CIVIL-557

# Decision-aid methodologies in transportation

## Lecture 2: Deterministic methods and discrete metrics

Tim Hillel

Transport and Mobility Laboratory TRANSP-OR
École Polytechnique Fédérale de Lausanne EPFL

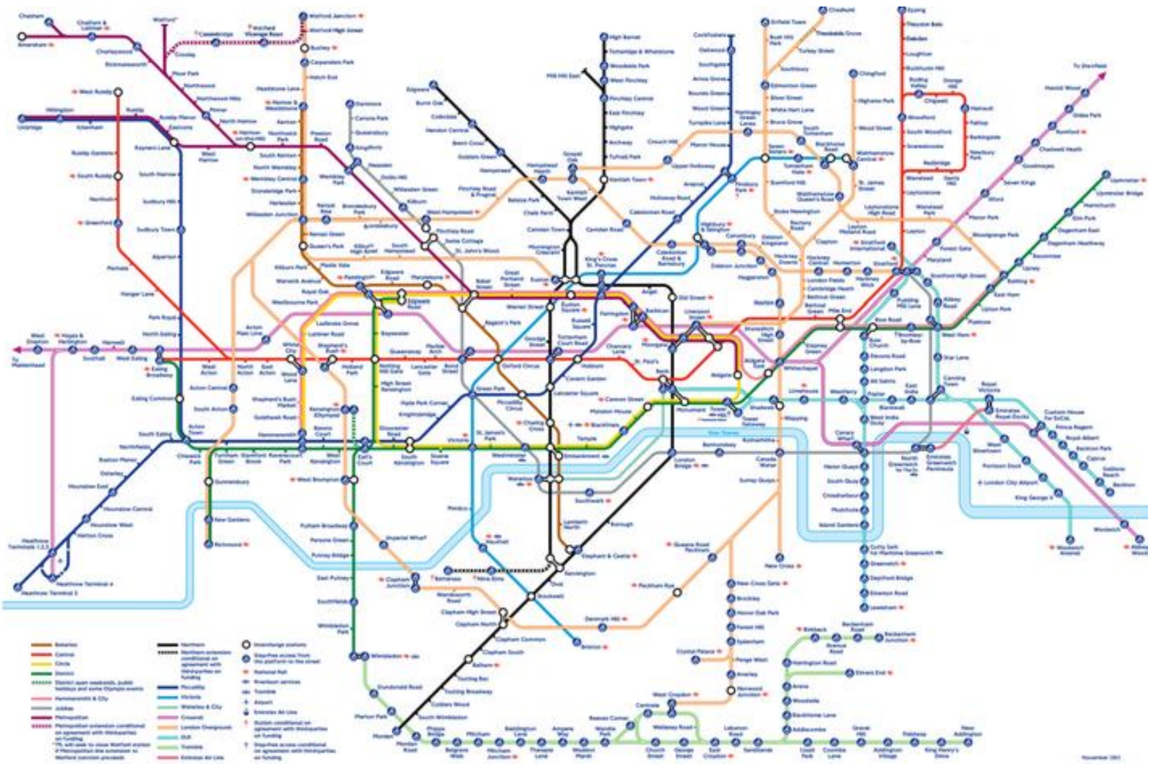
# Motivation

How to **aid** informed **decisions** for **transportation** network management and investment?

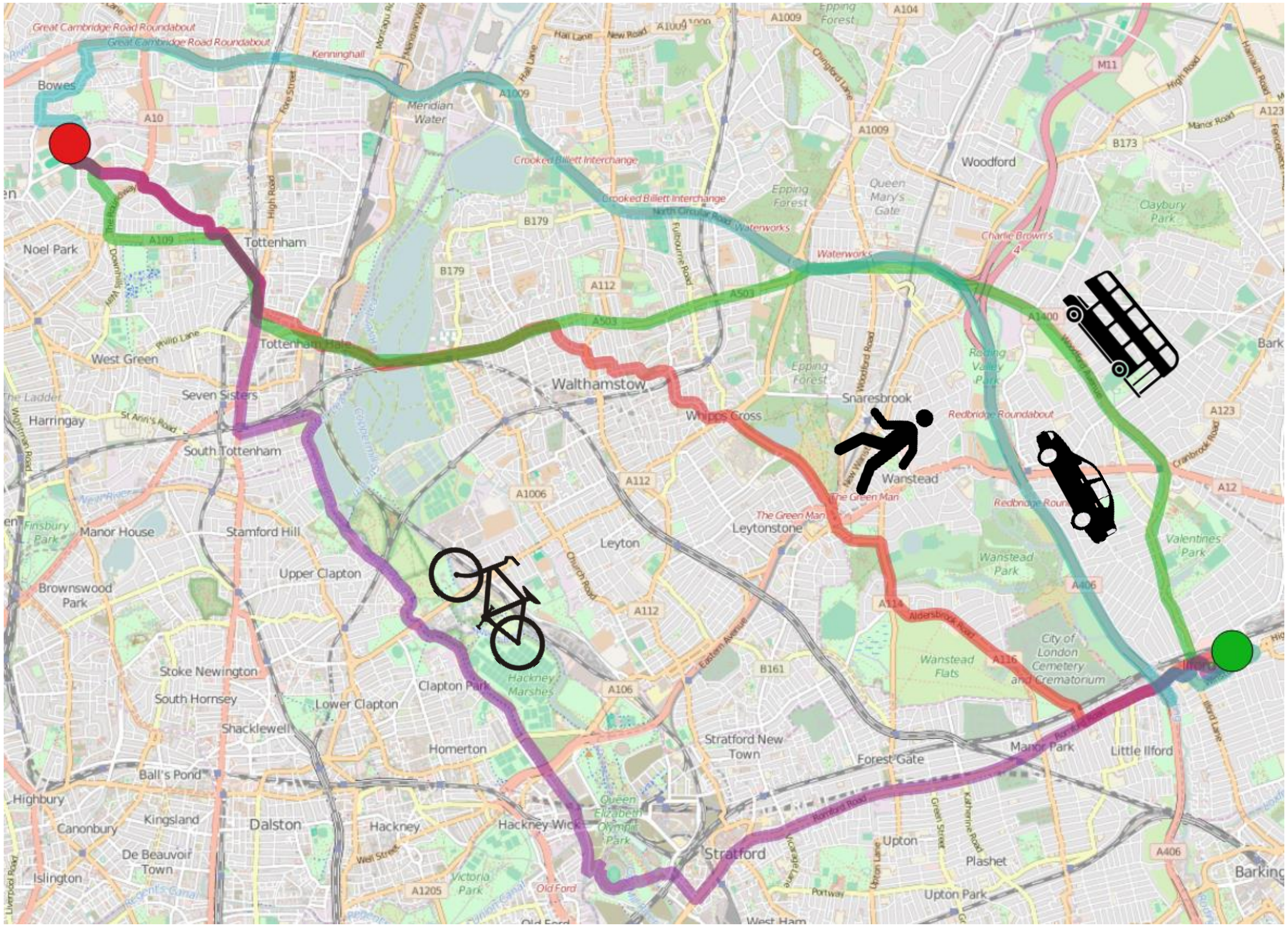
Understand how people interact with a transport network

# Case study

Transport for London



# Mode choice



Bowes
Noel Park
Tottenham
Tottenham Hale
West Green
Seven Sisters
Harringay
South Tottenham
Manor House
Stamford Hill
Upper Clapton
Brownswood Park
Stoke Newington
South Hornsey
Shacklewell
Lower Clapton
Clapton Park
Homerton
Highbury
Canonbury
Kingsland
Dalston
Hackney
Hackney Wick
De Beauvoir Town
Islington
Meridian Water
Walthamstow
Leyton
Whipps Cross
Snaresbrook
Wanstead
Leytonstone
Stratford New Town
Stratford
Forest Gate
Manor Park
Little Ilford
Upton
Plashet
Upton Park
Woodford
Queen Mary's Gate
Claybury Park
Valentines Park
Wanstead Park
Wanstead Flats
City of London Cemetery and Crematorium
Barking

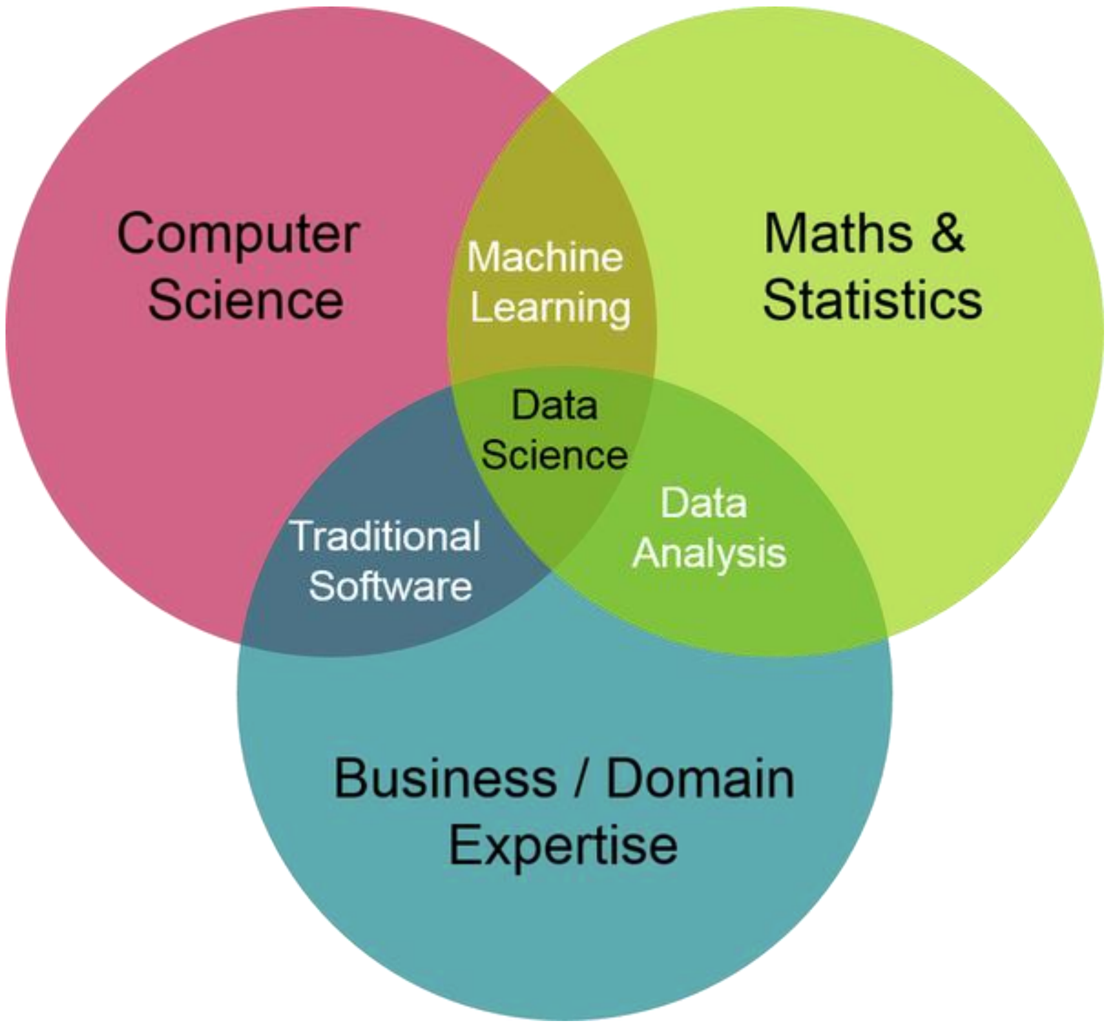
# Last week

- Introduce and use tools
- Including:
  - Python
  - Jupyter
  - Numpy
  - Matplotlib
  - Pandas

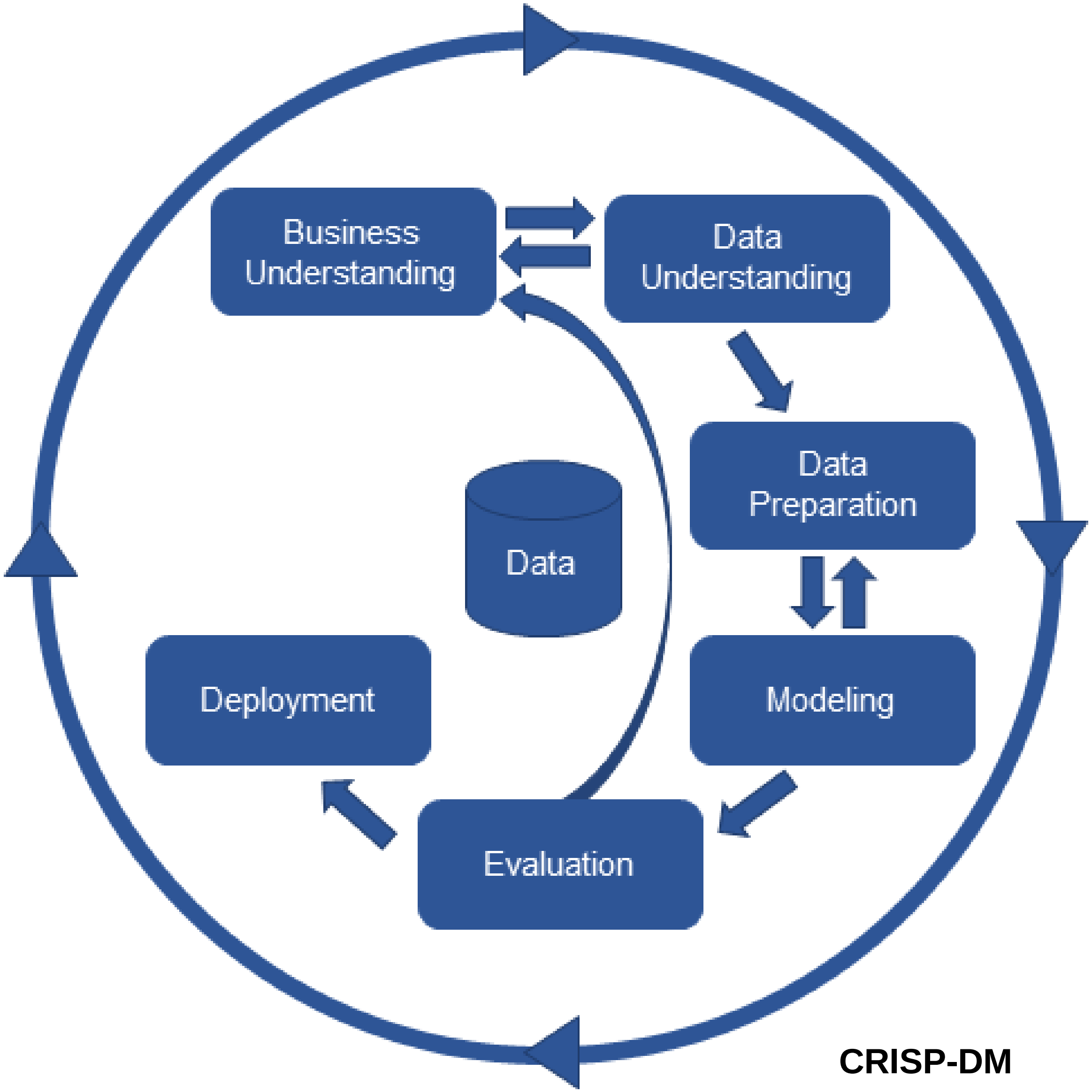
# Today

- Data science process
- Dataset
- Deterministic methods
  - K-Nearest Neighbours (KNN)
  - Decision Tree (DT)
- Discrete metrics

# Data science



# Data science




CRISP-DM

# Dataset

Visualised last week in Pandas

***Thoughts?***

Tim Hillel, Mohammed Z E B Elshafie, and Ying Jin (2018). “Recreating Passenger Mode Choice-Sets for Transport Simulation: A Case Study of London, UK”. in: *Proceedings of the Institution of Civil Engineers - Smart Infrastructure and Construction* 171.1, pp. 29–42

# Dataset building process

**Historical trip data**

London Travel Demand Survey (LTDS)

- Annual rolling **household travel survey**
- Each household member fills in **trip diary**

3 years of data (2012/13-2014/15)

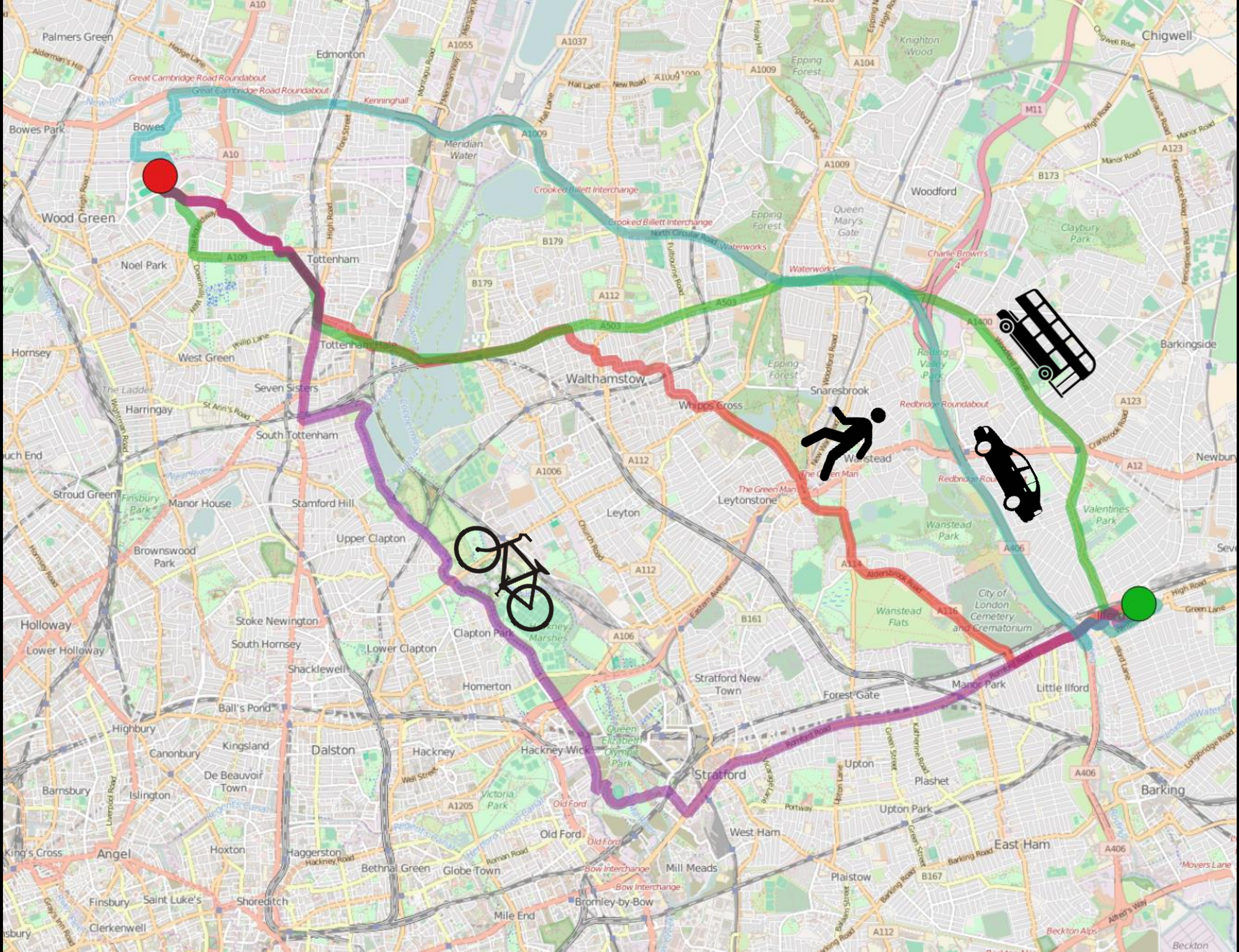
- ~130,000 trips

# Dataset building process

Historical trip data

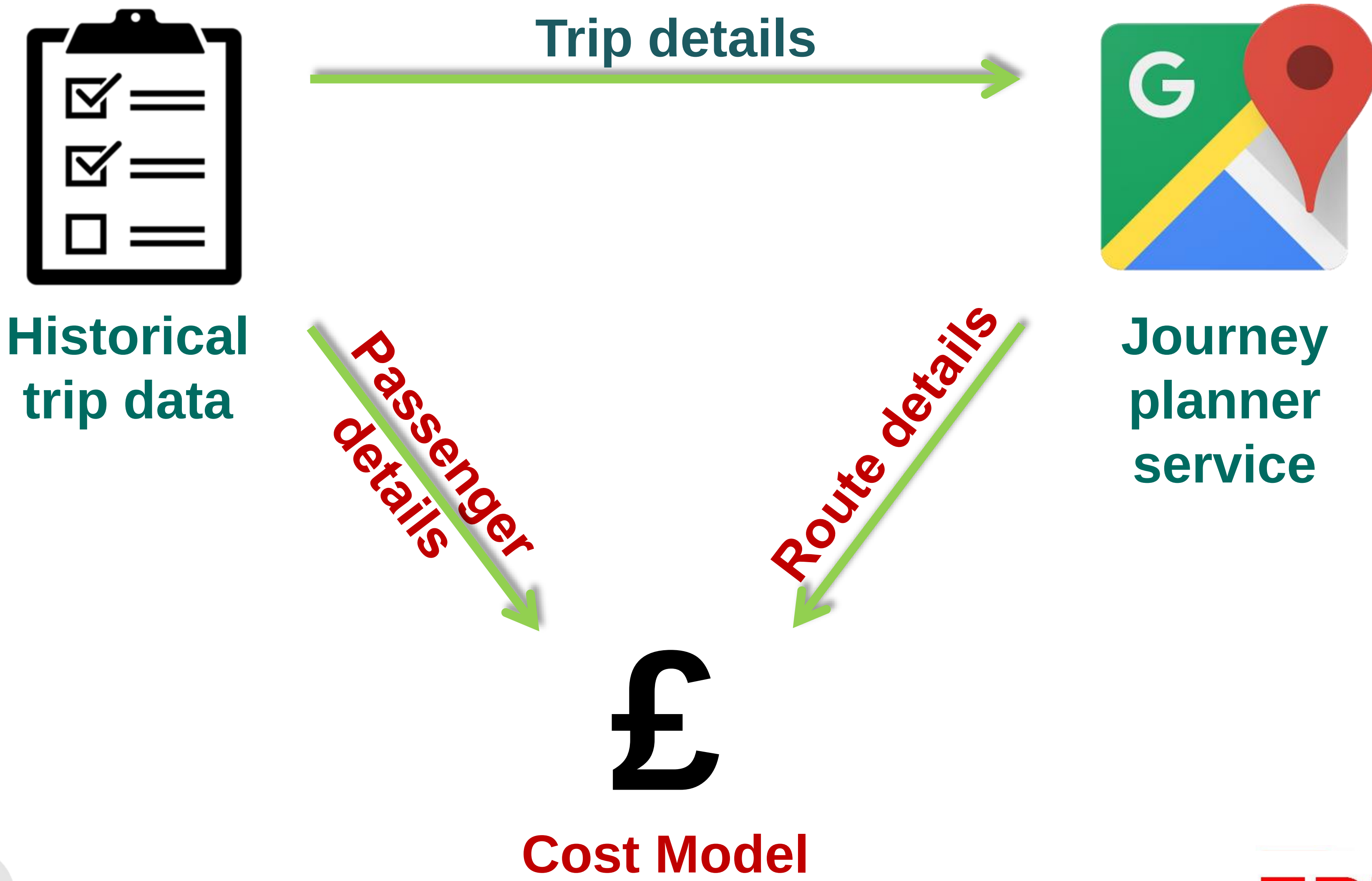
Trip details

Journey planner service



Palmers Green
Edmonton
Chigwell
Bowes Park
Bowes
Great Cambridge Road Roundabout
Kenninghall
Meridian Water
Epping Forest
Knighton Wood
Woodford
Wood Green
Noel Park
Tottenham
Crooked Billett Interchange
North Circular Road
Waterworks
Queen Mary's Gate
Claybury Park
Charlie Brown's
Hornsey
West Green
Tottenham Hale
Walthamstow
Snaresbrook
Barkingside
Seven Sisters
Harringay
The Ladder
South Tottenham
Whipps Cross
Redbridge Roundabout
Wanstead
The Green Man
Newbury
Stroud Green
Manor House
Stamford Hill
Leytonstone
Leyton
Wanstead Park
Valentines Park
Finsbury Park
Upper Clapton
Brownswood Park
City of London Cemetery and Crematorium
Wanstead Flats
Holloway
Stoke Newington
Clapton Park
Hackney Marshes
Lower Holloway
South Hornsey
Lower Clapton
Ilford
Shacklewell
Homerton
Stratford New Town
Forest Gate
Manor Park
Little Ilford
Ball's Pond
Highbury
Canonbury
Kingsland
Dalston
Hackney
Hackney Wick
Queen Elizabeth Olympic Park
Stratford
Upton
Plashet
Barking
Barnsbury
Islington
De Beauvoir Town
Victoria Park
Old Ford
Upton Park
West Ham
King's Cross
Angel
Hoxton
Haggerston
Bethnal Green
Globe Town
East Ham
Mill Meads
Plaistow
Bow Interchange
Finsbury
Saint Luke's
Shoreditch
Bromley-by-Bow
Mile End
Clerkenwell
Beckton

# Dataset building process



# Dataset

- 82,350 trips and alternatives for each major mode – walking, cycling, public transport (combined), rail, bus, driving:
  - Path
  - Duration
  - Traffic variability
  - Journey purpose
  - Socio-economic class
  - Departure time

# Dataset

## Feature vector → Class label

- Start time
- Journey purpose
- Vehicle ownership
- Fare type
- Alternative specific constants:
  - Duration
  - Cost
  - Typical traffic
  - Congestion charge

- Mode taken!

# Machine learning

- Supervised learning
- Unsupervised learning
- Reinforcement learning
- …and more:
  - Semi-supervised learning
  - Generative models
  - Active learning etc.

# Supervised learning

- Regression – **Continuous value**
- Classification - **Discrete class**

# Supervised classification

- Predict, for a feature vector $x$, the class label $y$

- Start with **binary** case
  - Was a trip made by car or not?

- **Discrete** classification
  - Predict 1 or 0

# General approach

**Fit** model on some data (**train set**):

- $N$ instances (rows), comprising of:
  - Features $x$ (columns)
  - Labels $y$

- Predict for **unseen** data:
  - $y$ (unknown) from $x$ (known)

# General approach

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$$

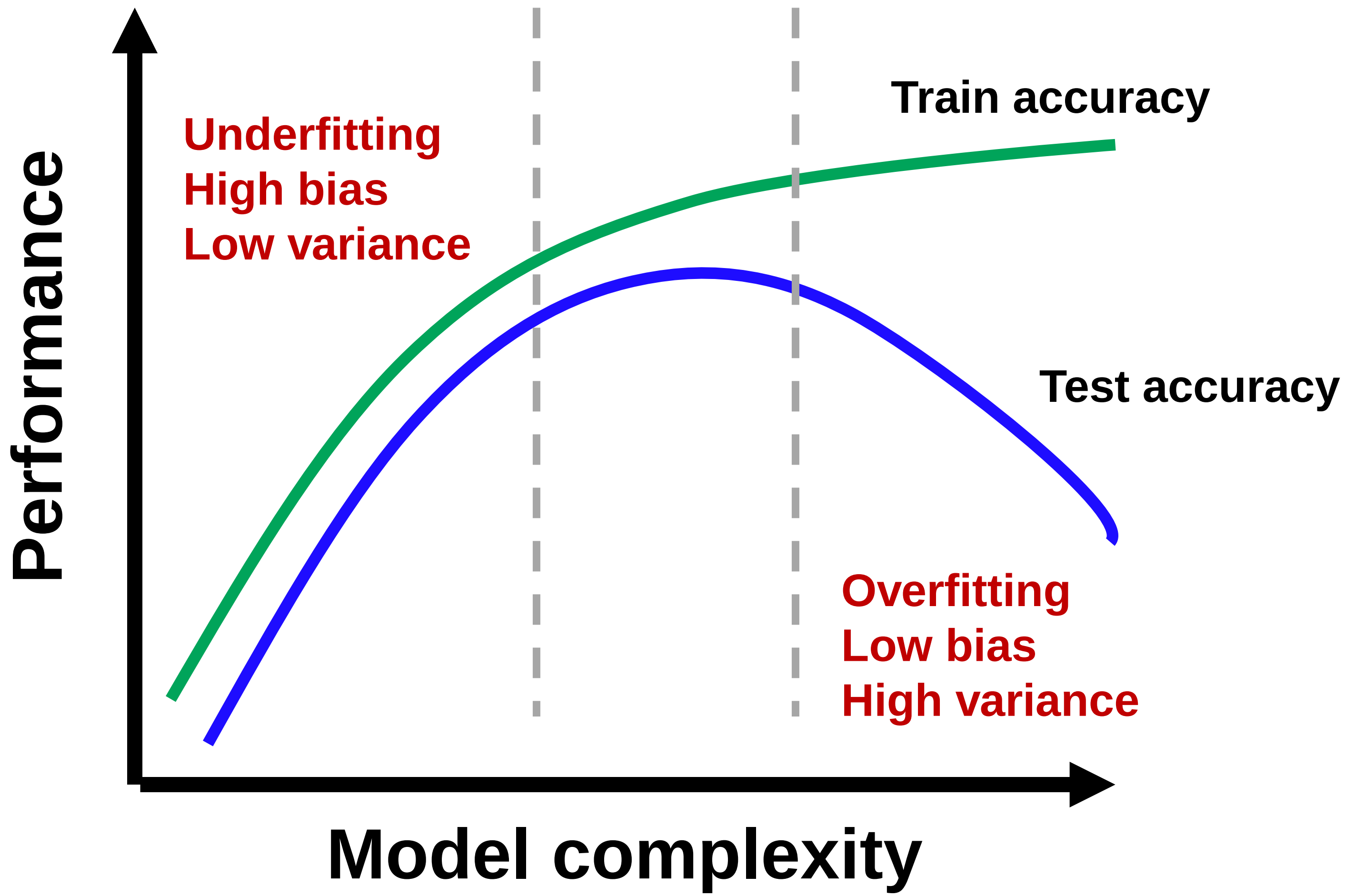
$$\boldsymbol{y} = \boldsymbol{f}'(\boldsymbol{x})$$

# Model evaluation

How do we predict how model will **perform** on unseen data?

Evaluate the model on an unseen **test set**!

# Bias-variance trade-off



# Train-test sampling

How do we define the **train** and **test** sets?

For now, we will use **random sampling** with 80:20 train:test split

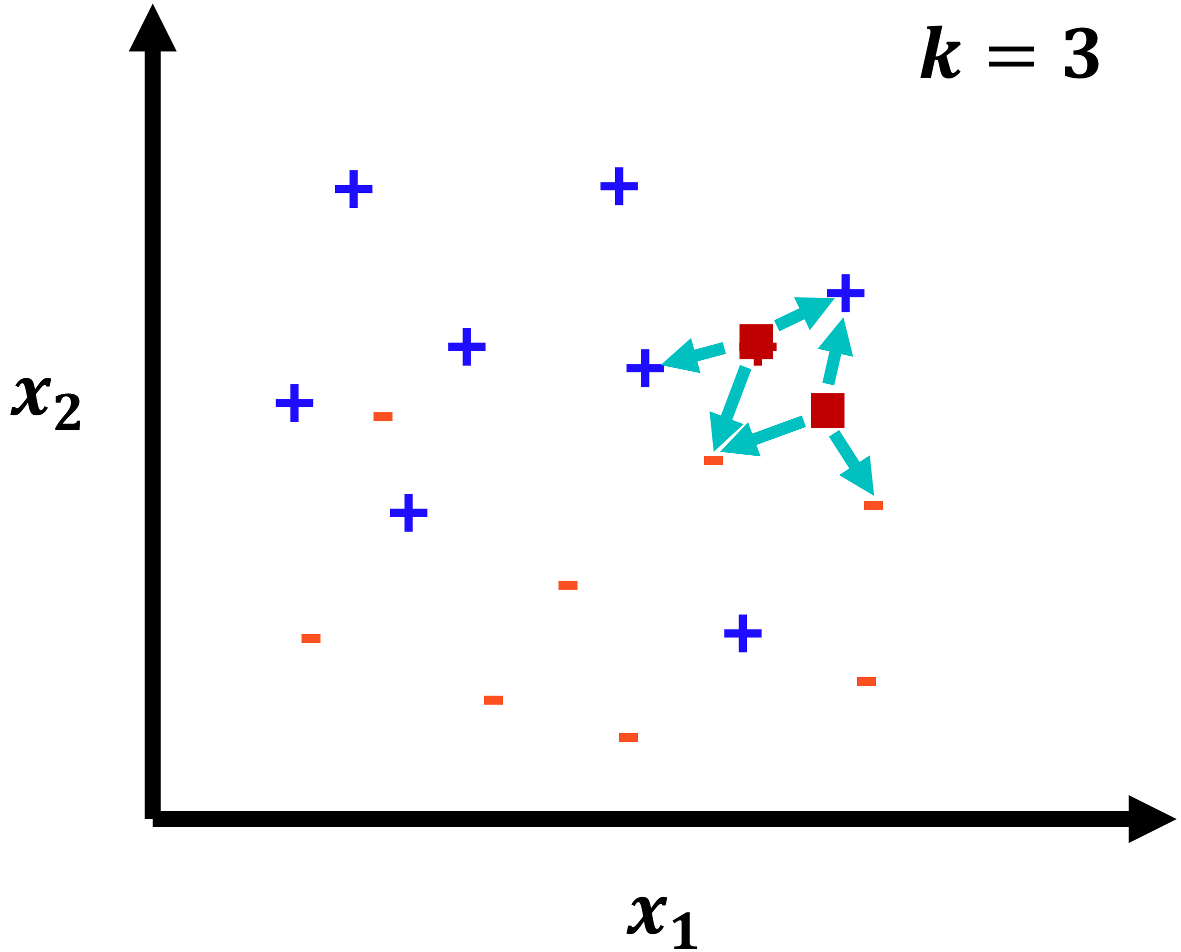
# Metrics

How do we assess model performance?

For now, we consider **classification accuracy:**

What **proportion** did we get **right**?

# K-Nearest Neighbours




*The* ***data*** *is our* ***model****!*

# In words…

1. Compute distance between the **candidate** point and all other neighbours in the **train** set:
   – For now, Euclidean distance:

$$d(p, q) = \sqrt{\sum_i (p_i - q_i)^2}$$

2. Select the $k$-nearest neighbours
3. Determine the candidate class from the $k$-nearest neighbours
   – For now, take the majority vote!

# Notebook 1: Implementing k-NN

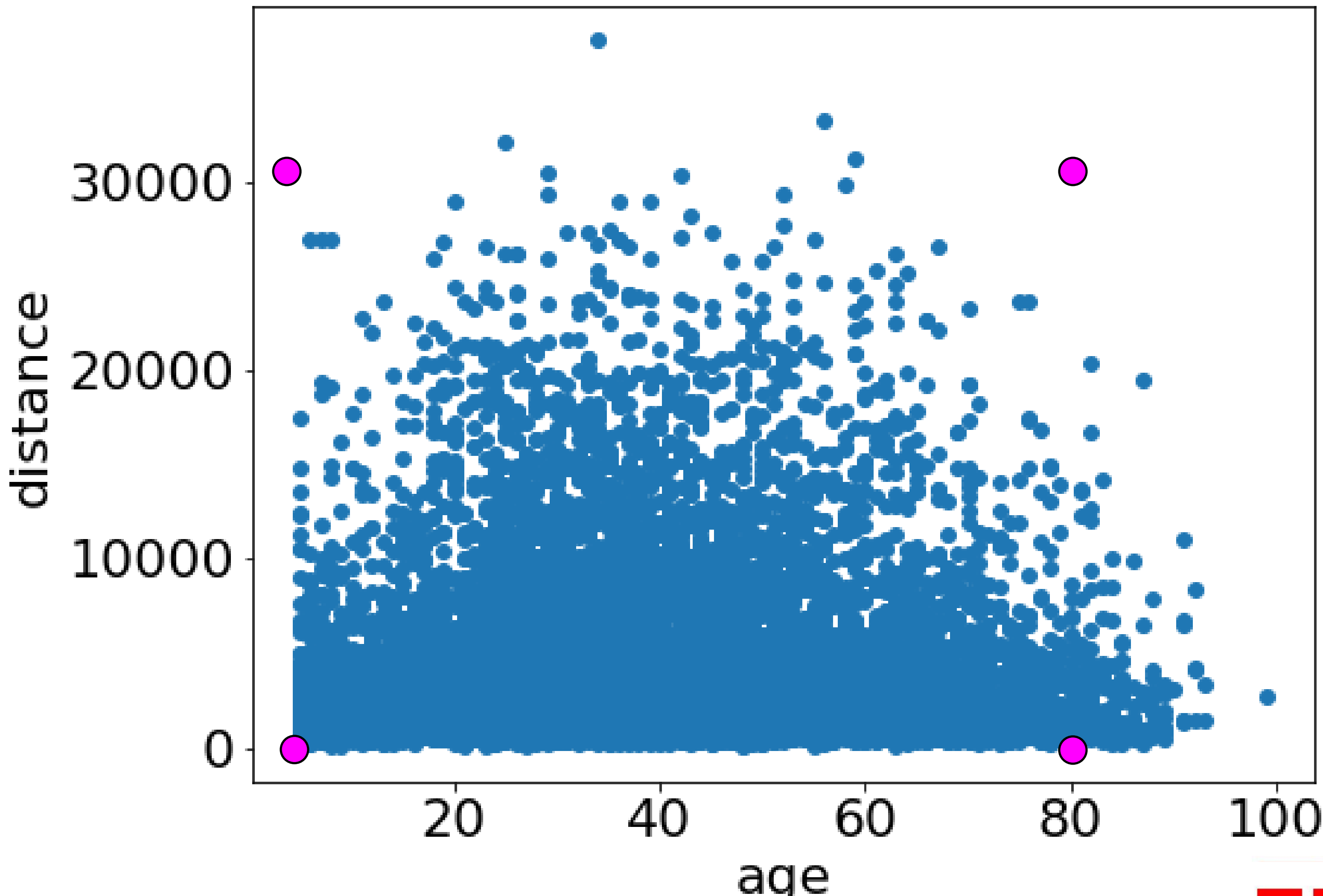
# Areas for improvement

*Ideas?*

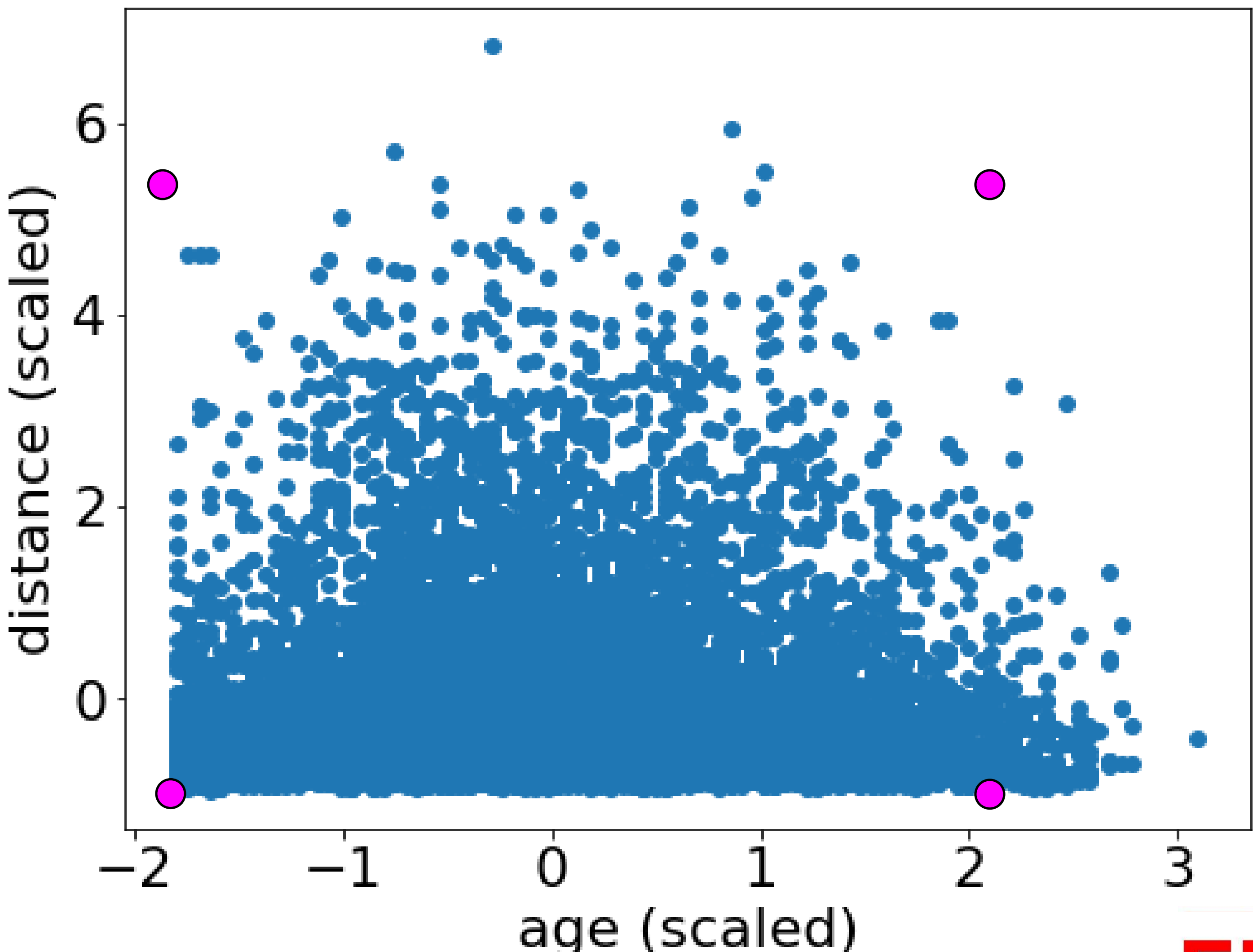
# Areas for improvement

- Data preparation
  - Scaling the data
- Model optimisation
  - Choice of $k$
  - Distance metric
  - Improvements on majority vote
- Model evaluation
  - Alternatives to accuracy

# Data scaling



# Data scaling




TRANSP-OR

EPFL

# Model optimisation

- Introduced three model *hyperparameters*
  - $\boldsymbol{k}$
  - Distance metric
  - Decision rule

- All of these can be modified to improve *out-of-sample* performance
  - $\boldsymbol{k} = 1,2,3, \dots, 100$ ? Bigger?
  - Manhattan distance? Minkowski distance?
  - Distance based weighting?

TRANSP-OR

EPFL

# Model optimisation

How to select model **hyperparameters**?

**Evaluate** different values on **unseen data**!

# Model evaluation

Is **accuracy** the best policy?

Consider the **confusion matrix**

# Confusion Matrix

| | PREDICTED CLASS | | |
|---|---|---|---|
| ACTUAL CLASS | | 1 | 0 |
| | 1 | TP | FN |
| | 0 | FP | TN |

**Accuracy = (TP+TN) / (TP+TN+FP+FN)**

# Confusion Matrix

| | PREDICTED CLASS | | |
|---|---|---|---|
| ACTUAL CLASS | | 1 | 0 |
| | 1 | 990 | 0 |
| | 0 | 10 | 0 |

**Accuracy = 99% !**

**Known as *accuracy paradox***

# Precision and recall

| | | | |
|---|---|---|---|
| | PREDICTED CLASS | | |
| ACTUAL CLASS | | 1 | 0 |
| | 1 | TP | FN |
| | 0 | FP | TN |

**Precision (+ve) = TP / TP+FP**

**Recall (+ve) = TP / TP+FN**

# Precision and recall

| | PREDICTED CLASS | | |
|---|---|---|---|
| ACTUAL CLASS | | 1 | 0 |
| | 1 | 990 | 0 |
| | 0 | 10 | 0 |

**Precision (+ve) = 0.99%**

**Recall (+ve) = 100%**

# Precision and recall

| | PREDICTED CLASS | | |
|---|---|---|---|
| ACTUAL CLASS | | 1 | 0 |
| | 1 | TP | FN |
| | 0 | FP | TN |

**Precision (-ve) = TN / TN+FN**

**Recall (-ve) = TN / TN+FP**

# Precision and recall

| | PREDICTED CLASS | | |
|---|---|---|---|
| ACTUAL CLASS | | 1 | 0 |
| | 1 | 990 | 0 |
| | 0 | 10 | 0 |

**Precision (-ve) = 0%**

**Recall (-ve) = 0%**

# Multinomial case - Precision

| | PREDICTED CLASS | | | | | |
|---|---|---|---|---|---|---|
| ACTUAL CLASS | | 0 | 1 | 2 | 3 | … |
| | 0 | | 10 | | | |
| | 1 | 20 | 100 | 40 | 10 | 30 |
| | 2 | | 5 | | | |
| | 3 | | 15 | | | |
| | … | | 20 | | | |

**Precision = TP / TP+FP**

**= 100/150**

**= 67%**

TRANSP-OR

EPFL

# Multinomial case - Recall

| | | PREDICTED CLASS | | | | |
|---|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 | … |
| ACTUAL CLASS | 0 | | 10 | | | |
| | 1 | 20 | 100 | 40 | 10 | 30 |
| | 2 | | 5 | | | |
| | 3 | | 15 | | | |
| | … | | 20 | | | |

**Recall = TP / TP+FP**

**= 100/200**

**= 50%**

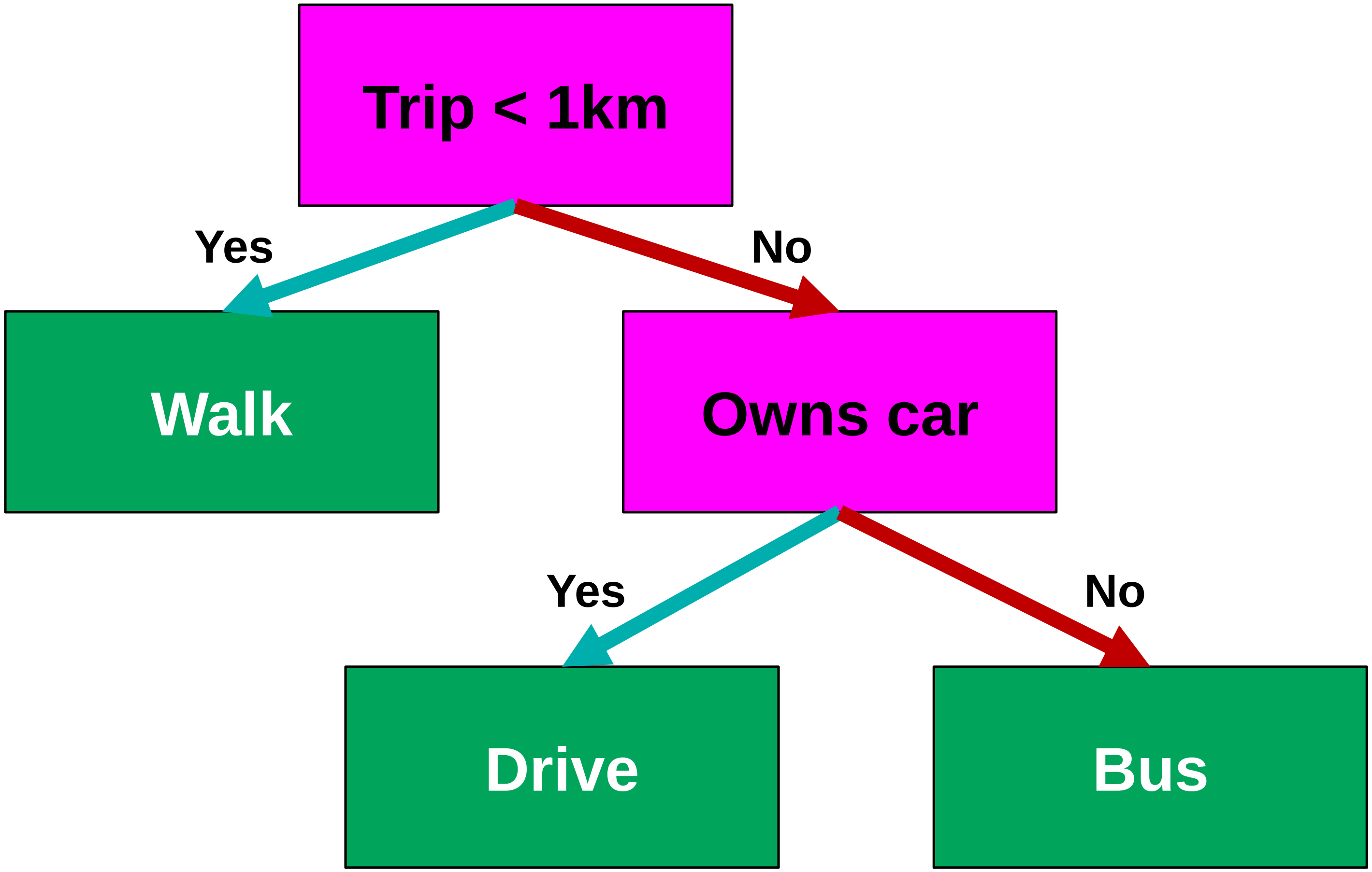
# Machine learning in Python

scikit learn

# Lab

## Notebook 2: k-NN in scikit-learn

# Decision trees




TRANSP-OR

EPFL

# Decision trees

- Recursive hierarchical structure of *binary* splits

To fit (each split):

- For **every feature:**
  - Sort the data over the feature
  - For **every possible split point**:
    - Calculate the **gain** in seperation
- Choose the split with the highest **gain**
- Repeat

# Separation

- Separation is the reduction in the *shuffeledness* of the data

- Two common metrics
  - GINI impurity
  - Entropy

# GINI Impurity

$$G(p) = \sum_{i=1}^{J} p_i(1 - p_i) = 1 - \sum_{i=1}^{J} {p_i}^2$$

where $p_i$ is the **proportion** of class $i$

# Entropy

$$H(p) = -\sum_{i=1}^{J} p_i \log_2 p_i$$

where $p_i$ is the **proportion** of class $i$

# When to stop splitting

- Maximum depth
- Minimum leaf node size
- Minimum split size
- Minimum gain from split
- Maximum number of leaf nodes
- Etc…

Lots of **hyperparameters!**

# Homework assignment

## Notebook 3: Decision trees