UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e da Computação

Euler Rodrigues de Sousa Faria

A system able to forecast and explain queue events on open pit mines

Um sistema capaz de prever e explicar eventos de fila em minas a céu aberto

Campinas

2019

UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e da Computação

Euler Rodrigues de Sousa Faria

A system able to forecast and explain queue events on open pit mines

Um sistema capaz de prever e explicar eventos de fila em minas a céu aberto

Dissertation presented to the School of Electrical and Computer Engineering at University of Campinas in partial fulfillment of the requirements for the degree of Master of Electrical Engineering, in the area of Computing Engineering.

Dissertação apresentada à Faculdade de Engenharia Elétrica e de Computação o da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtençãoo do tíulo de Mestre em Engenharia Elétrica, na área de Engenharia da Computação

Supervisor: Prof. Dr. Fernando José Von Zuben

Este exemplar corresponde à versão final da dissertação defendida pelo aluno Euler Rodrigues de Sousa Faria, orientada pelo Prof. Dr. Fernando José Von Zuben

---

Campinas

2019

COMISSÃO JULGADORA - DISSERTAÇÃO DE MESTRADO

Candidato: Euler Rodrigues de Sousa Faria RA: 209440

Data da Defesa: 08 de Março de 2019

Título da Tese: "A system able to forecast and explain queue events on open pit mines"

Prof. Dr. Fernando José Von Zuben (Presidente, FEEC/UNICAMP)

Prof. Dr. Bruno Henrique Groenner Barbosa (DEG/UFLA)

Prof. Dr. Mateus Giesbrecht (FEEC/UNICAMP)

A ata de defesa, com as respectivas assinaturas dos membros da Comissão Julgadora, encontra-se no SIGA (Sistema de Fluxo de Dissertação/Tese) e na Secretaria de PósGraduação da Faculdade de Engenharia Elétrica e de Computação.

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Rose Meire da Silva - CRB 8/5974

F225s

Faria, Euler Rodrigues de Sousa, 1993-
A system able to forecast and explain queue events on open pit mines / Euler Rodrigues de Sousa Faria. – Campinas, SP : [s.n.], 2019.

Orientador: Fernando José Von Zuben.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Séries temporais. 2. Aprendizado de máquina. I. Von Zuben, Fernando José, 1968-. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Um sistema capaz de prever e explicar eventos de fila em minas a céu aberto
Palavras-chave em inglês:
Time Series
Machine Learning
Área de concentração: Engenharia de Computação
Titulação: Mestre em Engenharia Elétrica
Banca examinadora:
Fernando José Von Zuben [Orientador]
Bruno Henrique Groenner Barbosa
Mateus Giesbrecht
Data de defesa: 08-03-2019
Programa de Pós-Graduação: Engenharia Elétrica

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0001-9094-0060
- Currículo Lattes do autor: http://lattes.cnpq.br/1522820862003933

Acknowledgements

First and undoubtedly I would like to give thanks for my mother Lucia who raise me up by herself against all of the financial difficulties and challenges that a mom can face by creating two children as a single mother. This victory is also yours.

Second for my brother Edgar Jr. who was always supportive to my dreams and stood by my side on the difficult moments of this life.

Third to my advisor Fernando Von Zuben, for the opportunity to trace this journey, for all the patience and knowledge taught throughout lessons and several meetings that he gladly arranged on his busy schedule, and the many contributions made for this thesis. I also would like to give thanks to Mateus Giesbrecht for the various advice about carreer, life and technical aspects of problems related to time series.

Moreover I would like to give thanks for Daniel Scalli Fonseca, my manager at IBM, who as soon as I was accepted in the M.Sc. program he trusted me to work in his team and keep studying making all arrangements for the flexibility necessary to accomplish this endeavour. Last but not least, I would like to give thanks for Edson Antonio, Data Science Program Leader at a multinational Mining Company, for all the discussions and ideation processes who instigate me and contributed for many of the solutions here proposed.

Abstract

This research project comprehensively analyzes queue time events of haul trucks at dumping time in open pit mines, tracking its main contributors factors. A data mining process is conducted to gather all relevant contributors at disposal. Based on this consolidated data, it is proposed a system, denoted as Oráculo. Oráculo is composed of three modules: a Predictor module consisting in a NARX model based on an Artificial Neural Network capable of forecasting the average queue time faced by haul trucks at dumping area with performance of 12% of RRMSE (Relative Root Mean Square Error) on the test set; a Decision maker module responsible for producing alarms of surge of queue events performing a F1-score of 0.71 on the test set; and an Explainer module able to stratify the main contributors for the alarms generated providing a leaver of action to avoid or minimize such counterproductive scenarios before they occur. The Oráculo system has been successfully tested in a real mine site in Brazil and it is currently being applied in a production environment of a multinational mining company. To the best of author's knowledge, there is no system published or in the market yet capable of performing the set of the capacities accomplished by Oráculo making it the first of its line.

Keywords: Queue-Events; Open-pit; Time Series; Machine Learning; Accountability.

Resumo

Este projeto de pesquisa analisa criteriosamente eventos de fila de caminhões fora de estrada no momento de básculo em minas a céu aberto, elencando seus principais contribuidores. Um processo de mineração de dados é conduzido a fim de reunir todas as características relevantes à disposição. Baseado neste conjunto de características, é proposto um sistema denotado por Oráculo. Oráculo é composto por três módulos, um módulo Preditor constituído por um modelo NARX baseado em uma Rede Neural Artificial capaz de prever o tempo médio de fila enfrentado pelos caminhões fora de estrada com performance de 12% de RRMSE (do inglês Relative Root Mean Square Error) no conjunto de teste; um módulo Decisor responsável por produzir alarmes de surtos de fila com 0.71 de F1-score no conjunto de teste; um módulo Esclarecedor capaz de apontar os principais contribuidores para os alarmes gerados provendo um apoio à tomada de decisão capaz de evitar ou minimizar tais cenários de forma antecipatória. O sistema Oráculo foi testado com sucesso em uma mina localizada no Brasil e está atualmente sendo aplicado em um ambiente de produção por uma empresa multinacional do ramo de mineração. Segundo o conhecimento do autor, não há nenhum sistema publicado ou no mercado até a atual data, capaz de executar o conjunto de capacidades realizadas pelo sistema Oráculo, o tornando o primeiro em sua linha.

Keywords:Eventos de Fila; Séries Temporais; Aprendizado de Máquina; Atribuição de causa

List of Figures

List of Tables

Contents

1 Introduction

Surface mining is the most common mining method worldwide, and open pit mining accounts for more than 60% of all surface output (HARTMAN; MUTMANSKY, 2002). Truck haulage is responsible for the majority of costs in a surface mining operation (ERCELEBI; BASCETIN, 2009), therefore it is highly desirable to maintain an efficient haulage system.

The productive cycle of truck haulages in open pit mines can be divided in different components: loading, hauling, maneuvering, dumping, returning and spotting as presented in Figure 1.1 (SOOFASTAEI *et al.*, 2015). During both loading and dumping time it often occurs queue events which are caused by a variety of reasons.

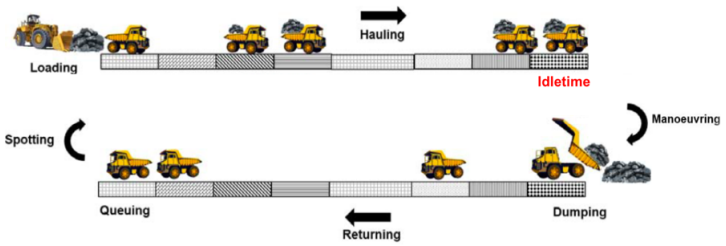


Figure 1.1 – Schematic of the Hauling operation, adapted from (SOOFASTAEI *et al.*, 2015).

In this study the time of the queue events at dumping area are called Idletime to distinguish it from time of queue events at loading area which will be called from now on Queuetime. Another important distinction adopted is between the terms prediction and forecasting as follows. In the field of machine learning the term prediction is used to represent the output of any type of model (e.g Regression, Classification, Clustering) which does not have direct relationship with the time-stamp or order of a sequence. On the other hand, the verb to forecast is used in the sense of predicting a step ahead value from a time series perspective. In other words it means the act of predicting something that will only be known in the future given the time the features are analyzed.

The surface mine is a highly dynamic environment in which a series of different events can suddenly emerge causing Idletime, Queuetime or different phenomena not described in this study. Nonetheless the main effective parameters regarding material transport in hauling operations according to the literature are: mine planning (whose goal is to optimise the return on investment by scheduling/planning operations and improving the preparation of the mineral product according to specifications); road condition (interferes directly on the haul trucks performance and operation safety requiring constant

monitoring and maintenance); truck and shovel matching (involves choosing a fleet of trucks and loaders that have the capacity to move the materials specified in the mine plan within a stipulated period aiming at reducing the overall cost of materials); swell factors (it is the amount of volume increase from Bank volume ,i.e. undisturbed, in place state, to Loose volume i.e. disturbed, excavated state, of the material due to voids, air pockets, added to the material after excavation); shovel and truck driver's ability, weather condition, payload distribution and payload variance. (KECOJEVIC; KOMLJENOVIC, 2010; NASHVER; SIGHBIN, 2007; BECKMAN, 2012; CHIRONIS, 1978).

There are a considerable amount of systems that assist the open pit mine operations, ranging from hauling operations (e.g. Modular's Dispatch System, Komatsu's HAC system) to truck's maintenance. All of those systems generate a vast amount of data that are generally used to create reports along the time and, in some more advanced automated mine operations, this data is used to monitor events and trigger actions in real time.

Even though taking actions in real time based on monitored events is valuable, forecasting abnormal behaviors is crucial for having an efficient haulage system. The key to accomplish this capability resides in the value extraction from the enormous amount of data produced by those systems. The use of Machine Learning techniques can take full advantage of this data by producing models and applications containing unique knowledge domain of the business necessities.

This research project comprehensively analyzes the Idletime events in open pit mines tracking its main contributor factors. A data mining process is conducted to gather all of those contributors in a single database preserving the pattern studied and timing coherence. Based on this consolidated data, it is proposed a system, denoted as Oráculo. The main objectives of Oráculo system are to forecast surges of Idletime and stratify its main contributors.

Oráculo is composed of three modules: a Predictor module consisting in a NARX model based on an Artificial Neural Network capable of forecasting the average queue time faced by haul trucks at dumping area with performance of 12% of RRMSE (Relative Root Mean Square Error) on the test set; a Decision maker module responsible for producing alarms of surge of queue events performing a F1-score of 0.71 on the test set; and an Explainer module able to stratify the main contributors for the alarms generated providing a leaver of action to avoid or minimize such counterproductive scenarios before they occur. The Oráculo system has been successfully tested in a real mine site in Brazil and it is currently being applied in a production environment of a multinational mining company. To the best of author's knowledge, there is no system published or in the market yet capable of performing the set of capacities accomplished by Oráculo, making it the

first of its line.

Besides introductory and concluding remarks, this dissertation contains three chapters: Literature Review, regarding every technical aspect used throughout the project; Methodology, explaining the procedures proposed for developing Oráculo system; Results, exhibiting the outcomes of the methodology adopted respecting the data disclosure policies of the enterprises involved.

2 Literature Review

In this chapter the technical aspects of this research are described in the following sections. First, time series models explaining the perspective taken to propose a solution for the problem addressed. Second, Artificial Neural Networks explaining its formal concepts, supervised learning process and auxiliary techniques used to assist the training process. Third, Local Interpretable Model-Agnostic Explanations describing the core concepts used on this research project.

The first two sections of this chapter were used to compose the Oráculo's first module (i.e the Predictor), while the third section was entirely used to compose the last module of the Oráculo system: the Explainer. The Decision Maker module will be discussed in details on the Methodology chapter.

2.1 Time series models

A realization of a stochastic process also known as time series is a sequence of numbers, or vectors, characterized by a family of covariances. Two different sequences can be representations of the same stochastic process if they have the same statistical characteristics. Once a time series is obtained (e.g. acquiring temperature values from a sensor in a constant sampling frequency during a certain period of time), one can identify a mathematical model capable of finding new realizations of the same time series from a white noise input. The problem of identifying this model is denoted by time series realization problem (GIESBRECHT, 2013).

A famous family of models for solving the realization problem of univariate time series are the autoregressive AR models and its variants: ARMA, ARIMA, SARIMA, in special the autoregressive with exogenous entries models: ARX, ARMAX, ARIMAX, SARIMAX,etc. Even though those models are powerful and easily applicable on a variety of real problems, they are limited to linear time series that are governed by Gaussian distributions and classical Bayesian theory. Additionally, the time series have to be stationary or be transformed into a stationary one by removing the trend component if possible. An excellent reference for Gaussian and Non-Gaussian models for time series can be found in (DURBIN; KOOPMAN, 2012). The Autoregressive model (AR(p)) for $t = 1, \ldots, n$ presented by Equation (2.1):

$$y_t = \phi_1 y_{t-1} + \ldots + \phi_p y_{t-p} + \zeta_t, \qquad \zeta_t \sim N(0, \sigma_\zeta^2) \tag{2.1}$$

where $y_{t-1}+\ldots+y_{t-p}$ are denoted as regressors, $\phi_1+\ldots+\phi_p$ are the parameters or coefficients to be estimated by applying any kind of numeric analysis technique (e.g. least squares, maximum likelihood), p is the number of lags and σ_ζ^2 is the variance of the Gaussian disturbance ζ_t that also has to be estimated. It is interesting to notice that the number of lags p forms a window that moves on time. Figure 2.1 exhibits this concept.

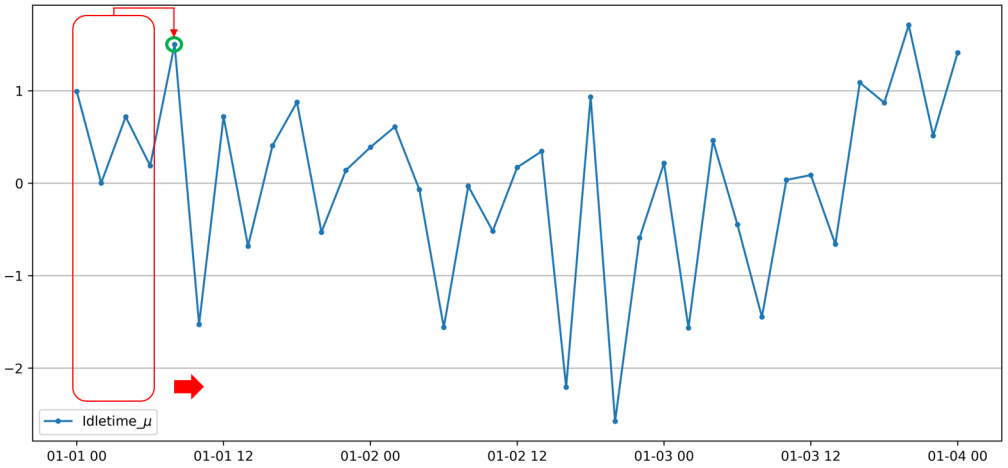


Figure 2.1 – Univariate autoregressive strategy with $p = 4$

In fact, this same strategy can be extended when exogenous inputs affects, as presented in Equation (2.2).

$$y_t = \phi_1 y_{t-1} + \ldots + \phi_p y_{t-p} + \varphi_1 u_{t-1} + \ldots + \varphi_k u_{t-k} + \zeta_t, \qquad \zeta_t \sim N(0, \sigma_\zeta^2) \tag{2.2}$$

where $u_{t-1} + \ldots + u_{t-k}$ are the regressors for the exogenous entries, $\varphi_1 + \ldots + \varphi_k$ the coefficients or parameters to be estimated and k the number of lags for the exogenous entries.

This capability of correlating multivariate time series using autoregressive models with exogenous entries configures a multiple input single output (MISO) system from a system identification perspective. Figure 2.2 exhibits this concept.

A challenge faced when applying the auto regressive models is the choice of p which resumes how long it is necessary to look back at past events in order to forecast the future. This problem can be addressed by analyzing the auto correlation function of the time series being modeled, by using AIC/BIC criterion or even using Spearman's or Pearson's correlation between the samples lagged and the output being predicted.

An interesting approach that seeks to overcome the limitations of linearity and normal distributions faced in AR models is the use of machine learning models such as Non-Linear Regressions, Random Forests and Support Vector Machines for solving the

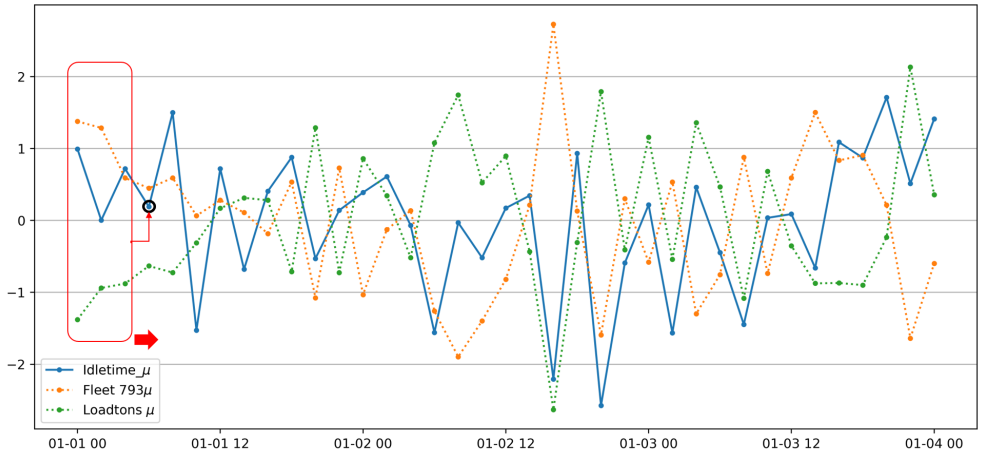


Figure 2.2 – Multivariate autoregressive strategy with $k = p = 4$

time series realization problem. Those models would have features (i.e. inputs) represented by the information highlighted on the red box in Figures 2.1 and 2.2.

For this research the Idletime faced by haul trucks at dumping area as well as its contributor factors were treated as a multivariate time series problem. All the discrete events involved in this problem were aggregated in the same frequency forming a time series for each feature analyzed. Moreover, for solving the realization problem it was devised an Artificial Neural Networks (ANN) using the autoregressive with exogenous entries approach, characterizing it as a Nonlinear-Autoregressive-Exogenous model (NARX).

2.2 Artificial Neural Networks

Artificial Neural Networks (ANN) have been exploited in a variety of applications, the majority of which are concerned with pattern recognition in one form or another (PARUNAK, 1998).

In the realm of engineering applications, ANN's have been applied in contexts ranging from materials (HAMMOOD; MAHDI, 2012),(KE-LUXIANG *et al.*, 2014) to mechanical engineering (RAHIMI-GORJI *et al.*, 2017).

The popularity of ANN's is mainly due to the universal approximation theorem, which states that a simple feed-forward network with one hidden layer containing a finite number of neurons can approximate any continuous function (CYBENKO, 1989). Although the theorem emphasizes the huge potential of ANN's, it does not touch the algorithmic learning ability of the parameters required to achieve its full potential.

2.2.1 Formalization

The basic unit of a neural network is known as neuron, node or hidden unit. The mathematical representation of a neuron is presented in Figure 2.3.

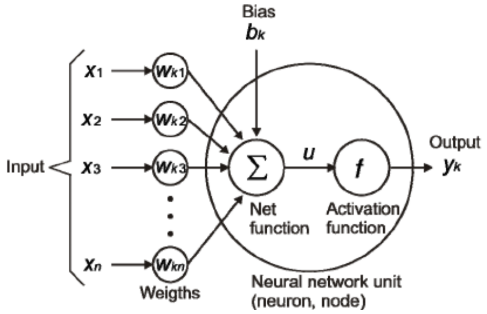


Figure 2.3 – Mathematical model of a neuron unit k.

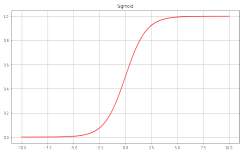
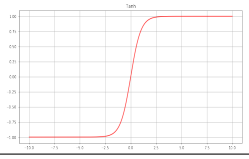
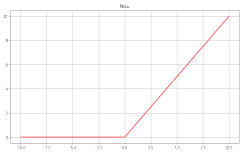
The output of a neuron k is given by Equation (2.2.1):

$$\begin{aligned} y_k = \quad & f(u_k) = f\Big(\textstyle\sum_{j=1}^{n} w_{kj}x_j + b_k\Big) \\ = \quad & f\Big(\textstyle\sum_{j=0}^{n} w_{kj}x_j\Big) = f(w_k^T x) \end{aligned} \tag{2.3}$$

where $w_{k0} = b_k$ and $x_0 = 1$.

The activation function $f(u_k)$ can be chosen from a variety of non-linear functions. Some of the most common examples are described on Table 2.1.

Table 2.1 – Three popular types of activation functions

Name	Graph	Function	Derivation	Image
Sigmoid		$f(x) = \dfrac{1}{1+e^{-x}}$	$f'(x) = f(x)(1-f(x))$	$(0,1)$
Tanh		$f(x) = \dfrac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	$f'(x) = 1-f(x)^2$	$(-1,1)$
ReLU		$f(x) = \begin{cases} 0, & \text{for } x < 0 \\ x, & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0, & \text{for } x < 0 \\ 1, & \text{for } x \geq 0 \end{cases}$	$[0,\infty)$

The rectifier linear unit (ReLU) has been widely used on convolutional neural networks (deep learning) for image applications due to its proven efficiency both in convergence and training time (KRIZHEVSKY *et al.*, 2017), (SIMONYAN; ZISSERMAN, 2014),(SZEGEDY *et al.*, 2015),(HE *et al.*, 2016). The original ReLU was further explored originating a family of activations functions: Leaky rectified linear unit (Leaky ReLU)(MAAS *et al.*, 2013), Parametric rectified linear unit (PReLU)(HE *et al.*, 2015), Randomized leaky rectified linear unit (RReLU)(XU *et al.*, 2015), Exponential linear unit (ELU)(CLEVERT *et al.*, 2015), Scaled exponential linear unit (SELU)(KLAMBAUER *et al.*, 2017), each one with its own particularities and benefits for specific applications.

The connections among neurons lead to the construction of the ANN's architecture. The more organized structures are called layers, in which the output of each neuron of a previous layer serves as input for all neurons of the subsequent layer.

A typical ANN architecture is the Multilayer Perceptron (MLP). The MLP is characterized by mapping inputs to the output by a linear combination of functions exhibiting a sigmoidal shape (i.e. Tanh, Sigmoid). The output of the MLP is obtained from Equation (2.4):

$$\hat{y}_k = \sum_{j=1}^{n} w_{kj} f\Big(\sum_{l=0}^{m} v_{jl} x_l \Big) + w_{k0} = \hat{\mathbf{g}}_k(x, \theta), k = 1, \ldots, r \tag{2.4}$$

where $\theta = [v_{10} v_{11} \ldots v_{1m} v_{2m} \ldots v_{nm} w_{10} w_{11} \ldots w_{1m} w_{2m} \ldots w_{nm}]^T$.

2.2.2 Supervised Learning

The target of this study is the prediction of a continuous value, characterizing a regression problem from a perspective of a machine learning problem. So it is desired to find a set of weights for the neural network that produces a mapping function capable of reproducing the relationship of the inputs and the output with reasonable performance.

Considering the availability of an input-output dataset $\{\mathbf{x}_{l,k}, y_{l,1}\}_{l=1,k=1}^{N,M}$, where $\mathbf{x}_{l,k}$ is the feature matrix with N rows or samples and M features, and $y_{l,1}$ is a vector with dimension $N \times 1$ containing the target. Using the autoregressive with exogenous entries approach, given by equation (2.2), for producing the input-output dataset would result in a total of $M = p + k$ features. In this case the features would be the regressors of the exogenous entries plus the output regressors (i.e. $y_{t-1} \ldots + y_p$) and the target would be the vector y_t with N samples.

One possible measurement of performance for regression models is the Mean Squared Error (MSE) of the predicted values $\hat{y}_l$ and the real values y_l as presented in

Equation (2.5), where N is the number of data points.

$$J(\theta) = \frac{1}{N}\sum_{l=1}^{N}(\hat{y}_l - y_l)^2 \tag{2.5}$$

The predicted values produced by the ANN can be described in terms of the weights, interpreted as free parameters to be adjusted during the learning process. Equation (2.6) provides a demonstration of how θ, the vector containing all ANN's weights, influences the MSE.

$$\begin{aligned} J(\theta) &= \tfrac{1}{N}\textstyle\sum_{l=1}^{N}(\hat{y}_l - y_l)^2 = \tfrac{1}{N}\sum_{l=1}^{N}(\hat{\mathbf{g}}_1(\mathbf{x}_l, \theta) - y_l)^2 \\ &= \tfrac{1}{N}\textstyle\sum_{l=1}^{N}\Big(\sum_{j=1}^{n} w_{1j} f\Big(\sum_{i=0}^{m} v_{ji} x_{li}\Big) + w_{10} - y_l\Big)^2 \end{aligned} \tag{2.6}$$

Considering P as the dimension of θ, it is possible to conclude that $J : \mathbb{R}^P \rightarrow \mathbb{R}^1$ which characterizes an error surface belonging to $\mathbb{R}^{P+1}$.

The supervised learning process of ANN's consists in finding θ that minimizes $J(\theta)$ given a dataset with N samples. This process can be seen as a non-linear optimization problem with no restrictions over θ, thus producing:

$$\theta' = \underset{\theta \in \Re^P}{\operatorname{argmin}} J(\theta) \tag{2.7}$$

A widely used algorithm to search for an optimal value of θ is the gradient descent technique (GD). The GD is a first-order iterative optimization algorithm capable of finding a local minimum of any continuous nonlinear function (e.g. the surface formed by $J(\theta)$).

The classical GD, also known as batch gradient descent, is calculated by taking the partial derivative of $J(\theta)$ for the entire dataset and then updating the weights towards the opposite direction of its gradient $\nabla_\theta J$.

The parameter that controls the step along this direction is known as the learning rate, represented by η in Equation (2.8). The greater the value of η, the larger is the step towards the closest local minimum on the $J(\theta)$ surface. Assuming the proper choice of the learning rate, including adaptive procedures to tune the learning rate along the iterations, GD converges to the global minimum of convex error surfaces and to a local minimum of non-convex surfaces, such as $J(\theta)$.

$$\theta_{t+1} = \theta_{t+1} - \eta(\nabla_\theta J(\theta)) \tag{2.8}$$

There are some important variations of the gradient descent that are applied to the supervised learning process of ANN's aiming at accelerating convergence and better managing computational resources. Some of them are described below based on literature review (RUDER, 2016).

Stochastic Gradient Descent:

The classic GD with an iterative step provided by Equation (2.8) computes the gradient for the whole dataset, whereas the stochastic gradient descent (SGD) executes the parameter updating process for each training example $\mathbf{x}_l$ and real value y_l as presented in Equation (2.9):

$$\theta_{t+1} = \theta_t - \eta(\nabla_\theta J(\theta; \mathbf{x}_l; y_l)) \tag{2.9}$$

The main advantages of SGD are the possibility of learning on the fly and in some circumstances it may be much faster than the classical gradient descent. Another interesting aspect of SGD is that, by updating the weights interactively, the loss function exhibits a high variance as presented in Figure 2.4. This fluctuation might lead to a better local minimum of the $J(\theta)$ surface, when compared to the local minimum of GD. This variance can hinder convergence, but slowly decreasing the learning rate along the iterations tends to promote convergence.

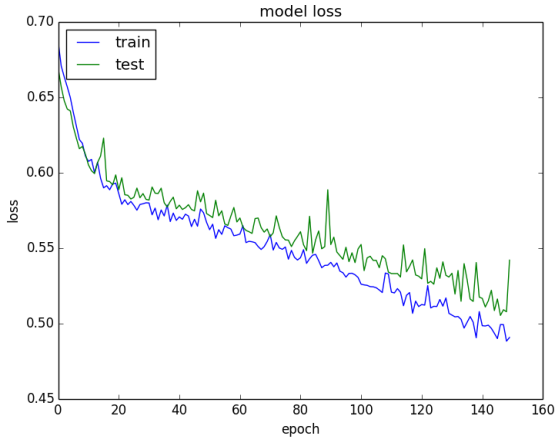


Figure 2.4 – Fluctuations verified along the training process

Mini-Batch Gradient Descent:

Another version of the classical gradient descent is the Mini-Batch Gradient Descent (MGD), which takes the best of GD and SGD by updating the weights for every

mini-batch with n training samples as presented in equation (2.10).

$$\theta_{t+1} = \theta_t - \eta(\nabla_\theta J(\theta; \mathbf{x}_{l:l+n}; y_{l:l+n}) \tag{2.10}$$

The MGD reduces the high variance noticed on SGD leading to a more stable convergence. The mini-batch involves a matrix of training samples so it can resort to matricial operations to implement optimization techniques, such as the use of Graphical Process Units (GPU) with Cuda software from NVIDIA.

Regularization techniques:

A valuable technique used in many supervised learning approaches for machine learning models is regularization. Regularization contributes to reduce overfitting by increasing the generalization capability of the models.

The most common regularization techniques are: L1, also called Lasso (TIBSHIRANI, 1996), L2 and dropout, which is used in particular for ANN's. As mentioned before, the supervised learning process of ANN's consists in finding the best combination of weights θ that minimizes $J(\theta)$ given a training data set. With regularization L1 or L2, Equation (2.5) becomes:

$$J(\theta) = \frac{1}{N}\sum_{l=1}^{N}(\hat{y}_l - y_l)^2 + \lambda\Omega(\theta) \tag{2.11}$$

where coefficient $\lambda \geq 0$ controls the strength of the regularization, so that the higher its values the higher is the regularization imposed on the weights.

For the L2 regularization, $\Omega(\theta)$ is equal to $||\theta||_2^2$, while for the L1 regularization, $\Omega(\theta) = ||\theta||_1$. The L2 regularization reduces the magnitude of all the weights simultaneously, while the L1 regularization reduces the magnitude of the weights but inducing sparsity in the model (i.e. setting part of the weights to zero), thus promoting a type of feature selection.

Calculating the derivative of L1 and L2 with respect to θ yields the following expressions:

$$\frac{\partial\Omega}{\partial\theta_i} = \begin{cases} 2 \times \theta_i^{(t)}, & \text{if } \Omega(\theta) = ||\theta||_2^2 \\ sign(w_i^{(t)}), & \text{if } \Omega(\theta) = ||\theta||_1 \text{ and } \theta_i^{(t)} \neq 0 \end{cases} \tag{2.12}$$

Another way of regularization is the dropout technique which randomly drops units or neurons and its connections from the ANN during training time. According with

SRIVASTAVA *et al.*,2014 this process prevents units from co-adapting too much and it generates an exponential number of thinned neural nets created by the original architecture as presented in Figure 2.5. Among its benefits one may highlight: it is responsible for avoiding overfitting and improving convergence in really deep neural networks. At test time it approximates the effect of averaging the predictions of all thinned networks generated along the dropout process in the training phase. In other words, one can see this technique at test time as an ensemble method of all thinned networks increasing the generalization capability.

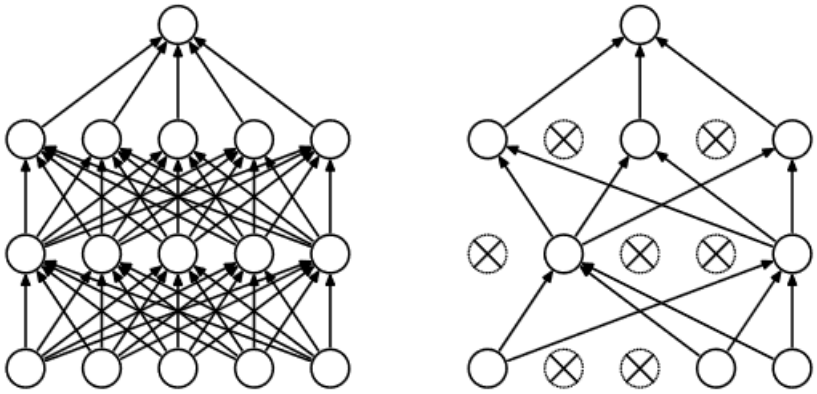


Figure 2.5 – Dropout Neural Net Model.Left: A standard neural net with 2 hidden layers.Right:An example of a thinned net produced by applying dropout to the network on the left.Crossed units have been dropped (SRIVASTAVA *et al.*, 2014)

2.2.3 Gradient Descent Optimization

With the rise of Deep Learning, the academic community has been developing a series of optimization algorithms for the gradient descent. Those algorithms aim to address some of the common challenges faced in minimizing highly non-convex error functions usually found in ANN's training process, such as: choosing a proper learning rate and getting trapped in suboptimal local minima of the error surface.

It will be provided a brief explanation of the two optimizer used in this research.

RMSprop

According to RUDER(2016), RMSprop is an unpublished adaptive learning rate method which was proposed by Geoff Hinton in Lecture 6e of his Coursera Class (HINTON, 2014).

The RMSprop follows a sequence of algorithms that compute adaptive learning rates for each parameter: Adagrad (DUCHI *et al.*, 2011) and Adadelta (ZEILER, 2012). The formulation of RMSprop's weight updates is:

$$\theta_{t+1} = \theta_t + \Delta\theta_t \tag{2.13}$$

The Δ term is derived from Adadelta and it is presented in Equation 2.14.

$$\begin{aligned} \Delta\theta_{t+1,i} = & \quad -\frac{\eta}{\sqrt{E[g^2]_{t,i}+\epsilon}} g_{t,i} \\ E[g^2]_{t+1,i} = & \quad 0.9E[g^2]_{t-1,i} + 0.1g_{t,i}^2 \end{aligned} \tag{2.14}$$

where $g_{t,i}$ is a short hand for the gradient descent with respect to every individual parameter $\theta_i \in \theta$ (i.e. $g_{t,i} = \nabla_\theta J(\theta, i)$) which confers the adapative cabability of RMSprop, Adadelta, Adagrad optimizers. According to ZEILER(2012), each dimension has its own dynamic rate. Since this dynamic rate grows with the inverse of the gradient magnitudes, large gradients have smaller learning rates and small gradients have large learning rates. This has a nice property, as in second order methods, that the progress along each dimension evens out over time.

The $E[g^2]_t$ term is an exponentially decaying average of the squared gradients depending on the previous average and the current gradient $E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma)g_t^2$. This average is a local estimate ensuring that learning continues to make progress even after many iterations of updates, which is an improvement compared to the Adagrad method that used a L2 norm of all past gradients which would vanish overtime.

Adaptive Moment Estimation

Adaptive Moment Estimation (Adam) is an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments (KINGMA; BA, 2014).

Adam also computes adaptive learning rates for each parameter. Moreover it stores an exponentially decaying average of past squared gradients v_t and an exponentially decaying average of past gradients m_t:

$$\begin{aligned} m_t = & \quad \beta_1 m_{t-1} + (1-\beta_1) g_t \\ v_t = & \quad \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \end{aligned} \tag{2.15}$$

m_t and v_t are the first (mean) and the second moment (uncentered variance) of the gradients. The authors of Adam initializes m_t and v_t as vectors of zeros. By doing

that, they observed that those parameters were biased towards zero specially on initial steps. To address this problem they proposed a new formulation of bias-corrected first and second moment estimates:

$$\begin{aligned} \hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\ \hat{v}_t &= \frac{v_t}{1-\beta_2^t} \end{aligned} \tag{2.16}$$

The weights updating rule assumes the form:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t \tag{2.17}$$

2.2.4 Monitoring training functions

With the availability of open source languages such as Python and the advent of deep learning, a series of interesting monitoring training functions that support the training process of ANN's has been developed. A considerable set of those functions can be found on packages such as Pytorch, Keras, TensorFlow or Caffe. For this research, three in special were used.

Early Stopping

The Early Stopping technique checks for every loss improvement in the validation set in comparison with the last iteration. In case it does not improve it starts counting until a threshold called patience (e.g. 15 epochs) is reached, thus promoting the interruption of the training process. In other words, the training process will continue as long as there is at least an improvement on the loss of the validation set every n epochs where n is known as patience threshold.

Model Checkpoint

The Model Checkpoint saves the weights of the ANN at every improvement of the validation loss. This is a simple technique, yet is really useful for keep tracking of improvement while testing different architectures and combinations of its parameters.

Learning Rate schedules

Another interesting technique used to assist the ANN's training process is the learning rate schedules. This method adjust the learning rate during training by e.g.

annealing, i.e. reducing, according to a pre-defined schedule such as a cosine function or when the optimization search process falls in a suboptimal minimum, reaching a plateau.

In fact, the biggest problems, pointed out by some authors like (DAUPHIN *et al.*, 2014), in optimizing highly non-convex error functions in high dimension space, such as ones usually found in ANN's training process, are the saddle points and not local minima. Those points are surrounded by high error plateaus that can dramatically slow down the learning process, and give the illusory impression of the existence of a local minimum.

Two learning rate annealing methods used on this research project were: Cyclical Learning Rate policy and Reducing Learning Rate on Plateau. A short description of those methods will be provided in what follows.

The Cyclical Learning Rate policy (CLR) proposed by (SMITH, 2015) uses a variant of learning rate annealing called stochastic gradient descent with restarts (SGDR), which gradually decreases the learning rate as training progresses. Based on a change of the learning rate from time to time configured by the schedule imposed, the search process are more prone to escape from the saddle points. Figure 2.6 shows how this behavior would look like in a known search space.

The period of the schedule imposed is configured by the number of iterations, which is the number of batches insides an epoch (e.g. a dataset with 100 samples and a batchsize of 10 samples would have 10 iterations). The authors suggest that the restart period of the schedule selected should be between 2 to 8 iterations. In short, for using CLR it is required to define the type of schedule (i.e. exponential, triangular, triangular2, etc..), the number of iterations for setting a cycle and the maximum and minimum values of the schedule function.

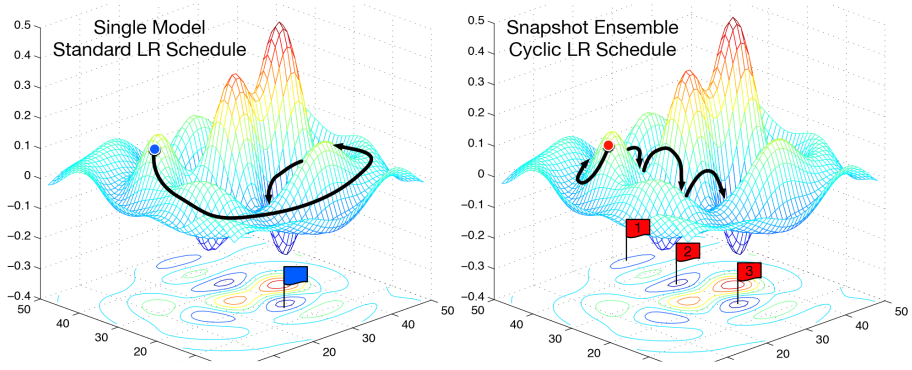


Figure 2.6 – Illustrative behavior of the Stochastic Gradient Descent with Restarts

The Reducing Learning Rate on Plateau also follows the same 'restarting' principle of SGDR. However it just reduces the learning rate by a pre-defined factor. This

reduction is done until the learning rate reaches a minimum value, also adjustable. After this minimum value for the learning rate is reached, the learning rate value is restarted to its original value moving on to another round of the method. The Reducing Learning Rate on Plateau is triggered by every time that there is no improvement on the validation loss for a predefined amount of epochs.

2.3 Local Interpretable Model-Agnostic Explanations

There is a latent demand for explaining the reason behind model's prediction for real applications. The academic community has been developing a series of approaches toward this goal. In fact the keyword regarding this subject on machine learning field is accountability, and can be found on proceedings like FAT (Fairness, Accountability and Transparency).

For this research, it was clear the necessity of explanations of the reasons for surges of Idletime at dumping areas in order to provide a lever of action to avoid or attenuate those scenarios before it happens. Consequently, it was used the Local Interpretable Model-Agnostic Explanations (LIME) (RIBEIRO *et al.*, 2016) to achieve this goal.

LIME is an algorithm that explains predictions of any machine learning model in a faithful way, by approximating it locally with an interpretable model. A brief explanation of the core concepts behind LIME will be provided in what follows.

The first important concept used in LIME is the interpretable data representation which uses representations that are understandable by humans. It is interesting to notice that those representations can vary from the original feature space which feed the model's predictions. An example given by the authors of LIME is the case of text classification where interpretable representations could be a binary vector representing the absence or presence of a word, whereas the classifier uses more complex and sometimes incomprehensible features such as word embeddings. The authors denote $x \in \Re^d$ as being the original representation of an instance being explained, while proposing $x^{'} \in \{0,1\}^{d'}$ as the binary vector of interpretable representations.

In the paper the authors define explanation as a model $g \in G$ where G is a class of potentially interpretable models such as decision trees, linear regressions, or failing rules list. Also, they provide visual or textual artifacts on the domain of interpretable representations $\{0,1\}^{d'}$. Even having the model's class G as highly interpretable, they measure the complexity of each model g as $\Psi(g)$ (e.g. depth of decision trees).

They denote the model being explained by $f : \Re^d \rightarrow \Re$ (e.g: for classifier f is the probability of x to belong to a certain class). Also, it is used a proximity measure

$\Pi_x(z)$ of an instance z to x in order to define the concept of locality around x. As a result, the explanations made by LIME is obtained by the following expression:

$$\xi(x) = \text{argmin}_{g \in G}\ \mathcal{L}(f, g, \Pi_x) + \Psi(g) \tag{2.18}$$

where $\mathcal{L}(f, g, \Pi_x)$ is a measure of how unfaithful g is in approximating f in the locality defined by Π_x.

The code for LIME is available on GitHub in a repository shared by the authors along with examples and a comprehensive documentation. The general steps of the application of LIME in Python is provided as follows.

First it is required to pass the data used to train the machine learning model, and an object containing the model already trained. LIME gathers information about the training data and stores it in the explainer object. The information is composed mainly of the type of features (i.e. numerical or categorical) and computes distributions of the features either using bins or mean and standard deviation, depending on the settings passed by the user.

For creating the explanations there is an object called *explainer()* that inherits attributes from the LIME object created on the first step. Then when an explanation is required by using a method of the *explainer* object, the prediction being explained is mutated creating by default 5000 new samples.

The type of mutation depends on some user settings and the data type of each feature. Then for each of the mutated samples a prediction is made using the trained machine learning model. In the next step all mutated samples are used to train another model $g \in G$ (e.g. linear model) only using a few features n_{features}. The number of features is selected through a couple of methods that can be selected by the users, the default is forward selection.

The model g is optimized by a greedy optimization strategy with objective function described in equation (2.18) and the feature importances are computed (e.g. the coefficients of the linear model). With this information the most important features will exhibit a positive or negative influence over the predicted value.

Those explanations that have a positive influence in each prediction are then considered explanatory attributes.

3 Methodology

In this chapter, it is presented the methodology adopted to develop the Oráculo system.The first section called Data mining explains how the features were collected. The next section denoted as Data Engineering explains which strategies were taken to preserve time coherence and join all relevant information available. The third section Data Cleaning, presents the process used to clean the data for training the models. The fourth section, Feature and Target Transformations, presents the scaling process done on the features and the target. The fifth section, Predictor's module, contains the approach and techniques used to develop, train, optimize, and select the models for building the Predictor module of Oráculo. The sixth section, Decision maker, discusses the strategy developed to trigger an alarm devoted to flagging a surge of Idletime. The seventh section, Explainer, discusses the use of LIME as an approach to stratify main contributors for the alarms generated by the Decision Maker's module.

3.1 Data mining

The surface mine is a highly dynamic environment where a huge variety of events can appear and disturb the productivity cycle causing Idletime or other problems not modeled in this project. Therefore, it was conducted a data mining process in order to identify possible reasons for the Idletime.

This process was performed along with the Mine Site Productivity's supervisors throughout a series of discussions, visits to the mine site in question and with the collaboration of the Information Technology area of the mining company involved. It was spent a considerable amount of time to understand all the different variables that somehow impact the system and where to retrieve them, from available datasets.

The resulting features are summarized in Table 3.1. The Litology and the Truck Fleet features are percentages of Loadtons and # Trucks respectively. For instance, in a time interval of one hour 1 thousand tons of material were dumped in a specific location, the composition of this amount of material is described in the Litology features (e.g. 800 tons of crispy hematite, 200 tons of calcite). In the same period 100 trucks were responsible for dumping this amount of material, those 100 trucks are stratified according with their fleet (e.g. 70 CAT 797 and 30 CAT 793D). Those two features work as a OneHot Enconding for all possible categories for the type of litologies and truck fleet available on the dataset.

For information security purposes the four original databases are discriminated in letters as well as the name of each litology and haul truck fleet.

Table 3.1 – Features description

Feature	Description	Databases sources
Loadtons	Amount of material being carried by all trucks[tons]	A
Litology	Type of material being carried [tons]	A
# Trucks	Number of total trucks that dumped on the Crusher	A
Truck Fleet	Number of trucks stratified by fleet	A
WTCS	Waiting time for Crusher Signal	B
CWCS	Count of waiting Crusher Signals	B
WTML	Waiting time caused by Mill Failures	D
CML	Count of Mill Failures	D

3.2 Data Engineering

There was no previous primary key in the 4 databases (i.e. A, B, C and D) which could be used to join all of this information on the same grain of analysis. Nonetheless, all of them were discrete events associated with a start time and end time of that specific event. For example, the main database called here as A contained the productivity cycle information such as: loading time, dumping time, Idletime, the amount of material carried by the truck as well as its litology, among others, and all of those events were associated with a specific time in seconds of a shift as presented in Figure 3.1 where it was chosen only two features in a morning shift.

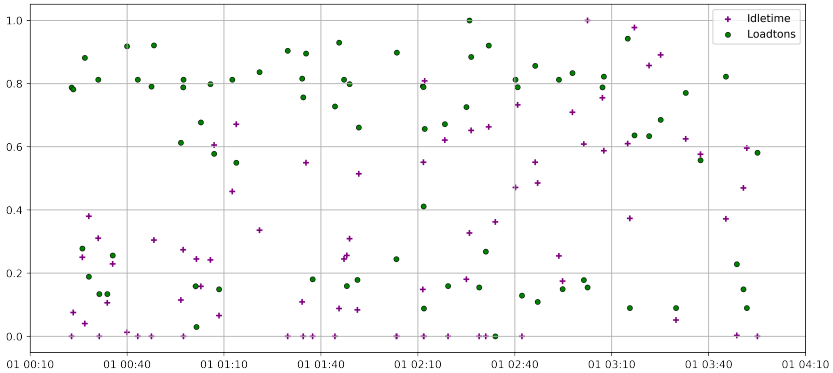


Figure 3.1 – Examples of features idletime and loadtons before aggregating

For addressing this problem, it was selected a set of time interval (e.g. every 30 minutes, every hour and every two hours) to aggregate all features of interest in each database separately, thus resulting in a primary key making possible the join of information in a time interval of analysis. This process is elucidated in Figure 3.2 where the same

features presented in Figure 3.1 were aggregated in a time interval of 2 hours. All of the features in both figures were scaled using normalization.

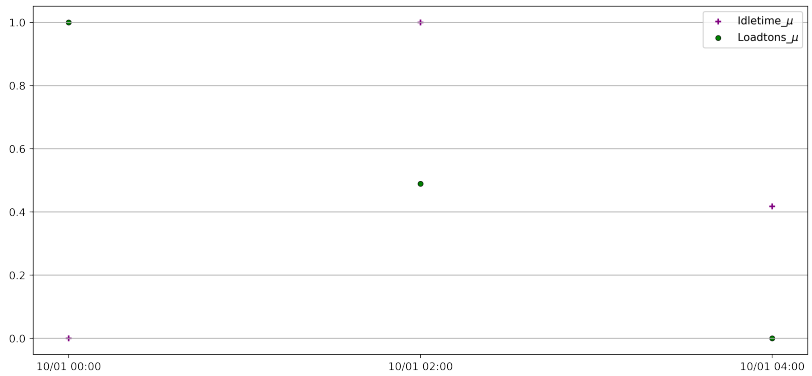


Figure 3.2 – Examples of features after aggregating

The loss of information while aggregating the features was inevitable and necessary to build a primary key to join all the relevant information from the different data sources. A strategy applied to minimize this loss of information was done by taken the average and standard deviation from each continuous feature aggregated on the desired time interval.

There were 4 types of litologies available on the dataset, 3 types of fleet, plus the 6 other features described in Table 3.1 yielded a total of 13 raw features.By aggregating and extracting the average and standard deviation of each raw feature excepted by # Truck and CWCS which were used only the sum, the resulting number of features were a 25.

This whole process was automatized in a code written in Python in order to facilitate the choice of the time interval to aggregate, to join the databases and to select which features would be brought from each one of the databases. The time intervals studied were 30 minutes, 1 hour and 2 hours. The discussion about of each time interval tested is presented in the Subsection 4.1.1 in Results chapter.

3.3 Data Cleaning

Evaluating the quality of the data it is a crucial step for training machine learning models. In real world applications, the data available often presents undesirable information which is produced due to different reasons such as: sensors or system failures, erroneous handmade annotations and abnormal events which have also to be properly treated depending on the type of the modeling process. This analysis was carefully done for each data source available before aggregating and joining the information. Also, the

distribution of the continuous values were presented and analyzed by the mine site supervisors in order to detect strange patterns.

After aggregating and joining all the information, it was noticed the presence of outliers which were then treated. The results of the approach taken are presented in Subsection 4.1.2 in Results chapter.

3.4 Feature and Target Transformations

An important step in training machine learning models is the features scaling process. This strategy is required for convergence purposes and to avoid feature predominance in models based on weights which somehow combine features (e.g. Logistic Regression, Linear Regression, ANN, Suport Vector Machines, etc..). In fact, not all machine learning models need feature matrix scaled such as Decision trees based algorithms (e.g. Random Forests, XGBoost, GBM, etc..).

In this research project the features were scaled using normalization approach as presented in Equation (3.1).

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{3.1}$$

Also the target was scaled using a $log(1 + y)$, which is a known strategy when working with time series problems. The models were trained and validated using this features and target transformations until a plateau was reached. The comparison of results obtained is presented in Results chapter.

After several attempts on improving the machine learning models performance, it was found that the most difficult scenario to forecast was the Idletime's oscillations specially the higher peaks. To address this situation it was used a Savitzky–Golay's smoothing filter implemented on the library *scipy.signal* in Python.

The Savitzky filter increases the signal-to-noise ratio preserving the pattern of the signal. This objective is accomplished in a convolution process by fitting successive sub-sets of adjacent data points with a low-degree polynomial by the method of linear least squares (SAVITZKY; GOLAY, 1964).

It was tested different sizes of window and polynomial orders that could provide a better fitting for the model without loosing the pattern studied. Figure 3.3 exhibits the smoothing process with the resulting choice of a window of 5 samples and a 3 degree polynomial.

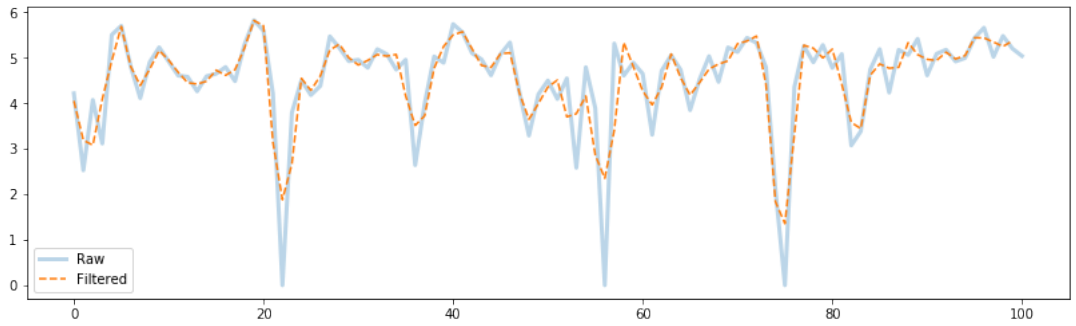


Figure 3.3 – Smoothing the data with Savitzky filter

3.5 Predictor's module

The goal of the Predictor's module of Oráculo system is to forecast the average of Idletime in the next time interval and not to predict its value on the current time. The prediction on the current time would be useless for the mine site operators since the event would have already happened by the time of acquiring and aggregating features. Therefore it was used an auto-regression approach with exogenous inputs for addressing this problem as discussed in Section 2.1.

For choosing the ideal amount of time steps (i.e. (p) at Equation (2.1)) required for forecasting the Idletime in the next time interval, it was done an analysis of auto-correlation plots of the target as well as a Pearson correlation among lagged features an the target being forecast. This discussion is presented in Subsection 4.1.4 in Results chapter.

After concluding this analysis, the resulting data available was split in a hold-out methodology (i.e. 80% for devising the models and 20% most recent data for test). The test set was let untouched during the devising models process reserving it only for performance comparison purposes. Then the 80% of the original database was divided again in a holdout methodology creating a training set and a validation set used to tune the devised machine learning models.

Two metrics were chosen as performance criterion for comparison of the obtained model: the Mean Absolute Error (MAE) demonstrated in Equation 3.2, and the Relative Root Mean Square Error (RRMSE), demonstrated in Equation 3.3.

$$\mathrm{MAE}(\theta) = \frac{1}{N}\sum_{l=1}^{N}|\hat{y}_l - y_l| \tag{3.2}$$

$$\mathrm{RRMSE}(\theta) = 100 \times \frac{\sqrt{\frac{1}{N}\sum_{l=1}^{N}(\hat{y}_l - y_l)^2}}{\mu_{y_l}} \tag{3.3}$$

where μ_{yl} is the mean value of the actual target being predicted.

The RRMSE can be seen as a performance comparison against the mean value of the target being predicted, in other words it is a sense of how better is the prediction of the model being evaluated against a straight line crossing the median value of the target. According with the literature (SHCHERBAKOV *et al.*, 2013; DESPOTOVIC *et al.*, 2016), a model performing a RRMSE $< 10\%$ is considered excellent, while $10\% <$ RRMSE $< 20\%$ is good, fair if $20\% <$ RRMSE $< 30\%$ and poor if RRMSE $> 30\%$.

A series of models were developed with the aim of reproducing new realizations of Idletime. The first model developed was a Linear Regression with L1 (Lasso) which objective function presented in Equation (3.4).

$$\min_{w} \frac{1}{2n_{\text{samples}}} ||Xw - y||_2^2 + \alpha ||w||_1 \tag{3.4}$$

where α is a constant and $||w||_1$ is the L1-norm of the parameter vector.

For finding the best α it was used a Grid Search approach with predefined sets of values which are elucidated in Subsection 4.2 in Results chapter.

The second model developed was a NARX model using an ANN. The ANN architecture devising process was done by trial and error. It was started with a simple MLP with one hidden layer of 100 neurons. The number of neurons and layers were adjusted step by step observing the performance on the validation set. It was also found that the ReLu activation function provided a better performance than the Sigmoid activation. Furthermore, it was used the ADAM optimzer, L2 regularization, two training monitoring functions and a Learning rate annealing function.

3.6 Decision maker

The Decision maker module was developed to alert the mine site operators of the possibility of Idletime's surge in front of the dumping area. Two strategies were developed to create this alarm devoted to flagging a surge of Idletime.

First it was created a simple threshold Φ as presented in Equation (3.5) based on the median γ value of Idletime extracted from the historical data used to train the Predictor module. Every time that the Predictor module forecasted a value greater than this threshold minus an ξ parameter to account for the error associated with the forecast an alarm was triggered. The idea behind the choice of the median value as a threshold was because there was no previous definition from the mine site supervisors of how to identify a surge of Idletime, so if the value forecasted was greater than the most common value

faced by the haul trucks it was considered a surge of Idletime. The results are discussed in the Results chapter.

$$\Phi = \gamma - \xi \tag{3.5}$$

In the second attempt it was developed a new threshold Θ obtained by the average value of the last two Idletime events as follows:

$$\Theta = \frac{\text{idletime}(t) + \text{idletime}(t-1)}{2} \tag{3.6}$$

In case of the output provided by the Predictor module be greater than Θ, a flag is activated (i.e a Boolean variable equals True) informing the user of a possible surge of Idletime in the next time interval.

For verifying the results of the Decision maker, a series of tests were conducted during the month of October 2018 in a real mine site located in Brazil. Since the Decision maker would detect a surge of Idletime based on the criterion developed it can be used classification metrics to evaluate the overall performance of the Decision Maker's module.Therefore, it was used the confusion matrix as presented in Table 3.2, F1 Score exhibited in Equation (3.9), Recall presented in Equation (3.8) and Precision presented in Equation (3.7).

Table 3.2 – Confusion Matrix

		Actual	
		Positive	Negative
Prediction	Positive	$TruePositive(TP)$	$FalsePositive(FP)$
	Negative	$FalseNegative(FN)$	$TrueNegative(TN)$

$$P = \frac{TP}{TP + FP} \tag{3.7}$$

$$R = \frac{TP}{TP + FN} \tag{3.8}$$

$$F_1 = 2 * \frac{P * R}{P + R} \tag{3.9}$$

The interpretation of Precision P for this problem is the rate of surges of Idletime detected that was indeed surges of Idletime, while Recall R can be interpreted as the capability of detecting all the existing surges of Idletime in the period evaluated. The F_1 score is the harmonic average of Precision and Recall. All values of these metrics range from 0 to 1. The closer the value to 1 better the performance.

3.7 Explainer

Every time an alert is generated by the Decision Maker's module, that specific sample of the features is passed to the LIME system, so it can indicate the top 3 features that had a positive influence on the behavior of the Predictor module. In other words, LIME explanations are used to provide the presumed reasons for the surge of Idletime created by the Decision Maker's module.

Aiming at investigating the performance in a real scenario, a series of tests were made in a staging environment (i.e. a nearly exact replica of the production environment) assisted by the mining supervisors. Basically, it was provided the surge of Idletime predictions 2 hours ahead along with the LIME stratification in case of alert. With those information the users can take actions to minimize the surge of Idletime events at the dumping area studied. Those tests were important to evaluate the overall performance of the system against the real scenario of the mine site studied and collect the feedback for enhancing the system performance.

4 Results

This chapter presents the results achieved by the Oráculo's module using data from a real mine site located in Brazil. The flow chart of the three Oráculo's modules is exhibited in Figure 4.1 where UI refers to user interface developed by a specialized team.

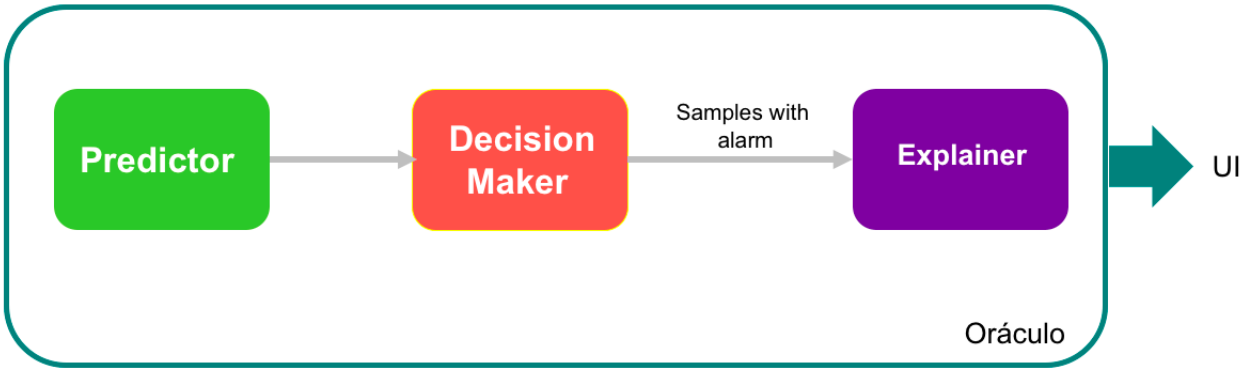


Figure 4.1 – Flow chart of the Oráculo System

Moreover, this chapter contains a section denoted as Data Preprocessing where it is presented all the data cleaning and data engineering approaches taken to prepare the data for devising the Predictor's module.

4.1 Data Preprocessing

4.1.1 Data Engineering

By testing different aggregating time intervals, it was noticed that the smaller the grain of analysis the more sparse and noisy was the resulting time series of Idletime, turning burdensome the forecasting process. Figure 4.2 exhibits this pattern for the following aggregating time intervals 30 minutes, 1 hour and 2 hours.

This matter of choosing the appropriate aggregating time interval was also discussed with the mine site supervisors for getting their perspective of what would be the best time interval to get a prediction of Idletime's surge events in such a way that it was possible to take actions for preventing this scenario. As a result, the optimal time interval was found to be every 2 hours which also provided a better convergence for the Predictor's module.

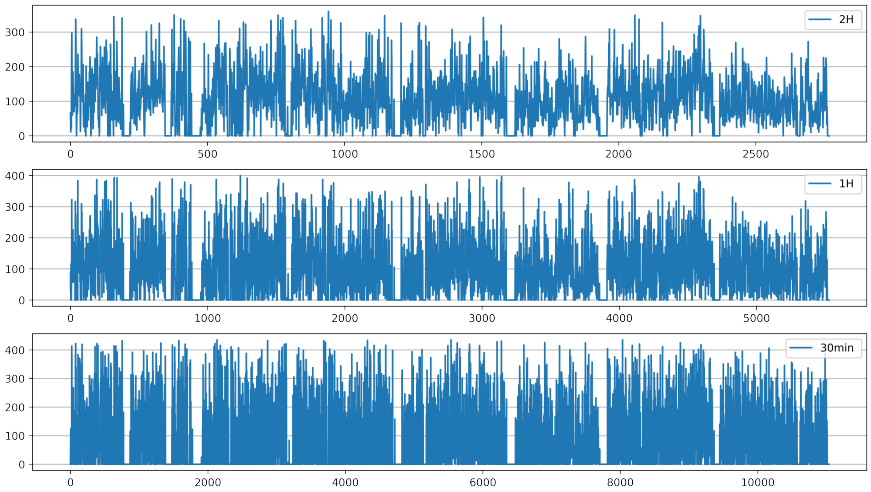
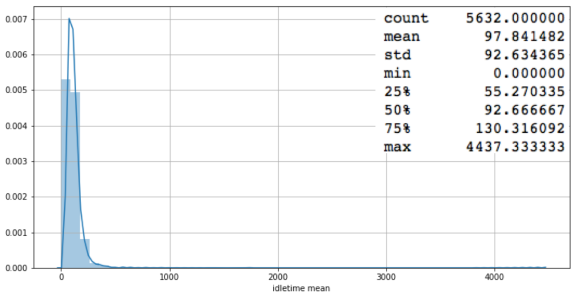


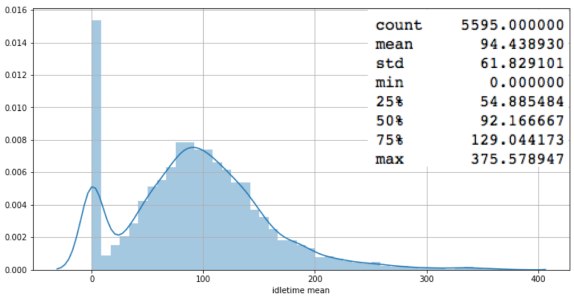
Figure 4.2 – Examples of Aggregating Time intervals

4.1.2 Data Cleaning

The distribution of Idletime was analyzed and it was noticed the presence of outliers. In order to deal with this problem, every value greater than the mean plus three standard deviations were removed. The result of this approach is exhibited in Figure 4.3.



(a) Before



(b) After

Figure 4.3 – Treatment of Outliers

The number of lost examples in this process was only 37.

Furthermore, the failures were analysed selecting only those which exhibited influence on the Idletime. This filtering process was also assisted by the mine site super-

visor. The list of failures are not exhibited due to information security purposes.

4.1.3 Feature and Target Transformations

After cleaning and adjusting the aggregating frequency of the data, the resulting feature matrix was scaled using a normalization approach as presented in Equation 3.1.

Another strategy adopted that provided a good convergence for the models was to normalize the target (i.e. the average of Idletime every 2 hours) using $log(1+y)$. The resulting distribution is presented Figure 4.4.

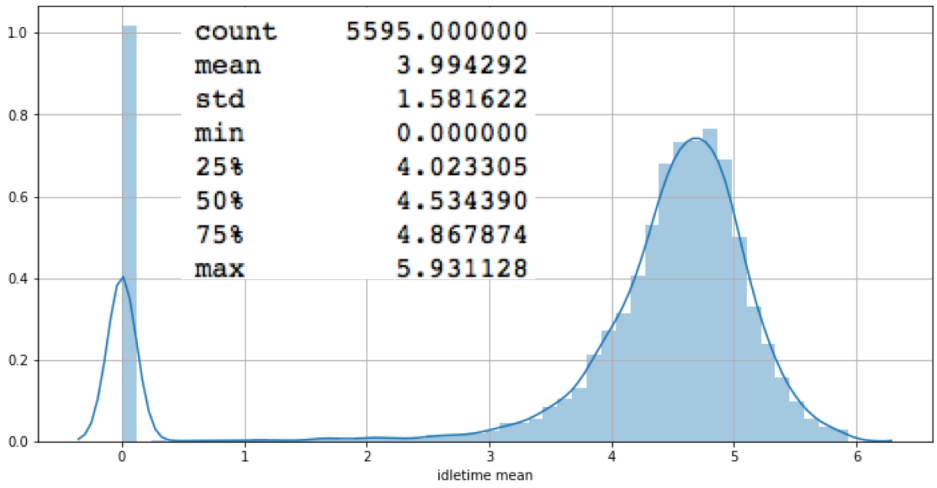


Figure 4.4 – Target normalized using $log(1+y)$

4.1.4 Analysis of the number of lags

Aiming at choosing the number of lags used in the autoregressive models two approaches were adopted. First it was analyzed the autocorrelation and partial autocorrelation graphs as presented in Figure 4.5. The QQ Plot and the Probability Plot, also presented in Figure 4.5, demonstrate how close is the the distribution of the time series analysed compared to a normal distribution

Based on this analysis, it was chosen 6 lags for forecasting the next Idletime. Later some tests were done with lag values greater than 6 which did not produced any improvements on the performance of the Predictor module.

The same number 6 of lags were also applied for the rest of features resulting in a feature matrix containing the 25 features at time t with its 6 lagged samples $t-1, \ldots, t-5$ plus the lagged samples of the target (i.e. mean and standard deviation of Idletime). The

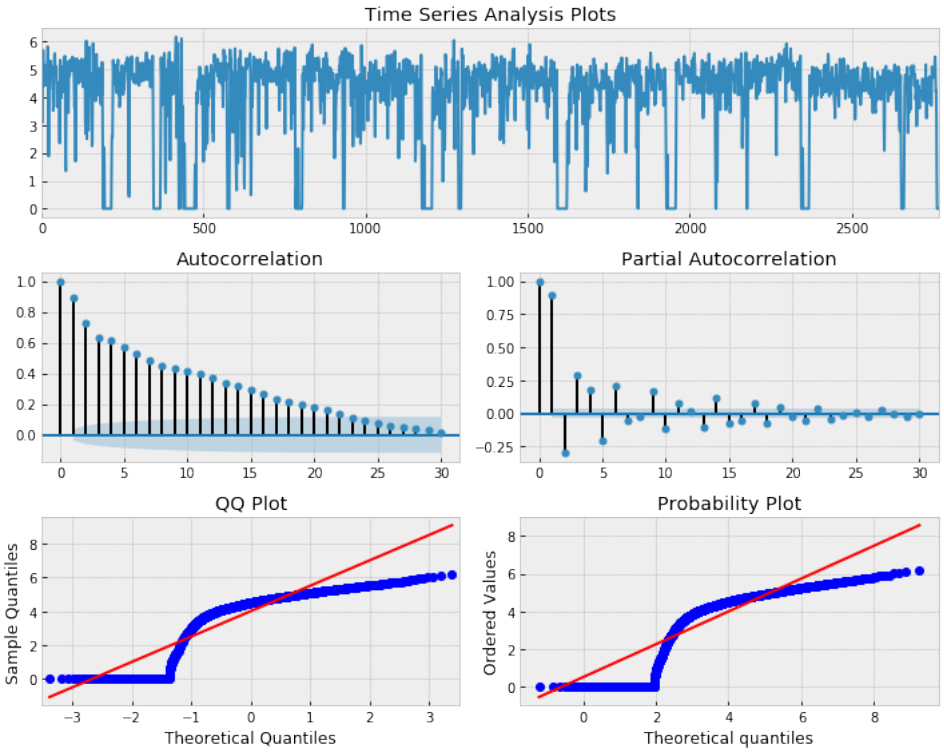


Figure 4.5 – Analysis of Idletime's average every two hours normalized using $log(1 + y)$

structure of this matrix is similar to the red box presented in Figure 2.2. This lagged samples choice resulted in a feature matrix of dimensions $182 \times n$ (i.e. 26×7).

For selecting the features, it was analyzed the Pearson's correlation between the Idletime at the next time interval (i.e. $t + 1$) with the features at the past (i.e. $t, t - 1, \ldots, t - 5$). As a result, it was derived a vector of dimension 182×1 with each row containing a value ranging from -1 to 1 also known as the statistical significance or p-value. Finally it was selected all features that exhibits a p-value less than -0.3 or greater than $+0.3$ resulting in a total of 111 features for devising the Predictor's module.

4.2 Predictor module

The goal of this module it to forecast the average of Idletime 2 hours ahead. For accomplishing this, different models were devised and evaluated.

The first model devised was a simple Linear Regression trained with L1 (Lasso) regularizer using *scikit-learn* library in Python.

For finding the best α it was used a Grid Search approach with predefined sets of values. The results are displayed on Table 4.1

After choosing the α that provided the best result on the validation set, the

Table 4.1 – Lasso Results

α	MAE Train	MAE Validation
3.0	1.20	0.91
1.0	1.01	0.79
1e−1	0.81	0.67
1e−2	**0.74**	**0.61**
1e−3	0.72	0.65
1e−4	0.72	0.67

performance of model was evaluated on the test set. As a result, it was obtained a MAE of 0.71. To help getting a better sense of performance it was also used the RRMSE. This metric yielded a value of 26% which is considered on the literature a reasonable model. Also, the predicted versus the actual value on the test set is provided on Figure 4.6.

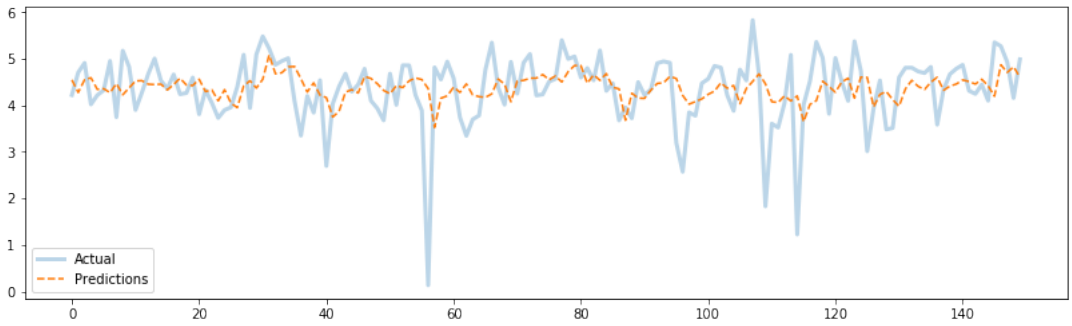


Figure 4.6 – Lasso test set results

Although the Linear Regression with Lasso regularizer provided a reasonable result on the test set (i.e. MAE of 0.71 and 26% of RRMSE), a non-linear model was also devised aiming for improving this performance.

The ANN architecture was devised by trial and error observing the performance on the validation set. The resulting architecture is presented in Figure 4.7.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_25 (Dense)             (None, 100)               19700
_________________________________________________________________
dense_26 (Dense)             (None, 300)               30300
_________________________________________________________________
dense_27 (Dense)             (None, 100)               30100
_________________________________________________________________
dense_28 (Dense)             (None, 1)                 101
=================================================================
Total params: 80,201
Trainable params: 80,201
Non-trainable params: 0
_________________________________________________________________
```

Figure 4.7 – ANN architecture

Aiming at training the ANN's weights it was used the Adam optimizer with a learning rate of 10^{-3} and a weight decay (i.e. L2 regularizer) of 10^{-5}. Two callbacks were used to monitor the training process: *EarlyStopping* and *ModelCheckPoint*. It was also found that by using the Cyclical learning rate policy with an exponential annealing it was possible to extend the number of epochs during the training process resulting in a better performance on the test set. Figure 4.8 demonstrates the model convergence during the training process.

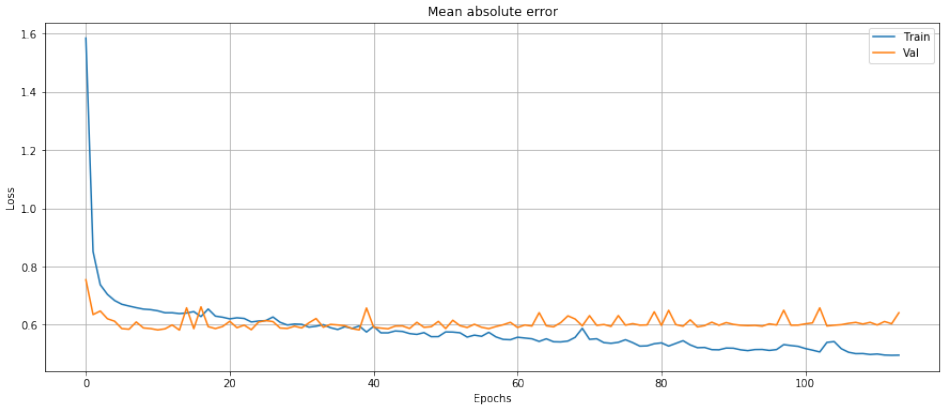


Figure 4.8 – ANN Training process

The MAE on the test set achieved with this approach was 0.66 with a RRMSE of 26%. Even though the RRMSE of both models were equal, the MAE on the test set of the ANN 0.66 was slightly better than the Lasso Regression 0.71. Also, the predicted versus the truth value on the test set is provided in Figure 4.9.

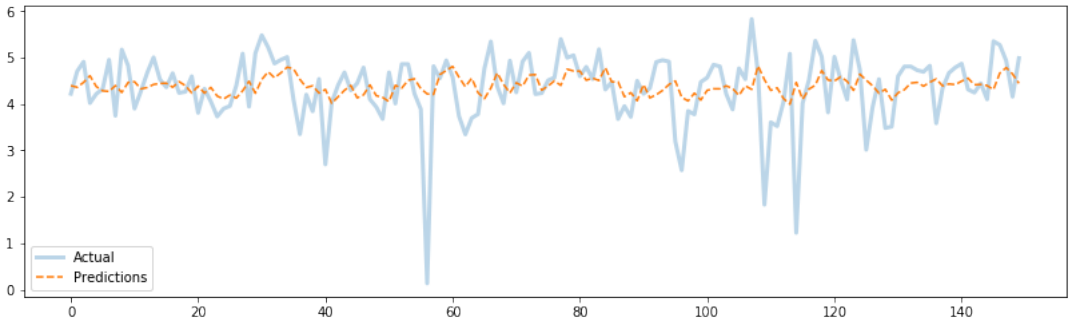


Figure 4.9 – ANN test set results

Using the target filtered by Savitzky filter, the models were retrained following the same steps previously discussed. The Lasso Regression with $alpha$ = 1e−2 yielded a MAE of 0.38 and a RRMSE of 13% on the test set. Figure 4.10 exhibits the predicted versus the actual values on the smoothed test set.

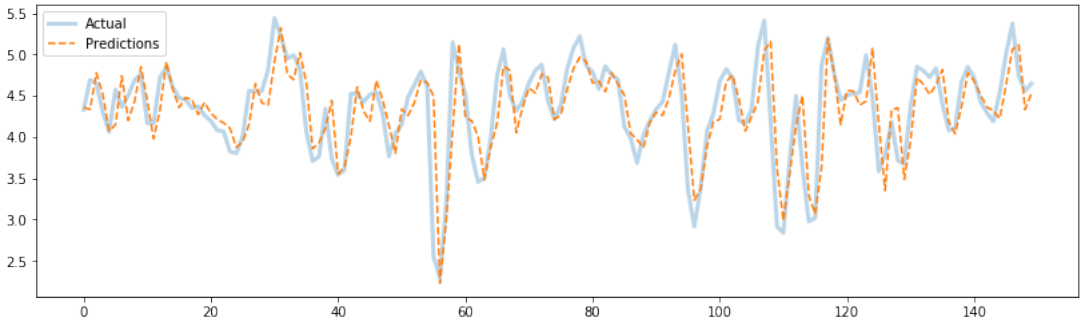


Figure 4.10 – Test set results for Lasso model with target smoothed

The ANN was trained using the same architecture presented in Figure 4.7 achieving a MAE of 0.32 and 11% of RRMSE on the smoothed test set, which is the best result achieved with the models devised. Figure 4.11 presents the predicted versus the actual values on the smoothed test set.

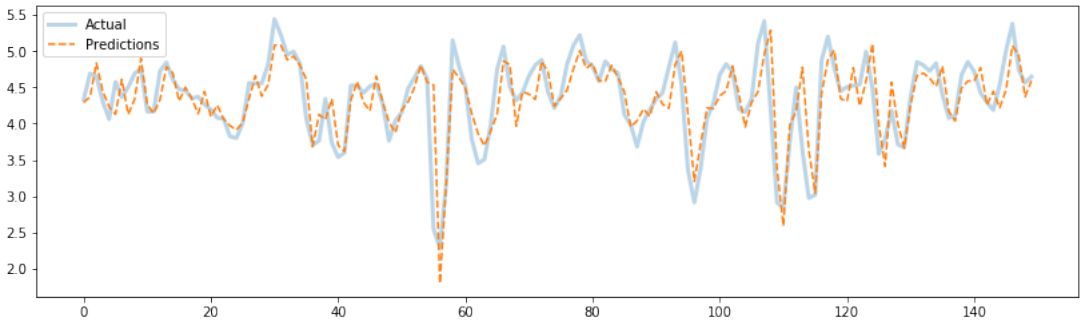


Figure 4.11 – Test set results for ANN model with smoothed target

The summary of the obtained results considering all the approaches described in this section are disposed in Table 4.2.

Table 4.2 – Summary of the obtained performance

		MAE	RRMSE
Noisy target	Lasso	0.71	26%
	ANN	0.66	26%
Smoothed target	Lasso	0.38	13%
	ANN	**0.32**	**11%**

Besides the best performance reached by using the ANN according to the metrics evaluated, it was also observed that the ANN could better anticipate higher peaks when compared to the Lasso model. This is a really important capability of the model, since the main goal of Oráculo system is forecasting surges of Idletime (i.e. higher peaks). An example of this situation is illustrated in Figure 4.12 on the peak between the 15th and 25th samples.

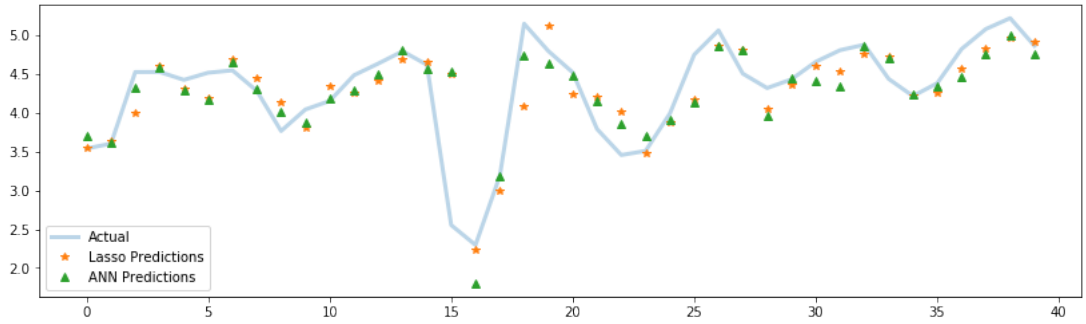


Figure 4.12 – Comparison of the performance of the proposed models for forecasting higher peaks

4.3 Decision maker

The first threshold Θ based on the mean value of the last two average of Idletime faced by the haul trucks, presented in Equation (3.6) yielded the results in Figure 4.13 and Table 4.3.

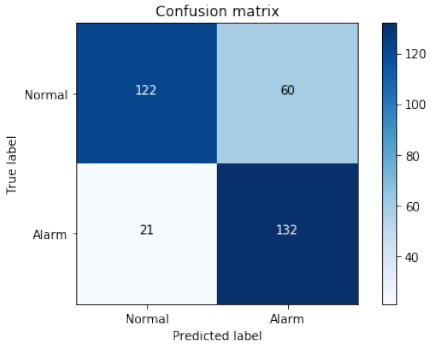


Figure 4.13 – Confusion Matrix for Θ

Table 4.3 – Results for Θ

Metrics	Value
F1-score	0.76
Precision	0.68
Recall	0.86

The second threshold Φ presented in Equation (3.5) yielded the results in Figure 4.14 and Table 4.4.

The two thresholds identify what would be a surge of Idletime creating a binary classification problem by using the forecasts of the Predictor module. In fact, each threshold has its particularities. Θ can be seen as a moving average that would detect a trend with the past two Idletime values, while Φ detects the Idletime that differs from the most common value on the historical data used to train the Predictor module.

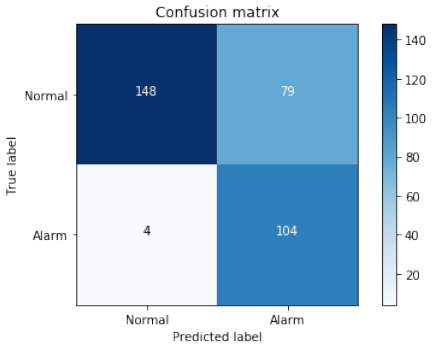


Figure 4.14 – Confusion Matrix for Φ

Table 4.4 – Results for Φ

Metrics	Value
F1-score	0.71
Precision	0.57
Recall	0.96

The Θ threshold is flexible and carries local information. However, it still consider low values of Idletime as a surge which in reality is not true.

With that interpretation in mind and after a series of discussions with the Mine Supervisors and comparative tests, it was chosen the Φ threshold for identifying a surge of Idletime. Additionally, a fact that contributes for this decision was in general the smaller number of False Negatives resulted when using the Φ threshold versus the ones yielded by using Θ.

4.4 Explainer module

The Explainer module is constituted mainly by LIME and it is activated every time an alarm of surge of Idletime is generated by the Decision Maker module. Then it is taken the top 3 features that had a positive influence on the output provided by the Predictor module. In other words, the features used to forecast the average of Idletime two hours ahead inform the presumed reason for the forecast that generated alarm.

The messages generated at the final user interface was created based on the users feedback in order to make it easier to understand and feasible to use. The final version was omitted from this thesis due to information security purposes. However, Figure 4.15 illustrates the idea behind the user interface as well as a real example, provided by the Mine Supervisors, of a successful surge of Idletime detected .

The column "Time of prediction" contains the time which the Oráculo system

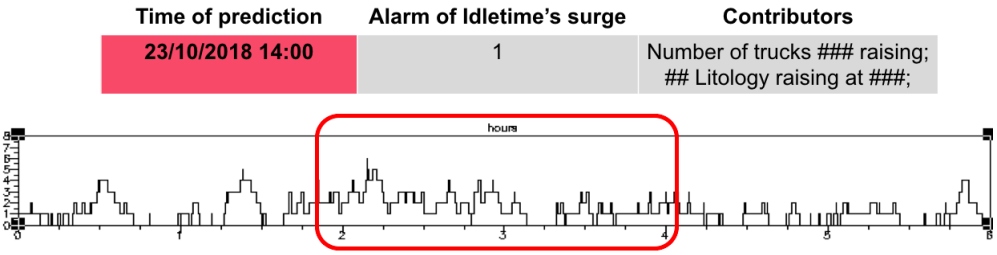


Figure 4.15 – A fictitious example of the Oráculo's operation

was activated meaning that the surge of Idletime will happen or not in the next 2 hours. If the color is red it means that the decision maker generated an alarm which will be confirmed by the value 1 in the column "Alarm of Idletime's surge". Finally the contributors found by LIME are disposed in a list in the column "Contributors" which refers to the features used to train the Predictor‘s module.

The graph right below the three columns is a metric from the Mine Supervisor that confirmed a Surge of Idletime indicated by Oráculo system on the given hour (i.e. the peak around 2:15) in a test made on a real mine site located in Brazil.

Conclusion

This dissertation presents and analyzes a successful application of an interpretable machine learning solution which is currently being used in a mining company in Brazil. The contact with the mine site supervisors and the visits made to the mine enhanced the capability of tracking good features to forecast the surge of queue events at the dumping area and greatly improved the overall understanding of the problem.

The Data Mining, Engineering and Cleaning processes were undoubtedly the most time consuming phase of this research project. Being able to automatize and develop a well documented code for this pipeline made feasible testing different groups of features and observing the impact of each one. Moreover, it contributed for later transferring this solution to a production environment.

Another important strategy of the methodology adopted was the auto-regressive perspective of the time series models. This strategy has opened a variety of possibilities in many practical problems faced on the daily work life in industrial plants with complex process to be monitored. Every problem that has a correlated time sequence can be benefited from the concepts here applied and discussed.

The power of Artificial Neural Networks with the state of the art techniques to augment the training processes were indeed essential. In special the supervised learnability of a function that maps the correlation among the target and all the features acquired with an associated error. This characteristic made it possible to overcome the tight schedule imposed by the stakeholders delivering a robust forecast model. Additionally, the capacity of forecasting higher peaks of Idletime besides the superior performance compared to the Lasso regression was fundamental.

The rule based system behind the Decision Maker module, opened a range of new possibilities for adding new features (e.g. new information from sensors, images, etc...) in the surge of Idletime identification process. This flexibility does not require historical data and can be updated easily as the mine site changes over time. This capacity outperforms the usage of a classifier machine learning model which would demand a historical data for learning the pattern in case.

The explanatory feature of the Oráculo's system granted by the LIME algorithm provided a leaver of action for the forecasts and the alarms generated by the Predictor and Decision Maker modules. This capacity is crucial and turned the forecasting followed by an automatic explanation more useful for the final users.

Future work and perspectives

There are some points of improvement identified by the author and summarized on the list below. Those points will be studied and might be applied for next projects depending on the results achieved.

- Usage of Recurrent Neural Networks or even Convolutional Neural Networks for improving the Predictor's module;
- Usage of ensemble or Meta Learning techniques for combining the forecast of different models aiming at improving the generalization capability;
- Selection of the number of lags in multivariate time series is a combinatorial analysis that creates an opportunity for developing a heuristic model to accomplish this goal.

Bibliography

BECKMAN, R. Haul trucks in australian surface mines. *2012: Australia: p. 87-96.*, 2012. Citado na página 15.

CHIRONIS, N. P. Coal age operating handbook of coal surface mining and reclamation. In: _____. [S.l.: s.n.], 1978. Vol. 6., cap. Alberta: Coal Age Mining Informational Services:, p. p. 153–169. Citado na página 15.

CLEVERT, D.; UNTERTHINER, T.; HOCHREITER, S. Fast and accurate deep network learning by exponential linear units (elus). *CoRR*, abs/1511.07289, 2015. Disponível em: <http://arxiv.org/abs/1511.07289>. Citado na página 21.

CYBENKO, G. Approximation by superpositions of a sigmoidal function. *Signals and Systems*, vol. 2, p. pp. 303–314, 1989. Citado na página 19.

DAUPHIN, Y. N.; PASCANU, R.; GÜLÇEHRE, Ç.; CHO, K.; GANGULI, S.; BENGIO, Y. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In: *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*. [s.n.], 2014. p. 2933–2941. Disponível em: <http://papers.nips.cc/paper/5486-identifying-and-attacking-the-saddle-point-problem-in-high-dimensional-non-convex-optimizat Citado na página 28.

DESPOTOVIC, M.; NEDIC, V.; DESPOTOVIC, D.; CVETANOVIC, S. Evaluation of empirical models for predicting monthly mean horizontal diffuse solar radiation. *Renewable and Sustainable Energy Reviews*, Elsevier BV, v. 56, p. 246–260, apr 2016. Disponível em: <https://doi.org/10.1016/j.rser.2015.11.058>. Citado na página 36.

DUCHI, J. C.; HAZAN, E.; SINGER, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, v. 12, p. 2121–2159, 2011. Disponível em: <http://dl.acm.org/citation.cfm?id=2021068>. Citado na página 26.

DURBIN, J.; KOOPMAN, S. J. *Time Series Analysis by State Space Methods*. [S.l.]: Oxford university press, 2012. Citado na página 17.

ERCELEBI, S.; BASCETIN, A. Optimization of shovel-truck system for surface mining. *Journal of the Southern African Institute of Mining and Metallurgy*, scieloza, v. 109, p. 433 – 439, 07 2009. ISSN 2225-6253. Disponível em: <http://www.scielo.org.za/scielo.php?script=sci_arttext&pid=S2225-62532009000700006&nrm=iso>. Citado na página 14.

GIESBRECHT, M. *PROPOSTAS IMUNO-INSPIRADAS PARA IDENTIFICAÇÃO DE SISTEMAS E REALIZAÇÃO DE SÉRIES TEMPORAIS MULTIVARIÁVEIS NO ESPAÇO DE ESTADO*. Tese (Doutorado) — School of Electrical and Computer Engineering, UNICAMP, 2013. Citado na página 17.

HAMMOOD, A. S.; MAHDI, H. Development artificial neural network model to study the influence of oxidation process and zinc-electroplating on fatigue life of gray cast iron.

International Journal of Mechanical & Mechatronics Engineering 12(74), 2012. Citado na página 19.

HARTMAN, H. L.; MUTMANSKY, J. M. *Introductory Mining Engineering*. [S.l.]: NJ: John Wiley & Sons, 2002. Citado na página 14.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015. Disponível em: <http://arxiv.org/abs/1502.01852>. Citado na página 21.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. [s.n.], 2016. p. 770–778. Disponível em: <https://doi.org/10.1109/CVPR.2016.90>. Citado na página 21.

HINTON, G. Neural networks for machine learning. Lecture 6a Overview of mini-batch gradient descent. 2014. Disponível em: <http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf>. Citado na página 25.

KE-LUXIANG; PU-YUXIANG; YOU-PINGWU. Prediction of the fatigue life of natural rubber composites by artificial neural network approaches. *Materials & Design*, Volume 57, p. Pages 180–185, 2014. Citado na página 19.

KECOJEVIC, V.; KOMLJENOVIC, D. Haul truck fuel consumption and co2 emission under various engine load conditions. *Mining Engineering*, 2010. Citado na página 15.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. Disponível em: <http://arxiv.org/abs/1412.6980>. Citado na página 26.

KLAMBAUER, G.; UNTERTHINER, T.; MAYR, A.; HOCHREITER, S. Self-normalizing neural networks. *CoRR*, abs/1706.02515, 2017. Disponível em: <http://arxiv.org/abs/1706.02515>. Citado na página 21.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, v. 60, n. 6, p. 84–90, 2017. Disponível em: <http://doi.acm.org/10.1145/3065386>. Citado na página 21.

MAAS, A. L.; HANNUN, A. Y.; NG, A. Y. Rectifier nonlinearities improve neural network acoustic models. In: *Proceedings of the 30th International Conference on Machine Learning,*. [S.l.: s.n.], 2013. Citado na página 21.

NASHVER, K.; SIGHBIN, J. Improving the organization of the shovel-truck systems in open-pit coal mines. *Jacksonville, USA: p. 31-36*, 2007. Citado na página 15.

PARUNAK, H. V. D. Book review: Neural networks for pattern recognition by christopher m. bishop (clarendon press, 1995). *SIGART Bulletin*, v. 9, n. 1, p. 41–43, 1998. Disponível em: <http://doi.acm.org/10.1145/294828.1067910>. Citado na página 19.

RAHIMI-GORJI, M.; GHAJAR, M.; KAKAEE, A.-H.; GANJI, D. D. Modeling of the air conditions effects on the power and fuel consumption of the si engine using neural networks and regression. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, Volume 39, p. pp 375–384, 2017. Citado na página 19.

RIBEIRO, M. T.; SINGH, S.; GUESTRIN, C. "why should I trust you?": Explaining the predictions of any classifier. *CoRR*, abs/1602.04938, 2016. Disponível em: <http://arxiv.org/abs/1602.04938>. Citado na página 29.

RUDER, S. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016. Disponível em: <http://arxiv.org/abs/1609.04747>. Citado 2 vezes nas páginas 23 and 25.

SAVITZKY, A.; GOLAY, M. J. E. Smoothing and differentiation of data by simplified least squares procedures. *Analytical Chemistry*, v. 36, p. 1627–1639, 1964. Citado na página 34.

SHCHERBAKOV, M. V.; BREBELS, A.; SHCHERBAKOVA, N. L.; TYUKOV, A. P.; JANOVSKY, T. A.; KAMAEV, V. A. A survey of forecast error measures. Citeseer, 2013. Citado na página 36.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. Disponível em: <http://arxiv.org/abs/1409.1556>. Citado na página 21.

SMITH, L. N. No more pesky learning rate guessing games. *CoRR*, abs/1506.01186, 2015. Disponível em: <http://arxiv.org/abs/1506.01186>. Citado na página 28.

SOOFASTAEI, A.; AMINOSSADATI, S. M.; KIZIL, M. S.; KNIGHTS, P. Simulation of payload variance effects on truck bunching to minimise energy consumption and greenhouse gas emissions. *Coal Operators' Conference*, 2015. Citado 2 vezes nas páginas 7 and 14.

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*, v. 15, p. 1929–1958, 2014. Disponível em: <http://jmlr.org/papers/v15/srivastava14a.html>. Citado 2 vezes nas páginas 7 and 25.

SZEGEDY, C.; LIU, W.; JIA, Y.; SERMANET, P.; REED, S. E.; ANGUELOV, D.; ERHAN, D.; VANHOUCKE, V.; RABINOVICH, A. Going deeper with convolutions. In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*. [s.n.], 2015. p. 1–9. Disponível em: <https://doi.org/10.1109/CVPR.2015.7298594>. Citado na página 21.

TIBSHIRANI, R. Regression shrinkage and selection via the lasso: a retrospective. *Journal of the Royal Statistical Society*, Series B (Methodological), p. 267–288, 1996. Citado na página 24.

XU, B.; WANG, N.; CHEN, T.; LI, M. Empirical evaluation of rectified activations in convolutional network. *CoRR*, abs/1505.00853, 2015. Disponível em: <http://arxiv.org/abs/1505.00853>. Citado na página 21.

ZEILER, M. D. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012. Disponível em: <http://arxiv.org/abs/1212.5701>. Citado na página 26.