INFO 4000
Informatics III

Data Science Specialization - Advanced

Week1 Intro and overview

Course Instructor

Jagannath Rao

raoj@uga.edu

Course Details (not in any sequence)

What will you learn - 1

1. Developing Application Programming interfaces (API's) to:
   I. *Do data integration of different sources of data*
   II. *Work with SQL databases*
   III. *Create datasets from sources on the internet and other sources*
   IV. *Deploy Classical ML and Deep Learning models*

2. Transfer Learning – Working with saved, pre-trained Deep learning models
   I. *Data augmentation concepts in image analytics*
   II. *Vision pretrained models – CNN training and fine tuning*
   III. *Transformer based models from PyTorch and Hugging face – Vision and NLP*
   IV. *Application development with pretrained models*

What will you learn - 2

3. LLMs – Application development
   - I. *A Real-world application using LLMs like -*
     - *Traffic flow insights gathering or*
     - *Customer Service chatbot .*

4. Analytics when Audio is the data
   - I. *Audio basics, feature extraction from audio data and analyzing audio data*
   - II. *Applications in audio analytics using ML / DL – Sound recognition or Music Classification*

What will you learn - 3

5. Communication techniques
    I. *MQTT - Understanding how it works*
    II. *Building applications which can use MQTT*

6. Independent work
    I. *Exercises to experiment with concepts learned*
    II. *Developing applications from scratch – Assignments*

7. Research work (INFO6000 students only)
    I. *To be picked from related topics in the course in coordination with the instructor,*

What I expect you to be very familiar with!

Foundation

1. INFO 2000 and 3000
2. Programming in Python – Intermediate level or more and familiarity with key libraries.
3. Ability to do Exploratory Data Analysis using Python.
4. Machine Learning and Deep Learning – Algorithms, developing models, testing and applying them using Sklearn and PyTorch.
5. Good grip over Image and Text analytics using CNNs and NLP.

In this course we will use Python version 3.10 or greater installed with Miniconda (preferred) or Ananconda distribution. Please make sure these are installed on your laptops.

Projects, assignments and exercises

1. During the course you will have **2 Homework assignments, 3 Mini Projects and a test** to complete. In addition, there will be practice exercises to do each week to master your understanding of basic concepts.

2. Grading:
   - a. Test (1x): 25%
   - b. HW assignments (2x): 20%
   - c. Mini Projects (3x) : 40%
   - d. Completing practice exercises of the week and: 15%
   - e. All grading has the requirement of submitting 'Reflections'

3. Grading Scheme:

>= 90%	A
>= 85% & < 90%	B+
>= 80% & < 85%	B
>= 75% & < 80%	C
>= 70% & < 75%	C-
>= 60% & <70%	D
Below that is	F

Grading criteria that will be followed

All assignments in this course will be about developing applications by implementing code and answering questions associated with the assignment:

1. Criteria 1 – Instructions given have all been implemented (40%)
2. Criteria 2 – Code runs without errors (20%)
3. Criteria 3 – Correctness of solution and that code is well structured, commented, includes citations and is easy to follow (25%)
4. Criteria 4 – Questions asked about the concepts are all answered (15%)
5. Criteria 5 - Work is original and there is no plagiarism of any sort (will be penalized) – Honesty rule will be applied.
6. The test will be graded purely on correct and wrong answers.
7. Unless permission was granted for a late submission (rare), 5% per day penalty for late submissions beyond a day will be applied and submissions beyond 5 days late will not be evaluated.

Honesty Code

UGA Student Honor Code: "I will be academically honest in all of my academic work and will not tolerate academic dishonesty of others." *A Culture of Honesty*, the University's policy and procedures for handling cases of suspected dishonesty, can be found at www.uga.edu/ovpi.

A special note regarding the "honesty policy" in the context of usage GenAI platforms like ChatGPT, Gemini, Co-Pilot, etc *– for this course, while you are free to use these platforms for learning & understanding concepts in a deeper way, the use of these platforms to generate code for solving complete assignments given to you is prohibited. Use of tools provided by the University to detect such plagiarism is not beyond this course's bounds.* ***If you use any of the GenAI tools for understanding & developing snippets of code, you will provide the citations for them, using a standard citation method, in a Markdown cell in the Jupyter Notebook or using comments in the VSCode editor. The citation shall include the tool used and the prompt for the specific code snippet that was generated. Not adhering to this when using GenAI tools will be a violation of the honesty policy.***

What are ‘Reflections’?

Every assignment given has a requirement for submitting reflections after the solution has been provided..

The reflections (very similar to what you may have done in earlier INFO courses) and here is the guideline:

After comparing the provided solution to your submission, identify 3 places where the provided solution differed from your own AND where that difference is due to something that you didn't understand correctly (about the problem, or the concept). Explain the issue and what you learned from it.

Briefly, in a few sentences for each:

1.
2.
3.

Grading will be done based on the assignment and reflection submission as one package.

Logistics and Tips

1. All course material and assignment submissions will / shall happen on eLC.

2. Grading (except final grading) and feedback will be on eLC.

3. Announcements of any type will be on eLC.

4. What students need to pay attention to have a successful semester
   Tip 1: Submit all work on time and within the dead-lines specified.
   Tip 2: Make sure your work and submission is complete in all respects – read instructions carefully.
   Tip 3: Show through your work that you have understood the concepts and take the time to write a paragraph
   or two on your findings / observations / assumptions / approach.
   Tip 4: Make the instructor's grading working easier.

6. It is a very intense course and packed with concepts and so please make sure you are diligent in keeping pace.

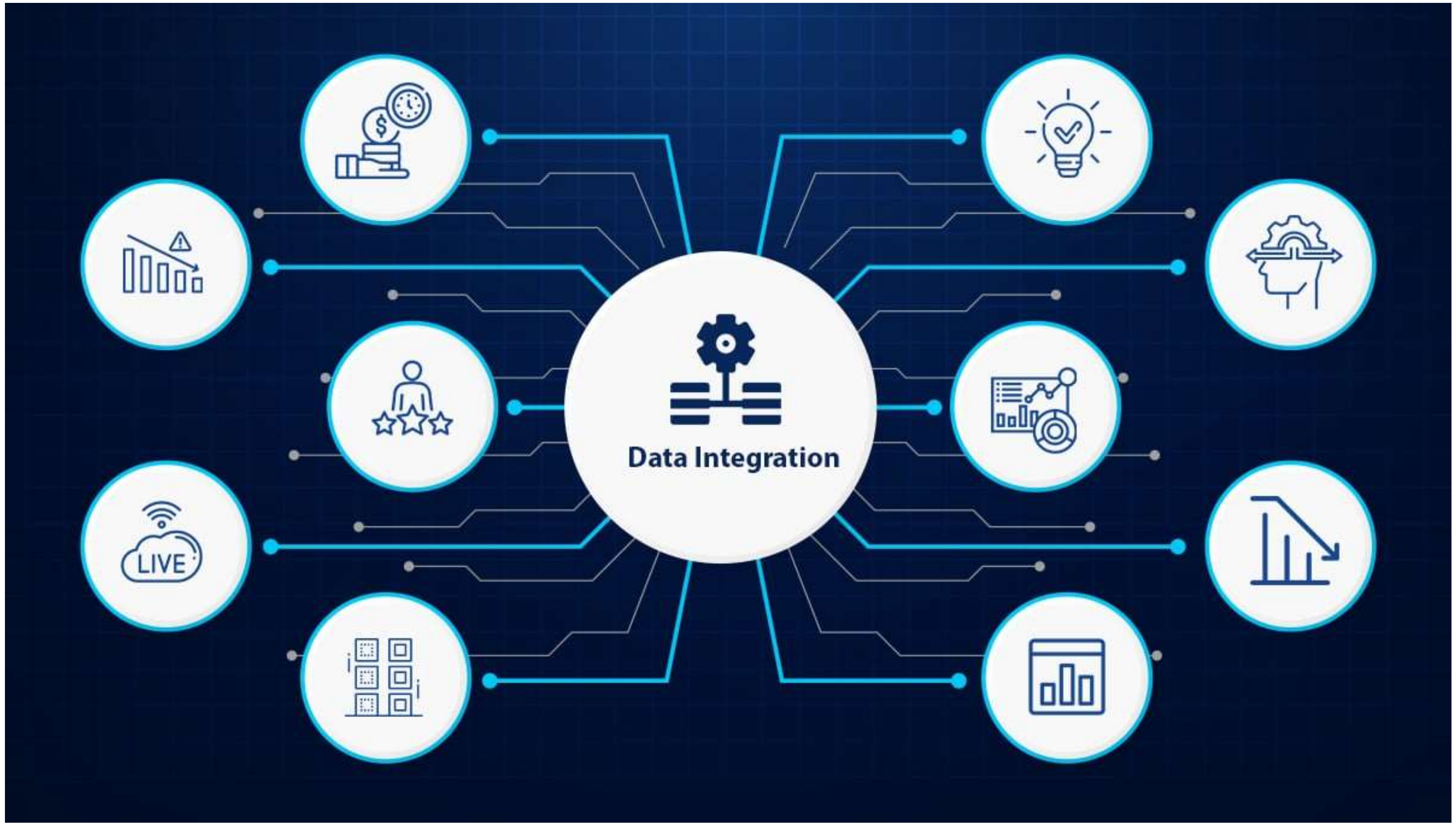
7. **Office Hours:** In person on Thursday between 11:30AM to 1PM OR on Zoom any other weekday between 10AM to 5PM.

Software you must have on your computer

1. Bash command line interface
2. Miniconda / Anaconda Python Distribution Individual edition (Python 3.11 and above).
3. Within the above installation is the editor Jupyter Lab in which you will do all the ML / DL related coding.
4. VSCode for script type Python programming (Streamlit, HTML, etc).
5. Make sure you have all this up and running from day 1.
6. Packages to be installed, which are not a part of the standard installation will be pointed out with instructions.
7. Please familiarize with the concept of "Virtual Environments" in the "conda" distribution as you are encouraged to work in virtual environments..

Overview of key topics in the course

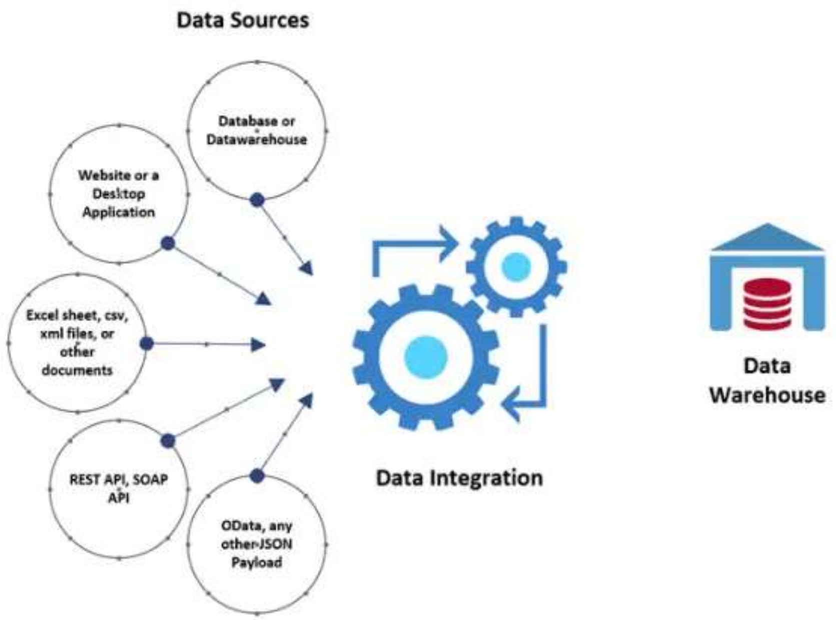
Data integration and data augmentation:
Creating a unified view dataset

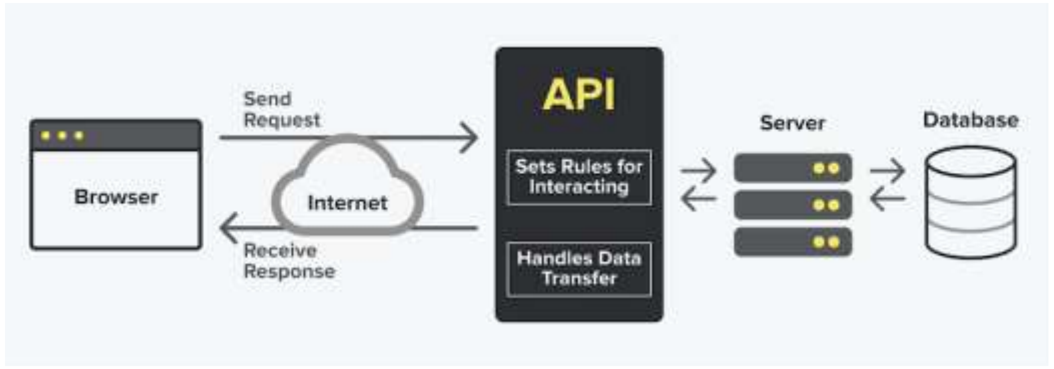


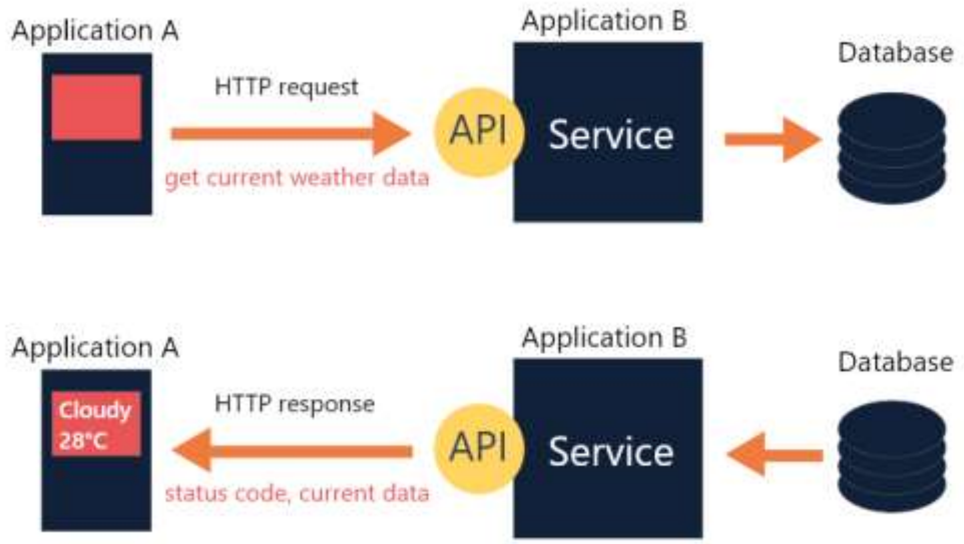
Dashboarding with ‘streamlit’

Building Visualization Systems like the cool one’s shown below

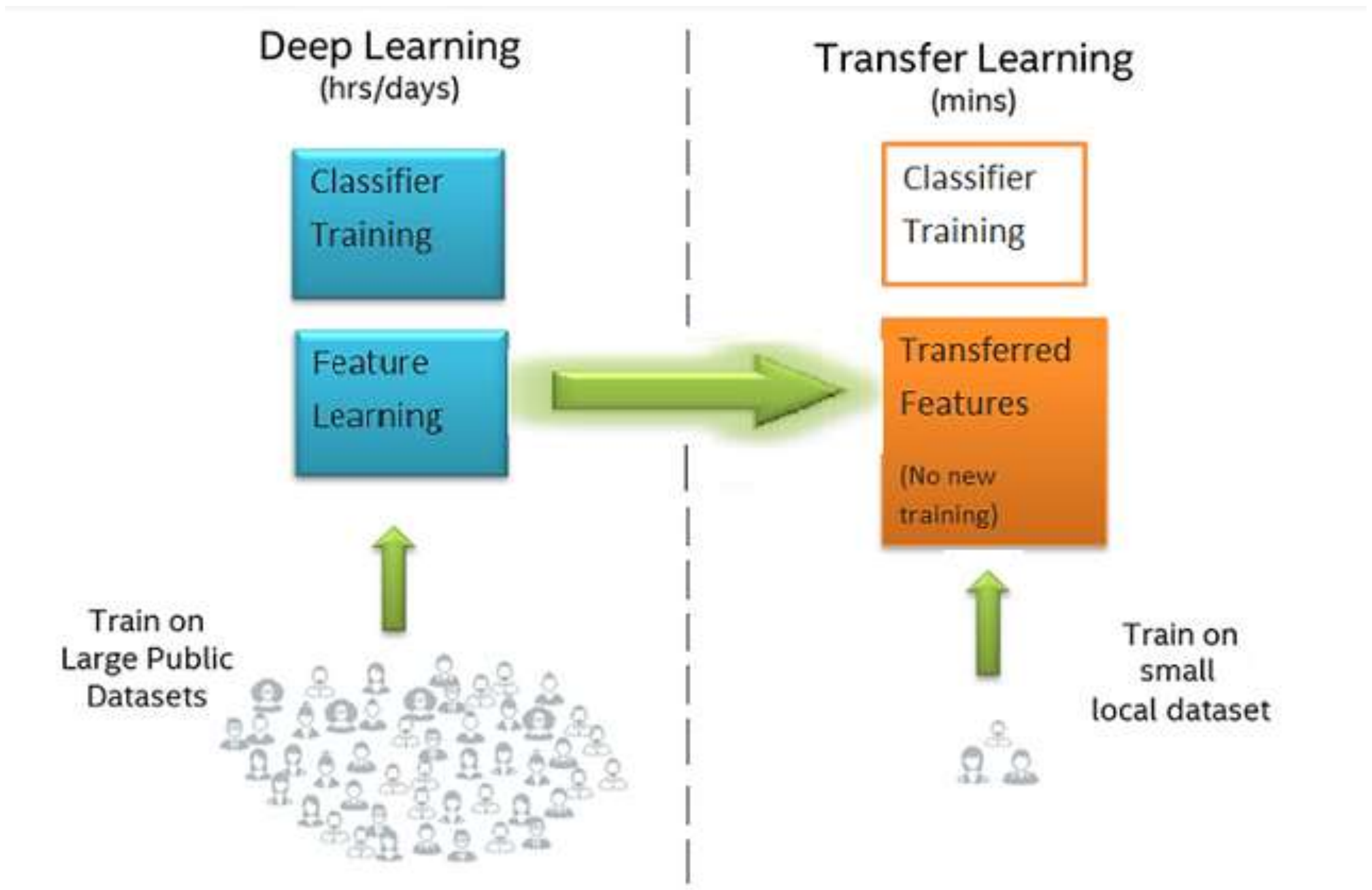
API – Application Programming Interface:
How to use / build APIs


Data Sources
Database or Datawarehouse
Website or a Desktop Application
Excel sheet, csv, xml files, or other documents
REST API, SOAP API
OData, any other JSON Payload
Data Integration
Data Warehouse

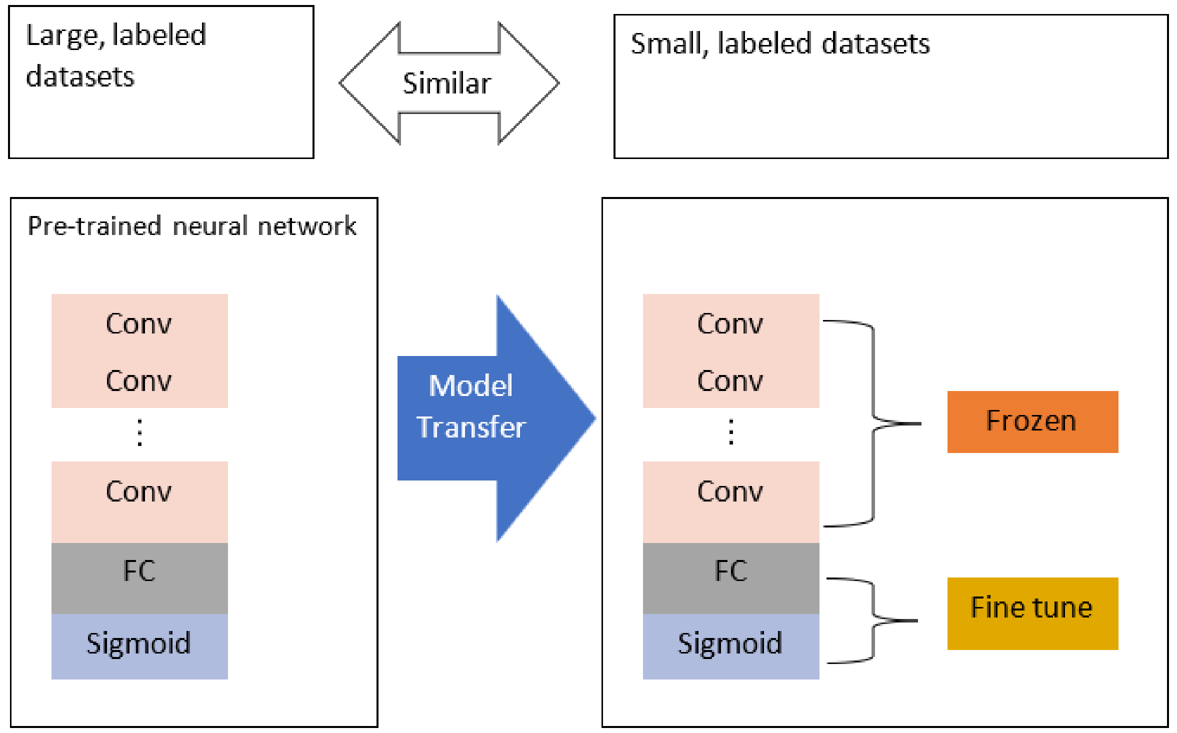


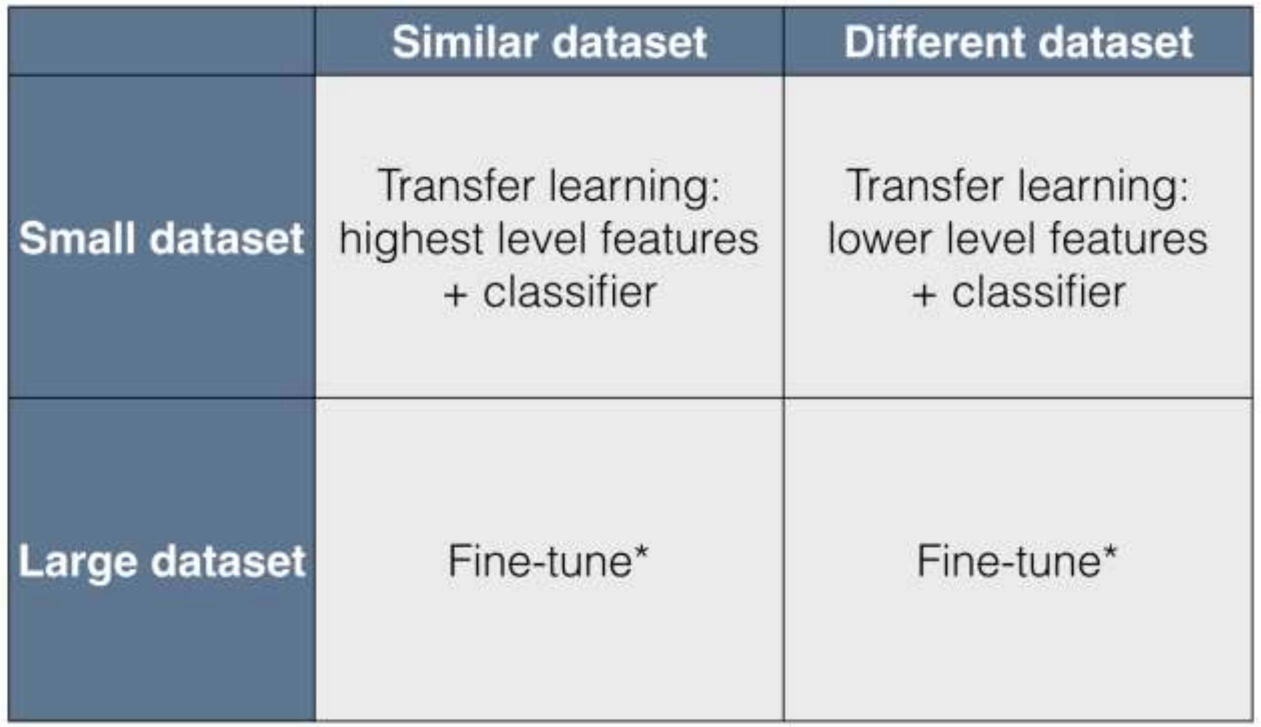


What is transfer learning?
Image Classification Context



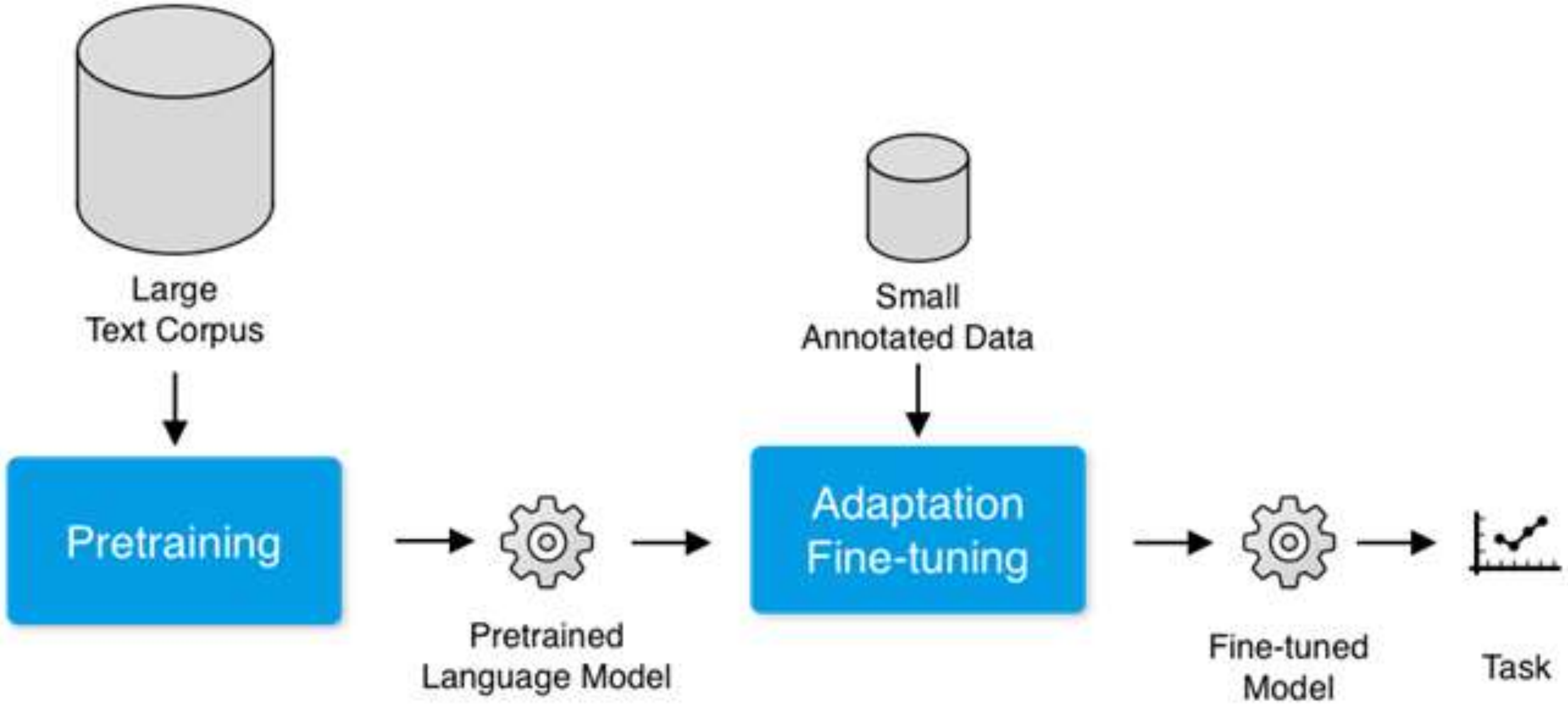
Freezing versus fine tuning





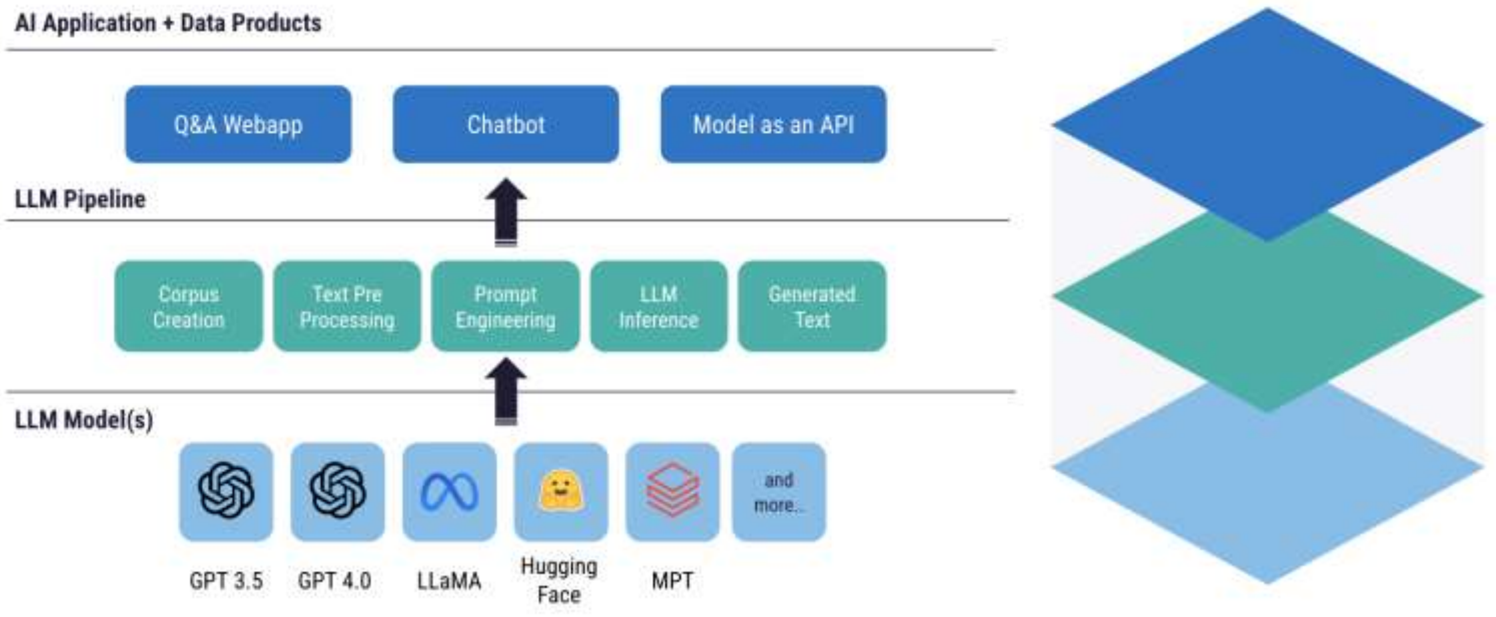
	Similar dataset	Different dataset
Small dataset	Transfer learning: highest level features + classifier	Transfer learning: lower level features + classifier
Large dataset	Fine-tune*	Fine-tune*

Transfer Learning:
NLP Context



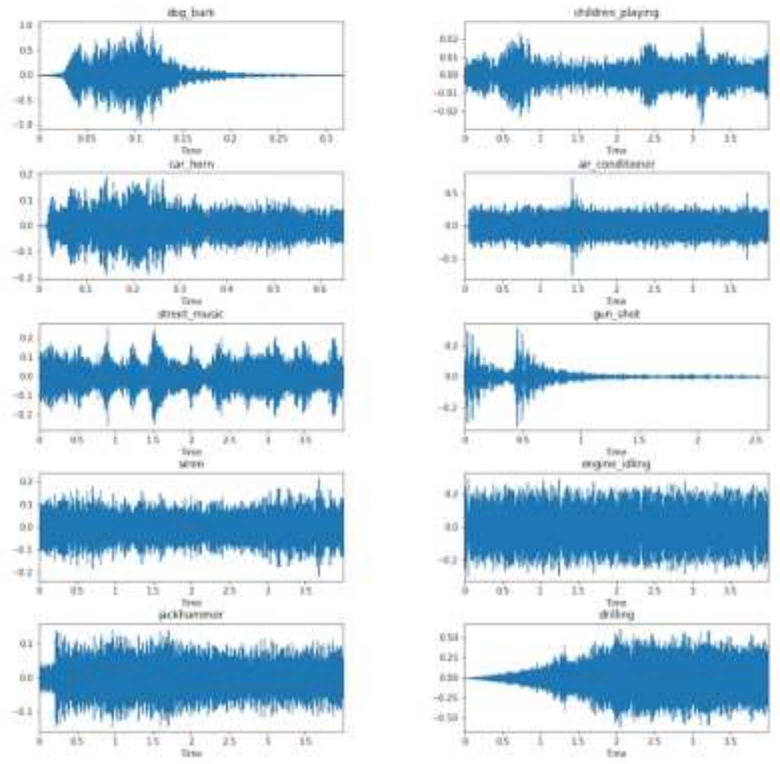
Building Applications with LLMs

1. LLM-based applications leverage the capabilities of Large Language Models (LLMs) to perform various tasks involving human language.
2. *Chatbots and Conversational AI*: LLMs power chatbots that can engage in natural language conversations with users, providing information or performing tasks.
3. *Question Answering:* LLMs can answer questions based on information they have been trained on or by accessing external knowledge sources.
4. Customer support: LLMs are transforming customer support by providing automated (yet deeply personalized) responses to inquiries. This technology enables businesses to offer 24/7 support without extensive human resources, improving customer satisfaction and operational efficiency.

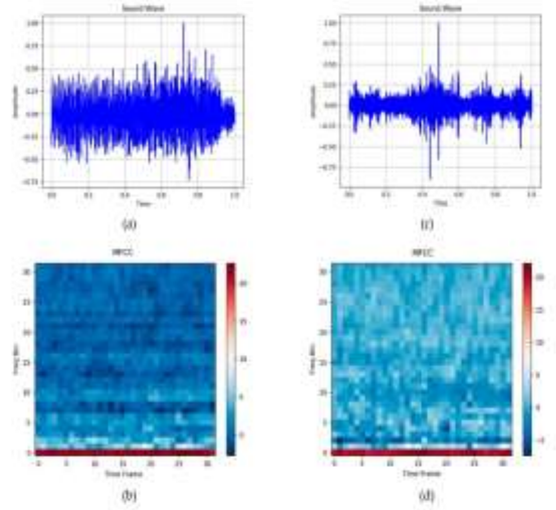


Audio analytics

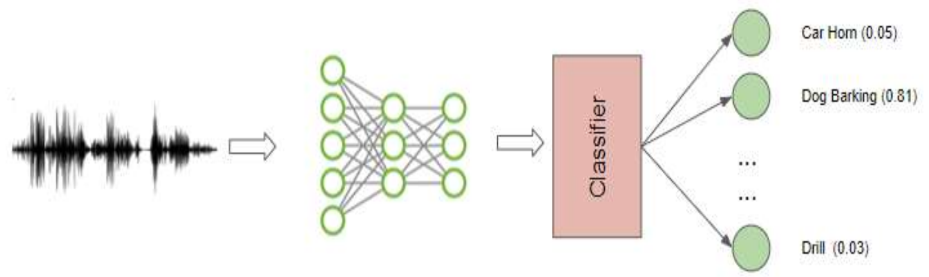
Audio Signals



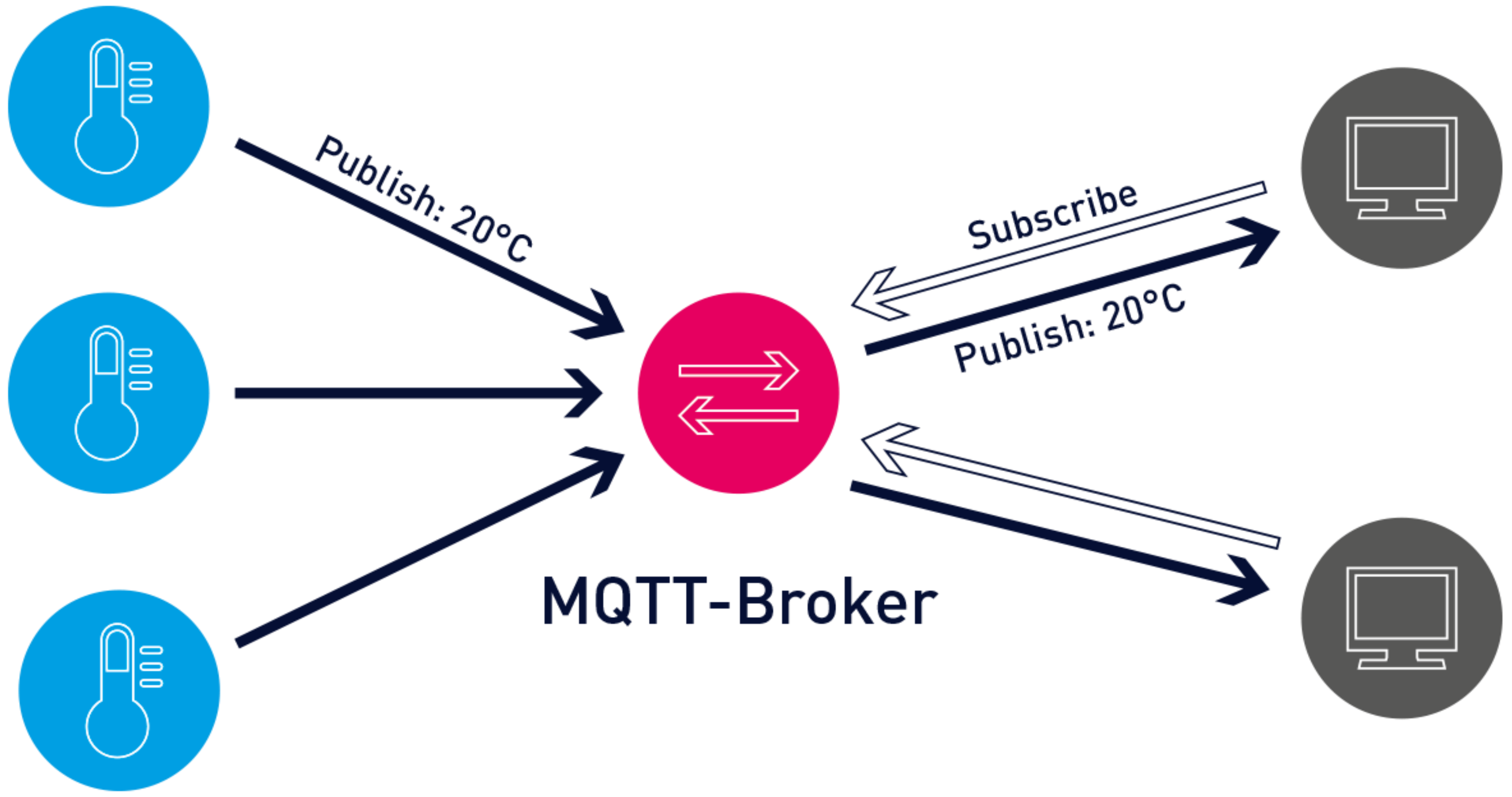
Feature extraction



Audio analytics applications



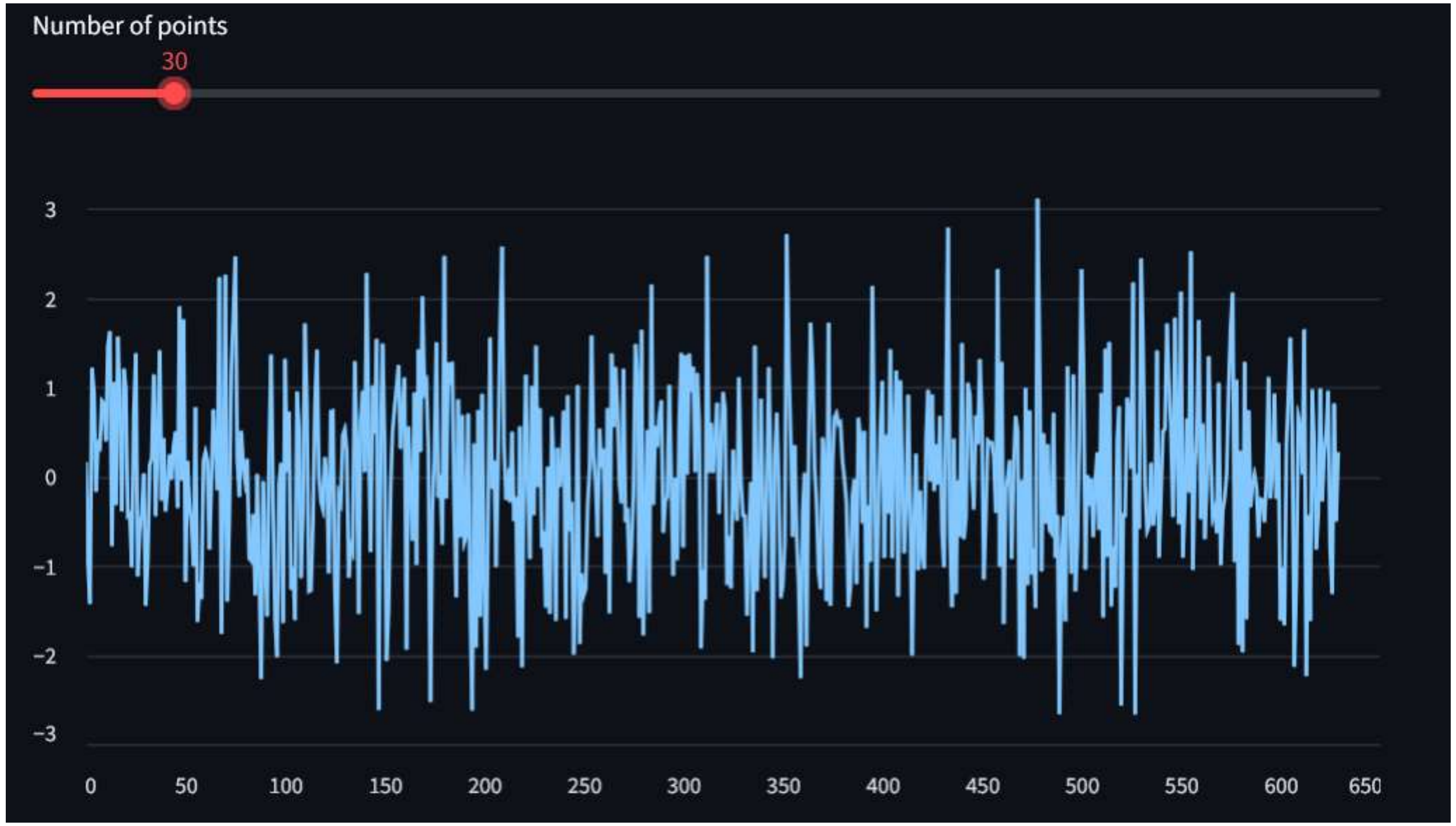
Communication Technologies:
Applications



App development with ‘streamlit’

(pip install streamlit)

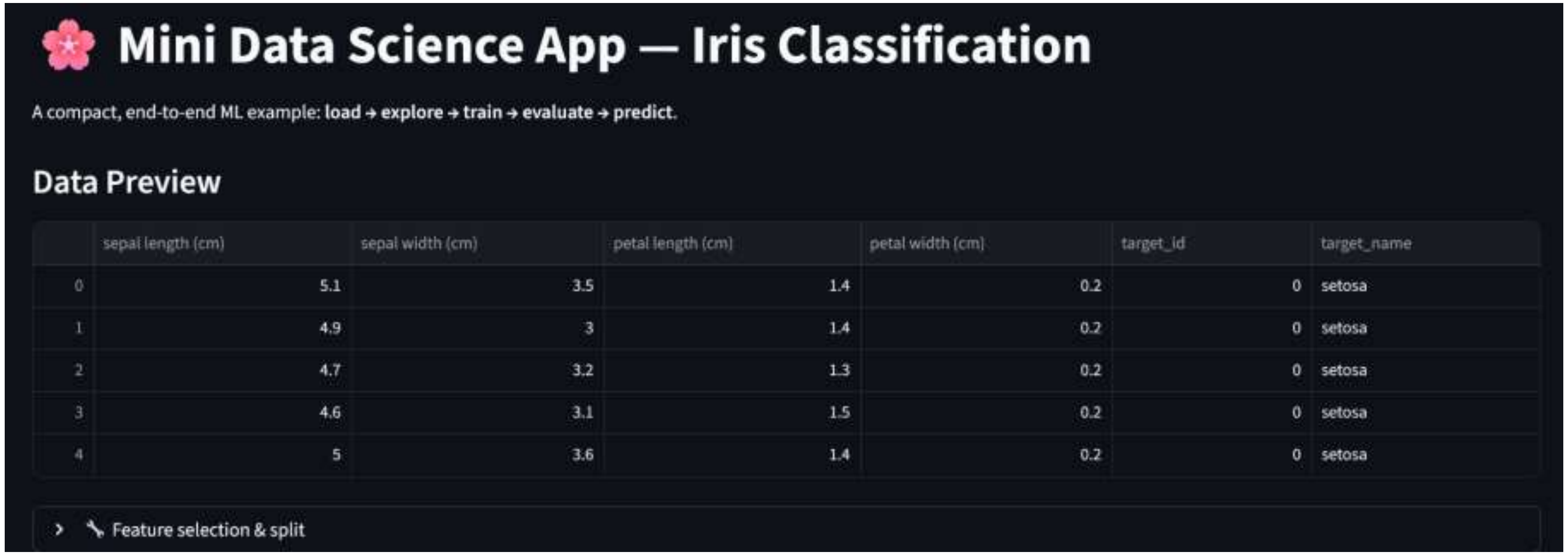
Let us build this simple data visualizing tool



<u>Example 1</u>

- Less than 15 lines of code
- No HTML/CSS/JS
- Interactive
- Deployable

Let us look at something data Science'y



Mini Data Science App — Iris Classification

A compact, end-to-end ML example: **load → explore → train → evaluate → predict.**

Data Preview

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target_id	target_name
0	5.1	3.5	1.4	0.2	0	setosa
1	4.9	3	1.4	0.2	0	setosa
2	4.7	3.2	1.3	0.2	0	setosa
3	4.6	3.1	1.5	0.2	0	setosa
4	5	3.6	1.4	0.2	0	setosa

Example 2

- Load data
- Plot and visualize
- Build Model
- Evaluate & predict

What is ‘streamlit’?

1. **Streamlit:** Is a python package which enables us to build complete applications and visualization dashboard in an easy way with a very small learning curve.

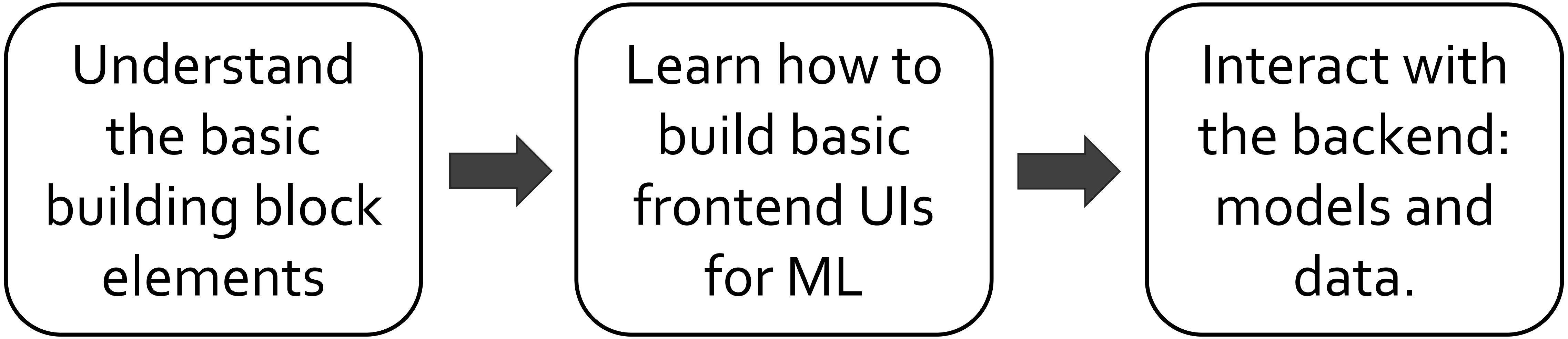
2. **Declarative Syntax:** Streamlit emphasizes simplicity through its declarative syntax. You primarily describe the elements you want to display on your web page, and Streamlit handles the underlying web development complexities.

3. **Widgets:** Streamlit offers a wide array of widgets, such as sliders, text inputs, checkboxes, and more, that allow users to interact with your application.

4. **Data Visualization:** Streamlit seamlessly integrates with popular data visualization libraries like Matplotlib, Plotly, and Altair, enabling you to create visually appealing charts and graphs within your web app.

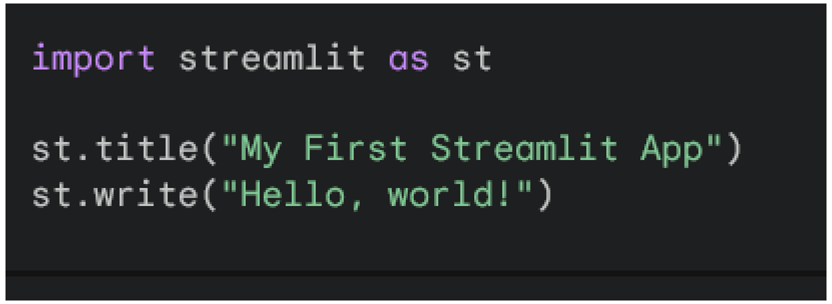
5. To use streamlit commands and features as extensively as you want, here is the link to the documentation - https://docs.streamlit.io/

Learning “streamlit” for Data Science purposes!


Understand the basic building block elements
Learn how to build basic frontend UIs for ML
Interact with the backend: models and data.

‘streamlit’ app elements

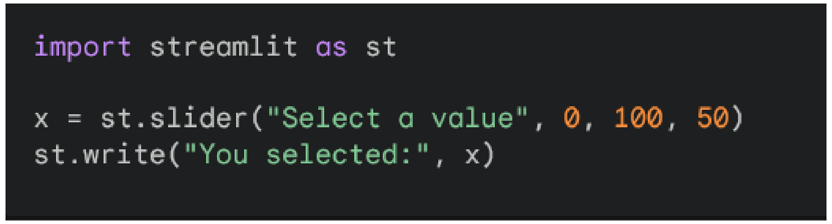
1. **Streamlit:** This code snippet creates a simple Streamlit app with a title and a greeting message.



```python
import streamlit as st

st.title("My First Streamlit App")
st.write("Hello, world!")
```

2. **Widgets:** This code features a slider that allows the user to select a value between 0 and 100. The selected value is then displayed below the slider.



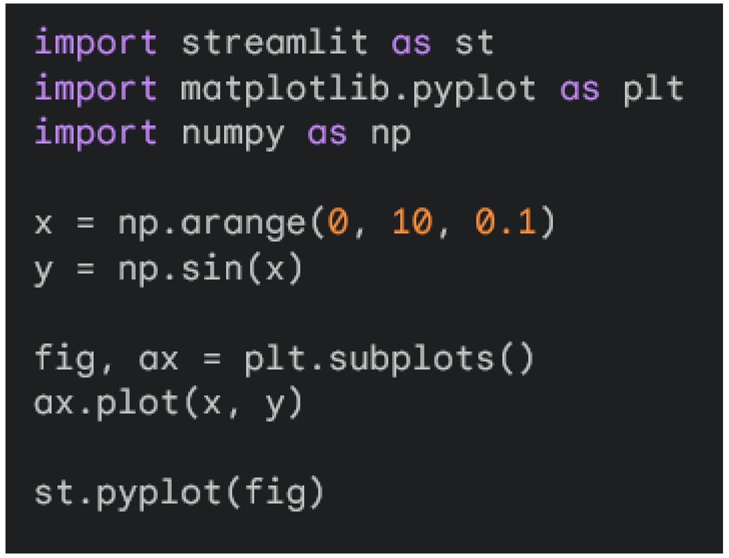
```python
import streamlit as st

x = st.slider("Select a value", 0, 100, 50)
st.write("You selected:", x)
```

1. **Data Visualization:** This code generates a sine wave plot using Matplotlib and displays it within the Streamlit app.

2. There are many such elements and a cheat sheet along with code examples have been uploaded on eLC under ‘resources.

Let us look at more examples



```python
import streamlit as st
import matplotlib.pyplot as plt
import numpy as np

x = np.arange(0, 10, 0.1)
y = np.sin(x)

fig, ax = plt.subplots()
ax.plot(x, y)

st.pyplot(fig)
```

Key points – Example 3 & 4

1. **st.columns:** This function creates a horizontal layout with the specified number of columns, allowing you to arrange elements side by side.

2. **with col1, with col2:** These context managers associate the subsequent Streamlit elements with the respective columns, ensuring they are placed in the correct locations.

3. **st.container:** This creates a container element that can be used to group multiple elements together and apply styling or layout options to them as a whole.

4. **st.image:** This function displays an image within your app. You'll need to provide the path to the image file.

5. **Matplotlib:** We use Matplotlib to create the line plot and bar chart, demonstrating how to integrate it with Streamlit.

Specific info about st.containers

1. st.container() is a quiet workhorse.
2. It lets you reserve a spot in the layout and fill it later, add multiple elements as a block, and even nest columns/forms/expanders inside.
3. Use it when you want a fixed “Results” area that updates after a button click, or when you want to group outputs cleanly.

Deploying model – Example 5

1. The app loads a sklearn model which has been saved.

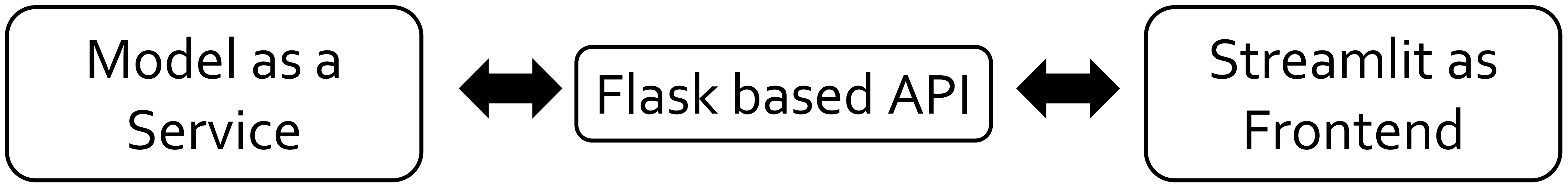
2. It used “joblib” to load the saved model and the model expects 2 numeric features as inputs.

3. The label is 1 or 0 and hence it is a classification model.

4. We can do a few checks to see if numbers have been entered.

5. The model gives the user a predictions based on inputs from the user.

Typical Architecture:
Streamlit as Frontend and API as Backend



Exercise 1

Exercise 1:
Build a streamlit based UI as per requirements below

1. Wide layout with a title and short caption.
2. Two columns: Left = Inputs, Right = Outputs.
3. Inputs include name (text), a number N (for how many points), optional image upload, and a Run button.
4. Use a results container on the right that displays:
   - A greeting if a name is provided.
   - A simple chart with N random points.
   - The uploaded image preview (if any).
5. Add an “Instructions” expander on the left and Tabs on the right (e.g., “Summary” & “Details”).
6. **Submit by 08/20/2025 midnight** -A single *.py that meets the checklist, a short note (3–5 sentences) describing where/why you used columns, container, tabs, and expander.

Data integration with APIs

What we did in INFO3000

User on a local machine:

1. Inputs the data for which a prediction is desired
2. Follows the input format provided by the application

Request and get response from a server:

1. Private or public
2. Runs and API which gives access to a saved model on the server
3. User data is sent to the model via the API and a prediction is returned

APIs – What we will additionally focus on

Read and Write to a Relational Database - SQLite

Request and get response from the internet

”requests” library

1. ‘requests’ allows us to send a simple request to an ‘url’ to get a response in a very easy way.

2. It is the easiest way to use APIs to retrieve data and populate a database.

3. *r = requests.get('https://api.github.com/user', auth=('user', 'pass’))*
   r.status_code 200
   r.headers['content-type'] 'application/json; charset=utf8’
   r.encoding 'utf-8’
   r.text '{"type":"User"…’
   r.json() {'private_gists': 419, 'total_private_repos': 77, …}

HTTP methods such as GET and POST, determine which action you’re trying to perform when making an HTTP request

One of the most common HTTP methods is GET. The GET method indicates that you’re trying to get or retrieve data from a specified resource. To make a GET request, invoke requests.get().

Documentation is at: https://docs.python-requests.org/en/master/api/

Getting HTML tables with Pandas

1. The pandas read_html() function is a quick and convenient way to turn an HTML table into a pandas DataFrame.

2. This function can be useful for quickly incorporating tables from various websites without figuring out how to scrape the site's HTML.

3. The unique point here is that this function call returns a list of tables – all the tables on that webpage are returned.

4. To make the table selection easier, use the match parameter to select a subset of tables.

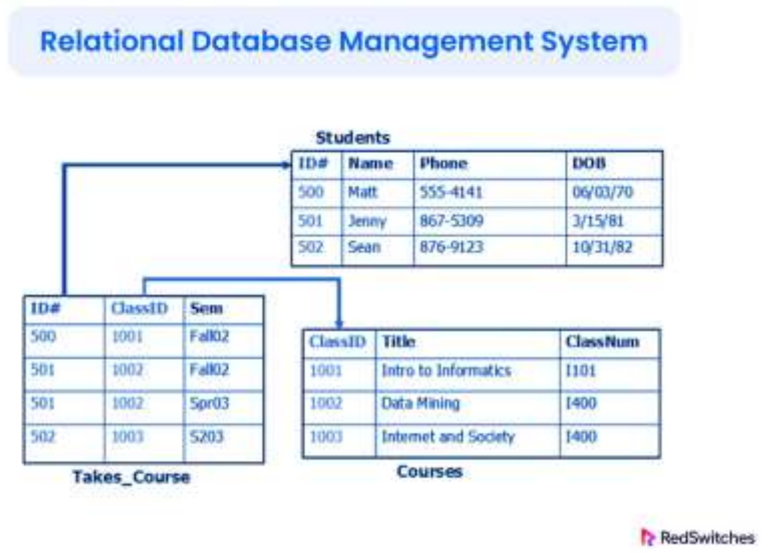
5. Of course, you must read and clean the tables so that you can use them for further analysis.

6. The pandas read_html() function is useful for quickly parsing HTML tables in pages - especially in Wikipedia pages. By the nature of HTML, the data is frequently not going to be as clean as you might need and cleaning up all the stray unicode characters is required and is time consuming.

SQL and Relational Databases

What is a relational database?

- Relational databases organize data into tables (also called relations). Each table represents a specific entity (e.g., customers, products, orders).
- **Tables and Columns:** Tables consist of rows (records) and columns (fields). Each row represents a single instance of the entity (e.g., a specific customer), while each column represents an attribute of the entity (e.g., customer name, address).
- **Primary Key:** A primary key uniquely identifies each row in a table (e.g., customer ID).
- **Foreign Key:** A foreign key is a column in one table that references the primary key of another table, creating a relationship between the two tables.
- **Relationships:** Relational databases support different types of relationships between tables:
  - Each row in one table is associated with a maximum of one row in another table.
  - **One-to-Many (or Many-to-One):** Each row in one table can be associated with many rows in another table, but each row in the second table can only be associated with one row in the first table.
  - **Many-to-Many:** Each row in one table can be associated with many rows in another table, and vice versa.
- **Structured Query Language (SQL):** SQL is the standard language used to interact with relational databases. It allows you to perform operations such as inserting, updating, deleting, and retrieving data.



Students

ID#	Name	Phone	DOB
500	Matt	555-4141	06/03/70
501	Jenny	867-5309	3/15/81
502	Sean	876-9123	10/31/82

ID#	ClassID	Sem
500	1001	Fall02
501	1002	Fall02
501	1002	Spr03
502	1003	S203

Takes_Course

ClassID	Title	ClassNum
1001	Intro to Informatics	I101
1002	Data Mining	I400
1003	Internet and Society	I400

Courses

Structured Query Language (SQL) basics

1. Think of a database as a big organized collection of information. SQL is how we ask questions, add data, change it, or remove it.

2. The Big Four Commands:

    SELECT: Retrieve data from the database.
    INSERT: Add new data to the database.
    UPDATE: Modify existing data.
    DELETE: Remove data.

    Basic SQL Syntax:
    Commands are written in all capital letters (though not strictly required).
    Each command usually starts on a new line for readability.
    Semicolons (;) are used to end each statement.

4. **SQLite:** Is a local, installed on your PC, database. It is Great for learning as it doesn't require a server setup and we will be using it in this course.

SQLite3 and SQL Overview

- The standard for relational database management systems (RDBMS)
- SQLite3: A database management system that manages data as a collection of tables in which all relationships are represented by common values in related tables
- SQL - Structured Query Language: The language used to talk to the database

Sqlite3 should be installed with your python package. If it is not, you can – 'pip install pysqlite3'.

Recommended is also to install the Sqlite browser so that you can take a look at the contents of your database and verify it when you are working on assignments:

https://sqlitebrowser.org/dl/

Learning Resources:

•**Online tutorials:**

- SQL Tutorial: https://www.sqltutorial.org/
- Codecademy Learn SQL: https://www.codecademy.com/learn/learn-sql

•**YouTube videos:** Search for "SQL for beginners"

Key SQL commands

SELECT

: Retrieves data from a database.
   Example: SELECT name, age FROM employees; (This gets the name and age of all employees)
INSERT: Adds new data into a table.
   Example: INSERT INTO employees (name, age) VALUES ('Alice', 30);
UPDATE: Modifies existing data in a table.
   Example: UPDATE employees SET age = 31 WHERE name = 'Alice';
DELETE: Removes data from a table.
   Example: DELETE FROM employees WHERE name = 'Alice';

SELECT Statement Details:

SELECT column1, column2, ...: Specifies which columns you want to retrieve. Use * to get all columns.
FROM table_name: Tells SQL which table to get data from.
WHERE condition: Filters the results based on a condition (optional).

Examples of WHERE Conditions:

WHERE age > 30 (Get employees older than 30)
WHERE name = 'Alice' (Get employees named Alice)
WHERE age > 30 AND salary < 50000 (Combine conditions with AND, OR, NOT)

Hands on:

requests, web scrape with pandas and sql database operations

End of Lesson