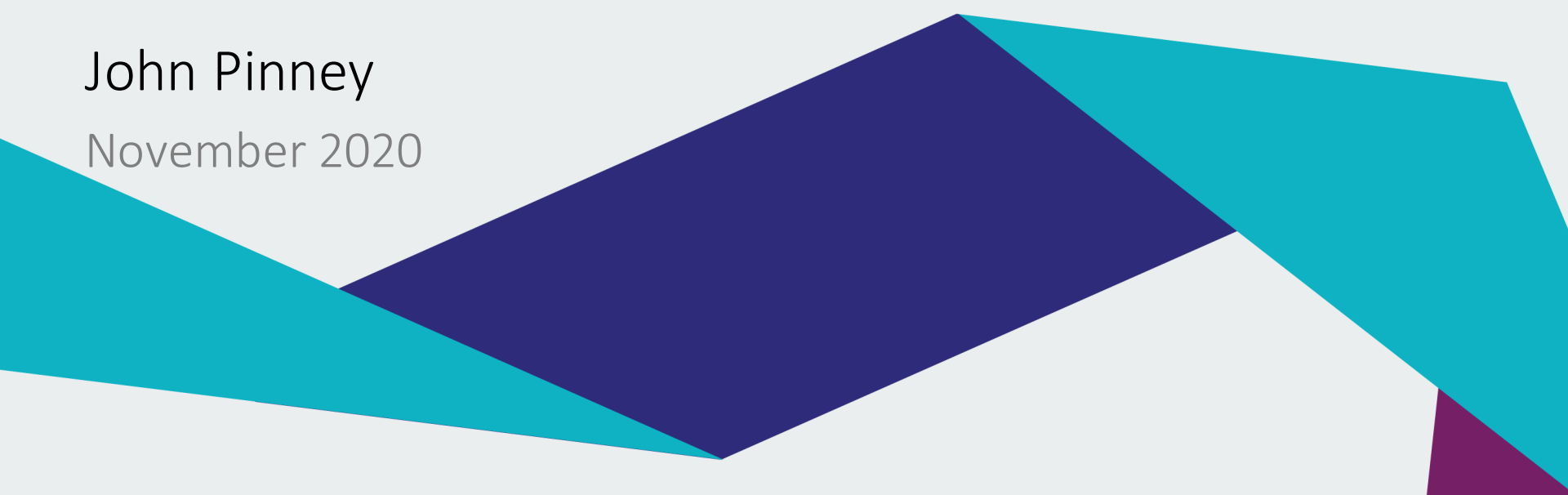


Introduction to Machine Learning

Part 2: Evaluating and improving performance

John Pinney

November 2020



Intended learning outcomes

After attending this workshop, you will be better able to:

- Explain the difference between supervised and unsupervised learning.
- Select a suitable machine learning method for a given application.
- Prepare your own training and testing data sets.
- Evaluate the performance of a machine learning experiment.

Overview

Evaluating performance

Train/Validate/Test

Cross-validation

Regression metrics

Classification metrics

ROC curve

Improving performance

Bias vs variance

Feature selection

Tree pruning

Ensemble methods

Bagging

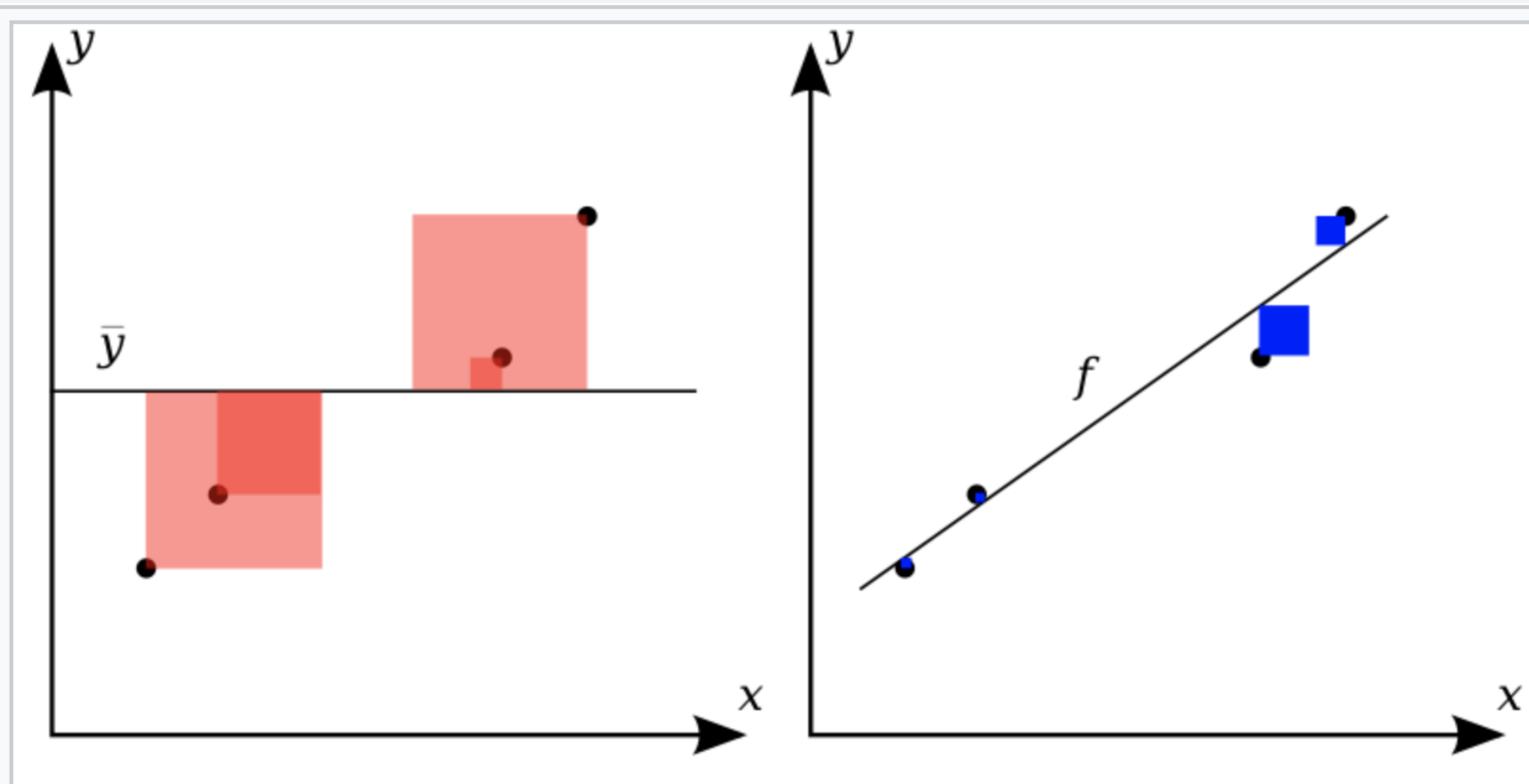
Boosting

Evaluating performance

How can we compare different ML methods for the same task?

- We need **performance metrics** that measure the similarity between the model's predictions and the **ground truth**.
- You are probably already familiar with the metric R^2 for assessing the fit between a linear model and the data.
A good model would have R^2 close to 1.

Coefficient of determination, R^2



$$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}$$



R^2 example

With the **abalone** dataset,

Use a **tree** to predict **rings** from the other features.

What is R^2 for this model?

Overfitting

- We saw in the example that the evaluation might look *very* different depending on whether we use the training dataset or an unseen dataset.
- A complex model might perform very well on the training data, but if it is **overfitted** then it will extend poorly to unseen data.
- We must always be careful to make sure that our evaluation metrics are calculated on a separate **testing dataset**.

Avoiding contamination

- In some circumstances, we need to take extra care to ensure that the test data is truly independent of the training data.
- Part of the skill in designing a good machine learning experiment is in recognising how to filter data to avoid this kind of contamination.
 - e.g. if predicting protein function from sequence, ensure that there are no pairs of protein sequences that are too closely related.

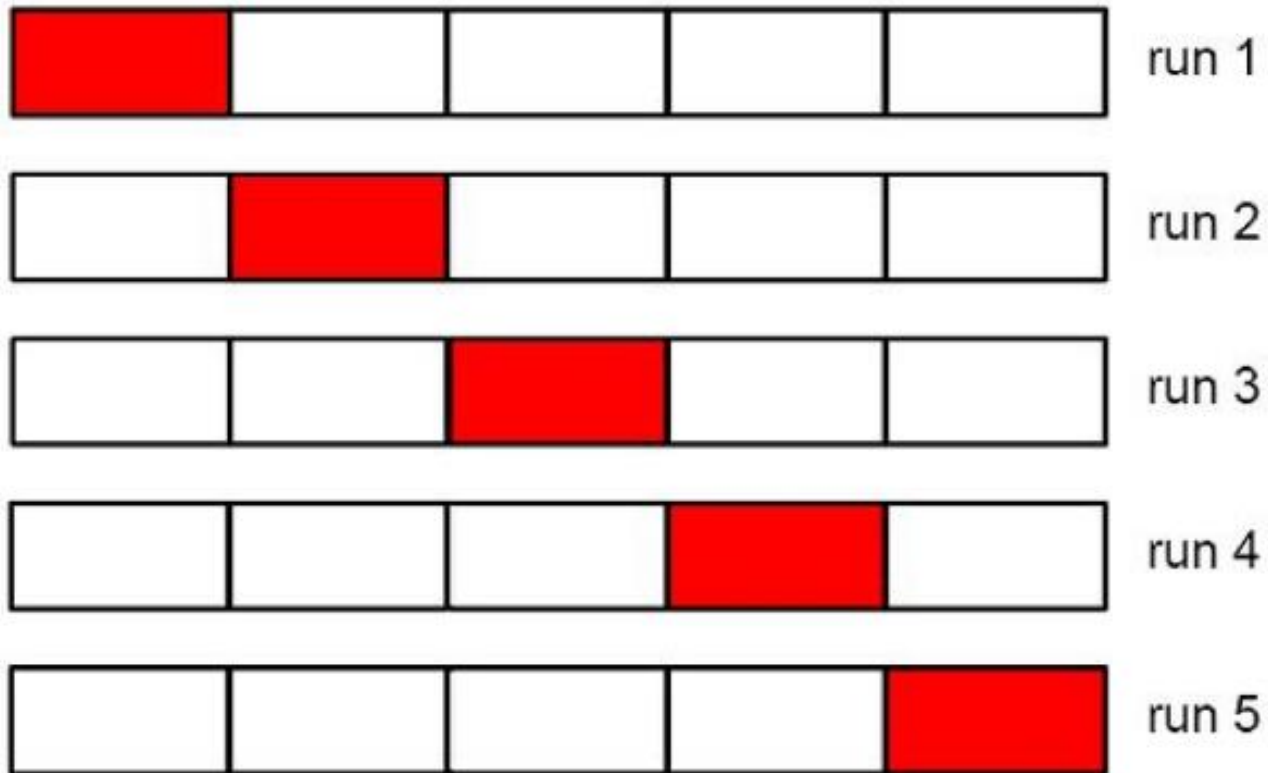
Train / validate / and test !

- It is crucial that the test data remains unseen during the development of the ML model.
- However, we may need to “tune” various hyperparameters or model architectures to arrive at an effective model.
- After the testing data is removed, we can therefore split the remaining data into “training” and “validation” sets so that we can get an idea of the performance of the current iteration of the model.

Cross-validation

- A very commonly used technique is to apply **k-fold cross-validation**, where
 - the data is split into k equally sized subsets.
 - we train on $(k-1)$ subsets and evaluate on the remaining subset.
 - we repeat so that each subset is used in validation.
 - we report the mean performance metric over the k folds.

Cross-validation (k=5)



Other metrics for regression

- There are several other metrics we can use to evaluate regression, e.g.
- **Mean absolute error (MAE)**
(smaller is better)
- **Root mean squared error (RMSE)**
(more sensitive to outliers than MAE)
- **Adjusted R^2**
(useful when comparing models with different numbers of variables)

R^2 exercise

With the **abalone** dataset,

Use a *tree* to predict **rings** from the other features.

What is R^2 for this model

using 5-fold cross-validation?

Does a *linear regression* perform better?

What is R^2 when the model is just a *constant*?

Evaluate on the test data and report a final R^2 for your preferred model.

Metrics for classification

Consider this **confusion matrix** for a binary classification

(predictions)	(ground truth)		
	$y = 1$	$y = 0$	Σ
	$\hat{y} = 1$	TP FP	$\hat{N}_+ = TP + FP$
	$\hat{y} = 0$	FN TN	$\hat{N}_- = FN + TN$
Σ	$N_+ = TP + FN$	$N_- = FP + TN$	$N = TP + FP + FN + TN$

We can define a number of metrics from these numbers, which capture different aspects of performance.

(predictions)	(ground truth)		Σ
	$y = 1$	$y = 0$	
$\hat{y} = 1$	TP	FP	$\hat{N}_+ = TP + FP$
$\hat{y} = 0$	FN	TN	$\hat{N}_- = FN + TN$
Σ	$N_+ = TP + FN$	$N_- = FP + TN$	$N = TP + FP + FN + TN$

sensitivity (recall) = proportion of N_+ that are detected.

specificity = proportion of N_- that are detected.

precision = proportion of \hat{N}_+ that are correct.

accuracy = proportion of N that are correct.

(predictions)	(ground truth)		Σ
	$y = 1$	$y = 0$	
$\hat{y} = 1$	TP	FP	$\hat{N}_+ = TP + FP$
$\hat{y} = 0$	FN	TN	$\hat{N}_- = FN + TN$
Σ	$N_+ = TP + FN$	$N_- = FP + TN$	$N = TP + FP + FN + TN$

sensitivity (recall) = TP / N_+

specificity = TN / N_-

precision = TP / \hat{N}_+

accuracy = $(TP + TN) / N$

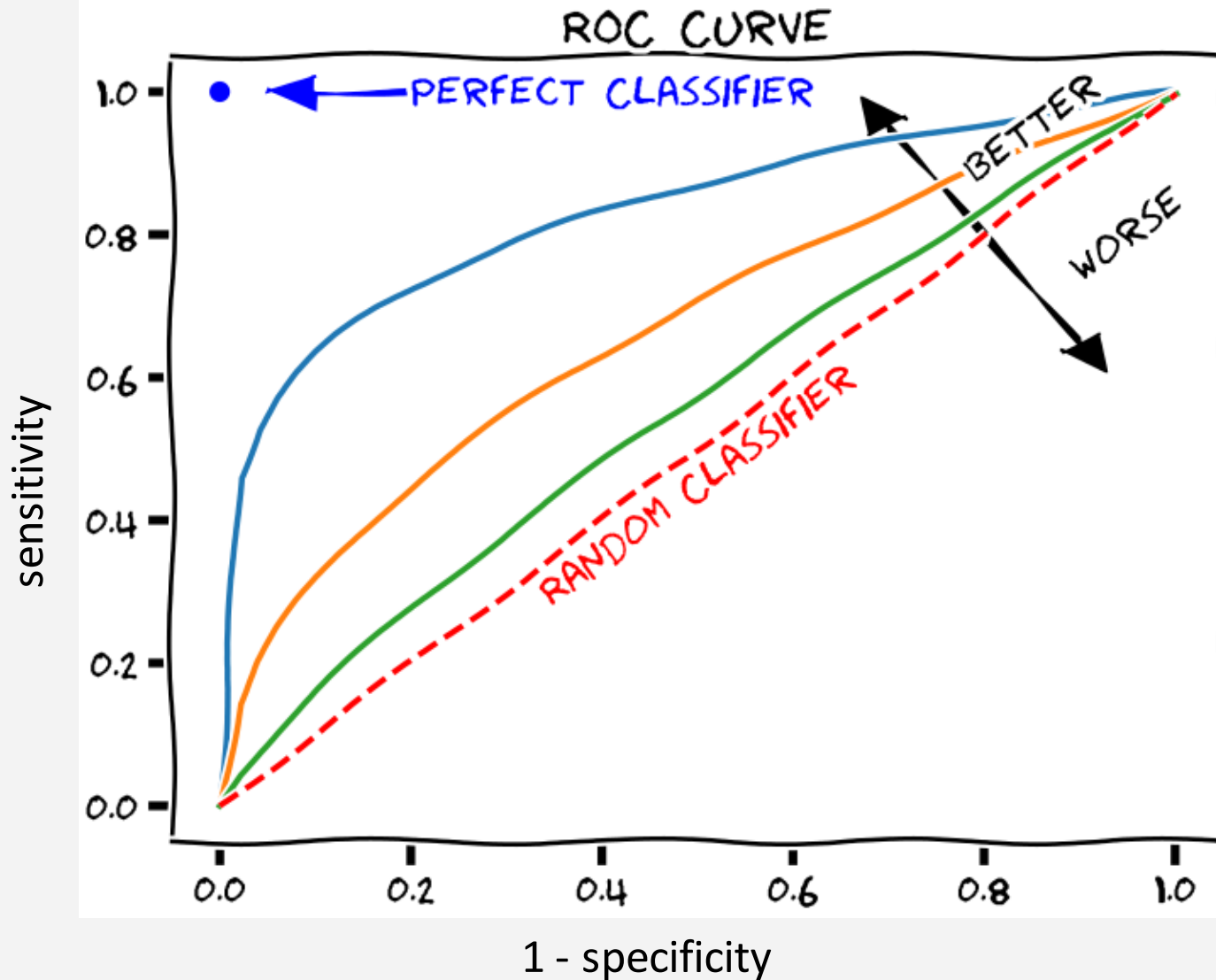
Receiver operating characteristic

There is always a **trade-off** between sensitivity and specificity. If a method reports some kind of **probability score** for its predictions then we could adjust a threshold to *tune* between maximum sensitivity and maximum specificity.

The **receiver operating characteristic** is a graphical way to examine performance in terms of this trade-off.

The **area under the ROC curve (AUC)** is often given as an overall performance metric that is easy to compare between different methods (1 = perfect classifier)

Receiver operating characteristic



Classification metrics exercise

With the **breast cancer** dataset,

Compare the performance of a *tree* and a *logistic regression* for the task of predicting **recurrence**.


Visualise the performance of the two methods (over 5-fold validation) on a ROC curve.

Improving performance

Bias versus variance

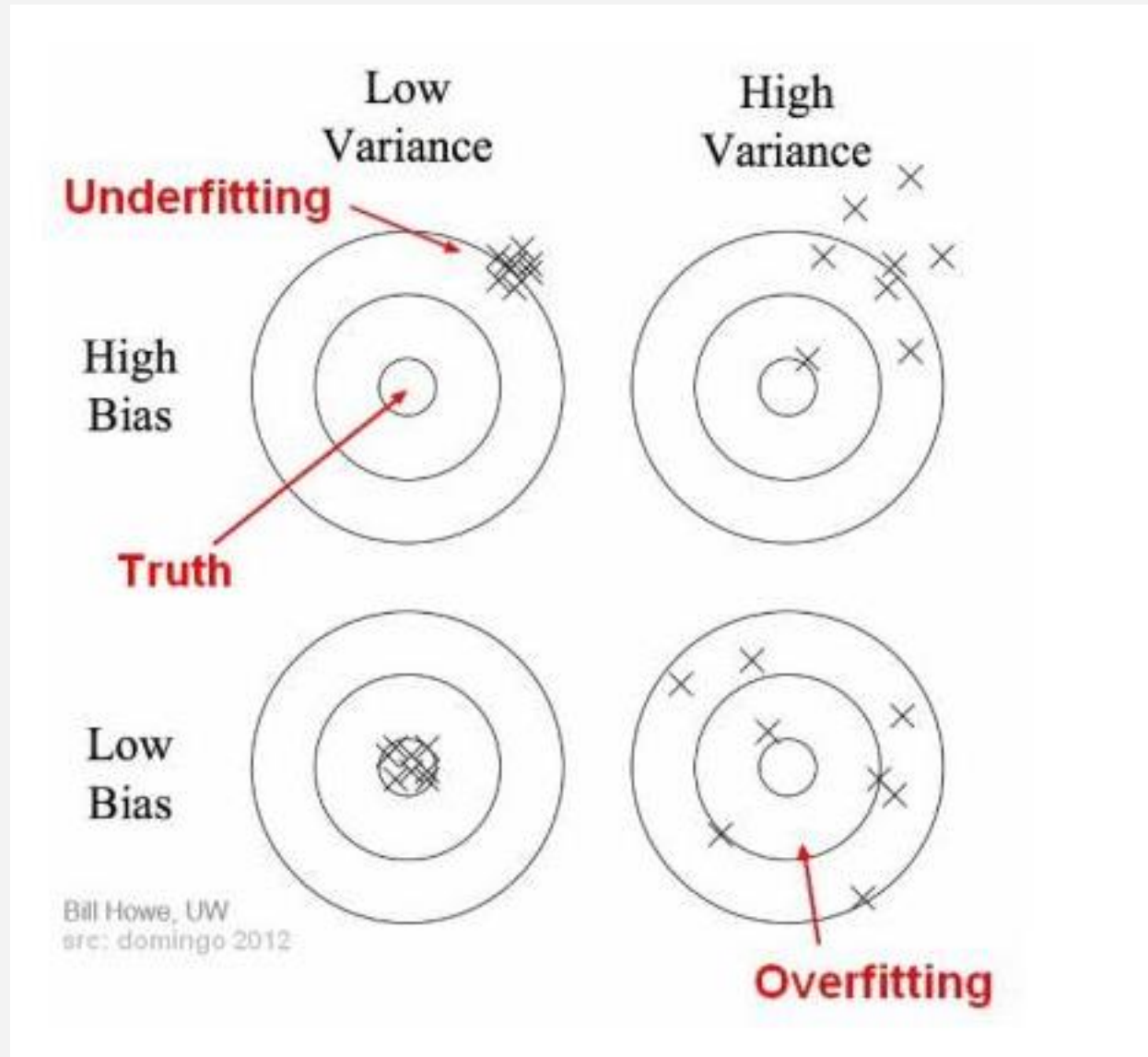
These terms have a special meaning when talking about about ML performance.

Consider a set of regression predictions made from your *testing dataset*. There are two different ways in which these can be wrong:

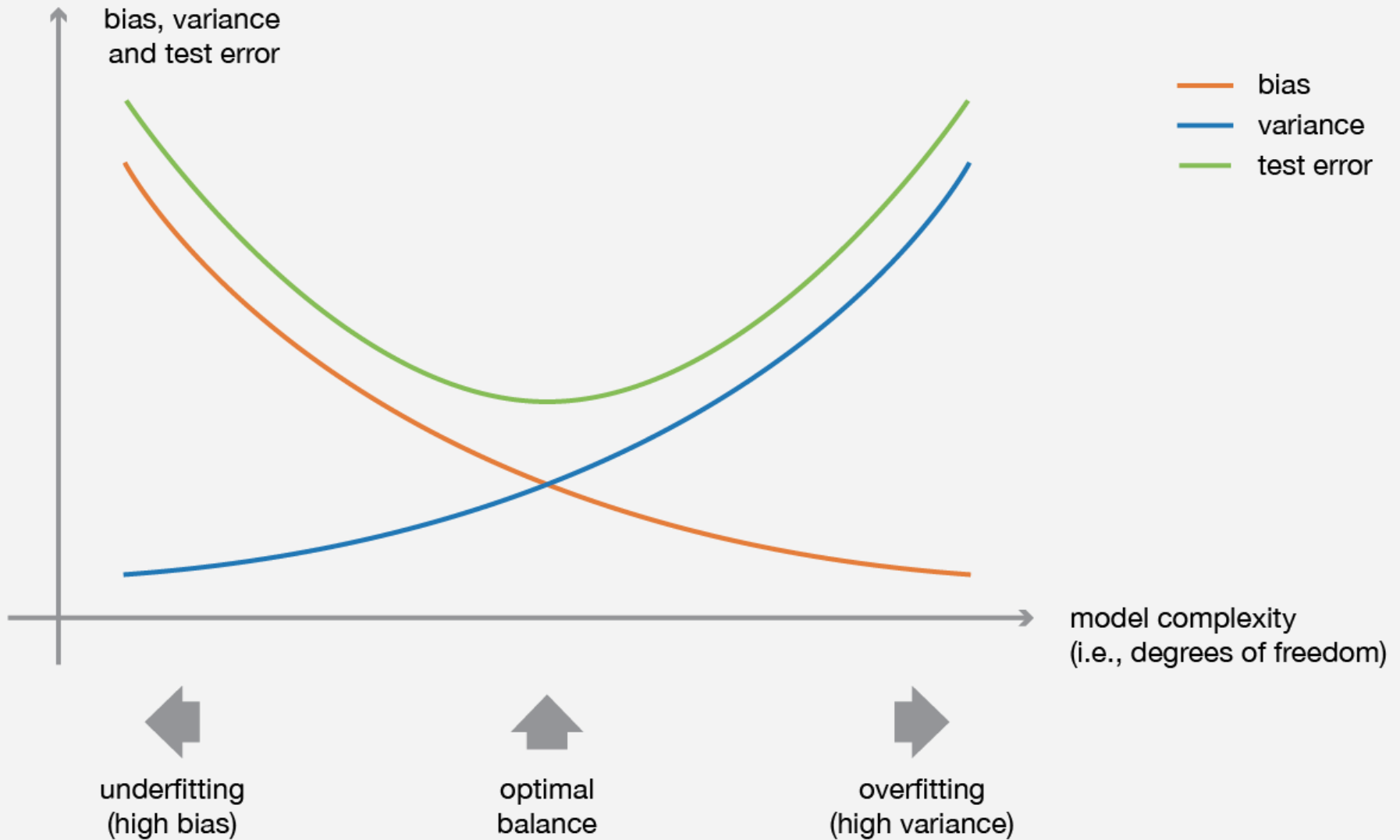
high bias => the model is too simple to describe the training data well (under-fitting), so there is a systematic error in the predictions. 

high variance => the model tries to replicate the training data in too much detail (over-fitting), leading to a lot of noise in the predictions.

Bias versus variance



Bias versus variance



What can we do to improve performance?

What can we do to improve performance?

A larger training dataset (if available) will reduce bias.

Feature selection tries to eliminate uninformative features. This makes models **simpler and less prone to over-fitting**.

Tree pruning is used to **reduce the variance** of a decision tree, by limiting the complexity of the model.

Ensemble methods **combine multiple weak learners** (also called *base models*) to make better overall predictions.

Feature selection

In situations with **large p + small n** (i.e. many features but comparatively few data), over-fitting is difficult to avoid.

However, if we can constrain the model to a simpler form, we might be able to reduce its variance. There are many methods for *feature selection* or *dimensionality reduction* that can help. For some learning methods, *regularisation* can be applied to achieve the same thing.

Intuitively, a good set of informative features would all be strongly correlated with the target, but uncorrelated with each other.

Feature selection example

With the **HDI** dataset,

Using *linear regression* to predict **HDI**, find the R^2 under 5-fold cross-validation.

Now using the *correlations* widget, choose three features that appear to be informative of **disease state**.

Using *select columns*, make a new model using only these three features and compare its performance to your original model.

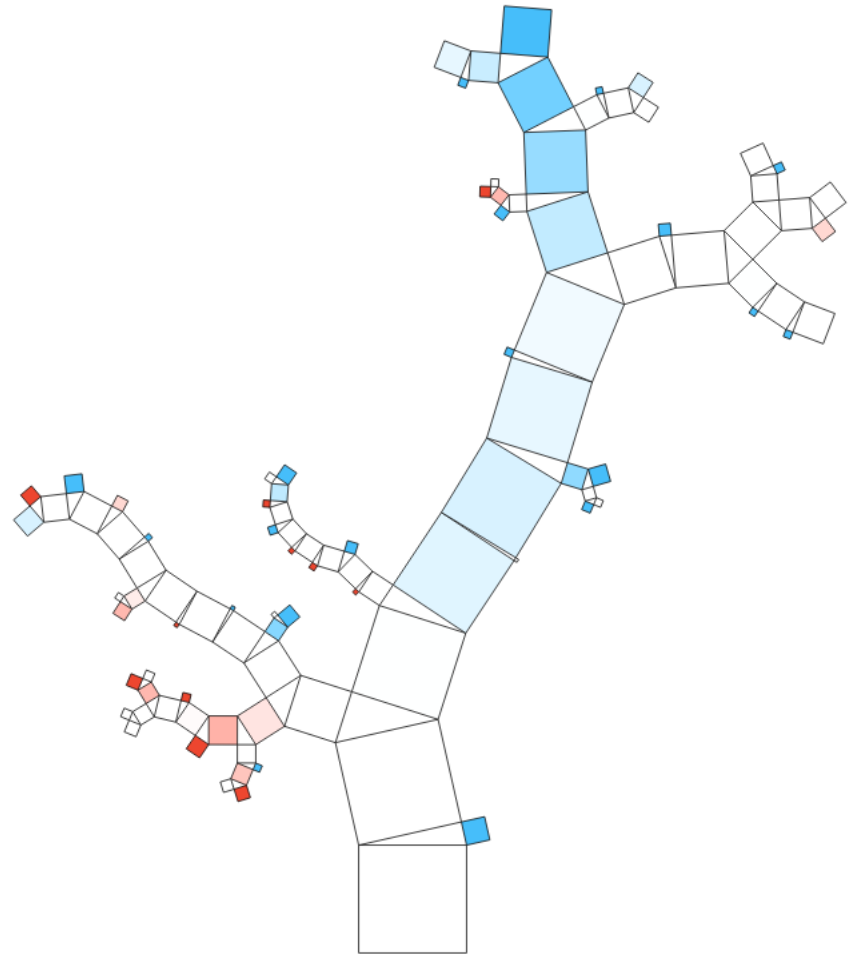
Compare the *bias* and *variance* of the two models visually with scatter plots.

As an alternative to feature selection, investigate how *lasso regression* could be used to force the coefficients for uninformative features to zero.

Tree pruning

Left unconstrained, a decision tree will be able to correctly classify every instance in the training data.

This results in a very complex model, with a large **depth** (number of splits before reaching a leaf node).



Tree pruning

A deep model is very likely to be overfitted, so will have high variance.

By controlling the maximum tree depth, or pruning the last few splits, we can simplify the model to reduce its variance.

Tree pruning exercise

With the **breast cancer** dataset,

Evaluate the performance of a *tree* for the task of predicting **recurrence**.

Can you improve performance by limiting the depth of the tree?

Ensemble methods

In an ensemble method, we combine multiple weak learners to produce an overall strong learner. Usually the component models are all of the same type.

We will need to choose
the base model, and
the way in which models are combined
here we will consider **bagging** and **boosting**.

Bagging

“Bagging” refers to **bootstrap aggregating**, i.e. a way to train many different versions of a model *in parallel*, then combine them to (hopefully) generate an ensemble model that has less variance than each component model.

Bootstrap sampling just means to sample data with replacement. This is a way to generate multiple datasets with similar properties to the original training data.

Once trained, the overall prediction of the ensemble on the testing data is obtained by majority voting.

Random forest

A very popular bagging method is called **random forest**.

This is an ensemble of *trees*, each built from a different **bootstrap sample** from the training data.

At each split in the tree-building process, the algorithm chooses a **random subspace of features** to consider. This ensures that each tree is sufficiently different from the others.

Trees vs forest example

With the **breast cancer** dataset,

Set aside test and validation datasets.

Use *data samplers* to create three bootstrap datasets from the remaining training data.

Train three corresponding *trees* to predict **recurrence** on the validation data and compare results. Would an ensemble model with majority vote improve performance?

Use the *random forest* to explore what happens when we introduce the random subspace method and supply more trees to the ensemble.

Boosting

In contrast to the parallel training performed in bagging, “boosting” methods train multiple models *sequentially*. The aim is to produce an ensemble model that is *less biased* than the base model.



The basic idea is that each model in the sequence gives more importance to the training examples that were poorly predicted by the model before.

AdaBoost is an example of a boosting algorithm, which is often used with *decision stumps* as the base model.

Summary of Part 2

Performance in supervised learning is evaluated using a variety of metrics, using a *test dataset*.

Cross-validation can be used to assess models during development.

Performance is affected by both **bias** and **variance**.

Feature selection is one way to reduce variance when there are many uninformative features.

Bagging and **boosting** are examples of **ensemble methods**, which combine multiple **weak learners** to improve performance.

Next time...

What are **neural networks** and how do they work?

How does **deep learning** differ from the supervised methods we have seen so far?