1 2 9 0

UNIVERSIDADE Đ
COIMBRA

Gonçalo Manuel Oliveira e Sousa

Bridge Cost Forecast

Dissertation in the context of the Master in Data Science and Engineering, advised by Prof. João Correia and Prof. Tiago Martins and presented to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

January of 2023

1 2 9 0

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE Ð
COIMBRA

Department of Informatics Engineering

Gonçalo Manuel Oliveira e Sousa

Bridge Cost Forecast

Dissertation in the context of the Master in Data Science and Engineering, advised by Prof. João Correia and Prof. Tiago Martins and presented to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

January of 2023

1 2 9 0

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE Ð
COIMBRA

DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

Gonçalo Manuel Oliveira e Sousa

Previsão do Custo de Pontes

Dissertação no âmbito do Mestrado em Engenharia e Ciência de Dados, orientada pelo Prof. João Correia e Prof. Tiago Martins e apresentada ao Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

janeiro de 2023

Acknowledgements

First and foremost, I would like to thank my advisers Professor João Correia and Professor Tiago Martins for the availability and full support given throughout this work, for helping me become a better Data Scientist and Engineer with their feedback and constructive opinions.

To my friends from Coimbra and São João da Madeira, who taught me a lot, both at an academic and personal level.

To my family, who gave me constant support and always trusted that I would, eventually, conclude this degree with success.

Last but not least, to my partner, Diana, who always believed in me, and motivated me to keep going even when my motivation wasn't there. Thanks for pushing me to go to the library once in a while, this thesis would not be done had it not been you!

This dissertation was partially funded by the project BEIS (Bridge Engineering Information System), supported by Operational Programme for Competitiveness and Internationalisation (COMPETE 2020), under the PORTUGAL 2020 Partnership Agreement, through the European Regional Development Fund (ERDF)

Abstract

The forecast of a construction project is a task of great importance when it comes to the management of a said project. Bridge construction is a very complex task with many unpredictable factors that can delay deadlines and increase final costs.

This thesis aims to formulate two different predicting models based on Artificial Intelligence (AI) that may help to alleviate and accelerate a process that is mostly manual and tedious. Those two models have different goals. The goal of the first is to estimate the cost of a bridge given multiple features. The second is to forecast the cost of a specific item for a given date.

The first goal was achieved with acceptable results, given the quality of the data, with an R^2 of 0.878, using a Multi-layer Perceptron. An exhaustive search to find its best parameters was performed.

The second goal was supported by a BERD's dataset of a real budget of a bridge between the cities of Porto and Gaia, used in conjunction with the UK's monthly costs of building materials to create a forecasting system. It is not ideal to use UK's cost values instead of Portuguese ones, but, there is a lack of good cost data and this is a merely a proof of concept. Nevertheless, the results were a success. The final version of this system underwent an exhaustive search for the best parameters of a TCNN and had a MAPE of 2.972 for a given item.

These results prove that it is possible, and useful, to use Artificial Intelligence (AI) for bridge cost forecasting. For the most part, these models show a promising collaboration between experts in budgeting and Artificial Intelligence (AI)

Keywords

Machine Learning, Cost estimation, Bill of Materials, Regression Analysis, Artificial Neural Networks

Resumo

A previsão de um projeto de construção é uma tarefa de grande importância quando se trata da gestão de um tal projeto. A construção de pontes é uma tarefa muito complexa, com muitos fatores imprevisíveis que podem atrasar os prazos e aumentar os custos finais.

Esta tese visa formular dois modelos diferentes de previsão baseados na Inteligência Artificial (IA) que podem ajudar a aliviar e acelerar um processo que é, na sua maioria, manual e enfadonho. Estes dois modelos têm objetivos diferentes. O primeiro objetivo é o de estimar o custo de uma ponte, dadas as múltiplas características. O segundo é prever o custo de um item específico para uma determinada data.

O primeiro objetivo foi alcançado com resultados aceitáveis, dada a qualidade dos dados, com um R^2 de 0,878, usando um Multilayer Perceptron. Foi realizada uma pesquisa exaustiva para encontrar os seus melhores parâmetros.

O segundo objetivo foi apoiado por um conjunto de dados do BERD de um orçamento real de uma ponte entre as cidades do Porto e Gaia, utilizado em conjunto com os custos mensais do Reino Unido de materiais de construção para criar um sistema de previsão. Não é ideal utilizar os valores de custos do Reino Unido em vez dos portugueses, mas faltam bons dados de custos e isto é apenas uma prova de conceito. No entanto, os resultados foram um sucesso. A versão final deste sistema passou por uma pesquisa exaustiva dos melhores parâmetros de um TCNN e teve uma MAPE de 2,972 para um determinado item.

Estes resultados provam ser possível, e útil, utilizar a Inteligência Artificial (IA) para a previsão de custos de pontes. Na sua maioria, estes modelos mostram uma colaboração promissora entre especialistas em orçamentação e Inteligência Artificial (IA)

Palavras-Chave

Machine Learning, Estimativa de Custos, Bill of Materials, Análise de Regressão, Redes Neuronais Artificiais

Contents

Acronyms

AI Artificial Intelligence.

ANN Artificial Neural Network.

ARIMA AutoRegressive Integrated Moving Average.

BoM Bill of Materials.

CNN Convolutional Neural Networks.

DT Decision Trees.

EBoM Engineering Bill of Materials.

FBNN Feedback Neural Network.

FFNN Feedforward Neural Network.

FFS Forward Feature Selection.

GBoM Generic Bill of Materials.

GBoMO Generic Bill of Materials and Operations.

GEP Generic End Product.

GRU Gated Recurrent Units.

GSP Generic Subassembly Products.

LR Linear Regressor.

LSTM Long Short-Term Memory.

MAPE Mean Absolute Percentage Error.

MBoM Maintenance Bill of Materials.

ML Machine Learning.

MLP Multi-Layer Perceptron.

MRO Maintenance, Repair and Overhaul.

MSE Mean Square Error.

PGP	Primary Generic Product.
PNN	Probabilistic Neural Network.
PU	Processing Unit.
RF	Random Forests.
RMSE	Root Mean Square Error.
RMSPE	Root Mean Square Percentage Error.
RNN	Recurrent Neural Networks.
SARIMA	Seasonal AutoRegressive Integrated Moving Average.
SI	International System of Units.
TCNN	Temporal Convolutional Neural Networks.
XGBoost	Extreme gradient boosting.

List of Figures

List of Tables

Chapter 1

Introduction

The following thesis report documents the work done from September 2021 to January 2023 in order to conclude the Master's degree in Data Science and Engineering. The thesis was supervised by professor João Correia and professor Tiago Martins, of the University of Coimbra. The thesis was done in collaboration with BERD which is a Portuguese civil engineering company founded in 2006 headquartered in the city of Matosinhos, near Porto, that specializes in bridge and viaduct construction.

The thesis is focused on the development of an analysis and forecast system of bridge construction costs given existing sets of data on previous budgets, processes, materials, quantities and costs. This data is made available by BERD, is tabular and is extracted from the current database of the company.

The chapter that follows introduces the scope and context of the thesis, its motivation, the major objectives and presents a summarized description of the document structure.

1.1 Context

The budgeting task in construction projects, in particular bridge construction, is an extremely important assignment for the success of said project. Currently this task, is, for the most part, performed by human construction cost estimator experts in very subjective manner. Such a subjective study is prone to human error and produces variable answers depending on whom the estimator is [2]. Furthermore, this process is a very tedious and time-consuming one, that require vast amounts of human resources and often leads to losses in efficiency and productivity, as well as extended deadlines that directly and negatively impact the decision-making processes, reflected in large losses of competitiveness and revenue for the company. It has been observed that the cost projection error at the early design stage can be as high as 20-40% of the final cost [27].

Over the last couple of decades, there has been an increase in the application of Artificial Intelligence (AI) in the construction sector, notably in the domain of cost estimation [3]. The implementation of a AI system aims to solve the flaws that are observed in the process of preliminary study and budgeting in the field of bridge engineering. Automating this process based on objective data using AI is important not just for enhancing efficiency and helping experts in the decision-making process, but also for reducing subjective human variables [2].

Methodologies utilizing AI, in particular, artificial neural networks (ANNs) have been shown to perform on pair with human experts [5]. Because they have the ability to learn from past data as well as generalize solutions for future projects, their potential applications in construction forecast is of great potential and importance.

1.2 Motivation

The motivation behind the creation of this system for BERD is, as previously mentioned, the benefic impact in the human resources of the company that an information system like this, if successful, can have. By simulating the best possible preliminary budget within minutes, taking as reference the state of the art, it would allow the employees that would previously be entrusted with the process of cost estimation of a bridge to save large amounts of time, thus freeing them to do other, more relevant tasks, allowing for better resource management. This could prevent unnecessary delays in deadlines and the increased costs that come along with it.

The three main problems that are observed in budgeting and preliminary study in bridge engineering are:

1. The large consumption of internal resources
2. The lack of transparency of the budgeting process
3. the limited range of technical options considered, which is currently confined to the knowledge of the specialists involved

1.3 Objectives

The solution presented in this thesis aims to help solve any of the three issues presented above. In order to solve these, we defined two main objectives:

- Implementation of **cost estimation** regression models for the total cost of building a bridge.
- Implementation of a cost forecasting model for specific items in a time window. In other words, a **time series** forecasting model.

To meet these objectives, the work of this thesis was structured according to the following sub-objectives:

- Implementation of exploratory data analysis of the clean datasets.
- Implementation of regression models to build a bridge cost forecast model on the more relevant data subsets.
- Documentation in the best feasible way to make the system as easy to integrate as possible and to allow for the integration of new data.

By delivering on these two objectives, we will be able to solve two of the three problems stated above, more specifically, the first and third problem. Firstly, if we achieve accurate enough models, and BERD opts in using them in their budgeting process, the consumption of internal resources will certainly decrease, as the cost of certain items would be quickly predicted, this way saving resources. Secondly, these models give BERD another technical option to worth considering. The only problem it does not solve is the second mentioned, mostly because we utilize not easily interpretable techniques, as we will discuss in the following sections.

1.4 Document Structure

The document follows the structure presented below.

- **Chapter 2 -** Provides some general insight about budgeting data structures (Bill of Materials) important in construction projects.
- **Chapter 3 -** The outcome of the research done is here explained. In this chapter, several cost prediction algorithms are presented, both for cost estimation (regression) and time series problems.
- **Chapter 4 -**All steps of the bridge cost estimation process are here described. From detail about the datasets to the discussion of the results.
- **Chapter 5 -** Similar to **Chapter 4** but for the time series forecasting process instead.
- **Chapter 6 -** Presents the conclusions and the perspective for future work.

Chapter 2

Background

In this chapter, I will give a background of Bill of Materials (BoM)s. First, I will explain what a BoM is, and then differentiate between Maintenance Bill of Materials (MBoM)s and Generic Bill of Materials (GBoM)s. It is important to understand what is a BoM because we need to understand how a budget is organized in a computational level, even if, in the context of this thesis, we only work with the individual costs of each item, or in other words, the nodes of a BoM.

2.1 Bill of Materials

A BoM is a diagram that lists all the components and associated quantities that are required to manufacture, assemble or repair one unit of a finished product or an end part of it et al. [8], its main objective is to simplify this process. This diagram provides us with a compact, inventory oriented representation of the requirements associated with an end product.

As can be seen in the work of Kallina et al. [20] a BoM is often represented as a tree structure with relationships between different components and levels of the tree. An example for a BoM as a tree structure can be seen in Figure 2.1. This figure depicts a very simplistic BoM tree, this representation is hierarchical, where the final product is positioned at the highest level, or root of the tree, and all levels underneath represent the materials required for the assembling of the product, these can be raw(or purchased parts) or assembled materials as stated by Cinelli et al.[8]. The raw materials or purchased parts are the leaves of the tree whereas the assembled materials are the nodes, these nodes have BoM subtrees of their own [35].

The order of the nodes (assembled materials), is not relevant. In the work of Romanowski et al. [35], they explain this with a great example, it does not matter whether we say that a car has a body, wheels and transmission or a car has a transmission, body, and wheels. As a result of this a single finished product can have several very different BoM trees, as there is no ground rule to follow, and each engineer has the freedom to develop their own BoM tree based on their individual understanding of how the product going to be manufactured.

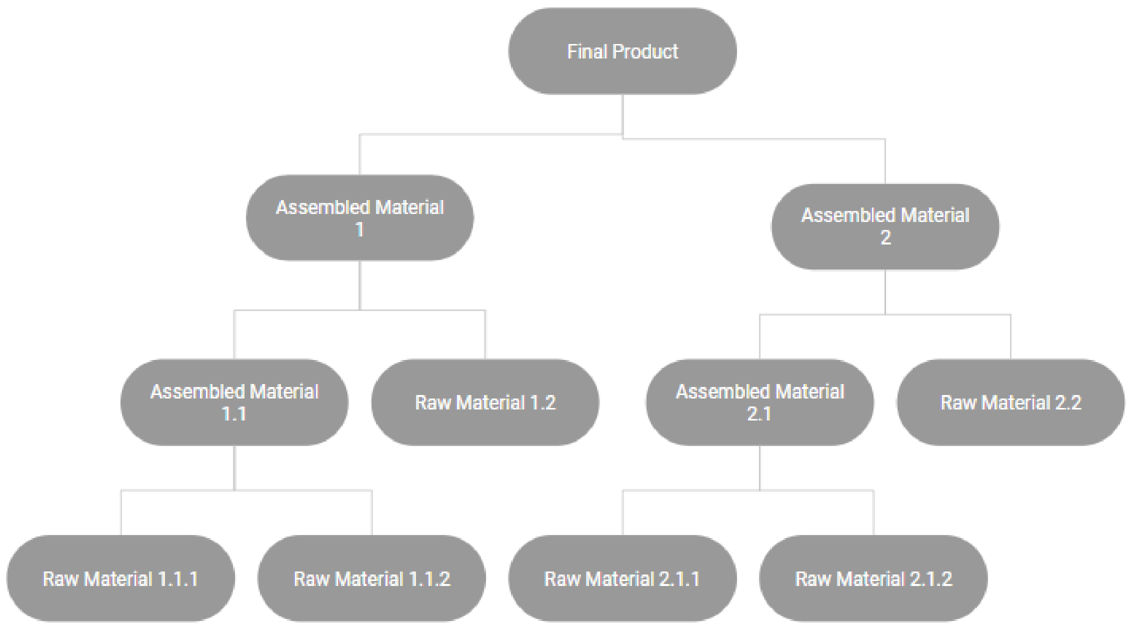

Figure 2.1: Bill of Material tree example

There is not one single possible BoM tree for a complex product, they can differ in three ways, as explained by Romanowski et al.[35]:

1. Structural differences, for example, the number of intermediate materials or materials at different levels.
2. Differences in material labels.
3. Differences in both structure and materials.

Thus, two different but similar BoM trees may have the same materials but have different structures, with some materials appearing at one level in one of the trees and a different level in another tree. In another hand, two different BoMs can have very identical structures but use different materials.

2.1.1 Maintenance Bill of Materials

As mentioned before, BoMs are not only used in the production of a product. For a large-scale project, like the one this thesis is focused on, a Maintenance, Repair and Overhaul (MRO) service is needed to ensure the production process, because of this a normal Engineering Bill of Materials (EBoM) it should be transformed into a MBoM. A mathematical model that ensures the correct transformation from an EBoM to a MBoM can be seen in the work of Liu et al. [23], here accurate mapping is possible using feature recognition rules and methods.

A MBoM is used for the management of product maintenance and is mainly based on the regular EBoM of a product. Where it differs is that a MBoM is more complex than an EBoM, because the relationships of product maintenance processes need to be added to the structural relationships of the parts [40]. While an EBoM is a technical design structure, a MBoM is a management structure. The transformation from an EBoM to a MBoM is necessary because the hierarchy relationship of components in a MBoM must be the reflection of the actual maintenance processes, instead of the manufacturing processes [23].

Figure 2.2shows us an example of a MBoM, as we can see it mainly includes two parts. The first part is the structure needed for the maintenance of the product, this expresses the function relationship of different layers in the tree, which reflects the fault causality among every layer of the product and also shows the maintenance minimum parts for a replacement, these are defined in the MBoM as replaceable components. The second part is the maintenance knowledge of each node in the structure, this can be maintenance data, like data regarding the replaceable component, or diagnosis data, for instance, failure knowledge [17].

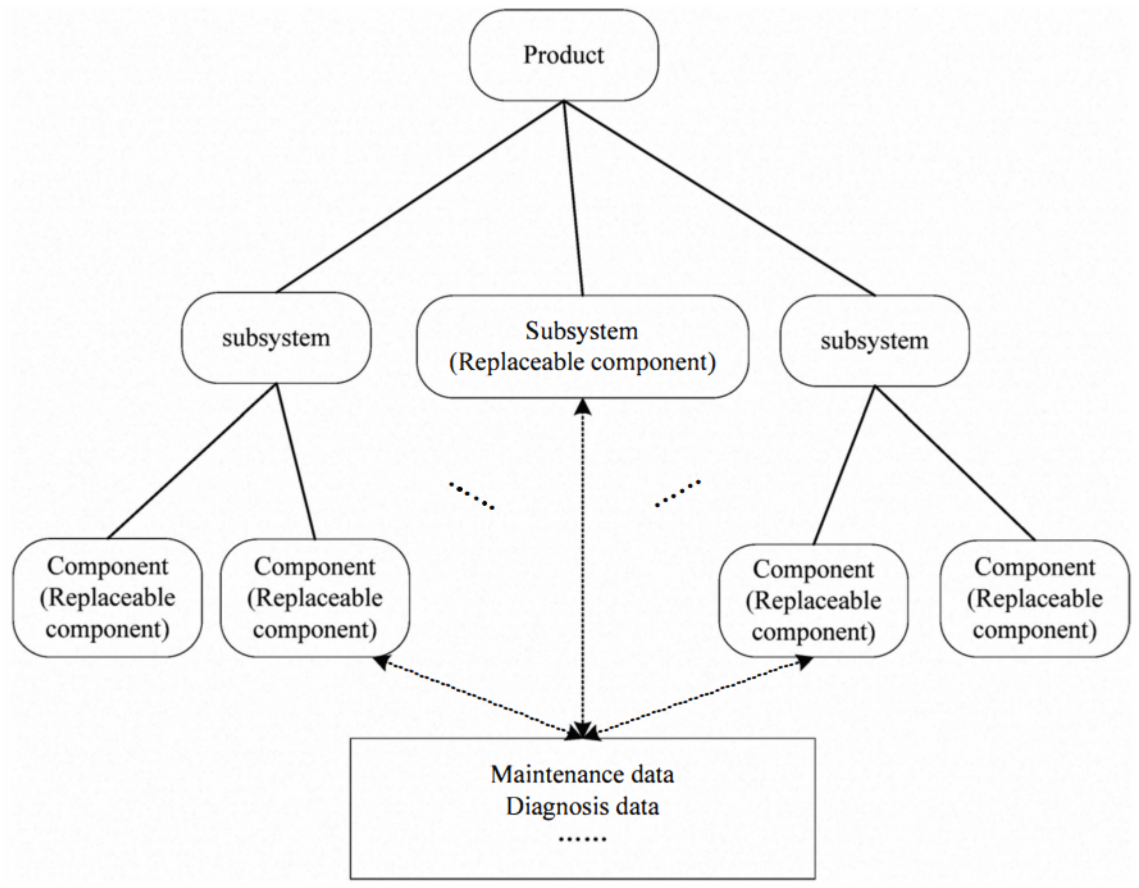
Product
subsystem
Subsystem (Replaceable component)
subsystem
Component (Replaceable component)
Component (Replaceable component)
Component (Replaceable component)
Component (Replaceable component)
Maintenance data
Diagnosis data
......

Figure 2.2: Maintenance Bill of Materials [17]

2.1.2 Generic Bill of Materials

The GBoM has two main aims, to help in the configuration of new variants and to aid the search for identical components, by grouping end products into families and combining the families into a single GBoM entity. In the work of Hegge et al.[14] the GBoM was initially introduced, it represents a single entity product structure that encompasses all design options for a set of similar product variants or product families. This makes a GBoM structure largely transparent, and it enables the avoidance of redundancy when compared to the option of having multiple regular BoMs, one for each product variant. The GBoM allows the user to describe a lot of variants with a limited amount of data.

The GBoM was later extended upon by Jiao et al.[16] by adding rounting data for high variety production, thus creating a Generic Bill of Materials and Operations (GBoMO) structure. This new structure is based on a variant strategy, instead of a generative one like the one discussed by Hegge et al. [14], this means that both the product structure and its operations sequence, for all product variants, are assumed to be common within each product family. For this reason, all changes in a product structure and its operations are regarded as belonging to a different product family. A GBoMO structure permits flexibility in handling the relationships between components and operations, by utilizing parameters it allows the user for a great variety of customized end products.

A procedure-oriented approach was explored by Olsen et al.[30], by combining a few constructs from programming languages like the procedure concept, variables, the input concept and if case statements. These constructs ensure that the GBoM is dynamic. In this approach, each component has a set of attributes that characterize its possible variants. The portrayal of the GBoM as a program capacitates the system to support the user in the product variant determination task [30].

As Romanowski et al.[35], a data mining approach to generate GBoMs was conceived, Kalagnanam et al.[19] and Maiorana et al.[26] undertook similar approaches but not for generic structures. The utilization of techniques like clustering, classification, text mining, association mining and graph based data mining offers a new viable alternative to manual coding such as the one explored in the work done by Olsen et al.[30], which can be very time-consuming. These techniques showed very promising results for BoM structure building.

To better understand the GBoM structure, we first need to understand three main concepts presented to us by Hegge et al.[14]: Primary Generic Product (PGP), Generic Subassembly Products (GSP) and Generic End Product (GEP).

Primary generic products

Primary generic products occupy the lowest level of a GBoM, these entities consist of a set of all variants of a particular primary product (also known as raw material or component). An easy way to identify different variants within a PGP is by utilizing different code values for each primary product. We can later utilize these codes to select the appropriate variant within a PGP by means of parameters. Each combination of parameters represents a different variant within a particular PGP [14].

For example, in Figure 2.3 a BoM of an office chair is displayed, where a possible PGP "Upholstery" could be described by a parameter "colour" with allowed parameter values, 8113, 8114, representing the colours red and blue, respectively.

This will enable us to, using these parameter values, describe an office chair seat variant or even a complete chair office variant in an indirect way, Hegge et al.[14].

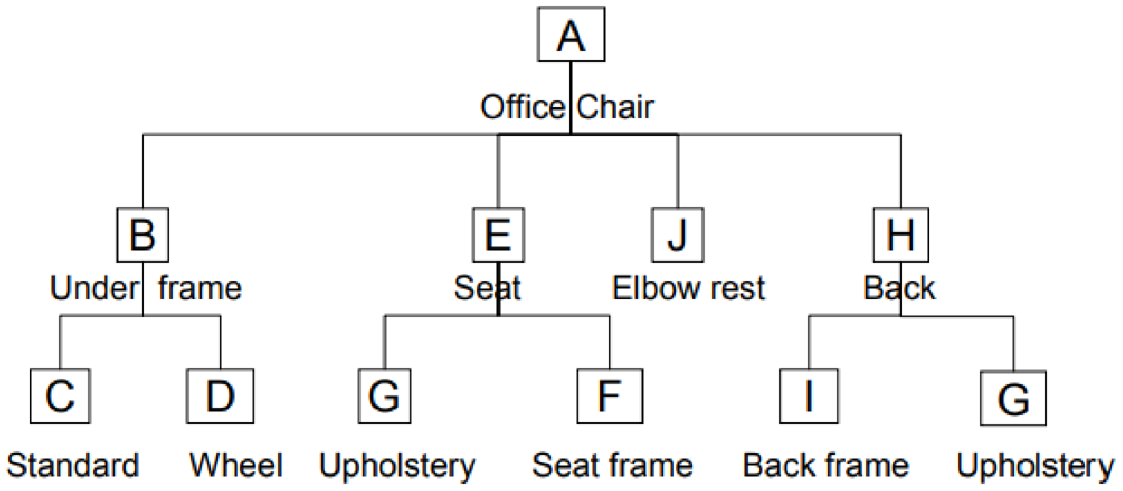

Figure 2.3: Office chair Bill of Materials, Romanowski et al.[35]

Generic Subassembly Products

Similarly to PGP, a GSP is a set of similar subassemblies variants. For example, in Figure 2.3, a GSP could be the office chair seat. These variants are very similar as they share a common structure, allowing a generic item (either a GSP or a PGP) to exist, Jiao et al.[16].

Generic End Product

A GEP corresponds to a set of different, but very similar physical products with similar BoM structures [14], continuing with our previous example of the office chair, the GEP would be the office chair itself, located at the highest level of the BoM, containing GSP variant sets and themselves containing many PGP variant sets.

Chapter 3

State of the Art

The area of study regarding the topic of this thesis is not new, and so, there is a lot of information available. In the following section, I will explain what feature selection is and how it helps us build better models, then I will contextualize the main forecasting algorithms that we are used in this thesis, after that review a few evaluation metrics that are vital in understanding the results of the thesis, and lastly I will review the related work.

3.1 Feature Selection

Feature selection is the process of selecting a subset of relevant features for use in model construction. The goal of feature selection is to identify the most informative features and to improve the accuracy, interpretability, and generalization of the model, states Jovic et al.[18].

There are several different techniques that can be used for feature selection, including:

- **Filter methods:** These are the most commonly used due to their relatively low computational cost and high efficiency. They analyze correlations and redundancies between features and targets. Examples are Correlation Criterion (CC) and Mutual Information (MI).
- **Wrapper methods:** These methods use the performance of a model to select features. They can become computationally expensive when the feature space dimensions large enough. An example is Naïve Bayes.
- **Embedded methods:** These methods use the model training process itself to select features. For example, a Random Forests (RF).

Several approaches exist in order to select the best subsets of features, one way is simply to test all possible combinations and select the best ones, this is called exhaustive search or brute-force search. With the use of this technique, the best possible combination of features will in fact be selected but, it will require training the model once for each combination of features, which is very computationally heavy, particularly if the number of features is large.

The most known and used approaches are much simpler computationally efficient, these are:

- **Forward Feature Selection:** A greedy search algorithm that starts with an empty set of features and then iteratively adds one feature at a time to the set, based on some performance criterion. The goal is to find the smallest set of features that still provides good performance. The process continues until a stopping criterion is met, such as a maximum number of features, or a threshold performance criterion.
- **Backward Elimination:** a process that starts with all features and iteratively removes one feature at a time, based on some performance criterion, until a stopping criterion is met. The goal is to find the smallest set of features that still provides good performance.

Both are good enough approaches, however, their greedy nature means that they may not always find the optimal feature set. They also don't consider the interactions of features, and the performance of a subset of features may depend on the other features in the subset, states Jovic et al.[18].

3.2 Forecast Algorithms

Cost estimation of construction projects can be very uncertain and difficult to accurately be executed, explains Alshmrani et al.[5]. Because of this, many Machine Learning (ML) models are currently available in practice to help humans get more precise budget estimations. Techniques such as regression analysis and Artificial Neural Network (ANN) have been explored for this task in particular. In the further sections, these will be expanded upon and a literature review relevant to the thesis will be presented.

3.2.1 Cost Estimation

Linear Regression

A regression model, is, very simplified, a statistical method for investigating the relationship between one variable and several other variables, and it is one of the oldest prediction models, states Yan et al.[47]. Three types of regression models exist: Linear regression, multiple linear regression and nonlinear regression.

Linear regression is the simplest out of the three. This regression algorithm tries to model the linear relationship between two variables, a dependent one, typically denoted as y and an independent one, depicted as x. This regression model is usually formulated as follows,

$$y = \beta_0 + \beta_1 x + \varepsilon, \tag{3.1}$$

where y and x are, as previously mentioned, the dependent and independent variables, respectively, β_0 is the y-intercept, β_1 is the gradient and ε is a random error. The error is commonly assumed to be normally distributed, states Yan et al.[47].

Multiple linear regression is very similar to linear regression, however, instead of having only independent variable x, it has multiple ones [47]. The dependent variable y is still, nonetheless, only one. Its regular form is as follows,

$$y = \beta_0 + \beta_1 x + ... + \beta_n x_n + \varepsilon, \tag{3.2}$$

where y remains the same dependent variable, $\beta_0, \beta_1, \beta_2, ..., \beta_n$ are the coefficients and $x_0, x_1, x_2, ..., x_n$ are the independent variables of the multiple linear regression model. Similarly to the linear regression model ε is the error, and it is also normally distributed [47].

The final regression model is nonlinear regression. This model accepts that the relationship between dependent and independent variables is a nonlinear one. One example, of many, nonlinear regression models can be written as

$$y = \frac{\alpha}{1 + e^{\beta x}} + \varepsilon, \tag{3.3}$$

where y is the dependent variable, x is the independent variable, α and β are parameters of the model, and ε is the error. These models are usually more complex than both models presented above as they require an estimation of multiple parameters, model selection, variable selection, outlier treatment, among other additional operations, explains Yan et al.[47].

To estimate linear and multiple linear regression models parameters β, there are several methods to choose from, for example, the least-squares approach, R^2 and the maximum likelihood approach. The least-squares method is the most popular method for estimating β [41], it minimizes the sum of the squares of the distance from the observed value and the fitted value provided by the regression model, this distance is commonly refereed as residual. The maximum likelihood method is a method that determines the parameter values such that they maximize the likelihood that the process described by the model produced the data that were actually observed [41]. R^2, however, measures the proportion of the variance for a dependent variable that's explained by an independent one, it explains to what extent the variance of one variable influences the variance of a second variable, for example, if the R^2 of a model is 0.5, then approximately half of the observed variation can be explained by the model's inputs [28].

Random Forest

In order to understand how RF work, we first need to understand how Decision Trees (DT) work, since RFs is an ensemble model that that uses a collection of DT.

A DT is a tree-like model made up of internal nodes (also called decision nodes) and leaf nodes (also called terminal nodes). The internal nodes represent a test on an input feature, and each branch represents the outcome of the test. The leaf nodes represent the predicted value of the target variable. The nodes are decision points that have conditions, these conditions are selected by the feature that best splits the data into subsets, exolains Westreich et al.[46], the nodes then expand the tree into more nodes and the process is repeated until a stopping criterion is met, often the maximum tree depth. The final result of the algorithm is a tree of decision nodes and leaf nodes.

As previously stated, RF is a DT-based ensemble, with each tree depending on a collection of random variables, states Cutler et al.[9]. When a RF receives an input, it builds a set number of DTs and averages the results given by them. To avoid correlation of different trees, RF increases tree diversity by expanding trees from different training data subsets [34]. This process is called bagging.

Extreme gradient boosting (XGBoost)

Extreme gradient boosting (XGBoost) is a scalable DT ensemble method similar to RF that, is designed for performance and speed. Unlike RFs that use the bagging ensemble technique, XGBoost uses, as the name suggests, a boosting ensemble technique instead. This technique, instead of creating several parallels DTs and averaging the result of each to obtain a result like the RF, works iteratively. This means that the model (a DT) will initially predict something and use the prediction errors for the following DT, thus it will give more weightage to the data points in which it made a wrong prediction in the next iteration, or in other words, it particularly targets samples with higher uncertainty, explains Abedi et al.[1]. This process continues until all DTs are built, and a result is forecasted. Unlike a RF that uses subsets of data for each DT, XGBoost utilizes the whole dataset for each.

Artificial Neural Networks

In the former subsection, we explored how regression algorithms can help humans in construction cost estimation tasks. Here we take a look at a more recent approach: ANNs. Neural networks can not only fairly accurately estimate construction projects costs by considering physical attributes of the project, but can also model complex and frequently misconceived interrelationships that might exist between variables [12]. In the work of Alex et al.[3] explain that ANN models are very useful for solving complex cost estimation problems, this is very much the case because ANN models are capable of:

1. Modelling the interdependencies that invariably arise between input data, unlike regression models which are an "a priori" model.

2. easily handling nonlinear relationships among parameters.

3. handling incomplete data sets, which are the norm when it comes to the practical reality of cost estimation data.

The classic ANN model is inspired by the structure and performance of the biological neural network. Although biological neural networks are much more complex, and we still do not fully understand them, ANNs show some of the biological network's characteristics, explains Yegnanarayana et al.[48].

A ANN consists of several interconnected Processing Units (PU), also known as neurons, these units incorporate two main components, a summing part and an output part. The unit's summing part receives multiple weighted input variables and computes a weighted sum, commonly referred to as the activation value. The output part later uses this activation value to calculate a signal based on a selected activation function [48]. A visual representation of an example Processing Unit (PU) can be seen in Figure 3.1, here an activation value x is calculated using a weighted sum of M input values a_i, the output signal s later uses the activation value x in the activation function $f(x)$. These activation functions are usually nonlinear, with three of the most commonly used being binary, sigmoid and ReLU, states Karlik et al.[21].

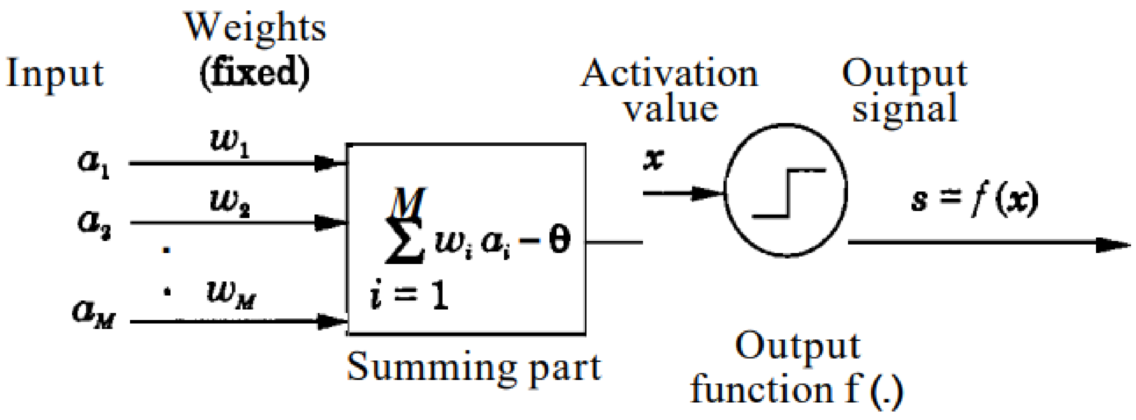

Figure 3.1: McCulloch-Pitts Model of a PU [48]

To complete a pattern recognition task, a ANN connects multiple of these PUs according to a specified topology. Dependent on the topology of the ANN the inputs of a PU may be the outputs from another PU. The strength of a connection between two PU affects the output of one PU, this strength is represented in the weight value associated with the connecting link between the two PUs, states Yegnanarayana et al.[48]. The weights of all connections are normalized and stored in a structure named weight vector. These weights are constantly adjusted in the learning process so that the given patterns may be stored in the ANN [48]. The weight of a particular PU in a moment $t + 1$ is determined by the equation below:

$$w(t+1) = w(t) + \Delta w(t), \tag{3.4}$$

where $\Delta w(t)$ corresponds to the change to be applied to the weight [48].

According to multiple different learning laws, such as Hebb's Law, Delta Law, and Widrow-Hoff LMS Learning Law, there are many possible approaches to execute this change to weight values of PU connections.

Ordinarily, ANNs are divided into layers of PUs. A layer's PUs is comparable in that they all have the same activation function. A Layer then can either connect with other layers, known as interlayer connection or, in some specific topologies, connect with itself, an intralayer connection.

There are two primary families of ANNs, Feedforward Neural Network (FFNN) and Feedback Neural Network (FBNN) wherethe FFNN is the most popular type of ANN. FFNNs are a type of ANN that only allows signals to travel one way: from input to output. This type of ANN is ordered into layers where the first layer is the input layer, the last layer is the output layer and every other layer in between these two are hidden layers. Figure 3.2 shows us a FFNN with one hidden layer.

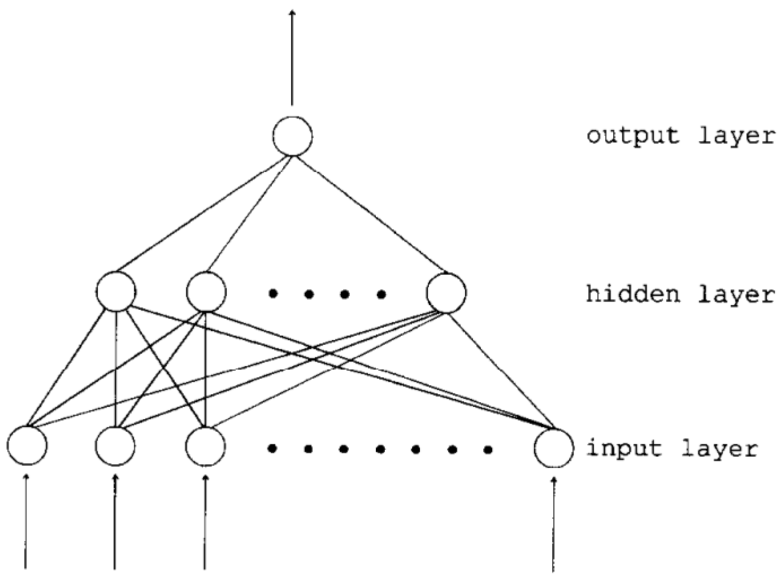

Figure 3.2: Feedforward neural network comprised of three layers [43]

In order for a FFNN to learn, a back-propagation training algorithm is used. In this algorithm, after each forward pass through the network, a backward pass is performed in order to adjust the weights of the ANN. In this step the back-propagation algorithm evaluates the predicted output given by the output layer when compared again the expected output, this happens through a cost function. The cost function can vary a lot, it can range from a simple mean squared error to a more complex cross-entropy function. Based on this evaluation, the model can adjust its weight parameters to get closer to the expected output. The ANN, using back-propagation, aims to minimize the cost function. This minimization is typically done employing a gradient descent [36].

A FFNN with multiple layers is called a Multi-Layer Perceptron (MLP) and they are the most widely studied and used ANN in practice, as explained by Yegnanarayana et al.[48].

3.2.2 Time Series Forecast

Seasonal Autoregressive Integrated Moving Average

Seasonal AutoRegressive Integrated Moving Average (SARIMA) is a statistical model that is used to analyze and forecast timeseries data. It combines three types of models:

1. Autoregressive (AR): a type of model that uses lagged values of the timeseries to predict future values. For example, if the value of a timeseries at time t is correlated with the value at time $t-1$, then an autoregressive model would be appropriate.

2. Integrated (I): a model that accounts for non-stationarity in a timeseries by differencing the data. Non-stationarity refers to the fact that the statistical properties of a timeseries change over time. For example, the mean of a non-stationary timeseries may increase over time, or the variance may change. By taking the difference of the timeseries at different points in time, a stationary timeseries is obtained, which makes it easier to model.

3. Moving Average (MA): a model that uses the residual errors from a prediction made using an autoregressive model to predict the next value in the series.

The combination of these three models gives us an AutoRegressive Integrated Moving Average (ARIMA) model. If the system has a seasonality component, it is necessary to use a SARIMA. This SARIMA model also includes a seasonal component, which takes into account the fact that certain patterns in the data may repeat at regular intervals (e.g. monthly, quarterly, etc.). The seasonal component is characterized by the period of the seasonality (e.g. 12 for monthly data), and the number of lags (e.g. 4 lags for quarterly data).

Recurrent Neural Networks

Recurrent Neural Networks (RNN) are a type of FFNN that allow previous outputs to be used as inputs while generating the output. This makes RNNs suitable for tasks that involve sequential data, such as language modeling, translation, and timeseries forecasting, which is the problem that we have at hand.

A RNN takes in a sequence of input data and produces a sequence of output data. At each step in the sequence, the RNN processes the current input and the previous hidden state to produce a new hidden state and output. The hidden state contains information about the past inputs processed by the RNN, allowing it to capture dependencies between time steps. It is a very deep FFNN that has a layer for each time step and its weights are shared across time, explains Sutskever et al.[42].

Although RNNs are a great tool for timeseries forecasting, because of their non-linear iterative nature, they are intrinsically difficult to train, particularly on tasks with long-range temporal dependencies [42]. This is because the hidden state must be propagated through the entire sequence, which can make it difficult for the gradients to flow back through the network. Additionally, RNNs may struggle to capture long-term dependencies in the data, especially if the data has a complex structure, states Zaremba et al.[49].

In the Figure 3.3 we can observe the traditional structure of this model, where each time step t has an activation $a^{<t>}$ and an output $y^{<t>}$.

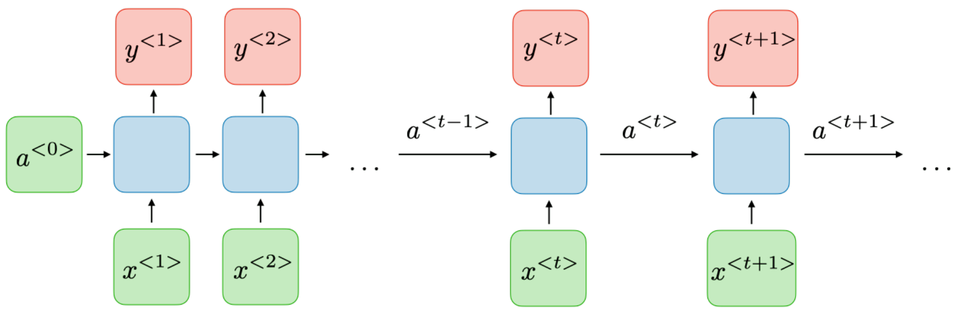

Figure 3.3: Architecture of a traditional RNN

Gated Recurrent Unit

Gated Recurrent Units (GRU) are a type of RNN. Like other RNNs, GRUs process a sequence element-by-element, maintaining an internal state that encodes information about the elements seen so far. However, GRUs differ from other RNNs in the way they update their internal state, which allows them to better capture long-term dependencies. GRU aims to solve the vanishing gradient problem, which comes with a standard RNN, as said by Cho et al.[6].

As one can see from Figure 3.4, GRU have two gates: a reset gate and an update gate. The reset gate determines what information to discard from the previous state, while the update gate determines what information to store. These gates are controlled by the current input and the previous hidden state, and are computed using sigmoid activation functions, states Chung et al.[7].

The output of the GRU is a weighted combination of the previous hidden state and the current input, where the weights are determined by the update gate. This allows the GRU to selectively choose what information to keep from the previous state and what information to incorporate from the current input, affirms Chung et al.[7].

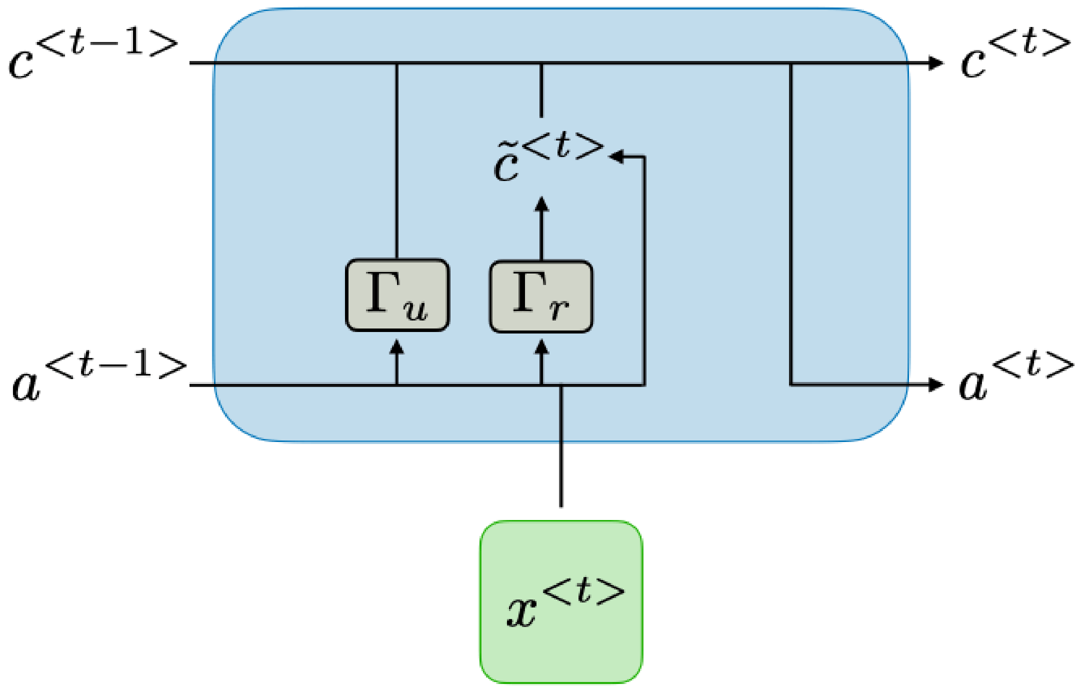

Figure 3.4: Architecture of a Gated Recurrent Unit

Long Short-Term Memory

Long Short-Term Memory (LSTM) is, like GRU, a type of RNN. One of the key features of LSTM is its ability to remember information for long periods of time, which makes it particularly useful for tasks that require long-term memory [7].

Similar to GRUs, LSTMs have the update and reset gate. Where they differ is that LSTM have two further gates: the forget gate, that determines which information should be discarded from the cell state and the output gate, which controls the flow of information out of the LSTM unit. By controlling the flow of information through these gates, LSTMs can selectively remember or forget information, and they can use this information to make predictions about future events, asserts Cho et al.[6]. These gates interactions can be seen in Figure 3.5

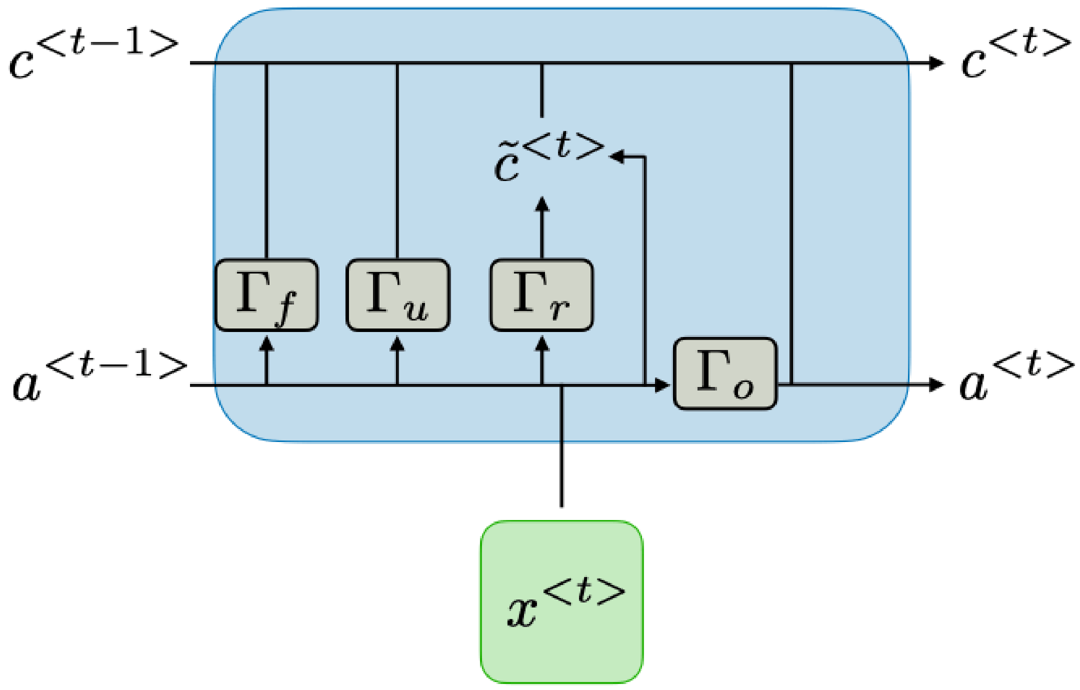

Figure 3.5: Architecture of a Long Short-Term Memory unit

Temporal Convolutional Neural Networks

Temporal Convolutional Neural Networks (TCNN) are a type of deep FFNN similar to Convolutional Neural Networks (CNN) that is designed to process sequential data, such as timeseries, instead of image data, in short, they are designed to operate on sequences of data rather than grids of pixels.

Because, TCNNs are a derivation of CNNs. In order to understand them, it is important to understand CNNs. As stated before, CNNs are primarily used for patterns recognition within image data, hence, its input will consist of images. CNNs are comprised of three main types of layers [31], as follows:

1. **Convolutional layers:** Applies a set of filters to the input data in order to extract features. The filters are learned during the training process and are designed to detect specific patterns in the input data, such as edges, shapes, and textures. The output of a convolutional layer is called a feature map. Afterward, it is usually applied an activation function (e.g., sigmoid, hyperbolic tangent, rectified linear unit - ReLU) The layers parameters focus around the use of learnable kernels that have small spatial dimension, but are spread over the entire depth of the input, reveals O'Shea et al.[31].

2. **Pooling layers:** Used to reduce the spatial dimensions of the feature maps. It does this by applying a pooling operation, such as max pooling or average pooling, to each subregion of the feature maps [31]. This helps to reduce the amount of computation required by the network by reducing the number of parameters that need to be learned.

3. **Fully-connected layers:** Standard neural network layer that connects all the input neurons to all the output neurons [31]. This allows the network to learn a rich set of features from the previous layers and make a prediction or classification of the input.

An example architecture of these layers can be seen in the in Figure 3.6.

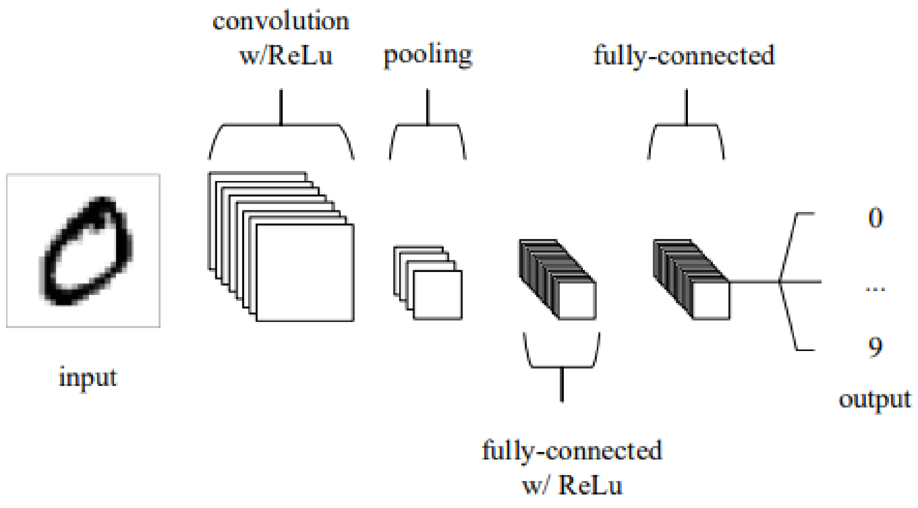

Figure 3.6: Architecture of a simple five layered Convolutional Neural Network [31]

TCNNs, on the other hand, are designed to process sequential data, such as time-series. They also consist of a series of convolutional layers, but they are applied to sequences of 1D data rather than grids of pixels. CNNs are able to capture context by stacking multiple convolutional layers, each of which extracts features from a larger window of the input data. In contrast, TCNNs use of dilated convolution, which allows them to capture context by effectively skipping over time steps, allowing it to capture longer-term dependencies in the data, affirms Pelletier et al.[33]. Dilated convolution is similar to traditional convolution from CNNs but with a few key differences. In this technique, the filter is also applied to a small window of the input data, but the spacing between the input values is increased, states Pandey et al.[32]. This spacing is controlled by a parameter called the dilation rate. How this process works can be observed in Figure 3.7.

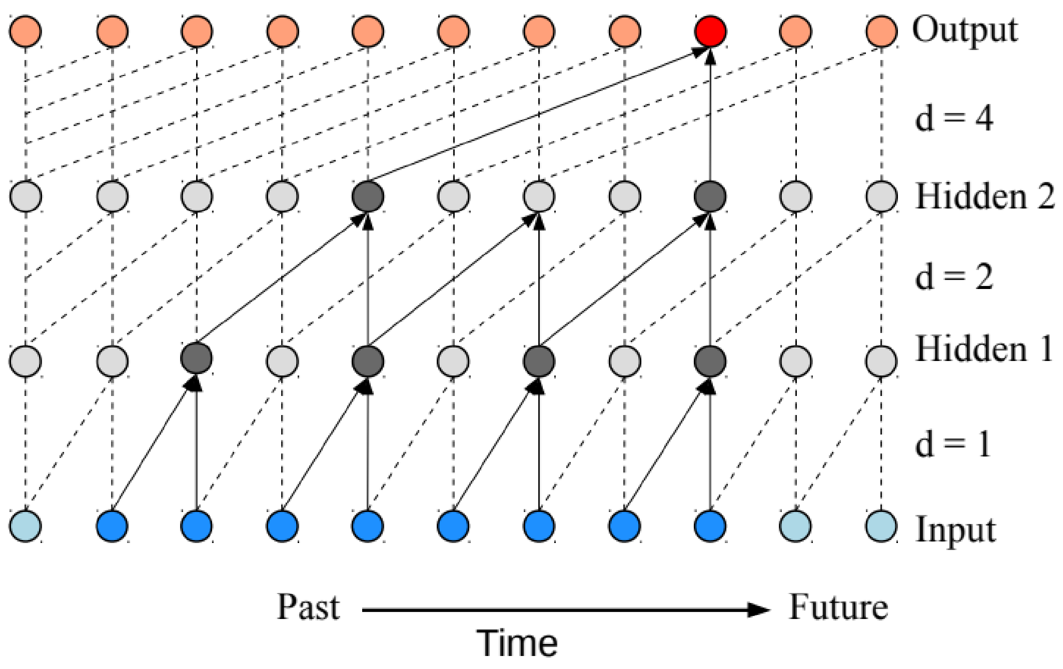

Figure 3.7: An example of dilated causal convolution [32]

3.3 Evaluation Metrics

Because both our objectives in this thesis are regressions, we can use the same evaluation metrics for both. The evaluation metrics will allow us to better understand the overall performance of our models, specially when we use multiple metrics at the same time. The metrics that are discussed in this section are:

- Mean Absolute Percentage Error (MAPE)
- Mean Square Error (MSE)
- Root Mean Square Error (RMSE)
- Root Mean Square Percentage Error (RMSPE)
- R^2

3.3.1 Mean Absolute Percentage Error

Mean Absolute Percentage Error (MAPE) measures the average magnitude of the error in percentage terms, calculated as the average absolute percentage error over the test, train or validation set.

Its formula is:

$$MAPE = \frac{1}{n} \cdot \sum_{t=1}^{n} \left\| \frac{A_t - F_t}{A_t} \right\| \tag{3.5}$$

where:

- n is the number of samples in the test set
- A_i is the actual value of the target variable for sample i
- F_i is the forecasted value of the target variable for sample i

In order to interpret the MAPE value, a lower value would indicate a better fit of the model to the data, as explains Naser et al.[29]. A value of 0 would mean that the model is a perfect fit, with no error at all. In practice, it is very rare to achieve a MAPE of 0.

Even though MAPE is a great evaluation metric, it has its drawbacks, for one, it can only be used when the quantity to predict is known to remain well above zero, as it cannot be used if there are actual zero or negative values [29].

3.3.2 Mean Square Error

Mean Square Error (MSE) measures the average squared difference between the predicted values and the true values.

To calculate MSE, it is necessary to calculate the difference between the predicted value and the true value for each sample in your dataset. These differences are called the residuals. Each of these residuals is squared, and the average resulting value is used, Vishwakarma et al.[44].

Mathematically, the MSE is calculated as:

$$MSE = \frac{1}{n} \sum_{t=1}^{n} (A_i - F_i)^2 \tag{3.6}$$

where n, A_i and F_i mean the same as their counterparts in MAPE.

MSE is commonly used as it is easy to calculate and interpret, and it penalizes large errors more heavily than small ones. Because of this, MSE is sensitive to outliers, since larger errors can significantly increase the overall MSE. As a result, it is often used in conjunction with other metrics that are less sensitive to outliers, explains Vishwakarma et al.[44]. It is also highly dependent on which fraction of data is used, with very different depending on the data, because of this it has a low reliability, states Shcherbakov et al.[39].

3.3.3 Root Mean Square Error

Root Mean Square Error (RMSE) is a measure of the difference between the forecasted values and the actual values in a dataset. It is one of the most used metrics in regression models, explains Vishwakarma et al.[44].

RMSE is very similar to the already described MSE, the only difference is that with RMSE the squared root of MSE is used. Because of this, its formula is:

$$RMSE = \sqrt{MSE} \tag{3.7}$$

or

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(A_i - F_i)^2} \tag{3.8}$$

The popularity of RMSE is mainly due to its easiness to understand and because it is in the same units as the original data. A lower RMSE indicates a better fit of the model to the data [39].

Due to its similarities with MSE it is impossible not to compare them. RMSE is generally better because it uses the same units as the original data, this makes it easier to compare different models or to compare a model to a baseline model. But RMSE doesn't come with its drawbacks, since it can be biased if the data is not normally distributed. In this case, MSE may be a more appropriate metric [44]. Similarly to MSE, RMSE is also sensitive to outliers [39].

3.3.4 Root Mean Square Percentage Error

Root Mean Square Percentage Error (RMSPE) is the percentage error equivalent of RMSE, it is calculated as the root mean square (RMS) of the percentage error between the predicted values and the true values. The percentage error is calculated as the absolute error divided by the true value, and is expressed as a percentage [44].

Hence, RMSPE formula is:

$$RMSPE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(\frac{A_i - F_i}{A_i})^2} \tag{3.9}$$

The RMSPE is a useful metric because it takes into account the relative size of the error. For example, if the true value is 100 and the predicted value is 105, the absolute error is 5. However, if the true value is 1000 and the predicted value is 1005, the absolute error is still 5, but the RMSPE would be lower because the relative size of the error is smaller [39]. Because of the stated, RMSPE is scale independent and Can be used to compare predictions from different datasets, as said by Naser et al.[29].

3.3.5 R^2

R^2 is the coefficient of determination or the square of the correlation coefficient R. Unlike other metrics described before, the goal is to maximize the R^2, hence maximizing R^2 towards the upper limit of 1.0 is comparable to minimizing MSE or RMSE, states Vishwakarma et al.[44]. R^2 values close to 1.0 indicates a strong correlation, therefore values below 0 indicate that there is a negative correlation or a relationship between the forecasted and the actual values is such that as the value of one increases, the other decreases. In practice, R^2 will be negative whenever the model's predictions are worse than a constant function that always predicts the mean of the data [29].

The formula for R^2 is:

$$R^2 = 1 - \frac{\sum_{t=1}^{n}(F_i - A_i)^2}{\sum_{t=1}^{n}(A_i - A_{mean})^2} \tag{3.10}$$

It is important to note that the R2 is sensitive to the number of independent variables in the model, and can be artificially inflated when adding variables that are not relevant to the model [4].

3.4 Related Work

Since our two objectives are not relatively new in the Artificial Intelligence (AI) scheme, it is important that we highlight and discuss the previous related work that are considered state of the art. By doing this, we will have a broader understanding of how the models perform in practice so that we can use this knowledge to our advantage. The following subsections show the related work for both our objectives.

3.4.1 Cost Estimation

Construction cost prediction of buildings based on 286 sets of data using multiple linear regression models was explored in the work by Lowe et al.[25]. This paper takes an unorthodox approach regarding cost estimation by rejecting the raw cost as a dependent variable, instead, the models were developed for cost/m^2, *log* of cost and *log* of the cost/m^2. The rejection of raw cost as a dependent variable is mainly due to the results when compared with the other three approaches, having very low effectiveness ($R^2 = 0.941$) in predicting cost/m^2 and a very high average error in the predicted cost of a project (65.3%). Because of these reasons, Lowe et al.[25] demonstrates that the raw cost as the dependent approach, for this problem is hand, will not be effective for the project prediction. After using forward selection to determine the best combination of variables, the best model found was the cost/m^2 model, both in predicting the cost and cost/m^2 as is shown by their R^2 scores, 0.944 and 0.668 respectively. The paper ends by comparing these results with an ANN approach. The best ANN is one that utilizes all variables available and a voting system using 100 different networks. This approach had a R^2 value of 0.789 for the predicted cost, slightly surpassing the best linear regression model, concluding that for this type of problem in particular an ANN approach tends to be superior. Even though regression models can be outperformed by more advanced techniques such as ANN, the models produced compare favorably with traditional, non-AI oriented methods.

In the work by Li et al.[22] multiple linear regression was utilized to predict the building cost of office buildings in Hong Kong. The data was collected from 14 office buildings, where 7 of them were selected randomly to construct the cost model and the remaining 7 were used for cross-validation. The model showed fairly good results, with R^2 = 0.96 score in construction cost estimation after feature selection. Much like the other studies regarding cost estimation, it concludes the great importance of multiple linear regression approaches when it comes to supplementing the judgmental forecast of cost advisors in the early stages of construction projects.

A great example for the visualization of the regression model development process is presented in Figure 3.8 from the work of Alshamrani et al.[5], which explores construction cost prediction models for university buildings. This process comprises three steps. The first step is preliminary data diagnostics, where procedures such as parameter selection are carried out, in this case study in particular the input parameters considered were building area, floor height, number of floors and structure and envelope types. Secondly is the model development process, in which the regression model is built, preliminary tests such as R^2 are undertaken and the residual values are analyzed. Lastly, the process of model validation is performed, in this step Alshamrani et al.[5] compares the actual observation with the predicted values in order to validate each developed model. The model validation process reveals fine results with an accuracy of 94.3%.

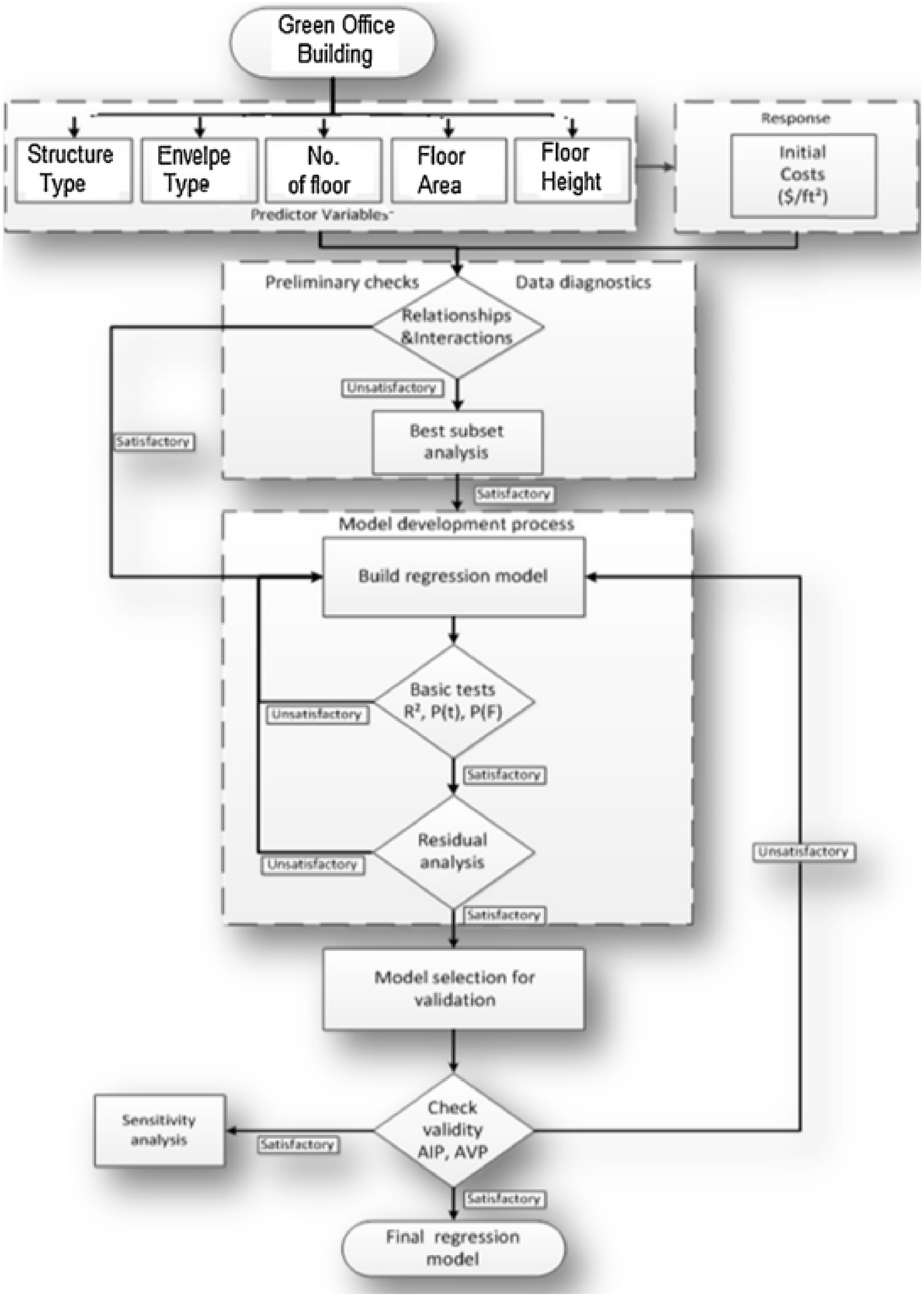

Figure 3.8: Regression model development process [5]

In [15] a RF model and a XGBoost model were put side by side in order to compare their forecasting abilities to predict the blood pressure of patients during hemodialysis. It utilized 7180 blood pressure records for the training dataset and 2065 for the test dataset, a split of roughly 80% and 20% respectively. This study found that RF (R^2=0.49, RMSE=16.24) had, surprisingly, a slightly better predictive performance than XGBoost (R^2=0.41, RMSE=17.65) on the testing dataset, even though XGBoost had better performance on the training dataset (XGBoost (R2=1.00, RMSE=1.83) versus RF(R^2=0.95, RMSE=6.64)). These results are likely an outcome of overfitting.

The study conducted by Zhang et al.[51], that tried to assess the basal heave stability for braced excavations in anisotropic clay, obtained the different results. XGBoost (R^2=0.995, RMSE=0.024, MAPE=0.009), as expected, performed better in all metrics than the RF (R^2=0.988, RMSE=0.040, MAPE=0.013) in the test dataset. The study ends concluding that both XGBoost and RF models proved to be promising for the prediction problem at hand, but the lack of high-quality datasets for this problem present a clear limitation.

As mentioned before, ANNs can be very useful in helping humans in construction cost estimates, this way reducing the enormous amount of internal resources. In the work done by Sawhney et al.[38] an approach based on Probabilistic Neural Network (PNN)s was developed in order to develop a system (IntelliCranes) to help specialists in the task of crane type selection for construction operations. The type of crane used in the training data was based on previous work where the cranes proved to be the best fit for the job. PNNs was the approach chosen mainly because PNNs can learn fairly well even on small datasets, and, in this case, there was insufficient data to train a classic neural network. Overall, according to the authors, the system performed reasonably well. It showed promising results given more available data, with an accuracy of 92.1% and most classification errors found to be in the two types of cranes with the least amount of available data. It concludes by reiterating the importance of ANNs approaches in construction projects.

A methodology for comparing the influence of feature selection in cost estimation of construction projects is presented by Elhag et al.[13]. The data used comprised information about the construction of 30 school projects. In this approach, two ANNs models were developed, one utilized 13 of 14 available features (model I) while the other only used 4 of them (model II). The procedure behind the feature selection is not revealed in the paper. The number of PUs in the single hidden layer also differs in between models, while model I has 13 PUs, model II contains only 3, these were determined by trial and error, selecting the number of PUs that gave the best result for each model. For metrics, the inverse of MAPE is employed. In the testing phase, the ANN model I and II achieved average inverse MAPE of 79.3% and 82.2% respectively. From these results, they concluded that fewer, more relevant features is better for the generalization capabilities of a ANN model.

In done by Adeli et al.[2] a ANN is elaborated in an effort to estimate the cost of reinforced-concrete pavements. To overcome the problem of overfitting and to improve the cost estimation outside the available data, [2] added a regularization term to the standard error term as means to smoothen the approximation function. The absolute error in the validation subset was found to be $7.22/m^3$ of pavement, with the average unit cost of the concrete pavement being around $39.2/m^3$. The authors of the paper attribute this error to the lack of sufficient data examples.

Lastly, Alex et al.[3] created a ANN model for water and sewer installation cost estimation based on data from a 14-year period, accounting for over 800 data entries. As one can see, Figure 3.9 illustrates a flowchart of the methodology used in the development of the model. The approach considered as input parameters project estimation data as well as geographical location and temperature forecast data (obtained from a temperature forecast model). The best model demonstrated a prediction accuracy (R^2) of roughly 80%. This value was derived through the testing of several network designs.

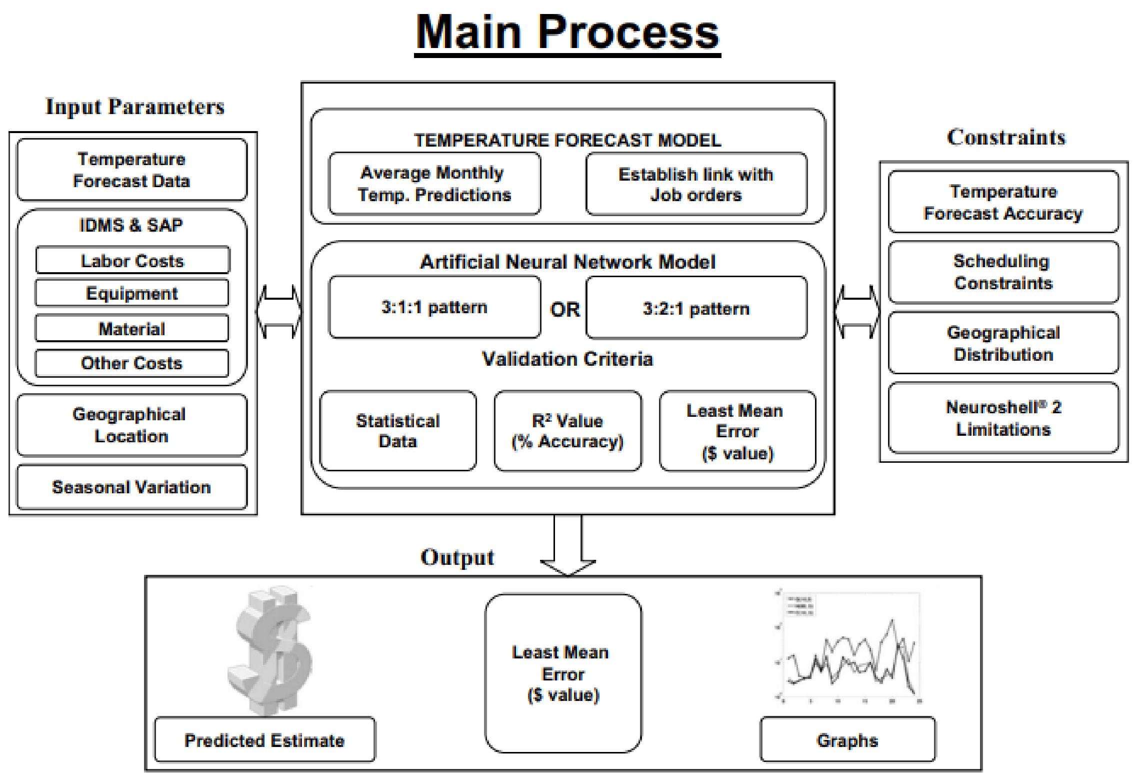
Main Process
Input Parameters
Temperature Forecast Data
IDMS & SAP
Labor Costs
Equipment
Material
Other Costs
Geographical Location
Seasonal Variation
TEMPERATURE FORECAST MODEL
Average Monthly Temp. Predictions
Establish link with Job orders
Artificial Neural Network Model
3:1:1 pattern
OR
3:2:1 pattern
Validation Criteria
Statistical Data
R² Value (% Accuracy)
Least Mean Error ($ value)
Constraints
Temperature Forecast Accuracy
Scheduling Constraints
Geographical Distribution
Neuroshell® 2 Limitations
Output
Predicted Estimate
Least Mean Error ($ value)
Graphs

Figure 3.9: Research methodology flowchart [3]

One common problem found in almost every cost estimation problem for construction projects is the lack of sufficient data entries to train the model and the unpredictable factors that are bound to construction projects and are impossible to avoid.

3.4.2 Time Series Forecast

In the study conducted by Sahoo et al.[37] the authors explored the suitability of RNNs and its variant LSTMs for low-flow hydrologic timeseries forecasting, caused by rainfall and water level in a river as well as physical properties. The results of this study let the authors conclude that LSTM model variant (R^2=0.943, RMSE=0.487) outperformed the vanilla RNN model (R^2=0.935, RMSE=0.516). They also conclude that LSTM can be used as a reliable technique for low-flow hydrologic forecasting.

In the work done by Zhang et al.[50], vanilla-RNNs, LSTMs and GRUs models were compared for short-term load electricity forecasting. The results show that the LSTM (MAPE=1.09) and GRU (MAPE=1.06) methods are slightly better than the vanilla-RNN (MAPE=1.66) according to the average results, with very similar results in the test dataset. The authors considered that the model is very effective.

The study done by Wan et al.[45] compared the effectiveness of LSTMs and TC-NNs for weather forecasting using timeseries data across a multitude of weather parameters such as rain, temperature, humidity, wind speed etc. The authors did not disclose the actual values of the metrics (MSE) for either models. However, they disclosed the Figure 3.10. As Figure 3.10 illustrates, the TCNN provided better results in 6 out of the 10 parameters, and in all 10 parameters the TCNN and RNN outperformed than the standard linear regression.

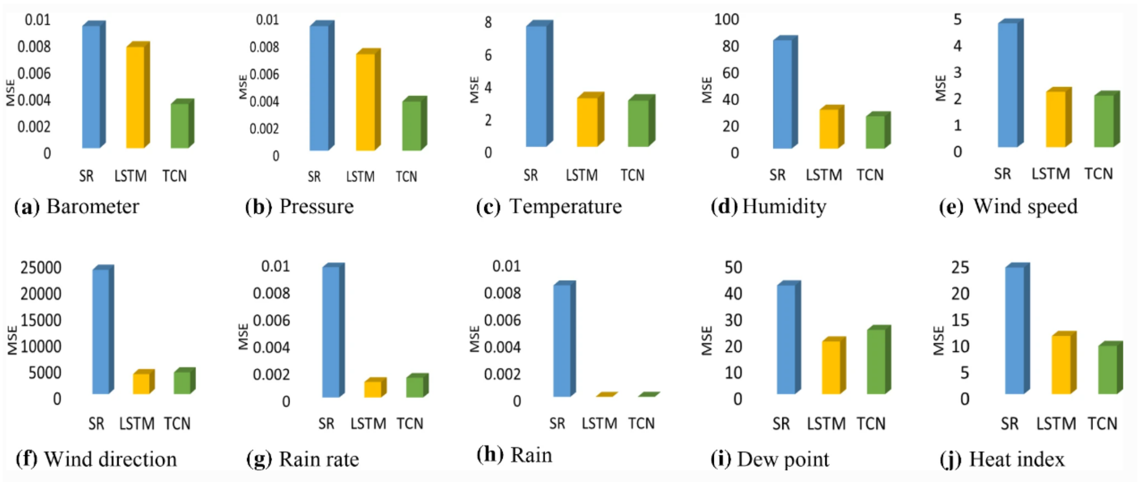

Figure 3.10: Analysis of different techniques in predicting different weather parameters: SR (standard regression), LSTM, TCNN [45]

In the study conducted by Liu et al.[24], LSTMs, GRUs, TCNNs and traditional ARIMA models, among many other models, were put side by side for stock closing price prediction, and 3 other timeseries datasets. The experimental results show that the TCNN converge faster and have better performance than the other models tested. For example, for the stock closing price dataset the TCNN had a MAPE of 0.027 compared to 0.037 of LSTM, 0.035 of GRU and 0.36 of ARIMA. As expected, the more traditional ARIMA model has the worst performance, by far, out of the 4 models described.

Lastly, Dubey et al.[11] used SARIMA and LSTM were used in order to forecast energy consumption. Amongst all metrics used, it is clear that the LSTM (RMSE=0.231, MAPE=3.221) performed considerably better than SARIMA (RMSE=0.550, MAPE=5.236). The authors reported that, as is to be expected, LSTM was found to be more prominent with the increasing lags and input data and is best in comparison SARIMA with all the cases.

Chapter 4

Bridge Cost Estimation

This chapter is focused on the first objective stated in 1.3. The main goal is to estimate a budget for a bridge construction project given a subset of features of the construction budget. To do so we will begin by explaining the budgeting datasets that we have access to, followed by the necessary data preprocessing actions, model building and finally the results that we have achieved.

4.1 Datasets

We have two main sets of datasets. Firstly, we have the initial dataset, that is in fact two similar datasets. These are the first data that we had access to and was provided by BERD. The data was in a very raw state and a significant work was put into the cleaning of these datasets. These datasets were later dropped as they had no real value for building cost estimation models. Secondly, is the dataset that was used for the model building and training, this dataset was not without its problems but was in a much more usable state than the previous ones.

4.1.1 Initial Datasets

The initial datasets provided by BERD, were extracted from the company's current database system, have a tabular structure and are made available in two different Excel files. These two datasets, are very similar but have overall distinct structures. The datasets made available are a dataset for salaries and a dataset for other and miscellaneous costs.

The datasets were not in any way ready for usage to train machine learning models. The data needed vast amounts of cleaning before being sent as input for AI models. This process is described in 4.2. The columns of both datasets can be seen in Table 4.1. Some columns had "N/A" signalized, which means not applicable, this was the case because that column in particular was not available in the corresponding dataset but was available in its counterpart dataset. As we can see, they have very similar structures, with 30 columns in common, the main difference being that one was targeting salaries costs and the other was targeting a multitude of different miscellaneous costs. The first is called "Salaries dataset" whilst the second is the "Other costs dataset". They are as follows:

- **Salaries dataset -** This dataset contains the information of salaries of 61 different professions related to bridge construction. It ranges from cleaning staff to architects and directors. The dataset has 25223 rows and 32 columns.

- **Other Costs dataset -**This dataset contains the information of salaries of 61 different professions related to bridge construction. It ranges from cleaning staff to architects and directors. The dataset has 25223 rows and 32 columns.

In the following Table 4.1 we can see all columns of both datasets and the percentage of missing values by column.

A preliminary Timeseries exploration of the data for the cost of both databases was performed. However, this was not very successful due to the lack of different timestamps in the "Date" column for each different category of items present in the "Original Description" column. Some categories of items had indeed only one time stamp, with the average being only 3 time stamps. With this amount of data, Timeseries visualization was not possible.

In Figure 4.1 is presented the category with the most variety of timestamps, "National minimum hour wage", it spans from May 4th of 2020 to July 6th of 2020. As we can clearly see it has a high standard deviation, this is mainly due to the large amount of different countries this data is extracted from, all with different wages.

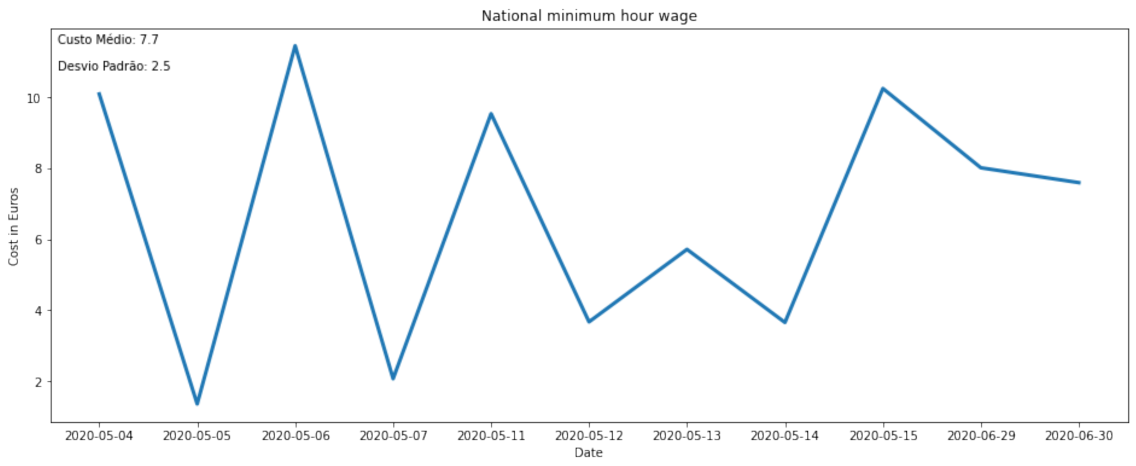

Figure 4.1: Minimum hourly wage Timeseries

As we will later describe in Section 4.2 the datasets have a multitude of problems from incoherent units to missing values, to a lack of overall data, as shown in its preliminary Timeseries exploration. Due to all these problems, even after the whole data cleaning executed shown in 4.2, both of these datasets were later dropped and not used in the training of the models.

4.1.2 Construction Data

This data was extracted from the Kaggle data sharing platform and is organized in tabular form in two CSV files. It contains construction costs for several projects, where each row represents a different construction project and each column is a construction item. In this way, each cell of these files is the quantity of the item present for the construction of the project (except the cells present in the last column that represent the total construction cost). In theory this dataset seems excellent for the challenge this thesis tries to face, however, its lack of clarity and explanation make the dataset not 100% reliable. It is, however, good for testing the case study we have in hand.

The dataset contains two CSV files, the main one has the construction costs in the format stated above, where each column represents a construction item and has a code number, whilst the other information about each item's code number like description and unit of measure. Hence, both files had to be used in conjunction for better interpretability.

4.2 Data Preprocessing

Data preprocessing is very important because it helps to ensure that the data being used for analysis or modeling is in a format that can be effectively and efficiently used, especially if the datasets are in such a raw state like the ones stated below. For the initial datasets, multiple techniques were performed to possibly salvage some of the data. However, the dataset could not been used for model training, even after all these operations.

4.2.1 Initial Datasets

As mentioned before, the data supplied by the company is still in a very raw state. Because of this, some data cleaning techniques were applied to both datasets in order to create two clean versions of said datasets.

Firstly the datasets were imported from the Excel files to two different pandas data-frames in python, to better auxiliate all further transformations to the data, as this is a very useful library for data analysis and transformation, which allows for powerful and easy-to-use operations.

In the following subsections, all these operations will be expanded upon and explained.

Unifying Cost Columns

In the datasets supplied there is no single column regarding the cost of each item, instead there are several different columns with this purpose. The "Other Value" column is the most important column for the cost, as it is the exact cost for the item in the other costs dataset. One problem of this column is its high percentage of missing values (55.0%).

To solve this problem, and to not lose more than half of the data, an aggregation with the "Average" and "Median" column was pursued. Once these three columns were joined, with priority firstly for the "Other Value" column, secondly for the "Average" and lastly for the "Median", the newly created "Cost" column has only 6.2% of missing values in the other costs dataset. A similar approach was done for the salaries dataset, but since the "Other Value" column is inexistent in this dataset only the "Average" and the "Median" were used, this left the "Cost" column with only 5.7% missing values in this dataset.

A diagram exemplifying this process can be seen in Figure 4.2.

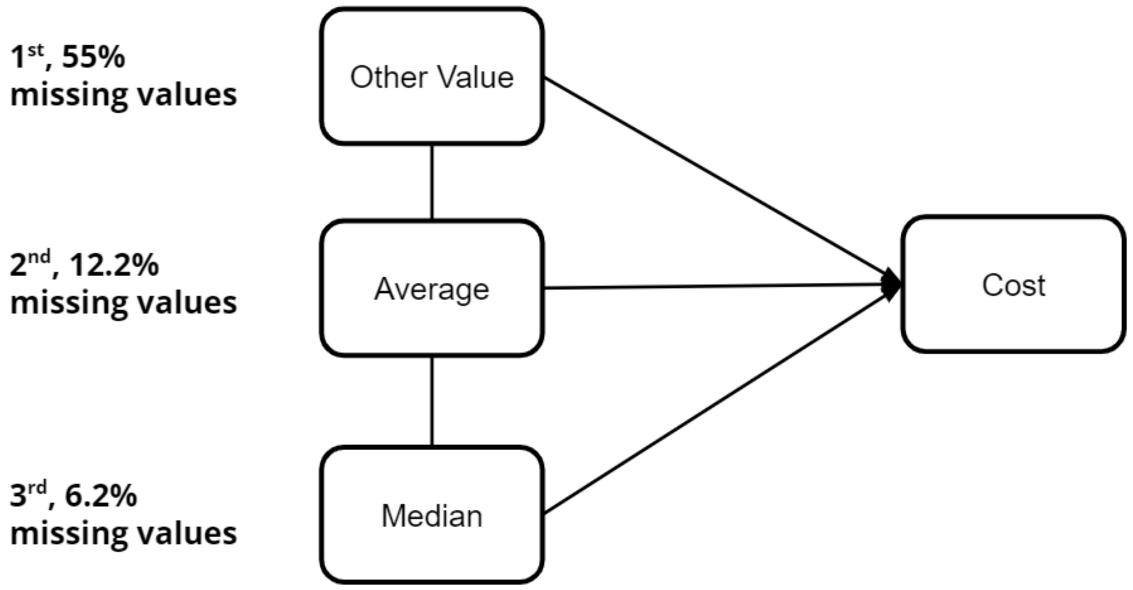

Figure 4.2: Creation of a new cost column diagram

Treating Missing Values

As we can see from Table 4.1 there is a lot of data missing from a great amount of columns, specially from the other costs dataset. The first step of this process was to remove all the useless columns, in other words columns with 100% of the values missing, or very close to this value.

The columns removed from the salaries dataset were: City, Expect. Max. Variat., Signs of speculation, Offer in the market and Scale Factor. In the other costs dataset the removed columns were: Item, Brand/Company, Min, Q10, Q25, Q75, Q90, Max, Average Bonus, Sample, Risk Profile. This left us with 27 columns in the salaries dataset and 28 columns in the other values dataset.

After the process in 4.2.1 there were still 5.7% and 6.2% missing values in the newly created column for the salaries and other costs datasets respectively. Because this data is very important and is not salvageable, these missing rows were removed from their corresponding datasets. The same approach was executed for the missing values in the Source Description and Unit columns, as this were the most important columns with missing values. This process could not be performed the same way for all columns as we have a large amount of missing values throughout most columns and if done blindly this process would leave the datasets completely empty.

Currency Conversion

When treating the "Currency" column, the first step was to deal with the missing values, more specifically in the other costs dataset. This column has 9.7% missing values, but these currencies can be inferred by the country column, as it has no missing values. Each currency missing was deduced by its "Country" column, this way the "Currency" column has no missing values.

The other costs dataset deals with 162 different currencies, whereas the salaries dataset has 12. This can add an unwanted complexity to the system. Because of this and to simplify the datasets for posterior AI model usage, the "Cost" column values were converted to their correspondent values in Euros at the timestamp available in the column "Date". This was done with the help of the "Exchange rate Api" and the "CurrencyConverter" module for python when the currency pair was not available in the former. Later the currency column was dropped since all values were in Euros.

Quantity Normalization

In the other costs dataset, the quantity of items is very varied. To simplify, when possible, this quantity value is normalized to 1, grouped by category given by the "Original Description" column. This procedure is not always possible for all categories, as some do not have linear costs. For example, the item with "Original Description", "Trucks 40ft", the cost is dependent on more than one variable, it is dependent both on the distance travelled by the truck and by the quantity of material transported, either in "m^3" or in "Ton". These cases need to be treated separately as they are more complex.

Unit Unification

Each category of items is dealt with separately when unit normalization is concerned. In the unit unification process, when more than one unit is available in the "Unit" column for the same category of items, a transformation is done in order to unify the units into a single one. When possible, International System of Units (SI) units are used. For example, "gallons" were converted into "liters", "tons" to "kilograms", etc.

Treating Miscellaneous Data

Because the categories ("Original Description") vary so much between each other, there was a need to perform data separation into many subsets of data based on their category. This way we avoid many of the problems encountered when trying to uniformize the whole datasets, more specifically the other costs dataset, which has much more varied data. This way we avoid, for example, the enormous problem that would occur with dealing with costs with and without "VAT", as nearly 70% of the values do not have a "VAT" associated, and if we used both values with and without "VAT" to train the same system it would develop an inaccurate system. By dividing categories into subsets, we would also divide it by whether it has a "VAT" associated with it. Later models developed for each category would also be separate.

Despite all the data cleaning done, there were still much more needed to be done, of which many of them needed to be manual. For example, misspelled errors when inputting the data, like currency names "GBO" and "GPB" instead of "GBP" for Great Britain Pound. And incoherence in the data, for example, sometimes a "Unit" is written as "kilogram" and some other times it is written as "kg". Many more miscellaneous errors like these were found throughout both datasets.

This dataset was later dropped and never used for any kind of model training.

4.3 Model Building

For the implementation of cost estimation regression models for the total cost of building a bridge, the following models were chosen:

- Linear Regression
- Random Forest
- XGBoost
- MLP

The first objective is accomplished by using the **Construction Data dataset**, as it has the cost of 1451 different construction projects. The data was also normalized to facilitate the models training.

Feature Selection

This dataset has, however, 3393 different items (or features). This can seem like a good thing at first, but this is not always the case when we are dealing with a large quantity of features, as most of these features act as noise for the models and cause them to have worse performance. As stated by Domingos in [10], a reduction in feature space can often lead to achieving better predictive performance.

In order to determine which of the features had the most importance, we trained a random forest regressor with 150 estimators and extracted a list of all features sorted in descending order by importance. The 20 most important features can be seen in Figure 4.3. As we can see from the figure, the importance of each feature is very disproportional for estimating the cost of building projects.

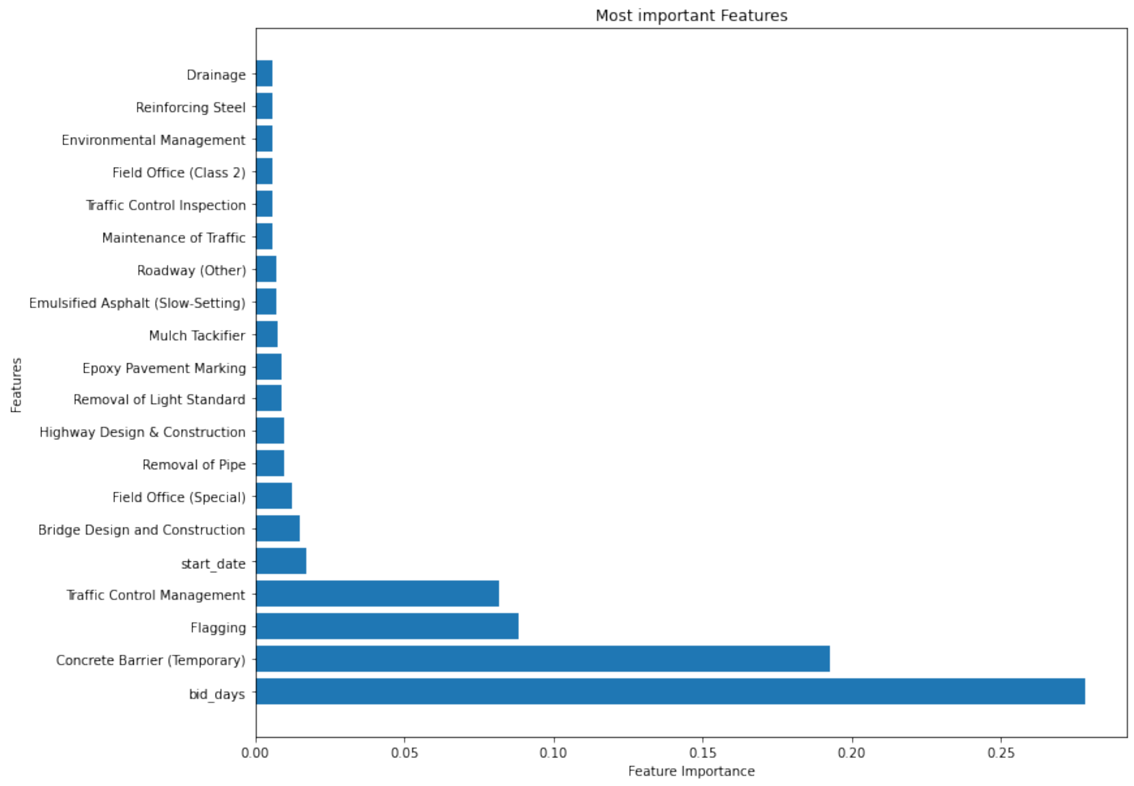

Figure 4.3: The 20 most important feature in the Construction Data dataset

By selecting the features with the most importance, we are able to increase the performance of our models. In order to do so, we used Forward Feature Selection (FFS). This process can be visualized from Figures 4.4 to 4.6. As explained previously, we increase the number of features one by one, by feature importance, until the standard deviation of the last 20 iterations is less than 0.005 or more than 120 features have been added. After that, the subset of features with the best performance is selected. As we can see from the figures below, more features does not always mean a better performance.

For the MLP we were not able to do the FFS duo to long duration of the training for each iteration conjugated with the cross validation that was performed. However, we utilized the features that were selected by the FFS for the XGBoost, just so we could have some feature selection rather than none.

Training and Optimization

The models were initially trained using the libraries default values provided for each one, in order to allow for a preliminary comparison. The RF and XGBoost were also trained using different numbers of estimators. The MLP, however,

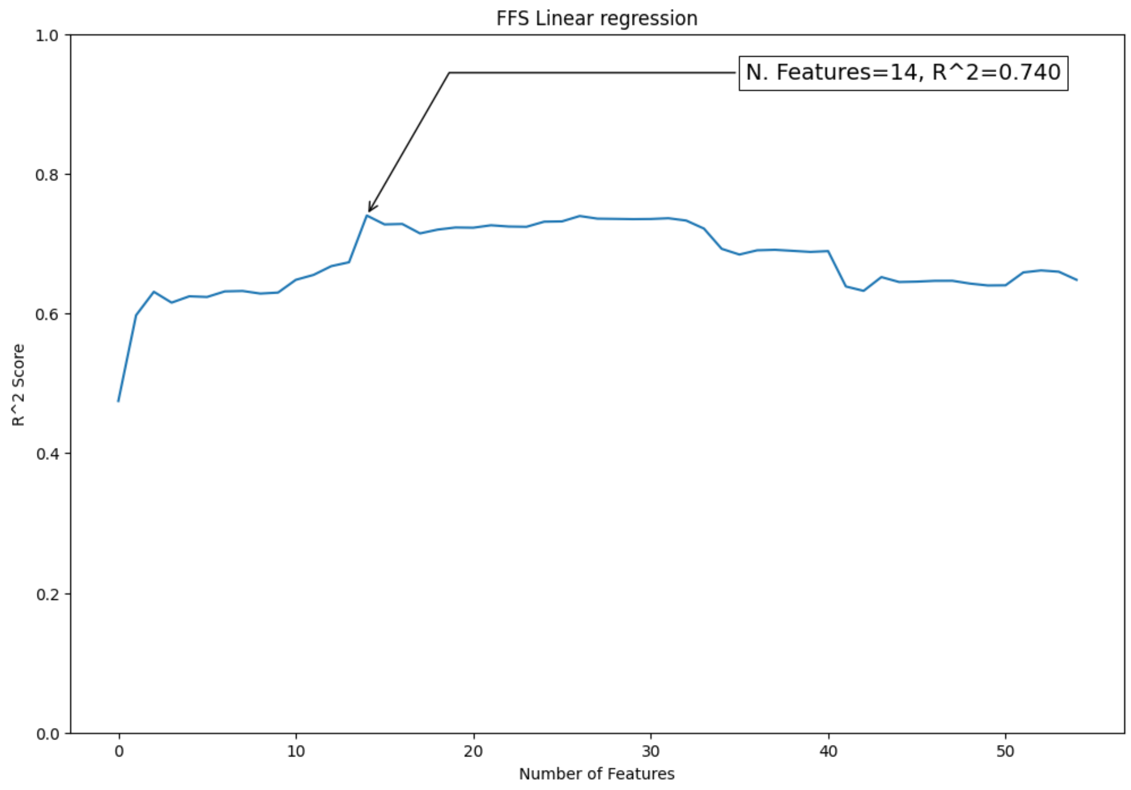

Figure 4.4: FFS of a linear regressor

underwent a process of hyperparameterization to better select the best possible model. The hyperparameterization method chosen was a grid search.

Model Validation

Data is frequently divided into training data and test data in order to test the prediction abilities of the generated model. Additionally, when comparing various models, testing out various features, or fine-tuning hyperparameters, an additional split of the training data and the creation of a validation set can be used. It's known as holdout validation. Cross-validation is an additional validation method. The data is divided into k equally sized segments or folds in the most typical scenario, known as k-fold cross-validation. Then, k iterations of training and validation are carried out, with each iteration holding out a different fold of the data for validation while using the remaining k—1 folds for training the model.

The trained models were compared using 5-fold cross-validation.

Scoring Metric

The chosen scoring metric for this task was the R^2 metric. It was mainly chosen because of its readability as its values range from 0 to 1, with a higher value indicating a better fit of the model.

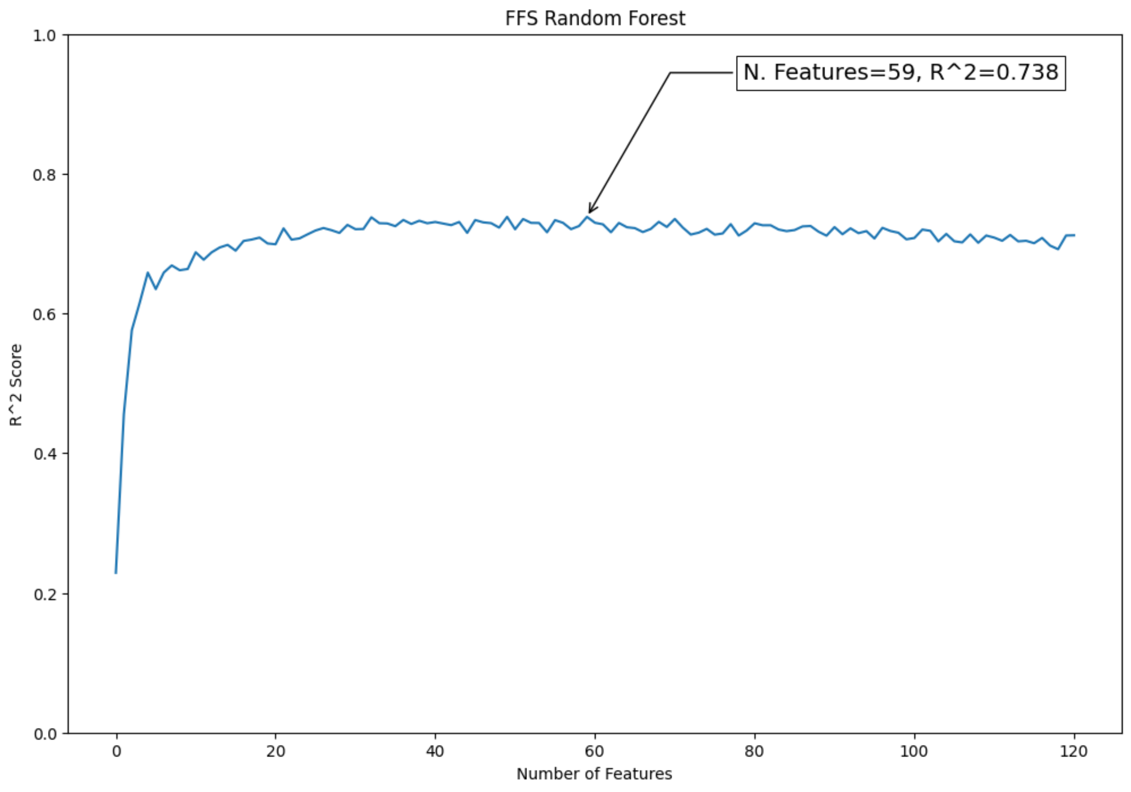

Figure 4.5: FFS of a random forest

4.4 Results and Discussion

The following Table 4.2 presents the average R^2 results for the 4 models developed, as well as its standard deviation in the 5-Fold cross validation. As it would be expected, the MLP was the model with the best overall performance, with an average R^2 of 0.878, which is significantly better than the others. It is also worth noting that the MLP model was the only one that had hyperparameterization done, so the results could have been more alike had this been performed for all other models. One surprise was that the XGBoost model was the worst performing one with only 0.686, whilst the more simple Linear Regressor (LR) was the second best with 0.740, with very similar results to the RF, that averaged 0.738, even though with a higher standard deviation. Most likely this is the case because of lack of good parameters for the XGBoost, as a simpler model like the LR does not need much parameterization.

These results are promising nevertheless, especially the MLP model. It shows some potential for bridge cost forecasting, given the right dataset. But, in order to have better results, some future work is needed. For instance, a grid search should be performed on the parameters of the LR, RF and XGBoost. A broader grid search should be performed on the MLP, and, given the time, a better feature selection should be executed for it as well.

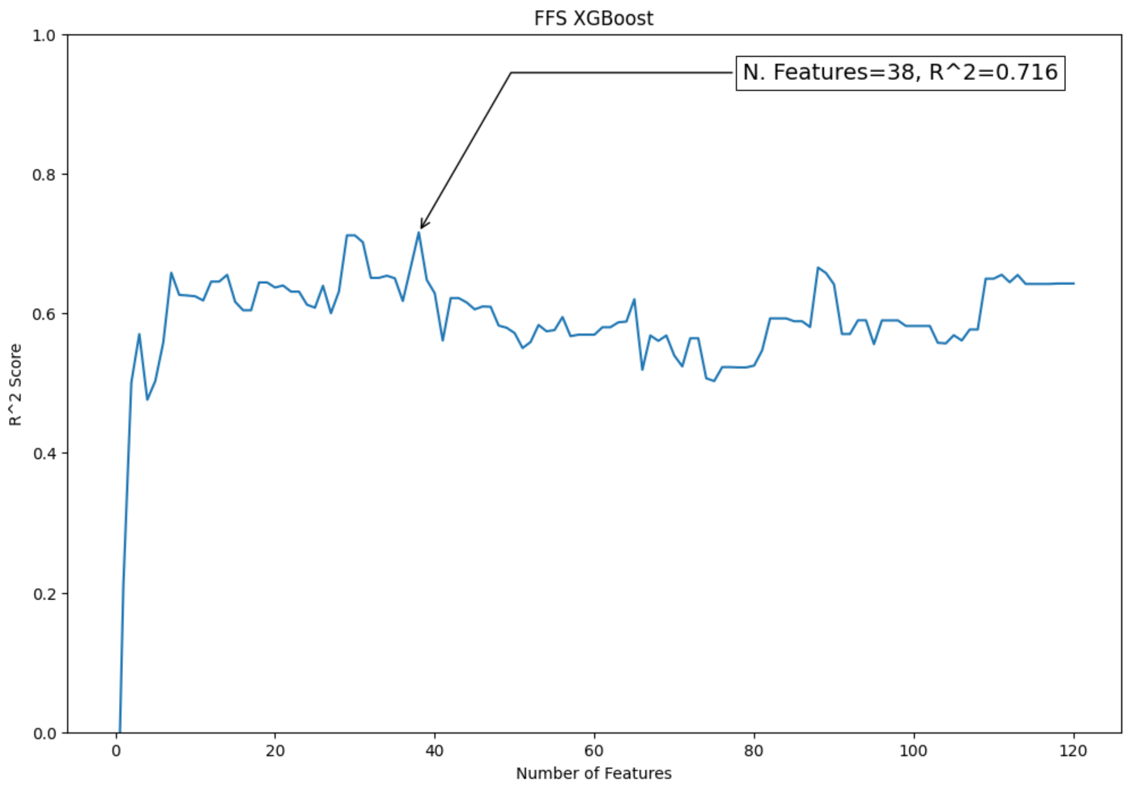

Figure 4.6: FFS of a XGBoost regressor

Table 4.1: Percentage of missing values by column

	Salaries dataset	**Other Costs dataset**
Priority	N/A	14.8%
Item	0.0%	100%
Code	0.0%	0.0%
Original Description	0.0%	0.0%
Source Description	0.0%	3.9%
Brand/Company	N/A	85.1%
Currency	0.0%	9.7%
Min	71.1%	90.8%
Q10	59.0%	100.0%
Q25	65.4%	98.8%
Average	41.4%	56.8%
Median	25.9%	94.0%
Q75	65.4%	98.8%
Q90	59.0%	100.0%
Max	71.1%	93.2%
Average Bonus (Cur/year)	42.1%	100.0%
Sample	69.9%	94.3%
Other Value	N/A	55.4%
Quantity	N/A	53.4%
Risk Profile	N/A	97.0%
Vat	N/A	72.8%
Unit	0.0%	0.4%
Source	0.0%	1.2%
Registed By	N/A	3.6%
Link	0.0%	3.6%
Country	0.0%	0.0%
State	25.9%	90.6%
Region	N/A	80.6%
City	100.0%	64.7%
Date	0.0%	0.0%
Reliability index	0.0%	36.5%
Expect. Max. Variat.	100.0%	64.6%
Perfectly (Matches)	0.0%	1.2%
Signs of speculation	100.0%	64.6%
Offer in the market	100.0%	64.6%
Scale Factor	100.0%	58.8%
Time Factor	N/A	69.3%
Last Updated	6.6%	1.5%
Observations	72.9%	74.7%
Pivot_Aux1	0.0%	N/A
Pivot_Aux2	0.0%	N/A

Table 4.2: Cost estimation models performance

	R^2	σ
Linear Regressor	0.740	0.095
Random Forest	0.738	0.132
XGBoost	0.686	0.081
Multilayer Perceptron	**0.878**	0.098

Chapter 5

Time Series Forecasting

This chapter is focused on explaining the whole process of fulfilling the second objective, time series forecasting. The goal is to create a forecasting system capable of forecasting the cost of an item chosen by the user for a specific date also chosen by the user. In a more real scenario we will have to forecast the cost of each item, hence, in order to be more precise with the budgeting, it is necessary to forecast items that comprise the overall budget of a bridge. Because of this, if we are able to have historical information of a given material, it should be possible to estimate a real budget at any given temporal point. Forecasting tests were performed by having cost values of specific items of a real bridge budget conjugated with historical cost values available in a UK database. It is worth noticing that the scenario is not real and not totally correct, as the UK database has the cost values for the UK and the bridge budget is referent to Portugal values mas it allows us to test our approach.

We will begin by explaining the datasets available, the model building process, and discuss the results achieved.

5.1 Datasets

For this objective, we have three available datasets. We have a real bridge budget dataset, that, we can use in conjunction with historical cost values of matching items. The datasets containing those historical cost values can either be the UK's monthly building materials and components costs or the weekly costs of petroleum products in European countries. Unfortunately, there is no match for any petroleum product in the real bridge budget dataset, so we had to use only the UK's dataset.

In the following subsections, we will explain those datasets.

5.1.1 Porto Bridge Budget

This dataset was also provided by BERD. It is perhaps one of the most important datasets used in this study, since it is a real budget estimation for a real bridge of the Porto metropolitan system over the Douro River with accesses between Porto (Campo Alegre) and Vila Nova de Gaia (Candal). This data is, however, of a more confidential nature than the previous ones, hence not a lot of information about it can be disclosed.

Similarly to the other data made available by the company, this data is organized in tabular form (Excel), which facilitates their interpretation. The main purpose and usage of this dataset is, when combined with a Timeseries dataset, to allow estimation of the cost of an item, that has a match in both datasets, for a given date selected by the user.

5.1.2 Petroleum Products

This dataset was also made available in a tabular format, also an Excel file, and refers to the price of 3 oil products (Automotive gas oil, Euro-super 95 and Fuel oil - Sulphur less than 1%) over a period of 4 years (2018 to 2022) with a weekly granularity for 10 European countries of interest (Belgium, Denmark, France, Germany, Italy, Netherlands, Poland, Portugal, Spain and the United Kingdom).

Temporal Analysis

A temporal analysis was performed on this dataset. This analysis can be seen in Figure 5.1, where one can clearly see the upward trend in price, in euros per 1000 liters, of Euro-super 95 fuel over time (yellow line) in Belgium.

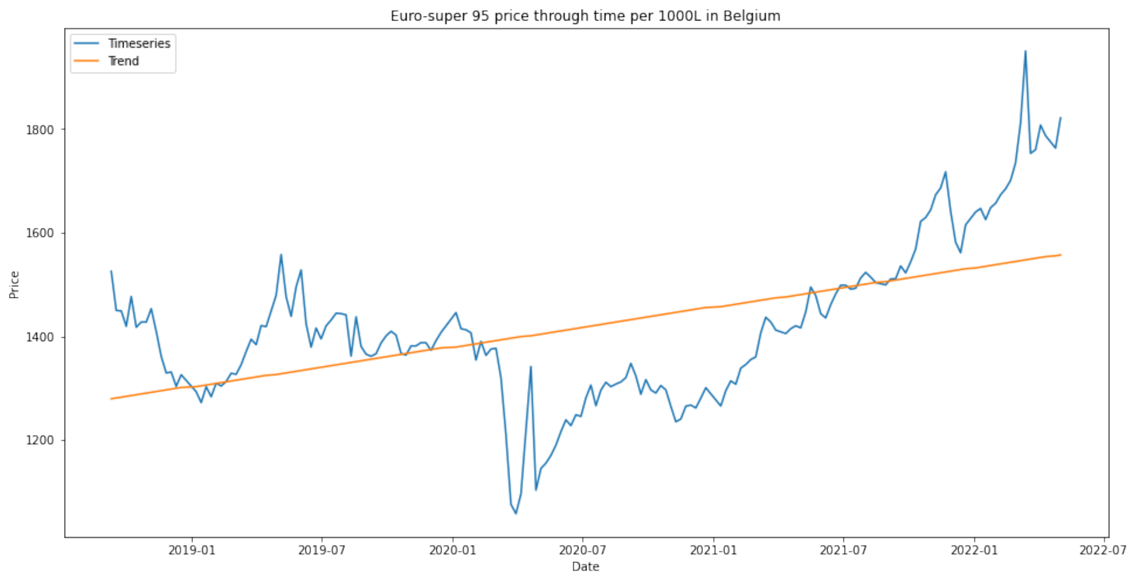

Figure 5.1: Euro-super 95 price per 1000 liters in Belgium Timeseries and Trend

5.1.3 Building Materials and Components: Monthly Costs

This dataset was collected from the official UK government statistics website and contains monthly costs over a 13-year period (2009 to 2022) of various construction items.

The wide variety of items and a long sample period make this dataset quite useful and essential for the challenge at hand. In this dataset, each column represents a different item, while each row represents a different date.

This dataset contains information about 40 different items, however the main items that we will be focusing as part of our study are **Ready-mixed Concrete** and **Fabricated structural steel**, as they are the ones with the better item match in the real bridge budget dataset.

The downside of this dataset is that its values do not represent an absolute cost of the item temporally, but rather a relative cost index, indexed to the cost of that same item in the year 2005. What this means is that the cost of that item in January 2005 is 100 and every other cost is relative to that, i.e. if the price is 20% more at any given date, then its value in the dataset would be 120. Either way, it is enough when we combine it with the budget estimate data set provided by the BERD.

Temporal Analysis

In Figures 5.2 and 5.3 we can observe the trend and seasonality of Ready-mixed concrete, respectively. The slope of the trend line (yellow) allows us to have a clear idea of how the cost of cement has increased over the years. What is not so clear to extract from the timeseries in Figure 5.3 is its seasonality, however, once we separate the timeseries seasonality and its noise we can quickly realize that Ready-mixed concrete has a very evident annual seasonality of its cost. The same analysis for fabricated structural steel can be seen in the Appendix A.

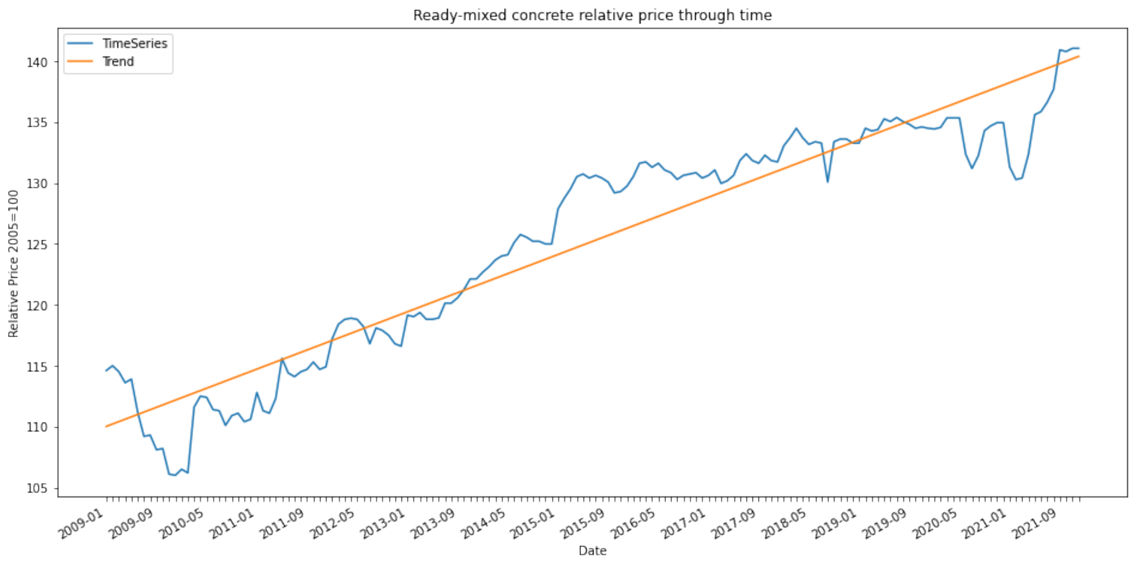

Figure 5.2: Ready-mixed concrete timeseries and trend

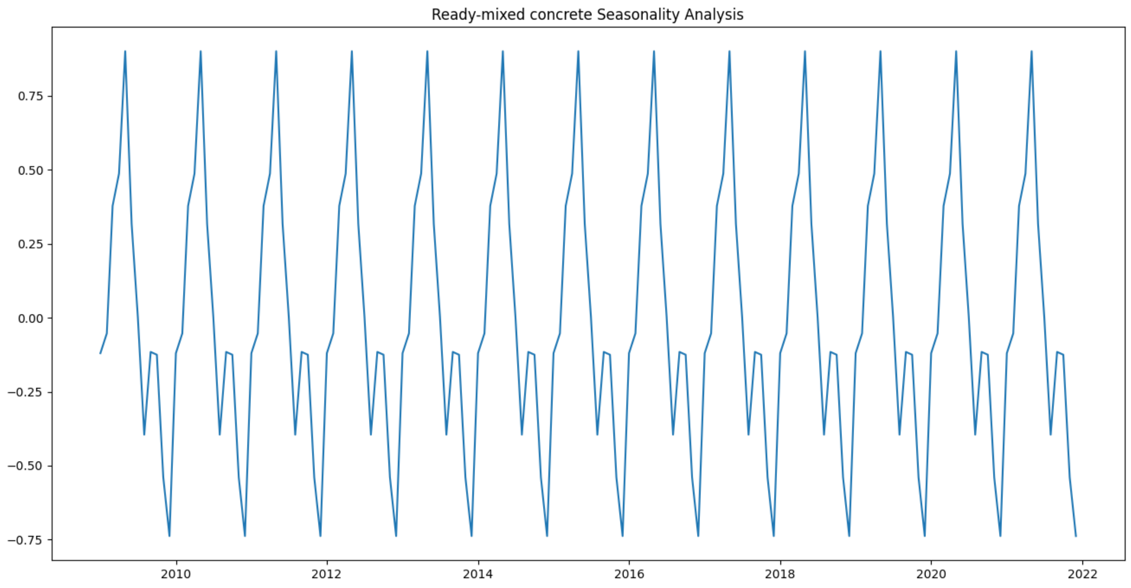

Figure 5.3: Ready-mixed concrete seasonality

5.2 Data Preprocessing

The data preprocessing described in the subsection below was a necessary one. Without this data preprocessing, it would be impossible to train any model on the raw data. This is due to the data being separated in multiple files, some with different formats between each other. Because of this, this was one of the most important tasks for the second objective (time series forecasting)

5.2.1 Building Materials and Components: Monthly Costs

The dataset is publically made available by the official UK government statistics website, what makes this a very reliable data source. However, it is not without its issues. The original data is divided in multiple Excel files (some with different formats from each other), one file per month, and is updated every month.

The data is very useful for our use case, however, because this data was not available in one single dataset, in order to use it for time series model training purposes a sort of gluing process had to be performed. Because of the irregularities between Excel files (some had an item that another didn't, an item had a slightly different name or was located in a different place, etc.) this process had to be done mostly manually and took a long time to perform, especially because the quantity of Excel files that exist (one for each month for a period of 13 years, a result 152 files).

This dataset had some missing values in certain items as well. Because of this, in order to use those timeseries, some values had to be interpolated based on previous and following values.

The train subset of the data was also normalized to facilitate the models training.

After all these preprocessing tasks were performed, the dataset was ready to be used to train AI regression models.

5.3 Model Building

For this task in particular, a system capable of forecasting the cost of a selected item for a specific date given by the user was developed. Essentially, the 3 following methods were established:

1. A method that trains a model given parameters by the user and makes a forecast using that same model.

2. A method that trains a series of models using a grid search and makes a forecast using the best model found.

3. A method that makes the forecast from a model that is already saved.

These 3 methods are divided into 2 different python files. The first and second methods are contained into a training file, whilst the other is in a forecasting file. The methods were separated so that if the user only wants to do a forecast and already has a model saved, they will find a more user-friendly amount of parameters.

Experimental Setup

To use the system above explained, the user has two choices, they can either use the command line to insert every parameter that they want to provide (default parameters are given if the user chooses not to change them, except for the mandatory parameters, as is for example the case of the target date for which to do the forecast) or they can use the command line with the name of a configuration file in a *json* format that contains the parameters to change.

The main parameters that can be changed, either by usage of a configuration file or with the command line, can be seen in Table 5.1.

Training and Optimization

For the creation of time series forecasting models for specific items of a bridge construction project, the following models were chosen:

- Recurrent Neural Network (Vanilla)
- LSTM
- GRU
- Temporal Convolutional Neural Network

All models were trained and optimized using the grid search hyperparameterization process stated above. This way, we can guarantee the best model possible out of the given parameters for the grid search.

The parameters used in the grid searches can be seen in Table 5.1 and are in a list format, values that are not in a list format are not part of the grid search but are important parameters as well.

Model Validation

For model validation, we used a train-test split of the time series data. For the train subset we used 80% of the data and the rest 20% was used for the test.

Scoring Metrics

We used 5 different scoring metrics to evaluate our models. By using more than one metric, we can have a better understanding of the performance of the models.

The chosen metrics were:

- Mean Absolute Percentage Error

Table 5.1: Time series system parameters and grid search values

Parameter	**Value**
Item	Ready-mixed concrete
Target Date	16/01/2023
Reference Date	01/07/2021
Dropout	[0,0.1,0.2]
RNN Hidden Layer Dimensions	[80,100,160]
Number of Hidden Layers	[1,3,5,7,9]
Batch Size	[16,32,64]
N. Epochs	[100,300,600]
RNN Model Flavor	["LSTM","GRU","RNN"]
Output Chunk Length	[1,3,6]
Periodicity	12
Train Begin Percentage	0
Train End Percentage	0.8
Dilatation Base	[2,3,4]
Kernel Size	[5,6,7]
N. Filters	[4,8,10,12]

- Root Mean Squared Error
- Root Mean Squared Percentage Error
- R^2

5.4 Results and Discussion

As previously stated, we try to forecast the cost of a specific item for a given date. To do so, we have to choose what item to train our models on. We tried to find the items that had the best match with items in the real bridge budget provided by BERD. The two best matches were:

- Ready-mixed concrete
- Fabricated structural steel

In the following Tables 5.2 and 5.3 we present the performances of all models trained for the validation subset. And in Figures 5.4 and 5.5 one can visualize the best performing models for each item.

The results show that the models were capable of producing good prediction capabilities. The TCNN and GRU as expected were the best performing models. While TCNN was clearly better for the *Ready-mixed concrete* item, GRU performed better for the *Fabricated structural steel*.

The remaining model forecasts can be seen in Appendix B.

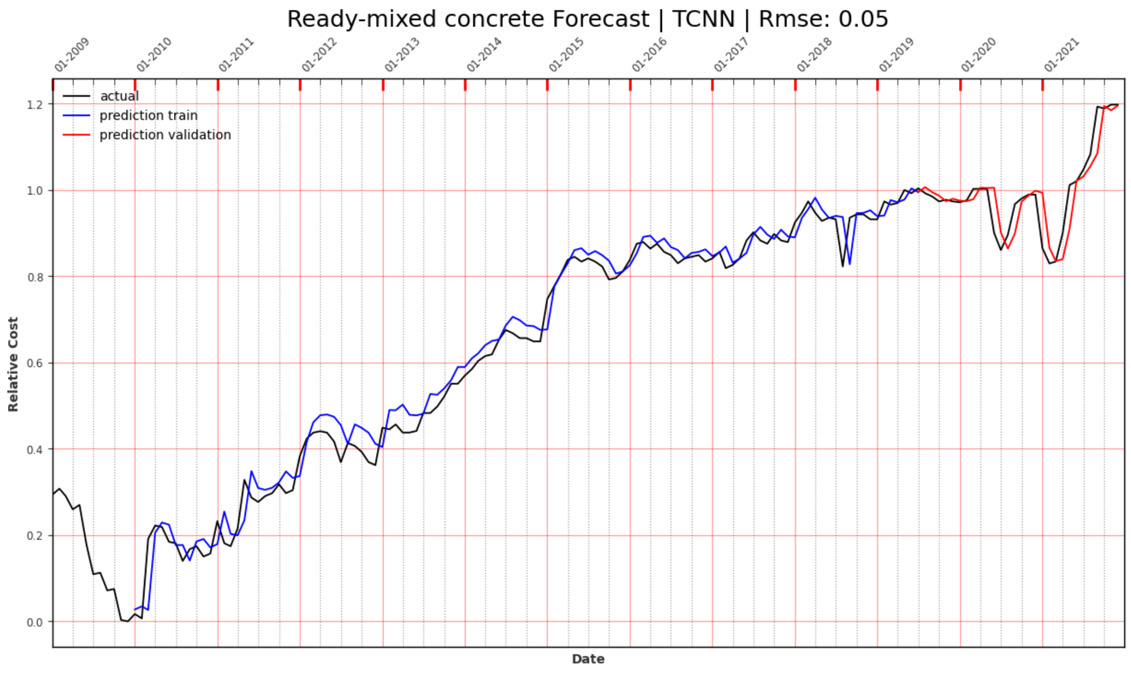

Figure 5.4: Ready-mixed concrete TCNN model forecast

Table 5.2: Ready-mixed concrete models performance

	MAPE	**RMSE**	**RMSPE**	R^2
Vanilla RNN	3.609	0.051	0.056	0.720
LSTM	3.667	0.050	0.055	0.737
GRU	3.445	0.049	0.052	0.755
TCNN	**2.972**	**0.046**	**0.049**	**0.777**

It is important to note that the validation subset coincides with the COVID-19 period, this clearly influenced construction material costs as can be seen by the large fluctuation in price over that period. This is more evident in Figure 5.5. Even so, the models were able to predict good results.

BERD stated that an error below 10% is a good enough error. Keeping this in mind, it is clear that the models performed very well.

Although these results are very promising, it is always possible and important to improve them. In future work, one possibility is to expand the grid search, or even use multiple time series for a more robust forecasting system.

5.4.1 Prediction Test

As previously stated, the whole objective of this system is to predict the cost of an item for a specific date. In order to put that to the test, we ran a simulation.

Because we need to have a comparison between the real value and the predicted value, we used the real value of *Ready-mixed concrete* in the real budget dataset and the predicted for the same date.

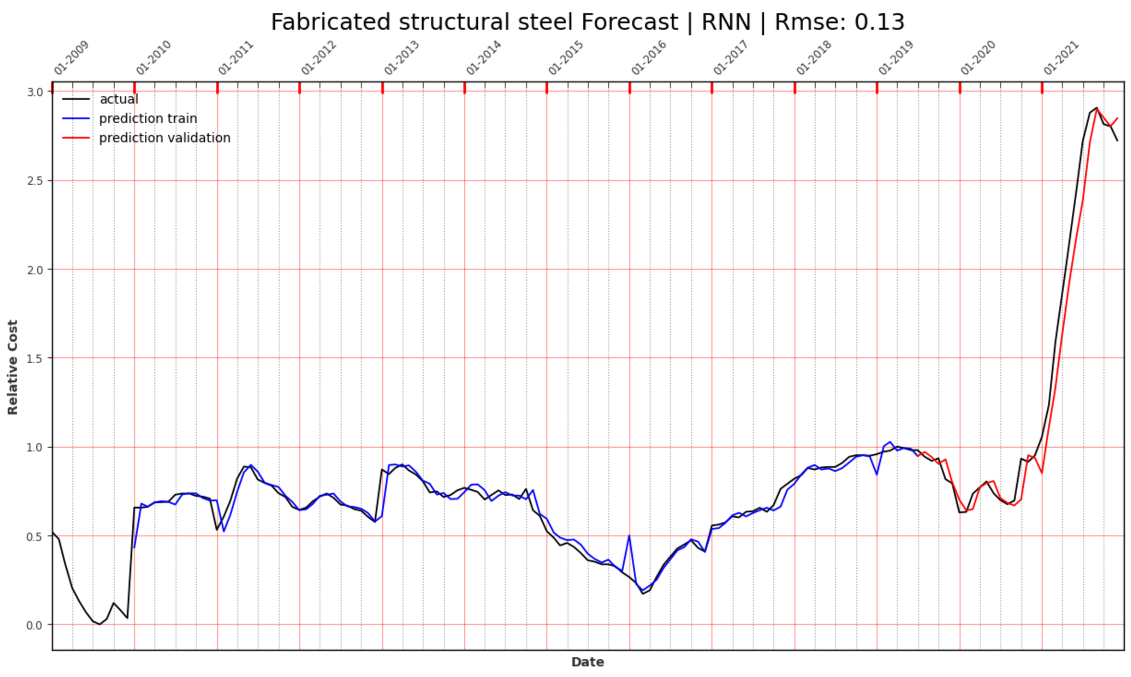

Figure 5.5: Fabricated structural steel GRU model forecast

Table 5.3: Fabricated structural steel models performance

	MAPE	**RMSE**	**RMSPE**	R^2
Vanilla RNN	8.945	0.197	0.112	0.942
LSTM	9.285	0.220	0.119	0.928
GRU	**6.731**	**0.134**	**0.092**	**0.974**
TCNN	8.552	0.172	0.114	0.956

The real cost value of *Ready-mixed concrete* in the budget is **€1528920** in July 1st 2021, whilst the model (TCNN as it was the best performing for this item), forecasted that the cost would be **€1591152**. This is an error of plus **€62232**, or plus **4.07%**. This is a good forecast because it is way below of the 10% margin of error that BERD considers to be a good compromise of acceptable error.

Chapter 6

Conclusion

The main obstacles of the first part of the thesis were the understanding of the problem and its main objectives, the context and the data. This can be seen in the rather significant work done in data cleaning the first dataset that ultimately led to nothing.

From the study of the state of the art, it is safe to assume the importance that the automation of bridge cost forecast can have for the success of the construction project. This thesis demonstrates that it is possible to deploy Artificial Intelligence (AI) models that can alleviate and accelerate a process that is mostly manual and tedious. The main obstacle in this kind of project is the lack of good and useful data that models can train upon.

Nevertheless, our results showed the capacity of these Artificial Intelligence (AI) in producing good forecasts. Even if some models perform better than others, for the most part, these models show a promising collaboration between experts in budgeting and AI.

We tackled two different scenarios of bridge cost forecasting, and observed low errors, even with low quality data on the first problem. This proves that it is possible to use an approach of this kind for bridge cost forecasting.

In the time series forecast problem, we executed multiple exhaustive searches on multiple models to better forecast two items that have an impact on the final budget. We also delivered models that have guarantees of a relative low error. It is worth noting that this system is a proof of concept and that some historical data may not be accurate because the cost values are referent to UK's costs and not Portugal ones. This is also something that could be improved, given quality data for Portuguese construction costs. Nevertheless, the system remains good at temporal cost forecast. A possible next step would be the continuous collaboration with a company like BERD, that can deliver quality data, and test the usefulness of different models for the use case at hand.

References

[1] Rahebeh Abedi, Romulus Costache, Hossein Shafizadeh-Moghadam, and Quoc Bao Pham. Flash-flood susceptibility mapping based on xgboost, random forest and boosted regression trees. *Geocarto International*, 37(19):5479–5496, 2022.

[2] Hojjat Adeli and Mingyang Wu. Regularization neural network for construction cost estimation. *Journal of construction engineering and management*, 124(1):18–24, 1998.

[3] Dinu Philip Alex, Mohamed Al Hussein, Ahmed Bouferguene, and Siri Fernando. Artificial neural network model for cost estimation: City of edmonton's water and sewer installation services. *Journal of construction engineering and management*, 136(7):745–756, 2010.

[4] David LJ Alexander, Alexander Tropsha, and David A Winkler. Beware of r 2: simple, unambiguous assessment of the prediction accuracy of qsar and qspr models. *Journal of chemical information and modeling*, 55(7):1316–1322, 2015.

[5] Othman Subhi Alshamrani. Construction cost prediction model for conventional and sustainable college buildings in north america. *Journal of Taibah University for Science*, 11(2):315–323, 2017.

[6] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[7] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[8] Matteo Cinelli, Giovanna Ferraro, Antonio Iovanella, Giulia Lucci, and Massimiliano M Schiraldi. A network perspective on the visualization and analysis of bill of materials. *International Journal of Engineering Business Management*, 9:1847979017732638, 2017.

[9] Adele Cutler, D Richard Cutler, and John R Stevens. Random forests. In *Ensemble machine learning*, pages 157–175. Springer, 2012.

[10] Pedro Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87, 2012.

[11] Ashutosh Kumar Dubey, Abhishek Kumar, Vicente García-Díaz, Arpit Kumar Sharma, and Kishan Kanhaiya. Study and analysis of sarima and lstm in forecasting time series data. *Sustainable Energy Technologies and Assessments*, 47:101474, 2021.

[12] Roy Duff, Margaret Emsley, Michael Gregory, David Lowe, Jack Masterman, and W Hughes. Development of a model of total building procurement costs for construction clients. In *14th Annual ARCOM Conference*, pages 210–218. Association of Researchers in Construction Management, 1998.

[13] TMS Elhag and AH Boussabaine. An artificial neural system for cost estimation of construction projects. In *Proceedings of the 14th ARCOM annual conference*, 1998.

[14] HMH Hegge and JC Wortmann. Generic bill-of-material: a new product model. *International Journal of Production Economics*, 23(1-3):117–128, 1991.

[15] Jiun-Chi Huang, Yi-Chun Tsai, Pei-Yu Wu, Yu-Hui Lien, Chih-Yi Chien, Chih-Feng Kuo, Jeng-Fung Hung, Szu-Chia Chen, and Chao-Hung Kuo. Predictive modeling of blood pressure during hemodialysis: A comparison of linear model, random forest, support vector regression, xgboost, lasso regression and ensemble method. *Computer methods and programs in biomedicine*, 195:105536, 2020.

[16] Jianxin Jiao, Mitchell M Tseng, Qinhai Ma, and Yi Zou. Generic bill-of-materials-and-operations for high-variety production management. *Concurrent Engineering*, 8(4):297–321, 2000.

[17] Yao Jing-zheng and Han Duan-feng. Research and application of ship design and maintenance integration model based on bom. In *2010 2nd International Conference on Mechanical and Electronics Engineering*, 2010.

[18] Alan Jović, Karla Brkić, and Nikola Bogunović. A review of feature selection methods with applications. In *2015 38th international convention on information and communication technology, electronics and microelectronics (MIPRO)*, pages 1200–1205. Ieee, 2015.

[19] Jayant Kalagnanam, Moninder Singh, Sudhir Verma, Michael Patek, and Yuk Wah Wong. A system for automated mapping of bill-of-materials part numbers. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 805–810, 2004.

[20] Dennis Kallina and Patrick Siegfried. Optimization of supply chain network using genetic algorithms based on bill of materials. *The International Journal of Engineering & Science*, 2021.

[21] Bekir Karlik and A Vehbi Olgac. Performance analysis of various activation functions in generalized mlp architectures of neural networks. *International Journal of Artificial Intelligence and Expert Systems*, 1(4):111–122, 2011.

[22] Heng Li, QP Shen, and Peter ED Love. Cost modelling of office buildings in hong kong: an exploratory study. *Facilities*, 2005.

[23] Min Liu, Jianbo Lai, and Weiming Shen. A method for transformation of engineering bill of materials to maintenance bill of materials. *Robotics and Computer-Integrated Manufacturing*, 30(2):142–149, 2014.

[24] Yujie Liu, Hongbin Dong, Xingmei Wang, and Shuang Han. Time series prediction based on temporal convolutional network. In *2019 IEEE/ACIS 18th International Conference on Computer and Information Science (ICIS)*, pages 300–305. IEEE, 2019.

[25] David J Lowe, Margaret W Emsley, and Anthony Harding. Predicting construction cost using multiple regression techniques. *Journal of construction engineering and management*, 132(7):750–758, 2006.

[26] Francesco Maiorana and Angelo Mongioj. A data mining approach for bill of materials for motor revision. In *2012 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 1091–1096. IEEE, 2012.

[27] KF Marr. Standards for construction cost estimates. *Transactions of the American Association of Cost Engineers,(Paper B-6)*, pages 77–80, 1977.

[28] Jeremy Miles. R-squared, adjusted r-squared. *Encyclopedia of statistics in behavioral science*, 2005.

[29] MZ Naser and Amir Alavi. Insights into performance fitness and error metrics for machine learning. *arXiv preprint arXiv:2006.00887*, 2020.

[30] Kai A Olsen, Per Sætre, and Anders Thorstenson. A procedure-oriented generic bill of materials. *Computers & industrial engineering*, 32(1):29–45, 1997.

[31] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.

[32] Ashutosh Pandey and DeLiang Wang. Tcnn: Temporal convolutional neural network for real-time speech enhancement in the time domain. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6875–6879. IEEE, 2019.

[33] Charlotte Pelletier, Geoffrey I Webb, and François Petitjean. Temporal convolutional neural network for the classification of satellite image time series. *Remote Sensing*, 11(5):523, 2019.

[34] V Rodriguez-Galiano, M Sanchez-Castillo, M Chica-Olmo, and MJOGR Chica-Rivas. Machine learning predictive models for mineral prospectivity: An evaluation of neural networks, random forest, regression trees and support vector machines. *Ore Geology Reviews*, 71:804–818, 2015.

[35] Carol J Romanowski and Rakesh Nagi. A data mining approach to forming generic bills of materials in support of variant design activities. *J. Comput. Inf. Sci. Eng.*, 4(4):316–328, 2004.

[36] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[37] Bibhuti Bhusan Sahoo, Ramakar Jha, Anshuman Singh, and Deepak Kumar. Long short-term memory (lstm) recurrent neural network for low-flow hydrological time series forecasting. *Acta Geophysica*, 67(5):1471–1481, 2019.

[38] Anil Sawhney and André Mund. Adaptive probabilistic neural network-based crane type selection system. *Journal of construction engineering and management*, 128(3):265–273, 2002.

[39] Maxim Vladimirovich Shcherbakov, Adriaan Brebels, Nataliya Lvovna Shcherbakova, Anton Pavlovich Tyukov, Timur Alexandrovich Janovsky, Valeriy Anatol'evich Kamaev, et al. A survey of forecast error measures. *World applied sciences journal*, 24(24):171–176, 2013.

[40] Zhang Shuai, Sun Shu-dong, Cai Zhi-qiang, and Dong Wei. Designed the servicing bom and integration based on xml. *Modern Manufacturing Engineering*, 1, 2008.

[41] Xiaogang Su, Xin Yan, and Chih-Ling Tsai. Linear regression. *Wiley Interdisciplinary Reviews: Computational Statistics*, 4(3):275–294, 2012.

[42] Ilya Sutskever. *Training recurrent neural networks*. University of Toronto Toronto, ON, Canada, 2013.

[43] Daniel Svozil, Vladimir Kvasnicka, and Jiri Pospichal. Introduction to multi-layer feed-forward neural networks. *Chemometrics and intelligent laboratory systems*, 39(1):43–62, 1997.

[44] Gaurav Vishwakarma, Aditya Sonpal, and Johannes Hachmann. Metrics for benchmarking and uncertainty quantification: Quality, applicability, and best practices for machine learning in chemistry. *Trends in Chemistry*, 3(2): 146–156, 2021.

[45] Renzhuo Wan, Shuping Mei, Jun Wang, Min Liu, and Fan Yang. Multivariate temporal convolutional network: A deep neural networks approach for multivariate time series forecasting. *Electronics*, 8(8):876, 2019.

[46] Daniel Westreich, Justin Lessler, and Michele Jonsson Funk. Propensity score estimation: neural networks, support vector machines, decision trees (cart), and meta-classifiers as alternatives to logistic regression. *Journal of clinical epidemiology*, 63(8):826–833, 2010.

[47] Xin Yan and Xiaogang Su. *Linear regression analysis: theory and computing*. World Scientific, 2009.

[48] Bayya Yegnanarayana. *Artificial neural networks*. PHI Learning Pvt. Ltd., 2009.

[49] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

[50] Bing Zhang, Jhen-Long Wu, and Pei-Chann Chang. A multiple time series-based recurrent neural network for short-term load forecasting. *Soft Computing*, 22(12):4099–4112, 2018.

[51] Wengang Zhang, Runhong Zhang, Chongzhi Wu, Anthony TC Goh, and Lin Wang. Assessment of basal heave stability for braced excavations in anisotropic clay using extreme gradient boosting and random forest regression. *Underground Space*, 2020.

Appendices

Appendix A

Time Series Analysis

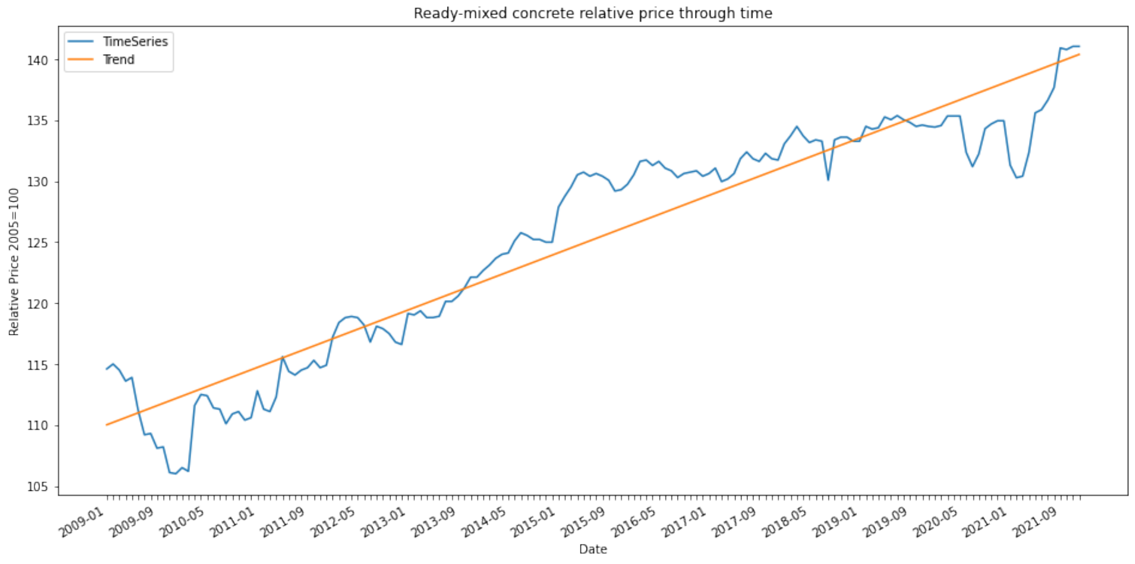

Figure A.1: Ready-mixed concrete timeseries and trend

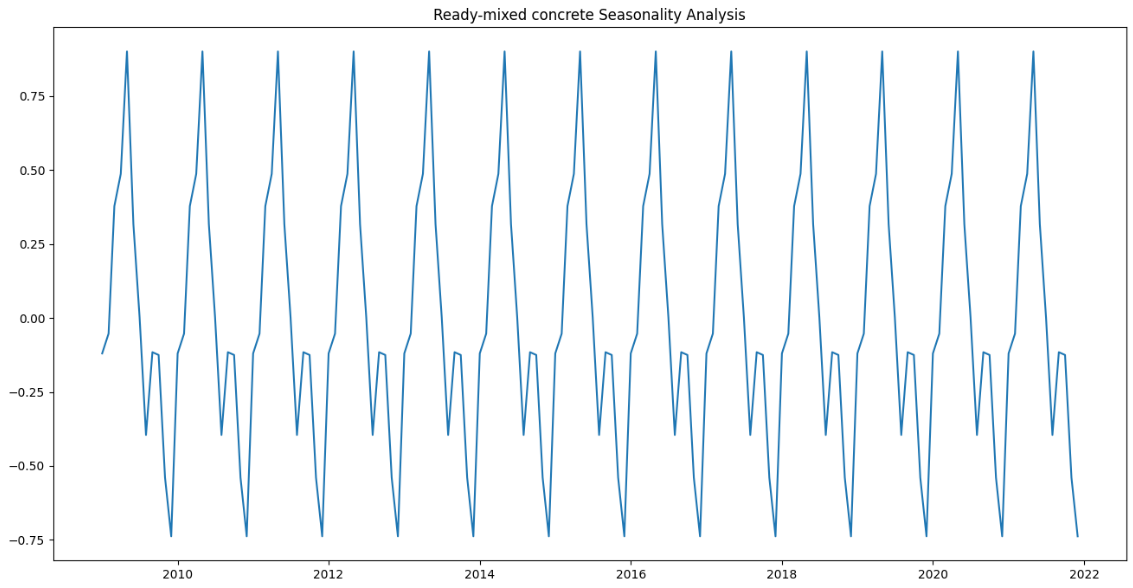

Figure A.2: Ready-mixed concrete seasonality

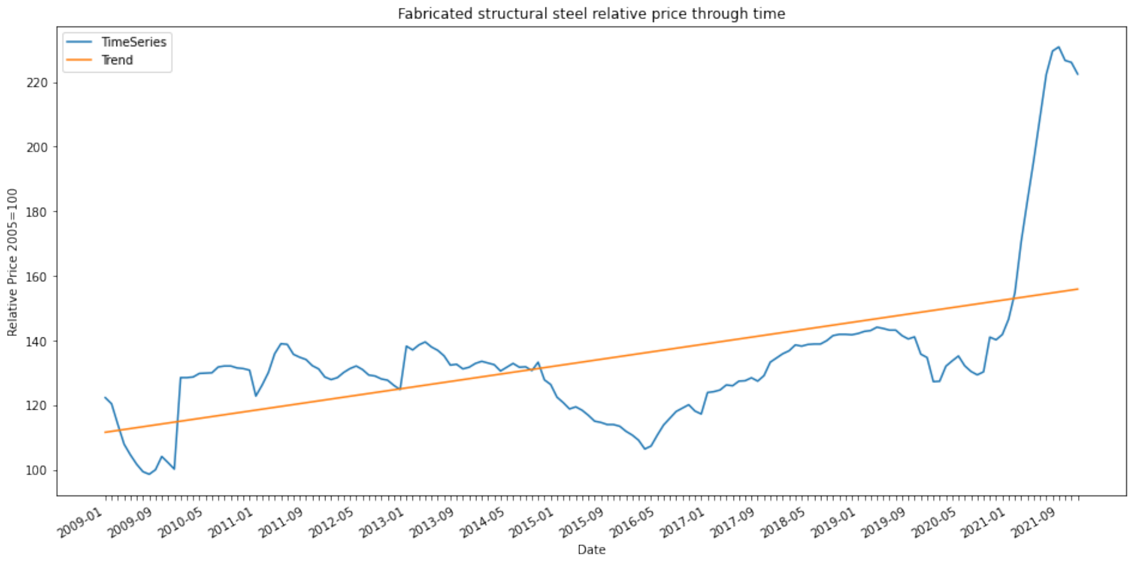

Figure A.3: Fabricated structural steel timeseries and trend

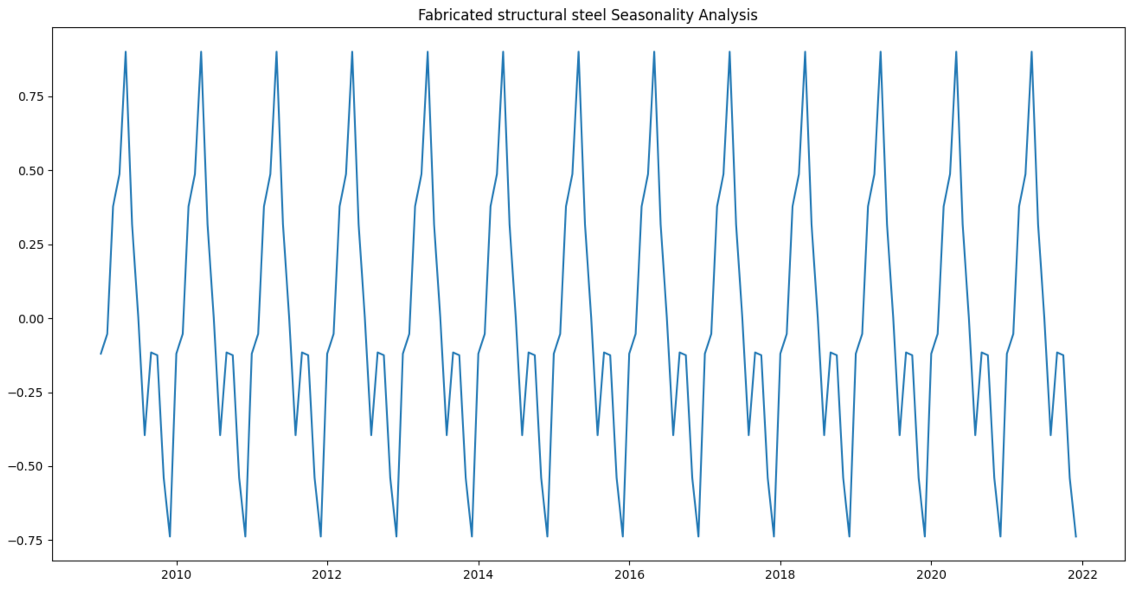

Figure A.4: Fabricated structural steel seasonality

Appendix B

Time Series Forecasting

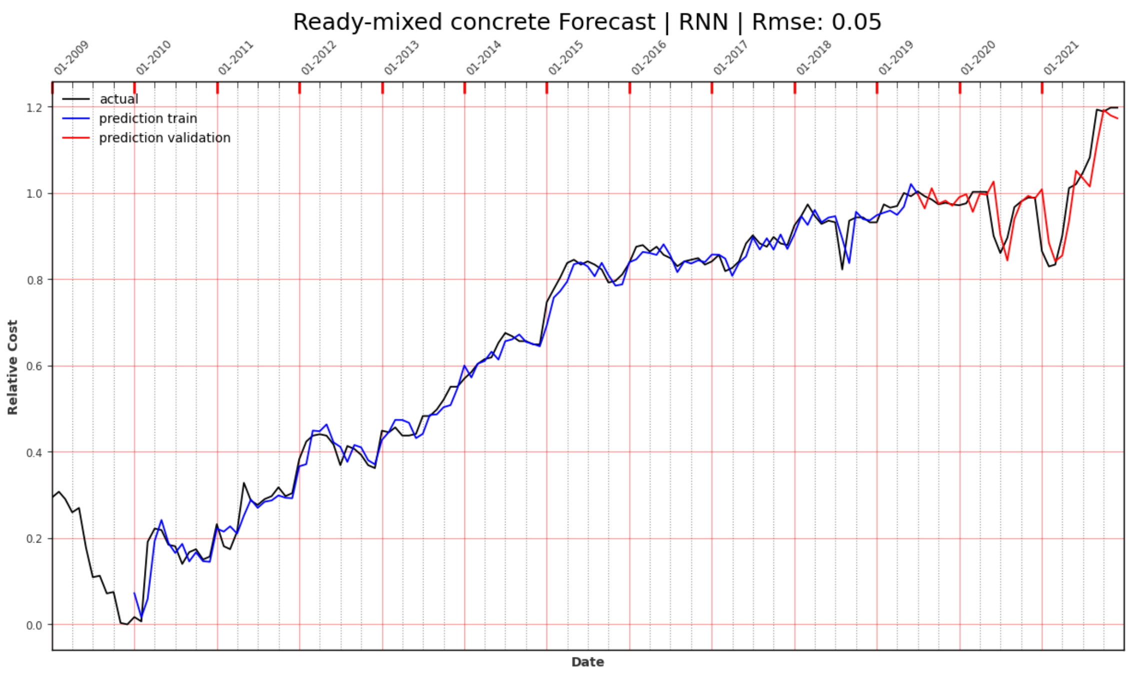

Figure B.1: Ready-mixed concrete GRU model forecast

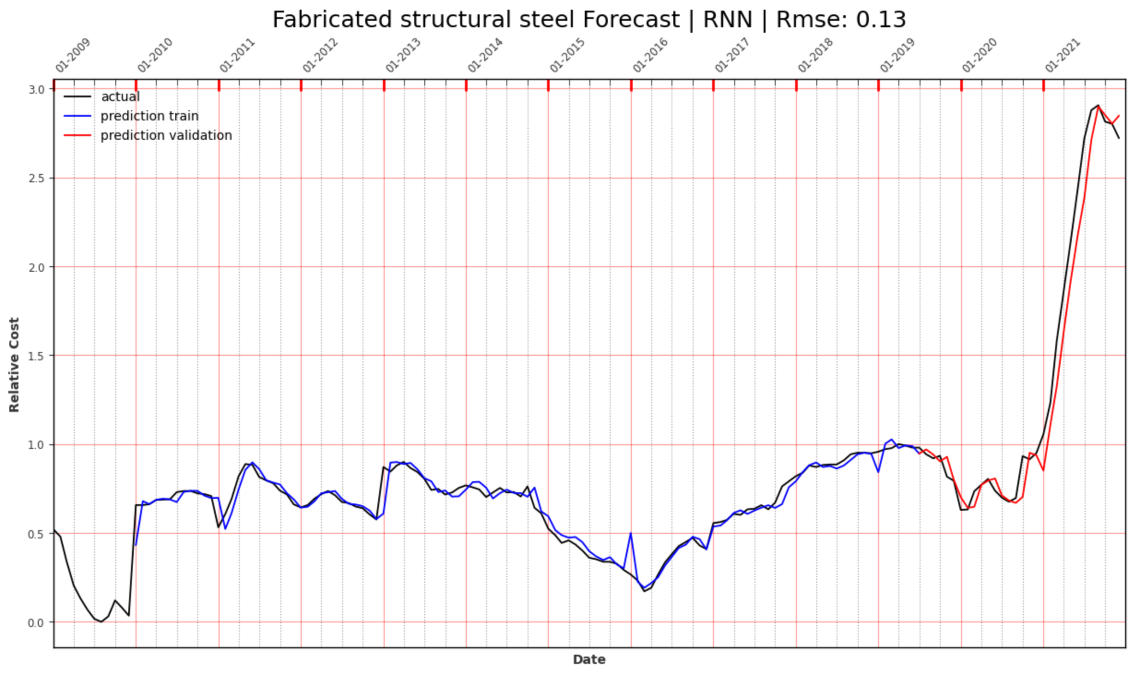

Figure B.2: Fabricated structural steel GRU model forecast

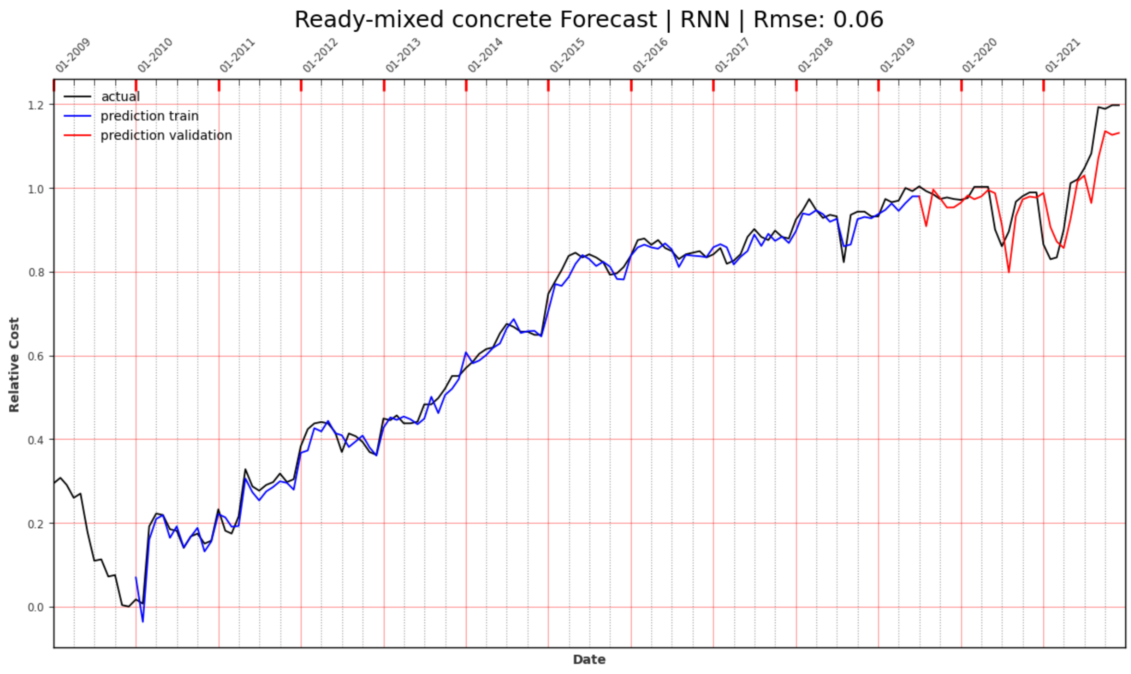

Figure B.3: Ready-mixed concrete LSTM model forecast

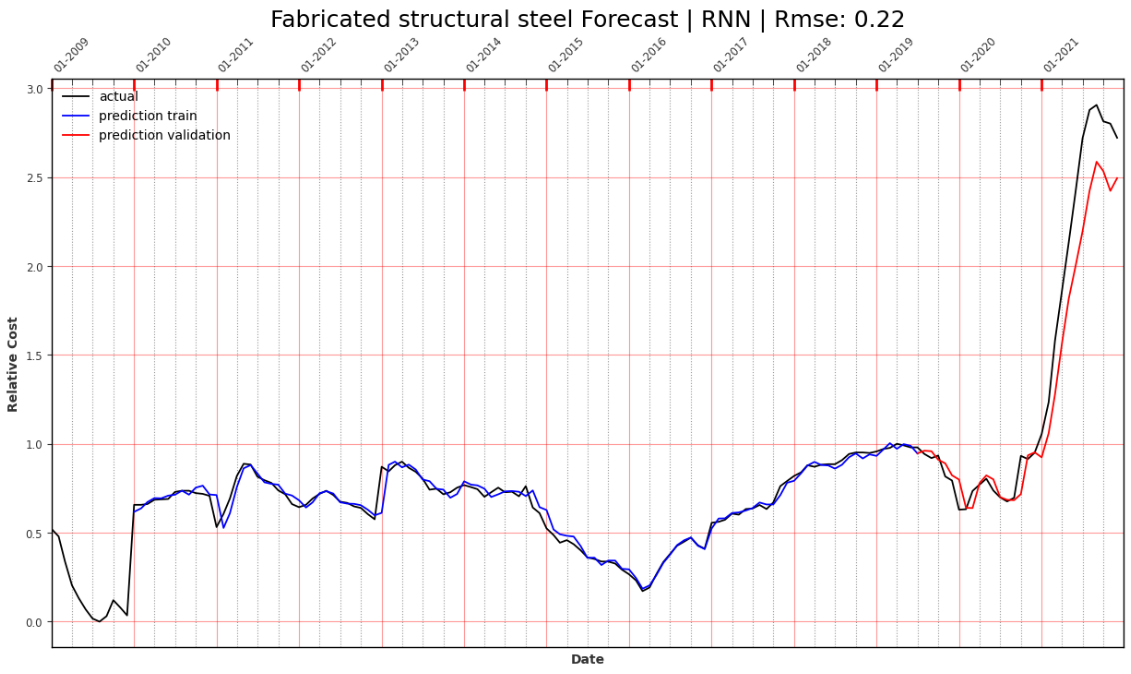

Figure B.4: Fabricated structural steel LSTM model forecast

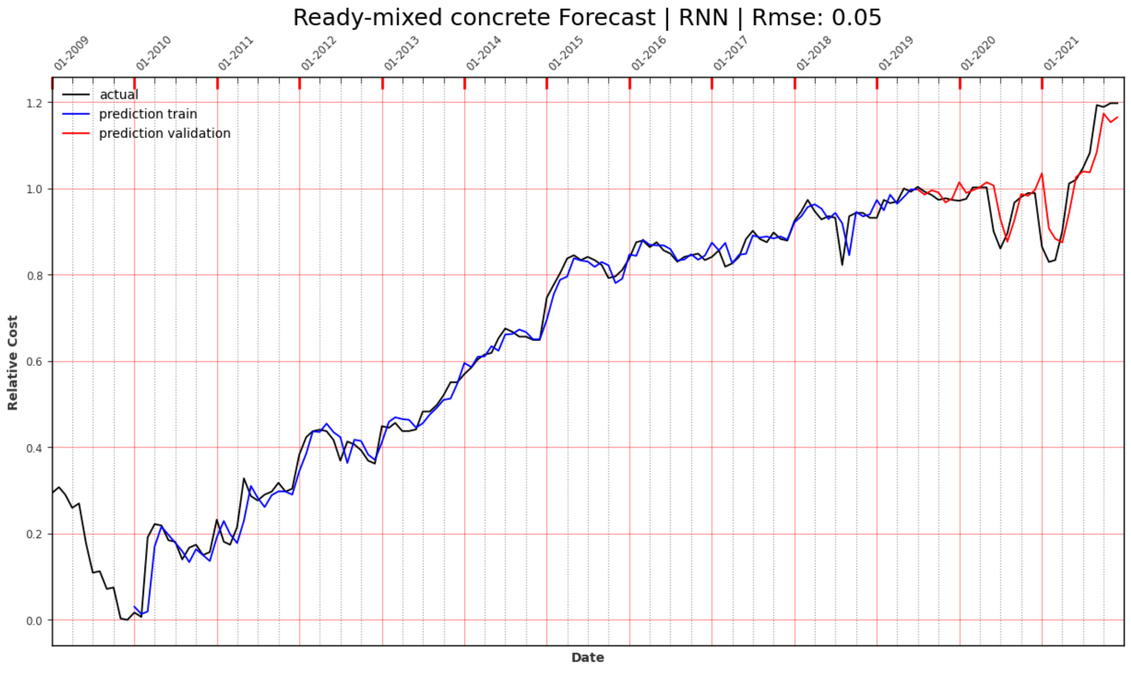

Figure B.5: Ready-mixed concrete Vanilla RNN model forecast

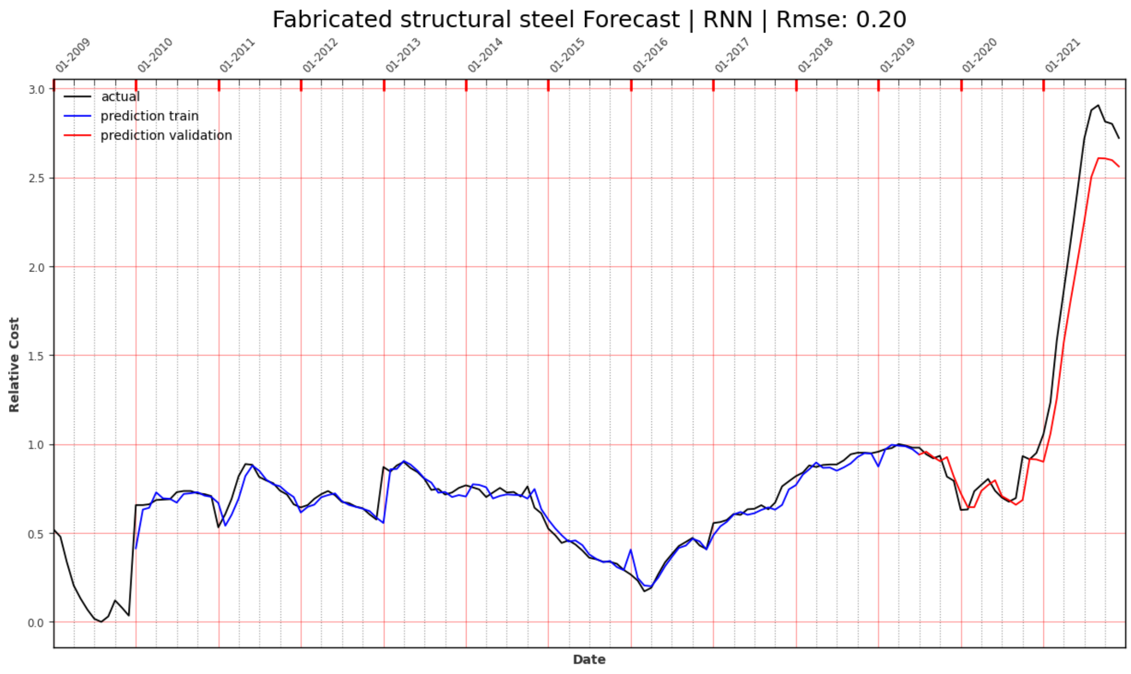

Figure B.6: Fabricated structural steel Vanilla RNN model forecast

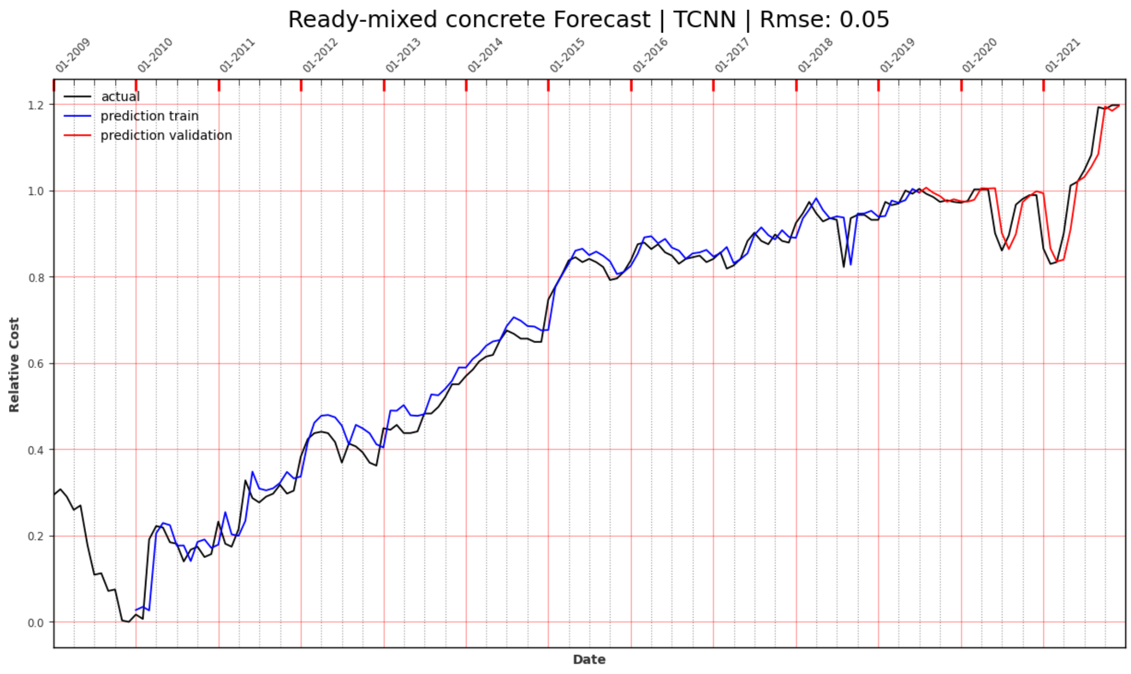

Figure B.7: Ready-mixed concrete TCNN model forecast

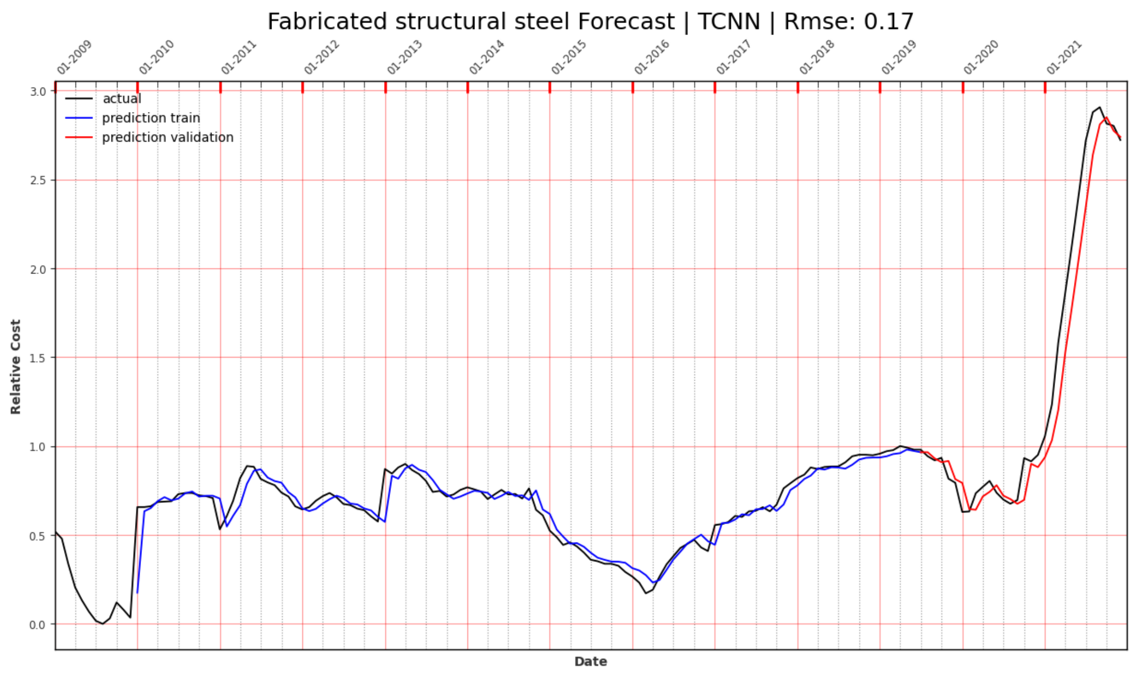

Figure B.8: Fabricated structural steel TCNN model forecast