COMP2200/COMP6200 — Week 5
k-Nearest Neighbours & Model Evaluation

Greg Baker

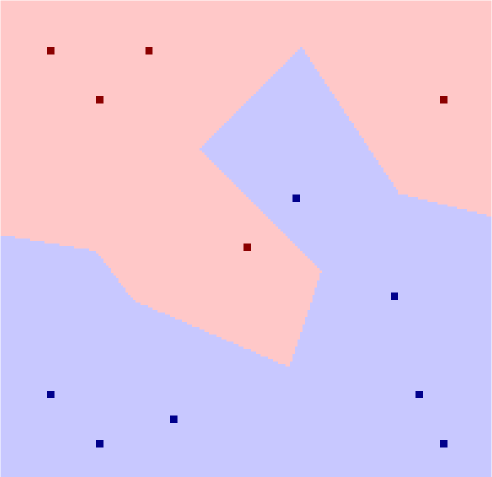
25th August 2025

Agenda

- 1-NN and k-NN
- Predicting NSW land value
- Red Rooster line: geospatial classification with k-NN
- Choosing an algorithm
- Metrics that actually mean something (not just accuracy)

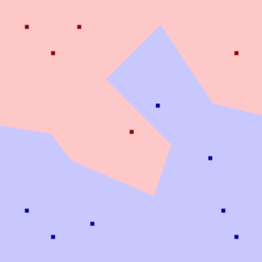
1-NN

- Classify a point by the label of its nearest neighbour
- Decision boundary forms a Voronoi diagram

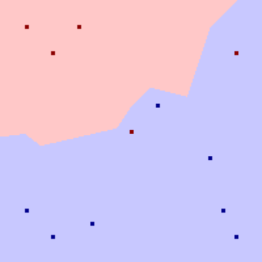


k-NN and the hyperparameter k

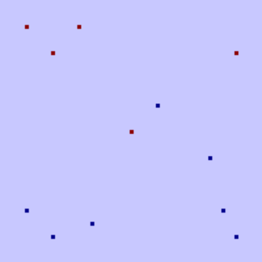
- Consider the majority label among the k closest points
- Larger k smooths the decision boundary; k is a hyperparameter



$k = 1$



$k = 3$



$k = 10$

k-NN in 90 seconds

- Keep the training data around.
- To classify a new point: find its k nearest neighbours; they vote.
- To regress: average the k neighbours' targets (often distance-weighted).

Dials to turn

- k: 1 (wiggly) $\rightarrow$ 15/31 (smooth)
- Weights: uniform vs distance
- Distance: Euclidean vs Manhattan
- **Scaling:** must normalise features

Thinking question about 1-NN

If you test on your training data with 1-NN, you get 100% accuracy. Why?

NSW Land Prices

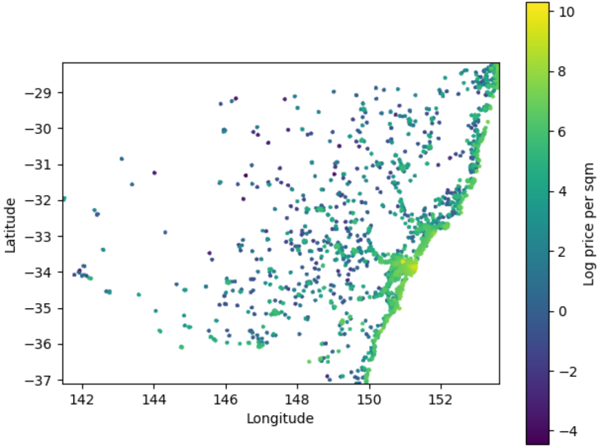
- Every year, the valuer general estimates how much the *unimproved land value* of each property in the state. (Georgist taxation)

  Unimproved = what it would be worth if there was no building on it

- Since we also know how big the property is, we can calculate the price per square metre of land
- Your land is probably worth about the same as your neighbours
  - When would this not be true?
  - (Harder) How could we handle this?
- So let's predict the land value of some bushland near the university

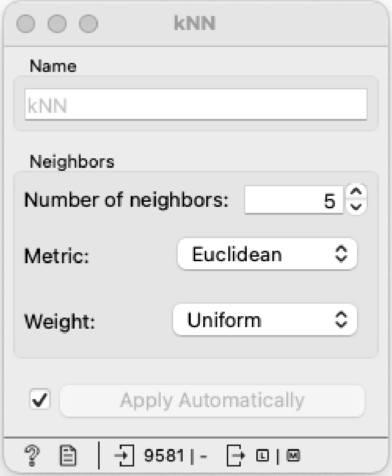
Sample data

`geolocated_landvalues.csv`



The land values are correct, but I did the geolocation in a hurry and might not be 100% accurate.

What we need (1/2): kNN

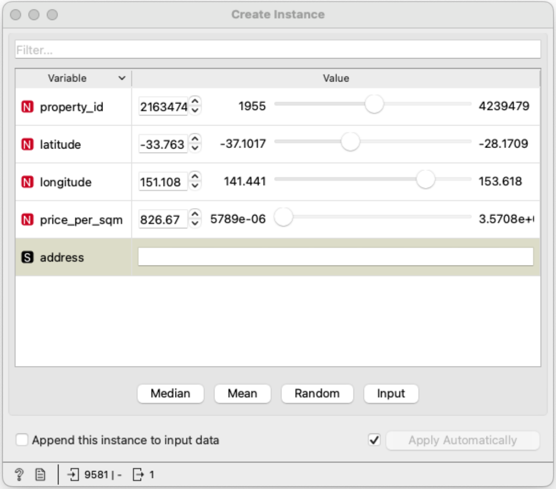


Number of neighbors
: The k in k-NN

Metric
: How do you measure closeness? $\sqrt{x^2 + y^2}$ or $|x| + |y|$?

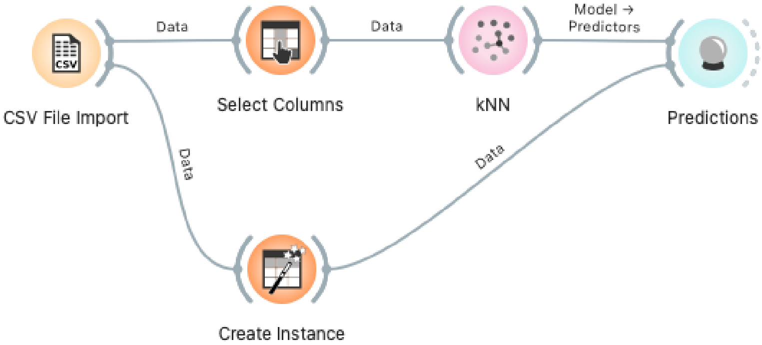
Weight
: Should a closer point count more towards the answer than a point further away?

What we need (2/2): Create a new instance



We only care about setting latitude and longitude.
We want to predict this one only, so we're turning off "Append this instance to input data"

Workflow


Data
Data
Model →
Predictors
CSV File Import
Select Columns
kNN
Predictions
Data
Data
Create Instance

Should we believe that number?

- How accurate was it?
- Should we have used kNN or should we use something else?
- We need to make predictions about something we already know the value of (Ben's week 2 and 3 lectures)

Should we believe that number?

- How accurate was it?
- Should we have used kNN or should we use something else?
- We need to make predictions about something we already know the value of (Ben's week 2 and 3 lectures)
- Hold that thought, and we'll come back to it

Train–test split in Orange

- Use **Data Sampler** to hold out test data (e.g. 80/20)
- Train on one output, make predictions on the other and see if they are right

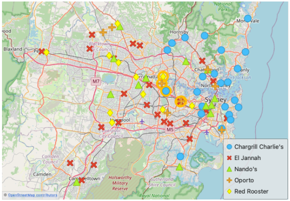
Classification algorithms we know

If a <u>line</u> or plane divides the data, try **logistic regression**

If <u>similar regions</u> exist <u>and extrapolation isn't needed</u>, try **k-NN**

The Red Rooster line – it's not obvious which to use

- Red Rooster is an Australian roast chicken chain
- Urban myth: almost no stores appear east of a line through Sydney
- Other Urban myth: Chargrill Charlie's doesn't appear west of that line
- Data: locations of fast-food chains across NSW
- Goal: predict whether a store is Red Rooster or Chargrill Charlie's based on location
- Illustrates classification and geographic bias



What sort of algorithm do we need?

- Is the task supervised or unsupervised?
- Classification or regression?
- Predict whether a chicken store is Red Rooster

What sort of algorithm do we need?

- Is the task supervised or unsupervised?
- **Supervised**
- Classification or regression?

- Predict whether a chicken store is Red Rooster

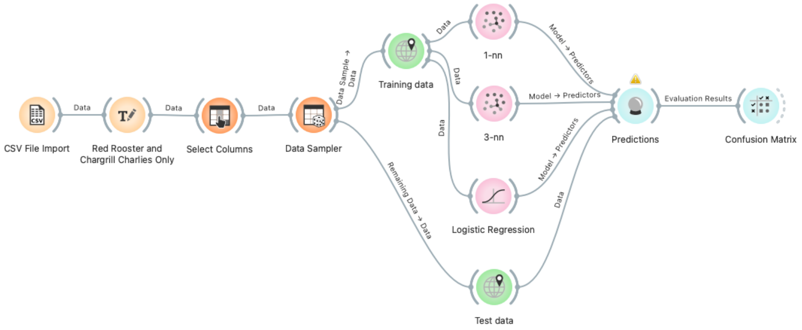
What sort of algorithm do we need?

- Is the task supervised or unsupervised?
- **Supervised**
- Classification or regression?
- **Classification**
- Predict whether a chicken store is Red Rooster

Modelling setup

- Features: latitude and longitude of each store
- Class label: `chain` (Red Rooster vs others)
- Compare logistic regression and k-NN

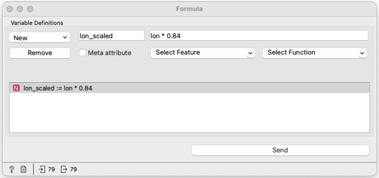
Our first comparison



(If you don't have the Geospatial extension, just skip the geomaps)

Problem 1: A geospatial gotcha (and a cheap fix)

- Degrees aren't metres, particularly near the south or north pole.
- At Sydney's latitude, 1° lon $\approx 0.84 \times 1^\circ$ lat.
- **Fix in Orange:** add `lon_scaled = lon * 0.84` (Formula)
- Better would be project into 3D coordinates (or based on travel distances)



Problem 2: Inconsistent answers

We get different answers each time we resend from the Data Sampler

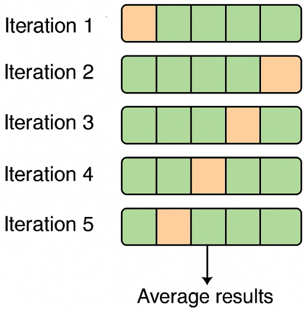
We need to run a train and test split many times to know the spread of answers

Why cross-validation?

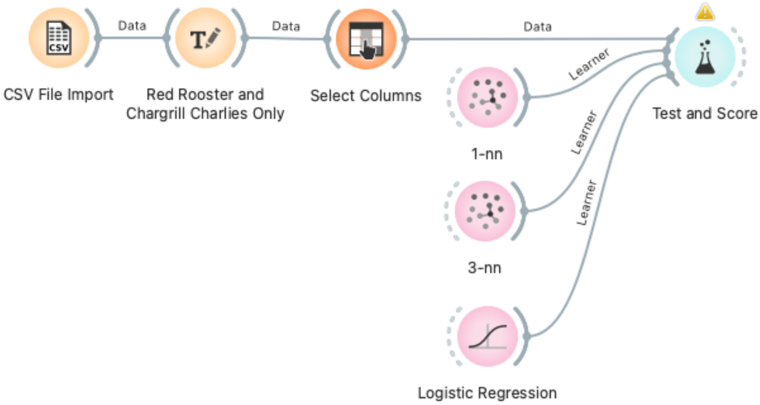
- A single test split can give a noisy estimate
- Cross-validation averages performance across folds
- Uses data more efficiently

How cross-validation works

1. Split data into k roughly equal folds
2. For each fold, train on $k-1$ parts and validate on the remaining part
3. Average the scores across folds



Cross-validation is much simpler too!



Cross-validation

- **Test & Score** widget runs *k*-fold cross-validation on the training data
- Use validation results to select between logistic regression and k-NN
- Hold out a final test set for unbiased accuracy estimates

The Garden of Forking Paths

- *El jardín de senderos que se bifurcan* explores many alternatives
- We can try many analyses; some appear better by coincidence
- Guard against this by holding out data

Three levels

Training data Usually around 80%. Set model parameters

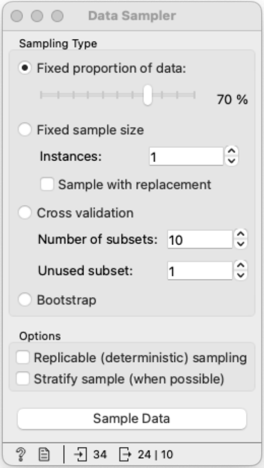
Validation data Around 10%. Choose models and hyperparameters

Test data Final 10%. Report unbiased accuracy

Training + Validation data is often cross validated.
((Training + Validation) + Test Data) is often cross validated

Train–validation–test in Orange

- First split off test data with **Data Sampler**
- Split the remainder again or use cross-validation for validation
- Tune models before a single final test evaluation



Results in detail

Predictions

Show probabilities for: Classes in data
☑ Show classification errors
Restore Original Order

	1-nn	error	3-nn	error	Logistic Regression	error	chain
1	1.00 : 0.00 : 0.00 : 0.00 : 0.00 → Chargrill Charlie's	1.000	0.33 : 0.00 : 0.00 : 0.00 : 0.67 → Red Rooster	0.333	0.44 : 0.00 : 0.00 : 0.00 : 0.56 → Red Rooster	0.443	Red Rooster
2	1.00 : 0.00 : 0.00 : 0.00 : 0.00 → Chargrill Charlie's	0.000	1.00 : 0.00 : 0.00 : 0.00 : 0.00 → Chargrill Charlie's	0.000	1.00 : 0.00 : 0.00 : 0.00 : 0.00 → Chargrill Charlie's	0.002	Chargrill Cha...
3	1.00 : 0.00 : 0.00 : 0.00 : 0.00 → Chargrill Charlie's	0.000	1.00 : 0.00 : 0.00 : 0.00 : 0.00 → Chargrill Charlie's	0.000	1.00 : 0.00 : 0.00 : 0.00 : 0.00 → Chargrill Charlie's	0.002	Chargrill Cha...
4	0.00 : 0.00 : 0.00 : 0.00 : 1.00 → Red Rooster	0.000	0.00 : 0.00 : 0.00 : 0.00 : 1.00 → Red Rooster	0.000	0.00 : 0.00 : 0.00 : 0.00 : 1.00 → Red Rooster	0.000	Red Rooster
5	1.00 : 0.00 : 0.00 : 0.00 : 0.00 → Chargrill Charlie's	0.000	1.00 : 0.00 : 0.00 : 0.00 : 0.00 → Chargrill Charlie's	0.000	0.99 : 0.00 : 0.00 : 0.00 : 0.01 → Chargrill Charlie's	0.006	Chargrill Cha...
6	1.00 : 0.00 : 0.00 : 0.00 : 0.00 → Chargrill Charlie's	0.000	1.00 : 0.00 : 0.00 : 0.00 : 0.00 → Chargrill Charlie's	0.000	0.98 : 0.00 : 0.00 : 0.00 : 0.02 → Chargrill Charlie's	0.017	Chargrill Cha...
7	1.00 : 0.00 : 0.00 : 0.00 : 0.00 → Chargrill Charlie's	0.000	1.00 : 0.00 : 0.00 : 0.00 : 0.00 → Chargrill Charlie's	0.000	1.00 : 0.00 : 0.00 : 0.00 : 0.00 → Chargrill Charlie's	0.001	Chargrill Cha...
8	1.00 : 0.00 : 0.00 : 0.00 : 0.00 → Chargrill Charlie's	0.000	1.00 : 0.00 : 0.00 : 0.00 : 0.00 → Chargrill Charlie's	0.000	1.00 : 0.00 : 0.00 : 0.00 : 0.00 → Chargrill Charlie's	0.003	Chargrill Cha...
9	1.00 : 0.00 : 0.00 : 0.00 : 0.00 → Chargrill Charlie's	0.000	1.00 : 0.00 : 0.00 : 0.00 : 0.00 → Chargrill Charlie's	0.000	0.99 : 0.00 : 0.00 : 0.00 : 0.01 → Chargrill Charlie's	0.007	Chargrill Cha...
10	1.00 : 0.00 : 0.00 : 0.00 : 0.00 → Chargrill Charlie's	1.000	0.67 : 0.00 : 0.00 : 0.00 : 0.33 → Chargrill Charlie's	0.667	0.60 : 0.00 : 0.00 : 0.00 : 0.40 → Chargrill Charlie's	0.601	Red Rooster

☑ Show perfomance scores   Target class: (Average over classes)

Model	AUC	CA	F1	Prec	Recall	MCC
1-nn	0.667	0.800	0.762	0.844	0.800	0.509
3-nn	1.000	0.900	0.893	0.912	0.900	0.764
Logistic Regression	1.000	0.900	0.893	0.912	0.900	0.764

10 | 10 | 10 | 3×10

Classification metrics

- **AUC** Area under curve: the plot of the true positive rate (TPR) against the false positive rate (FPR) at each threshold setting
- **CA** Classification Accuracy: proportion of all predictions that are correct
- **F1** : harmonic mean of precision and recall
- **Precision** : true positives over all predicted positives
- **Recall** : true positives over all actual positives
- **MCC** Matthews correlation coefficient (Pearson correlation between the true and predicted binary labels), $1 =$ perfect, $0 =$ random, –1 $=$ total disagreement.
$$\frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \in [-1, 1].$$

Reading the scoreboard (classification)

- **Accuracy** can mislead on imbalance.
- Prefer **MCC**, **F1**, **ROC AUC**.
- Use **Confusion Matrix** to see the cost of mistakes.

Reflect on results

- Which model scored higher?
- Did the outcome match your earlier prediction?
- If not, what might explain the difference?

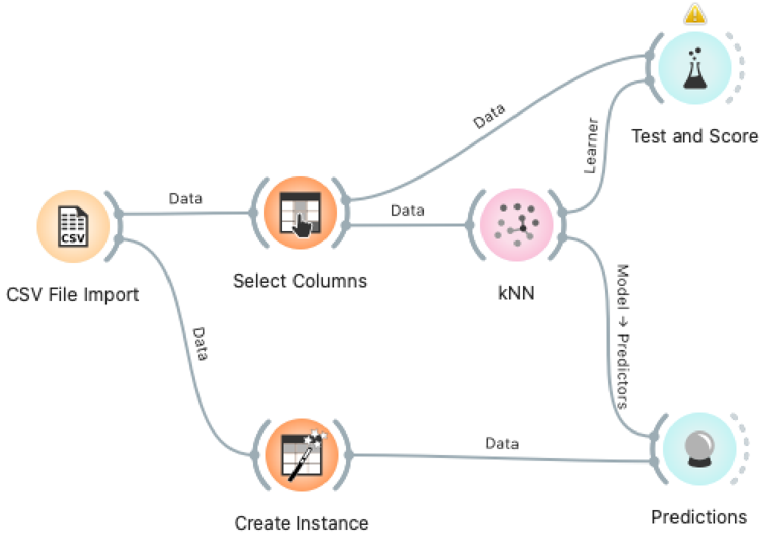
Which metric when?

- Rare positives and false negatives are costly $\Rightarrow$ **Recall** / **F1** or **PR AUC**.
- Balanced classes $\Rightarrow$ Accuracy is fine, but confirm with ROC AUC.

Hyper-parameter tuning

Each k is like a different model. We could try every k to see which one has the best metric

Orange workflow: land value regression (k-NN)



Interpreting regression metrics

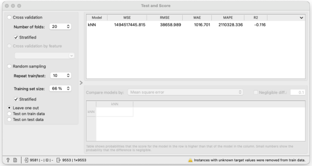
MSE punishes large misses. Take the square of the error and take the mean of that

RMSE punishes large misses. Same, but then take the square-root

MAE mean absolute error: don't bother squaring first

MAPE mean absolute percentage error

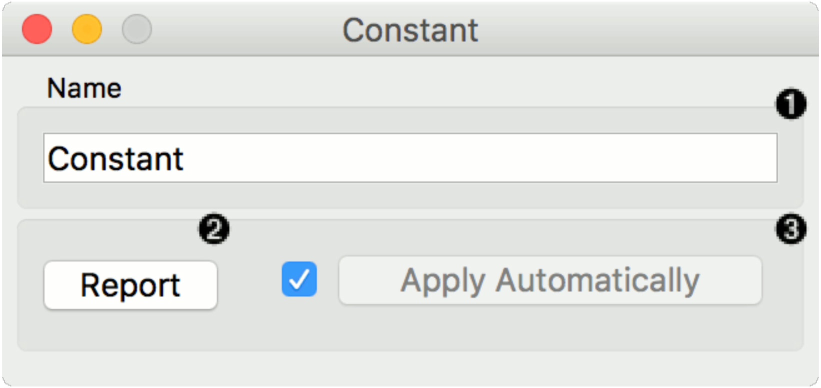
R^2 : how much of the signal in the data your model is capturing



Putting it together

- Hold out a test set for final evaluation
- Use cross-validation on the remaining data to choose k
- Example: k-NN classifier for Red Rooster prediction

Constant / dummy



- For classifiers: predict the most common
- For regressors: predict the mean

If you can't beat the dummy. . .

Common gotchas & quick checks

- Always scale/normalise features; otherwise distances are dominated by the widest-range feature.
- High dimensionality hurts kNN; reduce features or switch models if d grows.
- Class imbalance: ignore raw accuracy; use MCC/F1 and inspect the confusion matrix.
- Baselines: beat **Constant** (dummy) classifier/regressor; if you can't, your features aren't informative.
- Leakage alarms: target-derived features, near-duplicates across splits, or adding the test point to the training set.

Summary

- Use **k-NN** when nearby points should behave similarly; use **logistic regression** for roughly linear boundaries.
- **Preprocess:** scale features; for lat/lon, scale longitude by $\cos(\varphi)$ (or project).
- **Pick** k via cross-validation; try weights (uniform vs distance) and choose by task-appropriate metrics.
- **Evaluate:** prefer MCC/F1/ROC AUC over raw accuracy; for regression, report MAE/RMSE and R^2; avoid naive MAPE.
- **Guard rails:** hold out a test set; compare to a **Constant** baseline; avoid leakage; beware spatial splits.