MIT World Peace University

Python Programming
Second Year B. Tech, Semester 4

Learning Basics of the Pandas and Matplotlib Libraries

Assignment No. 7

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 20

April 24, 2023

Contents

1 Aim

Write a python code to read a .csv file using panda's module and print the first and last five records of the file. Using Matplotlib shows data analysis.

2 Objectives

1. To learn and implement Function of Pandas and Matplotlib modules.

3 Problem Statement

Use of Pandas module for data analysis and Matplotlib for data visualization.

4 Theory

4.1 Pandas

Pandas is a popular open-source Python library used for data manipulation and analysis. It provides high-performance, easy-to-use data structures and data analysis tools for working with structured data, such as tables or spreadsheets. Pandas is built on top of the NumPy library, which provides support for efficient array operations.

One of the key data structures in Pandas is the DataFrame, which is a two-dimensional labeled array with columns of potentially different types. DataFrames are used extensively in data analysis, as they allow users to easily manipulate and query data. For example, you can filter rows based on certain criteria, aggregate data, or merge different tables together.

Another important data structure in Pandas is the Series, which is a one-dimensional labeled array. A Series is similar to a column in a DataFrame, and can be used to represent time-series data, text data, or any other type of data that can be represented as a sequence of values. Series can be created from a Python list, dictionary, or NumPy array.

4.2 Matplotlib

Matplotlib is a data visualization library for Python that provides a variety of 2D and 3D plotting functions. It is widely used in the scientific community for creating high-quality visualizations of data. Matplotlib is built on top of NumPy, which makes it easy to work with arrays and manipulate data.

One of the key features of Matplotlib is its ability to create publication-quality plots with just a few lines of code. Matplotlib provides a wide range of plot types, including line plots, scatter plots, bar plots, and histograms. It also allows users to customize the appearance of plots, including colors, fonts, and labels.

Matplotlib is highly customizable, which makes it suitable for creating complex visualizations. For example, it allows users to create subplots, add annotations, and create animations. Matplotlib also provides support for different file formats, such as PNG, PDF, and SVG, which makes it easy to save and share visualizations.

4.3 Different types of Data Structures in Pandas

Pandas provides several data structures for working with structured data, including:

1. DataFrame - a two-dimensional labeled array with columns of potentially different types.
2. Series - a one-dimensional labeled array with homogeneous data.
3. Panel - a three-dimensional labeled array.
4. Panel4D - a four-dimensional labeled array.
5. PanelND - a labeled N-dimensional array.

The most commonly used data structure in Pandas is the DataFrame, which is used for storing and manipulating tabular data. A DataFrame is a two-dimensional labeled array with columns of potentially different types. It can be thought of as a spreadsheet or a SQL table, with rows and columns of data.

A Series is a one-dimensional labeled array with homogeneous data. It can be thought of as a single column of a DataFrame. Series are often used to represent time-series data or other types of data that can be represented as a sequence of values.

Panels are higher-dimensional data structures that are less commonly used than DataFrames and Series. Panels can be thought of as a three-dimensional labeled array, where each item in the array represents a DataFrame. Panel4D and PanelND are higher-dimensional versions of Panel, with four and N dimensions, respectively. These structures are less commonly used in practice, as most data can be represented using DataFrames or Series.

All of these data structures are designed to be flexible and efficient, allowing users to perform a wide range of data manipulation and analysis tasks. They support a variety of data types, including numerical, categorical, and textual data, and can be easily combined and transformed using built-in functions and methods.

4.4 Reading data from csv file

Pandas provides a simple and efficient way to read data from CSV files, which are commonly used for storing tabular data. The readcsv() function in Pandas can be used to read data from a CSV file and create a DataFrame object in memory.

Here is an example of reading data from a CSV file named data.csv using the readcsv() function:

```
    import pandas as pd

    df = pd.read_csv('data.csv', delimiter='\t')
```

The readcsv() function also supports a number of other parameters for handling different types of data, including delimiter and encoding. For example, you can use the delimiter parameter to specify a custom delimiter for the file, such as a tab or semicolon:

5 Platform

Operating System: Arch Linux x86-64
IDEs or Text Editors Used: Visual Studio Code with Jupyter
Interpreter: python 3.10.8

6 Input and Output

6.1 Input

Reading data from 'csv file' for data analysis operation.

6.2 Output

Data analysis and visualization of data.

7 Requirements

1. Python 3.7 or above
2. Pandas
3. Matplotlib

8 Code

[2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

[3]:
```
# Create a simple dataframe
toyota_df = pd.read_csv('../Lab/Assignment 7/Toyota.csv')
```

8.0.1 Reading First few Values

[4]:
```
toyota_df.head()
```

[4]:
```
   Unnamed: 0  Price   Age     KM FuelType  HP  MetColor  Automatic    CC  \
0           0  13500  23.0  46986   Diesel  90       1.0          0  2000
1           1  13750  23.0  72937   Diesel  90       1.0          0  2000
2           2  13950  24.0  41711   Diesel  90       NaN          0  2000
3           3  14950  26.0  48000   Diesel  90       0.0          0  2000
4           4  13750  30.0  38500   Diesel  90       0.0          0  2000

   Doors  Weight
0  three    1165
1      3    1165
2      3    1165
3      3    1165
4      3    1170
```

[5]:
```
toyota_df.tail()
```

[5]:
```
      Unnamed: 0  Price   Age     KM FuelType   HP  MetColor  Automatic    CC  \
1431        1431   7500   NaN  20544   Petrol   86       1.0          0  1300
1432        1432  10845  72.0     ??   Petrol   86       0.0          0  1300
1433        1433   8500   NaN  17016   Petrol   86       0.0          0  1300
1434        1434   7250  70.0     ??      NaN   86       1.0          0  1300
1435        1435   6950  76.0      1   Petrol  110       0.0          0  1600

      Doors  Weight
1431      3    1025
1432      3    1015
1433      3    1015
1434      3    1015
1435      5    1114
```

[6]:
```
automobile_df = pd.read_csv('../Lab/Assignment 7/Automobile_data.csv')
```

[7]:
```
automobile_df.head()
```

[7]:
```
   index      company   body-style  wheel-base  length engine-type  \
0      0  alfa-romero  convertible        88.6   168.8        dohc   
1      1  alfa-romero  convertible        88.6   168.8        dohc   
2      2  alfa-romero    hatchback        94.5   171.2        ohcv   
3      3         audi        sedan        99.8   176.6         ohc   
4      4         audi        sedan        99.4   176.6         ohc   

  num-of-cylinders  horsepower  average-mileage    price  
0             four         111               21  13495.0  
1             four         111               21  16500.0  
2              six         154               19  16500.0  
3             four         102               24  13950.0  
4             five         115               18  17450.0  
```

[8]:
```
automobile_df.tail()
```

[8]:
```
    index     company body-style  wheel-base  length engine-type  \
56     81  volkswagen      sedan        97.3   171.7         ohc   
57     82  volkswagen      sedan        97.3   171.7         ohc   
58     86  volkswagen      sedan        97.3   171.7         ohc   
59     87       volvo      sedan       104.3   188.8         ohc   
60     88       volvo      wagon       104.3   188.8         ohc   

   num-of-cylinders  horsepower  average-mileage    price  
56             four          85               27   7975.0  
57             four          52               37   7995.0  
58             four         100               26   9995.0  
59             four         114               23  12940.0  
60             four         114               23  13415.0  
```

8.0.2 Simple Series

[9]:
```
s = pd.Series([1,2,3,4,5])
```

[10]:
```
s
```

[10]:
```
0    1
1    2
2    3
3    4
4    5
dtype: int64
```

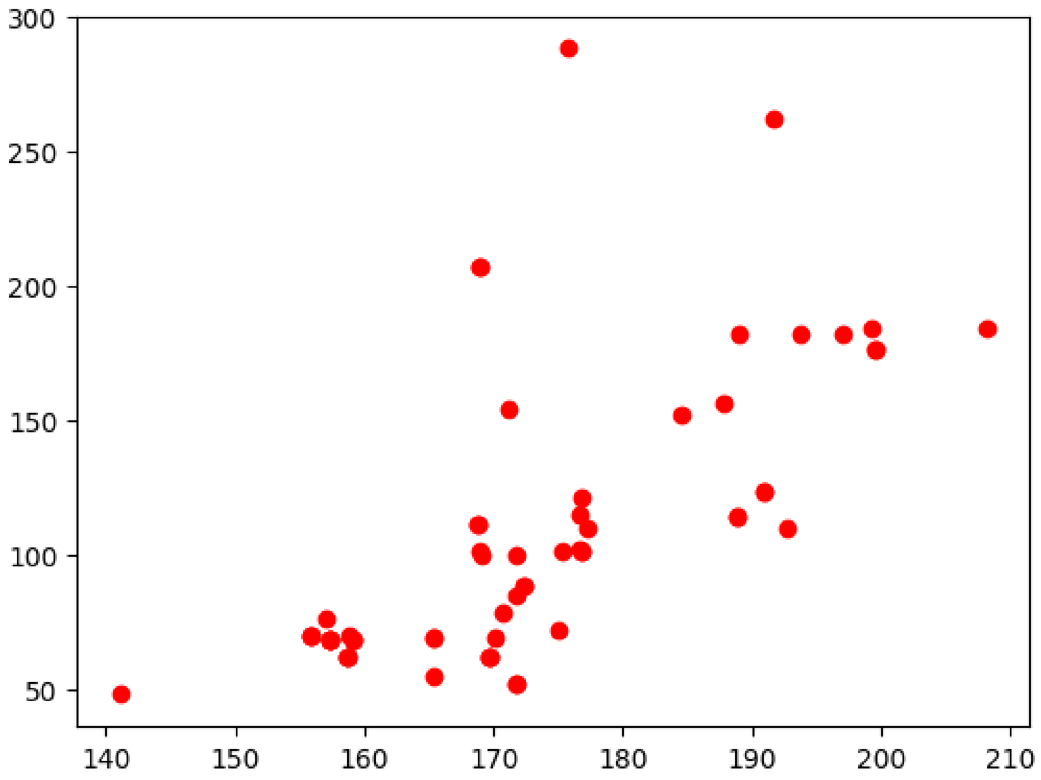
8.0.3 Slicing

[11]:
```
toyota_df['Price'][:10]
```

```
[11]: 0     13500
      1     13750
      2     13950
      3     14950
      4     13750
      5     12950
      6     16900
      7     18600
      8     21500
      9     12950
      Name: Price, dtype: int64
```

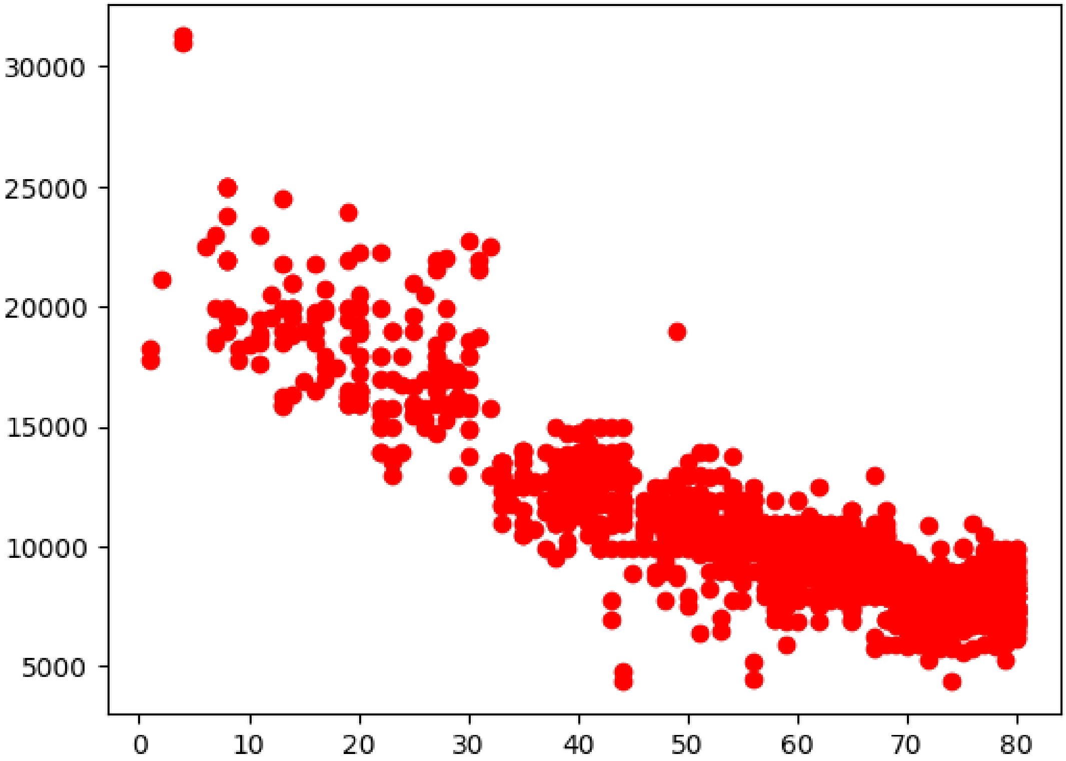
```
[12]: plt.plot(automobile_df['length'], automobile_df['horsepower'], 'ro')
```

```
[12]: [<matplotlib.lines.Line2D at 0x7f72240276d0>]
```



```
[13]: plt.plot(toyota_df['Age'], toyota_df['Price'], 'ro')
```

```
[13]: [<matplotlib.lines.Line2D at 0x7f7221f50bb0>]
```

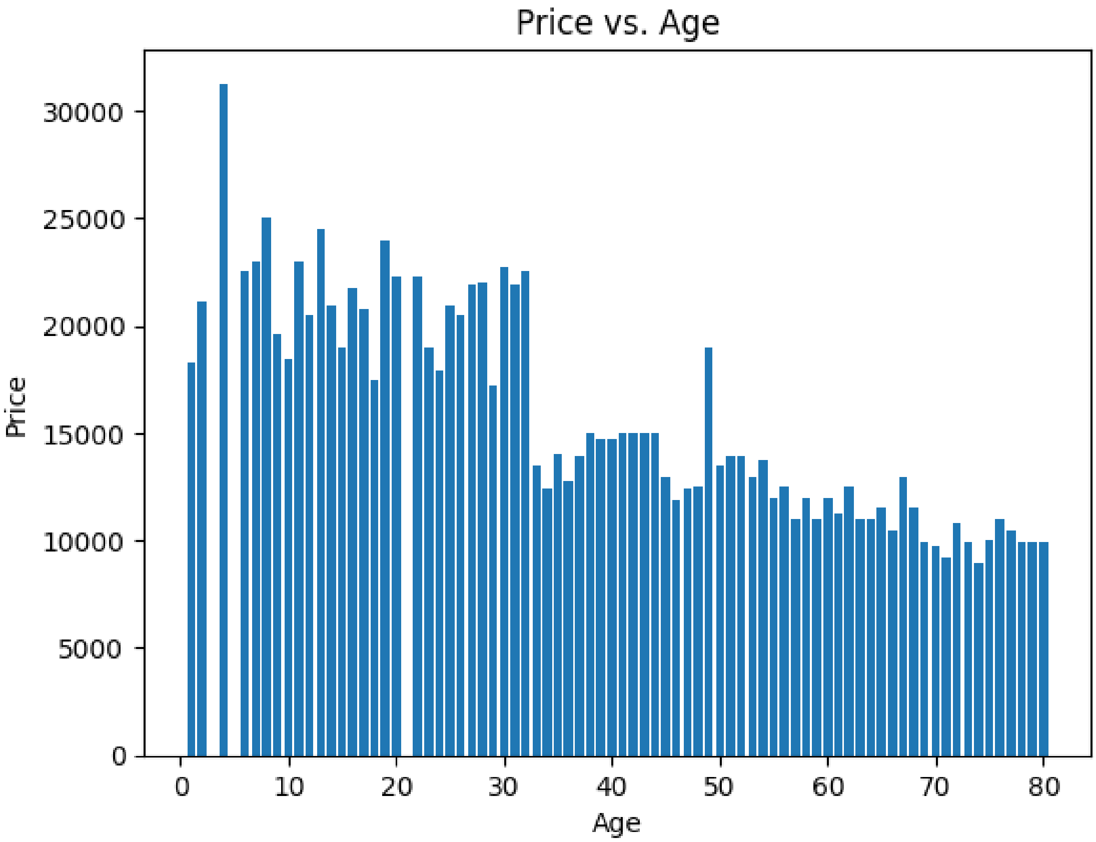


[14]:

```
plt.ylabel("Price")
plt.xlabel("Age")
plt.title("Price vs. Age")
plt.bar(toyota_df['Age'], toyota_df['Price'])
```

[14]:

```
<BarContainer object of 1436 artists>
```

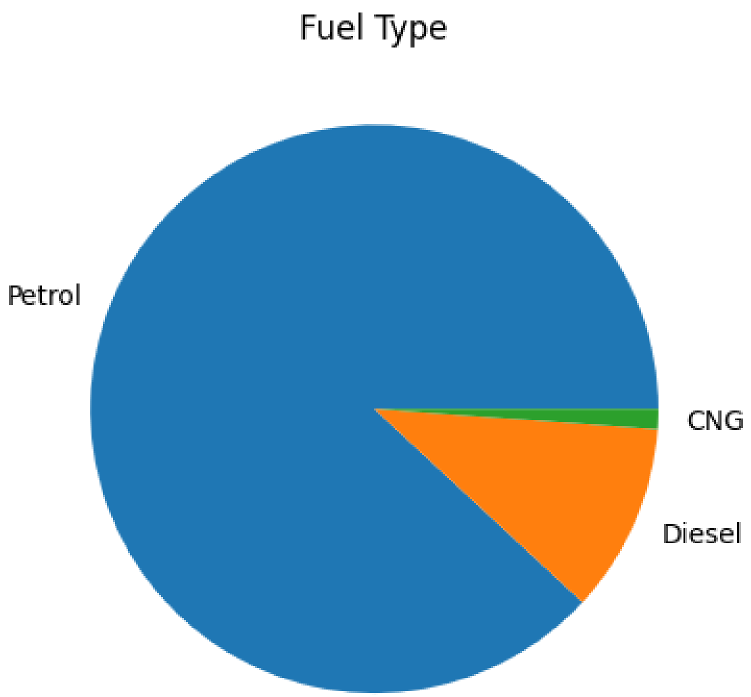


[15]:
```
fuel_type = pd.Series(toyota_df['FuelType'].values).value_counts()
print(fuel_type)
```

```
Petrol    1177
Diesel     144
CNG         15
dtype: int64
```

[16]:
```
plt.title('Fuel Type')
plt.pie(fuel_type, labels=fuel_type.index)
```

[16]:
```
([<matplotlib.patches.Wedge at 0x7f72212137c0>,
  <matplotlib.patches.Wedge at 0x7f72213d7670>,
  <matplotlib.patches.Wedge at 0x7f72212340d0>],
 [Text(-1.024006089442147, 0.40176053662026306, 'Petrol'),
  Text(1.0092010076630402, -0.4376223556125809, 'Diesel'),
  Text(1.0993157876137665, -0.038791740140453064, 'CNG')])
```



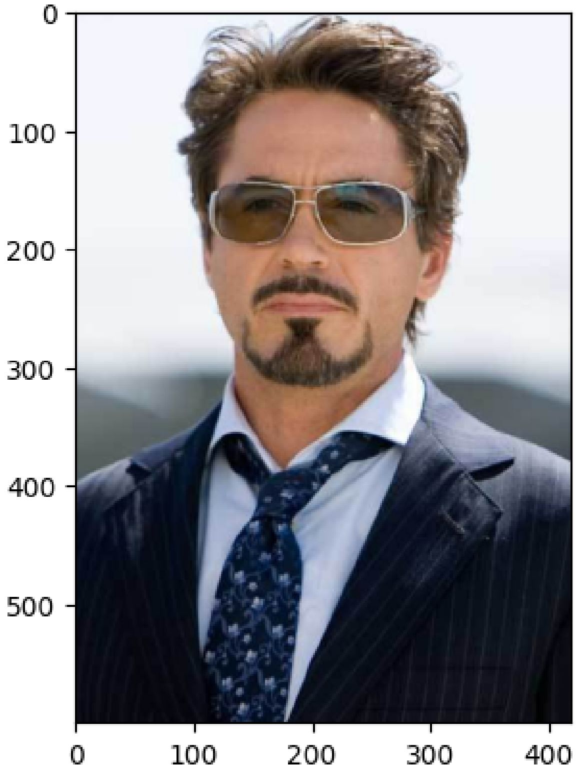
```
[19]: tony_image = plt.imread('tony.jpg')
```

```
[20]: tony_image
```

```
[20]: array([[[243, 243, 251],
              [243, 243, 251],
              [243, 243, 251],
              ...,
              [245, 245, 253],
              [245, 245, 253],
              [245, 245, 253]],

             [[243, 243, 251],
              [243, 243, 251],
              [243, 243, 251],
              ...,
              [245, 245, 253],
              [245, 245, 253],
              [245, 245, 253]],

             [[243, 243, 251],
              [243, 243, 251],
              [243, 243, 251],
              ...,
              [245, 245, 253],
              [245, 245, 253],
              [245, 245, 253]],

             ...,

             [[  9,   9,  17],
              [ 11,  11,  19],
              [ 14,  14,  22],
              ...,
              [ 15,  15,  23],
              [ 17,  17,  25],
              [ 20,  20,  28]],

             [[  9,   9,  17],
              [ 11,  11,  19],
              [ 14,  14,  22],
              ...,
              [ 15,  15,  23],
              [ 17,  17,  25],
              [ 20,  20,  28]],

             [[  9,   9,  17],
              [ 11,  11,  19],
              [ 14,  14,  22],
              ...,
              [ 15,  15,  23],
              [ 17,  17,  25],
              [ 20,  20,  28]]], dtype=uint8)
```

[21]:
```
plt.imshow(tony_image)
```

[21]:
```
<matplotlib.image.AxesImage at 0x7f72213461a0>
```



```
[ ]:
```

9 Conclusion

The Pandas was studied and understood. The functions of the Pandas library were also studied and implemented.

10 FAQ

1. 1. List out the key features of Panda Library?
   - Fast and efficient DataFrame object with default and customized indexing.
   - Tools for loading data into in-memory data objects from different file formats.
   - Data alignment and integrated handling of missing data.
   - Reshaping and pivoting of date sets.
   - Label-based slicing, indexing and subsetting of large data sets.
   - Columns from a data structure can be deleted or inserted.
   - Group by data for aggregation and transformations.
   - High performance merging and joining of data.
   - Time Series functionality.
2. 2. What are the different applications of Pandas.
   - Data Manipulation: Pandas provides a variety of operations to manipulate your data in the way you want. It provides a flexible way to access and manipulate your data.
   - Data Analysis: Pandas provide a fast and efficient way to conduct data analysis. It is the most preferred tool for data analysis in Python.
   - Data Cleaning: Pandas make it easy to clean messy data sets. It provides various functions to remove rows, columns, and change the data type of a DataFrame column.
   - Data Visualization: Pandas provide the visualization feature to plot your data in various forms. It provides a high-level interface for drawing attractive and informative graphics.
   - Machine Learning: Pandas is a popular library for machine learning in Python. It provides features for making machine learning easier and faster.
3. 3. List down the different types of graphs are supported by Matplot library.
   - Line Graphs.
   - Bar Charts.
   - Histogram.
   - Scatter Plot.
   - Area Plot.
   - Pie Chart.
   - Error Charts.
   - Power Spectra.
   - Contour Plot.
   - Spectrogram.