Current Students

Copyright and UWA unit content

Is it OK to download and share course material such as lectures, unit outlines, exam papers, articles and ebooks?

UWA is committed to providing easy access to learning material and many of your lectures are available for online access via the Lecture Capture System (LCS), accessible through the LMS. Your unit coordinator may make their lecture recordings available to download if they wish. You are allowed to access recorded lectures in the format they are supplied on the LCS – so if they are not made available to download, you must not use any software or devices to attempt to download them.

All recorded lectures and other course material, such as presentation slides, lecture and tutorial handouts, unit outlines and exam papers, are protected under the Copyright Act and remain the property of the University. You are not allowed to share these materials outside of the LMS – for example, by uploading them to study resource file sharing websites or emailing them to friends at other universities. Distributing course material outside of the LMS is a breach of the University Policy on Academic Conduct and students found to be sharing material on these sites will be penalised. University data, emails and software are also protected by copyright and should not be accessed, copied or destroyed without the permission of the copyright owner.

Other material accessible from the LMS or via the Library, such as ebooks and journal articles, are made available to you under licensing agreements that allow you to access them for personal educational use, but not to share with others.

Can I share my login details?

No! Pheme is your key to accessing a number of UWA's online services, including LMS, studentConnect, UWA email, your Library account, and Unifi. These services hold copyright material as well as your personal information, including your unit marks, enrolment information and contact details, so it is important that you do not share the access credentials with anyone else.

https://www.student.uwa.edu.au/learning/resources/ace/respect-intellectual-property/copyright-and-uwa-unit-content

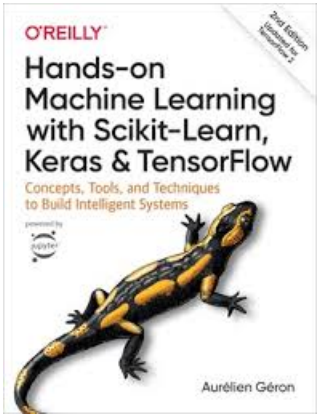
CITS5508 Machine Learning
Overview of a Machine Learning project

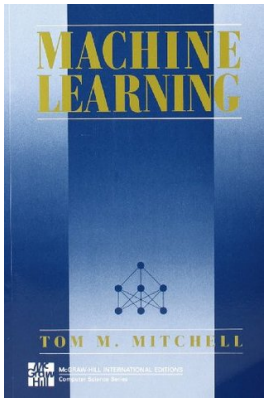
Débora Corrêa (Unit Coordinator and Lecturer)

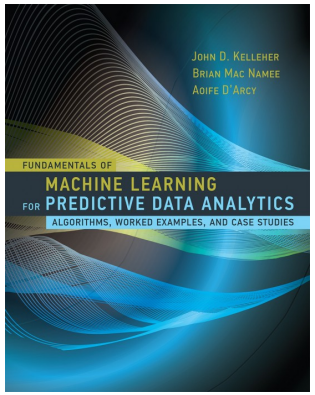
2024

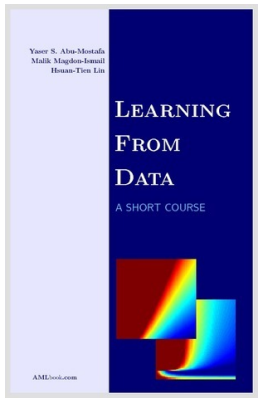
Today

Chapter 2 and chapter 3 of Aurélien Géron's textbook and complement material.









Today

- Continue our simple learning model
- Overview of a machine learning project
- Types of machine learning systems
- Challenges of machine learning

Recap - Learning

Machine learning algorithms automate the task of constructing a model that identifies the relationship between input features and the target variable within a given dataset.

A pattern exists and we have enough data on it.

We have a target function $f : \mathcal{X} \to \mathcal{Y}$. That is, $y = f(\mathbf{x})$.

And we have labelled input-examples in the form $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$.

The model selects $g \approx f$ from the hypothesis set $\mathcal{H}$.

Recap - Simple Learning Model

Our simple model is a weighted sum of features combined to form a score, and the result is compared to a threshold value:

$$h(\mathbf{x}) = sign\left(\left(\sum_{i=1}^{m} w_i x_i\right) + b\right) \quad or \quad h(\mathbf{x}) = sign\left(\left(\sum_{i=1}^{m} w_i x_i\right) + w_0\right)$$

And introduce coordinate $x_0 = 1$:

$$h(\mathbf{x}) = sign\left(\sum_{i=0}^{m} w_i x_i\right)$$

In the vector form,

$$h(\mathbf{x}) = sign\left(\mathbf{w}^T \mathbf{x}\right)$$

Recap - Simple Learning Model

The algorithm will seek the appropriate values for $\mathbf{w}$ based on the examples.

So we need a strategy to update the weights if the algorithm makes incorrect classifications, that is, if $y_i \neq h(\mathbf{x}_i)$.

A simple rule is:

$$\mathbf{w} \leftarrow \mathbf{w} + y_i \mathbf{x}_i$$

At each iteration, pick an example from the set of examples. If it is a misclassified point, update $\mathbf{w}$.

End-to-End Machine Learning Project

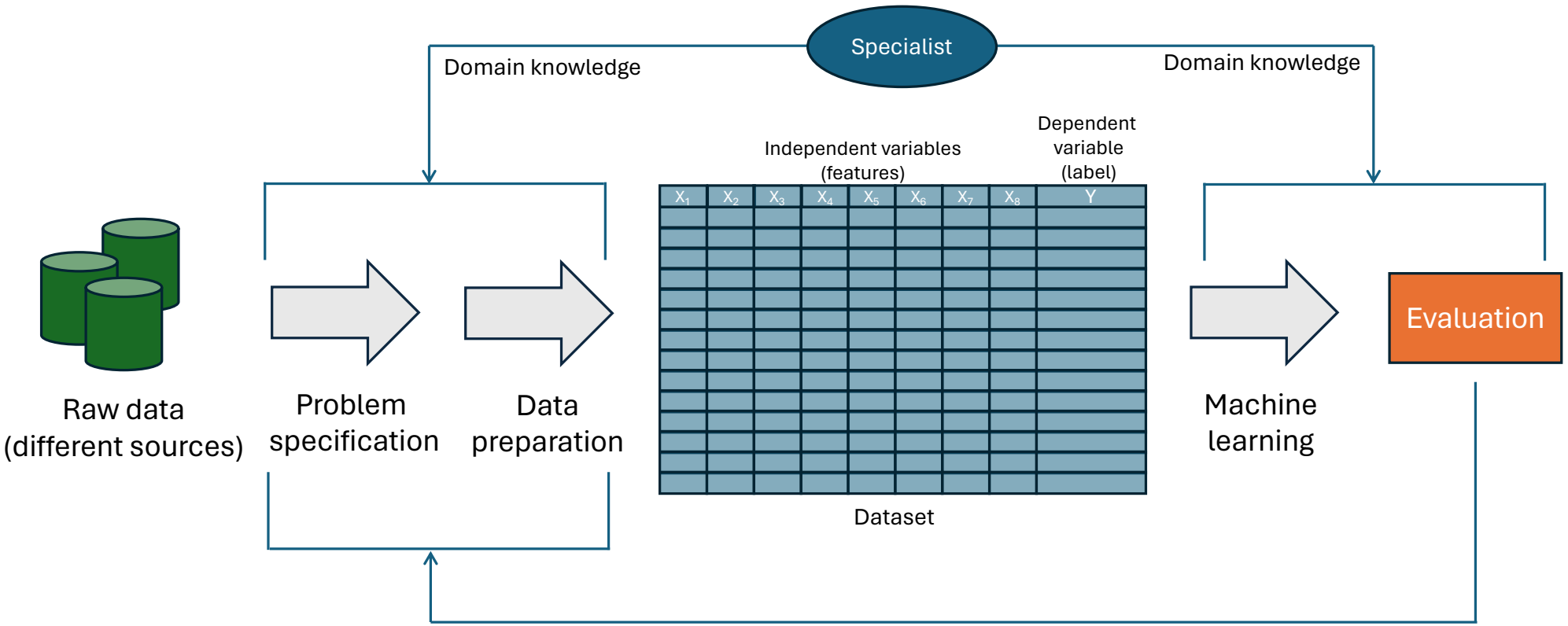
What do you think will is the most important aspect when doing a project in data analysis?

End-to-End Machine Learning Project (cont.)

You are a recently hired data scientist and are given a ML project. What do you need to consider?

1. Understand the problem and check assumptions.
2. Visualise and explore the data (also to support step 1).
3. Prepare the data for a ML algorithm (works in conjunction with step 2).
4. Select a model, train and validate it (can include fine-tuning).
5. Present your solution.
6. Launch, monitor, and keep checking assumptions.

End-to-End Machine Learning Project (cont.)



Problem specification includes business understanding and data understanding.

Frame the problem

What are useful questions here?

Frame the problem

What are useful questions here?

- What are the different data sources available and their types?

- Frame the problem in terms of ML. Is it a supervised task? A regression or classification?

- Are there data quality issues?

- What is the business objective?

- How does the company intend to use the model and benefit from it?

- How does the company approach the problem today?

- How will the interaction with the stakeholders be?

Working with real data

See, for example, text book for a list of places to get large public data sets.

We use a version of some old California housing prices.

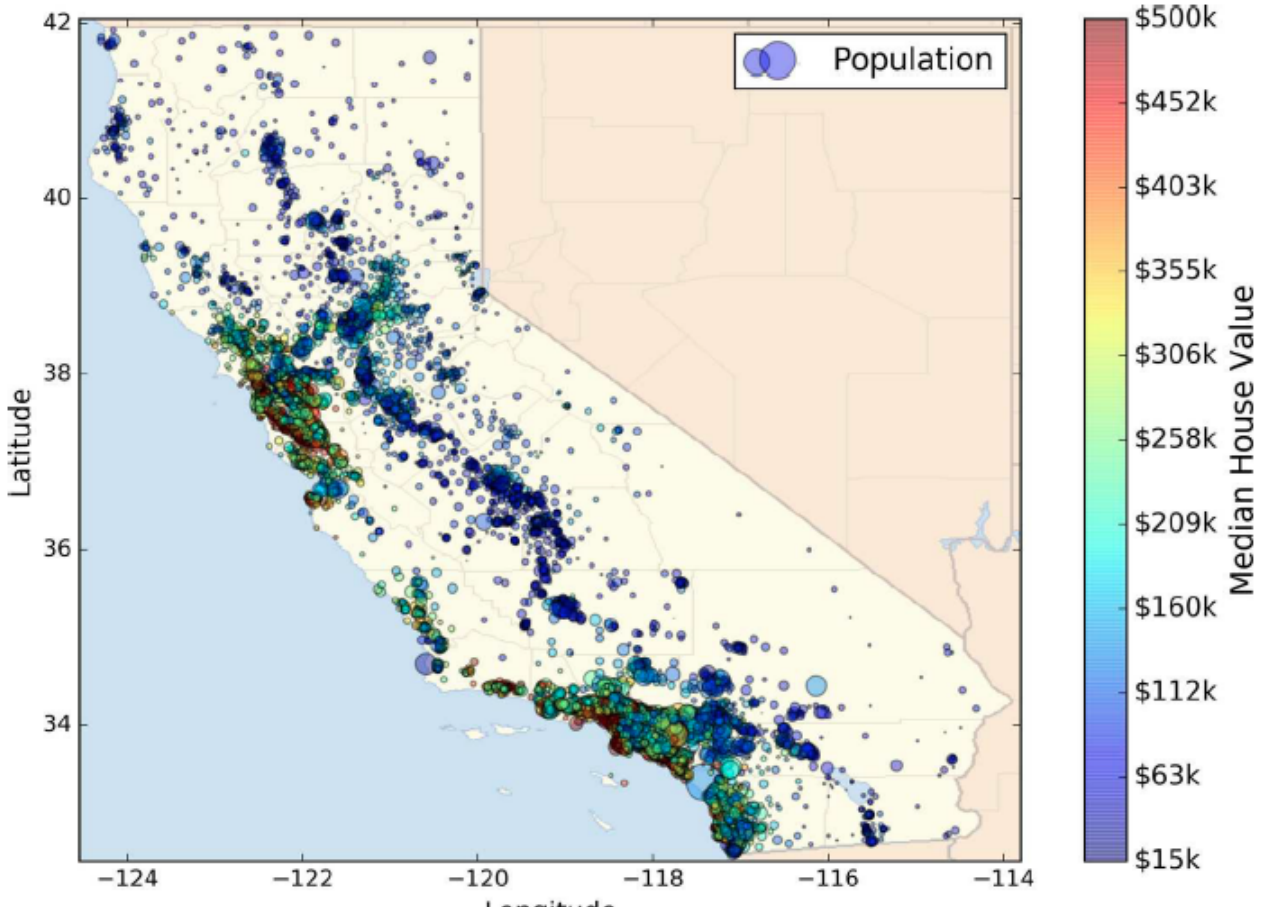


Figure 2-1. California housing prices

How to build the chapter notebook

The final, and very long, notebook can be found from the book's github site
https://github.com/ageron/handson-ml2/blob/master/02_end_to_end_machine_learning_project.ipynb

Go through it yourselves, implementing your own version step by step, while reading the chapter.

We only have time in lectures to look at a few highlights...

Take a quick look at the data structure

In [5]:
```
housing = load_housing_data()
housing.head()
```

Out[5]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	populatio
0	-122.23	37.88	41.0	880.0	129.0	322.0
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0
2	-122.24	37.85	52.0	1467.0	190.0	496.0
3	-122.25	37.85	52.0	1274.0	235.0	558.0
4	-122.25	37.85	52.0	1627.0	280.0	565.0

Histograms for each attribute

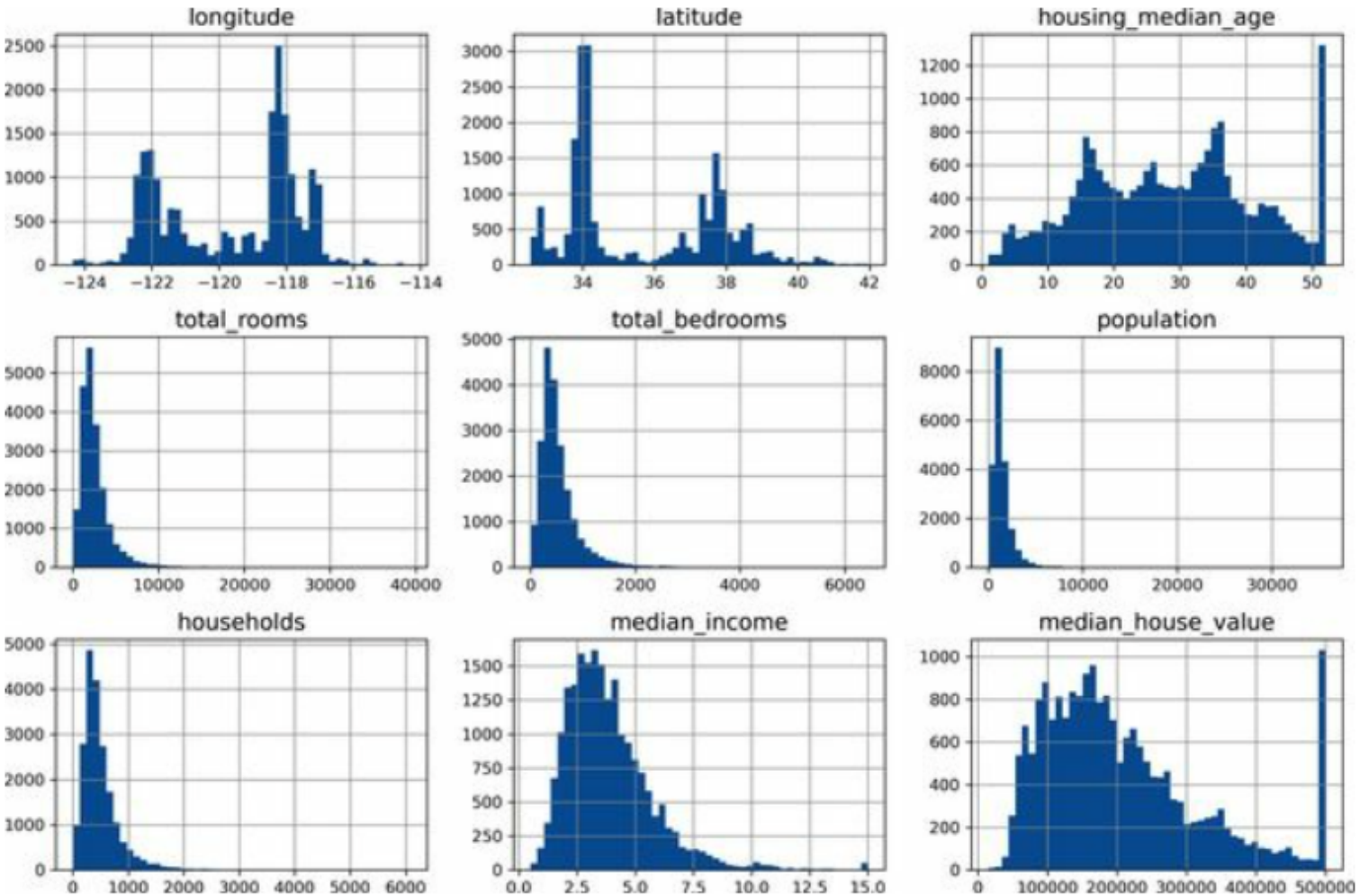


Figure 2-8. A histogram for each numerical attribute

Create a test set

Avoid *data snooping bias*: put the test set away now!
Typically 20% at random.
But be careful that you don't keep resampling.
Also consider stratified sampling.

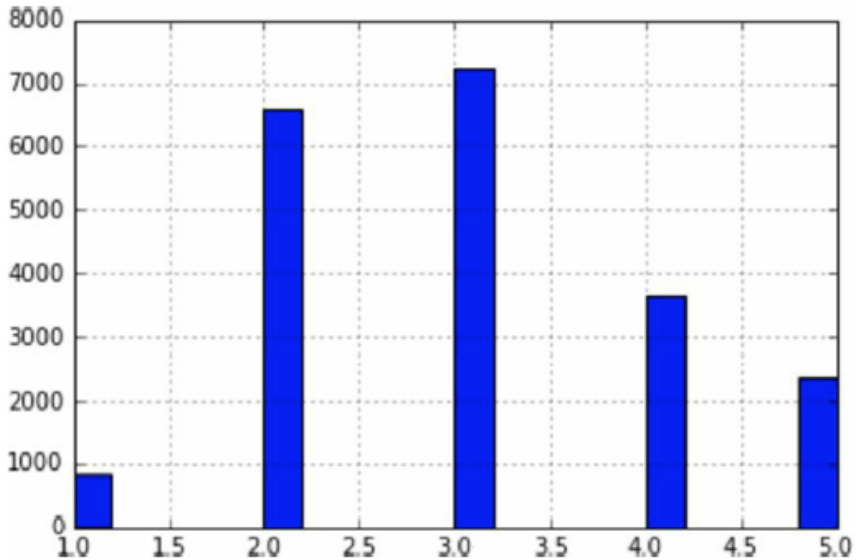


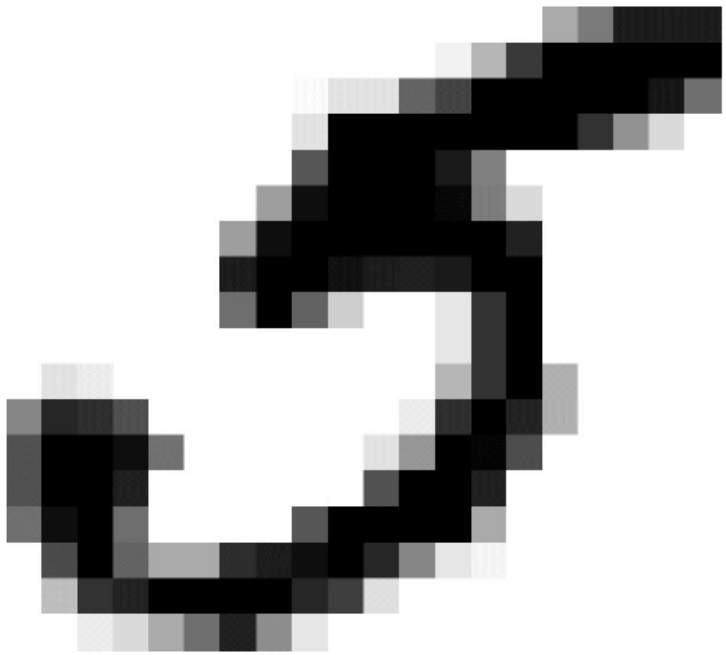
Figure 2-9. Histogram of income categories

Let's have a look at a classification task example

- MNIST: a very well studied dataset
- $70,000$ small images of hand-written digits
- easy to download via Scikit-Learn
- each image has 784 features as it is 28×28 grey-scale pixels
- each target, or label, is $0, 1, ..., 9$
- the text book has a Jupyter notebook for this chapter on its github site.

MNIST example

Easy to view via Matplotlib's `imshow()` function, e.g.,
`plt.imshow(X[36000],...)`



y[36000] is 5.0

More MNIST examples

Figure 3-1. A few digits from the MNIST dataset

Training a binary classifier

E.g., let's make a "5-detector" distinguishing between 5s and not-5s.
Set up the target and task.

```
y_train_5 = (y_train == 5)
y_test_5 = (y_test == 5)
```

Performance strategies: Cross-Validation

- Recall that we can use cross-validation to evaluate a technique.
- We break the training data into *K folds* (= distinct sets). Then for each fold, train a model on all the other folds, and measure the accuracy of prediction on that fold.
- Can use Scikit-Learn's `cross_val_score` method.
- In this example we measure *accuracy* which is ratio of correct predictions.
- With 3 folds we get

```
>>> from sklearn.model_selection import cross_val_score
>>> cross_val_score(sgd_clf, X_train, y_train_5, cv=3, scoring="accuracy")
array([0.96355, 0.93795, 0.95615])
```

Wow! Over 93% accuracy on all folds.

Performance measures: ... but there is a problem with using accuracy

Here is a different classifier on the same data.

```
from sklearn.base import BaseEstimator

class Never5Classifier(BaseEstimator):
    def fit(self, X, y=None):
        return self
    def predict(self, X):
        return np.zeros((len(X), 1), dtype=bool)
```

This one just says that every input is a not-5.
But it is over 90% accurate!!

Why?

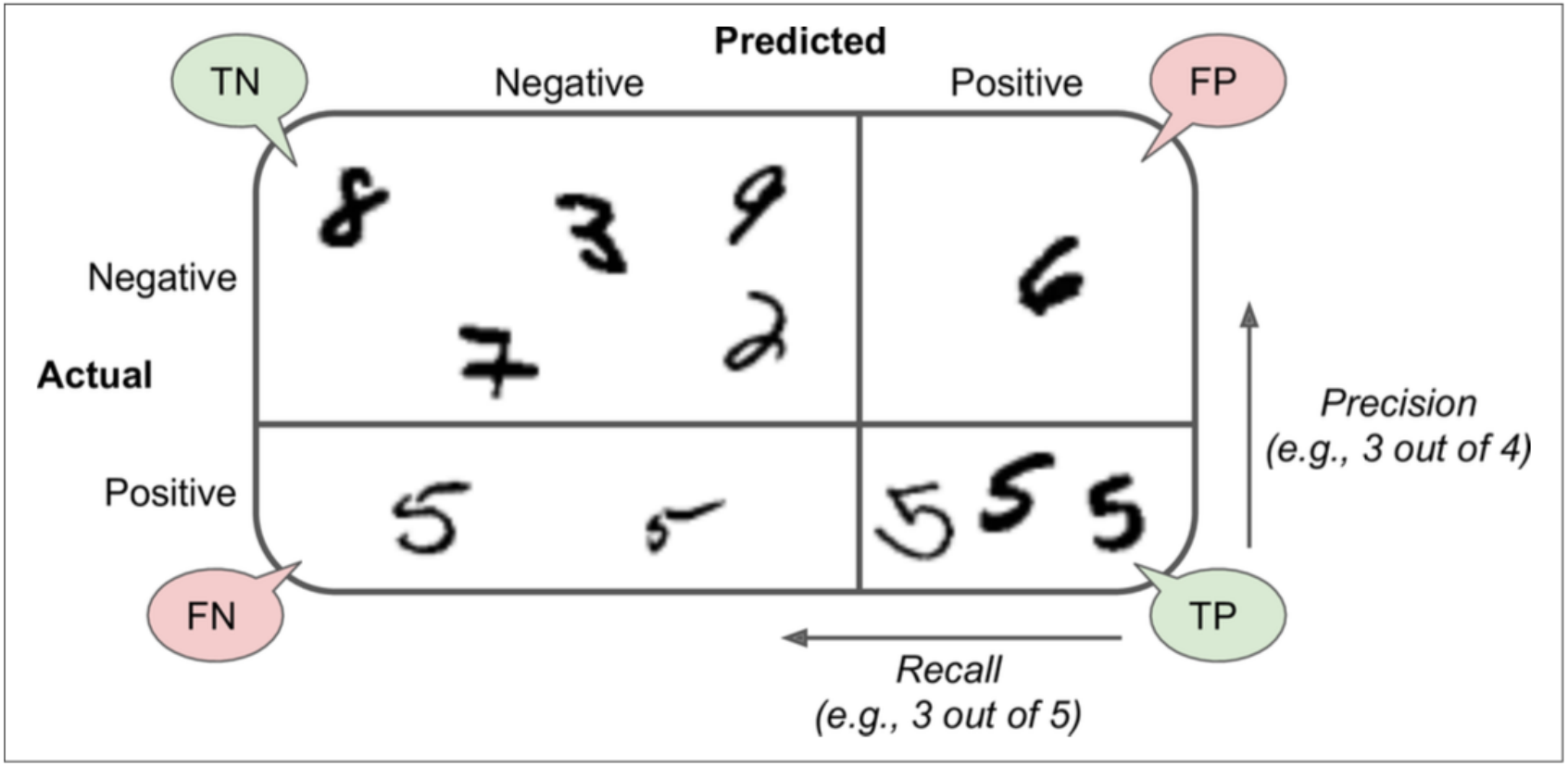
Accuracy has issues with *unbalanced datasets*.

Performance measures: confusion matrix

The general idea is to count the number of times instances of class A are classified as class B.

This gives us the confusion matrix which contains the true negatives, false positives, false negatives and true positives.

Performance measures: confusion matrix



Performance measures: Precision and Recall

We can extract some more concise metrics from the confusion matrix.

$$\text{precision} = \frac{TP}{TP + FP} \quad \text{accuracy of positive predictions } (=0.75)$$

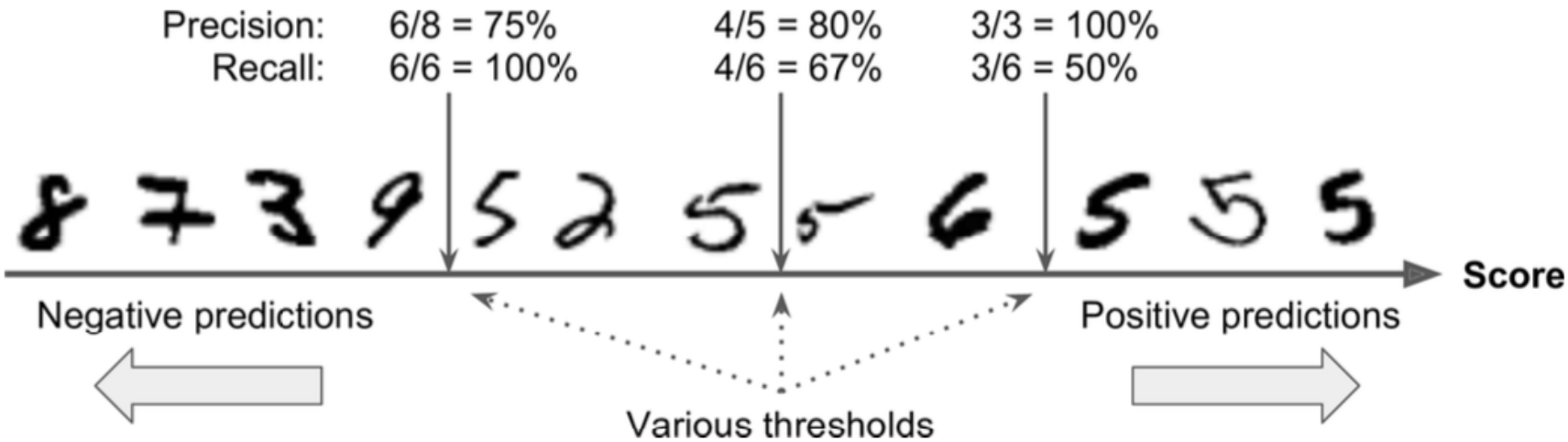
$$\text{recall} = \frac{TP}{TP + FN} \quad \text{true positive rate } (=0.6)$$

$$\text{F}_1 = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}}$$

which is the *harmonic mean* of precision and recall (approx 67% in our example).

Performance measures: Precision/Recall tradeoff

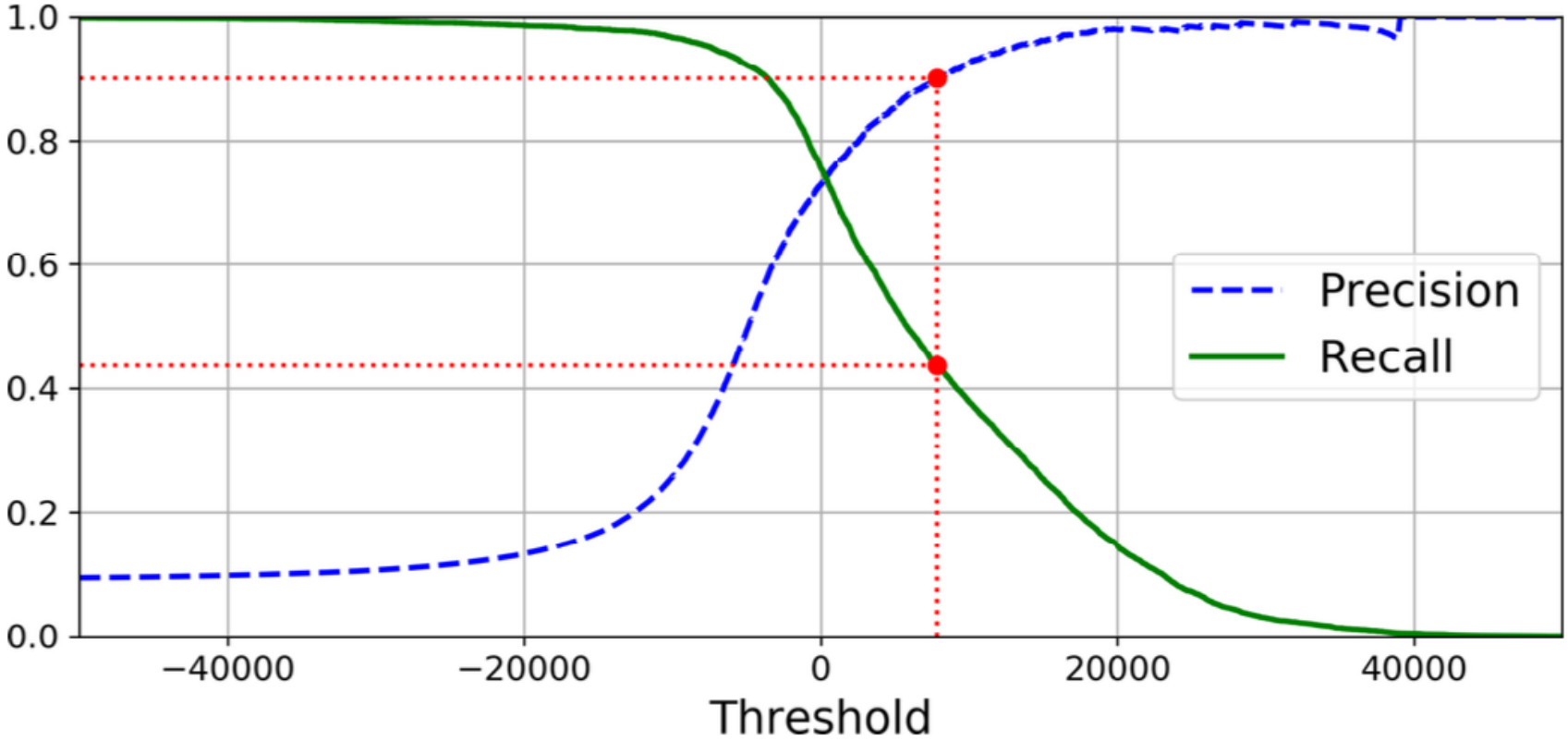
For some applications you may want higher precision or higher recall. But *you can not have both* ...



This is because classifiers typically use score calculated by a *decision function* and a *threshold* applied to the score.

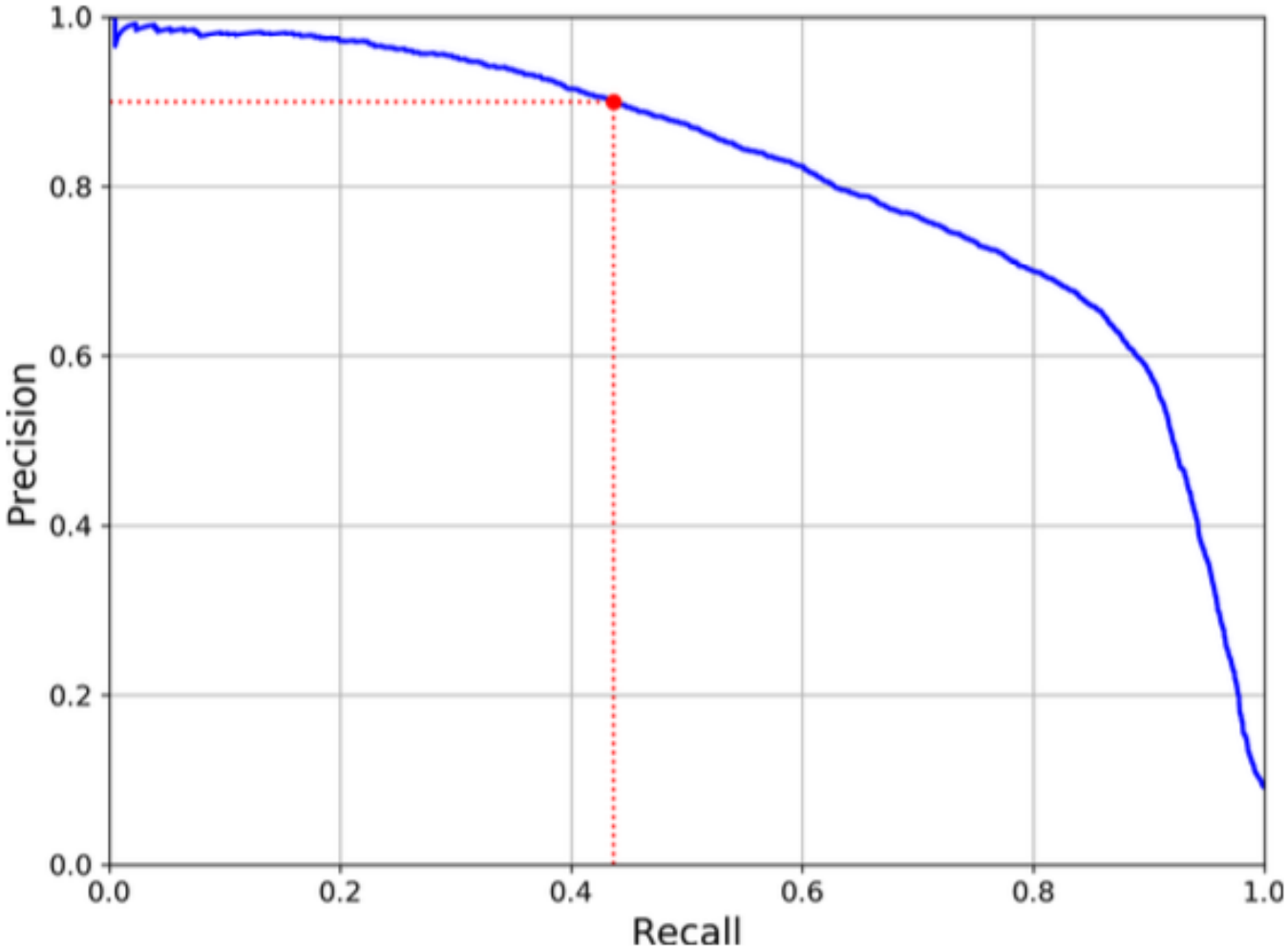
Performance measures: Precision/Recall tradeoff

You can get the score via the `decision_function()` method and use your own threshold.



Performance measures: Precision/Recall tradeoff

Or plot precision against recall (across choices of thresholds).



Performance measures: the ROC curve

Similarly is the *receiver operating characteristic (ROC) curve*, which plots the *true positive rate* (TPR) against the false positive rate (FPR) for varying threshold settings.

Performance measures: the ROC curve

Similarly is the *receiver operating characteristic (ROC) curve*, which plots the *true positive rate* (TPR) against the false positive rate (FPR) for varying threshold settings.

TPR (also known as **sensitivity** and **recall**) = proportion of positive instances that are correctly classified as positives

$= \frac{TP}{TP+FN}$

Performance measures: the ROC curve

Similarly is the *receiver operating characteristic (ROC) curve*, which plots the *true positive rate* (TPR) against the false positive rate (FPR) for varying threshold settings.

TPR (also known as **sensitivity** and **recall**) = proportion of positive instances that are correctly classified as positives

$$= \frac{TP}{TP+FN}$$

TNR (also known as **specificity**) = proportion of negative instances that are correctly classified as negatives

$$= \frac{TN}{FP+TN}$$

Performance measures: the ROC curve

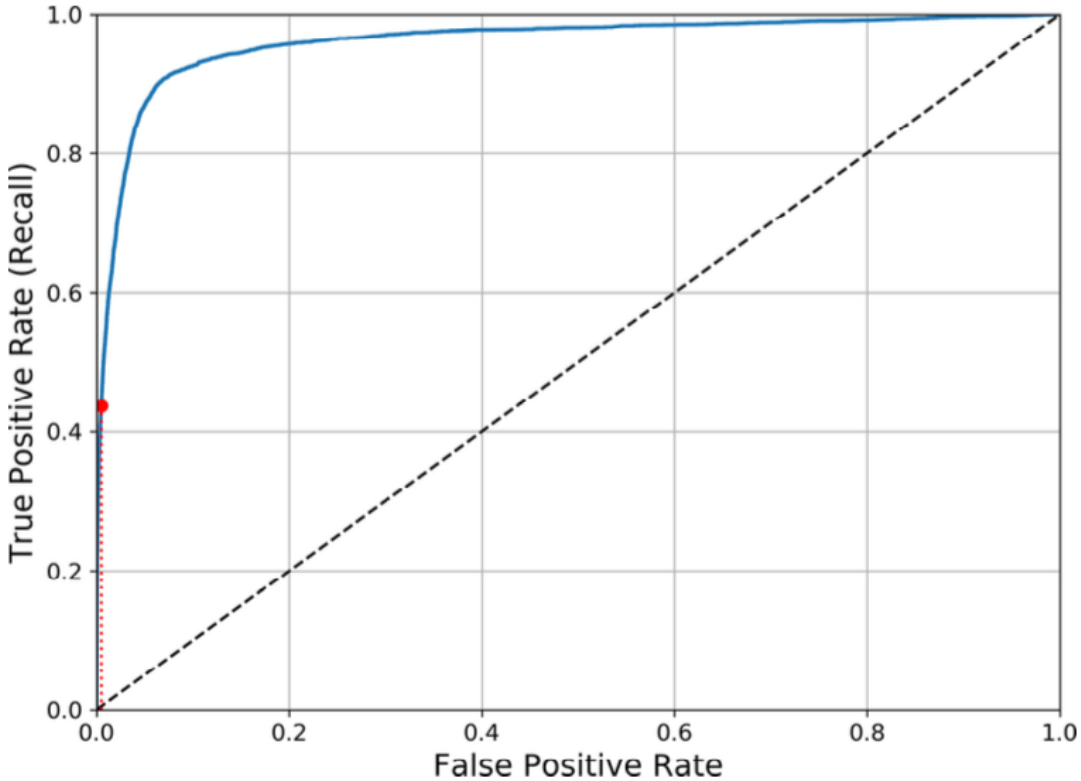
Similarly is the *receiver operating characteristic (ROC) curve*, which plots the *true positive rate* (TPR) against the false positive rate (FPR) for varying threshold settings.

TPR (also known as **sensitivity** and **recall**) = proportion of positive instances that are correctly classified as positives
$= \frac{TP}{TP+FN}$

TNR (also known as **specificity**) = proportion of negative instances that are correctly classified as negatives
$= \frac{TN}{FP+TN}$

FPR = proportion of negative instances that are incorrectly classified as positives
$= \frac{FP}{FP+TN} = 1 - \textbf{specificity}$

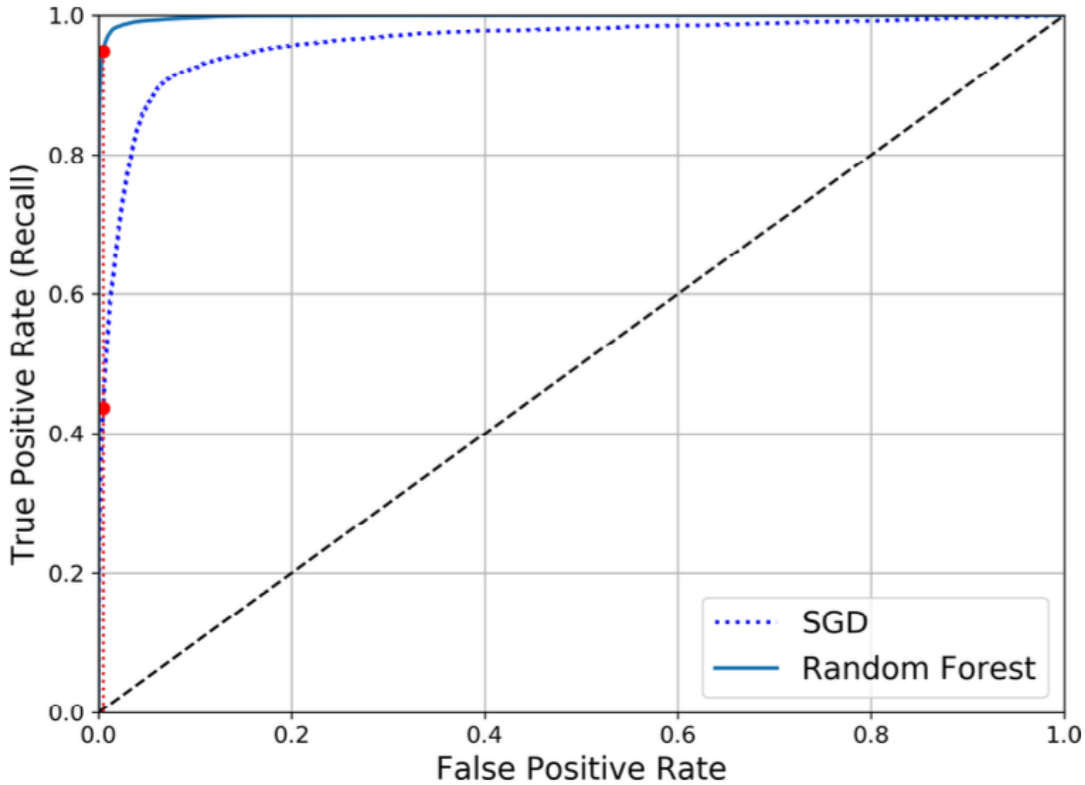
Performance measures: the ROC curve



We plot recall (vertical axis) against $1 -$ specificity (horizontal axis) for the ROC curve.

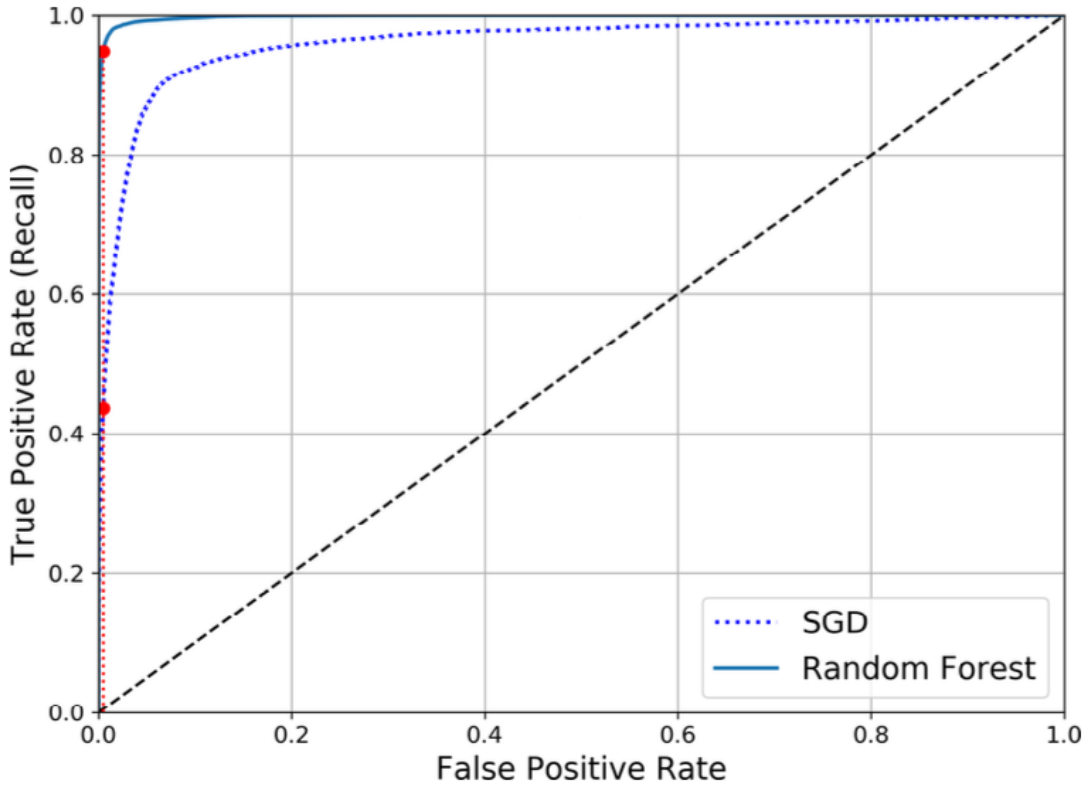
Performance measures: the ROC curve

A good classifier will have a large *area under the curve* (AUC).



Performance measures: the ROC curve

A good classifier will have a large *area under the curve* (AUC).



Which of these two algorithms is better?

Instance-based learning vs model-based learning

One more way to categorize Machine Learning systems is by how they generalize. Most Machine Learning tasks are about making predictions. This means that given a number of training examples, the system needs to be able to generalize to examples it has never seen before. Having a good performance measure on the training data is good, but insufficient; the true goal is to perform well on new instances.

There are two main approaches ...

Instance-based learning

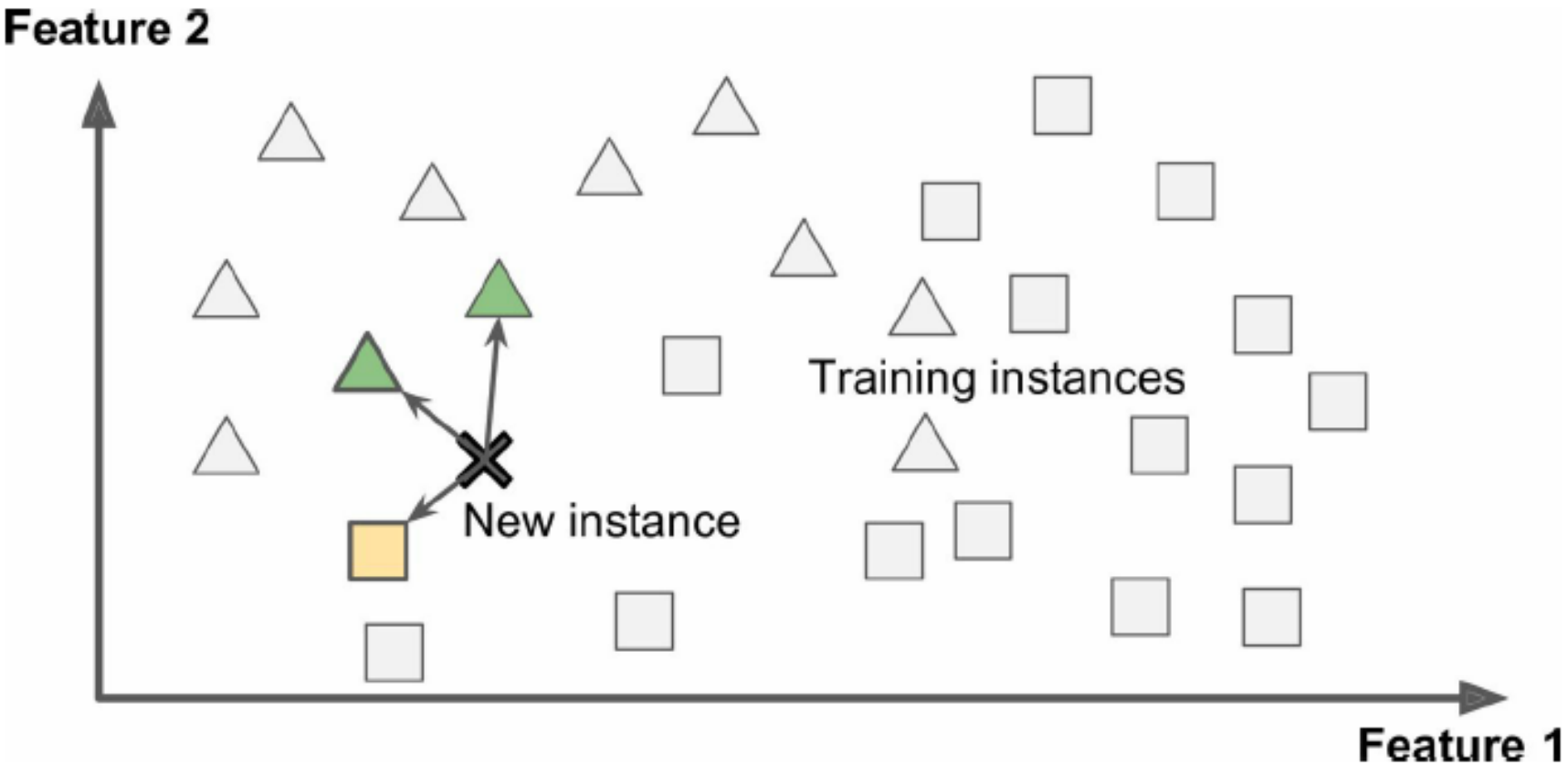


Figure 1-15. Instance-based learning

Model-based learning

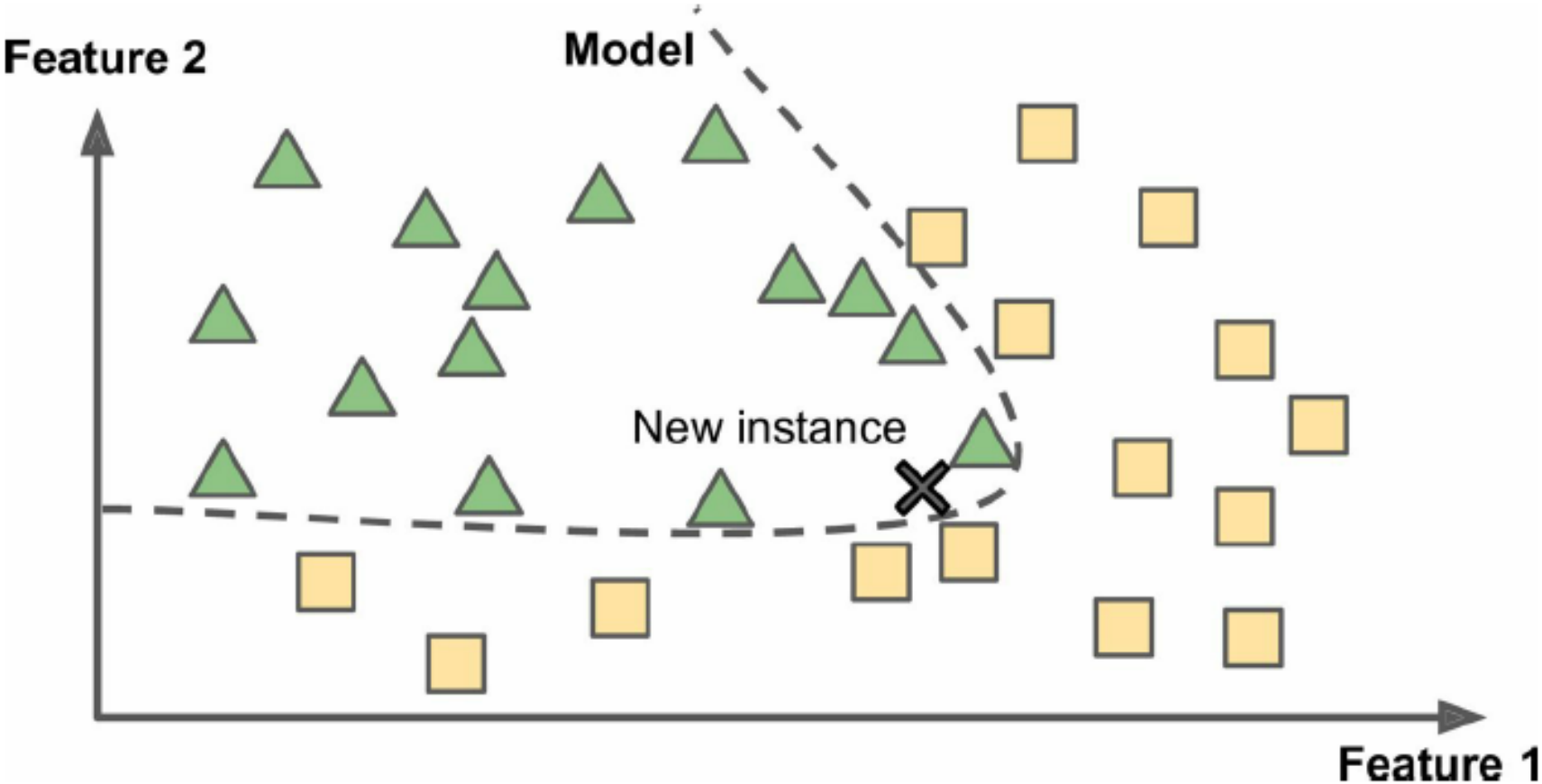


Figure 1-16. Model-based learning

Model-based learning

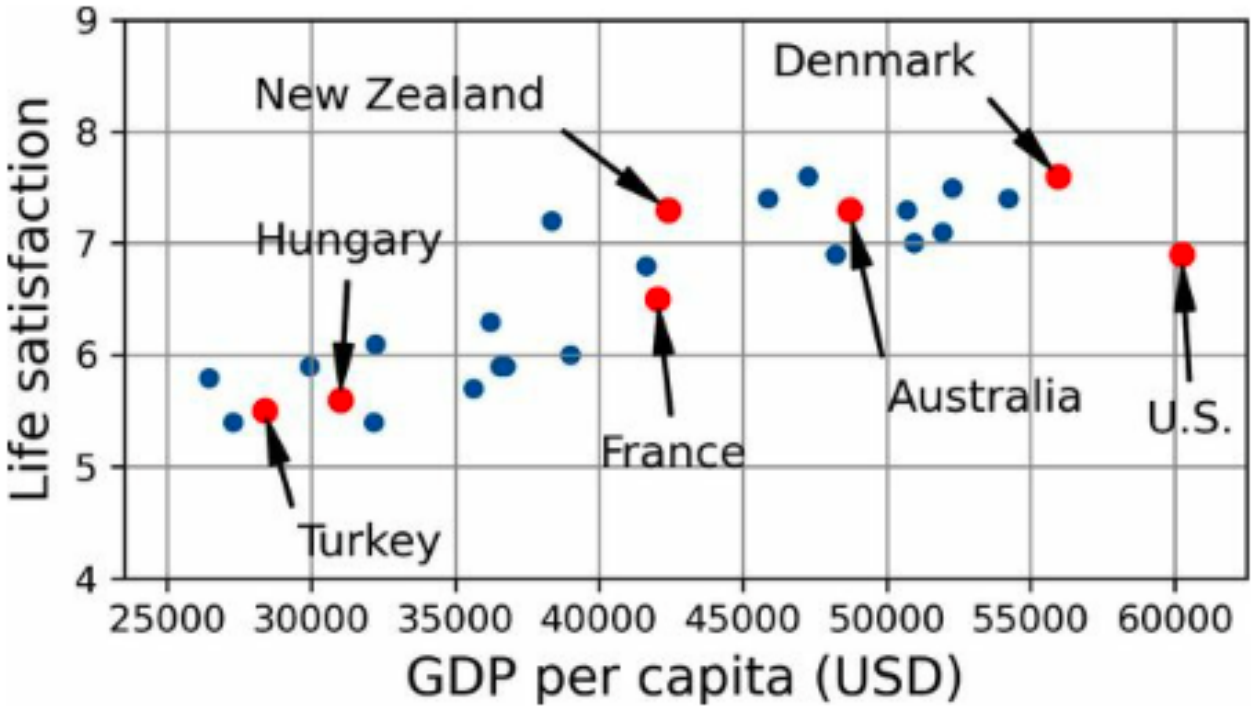


Figure 1-18. Do you see a trend here?

Model-based learning

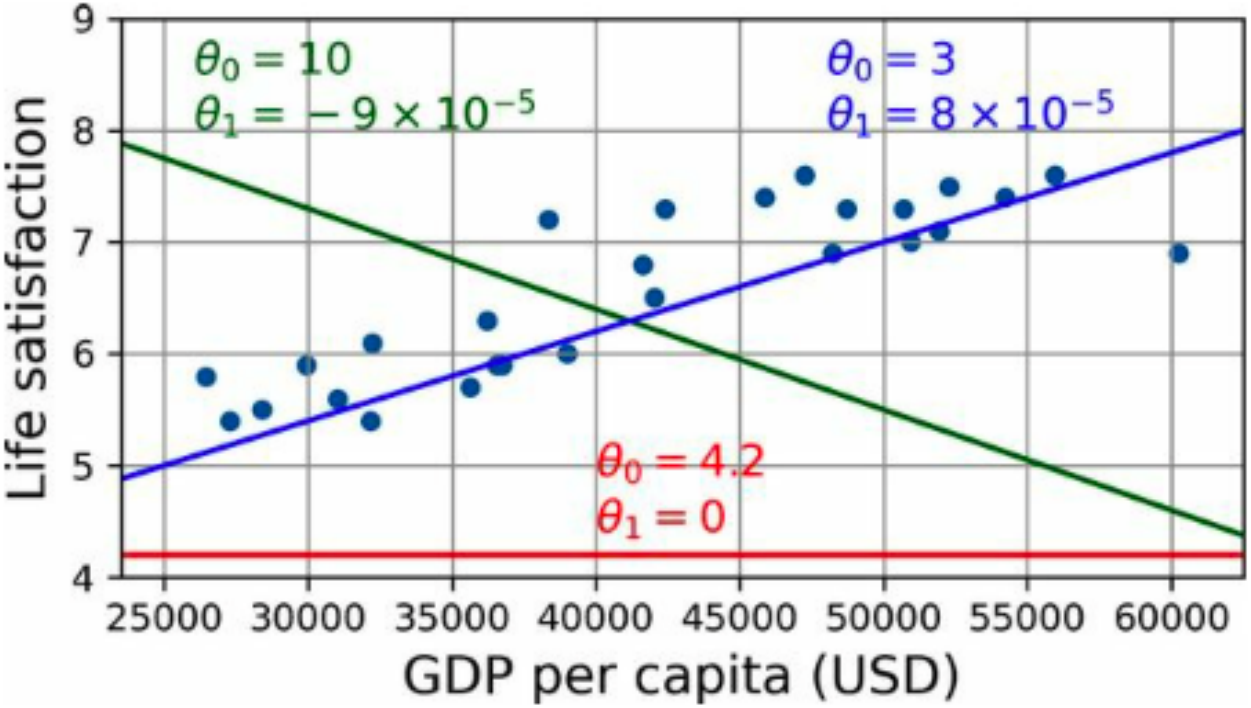


Figure 1-19. A few possible linear models

Model-based learning

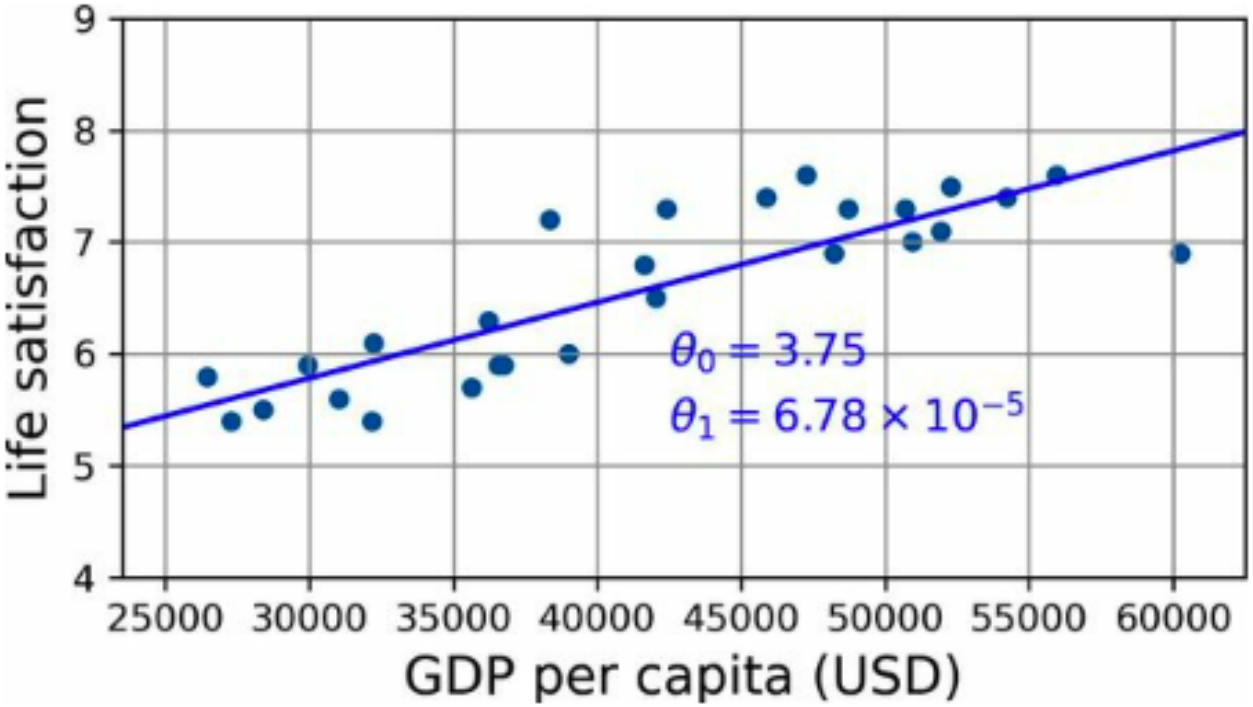


Figure 1-20. The linear model that fits the training data best

Supervised learning

Here are some of the most important supervised learning algorithms (covered in the book):

- k-Nearest Neighbors
- Linear Regression
- Logistic Regression
- Support Vector Machines (SVMs)
- Decision Trees and Random Forests
- Neural networks

Unsupervised learning

The training data is *unlabelled*.

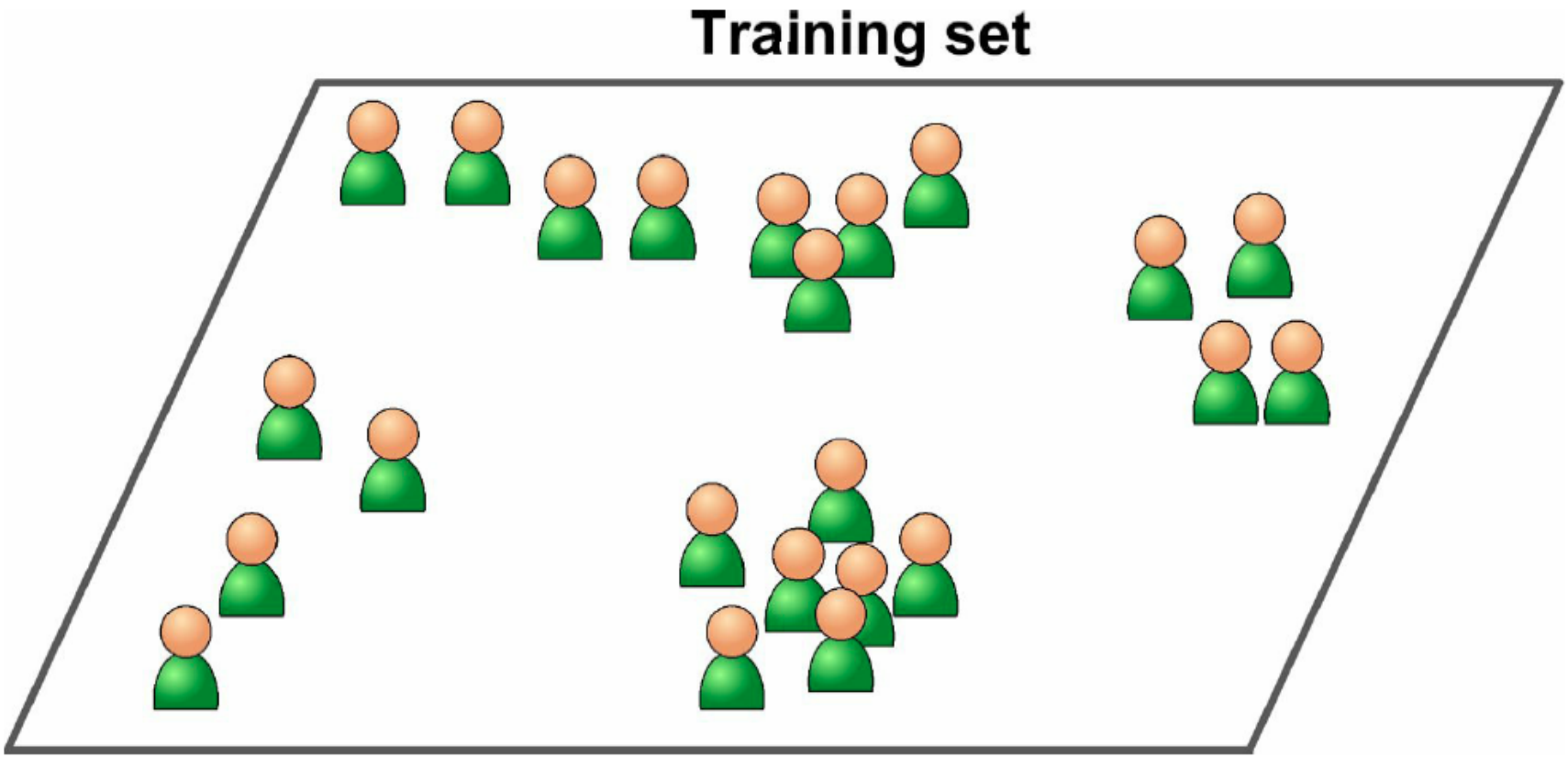


Figure 1-7. An unlabeled training set for unsupervised learning

Unsupervised learning: clustering

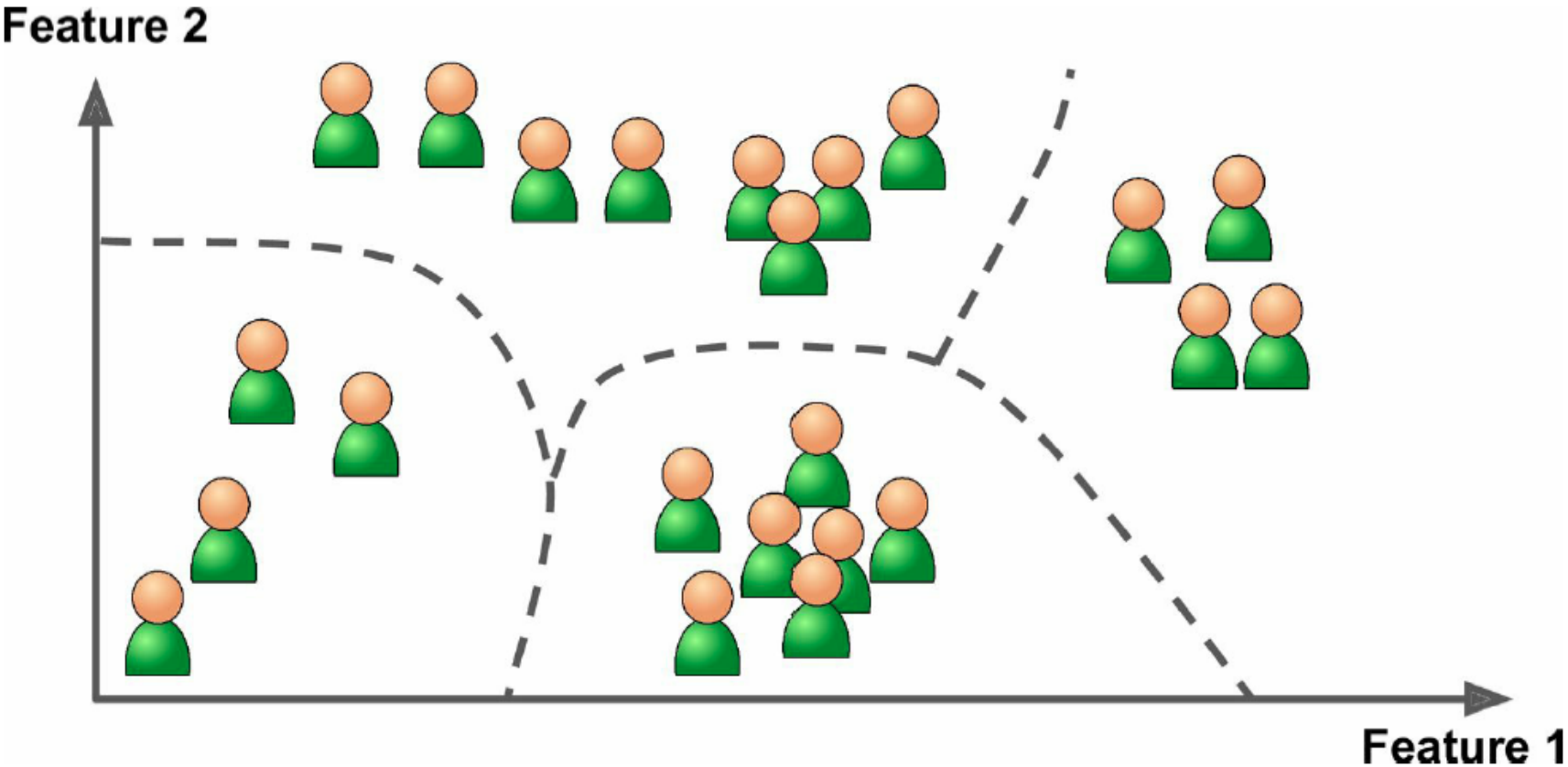


Figure 1-8. Clustering

Unsupervised learning: anomaly detection

Objective is to learn what "normal" data looks like, and then use that to detect abnormal instances.

Figure 1-10. Anomaly detection

Unsupervised learning

Here are some of the most important unsupervised learning algorithms:

- Clustering
  - k-Means
  - DBSCAN
  - Hierarchical Cluster Analysis (HCA)
- Visualization and dimensionality reduction
  - Principal Component Analysis (PCA)
  - Kernel PCA
  - Locally-Linear Embedding (LLE)
  - t-distributed Stochastic Neighbor Embedding (t-SNE)
- Association rule learning
  - Apriori
  - Eclat

Other types of machine learning systems

Many. Useful to classify them in broad categories based on:

- Whether or not they can learn incrementally on the fly (online learning versus batch learning)
- How they are supervised during training (supervised, semi-supervised, unsupervised, reinforcement learning).
- How they generalise (instance-based, model-based).

These criteria are not exclusive; you can combine them. We talked about some types today. Recommended reading includes studying about the other ones in the textbook.

Main challenges of machine learning

Main challenges of machine learning

- Insufficient quantity of training data
- Non-representative or poor-quality data
- Irrelevant features
- Overfitting or underfitting
- Sampling bias

Non-representative or poor-quality data

Some of the feature values may be incorrect, missing or may be mislabelled.

The training set is a non-representative sample of the possible instances in the domain.

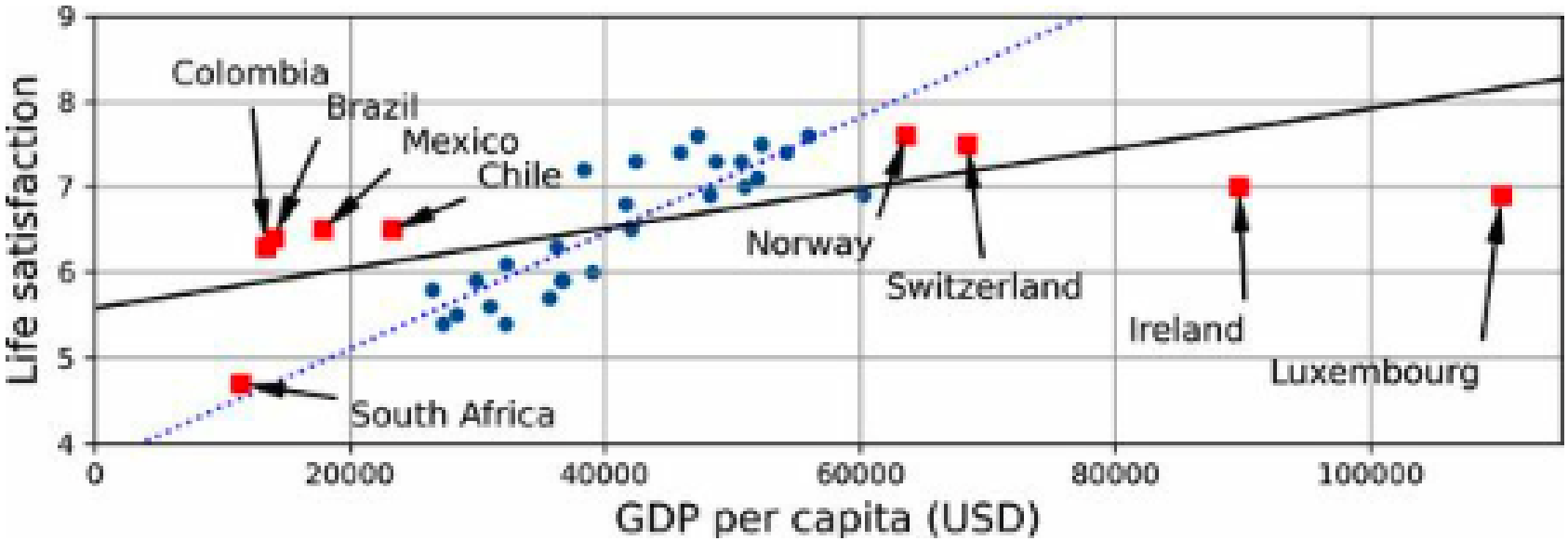


Figure 1-22. A more representative training sample

Non-representative or poor-quality data (cont.)

If your training data is poor, it will make detecting the underlying patterns harder for the system.

Much of the data scientist's effort is to understand the problem from the business perspective and prepare quality data.

You will have to deal with outliers, noise, errors, missing data, poor-quality measurements, incoherent labelling, etc.

Overfitting

The model is so complex that it fits to the dataset too closely and becomes sensitive to noise in the data. Some options:

- Simplify the model by selecting one with fewer parameters, by reducing the number of attributes, or by constraining it.
- Gather more training data.
- Check whether it is sensible to reduce the noise in the training data.

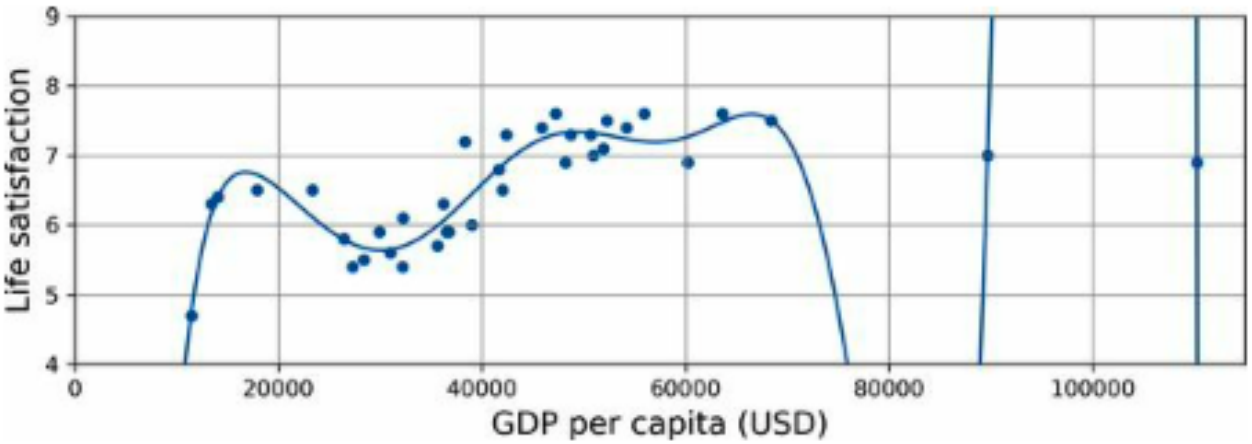


Figure 1-23. Overfitting the training data

Underfitting

Underfitting is the opposite of overfitting: it occurs when your model is too simple to learn the underlying structure of the data. For example, a linear model of life satisfaction is prone to underfit; reality is just more complex than the model, so its predictions are bound to be inaccurate, even on the training examples.
The main options to fix this problem are:

- Selecting a more powerful model, with more parameters.
- Feeding better features to the learning algorithm (feature engineering).
- Reducing the constraints on the model (e.g., reducing the regularization hyperparameter).

Sampling bias: example

US presidential election in 1936: Landon (Republican) against Roosevelt (Democratic).

The Literary Digest conducted a very large poll, sending mail to about 10 million people. It got 2.4 million answers, and predicted with high confidence that Landon would get 57% of the votes. Instead, Roosevelt won with 62% of the votes.

Problems with Literary Digest's sampling method:

- Lists of addresses were tended to favour wealthier people who were more likely to vote Republican.
- Less than 25% of the people who were polled answered.

Any other examples?

For this week

Attend the supervised lab. The Unit Coordinator and/or a casual Teaching Assistant will be there to help.

Set up Python (3.X) on your computer. Write a few simple programs to be familiar with the basic syntax.

Make sure your software is correctly installed and you can run the notebooks of the textbook (chapters 1, 2 and 3). Get familiarized with Jupyter Notebooks.

Read up to Chapter 4 (Training Models).

And that's all for the second lecture.

Have a good week.