

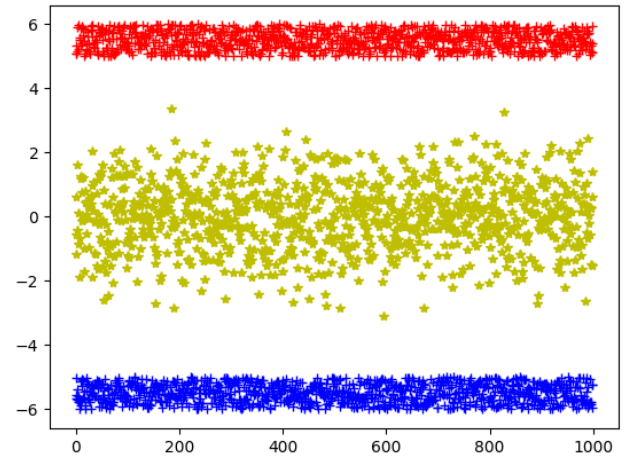
Content and Goals

- Descriptive statistics of data
- Introduction to inferential statistics
- Probability
- Common probability distributions
- Conditional probabilities, Bayes' theorem
- Outliers

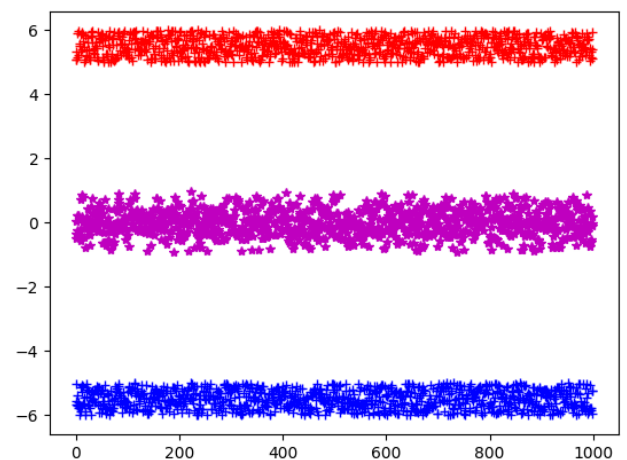
Probability

- Machine Learning is the science of finding relationships in data in the presence of uncertainty.

- Can you derive a relationship between the differently colored data points?



- Now?



Probability

- Machine Learning is the science of finding relationships in data in the presence of uncertainty.
- Uncertainties can have different sources:
 - ▶ Inherent randomness in the system: E.g. random fluctuations due to temperature, quantum systems, random choice of people in a poll, ...
 - ▶ Incomplete observation: A system can behave deterministically but you do not recognize it because you cannot measure all properties of a system. Example: shell game, as the presenter you know under which cup you place the ball, while from the contestant's point of view it is random.
 - ▶ Incomplete model: We neglect certain properties of the real system for the sake of simplicity of our model. E.g. you measure the pressure of a gas in a machine while neglecting temperature effects. However, if your measurement values are collected at different temperatures, the corresponding effect seems random to you.
 - ▶ Measurement uncertainty: Every measurement has an error adding a further source of uncertainty.

Uncertainty

- In summary, various effects add uncertainty to our data.
- In planned experiments, you try to avoid uncertainty as far as possible, e.g. using laboratory systems instead of real systems.
- In Machine Learning, they are usually unavoidable,
- We work with **probabilities**.
- A probability for a specific outcome of an experiment is the fraction of this specific outcome among all outcomes during an infinite number of repetitions of the experiment. → frequentist view
- A probability of an outcome is a measure of how certain we are that we see this outcome when we conduct the experiment. → Bayesian view

- A random variable X is a variable which takes different values or states.
- Two cases: either X is discrete (countable, also countably infinite), or continuous (not countably infinite).
- Example for discrete random variable: Throwing a die yields a number in $\{1, 2, 3, 4, 5, 6\}$.
- Example for an continuous random variable: Throw a dart arrow at a wall, measure the exact location (arbitrarily exact).
- Notice in reality you can map everything to a discrete case, due to measurement limitations. However, for practical reasons we consider everything continuous which is naturally a continuous variable.

Random Variables

Some more details

- Especially discrete variables can be subdivided further:
 - ▶ **categorical variable:** distinguishing different kinds of things, like e.g. species, color, gender, ...
 - ▶ categorical variables cannot be put into an order
 - ▶ **binary variable:** special case of categorical variable with exactly two categories
 - ▶ **ordinal variable:** Discrete variable with an order, e.g. a rank, (discrete) positions, price categories (cheap, normal, expensive), grades, ...

Random Variables

Data Scientist's view

- Data science is all about automatically extract relationships between random variables.
- Frankly speaking, this is statistics.
- Machine Learning just adds a bunch of automation.
- And usually removes (much) rigorosity.
- "Every model is wrong, but some are useful."

Random Variables

Modelling

- In modelling you typically differentiate between **independent** and **dependent** variables.
- Dependent variables are affected by changes in the independenten variables.
- Usually, this is just a question of definition:
 - ▶ The independent variables are those you measure or manipulate (in case you do experiments)
 - ▶ The dependent variables are those you observe.
 - ▶ Attention! Independent variables are not necessarily independent (of each other).
 - ▶ Which variable is the dependent one depends on the causality you assume.
 - ▶ Example: Time spent in front of a computer, and binary category "IT professional".
 - ▶ Are you an IT professional because you spent a lot of time in front of a computer or vice versa?

- A discrete random variable X is a variable which takes different values or states x_i .
- Each value x_i gets a probability $P(x_i)$ assigned for X taking the value x_i .
- $P(x)$ is the **probability mass function** assigning a probability to each value x .
- These probabilities must obey certain rules:
 - ▶ The domain of P is the whole range of possible values of X .
 - ▶ $0 \leq P(x_i) \leq 1$, with a probability of zero indicating that the value is impossible, while a probability of one indicates that the value is absolutely certain.
 - ▶ $\sum_i P(x_i) = 1$: The sum over all probabilities is one, some value will be taken.

Continuous Random Variables

- The case of a continuous random variable X is similar, but not equal.
- We deal with a probability density function $p(x)$.
- It does not represent the probability that X assumes the value x .
- Rather, the probability to find the value of X within an infinitesimal region Δx around x is given by $p(x)\Delta x$.
- Probability densities also obey certain rules:
 - ▶ The domain of p is the whole range of possible values of X .
 - ▶ $0 \leq p(x)$
 - ▶ $\int p(x)dx = 1$: The integral over all probabilities is one, some value will be taken.
- The probability of X assuming a value within the a region R (e.g. in one dimension an interval $[a; b]$) is given by

$$P(X \in R) = \int_R p(x)dx = \int_a^b p(x)dx \quad (1)$$

- with the last equality applying for the one-dimensional example.

- Probability mass functions and densities can be defined for several random variables, too.
- E.g. $P(X_1 = x, X_2 = y)$ is the **joint probability** that X_1 has the value x and at the same time X_2 has the value y .
- Notation: Random variables are separated by commas.
- If the probability (density) depends on a set of parameters those are noted right of the random variables separated by a semicolon, e.g. $P(X; \theta)$ in case the probability depends on parameters θ .

- If a probability (density) depends on several random variables but we are only interested in a subset of them we sum (integrate) over those we are not interested in.
- Example: Given $P(X_1, X_2)$ we are interested in $P(X_1)$ we get

$$P(X_1 = x) = \sum_y P(X_1 = x, X_2 = y) \quad (2)$$

- or in the continuous case

$$p(x) = \int p(x, y) dy \quad (3)$$

Conditional Probability

The probability that a random variable X_1 has a value $X_1 = x$ under the condition that another random variable X_2 has a value $X_2 = y$ is given by

$$P(X_1 = x | X_2 = y) = \frac{P(X_1 = x, X_2 = y)}{P(X_2 = y)} \quad (4)$$

Interpretation:

- Consider a group of 30 people from three different countries, with equal distribution, i.e. 10 people per country. Assume that every person only speaks the mother tongue of his or her country but no other (foreign) language.
- The random variables are the country C a person comes from and the language L the person speaks.
- The joint probability to find a person from the first country ($C = 1$) speaking the language of the first country ($L = 1$) is $P(C = 1, L = 1) = 10/30 \approx 0.33$.

Conditional Probability

$$P(X_1 = x | X_2 = y) = \frac{P(X_1 = x, X_2 = y)}{P(X_2 = y)} \quad (5)$$

Interpretation:

- The joint probability to find a person from the first country ($C = 1$) speaking the language of the first country ($L = 1$) is $P(C = 1, L = 1) = 10/30 \approx 0.33$.
- The conditional probability to find a person from country 1 given that he or she speaks the language of this country however is given by $P(C = 1 | L = 1) = \frac{P(C=1, L=1)}{P(L=1)} = \frac{10/30}{10/30} = 1$.
- That means if we knew the persons speaks language 1 we know for sure he or she is from country 1.
- Notice that you cannot condition on an impossible event.

Conditional Probability

- From the definition of the conditional probability we can derive the product rule of probability theory.
- For two random variables we directly see it from the previous definition:

$$P(X_1 = x, X_2 = y) = P(X_2 = y | X_1 = x) P(X_1 = x) \quad (6)$$

- For an arbitrary number of n random variables $X_i, i = 1, \dots, n$, we can easily extend it.
- $n = 3$:

$$\begin{aligned} P(X_1, X_2, X_3) &= P(X_3 | X_1, X_2) \cdot P(X_1, X_2) = P(X_3 | X_1, X_2) \cdot P(X_2 | X_1) \cdot P(X_1) \\ &= P(X_1) \cdot P(X_2 | X_1) \cdot P(X_3 | X_1, X_2) \end{aligned}$$

- or generally,

$$P(X_1, X_2, \dots, X_n) = P(X_1) \prod_{i=2}^n P(X_i | X_1, \dots, X_{i-1}). \quad (7)$$

Independence of Random Variables

- Two random variables are independent if their joint probability distribution factorizes according to

$$\forall x, y \quad p(X_1 = x, X_2 = y) = p(X_1 = x) p(X_2 = y) \quad (8)$$

- They are conditionally independent if this factorization applies dependent on a third variable:

$$\forall x, y, z \quad p(X_1 = x, X_2 = y | X_3 = z) = p(X_1 = x | X_3 = z) p(X_2 = y | X_3 = z) \quad (9)$$

- When working with data, your whole dataset is a collection of values for random variables.
- Example: Each column of the iris dataset you worked with in the introduction represents a random variable.
- What can you infer about these random variables from the values you got in the dataset?
- Let's talk about the term "distribution" first.

- A probability distribution how likely the possible values of a random variable occur.
- A probability distribution can be described
 - ▶ Graphically (however, this is not an exact way)
 - ▶ By all its moments (expectation (mean) values for the variables X , X^2 , X^3 , X^4 , ...)
 - ▶ Some by a closed formula.

Expectation Value and Variance

- Expectation value and variance are the first two moments of a probability distribution.
- We assume in the following we know the probability mass or density function, respectively, for our random variable.
- We will see afterwards how we compute these quantities from data.
 - ▶ The expected value, or mean, of a random variable X is defined as

$$E_{X \sim P}[X] = \sum_i x_i P(X = x_i) \quad \text{or} \quad E_{X \sim P}[X] = \int_{-\infty}^{\infty} x p(x) dx. \quad (10)$$

- ▶ The variance of a random variable X is defined as

$$\text{Var}(X) = E[(X - E[X])^2]. \quad (11)$$

- ▶ The standard deviation of a random variable X is given by $\text{Std}(X) = \sqrt{\text{Var}(X)}$.

Higher Moments

- The third moment is called the **skewness**.
- It measures the lack of symmetry in the distribution.
- It is the normalized third moment:

$$\text{skew}[X] = E \left[\left(\frac{X - \mu}{\sigma} \right)^3 \right] \quad (12)$$

- Normally distributed variables have a skewness of zero.

- The fourth moment is called the **kurtosis**.
- Kurtosis measures how much scores cluster at the end of the distribution (tails).
- It therefore is a measure for the pointyness
- It is the normalized third moment:

$$\text{kurt}[X] = E \left[\left(\frac{(X - \mu)}{\sigma} \right)^4 \right] \quad (13)$$

- Normally distributed variables have a kurtosis of zero.

Bernoulli Distribution

- For a single, binary random variable

$$P(X = 1) = p$$

$$P(X = 0) = 1 - p$$

$$P(x) = p^x(1 - p)^{1-x} \quad (x \in \{0; 1\})$$

$$E_x[x] = p$$

$$\text{Var}_x[x] = p(1 - p)$$

- Example: Tossing a coin.
- Can you deal with a non-binary random variable using the Bernoulli distribution? For example for throwing a (balanced) die?

Gaussian or Normal Distribution

- The Gaussian distribution or normal distribution is (due to its significance in statistics) probably the most widely used continuous probability distribution.
- It plays a central role in limit theorems in statistics.

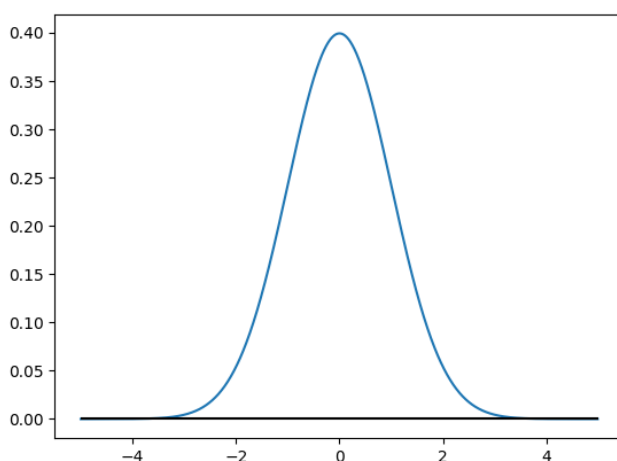
$$N(x; \mu, \sigma) = \sqrt{\frac{1}{2\pi\sigma^2}} \exp \left[-\frac{(x - \mu)^2}{2\sigma^2} \right] \quad (14)$$

- Mean value and variance are given by

$$E_{\mathcal{N}}[X] = \mu \quad \text{Var}_{\mathcal{N}}(X) = \sigma^2 \quad (15)$$

- Verify the mean value using the definition $E_{\mathcal{N}}[X] = \int_{-\infty}^{\infty} x \mathcal{N}(x; \mu; \sigma^2) dx$.

Gaussian or Normal Distribution



- To compute the empirical mean \bar{x} from data $\{x_i\}_{i=1}^N$, you calculate

$$\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i \quad (16)$$

- To compute the empirical variance s^2 from data $\{x_i\}_{i=1}^N$, you calculate

$$s^2 = \frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2 \quad (17)$$

Further Important Empirical Quantities

- The center of a distribution is often well described by the **median**.
- The median value of a set of a data sample $\{x_i\}_{i=1}^N$ is chosen such that 50% of the datapoints are larger and 50% are smaller (or equal).
- To compute it, first sort the values $\{x_i\}_{i=1}^N$ in ascending order.
 - ▶ In case you have an odd number of values there is a clear center value. It is found at position $\lceil \frac{N}{2} \rceil$ (when counting starts at 1)
 - ▶ In case of an even number take the average of the two center values $\frac{x_{N/2} + x_{N/2+1}}{2}$.
- The advantage of the median compared to the mean: It is less susceptible to outliers.
- *Example:* Consider the values 5.1, 5.2, 5.5, 5.3, 5.2, 10.3 with an obvious outlier. Compute mean and median.

Further Important Empirical Quantities

- **Mode** of the data
- Simple: The data point (or interval) that occurs the most in the data.
- Determine the mode by sorting the scores and look for the largest one.
- Example: For the Gaussian distribution the mean value is the mode.
- Attention: It may happen that several values have the same score.
- Multimodal distribution (bimodal in case of two)

Exploring Data Graphically

- Histogram plots
- Discrete random variable: Plot a histogram with a bar per possible value which shows the number of occurrences of that particular value or, better, the rate (number / total number of data points)
- Continuous random variable: Subdivide the range of values into disjunct intervals (bins) item A variant, especially in case of continuous variables, is a density plot.
- In this case the histogram counts are divided by the total counts and the bin width.
- The resulting graph has an integral of 1 when integrating over all values.

Notebook "Datascience"

Create a histogram plot per random variable in the iris dataset. Use matplotlib library.
<https://matplotlib.org/stable/gallery/statistics/hist.html>

Notebook "Datascience"

- Look again at the iris dataset.
- Compute mean, standard deviation, skew, kurtosis for each (numerical) data column.
- What do you observe?

- Until now, we looked almost exclusively on single (random) variables.
- Now we investigate the relation between different random variables.
- Our final goal, we want to determine the (most probable) value of a random variable Y provided we know the values of an arbitrary number of other random variables X_1, X_2, \dots
- Mathematically, we want a **model** F , such that $y = F(x_1, x_2, \dots)$

Statistical Learning Theory

- Consider a set of variables $x \in \mathbb{R}^M$ and a variable $y \in \mathbb{R}$.
- There is a joint probability distribution $P(x, y)$.
Usually a **learning problem** is formulated as finding a function

$$f : \mathbb{R}^M \rightarrow \mathbb{R} \tag{18}$$

$$y = f(X) \tag{19}$$

- For all (x, y) sampled from the distribution P .
- The values x are called the **features**
- y is called the **target** or **dependent variable**.

Kinds of Learning Problems

Classification

- Usually, we try to learn either a discrete target variable or a continuous one.
- We refer to discrete targets as **classes**.
- y can adopt several, distinguishable values, usually enumerated $0, 1, 2, \dots$ or denoted as "class 0" (C_0), "class 1" (C_1) and so on.
- A special case is **binary classification**, i.e. only two classes.
- Usually, classification is discussed for the binary case.
- Multi-class classification can be derived from the binary case noticing that it is comprised of several binary classifications in a row:
- C_0 or $\neg C_0$, if $\neg C_0$, decide C_1 or $\neg C_1$, if $\neg C_1$ decide C_2 or $\neg C_2$,

Kinds of Learning Problems

Regression

- In case of a continuous target variable y we refer to it as **regression**.
- A multidimensional target $y \in \mathbb{R}^n$ can be decomposed into n (independent) one-dimensional regression problems.
- Regression and classification, though different problems, can be mapped onto one another.
- E.g. a regression can be turned into a classification by turning the regression value $y \in \mathbb{R}$ into a probability:
 - ▶ For each class C_i predict a target variable y_i .
 - ▶ Map every y_i into the range $[0; 1]$ and normalize all y_i such that $\sum_i y_i = 1$.
- For that purpose, usually the softmax function is used

$$\tilde{y}_i = \frac{\exp y_i}{\sum_j \exp y_j} \quad (20)$$

- How do we use probabilities?
- The decision problem in terms of probabilities is finding the conditional probability to see a certain class or value for the target y depending on the values for the features x
 - ▶ $p(C_i|x)$, for classification, or
 - ▶ $p(y|x)$, respectively, for regression.
- This *posterior* probability can be derived from the prior probability $p(C_i)$ and the joint probability $p(x, C_i)$ according to Bayes' rule:

$$p(C_i|x) = \frac{p(x|C_i)p(C_i)}{p(x)} \quad (21)$$

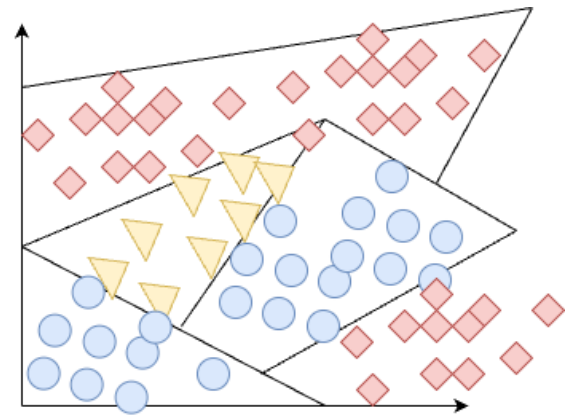
Example Case

Let's consider the following binary classification case:

- Medical image classification of tumor images.
- Given a medical grayscale image, where x represents the array of pixel values.
- Target is the prediction of whether the tumor is benign (class 0) or malign (class 1).
- For every value x we search for the right class, either C_0 or C_1 .
- We assume that the whole possible feature space ($x \in \mathbb{R}^n$) has two regions R_0 and R_1 such that $x \in R_0 \Rightarrow C_0$ and $x \in R_1 \Rightarrow C_1$.
- These regions form the whole feature space, i.e. $R_0 \cup R_1 = \mathbb{R}$.
- R_0 and R_1 are not necessarily connected.

Decision Regions in Feature Space

- For every class there is a subset of the whole feature space.
- All these subsets are disjoint (unique answer)
- The subsets are not necessarily connected.
- Between the decision regions we find the decision boundaries.
- Finding these decision boundaries is the central task of machine learning. As they usually are not trivial.



Decision Boundaries

Practical Part

- Visualize the decision boundaries in the iris dataset.
- To this end, select 3 pairs of features in a scatter plot and visualize where in this two-dimensional plane which class is located.
- Use e.g. the Python matplotlib package. You find some initial link in the notebook.
- What do you observe?
- We will discuss your results afterwards in the group.

Example Case

$$p(\text{error}) = p(x \in R_0, C_1) + p(x \in R_1, C_0) = \int_{R_0} p(x, C_1) + \int_{R_1} p(x, C_0) \quad (22)$$

- In the case of our binary image classification (benign / malignant) task, we try to decide according to x which class we assign our image to.
- We can do this using the probabilities $p(C_0|x)$ and $p(C_1|x)$.
- Our result is that class, for which this conditional probability is the largest.
- Remember, $p(C_i|x) = \frac{p(x, C_i)}{p(x)}$. So both conditional probabilities sum up to 1.

Example Case

In the case of our binary image classification (benign / malignant) task, we try to minimize our error:

$$p(\text{error}) = p(x \in R_0, C_1) + p(x \in R_1, C_0) = \int_{R_0} p(x, C_1) + \int_{R_1} p(x, C_0) \quad (23)$$

- What kind of errors are possible?

Example Case

In the case of our binary image classification (benign / malignant) task, we try to minimize our error:

$$p(\text{error}) = p(x \in R_0, C_1) + p(x \in R_1, C_0) = \int_{R_0} p(x, C_1) + \int_{R_1} p(x, C_0) \tag{24}$$

true vs. predicted	C_0	C_1
C_0	correct	error 1st class
C_1	error 2nd class	correct

Which error is worse (let's say C_0 is benign)?

Example Case

true vs. predicted	C_0	C_1
C_0	correct	error 1st class
C_1	error 2nd class	correct

- The error second class means, you scare a patient and conduct (potentially costly) further medical investigations.
- The error first class means, a patient potentially dies due to a misclassified tumor image.
- You have to consider the consequence of an error in your machine learning objective.
- A pure minimization of the total error probability is not always appropriate.

Consider the different error cases:

- Classify an event as a fraudulent event (e.g. credit card fraud detection).
- Manufacturing quality check (ok part vs. not-ok part)
- Product recommendations
- Image recognition in logistics: different parts shall be classified and assigned to the correct box.

Loss Function

- Introduce a loss function instead of measuring merely the prediction error.
- For our medical image recognition task, we could e.g. weight the first class loss much higher than the second class:

$$E[L] = 1 \cdot \int_{R_0} p(x, C_1) + 1000 \cdot \int_{R_1} p(x, C_0) \quad (25)$$

- In real applications, further options need to be evaluated.
- Using the specific probability values $p(C_i|x)$ for more than a 0/1-decision.
- The probability value itself can be interpreted as a confidence of the prediction.
- E.g. the prediction of benign could be rejected if $p(C_0|x) < \rho$ with some (reasonable) threshold value ρ , e.g. $\rho = 0.95$,
- In that case the model replies a third option, which just means "not sure enough".

Methods of Classification

- Probabilistic or generative approaches: Obtain and normalize the joint probabilities $p(x, C_i)$ according to Bayes' rule:

$$p(C_i|x) = \frac{p(x, C_i)}{p(x)} \quad (26)$$

- Apply a discriminative approach to obtain $p(C_i|x)$ directly.
- Find a discriminative function f , which assigns a class to each input $C_i = f(x)$. This is a completely non-probabilistic approach.

k-Nearest-Neighbor Classification

- Very basic machine learning algorithm.
- Idea: For a given x find a prediction for y .
- Search the k nearest neighbors of x within your training data based on a distance metric (usually Euclidean distance) and average the corresponding y values.
- Main parameter is the number of neighbors k to be considered.
- Usually, the distance metric is not primarily important.
- Algorithmic challenge: Implementing an effective nearest neighbor search on huge datasets. Tree algorithms outperform a simple brute force search.

k-Nearest-Neighbor Classification

Practical Part

- Implement k-NN classification for the Iris dataset.
- Use scikit-learn KNeighborsClassifier
- Familiarize with the scikit-learn interface, find some useful links in the notebook.
- Split the iris dataset: 90% are training data and 10% are only used to make predictions on.
- What do you have to take care for when splitting the dataset?
- Vary k and visualize its influence on your result.
- Work in teams of 2-3 people, get ready to present and discuss your solution in the group.

- k-NN has some serious computational limitations when it comes to huge amounts of data.
- There is another effect, which not only makes k-NN difficult but generally machine learning.
- With every algorithm where you interpolate or extrapolate, you actually require your data to lie closely together.
- How far is data apart in high dimensional spaces?

Curse of Dimensionality

Consider a dataset X consisting of N datapoints, each sampled randomly from $x \in [0; 1]^d$ ($d \in \mathbb{N}$). We look at the following metric:

$$M(x) = \exp(-10 \cdot \|x\|^2) \quad (27)$$

Practical Part

- What value $M(x_0)$ do you expect for the point x_0 closest to the origin?
- How does $M(x)$ behave for points far away from the origin?
- Create samples for every dimension $d \in \{1, 2, \dots, 20\}$ and compute for each the metric $M(x_0)$ of the point closest to the origin in each case.
- Average $M(x_0)$ over 100 datasets for each dimension value to smooth out randomness. Choose a value for N that seems reasonable to you.
- Plot the average distance $M(x_0)$ over the dimension d .
- What do you observe?

- The higher the space dimension the more far apart from the origin is the point with the smallest metric.
- This is quite counterintuitive to what we know from the three-dimensional space.
- We see that we cannot transfer our 3D experience to higher dimensions.
- Most (interesting) ML problems are based on a high-dimensional feature space.

Regression Loss Function

We treat regression problems (continuous target y) in a similar way.

- The quality of the function f is again determined by a **loss function** $L(x, y)$.
- In many cases we choose the **squared error loss**

$$L(x, y) = (y - f(x))^2. \quad (28)$$

- This loss is defined for a single sample.
- For a whole dataset sampled from the distribution P , our expected loss is given by

$$E_{(x,y)}[(y - f(x))^2] = \iint (y - f(x))^2 p(x, t) dx dt \quad (29)$$

$$E_{(x,y)}[L] = \iint (y - f(x))^2 p(x, y) dx dy \quad (30)$$

Using calculus of variations, we differentiate w.r.t the functional argument $f(x)$.

$$\frac{\delta E_{(x,y)}[L]}{\delta f(x)} = 2 \int (y - f(x)) p(x, y) dy \quad (31)$$

Solving for $f(x)$

$$f(x) = \frac{\int y p(x, y) dy}{p(x)} = \int y p(y|x) dy = E_y[y|x] \quad (32)$$

Which is the conditional average of y conditioned on x .

Loss Function - Decomposition

The loss can be decomposed into two parts. First, rewrite

$$L = (f(x) - y)^2 = (f(x) - E_y(y|x) + E_y(y|x) - y)^2 \quad (33)$$

$$= (f(x) - E_y(y|x))^2 + (E_y(y|x) - y)^2 + 2(f(x) - E_y(y|x))(E_y(y|x) - y) \quad (34)$$

The mean of the third (the mixed) term yields zero.

$$\iint 2(f(x) - E_y(y|x))(E_y(y|x) - y) p(x, y) dy \quad (35)$$

$$= \iint [2f(x)E_y(y|x) - 2f(x)y - 2E_y(y|x)^2 + 2yE_y(y|x)] p(x, y) dy \quad (36)$$

$$= 0 \quad (37)$$

- The remainder expression for the loss function thus consists of a sum of two parts

$$E[L] = E \left[(f(x) - E_y(y|x))^2 \right] + E \left[(E_y(y|x) - y)^2 \right] \quad (38)$$

- The first term is called the **bias**. It describes the deviation of our function to the expected true value.
- The second term, the **variance** is the intrinsic noise of the data. Note its independency of the model $f(x)$.
- The second term also reveals the minimum of the loss, which cannot be reduced further.

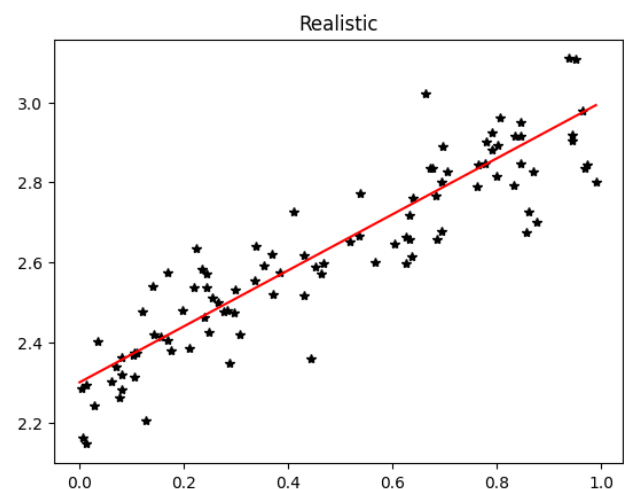
Linear Regression

Problem statement:

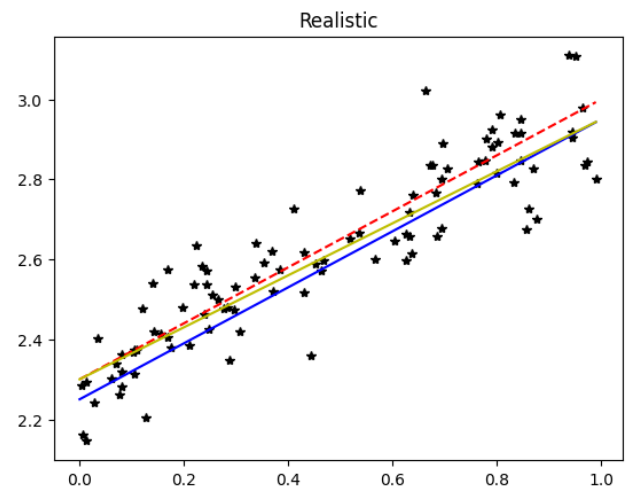
- A sample is characterized by a set of real features $x = (x_1, x_2, \dots, x_p)$
- We want to predict a real output (target) value y .
- We assume a linear relationship (or at least one that is well approximated by it) of the form

$$f(X) = w_0 + \sum_{i=1}^p x_i w_i \quad (39)$$

- with $p + 1$ parameters or coefficients $w = (w_0, w_1, w_2, \dots, w_n)$.



Which one is the right line?



Best Regression via Mean Squared Error Loss

Assume we have a train dataset of size N which we denote by

$$\mathcal{T} = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \dots, (\vec{x}_N, y_N)\}. \quad (40)$$

Here, the indices enumerate the samples.

We obtain the parameters \vec{w} by minimizing the residual sum of squares:

$$\text{RSS}(\vec{w}) = \sum_{i=1}^N (y_i - f(\vec{x}_i))^2 = \sum_{i=1}^N (y_i - w_0 - \sum_{j=1}^p x_{ij} w_j)^2 \quad (41)$$

where x_{ij} is the j -th component of the i th sample, i.e. of the vector \vec{x}_i (remember $\dim(\vec{x}_i) = p$).

- First, we want to get rid of the extra w_0 term: Therefore, we extend the feature vector $x = (x_1, x_2, \dots, x_p)$ by a leading 1 to become $x = (1, x_1, x_2, \dots, x_p)$. In other words, we introduce an $x_0 = 1$ component.
- Thus, $f(x) = x^T \cdot w$ with $w = (w_0, w_1, w_2, \dots, w_p)$.
- To simplify notation, I omit the vector arrow when it is clear, that x is a vector.
- X shall be the $N \times (p + 1)$ matrix containing all training examples.
- Each row j of this matrix corresponds to the features of the j -th training example with the leading 1 appended.
- Similarly, Y is the vector (or $N \times 1$ matrix) containing the corresponding targets.

Solution for Linear Regression

Hence we can write the residual sum of squares (RSS) as matrix-vector product,

$$\text{RSS}(w) = (Y - Xw)^T(Y - Xw). \quad (42)$$

Since we are interested in a minimum, we differentiate the RSS

$$\begin{aligned} \frac{\partial \text{RSS}}{\partial w} &= -2X^T(Y - Xw) \\ \frac{\partial^2 \text{RSS}}{\partial w \partial w^T} &= 2X^T X \end{aligned}$$

Looking for the first derivative being equal to zero yields

$$X^T(Y - Xw) = 0. \quad (43)$$

Solution for Linear Regression

$$X^T(Y - Xw) = 0. \quad (44)$$

Assuming that $X^T X$ is regular (i.e. has an inverse) we obtain a unique solution

$$\hat{w} = (X^T X)^{-1} X^T Y. \quad (45)$$

Consequently, we obtain our prediction \hat{Y} from Linear Regression via

$$\hat{Y} = X\hat{w} = X(X^T X)^{-1} X^T Y \quad (46)$$

In a perfect world (perfect linear relationship) we find $\hat{Y} = Y$.

- Note, we assume $X^T X$ is invertible.
- Though not strictly valid in general, non-invertibility can be circumvented by removing redundant samples and redundant features so that the assumption is still valid.

Multidimensional Calculus

How did we differentiate the matrix-vector expression?

Generally consider a function $f : \mathbb{R}^{N \times d} \mapsto \mathbb{R}$, which maps an $N \times d$ matrix A to a real number. The derivative of f with respect to A is defined as follows

$$\nabla_A f(A) = \begin{pmatrix} \frac{\partial f}{\partial A_{11}} & \cdots & \frac{\partial f}{\partial A_{1d}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial A_{N1}} & \cdots & \frac{\partial f}{\partial A_{Nd}} \end{pmatrix} \quad (47)$$

∇ is the differentiation operator. The above matrix is called **Jacobi matrix**, or short Jacobian.

Consider our $\text{RSS}(w)$, which we derive by the vector w :

$$\begin{aligned} & \nabla_w (Y - Xw)^T (Y - Xw) \\ &= \nabla_w ((Xw)^T Xw - (Xw)^T Y - Y^T (Xw) + Y^T Y) \\ &= \nabla_w (w^T (X^T X) w - Y^T (Xw) - Y^T (Xw)) \\ &= \nabla_w (w^T (X^T X) w - 2Y^T (Xw)) \\ &= 2X^T Xw - 2X^T Y \end{aligned}$$

where we used $(AB)^T = B^T A^T$, from which $A^T B = B^T A$ follows. This term appears in the scalar product in the third step (note that $(AB)^T = AB$ for a scalar product).

Probabilistic Interpretation

- Our data points (samples) are taken from a common distribution.
- Consider data with a linear relation between features x and target y
- plus a further, unsystematic term representing random noise.
- For every sample i we thus have the relation

$$y^{(i)} = w^T x^{(i)} + \epsilon^{(i)} \quad (48)$$

- The $\epsilon^{(i)}$ are independently and identically distributed (IID) noise terms which stem from a Gaussian with mean zero and variance σ^2 .

$$p(\epsilon^{(i)}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\epsilon^{(i)2}}{2\sigma^2}\right) \quad (49)$$

- Accordingly, the conditional probability of $y^{(i)}$ on the condition that x takes the values $x^{(i)}$ is given by

$$p(y^{(i)}|x^{(i)}; w) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y^{(i)} - w^T x^{(i)})^2}{2\sigma^2}\right) \quad (50)$$

- Note, the parameter vector w is not a conditional, thus it is separated by a semicolon in the arguments.
- The empirical quantity $p(y^{(i)}|x^{(i)}; w)$ is usually called **likelihood**.

- The conditional probability each sample is a Gaussian itself.
- For the full dataset we use that all samples are independent.
- Thus the likelihood for the total dataset is a product of the individual likelihoods:

$$\begin{aligned} p(Y|X; w) &=: L(w) = \prod_{i=1}^N p(y^{(i)}|x^{(i)}; w) \\ &= \prod_{i=1}^N \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y^{(i)} - w^T x^{(i)})^2}{2\sigma^2}\right) \end{aligned}$$

- What's a good choice for our parameters (i.e. parameter vector) w ?*

Probabilistic Interpretation

- For a good regression model, we need to *maximize* the likelihood.
- Practically, we switch to the logarithm of the likelihood (this is called **log likelihood**).
- The log turns the product into a sum, which we can much more easily maximize than a product.
- Due to monotonicity of log, the position of the maximum remains the same.

$$\begin{aligned}l(w) &= \log L(w) \\&= \log \prod_{i=1}^N \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y^{(i)} - w^T x^{(i)})^2}{2\sigma^2}\right) \\&= \sum_{i=1}^N \log \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y^{(i)} - w^T x^{(i)})^2}{2\sigma^2}\right) \\&= N \log \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2} \sum_{i=1}^N (y^{(i)} - w^T x^{(i)})^2\end{aligned}$$

Probabilistic Interpretation

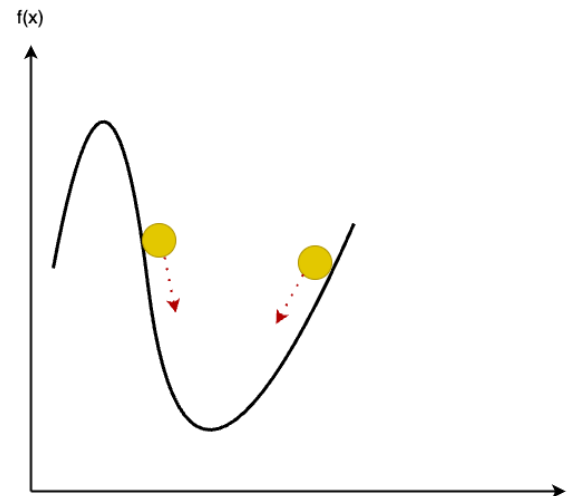
$$l(w) = N \log \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2} \sum_{i=1}^N (y^{(i)} - w^T x^{(i)})^2 \quad (51)$$

- Maximizing $l(w)$ is equivalent to minimizing

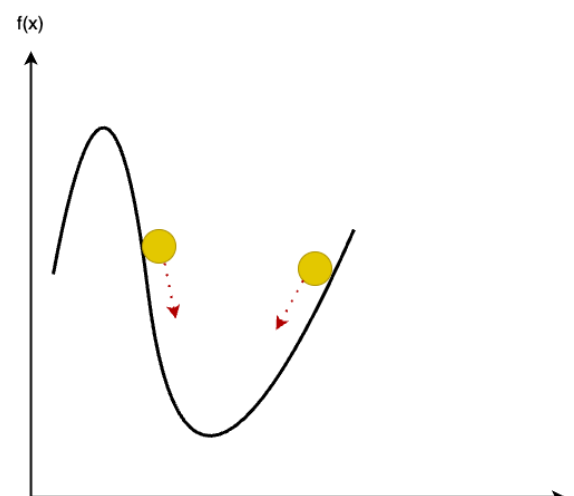
$$\sum_{i=1}^N (y^{(i)} - w^T x^{(i)})^2 \quad (52)$$

- What do we see from this result?
- Under the probabilistic assumptions we made (additive normal distributed noise) minimizing the residual sum of squares is equivalent to maximum likelihood.
- The result does not depend on the variance σ^2 of the noise.

- Though, linear regression has an analytical solution, in real problems it is usually solved numerically.
- A common algorithm is **gradient descent** (Deep Learning!).
- The idea is fairly simple: Follow the steepest gradient of the function to minimize, to find the minimum.



- At the position of the left yellow point the gradient is negative (increasing x leads to decreasing $f(x)$). Increasing the x -value by $-\alpha \cdot \text{gradient}$, which is positive, directly leads to the minimum x -coordinate.
- At the position of the right yellow point the gradient is positive you need to decrease the x -coordinate to arrive at the minimum, again add $-\alpha \cdot \text{gradient}$.
- The factor α is a weight for the gradient influencing the step size applied to the variable x . It will be called the **learning rate** in the following. We will explore it a little more later.



The algorithm works quite simple:

- For all w_j , $w_j \leftarrow w_j - \alpha$ loss gradient for current values of w :

$$w_j \leftarrow w_j - \alpha \sum_{i=1}^N (-2)(y^{(i)} - x^{(i)}w)x_j \quad (53)$$

- This parameter update is repeated until w converges.
- Since we use the full training dataset for every update, this variant of the algorithm is often called *batch gradient descent*.

- Especially for large training dataset this computation can become very time consuming.
- Therefore, often a variant called *stochastic gradient descent* is used by practitioners:

$$w_j \leftarrow w_j - \alpha \sum_{i \in M} (-2)(y^{(i)} - x^{(i)}w)x_j \quad (54)$$

- M is a random subset (therefore stochastic GD) of the training dataset of fixed size.
- Thus, computation time remains constant for big training sets.
- In the extrem case, $|M| = 1$, a parameter update is done by a randomly chosen single sample.

Practical Part

- Implement gradient descent for some simple functions.
- See the functions in the notebook.
- Vary the learning rate.
- Consider the simultaneous optimization problem. What is the implication of your findings?

Measure Regression Quality

- A good quantity to look at globally is the root mean absolute error:

$$\frac{1}{N} \sqrt{\sum_{i=1}^N (y^{(i)} - \hat{y}^{(i)})^2} \quad (55)$$

- The root ensures that we measure in the units of our target variable.
- On per sample level, this boils down to the absolute error $|y^{(i)} - \hat{y}^{(i)}|$ or, with direction, the absolute error $y^{(i)} - \hat{y}^{(i)}$.
- However, there the relative error is preferable to see the deviation from the true value:

$$\frac{y^{(i)} - \hat{y}^{(i)}}{y^{(i)}} \quad (56)$$

- Although for the fit of the parameters a global loss is used, during the evaluation you need to consider individual samples and look for error regions.

In statistics, often the R^2 score is used to evaluate regression models.

- To define and interpret R^2 we need two quantities.
- What is an easy best guess for a target variable, if you do not have any information (feature)?

In statistics, often the R^2 score is used to evaluate regression models.

- The scikit-learn score function for regression models typically uses R^2 .
- To define and interpret R^2 we need two quantities.
- What is an easy best guess for a target variable, if you do not have any information (feature)?
- You simply compute the mean value \bar{y} of all target values you know.
- The corresponding mean squared error is called the total sum of squares (TSS):

$$\text{TSS} = \sum_{i=1}^N (y^{(i)} - \bar{y})^2 \quad (57)$$

Measure Regression Quality

- The error of our model is expressed as the residual sum of squares RSS:

$$\text{RSS} = \sum_{i=1}^N (y^{(i)} - \hat{y}^{(i)})^2 \quad (58)$$

- The R^2 score now measures how much more of the variation in the data we can explain by our model compared to the most simple mean value model, i.e.

$$R^2 = \frac{\text{TSS} - \text{RSS}}{\text{TSS}}. \quad (59)$$

- $R^2 \leq 1$: If our model is perfect, $\text{RSS} = 0$, then $R^2 = 1$.
- If our model is really bad, RSS is high and might even exceed TSS. A negative R^2 is the consequence.
- The closer R^2 to 1 the more of the fluctuation in the data is explained by our model relative to the best guess mean model.

Data Normalization

For many ML algorithms it is beneficial to scale the data. Several methods exist, the two most popular are

- scaling to mean zero and variance 1: $x \rightarrow \frac{x - \text{mean}}{\text{std. dev.}}$
- scaling between 0 and 1: $x \rightarrow \frac{x - \min(x)}{\max(x) - \min(x)}$

Linear Regression Example

- We will build a simple linear regression for the diabetes dataset.
- You predict a continuous response variable, describing diabetes disease progression for patients after one year.
- Features are personal parameters like age, sex, BMI, as well as blood serum measurements.
- You find a dataset loader in the notebook.
- References
 - ▶ <https://www4.stat.ncsu.edu/~boos/var.select/diabetes.html>, 05.09.24
 - ▶ B. Efron, T. Hastie, I. Johnstone and R. Tibshirani Least Angle Regression, *Annals of Statistics* (with discussion), 407-499 (2004)

Linear Regression Example

Practical Part

- Split the data randomly into train and test sets (75%/25% is fine).
- Predict the target variable with linear regression.
- Use the scikit-learn implementation.
- Measure the errors of your prediction and try to figure out where the errors occur.
- Ablation study: Study the effect of data normalization on the quality of your model.

What is your opinion?

- Advantages
 - ▶ Simplicity
 - ▶ Understandability: Weights represent the importance of a feature (provided that data is properly normalized!)
- Disadvantages
 - ▶ Limited applicability, most of the world is non-linear.
 - ▶ Takes every feature into account (many features, many parameters)

Let's do an experiment.

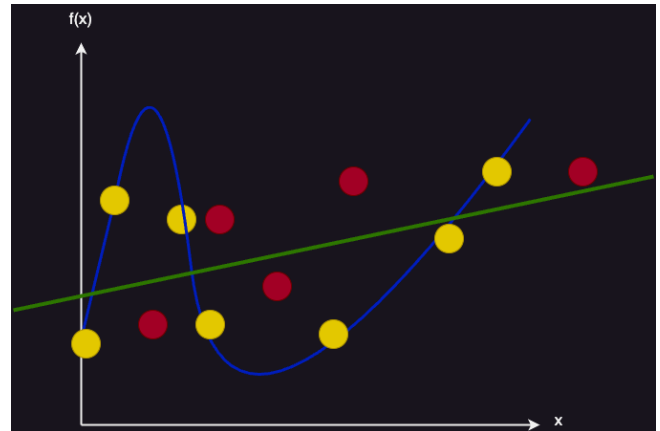
Practical Part

- Until now, only the features themselves have been considered.
- Add non-linear terms representing the interactions between the features.
- For second order, these terms are $x_i x_j$ for features x_i and x_j , third order $x_i x_j x_k$, and so on.
- Limit yourself to third order and interaction terms only (i, j, k mutually different)
- Re-do linear regression with these additional features.
- What do you observe?

- How many features did you add to the original 10 ones?
- What was the training and test error of your linear regression?
- Too many features can be a serious problem for any ML algorithm.
- A big part of the work is removing features that are either useless, redundant or in any other way potentially misleading.
- Note, a lot of this evaluation is based on domain knowledge on the problem at hand.
- However, often we do not know enough about the domain.
- E.g. the blood serum features, even if we consult biomedical literature, we do not know what parameters are behind these features (eventually, we could guess with sufficient knowledge).

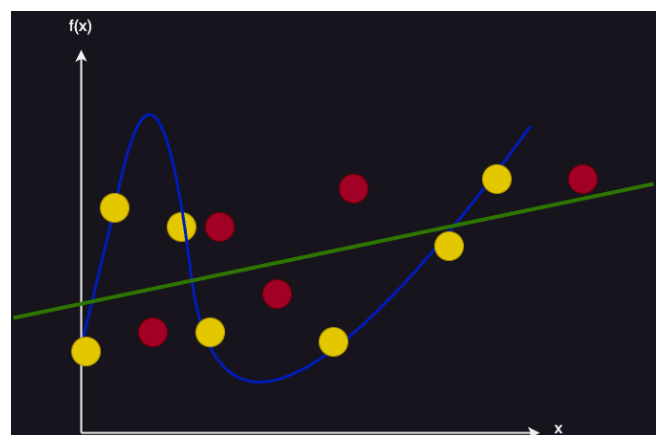
Regularization

- What happens with too many features or too many samples?
- ML models, which are *automatically* adapted, tend to **overfit** to data.
- Overfitting is characterized by a good performance on the training data, but a bad performance on test data.



Regularization

- ML models, which are *automatically* adapted, tend to **overfit** to data.
- Overfitting is characterized by a good performance on the training data, but a bad performance on test data.
- Consequently, the model does not perform well on unseen data, it does not **generalize** well.
- The model adapts too closely to the training data. In the extreme case it learns a prediction specifically for each training data point.



- The error of a machine learning model therefore cannot be measured on the training data.
- A specific test dataset is required to measure the generalization error of an ML model, i.e. the error it will make in operation.
- What we see in the training error is rather **underfitting**.
- During underfitting, the model does not perform well on the training data, specific aspects in the data are not learned.
- Contrary to overfitting, both, training and test error are high.

How to Deal with Over- and Underfitting

- Underfitting
 - ▶ More data
 - ▶ Data quality?
 - ▶ Data preprocessing, do we see relationships in the data which the model does not capture.
 - ▶ Add domain knowledge (if you know something about relationships between features and target)
 - ▶ Use another model (e.g. when using linear regression for a highly non-linear problem)
 - ▶ Analyse erroneous samples to dig into the reason for underfitting.
- Overfitting
 - ▶ Same as underfitting
 - ▶ Plus, regularization

- Overadaptation is caused by the models having many parameters that can be adapted.
- The automatic optimization of an ML model will lead to these parameters actually being adapted to the training data.
- Regularization limits the possibilities of adapting the model parameters.
- For regression, we use L2 regularization (Ridge) or L1 regularization (Lasso).
- Both approaches can also be found in deep learning, where regularization plays a crucial role.

Ridge Regression

- The loss function gets a further term which penalizes the sum of the adjustable parameters:

$$\text{Loss}(w) = \text{RSS}(w) + \lambda \|w\|^2 = \text{RSS}(w) + \lambda \sum_{i=1}^p |w_i|^2 \quad (60)$$

- We see the more every linear weight deviates from zero the higher this additional loss gets.
- The parameter λ controls how much this additional term influences the total loss.
- In the context of regression, this method is sometimes referred to as *shrinkage* because it shrinks the parameters towards zero.
- The same method is applied in neural networks, where it is referred to as *weight decay*.

- Lasso is another shrinkage method.
- The difference to Ridge is subtle but leads to other coefficient behavior.
- Lasso uses the L1 norm instead of the L2 norm:

$$\text{Loss}(w) = \text{RSS}(w) + \lambda \sum_{i=1}^p |w_i| \quad (61)$$

- Therefore, more parameters are forced towards zero.
- Hence, Lasso reveals the most important features.

Ridge Regression and Lasso

Practical Part

- Use Ridge and Lasso for the extended diabetes dataset (including the interaction terms)
- Look into the scikit-learn documentation to learn about the parameter "alpha".
- For which values of alpha do you get the best model?
- When applying Lasso, figure out which features have a non-zero coefficient.

See Notebook MAL03 for some error analysis techniques for diabetes dataset linear regression.

Generalization Error

- A machine learning model is trained on a training dataset \mathcal{T}
- The samples in \mathcal{T} are identically and independently distributed (iid) according to a distribution P .
- To test our model, we need a further test dataset with samples also iid according to P .
- We are interested in the error of our ML models on the test set (**generalization error**).
- Generalization error can only be estimated on an independent dataset, never used for ML training.

Generalization Error

- For a linear regression model, fit by least squares to a training dataset $\{(x^{(i)}, y^{(i)})\}_{i=1}^N$ drawn iid from distribution P .
- The optimal parameters found are w .
- The test dataset $\{(\tilde{x}^{(i)}, \tilde{y}^{(i)})\}_{i=1}^M$ is drawn iid from the same distribution.

With

$$\text{RSS}_{\text{train}} = \frac{1}{N} \sum_{i=1}^N (y^{(i)} - w^T x^{(i)})^2 \quad \text{and} \quad \text{RSS}_{\text{test}} = \frac{1}{M} \sum_{i=1}^M (\tilde{y}^{(i)} - w^T \tilde{x}^{(i)})^2 \quad (62)$$

For linear regression you can generally prove that

$$E[\text{RSS}_{\text{train}}] \leq E[\text{RSS}_{\text{test}}] \quad (63)$$

where the expectation is taken over everything that is random (e.g. the training dataset).

Dataset Split

- In ML we use the concept of **empirical risk minimization**.
- We develop a model and want to reduce the risk of wrong predictions.
- We prove the correctness of our model by empirical evaluation.
- The final test of every model is therefore done on an independent dataset (test dataset held-out dataset).
- During development, we fit our model (e.g. our parameters w of a linear regression) to a training dataset.
- During development of our ML solution, we usually need a further dataset to measure the generalization error of *model variants*.

Dataset Split

- For example, we want to compare Ridge regression to Lasso and check the error on an independent dataset.
- Or, we want to check how different model parameters (like α) influence our model.
- We cannot use the held-out (test) dataset because we cannot use it for any development activity without compromising its independence.
- Therefore, we need another dataset, on which we check our generalization error *during development*, called **validation dataset**.
- Typical splits are 70-15-15 or 60-20-20 of your whole data for training-validation-test.



Information Leakage

- Not only the ML training must exclusively use the training dataset.
- Every data operation that needs adaptation of parameters must be fitted on the training data exclusively.
- Validation and test datasets may not be used and just be transformed with parameters fitted on the training set.
- Example: Normalization
 - ▶ Finding minimum and maximum, or
 - ▶ Finding mean and standard deviation
- Example: Filling missing values, if the fill value is data dependent (we come back to this soon).
- Otherwise, your estimate of generalization error is wrong, usually too low.
- This kind of mistake is called **data or information leakage**.

- Furthermore, you need to ensure that all three datasets contain iid samples according to the same distribution.
- Usually, you can ensure this by sampling randomly from the full dataset.
- A common error is taking subsequent rows of the data and splitting by row numbers.
- Take care when dealing with time series!
- Note, during operation of your ML model, this common distribution from the initial dataset can change. This effect is called **data drift**, we talk about it later.



A single validation dataset is a good start, but not optimal for two reasons:

- Maybe you cannot afford splitting off 30-40% of your data for validation and test sets.
- The error measured on the validation dataset is a random variable. You might randomly select a particularly good (or bad) subset for validation.
- The solution for both problems is a popular approach called **cross validation**.
- It solves both problems (at the cost of higher resource consumption for calculations).

k-Fold Cross Validation

- The common cross validation (CV) algorithm is k-fold CV.
- Split the training dataset T of size $|T| = N$ into k parts:
 $T = T_1 \cup T_2 \cup \dots \cup T_k$ of equal size
 $|T_i| = |T|/k$.
- For each T_k train the model on $T \setminus T_k$ and determine the error on T_k
- Average all validation errors for each T_k to obtain an estimate for the generalization error.



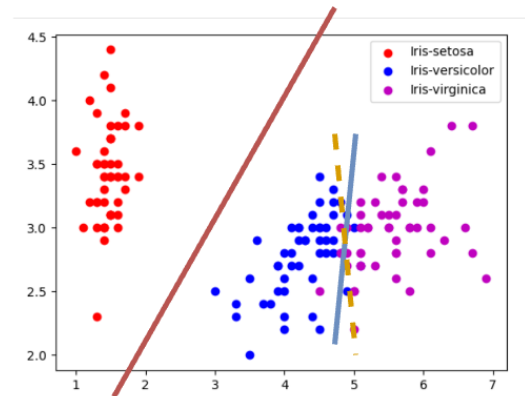
- Repeated k-fold CV: Repeat k-fold CV to add further runs by taking different random splits.
- "Leave-one-out CV": $k = |T|$ (i.e. $|T_k| = 1$)
- Stratified k-fold: In case labels are unbalanced, stratified k-fold keeps this imbalance, at least approx. (Note, balancing the dataset is often the better option, we will come back to that soon.)

Practical Part

- Use k-fold CV to assess whether Ridge or Lasso is better suited for the extended diabetes dataset.
- Vary the parameter α for both models.
- Evaluate your training and validation losses.
- Compare model training with and without normalization. What do you observe?
- Be careful to avoid data leakage during CV.

Linear Classification Models

- As for regression, linear feature-target relationships can occur in classification problems, too.
- Remember, the iris dataset plots.
- We can visibly draw separation lines into the spaces spanned by two features.
- However, we do not know (yet) the "best" separating line.



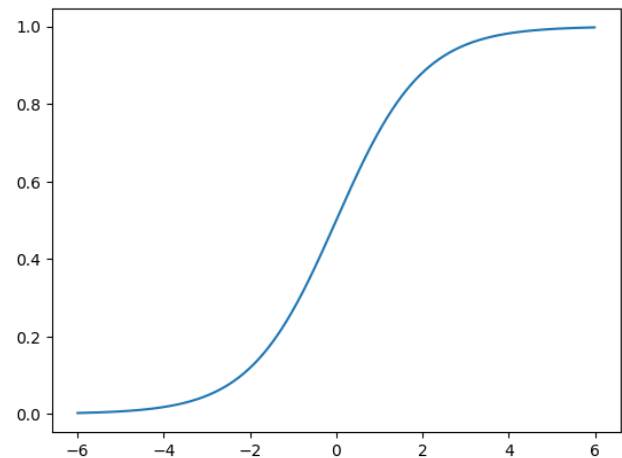
Logistic Regression

- A straightforward way of dealing with classification problems is to re-use what we know about regression.
- How do we map from the continuous valued target to a discrete, binary target?

- A straightforward way of dealing with classification problems is to re-use what we know about regression.
- How do we map from the continuous valued target to a discrete, binary target?
- We use the **logistic or sigmoid function**:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (64)$$

- Can you imagine how it works to transform regression to classification using that function?



Mathematics of Linear Classification

- The logistic function maps a continuous real variable x into the range $[-1; 1] \subset \mathbb{R}$.
- For convenience, we describe the theory for binary classification. We talked about the straightforward extension to arbitrary numbers of classes.
- Introduce a target variable $y = \sigma(wx)$.
- Using y we decide on the class by introducing a class probability

$$P(y = 1|x; w) = \sigma(wx)$$

$$P(y = 0|x; w) = 1 - \sigma(wx)$$

- compactly written:

$$P(y|x; w) = \sigma(wx)^y (1 - \sigma(wx))^{1-y} \quad (65)$$

$$P(y|x; w) = \sigma(wx)^y (1 - \sigma(wx))^{1-y} \quad (66)$$

For a set of iid training samples the likelihood of the parameters w are thus given by

$$\begin{aligned} L(w) &= P(y = 1|X; w) \\ &= \prod_i \sigma(wx^{(i)})^{y^{(i)}} (1 - \sigma(wx^{(i)}))^{1-y^{(i)}} \end{aligned}$$

$$L(w) = \prod_i \sigma(wx^{(i)})^{y^{(i)}} (1 - \sigma(wx^{(i)}))^{1-y^{(i)}}$$

For the log-likelihood we thus obtain

$$l(w) = \log L(w) = \sum_i [y^{(i)} \log(\sigma(wx^{(i)})) + (1 - y^{(i)}) \log(1 - \sigma(wx^{(i)}))]$$

To find the maximum of the log-likelihood we need its derivative.

$$l(w) = \log L(w) = \sum_i [y^{(i)} \log(\sigma(wx^{(i)})) + (1 - y^{(i)}) \log(1 - \sigma(wx^{(i)}))]$$

The derivative for a single training sample (x, y) is computed as

$$\begin{aligned} \frac{\partial}{\partial w_i} l(w) &= \left(y \sigma(wx)^{-1} - (1 - y)(1 - \sigma(wx))^{-1} \right) \frac{\partial \sigma(wx)}{\partial w_i} \\ &= \left(y \sigma(wx)^{-1} - (1 - y)(1 - \sigma(wx))^{-1} \right) \sigma(wx)(1 - \sigma(wx)) \frac{\partial (wx)}{\partial w_i} \\ &= (y(1 - \sigma(wx)) - (1 - y)\sigma(wx)) x_i \\ &= y(1 - \sigma(wx)) x_i \end{aligned}$$

using

$$\frac{d}{dx} \sigma(x) = \frac{1}{(1 + e^{-x})^2} (e^{-x}) = \frac{1}{(1 + e^{-x})} + \left(1 - \frac{1}{(1 + e^{-x})} \right) = \sigma(x)(1 - \sigma(x))$$

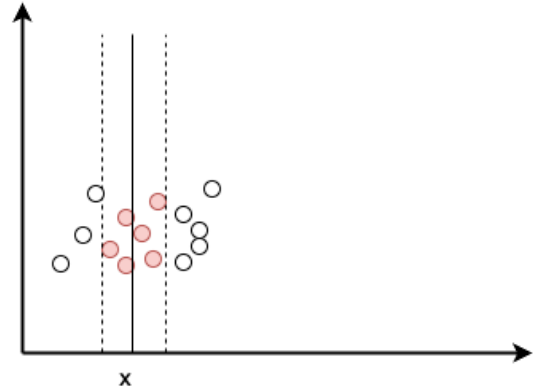
$$\frac{\partial}{\partial w_i} l(w) = y(1 - \sigma(wx)) x_i$$

- We can use this expression now to perform gradient descent updates.
- Even an analytical solution is possible.
- The function plays a huge role in deep learning, too.

Reminder: k-Nearest-Neighbors

- k-NN bases its prediction on the k points in the training data \mathcal{T} which are closest to target point x .
- This neighborhood of size k is denoted by $N_k(x)$.
- The target variables for these training points are averaged to obtain the prediction \hat{y} :

$$\hat{y}(x) = \frac{1}{k} \sum_{i|x_i \in N_k(x)} y_i \quad (67)$$



Reminder: k-Nearest-Neighbors

- k-NN bases its prediction on the k points in the training data \mathcal{T} which are closest to target point x .
- This neighborhood of size k is denoted by $N_k(x)$.
- The target variables for these training points are averaged to obtain the prediction \hat{y} :

$$\hat{y}(x) = \frac{1}{k} \sum_{i|x_i \in N_k(x)} y_i \quad (68)$$

- This classifier is *memory based*, it does not require a model.
- The data should be normalized in interval $[0; 1]$. (Why?)
- k-NN is particularly suitable for irregular decision boundaries.

Classification Metrics

Default metric for classification is the so-called **accuracy**:

$$\text{acc}(T) = \frac{1}{|T|} \sum_{(x,y) \in T} \text{id}[y = \hat{y}(x)] \quad (69)$$

(T denotes the dataset on which we measure the accuracy.)

- Remember, there are several cases of "incorrect" in (binary) classification:
- true positives (TP): correctly classified samples of class 1.
- true negatives (TN): correctly classified samples of class 0.
- false positives (FP): samples of class 0 erroneously classified as class 1.
- false negatives (FN): samples of class 1 erroneously classified as class 0.
- Accuracy treats FP and FN as wrong classifications without further differentiation.
- Remember the different consequences of each kind of error!

Classification Metrics

For a classification problem, the different errors are easily visualized with a confusion matrix.

	Predict Class 0	Predict Class 1
True Class 0	tn	fp
True Class 1	fn	tp

- With the small letter abbreviations, we refer to the number of TN, FN, ... i.e. $tp = |TP|$, $fp = |FP|$, and so on.
- Visual confusion matrixes, see practical part.
- For a non-binary classification problem, these concepts are applied per class.

	Predict Class 0	Predict Class 1
True Class 0	tn	fp
True Class 1	fn	tp

Based on these numbers, **precision** and **recall** are defined:

$$\text{precision} = \frac{tp}{tp + fp} \quad (70)$$

The precision is the fraction of correct classifications among all classifications for class 1.

$$\text{recall} = \frac{tp}{tp + fn} \quad (71)$$

The recall is the fraction of correct classifications among all truly positive samples.

The **F1 score** combines precision and recall.

$$F1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \quad (72)$$

Practical Part 1 - Logistic Regression vs. k-NN

- Find a toy problem in the notebook.
- Visualize the datapoints in the two-dimensional space.
- Guess how will k-NN and logistic regression perform on both datasets.
- Try it out. Use k-fold CV to find a good neighborhood size for k-NN.
- Present and discuss a solution in the group.


Practical Part 2 - Logistic Regression on the iris dataset

- Apply logistic regression to the iris dataset.
- Use, among other things, a confusion matrix to visualize your result.
- Compute all the above-mentioned metrics.
- How would you judge the quality of your model?




Practical Part

- Explore the dataset [1].
- <https://opendata.cern.ch/record/328>
- What is remarkable?

 Adam-Bourdarios, Claire; Cowan, Glen; Germain, Cecile; Guyon, Isabelle; Kégl, Balázs; Rousseau, David; (2014). Learning to discover: the Higgs boson machine learning challenge - Documentation. CERN Open Data Portal.
DOI:10.7483/OPENDATA.ATLAS.MQ5J.GHXA.

Practical Part

- Explore the dataset [1].
- <https://opendata.cern.ch/record/328>
- Replace -999 by missing value np.nan.
- Compute the conditional probability of each class under the condition a feature has a missing value.
- Interpret your results.

 Adam-Bourdarios, Claire; Cowan, Glen; Germain, Cecile; Guyon, Isabelle; Kégl, Balázs; Rousseau, David; (2014). Learning to discover: the Higgs boson machine learning challenge - Documentation. CERN Open Data Portal.
DOI:10.7483/OPENDATA.ATLAS.MQ5J.GHXA.

Missing Values

- Missing values are quite usual in datasets.
- Depending on the origin of the dataset various reasons are plausible for missing values:
 - ▶ Sensor error, measurement range exceeded
 - ▶ Missing reply in a survey
 - ▶ Data could not be recorded
 - ▶ Data definition was changed (columns added later)
 - ▶ Merge of different datasets
 - ▶ ...
- How would you judge different reasons for missing data?

How would you judge different reasons for missing data?

Scenario 1: In a questionnaire people are asked a couple of questions. On the final page, they can provide some personal information like their age. Answering these person questions is optional.

Scenario 2: In a manufacturing setup the quality of a produced part shall be judged using measure parameters. The weight of the part is a key parameter. The measurement range of the sensor covers the expected weight \pm tolerances. The sensor technically fails in 0.5% of all measurements and returns an error instead of a value. In addition, if the measured value is outside the measurement range the sensor will also return an error.

Missing Values

- What do you think about these values?

```
DER_mass_MMC
p(s|missing)= 0.07374680984253865
p(b|missing)= 0.9262531901574613
```

```
DER_mass_transverse_met_lep
```

```
DER_mass_vis
```

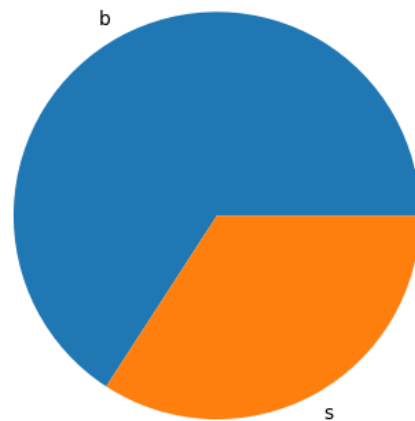
```
DER_pt_h
```

```
DER_deltaeta_jet_jet
p(s|missing)= 0.2982405950507796
p(b|missing)= 0.7017594049492204
```

```
DER_mass_jet_jet
p(s|missing)= 0.2982405950507796
p(b|missing)= 0.7017594049492204
```

```
DER_prodelta_jet_jet
p(s|missing)= 0.2982405950507796
p(b|missing)= 0.7017594049492204
```


- Yet another issue of the dataset.
- About 66% label "b", 34% label "s".
- We will deal with it later. Just keep that in mind when interpreting your conditional probabilities!



Treating Missing Values

How to treat missing values?


- First, check whether data is missing completely at random.
- That means $P(\text{missing})$ is independent of the other data values.
- Then you have several opportunities to cope with missing data
 - ▶ Leave out rows with missing values (if you have enough data)
 - ▶ Impute missing values.
- Otherwise, you need to take missing values into account. For example,
 - ▶ Introduce an indicator variable
 - ▶ Use ML algorithms that can deal with missing values (decision trees like CART)

- Several approaches are used:
- Constant replacement value (we saw this in the dataset).
- Average values (Avoid data leakage!)
- Learn the missing values from the other data using an ML algorithm or a statistical model. (Avoid data leakage!)
- Some useful pandas methods: `isna()`, `dropna()`, `fillna()`

Unbalanced Datasets

Case Study

- Download the census income dataset [1]. it is a binary classification problem.
- Familiarize with the data.
- Train and evaluate a logistic regression classifier on all features.
- You may face some issues with categorical features, being specified as string. Use `OrdinalEncoder` and `LabelEncoder` classes from `scikit-learn`.
- Avoid data leakage!
- For simplicity in this case study (!), replace missing values with -1.
- How good is your model?

 Census-Income (KDD) [Dataset], UCI Machine Learning Repository,
<https://doi.org/10.24432/C5N30T> (2000).

Reminder — Precision and Recall

The measures **precision** and **recall** are defined as follows:

$$\text{precision} = \frac{tp}{tp + fp} \quad (73)$$

The precision is the fraction of correct classifications among all classifications for class 1.

$$\text{recall} = \frac{tp}{tp + fn} \quad (74)$$

The recall is the fraction of correct classifications among all truly positive samples. *What values do you expect for your dataset in case the minority class is classified all wrong?*

Balanced Accuracy

The **balanced accuracy** is the (arithmetic) mean of true positive rate (sensitivity) and true negative rate (specificity):

$$\text{sensitivity} = \frac{tp}{tp + fn} \quad (75)$$

The precision is the fraction of correct classifications among all classifications for class 1.

$$\text{specificity} = \frac{tn}{tn + fp} \quad (76)$$

The denominators count the truly positive (or negative) samples.
Now the balanced accuracy is given by:

$$\text{balanced accuracy} = \frac{1}{2} \left(\frac{tp}{tp + fn} + \frac{tn}{tn + fp} \right) \quad (77)$$

For treating imbalance, basically two methods can be applied, **undersampling** and **oversampling**.

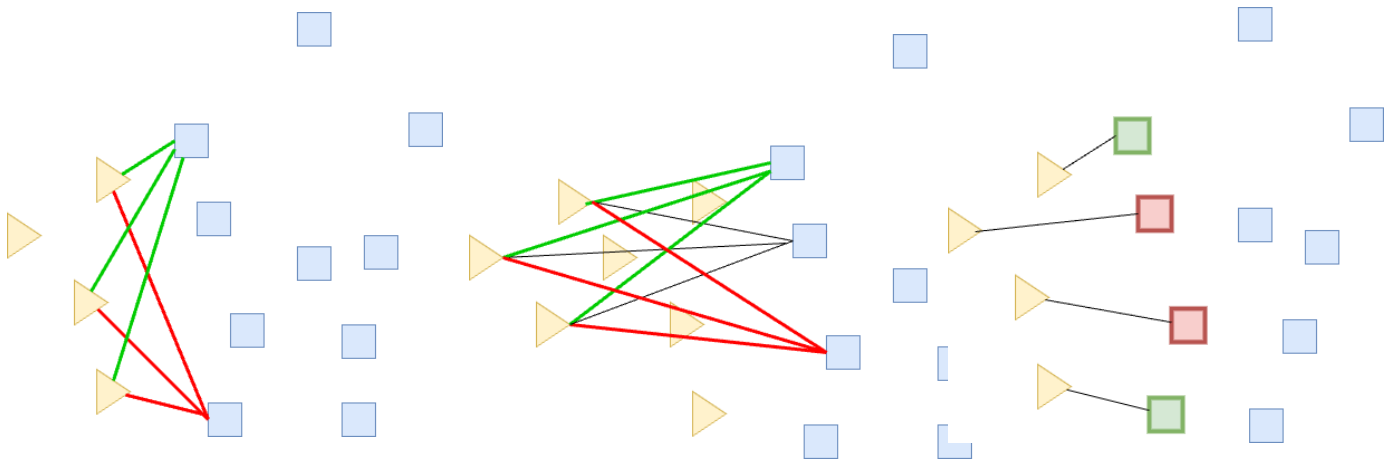
- Undersampling
 - ▶ Miority class: The class with the least number of samples, N_{\min} .
 - ▶ The other class(es) are called majority class(es).
 - ▶ Undersampling basically mean, leave away samples from the majority classes.
 - ▶ In the simplest form, randomly sample N_{\min} samples per class.
 - ▶ Is it a strategy for our census dataset from the case study?

Undersampling

- Undersampling can only be used if you can allow throwing away data.
- Anyway, there are smarter ways to keep the most interesting samples from the majority classes in your dataset.
- "Cleaning strategies" try to find samples from the majority clases which are either clearly classified and therefore less relevant to be kept or noisy and thus rather misleading.
- "Controlled undersampling" uses heuristicss to select samples.
- A particular controlled undersampling strategy is the "near miss" approach, which uses k nearest neighbors to select samples.

Undersampling

k Nearest Neighbors Heuristics



Facing Imbalance

- Oversampling
 - ▶ The opposite approach, upsample the minority classes.
 - ▶ Artificially increase the number of the underrepresented class(es) by drawing samples with replacement.
 - ▶ This is a typical **bootstrapping** approach.

- In statistics, bootstrapping refers to a method which generates more data for training than you actually have, in that sense pulling yourself up by your own bootstraps.
- \mathcal{T} , select $N \geq |\mathcal{T}|$ items of your dataset \mathcal{T} with replacement
- This way, you increase your dataset while still obeying the same data distribution.



Image source: https://de.wikipedia.org/wiki/Bootstrapping_%28Informatik%29 (public domain)

Bootstrapping for Uncertainty Estimation


You can also use bootstrapping to estimate uncertainty in your training process or to estimate the training error:


- Given a dataset \mathcal{T} , select $|\mathcal{T}|$ items of your dataset \mathcal{T} with replacement.
- Repeat this selection B times to get B datasets of size $|\mathcal{T}|$, which contain the items of \mathcal{T} .
- This way, you create B training datasets.
- Train a machine learning model \mathcal{M} on each of the B datasets, ending with a set of models $\mathcal{M}_1(\mathcal{T}_1), \mathcal{M}_2(\mathcal{T}_2), \dots, \mathcal{M}_B(\mathcal{T}_B)$, each of which is trained on another one of the bootstrap datasets.
- Create B different predictions $p_1(X), \dots, p_B(X)$ for a certain but fixed input X from your B different models and determine empirical mean and variance of the predictions:

$$\bar{p} = \frac{1}{B} \sum_{b=1}^B p_b, \quad \text{Var}(p) = \frac{1}{B-1} \sum_{b=1}^B (p_b - \bar{p})^2 \quad (78)$$

Oversampling

- Easy strategy: random oversampling.
- The minority class(es) are sampled with replacement until the number of samples corresponds for all classes to that of the majority class.
- More advanced strategies create new samples by interpolation. However, there are various strategies.
 - ▶ SMOTE (Synthetic Minority Oversampling TEchnique) [1]
 - ▶ ADASYN (ADAPtive SYNthetic oversampling) [2]
- Both generate new samples by using nearest neighbors of samples in the minority classes.
- The target value is determined by the k-nearest neighbor algorithm.
- ADASYN focuses on samples that are wrongly classified using a k-NN approach.

 N.V. Chawla, et.al., Smote: synthetic minority over-sampling technique. Journal of artificial intelligence research, 16:321-357 (2002).

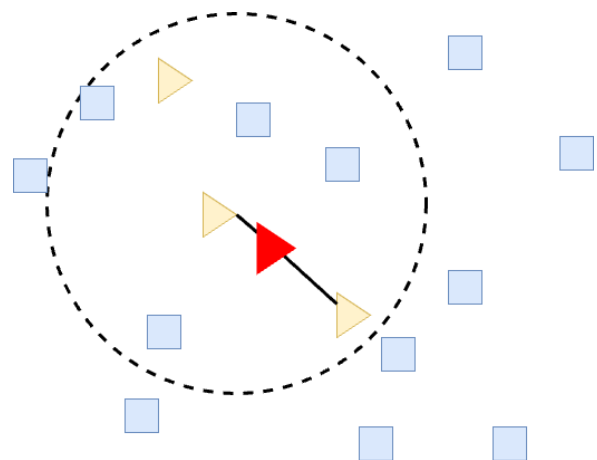
 H. He, et.al., Adasyn: adaptive synthetic sampling approach for imbalanced learning. In 2008 IEEE International Joint Conference on Neural Networks, IEEE World Congress on

Oversampling

- For the samples x_i of the minimum class, consider its k nearest neighbors.
- SMOTE selects a nearest neighbor x_{ni} randomly and creates a new sample according to

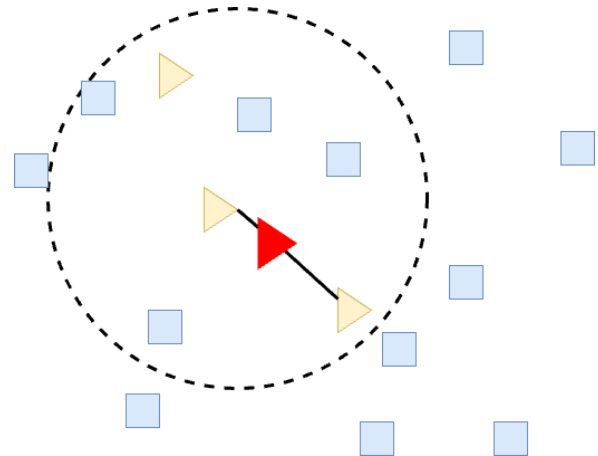
$$x_{\text{new}} = x_i + \lambda(x_{ni} - x_i) \quad (\lambda \in [0; 1]) \quad (79)$$

- Until the required number of samples to be generated is reached.
- Different variations of SMOTE exist.



Oversampling

- ADASYN uses the same interpolation rule.
- But, ADASYN creates the required number according to the neighborhood of each minority class sample x_i .
- It considers the ratio of majority class samples divided by total samples in the neighborhood of x_i and creates new samples according to that fraction.
- Therefore, ADASYS puts more focus on those samples, which could be harder to train.
- The parameter k is a tuning parameter for both algorithms.



Oversampling


Practical Part

- Consider oversampling for the census dataset.
- Benchmark a linear model (logistic regression) for all three oversampling methods.
- Use cross-validation.
- Try different values for $k_neighbors$.

- Until now, we have worked with linear regression and logistic regression, both methods assume (at least approximately) linear relationships between features and target.
- Most of our world is non-linear.

Practical Part - Dataset

- Open Notebook No. 7 and familiarize with the "liver disorders dataset" [1].
- Perform a brief data exploration.
- Try to predict the target variable using linear regression. What do you observe?

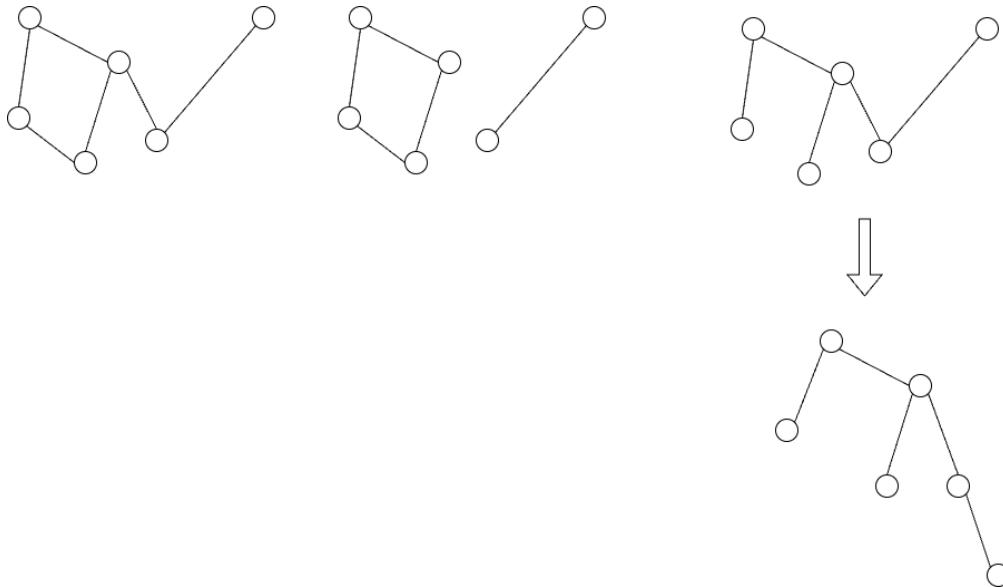
 Liver Disorders [Dataset]. (2016). UCI Machine Learning Repository.
<https://doi.org/10.24432/C54G67>.

Non-Linear Methods

- Often, linear models are not sufficient for real-world processes.
- Machine learning methods need to cover non-linear relationships.
- We will look at two approaches, which are suitable for non-linear relationships, trees and ensemble methods and kernel methods.

Decision Trees

Trees are graphs. A graph is a tuple (V, E) of a set of nodes V and a set of edges E .

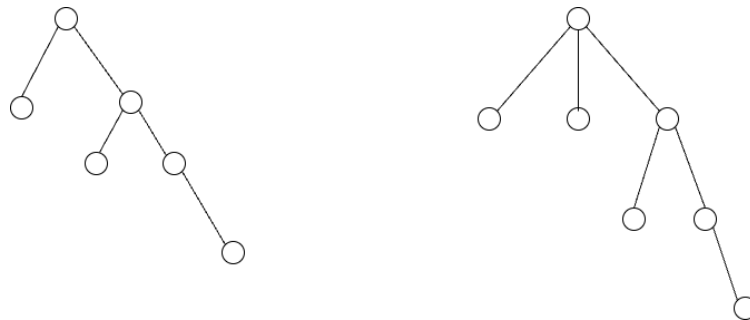


Decision Trees

- In a tree you can select one node to be the starting point, also called the *root* node of the tree.
- The root node has no *predecessors* but only *successors*.
- Nodes without successors are called *leaves* of the tree (to complete the botanical analogy).
- Nodes which are neither the root nor leaves have at least one predecessor and at least one successor.

Decision Trees

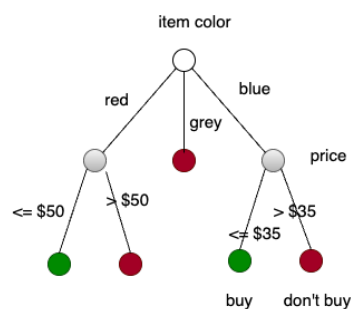
- In case all nodes in the tree have at maximum two successors the tree is called a *binary tree*.
- The following figure shows a binary tree on the left and a non-binary tree on the right.



(The big arrow indicates that the graph is isomorphically transformed to look more like a usually drawn tree.)

Decision Trees

- We will use the tree to model classification and regression problems.
- The basic idea is to use a node per feature.
- An edge is a certain value for that feature (categorical feature) or a range of values (continuous features)
- See Chapter 3 in the book of Tom Mitchell.
- Example:



Decision Trees

Guiding Example

(Mitchell Table 3.2)

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Decision Trees

Suitable problems

- Trees are well suited for a limited number of features
- Which ideally have few distinct, discrete values.
- However, they work with continuous values, too.
- From their basic idea they are well-suited for classification problems.
- However, it is possible to extend the method to continuous valued targets (regression trees).
- Tree methods usually are robust with respect to outliers in the data
 - ▶ classification error
 - ▶ erroneous feature values
 - ▶ missing feature values

- Tree methods are applied for regression problems, too. However, their natural application is classification since they have a finite set of leaf nodes.
- To be able to extend to regression, the regression problem needs a discrete formulation:
- The range of values is subdivided into a finite set of regions $\{R_m\}_{m=1}^M$.
- Within each region R_m the regression problem is solved by a simple model.
- Most simple model: a constant c_m :

$$y_i = \sum_{m=1}^M c_m 1(x_i \in R_m) \quad (80)$$

$$y_i = \sum_{m=1}^M c_m 1(x_i \in R_m) \quad (81)$$

- The constant c_m is determined from the training set as

$$c_m = \frac{1}{|\{x_i | x_i \in R_m\}|} \sum_{\{y_i | x_i \in R_m\}} y_i \quad (82)$$

- The main challenge for this algorithm is to find good split variables and split values to subdivide the feature space into the R_m regions.
- In the following we will consider our guiding example, which is a binary classification. As before, this simple case makes understanding the concepts easier.
- Extension to multi-class classification is straight forward. For regression we gave a hint regarding the methodology.

- One of the popular algorithms to build up a decision tree is the ID3 (Iterative Dichotomiser 3).
- It was further developed into C4.5 algorithm.
- An adaption allowing also for regression problems is the CART (classification and regression trees) algorithm.
- We will look into the basic principle of how these algorithms work elaborating on our guiding example with ID3.

- First step is to determine which feature to use at the root node.
- We choose the feature, which among all features classifies the data best.
- How do we measure that? We need a metric to measure **information gain**.

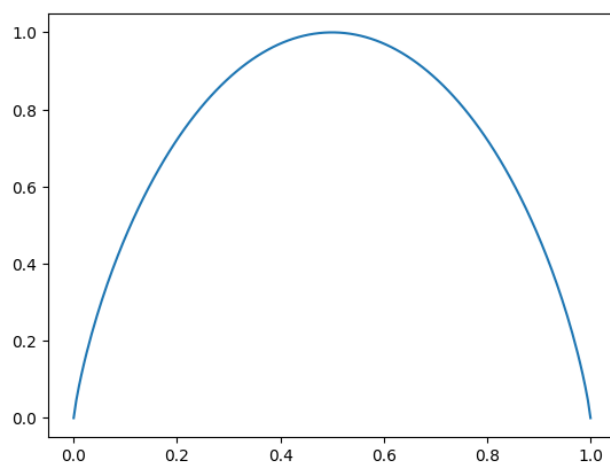
- A widely use measure in information theory is the entropy.
- Consider a binary classification setting with p_0 being the probability of class 0, while $p_1 = 1 - p_0$ is the probability of class 1. Then the entropy S is defined as

$$S = -p_0 \log_2 p_0 - p_1 \log_2 p_1 \quad (83)$$

- Note that $0 \log 0 = 0$.

Entropy

$$S = -p_0 \log_2 p_0 - p_1 \log_2 p_1 \quad (84)$$



- Generalization to an arbitrary number of classes C is straightforward:

$$S = - \sum_{i=0}^C p_i \log_2 p_i \quad (85)$$

- Coming back to our problem:
- We are searching a measure for information gain.
- Entropy is minimal in case the result is known for sure.
- Information gain = reduction of entropy

Information Gain

- If we are to select a feature for the root node, we split our dataset T according to one feature F , means we build subsets $\{T_v\}_{v \in \text{Values}(F)}$ where v are the possible values of feature F .
- The split has an entropy

$$S_F = \sum_{v \in \text{Values}(F)} \frac{|T_v|}{|T|} S(T_v) \quad (86)$$

- where $S(T_v)$ is the entropy on subset T_v for our classification problem.
- That means, if F is a feature that is indicative for the classification, it will group together data samples with equal class.

- Thus, it will reduce the entropy for many T_v subsets while not causing a corresponding increase in the others.
- Therefore it will reduce the entropy on the whole set T .
- Information gain for splitting the data according to feature F is therefore given by

$$\text{Gain}(F) = S - S_F = S - \sum_{v \in \text{Values}(F)} \frac{|T_v|}{|T|} S(T_v) \quad (87)$$

$$\text{Gain}(F) = S - S_F = S - \sum_{v \in \text{Values}(F)} \frac{|T_v|}{|T|} S(T_v) \quad (88)$$

- Let's consider an example from our guiding problem:
- From our 14 days we don't play tennis of 5. So $p_0 = 5/14$.
- The corresponding entropy is approx. 0.94
- Let's assume we split our data according to the feature $F = \text{"wind"}$, which has the values "weak", "strong"

$$\text{Gain}(F) = S - S_F = S - \sum_{v \in \text{Values}(F)} \frac{|T_v|}{|T|} S(T_v) \quad (89)$$

- We compute the information gain for that feature.

- ▶ wind = weak: $|F_{\text{weak}}| = 8$, $p_0 = 2/8 = 1/4$.
- ▶ wind = strong: $|F_{\text{strong}}| = 6$, $p_0 = 3/6 = 1/2$.
- ▶ entropy

$$S_{\text{wind}} = \frac{8}{14} S(1/4) + \frac{6}{14} S(1/2) \approx 0.89 \quad (90)$$

- ▶ Information gain: $S(5/14) - S_{\text{wind}} \approx 0.048$

Decision Tree

Practical Part

Compute the information gain for all features and select the feature for the root node of our decision tree.

Hint: Implement the entropy function in Python to re-use it.

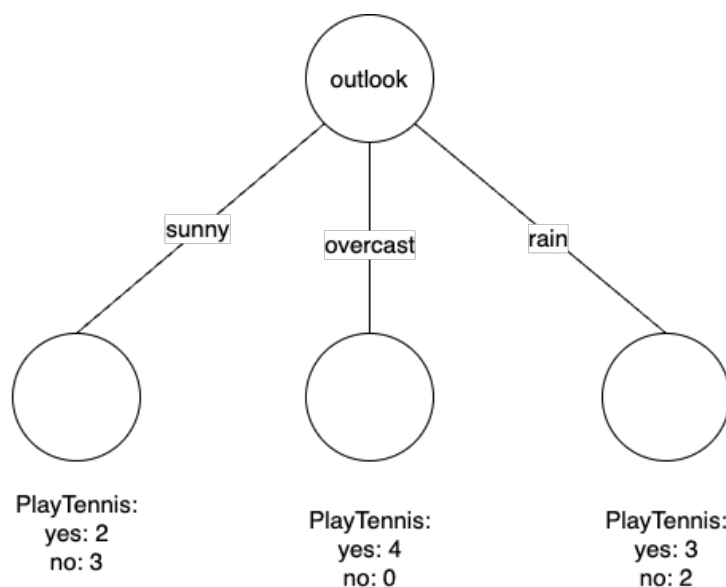
Practical Part - Solution

Compute the information gain for all features and select the feature for the root node of our decision tree.

$$\begin{aligned}S(5/14) - S_{\text{wind}} &\approx 0.048 \\S(5/14) - S_{\text{humidity}} &\approx 0.141 \\S(5/14) - S_{\text{outlook}} &\approx 0.256 \\S(5/14) - S_{\text{temperature}} &\approx 0.029\end{aligned}$$

Decision Tree

The resulting head of our tree is shown here:



Practical Part continued

Determine the next layer of nodes of the tree.

Practical Part continued - Solution

Quick solution:

- "sunny" node:

$$S(2/5) - S_{\text{wind}} \approx 0.019$$

$$S(2/5) - S_{\text{humidity}} \approx 0.97$$

$$S(2/5) - S_{\text{temperature}} \approx 0.57$$

- "overcast" node: tree ends here
- "rain" node: wind is decisive

- One big advantage of decision trees, you can read out its hypotheses for the classification, e.g.
 - ▶ if outlook=overcast then play tennis OR
 - ▶ if outlook=rain AND wind=weak then play tennis OR
 - ▶ ...
- You get hypotheses which are AND-concatenated values of features, which themselves are OR-concatenated (a disjunction of conjunctions).
- Though, in practical problems, these hypotheses might be quite unreadable...

Gini Criterion

An alternative criterion to measure the information gain is to measure the change of the Gini index instead of the entropy change.

The Gini index looks similar to the entropy but does not use the log function:

$$S_{\text{Gini}} = - \sum_{i=0}^C p_i (1 - p_i) \quad (91)$$

Practical Part final

Solve the example problem in Python.

- Familiarize with the sklearn class `DecisionTreeClassifier`.
- To this end, transform the data into a dataframe. You find the data below as a markdown formatted string.
- Check out the `OrdinalEncoder` from `sklearn.preprocessing`.
- Visualize your decision tree. Check out the `plot_tree()` method in `sklearn.tree`.

Thoughts on Decision Trees

The algorithm you implemented by hand was the so-called ID.3 algorithm.

- In principle, finding a decision tree consistent with our training data is not a problem with a unique solution.
- Usually, there exist many such trees.
- Is ID3 able to find the tree that generalizes best? Does it find the tree with the best performance on the training dataset?
- ID3 does not revise decisions met once: As soon as a feature for a node is found it is not backtracked at a later stage whether another choice would finally be better ("greedy" behavior).

Thoughts on Decision Trees

The algorithm you implemented by hand was the so-called ID.3 algorithm.

- This may lead to a local optimum which is not globally optimal.
- The way how ID3 meets its decisions favors shorter trees (as much information gain as possible at the beginning).
- However, it is not looking for the shortest possible tree but for those with the most informative features closest to the root.
- In principle, shorter trees usually imply simpler hypotheses. Usually, a simpler hypothesis is preferable compared to a longer, and thus more complex one.
- In philosophy, this discussion is linked with the term *Occam's razor* or *principle of parsimony*: Prefer the explanation which can be constructed with the smallest number of elements.

Overfitting in Trees

- Overfitting in trees can be a considerable problem.
- The hypothesis found by the decision tree can be too closely adapted to the specifics of the training dataset.
- Can be caused either by noise in the dataset
- Or by special cases
- Or by outliers.

- Reduced error pruning: Use a validation set to remove nodes Remove the subtree of a node and let the sub-tree node become the new leaf (majority voting the prediction). This is done only if it improves validation set performance.
- The improvement of ID3 (C4.5) considered **rule post pruning**:
 - ▶ Take every rule from the decision tree and remove conditions if they improve accuracy (validation set).
 - ▶ Rules are applied for prediction in the order of their accuracy.

- Take every rule from the decision tree and remove conditions if they improve accuracy (validation set).
- Rules are applied for prediction in the order of their accuracy.



- Example: For this rule set
 - ▶ if outlook=overcast then play tennis OR
 - ▶ if outlook=rain AND wind=weak then play tennis
- we would ask ourselves
 - ▶ can the condition "outlook=rain" be removed in the second rule and improve accuracy?
 - ▶ can the condition "wind=weak" be removed in the second rule and improve accuracy?

- A generally good method to counter overfitting is limitation of the complexity of the model.
- In case of a decision tree, one could *limit the tree depth*.
- Demand a *minimum of training samples to be represented by a leaf node*.
- Instead of pruning one prohibits leaf nodes which represent less than a configurable limit of samples.
- Usually, the ML software packages provide corresponding parameters in their API.

Practical Part - Liver Disorder Dataset

- Use a decision tree regressor for the liver dataset

Random Forests

- Are two ML models better than one?
- How to combine models?
- There have been and still are lots of efforts put into the question, how to combine ML models to get a better prediction on unseen data.
- Approaches: Hierarchical mixture of experts (still popular with LLMs: Mixtral)
- Decision trees can be combined to decision forests.

One technique for combining models is bagging.

- Basically speaking, to make a prediction \hat{y} for a set of input values X you average several models trained on bootstrap samples.
- Let f be a model to be trained on a training dataset T .
- Train f on a number of B bootstrap samples from the data (draw a dataset of full size with replacement).
- Training sets T_1, \dots, T_B yield models f_1, \dots, f_{T_B}

- Predict with every model $f_1(X), \dots, f_B(X)$
- For regression average their results: $\hat{y} = \frac{1}{B} \sum_{i=1}^B f_i(X)$
- For classification either use a majority vote, or average the model-internal classification scores
- *Example (2 models, binary classification): Model one predicts class 1 with probability 0.7, class 2 with probability 0.3, while model 2 predicts class 1 with 0.52, class 2 with 0.48. Majority vote would predict class 1 for sure, while the second approach predicts class 1 with a probability of 0.61.*
- To construct random forests, every model f_i is a decision tree.
- For classification, usually the second approach is selected.
- The class probability emerges from the number of samples in each leaf.

Mathematical Consideration

Bagging improves on the mean squared error.

Argument by Hastie, Tibshiranie, Friedman, chapter 8.7

- Consider a distribution P from which we draw our data.
- We train a bagging estimator on bootstrap samples directly sampled from P .
- Let f^* be a single estimator out of the bagging set and $f_{\text{bag}} = \mathbb{E}_P(f^*)$ be the mean over all bagging estimators.
- Let (x, y) be a sample from P .
- The mean squared error of our prediction using model f^* is given by

$$\text{MSE}(y, f^*(x)) = \mathbb{E}_P(y - f^*(x))^2 = \mathbb{E}_P(y + f_{\text{bag}} - f_{\text{bag}} - f^*(x))^2 \quad (92)$$

$$= \mathbb{E}_P(y - f_{\text{bag}})^2 + \mathbb{E}_P(f^*(x) - f_{\text{bag}})^2 \quad (93)$$

$$\geq \mathbb{E}_P(y - f_{\text{bag}})^2 \quad (94)$$

Mathematical Consideration

$$\begin{aligned} \text{MSE}(y, f^*(x)) &= \mathbb{E}_P(y - f^*(x))^2 = \mathbb{E}_P(y + f_{\text{bag}} - f_{\text{bag}} - f^*(x))^2 \\ &= \mathbb{E}_P(y - f_{\text{bag}})^2 + \mathbb{E}_P(f^*(x) - f_{\text{bag}})^2 \\ &\geq \mathbb{E}_P(y - f_{\text{bag}})^2 \end{aligned}$$

For the second equality we use

- $\mathbb{E}_P(f^* f_{\text{bag}}) = f_{\text{bag}}^2$ since f_{bag} is constant.
- $\mathbb{E}_P(y f^*) = \mathbb{E}_P(y f_{\text{bag}})$ since y and f^* are independent and using the previous equation.

$$\text{MSE}(y, f^*(x)) = \mathbb{E}_P(y - f_{\text{bag}})^2 + \mathbb{E}_P(f^*(x) - f_{\text{bag}})^2 \geq \mathbb{E}_P(y - f_{\text{bag}})^2$$

- From this equation we see that aggregation of estimators may decrease the mean squared error of the prediction.
- Random forests have a variance reduction effect. (Single trees are quite noisy in their predictions.)
- Noise is decreased by the averaging process.
- Combination with cross-validation: Out-of-box (OOB) samples: Use only those tree with a bootstrap training dataset that does not contain the validation sample $(x^*, y^*) \in T$ (T : complete dataset).
- Therefore, cross-validation can be done along the training.
- However, OOB estimates are considered too pessimistic and thus they are only used instead of CV in case CV is too time consuming.

Stacking

A further technique for combining models is stacking.

- Stacking works with an arbitrary number M of different models f_1, f_2, \dots, f_M . - The models can either be of the same kind with different hyperparameters or be completely different (e.g. decision tree and logistic regression).
- The idea is to build an average model for the final prediction, in other words, a linear regression model taking the outputs of each of the M models as inputs. So we are looking for the weight vector \hat{w} such that

$$\hat{w} = \operatorname{argmin}_w \mathbb{E}_P \left[Y - \sum_{m=1}^M w_m f_m(X) \right]^2 \quad (95)$$

with data (X, Y) being distributed according to P .

- The resulting model is never worse than any individual model

$$\mathbb{E}_P \left[Y - \sum_{m=1}^M \hat{w}_m f_m(X) \right]^2 \leq \mathbb{E}_P [Y - f_m(X)]^2 \forall m \quad (96)$$

- Otherwise \hat{w} were not the argmin.
- In practice, the expectation value for the regression is calculated using averaging with leave-one-out cross-validation.

- Let f_m^{-i} denote model m fitted to the training dataset $\{(x_j, y_j)\}_{j=1}^N$ without sample i .
Then

$$\hat{w} = \operatorname{argmin}_w \sum_{i=1}^N \left[y_i - \sum_{m=1}^M w_m f_m^{-i}(x_i) \right]^2 \quad (97)$$

- In principle, the "main" model, which is the one being trained on the predictions f_m from the different models needs not be a linear regression.
- Practitioners use all kind of models to train them on these predictions.

Practical Part - Liver Disorder Dataset

- Use a random forest regressor for the liver dataset.
- Try stacking decision trees on your own.
- Mutually compare the results, and compare with previous approaches.
- Do not only focus on metrics, but also look at wrongly classified samples.
- Ask yourself: Are there specific parts of my feature space that are particularly incorrectly classified, are those regions the same for linear models, decision tree and forests, ...?