A Hands-on Introduction to Machine Learning

2. Python

CAMBRIDGE UNIVERSITY PRESS

What is Python?

A scripting language available on every platform

Easy to learn, easy to use, quite extensible and robust

Where is Python?

- Many UNIX systems have it pre-installed
- Free to download and install for all platforms
- Command-line:
    - $ python
- IDE (integrated development environment)
    - Eclipse
    - Jupyter notebook
    - Anaconda
    - Spyder

Basics of Python

- One-line examples
- Arithmetic operators
- Logical operators
- Data types
- Control structures
  - Condition checking with ‘if’ and ‘else’
  - While loop
  - For loop

Statistics with Python

- Storing a set of numbers
- Numpy library
- Doing descriptive analysis
- Visualization with bar graph

Statistical Essentials

- Statistical elements can be measured and manifested in Python

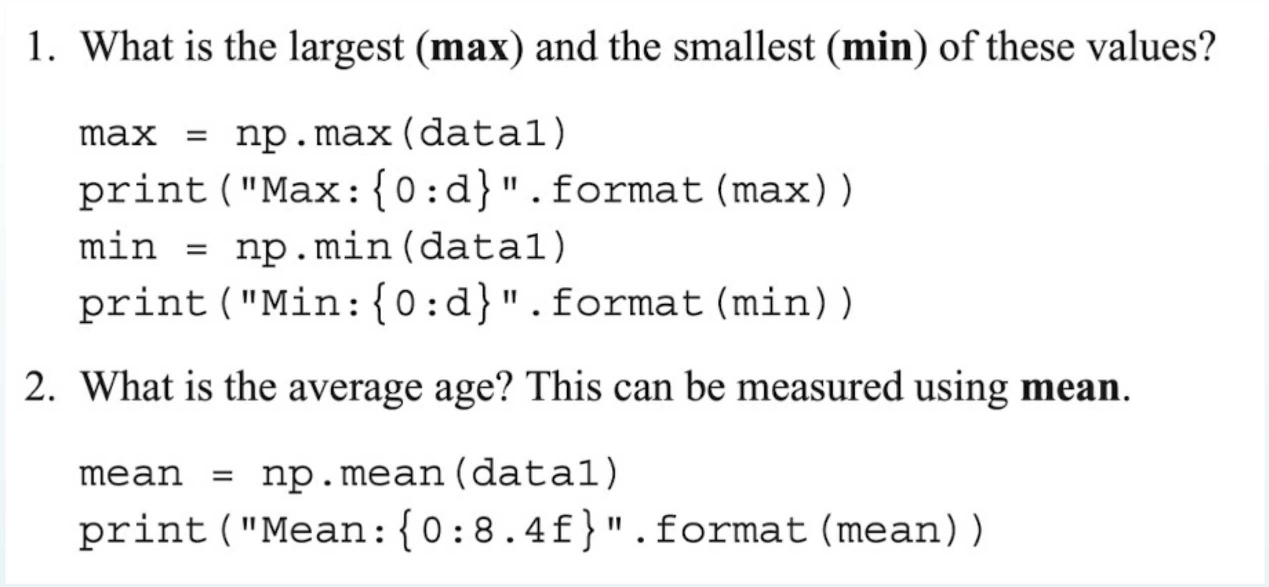
```
data1 =
[85,62,78,64,25,12,74,96,63,45,78,20,5,30,45,78,45,96,65,45,
74,12,78,23,8]
```

- To run this “dataset” we will use a very popular Python Library, NumPy

```
import numpy as np
```

- Primarily used for numerical computations and working with arrays and matrices.
- It provides efficient and optimized functions for mathematical operations, including linear algebra, statistics, etc.
- Widely used in scientific computing, data analysis, and machine learning tasks, NumPy serves as a foundational library for many other Python data science tools.

Statistical Essentials



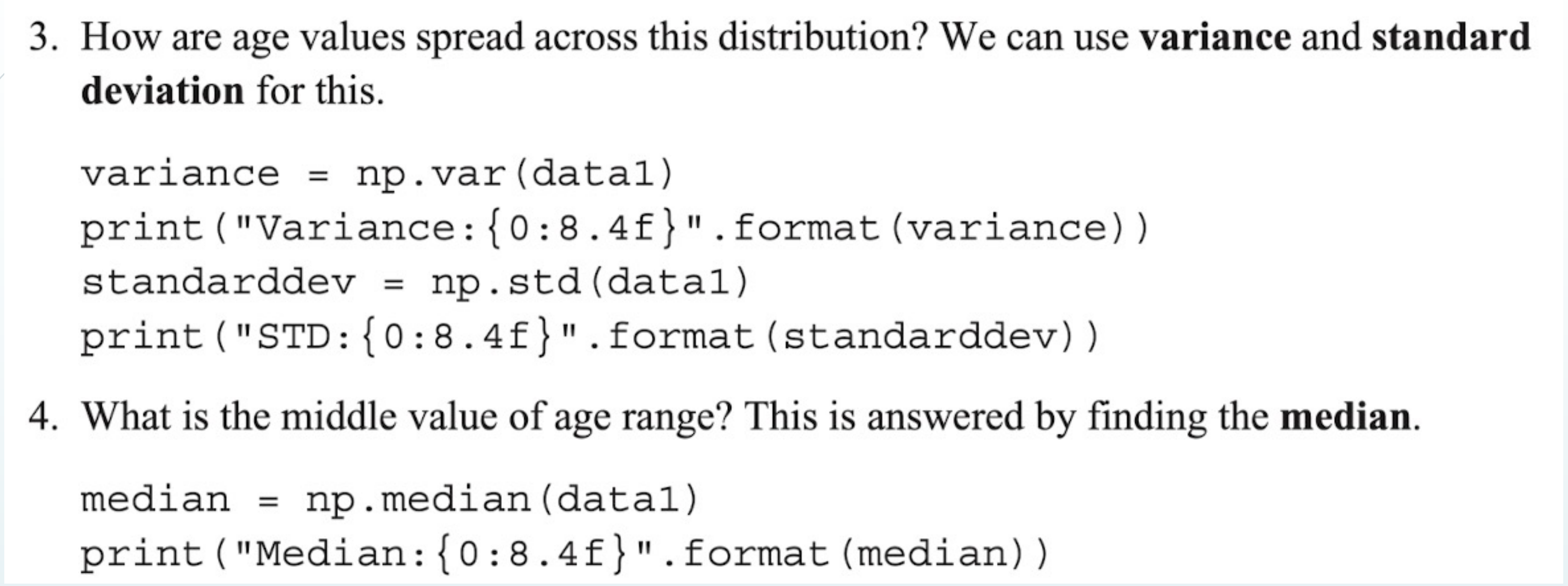
1. What is the largest (**max**) and the smallest (**min**) of these values?

```
max = np.max(data1)
print("Max:{0:d}".format(max))
min = np.min(data1)
print("Min:{0:d}".format(min))
```

2. What is the average age? This can be measured using **mean**.

```
mean = np.mean(data1)
print("Mean:{0:8.4f}".format(mean))
```

Statistical Essentials



3. How are age values spread across this distribution? We can use **variance** and **standard deviation** for this.

```
variance = np.var(data1)
print("Variance:{0:8.4f}".format(variance))
standarddev = np.std(data1)
print("STD:{0:8.4f}".format(standarddev))
```

4. What is the middle value of age range? This is answered by finding the **median**.

```
median = np.median(data1)
print("Median:{0:8.4f}".format(median))
```

Working with data

- Loading external data and using pandas library
- Plotting the data
- Correlation

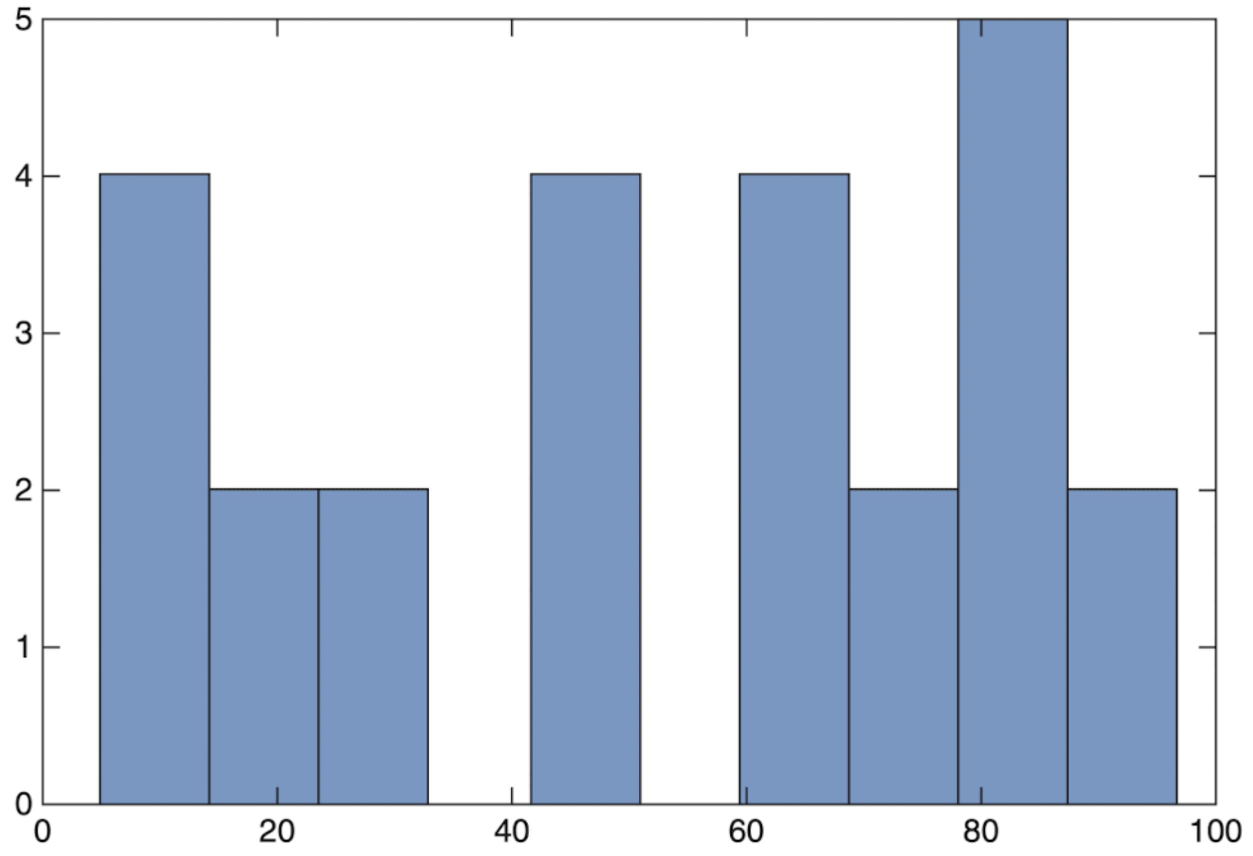
Plot Distribution

- To plot the whole distribution (a histogram) first we have to import the appropriate library:

```
import matplotlib.pyplot as plt
plt.figure()
hist1, edges1 = np.histogram(data1)
plt.bar(edges1[:-1], hist1, width = edges1[1:]-edges1[:-1])
```

- **Data Insight:** Reveals data structure, central tendencies, and variability.
- **Pattern Recognition**: Highlights trends, anomalies, and outliers in data.
- **Statistical Guidance**: Informs appropriate choice of statistical tests and models.
- **Effective Communication**: Simplifies complex data insights for broader understanding.

Plot Distribution

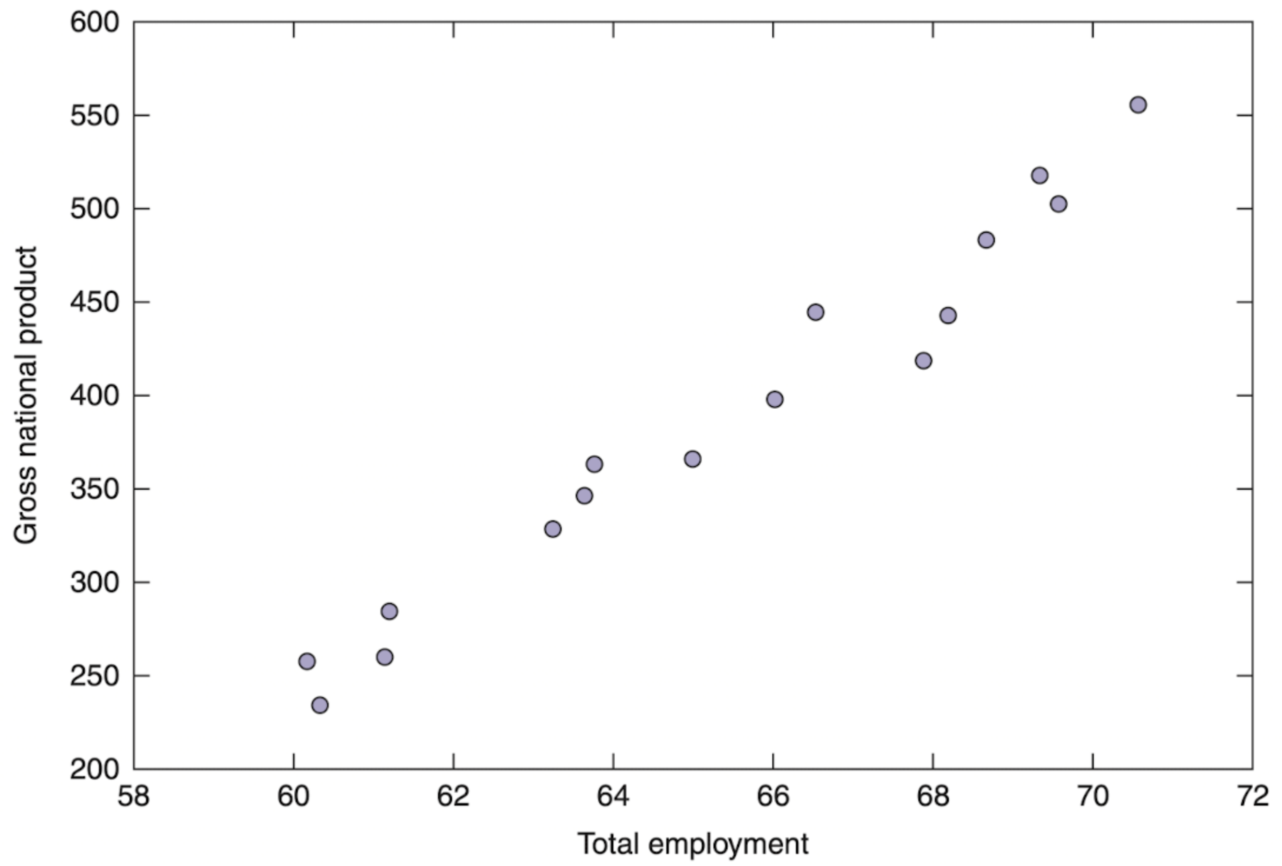


Scatterplot the Data

- Scatter plots in Python visually represent the relationship between two numerical variables, showing how one variable is affected by another.
- They help in identifying patterns, trends, and correlations between variables, or the absence thereof.
- Scatter plots are useful for spotting outliers or anomalies in data, which appear as points far from the general cluster.
- Provide a clear view of how data points are distributed across dimensions, offering insights into data density and spread.

Scatterplot the Data

```
import matplotlib.pyplot as plt
plt.scatter(df.Employed, df.GNP)
```



Correlation

- One of the most common tests we need to do while solving data-driven problems is to see if two variables are related.

- For this we can do a statistical test for **correlation**.

```
np.corrcoef(df.Employed,df.GNP)[0,1]
```

- The correlation coefficient value gives us an understanding of the strength and direction of the linear relationship between ‘Employed' and ‘GNP’.

- It's important to note that correlation does not imply causation. Even if two variables are correlated, it doesn't mean that one causes the other.

Correlation

- Correlation in Python quantifies the strength and direction of the relationship between two variables, indicating how one variable tends to change with the other.

- It's a fundamental statistical tool for understanding dependencies and testing hypotheses about relationships in data.

- Correlation analysis helps in feature selection for predictive modeling, identifying variables with significant mutual relationships.

- Aids in detecting multicollinearity in datasets, crucial for preprocessing in machine learning and data analysis workflows.

Summary

- Python: the most used language for doing data science
- Simple and easy to learn, yet versatile
- Most common tools for practicing Python: Jupyter, Anaconda
- Most useful libraries for data science: numpy, pandas, matplotlib, sklearn

- NEXT: **Class Activity Chapter 2**