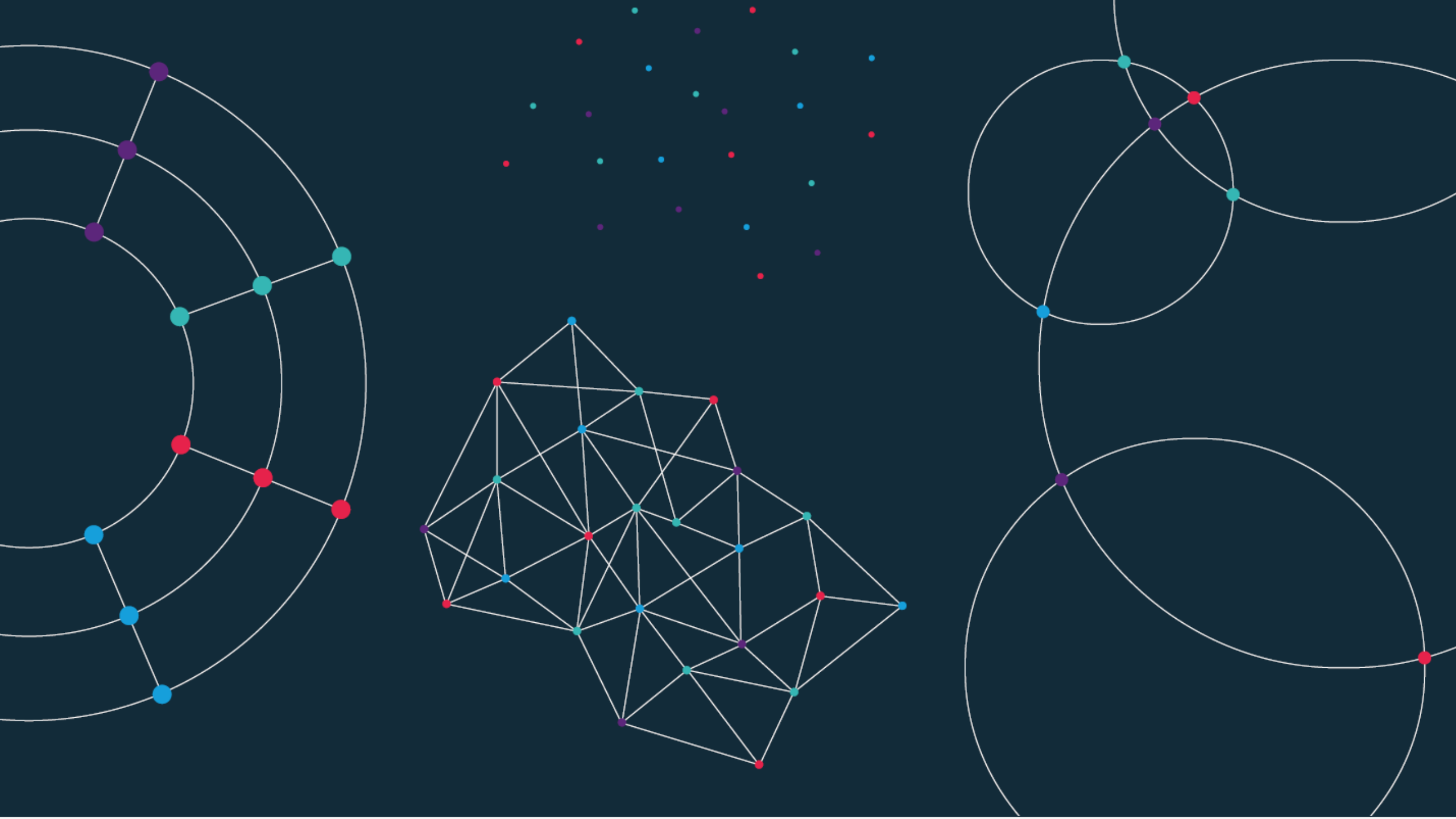
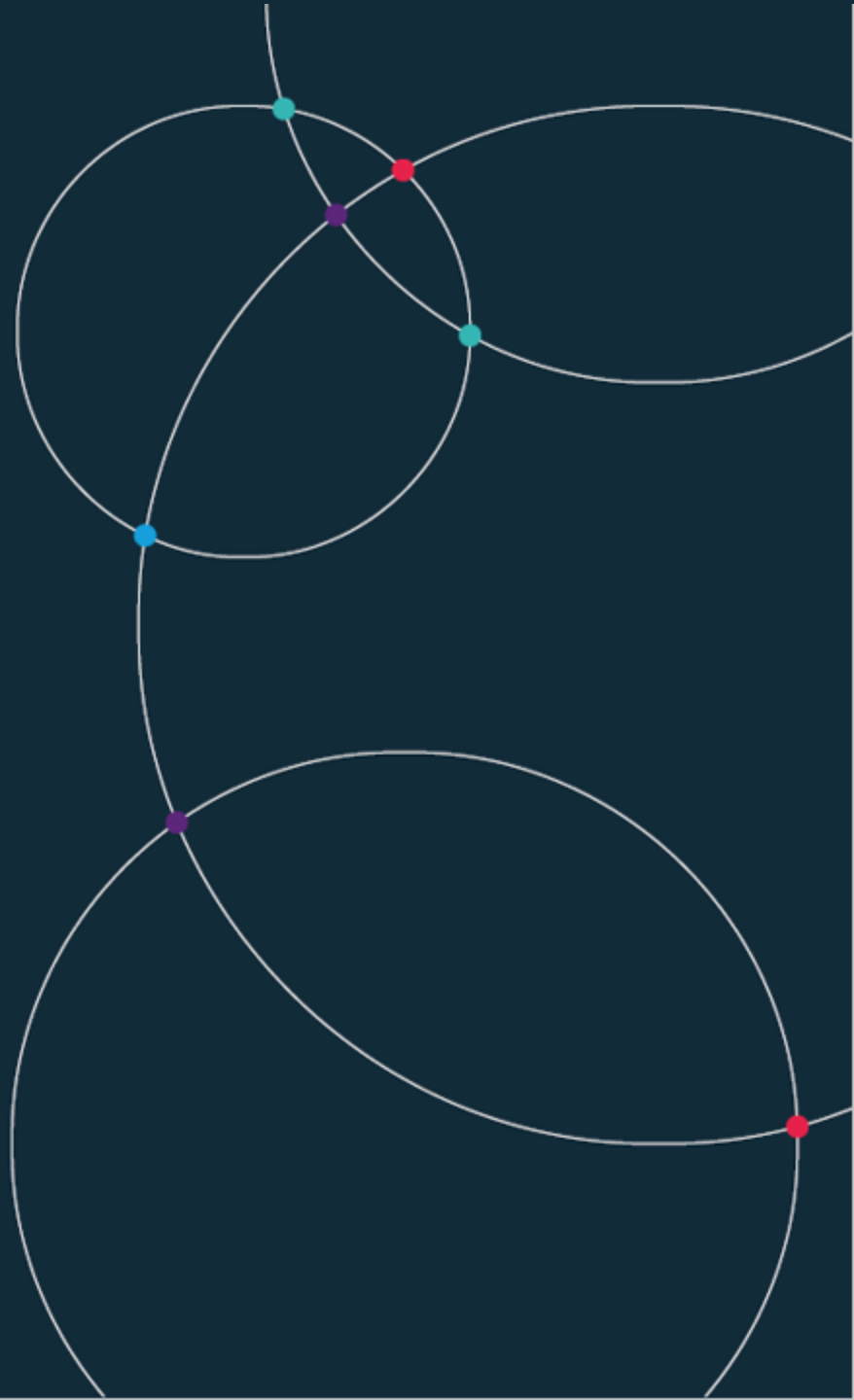


Automated Data Visualisation for Policymaking.

Session 10
Autumn 2025



Resources.

Places to find, and share

Resource 1: my site:
www.richarddavies.io/data-science

Resource 2: chart library.
www.richarddavies.io/library

Resource 3: course Google sheet.
Google sheet. Link.

Resource 4: course DropBox.
DropBox. Link.

Resource 5: Playfair Prize
www.playfairprize.com

Stock take: Nearly there!

Our final two sessions

- W1. Your Live Site. HTML, CSS, JavaScript. GitHub and embedding.
- W2. Data debating – chart design. Raw files, editing data. String and date functions.
- W3. The Language of Data. Why visualisation is vital in policy. Manipulating data with Python.
- W4. Data. Collection, storage and biases. “Tidy” data storage. Data access. APIs. Scraping with Python.
- W5. Programming. Control structures. Loops and conditionals. Combining APIs with loops in Python.
- W6. Break. No classes. No homework. Tip: Complete all your portfolio work and start on project.
- W7. Maps. Cartography history. Projections. Base maps. Putting data on maps.
- W8. Big data. Challenges with scale. A big data cookbook.
- W9. Advanced analytics. Creating visualisations that help investigate relationships.
- W10. Machine Learning. History. Regressions and classification. Clustering and dimensionality reduction.
- W11. Large Language Models. Gathering, cleaning, visualising data. Auditing output.

Stock take: Nearly there!

Our final two sessions

- W1. Your Live Site. HTML, CSS, JavaScript. GitHub and embedding.
- W2. Data debating – chart design. Raw files, editing data. String and date functions.
- W3. The Language of Data. Why visualisation is vital in policy. Manipulating data with Python.
- W4. Data. Collection, storage and biases. “Tidy” data storage. Data access. APIs. Scraping with Python.
- W5. Programming. Control structures. Loops and conditionals. Combining APIs with loops in Python.
- W6. Break. No classes. No homework. Tip: Complete all your portfolio work and start on project.
- W7. Maps. Cartography history. Projections. Base maps. Putting data on maps.
- W8. Big data. Challenges with scale. A big data cookbook.
- W9. Advanced analytics. Creating visualisations that help investigate relationships.
- W10. Machine Learning. History. Regressions and classification. Clustering and dimensionality reduction.
- W11. Large Language Models. Gathering, cleaning, visualising data. Auditing output.

Announcements.

Course news....

Project FAQ

Week 12 – project help

AI event

Course Feedback

Projects: FAQ

Frequently Asked Questions and Answers are on Moodle

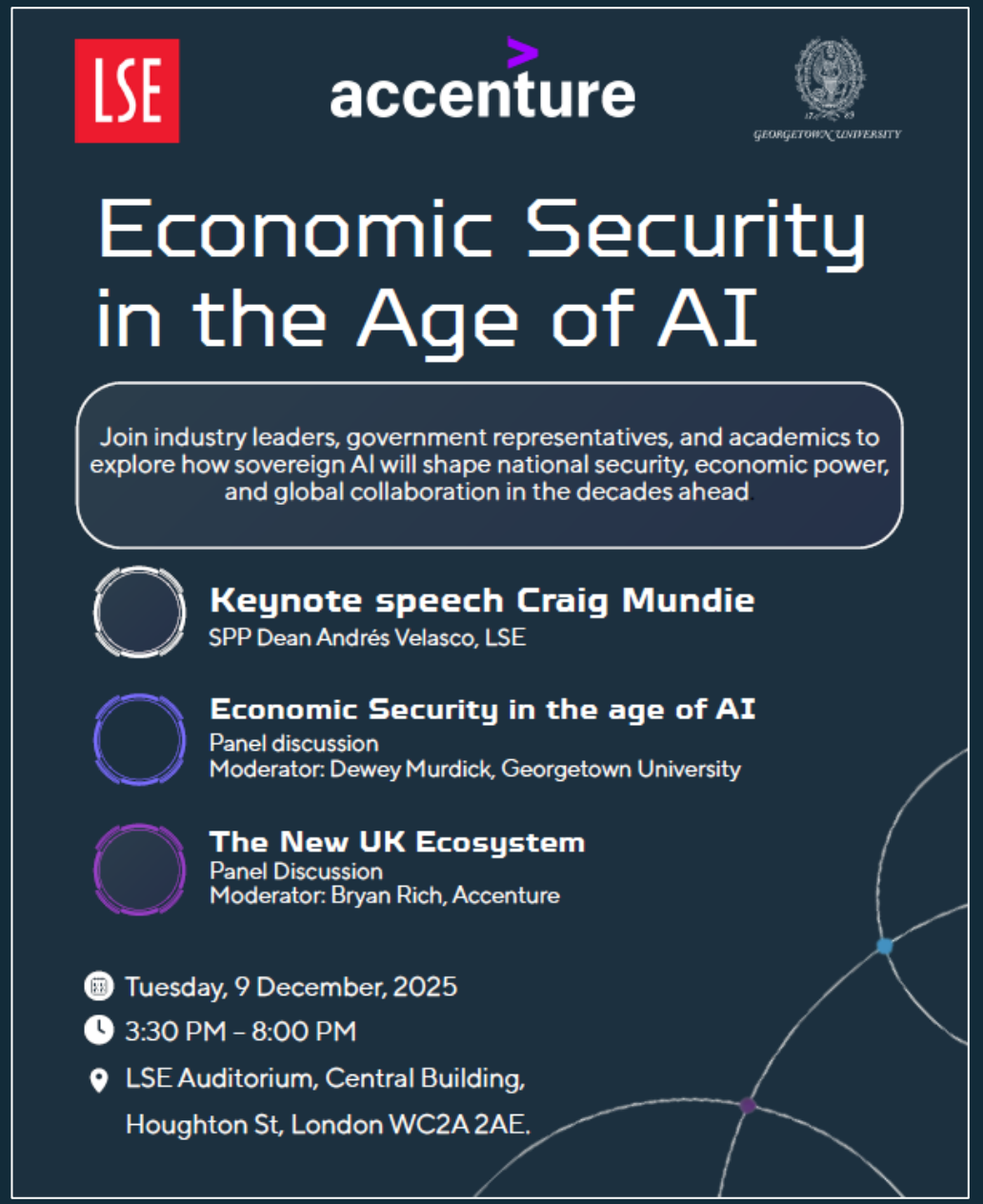
Project help: Week 12

Questions for the TA team and Growth Lab RAs.

AI event. Next Tuesday

A highly relevant event for anyone taking this course

- **To attend in person**, please register here: [Economic Security in the Age of AI – Fill in form]



Survey. Feedback

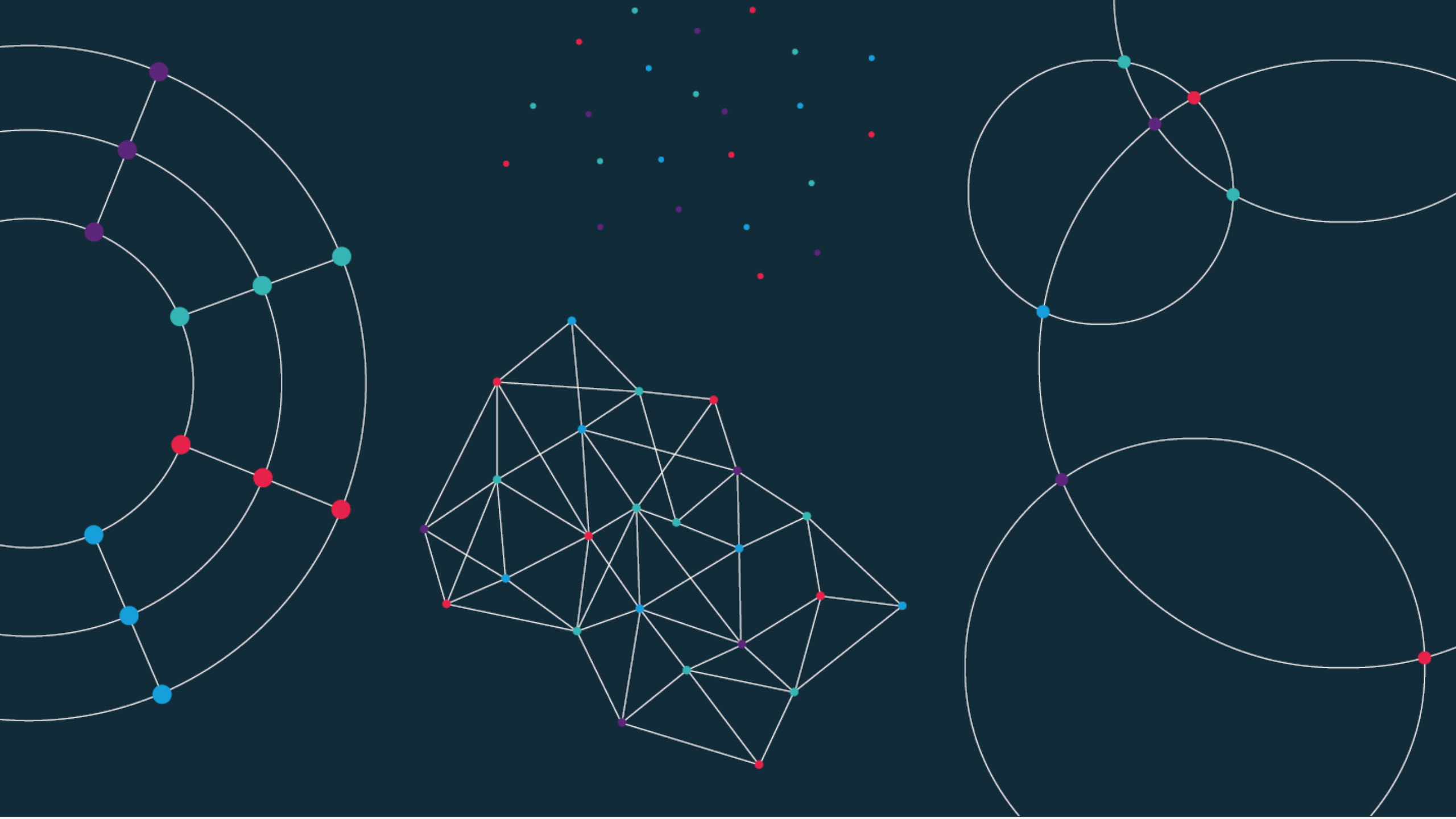
Please let us know what you think of the course

Take part in the LSE Online Survey

Don't miss out on your chance to let us know what you think.
To take the survey now, please scan the QR code:

or go directly to https://lse.surveys.evasysplus.co.uk

LSE TQARO



10. AI-1: Machine learning.

Teaching machines to analyse

Lecture 10.

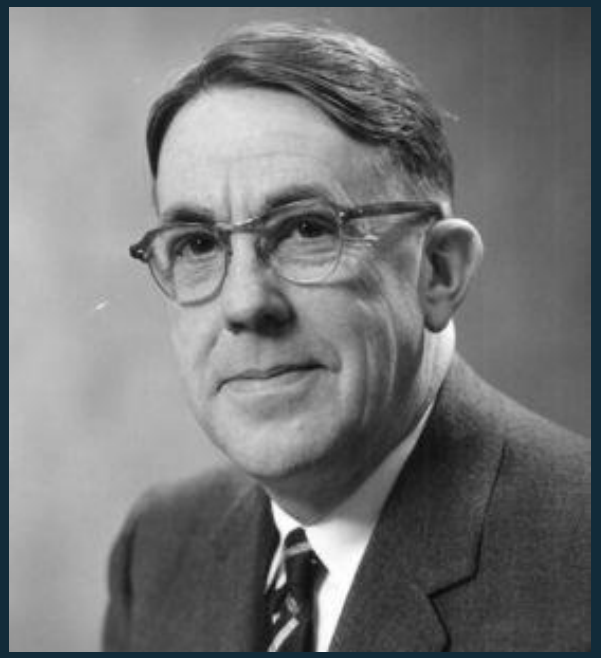
- 10.1 AI intro: history, modern pioneers, terminology, tools.
- 10.2 Supervised. Classification and regression
- 10.3 Unsupervised. Clustering. Collapsing.
- 10.4 Reinforcement.
- 10.5 Practical 10: Four ML examples
- 10.6 ML/AI Glossary

10.1 AI intro.

History, terminology

Arthur Samuel.

American pioneer of machine learning and AI



Arthur Samuel
(1901–1990)

IBM researcher (1949-1966) who demonstrated that computers could learn from experience, not just follow pre-programmed rules

Coined ‘Machine Learning’ (1959): *“Field of study that gives computers the ability to learn without being explicitly programmed”.*

Draughts/checkers program (1952-1962): World’s first self-learning program.

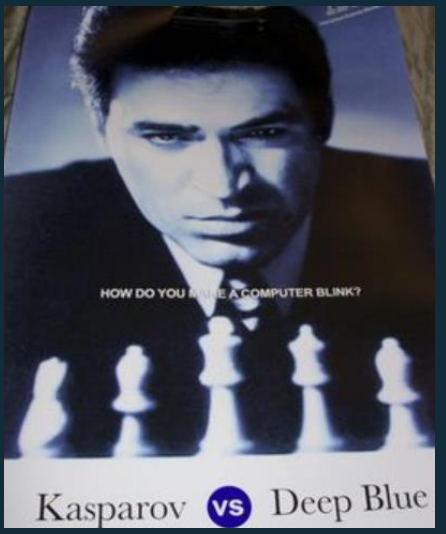
- Learned by playing thousands of games against itself
- In 1962, beat a Connecticut state checkers champion.
- Pioneered techniques: minimax algorithm, alpha-beta pruning, hill-climbing, learned evaluation functions.

Planted the seeds for modern ML (& reinforcement learning). Showed that learning, not just processing power, was key to intelligent behaviour.

A. L. Samuel. "*Some Studies in Machine Learning Using the Game of Checkers*" - IBM Journal of R&D, 1959.

AI: timeline.

From checkers to perceptrons, transistors and DeepSeek_R1.



1950. Alan Turing. The Turing Test.

1952. Arthur Samuel. Automated checkers.

1956. The Dartmouth Workshop. (John McCarthy, 1955).

1957. Perceptron model (Rosenblatt) the Mark 1 Machine (could learn from experience) in 1958.

1966. Automatic Language Processing Advisory Committee (ALPAC 1966).

1973. Lighthill Report (Lighthill, 1973).

1997. Deep Blue (IBM) beats Gary Kasparov. (Film: Game Over: Kasparov and the Machine, 2003).

2012. AlexNet. (Alex Krizhevsky) win the ImageNet challenge. Deep learning, using GPUs. [Source Code]

2016. AlphaGo (Google), beat Lee Sedol, Korean Go champion.

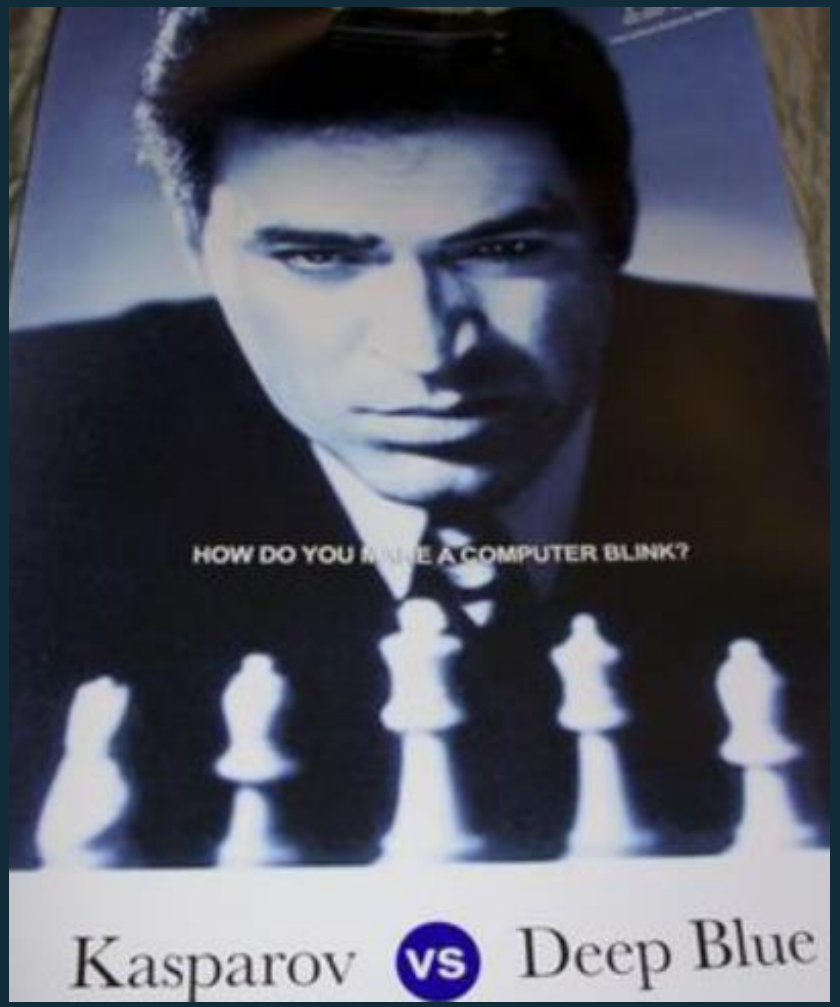
2017. Transformer architecture. Leads to LLMs.

2022. Launch of ChatGPT.

2025. DeepSeek-R1, partly open source reasoning model. (Nature, 2025).

Film.

An interesting documentary

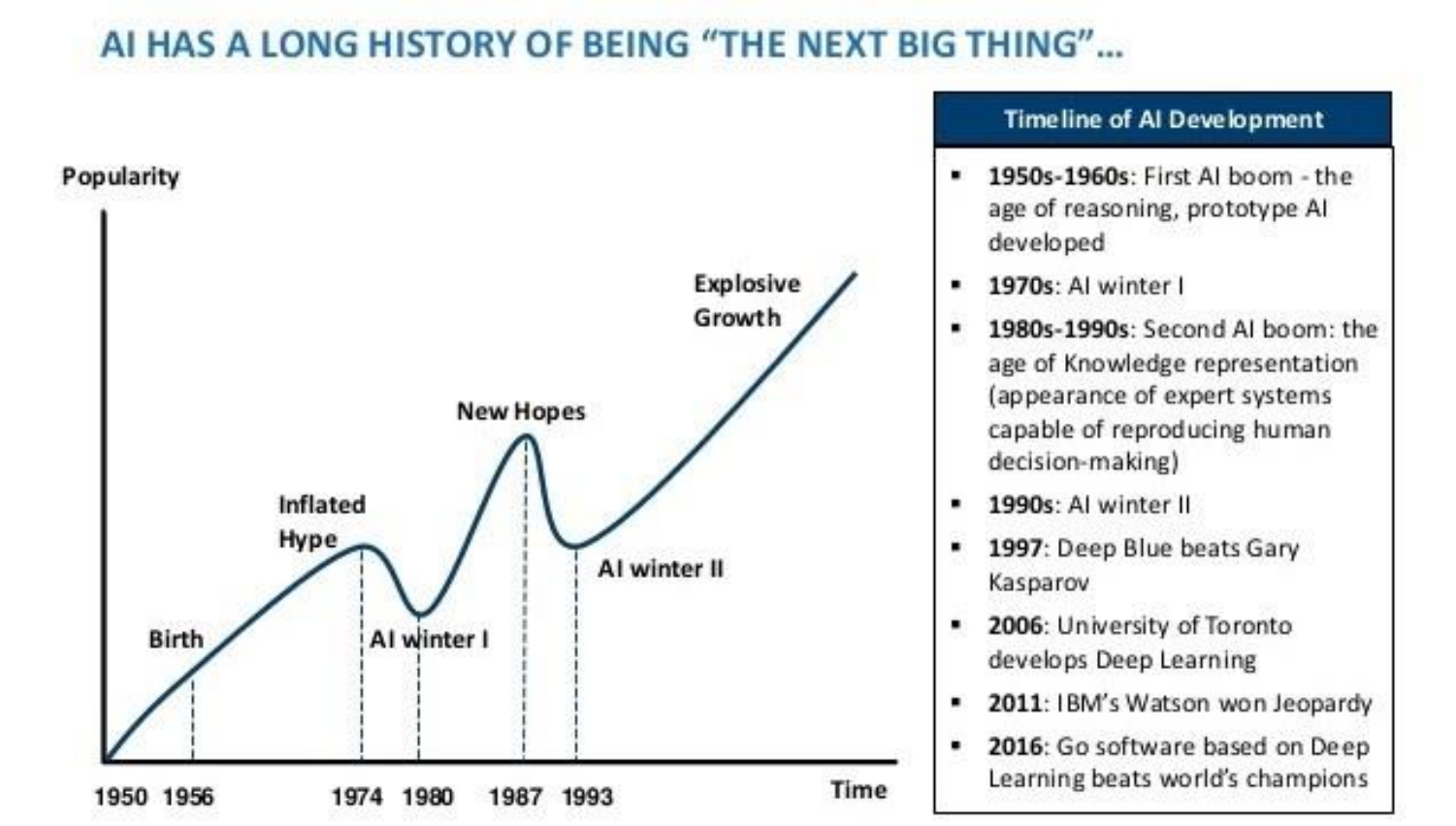


1997. Deep Blue (IBM) beats Gary Kasparov. (Film: Game Over: Kasparov and the Machine, 2003).

The AI rollercoaster.

Booms and winters

Given the recent excitement associated with AI, it is worth nothing that, historically, there have been various booms and “winters”.



Lim (2018)

AI winters.

Detail on two AI winters

1973-73

- In the UK, Artificial Intelligence: A General Survey (Lighthill, 1973) ended AI research for a decade. BBC debate. “in no part of the field have the discoveries made so far produced the major impact that was then promised”.
- In the US, DARPA shifted funding towards specific ‘missions’. AI projects (too general) were hard to fund. DARPA history.

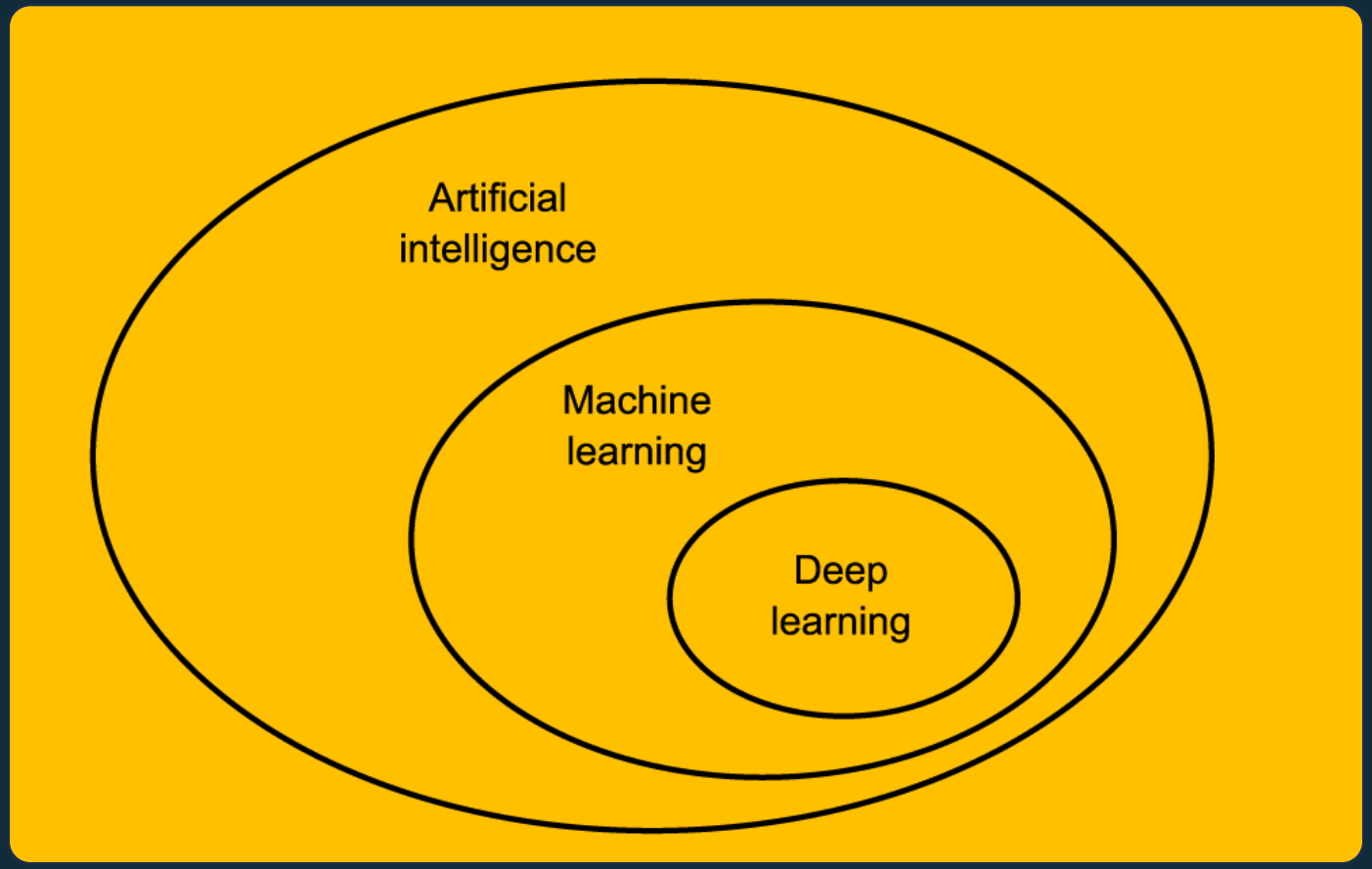
1980s

- Funding was now often commercial. Development moved to “expert systems”. The idea: feed in knowledge from humans, and have AI replicate answers.
- By the mid 1980s because clear that systems gave wrong (and dangerous) answers, with no knowledge of their limitations.

Schumann (2019 and 2019)

AI, ML and DL.

How they relate to one another



François Chollet (creator of Keras)
https://www.manning.com/books/deep-learning-with-python-second-edition

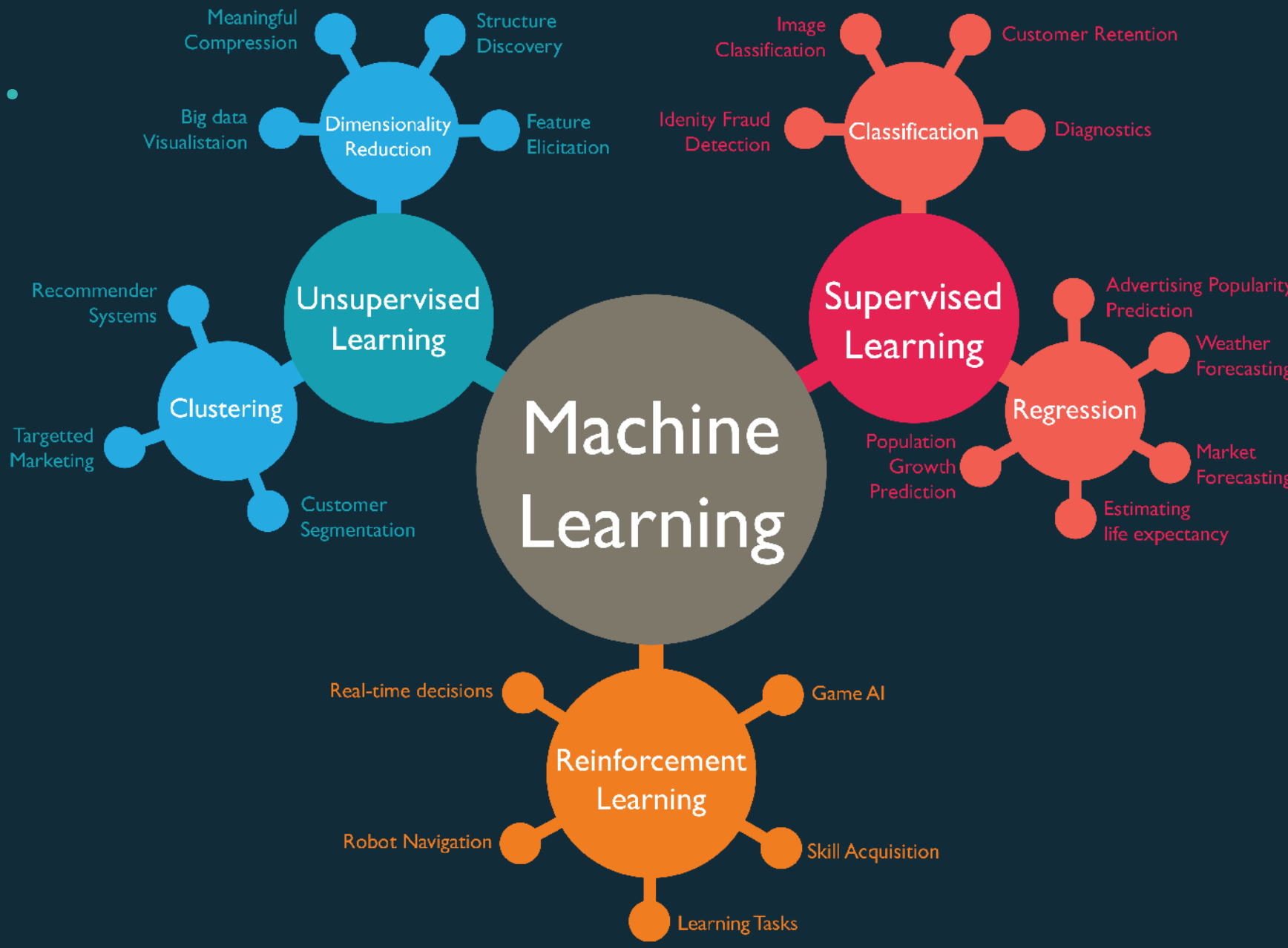
Plan:

This lecture: ML
- Supervised
- Unsupervised
- Reinforcement

Next lecture:
- Deep learning
- LLMs

Types of ML.

Three paradigms



Programming vs learning.

A paradigm shift

Classical programming.

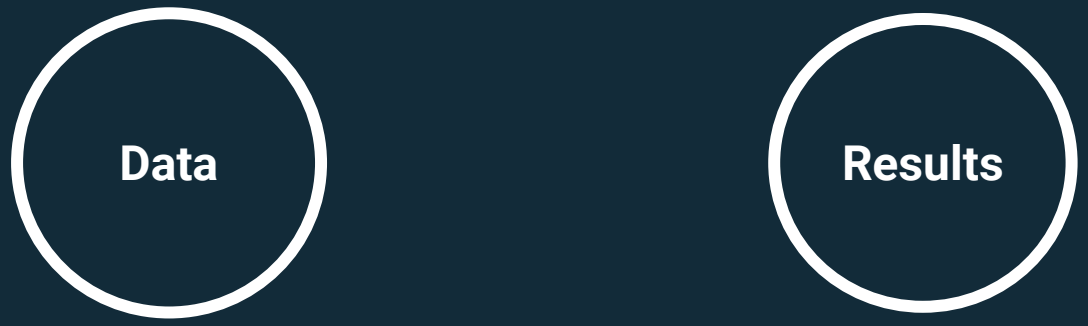
Data and rules give us results

Classical programming.

Data and rules give us results

Machine Learning.

Use data and results to uncover the rules



Machine Learning.

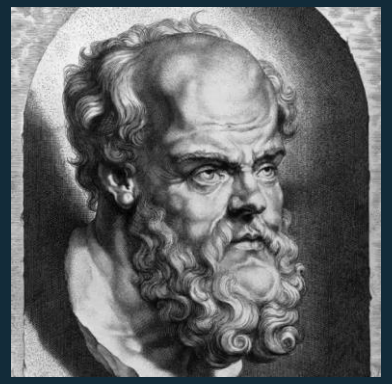
Use data and results to uncover the rules

Programming vs ML.

Paradigm shift

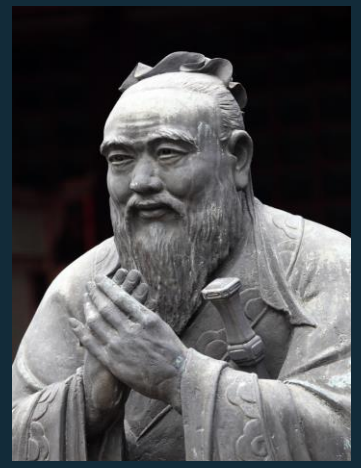
Deep thinking.

Name the thinker

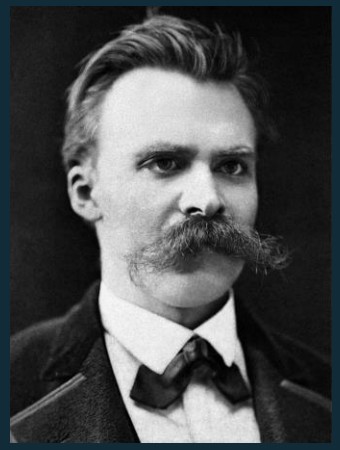


1

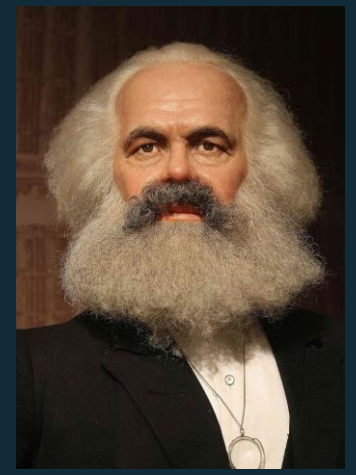
2



3



4



5

6

7

8

9

Brains vs Computers.

They aren't the same…

François Chollet (creator of Keras) https://keras.io/

"Deep learning models are not models of the brain.

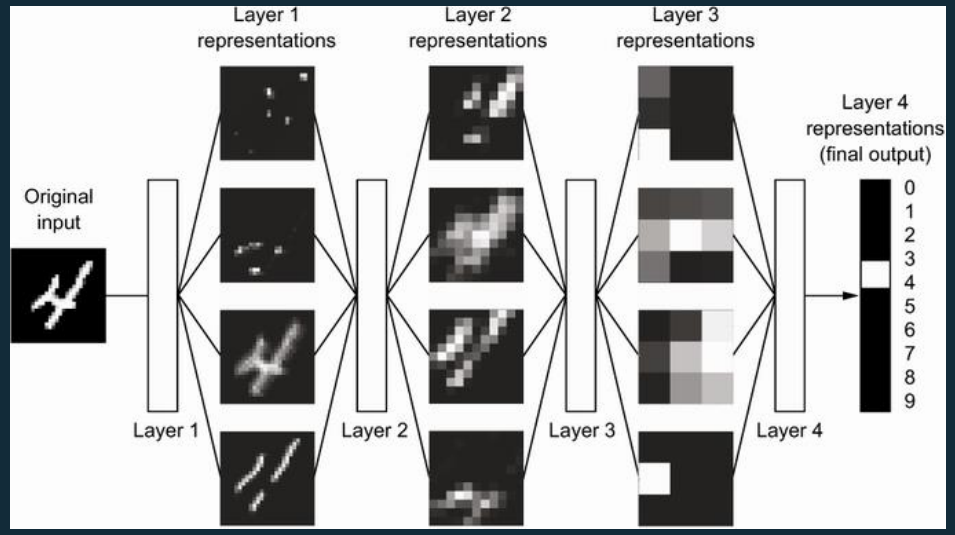
There's no evidence that the brain implements anything like the learning mechanisms used in modern deep learning models.

It would be confusing and counterproductive for newcomers to the field to think of deep learning as being in any way related to neurobiology; you may as well forget anything you may have read about hypothetical links between deep learning and biology.

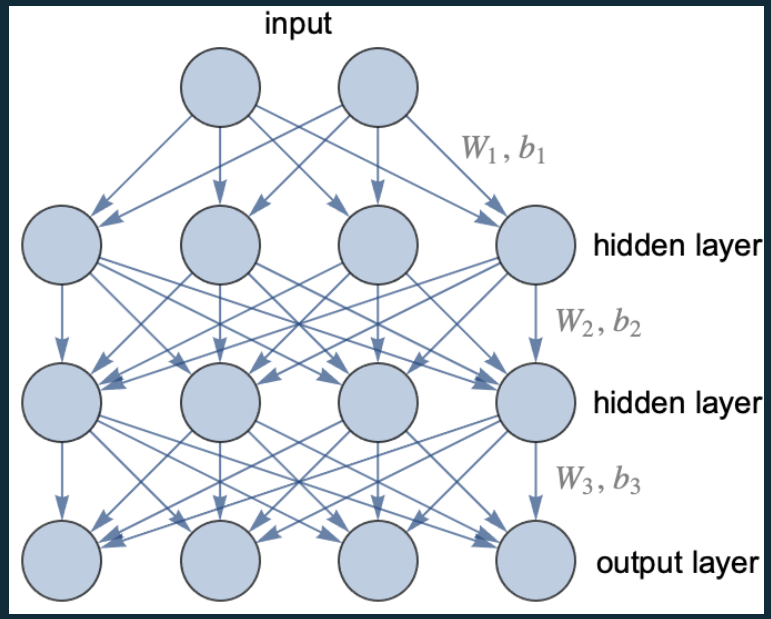
For our purposes, deep learning is a mathematical framework for learning representations from data."

Deep learning.

Is layered, or stacked, learning



https://livebook.manning.com/book/deep-learning-with-python-second-edition/chapter-1#55



multilayer perceptron
fully connected network
feed-forward neural network

Etienne Bernard

Introduction to Machine Learning

https://www.wolfram.com/language/introduction-machine-learning/

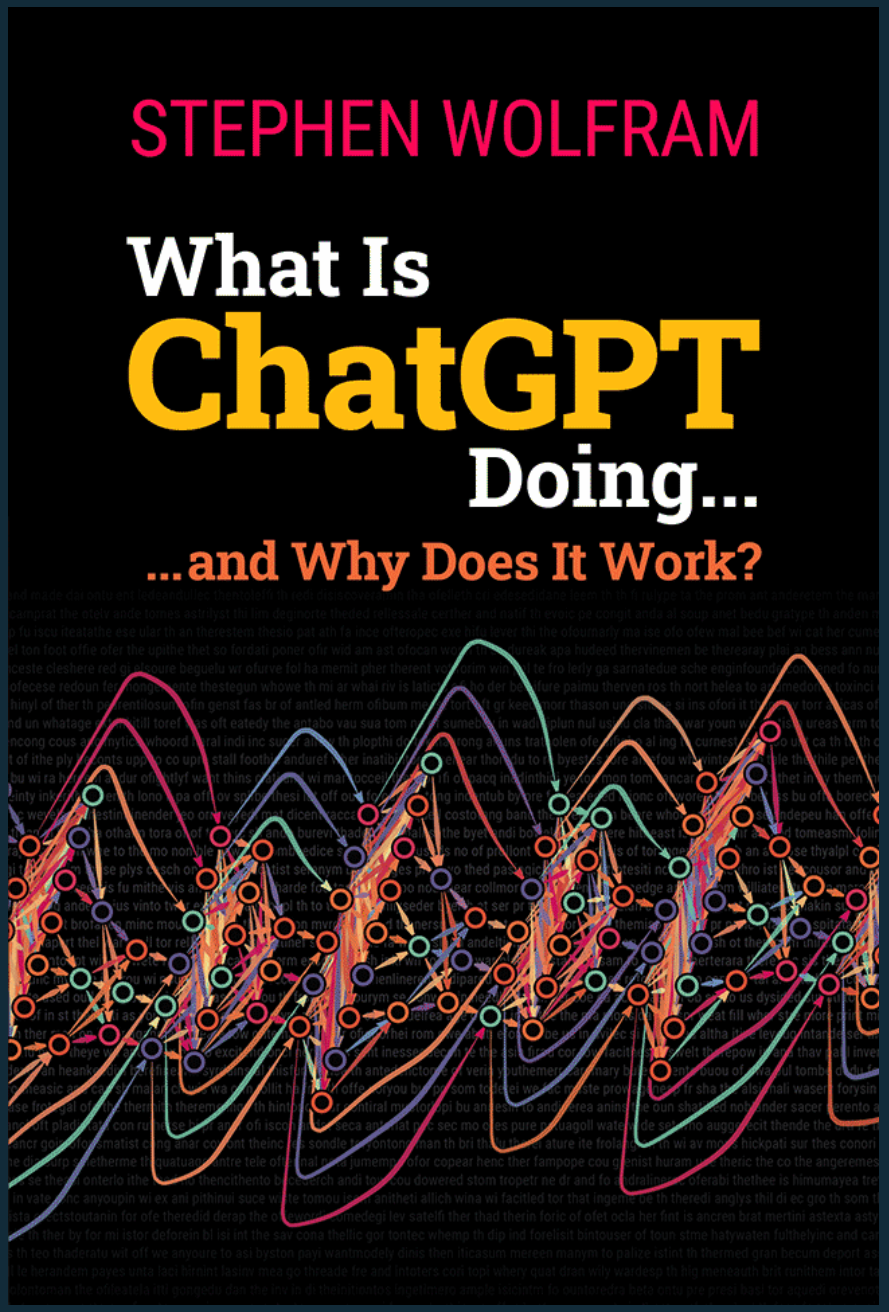
ML books.

Helpful open-source books

Stephen Wolfram
https://writings.stephenwolfram.com/2023/02/what-is-chatgpt-doing-and-why-does-it-work/

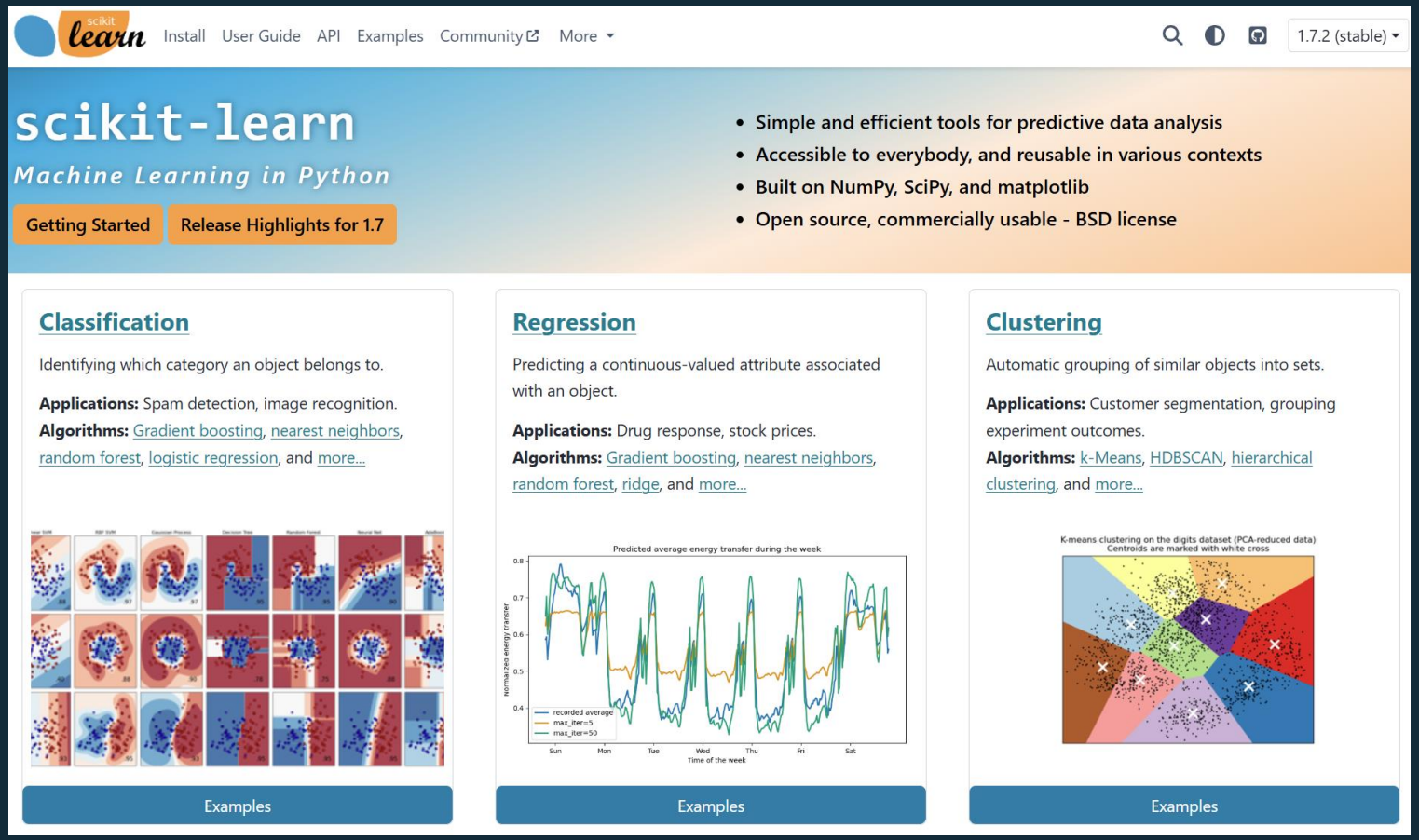
François Chollet
https://livebook.manning.com/book/deep-learning-with-python-second-edition/chapter-1

Ettienne Bernard
https://www.wolfram.com/language/introduction-machine-learning/



Tools: scikit-learn.

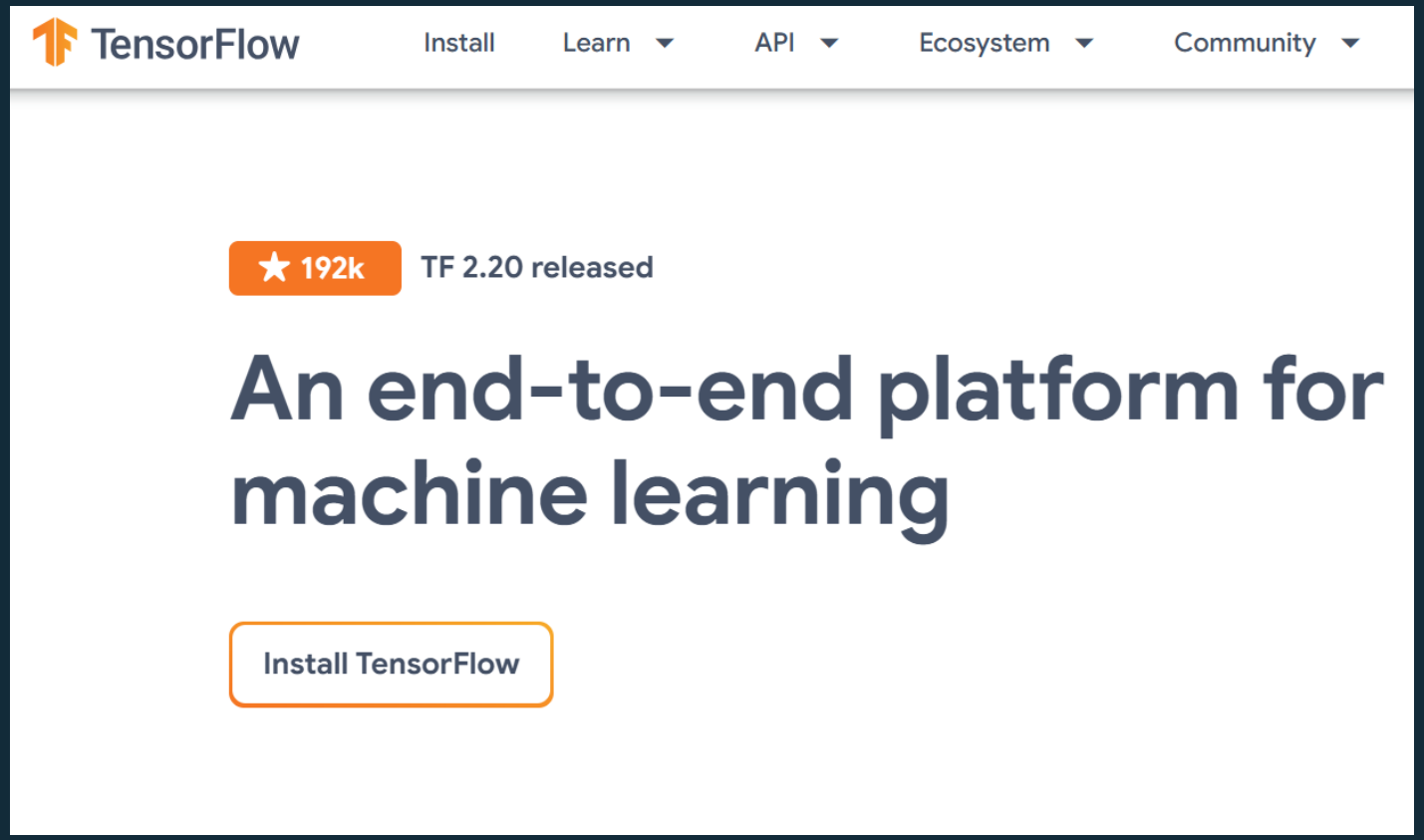
Machine learning in Python/Collab



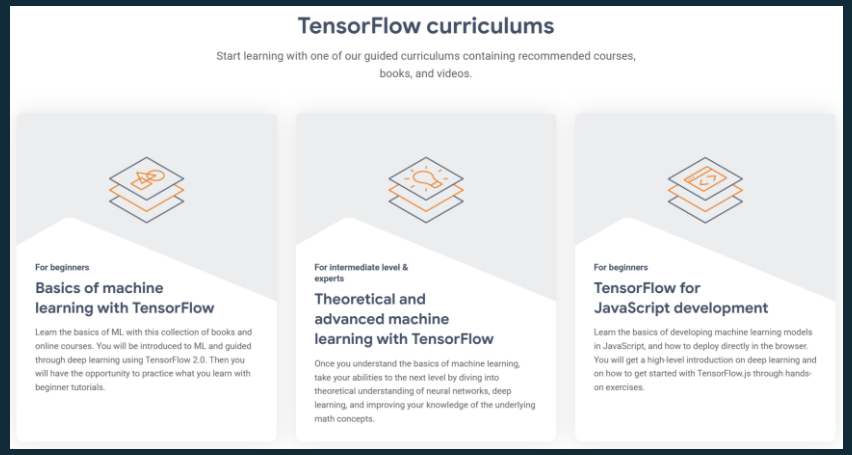
https://scikit-learn.org/stable/

Tools: TensorFlow.

Deep learning with Python



https://www.tensorflow.org/



https://www.tensorflow.org/resources/learn-ml

10.2 Supervised learning.

Data with labels

Supervised learning.

Intuition and examples

In supervised learning, the researcher oversees the algorithm.
In practical terms, this means feeding in lots of correct answers.

Examples include:

- Regression
- Classification

To do this, we need the dataset of correct answers.

This is referred to as labelled data.

Example: retail sales.

Structured data

Store	Date	Weather	Sales
London	2025-11-27	Muggy	£3,000
Manchester	2025-11-27	Cold	£4,000
Swansea	2025-11-27	Sunny	£5,000
London	2025-11-29	Cold	£4,000
Manchester	2025-11-29	Wet	£7,000
Swansea	2025-11-29	Sunny	£8,000

X

[Features]

y

[labels]

Example: retail sales.

Structured data

Store	Date	Weather
London	2025-11-27	Muggy
Manchester	2025-11-27	Cold
Swansea	2025-11-27	Sunny
London	2025-11-29	Cold
Manchester	2025-11-29	Wet
Swansea	2025-11-29	Sunny

Sales
£3,000
£4,000
£5,000
£4,000
£7,000
£8,000

Example: retail sales.

Structured data

Store	Date	Weather
London	2025-11-27	Muggy
Manchester	2025-11-27	Cold
Swansea	2025-11-27	Sunny
London	2025-11-29	Cold
Manchester	2025-11-29	Wet
Swansea	2025-11-29	Sunny
London	2025-12-04	Cold
Manchester	2025-12-04	Freezing
Swansea	2025-12-04	Sunny

Learning phase / Training phase

Evaluation phase / Prediction phase

Sales
£3,000
£4,000
£5,000
£4,000
£7,000
£8,000
?
?
?

How good is your data?

Good data is not just about accurate labels

Store	Date	Weather
London	2025-11-27	Muggy
Manchester	2025-11-27	Cold
Swansea	2025-11-27	Sunny
London	2025-11-29	Cold
Manchester	2025-11-29	Wet
Swansea	2025-11-29	Sunny
London	2025-12-04	Cold
Manchester	2025-12-04	Freezing
Swansea	2025-12-04	Sunny

How could we improve on this?

Discuss

In ML (as in econometrics) your analysis will be influenced by the data, and format of the data, that you feed in.

Think about this for your projects

Re-cap: Labelled data.

Labelled data = Input features + known outcomes

Training

Apples

Not apples

Not apples

Apples/
Not Apples

Each data point has both features (X) and a label (y)

Prediction/inference

Apples

We are learning an unknown function that approximates this relationship.

Fake notes

Can we train a machine to spot forgeries and fakes?

For centuries, forgers and mints have been locked in a cat and mouse game where advances in currency design are met by increases in forgery sophistication.

Accurate forgery of paper notes

Banknote forgers guilty, BBC News 1999

A new 'Series E' paper variant (2001) and eventually polymer notes (2016-2024)

3% counterfeit rate of £1 coins

2004 £1 One Pound Coin contemporary forgery

Twelve sided £1 coins (2017)

Nazi production of "near-perfect" notes

A Nazi "supernote" forgery. BoE museum, 1942-1945

Replacement of >£10 notes (1943 onwards)

Example: banknotes.

Can we train a machine to spot forgeries and fakes?

Between 2017 and 2021, Britain introduced new note and coin designs; prior to then, 3% of £1 coins were counterfeits and 719,000 counterfeit notes were withdrawn each year.

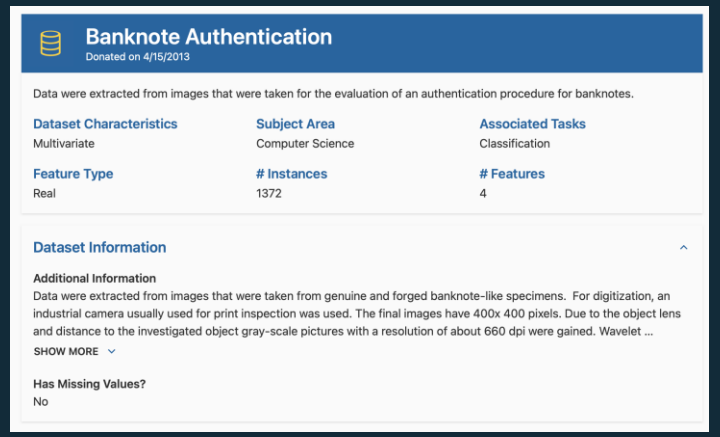
Monetary authorities have a new tool to identify fakes. AI models can quickly detect counterfeits with high accuracy.

This year, the US Secret Services is investing $1.9M in “AI-enabled counterfeit currency detection”.

Dataset
Indian and Thai bank notes

Meshram et al. (2021)
Field-quality images

Denominations, not fitness



Dataset
Real and forged notes

Lohweg (2013)
Numeric features from real forged and authentic notes.

ML for bank note verification

A modern machine learning pipeline

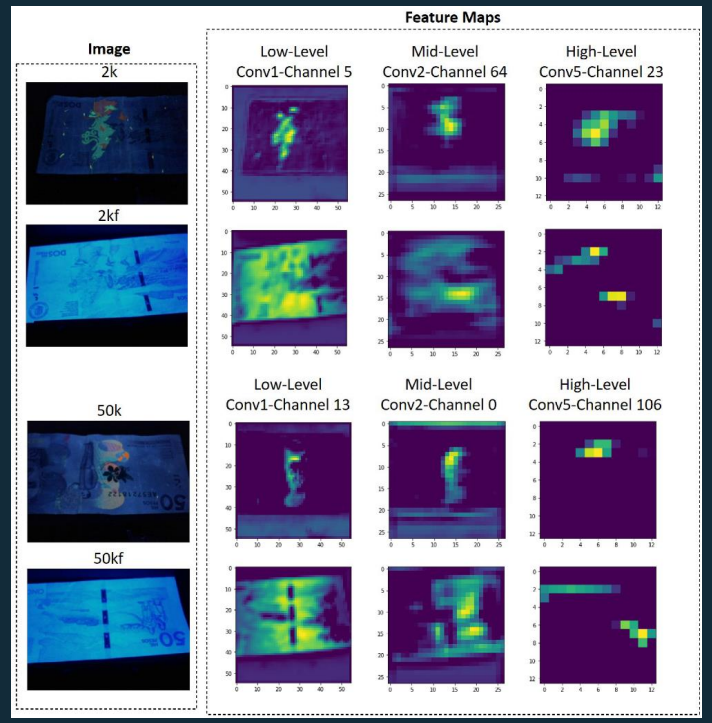
ML models can detect subtle micro-texture anomalies, printing inconsistencies or incorrect UV patterns.

Capture
RGB(+UV/IR, optionally)

Preprocess
Crop, align, normalise

ML Layer

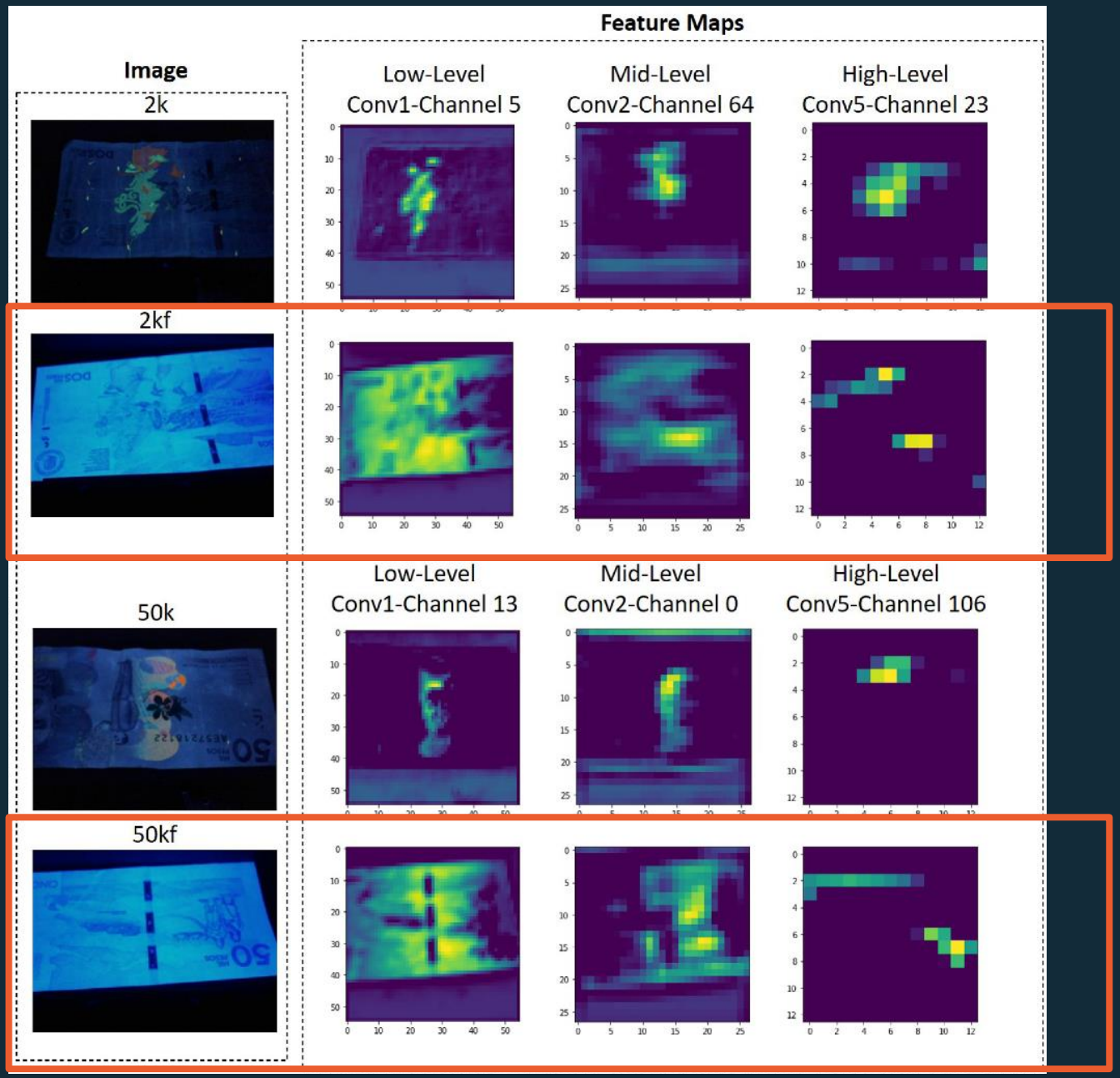
Decision layer
Risk scores and thresholds



The CNNs learn the *features* of banknotes, including edges and textures in early layer to security feature patterns in deeper layers.

Although individual CNN filters are not interpretable, systematic differences can be seen.

Fake Banknote Recognition Using Deep Learning, Pachon, Ballesteros, and Renza (2021)



Forgery

Forgery

Fake Banknote Recognition Using Deep Learning, Pachon, Ballesteros, and Renza (2021)

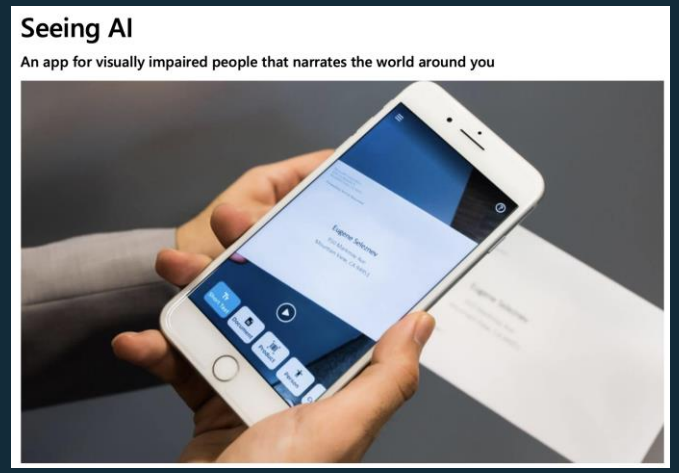
Accessibility

What else can ML do with currency images?

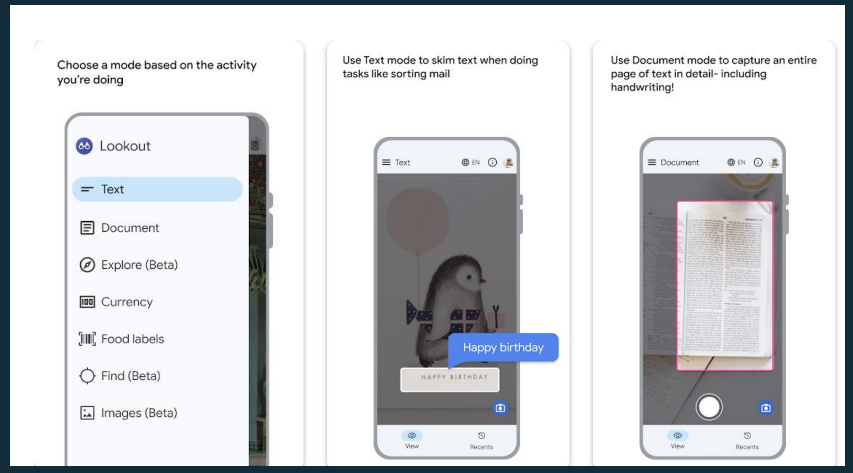
Currency recognition is a daily challenge for blind and partially-sighted people.
Some countries include tactile marks or varied note sizes, but others don't; US notes for example, or all the same size and have similar designs.

ML-enabled apps help solve this problem. With apps like Lookout or Seeing AI, users can point their phones at notes and documents for identification.

The tools and implementations differ but the ML foundations are the same.



Microsoft Seeing AI



Google Lookout

Supervised toolkits.

Approaches for regression and classification

Supervised Learning

Prediction

- Linear regression (OLS, Ridge, Lasso)
- Decision tree regression
- Random forest regression
- Support regressions

Classification

- Logistic regression
- Support vector machines
- Decision trees
- K nearest neighbours

Regression.

Each model has many implementation methods

Regression is a supervised learning for predicting continuous values (or probabilities of binary outcomes).

Multiple variants:

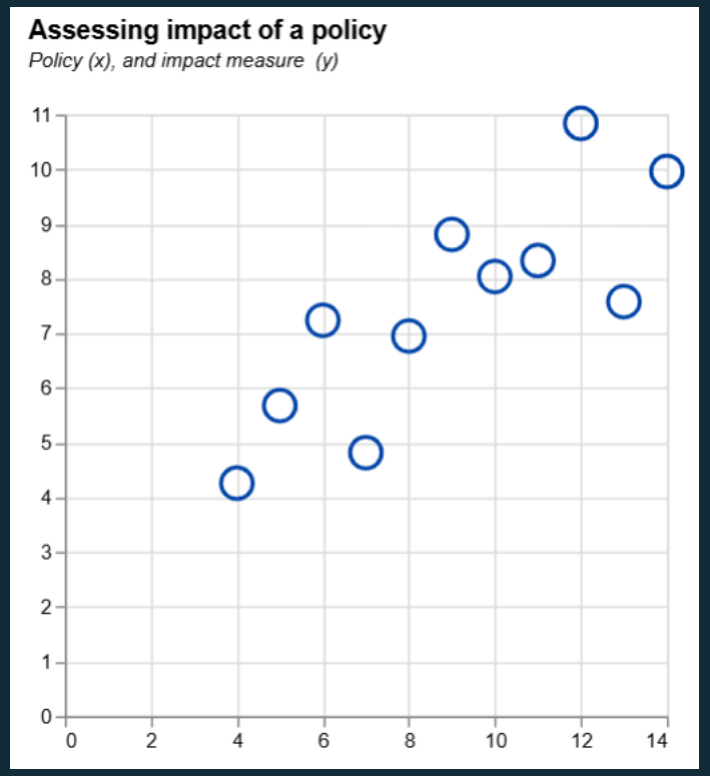
- Ordinary Least Squares (OLS) - Minimises squared errors, assumes normal distribution
- Ridge Regression - Adds L2 penalty to prevent overfitting, useful with multicollinearity
- Lasso Regression - Adds L1 penalty, can perform feature selection
- Elastic Net - Combines Ridge and Lasso penalties

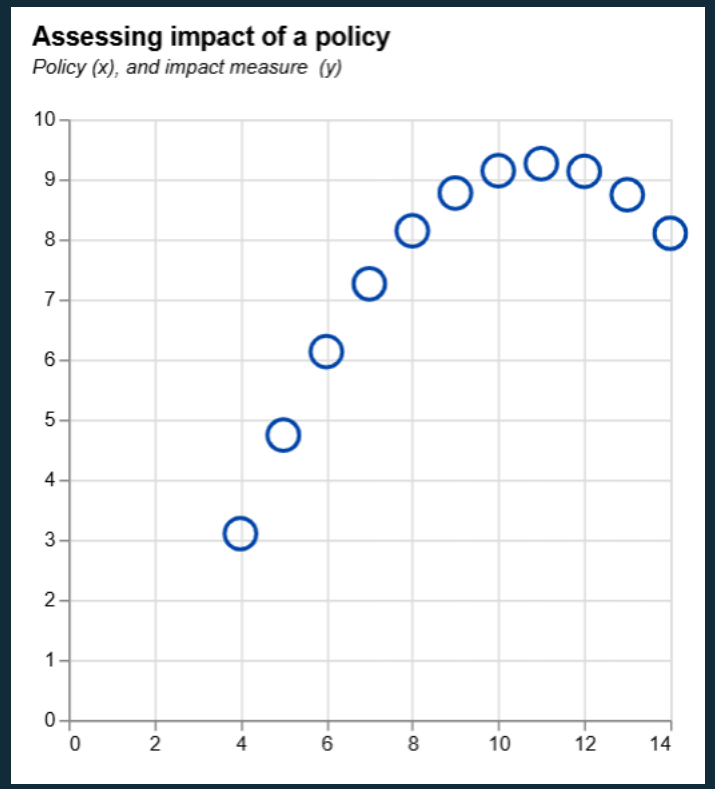
Find available linear models in Scikit-learn docs here

The right fit.

Under and over-fitting

Recall two of the Anscombe quartet datasets:





Using our terminology:

X = features

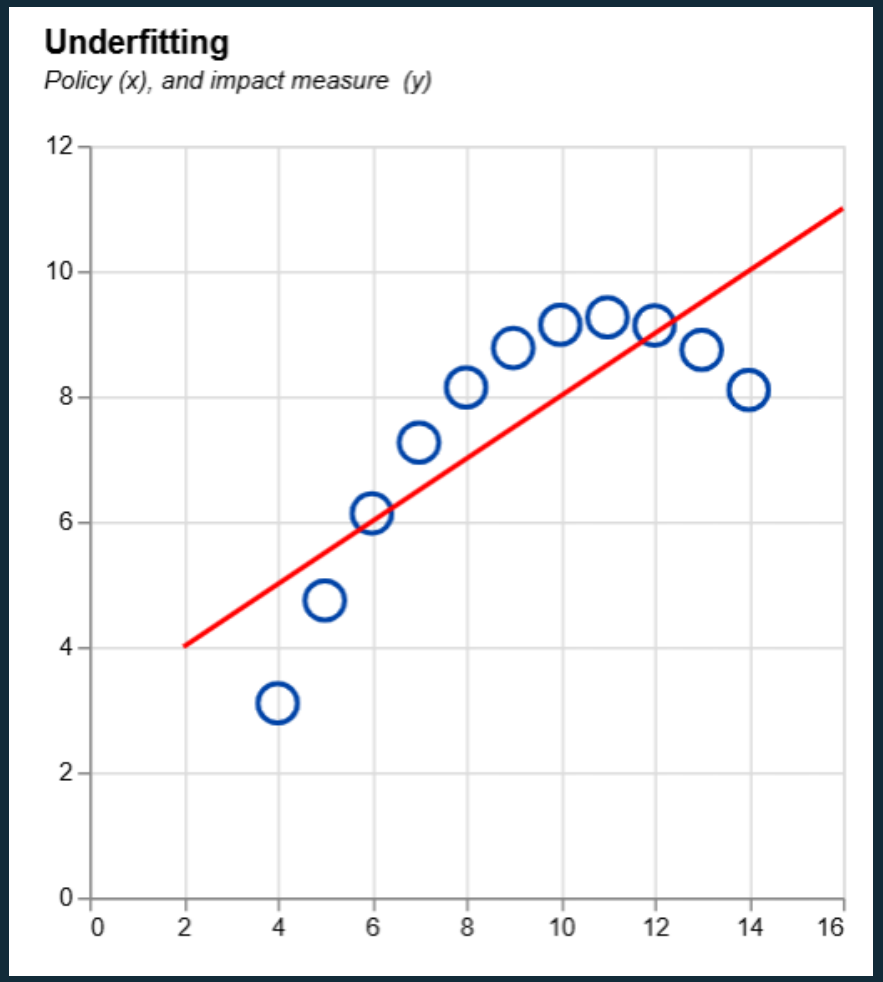
y = labels

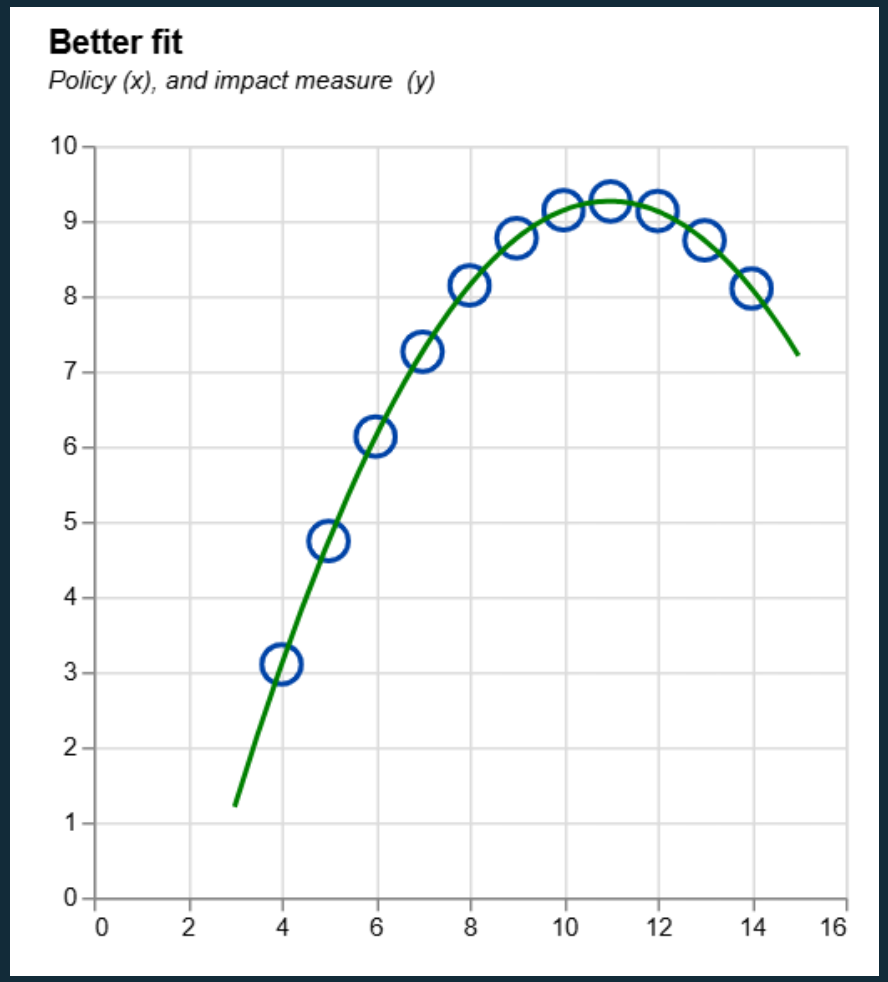
X is some policy [investment]

y is some outcome [GDP per capita]

Underfitting.

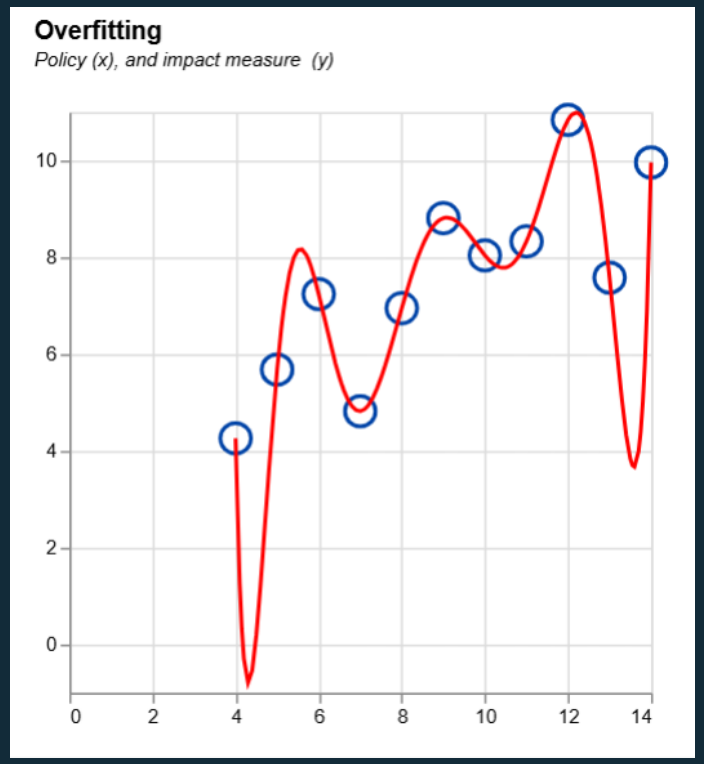
A line might constrain your model too much





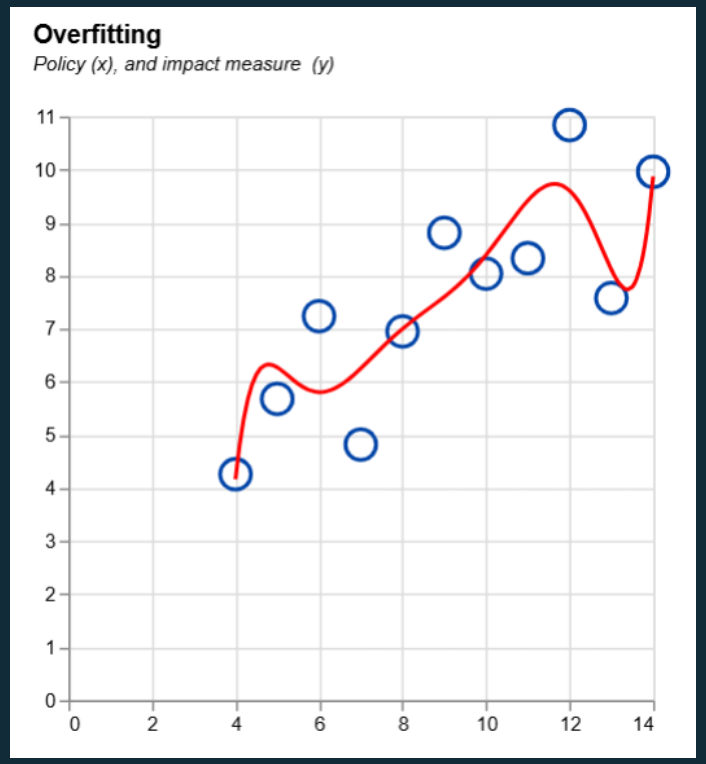
Overfitting.

A polynomial might make your model too complex to predict anything

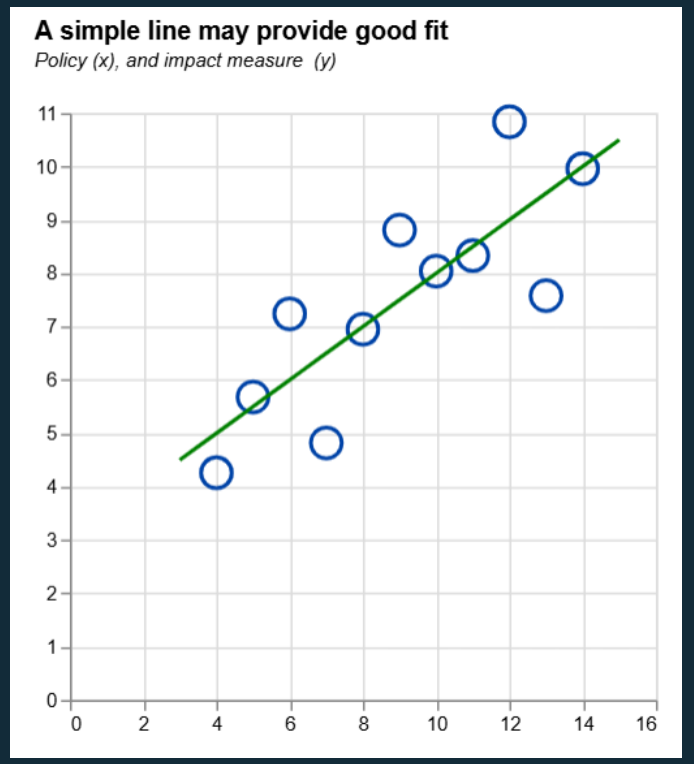


Order = 10

*[y = a + b * x + c * x^2 + ... + k * x^10]*



Order = 7



Order = 1

*[y = a + b * x]*

K-Nearest Neighbours.

Simple classification model with KNN

How it works: Classifies new points based on the majority vote of their k nearest neighbours

- Find the k closest points in your training data
- Assign the most common class among those neighbours
- "Tell me who your neighbours are, and I'll tell you who you are”

Example: Predicting US region from socioeconomic indicators

- Features: Median income ($) and firearm death rate (per 100,000)
- k=3: Look at 3 nearest states to classify Northeast vs South

→ Distance matters: Uses Euclidean distance $\sqrt{[(x_1-x_2)^2 + (y_1-y_2)^2]}$

KNeighborsClassifier Scikit-learn docs here

KNN: Scaling problem.

Introduction to scaling / normalising in ML

In KNN, classify on nearest points by Euclidean distance
$\sqrt{[(x_1-x_2)^2 + (y_1-y_2)^2]}$

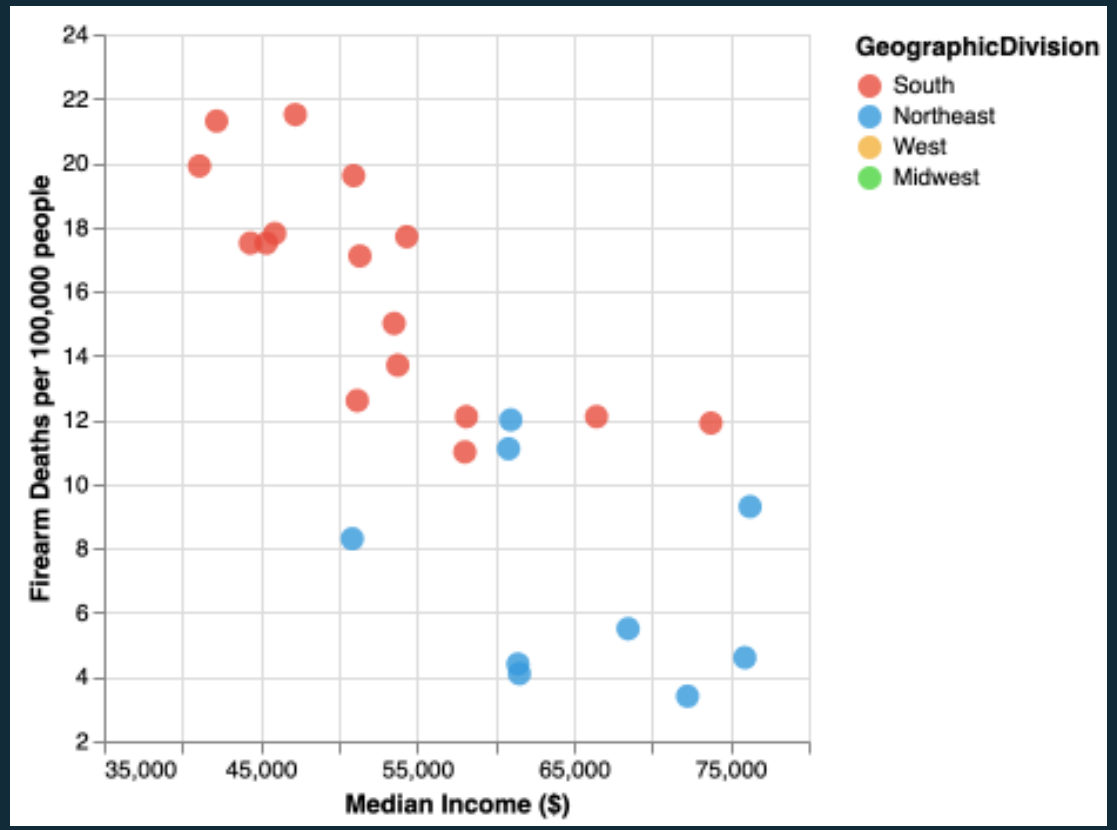
Problem: radically different scales break KNN

Example:

- Income: $40,000 - $90,000 (range ~50,000)
- Death rate: 3 - 25 per 100,000 (range ~20)
- Income differences dominate: $5{,}000^2 = 25{,}000{,}000$ vs $5^2 = 25$

Without scaling, our decision boundary is essentially a vertical line - only income matters.

Solution: Feature Scaling

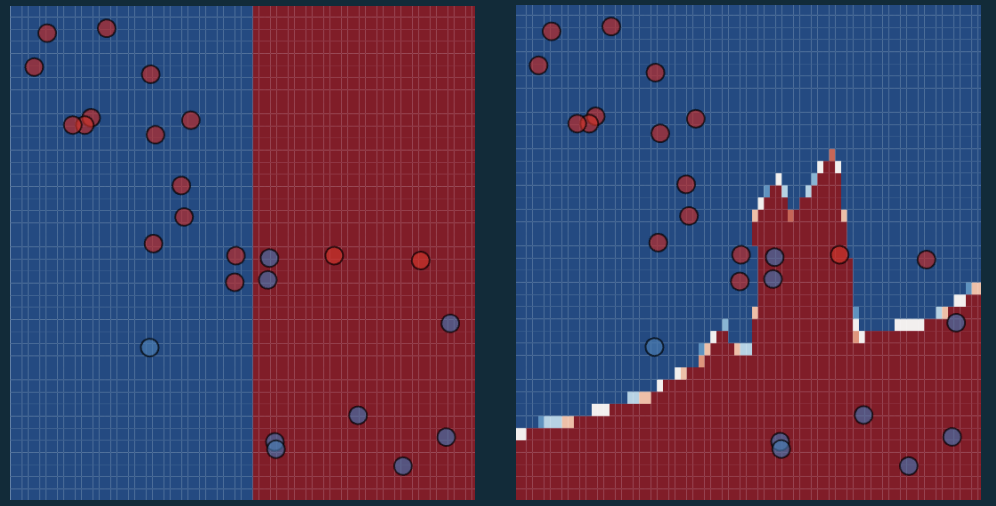


Feature scaling.

Standardisation is a common requirement for ML methods

StandardScaler: transform all features (variables) to mean=0, standard deviation=1.

Now each feature contributes equally to distance.



KNN US states example (k=3): before and after scaling firearms and income data.

Alternative scaling methods:

- MinMaxScaler: Scale features to [0,1] range
- RobustScaler: Uses median/IQR, robust to outliers
- Normaliser: Scale each sample to unit length

Always scale for distance-based algorithms (KNN, K-means, SVM), neural networks, gradient descent.

Not needed for tree-based models (Random Forest, XGBoost)

See a comparison of how different scaling methods affect data distribution here.

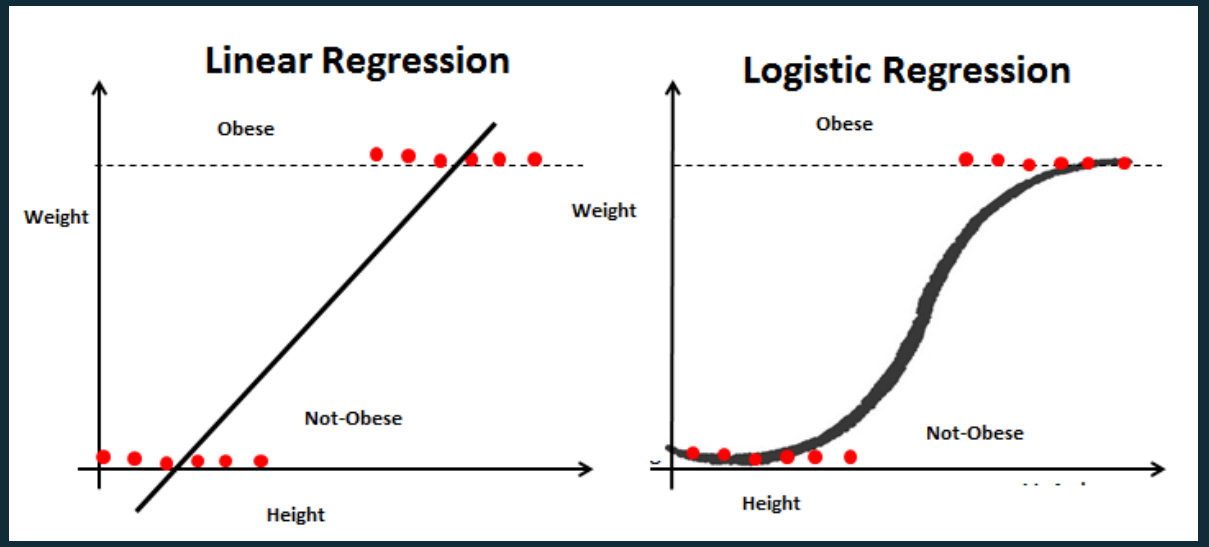
Scikit-learn docs: StandardScaler

Logistic regression.

Classification through probability

From lines to probabilities:

- Extends linear regression for classification tasks
- Uses logistic (sigmoid) function to map outputs to [0,1]
- Predicts probability of class membership, not values
- Decision boundary: where P(class) = 0.5



analyticsvidhya

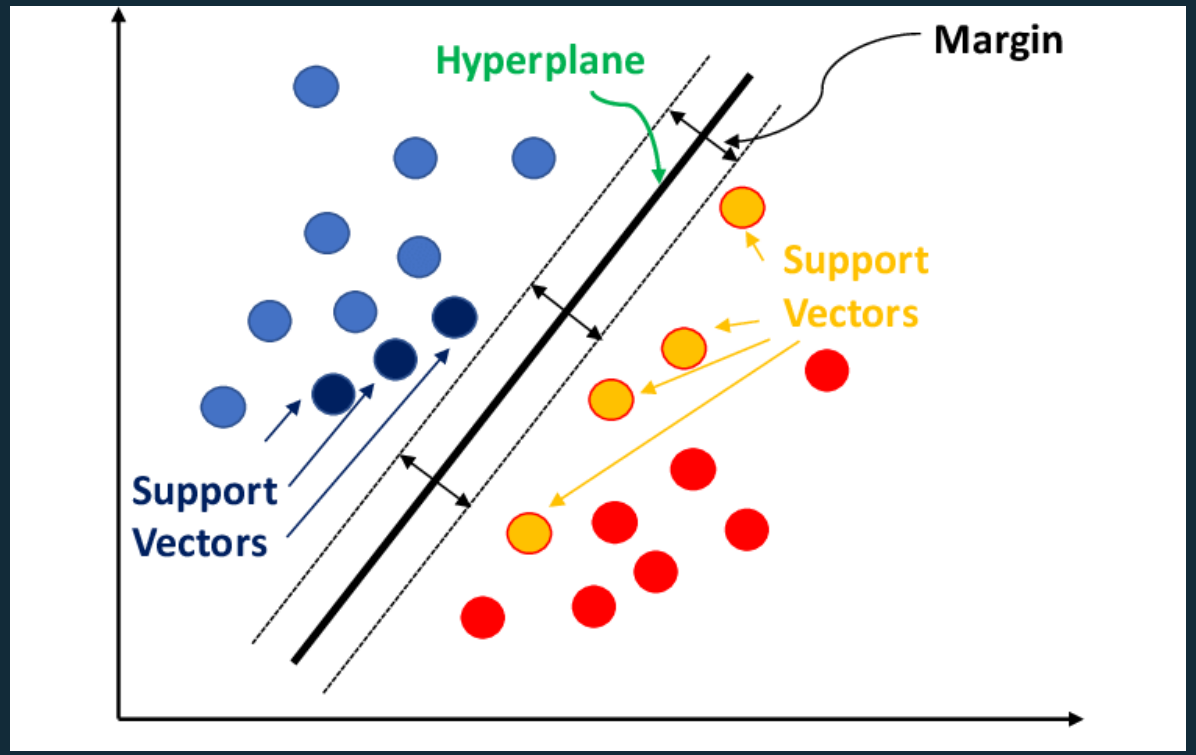
Key differences from linear regression:

- Linear: Predicts continuous values (income, temperature)
- Logistic: Predicts class probabilities (will vote/won't vote)
- Output: S-shaped curve instead of straight line

Support vector machines (SVM).

Identifies the best hyperplane to distinguish the data

- SVM works by finding the best way to separate two labelled groups of data.
- Best separator is a line (2D), plane (3D), or hyperplane (higher dimensions)
- "Support vectors" are the data points closest to the hyperplane.
- The SVM algorithm will choose the line that minimises the margin (perpendicular distance) between support vectors.

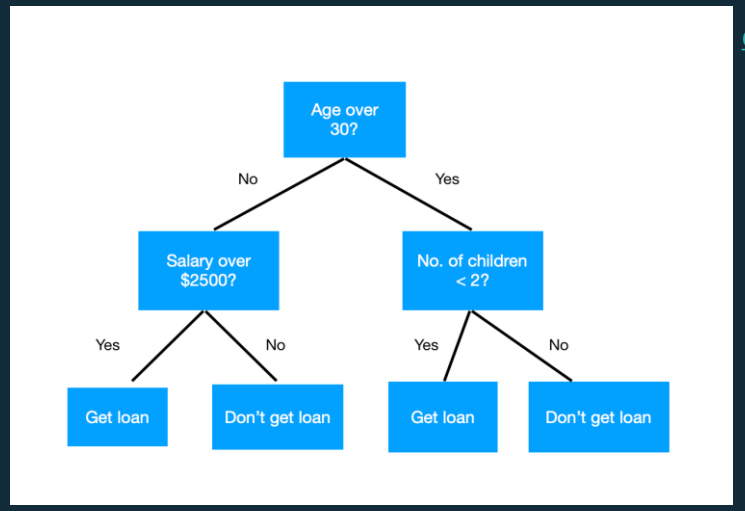


source

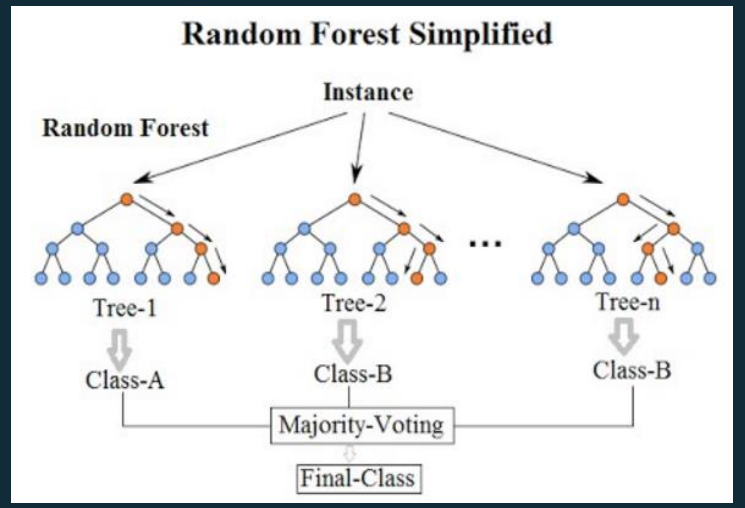
Decision trees, random forests.

Creates decision points to arrive at a predicted value

- Decision trees find the best parameters and split point to separate data into groups at each step.
- Continues splitting each node until data is perfectly classified.
- Can be pruned by removing branches to avoid overfitting.
- Random forests create many trees, each using a random subset of features.
- Averages predictions across all trees to reduce overfitting and improve accuracy.



eloquentarduino.github.io



liorsinai.github.io

House prices.

Discussion of approach

Example: house prices.

Some data to investigate house prices

Business and policy questions:

How much will a customer's house sell for?

How much is a house worth, and how much council tax should someone pay?

To answer these questions, reliable models of house prices are needed.

Basic idea:

- Take a dataset of recent sales (labelled data)
- Apply this to the houses that are for sale, or those you would like to levy a tax on

NEXT STEP:

- **Plug your data into the skikit-learn ML tools in Python!**
- **Everyone happy?**

Example: house prices.

Some data to investigate house prices

Business and policy questions:

How much will a customer's house sell for?

How much is a house worth, and how much council tax should someone pay?

To answer these questions, reliable models of house prices are needed.

Basic idea:

- Take a dataset of recent sales (labelled data)
- Apply this to the houses that are for sale, or those you would like to levy a tax on

BUT:

- **Never jump straight to the "black box" of tools (ML, regression, PCA).**
- Remember "Anscombe's Quartet" and the data dinosaur.
- Example: some UK house price data I prepared for you this week…

Example: UK house prices.

Some data to investigate UK house prices

Data source: Zoopla: https://www.zoopla.co.uk/
Dataset is in course DropBox – original and cleaned data.
It is also in my GitHub repo [Link to Data]

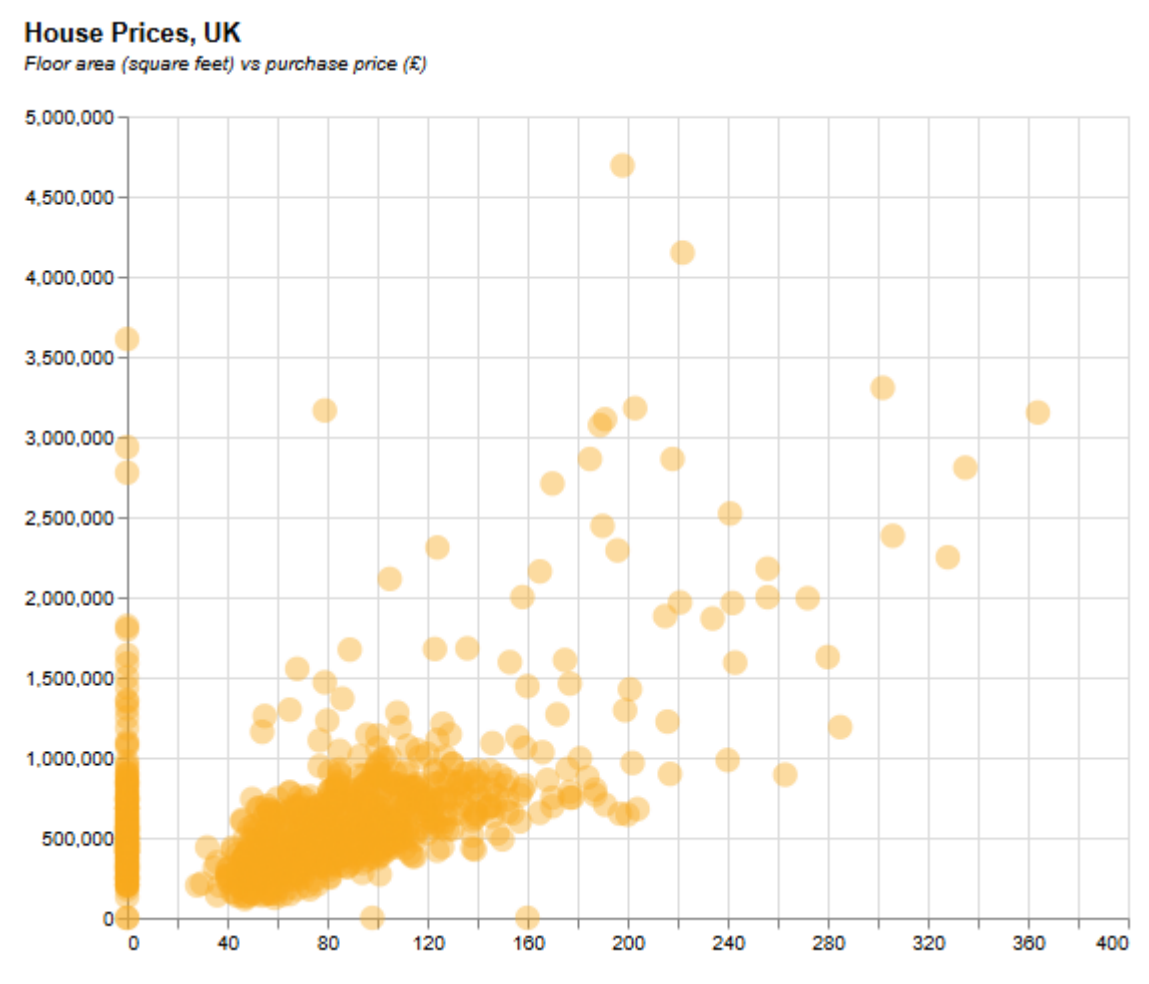
1	priceRent	priceSale	bathrooms	bedrooms	floorAreaSqm	livingrooms	latitude	longitude	outcode	postcode	propertytype	energyRating	address
2	950	235000		2			51.447113	-0.24290749	SW15	SW15 4LZ	Flat/Maisonette		22 Rushmere House,
3	2200	627000	1	3		2	51.535149	-0.4492566	UB10	UB10 0DE	Detached House		18 Baxter Close, Uxb
4	3000	643000			77		51.459309	-0.1172709	SW2	SW2 1EG	Flat/Maisonette	B	Flat 29, Somerset Pla
5	2550	625000		4	139	2	51.472393	0.1292385	DA7	DA7 4SY	Semi-Detached House		196 Brampton Road,
6	2700	815000	2	3	103	1	51.443314	-0.2175111	SW19	SW19 6ER	Mid Terrace House	D	110 Augustus Road,
7	2100	536000	2	3	102	1	51.525314	-0.41874129	UB4	UB4 0AE	End Terrace House		97 Whittington Aven
8	2350	672000	1	3	93	1	51.593266	-0.1782492	N2	N2 0SW	Terrace Property	D	62 Hamilton Road, L
9	3900	927000			85		51.501774	-0.022155	E14	E14 9DA	Flat/Maisonette	B	Flat 1308, Bagshaw E
10	2800	727000	2	3	118	2	51.446777	-0.32677761	TW1	TW1 3NR	Converted Flat	D	44A Church Street, T

Stage 1.

Explore the data, manually

Chart 1.

Plot a simple relationship



All of the charts in this worked example are in the course Dropbox.

[You are welcome to build on them, adapt them, and use them in your own work.]

Chart 2.

Filtering out extreme values

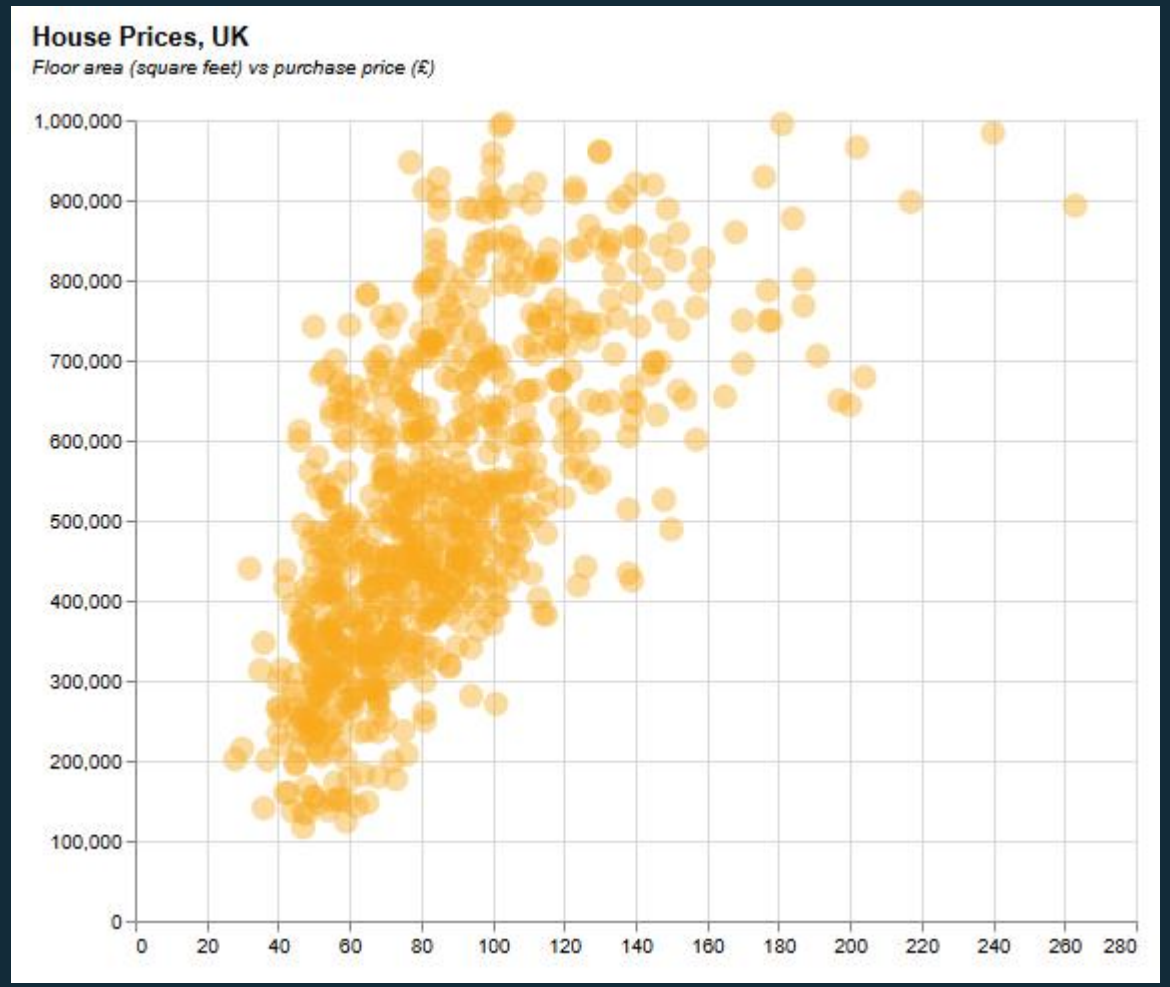


Chart 3.

Use a “facet” encoding to examine types of houses

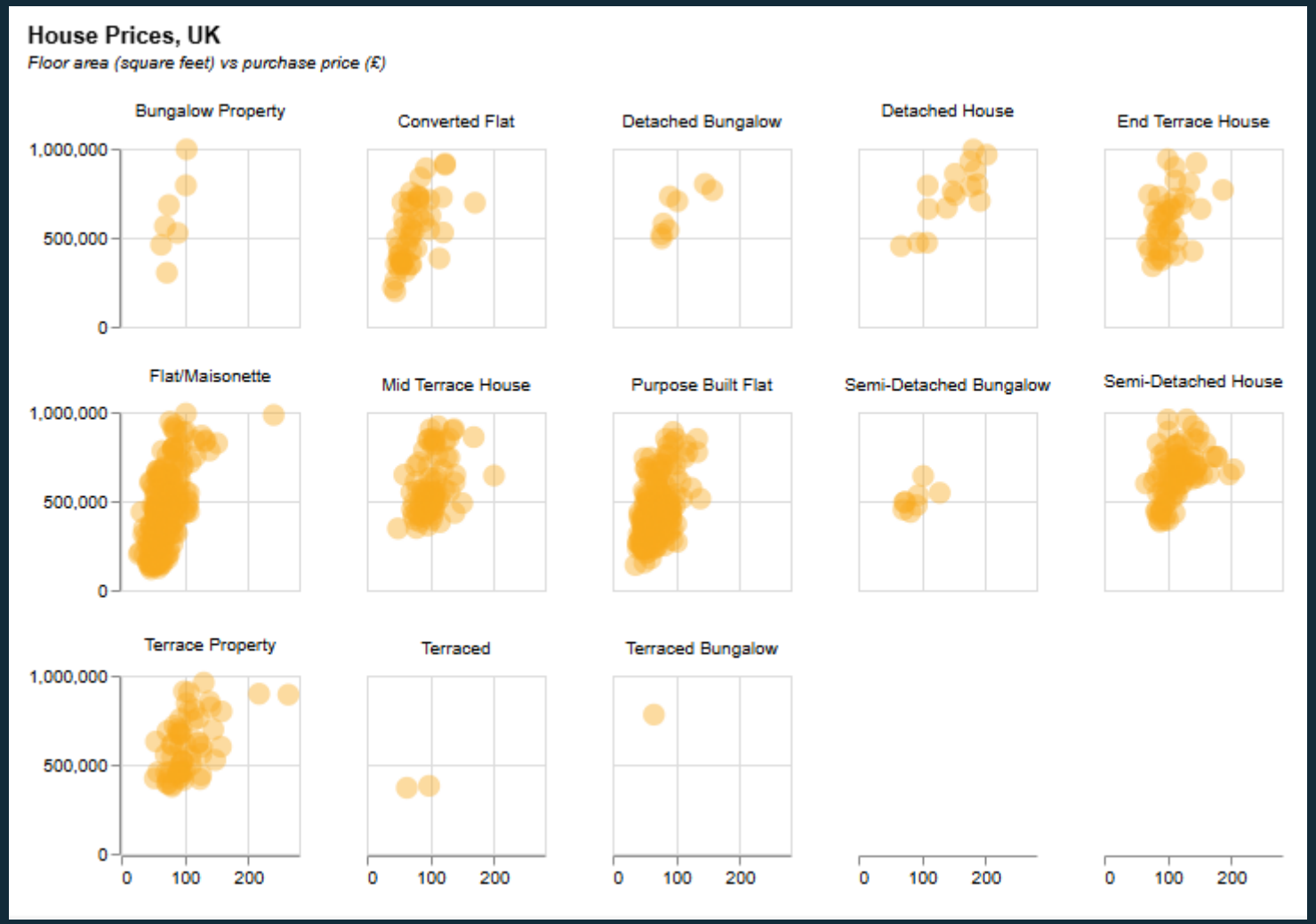


Chart 4.

Creating an interactive drop-down

Chart 5.

Property types in colour

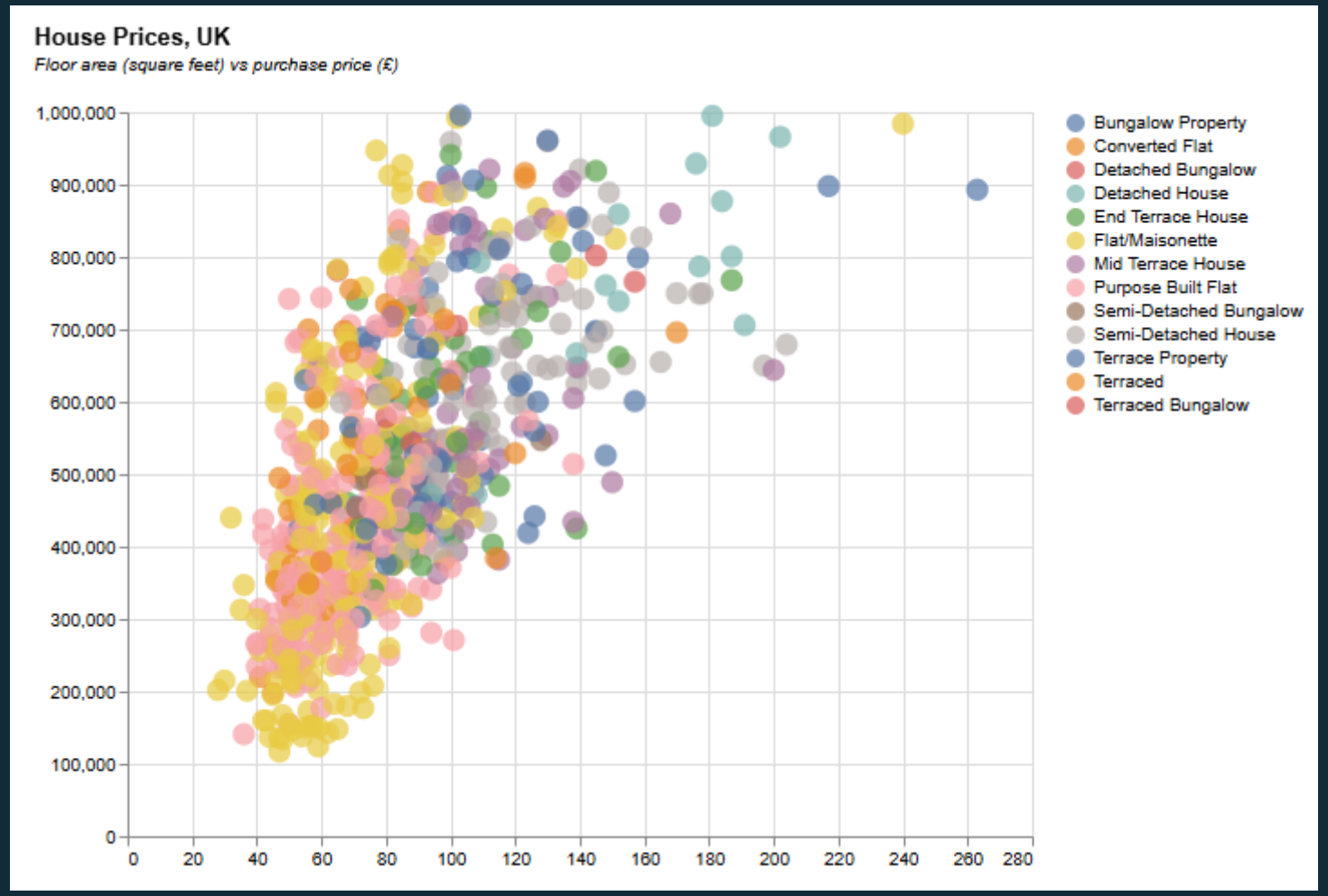
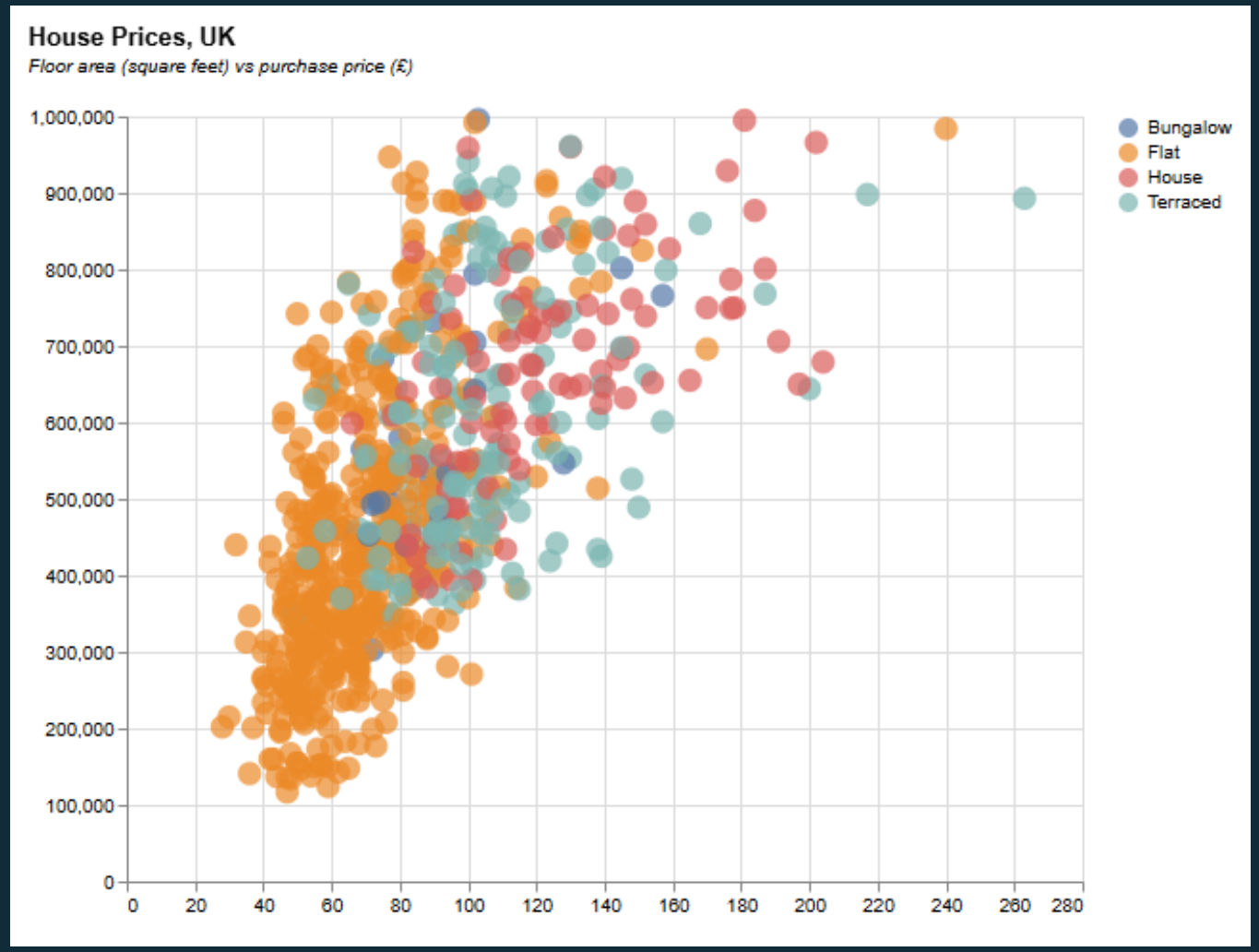


Chart 6.

Colour with simplified types

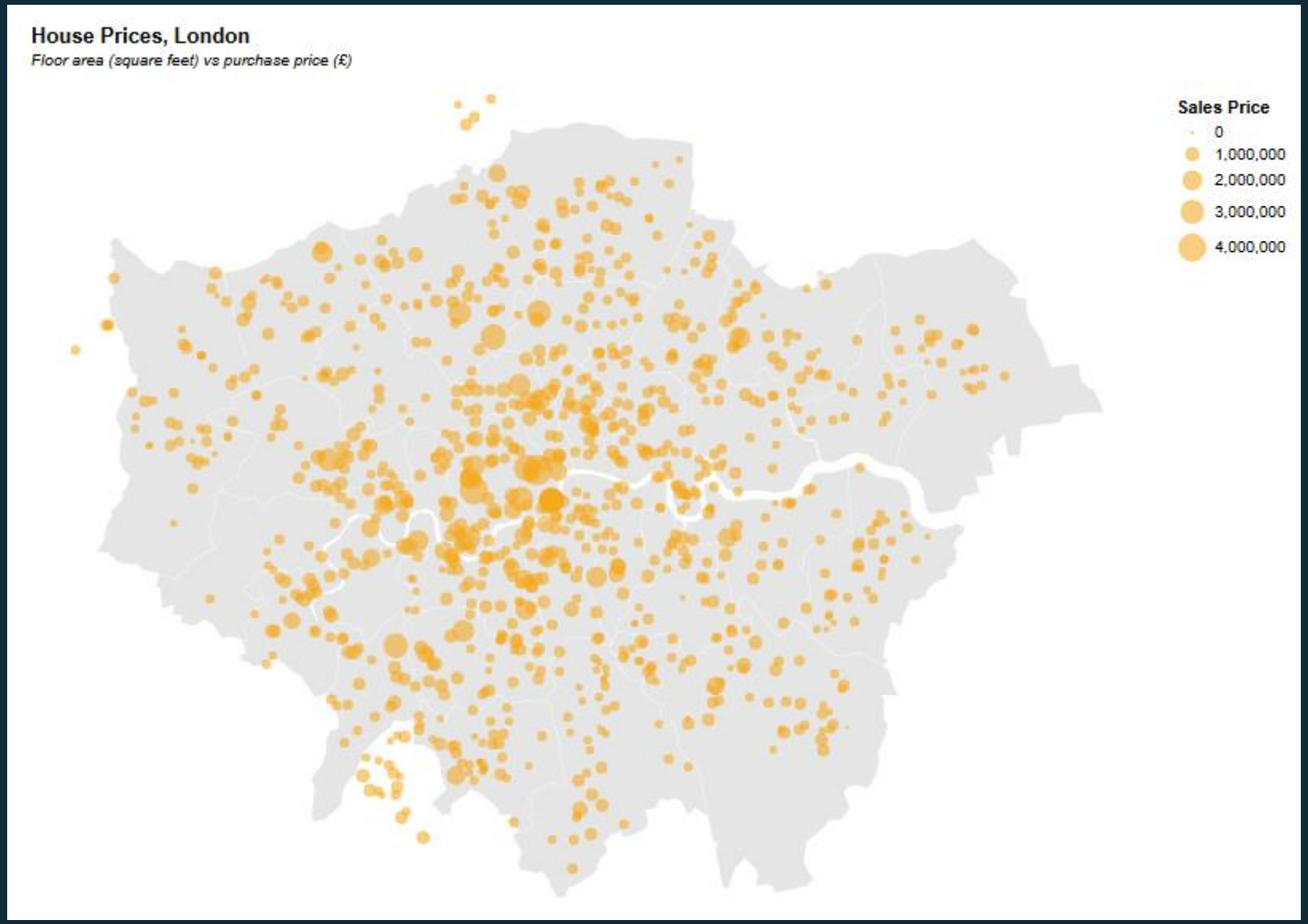


Map 1.

A surprise

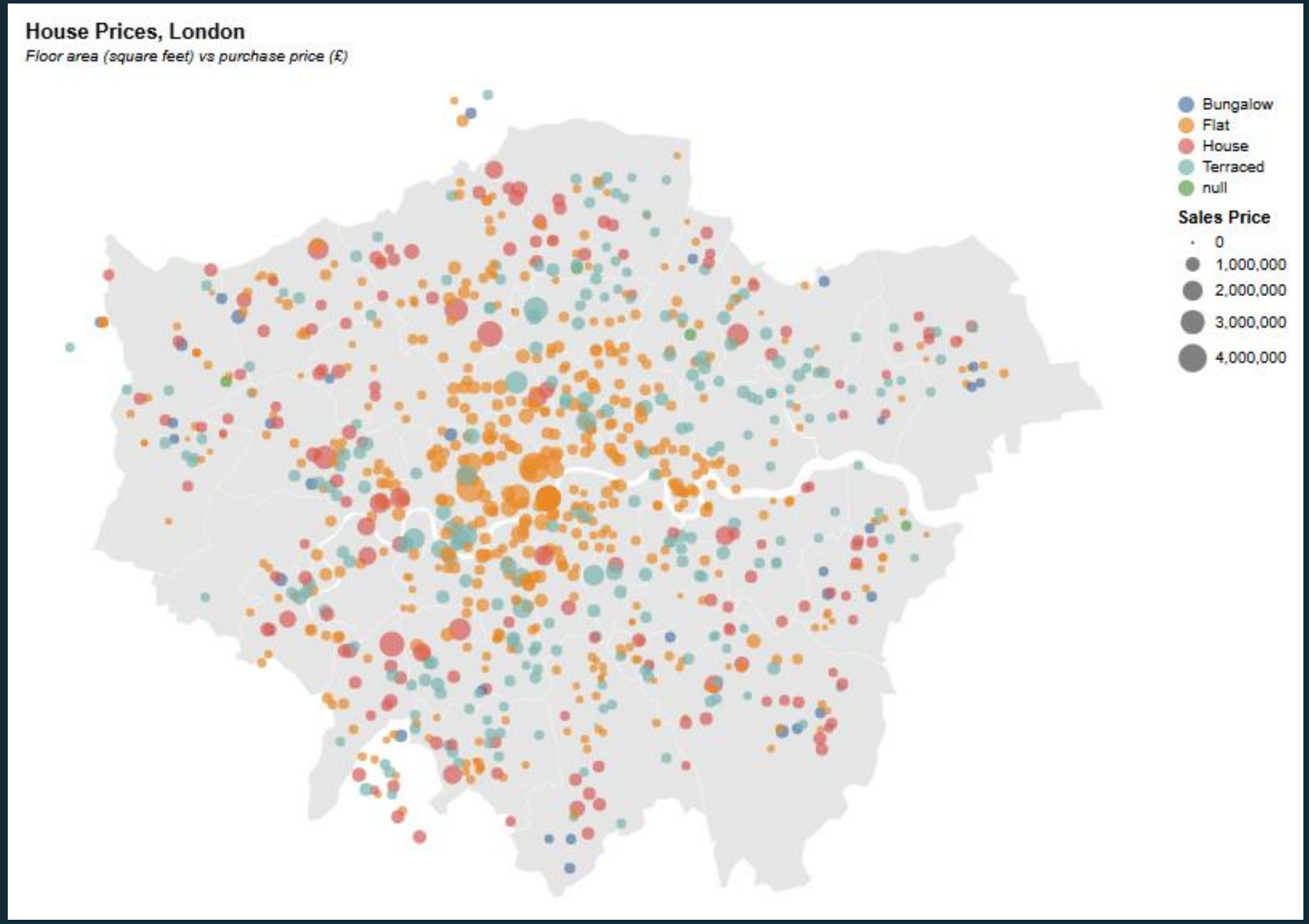
Map 2.

A new context to our data



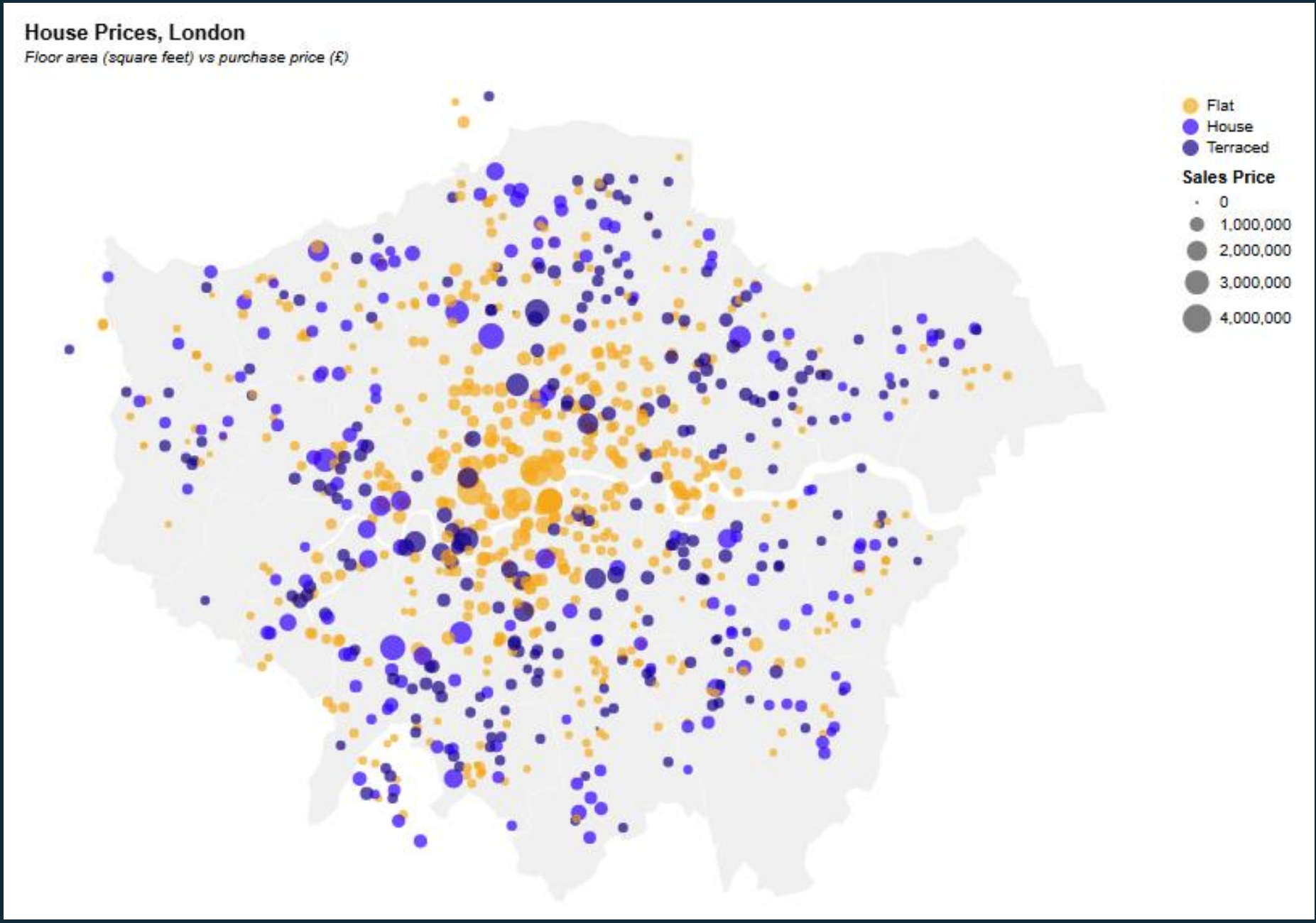
Map 3.

A new context to our data



Map 4.

Houses and Flats

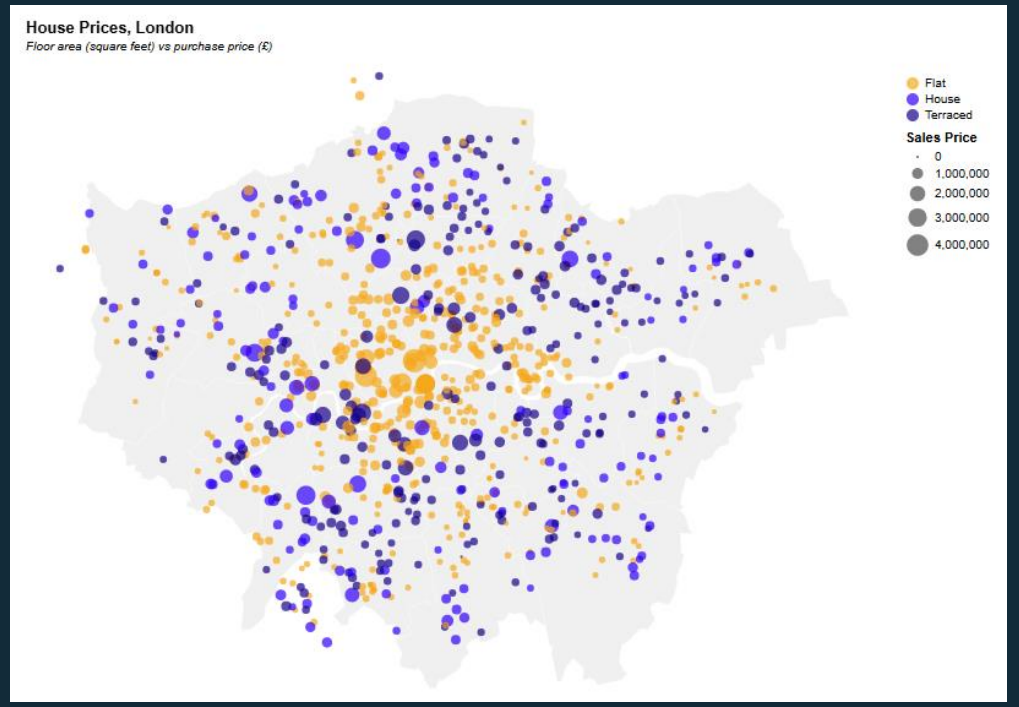


Conclusion.

The importance of pre-visualisation

Preparatory visualisation and analysis will help you:

- Avoid errors.
- Improve the power of the tools that you are using.



Stage 2.

Use the ML tools (supervised learning)

UK house price ML example.

Using ML to predict prices in London

Your final portfolio challenge is to use any ML tool for a policy relevant example.

You are welcome to use the London house price data, or any data of your own.

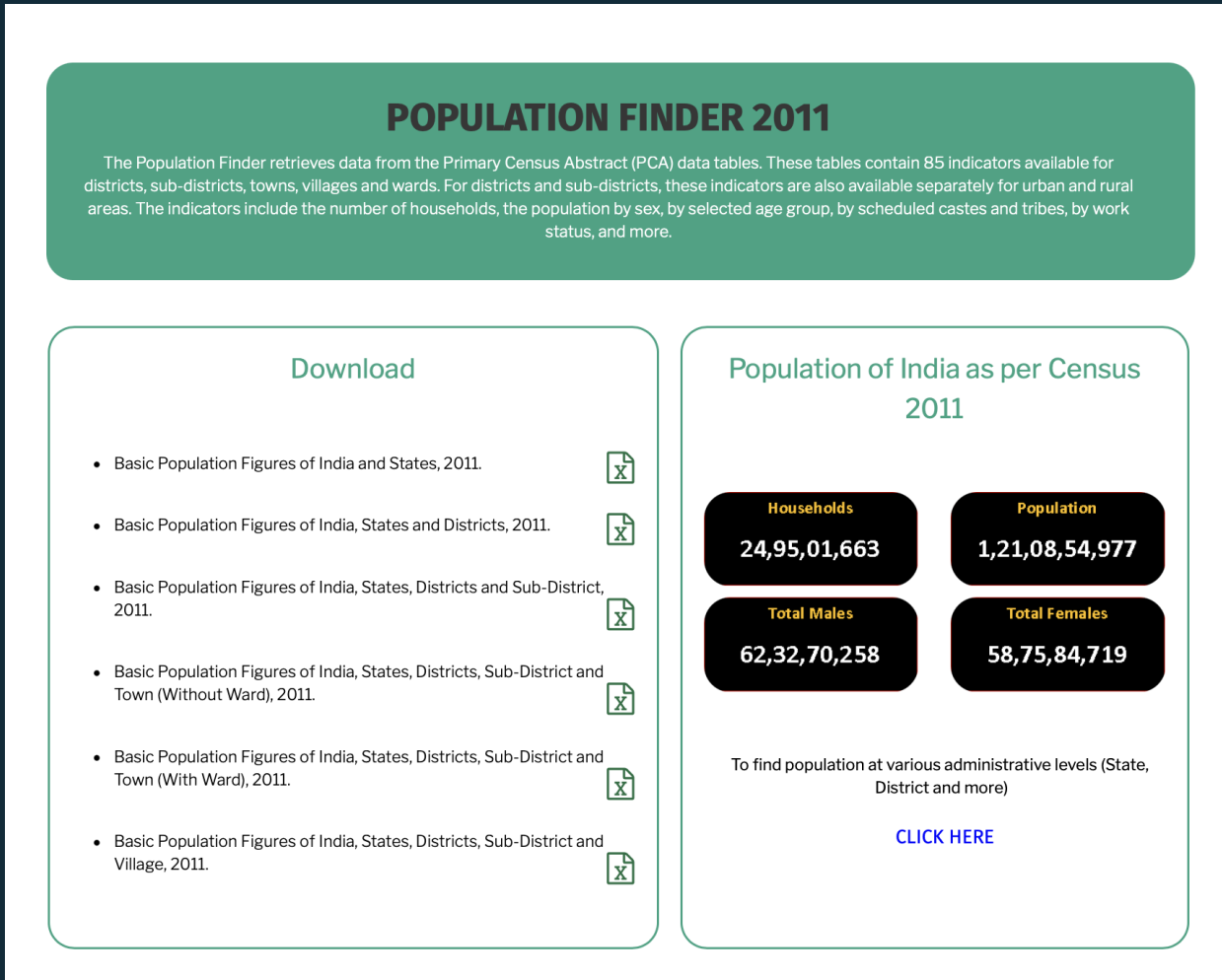
Idea: Indian census granular data .

India, 2011 census village-level data

Though 14 years ago, the 2011 Indian census offers data for over 600,000 villages.

Ideas?

- Clustering by socio-economic conditions
- Predict public service need
- Detect infrastructure gaps



10.3 Unsupervised learning.

Letting the data speak (no labels)

Unsupervised learning.

Intuition and examples

In unsupervised learning, the data is not labelled
The algorithm finds patterns in the data without your help

Examples include:

- Clustering
- Dimensionality reduction (Principal Component Analysis)

Correct answers are not needed

The data is unlabelled. It is cheaper, and can be larger.

Unsupervised learning.

Intuition and examples

In unsupervised learning, the data is not labelled
The algorithm finds patterns in the data without your help

Examples include:

- Clustering
- Dimensionality reduction (Principal Component Analysis)

Correct answers are not needed

The data is unlabelled. It is cheaper, and can be larger.

Unsupervised learning.

When to use?

Similarity and distance between data points play a predominant role.

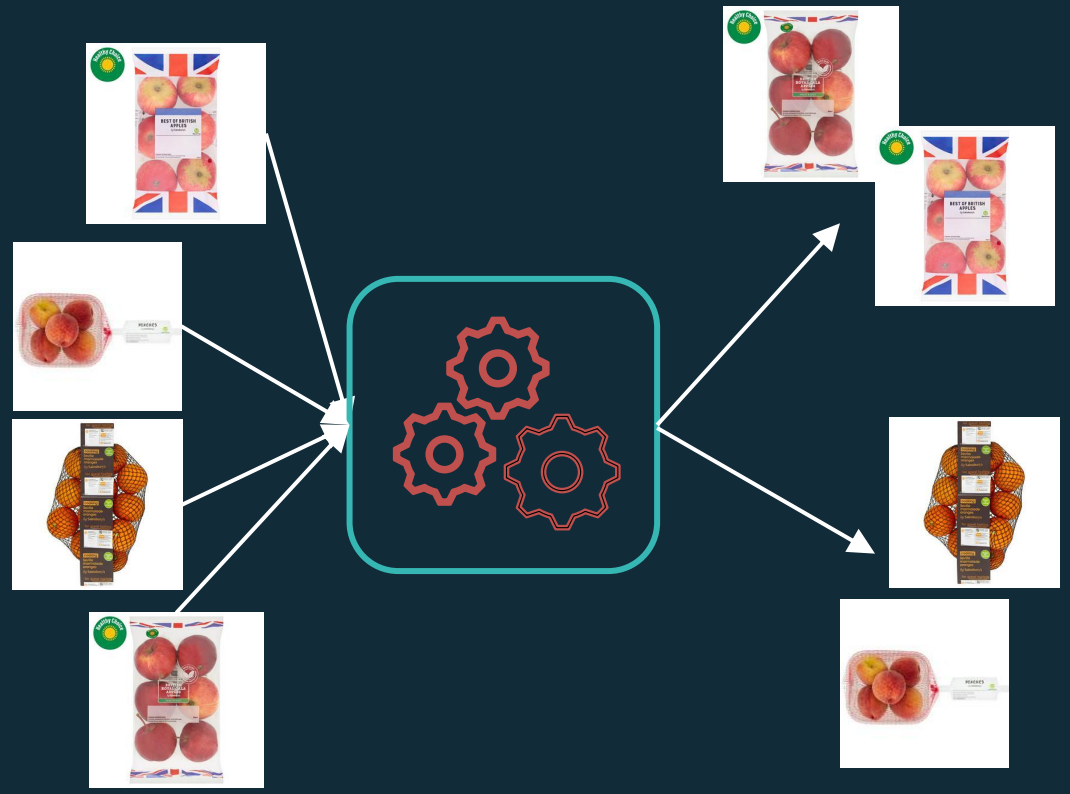
Information retrieval:

- Cluster documents & retrieve similar documents
- Features: set of words in vocabulary

Fraud detection:

- Cluster transactions and flag outliers
- Features: location, time of day etc

Ideas?

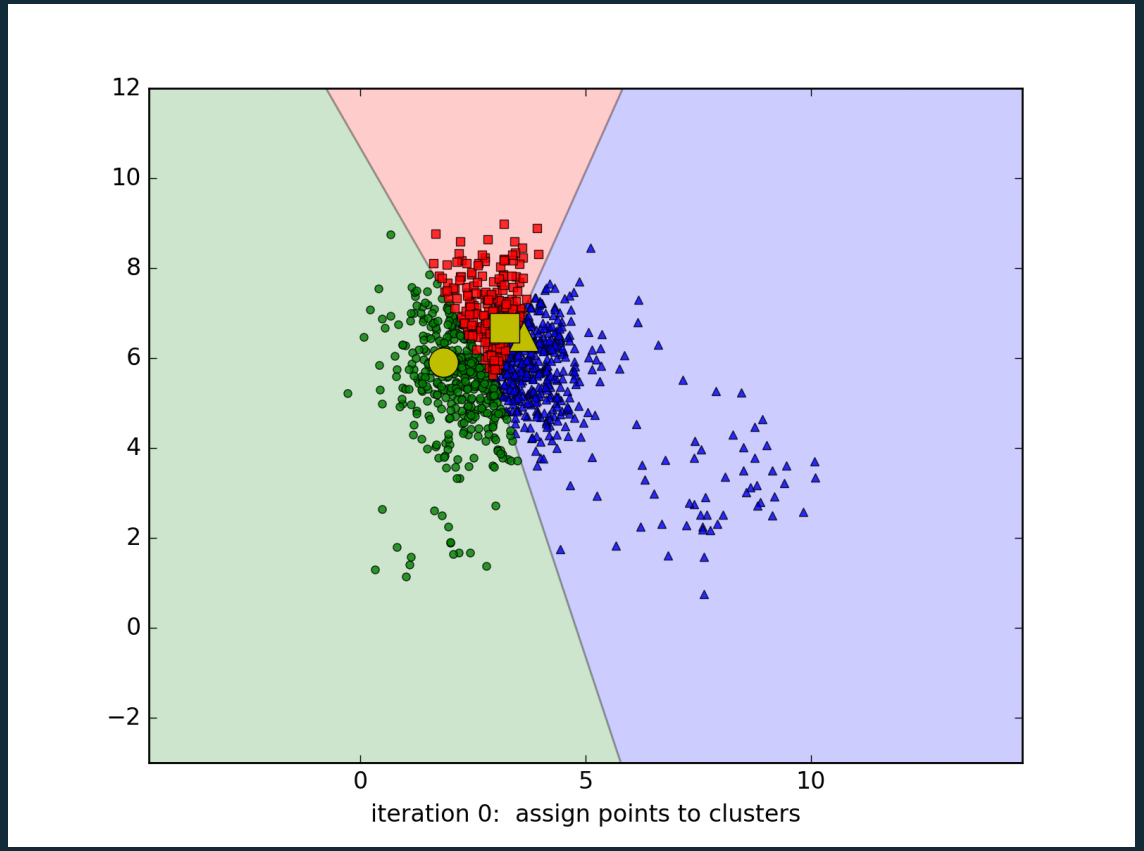


Identify unknown patterns in the feature space *V* based on unlabelled data.

K-means.

Searching for similar groups

- Partitions data into "k" clusters based on similarity parameters.
- Requires choosing k upfront.
- The algorithm iteratively assigns points to nearest cluster centre, then recalculates centres.
- Minimises within-cluster variance using distance metrics.
- The initial point selection can affect final clusters.

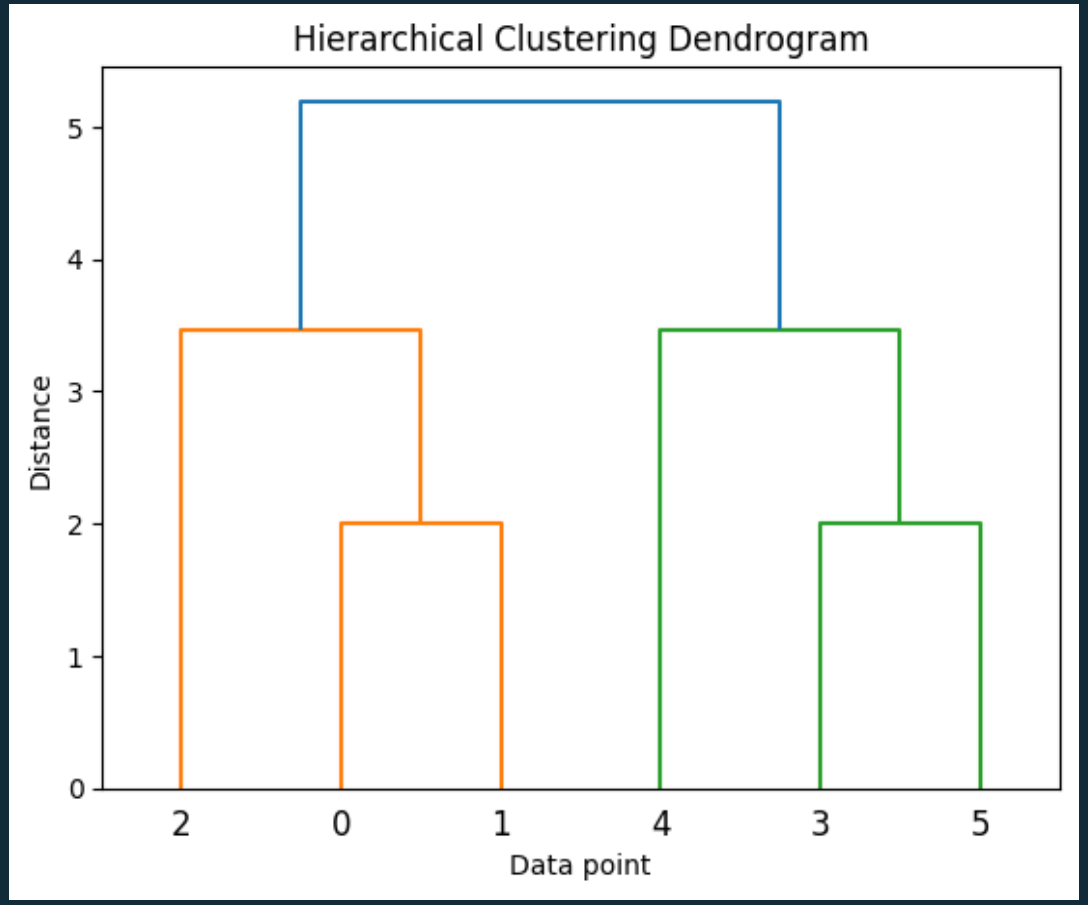


Machinelearningcoban.com

Hierarchical clustering (HAC).

The family tree of your data

- Builds a tree of clusters by merging or splitting groups step-by-step.
- Based on the parameters, each data point is more similar to data points within its cluster than in other clusters.
- Creates a visual tree diagram (dendrogram) showing relationships.
- No need to specify the number of clusters upfront. Instead, draw a horizontal line on the dendrogram at the desired grouping level.

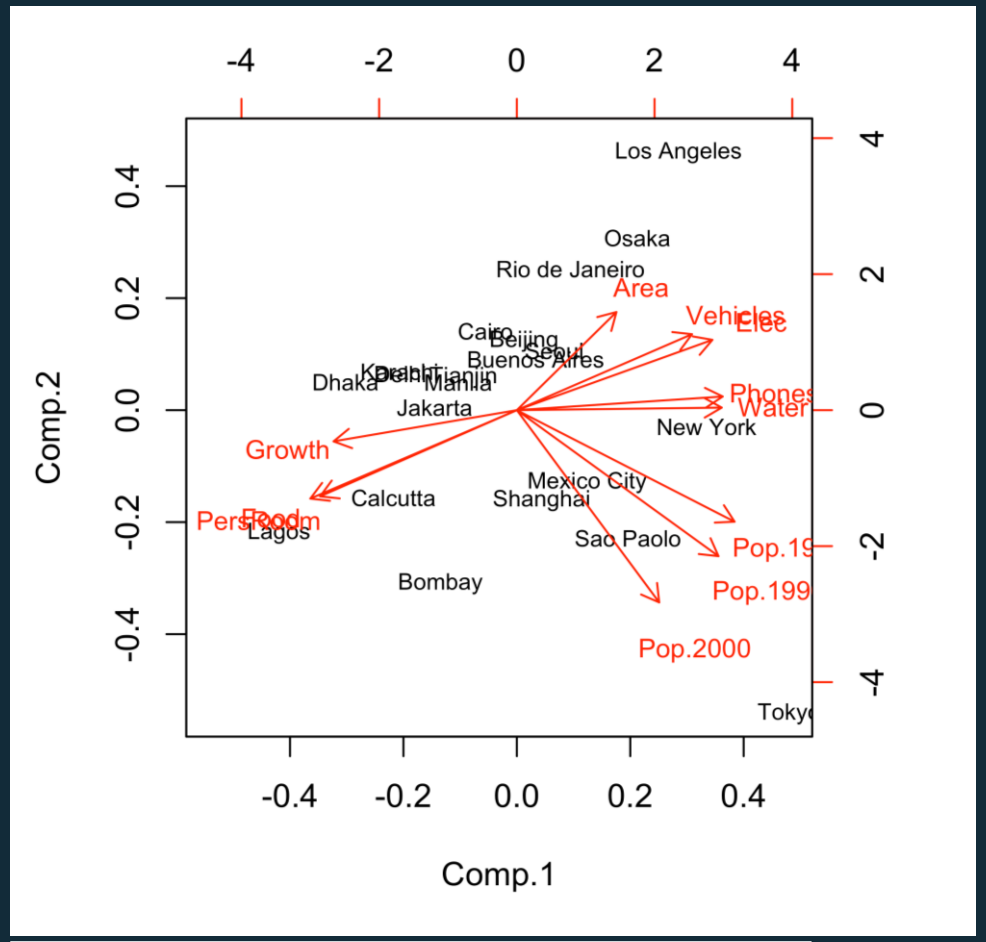


geeksforgeeks.org

Principal Component Analysis (PCA).

A statistical tool to reduce data dimensions

- Identifies "Principal Components" of a given set of parameters by preserving as much variance as possible.
- The idea is to reduce the number of dimensions, while preserving as much information as possible from the data.
- Calculate as many principal components as you want. More components increase dimensionality but also increase information



/pjbartlein.github.io

Discussion.

Types of problem

Discussion.

Classification vs Regression

Are these classification or regression problems?

- House price prediction
- Spam email detection
- Identifying students at risk of dropping out
- Estimating wait times in emergency rooms
- Sentiment analysis
- Predicting crime incidents

Discussion.

Classification vs Regression

Are these classification or regression problems?

- House price prediction – regression $[0, \ldots, 100k, \ldots 1m, . .]$
- Spam email detection – classification $\{not\ spam, spam\}$
- Identifying students at risk of dropping out – classification $\{high, low\}$
- Estimating wait times in emergency rooms – regression $[0, \ldots, 10mins, \ldots, 60mins, \ldots]$
- Sentiment analysis – **either**
- Predicting crime incidents – **either**

Discussion.

Classification vs Regression

Are these classification or regression problems?

- House price prediction - regression
- Spam email detection - classification
- Identifying students at risk of dropping out – classification
- Estimating wait times in emergency rooms - regression
- Sentiment analysis – **either** $\{positive, negative, neutral\}$ or $[0,1]$
- Predicting crime incidents – **either**

Regression for $[0, n]$, or could be classification
$\{low, medium, high\}$ crime area

Discussion.

Classification vs Regression

Are these classification or regression problems?

- House price prediction - regression
- Spam email detection - classification
- Identifying students at risk of dropping out – classification
- Estimating wait times in emergency rooms - regression
- Sentiment analysis – **either**
- Predicting crime incidents – **either**

How we frame the problem depends on our policy needs

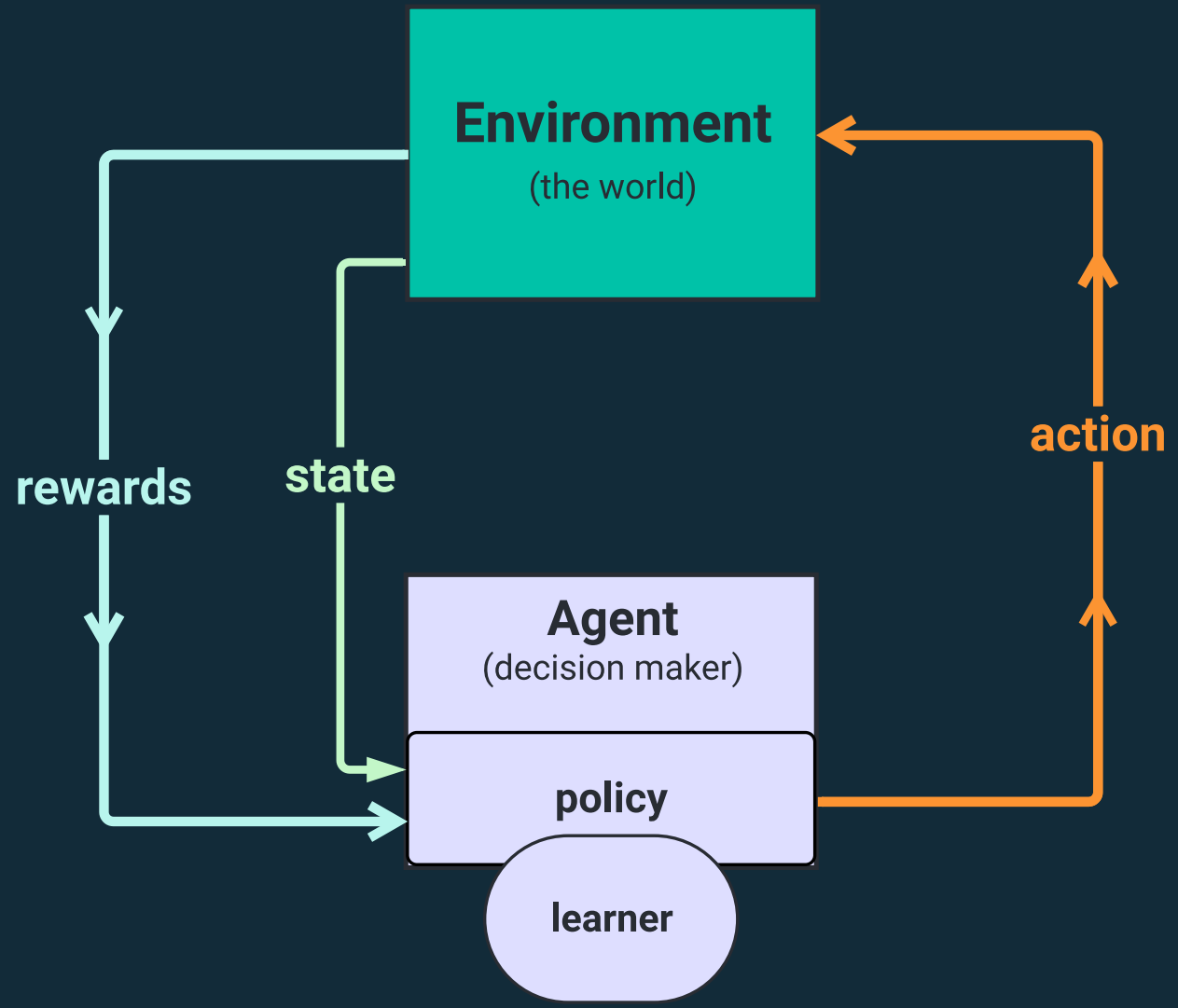
Sometimes we do regression then convert to categories for actionable policy thresholds

10.4 Reinforcement learning.

Feedback loop in an environment

Reinforcement Learning.

What is it?



Reinforcement Learning.

What is it?

Reinforcement Learning (RL):

An agent learns to make decisions by taking actions in an environment and receiving feedback (rewards or penalties).

Goal: learn a strategy that maximises cumulative reward over time.

RL is particularly powerful when:

- The optimal action isn't known in advance
- Actions have delayed consequences
- The environment is complex or changing

RL: Learning in action.

AlphaGo 2016

DeepMind's AlphaGo defeated master Go player Lee Se-dol 3-0 (BBC, 2016).

Early ML vs human programs relied on computational power:

- Checkers/Draughts (1950s): Arthur Samuel's program - early learning, but limited
- Chess (1997): Deep Blue defeats Kasparov by calculating 200 million positions/second

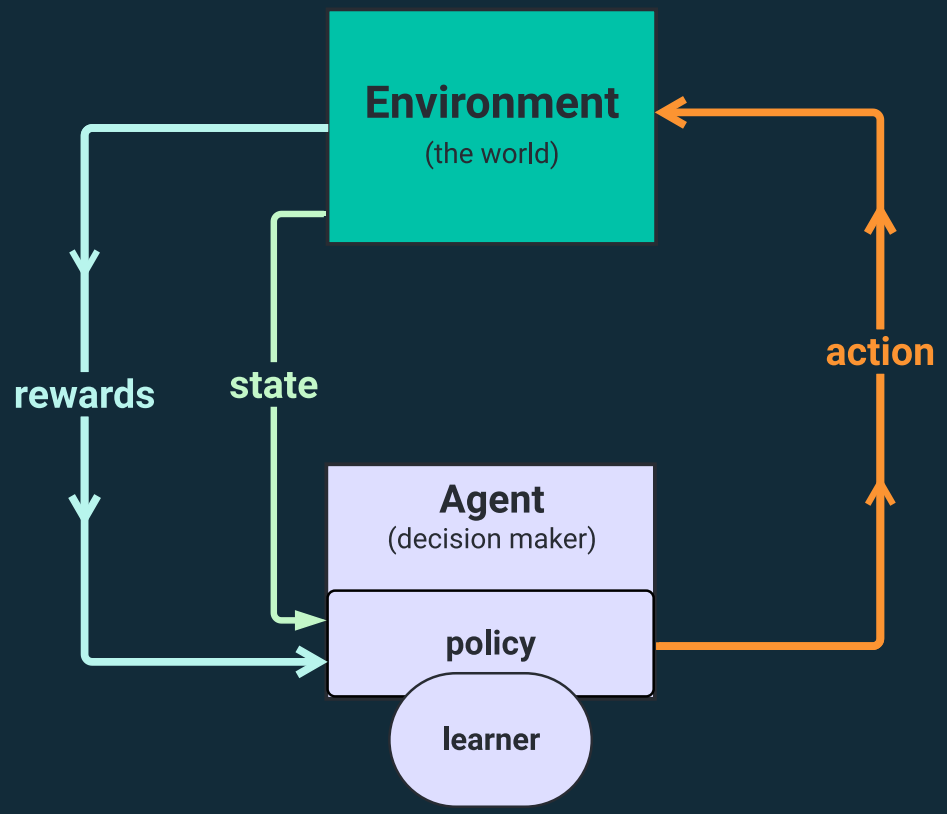
Go is much more complex for a computer to learn than chess, with in the universe. can't be brute forced) more possible positions than atoms

Would be impossible to learn from labelled examples alone.

RL allowed it to learn strategy through self-play, playing millions of times to get incrementally better.

Reinforcement Learning.

The setup: agent & environment



Learning problem requires:

Agent: the learner, and the decision maker.

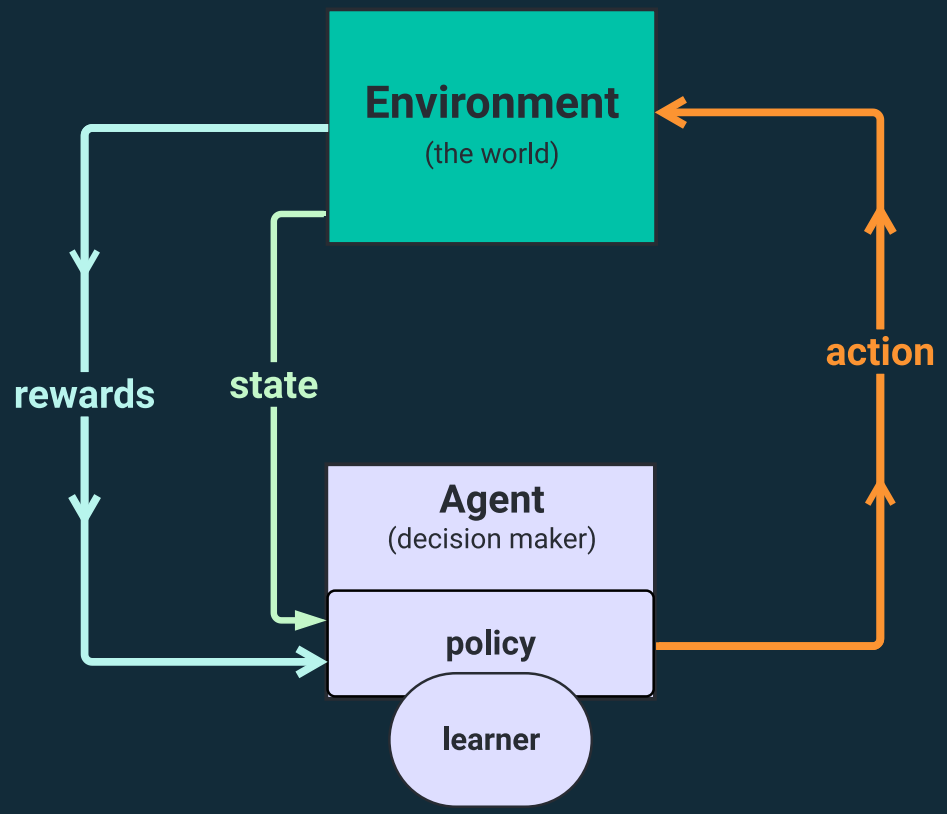
- Contains a Policy: the strategy for choosing actions (what to do in each situation)
- Contains a Learner: updates the policy based on experience

Environment: the space within which our agent interacts. It provides challenges and the desired objective.

- Presents situations (states) to the agent
- Responds to the agent's actions
- Provides feedback through rewards

Reinforcement Learning.

The setup: Actions, States, & Rewards



State: The current situation or observation

- What the agent knows about the environment right now

Action: The decision the agent makes

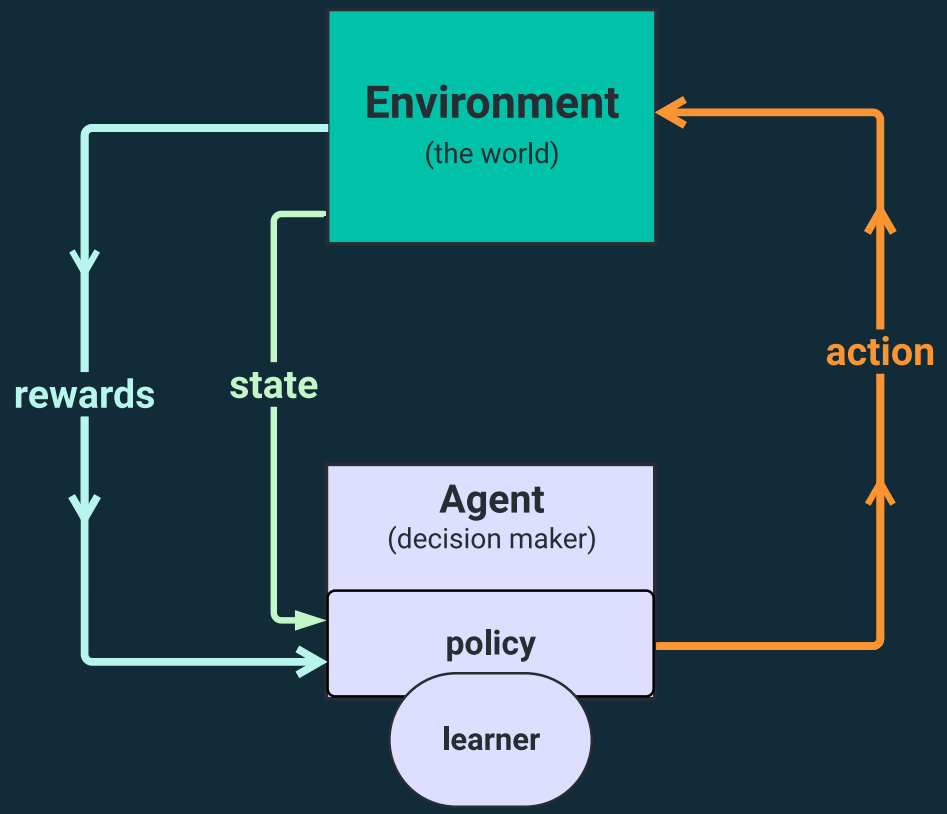
- What the agent does in response to the current state

Reward: Feedback signal from the environment

- Tells the agent how good/bad its action was
- Can be positive (good outcome) or negative (penalty)

Reinforcement Learning.

The setup: Full process

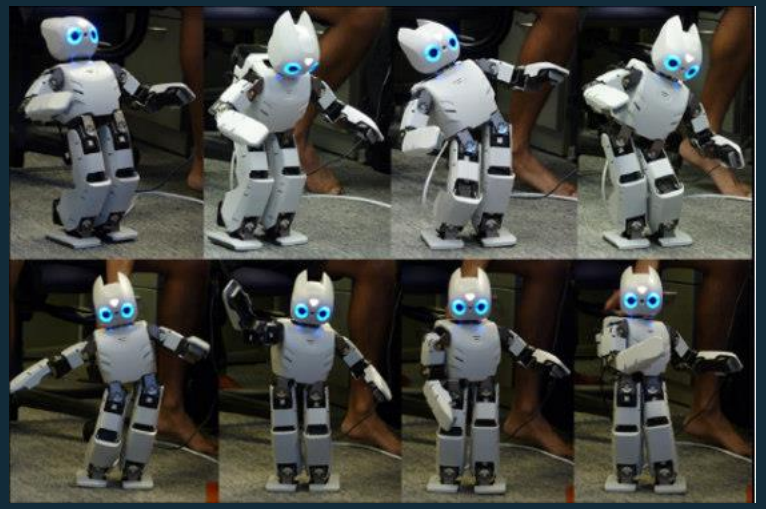


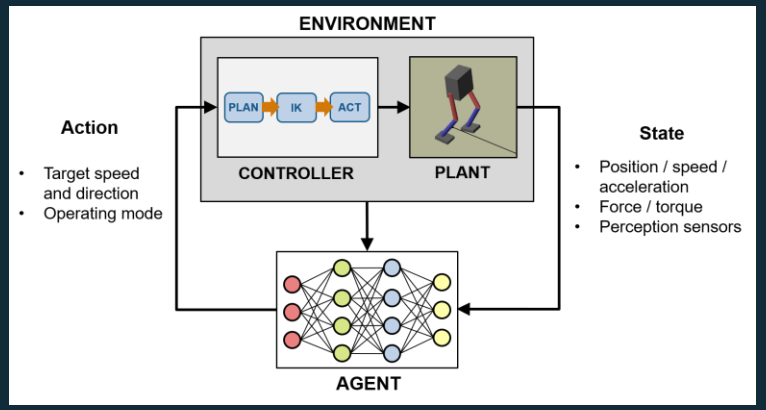
Learning process:

1. Agent observes State from Environment
2. Policy chooses an Action based on current state
3. Environment responds with new State and Reward
4. Learner updates the Policy to improve future decisions
5. Repeat thousands/millions of times → Policy improves

RL: Real World.

An example: teaching a robot to walk



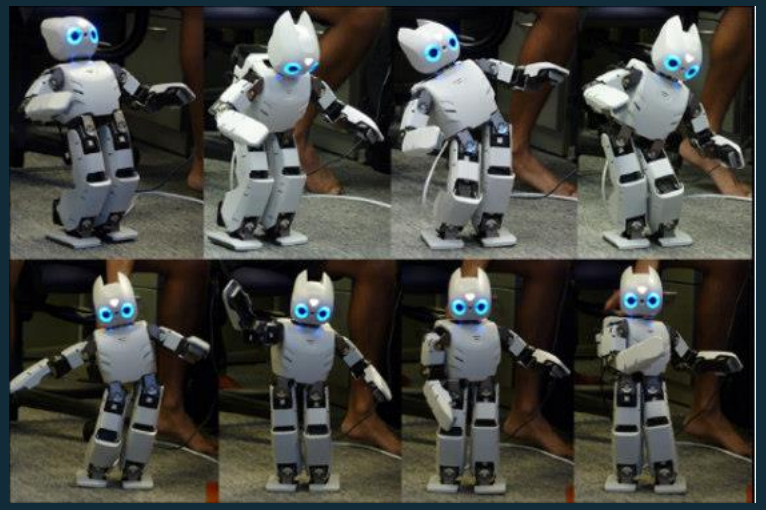


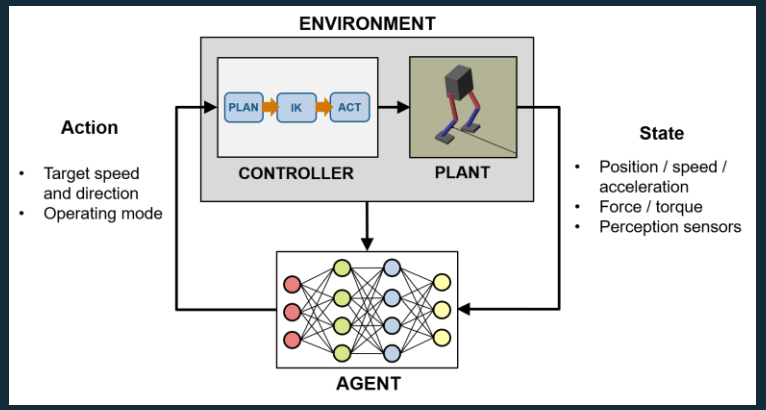
[blog: controlling humanoid rebots]

Component	Example
Agent	The robot's control system (brain)
Policy	Strategy for moving joints: "How much should I bend each joint right now?"
Learner	Algorithm that adjusts the policy after each attempt
Environment	Physics simulation (gravity, friction, ground)
State	Current position, joint angles, velocity, balance
Actions	Motor commands: torque/angle for each joint
Rewards	+1 for each step forward, -10 for falling over, -0.1 for using too much energy

RL: Real World.

An example: teaching a robot to walk





blog: controlling humanoid rebots

Learning journey:

- Episode 1-100: Random flailing → falls immediately → learns "falling = bad”
- Episode 100-500: Discovers standing → stays upright → learns "balance matters”
- Episode 500-2000: Tries different gaits → finds walking → learns "coordinated movement works”
- Episode 2000+: Refines efficient walking → optimises speed vs. energy

(For simulated environments, this could be many millions of trials).

Key: No one programmed “how to walk”. The robot ‘discovered’ it through trial and error, guided by a reward signal.

RL: Challenges.

Core dilemma: exploration vs exploitation

Exploration vs Exploitation. Should the agent:

- Exploit: use what is already knows (play it safe, maximise current reward
- Explore: try new actions to potentially discover better strategies (take risks, learn more)

(Like trying a new restaurant vs one you already know)

Credit assignment problem. Which actions led to success?

- Benefits of certain actions may appear many steps later - which specific decisions along the way deserve credit?
- Solution approaches: Track long-term consequences, discount future rewards.

RL: Strategies.

Learning strategies

Value-based methods (e.g., Q-Learning)

- Learn the "value" of each action in each state
- "How good is it to take action A in state S?"
- Choose actions with highest estimated value

Policy-based methods (e.g., Policy Gradients)

- Directly learn the strategy (policy) without estimating values
- Adjust the policy based on what led to good outcomes
- Better for continuous actions (e.g., robot joint angles)

Model-based methods

- First learn how the environment works (build a model)
- Then plan using that model
- More sample efficient but computationally expensive

Reinforcement Learning.

Resources

RL is challenging to build from scratch, but powerful tools exist to help:

Gymnasium (formerly OpenAI Gym): Python package for reinforcement learning.
Includes reference environments and built-in user interfaces showing the problem.

- Provides standardised environments (CartPole, MountainCar, robotic control, etc.)
- Built-in visualisation to watch your agent learn
- Docs: gymnasium.farama.org/index.html

Other tools:

- Stable-Baselines3: Pre-built RL algorithms (PPO, DQN, A2C) ready to use
- Ray RLlib: For large-scale, distributed RL training
- TensorFlow/PyTorch: For building custom RL algorithms

Recap.

Types of learning

Summary.

Supervised v Unsupervised v Reinforcement

We've seen three distinct examples of machine learning, differentiated by the type of feedback during learning

- In **unsupervised**, we're trying to identify unknown patterns or structures in the input (unlabelled data), without any feedback about correctness.
- In **supervised**, we are learning an unknown function to get from a set of known inputs to their known outputs. The training data explicitly tells us the correct answer for each example (i.e. *labelled*).
- In **reinforcement**, we learn from rewards and punishments following a sequence of actions in an environment. We don't know the 'correct' action for each situation, only whether the overall strategy led to good or bad outcomes.

10.5 Practical.

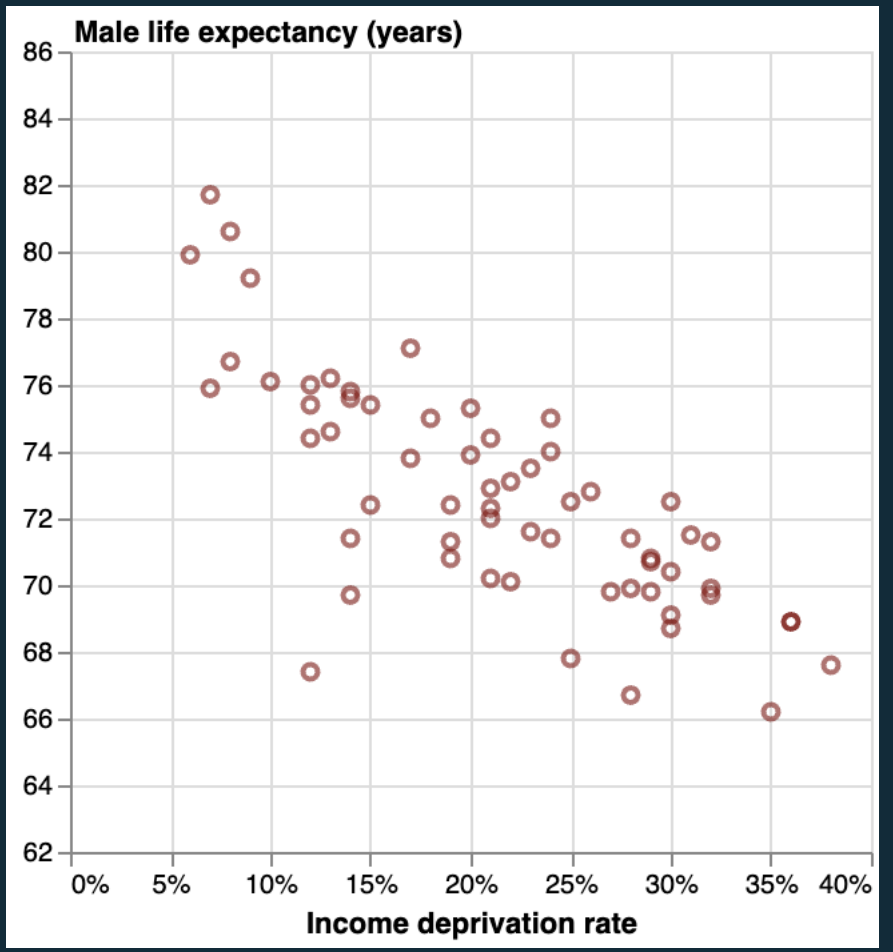
Worked examples: SL, UL, RL.

SL: Linear Regression.

Regression with sci-kit learn – the “Glasgow Effect”

Glasgow health data for 61 neighbourhoods (data)

1	areaName	incomeDeprevation	employmentDeprivation	childPoverty	femaleLE	maleLE	disabilityRate
2	Anniesland, Jordanhill and Whiteinch	0.14	0.15	0.14	80.8	75.8	0.19
3	Arden and Carnwadric	0.26	0.25	0.34	76.0	72.8	0.22
4	Baillieston and Garrowhill	0.12	0.12	0.14	81.6	76.0	0.21
5	Balornock and Barmulloch	0.29	0.27	0.38	78.2	70.8	0.30
6	Bellahouston, Craigton and Mosspark	0.20	0.18	0.22	80.5	73.9	0.29
7	Blackhill and Hogganfield	0.24	0.23	0.33	77.5	75.0	0.25
8	Blairdardie	0.18	0.16	0.20	79.4	75.0	0.25
9	Broomhill and Partick West	0.12	0.11	0.19	79.7	75.4	0.17



Appears to be a strong negative relationship:
neighbourhoods with higher income deprivation have lower male life expectancy.

Our supervised learning problem is to learn a function that approximates this relationship.

SL: Linear Regression.

Prepare training data

1. Separate data in input data and output data

1	areaName	incomeDeprevation	employmentDeprivation	childPoverty	femaleLE	maleLE	disabilityRate
2	Anniesland, Jordanhill and Whiteinch	0.14	0.15	0.14	80.8	75.8	0.19
3	Arden and Carnwadric	0.26	0.25	0.34	76.0	72.8	0.22
4	Baillieston and Garrowhill	0.12	0.12	0.14	81.6	76.0	0.21
5	Balornock and Barmulloch	0.29	0.27	0.38	78.2	70.8	0.30
6	Bellahouston, Craigton and Mosspark	0.20	0.18	0.22	80.5	73.9	0.29
7	Blackhill and Hogganfield	0.24	0.23	0.33	77.5	75.0	0.25
8	Blairdardie	0.18	0.16	0.20	79.4	75.0	0.25
9	Broomhill and Partick West	0.12	0.11	0.19	79.7	75.4	0.17

```
# Select subset of columns and keep as a DataFrame. Convert to numpy array with .values
X = data[['incomeDeprevation']].values
# Select target column (ie the labels)
y_male = data['maleLE']
```

SL: Linear Regression.

Fit the model

2. Train the model to estimate the function between our inputs (X) and outputs (y).

```
from sklearn import linear_model
# Create and fit the linear regression model
model_male = linear_model.LinearRegression()
model_male.fit(X, y_male)
```

After fitting the model, we can check the model parameters (i.e. coefficients and intercept)

```
print(model_male.coef_)
➔ [-31.00897722]

 print(model_male.intercept_)
-> 79.23912187855267
```

Simple linear model:
y = 79.24 - 31.01X

SL: Linear Regression.

Generating predictions

3. After fitting the model, we can use '.predict()' to generate predictions over our input data.

```
# Generate predictions and add new column to original dataframe
data['predicted_maleLE'] = model_multi.predict(X)
```

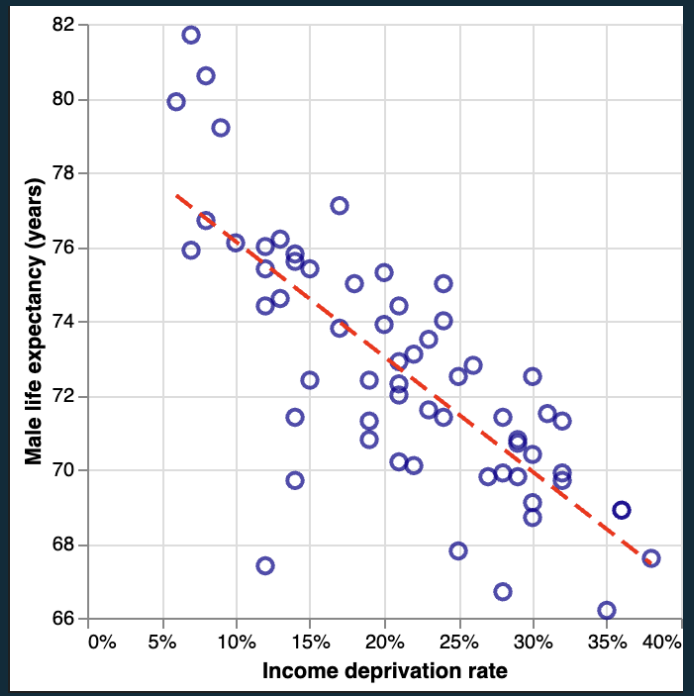
areaName	incomeDeprevation	employmentDeprivation	childPoverty	femaleLE	maleLE	disabilityRate	predicted_maleLE
Anniesland, Jordanhill and Whiteinch	0.14	0.15	0.14	80.8	75.8	0.19	74.897865
Arden and Carnwadric	0.26	0.25	0.34	76.0	72.8	0.22	71.176788
Baillieston and Garrowhill	0.12	0.12	0.14	81.6	76.0	0.21	75.518045

Now for every input x ('incomeDeprivation'), we have a predicted output y (male life expectancy).

SL: Linear Regression.

Visualise regression

areaName	incomeDeprevation	employmentDeprivation	childPoverty	femaleLE	maleLE	disabilityRate	predicted_maleLE
Anniesland, Jordanhill and Whiteinch	0.14	0.15	0.14	80.8	75.8	0.19	74.897865
Arden and Carnwadric	0.26	0.25	0.34	76.0	72.8	0.22	71.176788
Baillieston and Garrowhill	0.12	0.12	0.14	81.6	76.0	0.21	75.518045



Visualise the model fit by layering the predicted y-values with our actual actual data

Simple linear model:
y = 79.24 - 31.01X

SL: Multiple Regression.

Beyond one variable

Moving from single to multiple input variables is easy.
– just select multiple columns from our DataFrame

```
# Select input data and convert to matrix format
X_multi = data[['incomeDeprevation', 'employmentDeprivation', 'childPoverty']].values

# Select target column (ie the labels)
y_male = data['maleLE']



# The rest is the same ...
model_male = linear_model.LinearRegression()
model_male.fit(X_multi, y_male)

print(model_male.coef_)  ->  [31.81011925, -32.56560719, -29.15891298, 8.34715914]
print(model_male.intercept_)  ->  79.26039195704499

# Generate predictions on input matrix and add to new column
data['pred_multi'] = model_multi.predict(X_multi)
```

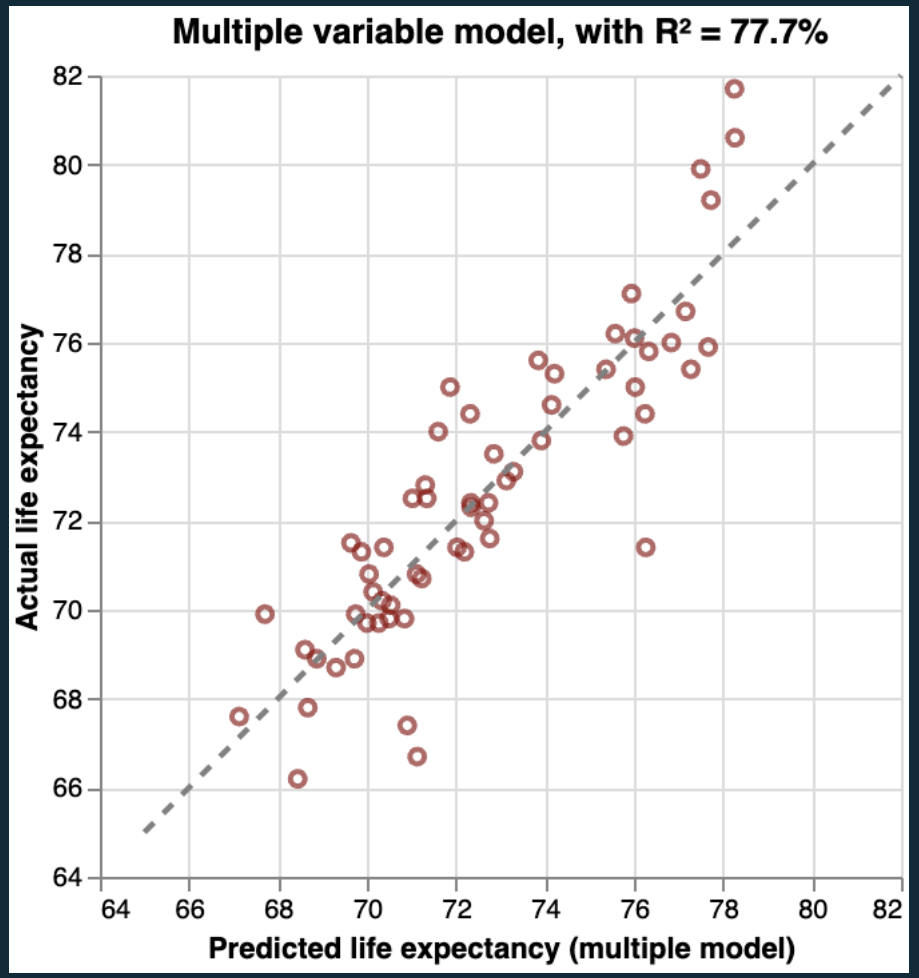
SL: Multiple Regression.

Visualising results, example 1

Once we move beyond two dimensions in our analysis, visualising the model parameters and results can become more complicated.

A helpful way to start is to think about what we are interested in, e.g.

- How do y-predicted compare to actual y-values?
- How do predictions vary between our multi-variate and single-variate models?



Plotting predicted y against actual y values

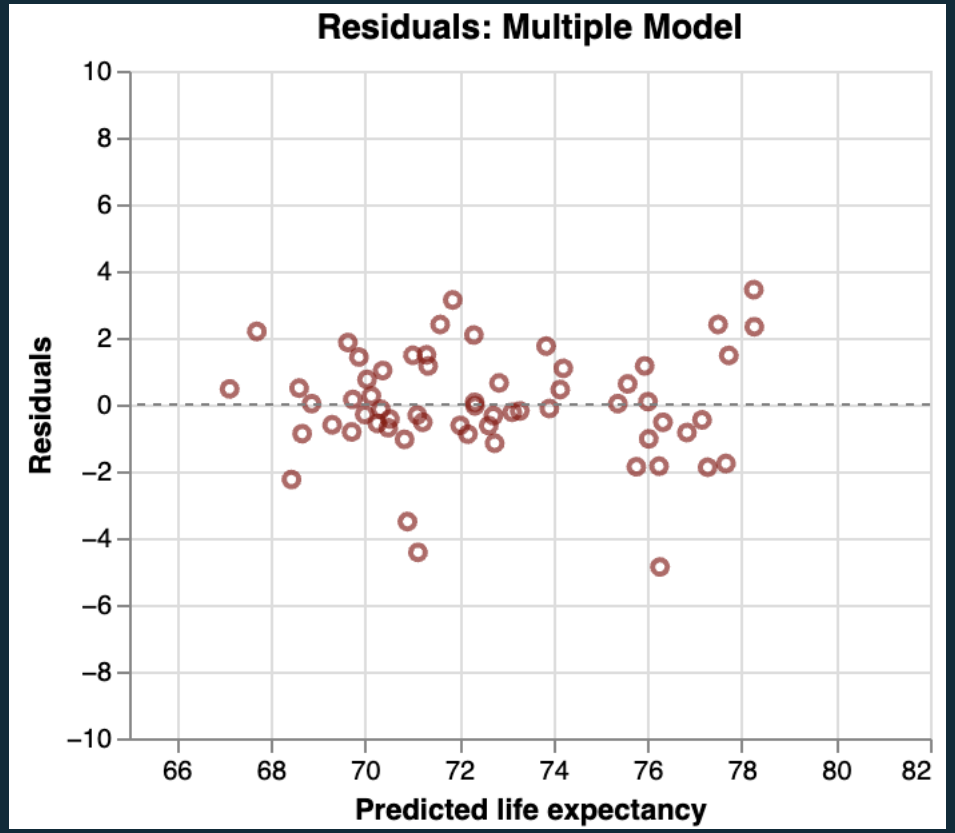
SL: Multiple Regression.

Visualising results, example 2 - residuals

Once we move beyond two dimensions in our analysis, visualising the model parameters and results can become more complicated.

A helpful way to start is to think about what we are interested in, e.g.

- How do y-predicted compare to actual y-values?
- How do predictions vary between our multi-variate and single-variate models?



Calculating residuals (y_actual – y_predicted) and plotting against predictions.

10.6 Glossary.

AI terms & resources

Python: Pandas methods.

Useful methods for summarising data

Basic DataFrame Information

- .shape: Returns a tuple of (rows, columns) showing the dimensions of your DataFrame
- .columns: Returns an array of all the column headers
- .dtypes: Shows the data type of each column (e.g., int64, float64, object)
- .info(): Provides a concise summary including column names, non-null counts, and memory usage

Data Inspection

- .head(n=5): Displays the first n rows (default 5) of the DataFrame
- .tail(n=5): Displays the last n rows (default 5) of the DataFrame
- .sample(n=5): Returns a random sample of n rows from the DataFrame

Summary Statistics

- .describe(): Returns a table of key summary statistics for numerical columns (count, mean, std, min, quartiles, max)
- .describe(include='all'): Includes statistics for all column types, not just numerical
- .value_counts(): For a Series, shows unique values and their frequencies
- .nunique(): Returns the number of unique values in each column
- .unique(): Returns array of unique values (for Series only)

Key terms.

Part 1

Machine Learning (ML). A branch of AI where algorithms learn patterns from data rather than fixed rules, improving performance through experience. Models trained on past examples can then predict, classify or create new outputs.

Artificial Intelligence (AI). Umbrella term for systems that mimic human cognitive functions. They learn from data (adaptability) and act with some independence (autonomy) to solve tasks.

Generative AI. Subset of AI that creates new content—text, images, audio, code or video—rather than just analysing data. Deep models learn the patterns of huge datasets and, from a prompt, synthesise plausible originals.

Deep Learning. Advanced machine learning that trains very large, multi-layer (“deep”) neural networks to extract increasingly abstract features from data. Deep architectures enable learning hierarchical representations without manual feature engineering.

Key terms.

Part 2

Foundation (Base) Model. Large AI systems pre-trained on vast, varied data, giving it broad, general knowledge. These models capture broad knowledge and patterns that can be refined for specific applications.

Multimodal Model. AI that natively links several data types—text, images, audio, video—inside one network. It can describe a picture, draw an image from a caption or sync speech with animation using shared “sense-making”.

Neural Networks (Artificial). Brain-inspired webs of simple computing units (“neurons”) organised in layers. They process data through weighted connections, learn patterns via repeated exposure, and excel at tasks requiring pattern recognition.

Generative Adversarial Networks (GANs). AI systems using two competing networks: a generator creating synthetic data and a discriminator identifying fakes. This competition drives the generator to produce increasingly realistic outputs mimicking training data.

Key terms.

Part 3

Transformers. Neural networks that process entire sequences simultaneously, using attention mechanisms to weigh relationships between elements regardless of distance, enabling powerful language understanding.

Self-attention. Mechanism allowing models to weigh relationships between all sequence elements by computing relevance scores, helping capture context regardless of positional distance.

Large Language Models (LLMs). Neural networks trained on vast amounts of text data to predict and generate human language. These models learn statistical patterns in language, enabling text generation, translation, summarisation, and conversational abilities.

Diffusion. Generative models that gradually transform random noise into structured data by learning to reverse a process that adds noise. They create high-quality images by iteratively denoising random patterns.

Automated Data Visualisation for Policymaking.

Session 10
Autumn 2025