Northeastern University

Module 4 Capstone Project Draft Report

Priyanka Singh, Riya Parimal Dalal and V S N Sai Krishna Mohan Kocherlakota

ALY6140

Prof. Roy Wada

05/12/2023

Introduction

We have used a dataset "The Current Population Survey (CPS)" ***(ICPSR, 2019)*** is a useful dataset for socioeconomic research and various analysis. The chosen dataset consists of 5000 rows and 13 variables. The variables consist of gender, income, age, employment status, weight etc. and have several entries in it. This report aims to provide an overview of the data set chosen which will highlight major points like dimensions, questions to investigate, summary statistics, graphical visualizations, and analysis for the same.

Dimensions of Dataset

The selected dataset comprises a substantial sample size, meeting the requirement of 5,000 records and includes 13 usable features. The dimensions of the dataset play a crucial role in ensuring the robustness of the analysis that will be done. Sharing a SS of the same from the code run in Jupyter Notebook. ***(Geeksfor Geeks, 2023)***

Output –

```
Modules
xlsx imported
```

Code –

```
print ("Initial Data Exploration:")
print ("Dimensions of the dataset:", df.shape)
print ("First Few rows of the dataset:n",df.head())
```

Output –

```
Initial Data Exploration:
Dimensions of the dataset: (5000, 13)
First Few rows of the dataset:
                 HEFAMINC   HWHHWGT       HRINTSTA       PREXPLF       PESEX
0  (04) 10,000 to 12,499  1836.375  (1) Interview  (1) Employed  (2) Female
1  (10) 35,000 to 39,999  1542.311  (1) Interview  (1) Employed    (1) Male

              PENATVTY  PRTAGE
0  (057) United States    21.0
1  (057) United States    49.0

                                              PRDTOCC1   PWFMWGT   PWLGWGT
0  (22) Transportation and material moving occupa...  1836.375  2626.141
1  (20) Installation, maintenance, and repair occ...  1542.311  2205.888

    PWORWGT   PWSSWGT  PWVETWGT
0  7019.797  1836.375  1750.363
   1  6132.680  1542.311  1487.825
```

Code–

```
print ("Overview of columns and data types: n",df.info())  # Overview of columns and data types
```

Output –

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Columns: 444 entries, Unnamed: 0 to PWNRWGT
dtypes: float64(57), int64(17), object(370)
memory usage: 16.9+ MB
```

Code –

```
print(df.describe())  # Summary statistics for numerical columns
```

Output –

```
           HWHHWGT       PRTAGE      PWFMWGT       PWLGWGT       PWORWGT
count  5000.000000  4179.000000  5000.000000   5000.000000   5000.000000
```

```
mean   2388.882525    44.086863  2423.032584   2109.146015   2264.149462
std    1494.271754    23.620775  1537.350366   2391.514162   5023.167621
```

We're working with a dataset "Population.xlsx" that contains various columns related to demog raphic and possibly weight-related information. The dataset has 5000 entries across 13 columns .

Here's an overview of the data:

- **Dimensions:** The dataset contains 5000 rows and 13 columns.
- **Data Types:** The columns have a mix of data types - seven columns are numeric (float64) while the remaining six columns are categorical (object).
- **Missing Values:** Some columns have missing values:
- 'HEFAMINC', 'PREXPLF', 'PESEX', 'PENATVTY', 'PRTAGE', and 'PRDTOCC1' have varying n umbers of non-null entries.
- **Summary Statistics:** The 'describe()' function provided summary statistics for the numer ic columns. For instance, HWHHWGT' has a mean of approximately 2388.88 with a mini mum of 0 and a maximum of 9065.66.
- 'PRTAGE' (participant age) ranges from 0 to 85 years, with a mean of around 44 years.

Variables

1. HEFAMINC – This variable depicts the Family Income of the various members of a family and is a numerical value.
2. HWHHWGT – This variable indicates the household weight.
3. HRINTSTA – This indicates the Interview Status and has 4 categories – Interviews, Type A non-interview, Type B non-interview and Type C non -interview.
4. PREXPLF – This indicates the employment status of the population whether a person is employed or non-employed.
5. PESEX – Gender of the population, male or female.
6. PENATVTY – This indicates the country of birth.
7. PRTAGE – Age of the population
8. PRDTOCC1 – This variable indicates the occupation of the people.
9. PWFMWGT – It indicates the Family weight.
10. PWLGWGT - Longitudinal Weight
11. PWORWGT - Outgoing Rotation Weight
12. PWSSWGT – Final Weight
13. PWVETWGT – Veterans Weight

Converting Data Types

Code –

```
# Convert object columns to categorical types
object_columns = df.select_dtypes(include=['object']).columns
df[object_columns] = df[object_columns].astype('category')
# Convert numeric columns to appropriate types for reduced memory usage
# Assuming the columns contain integers
numeric_columns = df.select_dtypes(include=['int64']).columns
df[numeric_columns] = df[numeric_columns].apply(pd.to_numeric, downcast='integer')

# Check memory usage after these changes
print(df.info())
```

Output –

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 13 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   HEFAMINC  4179 non-null   category
 1   HWHHWGT   5000 non-null   float64
 2   HRINTSTA  5000 non-null   category
 3   PREXPLF   1953 non-null   category
 4   PESEX     4179 non-null   category
 5   PENATVTY  4179 non-null   category
 6   PRTAGE    4179 non-null   float64
dtypes: category(6), float64(7)
memory usage: 307.5 KB
```

The code showcases how to optimize memory usage for a DataFrame in Python, specifically for reducing memory usage by converting columns to appropriate data types.

- **Converting Object Columns to Categorical Types:** The code identifies columns with object data types (‘'object'‘) and converts them to the categorical data type (‘'category'‘). This can significantly reduce memory usage, especially for columns with a limited number of unique values that repeat frequently.

- **Converting Numeric Columns to Smaller Numeric Types:** Assuming the numeric columns contain integers, the code intends to downcast them to the smallest appropriate integer type using ‘pd.to_numeric’ with the ‘downcast='integer’‘parameter. However, the example code

provided uses ‘int64’ as the selected type, which might not necessarily be the most memory-efficient type in all cases.

- **Check Memory Usage:** After these changes, the code prints the Data Frame information using ‘df.info()’ to showcase the memory usage reduction. It shows the data types of each column and the total memory usage of the Data Frame.

This process is beneficial for large datasets where memory optimization becomes crucial for faster processing and reduced memory footprint. The conversion to categorical and smaller numeric types can considerably decrease memory usage without compromising the integrity of the data.

Code –

```
df.to_excel('cleaned_file.xlsx', index=False)  # Save cleaned data to a new file
print("downloaded clean data")
# Keep a reference to the cleaned DataFrame
cleaned_df = df.copy()  # Assuming df is the cleaned DataFrame
print("copied clean data")
```

The code performs two key operations:

1. Saving Cleaned Data to a New File: The line ‘df.to_excel('cleaned_file.xlsx', index=False)’ saves the cleaned Data Frame (‘df’) into an Excel file named ‘'cleaned_file.xlsx'‘. The ‘index=False’ parameter ensures that the Data Frame index is not saved as a separate column in the Excel file.

2. Creating a Reference to the Cleaned Data Frame: The line ‘cleaned_df = df.copy()’ makes a copy of the cleaned Data Frame (‘df’) and assigns it to a new variable ‘cleaned_df’. This step can be useful for further analysis or manipulation while preserving the original cleaned dataset.

Summary Statistics Table for Categorical Variables & Numerical Variables

1. **Numerical Variables -** The ‘describe()’ method applied to the ‘cleaned_df’ DataFrame ge nerated descriptive statistics for its numerical columns.

   Here's what the output shows:
   - **Count:** The count row indicates the number of non-null entries in each numerical column. For instance, ‘PRTAGE’ has 4179 non-null entries out of the total 5000 e ntries, implying missing values in this column.
   - **Mean:** Represents the average value for each column.
   - **Standard Deviation (std):** Measures the dispersion or spread of the values in ea ch column. Larger values indicate greater variability from the mean.

- **Minimum and Maximum:** Show the smallest and largest values in each column, r espectively.
- **Percentiles (25%, 50%, 75%):** Provide values below which a given percentage of data falls. For instance, 25% of the values in 'HWHHWGT' are less than 1004.82, while 75% are less than 3460.50

These statistics offer insights into the distribution, central tendency, and variability o f the numerical data in the cleaned DataFrame. They aid in understanding the range of values, presence of outliers, and general summary characteristics of each numeric al column.

Code –

```
num_summary = cleaned_df.describe()
```

Output –

```
Descriptive statistics for numerical columns of cleaned data:
            HWHHWGT       PRTAGE      PWFMWGT       PWLGWGT       PWORWGT
count  5000.000000  4179.000000  5000.000000   5000.000000   5000.000000
mean   2388.882525    44.086863  2423.032584   2109.146015   2264.149462
std    1494.271754    23.620775  1537.350366   2391.514162   5023.167621
min       0.000000     0.000000     0.000000      0.000000      0.000000
25%    1004.822000    24.000000   979.526525      0.000000      0.000000
50%    2904.956500    46.000000  2903.490000    493.154950      0.000000
75%    3460.504000    63.000000  3520.630500   4508.279500      0.000000
max    9065.664000    85.000000  9639.847000  12966.150000  35534.110000
```

2. Categorical Variables

- **Value Counts or Percentages for Categorical Columns:** The output presents the normalized value counts or percentages for each unique category within categori cal columns.
- For the column 'HEFAMINC', '(01) Less than $5,000' comprises approximately 2.0 6% of the data.
- Similarly, other categories in different columns show their respective proportion s within the dataset.
- **Presence of Missing Values:** Some categories within the categorical columns see m to have NaN values (missing data) because the percentage for those categorie s is shown as NaN in the summary. This suggests that those specific categories mi ght not exist or have no occurrences in the dataset.

This summary provides an overview of the distribution or prevalence of different categories within each categorical column, allowing for a quick understanding of the composition and relative importance of various categories in the dataset.

Code –

```
cat_summary = cleaned_df.select_dtypes(include='category').apply(lambda x: x.value_counts(normalize=True))
print("Value counts or percentages for categorical columns of cleaned data: n", cat_summary)
```

Output –

```
   Value counts or percentages for categorical columns of cleaned data:
    HEFAMINC  HRINTSTA
(01) Less than $5,000                              0.020579
(01) Management occupations                             NaN
(02) 5,000 to 7,499                                0.011247
(02) Business and financial operations occupations      NaN
(03) 7,500 to 9,999                                0.018425
                                                   PRDTOCC1
(01) Less than $5,000                                   NaN
(01) Management occupations                        0.105210
(02) 5,000 to 7,499                                     NaN
(02) Business and financial operations occupations 0.053111
[139 rows x 6 columns]
```

3. **Cross Tabulation -** The provided code above provides a cross-tabulation using ‘pd.crosstab’ between two categorical columns, namely 'PREXPLF' and 'PESEX', from the ‘cleaned_df’ Data Frame. It creates a table that showcases the frequency distribution of occurrences for each unique combination of categories between 'PREXPLF' and 'PESEX'.

   Output Interpretation:

   For instance -

   - Under 'PREXPLF', there are two categories: '(1) Employed' and '(2) Unemployed'.
   - Under 'PESEX', there are two categories: '(1) Male' and '(2) Female'.
   - The table displays the counts of occurrences where, for example, 956 entries correspond to '(1) Employed' and '(1) Male', 930 entries correspond to '(1) Employed' and '(2) Female', and so on.

- This cross-tabulation helps understand the relationships or associations between categorical variables by displaying how the categories within one variable relate to the categories of another variable in the dataset. It's useful for exploring potential patterns or dependencies between categorical variables.

Output –

```
Cross-tabulation between 'PREXPLF' and 'PESEX':
PESEX           (1) Male  (2) Female
PREXPLF
(1) Employed         956         930
(2) Unemployed        35          32
```

Data Visualization

1. **Histogram for 'AGE' Column:** Displays the distribution of values in the 'AGE' column using a histogram with 20 bins.

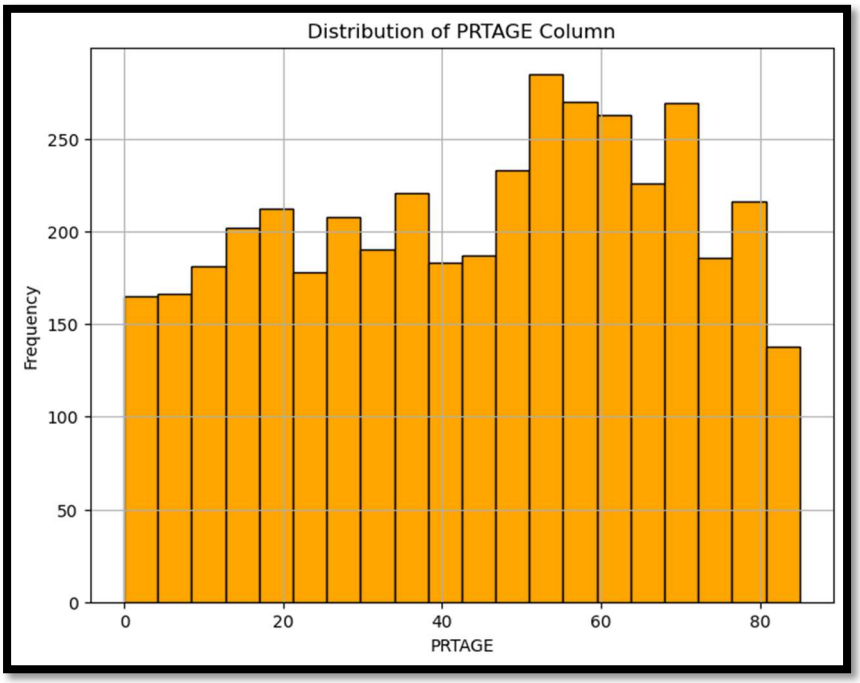

Figure 1: - Histogram of Age

The graph above illustrates the frequency distribution of the 'age' column in the population dataset. It indicates a higher frequency of individuals aged between 50 and 60, with a frequency exceeding 250. There is a lower frequency among the elderly, specifically those aged 80 and above, followed by children ranging from 0 to 10 years old.

2. **Boxplots for 'Household Weight' and 'Family Weight' Columns:** Shows the distribution, median, quartiles, and any outliers for the 'Household Weight' and 'Family Weight' columns using boxplots.

   Box plot for Household weights (HWHHWGT):

   - Median (Q2 or 50th percentile): 2904.9565
   - Quartiles (Q1, Q3): Q1 (25th percentile): 1004.822, Q3 (75th percentile): 3460.504
   - Interquartile Range (IQR):Q3 - Q1 = 3455.682
   - Whiskers:
   - Lower whisker: Typically, 1.5 * IQR below Q1, which would be around -3813.474 (but cannot be less than 0)
   - Upper whisker: Typically, 1.5 * IQR above Q3, which would be around 7928.8, but the maximum value is 9065.664, so the upper whisker extends to the maximum value within 1.5 * IQR range.
   - Outliers are present at 7500, 8500, and 9065.664(approximately).
   - The box plot would show a box ranging from approximately 1004.822 to 3460.504, with the median line at 2904.9565. The whisker would extend up to the maximum value within the acceptable range (1.5 * IQR) or the actual maximum value if it's within that range. Additionally, three outliers are observed, displayed as individual points outside the whiskers.

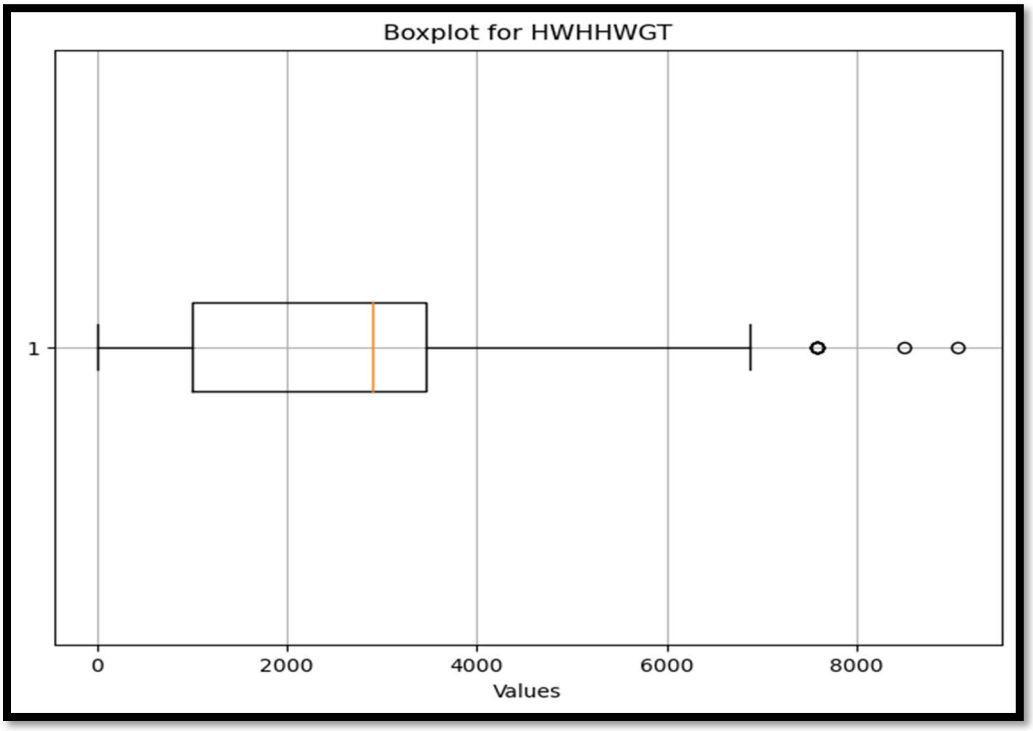

Figure 2: - Box plot of Household Weights

Box plot for Family weights (PWFMWGT):

- **Median (Q2 or 50th percentile):** 3194.5992 (represented by the line within the box)
- **Quartiles (Q1, Q3):** Q1 (25th percentile): 2286.8206 (bottom edge of the box), Q3 (75th percentile): 3669.1875 (top edge of the box)
- **Interquartile Range (IQR):** 1382.3669 (difference between Q3 and Q1)
- **Whiskers:**
    - Lower whisker extends to approximately 218.1183 (minimum value within 1.5 times IQR from Q1)
    - Upper whisker extends to 5861.9294 (maximum value within 1.5 times IQR from Q3)
- Outliers: Five outliers were identified at 7907, 8007, 8235, 8557, 9065.663 (approximately) (points beyond the whiskers).

The box plot would show a box ranging from approximately 2286.82 to 3669.1875, with the median line at 3194.5992. The whisker would extend up to the maximum value within the acceptable range (1.5 * IQR) or the actual maximum value if it's within that range. Additionally, five outliers are observed, displayed as individual points outside the whiskers.

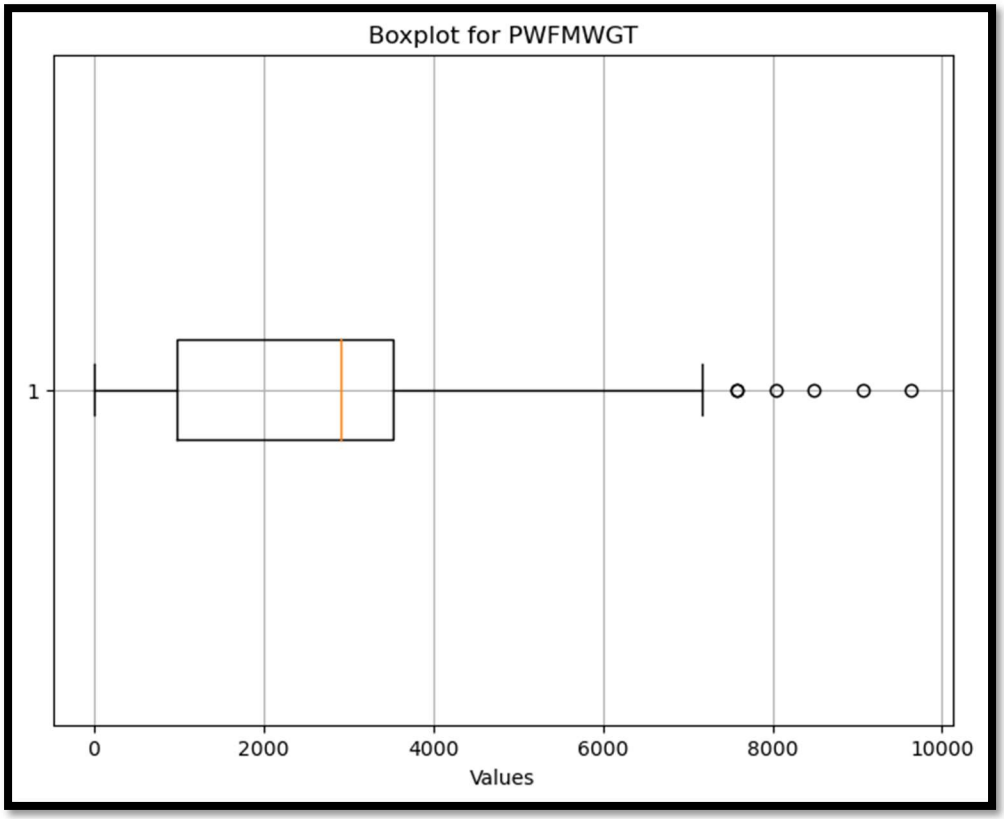

Figure 3: - Box plot of Family Weights

3. **Bar Plot for 'Interview Status' Column:** Represents the count of different categories in the 'Interview Status' column using a bar plot.
    - Interview: There are 4179 instances where the status indicates an interview.
    - Type B non-interview: This category, labeled as 'Type B non-interview,' comprises 801 instances where it signifies a non-interview status of a certain type (specific to Type B).
    - Type A non-interview: There are 18 instances categorized as 'Type A non-interview,' indicating a different non-interview status from Type B.
    - Type C non-interview: This category contains only 2 instances categorized as 'Type C non-interview,' representing a specific non-interview status different from Type A and Type B.
    - These categories likely represent different types or statuses related to interviews and non-interviews within the dataset, providing insights into the distribution of interview statuses among the data.

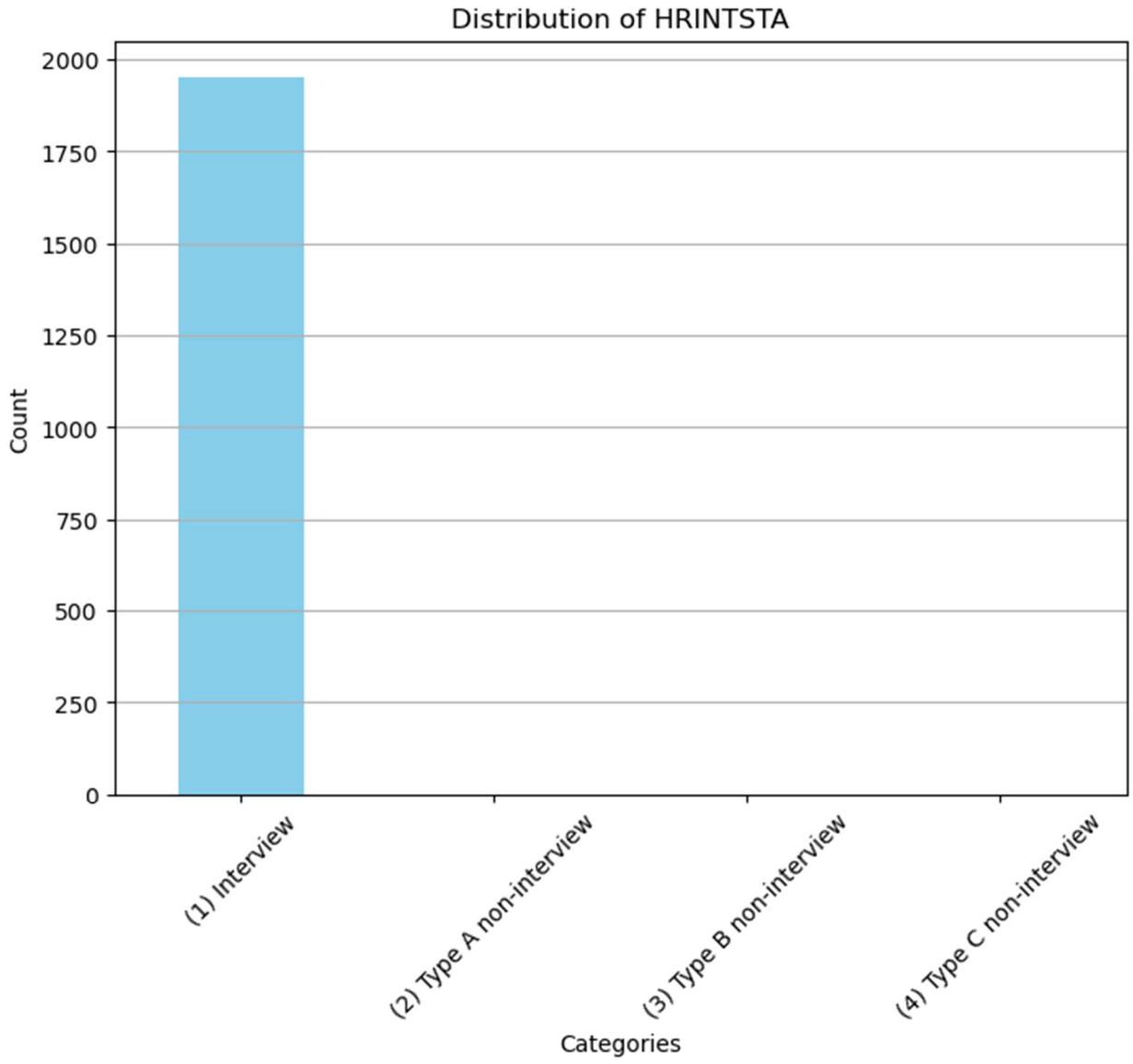

Figure 4: - Bar plot of Interview Status

4. **Pie Chart for 'PESEX' Column:** Illustrates the distribution of categories in the 'PESEX' column using a pie chart.
   - Male (49.3%): This segment would represent slightly less than half of the pie chart, indicating the proportion of males in the dataset.
   - Female (50.7%): This segment would occupy the larger portion of the pie chart, representing most of the dataset, which comprises females.
   - This pie chart offers a quick and easy-to-understand visualization of the gender distribution, highlighting that while males constitute almost half of the dataset, the majority—more than half—is constitute of females.

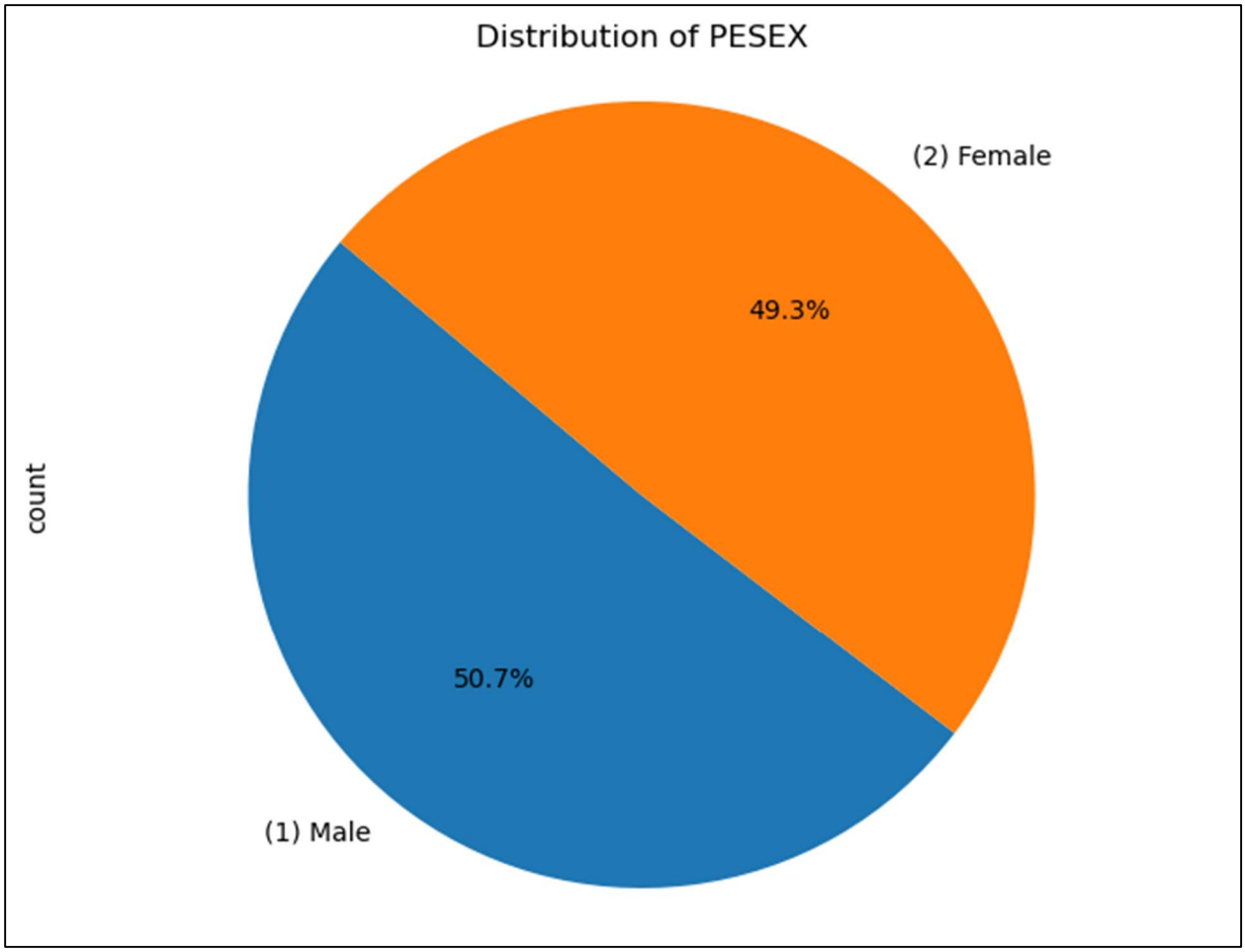

Figure 5: - Pie Chart of Gender

Models

1. **Importing Libraries:**
    - 'pandas' is imported as 'pd' for data manipulation.
    - Specific modules from scikit-learn ('LinearRegression', 'SimpleImputer', 'train_test_split') are imported for machine learning tasks.
2. **Data Preparation Steps:**
    - 'cleaned_df' is assumed to be a DataFrame.
    - 'cleaned_numeric' is created by selecting columns of numeric type using 'select_dtypes(include=['number'])'.
    - Missing values in these numeric columns are filled with their mean values using 'fillna()'.
3. **Printed Information:**

- ‘cleaned_numeric.info’ seems to attempt printing the information about the cleaned numeric DataFrame.
- However, this code seems incorrect as it doesn't include the parentheses ‘()’ after ‘info’. It should be ‘cleaned_numeric.info()’ to call the ‘info()’ method, which provides information about the DataFrame, including the data types, memory usage, and non-null counts for each column.

Three models for analysis are as follows –

1. **Linear Regression –** Linear regression is a statistical method used to model the relationship between a dependent variable and one or more independent variables by fitting a linear equation to observed data. It assumes a linear relationship between the independent and dependent variables.
   In its simplest form, with one independent variable (univariate linear regression), the equation for a straight line is:
   [y = mx + c]
   - (y) represents the dependent variable.
   - (x) represents the independent variable.
   - (m) is the slope of the line, representing the change in (y) for a unit change in (x).
   - (c) is the intercept, representing the value of (y) when (x) is zero.

   The regression analysis provides insights into the relationships between variables, the significance of predictors, the model's goodness of fit (R-squared), and the potential impact of each predictor on the dependent variable.

   Output –

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                PWSSWGT   R-squared:                       0
.963
Model:                            OLS   Adj. R-squared:                  0
.963
Method:                 Least Squares   F-statistic:                 3.610
e+04
Date:                Mon, 04 Dec 2023   Prob (F-statistic):
0.00
Time:                        12:36:09   Log-Likelihood:                -28
751.
No. Observations:                4179   AIC:                         5.751
e+04
Df Residuals:                    4175   BIC:                         5.754
e+04
Df Model:                           3
Covariance Type:            nonrobust
```

```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         33.3018     12.565      2.650      0.008       8.668      57.935
HWHHWGT        0.0044      0.008      0.523      0.601      -0.012       0.021
PWFMWGT        0.9917      0.008    123.326      0.000       0.976       1.007
PRTAGE        -0.3467      0.156     -2.226      0.026      -0.652      -0.041
==============================================================================
Omnibus:                     2566.544   Durbin-Watson:                   1.999
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          2968937.091
Skew:                           1.432   Prob(JB):                         0.00
Kurtosis:                     133.547   Cond. No.                     1.51e+04
==============================================================================
```

2. **Logistic Regression –** Logistic Regression serves as a fundamental Machine Learning classification algorithm, primarily designed to estimate the probability of a categorical dependent variable. In this context, the dependent variable is typically binary, encapsulating data coded as 1 (representing positive outcomes like success) or 0 (denoting negative outcomes like failure). Put simply, logistic regression models the probability, P(Y=1), as a function of the predictor variables (X).

```
              precision    recall  f1-score   support

  (1) Employed       0.96      1.00      0.98       377
  (2) Unemployed       0.00      0.00      0.00        14

  accuracy                           0.96       391
  macro avg       0.48      0.50      0.49       391
  weighted avg       0.93      0.96      0.95       391
```

3. **Lasso Regression Model** - Lasso regression, short for Least Absolute Shrinkage and Selection Operator, is a regression technique used for feature selection and regularization to prevent overfitting in statistical models, especially in the context of linear regression. Like ridge regression, lasso regression adds a penalty term to the ordinary least squares objective function. However, while ridge regression penalizes sum

of squares of coefficients (L2 norm penalty), lasso penalizes the sum of absolute values of the coefficients (L1 norm penalty).
Higher values of (lambda) result in more shrinkage of coefficients towards zero, effectively performing variable selection by setting some coefficients to exactly zero.

Lasso regression encourages sparsity in the model by forcing some coefficients to be precisely zero, effectively eliminating certain features from the model. This property makes lasso regression useful for feature selection when dealing with datasets containing many irrelevant or less important features.

In Python, you can perform lasso regression using libraries such as scikit-learn, which provides the 'Lasso' class for fitting lasso regression models to data. It allows you to adjust the regularization strength (lambda)) and explore the impact of the penalty on the coefficients and feature selection.

Output – `Coefficients: [ 0.04128686  0.95263669 -1.64040944]`

4. **Decision Tree Classifier** - A Decision Tree Classifier is a supervised machine learning algorithm used primarily for classification tasks. It operates by recursively partitioning the feature space into distinct regions or classes based on a sequence of decision rules inferred from the training data.

Output –
```
|--- HWHHWGT <= 4604.87
|   |--- PRTAGE <= 25.50
|   |   |--- HWHHWGT <= 3733.76
|   |   |   |--- HWHHWGT <= 3721.72
|   |   |   |   |--- HWHHWGT <= 929.61
|   |   |   |   |   |--- PRTAGE <= 16.50
|   |   |   |   |   |   |--- class: (2) Unemployed
|   |   |   |   |   |--- PRTAGE >  16.50
|   |   |   |   |   |   |--- PRTAGE <= 24.50
|   |   |   |   |   |   |   |--- class: (1) Employed
|   |   |   |   |   |   |--- PRTAGE >  24.50
|   |   |   |   |   |   |   |--- HWHHWGT <= 455.84
|   |   |   |   |   |   |   |   |--- class: (1) Employed
|   |   |   |   |   |   |   |--- HWHHWGT >  455.84
|   |   |   |   |   |   |   |   |--- class: (2) Unemployed
|   |   |   |   |--- HWHHWGT >  929.61
|   |   |   |   |   |--- HWHHWGT <= 2214.90
|   |   |   |   |   |   |--- class: (1) Employed
|   |   |   |   |   |--- HWHHWGT >  2214.90
|   |   |   |   |   |   |--- HWHHWGT <= 2223.81
|   |   |   |   |   |   |   |--- class: (2) Unemployed
|   |   |   |   |   |   |--- HWHHWGT >  2223.81
|   |   |   |   |   |   |   |--- PRTAGE <= 24.50
|   |   |   |   |   |   |   |   |--- PRTAGE <= 23.50
|   |   |   |   |   |   |   |   |   |--- HWHHWGT <= 2692.66
|   |   |   |   |   |   |   |   |   |   |--- HWHHWGT <= 2615.87
```

Conclusion

This exploratory data analysis provides initial insights into the dataset, highlighting the distributions and relationships between key variables. This preliminary exploration of the Current Population Survey dataset lays the groundwork for a comprehensive Capstone Project. The dataset's dimensions, encompassing over 5000 records and 13 variables, ensure a robust analysis. The questions help us to understand the sociodemographic aspects, guiding the investigation toward various factors influencing employment, income, and weight other critical variables.

It looks like we've executed a comprehensive data analysis pipeline, ranging from data loading and cleaning to exploratory data analysis, linear regression, logistic regression, Lasso regression, and decision tree modeling. Each step seems to have been handled methodically, and you've utilized various visualization techniques to explore and understand the data.

References

1) Current Population Survey, September 2017: Volunteering and Civic Life supplement. (2019, May 20). https://www.icpsr.umich.edu/web/ICPSR/studies/37303/variables?q=employment
2) GeeksforGeeks. (2023, September 4). Pandas DF.size df.shape and Df.Ndim methods. https://www.geeksforgeeks.org/python-pandas-df-size-df-shape-and-df-ndim/
3) Verma, J. (2020, October 7). How to calculate summary statistics in Python? - AskPython. https://www.askpython.com/python/examples/calculate-summary-statistics
4) Summary Statistics for Categorical data: Summary Statistics for Categorical Data cheatsheet | Codecademy. (n.d.). Codecademy. https://www.codecademy.com/learn/stats-summary-statistics-for-categorical-data/modules/stats-summary-statistics-for-categorical-data/cheatsheet
5) How to calculate summary statistics — pandas 2.1.3 documentation. (n.d.). https://pandas.pydata.org/docs/getting_started/intro_tutorials/06_calculate_statistics.html
6) Python, R. (2023, June 26). Logistic regression in Python. https://realpython.com/logistic-regression-python/

Appendix

The appendix section will include relevant portions of code used for data preprocessing, analysis, and model building. Summary statistics or visualizations that provide more detailed insights. A concise description of each variable in the dataset, including data types and possible values. The Python Script **"Group5_ALY6140_Capstone Final Project Draft Report"** and **Report "Group5_ALY6140_Capstone Final Project Draft Report"** is made and ready for submission.