UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
ESCUELA DE POSTGRADO Y EDUCACIÓN CONTINUA
DEPARTAMENTO DE INGENIERÍA MATEMÁTICA

ASTRONOMICAL SEEING PREDICTION AT PARANAL OBSERVATORY

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIA DE DATOS

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL MATEMÁTICO

ARTURO IGNACIO LAZCANO GONZÁLEZ

PROFESOR GUÍA:
FRANCISCO FÖRSTER BURÓN

PROFESOR CO-GUÍA:
JAIME SAN MARTÍN ARISTEGUI

MIEMBROS DE LA COMISIÓN:
JOSEPH PAUL ANDERSON
JOAQUÍN FONTBONA TORRES

SANTIAGO DE CHILE
2024

RESUMEN DE LA TESIS PARA OPTAR
AL GRADO DE MAGÍSTER EN CIENCIA
DE DATOS Y MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL MATEMÁTICO
POR: ARTURO IGNACIO LAZCANO GONZÁLEZ
FECHA: 2024
PROF. GUÍA: FRANCISCO FÖRSTER BURÓN
PROF. CO-GUÍA: JAIME SAN MARTÍN ARISTEGUI

PREDICCIÓN DE LA VISIBILIDAD ASTRONÓMICA EN EL OBSERVATORIO PARANAL

La visión o visibilidad astronómica (en inglés, *seeing*) es una variable de turbulencia óptica que refiere al efecto distorsionador de la atmósfera sobre las imágenes capturadas por los observatorios de objetos astronómicos. Este parámetro se puede calcular en tiempo real con diferentes instrumentos, entre ellos, el *Differential Image Motion Monitor* (DIMM). Sin embargo, los observatorios requieren usualmente una predicción de antemano para poder planificar la observación de algún objeto astronómico. Aquí es donde entra el área de simulación numérica y aprendizaje de máquinas, siendo dos enfoques muy utilizados en la predicción de variables atmosféricas.
El Observatorio Europeo Austral (ESO, por sus siglas en inglés) es la organización intergubernamental de ciencia y tecnología más importante en astronomía, la cual opera en tres sitios diferentes: La Silla, Paranal y Chajnantor. De estos, Paranal es el enfoque de esta tesis, donde se encuentra el *Very Large Telescope* (VLT). Así, el objetivo de este trabajo es predecir el valor promedio del seeing una hora hacia el futuro, utilizando datos provenientes de la ESO y un marco enteramente enfocado en aprendizaje de máquinas.

En esta tesis se ha desarrollado una modificación extensa de la base de datos proveniente de la ESO, junto con un análisis de esta, con el fin de generar los mejores resultados posibles. Además, se han probado de manera exhaustiva diferentes modelos para evaluarlos y seleccionar el óptimo en términos de predicción.

El resultado obtenido fue de 0.151 de RMSE y 0.753 de Success Rate, lo que refleja una mejora en un 14.2% y un 4.15%, respectivamente, con respecto al modelo de predicción actual. Esto se traduce a predicciones más precisas junto con un mayor porcentaje de observaciones que cumplieron los requisitos de *seeing* solicitados.
Comparado con el modelo que se usa actualmente en la ESO, esta nueva implementación refleja, en promedio, 28 horas ganadas sobre ejecuciones que cumplen los requisitos de *seeing* por semestre, donde 18 son de las observaciones más importantes y 10 son de aquellas de menor grado. Junto a esto, se redujo, en promedio, 40 horas el tiempo en el cual se usa el telescopio para una ejecución fallida por semestre, es decir, se observan más ejecuciones de calidad, se ocupa menos el telescopio en ejecuciones que no cumplen los requisitos solicitados y se gana la capacidad de llenar la cola de nuevas OBs (Observation Blocks), es decir, ejecutar nuevas observaciones astronómicas.

RESUMEN DE LA TESIS PARA OPTAR
AL GRADO DE MAGÍSTER EN CIENCIA
DE DATOS Y MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL MATEMÁTICO
POR: ARTURO IGNACIO LAZCANO GONZÁLEZ
FECHA: 2024
PROF. GUÍA: FRANCISCO FÖRSTER BURÓN
PROF. CO-GUÍA: JAIME SAN MARTÍN ARISTEGUI

ASTRONOMICAL SEEING PREDICTION AT PARANAL OBSERVATORY

Astronomical seeing is an optical turbulence variable that refers to the distorting effect of the atmosphere on images captured by observatories of astronomical objects. This parameter can be calculated in real-time using different instruments, including the Differential Image Motion Monitor (DIMM). However, observatories usually require a prediction in advance to plan the observation of an astronomical object. This is where numerical simulation and machine learning come into play, as they are two widely used approaches for predicting atmospheric variables.
The European Southern Observatory (ESO) is the most important intergovernmental science and technology organization in astronomy, operating in three different sites: La Silla, Paranal, and Chajnantor. Among these, Paranal is the focus of this thesis, where the Very Large Telescope (VLT) is located. Thus, the objective of this work is to predict the average value of seeing one hour into the future, using data from the ESO and an entirely machine learning-focused framework.

In this thesis, an extensive modification of the database from the ESO has been developed, along with an analysis of it, to generate the best possible results. Additionally, different models have been exhaustively tested to evaluate and select the optimal one in terms of prediction.

The result obtained was 0.151 RMSE and 0.753 Success Rate, reflecting improvements of 14.2% and 4.15%, respectively, compared to the current prediction model. This translates to more accurate predictions along with a higher percentage of observations meeting the requested seeing requirements.
Compared to the model currently used at the ESO, this new implementation reflects an average gain of 28 hours per semester for executions that meet the *seeing* requirements, with 18 hours from the most important observations and 10 hours from those of lower rank. Additionally, the time spent using the telescope for failed executions was reduced by an average of 40 hours per semester. This means more high-quality executions are observed, the telescope is used less for executions that do not meet the required standards, and there is increased capacity to fill the queue with new OBs (Observation Blocks), that is, to conduct new astronomical observations.

Por una cabeza,
todas las locuras, su boca que besa,
borra la tristeza, calma la amargura.

- Carlos Gardel

Agradecimientos

Agradezco primero a mis profesores guías, Francisco Förster y Jaime San Martín, por el gran apoyo y transmisión de sabiduría en todo este proceso de tesis. Su constante ayuda junto con las personas de la ESO; Joseph Anderson, Natalie Behara, Ángel Otarola, Elyar Sedaghati y Faviola Molina que hicieron de este proceso algo tremendamente disfrutable y enriquezedor.

Agradezco a mis compañeros y compañeras de la universidad que tuve todos estos años. La unión que tuvimos desde el primer minuto y las constantes risas son lo más valioso que me llevo de la universidad. Gracias Seba, Nico, Toro, Samu y Dani.

Agradezco a mis compañeros del colegio, los cuales fueron y son un pilar en mi trayectoria. Su amistad me mantiene firme a lo largo de situaciones complicadas. Gracias, por sobre todo, Alex.

Agradezco a mi amigo de la infancia Joaco. Hemos estado juntos desde que nacimos y solo espero que podamos seguir esta hermosa amistad.

Agradezco a mi polola, Ale, la cual fue probablemente el mayor apoyo en este proceso y de la cual me siento muy orgulloso de lo que hemos logrado. Gracias por estar ahí para mi.

Agradezco, por último, a toda mi familia. Madre, Abue, Mati, Alejandro, Yoyo, Kari, Mane, Cristián, primos y primas por todo. Sin ustedes no habría llegado aquí, especialmente a ti Abue que me recibiste en todo este viaje de la universidad. Esto es por y para ustedes.

Gracias por todo Mamá, te amo. Espero que el tata se pueda sentir orgulloso de esto.

Table of Content

List of Tables

List of Figures

Chapter 1

Introduction

1.1. European Southern Observatory

The European Southern Observatory (ESO) is an intergovernmental organization with headquarters located in Garching, near Munich, Germany and three unique world-class observing sites in the Atacama Desert: La Silla, Paranal and Chajnantor.
ESO's first site is at La Silla, a 2400 m high mountain 600 km north of Santiago de Chile. At the time of writing, it is equipped with several optical telescopes with mirror diameters of up to 3.6 metres. While La Silla continues to lead in the field of astronomy and remains the second most scientifically productive ground-based astronomical observatory, the Paranal site, standing at an altitude of 2600 meters, represents the pinnacle of European astronomy. Home to the renowned Very Large Telescope array (VLT), Paranal is strategically located approximately 130 kilometers south of Antofagasta in Chile, positioned 12 kilometers inland from the Pacific coast, in one of the driest regions globally. Scientific activities commenced at Paranal in 1999 and have since led to numerous highly successful research programs.

The VLT is an extraordinary telescope that utilizes state-of-the-art technology. Rather than being a singular entity, it comprises four Unit Telescopes (UT), each boasting an 8.2-meter main mirror. Remarkably, a single Unit Telescope can capture images of celestial objects as faint as magnitude 30 in just one hour, equivalent to observing objects four billion times fainter than those visible to the naked eye.

The upcoming endeavor following the VLT is the construction of the Extremely Large Telescope (ELT), featuring a 39-meter primary mirror. Positioned as "the world's biggest eye on the sky", this telescope will hold the title of the largest optical/near-infrared telescope globally. The ELT aims to tackle numerous unresolved questions in astronomy, potentially revolutionizing our understanding of the universe.
Given the green light for construction in late 2014, the ELT is slated to achieve its first light for scientific observations by 2027.

Although there are multiple projects and astronomy instruments at the European Southern Observatory worth discussing for their functionality and achievements, we will focus on the VLT in Paranal and its future version, the ELT. This choice is because the initial focus of this study was on the VLT, with a potential extension to include the ELT.

At Paranal Observatory there are four 8 m telescopes. Most of the time at the telescopes, they are operating in "Service Mode" where they have a database of valid observations to execute. At any given time t, they select the "optimal" observation to execute. This selection depends on a lot of conditions like target coordinates, Moon phase, scientific ranking, and others. Additionally, this depends on atmospheric conditions. Each observation has "constraints" that describe under which atmospheric conditions the observation should be executed. After the observation has been completed, a quality control assessment is achieved to decide if the observations were successful or not. If the atmospheric conditions evolve to be out of constraints during the observation, then the observation may be repeated, which is considered a loss of telescope time.
The main atmospheric parameter that is used that varies the most is the "seeing" parameter which is related to the optical turbulence of the atmosphere.
Having accurate predictions for the seeing on hour timescales would be extremely useful to improve the selection of the optimal observation for execution.

Currently, the model utilized for predicting atmospheric seeing at the European Southern Observatory is denoted as the "Persistence Model" (PM), also sometimes referred to as the "Stationary Model" or "precast". It operates by calculating a 10-minute average for the observed seeing conditions, with the assumption that the seeing will remain consistent at this average level for a duration of one hour.
Efforts have been made to create a new model aimed at surpassing the performance of the stationary model, however, the outcomes have thus far not yielded optimal results.
The primary focus of these models were the utilization of Machine Learning (ML), Deep Learning (DL), and numerical simulations for atmospheric modeling. Furthermore, alternative endeavors at different observatories involved the exploration of hybrid models, characterized by the integration of numerical systems with machine learning methodologies to forecast atmospheric seeing conditions or, even better, optical turbulence parameters.

1.2. Astronomical Turbulence

Atmospheric optical conditions are crucial for selecting the best astronomical sites of the world. Among them, the seeing is one of the most important because is directly related with the spatial resolution of images taken by the observatories in the world.
Astronomical seeing is one of the optical turbulence variables along with isoplanatic angle and coherence time that is caused by the Earth atmospheric turbulence. Thermal convection and winds produce turbulence cells having different optical refraction indexes, leading to perturbations and distortions of the incoming light wavefronts.
This behaviour of the atmosphere is seen at the eyepiece as a blurred, moving, or scintillating image. There are roughly 3 main areas where atmospheric turbulence occurs. Near ground (0 – 100 m approximately) central troposphere (100 m – 2 km), and High troposphere (6 - 12 km). These altitude values can be really different in reality depending on the location. Details are explained below.

1. Near ground turbulence.

The air closest to the ground hosts the majority of atmospheric turbulence. This turbulence primarily comes from variations in heat radiation from different density areas, such as buildings, leading to localized convection currents. During the day, the Sun heats the ground, which in turn radiates heat at night, contributing to these currents. Observing over consistent topography is advantageous, as they release stored heat more gradually and uniformly, minimizing turbulence effects.

2. Central troposphere turbulence.

   The turbulence experienced at these altitudes is predominantly influenced by the topography situated upwind of the observing location. When airflow encounters a mountain peak, turbulence is induced, forming turbulent eddies that lead to shimmering or scintillating images. This phenomenon can persist for up to 100 kilometers downwind from the peak. Therefore, it is advisable to select observation sites where prevailing winds have traversed over consistent terrain. This facilitates the establishment of a laminar flow, resulting in stable and clear images.

3. High troposphere turbulence.

   At this altitude, the effects are primarily driven by swiftly moving "rivers" of air called jet streams. Wind shears occurring around the 200-300 mb altitude level can result in images appearing stable yet notably blurred, lacking intricate details. Unfortunately, observers have limited means to counteract these effects. However, forecasts are possible to predict whether a jet stream is positioned over the observation area, offering valuable insight for planning astronomical activities.

A rigorous quantification of the seeing effect depends on the exposure time. For most astronomical observations, seeing is quantified for the so-called long exposure case, in which the exposure is longer than the time in which the wavefront phase inhomogeneities larger than the telescope pupil pass through it. In practice for a large telescope this is an exposure of a duration of the order of 10 to 30 seconds. As a consequence the image motion effect of seeing will be summed-up in an overall blur effect.

The seeing conditions from a mathematical point are described by two fundamental parameters t_0 and r_0 in the Kolmogorov's turbulence theory. Both t_0 and r_0 parameters are a function of the wavelength of the light used for imaging.
t_0 is known as the coherence time and it is equals to the time interval over which the rms wavefront error due to the turbulence is 1 radian, or $\lambda/(2\pi)$. In short this is the time over which the image of a star can be considered frozen.

It's important to note that the quality of seeing isn't consistent across the entire field of view (FOV) of a telescope. This occurs because light from two celestial objects might travel slightly different paths through the atmospheric turbulence of Earth. When objects are separated by a "large" angle, the wavefront distortion affecting one object won't necessarily apply to the other, as their turbulence-induced errors differ, leading to a lack of correlation. The angle at which wavefront errors are nearly identical is termed the isoplanatic angle, a crucial parameter in understanding atmospheric turbulence effects. Figure 1.1 shows an illustration of this behavior.

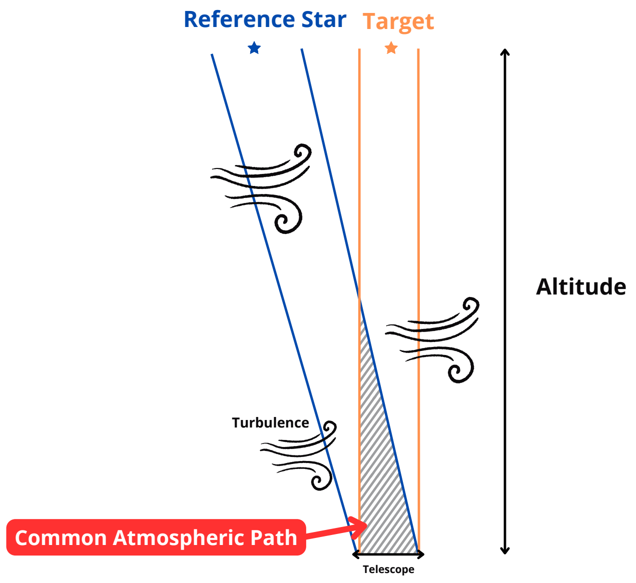


Figure 1.1: Illustration of common atmospheric path.

The term r_0, also known as Fried's parameter or Fried's coherence length, quantifies the quality of optical transmission through the atmosphere, attributable to random inhomogeneities in its refractive index. Measured in centimeters, r_0 represents the diameter of a circular area where the rms wavefront error induced by atmospheric passage equals 1 radian or $\lambda/(2\pi)$. Essentially, r_0 is the coherence length from the all atmosphere contribution and can be seen as the average turbulence cell size.

For practical applications in calculating the seeing, equations 1.1 and 1.2 (equations (13) and (14) in [1]) can be used. Here, σ^2 denotes the variance of the differential image motion, with the subscripts l and t referring to longitudinal and transverse motion, respectively, both for $d \geq 2D$.

$$\sigma_l^2 = 2\lambda^2 r_0^{-5/3}(0.179D^{-1/3} - 0.0968d^{-1/3}) \tag{1.1}$$

$$\sigma_t^2 = 2\lambda^2 r_0^{-5/3}(0.179D^{-1/3} - 0.145d^{-1/3}) \tag{1.2}$$

Where D is the diameter of the scope apertures of the DIMM instrument and d is the is the distance between the center of the apertures. The variables σ_l and σ_t are measured using the DIMM instrument and $\lambda = 500$ nm (or 500^{-9} m) is the wavelength used to calculate the turbulence.
It is worth to note that every variable involved is these equations are measured in meters.

Finally, to obtain the seeing value we need to calculate the following:

$$\mathit{seeing}\ [\text{rad}] = 1.02 \cdot \frac{r_0}{\lambda} \tag{1.3}$$

To calculate the documented seeing (as well as the one in the Paranal Observatory database), we need to convert it to arcseconds, which is:

$$seeing\ [\text{arcsec}] = 1.02 \cdot \frac{r_0}{\lambda} \cdot \frac{360}{2\pi} \cdot 36600 \tag{1.4}$$

1.3. General Objective

This thesis aims to primarily concentrate on predicting the hourly average of atmospheric seeing utilizing a diverse array of machine learning techniques applied to the minute-by-minute data collected by ESO. We focus on the hourly average because most individual science observations last about one hour and thus this prediction matches the average duration of observations for which ESO conducts quality control on the measured seeing during the observation.
This problem presents multiple approaches for treatment. It can be approached as a regression problem, where the objective is to predict a continuous value in $\mathbb{R}_+$ utilizing all available data. This predicted value represents the anticipated average seeing for the upcoming hour. Conversely, the problem can be viewed as a classification problem, where the goal is to predict the range within which the average seeing will fall. Alternatively, the problem can be approached as a pure forecasting task, entailing the prediction of seeing values at regular intervals of time. Once an hour's worth of values is obtained, they can be averaged to derive the final prediction.
For this work, we will focus on regression and classification problems. This decision is based on the fact that forecasting requires significantly more computational resources due to its complexity. Additionally, forecasting data further into the future tends to introduce more noise due to the chaotic behavior of seeing. Moreover, for predictions for the real-time scheduling of observations ESO does not require a model capable of forecasting an entire night of seeing values; instead, they need a model that can be run a few times. In this context, a regression/classification approach is ideal.
While a forecast of turbulence values for the entire night would certainly be useful, it falls outside the scope of this work.

As we mentioned before, the primary objective of this thesis is to develop a new model capable of surpassing the performance of the existing stationary model, leveraging the dataset provided by ESO. Therefore, the aim is to explore each approach comprehensively in order to draw conclusions regarding their effectiveness.

The necessity for this research arises from the belief that there exists potential to enhance the prediction of average seeing beyond the current methodology of relying solely on a 10-minute average. Additionally, the recent surge in advancements in artificial intelligence serves as a motivating factor for the development of such models.
Another strong reason is the ongoing development of the ELT at Cerro Armazones. With this, importance of accurately predicting atmospheric seeing is becoming even more critical for optimizing operational decision-making. An inaccurate prediction can lead to significant losses in telescope time, personnel efforts, and, inevitably, financial resources.

1.4. Specific Objectives

To fulfill the general objectives, we also concentrated on specific tasks aimed at deriving the desired conclusions.The primary specific objectives of this thesis include:

1. Exploratory Data Analysis (EDA).
    a) Basic information about the database.
    b) Data distribution and plots.
    c) Correlation between features.
2. Performing Feature Engineering.
    a) Add time-based features.
    b) Add seeing-related columns representing different times.
    c) Remove less important variables.
3. Experimenting with various models and approaches.
    a) Try basic ML algorithms.
    b) Use basic and state-of-the-art DL models.
    c) Adjust the last two points for regression and classification problems.
4. Perform Hyperparameter tuning for models.
    a) Choose the critical hyperparameters to adjust their values.
    b) Use different techniques to get optimal values for these hyperparameters.
5. Provide interpretability of the optimal model.
    a) Calculate feature importance, if possible.
    b) Explain the model's output and its relation to the input data.
6. Give conclusions with the selection of an optimal model for implementation.
    a) Choose the best model and the best approach to this problem of seeing prediction with a optimize implementation.
    b) Provide a conclusion for each method explored in this study and discuss potential paths for future research.

The construction of these specific objectives is based on a typical machine learning problem, providing us with a flexible working path.

Before going directly into the data, we need to explore existing information about this problem and any previous attempts to predict future seeing. Following this, we will present a theoretical framework, making this thesis self-contained, covering the concept of seeing and other mathematical tools necessary for predicting values and evaluating different models. Then, we will discuss the database provided by ESO, which we will utilize to address this problem and we will talk about the modifications we believe are most effective for improving

model performance.
Finally, we will experiment with multiple models and assess their performance to draw a comprehensive conclusion on this topic.

All coding aspects were developed and deployed in Python, utilizing common libraries for data reading and preprocessing, as well as traditional machine learning features. For advanced deep learning models we used the tsai library [2]. For visualizations, we primarily used the Plotly[1] library.

Almost all code-related tasks were implemented using Google Colaboratory. The machine specifications include an Intel Xeon CPU with 2 virtual CPUs and 13 GB of RAM. Additionally, by changing the runtime type, we gain access to an NVIDIA Tesla T4 GPU with 16 GB of VRAM for a limited time.

[1] https://plotly.com/

Chapter 2

Literature Review

In this chapter, we will review some existing work of astronomical seeing forecasting and what tools can be used to do that. This is a difficult task so there has been a few attempts trying new things to outperform the previous results, but nowadays as we have more data, it is more common to see data-driven models.
In the next sections, we will discuss various approaches that can be employed to predict seeing values.

2.1. Prediction based on meteorological models

Since the start of weather and atmospheric parameter forecasting, numerous numerical models have been developed for this purpose. One such model is the non-hydrostatic mesoscale model (MESO-NH [3]), which is utilized by many observatories, often in modified versions, to predict various weather parameters. Another approach involves employing the Weather Research and Forecasting (WRF [4]) model. Currently, numerous methods are derived from Tatarskii theory [5].
Although meteorological models are a powerful tool for prediction tasks, they are beyond the scope of this work. However, as future work, incorporating these models as a potential tool for predictions could provide a strong and robust fusion for optical turbulence variables.

2.2. Prediction based on machine learning

Given the significant advancements in machine learning over recent years, there is a high motivation to explore these models for the task of seeing prediction. These attempts range from classical machine learning approaches to deep learning (DL) models including Multi-Layer Perceptron (MLP), Long Short-Term Memory (LSTM), Convolution Neural Networks (CNN) and even state-of-the-art models like MINIROCKET [6] and transformer based models like Time Series Transformer (TST [7]) that are based in attention modules [8] and more.
As evidenced in the literature [9–12], ML models offer a promising approach for forecasting and prediction challenges. Thus, the primary focus of this thesis will center on exploring ML methodologies, from classical to state-of-the-art algorithms and how efficiently they work, in terms of pure prediction error, training and prediction time as well as the difficulty of implementation.

2.3. Prediction based on hybrid models

We have discussed numerical and machine learning models independently, however, there is natural idea of integrating them into a unified model. Attempts to explore this concept can be found in [13].

At the time of writing, we couldn't find many hybrid models, as they may not be the primary approach for the forecasting task. The most common methods are to implement numerical and complex models or pure machine learning techniques.

Chapter 3

Theoretical Framework

3.1. Classic Machine Learning Models

Machine learning models are algorithms designed to predict the categorical class (classification) or a continuous value (regression) based on input features. They use a train data set to adjust or learn patterns within the data that enable accurate predictions.
Classic ML methods refer to models that are simpler than deep learning models and are typically easier to interpret, requiring fewer computational and time resources.
For simplicity, the next subsections will explain how the models work in a classification task. Minor changes are necessary to transform it to a regression problem.

3.1.1. Isotonic Regression

This model is only used for regression problems and fits a non-decreasing real function to 1-dimensional data. That being said, this is one of the simplest models we are going to test because this will only take into account the seeing parameter and will try to make a piecewise increasing linear function to make predictions.

The mathematical statement of the problem is, given n points of the form (x_i, y_i):

$$\begin{aligned} \min \ & \sum_{i=1}^{n} w_i (y_i - \hat{y}_i)^2 \\ \text{s.t. } & \hat{y}_i \leq \hat{y}_j \text{ for all } (i,j) \in \{(i,j) : x_i \leq x_j\} \\ & w_i > 0 \end{aligned} \tag{3.1}$$

3.1.2. Stochastic Gradient Descent

Stochastic Gradient Descent or SGD is, strictly speaking, a merely optimization technique and does not correspond to a ML model, but it can be an estimator depending on the parameters. For example, if we optimize the problem through the ordinary least squares fit and use a $l2$ regularization, this will be equivalent to a Ridge regression.
The reason to test this model is the amount of data that we will work on and the ease of implementation it offers.

For the mathematical formulation, we try to minimize a objective function with the form

$$Q(w) = \frac{1}{n}\sum_{i=1}^{n} Q_i(w) \tag{3.2}$$

where the parameter w that minimizes $Q(w)$ is to be estimated. Then, the true gradient of $Q(w)$ is approximated by the value $w := w - \eta \nabla Q_i(w)$ at a single sample. This is an iterative method so several steps will be made until the algorithm converges.
Here, η is called *learning rate* and is one of the most crucial parameters in this model as well as DL models.

3.1.3. Decision Tree

Decision Trees (DT) represent a non-parametric approach within supervised learning, serving as a method for both classification and regression tasks. It is important to note that it operates as a piecewise linear model, with each neighborhood defined in a non-linear way.
The process by which decision trees operate involves a data point $x = (x_1, x_2, x_3, \ldots, x_k, y)$ entering the tree and progressing to subsequent nodes based on certain feature values until it arrives at a leaf node (nodes that have no additional nodes coming off them), where a prediction is obtained.

Choosing the best attribute at each node is done by calculating the information gain. The way this is calculated is first calculating the entropy:

$$H(S) = -\sum_{c \in \mathcal{C}} p(c) \log_2 p(c) \tag{3.3}$$

Where S represents the data set, c represent a class in the dataset where $\mathcal{C}$ is the collection of all possible classes and $p(c)$ is the fraction of data points that belong to class c.
Then, the information gain is calculated by:

$$IG(S, a) = H(S) - \sum_{v \in vals(a)} \frac{|S_a(v)|}{|S|} H(S_v) \tag{3.4}$$

Where $vals(a)$ is the value of the a^{th} attribute and $S_a(v) = \{x \in S | x_a = v\}$ with x_a being the a^{th} attribute of example x.

Another important function to measure the quality of the splits is the Gini Impurity:

$$I_G = 1 - \sum_{c \in \mathcal{C}} p(c)^2 \tag{3.5}$$

This is the probability of misclassifying random data point in the dataset if it were labeled according to the class distribution of the dataset.

3.1.4. Ensemble models

Ensemble models in machine learning are techniques that combine the predictions from multiple individual models to produce a more accurate and robust final prediction. These individual models will be decision trees for all the ensemble models we consider in this study.
The idea behind ensemble learning is to leverage the diversity of multiple models to compensate for the weaknesses of individual models and improve overall performance. These

methods can address both classification and regression tasks and are among the most frequently utilized models due to their performance, efficiency in training and prediction, as well as their straightforward implementation.

There are four algorithms that we will focus on. These are *Random Forest* [14], *XGBoost* [15], *LightGBM* [16] and *CatBoost* [17]. These models are one of the most used in machine learning tasks due to their performance and the amount of hyperparameters that can be tuned to tackle the problem with a more suitable model.

3.2. Deep Learning Models

Deep learning models are a powerful class of artificial neural networks characterized by their deep architectures, often consisting of multiple layers of interconnected nodes. Unlike traditional machine learning algorithms, which rely on handcrafted features, deep learning models can learn intricate patterns and representations of data through the iterative process of forward and backward propagation.

3.2.1. Multilayer Perceptron

A multilayer perceptron (MLP) is a type of feedforward artificial neural network that consists of multiple layers of interconnected nodes, or neurons. It is one of the simplest and most common architectures in deep learning. In an MLP, each neuron in one layer is connected to every neuron in the subsequent layer, forming a dense network of connections. The layers in an MLP typically include an input layer, one or more hidden layers, and an output layer.
The input layer receives the raw input data, which is then passed through the network. Each neuron in the hidden layers applies a weighted sum of the inputs, followed by an activation function to produce an output. This output is then passed as input to the next layer. The final output layer produces the model's prediction.

We'll denote x_j to the input units and the activation of the output unit as y. The units in the l-th hidden layer will be denoted $h_i^{(l)}$ and L the numbers of layers. Therefore, the network's model can be expressed as:

$$\begin{aligned}
h_i^{(1)} &= \phi^{(1)}\left(\sum_j w_{ij}^{(1)} x_j + b_i^{(1)}\right) \\
h_i^{(2)} &= \phi^{(2)}\left(\sum_j w_{ij}^{(2)} h_j^{(1)} + b_i^{(2)}\right) \\
&\vdots \\
y_i &= \phi^{(L)}\left(\sum_j w_{ij}^{(L)} h_j^{(L-1)} + b_i^{(L)}\right)
\end{aligned} \tag{3.6}$$

We differentiate between $\phi^{(1)}$ and $\phi^{(2)}$ to account for the possibility of different activation functions across various layers. The equations shown in (3.6) can be reformulated in a vectorized form, incorporating an activation vector $\mathbf{h}^{(l)}$, a weight matrix $\mathbf{W}^{(l)}$, and a bias vector $\mathbf{b}^{(l)}$ giving the following equations:

$$
\begin{aligned}
\mathbf{h}^{(1)} &= \phi^{(1)}\Big(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}\Big) \\
\mathbf{h}^{(2)} &= \phi^{(2)}\Big(\mathbf{W}^{(2)}\mathbf{h}^{(1)} + \mathbf{b}^{(2)}\Big) \\
&\vdots \\
\mathbf{y} &= \phi^{(L)}\Big(\mathbf{W}^{(L)}\mathbf{h}^{(L-1)} + \mathbf{b}^{(L)}\Big)
\end{aligned}
\tag{3.7}
$$

3.2.2. Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are another class of artificial neural networks designed to handle sequential data by introducing connections between units within the network that form directed cycles. Unlike a MLP, which process data in a single direction, RNNs have connections that loop back on themselves, allowing them to exhibit temporal dynamics and capture dependencies over time.

In an RNN, each unit, or neuron, maintains an internal state that represents information from previous time steps. This internal state, often referred to as the “memory” of the network, is updated at each time step based on both the current input and the previous state. This recurrent structure enables RNNs to model sequences of arbitrary length and capture patterns in sequential data.
The Figure (3.1) represents a general RNN and we can see their chain-like nature that can represents such type of data.

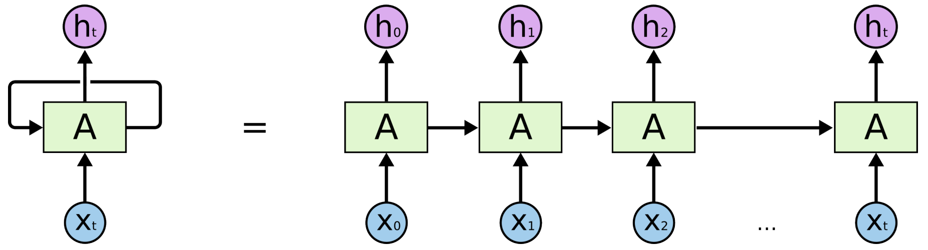


Figure 3.1: RNN diagram (image from [18])

Traditional RNNs can struggle with capturing long-term dependencies in sequences due to (probably) the main problem which is vanishing gradients, where the gradients used to update the network’s parameters vanish exponentially over time. To address this limitation, several variants of RNNs have been developed, such as Long Short-Term Memory (LSTM [19]) networks and Gated Recurrent Units (GRU [20]). These architectures incorporate mechanisms to better retain and update information over long sequences, making them more effective for tasks requiring modeling of long-range dependencies.

3.2.3. Convolutional Neural Networks

Convolutional Neural Networks (CNNs [21]) are another class of artificial neural networks designed originally to handle structured grid-like data, such as images or audio spectrograms. However, it is usually a good method to try in regression/classification tasks due to their (usually) lower training time.

CNNs are characterized by their architecture, which includes convolutional layers, pooling layers, and fully connected layers. Convolutional layers are the core building blocks of CNNs and consist of filters, also known as kernels, that slide over the input data to perform convolutions. These convolutions can capture local patterns or features. By stacking multiple convolutional layers, CNNs can learn increasingly complex and abstract features from raw input data.

Pooling layers are situated between convolutional layers and serve to downsample the feature maps produced by the convolutions. Common pooling operations include max pooling and average pooling, which reduce the spatial dimensions of the feature maps while preserving important features.

Fully connected layers are typically located at the end of the network and are responsible for combining the high-level features learned by the convolutional layers to make final predictions.
Figure (3.2) represents a simple CNN model. Usually, the net is more interconnected but it serves as an illustration.

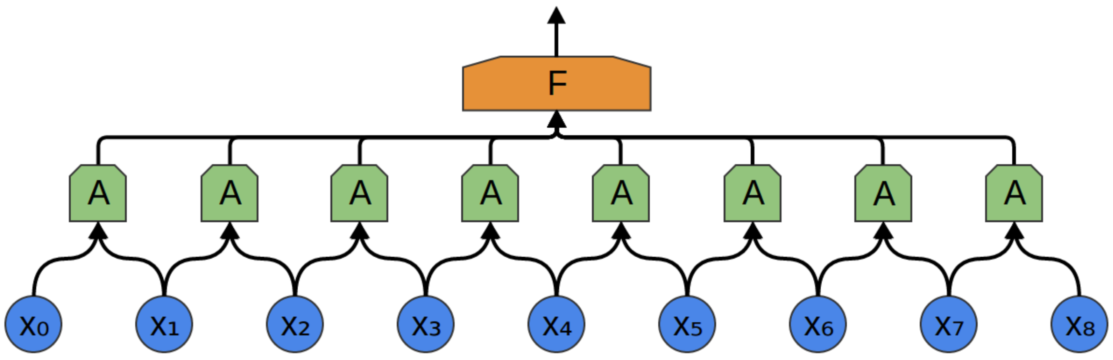


Figure 3.2: CNN diagram (image from [22])

3.3. Metrics

3.3.1. Mean Absolute Error

Mean Absolute Error (MAE) is a metric that measures the average absolute difference between predicted values and actual values in a dataset. It reflects the magnitude of errors without considering their direction, making it easy to interpret as the average error size.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |\hat{y}_i - y_i| \tag{3.8}$$

Where y_i is the true value and $\hat{y}_i$ the prediction. MAE is measured in the same units as the target variable, making it easy to interpret. A lower MAE indicates better model performance, as it reflects smaller average errors between predictions and actual outcomes.

3.3.2. Mean Squared Error and Root Mean Squared Error

Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) are common metrics used to evaluate the performance of regression models. They measure the average of the squared differences between the predicted values and the actual values in a dataset and an average magnitude of the errors between the predicted values and the actual values, respectively.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \tag{3.9}$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \tag{3.10}$$

MSE is measured in the square of the units of the target variable while RMSE is measured in the same units as the target variable, which makes it more interpretable. Both of them indicates a better performance the lower the value.
Minimizing RMSE (or MSE) will focus more on reducing large errors, while minimizing MAE will focus on reducing the average error.

3.3.3. Precision

Precision is important metric used to evaluate the performance of classification models. It measures the proportion of correctly predicted positive cases out of all cases that are predicted as positive.

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}} \tag{3.11}$$

Precision ranges between 0 and 1, where a higher value represents a better performance of the model.

3.3.4. Recall

Recall is another important metric used to evaluate the performance of classification models. It measures the proportion of correctly predicted positive cases out of all actual positive cases.

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}} \tag{3.12}$$

Recall ranges between 0 and 1, where a higher value represents a better performance.

3.3.5. Accuracy

Accuracy is a fundamental metric used to evaluate the overall performance of classification models. It measures the proportion of correctly predicted instances (both positive and negative) out of all instances in the dataset.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{3.13}$$

Accuracy also ranges between 0 and 1, where a higher value represents a better performance.

3.3.6. f1-score

The f1-score is a metric commonly used to evaluate the performance of classification models, especially in scenarios where class imbalance is present. It is a harmonic mean of precision and recall, providing a single score that balances both metrics.

$$\text{f1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FP} + \text{FN}} \tag{3.14}$$

f1-score also ranges between 0 and 1, where a higher value represents a better performance.

3.3.7. Diagonal Average

The Diagonal Average metric involves segmenting predictions into predefined bins, in this case, $(0.15, 0.5], (0.5, 0.6], (0.6, 0.7], (0.7, 0.8], (0.8, 1.0], (1.0, 1.3], (1.3, 2.2]$ and $(2.2, +\infty)$, and then visualizing them in a confusion matrix to compare predictions against true values. Then by normalizing the matrix over the rows (representing true values) and computing the diagonal average, that is, the average recall in classification problems, we obtain model performance. Note that a higher Diagonal Average signifies better model performance.

If we have a square matrix A with size n, then,

$$\text{Diagonal Average} = \frac{1}{n} \sum_{i=1}^{n} A_{ii} \tag{3.15}$$

3.3.8. Success Rate

The Success Rate metric refers to fulfilling observation seeing prerequisites. For example, if an observation requires an average visibility within the range of 0.5 to 0.6 for an hour (average), the criteria are considered met even if the actual visibility falls below 0.5 (better seeing). Consequently, this metric does not penalize predictions exceeding the actual values. It's worth to note that this metric primarily assesses model performance in the lower visibility ranges (below 1.0). This is because if we employed this metric as a model loss function, predictions would result in the model favoring exceedingly high values because then every prediction will be higher than the actual value and will score perfectly based on the Success Rate.

If we have a square matrix A with size n and a total of N observations, then one way to obtain this metric is,

$$\text{Success Rate} = 1 - \frac{1}{N} \sum_{j<i}^{n} A_{ij} \tag{3.16}$$

Chapter 4

Database

The thesis utilizes a database[2] provided by the European Southern Observatory, with various time periods (p97 to p111) spanning six months each, from April 6, 2016, to September 30, 2023. This dataset has already been cleaned by the ESO, putting all measurements onto a common time series, removing null values and any unnatural outliers. Additionally, ESO performed preprocessing to ensure there were no gaps in the data except for the transitions between nights. That is, the only gaps in the data occur when one night ends and the next measurement by the instruments is taken the following night, so there are no gaps between rows of data within the same night.

The train database (p97 to p109) contains a total of 1 174 104 rows and 168 columns. Three columns have been excluded from the dataset due to redundancy, as they contain duplicate information presented in a less optimal format compared to other columns so the total of useful columns is 165.
The provided data is captured at one-minute intervals, with each row representing a distinct minute. Data collection typically spans from around 22:00/23:00 to 8:00/9:00 UTC (Universal Time), varying slightly depending on the season.
It is worth to note that there is no data at day time so this is considered a discontinuous time series. This fact makes it really hard to use some state-of-the-art models that require a fixed frequency in the data directly. Additionally, in the future, as the ELT is being constructed, there are plans to use instruments that can measure seeing 24 hours a day, providing a better and more detailed time series for these optical turbulence variables.

On the other hand, the test set has the same qualities with the only difference that there are 199 314 rows and it is a entire year (p110 and p111), from 2022-10-01 to 2023-09-30.

The initial memory usage of the database was 1812.912 MB, before optimizing certain data types for efficiency. Following the optimization process, the database's memory usage decreases significantly to 895.469 MB. These modifications were implemented to minimize memory usage, particularly for features where high precision in measurements was unnecessary.

[2] https://www.eso.org/sci/facilities/paranal/astroclimate/ASMDatabase.html

4.1. Features Explanation

4.1.1. Target Variables

As previously mentioned, the dataset include 165 pertinent features utilized in this study. Among these, three are designated as target variables, indicating the focus of prediction efforts. These features, namely “targetSeeing1hr”, “targetMin1hr”, and “targetMax1hr” represent the average, minimum, and maximum values of seeing measurements, respectively, taken one hour after the initial observations. For example, at 1:30 AM, each column will represent data at 1:30 AM, such as wind speed, absolute humidity, temperature, etc. The only exceptions are the target variables. Therefore, targetSeeing1hr will represent the average seeing from 1:31 to 2:30 AM, and the same applies to the other two target variables. These targets are all float numbers so a regression model is the first that comes to mind to try and predict these values.

The variable “targetSeeing1hr” likely holds the most significance of the three as it encapsulates the average behavior of seeing, which is closely related to the stationary model employed at ESO because that is what they are trying to predict, as this is used for quality control of the observations (whether an observation is successful or not). Nevertheless, both the minimum and maximum values provide valuable insights, offering perspectives on the best and worst-case scenarios for the seeing values.

In Figure 4.1, we can observe an example of three consecutive nights and their target values. Note how the targetMin1hr and targetMax1hr variables exhibit a few jumps throughout the night. This is because the seeing value can be quite chaotic, experiencing significant lows and highs, which directly impacts these features. However, the targetSeeing1hr variable is much smoother. By definition, this variable is an average of 60 seeing values, so it makes sense that it doesn't exhibit sudden jumps. The large gaps in the center represent periods where no data is available (during daytime).

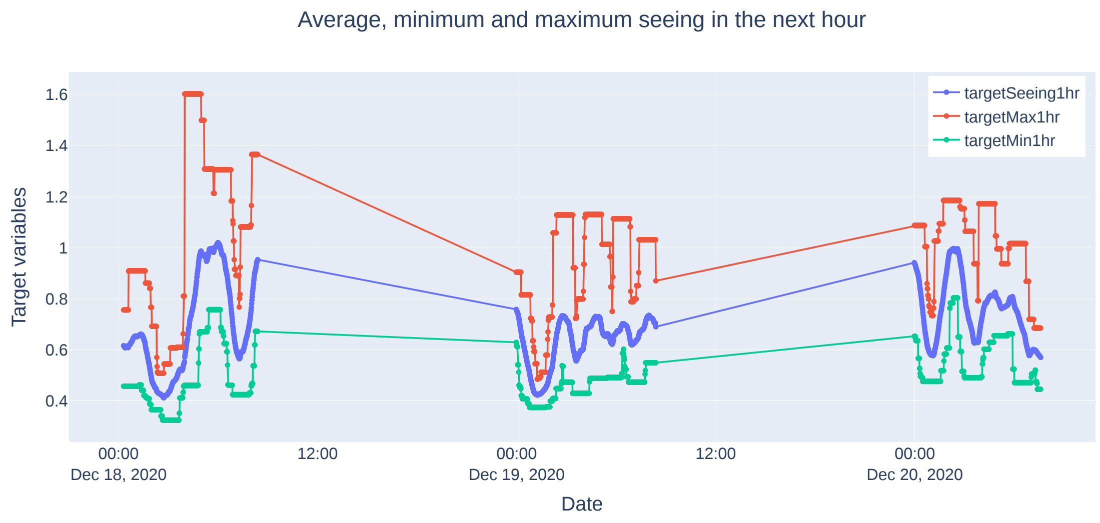


Figure 4.1: Target columns from December 18th to 20th, 2020.

4.1.2. Input Variables

For constructing a regression or classification model, it's imperative to have input variables that the model can interpret. Fortunately, all columns in the dataset are of numeric types, eliminating any potential issues in this regard. With that said, we proceed to explain the 162 features in general terms.

Initially, the dataset includes a date column representing values in the format of year-month-day hour:minutes:seconds. Subsequently, it encompasses various Differential Image Motion Monitor (DIMM) variables, absolute and relative humidity measurements at different altitudes as well as temperature at these heights. Additionally, Multi-Aperture Scintillation Sensor (MASS) variables with some variables from MASS-DIMM combined profiles, as well as wind direction and speed measurements at varying altitudes, are included.
The specific variable names are provided in Annex A.1. For detailed definitions of each variable, please refer to the page Paranal Ambient Query Forms.

In Figures 4.2, 4.3, and 4.4, we can see some of the variables present in the database, including absolute humidity, relative humidity, and temperature. These variables are measured at different heights: 0, 10, 30, 50, 75, 100, 125, 150, 200, 250, 325, 400, 475, 550, 625, 700, 800, 900, 1000, 1150, 1300, 1450, 1600, 1800, 2000, 2200, 2500, 2800, 3100, 3500, 3900, 4400, 5000, 5600, 6200, 7000, 8000, 9000, and 10 000 meters. In total, we have 39 columns for each parameter, resulting in a total of 117 features related to humidity and temperature.

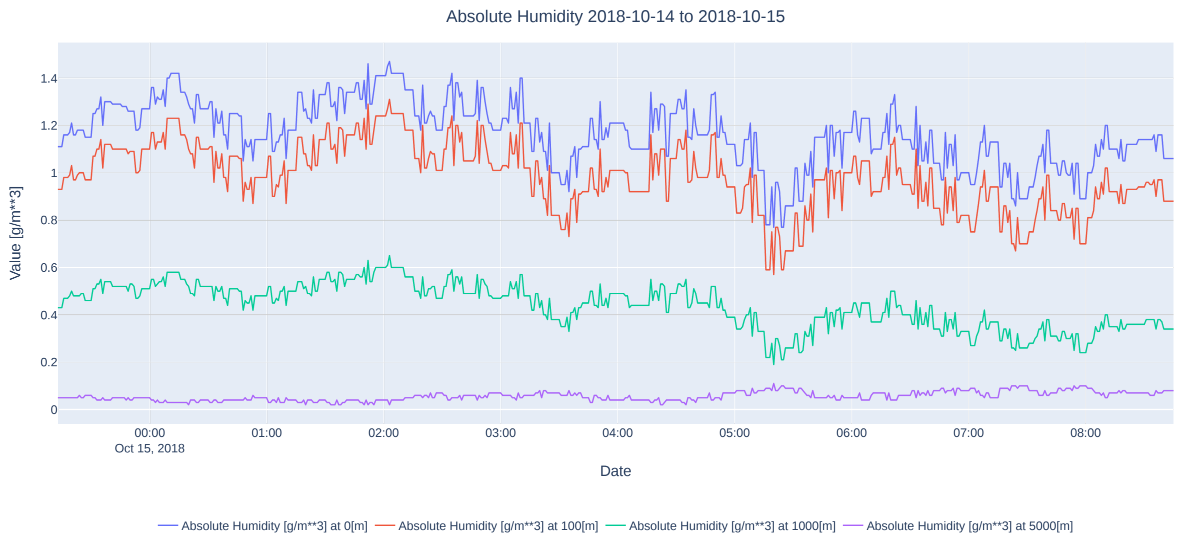


Figure 4.2: Absolute humidity at different heights from October 15th, 2018.

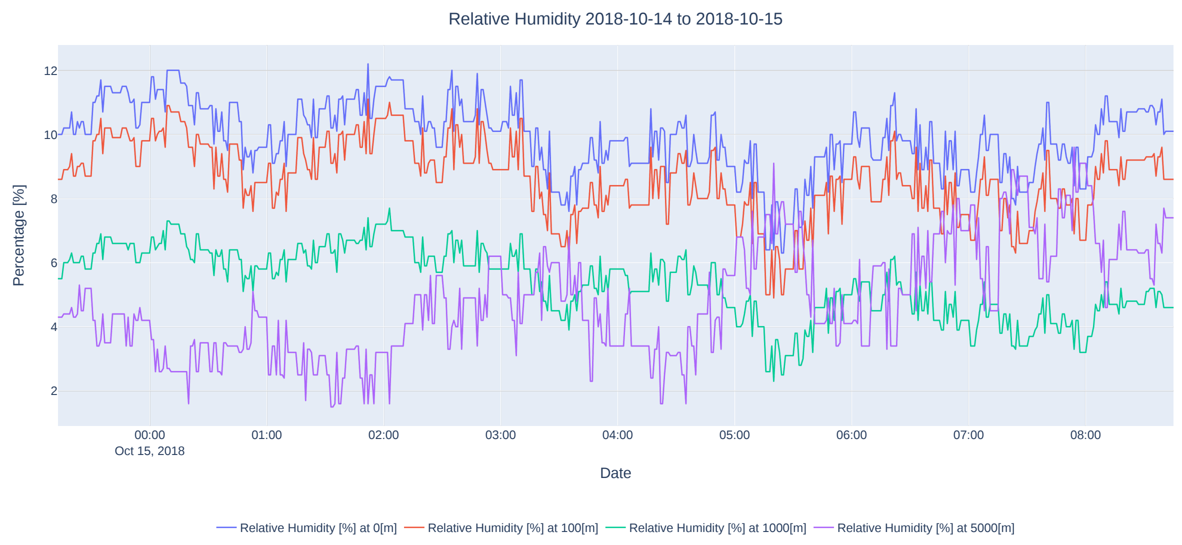


Figure 4.3: Relative humidity at different heights from October 15th, 2018.

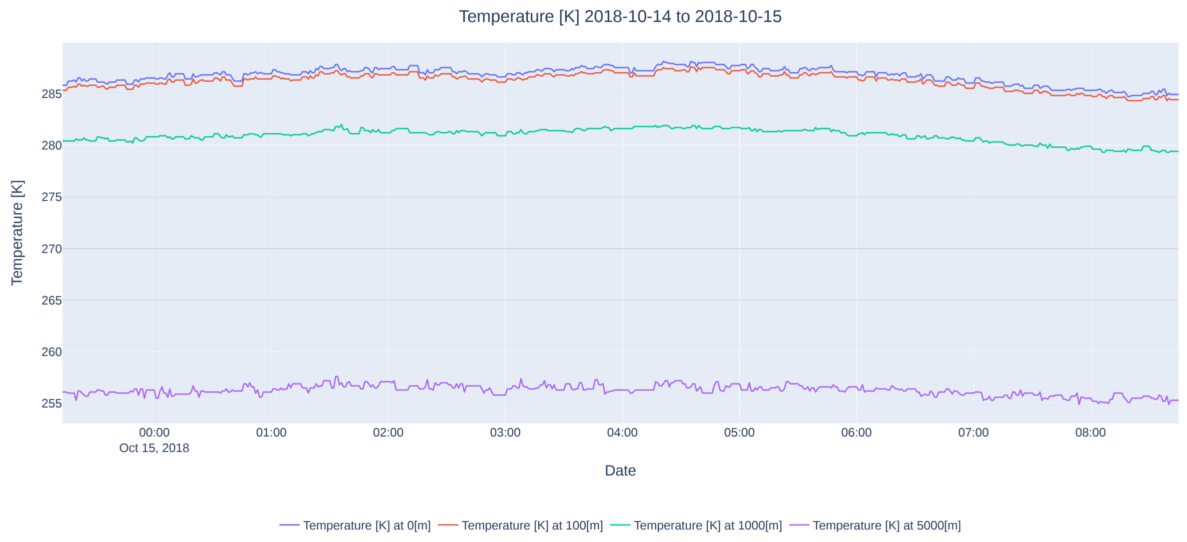


Figure 4.4: Temperature at different heights from October 15th, 2018.

Now, the next plot shows the main input feature among all of these: the DIMM Seeing [arcsec]. This column indicates the seeing value at any given time during the night. As mentioned before, this optical turbulence variable is very chaotic and can experience sudden jumps. Figure 4.5 reveals the seeing measurements for one night.

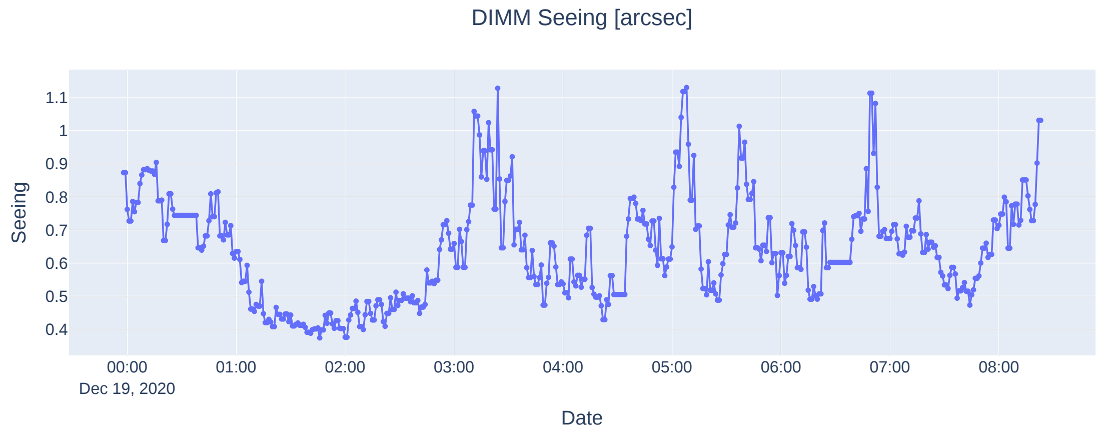


Figure 4.5: DIMM Seeing in arcsec from December 19th, 2020.

Next, Figure 4.6 shows the distribution of this variable. For visualization purposes, the X-axis is limited to 2.5 arcsec. Higher values than this are not very common, and when we look at the target variable (average of seeing), these values are even more uncommon.

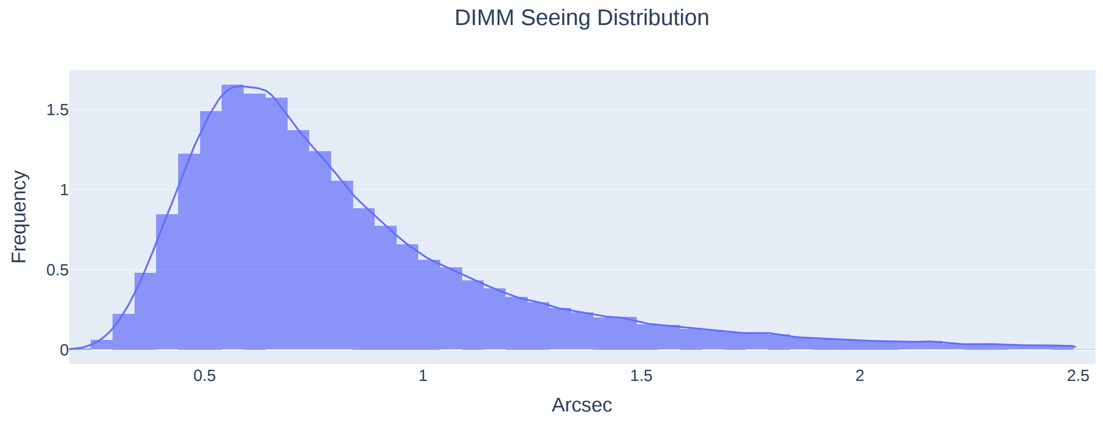


Figure 4.6: Distribution plot for DIMM Seeing [arcsec]

In order to enable the model to predict these target variables effectively, significant modifications will be made to these columns. These adjustments will involve the removal of less significant features, capturing the periodic nature of time-related features, and generating features related to seeing. Further elaboration on these modifications will be provided in the next chapter.

Chapter 5

Methodology

The methodology utilized in this research is founded upon a classical data science approach known as CRISP-DM (Cross-Industry Standard Process for Data Mining). It entails a systematic progression beginning with a thorough comprehension of the problem and objectives, followed by an examination of the available data. Then, feature engineering, modeling, and evaluation of the results are conducted. While the final step typically involves deployment, it is noteworthy that the European Southern Observatory already has an operational model. Altering this model would affect numerous internal areas within the observatory, so deployment will not be the primary focus of this process. However, we will run virtual simulations to compare different prediction models, meaning they will be "deployed" in an operational environment in some sense.
This entire methodology operates in a cyclical manner, where data is prepared in different ways, numerous models are tested and evaluated using various metrics, iteratively, until the desired result is achieved.

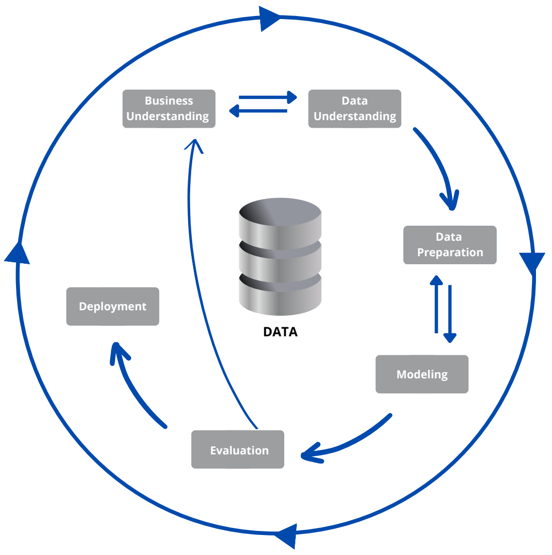


Figure 5.1: CRISP-DM methodology for data science projects.

To fulfill the initial objectives, we will first conduct an exploratory data analysis. This step involves understanding the data in detail. We will achieve this by providing numeric descriptions of the main features and creating graphs to make the data more visually understandable. Next, we will study the correlation between features and target variables. Additionally, we will analyze the lag plot of the seeing values to determine if there is a correlation with past values, which will help us conclude if a machine learning approach to this problem is suitable.

After understanding the data in detail, the next step is the feature engineering process. This involves making modifications to the database to improve the model's predictive performance. There are two main reasons for this step:

1. Removing less useful features, such as highly correlated characteristics or those with low importance (identified by different algorithms), and intelligently creating new ones can positively impact the model's performance by reducing error.
2. Having fewer features improves the model's efficiency, which is important if we want to train the model multiple times with validation datasets and perform a hyperparameter search.

By refining the dataset, we can enhance both the accuracy and the training efficiency of the model.

The feature engineering process was a primary focus of this thesis project. This is because this step can significantly improve a model's performance and largely depends on one's creativity in generating new features. Therefore, this step involved numerous modifications to compare the results of different models and continually add more columns to the dataset.

After completing all the previous steps, it's time to try out some models and assess their performance on this task. From the start, this problem was treated as a regression task, so that's our first attempt. One of the challenges we encountered is dealing with multiple metrics that are important for each model, making it more difficult to select the optimal one. As mentioned before, this is an iterative process, so the models presented in this thesis are the final ones selected based on all metrics and time performance.
The next step was to try classification models and evaluate how well they adapt to this problem. This will help us conclude which type of model to select as the optimal one.

The idea of selecting a few optimal models is to allocate more time for refining these models and adapting them more effectively. Therefore, a hyperparameter search will be conducted for these selected models. Then, final comparisons will be made between these fine-tuned models, which will include their performance on the test set.

To evaluate each presented model, we will partition the data into training, validation, and test sets. Note that there exists a test set, but we will not use it until we have selected the best models. Therefore, for this selection, we will only focus on the validation set performance and their average. Since this is a time series problem, the train and validation sets will be created using the TimeSeriesSplit[3] function with a number of splits = 5 and a gap of 1000

[3] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.TimeSeriesSplit

data points, ensuring that there is a gap of approximately two days between each split. That gives the following cross validation splits, where each number is a different evaluation so we have a new train and validation sets:

1. Train index = $[0, 194\,683]$, validation index = $[195\,684, 391\,364]$
2. Train index = $[0, 390\,364]$, validation index = $[391\,365, 587\,045]$
3. Train index = $[0, 586\,045]$, validation index = $[587\,046, 782\,726]$
4. Train index = $[0, 781\,726]$, validation index = $[782\,727, 978\,407]$
5. Train index = $[0, 977\,407]$, validation index = $[978\,408, 1\,174\,088]$

In conclusion, the above represents a time series scenario where a model is trained with some of the data and validated with future points that the model hasn't seen. Note that the number of training points is increasing, so one might expect better performance with each split. However, due to the chaotic nature of the target variable, this may not always be the case.

5.1. Exploratory Data Analysis

First, we will examine the data from numerical perspective. Then we will plot some variables and observe how they behave over time.

Here we can see Table 5.1 and Table 5.2 with statistical information about the target variables. They show the mean, minimum and maximum values with the standard deviation and the 25th, 50th, and 75th percentiles.
These tables were created separately to determine if the training and test datasets exhibit similar target variables. If big discrepancies exist, we would need to clean the data and attempt to align the distribution of these target variables between the training and test set.

Table 5.1: Numeric description of target variables for training set.

	targetSeeing1hr	**targetMin1hr**	**targetMax1hr**
mean	0.855	0.647	1.178
min	0.225	0.175	0.254
25%	0.579	0.442	0.773
50%	0.727	0.550	1.003
75%	0.988	0.738	1.393
max	4.827	4.202	7.530
std	0.432	0.335	0.615

Table 5.2: Numeric description of target variables for test set.

	targetSeeing1hr	**targetMin1hr**	**targetMax1hr**
mean	0.843	0.640	1.154
min	0.264	0.228	0.295
25%	0.594	0.454	0.786
50%	0.751	0.565	1.023
75%	0.986	0.743	1.377
max	3.439	2.723	5.913
std	0.371	0.286	0.545

We can observe that the two sets are quite similar, despite the difference in the amount of data in each set (approximately a 6:1 ratio for the train and test sets). The most significant disparity may lie in the maximum values of the three target variables. However, since our primary concern is with the lower values of seeing, this discrepancy should not be an issue for the models that we will evaluate.

Now that we have identified several characteristics of the target variables, we will examine their distribution. The bin ranges for the histogram plot is the standard practice at ESO, and we will maintain these values consistently throughout the study.

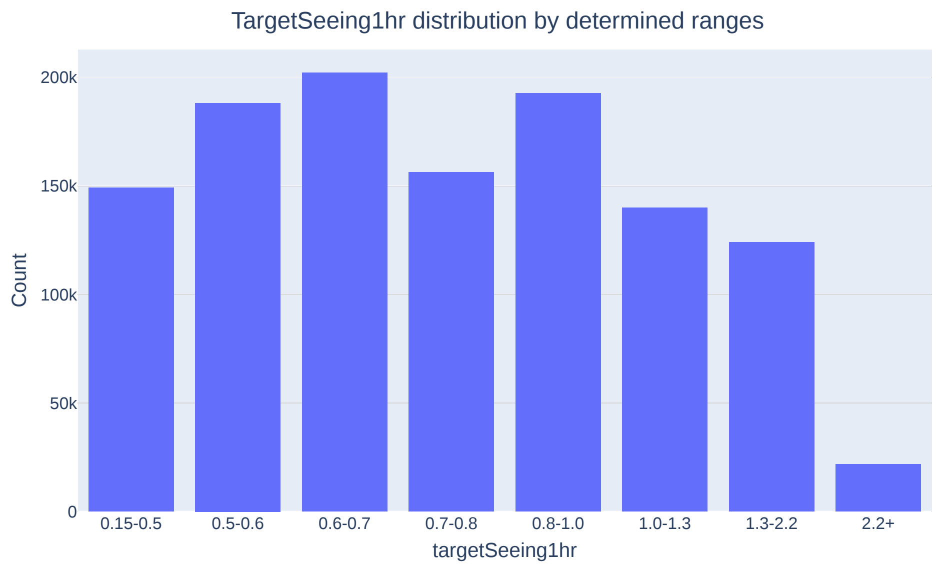


Figure 5.2: Histogram of the values of "targetSeeing1hr" data.

Figure 5.2 provides an approximate distribution of the average seeing values. We observe that each range of values is fairly similar in quantity, except for the last one (2.2+). This disparity is not concerning, as this range is of lesser importance to us and to the ESO, because almost every observation requires a better visibility. Consequently, we will not focus on using undersampling or oversampling techniques. However, we will explore models that inherently address class imbalances, such as the Balanced Random Forest.

Next, we will observe similar histograms for the other two target variables:

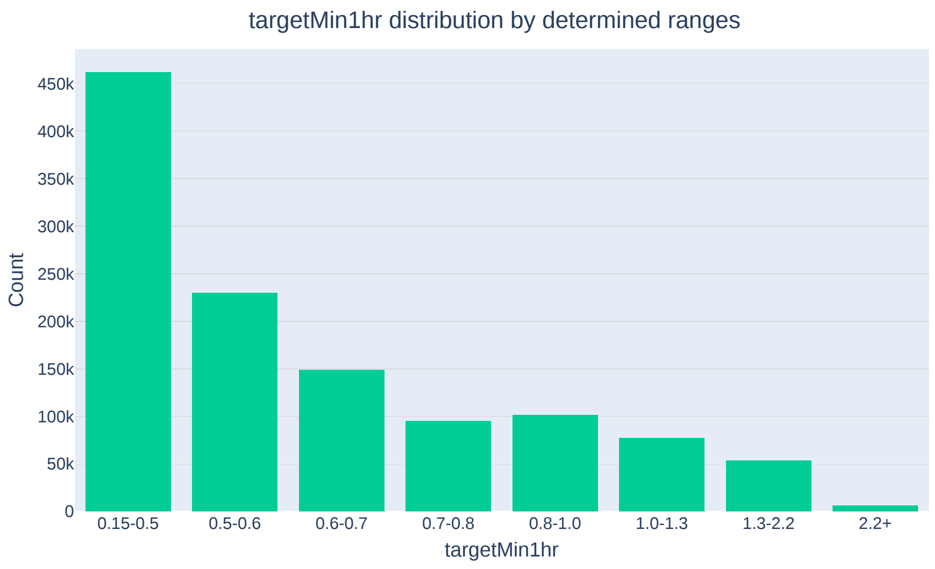


Figure 5.3: Histogram of the values of “targetMin1hr” data.

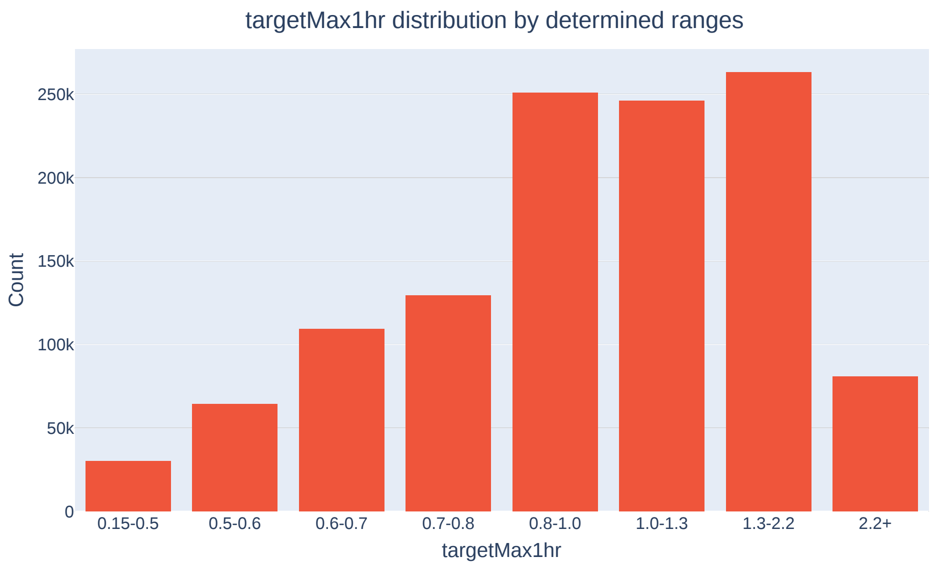


Figure 5.4: Histogram of the values of “targetMax1hr” data.

Figures 5.3 and 5.4 show the distribution of data across bins. In both cases, instances of seeing values surpassing 2.2 are infrequent. Notably, the range (0.15, 0.5] contains fewer data points for the targetMax1hr variable. This discrepancy arises because if, for any minute, the seeing value exceeds 0.5 (which is plausible considering the nature of seeing), the target-Max1hr value will shift to a higher bin.
In the case of the targetMin1hr variable, it is evident that the majority of data points fall within the (0.15, 0.5] range. Therefore, if we intend to employ a classification model, we must ensure that the model does not only predict this “class” as a result of a naive prediction.

Continuing in the same direction of visualizing the data, Figure 5.5 presents a (Pearson) correlation plot of all available features using a random subset of 100 000 points from the training set, a measure taken due to time and computational resource constraints.

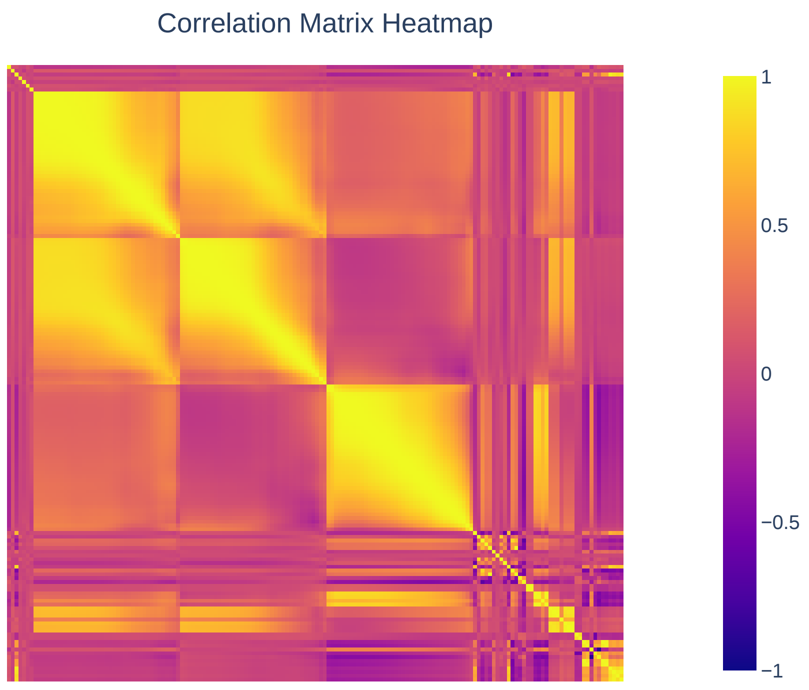


Figure 5.5: Pearson correlation of all features for a sample of 100 000 data points.

The big 3×3 block that we observe in the heatmap is caused by 3 variables taken at different heights. These variables are absolute humidity [g/m^3], relative humidity [%] and temperature [K]. The altitude that these measures were taken are again 0, 10, 30, 50, 75, 100, 125, 150, 200, 250, 325, 400, 475, 550, 625, 700, 800, 900, 1000, 1150, 1300, 1450, 1600, 1800, 2000, 2200, 2500, 2800, 3100, 3500, 3900, 4400, 5000, 5600, 6200, 7000, 8000, 9000 and 10 000 meters. Therefore, each square of this 3×3 block is a 39×39 grid, where the upper-left squares represent lower altitudes and the bottom-right squares represent higher altitudes.
As we can see, these values are highly correlated even at different heights so we can do a lot of modifications in order to reduce the number of variables, training and prediction time of the models.
The smaller yellow squares located in the bottom-right corner represent the target variables, as well as the predictions generated by the existing model at ESO, alongside additional measurements of wind, temperature, and humidity obtained from another sensor, in this case, LHATPRO.

The final visualization for this EDA section can be observed in Figure 5.6, showing lag plots at various time intervals. These graphs are used to investigate whether the target variable we aim to predict exhibits any correlation with itself over different time shifts. Included are lag plots at intervals of 1, 15, 30, 60, 90, and 120 minutes.
For computational reasons, the plots are generated using a subset of 100 000 data points. This subset of data was taken in such a way that it is not shuffled, meaning, 2 indices were selected, and all corresponding elements between these 2 selected rows were taken.

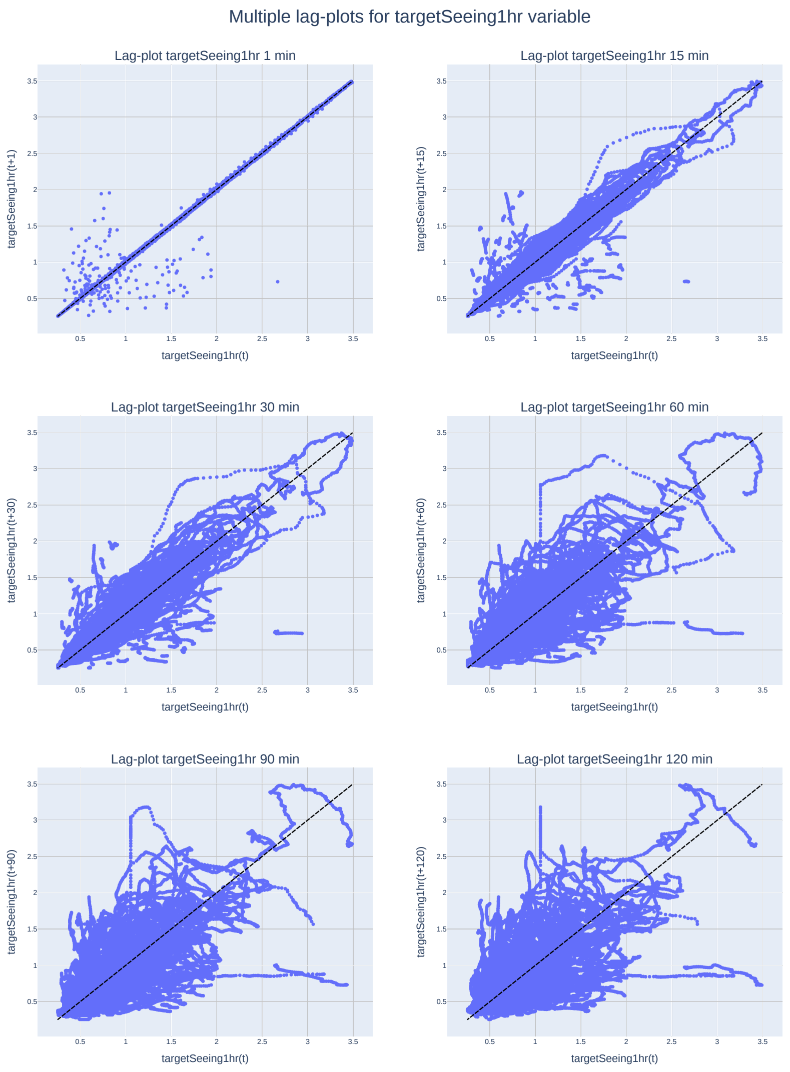


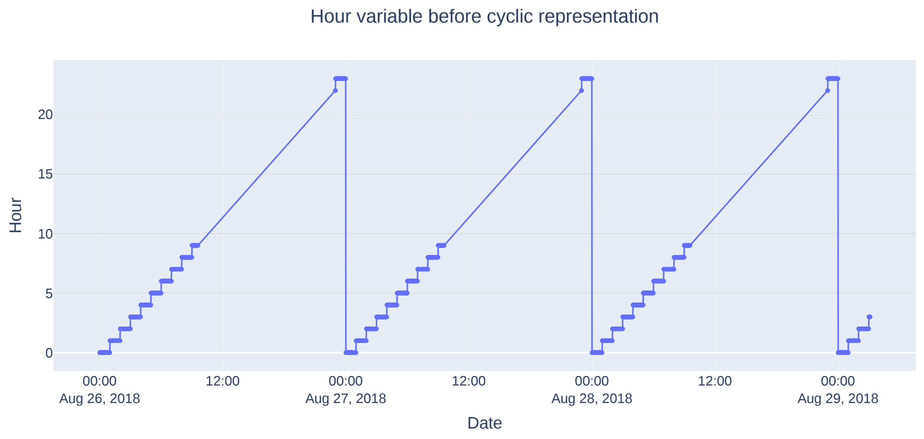
Figure 5.6: Lag plots for the principal target variable ("targetSeeing1hr"). The black dashed line is for reference.

Here, we observe that the main target variable demonstrates correlation with itself across various time shifts. This suggests that employing a machine learning approach to predict this value is indeed a suitable strategy to address the problem.
The black dashed line serves as a reference, indicating perfect correlation between a feature and its time-shifted values. Although some points deviate significantly from this line even in the 1-minute lag plot, there are two primary reasons for this: the sudden fluctuations that seeing can exhibit and the variance between consecutive rows, where the first may represent the final measurement of the observatory at night and the subsequent one could be the initial measurement of the following night. This discontinuity in the time series is not likely to be a significant problem because the number of nights from the start to the end of the database is negligible compared to the amount of continuous data collected throughout the night.

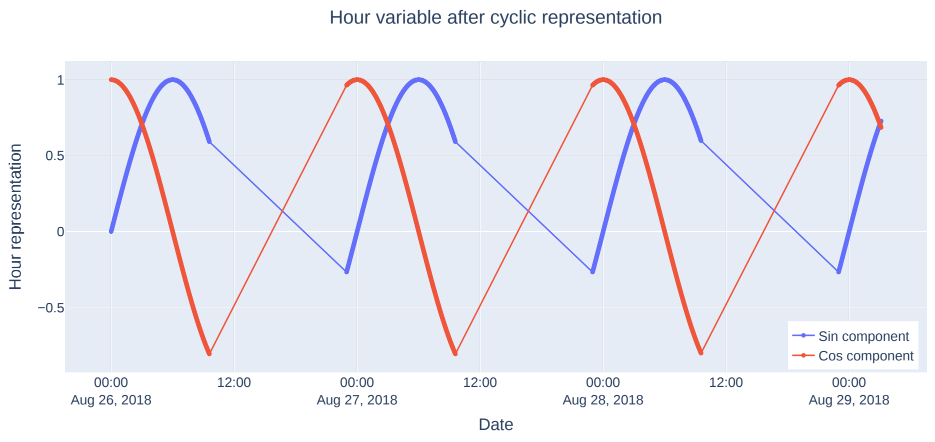
5.2. Feature Engineering

In this section, our focus will be on modifying and refining the database to improve the performance of our models. Our objective is to achieve improved prediction efficiency, aiming for better prediction outcomes and reduced training and prediction times. Thus, the objective is to reduce the number of input features. To do this, we compare the correlation between the target variables to the input features and run a few algorithms to get the feature importance of each variable and then decide if we are going to drop them depending on the performance changes.

First, we adjust the “Date” variable. Since this column is of datetime type, we will extract only the month and hour components, considering the year, day, and minutes to be less relevant. Secondly, we change the month and hour variables into periodic representations by converting them into radians and subsequently deriving the sine and cosine components of these values.



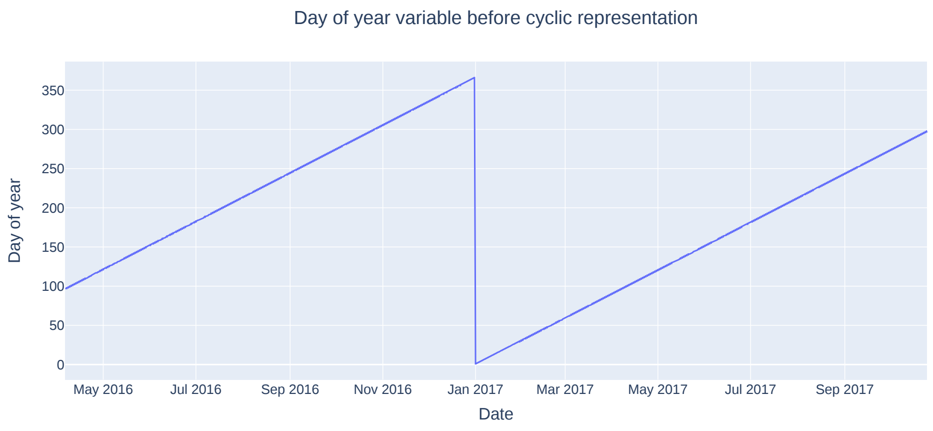
(a) Hour value obtained by the “Date” variable from April 6th to April 9th, 2016.



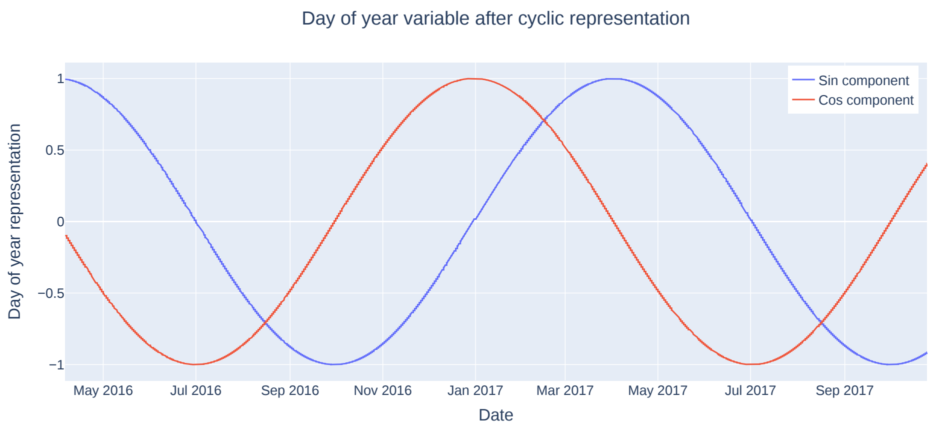
(b) Time of day cyclic representation from April 6th to April 9th, 2016.

Figure 5.7: Hour variable before and after modifications.

Here in Figure (5.7.a), we observe the hour column before the modification, which consists of discrete values ranging from 0 to 23. Notably, there are no values recorded after 9, at least for that particular night. Typically, there is no data available after 9:00, and the first recorded entries of the night typically start at 23:00. Following the transformation, in Figure (5.7.b) we can see the representation of the hours appears more continuous. However, the entire cyclical form cannot be fully discerned due to the absence of data from 9:00 to 23:00.



(a) Day of year plot obtained by the "Date" variable from April 6th, 2016 to October 25th, 2017.

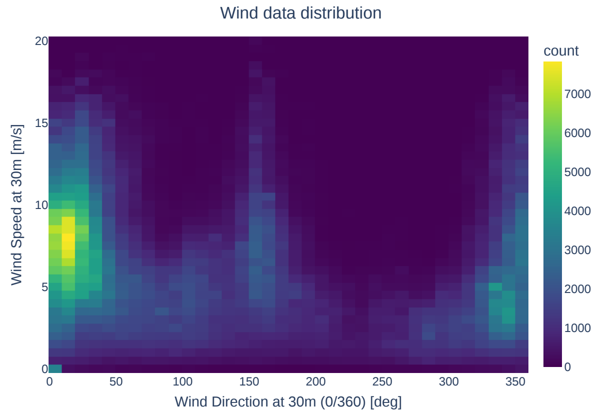


(b) Day of year cyclic representation from April 6th, 2016 to October 25th, 2017.

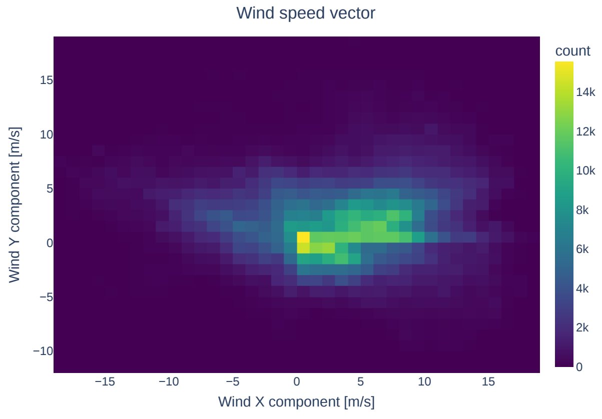
Figure 5.8: Day variable before and after modifications.

In Figure (5.8), we can observe a similar pattern to the previous figure. However, the only point of discontinuity occurs when the year changes. Since we have more data available (almost every days), we can fully observe the cyclical behavior of the day variable after the transformation.

Next, we modify the measurement unit of two variables: Wind direction at 10 and 30 meters. This adjustment is necessary because these features were initially recorded in degrees, which might not effectively capture their cyclic nature within the model. Therefore, we will convert these columns into sine and cosine components to better represent their periodic characteristics.



(a) Wind direction and velocity before modification.



(b) Wind direction and velocity after modification.

Figure 5.9: Wind data distribution heatmap with all training data included.

Here in Figure (5.9), we can observe the distribution of wind speed and direction at 30 meters, with the majority of points concentrated between 0 and 30 degrees, and a velocity ranging from 5 to 10 m/s. After modification, we can also see the X and Y components of the wind speed. This is calculated by converting degrees to radians, and then obtaining the sine and cosine components, which are then multiplied by the magnitude of the wind speed.

The next step involved incorporating seeing statistics, which included calculating the average, minimum, and maximum values over the past 5, 10, and 15 minutes. These values have proven to be one of the best combinations of statistical information through trial and error.
After adding some statistics of the main features, we wanted to gather additional information, such as the first derivative of all the time series columns. Specifically, we focused on the seeing, air temperature, and wind speed values. To calculate the derivative, the only option is to use backward finite differences. This is because, in the real use case, we wouldn't have access to future values, such as the seeing 5 minutes ahead. The formula is as follows:

$$\left(\frac{dy}{dt}\right)_i = \frac{y_i - y_{i-1}}{t_i - t_{i-1}} \tag{5.1}$$

With this, and given that all the data is collected at one-minute intervals, Equation 5.1 simplifies to calculating $y_i - y_{i-1}$. However, we decided instead to use the present values of seeing, temperature, and wind speed, along with the values from 5, 10, and 15 minutes in the past, to include these first derivatives in the input data.
Following that, we generated several new variables, including the temperature difference between 30 and 2 meters, as well as the wind speed and direction differences between 30 and 10 meters. In the same process, we constructed Exponential Moving Average (EMA) features of air temperature, absolute humidity, and relative humidity at 30 meters. Similarly, we applied the EMA methodology to other variables such as wind speed V at 20 [m], maximum wind speed at 10 [m], MASS-DIMM Tau0, and current DIMM Seeing, all utilizing different configurations.
The method used to construct these EMA variables is detailed below:

The expression we use to calculate this values is using normalized weights (adjusting the calculations). This ensures that the EMA values are comparable across different time spans and helps maintain the consistency of the EMA calculation. Mathematically, the weights we use are $w_i = (1-\alpha)^i$, so for a observation $\vec{x}_i = (x_0, x_1, ..., x_t)$, the exponential weight moving average is given by:

$$y_t = \frac{x_t + (1-\alpha)x_{t-1} + (1-\alpha)^2 x_{t-2} + ... + (1-\alpha)^t x_0}{1 + (1-\alpha) + (1-\alpha)^2 + ... + (1-\alpha)^t} \tag{5.2}$$

If we don't normalize the weights, the exponential weighted values are calculated recursively by the formula:

$$\begin{aligned} y_0 &= x_0 \\ y_t &= (1-\alpha) \cdot y_{t-1} + \alpha \cdot x_t \end{aligned}$$

Our goal was to generate these variables using different parameters and evaluate their informational value to the model through feature importance analysis and Pearson correlation. We concluded that the best way to add this type of features into the input data is with the method ewm[4] with parameters $\alpha = 0.3$, $\alpha = 0.5$ or $\alpha = 0.8$ and *adjust*=True, that is, normalize the weights.

The next step was to consider how to handle a large number of data points for absolute humidity, relative humidity, and temperature at different heights.
Given the obvious high correlation among these variables, using them directly didn't seem optimal for performance reasons. Instead, two smarter approaches were considered. The first approach was to use only a small subset of these variables, for example, absolute humidity, relative humidity, and temperature at 0, 500, 1000, 5000, and 10 000 m. The second approach was to calculate the largest eigenvalue of the matrices formed by these variables at different heights.

As we can see in Figure 5.10, there are small submatrices within each large matrix defined by the three variables. We decided to calculate the first eigenvalue of all three submatrices within each large block. This resulted in nine eigenvalues and their corresponding eigenvectors, which we used to obtain new features. By doing this, we eliminated all 117 original columns and added only these nine new variables. This allowed us to test a few models and evaluate their new performance and feature importance.

[4] https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.ewm.html

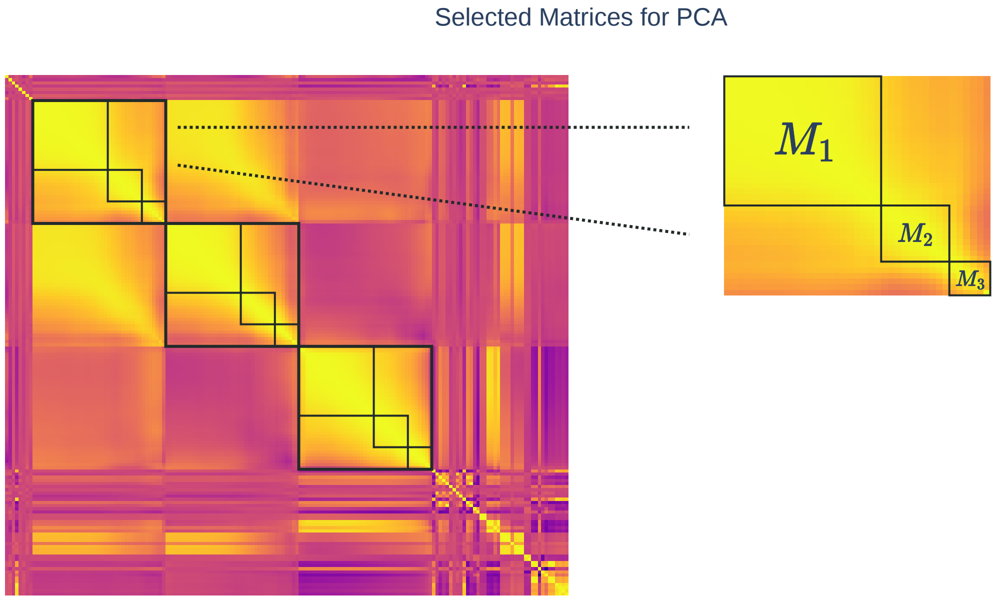


Figure 5.10: Chosen Matrices for PCA: In this plot, an example of the absolute humidity variables with their corresponding submatrices is shown.

All the calculations of these eigenvectors were performed using the PCA algorithm with the following details: For absolute humidity, the three submatrices were from 0 to 1450 [m], 1600 to 4400 [m], and 5000 to 10 000 [m]. For relative humidity, the three submatrices were from 0 to 1450 [m], 1600 to 6200 [m], and 7000 to 10 000 [m]. For temperature, the three submatrices were from 0 to 625 [m], 700 to 3100 [m], and 3500 to 10 000 [m]. These heights were selected manually by examining the correlation plot (Figure 5.5). Adding these new features to the model resulted in better performance compared to using the unmodified features. However, this approach did not improve the model's performance more than simply adding a few of the original raw features, so this procedure was ultimately rejected. This was demonstrated by feature importance methods, such as internal methods for some prediction models or permutation methods, where the values of certain columns are shuffled to compare model performance.

The final step involved in this feature engineering process was eliminating features that we deemed unnecessary for inputting into the models. This decision was based on either redundant information being available in other formats or the model assigning lower significance to those particular features.
In the end, we have 47 input features, a column containing the actual model PM that we don't use for any purpose other than comparing results, and 3 target variables, giving a total of 51 columns.
Finally, we apply a scaling method to all the data. The scaling algorithm applied was standard scaler[5]. The decision of not using the minmax scaler[6] was based on that we do not

[5] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html.
[6] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html.

know if the maximum value in the training set will be a maximum in the test set and so with the minimum.
The next figure represents the Pearson correlation between all features we consider important for this task. It's important to note that we intentionally didn't eliminate all correlated features. The decision not to eliminate all correlated features is driven by our search of better performance, and having a number of columns between 40 and 60 appears suitable for this purpose.

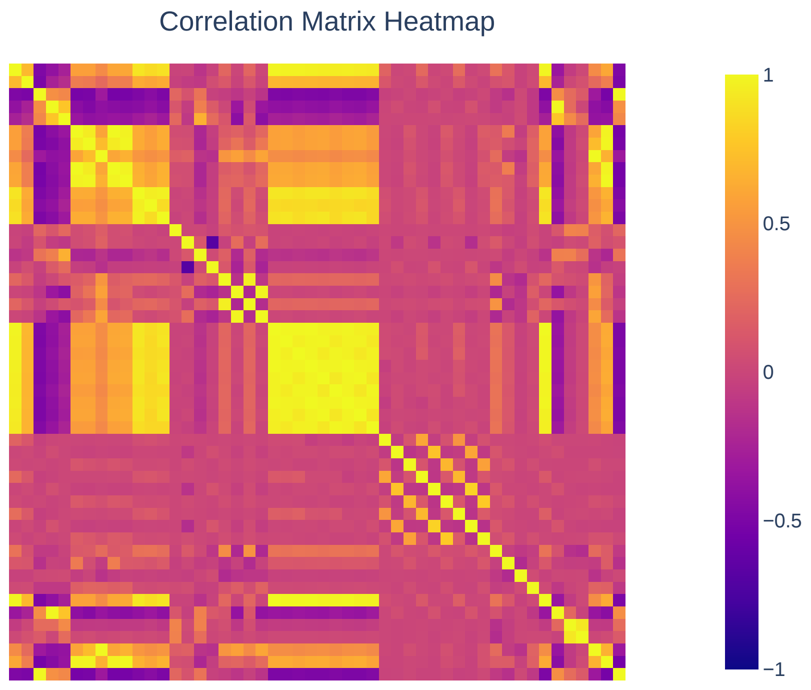


Figure 5.11: Correlation plot for a sample of 100 000 training data after all the feature engineering.

All the previous changes mentioned were integrated into a pipeline to make the process efficient and clear.

5.3. Regression Problem

The first approach to solve the problem involved is employing a regression model, aimed at predicting a continuous value. Using the features explained in the previous section, the goal is to predict the variable "targetSeeing1hr".
As we mentioned earlier, there are two more target variables, however, we are going to first compare the performance of the models with only the main target variable and then we will select a few of the best models and try again to compare how good are they with all of the target variables.

To perform a regression task, we don't require many changes to the database. As previously mentioned, redundant columns have been removed, and the data types have been modified according to the features involved. Therefore, we can proceed to train a regression model directly and evaluate its performance.

5.4. Classification Problem

An alternative strategy involves treating this problem as a classification task. Here, we divide the target variable into predefined ranges, as determined by ESO (the same as previous sections), and encode them into numerical values ranging from 0 to 7. All other features remain unchanged.

The evaluation of the model differs from regression methods, as metrics like MSE, RMSE, or MAE are no longer applicable. Instead, we assess the model's performance using metrics such as precision, recall, accuracy, and F1-score. Personalized metrics like Diagonal Average and Success Rate can still be calculated without any complications.

The purpose of employing both regression and classification approaches is to identify whether a particular model yields superior metrics.
It's worth noting that we can use these two distinct methods due to the nature of how observations at ESO specify desired seeing ranges. By using these ranges, we can create classes and then proceed to evaluate both regression and classification tasks, comparing the outcomes using metrics that works for both models.

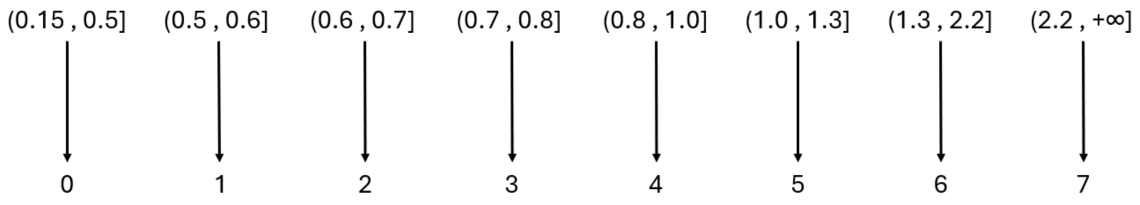


Figure 5.12: Label conversion from continuous to discrete values.

From now on, we will present the results of both approaches mentioned. Our primary focus is to examine and select the best model based on their metrics. It's important to note that Diagonal Average and Success Rate are two of the most crucial metrics for selecting a particular model. Additionally, low prediction time is essential since we need to predict the average visibility for the next hour every minute.
Having said that, the methodology used to develop these models involved a process of trial and error, including creating new variables and making modifications in an attempt to achieve the best performance.

Chapter 6

Results

In this chapter, we will present the outcomes achieved by each model we experimented with. It's important to mention again that we use classification and regression models for this task. As detailed in previous chapters, the dataframe were modified accordingly.
Then, in the concluding section of this chapter, we will identify the best-performing models for additional optimization through hyperparameter search. We will also implement the rolling window technique (adding past values as new columns) on both the training and testing dataframes to see if there exist a performance improvement.

While the metrics we present vary based on the specific task at hand, both Diagonal Average and Success Rate will be consistently featured across all models so this metrics will be the key to pick the best models.
Since the regression problem is the primary and most intuitive method for addressing this issue, the root mean square error metric will also hold significant importance in determining the better models. This is also highlighted by the fact that the current model employed at ESO operates as a regression model, as do other attempts to predict seeing and other optical turbulence variables.

The outcomes will be visualized using a contingency matrix plot, aligning with the approach adopted by ESO to demonstrate model performance. However, this contingency matrix will be exclusively showcased within the Regression Models section to prevent overwhelming visual saturation. Conversely, within the Classification Models section, a table detailing the performance of each model will be presented.

Lastly, we will address time performance. While training time may not hold significant weight for ESO given their computational resources, it holds considerable importance for us due to time constraints, as we cannot afford to train a complex model over five splits and then retrain it again for evaluating the test set. Conversely, prediction time is of great importance to ESO, as their objective revolves around forecasting the average seeing every minute for the next hour. Therefore, a prolonged prediction time wouldn't help us achieve our goal.
We will initially present the straightforward stationary model employed at the ESO called "PM". As previously mentioned, this model involves averaging the DIMM Seeing values over the past 10 minutes and utilizing this average as a prediction for the average seeing over the next hour.
Despite its simplicity, this model doesn't yield poor metrics. This is attributed to the inherent

difficulty in predicting optical turbulence variables like seeing, given their chaotic nature.

6.1. Regression Models

In this section, we will review the regression models in detail, focusing on their error-based and time performance. At the end of the section, a table with the results will be shown.

6.1.1. PM: Actual model at the European Southern Observatory

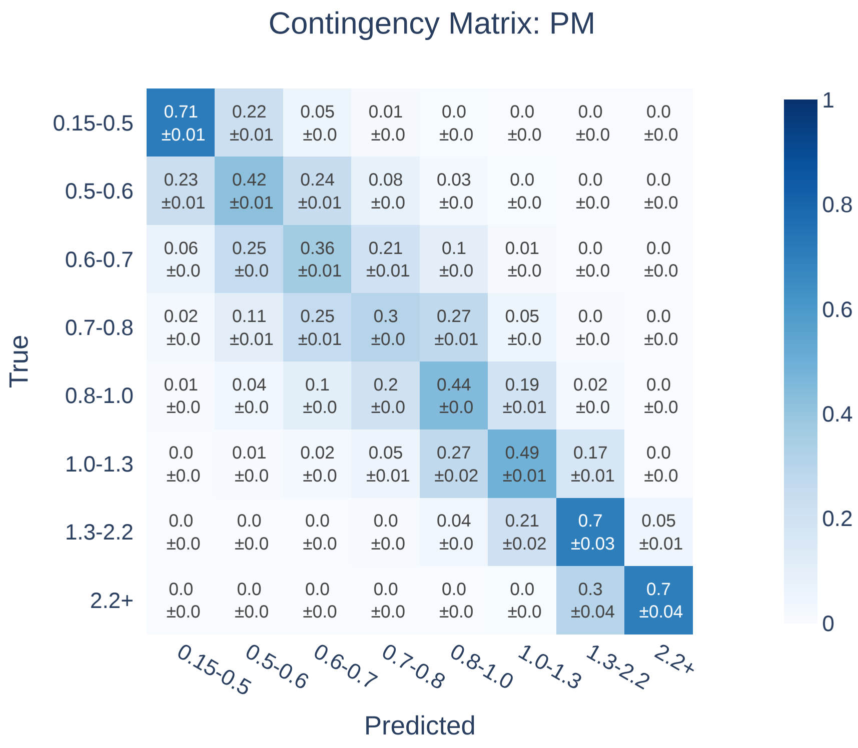


Figure 6.1: Display of confusion matrix of stationary model. The matrix is normalized by the rows (true values).

The metric values for this model are:

- Root Mean Square Error: 0.180
- Mean Square Error: 0.032
- Mean Absolute Error: 0.122
- Diagonal Average: 0.516
- Success Rate: 0.727

The training time is omitted because this isn't a conventional model, it is just a heuristic about the target value and it is calculated by a 10 minutes average of the DIMM Seeing value in arcsec.

The prediction time is obtained by calculating this 10-minutes average through all the validation sets and then taking the mean. The outcome of this procedure is that the prediction was 0.007 (s).

6.1.2. Isotonic

We'll begin by testing this initial model due to its simplicity and the constraint of utilizing only one feature as input. We've chosen "EMA Seeing 10 min" as the input variable (exponential moving average of seeing values with a 10-minutes span), as it yielded the most favorable outcome through iterative testing.
The hyperparameters for this model were $y_{min} = 0.15$, $y_{max} = 4.5$, *increasing*="auto" and *out_of_bounds*="clip".
Figure 6.2 show the confusion matrix display for this Isotonic model.

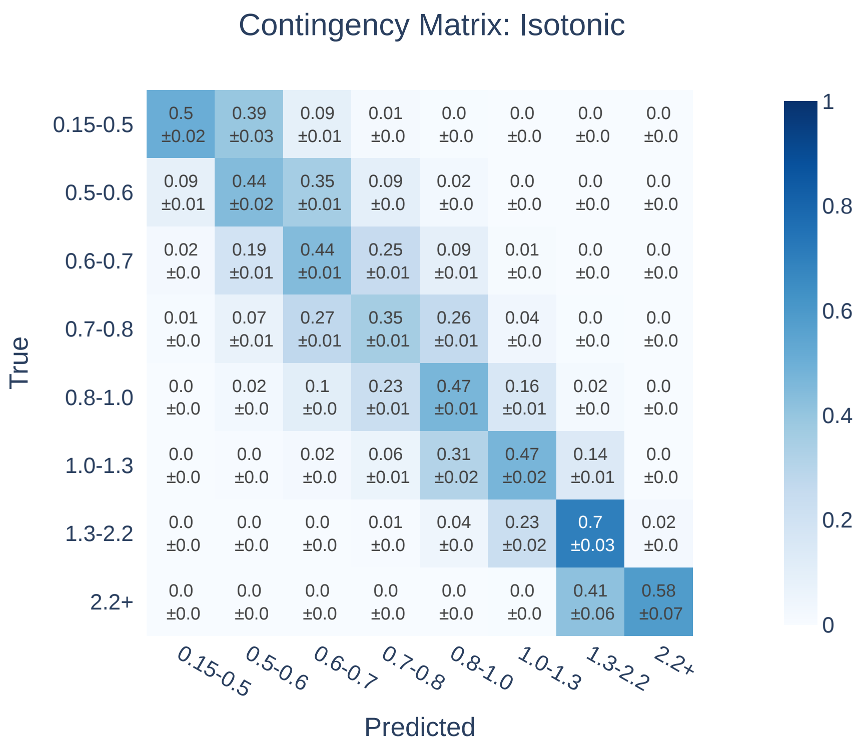


Figure 6.2: Display of confusion matrix of Isotonic regression model. The matrix is normalized by the rows (true values).

The metric values for this model are:

- Root Mean Square Error: 0.174
- Mean Square Error: 0.030
- Mean Absolute Error: 0.118
- Diagonal Average: 0.494
- Success Rate: 0.758

The training time was 2.621 (s) and the prediction time was 0.010 (s).

We expected this model to be less accurate than others. However, given the challenging nature of predicting the seeing, we included it for the sake of completeness. Although the results are decent considering the simplicity of the model.

6.1.3. SGD

The hyperparameters for this model were the default by the sklearn library. Those are: squared error loss, l2 penalty, *alpha*=0.0001, *eta0*=0.01, *learning_rate*="invscaling" and only one parameters was changed: *max_iter*=10 000.
Figure 6.3 show the confusion matrix display for this SGD model.

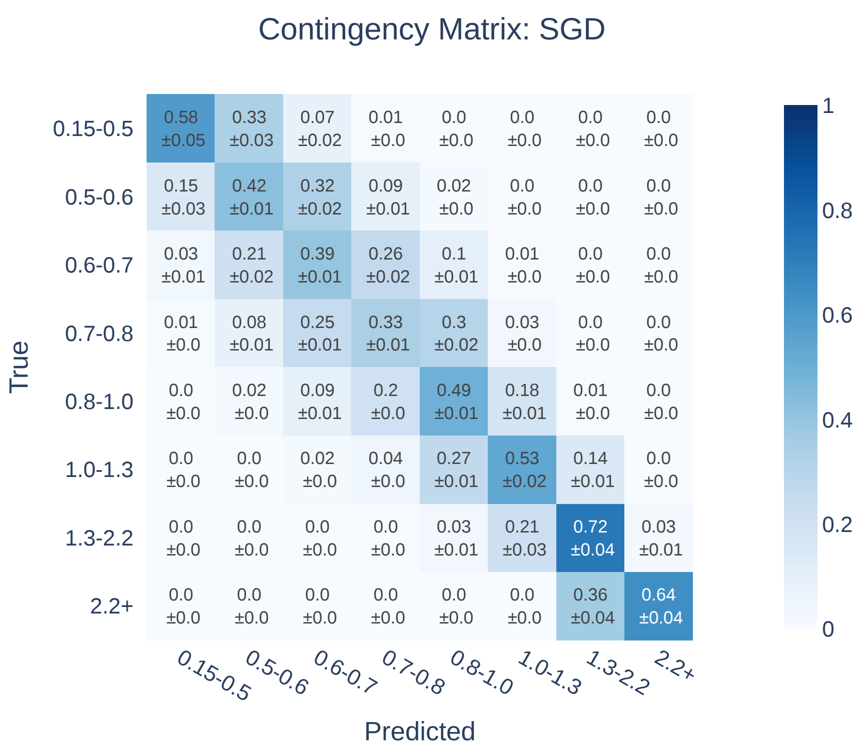


Figure 6.3: Display of confusion matrix of SGD regression model. The matrix is normalized by the rows (true values).

The metric values for this model are:

- Root Mean Square Error: 0.159
- Mean Square Error: 0.025
- Mean Absolute Error: 0.111
- Diagonal Average: 0.515
- Success Rate: 0.765

The training time was 24.391 (s) and the prediction time was 0.028 (s).

This model demonstrates some improvement and surpasses all the metrics of the PM model. The best results are in the $(1.3, 2.2]$ and 2.2+ ranges. However, this model does not provide extensive hyperparameter optimization, which may be a reason not to choose it for the next step (hyperparameter optimization and evaluation on the test set).

6.1.4. DT

Here we try the most simple tree model as a regressor. For the hyperparameters, we use a *min_samples_split*=20 and a *max_depth*=7. That was all for training time purposes and rmse performance for the tree.
We didn't search or change these parameters because we think ensemble models will outperform the DT model because multiple trees typically reduce variance. so we will focus on their hyperparameters instead.
Figure 6.4 show the confusion matrix display for this decision tree model.

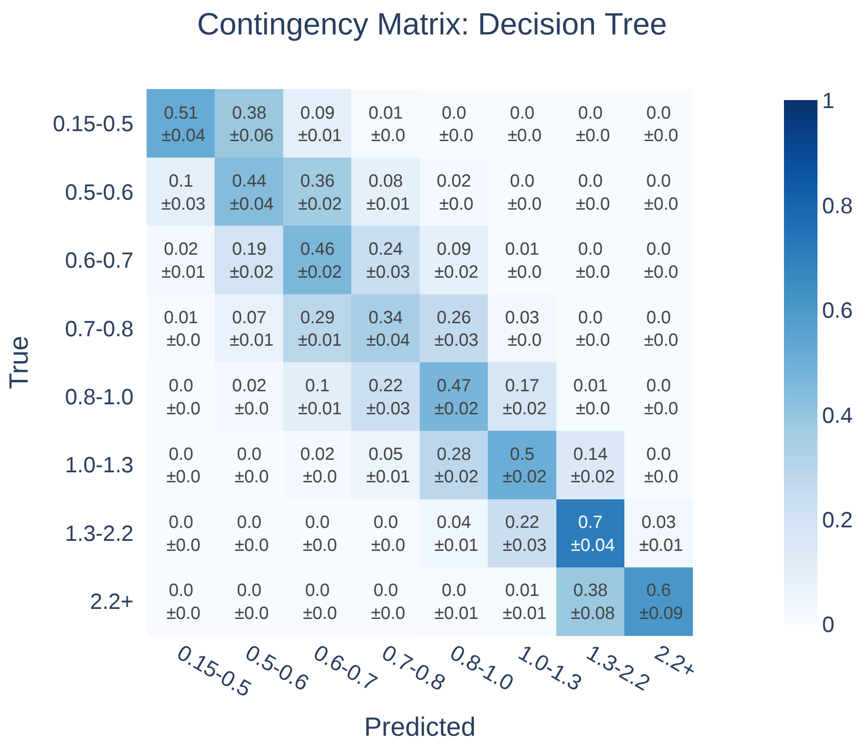


Figure 6.4: Display of confusion matrix of Decision Tree regression model. The matrix is normalized by the rows (true values).

The metric values for this model are:

- Root Mean Square Error: 0.172
- Mean Square Error: 0.030
- Mean Absolute Error: 0.116
- Diagonal Average: 0.503
- Success Rate: 0.764

The training time was 143.810 (s) and the prediction time was 0.020 (s).

As we can see, the performance of a single tree is similar to that of the Isotonic model and slightly better than that of the stationary model. Although it is a simple model, its ease of interpretation is the main reason we tried it. As expected, we anticipate better results from ensemble models that use these decision trees as their basis.

6.1.5. RF

Now we present the first ensemble model and the metrics obtained. To address overfitting and training time, we adjusted the default hyperparameters through trial and error as follows: *max_depth*=7 and *min_samples_split*=20.

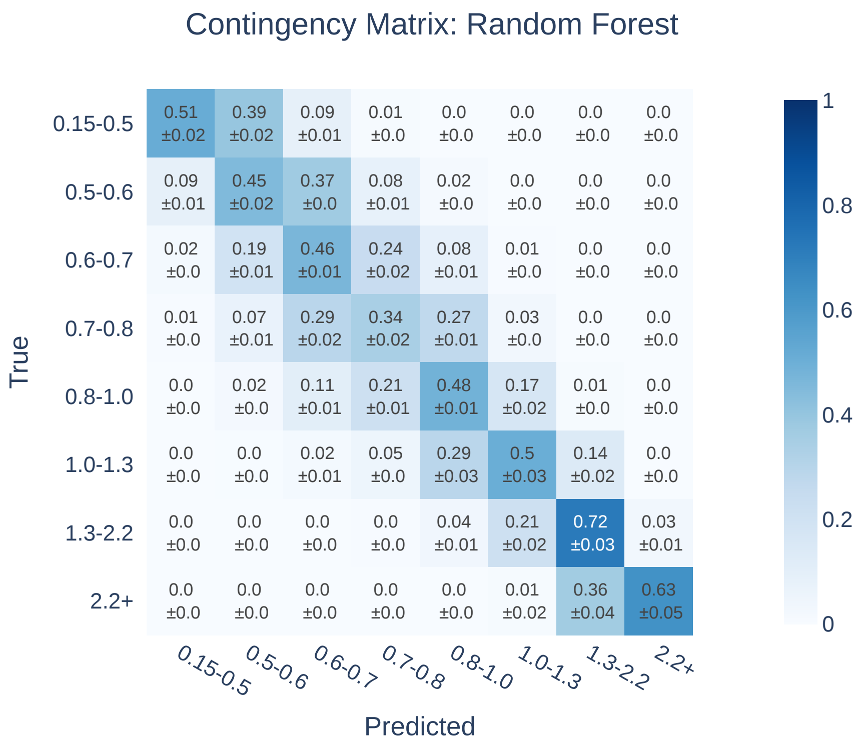


Figure 6.5: Display of confusion matrix of Random Forest regression model. The matrix is normalized by the rows (true values).

The metric values for this model are:

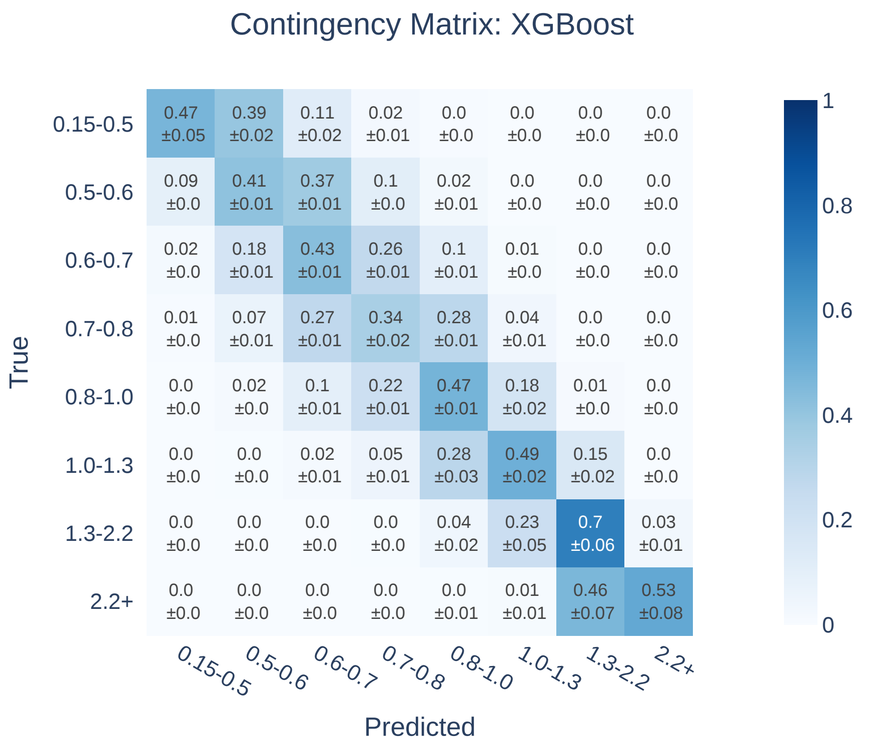
- Root Mean Square Error: 0.165
- Mean Square Error: 0.027
- Mean Absolute Error: 0.112
- Diagonal Average: 0.511
- Success Rate: 0.768

The training time was 4303.685 (s) and the prediction time was 0.489 (s).

We can already see that this model performs well compared to the stationary model; however, it is still slightly worse than the SGD regression and the training time was significantly higher compared to the previous models shown. On the other hand, we observe decent predictions at higher bins but very poor performance at the lower ones, specifically in the range of 0.15-0.5, which is the best attribute of the stationary model.

6.1.6. XGBoost

Here we present one of the most well-known ensemble algorithms: Extreme Gradient Boosting [15]. As a first attempt, we use the default hyperparameters. However, this model allows for a considerable amount of modifications.



True \ Predicted	0.15-0.5	0.5-0.6	0.6-0.7	0.7-0.8	0.8-1.0	1.0-1.3	1.3-2.2	2.2+
0.15-0.5	0.47 ±0.05	0.39 ±0.02	0.11 ±0.02	0.02 ±0.01	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0
0.5-0.6	0.09 ±0.0	0.41 ±0.01	0.37 ±0.01	0.1 ±0.0	0.02 ±0.01	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0
0.6-0.7	0.02 ±0.0	0.18 ±0.01	0.43 ±0.01	0.26 ±0.01	0.1 ±0.01	0.01 ±0.0	0.0 ±0.0	0.0 ±0.0
0.7-0.8	0.01 ±0.0	0.07 ±0.01	0.27 ±0.01	0.34 ±0.02	0.28 ±0.01	0.04 ±0.01	0.0 ±0.0	0.0 ±0.0
0.8-1.0	0.0 ±0.0	0.02 ±0.0	0.1 ±0.01	0.22 ±0.01	0.47 ±0.01	0.18 ±0.02	0.01 ±0.0	0.0 ±0.0
1.0-1.3	0.0 ±0.0	0.0 ±0.0	0.02 ±0.01	0.05 ±0.01	0.28 ±0.03	0.49 ±0.02	0.15 ±0.02	0.0 ±0.0
1.3-2.2	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.04 ±0.02	0.23 ±0.05	0.7 ±0.06	0.03 ±0.01
2.2+	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.01	0.01 ±0.01	0.46 ±0.07	0.53 ±0.08

Figure 6.6: Display of confusion matrix of XGBoost regression model. The matrix is normalized by the rows (true values).

The metric values for this model are:

- Root Mean Square Error: 0.178
- Mean Square Error: 0.038
- Mean Absolute Error: 0.121
- Diagonal Average: 0.480
- Success Rate: 0.770

The training time was 306.999 (s) and the prediction time was 2.254 (s).

Here we can observe that the error-based metrics are poor compared to the other models. However, the success rate is higher. Given this and the fact that it didn't take long to train, this model could be a good candidate for tuning and parameter adjustment to better address this problem.

6.1.7. LightGBM

Now it's time to consider another well-known ensemble model: Light Gradient Boosting Machine [16]. The fact that it is usually faster and provides results comparable to XGBoost are sufficient reasons to try this model for predicting the average seeing. In Figure 6.7, we can see the contingency matrix:

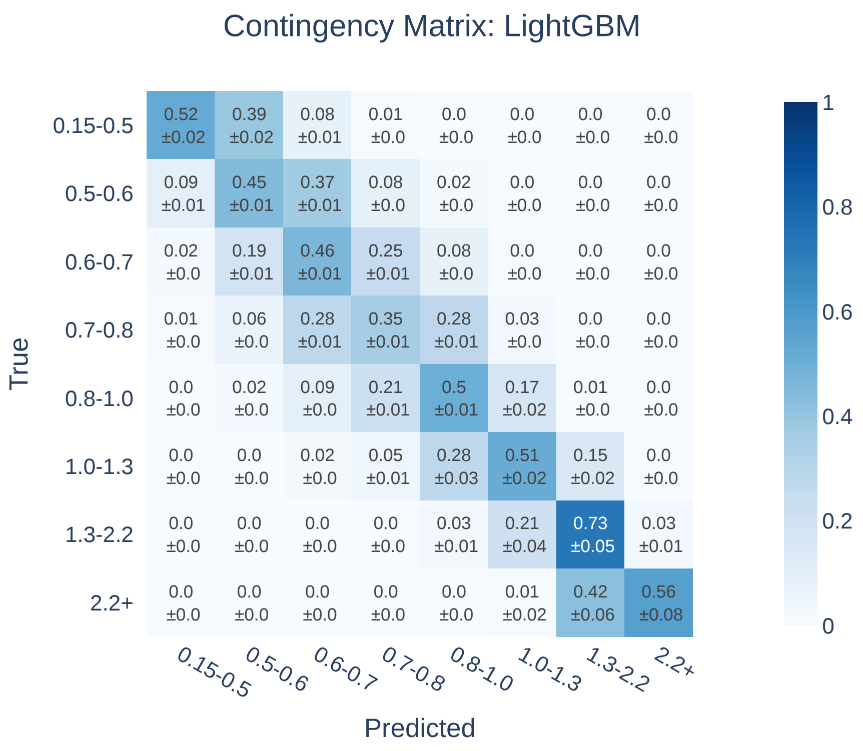


Figure 6.7: Display of confusion matrix of LightGBM regression model. The matrix is normalized by the rows (true values).

The metric values for this model are:

- Root Mean Square Error: 0.162
- Mean Square Error: 0.026
- Mean Absolute Error: 0.110
- Diagonal Average: 0.510
- Success Rate: 0.775

The training time was 202.289 (s) and the prediction time was 1.617 (s).

Given these metrics, we can already say that this is the best model so far. However, parameter tuning and other modifications might lead to another model being optimal.
As we mentioned earlier, this model is faster than XGBoost and gives better results. However, the issue of poor predictions in the lowest range remains.

6.1.8. CatBoost

Finally, we present the last ensemble model: CatBoost [17]. Although this algorithm is primarily designed to handle categorical data, which is not our case, we wanted to try it for the sake of completeness regarding the most popular tree-based ensemble models.

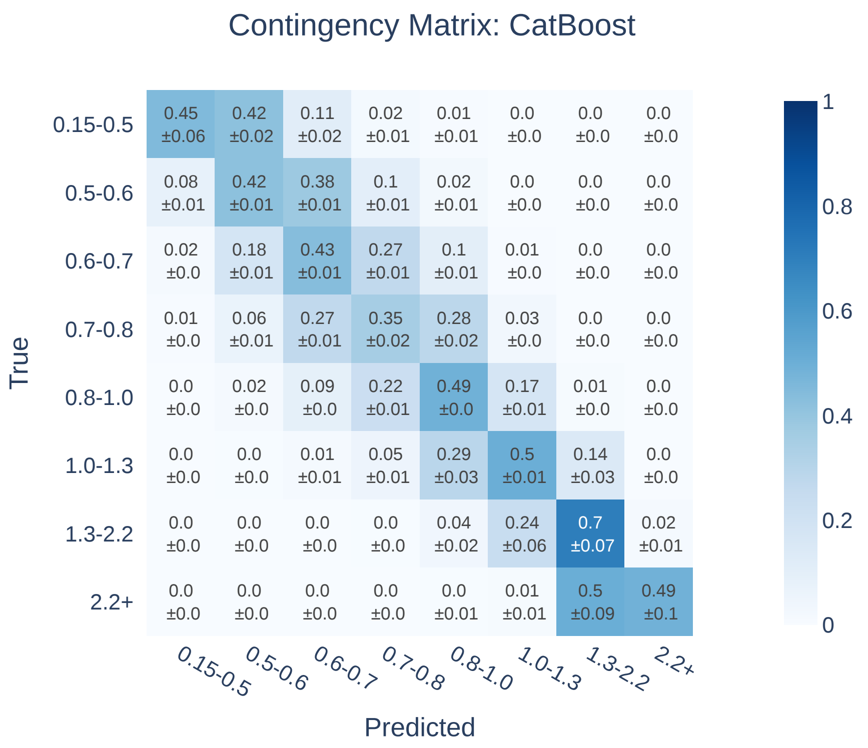


Figure 6.8: Display of confusion matrix of CatBoost regression model. The matrix is normalized by the rows (true values).

The metric values for this model are:

- Root Mean Square Error: 0.176
- Mean Square Error: 0.031
- Mean Absolute Error: 0.119
- Diagonal Average: 0.478
- Success Rate: 0.773

The training time was 673.489 (s) and the prediction time was 0.232 (s).

We can observe that the results are quite similar across all ensemble models. With sufficient hyperparameter optimization, it is likely that all these models will achieve similar, strong performance, surpassing the PM metrics. However, the main difference may lie in the training time, which can also be misleading because each model has different default hyperparameters.

6.1.9. MLP

From now on, we are going to showcase basic and advanced deep learning algorithms, starting with the most well-known one, the Multi-Layer Perceptron or Feed Forward Artificial Neural Network. For this construction, we compared the performance of a few architectures by varying the number of layers and the number of units per layer, and concluded that the most basic one works better. Therefore, the model consists only of a dense layer with 8 units and a dense layer with 1 unit for regression purposes. The batch size is 512, and we train the model for 10 epochs.

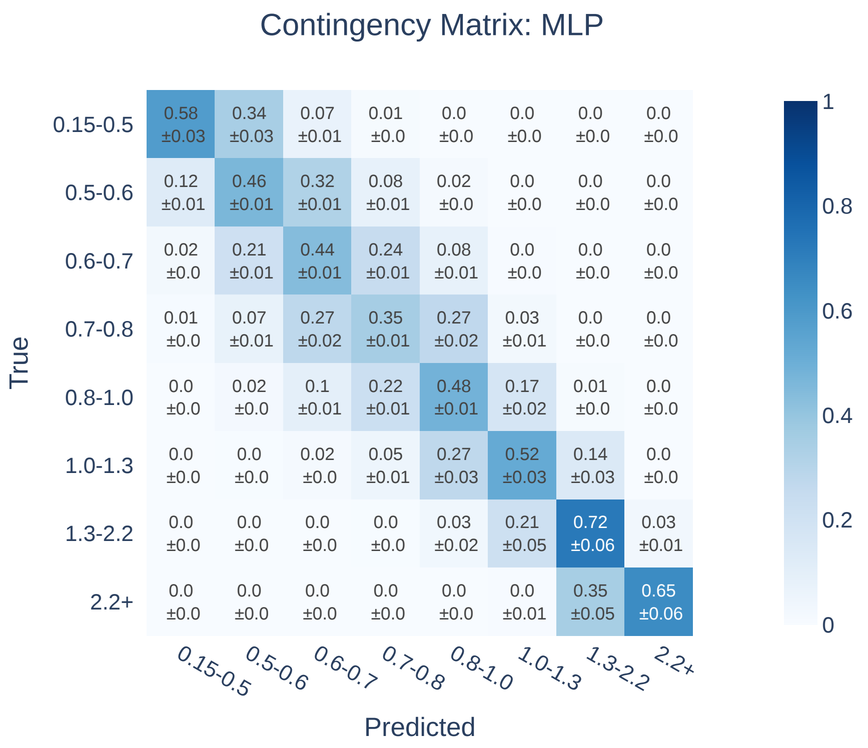


Figure 6.9: Display of confusion matrix of MLP regression model. The matrix is normalized by the rows (true values).

The metric values for this model are:

- Root Mean Square Error: 0.158
- Mean Square Error: 0.025
- Mean Absolute Error: 0.111
- Diagonal Average: 0.513
- Success Rate: 0.772

The training time was 812.934 (s) and the prediction time was 17.679 (s).

Here we can see that we obtained the lowest value for the RMSE compared to all the previous models. Additionally, we can observe a decent result in the contingency matrix. However, we still can't beat the prediction in the first range of the stationary model.

Sliding Window Dataframe

One of the most commonly used data transformations for time series is sliding windows. This technique involves expanding the number of columns with the purpose of capturing past data. The window size refers to how many past time steps we want to capture. Keep in mind that the number of columns will grow linearly with respect to the original number of columns. So, if I have a window size of 5 and M columns, I will have a dataframe of $6M$ columns after the transformation ($5M$ of past data and M of the actual time). Therefore, we need to be cautious about how much past data we capture and which specific data we include, considering memory constraints and the potential for adding noise or highly correlated features, which could worsen model performance.
Note that in these types of time series problems, we typically use sliding windows on the target variable to incorporate its past values as relevant input features. However, in this case, we cannot do that directly. The variable "targetSeeing1hr" and the other two are the average, minimum and maximum values of seeing values over one hour. Therefore, if we had, for example, "targetSeeing1hr(t-1)", we would incorporate a lot of future information that we don't have access in practice. As a result, we won't use sliding windows on any of the target variables.

This technique is primarily used to enable the testing of RNN models. This is because these types of deep learning models require input data in the shape of (number of observations, length of input sequence, number of features). With this data format, we can incorporate previous data into a single prediction and expect (maybe) a better result.
Another model that can be used with this type of data is a convolutional model. Both models receive an input shape of (*batch size, time steps, feature dimensions*), so this windowed data will be used to try both RNN models and convolutional models.

To run all of these models, we used a window size of 5. As mentioned before, this means we take 5 time steps and all their features to make one prediction. This window size was primarily based on memory constraints, and we found it to be effective given the past correlation with future values.

Due to memory constraints, the batch size of all deep learning models will be 512. Additionally, for time efficiency, every model will be trained for 10 epochs. Although this number is typically low for this type of problem, the models need to be trained five times for cross-validation, and we found that their performance was similar to previous attempts, with only slight improvements (if any) after a few more epochs.

Regarding model architectures, each model was kept as simple as possible. For our case, the LSTM and GRU models consist of only 8 units with a tanh activation function and a final dense layer for the regression task. The convolutional model includes a convolutional layer with 8 filters, a kernel size of 5, and a ReLU activation function. For the classification

task, the only modifications made were in the final layer, changing the output from 1 to the length of the classes, which is 8 classes.
In all the models mentioned above, the optimizer used was Adam [23], and the loss function employed was Mean Squared Error (MSE).

6.1.10. LSTM

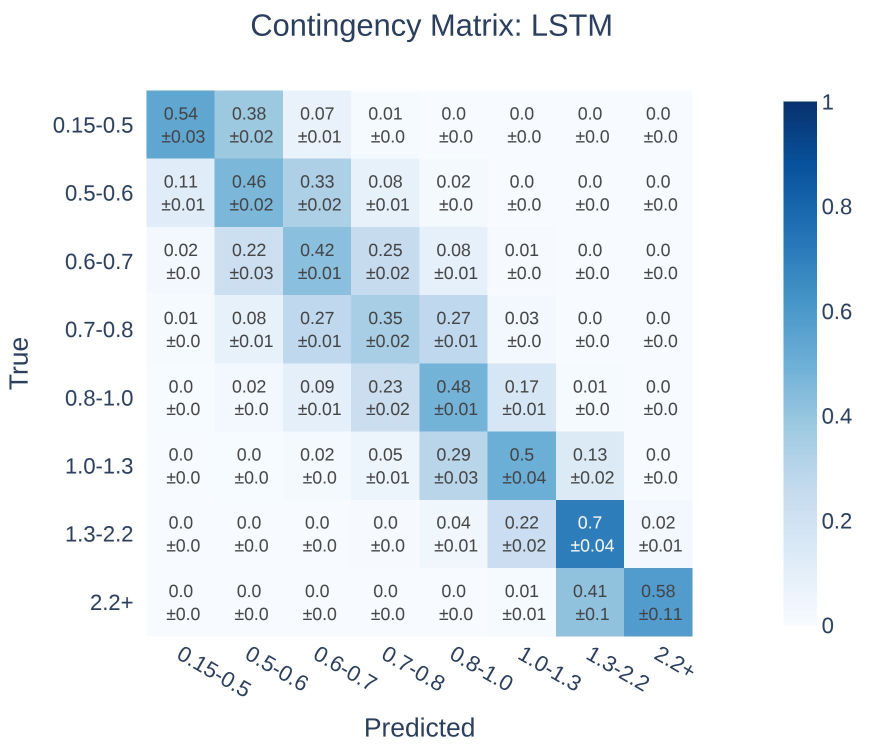


Figure 6.10: Display of confusion matrix of LSTM regression model. The matrix is normalized by the rows (true values).

The metric values for this model are:

- Root Mean Square Error: 0.167
- Mean Square Error: 0.028
- Mean Absolute Error: 0.114
- Diagonal Average: 0.504
- Success Rate: 0.755

The training time was 459.005 (s) and the prediction time was 20.445 (s).

With the parameters described above, the LSTM model results in 1792 parameters for the LSTM layer and 9 parameters for the final dense layer, giving a total of 1801 trainable parameters for this architecture.

In Figure 6.10, we can see expected results. The LSTM model performs well, but it is not significantly better than previous attempts. This conservative perspective might be due to the simplicity of the model architecture and the fact that it was trained for only a few epochs.

6.1.11. GRU

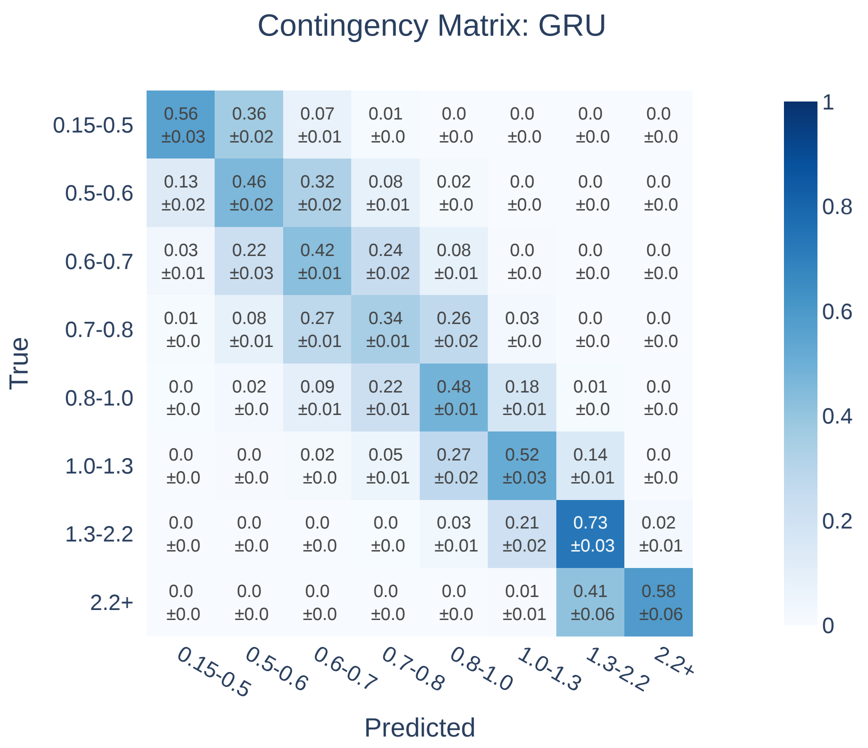


Figure 6.11: Display of confusion matrix of GRU regression model. The matrix is normalized by the rows (true values).

The metric values for this model are:

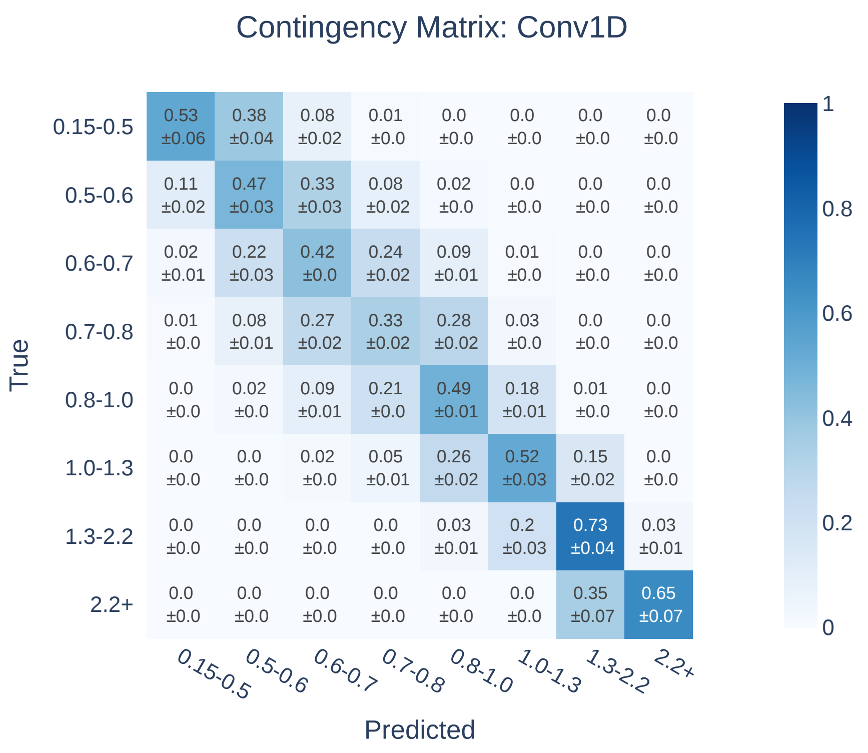
- Root Mean Square Error: 0.163
- Mean Square Error: 0.027
- Mean Absolute Error: 0.112
- Diagonal Average: 0.510
- Success Rate: 0.757

The training time was 458.294 (s) and the prediction time was 22.700 (s).

Under the same conditions, the GRU model results in 1368 parameters for the GRU layer and 9 parameters for the final dense layer, giving a total of 1377 trainable parameters for this architecture.

Given the performance of this model and comparing it with the LSTM, we conclude that the GRU model is superior in all aspects. It has better performance metrics, similar training and prediction times, uses fewer parameters, and provides more accurate predictions in the first bins, which range from 0.15 to 0.5 arcseconds.

6.1.12. CONV1D



True \ Predicted	0.15-0.5	0.5-0.6	0.6-0.7	0.7-0.8	0.8-1.0	1.0-1.3	1.3-2.2	2.2+
0.15-0.5	0.53 ±0.06	0.38 ±0.04	0.08 ±0.02	0.01 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0
0.5-0.6	0.11 ±0.02	0.47 ±0.03	0.33 ±0.03	0.08 ±0.02	0.02 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0
0.6-0.7	0.02 ±0.01	0.22 ±0.03	0.42 ±0.0	0.24 ±0.02	0.09 ±0.01	0.01 ±0.0	0.0 ±0.0	0.0 ±0.0
0.7-0.8	0.01 ±0.0	0.08 ±0.01	0.27 ±0.02	0.33 ±0.02	0.28 ±0.02	0.03 ±0.0	0.0 ±0.0	0.0 ±0.0
0.8-1.0	0.0 ±0.0	0.02 ±0.0	0.09 ±0.01	0.21 ±0.0	0.49 ±0.01	0.18 ±0.01	0.01 ±0.0	0.0 ±0.0
1.0-1.3	0.0 ±0.0	0.0 ±0.0	0.02 ±0.0	0.05 ±0.01	0.26 ±0.02	0.52 ±0.03	0.15 ±0.02	0.0 ±0.0
1.3-2.2	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.03 ±0.01	0.2 ±0.03	0.73 ±0.04	0.03 ±0.01
2.2+	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.35 ±0.07	0.65 ±0.07

Figure 6.12: Display of confusion matrix of Conv1D regression model. The matrix is normalized by the rows (true values).

The metric values for this model are:

- Root Mean Square Error: 0.158
- Mean Square Error: 0.025
- Mean Absolute Error: 0.110
- Diagonal Average: 0.518
- Success Rate: 0.770

The training time was 351.655 (s) and the prediction time was 16.760 (s).

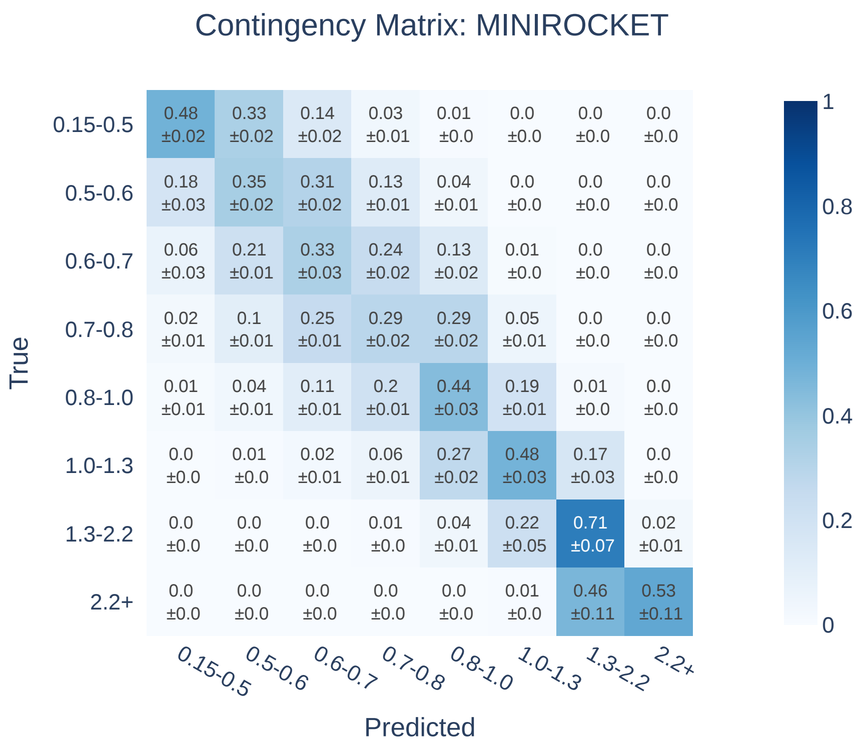
For this model, the Conv1D layer results in 1888 parameters and the final dense layer in 9 parameters, giving a total of 1897 trainable parameters.

Finally, we have the convolutional model. Of all four deep learning models tested, this one and the MLP had the best performance. Additionally, the training time was approximately

24% lower than that of the RNN models, which is a promising sign if we want to search for better parameters and architectures for the convolutional model.

6.1.13. MINIROCKET

Speaking now of state-of-the-art models, ROCKET is a distinctive approach to time series classification/regression compared to traditional machine learning models. ROCKET uses a large set of fixed, non-trainable convolutions applied independently to the time series. It extracts features from each convolution output. ROCKET evolved into MINIROCKET, refining convolutions based on pre-defined sets that have shown equal or better effectiveness.



True \ Predicted	0.15-0.5	0.5-0.6	0.6-0.7	0.7-0.8	0.8-1.0	1.0-1.3	1.3-2.2	2.2+
0.15-0.5	0.48 ±0.02	0.33 ±0.02	0.14 ±0.02	0.03 ±0.01	0.01 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0
0.5-0.6	0.18 ±0.03	0.35 ±0.02	0.31 ±0.02	0.13 ±0.01	0.04 ±0.01	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0
0.6-0.7	0.06 ±0.03	0.21 ±0.01	0.33 ±0.03	0.24 ±0.02	0.13 ±0.02	0.01 ±0.0	0.0 ±0.0	0.0 ±0.0
0.7-0.8	0.02 ±0.01	0.1 ±0.01	0.25 ±0.01	0.29 ±0.02	0.29 ±0.02	0.05 ±0.01	0.0 ±0.0	0.0 ±0.0
0.8-1.0	0.01 ±0.01	0.04 ±0.01	0.11 ±0.01	0.2 ±0.01	0.44 ±0.03	0.19 ±0.01	0.01 ±0.0	0.0 ±0.0
1.0-1.3	0.0 ±0.0	0.01 ±0.0	0.02 ±0.01	0.06 ±0.01	0.27 ±0.02	0.48 ±0.03	0.17 ±0.03	0.0 ±0.0
1.3-2.2	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.01 ±0.0	0.04 ±0.01	0.22 ±0.05	0.71 ±0.07	0.02 ±0.01
2.2+	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.01 ±0.0	0.46 ±0.11	0.53 ±0.11

Figure 6.13: Display of confusion matrix of MINIROCKET regression model. The matrix is normalized by the rows (true values).

The metric values for this model are:

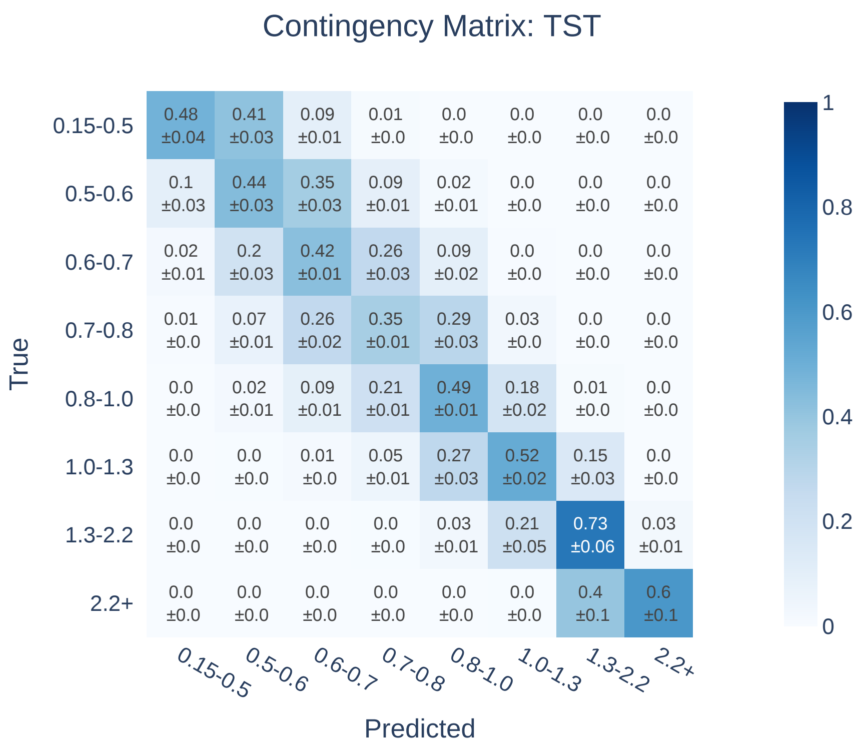
- Root Mean Square Error: 0.185
- Mean Square Error: 0.034
- Mean Absolute Error: 0.132
- Diagonal Average: 0.450
- Success Rate: 0.736

The training time was 1396.839 (s) and the prediction time was 7.443 (s).

Figure 6.13 demonstrates that the default configuration of this model does not fit this problem well. Additionally, complex models of this type typically require significant time for fine-tuning and training multiple times to perform effectively, which we do not have. Therefore, for our work, we will not consider MINIROCKET as an optimal model.

6.1.14. TST

The Time Series Transformer (TST) model, based on [7], was implemented using the tsai library [2]. The implementation and training for this problem utilized default hyperparameters.



True \ Predicted	0.15-0.5	0.5-0.6	0.6-0.7	0.7-0.8	0.8-1.0	1.0-1.3	1.3-2.2	2.2+
0.15-0.5	0.48 ±0.04	0.41 ±0.03	0.09 ±0.01	0.01 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0
0.5-0.6	0.1 ±0.03	0.44 ±0.03	0.35 ±0.03	0.09 ±0.01	0.02 ±0.01	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0
0.6-0.7	0.02 ±0.01	0.2 ±0.03	0.42 ±0.01	0.26 ±0.03	0.09 ±0.02	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0
0.7-0.8	0.01 ±0.0	0.07 ±0.01	0.26 ±0.02	0.35 ±0.01	0.29 ±0.03	0.03 ±0.0	0.0 ±0.0	0.0 ±0.0
0.8-1.0	0.0 ±0.0	0.02 ±0.01	0.09 ±0.01	0.21 ±0.01	0.49 ±0.01	0.18 ±0.02	0.01 ±0.0	0.0 ±0.0
1.0-1.3	0.0 ±0.0	0.0 ±0.0	0.01 ±0.0	0.05 ±0.01	0.27 ±0.03	0.52 ±0.02	0.15 ±0.03	0.0 ±0.0
1.3-2.2	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.03 ±0.01	0.21 ±0.05	0.73 ±0.06	0.03 ±0.01
2.2+	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.4 ±0.1	0.6 ±0.1

Figure 6.14: Display of confusion matrix of TST regression model. The matrix is normalized by the rows (true values).

The metric values for this model are:

- Root Mean Square Error: 0.162
- Mean Square Error: 0.026
- Mean Absolute Error: 0.113
- Diagonal Average: 0.503
- Success Rate: 0.775

The training time was 4388.100 (s) and the prediction time was 10.917 (s).

Finally, Figure 6.14 illustrates the performance of the TST model. While it outperforms MINIROCKET and other models, further configurations are necessary to fully exploit its potential. Considering the metrics and for simplicity, we will focus on alternative models rather than continuing with TST.

6.1.15. Summary

Table 6.1: Regression metrics table for all models.

	RMSE	MSE	MAE	Diagonal Average	Success Rate
PM	0.180	0.032	0.122	0.516	0.727
Isotonic	0.174	0.030	0.118	0.494	0.758
SGD	0.159	0.025	0.111	0.515	0.765
DT	0.172	0.030	0.116	0.503	0.764
RF	0.165	0.027	0.112	0.511	0.768
XGBoost	0.178	0.038	0.121	0.480	0.770
LightGBM	0.162	0.026	**0.110**	0.510	**0.775**
CatBoost	0.176	0.031	0.119	0.478	0.773
MLP	**0.158**	**0.025**	0.111	0.513	0.772
LSTM	0.167	0.028	0.114	0.504	0.755
GRU	0.163	0.027	0.112	0.510	0.757
Conv1D	0.158	0.025	0.110	**0.518**	0.770
MINIROCKET	0.185	0.034	0.132	0.450	0.736
TST	0.162	0.026	0.113	0.503	0.775

Table 6.2: Time performance table for all regression models.

	Training Time (s)	Prediction Time (s)
PM	**0.000**	**0.007**
Isotonic	2.621	0.010
SGD	24.391	0.028
DT	143.810	0.020
RF	4303.685	0.489
XGBoost	306.999	2.254
LightGBM	202.289	1.617
CatBoost	673.489	0.232
MLP	812.934	17.679
LSTM	459.005	20.445
GRU	458.294	22.700
Conv1D	351.655	16.760
MINIROCKET	1396.839	7.443
TST	4388.100	10.917

6.2. Classification Models

In this section, we test nearly all the regression models in their classification versions. Additionally, we include a few other classic models, such as Logistic Regression and Balanced Random Forest, for completeness.
The implementation of Balanced Random Forest was based on using the imblearn library [24]. Although this isn't a highly imbalanced problem, as shown in Figure 5.2, this approach typically yields good results. As for the other models, minimal changes were made to adapt them to a classification problem. For classification models, the recall metric is the same as the Diagonal Average.

Table 6.3: Classification metrics table for all models.

	Precision	**Recall**	**Accuracy**	**f1-score**
PM	0.514	0.516	0.479	0.514
SGD	0.383	0.416	0.367	0.387
LogisticRegression	0.540	0.514	0.500	0.515
DT	0.527	0.496	0.485	0.503
RF	**0.551**	0.521	**0.503**	**0.529**
BalancedRF	0.515	**0.543**	0.492	0.525
XGBoost	0.499	0.462	0.459	0.473
LightGBM	0.506	0.488	0.476	0.492
CatBoost	0.510	0.461	0.460	0.474
MLP	0.542	0.513	0.500	0.516
LSTM	0.541	0.516	0.497	0.521
GRU	0.541	0.517	0.501	0.522
Conv1D	0.544	0.516	0.501	0.521
MINIROCKET	0.494	0.459	0.446	0.470
TST	0.519	0.506	0.484	0.507

Table 6.4: Custom metrics table for classification models.

	Diagonal Average	**Success Rate**
PM	0.516	0.727
SGD	0.421	0.662
LogisticRegression	0.514	0.715
DT	0.496	0.718
RF	0.521	0.728
BalancedRF	**0.543**	**0.742**
XGBoost	0.462	0.703
LightGBM	0.488	0.716
CatBoost	0.461	0.698
MLP	0.513	0.706
LSTM	0.516	0.722
GRU	0.517	0.720
Conv1D	0.516	0.713
MINIROCKET	0.459	0.702
TST	0.506	0.713

Table 6.5: Time performance table for all classification models.

	Training Time (s)	**Prediction Time (s)**
PM	**0.000**	**0.040**
SGD	224.652	0.087
LogisticRegression	271.257	0.135
DT	166.448	0.040
RF	844.690	2.276
BalancedRF	771.556	6.586
XGBoost	600.532	5.969
LightGBM	667.431	12.716
CatBoost	7627.385	1.624
MLP	561.391	30.452
LSTM	962.156	38.071
GRU	659.386	20.984
Conv1D	471.483	20.962
MINIROCKET	1431.916	7.460
TST	4315.641	10.359

Given the results from regression and classification tasks, we need to select the best models for hyperparameter optimization and adapt the model to this problem effectively. Although some metrics, such as mean squared error versus accuracy, are not directly comparable, we will use Diagonal Average and Success Rate to select a few models. Additionally, the European Southern Observatory uses continuous prediction, requiring a singular seeing value instead of a range as in classification tasks. For simplicity and consistency, we will select a few regression models and one classification model and report the results.

The selected models were chosen based on their metrics and their capability to modify hyperparameter settings, making parameter search worthwhile. That being said, the models we are going to optimize are the following: LightGBM, MLP, Conv1D, GRU, and Balanced Random Forest. The next section will provide details about this process and report the final results for these models.

6.3. Hyperparameters Optimization

In this section, we will proceed to search for the best hyperparameters of our top-performing models in order to achieve better performance on this problem. For this task, we will utilize the Optuna[7] library, which speeds up the search significantly compared to exhaustive searches over parameter grids.

The procedure we are going to follow differs slightly from the previous cross-validation using time series splits. This time, we will train the model using all periods from p97 to p107 and reserve p108 and p109 as the validation set. This approach aims to accelerate the hyperparameter search by using a single training and validation set with a larger number of samples. This is different from the initial splits, where the train set was smaller and may not have yielded the best performance. Additionally, this approach is chosen instead of randomly selecting a previous split because the five different splits used for training and validation sets often yield significantly different metrics. We aim to avoid selecting a random split that might result in a better model simply because it was "easier" to predict. This variation can be attributed to natural changes in weather and optical turbulence, which make some time periods easier to predict than others.

The objective function to optimize in this case is the mean squared error (MSE). While it could be the RMSE, the square root function is strictly increasing, allowing us to optimize the MSE and achieve the same results without additional calculations. The decision to use the MSE as the objective function is also based on the fact that the Success Rate might not be optimal; the model could consistently predict high seeing values and still meet the requirements of a particular observation. Additionally, we will not use Diagonal Average because we do not value each range equally. We understand that high ranges, such as (1.3, 2.2) and 2.2+, are less important compared to low ranges like (0.15, 0.5), (0.5, 0.6), and so on.

The variables we focus on in our search are hyperparameters specific to various machine learning models, such as the number of iterations, learning rate, and number of estimators for tree-based models. For deep learning models, the focus will be on parameters like the number of epochs, layer configurations, different architectures, and loss functions.

Once the grid search for these parameters is completed, we will evaluate the models on the test set, which consists of data from p110 and p111 that the model has not seen before,

[7] https://optuna.org/

and draw final conclusions.

The next table shows the final performance of the models in p110 and p111.

Table 6.6: Metrics for the best models, with the actual model PM included for comparison.

	RMSE	MSE	MAE	Diagonal Average	Success Rate
PM	0.176	0.031	0.121	0.507	0.723
LightGBM	**0.151**	**0.023**	**0.106**	0.502	**0.753**
MLP	0.158	0.025	0.112	0.468	0.741
LSTM	0.160	0.026	0.110	0.478	0.739
GRU	0.159	0.025	0.112	0.494	0.721
Conv1D	0.156	0.026	0.108	0.477	0.726
BalancedRF	-	-	-	**0.540**	0.726

Given the results in Table 6.6, we conclude that LightGBM is the best model for regression, while BalancedRF is the best for classification. It is worth mentioning that BalancedRF does not allow much hyperparameter optimization. We have already made the few modifications possible to achieve the best results.

Figure 6.15 and 6.16 shows different visualizations of the hyperparameter search process. We can observe the parallel coordinates plot representing every combination of hyperparameters and the value they obtain for the objective function.
Additionally, we can see the feature importance, which represents how important it is for the model to change each hyperparameter value.

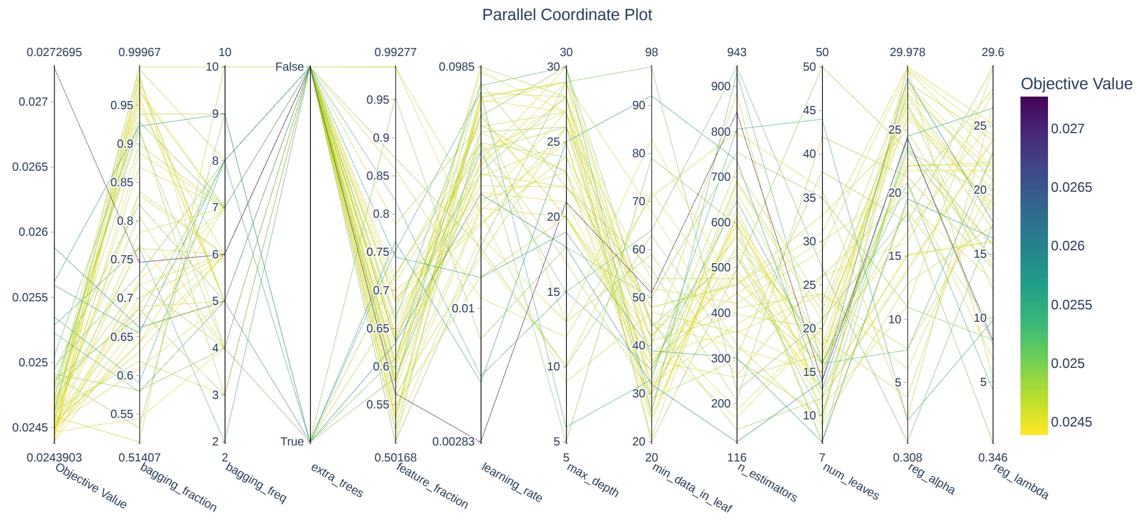


Figure 6.15: Parallel coordinates plot for 60 trials of LightGBM model. Plot restricted to objective values ≤ 0.028 for visualization purposes.

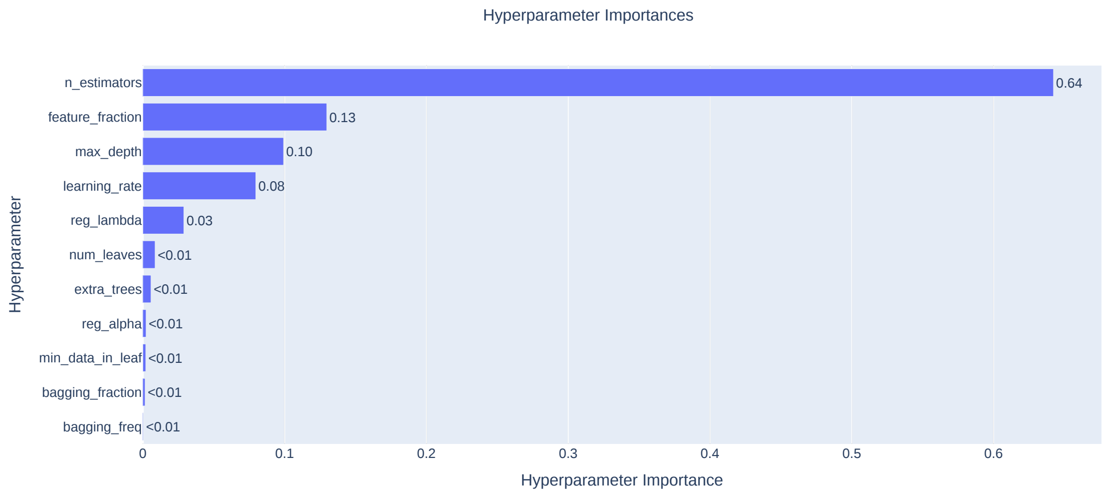


Figure 6.16: Hyperparameter importance plot for 60 trials of LightGBM model.

Here we conclude that, based on the previous figures, the changes in the *n_estimators* was the most influential to the model. This isn't a surprise for tree-based models. On the other hand, we observe that the model tried many combinations and achieved a very low objective value (MSE), making it difficult to evaluate a model with just one set of hyperparameters. Therefore, we can only rely on the minimum MSE metric to choose a specific combination.

Table 6 shows the hyperparameter grid search that we fed into the Optuna study. We also display the data type for each parameter and the optimal value found. It is worth mentioning that we tried other hyperparameters, such as *boosting_type* and others, but the model showed significantly high values, worsening the results.

Table 6.7: Hyperparameter grid search for LightGBM regressor.

	Hyperparameter space	dtype	Optimal value
n_estimators	[50, 1000]	int	755
learning_rate	[0.001, 0.1]	float	0.028
num_leaves	[7, 50]	int	12
max_depth	[5, 30]	int	27
min_data_in_leaf	[20, 100]	int	89
feature_fraction	[0.5, 1.0]	float	0.681
bagging_fraction	[0.5, 1.0]	float	0.616
bagging_freq	[0, 10]	int	5
reg_alpha	[0, 30]	float	26.586
reg_lambda	[0, 30]	float	15.712
extra_trees	[False, True]	bool	False

6.4. Multi-Output Problem

Having selected the best model in the previous section, we now move on to addressing the multi-output problem. This involves training and predicting multiple vectors instead of just one. Some models inherently support this capability, while others do not. For models that do not support multi-output directly, the solution is to create multiple models, each designed to predict its corresponding output vector.
In this case, we have three target variables: “targetMin1hr”, “targetMax1hr”, and “targetSeeing1hr”, with the latter being the target used in all previous tests as it is considered the most important one.

These additional target variables are originally created to address the main challenge of predicting seeing values. While the concept came from the European Southern Observatory, this is one of the first attempts to forecast these values. However, as shown in the EDA section, the difference in sample sizes across different seeing ranges could be a challenge in training a robust model. Therefore, we need to interpret the results cautiously. For example, a model that only predicts the “targetMin1hr” variable within the range of 0.15-0.5 might yield satisfactory results, but it may not be practically useful.

To address this problem, the solution is quite simple: training a model for each target variable. In this case, we chose the optimal LightGBM regressor, which was optimized with the main target variable. We wanted to see if this model could also effectively predict the other two targets. Therefore, we used the MultiOutputRegressor[8] class from the sklearn library.

Here in Table 6.8, we can see the results of the optimal model trained for each target variable. For obvious reasons, the model has the same result for the targetSeeing1hr variable. The other two targets are displayed, and we can conclude that the targetMin1hr variable is easier to predict than the targetMax1hr variable. However, it is difficult to compare them directly because the data itself is significantly different. Additionally, we would need a well-defined baseline to determine if the model performed better than some naive prediction or similar approach.

Table 6.8: Regression metrics table for the three target variables.

	targetSeeing1hr	targetMin1hr	targetMax1hr
RMSE	0.151	0.117	0.292
MSE	0.023	0.014	0.085
MAE	0.106	0.081	0.197
Diagonal Average	0.502	0.414	0.342
Success Rate	0.753	0.726	0.811

Finally, the reason this approach wasn’t fully explored is that these target variables didn’t represent reality very well. First, observation requirements don’t ask for a minimum seeing,

[8] https://scikit-learn.org/stable/modules/generated/sklearn.multioutput.MultiOutputRegressor.html

only a maximum value, so targetMin1hr might not be the optimal target in this problem, though it could provide some information in a specific prediction. Second, although targetMax1hr could be a good target, it is currently very sensitive to seeing values because it is calculated as the absolute maximum over an hour. This might not be robust for an entire hour, as observations with a low average seeing value are still acceptable even if the seeing increases slightly for a few minutes.

6.5. Failed Approaches

As mentioned previously in the methodology, there were numerous approaches that we attempted which did not work, or where we ultimately selected a specific configuration to use. Additionally, some models we tested had the potential for better results but did not perform well in the end. This section explains the approaches we tried and why they did not succeed in a significant way.

First, we aimed to capture the features of absolute humidity, relative humidity, and temperature in a smart way to reduce computational calculations. To achieve this, we used the PCA algorithm as explained in the EDA section. However, this approach did not improve our model's results. Instead, using a few raw columns proved to be a better option. Based on the data and metrics, we decided to remove the PCA step.
Another feature engineering step that was modified multiple times was the EMA (Exponential Moving Average) calculations for the features. We experimented with most of the features and ultimately selected the ones listed in Annex A.2. Additionally, the parameters for the weighted calculations were finalized after several trials with different weights and spans.

Regarding the models themselves, we also tried different approaches. First, we attempted a residual model, which involved fitting a linear regressor to capture the trend of the data and then fitting another non-linear regressor to capture the residuals of that prediction. The final prediction was made by combining these two models. However, this approach did not perform better than a simple linear least squares model with L_2 regularization (Ridge regression).
Another method we tried was stacking regressors[9]. This involved defining a few estimators and then using a final regression model to make the final predictions. Similar to the residual model, the stacked model performed well, but it was not significantly better than other models and it was more complex. Therefore, we decided not to consider these methods in the subsequent steps.
Other commonly used models that we did not include are Support Vector Machines (SVM) and k-nearest neighbor (KNN). The main reasons for this are the very long training time required by SVM and the long prediction time required by the KNN model. These time-related reasons are the primary motives for not including them in this thesis.
Additionally, we experimented with using a windowed dataframe as input for different models. However, this approach did not improve the metrics and only increased the training time.

Regarding more advanced models, we evaluated the performance of convolutional models and naturally tried the Temporal Convolution Network (TCN [25]). However, we did not

[9] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.StackingRegressor.html

observe better performance with the TCN compared to the standard CNN. Therefore, in favor of simplicity, we decided to focus on the CNN.
Additionally, given the typically good results of MLP and LSTM networks and the significant contributions of transformers in recent years, we tried MLP and LSTM networks with Attention modules [8]. While these models performed better than the current stationary model in some metrics, they did not outperform other approaches such as LightGBM or CNN.

The last aspect we did not focus on extensively was the multi-output problem, which involved predicting the minimum and maximum values observed within an hour. There were two main reasons for this. First, we encountered an imbalanced problem, which appeared to be caused by sudden jumps in the data, making it a potentially "false" imbalanced. One possible solution could be to capture the minimum or maximum value with a threshold or by ignoring some outliers. Second, the ESO currently does not work with minimum values. Observations only require a maximum value, and even then, the targetMax1hr variable might not be representative of practical operations. For example, if the average seeing is 0.8 arcseconds over an hour, but it briefly spikes to 1.3 arcseconds in one minute, the observation is likely still valid despite the momentary increase in seeing. Perhaps a classification approach could be useful in this case, as it provides a maximum value based on a probability threshold.

6.6. Feature Importance and Interpretability

The final step in this data science process, excluding deployment (which is beyond our current scope), is to show the feature importance. This tells us, in general terms, how the model treats each variable and how much importance they have. We can then interpret the model's behavior both individually and globally to understand how it makes predictions. This approach helps us to understand the model better, preventing it from being perceived as a completely black box algorithm.

The method we will use to present these graphs involves the internal functions that models have for this purpose. For example, tree-based models provide their own feature importance metrics, such as the gain for each split or the frequency with which a variable is used to split the data. Additionally, we will employ an entirely different framework: the Shapley values and the corresponding SHAP[10] library. This framework offers interpretability for a wide range of machine learning models, allowing us to understand how a model made specific predictions on an individual level, as well as providing a global interpretation of the features.

First, we will display the internal feature importance function offered by the LightGBM model. The only parameter that can change in this function is the importance type. The first option, *split*, indicates how many times a feature is used to split the data across all trees. The second option, *gain*, represents the average gain of the feature when it is used in trees. Figure 6.17 and 6.18 illustrates the two types of importance in the LightGBM model.

[10] https://shap.readthedocs.io/en/latest/

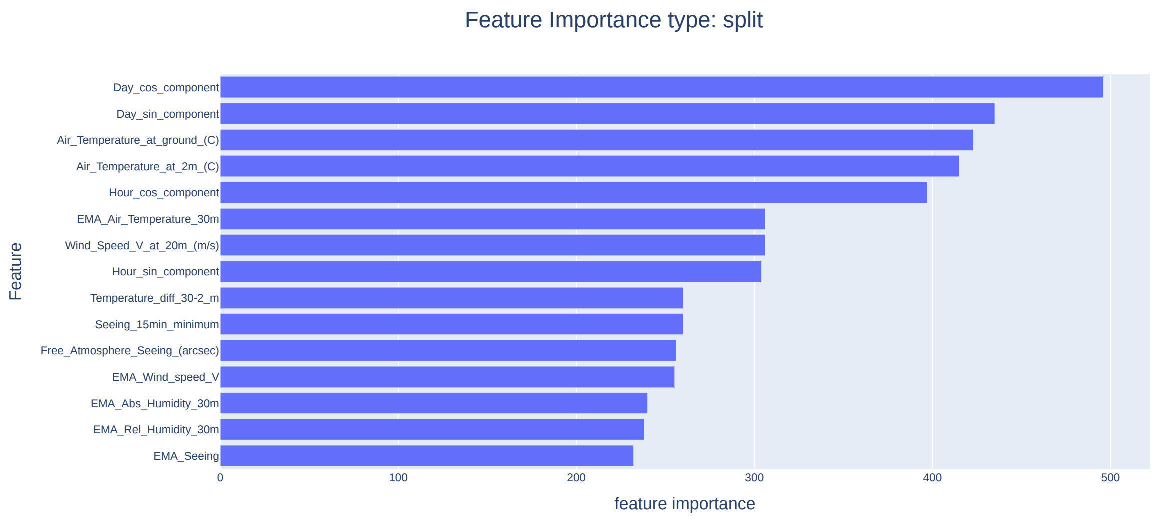


Figure 6.17: Feature importance of the LightGBM regressor using the *split* method. The 15 most important features are displayed.

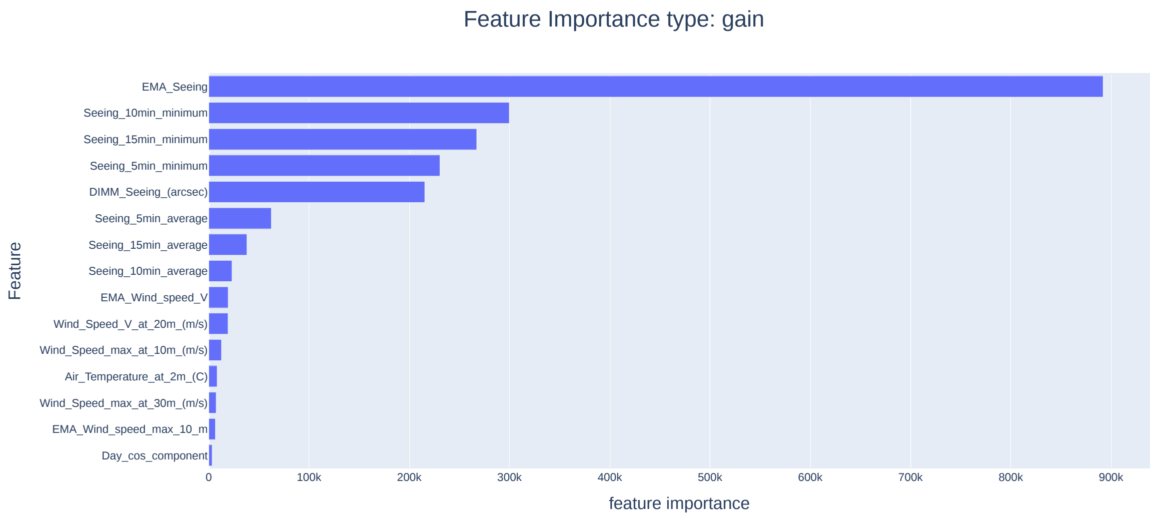


Figure 6.18: Feature importance of the LightGBM regressor using the *gain* method. The 15 most important features are displayed.

In Figure 6.17, we observe that most features with high split importance are not related to seeing. The time of day (with sinusoidal decomposition) is the most important feature in terms of split importance. This indicates that the time of night plays a more significant role in the decision-making process for seeing predictions than other variables. On the other hand, Figure 6.18 shows that many seeing-related features are very important, including also some wind variables and a temperature variable. However, these are not as significant as the most important variable in gain terms: the EMA Seeing. This variable is an exponential moving average of the seeing column. This makes sense since seeing highly depends on past values, making it difficult for an external variable to explain seeing better than its past values.

Now, speaking of the SHAP values calculations, we will use a sample of 100 000 data points from the test set which is sufficient to explain the entire dataset. This ensures that the model interpretation focuses on predictions for the final periods p110 and p111. It's important to note that these calculations are computationally expensive, which is why we do not use the entire dataset for this purpose. Next, with everything calculated, Figure 6.19 presents the general contribution of the 20 most important variables. On the X-axis, we have the SHAP values, which translate into the impact on model predictions. On the Y-axis, we have the input features sorted by importance. The colors represent the value of each specific feature, with blue indicating low values and red indicating high values. The violin shapes show the distribution of the data points used to calculate the SHAP values. This plot indicates that higher values of seeing-related features, such as EMA Seeing and EMA Wind Speed V, are more important to the model.

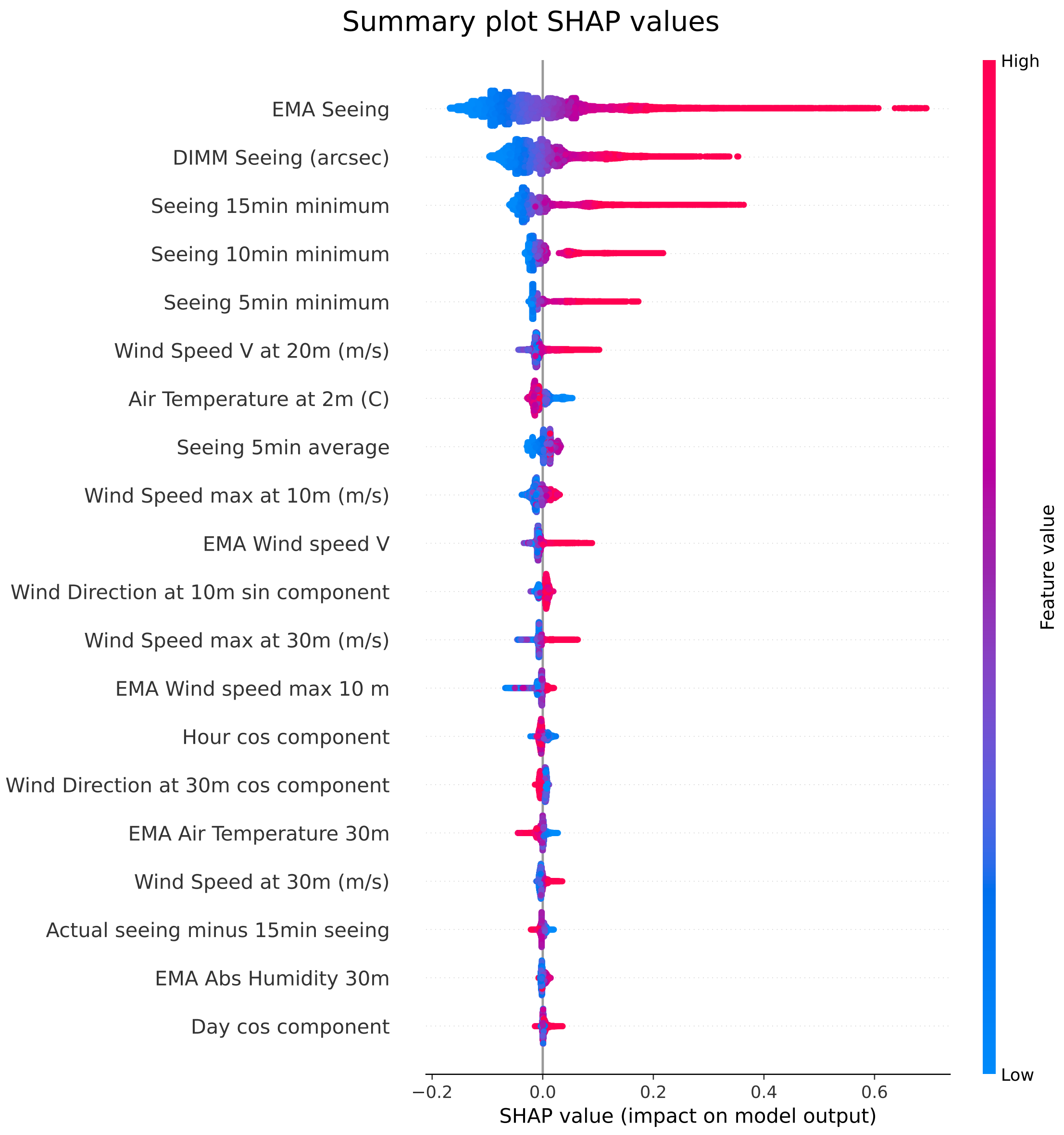


Figure 6.19: Summary plot of the 20 most important variables by SHAP values.

To see the exact relationship of the most important features with the model output, we can see Figure 6.20. This plot shows the dependence of the selected features on the corresponding SHAP values (impact on model output). The feature used to color the dependence plot is chosen by the algorithm itself, based on the calculation of the strongest SHAP interaction values.

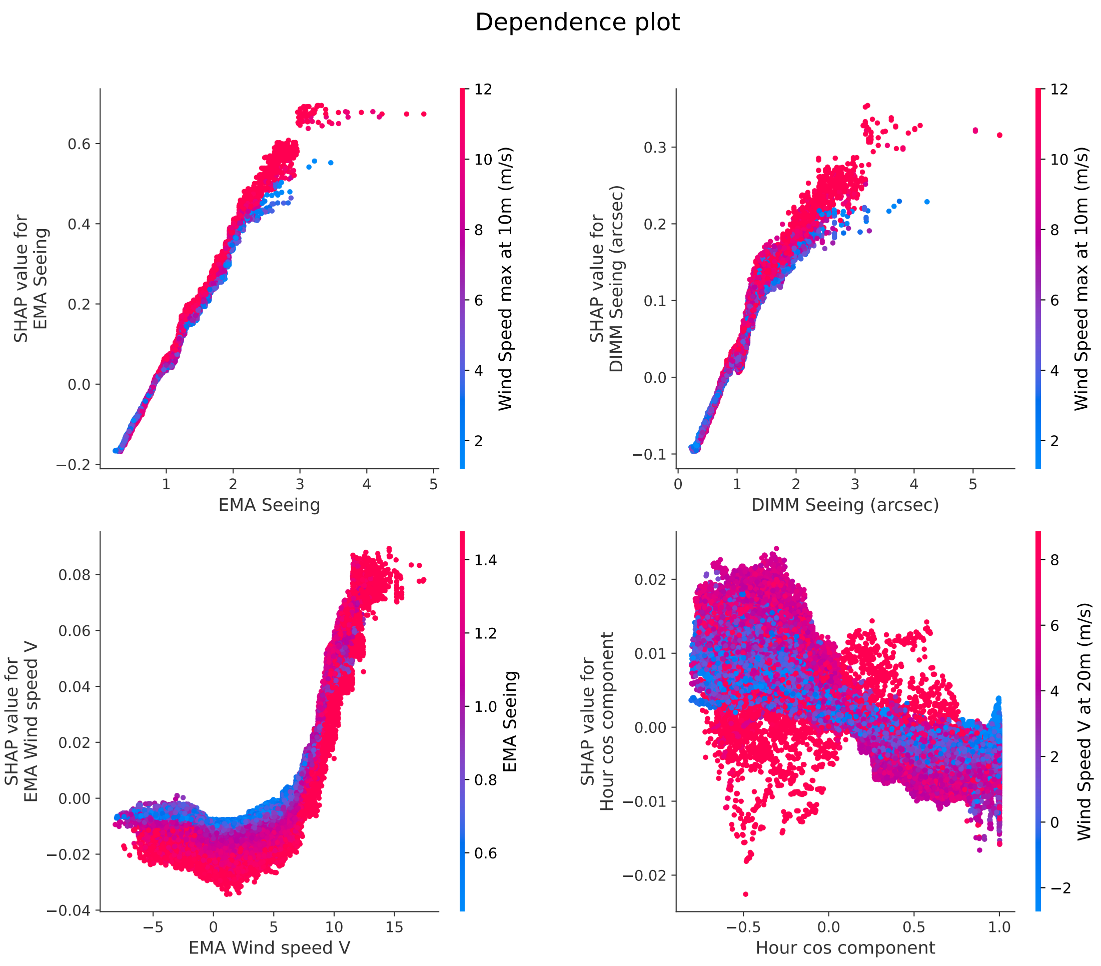


Figure 6.20: Dependence plot for EMA Seeing, DIMM Seeing (arcsec), EMA Wind speed V and Hour cos component features.

In Figure 6.20, we observe how the most important variables have varying impacts on the model output depending on their values. Additionally, related variables are represented in different colors. For example, we notice that the SHAP values of EMA Seeing and DIMM Seeing (arcsec) are very similar. This similarity makes sense because one variable is directly calculated using the other, and the color representation in the plot follows this relationship. From this, we can conclude that the seeing values exhibit a linear relationship with respect to their impact on model outputs, with slightly greater impacts observed for higher wind speed values.
On the other hand, the EMA Wind speed V variable exhibits a nonlinear shape where higher values indicate greater impact and higher values of EMA Seeing are spread uniformly across the plot, while lower values cluster near zero on the Y-axis. Additionally, variables like the Hour cos component show a less distinct impact shape, with colors failing to clearly group the data and provide informative insights.

6.7. Simulations at the European Southern Observatory

Given all the results obtained in the previous sections, we can now compare the LightGBM model with the stationary model in terms of practical use. To explain everything in detail, we first need to clarify that this comparison is about short-term scheduling, that is, scheduling an execution for one or two hours into the future.
First, the Paranal Science Operations (PSO) filter, rank, and select the optimal Observation Block (OB) at any time t for execution at each telescope. This process is known as Short Term Scheduling (STS) of OBs. The ranks can be one of three types: A, B, or C. Typically, A and B ranks are grouped together because they represent important OBs. The only difference is that an A-ranked OB, if not observed, will carry forward into the following semesters until it is observed, whereas a B-ranked OB does not have this benefit. After filtering, ranking, and selection, the observation is executed and then graded. The grade refers to various criteria and is also expressed with the letters A, B, and C. If all of these criteria are met, the OB is graded as A. If one or more of these conditions are not met by 10% or less, the OB is ranked as B. Otherwise, the OB is ranked as C, which is considered a loss of telescope time, meaning a failed execution. If an OB is ranked C, it will return to the queue to be observed again.

The ranking and grading of OBs serve not only to maximize the number of A/B grade OBs but also to minimize the number of C grade OBs, thereby reducing the loss of telescope time. The rank of each OB is important because ESO prioritizes observing A/B ranked OBs and places less emphasis on C ranked OBs, even if a C ranked OB could potentially be an A/B grade observation. For example, if two different prediction models, X and Y, complete 20 hours and 18 hours of observations respectively, but 10 of those 20 hours from model X are from A/B ranked OBs and 16 of the 18 hours from model Y are from A/B ranked OBs, model Y is preferred. Therefore, the rank of each OB is as important as the grade.

Based on the information provided, we proceed to simulate the LightGBM model to make predictions for periods p99 to p111. The model was trained using all available past data. For instance, for period p100, the training data included periods p97, p98, and p99, and so on. Note that period p97 and p98 are not in the simulations due to data problems and p105 is excluded from this comparison due to the pandemic. The configuration of this model differs slightly from the best combination found in the previous section. The hyperparameters for this optimal model are as follows: 853 estimators, 0.017 learning rate, 26 num leaves, 12 max depth, 55 min data in leaf, 0.622 feature fraction, 0.945 bagging fraction, 9 bagging frequency, 12.837 reg alpha, 3.206 reg lambda, and no extra trees. For the objective, we chose quantile with alpha = 0.65.
The four Unit Telescopes (UTs) at Paranal were used to compare the actual model (PM) with the LightGBM regressor. Additionally, each semester, there are OBs in each queue for each UT. It's important to note that these simulations involve a significant amount of time (2016 to 2023), so changes in weather conditions and their evolution are also major factors affecting the performance of each model.

The details of the LightGBM performance by period are provided in Annex C. These tables show the hours gained or lost by this ML model in comparison to the current stationary model. For A/B graded OBs, a higher value is better, whereas for C graded OBs, a lower value is better. Table 6.9 shows the average performance of all periods by UT:

Table 6.9: Hours gained and lost on average of each UT for all A/B/C graded and ranked OBs compared to the stationary model.
Rank: Scientific importance.
Grade: Whether the observation was successful or not, depending on astronomical turbulence.

	A/B graded OBs [hours]		**C graded OBs [hours]**	
	A/B rank	C rank	A/B rank	C rank
UT1	16.04	-7.21	-59.83	3.81
UT2	43.12	52.01	-106.36	10.60
UT3	56.54	27.52	-138.48	-0.87
UT4	85.84	41.53	-143.33	-5.20
Sum	201.54	113.85	-448.00	8.34
Per Semester	**18.32**	**10.35**	**-40.73**	**0.76**

This analysis highlights four important aspects. Firstly, for C graded OBs, the only case where the current model (PM) performs better is with the C ranked OBs (the least important), with an average difference of just 0.76 hours per semester, which is a tiny and negligible value. On the other hand, the optimal LightGBM model gains an average of 40.73 hours per semester where the stationary model had a loss of telescope time due to C graded OBs. For the A/B graded OBs, the ML prediction performs significantly better, gaining an average of 10.35 hours per semester for C ranked OBs and 18.32 hours per semester for A/B ranked OBs, which are the most important.

On average, these gains of 28.67 hours translate to more efficient predictions of seeing conditions. Additionally, we gained 40 hours by not observing C graded OBs. This all translates to a model more capable of emptying the queue than the stationary prediction, allowing for new executions of different OBs to be added to the queue, thus achieving even better practical results than those shown in Table 6.9 for observing OBs.

Chapter 7

Conclusion

In this thesis, we have conducted a comprehensive review of the seeing prediction problem for the Very Large Telescope at Paranal in the European Southern Observatory. The problem was approached from a data science perspective, employing mathematical tools to optimize model predictions as much as possible. The previous chapter demonstrates that the optimal model for this task is the LightGBM regressor.

This thesis fulfilled all the specific and general objectives. We gained a deep understanding of the data used and performed extensive feature engineering, which significantly improved the model's performance. Additionally, we tested numerous models to robustly compare their performance and then optimized the better ones to adapt them as effectively as possible to this problem. Finally, we made general interpretations of the best model predictions to achieve a more comprehensive result. In summary, we succeeded in predicting the seeing values, providing better predictions in each period compared to the stationary model.

The LightGBM regressor is the optimal model, following our progression from basic models to advanced applications like transformers. Additionally, we spent considerable time creating and testing new features to improve the model's performance.
From an academic perspective, the best metrics obtained by this model are 0.151 RMSE, 0.023 MSE, 0.106 MAE, 0.502 Diagonal Average and 0.753 Success Rate. Compared to the current PM model used at the ESO, this translates to a -0.99% decrease in Diagonal Average but improvements of 14.2%, 25.81%, 12.4% and 4.15% in RMSE, MSE, MAE and Success Rate respectively.
Then, conducting an interpretability section allowed us to evaluate how the model makes predictions and identify the most important features. This information is relevant and valuable for this work and even for future projects.

From a practical use perspective, a slight modification to this model resulted in an average gain of 18.32 hours of A/B rank and 10.35 hours of C rank OBs meeting the seeing requirements per semester compared to the stationary model. Additionally, we gained approximately 40 hours of telescope time by not observing C graded OBs compared to the stationary model. This improvement enhances our capability to clear the observation queue and add new OBs for execution.

7.1. Limitations

The model's limitations are very clear. On the one hand, we face a natural (and irreparable) issue: seeing values are highly chaotic and can change drastically within seconds or minutes, making accurate predictions for one hour ahead quite difficult even if its the average. Also, since seeing is usually measured at night, we lack data during the day, resulting in discontinuous and irregular time series. This issue could potentially be resolved with an instrument capable of measuring these variables throughout the day, a solution that the ESO is reportedly working on.

On the other hand, we face data limitations. The available data only starts from 2016, even though the ESO has been collecting data for many more years. This gap is due to the implementation of new and more accurate instruments.

Finally, there are specific limitations of the LightGBM regressor model. Some of these limitations are inherent to its architecture, which doesn't allow for much flexibility to experiment with new approaches. Other limitations involve hyperparameters that can significantly impact performance, although they are fewer in number compared to other models. This is where deep learning models have an advantage over classic machine learning models; with more resources, data, and architectural changes, it is likely that a DL model will eventually outperform the LightGBM regressor or other classic algorithms. However, this work demonstrates that for a simple regression problem, a classic ML model still performs best and is a quick method to implement.

7.2. Impact and Contribution

This work originated from the idea that seeing could be predicted more accurately than just a ten-minute average. However, it also had another purpose: using VLT data and predictions to develop a new model that could potentially be applied to the ELT in the future. That being said, our efforts have been directed towards exploring every possible solution comprehensively and extensively within the given time frame. With the final model, we conclude that there is a significant improvement compared to the current model. Implementing this model could result in gaining more telescope time due to better seeing predictions. Additionally, it would reduce the time the telescope is operational but not meeting the seeing requirements, thus minimizing the loss of telescope time.

7.3. Future Work

There are several potential ways to improve the seeing predictions. These include:

- Using data from nearby locations: Seeing and wind values from nearby areas could significantly enhance the accuracy of predictions.
- Exploring new features: This could involve both, variables measured by instruments and those generated through creative feature engineering.
- Including daytime data: Adding data collected during the day could help address the issue of discontinuous and irregular time series.

- Retraining the model every period (semester): This approach could ensure more accurate and up-to-date predictions.
- Implementing state-of-the-art models for time series or regression problems: Although challenging due to the rapid development of advanced techniques like transformers, this approach holds potential for significant improvements.

Moreover, there could be other approaches to this problem. In this thesis, a classification task was examined; however, it wasn't explored to its full potential due to the manner in which the ESO operates. Another approach, which could be more detailed but more complex and computationally expensive, is to apply a forecasting model that could predict N (≥ 60) points into the future. This would allow for the calculation of any desired values, including the 60-minute average, which is the principal target variable in this thesis.

Finally, large-scale meteorological events could significantly impact this work and similar projects. The transition from La Niña to El Niño has led to an increase in extreme weather events, including higher temperatures and changes in precipitation patterns. This and other big events suggests that prediction projects should be updated periodically.

Bibliography

[1] Sarazin, M. y Roddier, F., "The ESO differential image motion monitor", Astronomy and Astrophysics, vol. 227, pp. 294–300, 1990.

[2] Oguiza, I., "tsai - a state-of-the-art deep learning library for time series and sequential data". Github, 2023, https://github.com/timeseriesAI/tsai.

[3] Lafore, J.-P., Stein, J., Asencio, N., Bougeault, P., Ducrocq, V., Duron, J., Fischer, C., Héreil, P., Mascart, P., Masson, V., Pinty, J.-P., Redelsperger, J.-L., Richard, E., y Arellano, J., "The Meso-NH Atmospheric Simulation System. Part I: Adiabatic formulation and control simulations", Annales Geophysicae, vol. 16, 1997, doi:10.1007/s005850050582.

[4] Skamarock, W., Klemp, J., Dudhia, J., Gill, D., Barker, D., Wang, W., y Powers, J., "A Description of the Advanced Research WRF Version 3", vol. 27, pp. 3–27, 2008.

[5] Tatarskii, V. I., The effects of the turbulent atmosphere on wave propagation. 1971.

[6] Dempster, A., Schmidt, D. F., y Webb, G. I., "MINIROCKET: A Very Fast (Almost) Deterministic Transform for Time Series Classification", arXiv e-prints, p. arXiv:2012.08791, 2020, doi:10.48550/arXiv.2012.08791.

[7] Zerveas, G., Jayaraman, S., Patel, D., Bhamidipaty, A., y Eickhoff, C., "A Transformer-based Framework for Multivariate Time Series Representation Learning", arXiv e-prints, p. arXiv:2010.02803, 2020, doi:10.48550/arXiv.2010.02803.

[8] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., y Polosukhin, I., "Attention is all you need", in Advances in Neural Information Processing Systems (Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., y Garnett, R., eds.), vol. 30, Curran Associates, Inc., 2017, https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[9] Cherubini, T., Lyman, R., y Businger, S., "Forecasting seeing for the Maunakea observatories with machine learning", Monthly Notices of the Royal Astronomical Society, vol. 509, pp. 232–245, 2021, doi:10.1093/mnras/stab2916.

[10] Hou, X., Hu, Y., Du, F., Ashley, M., Pei, C., Shang, Z., Ma, B., Wang, E., y Huang, K., "Machine learning-based seeing estimation and prediction using multi-layer meteorological data at dome a, antarctica", Astronomy and Computing, vol. 43, p. 100710, 2023, doi:https://doi.org/10.1016/j.ascom.2023.100710.

[11] Milli, J., Rojas, T., Courtney-Barrer, B., Bian, F., Navarrete, J., Kerber, F., y Otarola, A., "Turbulence nowcast for the Cerro Paranal and Cerro Armazones observatory sites", in Adaptive Optics Systems VII (Schreiber, L., Schmidt, D., y Vernet, E., eds.),

vol. 11448 de Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, p. 114481J, 2020, doi:10.1117/12.2561364.

[12] Milli, J., Gonzalez, R., Fluxa, P. R., Chacon, A., Navarette, J., Sarazin, M., Pena, E., Carrasco-Davis, R., Solarz, A., Smoker, J., Martayan, C., Melo, C., Sedaghati, E., Mieske, S., Hainaut, O., y Tacconi-Garman, L., “Nowcasting the turbulence at the Paranal Observatory”, arXiv e-prints, p. arXiv:1910.13767, 2019, doi:10.48550/arXiv.1910.13767.

[13] Masciadri, E., Turchi, A., y Fini, L., “Optical turbulence forecasts at short time-scales using an autoregressive method at the Very Large Telescope”, Monthly Notices of the Royal Astronomical Society, vol. 523, pp. 3487–3502, 2023, doi:10.1093/mnras/stad1552.

[14] Breiman, L., “Random Forests”, Machine Learning, vol. 45, pp. 5–32, 2001, doi:https://doi.org/10.1023/A:1010933404324.

[15] Chen, T. y Guestrin, C., “XGBoost: A Scalable Tree Boosting System”, arXiv e-prints, p. arXiv:1603.02754, 2016, doi:10.48550/arXiv.1603.02754.

[16] Ke, Guolin and Meng, Qi and Finley, Thomas and Wang, Taifeng and Chen, Wei and Ma, Weidong and Ye, Qiwei and Liu, Tie-Yan, “Lightgbm: A Highly Efficient Gradient Boosting Decision Tree”, vol. 30, pp. 3146–3154, 2017.

[17] Veronika Dorogush, A., Ershov, V., y Gulin, A., “CatBoost: gradient boosting with categorical features support”, arXiv e-prints, p. arXiv:1810.11363, 2018, doi:10.48550/arXiv.1810.11363.

[18] Olah, C., “Understanding LSTM Networks”, 2015, https://colah.github.io/posts/2015-08-Understanding-LSTMs/ (Accessed 2024-05-30).

[19] Hochreiter, S. y Schmidhuber, J., “Long Short-Term Memory”, Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[20] Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., y Bengio, Y., “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation”, in Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, pp. 1724–1734, Association for Computational Linguistics, 2014, doi:10.3115/v1/D14-1179.

[21] LeCun, Yann and Bengio, Y. and Hinton, Geoffrey, “Deep Learning”, Nature, vol. 521, pp. 436–44, 2015, doi:10.1038/nature14539.

[22] Olah, C., “Conv Nets: A Modular Perspective”, 2014, https://colah.github.io/posts/2014-07-Conv-Nets-Modular/ (Accessed 2024-05-30).

[23] Kingma, D. P. y Ba, J., “Adam: A Method for Stochastic Optimization”, arXiv e-prints, p. arXiv:1412.6980, 2014, doi:10.48550/arXiv.1412.6980.

[24] Lemaître, G., Nogueira, F., y Aridas, C. K., “Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning”, Journal of Machine Learning Research, vol. 18, no. 17, pp. 1–5, 2017, http://jmlr.org/papers/v18/16-365.html.

[25] Bai, S., Zico Kolter, J., y Koltun, V., “An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling”, arXiv e-prints, p. arXiv:1803.01271, 2018, doi:10.48550/arXiv.1803.01271.

ANNEXES

Annex A. Database features

A.1. Initial features

1. Date
2. Target RA [deg]
3. Target DEC [deg]
4. DIMM Seeing [arcsec]
5. Relative Flux RMS
6. Relative Flux RMS base time [s]
7. n_night
8. LHATPRO ID
9. Absolute Humidity [g/m^3] at 0[m]
10. Absolute Humidity [g/m^3] at 10[m]
11. Absolute Humidity [g/m^3] at 30[m]
12. Absolute Humidity [g/m^3] at 50[m]
13. Absolute Humidity [g/m^3] at 75[m]
14. Absolute Humidity [g/m^3] at 100[m]
15. Absolute Humidity [g/m^3] at 125[m]
16. Absolute Humidity [g/m^3] at 150[m]
17. Absolute Humidity [g/m^3] at 200[m]
18. Absolute Humidity [g/m^3] at 250[m]
19. Absolute Humidity [g/m^3] at 325[m]
20. Absolute Humidity [g/m^3] at 400[m]
21. Absolute Humidity [g/m^3] at 475[m]
22. Absolute Humidity [g/m^3] at 550[m]
23. Absolute Humidity [g/m^3] at 625[m]
24. Absolute Humidity [g/m^3] at 700[m]
25. Absolute Humidity [g/m^3] at 800[m]
26. Absolute Humidity [g/m^3] at 900[m]
27. Absolute Humidity [g/m^3] at 1000[m]
28. Absolute Humidity [g/m^3] at 1150[m]
29. Absolute Humidity [g/m^3] at 1300[m]
30. Absolute Humidity [g/m^3] at 1450[m]
31. Absolute Humidity [g/m^3] at 1600[m]
32. Absolute Humidity [g/m^3] at 1800[m]
33. Absolute Humidity [g/m^3] at 2000[m]
34. Absolute Humidity [g/m^3] at 2200[m]
35. Absolute Humidity [g/m^3] at 2500[m]
36. Absolute Humidity [g/m^3] at 2800[m]
37. Absolute Humidity [g/m^3] at 3100[m]
38. Absolute Humidity [g/m^3] at 3500[m]
39. Absolute Humidity [g/m^3] at 3900[m]
40. Absolute Humidity [g/m^3] at 4400[m]
41. Absolute Humidity [g/m^3] at 5000[m]
42. Absolute Humidity [g/m^3] at 5600[m]

43. Absolute Humidity [g/m^3] at 6200[m]
44. Absolute Humidity [g/m^3] at 7000[m]
45. Absolute Humidity [g/m^3] at 8000[m]
46. Absolute Humidity [g/m^3] at 9000[m]
47. Absolute Humidity [g/m^3] at 10000[m]
48. Relative Humidity [%] at 0[m]
49. Relative Humidity [%] at 10[m]
50. Relative Humidity [%] at 30[m]
51. Relative Humidity [%] at 50[m]
52. Relative Humidity [%] at 75[m]
53. Relative Humidity [%] at 100[m]
54. Relative Humidity [%] at 125[m]
55. Relative Humidity [%] at 150[m]
56. Relative Humidity [%] at 200[m]
57. Relative Humidity [%] at 250[m]
58. Relative Humidity [%] at 325[m]
59. Relative Humidity [%] at 400[m]
60. Relative Humidity [%] at 475[m]
61. Relative Humidity [%] at 550[m]
62. Relative Humidity [%] at 625[m]
63. Relative Humidity [%] at 700[m]
64. Relative Humidity [%] at 800[m]
65. Relative Humidity [%] at 900[m]
66. Relative Humidity [%] at 1000[m]
67. Relative Humidity [%] at 1150[m]
68. Relative Humidity [%] at 1300[m]
69. Relative Humidity [%] at 1450[m]
70. Relative Humidity [%] at 1600[m]
71. Relative Humidity [%] at 1800[m]
72. Relative Humidity [%] at 2000[m]
73. Relative Humidity [%] at 2200[m]
74. Relative Humidity [%] at 2500[m]
75. Relative Humidity [%] at 2800[m]
76. Relative Humidity [%] at 3100[m]
77. Relative Humidity [%] at 3500[m]
78. Relative Humidity [%] at 3900[m]
79. Relative Humidity [%] at 4400[m]
80. Relative Humidity [%] at 5000[m]
81. Relative Humidity [%] at 5600[m]
82. Relative Humidity [%] at 6200[m]
83. Relative Humidity [%] at 7000[m]
84. Relative Humidity [%] at 8000[m]
85. Relative Humidity [%] at 9000[m]
86. Relative Humidity [%] at 10000[m]
87. Temperature [K] at 0[m]
88. Temperature [K] at 10[m]
89. Temperature [K] at 30[m]
90. Temperature [K] at 50[m]
91. Temperature [K] at 75[m]
92. Temperature [K] at 100[m]
93. Temperature [K] at 125[m]
94. Temperature [K] at 150[m]
95. Temperature [K] at 200[m]
96. Temperature [K] at 250[m]
97. Temperature [K] at 325[m]
98. Temperature [K] at 400[m]
99. Temperature [K] at 475[m]
100. Temperature [K] at 550[m]
101. Temperature [K] at 625[m]

102. Temperature [K] at 700[m]
103. Temperature [K] at 800[m]
104. Temperature [K] at 900[m]
105. Temperature [K] at 1000[m]
106. Temperature [K] at 1150[m]
107. Temperature [K] at 1300[m]
108. Temperature [K] at 1450[m]
109. Temperature [K] at 1600[m]
110. Temperature [K] at 1800[m]
111. Temperature [K] at 2000[m]
112. Temperature [K] at 2200[m]
113. Temperature [K] at 2500[m]
114. Temperature [K] at 2800[m]
115. Temperature [K] at 3100[m]
116. Temperature [K] at 3500[m]
117. Temperature [K] at 3900[m]
118. Temperature [K] at 4400[m]
119. Temperature [K] at 5000[m]
120. Temperature [K] at 5600[m]
121. Temperature [K] at 6200[m]
122. Temperature [K] at 7000[m]
123. Temperature [K] at 8000[m]
124. Temperature [K] at 9000[m]
125. Temperature [K] at 10000[m]
126. Free Atmosphere Seeing [arcsec]
127. Free Atmosphere Seeing RMS
128. MASS Tau0 [s]
129. MASS Tau0 RMS
130. MASS Theta0 [arcsec]
131. MASS Theta0 RMS
132. MASS Turb Altitude [m]
133. MASS Turb Altitude RMS
134. MASS-DIMM Cn2 fraction at ground
135. MASS-DIMM Seeing [arcsec]
136. MASS-DIMM Tau0 [s]
137. MASS-DIMM Theta0 [arcsec]
138. MASS-DIMM Turb Altitude [m]
139. MASS-DIMM Turb Velocity [m/s]
140. Air Pressure at ground [hPa]
141. Air Pressure Normalised [hPa]
142. Air Temperature at 30m [C]
143. Air Temperature at 2m [C]
144. Air Temperature at ground [C]
145. Air Temperature below VLT [C]
146. Dew Temperature at 30m [C]
147. Dew Temperature at 2m [C]
148. Dew Temperature below VLT [C]
149. Rain intensity below VLT [%]
150. Relative Humidity at 30m [%]
151. Relative Humidity at 2m [%]
152. Relative Humidity below VLT [%]
153. Wind Direction at 30m (0/360) [deg]
154. Wind Direction at 10m (0/360) [deg]
155. Wind Speed at 30m [m/s]
156. Wind Speed at 10m [m/s]
157. Wind Speed U at 20m [m/s]
158. Wind Speed V at 20m [m/s]
159. Wind Speed W at 20m [m/s]
160. Wind Speed max at 30m [m/s]
161. Wind Speed max at 10m [m/s]
162. PM
163. targetSeeing1hr
164. targetMax1hr
165. targetMin1hr

A.2. Final features

1. DIMM Seeing (arcsec)
2. Free Atmosphere Seeing (arcsec)
3. MASS-DIMM Tau0 (s)
4. Air Temperature at 2m (C)
5. Air Temperature at ground (C)
6. Wind Speed at 30m (m/s)
7. Wind Speed at 10m (m/s)
8. Wind Speed V at 20m (m/s)
9. Wind Speed max at 30m (m/s)
10. Wind Speed max at 10m (m/s)
11. Day sin component
12. Hour sin component
13. Day cos component
14. Hour cos component
15. Wind Direction at 30m sin component
16. Wind Direction at 30m cos component
17. Wind Direction at 10m sin component
18. Wind Direction at 10m cos component
19. Seeing 5min average
20. Seeing 5min minimum
21. Seeing 5min maximum
22. Seeing 10min average
23. Seeing 10min minimum
24. Seeing 10min maximum
25. Seeing 15min average
26. Seeing 15min minimum
27. Seeing 15min maximum
28. Actual seeing minus 5min seeing
29. Air temperature at 30m (actual minus 5min)
30. Wind speed at 30m (actual minus 5min)
31. Actual seeing minus 10min seeing
32. Air temperature at 30m (actual minus 10min)
33. Wind speed at 30m (actual minus 10min)
34. Actual seeing minus 15min seeing
35. Air temperature at 30m (actual minus 15min)
36. Wind speed at 30m (actual minus 15min)
37. Temperature diff 30-2 m
38. Wind speed max diff 30-10 m
39. Wind direction sin diff 30-10 m
40. Wind direction cos diff 30-10 m
41. EMA Seeing
42. EMA Air Temperature 30m
43. EMA Abs Humidity 30m
44. EMA Rel Humidity 30m
45. EMA Wind speed V
46. EMA Wind speed max 10 m
47. EMA MASS-DIMM Tau0
48. PM
49. targetSeeing1hr
50. targetMax1hr
51. targetMin1hr

Annex B. Classification models matrices

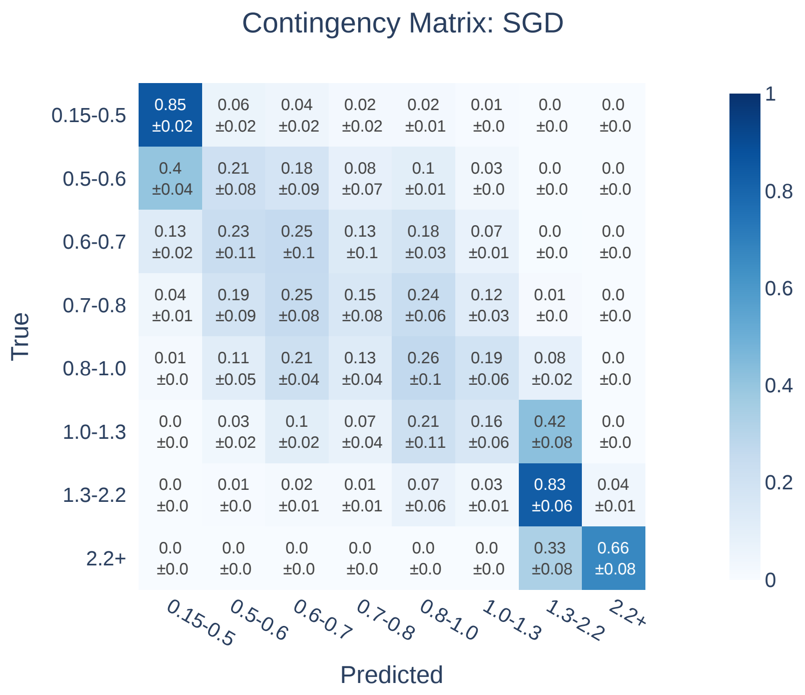


Figure B.1: Display of confusion matrix of SGD classification model. The matrix is normalized by the rows (true values).

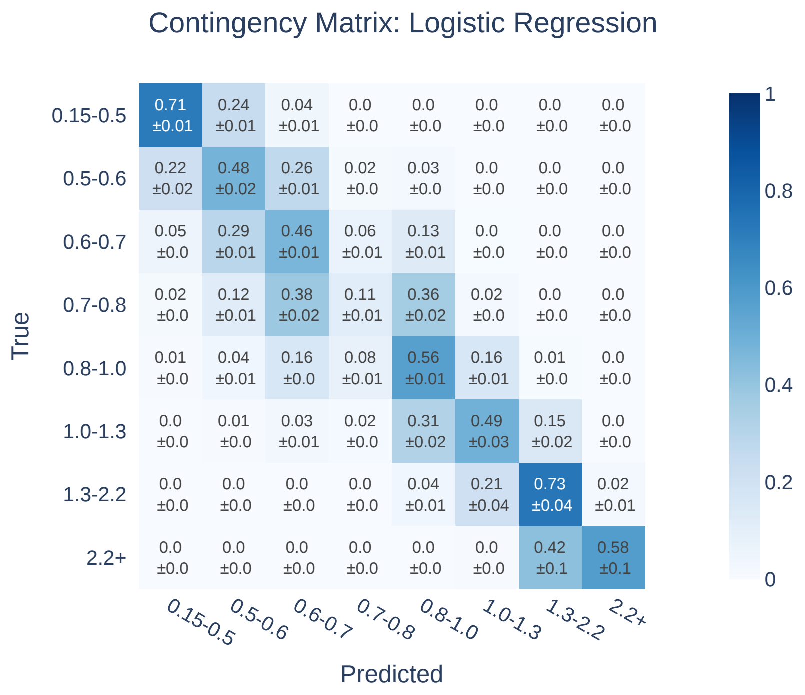


Figure B.2: Display of confusion matrix of Logistic Regression model. The matrix is normalized by the rows (true values).

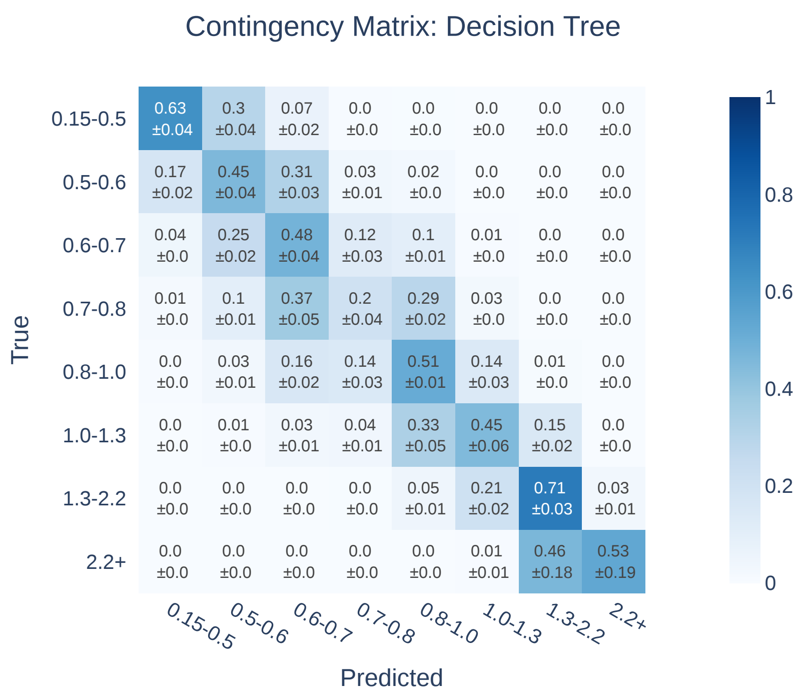


Figure B.3: Display of confusion matrix of Decision Tree classification model. The matrix is normalized by the rows (true values).

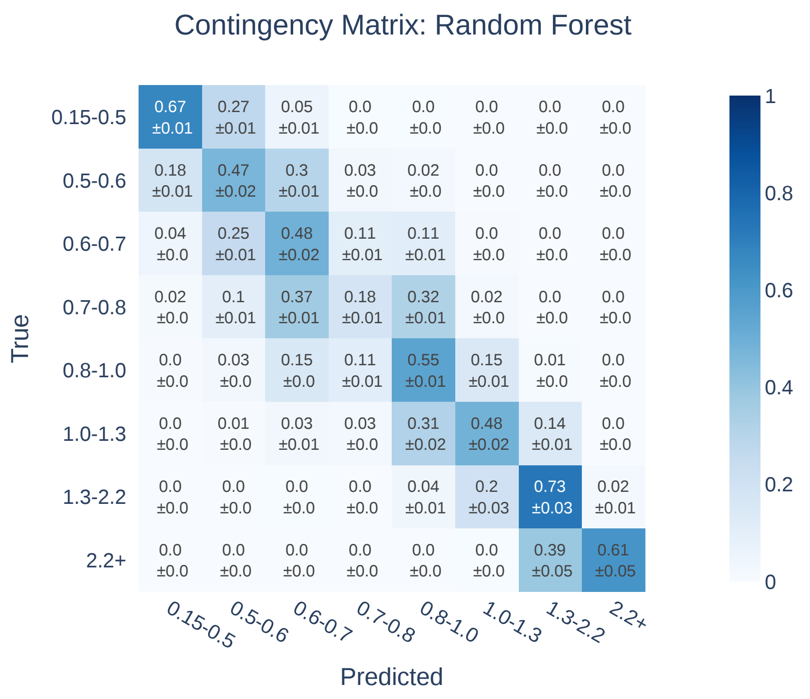


Figure B.4: Display of confusion matrix of Random Forest classification model. The matrix is normalized by the rows (true values).

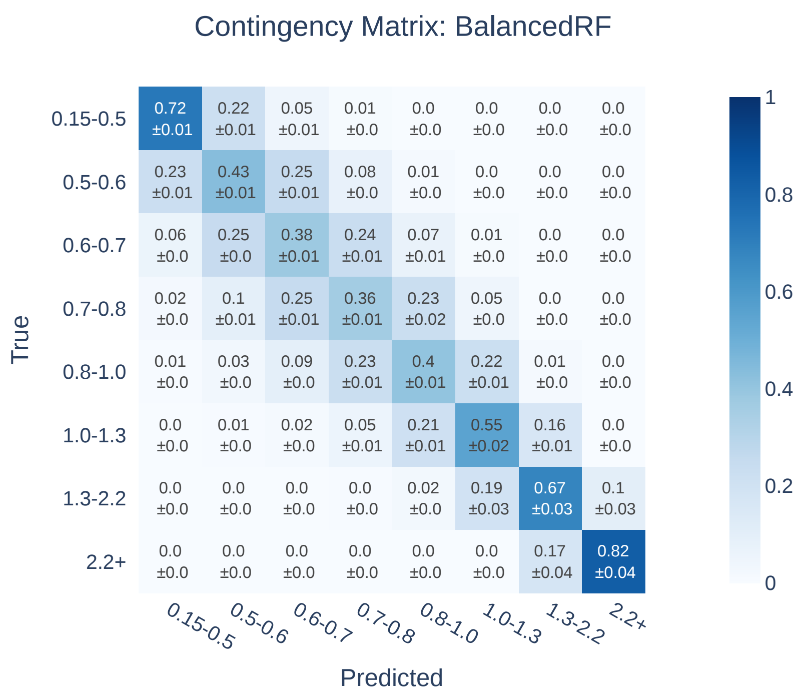


Figure B.5: Display of confusion matrix of Balanced Random Forest classification model. The matrix is normalized by the rows (true values).

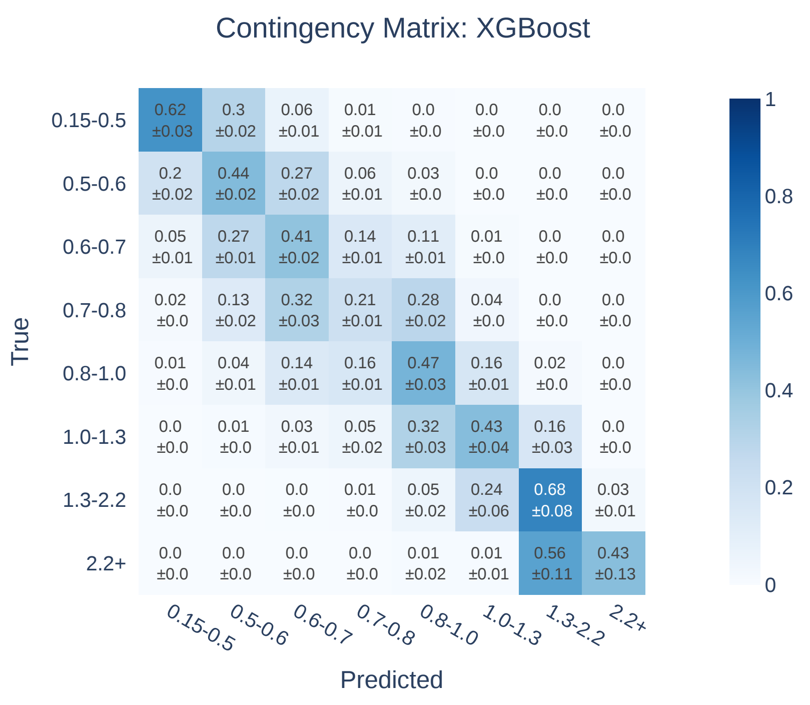


Figure B.6: Display of confusion matrix of XGBoost classification model. The matrix is normalized by the rows (true values).

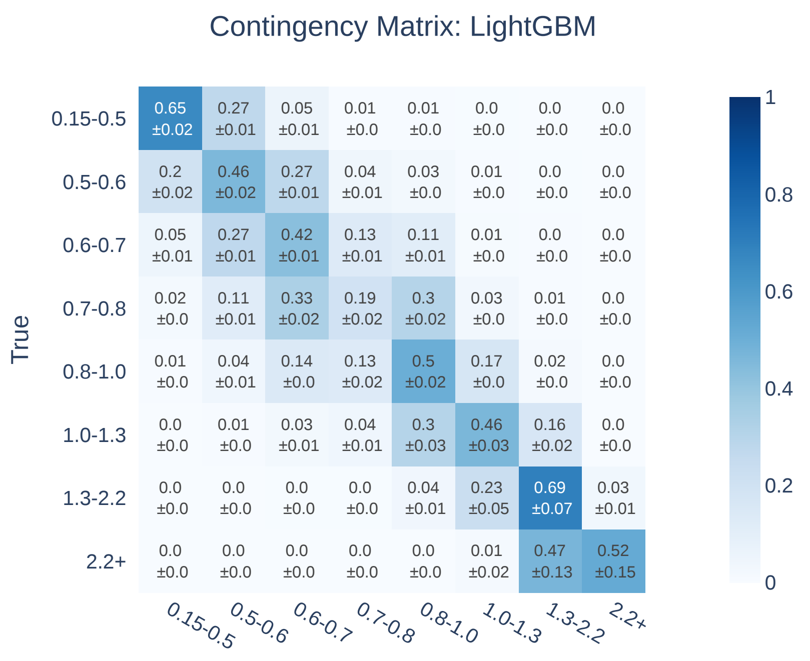


Figure B.7: Display of confusion matrix of LightGBM classification model. The matrix is normalized by the rows (true values).

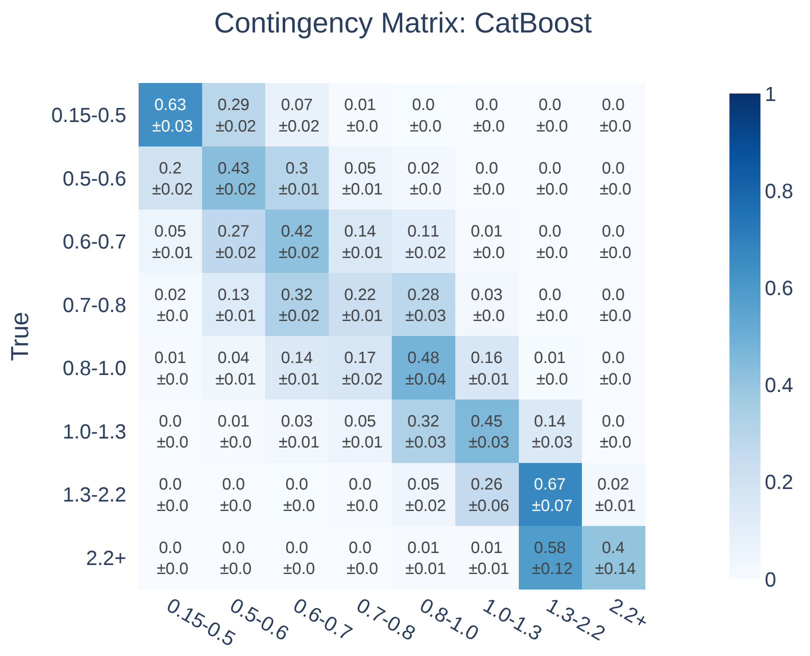


Figure B.8: Display of confusion matrix of CatBoost classification model. The matrix is normalized by the rows (true values).

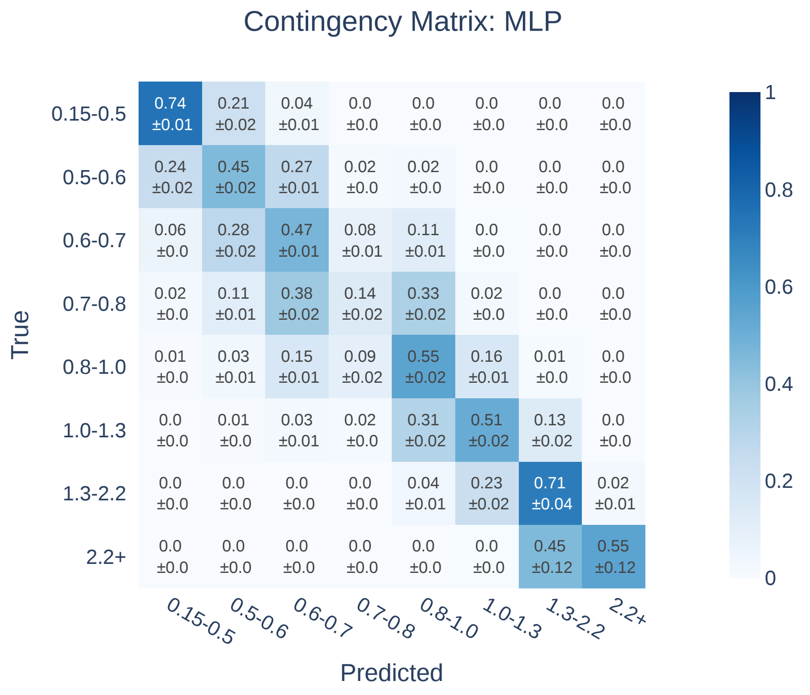


Figure B.9: Display of confusion matrix of MLP classification model. The matrix is normalized by the rows (true values).

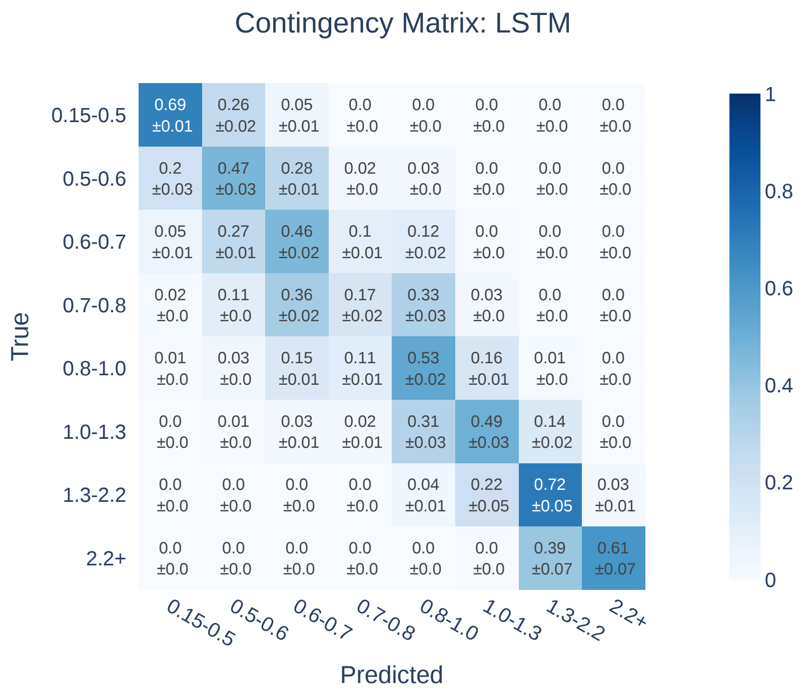


Figure B.10: Display of confusion matrix of LSTM classification model. The matrix is normalized by the rows (true values).

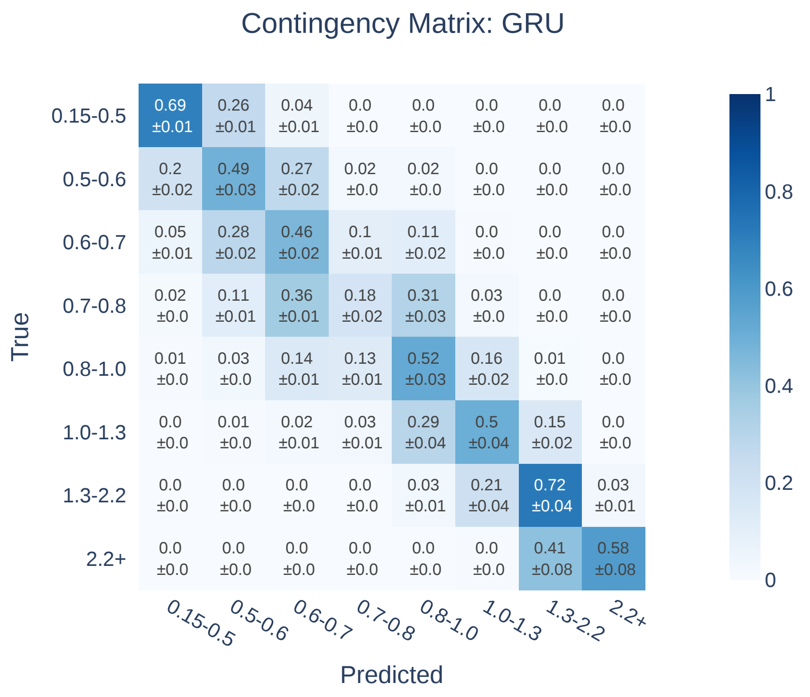


Figure B.11: Display of confusion matrix of GRU classification model. The matrix is normalized by the rows (true values).

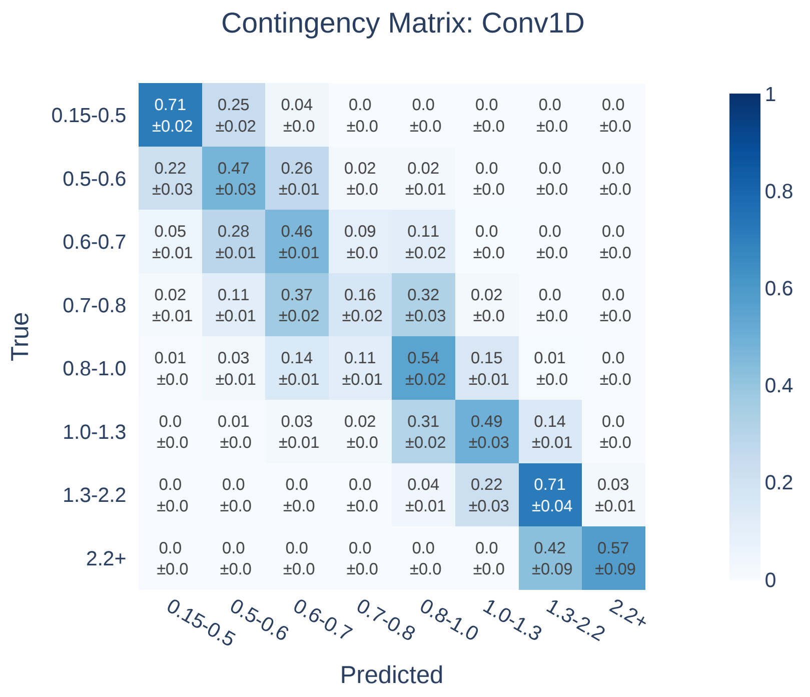


Figure B.12: Display of confusion matrix of Conv1D classification model. The matrix is normalized by the rows (true values).

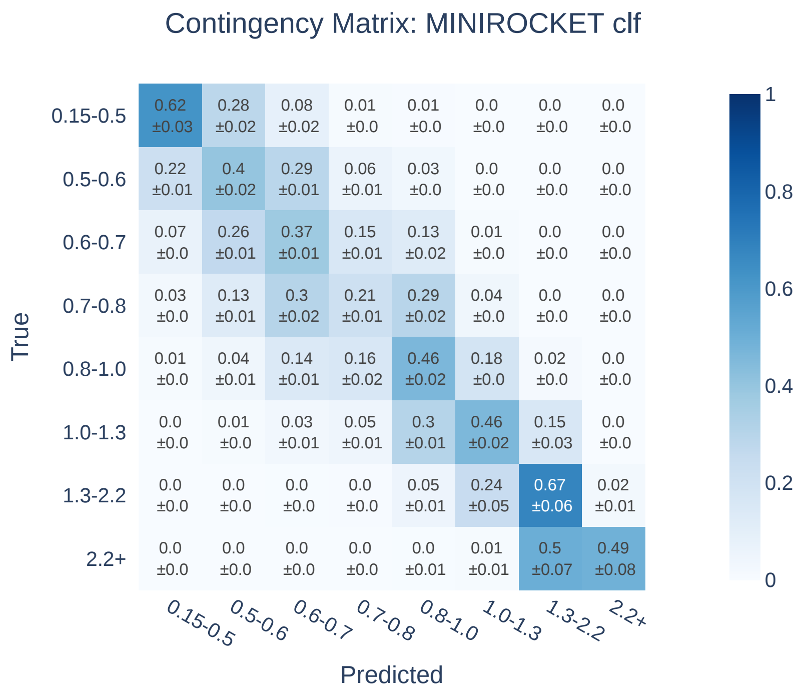


Figure B.13: Display of confusion matrix of MINIROCKET classification model. The matrix is normalized by the rows (true values).

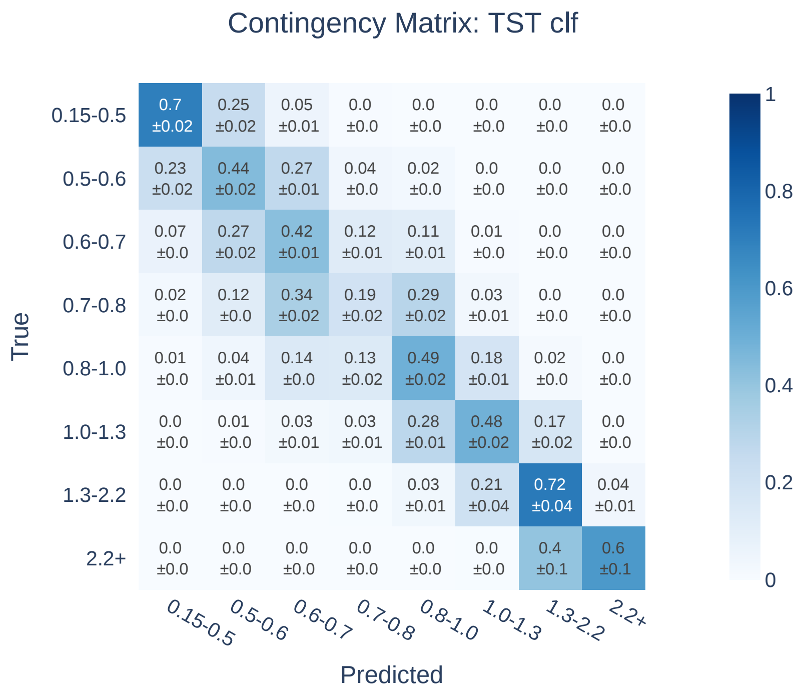


Figure B.14: Display of confusion matrix of TST classification model. The matrix is normalized by the rows (true values).

Annex C. Simulations per period

Table C.1: Hours gained/lost of LightGBM versus PM for UT1.

	UT1 A/B graded OBs [hours]		**UT1 C graded OBs [hours]**	
	A/B rank	C rank	A/B rank	C rank
P99	7.96	-12.39	1.80	5.90
P100	0.67	-4.06	-0.15	1.75
P101	5.42	-3.69	-1.34	3.53
P102	-0.07	-0.95	3.77	-4.09
P103	-4.25	0.97	-2.59	8.51
P104	3.07	0.00	-4.99	0.00
P105	-	-	-	-
P106	-6.93	2.20	-10.33	-0.94
P107	-1.07	0.00	-7.12	-4.58
P108	12.17	-0.70	-16.83	-1.00
P109	-0.40	-0.28	-9.79	-6.74
P110	0.96	2.09	-5.30	1.11
P111	-1.49	9.60	-6.96	0.36

Table C.2: Hours gained/lost of LightGBM versus PM for UT2.

	UT2 A/B graded OBs [hours]		**UT2 C graded OBs [hours]**	
	A/B rank	C rank	A/B rank	C rank
P99	7.10	21.90	-26.21	6.41
P100	0.78	-3.50	2.86	-0.56
P101	-0.31	-0.73	0.06	-5.46
P102	-1.93	0.35	0.22	2.09
P103	16.10	2.61	-20.32	7.09
P104	-3.10	6.77	-3.73	2.77
P105	-	-	-2.97	0.00
P106	1.00	9.20	-12.65	-0.72
P107	13.51	10.41	-23.85	-0.42
P108	7.83	2.80	-9.49	0.00
P109	-9.42	9.65	-4.75	0.43
P110	3.35	-3.49	-0.88	-0.26
P111	8.21	-3.96	-4.65	-0.77

Table C.3: Hours gained/lost of LightGBM versus PM for UT3.

	UT3 A/B graded OBs [hours]		**UT3 C graded OBs [hours]**	
	A/B rank	C rank	A/B rank	C rank
P99	4.84	3.71	-13.42	-0.07
P100	-6.06	3.42	-2.11	0.03
P101	-2.06	0.21	-2.75	-0.86
P102	1.31	-3.37	-2.77	-0.21
P103	-2.88	0.36	-23.44	1.07
P104	-7.82	0.19	8.21	-0.44
P105	-	-	-	-
P106	0.00	0.00	-7.26	0.00
P107	20.83	7.97	-38.96	1.40
P108	21.02	5.97	-21.13	-1.52
P109	9.69	12.56	-19.16	-4.19
P110	5.71	-2.97	-1.48	0.69
P111	11.96	-0.53	-14.21	3.23

Table C.4: Hours gained/lost of LightGBM versus PM for UT4.

	UT4 A/B graded OBs [hours]		**UT4 C graded OBs [hours]**	
	A/B rank	C rank	A/B rank	C rank
P99	11.73	-0.44	-8.17	0.56
P100	7.11	-2.40	-1.47	-0.53
P101	-1.88	3.57	0.87	2.17
P102	10.54	1.58	-7.21	-1.24
P103	9.06	8.61	-18.13	-4.17
P104	16.21	4.00	-18.83	1.28
P105	-	-	-	-
P106	-22.60	-1.30	-25.89	-3.02
P107	-1.68	1.29	-17.07	-2.75
P108	11.89	11.70	-22.70	0.45
P109	-7.64	17.13	-12.17	1.02
P110	55.85	-5.10	-7.75	-0.95
P111	-2.75	2.89	-4.81	1.98