

Short-course & Hands-on Workshop: Artificial Intelligence Algorithms for Everyone

June 23 to June 26, 2025

Dr. R.M. (Rolando) Gonzales Martinez



university of
groningen

Let's get to know each other

1. Name
2. Where are you right now
3. How much do you know about statistics and AI in a scale from 1 (very little) to 10 (a lot)?
4. How much do you know about coding and coding in Python in a scale from 0 (nothing) to 10 (a lot)
5. Hobby



June 23
(Monday)

Introduction to artificial intelligence and Python programming (homogenization)

- The history of the spring and winters in artificial intelligence: from automatons to deep artificial neural networks and up to the singularity of human-level artificial intelligence
- Introduction to Python programming of artificial intelligence algorithms, elastic nets and vibe coding

June 24
(Tuesday)

Machine learning algorithms

- Machine learning algorithms for classification and regression: elastic nets, XGBoost, artificial neural networks, support vector machines, random forests
- Time Series Forecasting : modeling and predicting time series data in practice with META's Prophet and long short-term memory models
- Model evaluation, feature engineering, feature selection, and fine-tuning: metrics based on error minimization and the confusion matrix, Bayesian hypertuning

June 25
(Wednesday)

Deep learning algorithms and recent trends in Artificial Intelligence

- Deep learning algorithms and Large Language Models (LLMs): the Transformers' architecture
- Semantic and sentiment analysis of textual data with RoBERTa
- Ethical considerations on the application of AI algorithms and mitigation strategies
- Recent advances in AI: liquid neural networks, how to use DeepSeek without worrying about privacy issues, spatial machine learning, systematic reviews and literature reviews in 5-minutes with GPTs, quantum computing, liquid neural networks

June 26
(Thursday)

Presentations:

- Using Mobile Phone Surveys (MPS) to measure child mortality during crisis periods (Kassoum Dianou)
- Ways in which AI can support autonomous navigation in Space Missions (Duna Meya i Mendoza)

Practical applications

- AI as psychological assistants & therapists
- Determinants of salary (income)
- Credit scoring
- Demographic (population) forecasts (time series forecasting)
- Literature reviews with ChatGPT and DeepSeek
- Locally running LLMs

Other examples:

- Spatial machine learning
- Semantic and sentiment analysis of textual data with RoBERTa

Schedule (CET)

- First section: 2:30 PM to 3:15 PM

-
- First break: 3:15 PM to 3:30 PM

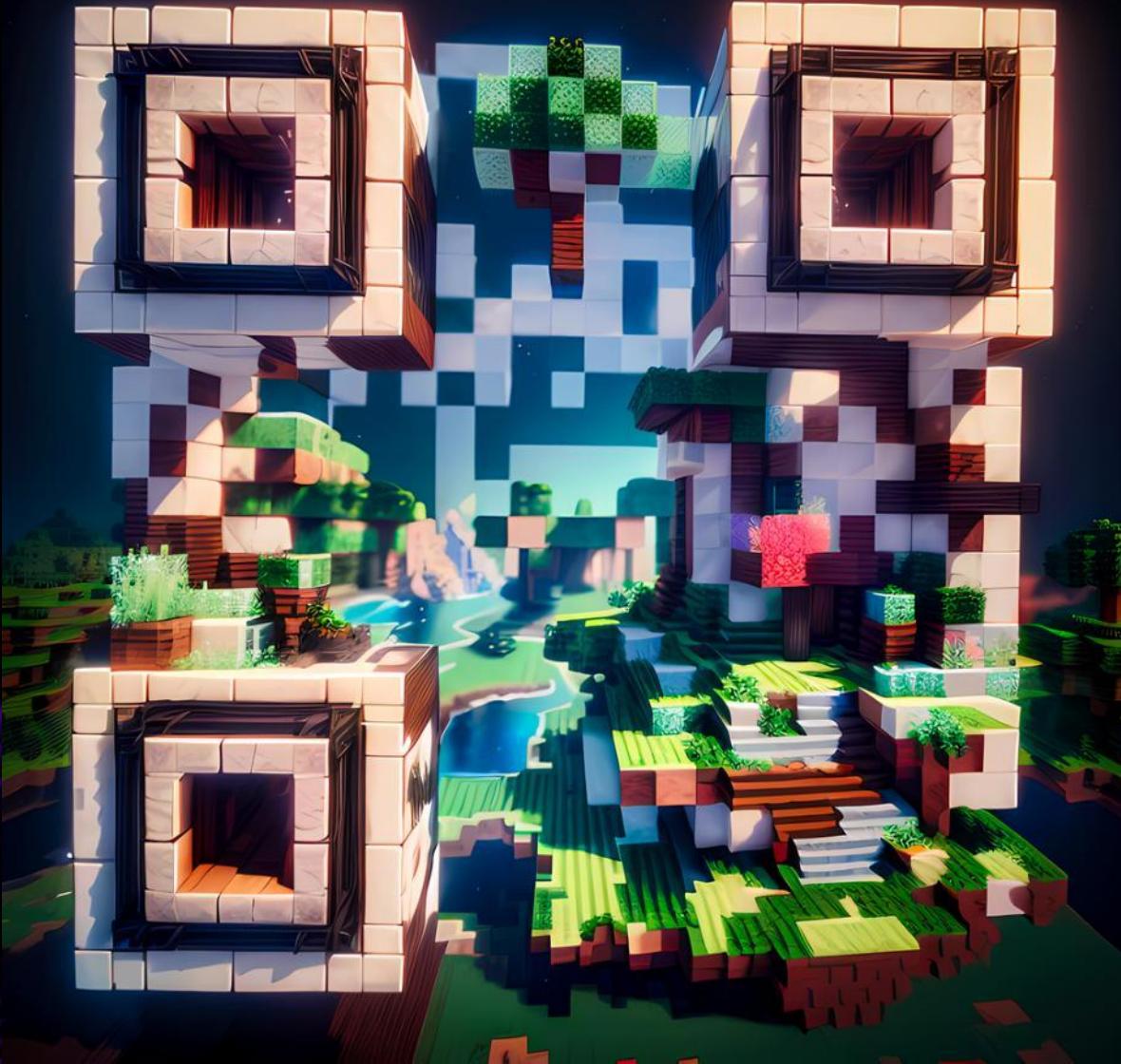
- Second section: 3:30 PM to 4:15 PM

-
- Second break: 4:15 PM to 4:30 PM

- Third session: 4:30 PM to 5:00 PM

-
- Optional extra space for questions and comments: 5:00 PM to 5:30 PM

GitHub repository



https://github.com/rogon666/AI_workshop

- Presentations
- Databases
- Python scripts
- Additional lectures
(papers, books)

June 23
(Monday)

Introduction to artificial intelligence and Python programming (homogenization)

- The history of the spring and winters in artificial intelligence: from automatons to deep artificial neural networks and up to the singularity of human-level artificial intelligence
- Introduction to Python programming of artificial intelligence algorithms

June 24
(Tuesday)

Machine learning algorithms

- Machine learning algorithms for classification and regression: elastic nets, XGBoost, artificial neural networks, support vector machines, random forests
- Time Series Forecasting : modeling and predicting time series data in practice with META's Prophet and long short-term memory models
- Model evaluation, feature engineering, feature selection, and fine-tuning: metrics based on error minimization and the confusion matrix, Bayesian hypertuning

June 25
(Wednesday)

Deep learning algorithms and recent trends in Artificial Intelligence

- Deep learning algorithms and Large Language Models (LLMs): the Transformers' architecture
- Semantic and sentiment analysis of textual data with RoBERTa
- Ethical considerations on the application of AI algorithms and mitigation strategies
- Recent advances in AI: liquid neural networks, how to use DeepSeek without worrying about privacy issues, spatial machine learning, systematic reviews and literature reviews in 5-minutes with GPTs, quantum computing, liquid neural networks

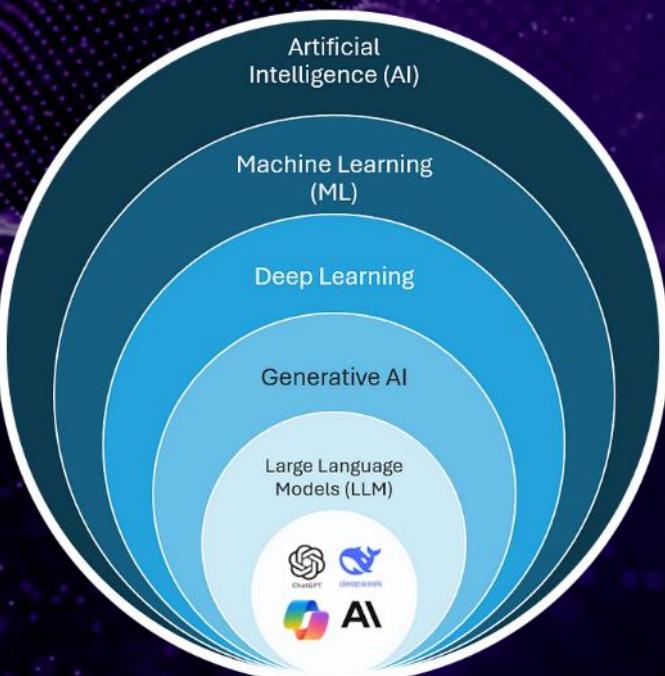
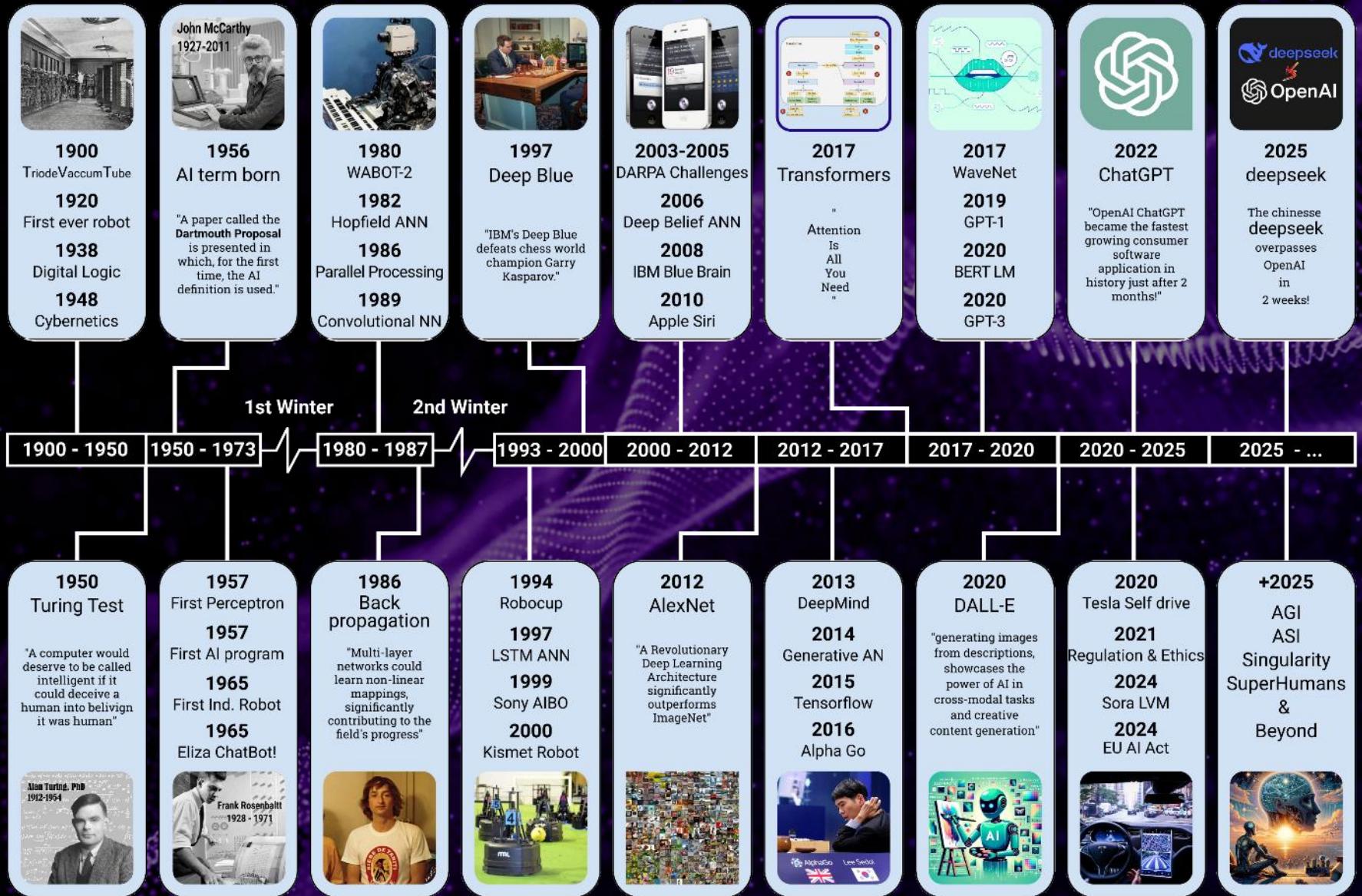
June 26
(Thursday)

Using Mobile Phone Surveys (MPS) to measure child mortality during crisis periods: challenges and implications for health systems in Low- and Middle-Income Countries

Kassoum Dianou

Joseph Ki-Zerbo University and Catholic University of Louvain

AI: history and definition



AI: history and definition

“The exciting new effort to make computers think... machines with minds, in the full and literal sense.”

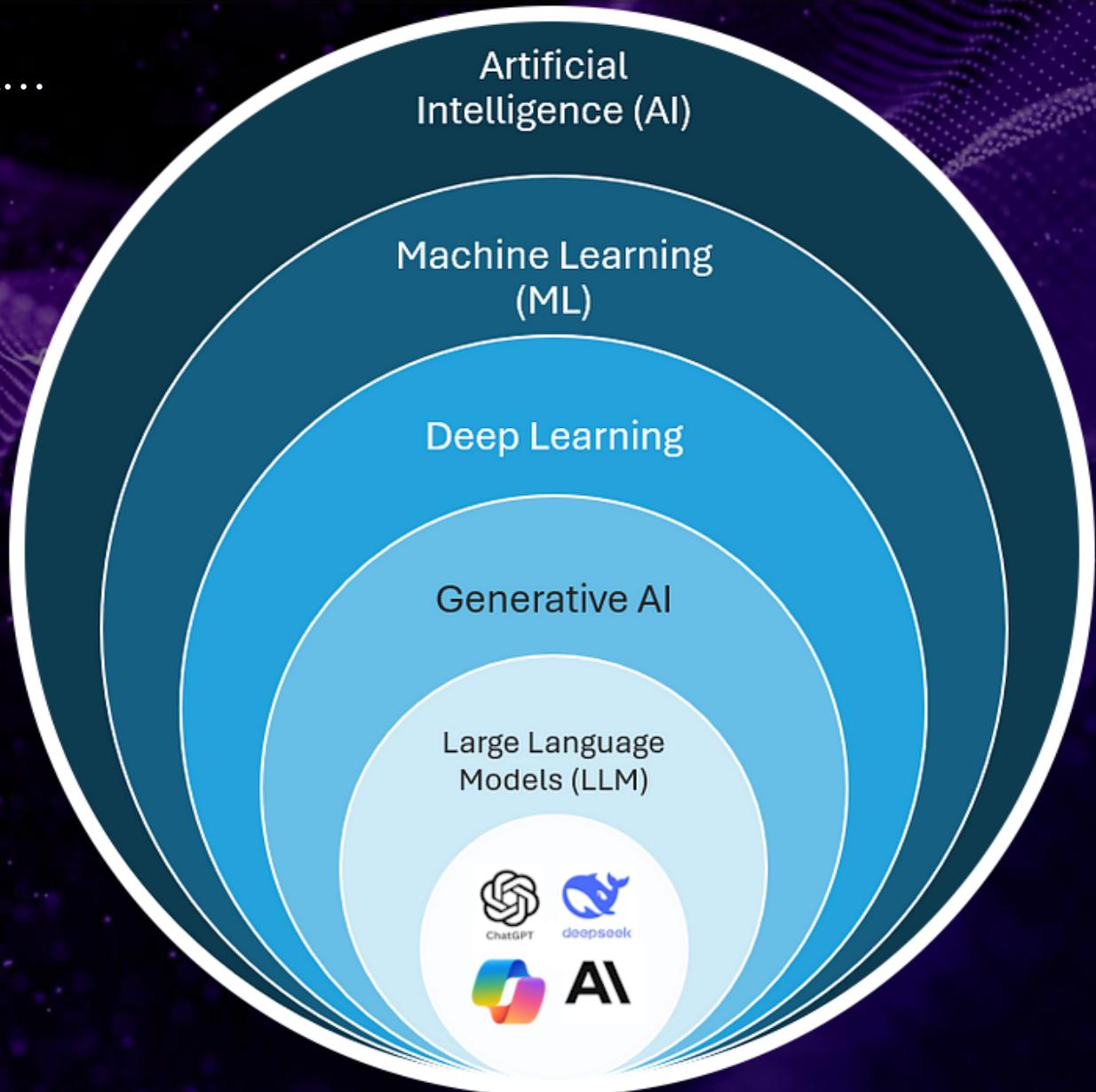
— Minsky, M. (1968). *Semantic Information Processing*. MIT Press.

“Probabilistic AI seeks to understand how to design systems that reason and act under uncertainty... using tools from probability theory and statistics.”

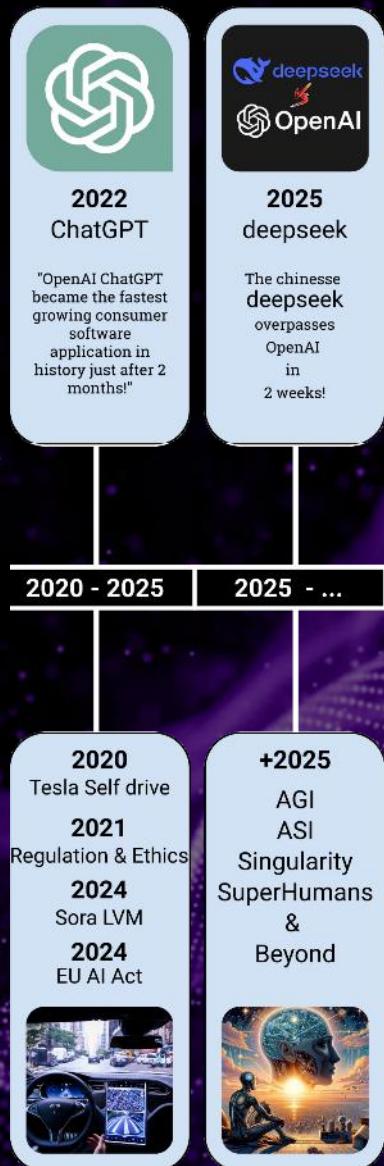
— Russell, S., & Norvig, P. (2020). *Artificial Intelligence: A Modern Approach* (4th ed.). Pearson.

“The essence of intelligence is inference from incomplete information, which the Bayesian paradigm addresses with normative optimality.”

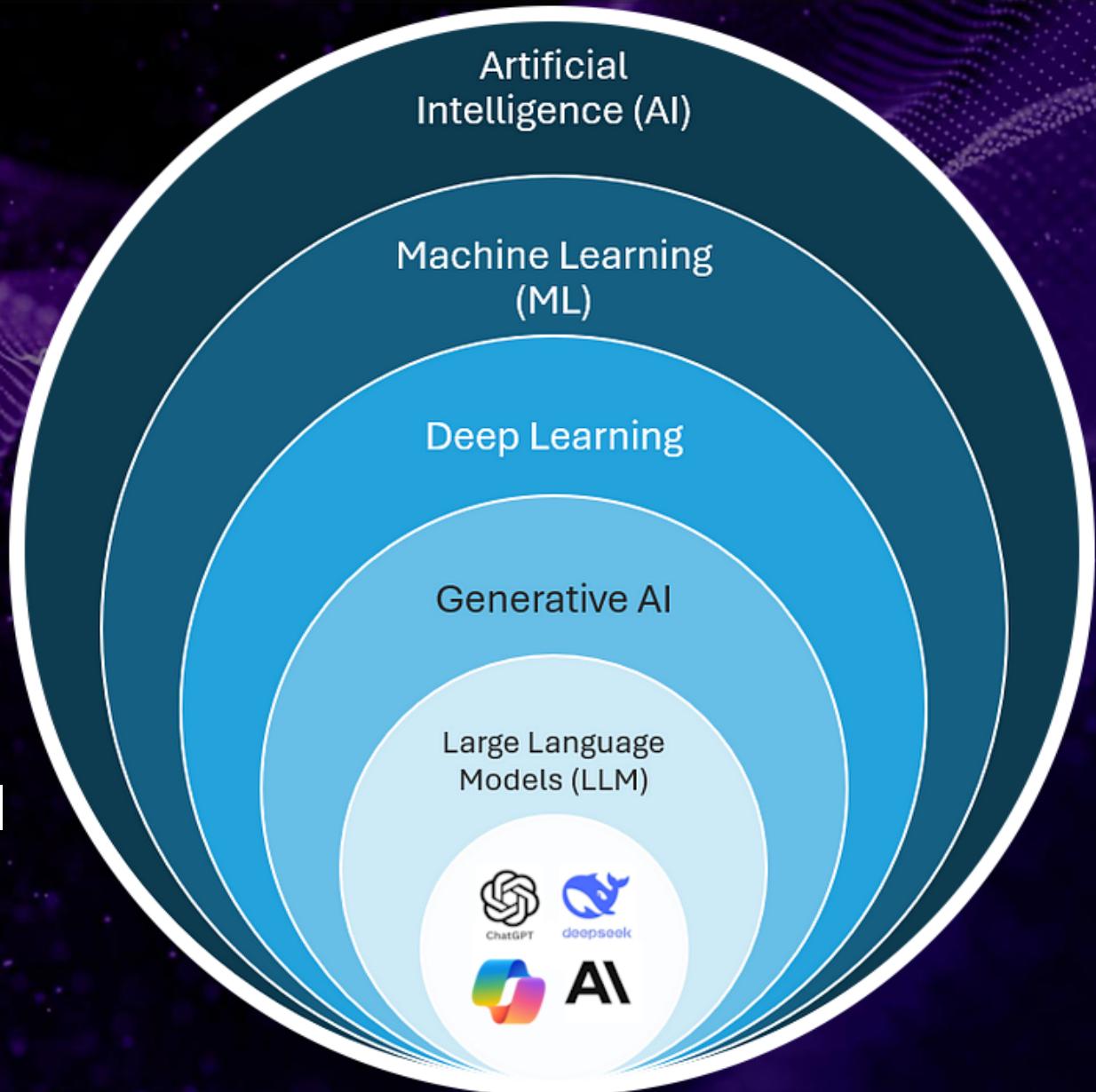
— Ghahramani, Z. (2015). *Probabilistic machine learning and artificial intelligence*. Nature, 521(7553), 452–459.



AI: history and definition



Narrow AI
vs.
AGI, ASI



AI algorithms and AI models

Algorithm:

A series of logical steps or instructions used to perform a specific task

```
Output: salary
Input: age, experience, education
if age > 6
    for (age) t = 1:n
        education(t) = f(studies)
        while (education & age) > acceptable
            for (years) e = 1:m
                experience = e
            end
            salary(t) = f(education, age, experience, θ)
        end
    end
end
```

AI algorithms and AI models

Model:

Simplified representation of a process, system, or phenomenon from the real world. It is a mathematical or statistical construct.

$$\text{salary} = f(\text{education}, \text{age}, \text{experience})$$

$$\text{salary} = f(\text{education}, \text{age}, \text{experience}; \Theta)$$

$$\Theta = [\theta_{\text{educ}}, \theta_{\text{age}}, \theta_{\text{expe}}]$$

$$\text{salary} = f(\text{education}, \text{age}, \text{experience}; \theta_{\text{educ}}, \theta_{\text{age}}, \theta_{\text{expe}})$$

Assuming that $f(\cdot)$ is linear:

$$\text{salary} = \theta_0 + \theta_{\text{educ}} \cdot \text{education} + \theta_{\text{age}} \cdot \text{age} + \theta_{\text{expe}} \cdot \text{experience}$$

Adding an i.i.d. term: $\varepsilon \sim \mathcal{N}(\mu, \sigma^2)$

$$\text{salary} = \theta_0 + \theta_{\text{educ}} \cdot \text{education} + \theta_{\text{age}} \cdot \text{age} + \theta_{\text{expe}} \cdot \text{experience} + \varepsilon$$

AI algorithms and AI models

$$\text{salary}_i = \theta_0 + \theta_{\text{educ}} \cdot \text{education}_i + \theta_{\text{age}} \cdot \text{age}_i + \theta_{\text{expe}} \cdot \text{experience}_i + \varepsilon_i$$

Estimation Algorithm:

Input: salary, education, age, experience data

Output: estimators of $\theta_{\text{educ}}, \theta_{\text{age}}, \theta_{\text{expe}}$

1. Matrix Representation:

$$\Theta = [\theta_{\text{educ}}, \theta_{\text{age}}, \theta_{\text{expe}}]$$

$$X = [\text{education}_i, \text{age}_i, \text{experience}_i]$$

$$y = [\text{salary}_1, \text{salary}_2, \dots, \text{salary}_n]$$

$$y = X\Theta + \varepsilon$$

$$y - X\Theta = \varepsilon$$

2. Matrix Operations: Calculation of transposed and inverse matrices: X' , $(X'X)^{-1}$

3. OLS Estimators: $\hat{\Theta} = (X'X)^{-1}X'y \quad i = 1, 2, \dots, n$

$$\hat{\Theta} = [\hat{\theta}_{\text{educ}}, \hat{\theta}_{\text{age}}, \hat{\theta}_{\text{expe}}]$$

Sample partitions in AI

- **Train-Test Split:** The most basic and common method. The data is divided into two parts: a **training set** to **train** the model (estimate the model parameters) and a **test set** to evaluate the model's performance on unseen data.
- **Hold-Out with Validation:** Similar to the train-test split, but includes a third validation set:
 - **Training (Train):** To adjust the model parameters.
 - **Validation:** To adjust hyperparameters and prevent overfitting.
 - **Test:** For the final evaluation of the model.

Sample partitions in AI

Data Partitioning in 4 Groups:

Training (train): $i = 1, \dots, m$

person	salary	education	age	experience
$i=1$	2640	24	40	10
...
$i=8$	6400	30	57	12
...
$i=m$	3555	29	32	9

y_{train} $X_{(\text{train})}$

The training set is used to train the model (estimate the model parameters).

Evaluation (test): $i = m, m + 1, \dots, n$

person	salary	education	age	experience
$i=m$	5656	27	38	5
$i=m+1$	3322	24	37	10
...
$i=n$	4300	32	40	13

y_{test} X_{test}

The test set is used to evaluate the model's performance on unseen data.

Sample partitions in AI

Data Partitioning:

Training (train): $i=1, \dots, m$

$$salary_i = \theta_0 + \theta_{educ}education_i + \theta_{age}age_i + \theta_{expe}experience_i + \varepsilon_i$$

$$\boldsymbol{\theta} = [\theta_{educ}, \theta_{age}, \theta_{expe}],$$

$$X_{\text{train}} = [education_i, age_i, experience_i],$$

$$y_{\text{train}} = [salary_1, salary_2, \dots, salary_m]$$

$$y_{\text{train}} = X\boldsymbol{\theta} + \varepsilon$$

$$\underset{\substack{\boldsymbol{\theta} \in \mathbb{R} \\ i=1,2,\dots,m}}{\operatorname{argmin}} \varepsilon: \hat{\boldsymbol{\theta}} = (X_{\text{train}}'X_{\text{train}})^{-1}X_{\text{train}}'y_{\text{train}}$$

Evaluation (test): $i = m, m+1, \dots, n$

$$y_{\text{predicted}} = X_{\text{test}}\hat{\boldsymbol{\theta}}$$

Mean Squared Error (MSE):

$$MSE = \frac{1}{m} \sum_{i=m}^n (y_{\text{test},i} - y_{\text{predicted},i})^2$$

The training set is used to train the model (estimate the model parameters).

The test set is used to evaluate the model's performance on unseen data.

Jargon

Econometrics and Statistics

Estimation

Parameters

Explanatory variables (X).

Dependent variable (y)

Estimation in the full sample*
(limited samples).

Purpose: inference, prediction.

Maximization of in-sample fit.

(*) In general, but not always, e.g., rolling estimation.

Machine learning

Learning, training

Weights

Features/inputs

Target/output

Data partitioning.(train/test)
(big data)

Purpose: prediction

Minimization of prediction error

Machine learning in AI

Given an input space X (feature space) and an output space Y , machine learning is an optimization problem where the goal is to find a function $f: X \rightarrow Y$ that predicts the output $y \in Y$ given an input $x \in X$, with f^* being optimal in terms of **generalization** and **regularization**:

$$\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\} \quad x_i \in \mathcal{X} \quad y_i \in \mathcal{Y}$$

$$f^* = \arg \min_{f \in \mathcal{F}} \mathbb{E}_{(x,y) \sim P} [L(f(x), y)]$$

$$f^* = \arg \min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n L(f(x_i), y_i)$$

$$f^* = \arg \min_{f \in \mathcal{F}} \left[\frac{1}{n} \sum_{i=1}^n L(f(x_i), y_i) + \lambda R(f) \right]$$

Machine learning in AI

Given a set of functions (models), M_1, M_2, \dots, M_k , it is necessary to consider performance measures for continuous and discrete variables when deciding on the best function (model):

$$M_1, M_2, \dots, M_k$$

$$\hat{\beta}_{\text{train},i} = \arg \max_{\beta} \log \mathcal{L}(\beta; X_{\text{train},i}, y_{\text{train}})$$

$$\hat{y}_{\text{test},i} = g^{-1}(X_{\text{test},i} \hat{\beta}_{\text{train},i})$$

$$\text{MSE}_i = \frac{1}{n_{\text{test}}} \sum_{j=1}^{n_{\text{test}}} (y_{\text{test},j} - \hat{y}_{\text{test},ij})^2$$

$$R_i^2 = 1 - \frac{\sum_{j=1}^{n_{\text{test}}} (y_{\text{test},j} - \hat{y}_{\text{test},ij})^2}{\sum_{j=1}^{n_{\text{test}}} (y_{\text{test},j} - \bar{y}_{\text{test}})^2}$$

$$\text{AUC-ROC}_i = \text{AUC}(\text{Curva ROC}_i)$$

$$M_{\text{best}} = \arg \min_i \text{MSE}_i$$

$$M_{\text{best}} = \arg \max_i R_i^2$$

$$M_{\text{best}} = \arg \max_i \text{AUC-ROC}_i$$

Elastic Nets

Elastic Nets are based on a regularization that aims to find a solution that minimizes the loss function with a combined penalty of L1 (Lasso) and L2 (Ridge).

$$\min_{\beta} \left\{ \frac{1}{2N} \sum_{i=1}^N \left(y_i - \sum_{j=1}^p X_{ij} \beta_j \right)^2 + \lambda_1 \sum_{j=1}^p |\beta_j| + \lambda_2 \sum_{j=1}^p \beta_j^2 \right\}$$

Elastic Nets are useful in situations where the predictor variables are highly correlated or when the number of variables is greater than the number of observations.

Elastic Nets

Lasso (Least Absolute Shrinkage and Selection Operator, L1), Ridge (L2), and Elastic Nets are regularization techniques that help reduce overfitting and deal with multicollinearity and redundant variables.

$$\hat{\beta}_{\text{ridge}} = \arg \min_{\beta} \left(\sum_{i=1}^m (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^n \beta_j^2 \right)$$

$$\hat{\beta}_{\text{lasso}} = \arg \min_{\beta} \left(\sum_{i=1}^m (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^n |\beta_j| \right)$$

$$\hat{\beta}_{\text{elastic}} = \arg \min_{\beta} \left(\sum_{i=1}^m (y_i - \hat{y}_i)^2 + \lambda_1 \sum_{j=1}^n |\beta_j| + \lambda_2 \sum_{j=1}^n \beta_j^2 \right)$$

Elastic Nets

Lasso (Least Absolute Shrinkage and Selection Operator, L1), Ridge (L2), and Elastic Nets are regularization techniques that help reduce overfitting and deal with multicollinearity and redundant variables.

In `sklearn.linear_model.ElasticNet`, you're solving the following regularized least-squares problem on your data $(X, y) \in \mathbb{R}^{n \times p} \times \mathbb{R}^n$:

$$\min_{w \in \mathbb{R}^p, b \in \mathbb{R}} \frac{1}{2n} \|y - Xw - b\mathbf{1}\|_2^2 + \alpha \left(\underbrace{\ell_1\text{-term}}_{l_1\text{-ratio} \|w\|_1} + \underbrace{\ell_2\text{-term}}_{(1-l_1\text{-ratio}) \frac{1}{2}\|w\|_2^2} \right)$$

$\alpha > 0$ scales the overall penalty strength.

$$\|w\|_1 = \sum_{j=1}^p |w_j|.$$

$$\|w\|_2^2 = \sum_{j=1}^p w_j^2.$$

$l_1_ratio \in [0, 1]$ interpolates between **pure Lasso** and **pure Ridge**.

Python

- Programming language.
- Python was conceived in the late 1980s by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands.
- 2000: Python 2.0
- 2018: Guido van Rossum, the 'benevolent dictator for life' of Python, responsible for the project's decision-making, leaves the project.
- 2025: Python 3.x

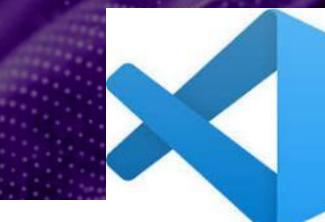
Used by Wikipedia, Google, Yahoo!, CERN, NASA, Facebook, Amazon, Instagram, Spotify.



Python

IDE (Integrated Development Environments):

- PyCharm
- Jupyter
- Spyder
- Python IDLE
- Thonny
- ...



Th



A screenshot of the Spyder IDE interface. The top bar shows the title "Spyder (Python 3.9)". The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, Help. The toolbar has icons for file operations like Open, Save, Run, and Stop. The main area shows three tabs: temp.py, Hao.py, and IRT_Andrew.py. The code editor displays Python script code. To the right is a Variable Explorer pane showing a table with columns Name, Type, Size, and Value. Below the editor is a Console tab with the IPython console output. The bottom status bar shows "LSP Python: ready", "conda: base (Python 3.9.13)", and memory usage "Mem 59%".

```
# -*- coding: utf-8 -*-
"""
Created on Mon Jul 24 07:22:02 2023
@author: Rolando
"""

from IPython import get_ipython
get_ipython().magic('reset -sf')

import pandas as pd

import os
os.getcwd()
db_in = 'Z:\UpWork\UpWork2023\Hao'
os.chdir(db_in)

# %% Loading data
du = pd.read_csv("5_esm.csv", sep=',')
du.head()

do = pd.read_csv("DisplayOn.csv", header = None)
do.head()
column_names = ['id', 'flag'] # Replace with your desired column names
do.columns = column_names

da = pd.read_csv("AppUsage.csv", header = None)
da.head()
column_names = ['id','app','text'] # Replace with your desired column names
da.columns = column_names
da['text'] = da['text'].str.lower()

da['Facebook'] = da['text'].str.extract(r'(facebook)')
# da['Facebook messenger'] = da['text'].str.extract(r'(????)')
da['Instagram'] = da['text'].str.extract(r'(instagram)')
da['Twitter'] = da['text'].str.extract(r'(twitter)')
da['Snapchat'] = da['text'].str.extract(r'(snapchat)')
da['TikTok'] = da['text'].str.extract(r'(tiktok)')
da['YouTuhe'] = da['text'].str.extract(r'(youtube)')
```

Python and Spyder

The screenshot shows the Spyder Python IDE interface. The top menu bar includes File, Edit, Search, Run, Debug, Consoles, Projects, Tools, View, and Help. The toolbar below has icons for file operations like Open, Save, Run, and Stop. The current workspace path is Z:\UpWork\UpWork2023\Andrew_Bayes.

The left pane displays three open files: temp.py (marked with a yellow triangle), Hao.py (marked with an orange triangle), and IRT_Andrew.py. The IRT_Andrew.py file contains the following code:

```
# -*- coding: utf-8 -*-
"""
Created on Fri Jul 21 18:30:39 2023

@author: Rolando
"""

from IPython import get_ipython
get_ipython().magic('reset -sf')

db_in = 'Z:\\UpWork\\UpWork2023\\Andrew_Bayes'
import os
os.getcwd()
os.chdir(db_in)

# data handling libs
import numpy as np
import pandas as pd

# stan modeling lib
import cmdstanpy
cmdstanpy.install_cmdstan()
cmdstanpy.install_cmdstan(compiler=True) # only valid on Windows
from cmdstanpy import CmdStanModel

# plotting libs
import matplotlib.pyplot as plt
import plotnine as p9
import plotly.express as px
import plotly.graph_objects as go
import plotly.io as pioy

## Loading data
dentist_data = pd.read_csv("caries.csv", skiprows=3)
dentist_data.head()

## Defining parameters of the MCMC and functions
# dictionary to store different models
model = {'sbc':{}, 'ppc':{}, 'cv':{}}
```

The right pane features a Variable Explorer showing the following variables:

Name	Type	Size	Value
db_in	str	33	Z:\UpWork\UpWork2023\Andrew_Bayes
dentist_data	DataFrame	(19345, 3)	Column names: coder, item, response
I	int	1	3869
J	int	1	5
model	dict	3	{'sbc':{}, 'ppc':{}, 'cv':{}}

The bottom pane is the IPython Console, which shows the following output:

```
...: num_draws = NUM_POSTERIOR_SAMPLES/THINNING_GAP*CHAINS
...: def run_stan(model, data, show_console=True):
...:     # generic function to run stan model given a set of config parameters
...:
...:     fit = model.sample(data = data,
...:                         chains = CHAINS,
...:                         iter_warmup = WARMUP_SAMPLES,
...:                         iter_sampling = NUM_POSTERIOR_SAMPLES,
...:                         thin = THINNING_GAP,
...:                         show_progress = False,
...:                         show_console = show_console)
...:
...:     return fit
...:
...: # SBC config variables
...: NUM_SIMS = 100
Installing CmdStan version: 2.32.2
Install directory: C:\Users\Rolando\.cmdstan
CmdStan version 2.32.2 already installed
23:25:19 - cmdstanpy - INFO - Add C++ toolchain to $PATH: C:\Users\Rolando\.cmdstan\RTools40
Installing CmdStan version: 2.32.2
```

The status bar at the bottom indicates LSP Python: ready, conda: base (Python 3.9.13), Line 24, Col 1, UTF-8, CRLF, RW, Mem 59%.

Python and Jupyter

Jupyter Notebooks are interactive computational documents that seamlessly combine code, text, equations, visualizations, and outputs, all in a browser-based environment.



https://jupyter.org/try-jupyter/lab/

File Edit View Run Kernel Tabs Settings Help

+ / notebooks /

Name	Modified
Intro.ipynb	3d ago
Lorenz.ipynb	3d ago
r.ipynb	3d ago
sqlite.ipynb	3d ago

Intro.ipynb

+

Markdown

Introduction to the JupyterLab and JupyterNotebooks

This is a short introduction to two of the flagship tools created by the Jupyter Project: JupyterLab and JupyterNotebooks.

⚠ Experimental! ⚠: This is an experimental interface provided by the Jupyter Project. It is designed to work with popular packages for scientific computing, in your browser. There is no need to install locally. You may also encounter some bugs or unexpected behavior. Please report issues to the [JupyterLite repository](#).

JupyterLab

JupyterLab is a next-generation web-based user interface for Project Jupyter. It integrates code editors, terminals, and custom components in a flexible, integrated, and extensible way.

For an overview of the JupyterLab interface, see the [JupyterLab Web Site](#). To learn more about the JupyterLab API, see the [API Reference](#).

See Also: For a more in-depth tour of JupyterLab with a full environment, see the [JupyterLab Tour](#).

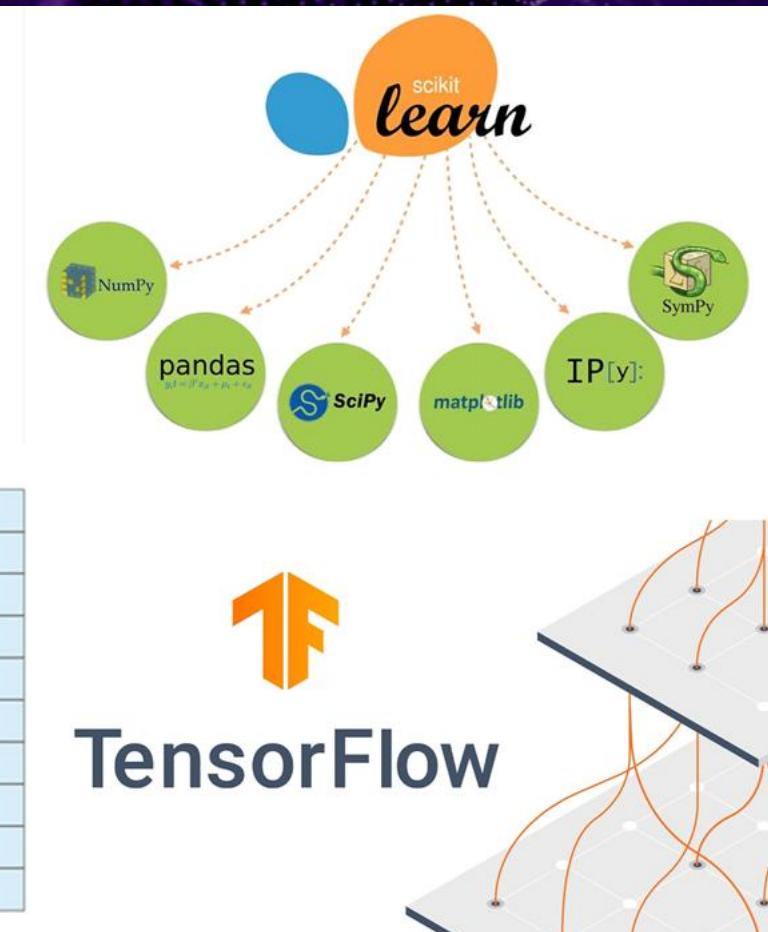
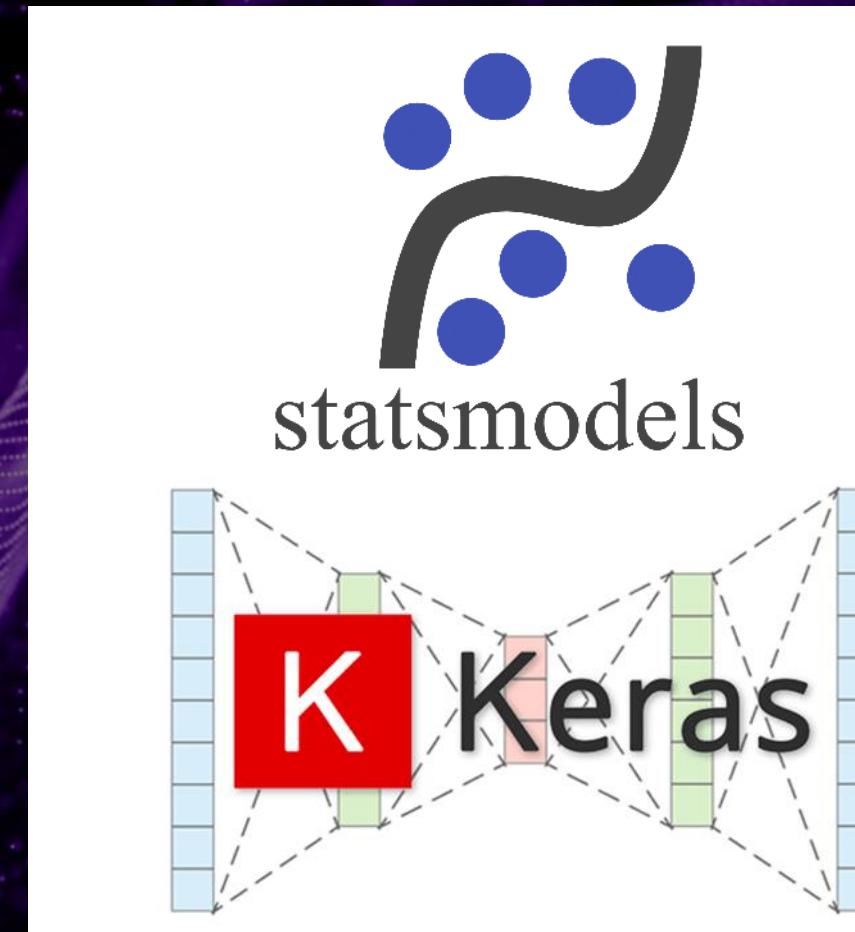
Jupyter Notebooks

Jupyter Notebooks are a community standard for communicating and sharing data science work. They are used in many fields, including data analysis, machine learning, and scientific computing.

Python

Specialized libraries/packages:

- Statsmodels
- Numpy
- Pandas
- Matplotlib
- Scikit-learn
- Tensorflow
- Keras



Python

Scikit-learn

<https://scikit-learn.org/stable/>

- Classification
- Regression
- Clustering
- Dimensionality Reduction
- Model selection
- Data preprocessing (data preparation)

scikit-learn

Machine Learning in Python

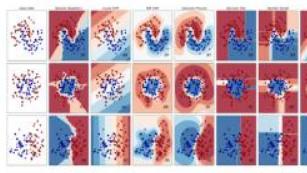
Getting Started Release Highlights for 1.3 GitHub

- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.
Algorithms: Gradient boosting, nearest neighbors, random forest, logistic regression, and more...

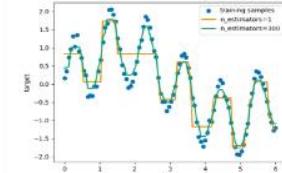


[Examples](#)

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: Gradient boosting, nearest neighbors, random forest, ridge, and more...

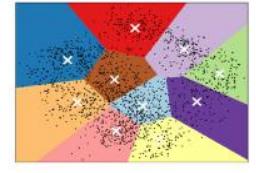


[Examples](#)

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, HDBSCAN, hierarchical clustering, and more...

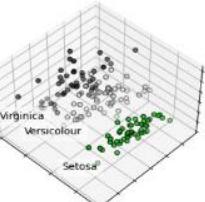


[Examples](#)

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency
Algorithms: PCA, feature selection, non-negative matrix factorization, and more...

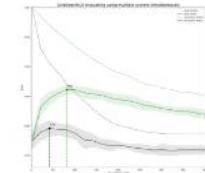


[Examples](#)

Model selection

Comparing, validating and choosing parameters and models.

Applications: Improved accuracy via parameter tuning
Algorithms: grid search, cross validation, metrics, and more...

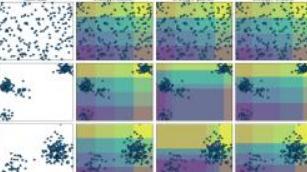


[Examples](#)

Preprocessing

Feature extraction and normalization.

Applications: Transforming input data such as text for use with machine learning algorithms.
Algorithms: preprocessing, feature extraction, and more...



[Examples](#)

Python and Anaconda

A screenshot of a web browser displaying the Anaconda website. The URL 'anaconda.com' is visible in the address bar. The page features a dark header with the Anaconda logo and navigation links for Products, Solutions, Resources, and Company. A prominent call-to-action section in the center reads 'Introducing The Anaconda AI Platform for secure open source development and governance' followed by a large, bold title 'Advance AI with Clarity and Confidence'. Below this, a subtitle says 'Simplify, safeguard, and accelerate AI value with open source.' Two buttons at the bottom are 'Sign Up for Free' and 'Get a Demo'.

Advance AI with Open Source | An X

anaconda.com

ANACONDA. Products Solutions Resources Company

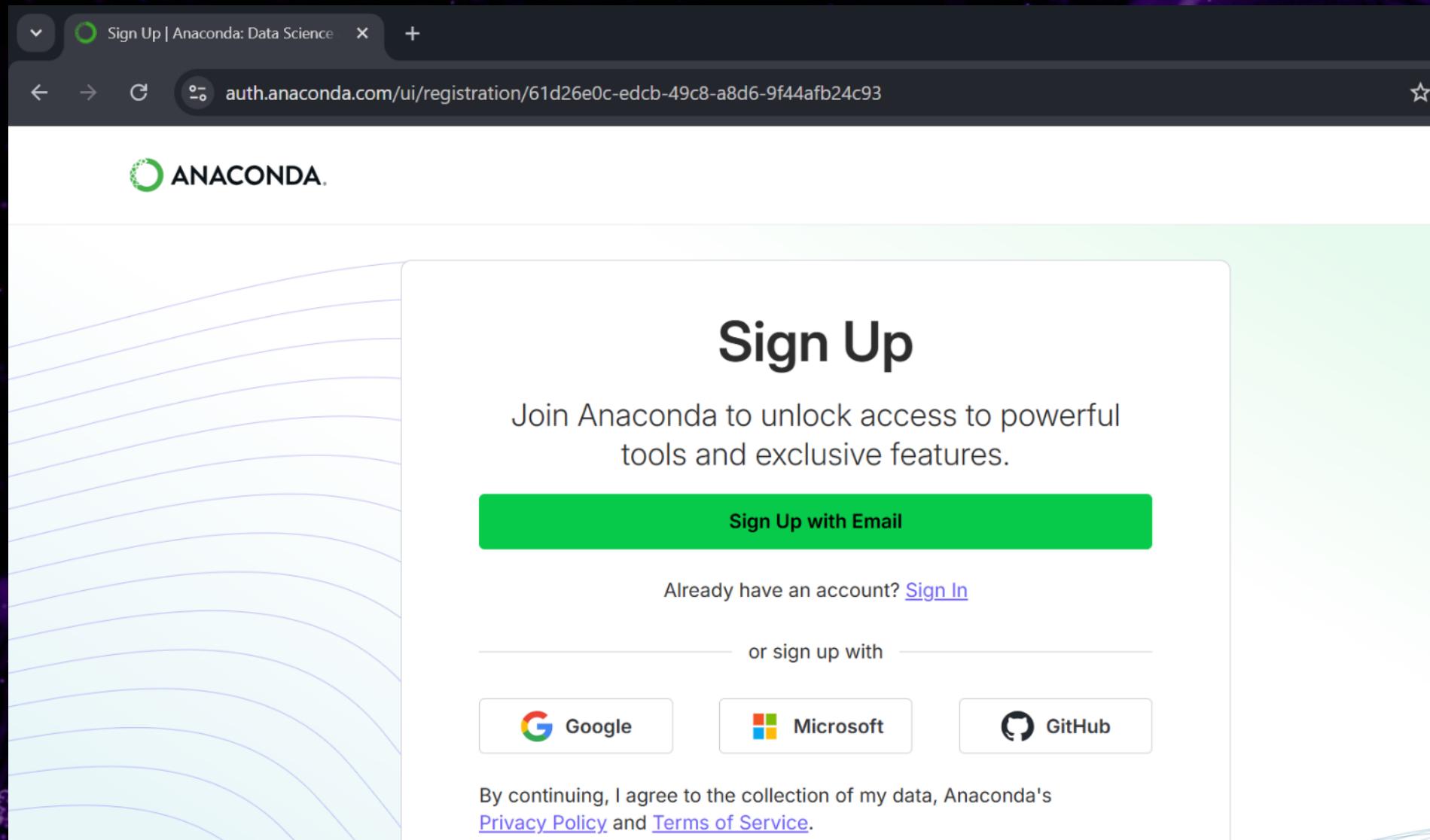
Introducing The Anaconda AI Platform for secure open source development and governance >

Advance AI with Clarity and Confidence

Simplify, safeguard, and accelerate AI value with open source.

Sign Up for Free > Get a Demo >

Python and Anaconda



Python and Anaconda

Create New Project

Project Title *

 11 / 120

Environment Location:

anaconda-2022.05-py39



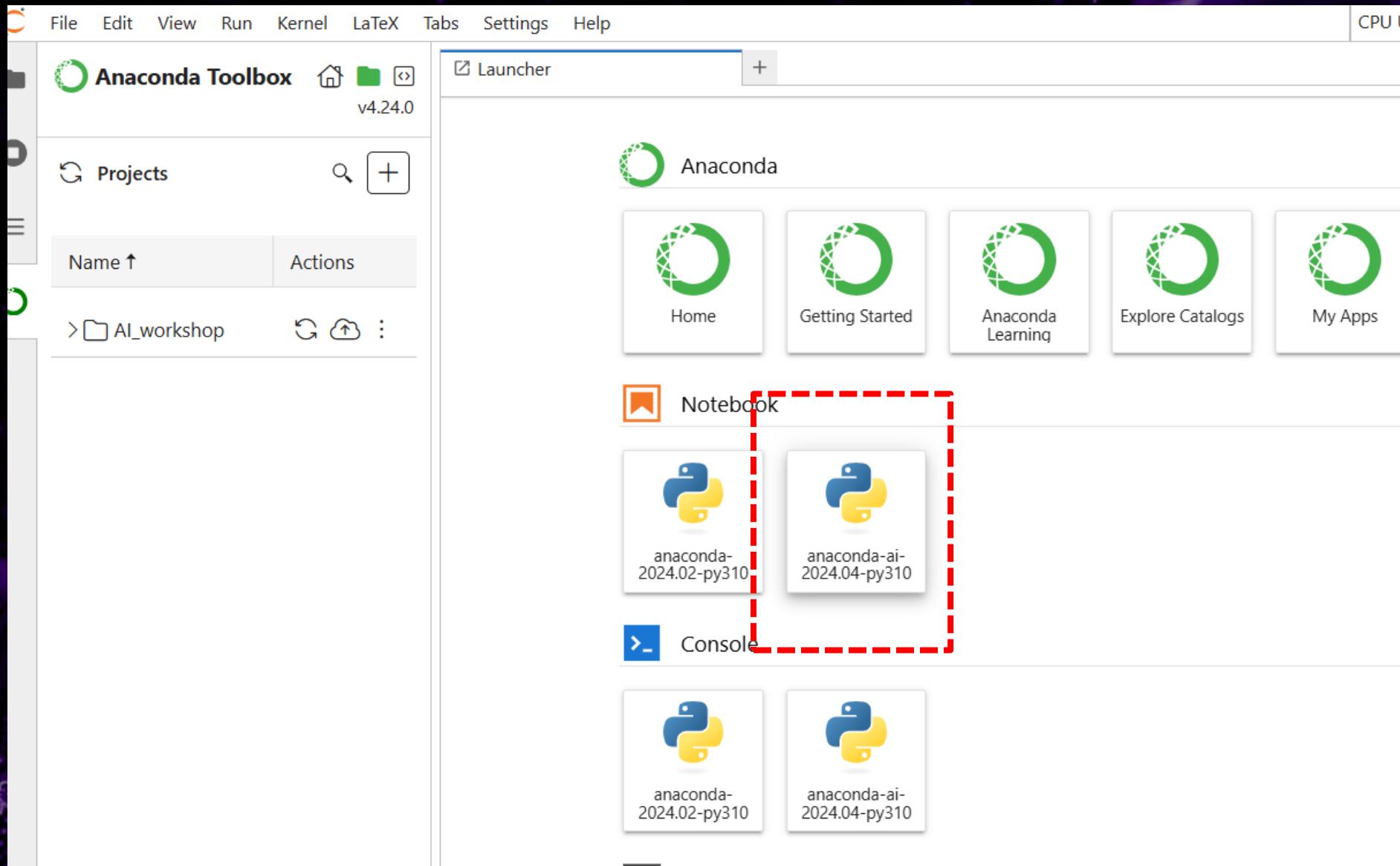
Upload Files

Transfer Files



Drag and drop files here to upload.

Python and Anaconda



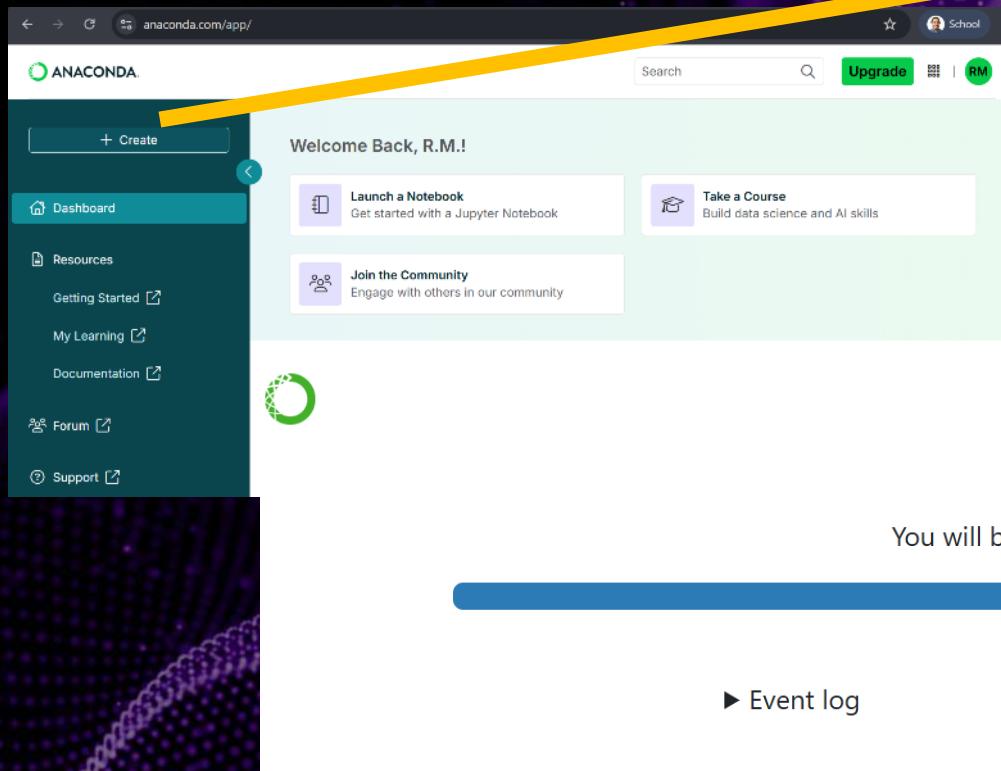
Python and Anaconda

The screenshot shows the Anaconda application interface. At the top, there's a header bar with a back arrow, forward arrow, refresh icon, and a URL field containing "anaconda.com/app/". To the right of the URL is a star icon, a user profile icon labeled "School", and a green "Upgrade" button. Below the header is the Anaconda logo and a search bar with a magnifying glass icon.

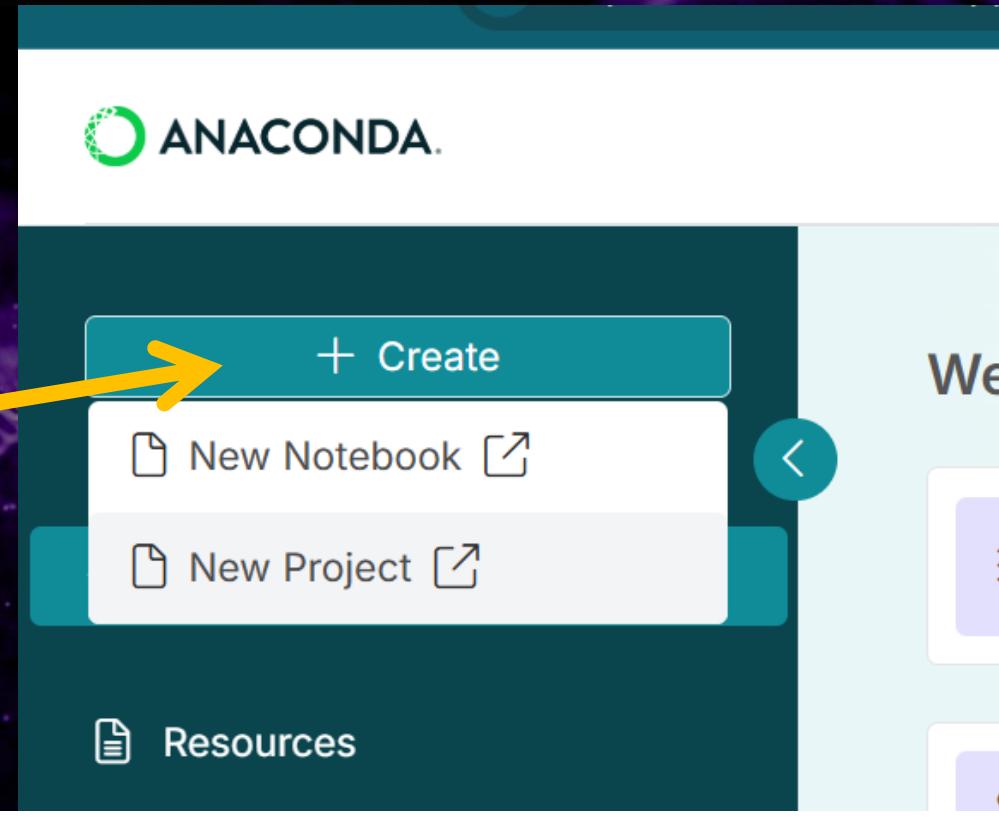
The main area is divided into sections. On the left, a dark sidebar contains a "Dashboard" button (which is highlighted in teal), "Resources", "Getting Started", "My Learning", "Documentation", "Forum", and "Support". Above the sidebar is a teal button labeled "+ Create".

In the center, a welcome message reads "Welcome Back, R.M.!". Below it are three cards: "Launch a Notebook" (with a notebook icon), "Take a Course" (with a graduation cap icon), and "Join the Community" (with a people icon). Further down, a section titled "Explore Anaconda" features four colored cards with icons: a yellow one with a smiley face, a white one with a blue paperclip, a green one with a globe, and a blue one with a document icon.

Python and Anaconda



The screenshot shows the Anaconda application interface. On the left is a sidebar with options like Dashboard, Resources, Getting Started, My Learning, Documentation, Forum, and Support. The main area has a teal header with the Anaconda logo and a search bar. Below the header are three cards: 'Launch a Notebook' (Get started with a Jupyter Notebook), 'Take a Course' (Build data science and AI skills), and 'Join the Community' (Engage with others in our community). A large green button labeled '+ Create' is positioned at the top right of the main area. A yellow arrow points from the text 'Click here to start' to this '+ Create' button.



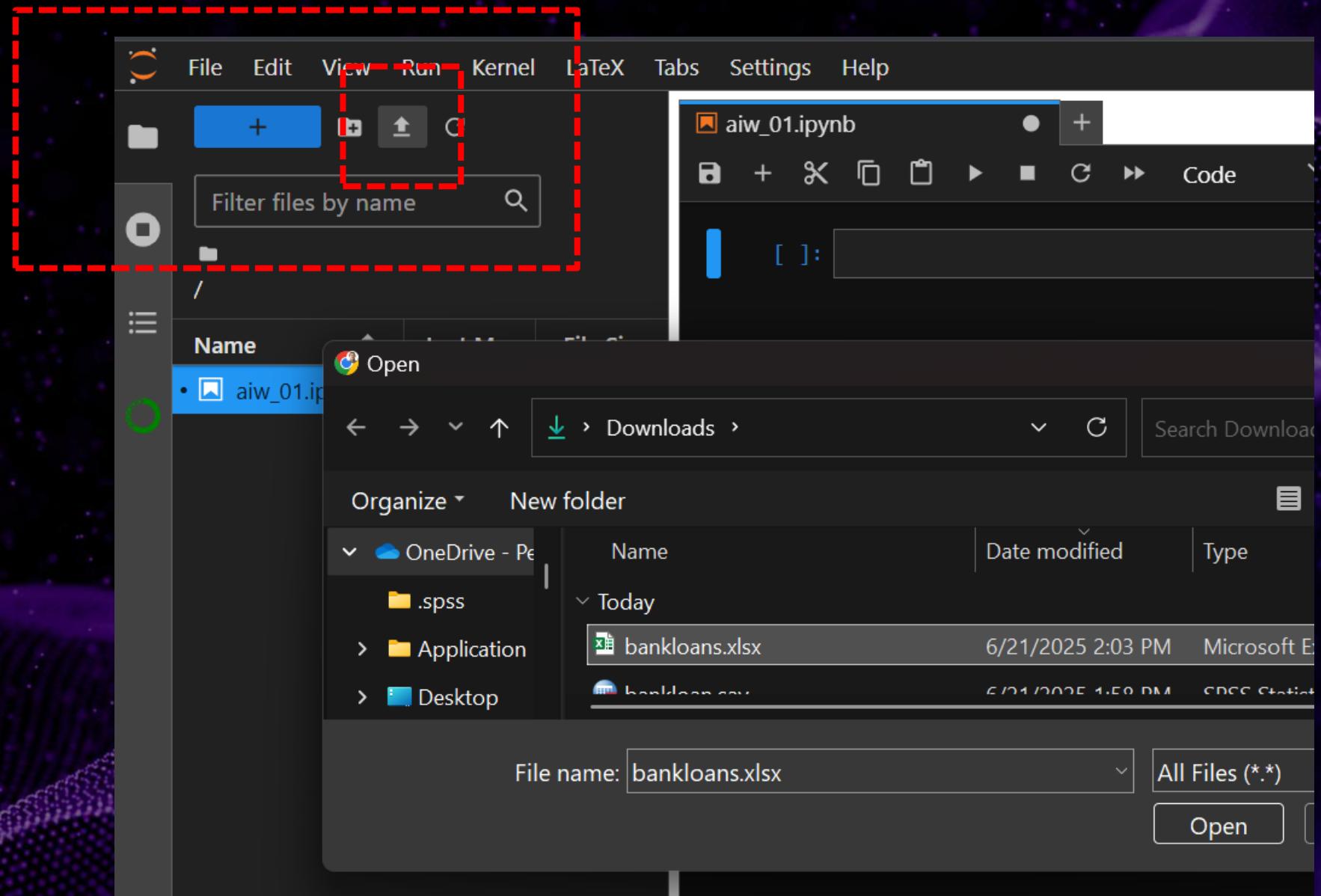
Your server is starting up.

You will be redirected automatically when it's ready for you.

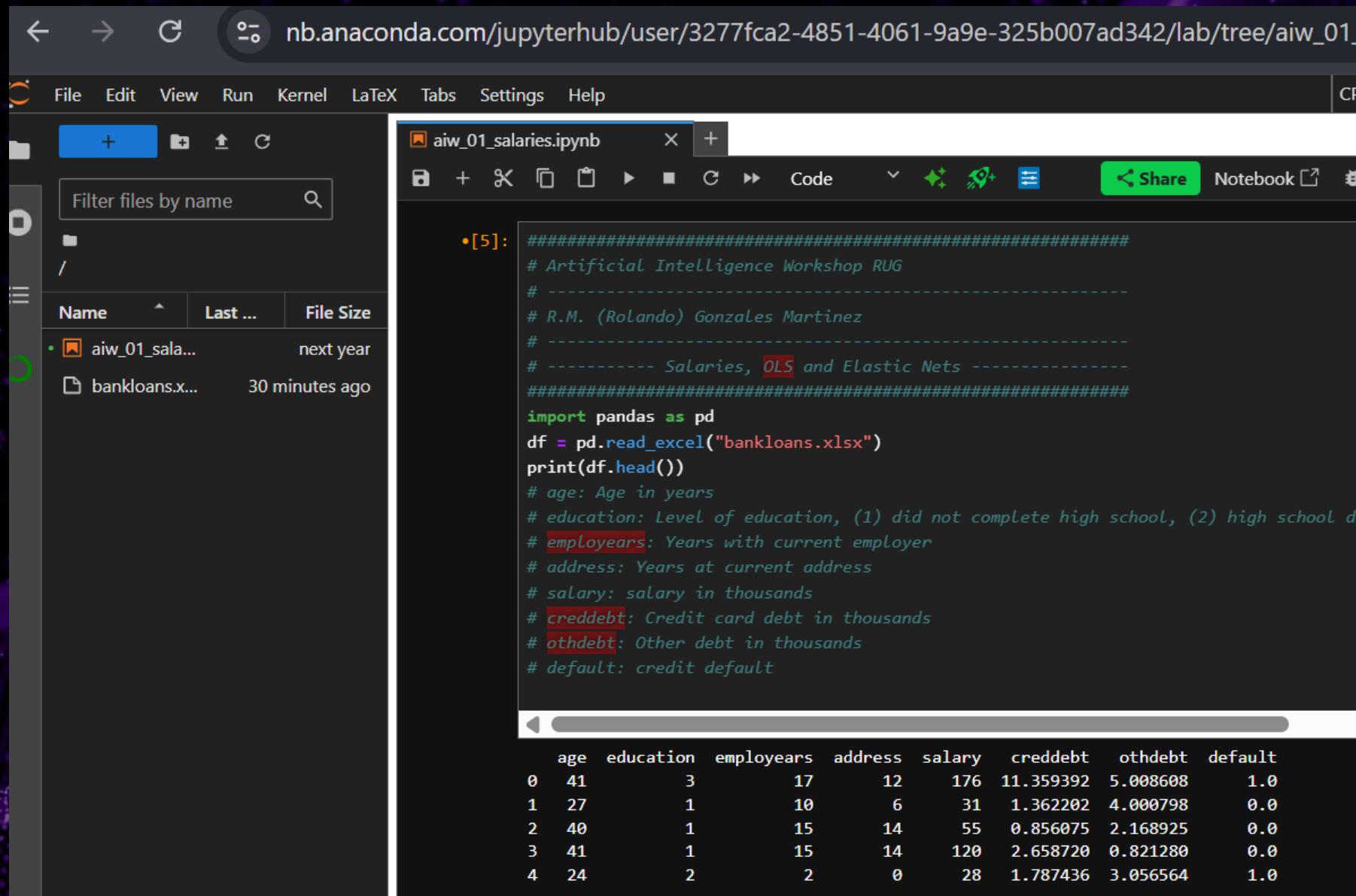
Spawning server...

▶ Event log

Python and Anaconda



Python and Anaconda

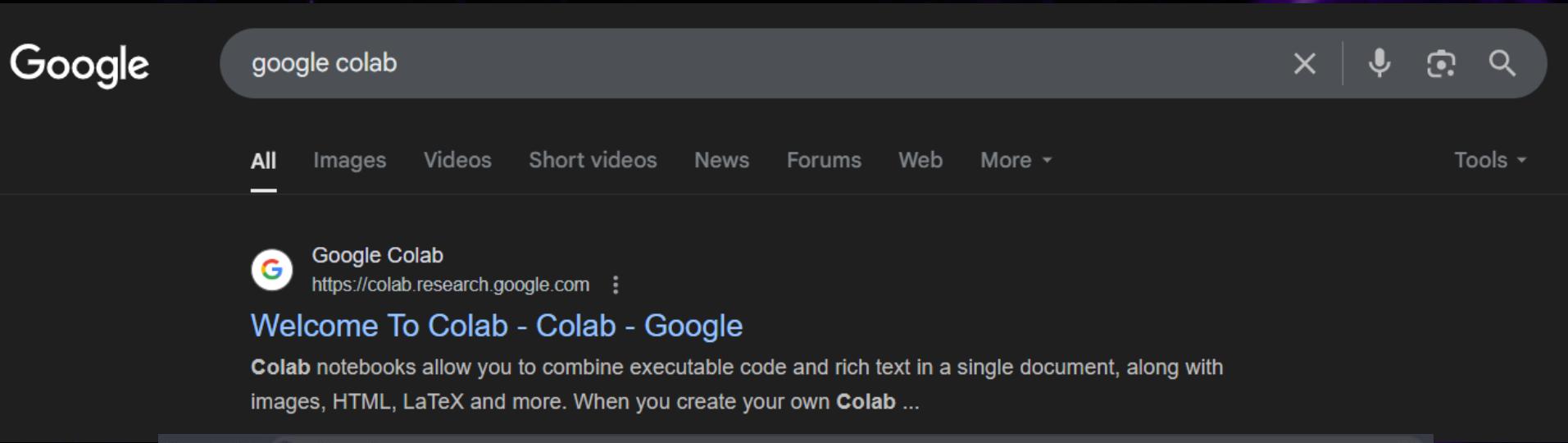


The screenshot shows a Jupyter Notebook interface running on nb.anaconda.com. The notebook is titled "aiw_01_salaries.ipynb". The code cell at index 5 contains Python code for reading a bank loans dataset and printing its head. The output below the code cell displays the first five rows of the dataset.

```
# Artificial Intelligence Workshop RUG
# -----
# R.M. (Rolando) Gonzales Martinez
# -----
# ----- Salaries, OLS and Elastic Nets -----
#####
import pandas as pd
df = pd.read_excel("bankloans.xlsx")
print(df.head())
# age: Age in years
# education: Level of education, (1) did not complete high school, (2) high school d
# employyears: Years with current employer
# address: Years at current address
# salary: salary in thousands
# creddebt: Credit card debt in thousands
# othdebt: Other debt in thousands
# default: credit default
```

	age	education	employyears	address	salary	creddebt	othdebt	default
0	41		3	17	12	176	11.359392	5.008608
1	27		1	10	6	31	1.362202	4.000798
2	40		1	15	14	55	0.856075	2.168925
3	41		1	15	14	120	2.658720	0.821280
4	24		2	2	0	28	1.787436	3.056564

Python and Google Colab

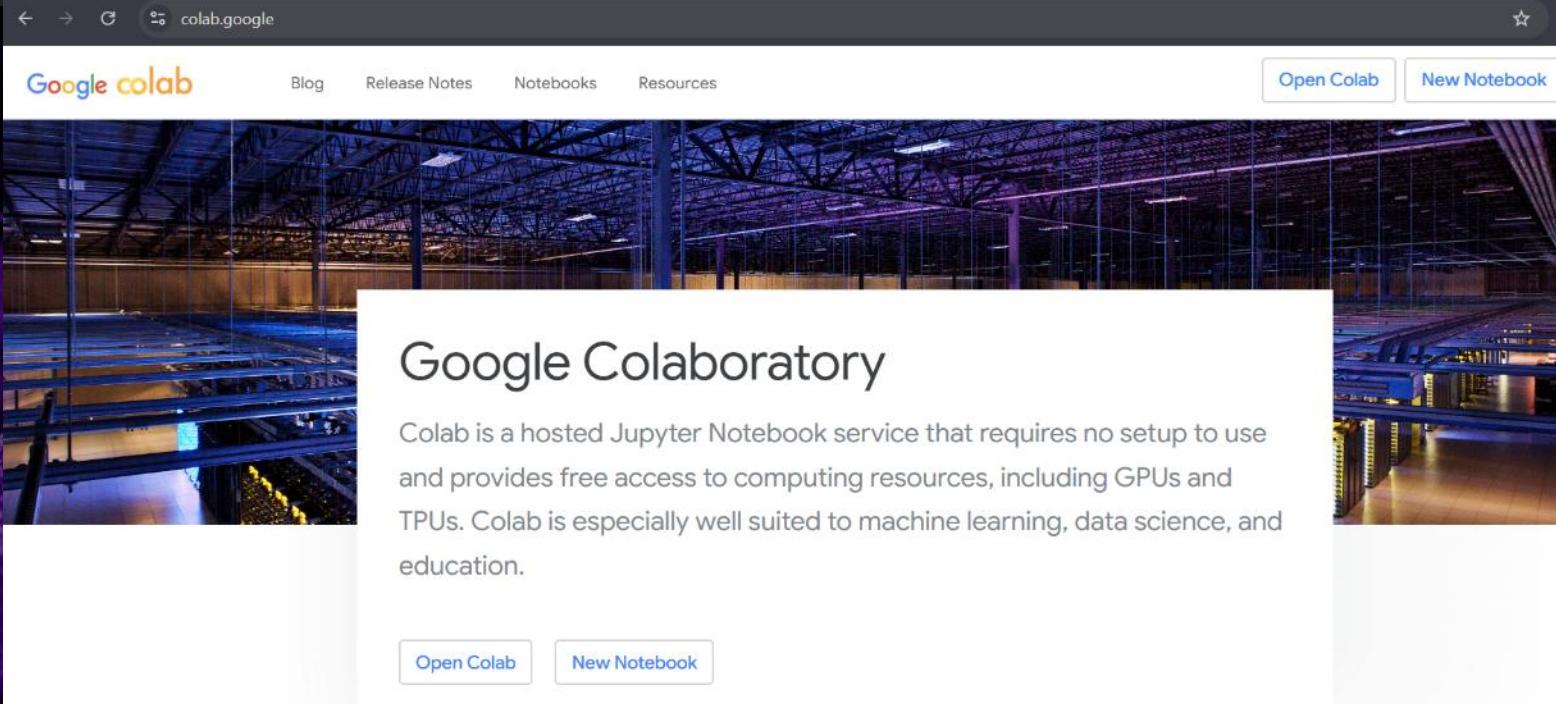


A screenshot of a Google search results page. The search bar at the top contains the query "google colab". Below the search bar, there are several navigation links: All, Images, Videos, Short videos, News, Forums, Web, More, and Tools. The first result is a link to the Google Colab homepage, which has a thumbnail image of a server room and the title "Google Colaboratory". The snippet below the title describes Colab as a hosted Jupyter Notebook service.

Google Colab
https://colab.research.google.com

Welcome To Colab - Colab - Google

Colab notebooks allow you to combine executable code and rich text in a single document, along with images, HTML, LaTeX and more. When you create your own Colab ...



The homepage of Google Colab. At the top, there is a navigation bar with links to Blog, Release Notes, Notebooks, and Resources. On the right side of the header are two buttons: "Open Colab" and "New Notebook". The main content area features a large image of a server room with many racks of equipment. Overlaid on this image is the title "Google Colaboratory". Below the title, a paragraph of text explains what Colab is: "Colab is a hosted Jupyter Notebook service that requires no setup to use and provides free access to computing resources, including GPUs and TPUs. Colab is especially well suited to machine learning, data science, and education." At the bottom of the page are two buttons: "Open Colab" and "New Notebook".

Google colab

Blog Release Notes Notebooks Resources

Open Colab New Notebook

Google Colaboratory

Colab is a hosted Jupyter Notebook service that requires no setup to use and provides free access to computing resources, including GPUs and TPUs. Colab is especially well suited to machine learning, data science, and education.

Open Colab New Notebook

Python and Google Colab

The screenshot shows the Google Colab interface at colab.research.google.com. A red dashed box highlights the sidebar and the main content area where the Gemini API integration is displayed.

Welcome To Colab

File Edit View Insert Runtime Tools Help

Commands + Code + Text ▶ Run all Copy to Drive

Table of contents

- Welcome to Colab!
- Getting started
- Data science
- Machine learning
- More Resources
- Featured examples
- + Section

Open notebook

- Examples >
- Recent** > (highlighted)
- Google Drive >
- Github >
- Upload >

Welcome to Explore the Gemini API gives you multimodal, so you can...

How to get started?

- Go to [Google AI Studio](#)
- Create an API key
- Use a quickstart file

Discover Gemini's advanced features

- Play with Gemini's image generation
- Discover the [multimodal API](#)
- Learn how to [analyze text with Gemini](#)
- Unlock the power of Gemini's AI

Explore complex use cases

- Use Gemini groups
- Extract invoices and receipts
- Create illustrations from text descriptions

To learn more, check out the [Gemini documentation](#).

Colab now has AI features for generating code, explaining code, and translating code between Python, or a seasoned developer.

+ New notebook

Python and Google Colab

The screenshot shows the Google Colab interface. The top navigation bar includes the Colab logo, file name "Untitled2.ipynb", and standard menu options: File, Edit, View, Insert, Runtime, Tools, Help. Below the menu is a toolbar with "Commands", "+ Code", "+ Text", and "Run all". A red dashed box highlights the "Files" sidebar on the left, which contains icons for upload, download, search, and folder operations, along with a "sample_data" folder and a "bankloans.xlsx" file. The main workspace contains Python code for reading an Excel file and displaying its head. A preview of the data frame is shown at the bottom.

```
#####
# Artificial Intelligence Workshop RUG
# -----
# R.M. (Rolando) Gonzales Martinez
# -----
# ~ ~ ~ ~ ~ Salaries ~ ~ ~ ~ ~ ~ ~ ~ ~
#####

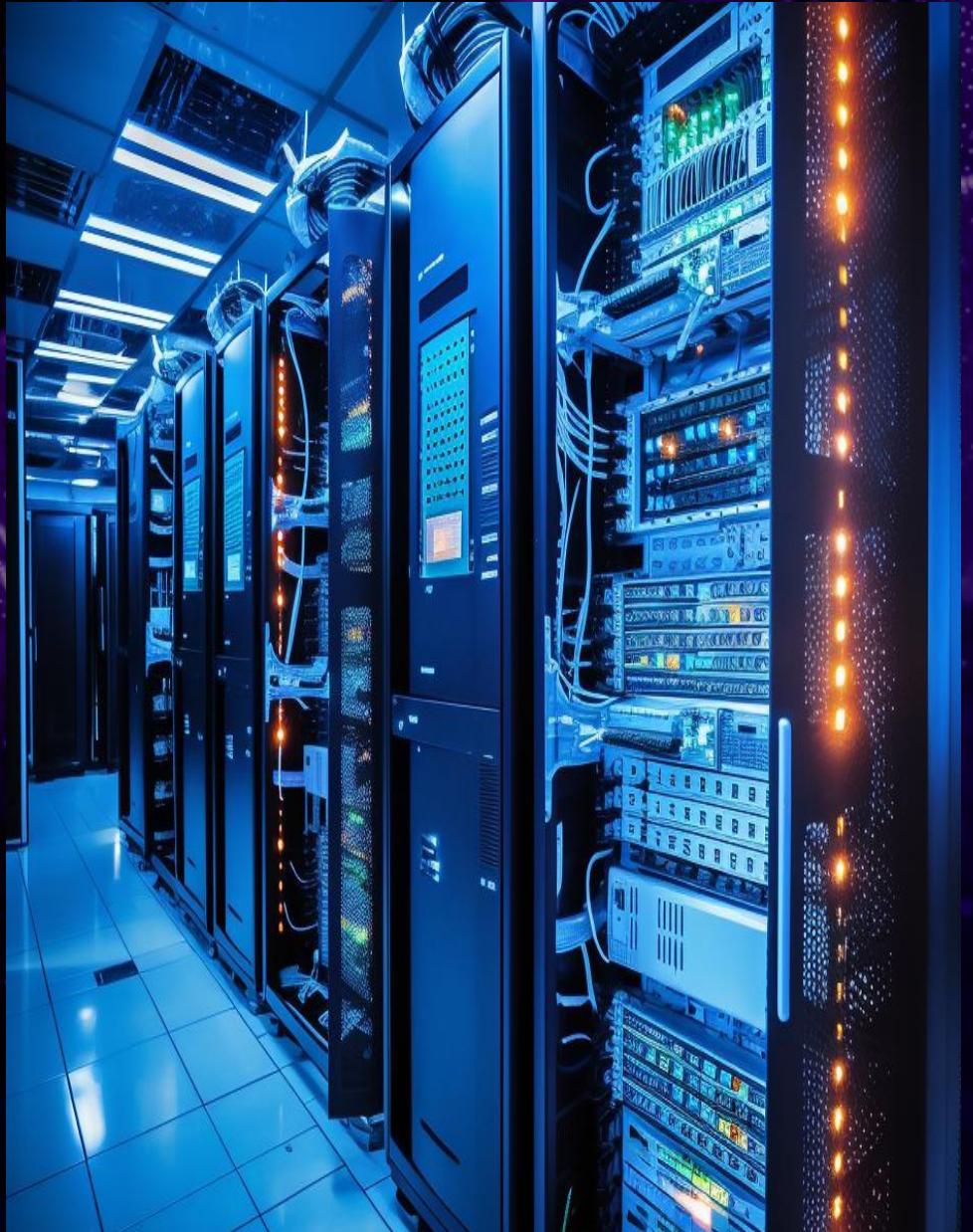
import pandas as pd
df = pd.read_excel("bankloans.xlsx")
print(df.head())
# age: Age in years
# education: Level of education, (1) did not complete high school, (2) high school
# employears: Years with current employer
# address: Years at current address
# salary: salary in thousands
# creddebt: Credit card debt in thousands
# othdebt: Other debt in thousands
# default: credit default
```

	age	education	employears	salary	creddebt	othdebt	default
0	41	3	17	176	11.359392	5.008608	1
1	27	1	10	31	1.362202	4.000798	0
2	40	1	15	55	0.856075	2.168925	0
3	41	1	15	120	2.658720	0.821280	0
4	24	2	2	28	1.787436	3.056564	1

Big data

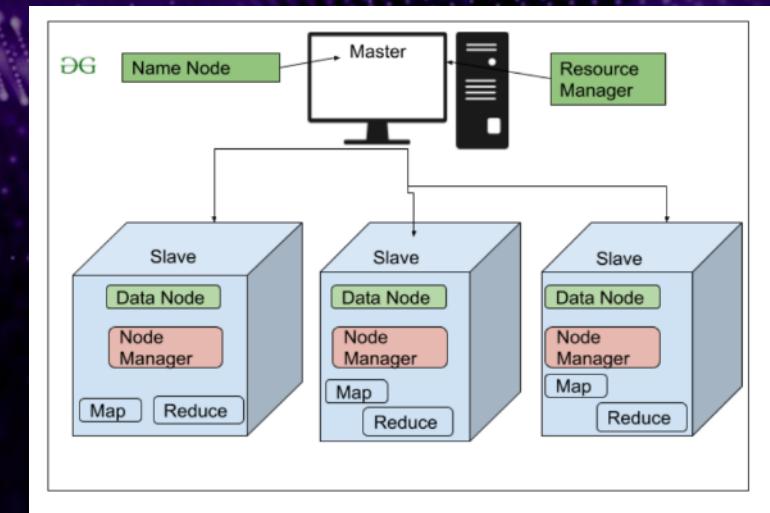
3 V's:

- **Volume:** Terabytes, petabytes, or even exabytes that cannot be managed or processed by conventional systems (sensors, social media, transaction logs).
- **Velocity:** generated at an incredibly fast pace, in real time or at very short intervals.
- **Variety:** Big data can present a wide range of formats and types, such as text, images, audio, video, geospatial data, structured, and unstructured data.



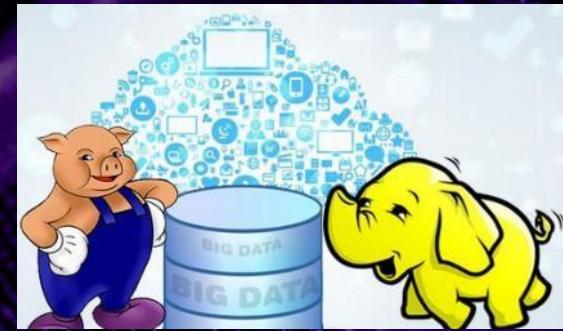
Distributed storage and processing of large datasets

- **Apache Hadoop:** a system designed to store large amounts of data in clusters of servers. It divides files into blocks and distributes them across the nodes of the cluster.
- **Master-slave structure:** The NameNode acts as the master and is responsible for maintaining the metadata of the file system, including file names, block locations, and permissions. The DataNodes are the slave nodes that store the actual data blocks.



Hadoop Ecosystem

- **Hive:** a data storage and query system that allows SQL-like queries on data stored in a distributed manner (HDFS).
- **Pig:** a language for performing data transformations and analysis in Hadoop.
- **Spark:** an in-memory data processing system that is faster than MapReduce for certain types of tasks.
- **YARN:** a resource manager that enables multiple applications to run efficiently on the same cluster.

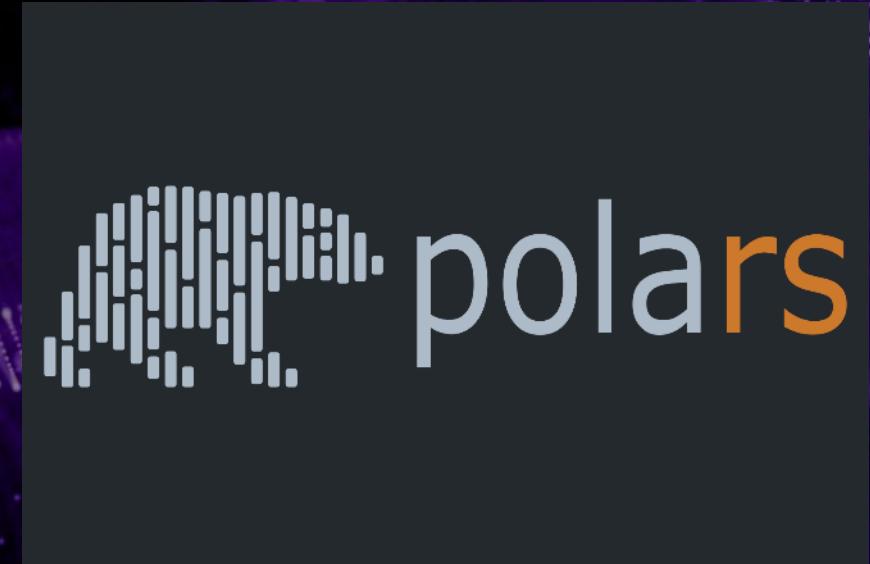


VS



More accessible options

- Parallel computing (CPU)
- GPU, TPU
- Cloud computing (CPU, GPU)
- Use of libraries like Polars



Laboratory: Python scripts

Python scripts:

- `aiw_00_elizaAI.ipynb`: AI-based psychological counseling system
- `aiw_01_salaries.ipynb`: Python script for OLS regression and Elastic Nets
- `aiw_02_credit.ipynb`: Python script for credit scoring
- `aiw_03_creditlm.ipynb`: Python script for credit scoring with machine learning logic

AI as psychological assistants & therapists

- Issue: increasing demand for psychotherapy and limited access to specialists
Goal: emulating professional therapists' conversational styles to provide cognitive, social, and emotional support



Fig. 1. An example of Ryan the CompanionBot interacting with a user in the experimental setup.

Socially Assistive Robotics (SAR)

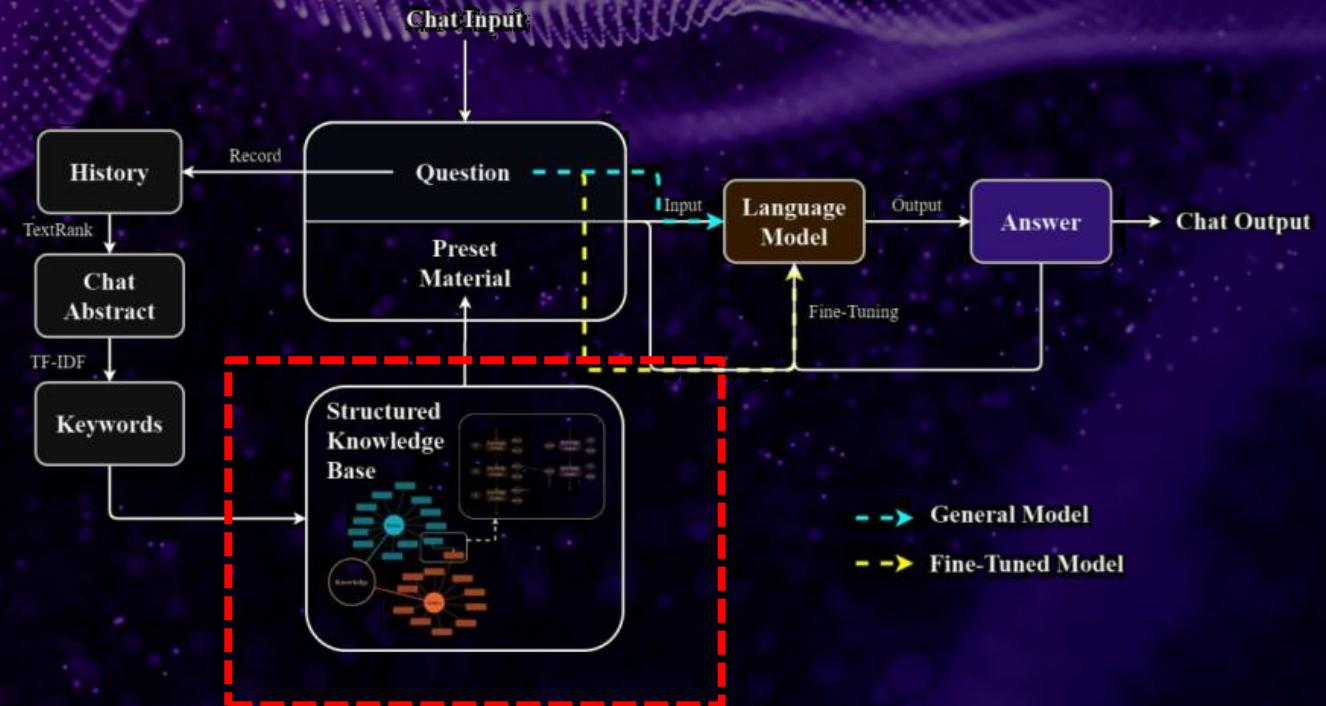
Dino, F., Zandie, R., Abdollahi, H., Schoeder, S., & Mahoor, M. H.. Delivering cognitive behavioral therapy using a conversational social robot. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 2089-2095). IEEE.

AI as psychological assistants & therapists

- Issue: increasing demand for psychotherapy and limited access to specialists
Goal: emulating professional therapists' conversational styles to provide cognitive, social, and emotional support

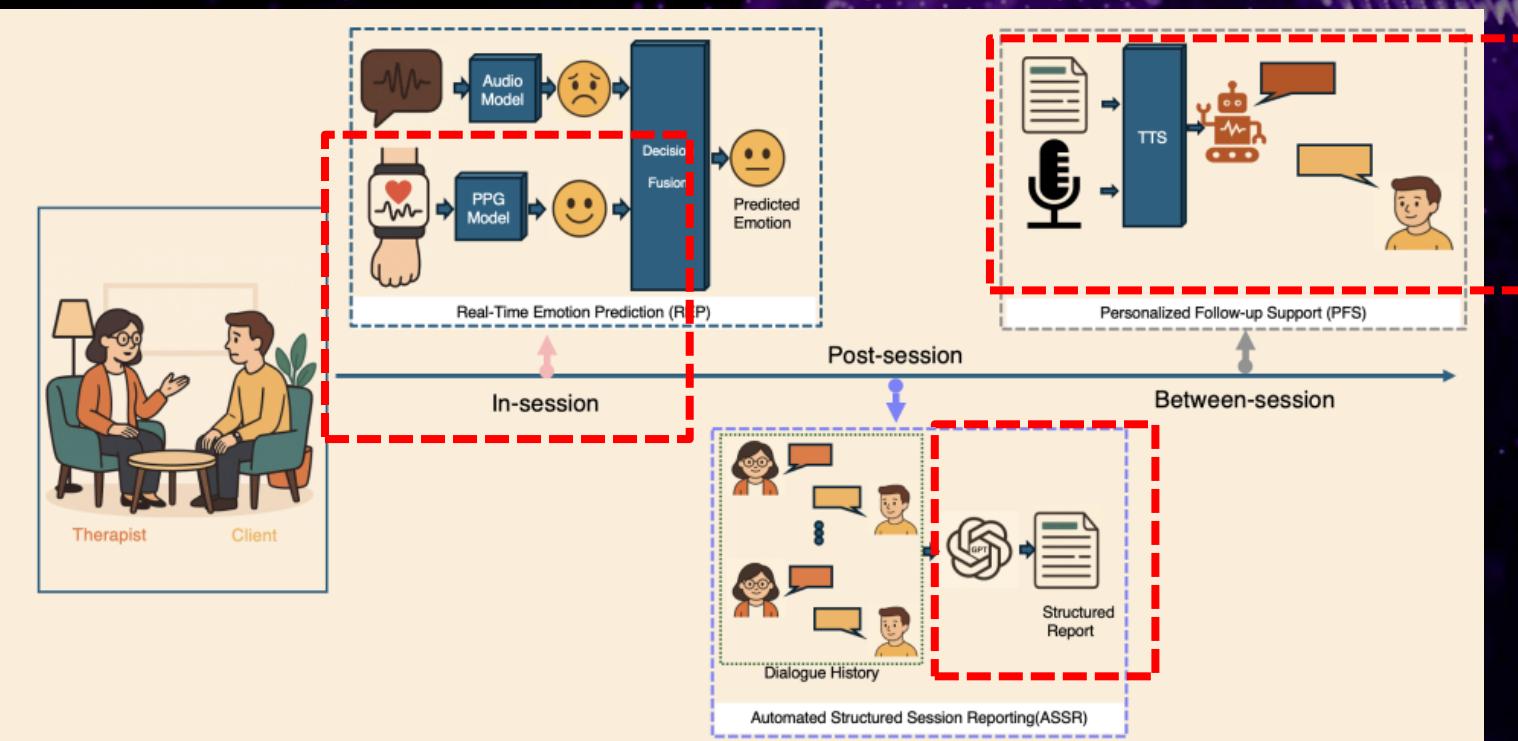


Zhang, H., Qiao, Z., Wang, H., Duan, B., & Yin, J. (2024). VCounselor: a psychological intervention chat agent based on a knowledge-enhanced large language model. *Multimedia Systems*, 30(6), 363.



AI as psychological assistants & therapists

- Issue: increasing demand for psychotherapy and limited access to specialists
Goal: emulating professional therapists' conversational styles to provide cognitive, social, and emotional support



Liu, X., Xu, J., & Sun, T. (2025). PsyCounAssist: A Full-Cycle AI-Powered Psychological Counseling Assistant System. arXiv preprint arXiv:2504.16573.

Model	Dataset	Accuracy	F1 Score
Random Forest	Validation	0.954	0.957
	Test	0.963	0.964
Gradient Boosting	Validation	0.844	0.858
	Test	0.847	0.858
AdaBoost	Validation	0.711	0.747
	Test	0.717	0.749
Support Vector Machine	Validation	0.559	0.708
	Test	0.553	0.701
Naive Bayes	Validation	0.558	0.707
	Test	0.554	0.702

AI as psychological assistants & therapists: Rogerian psychotherapy

Rogerian psychotherapy is as person-centered, client-centered, or non-directive therapy:

- Non-directive: The therapist listens closely, reflects feelings, clarifies thoughts, but doesn't interpret, advise, or diagnose Provides a supportive, empathic environment,
- Patients gain self-awareness and move toward congruence, bridging the gap between their real and ideal selves

ELIZA was one of the earliest chatbot attempts to embody Rogerian therapy.

- Created in the mid-1960s by Joseph Weizenbaum (MIT), ELIZA employed a script named DOCTOR to mimic a non-directive, reflective Rogerian psychotherapist
- Issue: ELIZA quickly breaks on conversations after some time

Vibe coding to create an ELIZA Rogerian psychotherapy prototype

What is vibe coding?

- New paradigm in programming: intentions are expressed in natural language and AI tools generate the corresponding code.
- Coined by Andrej Karpathy in 2025
- Shifts focus from syntax to creative design
- Makes coding more accessible and intuitive

Vibe coding to create an ELIZA Rogerian psychotherapy prototype

Vibe coding emphasizes the creative 'vibe' of development over traditional rules:

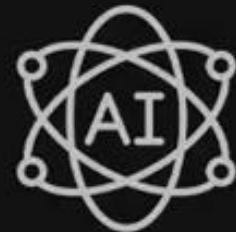
- Encourages human-AI collaboration
- Lowers the barrier for non-programmers
- Focuses on high-level logic and design

Vibe coding to create an ELIZA Rogerian psychotherapy prototype

How to Vibe Code in Python?



Describe what
you want



AI generating
the code



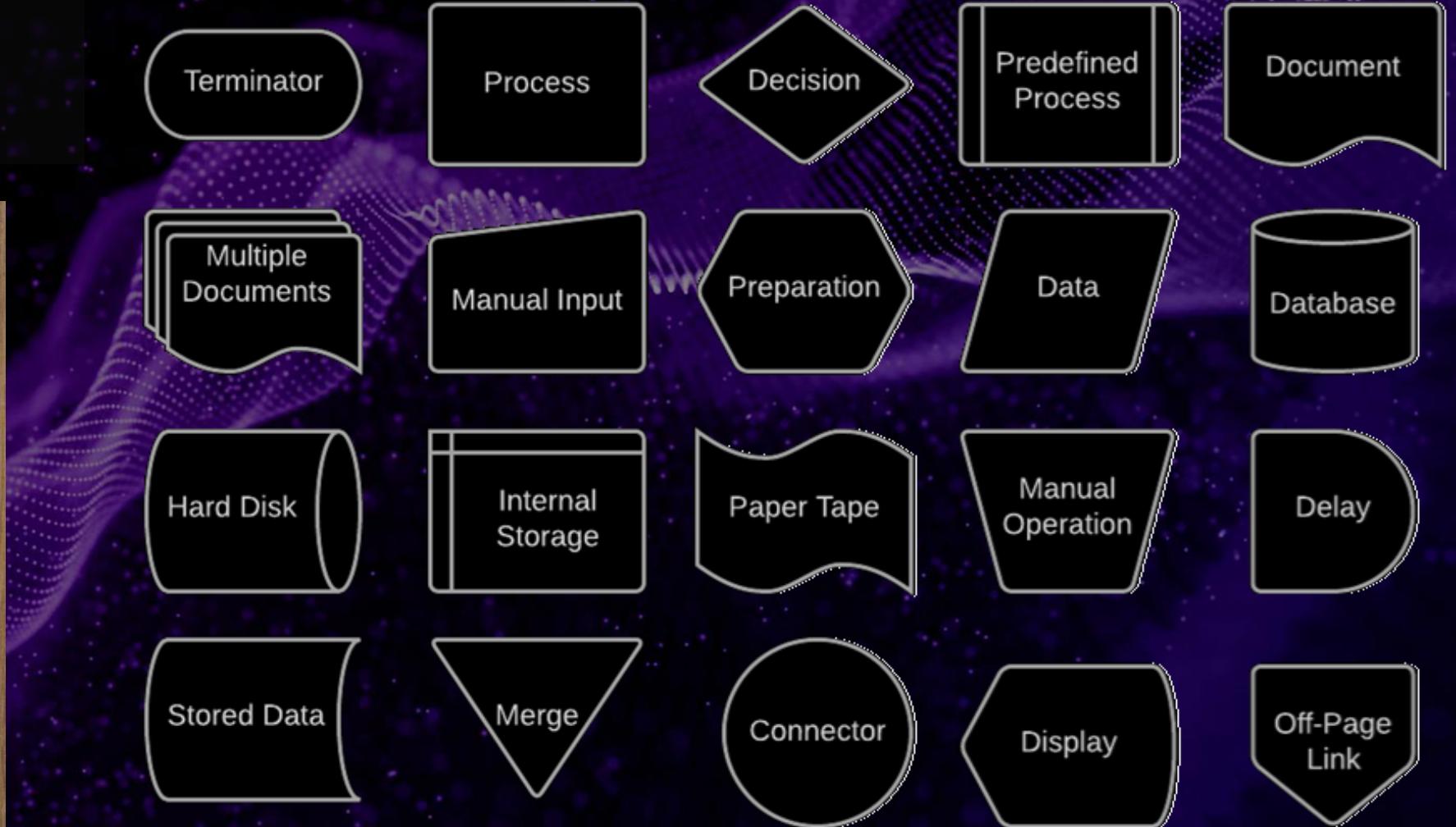
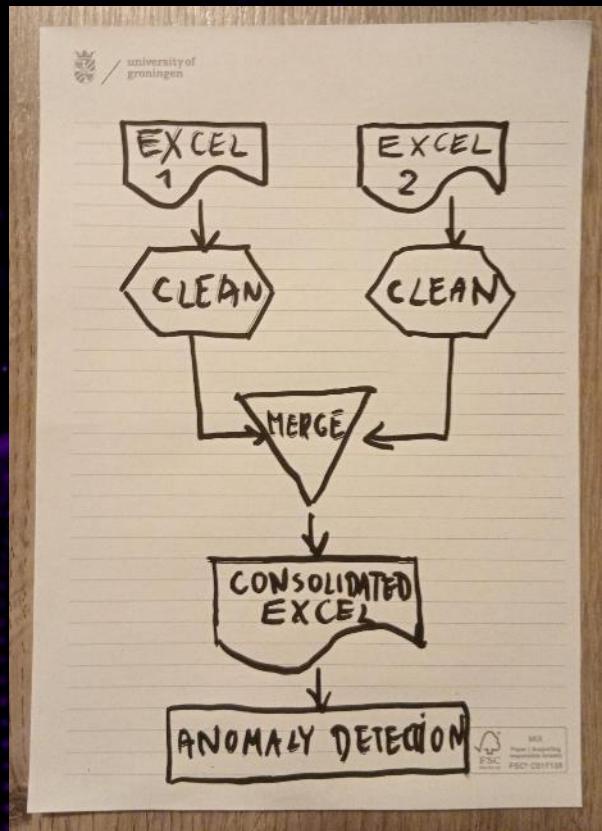
Testing and
refining code



Iterate

Vibe coding with Python in Practice

Automatizing flowchart's design



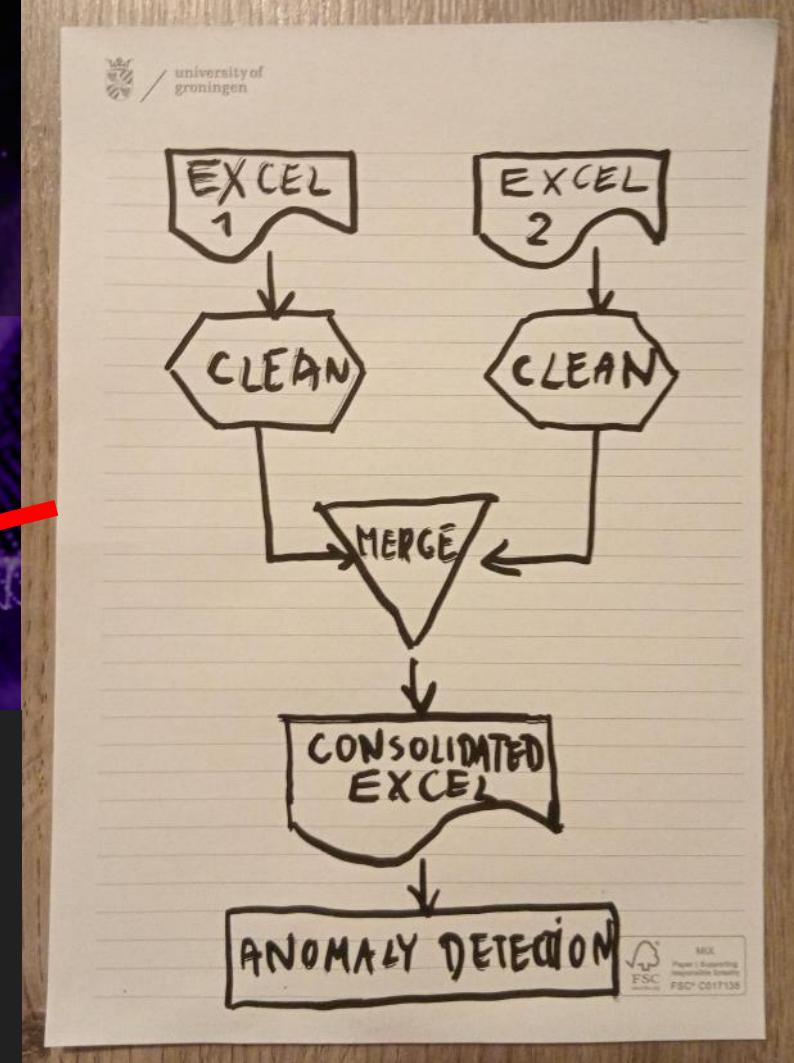
Vibe coding with Python

Automatizing flowchart's design



make a flowchart in Tikz and a script in python based on the figure attached

+ Tools



Vibe coding with Python

Automatizing flowchart's design with ChatGPT

```
latex
\documentclass{article}
\usepackage{tikz}
\usetikzlibrary{shapes.geometric, arrows.meta, positioning}

\tikzstyle{startstop} = [draw, rectangle, rounded corners, minimum width=3cm, minimum height=1cm,
\tikzstyle{process} = [draw, diamond, aspect=2, minimum width=3cm, minimum height=1cm, text center]
\tikzstyle{merge} = [draw, isosceles triangle, isosceles triangle apex angle=60, shape border rotat\tikzstyle{arrow} = [thick, ->, >=stealth]

\begin{document}

\begin{tikzpicture}[node distance=1.7cm and 2.5cm]

% Top row
\node (excel1) [startstop] {Excel 1};
\node (excel2) [startstop, right-of=excel1] {Excel 2};

% Cleaning
\node (clean1) [process, below=of excel1] {Clean};
\node (clean2) [process, below=of excel2] {Clean};

% Center anchor
\path (clean1) -- (clean2) coordinate[midway] (middle);

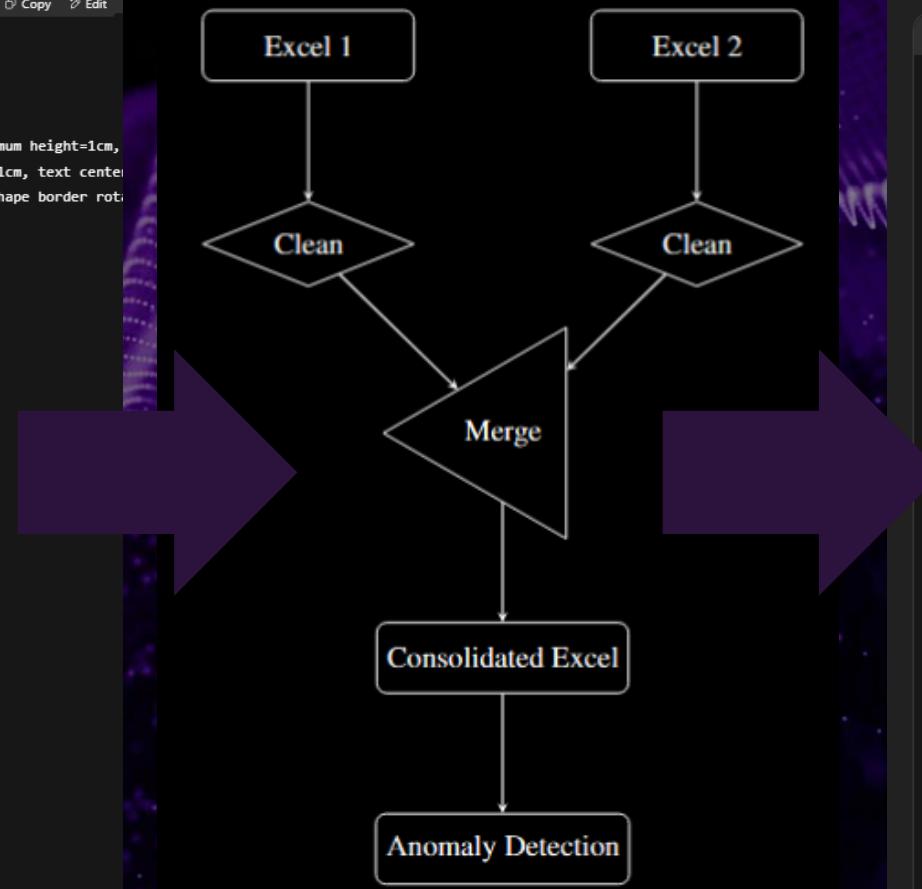
% Merge node centered under clean1 and clean2
\node (merge) [merge, below=1.7cm of middle] {Merge};

% Output nodes
\node (consolidated) [startstop, below=of merge] {Consolidated Excel};
\node (anomaly) [startstop, below=of consolidated] {Anomaly Detection};

% Arrows
\draw [arrow] (excel1) -- (clean1);
\draw [arrow] (excel2) -- (clean2);
\draw [arrow] (clean1) -- (merge);
\draw [arrow] (clean2) -- (merge);
\draw [arrow] (merge) -- (consolidated);
\draw [arrow] (consolidated) -- (anomaly);

\end{tikzpicture}

```



Python Script:

```
python
import pandas as pd
from sklearn.ensemble import IsolationForest

# === Step 1: Load Excel Files ===
df1 = pd.read_excel('excel1.xlsx')
df2 = pd.read_excel('excel2.xlsx')

# === Step 2: Clean Function ===
def clean_data(df):
    # Remove text qualifiers, trim spaces, drop empty rows
    df.columns = df.columns.str.strip().str.replace("'", "")
    for col in df.select_dtypes(include='object').columns:
        df[col] = df[col].str.strip().str.replace("'", "")
    df.dropna(how='all', inplace=True)
    return df

df1_clean = clean_data(df1)
df2_clean = clean_data(df2)

# === Step 3: Merge Files ===
consolidated = pd.concat([df1_clean, df2_clean], ignore_index=True)

# === Step 4: Save Consolidated Excel ===
consolidated.to_excel('consolidated.xlsx', index=False)

# === Step 5: Anomaly Detection ===
# Assume numerical columns for anomaly detection
num_cols = consolidated.select_dtypes(include='number').columns
iso_forest = IsolationForest(contamination=0.05, random_state=42)
consolidated['anomaly'] = iso_forest.fit_predict(consolidated[num_cols])
```

Vibe coding with Python

Vibe coding: Pros and cons

- ✓ Fast development and prototyping
- ✓ Lower entry barrier for beginners
- ✓ Focus on ideas over syntax

⚠ Requires testing and review
(debugging)

⚠ Potential over-dependence on AI
⚠ Security and code quality
concerns



Accessibility



Speed



Innovation



Engagement



Code Quality



Security



Maintenance

Key takeaways

1. Is artificial intelligence really “intelligent”?
2. What is the difference between traditional statistical/econometric modelling and machine learning (hint: think about partitions)
3. Will AI replace humans and displaced jobs?

