DIGITAL CULTURE
ITMO UNIVERSITY

Ensemble Learning

High School of Digital Culture
ITMO University
dc@itmo.ru

Contents

1 Heuristic Arguments

Hello everyone! By now you've learned multiple data processing techniques using which you can, for example, identify the most important features, reduce data dimensionality, construct different classifiers, perform regression analysis, evaluate the accuracy of a constructed model, and many more. Earlier, when solving problems, we followed a certain pattern. We took a method (or an algorithm), adjusted its parameters on input data, evaluated the accuracy, and, if it is high enough, the problem is solved. But how appropriate or effective is the described approach?

What are we trying to say? Well, none of the machine learning methods is universal. Each method works well on certain data, or in a certain case. It's naive to assume that one method can handle any dataset. Most algorithms assume that data satisfies a large set of synthetic conditions, which is rare in practice.

In this module, we will focus on ensembles that combine multiple models. The idea of ensembles isn't new, and it is inherently understood. It's like collecting expert opinions on some matter to create a rule. Then, this rule is applied to new data to make a conclusion. In machine learning, the experts are different algorithms trained on the same data, and the rule is a conclusion made based on the set of predicted answers. We will discuss all these things a little bit later.

One more way to understand the ensembles is the parable of the Blind Men and an Elephant[1]. It is a story of a group of blind men, each having a different, but correct description of the elephant, although based on their limited experience . The best thing to do would be to gather opinions and discuss differences before making a conclusion.

We are strongly motivated to learn. However, here's the 'but'. We need different datasets to train different models. However, in practice, we usually have one sample. Can we obtain some other, so-called pseudosamples? Well, let's begin our discussion with this purely statistical issue.

2 Resampling

2.1 Introduction

You already know that many machine learning methods are based on the apparatus of mathematical statistics. But what do statistics do? Well, statistics deal with inferring the characteristics of the population. To handle these characteristics in practice, obtain their concrete values, or analyze, compare, and use

[1]Elephant and blind sages by Blanca Marti for Equilibre.
https://wildequus.org/2014/05/07/sufi-story-blind-men-elephant/

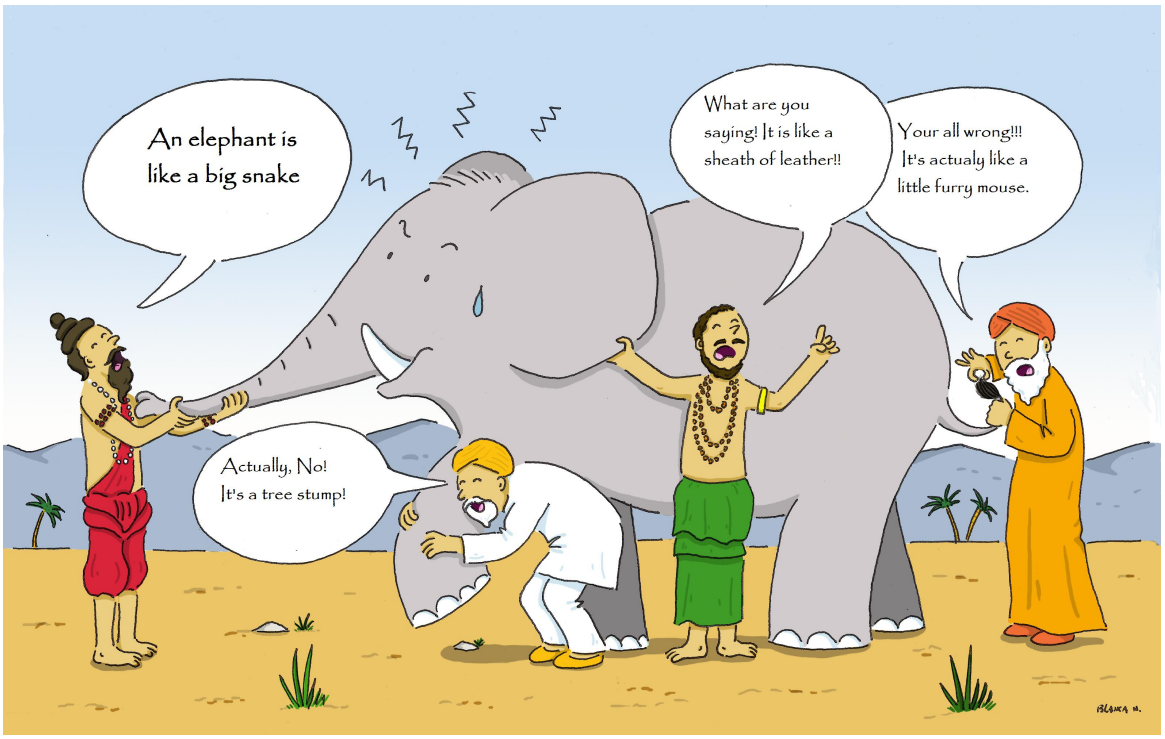


Figure 1: The Blind Men and the Elephant.

them to solve applied tasks, the statistical apparatus requires the instances of the population called a sample. Resampling methods, which we are going to discuss, allow obtaining samples (as follows from the name). You may wonder what kind of samples they are.

Before we move on to this (and give some examples), let's start from the very beginning and explain what resampling methods are, and why do we need them. Resampling methods, or methods of drawing repeated samples, allow us to create a set of new samples from the original sample. But why do we need it? It is a good question that we will elaborate on later, but now we can answer it for practical purposes, or more accurate estimation. Here's the plan. For each obtained sample, we calculate the statistic of interest, average the obtained values, and get the value that is not worse than that calculated using the original sample.

Thus, resampling methods are methods of modeling the population behavior based on a sample from it. Well, let's emphasize it. Unlike in the methods discussed earlier, new samples are not drawn from the population, but they are formed from the original sample drawn from the population. We used this technique before. For example, cross-validation that we discussed before is a resampling method.

Now we can focus on why. Why is the original sample bad? Actually, it's not bad, but it leads to many problems. Let's name a few.

1. A wrong assumption about a population distribution. Many statistical methods are based on some assumptions about a population distribution.

However, when the assumption is not correct, or the sample size is small (which can happen due to various reasons), parametric statistical methods based on theoretical, well-studied distributions will not be much of a use. In this case, it would be better to consider a so-called non-parametric model.

2. Non-random samples. The classical, 'sterile' assumption of many statistical methods is that a sample is random. However, sometimes a sample is not random at all. In practice, it's much faster and cheaper to collect such samples (for example, self-selected samples). With this in mind, how to apply the central dogmas of mathematics?

3. Small sample size. To obtain satisfactory results, many statistical methods require a sufficient sample size (because results are usually asymptotic when $n \to +\infty$). When the sample size is small, methods either do not work, or they are far from being accurate.

4. No way to calculate the desired characteristics of a statistic.

To shed the light on the last problem, let's consider a purely theoretical example. Let X_1, X_2, …, X_n be a sample of size n from the population with a uniform distribution $\mathsf{U}_{0,\theta}$ on the interval $[0, \theta]$, where $\theta > 0$. How to estimate the parameter θ?

The expected value of a random variable ξ with a uniform distribution $\mathsf{U}_{a,b}$ on the interval $[a, b]$ is

$$\mathsf{E}\xi = \frac{a+b}{2}.$$

Thus, for the random variable ξ with a distribution $\mathsf{U}_{0,\theta}$,

$$\mathsf{E}\xi = \frac{\theta}{2} \Rightarrow \theta = 2\mathsf{E}\xi.$$

It is also known that a good estimator of the expected value of the random variable with a distribution of any form is a sample mean $\overline{X}$. Thus, to estimate the parameter θ, we can use the following statistic:

$$\widehat{\theta} = 2\overline{X}.$$

We can calculate the basic characteristics of the written statistic including expected value (or mean), variance, and standard deviation. We can also construct the asymptotic confidence interval and do many more things.

Remark 2.1.1 *A better estimator (maximum-likelihood estimator) of the parameter θ is the nth term of the variational series $X_{(n)}$:*

$$\widehat{\theta} = X_{(n)},$$

for which we can also calculate the described characteristics, but it's technically more complicated and will not contribute to our discussion.

On the other side, we can use the well-known method of moments to offer other estimators of the parameter θ, for example:

$$\widehat{\theta} = \sqrt{3 \cdot \overline{X^2}} = \sqrt{3 \cdot \frac{X_1^2 + \ldots + X_n^2}{n}}.$$

It's not easy to obtain the expected value or variance for the written statistic analytically. But resampling methods can help. The two methods that we are going to consider are jackknife and bootstrap.

2.2 Jackknife

2.2.1 About the Method

Jackknife is a resampling method originally developed by Maurice Henri Quenouille who had given corrections for the bias of the statistic $\widehat{\theta}$ for small n. Later, John Wilder Tukey discovered that this method can be used to construct accurate characteristics of the statistic $\widehat{\theta}$ and even to construct confidence intervals. That's why the method is called jackknife (as proposed by Tukey). Jackknife derives its name from the resemblance to a compact folding knife, which is a rough-and-ready tool that can improvise a solution for a variety of problems. In fact, the jackknife is designed to replace various purpose-designed methods.

Well, let's move on to the method. Assume that the statistical experiment has provided a sample $X_1, X_2, ..., X_n$ of size n from the population of ξ.

Remark 2.2.1 *An ordinary sample $X_1, X_2, ..., X_n$ from a statistical population ξ is a set of independent random variables distributed identically to ξ.*

Now let's describe how samples are augmented using the jackknife. We will begin with the definition.

Definition 2.2.1 *Jackknife samples $X_{[1]}, X_{[2]}, ..., X_{[n]}$ are sets formed from the original sample as follows:*

$$X_{[i]} = (X_1, ..., X_{i-1}, X_{i+1}, ..., X_n).$$

To put it differently, ith jackknife sample is obtained from the original one by deleting the ith element. Note that we obtained n pseudosamples instead of one sample.

Well, let θ be a characteristic of the population, and

$$\widehat{\theta} = \widehat{\theta}_n(X_1, X_2, ..., X_n)$$

be its estimator based on the sample $X_1, X_2, ..., X_n$. Let's introduce the next definition.

Definition 2.2.2 *A partial estimator* $\widehat{\theta}_{(-i)}$, $i \in \{1, 2, ..., n\}$, *of the parameter* θ *on the jackknife sample* $X_{[i]}$ *is the statistic*

$$\widehat{\theta}_{(-i)} = \widehat{\theta}_{n-1}\left(X_{[i]}\right) = \widehat{\theta}_{n-1}(X_1, ..., X_{i-1}, X_{i+1}, ..., X_n).$$

Well, instead of one estimator of the desired characteristic based on the original sample, we obtained n partial estimators. So, let's consider the example.

Example 2.2.1 *Let the parameter* θ *be the expected value of a population. Without additional information about the distribution, we can reasonably consider a sample mean as its estimator:*

$$\widehat{\theta} = \overline{X_n} = \frac{1}{n}\sum_{i=1}^{n} X_i.$$

Thus, a partial estimator $\widehat{\theta}_{(-i)}$, $i \in \{1, 2, ..., n\}$ *of the parameter* θ *on the jackknife sample* $X_{[i]}$ *is the statistic*

$$\widehat{\theta}_{(-i)} = \overline{(X_{[i]})_{n-1}} = \frac{1}{n-1}\sum_{j=1,\ j\neq i}^{n} X_j.$$

The obtained statistic is nothing but the sample mean of the i*th jackknife sample.*

For convenience, let's introduce the following definition for an arithmetic mean of partial estimators:

$$\widehat{\theta}_{(\bullet)} = \frac{1}{n}\sum_{i=1}^{n} \widehat{\theta}_{(-i)}.$$

To define a jackknife estimator, we will introduce the concept of a pseudovalue first.

Definition 2.2.3 *The value*

$$\widetilde{\theta}_i = n\widehat{\theta}_n - (n-1)\widehat{\theta}_{(-i)}, \quad i \in \{1, 2, ..., n\}$$

is called the i*th pseudovalue of the parameter* θ *constructed based on a jackknife sample* $X_{[i]}$.

Now it should be clear how a jackknife estimator is defined.

Definition 2.2.4 *A jackknife estimator* $\widehat{\theta}_{jack}$ *of the parameter* θ *is computed as:*

$$\widehat{\theta}_{jack} = \frac{1}{n}\sum_{i=1}^{n} \widetilde{\theta}_i.$$

Remark 2.2.2 *Let's rewrite the last definition in another form. Since* $\widetilde{\theta}_i = n\widehat{\theta}_n - (n-1)\widehat{\theta}_{(-i)}$,

$$\widehat{\theta}_{jack} = \frac{1}{n}\sum_{i=1}^{n}\widetilde{\theta}_i = \frac{1}{n}\sum_{i=1}^{n}\left(n\widehat{\theta}_n - (n-1)\widehat{\theta}_{(-i)}\right) =$$

$$= n\widehat{\theta}_n - \frac{n-1}{n}\sum_{i=1}^{n}\widehat{\theta}_{(-i)} = n\widehat{\theta}_n - (n-1)\widehat{\theta}_{(\bullet)}.$$

Thus, the jackknife estimator of the parameter θ *is* n *original estimators based on the sample* $X_1, X_2, ..., X_n$ *minus* $(n-1)$ *the arithmetic mean of partial estimators.*

Example 2.2.2 *Let's consider an example of calculating an estimator using the jackknife on a given sample. Let* $X = (1, 4, 7, 9)$ *be the sample of size* 4, *and* θ *be an unknown expected value of the population. Let's estimate it using the sample mean* $\overline{X}$, *that is,* $\widehat{\theta} = \overline{X}$. *Thus, we obtain that*

$$\widehat{\theta} = \frac{1+4+7+9}{4} = 5.25.$$

The jackknife sample $X_{[1]}$ *consists of the following elements:*

$$X_{[1]} = (4, 7, 9),$$

and a partial estimator based on it equals:

$$\widehat{\theta}_{(-1)} = \widehat{\theta}(X_{[1]}) = \frac{4+7+9}{3} = \frac{20}{3}.$$

Then, the first pseudovalue can be calculated as

$$\widetilde{\theta}_1 = 4 \cdot 5.25 - 3 \cdot \frac{20}{3} = 1.$$

The remaining jackknife samples are defined in the same way:

$$X_{[2]} = (1, 7, 9), \quad X_{[3]} = (1, 4, 9), \quad X_{[4]} = (1, 4, 7),$$

based on them, the partial estimators are obtained:

$$\widehat{\theta}_{(-2)} = \frac{17}{3}, \quad \widehat{\theta}_{(-3)} = \frac{14}{3}, \quad \widehat{\theta}_{(-4)} = 4,$$

and, after that, the pseudovalues are calculated based on them:

$$\widetilde{\theta}_2 = 4, \quad \widetilde{\theta}_3 = 7, \quad \widetilde{\theta}_4 = 9.$$

As you can see, the obtained pseudovalues match the elements of the original sample. That's why the estimator $\widehat{\theta}_{jack}$ will be the same as the original estimator $\widehat{\theta} = \overline{X}$:

$$\widehat{\theta}_{jack} = \frac{1}{4}\sum_{i=1}^{n}\widetilde{\theta}_i = \frac{1+4+7+9}{4} = 5.25 = \overline{X}.$$

It is also true that the sample mean is invariant to the jackknife.

Example 2.2.3 *Let θ be an unknown expected value of the population. To estimate it, it's convenient to use the sample mean $\overline{X}$. Then,*

$$\widehat{\theta}_n = \overline{X},$$

$$\widehat{\theta}_{(\bullet)} = \frac{1}{n}\sum_{i=1}^{n}\widehat{\theta}_{(-i)} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{n-1}\sum_{j=1,\ j\neq i}^{n} X_j =$$

$$= \frac{1}{n(n-1)}\left((n-1)X_1 + (n-1)X_2 + ... + (n-1)X_n\right) = \frac{1}{n}\sum_{i=1}^{n} X_i = \overline{X},$$

hence,

$$\widehat{\theta}_{jack} = n\widehat{\theta}_n - (n-1)\widehat{\theta}_{(\bullet)} = n\overline{X} - (n-1)\overline{X} = \overline{X}.$$

and the jackknife estimator matches the original one.

It's easy to understand that a more general statement is also true.

Lemma 2.2.1 *Assume that we have the sample $X_1, X_2, ..., X_n$ and the function $f : \mathbb{R} \to \mathbb{R}$. If the estimator of the parameter θ has the form*

$$\widehat{\theta} = \widehat{\theta}_n(X_1, X_2, ..., X_n) = \frac{1}{n}\sum_{i=1}^{n} f(X_i),$$

then

$$\widehat{\theta}_{jack} = \widehat{\theta}.$$

Proof. The proof is straightforward. Let's look at it one more time to recall the introduced notations.

$$\widehat{\theta}_{(\bullet)} = \frac{1}{n}\sum_{i=1}^{n}\widehat{\theta}_{(-i)} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{n-1}\sum_{j=1,\ j\neq i}^{n} f(X_j) =$$

$$= \frac{1}{n(n-1)}\left((n-1)f(X_1) + (n-1)f(X_2) + ... + (n-1)f(X_n)\right) =$$

$$= \frac{1}{n}\sum_{i=1}^{n} f(X_i) = \widehat{\theta}_n.$$

Therefore,

$$\widehat{\theta}_{jack} = n\widehat{\theta}_n - (n-1)\widehat{\theta}_{(\bullet)} = n\widehat{\theta}_n - (n-1)\widehat{\theta}_n = \widehat{\theta}_n.$$

□

It follows from the theorem that for the estimators of, for example, moments of a random variable, the jackknife gives nothing new because the estimator of the kth moment has the form:

$$\widehat{\theta} = \widehat{\theta}_n(X_1, X_2, ..., X_n) = \frac{1}{n}\sum_{i=1}^{n} X_i^k$$

and satisfies the theorem condition given that $f(x) = x^k$.

At the same time, if the considered statistic $\widehat{\theta}$ has a more complicated structure, the jackknife method can reduce its bias (if applicable). Let's talk about it in detail.

2.2.2 Jackknife Estimator of Bias

Well, let's begin by studying the properties of the jackknife estimator. Since

$$\widehat{\theta}_{jack} = n\widehat{\theta}_n - (n-1)\widehat{\theta}_{(\bullet)},$$

it's reasonable to consider the value

$$\widehat{\theta}_{jack} - \widehat{\theta}_n = (n-1)(\widehat{\theta}_n - \widehat{\theta}_{(\bullet)}),$$

which shows the deviation of the estimator from the jackknife estimator.

Definition 2.2.5 *The value*

$$\widehat{\mathsf{bias}}_{jack} = (n-1)(\widehat{\theta}_n - \widehat{\theta}_{(\bullet)})$$

is a jackknife estimator of the bias of the statistic $\widehat{\theta}_n$.

It's logical to assume that if an original estimator is unbiased, the obtained estimator will also be unbiased. Well, this assumption is correct.

Lemma 2.2.2 *Let* $\widehat{\theta}_n$ *be an unbiased estimator of the parameter* θ*, that is,*

$$\mathsf{E}_\theta \widehat{\theta}_n = \theta.$$

Thus, $\widehat{\theta}_{jack}$ *is also an unbiased estimator of the parameter* θ.

Proof. Since

$$\widehat{\theta}_{jack} - \widehat{\theta}_n = (n-1)(\widehat{\theta}_n - \widehat{\theta}_{(\bullet)}) = \widehat{\mathsf{bias}}_{jack},$$

then

$$\widehat{\theta}_{jack} = \widehat{\theta}_n + \widehat{\mathsf{bias}}_{jack}.$$

Then,

$$\mathsf{E}_\theta \widehat{\theta}_{jack} = \mathsf{E}_\theta \widehat{\theta}_n + \mathsf{E}_\theta \widehat{\mathsf{bias}}_{jack} = \theta + \mathsf{E}_\theta \widehat{\mathsf{bias}}_{jack},$$

where the last equality is true due to the unbiasedness of the estimator $\widehat{\theta}_n$. However,

$$\mathsf{E}_\theta \widehat{\mathsf{bias}}_{jack} = \mathsf{E}_\theta \left((n-1)(\widehat{\theta}_n - \widehat{\theta}_{(\bullet)}) \right) = (n-1)(\mathsf{E}_\theta \widehat{\theta}_n - \mathsf{E}_\theta \widehat{\theta}_{(\bullet)}) =$$

$$= (n-1) \left(\theta - \mathsf{E}_\theta \widehat{\theta}_{(\bullet)} \right).$$

At the same time,

$$\mathsf{E}_\theta \widehat{\theta}_{(\bullet)} = \mathsf{E}_\theta \left(\frac{1}{n} \sum_{i=1}^{n} \widehat{\theta}_{(-i)} \right) = \frac{1}{n} \sum_{i=1}^{n} \mathsf{E}_\theta \widehat{\theta}_{(-i)} = \theta.$$

Thus, relying on the assumption of the unbiasedness of $\widehat{\theta}_n$, we obtain that $\mathsf{E}_\theta \widehat{\mathsf{bias}}_{jack} = 0$. Then, $\mathsf{E}_\theta \widehat{\theta}_{jack} = \theta$. □

What happens to a biased estimator? We can consider, for example, the estimator S^2 of the sample variance

$$S^2 = \frac{1}{n} \sum_{i=1}^{n} (X_i - \overline{X})^2.$$

It's biased, and if the variance of the population $\mathsf{D}\xi$ equals σ^2, then

$$\mathsf{E}S^2 = \frac{n-1}{n} \sigma^2,$$

and the bias (the difference between the expected value of the estimator and the true value of the parameter) equals

$$\mathsf{E}S^2 - \sigma^2 = -\frac{\sigma^2}{n},$$

and the bias decreases when the sample size n increases at the rate of n^{-1}. Now we can find the bias of the estimator constructed using the jackknife. Since

$$\mathsf{E}_\theta \widehat{\mathsf{bias}}_{jack} = (n-1) \left(\mathsf{E}_\theta \widehat{\theta}_n - \mathsf{E}_\theta \widehat{\theta}_{(\bullet)} \right) = (n-1) \left(\frac{n-1}{n} \sigma^2 - \frac{1}{n} \sum_{i=1}^{n} \frac{n-2}{n-1} \sigma^2 \right) =$$

$$= (n-1)\sigma^2\left(\frac{n-1}{n} - \frac{n-2}{n-1}\right) = \frac{\sigma^2}{n},$$

then

$$\mathsf{E}_\theta\widehat{\theta}_{jack} = \mathsf{E}_\theta\widehat{\theta}_n + \mathsf{E}_\theta\widehat{\mathsf{bias}}_{jack} = \frac{n-1}{n}\sigma^2 + \frac{\sigma^2}{n} = \sigma^2.$$

What conclusion can be drawn? Well, the jackknife estimator based on the estimator S^2 is unbiased. We can show that it matches the unbiased estimator S_0^2 that is equal to

$$\widehat{\theta}_{jack} = S_0^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X})^2.$$

Example 2.2.4 *Assume that the population of ξ has a distribution $\mathsf{U}_{\theta,\theta+1}$ with an unknown parameter θ. Then, the estimator of the parameter θ (according to maximum likelihood estimation) based on the sample $X_1, X_2, ..., X_n$ is given by the following analytical expression:*

$$\widehat{\theta} = \widehat{\theta}_n(X_1, X_2, ..., X_n) = X_{(1)},$$

where $X_{(1)}$ is the first term of the variational series. Let $\theta = 2.5$ and the set

$$(2.60, 3.26, 2.75, 2.64, 2.83, 2.58, 3.17, 3.31, 3.48, 3.14)$$

be a sample of size 10 with a uniform distribution $\mathsf{U}_{2.5,3.5}$, which elements are rounded to the nearest hundredth. According to the chosen statistics, this sample is used to estimate θ based on the smallest element of the original sample:

$$\widehat{\theta} = X_{(1)} = 2.58.$$

Let's create $n = 10$ jackknife sets. The first set $X_{[1]}$ will consist of the following elements (the first element of the original sample is excluded):

$$(3.26, 2.75, 2.64, 2.83, 2.58, 3.17, 3.31, 3.48, 3.14).$$

Let's find the partial estimator $\widehat{\theta}_{(-1)}$ that returns the smallest element of the set $X_{[1]}$. Then,

$$\widehat{\theta}_{(-1)} = 2.58.$$

Having performed the same steps for the nine remaining pseudosamples, we obtain that the jackknife estimator will be equal to:

$$\widehat{\theta}_{jack} = n\widehat{\theta}_1 - (n-1)\frac{1}{n}\sum_{i=1}^{n}\widehat{\theta}_{(-i)} \approx 2.56.$$

This jackknife estimator will be closer to the true value of $\theta = 2.5$ *rather than an original one.*

Let's perform modeling for larger samples. Fig. 2 shows that both estimators approach the true value of the parameter θ *as the sample size* n *increases. However, the original estimator* $\widehat{\theta} = X_{(1)}$ *(the blue points in the figure) is always greater than the true value that equals* 2.5 *(it corresponds to the green straight line). The jackknife estimators lie on different sides of the true value and are generally closer to the true value than the original ones.*

Thus, we have changed the bias from always positive to being less than or of a different sign. A simple example will make the theory less abstract. If the weighing scales in a shop are inaccurate in favor of customers (if the scales show smaller numbers), the shop will lose money with every purchase. If the weighing scales are sometimes inaccurate in favor of the shop and sometimes in favor of customers, nobody loses money on average.

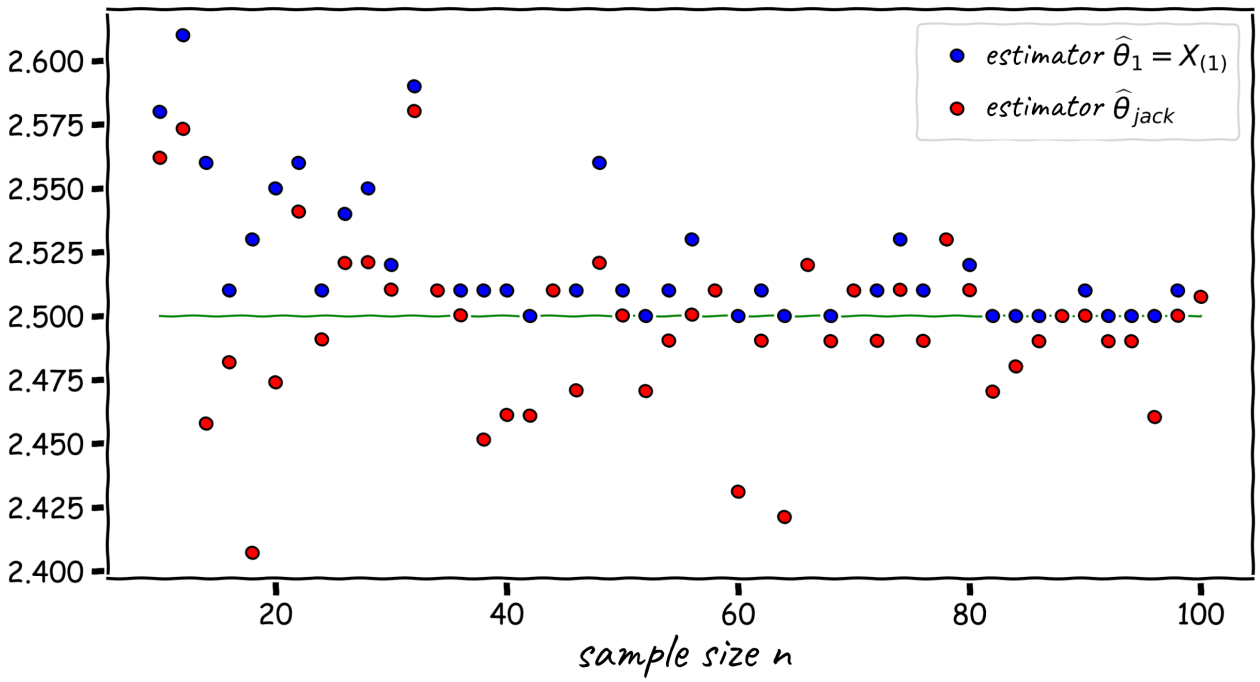


Figure 2: Relationship between the estimators $\widehat{\theta_1}$ and $\widehat{\theta}_{(jack)}$ and the sample size n

It turns out that a more general statement is also true.

Theorem 2.2.1 *Let* $\widehat{\theta} = \widehat{\theta}_n(X_1, X_2, ..., X_n)$ *be the estimator of the parameter* θ*, and*

$$\mathsf{E}_\theta \widehat{\theta} = \theta + \frac{a_1(\theta)}{n} + \frac{a_2(\theta)}{n^2} + O(n^{-3}).$$

Then,

$$\mathsf{E}_\theta \widehat{\theta}_{jack} = \theta - \frac{a_2(\theta)}{n(n-1)} + O(n^{-2}).$$

To put it differently, the theorem states that if the bias $\mathsf{E}_\theta\widehat{\theta} - \theta$ decreases when n increases at the rate of n^{-1}, the bias of the jackknife estimator will decrease at the rate of n^{-2}.

Proof.

We calculate

$$\widehat{\mathsf{E}_\theta \mathsf{bias}}_{jack} = (n-1)\left(\mathsf{E}_\theta\widehat{\theta}_n - \mathsf{E}_\theta\widehat{\theta}_{(\bullet)}\right).$$

Since

$$\mathsf{E}_\theta\widehat{\theta}_{(-i)} = \theta + \frac{a_1(\theta)}{n-1} + \frac{a_2(\theta)}{(n-1)^2} + O(n^{-3}),$$

then

$$\mathsf{E}_\theta\widehat{\theta}_{(\bullet)} = \frac{1}{n}\sum_{i=1}^{n}\left(\theta + \frac{a_1(\theta)}{n-1} + \frac{a_2(\theta)}{(n-1)^2} + O(n^{-3})\right) = \theta + \frac{a_1(\theta)}{n-1} + \frac{a_2(\theta)}{(n-1)^2} + O(n^{-3}).$$

Then,

$$\widehat{\mathsf{E}_\theta \mathsf{bias}}_{jack} = (n-1)\left(a_1(\theta)\left(\frac{1}{n} - \frac{1}{n-1}\right) + a_2(\theta)\left(\frac{1}{n^2} - \frac{1}{(n-1)^2}\right) + O(n^{-3})\right) =$$

$$= -\frac{a_1(\theta)}{n} - \frac{(2n-1)a_2(\theta)}{n^2(n-1)} + O(n^{-2}).$$

Therefore,

$$\mathsf{E}_\theta\widehat{\theta}_{jack} = \mathsf{E}_\theta\widehat{\theta} + \widehat{\mathsf{E}_\theta \mathsf{bias}}_{jack} =$$

$$= \theta + \frac{a_1(\theta)}{n} + \frac{a_2(\theta)}{n^2} + O(n^{-3}) - \frac{a_1(\theta)}{n} - \frac{(2n-1)a_2(\theta)}{n^2(n-1)} + O(n^{-2}) =$$

$$= \theta - \frac{a_2(\theta)}{n(n-1)} + O(n^{-2}).$$

□

The jackknife estimator allows correcting bias so the bias converges to zero at a faster rate.

Remark 2.2.3 *Note that the expected value condition formulated in the theorem is rare in practice. In particular,*

$$S^2 = \frac{n-1}{n}\sigma^2 = \sigma^2 - \frac{\sigma^2}{n}$$

satisfies the condition of the theorem. In this case, $a_1(\theta) = -\sigma^2$ *(since the parameter is* σ *or* σ^2*), and* $a_2(\theta) = 0$*. As we have seen before, the jackknife estimator based on* S^2 *is unbiased and matches* S_0^2*.*

2.2.3 Jackknife Estimator of Variance

Let's estimate the variance of $\widehat{\theta}_n$. The unbiased sample variance of the pseudovalues defined by the equalities

$$\widetilde{\theta}_i = n\widehat{\theta}_n - (n-1)\widehat{\theta}_{(-i)},$$

equals

$$S_0^2(\widetilde{\theta}) = \frac{1}{n-1}\sum_{i=1}^{n}(\widetilde{\theta}_i - \widehat{\theta}_{jack})^2,$$

since

$$\widehat{\theta}_{jack} = \frac{1}{n}\sum_{i=1}^{n}\widetilde{\theta}_i.$$

Definition 2.2.6 *A jackknife estimator of the variance of $\widehat{\theta}_n$ is a value*

$$\widehat{\mathsf{Var}}_{jack} = \frac{S_0^2(\widetilde{\theta})}{n} = \frac{1}{n(n-1)}\sum_{i=1}^{n}(\widetilde{\theta}_i - \widehat{\theta}_{jack})^2.$$

Remark 2.2.4 *Note that when θ is an expected value of the population and its estimator is $\widehat{\theta} = \widehat{\theta}_n(X_1, X_2, ..., X_n) = \overline{X}$, then*

$$\widetilde{\theta}_i = (X_1 + X_2 + ... + X_n) - (X_1 + ... + X_{i-1} + X_{i+1} + ... + X_n) = X_i.$$

As calculated earlier,

$$\widehat{\theta}_{jack} = \overline{X},$$

therefore,

$$\widehat{\mathsf{Var}}_{jack} = \frac{1}{n(n-1)}\sum_{i=1}^{n}(X_i - \overline{X})^2 = \frac{\sigma^2}{n} = \mathsf{D}_\theta\overline{X}.$$

It turns out that the expression for the jackknife estimator of the variance can be rewritten as follows:

$$\widehat{\mathsf{Var}}_{jack} = \frac{1}{n(n-1)}\sum_{i=1}^{n}(\widetilde{\theta}_i - \widehat{\theta}_{jack})^2 = \frac{n-1}{n}\sum_{i=1}^{n}\left(\widehat{\theta}_{(-i)} - \widehat{\theta}_{(\bullet)}\right)^2.$$

Without going into details, we would like to note that if θ_n satisfies regular conditions, for example, if θ_n is a smooth function of the sample mean $\overline{X}$, then the jackknife estimator of the variance of $\widehat{\theta}_n$ is consistent. If the original statistic is not as smooth as, for example, the sample median, the jackknife is useless because its estimators will likely be inconsistent.

2.3 Bootstrap

Let's consider another method for generating repeated pseudosamples called the bootstrap. This technique was proposed in 1979 by Bradley Efron and gained popularity since then. As the jackknife, the bootstrap allows constructing confidence intervals, estimating different characteristics of the considered statistic, such as bias and variance, and do other things. At the same time, the scope of application of this method is wider than that of the jackknife due to greater computational complexity.

2.3.1 General View of Bootstrap Estimators

Let's arrive at the bootstrap logically. As usual, let θ be an unknown parameter (or characteristic) of a distribution of the population of ξ, and $X_1, X_2, ..., X_n$ be a sample from the population of ξ, and

$$\widehat{\theta} = \widehat{\theta}_n(X_1, X_2, ..., X_n)$$

be a (consistent) estimator of θ. On a given sample (as a result of the experiment), we can obtain only a particular (one) value of our estimator $\widehat{\theta}$. Thus, we cannot estimate the mean or spread, and we cannot construct a confidence interval for θ. What should we do then?

First, let's consider an ideal case. Say, we can derive more than one sample from the population, for example, B samples. Thus, we have B sets

$$X_1^{(j)}, X_2^{(j)}, ..., X_n^{(j)}, \quad j \in \{1, 2, ..., B\}$$

of independent and identically (to ξ) distributed random variables. For each, we can calculate the value of our statistic $\widehat{\theta}$

$$\widehat{\theta}^j = \widehat{\theta}_n^j(X_1^{(j)}, X_2^{(j)}, ..., X_n^{(j)}), \quad j \in \{1, 2, ..., B\}.$$

Reasonable estimators of the expected value and variance of $\widehat{\theta}$ are

$$\widehat{\theta}_{(\bullet),B} = \frac{1}{B}\sum_{j=1}^{B}\widehat{\theta}^j$$

and

$$\widehat{\mathsf{Var}}_B = \frac{1}{B-1}\sum_{j=1}^{B}\left(\widehat{\theta}^j - \frac{1}{B}\sum_{j=1}^{B}\widehat{\theta}^j\right)^2 = \frac{1}{B-1}\sum_{j=1}^{B}\left(\widehat{\theta}^j - \widehat{\theta}_{(\bullet),B}\right)^2.$$

We can support the discussed things with theory, in particular, with the well-known Monte Carlo method. It all looks good, but here's the problem. Usually, we cannot draw as many samples from the population as we need. So, what can we do? How to put this logic into practice? The bootstrap is a solution.

2.3.2 Bootstrap Method

Here's the idea of the bootstrap. Let $X_1, X_2, ..., X_n$ be a sample. Based on it, we can model the distribution of ξ and derive the necessary samples from the modeled distribution. This approach will allow us to apply the apparatus described earlier.

According to statistics, the true distribution of ξ based on the sample $X_1, X_2, ..., X_n$ is converged by an empirical one with a distribution function

$$F_n^*(x) = \frac{1}{n}\sum_{i=1}^{n} \mathsf{I}(X_i < x),$$

where $\mathsf{I}(A)$ is an indicator of the event A, which analytical representation is as follows:

$$\mathsf{I}(A) = \begin{cases} 1, & A \text{ occurred} \\ 0, & A \text{ has not occurred} \end{cases}.$$

Based on the distribution function, we can easily model the samples of the necessary size. But what is it equivalent to? It's equivalent to creating samples by way of the random selection of n objects with a return from the original sample.

Definition 2.3.1 *Bootstrap samples of size n from the sample $X_1, X_2, ..., X_n$ are sets*

$$X_1^{(j)}, X_2^{(j)}, ..., X_n^{(j)},$$

where $X_i^{(j)} \in \{X_1, X_2, ..., X_n\}$.

The number of samples that we can obtain this way is very high. The index j can take values from $\{1, 2, ..., n^n\}$. For example, for the bootstrap sample of size 12, the number of bootstrap samples will be equal to

$$12^{12} = 8\,916\,100\,448\,256.$$

Some people don't even know how to read this number out loud. That's why the number of bootstrap samples in practice is much smaller than the real one.

Remark 2.3.1 *Since the elements are chosen and then returned to the sample, bootstrap samples usually contain duplicates from the original sample. And here's the interesting thing. When n increases, the rate of the unique elements of the original sample, which are present in the bootstrap sample, approaches ≈ 0.632.*

Assume that we are drawing the sample $S_1, S_2, ..., S_n$. For each element of the original sample $X_1, X_2, ..., X_n$, the probability that $S_i \neq X_j$ equals

$$\mathsf{P}\left(S_i \neq X_j\right) = 1 - \frac{1}{n}.$$

Since the selection is random,

$$\mathsf{P}(S_1 \neq X_j, S_2 \neq X_j, ..., S_n \neq S_j) = \left(1 - \frac{1}{n}\right)^n \xrightarrow[n\to+\infty]{} e^{-1} \approx 0.368.$$

Thus, the probability that the element X_j will be in the drawn sample equals

$$1 - e^{-1} \approx 0.632,$$

and it is true for any element of the original sample, that is, given any $j \in \{1, 2, ..., n\}$.

When bootstrap samples are generated, the algorithm of finding the estimators of the expected value and the variance of the estimator $\widehat{\theta}$ is as follows.

1. Let there be B generated bootstrap samples.
$$X_1^{(j)}, X_2^{(j)}, ..., X_n^{(j)}, \quad j \in \{1, 2, ..., B\}.$$
2. For each of them, the value of $\widehat{\theta}$ is calculated,
$$\widehat{\theta}^j = \widehat{\theta}_n^j(X_1^{(j)}, X_2^{(j)}, ..., X_n^{(j)}), \quad j \in \{1, 2, ..., B\}.$$
3. As the estimators of the expected value and the variance of $\widehat{\theta}$, we take
$$\widehat{\theta}_{(\bullet),B} = \frac{1}{B}\sum_{j=1}^{B}\widehat{\theta}^j$$
and
$$\widehat{\mathsf{Var}}_B = \frac{1}{B-1}\sum_{j=1}^{B}\left(\widehat{\theta}^j - \frac{1}{B}\sum_{j=1}^{B}\widehat{\theta}^j\right)^2 = \frac{1}{B-1}\sum_{j=1}^{B}\left(\widehat{\theta}^j - \widehat{\theta}_{(\bullet),B}\right)^2,$$
respectively.

Definition 2.3.2 *The introduced values*

$$\widehat{\theta}_{(\bullet),B} = \frac{1}{B}\sum_{j=1}^{B}\widehat{\theta}^j$$

and

$$\widehat{\mathsf{Var}}_B = \frac{1}{B-1}\sum_{j=1}^{B}\left(\widehat{\theta}^j - \frac{1}{B}\sum_{j=1}^{B}\widehat{\theta}^j\right)^2 = \frac{1}{B-1}\sum_{j=1}^{B}\left(\widehat{\theta}^j - \widehat{\theta}_{(\bullet),B}\right)^2,$$

are called the bootstrap estimators of the expected value and variance of $\widehat{\theta}$, respectively.

Example 2.3.1 *Let's get back to the example when the population has a distribution* $\mathsf{U}_{\theta,\theta+1}$ *with the parameter* $\theta = 2.5$. *We perform modeling and find the true value of the estimator variance, as well as the jackknife and bootstrap estimators of the variance of* $\widehat{\theta} = X_{(1)}$. *Note that when values of* n *are small, the results*

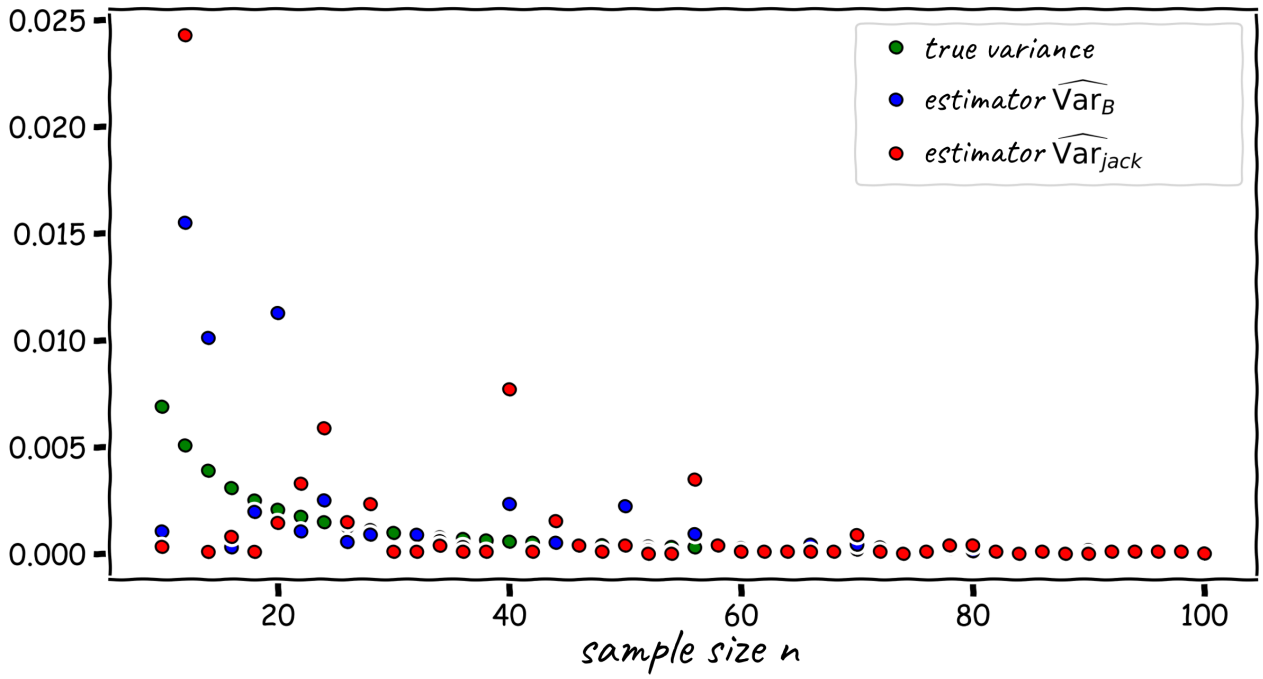


Figure 3: Relationship between the variance estimators $\widehat{\mathsf{Var}}_{jack}$ and $\widehat{\mathsf{Var}}_B$ and the sample size n

vary, but the jackknife estimator of the variance is usually smaller. On the other hand, starting from some n, *the bootstrap estimator of the variance is smaller or coincides with the jackknife estimator of the variance. In practice, the jackknife method is more suitable for smaller samples while the bootstrap is preferred for large amounts of data.*

2.3.3 Constructing Confidence Intervals

The construction of the same confidence intervals for θ is based on the ideas that we've just discussed. Let's consider the most simple case. If we assume that

$$\frac{\widehat{\theta} - \theta}{\sqrt{\mathsf{D}_\theta \widehat{\theta}}} \xrightarrow[n \to +\infty]{\mathsf{d}} \eta \sim \mathsf{N}_{0,1},$$

or close to normal, and $\widehat{\mathsf{Var}}_B$ is a consistent estimator of $\mathsf{D}_\theta \widehat{\theta}$, then the asymptotic confidence interval of confidence level $(1 - \varepsilon)$ for the parameter θ is constructed

as usual and has the form

$$\left(\widehat{\theta} - \tau_{1-\varepsilon/2}\sqrt{\widehat{\mathsf{Var}}_B},\ \widehat{\theta} + \tau_{1-\varepsilon/2}\sqrt{\widehat{\mathsf{Var}}_B}\right),$$

where $\tau_{1-\varepsilon/2}$ is the quantile at level $(1-\varepsilon/2)$ of a standard normal distribution.

2.3.4 Final Notes

Resampling is a broad theme to cover, and we cannot squeeze it into one module. The considered approaches and techniques are only a part of the advantages that the bootstrap gives to a researcher. As has been done in the jackknife, the bootstrap can estimate and improve the bias of the original estimator $\widehat{\theta}$, and many more.

Moreover, the things we discussed before related to the non-parametric bootstrap because we didn't make assumptions about the distribution of ξ. At the same time, if the relationship between the distribution of ξ and the parameter of interest is known, the method can be significantly strengthened. In such a case, bootstrap samples are generated from the empirical distribution function, where we can replace the unknown parameter θ with its estimator. In this case, the number of bootstrap samples can be unlimitedly large.

Now we are ready to apply the studied methods for ensemble construction in practice.

3 Ensemble Learning

3.1 General Definition of Ensemble Learning

As we have mentioned before, ensemble learning is a machine learning approach, in which several models (that are also called **weak learners**) are trained to solve the same problem and then combined for better results. The most important question here is how to combine weak learners.

Well, assume that we consider supervised learning. We have a (particular) training sample $x_1, x_2, ..., x_n$ with the responses $y_1, y_2, ..., y_n$, where $x_i \in X$, $y_i \in Y$, $i \in \{1, 2, ..., n\}$.

Remark 3.1.1 *Let's clarify the introduced notations. For example, if $X_1, X_2, ..., X_p$ are numeric predictors (an ordinary case), it's reasonable to consider $\mathbb{R}^p$ as the set X. Thus, we can consider a particular observation as the vector*

$$x_i = (x_{i1}, x_{i2}, ..., x_{ip}) \in \mathbb{R}^p.$$

When considering the classification problem, the set Y usually has the form $Y = \{1, 2, 3, ..., M\}$, where M is a number of classes (of course, classes can be numbered differently). When considering the regression problem, Y is usually the same as $\mathbb{R}$ or is being its 'continuous' subset.

Well, let's define the term algorithm.

Definition 3.1.1 *Let $b(x) : X \to R$ be a base algorithm trained on a training dataset, R be an auxiliary set, and $C : R \to Y$ be a so-called decision rule. The algorithm using the base algorithm b with the decision rule C will be*

$$a(x) = C(b(x)).$$

We will use examples to explain what a base algorithm and the set R are.

Example 3.1.1 *For example, assume that $Y = \{-1, 1\}$ and that we are solving a binary classification problem. Let the base algorithm $b(x)$ be an algorithm of logistic regression. Hence, $b(x)$ outputs a value between 0 and 1, that is, the probability of assigning the object to the class $+1$, and the set R is $[0, 1]$. The decision rule for the cutoff value of 0.5 will be as follows. If $b(x) \geq 0.5$, the observation will fall into the class $+1$ or -1 otherwise. The logistic regression algorithm with the selected decision rule can be described as follows:*

$$a(x) = \begin{cases} 1, & b(x) \geq 0.5 \\ -1, & b(x) < 0.5 \end{cases}.$$

The decision rule can be chosen differently. It's up to a researcher.

Remark 3.1.2 *For example, we can differently handle the cutoff value of 0.5 from the previous example. The most common approach is to add an additional class 0 for objects that are difficult to classify. In this case, the algorithm sticks to the principle of keeping silence rather than lying, which can be written as follows:*

$$a(x) = \begin{cases} 1, & b(x) > 0.5 \\ 0, & b(x) = 0.5 \\ -1, & b(x) < 0.5 \end{cases}.$$

Well, the choice of the cutoff value being equal to 0.5 is rather subjective.

Let's consider another example.

Example 3.1.2 *Assume we are solving a binary classification problem, $Y = \{-1, 1\}$, and the base algorithm $b(x)$ is an SVM algorithm. In this case, $R = \mathbb{R}$,*

since $b(x)$ outputs random numbers, and the decision rule with extra class 0 can be given as follows: if $b(x) > 0$, then the observation x is assigned the class $+1$; if $b(x) < 0$, then -1; and if $b(x) = 0$, then 0. It can be analytically represented as follows:

$$a(x) = \operatorname{sign} b(x) = \begin{cases} 1, & b(x) > 0 \\ 0, & b(x) = 0 \\ -1, & b(x) < 0 \end{cases}.$$

Example 3.1.3 *For example, assume we are solving an M-class classification problem, $Y = \{1, 2, ..., M\}$, and $b_i(x)$ is a base algorithm of logistic regression that returns the probability of assigning an object to the class under the number $i \in \{1, 2, ..., M\}$ ($b_i(x) = p \in [0, 1]$ means that the object x belongs to the class i with the probability p and does not belong to the class i with the probability $(1 - p)$). The auxiliary set R is, once again, a closed interval $[0, 1]$. Thus, the decision rule can be, for example, as follows. An object is assigned to the class, for which the base algorithm will output the greatest number. To put it differently,*

$$a(x) = \underset{i \in \{1,2,...,M\}}{\arg\max}\ b_i(x).$$

If there are several potential classes, the observation can be assigned to any class or the extra class 0 (if applicable).

Example 3.1.4 *Assume that a regression problem is being solved, and $b(x)$ is a base algorithm of multivariate linear regression. Well, a decision rule is not necessary (since it is an identity function), and*

$$a(x) = C(b(x)) = b(x),$$

therefore, $R = \mathbb{R}$.

Now that we know the definition of the algorithm, we can define an ensemble.

Definition 3.1.2 *Let $b_i(x) : X \to R$ be base algorithms, $i \in \{1, 2, ..., k\}$. Moreover, let $F : R^k \to R$ be an adjustment rule and $C : R \to Y$ be a decision rule. Thus, the ensemble of algorithms $b_1, ..., b_k$ with the adjustment rule F and decision rule C is the algorithm*

$$a(x) = C(F(b_1(x), b_2(x), ..., b_k(x))).$$

Let's clarify the introduced definition using an example.

Example 3.1.5 *Assume that a regression problem is being solved, and $b_i(x)$ is a base algorithm, $i \in \{1, 2, ..., k\}$ (for example, the algorithms of polynomial regression with polynomials of different degrees). Thus, as an adjustment rule, we can consider, for example, simple voting:*

$$F(b_1(x), b_2(x), ..., b_k(x)) = \frac{1}{k} \sum_{i=1}^{k} b_i(x)$$

is the usual averaging of outputs. In this case, no decision rule is needed (since it's an identity function), and the prediction can be obtained using the following algorithm:

$$a(x) = \frac{1}{k} \sum_{i=1}^{k} b_i(x).$$

In this example, all the algorithms have the same weight, and neither stands out. Here's one more example that generalizes the previous one.

Example 3.1.6 *Assume that a regression problem is being solved, and $b_i(x)$ is a base algorithm, $i \in \{1, 2, ..., k\}$. Thus, as an adjustment rule, we can consider, for example, weighted voting: Let ω_i, $i \in \{1, 2, ..., n\}$ be weights, that is, $\omega_i \geq 0$ and*

$$\sum_{i=1}^{k} \omega_i = 1.$$

Thus, we can set an adjustment rule as follows:

$$F(b_1(x), b_2(x), ..., b_k(x)) = \sum_{i=1}^{k} \omega_i b_i(x).$$

A decision rule is also an identity function, and the prediction can be made by the following algorithm:

$$a(x) = \sum_{i=1}^{k} \omega_i b_i(x).$$

In the considered example, the more reliable (or the most preferred) algorithm should have a larger weight.

Example 3.1.7 *Assume we are solving an M-class classification problem, $Y = \{1, 2, ..., M\}$ and $a_i(x)$, $i \in \{1, 2, ..., k\}$ are algorithms based on some base algorithms, that is, those that predict a class. As an adjustment rule, we can consider the so-called majority voting.*

$$F(a_1(x), a_2(x), ..., a_k(x)) = \mathsf{mode}(a_1(x), a_2(x), ..., a_k(x)),$$

where mode *returns the mode of the dataset, which is the most frequently occurring value in the dataset. If there are several such values, we can take the value of F as a mode or assign the extra class* 0*. Then, the ensemble (with an identity-function decision rule) can be defined as*

$$a(x) = \mathsf{mode}(a_1(x), a_2(x), ..., a_k(x)).$$

As before, we can use weighted voting. Without further details, we would like to note that an ensemble can be defined as

$$a(x) = \underset{m \in \{1,2,...,M\}}{\arg\max} \sum_{i=1}^{k} w_i \mathsf{I}\left(a_i(x) = m\right),$$

and when there are many possible classes, we can use any of the approaches described earlier. The described rule chooses the class corresponding to the largest sum, and a decision of each classifier is also assigned some weight that affects the entire sum.

The given adjustment functions are not all that are possible, but they are the most frequent. The choice of an adjustment function is a challenge, and it often depends on the domain.

Base algorithms and algorithms using them can be obtained by different machine learning algorithms or various training samples. Let's move on to concrete algorithms, starting with bagging. It uses the discussed statistical apparatus of the bootstrap.

3.2 Bagging

Bagging (**B**ootstrap **agg**regat**ing**) which is one of the ensemble learning approaches is based on independent learning of base algorithms on different samples. Let's formally describe the algorithm using the introduced notations.

1. Let $X_1, X_2, ..., X_n$ be a sample of size n, and $b_1(x), b_2(x), ..., b_t(x)$ be base algorithms.

2. Using $X_1, X_2, ..., X_n$, create t independent subsamples

$$X_1^{(j)}, X_2^{(j)}, ..., X_l^{(j)}, \quad j \in \{1, 2, ..., t\}$$

   of size $l \leq n$ using the bootstrap.

3. For each $j \in \{1, 2, ..., t\}$, train the base algorithm $b_j(x)$ using the sample $X_1^{(j)}, X_2^{(j)}, ..., X_l^{(j)}$.

4. Form an ensemble

$$a(x) = C\left(F(b_1(x), b_2(x), \ldots, b_t(x))\right).$$

Remark 3.2.1 *Bagging is often used to reduce the variance of a particular base algorithm $b(x)$. To do so, it is assumed that $b_1(x) = b_2(x) = ... = b_t(x) = b(x)$. One algorithm is trained on different subsamples drawn from the original sample. All steps of the algorithm are kept.*

Remark 3.2.2 *Let's note that the choice of the size of subsample l is another task. The obtained sample should be representative for learning and prevent overfitting. Moreover, the larger the sample size, the longer it takes to train the model. Besides, if t is large, the computing power will be another issue.*

3.2.1 Example of Classification Problem

This time, we are going to apply bagging to a classification problem. We will consider the football statistics limited to two predictors, namely X_1 is shots on target; X_2 is possession. This limitation will allow us to visualize the results in the plane in a clear way. The size of the original sample is relatively small. It contains 34 observations only. The data looks as follows (the response in the first column equals 1 if the team has won or 0 if it has lost):

Win or loss	Shots on target	Possession
1	7	40
0	0	60
0	3	43
1	4	57
...	...	...

Our goal is to classify a team based on the data on the number of shots on target and possession percent. One class is for winners and another for losers. We will consider a logistic regression algorithm as a base one. The result of the logistic regression application to the original data is shown in fig. 4. The plain is split into two classes by the blue straight line. The yellow objects are wins, and green objects are losses.

Let's use bagging to form $t = 5$ samples (for clear visualization) of size 34 and choose majority voting as an ensemble. Thus,

$$a(x) = \mathsf{mode}(b_1(x), b_2(x), b_3(x), b_4(x), b_5(x)),$$

where

$$b_i(x) = \begin{cases} 1, & a_i(x) \geq 0.5 \\ 0, & a_i(x) < 0.5 \end{cases},$$

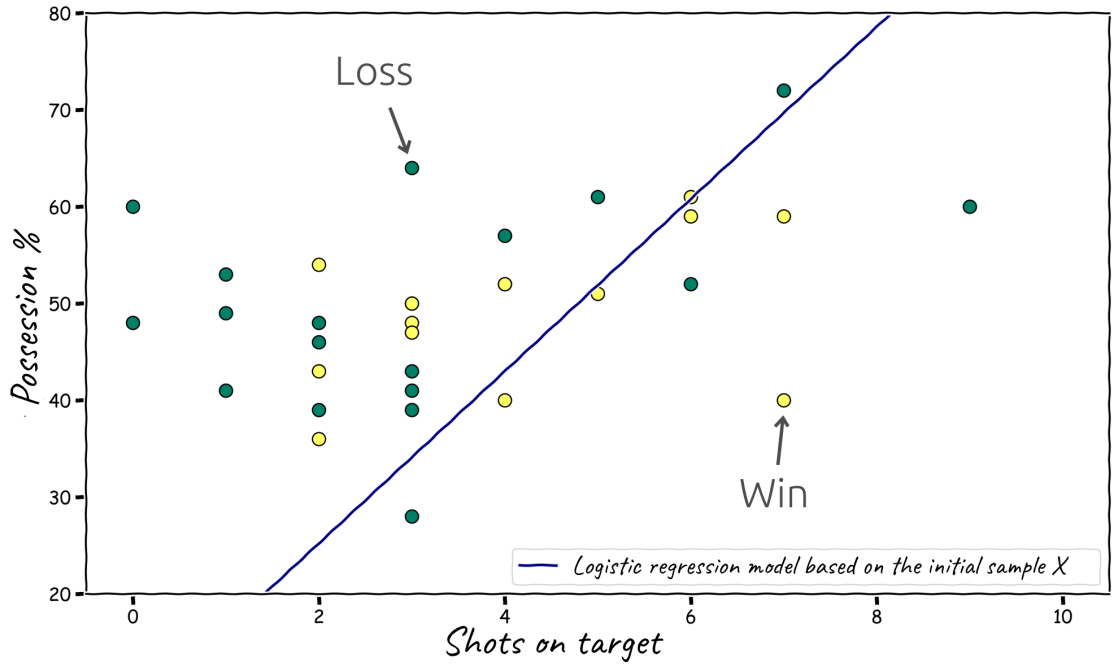


Figure 4: Logistic regression classification.

and $a_i(x)$ is a base algorithm of logistic regression that gives the probability value in the closed interval $[0, 1]$.

The easy way to compare the results is to plot the separating straight lines (that are obtained using logistic regression) in the plane. Fig. 5 shows that the results differ significantly. The normals of all the hyperplanes are pointed towards yellows (to the bottom right).

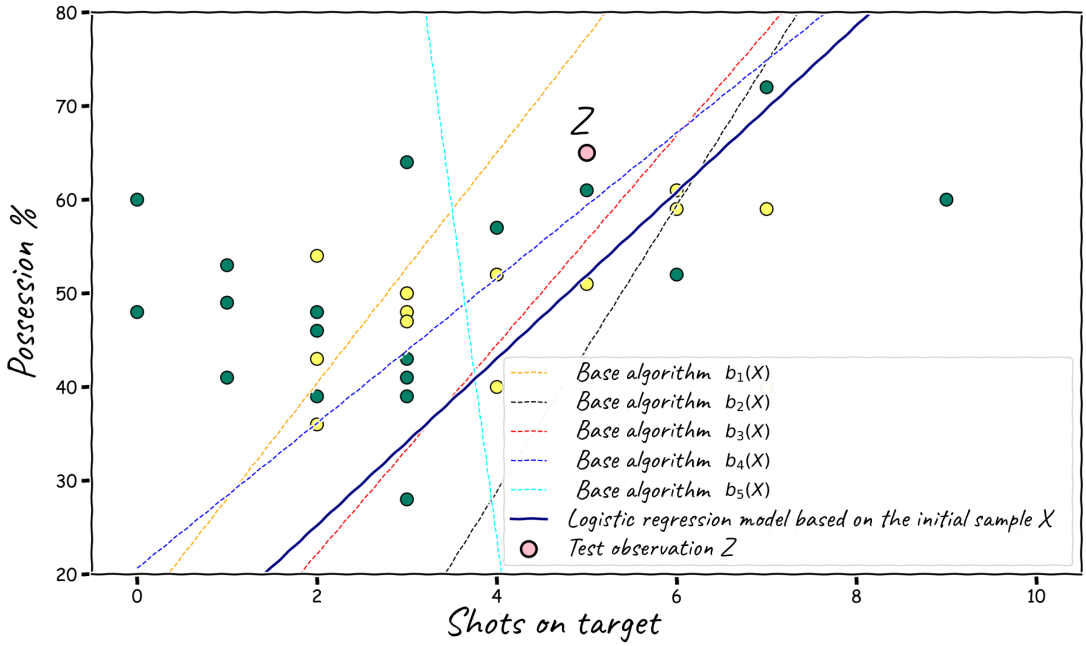


Figure 5: The base algorithms that use bagging and test observation Z.

Let's consider a new test observation $Z = (5, 65)$ (fig. 5) that corresponds to the following case. The team had 5 shots on target and 65% of possession in the match.

We can compare the classification result using logistic regression and the constructed ensemble. The logistic regression algorithm gives the probability of

$$\mathsf{P}_+ \approx 0.39$$

that a new object will be assigned to the class of yellows. Thus, given the cutoff value of 0.5, it will be assigned to greens (losers). It follows from the geometrical considerations because the object of interest lies on that side of the plane that belongs to the class of greens.

Similar calculations are performed for each base algorithm $b_i(Z)$, and predictions are equal to

$$\mathsf{P}_{b_1+} \approx 0.63, \mathsf{P}_{b_2+} \approx 0.39, \mathsf{P}_{b_3+} \approx 0.45, \mathsf{P}_{b_4+} \approx 0.38, \mathsf{P}_{b_5+} \approx 0.75.$$

Two algorithms assign the object to yellows (1st and 5th), and the remaining three assign it to greens. It's clear without calculations. The figure shows where the test object is located relative to the separating straight lines. Since the ensemble uses majority voting, the object is assigned to the class of greens.

3.2.2 Example of Regression Problem

Now let's solve a regression problem using ensemble learning. A training set is the previously used data about shopping time and the number of purchases. The data is given in the table. The number of purchases made by a customer will be the predictor X_1, and the time spent in the store will be the response Y.

No.	Shopping time (min)	Number of purchases
1	10	15
2	5	12
3	12	18
4	25	30
5	1	3
6	18	20
7	11	14
8	7	10
9	19	20
10	15	13

The regression equation of Y on X_1 (with rounded coefficients) has the following form:

$$Y = 4.06 + 0.93X_1.$$

The respective straight line is constructed in the plane (fig. 6). The prediction of the shopping time necessary to buy 27 items is calculated by plugging the value $X_1 = 27$ in the obtained function:

$$4.06 + 0.93 \cdot 27 = 29.17.$$

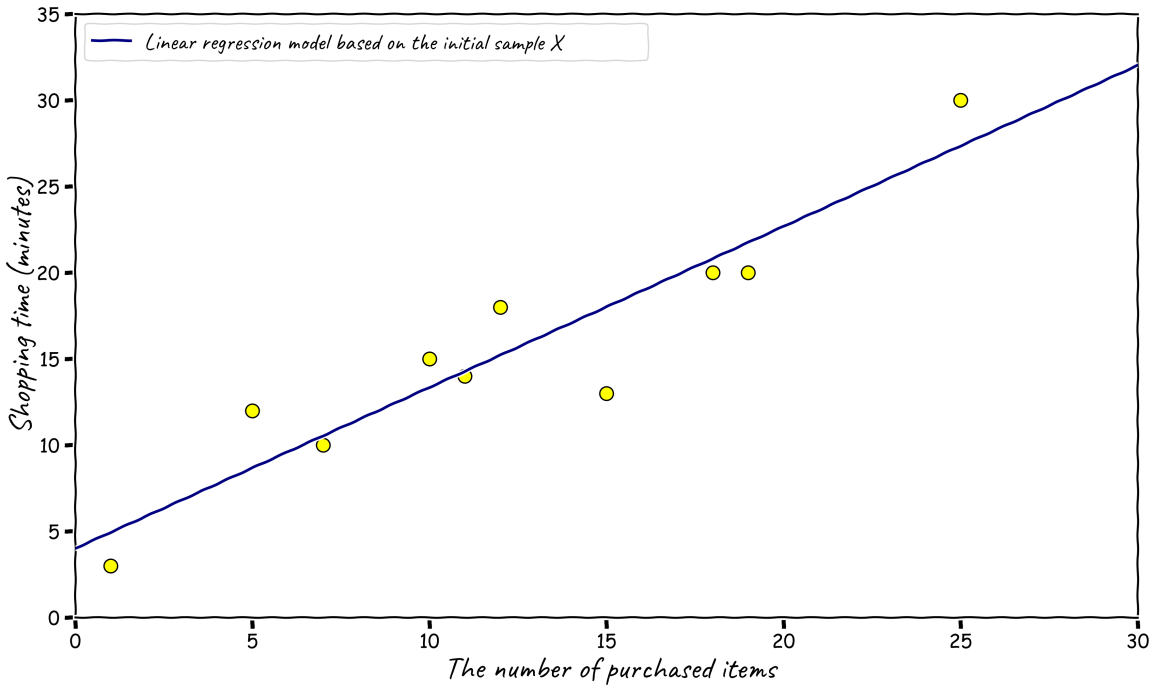


Figure 6: The relationship between shopping time and the number of purchased items.

As in the previous example, we are going to use bagging. Linear regression will be base algorithms, and simple voting (averaged results) will be the adjustment rule. We model $B = 1000$ bootstrap samples. For five of them, fig. 7 shows the straight lines that correspond to the algorithms $b_i(x)$, and each algorithm can be used for predicting.

Let's illustrate by a vertical dashed line in fig. **??** the value of $X_1 = 27$, for which we are going to make a prediction. Well, the base algorithm $b_2(x)$ predicts about 24 minutes as shown in the figure. The figure also shows the spread of the results. They vary from 24 to 31 minutes.

What about the mean of a thousand results. It will be about 28.66 minutes:

$$\frac{1}{1000}\left(31.21 + 23.94 + 27.48 + 30.84 + 25.49 + \ldots\right) = 28.66,$$

which is a half a minute more than the prediction made by the original model.

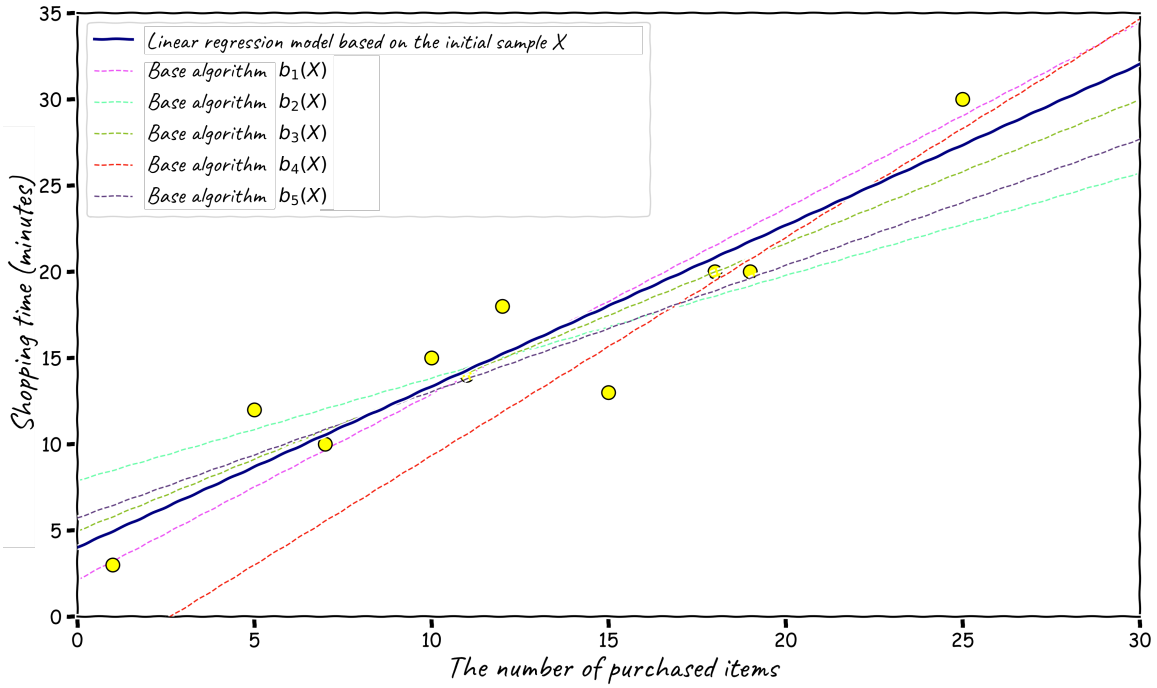


Figure 7: Base algorithms that use the bootstrap.

3.3 AdaBoost for Binary Classification

Let's consider one more ensemble technique called boosting. Its main difference from bagging is that weak learners (base algorithms) are learning successively based on the experience of the previous learner rather than learning independently. In this way, each subsequent learner reinforces the importance of the observations from the training dataset, which have been incorrectly classified by the predecessors.

Let's consider the AdaBoost algorithm to solve a binary classification problem. Let $Y = \{-1, 1\}$, $x_1, x_2, ..., x_n$, be a sample (training data) of size n, $x_i \in X$, $i \in \{1, 2, ..., n\}$, and B be a set of classification algorithms:

$$B = \{b \in B : \ b : X \to \{-1, 1\}\}.$$

To form the ensemble, we will consider the adjustment rule of weighted voting. The ensemble can be given by the following analytical expression:

$$a(x) = \text{sign}\left(\sum_{t=1}^{T} \omega_t b_t(x)\right),$$

where $b_t(x) \in B$, ω_t are weights (normalized or not), $t \in \{1, 2, ..., T\}$, T is the number of trained algorithms. The choice of classification algorithms and weights before them will be iterative to minimize the number of mistakes on the training

data. The number of mistakes can be given by the following analytical expression:

$$Q_T = \sum_{i=1}^{n} \mathsf{I}(a(x_i) \neq y_i) = \sum_{i=1}^{n} \mathsf{I}\left(y_i \sum_{t=1}^{T} \omega_t b_t(x_i) < 0\right),$$

where $x_i \in X$, I is an indicator. Each term in the first expression equals one if the ensemble incorrectly classifies the training object. Our task is to minimize the function Q_T on the training data. The reason for that is simple: the fewer mistakes, the better.

Minimization of the written expression is a difficult task because the function is not smooth. However, if we find a good function $\widetilde{Q}_T$ for which $Q_T \leq \widetilde{Q}_T$, then the minimization of the latter will decrease the original function. Since

$$Q_T = \sum_{i=1}^{n} \mathsf{I}\left(y_i \sum_{t=1}^{T} \omega_t b_t(x_i) < 0\right) \leq \sum_{i=1}^{n} \exp\left(-y_i \sum_{t=1}^{T} \omega_t b_t(x_i)\right) = \widetilde{Q}_T,$$

the written function $\widetilde{Q}_T$ can be used. It is used in the AdaBoost algorithm. Let's rewrite it as follows:

$$\widetilde{Q}_T = \sum_{i=1}^{n} \exp\left(-y_i \sum_{t=1}^{T-1} \omega_t b_t(x_i)\right) e^{-y_i \omega_T b_T(x_i)}$$

and denote

$$\alpha_i = \exp\left(-y_i \sum_{t=1}^{T-1} \omega_t b_t(x_i)\right).$$

We normalize the coefficients so that their sum equals one and obtain:

$$\alpha_0 = \sum_{i=1}^{n} \alpha_i, \quad \alpha_i = \frac{\alpha_i}{\alpha_0}, \quad i \in \{1, 2, ..., n\}.$$

Remark 3.3.1 *The last equality should be understood as follows: α_i/α_0 is calculated first and then a new value is assigned to the variable α_i. It is often denoted as*

$$\alpha_i \leftarrow \frac{\alpha_i}{\alpha_0}, \text{ or } \alpha_i := \frac{\alpha_i}{\alpha_0}.$$

Remark 3.3.2 *Note that the expression for $\widetilde{Q}_T$ is rewritten as follows:*

$$\widetilde{Q}_T = \sum_{i=1}^{n} \alpha_i e^{-y_i \omega_T b_T(x_i)}$$

and the coefficients of α_i do not depend on the algorithm b_T. They depend only on the algorithms b_i under the numbers $i < T$. Therefore, they can be found iteratively.

We need to explain how to obtain the coefficients of ω_i, $i \in \{1, 2, ..., T\}$, but we will do it in the algorithm.

Let's write the step-by-step algorithm for the training dataset $x_1, x_2, ..., x_n$ of size n, the given set of algorithms B, and the number of used algorithms T.

1. Initialize original weights as follows:
$$\alpha_1 = \alpha_2 = ... = \alpha_n = \frac{1}{n}.$$
2. For all $t \in \{1, 2, ..., T\}$
   (a) Find such an algorithm b_t from the set B that minimizes the number of mistakes on the training dataset with weights $\alpha_1, \alpha_2, ..., \alpha_n$:
   $$b_t = \arg\min_{b \in B} \sum_{i=1}^{n} \alpha_i \cdot \mathsf{I}(b(x_i) \neq y_i).$$
   If there are several algorithms, choose any.
   (b) Let
   $$\varepsilon_t = \sum_{i=1}^{n} \alpha_i \cdot \mathsf{I}(b_t(x_i) \neq y_i)$$
   be the found minimum value for an algorithm b_t. Assume that
   $$\omega_t = \frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t}.$$
   (c) Recalculate $\alpha_i = \alpha_i e^{-y_i \omega_t b_t(x_i)}$, $i \in \{1, 2, ..., n\}$.
   (d) Normalize the obtained coefficients:
   $$\alpha_0 = \sum_{i=1}^{n} \alpha_i, \quad \alpha_i = \frac{\alpha_i}{\alpha_0}, \quad i \in \{1, 2, ..., n\}.$$
3. Form a final ensemble as follows:
$$a(x) = \text{sign}\left(\sum_{t=1}^{T} \omega_t b_t(x)\right).$$

It can be proved that, given some conditions with respect to the set of algorithms B, the minimum $\widetilde{Q}_T$ is attained at the specified parameters ω_i, $i \in \{1, 2, ..., T\}$

3.3.1 Example

Let's consider another artificial example to practice applying the algorithm. We will omit step 2a of choosing an optimal algorithm b_t and select it ourselves for ease of demonstration.

Let there be a sample of size $n = 12$. The yellow points correspond to the class $+1$. The green points correspond to the class -1. Assume that base algorithms of classification can divide the plane by horizontal or vertical straight lines. None of such straight lines can divide the plane and make no mistakes. Assume that $T = 3$. Thus, our task is to create three classification algorithms

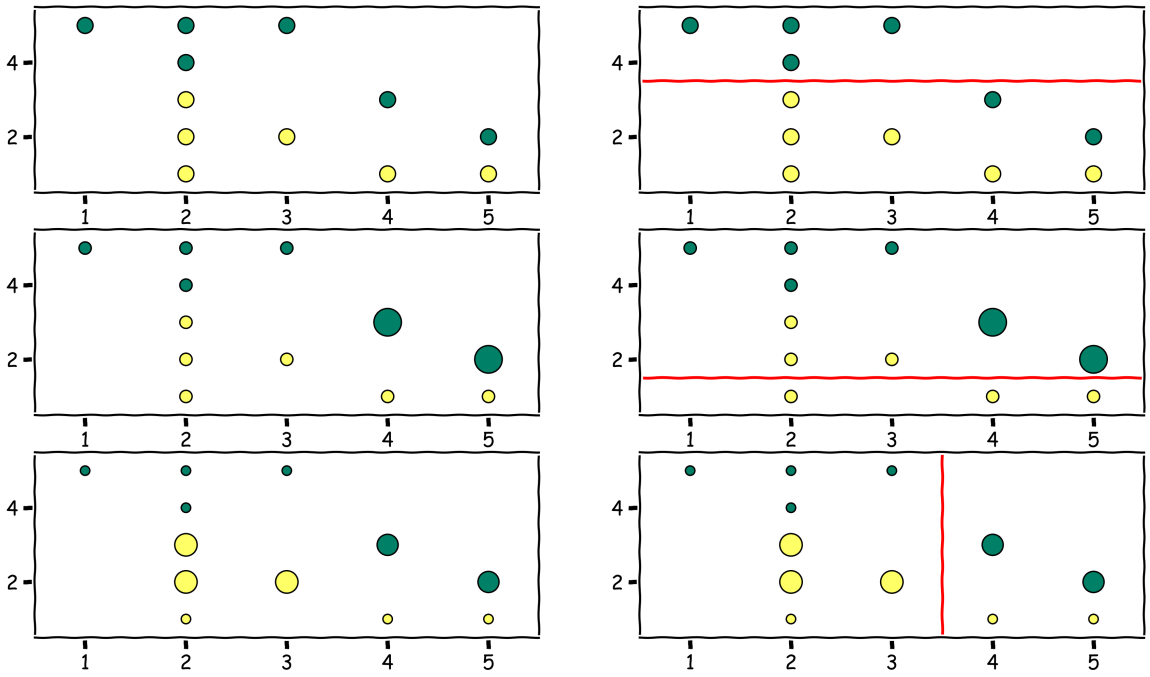


Figure 8: Three iterations of AdaBoost.

and assign them proper weighs. Let's consider each iteration in detail.

1. First, we initialize the values $\alpha_1, \alpha_2, ..., \alpha_{12}$. They are equal to

$$\alpha_1 = \frac{1}{12}, \ \alpha_2 = \frac{1}{12}, \ldots, \ \alpha_{12} = \frac{1}{12}.$$

Let the base algorithm $b_1(x)$ be represented by the horizontal line (it assigns the upper half-plane to greens and the lower to yellows). The two green objects are correctly classified. They are highlighted with blue in the table. Let's find ε_1 from the expression

X	$(4,1)$	$(2,4)$	$(5,1)$	$(3,2)$	$(1,5)$	$(3,5)$	$(2,1)$	$(2,2)$	$(2,5)$	$(4,3)$	$(2,3)$	$(5,2)$
y_i	$+1$	-1	$+1$	$+1$	-1	-1	$+1$	$+1$	-1	-1	$+1$	-1
$b_1(x_i)$	$+1$	-1	$+1$	$+1$	-1	-1	$+1$	$+1$	-1	$+1$	$+1$	$+1$

$$\varepsilon_1 = \frac{1}{12}\sum_{i=1}^{12} \mathsf{I}(b_1(x_i) \neq y_i) = \frac{1}{12} \cdot 2 \approx 0.167.$$

Now we can find the value of ω_1:

$$\omega_1 = \frac{1}{2}\ln\left(\frac{1-\varepsilon_1}{\varepsilon_1}\right) \approx \frac{1}{2}\ln\left(\frac{1-0.167}{0.167}\right) \approx 0.805.$$

We can find new weights for the objects by way of recalculation and using the relation

$$\alpha_i = \alpha_i e^{-y_i\omega_1 b_1(x_i)}, i \in \{1, 2, ..., n\}.$$

For example, α_1 is found as

$$\alpha_1 \approx \frac{1}{12} \cdot e^{-1 \cdot 0.805} \approx 0.037.$$

The remaining values are obtained similarly and given in the table. Let's normalize the obtained weights.

Based on the weight values, we find their sum $\alpha_0 \approx 0.745$ and divide each weight by α_0. The obtained values specified in the last row of the table are new weights for the next iteration (obviously they are rounded).

X	$(4,1)$	$(2,4)$	$(5,1)$	$(3,2)$	$(1,5)$	$(3,5)$	$(2,1)$	$(2,2)$	$(2,5)$	$(4,3)$	$(2,3)$	$(5,2)$
y_i	$+1$	-1	$+1$	$+1$	-1	-1	$+1$	$+1$	-1	-1	$+1$	-1
$b_1(x_i)$	$+1$	-1	$+1$	$+1$	-1	-1	$+1$	$+1$	-1	$+1$	$+1$	$+1$
$\alpha_i \cdot e^{-y_i\omega_t b_1(x_i)}$	0.037	0.037	0.037	0.037	0.037	0.037	0.037	0.037	0.037	0.186	0.037	0.186
$\frac{\alpha_i \cdot e^{-y_i\omega_t b_1(x_i)}}{\alpha_0}$	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.25	0.05	0.25

All weights are smaller than the original ones ($0.05 < 0.083$), except for the two that are greater. As you might guess, these are incorrectly classified objects. In the next iteration, the next base algorithm will pay special attention to them.

2. Now we can move on to the second step. The weights are as follows:

$$\alpha_i \approx \begin{cases} 0.25, & i \in \{10, 12\} \\ 0.05, & i \in \{1, 2, ..., 12\} \setminus \{10, 12\} \end{cases}.$$

To make the difference clearer, let's change the size of the points in figure 8 with respect to the updated weights. The selected base algorithm $b_2(x)$ divides the plane as shown in the second figure (it also assigns the upper part to greens and the lower to yellows). We classify each object using the selected algorithm. The data is given in the table.

X	$(4,1)$	$(2,4)$	$(5,1)$	$(3,2)$	$(1,5)$	$(3,5)$	$(2,1)$	$(2,2)$	$(2,5)$	$(4,3)$	$(2,3)$	$(5,2)$
y_i	$+1$	-1	$+1$	$+1$	-1	-1	$+1$	$+1$	-1	-1	$+1$	-1
$b_2(x_i)$	$+1$	-1	$+1$	-1	-1	-1	$+1$	-1	-1	-1	-1	-1
α_i	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.25	0.05	0.25

The weight of the wrong elements equals 0.05. Thus,

$$\varepsilon_2 \approx 3 \cdot 0.05 = 0.15,$$

and then

$$\omega_2 \approx \frac{1}{2} \ln \left(\frac{1 - 0.15}{0.15} \right) \approx 0.867.$$

We perform the same calculations as before, normalize the weights ($\alpha_0 \approx 0.714$), and fill out the table using the data.

X	$(4,1)$	$(2,4)$	$(5,1)$	$(3,2)$	$(1,5)$	$(3,5)$	$(2,1)$	$(2,2)$	$(2,5)$	$(4,3)$	$(2,3)$	$(5,2)$
y_i	$+1$	-1	$+1$	$+1$	-1	-1	$+1$	$+1$	-1	-1	$+1$	-1
$b_2(X)$	$+1$	-1	$+1$	-1	-1	-1	$+1$	-1	-1	-1	-1	-1
$\alpha_i \cdot e^{-y_i \omega_t b_2(x_i)}$	0.021	0.021	0.021	0.119	0.021	0.021	0.021	0.119	0.021	0.105	0.119	0.105
$\frac{\alpha_i \cdot e^{-y_i \omega_t b_2(x_i)}}{\alpha_0}$	0.029	0.029	0.029	0.167	0.029	0.029	0.029	0.167	0.029	0.147	0.167	0.147

3. And now the last step. We have three categories of weights. 0.029 for the points that were correctly classified by all the considered base algorithms (the weight for them continues to decrease). 0.167 for the points that were incorrectly classified (the weights continue to grow). 0.147 for the points, which weight was increased in the last iteration and which were correctly classified this time (due to that, their weights have slightly decreased).

   The weights:

$$\alpha_i \approx \begin{cases} 0.029, & i \in \{1,2,3,5,6,7,9\} \\ 0.147, & i \in \{10,12\} \\ 0.167, & i \in \{4,8,11\} \end{cases}.$$

   Let the selected algorithm $b_3(x)$ divide the plane by a vertical straight line as shown in the lower figure (the classification is as follows: yellows on the left and greens on the right). The classifier makes mistakes in six cases (the respective columns are highlighted in the table and have the same weight).

X	$(4,1)$	$(2,4)$	$(5,1)$	$(3,2)$	$(1,5)$	$(3,5)$	$(2,1)$	$(2,2)$	$(2,5)$	$(4,3)$	$(2,3)$	$(5,2)$
y_i	$+1$	-1	$+1$	$+1$	-1	-1	$+1$	$+1$	-1	-1	$+1$	-1
$b_3(x_i)$	-1	$+1$	-1	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$	-1	$+1$	-1
ω_3	0.029	0.029	0.029	0.167	0.029	0.029	0.029	0.167	0.029	0.147	0.167	0.147

   Thus, the error rate will be

$$\varepsilon_3 \approx 6 \cdot 0.029 \approx 0.174,$$

   and the value $\omega_3 \approx 0.779$. Let's transfer the results to the table (recalculation and weight normalization):

X	$(4,1)$	$(2,4)$	$(5,1)$	$(3,2)$	$(1,5)$	$(3,5)$	$(2,1)$	$(2,2)$	$(2,5)$	$(4,3)$	$(2,3)$	$(5,2)$
y_i	$+1$	-1	$+1$	$+1$	-1	-1	$+1$	$+1$	-1	-1	$+1$	-1
$b_3(X)$	-1	$+1$	-1	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$	-1	-1	$+1$
$\omega_3(i) \cdot e^{-\alpha_3 y_i b_3(X)}$	0.064	0.064	0.064	0.076	0.064	0.064	0.013	0.076	0.064	0.067	0.076	0.067
$\frac{\omega_3(i) \cdot e^{-\alpha_3 y_i b_3(X)}}{Z_3}$	0.084	0.084	0.084	0.1	0.084	0.084	0.018	0.1	0.084	0.089	0.1	0.089

The final algorithm is formed as follows:

$$a(x) = \text{sign}\left(\sum_{t=1}^{3} \omega_t b_t(x)\right) \approx \text{sign}\left(0.805 \cdot b_1(x) + 0.867 \cdot b_2(x) + 0.779 \cdot b_3(x)\right).$$

Thus, the classification of a test object, for example, $x = (4, 1)$, using the ensemble can be obtained as:

$$a(x) = \text{sign}\left(0.805 \cdot (+1) + 0.867 \cdot (+1) + 0.779 \cdot (-1)\right) = \text{sign}\left(0.893\right) = +1.$$

The ensemble assigns it to greens.

Moreover, we can visualize the ensemble result by coloring the regions obtained using the base algorithms (fig. 9). The region color corresponds to classes to which the objects will be assigned. As you can see, the objects are correctly classified.

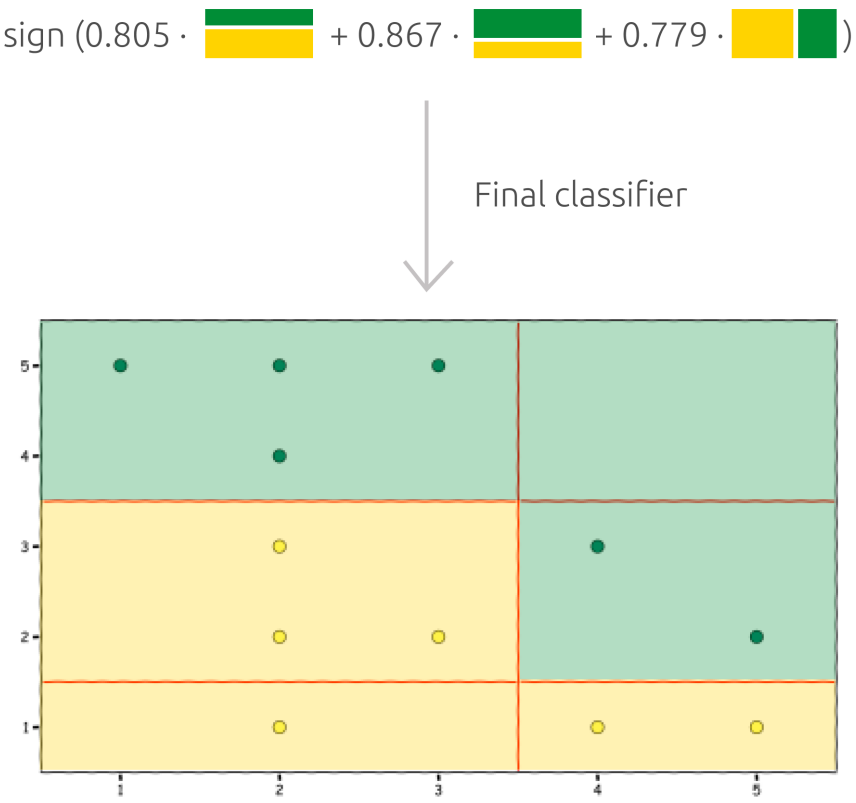


Figure 9: The final classifier obtained by AdaBoost.

3.4 Stacking

The traditional approaches of ensemble learning that we considered before used several base models based on different combinations of input data. Then, the results of this model were summarized by voting, weighted voting, averaging of the results, and so on. So, here's another question. Why does the composition require such simple operations as averaging and voting? Perhaps, we can give another algorithm (a so-called meta-algorithm) the right to choose the decision rule.

It's the idea behind stacking. This technique trains different base models and combines them by training a so-called meta-model. A meta-model is an additional model that outputs predictions based on the predictions of the base models. Thus, instead of preplanned solution such as voting, we give the meta-model the right to choose the adjustment rule.

There are several stacking algorithms, but we will review only one of them. Note that the notations used to describe this algorithm differ from the notations we used earlier.

1. Let $X = \{x_1, x_2, ..., x_n\}$ be a training sample with the responses $Y = \{y_1, y_2, ..., y_n\}$, and $b_1, b_2, ..., b_k$ be base algorithms.

2. X is segmented into k almost equal disjoint folds

$$X = X_1 \cup X_2 \cup \ldots \cup X_k.$$

   Usually, k is a small number (less than 10).

3. For each $i \in \{1, 2, ..., k\}$ and $j \in \{1, 2, ..., k\}$, the base algorithm b_i is trained on the dataset

$$X_{[-j]} = X \setminus X_j,$$

   and tested on X_j when the response set Y_{ji} is obtained.

4. To train the meta-model, meta-data (meta-predictors) is formed as follows:

$$X_{meta} = \begin{pmatrix} Y_{11} & Y_{12} & \ldots & Y_{1k} \\ Y_{21} & Y_{22} & \ldots & Y_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ Y_{k1} & Y_{k2} & \ldots & Y_{kk} \end{pmatrix}.$$

   For the model training, the responses will be the original responses of Y.

5. The trained meta-model is an ensemble based on stacking.

3.4.1 Simple Example

To demonstrate the steps of the algorithm, we will use the following data. Imagine that two persons are playing darts. The target, a darts board, consists of a unit circle with its center at the origin $(0, 0)$. Let X_1 be plotted on the horizontal axis and X_2 on the vertical axis. Then, the pair (X_1, X_2) are the coordinates of a dart that hits the target. Let the response 0 correspond to the first player (green points in fig. 10), and the response 1 to the second (yellow points).

Let's create a classification model to predict who is throwing a dart based on the coordinates of the dart.

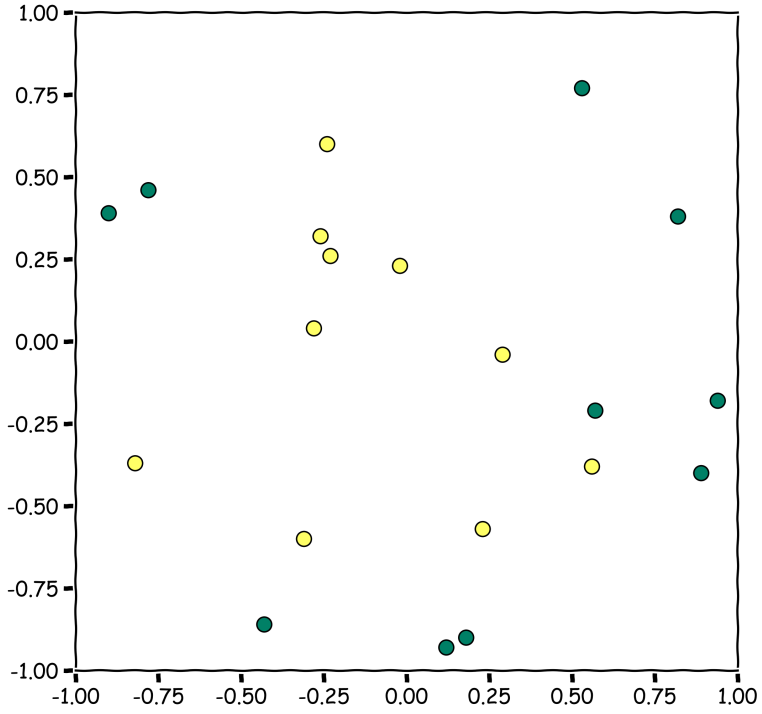
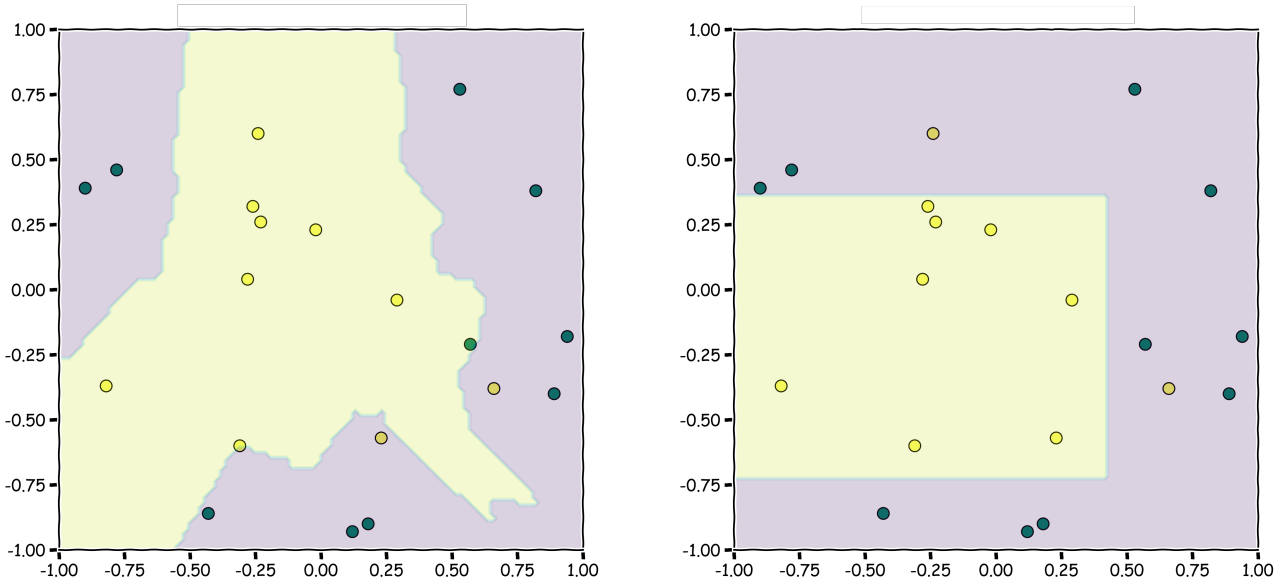


Figure 10: The input data about hitting the target.

For stacking, we will use 4 disjoint folds segmented at random. The base algorithms will be k-nearest neighbors ($k = 3$) and a decision tree (the maximum depth is 4, and the splitting criterion is entropy). To compare the results, we will apply the algorithms to the input data. The figures show that k-nearest neighbors (fig. 11a) made three mistakes, while the decision tree (fig. 11b) made two.

When four iterations of the stacking algorithm are completed, we obtain the responses of the selected base algorithms for each fold. The obtained results can be written in the table: In the introduced notations, we obtained meta-data



(a) Classification model: k-nearest neighbors. (b) Classification model: decision tree.

Figure 11: Comparison of classification models.

Fold No.	X_1	X_2	Y_{kNN}	Y_{DT}	Y
4	0.94	-0.18	0	0	0
3	0.12	-0.93	1	1	0
...	...	...	...	...	...
3	0.18	-0.90	1	1	0
2	-0.82	-0.37	1	0	1
...	...	...	...	...	...
1	0.29	-0.04	1	1	1

(meta-predictors) to train the meta-model

$$X_{meta} = \begin{pmatrix} 0 & 0 \\ 1 & 1 \\ \vdots & \vdots \\ 1 & 1 \\ 1 & 0 \\ \vdots & \vdots \\ 1 & 1 \end{pmatrix}.$$

We are going to train the meta-algorithm $a(x)$ based on the predictors X_{meta} and the response Y. We will use logistic regression as a meta-algorithm. The result is shown in fig. 12. As you can see, the final model has combined the features of both models.

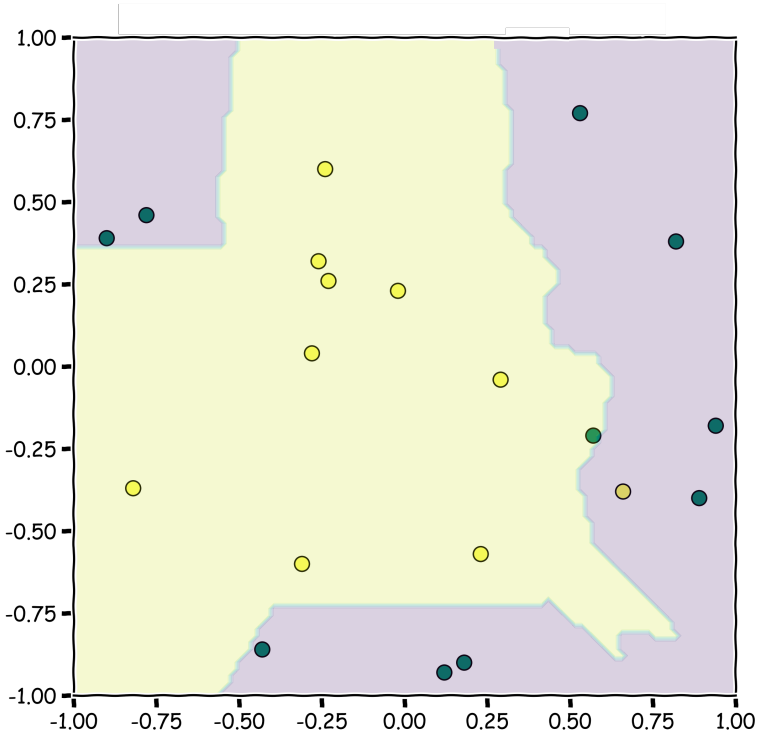


Figure 12: Classification model: stacking.

3.5 Random Forest

Earlier we considered an algorithm for constructing a decision tree. It can be easily modified using bagging. The constructed decision tree can achieve zero error on any training sample while being unstable on the test sample.

The random forest is an ensemble of decision trees based on bagging with one important addition (of one more random thing) of feature randomness. Thus, each tree in the forest will be trained on a random bootstrap sample and a set of features from the input training dataset. All of this makes trees different from each other, and the model becomes more resistant. A common recommendation is to use $m = [\frac{p}{3}]$ features for regression problems and $m = [\sqrt{p}]$ features for classification (square brackets denote the integer part of the number).

To sum up, let's write down the algorithm for constructing a random forest. Assume there is the training sample X of size n with p predictors. Then, for each tree $t = \{1, 2, \ldots, T\}$, it is necessary to:

1. Make a bootstrap sample X_t.
2. Randomly select m predictors from the original p predictors.
3. Construct a decision tree based on the sample X_t by keeping m predictors.
4. Repeat steps 1-3 T times.

5. Depending on the task, create a decision rule and an ensemble.

Another random forest advantage is that no cross-validation or test sample is needed to estimate misclassification. Trees are created using the bootstrap. As we know, the bootstrap sample consists of approximately 67% objects of the original sample. Thus, we can estimate the remaining ones.

Let's turn to synthetic data and consider the classification problem by way of comparing three approaches: decision trees, bagged decision trees, and a random forest (fig. 13). As you can see, decision trees are prone to overfitting. We can tell it by the corners of the borders and the protruding regions. In the case of bagging and the random forest, the borders are smoother, and there are no evident protrusions. In fact, the area looks whole. Overfitting rarely happens to a random forest classifier, in particular, when there are plenty of trees.

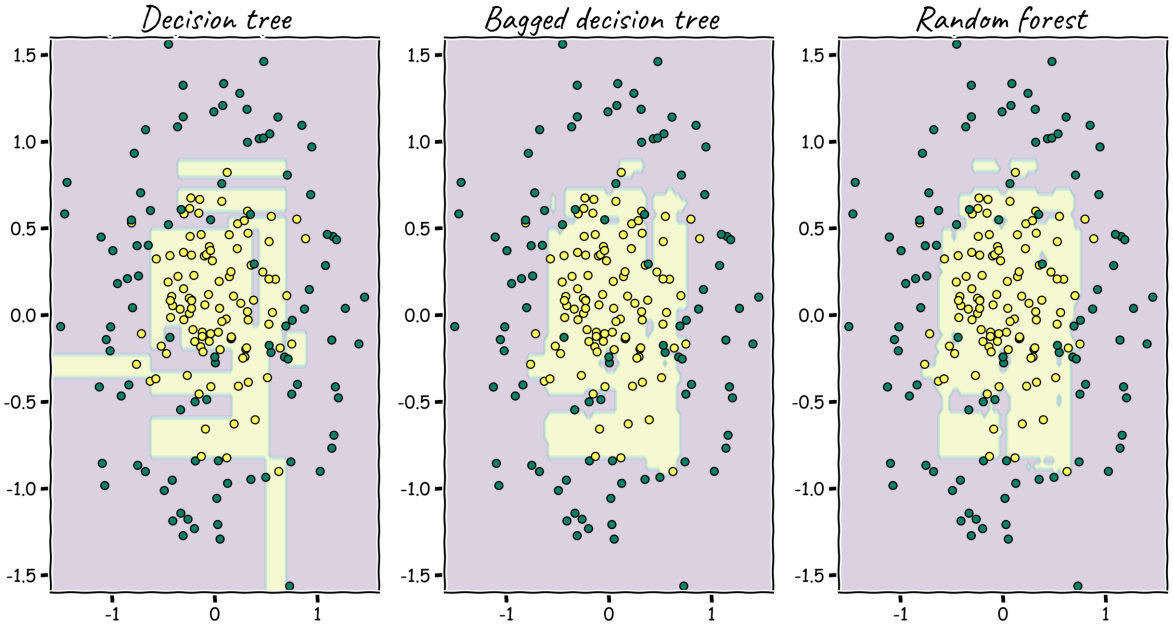


Figure 13: Model comparison.

The main limitation of a random forest is that multiple trees can make the algorithm slow and ineffective for real-time predictions. In general, these algorithms learn fast but, after that, they are slow at making predictions.

4 Multiclass Classification

Another important theme is a multiclass classification that uses logistic regression and SVM. We have already reviewed some of the approaches when considering the algorithms and ensembles.

The question is how to apply binary classification techniques when there are multiple classes. There are many ways to do it, but we will focus on the most common ones: one-vs-all and all-vs-all classification.

4.1 One-vs-All Classification

The first technique that we are going to consider is one-vs-all classification. Its principle is easy to guess from the name. Since we can classify an object by assigning it to one of two classes, when there are many classes, the model needs to answer whether the object belongs to the mth class, $m \in \{1, 2, ..., M\}$. Then, a decision rule is created based on the answers of the algorithm.

This approach that relies on a binary classifier is implemented in the following way. When the model is trained to determine if an object belongs to the class m, the elements of the mth class are assigned the response $+1$, and all the remaining the response -1. Let's move on to the formal algorithm.

1. Assume we are solving an M-class classification problem, $Y = \{1, \ldots, M\}$, and $b_i(x)$, $i \in \{1, 2, ..., M\}$, are base algorithms of logistic regression that output the value in the closed interval $[0, 1]$. Then for each $i \in \{1, 2, ..., M\}$

   (a) All objects that belong to the class i are assigned to $+1$ and -1 otherwise.

   (b) The binary classifier $b_i(x)$ is trained on the dataset with new responses.

2. The final classifier $a(x)$ will output the class corresponding to the number of the most confident binary classifier $b_i(x)$:

$$a(x) = \underset{i \in \{1,2,...,M\}}{\arg\max}\ b_i(x).$$

   When there are many possible values of i, the object can be assigned to any class or the extra class 0.

Base algorithms of logistic regression can be replaced by base algorithms of SVM, for example.

Remark 4.1.1 *In other words, when the classifier is based on logistic regression, we choose such a class i, for which the base algorithm $b_i(x)$ gives the highest probability. If we are considering SVM, then the class, for which the value of the classifier is the greatest $b_i(x)$.*

Remark 4.1.2 *Of course, the described decision rule is one of many possibilities. For example, when, for each i, the logistic regression outputs a value that is less than 0.5, there's no need to assign the object to a class due to a high level of*

uncertainty. In this case, we can change the decision rule, and the multiclass classifier $a(x)$ *can take the following form (with the extra class* 0 *and the principle of keeping silence rather than lying):*

$$a(x) = \begin{cases} b(x), & b(x) \geq 0.5 \\ 0, & b(x) < 0.5 \end{cases}, \quad b(x) = \underset{i \in \{1,2,\ldots,M\}}{\arg\max}\ b_i(x).$$

No doubt, it is the responsibility of a researcher to choose the right decision rule.

The simplicity of this algorithm comes with a cost. For example, you may find out during the training that all samples have almost the same sizes. Because of it, when each algorithm $b_i(x)$ is trained, the representatives of the negative class will dominate the representatives of the positive class. Therefore, the positive class will be underestimated, and the classifier will not be confident.

4.1.1 Example

Let's apply the algorithm to the data about sweetness and crispness of food. The table shows that the data is divided into three classes. The figure shows that the data is easy to group. Therefore, the multiclass classifier that relies on base algorithms of logistic regression should do well.

Table 1: Training data.

Product	Sweetness	Crispness	Class
banana	10	1	fruit
orange	7	4	fruit
grapes	8	3	fruit
shrimp	2	2	protein
bacon	1	5	protein
nuts	3	3	protein
cheese	2	1	protein
fish	3	2	protein
cucumber	2	8	vegetable
apple	9	8	fruit
carrot	4	10	vegetable
celery	2	9	vegetable
lettuce	3	7	vegetable
pear	8	7	fruit

First, for ease, we will enumerate the input classes as follows: 1 for fruit, 2 for vegetables, 3 for proteins. In the first iteration, when $i = 1$, we will train the

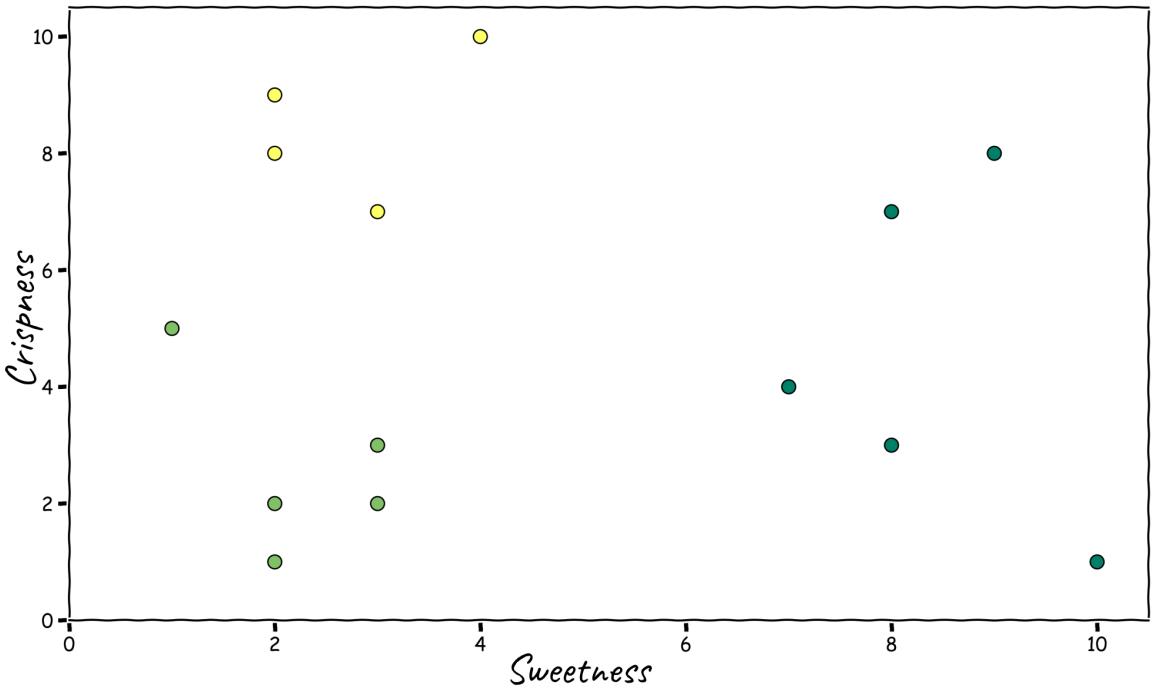


Figure 14: Visualization of the table data.

classifier to distinguish fruit from other objects. Based on the rule formulated in the algorithm, the data is assigned a new response Y^*. The results are shown in table 2.

Table 2: Iteration $k = 1$.

Product	Sweetness	Crispness	Class Y	Class Y^*
banana	10	1	1	+1
orange	7	4	1	+1
grapes	8	3	1	+1
shrimp	2	2	3	−1
bacon	1	5	3	−1
nuts	3	3	3	−1
cheese	2	1	3	−1
fish	3	2	3	−1
cucumber	2	8	2	−1
apple	9	8	1	+1
carrot	4	10	2	−1
celery	2	9	2	−1
lettuce	3	7	2	−1
pear	8	7	1	+1

Having trained the model, we use the logistic regression algorithm to obtain a

separating straight line defined by the equation (the coefficients are rounded to the nearest hundredth):

$$-6.23 + 1.22X_1 - 0.12X_2 = 0.$$

The steps are similar when $i = 2$ and $i = 3$. It leads us to two more straight lines defined by the equations:

$$7.52 - 0.95X_1 - 0.97X_2 = 0,$$

$$-5.53 - 0.57X_1 + 1.09X_2 = 0.$$

The plotted straight lines show that each of the constructed models solves the problem and makes no mistakes on the training data.

Now the classification of a new test observation is easy. Based on the algorithm, it is necessary to select the class that corresponds to the most confident classifier. Let's take the object Bell Pepper with the coordinates $(6, 9)$, find the value of Ψ_i for each classifier:

$$\Psi_1 = -6.23 + 1.22 \cdot 6 - 0.12 \cdot 9 = 0.01.$$

$$\Psi_2 = 7.52 - 0.95 \cdot 6 - 0.97 \cdot 9 = -6.91,$$

$$\Psi_3 = -5.53 - 0.57 \cdot 6 + 1.09 \cdot 9 = 0.86.$$

Next, we can find the probabilities of assigning an object to the positive class (each time to its own) by the formula:

$$\mathsf{P}_+ = \frac{1}{1 + e^{-\Psi}}.$$

We plug the obtained values in it to obtain the probabilities of assigning the object Bell Pepper to the class of fruit, protein, and vegetables respectively:

$$\mathsf{P}_+(\Psi_1) = \frac{1}{1 + e^{-0.01}} \approx 0.503,$$

$$\mathsf{P}_+(\Psi_2) = \frac{1}{1 + e^{6.91}} \approx 0.001,$$

$$\mathsf{P}_+(\Psi_3) = \frac{1}{1 + e^{-0.86}} \approx 0.703.$$

Hence, Bell Pepper will be classified as a vegetable.

Note that if a test object lies within the triangle (in the figure), which is formed by intersecting straight lines, the classification with respect to each of the trained algorithms will not be confident (the probabilities will be less than 0.5). For such objects, we need to consider whether we need to classify them at all.

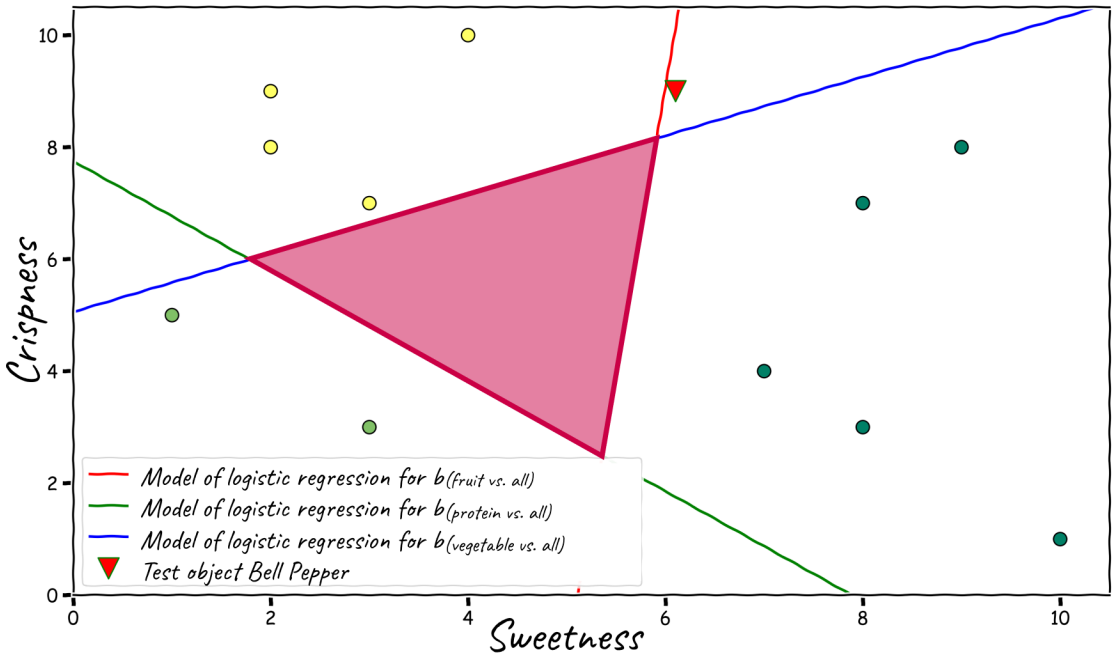


Figure 15: The base models of logistic regression.

4.2 All-vs-All Classification

The second approach is like the first one. It is based on the same capability to assign the object to one of two classes. Earlier, we collided each class with the remaining ones. Now, we are going to pit each class against every other class. Let's move on to the algorithm.

1. Assume we are solving an M-class classification problem, $Y = \{1, 2, ..., M\}$, and $a(x)$ is an algorithm of binary logistic regression that outputs the class $+1$ or -1. Then for each $i, j \in \{1, 2, ..., M\}$, $i < j$

   (a) Draw a subsample X_{ij} from the original sample X, which contains all the objects that belong to the classes i and j.

   (b) Assign the elements of the class i the response $+1$, and the elements of the class j the response -1.

   (c) Train the algorithm $a(x)$ and create a new algorithm $a_{ij}(x)$ that assigns correct numbers to the classes

   $$a_{ij}(x) = \begin{cases} i, & a(x) = +1 \\ j, & a(x) = -1 \end{cases}.$$

   (d) Assume that $a_{ji}(x) = a_{ij}(x)$, $a_{ii}(x) = 0$.

2. The final classifier $a(x)$ outputs the class based on the majority voting of all the trained algorithms:

$$a(x) = \underset{m \in \{1,2,\ldots,M\}}{\arg\max} \sum_{i=1}^{K} \sum_{j=1}^{K} \mathsf{I}(a_{ij}(X) = m).$$

When there are many possible values of m, the object can be assigned to any class or the extra class 0.

4.2.1 Example

Let's return to the same data shown in table 1. The response consists of three unique values: fruit, vegetables, protein. Thus, the pairs of the colliding classes are (fruit, vegetables), (fruit, protein), (vegetables, protein). Let's create the subsample for the first pair (table 3).

Table 3: Training subsample $X_{\text{(fruit, vegetable)}}$.

Product	Sweetness	Crispness	Class
banana	10	1	fruit
orange	7	4	fruit
grapes	8	3	fruit
cucumber	2	8	vegetable
apple	9	8	fruit
carrot	4	10	vegetable
celery	2	9	vegetable
lettuce	3	7	vegetable
pear	8	7	fruit

We are going to use logistic regression as a classification algorithm. The modeling leads us to the equation of a separating straight line:

$$-2.11 + 0.95X_1 - 0.44X_2 = 0,$$

which separates fruit from vegetables (because the information about the proteins denoted by the lime-green points in the bottom left corner was not used for the training). The normal to the straight line with the coordinates $(0.95, -0.44)$ is pointed towards the green points (fruit).

The subsamples $X_{\text{(fruit, protein)}}$ and $X_{\text{(vegetable, protein)}}$ are formed similarly. We obtained for them the equations of the following separating straight lines:

$$-6.36 + 1.04X_1 + 0.28X_2 = 0,$$

$$-6.96 + 0.33X_1 + 1.07X_2 = 0.$$

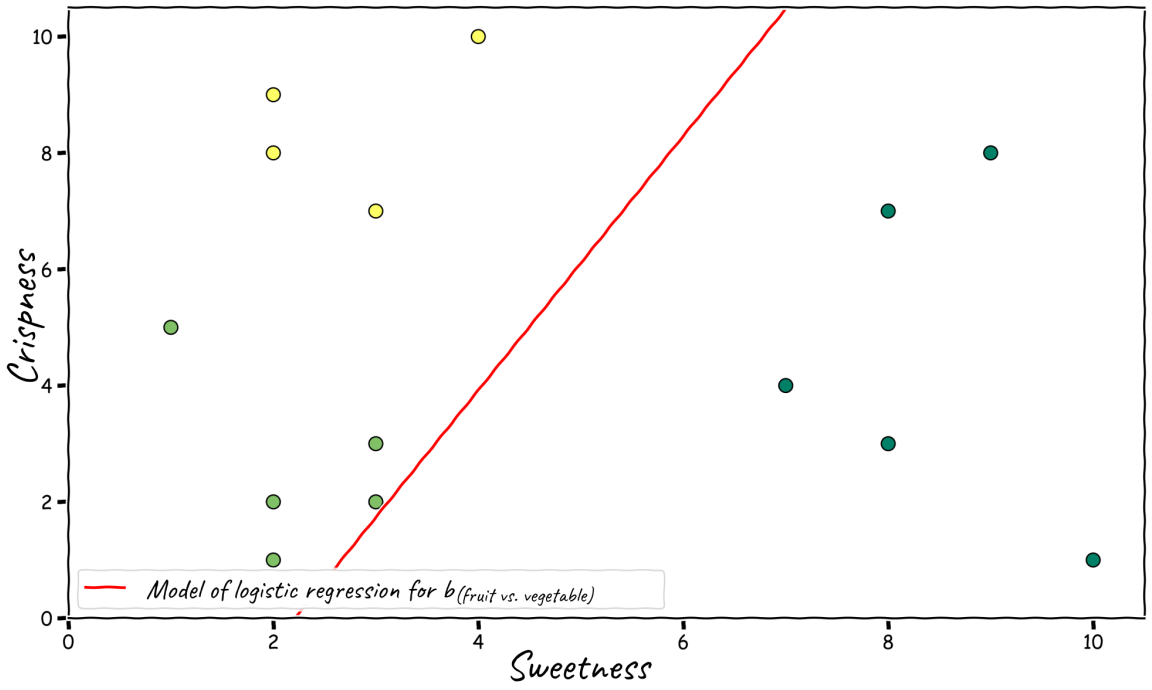


Figure 16: Training data and regression for the classes of fruit and vegetables.

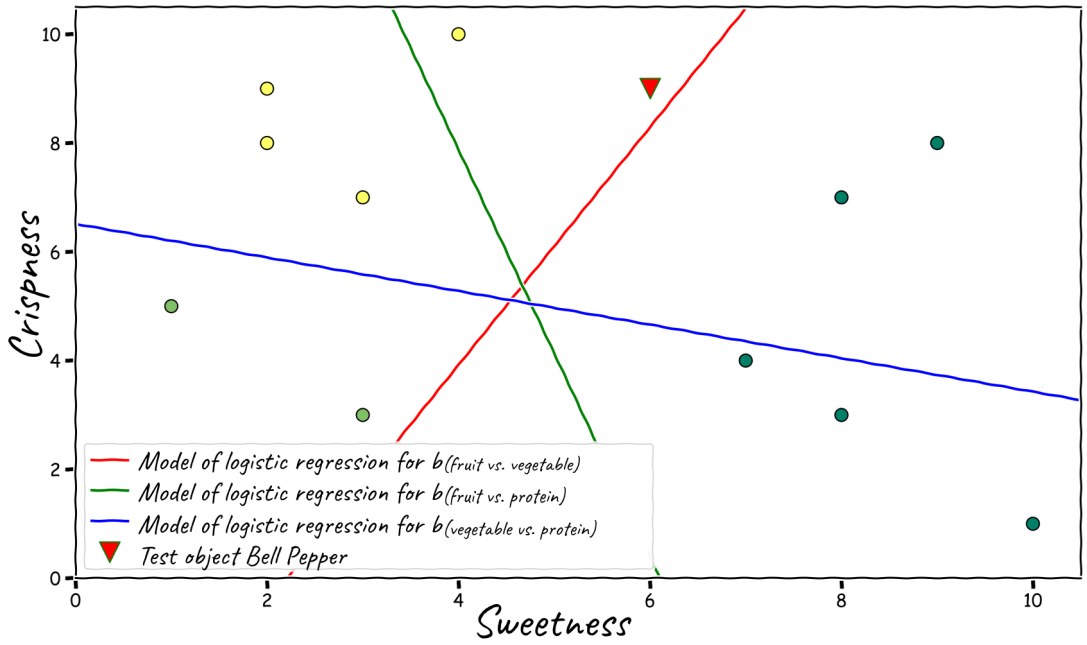


Figure 17: Training data and base algorithms.

All the straight lines are shown in fig. 17.

Now let's test our model. We will take the object Bell Pepper with the coordinates (6,9) to find the values of Ψ_i for each classifier and the corresponding probabilities:

$$\Psi_1 = -2.11 + 0.95 \cdot 6 - 0.44 \cdot 9 = -0.37,\ \mathsf{P}_+(\Psi_1) = \frac{1}{1+e^{0.37}} \approx 0.41,$$

$$\Psi_2 = -6.36 + 1.04 \cdot 6 + 0.28 \cdot 9 = 2.4,\ \mathsf{P}_+(\Psi_2) = \frac{1}{1+e^{-2.4}} \approx 0.92,$$

$$\Psi_3 = -6.96 + 0.33 \cdot 6 + 1.07 \cdot 9 = 4.65,\ \mathsf{P}_+(\Psi_3) = \frac{1}{1+e^{-4.65}} \approx 0.99.$$

Well, the first model assigns the object Bell Pepper to the class of vegetables ($\mathsf{P}_+ < 0.5$). The second model assigns it to the class of fruits, and the third to the class of vegetables. Thus, the object Bell Pepper is assigned to the class of vegetables by the majority of votes.