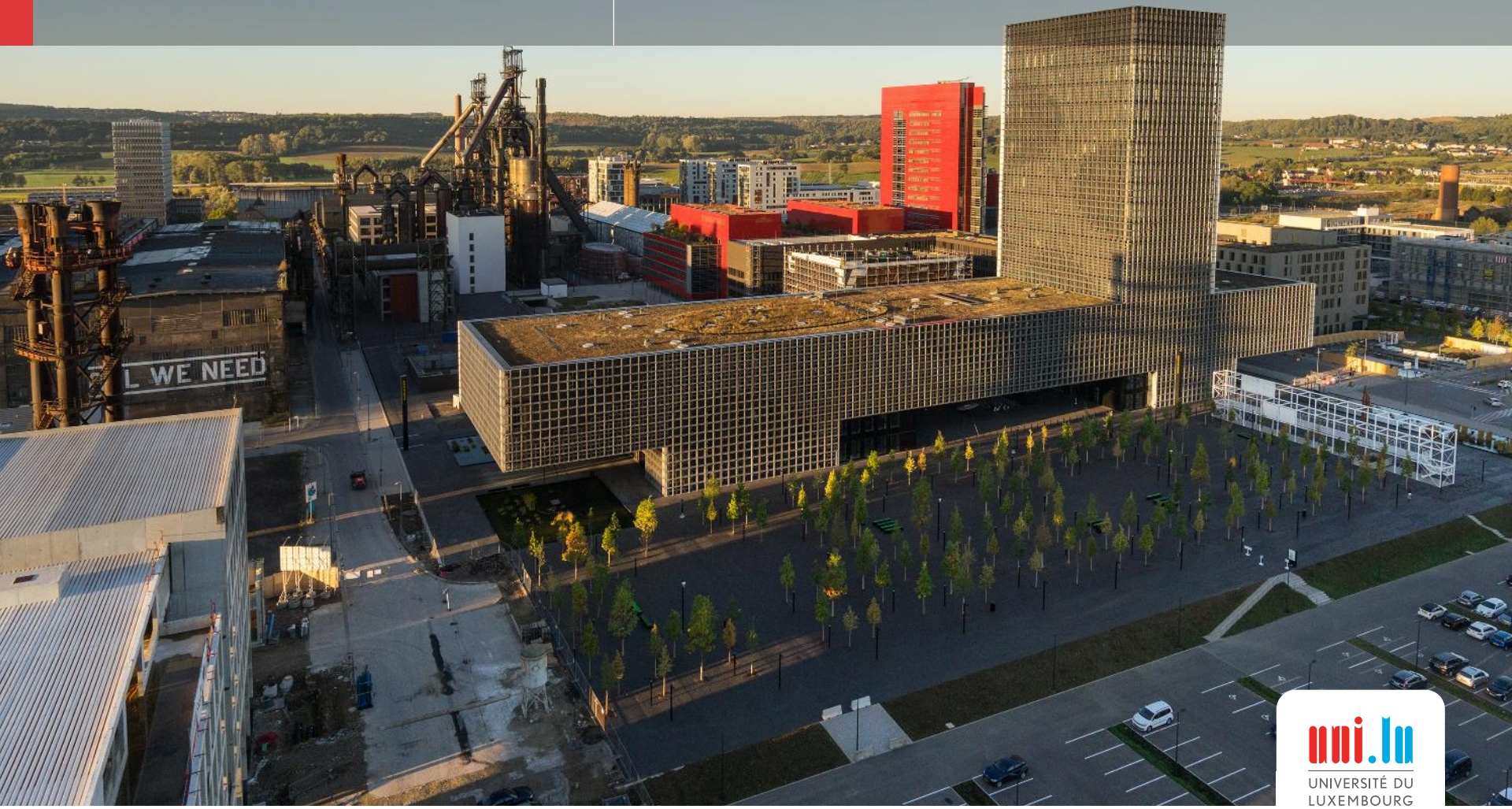
University of Luxembourg

Multilingual. Personalised. Connected.

Software engineering – Part 2

Martin GUBRI, 02/04/2020



UNIVERSITÉ DU LUXEMBOURG

Welcome to the 2nd part of the course on Software development & Machine Learning

Martin Gubri

PhD student at SnT in the Serval group.
Working on Machine Learning Security and Reliability, focusing on adversarial examples.

- Master in Statistics and Econometrics, Toulouse School of Economics, 2014
- Specialized Master in Data Science, Ensae ParisTech, 2015
- Data Scientist in Freelance, 2017-2019
- Ford–Mozilla Technology Exchange Fellow, ONG Derechos Digitales, 2016

1. Machine Learning Basics
    1. Definition of Machine Learning & Deep Learning
    2. Tasks in Machine Learning
    3. Short introduction to Statistical Learning Theory
    4. Short recalls of Statistics

2. Intro to Machine Learning Engineering
    1. When to (not) use Machine Learning
    2. Overview of the life cycle of ML project
    3. Details of the Model Design phase of the life cycle

UNIVERSITÉ DU LUXEMBOURG

Machine Learning Basics

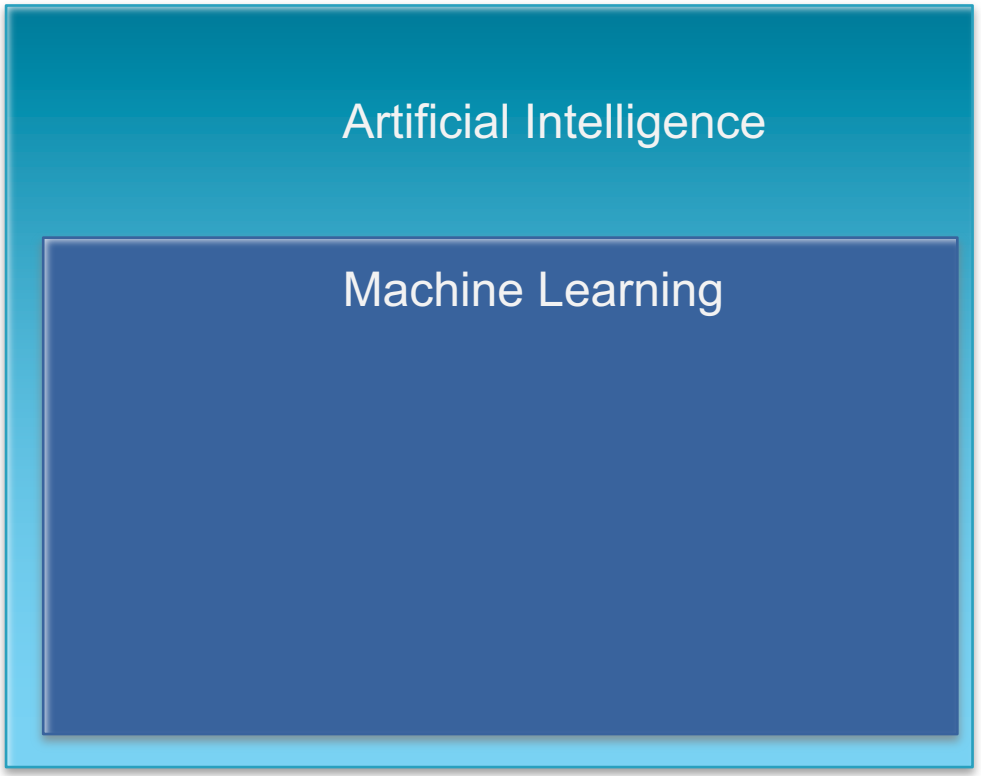
Definitions

Definition of Machine Learning

AI: “any device that perceives its environment and takes actions that maximize its chance of successfully achieving its goals” (source)

Machine Learning
Performing a specific task without beeing explicilty programmed for and using data instead.

“How can computers learn to solve problems without being explicitly programmed?”
Arthur Samuel (1959)

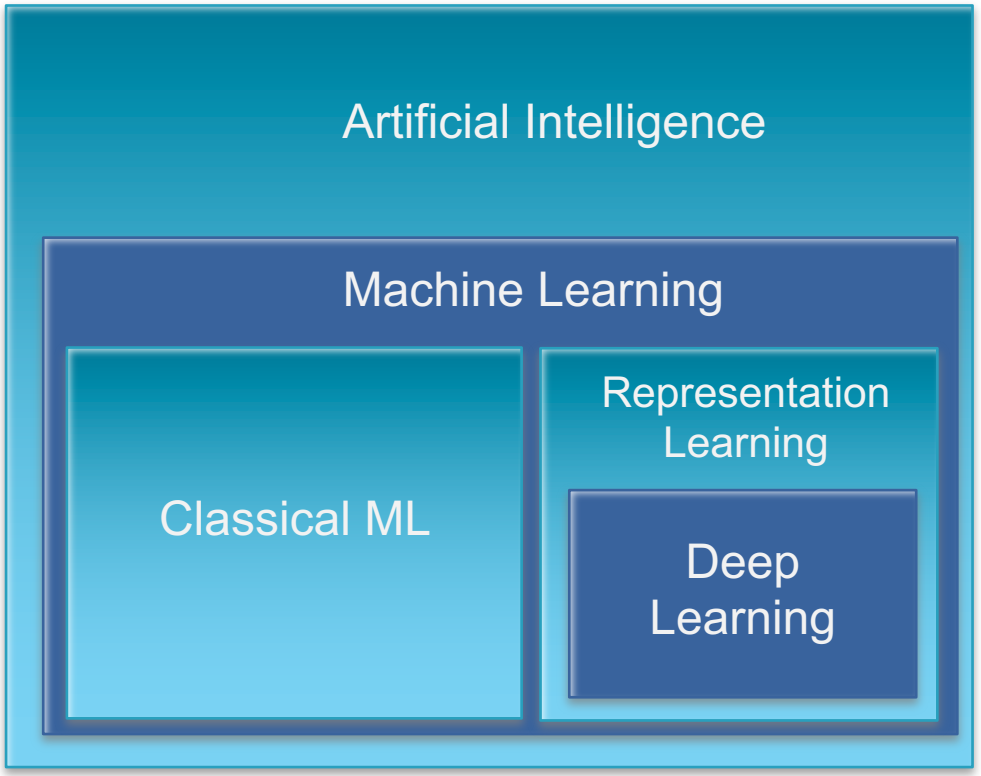


Definition of Machine Learning

Classical Machine Learning
The engineer defines features from raw data. The model learns from this fixed representation

Representation Learning
Learn both the features from raw data and how use them to perform a specific task. Replaces manual feature engineering

Deep Learning
Representation Learning with Neural Networks



Vocabulary

- *Example*: a person or an object of interest. Usually a row of a dataset
- *Feature*: input variable. Usually a column of a dataset (1-dimensional array)
- *Label*: output variable ("answer" or "result"). Usually a column of a dataset
- *Dataset*: set of examples. Usually represented as a matrix (2-dimensional array)

More: https://developers.google.com/machine-learning/glossary

UNIVERSITÉ DU LUXEMBOURG

Machine Learning Basics

Type of Tasks

Supervised Learning:

Available data for each example:

- Input: features (X)
- Output: label (y)

Goal: Learning a mapping function f

$$\text{f: } \mathcal{X} \rightarrow \mathcal{Y}$$
$$X \mapsto \text{y}$$

Spam filtering

Spambase
https://archive.ics.uci.edu/ml/datasets/Spambase

Example: an email

Features: frequencies of some characters or words (“$”, “viagra”, etc.)

Label:
1 = user marked the email as spam
0 = user marked the email as ham

→ Binary classification

Source image: “Spam” sketch, Monty Python's Flying Circus, 1972

Optical character recognition (OCR)

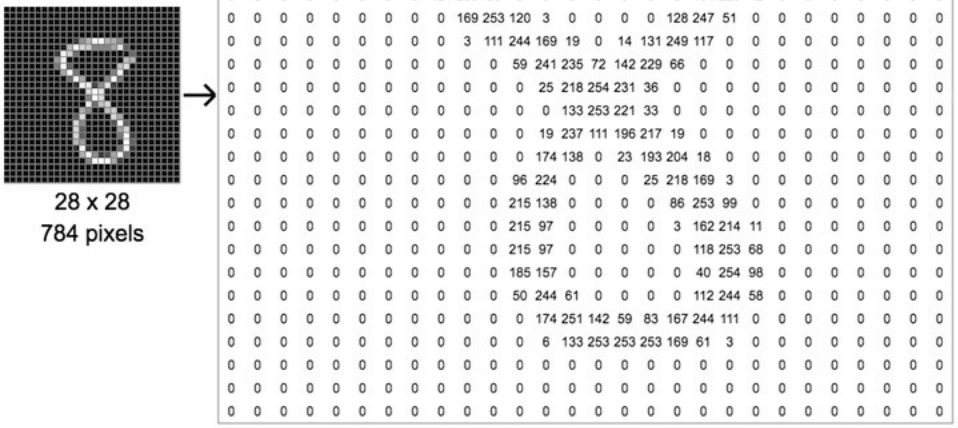
MNIST dataset
http://yann.lecun.com/exdb/mnist/

Example: grayscale image of a digit

Features: (flattened) pixels values

Label: value of the digit (0-9)

→ Multiclass classification



House price prediction

Boston Housing dataset
https://www.cs.toronto.edu/~delve/data/boston/bostonDetail.html

Example: a house

Features: average number of rooms, age of the house, per capita crime rate of town, etc.

Label: price of the house

→ Regression (continuous target variable)

Source image: https://www.kaggle.com/c/boston-housing

Unsupervised Learning:

Available data for each example:

- only features (X)
- no label

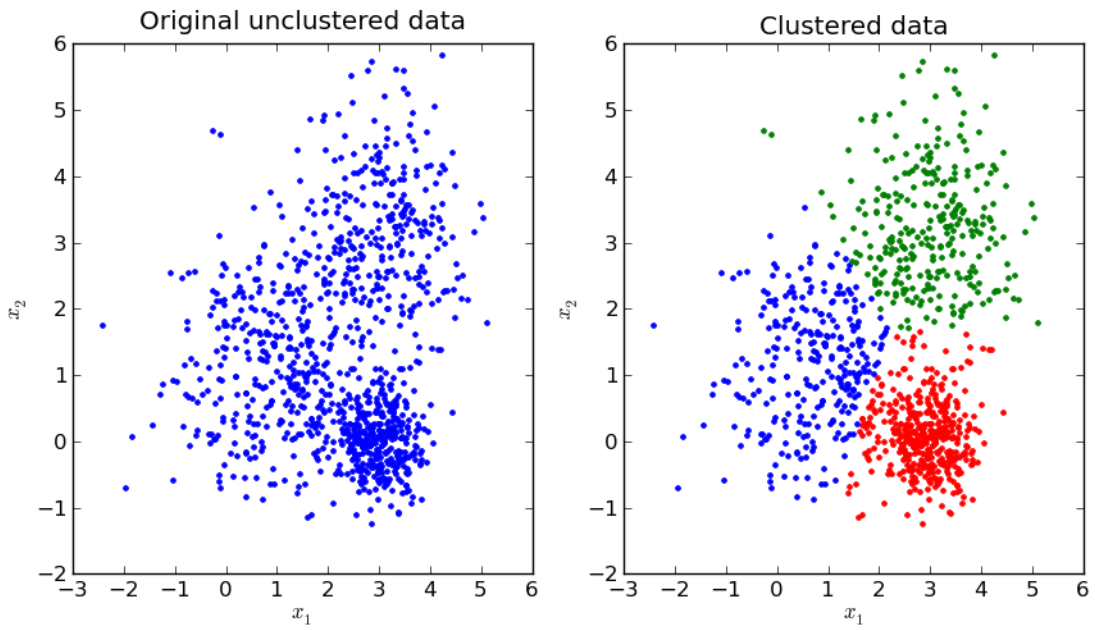
Goal: discovering patterns in the data

Market segmentation

Example: a client

Features: client bought recently, filled a complain, bought a specific type of products, etc.

No label, just a bunch of clients.



Goal: discover groups of similar clients, dissimilar to other groups. To have differentiated communication, offers, etc.

→ Clustering

Source image: http://pypr.sourceforge.net/_images/kmeans_2d.png

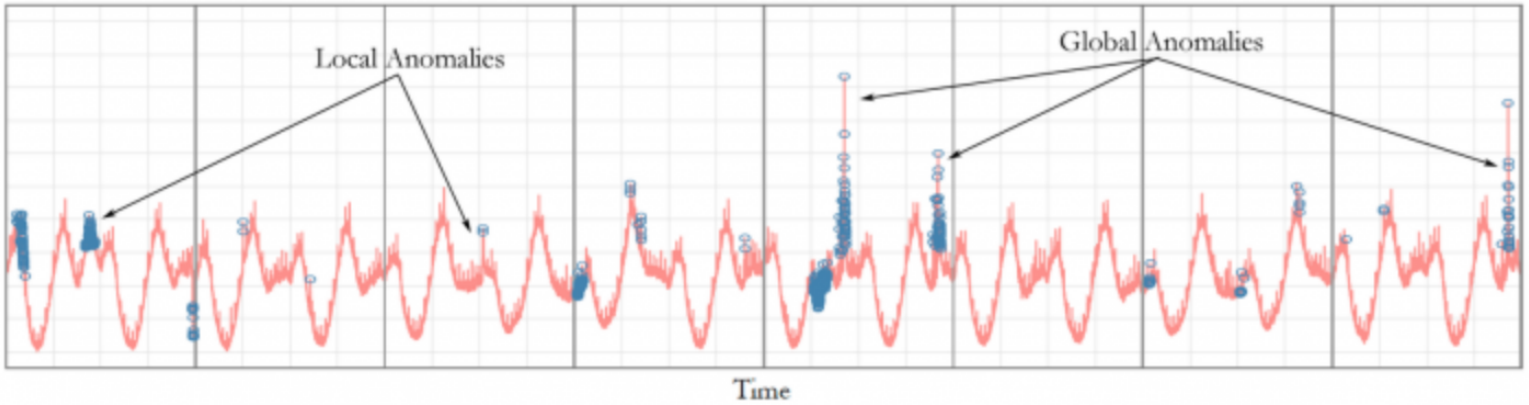
Intrusion detection systems

Example: state of a system at time t

Features: number of requests

No label, just a lot of requests

→ Anomaly detection



Source image: https://blog.twitter.com/engineering/en_us/a/2015/introducing-practical-and-robust-anomaly-detection-in-a-time-series.html

UNIVERSITÉ DU LUXEMBOURG

Machine Learning Basics

Short Recall of Statistics

Descriptive Statistics

Statistic	*Formula*	*Used For*
Sample mean (average)	$\bar{x} = \frac{\sum x}{n}$	Measure of center; affected by outliers
Median	*n* odd: middle value of ordered data *n* even: average of the two middle values	Measure of center; not affected by outliers
Sample standard deviation	$s = \sqrt{\frac{\sum (x - \bar{x})^2}{n-1}}$	Measure of variation; "average" distance from the mean
Correlation coefficient	$r = \frac{1}{n-1} \sum \frac{(x - \bar{x})(y - \bar{y})}{s_x s_y}$	Strength and direction of linear relationship between *X* and *Y*

Source image: https://www.dummies.com/education/math/statistics/statistics-for-dummies-cheat-sheet/

Descriptive Statistics

Don’t rely only descriptive statistics. Use data visualization (among others).

X Mean: 54.2659224
Y Mean: 47.8313999
X SD : 16.7649829
Y SD : 26.9342120
Corr. : -0.0642526

Source image:
https://www.autodeskresearch.com/publications/samestats

Inferential Statistics

Inferential Statistics: infer properties of population from a sample

Simpsons Paradox

- https://www.youtube.com/watch?v=sxYrzzy3cq8
- https://www.youtube.com/watch?v=ebEkn-BiW5k (optional)

Hypothesis testing (statistical test)

- https://www.youtube.com/watch?v=I10q6fjPxJ0
- https://www.youtube.com/watch?v=VK-rnA3-41c (optional)

A/B test

- Application of Statistical test in Software Engineering
- https://www.youtube.com/watch?v=zFMgpxG-chM (mandatory)
- https://youtu.be/VQpQ0YHSfqM?t=140 (you can start at 2:20)

UNIVERSITÉ DU LUXEMBOURG

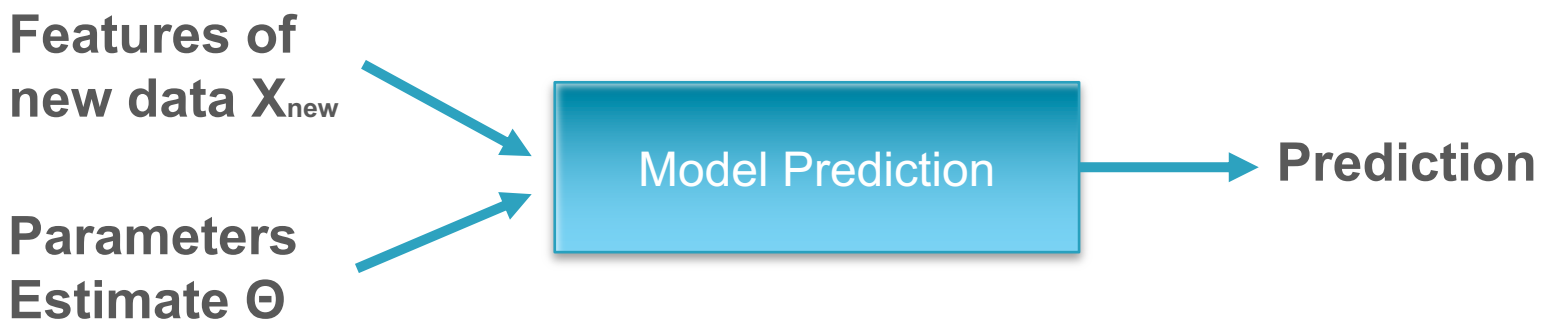
Machine Learning Basics

Overview

Overview of the 2 phases

Training phase

Operational phase



Hyperparameters & Parameters

Parameters

Internal values of the model learned during training

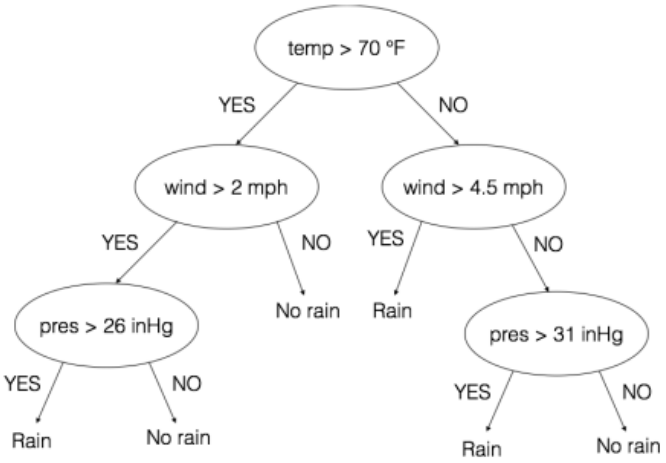
Learned values are called “Parameters Estimate”

E.g. feature and threshold of each node in a decision tree

Hyperparameters

Values that influence the model and cannot be learn

E.g depth of a decision tree



Example of a Decision Tree

Source image:http://ebubekirsezer.com/en/decision-treekarar-agaci/

UNIVERSITÉ DU LUXEMBOURG

Machine Learning Basics

Some considerations of Statistical Learning Theory

How to learn from data?

$f(x_i)$: the model's prediction for i-th example

Loss: a measure of how far a model's predictions are from its label

In binary classification: $L(y_i, f(x_i)) = \mathbb{I}(y_i \neq f(x_i))$

Empirical Risk: Average of the loss across the dataset's examples

$$R_{emp}((X, y), f) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i))$$

Empirical Risk Minimization: finding a model that minimize the empirical risk

$$\min_f R_{emp}((X, y), f)$$

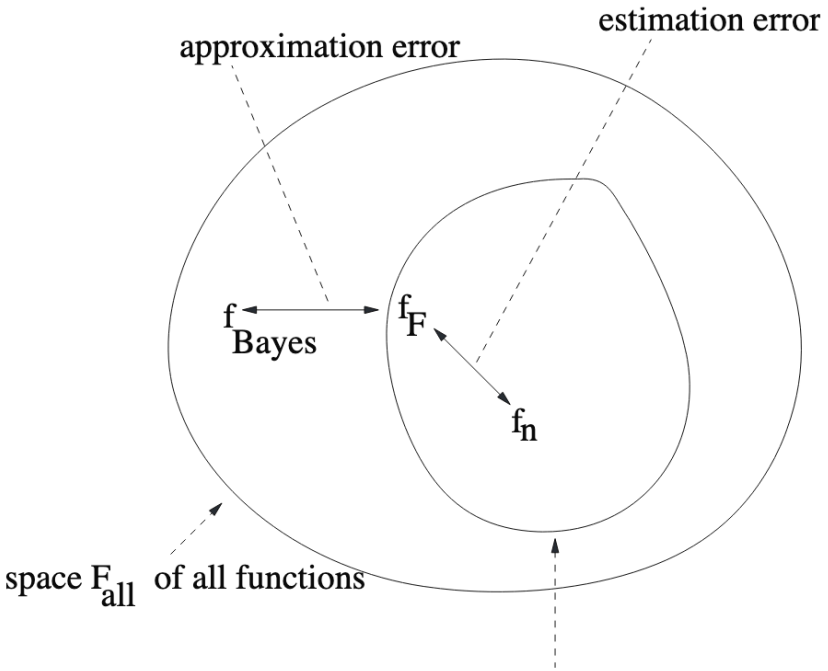
Decomposition of the error

$$R(f_n) - R(f_{Bayes}) = \underbrace{\left(R(f_n) - R(f_{\mathcal{F}})\right)}_{\text{estimation error}} + \underbrace{\left(R(f_{\mathcal{F}}) - R(f_{Bayes})\right)}_{\text{approximation error}}$$

your trained model

"best" unknown model possible with the current set of features

best unknown model possible with the chosen algorithm (and the current set of features)



Source image:
U. von Luxburg and B. Scholkopf. "Statistical Learning Theory: Models, Concepts, and Results",2011.

Decomposition of the error

$$R(f_n) - R(f_{Bayes}) = \underbrace{\left(R(f_n) - R(f_{\mathcal{F}})\right)}_{\text{estimation error}} + \underbrace{\left(R(f_{\mathcal{F}}) - R(f_{Bayes})\right)}_{\text{approximation error}}$$

VARIANCE ⟷ BIAS

Source image:
U. von Luxburg and B. Scholkopf.
“Statistical Learning Theory: Models, Concepts, and Results”,2011.

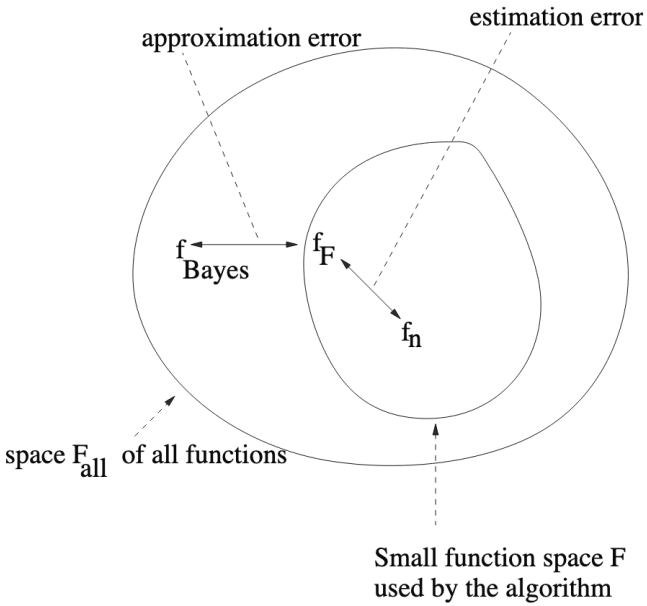
Bias-Variance tradeoff

Biais

Large approximation error, small estimation error → **Underfit**

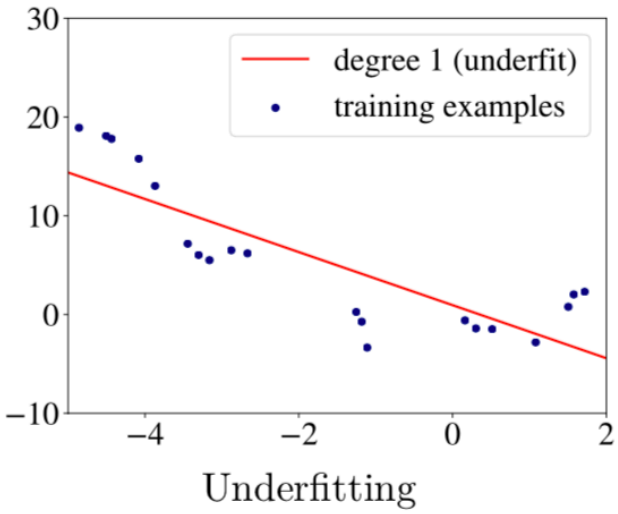
Variance

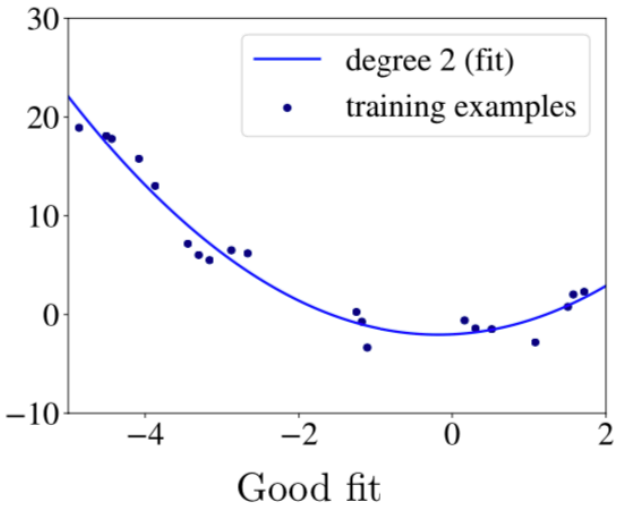
Large estimation error, small approximation error → **Overfit**

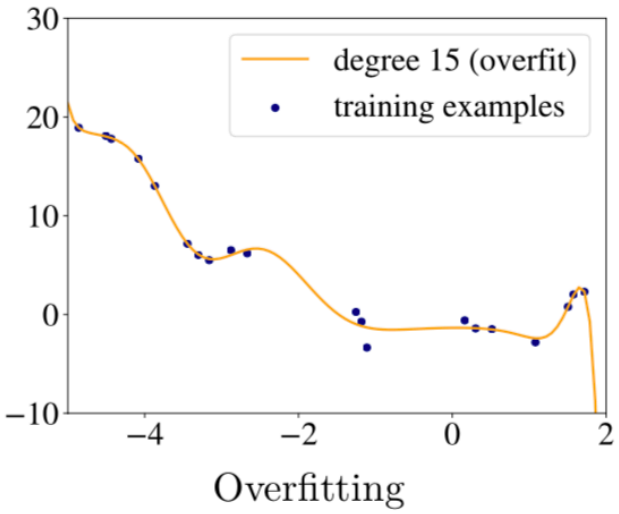


Overfitting & Underfitting

Bias-Variance tradeoff







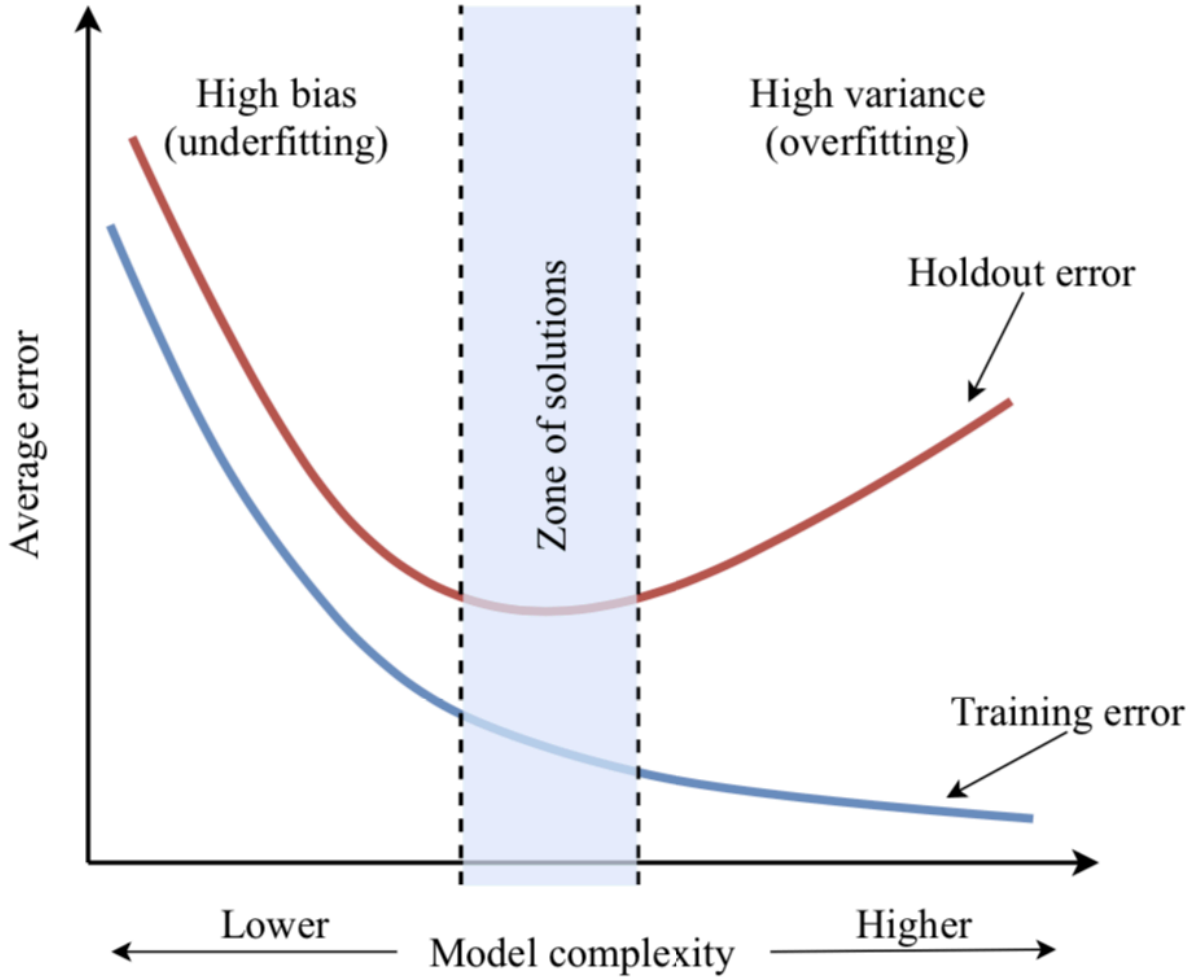
Oversimplification of the data

Learning the true signal

Fitting the noise of the data

Source image:
A. Burkov, "Machine Learning Engineering".

Bias-Variance tradeoff



Source image:
A. Burkov, "Machine Learning Engineering".

UNIVERSITÉ DU LUXEMBOURG

Overview of the life cycle of ML project

ML project lifecycle

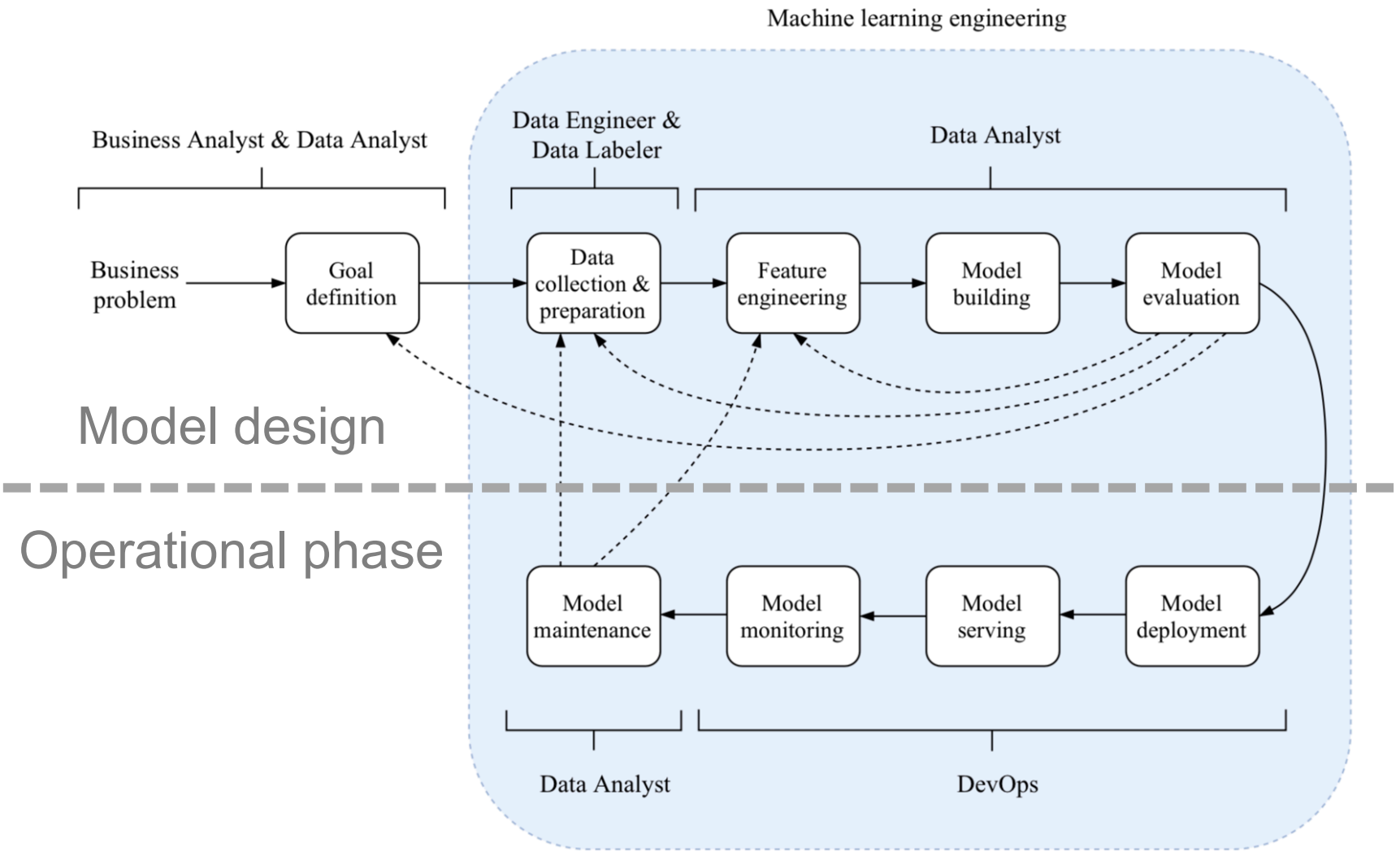


Figure 3: Machine learning project life cycle.

Source image:
A. Burkov, "Machine Learning Engineering".

ML project lifecycle

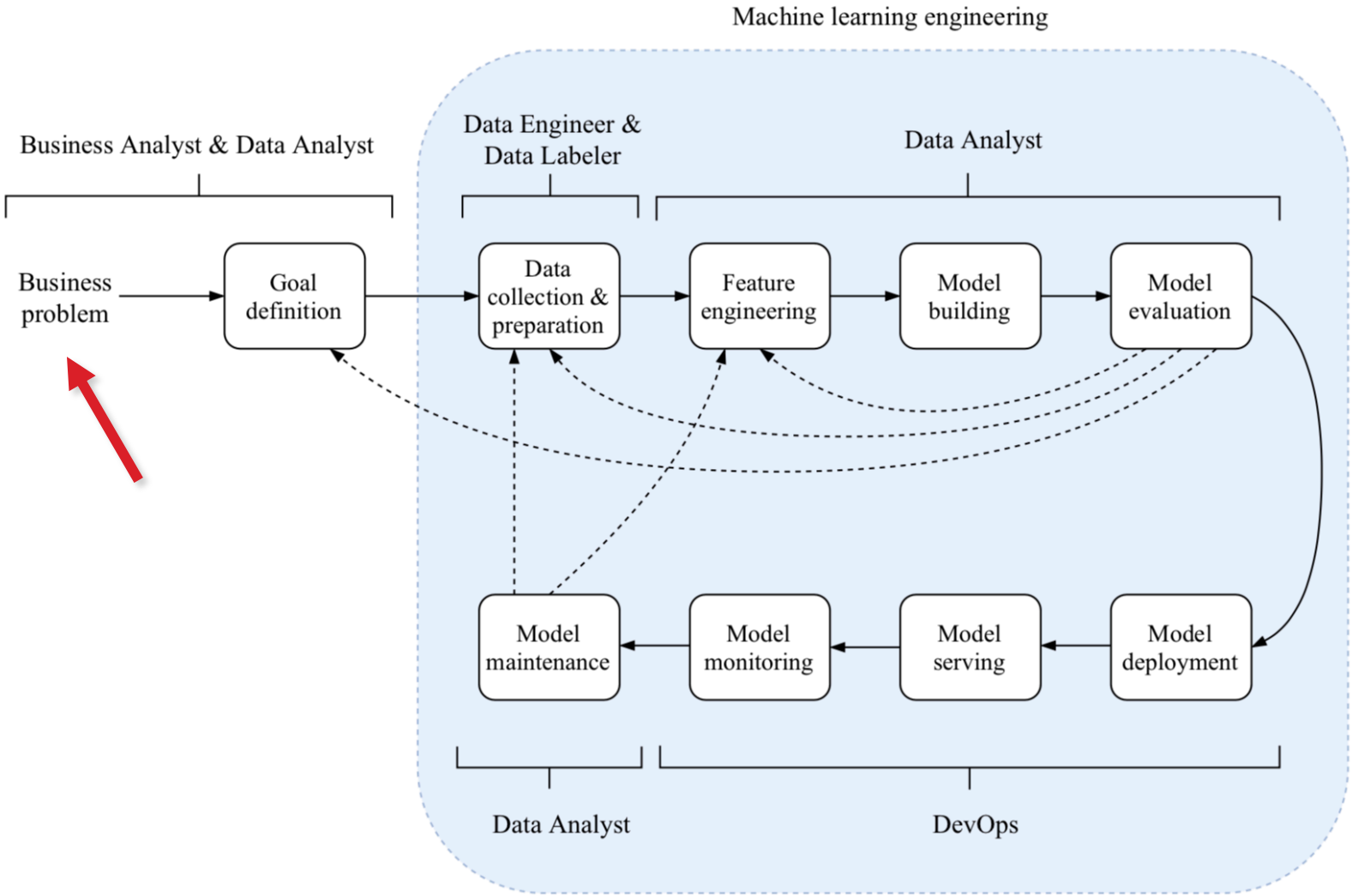


Figure 3: Machine learning project life cycle.

Source image:
A. Burkov, "Machine Learning Engineering".

When to use Machine Learning?

Consider using ML in those situations:

- the problem is too complex for coding
- the problem is constantly changing
- it is a perceptive problem (e.g. image, speech, and video recognition)
- it is an unstudied phenomenon
- the problem has a simple objective (e.g. yes/no decision, single number target)
- when it is cost-effective
  - 3 main costs: building the model, infrastructure to serve the model, model maintenance

Source content:
A. Burkov, “Machine Learning Engineering”.

When to **not** use Machine Learning?

When to **not** use Machine Learning?

Probably better to not use ML when:

- getting right data is too complex or impossible
  - no relation between features and labels
  - the phenomenon has too many outcomes while you cannot get enough examples to represent those outcomes
- you know how to solve the problem using traditional software development at a lower cost
  - e.g. you can fill an exhaustive lookup table manually by providing the expected output for any input
- every decision made by the system has to be explainable
  - can be a GDPR requirement
  - idem for decision changes in similar situations over time
- the cost of an error made by the system is too high

Source content:
A. Burkov, “Machine Learning Engineering”.

ML project lifecycle

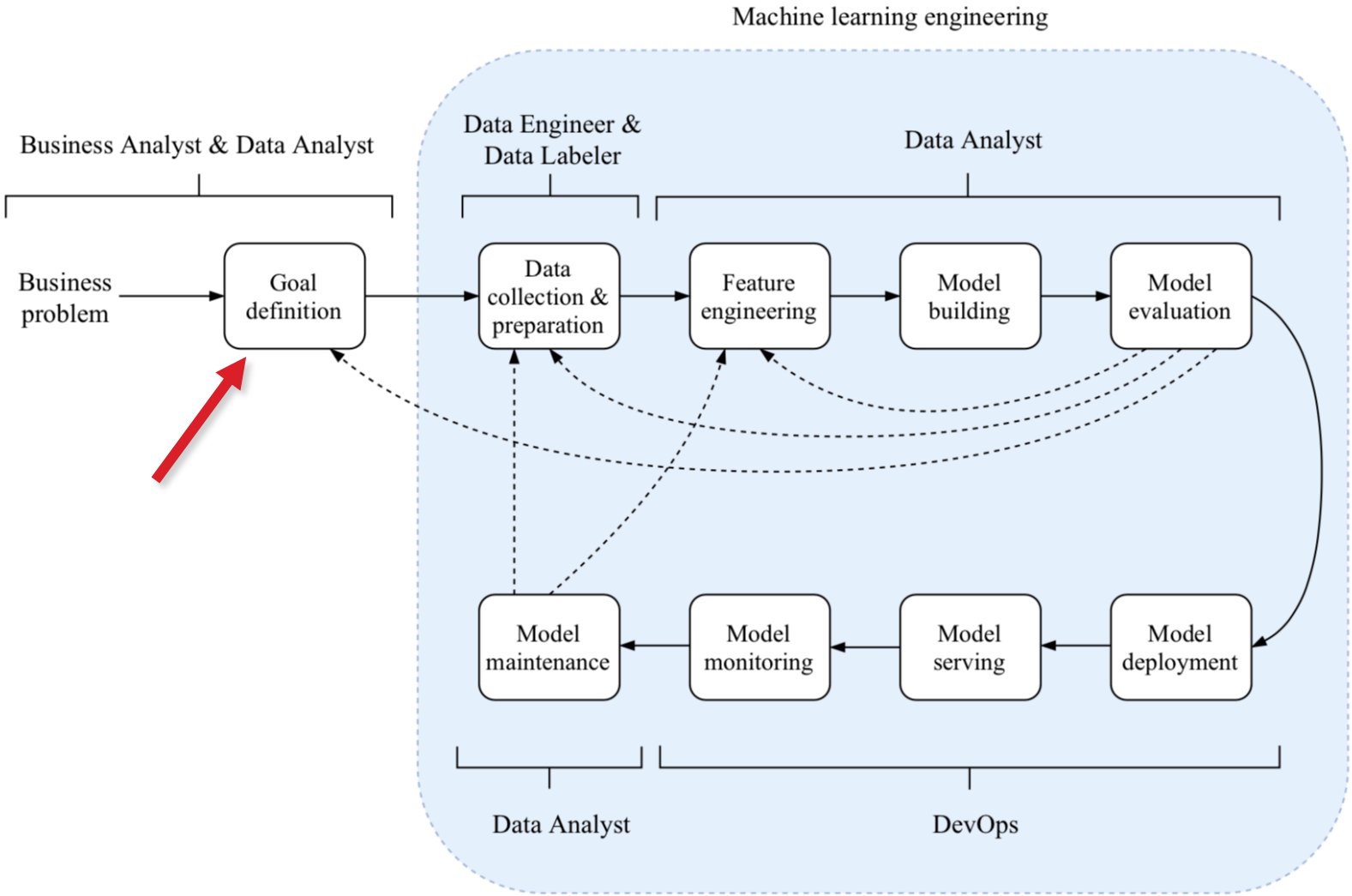


Figure 3: Machine learning project life cycle.

Source image:
A. Burkov, "Machine Learning Engineering".

Goal definition

Write down the elements of your problem

- What is my objective?
- What is my task?
- What is an example?
- What is the expected output?
- What are acceptable behaviors? Unacceptable behaviors?
  - Think of performances, security, discriminations, privacy, ethical issues, etc.
- How to measure them?
- **Lastly**, do I have available data? If not, can I build a dataset?

Adapted from:
A. Burkov, "Machine Learning Engineering".

Goal definition

Example

Objective: be the best email provider

Task: automatically detect spam

Example: an email

Output: spam / ham

Acceptable behaviors:

- classify most of the spam, measured by probability of detection (sensitivity or recall) of at least 75%

Unacceptable behaviors:

- (performances) should only classify a very small amount of ham as spam, measured by specificity (true negative rate) of at least 95%
- (security) spammer should not fond easy way to fool the spam filter
- (privacy) email content should not be leaked from the trained model

Goal definition

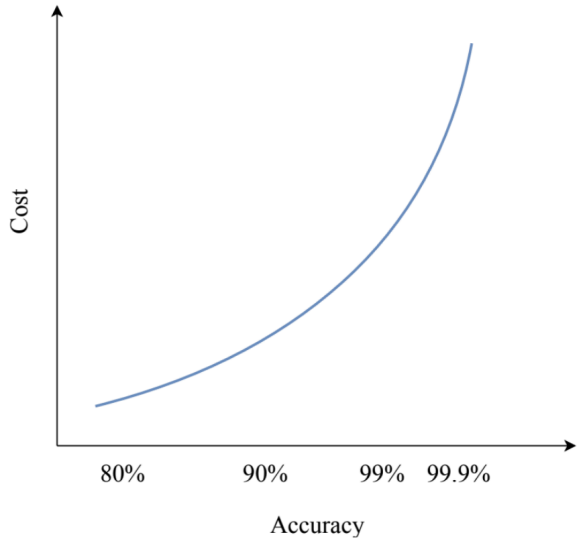
Content & figure from: A. Burkov, "Machine Learning Engineering".

Conduce a **Cost-Benefit Analysis**

- Learn more: https://en.wikipedia.org/wiki/Cost%E2%80%93benefit_analysis

To estimate the **cost** consider:

- the difficulty of the problem
  - existing library
  - computation to train & serve the model
- the cost of data
  - can data be generated automatically?
  - cost of manual annotation
    labelling cats & dogs images ≠ doctors labelling X-ray scans
  - how many examples are approximately needed
- the needed accuracy
  - how costly is each wrong prediction?
  - lowest accuracy level below which the model is impractical



ML project lifecycle

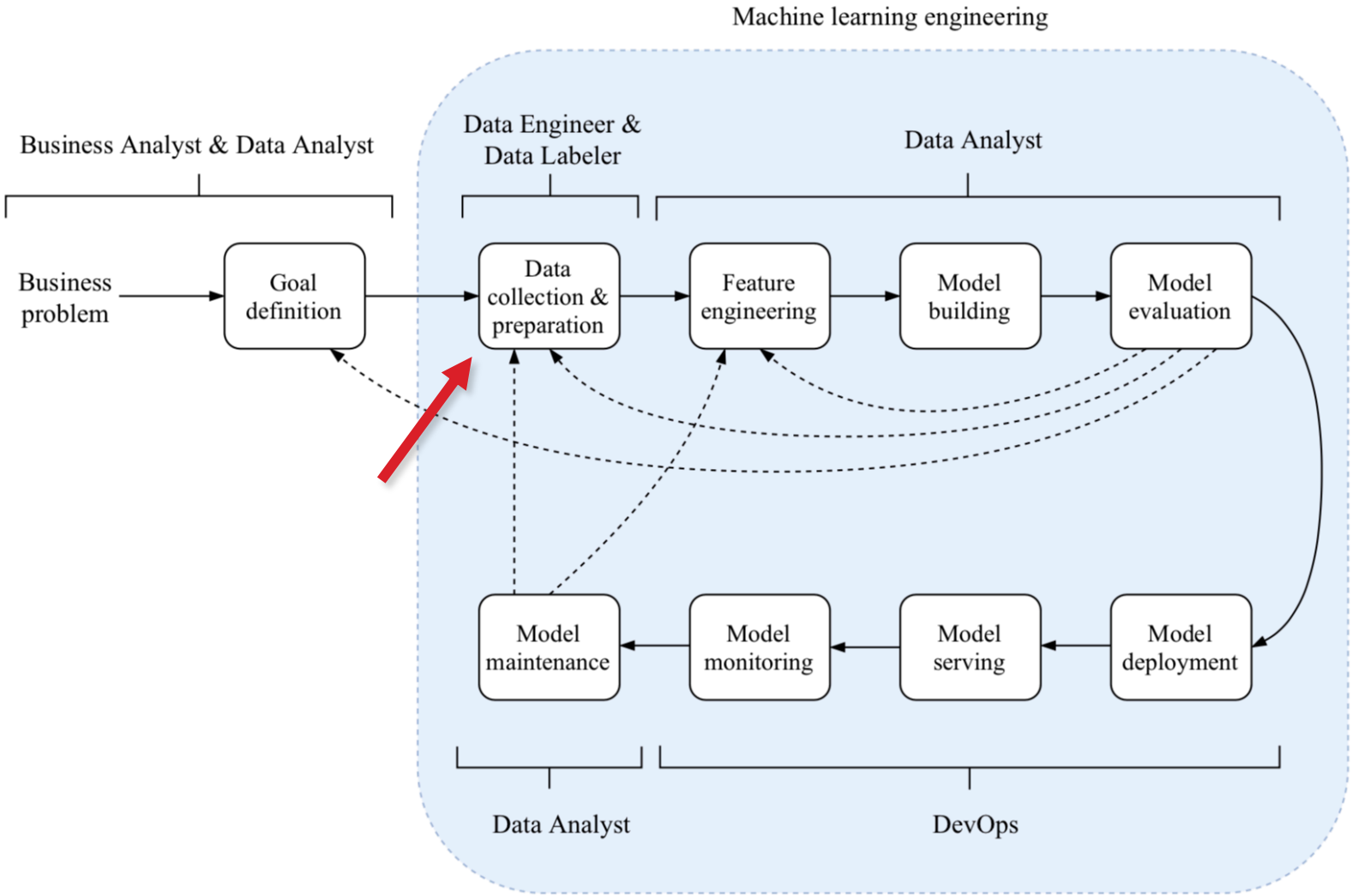


Figure 3: Machine learning project life cycle.

Source image:
A. Burkov, "Machine Learning Engineering".

Data collection & preparation

In industry: 80% building dataset, 20% playing with it

Steps:

- Existing data:
  - access to existing data (negotiation, paperwork, anonymization, etc.), database work, merging datasets, etc.
- Creating a dataset
  - can be very long
  - collecting data, statistical sampling, scraping, survey, labelling data, etc.
- Data cleaning
  - Domain validity, outliers, errors, duplicates, expired data, etc.
- Data augmentation (optional)
  - e.g. image rotation, image mirroring

Data collection & preparation

Advices

Think of **reproducibility** from the beginning

- you might have to extract the same data again in the future: bugs, adding features, update data, etc.
- avoid manual labor on data in Excel: error-prone & difficult to traceback

Always check raw data and processed data

- Don't trust your implementation
- Keep backups of both datasets

Don't assume that your data is valid

- Check domain validity (e.g. postal code, age)
- Properly encode missing values (e.g. "-1", "99999" → NaN)

ML project lifecycle

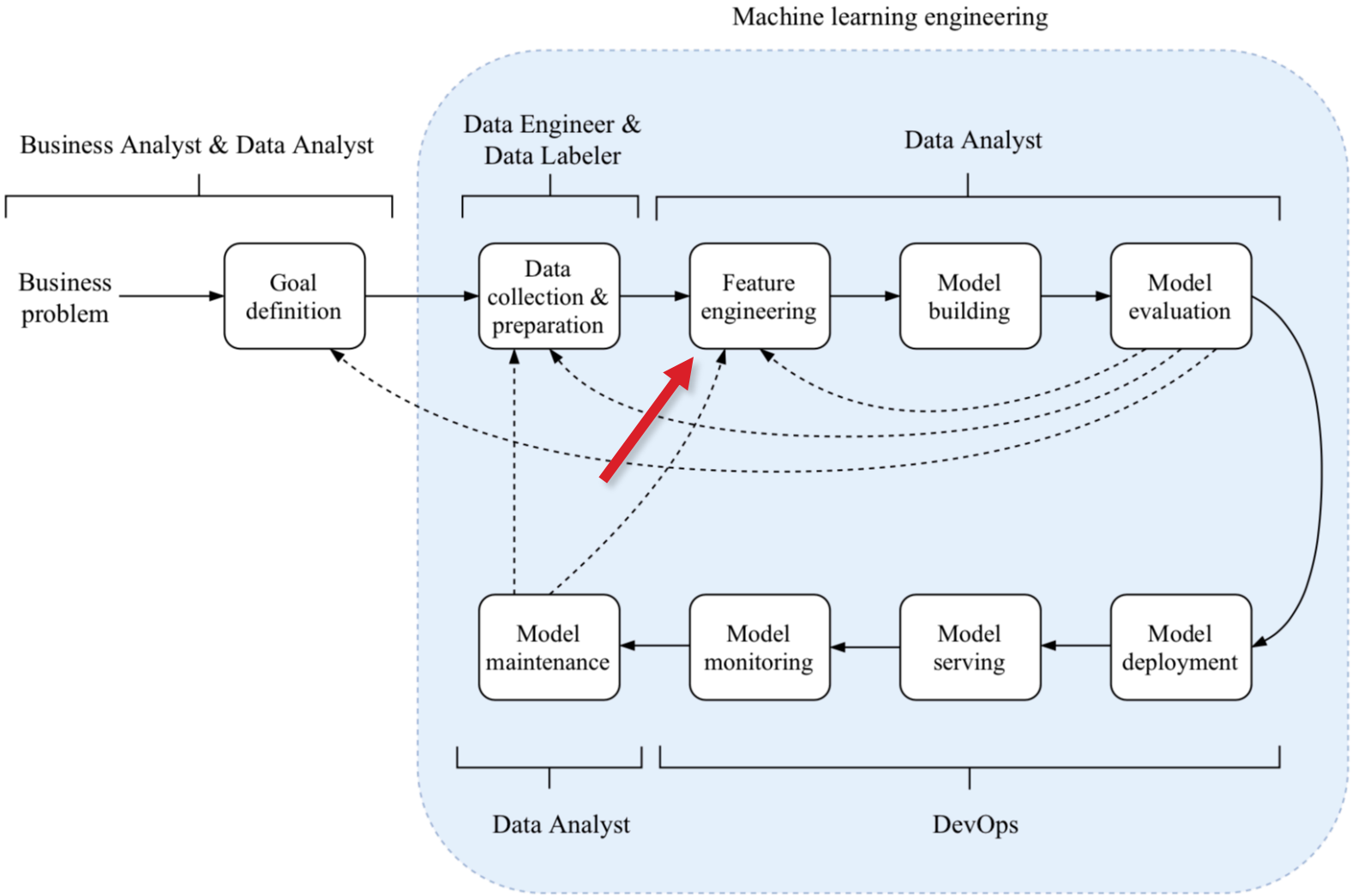


Figure 3: Machine learning project life cycle.

Source image:
A. Burkov, "Machine Learning Engineering".

Feature engineering

Goal

Transform raw data into tidy data with numerical or categorical features

Example:

Plain text email → Frequencies of “$”, “!”, “#”, “viagra”, “buy”, “mail”, etc.
Average length of uninterrupted sequences of capital letters.
Etc.

Feature engineering

Tidy data

Follow principles of **tidy data**

- 1 row ←→ 1 example
- 1 column ←→ 1 feature
- single dataset

Sources of messiness:

- Column headers are values, not variable names
- Multiple variables are stored in one column
- Variables are stored in both rows and columns
- Multiple types of experimental unit stored in the same table
- One type of experimental unit stored in multiple tables

Source content:
Wickham, H. (2014). Tidy Data. Journal of Statistical Software, 59(10)

Feature engineering

Tidy data

	John Smith	Jane Doe	Mary Johnson
treatmenta	—	16	3
treatmentb	2	11	1

person	treatment	result
John Smith	a	—
Jane Doe	a	16
Mary Johnson	a	3
John Smith	b	2
Jane Doe	b	11
Mary Johnson	b	1

See more examples of bad practices:
https://www.jstatsoft.org/article/view/v059i10

Source images:
Wickham, H. (2014). Tidy Data. Journal of Statistical Software, 59(10)

Steps

1. Defining the features to represent well your examples
   - Usually involve domain experts
   - Document everything into a schema file (spreadsheet or Json with name, type, definition, missing values, source, etc.)

2. Implement their computation
   - reproducibility + checks
   - data manipulation: filter, transform, aggregate, sort

Summarizing data

You can compute **mean & standard deviation** for each feature to aggregate

Example of Churn Analysis

User

User ID	Gender	Age	...	Date Subscribed
1	M	18	...	2016-01-12
2	F	25	...	2017-08-23
3	F	28	...	2019-12-19
4	M	19	...	2019-12-18
5	F	21	...	2016-11-30

Order

Order ID	User ID	Amount	...	Order Date
1	2	23	...	2017-09-13
2	4	18	...	2018-11-23
3	2	7.5	...	2019-12-19
4	2	8.3	...	2016-11-30

Call

Call ID	User ID	Call Duration	...	Call Date
1	4	55	...	2016-01-12
2	2	235	...	2016-01-13
3	3	476	...	2016-12-17
4	4	334	...	2019-12-19
5	4	14	...	2016-11-30

Figure 24: Relational data for churn analysis.

User features

User ID	Gender	Age	Mean Order Amount	Std Dev Order Amount	Mean Call Duration	Std Dev Call Duration
2	F	25	12.9	7.1	235	0
4	M	19	18	0	134.3	142.7

Figure 25: Synthetic features based on sample mean and standard deviation.

Steps

3. Encoding categorical features
   - Models accept **only numerical** values
   - Categorial features should be encoded as integers (**one-hot encoded**):
     - “male”, “female”, “female” → 0, 1, 1
     - “bachelor”, “master”, “bachelor”, “high school”, “master” →

0	1	0
0	0	1
0	1	0
1	0	0
0	0	1

4. Discretization of continuous variable *(optional):*
   - Age: 8, 65, 42, 15 → “-18”, “50+”, “18-50”, “-18” →

1	0	0
0	0	1
0	1	0
1	0	0

Steps:

4. **Missing data treatment**
   - Most models don't accept missing data
   - Solutions: imputation, categorization, etc.

5. **Data Normalization of continuous features**
   - I.e. mean removal and variance scaling
   - Almost all models performs better with normalized data

6. Other features transformation
   - E.g, polynomial features: (X_1, X_2) to $(1, X_1, X_2, X_1^2, X_1X_2, X_2^2)$

Practical advices

Informative features (high predictive power)

- Constant data are useless
- Totally unrelated features are not useful
- Duplicated features are not useful (e.g. weight in kg and in pounds)

Always check the presence of missing data

```
1  def transform_gender(value):
2      if value == "male":
3          return 0
4      else:
5          return 1
```

Use all data analysis tools to help you

- descriptive statistics (mean, standard deviation, min, max, quantiles, absolute and relative frequencies, etc.), data visualization, etc.

Use small sample to develop and debug

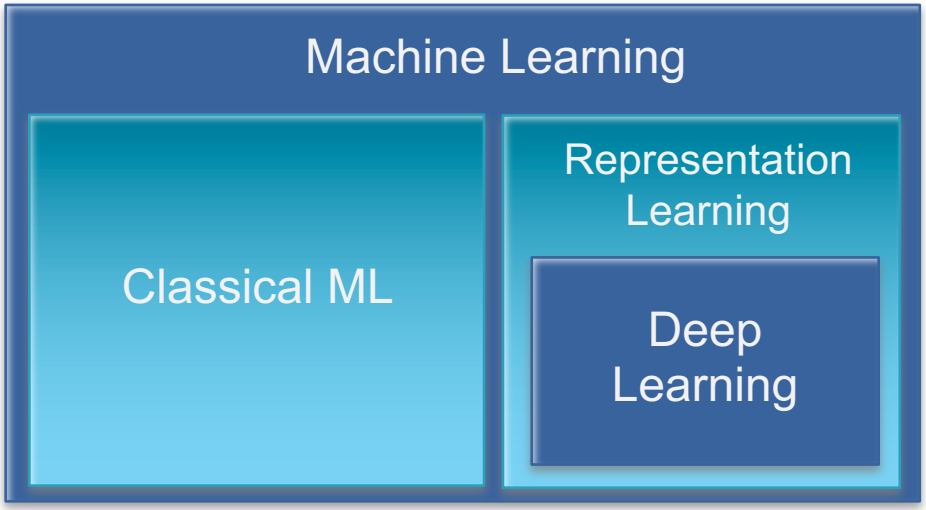
- Unit test for each data extractor

Deep Learning

No feature engineering in Deep Learning

But has a cost

- number of examples (in millions)
- complexity



When to **not** use Deep Learning?

ML project lifecycle

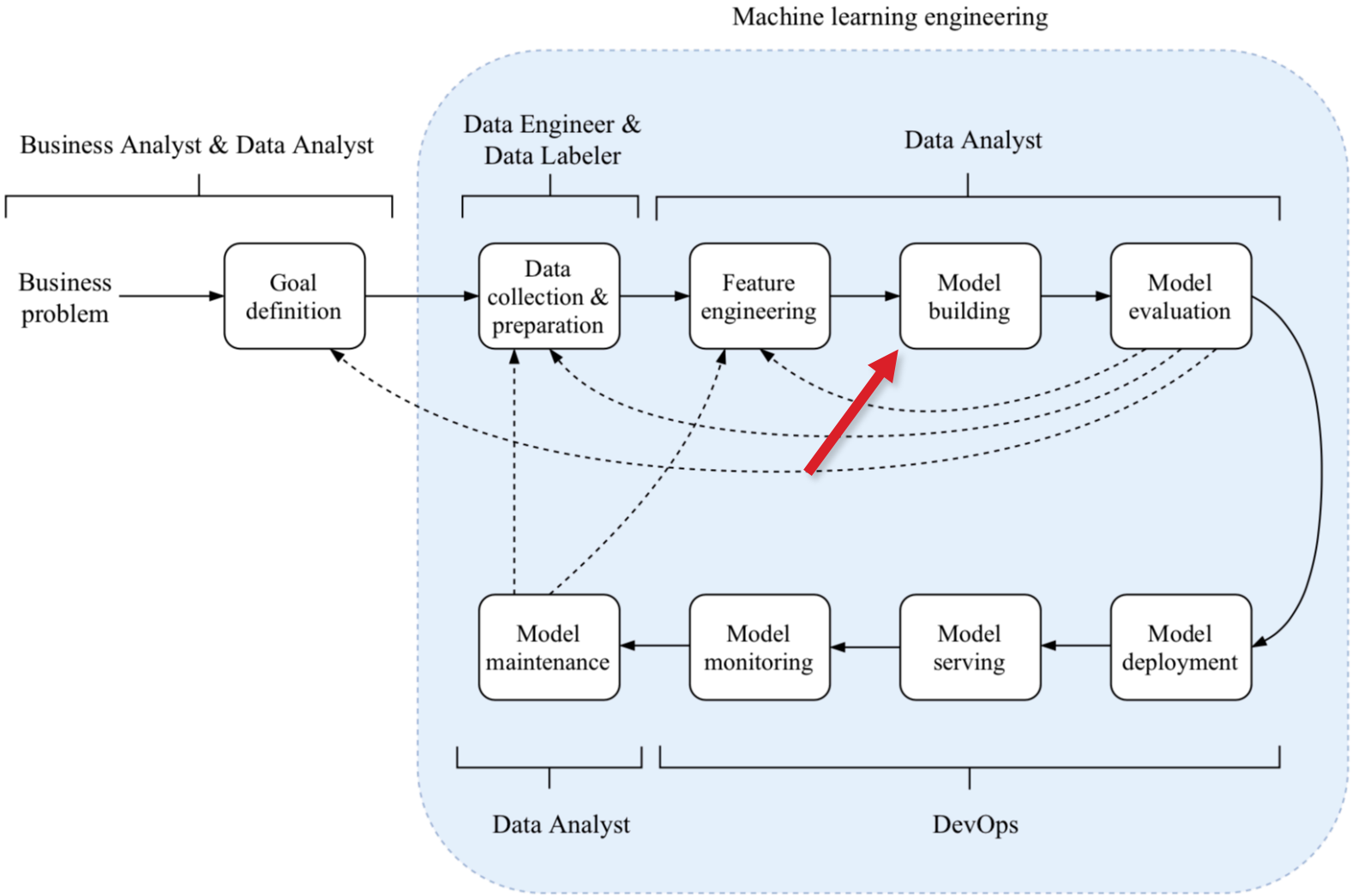


Figure 3: Machine learning project life cycle.

Source image:
A. Burkov, "Machine Learning Engineering".

No free lunch theorem

Theory said that there is no universally best model (Wolpert 1996)
Learning is impossible unless we make assumptions on the underlying distribution.

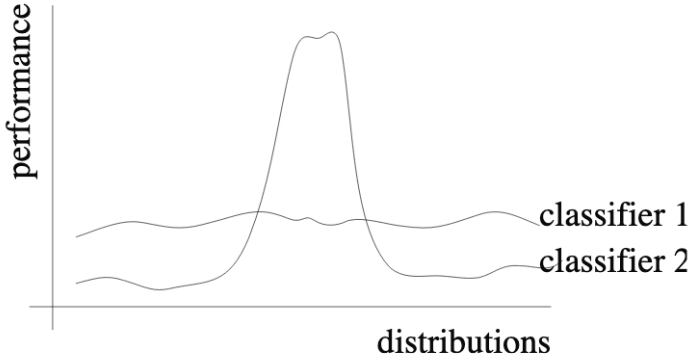


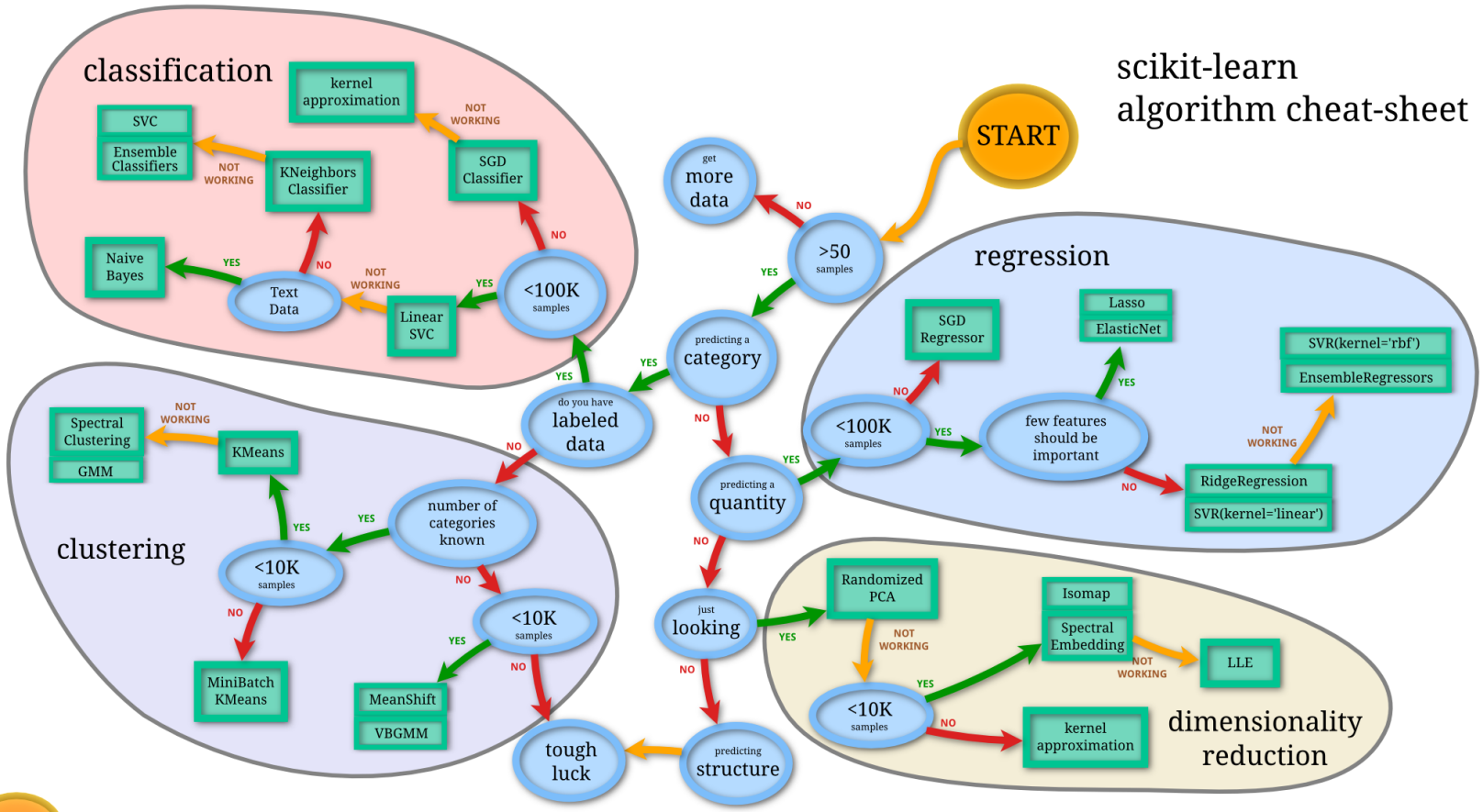
Figure 7. No free lunch theorem: Classifier 1 depicts a general purpose classifier. It performs moderately on all kinds of distributions. Classifier 2 depicts a more specialized classifier which has been tailored towards particular distributions. It behaves very good on those distributions, but worse on other distributions. According to the no free lunch theorem, the average performance of all classifiers over all distributions, that is the area under the curves, is identical for all classifiers.

Source image:
U. von Luxburg and B. Scholkopf. "Statistical Learning Theory: Models, Concepts, and Results",2011.

Choosing a model

Practical advices

- Some empirical guide can help you, e.g this sklearn flowchart: https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html



Source image: Sklearn documentation

Practical advices

- Select models with the following criterion:
  - Explainability
  - Number of examples and features
  - Non-linearity of the data
  - Computational complexity
  - Prediction speed
- Choose model with best performances always evaluated with Cross Validation
- Don’t forget to tune hyperparameters of model
  - Different strategies: grid search, random search, etc.

Model Building Strategy

1. Define a performance metric P.
2. Shortlist learning algorithms.
3. Choose a hyperparameter tuning strategy T.
4. Pick a learning algorithm A.
5. Pick a combination H of hyperparameter values using the tuning strategy T.
6. Use the training set and build the model M using the algorithm A parametrized with hyperparameter values H.
7. Use the validation set and calculate the value of the metric P for the model M.
8. Decide:
    1. If there are still untested hyperparameter values, pick another combination H of hyperparameter values using the strategy strategy T and go to step 6.
    2. Otherwise, pick a different learning algorithm A and go to step 5, or go to step 9 if there are no more learning algorithms to try.
9. Return the model for which the value of metric P is maximized.

Source content:
A. Burkov, "Machine Learning Engineering".

ML project lifecycle

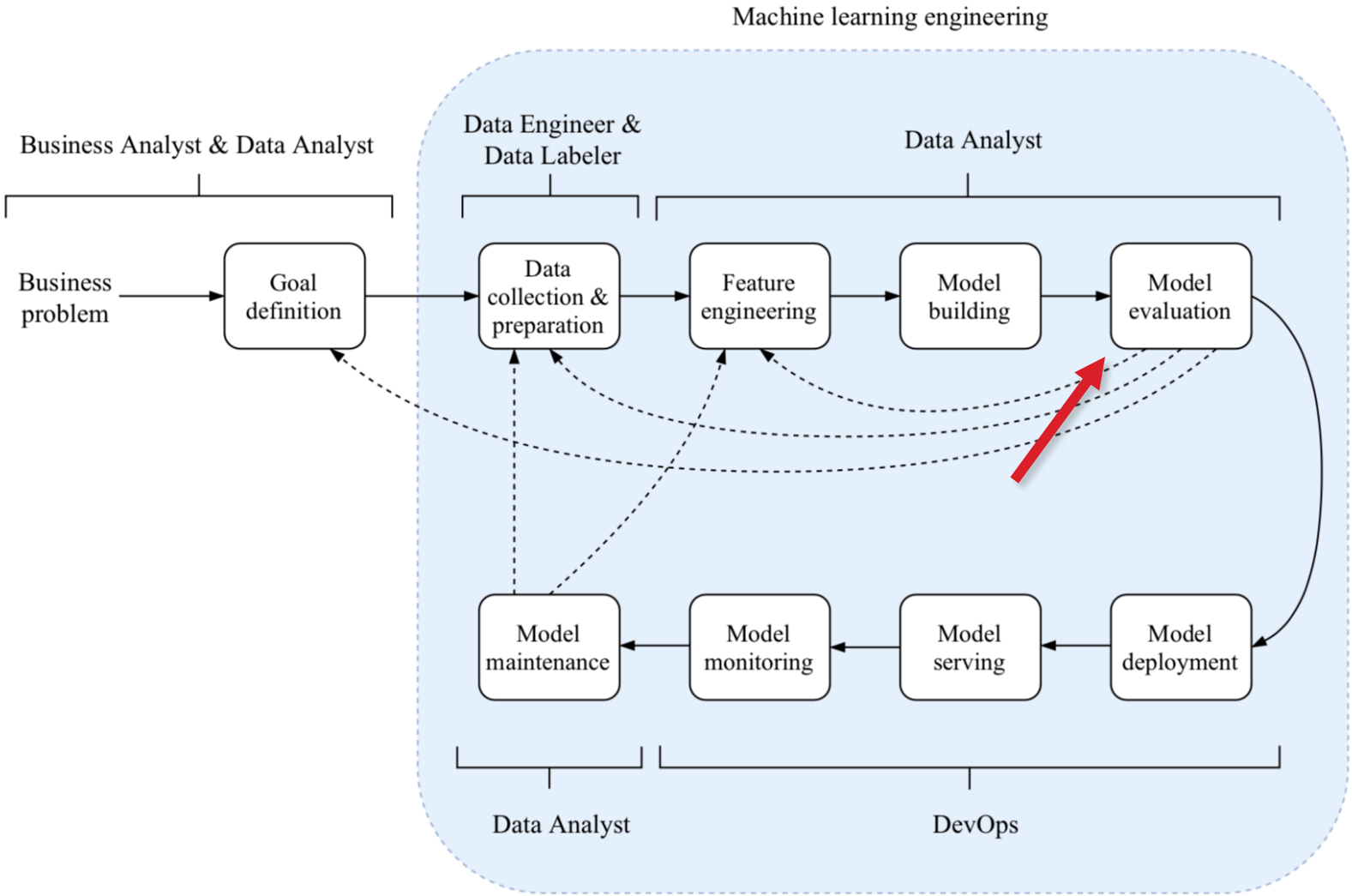


Figure 3: Machine learning project life cycle.

Source image:
A. Burkov, "Machine Learning Engineering".

Overfitting

Always use **Cross-Validation!**

- Shuffle the examples
- Split them into 3 distinct sets:
  1. Training set → for model training
  2. Validation set → for tuning hyperparamers & choosing models
  3. Test set → for independent evaluation of your final model
- Good practice: split before at the beginning of data preparation step

Extreme case:

- Your model simply memorizes its training examples but returns random labels for new examples.
- Need to evaluate on “unseen” examples.

Model evaluation

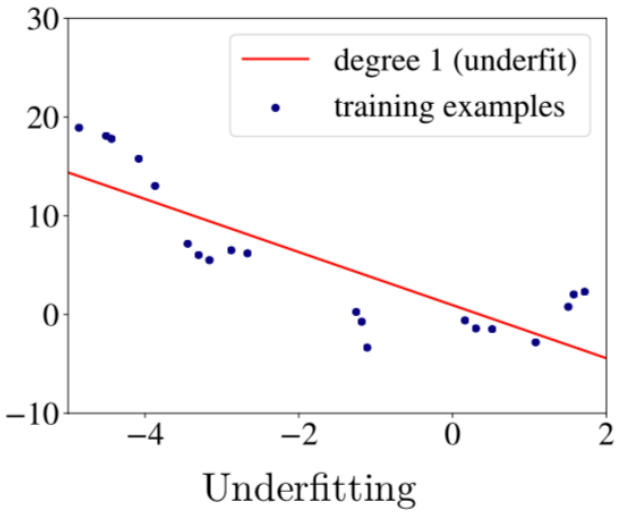
Choose an appropriate metric for your problem

- Report it computed on the test set
- Baseline:
  - performance of the simplest algorithm, usually random labelling
  - human performance

Be careful to class imbalance problem

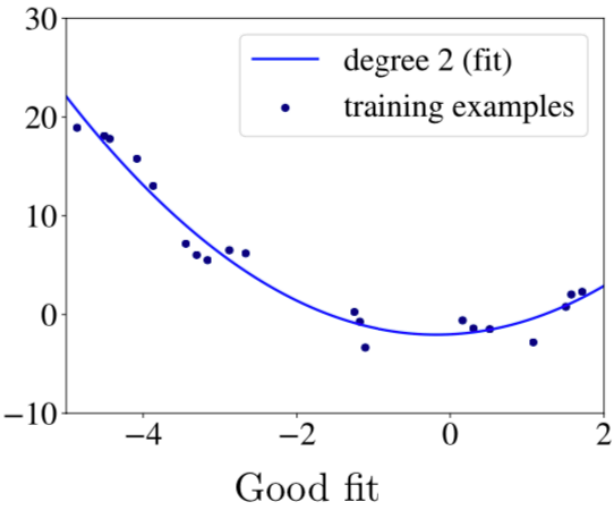
- Consider credit card fraud detection
- Probably have 1% of fraud among all transactions
- Accuracy isn't appropriate metric here: you can achieve 99% accuracy, just by classifying all transactions as genuine, which has 0 practical use
- In that case, metrics

Detecting Underfitting / Overfitting



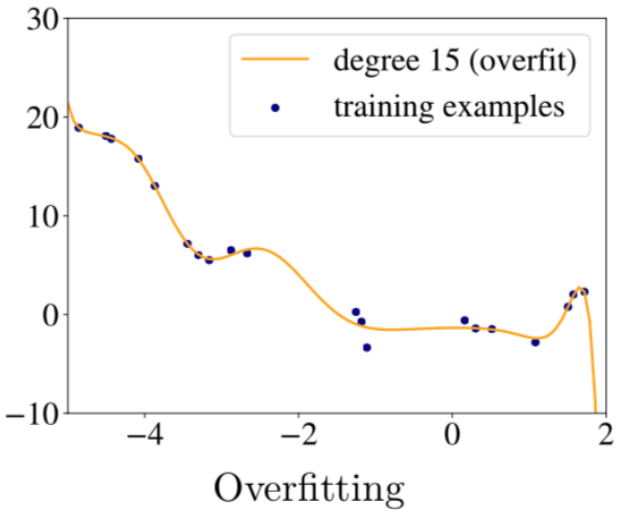
Underfitting

Oversimplification of the data



Good fit

Learning the true signal

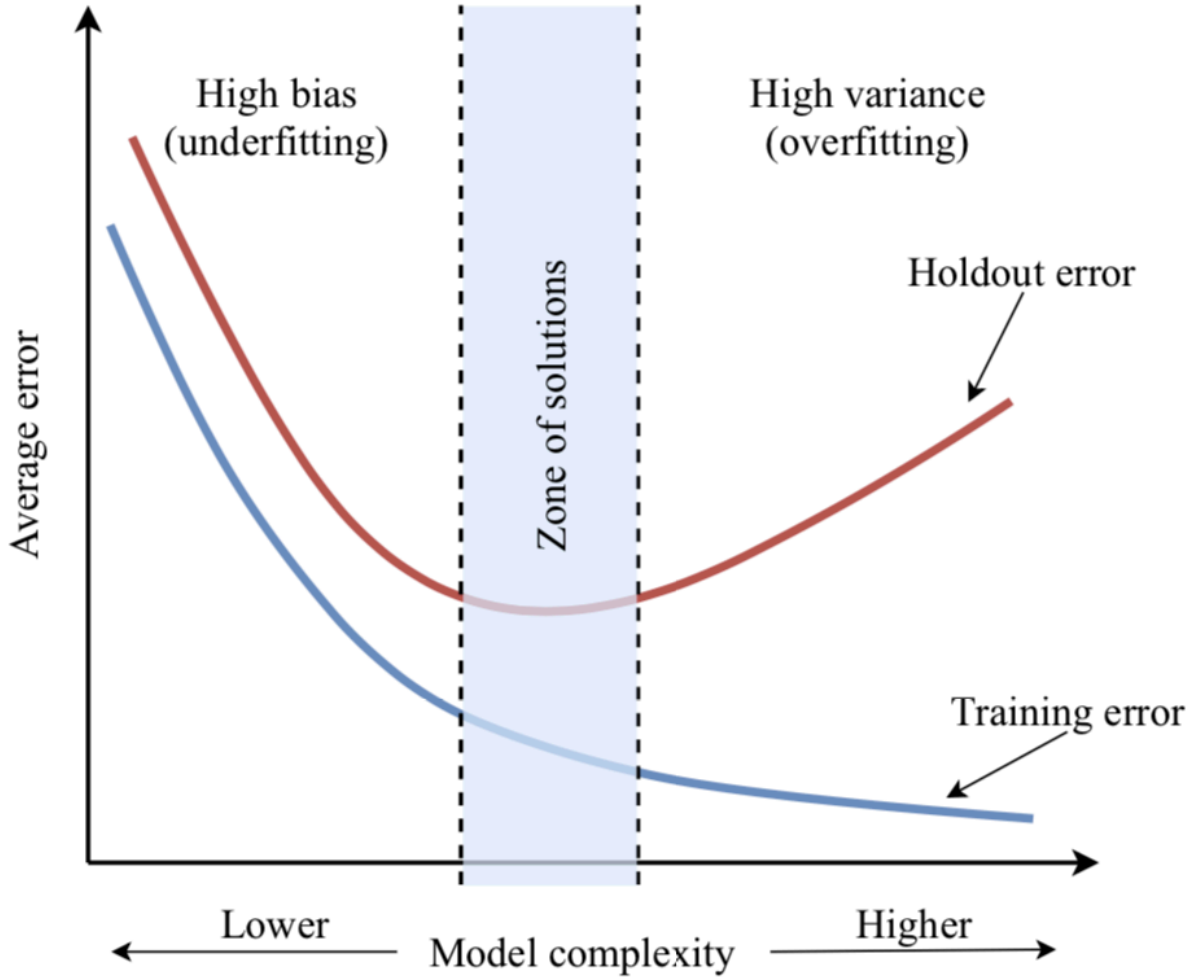


Overfitting

Fitting the noise of the data

Source image:
A. Burkov, “Machine Learning Engineering”.

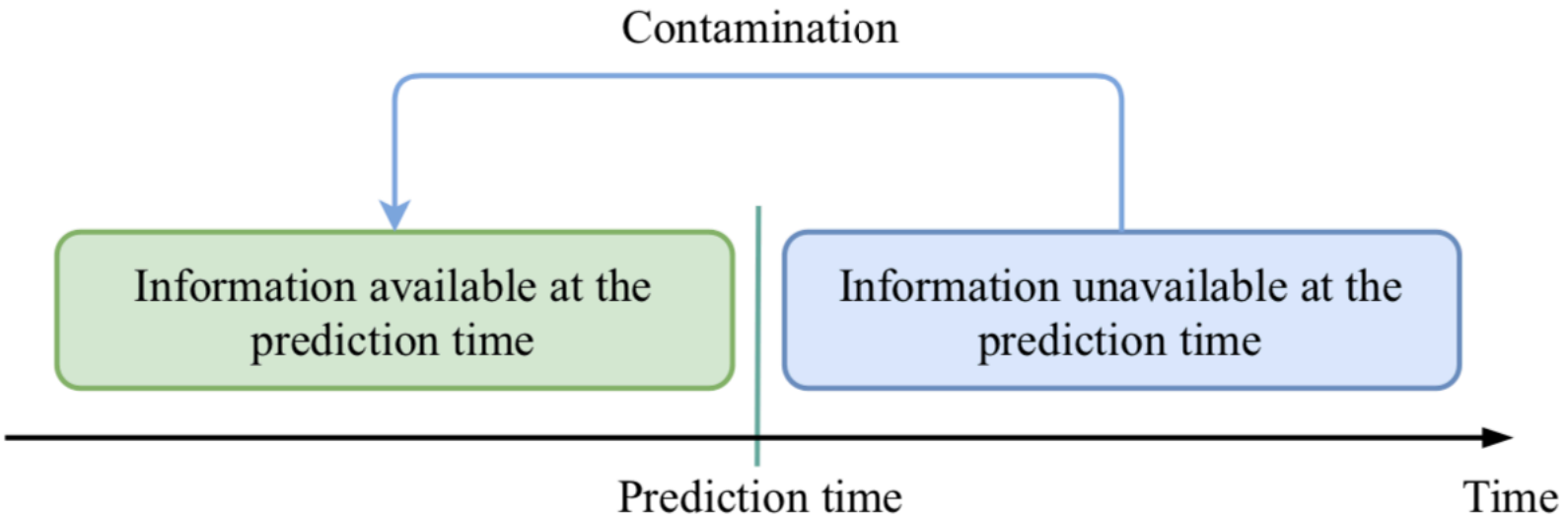
Detecting Underfitting / Overfitting



Source image:
A. Burkov, "Machine Learning Engineering".

Data Leakage

Be careful to **data leakage** if your examples are not independent



Source image:
A. Burkov, "Machine Learning Engineering".

ML project lifecycle

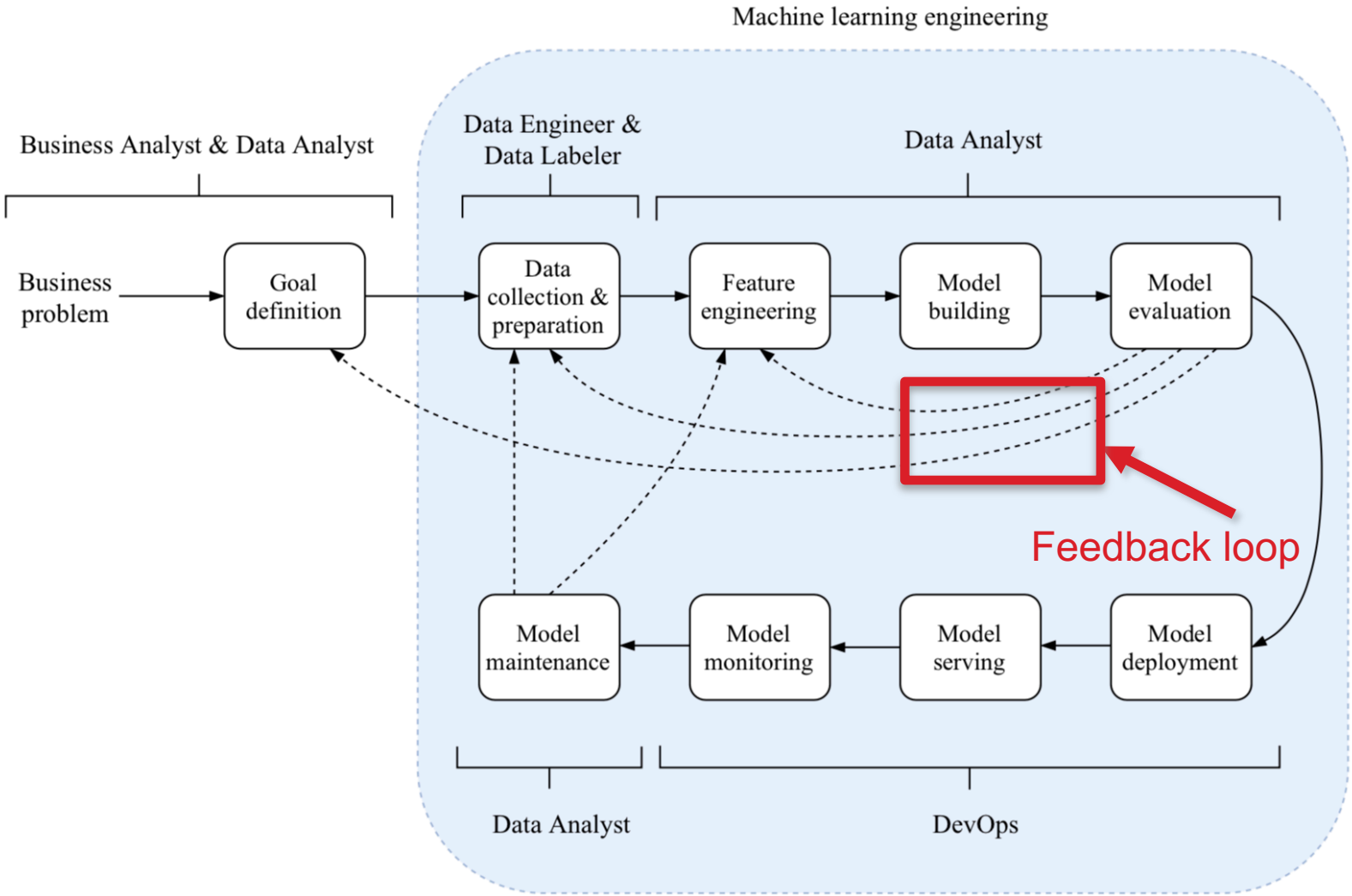


Figure 3: Machine learning project life cycle.

Source image:
A. Burkov, "Machine Learning Engineering".

ML project lifecycle

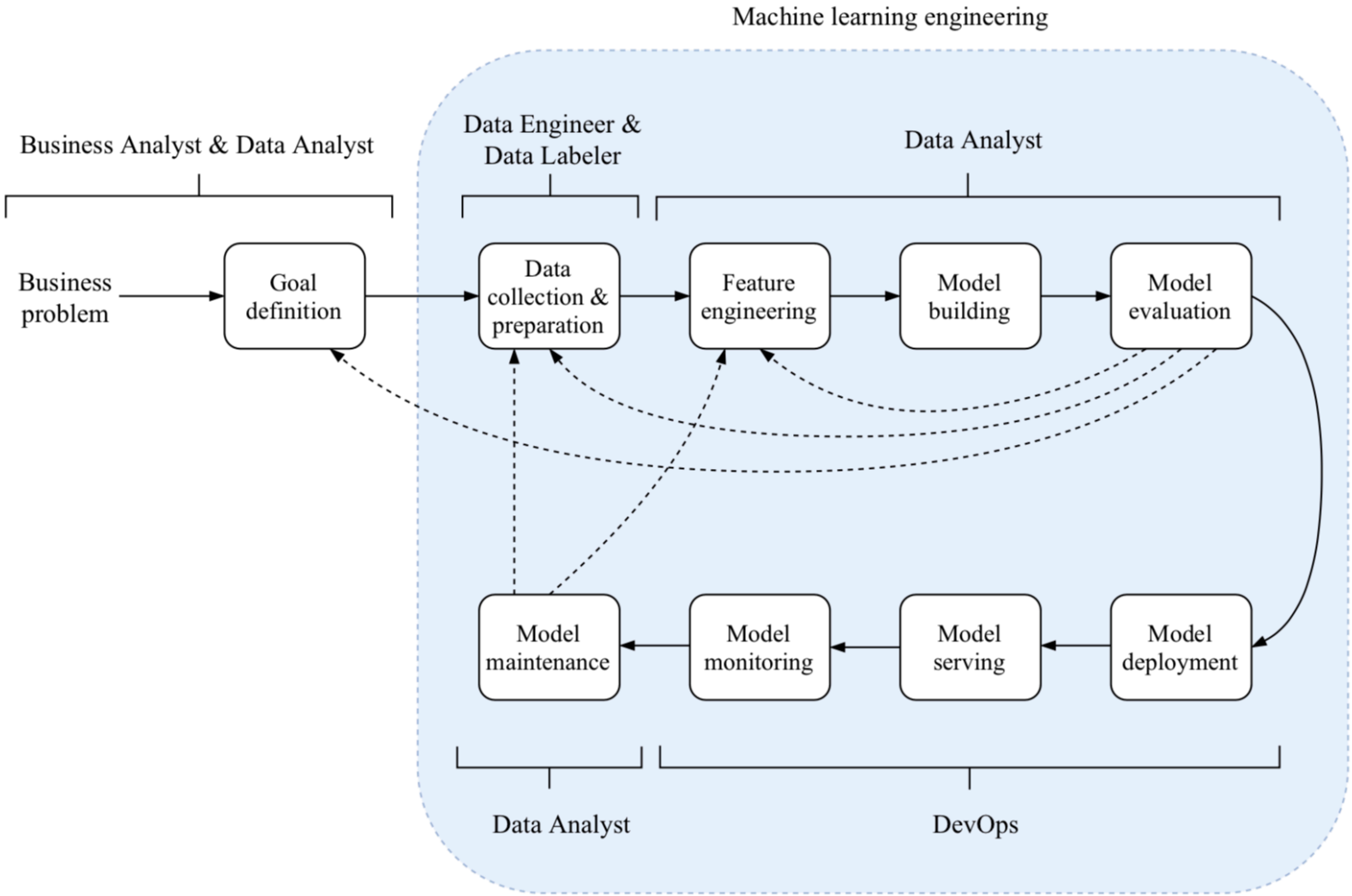


Figure 3: Machine learning project life cycle.

Source image:
A. Burkov, "Machine Learning Engineering".

More information

- List of datasets: https://en.wikipedia.org/wiki/List_of_datasets_for_machine-learning_research
- Getting started with Sklearn: https://scikit-learn.org/stable/getting_started.html
- Sklearn cheat sheet: https://www.datacamp.com/community/blog/scikit-learn-cheat-sheet
- The hundred-page Machine Learning Book: http://themlbook.com

References

- A. Burkov. “Machine Learning Engineering”. Draft available at mlebook.com
- K. Murphy. “Machine Learning: a Probabilistic Perspective”. MIT Press, 2012
- U. von Luxburg and B. Scholkopf. “Statistical Learning Theory: Models, Concepts, and Results”. In Handbook of the History of Logic, Vol. 10: Inductive Logic, volume 10, pages 651–706, 2011
- H. Wickham. “Tidy Data”. Journal of Statistical Software, 59(10), 2014