What is machine learning

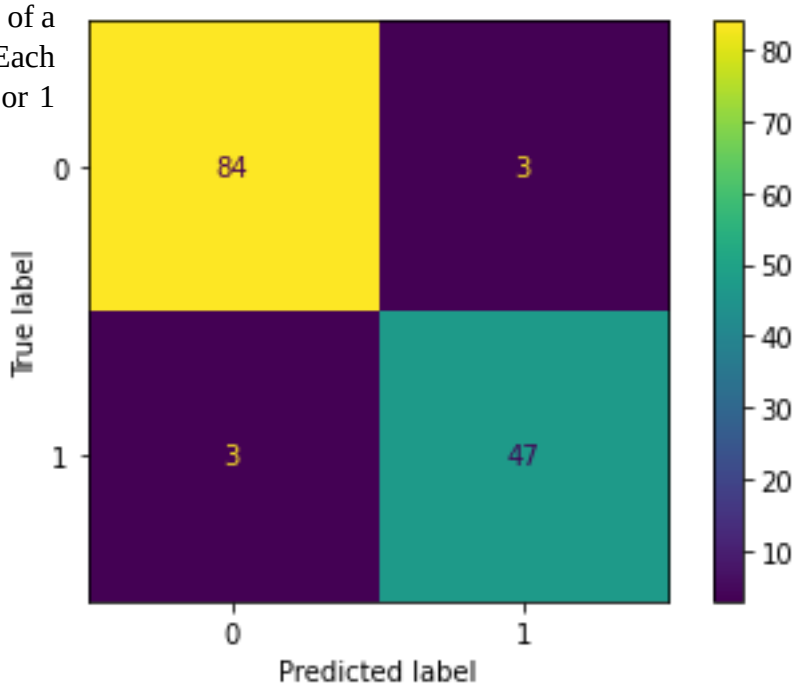
Machine learning is a growing branch of computational algorithms that aims to emulate human intelligence by learning from the surrounding environment. They are considered the *workhorse* of the new era of so-called big data. It is a field of artificial intelligence that involves creating algorithms that learn from data and experience. Techniques based on machine learning have been successfully applied in various fields, from pattern recognition, computer vision, aerospace engineering, finance, entertainment, and computational biology to biomedical and medical applications. An intelligent system should be able to adapt to the environment it is in at any given time, meaning it should learn from its past mistakes and instead repeat its successes. Scientists used to think that creating artificial intelligence required a new way of thinking, which would be more than unattainable for us. As it turned out, all humanity needed was data and sufficient computational power to apply our models to a large amount of data. The goal of machine learning is to build systems that can automatically adjust their behaviour based on the data they receive. Unlike traditional programming methods, in which programmers must manually input rules and algorithms, machine learning allows computers to learn from data and draw conclusions on their own. This enables computers to solve increasingly complex problems and perform increasingly complex tasks.

Types of machine learning

There are three types of machine learning: supervised learning, unsupervised learning, and reinforcement learning. The latter occurs when our model operates in a certain environment. Assisted learning is a machine learning technique in which the learning algorithm is supported by a human. Unlike supervised learning, in which the algorithm is completely guided by the training data, in assisted learning, the human plays the role of a moderator who provides guidance and assistance to the algorithm in its learning. One example of assisted learning is using a machine learning algorithm to recognize objects in photos. In this case, the algorithm is equipped with a set of training data that contains photos of various objects, such as cars, buildings, and people, along with labels specifying what they represent. The algorithm tries to learn to recognize these objects based on the training data. If the algorithm has difficulty recognizing a particular object in the photo, a human can help by adding additional training data or by pointing the algorithm to the characteristics of the object it should look for in order to recognize it. This allows the algorithm to learn faster and better than in the case of supervised learning, where its learning is entirely dependent on the training data provided to it. In this work, we will only focus on supervised and unsupervised models. The latter refers to when we deal with labelled data. Supervised learning is a machine learning technique in which the algorithm learns from training data that is already labelled with labels specifying what they represent. For example, if we want to teach an algorithm to recognize objects in photos, we can provide it with a set of photos of various objects, such as cars, buildings, and people, along with labels specifying what they represent. The algorithm will learn to recognize these objects based on the training data. In contrast, in unsupervised learning, the algorithm is not given any labelled training data and must find patterns and relationships in the data on its own. In contrast to assisted learning, where a human plays the role of a moderator and provides guidance to the algorithm, in supervised learning, the algorithm is completely guided by the training data provided to it. This can make it more effective at recognizing known objects, but it may have

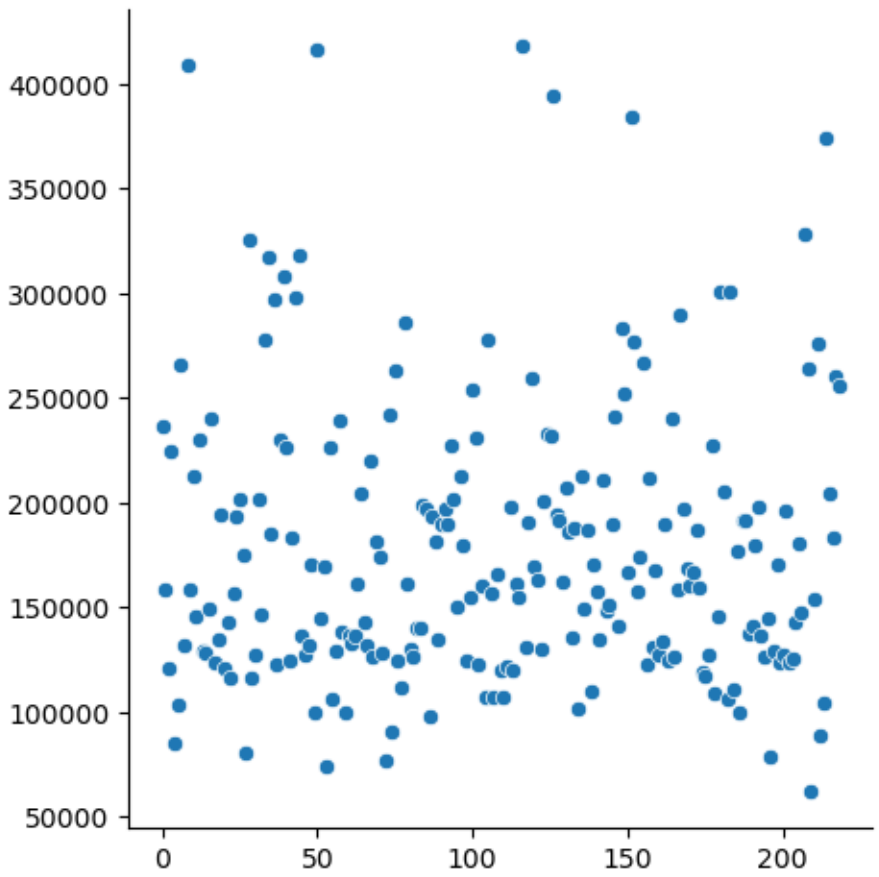
difficulty recognizing new or unknown objects. There are two types of supervised learning: regression and classification.

Classification is a type of machine learning that aims to assign data to specific categories or classes. For example, a classification algorithm can be used to recognize objects in photos, classify text as positive or negative, or determine if medical data indicates a certain disease. To perform classification, the algorithm uses training data that is already labelled with labels specifying the belonging to particular categories or classes. The algorithm learns from this data and creates models that allow it to assign new data to the appropriate categories or classes. Classification is often used in various fields such as medicine, advertising, security, and others where it is important to automatically assign data to specific categories. This is an example of the result of a binary classification model: Each point has been classified as 0 or 1 (binary classification).



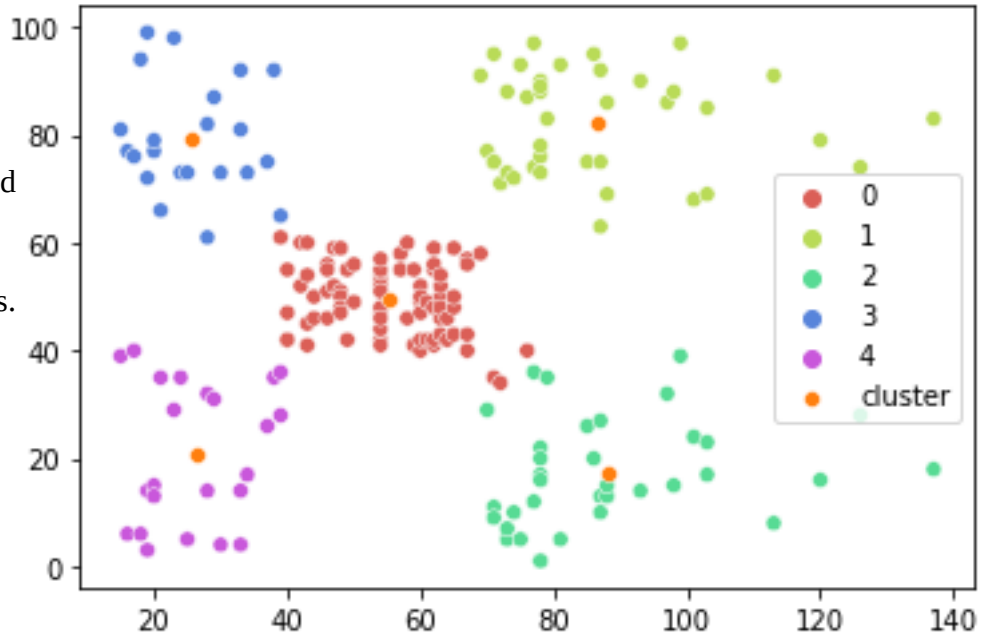
Regression is a type of machine learning that aims to predict numerical values based on data. For example, a regression algorithm can be used to predict real estate prices based on their characteristics such as size, location, and age, or to predict the number of products sold based on factors such as price and advertising campaigns. To perform regression, the algorithm uses training data that contains pairs of feature values and numerical values to be predicted. The algorithm learns from this data and creates models that allow it to predict numerical values for new data. Regression is often used in various fields such as economics, medicine, finance, and others where it is important to predict numerical values based on data. This is an example of

the result of a regression model: Each point corresponds to one row with certain attributes, based on which the model has calculated a certain value.



Unsupervised learning is a machine learning technique in which the algorithm learns from data that is not labelled with labels specifying the belonging to particular categories or classes. Instead, the algorithm searches for patterns and structures in the data to understand its character. For example, an unsupervised learning algorithm can be used to analyse transaction data to find groups of customers with similar purchasing habits. Or it can be used to analyse medical data to find patients with similar characteristics and medical history. Unsupervised learning is often used in various fields where it is important to understand the structure and patterns in data without prior labelling of the data with labels. It can also be used as a prelude

to supervised or assisted learning to help the algorithm better understand the data. This is an example of the result of such a model:



The model was tasked with dividing the data points into 5 groups. These groups are often called clusters.

Datasets

Datasets in machine learning play a crucial role in the process of creating machine learning models. These are sets of data that are used to train the model and test its accuracy. Datasets are typically divided into two parts: a training set and a test set. The training set is used to train the model and search for patterns and relationships in the data. The test set is used to test the accuracy of the model on new, unseen data. It is important that data sets are sufficiently large and representative of the problem being solved. If the dataset is too small or not properly chosen, the model may be inaccurate. There are various methods for selecting and preparing datasets, such as splitting into random subsets, dividing according to criteria that determine the target group, or conducting data exploration to identify important features and remove missing or irrelevant data. In addition, sometimes it is necessary to perform additional data processing, such as normalization or scaling, to ensure better model performance. In machine learning, it is also important that data sets are balanced, meaning that all classes in the set are represented in the appropriate proportions. Otherwise, the model may perform better on some classes and be less accurate for others. In summary, datasets are a key element in the process of creating machine learning models, and their proper selection and preparation can significantly impact the accuracy and effectiveness of the model.

Data preprocessing

Scaling data is the process of changing the range of values of data so that they are on the same level. This is important because many machine learning models work better when the

data is on the same scale. For example, if one attribute has values from 0 to 1000 and another attribute has values from 0 to 1, the model may be more sensitive to the attribute with the smaller range of values. There are many ways to scale data, including:

1) Min-max scaling: involves changing the range of values of a given attribute to a specified range (e.g., from 0 to 1).
   a) In this case, the most commonly used range is from 0 to 1. To perform Min-Max scaling, you should follow these steps:
      i) Find the minimum and maximum of the given attribute. You can do this by searching the entire dataset and recording the largest and smallest value.
         (1) For each value of attribute x, calculate the new value using the formula: $x_{scaled} = \frac{x - min}{max - min}$, where min and max are the minimum and maximum of the given attribute, respectively.
2) Normalized scaling: involves changing the range of values of a given attribute so that the average value is 0 and the standard deviation is 1.
   a) To normalize data using normalized scaling, you can use the following formula: $x_{normalized} = \frac{x - \mu}{\sigma}$
      where x is the original value and $x_{normalized}$ is the normalized value. μ represents the mean of the data and σ represents the standard deviation.
3) Logarithmic scaling: involves changing the range of values of a given attribute by applying the logarithm.
   a) To normalize data using logarithmic scaling, you can use the following formula: $x_{normalized} = log(x + 1)$
      where x is the original value and $x_{normalized}$ is the normalized value. Adding 1 to x is necessary to avoid dividing by 0 for x=0. Logarithmic scaling is especially useful when the data has large value ranges or there is mutual correlation between them.
4) Exponential scaling: involves changing the range of values of a given attribute by applying the exponent.
   a) To normalize data using exponential scaling, you can use the following formula: $x_{normalized} = e^x$
      where x is the original value and $x_{normalized}$ is the normalized value. Exponential scaling is especially useful when the data has a normal distribution or there is a mutual correlation between the values.

These are just a few examples of data scaling methods. The appropriate scaling method will depend on the characteristics of the data and the requirements of the machine learning model being used.

Cost function

A cost function is a mathematical tool used to determine the quality of a statistical or machine learning model. Its main purpose is to evaluate how well the model predicts output data based on input data. The cost function is usually defined as the sum of squared errors (SSE) between the actual values of the output data and the predicted values of the model. The smaller the value of the cost function, the better the model predicts the output data. The cost function is used to optimize the model by minimizing its value. To do this, optimization algorithms such

as gradient descent are used, which involve gradually reducing the value of the cost function by changing the parameters of the model. There are different types of cost functions that are used depending on the type of model and the data being worked with. Examples of such functions include:

1. Linear cost function: used in linear models where the values of the output data are linearly dependent on the input data.
2. Logarithmic cost function: used in logistic regression models where the goal is to predict the probability of a certain event occurring.
3. Curve cost function: used in binary classification models where the goal is to predict belonging to one of two classes.
4. Cross-entropy curve cost function: used in multi-class classification models where the goal is to predict belonging to one of many classes.

Gradient descend

Gradient descent is an optimization algorithm that seeks optimal values for model parameters by iteratively modifying them in the direction of minimizing error. To do this, the algorithm calculates the gradient of error with respect to each model parameter and modifies the parameter values according to the gradient. The math involved in the optimization process using gradient descent is as follows: Cost function. For a linear regression model, the cost function can be represented in the form of the following equation:

$$E = \sum_{i=1}^{n} (y_i - y_{pred,i})^2$$

where:

- y_i is the true values of the target variable for training data.
- $y_{pred,i}$ is the model's predictions for the training data.
- E is the error.

The goal of this algorithm is to find optimal values for the model parameters that minimize the cost function. The gradient is the changes that need to be made to the parameters in order to minimize the error. It can be calculated using the derivatives of the cost function with respect to each model parameter. In the case of linear regression, the gradient can be represented in the form of the following equations:

$$\frac{\partial E}{\partial w_1} = \sum_{i=1}^{n} 2 * (y_i - y_{pred,i})(-x_{1,i})$$

$$\frac{\partial E}{\partial w_2} = \sum_{i=1}^{n} 2(y_i - y_{pred,i})(-x_{2,i})$$

...

$$\frac{\partial E}{\partial w_n} = \sum_{i=1}^{n} 2\left(y_i - y_{pred,i}\right)\left(-x_{n,i}\right)$$

$$\frac{\partial E}{\partial b} = \sum_{i=1}^{n} 2\left(y_i - y_{pred,i}\right) * (-1)$$

where:

- $w_1, w2, \ldots, wn$ are the model parameters that determine the impact of each input data on the target variable.
- $x_{1,i}, x_{2,i}, \ldots, x_{n,i}$ are the input data.
- b is the bias term that determines additional impact on the target variable.

The optimization process using gradient descent involves iteratively modifying the values of the model parameters according to the gradient. This can be done using the following equations:

$$w_1 = w_1 - \alpha * \frac{\partial E}{\partial w_1}$$

$$w_2 = w_2 - \alpha * \frac{\partial E}{\partial w_2}$$

$$\ldots$$

$$w_n = w_n - \alpha * \frac{\partial E}{\partial w_n}$$

$$b = b - \alpha * \frac{\partial E}{\partial b}$$

where:

- α is the learning rate, which determines the size of the step taken in the direction of the gradient.

The optimization process continues until the cost function reaches a minimum value or until a predefined number of iterations is reached. The value of alpha is an important hyperparameter that should be set appropriately for the given problem. A value of alpha that is too small will cause a long learning time, while a value that is too large may cause instability or lack of progress in optimization.

Classification

The simplest algorithm that can be used for classification is called KNN (k-nearest-neighbours). The algorithm counts the distance from our variable to known points, and then classifies it based on the number of neighbours belonging to a particular category among the k nearest neighbours.

To implement KNN:

1. Initialize the variable k.
2. Calculate the distance from all points for the given variable.
3. Record the distance from the given point and its value in a data structure.
4. Sort the data structure in ascending order by distance.
5. Select the nearest k points.
6. Assign the appropriate value to our variable.

This algorithm is not very effective with large data sets. It is simply computationally inefficient. When working with fifty thousand points and calculating the distance from each of these fifty thousand points for each row in the test set, the algorithm simply takes too long. For this reason, KNN is only used when we have relatively little data, and is not used for more than five thousand rows.

Logistic regression is a type of classification model that predicts the probability of a given example belonging to a particular class. The logistic regression model is based on the theory of probability and mathematical statistics. It works by predicting the probability of a given example belonging to a particular class based on its features. It can be used in many different situations, such as analysing medical, financial, or marketing data. The logistic regression model is most commonly trained using the gradient descent method, which allows it to find the optimal values for the model parameters. Once trained, the model is able to predict the probability of a given example belonging to a particular class based on its features. Logistic regression is a popular tool in machine learning because it is easy to implement and gives good results in many applications. It can be used for binary classification or multi-class classification, depending on the number of classes to which the examples belong. Logistic regression is often used to understand the relationship between features and classes. It allows you to identify important features that influence the probability of a given example belonging to a particular class, as well as determine how much each of these features affects this membership. This can be helpful in building better classification models or interpreting the results of other models. In logistic regression, the probability of belonging to a particular class is calculated using the model hypothesis $h_\theta(x)$. This hypothesis is defined as:

$$h_\theta(x) = 1/\left(1 + e^{(-\theta^T x)}\right)$$

Where θ is the vector of model parameters and x is the vector of features for a given sample. This hypothesis is interpreted as the probability of belonging to class 1 for a given sample. In particular, if $h_\theta(x) > 0.5$, the sample is predicted to belong to class 1, and if $h\_\theta(x) < 0.5$, the sample is predicted to belong to class 0. The goal in training a logistic regression model is to find the optimal values for the parameters θ that best predict the class membership of the training examples.

There are two types of logistic regression:

- Binary
- Multiclass (also called polytomous), where the goal is to predict the probability of one of several (more than two) possible outcomes.

To calculate the probability of belonging to each class, you can use "one-hot encoding," where each class is represented by one column of ones and the rest of the values are zeros. For example, for three classes we could have the following labels:

Class 1: [1, 0, 0]

Class 2: [0, 1, 0]

Class 3: [0, 0, 1]

In this case, the model hypothesis is defined as:

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

Where θ is the matrix of model parameters and x is the vector of features for a given sample. The θ matrix has size $K \times (n+1)$, where K is the number of classes and n is the number of features. This hypothesis is interpreted as a vector of probabilities of belonging to the different classes for a given sample. For example, if $h_\theta(x) = [0.8,0.1,0.1]$, the sample is predicted to belong to class 1 with probability 0.8 and to classes 2 and 3 with probability 0.1. To calculate the probability of belonging to each class, you can use the $softmax$ function, which is defined as:

$$softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}$$

Where $x_i$is the i-th element of the vector $h_\theta(x)$. The softmax function ensures that all elements of the $h_\theta(x)$ vector are positive and sum to 1, which means they are interpreted as probabilities of belonging to the different classes. For example, if $h_\theta(x) = [3,1,0]$, then $softmax\big(h_\theta(x)\big) = [0.8,0.1,0]$, which means the sample is predicted to belong to class 1 with probability 0.8 and to classes 2 and 3 with probability 0.1. Since the $softmax$ function ensures that all elements of the $h_\theta(x)$ vector are positive and sum to 1, it is often used in multiclass logistic regression to calculate the probability of belonging to each class.

The cost function for logistic regression is usually referred to as the logistic curve cost function or the log-loss function. It is defined as:

$$\mathrm{J}(\theta) = -\frac{1}{\mathrm{S}}\mathrm{m}\sum_{\mathrm{i}=1}^{\mathrm{m}}\left[\mathrm{y}^{(\mathrm{i})}\log\left(\mathrm{h}_\theta\left(\mathrm{x}^{(\mathrm{i})}\right)\right) + \left(1-\mathrm{y}^{(\mathrm{i})}\right)\log\left(1-\mathrm{h}_\theta\left(\mathrm{x}^{(\mathrm{i})}\right)\right)\right]$$

where:

- $h_\theta(x)$ is the model's hypothesis, defined as:
  - $\mathrm{h}_\theta(\mathrm{x}) = \frac{1}{1+\mathrm{e}^{-\theta^{\mathrm{T}}\mathrm{x}}}$
- θ is a vector of the model's parameters
- m is a number of training data samples
- $x^{(i)}$ is a vector of features for the i-th sample
- $y^{(i)}$ is a true label for the i-th sample (0 or 1).

The cost function is minimized by optimizing the parameters θ using a machine learning algorithm such as gradient descent. In general, the goal is to find a θ that minimizes the model error, i.e., the difference between the true label and the label predicted by the model.

The cost function for multiclass logistic regression is usually referred to as the logistic curve cost function or the log-loss function. It is defined as:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}\left[y_k^{(i)} log\left(h_\theta\left(x^{(i)}\right)\right) + \left(1 - y_k^{(i)}\right) log\left(1 - h_\theta\left(x^{(i)}\right)\right)\right]$$

where:

- m is a number of samples,
- K is a number of classes,
- $h_\theta(x)$ is the model's hypothesis,
- $y^{(i)}$ is a label of $x^{(i)}$.

In the vector $y^{(i)}$, each class is represented by one column of ones and the rest zeros, as in the case of "one-hot" encoding. The logistic curve cost function is calculated for each sample and summed, then divided by the number of samples. This is the so-called average log-loss. The goal is to find the parameters θ that minimize the logistic curve cost function. This can be done using an optimization algorithm such as gradient descent. To do this, the gradient of the cost function with respect to the parameters θ is calculated and applied to the current parameter values to update them in the direction of minimizing the cost function. This process is repeated until a satisfactory accuracy is achieved or after a maximum number of iterations is reached. Once the optimal parameters θ are found, the model can be used to predict the probability of belonging to each class for new samples using the $softmax$ function.

Support Vector Machine (SVM) is a machine learning algorithm that is used for classification and regression. The main goal of SVM is to find a hyperplane that best separates the samples belonging to different classes. If we have training data consisting of samples $x_1, x_2, \dots, x_n$belonging to one of two classes $y_i \in -1,1$, then SVM seeks a hyperplane that best separates them. This hyperplane is represented by a vector w and an offset b, and can be written as:

$$w^T x + b = 0$$

where w^T denotes the transpose of vector w.

The distance of sample x_i from the hyperplane is equal to:

$$\frac{w^T x_i + b}{|w|}$$

The goal of SVM is to find a hyperplane such that the distance of the samples from it is as large as possible (called the margin). We can achieve this by optimizing the following equation:

$$\frac{1}{|w|}\min_{w,b}\sum_{i=1}^{n}\max\left(0,1-y_i(w^T x_i+b)\right)$$

The first part of this equation ensures that the margin is as large as possible, while the second part ensures that all the samples are adequately far from the hyperplane. In the case where the data is not linearly separable, SVM uses a so-called kernel to transform the data into higher dimensions such that it becomes linearly separable. The kernel is a function that maps data from lower dimensions to higher dimensions. This makes it possible to find a hyperplane that separates the data even when they are non-linearly separable in lower dimensions. Once trained, SVM is able to predict the class of new cases based on which side of the hyperplane they are on. If a new case x is on the side of the hyperplane for which $w^T x + b > 0$, it is classified as class 1, while if $w^T x + b < 0$, it is classified as class -1. SVM is effective and is particularly effective for small data sets. However, their operation is time-consuming and they can be sensitive to noise in the data. SVM is also sensitive to feature scaling, so some data transformation may be necessary before using them.

Regression

To use the KNN algorithm for regression, it is first necessary to familiarize oneself with the input data and determine the target variable that we want to predict. Then, as in the case of classification, the distance of each point in the test data must be calculated between points in the training set. Finally, the k nearest points must be determined and their average value is taken as the model's prediction for that point. This algorithm still faces the same problems as it did in the case of classification. It is usually used only when we have very few points and training more complex models will not give us accurate results. This method is most often used only for classification.

Linear regression is one of the simplest and most commonly used regression models that are used to predict the value of the target variable based on the input data. This model is based on the assumption that there is a linear relationship between the target variable and the input data. In the case of linear regression, the goal is to find the optimal values of the model parameters so as to obtain the best linear regression on the training data. This can be done using the gradient descent algorithm or other optimization algorithms. Once the optimal parameters of the model have been found, it can be used to predict the value of the target variable for new input data. The accuracy of the model can be assessed by calculating the difference between the model's predictions and the actual values for the test set. The linear regression model is based on the assumption that there is a linear relationship between the target variable y and the input data x. It can be represented in the form of the following equation:

$$y = w1 * x1 + w2 * x2 + \ldots + wn * xn + b$$

where:

- y is a target variable which we want to predict.

- $x1, x2, \ldots, xn$ are all entry data.
- $w1, w2, \ldots, wn$ are model's parameters, which determine the influence of every entry data feature on the target variable
- b is the bias term that determines additional impact on the target variable.

The goal of the linear regression model is to find the optimal values of the parameters $w1, w2, \ldots, wn$, and b so as to obtain the best linear regression on the training data. This can be done using the gradient descent algorithm or other optimization algorithms. The gradient descent algorithm works by iteratively modifying the values of the model parameters in the direction of minimizing the error. The error is calculated as the sum of the squares of the differences between the model's predictions and the actual values for the training data. It can be represented in the form of the following equation:

$$E = \sum_{i=1}^{n} \left(y_i - y_{pred,i}\right)^2$$

where:

- y are real values of the target variable for the training data.
- y_{pred} are model's predictions for the training data.
- E is a loss.

To summarize, linear regression is a simple and effective way of predicting the value of the target variable based on the input data when there is a linear relationship between them. It is widely used in many fields such as econometrics, finance, and management science.

Polynomial regression is a type of regression that allows for fitting a polynomial of any degree to the data. This is especially useful when the data is not linear and the relationship between the dependent variable and the independent variables is more complex. To build a polynomial regression model, it is necessary to choose an appropriate degree of the polynomial and use the least squares method to find the optimal parameters of the polynomial that best fit the data. The polynomial regression model can be interpreted as a function that predicts the value of the dependent variable based on the values of the independent variables.

Let's assume that we have training data consisting of the pairs $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, where x_i are the independent variables and y_i are the values of the dependent variable for each pair. The polynomial regression model is in the form:

$$\hat{y} = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_d x^d$$

where $\hat{y}$ is the predicted value of the explained variable for a given x and $\beta_0, \beta_1, \ldots, \beta_d$ are the parameters of the model that need to be found. This can be done using the least squares method, which involves minimizing the sum of squared errors (SSE) between the predicted value $\hat{y}$ and the actual value y:

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

To find the parameters $\beta_0, \beta_1, \ldots, \beta_d$ we must solve the equation:

$$\left[\sum_{i=1}^{n} x_i^2 \quad \sum_{i=1}^{n} x_i \quad \sum_{i=1}^{n} 1 \; \sum_{i=1}^{n} x_i^3 \quad \sum_{i=1}^{n} x_i^2 \quad \sum_{i=1}^{n} x_i \ldots \quad \ldots \quad \ldots \sum_{i=1}^{n} x_i^{d+1} \quad \sum_{i=1}^{n} x_i^d \quad \sum_{i=1}^{n} x_i^{d-1}\right]\left[\beta_0 \; \beta_1 \ldots \beta_d\right] =$$
$$= \left[\sum_{i=1}^{n} y_i \quad \sum_{i=1}^{n} x_i y_i \ldots \sum_{i=1}^{n} x_i^d y_i\right]$$

After solving the equation, the optimal parameters of the model can be obtained. Then, the model can be used to predict the value of the explained variable for new data by predicting $\hat{y}$ for new values of x. The quality of the model can also be assessed using various quality measures such as the mean squared error (MSE) or the coefficient of determination (R^2).

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

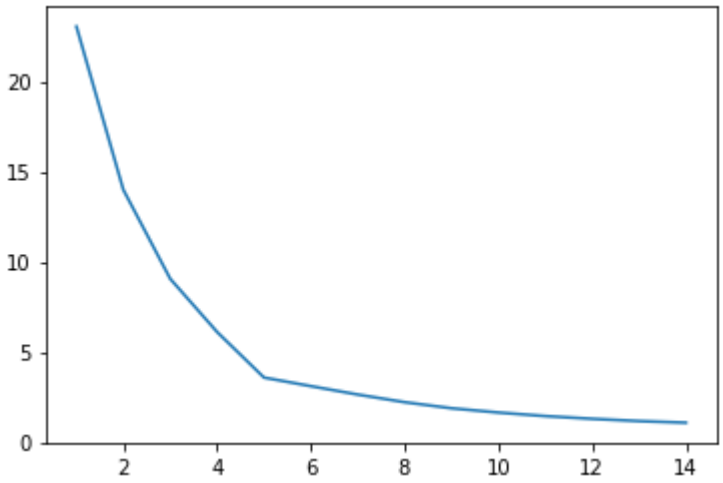
$$R^2 = 1 - \frac{SSE}{SST}$$

where SST is the total sum of squares between the actual values of y and their mean. The value of R^2 is always between 0 and 1, where a value closer to 1 indicates a better fit of the model to the data.

To summarize, polynomial regression is a method that allows for fitting a polynomial of any degree to the data, which allows for modelling more complex relationships between variables. It can be used for predicting the value of the explained variable for new data and assessing the quality of the model using various quality measures.

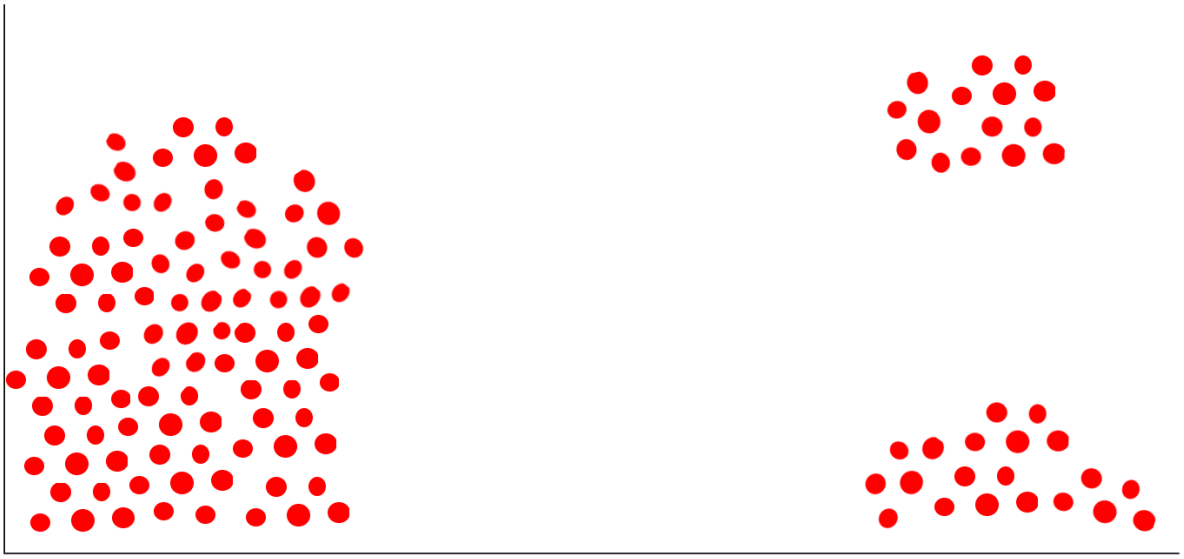
Unsupervised learning

K-means clustering is a popular data clustering algorithm that involves dividing data into k clusters (groups) so that the average distance between data points and the centroid (mean) of each cluster is minimized. The centroid of a cluster is the arithmetic mean of all points in the cluster. The k-means algorithm is iterative and consists of the following steps: Choose k random points as the initial centroids of the clusters. For each data point, assign it to the nearest centroid. For each cluster, calculate a new centroid as the arithmetic mean of all points in the cluster. Repeat steps 2 and 3 until the centroids do not change or the maximum number of iterations is reached. K-means is fast and simple to implement, but it has several limitations. Firstly, the number of clusters to be obtained must be known, which may be difficult to determine. However, this does not mean it is impossible. One of the simpler methods is to compare the sums of the distances of each point to its centroid squared, for different numbers of clusters. Then a simple graph is plotted, which looks approximately like this:

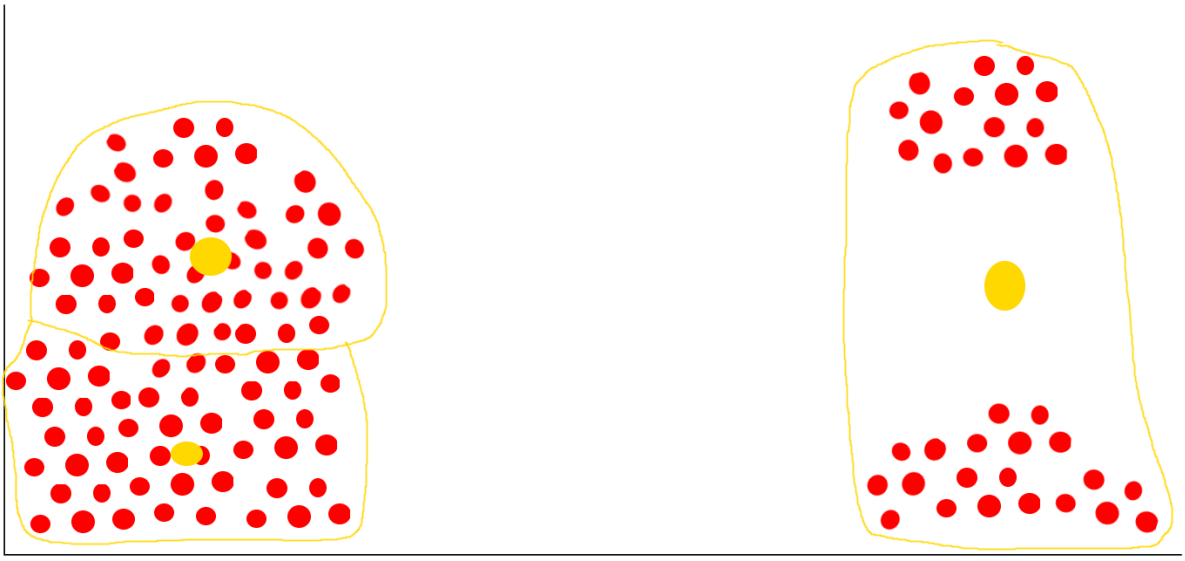


The Oy axis is the sum of distances, while the Ox axis is the number of clusters. It is commonly agreed that the appropriate number of clusters is indicated by *the elbow*. In this case, the optimal number of clusters would be 5 or 4. *The elbow* is not defined in any way, it is simply a term to help choose the parameter k.

Secondly, the algorithm is sensitive to the initial centroid values and may give different results depending on their random selection. For example, with this set of data, it is clear that the number of clusters is 3, and we are even able to immediately identify these groups:



The problem is that the points are chosen at random and the final outcome may look like this:



Thirdly, the k-means algorithm is particularly sensitive to outliers and may give unexpected results if there are large deviations from the mean in the data. The k-means algorithm is often used in data analysis, such as market segmentation, data cluster analysis, or dimensionality reduction. It can also be used to find similarities between data and classify them into appropriate groups.

There is an *improved* version of this algorithm, known as *k-means++*, which eliminates the second problem of the *inferior* version. The first point is still chosen randomly, but the second point must be chosen at a location of an existing point that is farthest from the first point. Subsequent points are also chosen based on their distances from the already selected centroids.