Pontifícia Universidade Católica
do Rio de Janeiro

Rodolfo Spinelli Teixeira

Application of machine learning algorithms to predict fuel efficiency based on trip parameters: a heavy haul railway case of study

Dissertação de Mestrado

Dissertation presented to the Programa de Pós–graduação em Engenharia Mecânica of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Engenharia Mecânica.

Advisor: Prof. Ivan Fabio Mota de Menezes

Rio de Janeiro
October 2021

Pontifícia Universidade Católica
do Rio de Janeiro

Rodolfo Spinelli Teixeira

Application of machine learning algorithms to predict fuel efficiency based on trip parameters: a heavy haul railway case of study

Dissertation presented to the Programa de Pós–graduação em Engenharia Mecânica of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Engenharia Mecânica. Approved by the Examination Committee:

Prof. Ivan Fabio Mota de Menezes
Advisor
Departamento de Engenharia Mecânica – PUC-Rio

Prof. Anderson Pereira
Departamento de Engenharia Mecânica – PUC-Rio

Prof. Saul de Castro Leite
UFABC

Rio de Janeiro, October 6th, 2021

Rodolfo Spinelli Teixeira

Majored in mechanical engineering by the Federal University of Juiz de Fora (Juiz de Fora-MG, Brazil) in 2017.

Bibliographic data

S. Teixeira, Rodolfo

Application of machine learning algorithms to predict fuel efficiency based on trip parameters: a heavy haul railway case of study / S. Teixeira, Rodolfo; advisor: Menezes, Ivan Fábio Mota de. – 2021.

68 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Mecânica, 2021.

Inclui bibliografia

1. Engenharia Mecânica – Teses. 2. energy consumption. 3. fuel efficiency. 4. railway. 5. heavy haul. 6. machine learning. 7. artificial neural network. 8. random forest. 9. partial dependence plot. 10. accumulated local effect plot. I. Menezes, Ivan Fábio Mota de. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Mecânica. III. Título.

CDD: 621

In memory of my loved brother,
João Victor Spinelli Teixeira.

Acknowledgments

First I would like to thank my advisor professor Ivan Menezes for his support and knowledge during this period.
I also would like to thank Dr. Renan Finotti for his major contribution regarding all the technical subjects related to the machine learning field.
Then I would like to thank my parents Messias and Sonia for their support and encouragement. My dear partner Daiana, for her patient and love with me during all this time.
A special thanks to my loved brother João Victor who inspired me to follow his paths as an engineer and who always supported my decisions.
Finally, I would like to thank PUC-Rio professors and all staff.
This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001

Abstract

S. Teixeira, Rodolfo; Menezes, Ivan Fábio Mota de (Advisor). **Application of machine learning algorithms to predict fuel efficiency based on trip parameters: a heavy haul railway case of study**. Rio de Janeiro, 2021. 68p. Dissertação de Mestrado – Departamento de Engenharia Mecânica, Pontifícia Universidade Católica do Rio de Janeiro.

Fuel consumption in companies in the rail transport sector represents one of the largest operating expenses and one of the biggest concerns in terms of pollutant emissions. The high fuel consumption also entails a high representation in the emissions scope matrix (more than 90% of railroad emissions come from fossil fuel consumption). Aiming to seek constant operational improvement, numerous studies have been carried out proposing new tools to reduce fuel consumption in the operation of a freight train. In this way, it is important to highlight the improvement of train driving parameters that can be calibrated to reduce fuel consumption. To accomplish this goal, the present work implements two machine learning models to predict the energy efficiency of a freight train: random forest and artificial neural networks. The random forest achieves the best performance against the models, with an accuracy of 91%. To calculate how much each parameter influences the prediction model, this work also uses the technique of accumulated local effects for each parameter related to energy efficiency. The final results show that, within the four analyzed calibration parameters, the traction per transported ton indicator presented greater representation in terms of absolute impact on the energy efficiency of a freight train

Keywords

energy consumption; fuel efficiency; railway; heavy haul; machine learning; artificial neural network; random forest; partial dependence plot; accumulated local effect plot.

Resumo

S. Teixeira, Rodolfo; Menezes, Ivan Fábio Mota de. **Aplicação de algoritmos de aprendizado de máquina para prever eficiência energética baseado em parâmetros de viagem: estudo de caso de uma ferrovia de transporte de carga**. Rio de Janeiro, 2021. 68p. Dissertação de Mestrado – Departamento de Engenharia Mecânica, Pontifícia Universidade Católica do Rio de Janeiro.

O consumo de combustível em empresas do setor de transporte ferroviário representa um dos maiores gastos operacionais e uma das maiores preocupações em termos de emissões de poluentes. O alto consumo em combustíveis acarreta também em uma alta representatividade na matriz de escopo de emissões (mais de 90% das emissões de ferrovias são provenientes do consumo de combustível fóssil). Com o viés de se buscar uma constante melhora operacional, estudos vêm sendo realizados com a finalidade de se propor novas ferramentas na redução do consumo de combustível na operação de um trem de carga. Nesse ramo, destaca-se o aperfeiçoamento dos parâmetros de condução de um trem que são passíveis de calibração com o objetivo de reduzir o consumo de combustível. Para chegar a esse fim, o presente trabalho implementa dois modelos de aprendizado de máquina (machine learning) para prever a eficiência energética de um trem de carga, são eles: floresta randômica e redes neurais artificiais. A floresta randômica obteve o melhor desempenho entre os modelos, apresentando uma acurácia de 91%. Visando calcular quanto cada parâmetro influencia no modelo de previsão, este trabalho também utiliza técnica de efeitos acumulados locais em cada parâmetro em relação à eficiência energética. Os resultados finais mostraram que, dentro dos quatro parâmetros de calibração analisados, o indicador de tração por tonelada transportada apresentou maior representatividade em termos de impacto absoluto na eficiência energética de um trem de carga.

Palavras-chave

consumo energético; eficiência energética; ferrovia; transporte pesado; aprendizado de máquina; redes neurais artificiais; florestas aleatórias; gráfico de dependência parcial; gráfico de valores acumulados locais.

Table of contents

List of Figures

List of Tables

List of Abreviations

CO_2 – Carbon dioxide

DNIT – Departamento Nacional de Infraestrutura e Transporte

FE – Fuel Efficiency

ML – Machine Learning

RF – Random Forest

ANN – Artificial Neural Network

SVM – Support Vector Machines

GB – Gradient Boosting

DT – Decision Trees

HPT – Horse Power per Tonnage

MSE – Mean Squared Error

PDP – Partial Dependence Plot

ALE – Accumulated Local Effect

TPE – Tree of Parzen Estimators

SMBO – Sequential Model-Based Global Optimization

Wisdom is a shelter
as money is a shelter,
but the advantage of knowledge is this:
Wisdom preserves those who have it.

Ecclesiastes 7:12, *Holly Bible.*

1
Introduction

It is evident that the world environment has been changing due to human intervention, such as, deforestation and the increasing of greenhouse gases emission. The former one, has been the focus in many technical studies and government coalitions, such as the Paris Agreement, which aims to reduce the number of gases generated by human activities such as the transportation sector. In this way, according to a study led by the United Nations [1], the transportation sector is responsible for more than 13% of CO_2 emissions in the world, being one of the main gas contributors to increase the climate change nowadays. The contribution of each sector on green house gases emission is showed in Figure 1.1. It is important to emphasize in this Figure, that transportation sector is ranking at third place as a major contributor. Consequently, this sector is considered strategic for policy regulations to ensure lower emissions and fossil fuel consumption.

To reach a sustainable level of CO_2 emission, it is essential to search for new trends and devices to achieve better fuel efficiency. The Brazilian railway sector has been committed to reduce fuel consumption and greenhouse gases emission, by means of new technologies and optimized operations. As reported by Brazilian official infrastructure department (DNIT) website [2], railway will be a strategic logistic operator and responsible for sustainable growth due to its higher fuel efficiency and lower emissions when compared to road transportation. Therefore, predicting energy efficiency and understanding which factors interfere with final consumption are vital to meet these strategic requirements.

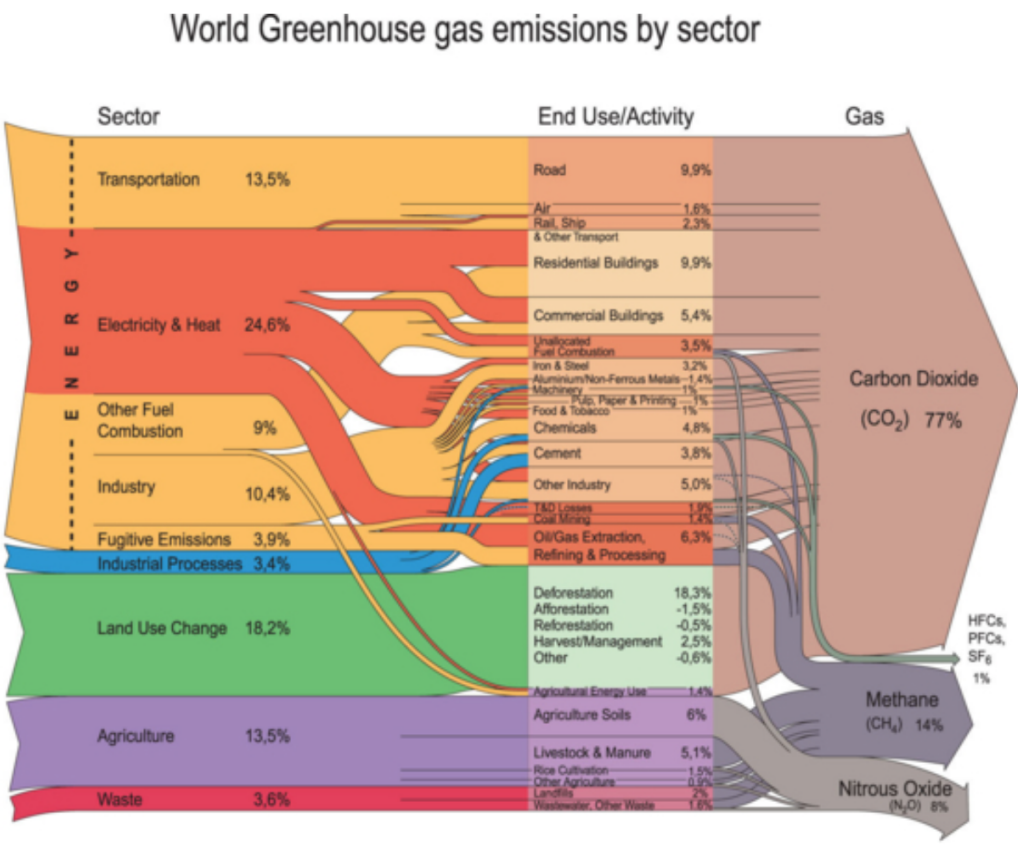

Figure 1.1: World Greenhouse gas emissions by sector [1]

1.0.1 Objectives

The main focus of this work is to apply machine learning techniques to model the fuel consumption as a prediction model. Two techniques will be studied and the optimal one will be used aiming to analyze which parameters have the higher influence on the heavy haul transportation.

1.0.2 Thesis Organization

This work is divided in five chapters. Chapter 2 presents the literature review in which a series of machine learning and conventional techniques will be explored regarding prediction of fuel efficiency. The problem statement will be addressed in Chapter 3 with the objective to define the whole process: from start of a trip by the locomotive driver to data acquisition. Moreover the variables presented in this work dataset will be explained along with their data distribution. In chapter 4 the methodology section will be placed to describe the mathematical formulation used in each prediction model. Chapter 5 will show the results generated by the two machine learning techniques and the

optimal one will be chosen in order to extract feature dependence regarding energy efficiency. Finally, Chapter 6 will present the main conclusions of this work, together with suggestions for future work.

2
Brief Literature Review

Fuel consumption is major concern in transportation field due to its economical and environmental impact [3]. Hence, a lot of researches have been conducted by industry and academy as means to simulate real conditions and to calculate accurately energy needs. In this way, the next two topics describes how machine learning and other methods are being used to accurately predict fuel consumption and efficiency in transportation field.

2.1
Machine learning algorithm to predict fuel consumption

Machine Learning (ML) algorithms are being extensively used in plentiful fields from healthcare diagnoses to gambling strategies. The main objective behind applying these algorithms is to forecast desired results, as they are commonly nonlinear problems. A recent example is a study led by Zou et al. [4] n which they used decision trees, Random Forest (RF), and Artificial Neural Network (ANN) to predict diabetes mellitus in a classification problem. The final results proved that prediction with RF could reach the highest accuracy, reaching 81%. In the transportation field, one of the main goals is to use regression techniques to predict fuel consumption, due to its crucial impact in cost. In their work, Yao and Moawad [5] employed a Large Scale learning and prediction process using ANN to predict fuel economy by supplying parameters, such as, vehicle glider mass and engine power. For their particular problem, the time spent on simulations were extremely high and by using ANN they could save up to 64% in time procurement. Gaussian Process was used by Xu and Zhao [6] to address fuel consumption in 100 km traveled by automobiles. According to the authors, compared with ANN and other complex methods, Gaussian Processes has the advantage of easy applicability under obtaining better performance conditions and there is no need for hyperparameter tuning. They concluded that this machine learning algorithm could lead to up to 0.07% relative error. A different approach was explored by Gkerekos et al. [7] in which was studied a broad number of predictors, in particular RF, ANN, Lasso Regression, Support Vector Machines (SVM) and Decision Trees (DT) to predict fuel consumption of vessels. RF and

extra tree models were found to be the most accurate for their research, achieving 90% of accuracy on the test set. As a suggestion for this study, the authors advised exploring hyperparameter optimization to achieve better results. Addressing a similar problem Karagiannidis at el. [8] focused on applying ANN to predict vessel fuel consumption based on trip and weather parameters. Results demonstrated that with proper data prepossessing, it is possible to achieve an increased performance of the ship propulsion models, which consequently, increase the awareness of the ship's performance condition. Moreover, improvements on the effective decisions regarding strategies and operational measures to reduce fuel oil consumption, led to reduction in emissions. Fuel consumption could be predicted with a 98.7% accuracy after all the steps regarding data treatment. With the same goal to forecast fuel consumption, a different ML approach was employed by Kand and Hensen [9] implementing an ensemble learning techniques to improve fuel burn prediction in aviation segment. The Lasso-based stacking was found to reduce the mean squared prediction error by 50% over the current aviation system. Fuel Consumption prediction for fleet vehicles using ML was explored by Wickramanayake and Bandara at [10]. Given available time series data, the authors evaluated three different ML techniques to predict fuel consumption for long distance public buses: RF, gradient boosting and ANN. Through their work, it was proved that RF achieved better results when compared with the other ML algorithms used.

2.2 Review of ML algorithm to predict train fuel consumption

Different techniques have been used in the literature to correctly predict the amount of fuel spent on a train trip. For example, Xia and Zhang et al. [11] applied a control system to model train components, such as braking and acceleration, to derive final fuel consumption.. Different types of control system were analyzed such as open and closed loops, however the former achieved better results for energy modeling. Nonetheless the finds of this work could not be applied for real situations were external factors take place. By modeling the train as a discrete mass system, Shi et al. [12] considered only longitudinal dynamics to calculate train fuel consumption since lateral and vertical could be neglected. Main findings in their work were about ensuring coupler forces to meet standard operational values. Moreover, due to its characteristics of log-steep downgrades, energy consumption accounted for more than 80% of the total energy consumption. Another technique would be using physical and empirical deterministic equations to calculate tractive effort, train resistance,

distance for acceleration, breaking distance and time in order to calculate fuel consumption in Train Performance Simulator as described by Hoyt and Levary at [13]. They found out that the most beneficial operating conditions for the specified configuration studied would be a train formed by 65 to 83 wagons. Sun et al. [14] devoted to define a traction energy consumption model to accurately analyze and predict the energy consumption which is the main energy consumption in urban rail transit systems. Based on four years of railway energy consumption data the authors applied linear regression with mean absolute percentage as loss function. They found out a 6.3% error on their best model, proving a good-fit between the model estimates and the real measurements.

All these techniques have high accuracy in terms of calculating fuel consumption when no external factor comes to place. However, those methods does not take into account important factors related to traffic and human behavior, such as, number of stops and locomotive driver behavior. Moreover, there are few studies related to the heavy haul segment, which is the subject of this research. A good solution to address it would be employing ML techniques based on a data set that could include all these factors to predict the final fuel consumption.

To the best of the author's knowledge, there are few works related to the employment of ML techniques to predict fuel consumption based on trip parameters. Therefore, this will be the major impact of the present work.

3
Problem Statement

An important key performance indicator that measures a railway company regarding energy efficiency is total fuel consumption spent on a trip by total load transported times the total distance traveled also know as fuel efficiency (FE). Mathematically, this indicator is defined as

$$FE = \frac{l}{W \cdot d}, \tag{3-1}$$

where l is the total fuel used between the beginning and ending point in liters, d the total distance in kilometers and W the amount of load transported in tons. There are a variety of factors that influence fuel efficiency of a train trip, such as: train movement resistance, rail to wheel contact resistance, gravitational energy grade, brake energy, weather condition, train formation, loading, type of locomotives being used, and rail traffic. As stated in the previous section, there is no method nowadays capable to calculate accurately the fuel consumption and fuel efficiency when all these factors come together. Therefore, this work presents two major contributions:

- Create a model capable of predicting fuel efficiency accurately;
- Explain relevance of adjustable features on fuel efficiency.

On the following topic an overall explanation of how the input data is acquired will be given and also a description of each feature used in this work to predict fuel efficiency.

3.1
Data set description

The data used in this work were collected by an internal system of a Brazilian railway company during the year of 2020, in which each instance has summarized information about the complete journey conducted by an operator. Any instance represents data regard to an origin and a destination, which in this work it is considered ten distinct origins and destination pairs. The flow chart below in Figure 3.1 illustrates how the process takes place from the start of the journey given by the operator to the final database of the company.

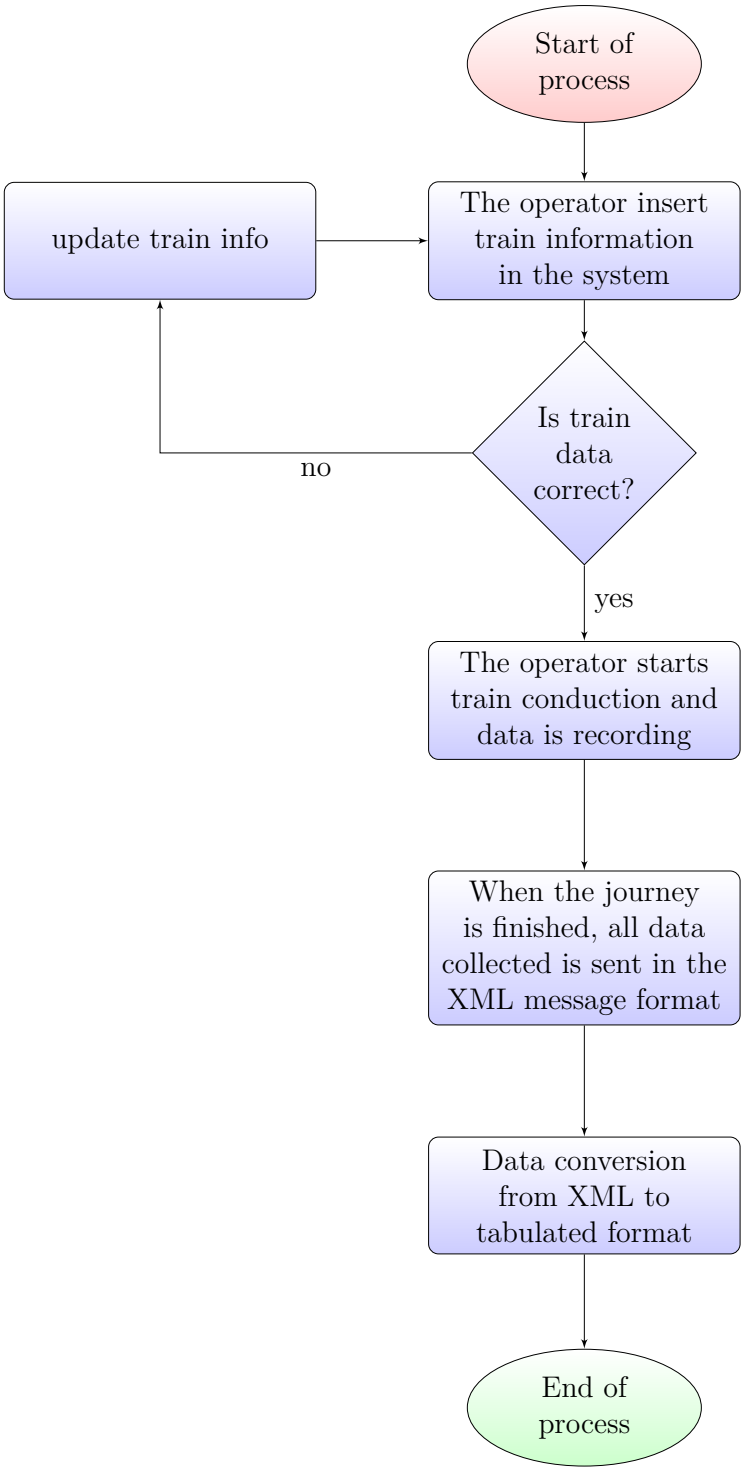

Figure 3.1: Flowchart of trip process

Therefore, for each trip (data instance), in which a locomotive driver conducts the train from the origin to the destination, a set of features related to the trip is summarized and sent to the company database. The main features of the dataset are listed and described in Table 3.1.

Feature	Description
Percentage of kilometers in auto	Total distance in semiautomatic conduction by total trip distance [%]
Run Time	Total trip time while train is in movement[h]
Stopped time	Accumulated trip time while in stopped state between the start and the end of the journey [h]
Trip Mean Velocity	Averaged trip velocity [km/h]
Trail Units	Number of locomotives trailing the lead locomotive [-]
Slope of profile	Total altitude variation by the total distance in the journey [%]
DPUS	The number of locomotives in distributed power system
Weight	Total train weight in tonnage [ton]
Length	Total length of the train [m]
Empties	Total empty wagons in train [-]
HPT	Horsepower per metric ton for this trip [hp/ton]
Lead train line hold idle distance	Distance traveled, in kilometers, while lead locomotive is in idle Mode [-]
MTO	Programmable locomotive parameter for conduction strategy [-]

Table 3.1: Description of main features of dataset used in this work

In order to better understand the features, a histogram and a discussion will be done for each of them. The fuel efficiency will be normalized by its maximum value to preserve this key data for this company.

3.1.1 Percentage of kilometers in auto

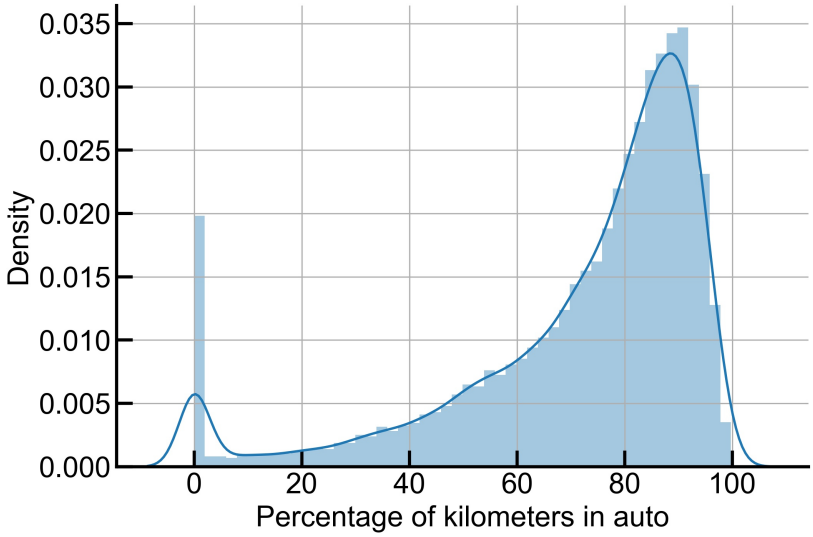

Figure 3.2: Histogram of percentage of trip kilometers in semiautomatic conduction

As illustrated in Figure 3.2, this feature is highly skewed to the right, indicating that the majority of the data has high percentage being conducted by the semiautomatic system, which is one of the company goals. There is a peak on the left disturbing the distribution that leads to a false impression of data error. However, this is represented by trip instances where the automatic system were not available or when a whole manual conduction is necessary for training proposes. An important note should be taken about this feature for better comprehension. The term semiautomatic conduction observed in Table 3.1 is used because there is an embedded system capable of conducting the train automatically. Nonetheless, there is a speed limitation to accomplish it. Therefore, part of journey is conducted manually by the train operator.

3.1.2 Run time hours

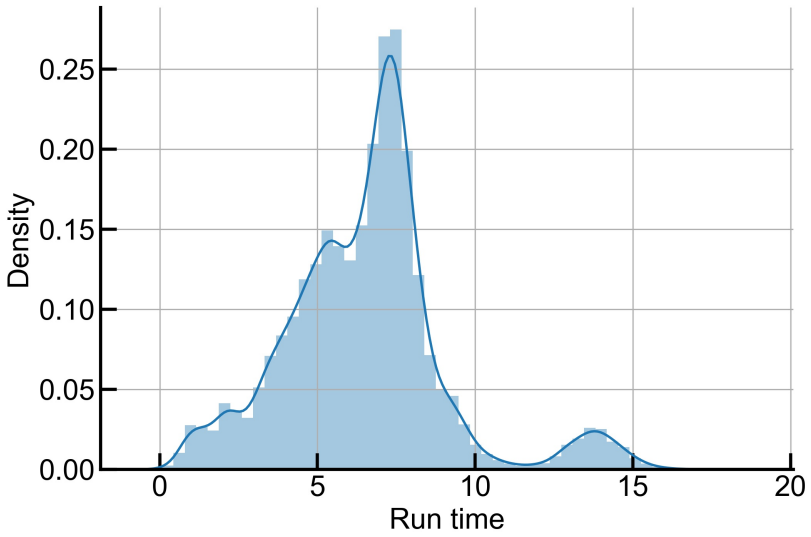

Figure 3.3: Histogram of run time

Run time hours have no distribution tendency, as illustrated in Figure 3.3, due to the dynamic scenario of trip initiation, controlled by the company operational control center. Moreover, there is a plenty of external factors that could influence on run time hour of a trip, such as, rail maintenance, rail speed restrictions, terminal queue or even systems failures.

3.1.3 Stopped time

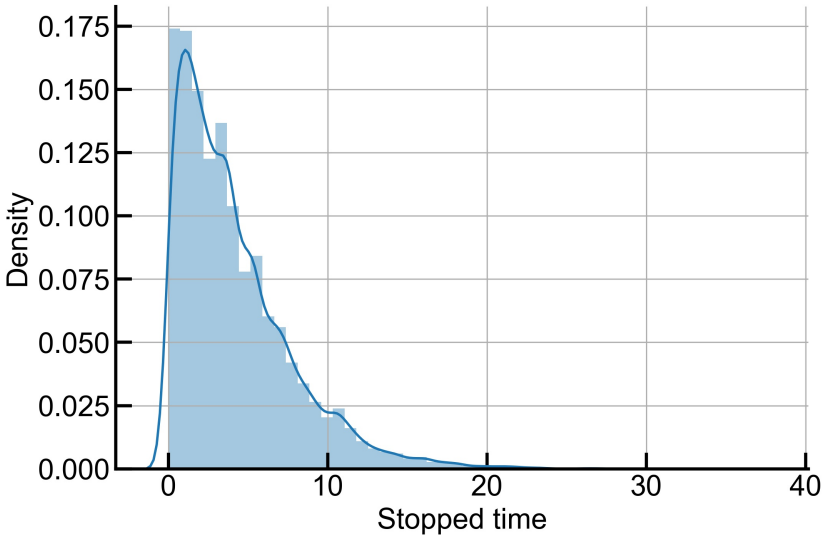

Figure 3.4: Histogram of stopped time

As shown in Figure 3.4, the stopped time has a highly skewed distribution to the left, indicating a good operational distribution. One of the major efforts in operation is to reduce stopped time.

3.1.4 Trip mean velocity

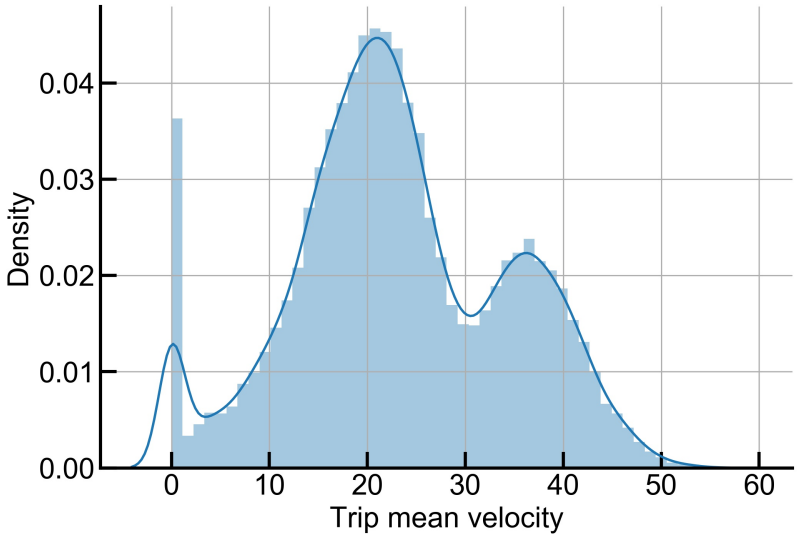

Figure 3.5: Histogram of trip mean velocity

Figure 3.5 has two peaks characterized by regions where the maximum allowable velocity correlates with the values showed. Regions with higher trip mean velocities can also be interpreted as rural areas where a higher speed limit is allowed.

3.1.5 Trail Units

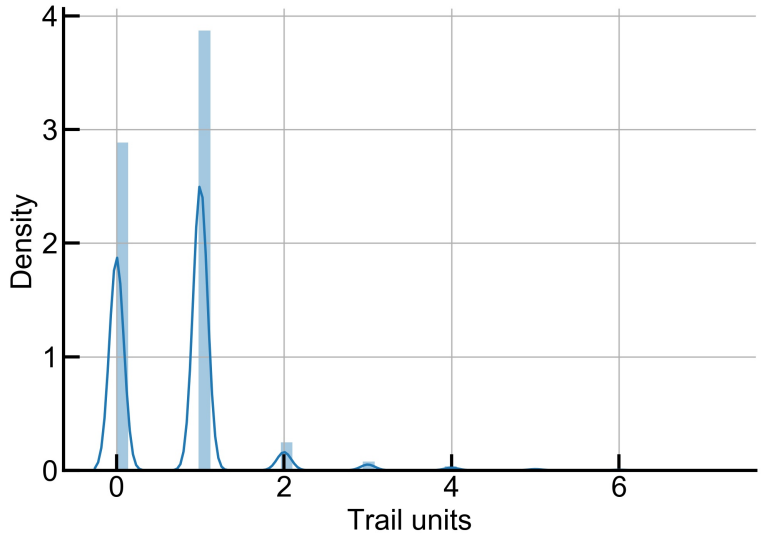

Figure 3.6: Histogram of number of trail units

Distribution of trail unit locomotive is represented by Figure 3.6 as a discrete variable. This parameter is related to train configuration and can be associated with train length and weight. Therefore, light trains will be located on the left side of this graph, while heavy ones will follow the opposite direction.

3.1.6 Slope of profile

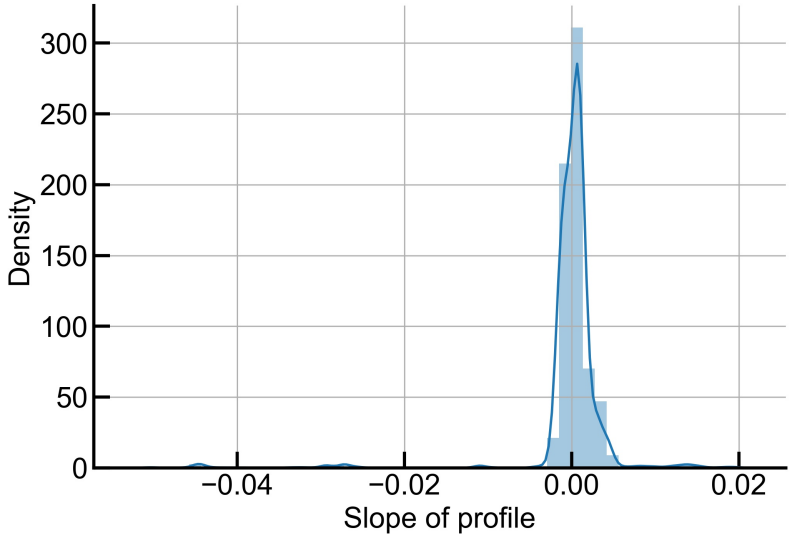

Figure 3.7: Histogram of slope of profile

The slope of profile is mainly distributed between 0.006 and -0,006 (Figure 3.7), which indicates a predominance of a flat track profile.

3.1.7 DPUS

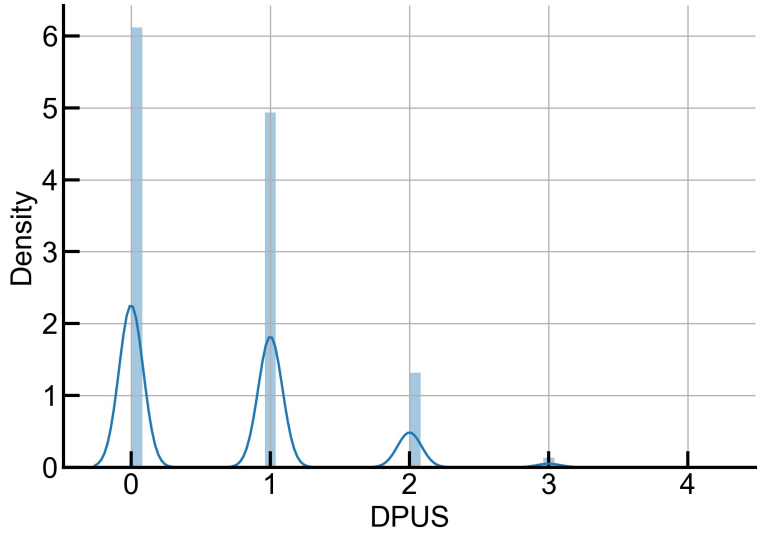

Figure 3.8: Histogram of number of locomotives in distributed power

Distribution data of the total number of distributed power units in a train is revealed in Figure 3.8. As it can be seen, there is a major predominance of trains using this system (DPUS > 0). Distributed power system is a vital tool to form long trains, therefore, these two parameters tend to be high correlated.

3.1.8 Train weight

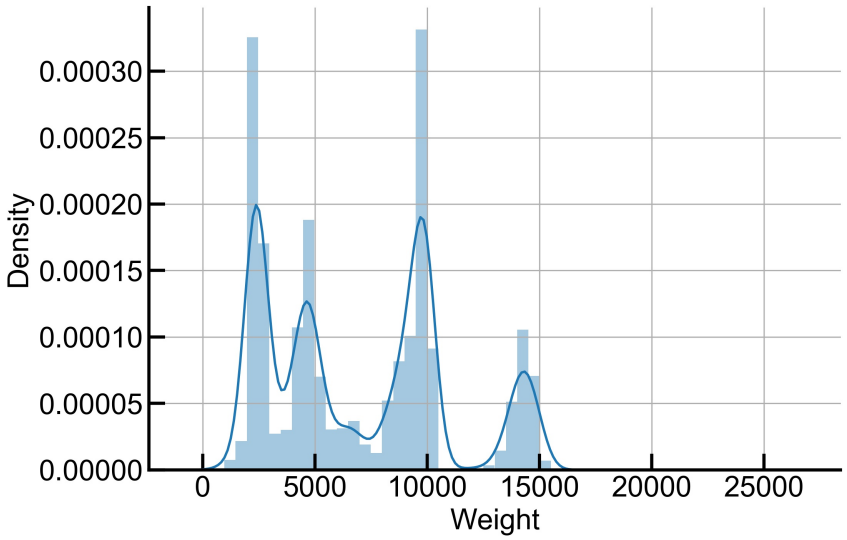

Figure 3.9: Histogram of Train weight

The train weigh distribution chart is showed in Figure 3.9. It is noticed that there is four peaks, representing the majority of the train types of the company.

3.1.9 Length

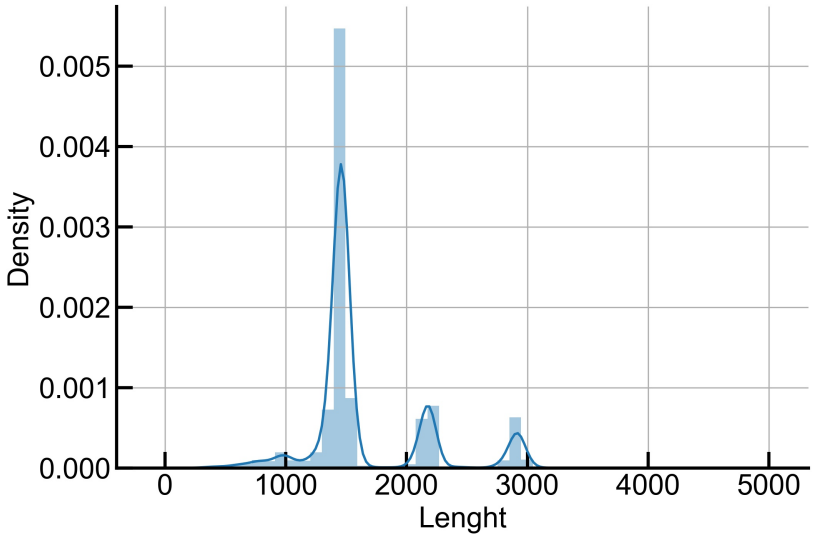

Figure 3.10: Histogram of train length

In the same way, there are three peaks for train length (Figure 3.10) which represent common train length practiced in 2020.

3.1.10 Empties

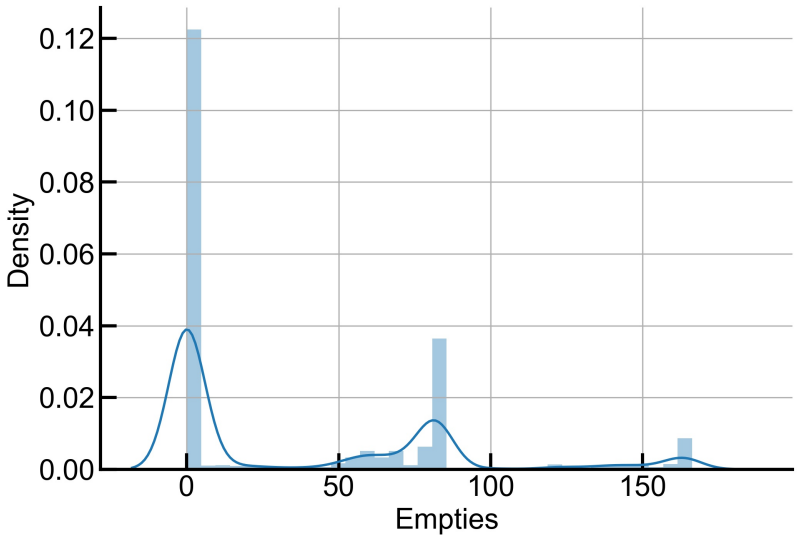

Figure 3.11: Histogram of number of empty wagons

The distribution chart for number of empties wagons in a train is presented in Figure 3.11. The first peak considers loaded trains and the other common train types for unloaded trips.

3.1.11 HPT

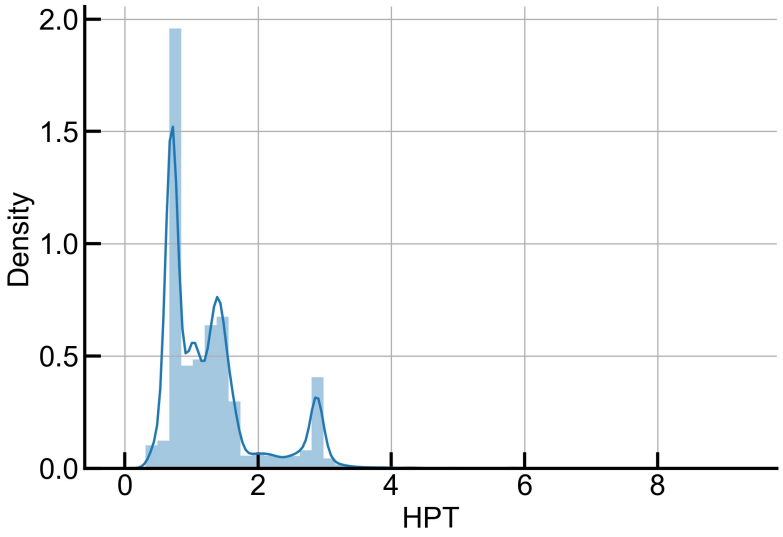

Figure 3.12: Histogram of horse power per tonnage

HPT distribution chart is showed in Figure 3.12. This is an important operational parameter because measures how much power is needed to transport one ton. Therefore, a highly skewed to the left would be a good signal as it can be seen above.

3.1.12 Lead train line hold idle distance

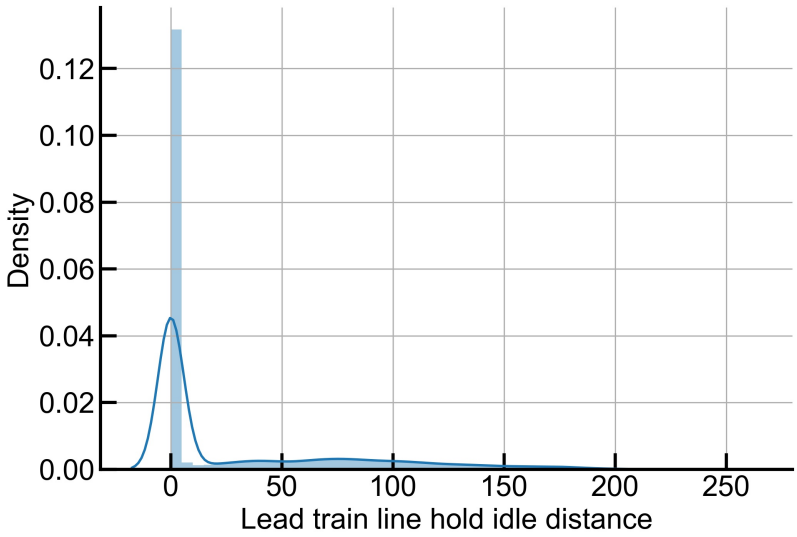

Figure 3.13: Histogram of lead train line hold idle distance

As described in Table 3.1, this parameter sums the total distance traveled while lead locomotive was in idle state. This state can be seen as a low consumption rate in which up to 98% of fuel can be saved when compared to the highest consumption rate state. Therefore, increasing this parameter locomotive will be using less fuel and consequently a drop in fuel efficiency will be noticed.

3.1.13 MTO

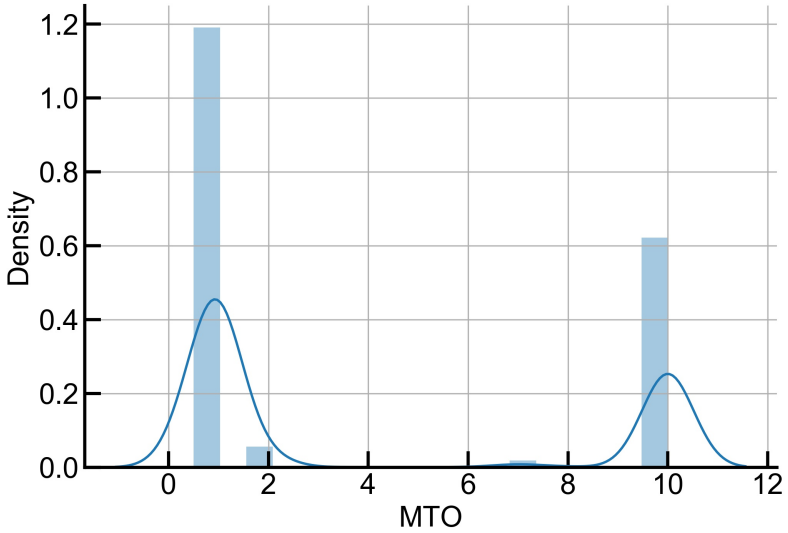

Figure 3.14: Histogram of MTO

As describe in Table 3.1, MTO is a programmable parameter in which an operation strategy ranging from 0.5 to 10 can be selected to save fuel, where 10 is the maximum fuel saving strategy value.

3.1.14 Fuel Efficiency

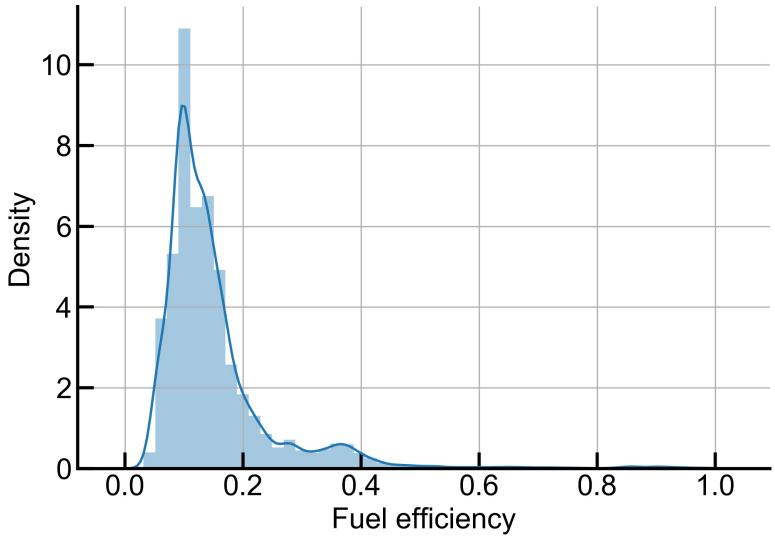

Figure 3.15: Histogram of normalized fuel efficiency

Finally, the normalized fuel efficiency distribution plot, the target parameter of this work, is presented in Figure 3.15. A highly skewed distribution area to the left side of the curve is observed. A correlation matrix will be displayed and discussed in the next section to know all input features on fuel efficiency.

As means to explain the relevance of adjustable features on fuel efficiency (second goal of this work), the impact of four manageable variables will be studied on the following chapters. These variables are considered crucial by railway specialists, and therefore, local effects of them on fuel efficiency is necessary. These four variables are: trip mean velocity, stopped time, MTO, and HPT.

3.2 Data set correlation matrix

As means to have an initial intuition about the behavior of each independent variable on the dependent variable, a correlation matrix will be calculated based on Spearman's rank correlation coefficient. According to Hinkle [15], this correlation coefficient is appropriate when variables are skewed or ordinal and robust when extreme values are present. A monotonic behavior can be estimated and used as a reference by this correlation as it was stated as an initial

goal. For a correlation between dependent and independent variables, the formula for calculating the sample Spearman's correlation coefficient (ρ) is given by:

$$\rho = \frac{6 \sum d_i^2}{n(n^2 - 1)}, \tag{3-2}$$

where d_i is the difference in paired ranks and n is the number of instances.

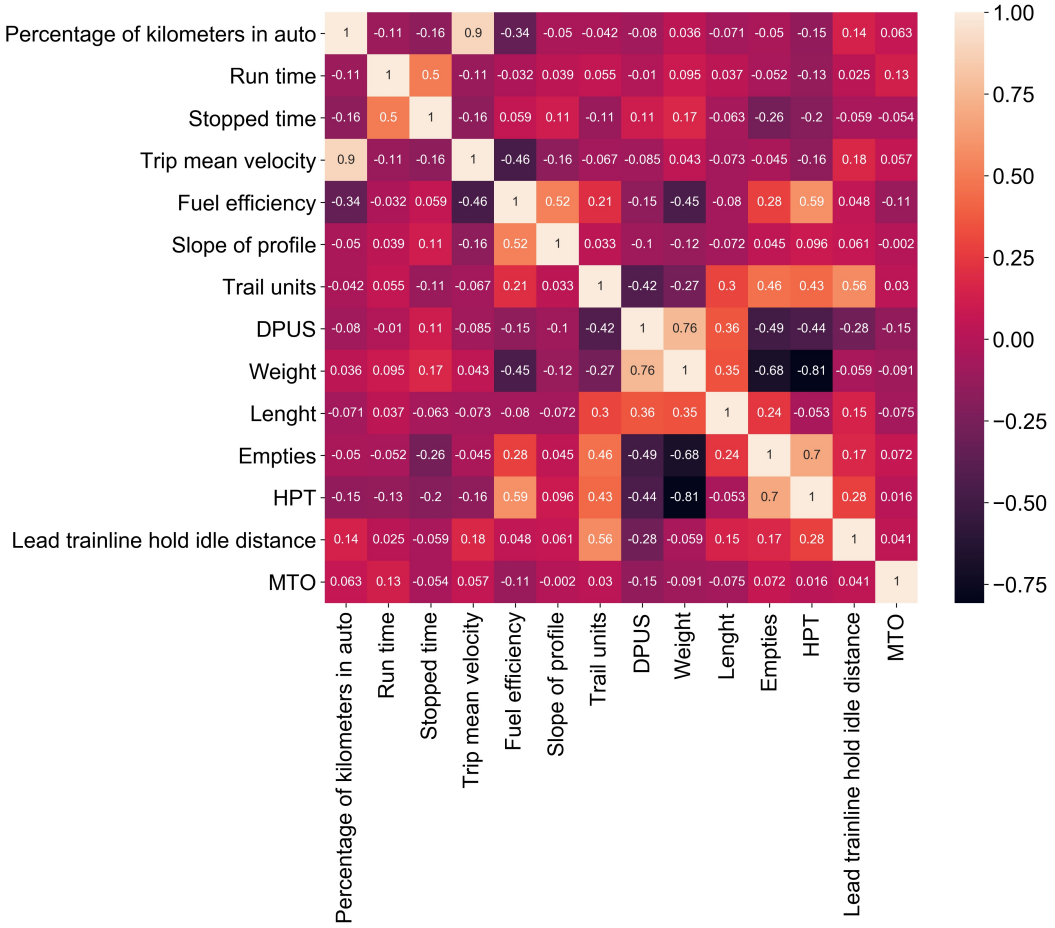

Figure 3.16: Spearman's correlation coefficient matrix

Figure 3.16 shows the Spearman correlation matrix of the data set. Analyzing it, four major observations can be made about the strength of each dependent variable on the independent variable. The first one would be a low negative correlation between fuel efficiency and three variables: percentage of kilometers in auto, trip mean velocity, and train weight. This is because the coefficient correlation between those variables and fuel efficiency is between -0.3 and -0.5, as stated by Hinkle [15]. Consequently, when there is an increase in these variables, a decrease in fuel efficiency can be expected. Second, there is a low positive (0.3 to 0.5) correlation between the independent variable and two dependent variables: trail units and the number of empty wagons. Therefore, when there is an increment in these variables, the same behavior

in fuel efficiency is expected. Third, a moderate positive correlation (0.5 to 0.7) between two features and fuel efficiency is observed: slope of profile and horsepower per tonnage (HPT). A consequence of this relation would be a moderate increase in fuel efficiency when there is an increment in these variables. The last observation would be a negligible correlation between fuel efficiency and the other variables, which indicates that there is no major impact on fuel efficiency when a change is made in these variables.

4
Methodology

This chapter discusses all methods used in this work to predict and explain fuel efficiency. Before modeling the problem using ML algorithms, a data pre-processing step will be placed. Based on the literature review presented in Chapter 2, it will be applied two ML models, which are RF and ANN. In order to improve predictions for both models and discard overfitting issues, a Bayesian Optimization and a cross validation technique will be discussed and used. Finally, accumulate local effects plots will be detailed as means to explain crucial feature's impacts on the target variable.

4.1
Data pre-processing

As a first step, it is necessary to remove outliers from the whole dataset generated by the process exemplified in flow chart illustrated in Figure 3.1. This is a important step before feed the models with the data, due to the necessity to remove instances that were affected by system errors. Therefore, the first part of this chapter will focus on the algorithm developed to remove outliers presented in the raw database. As reported by Prasad and Krishna [16], normal data objects follow a generating mechanism and abnormal objects deviate through this generating mechanism. Hence, considering a normal dataset distribution, an algorithm aiming to remove outlier values can compute parameters premising that data points are set up by such a distribution (mean and standard deviation). Before entering the algorithm itself, it is essential to take a nonparametric test to evaluate if the variables follow a normal distribution. As means to accomplish it, the Kolmogorov-Smirnov test will be used. Statistically, this test measures the maximum distance between the empirical cumulative distribution of the sample variable and the cumulative distribution function of a reference distribution (in this case, the normal distribution), according to Daniel [17]. Mathematically, it is expressed by:

$$D_n = \sup_x |F_n(x) - F(x)|, \tag{4-1}$$

where D_n is KS-statistic, and $F_n(x)$ and $F(x)$ are the cumulative distribution function of the empirical and reference, respectively. Therefore, accord-

ing to a significance level τ (usually taken as 0.05), the null and alternative hypotheses can be defined:

$$\begin{cases} H_0 : \text{variable follows a normal distribution} \\ H_a : \text{variable does not follow a normal distribution,} \end{cases} \tag{4-2}$$

in which, if $D_n < \tau$, the null hypotheses is maintained.

After confirming that the variables are normally distributed, the outliers can be removed removed following the same approach used in [16], where outliers are defined as points with low probability of occurrence, or deviate 3 times the standard deviation from the mean of a particular variable. The algorithm outlined below explain how this strategy was defined.

Algorithm 1 Removal of outliers from dataset

Data: X^d,O,D,I
Result: H
$R \leftarrow Unique(O, D)$;
$m \leftarrow Length(X^d)$;
for *i in R* **do**
  $\mu_i \leftarrow Mean(X^d_{R=i})$;
  $\sigma_i \leftarrow Std(X^d_{R=i})$;
  $U_i \leftarrow \mu_i + 3\sigma_i$;
  $L_i \leftarrow \mu_i - 3\sigma_i$;
end
for *j in m* **do**
  if $X^d_j < U_{R=R_j}$ **and** $X^d_j > L_{R=R_j}$ **then**
    $H \leftarrow I \cup I_j$;
end

As a first step, variables used in the algorithm are defined appropriately: X^d as vector of the desired feature to remove outliers, O as origin vector, D as destination vector and finally I as data index vector. Following, an unique tuple vector R is defined to store single values containing pairs of origin and destinations that will be used on the first iteration. Then, length of database is stored as m for the second loop. The main objective of the first iteration is to calculate upper and lower desired feature thresholds for each distinguish pair of origin and destination. Once defined the thresholds, they are used on the second loop in order to return all the indexes H that rely on these values.

4.2 Random Forest

Random Forest regressor is part of ensemble methods which uses a variety of estimators with a defined learning algorithm aiming to improve robustness over a single estimator [18]. Regarding ensemble methods there are two major methods: averaging/bagging and boosting. The first one is based on the subject of raising a set of estimators independently and then averaging their results. Finally, the combined estimator is usually better than any of the single base estimator due to the reduce in variance [18]. Beside averaging, boosting are known by the sequential base estimator generated in a way that improves the bias of the previous estimator in the sequence. The main idea is combining several weak estimators to formulate an improved predictor. In this context, RF regressor is considered as an averaging method of decision trees.

4.2.1 Decision Tree

DT are ML models that use input data space to create a series of data segregation/decision to predict the target value, Pedregosa at el. [19]. Therefore, for each node (decision point) there is a threshold value based on a specific variable that will divide the data in two sub-space. With the objective to illustrate this step, part of a DT, containing the first three depths (or steps), is depicted on Figure 4.1.

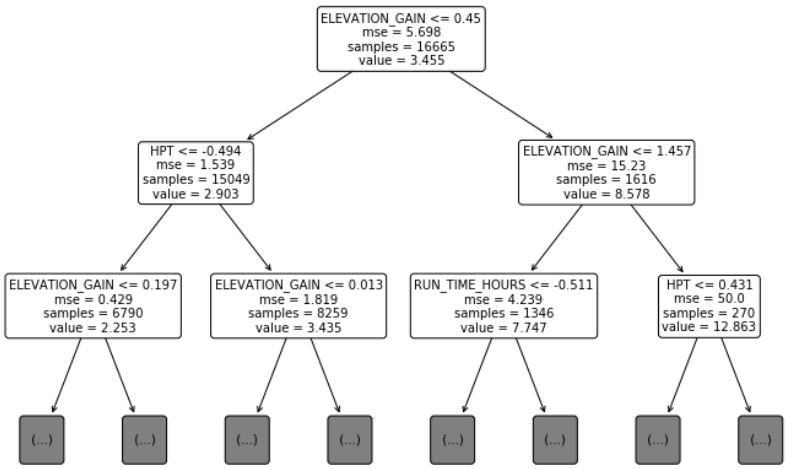

Figure 4.1: First three steps of a DT

Analyzing Figure 4.1, the first node segregates the input space into two major groups: data instances in which elevation gain variable are less than or equal to 0.45 and the data instances in which elevation gain are greater than 0.45. The input space of the former one will then be evaluated again in another decision, but now checking if the elevation gain is less than or equal to 1.457. This process continues until reaches a criteria that will be discussed in the mathematical formulation section. As can be seen, for each divided group, there is a quantity of data samples, mean squared error (mse) and also predicted mean values (value).

4.2.1.1 Mathematical formulation

Considering a training vector denoted by $\{x_i|x_0, x_1, x_2, x_3, ..., x_B\} \in \mathbb{R}^B$, and label vector $y \in \mathbb{R}^K$, where B and K are the size of these vectors, a decision tree recursively separates the feature domain such that instances with similar target values are grouped. Considering the data at node m defined by Q_m with N_m samples, accoring to Pedregosa at el. [19], for each possibility division $\theta = (j, t_M)$ in a feature j and threshold t_M, there will be Q_m^{left} and Q_m^{right} such that:

$$Q_m^{left}(\theta) = \{(x, y)|x_j \leq t_m\}, \tag{4-3}$$

and

$$Q_m^{right}(\theta) = Q_m / Q_m^{left}. \tag{4-4}$$

The quality of a possible segregation in node m is then calculated as:

$$G(Q_m, \theta) = \frac{N_m^{left}}{N_m} J(Q_m^{left}(\theta)) + \frac{N_m^{right}}{N_m} J(Q_m^{right}(\theta)), \tag{4-5}$$

in which a loss function $J(\cdot)$ is defined by the mean squared error (MSE) as:

$$J(Q_m) = \frac{1}{N_m} \sum_{y \in Q_m} (y - \bar{y_m})^2, \tag{4-6}$$

in order to select the best parameters that minimize the loss function, as:

$$\theta^* = \arg\min_{\theta} G(Q_m, \theta). \tag{4-7}$$

Finally, an iteration step takes place for subsets $Q_m^{left}(\theta)$ and $Q_m^{right}(\theta)$ until the maximum allowable depth is reached, $N_m < \min_{samples}$ or $N_m = 1$.

4.3 Artificial Neural Networks

ANN or multi-layer perceptron are ML algorithm that determine a function $f : \mathbb{R}^B \rightarrow \mathbb{R}^K$ through a training set with dimensions B and K for input and output respectively. ANN is capable of modeling non-linear problems by connections between input, hidden layers and output, as can be seen on Figure 4.2.

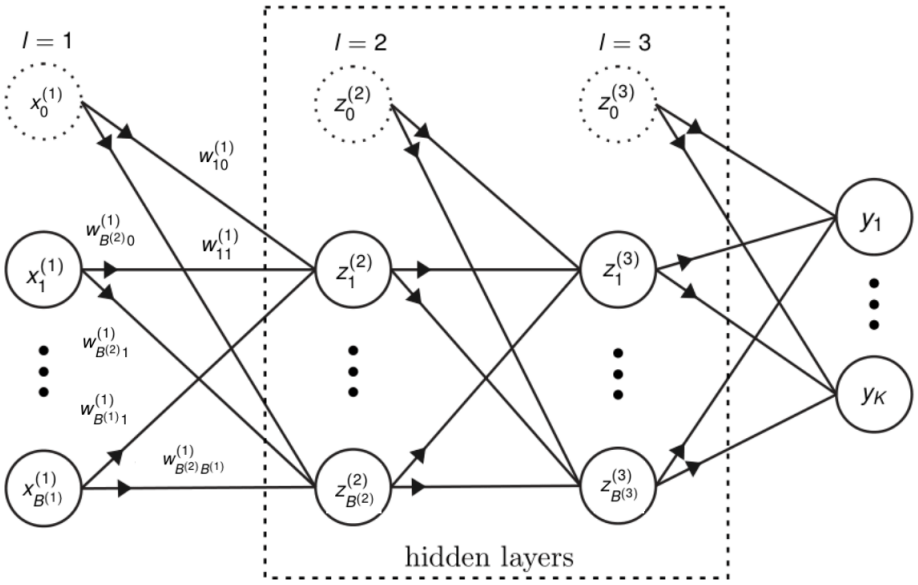

Figure 4.2: Example of a neural network with two hidden layer

The first layer $\{x_i|x_0, x_1, x_2, x_3, ..., x_B\}$ on a ANN is named input layer and has $m+1$ dimension where x_0 is the bias term and m number of features. According to Bishop [20], bias term allows for any fixed offset in the data and is commonly addressed as unit value. On the first hidden layer, each neuron indexed by $j = 1, 2, 3, ..., M$ converts values from the input layer with a weighted linear summation and are named activations:

$$a_j = \sum_{i=1}^{B} w_{jk}^{(1)} x_i, \tag{4-8}$$

these activations followed by a non-linear activation function $g : R \rightarrow R$ then formulates the hidden units:

$$z_j = g(a_j). \tag{4-9}$$

In the beginning of the process, the weights are randomly initialized in a feedfoward propagation following a uniform or normal distribution, as stated by Bishop [20], and then the error is calculated by a loss function, such as the mean squared error, given by:

$$J(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left[y(\mathbf{x},\mathbf{w}) - y_n\right]^2. \tag{4-10}$$

In order to minimize the loss function, an optimization method is applied to adjust the weights. Generally, the most common method for ANN is the backpropagation as will be described in the mathematical formulation section. Finally, the output layer receives activation values from the last hidden layer containing k neurons and transforms them into an output value as described:

$$y(\mathbf{x},\mathbf{w}) = \sigma\left(\sum_{k=1}^{K} w_{jk}^{l} z_k^{l-1}\right), \tag{4-11}$$

where σ is an activation function for the output layer. In this present work, a variety of activation functions were tested in the hyperparameter tunning process in order to select the best values for the training process.

4.3.1 Activation functions

The type of activation function determines the accuracy of the ANN model studied and, therefore, is considered a hyperparamter to be refined. By the nature of this work, where it will be employed more then one hidden layer, and considering the problem of the vanishing gradient addressed by Glorot et al. [21], three activation functions will be employed: linear, rectified linear and exponential linear. The choice of these functions is focused on the benefits of non-saturation of the learning process. This problem is often observed on sigmoid and tangent hyperbolic functions where there is a region where a saturation is observed when calculating the gradient of the loss function.

Linear function The linear activation function, also known as the identity function, is defined by following expression:

$$\sigma_{linear}(x) = x,$$

for $x \in \mathbb{R}$.

Rectifier linear function (ReLu)

The ReLu function is a piecewise linear function defined as expression:

$$\sigma_{ReLu}(x) = \begin{cases} 0 & x < 0 \\ x & x \geq 0, \end{cases}$$

for $x \in \mathbb{R}$.

Exponential linear function (ELu)

The ELu function are also piecewise but with an exponential decay on the left side, i.e.:

$$\sigma_{ELu}(x) = \begin{cases} e^x - 1 & x < 0 \\ x & x \geq 0, \end{cases}$$

for $x \in \mathbb{R}$.

4.3.2 Mathematical formulation

As stated by Nielsen [22], the backpropagation algorithm is the main driver of learning in ANN, therefore, this section will focus on the mathematical formulation. The main goal is to minimize the error of the loss function and this can be achieved by finding a weight matrix such that $\nabla J(\mathbf{w}) = 0$. However, finding the analytical solution of this equation is not possible due to its complexity. The task of calculating and storing the entire Hessian matrix $(J(w))$ for functions such as the loss functions of neural nets takes an infeasible amount of memory which leads to numerical solutions, as reported by Bishop [20]. Nonetheless, numerical procedures for addressing these problems are available and optimization of continuous nonlinear functions is a widely studied problem. Many algorithms concern to select initial values for the weight matrix and then moving to the search space in a succession of steps such that

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \Delta\mathbf{w}^{(\tau)}, \tag{4-12}$$

in which τ stands for the iteration steps or epochs. Different algorithms concern a variety of choices for the weight vector update $\Delta\mathbf{w}^{(\tau)}$. There are simple methods, such as steepest descent or gradient descent, that use first-order information, in order to take a small step in the opposite direction so that

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta\nabla J(\mathbf{w}^{\tau}), \tag{4-13}$$

consequently, evaluation of $\nabla J(\mathbf{w})$ at each iteration step is required as well as setting a learning rate η for this optimization problem . Gradient computation for the whole dataset at each iteration requires a huge amount of memory when dealing with high dimensional problems. As means to overwhelm this issue, stochastic gradient descent technique is widely used, achieving rapid iterations in trade for a small convergence rate [23]. This technique is based on

setting a batch size, or number of examples for training, and randomly shuffle instances in order to calculate Equation 4-13.

Significant improvements on the basic stochastic gradient descent algorithm have been incorporated, especially in the ML field. In this area, the optimal value of the learning rate parameter (i.e., the step size) is not necessarily computed in each step of the process. Tuning this hyperparameter too high can cause the algorithm to diverge, on the other hand configuring it too low leads to a slow convergence [24]. Hence, three different adaptive learning methods were used in this work through TensorFlow library [25], an open source framework implemented in Python to run and execute ML algorithms.

Adam

Adam stands for Adaptive Moment Estimation and the idea behind this algorithm is to use past information about the gradient similar to the psychics concept of momentum. *Adam* follows the dynamics of a heavy ball with friction and thus prefers flat minima in the objective landscape [26]. The method uses the moving averages of gradients and squares of the gradients to update weight parameters. These averages are initialized as zero, and are calculated as

$$\mathbf{m}^{\tau} = \beta_1 \mathbf{m}^{\tau-1} + (1-\beta_1)\nabla J(\mathbf{w}^{\tau}), \tag{4-14}$$

$$\mathbf{v}^{\tau} = \beta_2 \mathbf{v}^{\tau-1} + (1-\beta_2)\nabla J(\mathbf{w}^{\tau})^2, \tag{4-15}$$

where $\mathbf{m}_{\tau}$ and $\mathbf{v}_{\tau}$ are computations of first and second moment of the gradient, respectively. In order to overcome bias factor toward zero and initialization setups for the decay factors β_1 and β_2, Kingma and Adam [27] suggest calculating bias-corrected first and second moment:

$$\hat{\mathbf{m}}^{\tau} = \frac{\mathbf{m}^{\tau}}{1-\beta_1^{\tau}}, \tag{4-16}$$

and

$$\hat{\mathbf{v}}^{\tau} = \frac{\mathbf{v}^{\tau}}{1-\beta_2^{\tau}}, \tag{4-17}$$

which allows the following update rule for Adam:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{\tau} - \frac{\eta}{\sqrt{\hat{\mathbf{v}}^{\tau}} + \epsilon}\hat{\mathbf{m}}^{\tau}, \tag{4-18}$$

where 0.9, 0.999 and 10^{-8} are common values for β_1, β_2 and ϵ [20].

Adadelta

Adadelta is an improved version of adaptive subgradient method Adagrad which main goal is to allocate low learning rates for features with frequently appearance and high learning rates for features with seldom appearance

[28]. Instead of adding past squared gradients, Adadelta narrows the window of accumulated past gradients to some settled size. With this windowed accumulation, the denominator of Adagrad cannot accumulate to infinity and instead becomes a local estimate using new gradients. This guarantees that learning process keeps to make advancements even tough past iterations of updates have been done, conforming to Zeiler [29]. In terms of storing preceding, squared gradients is ineffective. This happens because there is an aggregation of the exponentially decaying average of the squared gradients. Consider at each epoch τ the running average is $E[\nabla J(\mathbf{w}^{\tau})^2]$, then it can be computed:

$$E[\nabla J(\mathbf{w}^{\tau})^2] = \rho E[\nabla J(\mathbf{w}^{\tau-1})^2] + (1-\rho)\nabla J(\mathbf{w}^{\tau})^2, \tag{4-19}$$

where ρ is a decay constant analogous to that used in the *Adam* optimization. Since it is required the square root of this quantity in the parameter updates, this effectively becomes the root mean square (RMS) of past squared gradients up to epoch τ

$$RMS[\nabla J(\mathbf{w}^{\tau})] = \sqrt{E[\nabla J(\mathbf{w}^{\tau})^2] + \epsilon}. \tag{4-20}$$

Since the left hand side of equation 4-20 is unknown, it will be approximated with the RMS of parameter updates until the previous time step. Replacing the learning rate η in the past update rule with $RMS[\nabla J(\mathbf{w}^{\tau-1})]$, finally produces the *Adadelta* update rule:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{\tau} - \frac{RMS[\nabla J(\mathbf{w}^{\tau-1})]}{RMS[\nabla J(\mathbf{w}^{\tau})]}\nabla J(\mathbf{w}^{\tau}), \tag{4-21}$$

RMSProp

RMSProp stands for Root Means Square Propagation and was proposed by Hinton et al. [30]. It is identical to *Adadelta* on the first update vector and has the same concept of manipulating the running average of previous values in the optimization process:

$$E[\nabla J(\mathbf{w}^{\tau})^2] = 0.9E[\nabla J(\mathbf{w}^{\tau-1})^2] + 0.1\nabla J(\mathbf{w}^{\tau})^2, \tag{4-22}$$

Therefore, update rule for *RMSProp* can be defined:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{\tau} - \frac{\eta}{\sqrt{E[\nabla J(\mathbf{w}^{\tau})^2] + \epsilon}}\nabla J(\mathbf{w}^{\tau}), \tag{4-23}$$

Once defined the appropriate algorithm to update the weight matrix, it is required to find an adequate approach for evaluating the gradient of an error function $J(\mathbf{w})$ for a feed-forward ANN. This procedure is achieved sending the updating message forward and backward through the network and is known

as backprogagation [20].

4.3.3 Early stopping and dropout

ANN models such, as the fully connected multi-layer perceptron, frequently have an extensive space parameter. This large parameter space can achieve up to millions of neurons in image recognition problems and this could led to a common problem in training step, known as overfitting [31]. According to Everitt [32], overfitting is the production of an analysis that corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably. In order to avoid overfitting, two techniques were applied when training the ANN: dropout and early stopping.

Early Stopping

A typical curve used to check how a neural network is performing over the training epochs is showed on Figure 4.3.

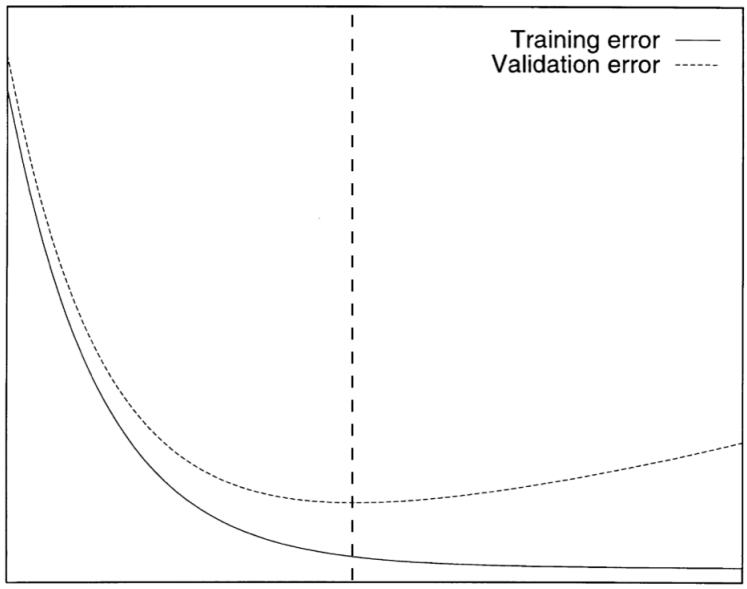

Figure 4.3: Training and validation curves tendency adapted from [33]

Early stopping is used frequently because it is simple to comprehend, implement and has been recognized to be superior to regularization methods, such as weight decay methods as was proved by Finnoff and Hergert [34]. On the present work, a stopping criteria relying on the sign of the changes in the generalization error (validation curve) will be implemented. Therefore, an early stop will be achieved when the following criteria is satisfied for $s = 1, 2, 3, ..., S$:

$$J_{va}(\tau) > J_{va}(\tau - s), \tag{4-24}$$

where S is a the patience value to be defined, J_{va} is the validation error calculated by the loss function 4-10 and epoch τ.

Dropout

Another method to prevent overfitting is the dropout. As reported by Srivastava et al. [35], refers to dropping out units (hidden and visible) in ANN by removing units, temporarily, along with all its previous and outgoing connections, as can be seen on Figure 4.4 and 4.5. Decision of which neuron to drop is defined randomly. In the simplest case, each unit is retained with a fixed probability independent of other units, where it can be chosen using a validation set. Arbitrarily, a simply probability rate is set at 0.5, which seems to be approximated to optimal for a variety range of problems. Nonetheless, for the input layer, the optimal probability of retention is usually closer to 1 other than to 0.5.

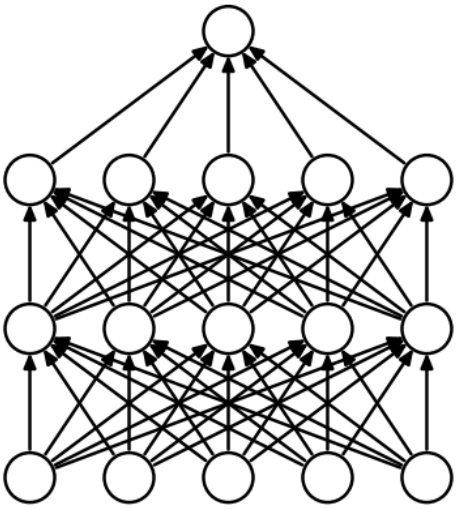

Figure 4.4: Dense connected ANN [35]

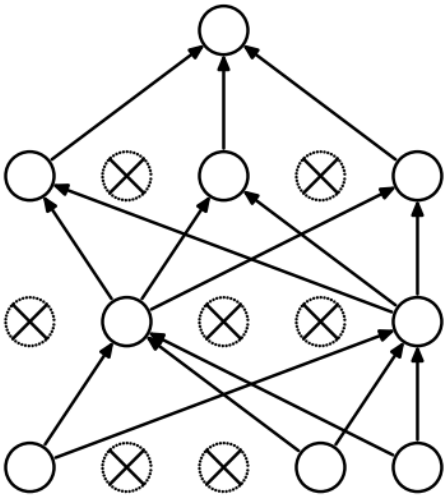

Figure 4.5: ANN after dropout [35]

In their paper, Srivastava et al. [35] applied dropout on five different databases of image recognition in order to understand the results from a diverse range of data. Results showed an improvement up to 43.48 % when applying this technique.

4.4 Hyperparameter Optimization

Hyperparamter optimization or tunning is an important part of the training process, since calibrating these values leads to performance improvements on the model. Mathematically can be seen as a nonlinear optimization such that

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in E} f(\mathbf{x}), \tag{4-25}$$

where E is the search space for the hyperparamters $\mathbf{x}$, $\mathbf{x}^*$ is an optimal hyperparamter and $f(\mathbf{x})$ is the model function. Pinto et al. [36] and Coates and Lee [37] demonstrate that the challenge of hyperparameter optimization in high-dimensional multilayer models is a direct barrier to scientific development. Since in many situations this part of the machine learning pipeline is handily or randomly treated. Hence to overcome this challenge, Bergstra et al. [38] proposed a technique named Tree of Parzen Estimators (TPE) to automatically select hyperparameter values based on defined distribution of them. TPE are Sequential Model-Based Global Optimization (SMBO) algorithms that have been employed in many situations where evaluation of the output function is costly [39] [40]. Therefore, a surrogate function (probability model) is approximated to $f(x)$ achieving a cheaper evaluation process. As a first step Bergstra et al. [41], a probabilistic regression model Q is initiated benefiting from part of domain E. Then, new values inside the domain are generated by optimizing an acquisition function S that uses the current model as a less expensive surrogate for the $f(x)$. For each iteration observed on the pseudo-code illustrated below, there is an evaluation of the result that is stored in the historical set $F = (x_1, y_1), ..., (x_i, y_i)$ which is used to upgrade the regression model for developing the next proposal.

Algorithm 2 SMBO

Data: f, E, S, Q
Result: F
$F \leftarrow InitSamples(f, E)$;
for $i \leftarrow |F|$ *to* T **do**
  $p(y|x, F) \leftarrow FitModel(Q, F)$;
  $x_i \leftarrow \arg\max_{x\in E} S(x, p(y|x, F))$;
  $y_i \leftarrow f(x_i)$;
  $F \leftarrow F \cup (x_i, y_i)$;
end

Once defined the routine to select the best hyperparamters based on SMBO, it is necessary to select a probabilistic regression model (Q). As proved

by Bergstra et al. [38] TPE showed better results when compared to other Bayesian optimization models and, for this reason, it will be the focus of this work. TPE defines domain variables when objective function is portioned between a threshold y^*

$$p(\mathbf{x}|y, F) = \begin{cases} l(x) & y < y^* \\ g(x) & y \geq y^*. \end{cases}$$

According to Bergstra et al. [41], with these two distributions, one can optimize a closed form term proportional to acquisition function S, that in this case is the expected improvement:

$$EI_{y^*}(x) = \int_{-\infty}^{y^*} (y^* - y) \frac{p(x|y, F)p(y)}{p(x)} dy, \tag{4-26}$$

defining $\gamma = p(y < y^*)$ and $p(x) = \int p(x|y, F)p(y)dy = \gamma l(x) + (1 - \gamma)g(x)$, it can be proved that

$$EI_{y^*}(x) = \frac{\gamma y^* l(x) - l(x) \int_{-\infty}^{y^*} p(y)dy}{\gamma l(x) + (1 - \gamma)g(x)} \propto \left(\gamma + \frac{g(x)}{l(x)}(1 - \gamma)\right)^{-1} \tag{4-27}$$

The right hand side of Equation 4-27 demonstrates that maximizing improvements is equal to selecting points with high probability under $l(x)$ and low probability under $g(x)$.

4.5 K-Fold cross-validation

When there is no segregation between training and testing datasets, the predicting model tends to overfit [19]. A common mistake in the training process when there is no segregation of the dataset for training and testing is called overfitting. This happens when the model has no capability of generalization. In other words, the ML algorithm can not predict correctly new instances based on the learning parameters used. To prevent it, an usual practice when performing a supervised ML experiment is holding out part of the available data as a test set in a randomly process. Therefore, for this work, 80% of the dataset will be hold for training proposal while the remaining will be hold for testing.

During evaluation process of different hyperparameters for the prediction models, such as the weight matrix $\mathbf{w}$ in ANN, there is still a risk of overfitting on the test set because the parameters can be tweaked until the estimator performs optimally. This way, knowledge about the test set can "leak" into the model and evaluation metrics no longer report on generalization performance

as reported by Pedregosa et al. [19]. To address this situation, another part of the dataset should be held out as a validation set. Therefore, the whole process would be:

- Training these values in the training dataset to return improved learning parameters;
- Evaluating hyperparameters settings on the validation dataset in order to return the best values;
- Testing optmal learning parameters in the test dataset to return final model.

Nonetheless, segregating the available data into three sets diminishes the number of instances which can be used for learning the model, and the results can rely on a specific random choice for the pair of train and validation sets.

A solution for this would be applying cross-validation. In this procedure, a test set should still be held out for final evaluation, but the validation set is not necessary when doing cross-validation. In the basic approach, called k-fold cross-validation, the training set is split into k smaller sets. The following procedure is followed for each of the folds:

- A model is trained using $k - 1$ of the folds as training data;
- The final model is validated on the remaining part of the data.

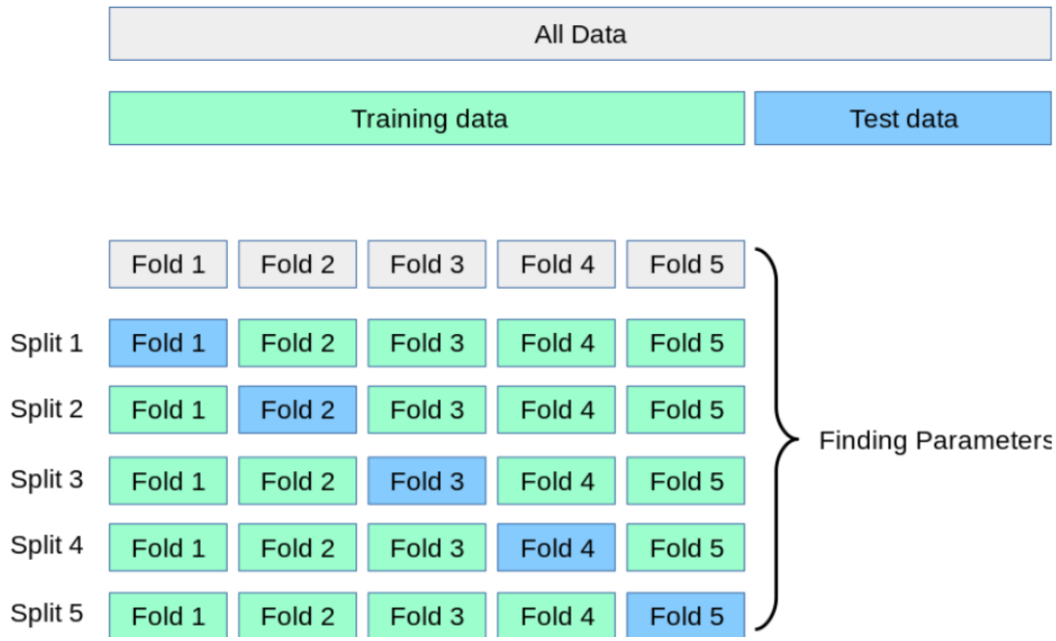

Figure 4.6: K-fold Cross Validation schema for 5 folds [19]

The performance calculated by k-fold cross-validation is then the average of the values computed in the iterations for each folder. In this present work, performance will be measured by the coefficient of determination that can be calculated:

$$R^2 = 1 - \frac{\sum_i (y_i - \bar{y})^2}{\sum_i (y_i - f_i)^2}, \tag{4-28}$$

where $\bar{y}$ is the mean of fuel efficiency and denominator represents the sum of squares of residuals. This coefficient can be ranged from 0 to 1 and the better model will be the one with a coefficient closer to 1.

This technique, illustrated by Figure 4.6, can be computationally expensive, but does not misuse data as it happens when a validation set is segregated to extract optimal hyperparameters settings. In this present work, for both methods, it will be used a 10 fold cross-validation strategy.

4.6 Accumulated Local Effect

Both methods studied in this work to predict fuel efficiency are treated as black-box algorithm since there is a difficulty to understand each parameter that influences to the target value. Hopefully, there is a large field of research in progress that deals with these methods in order to explain methods such as RF and ANN. According to Linardatos et al. [42], this field is also known as eXplainable Artificial Intelligence (XAI) and employees several techniques to interpret from ensemble methods to ANN. A visualization tool proposed by Friedman [43] helps to comprehend any predictive model by plotting the impact of explicit variables or subgroup of variables on a the model's predictions, these are called Partial Dependence Plots or (PDPs). Mathematically it is demonstrated by averaging the predictions over a marginal distribution:

$$\begin{aligned} \hat{f}_{x_S,PDP}(x_S) &= E_{X_C}[\hat{f}(x_S, X_C)] \\ &= \int_{x_C} \hat{f}(x_S, x_C)\mathbb{P}(x_C)dx_C \end{aligned} \tag{4-29}$$

As reported by [44], Equation 4-29 is the value of the prediction function f, at feature values x_S, averaged over all features in x_C. Averaging means calculating the marginal expectation E over the features in set C, which is the integral over the predictions weighted by the probability distribution. PDPs are suitable when there is no correlation between features. This happens because the computation of a PDP for a variable that is highly correlated with other variables implicates averaging predictions of artificial data instances that could not be possible to exist in the real world. As means to overwhelm this

sensibility, Accumulated Local Effect (ALE) plots were proposed by Apley and Zhu [45] and simply average the changes in the predictions and accumulate them over a grid

$$\begin{aligned}\hat{f}_{x_S,ALE}(x_S) &= \int_{z_{0,1}}^{x_S} E_{X_C|X_S}[\hat{f}^S(X_S, X_C)|X_S = z_S]dz_S - c \\ &= \int_{z_{0,1}}^{x_S}\int_{x_C} \hat{f}^S(x_S, x_C)\mathbb{P}(x_C|z_S)dx_C dz_S - c,\end{aligned} \tag{4-30}$$

where c is a constant. Equation 4-30 has three consequences when compared to PDP. The first one is on the average process in which PDP happens over the predictions itself while on ALE this occurs over the changes, expressed by:

$$\hat{f}^S(x_S, x_C) = \frac{\partial \hat{f}(x_S, x_C)}{\partial x_S}. \tag{4-31}$$

Secondly, local gradients are accumulated over the range of features in set S, which gives the effect of the feature on the prediction. In the computation process, z is replaced by a grid of intervals over which is computed the variations in the prediction. In summary, oppositely on averaging the predictions, the ALE method computes the prediction differences conditional on features S and integrates the derivative over features S to estimate the effect [44]. This two step process, that mathematically cancel each other, has the power to isolate the effect of the feature of interest and blocks the effect of correlated features.

5
Numerical Results

As describe in the previous chapters, two ML algorithms were used in this work based on 2020 whole year dataset i.e., 38290 raw data instances, which 13 independent variables were used. The results for RF and ANN for both training and test set will be presented is this chapter. Moreover, a brief summary of the system and libraries used to run all the experiments is described in Table 5.1

System/package	*Description*
Operational System	MAC OS Catalina 10.15.7
Processor	2.6 GHz Dual-Core Intel Core i5
Memory	8 GB 1600 MHz DDR3
Graphics	Intel Iris 1536 MB
Python	3.7
Scikit-learn	0.24.2
TensorFlow	2.5.0
Hyperopt	0.2.5
Dalex	1.0.0

Table 5.1: Specification of the system and libraries used to execute all experiments.

5.1
Data pre-processing

As the first step in the pre-processing procedure, the KS-statistic test was calculated for two independent variables: trip kilometers and run-time hours. The target variable was also evaluated with this same approach to guarantee the consistency of the data. The KS-statistic was employed for each unique pair of origin and destination (R), as can be seen in Table 5.2.

R	*Trip Kilometers*	*Run Time Hours*	*Fuel Efficiency*
R_1	0.0389	0.0029	0.0478
R_2	0.0022	0.0087	0.0187
R_3	0.0019	0.0067	0.0478
R_4	0.0012	0.0361	0.0089
R_5	0.0365	0.0291	0.0012
R_6	0.0189	0.0090	0.0090
R_7	0.0026	0.0125	0.0010
R_8	0.0478	0.0161	0.0498
R_9	0.0231	0.0091	0.0023
R_{10}	0.0275	0.0060	0.0456
R_{11}	0.0256	0.0008	0.0378
R_{12}	0.0117	0.0078	0.0081
R_{13}	0.0389	0.0026	0.0020
R_{14}	0.0111	0.0489	0.0098
R_{15}	0.0001	0.0456	0.0310
R_{16}	0.0009	0.0395	0.0040

Table 5.2: KS-statistic result for each variable and unique tuple of origin and destination

Therefore, as the nonparametric tests illustrated above, no value reached the criteria value of 0.05, supporting the null hypotheses (variables follow a normal distribution for each pair of origin and destination dataset). Similarly, the variables follow a normal distribution, and the outlier removal algorithm can be used without any significant loss in the dataset. After running the procedure, a drop of 1.5% of the data was observed, resulting in 37724 data instances for the whole 2020 year.

5.2 Hyperparameter Optimization

In order to select reasonable values for the hyperparamteres needed for both models, TPE were used through *Hyperopt* library. Therefore, the search space E for RF and ANN will be detailed in Table 5.3 and Table 5.5, respectively.

Hyperparameter	*Distribution*	*Bounds*
N Estimators	Uniform	[1,2000]
Max Features	Choice	$\{13,\sqrt{13}\}$
Max Depth	Uniform	[10,110]
Min Samples Split	Choice	{2,22,44}
Min Samples Leaf	Choice	{1,2,4}
Bootstrap	Choice	{True, False}

Table 5.3: Probability distribution and bounds for RF hyperparameters

Left columns of Table 5.3 stands for the hyperparamters name in RF model. The middle one describes which probability distribution the corresponding hyperparamters relies on or if it is a choice between distinguish values. The column Bounds defines the limits for the probabiliry distribution or the values available in a choice. The first hyperparamter is the number of trees in the RF model. Max features represent the number of features to consider when looking for the best split. Max depth, the maximum depth of a tree. Min samples split, the minimum number of samples required to split an internal node. Min samples leaf, the minimum number of samples required to be at a leaf node. Finally bootstrap, a boolean hyperparameter that decides Whether bootstrap samples are used when building trees.

Defined the limit space, TPE can now be applied with a 200 iterations or function evaluations based on the search space described by Table 5.3.

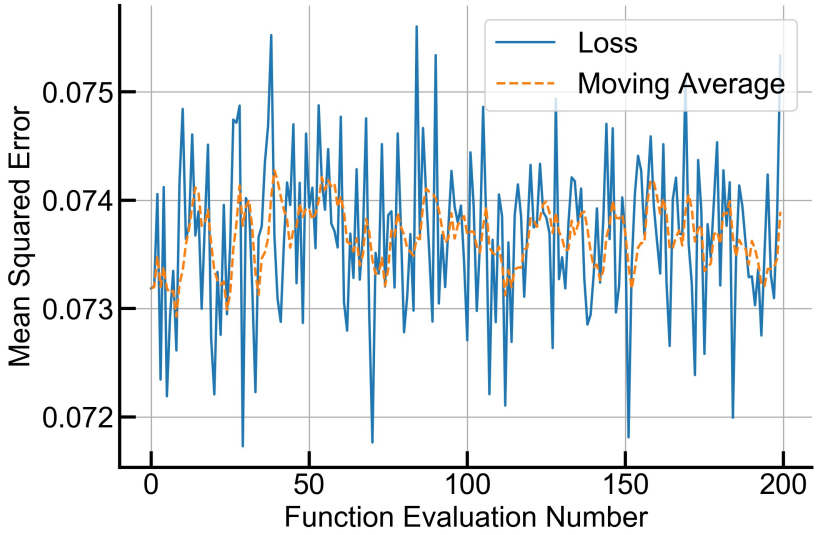

Figure 5.1: MSE loss evolution when applying TPE for RF hyperparamter optimization

Figure 5.1 illustrates evolution of MSE loss for each iteration of TPE. Despite varying all the six hyperparamters listed in table 5.3, MSE appears to

follow no tendency as can be confirmed by the five iteration moving average. Finally, the best hyperparameter setting is then showed in Table 5.4 which is represented by iteration 29 of Figure 5.1.

Hyperparameter	*Best value*
N Estimators	428
Max Features	$\sqrt{13}$
Max Depth	75
Min Samples Split	44
Min Samples Leaf	4
Bootstrap	True

Table 5.4: Best choice of hyperparamter setting for RF model according to the TPE algorithm

Following the same procedure presented for RF, hyperparameters evaluations for ANN is showed in Table 5.5. The first row of this table represents the number of hidden layers that will be supplied for the TPE algorithm. Next six hyperparameters addresses number of hidden units and dropout rate for each layer. Batch size, number of epochs, optimizer algorithm, activation function and kernel initializer for weight matrix initial values are also defined as hyperparameters for tuning. An important observation should be noted for fixed number of epochs. This happens to guarantee isonomy between all the interactions that will take place.

Hyperparameter	*Distribution*	*Bounds*
Layers	Choice	{1,2,3}
Neurons first layer	Uniform	[13,500]
Dropout rate first layer	Uniform	[0.1,0.85]
Neurons second layer	Uniform	[13,200]
Dropout rate second layer	Uniform	[0.1,0.85]
Neurons third layer	Uniform	[13,100]
Dropout rate third layer	Uniform	[0.1,0.85]
Batch size	Uniform	[28,128]
N Epochs	Choice	{100}
Optimizer	Choice	{*Adam*, *Adadelta*, *RMSProp*}
Activation	Choice	{*Relu*, *Elu*, *Linear*}
Kernel Initializer	Choice	{*Uniform*, *Normal*}

Table 5.5: Probability distribution and bounds for ANN hyperparameters

With limit space for ANN established, TPE can be evaluated using the same number of iterations as used for RF model. Results for the 200 iterations is represented by Figure 5.2.

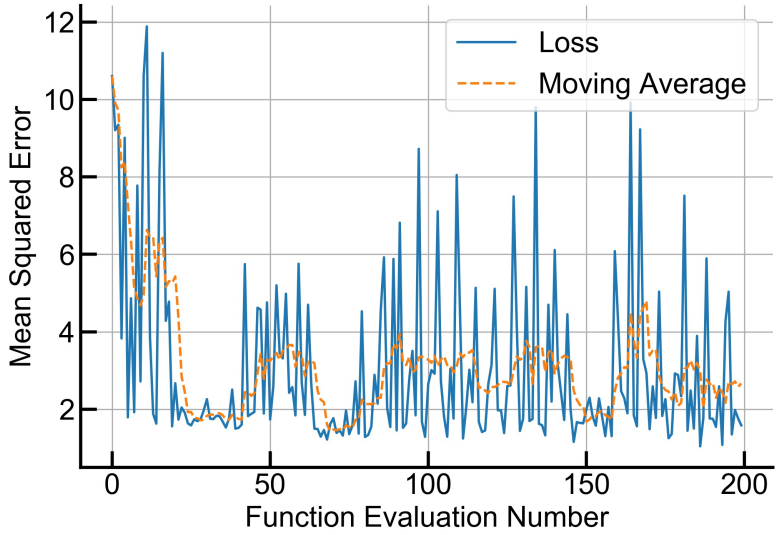

Figure 5.2: MSE loss evolution when applying TPE for ANN hyperparamter optimization

From Figure 5.2, it can be seen that moving average of MSE diminishes in first 25 iterations and then oscillates around 2.2 MSE horizontal line. Finally, best hyperparameters setting is achieved in iteration number 186 which are listed in Table 5.6

Hyperparameter	*Best value*
Layers	2
Neurons first layer	237
Dropout rate first layer	0.18
Neurons second layer	194
Dropout rate second layer	0.13
Batch size	77
Optimizer	*RMSProp*
Activation	*Relu*
Kernel Initializer	*Normal*

Table 5.6: Best choice of hyperparamter setting for ANN model according to the TPE algorithm

Therefore, hyperparameter settings for both models are well defined using TPE as tool in optimization for the search space. As a next step, these values will feed the models in training and testing procedure that will be discussed in next section.

5.3 Models evaluation

Based on the hyperparameters defined in the previous section, the training and testing procedure can now be evaluated. For the RF model, there is no learning parameter to calibrate as it happens in ANN, in which there is a need for estimating the weight matrix **w**. This procedure is accomplished using early stopping in order to avoid overfitting and is best illustrated by Figure 5.3, in which testing error stops improvement after 1367 epochs.

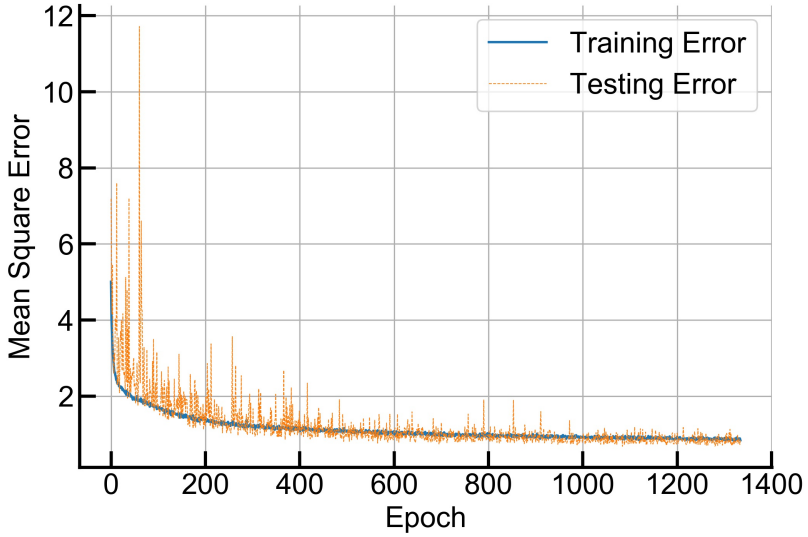

Figure 5.3: Training and testing error curves for ANN with reasonable hyperparameters and early stopping trigger

With all parameters settled, models can now be evaluated on the training and test set. For this work, 80% of data were segregated for training proposal while the remaining was set aside for the test set. Figures 5.4 and 5.5 showed results in training and test dataset for RF and ANN, respectively. These figures showed normalized real values on the horizontal axis and normalized predicted values on the vertical axis, therefore the best model will be the one with high co-linearity with the dashed line represented by linear function $y(x) = x$.

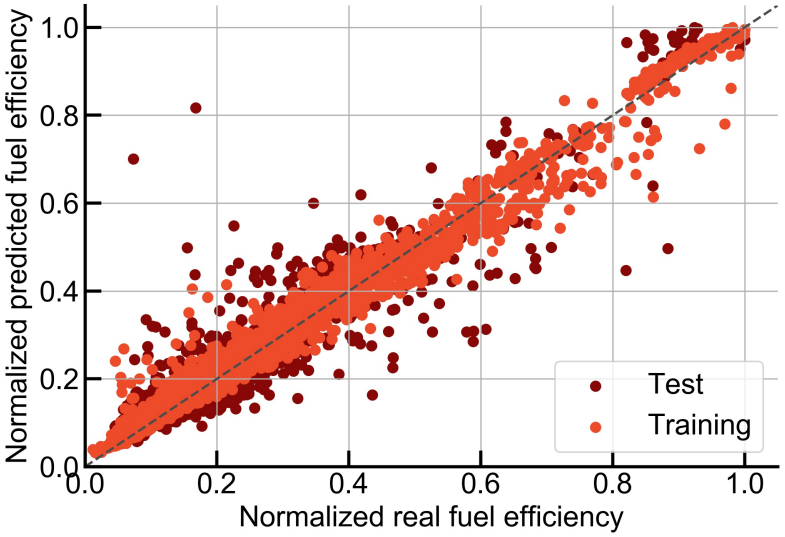

Figure 5.4: RF plot comparison between real and predicted results

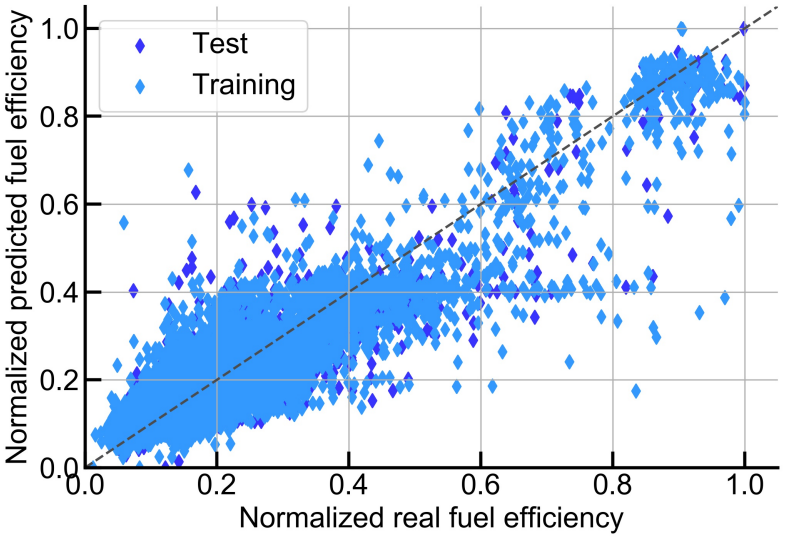

Figure 5.5: ANN plot comparison between real and predicted results

From Figure 5.4 it can be seen that RF proved better performance when compared to ANN. However, to avoid overfitting and discard any possibility that a model could be performed better based on random selection of test dataset, a k-fold cross-validation technique was used and the results are represented by Table 5.7.

Fold	*Random forest*	*Neural network*
# 1	0.92	0.78
# 2	0.91	0.86
# 3	0.93	0.87
# 4	0.91	0.87
# 5	0.95	0.81
# 6	0.89	0.86
# 7	0.93	0.89
# 8	0.92	0.88
# 9	0.93	0.85
#10	0.87	0.73
$\mu \pm \sigma$	0.91 ± 0.02	0.84 ± 0.05

Table 5.7: R^2 for 10 fold cross validation technique

First column of Table 5.7 represents the folder generated by the methodology described in the previous chapter regarding K-fold cross-validation. Second column represents the coefficient of determination calculated for each folder in RF prediction model. Next column follows the same logic but for ANN model. From this table it can be seen that both estimators do not present a constant coefficient of determination for the different folds. Therefore, a simple statistical analysis should be evaluated in order to select the best model. The last row contains information about the mean μ and variance σ for the model predictions studied. It can be noted that RF presented a higher mean and lower variance, therefore this estimator will be used in the next section in order to calculate the ALE plot for the four features already defined.

5.4 ALE plots

As means to analyze which manageable feature could have a deeper impact in fuel efficiency, ALE plots for the four dependent variables are presented in this section using RF as back-end prediction model. In order to isolate any profile interference, the results produced here are related to a specific tuple of origin and destination of the vector $\mathcal{R}$ defined in data cleaning process. First one is how trip mean velocity impacts fuel efficiency and is showed in Figure 5.6.

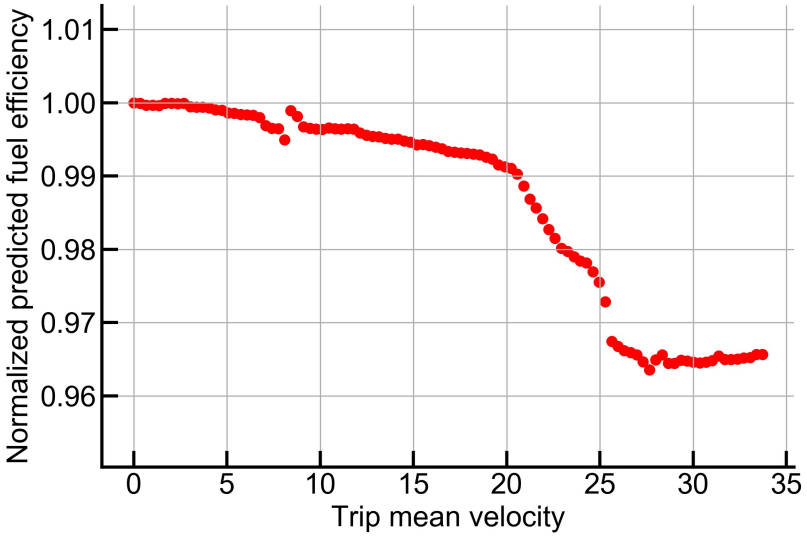

Figure 5.6: Accumulated local effect plot for trip mean velocity

Trip mean velocity follows an almost negative constant rate of change from 0 to 20 km/h when then a higher rate is observed until approximately an optimal 27 km/h for fuel efficiency. After this threshold, an increase in trip mean velocity represent an increase in fuel efficiency.

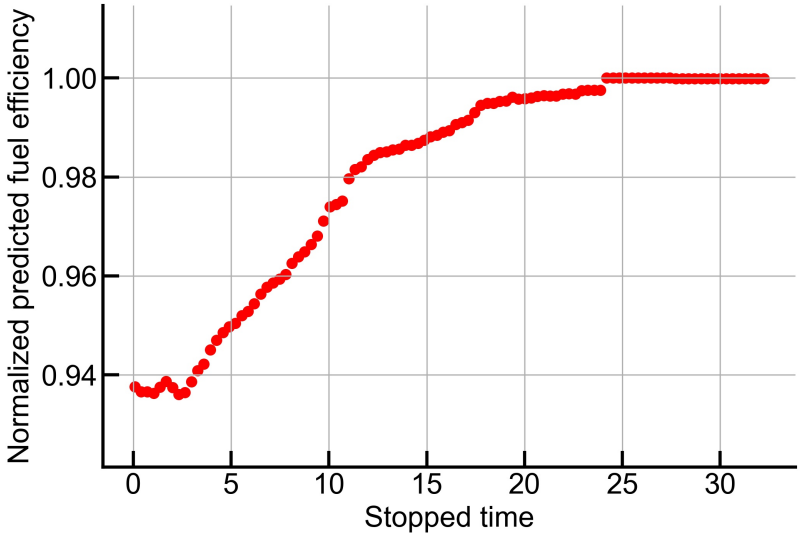

Figure 5.7: Accumulated local effect plot for stopped time

Accumulated local effect for stopped time is then observed in Figure 5.7. A positive correlation between this feature and fuel efficiency is observed after 3 hours of stopped time. This correlation is almost linear until it reaches 20 hours of stopped time.

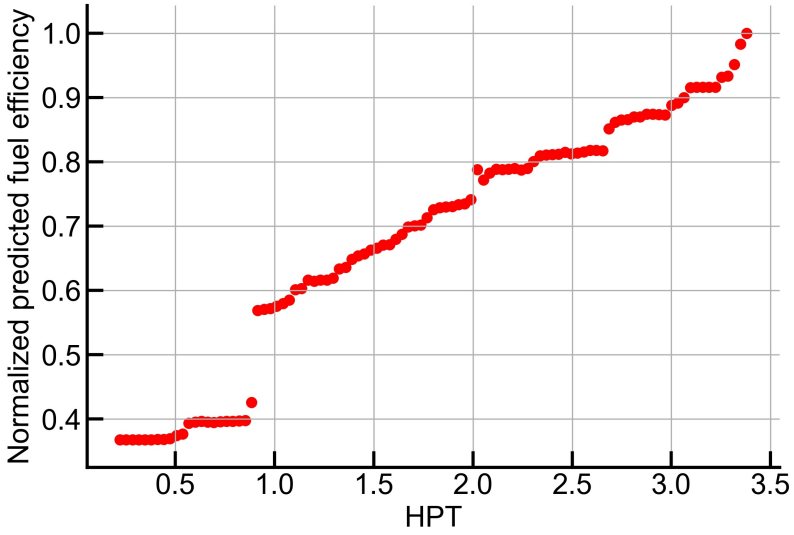

Figure 5.8: Accumulated local effect plot for HPT

Same positive correlation between dependent and independent variable is also observed for HPT, as it can be confirmed in Figure 5.8. There is a discontinuation around HPT 1 which represents train configurations that better utilize locomotive traction available.

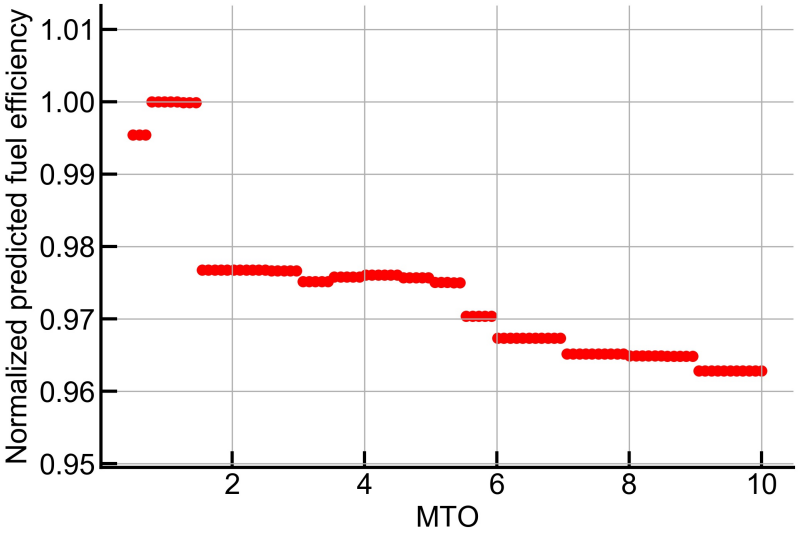

Figure 5.9: Accumulated local effect plot for MTO

MTO settings are ranged in Figure 5.9 to show accumulated local effects of this feature in fuel efficiency. From this figure it can be seen that there is a relevant drop from 0.5 to 2 and then an negative constant rate that remains until reaches MTO 10.

Feature	*Maximum impact on fuel efficiency*
Trip Mean velocity	3.64%
Stopped time	6.39%
HPT	63.19%
MTO	3.72%

Table 5.8: Manageable features and maximum impact on fuel efficiency

Finally, a summarized table for these four manageable features is reported in Table 5.8. The maximum impact on fuel efficiency is calculated by the difference between the maximum and minimum values divided by the minimum one. This table quantifies how much each feature can influence the target variable, fuel efficiency. Therefore, the most critical manageable variable would be HPT which could bring up to 63.19% on fuel efficiency for this specific tuple of origin and destination. However, this is reached when there is an operational condition that satisfies a particular train formation. Following the rank, the stopped time comes second, proving a fuel-saving up to 6.39%. It can be reached by optimizing train stops that are highly correlated to stopped time. For the specific segment studied, MTO and trip mean velocity are in the third and fourth positions with a maximum saving of 3.72% and 3.64%, respectively.

6
Conclusions

Fuel efficiency have been demonstrate a promising field in industry as means to cut costs and also to achieve reduction in gases that are harmful to environment such as green house gases. As reported in the introduction chapter, transport sector accounts for about 13% of this emission, ranking fourth place by sector. This key position led to many researches in this field regarding how to predict fuel consumption or fuel efficiency in an accurately model. Literature review showed that, for railway heavy haul segment, there are few studies using ML techniques to address this common problem. Same statement can be addressed about how manageable operational features could impact fuel efficiency, such as application of PDP or Ale plots. Therefore, two major contributions are reported by this work:

- From literature review it was evident that random forest and neural network were the most common and efficient machine learning algorithm used by researchers in related areas. Consequently, these techniques were applied in this study. As a first step, an algorithm for data cleaning was employed in order to remove any data inconsistency. Afterwards, for both methods, Bayesian Optimization was applied to search for best hyperparameter settings. In order to reduce overfitting, a 10 fold cross-validation strategy was used and proved that random forest achieved a higher accuracy with mean and standard deviation given by: 0.91 ± 0.02.
- As random forest proved to perform better, this machine learning technique was then applied to an algorithm named accumulated local effect. This procedure aims to retrieve partial dependence of a specific feature in the final target, but isolating any correlation that other features could have in the final result. For the four manageable parameters analyzed in a specific segment, HPT demonstrated to have higher impact on fuel efficiency.

6.0.1
Suggestions for Future Work

This research has the potential to guide a series of subsequent works. In order to improve the accuracy of the models, a broad range of data could be employed by expanding from one to two or more years of data collection. Moreover, data from different railways could be joined in the dataset with the objective to achieve an optimal generalization for the models studied. A final suggestion would be employing different regression techniques such as GB and SVM to test for more accurate models.

Bibliography

[1] NATIONS, U.. **World greenhouse gas emissions by sector**, aug 2020.

[2] DE FREITAS, T. G.. **Relatório de atividades 2020**, jul 2021.

[3] FRANCO, I. G.; MARCOS, E. S.. **Technologies and potential developments for energy efficiency and co2 reductions in rail systems**. Technical report, International Union of Railways, 2016.

[4] ZOU, Q.; QU, K.; LUO, Y.; YIN, D.; JU, Y. ; TANG, H.. **Predicting diabetes mellitus with machine learning techniques**. Frontiers in Genetics, 9, Nov. 2018.

[5] YAO, J.; MOAWAD, A.. **Vehicle energy consumption estimation using large scale simulations and machine learning methods**. Transportation Research Part C: Emerging Technologies, 101:276–296, Apr. 2019.

[6] XU, X.; ZHAO, Y.. **Prediction of fuel consumption per 100km for automobile engine based on gaussian processes machine learning**. Applied Mechanics and Materials, 34-35:1951–1955, Oct. 2010.

[7] GKEREKOS, C.; LAZAKIS, I. ; THEOTOKATOS, G.. **Machine learning models for predicting ship main engine fuel oil consumption: A comparative study**. Ocean Engineering, 188:106282, Sept. 2019.

[8] KARAGIANNIDIS, P.; THEMELIS, N.; ZARAPHONITIS, G.; SPANDONIDIS, C. ; GIORDAMLIS, C.. **Ship fuel consumption prediction using artificial neural networks**. 11 2019.

[9] KANG, L.; HANSEN, M.. **Improving airline fuel efficiency via fuel burn prediction and uncertainty estimation**. Transportation Research Part C: Emerging Technologies, 97:128–146, Dec. 2018.

[10] WICKRAMANAYAKE, S.; BANDARA, H. D.. **Fuel consumption prediction of fleet vehicles using machine learning: A comparative study**. In: 2016 MORATUWA ENGINEERING RESEARCH CONFERENCE (MERCon). IEEE, Apr. 2016.

[11] XIA, X.; ZHANG, J.. **Modeling and control of heavy-haul trains [applications of control]**. IEEE Control Systems Magazine, 31(4):18–31, 2011.

[12] SHI, J.; REN, S. ; ZHANG, M.. **MODEL-BASED ASSESSMENT OF LONGITUDINAL DYNAMIC PERFORMANCE AND ENERGY CONSUMPTION OF HEAVY HAUL TRAIN ON LONG-STEEP DOWNGRADES**. Transport, 34(3):250–259, Mar. 2019.

[13] HOYT, E. V.; LEVARY, R. R.. **Assessing the effects of several variables on freight train fuel consumption and performance using a train performance simulator**. Transportation Research Part A: General, 24(2):99–112, Mar. 1990.

[14] SUN, X.; MA, Z.; YAO, E. ; WU, X.. **Modeling the traction energy consumption for urban rail line considering operation characteristics**. In: GREEN INTELLIGENT TRANSPORTATION SYSTEMS, p. 707–723. Springer Singapore, July 2017.

[15] HINKLE, D.. **Applied statistics for the behavioral sciences**. Houghton Mifflin Hi Marketing (distributor, Boston, Mass. London, 2003.

[16] PRASAD, Y. S.; KRISHNA, G. R.. **Statistical anomaly detection technique for real time datasets**. 2013.

[17] DANIEL, W.. **Applied nonparametric statistics**. PWS-KENT Pub, Boston, 1990.

[18] BREIMAN, L.. Machine Learning, 45(1):5–32, 2001.

[19] PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M. ; DUCHESNAY, E.. **Scikit-learn: Machine learning in Python**. Journal of Machine Learning Research, 12:2825–2830, 2011.

[20] BISHOP, C.. **Pattern recognition and machine learning**. Springer, New Delhi, 2013.

[21] GLOROT, X.; BORDES, A. ; BENGIO, Y.. **Deep sparse rectifier neural networks**. In: Gordon, G.; Dunson, D. ; Dudík, M., editors, PROCEEDINGS OF THE FOURTEENTH INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE AND STATISTICS, volumen 15 de **Proceedings**

of Machine Learning Research, p. 315–323, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.

[22] NIELSEN, M.. **Neural Networks and Deep Learning**. Determination Press, 2015.

[23] SRA, S.. **Optimization for machine learning**. MIT Press, Cambridge, Mass, 2012.

[24] GOODFELLOW, I.. **Deep learning**. The MIT Press, Cambridge, Massachusetts, 2016.

[25] ABADI, M.; AGARWAL, A.; BARHAM, P.; BREVDO, E.; CHEN, Z.; CITRO, C.; CORRADO, G. S.; DAVIS, A.; DEAN, J.; DEVIN, M.; GHEMAWAT, S.; GOODFELLOW, I.; HARP, A.; IRVING, G.; ISARD, M.; JIA, Y.; JOZEFOWICZ, R.; KAISER, L.; KUDLUR, M.; LEVENBERG, J.; MANÉ, D.; MONGA, R.; MOORE, S.; MURRAY, D.; OLAH, C.; SCHUSTER, M.; SHLENS, J.; STEINER, B.; SUTSKEVER, I.; TALWAR, K.; TUCKER, P.; VANHOUCKE, V.; VASUDEVAN, V.; VIÉGAS, F.; VINYALS, O.; WARDEN, P.; WATTENBERG, M.; WICKE, M.; YU, Y. ; ZHENG, X.. **TensorFlow: Large-scale machine learning on heterogeneous systems**, 2015. Software available from tensorflow.org.

[26] HEUSEL, M.; RAMSAUER, H.; UNTERTHINER, T.; NESSLER, B. ; HOCHREITER, S.. **Gans trained by a two time-scale update rule converge to a local nash equilibrium**. 12 2017.

[27] KINGMA, D.; BA, J.. **Adam: A method for stochastic optimization**. International Conference on Learning Representations, 12 2014.

[28] DUCHI, J.; HAZAN, E. ; SINGER, Y.. **Adaptive subgradient methods for online learning and stochastic optimization**. Journal of Machine Learning Research, 12(61):2121–2159, 2011.

[29] ZEILER, M. D.. **Adadelta: An adaptive learning rate method**, 2012.

[30] HINTON, G. E.; SRIVASTAVA, N.; KRIZHEVSKY, A.; SUTSKEVER, I. ; SALAKHUTDINOV, R.. **Improving neural networks by preventing co-adaptation of feature detectors**. ArXiv, abs/1207.0580, 2012.

[31] GEMAN, S.; BIENENSTOCK, E. ; DOURSAT, R.. **Neural networks and the bias/variance dilemma**. Neural Computation, 4:1–58, 01 1992.

[32] EVERITT, B.. **The Cambridge dictionary of statistics**. Cambridge University Press, Cambridge New York, 2010.

[33] PRECHELT, L.. **Automatic early stopping using cross validation: quantifying the criteria**. Neural Networks, 11(4):761–767, June 1998.

[34] FINNOFF, W.; HERGERT, F. ; ZIMMERMANN, H. G.. **Improving model selection by nonconvergent methods**. Neural Networks, 6(6):771–783, Jan. 1993.

[35] SRIVASTAVA, N.; HINTON, G.; KRIZHEVSKY, A.; SUTSKEVER, I. ; SALAKHUTDINOV, R.. **Dropout: A simple way to prevent neural networks from overfitting**. Journal of Machine Learning Research, 15(56):1929–1958, 2014.

[36] PINTO, N.; DOUKHAN, D.; DICARLO, J. J. ; COX, D. D.. **A high-throughput screening approach to discovering good forms of biologically inspired visual representation**. PLoS Computational Biology, 5(11):e1000579, Nov. 2009.

[37] COATES, A.; NG, A. ; LEE, H.. **An analysis of single-layer networks in unsupervised feature learning**. Journal of Machine Learning Research - Proceedings Track, 15:215–223, 01 2011.

[38] BERGSTRA, J.; YAMINS, D. ; COX, D.. **Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures**. In: ICML, 2013.

[39] HUTTER, F.. **Automated Configuration of Algorithms for Solving Hard Computational Problems**. PhD thesis, University of British Columbia,, 2009.

[40] HUTTER, F.; HOOS, H. H. ; LEYTON-BROWN, K.. **Sequential model-based optimization for general algorithm configuration**. In: Coello, C. A. C., editor, LEARNING AND INTELLIGENT OPTIMIZATION, p. 507–523, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[41] BERGSTRA, J.; BARDENET, R.; BENGIO, Y. ; KÉGL, B.. **Algorithms for hyper-parameter optimization**. In: Shawe-Taylor, J.; Zemel, R.; Bartlett, P.; Pereira, F. ; Weinberger, K. Q., editors, ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, volumen 24. Curran Associates, Inc., 2011.

[42] LINARDATOS, P.; PAPASTEFANOPOULOS, V. ; KOTSIANTIS, S.. **Explainable AI: A review of machine learning interpretability methods**. Entropy, 23(1):18, Dec. 2020.

[43] FRIEDMAN, J. H.. **Greedy function approximation: A gradient boosting machine.** The Annals of Statistics, 29(5), Oct. 2001.

[44] MOLNAR, C.. **Interpretable Machine Learning**. 2019. `https://christophm.github.io/interpretable-ml-book/`.

[45] APLEY, D. W.; ZHU, J.. **Visualizing the effects of predictor variables in black box supervised learning models**. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 82(4):1059–1086, June 2020.