Artificial Intelligence and Machine Learning for Networks and IoT

Class 1 — Introduction

Pedro Braconnot Velloso

Who am I?

- Pedro Braconnot Velloso
- Associate professor
- Bureau 35.1.50

- Interests
  - Internet of Things
  - Wireless networks
  - Data science
  - Anomaly detection
  - Security

Syllabus

- Introduction
- Practical skills and Linear Regression
- End-to-End project
- Classification
- Artificial Neural Network
- Unsupervised Learning
- TensorFlow

Syllabus

- Dimensionality Reduction (Agathe Blaise)
- Reinforcement Learning
- Training Deep Neural network

- Classes include labs
  - Covering the mains models and algorithms

Bibliography

- Aurélien Géron, "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow", 2nd Edition, O'Reilly Media, 2019
- James, G., Witten, D., Hastie, T., Tibshirani, R. "An Introduction to Statistical Learning”, 2nd edition, 2021
- Tom Mitchell, ”Machine Learning”, McGraw-Hill Science, 1997

Grades

- Components
  - 2 Lab exercises (L_1, $L_{2,}$)
  - Final exam (E)
- Final grade (FG)

$$FG = \frac{7 \times E + 3 \times (L_1 + L_2)}{10}$$

Introduction

Definitions

"The use and development of computer systems that are able to learn and adapt without following explicit instructions, by using algorithms and statistical models to analyse and draw inferences from patterns in data."

Oxford Languages

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E"

Tom Mitchell

Main idea

- Understand how to program computers
    - To learn to improve automatically with experience
    - Beginning in1950

- Machine learning is inherently a multidisciplinary field
    - Artificial intelligence, probability and statistics, computational complexity theory, control theory, information theory, philosophy, psychology, neurobiology, and others

Traditional applications

- Traditional applications
  - Speech recognition
  - Data mining
  - Data Prediction
  - Game
    - Backgammon
    - Chess

Machine Learning recent success

- Data availability
- Powerful and cheaper resources
  - Hardware
    - Tensor Processing Unit (TPU) — google
      - AI accelerator (Application-Specific Integrated Circuit — ASIC)
  - Public cloud computing

New applications

- Agriculture
- Adaptive website
- Affective computing
- Astronomy
- Automated decision-making
- Banking
- **Bioinformatics**
- Brain–machine interfaces
- Cheminformatics
- **Climate science**
- **Computer vision**
- Credit-card fraud detection
- DNA sequence classification
- Economics
- **Financial market analysis**
- General game playing
- Handwriting recognition
- Information retrieval
- Insurance
- **Internet fraud detection**
- Knowledge graph embedding
- Linguistics
- Marketing
- **Medical diagnosis**
- Natural language
- Online advertising
- Optimization
- **Recommender systems**
- Robot locomotion
- Search engines
- Sentiment analysis
- Sequence mining
- Software engineering
- Structural health monitoring
- Syntactic pattern recognition
- **Telecommunication**
- Theorem proving
- **Time-series forecasting**
- User behavior analytics

New applications

Connected systems

Well-defined learning problems

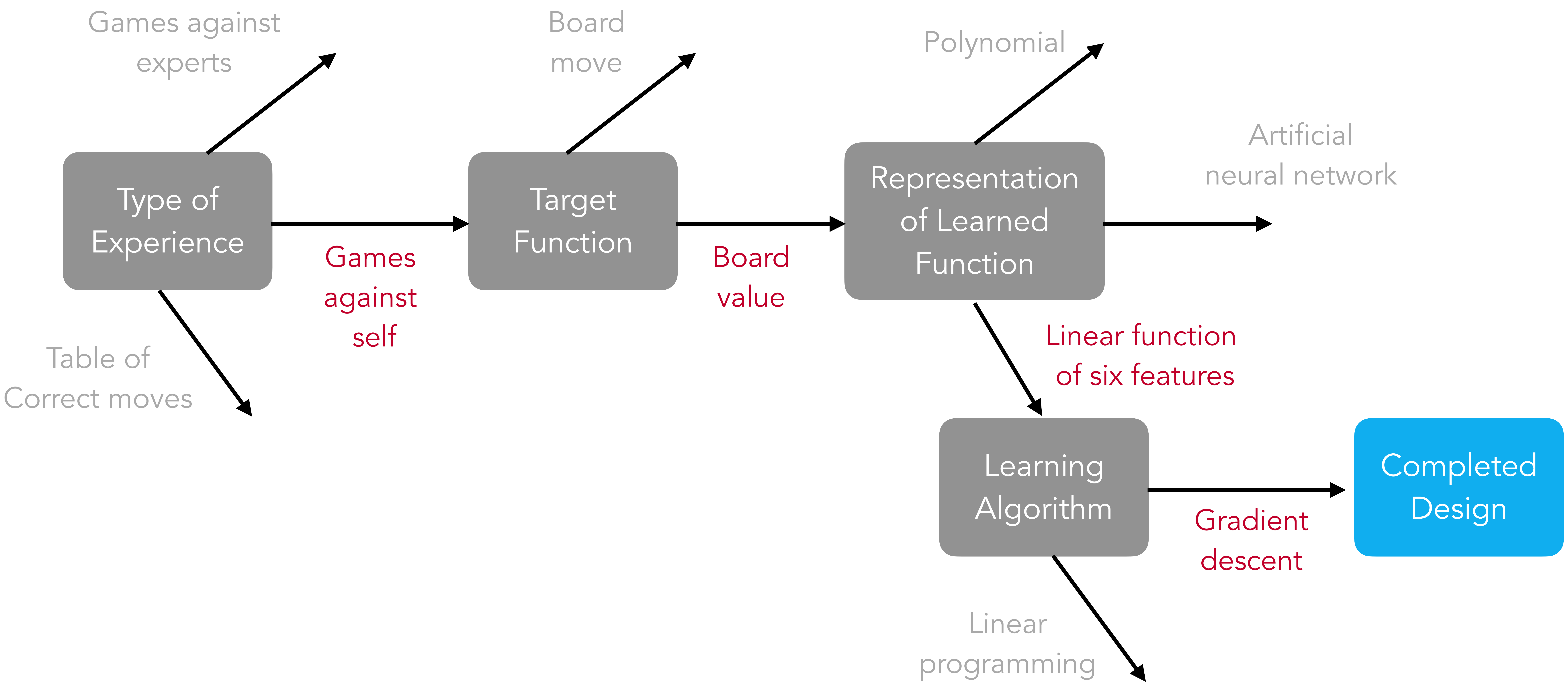
> "A computer program is said to learn from experience **E** with respect to some class of tasks **T** and performance measure **P**, if its performance at tasks in **T**, as measured by **P**, improves with experience **E**"
>
> Tom Mitchell

- Identifying these 3 features

Task T	Playing checkers
Performance measure P	Percent of games won against opponents
Training experience E	Playing against itself

source: https://www.dutchcrafters.com/Amish-Wooden-Checker-Board/

Designing the checkers learning program



Exercise

- A handwriting recognition learning problem

Task T	Recognizing and classifying handwritten words within images
Performance measure P	Percent of words correctly classified
Training experience E	A database of handwritten words with given classifications

Approaches

- Models
  - The part of a Machine Learning system that learns and makes predictions
    - Neural networks and random forests are examples of models

- Supervised Learning
- Unsupervised Learning
- Reinforcement learning

Spam detection system

- Spam detection problem

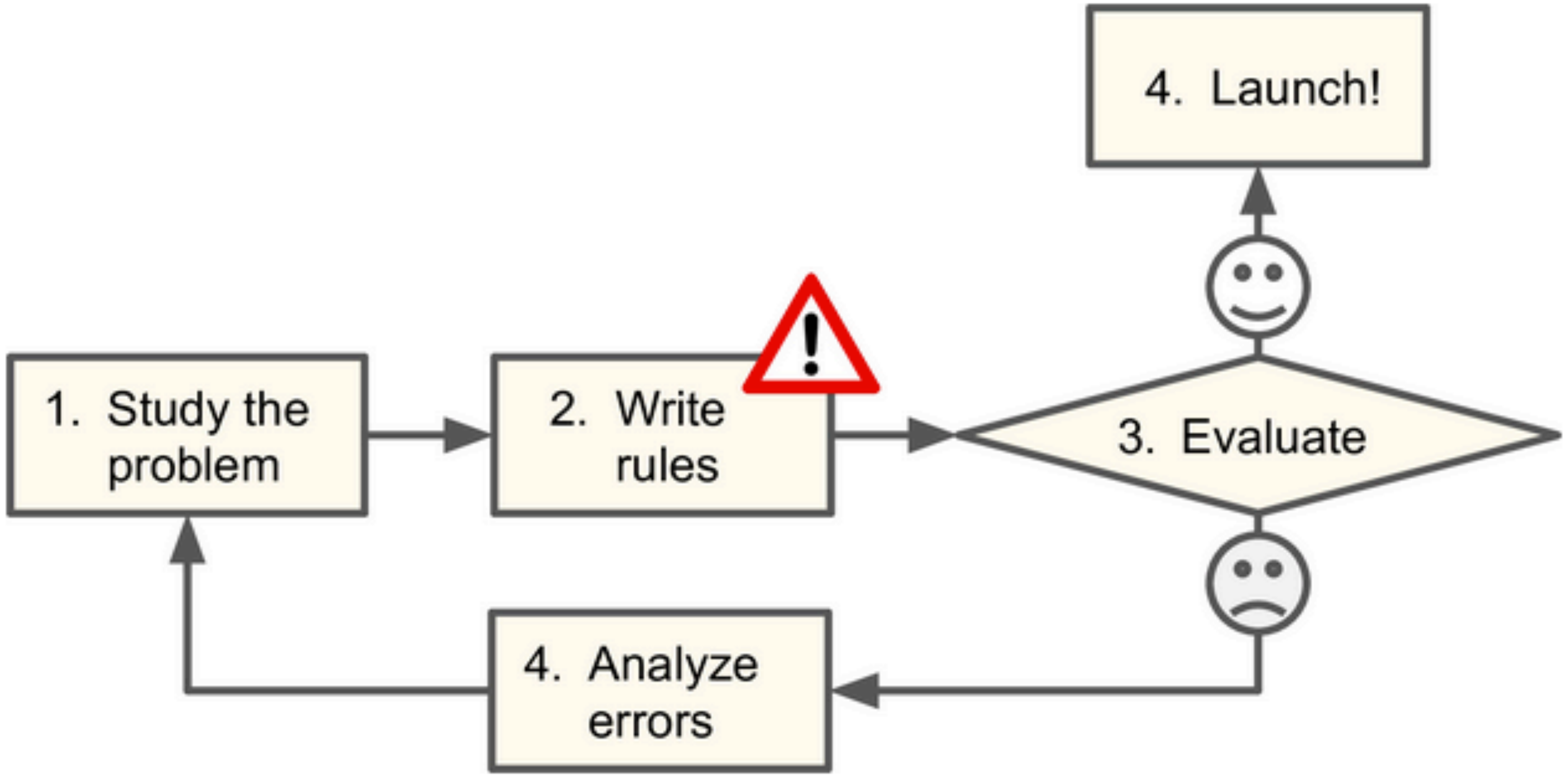
Task T	To flag spam for new emails
Performance measure P	To be defined
Training experience E	The training data

- Performance P
  - The ratio of correctly classified emails
    - Accuracy
      - It is often used in classification tasks

Spam detection with traditional programming

- Examine what spam typically looks like
  - Notice that common words, phrases, patterns in
    - The subject
      - "4U," "credit card," "free," and "amazing"
    - The sender's name
    - The email's body
    - Other parts of the email
- Write a detection algorithm for each of the patterns
  - Program would flag emails as spam if a number of these patterns were detected
- Test your program and repeat steps 1 and 2 until it was good enough to launch

Block diagram



source: Géron, A. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow"

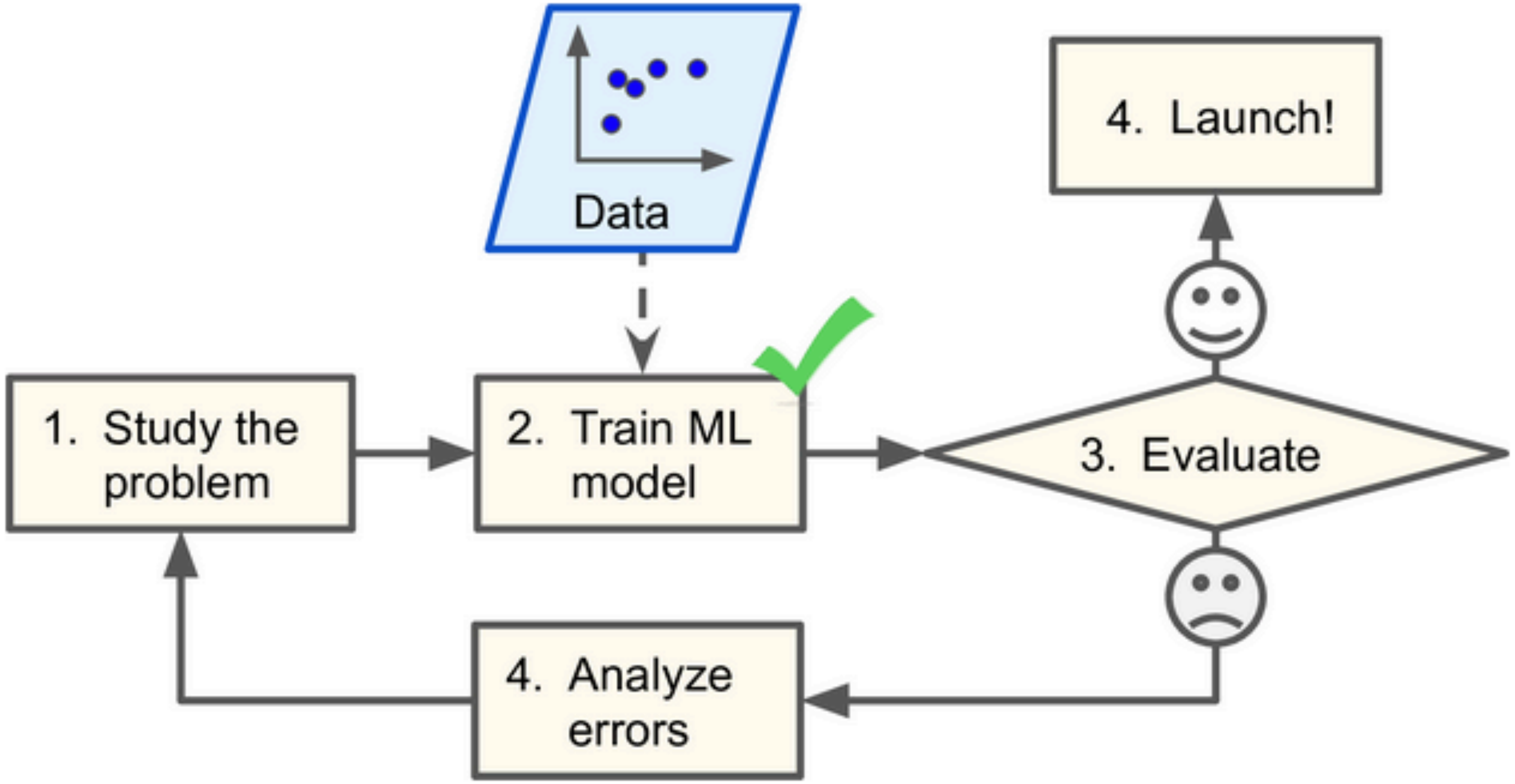
ML vs Traditional programming

- Since the problem is difficult
    - Program will likely become a long list of complex rules
        - Hard to maintain
- ML-based spam filter
    - Automatically learns
        - Words and phrases are good predictors of spam
            - Detecting unusually frequent patterns of words
    - The program is much shorter, easier to maintain, and most likely more accurate

ML vs Traditional programming

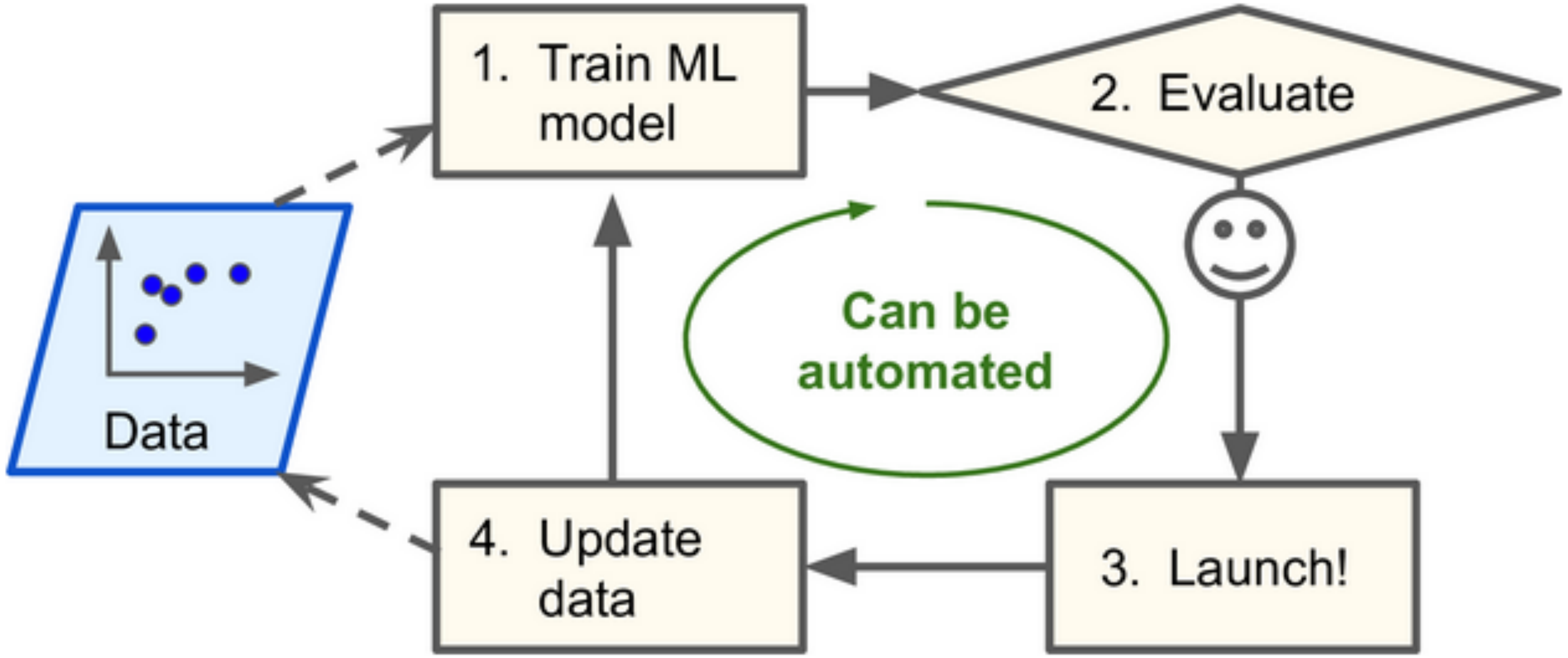
- When spammers notice that all emails containing “4U” are blocked
  - Change the strategy using“For U”
  - Traditional programming techniques
    - Need to be updated to flag “For U” emails
    - Writing new rules forever
- ML-based techniques automatically notices
  - “For U” has become unusually frequent in spam flagged by users
    - It starts flagging them without our intervention

Machine learning approach



What happens when spammers change their pattern?

Automatically adapting to change



source: Géron, A. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow"

ML application

- ML can be applied to problems
  - Too complex for traditional approaches
  - Have no known algorithm

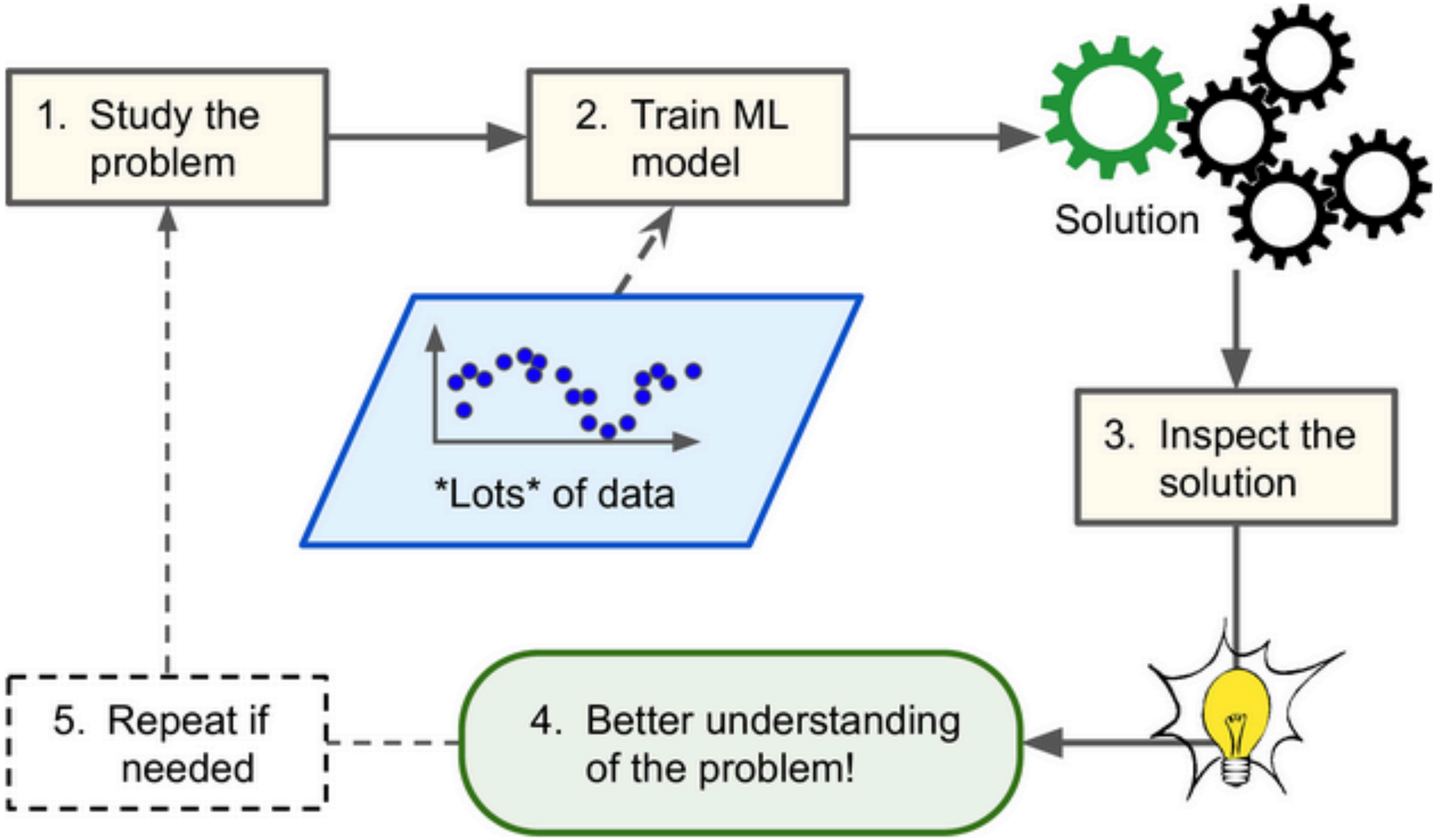
ML application

- Write a program capable of distinguishing the words “one” and “two”
  - Notice that the word “two” starts with a high-pitch sound (“T”)
  - Hardcode an algorithm that measures high-pitch sound intensity
    - Distinguish ones and twos
  - This technique will not scale to thousands of words
    - Spoken by millions of very different people
    - In noisy environments
    - Several languages

Helping humans learn

- ML models can be inspected to see what they have learned
  - Although for some models this can be tricky
- Data mining
  - Digging into large amounts of data to discover hidden patterns
- Example of a spam filter
  - Reveals the list of words and combinations of words
    - It believes are the best predictors of spam
  - Sometimes this will reveal unsuspected correlations or new trends
    - Lead to a better understanding of the problem

Machine Learning can help humans learn



source: Géron, A. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow"

Useful applications for ML

- Problems for which existing solutions require a lot of fine-tuning or long lists of rules
  - ML model can simplify code and perform better
- Complex problems
  - Using a traditional approach yields no good solution
- Dynamic environments
  - ML system can easily be retrained on new data
- Getting insights about complex problems and large amounts of data

Machine learning classification

- There are different types of Machine Learning classifications
- Type of training supervision
  - Supervised, Unsupervised, Semi-supervised, Self-supervised, and others
- Learning mode
  - Whether or not they can learn incrementally on the fly
    - Batch
    - Online

Machine learning classification

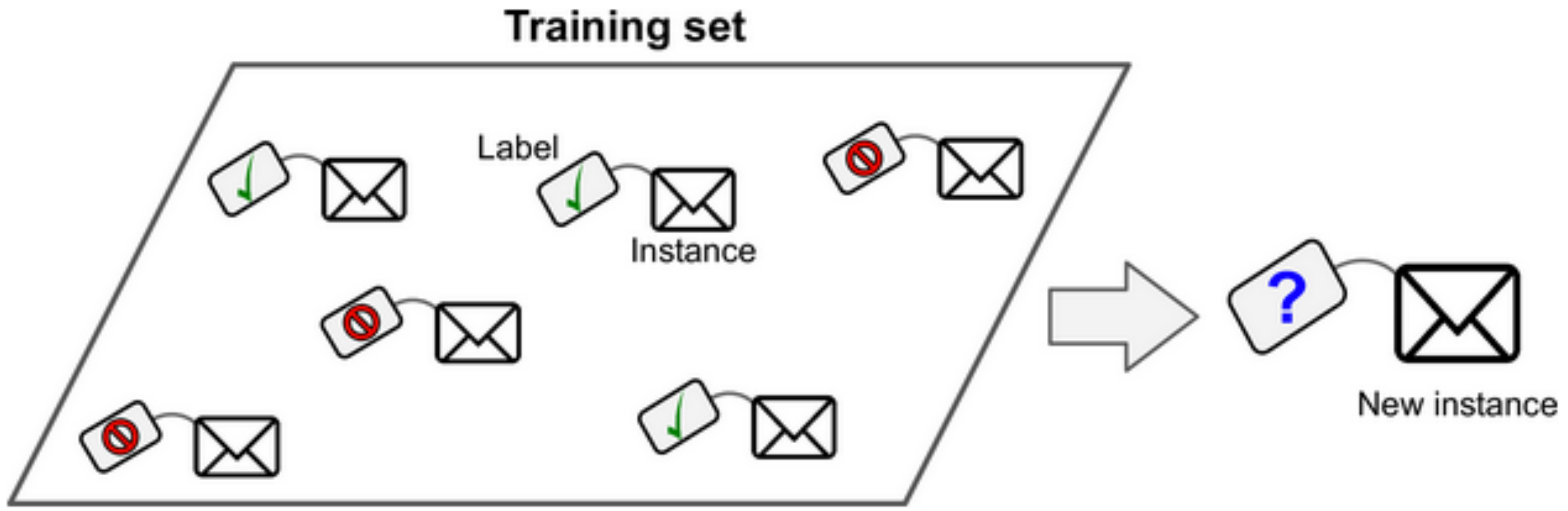
- Instance-based
  - Simply comparing new data points to known data points
- Model-based
  - Detecting patterns in the training data
  - Building a predictive model

Training supervision classification

- Classified according to the amount and type of supervision
  - During training
- There are many categories
  - Supervised learning
  - Unsupervised learning
  - Self-supervised learning
  - Semi-supervised learning
  - Reinforcement Learning

Supervised learning

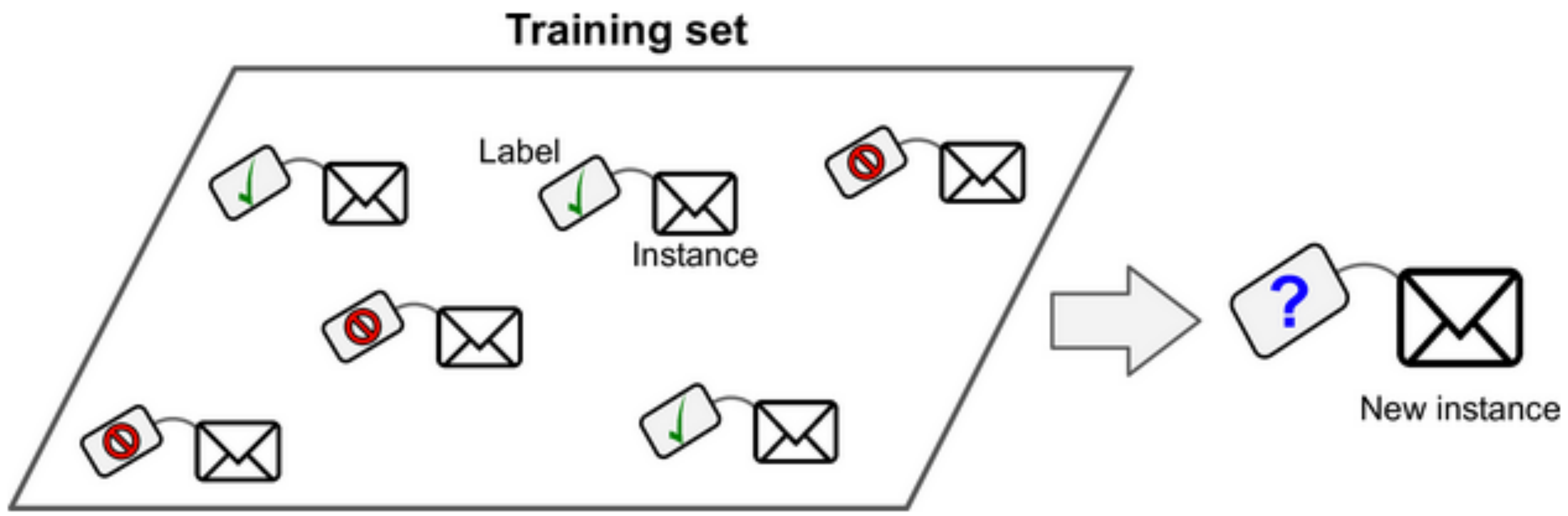
- Learning tasks
  - Classification
    - The spam filter is a good example
      - It is trained with many emails along with their labels (spam or not)
      - it must learn how to classify new emails.



source: Géron, A. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow"

Supervised learning

- Learning tasks
  - Regression
    - Predict a target numeric value
      - The price of a car given a set of features
        - Mileage, age, brand, etc.
      - To train the system
        - Give it many examples of cars
          - Including both their features and their targets (prices)

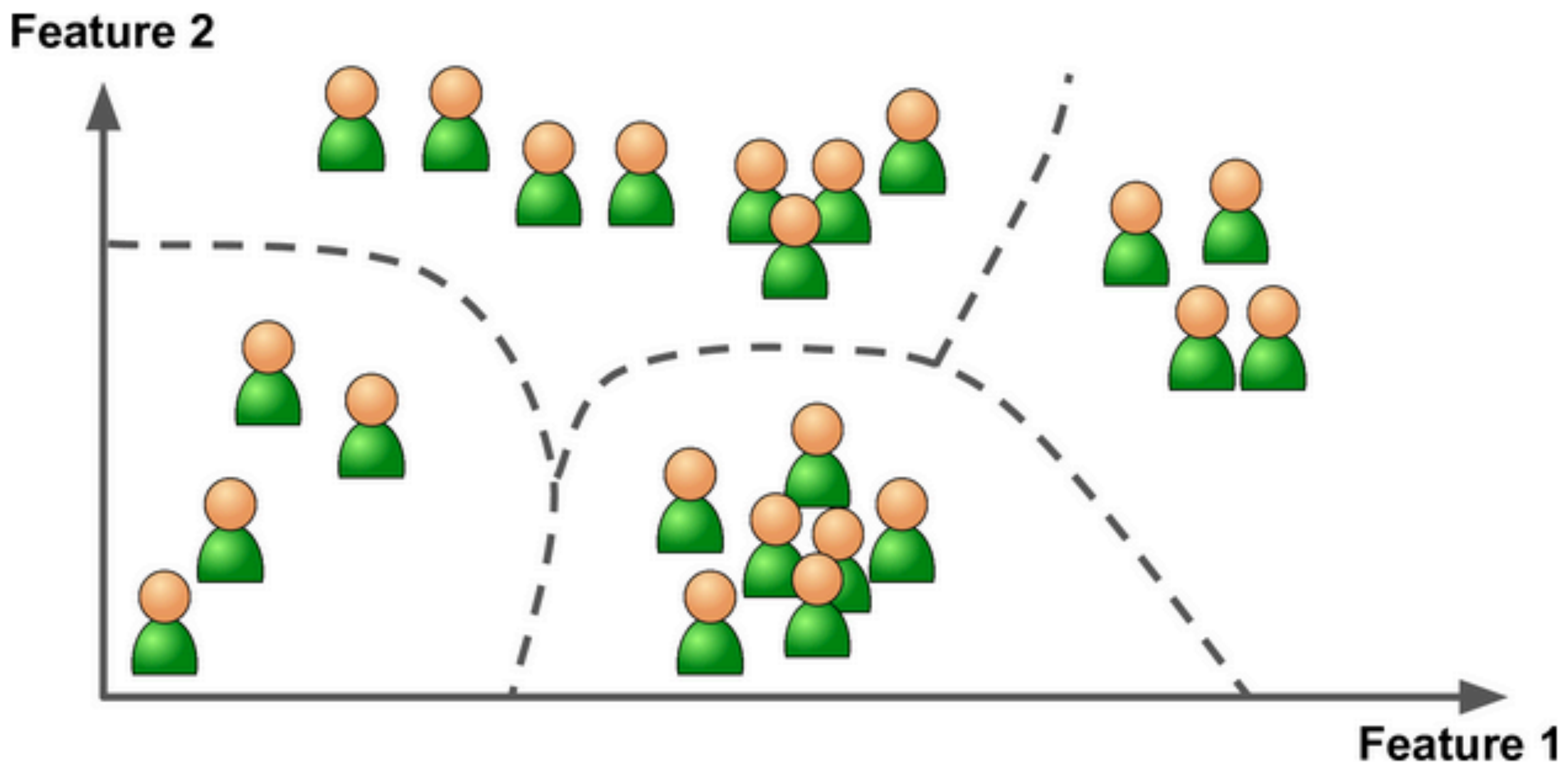


Supervised learning

- The word **target** is more common in regression tasks
- The word **label** is more common in classification tasks
- Features are sometimes called predictors or attributes
- Regression models can be used for classification
    - Logistic Regression is commonly used for classification
        - As it can output a value
            - Corresponds to the probability of belonging to a given class
                - 22% chance of being spam

Unsupervised learning

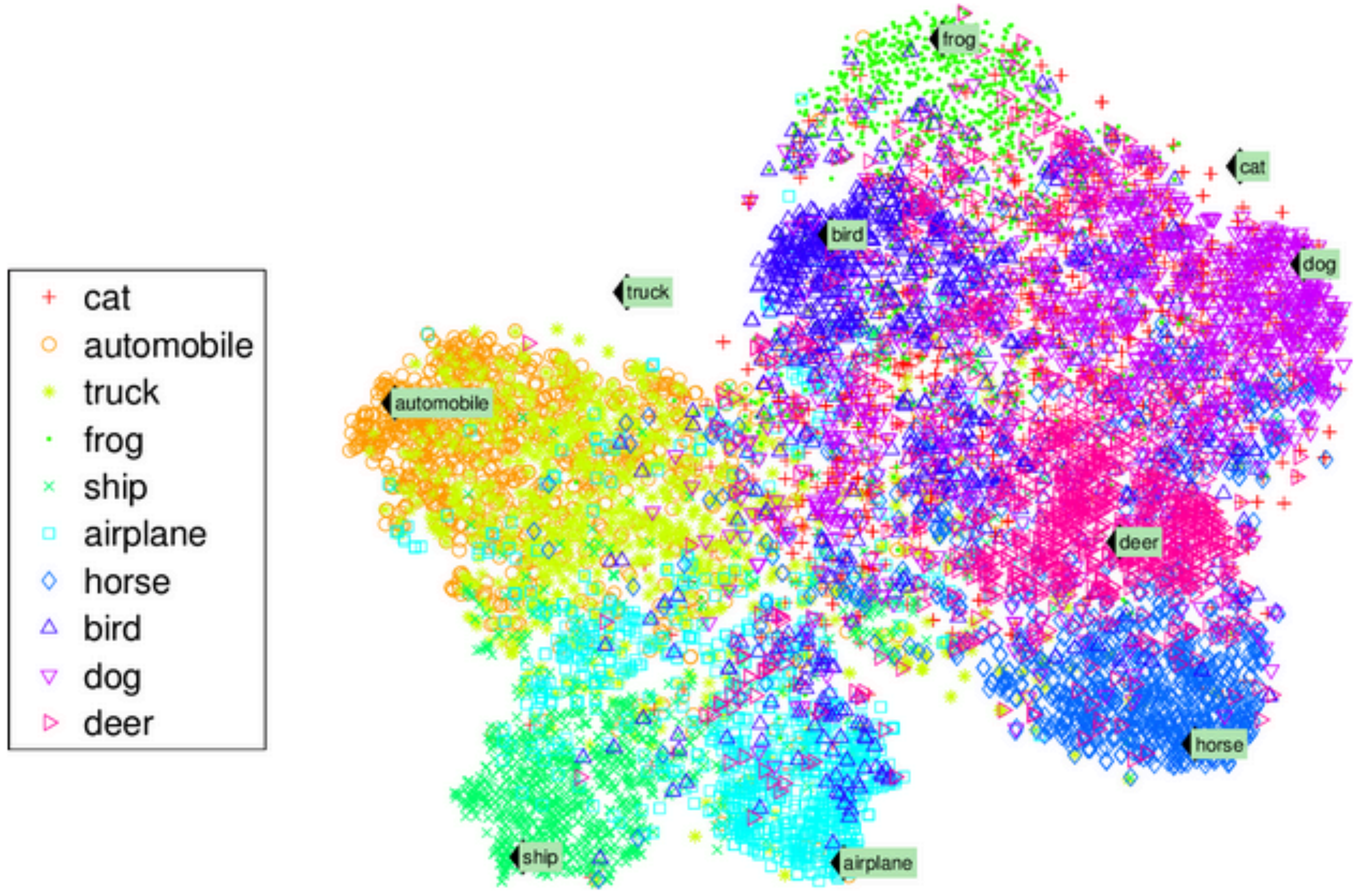
- The training data is unlabeled
  - The system tries to learn without a teacher
- Example: a lot of data about a blog's visitors
  - Run a clustering algorithm to try to detect groups of similar visitors
  - Never tell the algorithm which group a visitor belongs to
- Hierarchical clustering algorithm
  - May also subdivide each group into smaller groups
  - Help you target your posts for each group



source: Géron, A. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow"

Unsupervised learning

- Visualization algorithms
  - Receives a lot of complex and unlabeled data
    - Output a 2D or 3D representation of your data
      - Can easily be plotted
  - Try to preserve as much structure as they can
    - Allows to understand how the data is organized
      - Perhaps identify unsuspected patterns



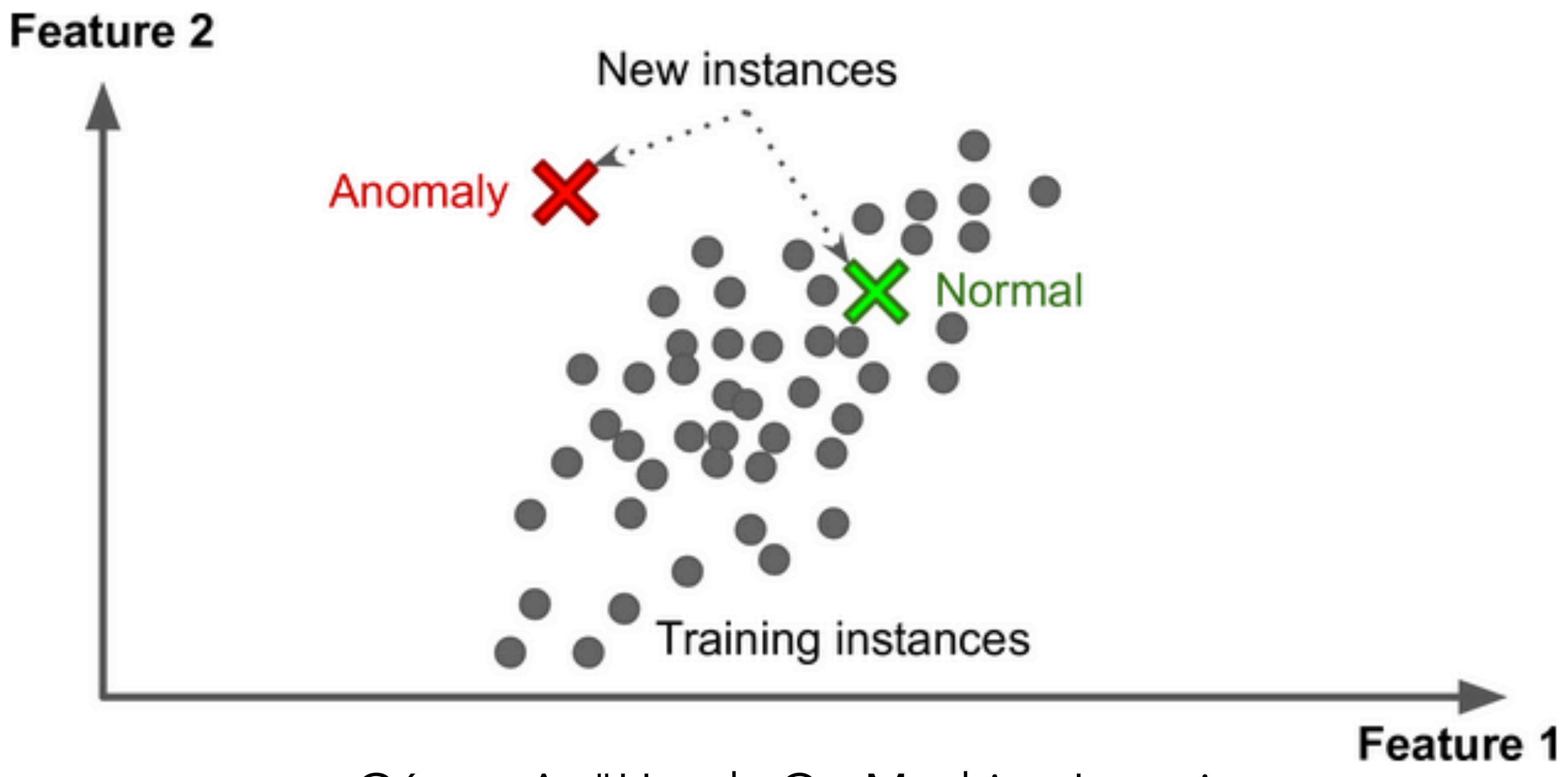
source: Géron, A. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow"

Dimensionality reduction

- Simplify the data without losing too much information
- Feature extraction
  - Merge several correlated features into one
  - Car’s mileage may be strongly correlated with its age
    - Merging them into one feature that represents the car’s wear and tear
- Reducing the dimension of your training data
  - Before you feed it to another Machine Learning algorithm
    - Run much faster
    - The data will take up less disk and memory space
    - It might even perform better

Anomaly detection

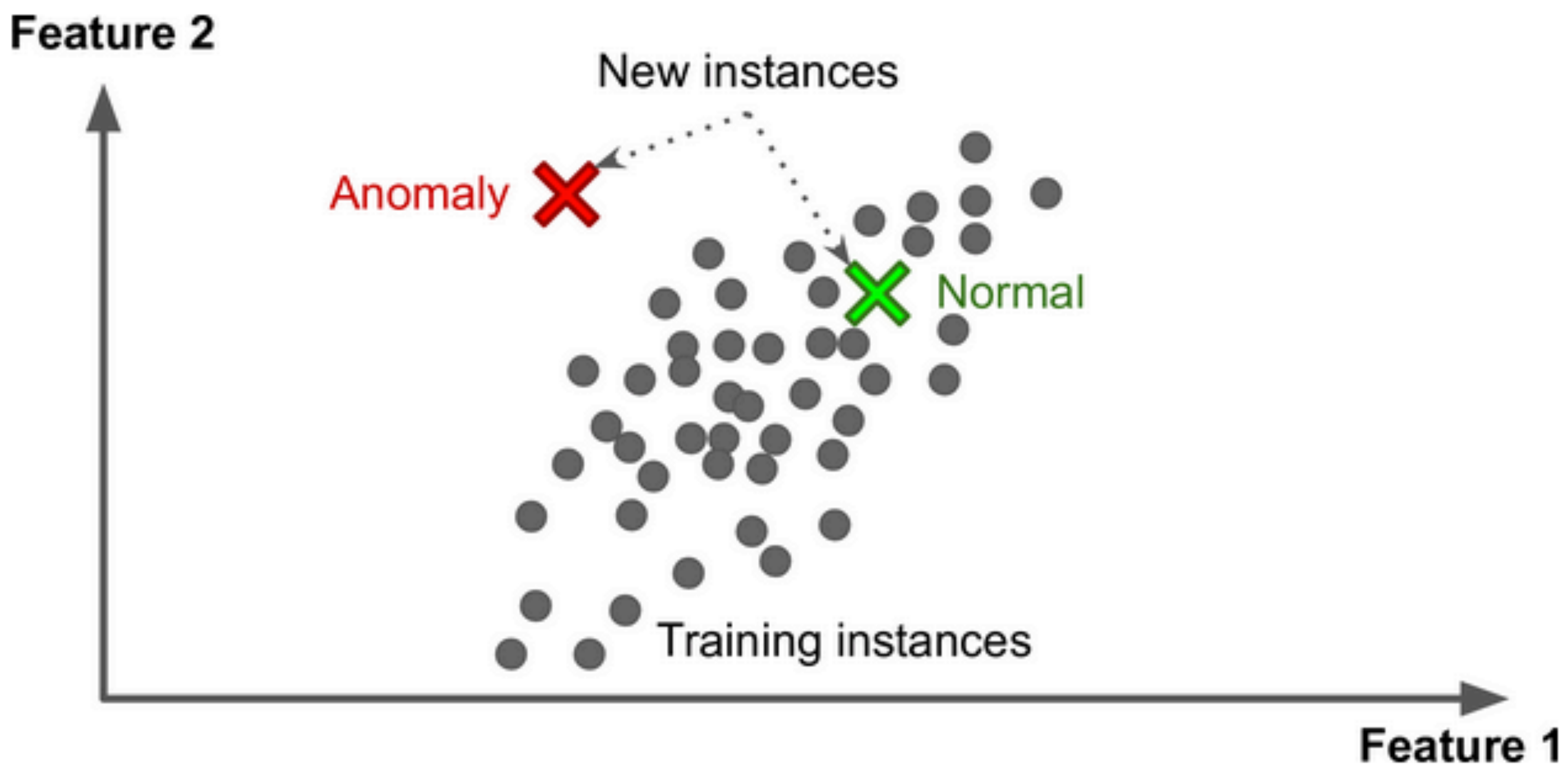
- Detecting unusual credit card transactions to prevent fraud
- Catching manufacturing defects
- Intrusion detection systems
- Automatically removing outliers from a dataset
- The system is shown mostly normal instances
  - Learns to recognize them
  - For new instances
    - Must tell whether it looks like a normal one or whether it is likely an anomaly



source: Géron, A. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow"

Anomaly detection

- Novelty detection
  - Detect new instances
    - Look different from all instances in the training set
- Imagine a thousands of pictures of dogs
    - 1% of these pictures represent Chihuahuas
    - Anomaly vs novelty



source: Géron, A. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow"

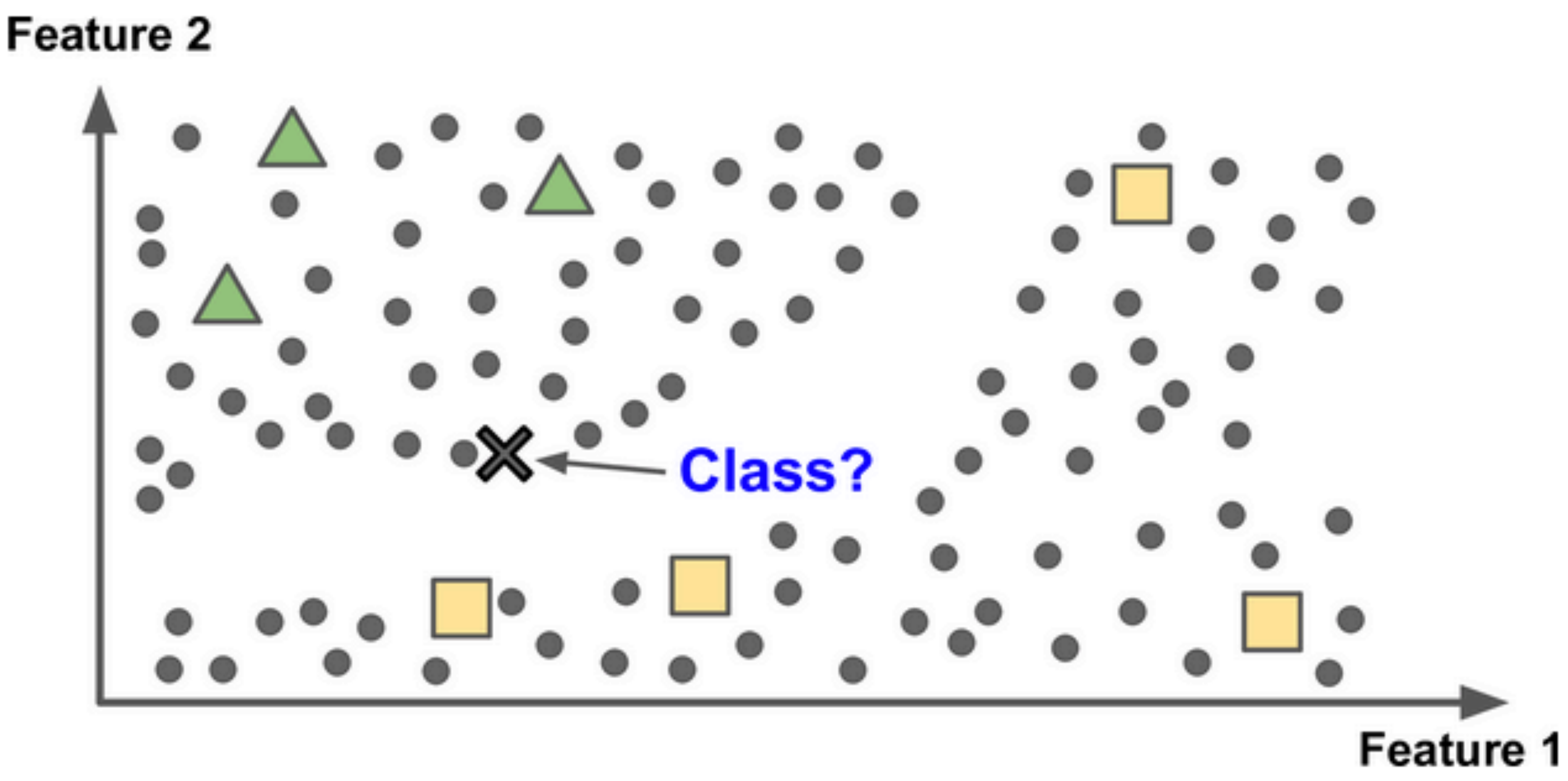
Unsupervised learning

- Rule learning
  - Dig into large amounts of data and discover interesting relations between attributes
  - Example: a supermarket
    - Running an association rule on the sales logs
      - May reveal that people who purchase
        - Barbecue sauce and potato chips also tend to buy steak
      - Place these items close to one another

Semi-supervised learning

- Labeling data is usually time-consuming and costly
  - Often —> plenty of unlabeled instances and few labeled instances
  - Some algorithms can deal with data that is partially labeled

- Semi-supervised learning with two classes
  - triangles and squares



source: Géron, A. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow"

Semi-supervised learning

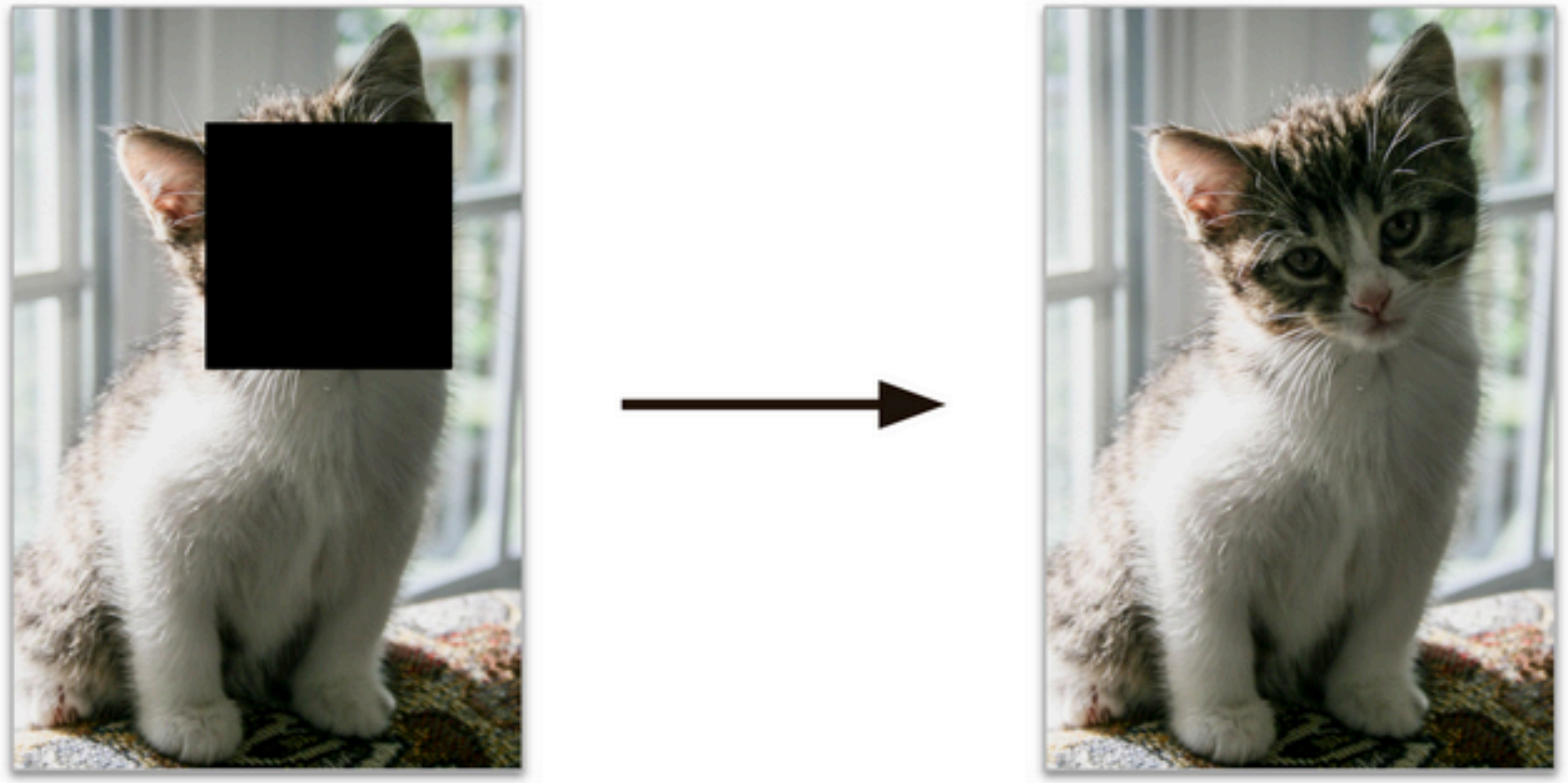
- Most semi-supervised learning algorithms are combinations of unsupervised and supervised algorithms
- A clustering algorithm may be used to group similar instances together
  - Every unlabeled instance can be labeled
    - With the most common label in their cluster
  - Once the whole dataset is labeled
    - It is possible to use any supervised learning algorithm

Semi-supervised learning

- Some photo-hosting services
  - Google Photos
  - The user upload all its family photos to the service
  - Unsupervised part of the algorithm — clustering
    - Automatically recognizes
      - The same person **A** shows up in photos 2, 4, and 10
      - Another person **B** shows up in photos 1, 5, and 7
  - Supervised part
    - The user adds one label per person
      - Able to name everyone in every photo

Self-supervised learning

- Involves generating a fully labeled dataset from a fully unlabeled one
- Once the whole dataset is labeled
  - Any supervised learning algorithm can be used
- With a large dataset of unlabeled images
  - Randomly mask a small part of each image
    - Train a model to recover the original image
    - Masked images are used as the inputs to the model
      - The original images are used as the labels



source: Géron, A. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow"

Self-supervised learning

- The resulting model may be quite useful
    - Repair damaged images or to erase unwanted objects from pictures
- More often a model trained using self-supervised learning is not the final goal
    - Adjust and fine-tune the model for a slightly different task

Self-supervised learning

- It can be considered to be a part of unsupervised learning
  - Deals with fully unlabeled datasets
- It uses generated labels during training
  - It is closer to supervised learning

Self-supervised learning

- "Unsupervised learning" is generally used when dealing with tasks
  - Clustering
  - Dimensionality reduction
  - Anomaly detection
- Self-supervised learning usually focuses on the same tasks as supervised learning
  - Classification and Regression

Transfer Learning

- Transferring knowledge from one task to another
- It's one of the most important techniques in ML today
    - Especially when using deep neural networks
        - Neural networks composed of many layers of neurones

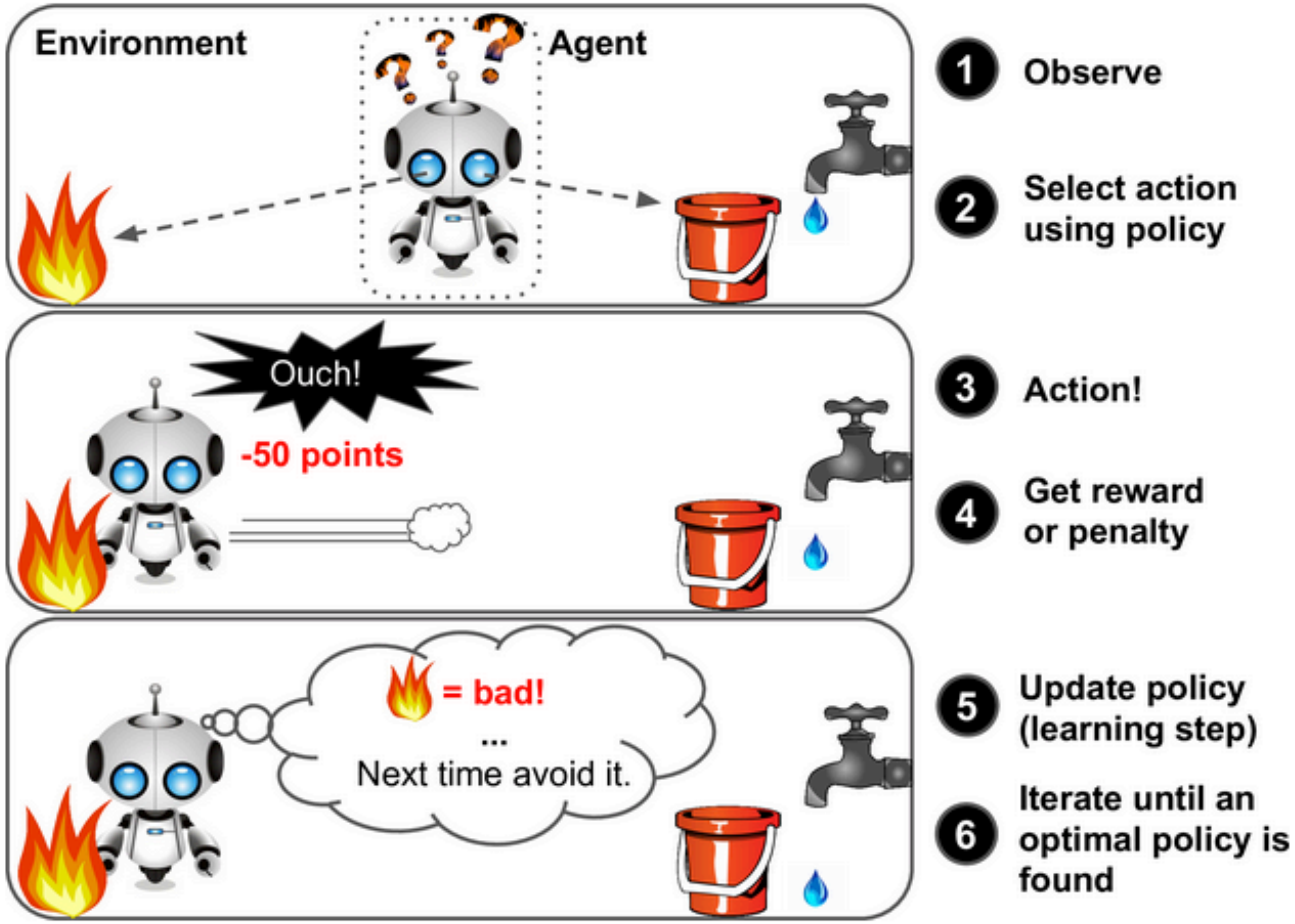
Reinforcement Learning

- Reinforcement Learning is a very different thing
- The learning system
    - An **agent** can
        - Observe the **environment**
        - Select and perform **actions**
        - Get **rewards** in return
            - Penalties in the form of negative rewards

Reinforcement Learning

- It must learn by itself what is the best strategy — policy
  - To get the most reward over time
  - A policy defines what action the agent should choose
    - When it is in a given situation

Reinforcement learning


Environment
Agent
Ouch!
-50 points
= bad!
...
Next time avoid it.
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

source: Géron, A. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow"

Reinforcement learning

- Many robots implement RL algorithms to learn how to walk
- DeepMind’s AlphaGo program
  - Beat the world’s number one ranked player at the game of Go (2017)
  - It learned its winning policy by analyzing millions of games
  - Playing many games against itself
  - Offline learning

Batch versus Online Learning

- Another criterion used to classify Machine Learning systems
  - The capability of learning incrementally from a stream of incoming data
- Batch learning
- Online learning

Batch learning

- The system is incapable of learning incrementally
  - It must be trained using all the available data
  - Generally takes a lot of time and computing resources
    - Typically done offline
- Offline learning
  - First the system is trained
  - It is launched into production and runs without learning anymore
  - It just applies what it has learned

Model's performance

- Model rot or data drift phenomenon
  - Model's performance tends to decay slowly over time
    - The world continues to evolve while the model remains unchanged
  - The solution is to regularly retrain the model on up-to-date data

Model's performance

- How often you need to do that depends on the use case
  - A model that classifies pictures of cats and dogs
    - Its performance will decay very slowly
  - A model that deals with fast-evolving systems
    - Intrusion Detection System (IDS)
      - It is likely to decay quite fast

Model's performance

- Even a model trained to classify pictures of cats and dogs
    - May need to be retrained regularly
        - Cameras keep changing
            - Different image formats, sharpness, brightness, and size ratios
        - People may
            - Love different breeds next year
            - Decide to dress their pets with tiny hats

Adapting to change

- For a batch learning system to know about new data
- Need to train a new version of the system from scratch on the full dataset
  - Not just the new data
  - Replace the old model with the new one
- The whole process of training, evaluating, and launching a ML system
  - Can be automated fairly easily
- Even a batch learning system can adapt to change
  - Simply update the data and train a new version of the system from scratch as often as needed

Adapting to change

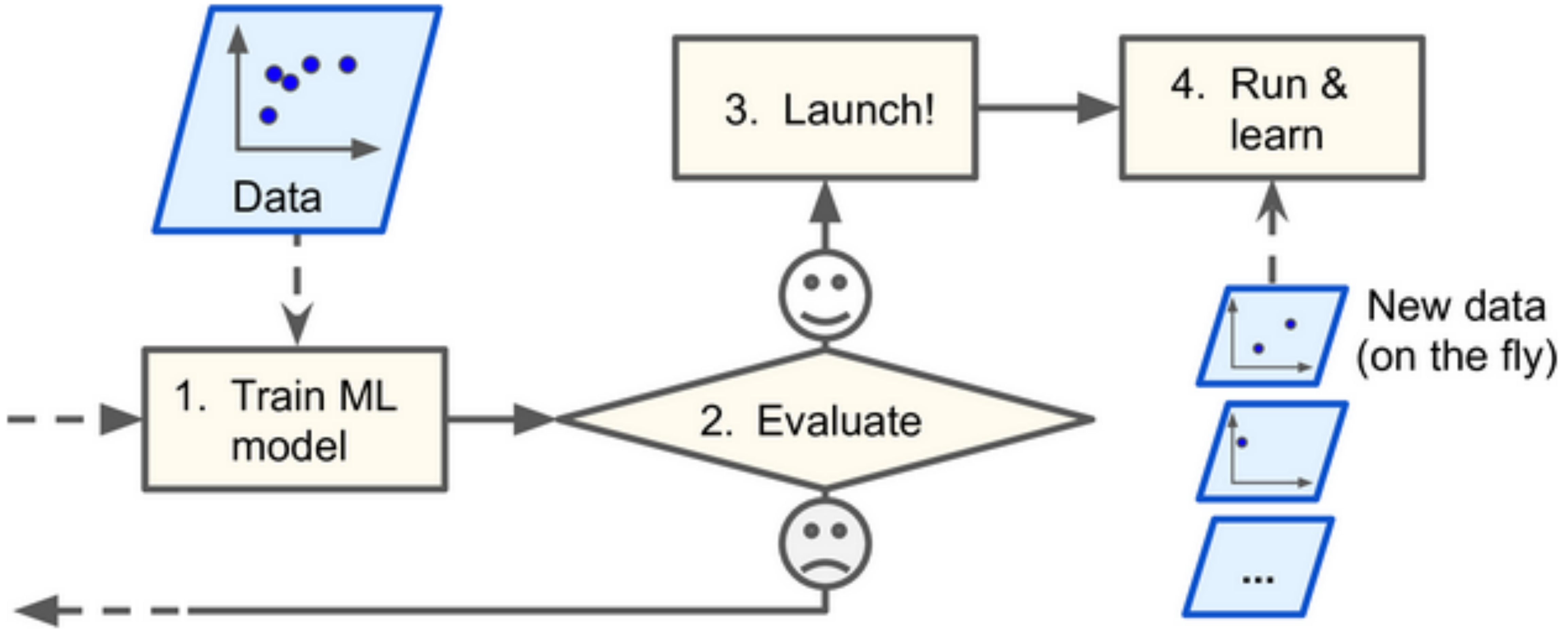
- This solution is simple and often works fine
  - Training using the full set of data can take many hours
    - Typically train a new system only every 24 hours or even just weekly
  - Adapt to rapidly changing data demands a more reactive solution
    - Predict stock prices

Adapting to change

- Training on the full set of data requires a lot of computing resources
  - CPU, memory space, disk space, disk I/O, network I/O
  - For huge amounts of data
    - Batch learning algorithms may not be feasible
  - Batch learning for limited resources systems might also be prohibitive
    - IoT applications, a smartphone application or a rover on Mars
      - Carrying around large amounts of training data
      - Taking up a lot of resources to train for hours every day

Online learning

- Train the system incrementally by feeding it data instances sequentially
  - Individually or in small groups called mini-batches.
- Each learning step is fast and cheap
  - The system can learn about new data on the fly, as it arrives

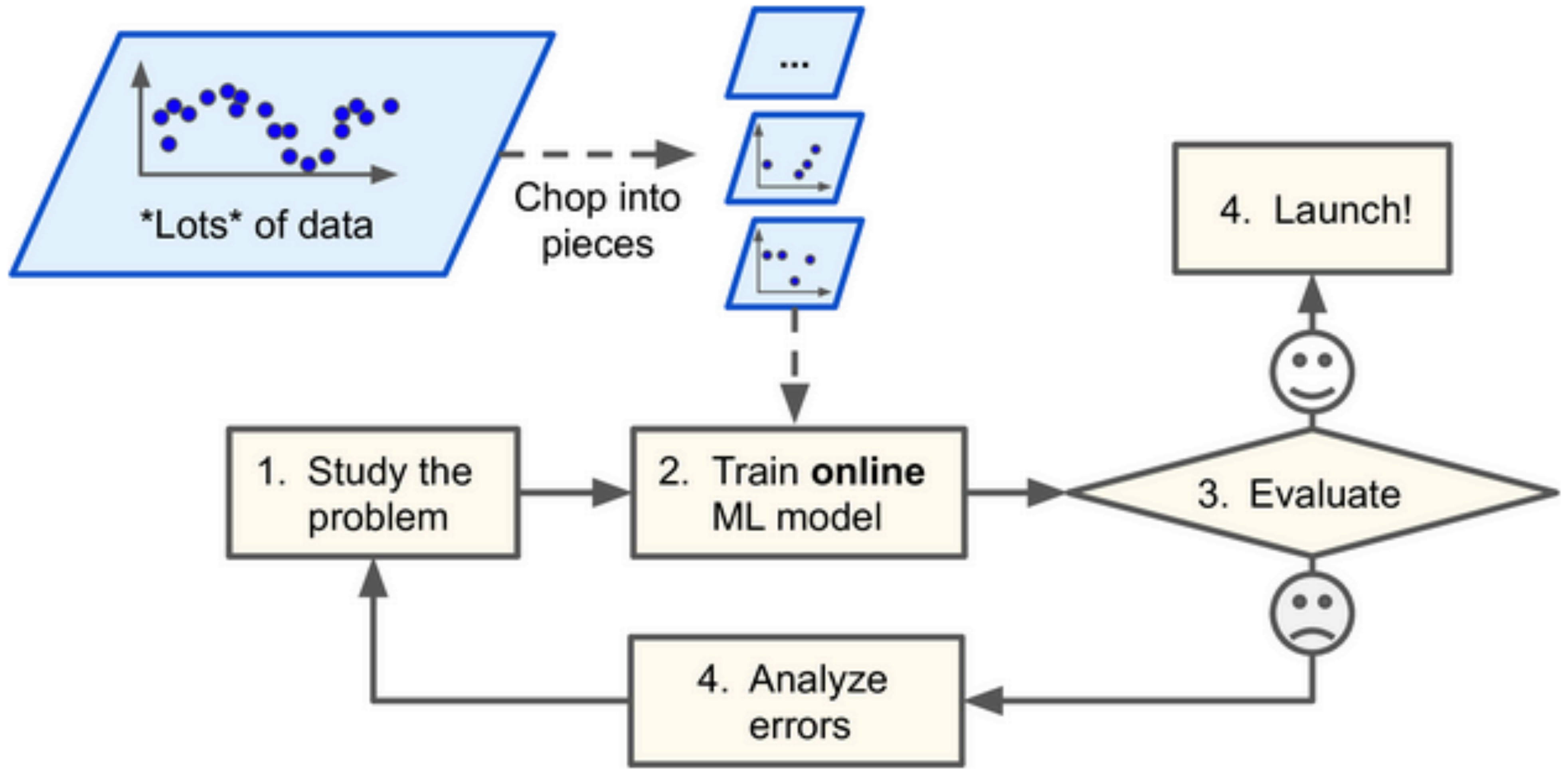


source: Géron, A. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow"

Online learning

- Useful for systems
  - Need to adapt to change extremely rapid
  - Have limited computing resources
    - Mobile devices
- Also used to train models on huge datasets
  - Cannot fit in one machine's main memory
  - Online learning can be a confusing name
    - Think of it as incremental learning

Using online learning to handle huge datasets



source: Géron, A. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow"

Online learning

- Learning rate
  - How fast it should adapt to changing data
  - High learning rate
    - The system will rapidly adapt to new data
    - Tend to quickly forget the old data
  - Low learning rate
    - The system will have more inertia
    - Less sensitive to noise in the new data or to sequences of non-representative data points (outliers)

Big challenge in online learning

- Bad data is fed to the system,
  - The system’s performance will decline
    - Depending on the data quality and learning rate
  - In a live system —> clients will notice
- Bad data could come from a bug
  - A malfunctioning sensor on a robot
- Someone trying to game the system

Big challenge in online learning

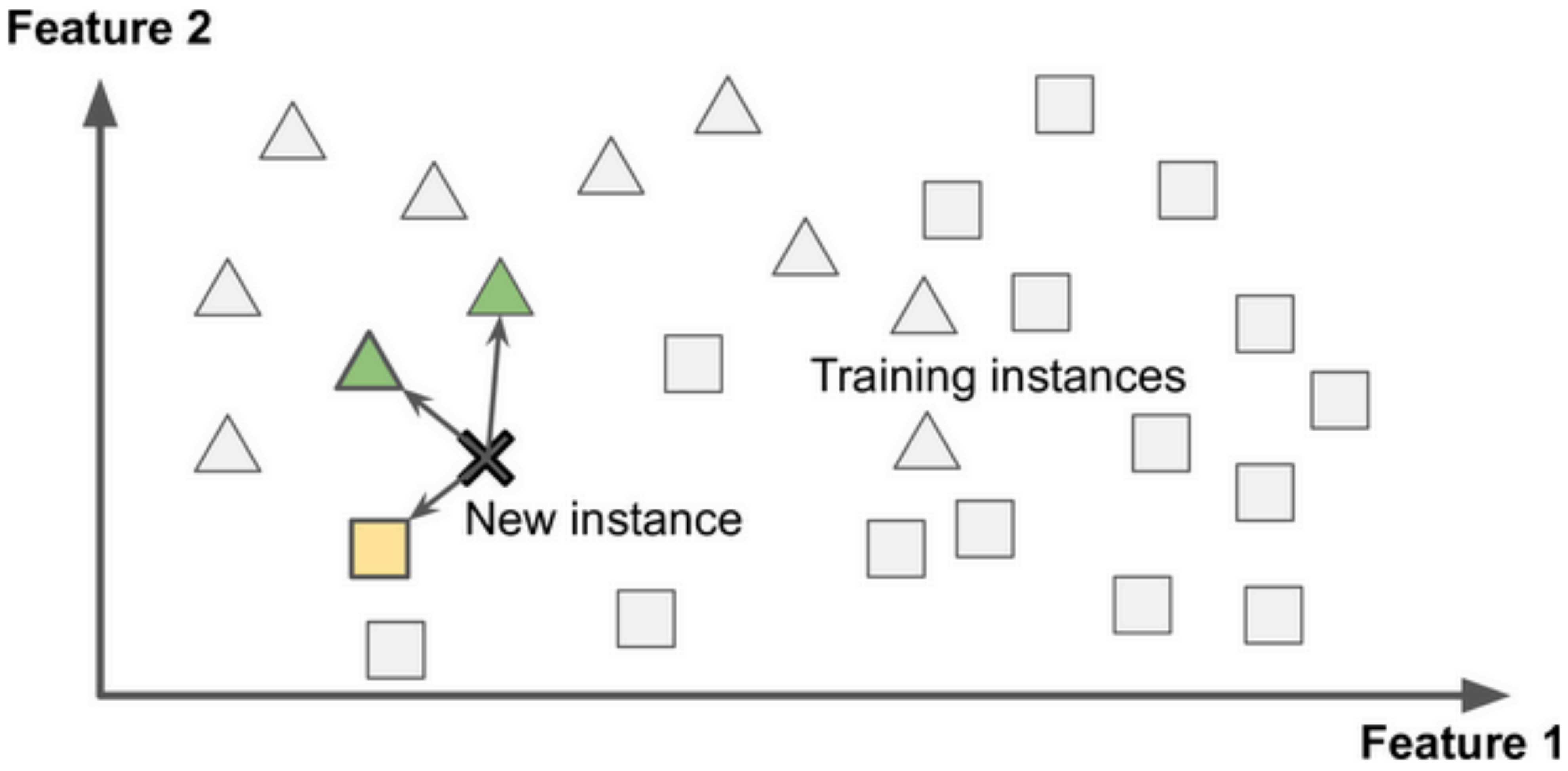
- To reduce this risk
  - We need to monitor your system closely
- A drop in performance is detected
  - Promptly switch learning off
  - Possibly revert to a previously working state
- Monitor the input data and react to abnormal data
  - Using an anomaly detection algorithm

ML system generalization

- Most ML tasks are about making predictions
  - Given a number of training examples
    - The system needs to be able to make good predictions
      - For inputs it has never seen before → generalize
      - Having a good performance measure on the training data is good
      - The true goal is to perform well on new instances
- Two main approaches
  - Instance-based learning
  - Model-based learning

Instance-based learning

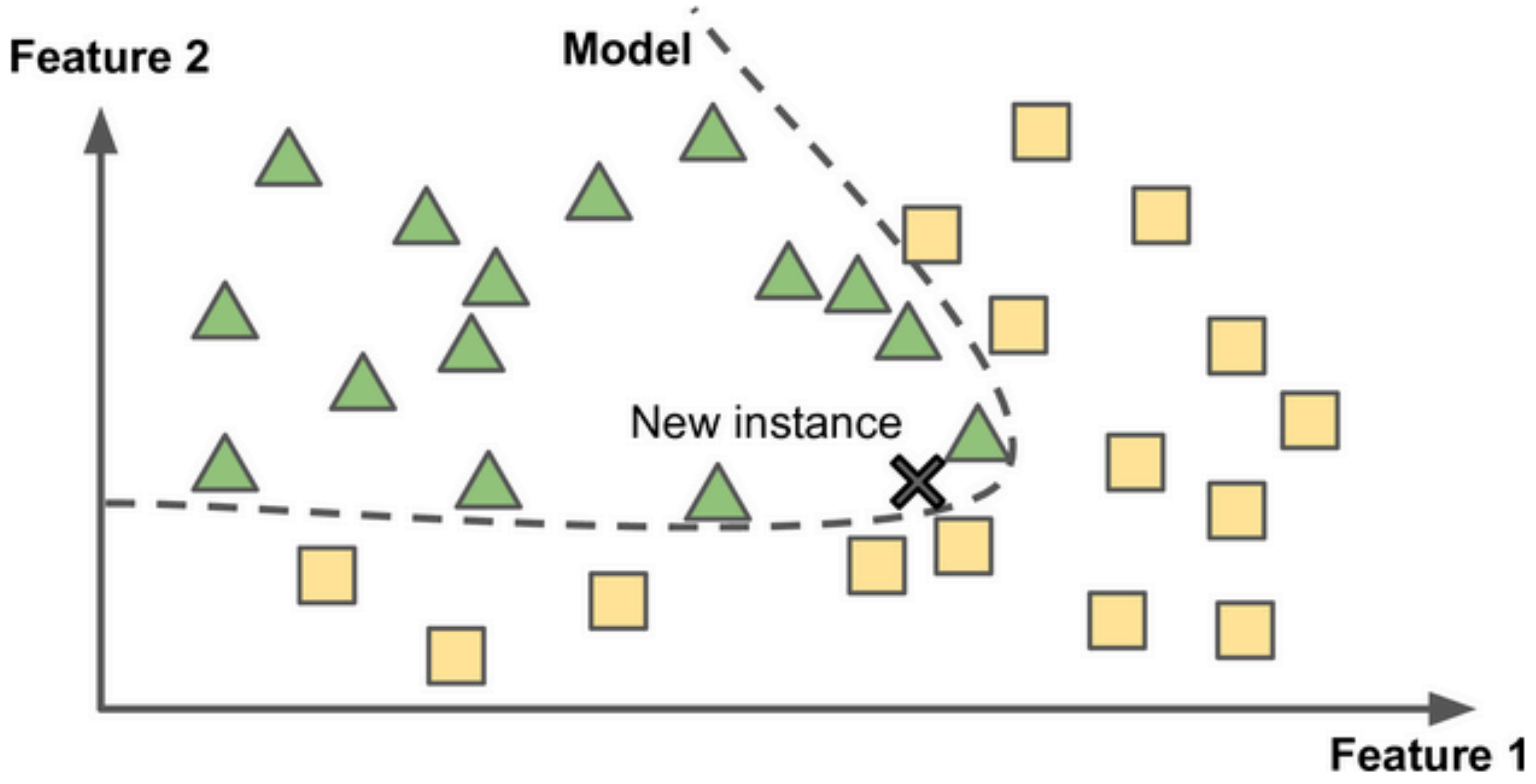
- Simplest solution: Learn by heart
  - Spam filter
    - Flag all emails that are identical to emails that have already been flagged by users
- The spam filter could be programmed to also flag emails that are very similar to known spam emails
  - Requires a measure of similarity between two emails
  - A very basic similarity measure between two emails
    - Count the number of words they have in common
    - The system would flag an email as spam
      - A certain number of common words



source: Géron, A. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow"

Model-based learning

- Build a model of the examples
    - Use that model to make predictions

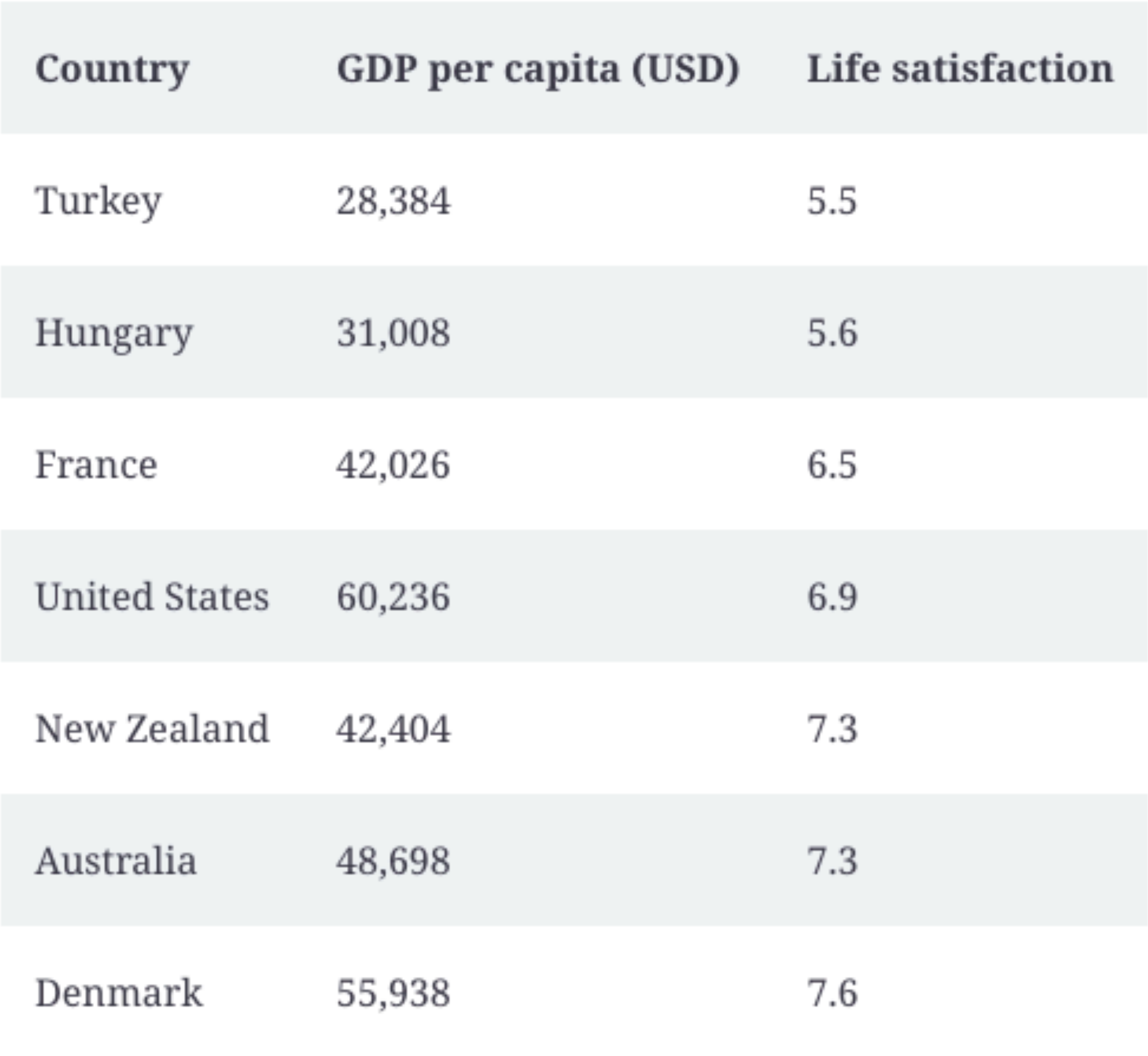


source: Géron, A. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow"

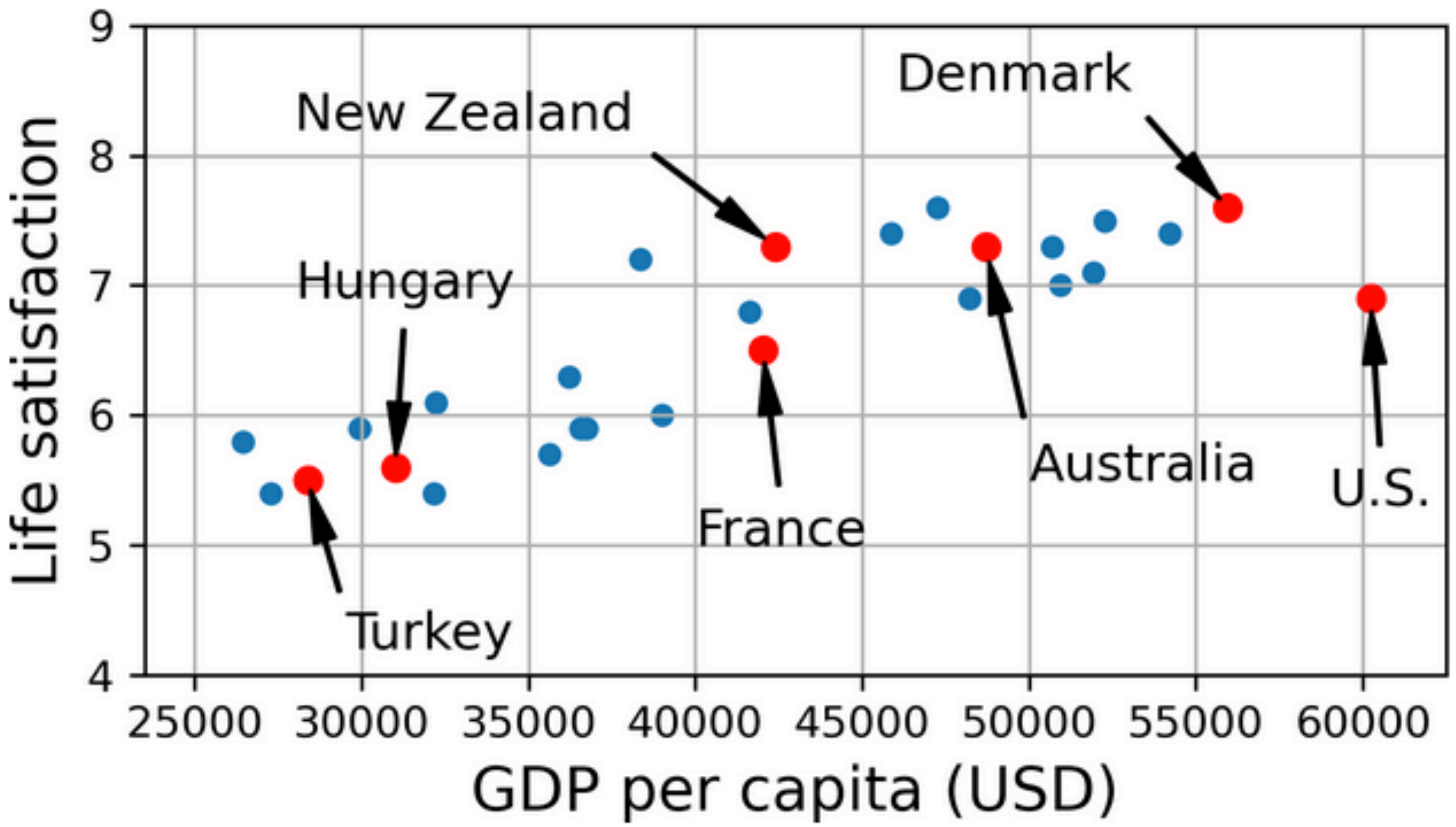
A typical Machine Learning workflow

- Money makes people happy?
- Download
  - The Better Life Index data from the OECD's website
  - The World Bank stats about Gross Domestic Product (GDP) per capita
  - Join the tables and sort by GDP per capita

GDP per capita



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
New Zealand	42,404	7.3
Australia	48,698	7.3
Denmark	55,938	7.6



Although the data is noisy — partly random

source: Géron, A. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow"

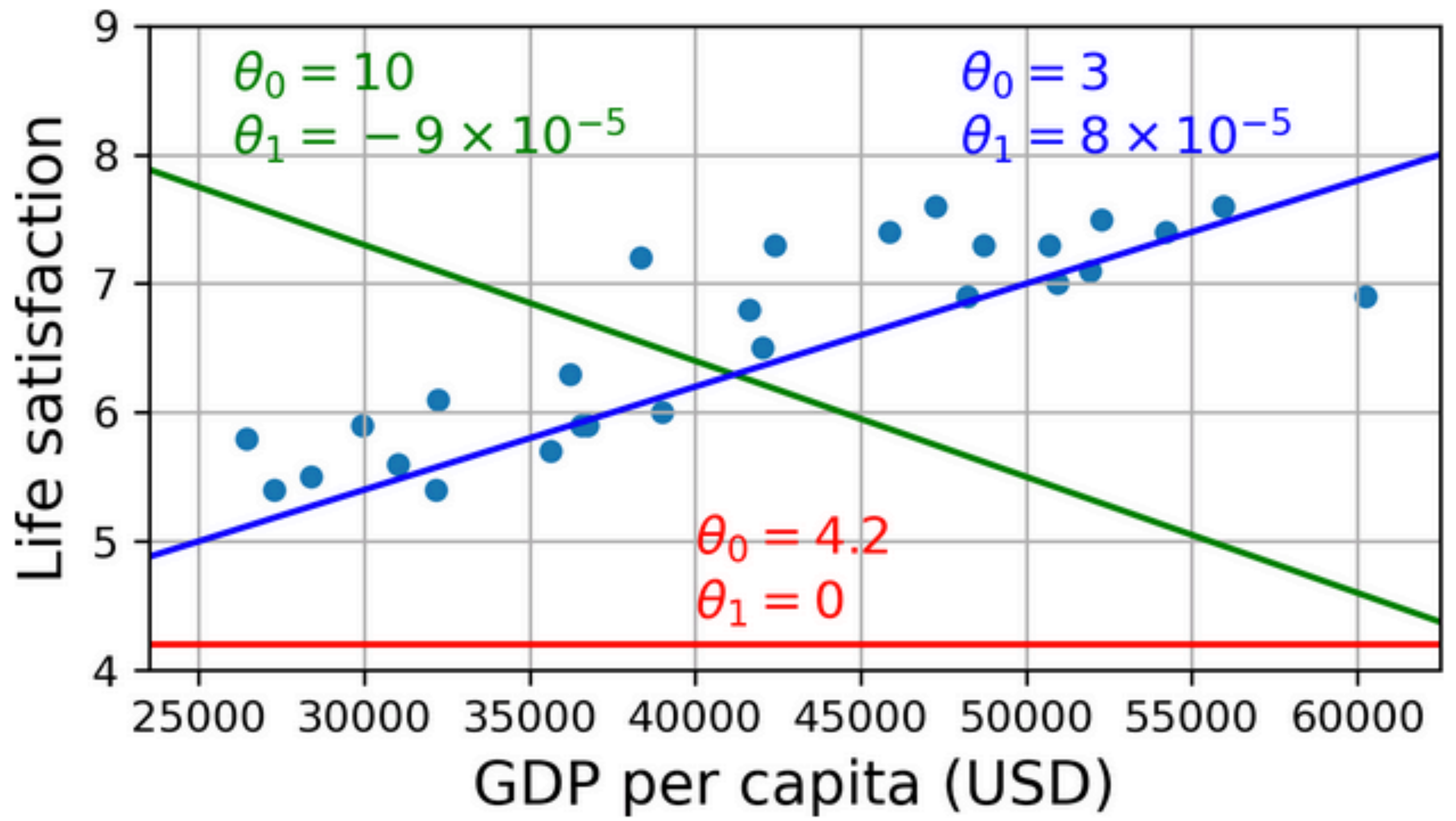
Model selection

- It looks like life satisfaction goes up more or less linearly
    - As the country's GDP per capita increases
    - Linear model
        - One attribute —> GDP

$$life\_satisfaction = \theta_0 + \theta_1 \times GPD\_per\_capita$$

Linear model

- The model has two model parameters
  - θ_0 and θ_1
- By tweaking these parameters
- The model can represent any linear function



source: Géron, A. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow"

Training the model

- Defining the parameter values θ_0 and θ_1

How can we know which values will make your model perform best?

- Specify a performance measure
  - Utility function
    - How good the model is
  - Cost function
    - How bad the model is

Linear regression

- Linear Regression problems
  - Typically use a cost function
    - Measures the distance between the linear model's predictions
      - The training examples

The optimal parameter values are $\theta_0 = 3.75$ and $\theta_1 = 6.78 \times 10^{-5}$

- Train
  - Linear Regression algorithm
    - Feed it our training examples
    - Finds the parameters that make the linear model fit best to our data

Warning

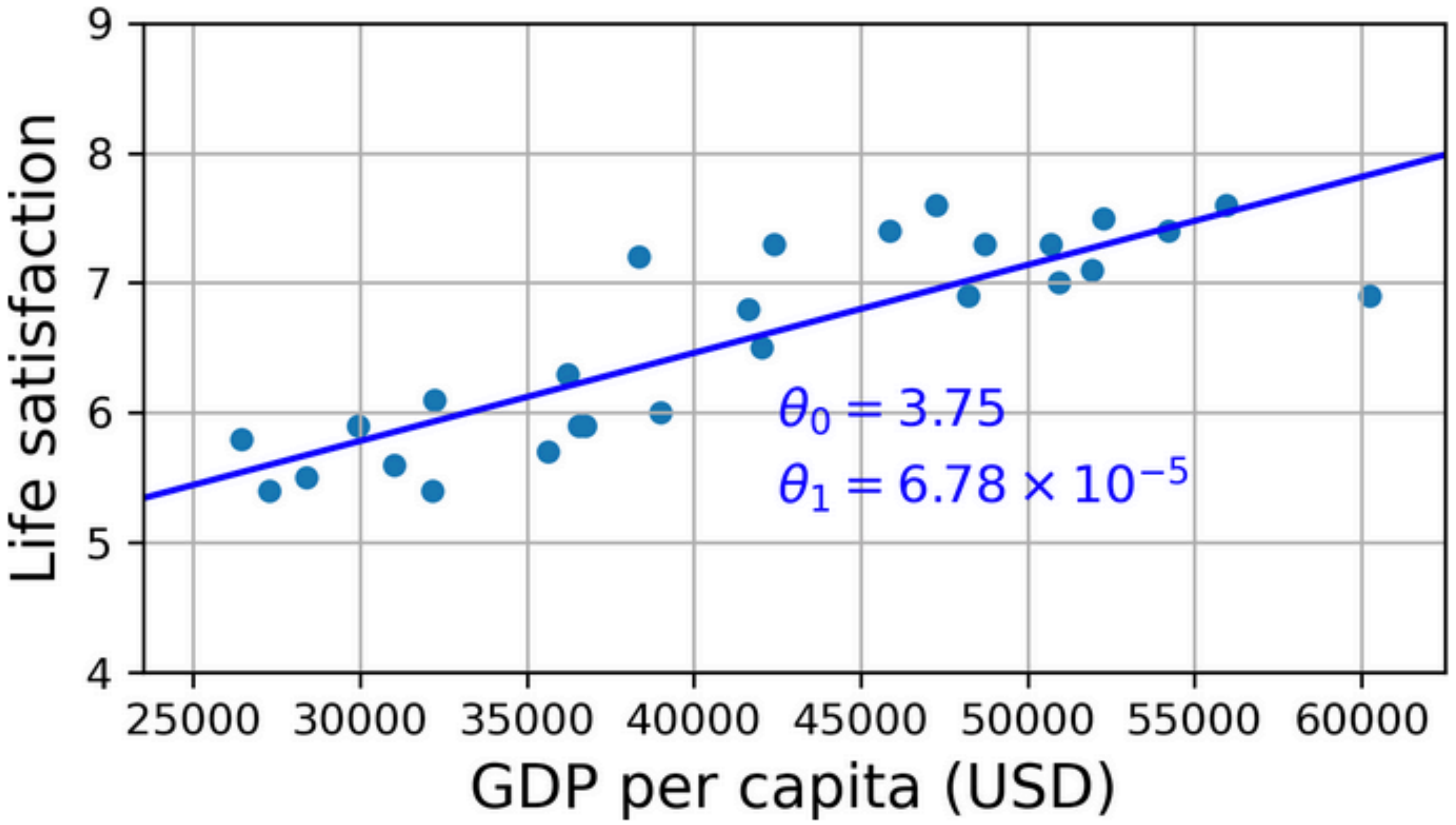
- The same word “model” can refer to
  - A type of model
    - Linear Regression
  - Fully specified model architecture
    - Linear Regression with one input and one output
  - The final trained model ready to be used for predictions
    - Linear Regression with one input and one output
    - Using $\theta_0 = 3.75$ and $\theta_1 = 6.78 \times 10^{-5}$

Warning

- Model selection consists in choosing
  - The type of model and fully specifying its architecture
- Training a model
  - Running an algorithm to find the model parameters
    - Best fit the training data and make good predictions on new data

Predictions

- The model fits the training data as closely as possible
  - Linear model

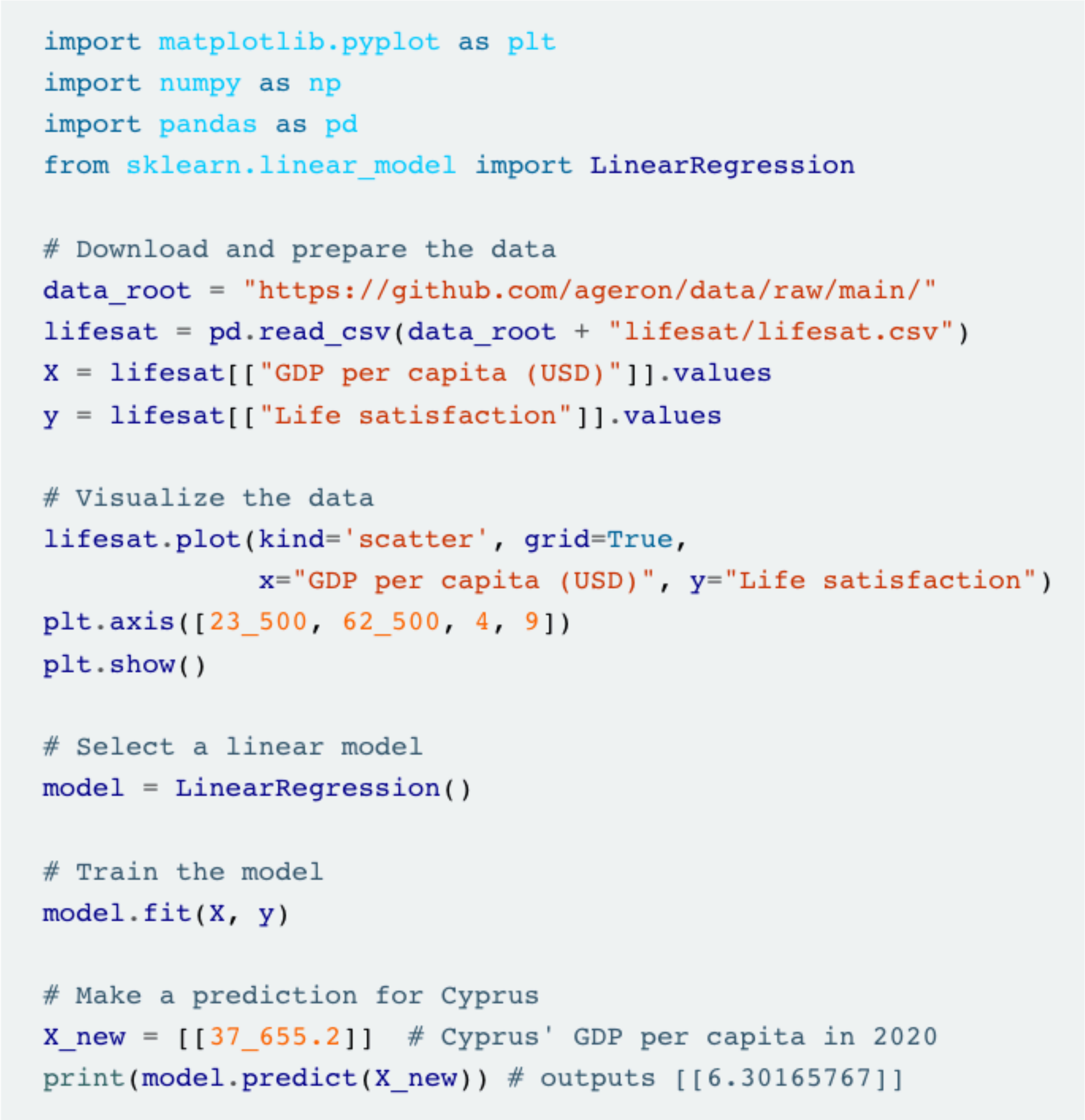


source: Géron, A. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow"

- We are ready to run the model to make predictions
- How happy Cypriots are?
  - Cyprus's GDP per capita —> $37,655
  - Applying to the model
    - Life satisfaction —> 3.75 + 37,655 × 6.78 × 10^{-5} = 6.30

Python code example

- Loads the data
- Separates the inputs **X** from the labels **y**
- Creates a scatterplot for visualization
- Trains a linear model
- Makes a prediction



```python
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression

# Download and prepare the data
data_root = "https://github.com/ageron/data/raw/main/"
lifesat = pd.read_csv(data_root + "lifesat/lifesat.csv")
X = lifesat[["GDP per capita (USD)"]].values
y = lifesat[["Life satisfaction"]].values

# Visualize the data
lifesat.plot(kind='scatter', grid=True,
             x="GDP per capita (USD)", y="Life satisfaction")
plt.axis([23_500, 62_500, 4, 9])
plt.show()

# Select a linear model
model = LinearRegression()

# Train the model
model.fit(X, y)

# Make a prediction for Cyprus
X_new = [[37_655.2]]  # Cyprus' GDP per capita in 2020
print(model.predict(X_new)) # outputs [[6.30165767]]
```

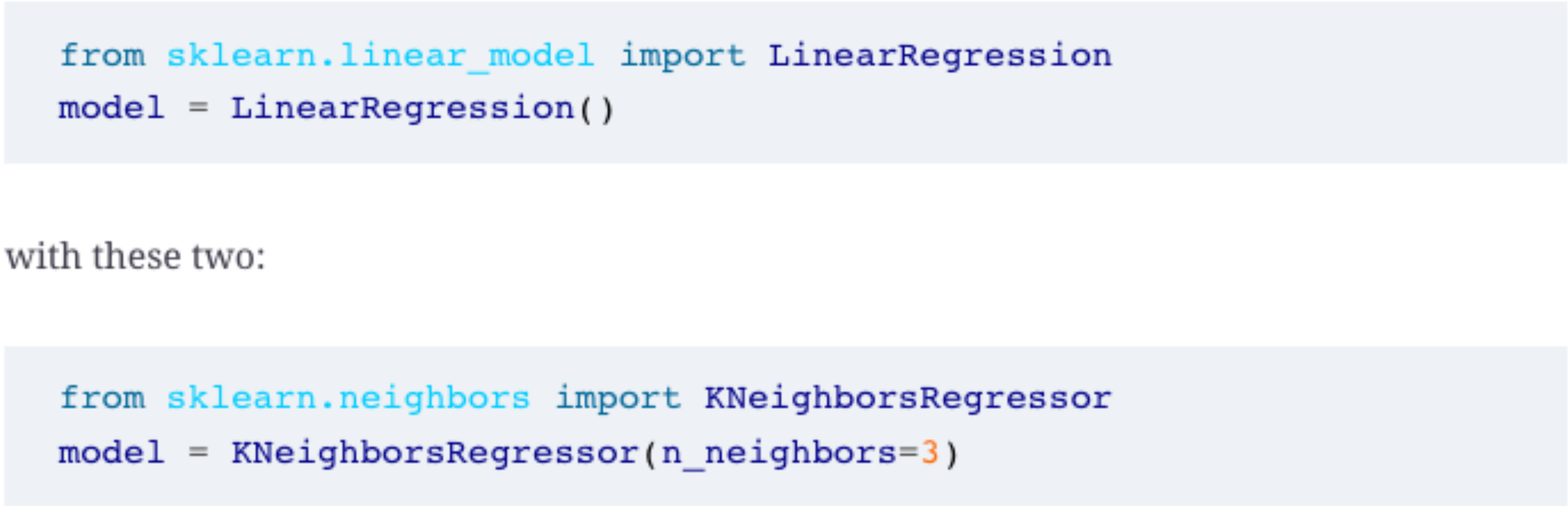
source: Géron, A. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow"

Another alternative

- With an instance-based learning algorithm
  - Uruguay has the closest GDP per capita to that of Cyprus ($38,341)
    - Uruguay’ life satisfaction is 7.2
      - Life satisfaction of Cyprus —> 7.2
  - Take a look at the two next-closest countries
    - Lithuania and Slovenia —> life satisfaction of 5.9
      - Averaging these three values —> 6.33
        - Close to our model-based prediction
        - This simple algorithm is called k-Nearest Neighbors regression, **k** = 3

K-Nearest Neighbours regression code

- Replacing the Linear Regression model
  - k-Nearest Neighbors regression in the previous code
    - Simple as replacing these two lines:



```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
```

with these two:

```
from sklearn.neighbors import KNeighborsRegressor
model = KNeighborsRegressor(n_neighbors=3)
```

source: Géron, A. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow"

In summary

- Studied the data
- Selected a model
- Train it on the training data
  - The learning algorithm searches for the model parameter values
    - Minimize a cost function
- Applied the model to make predictions on new cases
  - Inference
    - Hoping that the model will generalize well

Main challenges

- Main task is to
  - Select a model
  - Train it on some data
- What can go wrong
  - “bad model”
  - “bad data”

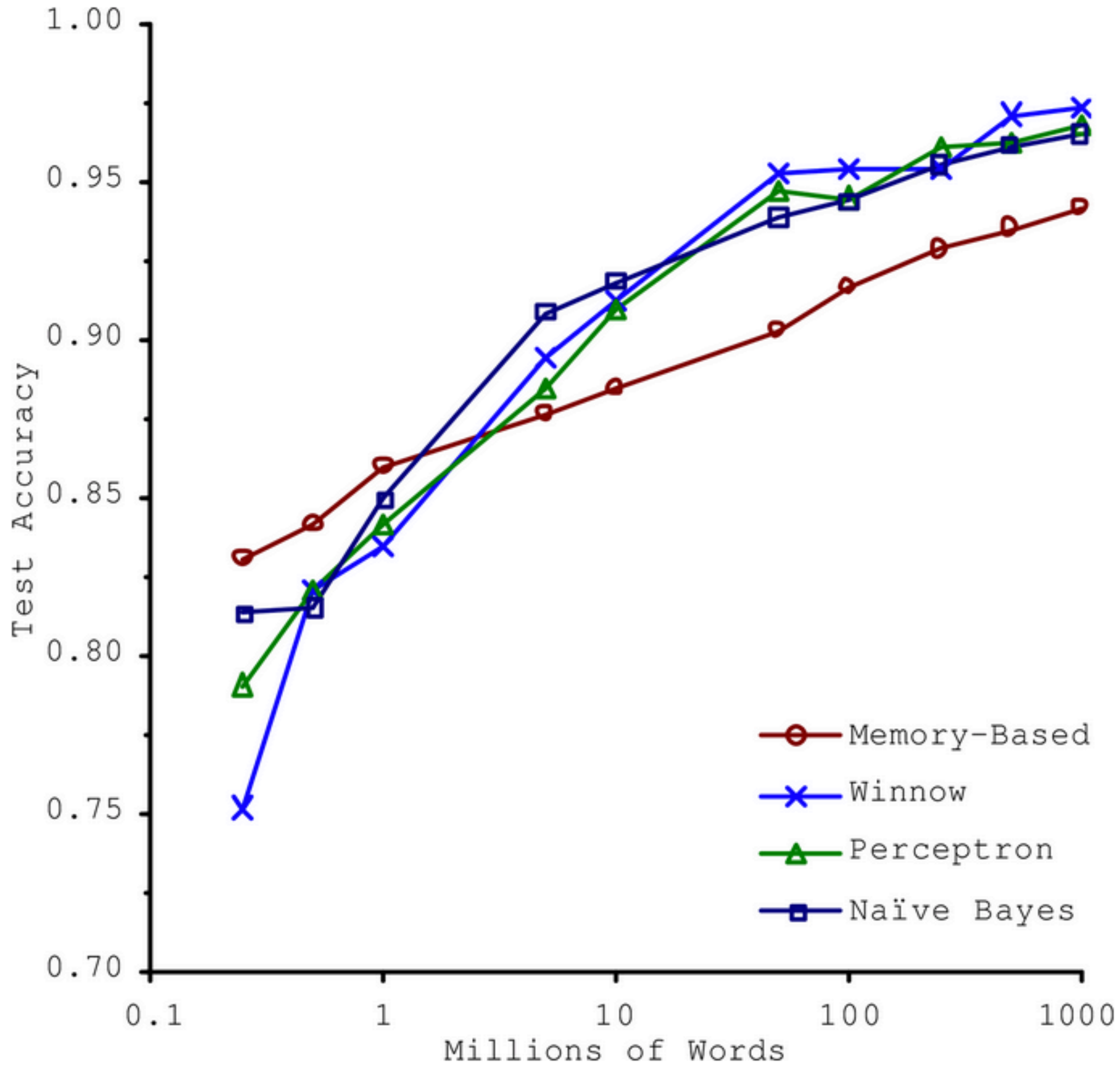
What might lead to “bad data”?

Insufficient Quantity of Training Data

- For a child to learn what an apple is
  - Point to an apple and say “apple”
  - Repeating this procedure a few times
    - The child will be able to recognize apples
      - All sorts of colors and shapes
- Machine Learning algorithms takes a lot of data to work properly
  - Very simple problems
    - Need thousands of examples
  - Complex problems such as image or speech recognition
    - Need millions of examples

The Unreasonable Effectiveness of Data

- Michele Banko, Eric Brill, "Scaling to Very Very Large Corpora for Natural Language Disambiguation", ACL 2001
  - Showed that very different Machine Learning algorithms
    - Including fairly simple ones
    - Performed almost identically well
      - A complex problem of natural language disambiguation
      - When given enough data



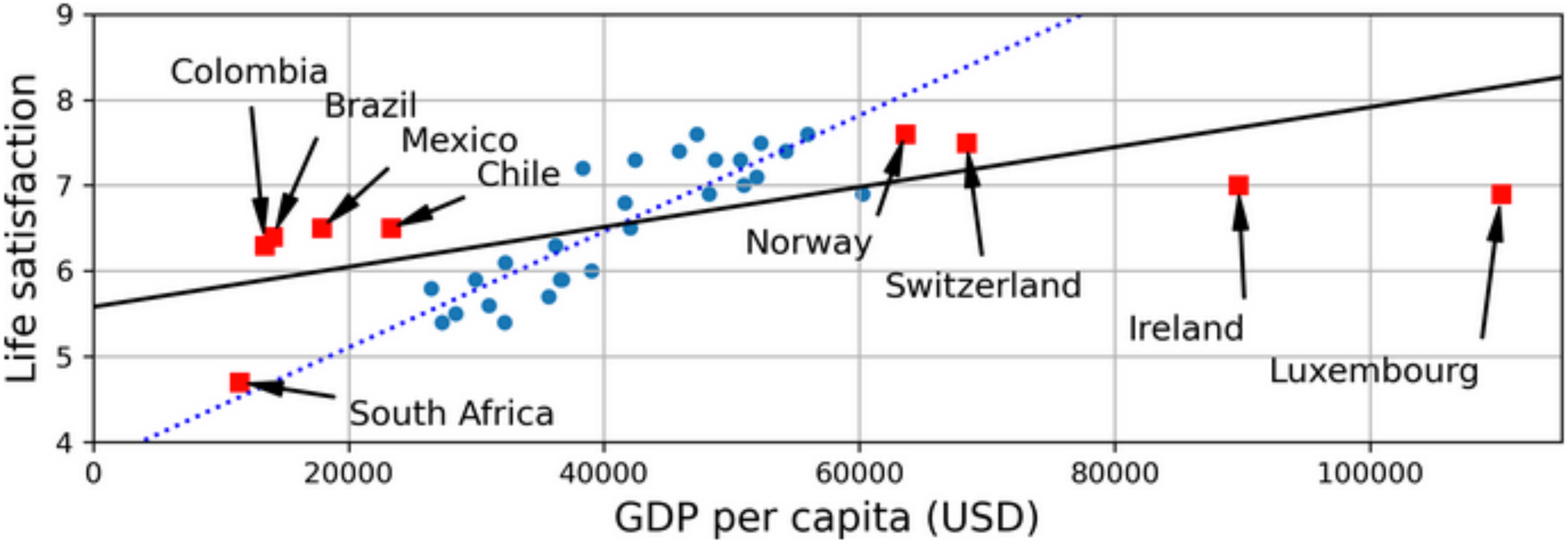
source: Géron, A. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow"

Nonrepresentative Training Data

- It is crucial that your training data be representative
  - New cases you want to generalize
- Example
  - The set of countries we used earlier for training the linear model was not perfectly representative
    - Any country with a GDP per capita
      - Lower than $23,500 or higher than $62,500

Nonrepresentative Training Data

- Does the simple linear model work well?



source: Géron, A. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow"

Nonrepresentative training set

- We trained a model that is unlikely to make accurate predictions
    - Especially for very poor and very rich countries
- Using a training set
    - Representative of the cases you want to generalize
    - This is often harder than it sounds
        - Sample too small —> sampling noise
        - Very large samples can be non-representative
            - Sampling method is flawed —> sampling bias

Sampling bias example

- One of the most famous example of sampling bias
  - During the US presidential election in 1936 — > Landon against Roosevelt
  - A very large poll was con
    - Sending
      - It got 2 wers
      - Predicted with high confidence that Landon would get 57% of the votes
      - Roosevelt won with 62% of the votes

Where is the flaw?

Sampling bias example

- To obtain the addresses to send the polls to
  - Used telephone directories
  - Lists of magazine subscribers
  - Club membership lists
  - All of these lists tended to favor wealthier people
    - More likely to vote Republican —> Landon
- Non-response bias
  - Less than 25% answers
    - Potentially ruling out different key groups

Sampling bias example

- Build a system to recognize funk music videos
  - Build your training set
    - Search for “funk music” on YouTube
    - The search results are likely to be biased toward popular artists
      - “Funk carioca” videos
        - Sound nothing like James Brown

Poor-Quality Data

- Training data is full of errors, outliers, and noise
  - Due to poor-quality measurements
  - Harder for the system to detect the underlying patterns
    - Less likely to perform well

Poor-Quality Data

- Spend time cleaning up your training data
  - Instances clearly outliers
    - Simply discard them or try to fix the errors manually
  - Missing features in some instances
    - 5% of your customers did not specify their age
      - Ignore this attribute altogether
      - Ignore these instances
      - Imputation — > Fill in the missing values
      - Train one model with the feature and one model without it

Irrelevant Features

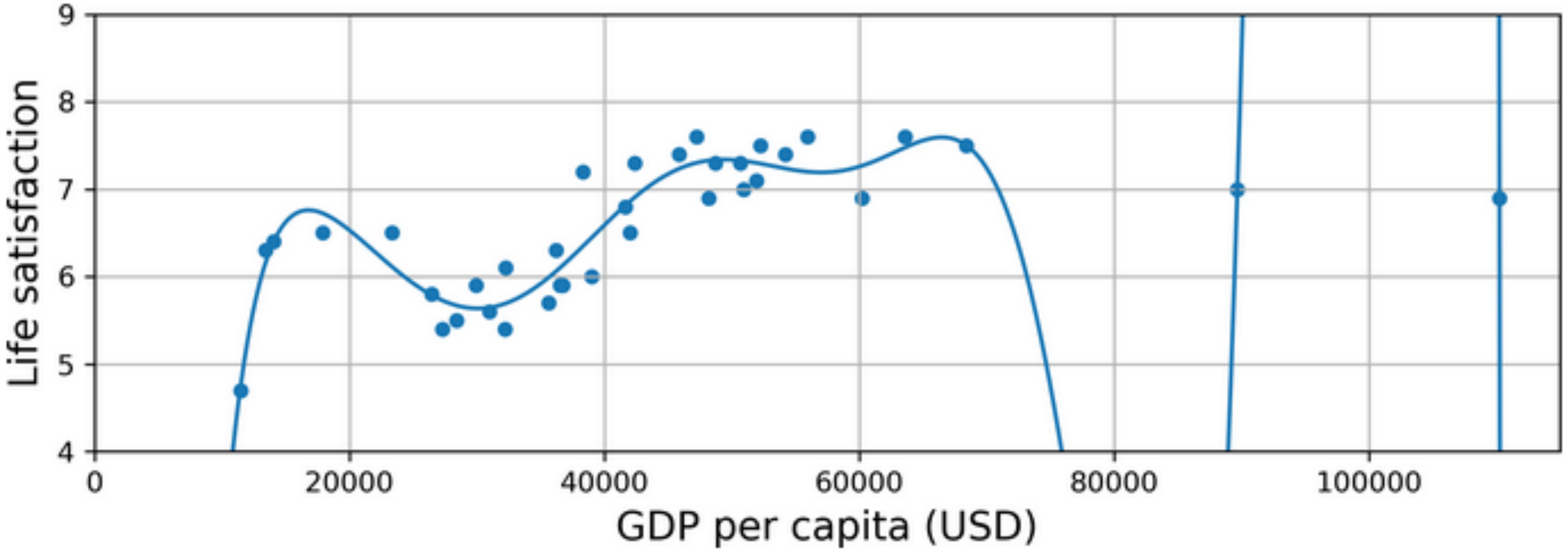
- Feature engineering
  - Finding a good set of features to train on
  - Feature selection
    - Selecting the most useful features to train on among existing features
  - Feature extraction
    - Combining existing features to produce a more useful one
    - Dimensionality reduction algorithms can help
  - Creating new features by gathering new data

Overfitting the Training Data

- Overgeneralizing —> overfitting
  - When visiting a foreign country and the taxi driver rips you off
    - Tempted to say that all taxi drivers in that country are not cool
  - It means that the model performs well on the training data
    - It does not generalize well

Overfitting the Training Data

- A high-degree polynomial life satisfaction model
  - Strongly overfits the training data



source: Géron, A. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow"

Complex models

- Complex models such as deep neural networks can detect subtle patterns in the data
    - If the training set is noisy or if it is too small
        - The model is likely to detect patterns in the noise itself
        - These patterns will not generalize to new instances

Complex models

- Including uninformative attributes —> the country's name
    - A complex model may detect patterns
        - All countries in the training data with a "**w**" in their name
            - A life satisfaction greater than 7
            - New Zealand (7.3), Norway (7.6), Sweden (7.3), and Switzerland (7.5)
            - The **w**-satisfaction rule generalizes to Rwanda or Zimbabwe?
            - This pattern occurred in the training data by pure chance
                - The model cannot tell whether a pattern is real or simply the result of noise in the data

Overfitting

- Happens when the model is too complex
  - Relative to the amount and noisiness of the training data
- Simplify the model by selecting one with fewer parameters
  - A linear model rather than a high-degree polynomial model
  - Reducing the number of attributes in the training data
  - Constraining the model
- Gather more training data
- Reduce the noise in the training data
- Fix data errors and remove outliers

Regularization

- Constraining a model to make it simpler and reduce the risk of overfitting
- Our linear model has two parameters, θ_0 and θ_1
    - 2 degrees of freedom to adapt the model to the training data
        - Height (θ_0) and the slope (θ_1) of the line

Regularization

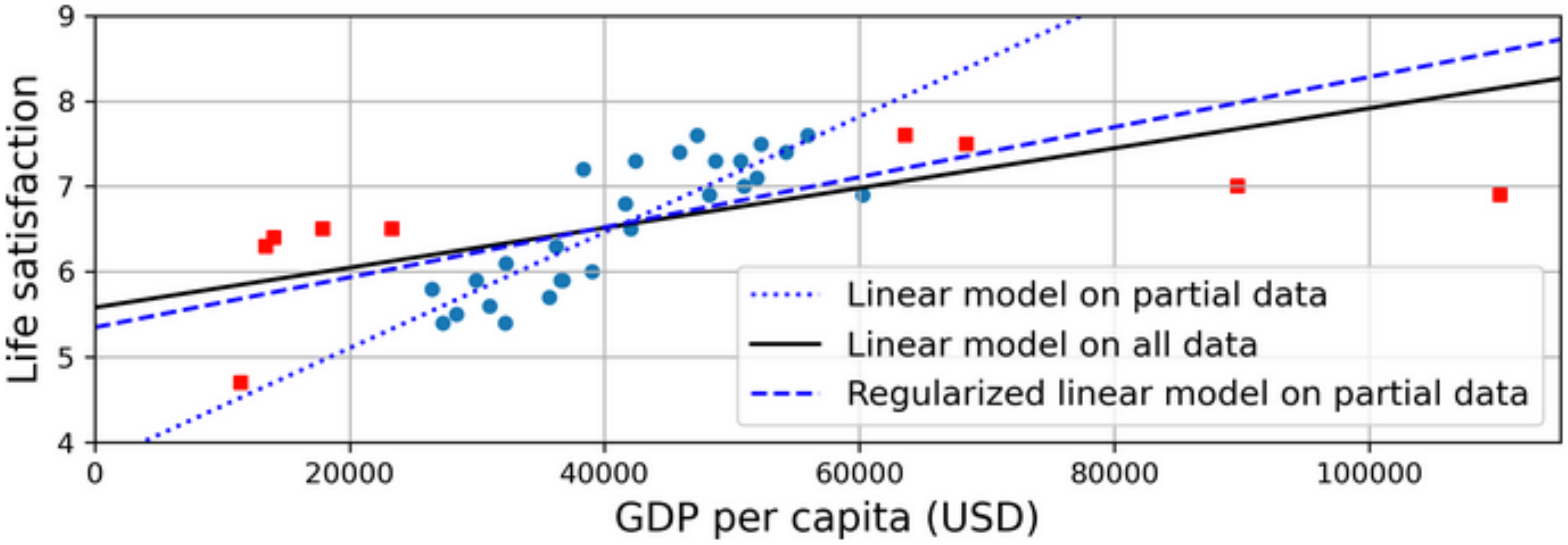
- Forcing $\theta_1 = 0$
  - Only one degree of freedom
  - Harder to fit the data properly

Find the right balance between fitting the training data perfectly and keeping the model simple enough to ensure that it will generalize well

- It will produce a model that's simpler than one with two degrees of freedom
- More complex than one with just one

Regularization

- Regularization forced the model to have a smaller slope
  - It does not fit the training data (circles) as well as the first model
  - It generalizes better to new examples (squares)



source: Géron, A. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow"

Regularization reduces the risk of overfitting

- The amount of regularization to apply during learning
  - Controlled by a hyperparameter
- A hyperparameter is a parameter of a learning algorithm
  - Not of the model
  - It is not affected by the learning algorithm itself
  - Set prior to training and remains constant during training

Underfitting the Training Data

- Underfitting is the opposite of overfitting
- It occurs when your model is too simple to learn the underlying structure of the data
- A linear model of life satisfaction is prone to underfit
  - Reality is just more complex than the model
  - Inaccurate predictions even on the training examples

Underfitting the Training Data

- Solutions
  - Select a more powerful model with more parameters
  - Feature engineering
    - Feed better features to the learning algorithm
- Reduce the constraints on the model
  - Reduce the regularization hyperparameter

Summary

- 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
    - Instance-based or model-based

Summary

- ML project
  - Gather data in a training set
    - Feed the training set to a learning algorithm
    - Model-based
      - Tunes some parameters to fit the model to the training set
      - Be able to make good predictions on new cases as well
    - Instance-based
      - It just learns the examples by heart and generalizes to new instances
        - Using a similarity measure to compare them to the learned instances

Summary

- The system will not perform well if
  - Training set is too small
  - The data is not representative
  - Noisy
  - Polluted with irrelevant features
- Model needs to be
  - Not too simple —> underfit
  - Not too complex —> overfit

Testing and Validating

- How to know how well a model will generalize to new cases ?
  - Put the model in production and monitor how well it performs
    - Too risky if the model is not good
  - A better option is to split your data into two sets
    - The training set and the test set
- Generalization error
  - The error rate on new cases
  - Evaluating the model on the test set
    - Give an estimate of this error

Testing and Validating

- Low training error
  - The model makes few mistakes on the training set
  - High generalization error
    - The model is overfitting the training data.

Training and testing sets

- Percentage depends on the size of the dataset
  - Commonly used
    - 80% of the data for training
    - 20% for testing
  - Dataset with 10 million instances
    - 1% for testing means your test set will contain 100,000 instances
      - Probably more than enough to get a good estimate of the generalization error

Hyperparameter Tuning and Model Selection

- Hesitation between types of models
  - Train both and compare how well they generalize using the test set
- Applying some regularization to avoid overfitting
  - How do you choose the value of the regularization hyperparameter?
    - Train 100 different models using 100 different values for this hyperparameter
  - The best hyperparameter value is found!
    - Produces a model with the lowest generalization error —> 5%
    - In production it produces 15% errors

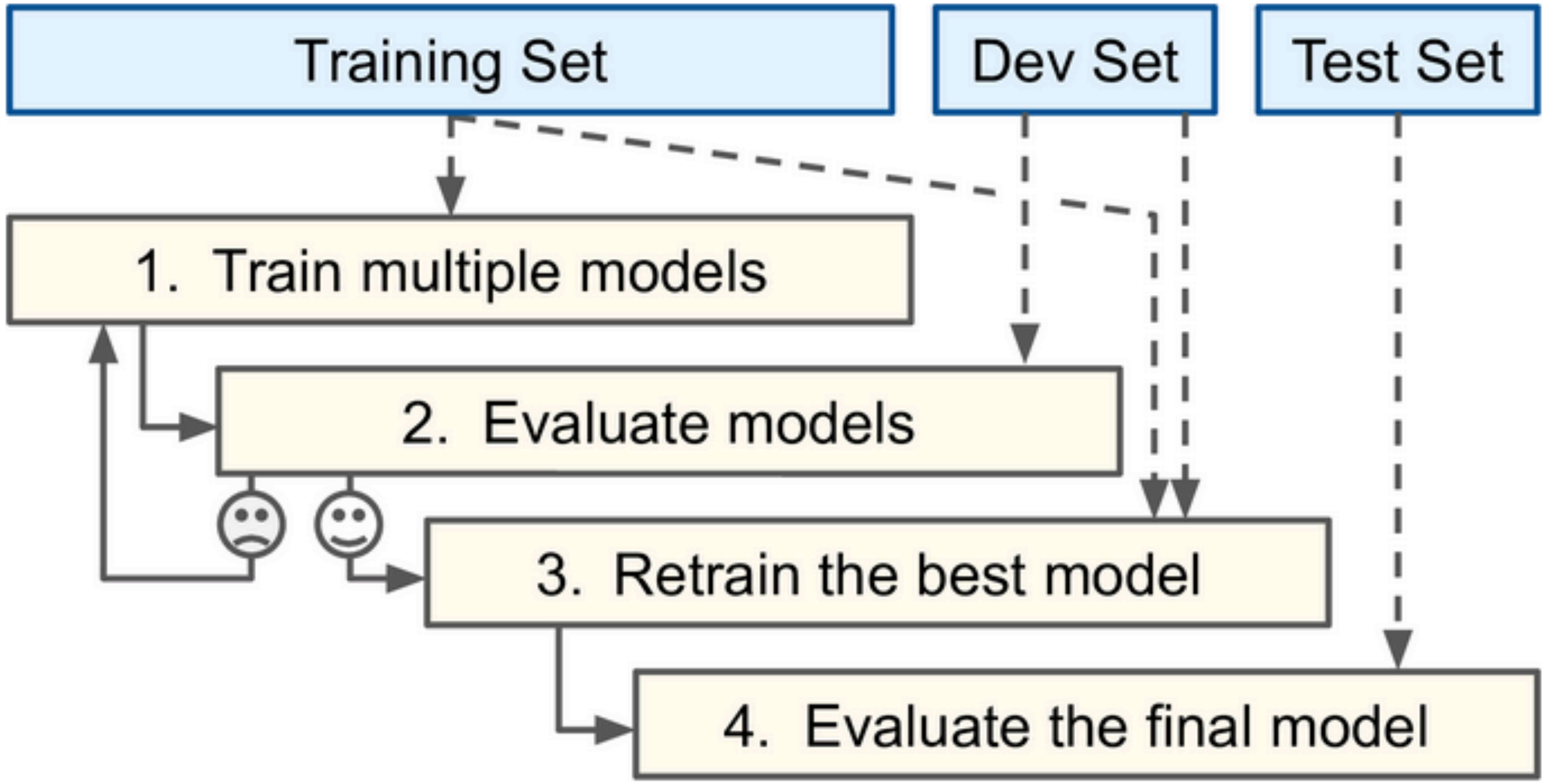
Hyperparameter Tuning and Model Selection

- Measuring the generalization error multiple times on the test set
  - Adapt the model and hyperparameters to produce the best model for that particular set
  - It means that the model is unlikely to perform as well on new data

Holdout validation

- Separate part of the **training** set to evaluate several candidate models and select the best one
    - **Validation** set or the **Development** set
- Train multiple models with various hyperparameters
    - On the reduced **training** set
        - The full **training** set minus the **validation** set
- Select the model that performs best on the **validation** set
- Train the best model on the full **training** set
- Evaluate this final model on the **test** set to get an estimate of the generalization error

Validation



source: Géron, A. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow"

Validation

- When the **validation** set is too small
  - The model evaluations will be imprecise
    - Might end up selecting a suboptimal model by mistake
- When the **validation** set is too large
  - The remaining **training** set will be much smaller than the full **training** set
  - Not ideal to compare candidate models
    - Trained on a much smaller **training** set
      - The final model will be trained on the full **training** set

Validation

- One solution to this problem
  - Perform repeated cross-validation using many small **validation** sets
  - Each model is evaluated once per validation set after it is trained on the rest of the data
  - Averaging out all the evaluations of a model
    - A much more accurate measure of its performance
  - There is a drawback
    - The training time is multiplied by the number of **validation** sets

Data mismatch

- When it is easy to get a large amount of data for training
  - Not perfectly representative of the data that will be used in production
  - Creating a mobile app to take pictures of flower
    - Automatically determine their species
    - Download millions of pictures of flowers on the web
      - Might not be perfectly representative of the pictures
        - Will actually be taken using the app on a mobile device

Data mismatch

- Both the **validation** set and the **test** set must be
  - As representative as possible of the data you expect to use in production
  - Should be composed exclusively of representative pictures
  - Shuffle them and put half in the **validation** set and half in the **test** set
    - Making sure that no duplicates or near-duplicates end up in both sets

Data mismatch

- After training your model on the web pictures
  - Bad performance of the model on the **validation** set
    - We cannot know the reason
      - The model has overfit the **training** set
      - The mismatch between the web pictures and the mobile app pictures

No Free Lunch Theorem

- A model is a simplified representation of the data
- The simplifications are meant to discard the superfluous details
  - Unlikely to generalize to new instances
- When you select a particular type of model
  - Implicitly making assumptions about the data
    - A linear model
      - Assuming the data is fundamentally linear
      - The distance between the instances and the straight line is just noise

No Free Lunch Theorem

- If you make absolutely no assumption about the data
  - There is no reason to prefer one model over any other
  - For some datasets the best model is a linear model
    - For other datasets it is a neural network
  - There is no model that is a priori guaranteed to work better
  - The only way to know for sure which model is best is to evaluate them all
    - This is not possible
    - In practice you make some reasonable assumptions about the data and evaluate only a few reasonable models

le cnam

Artificial Intelligence and Machine Learning for Networks and IoT

Class 1 — Introduction

Pedro Braconnot Velloso