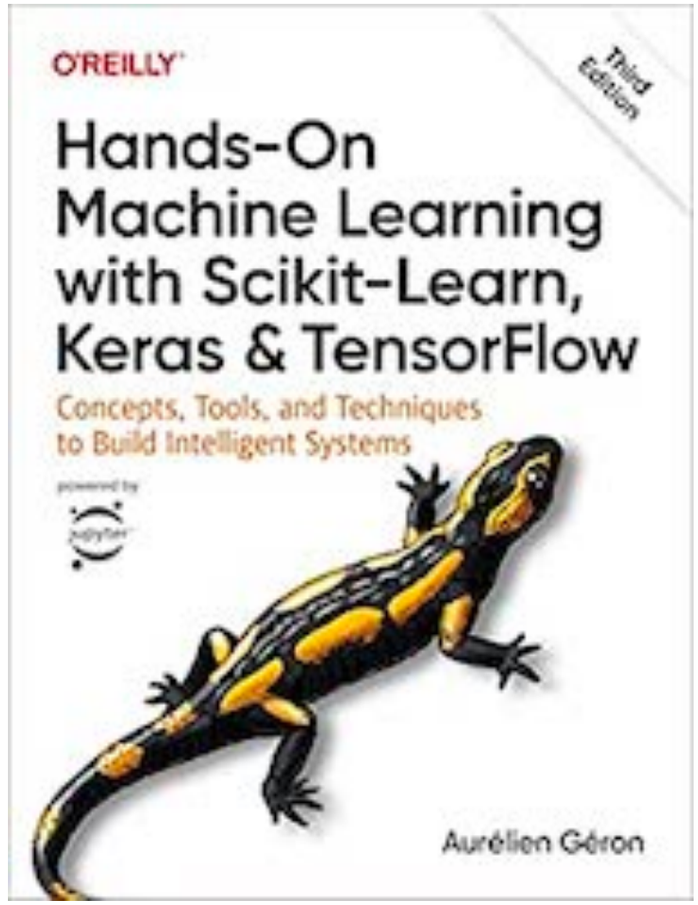
Machine Learning Security

1 The Machine Learning Landscape



Concepts

- Supervised versus unsupervised learning
- Online versus batch learning
- Instance-based versus model-based learning
- Workflow of a ML project
- Challenges
- How to evaluate and fine-tune a ML system

What Is Machine Learning?

What Is Machine Learning?

- The science (and art) of programming computers so they learn from data
- Example: **spam filter**
  - Learns from examples of spam emails, flagged by users, and regular emails (the *training set*)
  - The part of a ML system that learns and makes predictions is called a *model.*
  - Neural networks and random forests are examples of models

Why Use Machine Learning?

Traditional Programming

- To create a spam filter

  1. Examine spam examples and find words that appear often, like "4U", "credit card", "free", "amazing", or other paterns in sender's name or email body

  2. Write an algorithm to detect each pattern, flag email as spam if a number of these patterns are detected

  3. Test the program and repeat steps 1 and 2 until it's good enough to launch

Traditional Programming

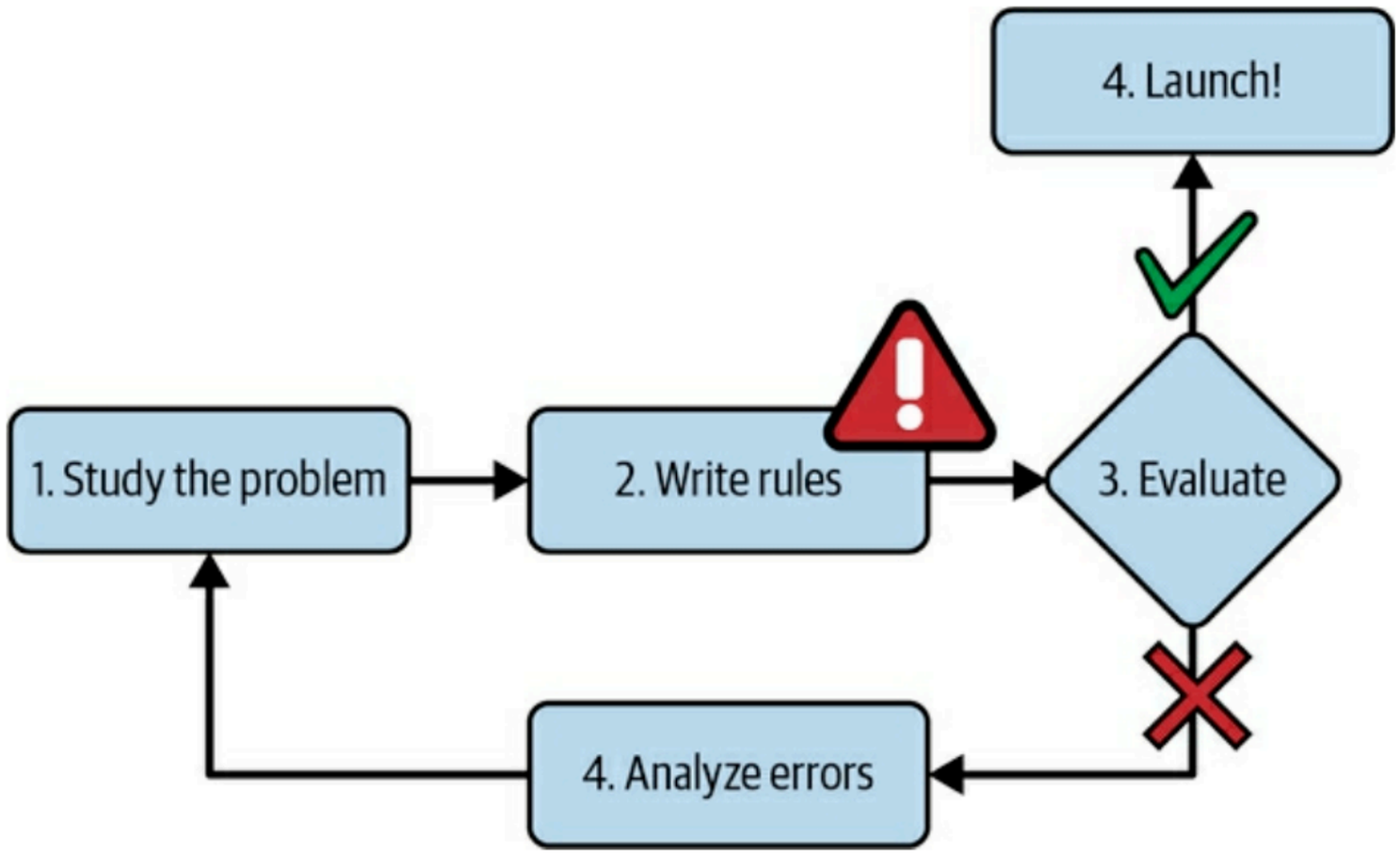


Figure 1-1. The traditional approach

- Program uses a long list of rules
- Difficult to maintain

Machine Learning

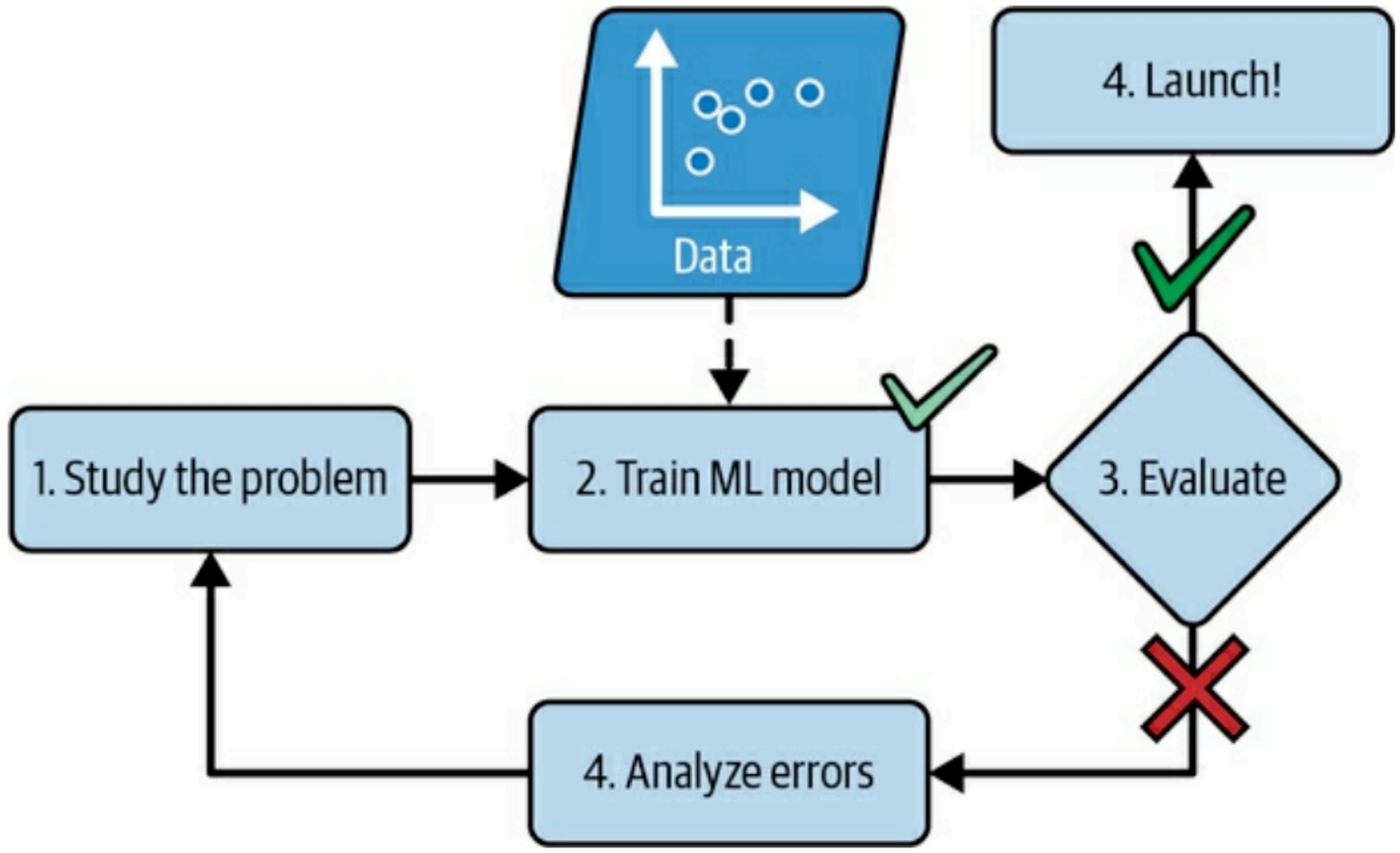


Figure 1-2. The machine learning approach

- Learns words and phrases that can predict spam
- Program is shorter, easier to maintain, and more accurate

Machine Learning

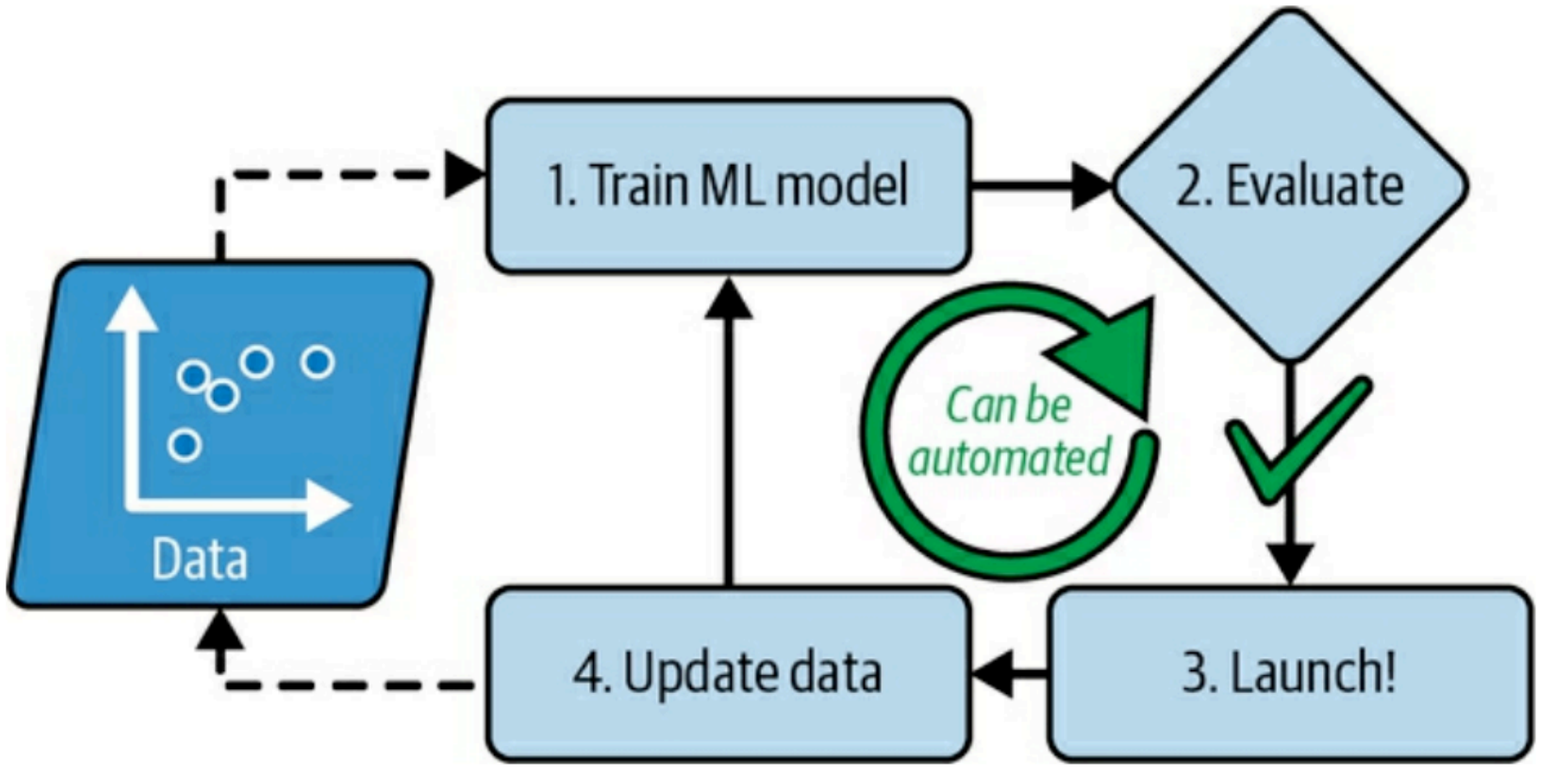


Figure 1-3. Automatically adapting to change

- Automatically adapts to changing tactics of spammers

Other Problems

- Some problems are too complex for traditional approaches
  - Or have no known algorithm
- **Speech recognition**
  - Trained on many example recordings

Helping Humans Learn

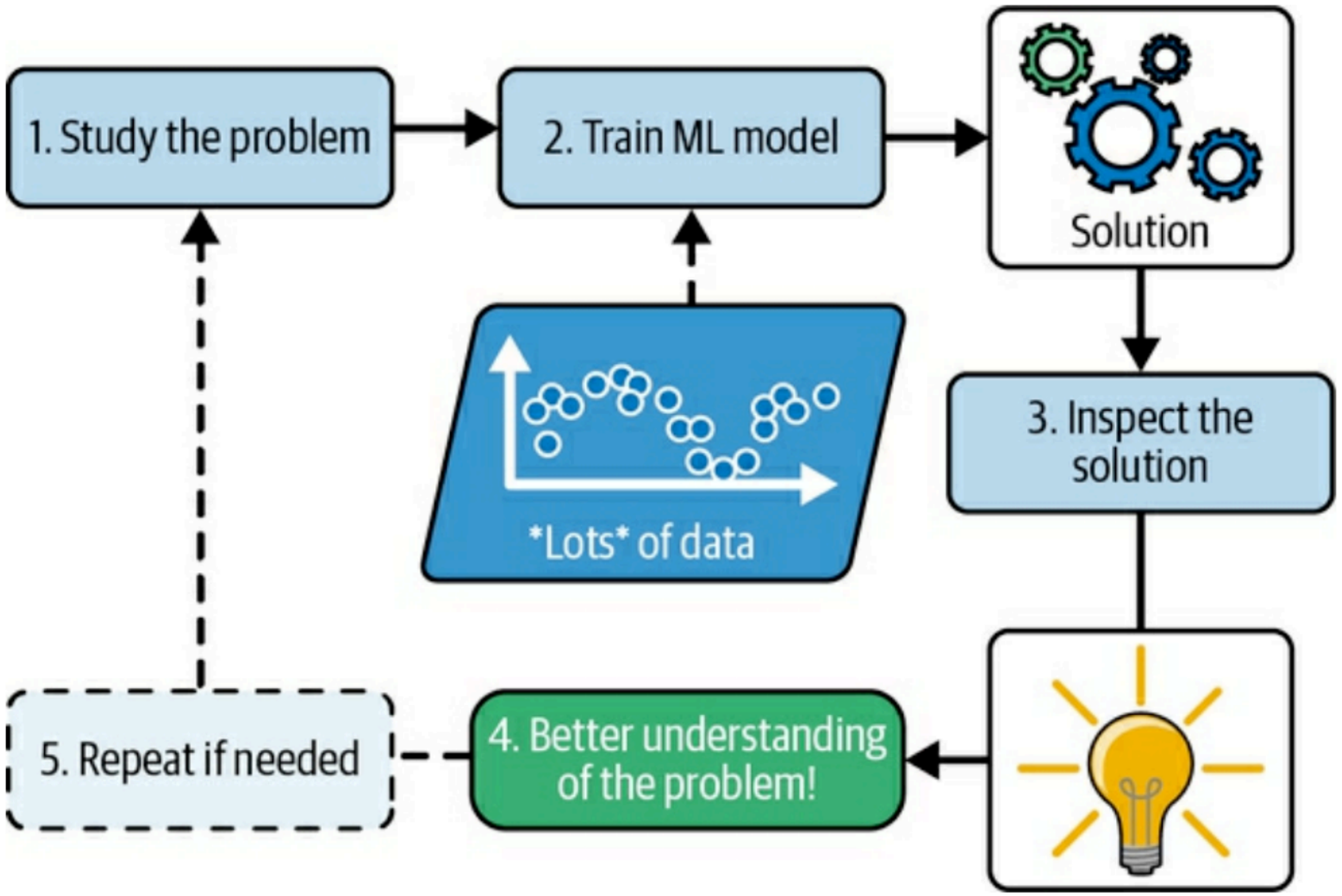


Figure 1-4. Machine learning can help humans learn

Machine Learning is Great For

- Problems for which existing solutions require
  - a lot of fine-tuning, or
  - long lists of rules
- Complex problems for which using a traditional approach yields no good solution
- Fluctuating environments (a machine learning system can easily be retrained on new data, always keeping it up to date)
- Getting insights about complex problems and large amounts of data

Examples of Applications

Examples of Applications

- Analyzing images of products on a production line to automatically classify them
    - Image classification typically uses **Convolutional Neural Networks (CNNs)** or **Transformers**
- Detecting tumors in brain scans
- Automatically classifying news articles
    - This is **Natural Language Processing (NLP)**, and more specifically text classification, which can be tackled using **Recurrent Neural Networks (RNNs)** and **CNNs**, but **Transformers** work even better

Examples of Applications

- Creating a chatbot or a personal assistant
- Forecasting your company's revenue next year, based on many performance metrics
- Making your app react to voice commands
- Detecting credit card fraud
- Segmenting clients based on their purchases so that you can design a different marketing strategy for each segment
- Representing a complex, high-dimensional dataset in a clear and insightful diagram
- Recommending a product that a client may be interested in, based on past purchases
- Building an intelligent bot for a game

Types of Machine Learning Systems

Criteria

- How they are supervised during training
  - **supervised**, **unsupervised**, **semi-supervised**, **self-supervised**, and others
- Whether or not they can learn incrementally on the fly
  - **online** versus **batch** learning
- Whether they work by simply comparing new data points to known data points, or
  - instead by detecting patterns in the training data and building a predictive model
  - much like scientists do
  - **instance-based** versus **model-based** learning

Training Supervision

- Supervised Learning
- Unsupervised Learning
- Self-Supervised Learning
- Semi-Supervised Learning
- Reinforcement Learning

Supervised Learning

- Training data has labels
    - Indicating desired solution

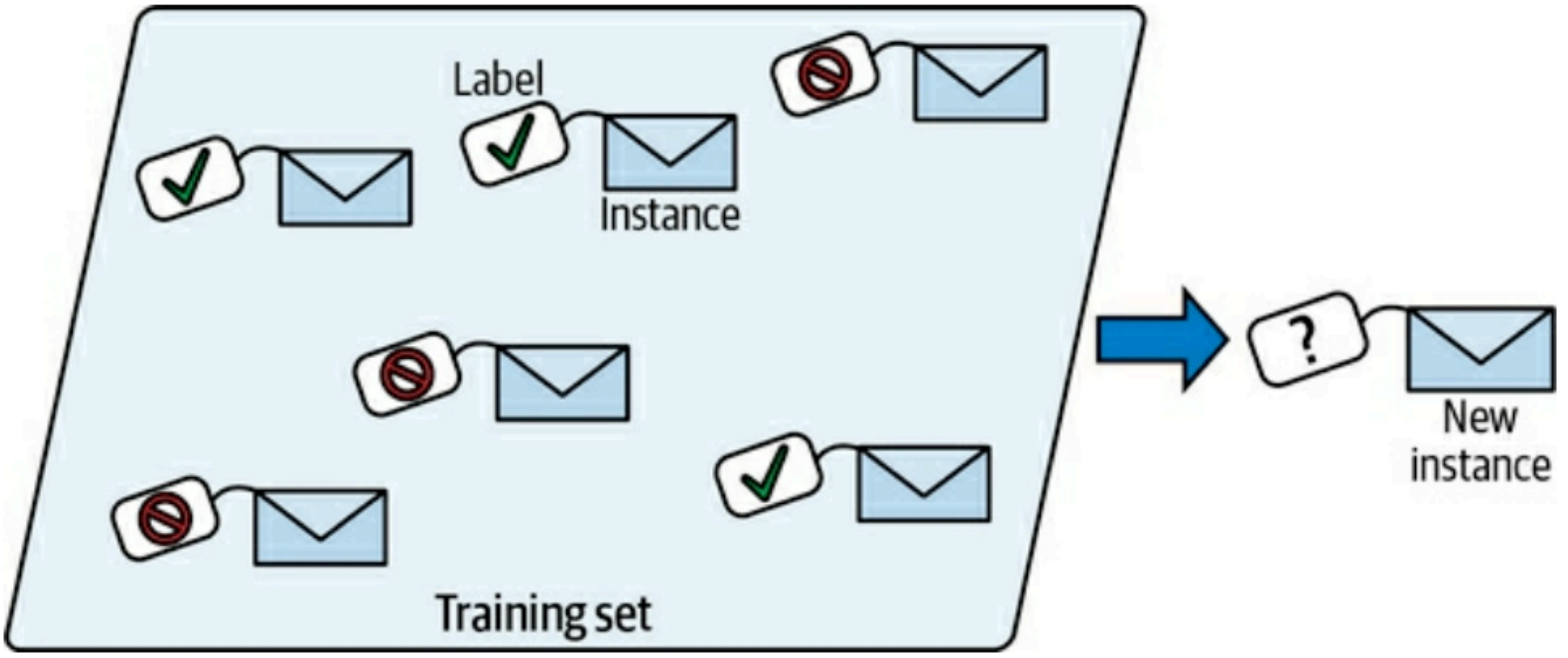


Figure 1-5. A labeled training set for spam classification (an example of supervised learning)

Supervised Learning Tasks

- **Classification**
  - Sort input samples into categories, like a spam filter
- **Regression**
  - Predict a target numerical value, like the price of a car, given a set of features, like mileage, age, brand, etc.

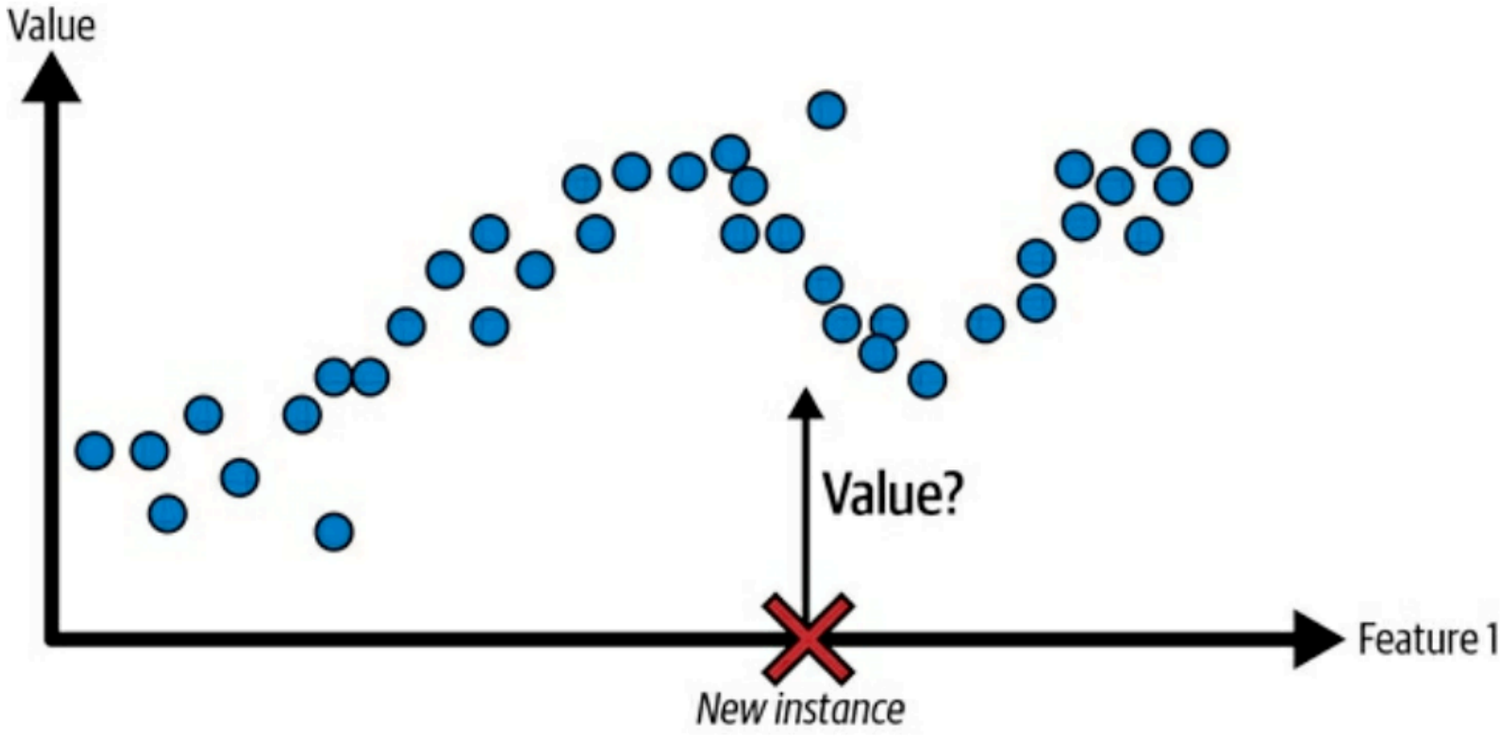


Figure 1-6. A regression problem: predict a value, given an input feature (there are usually multiple input features, and sometimes multiple output values)

Unsupervised Learning

- Training data is unlabeled

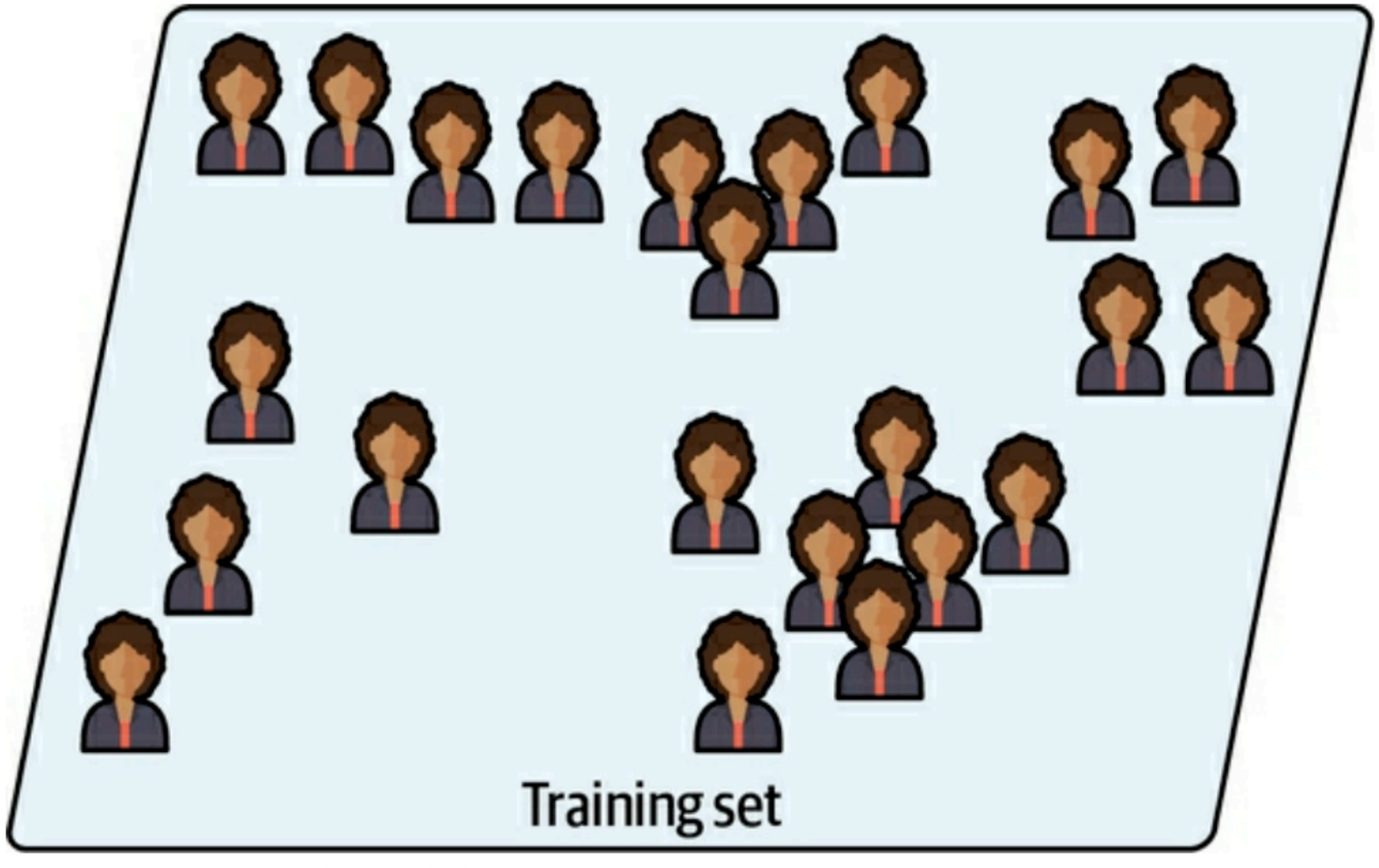


Figure 1-7. An unlabeled training set for unsupervised learning

Unsupervised Learning

- **Clustering** algorithm
  - Sorts data into groups

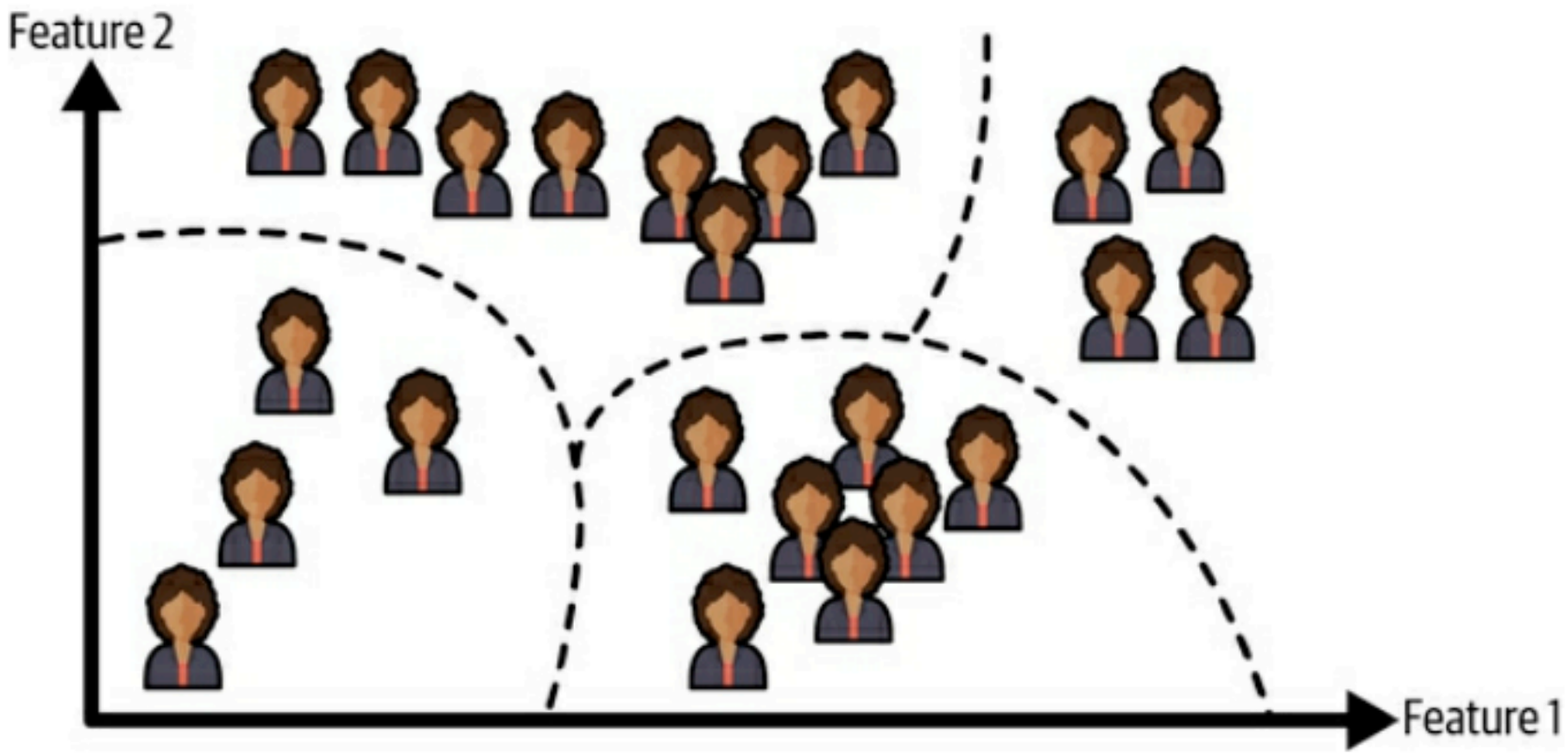


Figure 1-8. Clustering

Unsupervised Learning

- **Visualization**

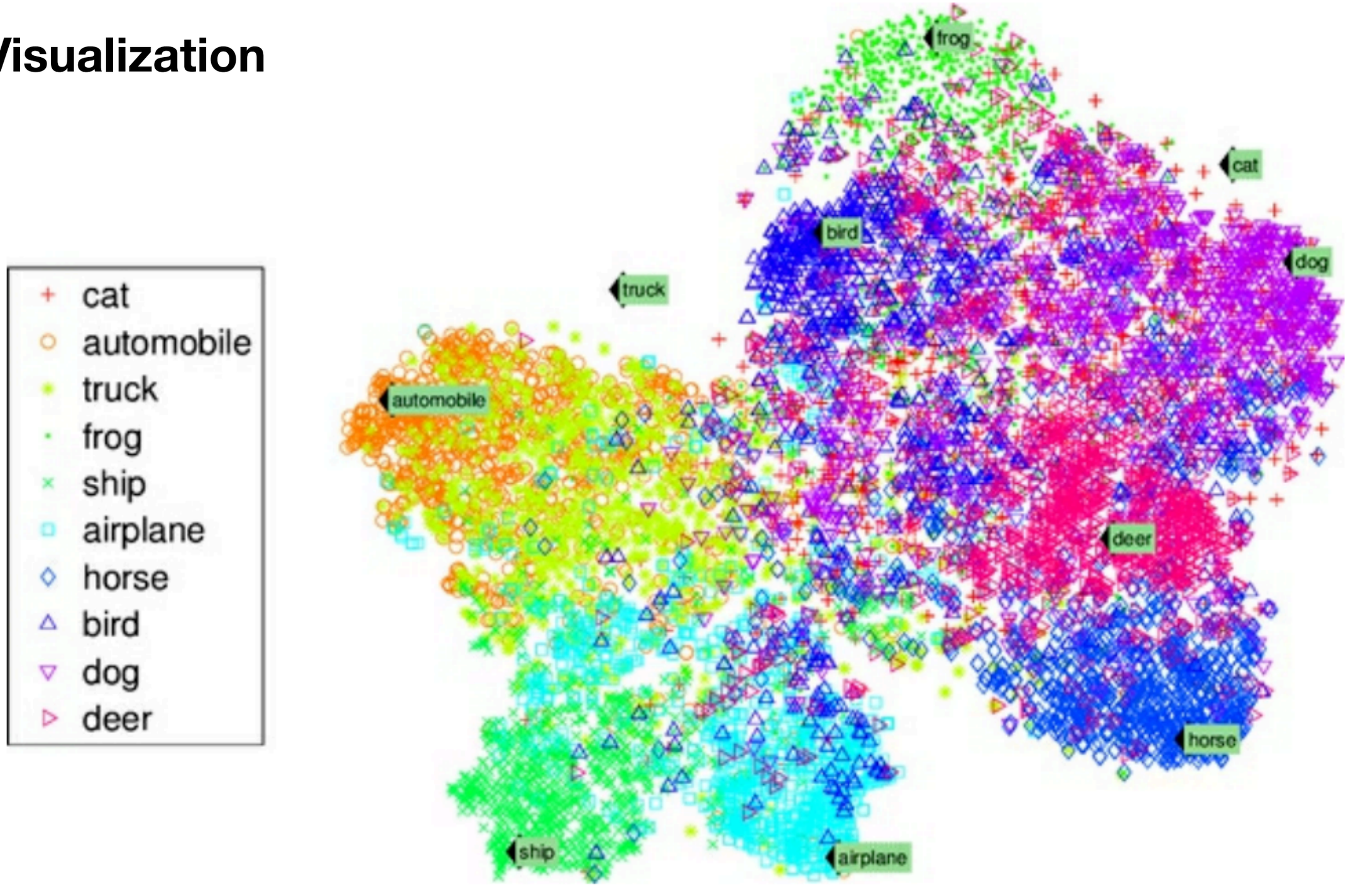


Figure 1-9. Example of a t-SNE visualization highlighting semantic clusters[2]

Dimensionality Reduction

- A way to simplify data
- without losing too much information
- Example: merge correlated features into one
  - For a car, combine *milage* and *age* into *wear-and-tear*
  - This is called *feature extraction*

Unsupervised Learning

- **Anomaly detection**
  - Find unusual credit card transactions
  - Find manufacturing defects

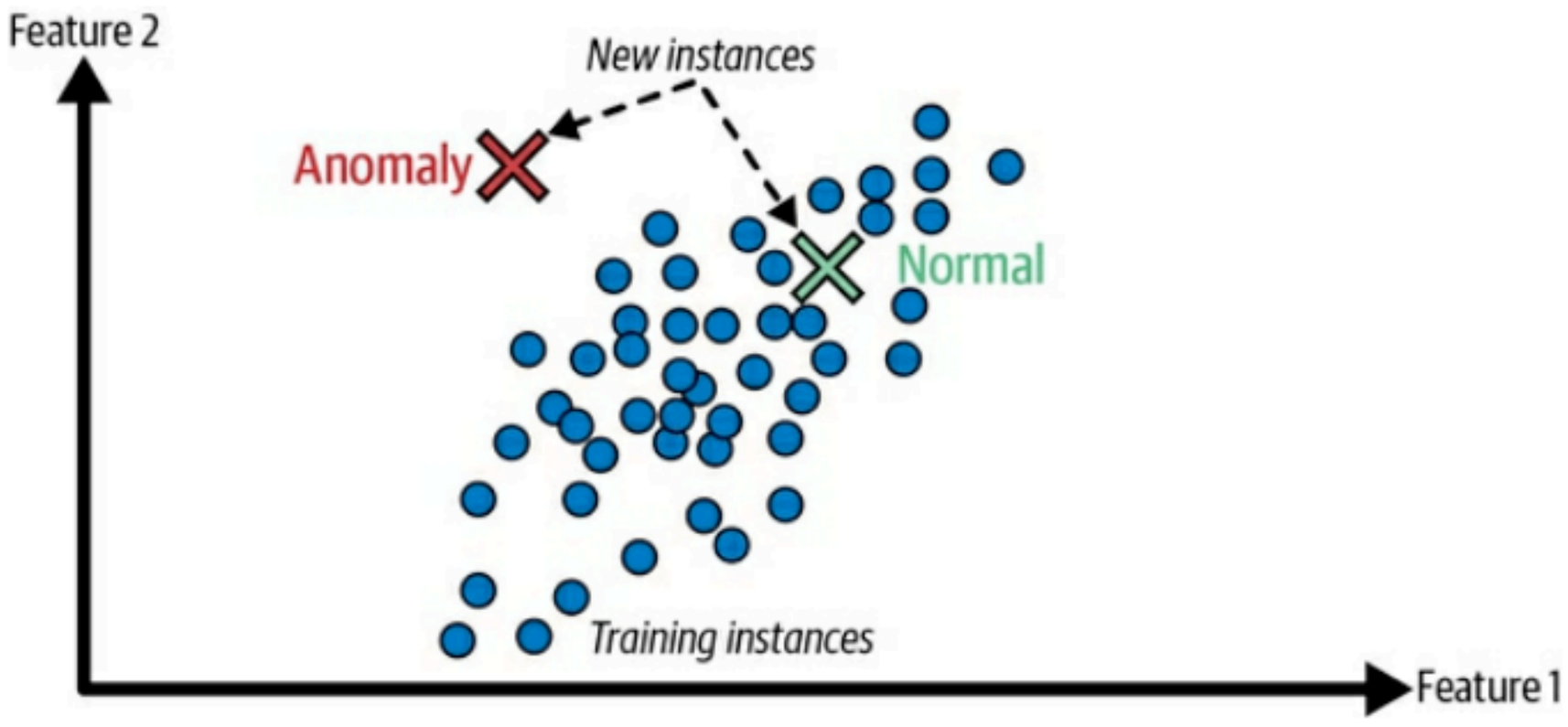


Figure 1-10. Anomaly detection

Unsupervised Learning

- **Association rule learning**
  - Discover interesting relations between attributes
  - Find items customers purchase together

Semi-Supervised Learning

- First unsupervised model groups similar images together
- Then it asks the user to label a group at a time

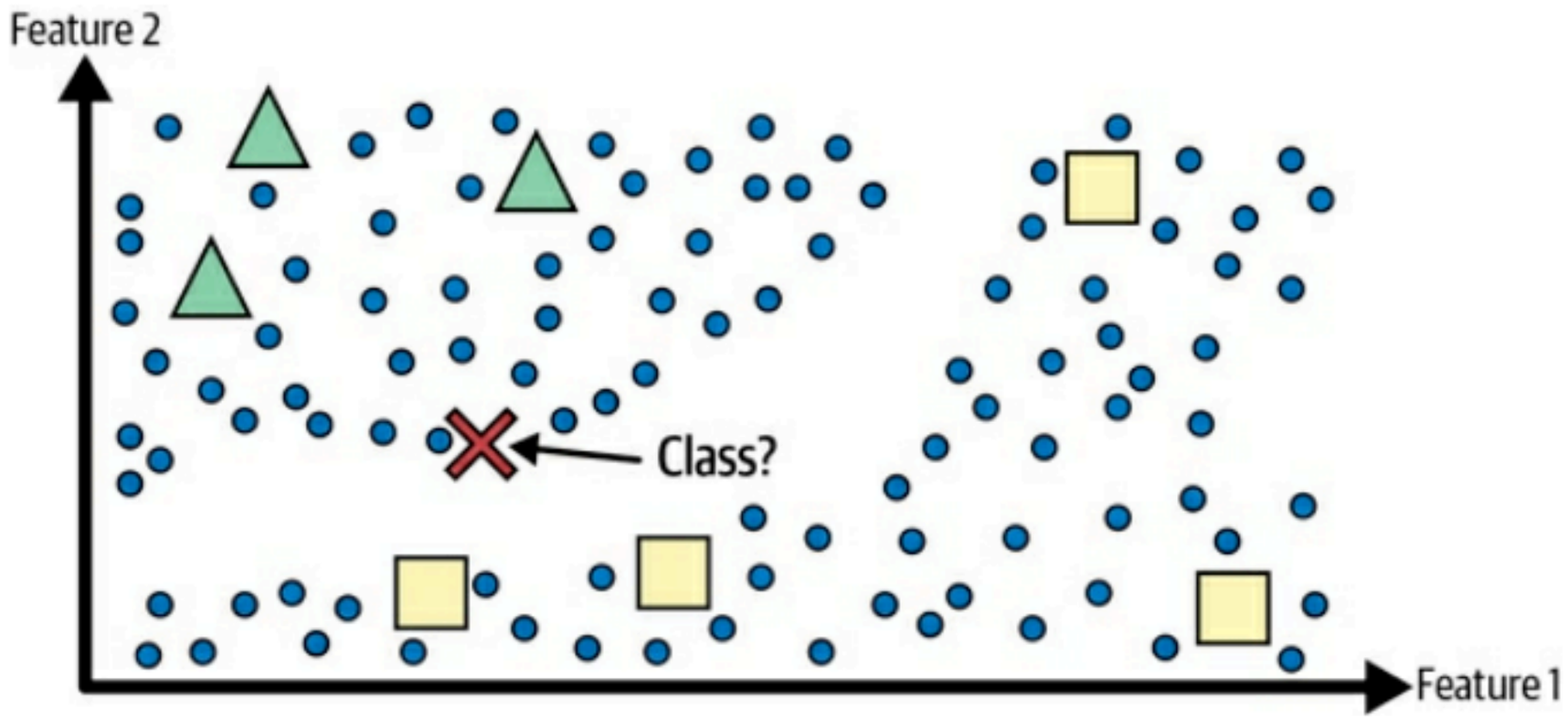


Figure 1-11. Semi-supervised learning with two classes (triangles and squares): the unlabeled examples (circles) help classify a new instance (the cross) into the triangle class rather than the square class, even though it is closer to the labeled squares

Self-Supervised Learning

- Generates a labeled dataset from an unlabeled one
    - Example: mask part of an image, train a model to recover the original image

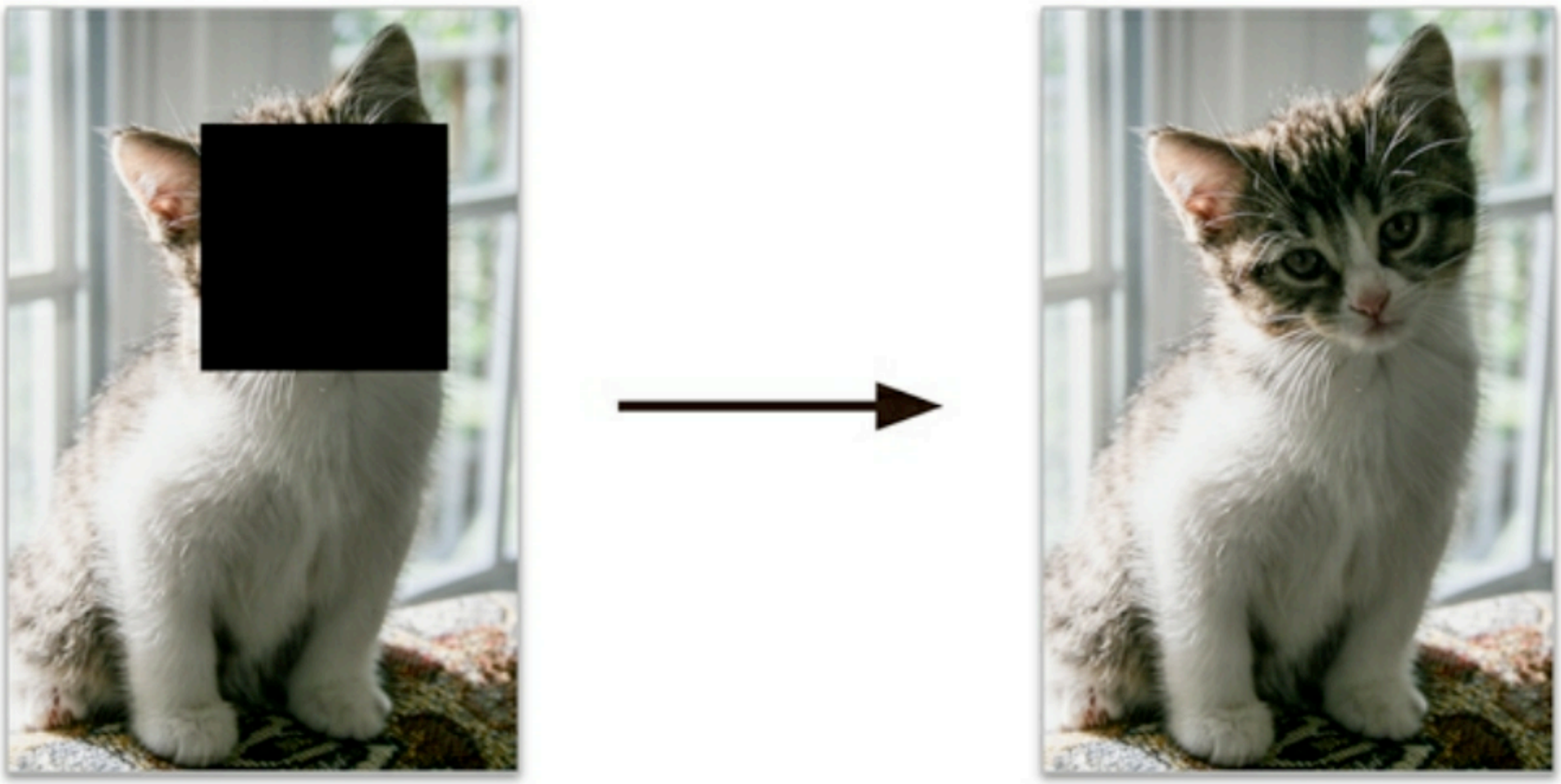
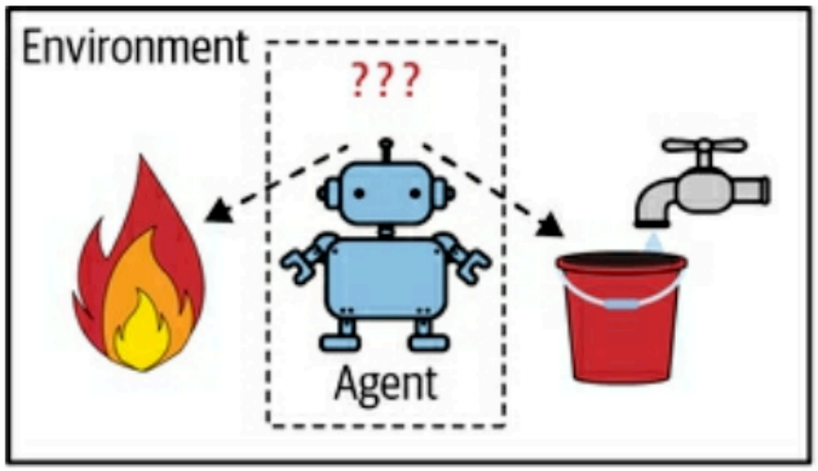


Figure 1-12. Self-supervised learning example: input (left) and target (right)

Reinforcement Learning

- The learning system, called an **agent**
  - Observes the environment
  - Selects and performs actions
  - Gets **rewards** or **penalties**
- Like a robot learning to walk

Reinforcement Learning



1. Observe
2. Select action using policy
3. Action!
4. Get reward or penalty
5. Update policy (learning step)
6. Iterate until an optimal policy is found

Figure 1-13. Reinforcement learning

Ch 1a

Batch v. Online Learning

- **Batch learning**
  - System cannot learn incrementally
  - Must be trained using all the training data
  - Typically model is first trained **offline**
  - Then run in production without learning anymore
- Performance decays slowly over time
  - Because world evolves while model remains unchanged
  - This is **model rot** or **data drift**
- The solution is to regularly retrain the model with up-to-date data

Batch v. Online Learning

- **Online learning**
  - Train system incrementally, feeding it new data
  - Good for rapidly changing data, like stock prices
  - Or weak devices like cell phones
  - Or with huge datasets that cannot fit into main memory
- **Learning rate**
  - If the model learns quickly, it also forgets old data quickly
  - A low learning rate has more inertia; slow to learn new things, but resistant to noise

Instance-Based v. Model-Based Learning

- How does a model **generalize**?
  - Make predictions about new data after learning
- **Instance-Based learning**
  - System learns training data by heart
  - Creates a **measure of similarity** to see how close a new input is to the known ones

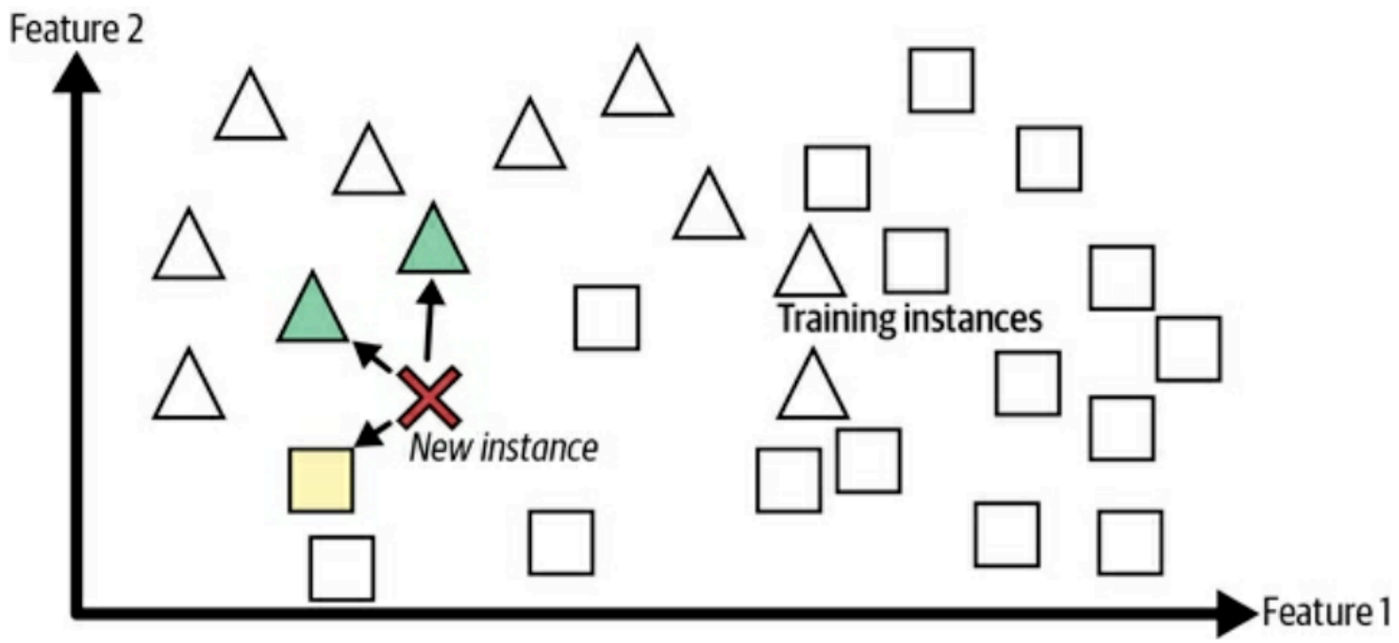


Figure 1-16. Instance-based learning

Instance-Based v. Model-Based Learning

- **Model-Based learning**
  - System builds a model from the training data
  - Uses that model to make predictions

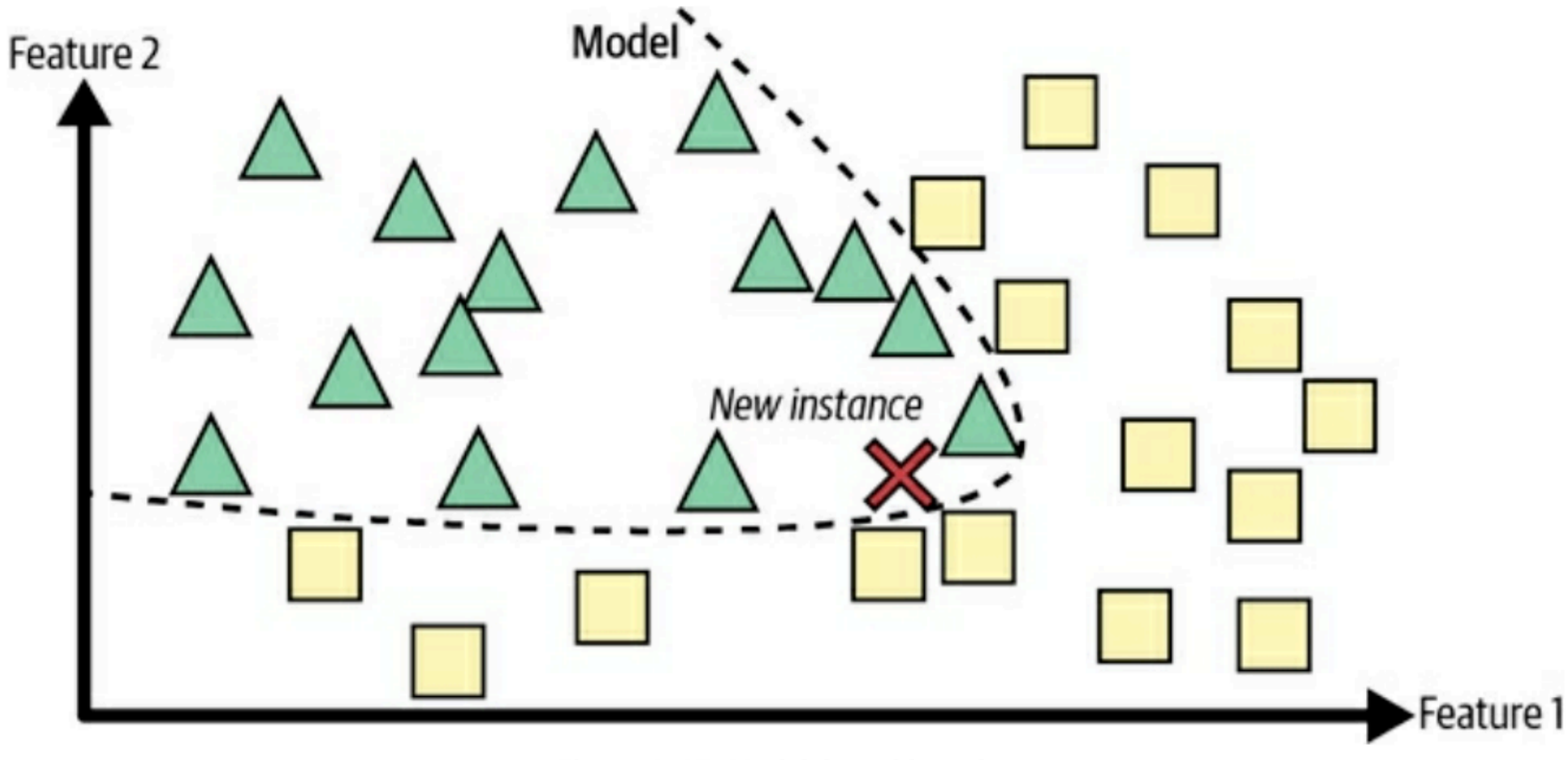


Figure 1-17. Model-based learning

Example: Money and Happiness

- Gather data like this

Table 1-1. Does money make people happier?

Country	GDP per capita (USD)	Life satisfaction
Turkey	28,384	5.5
Hungary	31,008	5.6
France	42,026	6.5
United States	60,236	6.9

Example: Money and Happiness

- There's a trend
- **Model selection**
  - **Linear model** with just one attribute: GDP per capita

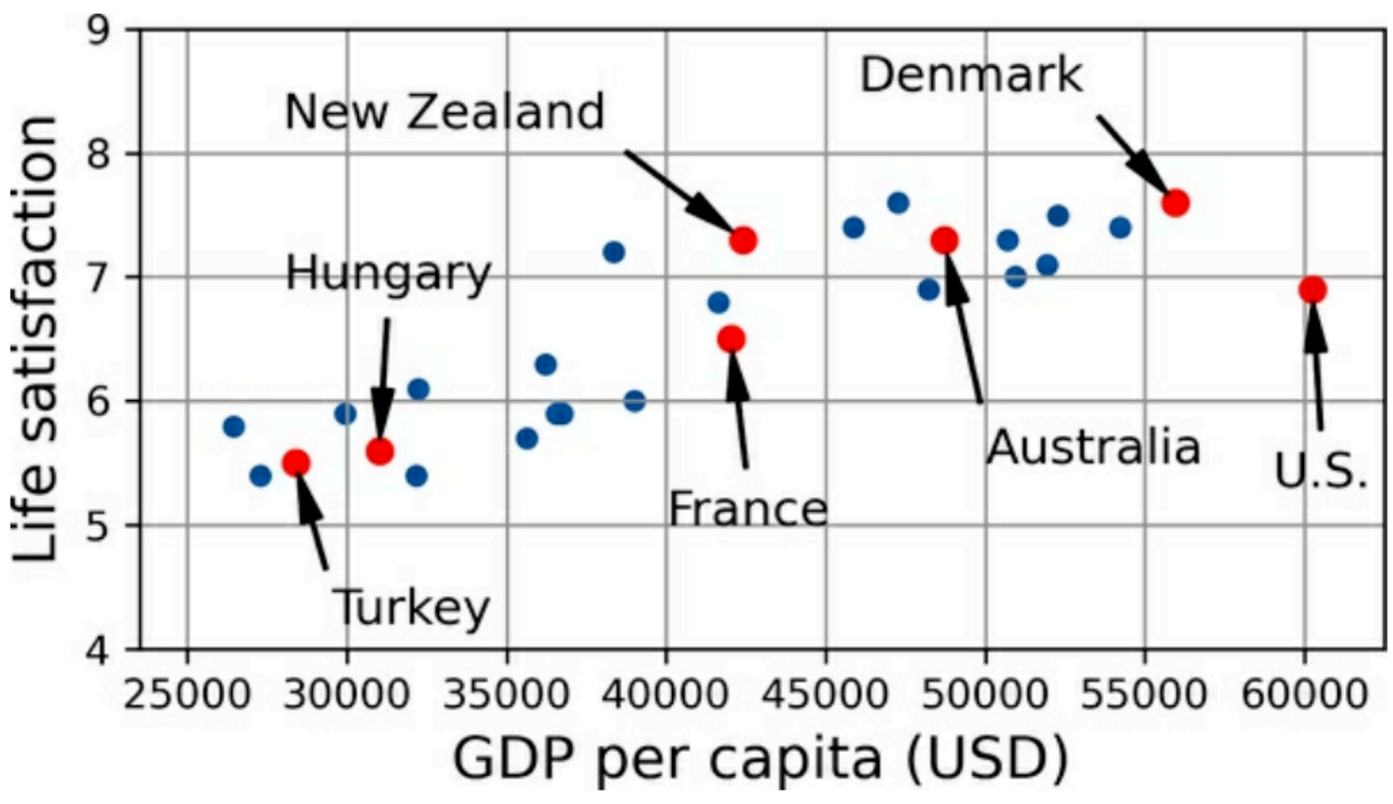


Figure 1-18. Do you see a trend here?

Example: Money and Happiness

- Adjusting the two parameters θ_0 and θ_1 can create any line

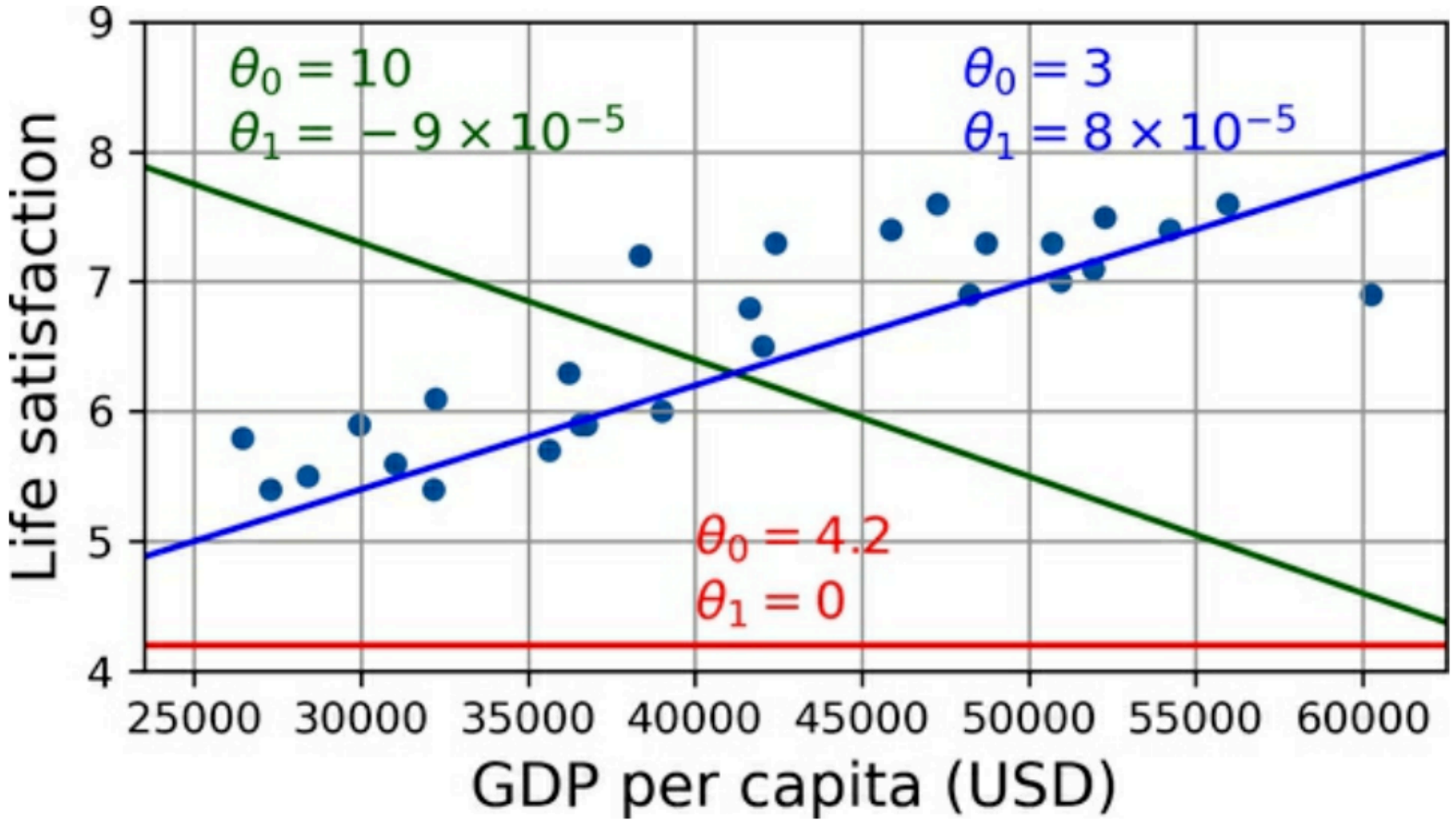


Figure 1-19. A few possible linear models

Example: Money and Happiness

- To select the best line, you need a performance measure
- **Utility function** or **fitness function** measures how good the function is, or
- **Cost function** to measure how bad it is

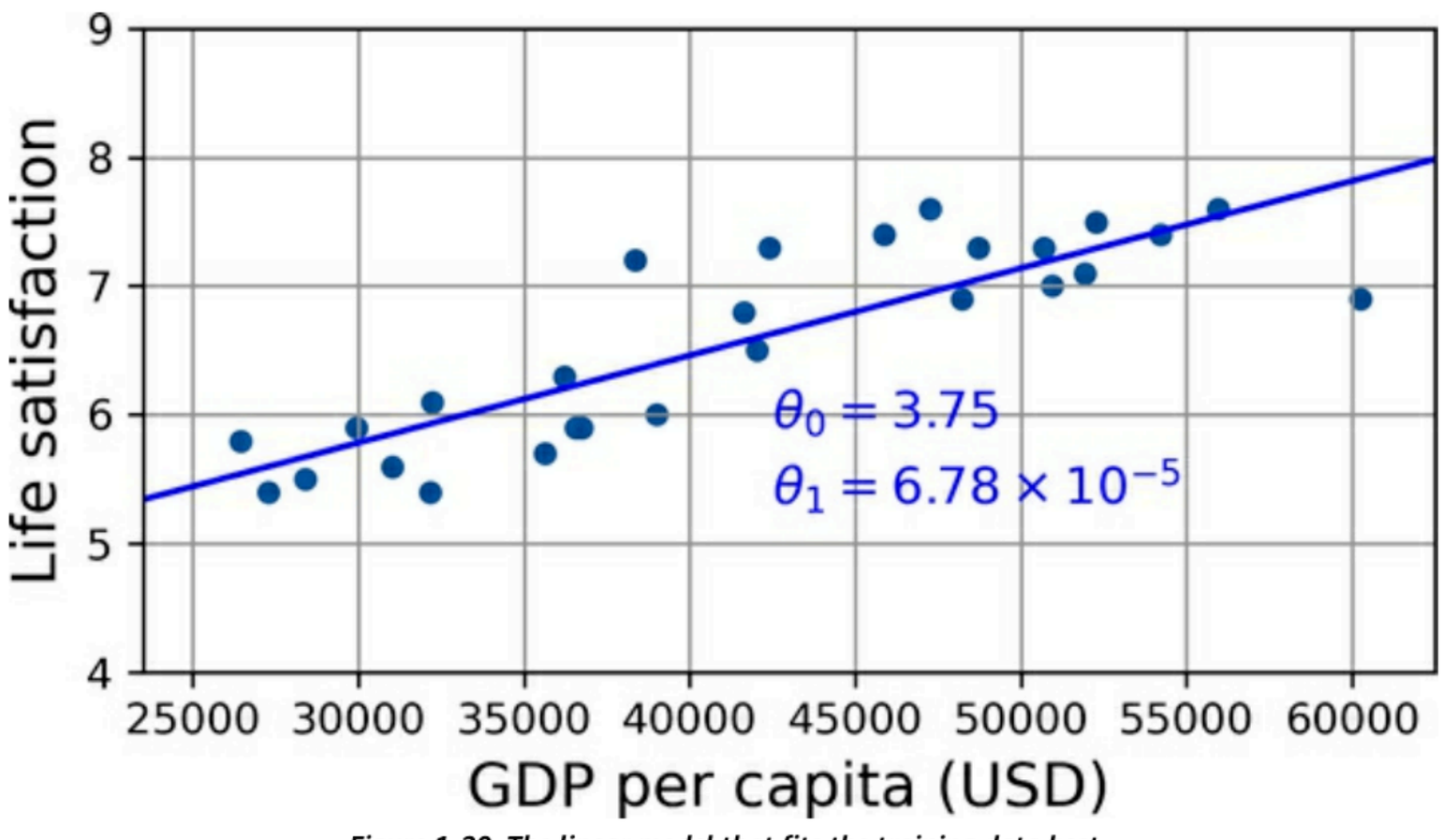


Figure 1-20. The linear model that fits the training data best

Main Challenges of Machine Learning

Main Challenges of Machine Learning

- Insufficient quantity of training data
- Nonrepresentative training data
- Poor-quality data
- Irrelevant features
- Overfitting the training data
- Underfitting the training data

Insufficient Quantity Of Training Data

- Many different algorithms all work as well
- If there's enough training data

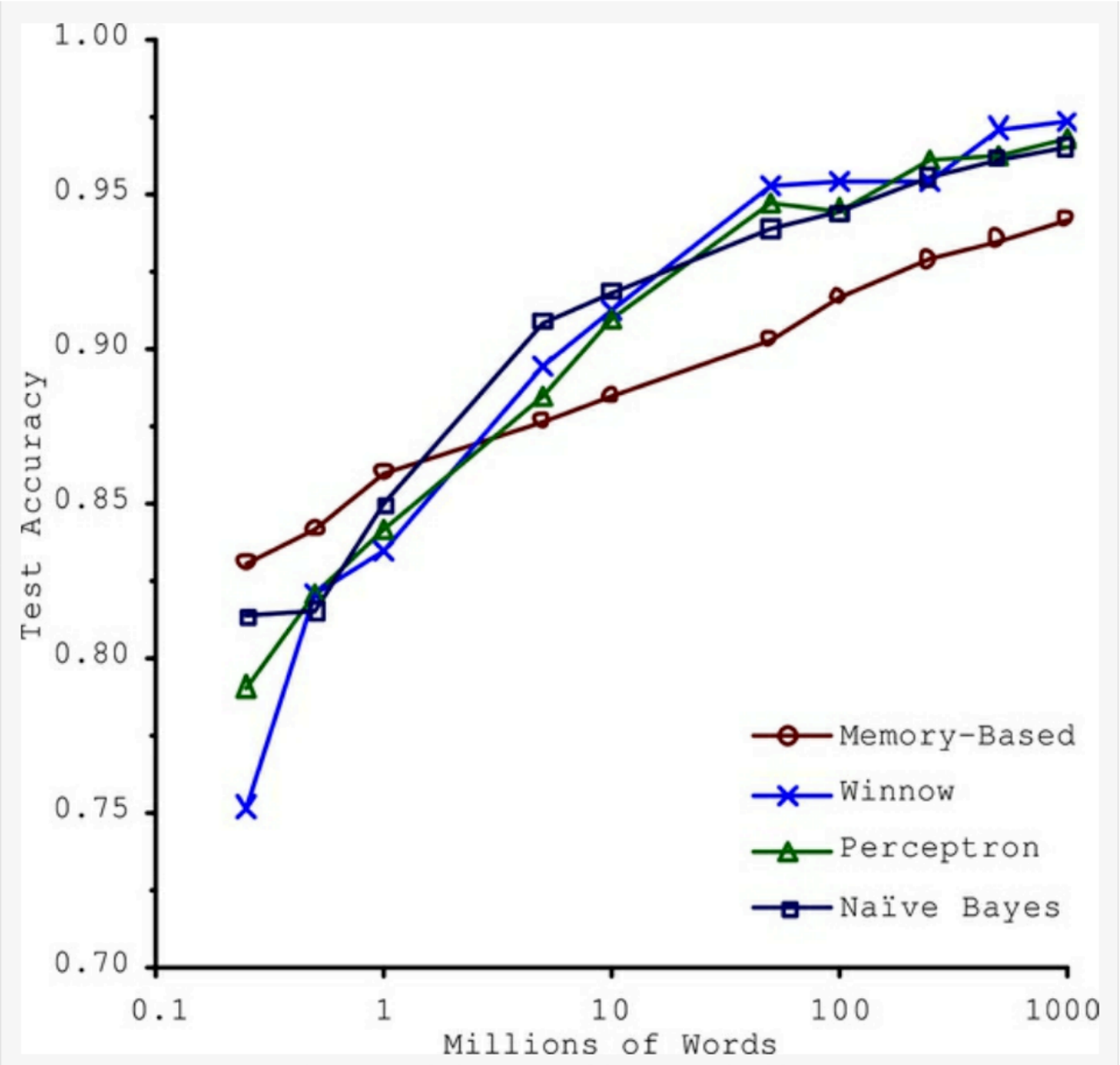


Figure 1-21. The importance of data versus algorithms[8]

Nonrepresentative Training Data

- The model we made earlier included only the blue dots
- Adding the red squares shows it to be seriously wrong
- No linear model works well here

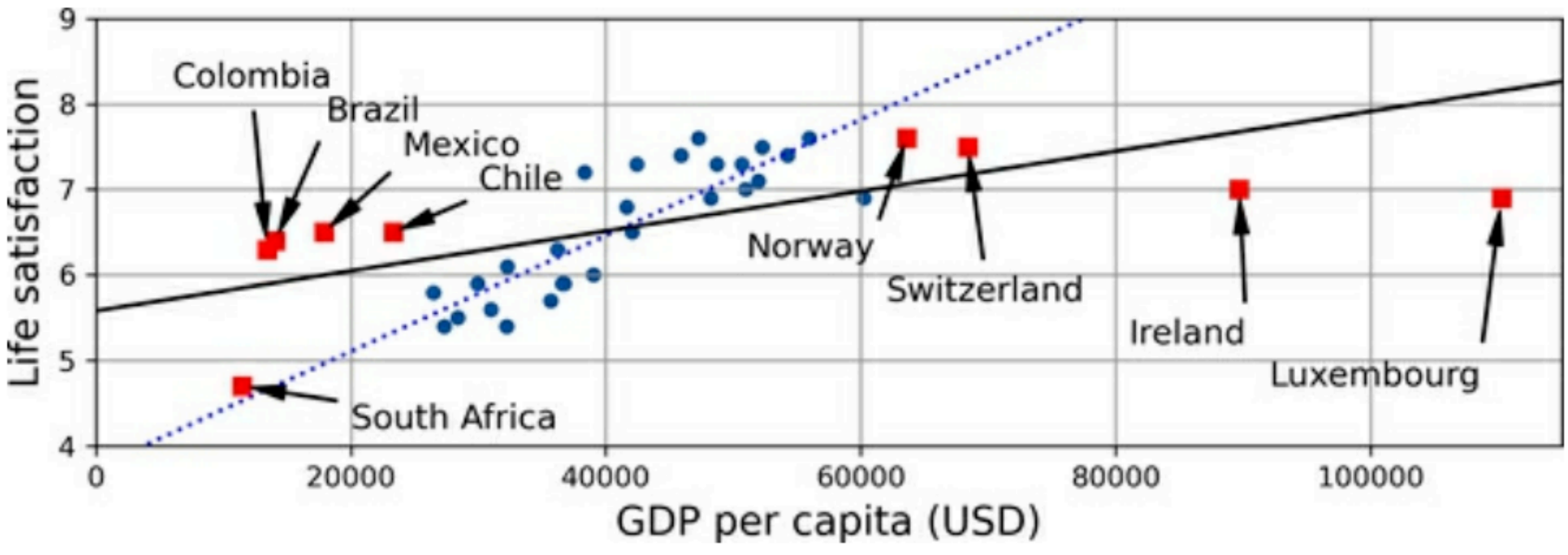


Figure 1-22. A more representative training sample

Poor-Quality Data

- Data scientists spend considerable time cleaning up training data
  - Discarding outliers, or fixing errors manually
  - If some instances are partially missing data
    - e. g. 5% of customers did not specify their age
- You may decide to ignore that attribute entirely, or
  - train one model with it, and one without it

Irrelevant Features

- **Feature engineering**
  - Choosing a good set of features to train on
  - These are the steps
    - **Feature selection** from the available features
    - **Feature extraction** by combining available features
    - Creating new features by gathering new data

Overfitting The Training Data

- A high-degree polynomial has a low loss function
- But it does not really represent the actual trend of the underlying reality
- **Regularization** is making a model simpler to reduce the risk of overfitting

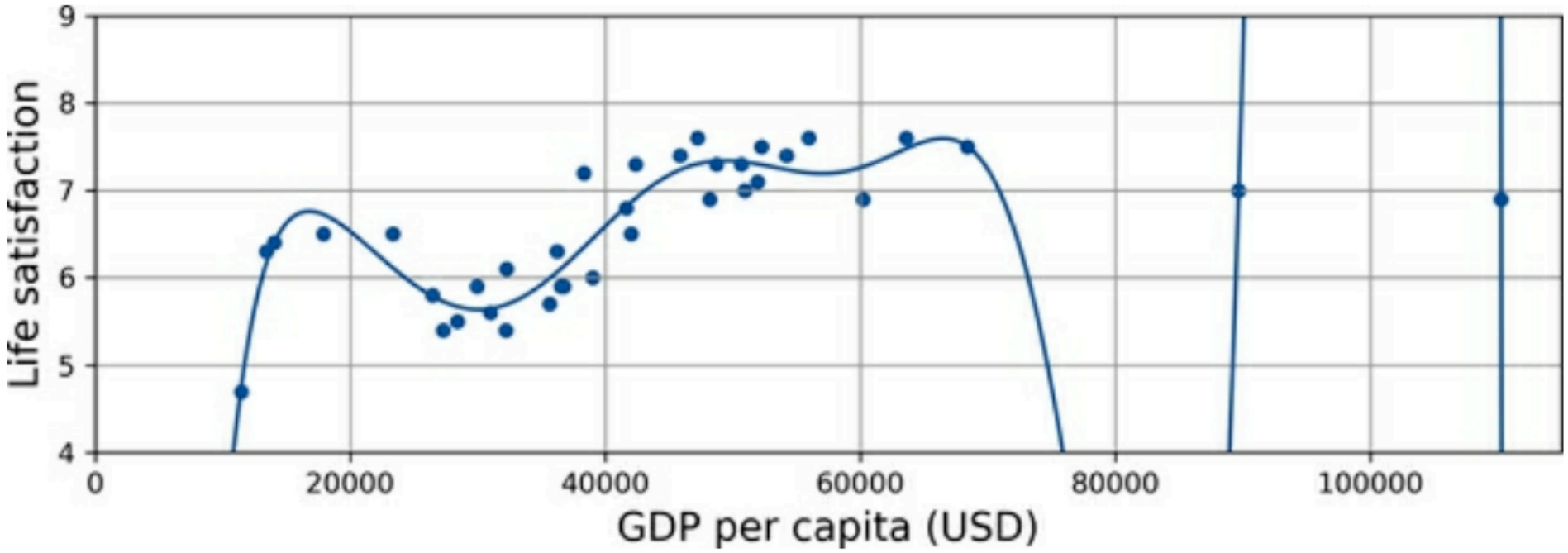


Figure 1-23. Overfitting the training data

Overfitting The Training Data

- A **hyperparameter** controls the amount of regularization
- It lowers the slope of the curve in the model below

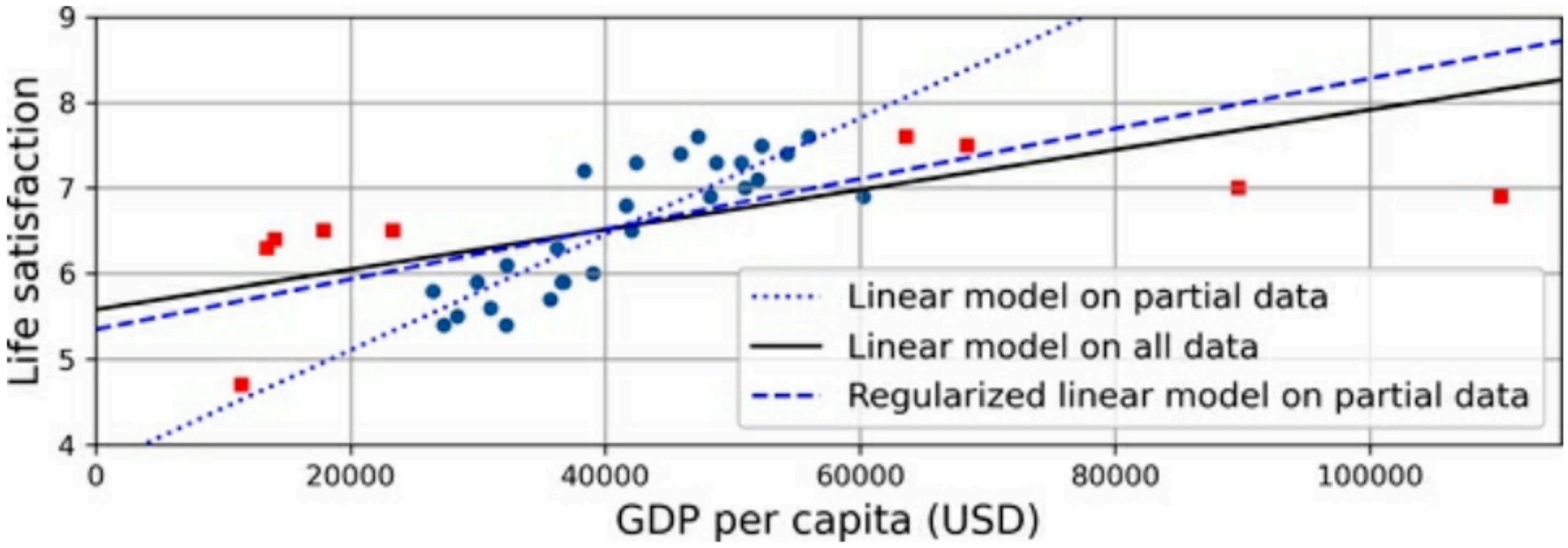


Figure 1-24. Regularization reduces the risk of overfitting

Underfitting The Training Data

- The model is too simple and fails to learn the underlying structure of the data
- Options to fix this problem
  - Select a more powerful model, with more parameters
  - Feed better features to the algorithm (feature engineering)
  - Reduce constraints on the model, such as the regularization hyperpameter

Stepping Back

- **Machine learning** is about making machines get better at some task by learning from data, instead of having to explicitly code rules.
- There are many different types of ML systems: **supervised** or not, **batch** or **online**, i**nstance-based** or **model-based**
- In an ML project you gather data in a **training set**, and you feed the training set to a **learning algorithm**.
- If the algorithm is **model-based**, it tunes some parameters to fit the model
- If the algorithm is i**nstance-based**, it just learns the examples by heart and generalizes to new instances by using a **similarity measure** to compare them to the learned instances.

Stepping Back

- The system will not perform well if your training set is too small, or if the data is not representative, is noisy, or is polluted with irrelevant features (garbage in, garbage out).

- Lastly, your model needs to be neither too simple (in which case it will underfit) nor too complex (in which case it will overfit).

Testing and Validating

Testing and Validating

- Split data into **training set** and **test set**
- The error rate on new cases is called the **generalization error** or the **out-of-sample error**
- If the training error is low, but the generalization error is high, your model is overfitting the data
- If you are using a **regularization hyperparameter**, you need to also split some of the data out into a **validation set** to select the best value for the hyperparameter
- **Cross-validation** uses many different small validation sets, repeating the training process for each one, to get a more accurate hyperparameter value

Testing and Validating

- **Data mismatch**
    - Suppose you want to train a iphone app tp recogmize flowers from photos
    - You can download flower photos from the Web, but they are not the same as photos taken with an iPhone
- Make sure both the validation set and the test set are real representative samples
    - Actual iPhone pictures
- That way you'll know how good your model will be in actual use

Ch 1b