Module 3

Introduction to Machine Learning (ML)

Cynthia Sandor

Edmond J Safra Assistant Professor
UKRI Future Leader Fellow
UK Dementia Research Institute Group Leader

UK Dementia Research Institute
Imperial College London
THE MICHAEL J. FOX FOUNDATION FOR PARKINSON'S RESEARCH
Founding funders:
UKRI Medical Research Council
EDMOND J. SAFRA PHILANTHROPIC FOUNDATION

Announcements

- **Next workshop:** 12th November, RCS1 212A – Versatile Teaching Laboratory (VTL) 2A, South Kensington (with Nathan Skene)
- **Today:** Introduction to Machine Learning + Tutorial 3

If you have not finished Tutorial 3, please move directly to Tutorial 4.

- **End of session:** Hackathon expectations overview.
- **Tuesday:** All tutorial solutions will be released.

Reminder:

- 13/14/19/20 November = Consolidation days
- Finish Tutorials + Challenge.
- Questions → ask in the **General** MT channel.

Dr Nathan Skene

What We'll Do Today

- **Lecture:** Short introduction to Introduction to Machine Learning

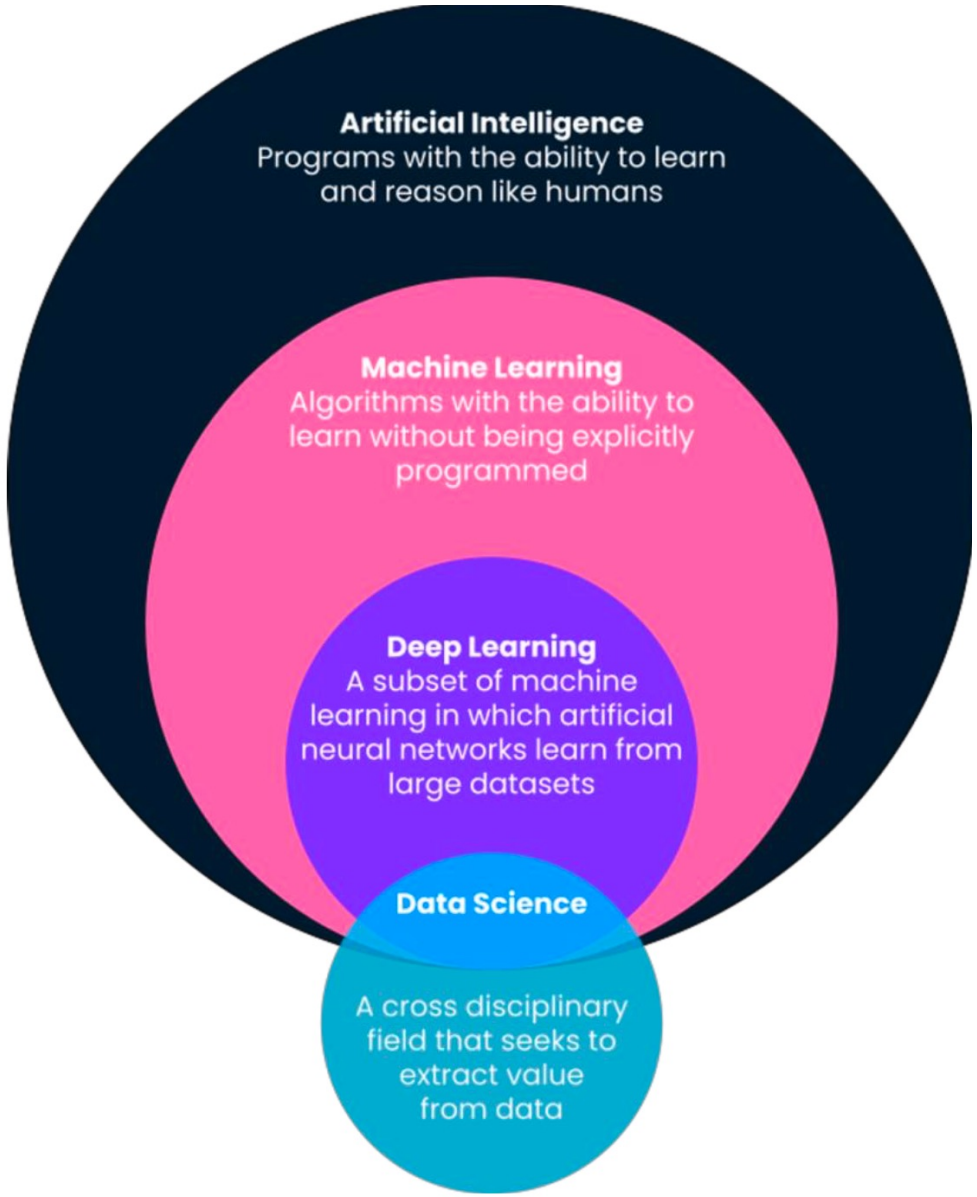
- **Tutorial / Challenge** — You will learn how to apply **supervised** and **unsupervised** ML methods to real Parkinson's disease datasets.

  In this tutorial you will:
  Prepare data (split, scale, handle missing values).
  Perform **regression** (predict tremor severity).
  Perform **classification** (predict disease status).
  Understand **feature selection** and the **curse of dimensionality**.
  Evaluate models using **residuals**, **coefficients**, and **confusion matrices**.
  Explore **unsupervised learning** (PCA, UMAP, K-means).

  Datasets provided include **proteomics** and **clinical** features.

Marirena Bafaloukou

Differences Between AI, Data Science, Machine Learning and Deep Learning



Traditional (Non–ML) Coding

Rule-based Programming — You Write the Rules

Imagine we want to decide if a student will pass based on how many hours they study.
We can write *if/else rules* — simple, but rigid.

```
def will_pass(hours):
    if hours >= 3:
        return "Pass"
    else:
        return "Fail"

print(will_pass(5))  # Pass
print(will_pass(1))  # Fail
```

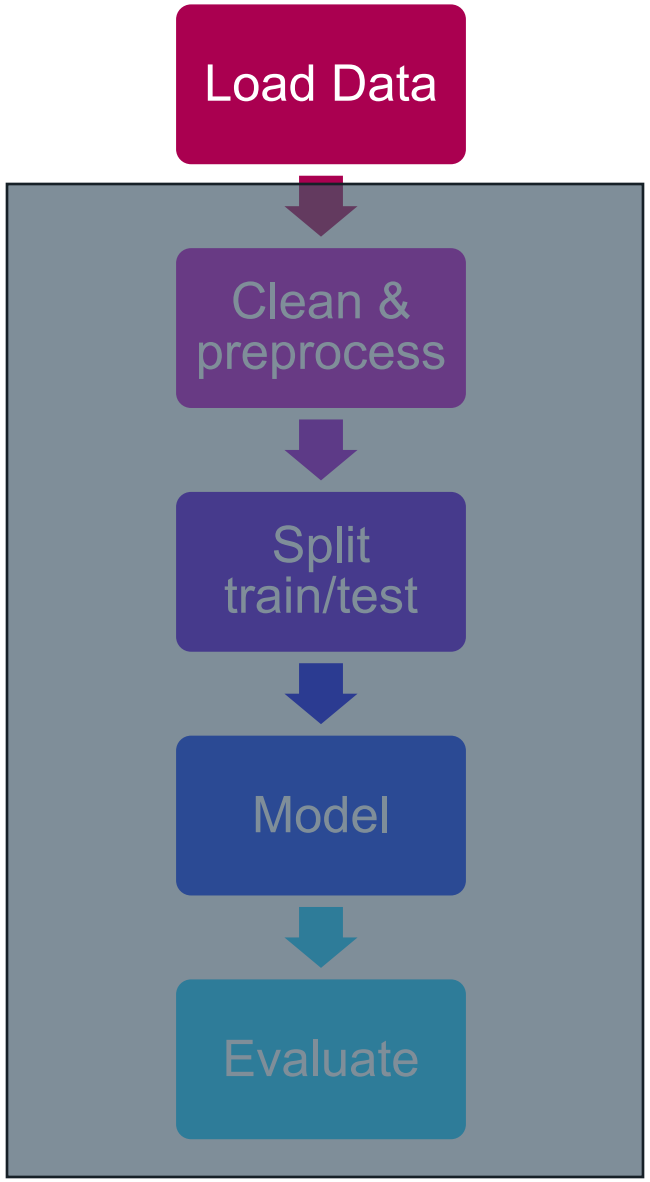
Machine Learning Coding

"Machine Learning — The Computer Learns the Rules"

```python
from sklearn.linear_model import LogisticRegression
import numpy as np

# Training data: [hours studied]
X = np.array([[1], [2], [3], [4], [5]])
y = np.array([0, 0, 0, 1, 1])  # 1 = Pass, 0 = Fail

model = LogisticRegression()
model.fit(X, y)

# Predict for a new student
print(model.predict([[2.5]]))  # -> [0] (Fail)
print(model.predict([[4.5]]))  # -> [1] (Pass)
```

In traditional coding, I *program the rule*.
In machine learning, I *show examples* and the computer *discovers* the rule."

Machine Learning's pipeline



UK Dementia Research Institute

Get Dataset: Voice recording people with Parkinson's disease

```
# UCI Parkinson's dataset (small and open)
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/parkinsons/
```

Output:

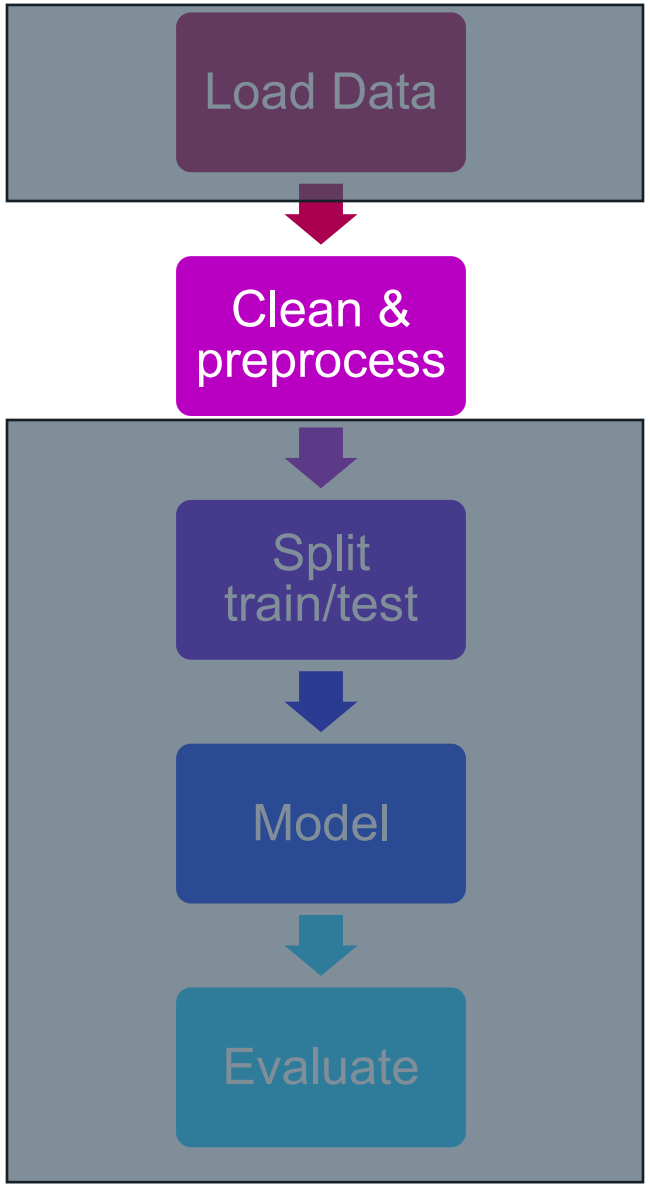
name	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Shimmer	NHR	HNR	status
phon_R01_S01_1	119.992	157.302	74.997	0.00784	0.04374	0.02211	21.033	1
phon_R01_S01_2	122.400	148.650	113.819	0.00968	0.06134	0.01958	19.085	1
phon_R01_S01_3	116.682	131.111	111.555	0.01050	0.05982	0.02144	20.651	1
phon_R01_S01_4	116.676	137.871	111.366	0.00997	0.06400	0.02161	20.644	1
phon_R01_S01_5	116.014	141.781	110.655	0.01284	0.06426	0.02085	20.419	1

Get Dataset: Voice recording people with Parkinson's disease

- **Rows:** 195 voice recordings (not 195 different people — some subjects have multiple recordings).
- **Columns:** 23 biomedical voice measures extracted from the speech signal.
- **Target variable:**
  - `status = 1` → the subject has Parkinson's disease
  - `status = 0` → the subject is healthy (control)

`MDVP:Fo(Hz)`	Average vocal fundamental frequency (mean pitch)
`MDVP:Fhi(Hz)`	Maximum vocal fundamental frequency
`MDVP:Flo(Hz)`	Minimum vocal fundamental frequency
`MDVP:Jitter(%)`, `MDVP:Shimmer`	Measures of variation in frequency and amplitude — how "shaky" the voice is
NHR	Noise-to-harmonics ratio — more noise indicates voice instability
HNR	Harmonics-to-noise ratio
RPDE, D2, PPE	Nonlinear dynamic features that capture irregularities in the voice signal

Machine Learning's pipeline

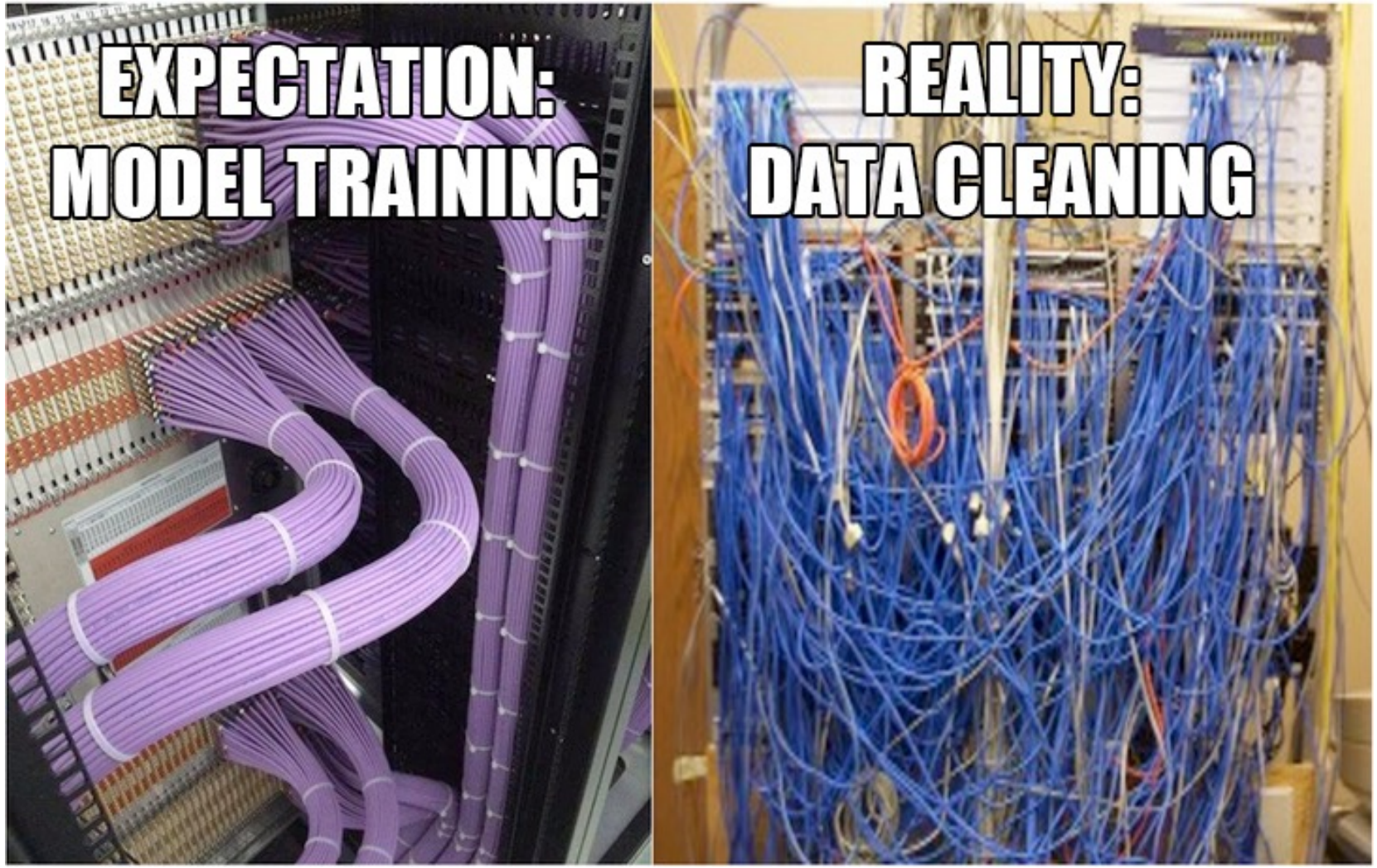


UK Dementia Research Institute

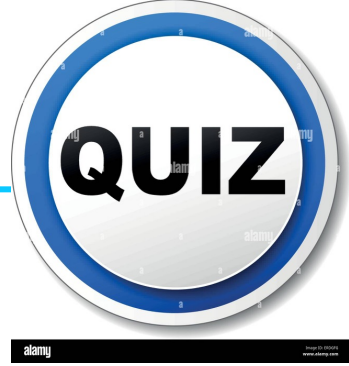
Clean Dataset: Remove duplicates, Fill missing values, Standardize

```python
import pandas as pd
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler

# Example: df is your DataFrame
# ------------------------------------

# 1️⃣ Remove duplicates
df = df.drop_duplicates()

# 2️⃣ Fill missing values (numerical columns → replace NaN with median)
num_cols = df.select_dtypes(include=["float64", "int64"]).columns
imputer = SimpleImputer(strategy="median")
df[num_cols] = imputer.fit_transform(df[num_cols])

# 3️⃣ Standardize numeric columns (mean = 0, std = 1)
scaler = StandardScaler()
df[num_cols] = scaler.fit_transform(df[num_cols])
```

Garbage in → Garbage out



What Happens If We Skip Data Cleaning?



Q1 – Duplicates
You trained a Parkinson's model and got 98% accuracy, but half the rows were duplicated.
👉 What happens when you test on new patients?

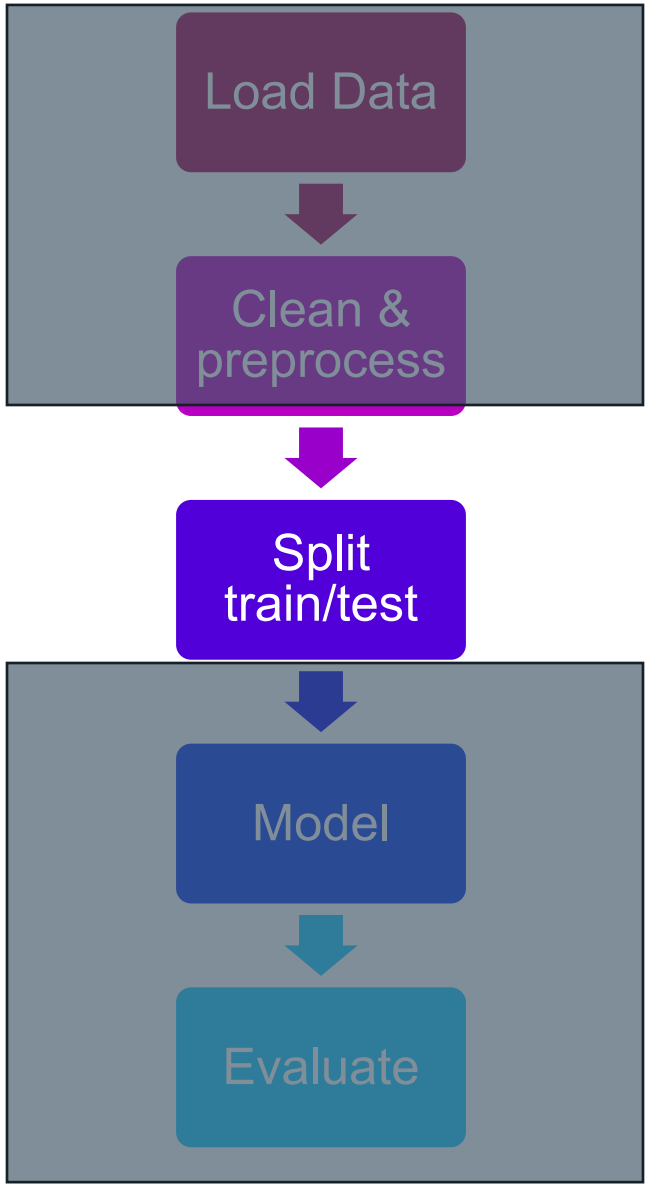
Q2 – Missing Values
Some voice features are missing for several recordings.
👉 What happens if you feed NaNs directly into the model?

Q3 – Unscaled Features One feature ranges 0–0.01 (Jitter%) and another 0–200 (F0 Hz).
👉 Without scaling, which feature dominates the model?

Q4 – Outliers A single subject's "frequency = 2000 Hz" is clearly wrong.
👉 How could this affect your model?

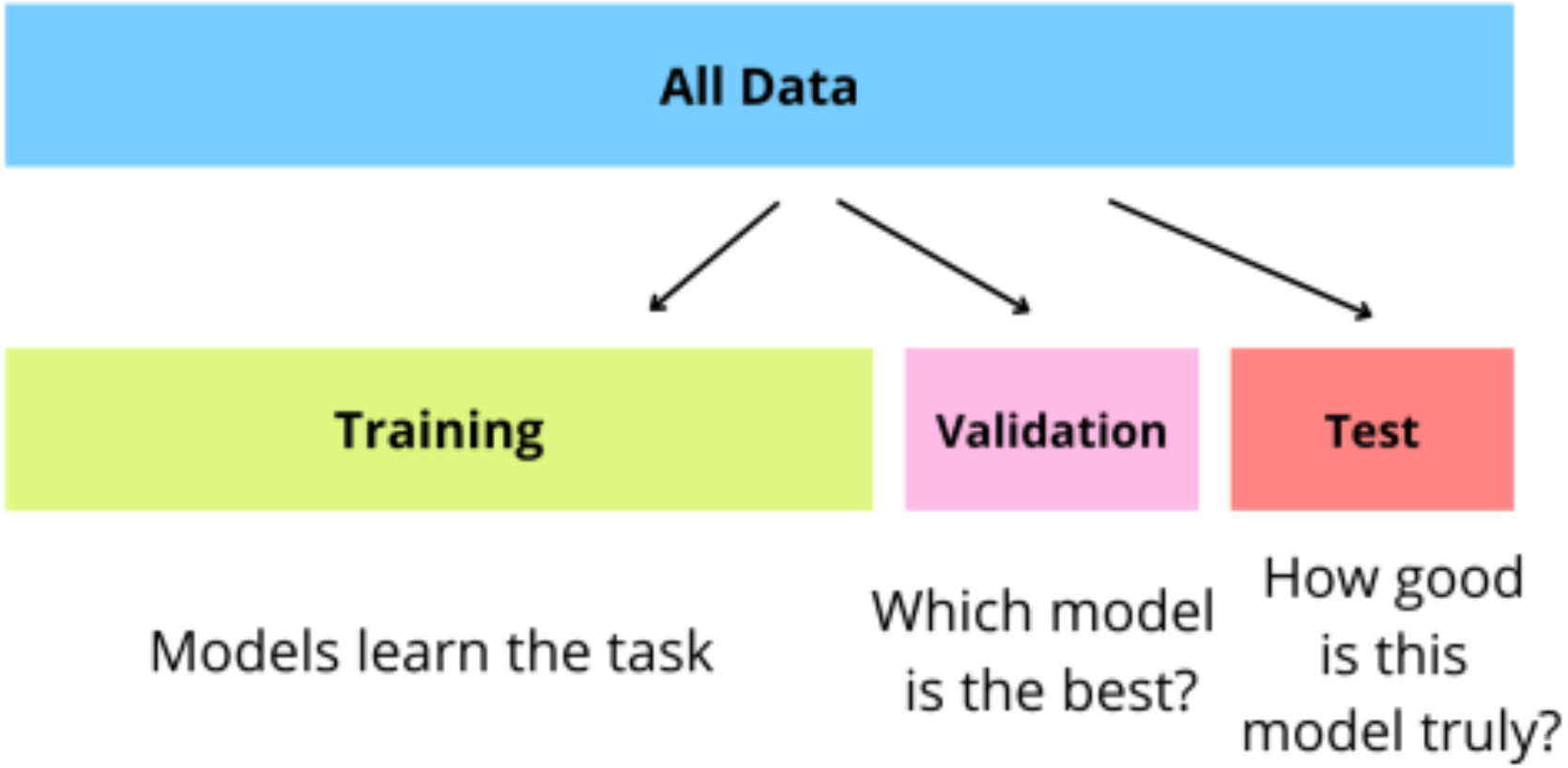
Q5 – Inconsistent Formatting The "Age" column mixes numbers and the string "unknown."
👉 What happens during preprocessing?

Machine Learning's pipeline



UK Dementia Research Institute

Train, Validation, and Test — What's the Difference?



Predicting Age from Tremor Severity

Let’s say we have data from 10 Parkinson’s patients:

Patient	Tremor Score (X)	Age (Y)
1	2	45
2	3	47
3	4	50
4	5	53
5	6	55
6	7	60
7	8	62
8	9	65
9	10	66
10	11	70

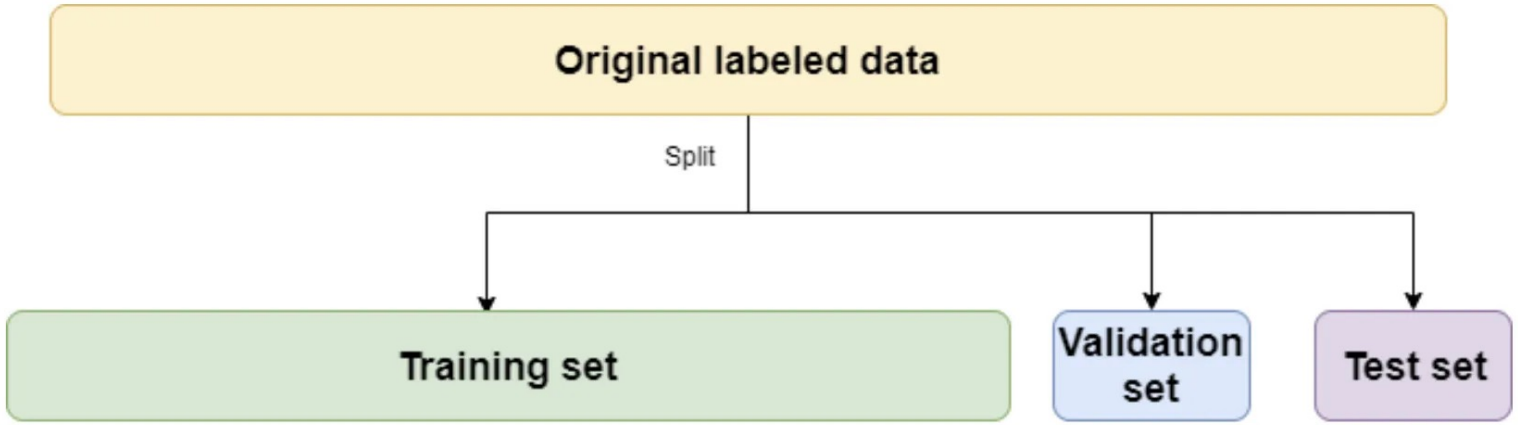
We want to find a simple line:

$$\hat{Y} = a + bX$$

that predicts **age (Y)** from **tremor score (X)**.

UK Dementia Research Institute

Step1: Split your dataset



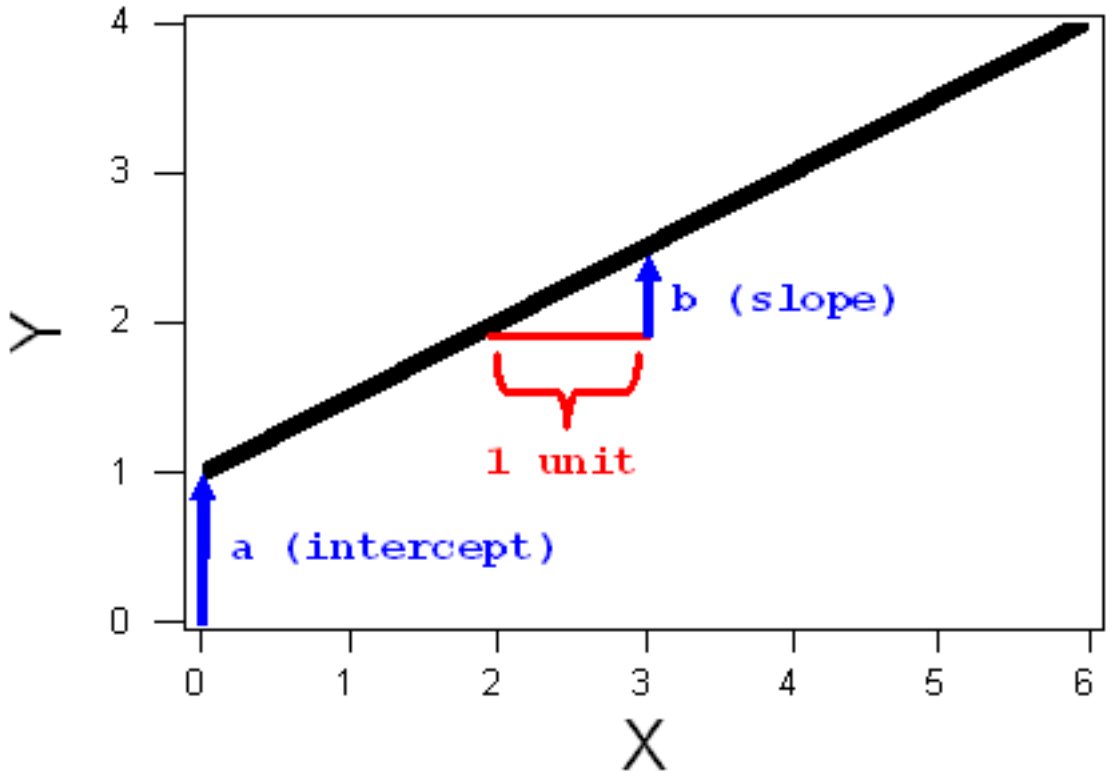
Set	Patients	Purpose
Training	1–6	Learn a and b
Validation	7–8	Tune model (check performance)
Test	9–10	Final evaluation

Step2: Training

On the **training set (1–6)** we compute the best-fit line.

Let's say the model learns:

$$\hat{Y} = 40 + 3X$$



Step3: Validation

Now we check how well this line predicts for **patients 7–8** (not used in training).

X (Tremor)	True Y (Age)	Predicted Y = 40 + 3X	Error
8	62	64	+2
9	65	67	+2

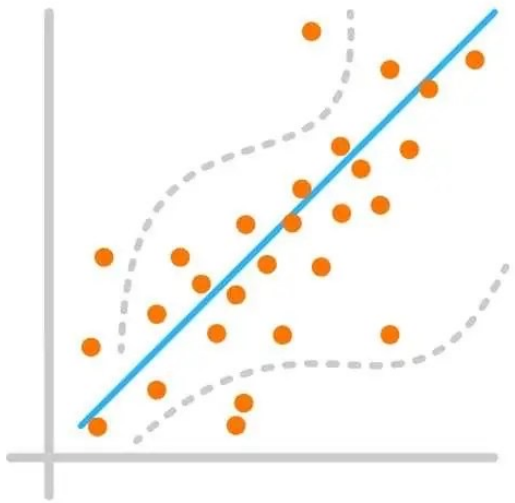
Average validation error ≈ **2 years** → good!

But suppose we tried a slightly different model,

$\hat{Y} = 42 + 2.5X$

Validation error becomes **5 years** → worse.

So we keep the first model (40 + 3X).

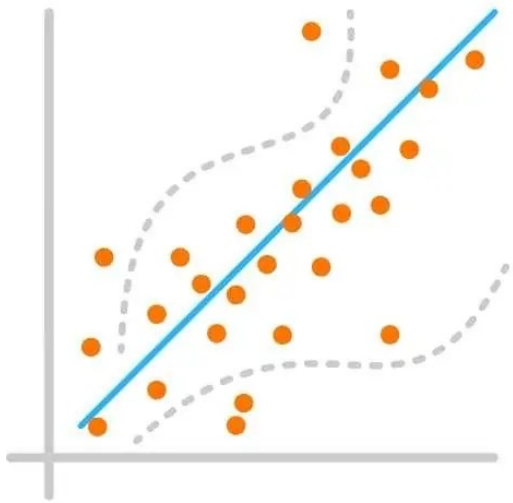


Step4: Test

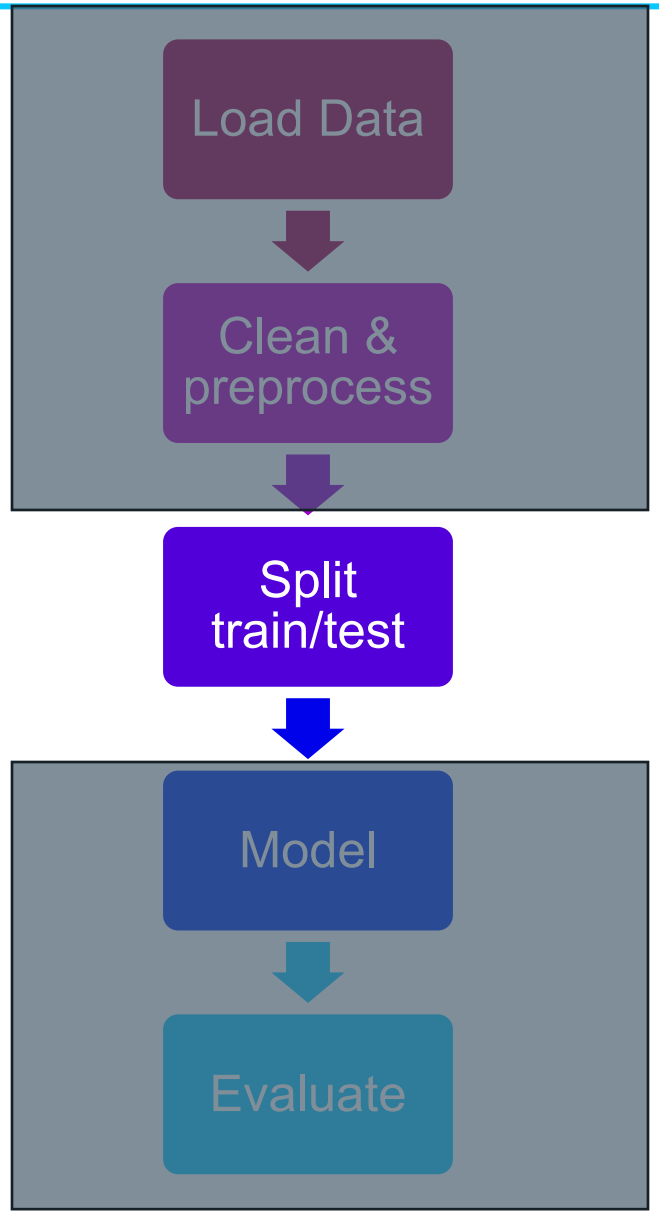
Now test the final model on **patients 9–10** (unseen data).

X	True Y	Predicted Y = 40 + 3X	Error
10	66	70	+4
11	70	73	+3

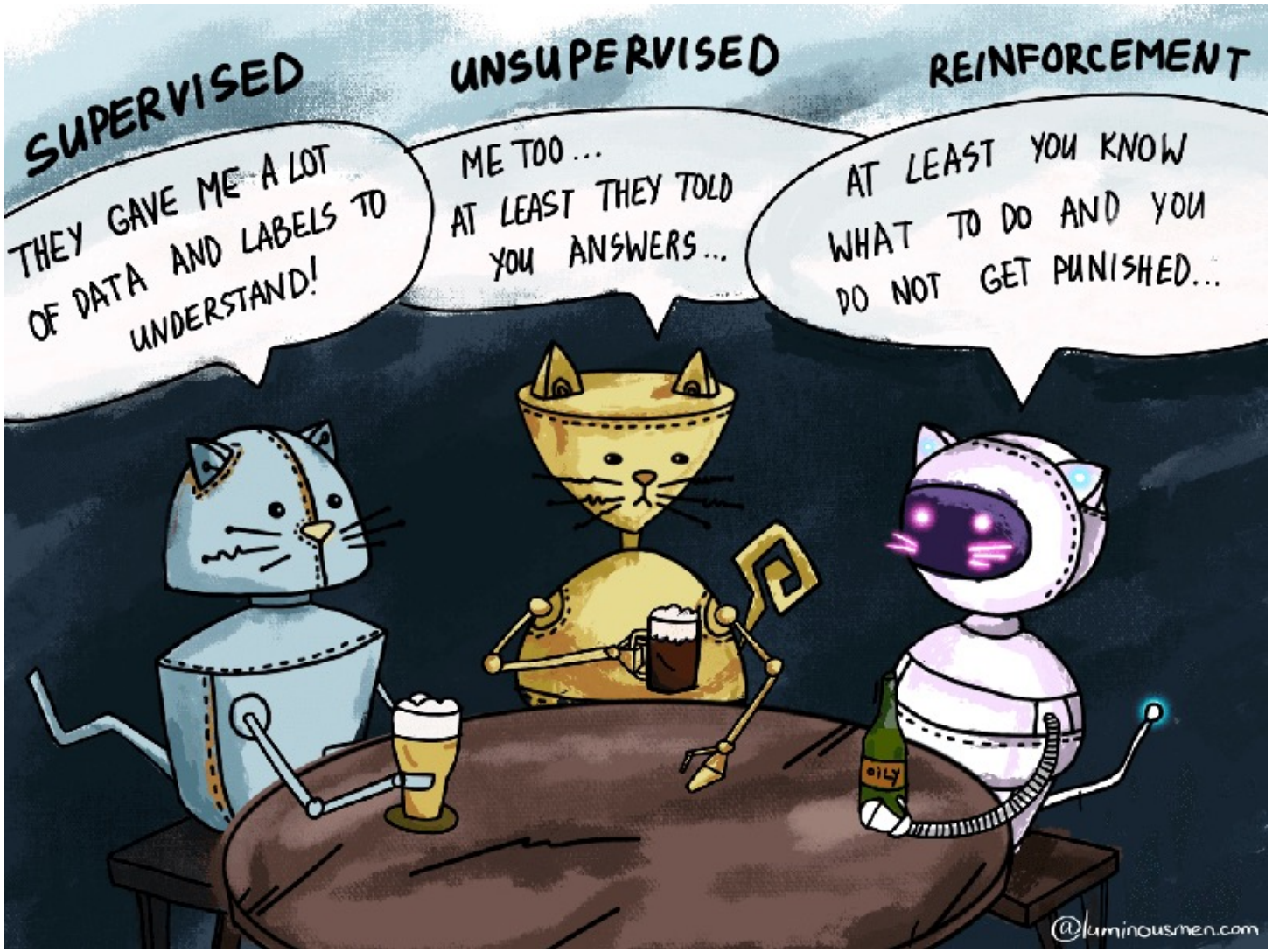
Average test error ≈ **3.5 years** → realistic estimate of how well model performs on new patients.



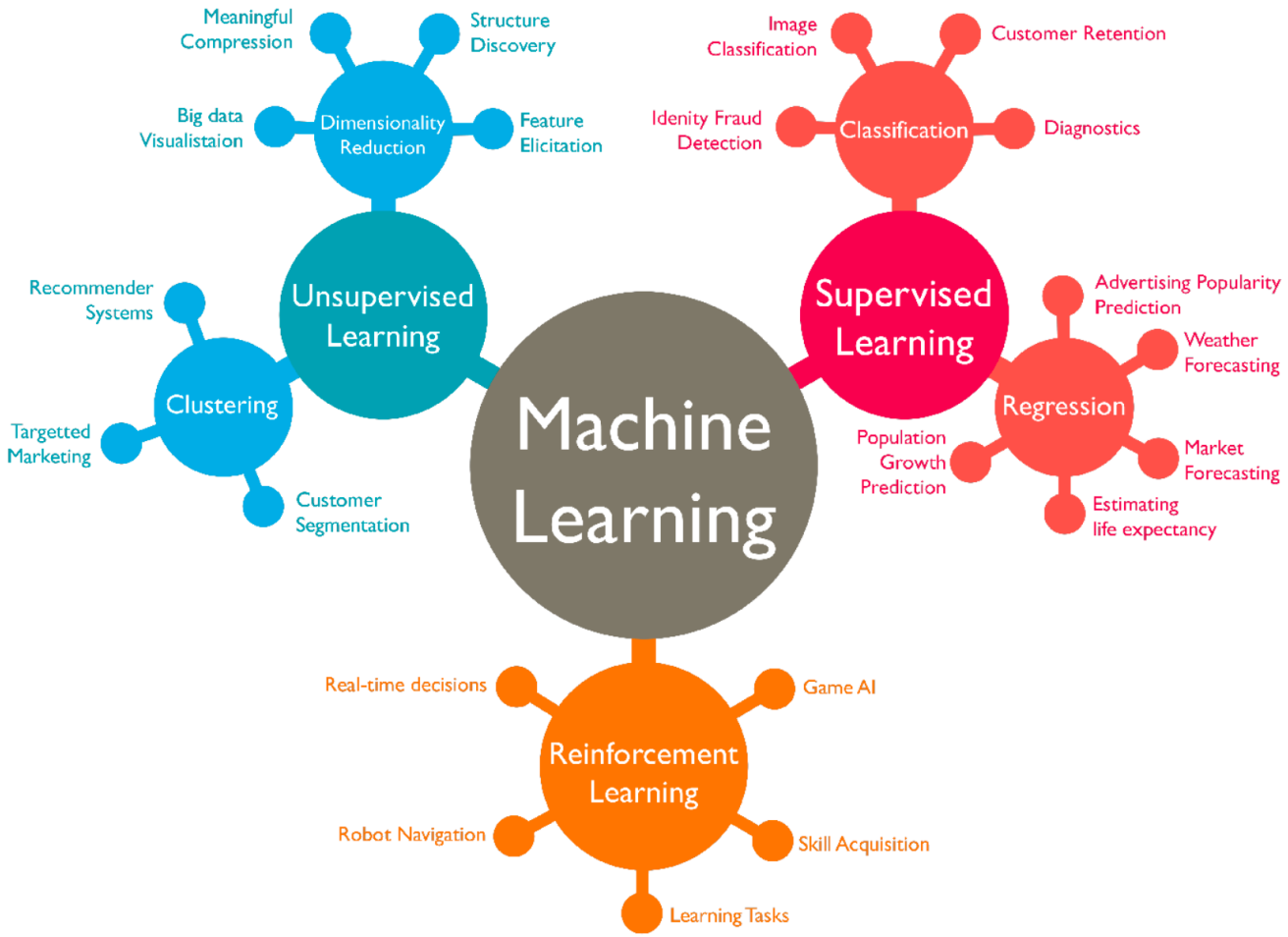
Machine Learning's pipeline



Types of Machine Learning Model



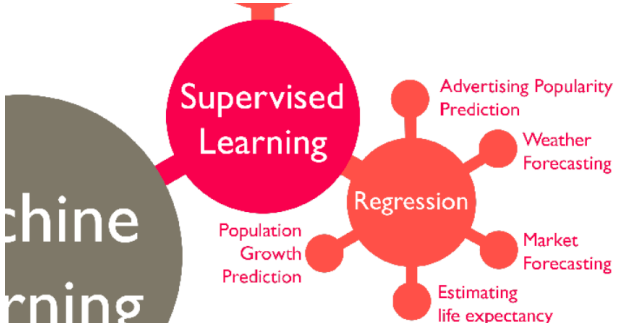
Types of Machine Learning Model

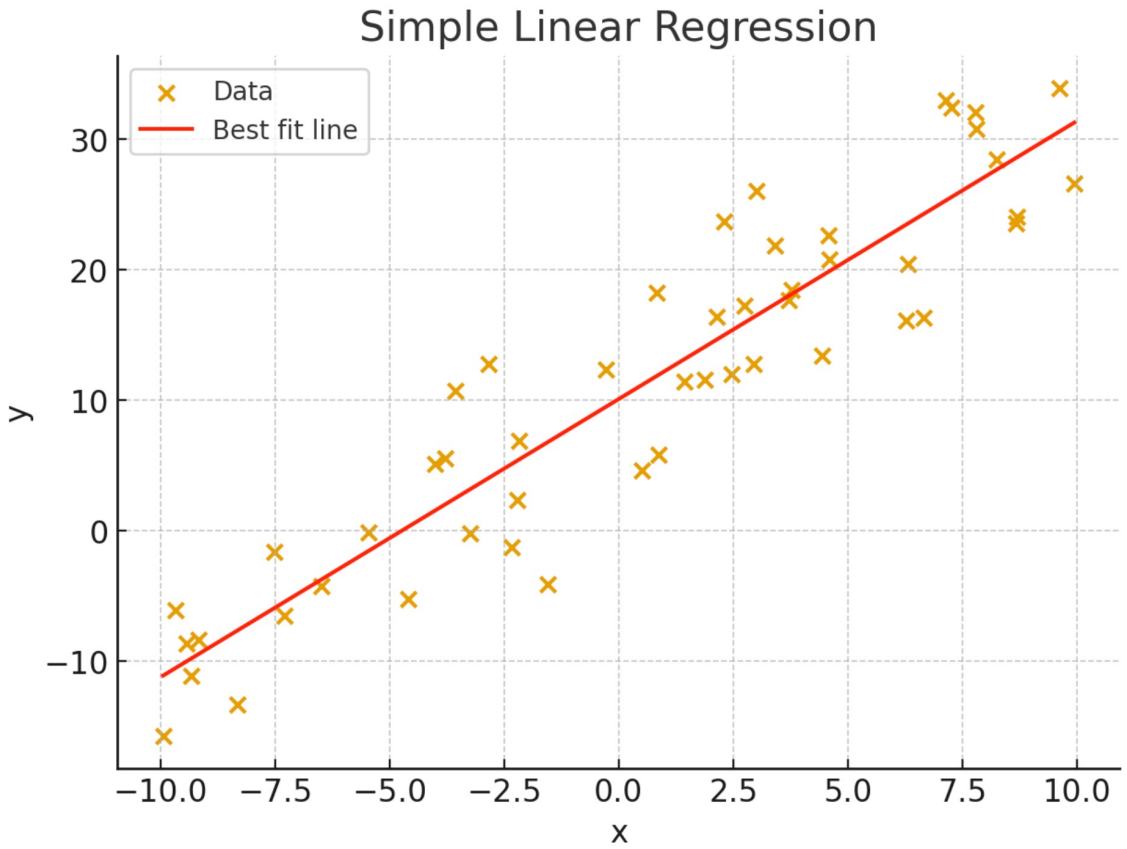


UK Dementia Research Institute

Supervised ML: Regression

The **simplest and most widely known machine learning method** is **Linear Regression**.





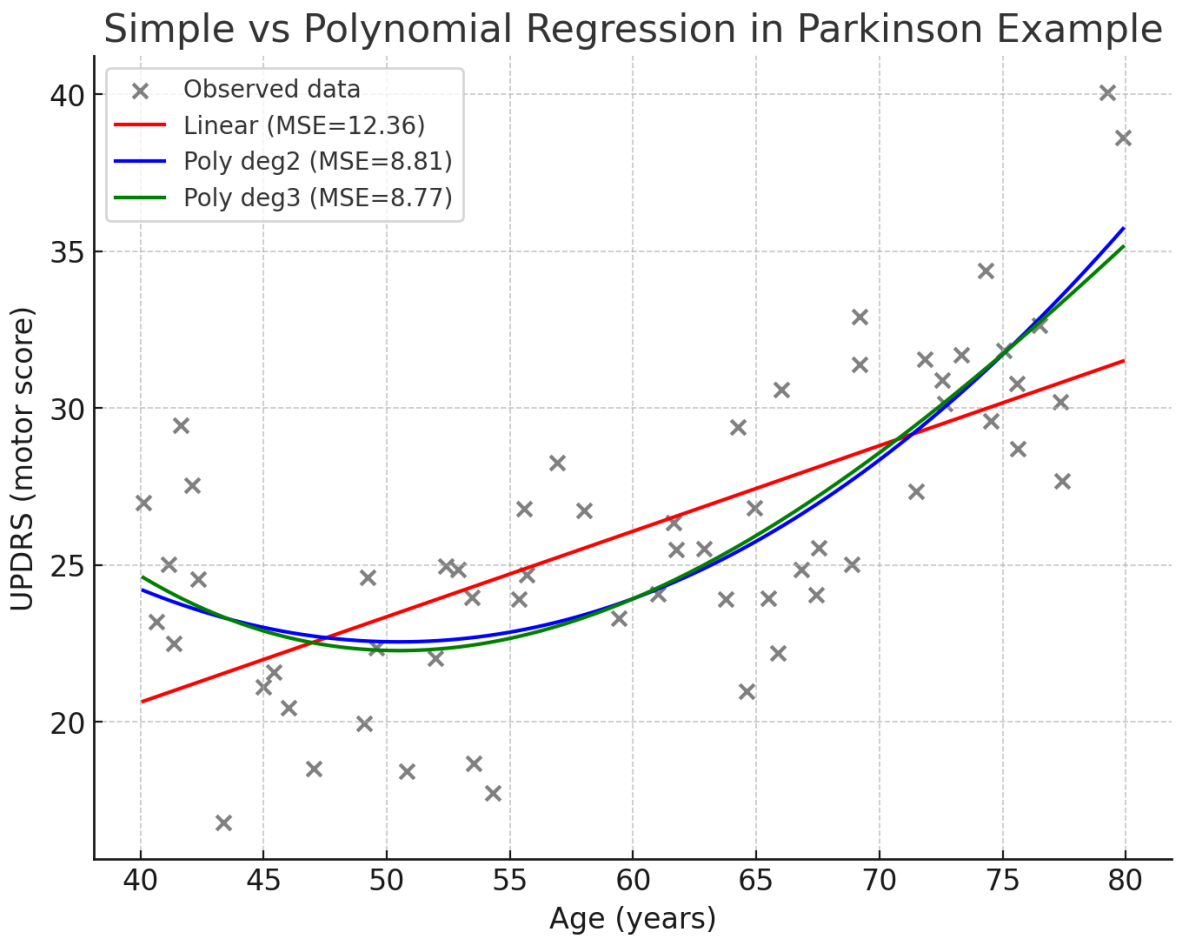
$y = 10 + 2.13x$

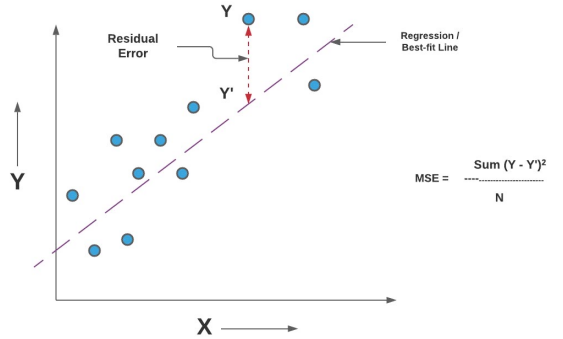
Predict **motor severity (UPDRS score)** using **age**:

UK Dementia Research Institute

Supervised ML: Simple Regression vs Polynomial regression

$$y = 10.26 + 2.13x + 0.99x^2)$$

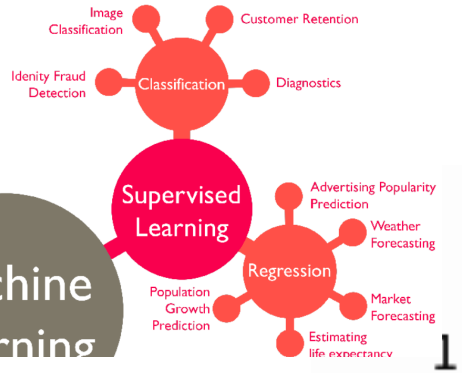




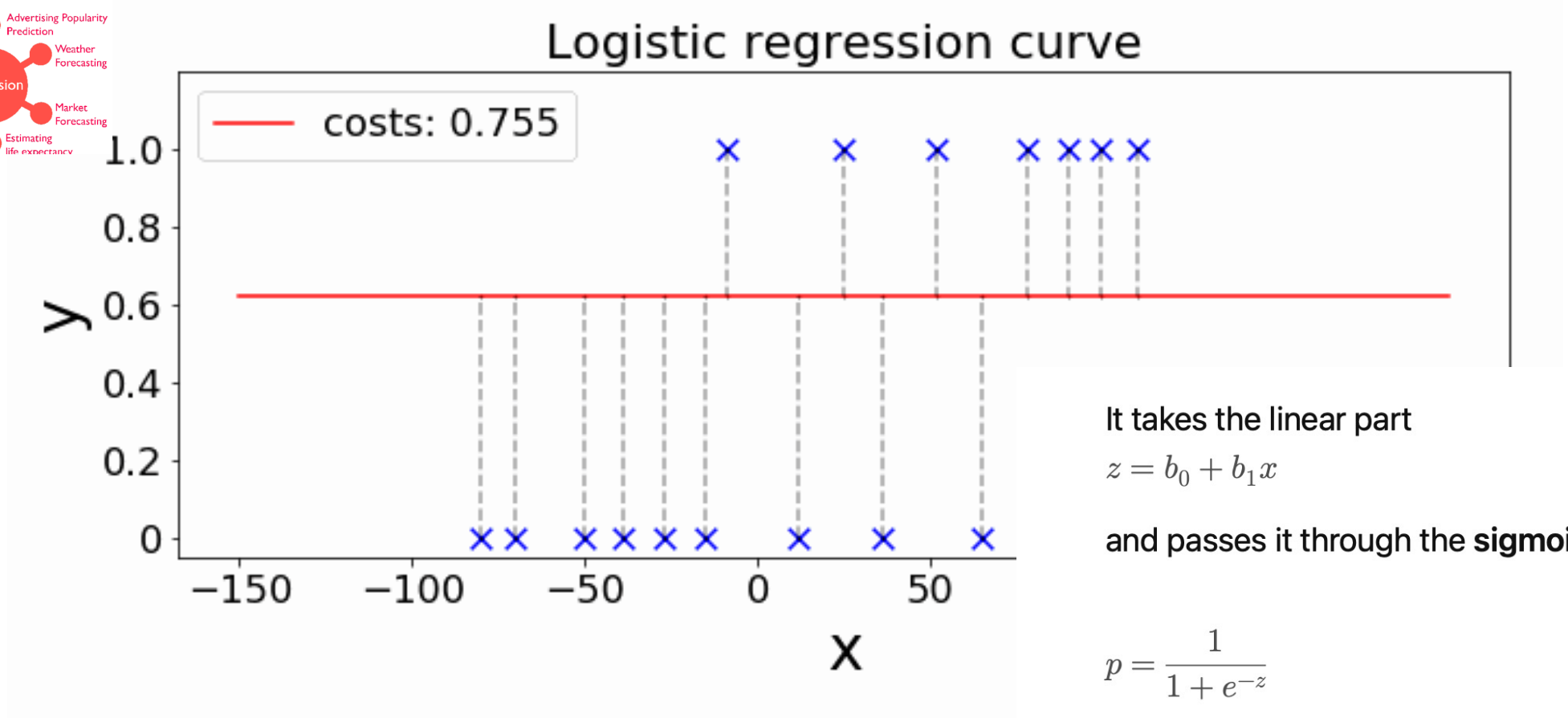
MSE stands for **Mean Squared Error** — it's one of the most common ways to **measure how well a regression model fits the data**.

UK Dementia Research Institute

Supervised ML: Classification



Linear regression predicts a value.
Logistic regression predicts a **probability** (then a class).



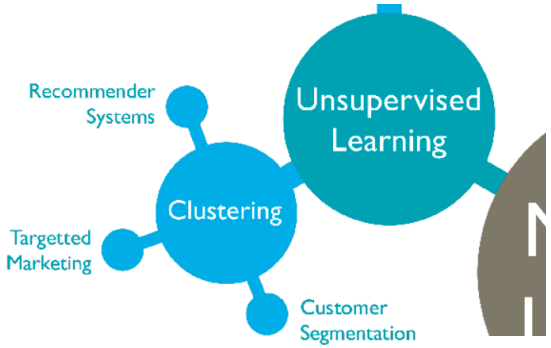
It takes the linear part

$$z = b_0 + b_1 x$$

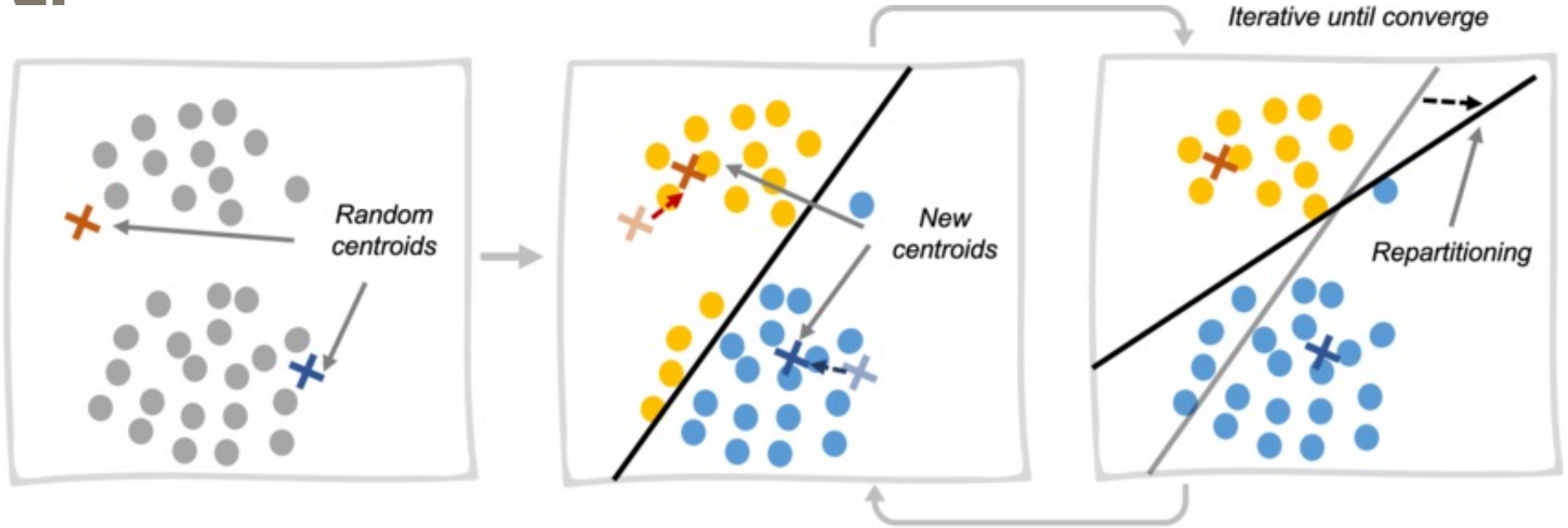
and passes it through the **sigmoid function**:

$$p = \frac{1}{1 + e^{-z}}$$

Unsupervised ML: K-means Clustering



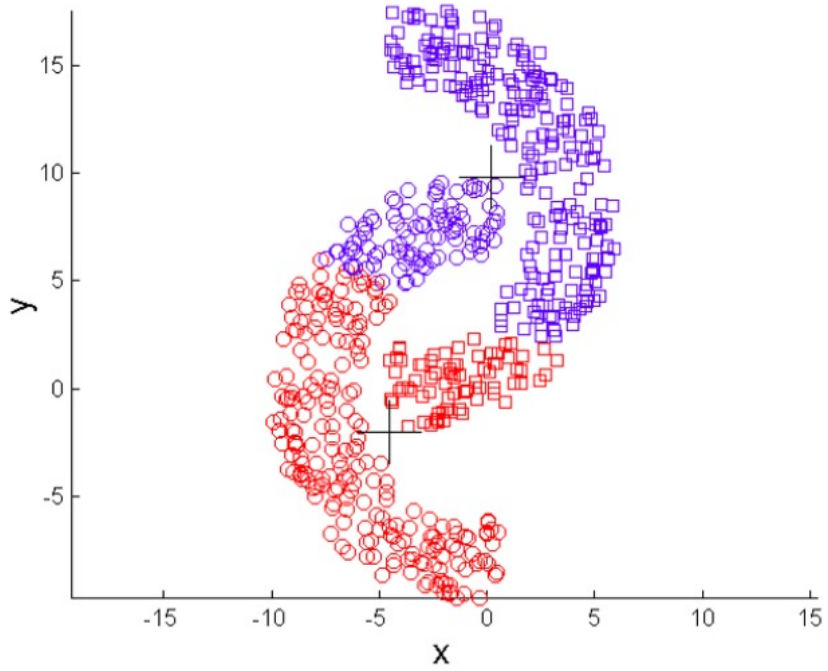
Application Identify **disease subtypes**



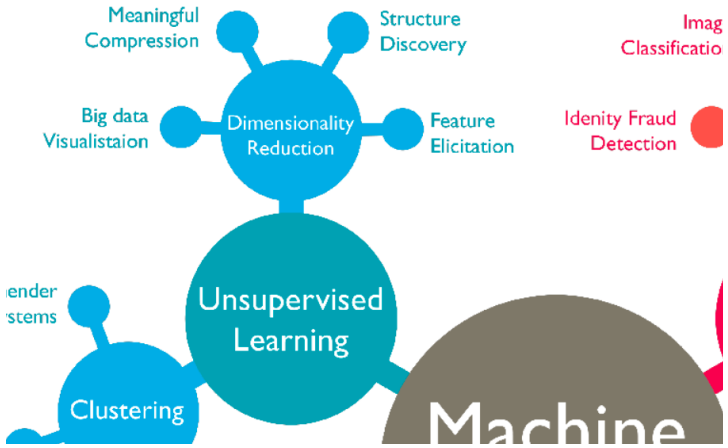
Input: distance matrix *D* & number of clusters *k*

Unsupervised ML: K-means Clustering

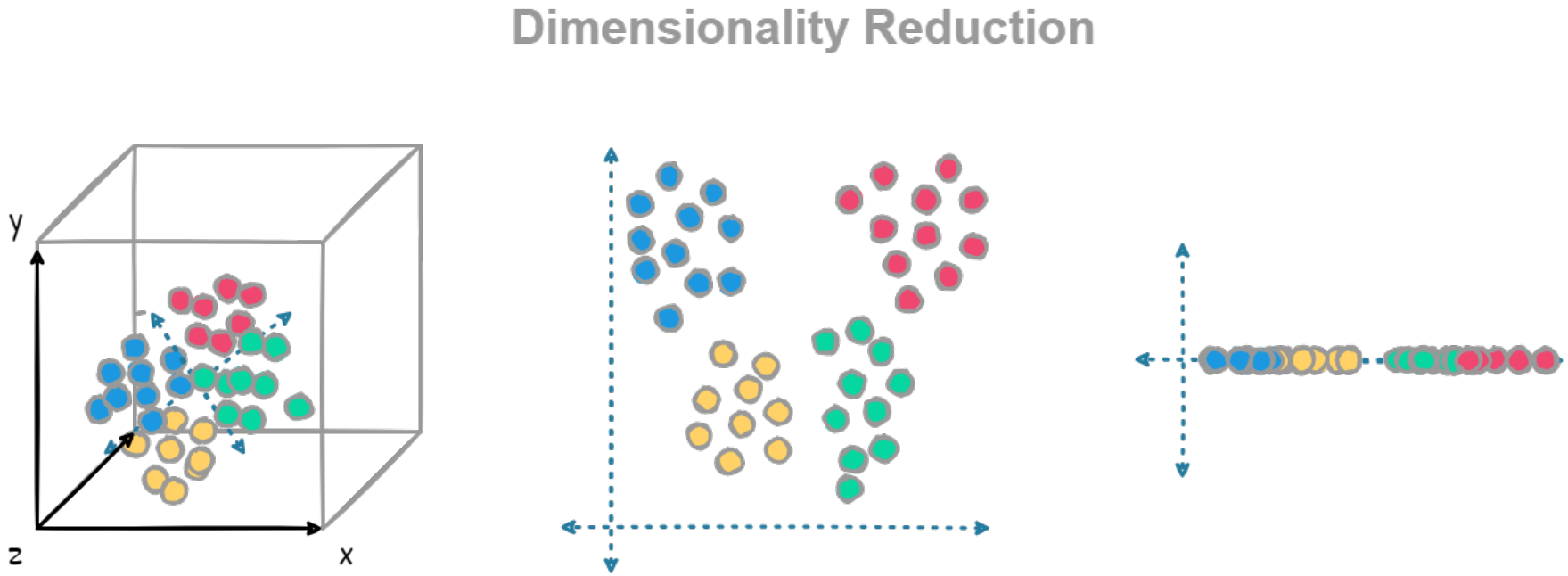
- You must choose k in advance (elbow method)
- It assumes clusters are spherical and equally sized
- It is sensitive to outliers
- It only works with numeric data
- It depends on initialization
- It assumes every point belongs to one cluster



Unsupervised ML: Dimensionality Reduction



Transforms data into a **lower-dimensional space** to simplify analysis and visualization.

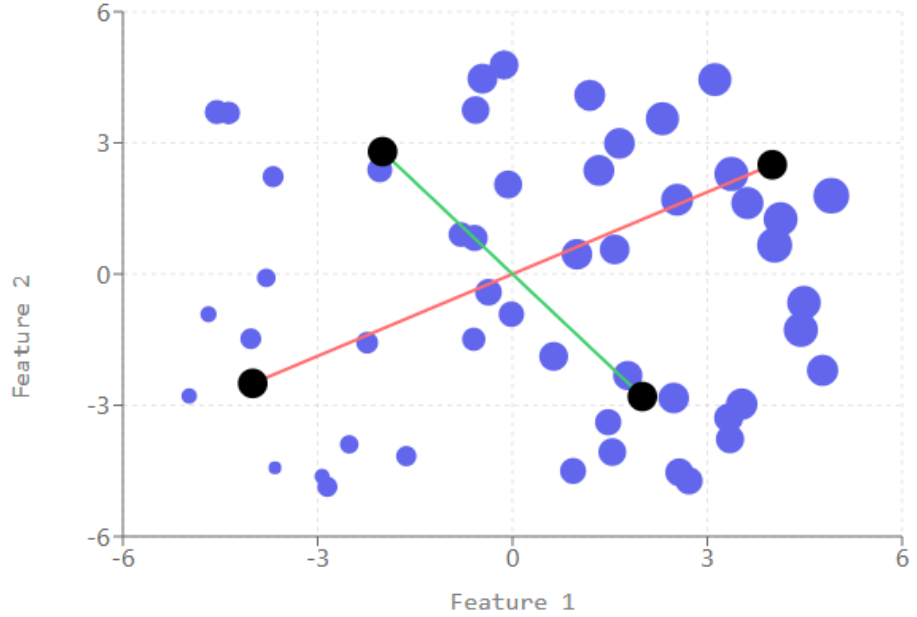


UK Dementia Research Institute

Unsupervised ML: Principal Component Analysis

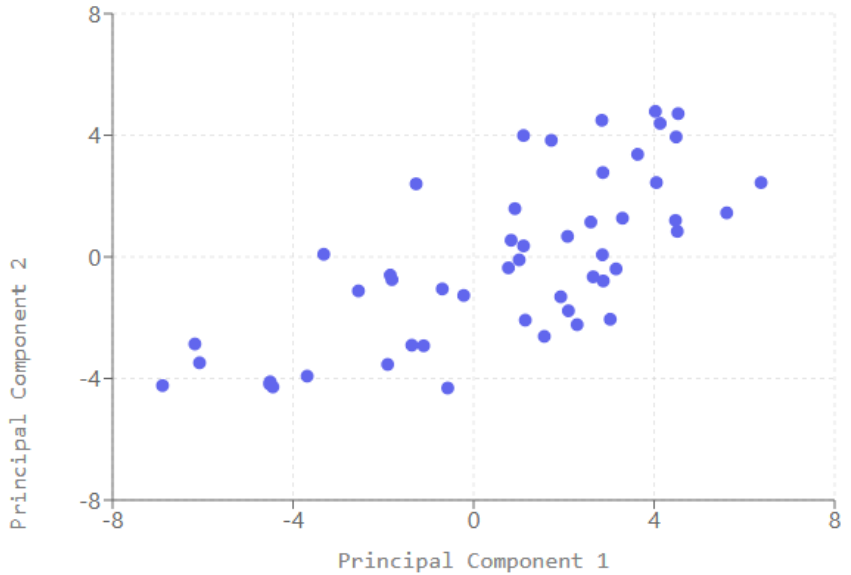
Transforms a set of correlated variables into a set of uncorrelated variables called principal components.

→ Finding Principal Components



✎ *PCA finds the directions of maximum variance in the data*

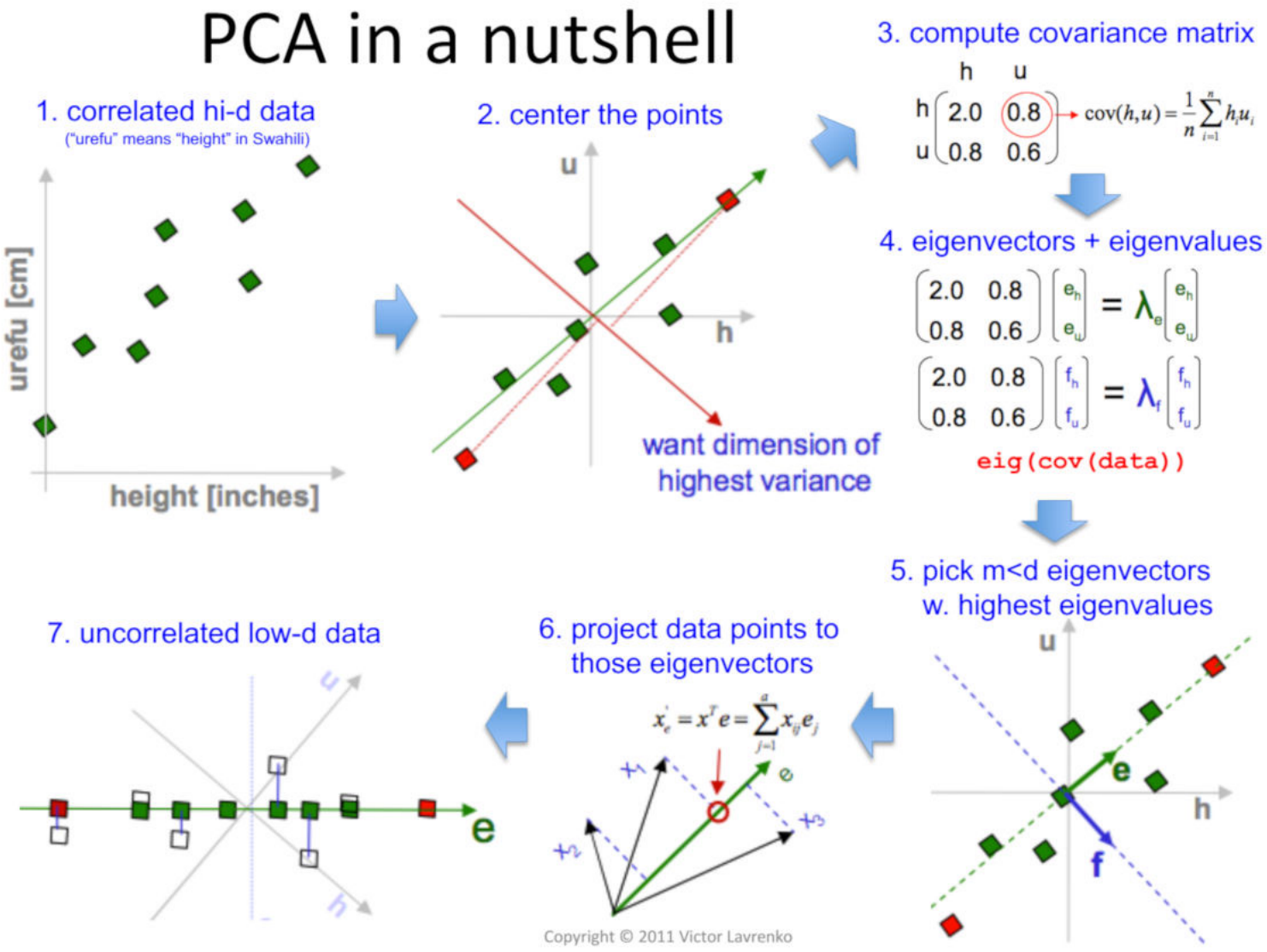
→ Projected 2D Data (After PCA)



✎ *Data is now represented in 2D while preserving most of the structure*

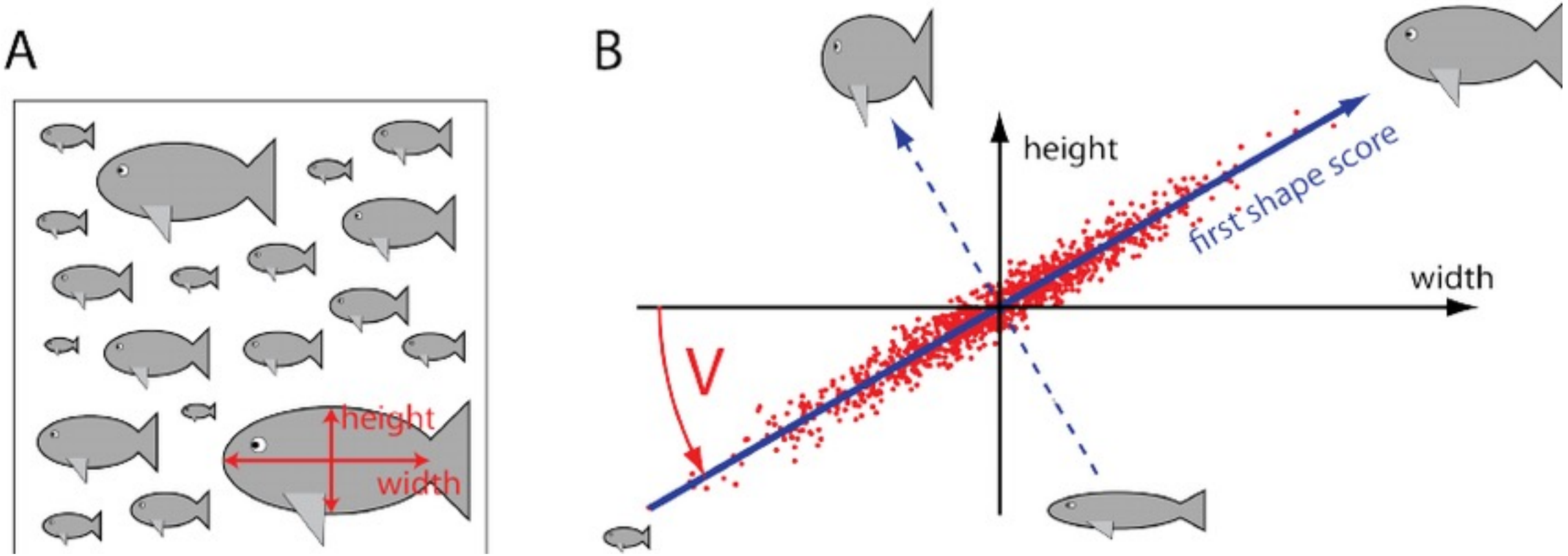
UK Dementia Research Institute

Unsupervised ML: Principal Component Analysis

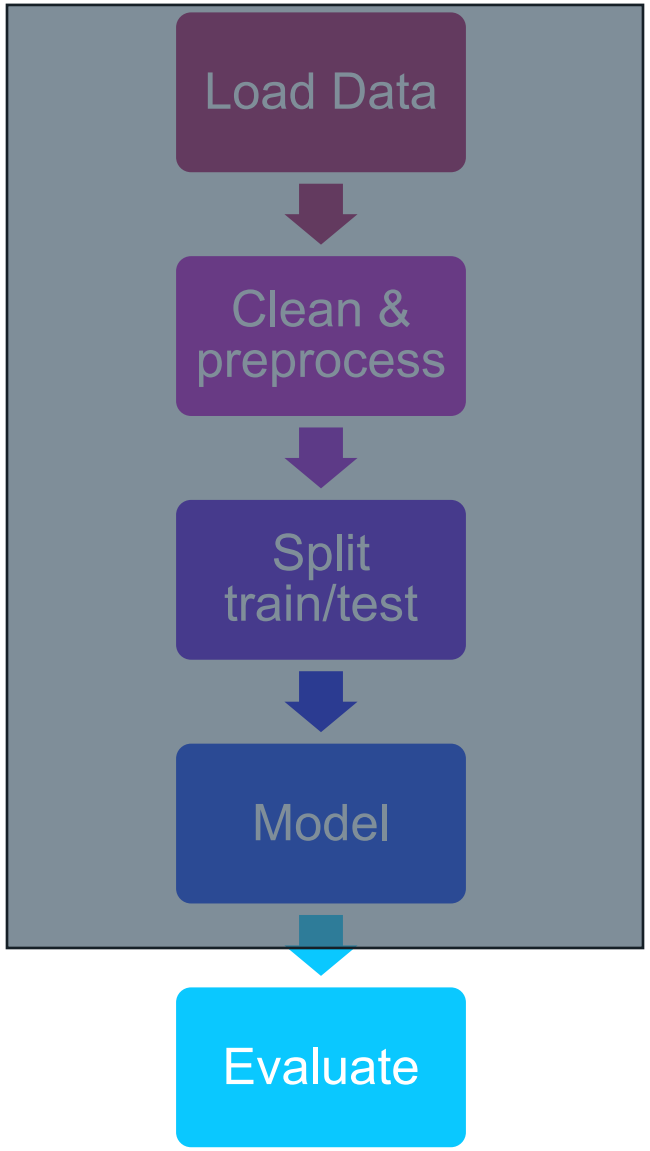


UK Dementia Research Institute

Unsupervised ML: Principal Component Analysis

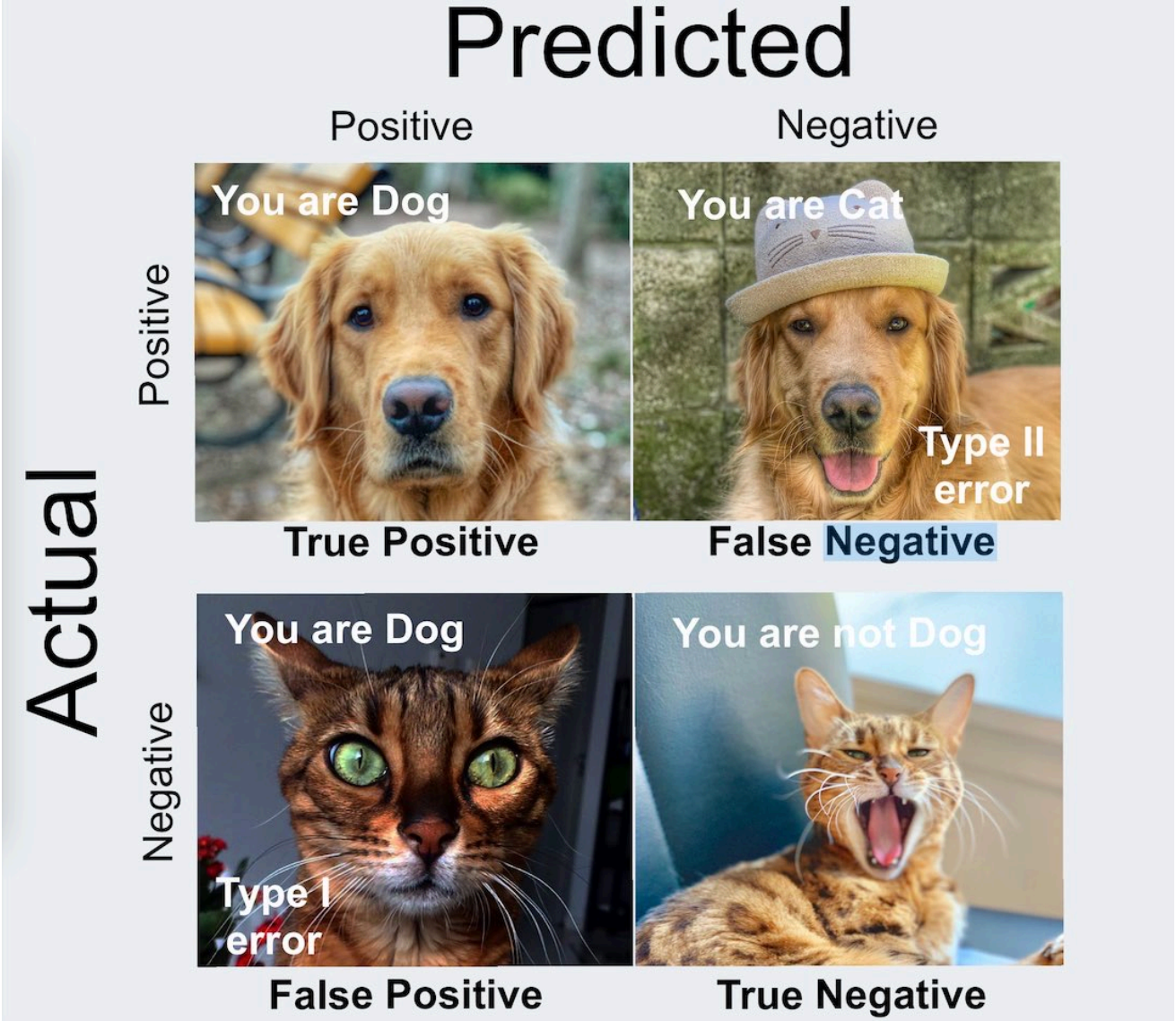


Machine Learning's pipeline



UK Dementia Research Institute

Confusion matrix



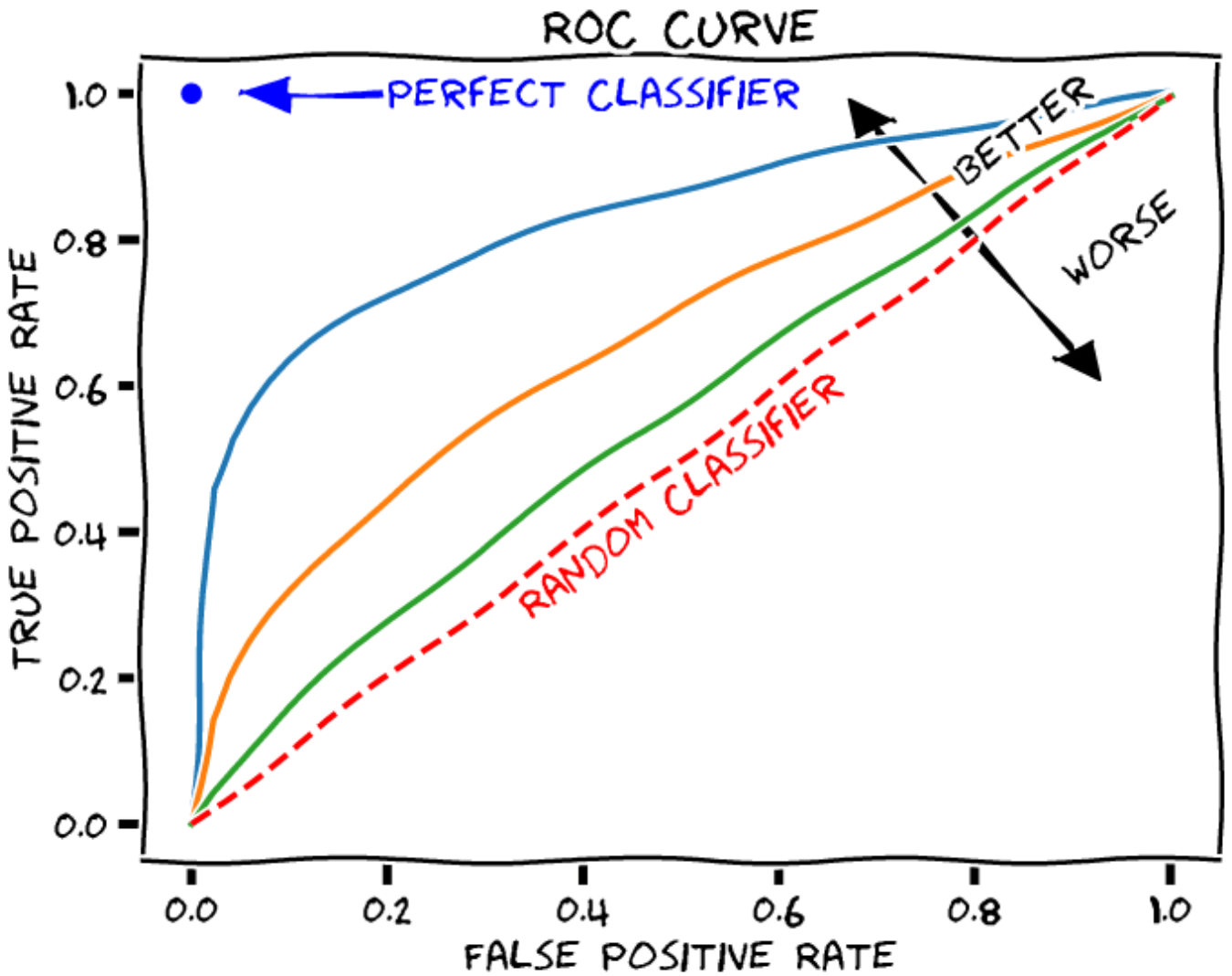
A **confusion matrix** is a table used to measure how well a classification model performs

What is Precision, Recall, and F1-score?

Key: **tp** = True Positive, **tn** = True Negative, **fp** = False Positive, **fn** = False Negative

Metric Name	Metric Forumla	Code	When to use
Accuracy	$\textbf{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn}$	`tf.keras.metrics.Accuracy()` or `sklearn.metrics.accuracy_score()`	Default metric for classification problems. Not the best for imbalanced classes.
Precision	$\textbf{Precision} = \frac{tp}{tp + fp}$	`tf.keras.metrics.Precision()` or `sklearn.metrics.precision_score()`	Higher precision leads to less false positives.
Recall	$\textbf{Recall} = \frac{tp}{tp + fn}$	`tf.keras.metrics.Recall()` or `sklearn.metrics.recall_score()`	Higher recall leads to less false negatives.
F1-score	$\textbf{F1-score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$	`sklearn.metrics.f1_score()`	Combination of precision and recall, usually a good overall metric for a classification model.
Confusion matrix	NA	`Custom function` or `sklearn.metrics.confusion_matrix()`	When comparing predictions to truth labels to see where model gets confused. Can be hard to use with large numbers of classes.

ROC Score or Area under ROC curve



Evaluate the model

	Accuracy	Precision	Recall	F1 Score	ROC Score
Support Vector Machines	0.894255	0.886179	0.905316	0.895645	0.894227
Bagging	0.899251	0.884800	0.918605	0.901385	0.899202
Random Forests	0.903414	0.905000	0.901993	0.903494	0.903417
XG Boost	0.910908	0.909091	0.913621	0.911350	0.910901
Adaptive Boosting	0.913405	0.913621	0.913621	0.913621	0.913405
Gradient Boosting	0.913405	0.906863	0.921927	0.914333	0.913384
K-Nearest Neighbors	0.861782	0.803621	0.958472	0.874242	0.861540
Multi-layer Perceptron	0.905912	0.900164	0.913621	0.906843	0.905892
Linear Discriminant Analysis	0.848460	0.824074	0.887043	0.854400	0.848363
Quadratic Discriminant Analysis	0.855953	0.826484	0.901993	0.862589	0.855838
Gaussian Naïve Bayes Classifier	0.823480	0.776989	0.908638	0.837672	0.823267

	Accuracy	Precision	Recall	F1 Score	ROC Score
Support Vector Machines	7	6	8	7	7
Bagging	6	7	3	6	6
Random Forests	5	4	9	5	5
XG Boost	3	2	4	3	3
Adaptive Boosting	1	1	4	2	1
Gradient Boosting	1	3	2	1	2
K-Nearest Neighbors	8	10	1	8	8
Multi-layer Perceptron	4	5	4	4	4
Linear Discriminant Analysis	10	9	11	10	10
Quadratic Discriminant Analysis	9	8	9	9	9
Gaussian Naïve Bayes Classifier	11	11	7	11	11

Let's Code

Group 1	Group 2	Group 3	Group 4	Group 5	Group 6	Group 7	Group 8
Philip Press	Viola Ceriani	Duru Okay	Yossra Serroukh	Harsimran Kaur	Jenna Lin	Muireann Hogan	Lucia Jing
Margarida Neves	Anya Kaur Haddon	Chloe Kam	Rebekah Bourne	Luca Pastorello	Mustafa Gunaydin	Edona Bajrami	Zishan Lin
Olivia Pownall	Anais Chia	Wenbo Liao	Hattie Oliver	Luoyuan Zhang	Laura Rakiec	Francesca Murley-Holme	Stephanie Sun
Charlotte Yu	Shobana Chandrashekar	Yunyi Gao	Thishany Kuganeswaran	Ellie Carre	Hanna Aitessaid	Ruben Thiagarajah-Fernar	Neera Gahir
Chun Hei Leung	Katie Hay	Anas Saleem	Hanyue Pang	Veronika Shevchenko	Alisija Dabasinskaite	Felix Varenne	Chuyi Zhang
Andrea Fan	Nina Jeffrey	Chi U Chau	Zeinab Ben Halim	Amina Bououdine	Ema Ferra	Ruofan CAO	Lili Yassin
Adelina Shahata	Asma Abdullahi	YINUO Wang	Isabella Coloru	Krystal Tan	Temi Lana	Xinrui Fan	Keya Tanwani
Tanaka Udugama Jal	Vea Bley	Mari Hronska	Lucas Yebra Garcia	Sarah Kurbanov			

https://github.com/Sandoretal/Module_3/tree/main/tutorial_4