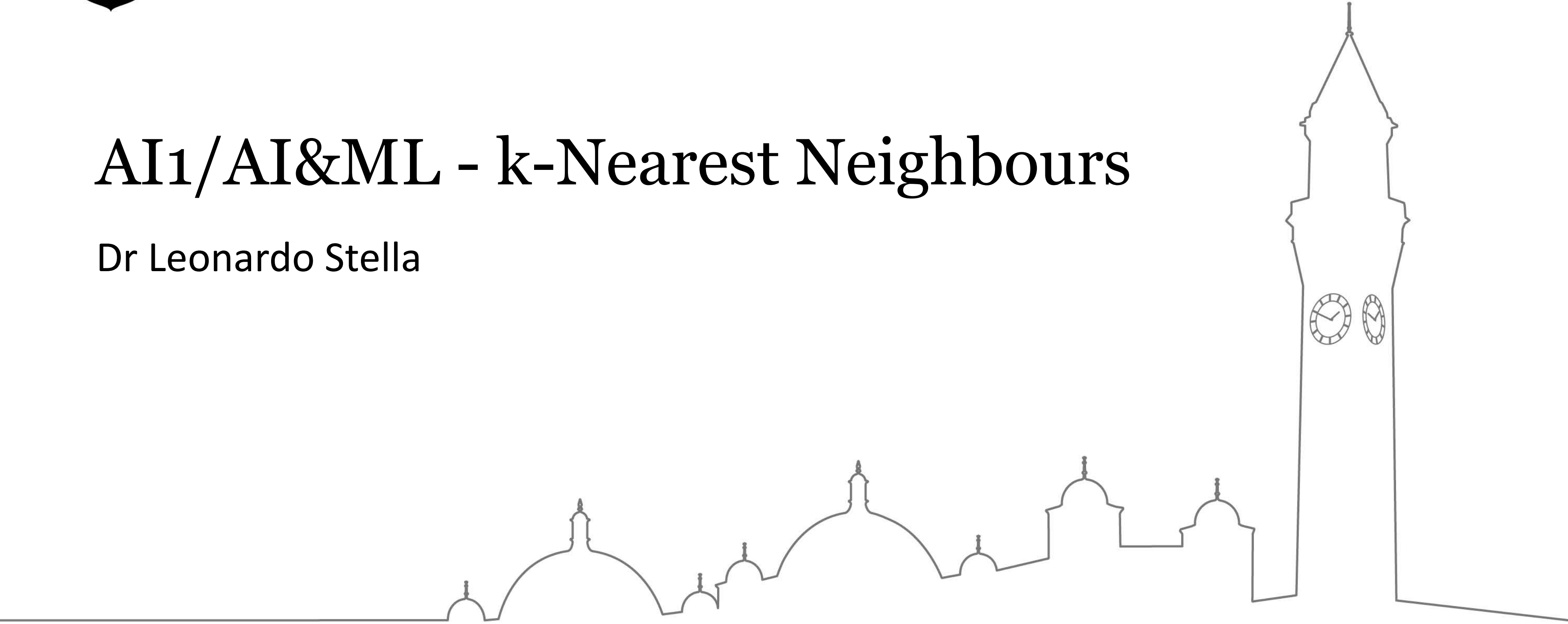
UNIVERSITY OF BIRMINGHAM

AI1/AI&ML - k-Nearest Neighbours

Dr Leonardo Stella



Aims of the Session

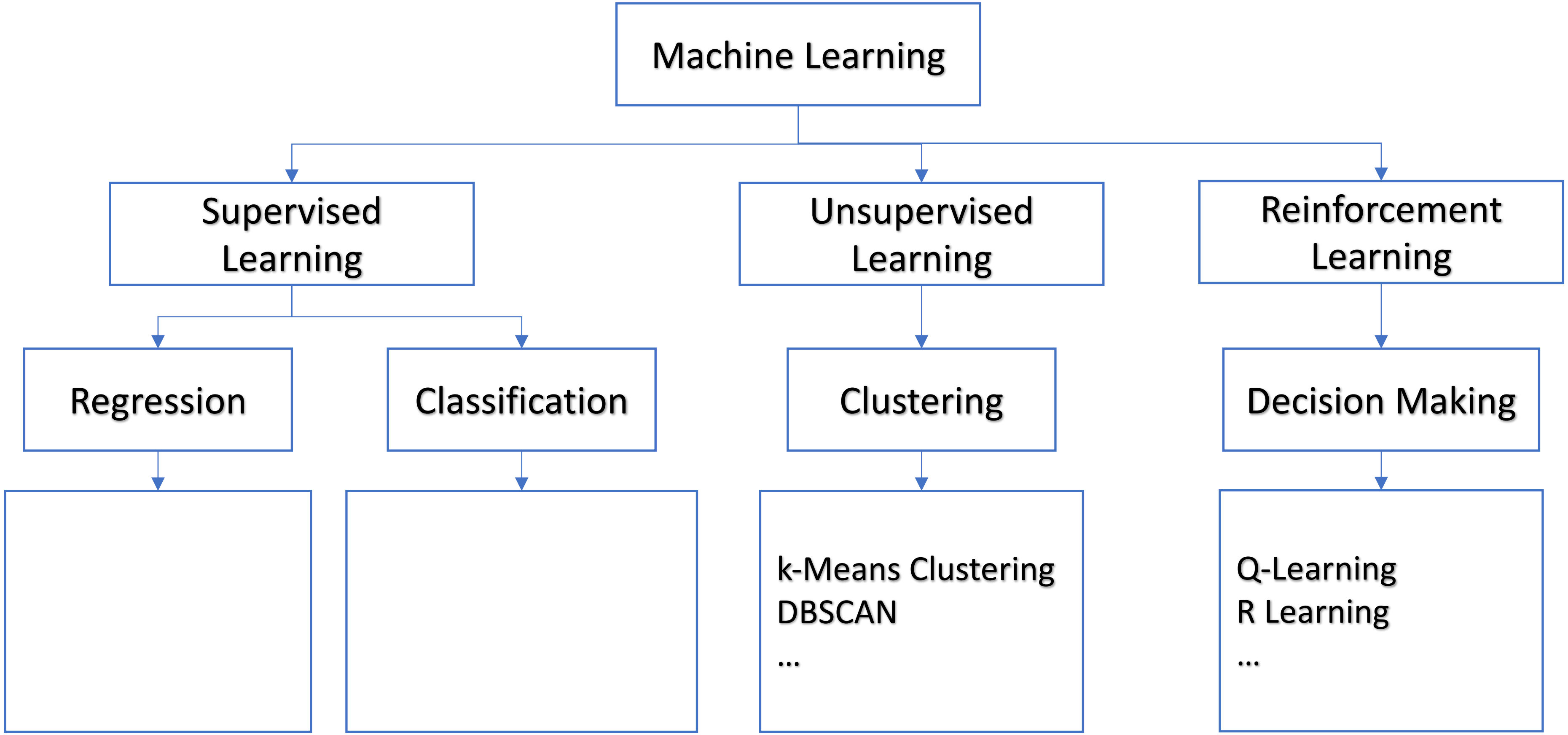
This session aims to help you:

- Explain the steps of k-Nearest Neighbours
- Apply k-Nearest Neighbours to problems involving numeric, ordinal and categorical input attributes
- Use evaluation procedures to estimate the performance of a predictor

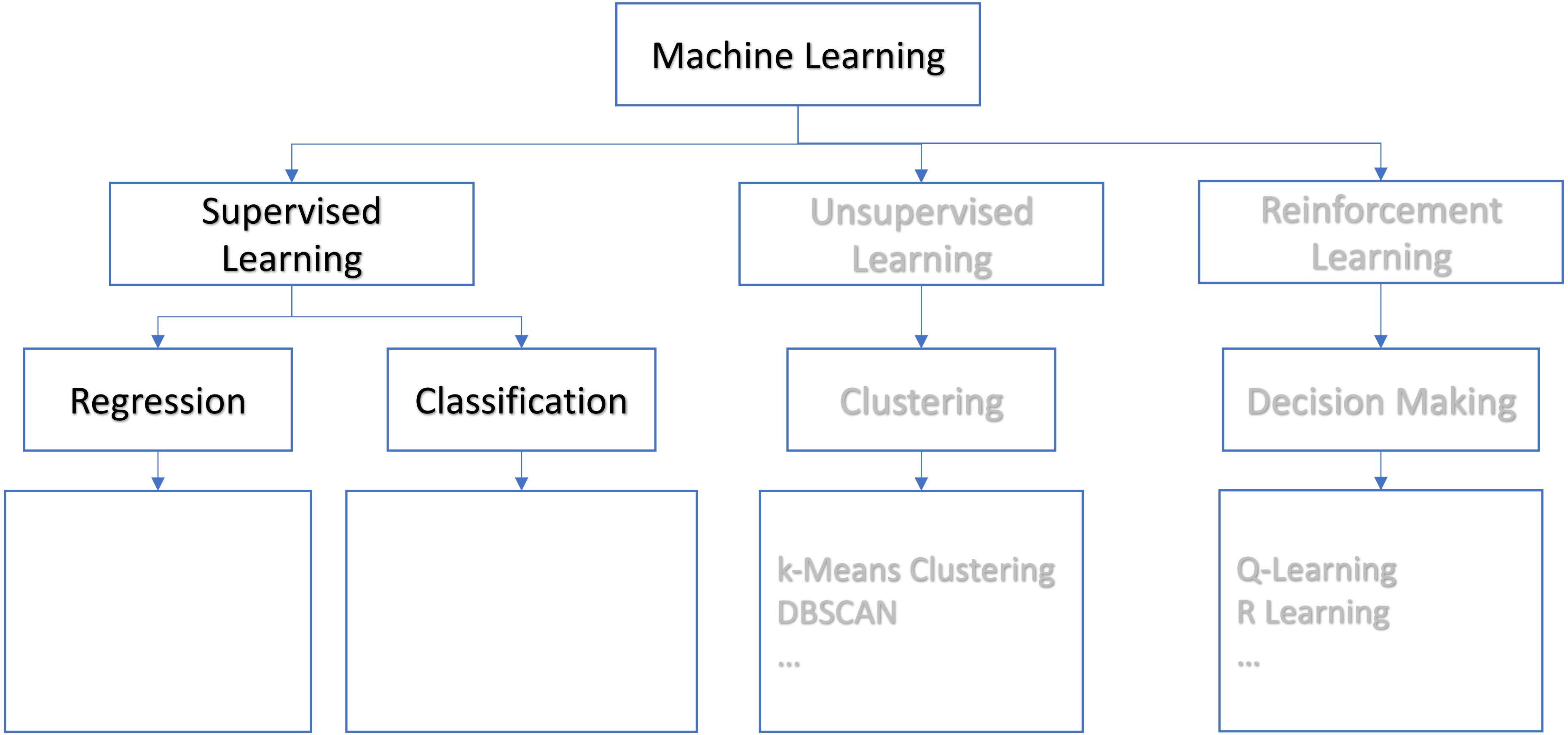
Overview

- **Introduction**
- k-Nearest Neighbours
- k-NN algorithm and pros/cons
- Evaluation Procedures

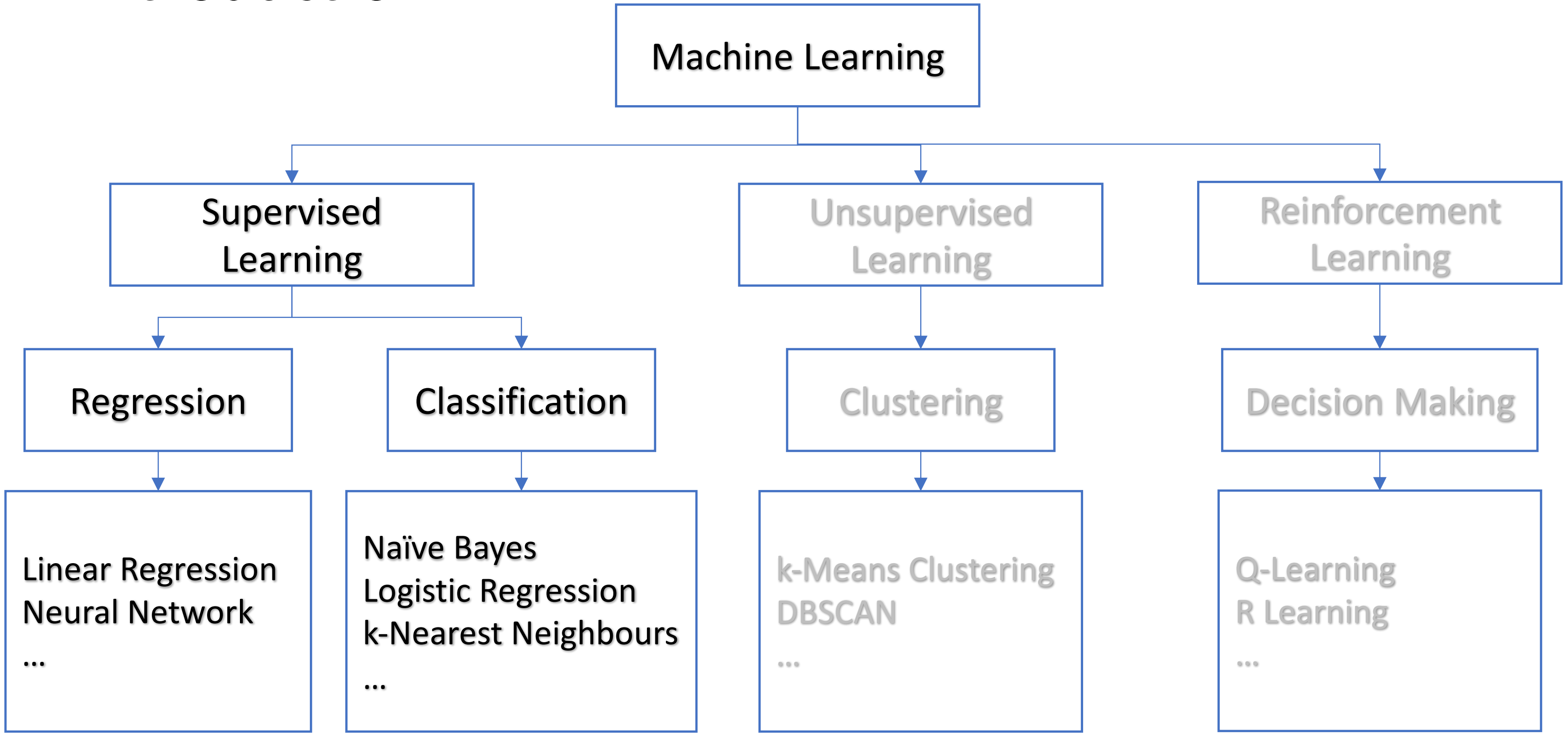
Introduction



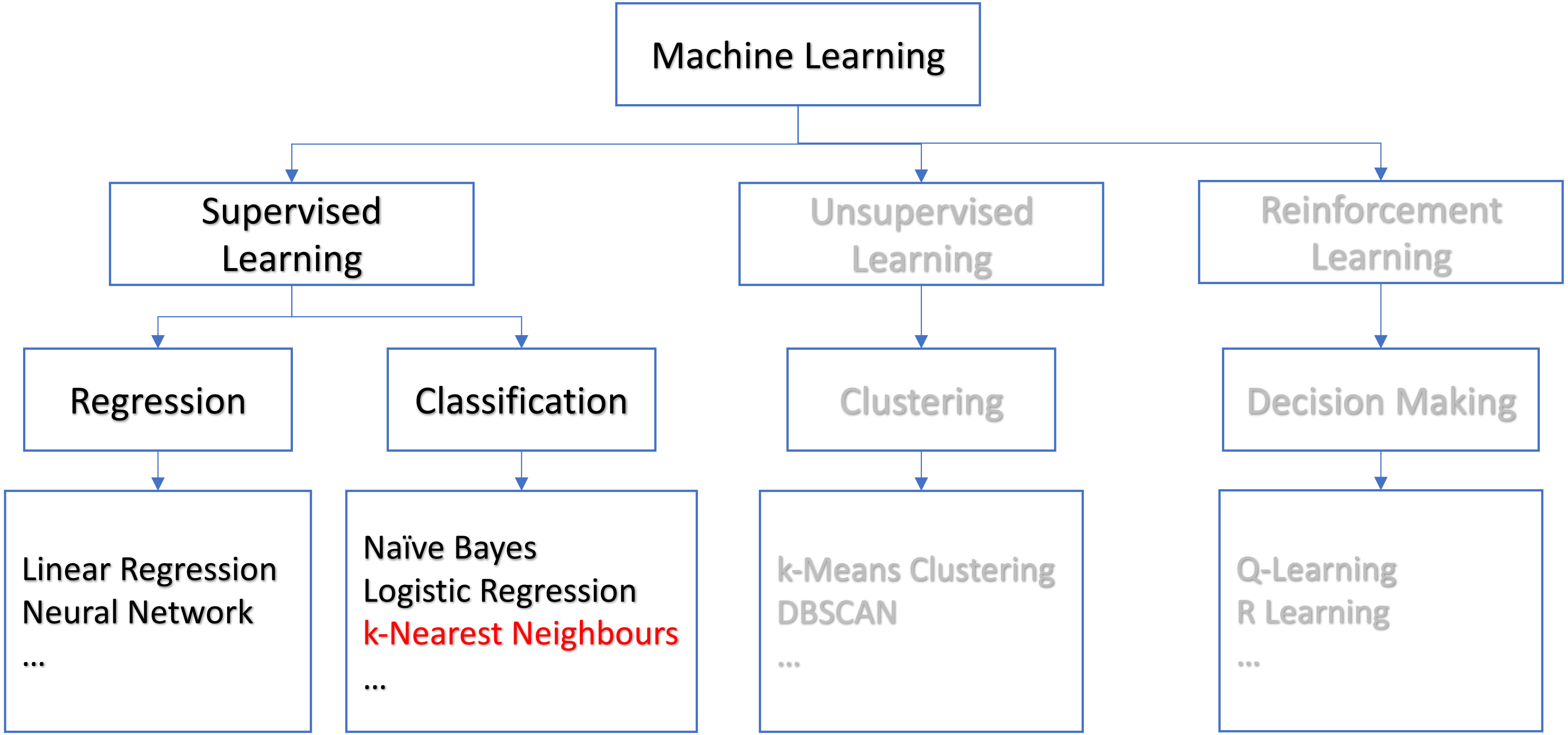
Introduction



Introduction



Introduction



Introduction

- Linear regression (as well as neural networks)
  - Uses the training data to estimate a fixed set of parameters $\boldsymbol{w}$
  - Defines our hypothesis $h_{\boldsymbol{w}}(\boldsymbol{x})$, thus making the training data useless

- A learning model that summarises data with a set of parameters of fixed size is called **parametric model**

Introduction

- Linear regression (as well as neural networks)
  - Uses the training data to estimate a fixed set of parameters $\boldsymbol{w}$
  - Defines our hypothesis $h_{\boldsymbol{w}}(\boldsymbol{x})$, thus making the training data useless

- A learning model that summarises data with a set of parameters of fixed size is called **parametric model**

- In this lecture, we will be using the following notation:
  - Variables are denoted by lowercase letters, e.g., x or y
  - Vectors are denoted by letters in bold, e.g., $\boldsymbol{x}$

Nonparametric Models

- A nonparametric model is a model that cannot be characterised by a bounded set of parameters
  - For instance, suppose that each prediction we make will consider all training examples, including the one from the previous prediction(s)
  - The set of examples grows over time, thus nonparametric

Nonparametric Models

- A nonparametric model is a model that cannot be characterised by a bounded set of parameters
  - For instance, suppose that each prediction we make will consider all training examples, including the one from the previous prediction(s)
  - The set of examples grows over time, thus nonparametric
- This approach is also called **instance-** or **memory-based learning**
  - The simplest method for instance-based learning is table lookup
  - For table lookup, we put all training examples in a table, and when looking for a value, we return the corresponding value
  - Problem: if the value does not exist, then a default value is returned

Nonparametric Models

- We can improve on table lookup with a slight variation:
  - Given a query $\boldsymbol{x}_q$, find the k examples that are *nearest* to $\boldsymbol{x}_q$

- This variation is called **k-Nearest Neighbours** (or **k-Nearest Neighbors**)

- Also, we can use the abbreviation k-NN

- The notation $NN(k, \boldsymbol{x}_q)$ denotes the set of k nearest neighbours

Overview

- Introduction
- **k-Nearest Neighbours**
- k-NN algorithm and pros/cons
- Evaluation Procedures

k-Nearest Neighbours (k-NN or KNN)

- Consider the following two-dimensional problem
  - Two classes: green or red $(y \in \{g, r\})$
  - New example (blue) to classify (majority vote)

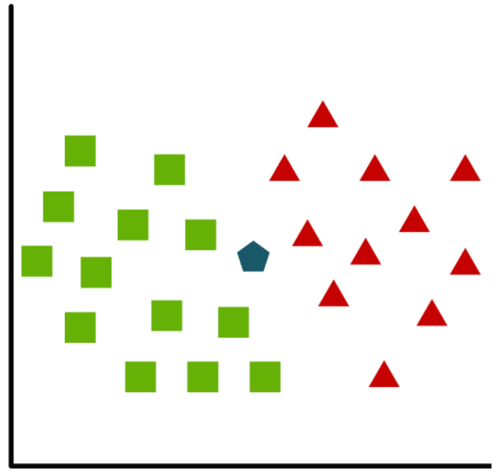


Image: taken from Ashraf *et al*., “A Review of Intrusion Detection Systems Using Machine and Deep Learning in Internet of Things: Challenges, Solutions and Future Directions”, *Electronics*, vol. 9, no. 7,2020.

k-Nearest Neighbours (k-NN or KNN)

- Consider the following two-dimensional problem
  - Two classes: green or red ($y \in \{g, r\}$)
  - New example (blue) to classify (majority vote)
  - Look at the k nearest neighbours
  - Let k = 3, to avoid issues
  - We want to predict the class of the new example

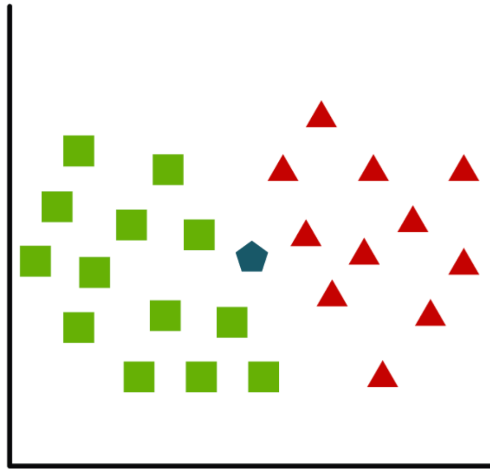


Image: taken from Ashraf *et al*., "A Review of Intrusion Detection Systems Using Machine and Deep Learning in Internet of Things: Challenges, Solutions and Future Directions", *Electronics*, vol. 9, no. 7,2020.

k-Nearest Neighbours (k-NN or KNN)

- Consider the following two-dimensional problem
  - Two classes: green or red $(y \in \{g, r\})$
  - New example (blue) to classify (majority vote)
  - Look at the k nearest neighbours
  - Let k = 3, to avoid issues
  - We want to predict the class of the new example
  - In this case, we predict green
  - Intuitively, a distance metric on the **input space**

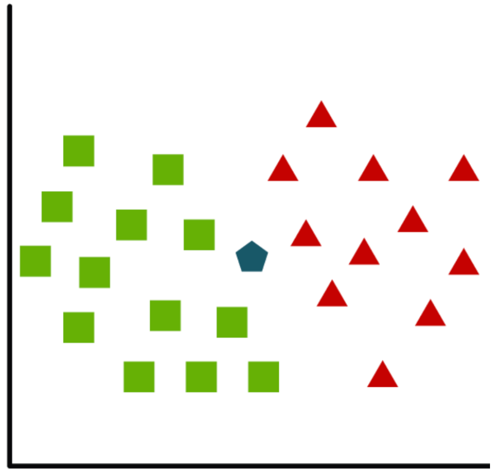


Image: taken from Ashraf *et al*., "A Review of Intrusion Detection Systems Using Machine and Deep Learning in Internet of Things: Challenges, Solutions and Future Directions", *Electronics*, vol. 9, no. 7,2020.

Distance Metrics

- Consider a problem with n dimensions, $\boldsymbol{x}^{(q)}$ being the new example
- The **Minkowski distance** (or L^p norm) is defined as

$$L^p\left(\boldsymbol{x}^{(q)}, \boldsymbol{x}^{(i)}\right) = \sqrt[p]{\sum_{j=1}^{n} \left|x_j^{(q)} - x_j^{(i)}\right|^p}$$

Distance Metrics

- With $p = 1$, the previous distance reduces to the **Manhattan distance**
- With Boolean attribute values, the so-called **Hamming distance** is used
- In general, the **Euclidean distance** is used, namely, when $p = 2$

$$L^2\left(\boldsymbol{x}^{(q)}, \boldsymbol{x}^{(i)}\right) = \sqrt[2]{\sum_{j=1}^{n} (x_j^{(q)} - x_j^{(i)})^2}$$

Example

- Let us consider a problem with 2 dimensions (green and red), $\boldsymbol{x}^{(4)} = [0.1, 0.6]$ being the new example and the following training examples are given: $\boldsymbol{x}^{(1)} = [0.4, 0.2] \in \{green\}$, $\boldsymbol{x}^{(2)} = [0.4, 0.1] \in \{green\}$, $\boldsymbol{x}^{(3)} = [0.2, 0.6] \in \{red\}$

- Let us use the Euclidean distance to predict the class of $\boldsymbol{x}^{(4)}$, with k = 1

$$L^2\left(\boldsymbol{x}^{(4)}, \boldsymbol{x}^{(1)}\right) = \sqrt[2]{\sum_{j=1}^{n} (x_j^{(4)} - x_j^{(1)})^2} =$$

Example

- Let us consider a problem with 2 dimensions (green and red), $\boldsymbol{x}^{(4)} = [0.1, 0.6]$ being the new example and the following training examples are given: $\boldsymbol{x}^{(1)} = [0.4, 0.2] \in \{green\}$, $\boldsymbol{x}^{(2)} = [0.4, 0.1] \in \{green\}$, $\boldsymbol{x}^{(3)} = [0.2, 0.6] \in \{red\}$

- Let us use the Euclidean distance to predict the class of $\boldsymbol{x}^{(4)}$, with k = 1

$$L^2\left(\boldsymbol{x}^{(4)}, \boldsymbol{x}^{(1)}\right) = \sqrt[2]{\sum_{j=1}^{n} (x_j^{(4)} - x_j^{(1)})^2} = \sqrt{(x_1^{(4)} - x_1^{(1)})^2 + (x_2^{(4)} - x_2^{(1)})^2}$$

Example

- Let us consider a problem with 2 dimensions (green and red), $\boldsymbol{x}^{(4)} = [0.1, 0.6]$ being the new example and the following training examples are given: $\boldsymbol{x}^{(1)} = [0.4, 0.2] \in \{green\}$, $\boldsymbol{x}^{(2)} = [0.4, 0.1] \in \{green\}$, $\boldsymbol{x}^{(3)} = [0.2, 0.6] \in \{red\}$

- Let us use the Euclidean distance to predict the class of $\boldsymbol{x}^{(4)}$, with k = 1

$$L^2\left(\boldsymbol{x}^{(4)}, \boldsymbol{x}^{(1)}\right) = \sqrt[2]{\sum_{j=1}^{n} (x_j^{(4)} - x_j^{(1)})^2} = \sqrt{(x_1^{(4)} - x_1^{(1)})^2 + (x_2^{(4)} - x_2^{(1)})^2}$$

$$= \sqrt{(0.1 - 0.4)^2 + (0.6 - 0.2)^2} = \sqrt{0.09 + 0.16} = 0.5$$

Example

- We repeat the process for all the points:

$$L^2\left(\boldsymbol{x}^{(4)}, \boldsymbol{x}^{(1)}\right) = 0.5$$
$$L^2\left(\boldsymbol{x}^{(4)}, \boldsymbol{x}^{(2)}\right) =$$
$$L^2\left(\boldsymbol{x}^{(4)}, \boldsymbol{x}^{(3)}\right) =$$

Example

- We repeat the process for all the points:

$$L^2\left(\boldsymbol{x}^{(4)}, \boldsymbol{x}^{(1)}\right) = 0.5$$
$$L^2\left(\boldsymbol{x}^{(4)}, \boldsymbol{x}^{(2)}\right) = 0.583$$
$$L^2\left(\boldsymbol{x}^{(4)}, \boldsymbol{x}^{(3)}\right) = 0.1$$

Example

- We repeat the process for all the points:

$$L^2\left(\boldsymbol{x}^{(4)}, \boldsymbol{x}^{(1)}\right) = 0.5$$
$$L^2\left(\boldsymbol{x}^{(4)}, \boldsymbol{x}^{(2)}\right) = 0.583$$
$$L^2\left(\boldsymbol{x}^{(4)}, \boldsymbol{x}^{(3)}\right) = 0.1$$

Example

- We repeat the process for all the points:

$$L^2\left(\boldsymbol{x}^{(4)}, \boldsymbol{x}^{(1)}\right) = 0.5$$
$$L^2\left(\boldsymbol{x}^{(4)}, \boldsymbol{x}^{(2)}\right) = 0.583$$
$$L^2\left(\boldsymbol{x}^{(4)}, \boldsymbol{x}^{(3)}\right) = 0.1$$

- What happens if k = 2? What if k = 3?

Overview

- Introduction
- k-Nearest Neighbours
- **k-NN algorithm and pros/cons**
- Evaluation Procedures

k-NN Algorithm

```
Input: training examples $x^{(i)} \in \boldsymbol{x}$ and their corresponding class $y^{(i)}$, a new query example $x^{(q)}$, number of neighbours k
Output: prediction of the new query example $x^{(q)}$
For each training example $x^{(i)} \in \boldsymbol{x}$
    Calculate the distance between the training example $x^{(i)}$ and the new query example $x^{(q)}$
    Keep the best k distances (the shortest distance) in a data structure T
Return the majority (or plurality in case of non-binary classification) vote (or average/median) of the class $y^{(i)}$ for the first k entries of T
```

k-NN for Regression Problems

- Consider the following two-dimensional problem
  - Let us consider examples that take a value between 1 and 5 ($y \in [1, 5]$)
  - Each example has a value in y, e.g., 3.2 or 4.1

k-NN for Regression Problems

- Consider the following two-dimensional problem
  - Let us consider examples that take a value between 1 and 5 ($y \in [1, 5]$)
  - Each example has a value in y, e.g., 3.2 or 4.1
  - We calculate the distances between the new query example and all the other examples in the training set
  - Look at the k nearest neighbours
  - We predict the value of the new example
  - In this case, we predict the average or median of the values of the k nearest neighbours

Example

- Let us consider a problem with 2 dimensions (green and red), $\boldsymbol{x}^{(4)} = [0.1, 0.6]$ being the new example and the following training examples are given: $\boldsymbol{x}^{(1)} = [0.4, 0.2] \in \{3.6\}$, $\boldsymbol{x}^{(2)} = [0.4, 0.1] \in \{3.9\}$, $\boldsymbol{x}^{(3)} = [0.2, 0.6] \in \{2.2\}$

- Let k = 2. We know the distances from the previous example:

$$L^2\left(\boldsymbol{x}^{(4)}, \boldsymbol{x}^{(1)}\right) = 0.5$$
$$L^2\left(\boldsymbol{x}^{(4)}, \boldsymbol{x}^{(2)}\right) = 0.583$$
$$L^2\left(\boldsymbol{x}^{(4)}, \boldsymbol{x}^{(3)}\right) = 0.1$$

Example

- Let us consider a problem with 2 dimensions (green and red), $\boldsymbol{x}^{(4)} = [0.1, 0.6]$ being the new example and the following training examples are given: $\boldsymbol{x}^{(1)} = [0.4, 0.2] \in \{3.6\}$, $\boldsymbol{x}^{(2)} = [0.4, 0.1] \in \{3.9\}$, $\boldsymbol{x}^{(3)} = [0.2, 0.6] \in \{2.2\}$

- Let k = 2. We know the distances from the previous example:

- The average is: $y^{(4)} = 2.9$

Problem with Numeric Independent Variables

- Different numeric attributes may have different scales

- For example, if x_1 is in [0,1] and x_2 is in [1, 10], x_2 will affect the distance more

Problem with Numeric Independent Variables

- To avoid this problem, we normalise the numeric input attributes of all data as in the following

$$normalise\left(x_j^{(i)}\right) = \frac{x_j^{(i)} - \min\limits_j}{\max\limits_j - \min\limits_j}$$

- Another approach is to calculate mean μ_j and standard deviation σ_j for each dimension j as: $(x_j^{(i)} - \mu_j)/\sigma_j$

Example

- Consider examples with two dimensions, where x_1 represents the age of a patient and and x_2 represents their weight, $y \in \{yes, no\}$

- Let us calculate the normalised values for $\boldsymbol{x}^{(1)}$:

Days	x_1	x_2	y
$\boldsymbol{x}^{(1)}$	14	70	yes
$\boldsymbol{x}^{(2)}$	12	90	no
$\boldsymbol{x}^{(3)}$	15	66	yes

Example

- Consider examples with two dimensions, where x_1 represents the age of a patient and and x_2 represents their weight, $y \in \{yes, no\}$

- Let us calculate the normalised values for $\boldsymbol{x}^{(1)}$:

- $normalise\left(x_1^{(1)}\right) = \frac{x_1^{(1)} - \min_1}{\max_1 - \min_1} =$

- $normalise\left(x_2^{(1)}\right) = \frac{x_2^{(1)} - \min_2}{\max_2 - \min_2} =$

Days	x_1	x_2	y
$\boldsymbol{x}^{(1)}$	14	70	yes
$\boldsymbol{x}^{(2)}$	12	90	no
$\boldsymbol{x}^{(3)}$	15	66	yes

Example

- Consider examples with two dimensions, where x_1 represents the age of a patient and and x_2 represents their weight, $y \in \{yes, no\}$

- Let us calculate the normalised values for $\boldsymbol{x}^{(1)}$:

- $normalise\left(x_1^{(1)}\right) = \frac{x_1^{(1)} - \min_1}{\max_1 - \min_1} = \frac{14 - 12}{15 - 12} = 0.667$

- $normalise\left(x_2^{(1)}\right) = \frac{x_2^{(1)} - \min_2}{\max_2 - \min_2} = \frac{70 - 66}{90 - 66} = 0.167$

Days	x_1	x_2	y
$\boldsymbol{x}^{(1)}$	14	70	yes
$\boldsymbol{x}^{(2)}$	12	90	no
$\boldsymbol{x}^{(3)}$	15	66	yes

k-NN Algorithm with Normalisation

Input: training examples $x^{(i)} \in \boldsymbol{x}$ and their corresponding class $y^{(i)}$, a new query example $x^{(q)}$, number of neighbours k

Output: prediction of the new query example $x^{(q)}$

```
For each training example $x^{(i)} \in \boldsymbol{x}$
    Calculate the normalised distance between the training example $x^{(i)}$ and the new query example $x^{(q)}$
    Keep the best k distances (the shortest distance) in a data structure T
Return the majority (or plurality in case of non-binary classification) vote (or average/median) of the class $y^{(i)}$ for the first k entries of T
```

Different Input Attributes

- For numeric input attributes, e.g., age in [0, 100], we calculate the distance as shown in previous examples
- For ordinal input attributes, e.g., sunny in {yes, no}, we can convert the values to numeric values: yes = 1, no = 0
- For categorical input attributes, e.g., phone_brand in {samsung, apple, nokia}, we can use the following approach:
  - If the value of the query example is the same as the value for example i, then their difference is 0. Formally, if $\left(x_j^{(q)} = x_j^{(i)}\right)$, then $\left(x_j^{(q)} - x_j^{(i)}\right) = 0$
  - Otherwise, their difference is 1. Formally, if $\left(x_j^{(q)} \neq x_j^{(i)}\right)$, then $\left(x_j^{(q)} - x_j^{(i)}\right) = 1$

Summary

k-NN Learning Algorithm

- The algorithm does not have proper training
- We simply store all training data, which increase over time
- We normalise by calculating the minimum and maximum in the training data

k-NN Model

- All training data, the values of the numeric input attributes

k-NN prediction for an instance ($\boldsymbol{x}^{(\boldsymbol{q})}, y =?$)

- Find the k nearest neighbours whose distance to $\boldsymbol{x}^{(i)}$ is the smallest
- For classification problems, majority vote. For regression problems, average/median

Pros and Cons of k-NN

Pros

- Training is simple and fast: just store training data
- Find the class of the new example based on most similar examples present in the training data

Cons

- It uses large space in memory: we need to store all data
- Running the algorithm can be slow if we have many training examples and many dimensions

Complexity

- Given a set of N examples and a query example, we calculate the distance to the query example from each one and keep the best k

Complexity

- Given a set of N examples and a query example, we calculate the distance to the query example from each one and keep the best k
- The above runs in $O(N)$ with a (sequential) table
- However, these methods are designed for large datasets. Therefore, better data structure are desirable

Complexity

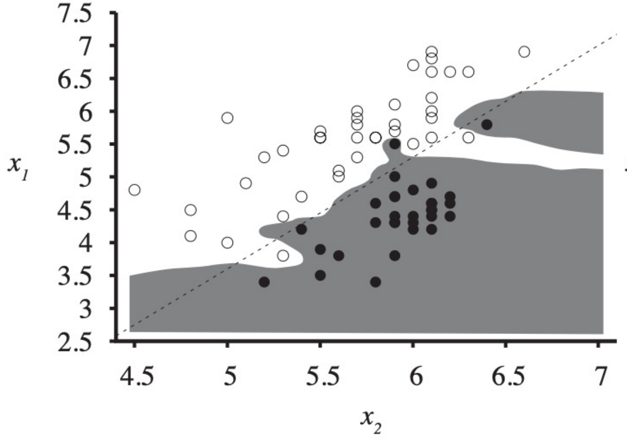
- Given a set of N examples and a query example, we calculate the distance to the query example from each one and keep the best k

- The above runs in $O(N)$ with a (sequential) table

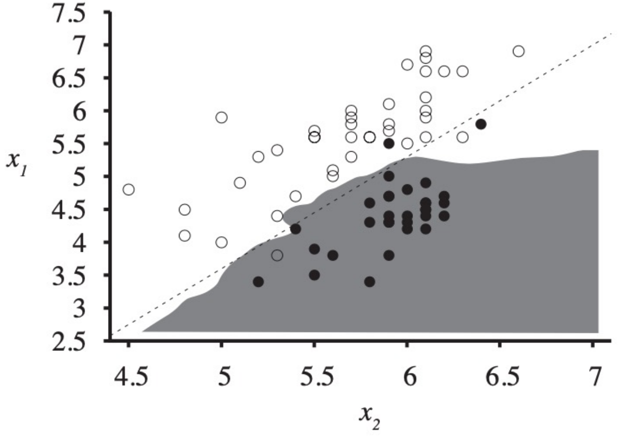
- However, these methods are designed for large datasets. Therefore, better data structure are desirable

- A binary tree runs in $O(\log N)$, whereas a hash table runs in $O(1)$

Discussion



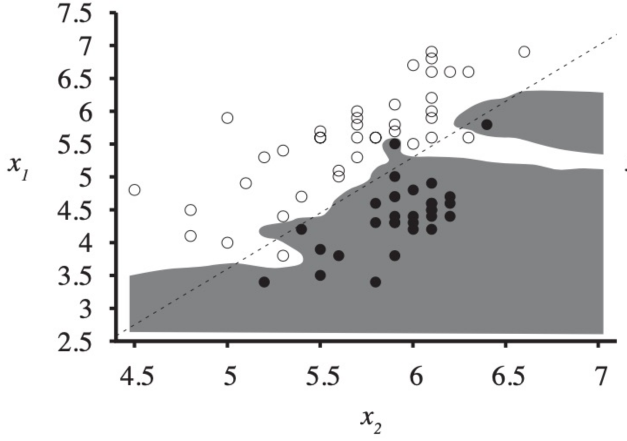
$(k = 1)$



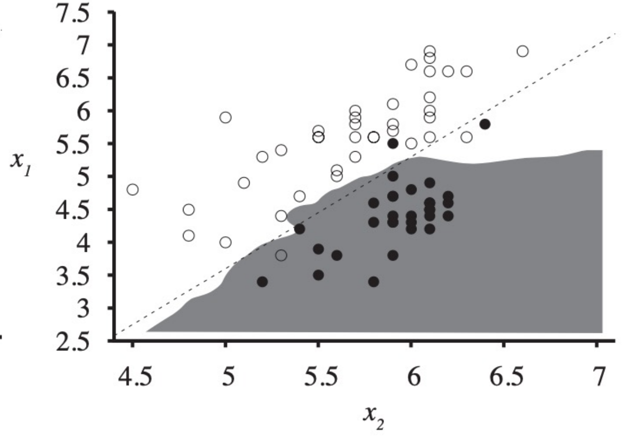
$(k = 5)$

- Nonparametric methods are still subject to under- and overfitting

Discussion



$(k=1)$



$(k=5)$

- Nonparametric methods are still subject to under- and overfitting
- In the above case, 1-NN is overfitting as it reacts too much to the black outlier in the upper right and the white at (5.4, 3.7)
- The 5-NN decision boundary is good; higher k would underfit

Overview

- Introduction
- k-Nearest Neighbours
- k-NN algorithm and pros/cons
- **Evaluation Procedures**

Evaluation Procedures

- A supervised learning method consists of 3 main elements:
  - Model, namely, the form of function we want to learn (with free parameters)
  - Cost function, namely, the misfit between a particular function from the model (provided a training set)
  - Training algorithm, namely, gradient descent minimisation of cost function

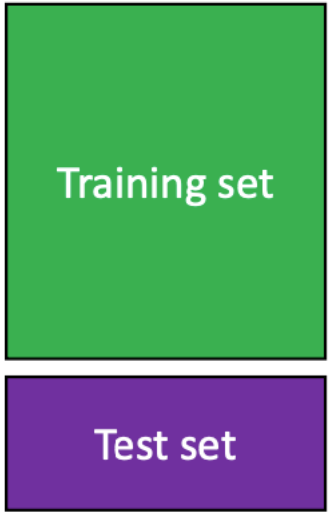
- Running the training algorithm on training data learns the “best” values of the free parameters, yielding a predictor

This part is adapted from Ata Kaban’s slides and worked examples from Andrew Moore’s tutorial slides
https://sites.astro.caltech.edu/~george/aybi199/AMooreTutorials/

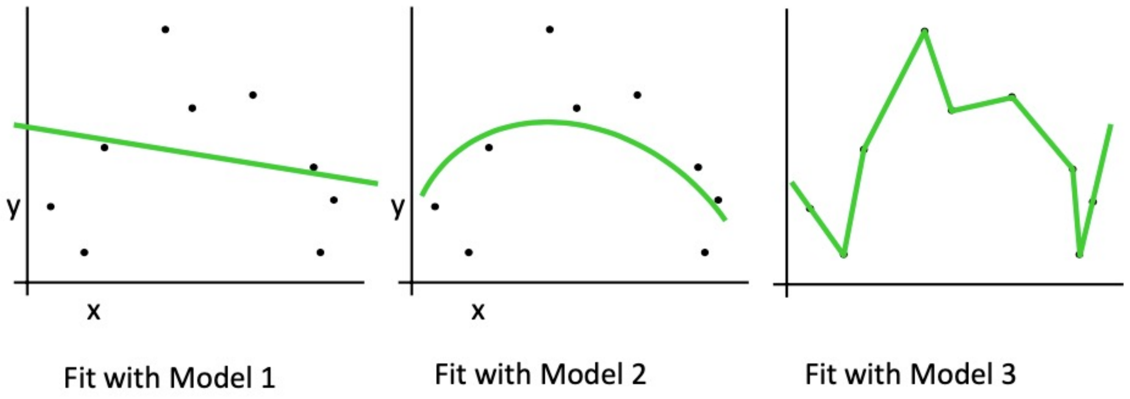
Evaluation of Predictor before Deployment

- We use evaluation procedures to determine how good our model is. In other words, we want to estimate the future performance of a predictor

- To do so, we randomly split the available annotated data into:
  - A **training set** – to estimate all the free parameters
  - A **test set** – to evaluate the trained predictor before deployment



Choosing the Model



- Hyperparameters are “higher-level” free parameters
- Each hyperparameter value corresponds to a different model
- Which model? We need a **criterion to estimate future performance**

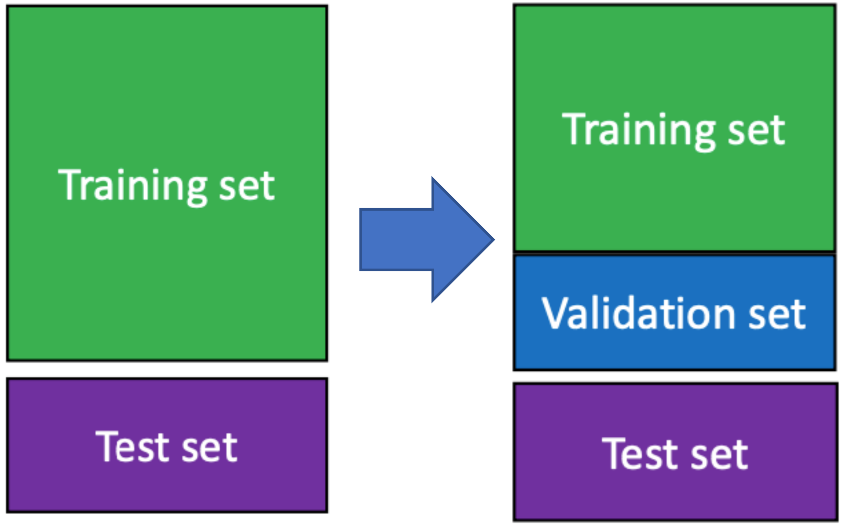
Evaluating Models for Model Choice

- Do not confuse this with evaluating a predictor (model already chosen)
- The training set is used for training within a chosen model
- The test set is used to evaluate the performance of the trained predictor
- **None of the above can be used to choose the model!**
  - If we are tempted to use the test set, we no longer have an independent dataset to evaluate the final predictor before deployment

Evaluating Models for Model Choice

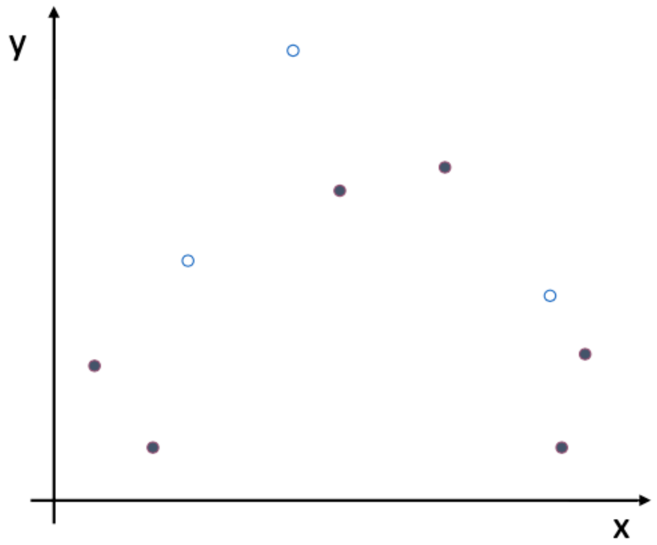
- Idea: to choose between models or hyperparameters, take a subset of the training set and create a validation set

- Methods
  - Holdout validation
  - Cross-validation
  - Leave-one-out validation

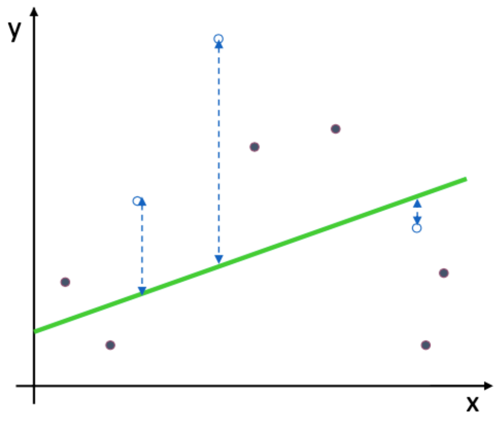


Method 1: Holdout Validation

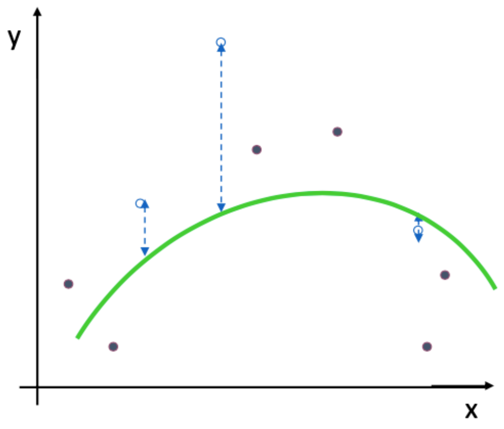
1. Randomly choose 30% of data to form a validation set
2. Keep the rest in the training set
3. Train your model on the training set
4. Estimate the test performance on the validation set
5. Choose model with lowest validation error
6. Re-train with chosen model on joined training and validation to obtain predictor
7. Estimate future performance on test set
8. Ready to deploy predictor



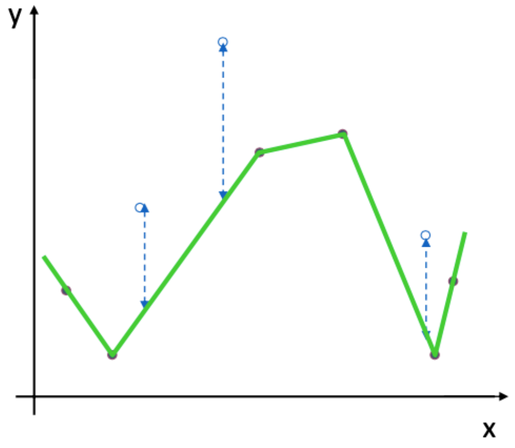
Example: Holdout Validation



Model 1
Mean Squared Validation Error = 2.4

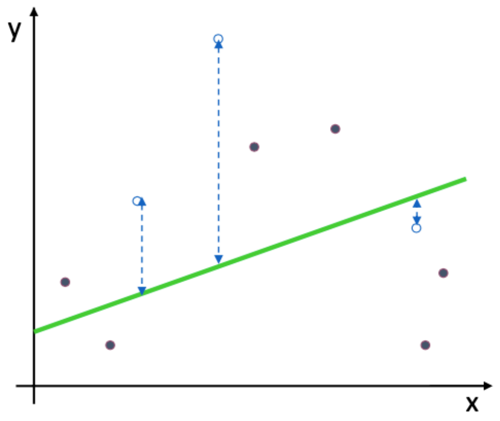


Model 2
Mean Squared Validation Error = 0.9

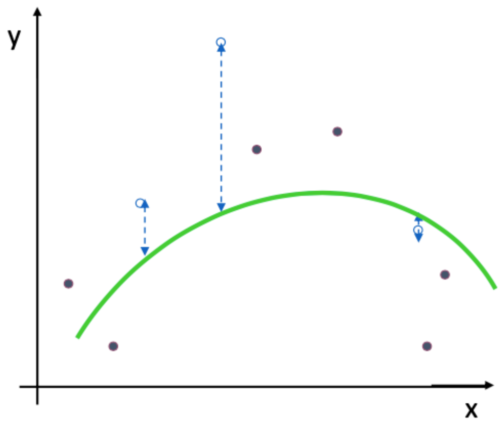


Model 3
Mean Squared Validation Error = 2.2

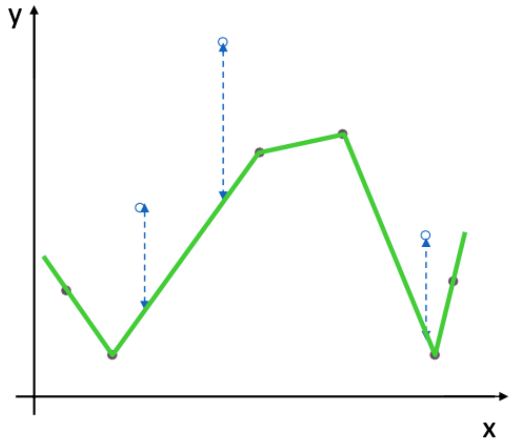
Example: Holdout Validation



Model 1
Mean Squared Validation Error = 2.4



Model 2
Mean Squared Validation Error = 0.9



Model 3
Mean Squared Validation Error = 2.2

Method 1: Holdout Validation, Step 4

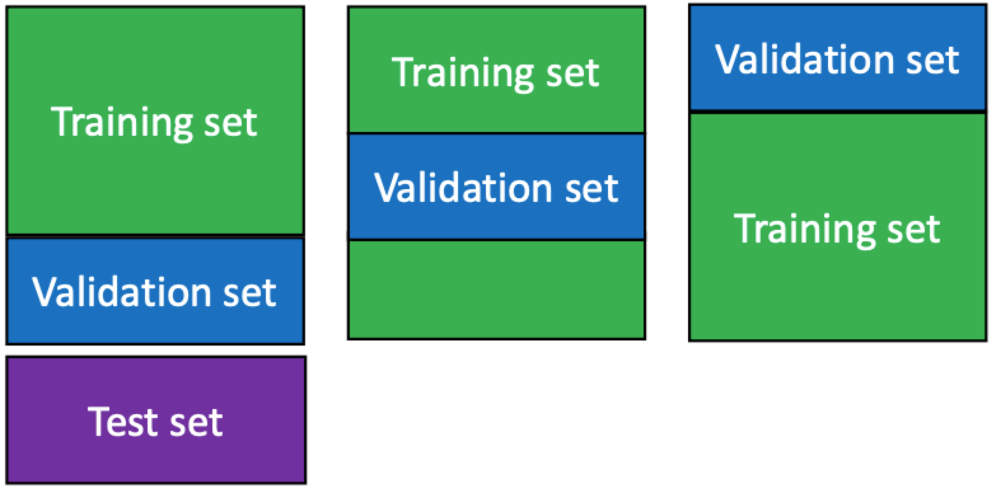
4. Estimate the test performance on the validation set

- Different approaches for regression and for classification:
  - In regression, we compute the cost function (mean square error) on the examples of the validation set (instead of the training set)
  - In classification, we compute the 0-1 error metric on validation set, i.e.,

$$\frac{\#\text{ wrong predictions}}{\#\text{ predictions}} = 1 - \text{Accuracy}$$

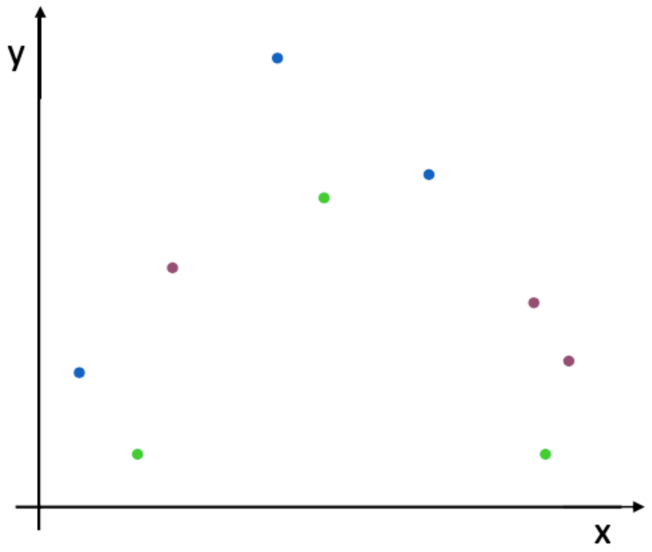
Method 2: k-Fold Cross-Validation

1. Randomly split the training set into k disjoint sets of equal size (k=3 in this example)
2. Use the k-1 of those together for training
3. Use the remaining one for validation
4. Permute the k sets and repeat k times
5. Average the performance on the k validation sets



Example: k-Fold Cross-Validation

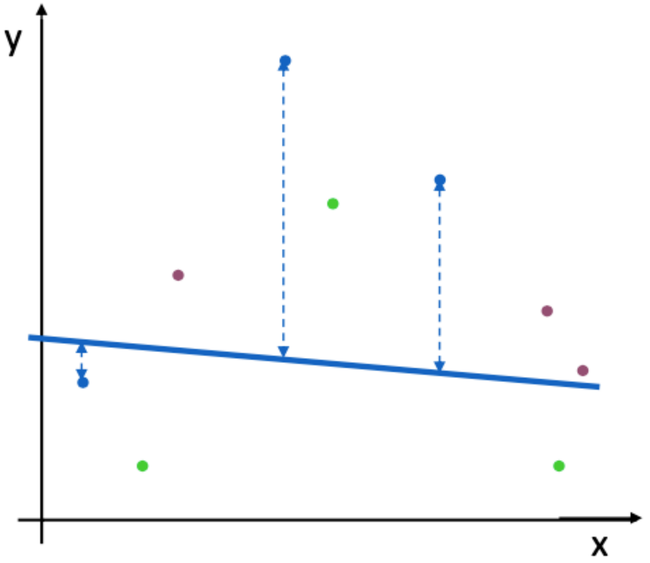
Randomly split the dataset into k=3 partitions



Example: k-Fold Cross-Validation

Randomly split the dataset into k=3 partitions

Blue partition: train on all the data <u>except</u> the blue partition. Compute the validation error using the data in the blue partition

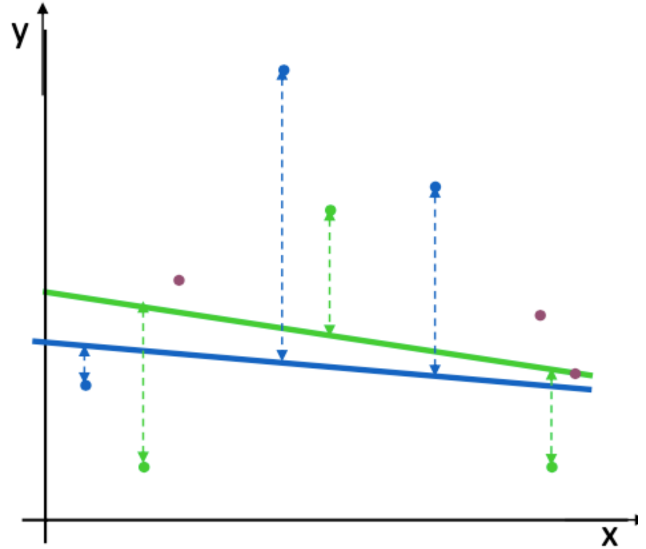


Example: k-Fold Cross-Validation

Randomly split the dataset into k=3 partitions

Blue partition: train on all the data <u>except</u> the blue partition. Compute the validation error using the data in the blue partition

Green partition: train on all the data <u>except</u> the green partition. Compute the validation error using the data in the green partition



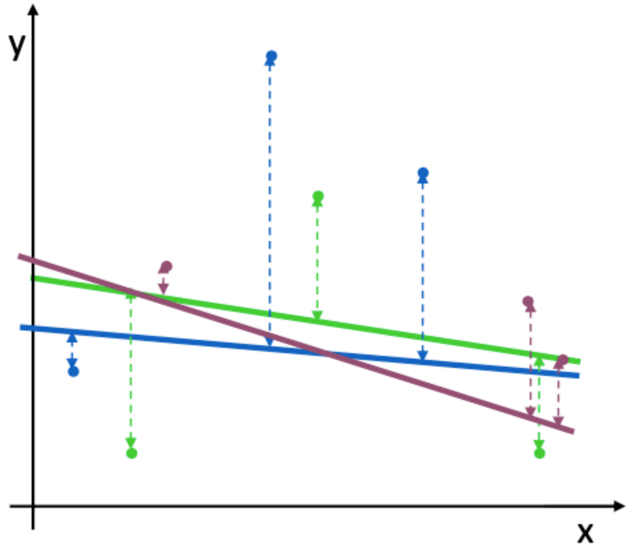
Example: k-Fold Cross-Validation

Randomly split the dataset into k=3 partitions

Blue partition: train on all the data <u>except</u> the blue partition. Compute the validation error using the data in the blue partition

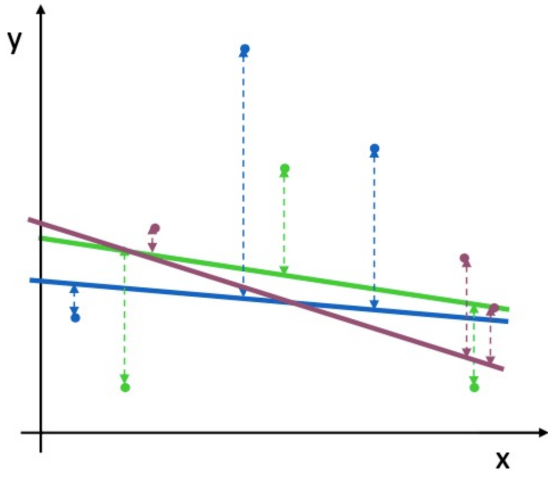
Green partition: train on all the data <u>except</u> the green partition. Compute the validation error using the data in the green partition

Purple partition: train on all the data <u>except</u> the purple partition. Compute the validation error using the data in the purple partition

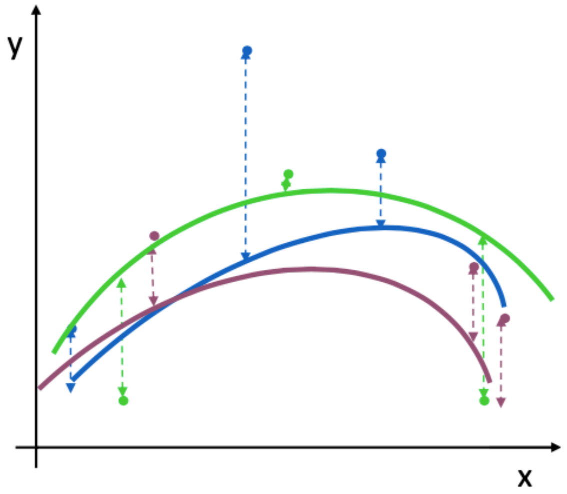


Example: k-Fold Cross-Validation

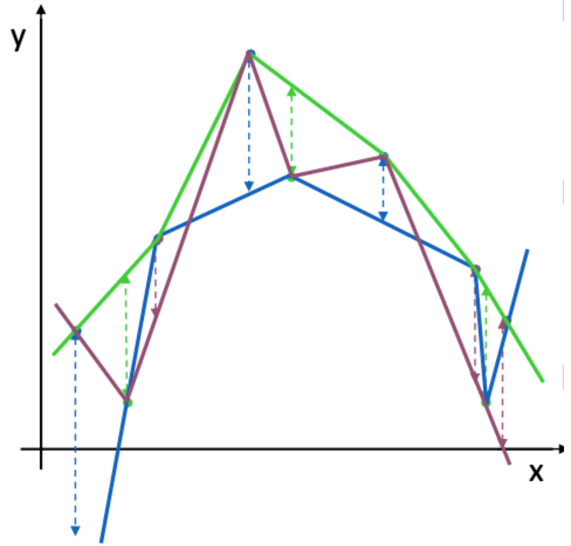
Take the mean of these errors



Model 1
$\text{MSE}_{3FOLD}=2.05$



Model 2
$\text{MSE}_{3FOLD}=1.11$



Model 3
$\text{MSE}_{3FOLD}=2.93$

Method 3: Leave-One-Out Validation

- We leave out **a single example** for validation, and train on all the remaining annotated data
- For a total of N examples, we repeat this process N times, each time leaving out a single example
- Take the <u>average of the validation errors</u> as measured on the left-out examples

Method 3: Leave-One-Out Validation

- We leave out **a single example** for validation, and train on all the remaining annotated data
- For a total of N examples, we repeat this process N times, each time leaving out a single example
- Take the average of the validation errors as measured on the left-out examples

- Same as N-fold cross-validation where N is the number of labelled examples

Advantages and Disadvantages

	Advantages	**Disadvantages**
Holdout validation	Computationally cheapest	Most unreliable if sample size is not large enough
3-fold	Slightly more reliable than holdout	• Wastes 1/3-rd annotated data. • Computationally 3-times as expensive as holdout
10-fold	• Only wastes 10% • Fairly reliable	• Wastes 10% annotated data • Computationally 10-times as expensive as holdout
Leave-one-out	Doesn't waste data	Computationally most expensive

$$\begin{tikzcd} \text{Large sample} \arrow[d, leftrightarrow] \\ \text{Small sample} \end{tikzcd}$$

Aims of the Session

You should now be able to:

- Explain the steps of k-Nearest Neighbours
- Apply k-Nearest Neighbours to problems involving numeric, ordinal and categorical input attributes
- Use evaluation procedures to estimate the performance of a predictor

References

- Russell, A. S., and Norvig, P. (2010), *Artificial Intelligence A Modern Approach*, 3rd Edition. Prentice Hall.
    - Chapter 18 – Learning from Examples (Section 18.8 up to 18.8.1, Section 18.8.4)