FEDERAL UNIVERSITY OF SANTA CATARINA
TECHNOLOGY CENTER
DEPARTMENT OF AUTOMATION AND SYSTEMS ENGINEERING
UNDERGRADUATE PROGRAM IN CONTROL AND AUTOMATION ENGINEERING

Romero Perardt Magalhães Brito

Machine Learning Approach to Forecast Sparse Tire Test Demand

Florianópolis
2026

Romero Perardt Magalhães Brito

Machine Learning Approach to Forecast Sparse Tire Test Demand

Final report of the subject DAS5511 (Course Final Project) as a Concluding Dissertation of the Undergraduate Program in Control and Automation Engineering of the Federal University of Santa Catarina.
Supervisor: Prof. Eduardo Camponogara, Dr.
Co-supervisor: Gauthier Delorme, Eng.

Florianópolis
2026

Ficha catalográfica gerada por meio de sistema automatizado gerenciado pela BU/UFSC.
Dados inseridos pelo próprio autor.

Perardt Magalhães Brito, Romero
Machine Learning Approach to Forecast Sparse Tire Test Demand / Romero Perardt Magalhães Brito ; orientador, Eduardo Camponogara, coorientador, Gauthier Delorme, 2026.
46 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia de Controle e Automação, Florianópolis, 2026.

Inclui referências.

1. Engenharia de Controle e Automação. 2. Demand Forecasting. 3. Zero-inflated Regression. 4. Imbalanced Classification. 5. Hurdle Models. I. Camponogara, Eduardo. II. Delorme, Gauthier. III. Universidade Federal de Santa Catarina. Graduação em Engenharia de Controle e Automação. IV. Título.

Romero Perardt Magalhães Brito

Machine Learning Approach to Forecast Sparse Tire Test Demand

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

Florianópolis, February 8, 2026.

Prof. Marcelo De Lellis Costa de Oliveira, Dr.
Course Coordinator

Examining Board:

Prof. Eduardo Camponogara, Dr.
Advisor
UFSC/CTC/EAS

Gauthier Delorme, Eng.
Supervisor
Michelin

João Sérgio Silva Santos
Evaluator
UFSC/CTC/EAS

Prof. Edson Roberto De Pieri, Dr.
Board President
UFSC/CTC/EAS

ACKNOWLEDGEMENTS

First, I would like to thank my parents for encouraging me to follow the academic path that brought me to places even they had not imagined.

I would also like to thank the Federal University of Santa Catarina for opening to me a world of knowledge and possibilities. I am equally grateful to CentraleSupélec, the french engineering school that hosted me during the two years of my double-degree program, where I had the opportunity to meet extraordinary people and achieve things I had never imagined myself capable of.

I would like to thank Michelin for offering me a six-month internship that led to this work and for providing the time and resources to explore such an interesting subject. In particular, I thank Gauthier Delorme and his team for their warm welcome and invaluable guidance throughout the project.

Finally, I would like to thank my academic supervisor, Prof. Eduardo Camponogara, for helping me navigate a complex and challenging problem and for his support in shaping this work into a coherent whole.

DISCLAIMER

Clermont-Ferrand, February 8th, 2026.

As representative of the Michelin Research and Development Center in which the present work was carried out, I declare this document to be exempt from any confidential or sensitive content regarding intellectual property, that may keep it from being published by the Federal University of Santa Catarina (UFSC) to the general public, including its online availability in the Institutional Repository of the University Library (BU). Furthermore, I attest knowledge of the obligation by the author, as a student of UFSC, to deposit this document in the said Institutional Repository, for being it a Final Program Dissertation (*"Trabalho de Conclusão de Curso"*), in accordance with the *Resolução Normativa n° 126/2019/CUn*.

Gauthier Delorme, Eng.
Michelin

ABSTRACT

In the development of new tires, the testing phases play a key role, as they assess whether the proposed tire meets the required performance standards. At Michelin, these tests are carried out by an internal entity that must know, well in advance, the monthly demand for each test resource — namely, the workload for each group of test machines, in order to ensure sufficient capacity. However, the forecasting method currently employed is not able to capture the true demand, preventing the company from achieving an equilibrium between demand and capacity. As a result, backorders arise in the workshops, causing delays in ongoing projects. Therefore, a new forecasting method should be considered. Since test demand can be structured into hierarchical levels, a bottom-up approach is proposed in which the demand for each test is predicted at the project level using Machine Learning (ML) techniques in order to fully leverage the large amount of available data. At this level, however, demand is sparse: each project uses only a small subset of the available test resources and only during specific months. Consequently, demand prediction becomes a zero-inflated regression problem. To address this issue, a composite modeling approach is proposed. First, a classifier acts as a hurdle by determining whether a project will request a given test resource. If so, a subsequent regression model predicts the associated workload. The classification task is highly imbalanced, as projects request test resources only a small fraction of the time. This two-stage logic is applied both to the full forecast horizon and to each individual month within the horizon. Nevertheless, only the classifier associated with the full forecast horizon, designed with tree-based models from the Scikit-learn Python library, is implemented and evaluated. This approach aims to serve as the first step to a more accurate demand forecast, enabling the company to develop a strategic view of testing requirements, reduce backorders, and avoid delays in ongoing projects.

Keywords: Demand Forecasting. Hierarchical Forecasting. Zero-inflated Regression. Imbalanced Classification. Hurdle Models.

RESUMO

No desenvolvimento de novos pneus, as fases de teste desempenham um papel fundamental, pois avaliam se o pneu proposto atende aos requisitos de desempenho exigidos. Na Michelin, esses testes são realizados por uma entidade interna que precisa conhecer, com bastante antecedência, a demanda mensal de cada recurso de teste — isto é, a carga de trabalho de cada grupo de máquinas de ensaio — a fim de garantir capacidade suficiente. No entanto, o método de previsão atualmente utilizado não consegue capturar a demanda real, impedindo a empresa de alcançar um equilíbrio entre demanda e capacidade. Como consequência, surgem pedidos em atraso nas oficinas, causando atrasos em projetos em andamento. Portanto, um novo método de previsão deve ser considerado. Como a demanda por testes pode ser estruturada em níveis hierárquicos, propõe-se uma abordagem *bottom-up*, na qual a demanda de cada teste é prevista no nível de projeto utilizando técnicas de Aprendizado de Máquina (ML, do inglês *Machine Learning*), de modo a explorar plenamente a grande quantidade de dados disponíveis. Nesse nível, entretanto, a demanda é esparsa: cada projeto utiliza apenas um pequeno subconjunto dos recursos de teste disponíveis e somente em meses específicos. Consequentemente, a previsão da demanda torna-se um problema de regressão com inflação de zeros (*zero-inflated regression*). Para tratar essa questão, é proposta uma abordagem de modelagem composta. Primeiramente, um classificador atua como um filtro, determinando se um projeto solicitará um determinado recurso de teste. Em caso afirmativo, um modelo de regressão subsequente prevê a carga de trabalho associada. A tarefa de classificação é altamente desbalanceada, uma vez que os projetos solicitam recursos de teste apenas em uma pequena fração do tempo. Essa lógica em dois estágios é aplicada tanto ao horizonte completo de previsão quanto a cada mês individual dentro desse horizonte. No entanto, apenas o classificador associado ao horizonte completo de previsão, desenvolvido com modelos baseados em árvores da biblioteca Scikit-learn em Python, é implementado e avaliado. Essa abordagem tem como objetivo servir como o primeiro passo rumo a uma previsão de demanda mais precisa, permitindo que a empresa desenvolva uma visão estratégica das necessidades de teste, reduza os pedidos em atraso e evite atrasos em projetos em andamento.

Palavras-chave: Previsão de Demanda. Classificação Desbalanceada. Regressão com Inflação de Zeros.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

AP	Average Precision
AUC	Area Under the Curve
GST	Specialized Testing Group
ML	Machine Learning
OE	Original Equipment
OEM	Original Equipment Manufacturer
PCA	Principal Component Analysis
RD	Research and Development
ROC	Receiver Operating Characteristic
RT	Retail
SC	Supply Chain
SOP	Sales and Operations Planning

CONTENTS

1 INTRODUCTION

At first sight, a tire might seem like a simple object; nevertheless, it represents a remarkable technological advance, as it is a highly complex system with many different layers. The first tires appeared in the late nineteenth century, and since then they have not stopped evolving. In order to keep improving tire performance, companies such as Michelin, Bridgestone, and Goodyear invest heavily in the research and development of new tires.

Michelin is a French multinational tire manufacturer founded in 1889 by the brothers Édouard and André Michelin in Clermont-Ferrand. The company has played a key role in the history of tire development, as it made major advances in tire design, for instance by inventing the first removable tires in 1891 and the radial tire in 1946, a technology that has more than doubled tire lifetime (Michelin, 2026a). Furthermore, Michelin continues to innovate, for example with Uptis, its airless tire (Michelin, 2026b). The company manufactures tires for passenger cars, motorcycles, heavy-duty machinery, aircraft, and even space shuttles.

Today, Michelin's core of innovation is its Research and Development (RD) center in Ladoux, where more than 3,500 engineers and researchers work on developing new materials, designs, and methods (Campo, 2025). The main activity of the Michelin RD Center is the development of new and more efficient tires. Due to the many different types of tires (car, agriculture, industrial, motorcycle, etc.) and the various ways to increase tire performance (applying new materials, layers, tread patterns, etc.), many projects are being carried out at Michelin. It is also in this center that the present work was conducted, more precisely in the Supply Chain team responsible for developing tools to facilitate the center's internal processes, such as managing requests to build tire prototypes and billing completed tests.

The projects are structured through development loops. After finishing a new tire design, prototypes are made and tested. Then, if the performance satisfies quality and safety requirements, it may proceed to an industrial ramp-up and go to market; if not, modifications to the original design are made and new tests performed. There are many different types of tests, for instance to measure noise, maximum speed limit, rolling resistance, braking energy, etc. The testing phase is necessary both to assess the performance of the tire and to certify it so that it can be sold on the market. Furthermore, the tests can be divided into objective tests, which means that direct measurements are taken, and subjective tests, where pilots test the tire on a vehicle and evaluate it based on their experience.

At Michelin, the tires are tested by an internal organization called Specialized Testing Group (GST), whose main facilities are located on the same site as the RD offices. For this work, the focus is on the objective tests conducted in GST's workshops.

The workshop is organized in such a way that the use of its resources is optimized; therefore, to perform a test, it is necessary to know about it in advance, so that GST can be ready by preparing the machines and allocating employees. That is why it is important to have a test demand forecast, which can be made for an operational (a few weeks) or tactical time horizon (in this case, 18 months). The latter will be the focus of this work. The 18-month horizon forecast allows the workshop, for instance, to hire and train new employees if the future demand for a specific test is too high.

Every project is planned using specific software called Kephren, where all the resources, including the tests, that are needed for its completion are listed and detailed manually by a planner. Then, the software can forecast resource demand. However, the resulting test demand forecast is not reliable enough, as it fails to capture the true demand. This is mostly due to a few reasons: the fact that the real demand for the tests is recorded in different software, which is not connected to Kephren; that the standard load for each test used by Kephren to compute the forecasts is not accurate; etc.

1.1 OBJECTIVES

The goal of this project is then to develop a new way of predicting test demand, so that prediction bias is reduced, and the need to manually list all the tests required by each project is eliminated. This should be achieved using Artificial Intelligence and Machine Learning (ML) techniques, since there is an increasing amount of available data related to the subject. Thus, this can be broken down into specific objectives as follows:

- Identify the key factors that impact test demand for the projects;
- Propose a methodology to predict test demand using a Machine Learning approach;
- Develop a proof of concept to validate the chosen approach within a smaller perimeter, i.e., a few of the test resources;
- Elaborate a roadmap with the next steps so that the model can be generalized to all other tests and further deployed in production.

By achieving this, the project aims to provide a reliable forecast that will benefit the enterprise by:

- Allowing GST to better optimize the organization of its workshops, thereby gaining flexibility and reducing costs;
- Reducing delays in the progress of ongoing projects due to back orders at GST;

- Facilitating the demand review phase in the Sales and Operations Planning (SOP) process.

Given that there are many tests available, each project may choose to perform a different subset of tests. The final goal of this work is to predict, for each project, the monthly load that will be used from each test over the forecast horizon. Since this is a fairly complex problem, it can be broken down into sub-problems:

- The first is to predict which test resources each project will use;
- The second is to determine, for each test, how much of it the project will use, meaning the total test load for the whole time period;
- The last sub-problem is to determine how the predicted total load will be distributed across the months when the test will be needed.

DOCUMENT STRUCTURE

The present work is organized as follows: the second chapter aims to provide the theoretical background needed to understand the problem by presenting an overview of the classical steps to solve a machine learning problem, as well as an explanation of tree-based models and zero inflation; the third chapter presents the problem in detail, revealing the hierarchical nature of the forecast as well as characterizing it as a zero-inflated regression problem at the bottom level; the fourth chapter presents the proposed composite model architecture; the fifth chapter details the implementation of the first component of the composite model, a classifier that decides whether a project will request a test resource in the following months; finally, the last chapter presents the conclusion and further work.

2 THEORETICAL BACKGROUND

The first section of this chapter presents an overview of the steps commonly taken when solving a machine learning problem, from model development to production launch. Section 2.2 dives into tree-based models, namely Decision Trees and Random Forests, a family of ML algorithms that has presented the best results in solving the present problem. Finally, the last section shows the most commonly used strategies to deal with zero inflation in regression tasks.

2.1 MACHINE LEARNING MODEL BUILDING PROCESS

> Nearly all deep learning algorithms can be described as particular instances of a fairly simple recipe: combine a specification of a dataset, a cost function, an optimization procedure and a model.(Goodfellow; Bengio; Courville, 2016, p. 152).

In order to build an ML model, it is essential to have data from which the model can learn. Therefore, the first step is to define which information is relevant to the problem and then collect it from possibly different data sources. For example, to predict test demand from different projects, the history of conducted tests and project information should be gathered. In most cases, the data is gathered in tabular form, in which the rows represent the examples and the columns the attributes.

After this gathering phase, the raw data should pass through a cleaning process, where problems such as missing values, duplicates, and inconsistencies are addressed. Afterwards, comes the feature engineering process, in which new features are built from the data. For instance, in time series data it may be interesting to create lagged features or a new column that indicates the presence of special dates such as holidays. Therefore, it is a step where the ingenuity and creativity of the ML engineer shine.

Then, the data should be transformed. Notably, columns that contain categorical data, such as the type of tire (for passenger cars, motorcycles, agriculture, etc.), should be encoded. In order to encode them, there are three common methods:

- The first is label encoding, where a number is assigned to each category. However, it may lead to a false sense of order;
- The second is one-hot encoding, where for each category a binary column is added indicating whether the example belongs to it or not.
- The third is target encoding, where each category is replaced by the mean value of the target variable to be predicted. Notably, this should be done carefully to avoid data leakage.

In addition to encoding, most ML algorithms require the data to be normalized or standardized. However, this is not the case for tree-based algorithms, which will be the focus of the following section.

Finally, for supervised learning tasks, in order to choose the best model, the data should be divided into three different subsets:

- Training set, from which the models learn. Normally, it is the one that contains the largest part of the examples;
- Validation set, which is used to evaluate each model's ability to generalize to unseen data so that the best model can be chosen;
- Test set, which is used only after the final model is built, so that an unbiased measure of prediction quality can be obtained.

When an ML model is said to "learn," it is in fact minimizing a cost function. For classification tasks, the cost function is typically the negative log-likelihood, whereas for regression tasks it is often the Mean Squared Error (MSE). The optimization process is usually performed using gradient-descent-based algorithms, since in most cases a closed-form solution does not exist. Regarding models, there is a myriad of different ones. The next section will focus on a special family of them called tree-based models.

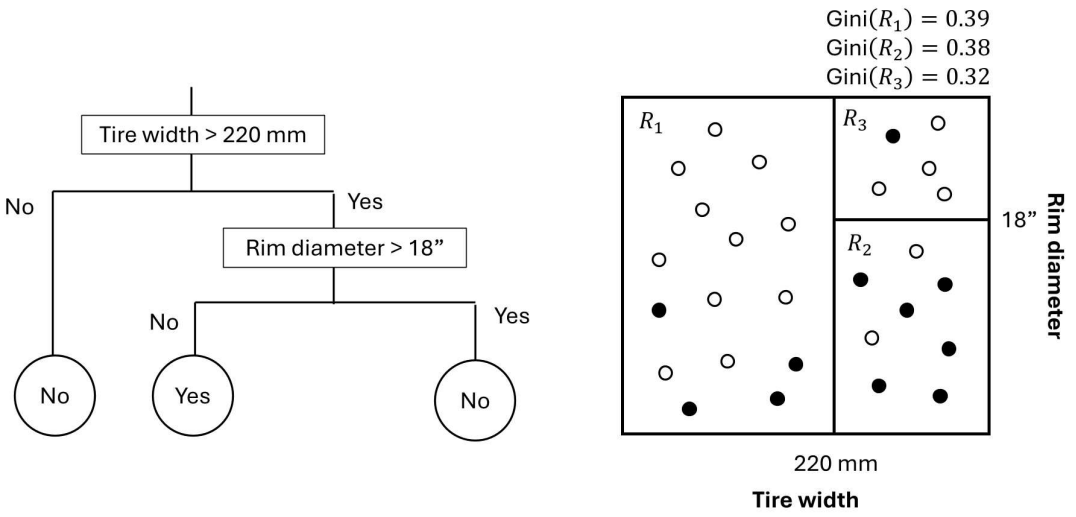
2.2 TREE-BASED MODELS

The decision tree is a non-parametric algorithm, meaning that it does not have a fixed number of parameters before fitting the model, as a linear regression model does, which implies that its size and complexity are proportional to the size of the training data (Goodfellow; Bengio; Courville, 2016, p. 115). Another point that differentiates decision trees from most other ML algorithms is that their results are easy to explain and interpret, as the steps taken by the tree can be visualized in a simple diagram, which can be very beneficial when sharing the model with business partners. In particular, Figure 1 shows a simple example of such a diagram, where a decision tree that predicts whether a particular tire will be tested or not by partitioning the feature space — composed of the rim diameter and the tire width — into different regions is depicted. Furthermore, decision trees can be used both for classification and regression tasks, and when a set of trees is combined using boosting or bagging techniques, they can provide even better predictions. For these reasons, decision trees are very popular among ML engineers.

2.2.1 Decision Trees

Decision trees work by partitioning the training examples into different regions in a clever way. Thus, when a new example arrives, it will fall into one of those regions,

Figure 1 – Fictitious example of a simple decision tree classifier



Source: Adapted from Gatnar (2002).

also known as leaf nodes. Then, for a regression task, the prediction will be the mean value of the target in that region, and for a classification task, it will be the most frequent class. Mathematically, the tree is represented as

$$\hat{y} = \sum_{k=1}^{K} a_k \mathbf{1}_{R_k}(x)$$

where R_k represents k-th region among the K different regions, $\mathbf{1}_{R_k}(\cdot)$ the indicator function that outputs 1 if the input belongs to region R_k or 0 if not, $\hat{y}$ the target estimation given by the tree, x the input example. Furthermore, for regression tasks $a_k = \frac{1}{n_k} \sum_{i=1}^{n_k} y_i^k$, with n_k being the number of training examples belonging to region R_k and y_i^k the target value of the i-th sample in region k; and for classification $a_k = \arg\max_j p(j|R_k)$ with j ranging over all possible classes and $p(j|R_k)$ the proportion of the class j inside the region R_k (Gatnar, 2002).

The regions are iteratively defined by splitting the previous regions into smaller ones so that their resulting impurity decreases. For classification tasks, the most commonly used measure of a region's impurity is the Gini index, which estimates the probability of miscategorizing an example that falls into that particular region. On the other hand, for regression, the impurity measure is the variance of the target variable in that region. The splits are made by looking at one attribute at a time, and, in most implementations, this divides the previous region into two. Therefore, when building the tree, the algorithm searches for the attribute that produces the split that most decreases the impurity. The Gini index is given by the following equation

$$\text{Gini}(R_k) = \sum_j p(j|R_k)(1 - p(j|R_k)).$$

For binary classification, the gini index equals $\text{Gini}(R_k) = 2p_k(1 - p_k)$, where p_k is the proportion of the first class examples in region R_k. Thus, the impurity is zero when the region's examples are all from the same class and highest when the number of examples is the same for both classes, since the region fails completely to differentiate the classes.

If the tree is allowed to grow indefinitely by splitting the regions into smaller and smaller ones, the impurity of each region may eventually reach zero. In the extreme case, each region will contain only one example, and therefore it will be perfectly fitted to the training data. However, this tree will most certainly be a poor predictor, unable to generalize its conclusions to unseen examples. This tendency to overfit the training data, together with the fact that small changes in the choice of training examples may lead to a quite different tree, makes up the weakness of decision trees. Nevertheless, there are strategies available to mitigate these effects. For the first, pruning techniques can be applied, which limit the tree-growing process. For the latter, a committee of different trees can be created using ensemble learning (Gatnar, 2002), which is detailed in the following section.

2.2.2 Random Forests

Instead of using the prediction from a single tree, one can take into account the majority vote (for classification tasks) or the average (for regression) of the predictions of an ensemble of different trees. The advantage of this approach is that, if the errors of the individual trees are independent, the probability that all the trees give an erroneous prediction decreases. However, in most cases

> ...the assumption of independence is unreasonable, because hypotheses are likely to be misled in the same way by any misleading aspects of the training data. But if the hypotheses are at least a little bit different, thereby reducing the correlation between their errors, then ensemble learning can be very useful (Russell; Norvig, 2010).

In order to build a set of different trees, one could train different models using different training sets generated from the same underlying distribution. However, the available data is usually limited; instead, one can try to artificially create different training sets from the original one. For that, there are two main techniques: the first is to take bootstrap samples of the original training set, meaning taking samples with replacement from the original set; the second is to consider only a subset of the available features when building each individual tree (Scikit-learn, 2026). By doing so, one can take advantage of the fact that changes in the training set might produce quite different trees,

as discussed in the previous section. Finally, the resulting estimator is called a Random Forest.

2.3 ZERO-INFLATION

Before diving into this work's problem, it is beneficial to analyze a similar one. For that reason, an example inspired by the work of Lewbel and Nesheim (2019) is studied. In their work, they analyzed the following example:

> ...we consider consumer demand for fresh fruit in the UK. In a typical store, there are more than two dozen types of fruit that consumers can choose among. Consumers typically choose from one to five different types of fruit to purchase, and buy varying quantities of each type... The types and quantities of fruits purchased vary greatly across households. While many different types of fruit are offered for sale, typical households only buy a small number of types. As a result, most consumers buy zero quantities of most categories of fruit, and therefore the vector of observed demands at the individual consumer level is sparse (Lewbel; Nesheim, 2019).

In such cases, classical regression approaches are not recommended, since they tend to overestimate the number of zeros and underestimate the magnitude of the target variable when it is non-zero (Abraham; Tan, 2009). Thus, other approaches should be considered. Traditionally, there are two kinds of models that deal with zero inflation, both known as zero-truncated models. The first is the Tobit model, which assumes that there is one underlying latent variable y^* alongside the observed one $y = \max(0, y^*)$. The second is the hurdle model, which assumes that there are two distinct ongoing processes: one that decides whether the target variable is zero and the other that decides its magnitude if it is non-zero. Likewise, a similar approach is described by Rožanec *et al.* (2025), using ML models, in which a first classifier decides whether the target variable is non-zero or not, and a second regressor predicts the value if the latter is true.

Furthermore, it could also be interesting to forecast the demand for different fruits for the entire country, or for each of its regions. Notably, at the country level, demand will most probably be non-zero for the whole year. Thus, the forecast will not suffer from zero inflation. Therefore, to predict the future values of the variable at different aggregation levels, hierarchical forecasting can be considered. In particular, predicting fruit demand for the following year, quarters, and months is also a type of hierarchical forecasting (Athanasopoulos *et al.*, 2024).

3 PROBLEM STATEMENT

The new Michelin tires are designed in its RD centers across the world, such as the ones in Campo Grande (Brazil), Greenville (United States of America), Ota (Japan), and Ladoux (France). Each center holds different projects, which are grouped into different entities, so that each entity develops a type of tire.

The new tires can be grouped into many categories, based on:

- The season: summer, winter, and all-season tires;
- The usage: bicycle tires, passenger cars, light trucks, trucks, agricultural, earth movers, etc.;
- The final customer: Original Equipment (OE) tires, which are directly sold to the Original Equipment Manufacturer (OEM) to be installed on brand-new vehicles, and Retail (RT) tires, which are sold in conventional stores;
- The maturation: concept tires, balise tires, and market tires.

The process of designing these tires might differ slightly, especially from a concept to a market tire. Nevertheless, they all share a common need, namely to make and test prototypes.

The tests are performed internally by GST, which has facilities that are usually located close to the RD centers, such as the one in Ladoux, so that new tires can be quickly tested. Nevertheless, a GST site can also receive tires from a distant RD center when it does not have a particular machine. In addition, there are some special cases, such as Almería in Spain, where large earth mover tires are tested, and Ivalo in Finland for winter tires.

There is a wide variety of tests, which can be broken into two categories: tests on machines and on-vehicle tests. The first typically measures performance indicators such as rolling resistance, speed limit, braking energy, etc. The latter usually relies on the driver's ability to assess qualitative aspects of tire performance on both dry and wet roads. Figure 2a shows one of Ladoux's circuits for tire testing, and Figure 2b shows a joint test conducted with Alpine for an OE tire.

The testing phase is crucial, not only because it is mandatory in order to sell tires, but also because it allows the tire designer to validate ideas and assess the true performance of prototypes. For that reason, most tests are conducted within an iterative loop of (re)designing and testing tires until the required performance is achieved. Therefore, if GST fails to deliver a test on time, the project might be delayed as well, which is extremely undesirable, since it might delay the delivery of the tire to the market or to the OEM client.

(a) Circuit in Ladoux. Source: La Montagne (2017).

(b) On-vehicle testing. Source: Automotive Testing Technology International (2025).

Figure 2 – Michelin Test Center at Ladoux.

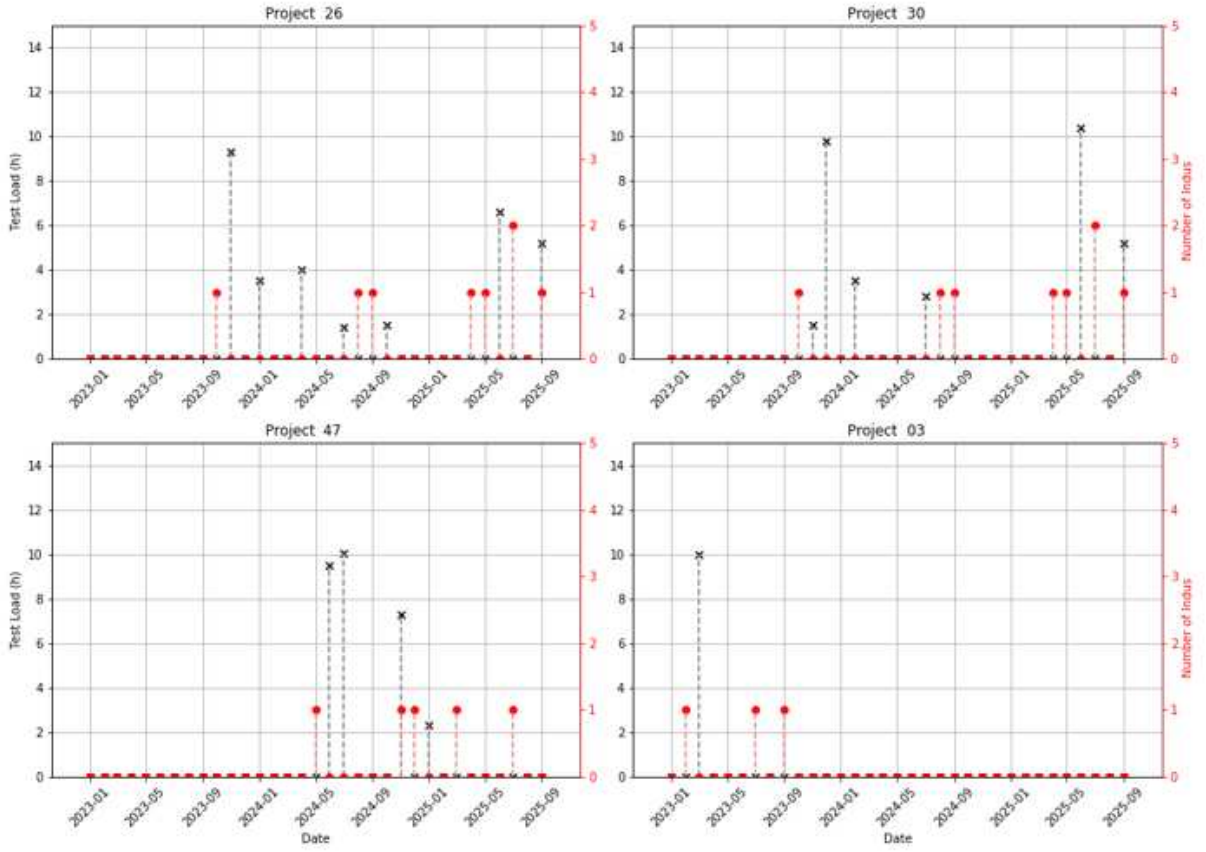
In order to avoid such delays, it is essential to know test demand well in advance, ideally 18 months ahead, so that GST can organize its workshops to ensure sufficient capacity, for instance by buying new machines and hiring new employees. Furthermore, in extreme cases where demand is higher than GST's maximum capacity, a trade-off should be reached so that projects with the highest demand are identified and asked to smooth their demand over the following months, ensuring that only a minimum number of projects are affected.

Notably, one should distinguish this particular need for a medium-term forecast from a short-term one, since both are needed. The latter is necessary for GST to plan its activities, namely allocating employees to the appropriate machines and scheduling their work. However, they differ, since the first relates to a tactical view while the latter relates to an operational one. Needless to say, the challenges involved in a long-horizon forecast are much greater.

Furthermore, for a particular test resource, the demand can be broken into different hierarchical levels: project, entity, and global level. The first level, at the bottom of the hierarchy, is the project level, since tests are primarily required by individual projects. At this level, the demand is highly sparse, as can be seen in Figure 3. This is due to the inherent nature of projects, since they normally request a test only after building a prototype, which will be referred to as an "indus" event (indicated in red in Figure 3), so that performance can be tested. Notably, having a good forecast at this level allows identification of which projects demand the most for a specific resource.

The second level is the entity level. Different projects are grouped into entities based on their similarities, for example, projects that design airplane tires. At this level, the demand corresponds to the sum of the demand from the individual projects. Figure 4b shows the demand for a test resource from a particular entity, decomposed into the demand from its projects, since each color represents a different project. Having a good forecast at this level is of great importance for the SOP demand review cycle, since, if for a certain month the demand is higher than the workshop's capacity, the SOP actors

Figure 3 – Examples of the project's demand for a specific test resource and, in red, the prototype buildings or "indus" events associate with it



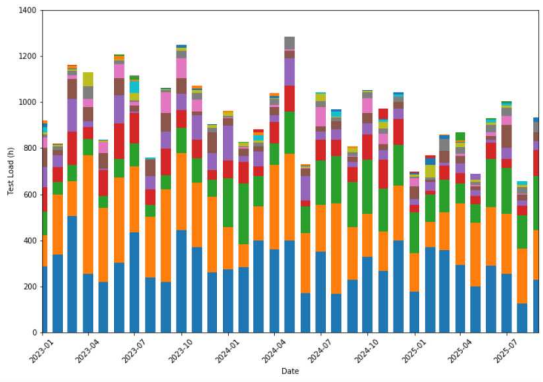
Source: Author.

can identify which entities they must act on in order to decrease their demand and avoid delays. When looking at how much each entity contributes to a test's global demand, one can see that it varies according to the test. Indeed, this is natural, since the entities are created as a function of the type of tires they build; therefore, a test dedicated to truck tires will have more demand coming from the entity that builds truck tires than from the one that builds sports car tires.

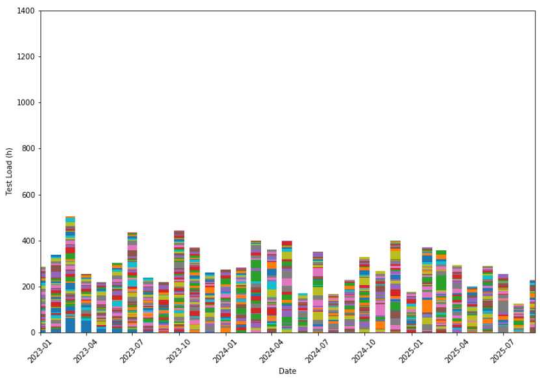
The last one, at the top of the hierarchy, is the global level, which is the sum of the demand coming from all entities. It is essential to have a good forecast at this level, since if the signal given here is false, the forecasts for the other levels become useless. Figure 4a shows the global demand for a specific test and how it can be decomposed into the different entities' demands, where each color represents a different entity. Although, in this work, we present these three levels, there are many other groupings possible; for instance, the entities are also grouped into macro-entities.

Naturally, there are two main approaches to forecast this demand, which are illustrated in Figure 5. The first is a bottom-up approach, which means predicting the

Figure 4 – Monthly demand for a single test resource.



(a) Global demand made of the different entities demand.



(b) Demand of a single entity made of the different project demands.

Source: Author.

demand for each project and then summing it to obtain predictions for the other levels. The second is a top-down approach, where the global demand is first predicted and then allocated to the entities (or projects) by estimating how much each entity (or project) will request from the total. Finally, if both approaches are feasible, they can be combined to build a consensus forecast.

Figure 5 – Hierarchy of the forecast levels

Source: Author.

Today, however, the company has in place a bottom-up forecast that fails to

provide a reliable signal at the global level, which leads to an increasing number of backorders at GST's workshops. Together with the fact that a large amount of data related to the tests is being stored, this has motivated the company's interest in searching for new forecasting approaches, in particular data-driven ones such as ML models.

Before diving into how such an approach might be designed, it is important to understand how the forecasts are made today. For that, one must first know that the RD projects are structured around a Gantt diagram built with a specific system, called Kephren, where all the development loops are planned in advance. In these diagrams, all the relevant information and resources needed by the project are listed, for instance where, when, and how many prototypes will be built, including the necessary tests.

The Gantt diagram is created by a Supply Chain (SC) analyst associated with the project, together with the rest of the team. Regarding the tests, the SC analyst will ask the project's engineer which resources will be needed and when, so that this information can be added to the system. In most cases, however, the SC analyst will add a standard package containing all the test resources typically requested by the entity's projects. Since, for each resource, the system has stored a standard duration, it can therefore estimate the load that each project will require for each test resource. Thus, the forecasts for the other levels can also be obtained. Finally, it is important to note that this software is not linked to GST, meaning that test requests will be made later using a different GST application.

Thus, despite all the effort spent on producing the forecasts, the final signal presents large errors that prevent it from being reliable for GST and the SOP actors. These errors are probably due to the use of outdated standard times and to the fact that engineers frequently indicate a subset of tests to be performed to the SC analyst, but afterwards request a completely different set in the GST application.

Notably, there are many ongoing projects at Michelin as well as many test machines, which means that a precise forecast at the project level is quite challenging. Since most projects only request a few of the test resources, it is important to break the forecasting problem into two parts: the first is predicting which test resources a project will use, and the second is how much it will request—in other words, predicting the test load.

Although there is plenty of test-related data available, most of the structured information is recent, only from 2023 onward. Thus, when choosing a top-down approach, most of the information related to the projects is aggregated and lost, risking leaving too little data to train an ML model. For that reason, the bottom-up approach was prioritized in the present work, with the hope that the high volume of project information might compensate for the short history.

3.1 FORMALIZATION OF THE PROBLEM

The following variables represents the information that is known beforehand.

- Set of projects: $P = \{p_1, p_2, \ldots, p_{N_P}\}$
- Each project have a list of attributes, called f_{p_i}, which contains for instance its entity and the type of tire that it aims to develop.
- Set of test resources: $R = \{r_1, r_2, \ldots, r_{N_R}\}$
- Time set: $T = \{t_1, t_2, \ldots, t_K\}$
- The set of the different entities: $E = \{E_1, E_2, \ldots, E_{N_E}\}$
- Each entity has a group of projects associated to it:

  $P_{E_i} = \{p \in P \mid \text{such that the project belongs to the entity } E_i \in E\}$
- Forecasting horizon: H

Then we can define the variables that we want to predict.

- The load used by the project p_i of the resource r_j at the time t_k: $x_{p_i r_j t_k}$
- The total load of the resource r_j used at the time t_k:

$$x_{r_j t_k} = \sum_{p_i \in P} x_{p_i r_j t_k}$$

Since most of the projects will only ask for a few resources, this sum can be broken into:

$$x_{r_j t_k} = \sum_{p_i \in P} x_{p_i r_j t_k} = \sum_{p_i \in P_{r_j,0}} x_{p_i r_j t_k} + \sum_{p_i \in P_{r_j,1}} x_{p_i r_j t_k} = \sum_{p_i \in P_{r_j,1}} x_{p_i r_j t_k}$$

where $P_{r_j,0}$ contains the projects that will not ask for the test resource r_j, meaning that

$$\forall p_i \in P_{r_j,0},\ \forall t_k \in T,\ x_{p_i r_j t_k} = 0.$$

- The total load of the resource r_j used by the entity E_i at the time t_k:

$$x_{E_i r_j t_k} = \sum_{p_i \in P_{E_i}} x_{p_i r_j t_k}$$

Then we can define the following vectors that represent how much of the resource r_j is used inside a time window t_{k+1} until t_{k+H} for the three hierarchical levels:

$$\mathbf{x}^H_{p_i r_j t_k} = [x_{p_i r_j t_k}\ x_{p_i r_j t_{k+1}}\ \cdots\ x_{p_i r_j t_{k+H}}]^\top$$

$$\mathbf{x}^H_{E_i r_j t_k} = [x_{E_i r_j t_k}\ x_{E_i r_j t_{k+1}}\ \cdots\ x_{E_i r_j t_{k+H}}]^\top$$

$$\mathbf{x}^H_{r_j t_k} = [x_{r_j t_k}\ x_{r_j t_{k+1}}\ \cdots\ x_{r_j t_{k+H}}]^\top$$

Therefore, the goal is to build a ML model that for each test resource r_j is able to predict which projects belong to $P_{r_j,1}$ and then forecast for those projects the future demand: at the project level, meaning the vector $\mathbf{x}^H_{p_i r_j t_k}$; at the entity level, $\mathbf{x}^H_{E_i r_j t_k}$; and at the global level, $\mathbf{x}^H_{r_j t_k}$. The prediction should be made using the information from f_{p_i}. Furthermore, thanks to the hierarchical structure of the problem, the following relations can be written:

$$\mathbf{x}^H_{E_i r_j t_k} = \sum_{p_i \in E_i} \mathbf{x}^H_{p_i r_j t_k}$$

$$\mathbf{x}^H_{r_j t_k} = \sum_{E_i \in E} \mathbf{x}^H_{E_i r_j t_k}$$

which means that, if a good prediction at the project level is achieved, then it is possible to easily obtain a prediction for the other levels too.

The next chapter proposes a solution that combines different classifiers and regressors in order to build a composite model that predicts the demand at the project level. Then, Chapter 5 deals with the creation of one of the composite model components, using the scikitlearn library from python, that should identify the set $P_{r_j,1}$ of projects that ask for a given test resource r_j.

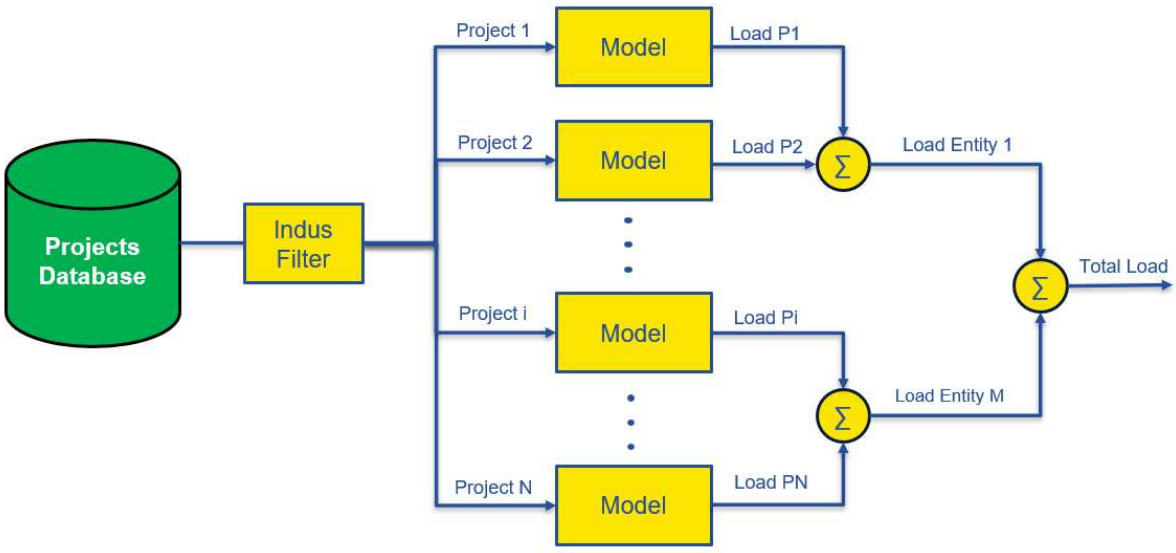
4 PROPOSED SOLUTION AND METHODOLOGY

The present chapter proposes a model architecture for a bottom-up approach to test forecasting.

Since the chosen approach is a bottom-up forecast, the most natural idea is to create a model that, given a project and a test resource as input, predicts the load for the project as output. Then, such a model can be repeatedly applied to each of the projects in the company's database. In the end, a forecast is made for each ongoing project. As for the other forecast levels, the entity-level forecast can be obtained by summing the forecasts of the entity's projects, and then, by summing all of the entity forecasts, one can obtain the forecast at the global level.

Nevertheless, there are too many projects in the database, and most of them will not use the test resource. One natural way of filtering out these projects is by selecting only the projects that build prototypes. However, sometimes tests are also requested by projects that do not build prototypes; this is the case for Conformity of Production (COP) projects, where samples of tires already in production are tested for quality control. The diagram in Figure 6 illustrates how the model should be applied.

Figure 6 – Proposed solution



Source: Author.

Table 2 illustrates that, most of the time, projects will not request the use of a test resource. The project's file from December 2024 was used, which contains almost 30,000 projects, of which only 8,113 had a prototype building associated with them. However, only a minority of 401 projects requested one of the three resources selected as an example in the next six months (from December 2024 to May 2025), which

represents only 1.35% of the projects in the database and 4.94% of the projects with an indus event. Still, it is worth noting that 35 projects without an indus event requested the test resource.

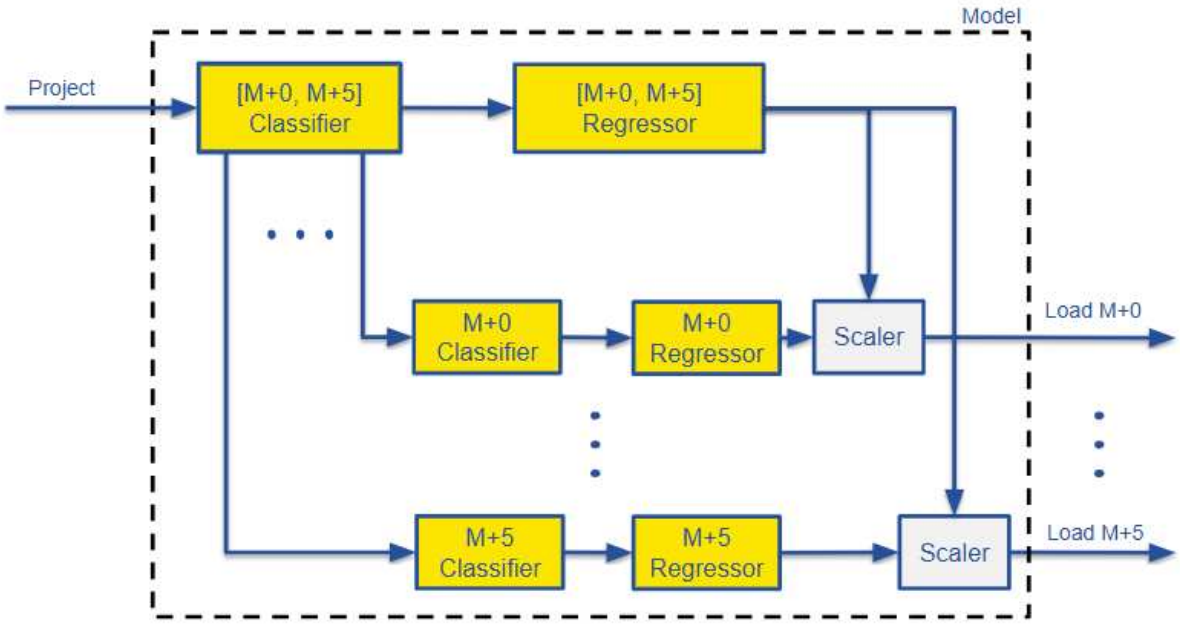
Table 2 – Distribution of the projects

	Number of projects
Project database	29681
Indus database	8113
Ask for a test resource	401
- Ask for resource 1	289
- Ask for resource 2	229
- Ask for resource 3	61
- Outside Indus database	35

Source: Author.

Therefore, if one wants to predict the test load demand for each project, most of the time the true value will be zero, which characterizes this task as a zero-inflated regression. Furthermore, this category of problems can be tricky, since if a model always predicts zero, it could still be considered good when evaluated with traditional metrics. To avoid such problems, a classifier is usually used before the regressor to filter out the examples where the expected output is zero, so that the regressor can focus only on the examples where the expected output differs from zero.

Figure 7 – Composite Model



Source: Author.

The structure of the composite model is shown in Figure 7. Naturally, predicting whether a project will use a test resource in a specific month is harder than predicting whether it will use it in the next six months. The same holds for predicting the test load. Moreover, predicting whether a project will use a test or not is a binary classification problem, since there are only two possible outcomes. In the composite model, a binary classifier first determines whether the project will request a test resource in the next six months, which is done by the [M+0, M+5] classifier. If so, a prediction of the total load is then made by the [M+0, M+5] regressor. Nevertheless, the goal is to have a prediction for each month, so this logic can be replicated for each month: the M+i classifier predicts whether the test resource will be used in month M+i, and the M+i regressor predicts the project's monthly load if resource usage is expected.

Therefore, the first classifier acts as a first filter, allowing only the projects that will request the test resource to be considered. Then, the individual classifiers indicate when the test will be requested. Finally, the regressors predict the workload of the test.

Since there will be a prediction for each individual month, one might ask what the purpose is of having a regressor that predicts the total load. Sometimes, a project might request a test one month after an indus event, and other times after two months; thus, consecutive classifiers might predict that the test will be requested in consecutive months, meaning that in such cases the sum of the predictions would likely indicate a load twice as large as the true one. Therefore, by scaling the individual predictions so that their sum equals the prediction of the total load, this problem can be avoided.

Nevertheless, determining whether a project will request a test resource falls into another special category of problems: an imbalanced classification task. As seen in Table 2, typically 95% of the projects will not request a specific test resource. Naturally, if a model always predicts that the test resource will not be used, it would still show a high accuracy of 95%. To deal with such problems, there are three main techniques:

- Undersample, which consists of randomly dropping examples of the majority class;
- Oversample, which consists of increasing the number of examples of the minority class, usually by making copies of them;
- Assigning a higher weight to the minority class examples in the loss function.

In summary, given a project, a test resource, and the current date, the model should be able to predict the load demand for each of the following six months. However, it is not wise to provide the project identification directly to the model; rather, project information should be provided so that the model can generalize its predictions to projects it has not seen before. Furthermore, the project information falls into three categories:

- Project characteristics, for instance the entity to which it belongs, the kind of tire it aims to develop, etc.;
- Test prior information related to the project, for instance the tests that the project has already requested, if any;
- Prototype building information, for instance whether the project has built or plans to build a prototype in the future.

In order to use this information, it is necessary to define how it should be provided to the ML model. The first type is straightforward, since for each attribute one can simply use an encoding technique. The second type is more difficult; one might suggest providing a boolean variable that indicates whether the project has used the resource in the past, and if so, how much. Nevertheless, the chosen approach was to provide as input the test load requested by the project for each of the recent past months. For the last type, a similar idea was applied for each of the past and upcoming months, since the prediction will also rely on the indus forecast. Then, for each month, a group of variables indicates the indus attributes; for instance, in month M-3, one variable would indicate whether there was an indus event, a second variable would indicate in which factory the prototype was made, a third would indicate how many prototypes were made, and in the case where no indus is predicted, the variables for that month are set to zero. Naturally, this strategy significantly increases the number of inputs provided to the model. Figure 8 shows how the data should be reorganized so that it can be given to the model.

Furthermore, one can see in Figure 8a the development loops, where the first two rows show when and where the tire prototypes were built, and the rest show the different tests that the project requested. Notably, some tests are requested almost every time after a prototype is built, such as resource 36, whereas others are rarely requested, such as resource 33. Unfortunately, there is no information linking a test to a prototype building event; although a test would normally relate to the previous indus, this is not always the case.

Then, Figure 8b illustrates how the timeline data presented in Figure 8a can be transformed to build different training examples. Each row represents a single example, characterized by a specific project, test resource, and date. Furthermore, when the trained model is applied in production, a similar dataset will be constructed and provided to the model, containing only data related to the current month. The model will then predict the output columns.

Figure 9 shows how the classifier that decides whether a project will use a test resource is structured. First, the project number is used to build features from three different databases: the first contains the project data, the second contains past test actuals, and the third contains the indus forecasts and past actuals. The implementation

Figure 8 – How the data should be transformed to feed the model.

Months		2023-01	2024-09	2024-10	2024-11	2024-12	2025-01	2025-02	2025-03	2025-04	2025-05	2025-06	2025-07	2025-08	2025-09	2025-10	2025-11
Indus	Indus Plant	NYH	CNO	0	0	0	0	0	0	NYH	BKR	0	CNO	0	NYH	0	CNO
	Indus Step	LOT	LOT	0	0	0	0	0	0	LOT	LOT	0	LOT	0	LOT	0	LOT
Tests	Resource 6	0	0	0	**9.00**	0	0	0	0	0	0	**20.00**	0	0	0	0	0
	Resource 36	0	**4.40**	**3.80**	0	0	0	0	0	0	0	**4.40**	0	**2.00**	**4.20**	0	0
	Resource 14	0	**0.40**	**0.30**	0	0	0	0	0	0	0	0	0	0	0	0	0
	Resource 1	0	**5.81**	**4.77**	0	0	0	0	0	0	0	**3.40**	0	0	**4.80**	0	0
	Resource 33	0	0	0	0	0	0	0	0	0	**2.12**	0	0	0	0	0	0
	Resource 42	0	0	**2.90**	**0**	**4.00**	0	0	0	0	0	**6.60**	**1.30**	0	**5.20**	0	0
	Resource 34	0	0	**4.20**	0	0	0	0	0	0	0	0	0	**2.10**	0	0	0
	Resource 4	0	0	0	0	0	0	0	0	0	**4.00**	0	0	0	0	0	0
	Resource 2	0	0	**1.00**	0	0	0	0	0	0	0	**3.00**	0	0	0	0	0

(a) Time Line with data for a specific project.

Input																Output					
Test	Time	Project Information			Past Test Load			Past Indus			Indus Forecast						Future Test Load					
		Type of tire	...	Entity	Load M-6	...	Load M-1	Indus M-6	...	Indus M-1	Indus M+0	Indus M+1	Indus M+2	Indus M+3	Indus M+4	Indus M+5	Load M+0	Load M+1	Load M+2	Load M+3	Load M+4	Load M+5
2	2025-05	-R	...	A	0	...	0	0	...	1	1	0	1	0	1	0	0	3.0	0	0	0	0
1	2025-05	-R		A	0		0	0		1	1	0	1	0	1	0	0	3.4	0	0	4.8	0
1	2025-06	-R		A	0		0	0		1	0	1	0	1	0	1	3.4	0	0	4.8	0	0

(b) Transformed data to be given to the model.

Source: Author.

of this classifier will be detailed in the next chapter. Then, the model predicts whether the project will use the test resource in the next six months, namely one if it will request it or zero if not. Moreover, a similar structure can be used for all the other individual components of the composite model.

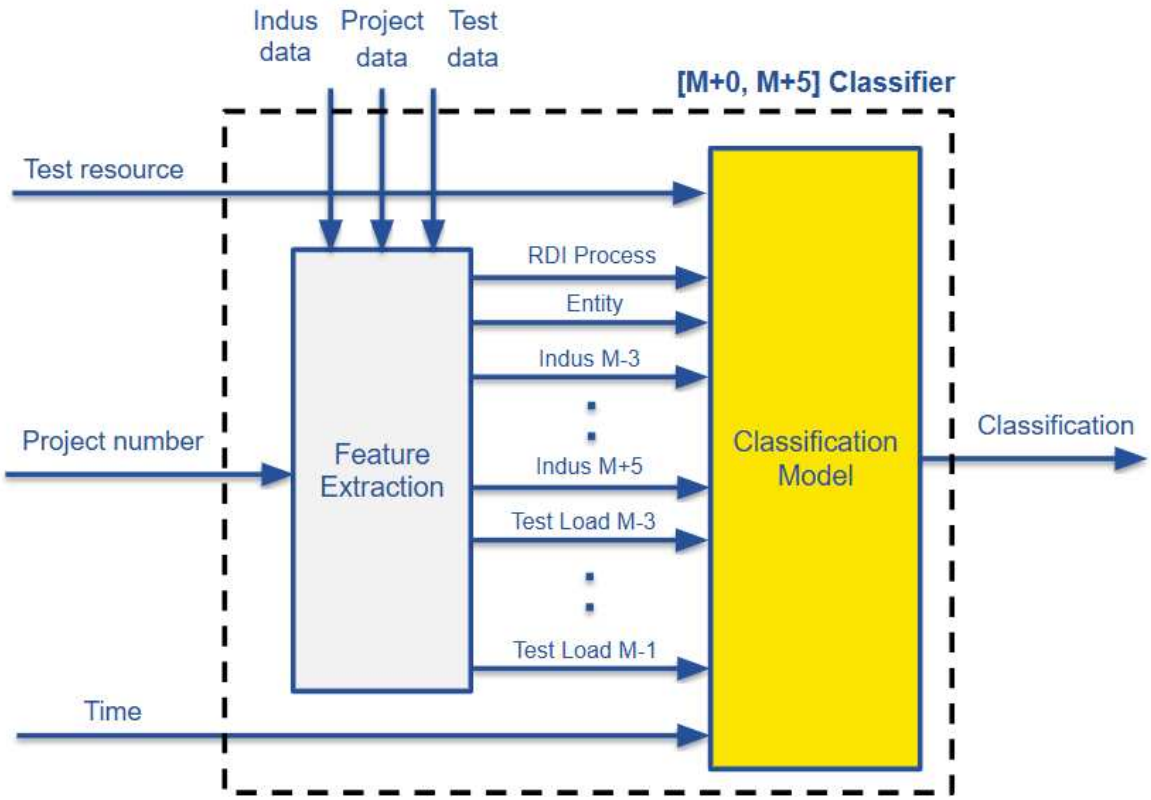
In order to evaluate the performance of the composite model, it is necessary to define which metrics should be used. Since the problem involves both classification and regression, two kinds of metrics are required. For the classifiers, one can use metrics derived from the confusion matrix (presented in Table 3). However, since the true negatives tend to be much larger than the other categories, metrics that incorporate them, such as accuracy and specificity (also known as the true negative rate), can be misleading. For this reason, it is more appropriate to use metrics such as precision, recall (also known as sensitivity or true positive rate), and their harmonic mean, namely F_1.

Precision indicates, among the projects predicted to use a test resource, how many actually use it:

$$Precision = \frac{TP}{TP + FP}.$$

Recall indicates, among the projects that actually use the resource, how many were

Figure 9 – Classifier



Source: Author.

Table 3 – Confusion matrix

	Predicted Negative	Predicted Positive
Actual Negative	TN	FP
Actual Positive	FN	TP

Source: Author.

predicted to use it:

$$Recall = \frac{TP}{TP + FN}.$$

Specificity is the recall of the negative class; it indicates, among the projects that actually do not use the resource, how many were correctly predicted as not using it:

$$Specificity = \frac{TN}{TN + FP}.$$

Finally, accuracy indicates what fraction of all examples were classified correctly:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}.$$

Then, for the cases where the test is correctly classified as being requested by the project, one can use traditional regression metrics such as the Mean Absolute Error

(MAE) to evaluate the prediction. However, projects that are incorrectly classified by the classifiers also impact the forecast at higher forecast levels. Thus, it is important to measure the prediction errors at the entity and global levels, which can also be done with standard metrics such as MAE.

Finally, it is important to note that there are many other possible architectures that could address the problem. For instance, one could imagine a different approach, where a prediction is made after each prototype building event for that particular month. However, this might be difficult, since there is no field information linking a prototype building to a test. Thus, one of the project's challenges is to select the architecture that best suits the problem. The next chapter will focus on the implementation of the first component of the composite model: the classifier that predicts whether a project will request a test resource in the following six months.

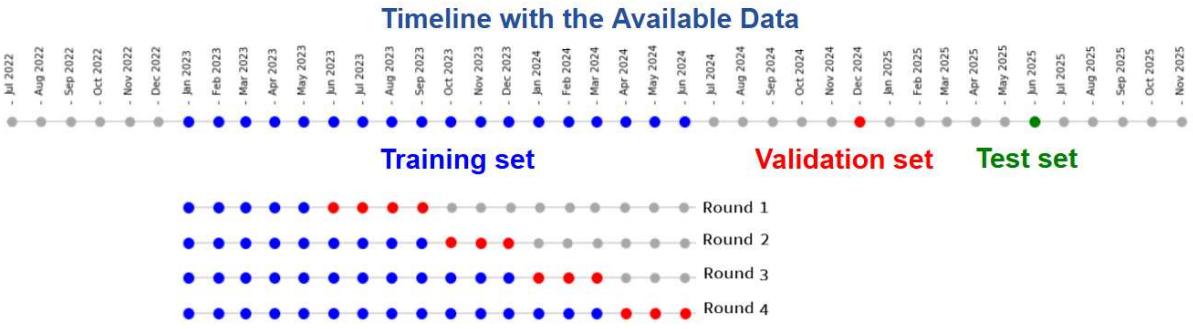
5 SOLUTION IMPLEMENTATION AND ANALYSIS

The present chapter describes how the first component of the composite model was built and evaluated.

In the previous chapter, it was defined how the model should be applied, meaning that for each project in the project database, a prediction is made by the classifier, which receives project-related information as input (as seen in Figure 9). Moreover, the expected output of the classification is either one, meaning that the project will request a test resource in the following six months, or zero, meaning that it will not. Thus, the goal is for the model to 'learn' the relationship between the inputs and the expected output. Therefore, it is necessary to train the model by providing it with past examples of input and output pairs.

For that reason, a set of examples was created using historic data. For each month, the file with the project information for that month was retrieved together with the test and indus actuals, so that the input-output pairs could be built. Knowing that only data from July 2022 to November 2025 was available, and that the model should receive information from the past six months and make a prediction for the next six months, examples could only be built from January 2023 to June 2025. Furthermore, the available examples were split into training, validation, and test sets; however, to avoid data leakage from the training set to the validation set, and from the validation set to the test set, months where the prediction would cover the same months were dropped. For instance, when predicting whether a project will request a test resource in the next six months, using data from July 2024 would create a five-month overlap with the data examples from June 2024. In Figure 10, the final partition is shown, where the examples in grey were only indirectly used.

Figure 10 – Partion of the available data into training, validation and test sets



Source: Author.

Although the original problem was to predict the monthly demand from each project for each test resource over a time horizon of 18 months, the classifier will only

make predictions for a six-month horizon. Nevertheless, if one wants an 18-month forecast, it can be obtained by applying the classifier three times in a row. Furthermore, another simplification was considered: instead of predicting the demand for all the resources, the model will only be applied to three resources. These three resources were selected based on their importance to the company and in the hope that they would be sufficiently representative of the other resources.

As discussed in the previous chapter, predicting if a project will use a test resource or not is an imbalanced classification task. Indeed, this can be clearly seen in Table 4, where from all the 623,181 examples generated, only 15,167 examples (2.43% of the total) asked for one of the three resources. Especially for resource 3, for which only 0.26% of the examples were asked. In order to address this imbalance, part of the training examples belonging to the majority class was removed. In other words, an undersampling technique was applied. The undersampling was performed to obtain a new class proportion: previously, 97% of the examples belonged to the majority class, whereas this value was reduced to 90%. Although the dataset remains imbalanced, more than 400,000 examples were removed. Furthermore, reducing the number of examples also shortened the model training time. However, the choice of the new proportion (90%) was arbitrary and could be the subject of a more detailed search.

In order to train the model, the data must be represented in numerical form, as discussed in Chapter 2. Therefore, the categorical features of each example were encoded using label encoding, in which an integer is assigned to each category. Given the large number of distinct categories in each input variable, one-hot encoding was not considered appropriate, as it would result in a very high-dimensional input space. Similarly, target encoding was also discarded, as it would introduce a large number of zeros due to the predominance of the zero class in the output.

Table 4 – Distribution of all the examples available

	Number of examples
All	623181
Ask for a test resource	15167
- Ask for resource 1	7586
- Ask for resource 2	5984
- Ask for resource 3	1597

Source: Author.

Then, an ML model should be chosen to serve as the classifier. Although many different options are available, the present work focused on one type of model: the random forest. This choice was made based on the following reasons:

- Tree-based models do not suffer much from having inputs with large dimensions, and thus do not require the use of dimensionality reduction techniques such as

Principal Component Analysis (PCA);

- Tree-based models do not suffer from having inputs at different scales, and thus do not require the use of scaling techniques such as normalization or standardization;
- Tree-based models can provide predictions with higher interpretability; for instance, one can see the importance assigned to each feature in the forecasts.

Afterwards, it is necessary to find the best hyperparameters to build the random forest, such as how many individual trees to use, how deep the individual trees can grow, etc. For that reason, a randomized search was performed over the grid of hyperparameters defined in Table 5. In the search, random combinations of hyperparameters are used to build the different models. Then, each model is evaluated using cross-validation to give an unbiased estimate of the models' future performance. In Figure 10, one can see how cross-validation is done by dividing the training data into different folds, so that in each round the first folds are used to train the model and the next fold is used to compute a performance score. Thus, in the end, if there are four rounds, four scores can be computed. Then, the search algorithm computes the mean score in order to compare the performance of the different models. By doing so, one avoids training the model with future information relative to the validation fold, thus avoiding data leakage.

Table 5 – Grid of hyperparameters to search

Symbol	Hyperparameter	Possible values
-	Class weight	(balanced, balanced subsample, none)
d_{max}	Maximum depth	(20, 30)
$n_{min,leaf}$	Minimum number of samples in a leaf	(4, 12, 20, 28, 36, 44)
$n_{min,split}$	Minimum number of samples to split	(4, 12, 20, 28, 36, 44)
$n_{max,samp}$	Maximum number of samples	(1.0, 0.8, 0.7, 0.5)
$n_{max,feat}$	Maximum number of features	(sqrt, 0.5, 1)
n_{est}	Number of estimators	(50, 100, 150, 200)

Source: Author.

The class weight parameter is used to counterbalance the imbalanced nature of the problem. When the option "balanced" is chosen, the loss function is weighted according to the frequency of each output class in the training data. When "balanced subsample" is selected, the weights are computed based on the class frequencies within each bootstrap sample rather than the original dataset. If "none" is chosen, no class weighting is applied.

The hyperparameters maximum depth, minimum number of samples per leaf, and minimum number of samples required to split a node control the growth of the individual trees in the random forest in order to reduce overfitting. The first specifies the maximum number of splits along any single path starting from the initial region. The second defines the minimum number of samples that must be present in a leaf

Table 6 – Search results

$t_{mean,fit}$	n_{est}	$n_{min,split}$	$n_{min,leaf}$	$n_{max,samp}$	$n_{max,feat}$	d_{max}	Class weight	s_1	s_2	s_3	s_4	s_{mean}
20	50	12	20	1.0	0.5	30		76.8	81.0	82.4	82.5	80.7
54	150	36	12	0.7	0.5	20	balanced subsample	79.3	80.4	81.5	80.6	80.5
8	100	4	4	0.5	sqrt	30	balanced	78.7	79.6	81.5	81.1	80.2
92	150	20	12	0.7	1.0	20	balanced subsample	80.3	80.6	80.3	79.5	80.2
54	150	4	28	0.7	0.5	30		76.1	80.3	81.0	81.8	79.8
77	200	12	20	0.8	0.5	20	balanced subsample	78.8	79.9	80.4	80.0	79.8
49	100	28	12	0.5	1.0	30	balanced subsample	80.1	79.4	79.9	79.5	79.7
106	150	20	28	1.0	1.0	20		76.1	80.0	81.2	81.1	79.6
123	200	36	28	0.8	1.0	30		75.3	80.1	81.2	80.5	79.3
48	100	20	20	0.5	1.0	30		75.4	79.5	81.0	81.0	79.2
29	50	12	20	0.7	1.0	20	balanced subsample	79.0	78.4	78.7	79.0	78.8
37	100	36	28	0.8	0.5	20	balanced subsample	78.3	78.9	78.7	79.0	78.7
64	100	28	44	1.0	1.0	30		74.1	79.2	80.2	80.1	78.4
68	200	28	36	0.8	0.5	20	balanced subsample	77.1	78.3	78.1	78.4	78.0
6	50	20	12	0.8	sqrt	20		75.7	77.0	76.7	78.5	77.0
10	100	36	12	0.7	sqrt	20	balanced subsample	76.1	76.2	77.1	76.4	76.5
77	200	20	44	0.5	1.0	30	balanced subsample	75.3	77.1	75.9	73.1	75.4
6	50	4	20	0.8	sqrt	30		73.0	76.1	75.5	76.3	75.2
25	200	20	28	1.0	sqrt	20	balanced subsample	74.1	74.6	75.2	75.1	74.7
16	200	44	44	0.7	sqrt	20	balanced	73.0	74.0	74.0	73.5	73.6

Source: Author.

node. The third specifies the minimum number of samples required in a node for it to be eligible for splitting.

The maximum number of samples and maximum number of features parameters help introduce diversity among the trees. The former limits the number of training examples provided to each tree: a value of 1.0 indicates that all available samples may be used, whereas a value of 0.8 means that only 80% of the samples are considered. Similarly, the latter controls how many features are used when constructing each tree. A value of 1 indicates that all features are considered; "sqrt" means that, if N is the total number of features, only $\sqrt{N}$ features are used; finally, a value of 0.5 indicates that only 50% of the features are considered.

Finally, the number of estimators specifies how many individual trees are built to form the random forest. In this search, forests with 50, 100, 150, or 200 trees were evaluated.

Furthermore, instead of testing every possible combination of the hyperparameters described in Table 5 to select the best one, a randomized search was performed, in which only a subset of all possible combinations is tested to use fewer computational resources. The results of the randomized search are presented in Table 6, in which s_i, expressed as a percentage, is the performance score of the tree built in round i of the cross-validation; s_{mean}, used to compare the different models, is the mean score over all rounds; and $t_{mean,fit}$ is the mean time, in seconds, taken to train the tree in each round.

From Table 6, one can see that the best Random Forest consists of 50 individual trees, with minimum numbers of samples per leaf and required to split set to 20 and 12, respectively, which helps prevent overfitting. Surprisingly, the best-performing configu-

ration does not use class weights to counteract the imbalanced nature of the problem, which may be justified by the fact that an undersampling procedure was already applied. Nevertheless, it is fairly hard to identify what makes this configuration the best, since the scores are very similar even across substantially different hyperparameter choices.

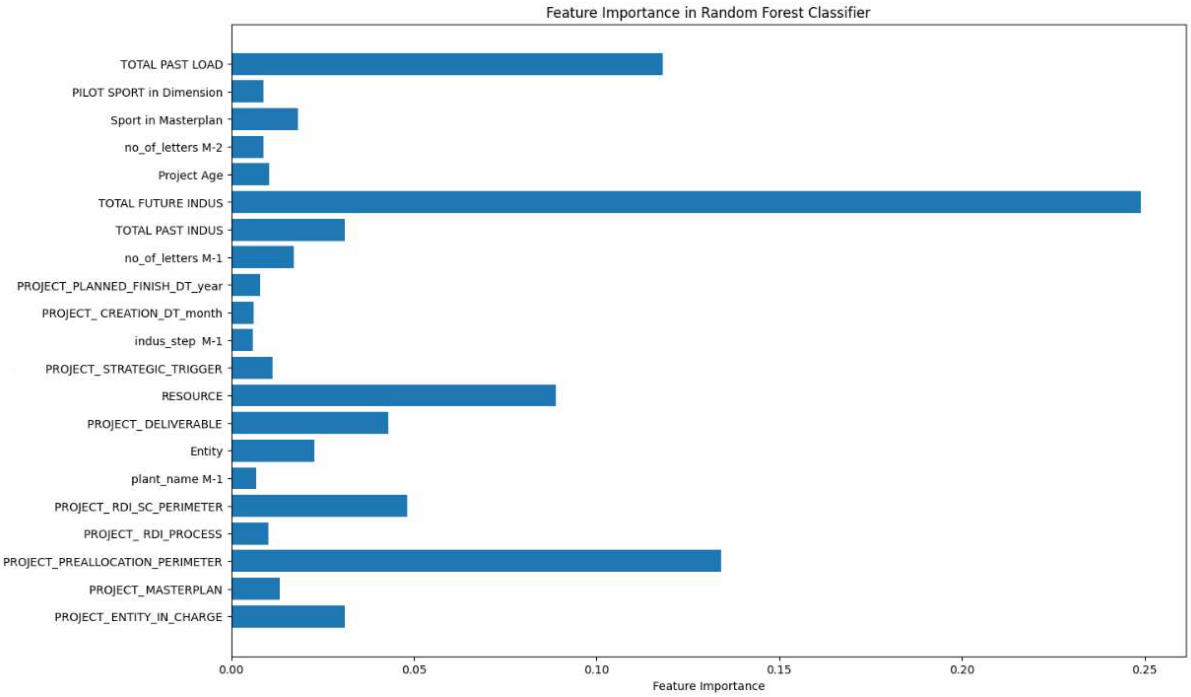
Then, with the Random Forest obtained from the randomized search, one can examine the features that are most relevant when making predictions. Thus, the feature importances were plotted and are shown in Figure 11. Indeed, these features are the ones already expected to be the most important, as they are coherent and consistent with the previously established hypotheses. As can be seen, the model inputs that best help to distinguish between the two possible classes are:

- "TOTAL FUTURE INDUS", which indicates if the project will build a prototype in the next six months or not;
- "PROJECT PREALLOCATION PERIMETER", which incorporates information about the geographic location of the project and the entity that it belongs;
- "TOTAL PAST LOAD", which indicates how much load the project has asked for the specific test resource in the past six months;
- "RESOURCE", which indicates which one of the three test resources the example relates too.

The performance of the model on the training set shows a recall of 90.3%, which means that the model detects most of the cases where the test resource is requested. Nevertheless, the precision of 52.6% indicates that about half of the cases where the model predicts the use of the resource are false positives. Therefore, the combination of these two metrics gives an F_1 score of 66.5%, which, although it seems low, effectively filters out most of the cases where the project will not request a test resource. Furthermore, one can see in Table 7 that performance slightly decreases on the validation set, with the F_1 score now equal to 58.6%.

Moreover, one can ask how the decision threshold can impact the performance of the model, since predictions are made by comparing the probability given by the Random Forest to a fixed threshold. In the standard case, the model predicts the use of the test resource if the probability given by the Random Forest is higher than 50%. Naturally, if this threshold is increased, fewer examples will be classified as positive; thus, precision might increase, at the potential cost of decreasing recall. To find the threshold that provides the best trade-off between precision and recall, a precision-recall curve can be generated. Figure 12a shows the curve generated using examples from the validation set. There, one can see, for instance, that a precision higher than 60% cannot be achieved by the model without reducing recall to less than 50%. After

Figure 11 – Most important features for the classifier



Source: Author.

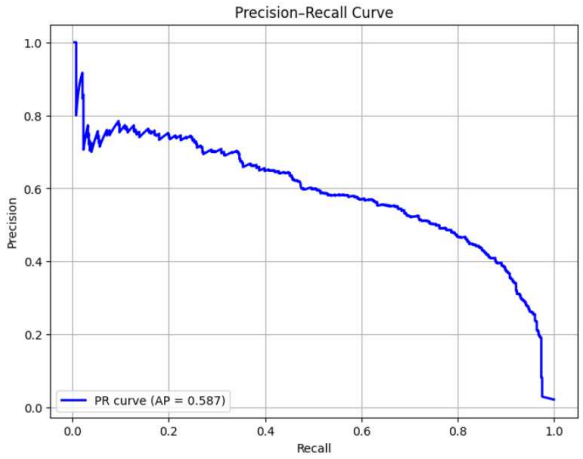
the desired pair of values is chosen, the corresponding threshold can be identified by plotting the metric values for different thresholds, as in Figure 12b.

In the case of searching for the model that best filters out the projects that will not use the test resource, one can select the threshold that gives the highest recall while maintaining a reasonable precision, so that the following monthly classifiers can be trained to improve the precision of the composite model. However, a threshold value of 0.7 was chosen, as the resulting model presents similar values of precision and recall. Therefore, the forecast will contain roughly the same number of projects in the prediction and in the test actuals, potentially resulting in a similar load being forecasted at the higher hierarchical levels. The Average Precision (AP) summarizes the information from the recall curve and can be used to compare the model obtained from the randomized search with different models.

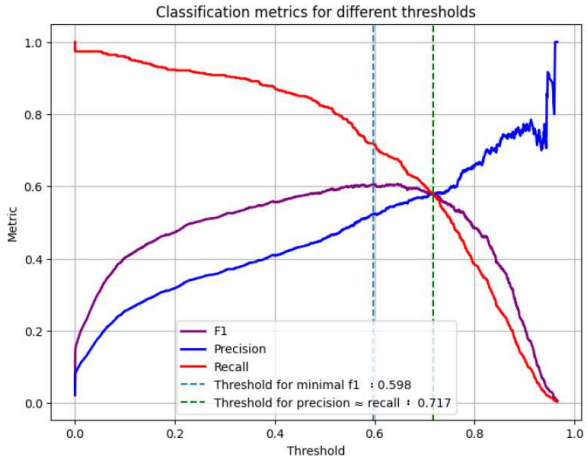
For standard classification problems, the Receiver Operating Characteristic (ROC) is used instead of the precision-recall curve, in the same way that the Area Under the Curve (AUC) is used instead of the AP score. However, for imbalanced classification problems, this is not recommended, since the ROC uses specificity instead of the precision score, which gives overly optimistic values because it takes true negatives into account.

Thus, with the threshold value defined, one can test the performance of the resulting model on the test set, as presented in Table 7. Indeed, recall and precision are

Figure 12 – Influence of the different thresholds.



(a) Precision-Recall curve.



(b) Precision, recall and F_1 score for the different thresholds.

Source: Author.

similar, and the F_1 score is also 59%. Furthermore, it is worth noting how the available examples were divided: the first part was used together with cross-validation to find and train the best model; the second was used to determine the best decision threshold; and the last was used to estimate the model's performance on unseen data.

Table 7 – Performance Summary

	Precision (%)	Recall (%)	F_1 (%)
Training Set	52.6	90.3	66.5
Validation set (standard threshold)	45.5	82.0	58.6
Validation set (new threshold)	57.8	57.7	57.7
Test set	60.2	58.0	59.1

Source: Author.

Finally, one can ask whether the model has similar performance for each test resource, since the previous metrics considered all of them at once. Indeed, as can be seen in Tables 8, 9, and 10, the models show similar performance, although it is slightly lower for resource 3, which has the smallest number of examples. Although the F_1 score is not impressively high, for the three resources the models are capable of filtering out most of the cases where the test resource is not requested, while giving a good estimation of how many projects will request it. For instance, among the 273 projects that used the first test resource, 160 were correctly classified. On the other hand, among the 8,416 projects that did not use the resource, 8,321 were correctly classified. Therefore, the resulting model exhibits high specificity but only moderate precision.

Table 8 – Confusion matrix for Resource 1

	Predicted Negative	Predicted Positive
Actual Negative	8321	95
Actual Positive	113	160

Source: Author.

Table 9 – Confusion matrix for Resource 2

	Predicted Negative	Predicted Positive
Actual Negative	8406	80
Actual Positive	86	117

Source: Author.

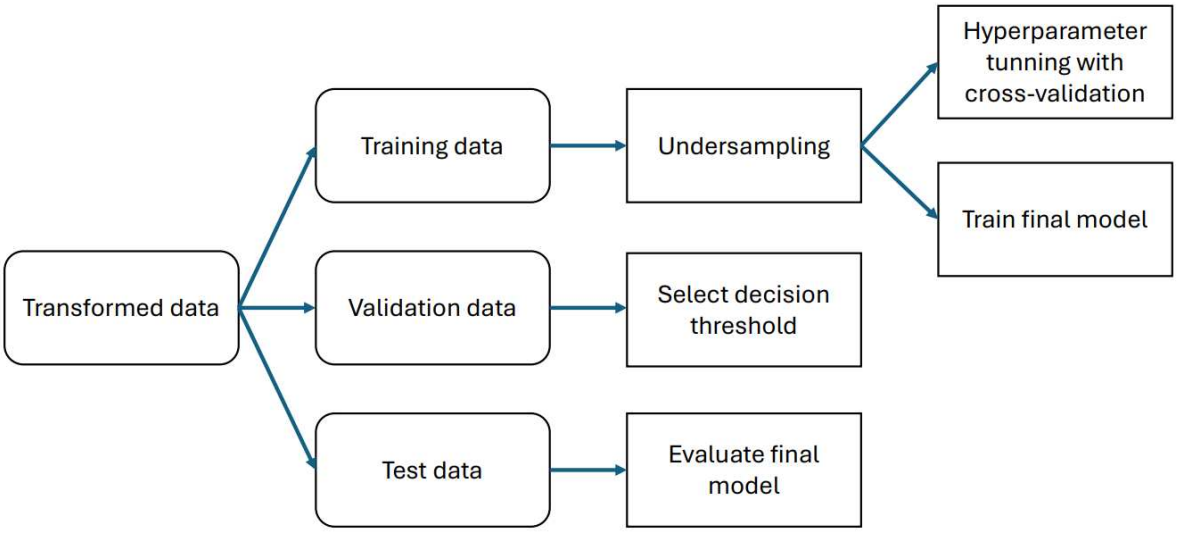
Table 10 – Confusion matrix for Resource 3

	Predicted Negative	Predicted Positive
Actual Negative	8632	21
Actual Positive	16	20

Source: Author.

Finally, Figure 13 summarizes how the available data were used to build the predictor. First, the data are gathered, cleaned, and encoded so that they can be used by the ML model. The dataset is then split into three subsets: the training set, which is undersampled and subsequently used to identify the best combination of Random Forest hyperparameters through cross-validation, as well as to train the final model; the validation set, which is used to analyze the influence of different decision thresholds on model performance and to select the one that best suits the needs of the project; and finally the test set, which is used to obtain a final estimate of the model's ability to predict whether a project will request a test resource over the following six months.

Figure 13 – Workflow for data usage and model training



Source: Author.

6 CONCLUSION AND FUTURE WORK

This final chapter presents a summary of the problem discussed in the present work and the main achievements, as well as the next steps toward the implementation of a machine learning–based model to forecast tire test demand.

6.1 CONCLUSIVE SUMMARY

The present work faced the challenging task of reformulating the way that the demand for tire testing is forecasted at Michelin's RD center. First, the demand was characterized as having different hierarchical levels, with project-level demand as its elementary unit. Then, two possible approaches were proposed: top-down or bottom-up. In the top-down approach, the demand can be considered "well-behaved," since it is, in most cases, continuous. However, the latter approach was chosen to predict the forecast at the project level, where the demand is sparse, because, by starting at the project level, the forecast for the other levels can be easily obtained.

Second, the demand was studied and characterized at the project level as being zero-inflated. In order to reduce its negative effects, the problem was divided into two sub-problems: first, predicting which projects will request the test resource, and only then predicting how much they will request. Nevertheless, the first sub-problem turned out to be an imbalanced classification task, which also requires special handling; in this case, undersampling.

Third, an architecture of a composite model that combines different classifiers and regressors was proposed in order to predict project-level test demand. Naturally, one may ask how such a model can be evaluated once built, so metrics that take into account the different dimensions of the problem were proposed.

Finally, the first and most critical component of the composite model was built and tested within a delimited scope. This choice allowed the feasibility of the proposed solution to be evaluated, as well as providing a base and guideline for the development of the subsequent models. Although the classifier can still benefit from better tuning to improve its performance, the results were considered promising, as it requires fewer resources than the previous forecasting method used in the company while providing a forecast with higher precision in terms of the projects that request the test resource. Thus, it should help to correctly identify the projects and entities with the highest demands and facilitate the SOP process. Although only the first of the three sub-problems presented in the introduction was addressed in this work, it provides a foundation for tackling the remaining components in future developments.

6.2 FUTURE WORK

Although efforts can still be made to improve the performance of the classifier described in Chapter 5, we suggest that the other components of the composite model should be developed first, so that the feasibility of the full model can be quickly evaluated and then optimized as an ensemble.

It would also be interesting to implement a top-down approach, in which a separate forecast of the test workload is made at the other levels, allowing them to be combined into a more robust forecast using a reconciliation technique. Finally, after a careful validation phase, the model should be deployed, and its forecast should be made available alongside the already existing forecast. In this way, the company can gradually replace the existing forecast with the new data-driven one.

REFERENCES

ABRAHAM, Zubin; TAN, Pang-Ning. A Semi-supervised Framework for Simultaneous Classification and Regression of Zero-Inflated Time Series Data with Application to Precipitation Prediction. *In:* 2009 IEEE International Conference on Data Mining Workshops. [*S. l.: s. n.*], 2009. p. 644–649. DOI: 10.1109/ICDMW.2009.80.

ATHANASOPOULOS, George; HYNDMAN, Rob J.; KOURENTZES, Nikolaos; PANAGIOTELIS, Anastasios. Forecast reconciliation: A review. **International Journal of Forecasting**, 2024. DOI: 10.1016/j.ijforecast.2023.10.010.

CAMPO, Patrice. **Comment Michelin conçoit ses pneus en équipes à Ladoux, près de Clermont-Ferrand**. La Montagne. 2025. Available from: https://www.lamontagne.fr/clermont-ferrand-63000/economie/comment-michelin-concoit-ses-pneus-en-equipes-a-ladoux-pres-de-clermont-ferrand_14714833/.

GATNAR, Eugeniusz. Tree-based Models in Statistics: Three Decades of Research. *In:* JAJUGA, Krzysztof; SOKOŁOWSKI, Andrzej; BOCK, Hans-Hermann (eds.). **Classification, Clustering, and Data Analysis**. Berlin, Heidelberg: Springer, 2002. p. 399–407.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. **Deep Learning**. [*S. l.*]: MIT Press, 2016. http://www.deeplearningbook.org.

LEWBEL, Arthur; NESHEIM, Lars. **Sparse demand systems: Corners and complements**. [*S. l.*], 2019. DOI: 10.1920/wp.cem.2019.4519.

MICHELIN. **Innovation: the VISION concept**. 2026b. Available from: https://www.michelin.com/en/group/activities/tires/vision-concept.

MICHELIN. **L'histoire Michelin**. 2026a. Available from: https://www.michelin.com/groupe/histoire.

ROŽANEC, Jože M.; PETELIN, Gašper; COSTA, João; CERAR, Gregor; BERTALANIČ, Blaž; GUČEK, Marko; PAPA, Gregor; MLADENIĆ, Dunja. Dealing with zero-inflated data: Achieving state-of-the-art with a two-fold machine learning approach. **Engineering Applications of Artificial Intelligence**, v. 149, p. 110339, 2025. ISSN 0952-1976. DOI: 10.1016/j.engappai.2025.110339.

RUSSELL, Stuart J.; NORVIG, Peter. **Artificial Intelligence**: A Modern Approach. Upper Saddle River, NJ: Pearson Education, 2010.

SCIKIT-LEARN. **Scikit-learn User Guide: Ensemble Methods**. Scikit-learn. 2026. Available from: https://scikit-learn.org/stable/modules/ensemble.html#forest.