Exploratory Data Analysis (EDA) Using Python Libraries

Contents

1. INTRODUCTION

1.1 Project Overview

Data analysis is a crucial aspect of extracting meaningful insights from raw information. In this project, our group undertakes the task of Exploratory Data Analysis (EDA) using Python, a process that involves understanding the structure, characteristics, and relationships within a given dataset. EDA serves as a foundational step before more advanced analytics or machine learning applications.

1.2 Objectives

The primary objectives of this project are:

- To gain practical experience in conducting EDA using Python.

- To enhance our skills in data loading, data manipulation, and data visualization.

- To extract meaningful insights and patterns from a real-world dataset.

- To present findings in a clear and organized manner.

1.3 Dataset Information

1.3.1 Dataset Name

The dataset selected for this project is **Employee Sample Data**. It has been chosen for its relevance to Folks in Human Resources actually deal with a lot of data. This data can be great for creating dashboards and summarizing various aspects of a company’s workforce. In this database, there are **1,000 rows of data** encompassing popular data points that HR professionals deal with on a regular basis.

1.3.2 Dataset Link

The dataset can be accessed https://www.thespreadsheetguru.com/sample-data/

1.3.3 Dataset Description

Columns in this Data Set:

Below is a list of all the fields of data included in the sample data.

Employee ID

Full Name

Job Title

Gender

Ethnicity

Age

Hire Date

Annual Salary (USD)

Bonus %

Department

Business Unit

Country

City

Exit Date

As we progress through the project, we aim to uncover meaningful insights, patterns, and potential correlations within this dataset, contributing to our understanding of data analysis in a real-world context.

2. GROUP MEMBERS

Our project team consists of three dedicated individuals, each contributing their skills and expertise to ensure the success of this EDA endeavour.

2.1 Student 1

- ❖ **Name**: Ramya.M
- ❖ **Role** : Testing and Research
- ❖ **Contact**: ramyamurthy052006@gmail.com

2.2 Student 2

- ❖ **Name**: Subhashini
- ❖ **Role**: Documentation and Analysis
- ❖ **Contact**: 6374240601

Working collaboratively, each team member brings a unique set of skills and perspectives to the project, fostering a dynamic and effective environment for exploration and analysis.

3. SETTING UP THE ENVIRONMENT

Before diving into the data analysis process, it is crucial to ensure that the necessary software and libraries are installed. This section outlines the steps to set up the environment for conducting Exploratory Data Analysis (EDA) using Python.

3.1 Install Python Software

The first step is to install the Python programming language.

Follow these steps:

- ❖ Visit the official [Python website](https://www.python.org/).
- ❖ Navigate to the "Downloads" section.
- ❖ Download the latest version of Python for your operating system (Windows, macOS, or Linux).
- ❖ Run the installer and follow the installation instructions.

To verify the installation, open a command prompt or terminal and type:

```
python --version
```

This should display the installed Python version.

3.2 Install Required Libraries

For this EDA project, we will be using two essential Python libraries:

3.2.1 Pandas Overview

Pandas is a powerful and widely-used open-source data manipulation and analysis library for Python. It provides data structures like DataFrames and Series, which are designed for efficient data analysis and manipulation. Pandas simplifies various tasks related to working with structured data, making it an essential tool for data scientists, analysts, and researchers.

Key Features of Pandas:

1. **DataFrame:** A two-dimensional labeled data structure with columns that can be of different types. It is similar to a spreadsheet or SQL table.
2. **Series:** A one-dimensional labeled array capable of holding any data type.
3. **Data Cleaning:** Pandas provides functions to handle missing data, duplicate values, and other data cleaning tasks.
4. **Data Alignment:** Easily align and manipulate data from different sources.
5. **Grouping and Aggregation:** Efficiently group data and perform aggregation operations.
6. **Time Series Analysis:** Powerful tools for handling time series data.
7. **Input/Output:** Read and write data in various formats, including CSV, Excel, SQL databases, and more.

Installing Pandas

This command will download and install the latest version of Pandas from the Python Package Index (PyPI). Make sure your Python environment is set up correctly before running this command.

```
pip install pandas
```

Verifying the Installation

After the installation is complete, you can verify it by importing Pandas in a Python script or Jupyter Notebook:

```
import pandas as pd
```

If no error occurs, Pandas is successfully installed

This statement imports Pandas and aliases it as `pd`, a common convention in the Pandas community. (https://pandas.pydata.org/)

Pandas is now ready to use in your Python environment!

3.2.2 Matplotlib Overview

Matplotlib is a comprehensive data visualization library for Python. It enables the creation of high-quality static, animated, and interactive visualizations in Python. Matplotlib is widely used for creating a variety of plots, charts, and graphs to explore and communicate insights from data.

Key Features of Matplotlib:

1. **Versatile Plotting:** Matplotlib supports a wide range of plots, including line plots, scatter plots, bar plots, histograms, pie charts, and more.

2. **Customization:** Users can extensively customize the appearance of plots, adjusting colors, labels, markers, and other visual elements.

3. **Publication-Quality Graphics:** Matplotlib is capable of producing publication-quality graphics for scientific journals and presentations.

4. **Interactive Visualizations:** Matplotlib provides tools for creating interactive plots, enhancing the exploration of data.

5. **Integration with NumPy:** Matplotlib seamlessly integrates with NumPy, making it easy to visualize data stored in NumPy arrays.

6. **Matplotlib.pyplot Interface:** The `pyplot` module in Matplotlib provides a convenient MATLAB-style interface for creating and customizing plots.

Installing Matplotlib

To install Matplotlib, you can use the package manager pip . Open your command prompt or terminal and run the following command:

```
pip install matplotlib
```

This command will download and install the latest version of Matplotlib from the Python Package Index (PyPI). Make sure your Python environment is set up correctly before running this command.

Verifying the Installation

After the installation is complete, you can verify it by importing Matplotlib in a Python script or Jupyter Notebook:

```
import matplotlib.pyplot as plt
```

If no error occurs, Matplotlib is successfully installed

This statement imports Matplotlib and aliases it as `plt`, a common convention in the Matplotlib community.

Matplotlib is now ready to use in your Python environment.

(https://matplotlib.org/)

These libraries play a crucial role in loading, manipulating, and visualizing the dataset during the EDA process. By ensuring they are installed, we create a solid foundation for our analysis.

4. DATA LOADING AND EXPLORATION

In this section, we focus on loading the dataset and conducting preliminary exploration to understand its structure and characteristics.

4.1 Read the Dataset

4.1.1 Using Pandas `read_csv()`

The dataset will be loaded into our Python environment using the powerful Pandas library. The `read_csv()` function is employed for this purpose.

```
import pandas as pd


# Specify the path to the dataset
dataset_path = "path/to/your/dataset.csv"


# Read the dataset into a Pandas DataFrame
df = pd.read_csv(dataset_path)
```

4.1.2 Display the First Five Rows

Let's display the initial rows of the dataset to gain a quick overview of its structure.

```
# Display the first five rows of the dataset
print(df.head())
```

4.2 Dataset Shape

To understand the size of the dataset, we print its shape using the `shape` attribute.

```
# Print the shape of the dataset

print("Dataset Shape:", df.shape)
```

4.3 Dataset Summary

4.3.1 Descriptive Statistics

Generating descriptive statistics provides an overview of the central tendency, dispersion, and shape of the dataset's distribution.

```
# Display descriptive statistics of the dataset

print("Descriptive Statistics:")

print(df.describe())
```

4.4 Columns and Data Types

Let's inspect the columns of the dataset and their corresponding data types.

```
# Display the columns and their data types

print("Columns and Data Types:")

print(df.dtypes)
```

By executing these steps, we lay the foundation for a deeper understanding of the dataset, setting the stage for subsequent exploratory analyses.

5. CONCLUSION

In this section, we summarize key findings, insights gained from the EDA process, and discuss the implications and potential next steps.

5.1 Key Findings

Demographic Overview:

- The dataset includes 1000 entries representing individuals.
- Age is a prominent feature, ranging from 25 to 65 years.
- The mean age is approximately 44.38 years.

Missing Values:

- The "Exit Date" column has missing values, indicating that individuals might still be associated with the organization.

Geographical Distribution:

- Individuals are associated with cities such as Seattle, Chongqing, Chicago, and Phoenix.

5.2 Insights from EDA

Age Distribution:

- The age distribution is relatively balanced, with a mean age around 44.38 years.
- Most individuals fall within the age range of 35 to 54 years.

City Distribution:

- The dataset captures individuals from diverse cities, suggesting a widespread geographical representation.

Exit Date Exploration:

- The presence of missing values in the "Exit Date" column indicates that not all individuals have left the organization or that exit dates are not uniformly recorded.

5.3 Implications and Next Steps

Further Exploration:

- Investigate the reasons for missing "Exit Date" values. It could be valuable to distinguish between active and inactive individuals.

Detailed Analysis:

- Conduct a more granular analysis of age distribution within specific cities to identify any patterns or trends.

Predictive Modeling:

- Explore the potential of predictive modeling to estimate or infer missing "Exit Date" values based on available data.

Employee Retention Strategies:

- Identify factors influencing employee exits and consider implementing retention strategies, especially for cities with higher attrition.

Feedback Collection:

- If applicable, gather feedback from individuals who have exited the organization to understand their reasons for leaving and inform future HR strategies.

6. APPENDIX

In this section, you can find the Python code used for the analysis and any additional screenshots that supplement the main content.

6.1 Python Code

6.1.1 Data Loading and Exploration

```
# Python code for loading and exploring the dataset
import pandas as pd


# Read the dataset using pandas read_csv()
df = pd.read_csv('Employee_Data.csv')


# Display the first five rows
print("First five rows of the dataset:")
print(df.head())


# Get the shape of the data
print("\nDataset shape:")
print(df.shape)


# Descriptive statistics
print("\nDescriptive statistics:")
print(df.describe())
```

6.1.2 Data Visualization

```
# Python code for data visualization
import matplotlib.pyplot as plt

# Selecting a column for visualization
selected_column = "example_column"

# Plotting a histogram
plt.figure(figsize=(10, 6))
plt.hist(df[selected_column], bins=30, color='skyblue', edgecolor='black')
plt.title(f'Histogram of {selected_column}')
plt.xlabel(selected_column)
plt.ylabel('Frequency')
plt.show()
```

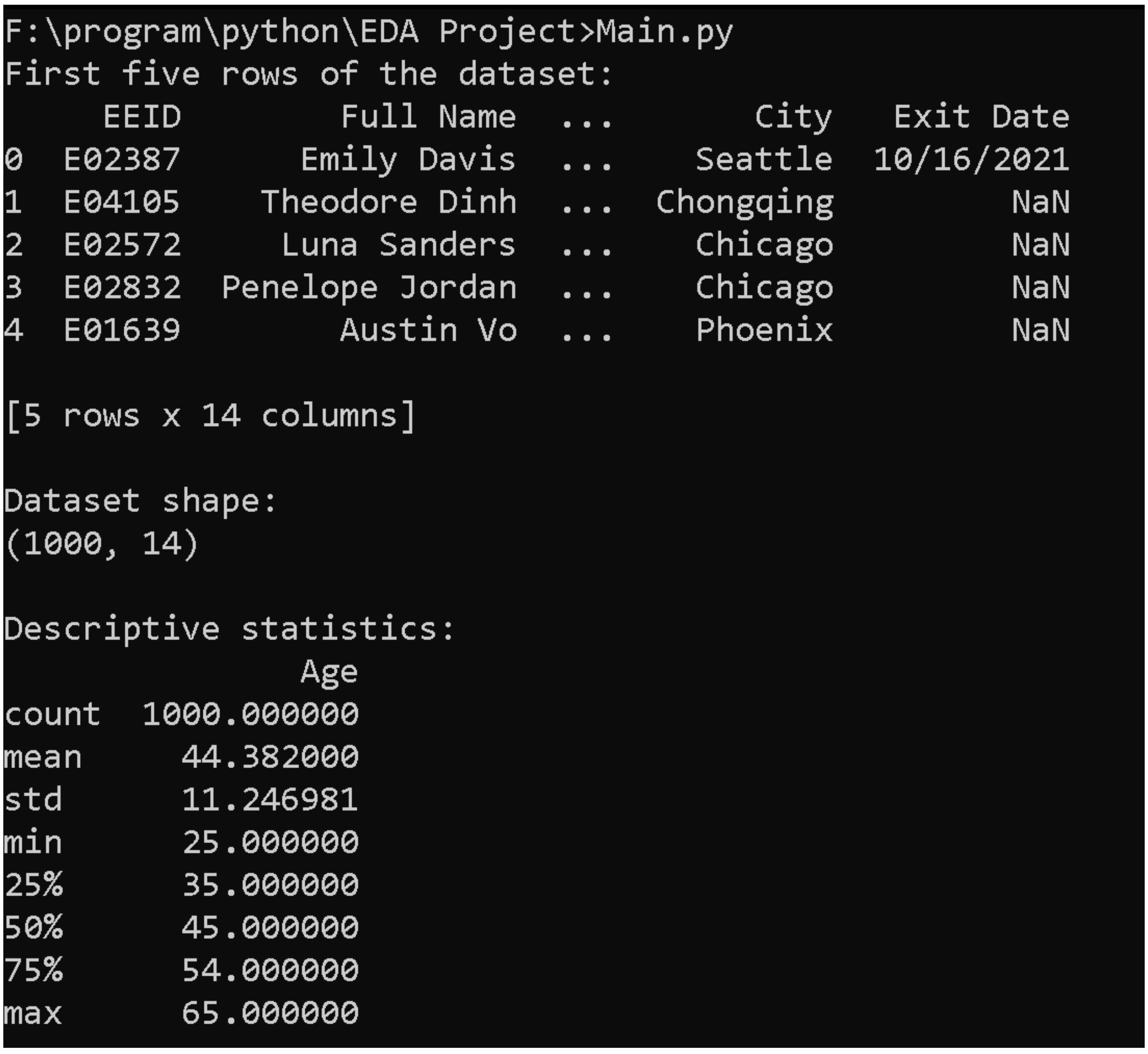
6.2 Complete Source Code:

```
import pandas as pd
import matplotlib.pyplot as plt

# Read the dataset using pandas read_csv()
df = pd.read_csv('Employee_Data.csv')

# Display the first five rows
print("First five rows of the dataset:")
print(df.head())
```

```
# Get the shape of the data
print("\nDataset shape:")
print(df.shape)

# Descriptive statistics
print("\nDescriptive statistics:")
print(df.describe())



# Selecting a column for visualization
selected_column = "Age"

# Plotting a histogram
plt.figure(figsize=(10, 6))
plt.hist(df[selected_column], bins=30, color='skyblue', edgecolor='black')
plt.title(f'Histogram of {selected_column}')
plt.xlabel(selected_column)
plt.ylabel('Frequency')
plt.show()
```

6.3 Additional Screenshots

Output For Pandas:



```
F:\program\python\EDA Project>Main.py
First five rows of the dataset:
     EEID        Full Name  ...       City   Exit Date
0  E02387      Emily Davis  ...    Seattle  10/16/2021
1  E04105    Theodore Dinh  ...  Chongqing         NaN
2  E02572     Luna Sanders  ...    Chicago         NaN
3  E02832  Penelope Jordan  ...    Chicago         NaN
4  E01639        Austin Vo  ...    Phoenix         NaN

[5 rows x 14 columns]

Dataset shape:
(1000, 14)

Descriptive statistics:
               Age
count  1000.000000
mean     44.382000
std      11.246981
min      25.000000
25%      35.000000
50%      45.000000
75%      54.000000
max      65.000000
```

Output for MatPlotLib:

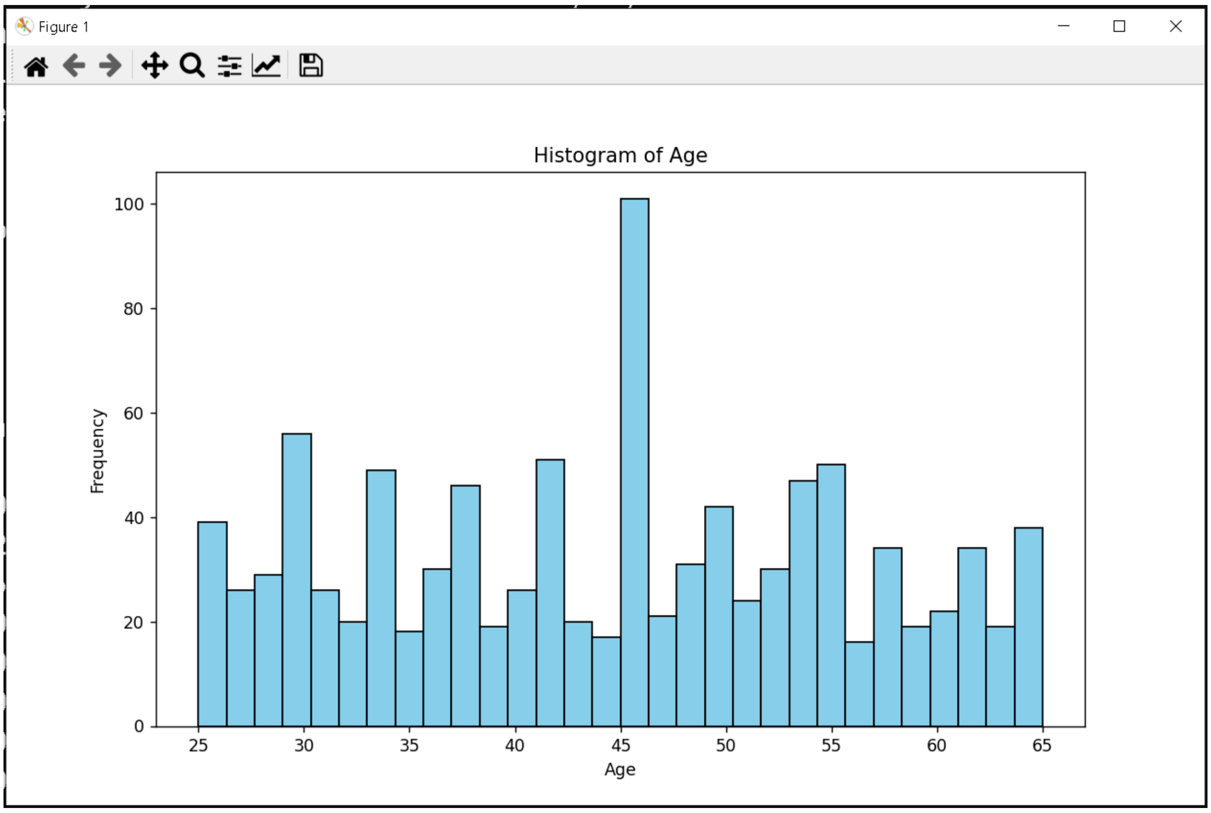


Figure 1
Histogram of Age
Frequency
Age