The Computational Toolkit: Linux & Python

Key Concept: We use a hybrid environment—Linux for system management and tool execution, and Python for data analysis and modeling.

Linux (The Operating System)

- Used to install software
- manage files
- run external programs like PaDEL

Python (The Brain)

- Used to download data
- clean it
- calculate features
- build machine learning models

Jupyter Notebook

- The interactive laboratory where we mix code, text, and plots.

Real-World Analogies for the Bioinformatics Toolkit

Understanding the roles of Linux, Python, and Jupyter through everyday comparisons.

Linux (The Operating System)

The Workshop & Foundation

Analogy: Provides infrastructure, tools, and power for organizing files and running programs.

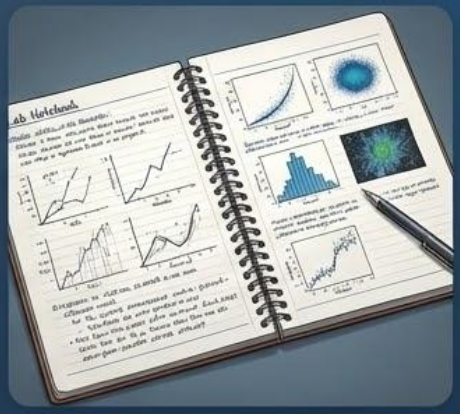
Python (The Brain)

The Skilled Craftsman & Logic

Analogy: Performs the work with logic and skill, transforming raw data into insights.

Jupyter Notebook (The Interactive Laboratory)

The Lab Notebook & Canvas



Analogy: An interactive record combining code, plots, and text in in a shareable document.


Linux

Linux Analogy: The Engine Behind Your Android Phone

1. The Foundation: The Linux Kernel

The core component that manages hardware, memory, and processes. It's the powerful, underlying foundation.

2. Building the System: The Android OS

A complete operating system built on top of the Linux kernel, including middleware, libraries, and a user interface framework.

3. Adding User Features: UI & Apps

The layer users interact with directly. Apps are like car features (radio, AC) that make the system usable and personal.

4. The User Experience: The Driver

Many students use Android phones daily for communication, learning, and entertainment, without a second thought.

5. The Hidden Connection: A Linux-Powered Device

Most don't realize their familiar Android phone is a powerful Linux-based device under the hood.

Open-Source: The Power of Collective Effort

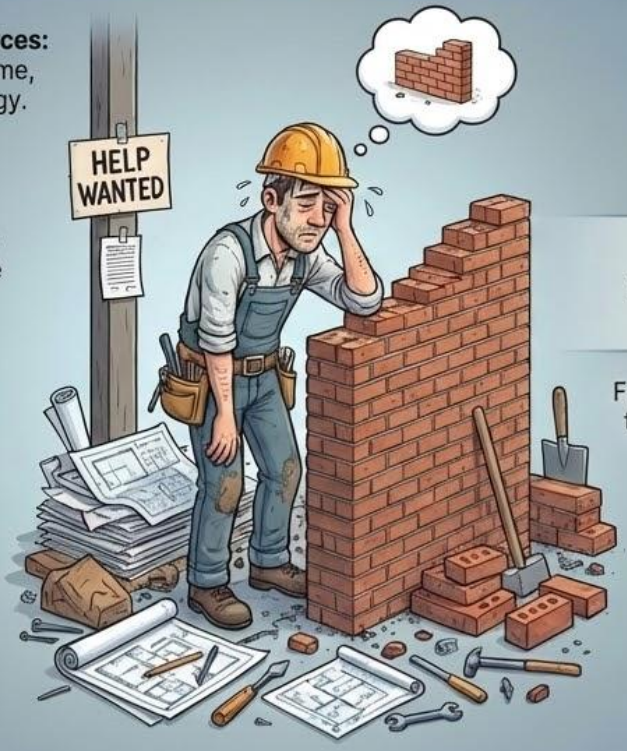
Contrasting solitary development with a vast community of contributors

Solitary Development (Closed Source)

Limited Resources: One person's time, skills, and energy.

Slow Progress: Bottlenecks are inevitable; updates are infrequent.

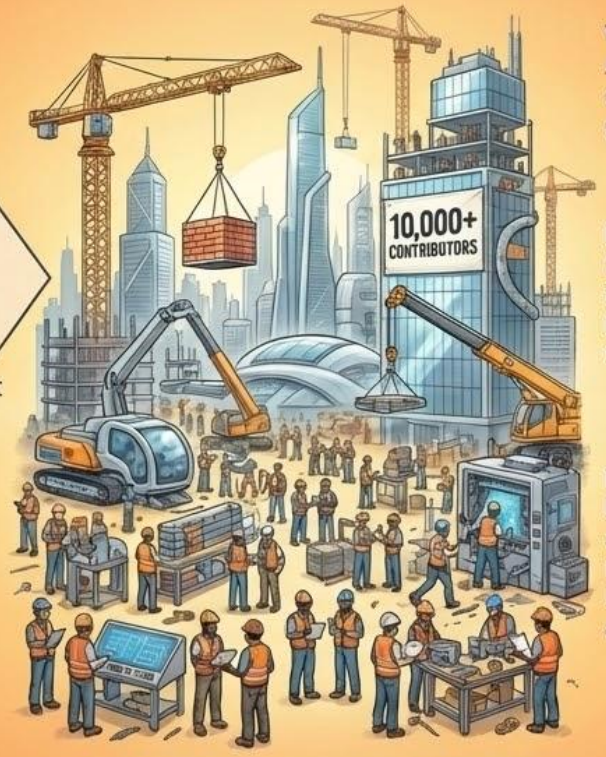
Single Point of Failure: If the developer stops, the project dies.



The Open-Source Advantage

From individual effort to collective power.

Community Collaboration (Open Source)



Vast Talent Pool: Access to 10,000+ diverse experts globally.

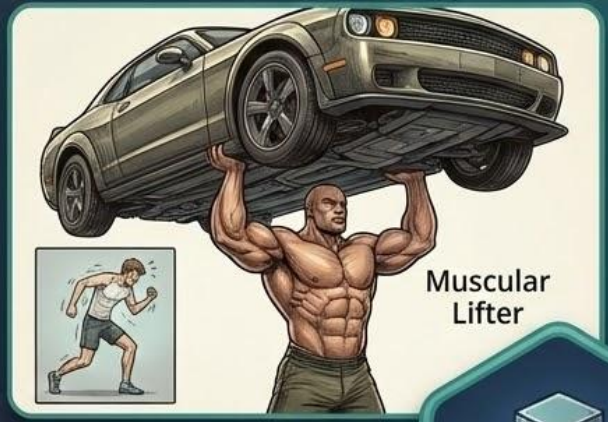
Rapid Innovation: Continuous improvements, bug fixes, and new features.

Resilience & Peer Review: Code is scrutinized by many, leading to higher quality and security.

High Performance and Scalability: Scaling Analogies

Understanding horizontal and vertical scaling through everyday comparisons.

Vertical Scaling (Scale Up)



Like upgrading to a single, more powerful machine (stronger CPU, more RAM) when the existing one is is overloaded.

Linux: Scalable Core

Horizontal Scaling (Scale Out)

Like adding more standard machines to a cluster to share the workload and increase overall capacity.

Powerful CLI & Automation: Real-World Analogies

Contrast manual, error-prone tasks with automated, reproducible workflows using Linux.

Manual Data Wrangling & Analysis

Like a researcher manually copying data from notebooks into spreadsheets, prone to errors and incredibly slow.

Bioinformatics: Manually renaming and organizing thousands of sequencing files, one by one.

Linux CLI: The Automation Engine

Automated Workflows & Pipelines

Like a high-throughput robotic system that automatically processes data with precision and speed, driven by a simple script.

Bioinformatics: Using a single command to automatically align, call variants, and generate reports for an entire dataset.

BASIC LINUX COMMANDS FOR BIOINFORMATICS

Essential tools for navigation, file management, and data processing

File & Directory Navigation

- **pwd** : Print Working Directory (current location)
- **cd** : Change Directory (navigate folders)
- **ls** : List directory contents
- **mkdir** : Make a new directory

File Inspection & Manipulation

- **cat** : Display file content
- **head / tail** : View beginning/end of file
- **cp** : Copy files
- **mv** : Move or rename files
- **rm** : Remove (delete) files

Data Processing & Filtering

- **grep** : Search for patterns in text
- **cut** : Extract sections from lines
- **sort** : Sort lines of text
- **uniq** : Report or omit repeated lines
- **awk** : Pattern scanning & processing
- **sed** : Stream editor for filtering/transforming

Question 1: Heavy Compute

What makes drug discovery compute-heavy, and why is Linux the standard for handling it?

Answer 1: Scalable Performance

Docking, virtual screening, and MD runs require massive computational power. Linux is the operating system that runs high-performance computing (HPC) clusters and cloud machines efficiently.

Question 2: Drug-Discovery Tooling

What is the primary ecosystem for most drug discovery tools, and how are they typically managed?

Answer 2: Open Ecosystem

The ecosystem is predominantly Linux-first, with major tools like RDKit, AutoDock Vina, GROMACS, OpenMM, and PyTorch. They are best installed and managed via Conda or Docker on Linux.

Question 3: Stable Long Jobs

Why is Linux preferred for running experiments that last for hours or days?

Answer 3: Reliability

Linux is built for stability and long runtimes, offering robust job logging and significantly fewer random failures or ‘breaks’ compared to other operating systems.

Question 4: Full Control

How does Linux enable the scaling of drug discovery workflows?

Answer 4: Automation + Reproducibility

Linux allows for complete scripting of workflows (prep → docking → scoring → ML), ensuring results are reproducible and easily scalable to thousands of compounds.


Python

Why Python is Essential for Bioinformatics

The versatile, high-level language for data analysis, scripting, and workflow development.

1. Extensive Library Ecosystem

Access specialized tools like Biopython, Pandas, NumPy, SciPy. Simplify complex data tasks.

2. Data Analysis & Visualization

Effortlessly wrangle large datasets. Create insightful plots with Matplotlib, Seaborn, Plotly.

Python

3. Scripting & Workflow Automation

Automate repetitive tasks. Build reproducible pipelines with tools like Snakemake.

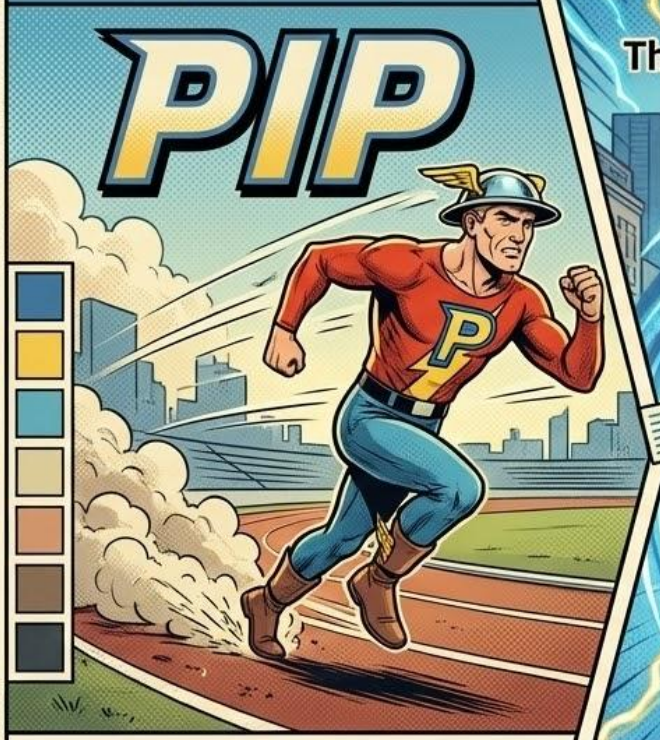
4. Machine Learning Integration

Seamlessly apply ML models with Scikit-learn, TensorFlow, PyTorch for predictive modeling.

PIP vs. UV:

The Python Package Manager Race

PIP (The Classic)



Standard, reliable, but can be slow.
The original Python package installer.

Written in Python | Standard speed resolution | Classic resolver

UV (The Speedster)

Ultra-fast, modern installer written in Rust. Designed for blazing performance.

Written in Rust | Blazing fast resolution | Optimized, modern resolver

The Speed Force of Dependency Management

Speed	Standard	Blazing Fast (Rust-powered)
Language	Python	Rust
Resolution	Classic, can be slow	Modern, optimized

Python Libraries Analogy: The Data Science Construction Project

1. Gathering Materials & Laying the Foundation

NumPy provides the raw numerical building blocks (arrays), while RDKit offers specialized materials for chemical structures.

2. Erecting the Structured Framework

Pandas organizes the raw data into structured, tabular dataframes, like a building's steel frame and floors.

3. Visualizing the Design & Progress

Matplotlib creates the core architectural drawings, while Seaborn adds stylistic and statistical visualizations.

4. Integrating Smart Systems & Automation

scikit-learn adds the "intelligence" (machine learning models) for automated insights and predictions.

5. The Completed Data Structure

The final, integrated data science pipeline, where all libraries work together to produce actionable insights.

Python Libraries: A powerful ecosystem for building data science and bioinformatics solutions.

Python Libraries: Sequence Analysis, Manipulation, and Cheminformatics

Biopython

The ‘Swiss Army Knife’ for bioinformatics. Purpose: Parsing file formats (FASTA, GenBank), sequence manipulation (translation, reverse complement), online database access (NCBI Entrez), and 3D structure analysis (PDB).

ATCG...TCATCG
ATCAACATTCTC
ATGGCTACTATC
ATCGCATCAG...

RDKit

Purpose: Cheminformatics and machine learning for chemistry.
Features: Molecular structure manipulation, fingerprint generation, and property prediction.

ChEMBL Webresource Client

Purpose: Access ChEMBL database programmatically.
Features: Retrieve bioactivity data, target information, and compound details.

PaDEL-Py

Purpose: Calculate molecular descriptors and fingerprints.
Features: Generate a wide range of chemical descriptors for structure-activity relationship (SAR) analysis.

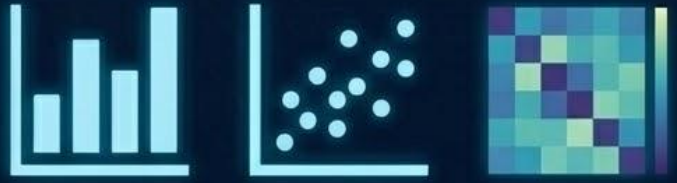
Python Libraries: Data Analysis & Visualization

Pandas

Purpose: Data manipulation and analysis. Handling large datasets (dataframes), filtering, and aggregation. Essential for tabular data.

Matplotlib / Seaborn

Purpose: Creating static, animated, and interactive visualizations. Seaborn provides a higher-level interface for attractive statistical graphics.



Python Libraries: Machine Learning & Scientific Computing

NumPy

Purpose: Fundamental package for scientific computing. Support for large, multi-dimensional arrays and matrices.

SciPy

Purpose: Built on NumPy, adds functionality for scientific and technical computing (e.g., optimization, statistics).

Scikit-learn

Purpose: Simple and efficient tools for predictive data analysis. Used for machine learning tasks like classification and clustering.


Alzheimers

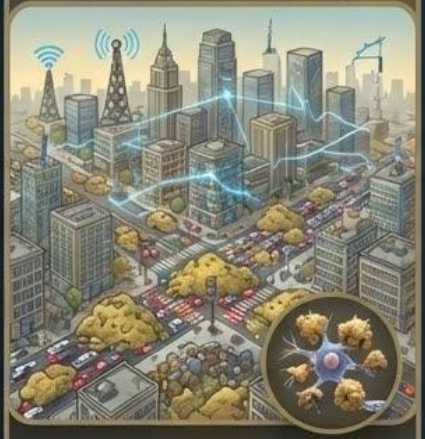
Understanding Alzheimer's: The 'Failing City' Analogy

A Visual Guide to the Brain's Decline, from Healthy Function to Neurodegeneration

1. The Healthy City (Healthy Brain)

- **Analogy:** Traffic flows freely. Deliveries & communication are constant.
- **Biology:** Neurons communicate effectively. Internal transport systems are perfect.

2. The Roadblocks (Amyloid Plaques)



- **Analogy:** Junk piles up, blocking traffic and preventing signals.
- **Biology:** Beta-amyloid proteins clump outside neurons, disrupting signaling.

3. Internal Failure (Tau Tangles)

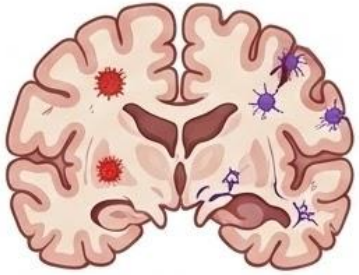
- **Analogy:** Internal systems (plumbing, wiring) get twisted and broken. Supplies fail.
- **Biology:** Tau proteins twist inside neurons, destroying transport systems.

4. The City Collapses (Alzheimer's Disease)

- **Analogy:** Blocked roads & failing insides lead to city-wide collapse. Essential services stop.
- **Biology:** Widespread neuron death causes brain atrophy, leading to memory loss.

ALZHEIMER'S DISEASE vs. GENERAL MEMORY LOSS: UNDERSTANDING THE DIFFERENCE

ALZHEIMER'S DISEASE



- Progressive Neurodegenerative Disease
- Caused by Protein Build-up & Brain Cell Death
- Irreversible Cognitive Decline
- Impacts Daily Function & Behavior

KEY DIFFERENCES

Feature	Alzheimer's Disease	General Memory Loss
Nature	Disease, Progressive	Symptom, Variable
Cause	Brain Pathology	Multiple Factors
Progression	Irreversible Decline	Often Manageable/ Reversible
Daily Life Impact	Severe, Disabling	Mild to Moderate

GENERAL MEMORY LOSS

- Symptom of Various Conditions
- Diverse Causes (Stress, Aging, etc.)
- Variable Progression (Stable/Reversible)
- Primarily Affects Recall

Understanding Alzheimer's Disease: A Complex Neurodegenerative Disorder

A progressive disease that destroys memory and other important mental functions.

1. Amyloid Plaques

Abnormal clumps of beta-amyloid protein build up between nerve cells, disrupting communication and causing cell damage.

2. Neurofibrillary Tangles

Twisted fibers of tau protein form inside brain cells, blocking the transport of nutrients and other essential molecules.

Alzheimer's Pathology

3. Brain Cell Death & Atrophy

Widespread neuron loss leads to significant brain shrinkage (atrophy), affecting memory, thinking, and behavior.

4. Progressive Symptoms

Symptoms worsen over time, from mild memory loss to the inability to carry on a conversation or respond to the environment.

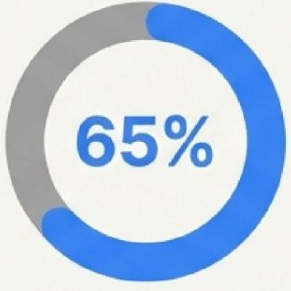
The Mission: Confronting Alzheimer's Disease

~50

million people globally

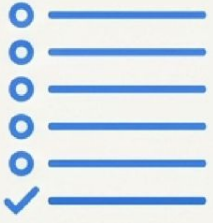
60-70%

of all dementia cases worldwide



#6

leading cause of death in the US



Despite its prevalence, no cure currently exists. Computational methods offer a new path to accelerate drug discovery, reducing the typical $2.6 billion cost and years of lab testing.

The Immense Challenge of Alzheimer's Disease

The Human Cost

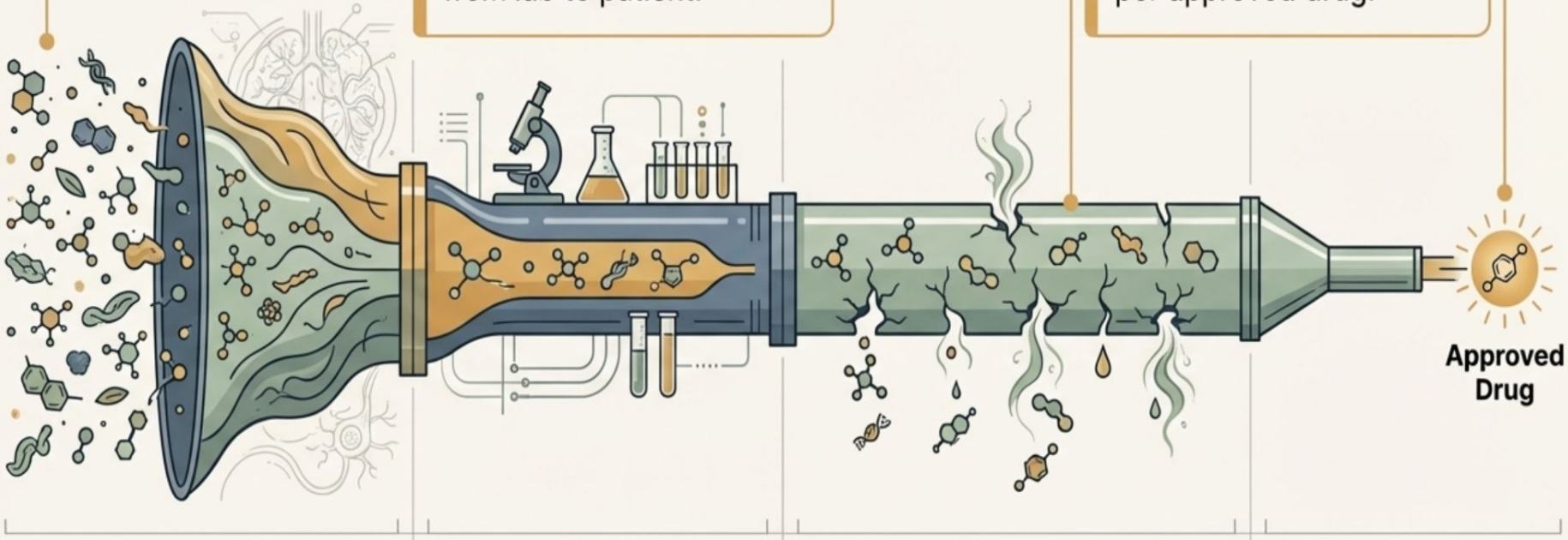
- A progressive neurodegenerative disease destroying memory and cognitive function.
- Causes **60–70%** of all dementia cases worldwide.
- Affects an estimated **50 million** people globally.
- **6th leading cause of death** in the United States.
- Currently, **no cure exists**.

The Drug Discovery Bottleneck

- Developing a new drug is incredibly slow and expensive.
- Average cost to bring a new drug to market: **$2.6 billion**.
- Our project aims to significantly reduce the time and cost in the crucial early-discovery phase.

Average Time: **10-15 years** from lab to patient.

Average Cost: **$2.6 billion** per approved drug.



Basic Research: Years 1-5

Pre-clinical Testing: Years 6-8

Clinical Trials: Years 9-15

FDA Approval

This process is too slow and expensive to effectively combat a crisis on the scale of **Alzheimer's**. We need a faster way to screen candidates.

Drug Discovery

Target Identification

High-Throughput Screening

Lead Optimization

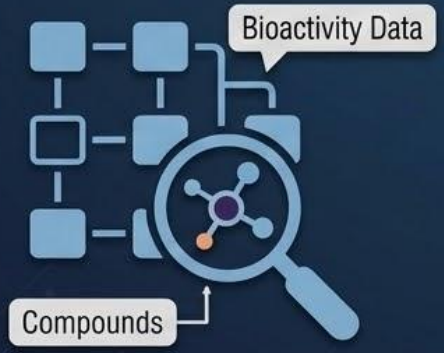
Preclinical Development

Clinical Trials

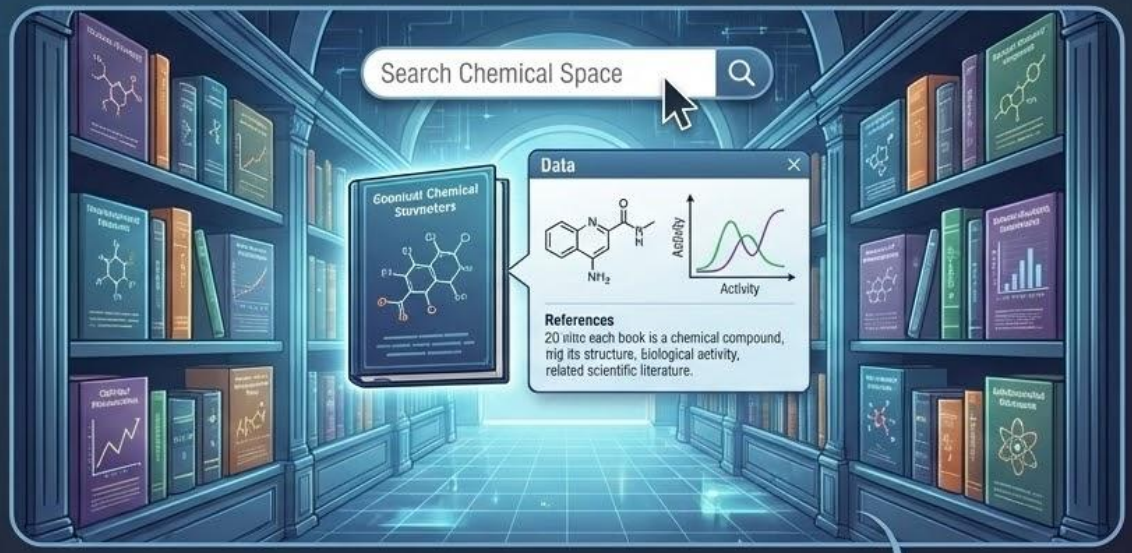
ChEMBL: A Library of Chemical Compounds

ChEMBL

(Chemical Database of EMBL)



What It Is & Analogy



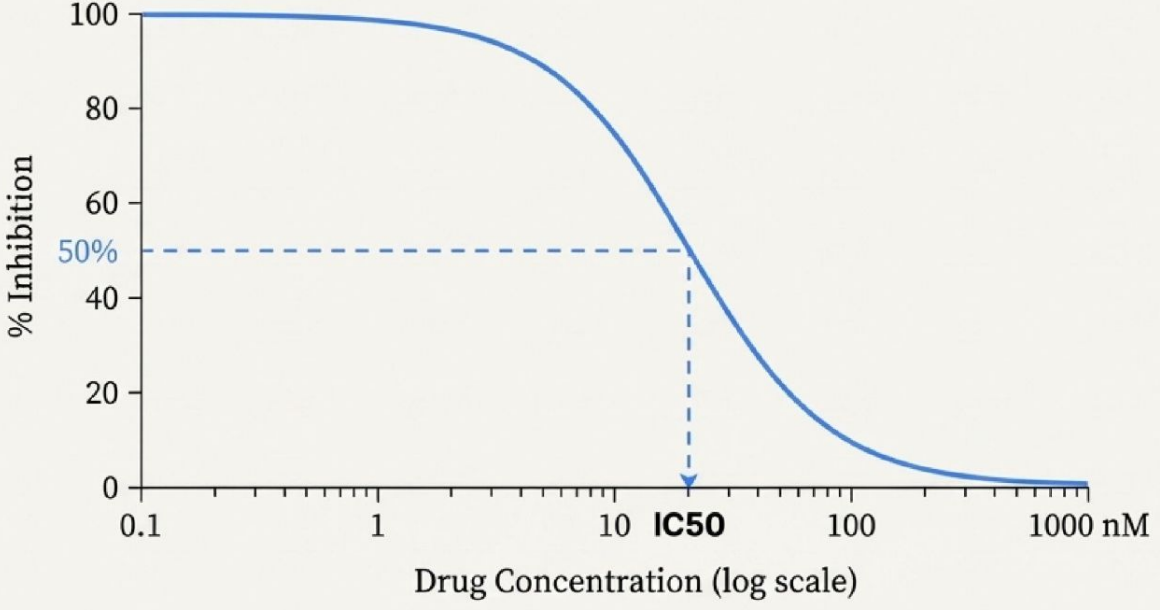
Think of it as a vast digital library where each book is a chemical compound, detailing its structure, biological activity, and related scientific literature.

Data Extraction → Analysis Pipeline

Measuring Success: How We Quantify a Drug's Potency

IC50 (Half Maximal Inhibitory Concentration)

"A quantitative measure that indicates how much of a drug is needed to inhibit a biological process by 50%."



Lower is Better.

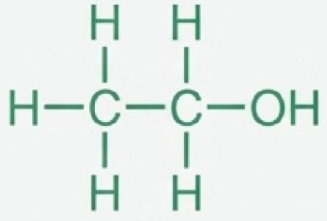
- A low IC50 (e.g., 5 nM) means a tiny amount is needed = a very potent drug.
- A high IC50 (e.g., 50,000 nM) means a huge amount is needed = a weak drug.

All data is standardized to Nanomolar (nM) units for consistency.

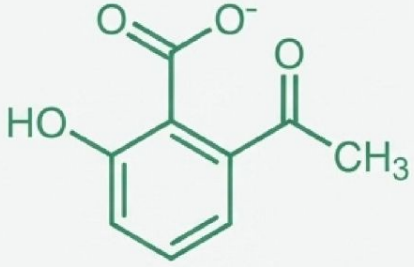
The Language of Molecules: SMILES Notation

How we represent a 3D chemical structure as text for computer processing.

SMILES: Simplified Molecular Input Line Entry System. A specification for describing molecular structures using short ASCII strings.



CCO



CC(=O)OC1=CC
=CC=C1C(=O)O

Ethanol

Aspirin

We use **Canonical SMILES** to ensure every unique molecule has exactly one unique text identifier, preventing ambiguity in the dataset.

Is It a Viable Drug? Applying Lipinski's Rule of 5

A rule of thumb to evaluate if a compound has properties that would make it a likely orally active drug. It helps filter for "drug-like" candidates.

The Rules

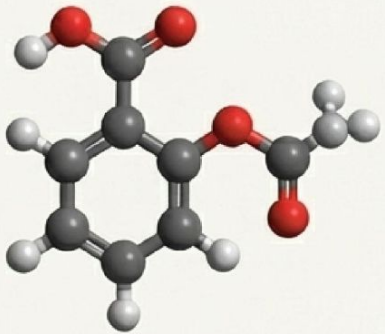
- **Molecular Weight (MW) < 500 Daltons:** Not too big.
- **LogP < 5:** Not too greasy (balances solubility).
- **Hydrogen Bond Donors < 5:** Controls interactions with water.
- **Hydrogen Bond Acceptors < 10:** Controls interactions with water.

Good Candidate

✓ ✓ ✓ ✓

Poor Candidate

✗ ✗ ✗ ✗



SMILES Representation

```
CC(=O)OC1=CC=CC=C1C(=O)O
```

Lipinski Descriptors

Molecular Weight	180.16
LogP	1.19
H-Bond Donors	1
H-Bond Acceptors	4

The QSAR Process Analogy: A Culinary Journey

1. Gathering Ingredients & Recipes

Collecting a library of diverse ingredients and historical recipes with known taste outcomes (activity data).

2. Measuring Flavor Properties

Quantifying key properties of ingredients (e.g., sugar content, acidity, bitterness) into numerical data.

3. Learning the "Flavor Profile"

The chef learns how different ingredient combinations and their properties combine to create specific tastes (the structure-activity relationship).

4. Blind Taste Testing (Validation)

Testing the chef's knowledge on new, untasted dishes to verify the accuracy of their flavor predictions.

5. Creating New Recipes & Predicting Tastes

Using the learned knowledge to predict the taste of brand new recipes before they are even cooked, guiding the creation of new culinary delights.

QSAR: Quantitative Structure-Activity Relationship - A powerful tool for accelerating drug discovery.

QSAR Process: From Structure to Activity

1. Data Collection & Curation

- Gather structure & activity data.
- Clean, standardize, validate.

2. Molecular Descriptor Calculation

- Generate numerical descriptors.
- Characterize structure.

3. Model Development & Training

- Apply ML algorithms.
- Learn structure-activity relationship.

4. Model Validation & Evaluation

- Assess predictive power.
- Generalization capability.

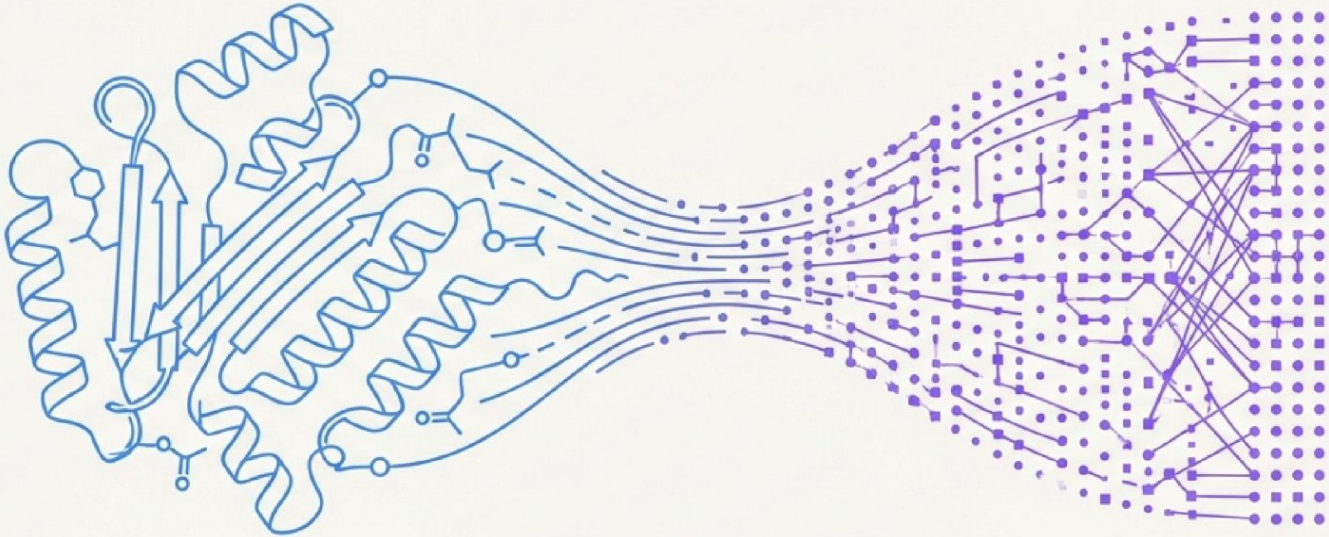
5. Prediction & Application

- Predict activity of new compounds.
- Virtual screening & lead optimization.

QSAR: Quantitative Structure-Activity Relatioship - A powerful tool for accelerating drug discovery.

From Biology to Bits: Predicting Alzheimer's Drugs with Machine Learning

A Computational Drug Discovery Project for Beta-amyloid A4 Protein Inhibitors



The process of building a Quantitative Structure-Activity Relationship (QSAR) model to accelerate the search for a cure.

The Biological Target: Pinpointing the Cause

The Key Player: Beta-amyloid A4 Protein (Aβ)

A small protein fragment produced when a larger protein (APP) is cut. In Alzheimer's, it accumulates and forms toxic clumps, or plaques.

ChEMBL Target ID: CHEMBL2487

The Disease Process: A Cascade of Damage

1 Aβ Aggregation
Proteins clump together.

2 Synaptic Dysfunction
Clumps disrupt neuron signals, affecting memory.

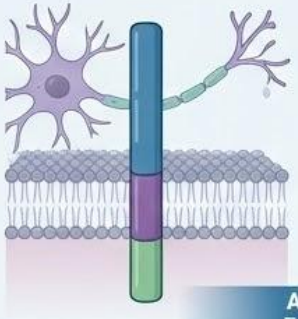
3 Inflammation & Stress
The brain's immune response causes collateral damage.

4 Neuronal Death
Neurons die, the brain shrinks, and cognitive function declines.

Our Goal: Find chemical compounds (inhibitors) that stop this process at its source.

The Amyloid Cascade & Drug Intervention

1. The Healthy State



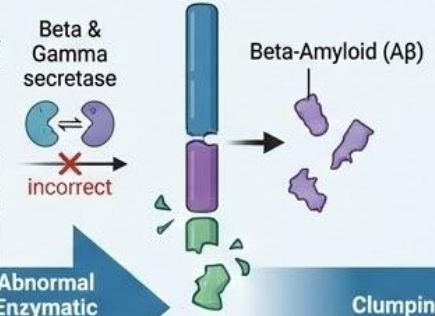
Abnormal Enzymatic Cleavage

APP (Amyloid Precursor Protein)

A normal protein found in the membrane of neuron cells. It helps with neuron growth and repair.

Enzymes usually cut this protein into harmless fragments that the body dissolves.

2. The Malfunction (The Trigger)



Clumping

Production of Beta-Amyloid (Aβ)

Enzymes (Beta and Gamma secretase) cut the APP incorrectly.

Instead of harmless fragments, they create sticky fragments called **Beta-amyloid monomers**.

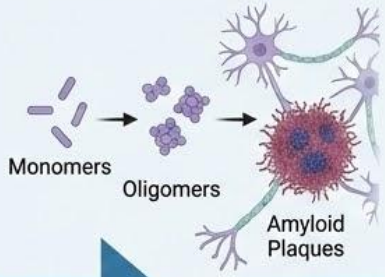
This is the **specific protein "target"** found in your ChEMBL dataset.

PROJECT FOCUS: Your machine learning model identifies molecules (inhibitors) designed to **BLOCK** this arrow.

Active Compounds (ICS0 < 1000nM): Successfully stop this clumping.

Inactive Compounds: Fail to stop this clumping.

3. Aggregation (The Buildup)

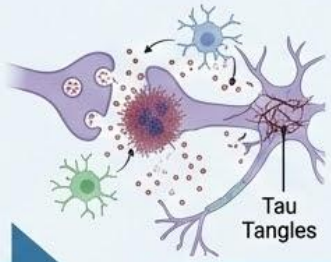


Disruption of Cell Signaling

Oligomers & Plaques

Monomers (single pieces) stick together to form **Oligomers** (small clumps). Oligomers clump further to form insoluble **Amyloid Plaques** between nerve cells.

4. The Toxicity (The Damage)



Neurodegeneration

Synaptic Failure & Inflammation

Plaques physically block signals between neurons (synapses). The body's immune system attacks the plaques, causing inflammation that harms nearby cells.

(Secondary Effect): This triggers **Tau Tangles** inside the cell, destroying the cell's transport system.

5. The Outcome (The Disease)

Neuronal Death & Atrophy

Brain cells die and the brain tissue shrinks (atrophy).

Symptoms: Memory loss, cognitive decline, loss of function.

Mining the Gold (Data Acquisition)

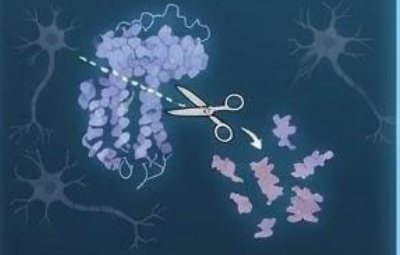
Querying the ChEMBL database to find verified biological data.

Phase 1: Biological Foundation & Data Mining

ChEMBL Database

- Massive, open-access database (EBI)
- Bioactive molecules with drug-like properties
- Primary source for experimental results

Target Protein: Beta-amyloid A4 (CHEMBL2487)

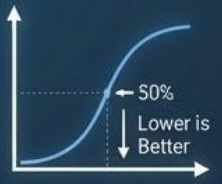


- Implicated in Alzheimer's Disease
- Cleaves into sticky Aβ fragments
- Aggregates into plaques (neuron death)
- **Goal:** Find binding compounds to inhibit activity

Inhibitor

- Chemical compound (drug)
- Binds to target protein, decreases activity
- **Context:** Stops Beta-amyloid aggregation

IC50 & Standard Units



IC50 (Half Maximal Inhibitory Concentration)

- Measure of potency (Concentration for 50% inhibition)
- Lower = Better (e.g., 5 nM = potent)

Standard Units (nM)

- Nanomolar (10^{-9} mol/L)
- Standardized for consistent comparison

Example: Data Acquisition

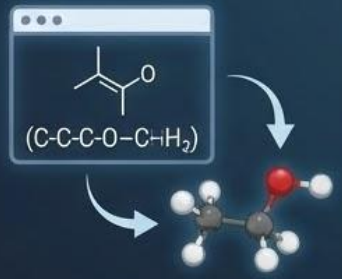
Fetching thousands of lab results for Beta-amyloid A4 (CHEMBL2487) so the AI has ‘ground truth’ to learn from.

Scrubbing the Data (Preprocessing)

Standardizing messy lab units and creating clean target variables.

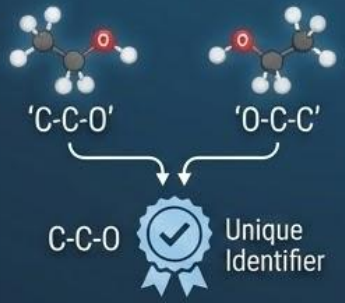
Phase 2: Data Cleaning & Labeling

SMILES



- Simplified Molecular Input Line Entry System (ASCII strings)
- **Example**: CCO represents Ethanol
- **Function**: Stores 3D structures as text for computer processing

Canonical SMILES



- A 'standardized' unique text representation
- Ensures every unique molecule has exactly one unique identifier (despite multiple ways to write SMILES)

Bioactivity Class

- Categorical label to bucket compounds based on potency
- **Active:** IC50 ≤ 1,000 nM (Strong candidates)
- **Intermediate:** 1,000 < IC50 < 10,000 nM (Borderline candidates)
- **Inactive:** IC50 ≥ 10,000 nM (Weak/Useless candidates)

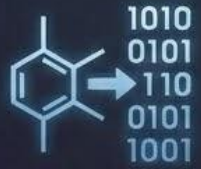
Missing Data (NA)

- Rows where critical values (e.g., standard_value, canonical_smiles) are undefined
- These rows are dropped during preprocessing

Example: Preprocessing

Converting wide-ranging IC50 values into normalized pIC50 scores so the model compares apples to apples.

Chemistry to Code (Feature Engineering)



Translating chemical structures (SMILES) into machine-readable math.

Phase 3: Feature Engineering (Chemistry to Math)

Lipinski's Rule of 5

- Rule of thumb to evaluate drug-likeness (oral activity prediction)
- **MW** < 500 Daltons (Size)
- **LogP** < 5 (Solubility)
- **H-Bond Donors** < 5
- **H-Bond Acceptors** < 10

LogP (Partition Coefficient)

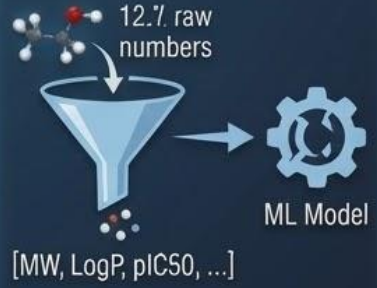
- Measures differential solubility in octanol/water
- **High LogP (>5):** Lipophilic (Fat-loving), traps in fat/toxic
- **Low LogP (<5):** Hydrophilic (Water-loving), dissolves in blood

pIC50

$$pIC_{50} = -\log_{10}(IC_{50})$$

- **Linearity:** Converts exponential IC50 to linear scale for regression
- **Intuition:** Converts "Lower is Better" to "Higher is Better"

Goal: ML-Ready Features



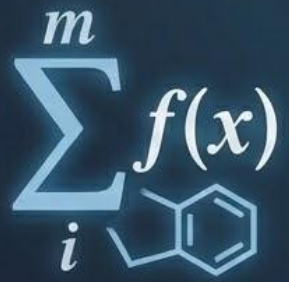
- Transform chemical data into numerical features
- These features serve as inputs for machine learning models
- **Final Output:** A dataset ready for training

Example: Feature Engineering

Using PaDEL to turn a drawn molecule into a Fingerprint (a barcode of 881 1s and 0s) the computer can understand.

Phase 4: Molecular Descriptors (The 'Fingerprints')

Molecular Descriptor



- Mathematical representation of physical/chemical properties
- e.g., molecular weight
- number of rings
- charge

Molecular Fingerprint

01011
01011
01011 Binary vector (array of 0s and 1s)

Represents presence/absence of specific substructures

Analogy: Like a barcode for a molecule.

PubChem Fingerprint

Specific system checking for 881 chemical substructures

Output: A list of 881 binary bits for every molecule

PaDEL-Descriptor

Software tool (Passive ADME Learning)

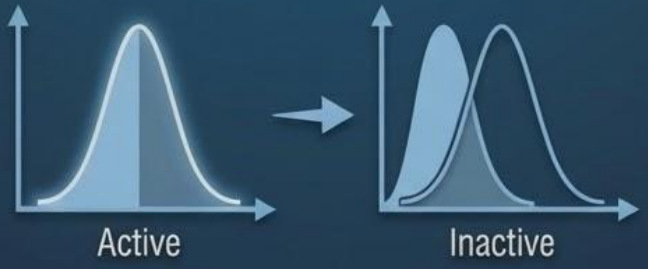
Calculates molecular descriptors and fingerprints from SMILES strings

Phase 5: Exploratory Data Analysis (EDA)

Chemical Space

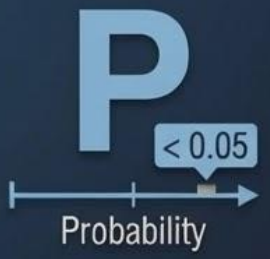
- A multidimensional descriptor space.
- **Visualization** (e.g., MW vs. LogP) checks if "Active" compounds cluster in "drug-like" regions.

Mann-Whitney U Test



- Non-parametric test comparing distributions of two groups (Active vs. Inactive).
- **Goal**: Assess if property differences are statistically significant, not random.

P-Value



- Probability that observed results are due to chance.
- **p < 0.05** indicates statistical significance (the difference is likely real).

The Prediction Engine (Machine Learning)

Training algorithms to map chemical structure to biological potency.

Phase 6: Model Building (Machine Learning)

Feature Selection (Variance Threshold)

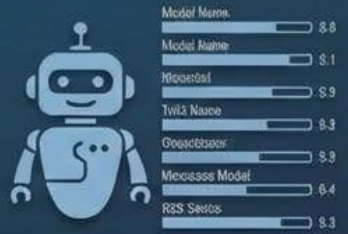
- Reduces input variables (X).
- **Process:** Removes low-variance columns (e.g., near-constant fingerprints) that provide little information.

Train/Test Split

Splits data into subsets:

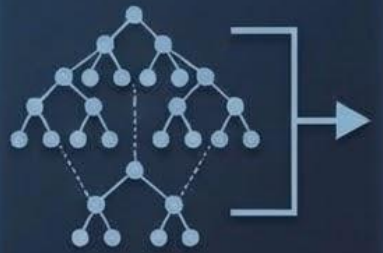
- **Training** (80%): Trains the model.
- **Testing** (20%): Evaluates performance on unseen data.
- **Purpose:** Prevents overfitting.

LazyPredict



- Python library for building baselines.
- Runs multiple models with default parameters and ranks them by performance.

Random Forest Regressor



- Ensemble method using multiple decision trees.
- Outputs the mean prediction of individual trees.
- **Status:** Top performer for QSAR data.

Example: Machine Learning

A Random Forest looks at the fingerprints and predicts:
“This new structure will likely kill the Alzheimer’s protein.”

Phase 7: Evaluation (Grading the AI)

R^2 (Coefficient of Determination)

- Proportion of variance in y explained by X.
- Range: 0 to 1.
- **Goal:** > 0.6 (good for QSAR).

RMSE (Root Mean Squared Error)

- Standard deviation of prediction errors (residuals).
- Shows concentration of data around the line of best fit.
- **Goal:** Lower value indicates better fit.

Scatter Plot (Experimental vs. Predicted)

- Visual comparison: Lab values (X-axis) vs. AI predictions (Y-axis).
- **Ideal:** All points fall on a straight diagonal line.