FEDERAL UNIVERSITY OF SANTA CATARINA
TECHNOLOGY CENTER
AUTOMATION AND SYSTEMS DEPARTMENT
UNDERGADUATE COURSE IN CONTROL AND AUTOMATION ENGINEERING

Paulo Hiroki Nishimoto

Analysis of machine learning methods and AutoML tools for indirect measurement of evaporation and condensation temperatures in variable speed compressors

Florianópolis
2020

Paulo Hiroki Nishimoto

Analysis of machine learning methods and AutoML tools for indirect measurement of evaporation and condensation temperatures in variable speed compressors

Final report of the subject DAS5511 (Course Final Project) as a Concluding Dissertation of the Undergaduate Course in Control and Automation Engineering at the Federal University of Santa Catarina in Florianópolis.
Supervisor: Prof. Rodolfo César Costa Flesch, Ph.D.
Co-supervisor:
Bernardo Barancelli Schwedersky, M.Sc;
Jonathan Krauß, M.Sc.

Florianópolis
2020

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Nishimoto, Paulo Hiroki
Analysis of machine learning methods and AutoML tools for indirect measurement of evaporation and condensation temperatures in variable speed compressors / Paulo Hiroki Nishimoto ; orientador, Rodolfo César Costa Flesch, coorientador, Jonathan Krauß, coorientador, Bernardo Barancelli Schwedersky, 2020.
69 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia de Controle e Automação, Florianópolis, 2020.

Inclui referências.

1. Engenharia de Controle e Automação. 2. Machine learning. 3. Automated machine learning. 4. Soft-sensing. 5. Refrigeration systems. I. Flesch, Rodolfo César Costa . II. Krauß, Jonathan . III. Schwedersky, Bernardo Barancelli IV. Universidade Federal de Santa Catarina. Graduação em Engenharia de Controle e Automação. V. Título.

Paulo Hiroki Nishimoto

Analysis of machine learning methods and AutoML tools for indirect measurement of evaporation and condensation temperatures in variable speed compressors

This dissertation was evaluated in the context of the subject DAS5511 (Course Final Project) and approved in its final form by the Undergaduate Course in Control and Automation Engineering

Florianópolis, October 26, 2020.

Prof. Hector Bessa Silveira, Dr.
Course Coordinator

Examining Board:

Prof. Rodolfo César Costa Flesch, Dr.
Advisor
UFSC/CTC/DAS

Prof. Eric Aislan Antonelo, Dr.
Evaluator
UFSC/CTC/DAS

Prof. Marcelo De Lellis Costa de Oliveira, Dr.
Board President
UFSC/CTC/DAS

This work is dedicated to my friends and my dear family.

ACKNOWLEDGEMENTS

Firstly to God, for blessing me with the knowledge, patience, and perseverance.

To my family and friends, for supporting me to keep doing this project and listening all my frustrations.

To my supervisor, professor Rodolfo Cesar Flesch, and the Ph.D. student Bernardo Barancelli Schwedersky for all the advice and feedback during this uncommon year that was 2020.

To the entire Fraunhofer IPT, especially Jonathan Krauß for giving me advice, suggestions, and guidance to improve my results in this project.

Finally, to all people that somehow supported me directly or in an indirect way, contributing to this work.

ABSTRACT

Refrigeration systems are present in various aspects of people's daily lives, either for thermal comfort, with the use of air conditioning, or for food conservation, with the use of freezers and refrigerators. With the high demand of the market, it is required that these products have more technology onboard to balance energy consumption with performance, and from the perspective of the industry, the investment in research and development becomes even more important to supply the market with high quality products that are aligned with low production cost. Considering this scenario, this work proposes a study of machine learning techniques and autoML tools combined with a model for estimating the rotation speed of variable speed refrigeration compressors, in a non-invasive way, in order to estimate evaporation and condensation temperatures. As the measurement of these quantities involves a high cost, so that it is only performed on test benches, it would be very beneficial to use indirect forms of measurement, turning the refrigerators into a more intelligent product. The results achieved by the machine learning models were promising, resulting in the estimation of the evaporation temperature with root mean square error of 1.38 °C, and for condensation temperature, 4.54 °C, highlighting the possibility of using machine learning in the context of estimating operating conditions of refrigeration systems with non-invasive methods.

Keywords: Machine learning. Automated machine learning. Soft-sensing. Refrigeration Systems

RESUMO

Os sistemas de refrigeração estão presentes em vários aspectos do cotidiano das pessoas, seja para conforto térmico, com o uso de condicionadores de ar, ou para a conservação de alimentos, com o uso de congeladores e refrigeradores. Com a alta demanda do mercado, é necessário que estes produtos tenham mais tecnologia embarcada para equilibrar o consumo de energia com o desempenho, e da perspectiva da indústria, o investimento em pesquisa e desenvolvimento torna-se ainda mais importante para abastecer o mercado com produtos de qualidade que estejam alinhados com o baixo custo de produção. Considerando este cenário, o presente trabalho propõe um estudo de técnicas de aprendizado de máquina e ferramentas de autoML combinadas a um modelo de estimação de velocidade de rotação de compressores de velocidade variável para refrigeração, de forma não invasiva, a fim de estimar as temperaturas de evaporação e condensação. Como a medição dessas grandezas envolve um alto custo, de forma que só é realizada em bancada de testes, seria muito vantajoso utilizar formas indiretas de medição, transformando os refrigeradores em produtos mais inteligentes. Os resultados alcançados pelos modelos de aprendizado de máquina foram promissores, resultando na estimação da temperatura de evaporação com raiz de erro médio quadrático de 1,38 $^{\circ}$C, e para temperatura de condensação, 4,54 $^{\circ}$C, evidenciando a possibilidade de se utilizar aprendizado de máquina no contexto de estimação de condições de operação de sistemas de refrigeração com métodos não invasivos.

Palavras-chave: Aprendizado de Máquina. Refrigeração. Sensoriamento Virtual.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

AC	Alternating current
ANNs	Artificial neural networks
AUTOML	Automatic machine learning
BLDC	Brushless direct current
BP	Back-propagation
CAGR	Compound annual growth rate
DC	Direct current
DT	Decision tree
ERT	Extremely randomized trees
GBM	Gradient boosting machines
GP	Genetic programming
HPO	Hyperparameter optmization
LIAE	Laboratory for instrumentation and automation of tests (from the Portuguese *Laboratório de instrumentação e automação de ensaios*)
MAF	Moving average filter
MIVIP	Indirect speed measurement by integral level (from the Portuguese *Medição indireta de velocidade pela integral do patamar*)
ML	Machine learning
MLP	Multi-layer perceptron
PCA	Principal components analysis
PWM	Pulse width modulation
RD	Research and development
ReLU	Rectified linear units
RF	Random forest
RMSE	Root mean squared value
SDK	Software development kit
SOM	Self organizing maps
XGB	Extreme gradient boosting

LIST OF SYMBOLS

F	Electromagnetic force
B	Magnetic field density
I	Current
E	Electromotive force
v	Speed of the conductor
θ_e	Electromagnetic position
θ_m	Mechanical position
p	Number of magnetic pole pair of the rotor
ω_m	Mechanical angular speed
ω_e	Electromagnetic angular speed
P_m	Mechanical power
T	Torque
r	Radius
b_k	Bias
u_k	Neuron
w	Weight of ANN
x	Input of ANN
y	Output of ANN
φ	Activation function of ANN
v_k	Activation potential of ANN
m	Number of epochs
e	Error
L	Inductance
R	Electrical resistance
E_f	Induced phase voltage
V_f	Phase voltage

CONTENTS

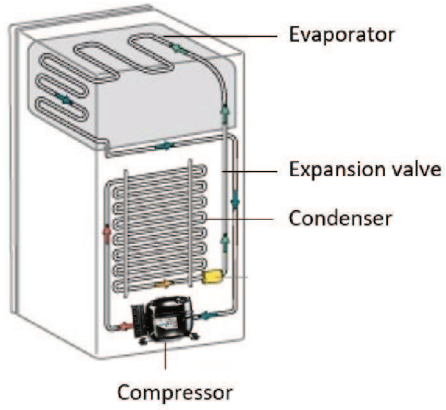
1 INTRODUCTION

Refrigeration systems are present in different ways in our lives and their importance is considerable [1]. They are present in the food and beverage sector, providing food preservation through the refrigerators and freezers [2], in the thermal comfort aspect, through air-conditioning systems, including apartment, houses, public and private ways of transportation, in industry, in many manufacturing processes, and also in the health context, preserving pharmaceuticals and medicines [3].

The refrigeration equipment market looks promising. According to [4], the global commercial refrigeration equipment market is expected to grow with a compound growth rate (CAGR) of 9% from 2019 to 2024. But, in another view, the energy consumption of the household and commercial appliances is something to be worried about. According to [5], the residential sector consumes 35% of the total power produced in the United States of America and the consumption of electricity for air-conditioning (cooling), freezers and refrigerators together represents 22% of it, according to [6]. The scenario in Brazil is similar. According to [7], the energy consumption by the residential and commercial sectors represented 26% and 17%, respectively, of all the country energy consumed in 2017 in the building sector. From these percentages, 32% and 33% of the energy, respectively, were destined for refrigeration systems.

To follow all the sales growth aligned with the environment care, the refrigeration industry invests in research and development (R&D) to improve the quality of the products, efficiency of the whole refrigeration system, thinking about sustainability and production cost aspects. The type of refrigeration system most used in practice is vapor compression, and this system is composed mainly by the compressor, the condenser, the expansion valve, and the evaporator, as shown in Figure 1.

Figure 1 – Main components in a refrigerator.



Source – Adapted from [8]

The compressor is the "heart" of the system, which acts as the pump that moves the refrigerant through the other components, performing the cooling process. To better extract the cooling capacity of the system and also relevant information, it is important to know its operating conditions, in this case, through the compressor inlet and outlet pressures, and what happens to the whole system with their variation, looking for the improvement of the whole process.

The evaporation and condensation temperatures are two of the main quantities that characterize the operating conditions of refrigeration systems. However, the measurement of these variables is typically performed by the refrigerant pressures (suction pressure - at the compressor inlet, and discharge pressure - at the compressor outlet) and their correlation with these temperatures [9]. Measuring these variables can bring valuable information about the system, but the cost of directly measuring these pressures makes it unfeasible for consumer end products.

The measurement of the suction and discharge pressures is commonly performed using pressure transducers [10], but this method is invasive, expensive, and can impact the durability of the product. The manufacture of a product with such measurements would only be feasible in a prototype character, not presenting itself as a definitive solution.

The indirect measurement of some interesting variables is possible through a soft-sensing approach, using several compressor measurements which are available and already used for its control. This is possible due to the fact that the refrigerant pressures, in the presence of friction and vibration, are largely responsible for the mechanical force to which the compressor is submitted, influencing the electrical quantities in the compressor supply, which are used for its speed control [11].

The adoption of learning techniques for the indirect measurement purposes is something that has been applied in the last years in the refrigeration industry, not only for estimation of operating conditions, like in [11] and [9], but also in other fields. References [12], [13] and [14] studied fault detection in refrigeration systems with neural network approaches. Prediction of the energy performance of a heating and cooling system was also proposed in [15]. Machine learning (ML) techniques were also used in [16] for the study of fault detection using ML ensemble methods, and in [17] for the estimation of sound power levels of refrigeration compressors.

It is noticeable the importance of non-invasive methods for measurements in refrigeration fields, which can be achieved with the use of learning techniques. The use of these techniques may imply the implementation of the process of estimating operating conditions in final products, which can make refrigerators become "smart" devices, improving performance and reducing energy consumption, among other benefits. Therefore, this work aims to study the implementation of different learning approaches, using ML algorithms as non-invasive methods, in order to validate their applicability in

the estimation of evaporation and condensation temperatures.

This project was developed in the Laboratory for Instrumentation and Automation of Tests (LIAE, from the Portuguese *Laboratório de Instrumentação e Automação de Ensaios*), at UFSC, in partnership with Nidec Global Appliance company, which is a global reference in technology for the household and commercial cold chain, counting on a broad, efficient, and competitive portfolio for household, food service, food retail, merchandisers, and special applications. The partnership between the laboratory and the company has yielded several studies in the area of refrigeration in recent years.

1.1 OBJECTIVES

The objective of this work is to evaluate different machine learning techniques to implement non-invasive methods for the evaporation and condensation temperatures estimation through measurements which can be done in an inverter. The machine learning techniques to be evaluated are the neural network, tree-based machine learning algorithms, and automatic ML (AutoML) approaches.

1.2 SPECIFIC OBJECTIVES

To achieve the general goal, some tasks were set to be managed. All the following topics were studied according to the necessity and not necessary in the presented order.

- Analyse the problem and search in the literature for applied solutions;
- Study the variable capacity refrigeration compressors that operate with brushless DC (BLDC) motors;
- Study and develop approaches of learning algorithms that are applicable to the problem;
- Search and implement automated ML tools, and study their particularities;
- Apply preprocessing techniques that can be valuable to improve the results;
- Structure and develop test setup configurations for all the learning tools;
- Analyse the obtained results and perform a benchmarking.

1.3 STRUCTURE OF THE DOCUMENT

The document is divided into 5 chapters. The theory, presented in chapter 2, covers the particularities of BLDC motors, mechanical apparatus used in the work, and characteristics of ML algorithms and AutoML tools are discussed. The implementation

of the proposed learning techniques, chapter 3, discuss how the data were preprocessed, how the model was built, and explains how the tests of the ML algorithms and AutoML tools were performed. In chapter 4, the results obtained with the traditional ML methods and AutoML tools are shown. Finally, the conclusions are exposed in chapter 5, where more critical analysis is made about the ML algorithms and AutoML tools used. In addition, are presented the future steps that can be taken into account for the proposed problem.

2 THEORETICAL BACKGROUND

This chapter covers the theoretical basis of the fields of knowledge used to carry out this work. Firstly, the characteristics of the compressor used are presented, followed by the machine learning tools considered in this study.

2.1 BLDC MOTORS

Electric machines are devices that can convert either mechanical energy to electrical energy or electrical energy to mechanical energy. In the first case, they are called generators and in the second case they are called motors [18]. There are a lot of different types of electric motors, but there are two main classifications: direct current (DC) motors and alternating current (AC) motors. This work was performed using DC motors.

Electric machines are made of magnetic materials or materials that have magnetic properties, and so do motors. Magnetic fields are the fundamental mechanism by which energy is converted from one form to another in motors. In permanent magnet DC motors, a current carrying wire in the presence of a magnetic field has a force induced on it, which is the basis of motor action. The magnitude of the force is defined as

$$F = BIL\sin(\theta), \tag{1}$$

where F is the magnitude of the electromagnetic force, B is the magnitude of the magnetic field density, I is the magnitude of the conductor current, L is the length of the conductor, and θ is the angular difference between B and I.

The movement of the conductor in the magnetic field induces an electromotive force (E), and it can be calculated as

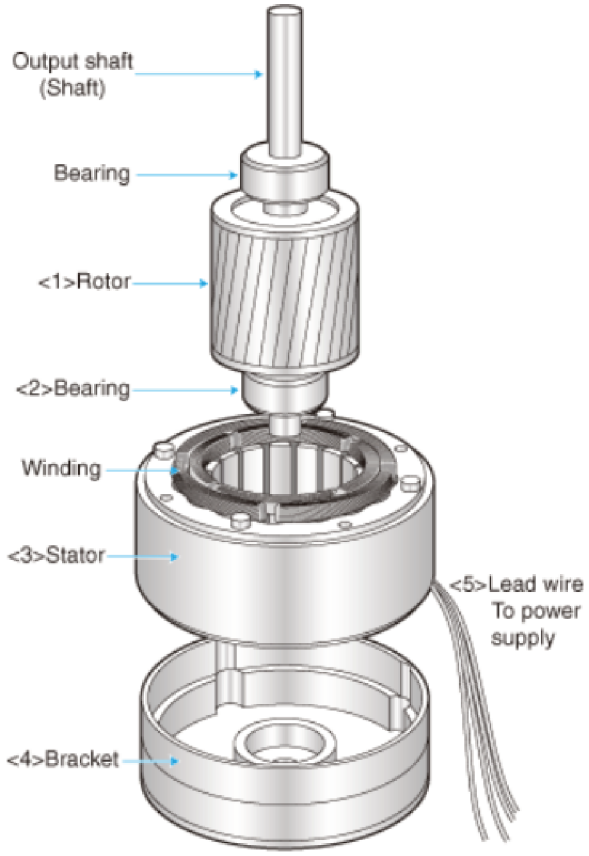
$$E = BLv, \tag{2}$$

where v is the speed of the conductor, B is the flux density of the magnetic field, and L is the length of the conductor.

According to [19], motors have two main parts: stator and rotor. The rotor is the moving part of the electric motor and it is usually located inside the stator. The rotor usually has conductors laid into it that carry currents, which interact with the magnetic field of the stator to generate the forces that turn the shaft, which delivers the mechanical power. The stator is the stationary part of the electric motor and usually consists of either windings or permanent magnets to generate the magnetic field necessary to rotate the rotor, shown in Figure 2.

In conventional DC motors, the rotor field is also generated by current through the coils wrapped around the rotor. This requires the use of a special tool to feed them

Figure 2 – Rotor and stator of electric motor.



Source – [20]

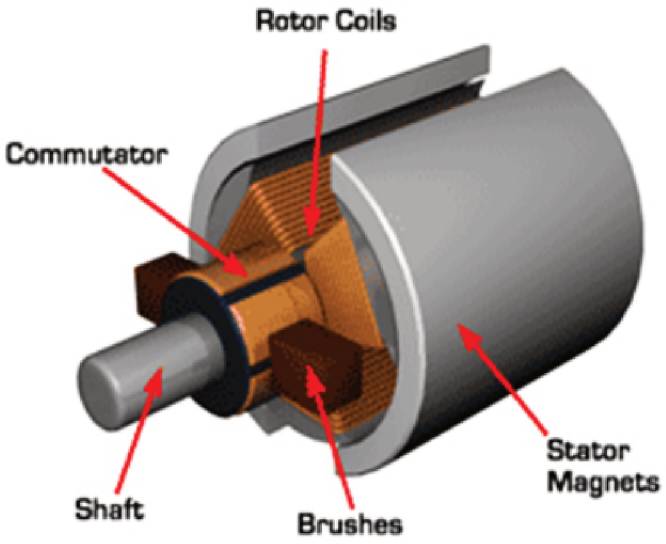
since the rotor will be rotating while the motor is in operation. This tool is called brush, represented in Figure 3, and it has a significant role in the engine operation.

In Brushless direct current (BLDC) motors, the electromagnetic field of the stator is generated by the current, like the permanent magnet DC motors, which passes through coils wrapped around it. The advantage is that the rotor is made of permanent magnets responsible for generating the other field, becoming unnecessary the use of brushes.

Another difference is that in a conventional DC motor, to keep it rotating, it is necessary to alternate the electromagnetic field in the rotor because the electromagnetic field is constant in the stator. To perform this task, a commutator, together with the brushes, are responsible to alternate this field. Instead of this method, in BLDC motors this switching process is done in the stator coils, with the use of an electronic inverter. A comparison of the characteristics of brushed and brushless DC motors is summarized in Table 1.

Figure 4 represents the BLDC motor studied in this work. It is a three-phase motor with two pairs of poles on the rotor, and the coils of the three phases (with

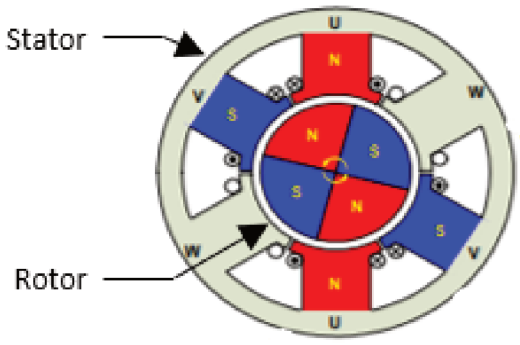
Figure 3 – Brushes in DC motor.



Source – [21]

impedances Z_u, Z_v and Z_w) are connected to each other in a star configuration, as shown in Figure 5, where N is the neutral point.

Figure 4 – Schematic representation of the BLDC motor used



Source – Adapted from [23]

In electric motors it is common to define two different position variables: mechanical position (θ_m) and electromagnetic position (θ_e) [19]. The expression that indicates the relation between them is given as:

$$\theta_e = p\theta_m\,, \tag{3}$$

where p is the number of magnetic pole pairs of the rotor.

Table 1 – Comparison between BLDC motor and brushed DC motor.

Feature	**BLDC motor**	**Brushed DC motor**	**Actual Advantage**
Commutation	Electronic commutation based on rotor position information	Mechanical brushes and commutator	Electronic switches replace the mechanical devices
Efficiency	High	Moderate	Voltage drop on electronic device is smaller than that on brushes
Thermal performance	Better	Poor	Only the armature windings generate heat, which is the stator and is connected to the outside case of the BLDC.;The case dissipates heat better than a rotor located inside of brushed DC motor.
Output Power/ Frame Size (Ratio)	High	Moderate/Low	Modern permanent magnet and no rotor losses.
Speed/Torque Characteristics	Flat	Moderately flat	No brush friction to reduce useful torque.
Dynamic Response	Flash	Slow	Lower rotor inertia because of permanent magnets.
Speed Range	High	Slow	No mechanical limitation imposed by brushes or commutator.
Electric Noise	Low	High	No arcs from brushes to generate noise, causing electromagnetic interference problems.
Lifetime	Long	Short	No brushes and commutator.

Source – [22],[23].

To visually understand these different positions, in Figure 4 it is possible to see that if the rotor rotates 180º mechanically, the magnetic poles will be in the original configuration.

It is possible to obtain the mechanical angular speed as the time derivative of the mechanical position, as shown in Equation (4). The electromagnetic angular speed (ω_e) is found using the same approach being given by:

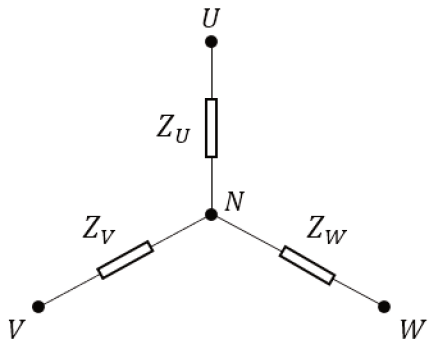
$$\omega_m = \frac{\mathrm{d}\theta_m}{\mathrm{d}t} . \tag{4}$$

2.1.1 BLDC motor control

Brushless DC motors use electric switches, performed by the electronic inverter, to perform current commutation, and thus continuously rotate the rotor. In this case, electric switches are connected in an H-bridge structure (inside the inverter) for a three-phase BLDC motor, as shown in Figure 6.

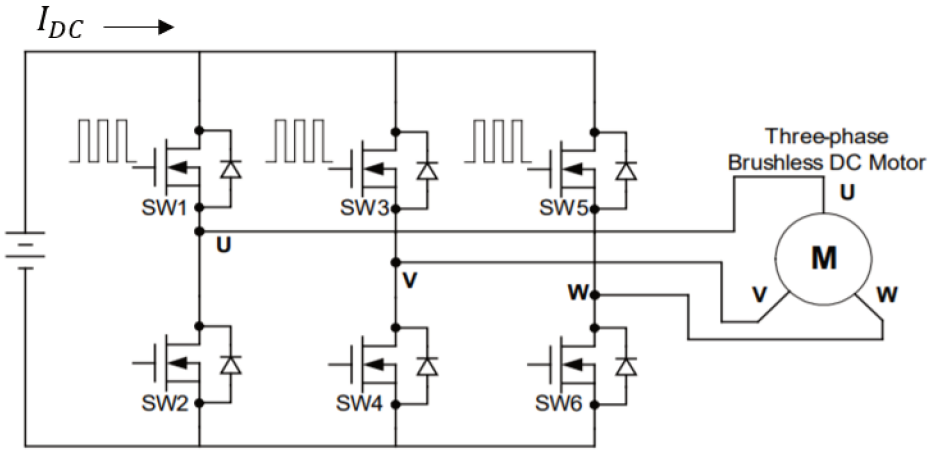
The switches are performed by transistors that switch the phases using Pulse

Figure 5 – Star connection



Source – Author

Figure 6 – Three-phase bridge



Source – Adapted from [23]

width modulation (PWM), which converts a DC voltage into a modulated voltage, thus, limiting the startup current, and controlling speed and torque.

The inverter converts the DC into Alternating current (AC) voltage and the transistors are driven in such a way that a rectangular current pulse coming from the bus (I_{DC}) gets into the motor through one phase and leaves through another, without it passing through the remaining phase, depending on the drive strategy [24].

Starting from the premise that

$$P_m = Fv\,, \tag{5}$$

where P_m is the power, F is the force and v is the the speed, and torque is presented

as

$$T = Fr\,, \tag{6}$$

where r is the radius. Isolating F and substituting in Equation (5), Equation (7) is obtained.

$$P_m = \frac{T}{r}v\,. \tag{7}$$

Substituting the linear speed (v) for ω, where ω is the angular speed (rad/s), the mechanical power delivered to the rotor of the BLDC motor, P_m, can be calculated as ([19], [24] and [25]):

$$P_m = T\omega = E_U I_U + E_V I_V + E_W I_W\,, \tag{8}$$

where U, V, W are the tree phases represented in Figure 6 and E is the induced voltage on each phase.

To maximize the efficiency of the motor the switches must be made at a specific time. They must be made in such a way that a rectangular current pulse between the phase with the highest positive induced voltage and another rectangular current pulse leaves the phase with the highest negative induced voltage. This detection can be done by using Hall effect sensors [24] or sensorless control techniques [26], [27].

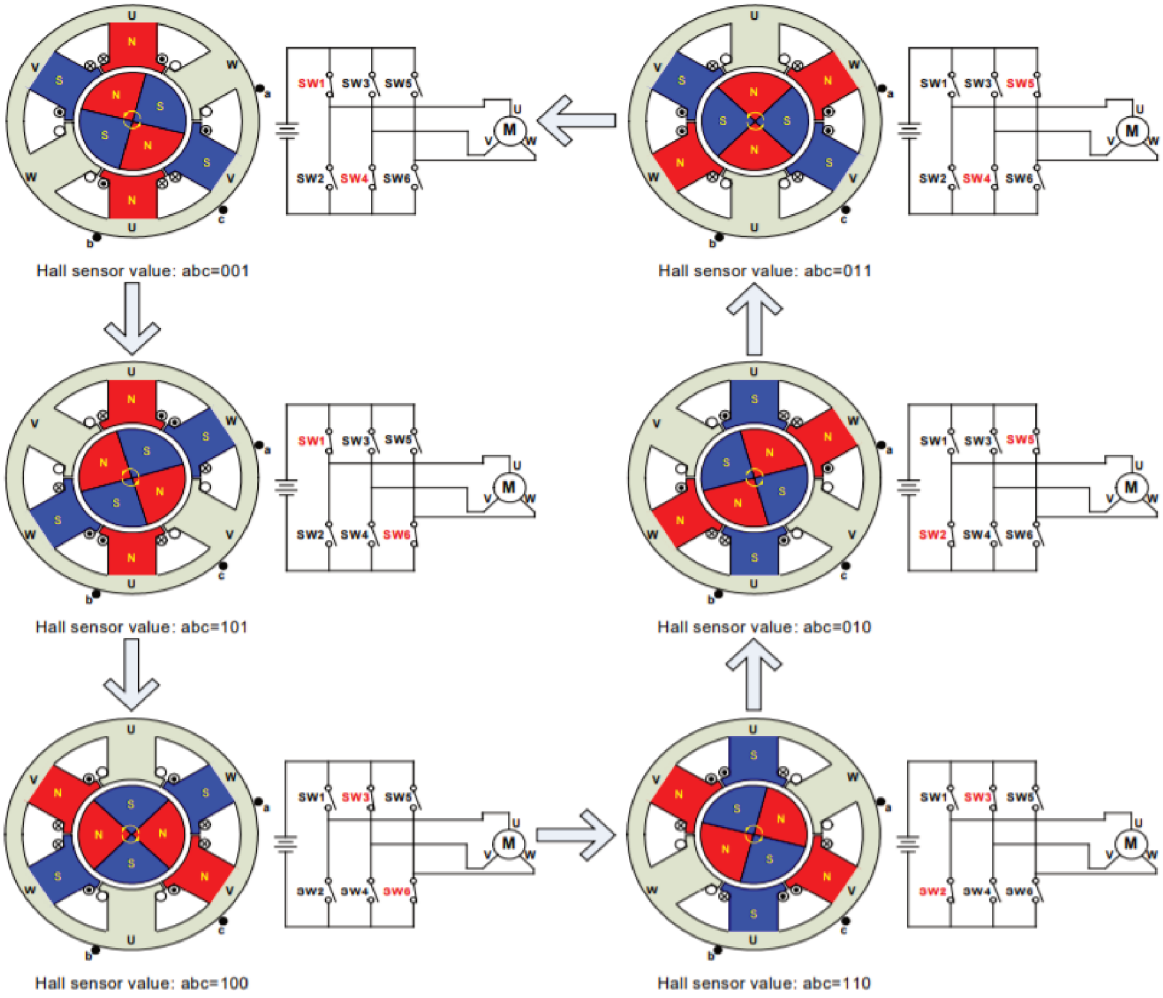
In order to simplify the explanation, the method based on the use of Hall sensors is used, but there are already more modern techniques that work without these types of sensors. A three-phase BLDC motor requires three Hall sensors to detect the rotor's position. Based on the physical position of the Hall sensors, there are two types of output: a 60° and a 120° phase shift. Using these three Hall sensor signals it is possible to determine the exact commutation sequence.

The commutation sequence can be visualized in Figure 7. The tree Hall sensors "a", "b", and "c" are positioned on the stator in 120° intervals, while the three-phase windings are in a star configuration.

In Figure 7 is possible to see that the Hall sensor state changes for every 60° rotation and the whole electrical cycle is completed after 6 steps. To be better explain how the direction of the current changes, Figure 8 shows a Hall sensor diagram that corresponds to the positions of the electromagnetic poles of the rotor in Figure 7.

To be easily understood, consider that every time that the "N" pole of the rotor is pointed to "a", "b", or "c" Hall sensor, it is set to 1. For example, starting with the upper left configuration of Figure 7, the "N" pole is pointed to "c", so, the Hall sensor value is abc = 001. This configuration is the start point of the Figure 8, where U phase is identified to "HIGH", V to "LOW" and W to "FLOAT", concluding that the current is flowing from U to V direction. This behavior continues to keep the rotor rotation. The "FLOAT" state is a transitory state, in which the phase that was recently turned off is discharged through the freewheel diodes that follow the transistors.

Figure 7 – Waveforms during the drive of a BLDC motor



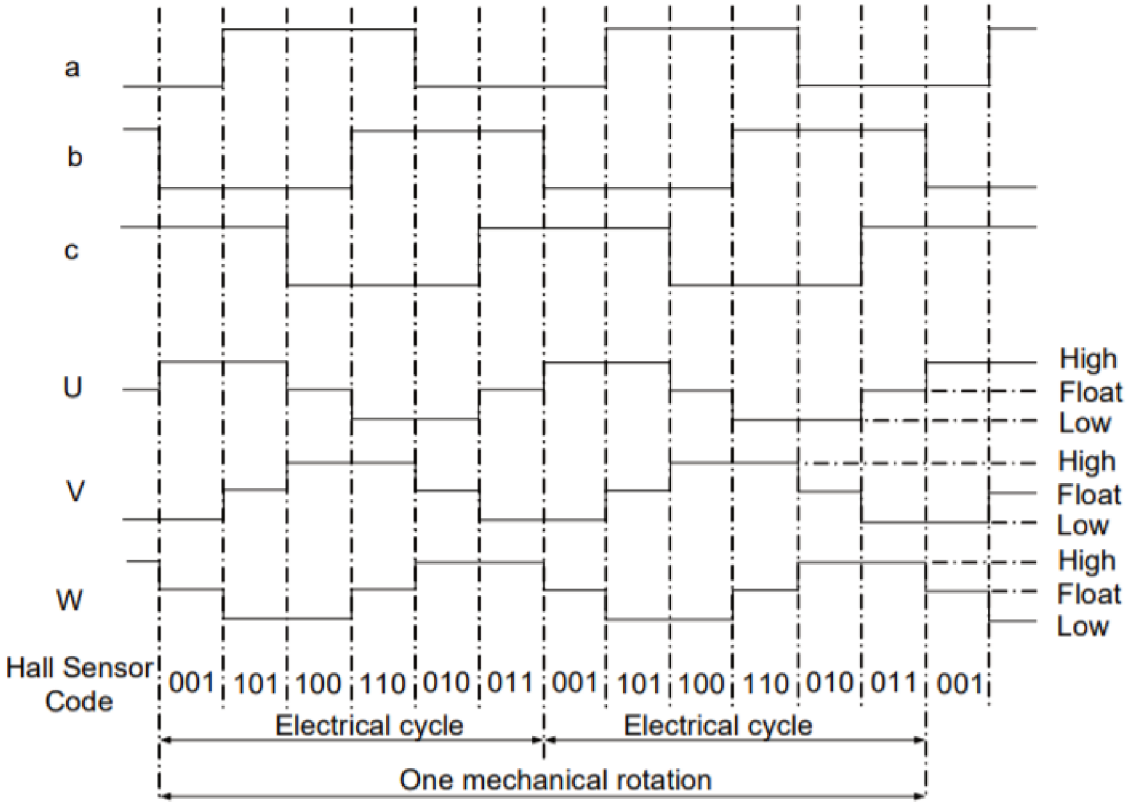
Source – [23]

2.2 ARTIFICIAL NEURAL NETWORKS

Artificial neural networks (ANNs) are computational modeling tools that have found application in many disciplines for modeling complex real-world problems [28]. They can be defined as structures comprised of simple processing elements called neurons, organized in an interconnected network, that are capable of performing massively parallel computational for data processing and knowledge [29].

The idea of ANN is not to replicate the biological human system but make use of what is known about the functionality of biological neural networks (NN) to solve complex problems [28]. Some characteristics of ANNs that are remarkable come from the similarity with biological NNs, and this is something that turns them so attractive, such as nonlinearity, high parallelism, robustness, fault and failure tolerance, learning, ability to handle imprecise and fuzzy information, and their capability to generalize [30].

Figure 8 – Three-phase BLDC motor sensor versus drive timing



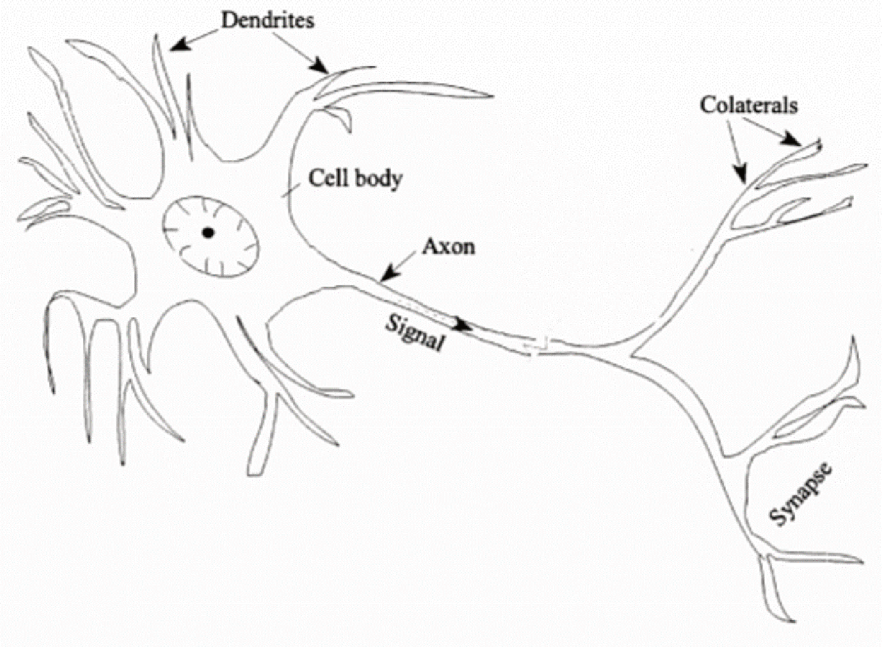
Source – [23]

Figure 9 represents the structure of a biological neuron, which is a basic building block of the human nervous system responsible for information processing. This basic unit is composed of 3 main parts: the dendrites, which are responsible for receiving the signals from the previous neuron and pass them over to the cell body; the body, responsible for the synthesis of all neuronal proteins; and the axon, which receives signals from the body and carries them away though the synapses (microscope gaps). The signal of information follows this sequence and passes it throw another neuron.

The artificial neurons are inspired by their biological counterparts, presenting a similar structure, which is presented in Figure 10. The analogy is that the connection between the nodes represents the axons and dendrites, and the connection weights represent the synapses.

In Figure 10 there are three basic elements: synaptic weights, which store much of the knowledge in the artificial neuron model, in which the weights are responsible for weighting the information that enters the neuron; summation, responsible for summing the input signals, weighted by the respective synaptic strengths of the neuron; and activation function, which maps the sum of the input data, weighted by synaptic weights,

Figure 9 – Schematic of a biological neuron



Source – Adapted from [28]

to the output value of the neuron, limiting the amplitude of this output.

The neuron can be also described in mathematical therms as:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \,, \tag{9}$$

$$y_k = \varphi(u_k + b_k) \,, \tag{10}$$

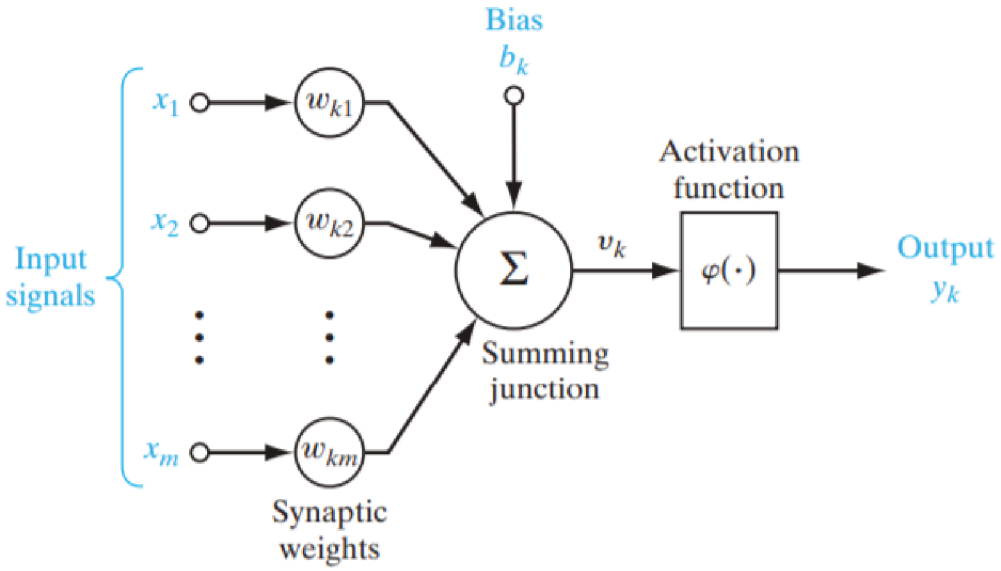
where $x_1, x_2, x_3 ... x_m$ are the input signals, $w_{k1}, w_{k2}, w_{k3} ... w_{km}$ are the synaptic weights of neuron k, u_k is the linear combiner output due to the input signals, b_k is the bias, $\varphi(\cdot)$ is the activation function, and y_k is the output signal of the neuron.

The purpose of the bias (b_k) presented in Figure 10 is to increase or decrease the net input of the activation function, depending on whether it is positive or negative, respectively. The use of bias b_k has the effect of applying an affine transformation to the output u_k, resulting in the activation potential

$$v_k = u_k + b_k \,, \tag{11}$$

where v_k is the activation potential. Depending on whether the bias b_k is positive or negative, the relationship between v_k of neuron k and the linear combiner output u_k is modified [31], as shown in Figure 11.

Figure 10 – Nonlinear model of a neuron k



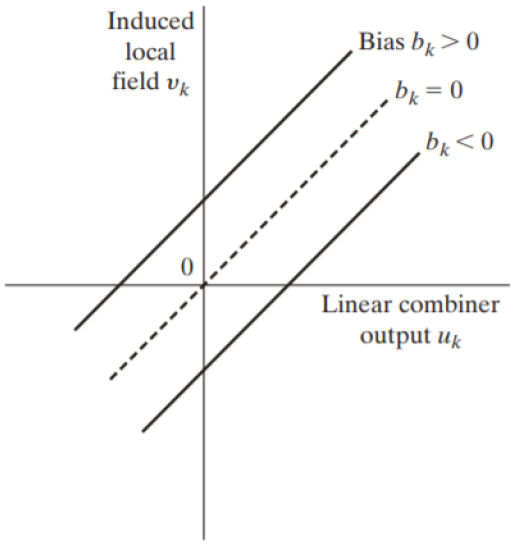
Source – [31]

So, combining Equation (10) and Equation (11), the final equation can be represented as

$$y_k = \varphi(v_k)\,. \tag{12}$$

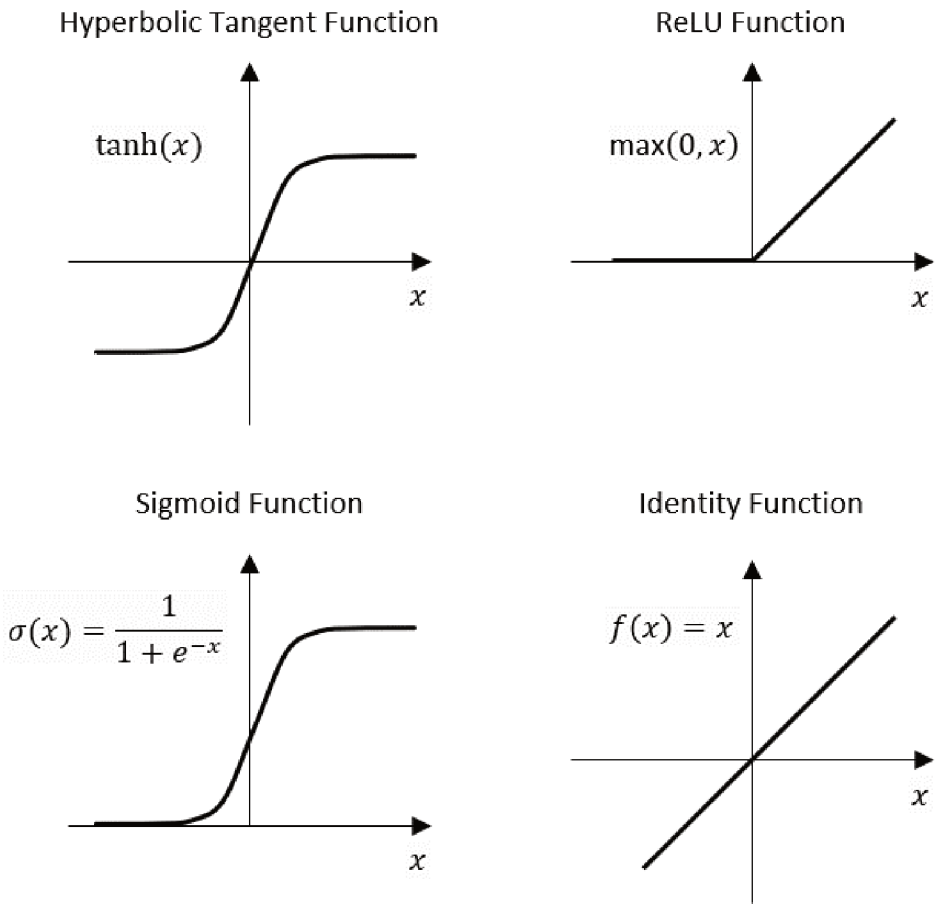
Several nonlinear functions can be chosen as activation function, with the main ones being sigmoid function, identity function, hyperbolic tangent function, rectified linear units (ReLu) function [32]. Figure 12 represents the main activation functions.

Figure 11 – Affine transformation produced by the presence of a bias



Source – [31]

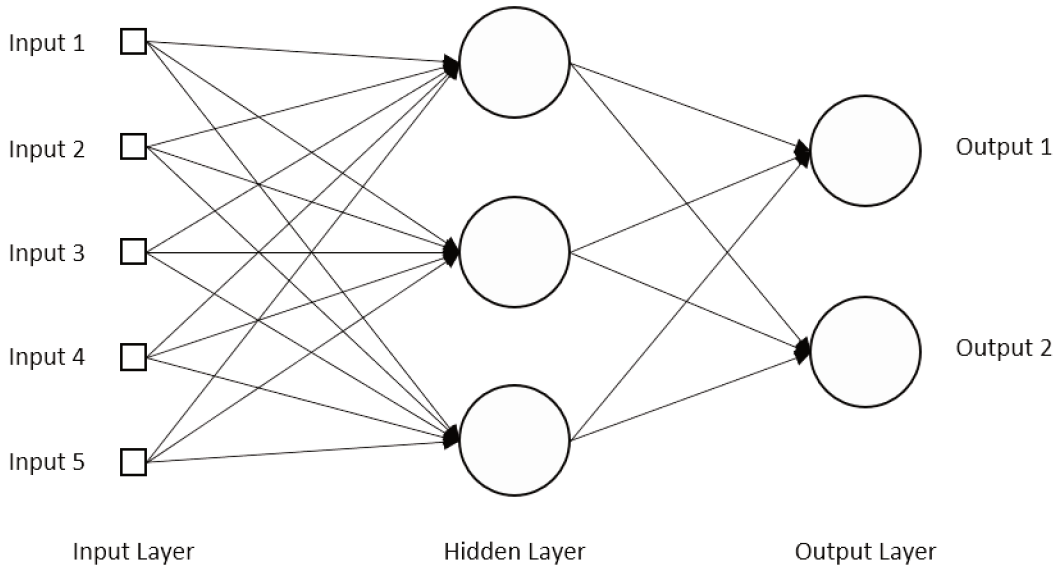
Figure 12 – Commonly used activation functions



A neural network is formed by several artificial neurons, which can be organized in different architectures [31]. The typical ANN architectures can be divided into two main categories, based on the information flow direction, which can be feedforward or recurrent. Feedforward networks are used to solve static problems, while recurrent NNs are used in problems where temporal context is needed. The main feedforward architectures are single-layer networks [34], multilayer perceptron (MLP) networks [31], radial basis function (RBF) networks [35], self organizing maps (SOM) [36], and convolutional networks [37].

For regression problems, the MLP network is a common choice, in which its structure is represented in Figure 13. This architecture presents at least one hidden layer, with the artificial neurons being fully connected. The hidden layers are formed by neurons with a nonlinear activation function, while the output layer has neurons with the identity function as activation function, in regression tasks. The number of neurons and hidden layers are the main hyperparameters in this architecture, influencing the learning ability of the algorithm.

Figure 13 – MLP structure



Source – Author

The learning ability of a NN occurs through an iterative process of adjustments applied to its synaptic weights and bias. Initially, the NN receives a stimulus from the environment, and through the process of weight adjustment, modifies its parameters. As a result, the network now responds in a new way to the environment, as a result of changes in its internal structure [31].

This internal modification in response to the environmental stimulus is obtained through training algorithms, which, through a pre-defined set of rules, including cost

functions, solve the learning problem. There are a lot of cost functions that can be used, but this work considered the mean squared error (MSE)

$$\lambda = \frac{1}{t}\sum_{j=1}^{t} e_j^2\,, \tag{13}$$

where λ is the MSE, t is the number of epochs (iterations) and e_j is the error (difference between the output generated by the current network and the expected output).

Tensorflow [38], which is an open-source machine learning library that supports a variety of applications, with a focus on training and inference on artificial neural networks, was used in this work.

2.3 TREE BASED METHODS

In this section, the tree-based methods used in this work are presented. These types of models use decision trees (DT) as a building block of the algorithm, which learns based on a set of if-then-else decision rules.

As the name suggests, this algorithm is structured in form of a tree and can be applied to classification and regression problems [39]. It works by breaking down a data set (input) into smaller subsets, while at the same time an associated DT is incrementally developed. In the end, a "tree" is formed by decision nodes and leaf nodes. The decision nodes have two or more branches, while leaf nodes represent a decision made. The topmost decision node in a tree corresponds to the best predictor, called root node, as detailed in Figure 14.

The selection of which input variable to use and the specific split or cut-point can be done using a learning algorithm or using a greedy algorithm, which minimizes a cost function. The greedy approach is a numerical procedure where all the values are lined up and different split points are tried and tested using a cost function. The split with the best cost (lowest cost) is selected.

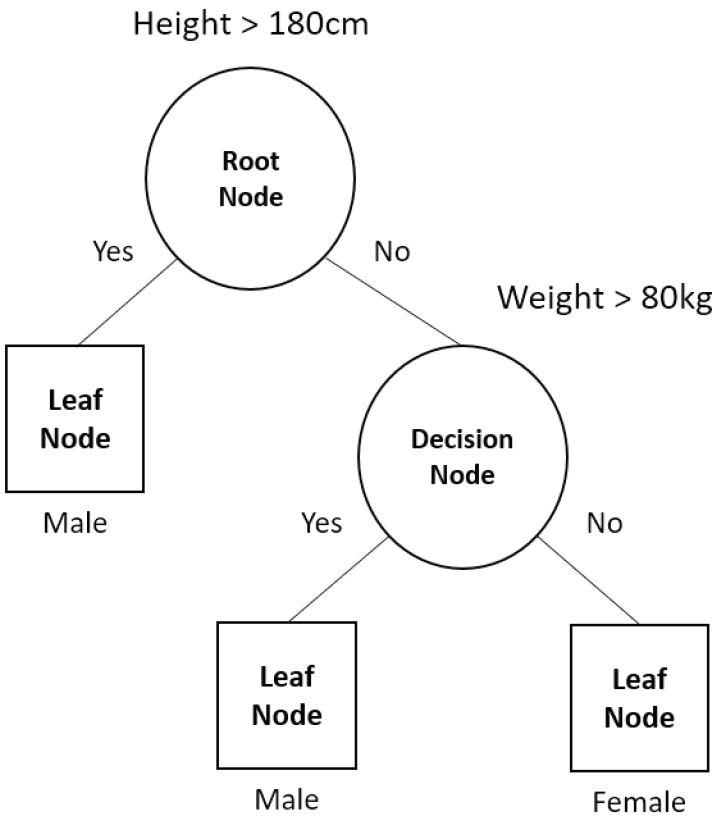
For regression problems, dealt in this work, the sum squared error is typically used as cost function, and it is expressed as

$$\lambda_{\text{DT}} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2\,, \tag{14}$$

where λ_{DT} is the MSE, N is the total number of iterations, y is the output and $\hat{y}$ is the predicted value for each iteration.

Generally, the tree construction ends using predefined stopping criteria. The most common stopping procedure is to use a minimum count on the number of training instances assigned to each leaf node [40]. If the count is less than some minimum then the split is not accepted and the node is taken as a final leaf node. The smaller

Figure 14 – Decision tree basic structure example



Source – Author

this number more specific the model for the training data is, and it tends to overfit. The bigger this number, more the model tends to be generalized and tends to underfit.

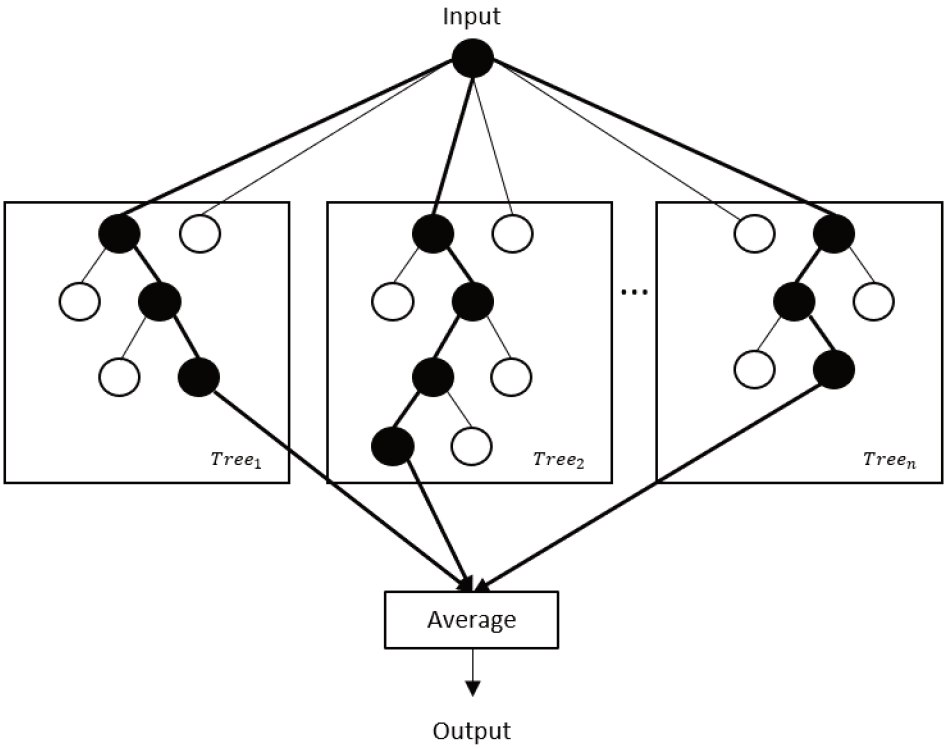
subsection 2.3.1 covers the random forest algorithm, which introduces new "features" to the DT algorithm. subsection 2.3.2 explains the extremely randomized tree with its particularities, and then, subsection 2.3.3 approaches the gradient boosting algorithm with its characteristics. All these algorithms are ensemble methods, which consist of a combination of algorithms, in this case, DTs, to achieve better performance.

2.3.1 Random Forest

A Random forest (RF) is an ensemble method that combines many DT and it includes two features in the learning process: random sampling of training data points when building trees and random subsets of features considered when splitting nodes. A simple representation of the structure of RF is shown in Figure 15.

The random sampling with replacements of training data is also known as bootstrapping, which means that some samples can be used multiple times in a single tree. The main idea is that by training each tree on a different data subset, overall, the entire

Figure 15 – Random forest bagging approach



Source – Author

forest will have lower variance but not at the cost of increasing the bias, although each tree might have high variance with respect to a particular set of the training data [41]. The high variance causes overfitting, and this phenomenon occurs when the model memorizes the training data by fitting it closely. The problem is that the model learns not only the actual relationships in the training data but also any noise that is present, as shown in Figure 16a. The low variance is the opposite, and the model may not be able to learn any relationship in the training data, and this phenomenon is called underfitting, as shown in Figure 16b.

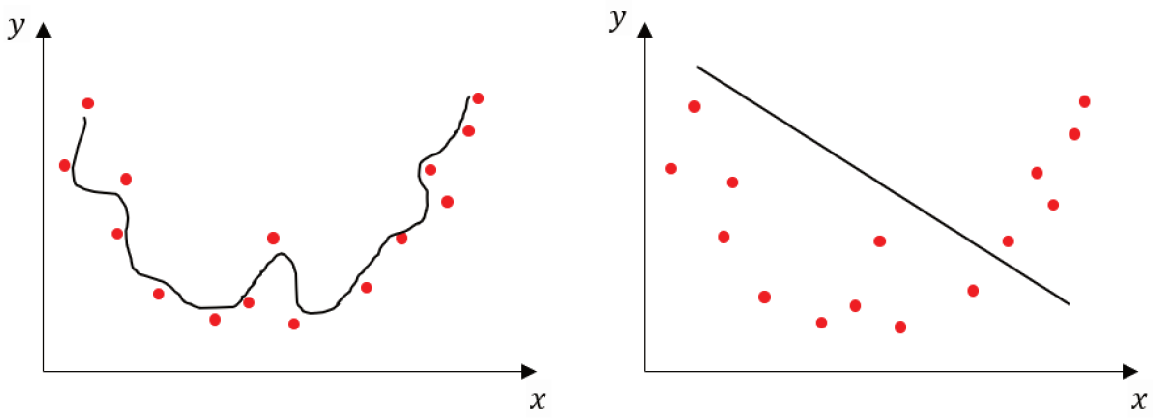
The random subsets of features considered when splitting nodes approach means that only a subset of all the features is considered for splitting each node in each decision tree. A typical way to select the maximum number of features is use

$$M_{features} = \sqrt{n_{features}}, \tag{15}$$

where $n_{features}$ represents the total number of inputs and $M_{features}$ represents the maximum number of features sampled randomly during each node splitting procedure.

During the test time, the predictions are made based on averaging the prediction of each decision tree considering different bootstrapped subsets of features, Figure 15,

Figure 16 – Behaviors of the models



(a) Representation of overfitting

(b) Representation of underfitting

and this process is called bagging.

2.3.2 Extremely Randomized Trees

The extremely randomized trees (ERTs) consist of a tree-based algorithm, formed as a DTs ensemble. This algorithm was first introduced in [42] and works by creating a large number of unpruned DTs from the training dataset, and each of them uses subsets of features ensemble in each split. Predictions are made by averaging the prediction of the DTs in the case of regression or using majority voting in the case of classification.

The difference of ERT from RF is that instead of using a set of DTs obtained by bootstrapping the training dataset, the ERT algorithm fits each DT on the whole training dataset. Like the RF, this algorithm will randomly sample the features at each split point of a DT, but instead of using a greedy algorithm to select an optimal split point, the algorithm selects a split point randomly [42].

By using this random approach instead of finding the best split, as done in RF, the computational cost associated with this algorithm is lower than the one of RF, making it a promising choice in many applications.

2.3.3 Extreme Gradient Boosting

The extreme gradient boosting (XGB), like RF and ERT, is a tree-based model algorithm that uses DT as its building blocks, to create a model based on an ensemble of these trees.

This algorithm was first introduced in [43] and it was an scalable implementation of Gradient boosting machines (GBM) [44]. It differs from other DT ensemble algo-

rithms described in this work by performing scalable boosting and gradient descent approaches to achieve the results.

The boosting aspect refers to the fact that instead of bagging approach, seen in subsection 2.3.1, the individual models are not built on completely random subsets of data and features, but sequentially by putting more weight on instances with wrong predictions. The general idea is that instances, which are hard to predict correctly, will receive more focus during training, with the model being able to exploit past mistakes to improve its performance [45].

The gradient approach is used to minimize the loss function. The idea is that in each round of training, the model parameters are adjusted, and its predictions, $\hat{y}_j$, are built, and compared to the correct outcome that is expected (y). The difference between prediction and real values represents the model error, that can be used to calculate the gradient,

$$\frac{\partial}{\partial \hat{y}_j} \lambda_{\text{xgb}} = \frac{\partial}{\partial \hat{y}_j} \sum_{j=1}^{m} (y_j - \hat{y}_j)^2 \,, \tag{16}$$

which consists of the partial derivative of the loss function. The loss function used was mean squared error (MSE), described as

$$\lambda_{\text{xgb}} = \frac{1}{m} \sum_{j=1}^{m} (y_j - \hat{y}_j)^2 \,, \tag{17}$$

where λ_{xgb} is the MSE, m is the number of examples, y is the real value and $\hat{y}_j$ is the predicted value. The term $\frac{1}{m}$ present in Equation (17) was removed in Equation (16) because it is a constant.

In ANN, gradient descent looks for the minimum of the loss function, learning the parameter (weights) for which the prediction error is the lowest in a single model. In GBM there is a combination of predictions of multiple models, so does not optimize the model parameters directly but the boosted model predictions.

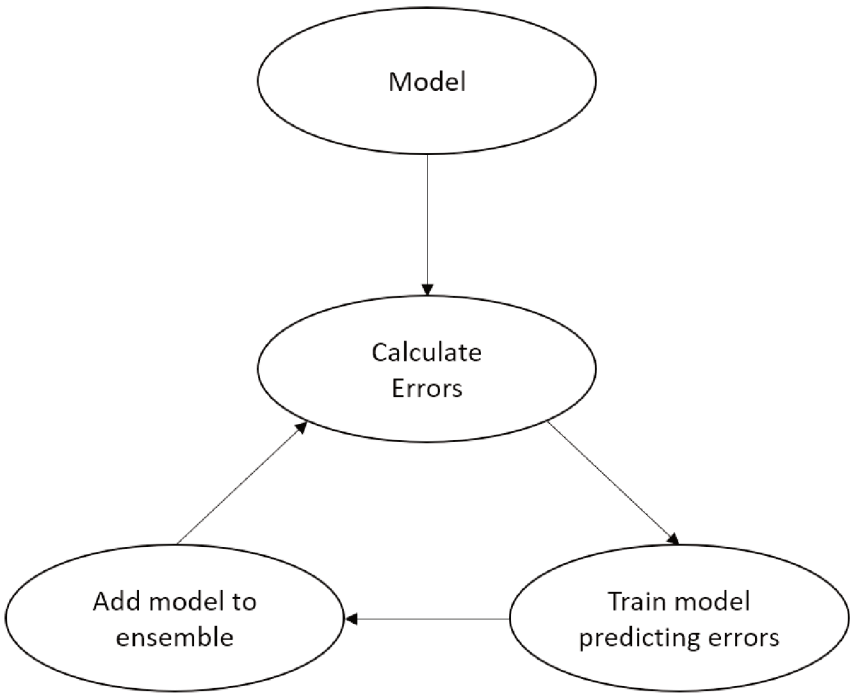
The combination of gradient and boosting approaches is summarized in Figure 17.

What differs XGB from GBM is the fact that it uses more accurate approximations to find the best tree model. There are two main approaches for that: computation of second-order gradients, which provides better and more information about the direction of gradients and how to get to the minimum of the loss function, and implementation of regularizers, which improve the generalization of the model.

2.4 AUTOML TOOLS

During the evolution of machine learning algorithms and approaches for better results, it was evident to many ML practitioners that extracting the best performance

Figure 17 – Gradient boosting concept



Source – Author

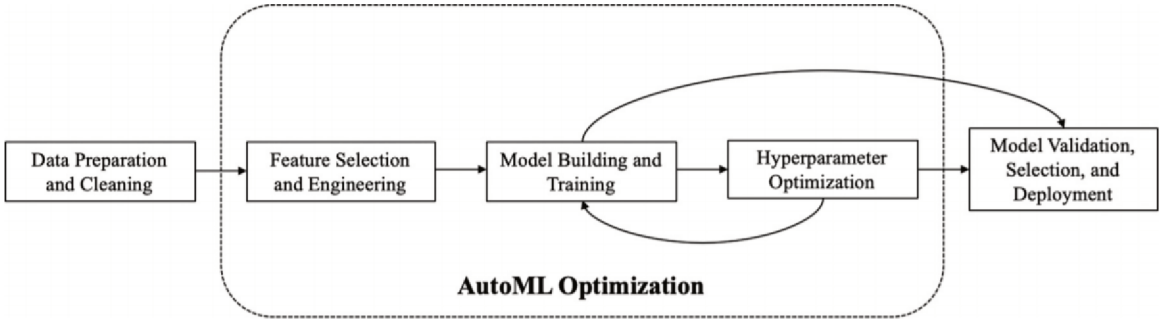
from machine learning models requires substantial human expertise. Developing good models from a dataset is almost an art form involving intuition, experience, and many manual tasks to tune algorithmic parameters. With the combination of market pressure for more ML engineers and data scientists, and the process of developing 'optimal' ML solutions emerged the idea of automating the ML tasks [46].

The history of autoML models is recent. The first autoML tool came based on wekas algorithm, called AutoWeka [47], introduced in 2013. After that, other autoML tools came in sequence based on scikit-learn (ML package for Python): Auto-sklearn [48] in 2014, Hyperopt-sklearn (introduced as automatic hyperparameter configuration) [49] in 2014, TPOT [50] in 2015, Data-robot [51] in 2015, Auto-ml in 2016, H2O-Automl [52] in 2016, Auto-Keras [53] in 2017, and many others. Besides that, some commercial tools emerged combined with cloud service: Cloud Automl [54] in 2017 that runs on Google Cloud platform, Amazon AWS [55] also in 2017, Microsoft AzureML [56] in 2018, Uber's Ludwig in 2018 [57] and many others.

Figure 18 shows the idea of what autoML tools cover in a traditional machine learning pipeline.

The idea in this work is to test some of them, including commercial and non-commercial autoML tools, and compare the results against the traditional ML algorithms. For this purpose, this document covers Hyperopt-Sklearn, TPOT, Cloud Automl, and Microsoft AzureML.

Figure 18 – AutoML concept structure



Source – [58]

2.4.1 Hyperopt-Sklearn

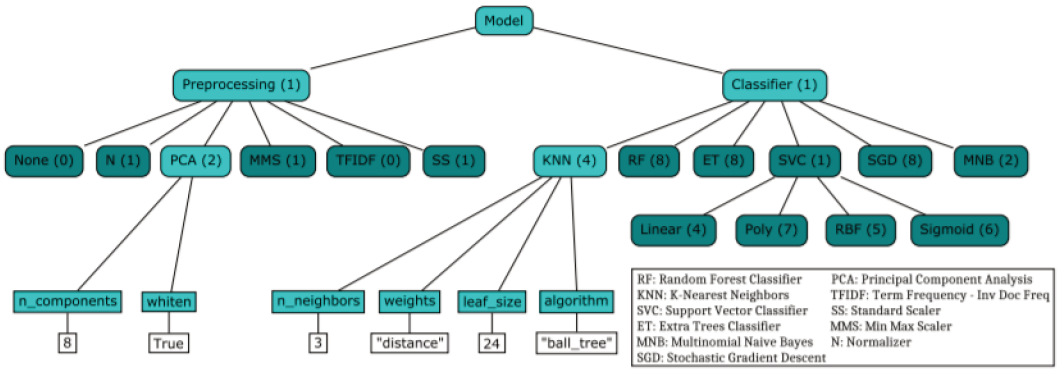
Hyperopt-Sklearn tries to achieve better results on algorithms based on Hyperopt [59] to describe a search space (characterized by a variety of types of variables: continuous, ordinal, categorical, [59]) over possible configurations of scikit-Learn components, including preprocessing and classification modules.

To use this tool it is necessary to define three things: the search domain, an object function, and an optimization algorithm. The search domain is specified via random variables, whose distributions should be chosen so that the most promising combinations have a high prior probability. The objective function maps a joint sampling of these random variables to a scalar-valued score that the optimization algorithm will try to minimize (the same idea of loss function mentioned in the sections above).

Model selection is the process of predicting which model performs better from among a set of possibilities. This process involves a process called Hyperparameter optmization (HPO) or simply parameter tuning (a process of finding the best values for the parameters in a ML algorithm), which traditionally is performed through grid search and random search approaches. Instead of using these conventional methods, Hyperopt-Sklean sets up a search space with random variable hyperparameters, uses scikit-learn to implement the objective function that performs model training and model validation, and uses Hyperopt to optimize the hyperparameters. The model selection is exemplified in Figure 19.

In this classification task example is possible to see that there are 6 possible preprocessing modules and 6 possible classifiers. The highlighted light blue nodes in the second level represent a principal components analyses approached with K-nearest neighbors algorithm, resulting in the final model configuration shown in the blue nodes of the third level. The white leaf nodes at the bottom show example values for their parent parameters.

Figure 19 – Hyperopt-sklearn model selection example



Source – [59]

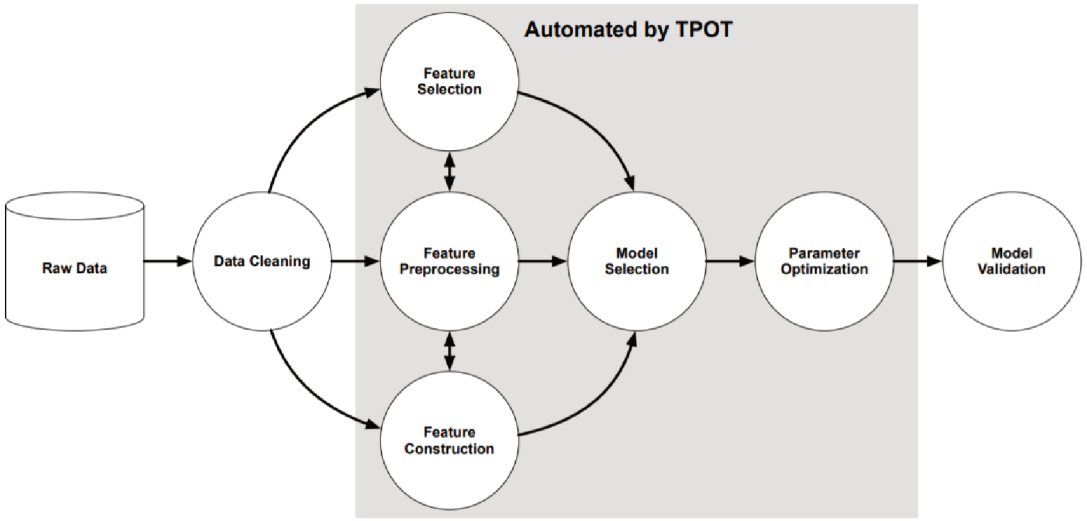
2.4.2 TPOT

Tree-based Pipeline Optimization Tool (TPOT) can be considered as a Python library for automated machine learning because this tool is built on top of scikit-learn. The idea is that TPOT will automate the most "tedious" part of machine learning by intelligently exploring a great number of possible pipelines to find the best one, using genetic programming (GP). It is important to be clear that some parts of the process of ML cannot be fully automated, like data collecting and preparation, which are important steps that depend on hands on parts.

According to [60], GP can be defined as a direct evolution of programs or algorithms for the purpose of inductive learning, which can be considered as a subset of machine learning. The idea is that the GP is inspired by biological evolution and its fundamental mechanisms. GP software systems implement an algorithm that uses random mutation, crossover, a fitness function, and multiple generations of evolution to resolve a defined task. GP can be used to discover a functional relationship between features in data (symbolic regression), and to group data into categories (classification) [61].

Before fitting a model, the user must prepare the data for modeling by performing an initial exploratory analysis (looking for missing data) and either correct or remove the outliers of the data (data cleaning). Next, the user may transform the data in some way to make it more suitable for modeling, for example, by normalizing the features (feature preprocessing), removing features that are not useful for modeling (feature selection), and/or creating new features from the existing data (feature construction). Afterward, the user must select a machine learning model to fit the data (model selection) and choose the model parameters that allow the model to make the most accurate classification from the data (parameter optimization). Lastly, the user must validate the model in some

Figure 20 – TPOT pipeline structure



Source – [50]

way to ensure that the model predictions generalize to data sets which have not been used for training (model validation), for example, by testing the model performance on a holdout dataset that was excluded from the earlier phases of the pipeline. All these steps mentioned in the grey sector are covered automatically by TPOT.

2.4.3 Cloud AutoML

Google provides various machine learning tools. They are divided into five sections: AutML Natural Language, proposed for text and document analyses and classification; AutoML Tables, recommended for structured data problems; AutoML Translation, for translation tasks; AutoML Video Intelligence, for classification of shots and videos according to determined labels; and AutoML Vision, for image classification tasks. In this work, AutoML Tables was used because the main goal was to use structured data to find the evaporation and condensation temperatures.

After importing the data, the platform provides information about missing values, correlation, cardinality, and distribution for each of the features. It is important to mention that Google only starts charging when the training process begins, so the previous steps are free of charge.

When the training process begins, AutoML Tables performs automatically feature engineering and the tasks include normalization and "bucketization" of numeric features, creation of one-hot encoding for categorical features, operation of basic processing for text features and extraction of date and time-related features from timestamp columns. All these preprocessing approaches are used to improve the learning ability of the

algorithms.

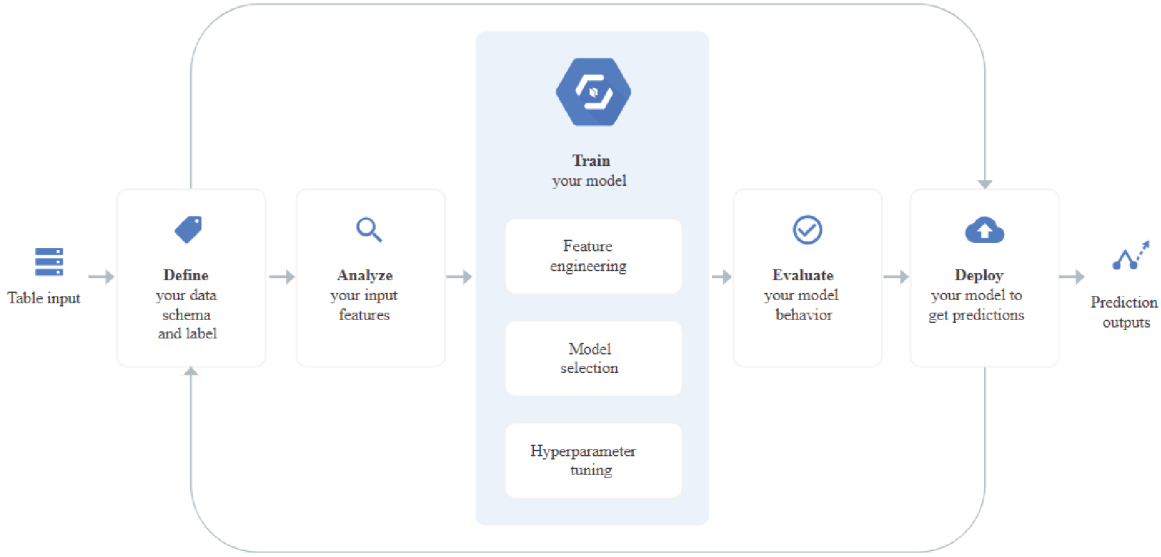
AutoML table performs parallel training, enabling training in different architectures at the same time. It contributes to finding the best model in a short time. Some model architectures that are included are linear regression, feedforward deep neural network, gradient booting decision tree, Ada net, and an ensemble of various model architectures.

As a result, the platform gives the evaluation of the model based on training, validation, and test sets using the best model found and the metrics chosen by the user before training starts. Besides that, it shows the feature importance of the data, which is possible to be downloaded.

Before the training process begins the user must inform how much time will be set as "maximum running time" for searching the best model. If AutoML Table finds the best model earlier, the training is automatically stopped; otherwise, it will be continued until the maximum running time set is reached.

All the processes mentioned above are illustrated in Figure 21.

Figure 21 – How AutoML table works



Source – [62]

2.4.4 Microsoft AzureML

Similar to Google AutoML, Microsoft Azure is an automated machine learning tool that can be really useful to non-expertise ML practitioners. It is also a commercial tool, so Microsoft charges per hour/training, in a similar manner as Google Cloud AutoML.

Before properly starting the training, the user must choose which kind of task is going to be performed. Two possible choices can be made: classification and regression. After that, the user must decide in which framework the task will be performed: in a software development kit (SDK) in Python or in the Web Studio (Microsoft platform). In this work the second option was chosen to really experience the tool capability, and this approach is also recommended for non-programmer users because it does not require any programming skills to set the configurations of the model.

The dataset should be imported and then the user can define which column of the dataset is the target for the training. Besides that, some configurations are possible to be done: maximum training time, the metric that is going to be used to find the best model, set different split for training and test sets of the data, enable early stopping function to avoid overfitting, and some other configurations.

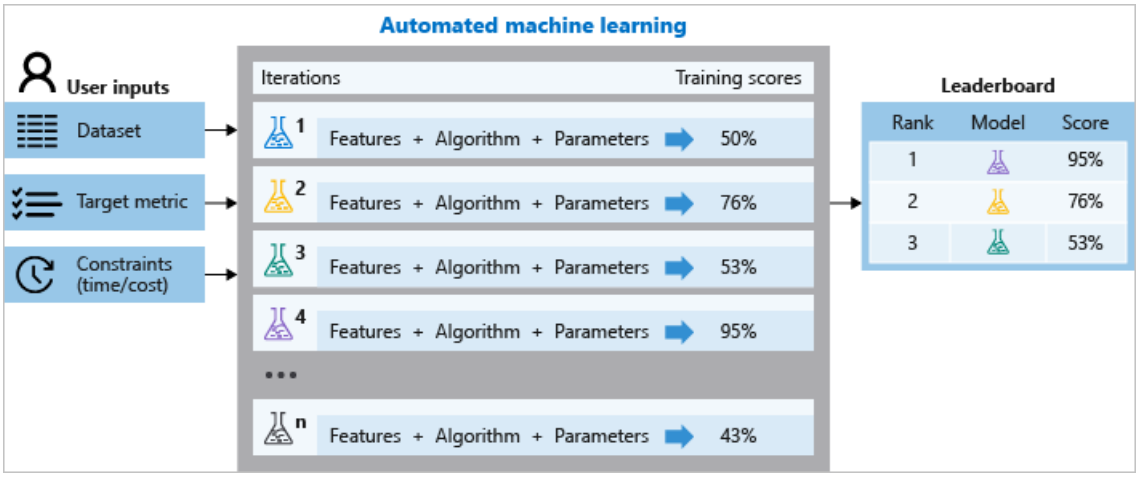
Some preprocessing approaches are also included in AzureML: normalization (stand scaler wrapper, min max scaler, max absolute scaler), principal component analysis (PCA), truncated SDV wrapper, and many others. This preprocessing part is automatically done after the training starts.

For the training task, AzureML uses both voting and stacking ensemble methods for combining models. The first one predicts based on the weighted average of predicted class probabilities (for classification tasks) or predicted regression targets (for regression tasks). The second one combines heterogeneous models and trains a meta-model based on the output from the individual models.

The last configuration that the user must define is where the training process should be performed: local compute or remote compute. The first one runs on the user's computer and the second one uses the cluster of Azure to do the task. The advantage of the cluster is the possibility of parallelization mentioned in subsection 2.4.3, which usually shortens the process for large datasets.

Figure 22 represents all the steps mentioned above.

Figure 22 – Schema of how Azure AutoML works



Source – [63]

3 IMPLEMENTATION OF THE PROPOSED LEARNING TECHNIQUES

In this chapter, the development of the traditional ML models is presented, namely ANN, RF, ERT, and XGB. The chapter also presents a comparison between them and AutoML tools, detailed in section 2.4. In section 3.1 the process to obtain the dataset, the feature creation, and the data preprocessing approach are detailed. In section 3.2 the process of splitting the data into train and test sets is explained, and besides that, the HPO approach used for the traditional ML methods is covered, together with the final parameters for each model obtained.

It is important to mention that some parts of this chapter were omitted in public version of the document to comply with clauses on ownership of intellectual property.

3.1 DATASET AND FEATURE CONSTRUCTION

The data necessary for the development of the models were obtained using a test rig, which was instrumented to measure and control the pressures associated with the suction and discharge of the compressor, and to measure evaporation and condensation temperatures. Using an experimental inverter, it was possible to measure, additionally, the input variables considered in this study. The nominal list was provided to the evaluation committee, but is not presented in this document for industrial property purposes.

In this work, the operating conditions varied between +34 °C and +54 °C for condensation temperature and between -30 °C and -10 °C for evaporation temperature. For each operating condition, the temperatures were measured considering 3 distinct average rotation speeds, consisting of 2100 rotations per minute (rpm), 2850 rpm, and 3600 rpm. These ranges were selected to cover typical application envelope conditions for domestic refrigeration, in order to guarantee that the models were trained considering the information of the main operating conditions that the compressor can be subject to.

The traditional ML pipeline includes not just the feature creation, training and validation parts, but also a "prior-step" focused on feature preprocessing, which can include the outlier remotion, completion of missing values, selection of main features, data augmentation, data balancing, among others. In the ML context, this process is important to avoid some pitfalls during the model development, which can appear when the data are imbalanced, or present spurious values.

After trying some data preprocessing approaches, such as normalization, standardization, PCA, and robust scaling, it was verified that the performance of the models did not change or barely changed for most of the algorithms, with the exception of the ANN, that improved with standardization. So it was decided to focus on the approach of parameter optimization.

The process of standardization consists of rescaling the features to have a mean of 0 and a standard deviation of 1. This technique is often applied in the preprocessing steps because it has the advantage of reducing the large scale difference in the features, if it exists, usually speeding up the algorithm training convergence, and sometimes, improving the results by itself.

3.2 MODEL DEVELOPMENT

The whole dataset was split into train and test sets. The train part corresponds to 70% of the dataset and the test part to the remaining 30%. It is important to mention that the test dataset was not used at any point during the model development phase and it was the same set used to compare the results of all learning techniques in order to achieve a fair performance evaluation. Besides that, when needed, 20% of the train set was used for validation purposes (to check how the model was performing) and this split was done in a shuffled way during the training for each algorithm.

For all the models an algorithm was used for the automatic selection of the hyperparameters, which are the main parameters that influence the performance of the final model. As exposed in subsection 2.4.1, the HPO consists of trying as many different possible combinations of values for the algorithm parameters and this approach is traditionally performed by using grid search and random search. The grid search tries all the possible combinations with the parameters values, which can be a considerable time consuming and computational cost task, depending on the number of parameters to optimize and the range of values to try each of them. The approach of the random search is to randomly get values of a defined range for each parameter, and try as many combinations as the user wants. This process usually requires less computational effort, it is faster, and it can find models that are as good as the ones found by grid search within a small fraction of the computation time [64]. So, in this work random search approach was chosen to perform the HPO.

The HPO was applied to the traditional learning algorithms: random forest, extremely randomized trees, extreme gradient boosting, and artificial neural networks. The AutoML tools have their own hyperparameter optimization processes and it is not necessary to manually apply one. It is important to mention that to define the best combinations of the parameter values, random search used the root-mean-square error (RMSE) as a metric for evaluation. RMSE, Equation (18), is a metric of the model error, varying between zero and infinite, with a value closer to zero representing a better model performance. This metric was chosen because it penalizes large errors, and also because the RMSE metric is presented in the same magnitude order of the model output, in degrees Celsius, making the interpretation of the results easier. The RMSE

is given by

$$\text{RMSE} = \sqrt{\frac{1}{m}\sum_{j=1}^{m}(y_j - \hat{y}_j)^2}\,, \tag{18}$$

where m is the total number of examples, y is the measured value, and $\hat{y}_j$ is the predicted value.

The process of random search present a different duration for each algorithm. The measurement of this time duration was done for comparison purposes, and it is shown in Table 2. The computer processor used is a Intel Core i5-7200U, 2,5 GHz-2,7 GHz, 8Gb of memory, running Windows 10-64bits as operational system.

Table 2 – Hyperparameters search space time duration.

Algorithm	**Evaporation - time (min)**	**Condensation - time (min)**
Random forest	39	49
Extremely randomized tree	16	20
XGBoost	180	420
Artificial neural network	16	25

Source – Author

4 EXPERIMENTAL RESULTS

4.1 TRADITIONAL ML MODEL RESULTS

In this section the results obtained with the best parameter configurations values are presented. For the evaluation of the model, it was checked for similarity between estimated and measured temperatures by comparing them. The metrics used to perform this comparison were RMSE and the coefficient R-squared (R^2), metrics usually applied in regression problems [65]. The R^2 indicates how well the model fits the data, varying between 0 and 1 or 0% and 100%, with results closer to 1 representing better model performance.

Figure 23 shows the results obtained for the RF algorithm, Figure 24 for the ERT algorithm, Figure 25 for XGB algorithm, and Figure 26 for ANN algorithm. All of them consist of the comparison between the measured and estimated temperature values for the test dataset of the evaporation and condensation temperatures, according to the rotation values. In the left side the estimations for the evaporation temperature are presented, on the right side the results for the condensation temperature are represented.

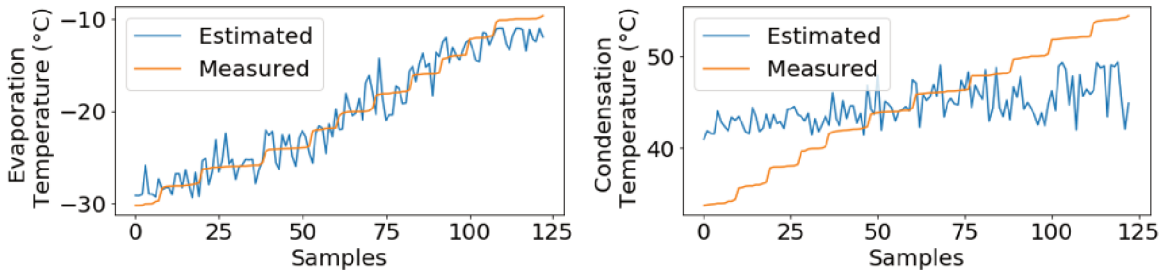
To have a deep analysis of the ranges of both temperatures at which the models could achieve better results, it was plotted the RMSE maps for all the models. The RF RMSE map is shown in Figure 27, the ERT RMSE map in Figure 28, the XGB RMSE map in Figure 29 and the ANN RMSE map in Figure 30. On the left side the results for the evaporation temperature are represented and on the right side the results for the condensation temperature are represented.

For all of the models, it was perceived that the estimations for evaporation temperature were more precise, which suggests that the data present more information about the evaporation temperature, which was also verified in the works of [11] and [9]. This behavior was previously expected because the characteristics of the systems influence more this temperature, or in other words, the dataset used for the work has more information related to the evaporation temperature. Between these temperatures, it was verified a better estimation for the 2100 rpm samples, and this is due to the fact that increasing the rotation, the system was lead to more critical operating conditions, reducing the accuracy of the model.

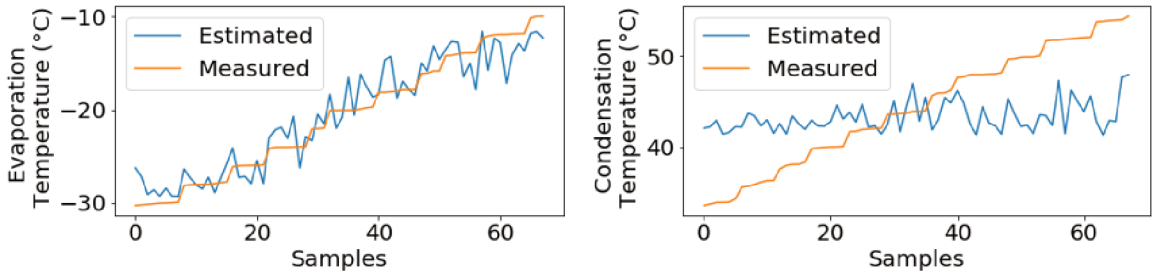
Going deep in the analyses of the quality of the models estimates, it was possible to check which regions of evaporation and condensation temperatures the models could bring better results, according to the RMSE maps. In general, for the region of condensation temperature around +52 °C and +54 °C, the error of the models were higher. The best results could be achieved for the range of +36 °C and +50 °C for condensation temperature and the whole range of the evaporation temperature.

The RF algorithm had the worse performance if compared to the other traditional ML algorithms, but the results were considered satisfactory for evaporation temperature,

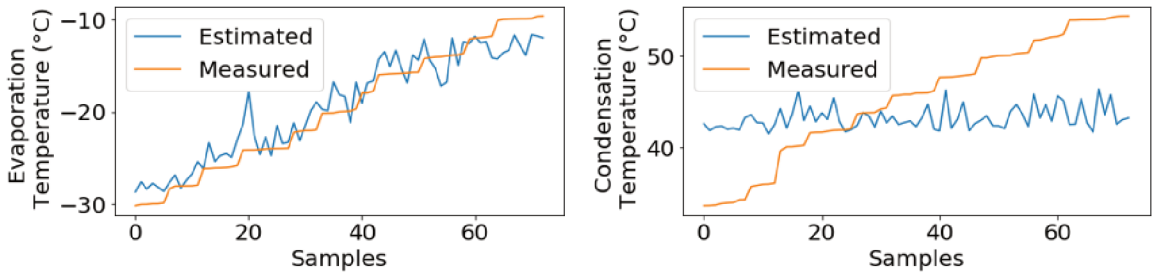
Figure 23 – Results using RF algorithm for evaporation and condensation temperatures



(a) 2100 rpm



(b) 2850 rpm



(c) 3600 rpm

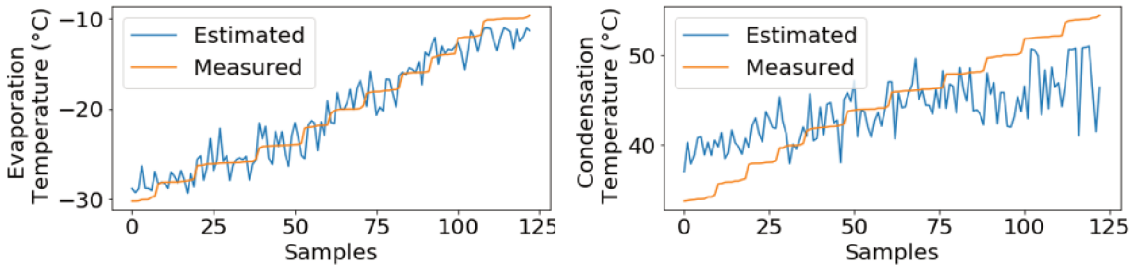
Source – Author

achieving 1.84 °C of RMSE, and 0.91 of R^2. The ERT had the best performance of all the tree-based models, achieving for evaporation and condensation temperatures, respectively, 1.68 °C and 5.22 °C of RMSE, and 0.93 and 0.31 of R^2. The XGB achieved slightly better results than RF, reaching for evaporation and condensation temperatures, respectively, 1.81 °C and 5.50 °C of RMSE, and 0.92 and 0.23 of R^2. Finally, the ANN had much better estimations among all the traditional ML algorithms, achieving for evaporation and condensation temperatures, respectively, 1.46 °C and 4.54 °C of RMSE, and 0.94 and 0.47 of R^2.

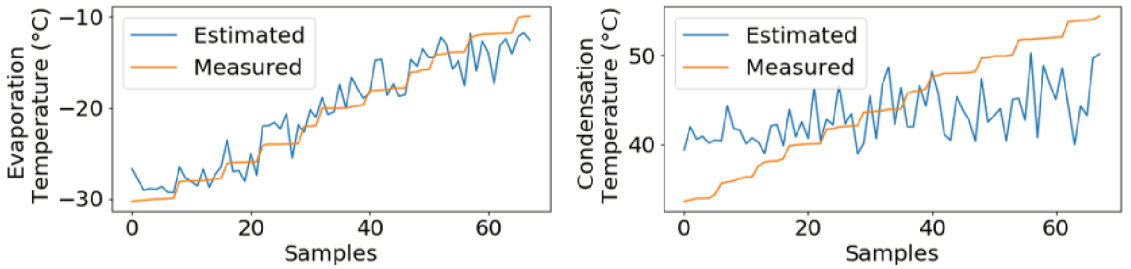
4.2 AUTOML TOOLS

In this section, the AutoML tools mentioned in the section 2.4 were tested with the default setup configuration of each tool. It is important to consider that this work

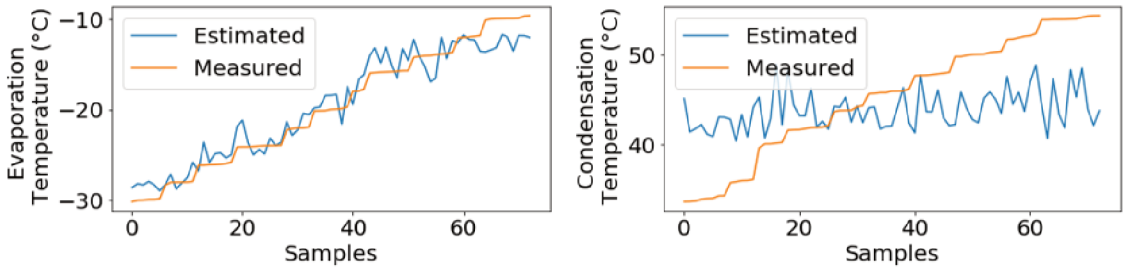
Figure 24 – Results using extremely randomized tree algorithm for evaporation and condensation temperatures



(a) 2100 rpm



(b) 2850 rpm



(c) 3600 rpm

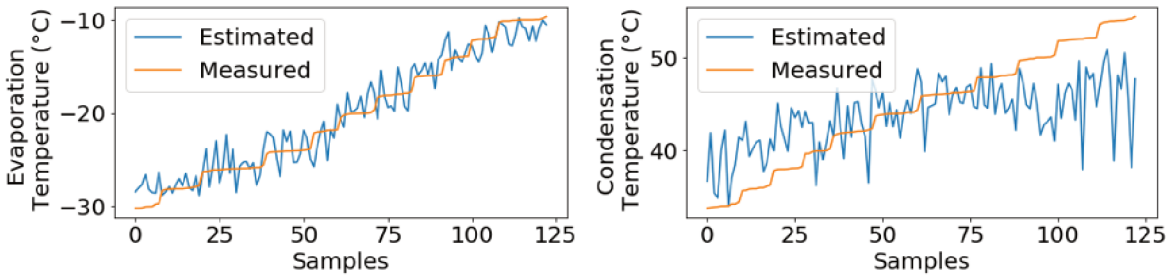
Source – Author

made use of commercial and non-commercial tools, which makes the procedure to obtain each model variable. The input features structure was the same as for traditional ML methods, including the same train, validation, and test sets, in order to have a fair comparison between all the approaches.

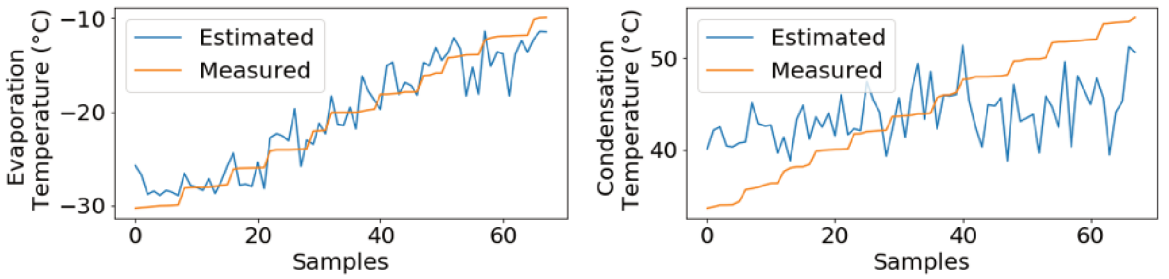
The metrics used to test the AutoML tools were the same as used for the traditional ML approach, and the structure of the graphics results containing the comparison of the estimated and measured temperatures remained the same. The Figure 31 shows the results obtained for the Hyperopt-Sklearn tool, the Figure 32 for the TPOT tool, the Figure 33 for Google Cloud-ML tool, and the Figure 34 for Microsoft Azure tool.

It was noticed that, in general, the AutoML tools could perform really well with the default configurations. To be possible to achieve these results, as mentioned in section 4.1, it was used the early stopping approach for all the AutoML tools, with

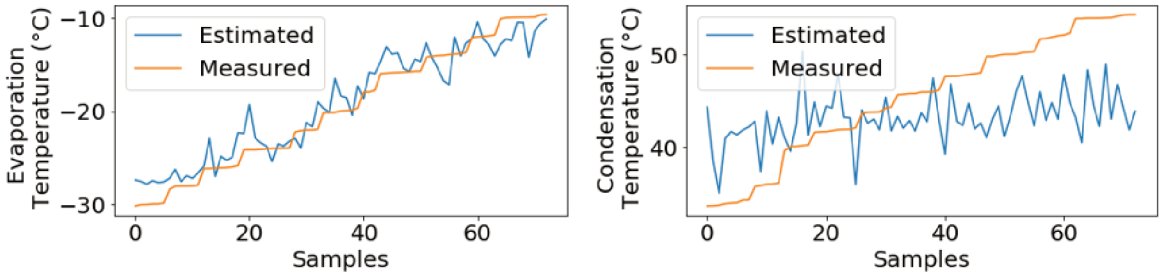
Figure 25 – Results using extreme gradient boosting algorithm for evaporation and condensation temperatures



(a) 2100 rpm



(b) 2850 rpm



(c) 3600 rpm

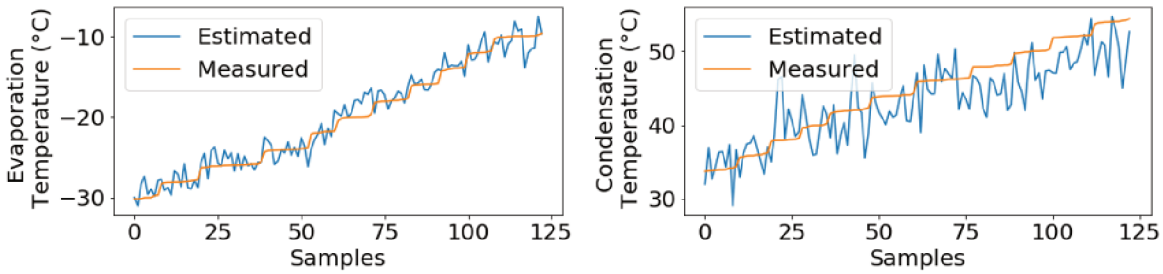
Source – Author

the exception of Hyperopt-Sklearn. Each of the tools also has its own preprocessing techniques, being not needed to develop a preprocessing pipeline for these models.

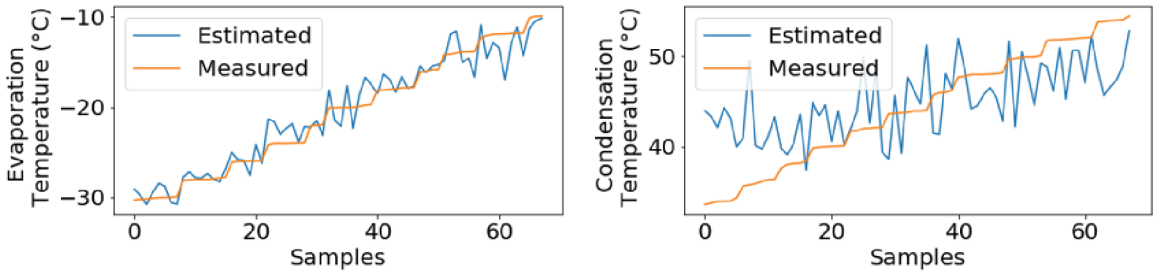
The maximum running time to find the best model for all of the AutoML tool was set to 60 minutes to have a fair comparison. Besides that, it was verified that using the evaporation estimations as input for the condensation model improved the results for Hyperopt-Sklearn and TPOT.

The RMSE maps were also plotted for the AutoML tools in the same structure used for the traditional ML methods. The Hyperopt-Sklearn RMSE map is shown in Figure 35, the TPOT in Figure 36, the Google Cloud-ML in Figure 37, and the Microsoft Azure in Figure 38.

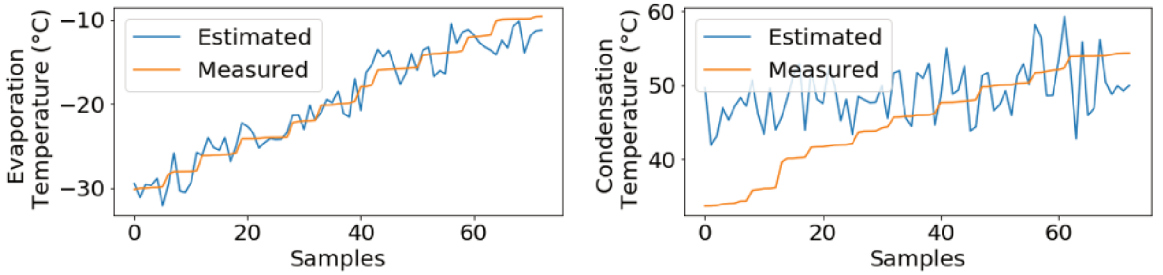
Figure 26 – Results using artificial neural network algorithm for evaporation and condensation temperatures



(a) 2100 rpm



(b) 2850 rpm



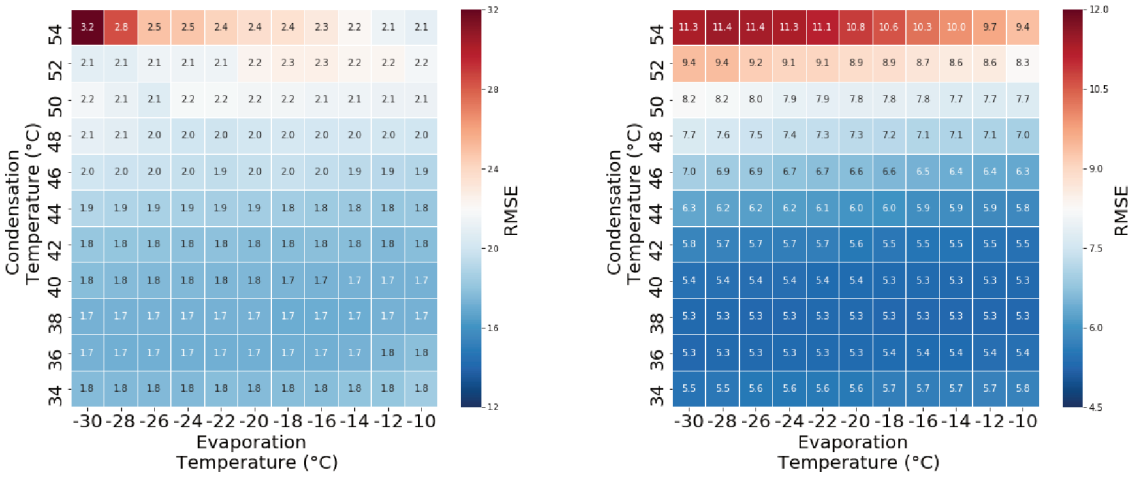
(c) 3600 rpm

Source – Author

As shown in section 4.1, the results of the models were worse in the same condensation temperature ranges, and the lower errors were concentrated in the same range of evaporation and condensation temperatures, corroborating with the results obtained in section 4.1.

In general, the results achieved by the AutoML tools were better than the ones of the traditional ML methods used, which can be seen as a promising approach for future works. Among the evaluated tools, Google Cloud-ML had the best result, reaching for evaporation and condensation temperatures, respectively, 1.38°C and 4.69°C of RMSE, and 0.95 and 0.44 of R^2; followed by TPOT with 1.44°C and 4.89°C of RMSE, and 0.94 and 0.39 of R^2; the Azure which achieved 1.65°C and 5.13°C of RMSE, and 0.93 and 0.33 of R^2, and the Hyperopt-Sklearn with 1.62°C and 5.38°C of RMSE, and 0.93 and 0.26 of R^2. The Azure and Hyeropt-Sklearn had a slight difference, with the Azure

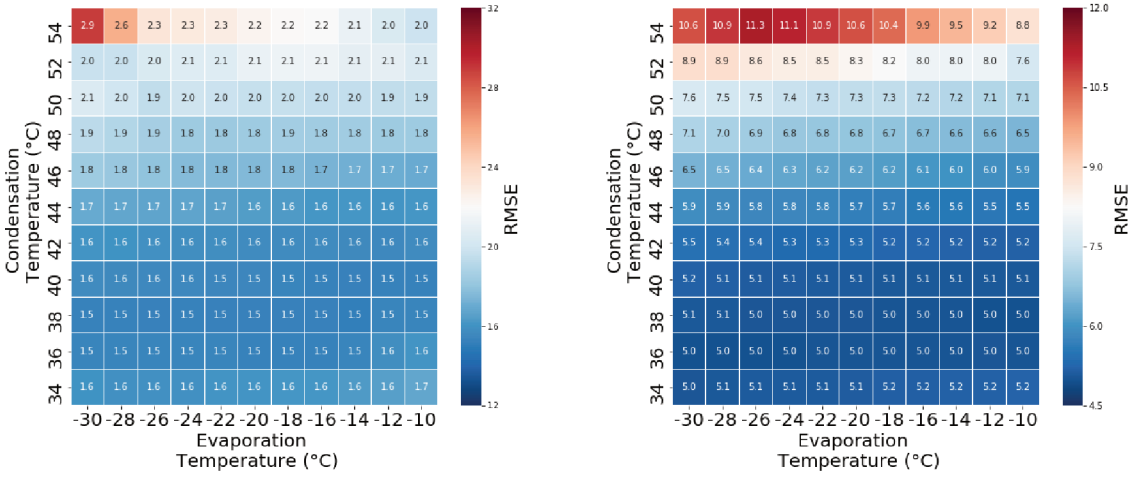
Figure 27 – RMSE map result for random forest algorithm



(a) RMSE for evaporation temperature

(b) RMSE for condensation temperature

Figure 28 – RMSE map result for extremely randomized tree algorithm



(a) RMSE for evaporation temperature

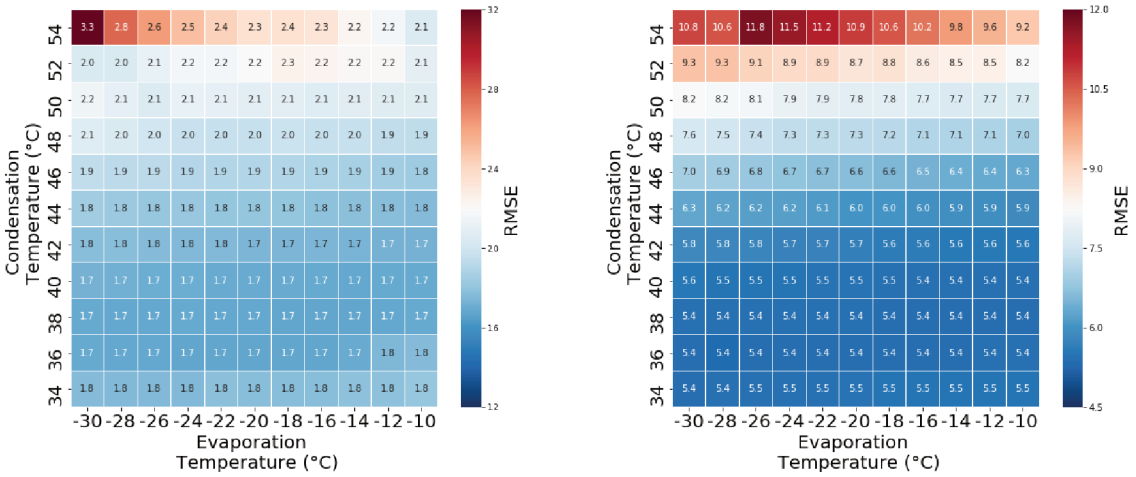
(b) RMSE for condensation temperature

achieving almost imperceptive worse performance in evaporation temperature, however a bigger difference for condensation temperature.

4.3 RESULTS DISCUSSION

As exposed in section 4.1 and section 4.2, the estimation for condensation temperature was harder, which resulted in a worse performance comparing to the evaporation temperature. However, it is known that the level of accuracy obtained by the results can be interesting for some applications, like fault prediction and detection of unwanted operating conditions. Considering the condensation temperature, the best

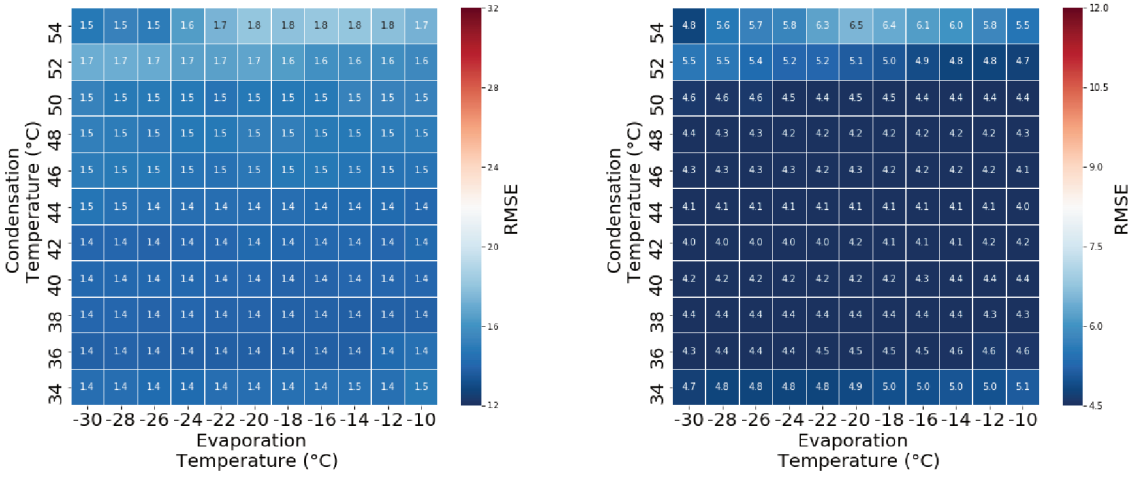
Figure 29 – RMSE map result for XGBoost algorithm



(a) RMSE for evaporation temperature

(b) RMSE for condensation temperature

Figure 30 – RMSE map result for ANN algorithm



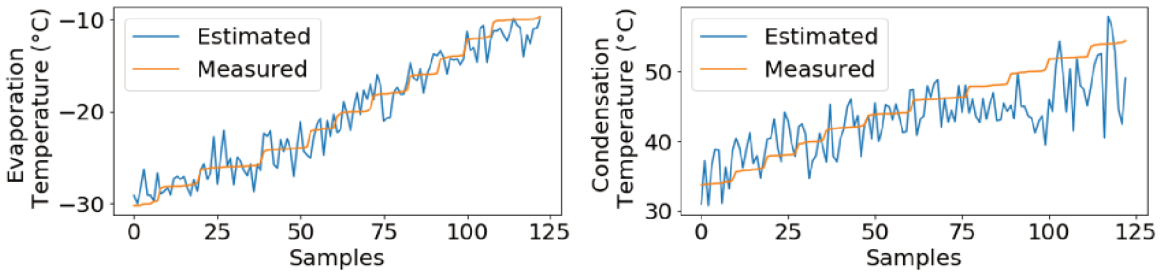
(a) RMSE for evaporation temperature

(b) RMSE for condensation temperature

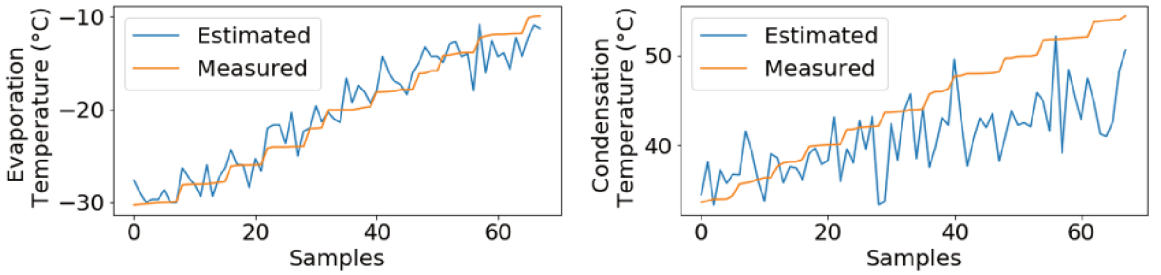
result was achieved by the ANN, followed by Google Cloud-ML and TPOT. For evaporation temperature, the best model was reached by Google Cloud-ML, followed by TPOT and ANN.

All the results are summarized in Table 3 and Table 4. It is important to emphasize that it was used the same test dataset for all the algorithms and AutoML tools to have a fair comparison.

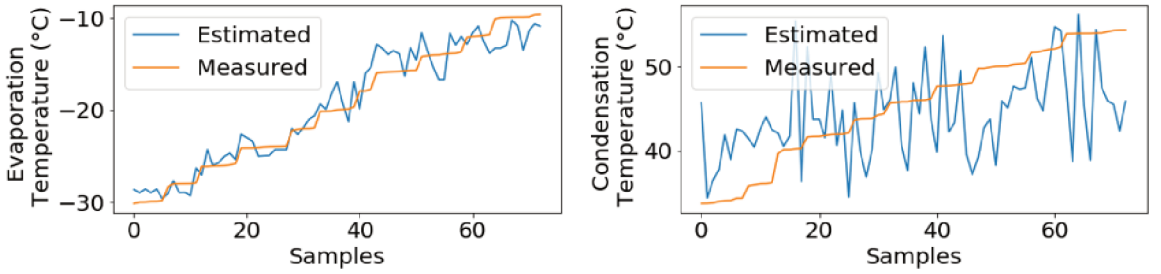
Figure 31 – Results using Hyperopt-Sklearn AutoML for evaporation and condensation temperatures



(a) 2100 rpm



(b) 2850 rpm



(c) 3600 rpm

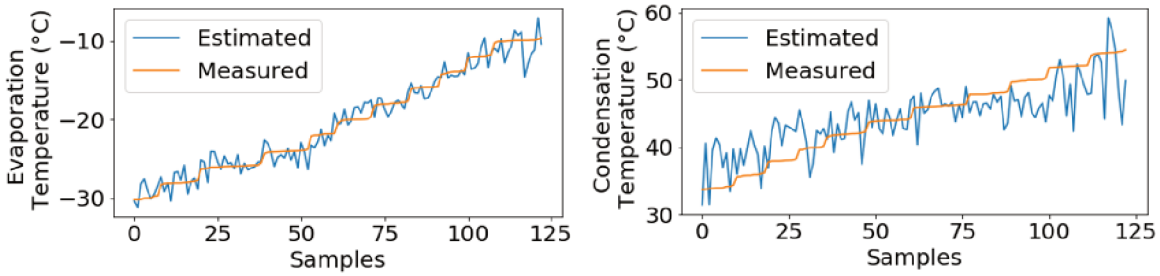
Source – Author

Table 3 – Results for evaporation temperature

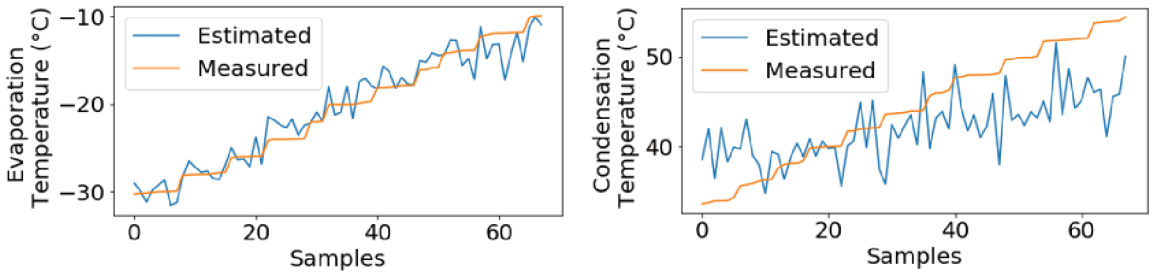
Algorithm	**RMSE (°C)**	R^2
Random forest	1.84	0.91
Extremely randomized tree	1.68	0.93
XGBoost	1.81	0.92
Artificil neural network	1.46	0.94
TPOT	1.44	0.94
Hyperopt-Sklearn	1.62	0.93
Google Cloud-ML	1.38	0.95
Azure-ML	1.65	0.93

Source – Author

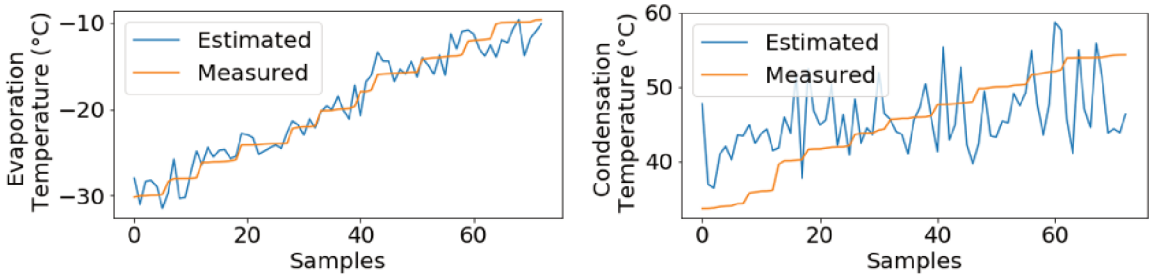
Figure 32 – Results using TPOT for evaporation and condensation temperatures



(a) 2100 rpm



(b) 2850 rpm



(c) 3600 rpm

Source – Author

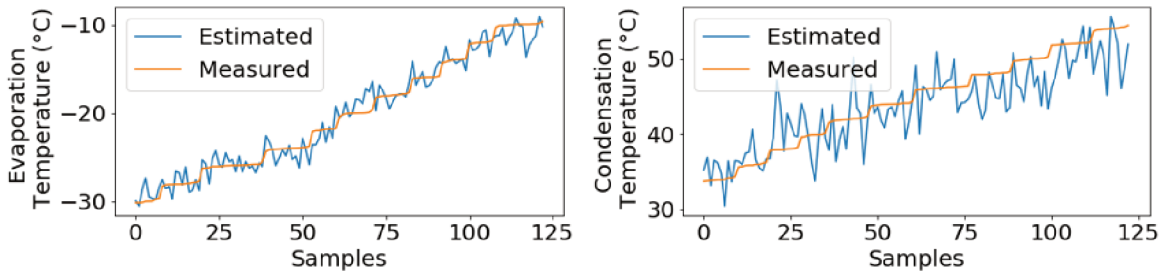
Table 4 – Results for condensation temperature.

Algorithm	**RMSE (°C)**	R^2
Random forest	5.75	0.16
Extremely randomized tree	5.22	0.31
XGBoost	5.50	0.23
Artificil neural network	4.54	0.47
TPOT	4.89	0.39
Hyperopt-Sklearn	5.38	0.26
Google Cloud-ML	4.69	0.44
Azure-ML	5.13	0.33

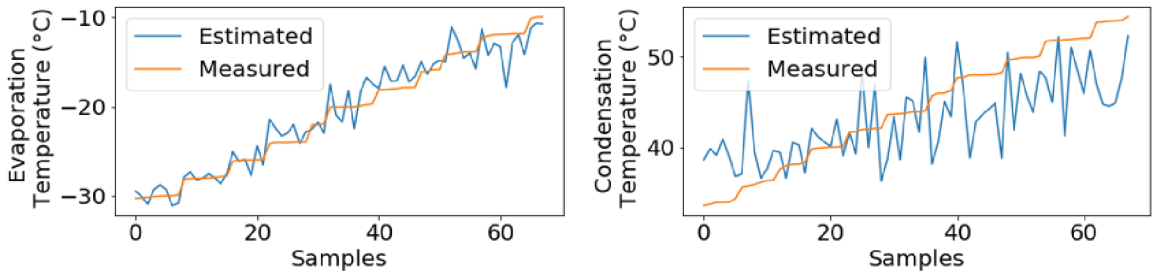
Source – Author

Comparing the results, it is possible to conclude that ANN, TPOT, and Google Cloud-ML achieved better results for both temperatures. This fact reinforces the capability of the AutoML tools to obtain good results, and so, they should be included in

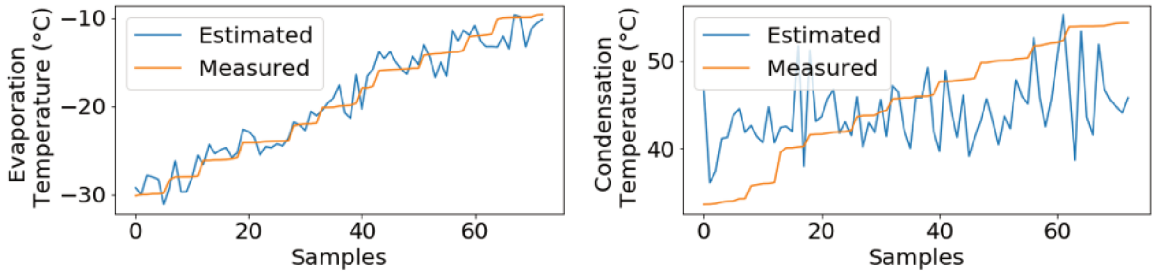
Figure 33 – Results using Google Cloud-ML for evaporation and condensation temperatures



(a) 2100 rpm



(b) 2850 rpm



(c) 3600 rpm

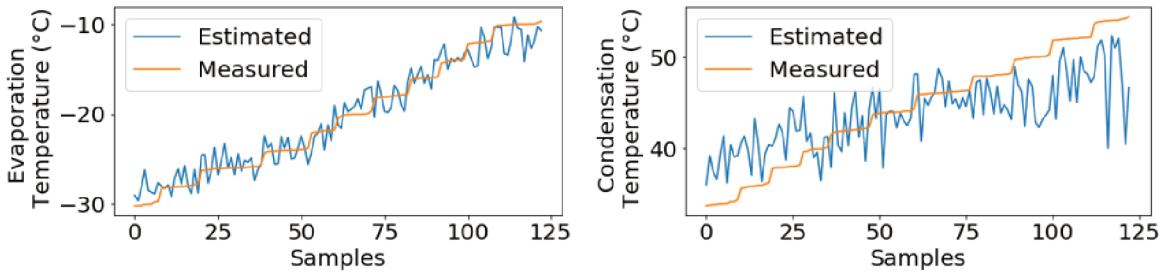
Source – Author

diverse project scopes as possible approaches.

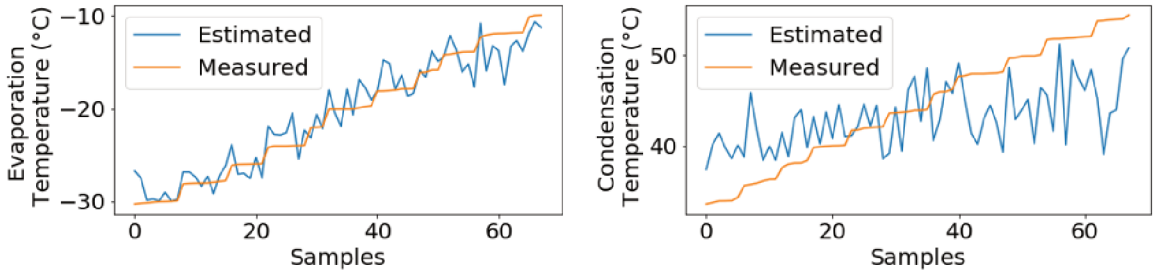
It is hard to select a better approach between traditional ML methods and AutoML tools, but it was concluded that a good results can be achieved with less effort using the second one. Besides that, this approach should be considered for non-programmers, mainly commercial tools like Google Cloud-ML and Azure, because they do not require programming expertise to be able to perform the data preprocessing, training, and testing steps.

In another view, the traditional ML approach also could achieve satisfactory results, and even it was not possible to reach the best one in this work for both temperatures, the research group that supported this work believes that even better results can be reached using traditional ML approach, but in a cost of a significant time and effort on it.

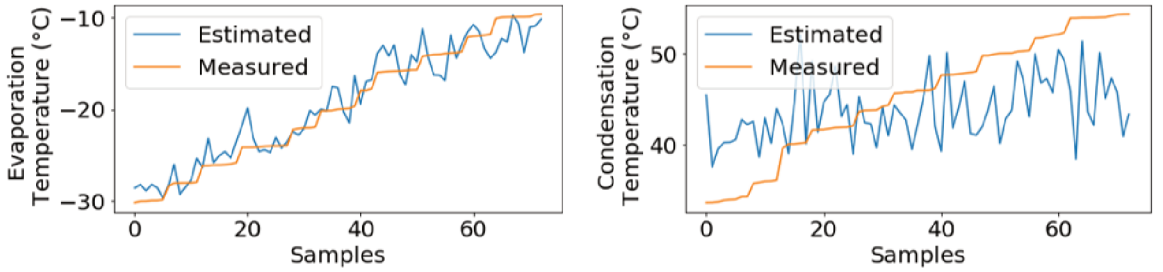
Figure 34 – Results using Azure platform for evaporation and condensation temperatures



(a) 2100 rpm



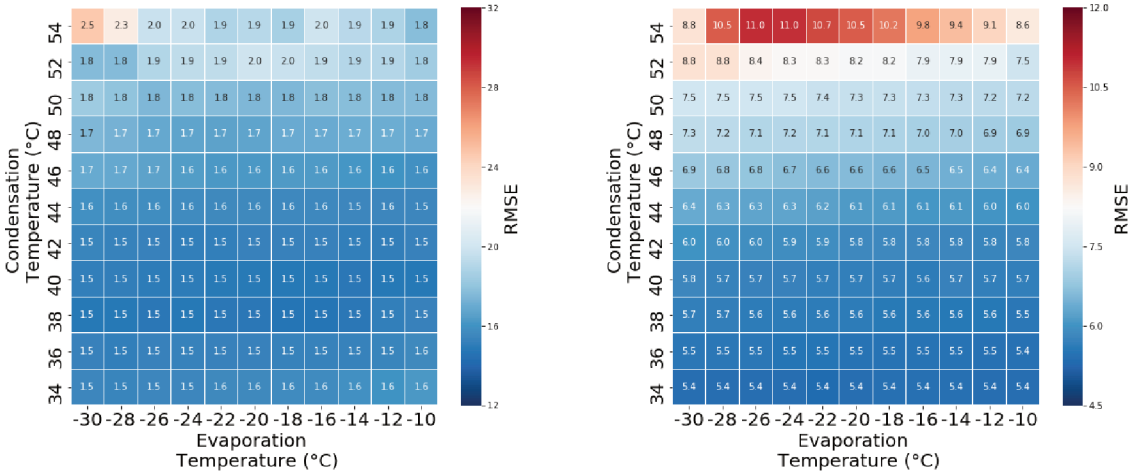
(b) 2850 rpm



(c) 3600 rpm

Source – Author

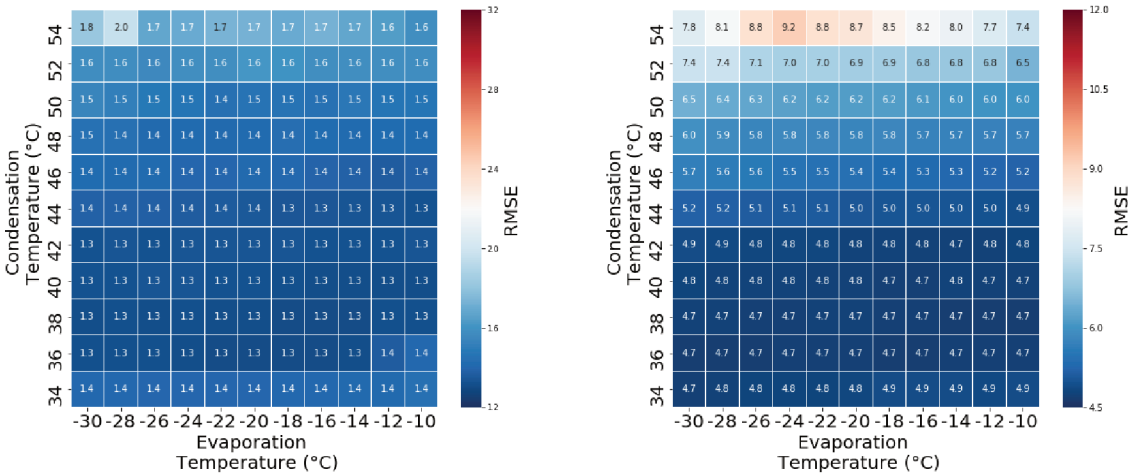
Figure 35 – RMSE map result for Hyperopt-Sklearn



(a) RMSE for evaporation temperature

(b) RMSE for condensation temperature

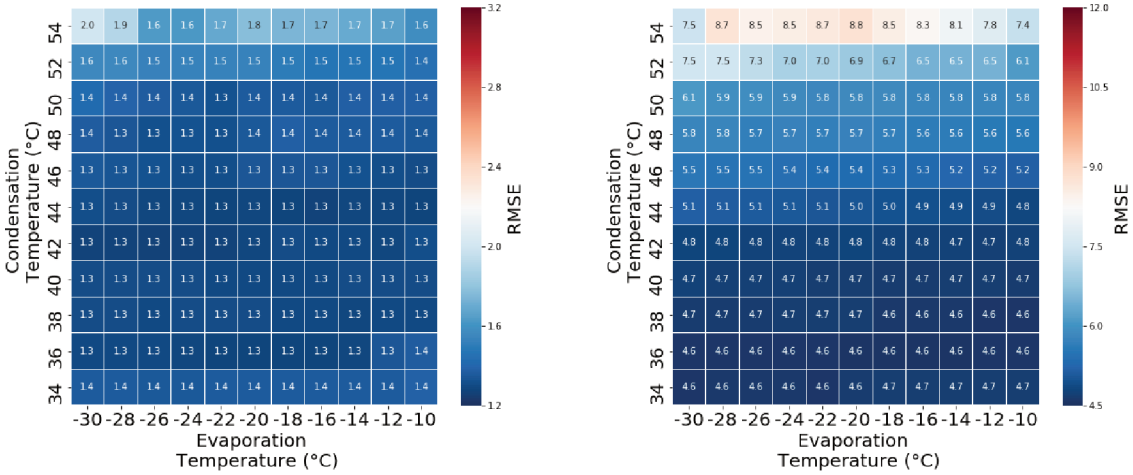
Figure 36 – RMSE map result for TPOT



(a) RMSE for evaporation temperature

(b) RMSE for condensation temperature

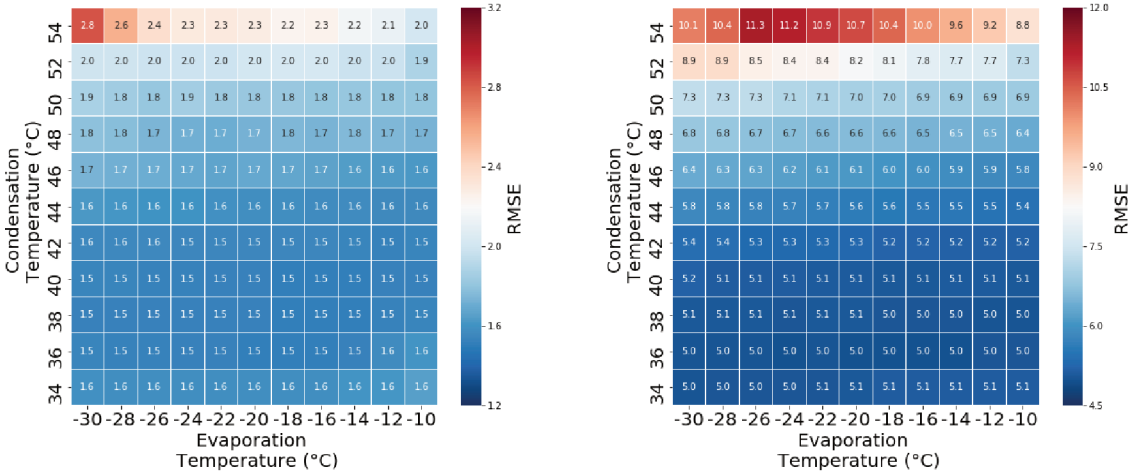
Figure 37 – RMSE map result for Google Cloud-ML



(a) RMSE for evaporation temperature

(b) RMSE for condensation temperature

Figure 38 – RMSE map result for Azure platform



(a) RMSE for evaporation temperature

(b) RMSE for condensation temperature

5 CONCLUSIONS AND NEXT STEPS

In this work the use of traditional ML algorithms and AutoML tools was evaluated to indirectly estimate the evaporation and condensation temperatures of a refrigeration compressor. The achieved results were satisfactory, corroborating for possible works of the company in implementing the non-invasive method to measure evaporation and condensation temperatures using ML algorithms in its products. Besides that, this work studied an alternative approach based on AutoML tools, which is not usual in the refrigeration field, but can bring good results, as shown in chapter 4, and has potential for saving resource and time in configuring all the traditional ML pipeline.

The AutoML tools were experienced and most of the tools analyzed during this work are not really fully automated, as the name suggests. The idea is really clear, which consists of doing automatically the feature selection and engineering, model building and training, and hyperparameter optimization, parts that represent the most time-consuming tasks in the ML pipeline. But in the end, most of the tools required efforts to configure model settings and demanded minor programming expertise, even some tools say the opposite in their descriptions and advertisements. Google Cloud-ML was the most easy-to-use tool among all tested, since it requires no programming expertise and the process of importing the dataset, obtaining the test results, and deploying the final model is intuitive. One thing that should be considered is the fact that it is a commercial tool, so it is not so transparent, meaning that it is hard to know what really happens in the background, which can pose as a limitation, depending on the final model use case.

The Azure automated ML is a really easy-to-use tool, from the upload of the dataset until the training and getting the final model results. But, afterward, the platform is not so clear about the deployment of the best model. Azure has a quick model deployment process, and it generates a REST API URL that can be used by the user. The platform recommends the user to use Microsoft Power BI [66], which is a commercial tool, to get the predictions on a new dataset using this URL, and it can be something unwanted by the user, since it gets tied to this proprietary solution. Another possible approach is using the Azure software development kit (SDK), in Python, performing all steps to obtain new estimations. However, this approach results in loosing the automated aspect. The last option, used in this work, is to download the model trained in pickle format (.PKL) and manually make the estimations.

This work used a dataset with just 1316 examples and obtained in satisfactory results. By using more samples, and consequently more information to the model building, it might be possible to obtain even better results. Besides that, the use of a dataset with different input features, combined with different feature creation approaches, should be considered to achieve better results, mainly for the condensation temperature, which is

harder to estimate with good precision.

Finally, the implementation on an electronic inverter prototype should be considered. The focus of this work was to get as good as possible estimation for the temperatures, not taking into account the implementation of the model in the electronic inverter. To do the implementation, a study of the computational capacity of the inverter microcontroller must be considered, and it will define the complexity of the models that can be used.

REFERENCES

[1] I. Dincer and M. Kanoglu, *Refrigeration systems and applications*, 2nd ed. Wiley Online Library, 2010.

[2] S. Rech, E. Finco, and A. Lazzaretto, “A multicriteria approach to choose the best renewable refrigeration system for food preservation”, *Renewable Energy*, 2020.

[3] M. S. Islam, A. Alam, S. Kamal, *et al.*, “Preservation of medicines & vaccines by thermoelectric (Peltier) refrigeration unit in Bangladesh perspective”, Ph.D. dissertation, Department of Mechanical Engineering, Military Institute of Science and Technology - MIST, 2017.

[4] Research and Markets, *Commercial refrigeration equipment market report: Trends, forecast, and competitive analysis*, Research and Markets, 2020. available from: https://www.researchandmarkets.com/reports/5003251/commercial-refrigeration-equipment-market-report#pos-8 (visited on 08/24/2020).

[5] D. L. P. Cyberaware and H. Interacting, “Intelligent buildings of the future”, 2016.

[6] E.I.A, *Annual energy outlook 2015*, U.S Energy Information Administration, 2015. available from: https://www.eia.gov/outlooks/aeo/pdf/0383(2015).pdf (visited on 10/08/2020).

[7] E. de pesquisa energética (EPE), *Uso de ar condicionado no setor residencial brasileiro, Perspectivas e contribuições para o avanço em eficiência energética*, U.S Energy Information Administration, 2018. available from: https://www.epe.gov.br/sites-pt/publicacoes-dados-abertos/publicacoes/PublicacoesArquivos/publicacao-341/NT%20EPE%20030_2018_18Dez2018.pdf (visited on 10/08/2020).

[8] *Termodinamica: Entenda o que é esse ramo da ciência e suas aplicações na refrigeração*, refrigerationclub, 2020. available from: https://refrigerationclub.com/pt-br/termodinamica-entenda-o-que-e-esse-ramo-da-ciencia-e-as-suas-aplicacoes-na-refrigeracao/ (visited on 11/27/2020).

[9] V. de Aguiar Serra, *Método de estimação de condições de operação em compressores herméticos de refrigeração por meio do perfil de corrente*, Monografia (Engenharia Elétrica), Universidade Federal de Santa Catarina - UFSC, 2020.

available from: https://repositorio.ufsc.br/handle/123456789/204337 (visited on 08/24/2020).

[10] P. O. Sotomayor, *Caracterização e simulação de compressores alternativos utilizando fluidos com baixo potencial de aquecimento global*, Tese (Doutorado em Engenharia Mecânica) - Programa de Pós-graduação em Engenharia Mecânica, Pontifícia Universidade Católica do Rio de Janeiro - PUC-Rio, 2013.

[11] C. B. Walendowsky, *Estimação de pressões de sucção e de descarga de compressores herméticos de refrigeração através de medições não invasivas*, Tese (Doutorado em Engenharia Mecânica) - Programa de Pós-graduação em Engenharia Mecânica, Universidade Federal de Santa Catarina - UFSC, 2017.

[12] N. Kocyigit, "Fault and sensor error diagnostic strategies for a vapor compression refrigeration system by using fuzzy inference systems and artificial neural network", *International Journal of Refrigeration*, vol. 50, pp. 69–79, 2015.

[13] M. Dinç, Ş. Ertekin, H. Özkan, C. Meydanlı, and V. Atalay, "Forecasting of product quality through anomaly detection", in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2019, pp. 357–366.

[14] Y. Zeng, H. Chen, C. Xu, Y. Cheng, and Q. Gong, "A hybrid deep forest approach for outlier detection and fault diagnosis of variable refrigerant flow system", *International Journal of Refrigeration*, vol. 120, pp. 104–118, 2020.

[15] J. K. Hwang, G. Y. Yun, S. Lee, H. Seo, and M. Santamouris, "Using deep learning approaches with variable selection process to predict the energy performance of a heating and cooling system", *Renewable Energy*, vol. 149, pp. 1227–1245, 2020.

[16] Z. Zhang, H. Han, X. Cui, and Y. Fan, "Novel application of multi-model ensemble learning for fault diagnosis in refrigeration systems", *Applied Thermal Engineering*, vol. 164, p. 114 516, 2020.

[17] A. S. B. d. S. Nascimento, R. C. C. Flesch, and C. A. Flesch, "Data-driven soft sensor for the estimation of sound power levels of refrigeration compressors through vibration measurements", *IEEE Transactions on Industrial Electronics*, vol. 67, no. 8, pp. 7065–7072, 2020. DOI: 10.1109/TIE.2019.2941124.

[18] P. Ponce, A. Molina, O. Mata, L. Ibarra, and B. MacCleery, *Power System Fundamentals*. CRC Press, 2017.

[19] D. C. Hanselman, *Brushless permanent magnet motor design*. The Writers' Collective, 2003.

[20] Nidec, *Components of a motor*, Nidec corporation. available from: `https://www.nidec.com/en/technology/motor/basic/00002/` (visited on 10/11/2020).

[21] D. Collins, *Are brushed motors suitable for industrial applications?*, Linear motion tips, 2019. available from: `https://www.linearmotiontips.com/are-brushed-motors-suitable-for-industrial-applications/` (visited on 10/13/2020).

[22] P. Yedamale, "Brushless DC (BLDC) motor fundamentals", *Microchip Technology Inc*, vol. 20, pp. 3–15, 2003.

[23] Y. Y. Jian Zhao, *Brushless DC motor fundamentals application note*, MPS - The future of analogy IC technology, 2011. available from: `https://media.monolithicpower.com/document/Brushless_DC_Motor_Fundamentals.pdf` (visited on 08/27/2020).

[24] G. da Fonseca Pereira, *Simplificação de um método de medição indireta da velocidade de motores de corrente contínua sem escovas*, Monografia (Engenharia de Controle e Automação), Universidade Federal de Santa Catarina - UFSC, 2018. (visited on 08/26/2020).

[25] T. Miller, "Permanent magnet and reluctance motor drives", *New York, Oxf. Univ. Press*, 1989.

[26] T.-W. Chun, Q.-V. Tran, H.-H. Lee, and H.-G. Kim, "Sensorless control of bldc motor drive for an automotive fuel pump using a hysteresis comparator", *IEEE Transactions on Power Electronics*, vol. 29, no. 3, pp. 1382–1391, 2013.

[27] N. Bianchi, S. Bolognani, J.-H. Jang, and S.-K. Sul, "Comparison of pm motor structures and sensorless control techniques for zero-speed rotor position detection", *IEEE Transactions on Power Electronics*, vol. 22, no. 6, pp. 2466–2475, 2007.

[28] I. A. Basheer and M. Hajmeer, “Artificial neural networks: Fundamentals, computing, design, and application”, *Journal of Microbiological Methods*, vol. 43, no. 1, pp. 3–31, 2000.

[29] R. Hecht-Nielsen, “Neurocomputer applications”, in *Neural Computers*, Springer, 1989, pp. 445–453.

[30] A. K. Jain, J. Mao, and K. M. Mohiuddin, “Artificial neural networks: A tutorial”, *Computer*, vol. 29, no. 3, pp. 31–44, 1996.

[31] S. Haykin, *Neural Networks and Learning Machines 3. ed*. Pearson Education India, 2010.

[32] R. A. Syed Danish Ali, *The evolution and core concepts of deep learning neural networks*, Analytics Vidhya, 2016. available from: `https://www.analyticsvidhya.com/blog/2016/08/evolution-core-concepts-deep-learning-neural-networks/` (visited on 08/28/2020).

[33] A. Navlani, *Neural networks models in R*, Data Camp, 2019. available from: `https://www.datacamp.com/community/tutorials/neural-network-models-r` (visited on 08/28/2020).

[34] S. Knerr, L. Personnaz, and G. Dreyfus, “Single-layer learning revisited: A stepwise procedure for building and training a neural network”, in *Neurocomputing*, Springer, 1990, pp. 41–50.

[35] C.-C. Lee, P.-C. Chung, J.-R. Tsai, and C.-I. Chang, “Robust radial basis function neural networks”, *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 29, no. 6, pp. 674–685, 1999.

[36] T. Kohonen, “The self-organizing map”, *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[37] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks”, in *European Conference on Computer Vision*, Springer, 2014, pp. 818–833.

[38] M. Abadi, P. Barham, J. Chen, *et al.*, "Tensorflow: A system for large-scale machine learning", in *12th Symposium on Operating Systems Design and Implementation*, 2016, pp. 265–283.

[39] D. Poojari, *Machine learning basics: Decision tree from scratch*, towards data science, 2019. available from: https://towardsdatascience.com/machine-learning-basics-descision-tree-from-scratch-part-i-4251bfa1b45c#:~:text=What%20is%20Decision%20Tree%20Algorithm,known%20as%20the%20root%20node. (visited on 11/27/2020).

[40] J. Brownlee, *Classification and regression trees for machine learning*, Machine Learning Mastery, 2016. available from: https://machinelearningmastery.com/classification-and-regression-trees-for-machine-learning/ (visited on 08/29/2020).

[41] W. Koehrsen, *An implementation and explanation of the random forest in Python*, Towards Data Science, 2018. available from: https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76#:~:text=When%20training%2C%20each%20tree%20in,times%20in%20a%20single%20tree.&text=At%20test%20time%2C%20predictions%20are,predictions%20of%20each%20decision%20tree. (visited on 08/29/2020).

[42] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees", *Machine learning*, vol. 63, no. 1, pp. 3–42, 2006.

[43] T. Chen, T. He, M. Benesty, V. Khotilovich, and Y. Tang, "Xgboost: Extreme gradient boosting", *R package version 0.4-2*, pp. 1–4, 2015.

[44] J. H. Friedman, "Greedy function approximation: A gradient boosting machine", *Annals of Statistics*, pp. 1189–1232, 2001.

[45] S. Elsinghorst, *Machine learning basics - gradient boosting xgboost*, blog, 2018. available from: https://www.shirin-glander.de/2018/11/ml_basics_gbm/ (visited on 09/01/2020).

[46] A. Truong, A. Walters, J. Goodsitt, K. Hines, C. B. Bruss, and R. Farivar, "Towards automated machine learning: Evaluation and comparison of automl approaches and tools", in *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, IEEE, 2019, pp. 1471–1479.

[47] L. Kotthoff, C. Thornton, H. H. Hoos, F. Hutter, and K. Leyton-Brown, "Auto-weka 2.0: Automatic model selection and hyperparameter optimization in weka", *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 826–830, 2017.

[48] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning", in *Advances in Neural Information Processing Systems*, 2015, pp. 2962–2970.

[49] B. Komer, J. Bergstra, and C. Eliasmith, "Hyperopt-sklearn: Automatic hyperparameter configuration for scikit-learn", in *ICML Workshop on AutoML*, Citeseer, 2014, p. 50.

[50] R. S. Olson, N. Bartley, R. J. Urbanowicz, and J. H. Moore, "Evaluation of a tree-based pipeline optimization tool for automating data science", in *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, 2016, pp. 485–492.

[51] *Datarobot usage examples*, 2018. available from: `https://github.com/datarobot/datarobot-sagemaker-examples` (visited on 09/02/2020).

[52] *H20 AutoML*, 2016. available from: `https://github.com/h2oai/h2o-3` (visited on 09/02/2020).

[53] H. Jin, Q. Song, and X. Hu, "Auto-keras: An efficient neural architecture search system", in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 1946–1956.

[54] *Cloud AutoML*, Google, 2017. available from: `https://cloud.google.com/automl/` (visited on 09/02/2020).

[55] *Amazon SageMaker*, Amazon AWS, 2017. available from: `https://aws.amazon.com/pt/sagemaker/` (visited on 09/02/2020).

[56] *Azure ML*, Microsoft, 2018. available from: `https://docs.microsoft.com/pt-br/azure/machine-learning/concept-automated-ml` (visited on 09/02/2020).

[57] Ludwig, *Ludwig*, Ludwig, 2020. available from: `https://ludwig-ai.github.io/ludwig-docs` (visited on 11/26/2020).

[58] J. Waring, C. Lindvall, and R. Umeton, “Automated machine learning: Review of the state-of-the-art and opportunities for healthcare”, *Artificial Intelligence in Medicine*, p. 101 822, 2020.

[59] J. Bergstra, D. Yamins, and D. D. Cox, “Hyperopt: A Python library for optimizing the hyperparameters of machine learning algorithms”, in *Proceedings of the 12th Python in science conference*, Citeseer, 2013, p. 20.

[60] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone, *Genetic programming*. Springer, 1998.

[61] D. Radečić, *Tpot: Automated machine learning in python*, Towards Data Science, 2020. available from: `https://towardsdatascience.com/tpot-automated-machine-learning-in-python-e56800e69c11` (visited on 09/04/2020).

[62] *Automl tables*, Google Cloud, 2020. available from: `https://cloud.google.com/automl-tables` (visited on 09/05/2020).

[63] *Azure: How automl works*, Microsoft Azure, 2020. available from: `https://docs.microsoft.com/pt-br/azure/machine-learning/concept-automated-ml` (visited on 09/05/2020).

[64] J. Bergstra and Y. Bengio, “Random search for hyper-parameter optimization”, *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 281–305, 2012.

[65] G. S. Handelman, H. K. Kok, R. V. Chandra, A. H. Razavi, S. Huang, M. Brooks, M. J. Lee, and H. Asadi, “Peering into the black box of artificial intelligence: Evaluation metrics of machine learning methods”, *American Journal of Roentgenology*, vol. 212, no. 1, pp. 38–43, 2019.

[66] Microsoft, *Microsoft Power BI*, Microsoft, 2020. available from: `https://powerbi.microsoft.com/pt-br/` (visited on 11/12/2020).