# Machine Learning – Basics

## Background

Michael Aydinbas

11. Oktober 2020

## Biographie

- Seit 2019 bei EXXETA in Karlsruhe
- MSc. in Psychologie in IT, TU Darmstadt
- BSc. in Umwelting.-Wiss. sowie in Psychologie in IT, TU Darmstadt

## Beratungskompetenz

- Rollen:  IT Consultant, Data Scientist
- Consulting Expertise: Forschung und Entwicklung, Automotive, Strategieberatung
- Technische Expertise: Data Science, Large-Scale Data Processing, Probabilistische Modelle, Web-Technologien

## Sprachen

- Englisch
- Deutsch

# Michael Aydinbas

## IT Consultant, Data Scientist

Data Science, Data Engineering, Big Data

## Auszug relevante Projekterfahrung

### Forschung

- Clustern von hochdimensionalen Zeitreihen mit Unsupervised-, Semi-Supervised- and Active-Learning-Verfahren
- Anomaliedetektion und Qualitätssteigerung von Sensornetzwerkdaten
- Thesis: *Realizing Cognitive User Models for Adaptive Serious Games,* TU Darmstadt

### Data Engineering & Data Science

- Anforderungsanalyse und Entwicklung eines Sequencing-Frameworks zur automatischen Erkennung von Fahrszenen
- Entwicklung einer Python Toolbox für Datenexploration/Datenanalyse auf Basis von Apache Spark
- Entwicklung eines konfigurierbaren, robusten, batch-basierten Messdatenkonverters
- Konzeptionierung und Implementierung eines Frameworks zur Erkennung und Korrektur von fehlerhaften Messsignalen sowie zur Erkennung des gefahrenen Prüfzyklus‘
- Einsatz von Apache Spark als Analyse-Engine für die Integration sowie Verarbeitung von Messdaten
- Entwicklung und Implementierung eines Prognosealgorithmus‘ zur Vorhersage von Prozesszuständen
- Erstellung von Tutorials, interaktiver Notebooks und Visualisierungs-Dashboards

### App Development

- Konzeption, Entwicklung und Deployment mehrerer Prototypen zur Analyse und Visualisierung von Messdaten
- Implementierung von Unit-Tests mit Python

# Webinar’s Agenda

## 1. Machine Learning Basics

- Definition
- Relation to other fields
- Little terminology

## 2. The ML Workflow

- Get data
- Prepare data
- Model selection and training
- Model evaluation
- Model improvement

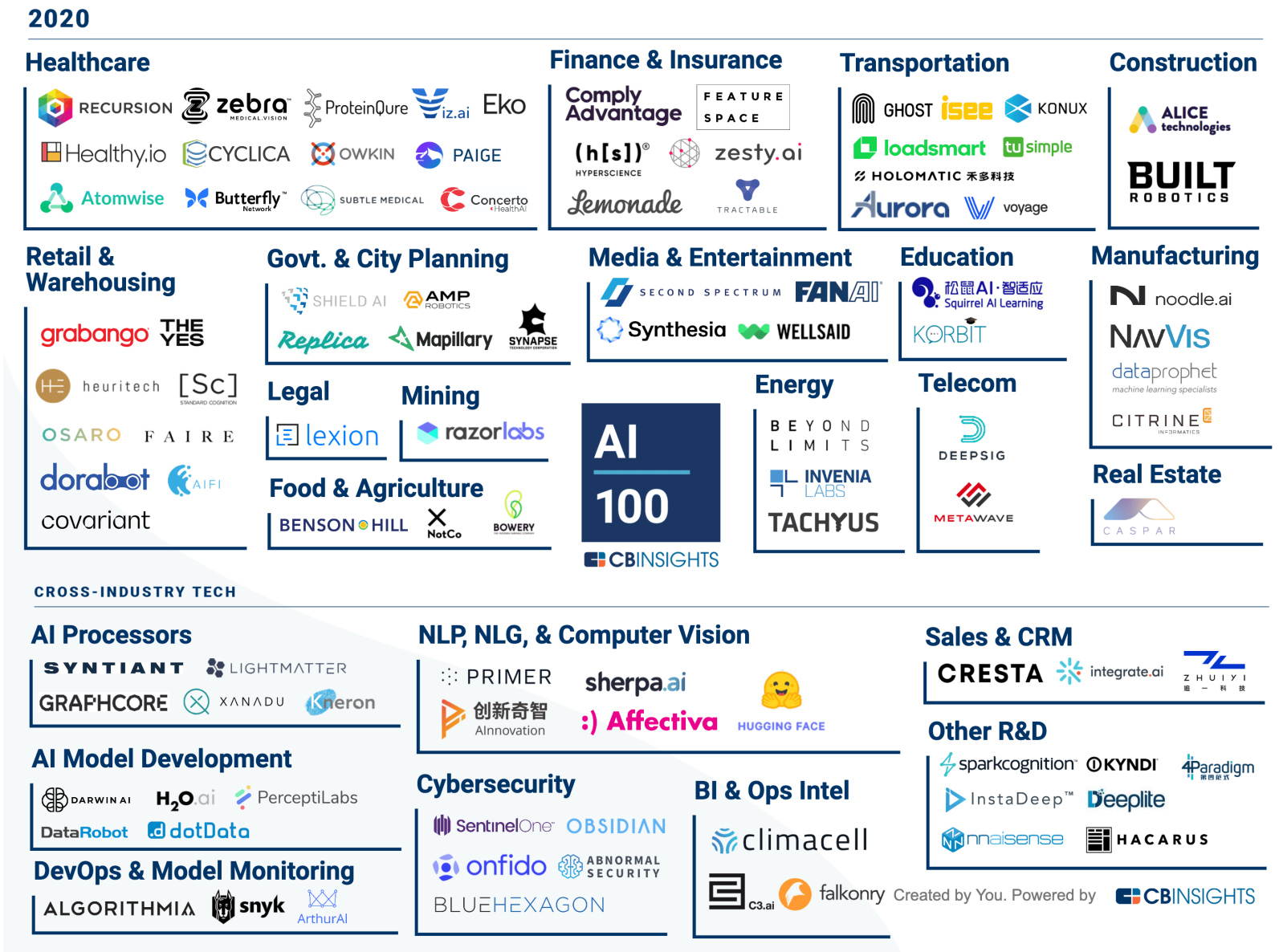
## 3. Resources

# Machine Learning
# What comes to your mind?

*“Every serious technology company now has an Artificial Intelligence team in place. These companies are investing millions into intelligent systems for situation assessment, prediction analysis, learning-based recognition systems, conversational interfaces, and recommendation engines. Companies such as Google, Facebook, and Amazon aren’t just employing AI, but have made it a central part of their core intellectual property.” - Kristian J. Hammond*

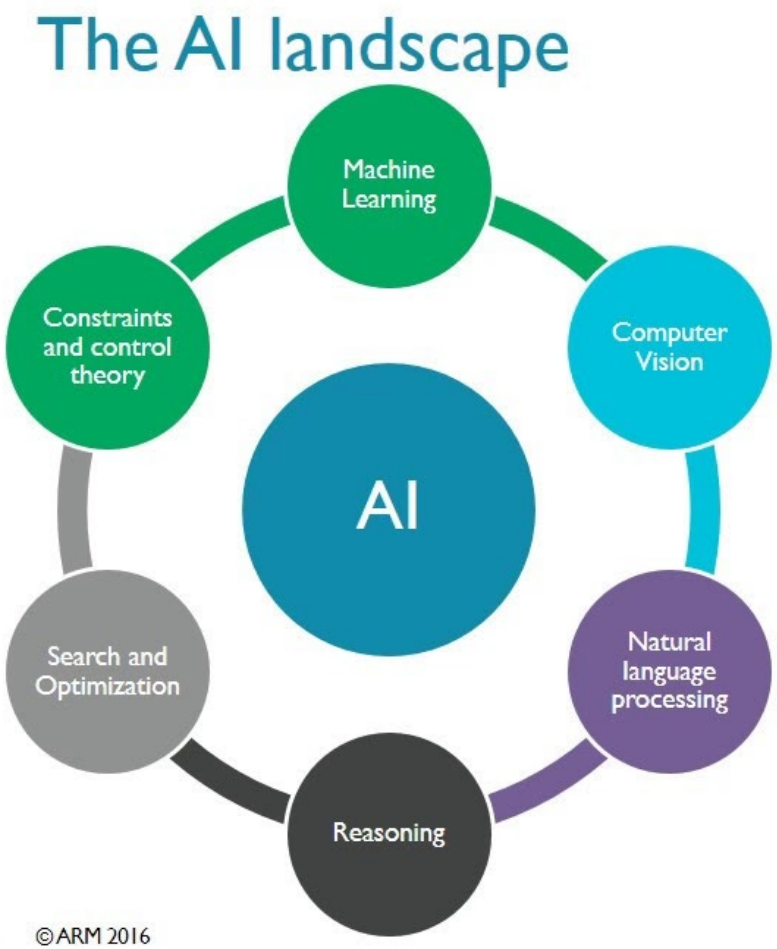
# Startups and AI

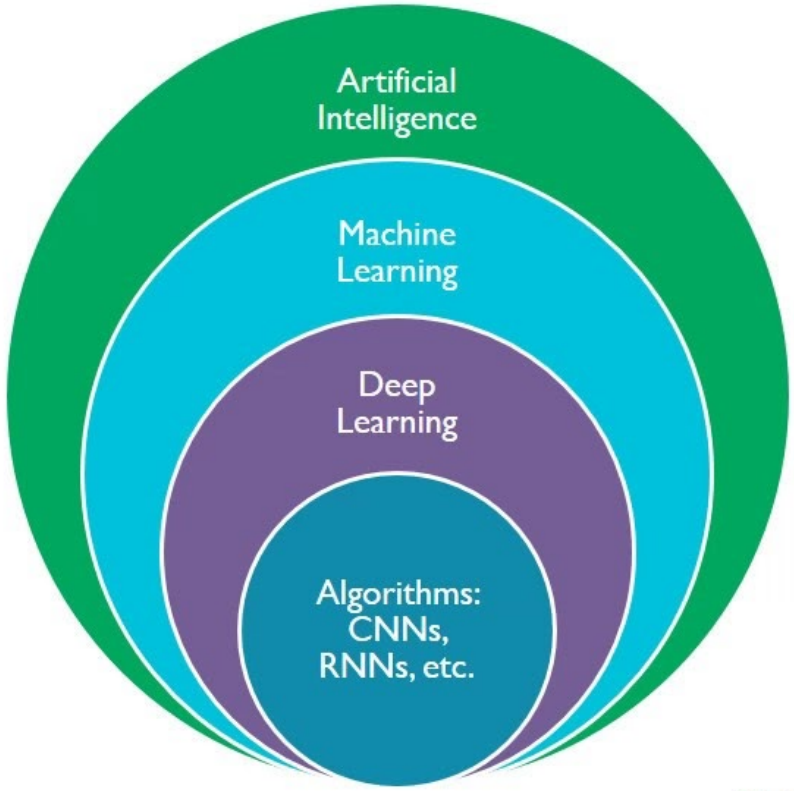
**The AI 100 is CB Insights' annual ranking of the 100 most promising AI startups in the world.**




https://www.cbinsights.com/research/artificial-intelligence-top-startups/

# Artificial Intelligence and Machine Learning

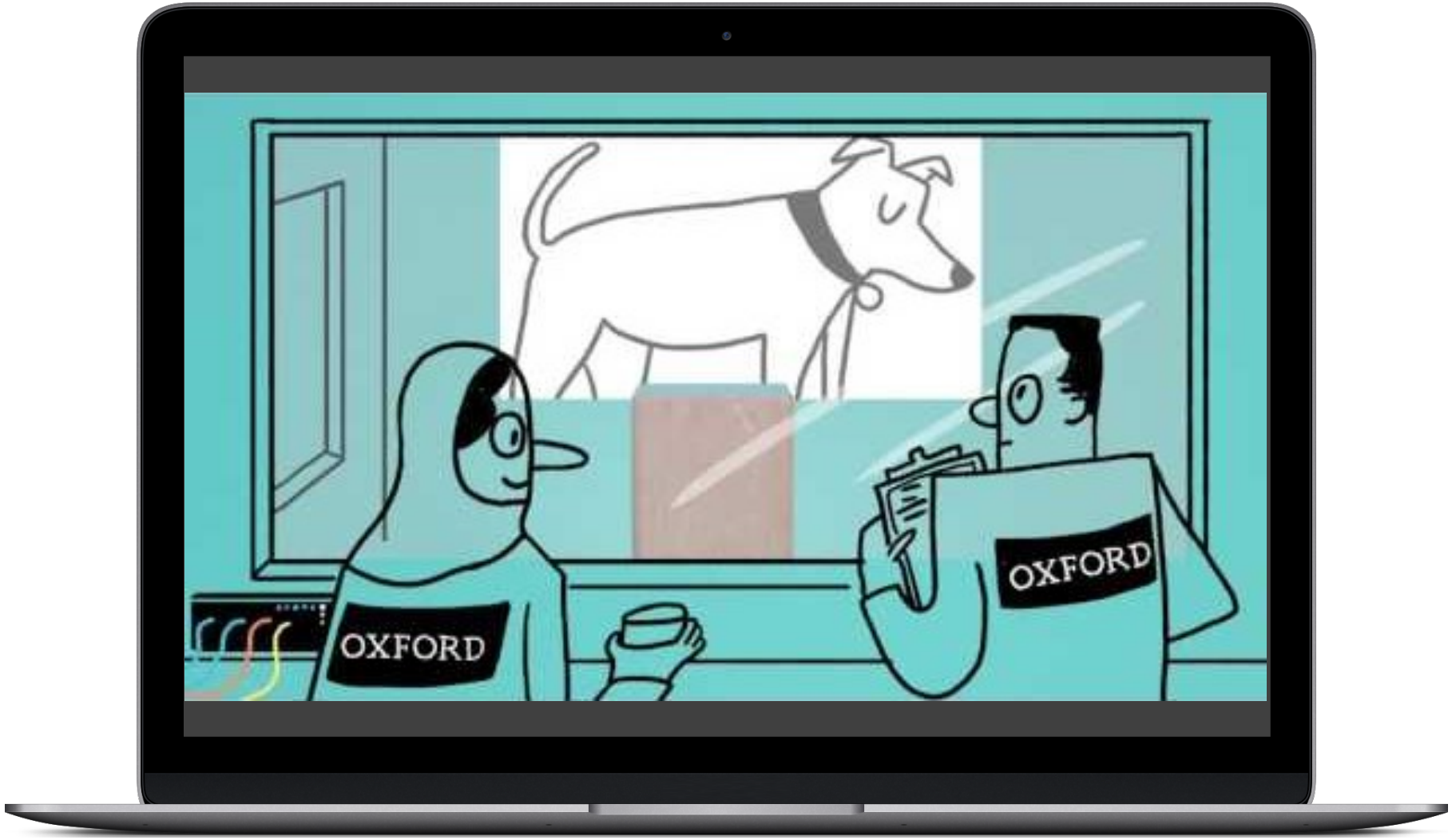





ARM

# What does “learning” mean?



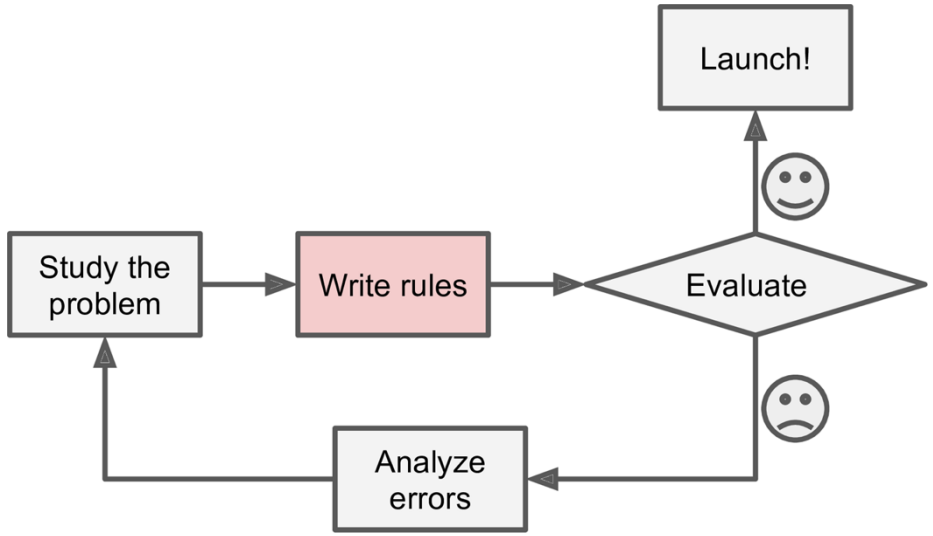

https://youtu.be/f_uwKZIAeM0

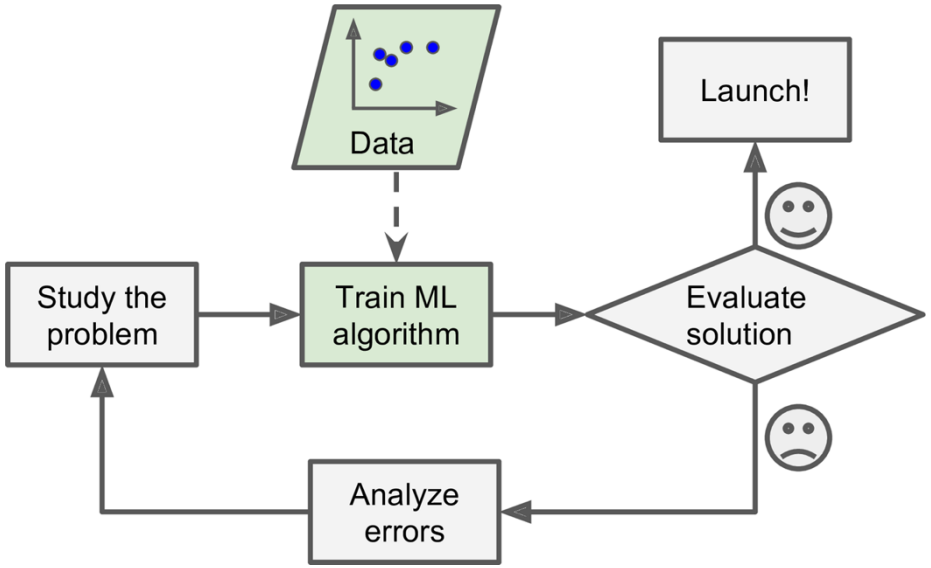
# What does “learning” mean?

1. “Machine learning is the science of getting computers to act without being explicitly programmed.” – Andrew Ng, Stanford University (based on Arthur Samuel, *1959)*
2. “Machine learning algorithms can figure out how to perform important tasks by generalizing from examples.” – University of Washington
3. “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

# Why Use Machine Learning?

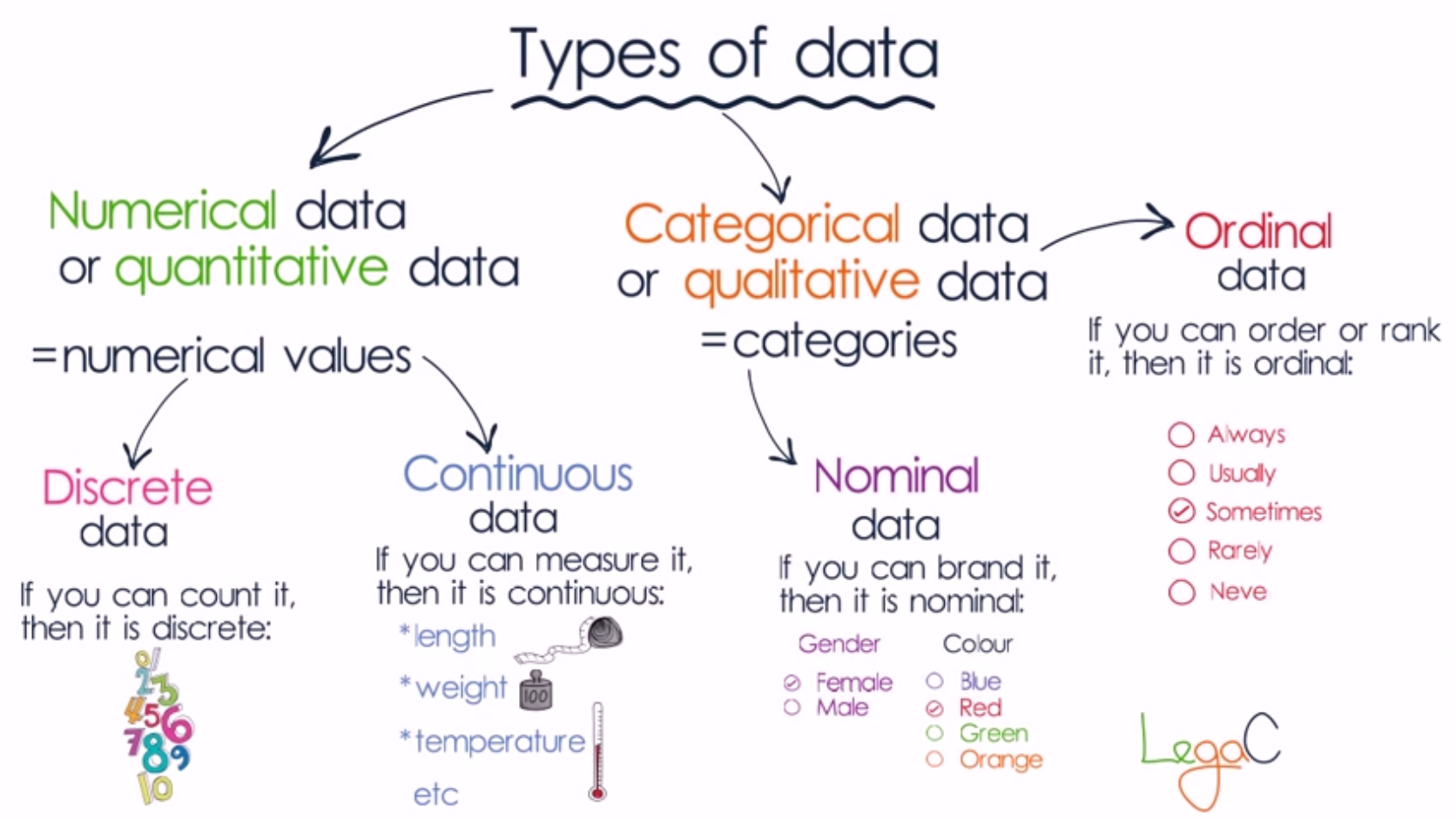
## How would you implement a spam filter?



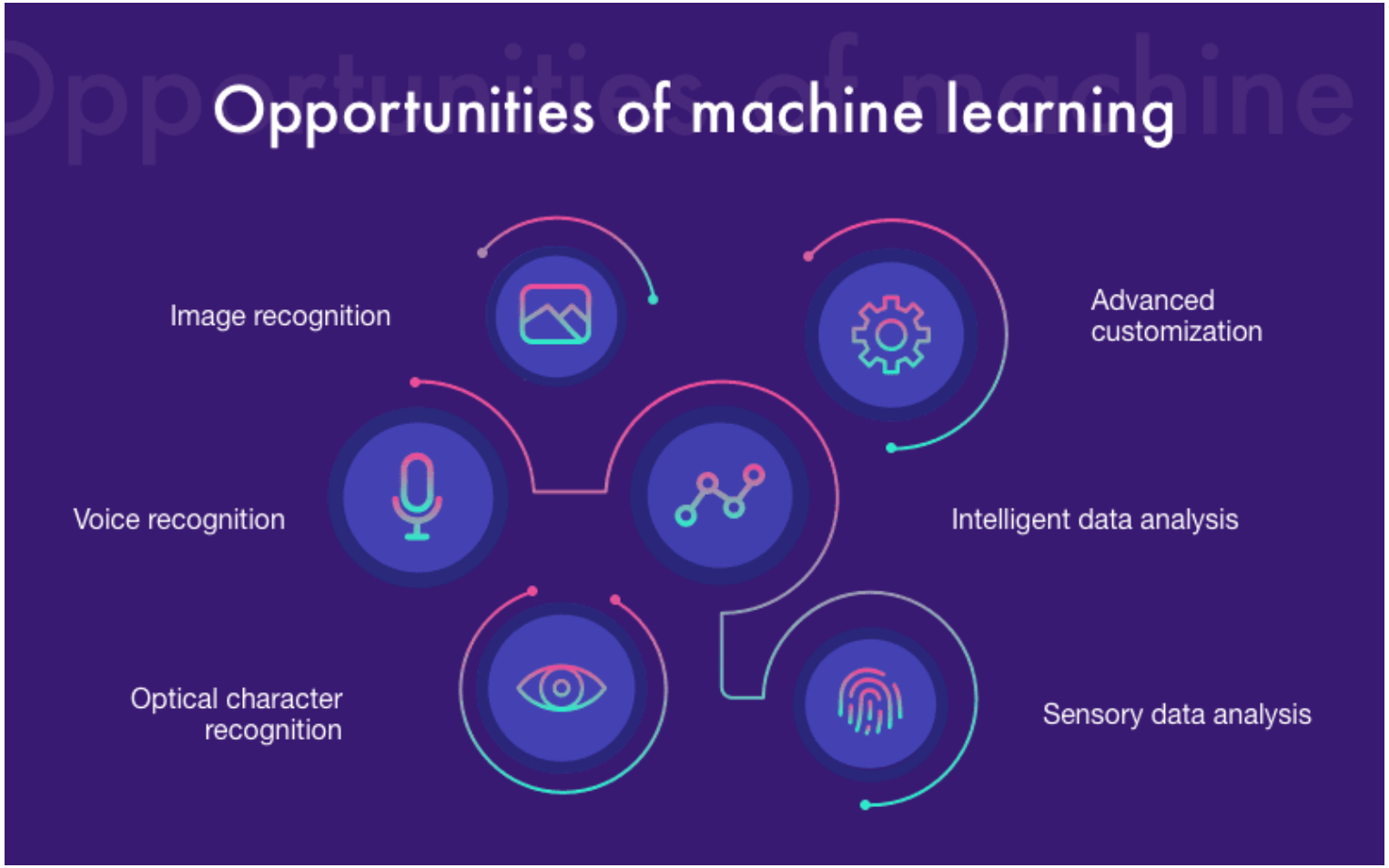




Aurélien Géron (2017), Hands-On Machine Learning with Scikit-Learn and TensorFlow, Publisher: O'Reilly Media, Inc., ISBN: 9781491962299

# What does our “data” look like?

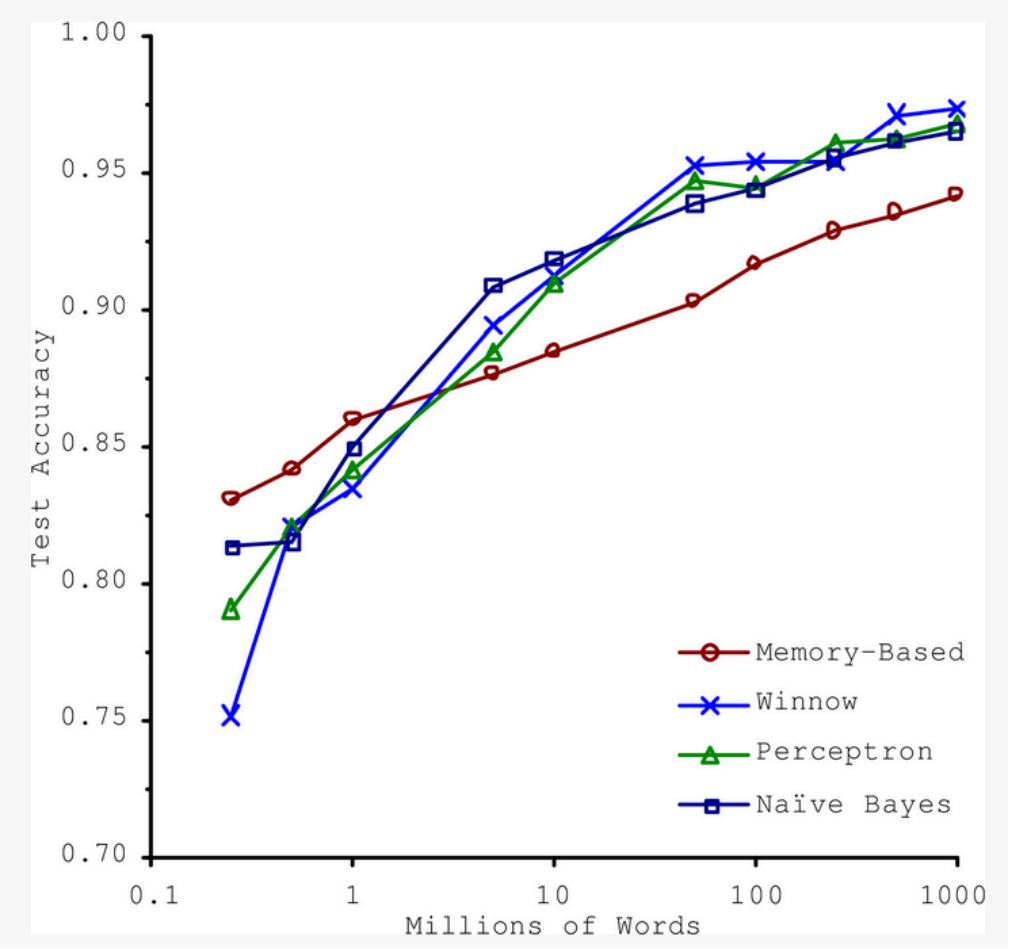


# What does our “data” look like?



# Main challenges of Machine Learning

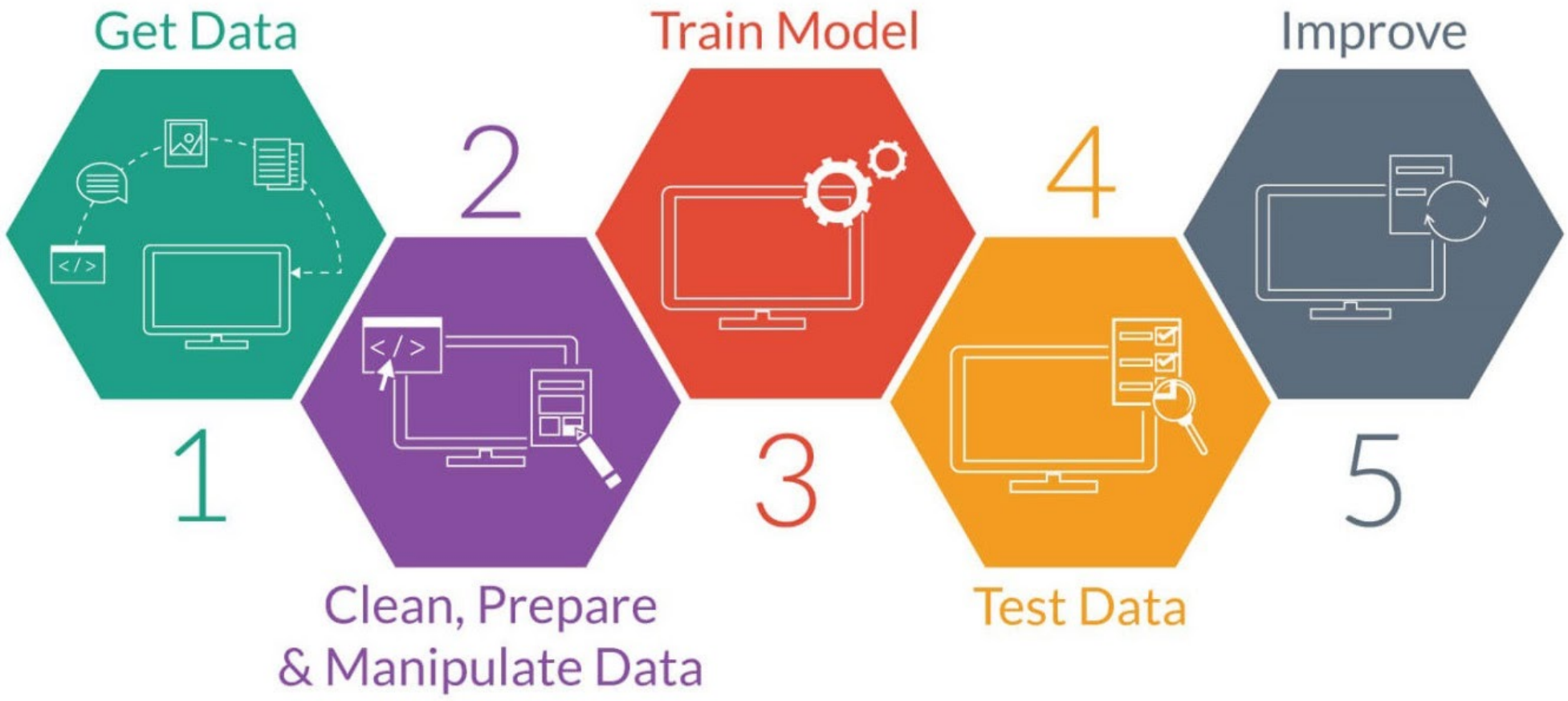
- **Insufficient Quantity of Training Data**
- **Nonrepresentative Training Data**
- **Poor-Quality Data**
- **Irrelevant Features**
- **Overfitting the Training Data**
- **Underfitting the Training Data**




Michele Banko and Eric Brill. 2001. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics* (*ACL '01*). Association for Computational Linguistics, USA, 26–33. DOI:https://doi.org/10.3115/1073012.1073017

# Machine Learning Workflow

# The Machine Learning workflow - simplified



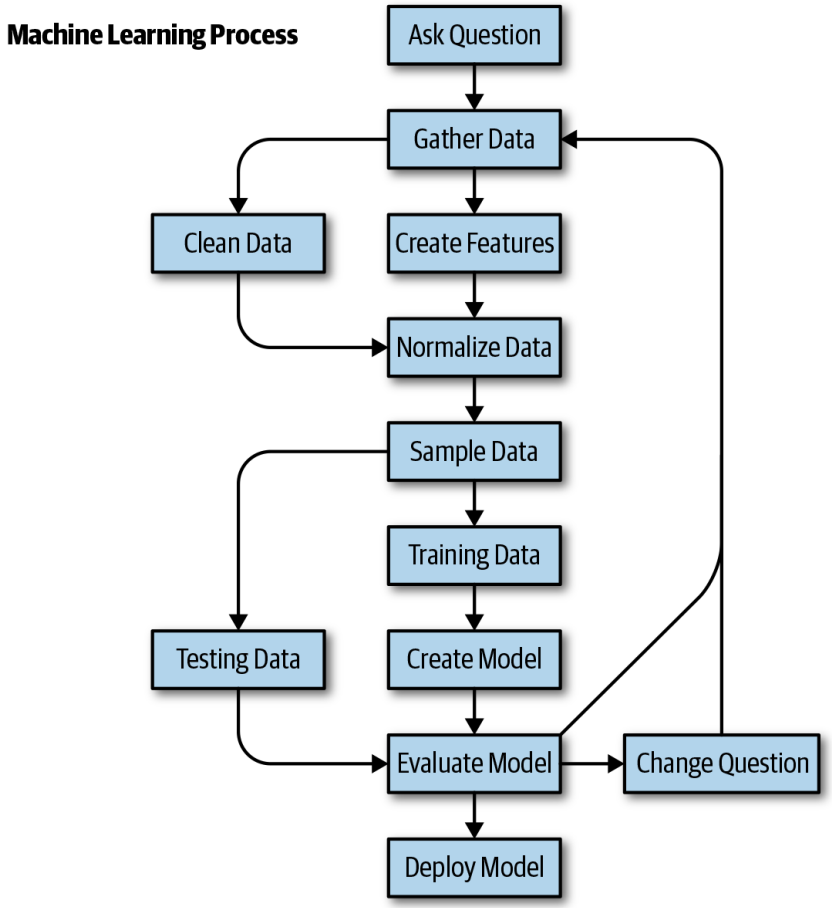

https://www.freecodecamp.org/news/every-single-machine-learning-course-on-the-internet-ranked-by-your-reviews-3c4a7b8026c0/

# The bigger picture – CRISP-DM


https://en.wikipedia.org/wiki/Cross-industry_standard_process_for_data_mining

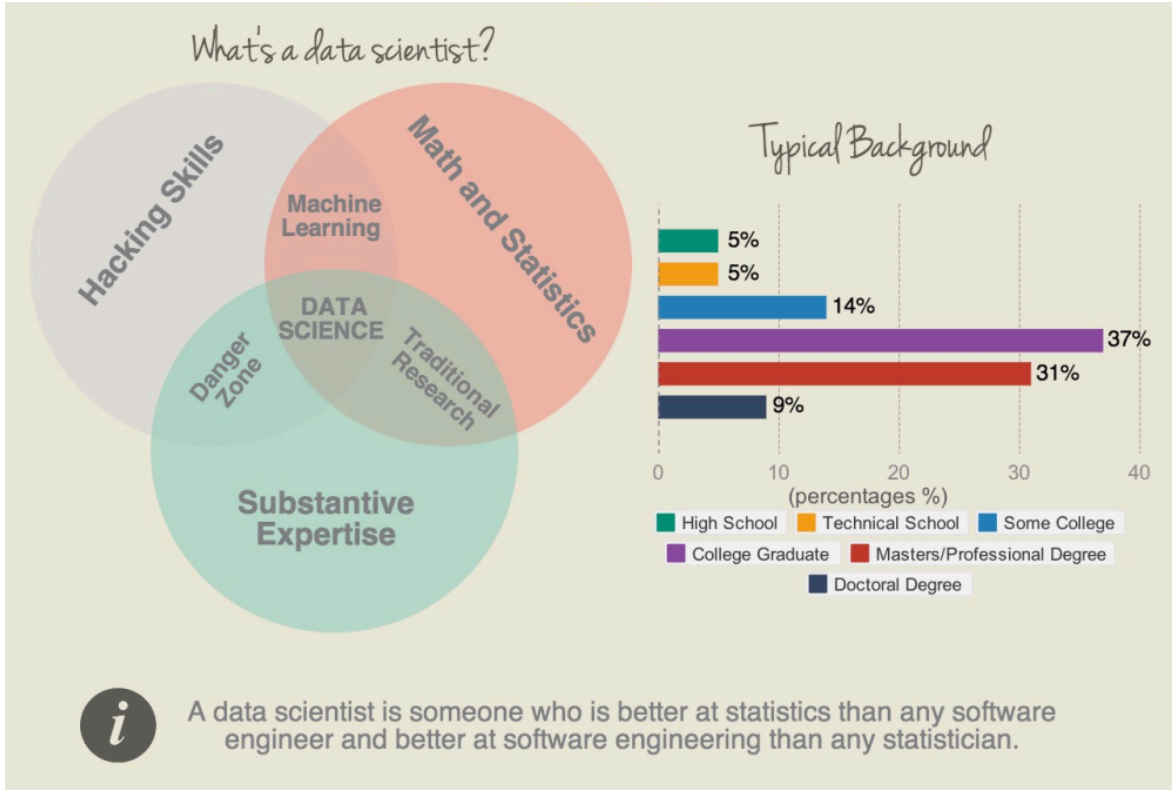
# A more detailed workflow

## Machine Learning Pocket Reference by Matt Harrison



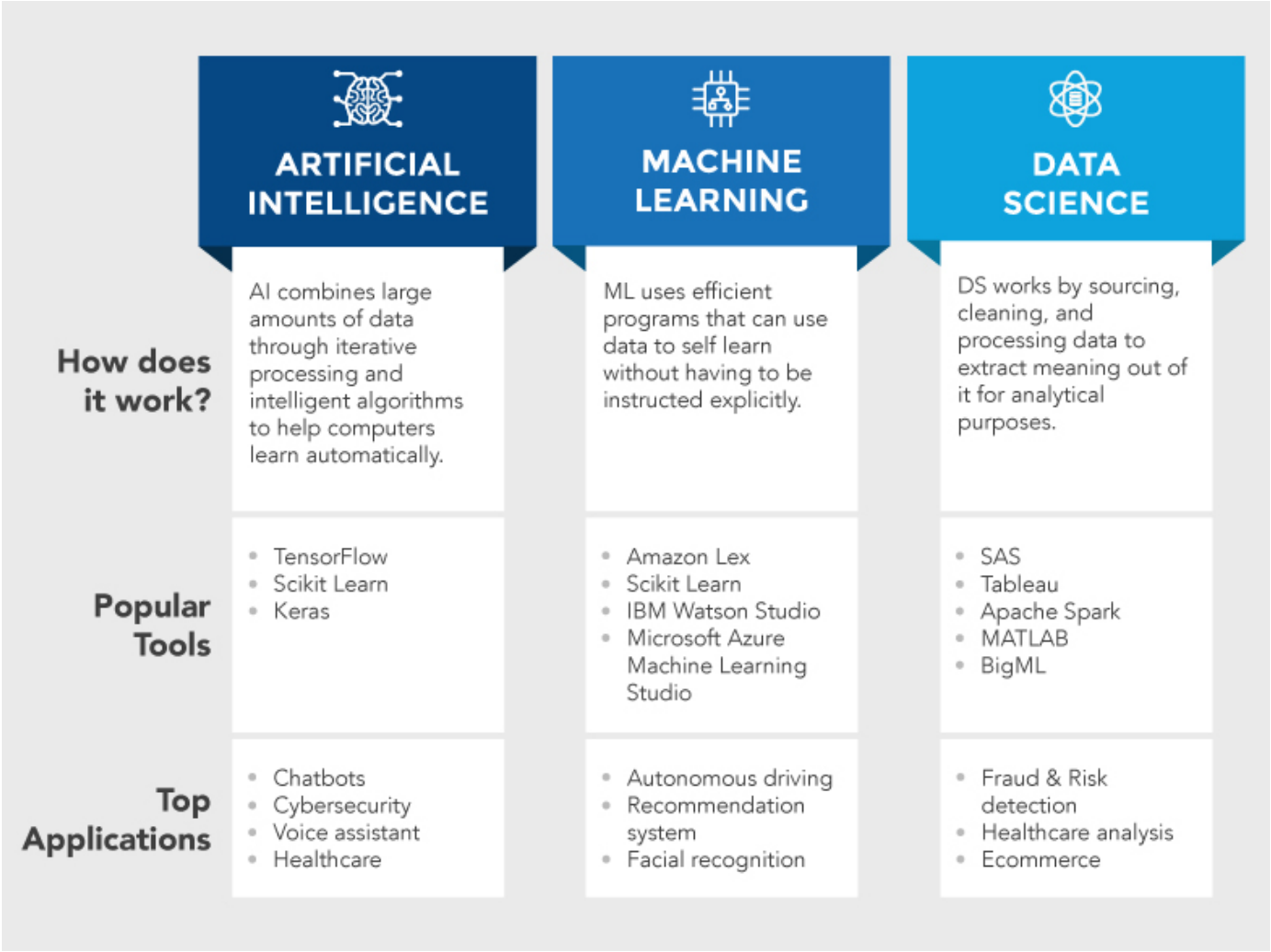
# The bigger picture – skillset of a data scientist


https://www.simplilearn.com/data-science-vs-data-analytics-vs-machine-learning-article



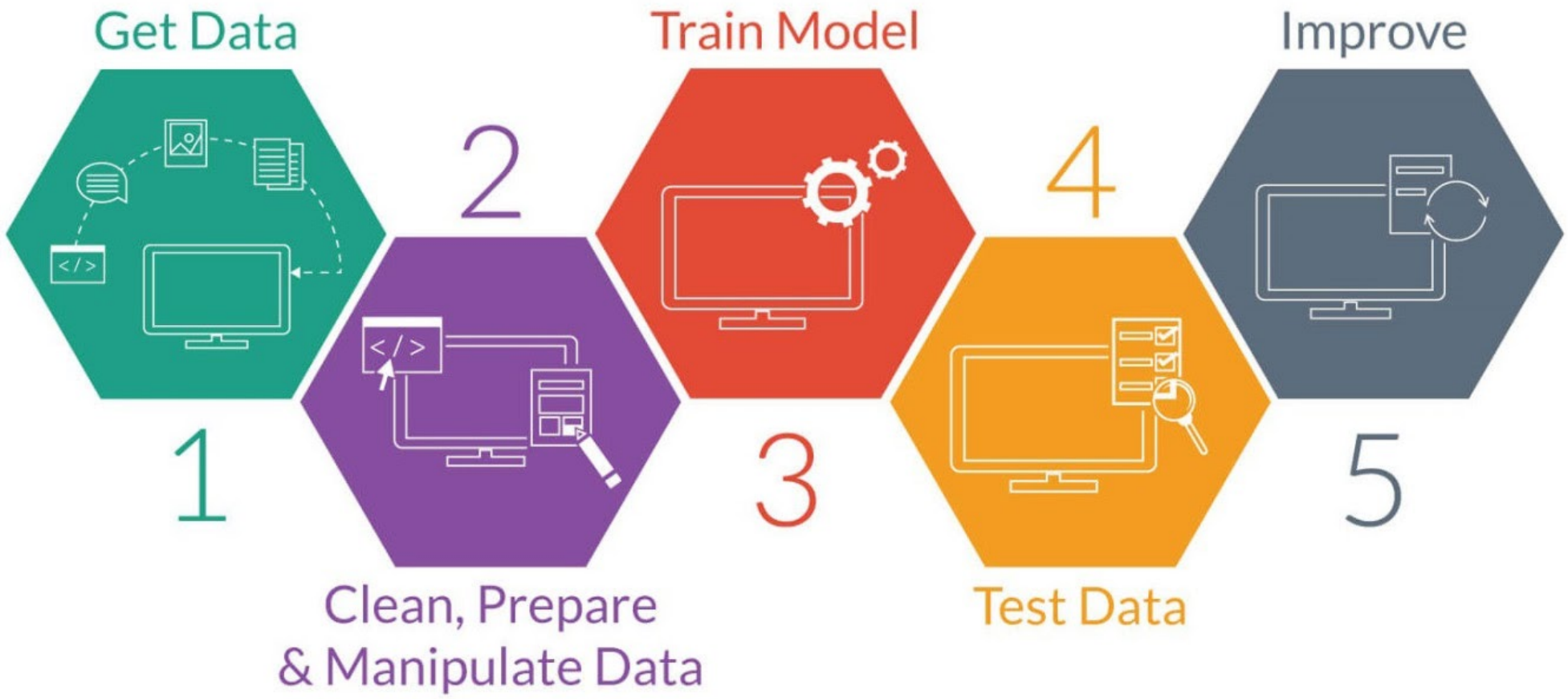

https://machinelearningmastery.com/become-data-scientist/

# The bigger picture – AI vs ML vs DS




https://www.mygreatlearning.com/blog/difference-data-science-machine-learning-ai/

# The Machine Learning workflow - simplified



# The Machine Learning workflow - example

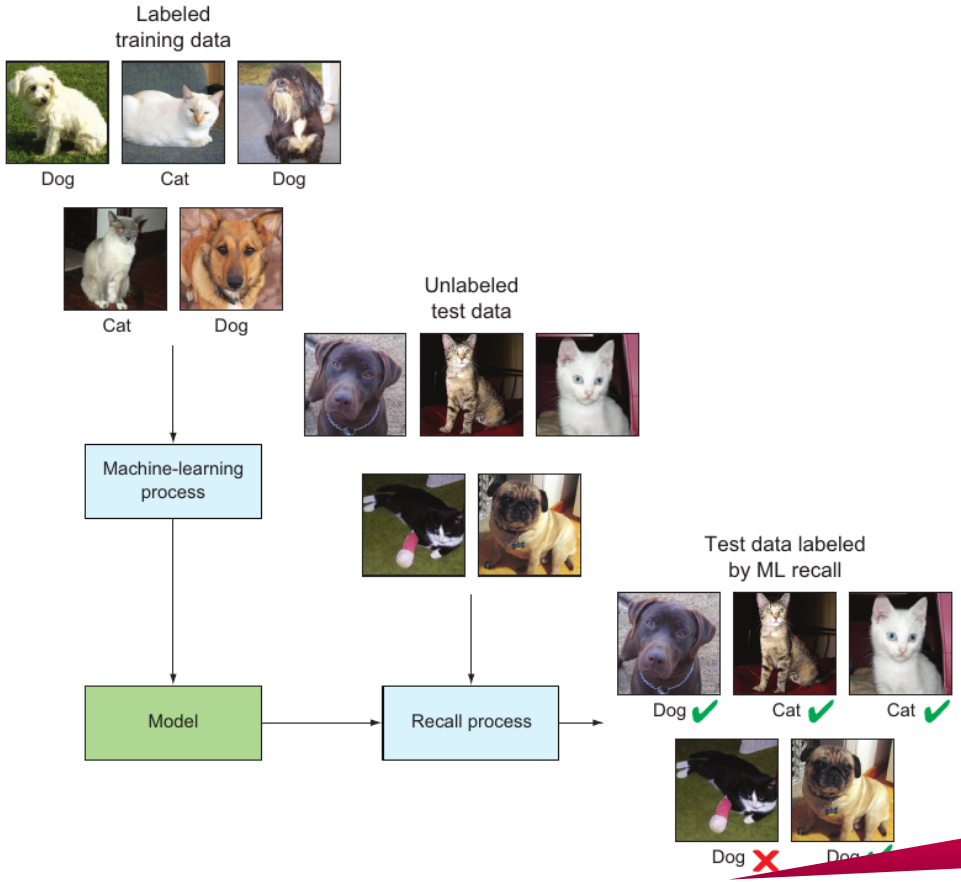



**Figure 1.1 Machine-learning process for the cats and dogs competition**

Any error is a valuable information by itself!

Henrik Brink, Joseph W. Richards, and Mark Fetherolf , Real-World Machine Learning, Manning, September 2016, ISBN 9781617291920

# Why you should care about your data set

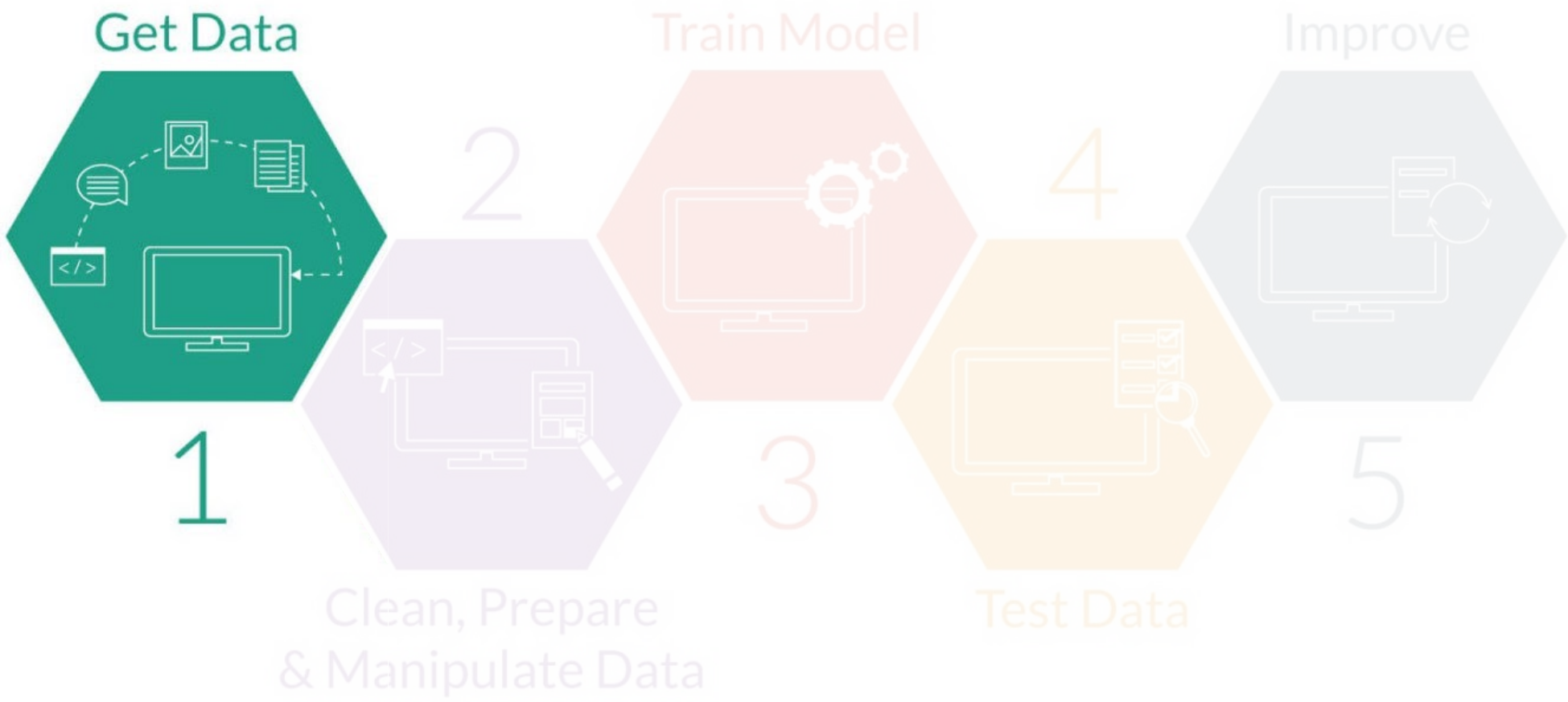
Detecting tanks

US Army wanted to use Neural Networks to detect tanks hidden behind trees

However...something went wrong

https://neil.fraser.name/writing/tank/

# Get Data


Get Data
1
Clean, Prepare
& Manipulate Data
2
Train Model
3
Test Data
4
Improve
5

# Get Data

## Machine Learning Terminology

Header

Rows/ Examples/ Instances

| | pclass | survived | name | sex | age | sibsp | parch | ticket | fare | cabin | embarked | boat | body | home.dest |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 1 | Allen, Miss. Elisabeth Walton | female | 29.0000 | 0 | 0 | 24160 | 211.3375 | B5 | S | 2 | NaN | St Louis, MO |
| **1** | 1 | 1 | Allison, Master. Hudson Trevor | male | 0.9167 | 1 | 2 | 113781 | 151.5500 | C22 C26 | S | 11 | NaN | Montreal, PQ / Chesterville, ON |
| **2** | 1 | 0 | Allison, Miss. Helen Loraine | female | 2.0000 | 1 | 2 | 113781 | 151.5500 | C22 C26 | S | NaN | NaN | Montreal, PQ / Chesterville, ON |
| **3** | 1 | 0 | Allison, Mr. Hudson Joshua Creighton | male | 30.0000 | 1 | 2 | 113781 | 151.5500 | C22 C26 | S | NaN | 135.0 | Montreal, PQ / Chesterville, ON |
| **4** | 1 | 0 | Allison, Mrs. Hudson J C (Bessie Waldo Daniels) | female | 25.0000 | 1 | 2 | 113781 | 151.5500 | C22 C26 | S | NaN | NaN | Montreal, PQ / Chesterville, ON |
| **5** | 1 | 1 | Anderson, Mr. Harry | male | 48.0000 | 0 | 0 | 19952 | 26.5500 | E12 | S | 3 | NaN | New York, NY |
| **6** | 1 | 1 | Andrews, Miss. Kornelia Theodosia | female | 63.0000 | 1 | 0 | 13502 | 77.9583 | D7 | S | 10 | NaN | Hudson, NY |
| **7** | 1 | 0 | Andrews, Mr. Thomas Jr | male | 39.0000 | 0 | 0 | 112050 | 0.0000 | A36 | S | NaN | NaN | Belfast, NI |
| **8** | 1 | 1 | Appleton, Mrs. Edward Dale (Charlotte Lamson) | female | 53.0000 | 2 | 0 | 11769 | 51.4792 | C101 | S | D | NaN | Bayside, Queens, NY |
| **9** | 1 | 0 | Artagaveytia, Mr. Ramon | male | 71.0000 | 0 | 0 | PC 17609 | 49.5042 | NaN | C | NaN | 22.0 | Montevideo, Uruguay |

Index

Features/Attributes/Dimensions

## How to make sense of your data

You can visualize your data

- mosaic plot
- ...

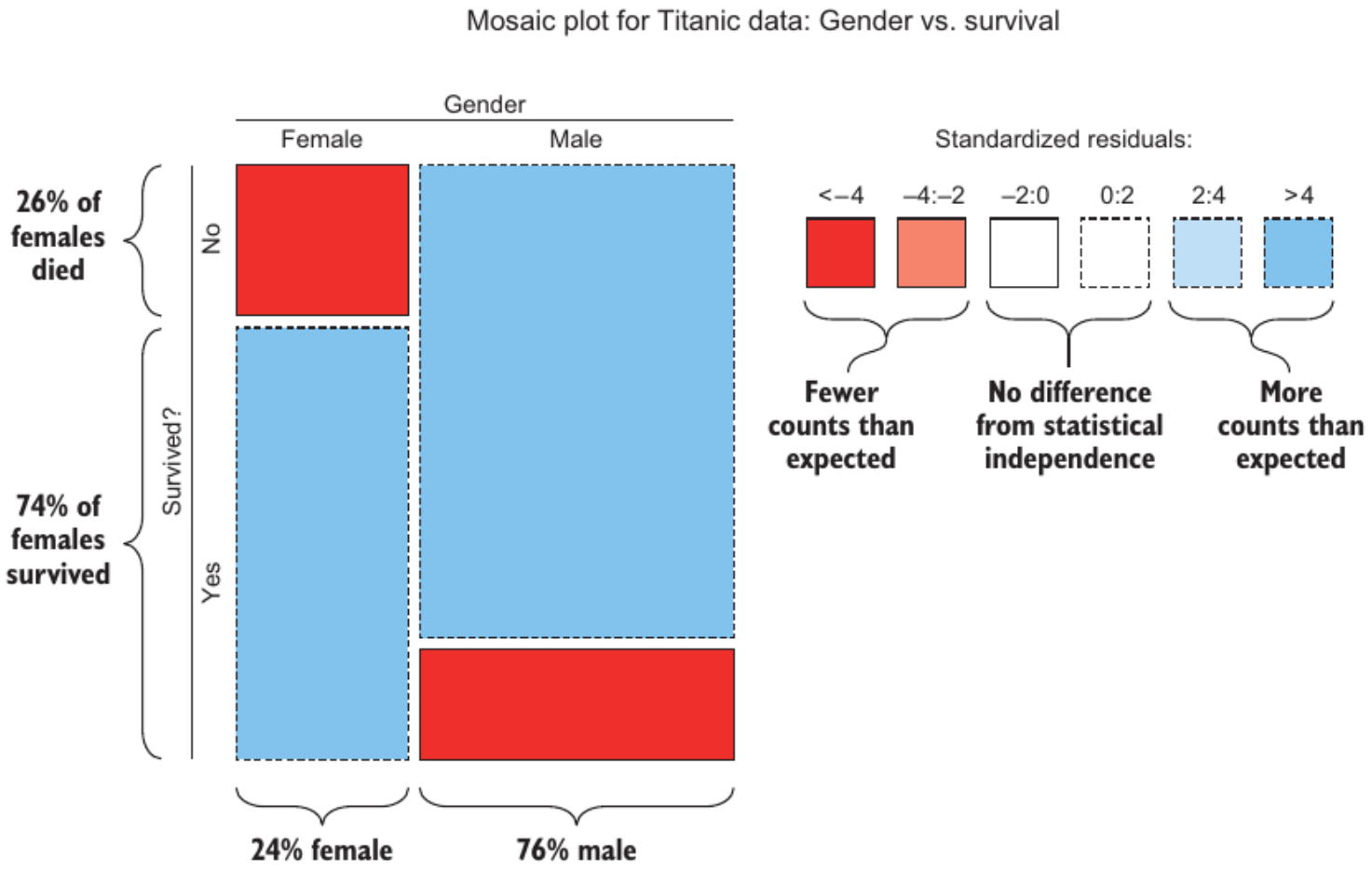



**Figure 2.12** Mosaic plot showing the relationship between gender and survival on the Titanic. The visualization shows that a much higher proportion of females (and much smaller proportion of males) survived than would have been expected if survival were independent of gender. "Women and children first."

## How to make sense of your data

You can visualize your data

- Mosaic plot
- Box plot
- ...

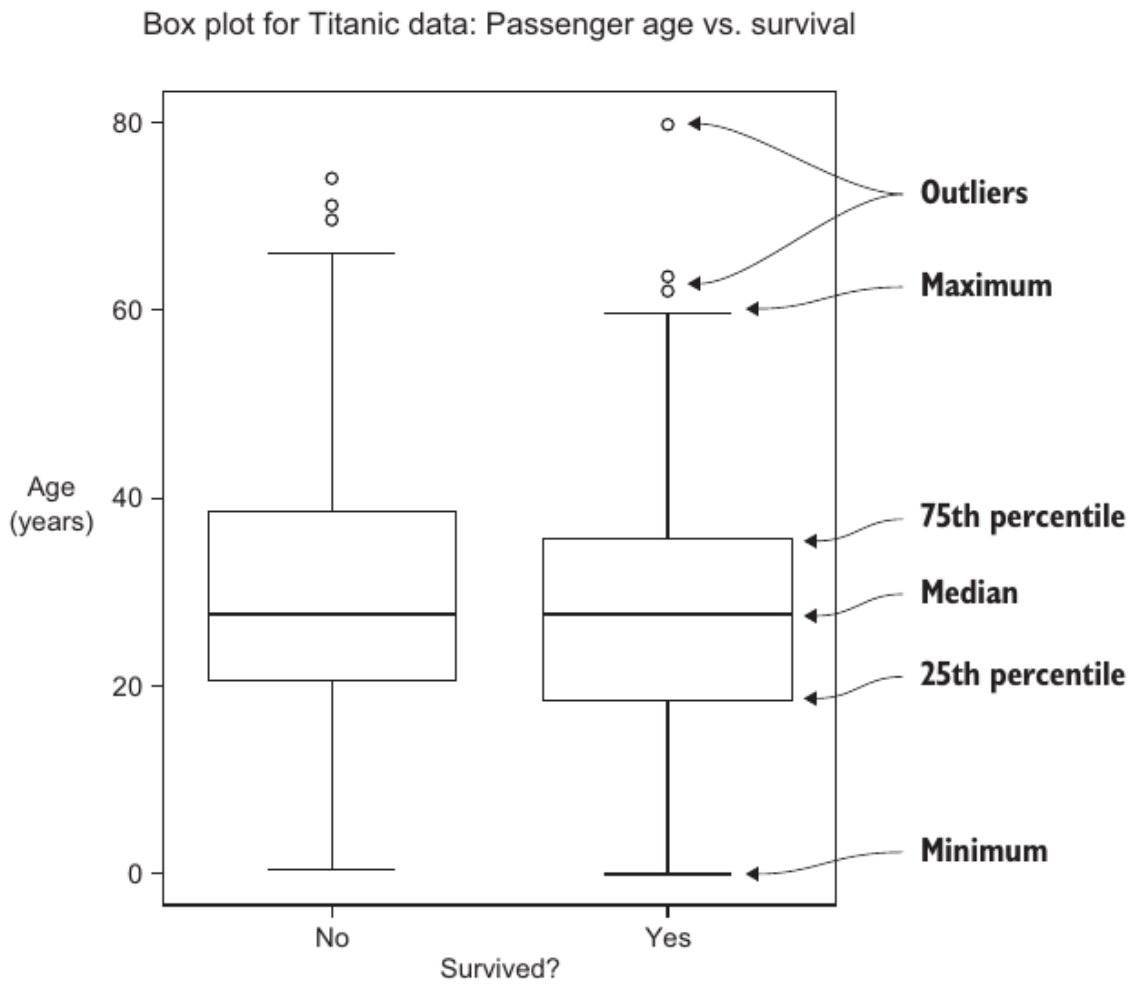



**Figure 2.14 Box plot showing the relationship between passenger age and survival on the Titanic. No noticeable differences exist between the age distributions for survivors versus fatalities. (This alone *shouldn't* be a reason to exclude age from the ML model, as it may still be a predictive factor.)**

## How to make sense of your data

You can visualize your data

- Mosaic plot
- Box plot
- Density plot
- Scatter plot (matrix)
- …

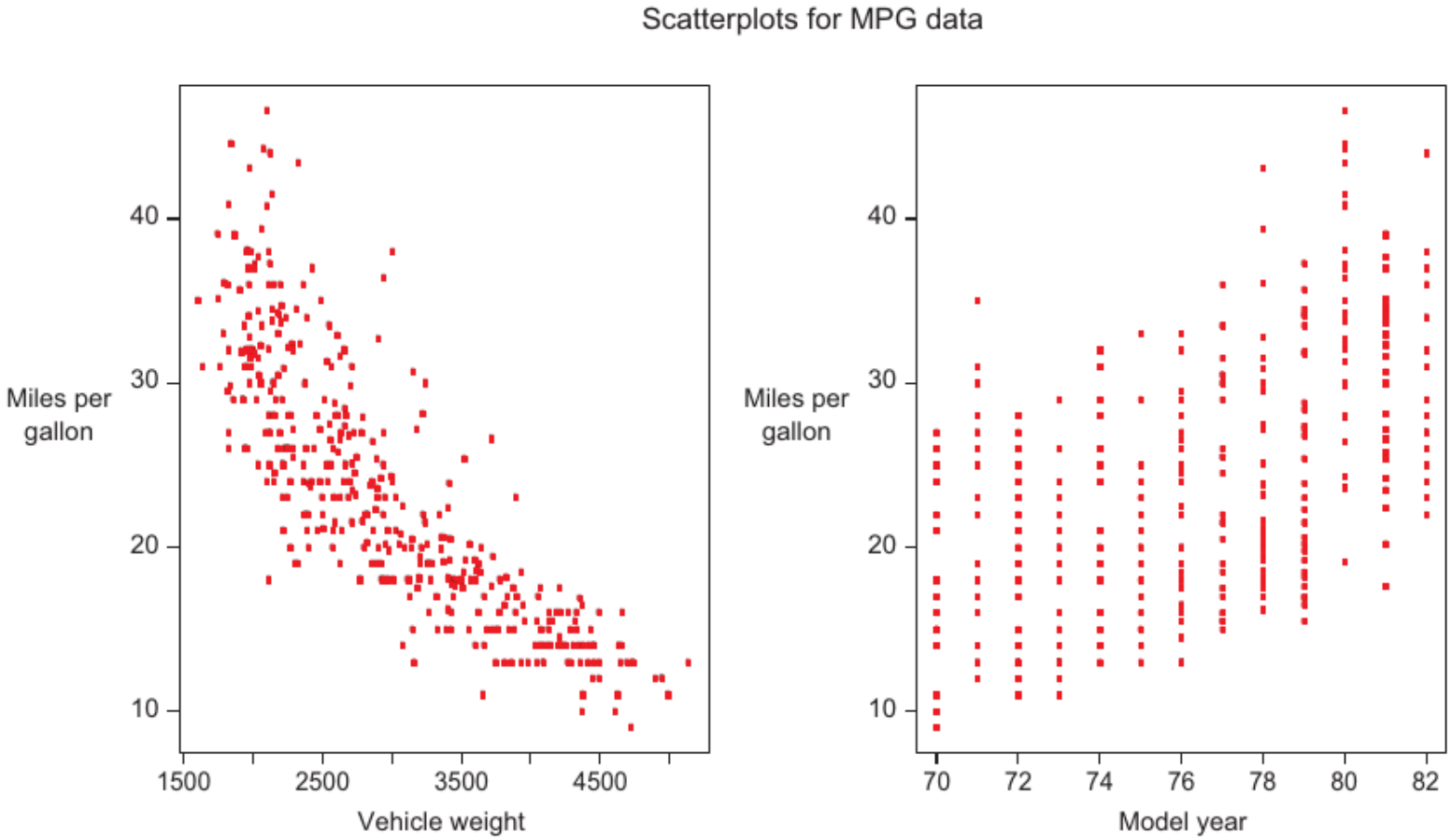
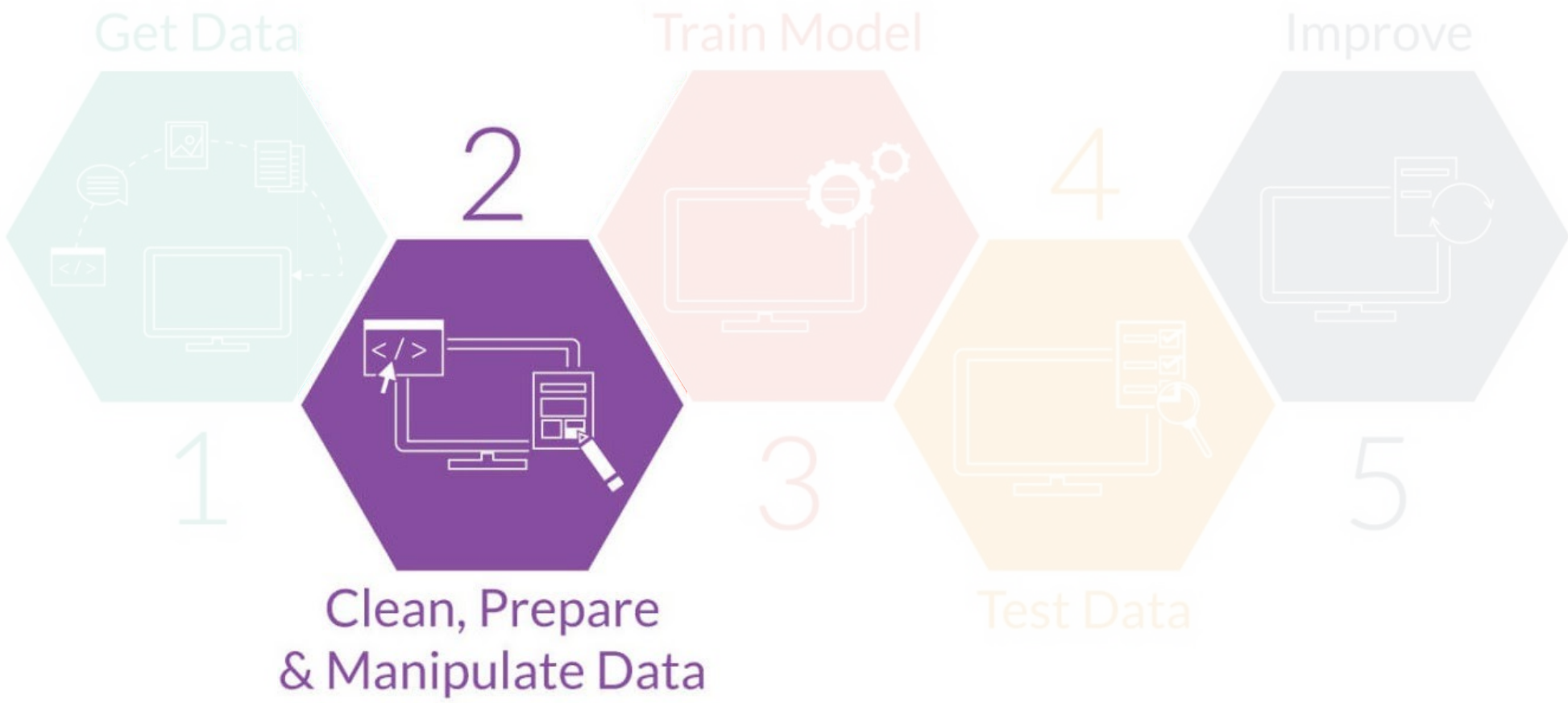



**Figure 2.17 Scatter plots for the relationship of vehicle miles per gallon versus vehicle weight (left) and vehicle model year (right)**

# Clean and Prepare Data



# Clean and Prepare Data

| PassengerId | Survived | Pclass | Gender | Age | SibSp | Parch | Ticket | Fare | Cabin | Embarked |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 3 | Male | 22 | 1 | 0 | A/5 21171 | 7.25 | | S |
| 2 | 1 | 1 | Female | 38 | 1 | 0 | PC 17599 | 71.2833 | C85 | C |
| 3 | 1 | 3 | Female | 26 | 0 | 0 | STON/02. 3101282 | 7.925 | | S |
| 4 | 1 | 1 | Female | 35 | 1 | 0 | 113803 | 53.1 | C123 | S |
| 5 | 0 | 3 | Male | 35 | 0 | 0 | 373450 | 8.05 | | S |
| 6 | 0 | 3 | Male | | 0 | 0 | 330877 | 8.4583 | | Q |


**Figure 2.7 The Titanic Passengers dataset has missing values in the Age and Cabin columns. The passenger information has been extracted from various historical sources, so in this case the missing values stem from information that couldn't be found in the sources.**

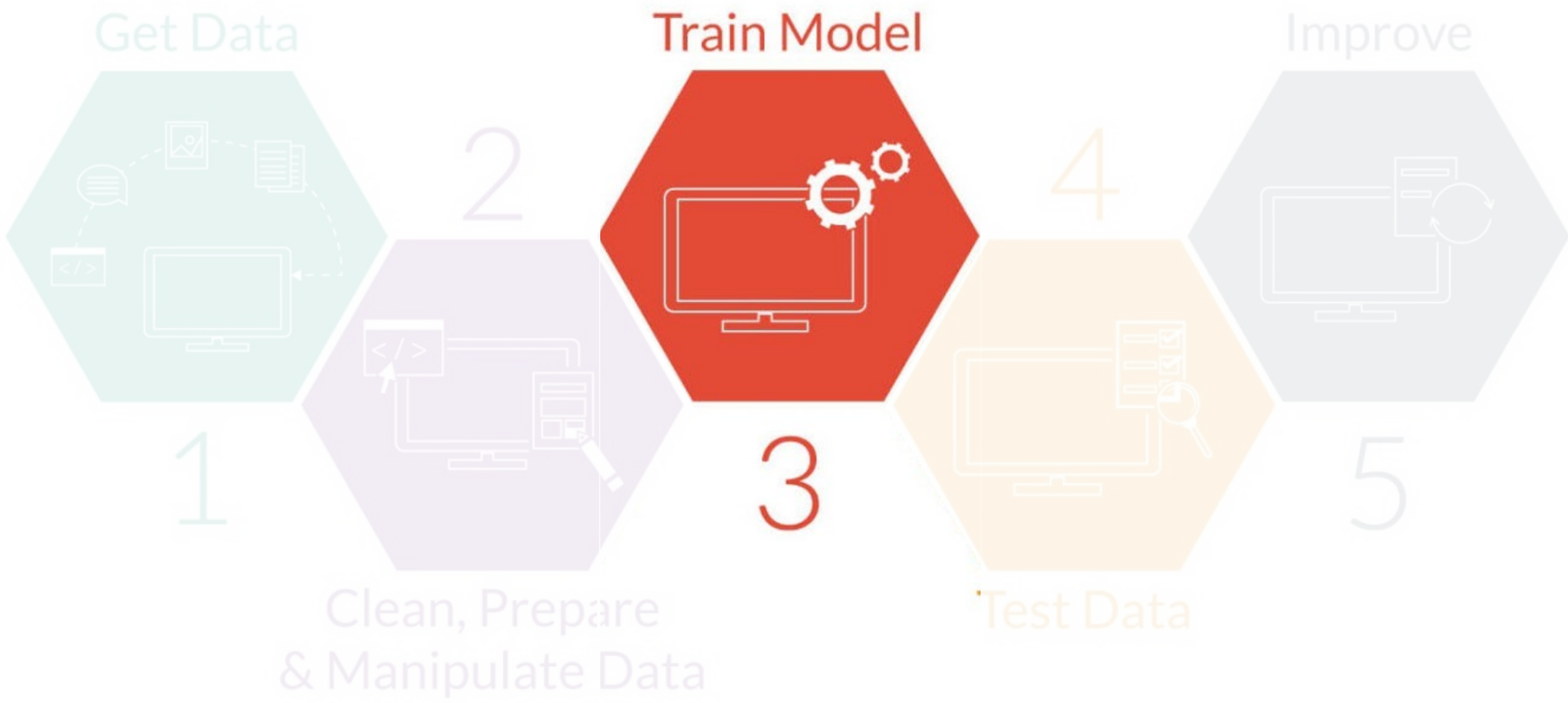
# Clean and Prepare Data

| Person | Name | Age | Income | Marital status |
|---|---|---|---|---|
| 1 | Jane Doe | 24 | 81,200 | Single |
| 2 | John Smith | 41 | 121,000 | Married |


| PassengerId | Survived | Pclass | Gender | Age | SibSp | Parch | Ticket | Fare | Cabin | Embarked |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 3 | Male | 22 | 1 | 0 | A/5 21171 | 7.25 | | S |
| 2 | 1 | 1 | Female | 38 | 1 | 0 | PC 17599 | 71.2833 | C85 | C |
| 3 | 1 | 3 | Female | 26 | 0 | 0 | STON/02. 3101282 | 7.925 | | S |
| 4 | 1 | 1 | Female | 35 | 1 | 0 | 113803 | 53.1 | C123 | S |
| 5 | 0 | 3 | Male | 35 | 0 | 0 | 373450 | 8.05 | | S |
| 6 | 0 | 3 | Male | | 0 | 0 | 330877 | 8.4583 | | Q |

**Figure 2.4 Identifying categorical features. At the top is the simple Person dataset, which has a Marital Status categorical feature. At the bottom is a dataset with information about Titanic passengers. The features identified as categorical here are Survived (whether the passenger survived or not), Pclass (what class the passenger was traveling on), Gender (male or female), and Embarked (from which city the passenger embarked).**

# Clean and Prepare Data



# Train Model

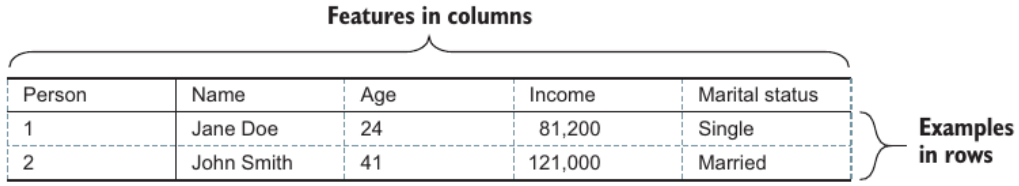



**Figure 1.8** In a tabular dataset, rows are called *instances* and columns represent *features*.

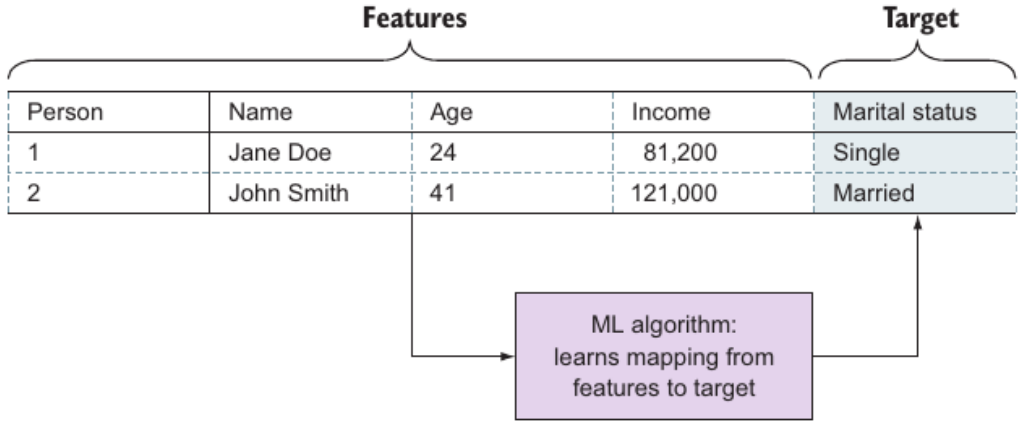
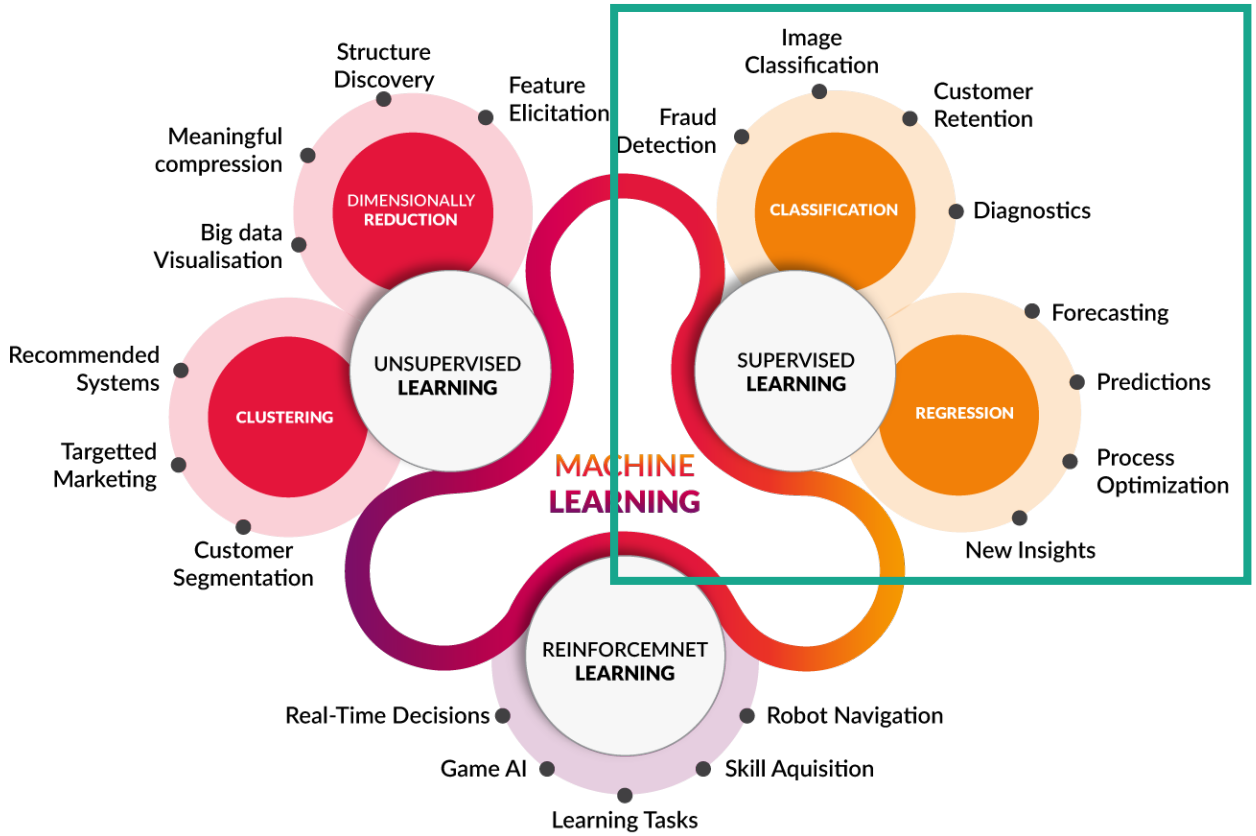
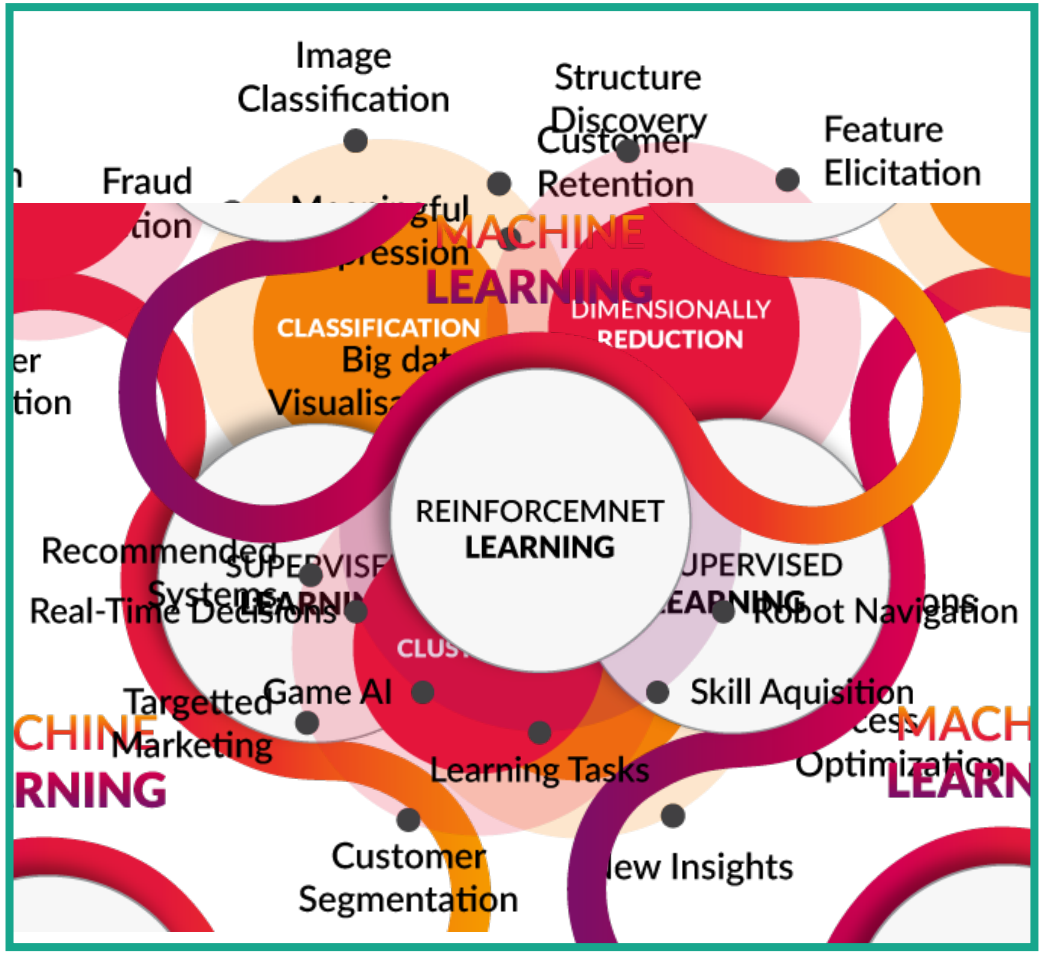



**Figure 1.9** The machine-learning modeling process

But what algorithm?

# The landscape of Machine Learning

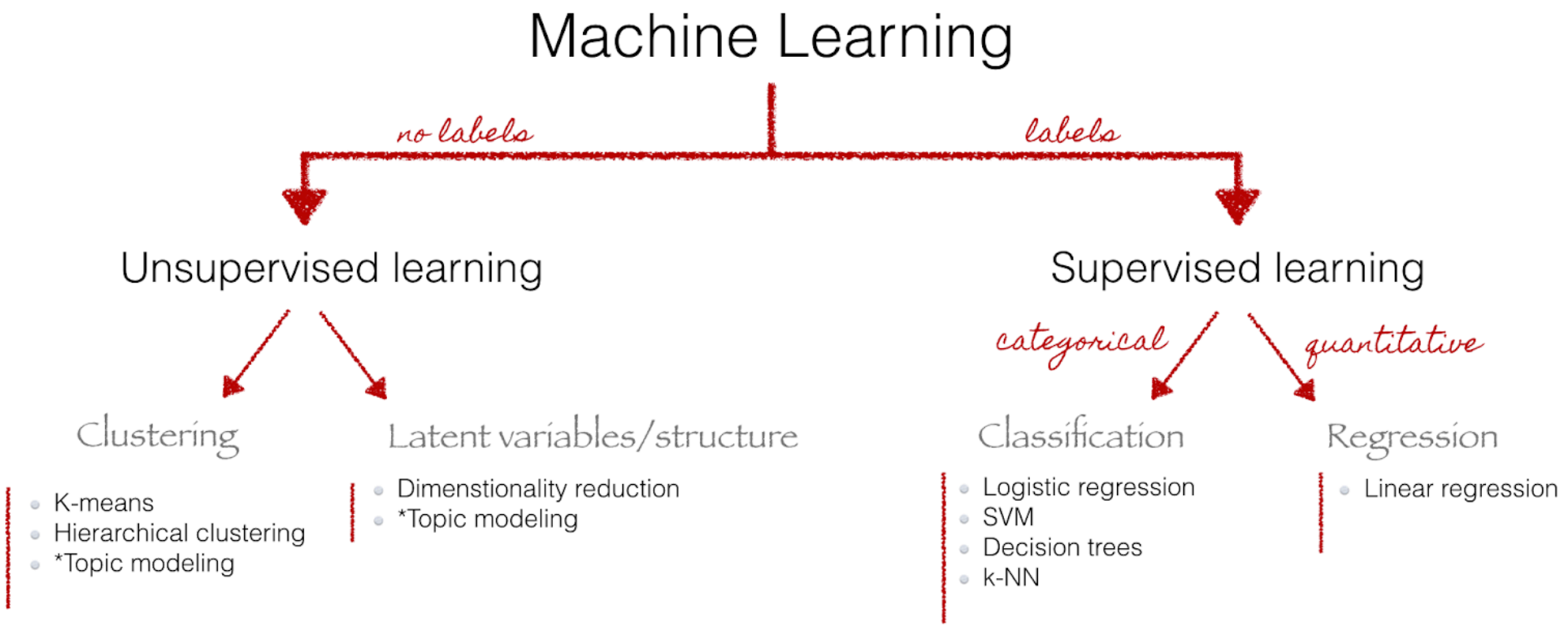



Source: Towards Data Science
Read further: https://litslink.com/blog/an-introduction-to-machine-learning-algorithms

# The landscape of Machine Learning

## Common algorithms



# Linear regression

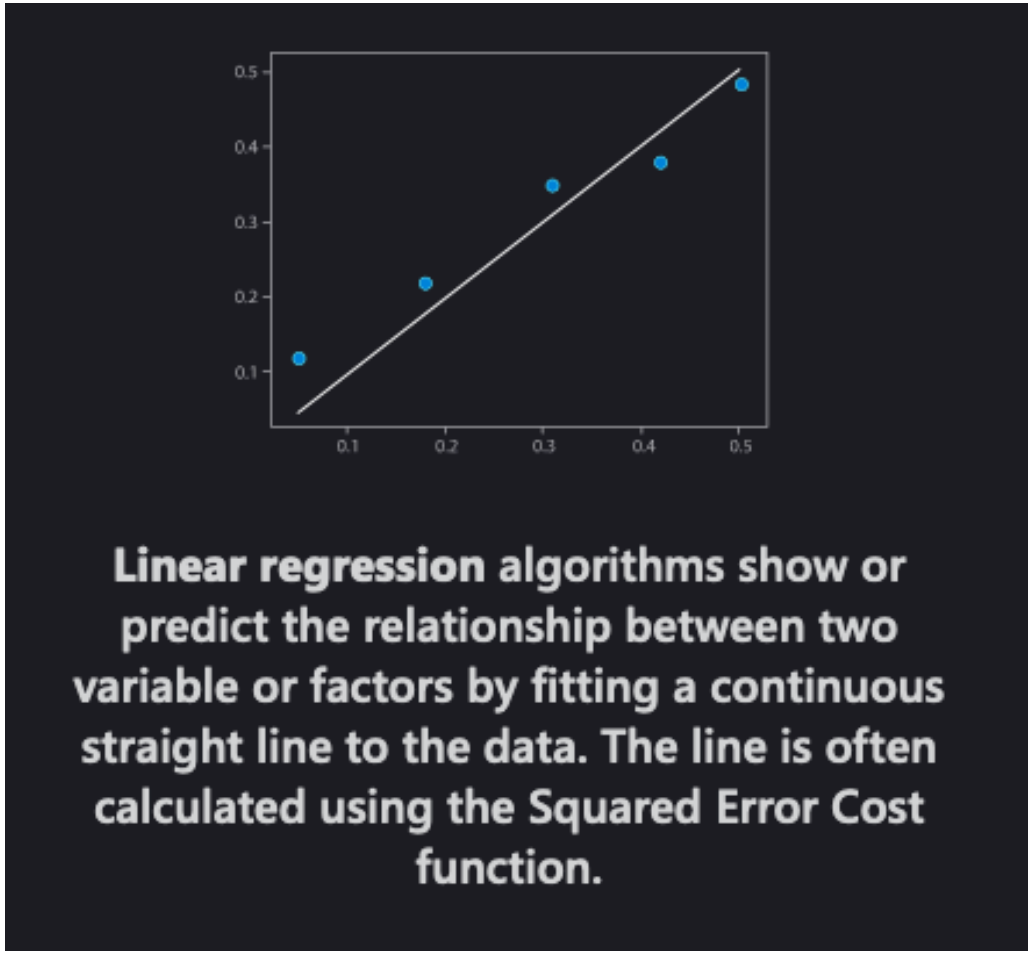
## Regression model

- We have a collection of labeled examples $\{(x_i, y_i)\}_{i=1}^{N}$, where $N$ is the size of the collection, $x_i$ is the $D$-dimensional feature vector of example $i = 1, \ldots, N$, $y_i$ is a real-valued target and every feature $x_i^{(j)}, j = 1, \ldots, D$, is also a real number.
- We want to build a model $f_{w,b}(x)$ as a linear combination of features of example $x$:

$$f_{w,b}(x) = wx + b$$

  where $w$ is a $D$-dimensional vector of parameters and $b$ is a real number.
- We will use the model to predict the unknown $y$ for a given $x$ like this: $y \leftarrow f_{w,b}(x)$. We want to find the optiomal values $(w^*, b^*)$.
- The objective function we want to minimize is

$$\frac{1}{N}\sum_{i=1}^{N}\left(f_{w,b}(x_i) - y_i\right)^2 \ (squared\ error\ loss).$$



# Logistic regression

## Classification model

- In logistic regression, we still want to model $y_i$ as a linear function of $x_i$, however, with a binary $y_i$, this is not straightforward.
- If we define the negative label as 0 and the positive label as 1, we would just need to find a simple continuous function whose codomain is (0,1).
- One function that has such a property is the standard logistic function (also known as sigmoid):

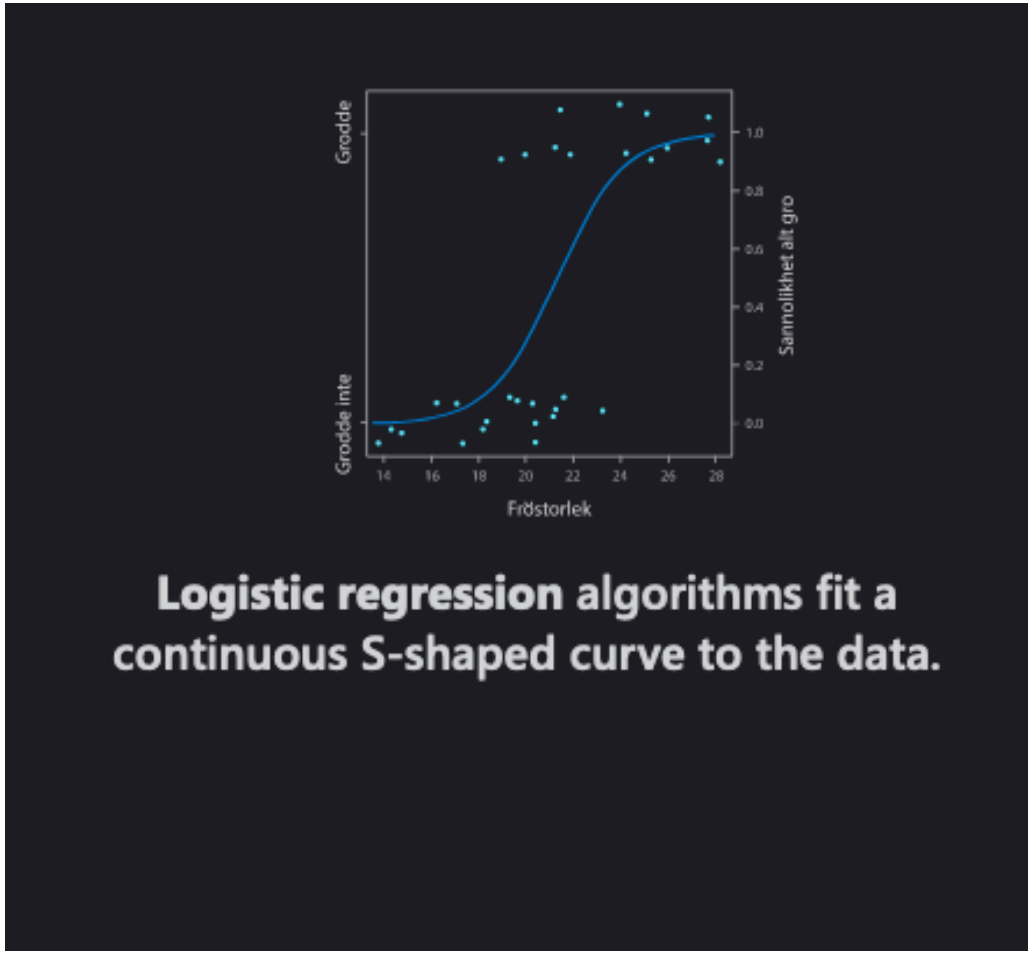
$$f(x) = \frac{1}{1 + e^{-x}}$$

- The logistic regression model looks like this

$$f_{w,b}(x) = \frac{1}{1 + e^{-(wx+b)}}.$$

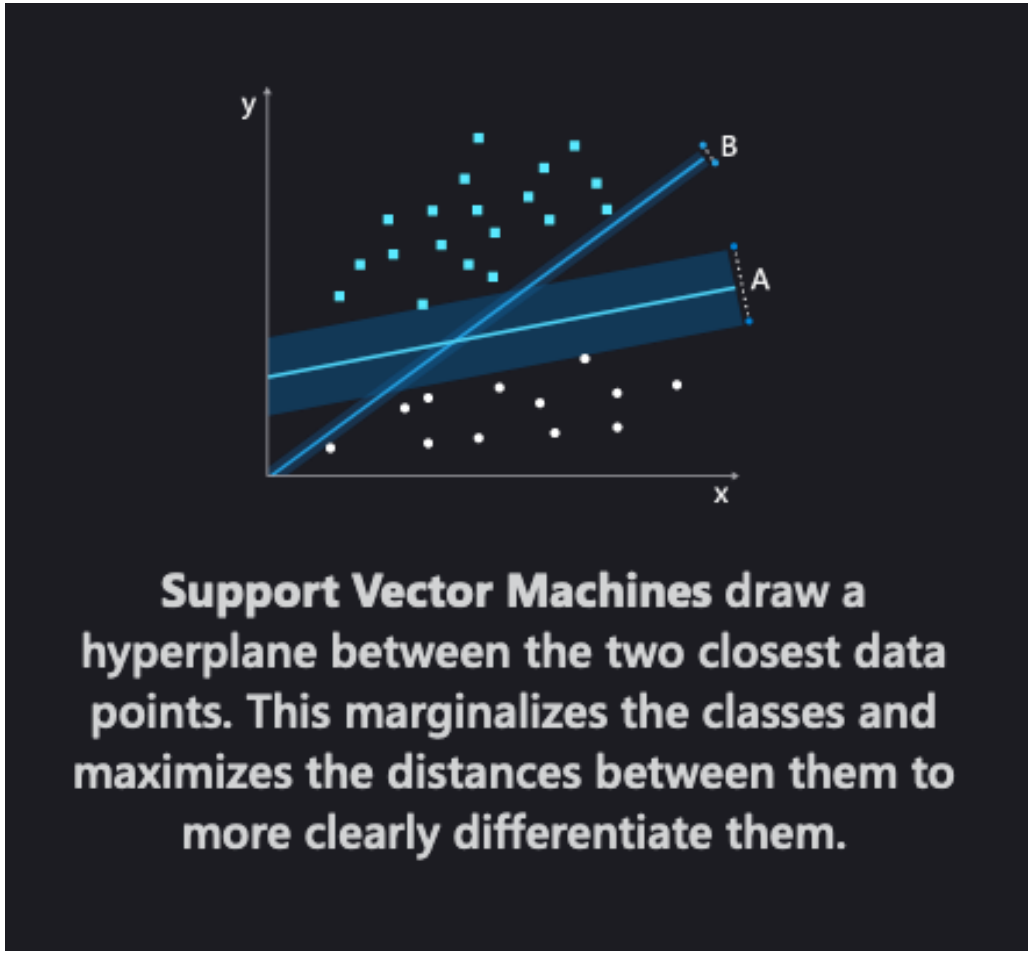
- If we optimize the values of $w$ and $b$ appropriately, we could interpret the output of $f(x)$ as the probability of $y_i$ being positive.
- The objective function in logistic regression is the likelihood of our training set according to the model:

$$L_{w,b} = \prod_{i=1\ldots N} f_{w,b}(x_i)^{y_i} \left(1 - f_{w,b}(x_i)\right)^{1-y_i} \quad (assumes\ iid)$$



# Support Vector Machine

## Classification



# Naïve Bayers

## Classification

- Naïve Bayes is a conditional probability model: given a problem instance to be classified, represented by a vector $x = (x_1, \dots, x_n)$ representing some $n$ features (independent variables), it assigns to this instance probabilities $p(C_k|x_1, \dots, x_n)$ for each of $K$ possible outcomes or classes $C_k$.
- The problem with the above formulation is that if the number of features $n$ is large or if a feature can take on a large number of values, then basing such a model on probability tables is infeasible. Using Bayes' theorem, the conditional probability can be decomposed as

$$p(C_k|x) = \frac{p(x|C_k)p(C_k)}{p(x)}$$

- In plain English, using Bayesian probability terminology, the above equation can be written as

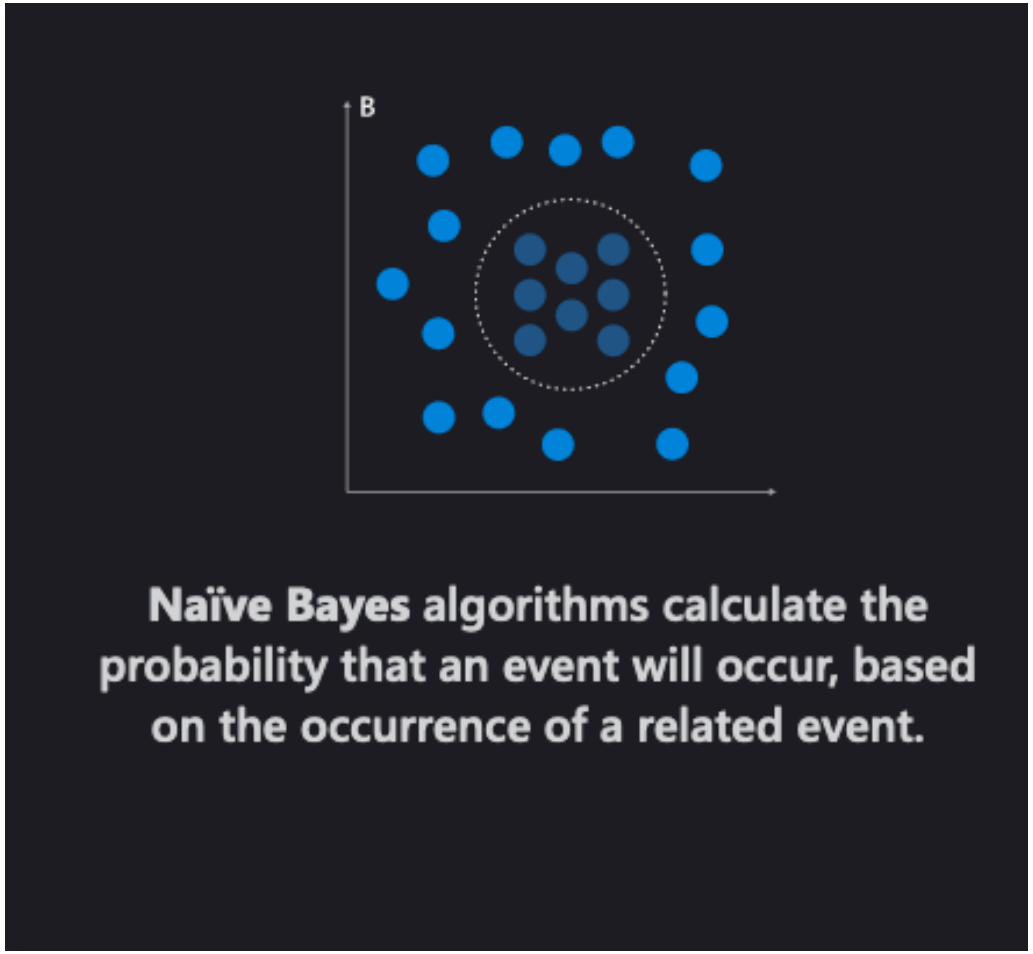
$$posterior = \frac{likelihood\ x\ prior}{evidence}$$

- Now the "naïve" conditional independence assumptions come into play: assume that all features in $x$ are mutually independent, conditional on the category $C_k$. Under this assumption

$$p(C_k|x) \propto p(C_k) \prod_{i=1}^{N} p(x_i|C_k)$$

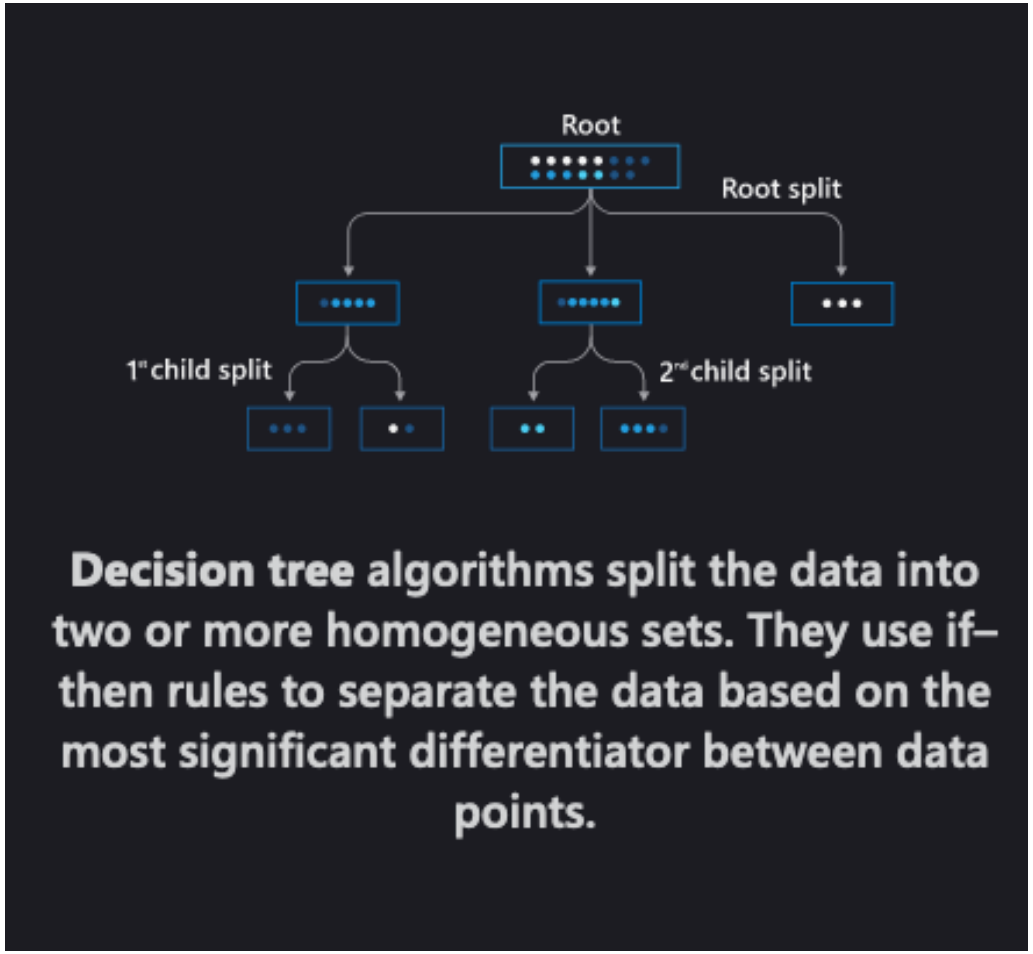
- A Bayes classifier assigns a class label $\hat{y} = C_k$ as follows:

$$\hat{y} = \underset{k \in \{1, \dots, K\}}{argmax}\ p(C_k) \prod_{i=1}^{N} p(x_i|C_k)$$



# Decision tree

## Classification

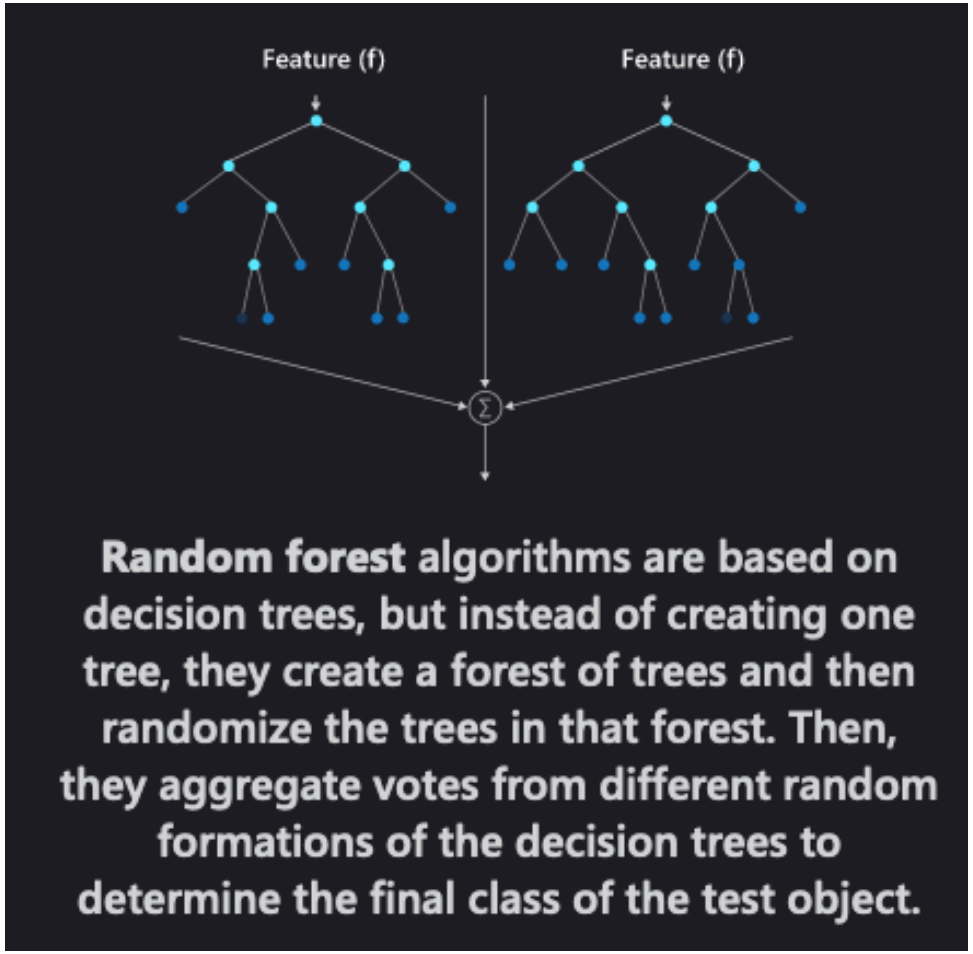


# K-Nearest neighbor

## Classification/Regression

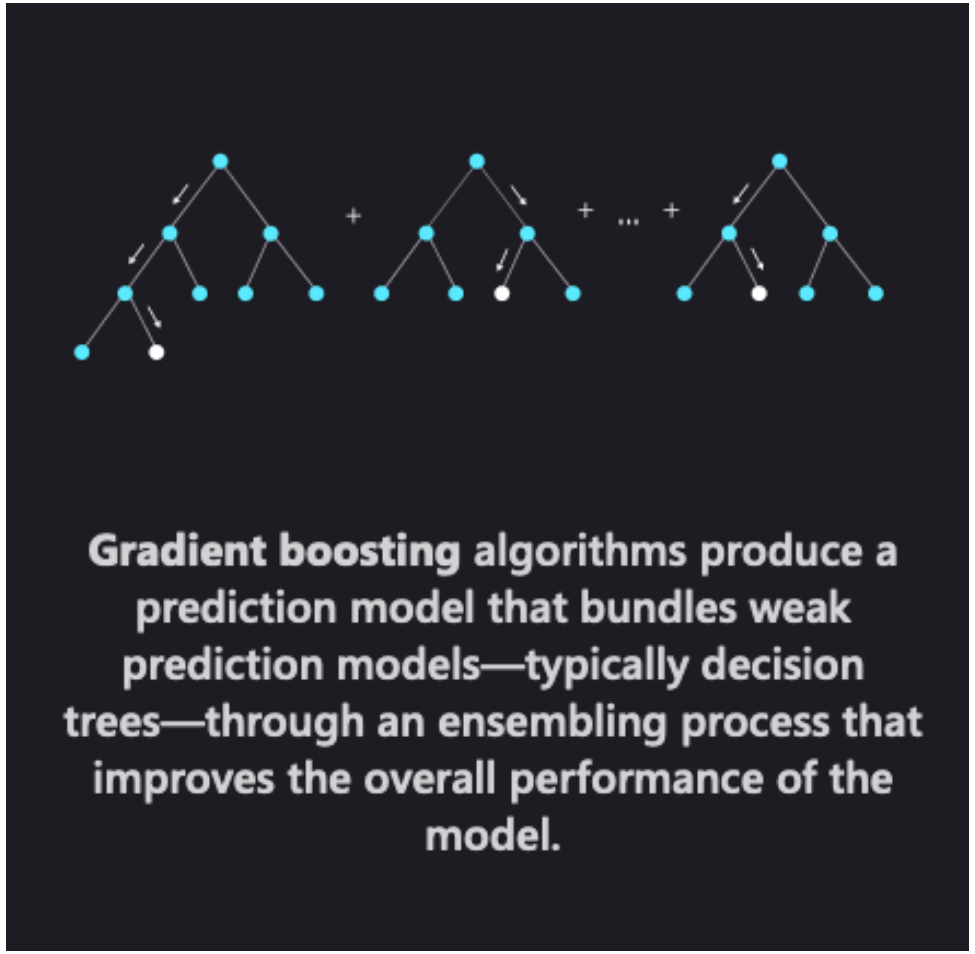
# Random forest

## Classification/Regression



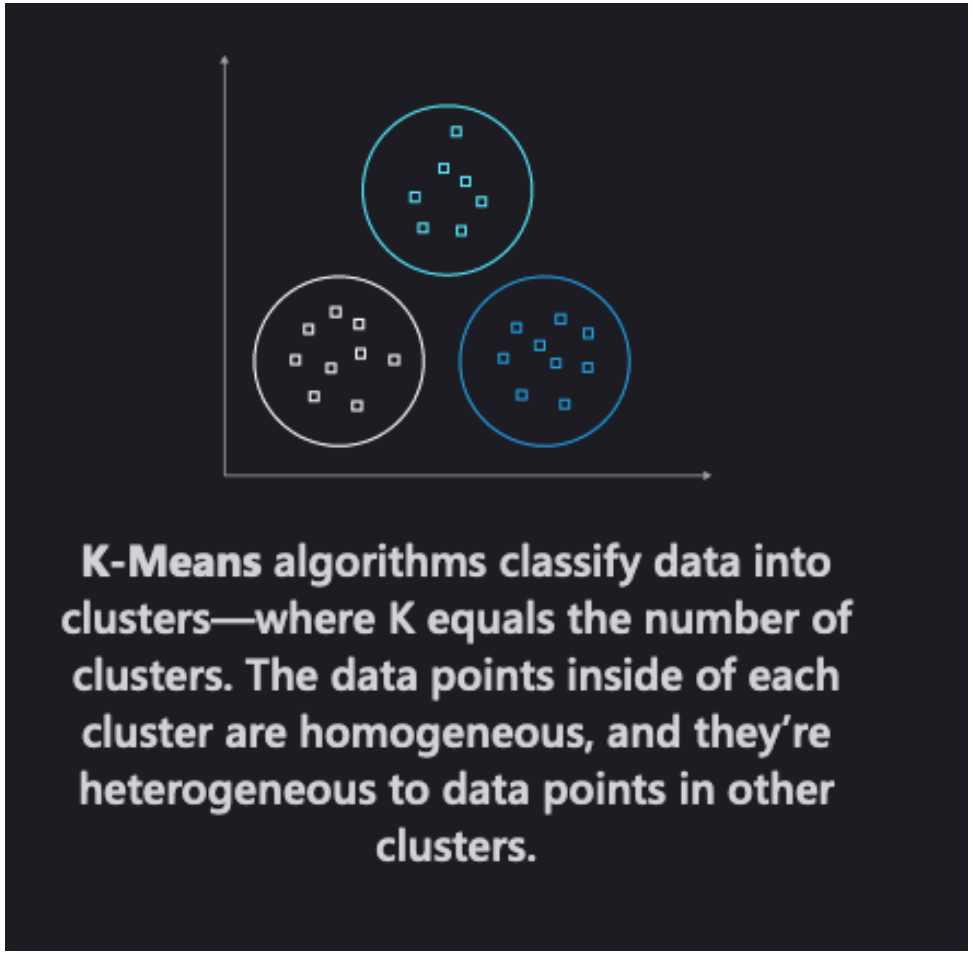
# Gradient boosting

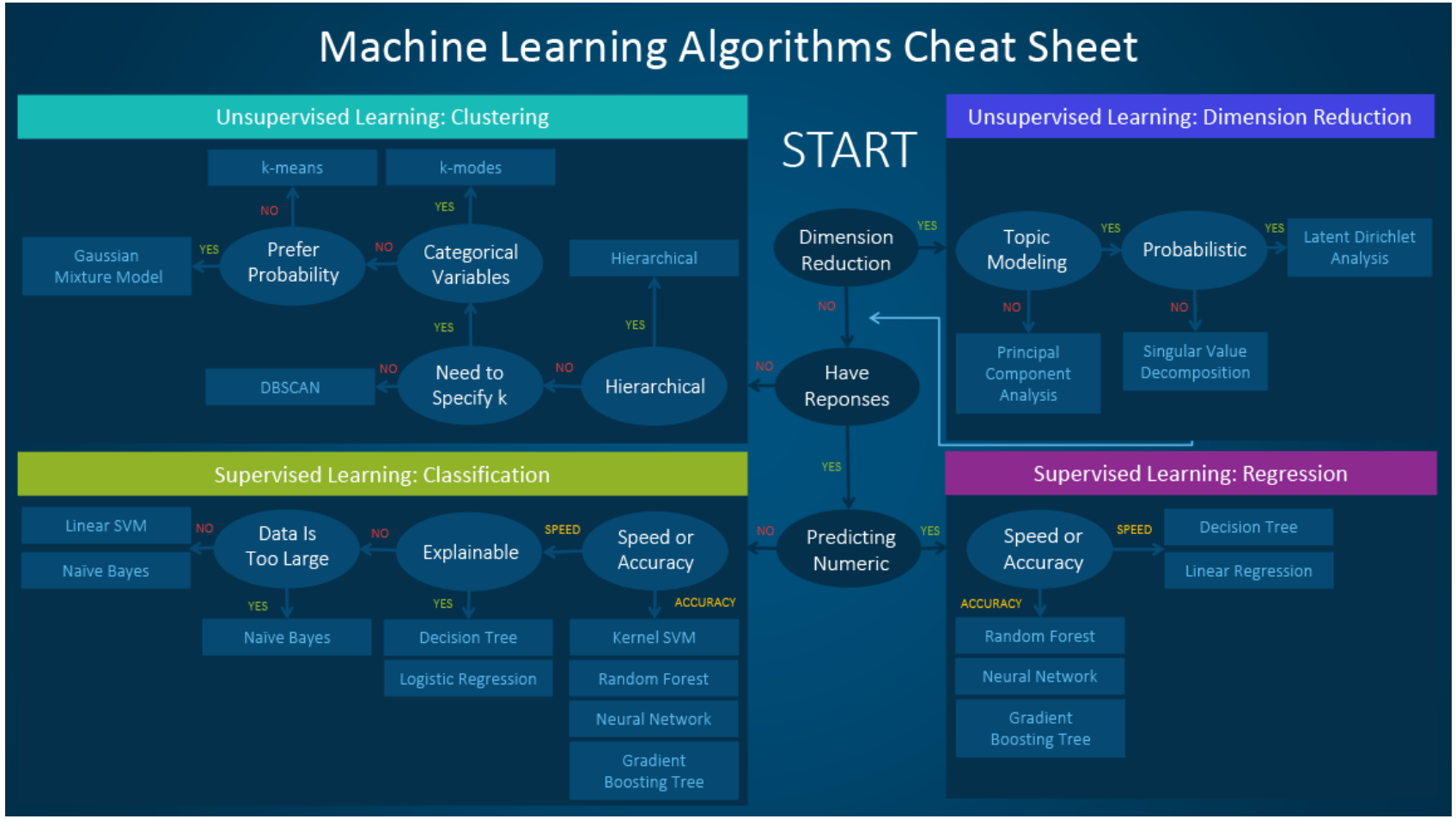
## Classification/Regression



# K-Means

## Clustering

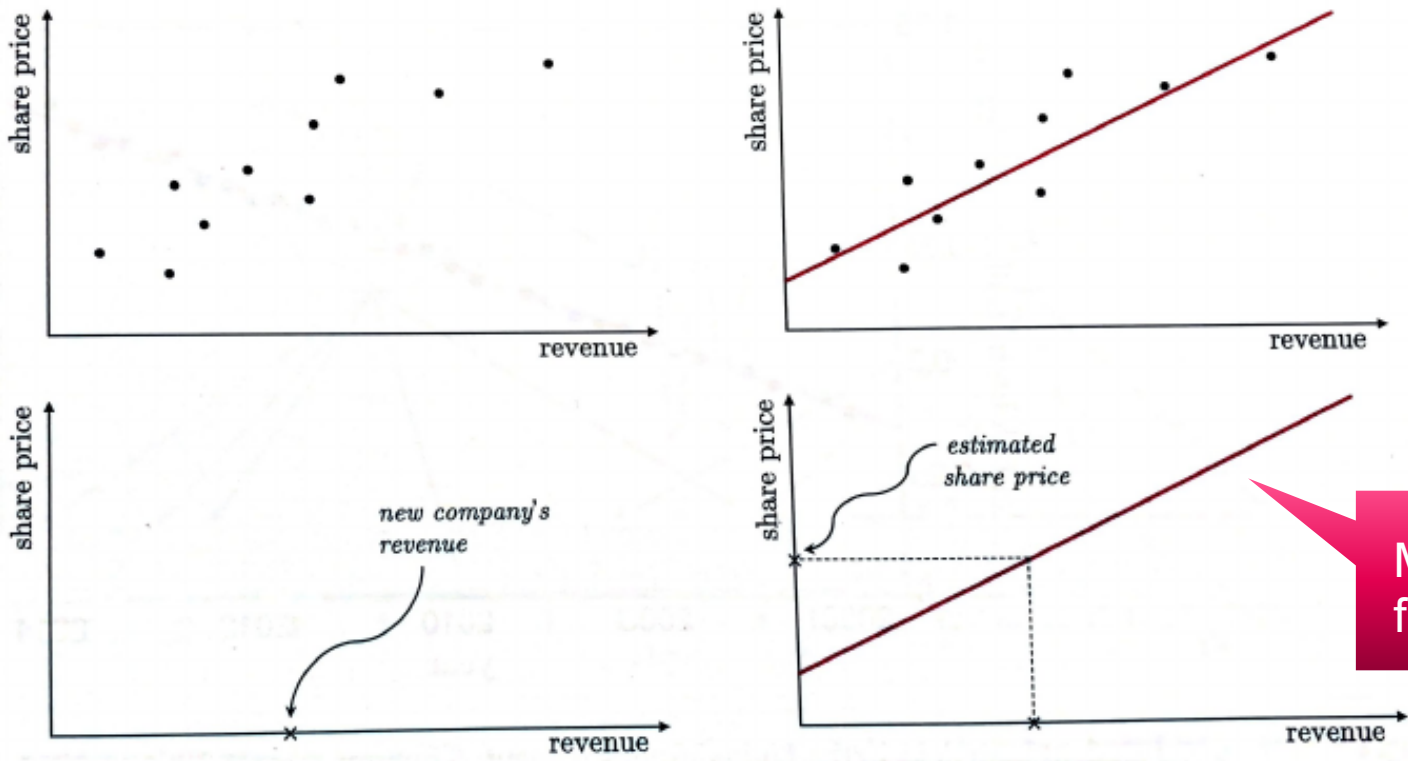





https://blogs.sas.com/content/subconsciousmusings/2017/04/12/machine-learning-algorithm-use/

# Classification vs Regression

## Regression




Model data – fitting curve

**Fig. 1.7** (top left panel) A toy training dataset of ten corporations with their associated share price and revenue values. (top right panel) A linear model is fit to the data. This trend line models the overall trajectory of the points and can be used for prediction in the future as shown in the bottom left and bottom right panels.

# Classification vs Regression

## Classification

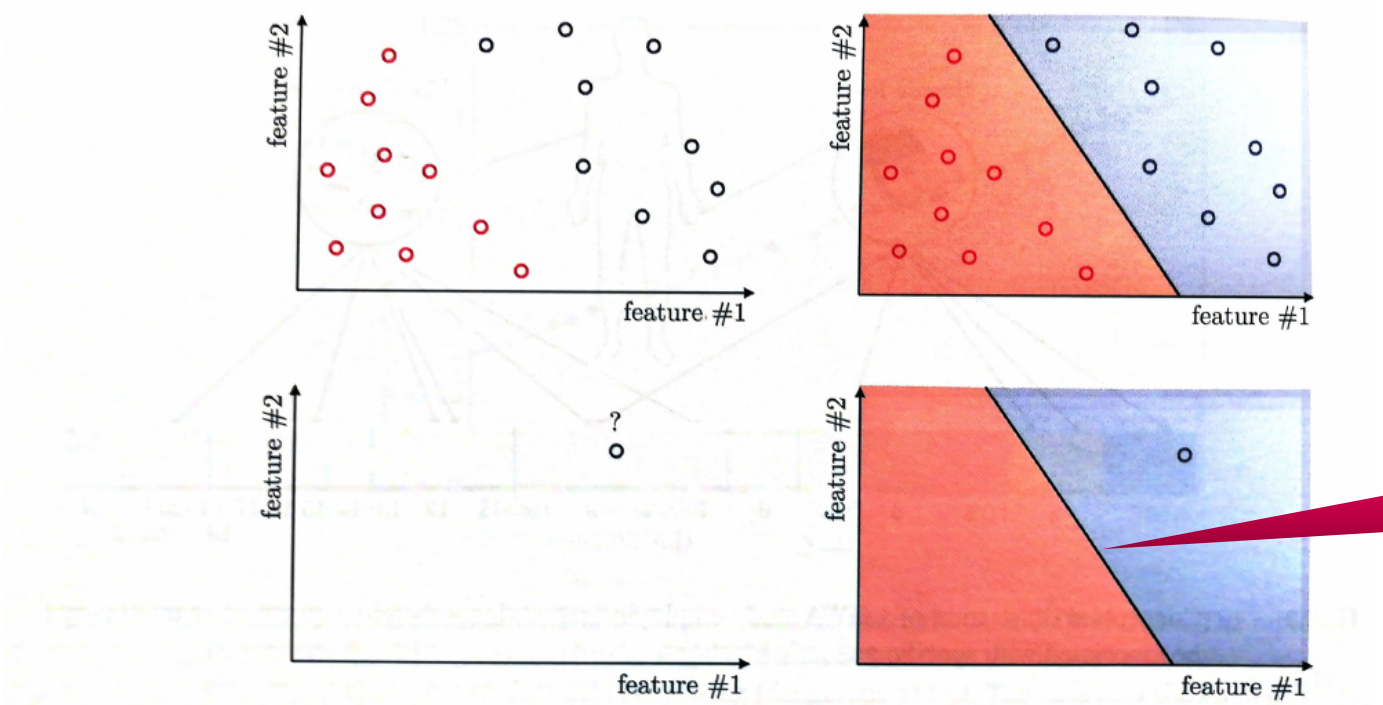
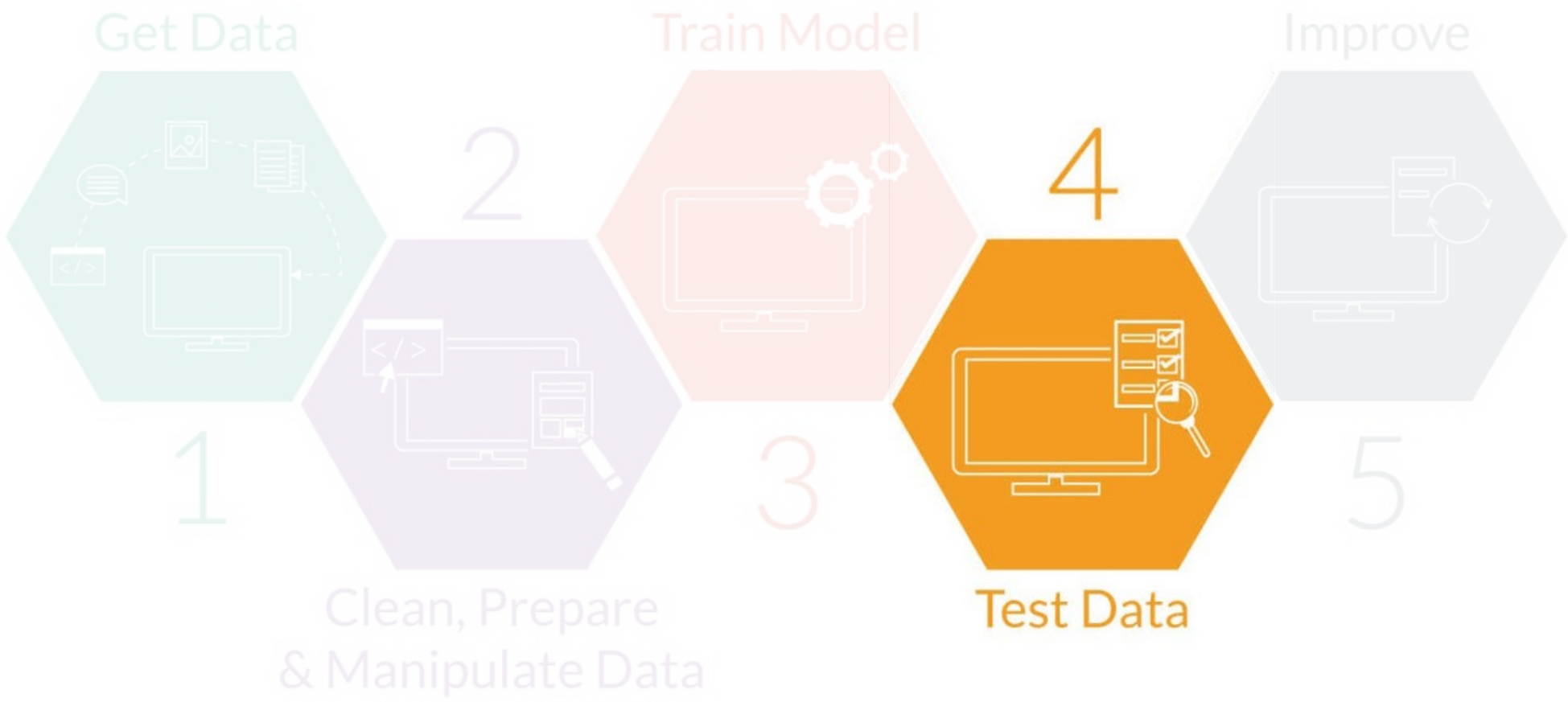



**Fig. 1.10** (top left panel) A toy 2-dimensional training set consisting of two distinct classes, red and blue. (top right panel) A linear model is trained to separate the two classes. (bottom left panel) A test point whose class is unknown. (bottom right panel) The test point is classified as blue since it lies on the blue side of the trained linear classifier.

Divide data – Decision plane

# Test Data


Get Data
Train Model
Improve
2
4
1
3
5
Clean, Prepare
& Manipulate Data
Test Data

# Test Data

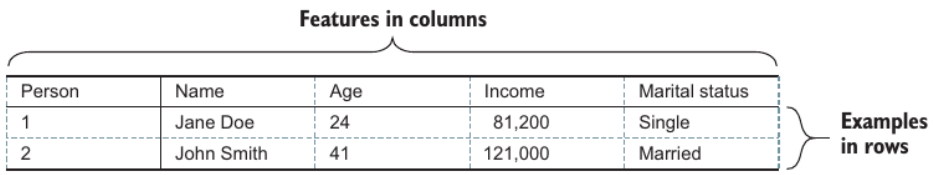
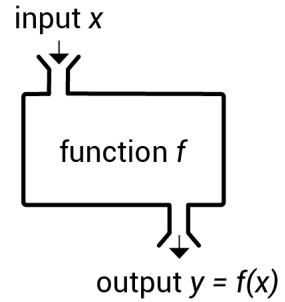



**Figure 1.8** In a tabular dataset, rows are called *instances* and columns represent *features*.



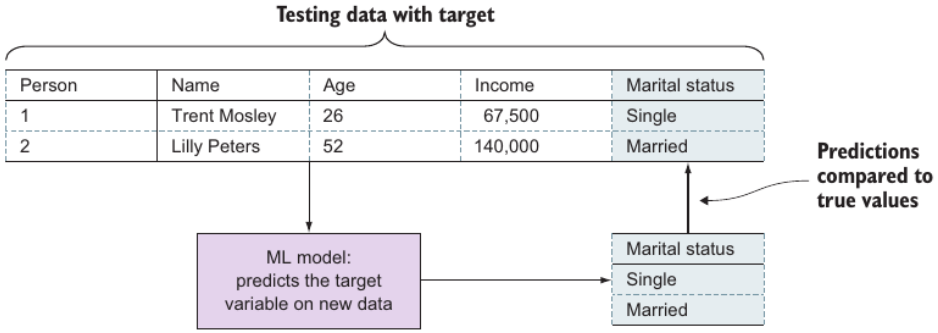




**Figure 1.11** When using a testing set to evaluate model performance, you "pretend" that the target variable is unknown and compare the predictions with the true values.

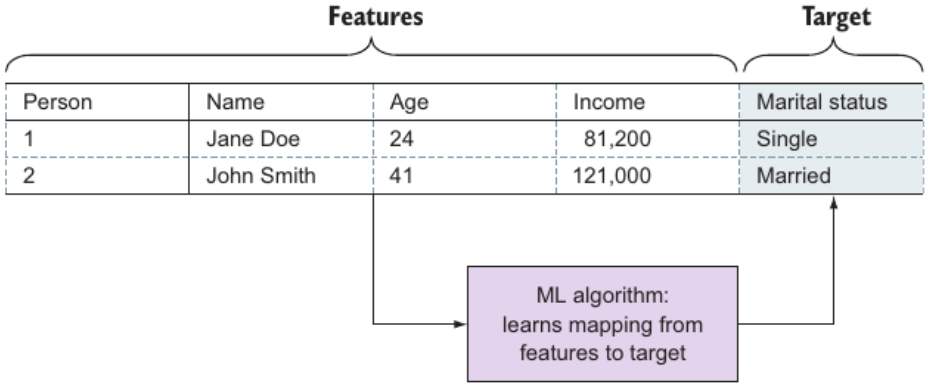



**Figure 1.9** The machine-learning modeling process

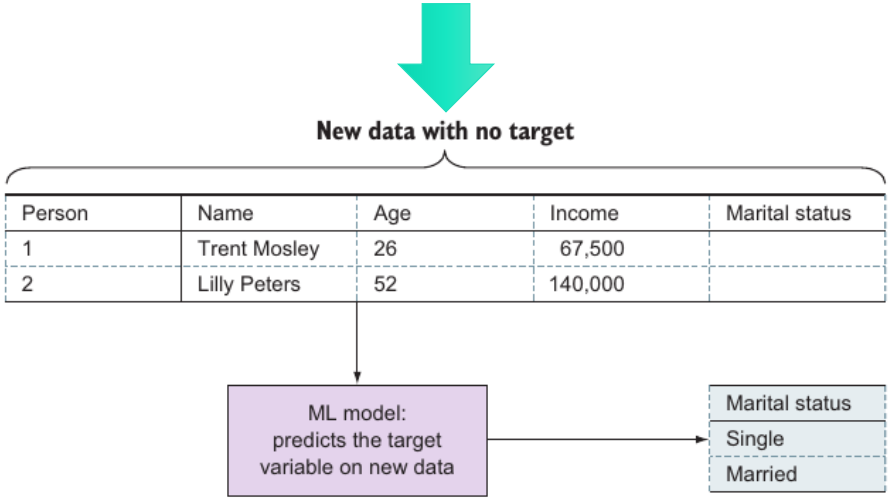



**Figure 1.10** Using the model for prediction on new data

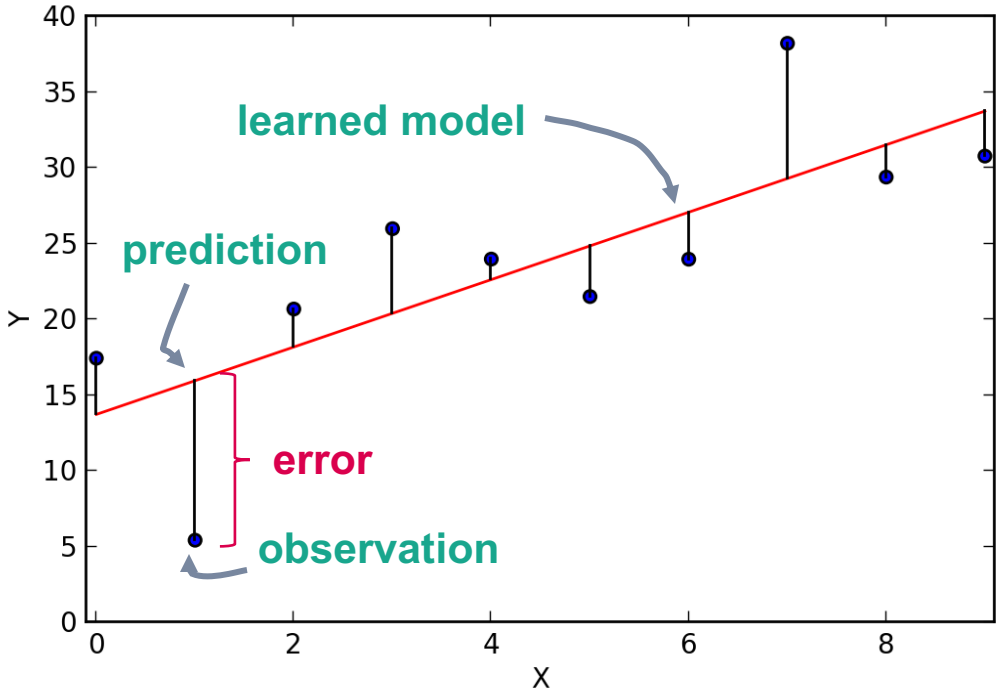
# How do we measure our model's performance?

## Regression

- (root) mean square error

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

Mean deviation



# How do we measure our model's performance?

## Classification

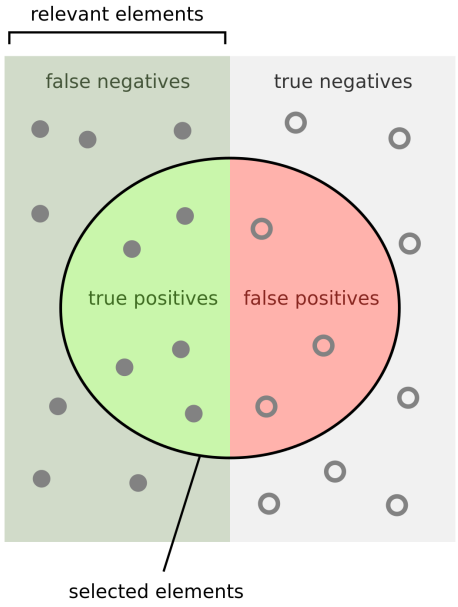
- (root) mean square error
- confusion matrix

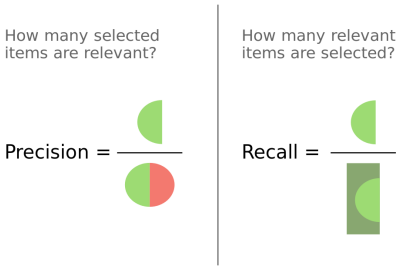
$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{All Samples}}$$

Percentage of correct predictions

| | | Predicted | |
|---|---|---|---|
| | | **Negative** | **Positive** |
| **Actual** | **Negative** | True Negative | False Positive |
| | **Positive** | False Negative | True Positive |

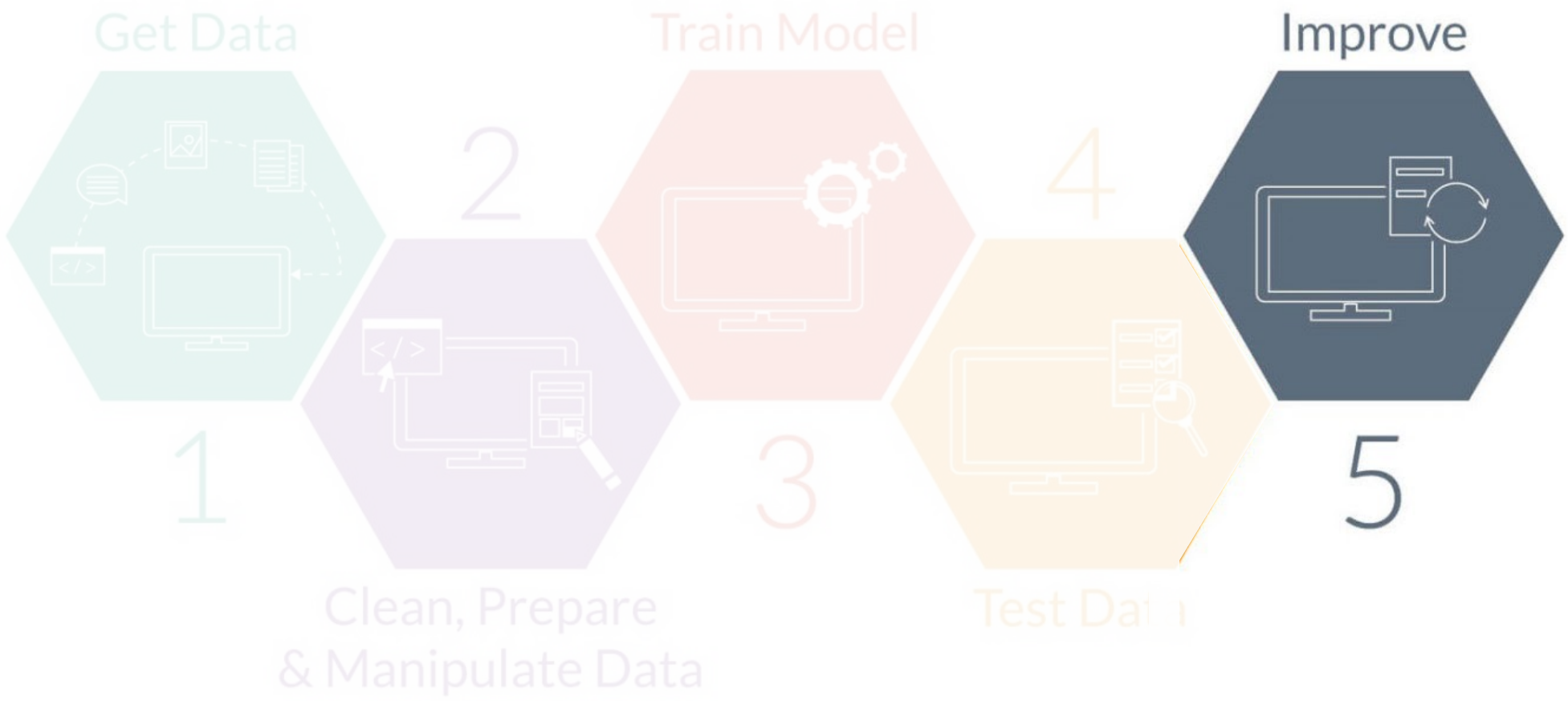
# Beyond accuracy



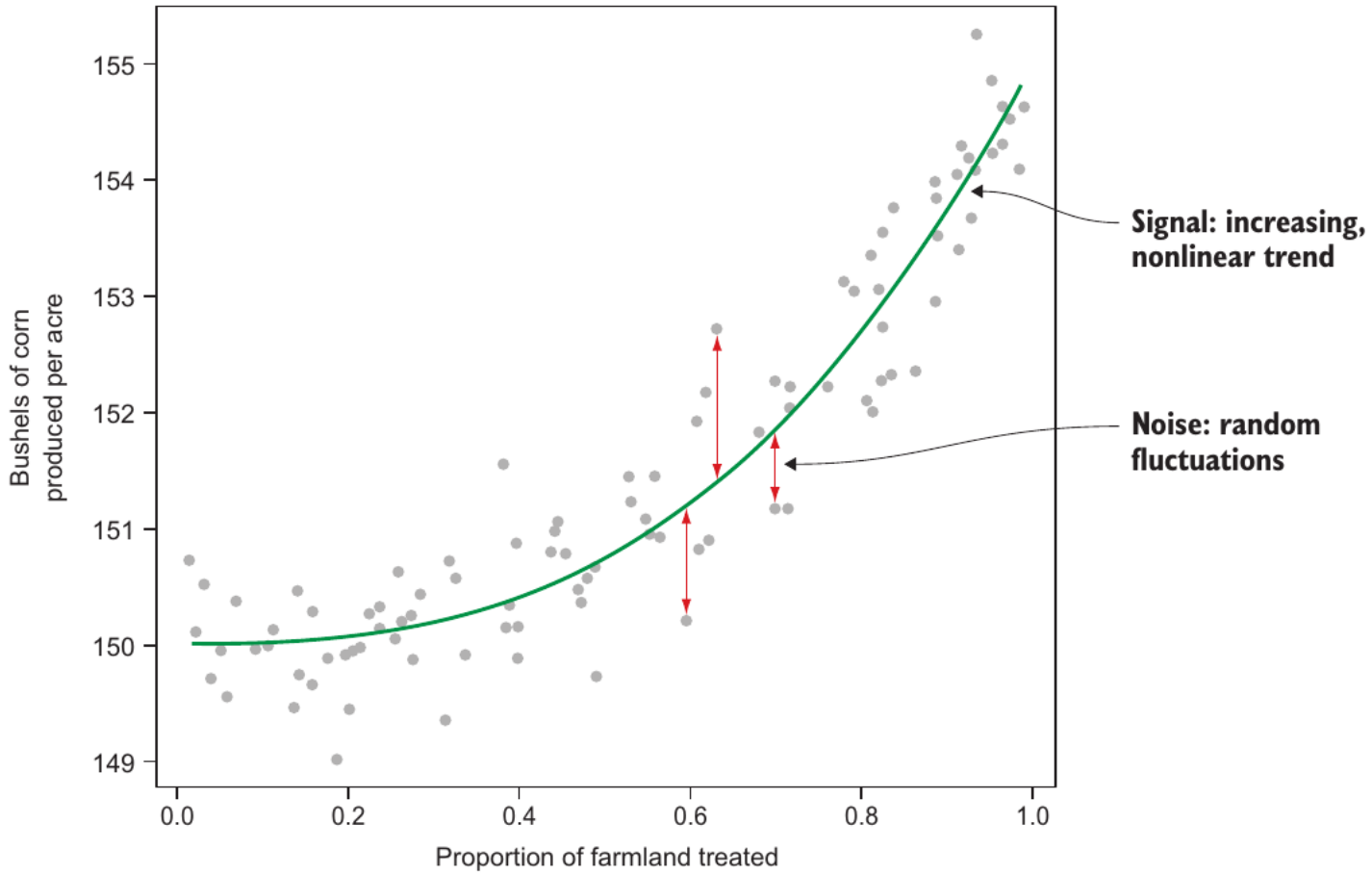

| | | True condition | | | | |
|---|---|---|---|---|---|---|
| | Total population | Condition positive | Condition negative | Prevalence = $\frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$ | Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$ | |
| Predicted condition | Predicted condition positive | **True positive** | **False positive**, Type I error | Positive predictive value (PPV), Precision = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$ | False discovery rate (FDR) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$ | |
| | Predicted condition negative | **False negative**, Type II error | **True negative** | False omission rate (FOR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$ | Negative predictive value (NPV) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$ | |
| | | True positive rate (TPR), Recall, Sensitivity, probability of detection, Power = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$ | False positive rate (FPR), Fall-out, probability of false alarm = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$ | Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$ | Diagnostic odds ratio (DOR) = $\frac{\text{LR+}}{\text{LR−}}$ | $F_1$ score = $2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$ |
| | | False negative rate (FNR), Miss rate = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$ | Specificity (SPC), Selectivity, True negative rate (TNR) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$ | Negative likelihood ratio (LR−) = $\frac{\text{FNR}}{\text{TNR}}$ | | |

# Improve Model



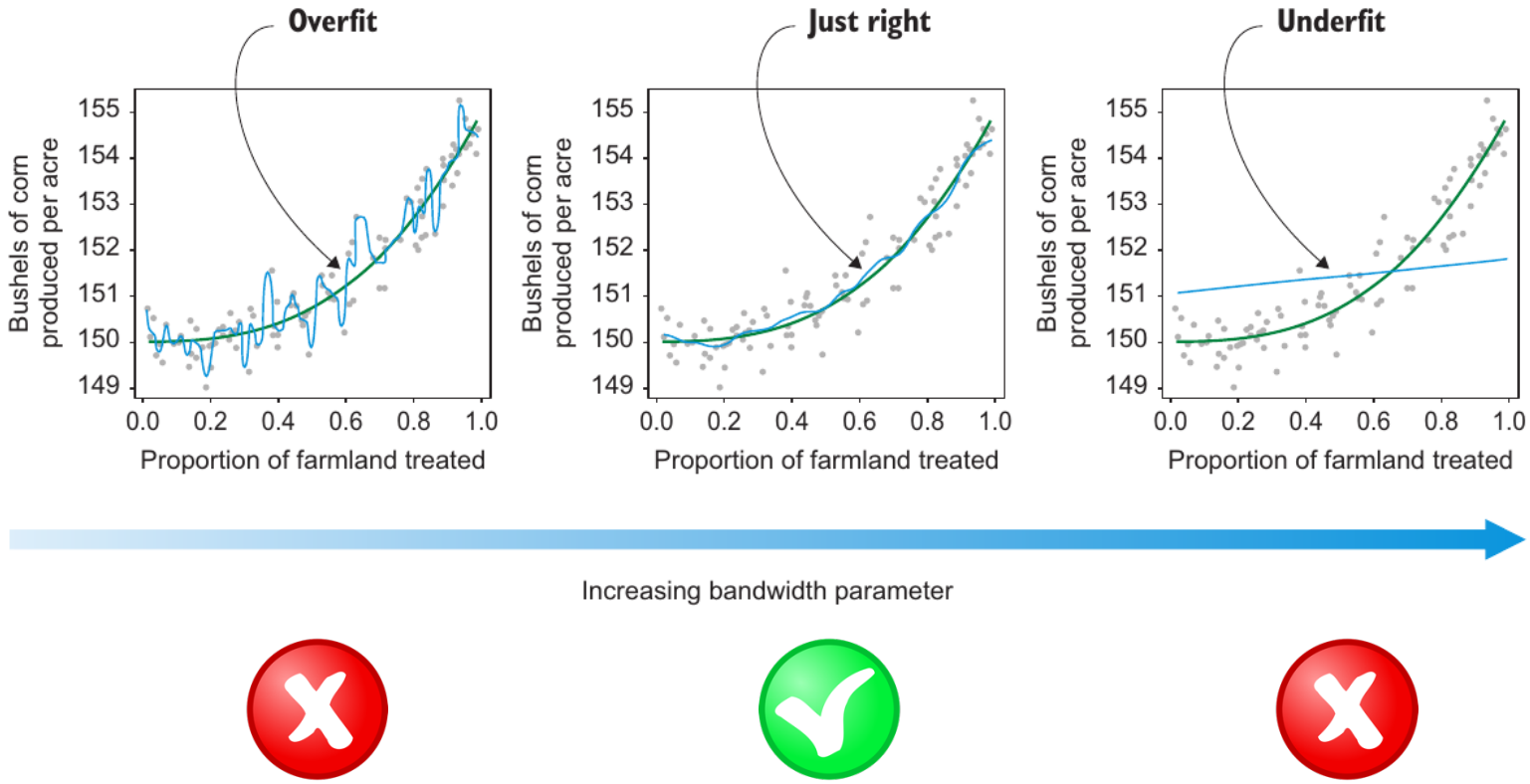
# Real world data

## signal-to-noise ratio



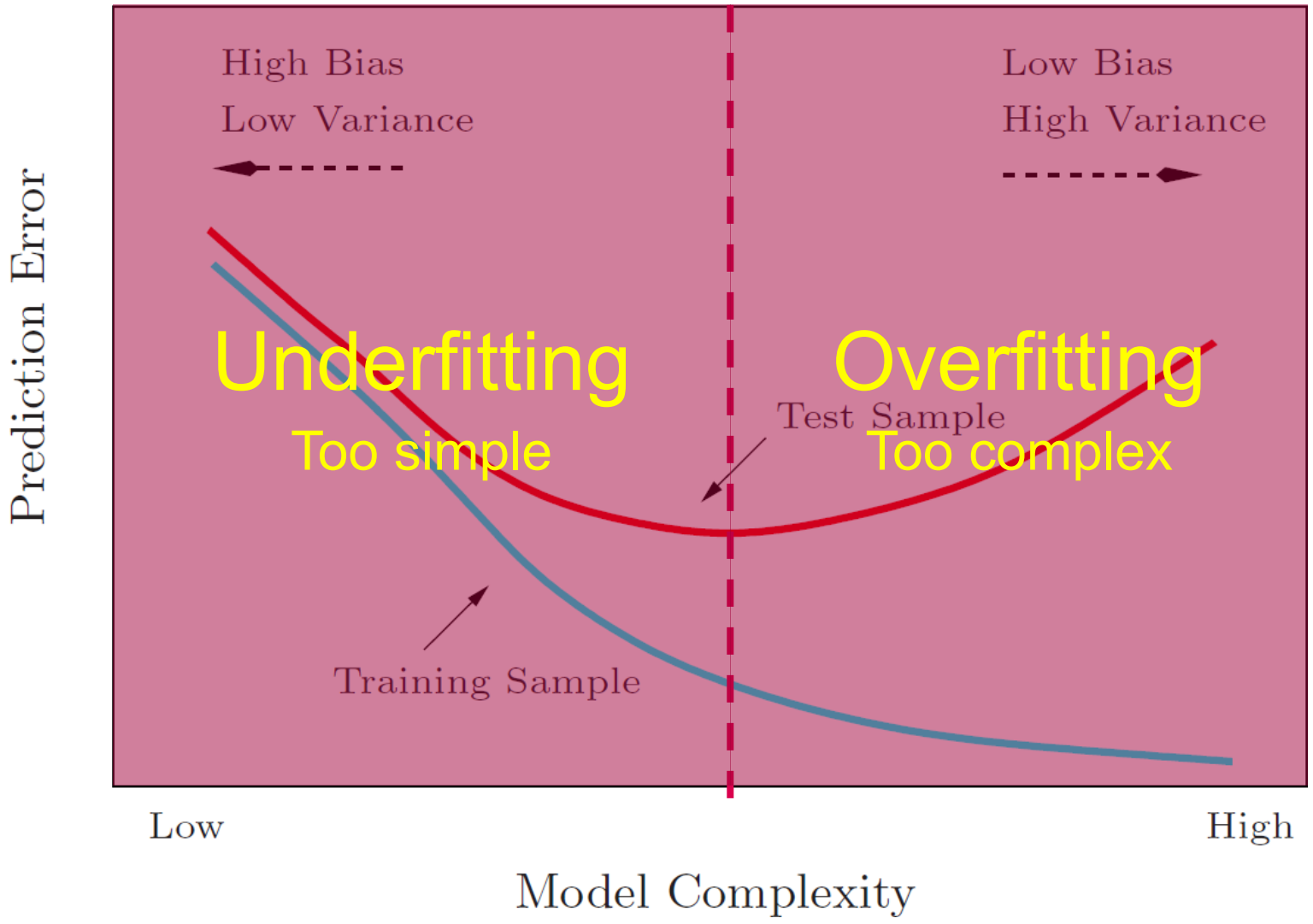
## What makes a good model?

It may happen that the models we build are underfitted or overfitted



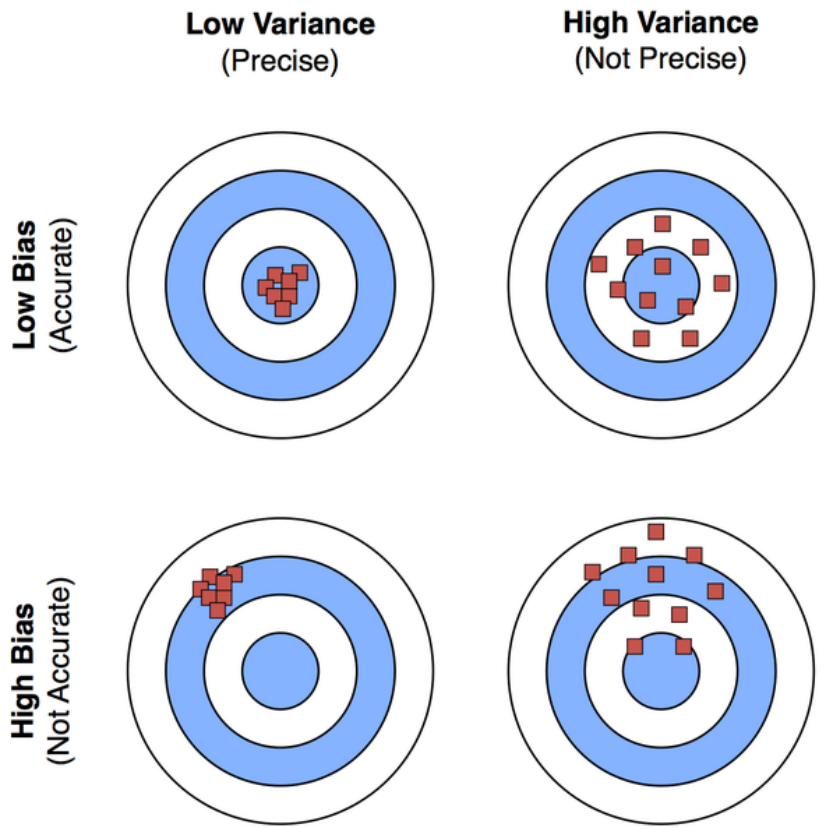
## How do we measure our model's performance?

When we have a performance measure, we can compare different models and their complexity



# How do we measure our model's performance?

## Variance-Bias Trade-off



# What about small data sets?

## Cross-validation

# What about large data sets?

## Curse of dimensionality

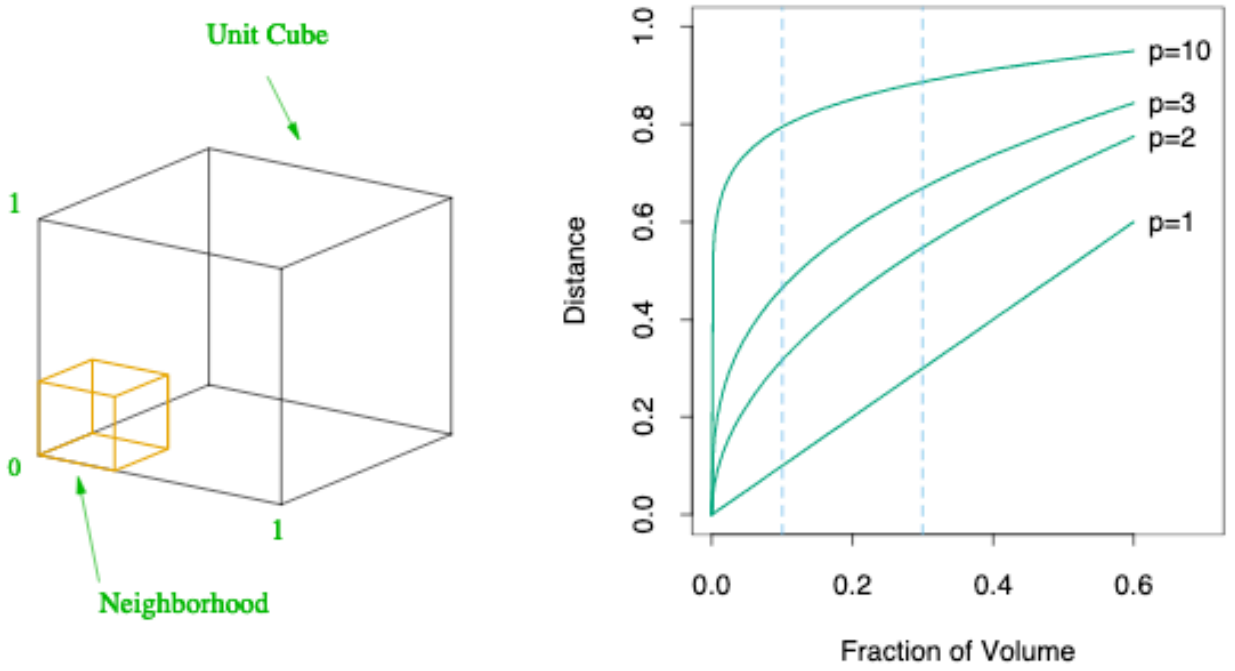
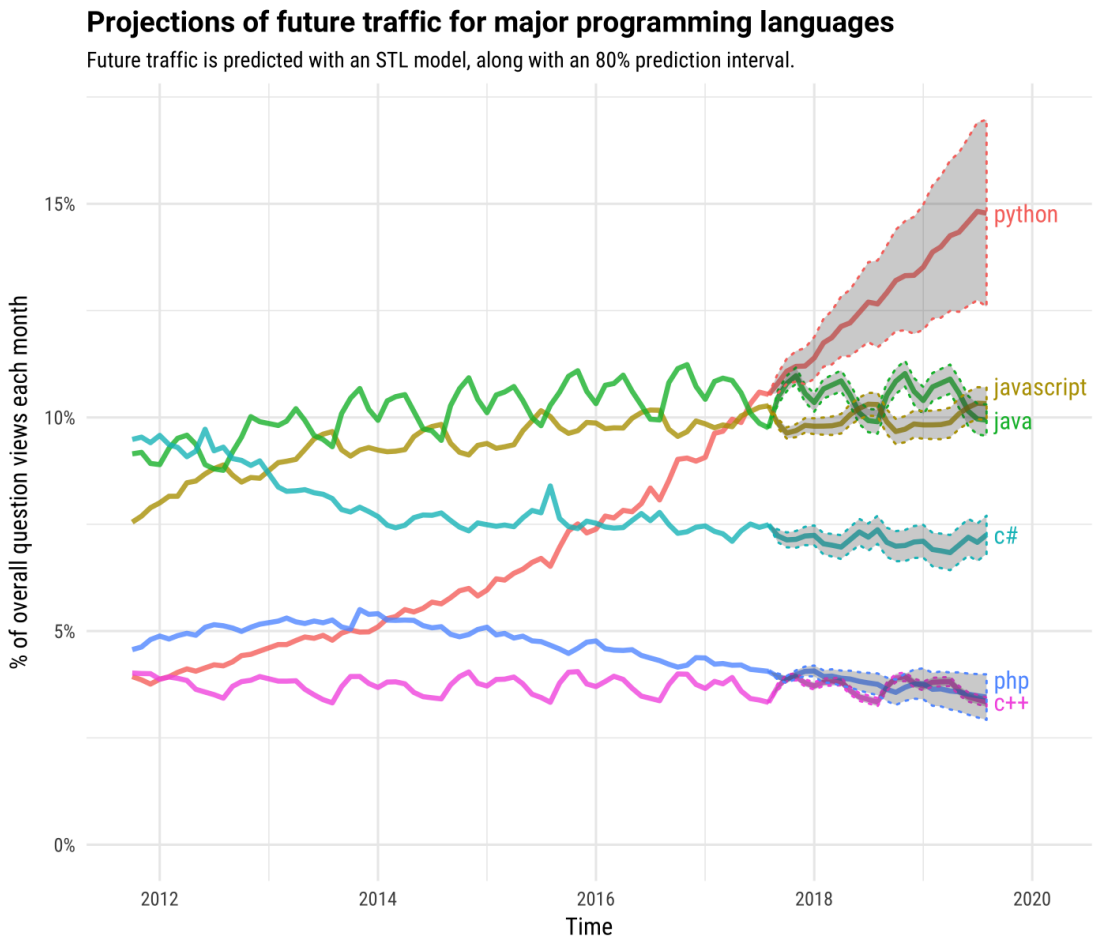



**FIGURE 2.6.** *The curse of dimensionality is well illustrated by a subcubical neighborhood for uniform data in a unit cube. The figure on the right shows the side-length of the subcube needed to capture a fraction r of the volume of the data, for different dimensions p. In ten dimensions we need to cover* 80% *of the range of each coordinate to capture* 10% *of the data.*

# Practice

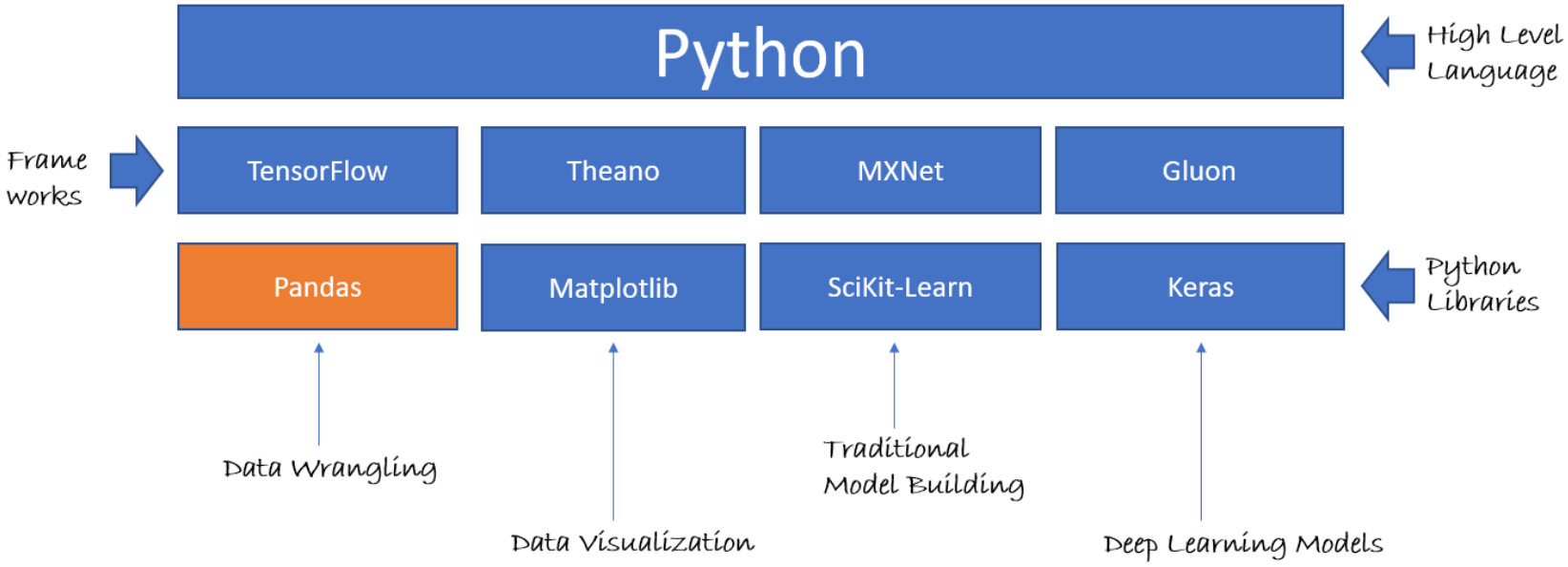
# Why Python




Source: Stack Overflow
https://medium.com/@UdacityINDIA/why-use-python-for-machine-learning-e4b0b4457a77

# Python ecosystem

Machine learning is a **method of data analysis** that automates **analytical model building**. In the applied space most these models are built in a language called **Python.**



# Python learning path

# Packages, Packages everywhere !

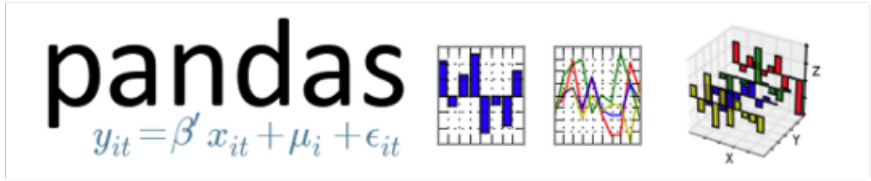
- Want to work with images — numpy, opencv, scikit
- Want to work in text — nltk, numpy, scikit
- Want to work in audio — librosa
- Want to solve **machine learning problem** — pandas, scikit
- Want to see the data clearly — matplotlib, seaborn, scikit
- Want to use **deep learning** — tensorflow, pytorch
- Want to do scientific computing — scipy
- Want to integrate web applications — Django
- Want to take a shower …. Well

# Machine Learning Landscape

## Python ecosystem



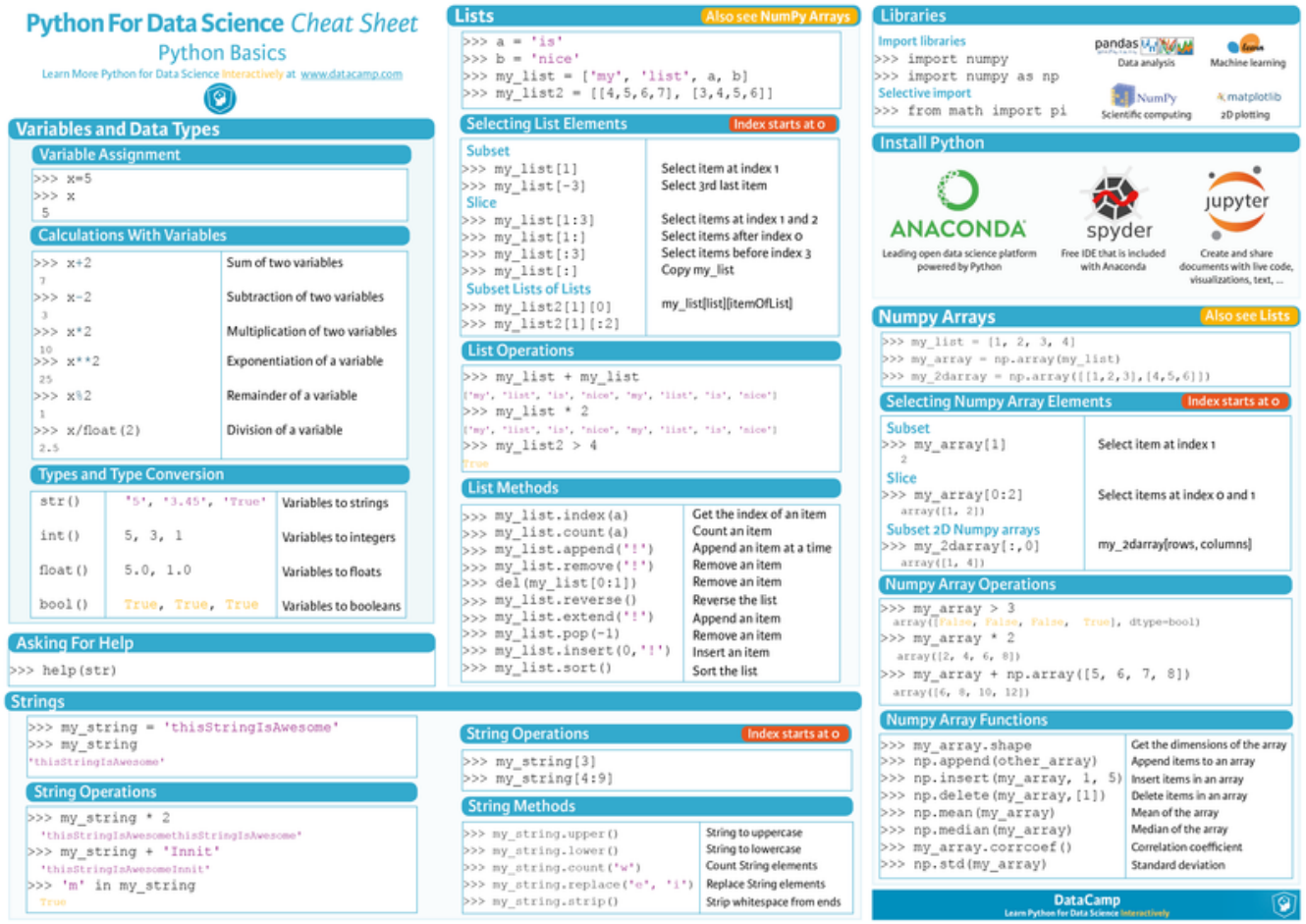






https://hackernoon.com/top-8-python-libraries-for-machine-learning-and-artificial-intelligence-y08id3031

# Cheat Sheets



## Python For Data Science *Cheat Sheet*

### Python Basics

Learn More Python for Data Science Interactively at www.datacamp.com

#### Variables and Data Types

**Variable Assignment**

```
>>> x=5
>>> x
5
```

**Calculations With Variables**

| | |
|---|---|
| `>>> x+2` → `7` | Sum of two variables |
| `>>> x-2` → `3` | Subtraction of two variables |
| `>>> x*2` → `10` | Multiplication of two variables |
| `>>> x**2` → `25` | Exponentiation of a variable |
| `>>> x%2` → `1` | Remainder of a variable |
| `>>> x/float(2)` → `2.5` | Division of a variable |

**Types and Type Conversion**

| | | |
|---|---|---|
| `str()` | `'5', '3.45', 'True'` | Variables to strings |
| `int()` | `5, 3, 1` | Variables to integers |
| `float()` | `5.0, 1.0` | Variables to floats |
| `bool()` | `True, True, True` | Variables to booleans |

#### Asking For Help

```
>>> help(str)
```

#### Lists

Also see NumPy Arrays

```
>>> a = 'is'
>>> b = 'nice'
>>> my_list = ['my', 'list', a, b]
>>> my_list2 = [[4,5,6,7], [3,4,5,6]]
```

**Selecting List Elements** — Index starts at 0

| | |
|---|---|
| **Subset** | |
| `>>> my_list[1]` | Select item at index 1 |
| `>>> my_list[-3]` | Select 3rd last item |
| **Slice** | |
| `>>> my_list[1:3]` | Select items at index 1 and 2 |
| `>>> my_list[1:]` | Select items after index 0 |
| `>>> my_list[:3]` | Select items before index 3 |
| `>>> my_list[:]` | Copy my_list |
| **Subset Lists of Lists** | |
| `>>> my_list2[1][0]` | my_list[list][itemOfList] |
| `>>> my_list2[1][:2]` | |

**List Operations**

```
>>> my_list + my_list
['my', 'list', 'is', 'nice', 'my', 'list', 'is', 'nice']
>>> my_list * 2
['my', 'list', 'is', 'nice', 'my', 'list', 'is', 'nice']
>>> my_list2 > 4
True
```

**List Methods**

| | |
|---|---|
| `>>> my_list.index(a)` | Get the index of an item |
| `>>> my_list.count(a)` | Count an item |
| `>>> my_list.append('!')` | Append an item at a time |
| `>>> my_list.remove('!')` | Remove an item |
| `>>> del(my_list[0:1])` | Remove an item |
| `>>> my_list.reverse()` | Reverse the list |
| `>>> my_list.extend('!')` | Append an item |
| `>>> my_list.pop(-1)` | Remove an item |
| `>>> my_list.insert(0,'!')` | Insert an item |
| `>>> my_list.sort()` | Sort the list |

#### Libraries

**Import libraries**

```
>>> import numpy
>>> import numpy as np
```

**Selective import**

```
>>> from math import pi
```

pandas — Data analysis; Machine learning; NumPy — Scientific computing; matplotlib — 2D plotting

#### Install Python

ANACONDA — Leading open data science platform powered by Python

spyder — Free IDE that is included with Anaconda

jupyter — Create and share documents with live code, visualizations, text, ...

#### Numpy Arrays

Also see Lists

```
>>> my_list = [1, 2, 3, 4]
>>> my_array = np.array(my_list)
>>> my_2darray = np.array([[1,2,3],[4,5,6]])
```

**Selecting Numpy Array Elements** — Index starts at 0

| | |
|---|---|
| **Subset** | |
| `>>> my_array[1]` → `2` | Select item at index 1 |
| **Slice** | |
| `>>> my_array[0:2]` → `array([1, 2])` | Select items at index 0 and 1 |
| **Subset 2D Numpy arrays** | |
| `>>> my_2darray[:,0]` → `array([1, 4])` | my_2darray[rows, columns] |

**Numpy Array Operations**

```
>>> my_array > 3
array([False, False, False,  True], dtype=bool)
>>> my_array * 2
array([2, 4, 6, 8])
>>> my_array + np.array([5, 6, 7, 8])
array([6, 8, 10, 12])
```

**Numpy Array Functions**

| | |
|---|---|
| `>>> my_array.shape` | Get the dimensions of the array |
| `>>> np.append(other_array)` | Append items to an array |
| `>>> np.insert(my_array, 1, 5)` | Insert items in an array |
| `>>> np.delete(my_array,[1])` | Delete items in an array |
| `>>> np.mean(my_array)` | Mean of the array |
| `>>> np.median(my_array)` | Median of the array |
| `>>> my_array.corrcoef()` | Correlation coefficient |
| `>>> np.std(my_array)` | Standard deviation |

DataCamp — Learn Python for Data Science Interactively

#### Strings

```
>>> my_string = 'thisStringIsAwesome'
>>> my_string
'thisStringIsAwesome'
```

**String Operations**

```
>>> my_string * 2
'thisStringIsAwesomethisStringIsAwesome'
>>> my_string + 'Innit'
'thisStringIsAwesomeInnit'
>>> 'm' in my_string
True
```

**String Operations** — Index starts at 0

```
>>> my_string[3]
>>> my_string[4:9]
```

**String Methods**

| | |
|---|---|
| `>>> my_string.upper()` | String to uppercase |
| `>>> my_string.lower()` | String to lowercase |
| `>>> my_string.count('w')` | Count String elements |
| `>>> my_string.replace('e', 'i')` | Replace String elements |
| `>>> my_string.strip()` | Strip whitespace from ends |

# Numpy Cheat Sheet

## Python For Data Science *Cheat Sheet*

### NumPy Basics

Learn Python for Data Science Interactively at www.DataCamp.com

### NumPy

The **NumPy** library is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays.

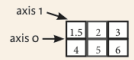
Use the following import convention:

```
>>> import numpy as np
```

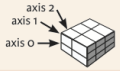
### NumPy Arrays

1D array


2D array




3D array




### Creating Arrays

```
>>> a = np.array([1,2,3])
>>> b = np.array([(1.5,2,3), (4,5,6)], dtype = float)
>>> c = np.array([[(1.5,2,3), (4,5,6)], [(3,2,1), (4,5,6)]],
                 dtype = float)
```

### Initial Placeholders

```
>>> np.zeros((3,4))
>>> np.ones((2,3,4),dtype=np.int16)
>>> d = np.arange(10,25,5)

>>> np.linspace(0,2,9)

>>> e = np.full((2,2),7)
>>> f = np.eye(2)
>>> np.random.random((2,2))
>>> np.empty((3,2))
```

Create an array of zeros
Create an array of ones
Create an array of evenly spaced values (step value)
Create an array of evenly spaced values (number of samples)
Create a constant array
Create a 2X2 identity matrix
Create an array with random values
Create an empty array

### I/O

#### Saving & Loading On Disk

```
>>> np.save('my_array', a)
>>> np.savez('array.npz', a, b)
>>> np.load('my_array.npy')
```

#### Saving & Loading Text Files

```
>>> np.loadtxt("myfile.txt")
>>> np.genfromtxt("my_file.csv", delimiter=',')
>>> np.savetxt("myarray.txt", a, delimiter=" ")
```

### Data Types

```
>>> np.int64
>>> np.float32
>>> np.complex
>>> np.bool
>>> np.object
>>> np.string_
>>> np.unicode_
```

Signed 64-bit integer types
Standard double-precision floating point
Complex numbers represented by 128 floats
Boolean type storing TRUE and FALSE values
Python object type
Fixed-length string type
Fixed-length unicode type

### Inspecting Your Array

```
>>> a.shape
>>> len(a)
>>> b.ndim
>>> e.size
>>> b.dtype
>>> b.dtype.name
>>> b.astype(int)
```

Array dimensions
Length of array
Number of array dimensions
Number of array elements
Data type of array elements
Name of data type
Convert an array to a different type

### Asking For Help

```
>>> np.info(np.ndarray.dtype)
```

### Array Mathematics

#### Arithmetic Operations

```
>>> g = a - b
 array([[-0.5,  0. ,  0. ],
        [-3. , -3. , -3. ]])
>>> np.subtract(a,b)
>>> b + a
 array([[ 2.5,  4. ,  6. ],
        [ 5. ,  7. ,  9. ]])
>>> np.add(b,a)
>>> a / b
 array([[ 0.66666667,  1.        ,  1.        ],
        [ 0.25      ,  0.4       ,  0.5       ]])
>>> np.divide(a,b)
>>> a * b
 array([[  1.5,   4. ,   9. ],
        [  4. ,  10. ,  18. ]])
>>> np.multiply(a,b)
>>> np.exp(b)
>>> np.sqrt(b)
>>> np.sin(a)
>>> np.cos(b)
>>> np.log(a)
>>> e.dot(f)
 array([[ 7.,  7.],
        [ 7.,  7.]])
```

Subtraction
Subtraction
Addition
Addition
Division
Division
Multiplication
Multiplication
Exponentiation
Square root
Print sines of an array
Element-wise cosine
Element-wise natural logarithm
Dot product

#### Comparison

```
>>> a == b
 array([[False,  True,  True],
        [False, False, False]], dtype=bool)
>>> a < 2
 array([True, False, False], dtype=bool)
>>> np.array_equal(a, b)
```

Element-wise comparison
Element-wise comparison
Array-wise comparison

#### Aggregate Functions

```
>>> a.sum()
>>> a.min()
>>> b.max(axis=0)
>>> b.cumsum(axis=1)
>>> a.mean()
>>> b.median()
>>> a.corrcoef()
>>> np.std(b)
```

Array-wise sum
Array-wise minimum value
Maximum value of an array row
Cumulative sum of the elements
Mean
Median
Correlation coefficient
Standard deviation

### Copying Arrays

```
>>> h = a.view()
>>> np.copy(a)
>>> h = a.copy()
```

Create a view of the array with the same data
Create a copy of the array
Create a deep copy of the array

### Sorting Arrays

```
>>> a.sort()
>>> c.sort(axis=0)
```

Sort an array
Sort the elements of an array's axis

### Subsetting, Slicing, Indexing

Also see Lists

#### Subsetting

```
>>> a[2]
 3
>>> b[1,2]
 6.0
```

Select the element at the 2nd index
Select the element at row 0 column 2 (equivalent to b[1][2])

#### Slicing

```
>>> a[0:2]
 array([1, 2])
>>> b[0:2,1]
 array([ 2. ,  5. ])
>>> b[:1]
 array([[1.5, 2., 3.]])
>>> c[1,...]
 array([[[ 3.,  2.,  1.],
         [ 4.,  5.,  6.]]])
>>> a[ : :-1]
 array([3, 2, 1])
```

Select items at index 0 and 1
Select items at rows 0 and 1 in column 1
Select all items at row 0 (equivalent to b[0:1, :])
Same as [1,:,:]
Reversed array a

#### Boolean Indexing

```
>>> a[a<2]
 array([1])
```

Select elements from a less than 2

#### Fancy Indexing

```
>>> b[[1, 0, 1, 0],[0, 1, 2, 0]]
 array([ 4. ,  2. ,  6. ,  1.5])
>>> b[[1, 0, 1, 0]][:,[0,1,2,0]]
 array([[ 4. ,  5. ,  6. ,  4. ],
        [ 1.5,  2. ,  3. ,  1.5],
        [ 4. ,  5. ,  6. ,  4. ],
        [ 1.5,  2. ,  3. ,  1.5]])
```

Select elements (1,0),(0,1),(1,2) and (0,0)
Select a subset of the matrix's rows and columns

### Array Manipulation

#### Transposing Array

```
>>> i = np.transpose(b)
>>> i.T
```

Permute array dimensions
Permute array dimensions

#### Changing Array Shape

```
>>> b.ravel()
>>> g.reshape(3,-2)
```

Flatten the array
Reshape, but don't change data

#### Adding/Removing Elements

```
>>> h.resize((2,6))
>>> np.append(h,g)
>>> np.insert(a, 1, 5)
>>> np.delete(a,[1])
```

Return a new array with shape (2,6)
Append items to an array
Insert items in an array
Delete items from an array

#### Combining Arrays

```
>>> np.concatenate((a,d),axis=0)
 array([ 1,  2,  3, 10, 15, 20])
>>> np.vstack((a,b))
 array([[ 1. ,  2. ,  3. ],
        [ 1.5,  2. ,  3. ],
        [ 4. ,  5. ,  6. ]])
>>> np.r_[e,f]
>>> np.hstack((e,f))
 array([[ 7.,  7.,  1.,  0.],
        [ 7.,  7.,  0.,  1.]])
>>> np.column_stack((a,d))
 array([[ 1, 10],
        [ 2, 15],
        [ 3, 20]])
>>> np.c_[a,d]
```

Concatenate arrays
Stack arrays vertically (row-wise)
Stack arrays vertically (row-wise)
Stack arrays horizontally (column-wise)
Create stacked column-wise arrays
Create stacked column-wise arrays

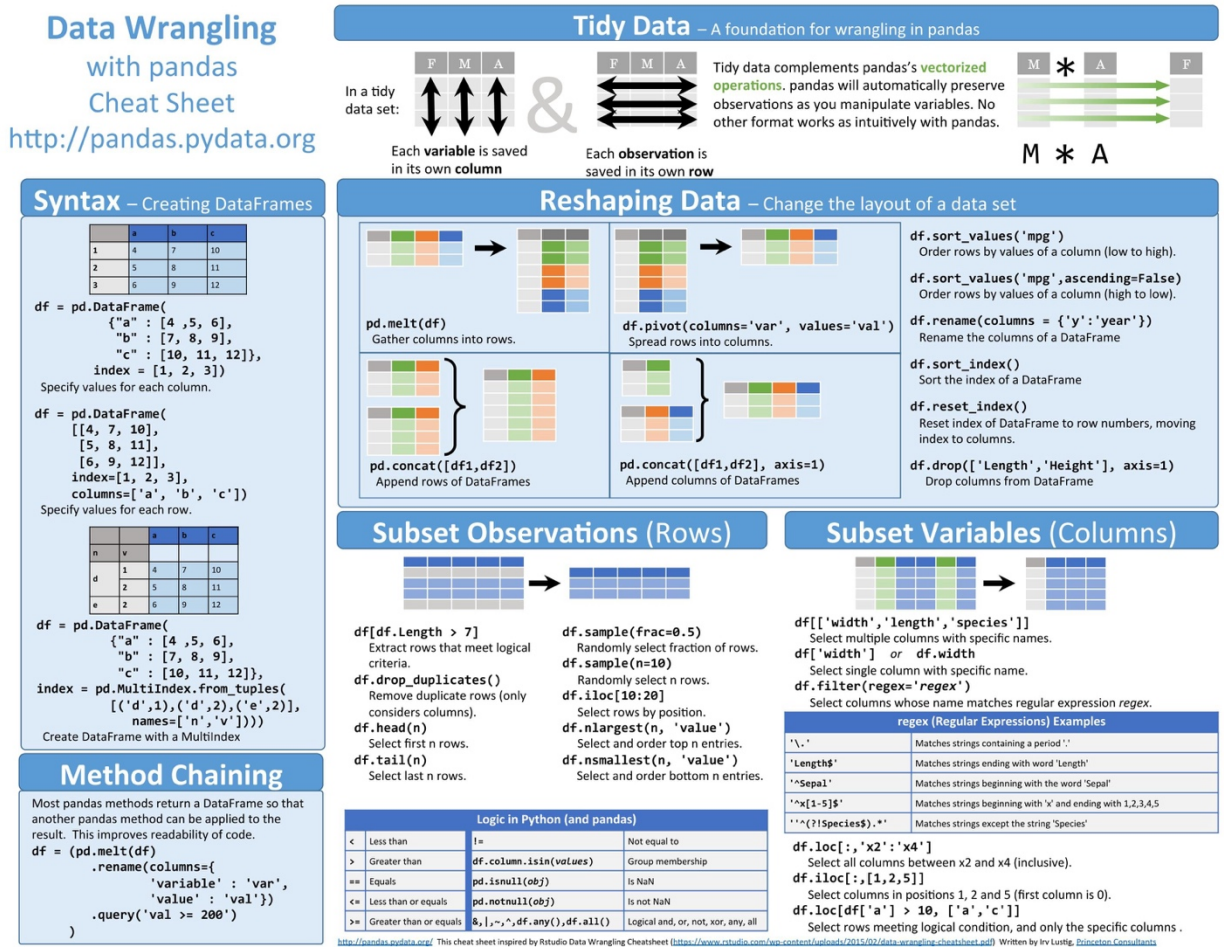
#### Splitting Arrays

```
>>> np.hsplit(a,3)
 [array([1]),array([2]),array([3])]
>>> np.vsplit(c,2)
[array([[[ 1.5,  2. ,  1. ],
         [ 4. ,  5. ,  6. ]]]),
 array([[[ 3.,  2.,  3.],
         [ 4.,  5.,  6.]]])]
```

Split the array horizontally at the 3rd index
Split the array vertically at the 2nd index

DataCamp
Learn Python for Data Science Interactively

# Pandas Cheat Sheet



## Data Wrangling with pandas Cheat Sheet

http://pandas.pydata.org

### Tidy Data – A foundation for wrangling in pandas

In a tidy data set:

Each **variable** is saved in its own **column**

Each **observation** is saved in its own **row**

Tidy data complements pandas's **vectorized operations**. pandas will automatically preserve observations as you manipulate variables. No other format works as intuitively with pandas.

M * A

### Syntax – Creating DataFrames

| | a | b | c |
|---|---|---|---|
| 1 | 4 | 7 | 10 |
| 2 | 5 | 8 | 11 |
| 3 | 6 | 9 | 12 |

```
df = pd.DataFrame(
        {"a" : [4 ,5, 6],
         "b" : [7, 8, 9],
         "c" : [10, 11, 12]},
        index = [1, 2, 3])
```

Specify values for each column.

```
df = pd.DataFrame(
    [[4, 7, 10],
     [5, 8, 11],
     [6, 9, 12]],
    index=[1, 2, 3],
    columns=['a', 'b', 'c'])
```

Specify values for each row.

| | | a | b | c |
|---|---|---|---|---|
| n | v | | | |
| d | 1 | 4 | 7 | 10 |
| | 2 | 5 | 8 | 11 |
| e | 2 | 6 | 9 | 12 |

```
df = pd.DataFrame(
        {"a" : [4 ,5, 6],
         "b" : [7, 8, 9],
         "c" : [10, 11, 12]},
index = pd.MultiIndex.from_tuples(
        [('d',1),('d',2),('e',2)],
         names=['n','v'])))
```

Create DataFrame with a MultiIndex

### Method Chaining

Most pandas methods return a DataFrame so that another pandas method can be applied to the result. This improves readability of code.

```
df = (pd.melt(df)
        .rename(columns={
                'variable' : 'var',
                'value' : 'val'})
        .query('val >= 200')
    )
```

### Reshaping Data – Change the layout of a data set

**pd.melt(df)**
Gather columns into rows.

**df.pivot(columns='var', values='val')**
Spread rows into columns.

**pd.concat([df1,df2])**
Append rows of DataFrames

**pd.concat([df1,df2], axis=1)**
Append columns of DataFrames

**df.sort_values('mpg')**
Order rows by values of a column (low to high).

**df.sort_values('mpg',ascending=False)**
Order rows by values of a column (high to low).

**df.rename(columns = {'y':'year'})**
Rename the columns of a DataFrame

**df.sort_index()**
Sort the index of a DataFrame

**df.reset_index()**
Reset index of DataFrame to row numbers, moving index to columns.

**df.drop(['Length','Height'], axis=1)**
Drop columns from DataFrame

### Subset Observations (Rows)

**df[df.Length > 7]**
Extract rows that meet logical criteria.

**df.drop_duplicates()**
Remove duplicate rows (only considers columns).

**df.head(n)**
Select first n rows.

**df.tail(n)**
Select last n rows.

**df.sample(frac=0.5)**
Randomly select fraction of rows.

**df.sample(n=10)**
Randomly select n rows.

**df.iloc[10:20]**
Select rows by position.

**df.nlargest(n, 'value')**
Select and order top n entries.

**df.nsmallest(n, 'value')**
Select and order bottom n entries.

### Subset Variables (Columns)

**df[['width','length','species']]**
Select multiple columns with specific names.

**df['width']** *or* **df.width**
Select single column with specific name.

**df.filter(regex='*regex*')**
Select columns whose name matches regular expression *regex*.

**regex (Regular Expressions) Examples**

| | |
|---|---|
| '\.' | Matches strings containing a period '.' |
| 'Length$' | Matches strings ending with word 'Length' |
| '^Sepal' | Matches strings beginning with the word 'Sepal' |
| '^x[1-5]$' | Matches strings beginning with 'x' and ending with 1,2,3,4,5 |
| '^(?!Species$).*' | Matches strings except the string 'Species' |

**df.loc[:,'x2':'x4']**
Select all columns between x2 and x4 (inclusive).

**df.iloc[:,[1,2,5]]**
Select columns in positions 1, 2 and 5 (first column is 0).

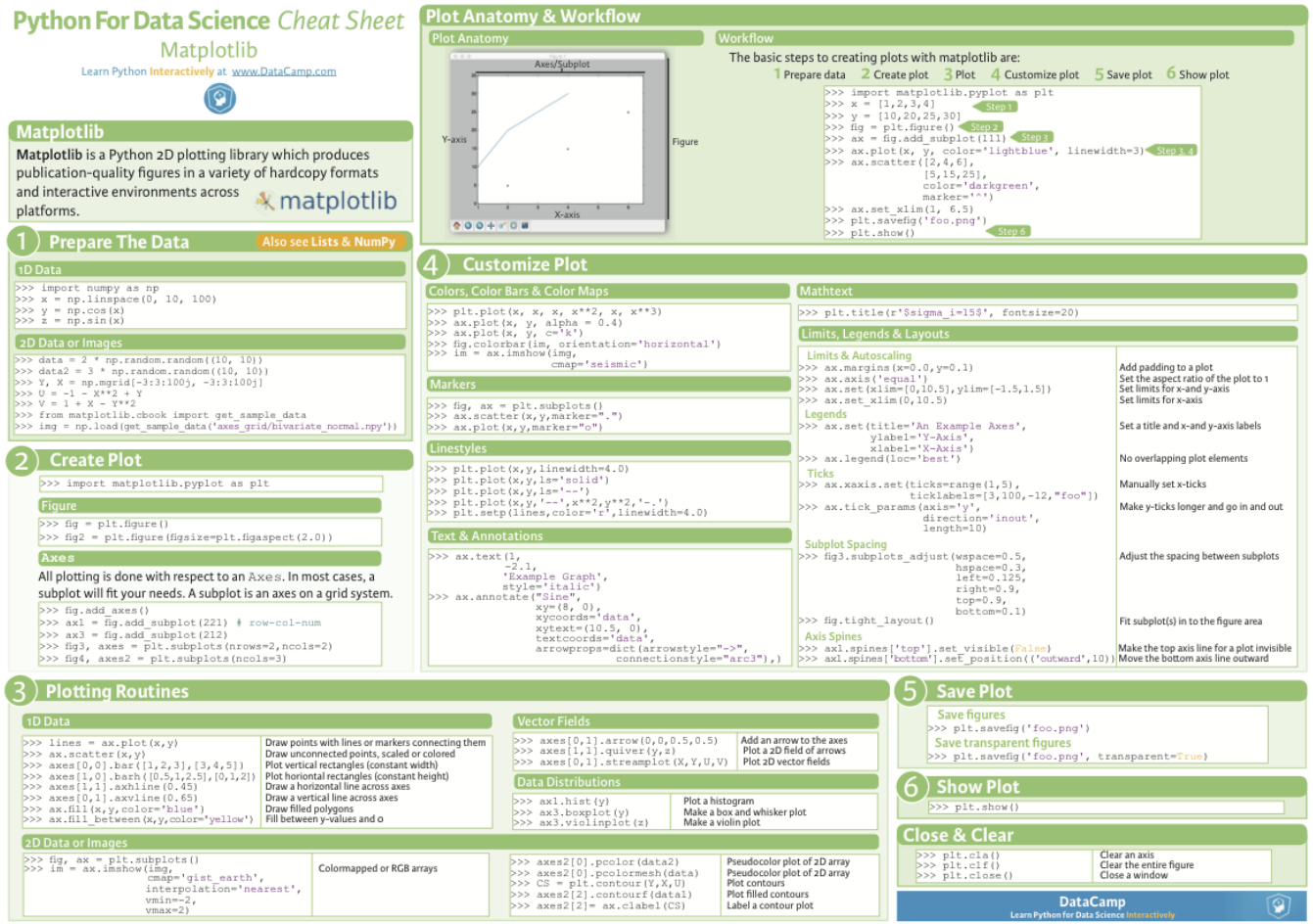
**df.loc[df['a'] > 10, ['a','c']]**
Select rows meeting logical condition, and only the specific columns .

**Logic in Python (and pandas)**

| | | | |
|---|---|---|---|
| < | Less than | != | Not equal to |
| > | Greater than | **df.column.isin(*values*)** | Group membership |
| == | Equals | **pd.isnull(*obj*)** | Is NaN |
| <= | Less than or equals | **pd.notnull(*obj*)** | Is not NaN |
| >= | Greater than or equals | **&,\|,~,^,df.any(),df.all()** | Logical and, or, not, xor, any, all |

http://pandas.pydata.org/ This cheat sheet inspired by Rstudio Data Wrangling Cheatsheet (https://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf) Written by Irv Lustig, Princeton Consultants

# Matplotlib Cheat Sheet



## Python For Data Science *Cheat Sheet*

Matplotlib

Learn Python Interactively at www.DataCamp.com

### Matplotlib

**Matplotlib** is a Python 2D plotting library which produces publication-quality figures in a variety of hardcopy formats and interactive environments across platforms.

matplotlib

### 1 Prepare The Data

Also see Lists & NumPy

#### 1D Data

```
>>> import numpy as np
>>> x = np.linspace(0, 10, 100)
>>> y = np.cos(x)
>>> z = np.sin(x)
```

#### 2D Data or Images

```
>>> data = 2 * np.random.random((10, 10))
>>> data2 = 3 * np.random.random((10, 10))
>>> Y, X = np.mgrid[-3:3:100j, -3:3:100j]
>>> U = -1 - X**2 + Y
>>> V = 1 + X - Y**2
>>> from matplotlib.cbook import get_sample_data
>>> img = np.load(get_sample_data('axes_grid/bivariate_normal.npy'))
```

### 2 Create Plot

```
>>> import matplotlib.pyplot as plt
```

#### Figure

```
>>> fig = plt.figure()
>>> fig2 = plt.figure(figsize=plt.figaspect(2.0))
```

#### Axes

All plotting is done with respect to an `Axes`. In most cases, a subplot will fit your needs. A subplot is an axes on a grid system.

```
>>> fig.add_axes()
>>> ax1 = fig.add_subplot(221) # row-col-num
>>> ax3 = fig.add_subplot(212)
>>> fig3, axes = plt.subplots(nrows=2,ncols=2)
>>> fig4, axes2 = plt.subplots(ncols=3)
```

### Plot Anatomy & Workflow

#### Plot Anatomy




#### Workflow

The basic steps to creating plots with matplotlib are:

1 Prepare data 2 Create plot 3 Plot 4 Customize plot 5 Save plot 6 Show plot

```
>>> import matplotlib.pyplot as plt
>>> x = [1,2,3,4]                    Step 1
>>> y = [10,20,25,30]
>>> fig = plt.figure()               Step 2
>>> ax = fig.add_subplot(111)        Step 3
>>> ax.plot(x, y, color='lightblue', linewidth=3)  Step 3, 4
>>> ax.scatter([2,4,6],
               [5,15,25],
               color='darkgreen',
               marker='^')
>>> ax.set_xlim(1, 6.5)
>>> plt.savefig('foo.png')
>>> plt.show()                       Step 6
```

### 4 Customize Plot

#### Colors, Color Bars & Color Maps

```
>>> plt.plot(x, x, x, x**2, x, x**3)
>>> ax.plot(x, y, alpha = 0.4)
>>> ax.plot(x, y, c='k')
>>> fig.colorbar(im, orientation='horizontal')
>>> im = ax.imshow(img,
                   cmap='seismic')
```

#### Markers

```
>>> fig, ax = plt.subplots()
>>> ax.scatter(x,y,marker=".")
>>> ax.plot(x,y,marker="o")
```

#### Linestyles

```
>>> plt.plot(x,y,linewidth=4.0)
>>> plt.plot(x,y,ls='solid')
>>> plt.plot(x,y,ls='--')
>>> plt.plot(x,y,'--',x**2,y**2,'-.')
>>> plt.setp(lines,color='r',linewidth=4.0)
```

#### Text & Annotations

```
>>> ax.text(1,
            -2.1,
            'Example Graph',
            style='italic')
>>> ax.annotate("Sine",
                xy=(8, 0),
                xycoords='data',
                xytext=(10.5, 0),
                textcoords='data',
                arrowprops=dict(arrowstyle="->",
                                connectionstyle="arc3"),)
```

#### Mathtext

```
>>> plt.title(r'$sigma_i=15$', fontsize=20)
```

#### Limits, Legends & Layouts

##### Limits & Autoscaling

```
>>> ax.margins(x=0.0,y=0.1)          Add padding to a plot
>>> ax.axis('equal')                 Set the aspect ratio of the plot to 1
>>> ax.set(xlim=[0,10.5],ylim=[-1.5,1.5])  Set limits for x-and y-axis
>>> ax.set_xlim(0,10.5)              Set limits for x-axis
```

##### Legends

```
>>> ax.set(title='An Example Axes',   Set a title and x-and y-axis labels
           ylabel='Y-Axis',
           xlabel='X-Axis')
>>> ax.legend(loc='best')            No overlapping plot elements
```

##### Ticks

```
>>> ax.xaxis.set(ticks=range(1,5),    Manually set x-ticks
                 ticklabels=[3,100,-12,"foo"])
>>> ax.tick_params(axis='y',          Make y-ticks longer and go in and out
                   direction='inout',
                   length=10)
```

##### Subplot Spacing

```
>>> fig3.subplots_adjust(wspace=0.5,  Adjust the spacing between subplots
                         hspace=0.3,
                         left=0.125,
                         right=0.9,
                         top=0.9,
                         bottom=0.1)
>>> fig.tight_layout()               Fit subplot(s) in to the figure area
```

##### Axis Spines

```
>>> ax1.spines['top'].set_visible(False)  Make the top axis line for a plot invisible
>>> ax1.spines['bottom'].set_position(('outward',10))  Move the bottom axis line outward
```

### 3 Plotting Routines

#### 1D Data

```
>>> lines = ax.plot(x,y)              Draw points with lines or markers connecting them
>>> ax.scatter(x,y)                   Draw unconnected points, scaled or colored
>>> axes[0,0].bar([1,2,3],[3,4,5])    Plot vertical rectangles (constant width)
>>> axes[1,0].barh([0.5,1,2.5],[0,1,2])  Plot horiontal rectangles (constant height)
>>> axes[1,1].axhline(0.45)           Draw a horizontal line across axes
>>> axes[0,1].axvline(0.65)           Draw a vertical line across axes
>>> ax.fill(x,y,color='blue')         Draw filled polygons
>>> ax.fill_between(x,y,color='yellow')  Fill between y-values and o
```

#### Vector Fields

```
>>> axes[0,1].arrow(0,0,0.5,0.5)      Add an arrow to the axes
>>> axes[1,1].quiver(y,z)             Plot a 2D field of arrows
>>> axes[0,1].streamplot(X,Y,U,V)     Plot 2D vector fields
```

#### Data Distributions

```
>>> ax1.hist(y)                       Plot a histogram
>>> ax3.boxplot(y)                    Make a box and whisker plot
>>> ax3.violinplot(z)                 Make a violin plot
```

#### 2D Data or Images

```
>>> fig, ax = plt.subplots()
>>> im = ax.imshow(img,               Colormapped or RGB arrays
                   cmap='gist_earth',
                   interpolation='nearest',
                   vmin=-2,
                   vmax=2)
```

```
>>> axes2[0].pcolor(data2)            Pseudocolor plot of 2D array
>>> axes2[0].pcolormesh(data)         Pseudocolor plot of 2D array
>>> CS = plt.contour(Y,X,U)           Plot contours
>>> axes2[2].contourf(data1)          Plot filled contours
>>> axes2[2]= ax.clabel(CS)           Label a contour plot
```

### 5 Save Plot

#### Save figures

```
>>> plt.savefig('foo.png')
```

#### Save transparent figures

```
>>> plt.savefig('foo.png', transparent=True)
```

### 6 Show Plot

```
>>> plt.show()
```

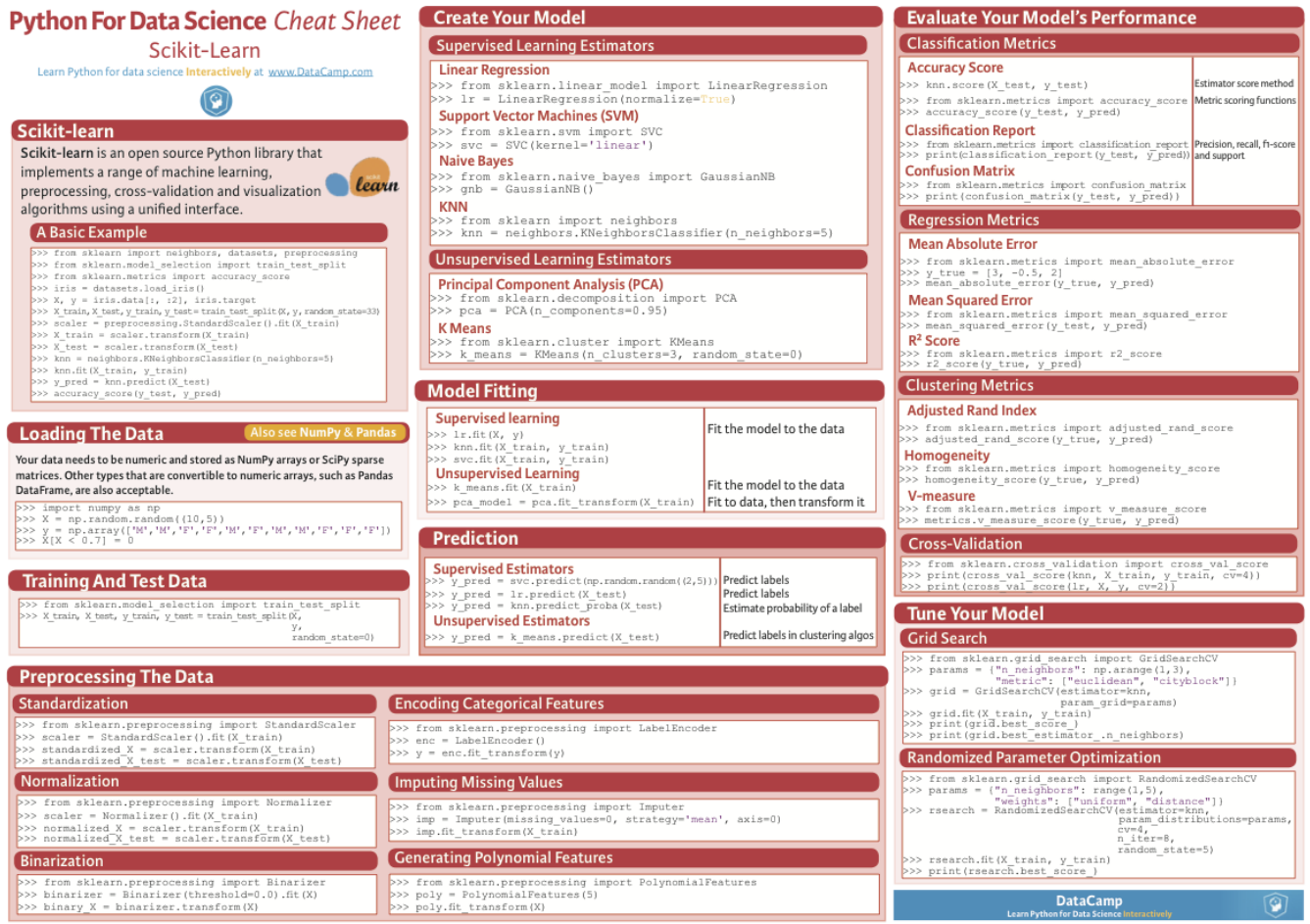
### Close & Clear

```
>>> plt.cla()                         Clear an axis
>>> plt.clf()                         Clear the entire figure
>>> plt.close()                       Close a window
```

DataCamp

Learn Python for Data Science Interactively

# Sscikit-Learn Cheat Sheet



## Python For Data Science *Cheat Sheet*

Scikit-Learn

Learn Python for data science Interactively at www.DataCamp.com

### Scikit-learn

**Scikit-learn** is an open source Python library that implements a range of machine learning, preprocessing, cross-validation and visualization algorithms using a unified interface.

#### A Basic Example

```
>>> from sklearn import neighbors, datasets, preprocessing
>>> from sklearn.model_selection import train_test_split
>>> from sklearn.metrics import accuracy_score
>>> iris = datasets.load_iris()
>>> X, y = iris.data[:, :2], iris.target
>>> X_train, X_test, y_train, y_test= train_test_split(X, y, random_state=33)
>>> scaler = preprocessing.StandardScaler().fit(X_train)
>>> X_train = scaler.transform(X_train)
>>> X_test = scaler.transform(X_test)
>>> knn = neighbors.KNeighborsClassifier(n_neighbors=5)
>>> knn.fit(X_train, y_train)
>>> y_pred = knn.predict(X_test)
>>> accuracy_score(y_test, y_pred)
```

### Loading The Data

Also see NumPy & Pandas

Your data needs to be numeric and stored as NumPy arrays or SciPy sparse matrices. Other types that are convertible to numeric arrays, such as Pandas DataFrame, are also acceptable.

```
>>> import numpy as np
>>> X = np.random.random((10,5))
>>> y = np.array(['M','M','F','F','M','F','M','M','F','F','F'])
>>> X[X < 0.7] = 0
```

### Training And Test Data

```
>>> from sklearn.model_selection import train_test_split
>>> X_train, X_test, y_train, y_test = train_test_split(X,
                                                        y,
                                                        random_state=0)
```

### Create Your Model

#### Supervised Learning Estimators

**Linear Regression**

```
>>> from sklearn.linear_model import LinearRegression
>>> lr = LinearRegression(normalize=True)
```

**Support Vector Machines (SVM)**

```
>>> from sklearn.svm import SVC
>>> svc = SVC(kernel='linear')
```

**Naive Bayes**

```
>>> from sklearn.naive_bayes import GaussianNB
>>> gnb = GaussianNB()
```

**KNN**

```
>>> from sklearn import neighbors
>>> knn = neighbors.KNeighborsClassifier(n_neighbors=5)
```

#### Unsupervised Learning Estimators

**Principal Component Analysis (PCA)**

```
>>> from sklearn.decomposition import PCA
>>> pca = PCA(n_components=0.95)
```

**K Means**

```
>>> from sklearn.cluster import KMeans
>>> k_means = KMeans(n_clusters=3, random_state=0)
```

### Model Fitting

| | |
|---|---|
| **Supervised learning** | |
| `>>> lr.fit(X, y)` | Fit the model to the data |
| `>>> knn.fit(X_train, y_train)` | |
| `>>> svc.fit(X_train, y_train)` | |
| **Unsupervised Learning** | |
| `>>> k_means.fit(X_train)` | Fit the model to the data |
| `>>> pca_model = pca.fit_transform(X_train)` | Fit to data, then transform it |

### Prediction

| | |
|---|---|
| **Supervised Estimators** | |
| `>>> y_pred = svc.predict(np.random.random((2,5)))` | Predict labels |
| `>>> y_pred = lr.predict(X_test)` | Predict labels |
| `>>> y_pred = knn.predict_proba(X_test)` | Estimate probability of a label |
| **Unsupervised Estimators** | |
| `>>> y_pred = k_means.predict(X_test)` | Predict labels in clustering algos |

### Preprocessing The Data

#### Standardization

```
>>> from sklearn.preprocessing import StandardScaler
>>> scaler = StandardScaler().fit(X_train)
>>> standardized_X = scaler.transform(X_train)
>>> standardized_X_test = scaler.transform(X_test)
```

#### Normalization

```
>>> from sklearn.preprocessing import Normalizer
>>> scaler = Normalizer().fit(X_train)
>>> normalized_X = scaler.transform(X_train)
>>> normalized_X_test = scaler.transform(X_test)
```

#### Binarization

```
>>> from sklearn.preprocessing import Binarizer
>>> binarizer = Binarizer(threshold=0.0).fit(X)
>>> binary_X = binarizer.transform(X)
```

#### Encoding Categorical Features

```
>>> from sklearn.preprocessing import LabelEncoder
>>> enc = LabelEncoder()
>>> y = enc.fit_transform(y)
```

#### Imputing Missing Values

```
>>> from sklearn.preprocessing import Imputer
>>> imp = Imputer(missing_values=0, strategy='mean', axis=0)
>>> imp.fit_transform(X_train)
```

#### Generating Polynomial Features

```
>>> from sklearn.preprocessing import PolynomialFeatures
>>> poly = PolynomialFeatures(5)
>>> poly.fit_transform(X)
```

### Evaluate Your Model's Performance

#### Classification Metrics

**Accuracy Score**

```
>>> knn.score(X_test, y_test)
```
Estimator score method

```
>>> from sklearn.metrics import accuracy_score
>>> accuracy_score(y_test, y_pred)
```
Metric scoring functions

**Classification Report**

```
>>> from sklearn.metrics import classification_report
>>> print(classification_report(y_test, y_pred))
```
Precision, recall, f1-score and support

**Confusion Matrix**

```
>>> from sklearn.metrics import confusion_matrix
>>> print(confusion_matrix(y_test, y_pred))
```

#### Regression Metrics

**Mean Absolute Error**

```
>>> from sklearn.metrics import mean_absolute_error
>>> y_true = [3, -0.5, 2]
>>> mean_absolute_error(y_true, y_pred)
```

**Mean Squared Error**

```
>>> from sklearn.metrics import mean_squared_error
>>> mean_squared_error(y_test, y_pred)
```

**$R^2$ Score**

```
>>> from sklearn.metrics import r2_score
>>> r2_score(y_true, y_pred)
```

#### Clustering Metrics

**Adjusted Rand Index**

```
>>> from sklearn.metrics import adjusted_rand_score
>>> adjusted_rand_score(y_true, y_pred)
```

**Homogeneity**

```
>>> from sklearn.metrics import homogeneity_score
>>> homogeneity_score(y_true, y_pred)
```

**V-measure**

```
>>> from sklearn.metrics import v_measure_score
>>> metrics.v_measure_score(y_true, y_pred)
```

#### Cross-Validation

```
>>> from sklearn.cross_validation import cross_val_score
>>> print(cross_val_score(knn, X_train, y_train, cv=4))
>>> print(cross_val_score(lr, X, y, cv=2))
```

### Tune Your Model

#### Grid Search

```
>>> from sklearn.grid_search import GridSearchCV
>>> params = {"n_neighbors": np.arange(1,3),
              "metric": ["euclidean", "cityblock"]}
>>> grid = GridSearchCV(estimator=knn,
                        param_grid=params)
>>> grid.fit(X_train, y_train)
>>> print(grid.best_score_)
>>> print(grid.best_estimator_.n_neighbors)
```

#### Randomized Parameter Optimization

```
>>> from sklearn.grid_search import RandomizedSearchCV
>>> params = {"n_neighbors": range(1,5),
              "weights": ["uniform", "distance"]}
>>> rsearch = RandomizedSearchCV(estimator=knn,
                                 param_distributions=params,
                                 cv=4,
                                 n_iter=8,
                                 random_state=5)
>>> rsearch.fit(X_train, y_train)
>>> print(rsearch.best_score_)
```

DataCamp
Learn Python for Data Science Interactively

# Scikit-learn Design

https://arxiv.org/abs/1309.0238

Scikit-Learn's API is remarkably well designed. These are the main design principles:

**Consistency**: All objects share a consistent and simple interface:

- **Estimators** Any object that can estimate some parameters based on a dataset is called an *estimator* (e.g., an imputer is an estimator). The estimation itself is performed by the fit() method, and it takes only a dataset as a parameter (or two for supervised learning algorithms; the second dataset contains the labels). Any other parameter needed to guide the estimation process is considered a hyperparameter (such as an imputer's strategy), and it must be set as an instance variable (generally via a constructor parameter).
- **Transformers** Some estimators (such as an imputer) can also transform a dataset; these are called *transformers*. Once again, the API is simple: the transformation is performed by the transform() method with the dataset to transform as a parameter. It returns the transformed dataset. This transformation generally relies on the learned parameters, as is the case for an imputer. All transformers also have a convenience method called fit_transform() that is equivalent to calling fit() and then transform() (but sometimes fit_transform() is optimized and runs much faster).
- **Predictors** Finally, some estimators, given a dataset, are capable of making predictions; they are called *predictors*. For example, the LinearRegression model in the previous chapter was a predictor: given a country's GDP per capita, it predicted life satisfaction. A predictor has a predict() method that takes a dataset of new instances and returns a dataset of corresponding predictions. It also has a score() method that measures the quality of the predictions, given a test set (and the corresponding labels, in the case of supervised learning algorithms).[18]

**Inspection:** All the estimator's hyperparameters are accessible directly via public instance variables (e.g., imputer.strategy), and all the estimator's learned parameters are accessible via public instance variables with an underscore suffix (e.g., imputer.statistics_).

**Nonproliferation** of classes Datasets are represented as NumPy arrays or SciPy sparse matrices, instead of homemade classes. Hyperparameters are just regular Python strings or numbers.

**Composition** Existing building blocks are reused as much as possible. For example, it is easy to create a Pipeline estimator from an arbitrary sequence of transformers followed by a final estimator, as we will see.

**Sensible defaults** Scikit-Learn provides reasonable default values for most parameters, making it easy to quickly create a baseline working system.

# Where to go from here

1. Learn the language
2. Learn the tools
3. Work on projects!

**Resources**

- **Machine Learning Mastery**
- **Hands-On Machine Learning with Scikit-Learn and TensorFlow**
- **The Complete Python Course for Machine Learning Engineers**
- **Cheat Sheet of Machine Learning and Python (and Math) Cheat Sheets**

# Questions

Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition by Aurélien Géron, Publisher: O'Reilly Media, Inc. Release Date: September 201

1. How would you define Machine Learning?
2. Can you name four types of problems where it shines?
3. What is a labeled training set?
4. What are the two most common supervised tasks?
5. Can you name four common unsupervised tasks?
6. What type of Machine Learning algorithm would you use to allow a robot to walk in various unknown terrains?
7. What type of algorithm would you use to segment your customers into multiple groups?
8. Would you frame the problem of spam detection as a supervised learning problem or an unsupervised learning problem?
9. What is an online learning system?
10. What is out-of-core learning?
11. What type of learning algorithm relies on a similarity measure to make predictions?
12. What is the difference between a model parameter and a learning algorithm’s hyperparameter?
13. What do model-based learning algorithms search for? What is the most common strategy they use to succeed? How do they make predictions?
14. Can you name four of the main challenges in Machine Learning?
15. If your model performs great on the training data but generalizes poorly to new instances, what is happening? Can you name three possible solutions?
16. What is a test set, and why would you want to use it?
17. What is the purpose of a validation set?
18. What is the train-dev set, when do you need it, and how do you use it?
19. What can go wrong if you tune hyperparameters using the test set?

# Answers

1. Machine Learning is about building systems that can learn from data. Learning means getting better at some task, given some performance measure.
2. Machine Learning is great for complex problems for which we have no algorithmic solution, to replace long lists of hand-tuned rules, to build systems that adapt to fluctuating environments, and finally to help humans learn (e.g., data mining).
3. A labeled training set is a training set that contains the desired solution (a.k.a. a label) for each instance.
4. The two most common supervised tasks are regression and classification.
5. Common unsupervised tasks include clustering, visualization, dimensionality reduction, and association rule learning.
6. Reinforcement Learning is likely to perform best if we want a robot to learn to walk in various unknown terrains, since this is typically the type of problem that Reinforcement Learning tackles. It might be possible to express the problem as a supervised or semisupervised learning problem, but it would be less natural.
7. If you don't know how to define the groups, then you can use a clustering algorithm (unsupervised learning) to segment your customers into clusters of similar customers. However, if you know what groups you would like to have, then you can feed many examples of each group to a classification algorithm (supervised learning), and it will classify all your customers into these groups.
8. Spam detection is a typical supervised learning problem: the algorithm is fed many emails along with their labels (spam or not spam).
9. An online learning system can learn incrementally, as opposed to a batch learning system. This makes it capable of adapting rapidly to both changing data and autonomous systems, and of training on very large quantities of data.
10. Out-of-core algorithms can handle vast quantities of data that cannot fit in a computer's main memory. An out-of-core learning algorithm chops the data into mini-batches and uses online learning techniques to learn from these mini-batches.
11. An instance-based learning system learns the training data by heart; then, when given a new instance, it uses a similarity measure to find the most similar learned instances and uses them to make predictions.
12. A model has one or more model parameters that determine what it will predict given a new instance (e.g., the slope of a linear model). A learning algorithm tries to find optimal values for these parameters such that the model generalizes well to new instances. A hyperparameter is a parameter of the learning algorithm itself, not of the model (e.g., the amount of regularization to apply).
13. Model-based learning algorithms search for an optimal value for the model parameters such that the model will generalize well to new instances. We usually train such systems by minimizing a cost function that measures how bad the system is at making predictions on the training data, plus a penalty for model complexity if the model is regularized. To make predictions, we feed the new instance's features into the model's prediction function, using the parameter values found by the learning algorithm.
14. Some of the main challenges in Machine Learning are the lack of data, poor data quality, nonrepresentative data, uninformative features, excessively simple models that underfit the training data, and excessively complex models that overfit the data.
15. If a model performs great on the training data but generalizes poorly to new instances, the model is likely overfitting the training data (or we got extremely lucky on the training data). Possible solutions to overfitting are getting more data, simplifying the model (selecting a simpler algorithm, reducing the number of parameters or features used, or regularizing the model), or reducing the noise in the training data.
16. A test set is used to estimate the generalization error that a model will make on new instances, before the model is launched in production.
17. A validation set is used to compare models. It makes it possible to select the best model and tune the hyperparameters.
18. The train-dev set is used when there is a risk of mismatch between the training data and the data used in the validation and test datasets (which should always be as close as possible to the data used once the model is in production). The train-dev set is a part of the training set that's held out (the model is not trained on it). The model is trained on the rest of the training set, and evaluated on both the train-dev set and the validation set. If the model performs well on the training set but not on the train-dev set, then the model is likely overfitting the training set. If it performs well on both the training set and the train-dev set, but not on the validation set, then there is probably a significant data mismatch between the training data and the validation + test data, and you should try to improve the training data to make it look more like the validation + test data.
19. If you tune hyperparameters using the test set, you risk overfitting the test set, and the generalization error you measure will be optimistic (you may launch a model that performs worse than you expect).

# CONNECTING WORLDS

# Kontakt


Michael Aydinbas

Data Scientist

@ michael.aydinbas@EXXETA.com

**m** +49 174 99 50 913