# Supervised Learning

@cambridgespark

CAMBRIDGE SPARK

# Fitting a response to data

**Classification**

- Training data: (point, class)
- Ex: Reading digits (image, {0, 1, ...9})

**Regression**

- Training data: (point, value)
- Ex: Predict salary (characteristics, salary)

CAMBRIDGE SPARK

# Building a supervised model

True relationship between x and y represented by a function **f**:

$$X \rightarrow f(x) = y$$

You want to build a model **f'**:

$$X \rightarrow f'(x) \approx y$$

CAMBRIDGE SPARK

# Building a supervised model

True relationship between x and y represented by a function **f**:

$$X \rightarrow f(x) = y$$

You want to build a model **f'**:

$$X \rightarrow f'(x) \approx y$$

Neural Networks can, in theory, represent any function.

CAMBRIDGE SPARK

# Training and testing a model

Workflow

- **TRAINING**: Consider some of the data (training data):
  - Build a model **f’** which works well on it
- **TESTING**: Consider the rest of the data (test data)
  - Check how **f’** is performing on that

# Training and testing a model

Workflow

- **TRAINING**: Consider some of the data (training data):
  - Build a model f' which works well on it
- **TESTING**: Consider the rest of the data (test data)
  - Check how f' is performing on that

The test data is necessarily distinct from the training data in order to assess how your model generalises

CAMBRIDGE SPARK

# Train-test split

- Usually around 70%-30% split
- Randomised for representativity
- Can be stratified if the data is imbalanced

When doing one hot encoding, what if the test set has new labels? Common source of bugs!

# Evaluating a model

**Loss function**: Measure of performance of a specific model evaluated on a given set of points S (usually training/test points)

E.g.: Mean Squared Error (MSE) for regression:

$$L_{\mathrm{MSE}}(\widehat{f}) = \sum_{i \in S} (\widehat{f}(x_i) - y_i)^2$$

CAMBRIDGE SPARK

# Training a model = Minimising the loss

Consider a parametric model **f'** defined by a vector of parameters β

Best β is such that the loss is minimised

# Training a model = Minimising the loss

Consider a parametric model f’ defined by a vector of parameters $\beta$

Best $\beta$ is such that the loss is minimised

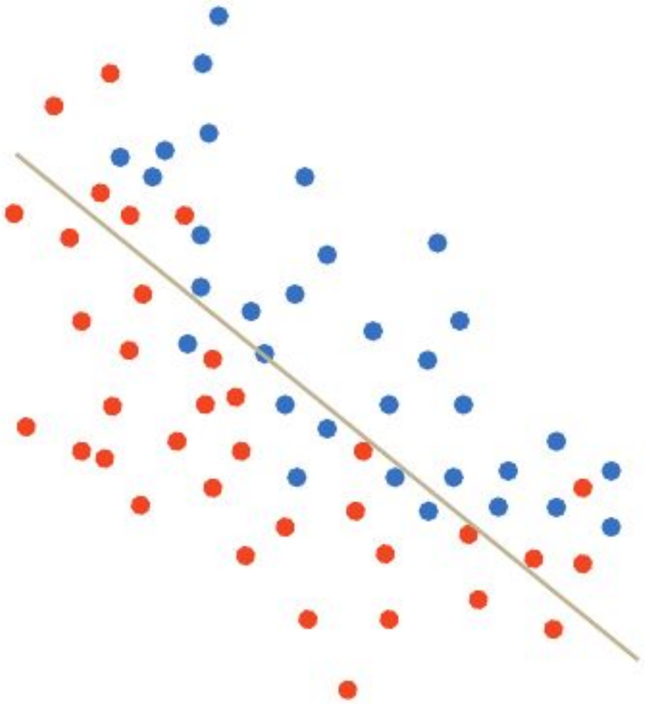
Training problem:

**Find $\beta^* \in \arg\min L(f')$**

# Training and testing: the cases

- Poor on training: **underfitting**

# Underfitting



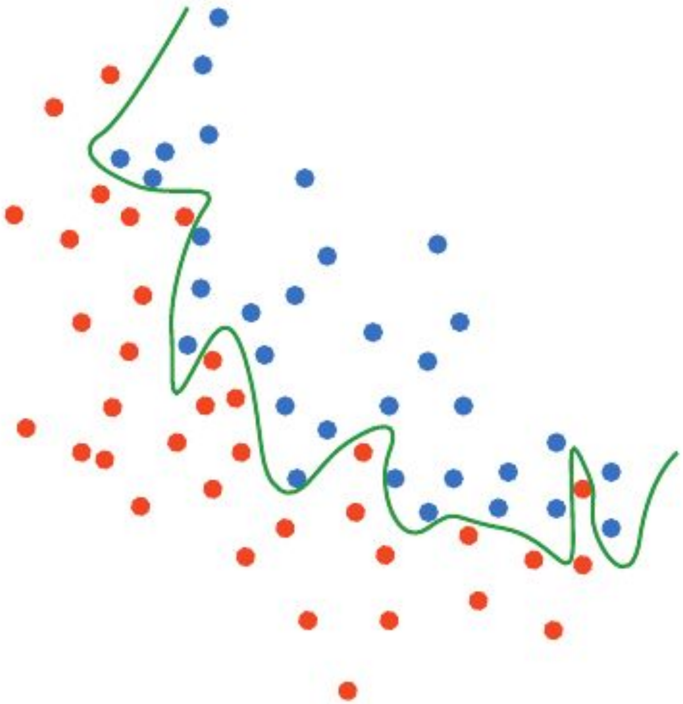
CAMBRIDGE SPARK

# Training and testing: the cases

- Poor on training: **underfitting**
- Good on training but poor on testing: **overfitting**

CAMBRIDGE SPARK

# Overfitting



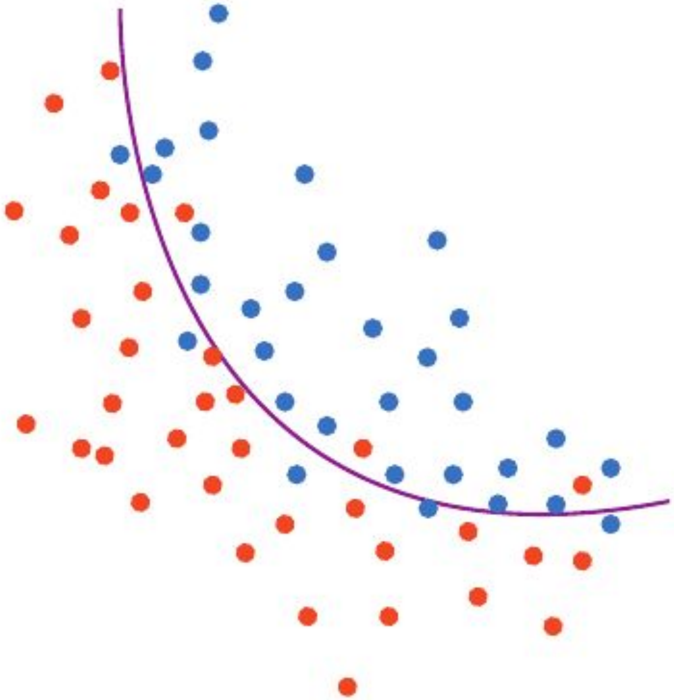
# Training and testing: the cases

- Poor on training: **underfitting**
- Good on training but poor on testing: **overfitting**
- Good on training and good on testing: **job done**

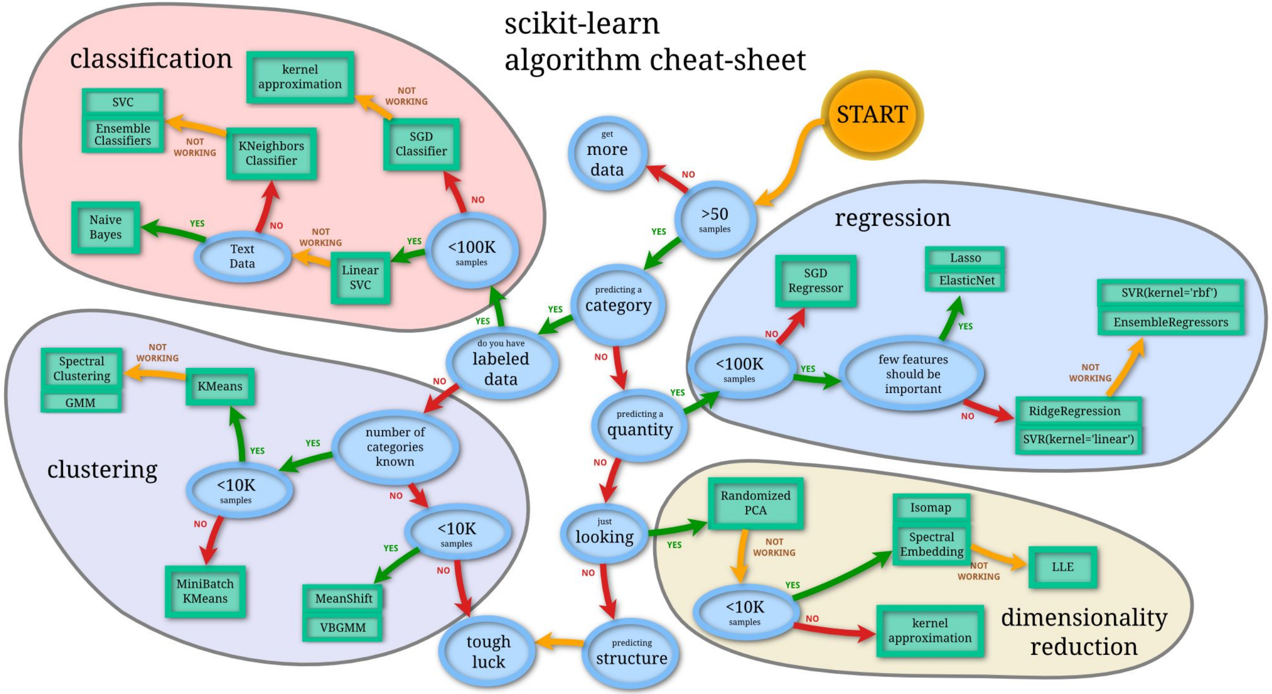
More about this later, now how do you build a model in Python?

CAMBRIDGE SPARK

# Good fit



# Scikit-learn algorithm map




CAMBRIDGE SPARK

Hands-on session

*01-supervised_learning.ipynb*
*Part 1: Preparing the data*

# A simple classifier: K-Nearest-Neighbours

Basic Idea:

- Consider the K training points closest to a test point
- Majority vote over which class the point should belong to

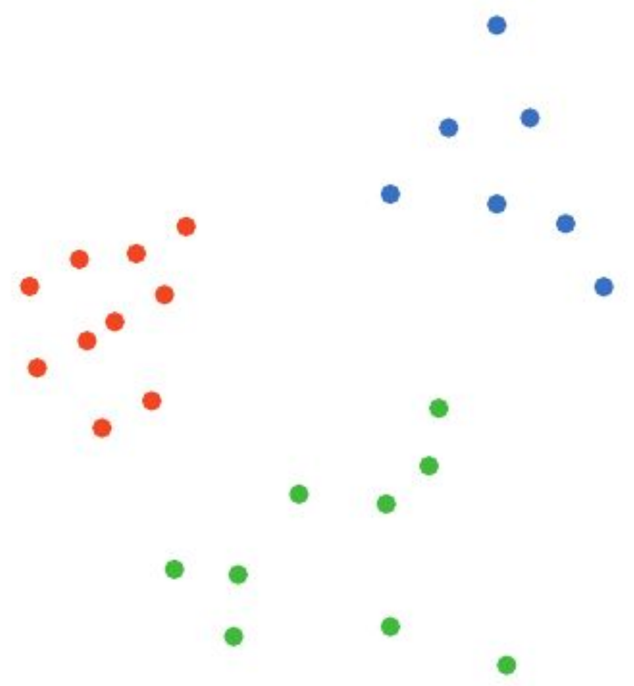
# A simple classifier: K-Nearest-Neighbours

Basic Idea:

- Consider the K training points closest to a test point
- Majority vote over which class the point should belong to

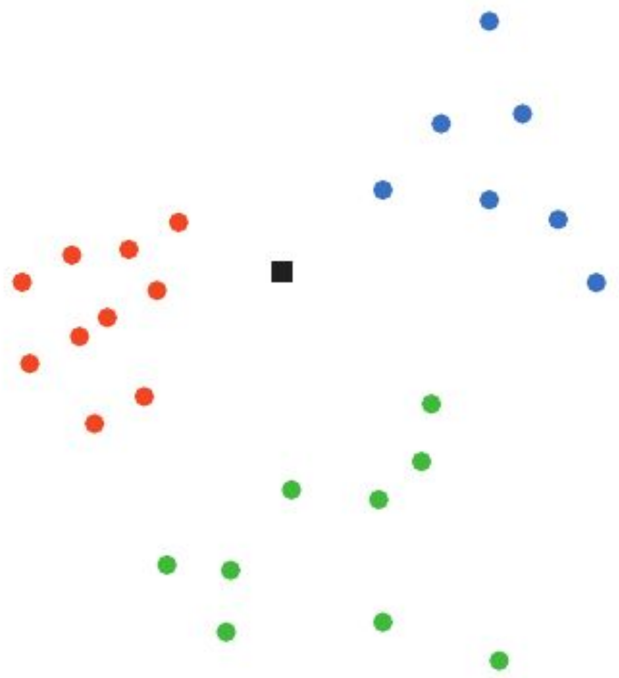
Let’s see...

# K-Nearest-Neighbours



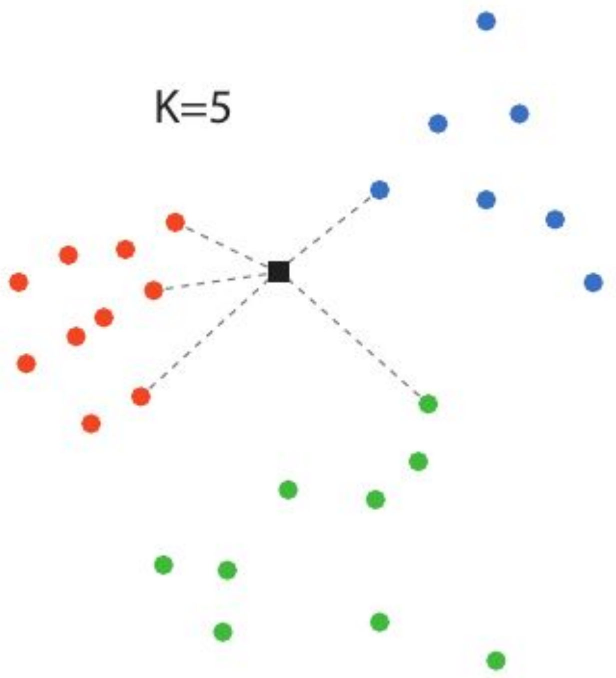
CAMBRIDGE SPARK

# K-Nearest-Neighbours

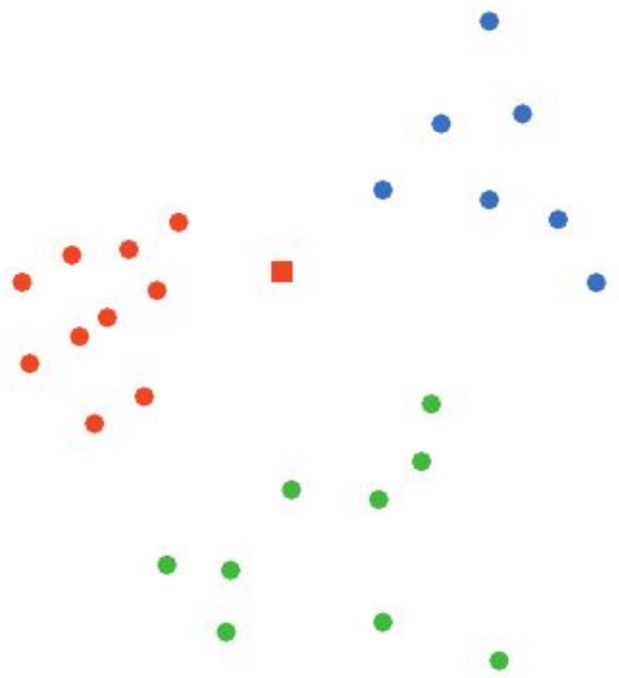


CAMBRIDGE SPARK

# K-Nearest-Neighbours



# K-Nearest-Neighbours



# Hyperparameter tuning

For kNN we only need to tune the K. How do we do it?

# Hyperparameter tuning

For kNN we only need to tune the K. How do we do it?

1. Train a model on training data
2. Test on the test data
3. Adjust the model
4. Repeat

CAMBRIDGE SPARK

# Hyperparameter tuning

For kNN we only need to tune the K. How do we do it?

1. Train a model on training data
2. Test on the test data
3. Adjust the model
4. Repeat

Leads to information about the test set being leaked into the model! (overfitting)

# Hyperparameter tuning

For kNN we only need to tune the K. How do we do it?

1. Train a model on training data
2. Test on the test data
3. Adjust the model
4. Repeat

Leads to information about the test set being leaked into the model! (overfitting)

You should only use the training set to adjust the model...

# Hyperparameter tuning

For kNN we only need to tune the K. How do we do it?

1. Train a model on training data
2. Test on the test data
3. Adjust the model
4. Repeat

Leads to information about the test set being leaked into the model! (overfitting)

You should only use the training set to adjust the model...

You should only use the test set for a final assessment of the model...

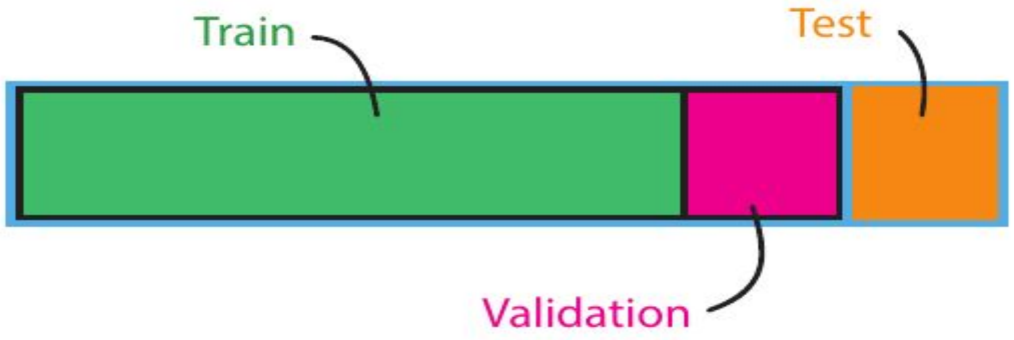
CAMBRIDGE SPARK

# Training-Validation-Testing

1. Take a slice of the training data -> validation set
2. Apply the same procedure as before to adjust the model but using the smaller training set and validation set
3. Report performance on the untouched test set

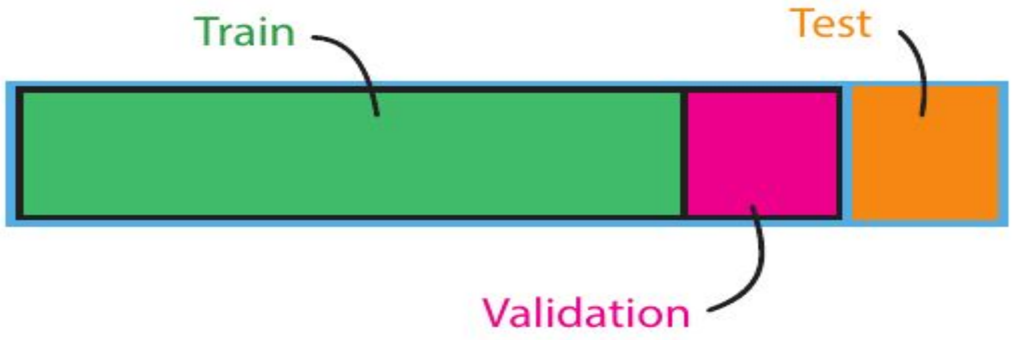
# Training-Validation-Testing

1. Take a slide of the training data -> validation set
2. Apply the same procedure as before to adjust the model but using the smaller training set and validation set
3. Report performance on the untouched test set



# Training-Validation-Testing

1. Take a slide of the training data -> validation set
2. Apply the same procedure as before to adjust the model but using the smaller training set and validation set
3. Report performance on the untouched test set




Nice but we're not using the whole training data...

CAMBRIDGE SPARK

# From validation to cross-validation

We would like to:

- Use something like validation to tune our model without looking at the test set
- Exploit all the data from the initial training set

**WARNING**: There will be a lot of K's in the following slides, those are **not** referring to the k in kNN but for k-fold cross validation.

# K-fold cross validation

1. Shuffle the training set and slice it in k batches

For each model to train and evaluate:

2. Pick one slide s of the k batches
3. Use the rest of the batches to train the model
4. Compute accuracy on slice s (validation)

# K-fold cross validation

1. Shuffle the training set and slice it in k batches

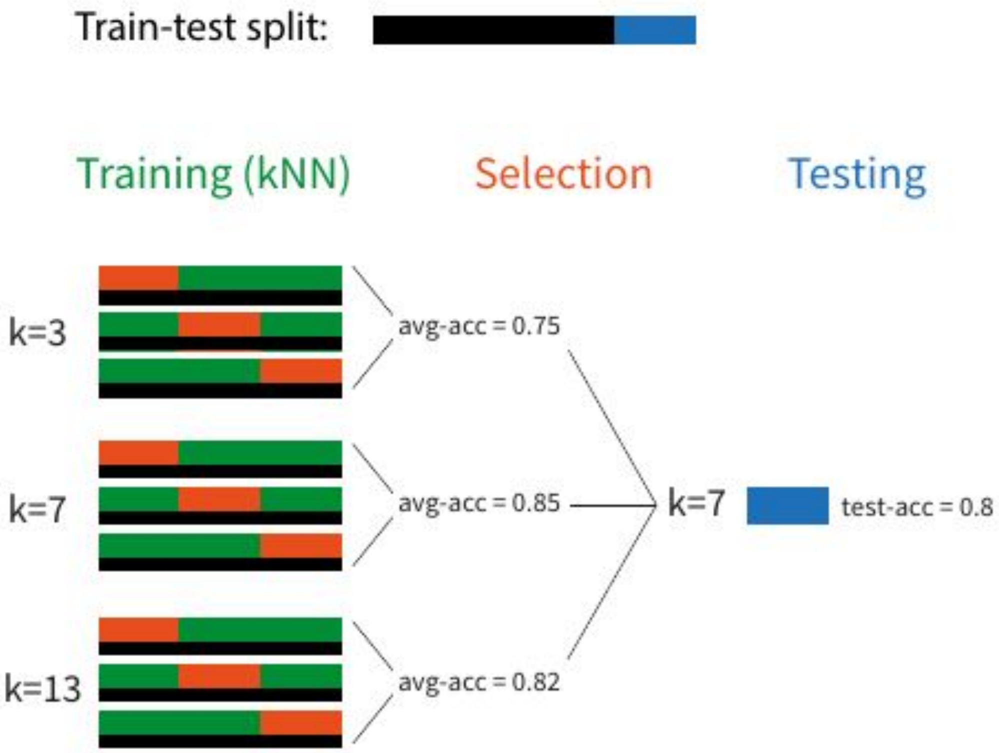
For each model to train and evaluate:

2. Pick one slide s of the k batches
3. Use the rest of the batches to train the model
4. Compute accuracy on slice s (validation)
5. Go back to 2) and repeat for all slices
6. Report the average accuracy

Pick the model with the highest average accuracy

CAMBRIDGE SPARK

# K-fold cross validation

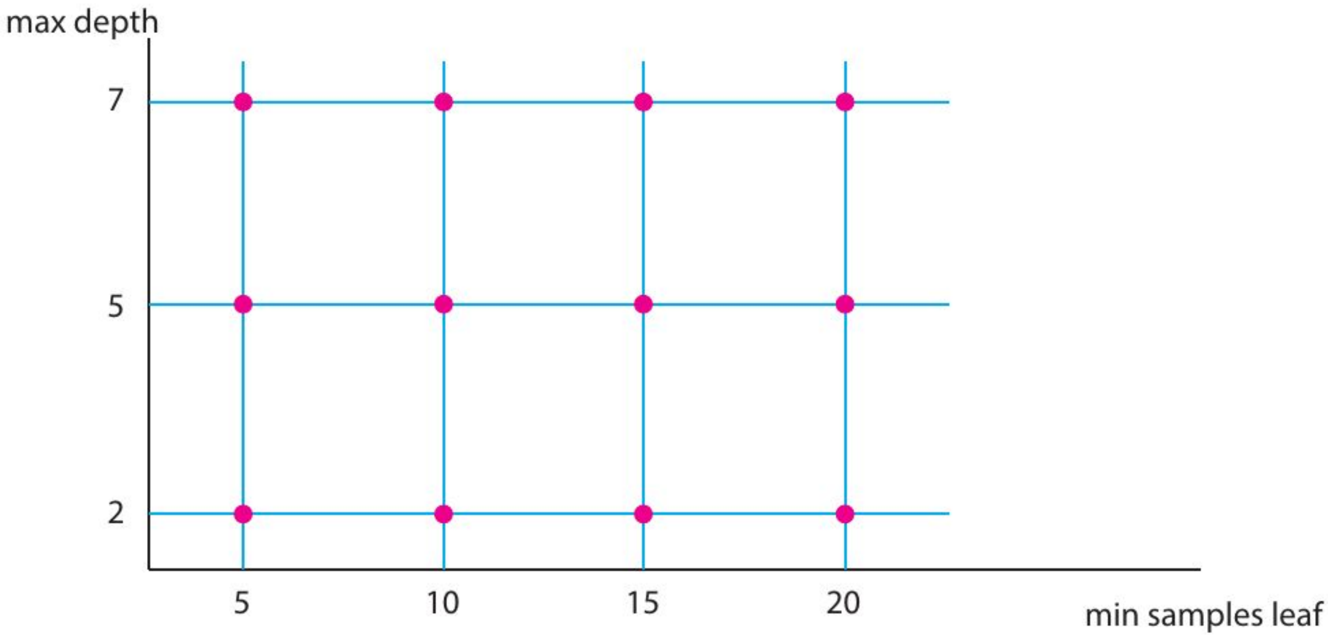



CAMBRIDGE SPARK

# K-fold cross validation

Now that we have a way to evaluate models with different parameters, how do we choose the parameters we want to test?

# Grid Search




CAMBRIDGE SPARK

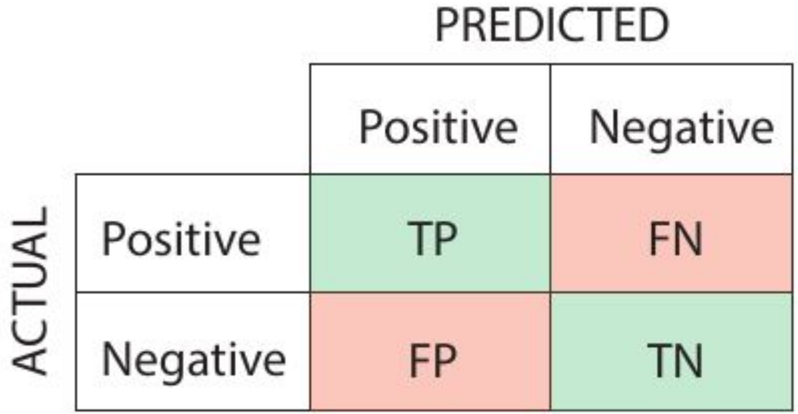
# Testing a binary classifier

On the test data, compare the predicted responses y' = f'(x) to the actual response y.

| | | PREDICTED | |
|---|---|---|---|
| | | Positive | Negative |
| ACTUAL | Positive | TP | FN |
| | Negative | FP | TN |

# Testing a binary classifier

On the test data, compare the predicted responses y’ = f’(x) to the actual response y.




- Predicted Jargon from clinical trials, positive/negative defined by context
- This is the confusion matrix

# Testing a binary classifier

On the test data, compare the predicted responses y’ = f’(x) to the actual response y.

| | | PREDICTED | |
|---|---|---|---|
| | | Positive | Negative |
| ACTUAL | Positive | TP | FN |
| | Negative | FP | TN |

**Accuracy**: (TP + TN) / N
**Sensitivity/Recall**: TP / (TP + FN)
**Precision**: TP / (TP + FP)

The metric of importance is usually dictated by context.

**Example**: Fraud

- Flagging a clean transaction as fraudulent (OK)
- Not flagging a fraudulent transaction (bad)

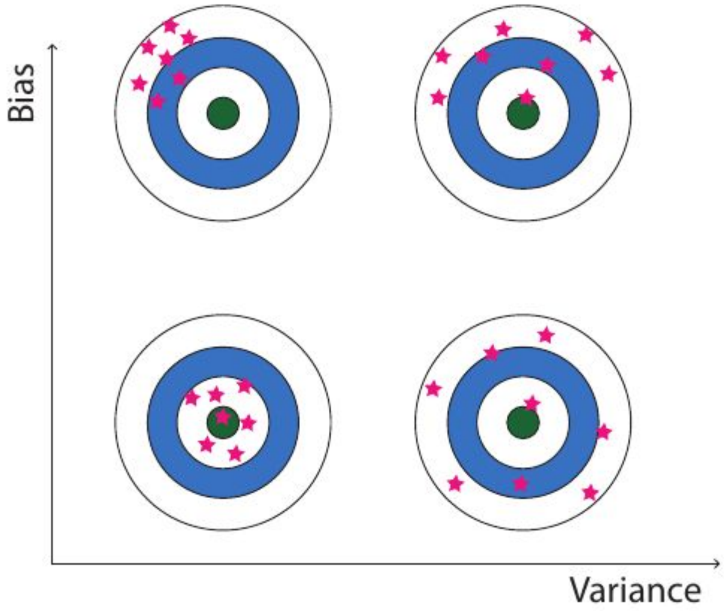
CAMBRIDGE SPARK

Hands-on session

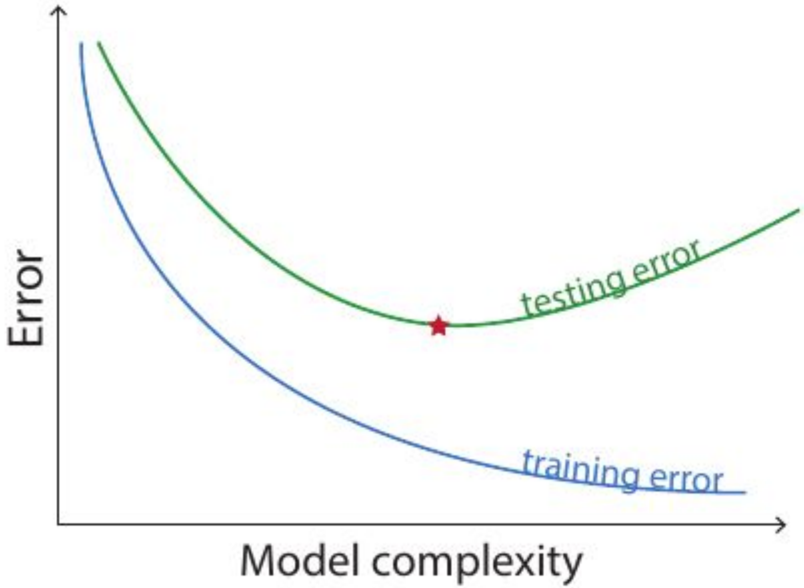
*02-supervised_learning.ipynb*
*Part 2: Training a model*

# Bias and Variance

One way of visualising the performances achieved by a model is to imagine that it corresponds to a dart thrower.



# Bias-Variance Tradeoff




CAMBRIDGE SPARK

# Bias-Variance Tradeoff