Intelligent Systems

Masters Degree in Mechanical Engineering

Group Project [EN]

Authors:

Francisco Pinto (ISTID 089888) francisco.v.pinto@tecnico.ulisboa.pt
Alexandre Gonçalves (ISTID 100121) alexandre.n.goncalves@tecnico.ulisboa.pt

Group 6

2024/2025 – 1st Quarter

Contents

1 Introduction

In medical applications, predicting heart attacks is crucial, requiring models that are both accurate and interpretable. Knowing this, the need to understand how models make predictions is required to build trust and ensure alignment with medical knowledge.

This project aims to compare different types of machine learning models, focusing on both performance and interpretabillity. We will use the "Heart Attack Analysis & Prediction Dataset" from Kaggle, which contains 303 records and 14 clinical features. The target variable is binary, indicating whether a patient is at risk of a heart attack.

2 Data Analysis

We start by processing the data for predicting heart disease using a dataset. The procedure includes exploratory data analysis, feature engineering, scaling, and feature selection. In table 1 we can find a list of the variables and their description.

Variable	Type	Description
Age	Numerical	Patient's age
Sex	Categorical (0/1)	Patient's sex
Cp	Categorical (0-3)	Degree of chest pain symptoms
Trestbps	Numerical	Resting blood pressure (mm Hg)
Chol	Numerical	Patient's cholesterol (mg/dl)
Fbs	Categorical (0/1)	Fasting blood sugar $\geq$ 120 mg/dL
Restecg	Categorical (0-2)	Resting ECG results
Thalach	Numerical	Maximum heart rate achieved
Exang	Categorical (0/1)	Exercise induced angina
Oldpeak	Numerical	ECG depression induced by exercise
Slope	Categorical (0-2)	Slope used during the exercise
Ca	Categorical (0-3)	Number of major vessels
Thal	Categorical (0-3)	Degree of thalassemia symptoms

Table 1: Variable Description

2.1 Histogram Functions

We create histograms for numeric and categorical variables to visualize their distributions. This visualization aids in understanding the underlying patterns in the data.

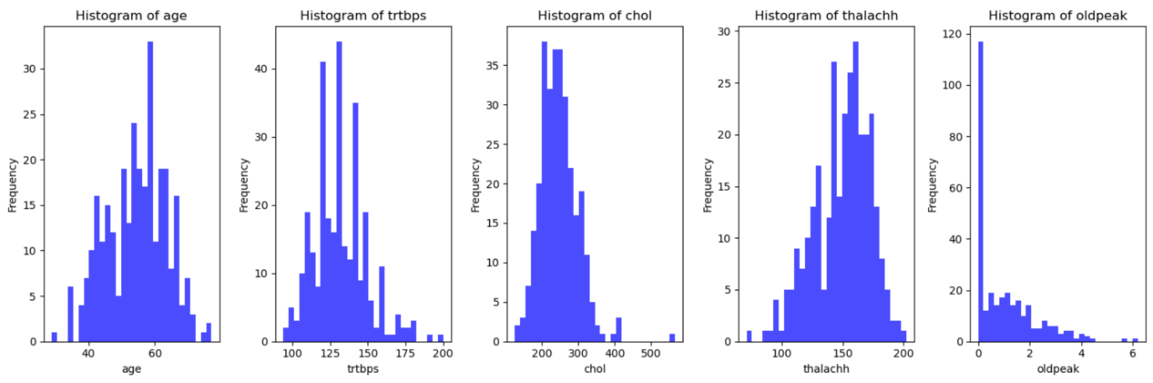


Figure 1: Histogram of Numeric variables

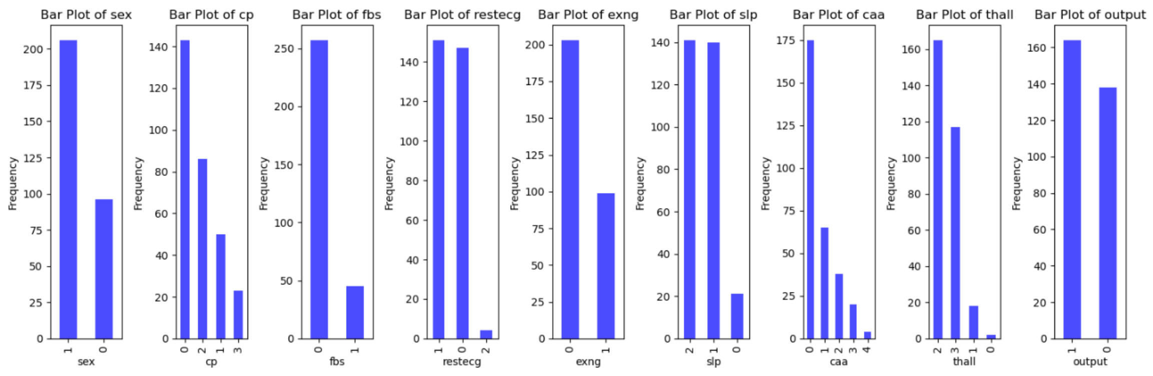


Figure 2: Histogram of Categorical variables

As we can see, numeric features display an asymmetric distribution, while the categorical ones are unbalanced.

3 Feature Engineering

In this stage, we separate the features from the target variable and split the data into training and test sets. This is essential for building and evaluating our model.

3.1 Outlier Functions

In our analysis, we utilize the Interquartile Range (IQR) method to identify and manage outliers. These outliers are then replaced with the corresponding limits to reduce their impact on the dataset.

This procedure enhances the reliability of our analysis by minimizing distortions caused by extreme values.

3.2 Target Balance

As we can see in Figure 3, although the classes are not balanced, there isn't much of a difference, so no alterations were made to the data in this section.

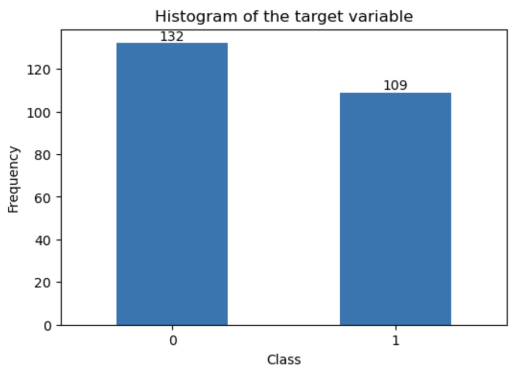


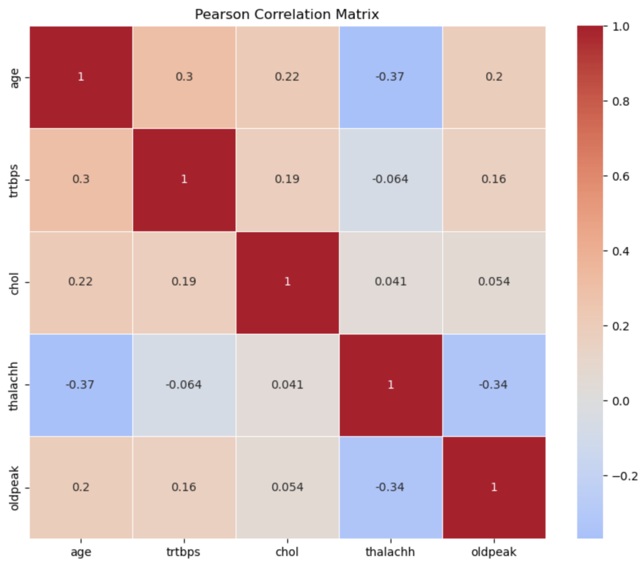
Figure 3: Target Balance

3.3 Feature Selection

In this section, we identify the most relevant features for predicting heart disease and remove the less important ones.

3.3.1 Correlation Matrix

Firstly, we analyzed the Pearson correlation matrix and, since there wasn't any pair of features with particular high linear correlation (Figure 4), no changes were made to the data.



	age	trtbps	chol	thalachh	oldpeak
age	1	0.3	0.22	-0.37	0.2
trtbps	0.3	1	0.19	-0.064	0.16
chol	0.22	0.19	1	0.041	0.054
thalachh	-0.37	-0.064	0.041	1	-0.34
oldpeak	0.2	0.16	0.054	-0.34	1

Figure 4: Pearson Correlation Matrix

3.3.2 Recursive Feature Elimination (RFE)

RFE is a feature selection technique that works by iteratively building a model and removing the weakest features to improve model performance. We create a logistic regression on the full set of features and then ranks the features by their importance, removing the least important one. This process is then repeated until the optimal number of features is identified.

The following method selected an optimal number of 12 features, removing only the age of each patient.

3.4 Scaling/Transformations

To standardize the dataset, we apply Standard Scaler, which transforms the feature values to have a mean of 0 and a standard deviation of 1. This ensures that all features contribute equally, what is extremely important when we are working with clustering.

The mentioned scaler was chosen since, as seen previously, some our features have asymmetrical distributions. It is also worth mentioning that the Robust Scaler was also taken into account, but we ended up not using it since we had already taken care of the outliers before.

3.5 Considerations

Before we start applying the data to different models, it is important to remember the context we are in, in order to identify which metrics/characteristics identify a good model.

Since we are trying to predict a health condition, our recall should be as high as possible, in order to minimize false negatives as much as possible. The number of false positives may increase, but this is often considered an acceptable trade-off given the stakes.

4 Machine Learning

On a first approach to this data set, we apply three different Machine Learning techniques that are commonly used in classification problems. These classifiers are described below. To each model, we try different combinations of parameters to find the one that outputs the best testing metrics. Since we have a relatively small dateset, to ensure robust evaluation and minimize the effect of data variability, we applied 5-fold cross-validation.

4.1 Evaluation Metrics

To allow for fair and thorough comparisons across models, each model's performance was assessed using several evaluation metrics:

- **Accuracy**: The proportion of correctly classified instances over the total instances.
- **Recall**: Measures the ability of the model to capture all positive cases.
- **Precision**: Indicates the percentage of correctly predicted positive cases out of all positive predictions.
- **F1-Score**: The harmonic mean of precision and recall, balancing both metrics.
- **Kappa Score**: A statistical measure of inter-rater agreement, reflecting how closely the predictions align with actual values beyond random chance.

4.2 Random Forest Classifier (RFC)

The Random Forest Classifier builds an set of decision trees during training. By aggregating the predictions of multiple trees, it enhances model accuracy and reduces the risk of overfitting. We tried different combination of `min_samples_split` and `min_samples_leaf` to see what outputs the best result.

- `min_samples_split`: This parameter determines the minimum number of samples required to split an internal node. We tested values of 2, 6 and 10.
- `min_samples_leaf`: This parameter specifies the minimum number of samples that must be present in a leaf node. We tested values of 1, 5 and 10.

4.3 XGBoost Classifier (XGB)

XGBoost is optimized for speed and high performance. It major ability is the capacity to handle complex patterns effectively while also incorporating regularization to prevent overfitting.

We tried different combinations of hyperparameters to find the best model configuration:

- `min_child_weight`: This parameter specifies the minimum amount of observations required to create a new node in the decision tree. We tested values such as 1, 5, and 10, which helps control overfitting by ensuring that the leaves have a sufficient number of observations.
- `max_depth`: This parameter determines the maximum depth of the trees. We explored values of 3, 5, and 7 to assess how tree depth influences model performance. A deeper tree can model more complex patterns but may also lead to overfitting.

4.4 Support Vector Machine (SVM)

SVM are primarily used for classification tasks. The main strength of SVM lies in its ability to find the optimal hyperplane that maximizes the margin between different classes in the feature space.

In our project, we tested the following hyperparameters:

- `C`: This parameter controls the trade-off between achieving a low training error and a low testing error. We tried values of 0.1, 1, and 10.
- `gamma`: This parameter defines how far the influence of a single training example reaches. We explored the parameters `'scale'` and `'auto'`.

4.5 Results of Model Comparison

The Random Forest Classifier achieved the best performance with `min_samples_split = 2` and `min_samples_leaf = 5`, while XGBoost excelled with `min_child_weight = 10` and `max_depth = 3`. The Support Vector Machine performed optimally with `C = 1` and `gamma = 'scale'`. The best results in the comparison are indicated in **bold** for clarity.

Table 2: Performance Metrics Comparison for Machine Learning Models

Model	Accuracy	Recall	Precision	F1-Score	Kappa
RFC	**0.851**	**0.851**	**0.854**	**0.849**	**0.688**
XGB	0.834	0.834	0.836	0.834	0.658
SVM	0.805	0.805	0.811	0.805	0.601

As shown in Table 2, the Random Forest Classifier (RFC) demonstrates superior performance compared to the other models, achieving the highest F1 score of 0.849, which indicates a strong balance of precision and recall.

5 Fuzzy Inference System

In an attempt to attain a more explainable model than the ones previously seen, we will now tackle the problem with a Sugeno-type FIS.

The process of building said model will start off with fuzzy clustering, then, we will estimate the antecedents by approximating the membership functions and, later on, the consequents. Since this model is meant to be interpretable and the data it will be using has been previously scaled, we will finalize by reverting the applied transformation, in order to make our model explainable with the values medical staff are used to working with and can comprehend.

5.1 Fuzzy Clustering

For each one of the clustering methods used, different numbers of clusters were tried and compared using a quality metric, in order to determine the optimal cluster structure. In the end, it was determined that the optimal number of clusters was in fact two.

In the beginning, the fuzzy c-means method was used this method led to some strange results. After some interpretation, we understood that the high and almost constant membership functions, paired with the close centroids meant that the clusters were in fact "competing" with each other for the same region on the data space, which is not ideal. After some thought and consideration, we reached the conclusion that this could be happening due to the high dimensionality of the data or simply because the cluster shape wasn't suitable to the fuzzy c-means method.

In order to solve this problem, the Fuzzy Semantic Twin Particle Swarm Optimization (FST-PSO) clustering method was used, since it is known for its better performance, specially in high-dimensional or complex datasets. After its application, the referred method solved the previous problem for some of the membership functions and allowed us to obtain the following parameters:

- **Cluster Centers**: The centroids representing the central tendency of each cluster.
- **Partition Matrix**: The partition matrix contains the degree of membership for each data point across clusters, with membership values ranging from 0 to 1.

5.2 Estimated Membership Functions

The membership functions for each input variable were derived based on the fuzzy clusters. These functions indicate the degree to which a particular input belongs to each fuzzy set and were approximated as Gaussian distributions.

5.3 Sugeno Fuzzy Inference System Parameters

Thanks to pyFUME, the following Sugeno-type FIS parameters were derived based on the previous results:

- **Consequent Parameters**: Coefficients defining the output mappings for each cluster, which were linear functions.
- **Fuzzy Rules**: Fuzzy rules generated from the cluster centers to perform predictions in the FIS.

5.4 Performance Evaluation

The performance of the fuzzy inference system was evaluated on a test dataset using the following metrics:

Accuracy	**Precision**	**Recall**	**F1-Score**	**Kappa Score**
0.787 / 1	0.828 / 1	0.750 / 1	0.787 / 1	0.575 / 1

Table 3: Fuzzy Model Evaluation Metrics

As can be seen, these metrics seem on-par with the other methods used.

5.5 Converting the Model to the Original Data Scale and Interpreting It

As previously mentioned, the data used was scaled during the feature engineering process. Unfortunately, up until now this meant that its values didn't hold any physical meaning. In order to correct this, both the membership functions were scaled in the following way:

$$\textbf{Gaussian Membership Functions}: \quad \sigma' = \frac{\sigma}{\sigma_{\text{data}}}, \quad \mu' = \mu\sigma_{data} + \mu_{data}$$

$$\textbf{Consequents}: \quad Y = aZ + c \Rightarrow Y = aX + \left(c - \frac{\mu_{data}}{\sigma_{\text{data}}}\right)$$

Even though this issue was mostly corrected with the change in clustering algorithm, some of the membership functions didn't really improve, such as the ones associated to chest pain (Figure 5). For this reason, only some of the variables are displayed (Figure 6). The fact that both the membership functions and the consequents are visible makes it easy to understand the features influencing its decision process.

	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	constant
0	-0.262711	0.097674	-0.002009	-0.000513	0.025580	0.143485	0.002075	-0.178311	-0.075183	0.046707	-0.107495	-0.080919	0.972633
1	-0.152544	0.070931	-0.003319	-0.000658	0.188784	-0.018731	-0.002600	-0.120633	-0.051297	0.163675	-0.111168	-0.252651	1.317023

Table 4: Table of Coefficients by Class

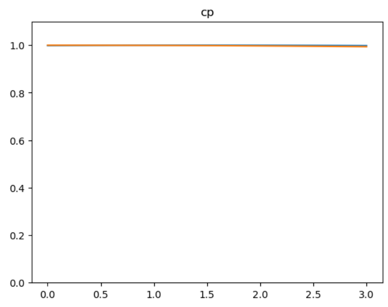
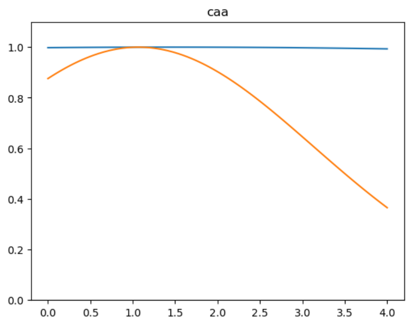
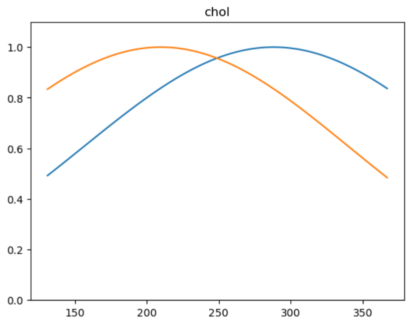


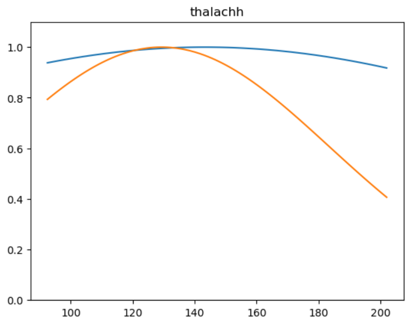
Figure 5: Chest Pain



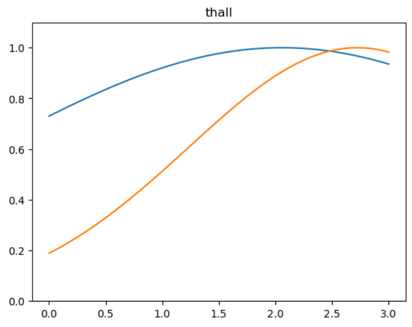
(a) Number of major vessels



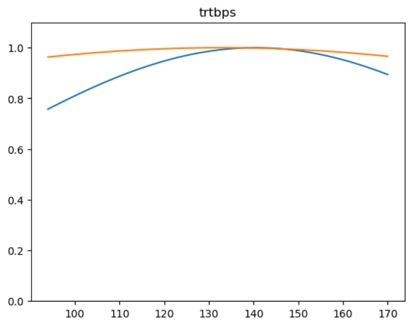
(b) Cholesterol



(c) Maximum heart rate



(d) Heart scan



(e) Resting blood pressure

Figure 6: Interpretable membership functions

6 Deep Learning

In an effort to create a more effective model than those previously used, we will explore various deep learning architectures. We will start by defining these architectures and training them using the Adam optimizer with a binary cross-entropy loss function.

The Adam optimizer adapts learning rates, speeding up convergence and providing stability across diverse tasks. Binary cross-entropy loss is ideal for binary classification, as it penalizes incorrect probabilities effectively, improving accuracy in distinguishing two classes.

All the output layers of the different models consists of a one neuron layer with a sigmoid activation function. The sigmoid function outputs values between 0 and 1, what is ideal for our classification problem.

6.1 Deep Neural Network (DNN)

The DNN consists of an input layer and then three hidden layers containing 64, 32, and 16 neurons. This structure allows the DNN to learn complex patterns and relationships in the data, which is crucial for our problem.

6.2 Long Short-Term Memory (LSTM)

Our LSTM starts with an input layer followed by a reshape layer to allow LSTM compatibility. The LSTM includes two layers, containing 64 and 32 neurons.

6.3 Convolutional Neural Network (CNN)

The CNN includes an input layer, followed by a reshape layer to allow Conv1D operations. This is followed by two convolutional layers, each succeeded by a max pooling layer, which helps in reducing dimensionality while retaining important features.

6.4 Multi-layer Perceptron (MLP)

Different MLP architectures were evaluated based on their hidden layer sizes, including configurations such as [5], [10], and [5, 5]. The MLPs consist of an input layer followed by one or more hidden layers, each with varying neuron counts. This flexibility allows MLPs to model complex relationships in the data effectively, depending on the architecture chosen.

6.5 Training and Evaluation

To assess the performance of the models on the training set, K-Fold cross-validation (with 5 splits) was employed. The results are in Table 5.

Table 5: Evaluation Metrics for Different Models

Model	Accuracy	Recall	Precision	F1-Score	Kappa
DNN	**0.838**	**0.908**	0.813	**0.857**	**0.665**
LSTM	0.718	0.823	0.723	0.760	0.429
CNN	0.672	0.707	0.686	0.693	0.319
MLP [5]	0.821	0.821	**0.837**	0.818	0.630
MLP [10]	0.776	0.776	0.781	0.774	0.537
MLP [5, 5]	0.764	0.764	0.769	0.763	0.515
MLP [5, 10]	0.772	0.772	0.776	0.771	0.529
MLP [10, 10]	0.760	0.760	0.767	0.758	0.505

The evaluation results indicate that the DNN outperformed the other models across most metrics, achieving the highest accuracy (0.838), recall (0.908), F1-Score (0.857), and Kappa (0.665). Its high recall underscores its effectiveness in identifying the positive class, what is extremely important when dealing with health cases. The high Kappa score suggests strong agreement with true labels.

The MLP models also performed well, particularly MLP[5], which achieved the second-highest accuracy (0.821) and F1-Score (0.818). Simpler MLP architectures outperformed more complex ones, indicating that additional layers or neurons may not enhance performance and could contribute to overfitting.

The CNN exhibited the lowest overall performance, with an accuracy of 0.672 and a Kappa of 0.319, implying that convolutional layers may not effectively capture the feature patterns necessary for this classification task.

In summary, the DNN demonstrated the best performance among the evaluated architectures, making it the most effective model for this dataset. The MLP[5] also showed competitive results, suggesting that simpler architectures can perform well.

7 Results

Table 6: Evaluation Metrics for Different Machine Learning Models

Model	Accuracy	Recall	Precision	F1-Score	Kappa
RFC	**0.851**	0.851	**0.854**	0.849	**0.688**
XGB	0.834	0.834	0.836	0.834	0.658
SVM	0.805	0.805	0.811	0.805	0.601
FIS	0.787	0.750	0.828	0.787	0.575
DNN	0.838	**0.908**	0.813	**0.857**	0.665
LSTM	0.718	0.823	0.723	0.760	0.429
CNN	0.672	0.707	0.686	0.693	0.319
MLP [5]	0.821	0.821	0.837	0.818	0.630
MLP [10]	0.776	0.776	0.781	0.774	0.537
MLP [5, 5]	0.764	0.764	0.769	0.763	0.515
MLP [5, 10]	0.772	0.772	0.776	0.771	0.529
MLP [10, 10]	0.760	0.760	0.767	0.758	0.505

After analyzing Table 6, we can draw several conclusions in the context of our project.

The RFC model demonstrates a robust overall performance, achieving the highest accuracy, precision and Kappa score, suggesting it is effective in delivering balanced predictions for heart attack risk. DNN, however, excels in recall and F1-Score (0.857), making it particularly well-suited for tasks focused on minimizing false negatives. High recall is crucial for heart attack prediction, as missing positive cases can lead to critical health risks.

We observe a precision-recall trade-off in the FIS model, which maintains relatively high precision but lower recall, indicating its potential use in scenarios where avoiding false positives is important, what is not ours.

The MLP configurations show that increasing model complexity does not consistently enhance performance. Simpler MLP structures, single-layer MLP [5], perform comparably or even better than more complex multi-layer configurations. This outcome suggests that simpler models may be sufficient for effective heart attack prediction in this dataset, potentially offering faster and more efficient deployment in real-world applications.

8 GitHub Link

You can find the code on GitHub at:
https://github.com/AlexGoncalves21/SInts_24

References

Chollet, F. (2017). *Deep learning with python* (2nd). Manning Publications.

Fuchs, C. E. M. (2022, March). *Pyfume* (Version 0.1.13) [Release Date: March 21, 2022]. https://github.com/CaroFuchs/pyFUME

Fuchs, C. E. M., Spolaor, S., Nobile, M. S., & Kaymak, U. (2020). Pyfume: A python package for fuzzy model estimation. *2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE).*

Hastie, T., Tibshirani, R., & Friedman, J. H. (2001). *The elements of statistical learning: Data mining, inference, and prediction.* Springer.

Jang, J.-S. R., Sun, C.-T., & Mizutani, E. (1997). *Neuro-fuzzy and soft computing: A computational approach to learning and machine intelligence.* Prentice Hall.