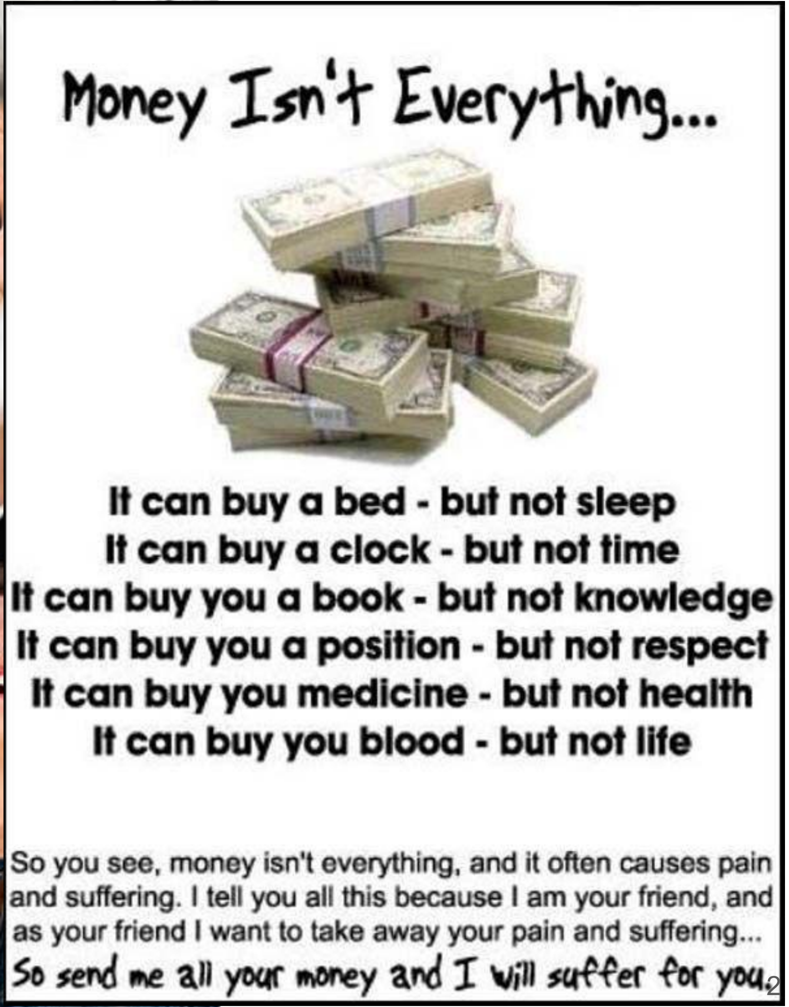
Machine Learning Fundamentals

N. Rich Nguyen

CS 4774

Fall 2019


Money Isn't Everything...
It can buy a bed - but not sleep
It can buy a clock - but not time
It can buy you a book - but not knowledge
It can buy you a position - but not respect
It can buy you medicine - but not health
It can buy you blood - but not life
So you see, money isn't everything, and it often causes pain and suffering. I tell you all this because I am your friend, and as your friend I want to take away your pain and suffering...
So send me all your money and I will suffer for you.

Would more money make people happier?
A Machine Learning Approach

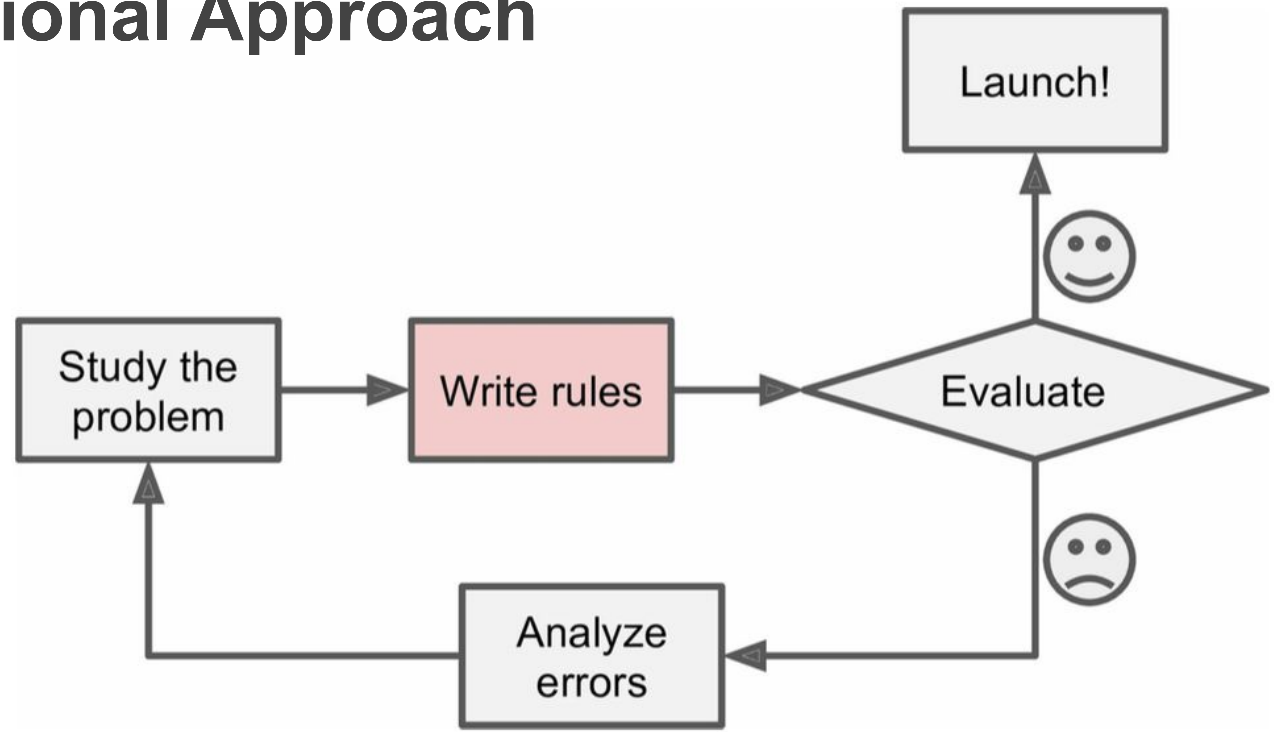
But first, let's learn some of the fundamentals!

Today's Learning Outcomes

- ❏ Understand problems for which ML is great
- ❏ Know some basic ML vocabulary
- ❏ Identify supervised tasks versus unsupervised tasks
- ❏ Take a sneak peak at linear regression
- ❏ Be aware of some challenges of ML and ways to fix them

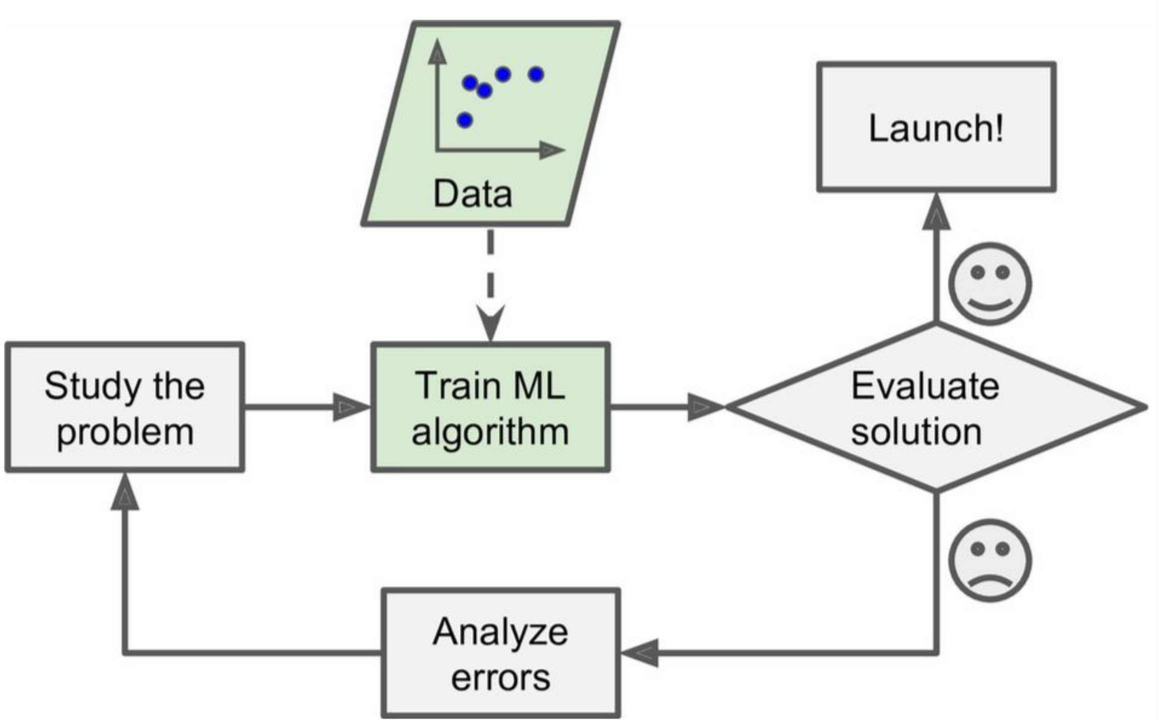
Suppose you are given the task to build a system that can distinguish spam emails. **How would you engineer such a system?**

Traditional Approach

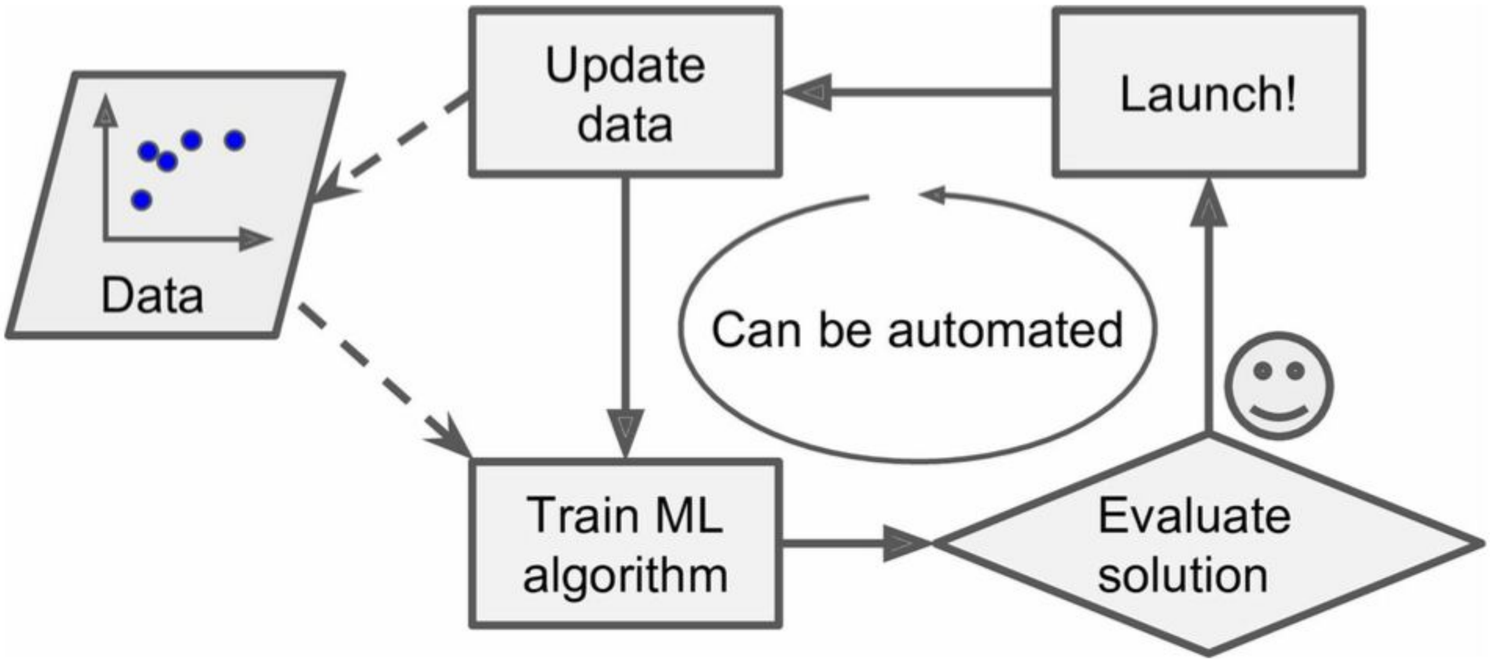


Problem: Hard to maintain more and more rules

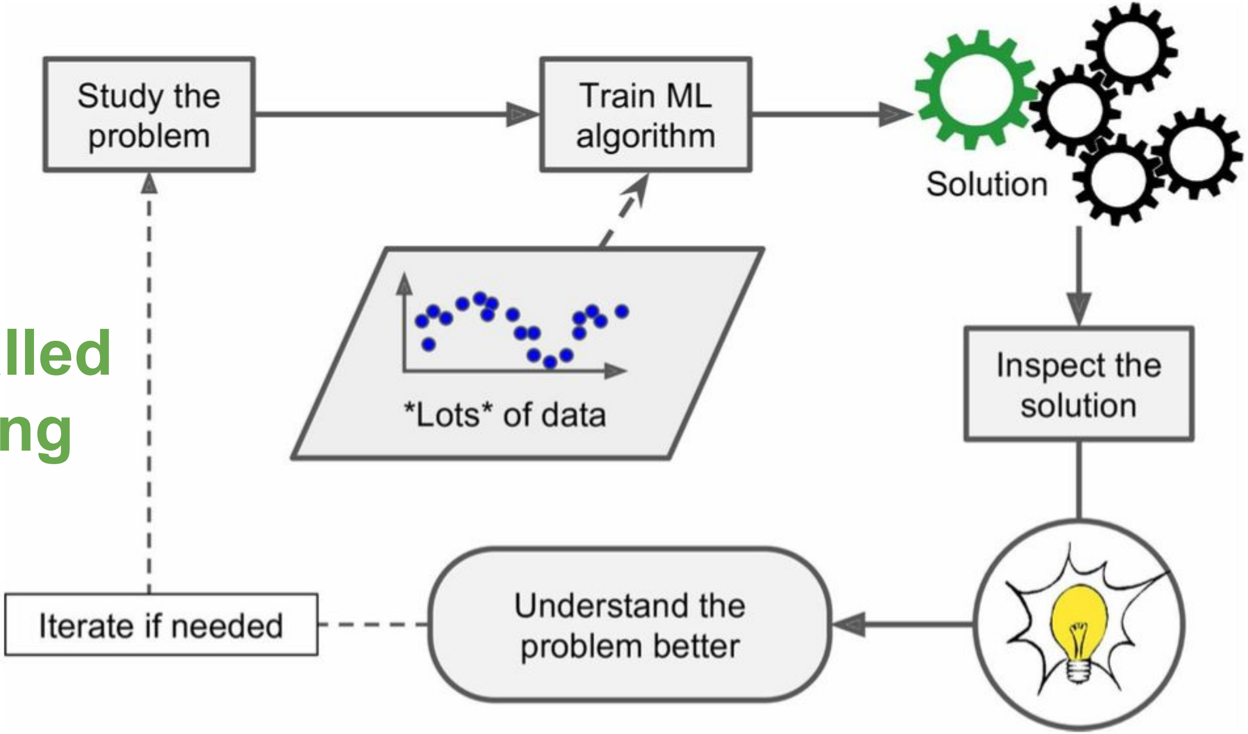
Machine Learning Approach



Adapting the change (ie. new data)



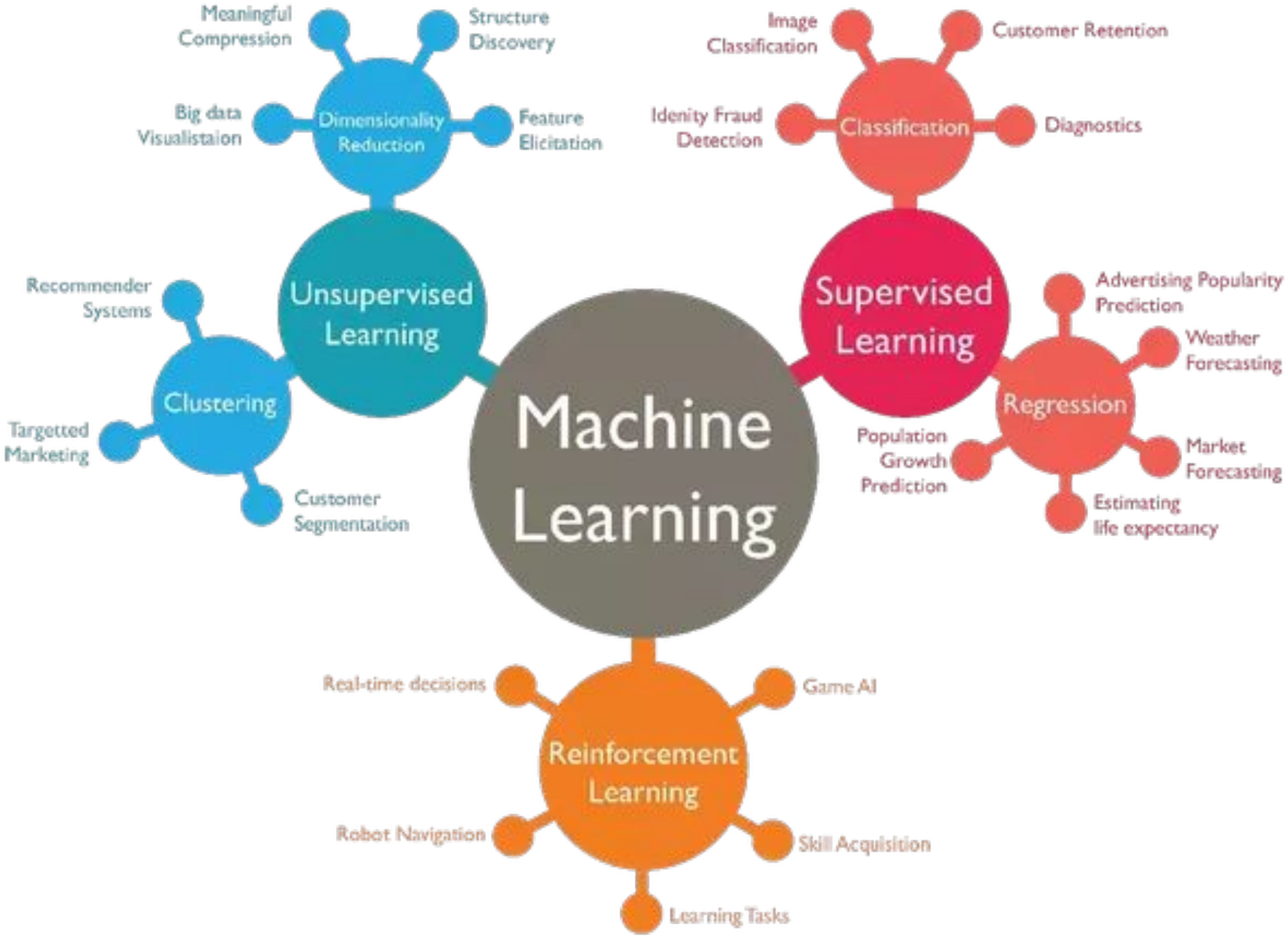
ML can help us gain insight about the problem



This is called Data Mining

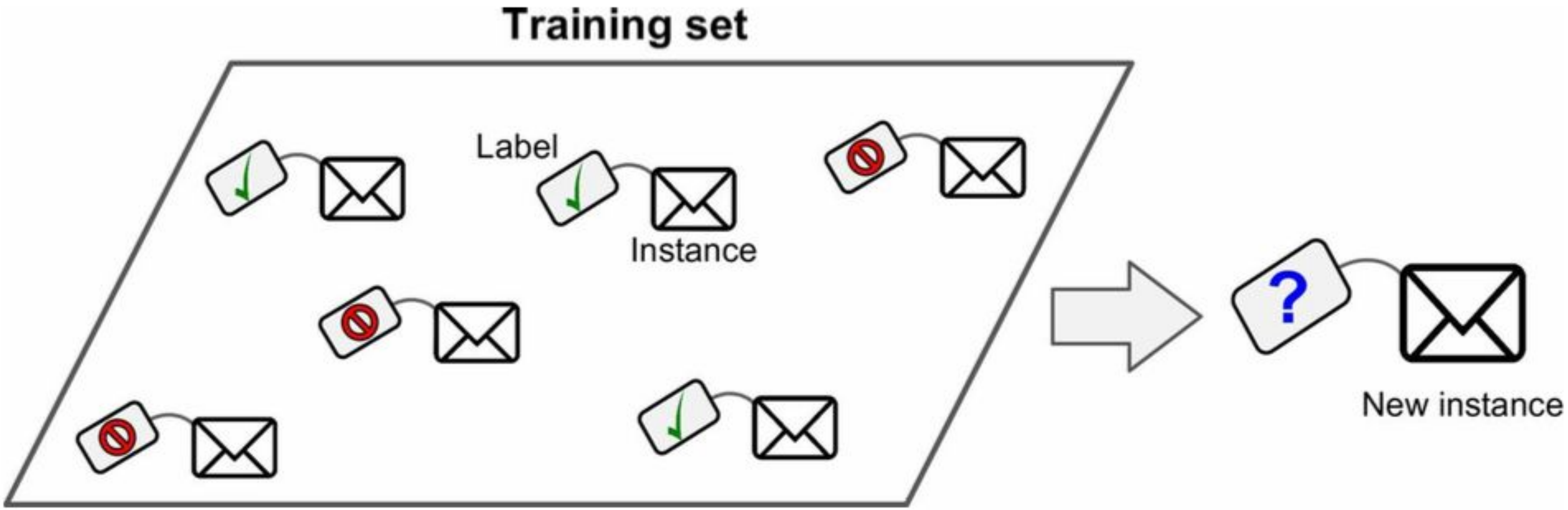
Type of Problems ML is great for

1. Problems which requires lots of **hand-tuning** and long list of rules
2. **Complex** problems for which there is no good traditional solution
3. **Changing** environment and need to **adapt to new data**
4. Getting insights from **large amount of data**


Machine Learning
Unsupervised Learning
Dimensionality Reduction
Meaningful Compression
Structure Discovery
Big data Visualistaion
Feature Elicitation
Clustering
Recommender Systems
Targetted Marketing
Customer Segmentation
Supervised Learning
Classification
Image Classification
Customer Retention
Idenity Fraud Detection
Diagnostics
Regression
Advertising Popularity Prediction
Weather Forecasting
Population Growth Prediction
Market Forecasting
Estimating life expectancy
Reinforcement Learning
Real-time decisions
Game AI
Robot Navigation
Skill Acquisition
Learning Tasks

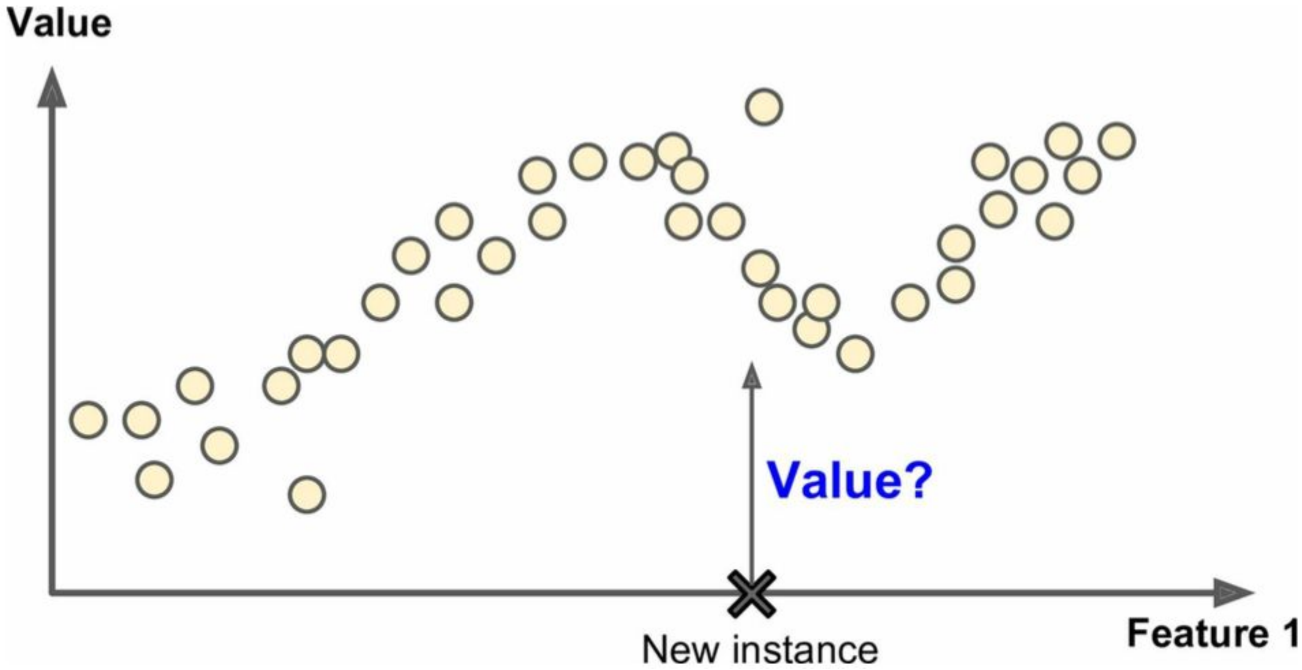
Supervised vs. Unsupervised

Supervised Learning - Classification



Predict a label: Spam or Ham

Supervised Learning - Regression



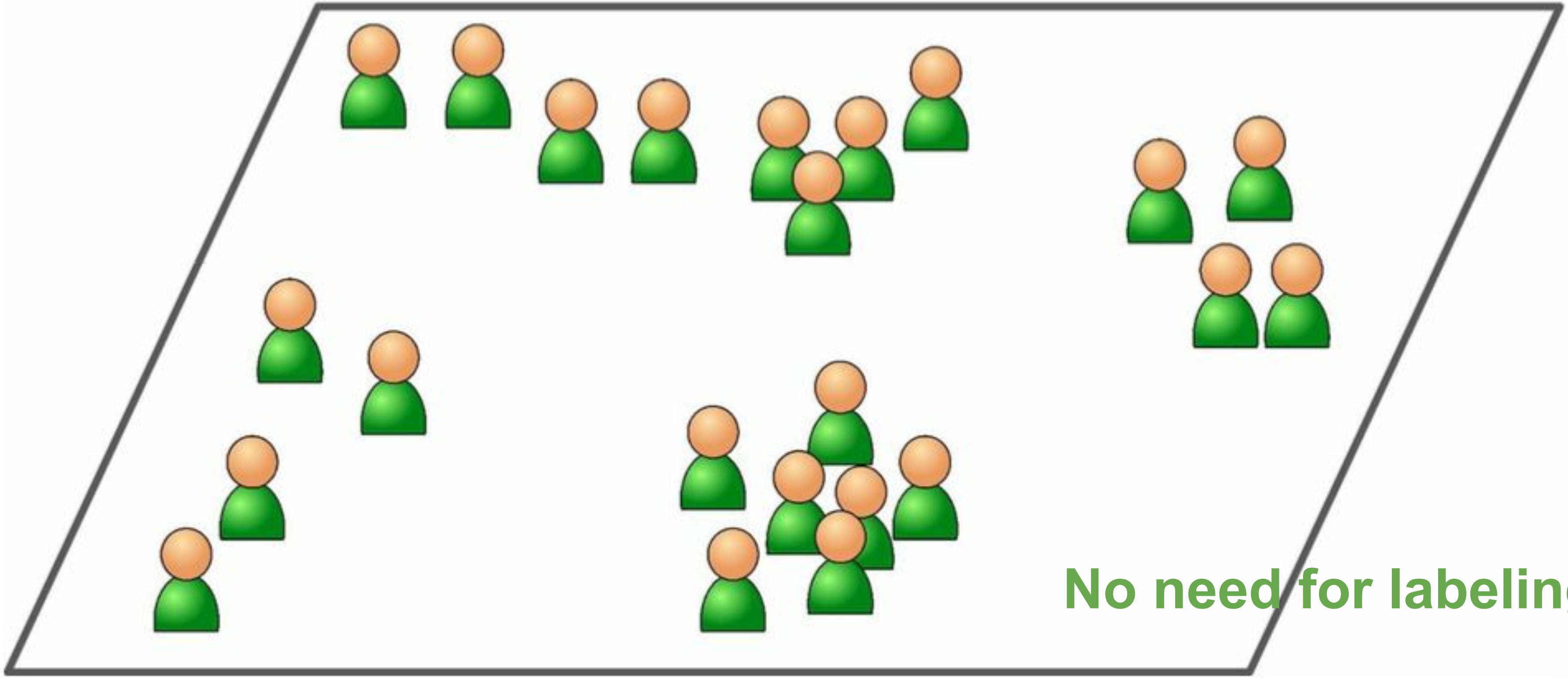
Predict a numeric value: House resale $$$

List of supervised learning algorithms

- k-Nearest Neighbors
- Linear Regression
- Logistic Regression
- Support Vector Machine
- Decision Trees
- Random Forests
- Back Propagation in Neural Networks

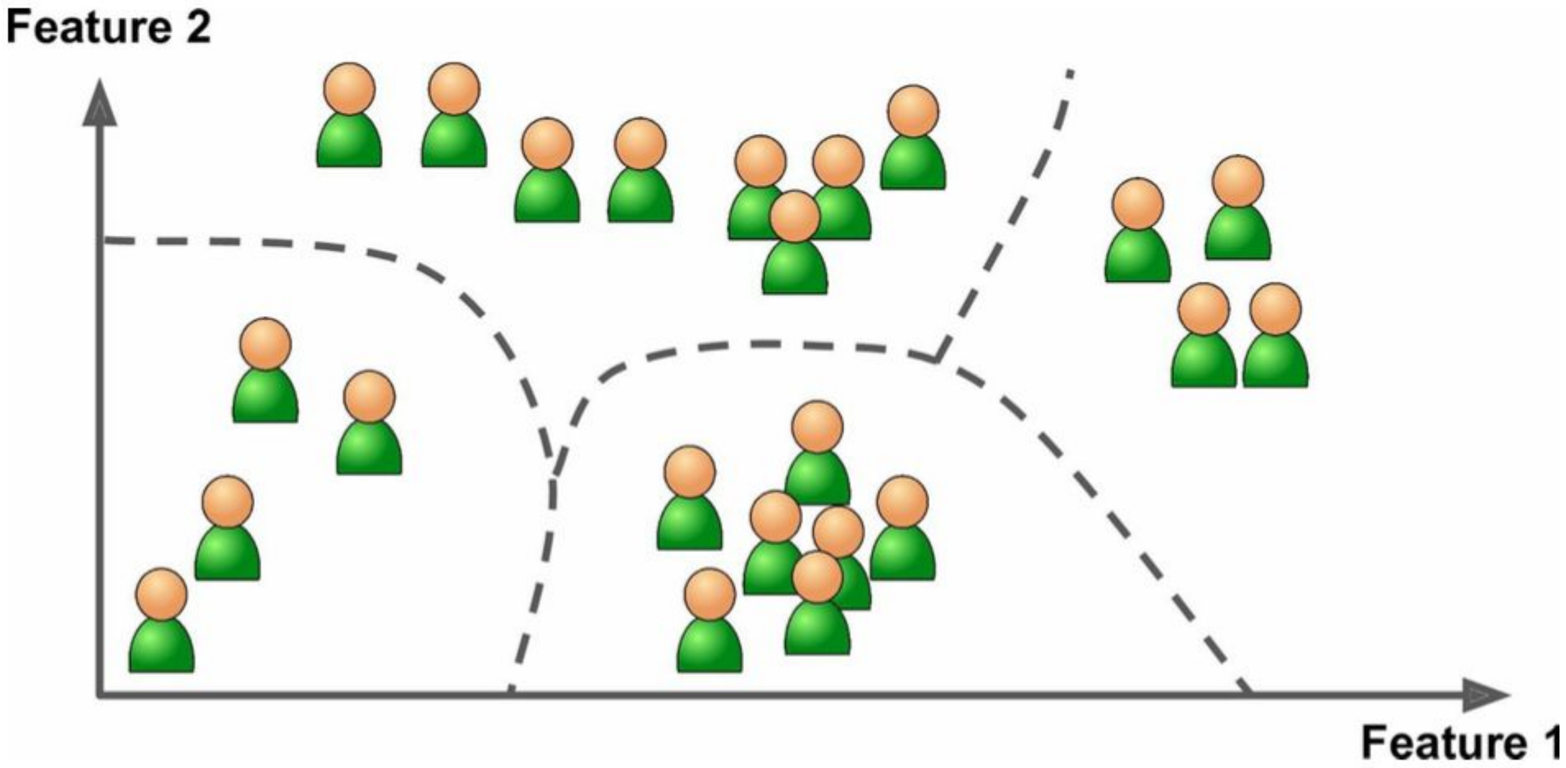
We will cover all of them in this course!

Unsupervised (learning without a teacher)

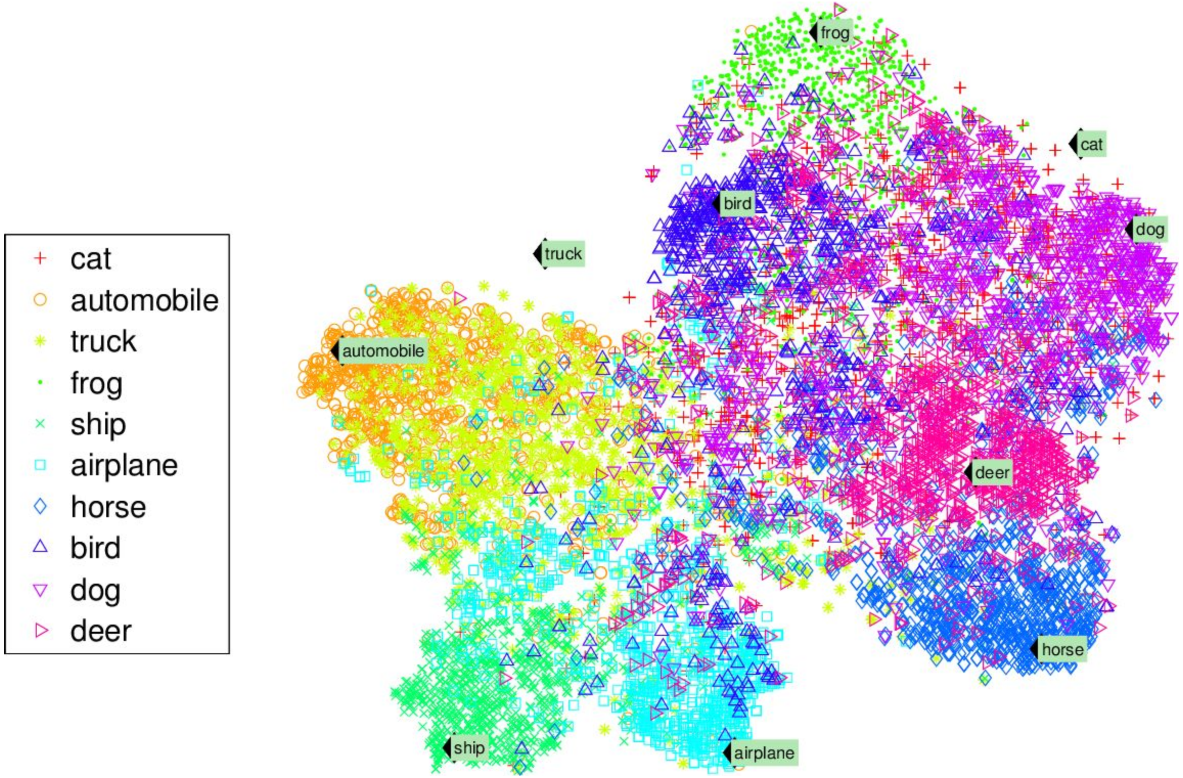


No need for labeling

Clustering (grouping blog visitors)



Visualization (preserve the structure of data)



Example of a t-SNE visualization highlighting semantic clusters

Dimensionality Reduction

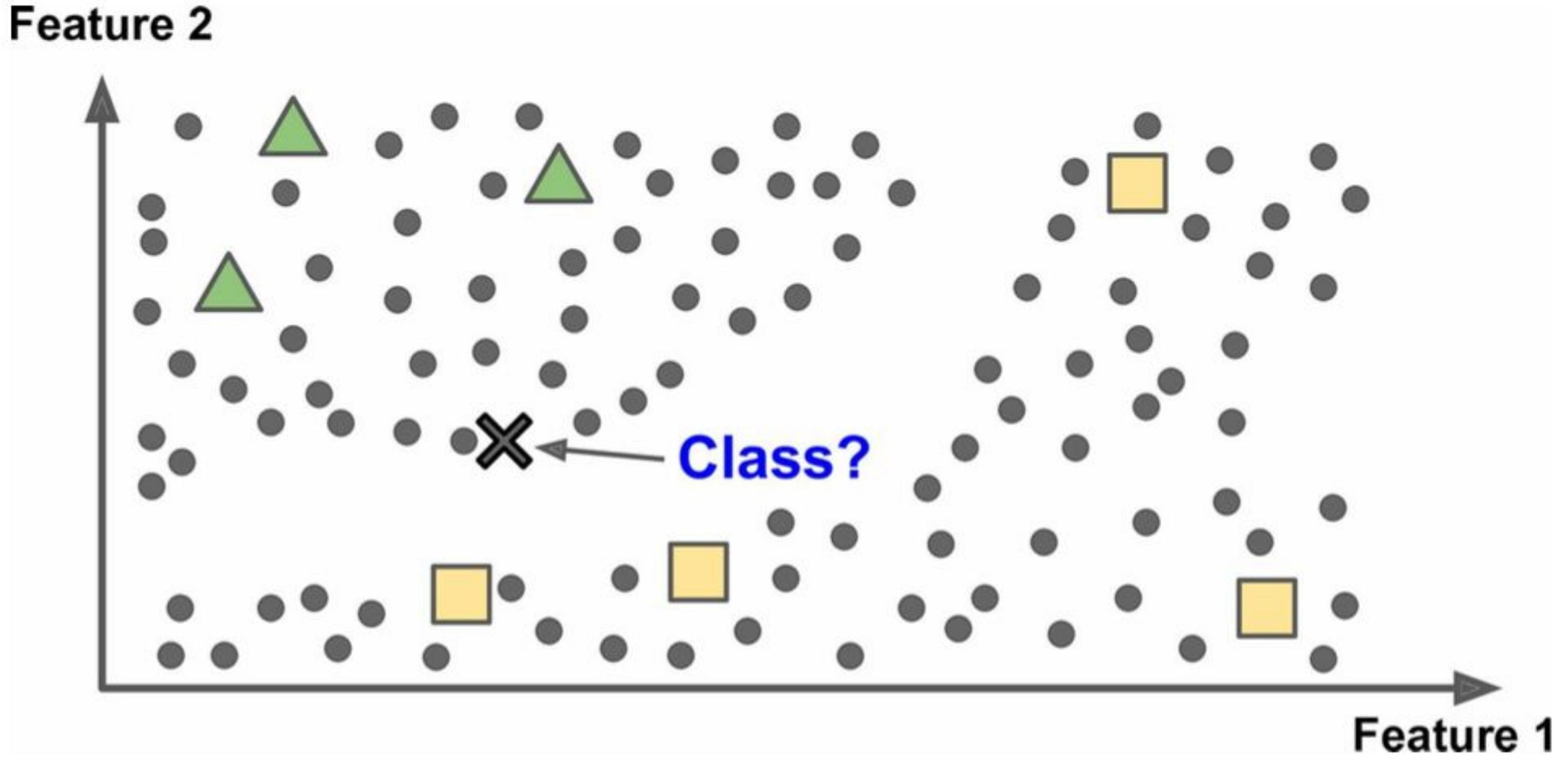
Simplify the data without losing too much information

Merge correlated features into one:

For example, car’s mileage may be very correlated with its age

Anomaly Detection

Semi-supervised Learning (partially labeled data)

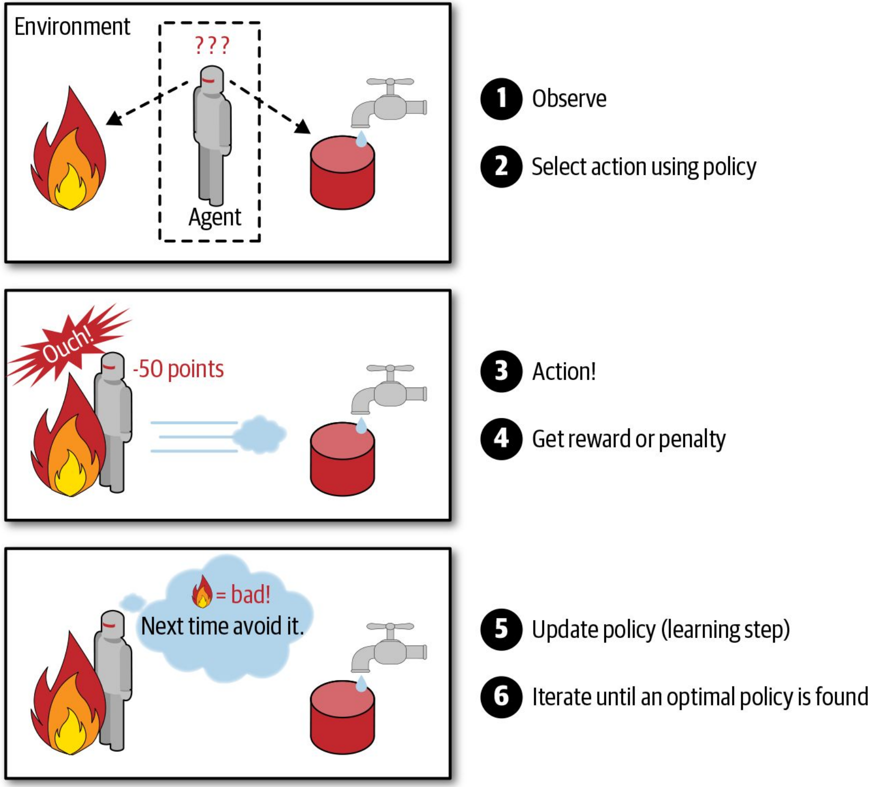


List of unsupervised learning algorithms

- K-means
- Hierarchical Clustering (HAC)
- Principal Component Analysis (PCA)
- Locally-Linear Embedding (LLE)

We will cover all of these in this course!

Reinforcement Learning (ie. DeepMind's AlphaGo)



If time permits, we might cover this too!

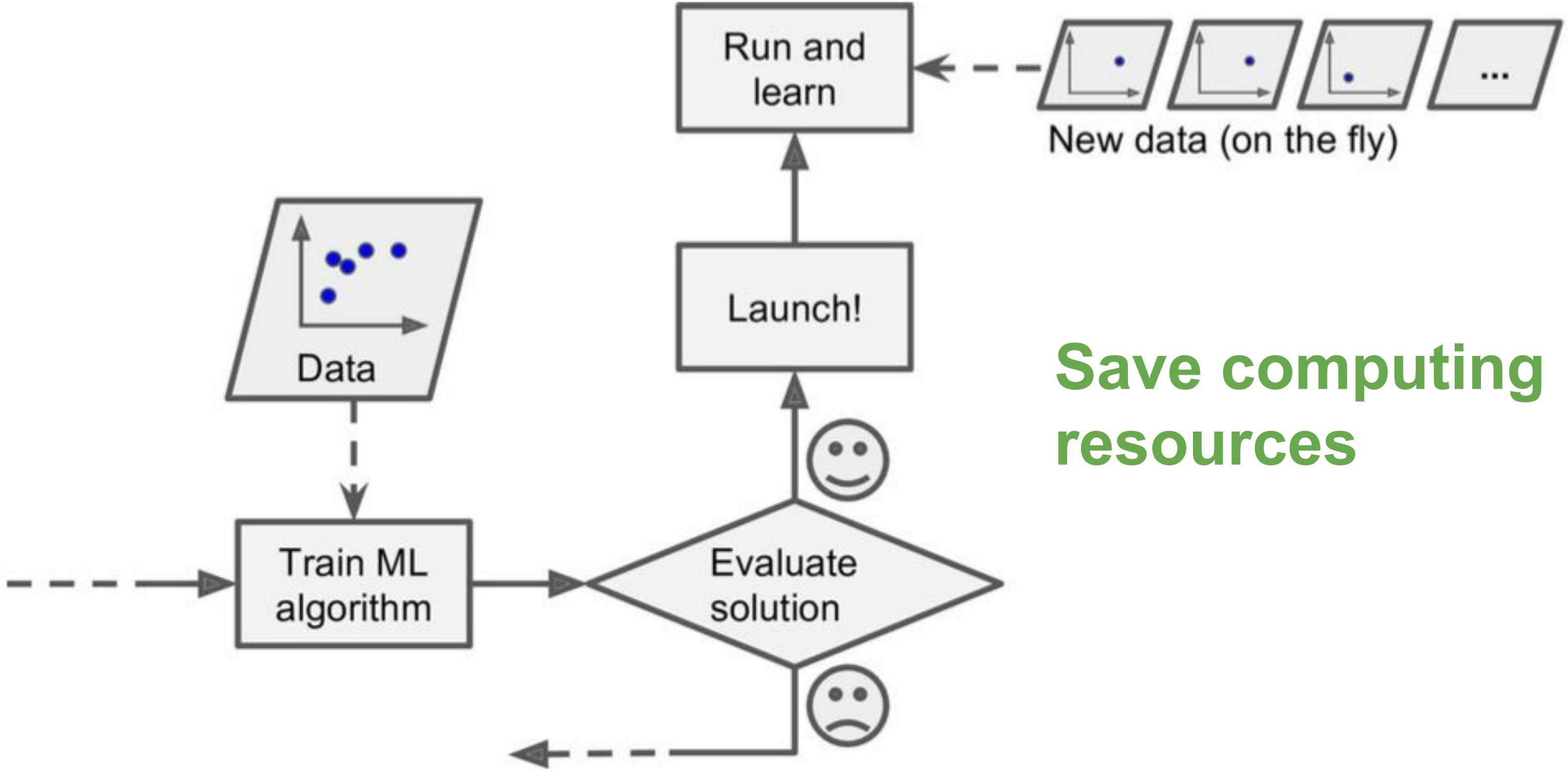
Batch Learning vs. Online Learning

Batch Learning (Offline Learning)

Must be trained using **all available data**

- Generally take time, so typically done offline
- Requires a **lot** of computing resources (CPU, memory, network I/O, ect.)
- Must be **retrained from scratch** for updated data

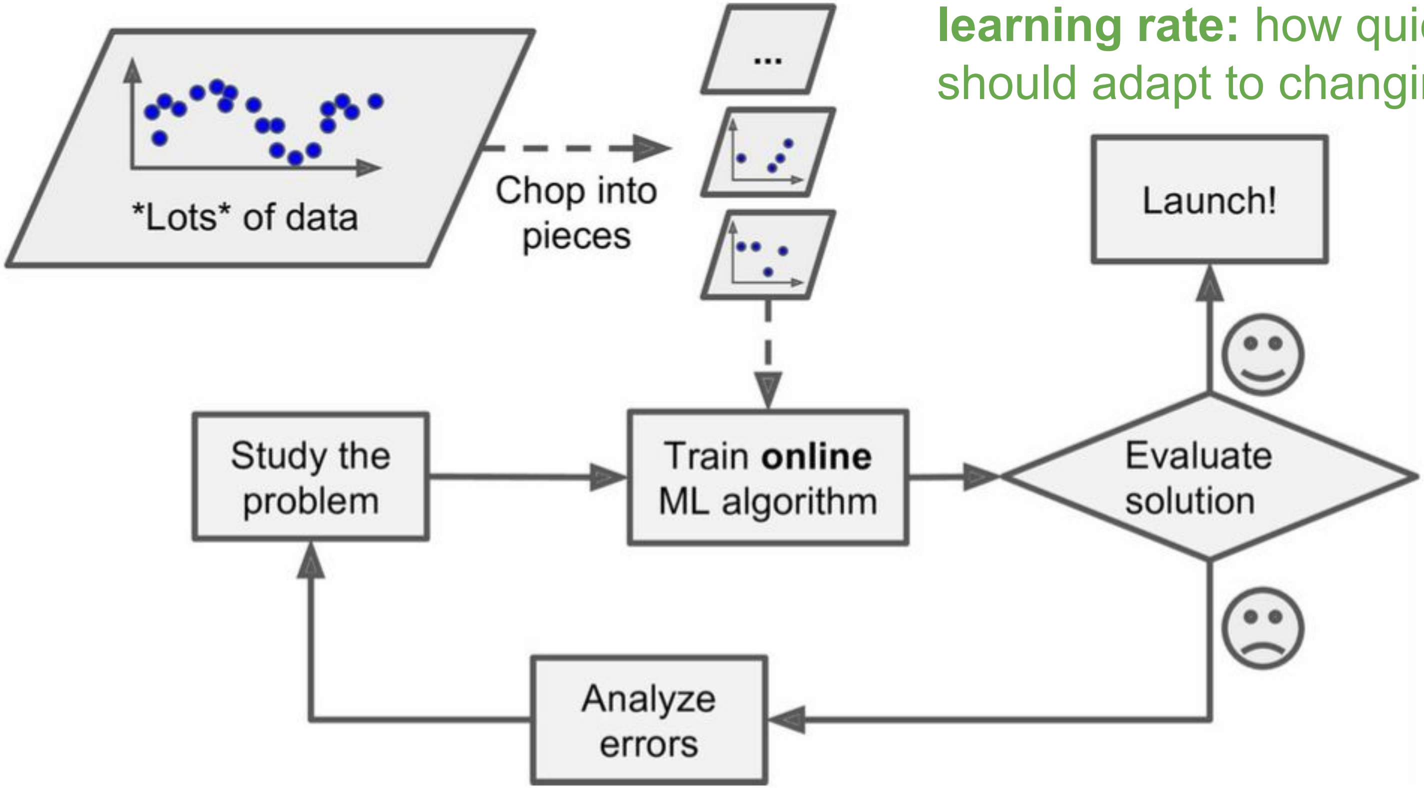
Online Learning (Incremental Learning)



Save computing resources

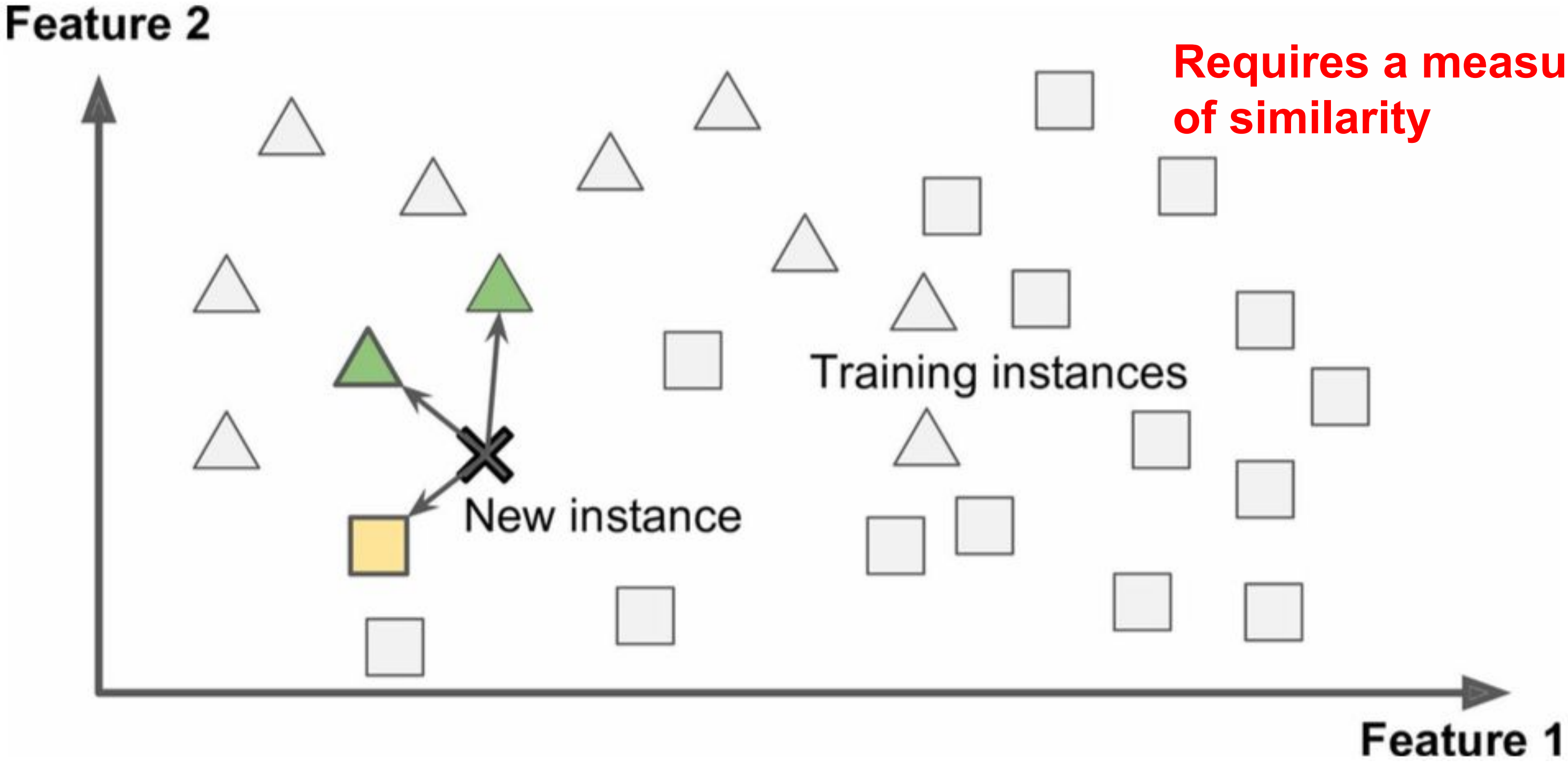
Handling Huge dataset (cannot fit all in memory)

learning rate: how quick a ML algorithm should adapt to changing data



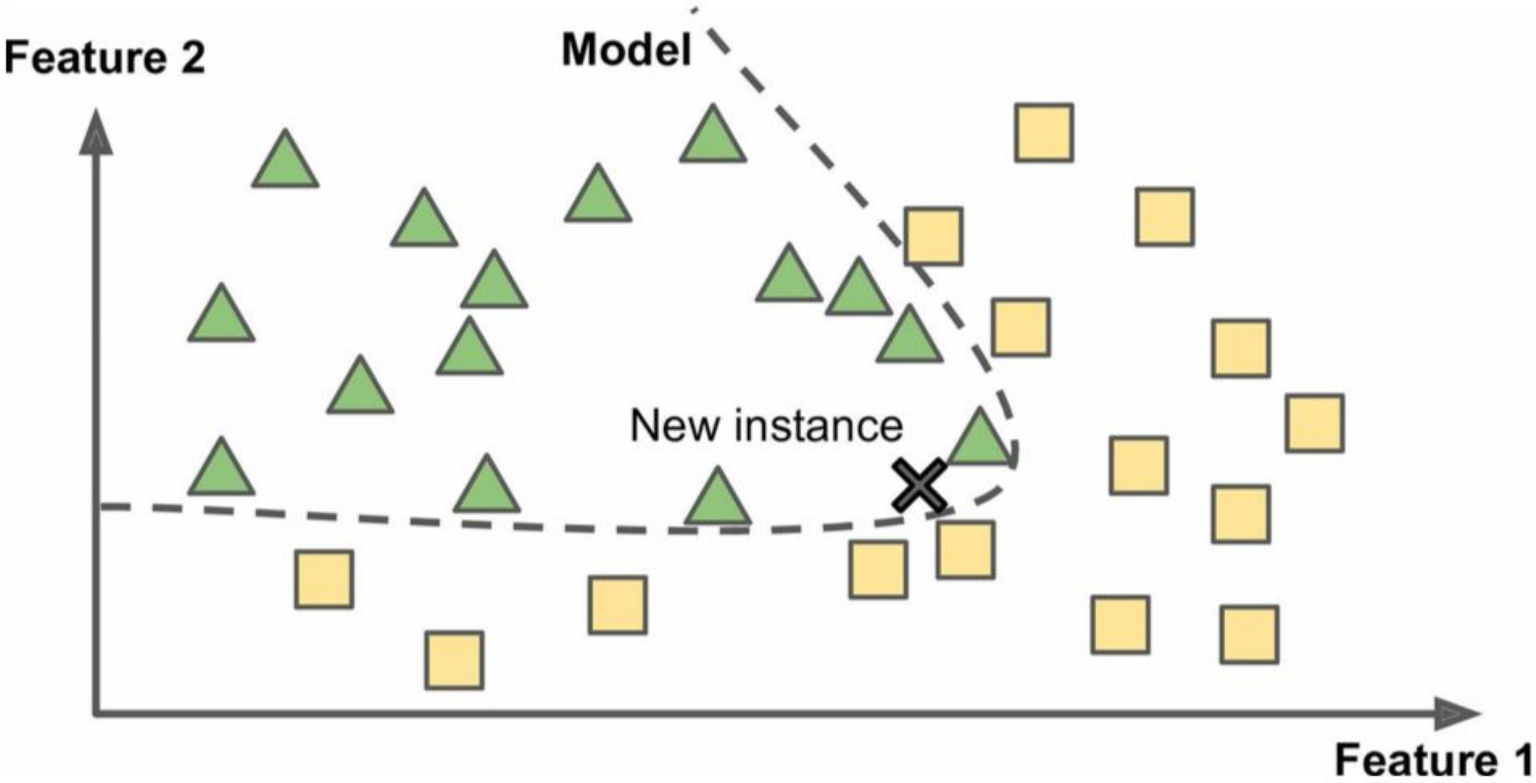
Instance-Based vs. Model-Based

Instance-based (ie. K-nearest neighbors algo)



Requires a measure of similarity

Model-based



Activity: Supervised vs. Unsupervised?

1. Brainstorm some problems that are of interest to you
2. Identify each of them whether it is a **supervised** or **unsupervised**, **instance-based** or **model-based**, **batch** or **online**.
3. Discuss with a **neighbor**
4. Share with the class

Are people happier in wealthy countries?
A Machine Learning Approach

The Data

Download the Better Life Index (from **OECD's website**)

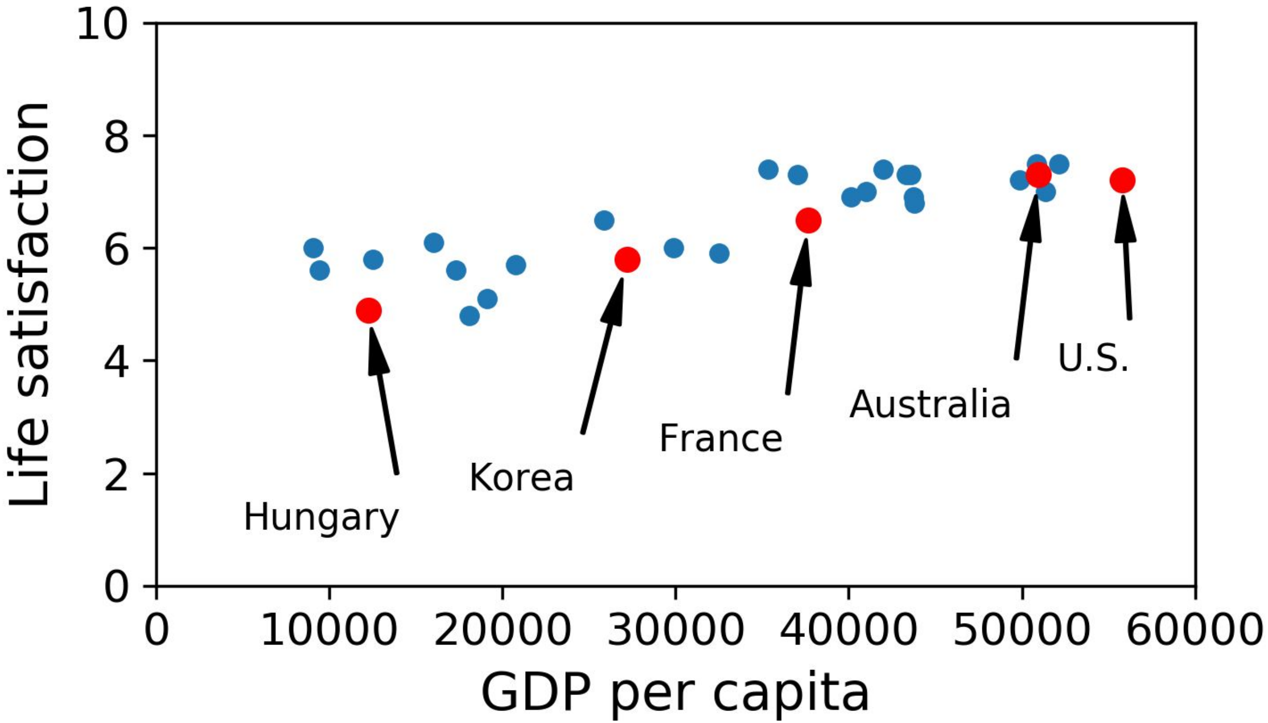
Download GDP per capita (from **IMF's website**)

Sort by GDP per capita

Country	GDP per capita (USD)	Life satisfaction
Hungary	12,240	4.9
Korea	27,195	5.8
France	37,675	6.5
Australia	50,962	7.3
United States	55,805	7.2

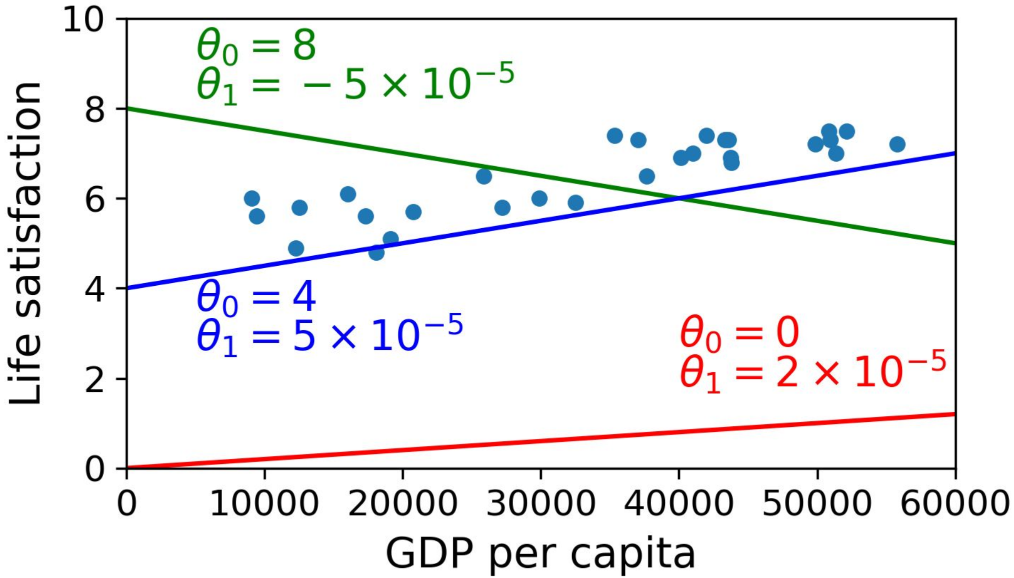
Let’s plot them out

Trendy?
Noisy? or so it seems...



Model Selection: A (simple) linear model

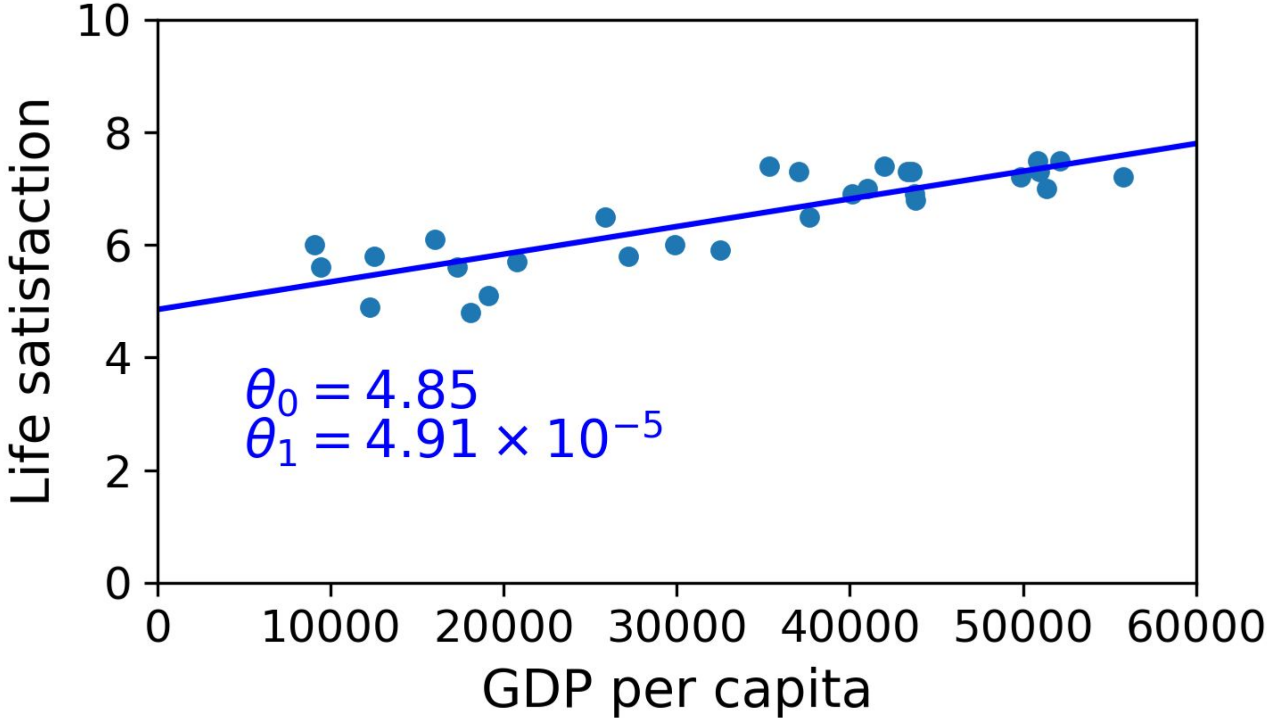
$$life\_satisfaction = \theta_0 + \theta_1 \times GDP\_per\_capita$$



Which one is better?
→ learn the model parameters which minimize some error

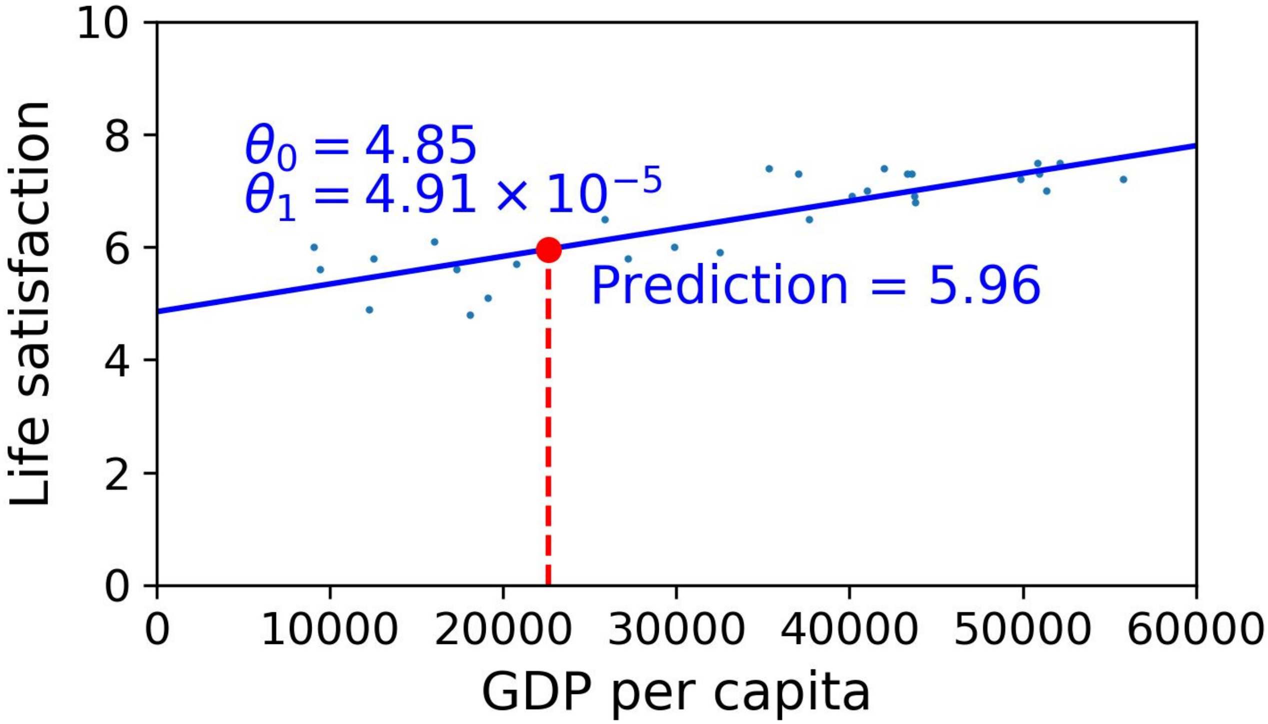
A model that fits the data best

Finding the parameters to fit best to the data is called **training** the model.



Using this model, can you predict a life satisfaction of Cyprus with GDP of $20,732?

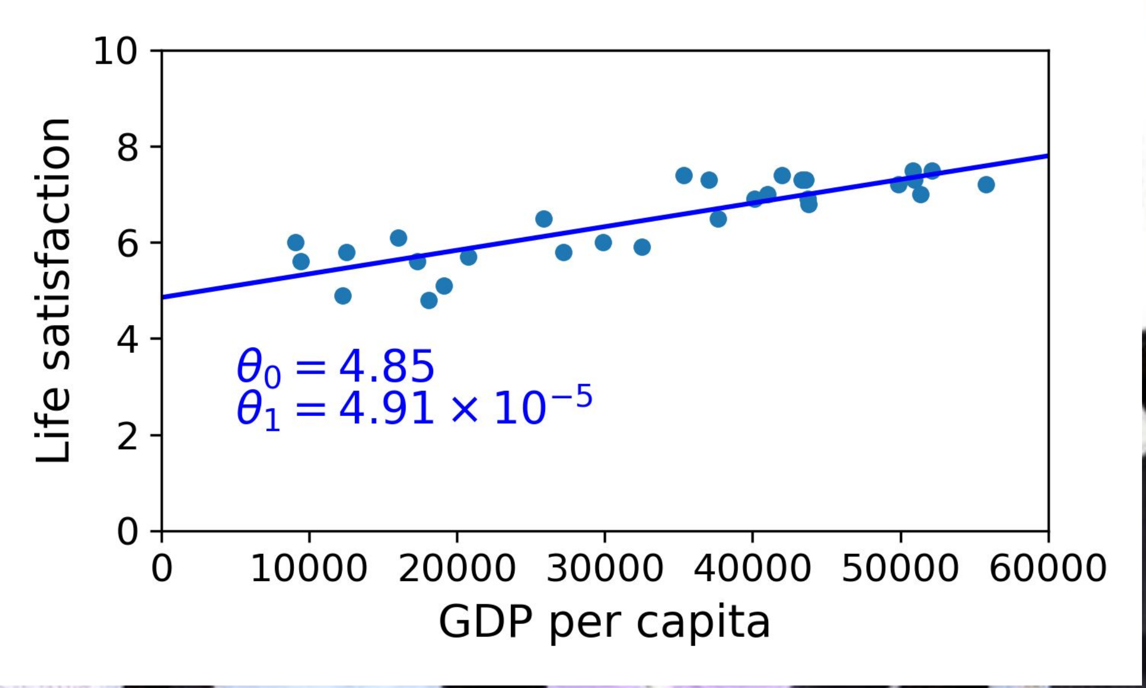
Inferencing (Predicting) the case of Cyprus



Sneak Peak at the code

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

# Load the data
oecd_bli = pd.read_csv("oecd_bli_2015.csv", thousands=',')
gdp_per_capita = pd.read_csv("gdp_per_capita.csv",thousands=',',delimiter='\t',
                             encoding='latin1', na_values="n/a")

# Prepare the data
country_stats = prepare_country_stats(oecd_bli, gdp_per_capita)
X = np.c_[country_stats["GDP per capita"]]
y = np.c_[country_stats["Life satisfaction"]]

# Visualize the data
country_stats.plot(kind='scatter', x="GDP per capita", y='Life satisfaction')
plt.show()

import sklearn.neighbors
model = sklearn.neighbors.KNeighborsRegressor(n_neighbors=3)
# Train the model
model.fit(X, y)

# Make a prediction for Cyprus
X_new = [[22587]]  # Cyprus' GDP per capita
print(model.predict(X_new)) # outputs [[ 5.96242338]]
```

Would more money make people happier?



Outcomes

If all go well, your model will make good predictions.

If not, you may need:

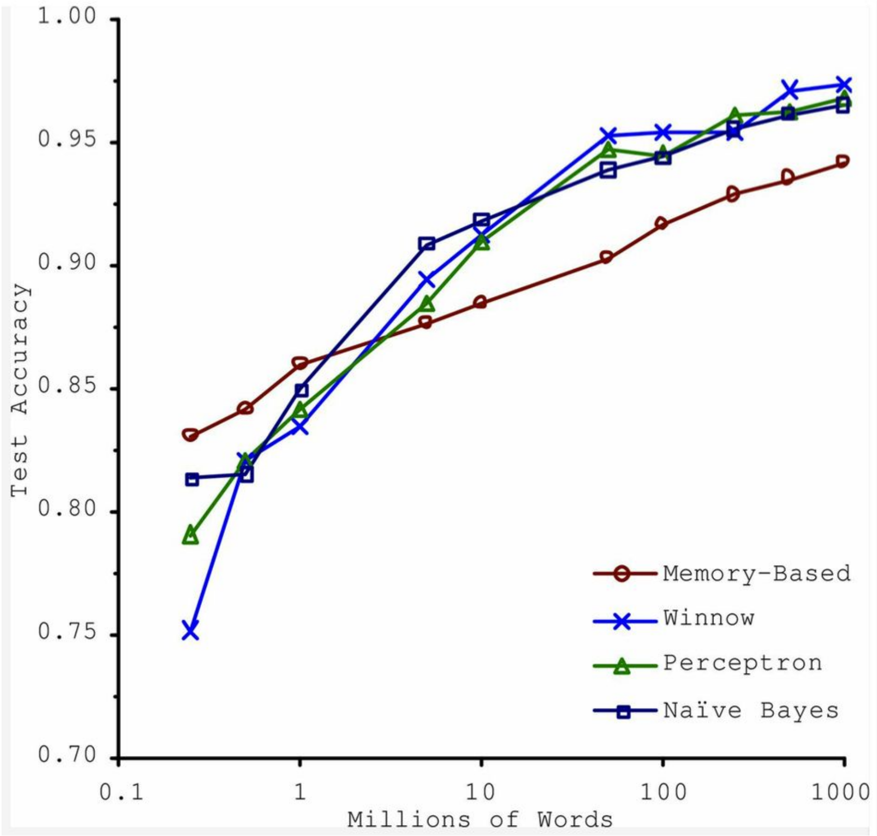
- More attributes or features (employment rate, health, pollution, ect)
- Better quality of training data
- More powerful model (non-linear model such as Polynomial Regression)

Some Challenges of Machine Learning

Either "bad data" or "bad algorithm"

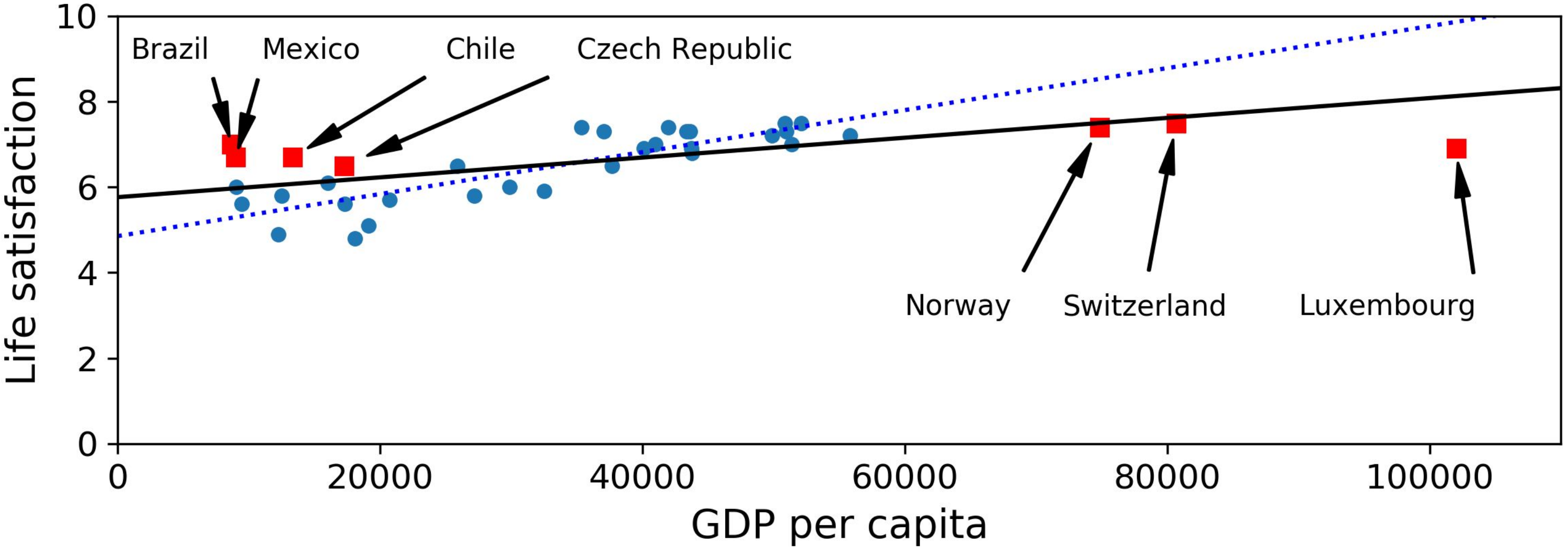
Bad Data

1. Insufficient Quantity of Training Data



The tradeoff between data vs. algorithms

2. Non-representative Training Data



Sampling noise: error associated with sampling a small dataset
Sampling bias: large portion of data is not representative due to sampling method

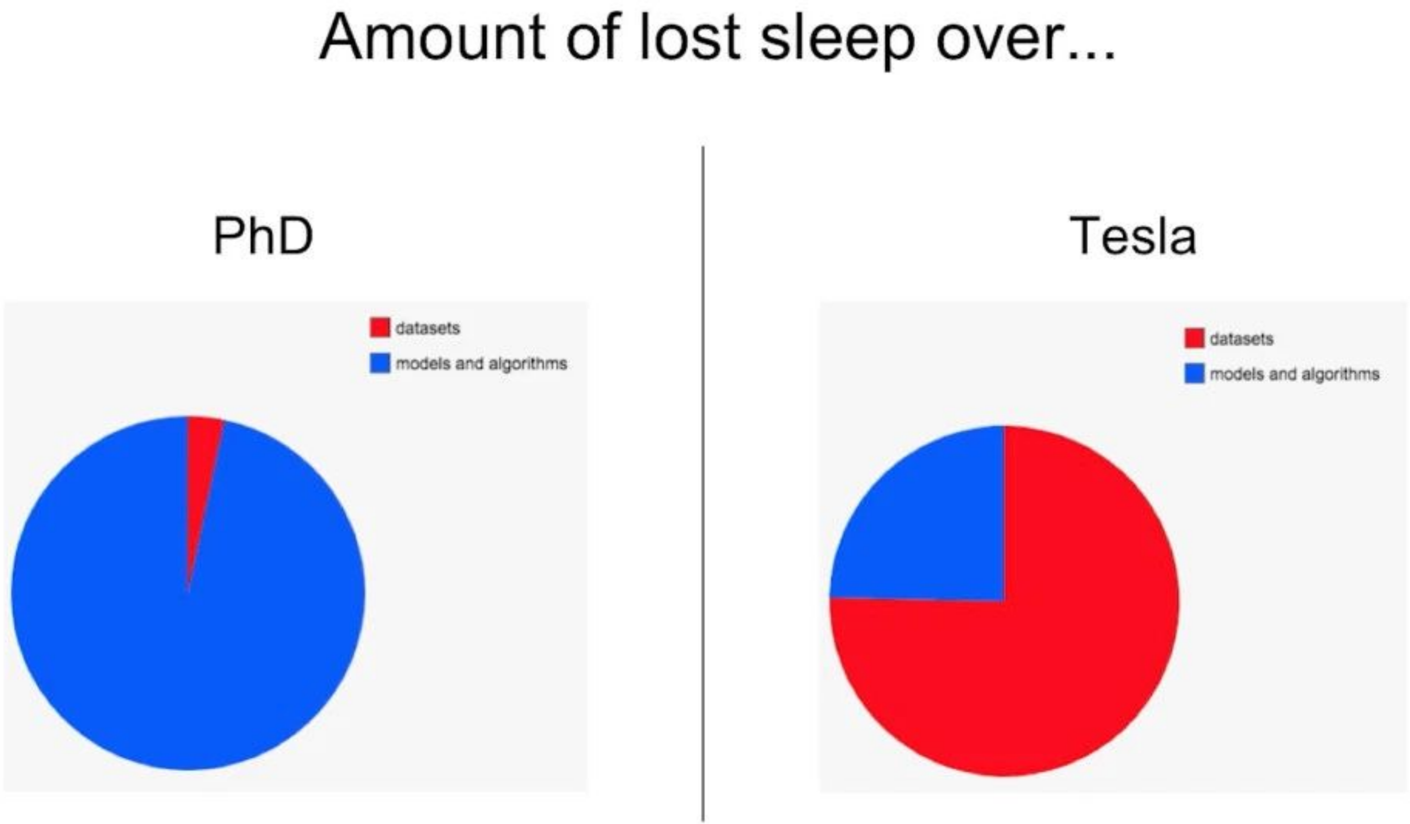
3. Poor Quality Data

- Full of errors (human and machine)
- Missing data (nonresponse)
- Outliers (Brazil vs. Luxembourg)
- Noise (by measurements)

MUCH of data scientist’s time is spent on…

cleaning up training data

Tesla’s AI Director on cleaning data



(link)

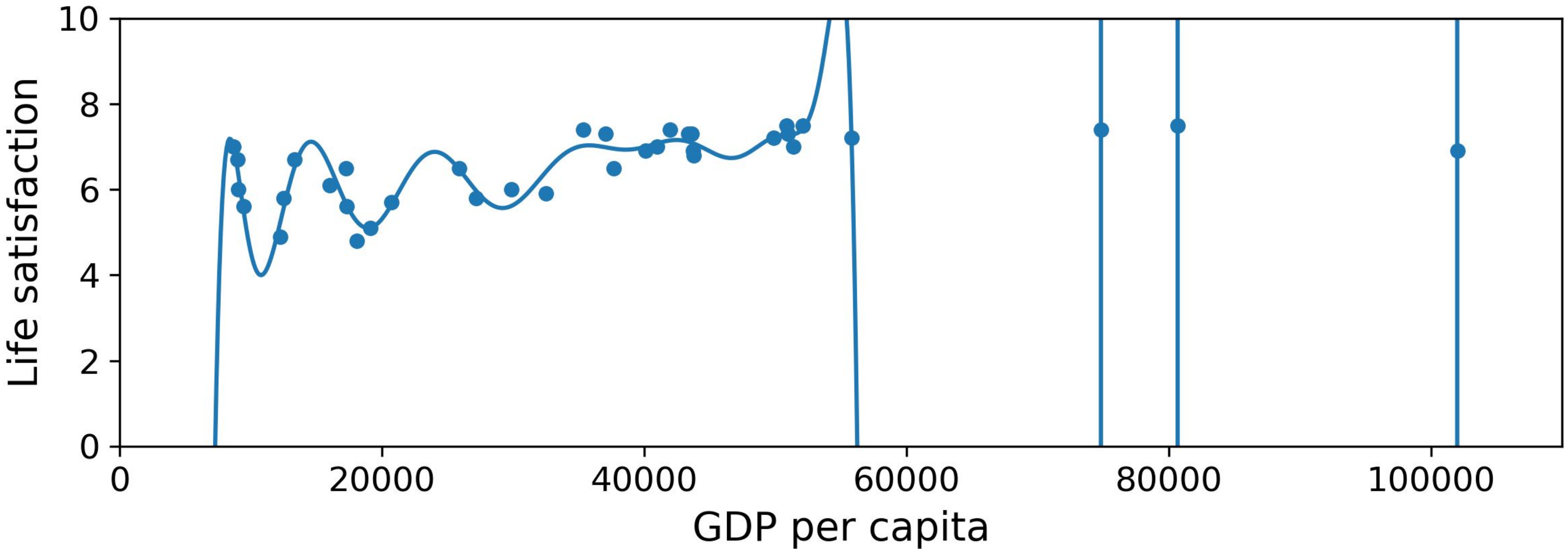
4. Irrelevant Features

GIGO principle: training data must have enough relevant features and not too many irrelevant ones → **feature engineering**

- Feature **Selection** → find useful ones
- Feature **Extraction** → combining existing features to make more useful one
- Feature **Creation** → create new by collecting new data

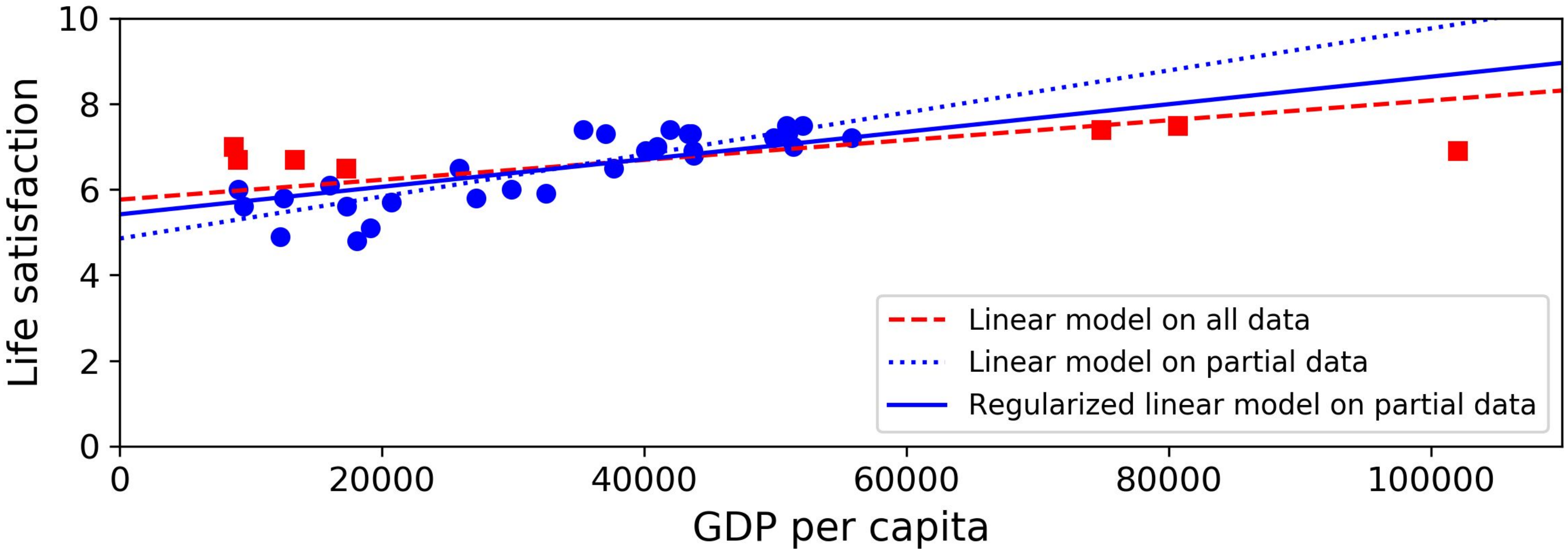
Bad Algorithm

Overfitting the Training Data



The model performs well on training data, but does not generalize well

Overcome overfitting with regularizations



Controlled by a hyperparameter (for the learning algorithm, not the model)

Underfitting the training data

Opposite of overfitting

The model too simple to learn the underlying structure (ie. horizontal line)

How to fix:

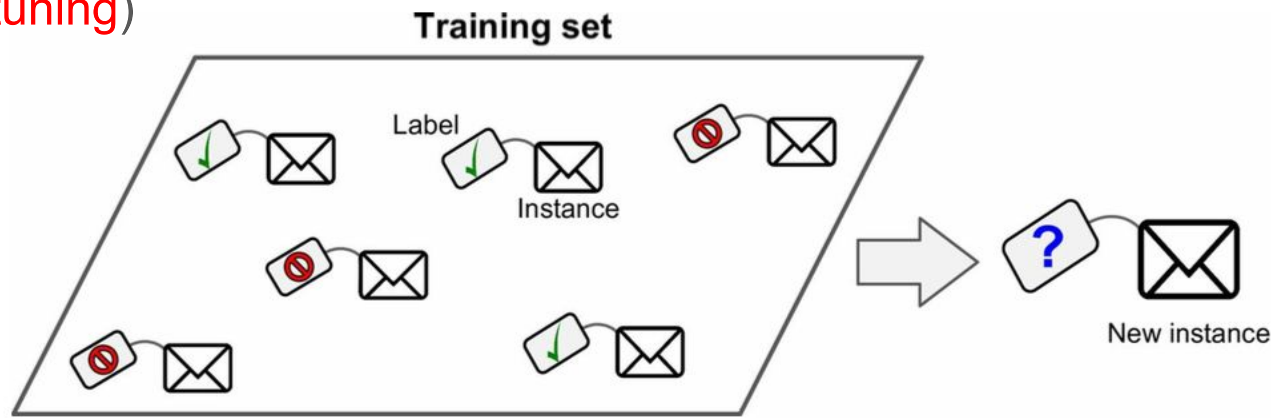
- Select more powerful model, more parameters
- Feed better features
- Reduce constraints (hyperparameters)

Evaluating ML Algorithms

Testing and Validating

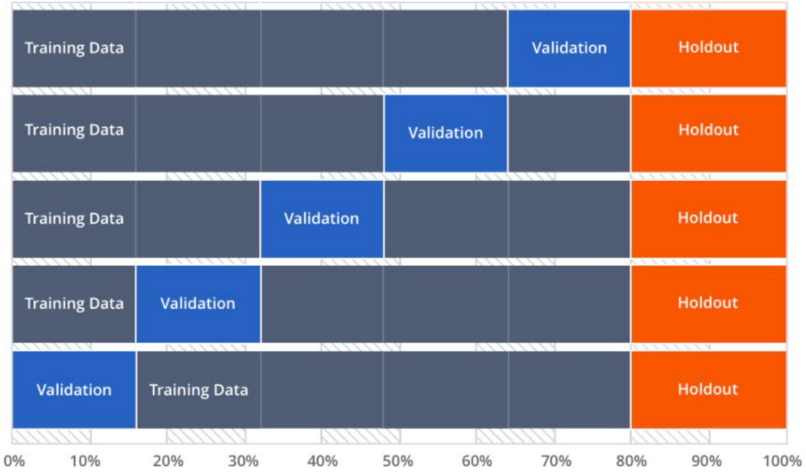
Split data into 2-3 partitions:

- **Training Set:** train your model → training error
- **Testing Set:** evaluate your model → generalization error
- **Validating Set:** If training error is low and generalization error is high → your algorithm overfits. Using validating set, choose value of hyperparameter → avoid overfitting (tuning)

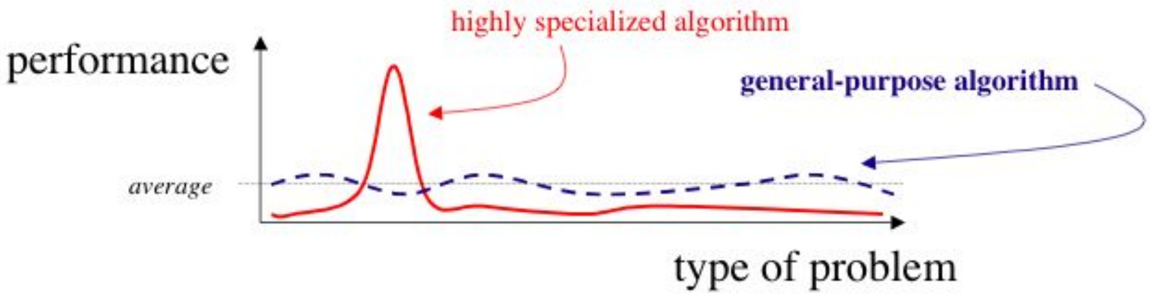


Cross Validation

1. Split data into complementary subsets, each model against a different combination of these subsets, and **validate** against the remaining parts.
2. Select the model type and hyperparameters which yields small training errors
3. The final model is trained using the hyperparameters on full training set
4. Measure the generalized error on the **test set (holdout)**.



No Free Lunch (NFL) Theorem



A model is just a **simplified** version of the observations (data)→ decide which part of the data to keep and which to discard → make some **assumptions** (ie. Linear Assumption in Linear Regression)

NFL Theorem (David Wolpert, 1996): if you make absolutely no assumption about the data, then there is no reason to prefer one model over any other. The only way to know for sure which model is best is to **evaluate them all**.

In practice, you make some **reasonable assumptions** about the data and evaluate **only a few reasonable models**.

Summary: Learning Outcomes

✓ Understand **problems** for which ML is great
✓ Know some basic ML **vocabulary**
✓ Identify **supervised** tasks versus **unsupervised** tasks
✓ Take a sneak peak at **linear regression**
✓ Be aware of some ML **challenges** and ways to fix them

By now, you know **quite a bit** about ML concepts :)

Coming up: A complete example of an end-to-end ML project.

Unused Slides

Activity: Data challenge?

1. Use the same ML problems that you have earlier
2. Predict which **challenges** about the data your problem might have. How would you **fix** them?
3. Discuss with a **neighbor**
4. Share with the class