Nonparametric Models for Machine Learning

Julian Heines
Supervisor: Haibin Zhao

Karlsruher Institut für Technologie, 76131 Karlsruhe, Germany
`ukvrb@student.kit.edu`

Abstract. This article provides an overview of nonparametric models in machine learning. Nonparametric models offer flexibility in modeling complex relationships between inputs and outputs, but come with the trade-off of increased computational cost and a potential for over fitting. The article summarizes various nonparametric methods such as K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Decision Trees, Kernel Methods and Nonparametric Neural Networks. The article concludes with a discussion on how to choose an appropriate nonparametric model for a given task and provides examples of applications where these methods have been successfully used.

Keywords: Nonparametric, Machine Learning, Advantages of Nonparametric Models, Disadvantages of Nonparametric Models, K Nearest Neighbor, Decision Trees, Support Vector Machines, Kernelmethods, Gausian Process, Nonparametric Neural Network, Choosing a Nonparametric Model

1 Introduction

An introduction to nonparametric models in machine learning is given. The second section discusses the advantages and disadvantages of nonparametric models. Therefore, their range of application is different from that of parametric models. There are different models for nonparametric learning. The third section deals with KNN, decision trees and random forests, SVM, kernel methods, and neural networks. Choosing the right model is difficult, so the fourth section is all about selecting nonparametric models for your use case. In the last section, a conclusion about nonparametric models will be drawn.

1.1 Machine Learning

Machine learning is often used to describe a computer that is performing a task that resembles human learning. As stated in [16], a machine learning model can be divided into 4 main components. Firstly, machine learning requires an environment that provides the data for learning. The next step is the learning itself, where some kind of process is applied. This process takes in data from the environment and applies some kind of algorithm to it. It then generates a model that represents the acquired knowledge. The repository is the component where the acquired knowledge is stored. The repository can be accessed by the model for the step called execution, where the acquired knowledge is applied and tested. Some algorithms can continue to learn from the execution. [4]

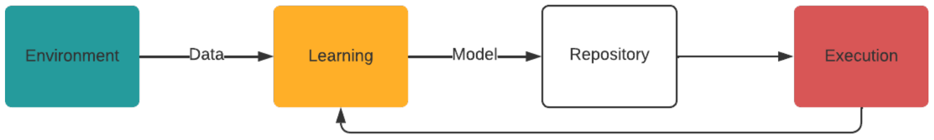

Fig. 1. A general machine learning model can be spitted into 4 main components.

1.2 Definition of "nonparametric"

Referring to the Model used in 1.1 we can derive different strategies. One question we can ask is what to store in our repository. In parametric models for machine learning, at the start, a function is chosen, which should model the given problem. The data is then used to tune the parameters of this function. Concluding the repository stores the function and its calculated parameters. Nonparametric models don't use a preset function for their training methods. These models calculate the function for a new data point by using the training data. Therefore, the repository holds the training data or an accumulation of it.

2 Motivation

This section describes the Advantages and Disadvantages of Nonparametric Machine Learning. It is also a comparison to the parametric methods because most of the advantages of Nonparametric models are disadvantages of parametric models.

2.1 Advantages

Let $y(x)$ be a correct output for an input x. Nonparametric methods are used for directly predicting $y(x)$ directly from the observed data without making assumptions about the form of the true model $g(x)$. A parametric function, on the other hand, always approximates $g(x)$ with some $g'(x)$. One advantage of nonparametric models is that we don't make presumptions about the true model. This allows for changing hyperparameters like the number of inputs or outputs of the model while processing data. It also prohibits us from making mistakes because of fewer adjustable parameters. Due to fewer restrictions, nonparametric methods' results can be reproduced in similar quality.[9] As mentioned, some of the models can change the number of output parameters. These Nonparametric models are used for solving Open World Assumption Problems.

Open World Assumption A Parametric Classifier often founds on the Closed World Assumption. It means that only a few predefined Classes of Objects can appear. This holds for many applications. Sometimes, we can't assume that we know every class beforehand. Especially in dynamic environments, it can be quite often that new classes appear, split from other classes, or disappear. Assuming that we don't know every class in our system is named Open World Assumption. A classifier for Open World Problems needs to take into account that he might not know each class yet. New objects that don't fit in any of the classes must be classified as a new class. This is called "unseen class discovery". [13]

2.2 Disadvantages

One disadvantage of nonparametric models is large computation time. The use of data for each computation causes the computation time to be directly related to the amount of data collected. As the volume of data increases, the computation time will also increase. The data must also be retained during the computation process. In contrast, parametric methods can discard the data after training the model just once, as they do not require the data for subsequent computations. Concluding, nonparametric models need more memory and computational resources. They also require a context that provides enough data to train the model. [2] Another disadvantage of nonparametric models is that they can be sensitive to noise in the data. Espaccialy instance based models, that use the training data directly for prediction, any noise present in the data can affect the predictions. This can lead to overfitting, where the model becomes too complex and performs poorly on new data. [1] The interpretation of nonparametric models can be very difficult. They dont have parameters witch can be used to understand the behavior of the model. [12]

3 Models for Non-Parametric learning

3.1 K Nearest Neighbor (KNN)

K-Nearest Neighbors (KNN) is a nonparametric method used for classification and regression. It is an instance-based learning method, meaning that the model stores instances of the training data. The algorithm works by storing all available cases and classifying new cases based on a similarity measure, such as distance functions. The new object is then classified by a majority vote of its k nearest neighbors, with k being a positive integer, typically small. By choosing k = 1, the new object is classified as its nearest neighbor.

KNN can be applied in both supervised and unsupervised learning. In supervised learning, the training data consists of labeled points, while in unsupervised learning, the training data consists of unlabeled points. In either case, the algorithm finds the k nearest neighbors of a new point, and uses the labels of those points to make a prediction for the label of the new point.

One advantage of KNN is that it is simple to implement and understand. It also has few hyperparameters, making it relatively insensitive to the choice of parameters. However, KNN can be computationally expensive as it requires storing all of the training data and calculating the distance between the new point and each stored point at prediction time. KNN can also be sensitive to the scale of the data and to the presence of irrelevant features. [3]

disance metrics Computing the "semantic distance" or similarity between a pair of points is important for nearest neighbor classification. The Mahalanobis distance is a common choice of distance metric when the input space is $\mathbb{R}^n$. It is calculated using the formula:

$$d_M(x, x_0) = (x - x_0)^T M (x - x_0)$$

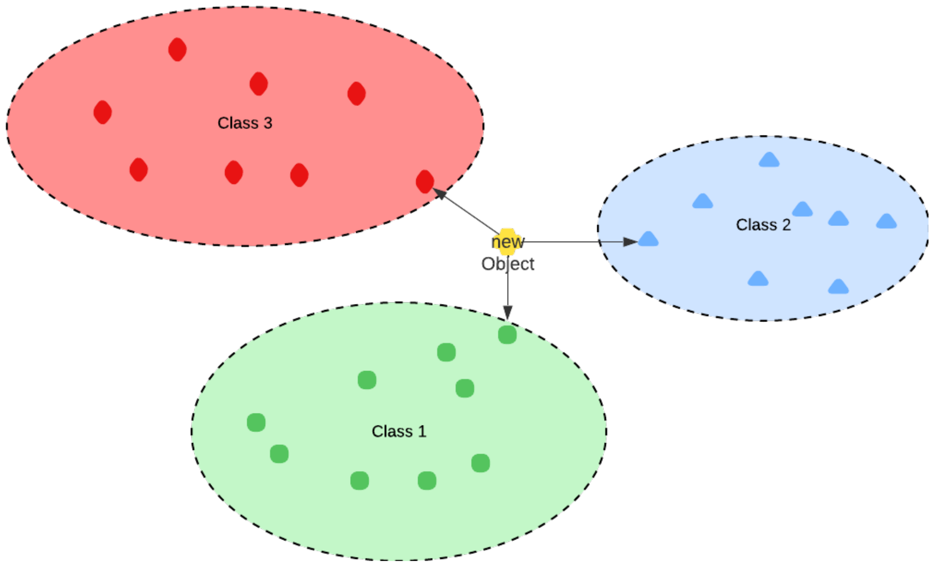

Fig. 2. A visualisation of a new object in 3NN. The object gets compared to the 3 nearest neighbors.

This distance metric can be learned using a method called Large Margin Nearest Neighbor (LMNN). The objective of LMNN is to minimize the distance between each point i and its target neighbors j that have the same class label, while also maximizing the margin between examples with incorrect labels. This can be achieved by minimizing the following function:

$$L(M) = (1-\alpha)\sum_{i=1}^{N}\sum_{j\in N_i} d_M(x_i,x_j)^2 + \alpha\sum_{i=1}^{N}\sum_{j\in N_i}\sum_{l=1}^{N}[y_i \neq y_l][m + d_M(x_i,x_j)^2 - d_M(x_i,x_l)^2]_+$$

Where N is the number of samples, N_i is the set of target neighbors of x_i, α and m are hyperparameters, and $[.]_+$ is the hinge loss function. The overall function is a convex function, which can be minimized using semidefinite programming or gradient-based optimization methods.[10] [5]

3.2 Decision Trees

A Decision Tree is a method mostly used for classification. The model hereby is represented by a tree. We can restrict Decision Trees to Binary Decision Trees because any ordered tree can be converted in a equivalent binary tree. The leaves of the tree are the classes of the classification. Each inner node represents a decision between two groups of classes. [11] [8]

Regression trees Regression trees are a type of decision tree that can be used for real-valued inputs. They consist of a set of nested decision rules, where at each node, the feature dimension of the input vector is compared to a threshold

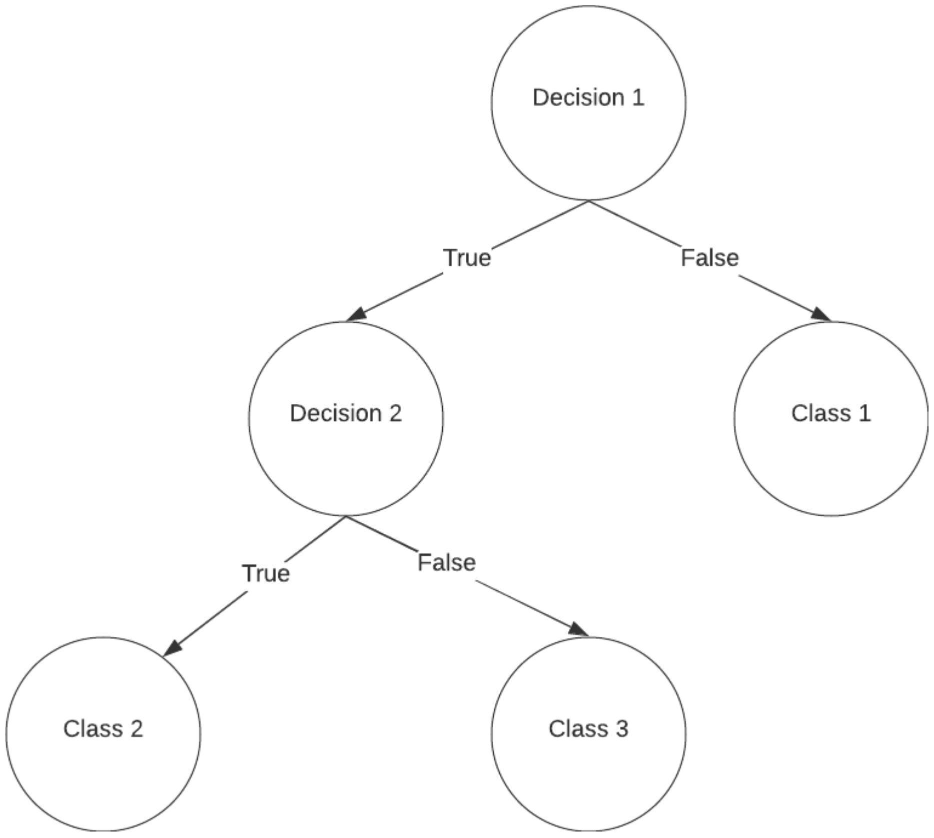

Fig. 3. A visualisation of a binary desision Tree.

value, and the input is passed down to the left or right branch based on whether it is above or below the threshold. The leaves of the tree specify the predicted output for any input that falls into that part of the input space.

The input space is partitioned into different regions, and a mean response is associated with each of these regions, resulting in a piecewise constant approximation of the input-output relationship. Formally, a regression tree can be defined as:

$$f(x;\Theta) = \sum_{j=1}^{J} w_j I(x \in R_j)$$

where R_j is the region specified by the j-th leaf node, w_j is the predicted output for that node, and $\Theta = (R_j, w_j) : j = 1 : J$, where J is the number of leaf nodes. The regions themselves are defined by the feature dimensions that are used in each split and the corresponding thresholds, on the path from the root to the leaf.

It is worth noting that by using enough splits, we can make a piecewise linear approximation to decision boundaries with more complex shapes. However, it may require a lot of data to fit such a model.

For classification problems, the leaves contain a distribution over class labels, rather than a single predicted output. The method for constructing such a tree is similar to that of a regression tree, but the splits are chosen based on reducing impurity measure of the leaf nodes. [7]

3.3 Support Vector Machines (SVMs)

The SVM is a nonparametric, instance-based machine learning method. It is used to approximate regression and construct multidimensional models of a function. The core of the SVM is a set of Vectors of the input space witch form the inputs of the nonlinear bias. The vectors are chosen so they represent the whole input space optimally. To calculate the model we than use a linear expansion of the bias.

Mathematicly the SVM involves finding a function that approximates a given set of training data by minimizing a specific type of functional. The functional has the form:

$$I(x, a, a^*) = \sum_{i=1}^{n} (a_i - a_i^*) K(x, x_i) + b$$

Where $x_i \in \mathbb{R}^n$, $y_i \in \mathbb{R}^l$, $I(x, w) = \sum_{i=1}^{n} w_i \phi_i(x)$, and (w, w) is the inner product of two vectors. The method then proceeds by assuming that only a few of the coefficients $(a_i - a_i^*)$ will deviate from zero and the corresponding vectors x_i are called support vectors (SVs). A quadratic optimization problem is then formulated and solved to find the coefficients a_i and a_i^*. The solution of this optimization problem involves maximizing the following function:

$$W(a^*, a) = -\frac{1}{\varepsilon} \sum_{i=1}^{n} (a_i + a_i^*) + \sum_{i=1}^{n} y_i (a_i - a_i^*) - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} (a_i - a_i^*)(a_j - a_j^*) K(x_i, x_j)$$

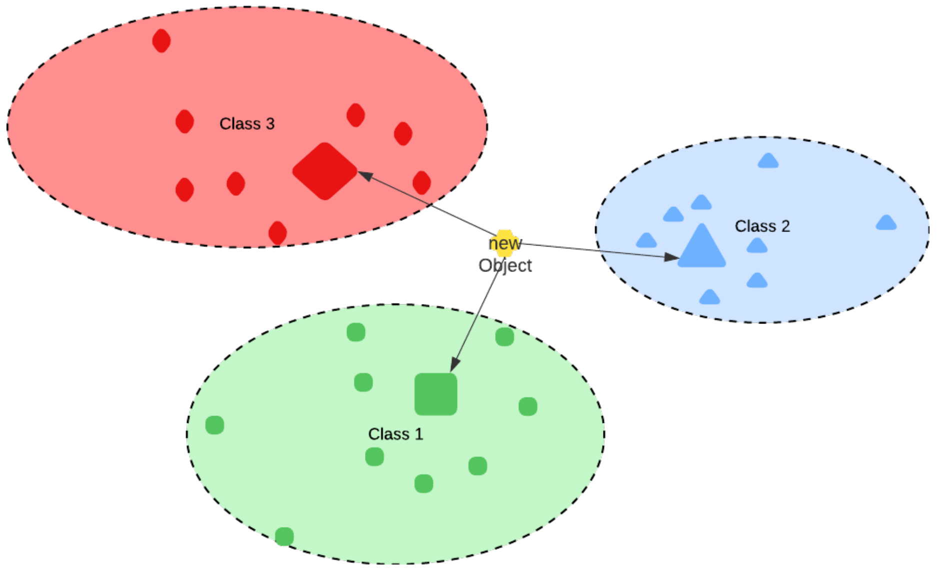

Fig. 4. A visualisation of a new object in SVM. The object gets compared to the 3 support vectors.

with constraints

$$\sum_{i=1}^{n}(a_i - a_i^*) = 0$$

,

$$0 \leq a_i, a_i^* \leq C$$

where C is a constant. It is importend to note that $K(x_i, x_j)$ is a Kernel function. The central point of the solution is that only some pairs $(a; -a_i)$ are non-zero. The corresponding vectors X_i are referred to as Support Vectors (SVs). [15]

3.4 Kernelmethods

One popular nonparametric method is kernel regression, which estimates $y(x)$ as a weighted average of the observed data, with weights determined by a kernel function K and its associated hyperparameters θ. The radial basis function kernel, with the isotropic Gaussian kernel as a commonly used example, is given by $K(x_i - x; \theta) = e^{-\theta||x_i - x||^2}$. Hyperparameters are often selected using cross-validation. Gaussian process (GP) models are similar to kernel regression models, but are based on a statistical framework that allows for the definition of a likelihood function that can be optimized to estimate the unknown hyperparameters without the need for cross-validation. [5]

Mercer Kernels A Mercer kernel, also called a positive definite kernel, is a symmetrical function that maps two input vectors, xi and xj, to a positive value and satisfies the property of having a positive definite Gram matrix. The Gram matrix is created by computing the dot product of every pair of vectors and its i-th row and j-th column holds the dot product of the i-th and j-th vectors. A matrix is considered positive definite if the dot product of any non-zero vector with itself is always positive, meaning all eigenvalues of a positive definite matrix are positive.

The most commonly used Mercer kernel is the squared exponential or RBF (Radial Basis Function) kernel, which measures the similarity between two vectors using the scaled Euclidean distance. It is defined as:

$$K(x, x_0) = \exp\left(-\frac{|x - x_0|^2}{2\sigma^2}\right)$$

where σ represents the length scale or bandwidth parameter. This kernel assigns high similarity between two vectors x and x0 when they are close in the input space and low similarity when they are far apart. [5]

Mercer Theorem Mercer's theorem states that any positive definite kernel can be represented as an inner product in a high-dimensional feature space. The kernel function $K(x_i, x_j)$ can be calculated by taking the inner product of some implicit feature vectors $\phi(x)$, which are defined through the eigenvectors of the kernel matrix. With this theorem, computations can be performed in an implicit feature space, rather than the original input space, and can be useful for working with non-linearly separable data. For instance, the quadratic kernel, $K(x, x_0) = (x, x_0)^2$, can be represented through Mercer's theorem as $K(x, x_0) = \phi(x)^T \phi(x_0)$, where $\phi(x)$ are the feature vectors.[5]

Examples of Mercer Kernels This passage describes some popular Mercer kernels. The first kernel is the RBF kernel, which is a stationary kernel, meaning its value only depends on the elementwise difference between the inputs. Another stationary kernel is the ARD kernel, which generalizes the RBF kernel by replacing Euclidean distance with Mahalanobis distance. Another kernel is the Matern kernel, which gives rise to rougher functions and can better model local "wiggles" in data. This kernel is parameterized by a parameter x, and the smoothness of functions generated by this kernel increase as x increases.[5]

Gausian Process Gaussian Processes (GPs) offer a approach for Bayesian non-parametric modeling of continuous functions. GPs define a distribution over functions that allows modeling of the function and its uncertainty. They are used in various applications including regression, classification, and time-series modeling.

A GP is specified by a mean function $m(x)$ and a positive definite covariance function (kernel) $K(x, x')$. The mean function sets the expected value of the function at any input, while the covariance function establishes the relationship between function values at different inputs. Together, the mean and covariance functions completely define the distribution of function values at any finite set of inputs.

One notable property of GPs is their closedness under conditioning. This means that if we know the function values at a set of inputs, we can use the mean and covariance functions to predict the distribution of function values at any other inputs, which is particularly useful in applications like regression and classification where the goal is to make predictions at new inputs.

Let's consider the example of noise-free observations. We have a training set $D = \{(x_i, y_i) : i = 1 : N\}$, where $y_i = f(x_i)$ is the noise-free observation of the function at input x_i. If we ask the GP to predict $f(x)$ at an input it has already seen, it should return $f(x)$ without uncertainty, serving as an interpolator of the training data. Now, consider a test set X of size ND, and we want to predict the function outputs $f = [f(x_1), ..., f(x_N)]$. By the definition of GP, the joint distribution $p(f_X, f|X, X)$ has the form:

$$p(f_X, f|X, X) = N(f_X|\mu_X, K_X, X)N(f|\mu, K_T X, K)$$

where $N(f_X|\mu_X, K_X, X)$ represents the multivariate Gaussian distribution of function values f_X at training inputs X with mean vector μ_X and covariance matrix K_X, X, and $N(f|\mu, K_T X, , K)$ is the multivariate Gaussian distribution of function values f at test inputs X with mean vector μ and covariance matrix $K_T X, K$. The matrices K_X, X, $K_T X, K$ are determined by the covariance function and describe the covariances between function values at different inputs. [5]

3.5 Nonparametric Neural Network

In machine learning, the goal is to identify the optimal hyperparameter θ that minimizes the criterion $c(\theta)$, such as validation error. Traditional methods like grid search, random search, and Bayesian optimization with Gaussian processes are popular, but they have limitations. Firstly, they require a full network training run to compute each c value, which can be time-consuming. Secondly, they don't use the opportunity to improve c further by adjusting θ during training.

The paper "Nonparametric Neuronal Networks" [6] introduces a new framework that solves these problems. The framework, called nonparametric neural networks, allows for dynamic adjustment of the network size during a single training run.

A parametric neural network is defined as a function $f(x)$ with a fixed number of layers, weight matrices of fixed dimension, and non-linear transformations. The optimization problem involves minimizing the error function and the regularizer over the weight matrices.

In a nonparametric neural network, the dimension of the weight matrices is unknown. The optimization problem now involves minimizing the error function and regularizer over both the dimension and weight matrices.

However, there's no guarantee that the optimization problem has a global minimum, and as the network becomes better with regards to the objective, it can become less desirable in practice. To avoid this, we use the fan-in and fan-out regularizers. These regularizers penalize either the incoming weights (fan-in) or the outgoing weights (fan-out) of each unit. This theorem forms the basis of the nonparametric network formulation.

4 Choosing a Nonparametric Model

If you only want to classify a few new data points and the input space is not high dimensional you can considder using KNN. This method is very simple and the output can be interpreted easily. For a few datapoints it can also be the fastest method. As [2] states one of the bigest problems for nonparametric learning are big input space as well as the amount of data. In this case SVM is a model you can choose. By representing the whole input space with a small set of vectors. This method is memory efficient and can calculate an output rather fast becaus only a smaller set of vectors needs to be taken an account. If you need a high interpretable classifier you can use a desission tree. By the structure of the tree it can be easily determined the models desisions. The nonparametric neuronal network approach can also be used for high interpetability because the model is still a neuronal network. Thus the outputted model can be interpreted by existing neuronal network frameworks.

4.1 Fields of Application

Nonparametric methods are used in different fields of application. I will only provide the general context of the application and link papers for specific research.

EEG feature extraction Today in research of brains activities, electroencephalogram (EEG) signals can be used to monitor a patient. They can be measured via attatching electrodes to the head. The difficult part today is processing the signals the brain sends. Nonparametric methods, especially gaussian process is used to analyse these signals. [12]

ETC measurement ETC is a metric of porous dry media. It can be used to model the behavior of fluid-satured sandstone. It is used in the context of oil drilling. Modeling this substance with SVM is more accurate than modeling it with ordinary neuronal neworks. [9] [14]

References

1. Aha, D.W., Kibler, D., Albert, M.K.: Instance-based learning algorithms. Machine learning **6**, 37–66 (1991)
2. Al-Jarrah, O.Y., Yoo, P.D., Muhaidat, S., Karagiannidis, G.K., Taha, K.: Efficient machine learning for big data: A review. Big Data Research **2**(3), 87–89 (2015). `https://doi.org/https://doi.org/10.1016/j.bdr.2015.04.001`, `https://www.sciencedirect.com/science/article/pii/S2214579615000271`, big Data, Analytics, and High-Performance Computing
3. Guo, G., Wang, H., Bell, D., Bi, Y., Greer, K.: Knn model-based approach in classification. In: Meersman, R., Tari, Z., Schmidt, D.C. (eds.) On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE. pp. 986–996. Springer Berlin Heidelberg, Berlin, Heidelberg (2003)
4. Gupta, A., Aggarwal, A., Bareja, M.: A review on classification using machine learning. International Journal of Information Systems & Management Science **1**(1) (2018)
5. Murphy, K.P.: Probabilistic machine learning: an introduction. MIT press (2022)
6. Philipp, G., Carbonell, J.G.: Nonparametric neural networks. CoRR **abs/1712.05440** (2017), `http://arxiv.org/abs/1712.05440`
7. Quinlan, J.R.: Induction of decision trees. Machine learning **1**(1), 81–106 (1986)

8. Rokach, L., Maimon, O.: Decision trees. Data mining and knowledge discovery handbook pp. 165–192 (2005)
9. Rostami, A., Masoudi, M., Ghaderi-Ardakani, A., Arabloo, M., Amani, M.: Effective thermal conductivity modeling of sandstones: Svm framework analysis. International Journal of Thermophysics **37**, 1–15 (2016)
10. Roth, P.M., Hirzer, M., Köstinger, M., Beleznai, C., Bischof, H.: Mahalanobis distance learning for person re-identification. Person re-identification pp. 247–267 (2014)
11. Rounds, E.: A combined nonparametric approach to feature selection and binary decision tree design. Pattern Recognition **12**(5), 313–317 (1980)
12. Shiman, F., Safavi, S.H., Vaneghi, F.M., Oladazimi, M., Safari, M.J., Ibrahim, F.: Eeg feature extraction using parametric and non-parametric models. In: Proceedings of 2012 IEEE-EMBS International Conference on Biomedical and Health Informatics. pp. 66–70 (2012). `https://doi.org/10.1109/BHI.2012.6211507`
13. Shu, L., Xu, H., Liu, B.: Unseen class discovery in open-world classification. arXiv preprint arXiv:1801.05609 (2018)
14. Vaferi, B., Gitifar, V., Darvishi, P., Mowla, D.: Modeling and analysis of effective thermal conductivity of sandstone at high pressure and temperature using optimal artificial neural networks. Journal of petroleum science and engineering **119**, 69–78 (2014)
15. Vapnik, V., Golowich, S., Smola, A.: Support vector method for function approximation, regression estimation and signal processing. In: Mozer, M., Jordan, M., Petsche, T. (eds.) Advances in Neural Information Processing Systems. vol. 9. MIT Press (1996), `https://proceedings.neurips.cc/paper/1996/file/4f284803bd0966cc24fa8683a34afc6e-Paper.pdf`
16. Wang, H., Ma, C., Zhou, L.: A brief review of machine learning and its application. In: 2009 International Conference on Information Engineering and Computer Science. pp. 1–4 (2009). `https://doi.org/10.1109/ICIECS.2009.5362936`