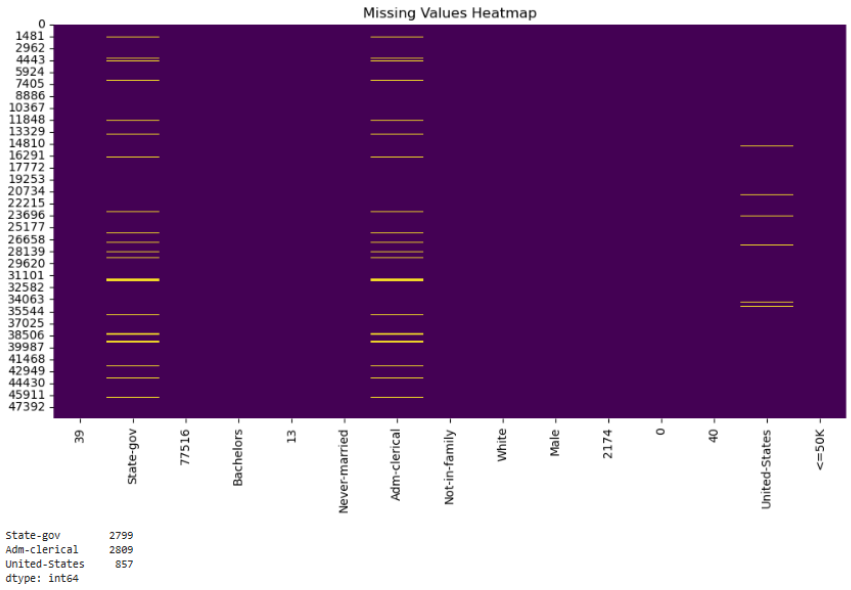

Figure 4: Missing Values Heat Map

The above figure presents a heatmap visualization of missing values within the dataset. Each row corresponds to an individual data entry, while each column represents a specific attribute. The heatmap provides a visual assessment of data completeness, with purple regions indicating available data and yellow lines representing missing values. This layout allows for an immediate understanding of the distribution and concentration of missing data across different columns. Such an analysis is essential in the early stages of data preprocessing, as it informs strategies for handling incomplete data.

In this dataset, three columns—`State-gov`, `Adm-clerical`, and `United-States`—exhibit missing values. Specifically, the `State-gov` column, which represents the work sector or employment class, has 2,799 missing values. The `Adm-clerical` column, indicating occupational type, contains 2,809 missing values. The `United-States` column, representing an individual's country of origin, has 857 missing entries.

Statistical Descriptive Summary of the Variables

	age	fnlwgt	education_num	capital_gain	capital_loss	hours_per_week	income
count	48841.000000	4.884100e+04	48841.000000	48841.000000	48841.000000	48841.000000	48841.000000
mean	38.643578	1.896664e+05	10.078029	1079.045208	87.504105	40.422391	0.239287
std	13.710650	1.056039e+05	2.570965	7452.093700	403.008483	12.391571	0.426652
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000	0.000000
25%	28.000000	1.175550e+05	9.000000	0.000000	0.000000	40.000000	0.000000
50%	37.000000	1.781470e+05	10.000000	0.000000	0.000000	40.000000	0.000000
75%	48.000000	2.376460e+05	12.000000	0.000000	0.000000	45.000000	0.000000
max	90.000000	1.490400e+06	16.000000	99999.000000	4356.000000	99.000000	1.000000

Figure 5: The summary of descriptive statistics for the numerical attributes

The above figure provides a summary of descriptive statistics for the numerical attributes within the dataset. This statistical overview includes essential metrics such as the **count**, **mean**, **standard deviation** (*std*), as well as the **minimum**, **maximum**, and various percentiles (25%, 50%, and 75%) for each numerical column. These values offer a foundational understanding of the distribution and central tendencies of key variables, enabling preliminary insights into the dataset's structure.

The `age` variable shows a mean of 38.64 years, with a minimum of 17 and a maximum of 90, indicating a wide age range among individuals in the dataset. The 25th percentile at 28 years and the 75th percentile at 48 years suggest that the majority of individuals fall within early adulthood to middle age. The standard deviation of 13.71 indicates moderate variability in age, reflecting a diverse age distribution.

The `fnlwgt` column, representing the final sampling weight, has a mean of 189,666 with a standard deviation of 105,603, and a maximum value nearing 1.5 million. This variable's high variability and large range, from a minimum of 12,285 to a maximum of nearly 1.5 million, signify its role in adjusting for population representation within sampled data, especially useful in census datasets. The wide spread suggests adjustments for individuals based on various demographic criteria.

The `education_num` variable, representing years of education, has a mean of 10.08 years, with most individuals having between 9 and 12 years of education (as indicated by the 25th and 75th percentiles). The maximum value of 16 years corresponds to higher education levels, while the standard deviation of 2.57 reflects some educational diversity.

In terms of financial variables, `capital_gain` and `capital_loss` display high levels of skewness, with most entries at zero, as indicated by the median (50% percentile) of zero for both variables. However, their maximum values reach 99,999 for capital gain and 4,356 for capital loss, suggesting a few individuals with significant financial gains or losses. The high standard deviation values for these columns further underscore the skewed nature of these distributions.

The `hours_per_week` variable, with a mean of 40.42 hours, closely aligns with standard full-time work hours in the United States. A relatively low standard deviation of 12.39 implies that most individuals work between 40 and 45 hours per week, as reflected in the 50% and 75% percentiles.

Lastly, the `income` variable, acting as the target attribute, has a mean value of 0.239, suggesting that around 23.9% of the individuals in the dataset fall into the high-income category (coded as 1). This class imbalance is notable and may require consideration in modeling techniques to ensure balanced predictive performance.

In summary, this descriptive statistics table provides a valuable snapshot of the dataset's numerical attributes, highlighting central tendencies, variability, and distributional characteristics. This statistical understanding is essential for informed data preprocessing, feature engineering, and model selection.

Descriptive statistics of Categorical Variables

	workclass	education	marital_status	occupation	relationship	race	sex	native_country
count	48841	48841	48841	48841	48841	48841	48841	48841
unique	8	16	7	14	6	5	2	41
top	Private	HS-grad	Married-civ-spouse	Prof-specialty	Husband	White	Male	United-States
freq	36705	15784	22379	8981	19716	41761	32649	44688

Figure 6: A summary of descriptive statistics for the categorical attributes

The above figure provides a summary of descriptive statistics for the categorical attributes in the dataset, giving insights into the distribution, frequency, and unique categories for each variable. This table includes essential fields such as **workclass**, **education**, **marital status**, **occupation**, **relationship**, **race**, **sex**, and **native country**. The table details

four key metrics: *count*, *unique*, *top*, and *frequency*, each of which aids in understanding the dataset's composition and dominant categorical values.

The `count` row indicates that each of these categorical columns has 48,841 non-null entries, confirming that there are no missing values for these variables. This completeness is advantageous as it reduces the need for imputation or data exclusion strategies, ensuring the integrity of the dataset for analytical purposes.

The `unique` row displays the number of distinct categories for each variable. For example, `workclass` has 8 unique categories, such as "Private," "Self-emp-not-inc," and "State-gov," representing various employment types. The `education` column shows 16 unique categories, capturing a broad spectrum of educational levels ranging from "Preschool" to "Doctorate." Similarly, `marital_status` has 7 categories, reflecting diverse relationship statuses such as "Married-civ-spouse" and "Divorced."

The `top` row identifies the most common category within each attribute. For instance, `workclass` is dominated by the "Private" category, with a frequency of 36,705 occurrences, as shown in the `freq` row. This suggests that a substantial proportion of the individuals in the dataset are employed in the private sector. Similarly, "HS-grad" is the most frequent category within the `education` column, appearing 15,784 times, indicating that high school graduation is the most common educational attainment in this dataset. In terms of `marital_status`, "Married-civ-spouse" appears as the top category, with 22,379 instances, suggesting that a significant portion of the dataset comprises married individuals.

The `race` and `sex` columns reveal demographic insights, with "White" being the most prevalent racial category, appearing 41,761 times, and "Male" as the most frequent gender, with a count of 32,649. Additionally, the `native_country` column is dominated by "United-States," with a frequency of 44,688, implying that most individuals in the dataset are U.S. natives. The relatively low variety in certain categories, such as `sex` with only 2 unique values, indicates potential demographic imbalances within the dataset.

In summary, this descriptive overview of categorical variables provides a comprehensive understanding of the data distribution and prevalent categories within each attribute. Such insights are critical for ensuring appropriate feature engineering and data preprocessing steps, which will ultimately enhance the accuracy and interpretability of any predictive modeling

efforts applied to this dataset.

Treating missing values

```
# Impute missing values in categorical columns with the mode (most frequent value)
for column in ['State-gov', 'Adm-clerical', 'United-States']:
    mode_value = data[column].mode()[0]  # Calculate the mode
    data[column].fillna(mode_value, inplace=True)  # Fill missing values with the mode

# Verify that missing values are now handled
missing_values_post_imputation = data.isnull().sum()
missing_values_post_imputation_summary = missing_values_post_imputation[missing_values_post_imputation > 0]
missing_values_post_imputation_summary
```

Figure 7: Treating missing values in categorical columns within the dataset

The above figure presents a Python code snippet designed to address missing values in specific categorical columns within the dataset. This code follows a structured approach, implementing **mode imputation** for the categorical columns `State-gov`, `Adm-clerical`, and `United-States`. Mode imputation replaces missing values in each column with the most frequently occurring category, a technique commonly employed in categorical data processing to retain the dominant pattern within the data.

Data Cleansing

Renaming the columns and typecasting

```
# Extensive Data Cleansing: Renaming columns for clarity, handling data types


# Step 1: Renaming columns for improved readability
# Mapping ambiguous column names to more descriptive names
data.columns = [
    'age',               # '39'
    'workclass',         # 'State-gov'
    'fnlwgt',            # '77516'
    'education',         # 'Bachelors'
    'education_num',     # '13'
    'marital_status',    # 'Never-married'
    'occupation',        # 'Adm-clerical'
    'relationship',      # 'Not-in-family'
    'race',              # 'White'
    'sex',               # 'Male'
    'capital_gain',      # '2174'
    'capital_loss',      # '0'
    'hours_per_week',    # '40'
    'native_country',    # 'United-States'
    'income'             # '<=50K'
]

# Step 2: Verify the new column names and display the first few rows to confirm changes
renamed_data_head = data.head()

# Step 3: Check data types and convert them if necessary
# For example, if 'education_num' should be a categorical ordinal feature, we may convert it accordingly
data['education_num'] = data['education_num'].astype(int)  # Ensure 'education_num' is integer
data['age'] = data['age'].astype(int)  # Ensure 'age' is integer



# Display cleaned data summary
cleaned_data_info = data.info()
renamed data head, cleaned data info
```

Figure 8: Data cleansing

The above figure demonstrates a comprehensive approach to data cleansing by renaming columns for improved clarity and verifying data types to ensure consistency. This code snippet is divided into structured steps that facilitate the transformation of ambiguous column names into more descriptive ones and make essential adjustments to data types for analytical accuracy.

Step 1 begins by renaming the columns within the `data` DataFrame. The new column names provide enhanced readability by replacing original ambiguous titles with descriptive alternatives. For instance, the column originally labeled as `39` is renamed to `age`, while `State-gov` is relabeled as `workclass`. This renaming process enhances the interpretability of the dataset, making it easier for data analysts to understand each attribute's significance without needing additional documentation or cross-referencing. This practice of renaming columns is especially valuable in collaborative projects or complex data analyses, where clarity of variable names is crucial for reducing misinterpretation.

Step 2 entails verifying the new column names by displaying the initial rows of the modified dataset using the `data.head()` function, stored here as `renamed_data_head`.

Step 3 involves checking and adjusting data types where necessary. This is achieved by explicitly casting columns to appropriate data types. For example, the code converts `education_num` and `age` columns to integers using `astype(int)`.

Finally, the `data.info()` function is used to display a summary of the cleaned dataset, labeled here as `cleaned_data_info`. This output provides an overview of the updated column names, data types, and memory usage, confirming that the transformations have been applied successfully.

Data Visualization

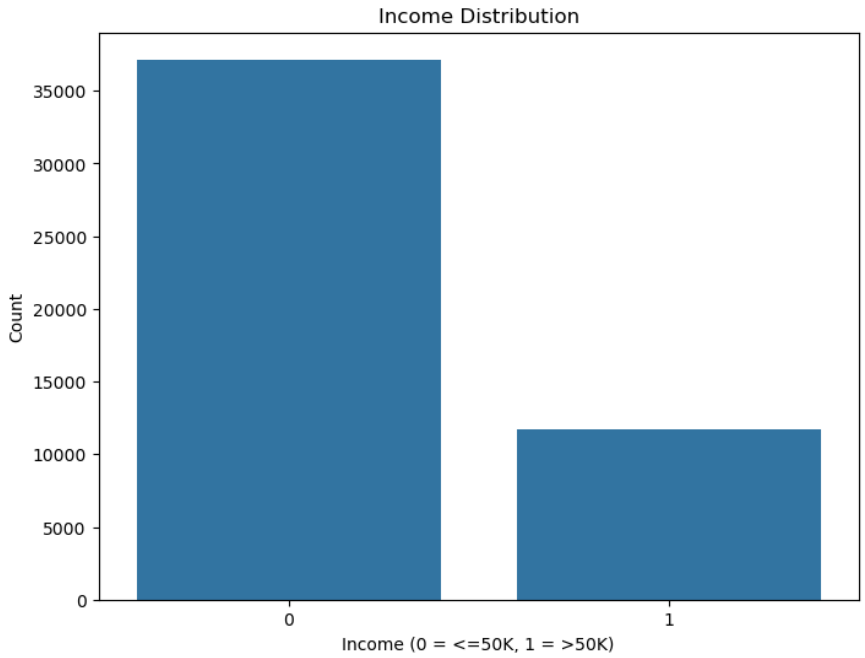

Figure 9: The distribution of the target variable

The distribution of the target variable The above figure illustrates the distribution of the target variable `income` within the dataset, depicting the count of individuals in two distinct income classes. The x-axis represents the income categories, where **0** corresponds to individuals earning $\leq 50K$ and **1** corresponds to individuals earning $> 50K$ annually. The y-axis shows the count of individuals within each income category, providing a visual

assessment of class balance in the dataset.

It is evident from the bar heights that there is an imbalance between the two income groups. The majority of individuals fall into the $\leq 50K$ income bracket, with over 35,000 entries, whereas the $> 50K$ category comprises a considerably smaller number of entries, likely around 11,000 to 12,000. This imbalance highlights a disparity in income levels within the dataset, which is common in real-world socioeconomic data.

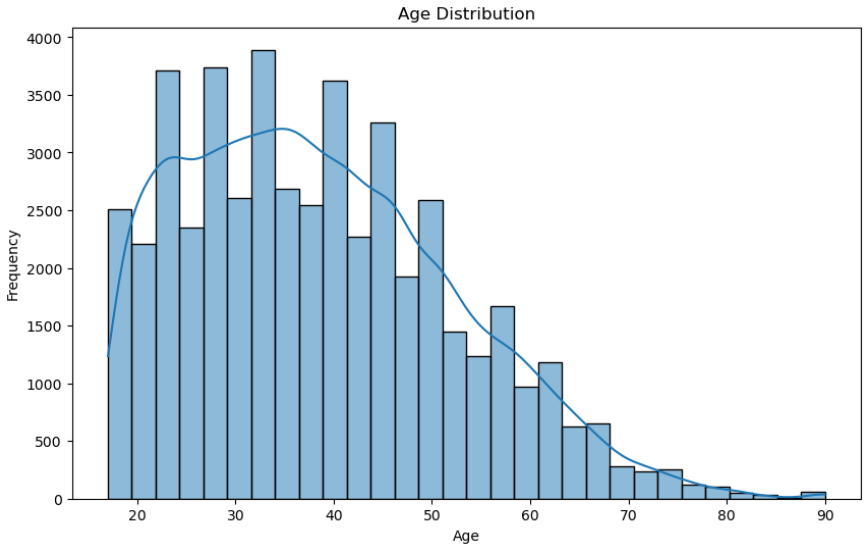

Figure 10: Distribution of `age`

The above figure illustrates the distribution of `age` within the dataset, providing a visual representation of the frequency of different age groups. The x-axis denotes **Age**, ranging from the minimum age in the dataset (around 17) to the maximum (90), while the y-axis represents the **Frequency** of individuals within each age bin. The histogram is overlaid with a density curve, offering additional insight into the distribution's shape and central tendency.

The age distribution displays a right-skewed pattern, with the highest concentration of individuals within the younger to middle-aged groups, particularly between 20 and 50 years. This pattern is indicative of a dataset dominated by individuals in their working years, which aligns with the economic and occupational context of the data. The age group around 30 to 40 years has the highest frequency, peaking close to 4,000 individuals, suggesting that this dataset may represent a typical workforce demographic where the majority are in their early to mid-career stages.

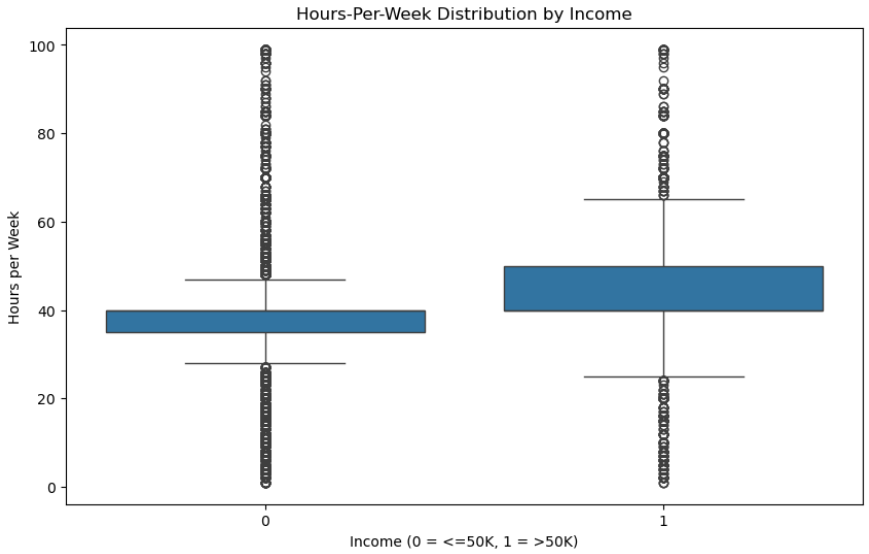

Figure 11: A box plot illustrating the distribution of `hours per week` worked

Hours-Per-Week Distribution by Income

The above figure presents a box plot illustrating the distribution of `hours per week` worked, segmented by income categories. The x-axis represents the two income classes, where **0** denotes individuals earning $\leq 50K$ annually and **1** denotes those earning $> 50K$. The y-axis measures the number of hours worked per week. This visualization is valuable for examining potential differences in working hours between lower and higher income groups, which may reflect variations in job type, work intensity, or labor market participation.

The box plot reveals that both income groups generally work close to full-time hours, with median values around 40 hours per week for each class. This central tendency suggests that a standard full-time work schedule is prevalent among both lower and higher income earners in the dataset. However, there is a slight increase in median weekly hours for the higher income group (**1**), indicating that individuals earning above 50K may, on average, work slightly longer hours than those earning less.

The spread of hours worked also differs between the two income categories. The interquartile range (IQR) for the lower income group (income **0**) is slightly narrower, suggesting that hours worked by individuals in this group are more concentrated around the median. In contrast, the higher income group exhibits a broader IQR, which implies greater variability in weekly hours among high earners. This could reflect flexible work patterns or diverse job

types among higher income individuals, some of whom may work significantly more or fewer hours.

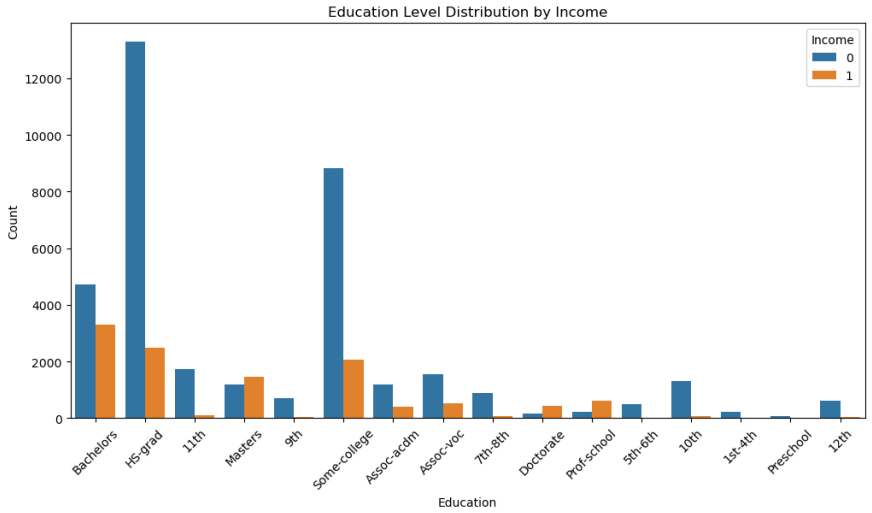

Figure 12: Education Level Distribution by Income

Education Level Distribution by Income The above figure provides a comparative bar plot of the `education` levels across two income groups in the dataset. The x-axis categorizes individuals by their highest level of education attainment, ranging from *Preschool* to *Doctorate*. The y-axis represents the **Count** of individuals within each educational category. Income levels are color-coded, with **0** (blue) representing individuals earning $\leq 50K$ and **1** (orange) representing those earning $> 50K$ annually.

The plot reveals that the income group earning $\leq 50K$ dominates across most education levels, particularly among individuals with a high school diploma (HS-grad), some college experience, or less than a high school education (e.g., 11th grade, 9th grade, or lower). The `HS-grad` category exhibits the highest count for the lower income group, with a substantial number of individuals earning below 50K. This trend suggests that, for individuals without advanced degrees, earning potential often remains constrained within the lower income bracket.

For individuals in the higher income group ($> 50K$), the distribution shifts significantly in categories associated with higher education. Notably, individuals with `Bachelors`, `Masters`, `Prof-school`, and `Doctorate` degrees show a higher representation within the $> 50K$ income category. In these categories, the gap between lower and higher income earners narrows,

reflecting that advanced education is strongly associated with a higher likelihood of achieving greater income levels. For instance, in the `Bachelors` category, there is a notable presence of individuals in both income groups, with a visible increase in the number of higher earners compared to education levels below a bachelor's degree.

The data also shows that advanced degrees (such as `Doctorate` and `Prof-school`) have a higher proportion of individuals in the $> 50K$ income bracket. This suggests that pursuing graduate or professional education significantly enhances income potential, positioning individuals for higher-paying roles or specialized fields.

In contrast, categories like `Some-college` and `HS-grad` have high counts for the lower income group, reflecting a potential ceiling on earnings without the completion of higher educational credentials.

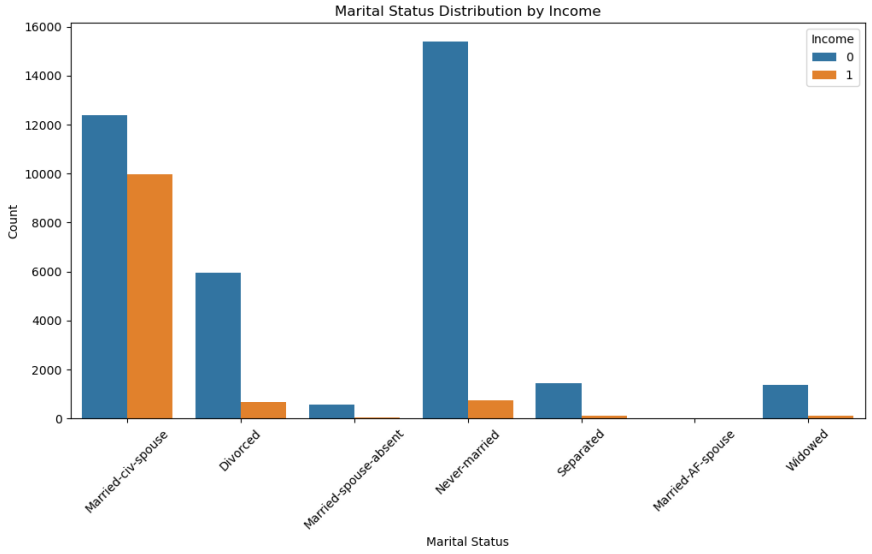

Figure 13: Comparative bar plot showing the distribution of `marital status` across two income groups

`Marital status` by income groups

The above figure presents a comparative bar plot showing the distribution of `marital status` across two income groups in the dataset. The x-axis categorizes individuals by their marital status, including categories such as *Married-civ-spouse*, *Divorced*, *Never-married*, *Separated*, *Married-AF-spouse*, and *Widowed*. The y-axis represents the **Count** of individuals within each marital status category. Income levels are color-coded, with **0** (blue)

representing individuals earning $\leq 50K$ annually and **1** (orange) representing those earning $> 50K$. This visualization allows for a detailed examination of how marital status correlates with income distribution, offering insights into the socio-economic impact of marital relationships.

The plot reveals that the `Married-civ-spouse` category, representing individuals in traditional civilian marriages, has the highest counts in both income groups, indicating that marriage is common across various income levels. However, there is a notable increase in the representation of higher income individuals ($> 50K$) within this group, with the counts for income **1** approaching those for income **0**. This trend suggests that individuals in this marital category may benefit from dual-income households or other economic advantages associated with marital status, which often contributes to higher household income.

The `Married-AF-spouse` and `Married-spouse-absent` categories, though less frequent, also demonstrate a majority representation in the lower income group. This suggests that individuals in non-traditional or long-distance marriages may face economic limitations that hinder their earning potential.

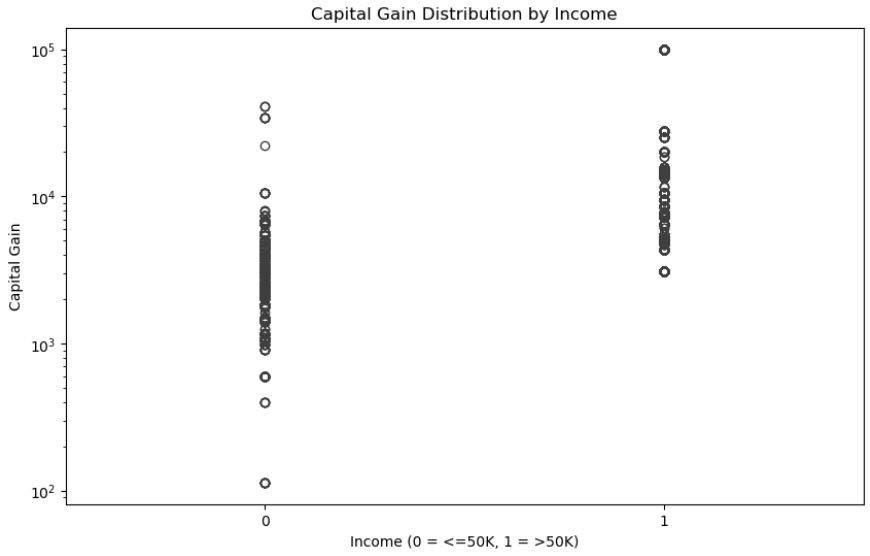

Figure 14: Capital Gain Distribution by Income

Capital Gain Distribution by Income

The above figure presents a comparative visualization of the `capital gain` distribution across two income groups within the dataset. The x-axis represents income classes, where **0**

corresponds to individuals earning $\leq 50K$ and **1** represents those earning $> 50K$. The y-axis, plotted on a logarithmic scale, captures the magnitude of **Capital Gain**, ranging from lower values (near 10^2) to the highest recorded values (approaching 10^5). This logarithmic scaling provides a clearer view of the dispersion of capital gain values, especially due to the highly skewed nature of this variable with significant outliers.

The distribution of `capital gain` reveals a distinct pattern between income groups. In the $\leq 50K$ income category, the majority of individuals have capital gains clustered around lower values, with only a sparse number of individuals reporting substantial capital gains. Conversely, within the $> 50K$ income group, there is a higher density of individuals with substantial capital gains, particularly values exceeding 10^3 and occasionally reaching 10^5. This suggests that larger capital gains are more frequently associated with the higher income group, which may reflect income-generating activities such as investments, asset sales, or dividends.

The use of a logarithmic scale emphasizes the skewness in capital gain distribution and highlights the presence of outliers, particularly among higher income earners. These outliers, with values above 10^4, are indicative of significant financial transactions or investments, which are generally less common among individuals in the lower income group. The clustering at low values for the lower income group suggests that capital gain activities are limited or minimal among individuals earning $\leq 50K$.

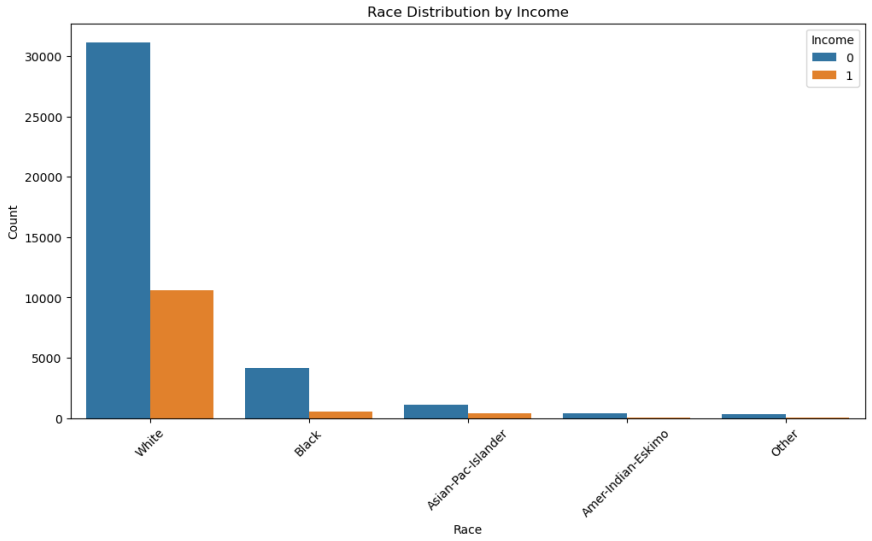

Figure 15: Race Distribution by Income

Race Distribution by Income

The above figure provides a bar plot illustrating the distribution of `race` across two income groups within the dataset. The x-axis categorizes individuals by race, including *White*, *Black*, *Asian-Pac-Islander*, *Amer-Indian-Eskimo*, and *Other*. The y-axis represents the **Count** of individuals in each racial category. Income levels are color-coded, with **0** (blue) representing individuals earning $\leq 50K$ and **1** (orange) representing those earning $> 50K$. This visualization offers insights into the racial composition of different income groups, potentially reflecting socioeconomic disparities along racial lines.

The distribution shows that the `White` racial category has the highest representation in both income groups, with a substantial disparity between those earning $\leq 50K$ and those earning $> 50K$. Specifically, the `White` group has a high count of individuals in the lower income bracket, but it also has the highest number of individuals in the higher income bracket. This suggests that, while there is diversity in income within this group, a considerable portion still falls into the lower income category.

The `Black` category, the second most populous group in this dataset, predominantly falls into the $\leq 50K$ income group, with only a small fraction in the $> 50K$ group. This distribution indicates a possible income disparity, where individuals within this racial category are more likely to earn lower incomes compared to the `White` group. Similar patterns are observed in the `Asian-Pac-Islander` and `Amer-Indian-Eskimo` categories, which also exhibit a majority in the lower income bracket, though their overall representation is significantly smaller.

Correlation Analysis

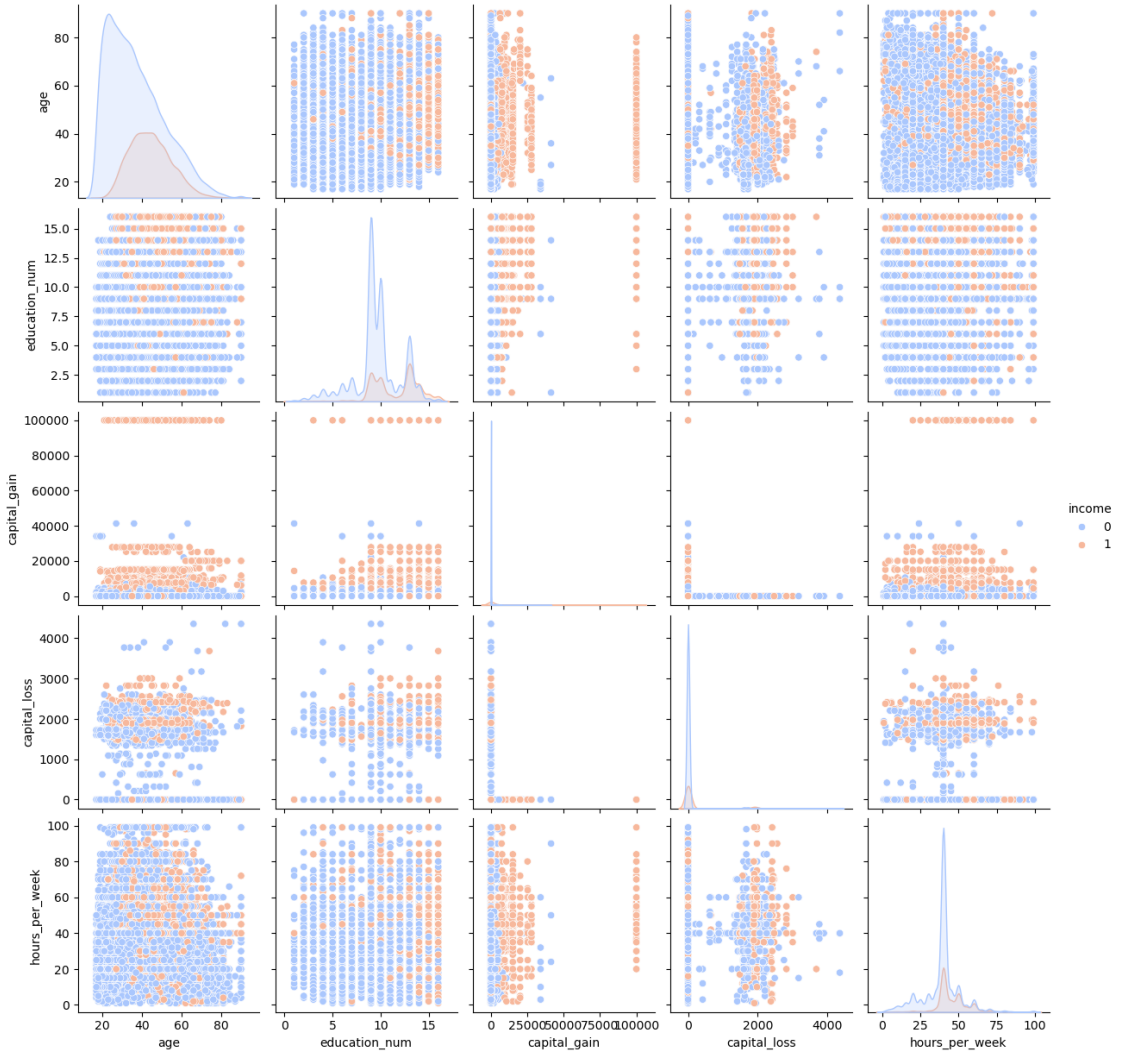

Figure 16: Pairplot Analysis of Numerical Variables by Income

The above figure presents a pairplot that visualizes the relationships among key numerical variables in the dataset, segmented by income levels. The income classes are denoted by color coding, where **blue** represents individuals earning $\leq 50K$ and **orange** represents individuals earning $> 50K$. This pairplot enables a comprehensive view of the distributions, correlations, and interdependencies of numerical features, including `age`, `education_num`, `capital_gain`, `capital_loss`, and `hours_per_week`.

The diagonal plots display kernel density estimates (KDEs) of each variable, providing insight into the univariate distributions of each feature across the income groups. For in-

stance, the `age` distribution shows a right-skewed pattern, with a higher concentration of individuals between 20 and 50 years, predominantly in the $\leq 50K$ category. However, the density curve for higher income individuals extends further into the older age ranges, suggesting that older individuals may be more likely to achieve higher income levels.

Examining the scatter plots, several notable relationships emerge. The relationship between `education_num` (years of education) and income shows a distinct separation, where individuals with higher educational levels (e.g., `education_num` above 12) are more likely to fall within the $> 50K$ income group. This relationship highlights the role of educational attainment as a strong predictor of income level.

The variables `capital_gain` and `capital_loss` exhibit highly skewed distributions, with most values clustered around zero, particularly in the $\leq 50K$ income group. A few outliers in both `capital_gain` and `capital_loss` stand out in the higher income category, indicating that substantial capital gains or losses are associated with higher income individuals. This aligns with socioeconomic patterns, where investments and financial gains are typically correlated with higher income.

For `hours_per_week`, the plot shows that individuals in the higher income bracket tend to work slightly more hours on average than those in the lower income bracket, though both groups primarily cluster around a 40-hour workweek. The range of hours is broader for the $> 50K$ group, suggesting that some high earners may work significantly more hours, likely in demanding or high-paying professions.

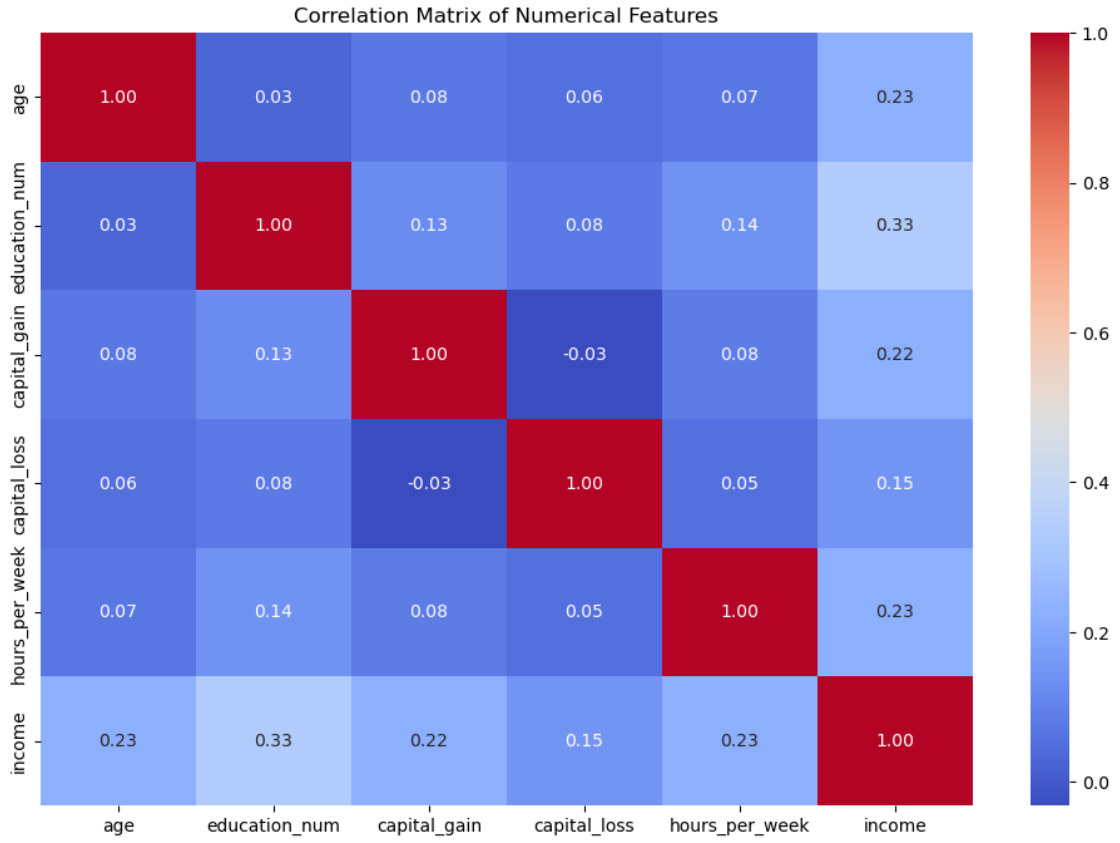

Figure 17: Correlation Matrix of Numerical Features

Correlation Matrix of Numerical Features

The above figure presents a correlation matrix for the numerical features in the dataset, offering insights into the linear relationships between variables. Each cell in the matrix quantifies the correlation between two variables, with values ranging from -1 (perfect negative correlation) to +1 (perfect positive correlation). The color gradient, from blue to red, visually represents the strength and direction of each correlation, where darker shades of red signify strong positive correlations, and darker shades of blue indicate weak or negative correlations.

Examining the correlation values, we observe that `education_num` (years of education) has the strongest positive correlation with `income` (0.33). This suggests that higher education levels are associated with a greater likelihood of higher income. This finding aligns with general socioeconomic patterns where educational attainment is often a predictor of income, reflecting the role of education in accessing higher-paying occupations.

`Capital_gain` also exhibits a notable positive correlation with `income` (0.22), though

weaker than that of `education_num`. This relationship implies that individuals with substantial capital gains are more likely to be in the higher income category, consistent with the notion that income-generating activities, such as investments, contribute to higher income levels. Additionally, `age` and `hours_per_week` show moderate positive correlations with `income` (0.23 each), suggesting that both increased working hours and older age are modestly associated with higher income. The positive correlation with `age` could reflect income growth with career progression, while the correlation with `hours_per_week` may indicate that higher earners are more likely to work extended hours.

Interestingly, `capital_loss` has a relatively low correlation with all other variables, including `income` (0.15), indicating that it has limited predictive value for income in this dataset. This lower correlation could suggest that while some individuals in the higher income bracket report capital losses, this attribute alone does not strongly determine income levels.

The correlations among the predictor variables reveal minimal multicollinearity, as no pair of features exhibits a high correlation with one another. For instance, `age` and `education_num` have a weak correlation (0.03), implying that they are largely independent of each other. The low correlations among independent variables are advantageous for modeling, as they reduce redundancy and improve the interpretability of models by ensuring that each variable contributes distinct information.

Standardization and Data Splicing

```python
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Select only numerical columns ('income' is the target variable)
numerical_features = ['age', 'fnlwgt', 'education_num', 'capital_gain', 'capital_loss', 'hours_per_week']
X = data[numerical_features]
y = data['income']

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Scale the numerical features for KNN
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Figure 18: Standardization

Data Preprocessing for K-Nearest Neighbors (KNN) Classification

The above figure presents a Python code snippet for data preprocessing, specifically tailored for implementing a K-Nearest Neighbors (KNN) classification model. The code follows a structured approach to prepare numerical features, split the dataset into training and testing subsets, and scale the data, which is essential for distance-based algorithms like KNN.

Initially, the code imports two fundamental libraries from `scikit-learn`: `train_test_split` from `model_selection` and `StandardScaler` from `preprocessing`. The `train_test_split` function is critical for dividing the dataset into training and testing sets, enabling model evaluation on unseen data. The `StandardScaler` is used to standardize features by removing the mean and scaling to unit variance, which is particularly important for algorithms that rely on distance metrics.

The first section of the code defines a list of `numerical_features`, which includes `age`, `fnlwgt` (final weight), `education_num` (years of education), `capital_gain`, `capital_loss`, and `hours_per_week`. These variables are selected from the dataset to serve as predictors for the KNN model. The target variable, `income`, is assigned to `y` and contains binary labels indicating whether an individual's income is above or below 50K. Selecting only numerical features is essential in KNN because the algorithm computes distances between data points, which requires continuous, scaled inputs for meaningful comparisons.

The second section of the code splits the dataset into training and testing sets using an 80-20 ratio, as specified by `test_size=0.3`. The `random_state=42` parameter ensures reproducibility by fixing the random seed, meaning that the split will be consistent across different runs. The split yields four subsets: `X_train`, `X_test`, `y_train`, and `y_test`, where `X` represents feature matrices and `y` represents target labels for each subset.

In the final section, the code applies feature scaling to the training and testing data using `StandardScaler`. The scaler is first fitted to the training data and then the test data ensuring that both sets are standardized based on the training data's distribution. **Standardization is crucial for KNN, as the algorithm's performance can be significantly impacted by differences in feature scales, with unscaled features potentially dominating the distance calculations.**

Part 2: K-Nearest Neighbors Classifier

```
Confusion Matrix:
 [[10198   968]
 [ 1825  1662]]
Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.91      0.88     11166
           1       0.63      0.48      0.54      3487

    accuracy                           0.81     14653
   macro avg       0.74      0.69      0.71     14653
weighted avg       0.80      0.81      0.80     14653
```

Figure 19: Confusion Matrix and Classification Report for Income Prediction Preliminary Model

Confusion Matrix and Classification Report for Income Prediction Preliminary Model

The above figure presents the confusion matrix and classification report for a K-Nearest Neighbors (KNN) classifier applied to predict income categories. The target variable, `income`, is binary, with **0** representing individuals earning $\leq 50K$ and **1** representing those earning $> 50K$. The performance metrics include **precision**, **recall**, **f1-score**, and **support**, providing a comprehensive assessment of the model's classification efficacy on both income classes.

The confusion matrix, displayed in the top section, summarizes the classifier's performance by showing the number of true positive, true negative, false positive, and false negative predictions. Specifically, the matrix indicates that:

- The model correctly predicted the $\leq 50K$ income class (label **0**) 10,198 times, with 968 instances misclassified as $> 50K$.
- For the $> 50K$ income class (label **1**), the model correctly classified 1,662 instances, while 1,825 cases were incorrectly predicted as $\leq 50K$.

This distribution indicates a higher accuracy for the $\leq 50K$ class, which is further explored in the classification report metrics.

The classification report provides a detailed breakdown of key performance metrics:

- **Precision**: Precision measures the accuracy of positive predictions. For class **0** ($\leq 50K$), the precision is 0.85, indicating that 85% of instances predicted as $\leq 50K$ were correct. For class **1** ($> 50K$), precision is lower at 0.63, reflecting more false positives in this class.

- **Recall**: Recall assesses the model's ability to correctly identify all positive instances. The recall for class **0** is high at 0.91, indicating that 91% of actual $\leq 50K$ cases were correctly identified. Conversely, class **1** has a lower recall of 0.48, suggesting that the model missed a significant proportion of $> 50K$ instances.

- **F1-Score**: The F1-score is the harmonic mean of precision and recall, providing a balanced measure of accuracy, particularly in cases of class imbalance. The F1-score for class **0** is 0.88, while it is 0.54 for class **1**, indicating that the model performs substantially better on the $\leq 50K$ class.

- **Support**: Support denotes the number of actual instances for each class in the test data, with 11,166 samples for class **0** and 3,487 for class **1**.

The **overall accuracy** of the model is 0.81, meaning that 81% of predictions across both classes were correct. The **macro average** F1-score, which treats both classes equally, is 0.71, while the **weighted average** F1-score, which accounts for class imbalance, is 0.80.

In summary, the model demonstrates high performance for the majority class ($\leq 50K$) but exhibits lower precision and recall for the minority class ($> 50K$). This imbalance highlights the model's limitation in identifying higher-income individuals accurately, which may require further adjustments, such as resampling techniques or hyperparameter tuning, to improve performance on the minority class.

ROC Curve for Initial K-Nearest Neighbors Preliminary Model (k=5)

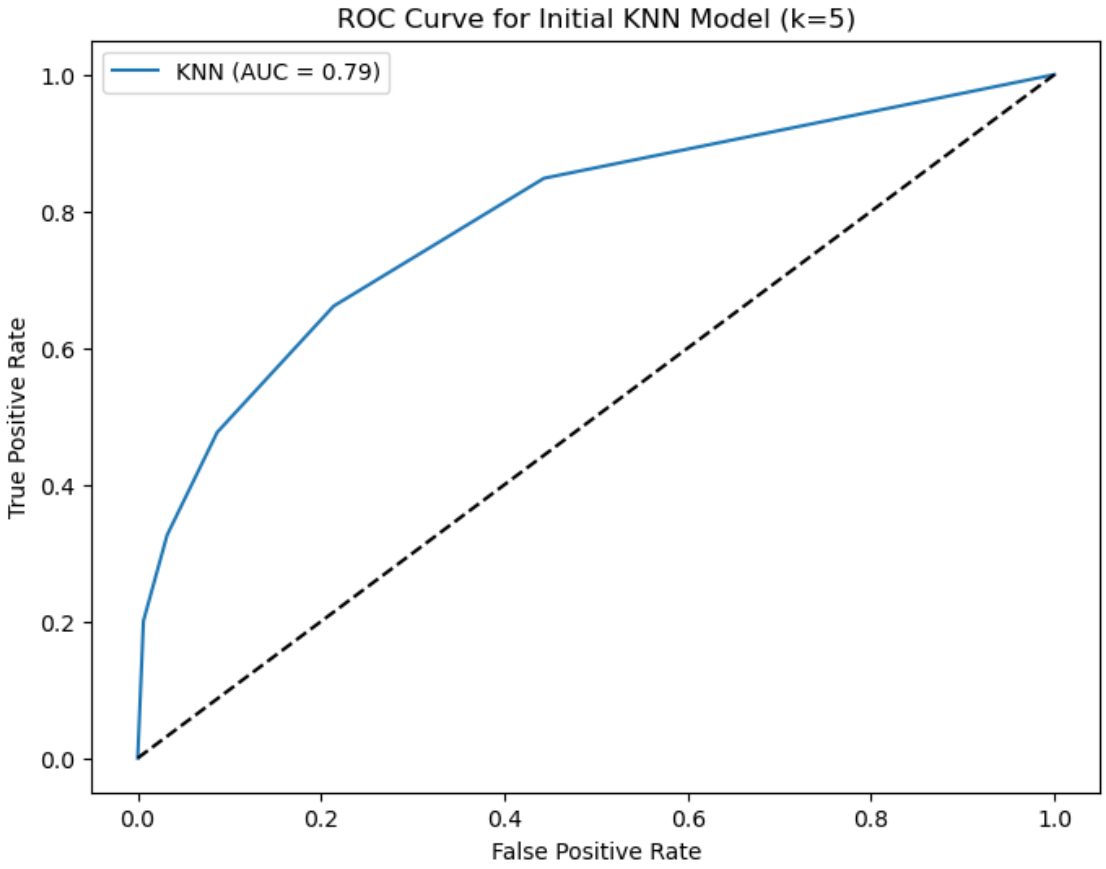

Figure 20: ROC Curve for Initial K-Nearest Neighbors Preliminary Model (k=5)

The above figure displays the Receiver Operating Characteristic (ROC) curve for an initial K-Nearest Neighbors (KNN) classifier with $k = 5$. This ROC curve is a graphical representation of the classifier's diagnostic ability by plotting the **True Positive Rate** (TPR, or Sensitivity) against the **False Positive Rate** (FPR) at various threshold settings. The ROC curve provides insights into the model's performance across all possible classification thresholds, facilitating the evaluation of the model's discriminatory power between classes.

The ROC curve for this KNN model demonstrates a relatively strong classification performance, as evidenced by the model's AUC (Area Under the Curve) value of 0.79. The AUC is a summary metric that quantifies the overall ability of the classifier to distinguish between the positive and negative classes, with values closer to 1 indicating excellent discrimination

and values near 0.5 suggesting random guessing. An AUC of 0.79 signifies that the model possesses a reasonably high level of predictive accuracy, as it can correctly differentiate between the income classes $\leq 50K$ and $> 50K$ about 79% of the time.

The curve itself deviates substantially from the *diagonal line of no-discrimination* (dashed line), which represents an AUC of 0.5 and reflects the performance of a random classifier. The distance of the ROC curve above this line indicates the classifier's effectiveness, particularly in minimizing false positives while maximizing true positives. The initial portion of the curve shows a steep rise in the TPR with only a modest increase in the FPR, suggesting that the model is effective in capturing positive instances at low levels of false positive errors.

However, as the FPR increases, the curve begins to flatten, indicating diminishing returns in the TPR as the classifier relaxes its threshold and includes more false positives. This behavior is typical in binary classification and highlights the trade-off between sensitivity and specificity, where the choice of an optimal threshold depends on the specific application requirements and acceptable levels of false positives versus false negatives.

The ROC curve provides valuable guidance in threshold selection. For applications prioritizing high sensitivity, the classifier can operate in regions of the curve with a high TPR, albeit with some tolerance for increased FPR. Conversely, for settings where precision is paramount, thresholds with lower FPRs can be selected, sacrificing some sensitivity for greater specificity.

In summary, this ROC curve and its associated AUC value of 0.79 indicate that the KNN model with $k = 5$ offers promising performance in distinguishing income classes. Nevertheless, further tuning, such as optimizing the k value or exploring additional features, could potentially enhance this model's performance.

Hyperparameter Tuning for K-Nearest Neighbors (KNN): Selection of Optimal K Value

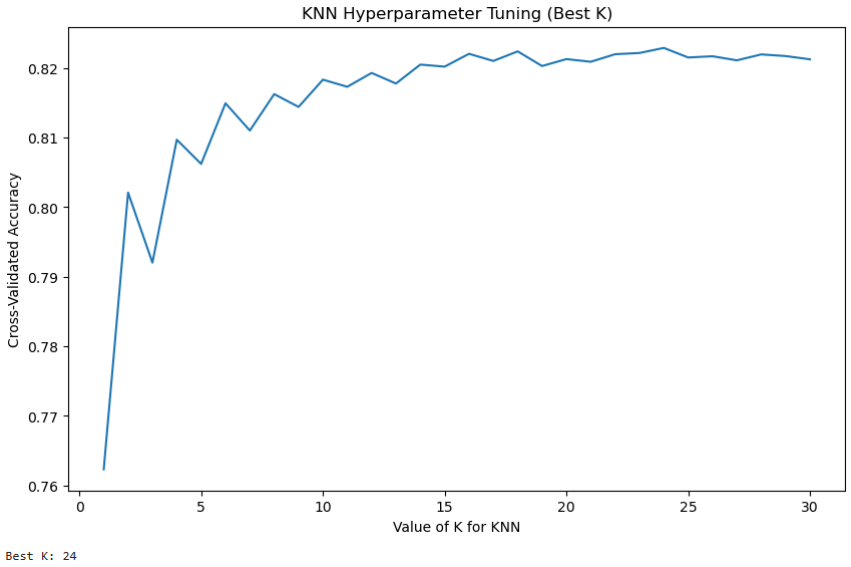

Figure 21: Hyperparameter Tuning

The above figure illustrates the process of hyperparameter tuning for a K-Nearest Neighbors (KNN) classifier by evaluating the model's cross-validated accuracy across various values of K (the number of nearest neighbors considered in classification). The x-axis represents different values of K, ranging from 1 to 30, while the y-axis denotes the cross-validated accuracy achieved for each corresponding K value. The goal of this tuning is to identify the optimal K that maximizes the model's predictive performance.

The curve demonstrates that cross-validated accuracy initially increases sharply as K rises from 1, indicating that the model's stability and generalization improve with the addition of more neighbors. This effect is particularly noticeable in the lower range of K, where small values such as 1 or 2 may lead to overfitting, causing the model to be highly sensitive to individual data points. The sharp initial increase in accuracy highlights the benefit of aggregating information from multiple neighbors, which mitigates the noise sensitivity of very low K values.

As K continues to increase, the accuracy gradually plateaus, fluctuating around a stable range between 0.81 and 0.82. This plateau suggests that further increases in K yield dimin-

ishing returns in terms of accuracy, as the model reaches an optimal balance between bias and variance. After approximately $K = 10$, the model achieves relatively stable accuracy, indicating that this range provides a reliable classification without substantial overfitting or underfitting.

The best K, indicated in the figure as $K = 24$, achieves the maximum cross-validated accuracy within this tuning range. This optimal K balances the trade-off between model complexity and generalization. A higher K smoothens the decision boundaries, thus improving generalization by reducing the impact of noisy instances, especially in datasets with potential outliers or minor variations in feature values. However, excessively high K values can lead to underfitting, as the decision boundary becomes overly smooth, thus potentially misclassifying data points from minority classes or subtle patterns.

Selecting an optimal K based on cross-validation provides a robust measure of the model's ability to generalize to unseen data, as this process evaluates performance across multiple subsets of the training data. This approach helps ensure that the selected K is not overly tuned to specific samples in the dataset, enhancing the model's reliability when deployed on new data.

In summary, the optimal $K = 24$ balances accuracy and generalization, making it the preferred choice for the KNN model in this classification task. This analysis underscores the importance of hyperparameter tuning in improving model performance and optimizing predictive power.

Confusion Matrix and Classification Report with Optimized K for K-Nearest Neighbors

```
Confusion Matrix with Best K:
 [[10602   564]
 [ 2052  1435]]
Classification Report with Best K:
              precision    recall  f1-score   support

           0       0.84      0.95      0.89     11166
           1       0.72      0.41      0.52      3487

    accuracy                           0.82     14653
   macro avg       0.78      0.68      0.71     14653
weighted avg       0.81      0.82      0.80     14653
```

Figure 22: Confusion Matrix and Classification Report with Optimized K for K-Nearest Neighbors

The above figure presents the confusion matrix and classification report for a K-Nearest Neighbors (KNN) classifier using the optimal K value identified through cross-validation. The optimal K, selected to maximize cross-validated accuracy, aims to improve model performance by balancing precision, recall, and overall accuracy.

The confusion matrix provides a summary of the model's performance across the two income classes. Specifically, it shows:

- True negatives (**0** predicted as **0**): 10,602 instances, where the model correctly classified individuals earning $\leq 50K$.
- False positives (**1** predicted as **0**): 564 cases, where individuals earning $> 50K$ were misclassified as $\leq 50K$.
- False negatives (**0** predicted as **1**): 2,052 cases, where the model incorrectly predicted $\leq 50K$ income for individuals earning $> 50K$.

- True positives (**1** predicted as **1**): 1,435 instances, accurately identifying individuals earning $> 50K$.

The classification report provides a comprehensive evaluation of the model's performance metrics for each class:

- **Precision**: Precision for class **0** ($\leq 50K$) is 0.84, indicating that 84% of instances predicted as $\leq 50K$ are correct. For class **1** ($> 50K$), precision improves to 0.72, reflecting a reduced false positive rate compared to the initial KNN model.

- **Recall**: Recall for class **0** remains high at 0.95, capturing 95% of true $\leq 50K$ cases. However, recall for class **1** is relatively low at 0.41, indicating that a significant portion of high-income individuals are missed by the model, likely due to the class imbalance.

- **F1-Score**: The F1-score for class **0** is 0.89, while for class **1** it is 0.52. The discrepancy in F1-scores suggests the model performs well in identifying the majority class ($\leq 50K$), but faces challenges in effectively classifying the minority class ($> 50K$).

- **Support**: Support reflects the number of actual instances for each class in the test set, with 11,166 samples for class **0** and 3,487 for class **1**.

The overall **accuracy** of the model is 0.82, which indicates that the optimized KNN model correctly classifies 82% of instances in the test set. The **macro average** and **weighted average** F1-scores, at 0.71 and 0.80 respectively, further highlight the model's limitations in handling class imbalance, as the macro average treats both classes equally while the weighted average accounts for class prevalence.

In summary, tuning the K parameter has improved the model's performance metrics, particularly precision for the $> 50K$ class, achieving an overall accuracy of 82%. However, the disparity in recall between income classes suggests that further adjustments, such as resampling techniques or alternative modeling approaches, may be necessary to enhance the model's sensitivity to high-income individuals.

ROC Curve for Optimized K-Nearest Neighbors Optimized Model (k=24)

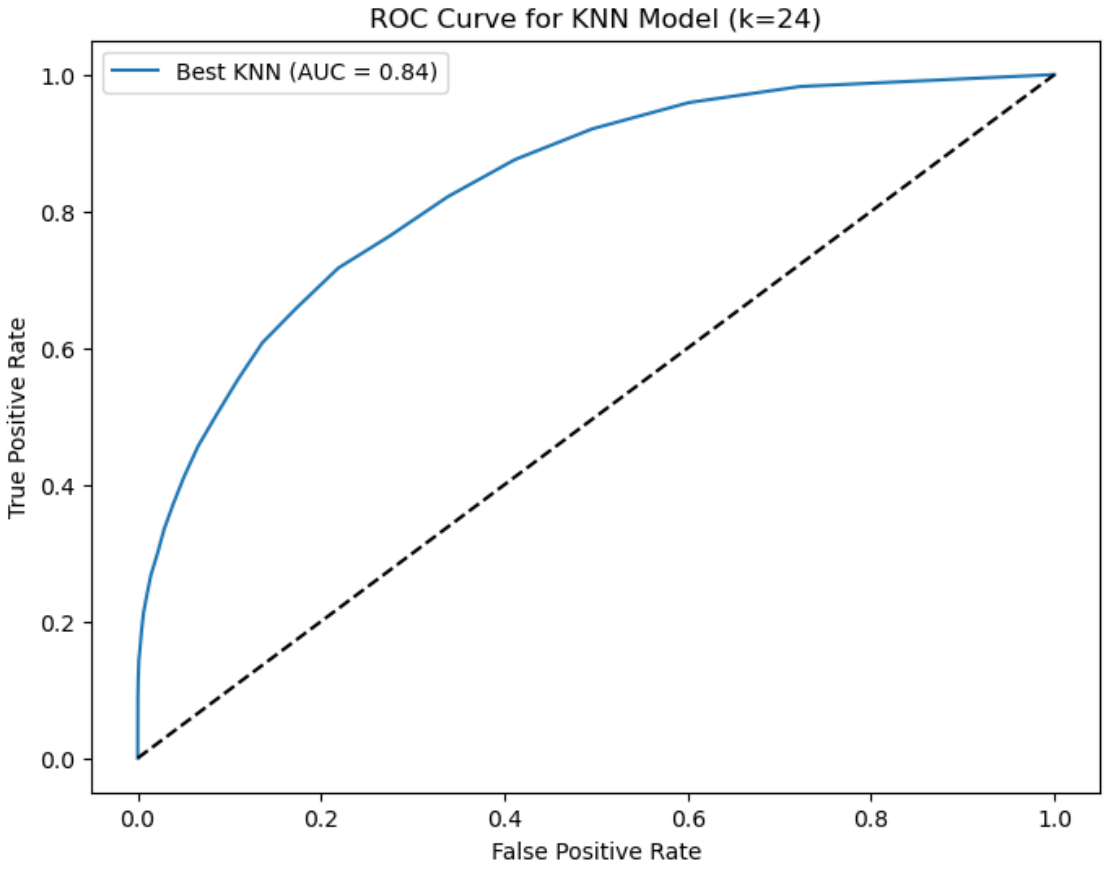

Figure 23: ROC Curve for Optimized K-Nearest Neighbors Model (k=24)

The above figure illustrates the Receiver Operating Characteristic (ROC) curve for an optimized K-Nearest Neighbors (KNN) classifier with $k = 24$, selected based on cross-validation to maximize classification performance. The ROC curve is a plot of the **True Positive Rate** (TPR, or sensitivity) against the **False Positive Rate** (FPR) across various classification thresholds, allowing for an evaluation of the model's ability to distinguish between the two income classes: $\leq 50K$ and $> 50K$.

The curve's overall shape and the Area Under the Curve (AUC) value provide insights into the model's discriminatory power. Here, the AUC for the optimized KNN model is 0.84, indicating a strong ability to separate income classes. An AUC value of 0.84 implies that there is an 84% chance that the model will correctly differentiate between a randomly

selected high-income individual and a randomly selected low-income individual. This is an improvement over the initial KNN model, which had an AUC of 0.79, suggesting that tuning the k parameter enhances the model's predictive performance.

The ROC curve departs significantly from the *diagonal line of no-discrimination* (dashed line), which represents an AUC of 0.5, or the performance of a random classifier. The distance of the curve above this diagonal indicates the model's effectiveness in minimizing false positives while maximizing true positives. At lower FPR values, the curve rises steeply, showing that the model is capable of achieving a high TPR with minimal false positives. This steep rise is particularly advantageous in applications where reducing false positives is crucial.

However, as the FPR increases, the ROC curve begins to flatten, indicating that further increases in TPR come at the cost of higher FPR. This flattening suggests that the model's benefit in correctly identifying positive instances diminishes as it relaxes the classification threshold. The shape of the ROC curve reflects the trade-off inherent in classification threshold selection, where more lenient thresholds may increase sensitivity (TPR) but also increase false positives.

The AUC value of 0.84 confirms that the model with $k = 24$ is an effective classifier, balancing sensitivity and specificity to improve classification accuracy. The selection of $k = 24$ represents an optimal trade-off between overfitting (at low k values) and underfitting (at very high k values). This optimized KNN model demonstrates enhanced performance over the initial model, making it more suitable for distinguishing between income groups in this dataset.

In summary, the ROC curve and AUC for the optimized KNN model highlight the effectiveness of hyperparameter tuning in improving classification accuracy, illustrating a well-calibrated model with strong discriminatory capabilities for income prediction.

Train vs Test Set Accuracy for Different K Values in K-Nearest Neighbors

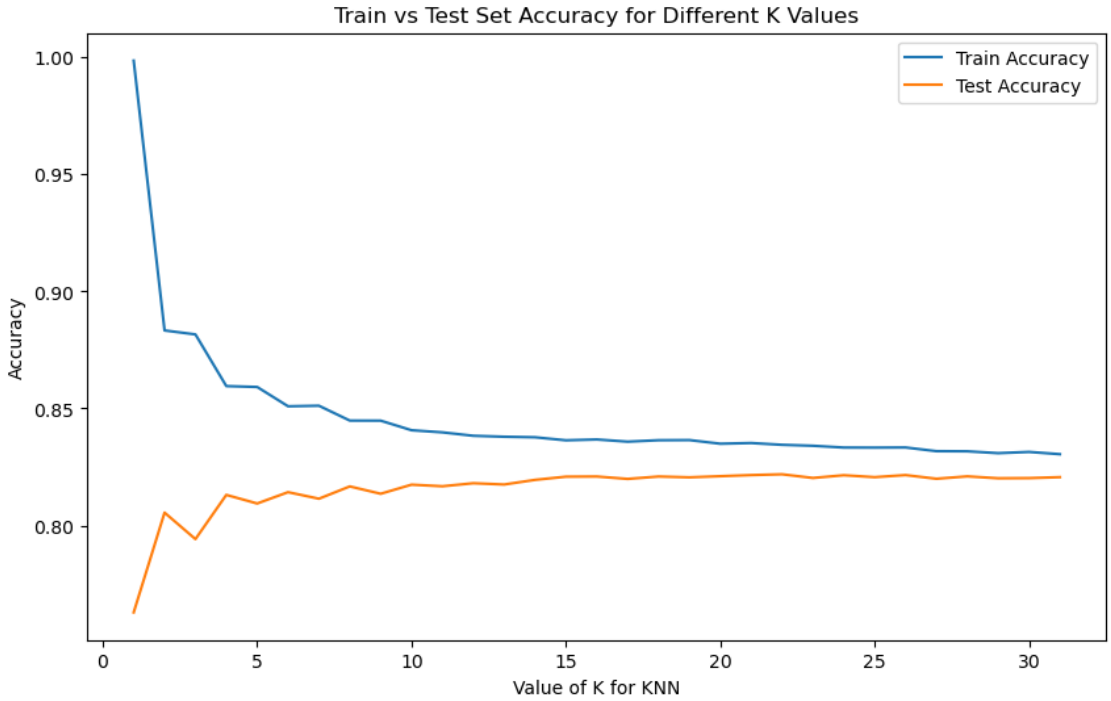

Figure 24: Train vs Test Set Accuracy

The above figure illustrates the relationship between the value of K (the number of nearest neighbors) and the classification accuracy on both the training and test datasets for a K-Nearest Neighbors (KNN) classifier. The x-axis represents the different values of K, ranging from 1 to 30, while the y-axis shows the classification accuracy. Two curves are depicted: **Train Accuracy** (in blue) and **Test Accuracy** (in orange), allowing for a comparative analysis of model performance as K increases.

In this figure, the **train accuracy** starts very high at $K = 1$, where the model achieves near-perfect accuracy on the training set. This phenomenon is a result of the KNN model's tendency to overfit when K is low, as it essentially memorizes the training data by focusing on the closest neighbors, which can lead to a strong alignment with training instances but poor generalization to unseen data.

As K increases, **train accuracy** gradually declines, stabilizing around a lower accuracy level. This decline reflects a shift in the model from memorizing the training data to creating

smoother, more generalized decision boundaries. The stabilization of train accuracy at higher K values indicates that the model's decision boundary becomes less complex, reducing the likelihood of overfitting by incorporating information from a larger neighborhood.

Conversely, **test accuracy** exhibits an upward trend as K increases from 1, indicating that increasing K improves the model's generalization ability and reduces overfitting. However, test accuracy reaches a peak around the middle of the range (near $K = 24$) and subsequently plateaus or slightly declines. This peak signifies the optimal balance between bias and variance: increasing K beyond this point introduces more bias, causing the model to underfit the data as it smooths over finer distinctions between classes, which slightly reduces test accuracy.

The gap between **train accuracy** and **test accuracy** narrows as K increases, demonstrating the reduction of overfitting. At lower K values, the significant disparity between train and test accuracy highlights the model's overfitting tendency, where it performs well on training data but fails to generalize to new data. As K approaches the optimal value, the alignment between train and test accuracy improves, achieving a more robust and generalizable model.

In conclusion, this analysis reveals that $K = 24$ yields the highest test accuracy, making it the most suitable choice for this KNN model. The plot effectively illustrates the trade-off between bias and variance across different K values, showing how tuning K can enhance model performance by improving generalization while minimizing overfitting.

Feature Importance for K-Nearest Neighbors Model Using Permutation Importance

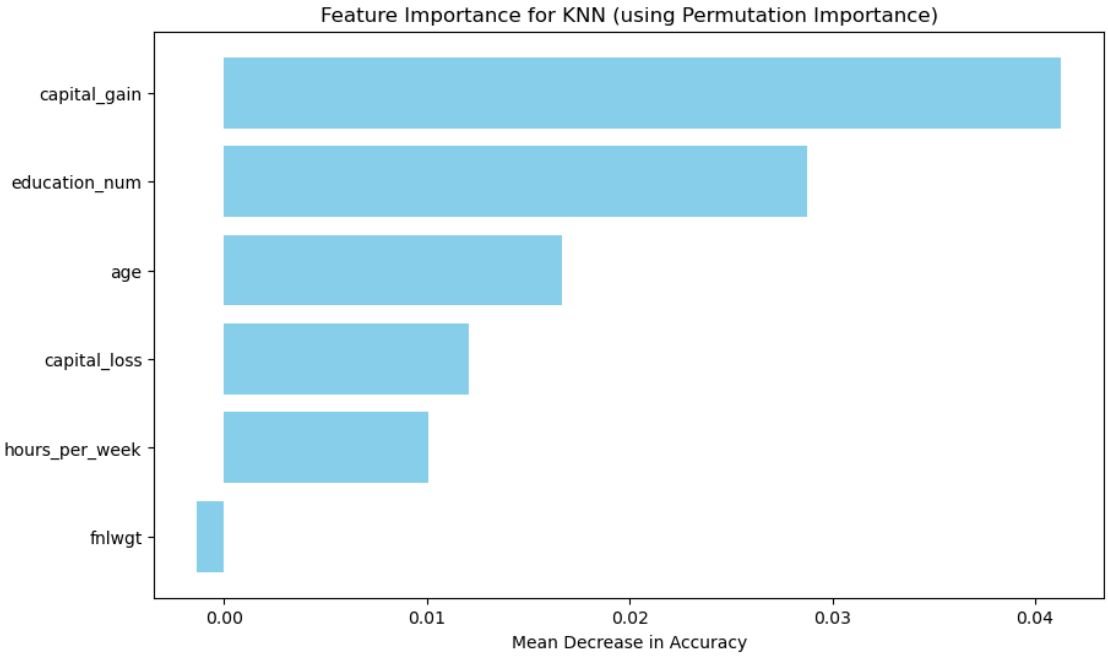

Figure 25: Feature Importance for K-Nearest Neighbors Model Using Permutation Importance

The above figure displays the feature importance of the variables used in a K-Nearest Neighbors (KNN) classification model to predict income classes ($\leq 50K$ and $> 50K$). Feature importance in this context has been determined using the **Permutation Importance** technique, which evaluates the significance of each feature by measuring the mean decrease in model accuracy when the values of that feature are randomly shuffled. A higher mean decrease in accuracy indicates that the feature plays a crucial role in the model's predictive performance, as its alteration significantly disrupts the classification accuracy.

In this figure, the x-axis represents the **Mean Decrease in Accuracy**, a metric that quantifies each feature's impact on model performance. The y-axis lists the individual features, ordered by their relative importance, with more influential features at the top. The permutation importance reveals substantial variability in the contribution of different features to the model.

- **Capital Gain**: This feature exhibits the highest mean decrease in accuracy, making it

the most influential variable in distinguishing income levels in this dataset. The strong predictive power of *capital gain* suggests that individuals with higher income are more likely to have substantial capital gains, aligning with expected financial patterns where higher earnings are associated with significant investment returns.

- **Education Number (education_num)**: Ranked second in importance, *education_num*, which represents the number of years of education, also significantly contributes to the model. This result aligns with socioeconomic theories suggesting a positive correlation between educational attainment and income potential, indicating that higher education levels are predictive of higher income categories.

- **Age**: The *age* feature demonstrates moderate importance, reflecting its relevance in income prediction. This importance may stem from income growth typically associated with experience and career progression, which often correlates with age.

- **Capital Loss**: Although less influential than capital gain, *capital loss* remains a relevant feature, likely capturing financial characteristics that may distinguish high-income individuals who experience periodic financial losses from investments.

- **Hours Per Week (hours_per_week)**: The average number of hours worked per week shows a modest contribution to the model's accuracy, suggesting that work hours provide some insight into income level, albeit to a lesser degree than other financial and educational features.

- **Final Weight (fnlwgt)**: This feature is the least important among the selected variables, as indicated by the minimal mean decrease in accuracy when shuffled. This low importance suggests that *fnlwgt*, which represents sample weight in census data, holds limited predictive power in this specific income classification task.

In summary, the permutation importance analysis reveals that *capital gain* and *education_num* are the most impactful features in the KNN model, highlighting their substantial roles in income prediction. The analysis underscores the influence of financial and educational attributes on income classification, while demographic variables like age and work hours provide secondary contributions.

Comparison of k-NN Model Results for Initial k and Optimized k Values

Table 1: Comparison of k-NN Model Results for Initial k (k=5) and Optimized k (k=24)

Metric	**Initial k (k=5)**	**Optimized k (k=24)**
Overall Accuracy	0.81	0.82
Precision (Class 0)	0.85	0.84
Precision (Class 1)	0.63	0.72
Recall (Class 0)	0.91	0.95
Recall (Class 1)	0.48	0.41
F1-Score (Class 0)	0.88	0.89
F1-Score (Class 1)	0.54	0.52
Macro Avg F1-Score	0.71	0.71
Weighted Avg F1-Score	0.80	0.80

The above figure presents a detailed comparison between the initial and optimized k-NN models, with $k = 5$ and $k = 24$, respectively, for income classification. This comparative table provides key performance metrics, including **Overall Accuracy**, **Precision**, **Recall**, and **F1-Score** for both income classes (*Class 0* and *Class 1*), as well as macro and weighted average F1-scores.

- **Overall Accuracy**: The optimized model with $k = 24$ achieves a slightly higher overall accuracy of 0.82 compared to the initial model's 0.81. This improvement, while marginal, indicates that the optimized k provides a more balanced and effective classification by reducing the model's sensitivity to outliers or noise that can affect the accuracy at lower k values.

- **Precision (Class 0 and Class 1)**: For *Class 0* (income $< 50K$), precision decreases slightly in the optimized model, moving from 0.85 to 0.84. However, for *Class 1* (income $> 50K$), precision improves significantly from 0.63 to 0.72. This increase in precision for Class 1 demonstrates that the optimized model has become better at correctly identifying higher-income individuals while reducing the number of false positives for this class.

- **Recall (Class 0 and Class 1)**: The optimized model shows an increase in recall

for *Class 0*, from 0.91 to 0.95, indicating a more comprehensive identification of lower-income individuals. However, the recall for *Class 1* slightly decreases from 0.48 to 0.41, suggesting a trade-off where the model now captures fewer high-income cases. This outcome could be attributed to the model's adjustment towards higher precision for Class 1, leading to a stricter classification threshold that slightly sacrifices recall.

- **F1-Score (Class 0 and Class 1)**: The F1-score for *Class 0* increases marginally from 0.88 to 0.89, while for *Class 1* it decreases from 0.54 to 0.52. The F1-score, which balances precision and recall, reflects the model's effectiveness in managing the trade-off between these metrics. For Class 1, the F1-score reduction suggests that while precision has improved, the decrease in recall has balanced out some of the gains.

- **Macro and Weighted Average F1-Scores**: Both the macro and weighted F1-scores remain unchanged at 0.71 and 0.80, respectively, indicating that the overall model performance across both classes is relatively stable between the initial and optimized k values.

In summary, tuning k from 5 to 24 has yielded an optimized model with a slight increase in overall accuracy and substantial improvements in precision for identifying higher-income individuals. The model's adjustments illustrate the typical trade-offs in classification performance metrics, balancing precision and recall to optimize predictive accuracy.

Summary and Interpretation

This report presents a comprehensive analysis of income classification among US citizens utilizing census data. The study, conducted as part of ALY 6020 at Northeastern University, employs a two-pronged approach: **Exploratory Data Analysis (EDA)** and the application of **K-Nearest Neighbors (KNN)** classification algorithm.

The investigation commences with a thorough EDA, encompassing *data preprocessing* and *cleansing* procedures. The dataset, comprising 48,841 entries across 15 columns, includes a diverse array of demographic and occupational attributes such as age, workclass, education, marital status, occupation, race, sex, and income.

The study leverages various Python libraries, including pandas for data manipulation, `numpy` for numerical computations, and `seaborn` and `matplotlib` for data visualization. These tools facilitate a comprehensive examination of the dataset's structure, composition, and statistical properties

The report meticulously details the dataset's characteristics, elucidating the nature of each variable and its potential significance in income prediction using the **K-Nearest Neighbors (KNN)** classification algorithm revealing **Capital gains** and **Education level** to be the two variables of utmost importance.

A *Cross-Validation* based optimization of the hyper-parameter resulted in 24 being the ideal number of neighbors, and the resulting model achieved high accuracy and precision.

References

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE transactions on information theory*, *13*(1), 21–27.

Dudani, S. A. (1976). The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*(4), 325–327.

Fix, E., & Hodges Jr, J. L. (1951). Discriminatory analysis-nonparametric discrimination: consistency properties. *California Univ Berkeley*.

Ougiaroglou, S., & Evangelidis, G. (2015). Dealing with noisy data in the context of k-nn classification. *Proceedings of the 19th Panhellenic Conference on Informatics*, 414–419.

Weinberger, K. Q., & Saul, L. K. (2009). Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, *10*(Feb), 207–244.

Zhang, L., Ding, R., & Markov, S. (2021). Forecasting earnings using k-nearest neighbors. *SSRN Electronic Journal*.

Zhang, M.-L., & Zhou, Z.-H. (2007). Ml-knn: A lazy learning approach to multi-label learning. *Pattern recognition*, *40*(7), 2038–2048.

(Cover & Hart, 1967) (Dudani, 1976) (Fix & Hodges Jr, 1951) (Ougiaroglou & Evangelidis, 2015) (Weinberger & Saul, 2009) (M.-L. Zhang & Zhou, 2007) (L. Zhang, Ding, & Markov, 2021)