Department of Artificial Intelligence and Data Science

Fundamentals of Data Science (AD45)

Academic Year : 2024-24, Batch 2023

Credits: 3:0:0

Text Books:

Joel Grus, “Data Science from Scratch”, 2nd Edition, O’Reilly Publications/Shroff Publishers and Distributors Pvt. Ltd., 2019. ISBN-13: 978- 9352138326

UNIT 1

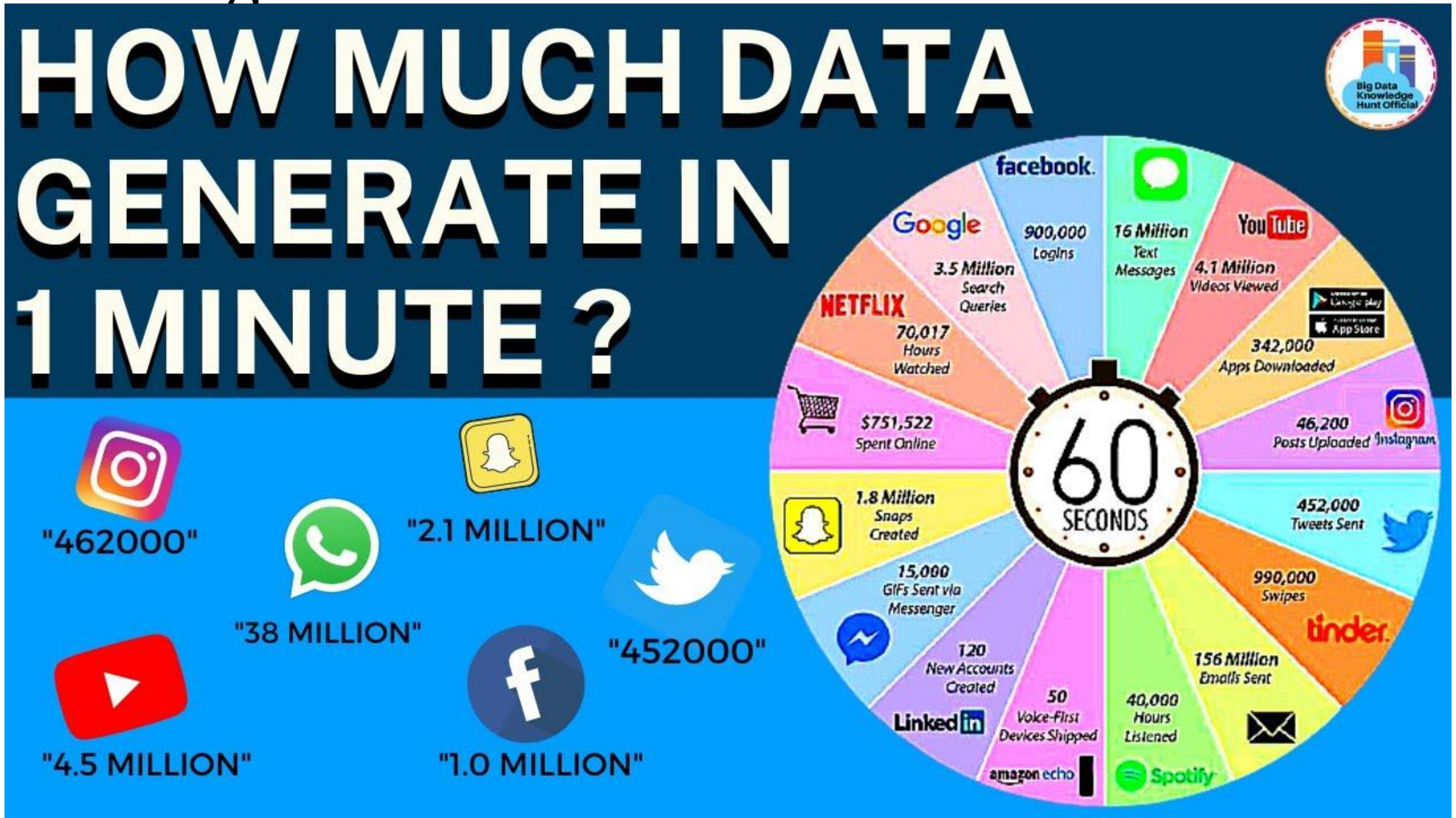
What is Data Science? Types of Data & Data Sources, Visualizing Data, matplotlib, Bar Charts, Line Charts, Scatterplots, Linear Algebra, Vectors, Matrices, Statistics, Describing a Single Set of Data, Correlation, Simpson’s Paradox, Some Other Correlational Caveats, Correlation and Causation. Probability, Dependence and Independence, Random Variables, Continuous Distributions, The Normal Distribution.

The Ascendance of Data

- We live in a world that’s drowning in data. Websites track every user’s every click.
- Your **smartphone** is building up a record of your location and speed every second of every day. “Quantified selfers” wear pedometers-on-steroids that are always recording their heart rates, movement habits, diet, and sleep patterns.
- **Smart cars** collect driving habits, smart homes collect living habits, and smart marketers collect purchasing habits.
- The **internet** itself represents a huge graph of knowledge that contains (among other things) an enormous cross-referenced encyclopedia; domain-specific databases about movies, music, sports results, pinball machines, memes, and cocktails; and too many government statistics (some of them nearly true!) from too many governments to wrap your head around.

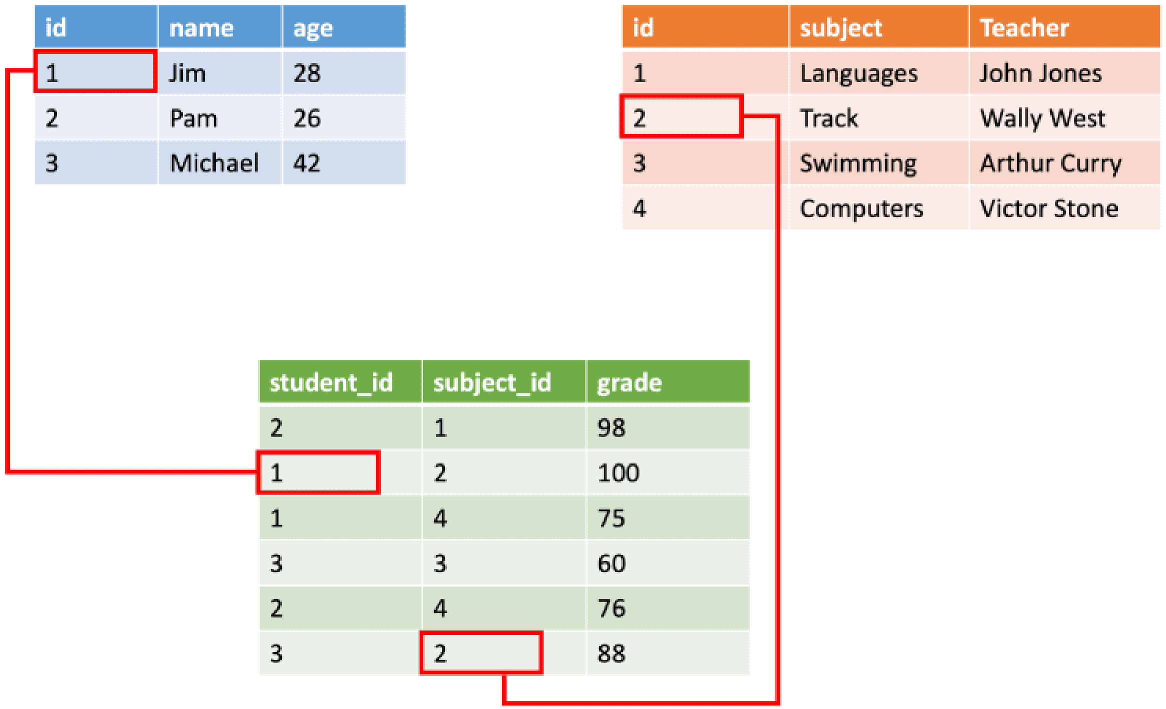
What Is Data Science?

- **What is Data?**
- **Data** is a collection of facts, numbers, measurements, observations. It can be raw or processed and is used to gain insights, make decisions, and drive technologies like AI and machine learning.



What Is Data Science?: Types of Data

- **Structured Data**
  - ❖ Organized and stored in a fixed format (e.g., databases, spreadsheets).
  - ❖ Examples:
    - ➢ Customer records (Name, Age, Email)
    - ➢ Sales data (Price, Quantity, Date)
    - ➢ SQL Databases



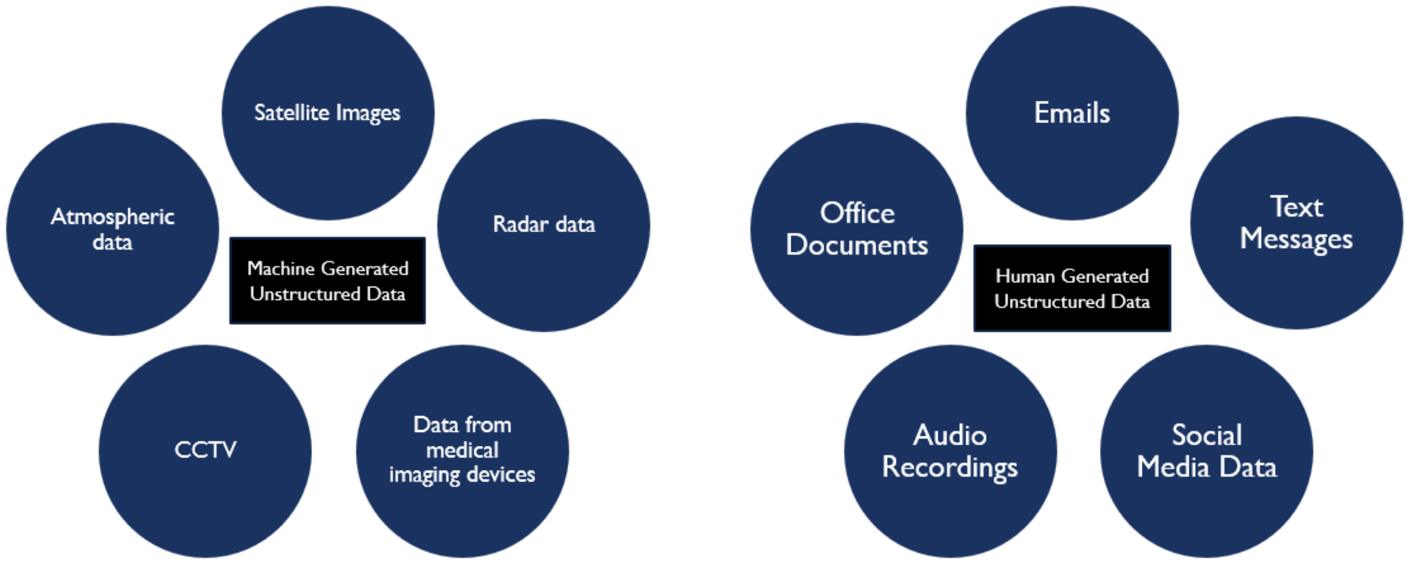
id	name	age
1	Jim	28
2	Pam	26
3	Michael	42

id	subject	Teacher
1	Languages	John Jones
2	Track	Wally West
3	Swimming	Arthur Curry
4	Computers	Victor Stone

student_id	subject_id	grade
2	1	98
1	2	100
1	4	75
3	3	60
2	4	76
3	2	88

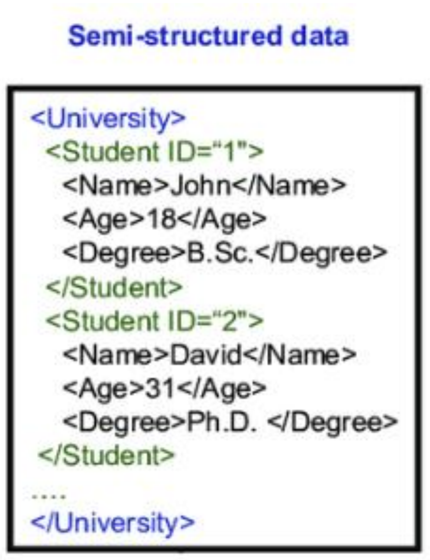
What Is Data Science?: Types of Data

- **Unstructured Data**
- No predefined format, making it harder to process.
- Examples:
  - Text documents, emails, social media posts
  - Images, videos, audio files
  - Sensor data, logs

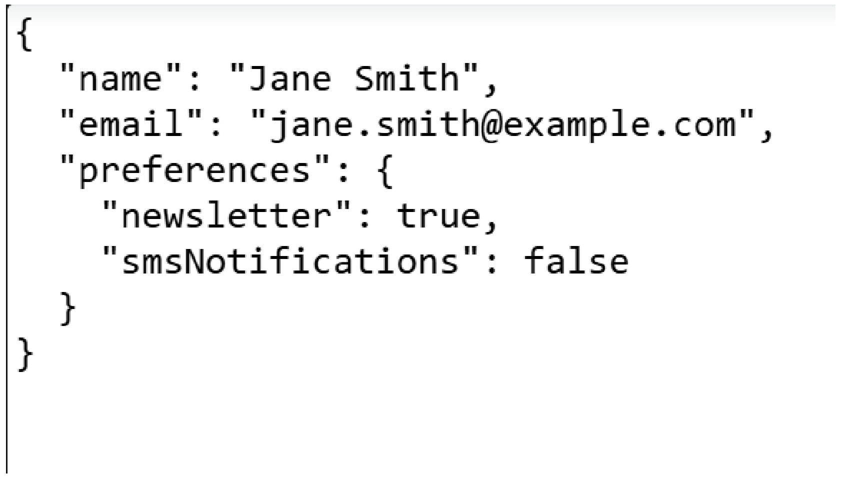


What Is Data Science?: Types of Data

- **Semi-Structured Data**
- Has some organization but doesn't fit traditional databases.
- Examples:
  - ❖ JSON, XML files
  - ❖ Emails with metadata (subject, timestamp)


Semi-structured data

```
<University>
  <Student ID="1">
    <Name>John</Name>
    <Age>18</Age>
    <Degree>B.Sc.</Degree>
  </Student>
  <Student ID="2">
    <Name>David</Name>
    <Age>31</Age>
    <Degree>Ph.D. </Degree>
 </Student>
....
</University>
```



```
{
  "name": "Jane Smith",
  "email": "jane.smith@example.com",
  "preferences": {
    "newsletter": true,
    "smsNotifications": false
  }
}
```

What Is Data Science?: Sources of Data

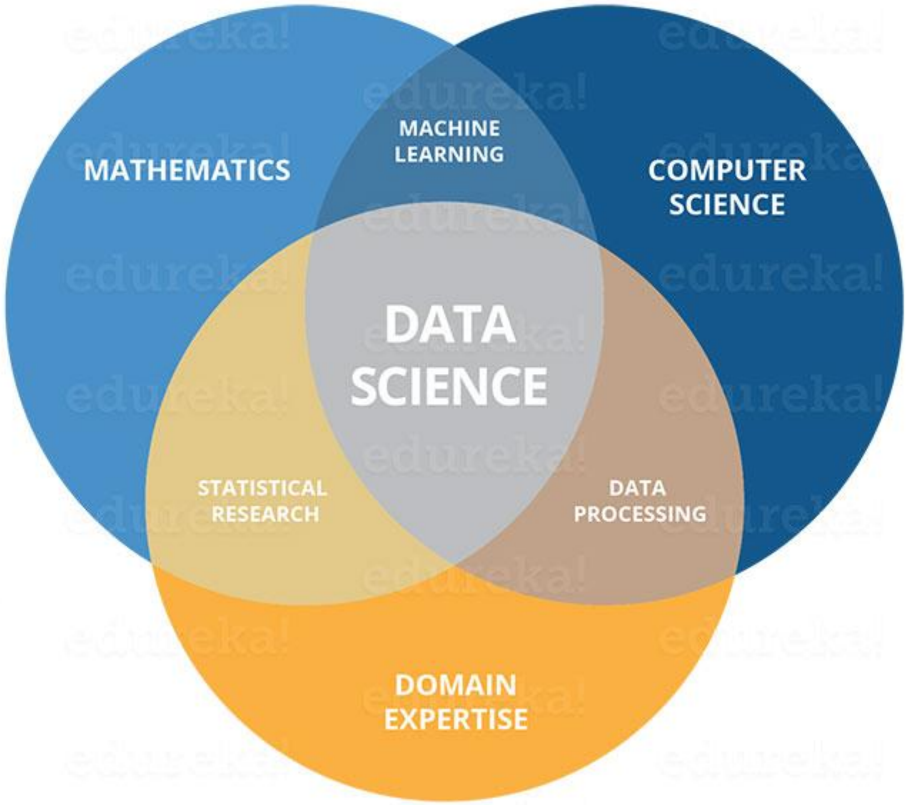
- **Digital Transactions** – Sales records, banking transactions.
- **Social Media** – Posts, likes, shares, and comments.
- **Sensors & IoT Devices** – Temperature sensors, smartwatches.
- **Healthcare Records** – Patient data, medical images.
- **Government & Research** – Census data, scientific experiments.

Why is Data Important?

- **Drives decision-making** – Businesses use data to improve operations.
- **Enables AI & Machine Learning** – Models learn from large datasets.
- **Improves efficiency** – Automates tasks and predicts trends.
- **Personalizes experiences** – Recommender systems (Netflix, Amazon).

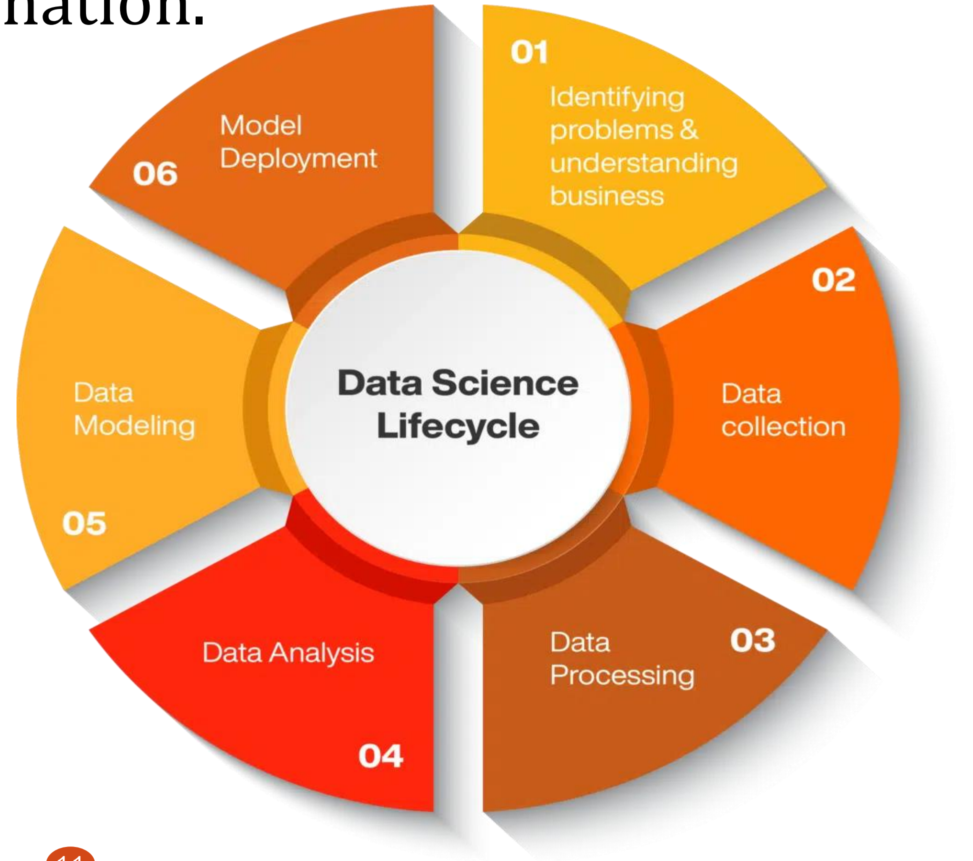
What Is Data Science?

- Data science is the **study** of data to **derive** meaningful insights for businesses. It's a multidisciplinary field that combines **statistics, math, computer engineering, and artificial intelligence** (AI).



Data Science Lifecycle

- This lifecycle encompasses **Six key stages**, that include—Identify problem, data collection, data pre-processing, data analysis, data modeling, and model deployment and dissemination.



Data Science Lifecycle

1. Identifying Problems & Understanding Business

This is the **foundation** of any data science project.

- **Objective:** Understand what the business wants to achieve.
- **Activities:**
  - Meet with stakeholders to gather requirements.
  - Define the problem clearly.
  - Translate business problems into data science problems.
  - Set project goals and success metrics.

Example: A retail company wants to reduce customer churn. The data scientist's job is to understand why customers leave and how to predict it.

2. Data Collection

Once the problem is defined, the next step is to **gather relevant data.**

- **Sources:** Databases, files, APIs, IoT devices, social media, etc.
- **Types:** Structured (tables), unstructured (images, text), or semi-structured (JSON, XML).
- **Goal:** Ensure the data is accurate, relevant, and sufficient for analysis.

Data Science Lifecycle

3. Data Processing

Also known as **Data Cleaning or Data Wrangling.**

- **Objective:** Prepare raw data for analysis.
- **Tasks:**
  - Handle missing or duplicate data.
  - Remove noise and outliers.
  - Convert data types, normalize/scale features.
  - Combine data from multiple sources (integration).

Clean data is **crucial** because poor-quality data leads to inaccurate models.

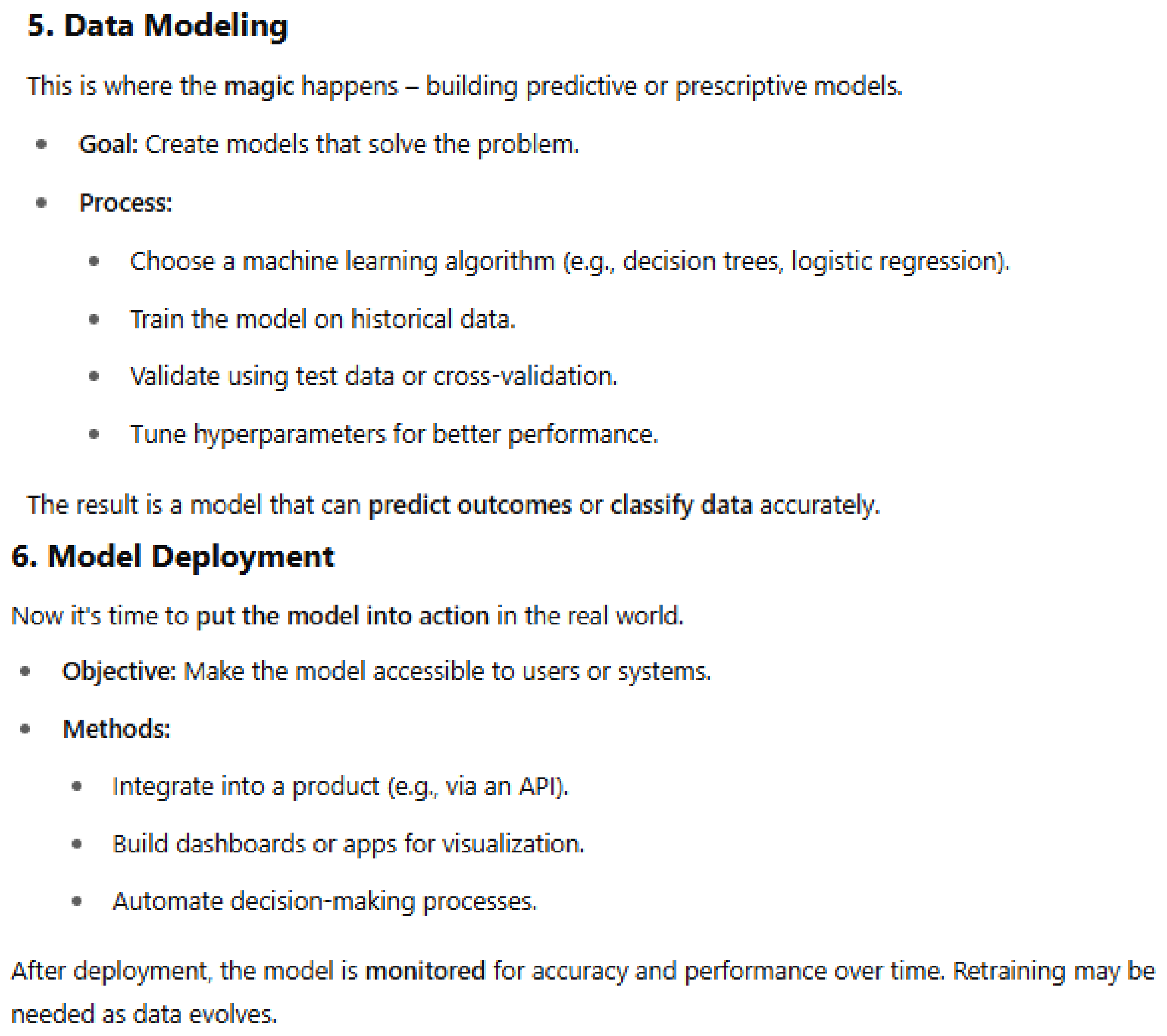
4. Data Analysis

This is the **exploratory phase** where you try to understand the patterns in the data.

- **Goal:** Gain insights and generate hypotheses.
- **Techniques:**
  - Descriptive statistics (mean, median, mode).
  - Data visualization (charts, graphs, heatmaps).
  - Correlation and trend analysis.

This helps identify **key features** and relationships that will feed into modeling.

Data Science Lifecycle



5. Data Modeling

This is where the **magic** happens – building predictive or prescriptive models.

- **Goal:** Create models that solve the problem.
- **Process:**
    - Choose a machine learning algorithm (e.g., decision trees, logistic regression).
    - Train the model on historical data.
    - Validate using test data or cross-validation.
    - Tune hyperparameters for better performance.

The result is a model that can **predict outcomes** or **classify data** accurately.

6. Model Deployment

Now it's time to **put the model into action** in the real world.

- **Objective:** Make the model accessible to users or systems.
- **Methods:**
    - Integrate into a product (e.g., via an API).
    - Build dashboards or apps for visualization.
    - Automate decision-making processes.

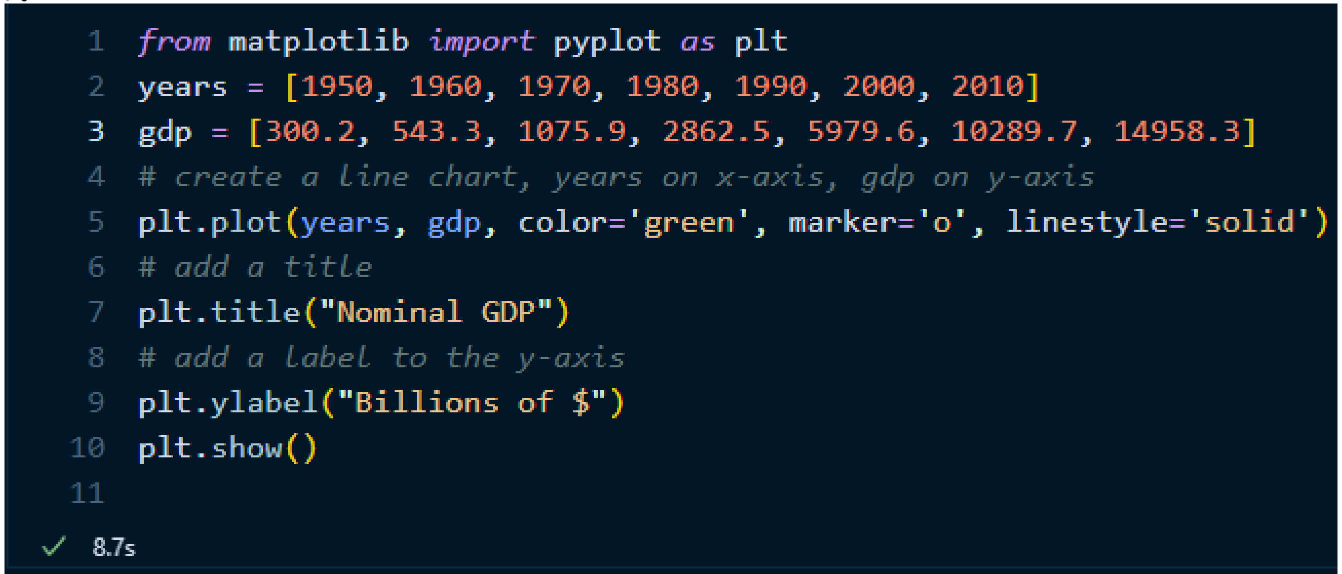
After deployment, the model is **monitored** for accuracy and performance over time. Retraining may be needed as data evolves.

Visualizing Data

- A fundamental part of the data scientist’s toolkit is data visualization.
- Data visualization is essential in today's world because it helps people understand complex data quickly and efficiently.
- Here are some key reasons why it is important:
- **Simplifies Complex Data**
- **Enhances Decision-Making**
- **Identifies Trends and Patterns**
- **Saves Time**

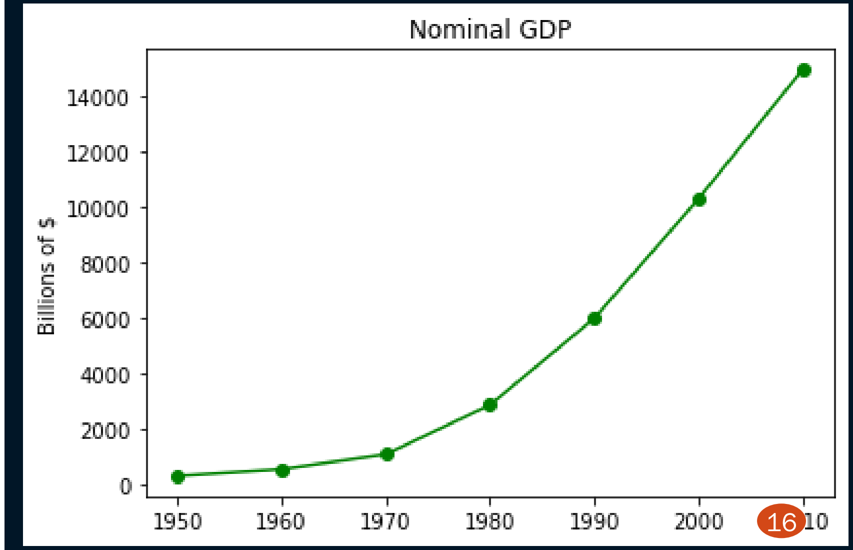
Visualizing Data

Python program to plot Line chart by assuming your own data.



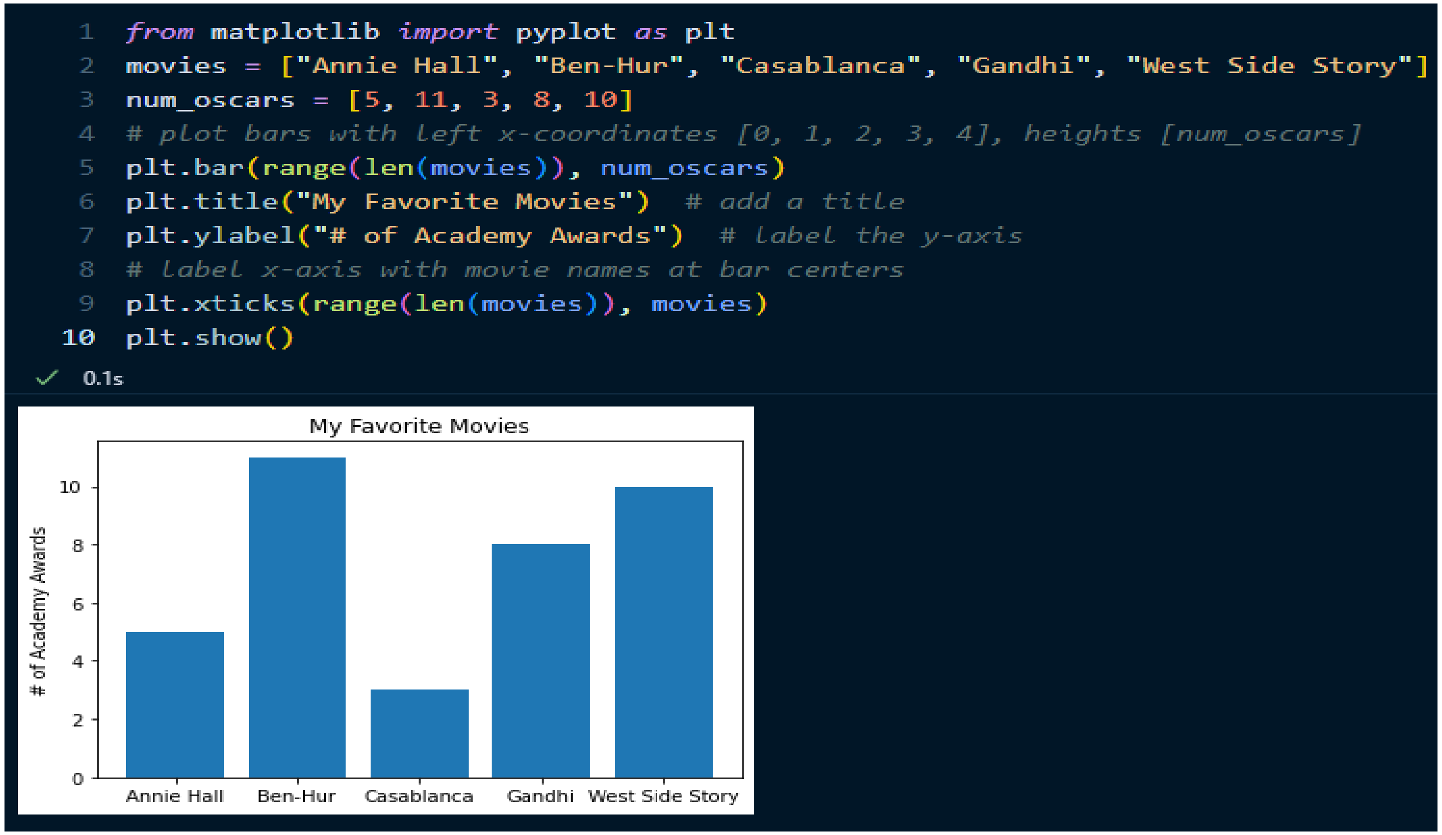
```python
from matplotlib import pyplot as plt
years = [1950, 1960, 1970, 1980, 1990, 2000, 2010]
gdp = [300.2, 543.3, 1075.9, 2862.5, 5979.6, 10289.7, 14958.3]
# create a line chart, years on x-axis, gdp on y-axis
plt.plot(years, gdp, color='green', marker='o', linestyle='solid')
# add a title
plt.title("Nominal GDP")
# add a label to the y-axis
plt.ylabel("Billions of $")
plt.show()

```

✓ 8.7s



Bar Charts:

- **Python program to plot bar chart by assuming your own data and explain the various attributes of bar chart.**
- A bar chart is a good choice when you want to show how some quantity varies among some discrete set of items.
- ***Syntax: plt.bar(x, height, width, bottom, align)***
- For instance, the below figure shows how many Academy Awards were won by each of a variety of movies.



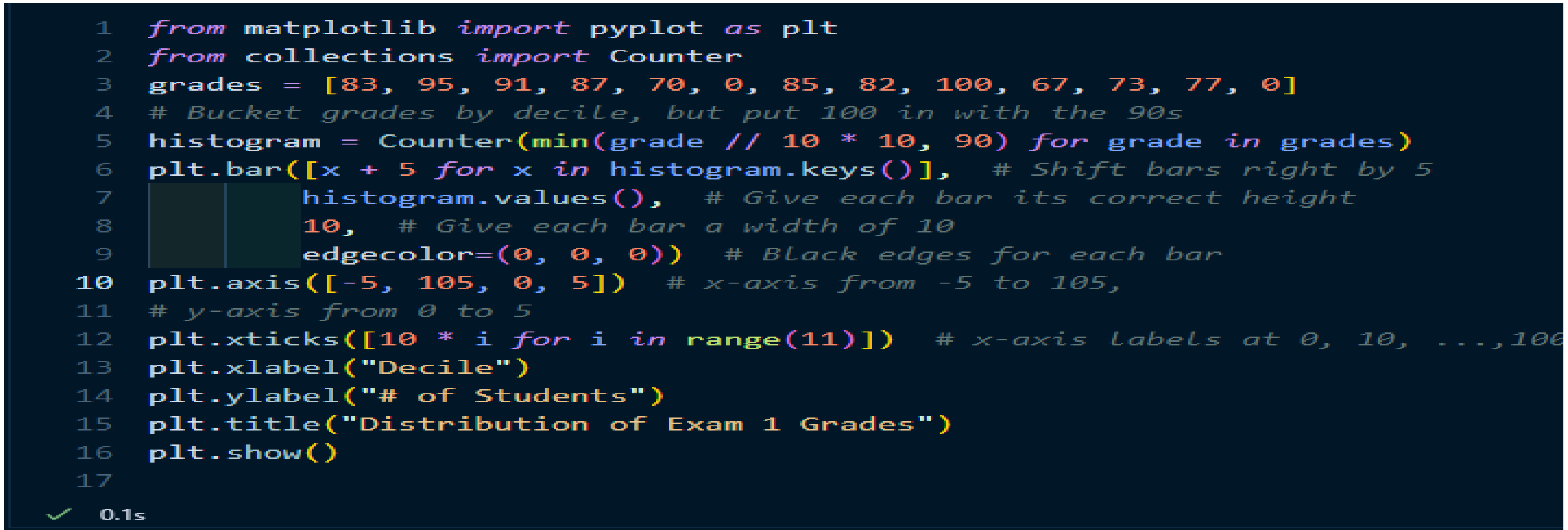
```python
from matplotlib import pyplot as plt
movies = ["Annie Hall", "Ben-Hur", "Casablanca", "Gandhi", "West Side Story"]
num_oscars = [5, 11, 3, 8, 10]
# plot bars with left x-coordinates [0, 1, 2, 3, 4], heights [num_oscars]
plt.bar(range(len(movies)), num_oscars)
plt.title("My Favorite Movies")  # add a title
plt.ylabel("# of Academy Awards")  # label the y-axis
# label x-axis with movie names at bar centers
plt.xticks(range(len(movies)), movies)
plt.show()
```

✓ 0.1s

- a

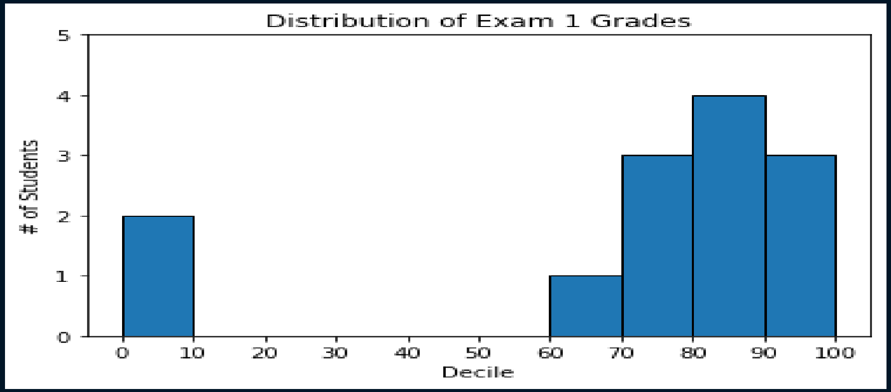
Bar Charts: Histograms

- A bar chart can also be a good choice for plotting histograms of bucketed numeric values, as in the below figure, in order to visually explore how the values are distributed.
- **Python program to plot Histogram by assuming your own data**



```python
from matplotlib import pyplot as plt
from collections import Counter
grades = [83, 95, 91, 87, 70, 0, 85, 82, 100, 67, 73, 77, 0]
# Bucket grades by decile, but put 100 in with the 90s
histogram = Counter(min(grade // 10 * 10, 90) for grade in grades)
plt.bar([x + 5 for x in histogram.keys()],  # Shift bars right by 5
        histogram.values(),  # Give each bar its correct height
        10,  # Give each bar a width of 10
        edgecolor=(0, 0, 0))  # Black edges for each bar
plt.axis([-5, 105, 0, 5])  # x-axis from -5 to 105,
# y-axis from 0 to 5
plt.xticks([10 * i for i in range(11)])  # x-axis labels at 0, 10, ...,10
plt.xlabel("Decile")
plt.ylabel("# of Students")
plt.title("Distribution of Exam 1 Grades")
plt.show()

```

```
✓ 0.1s
```

- 

Bar Charts

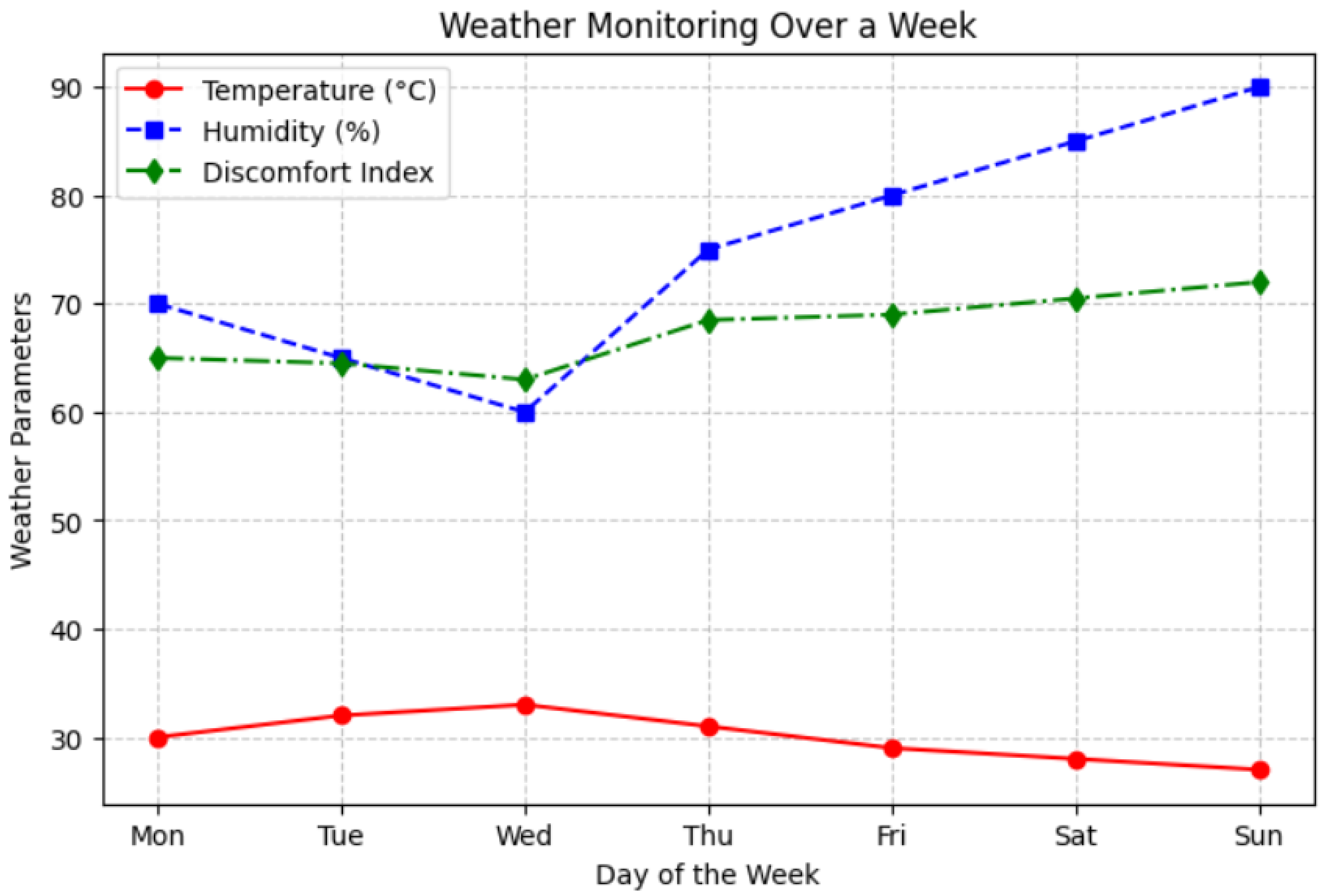
- The third argument to ***plt.bar*** specifies the bar width. Here we chose a width of 10, to fill the entire decile.
- We also shifted the bars right by 5, so that, for example, the "10" bar (which corresponds to the decile 10–20) would have its center at 15 and hence occupy the correct range.
- We also added a black edge to each bar to make them visually distinct.
- The call to ***plt.axis*** indicates that we want the x-axis to range from –5 to 105 (just to leave a little space on the left and right), and that the y-axis should range from 0 to 5.
- And the call to ***plt.xticks*** puts x-axis labels at 0, 10, 20, ..., 100.

Multiple Line Charts: using plot()

Python program to plot Multiple Line Charts by assuming your own data

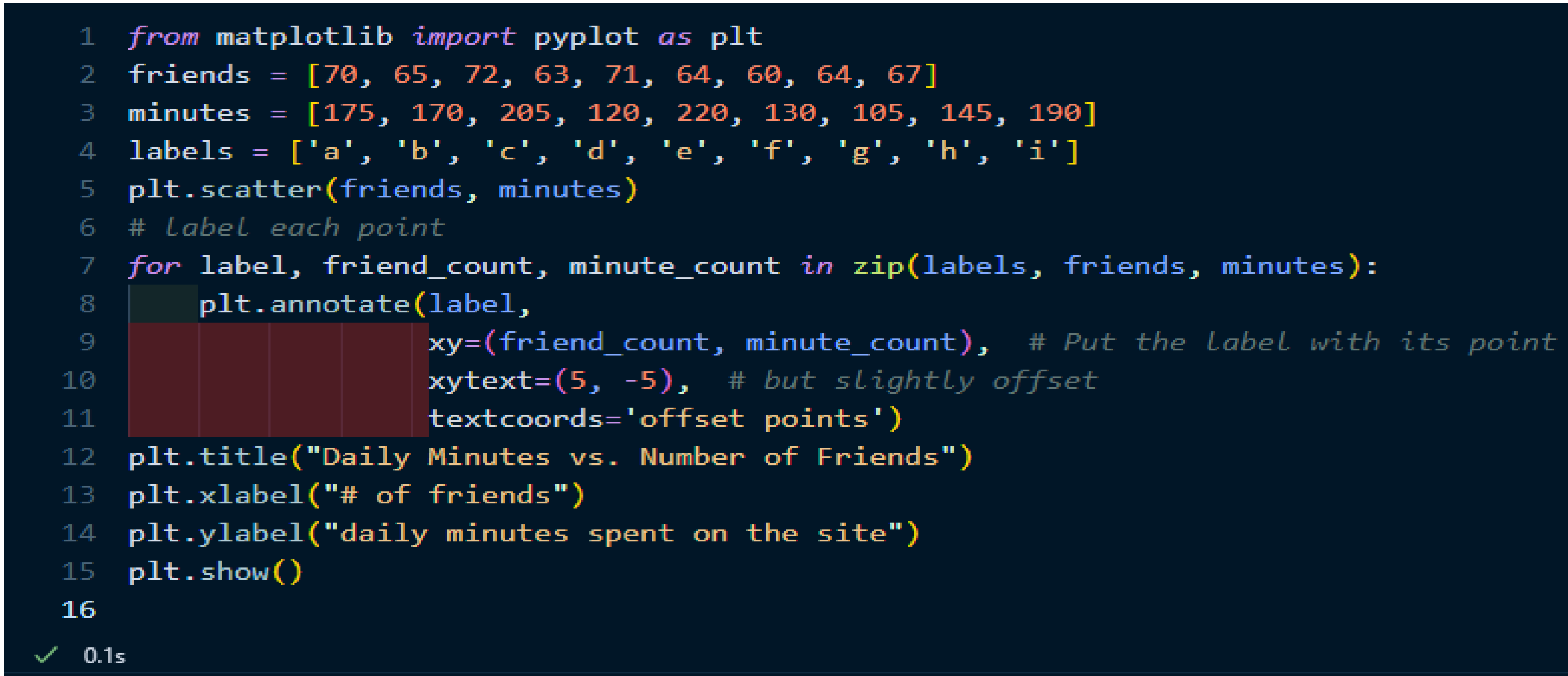
```
import matplotlib.pyplot as plt
# Sample data: Days of the week
days = ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"]
temperature = [30, 32, 33, 31, 29, 28, 27] # Temperature (°C)
humidity = [70, 65, 60, 75, 80, 85, 90] # Humidity (%)
# Discomfort Index (A rough measure of heat discomfort:Temp + 0.5 *
Humidity)
discomfort_index = [t + 0.5 * h for t, h in zip(temperature, humidity)]
plt.figure(figsize=(8, 5)) # Create the line plot
plt.plot(days, temperature, 'r-o', label="Temperature (°C)")  # Red solid line with circles
plt.plot(days, humidity, 'b--s', label="Humidity (%)")  # Blue dashed line with squares
plt.plot(days, discomfort_index, 'g-.d', label="Discomfort Index")  # Green dot-dashed line
with diamonds
plt.xlabel("Day of the Week")
plt.ylabel("Weather Parameters")
plt.title("Weather Monitoring Over a Week")
plt.legend() # Adding legend
plt.grid(True, linestyle="--",) # Grid for better readability
```

Multiple Line Charts



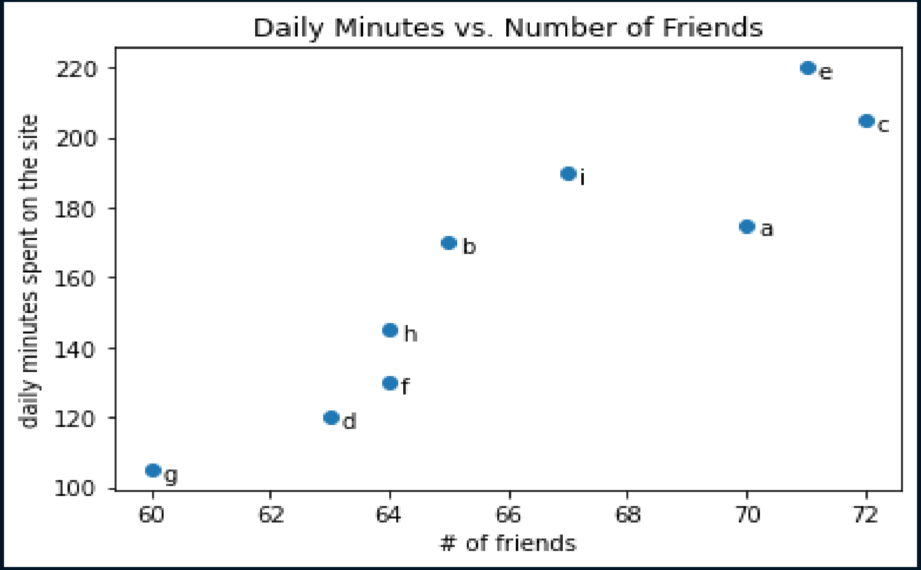
Scatterplots

- A scatterplot is the right choice for visualizing the relationship between two paired sets of data.
- For example, the below figure illustrates the relationship between the number of friends your users have and the number of minutes they spend on the site every day:
- **Python program to plot scatterplots by assuming your own data**



```python
from matplotlib import pyplot as plt
friends = [70, 65, 72, 63, 71, 64, 60, 64, 67]
minutes = [175, 170, 205, 120, 220, 130, 105, 145, 190]
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i']
plt.scatter(friends, minutes)
# label each point
for label, friend_count, minute_count in zip(labels, friends, minutes):
    plt.annotate(label,
                 xy=(friend_count, minute_count),  # Put the label with its point
                 xytext=(5, -5),  # but slightly offset
                 textcoords='offset points')
plt.title("Daily Minutes vs. Number of Friends")
plt.xlabel("# of friends")
plt.ylabel("daily minutes spent on the site")
plt.show()

```

✓ 0.1s



Linear Algebra: Vectors

- Linear algebra is the branch of mathematics that deals with vector spaces.

Physical quantity with magnitude and no direction is known as a **scalar** quantity and a physical quantity with magnitude and directions is known as a **vector** quantity.

Vectors are objects that can be added together to form new vectors and that can be multiplied by scalars (i.e., numbers), also to form new vectors

Vectors, are often a useful way to represent numeric data.

For example, if you have the heights, weights, and ages of a large number of people, you can treat your data as **three-dimensional vectors** [height, weight, age].

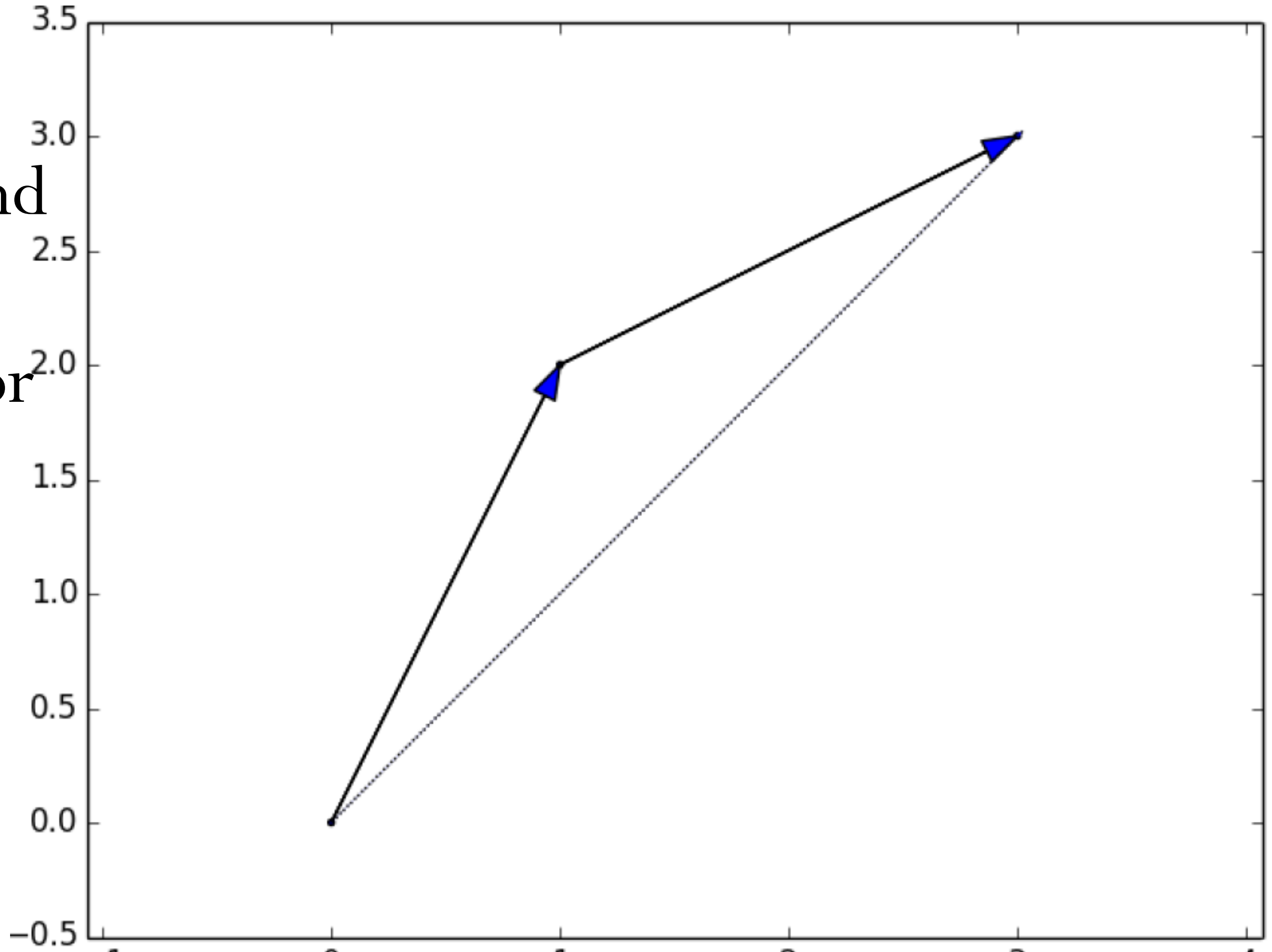
```
height_weight_age = [70, # inches,
                     170, # pounds,
                     40 ] # years
```

Vectors

- **Perform arithmetic on vectors.**
- Because Python lists aren't vectors (and hence provide no facilities for vector arithmetic), we'll need to build these arithmetic tools ourselves.
- Add two vectors. Vectors add component wise.
- This means that if two vectors **v and w** are the same length, their sum is just the vector whose first element is v[0] + w[0], whose second element is v[1] + w[1], and so on.
- (If they're not the same length, then we're not allowed to add them.)

Adding two Vectors

For example, adding the vectors [1, 2] and [2, 1] results in [1 + 2, 2 + 1] or [3, 3], as shown in Figure 4-1.



Adding two vectors

Add two Vectors

```
def vector_add(v, w):
  """adds corresponding elements"""
  return [v_i + w_i   for v_i, w_i in zip(v, w)]
print(vector_add([5, 7, 9], [4, 5, 6]))
```

To subtract two vectors we just subtract the corresponding elements:

```
def vector_subtract(v, w):
  """subtracts corresponding elements"""
  return [v_i - w_i   for v_i, w_i in zip(v, w)]
print(vector_subtract([5, 7, 9], [4, 5, 6]))
```

Component wise sum a list of Vectors

- We'll also sometimes want to component wise sum a list of vectors—that is, create a new vector whose first element is the sum of all the first elements, whose second element is the sum of all the second elements, and so on:

```
def vector_sum(vectors):
    """sums all corresponding elements"""
    result = vectors[0] # start with the first vector
    for vector in vectors[1:]: #loop over the others
        result = vector_add(result, vector) #add to result
        return result
print(vector_sum([[1, 2], [3, 4], [5, 6], [7, 8]]))
```

Multiply a vector by a scalar

- We'll also need to be able to multiply a vector by a scalar, which we do simply by multiplying each element of the vector by that number:

```
def scalar_multiply(c, v):
    """c is a number, v is a vector"""
    return [c * v_i for v_i in v]
print(scalar_multiply(2, [1, 2, 3]))
```

Compute Component wise Means

```
def vector_sum(vectors):
  """sums all corresponding elements"""
  result = vectors[0] # start with the first vector
  for vector in vectors[1:]: #loop over the others
    result = vector_add(result, vector) #add to result
    return result
def scalar_multiply(c, v):
  """c is a number, v is a vector"""
  return [c * v_i for v_i in v]
def vector_mean(vectors):
  """compute the vector whose ith element is the mean of the
  ith elements of the input vectors"""
  n = len(vectors)
  return scalar_multiply(1/n, vector_sum(vectors))

print(vector_mean([[1, 2], [3, 4], [5, 6]]))
```

```
[1.3333333333333333, 2.0]
```