Recovery of DESI BGS redshift measurements using Machine Learning

by

Sergio David Lobo Bolaño

Undergraduate Monograph

Submitted to the Department of Physics
in fulfillment of the requirements for
the degree of

Bachelor of Science in Physics

Advisor

Jaime Ernesto Forero-Romero

Department of Physics
UNIVERSIDAD DE LOS ANDES
COLOMBIA
November 28, 2018

Contents

List of Figures

List of Tables

Abstract

The DESI (Dark Energy Spectroscopic Instrument) project will conduct a five-year survey designed to cover 14.000 deg^2 by studying baryon acoustic oscillations (BAO) and redshift-space distortions (RSD). DESI needs simulations for its design, development, and operation, in particular, it uses simulations to evaluate the data pipelines and measurements of redshifts from the spectrometers. In particular, DESI uses a simulated survey of mock galaxies and their redshifts are compared with the -also simulated- redshift measurements of DESI. These redshift measurements, however, present differences with respect to the true redshifts values of the mock galaxies. It is necessary to correct these measurements so that the instrument can work properly when tested in the real world.

The object of this monograph was, therefore, to apply machine learning (ML) methods to this simulation data to correct the redshift measurements using observational variables as input.

After preprocessing the data, the variables selected as input for the ML models were the fluxes from the different observation bands, which are the $g, r, z, WISE1$, and $WISE2$ bands, as well as the redshift output of the instrument's pipeline and the true redshift from the mock galaxies as output. We trained and tested support vector regression (SVR) and kernel ridge regression (KRR) models using cross-validation and grid search, and found that an ensemble model of different KRRs is the best model with a coefficient of determination, $r^2 = 0.98$. This could be used to improve the accuracy of DESI when working in the five-year survey.

Chapter 1

Introduction

This chapter introduces the project, first, by establishing the objective of the monograph and after that, giving a brief introduction to the concepts needed throughout the document. We divide these concepts into two parts: DESI project and Machine Learning. First, we introduce the project and the main aspects of the instrument and survey, such as the target selection and simulations. After that, we introduce machine learning concepts, particularly related to supervised learning and kernel methods.

1.1 Objectives

1.1.1 General Objective

- Recover the true redshifts of Bright Galaxies using observational variables from the simulated DESI measurements, as an input to kernel methods of machine learning, to improve the accuracy of DESI when used in the real world

1.1.2 Specific Objective

- Characterize the dataset using a simple statistical and graphical procedures to select the set of meaningful features as input to the machine learning algorithms.
- Determine the set of computational parameters such as memory requirement, number of processors per node and size of dataset that performs the best on the cluster of the university restricted to constraints of resources, execution time and waiting time in the queue.
- Train and adjust the hyper-parameter of the models by using grid-search and cross-validation.
- Test and select the best model based on performance on unseen data and model simplicity.

1.2 DESI Project

The accelerated expansion of the universe is one of the most important phenomena in astrophysics, and in all modern physics, to such a point that it pushes forward the majority of the research done in observational cosmology [1]. DESI (Dark Energy Spectroscopic Instrument) will map one-third of the sky during a 5 years period[2]. DESI is a Stage IV ground-based dark energy experiment that will study baryon acoustic oscillations (BAO) and the growth of structure through redshift-space distortions (RSD) with a wide-area galaxy and quasar redshift survey. The survey will target four classes of extragalactic objects: Bright Galaxies (BGS), Luminous Red Galaxies (LRGs), Emission Line Galaxies (ELGs) and Quasi-stellar objects (QSOs) [2]. DESI is an instrument capable of taking up to 5000 simultaneous spectra in a range from 360 nm to 980 nm [3]. LRGs will be observed up to a $z = 1.0$. To explore the universe in a higher redshift, DESI will target ELG of up to $z = 1.7$. Quasars will be observed as direct signals from the underlying dark the matter distribution, and, to higher redshifts, $2.1 < z < 3.5$, using the ly-α forest absorption.

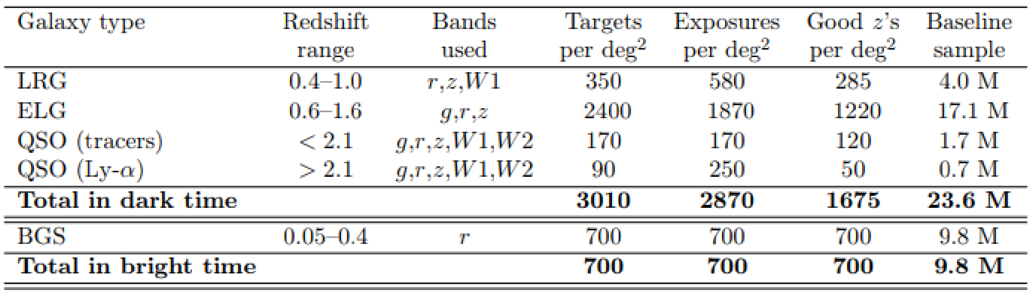
Before DESI stars to work, it needs simulations for its design, development, and testing. Some of this simulation focus on individual aspects, for example, hardware optimization, spectrographic analysis, fiber alignment, and target selection. In this project, we will use the result of the simulation of the redshifts of targets, and the simulation of the measurements of the instrument, to find the relationship between the expected measurement and actual measurement of the spectrograph.

1.2.1 Targets

The DESI survey will measure with high precision the baryon acoustic feature imprinted on the large-scale structure of the Universe, as well as the distortions of galaxy clustering due to redshift space effects. The survey will make spectroscopic observations of BGS, LRGs, ELG, and QSOs. To ensure highly efficient and spectroscopic completeness, the survey will select objects with spectral features expected to produce a reliable redshift determination or a Ly-α forest measurement within the DESI wavelength range [2]. Figure 1.1 shows a summary of the properties of each DESI target class.

Bright Galaxy Sample

The galaxy sample for the BGS will be a flux-limited, r-band selected sample of galaxies. The magnitude limit is determined by the total amount of observing bright time and the exposure times required to achieve our desired redshift efficiency. This target selection is, in essence, a deeper version of the galaxy target selection for the SDSS main galaxy sample (MGS). We explore the properties of the BGS target sample through mock catalogs created from numerical simulations. These mocks have



Galaxy type	Redshift range	Bands used	Targets per deg^2	Exposures per deg^2	Good z's per deg^2	Baseline sample
LRG	0.4–1.0	r,z,$W1$	350	580	285	4.0 M
ELG	0.6–1.6	g,r,z	2400	1870	1220	17.1 M
QSO (tracers)	< 2.1	g,r,z,$W1$,$W2$	170	170	120	1.7 M
QSO (Ly-α)	> 2.1	g,r,z,$W1$,$W2$	90	250	50	0.7 M
Total in dark time			**3010**	**2870**	**1675**	**23.6 M**
BGS	0.05–0.4	r	700	700	700	9.8 M
Total in bright time			**700**	**700**	**700**	**9.8 M**

Table 1.1: Summary of the properties of each DESI target class. The band listed are for the target selection, where g, r and z are optical photometry and $W1$ and $W2$ denote WISE infrared photometry [2].

identical properties to the MGS at low redshift, including the luminosity function, color distribution, and clustering properties. [2].

Luminous Red Galaxies

The lowest-redshift dark-time sample for DESI will come from targeting 350 candidate luminous red galaxies (LRGs) per square degree. These objects are both high in luminosity and red in rest-frame optical wavelengths due to their high stellar mass and lack of ongoing star formation. They exhibit strong clustering and a relatively high large-scale-structure bias, which enhances the amplitude of their power spectrum, and hence the BAO signal. Because of their strong 4000 $\AA$ breaks and their well-behaved red spectral energy distributions, low-redshift LRGs at $z < 0.6$ can be selected efficiently and their redshifts estimated based on SDSS-depth photometry. The BOSS survey has targeted 119 LRGs per deg^2 with $z \leq 0.6$ using SDSS imaging. DESI science analyses will incorporate existing BOSS spectroscopic samples (which cover 10,000 deg^2 of the DESI footprint) when available, as well as applying BOSS-like target selection algorithms (in regions not yet covered) to target LRGs at low z [2].

Emission Line Galaxies

Emission-line galaxies (ELGs) constitute the largest sample of objects that DESI will observe. The galaxies exhibit strong nebular emission lines originating in the ionized ("H II") regions surrounding short-lived but luminous, massive stars. ELGs are typically late-type spiral and irregular galaxies, although any galaxy actively forming new stars at a sufficiently high rate will qualify as an ELG. Because of their vigorous ongoing star formation, the integrated rest-frame colors of ELGs are dominated by massive stars, and hence will typically be bluer than galaxies with evolved stellar populations such as LRGs. The optical colors of ELGs at a given redshift will also span a larger range than LRGs due to the greater diversity of their star formation histories and dust properties

QSOs

The highest-redshift coverage of DESI will come from quasars (a.k.a. quasi-stellar objects, or QSOs), extremely luminous extragalactic sources associated with active galactic nuclei. QSOs are fueled by gravitational accretion onto supermassive black holes at the centers of these galaxies. The QSO emission can outshine that of the host galaxy by a large factor. Even in the nearest QSOs, the emitting regions are too small to be resolved, so QSOs will generally appear as point sources in images. These are the brightest population of astrophysical targets with a useful target density at redshifts $z > 1$ where the population peaks.

1.2.2 Simulations

Simulations are needed throughout the design, development, and operations of DESI. Some of these focus on individual aspects, e.g., to optimize a particular piece of the hardware design or to tune an individual algorithm. Other simulations provide mock data for the spectral extraction pipeline development prior to obtaining real data. End-to-end simulations will ensure system-level integration and scaling of the software, while also validating the design performance as a whole. Cosmological simulations are needed as input for particular pieces that require realistic cosmology to test their performance [3].

Cosmological Simulations

Cosmological simulations are critical inputs for the development, testing, and validation of the full DESI pipeline. In particular, they are essential to

- understand the effect of the tiling and fiber assignment strategies
- model the radial selection function of the different targets for the development of the large-scale structure catalog.
- provide a realistically clustered mock input catalog for algorithm development
- serve as input for the end-to-end simulation pipeline for system level testing

The cosmological simulations as needed by DESI are constructed with a multi-step procedure. First, given a cosmological model, an N-body simulation is run using either baryons plus dark matter or solely dark matter. Second, galaxies and quasars are populated within this N-body simulation: Gas-hydrodynamical simulations need a way of identifying the objects, while solely dark matter simulations require a scheme such as a statistical halo occupation distribution (HOD) model or a physical semi-analytic galaxy formation model [3].

Instrument Simulations

Instrument simulations guide the hardware design process and provide inputs for the spectral pipeline development. This is optimized both at the high-level to ensure that DESI meets the science requirements, and at the low-level for where the spectral extraction pipeline places requirements on the hardware design [3]. Figure

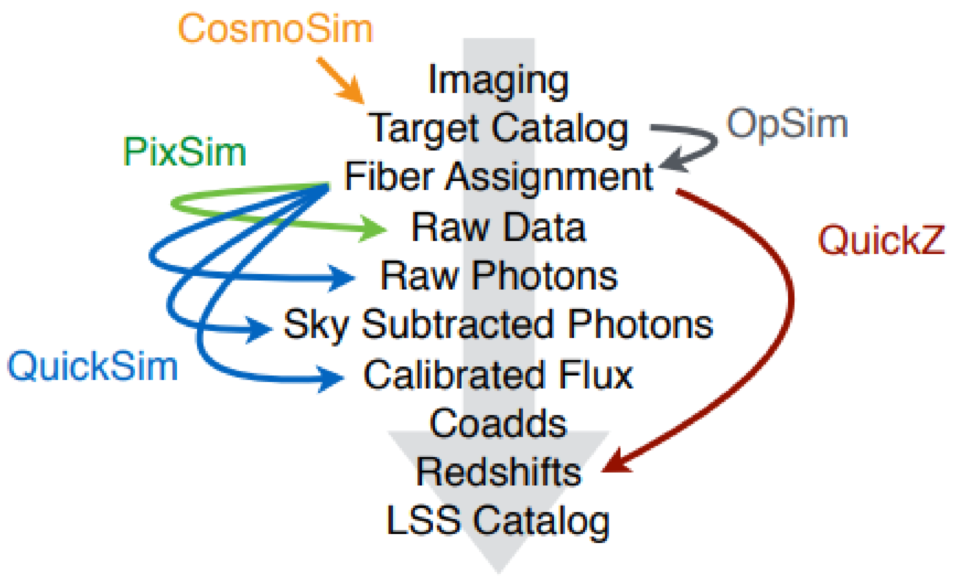


Figure 1.1: Diagram of different types of simulations for testing and improving DESI [3]

1.1 shows a diagram of the different simulation used to test and optimize DESI. This simulations are:

- CosmoSIM: input cosmology simulations to produce mock target catalogs.
- PixSim: pixel-level simulations of raw data to be used for spectroscopic pipeline development and study of detailed systematics.
- QuickSim: a fast spectral simulator that emulates the DESI spectroscopic resolution and throughput. It propagates signal, sky, and detector noise but bypasses the computationally expensive raw data pixel simulations and extractions.
- OpSim: a survey simulator that traces what tiles would be observed in what order based upon the real next field selector and a Monte Carlo realization of weather and observing conditions.
- QuickZ: an ultra-fast simulator that applies DESI efficiencies to an input target catalog to generate an output redshift catalog.

1.3 Machine Learning

Machine learning plays a key role in many areas of science, finance, and industry, some examples may be the prediction of medical conditions and diseases, prediction of price of a stock on the basis of company performance and economic data or estimating the amount of glucose in the blood of a diabetic person from the infrared absorption spectrum of the blood.

In general, the problem of machine learning consist typically of having a set of measurements, that can be either quantitative (the measurement of an instrument or a stock price) or categorical (such as a class, or category) that we want to predict based on a set of **features**, for example, the performance metrics of a company. For doing this, we have a **training** set of data, in which we know the relation between the outcome and the features, from this, we build a prediction model, or **learner**, and then use the model to predict the outcomes of features in a **test** set, to see if the model is able to **generalize** what it *learned* from the training set. A good learner is one that accurately predicts such an outcome [4]. This particular way of learning is called learning from example, or ***supervised learning***. There are other forms of learning, such as unsupervised learning and reinforcement learning, this two forms of learning will not be discussed in this monograph.

More formally, we can describe the general model of learning from example through three components[5]:

I A generator (G) of random vectors $x \in R^n$, drawn independently from a fixed but unknown probability distribution function F(x)

II A supervisor (S) who returns an output value y to every input vector x according to a conditional distribution function $F(y|x)$, also fixed but unknown. This is the general case, which includes the case where the supervisor uses a function $y = f(x)$.

III A learning machine (LM) capable of implementing a set of functions $f(x, \alpha), \alpha \in \Lambda$, where Λ is a set of parameters.

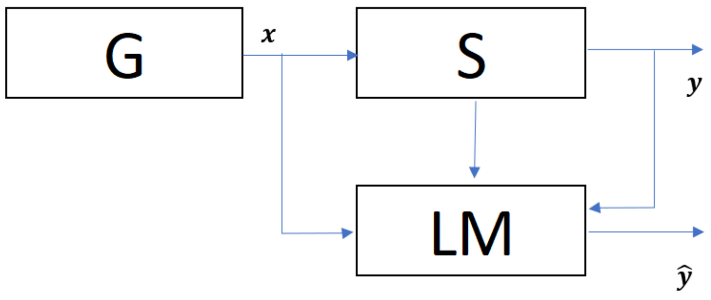


Figure 1.2: A model of learning from examples [5].

The description of the process is shown in figure 1.2. In order to choose the best available approximation to the supervisor´s response, one measures the *loss* or discrepancy between the response provided by the *LM*, $f(x,\alpha)$ and the supervisor's response (y), $L(y, f(x,\alpha))$. Consider the expected value of the loss, give by the *Risk Functional*:

$$R(\alpha) = \int L(y, f(x,\alpha))dF(x,y). \tag{1.1}$$

The goal is to find the function $f(x,\alpha_0)$ that minimizes the risk functional $R(\alpha)$, in the situation where the joint probability distribution function $F(x,y)$ is unknown and the only available information is contained in the **training set** [5]. The problem of supervised learning can be divided in three classes: the problem of patter recognition, regression estimation, and density estimation.

Patter recognition

In this problem the supervisor's response is categorical of two values $y = 0, 1$. For example, in a classification problem, whether a picture contains a dog or not. The loss function in this case is

$$L(y, f(x,\alpha)) = \begin{cases} 0, & \text{if } y = f(x,\alpha) \\ 1, & \text{if } y \neq f(x,\alpha) \end{cases} \tag{1.2}$$

Regression Estimation

Let the supervisor´s answer y be a real value, and let $f(x,\alpha), \alpha \in \Lambda$ be a set of real functions that contains the **regression function**

$$f(x,\alpha) = \int y dF(y|x). \tag{1.3}$$

The loss function for this problem is

$$L(y, f(x,\alpha)) = (y - f(x,\alpha))^2. \tag{1.4}$$

Density Estimation

Finally, consider the problem of density estimation from a set of densities $p(x,\alpha), \alpha \in \Lambda$. For this problem we consider the following loss function

$$L(p(x,\alpha)) = -\log p(x,\alpha). \tag{1.5}$$

In this monograph, we will make use of the regression problem, since we want to find the correct redshift from a series of observational features. For requirements of the project, the algorithms used have to be simple and keep the number of parameters low. The main algorithms for regression in machine learning include kernel method and artificial neural networks (NN). In recent years NNs have been used to solve lots of machine learning problems, however, they require an incredibly large number of parameters (as in deep learning where the number of parameters can reach the millions). Therefore, the kernel methods will be explored while the NNs will be left out.

1.3.1 Kernel Methods

The theory and algorithms of machine learning are very well developed for the case of linear models. However, real-world data often requires nonlinear methods to detect the kind of patterns and dependencies that allow successful predictions. This is where the kernel is used, by using a positive definite kernel, we can have the best of both worlds. The kernel corresponds to a dot product in a feature space, in this space, the estimation method is linear[6].

Suppose we are given empirical data

$$(x_1, y_1), ..., (x_n, y_n) \in \mathcal{X} \times \mathcal{Y}. \tag{1.6}$$

Here, the domain $\mathcal{X}$ is the set of *inputs*, where the predictor variables, x_i, are taken from; the $y_i \in \mathcal{Y}$ are called targets. Apart from being a set, $\mathcal{X}$, most have a structure that allows us to measure the similarity between input points so that the method can *generalize* to unseen data points. In the case of binary classification, for example, given a new input $x \in \mathcal{X}$, we want to predict the corresponding $y \in \pm 1$. Generalization means we want to predict a y such as the pair (x, y) is in some sense *similar* to the training examples. To this, we need similarity measures in $\mathcal{X}$ and ± 1. The letter is easier since in the target space two values can only be identical or different. For the input space, we need a 'similarity' function

$$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}, \quad (x, x') \to k(x, x') \tag{1.7}$$

satisfying, for all $x, x' \in \mathcal{X}$,

$$k(x, x') = \langle \Phi(x), \Phi(x') \rangle, \tag{1.8}$$

where Φ maps into some dot product space $\mathcal{H}$, sometimes called the *feature space*. The similarity measure k is called a *kernel*, and Φ is called its *feature map* [6]. The advantage of using such a kernel as a similarity measure is that it allows us to construct algorithms in dot product spaces. The *kernel function* can be in practical uses one of the following:

$$\text{linear} = \langle x, x' \rangle. \tag{1.9}$$

$$\text{polynomial} = (\gamma \langle x, x' \rangle + r)^d. \tag{1.10}$$

$$\text{rbf} = \exp(-\gamma ||x - x'||^2). \tag{1.11}$$

$$\text{sigmoid} = \tanh(\gamma \langle x, x' \rangle + r). \tag{1.12}$$

$$\tag{1.13}$$

Where γ, d, and r are hyper-parameters that must be tuned by hand and $\langle x, x' \rangle$ represents a dot product in the corresponding space.

Support Vector Machines and Support Vector Regression

A support vector machine constructs a hyperplane or set of hyperplanes in a high or infinite dimensional space, which can be used for classification, regression or other tasks. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training data points of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier [7]. For the case of a regressor, the basic idea is the following: in $\epsilon - SV$ regression [5], our goal is to find a function $f(x)$ that has at most ϵ deviation from the actually obtained targets y_i for all training data, and at the same time is as flat as possible. In other words, we do not care about errors as long as they are less than ϵ, but will not accept any deviation larger than this. For the simpler case, we begin by describing the case of a linear function as an example, taking the form

$$f(x) = \langle w, x \rangle + b. \tag{1.14}$$

In this case, *flatness* means that one seeks for small w. One way to ensure this is to write the optimization problem as follows:

$$\text{minimize} \quad \frac{1}{2}||w||^2 \tag{1.15}$$

$$\text{subject to} \quad \begin{cases} y_i - \langle w, x \rangle - b & \leq \epsilon \\ \langle w, x \rangle + b - y_i & \leq \epsilon \end{cases} \tag{1.16}$$

This assumes however, that such a function f exists, but it may not be the case where all the data points can be fit into a tube of size 2ϵ, therefore, one can introduce slack variables ζ and ζ^* to cope with the otherwise infeasible constraints of the previous optimization problem. Hence we arrive at the following formulation:

$$\text{minimize} \quad \frac{1}{2}||w||^2 + C\sum_{i=1}^{l}(\zeta_i + \zeta_i^*) \tag{1.17}$$

$$\text{subject to} \quad \begin{cases} y_i - \langle w, x \rangle - b & \leq \epsilon + \zeta_i \\ \langle w, x \rangle + b - y_i & \leq \epsilon + \zeta_i^* \\ \zeta_i, \zeta_i^* & \geq 0 \end{cases} \tag{1.18}$$

The constant C determines the trade-off between the flatness of f and the amount up to which deviations larger than ϵ are tolerated. Figure 1.3 shows the schematics of a linear SVM used for regression, the grey zone represents were the loss function does not account any penalty and the ζ shows the tolerance to deviations.

For non-linear functions, the data can be mapped into a high dimensional space, called kernel space. Therefore, replacing all instances of x in the previous equations with $k(x_i, x_j)$ yields the formulation of the optimization problem. This is a complete subject of mathematics and we can not dig further, but generally speaking, the optimization problem is solved using Lagrange multipliers and the Karush-Kuhn-Tucker conditions. The objective of the procedure is to find the weights w. Using

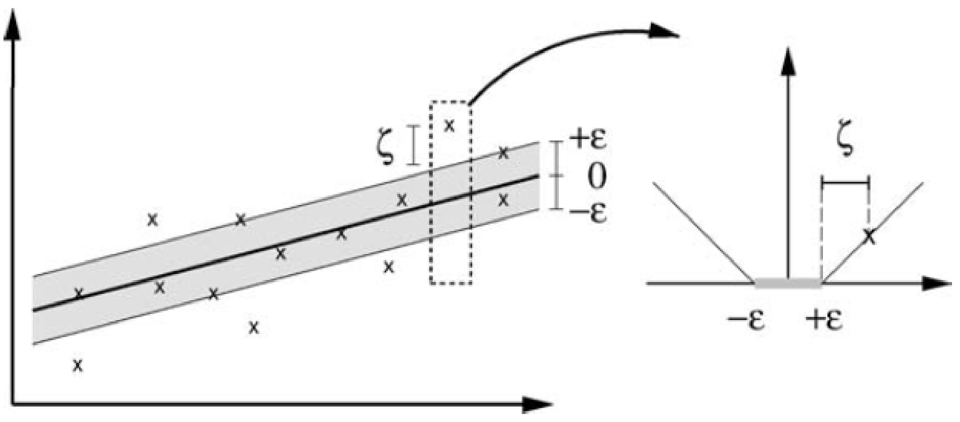


Figure 1.3: The soft margin loss setting for a linear SVM [6]

this two mathematical tools results in that the w can be written explicitly as a linear combination of the so-called **support vectors**, which are training samples that lie outside the boundary of the tube. In a sense, the complexity of a function's representation by SVs is independent of the dimensionality of the input space and depends only on the number of support vectors[6]. The decision boundary function is:

$$\sum_{i=1}^{n}(\alpha_i - \alpha_i^*)k(x_i, x) + b, \tag{1.19}$$

These parameters correspond to the dual coefficients of the optimization problem (also called Lagrange multipliers), α, α_i^*, the support vectors x_i and the independent term, the intercept b. Note that this is a hyper-plane on the feature space. Therefore, the solution in classification or regression is contained in this set of parameters only.

Kernel Ridge Regression

Kernel ridge regression combines Ridge Regression with the kernel trick. It thus learns a linear function in the space induced by the respective kernel and the data. For non-linear kernels, this corresponds to a non-linear function in the original space [8]. The form of the model learned by this method is identical to support vector regression but with different loss functions. In ridge regression, the regression coefficients are shrunk by imposing a penalty on their size. The ridge coefficients minimize a penalized residual sum of squares:

$$w = \arg\min_{w}\left[\sum_{i=1}^{N}(y_i - b - \langle w, x\rangle) + \lambda\sum_{j=1}^{p} w_i^2\right]. \tag{1.20}$$

Here $\lambda \geq 0$ is a complexity parameter that controls the amount of shrinkage: the larger the value of λ, the grater the amount of shrinkage. As the ϵ-insensitive loss functions of SVRs, the ridge solutions are not equivariant under scaling of the inputs, and so one normally standardizes the inputs before solving.

These methods will be used to tune the redshift prediction of the DESI instrument. To do this, it is therefore necessary to properly select the features, scale the data and tune the hyper-parameters of the method. We will explain how to do this in the following chapter.

Chapter 2

Methodology

2.1 Introduction

This chapter explains the procedure to achieve the objectives declared in the previous chapter. Each objective consists of a series of activities that produce an outcome, each one contributing to reaching the general objective of this monograph. First, we show a description of the data pre-processing, then the selection of computer parameters and finally, a description of the methodology for training and testing the models. Figure 2.1 shows the general methodology for the project, each step will be treated below.

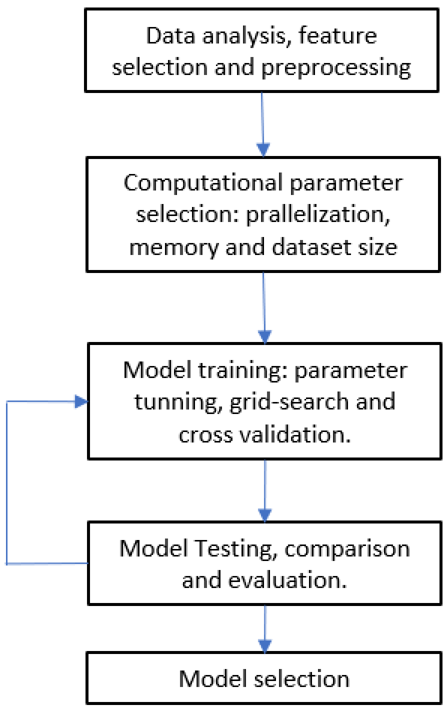

Data analysis, feature selection and preprocessing
Computational parameter selection: prallelization, memory and dataset size
Model training: parameter tunning, grid-search and cross validation.
Model Testing, comparison and evaluation.
Model selection

Figure 2.1: General methodology for the project.

2.2 Data analysis and pre-processing

The dataset consists of two files, one with the redshift measurement of DESI (TAR file) and the other with simulated true redshifts (TRUTH file). Apart from this, the files also contain several variables related to the parameters of the simulations, the class of a target, the observation conditions, etc. Since all of these variables represented as columns in the data files may not be important, we have to visualize and select which are important.

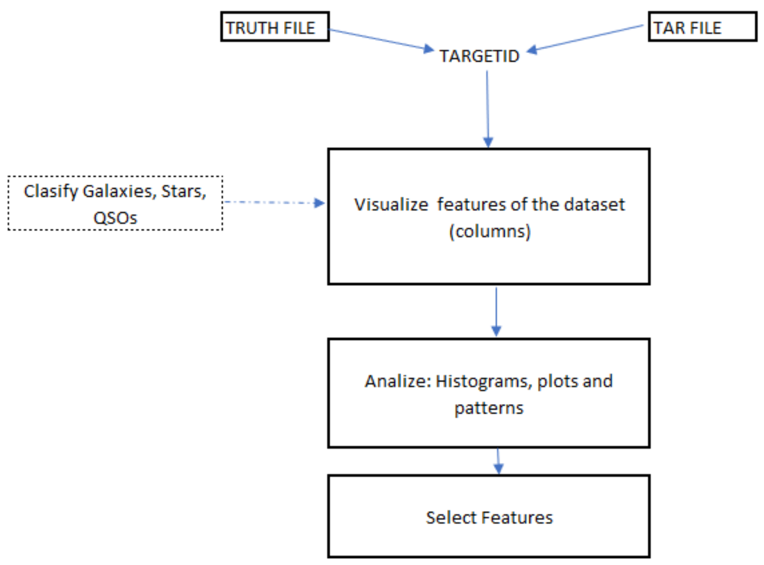

TRUTH FILE
TAR FILE
TARGETID
Clasify Galaxies, Stars, QSOs
Visualize features of the dataset (columns)
Analize: Histograms, plots and patterns
Select Features

Figure 2.2: General methodology to understand the dataset and extract valuable features.

Figure 2.2 shows the step-by-step method to understand the data. In the first step, the two files have to be linked by the variable TARGETID, this variable represents the identity of a target through the whole pipeline of the experiments (observation and measurements).

Next, we discriminate by target class (Galaxy, QSOs, etc...) and observe the behavior of the variables in the files with respect to redshift, for example, fluxes, magnitudes, etc ... From this, one can neglect some basic variables that contribute no relevant information to the problem. Then we execute a second analysis with more detail to see if the variables hold any relation with the redshift measurement by using histograms and approximating distributions. This is done to select which variables (or features) can be used to explain the differences between the true redshift of a galaxy and the redshift measured by DESI. The details of this can be found in Chapter 3.

After the features are selected, we found that treating the variables in logarithmic scale is visually more representative and also produced good results in the machine learning algorithms. Since the kernel methods used are non-invariant to scaling, we also scale the data so that the means are equal to zero and the variance to one of all the variables.

2.3 High-performance computing analysis

The training time of the model depends on the size of the training set, however, since the algorithms are going to be executed in parallel in the computing cluster of the University, the training time will also depend on the hardware. For example, the memory of the machine and the number of processors in which the code will be parallelized, as well as the number of processors that the computer (or node) actually has. To see what is the best combination of parameters to train the models in a reasonable amount of time up to a reasonable accuracy, it will be necessary to restrict also the size of the dataset used for training.

Figure 2.3 shows the step to accomplish this. First, we select the model and architecture (grid search/cross-validation, explained next) and the dataset size, in order to see how the training time scales with size, we select different sizes, in this case, 100, 1000, 5000, and 10000 points. Then, the algorithm has to be run in a certain number of processors, and with certain RAM requested to the cluster, in this case, we try 1, 4, 8 and 16 processors, and 16, 32, 64 and 128 GB of memory. The expected result is the set of parameters to run the complete training algorithm on a big dataset.

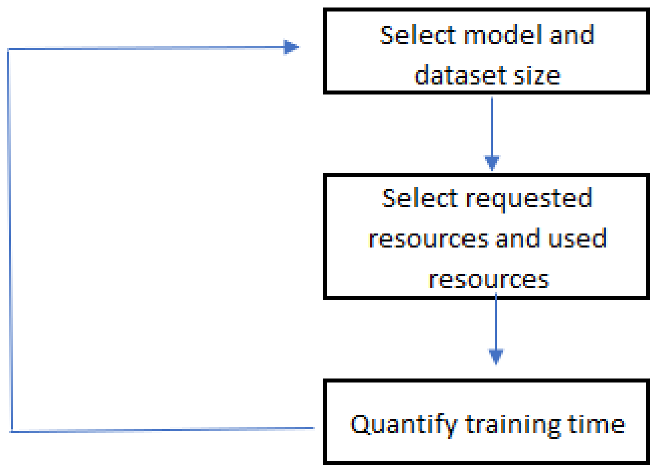

Select model and dataset size
Select requested resources and used resources
Quantify training time

Figure 2.3: Sequence of steps to gather enough information to understend the relation between computation time, memory, dataset size and processors.

2.4 Machine Learning

Once we understand the scalability of the training time with respect to hardware variables, it is time to train and test the models. For this the dataset will be divided in a training set, corresponding to 75% of the dataset and a test set (the rest 25 %) as shown in Figure 2.4. After this, according to the results of the previous section, we will split the training set into subsets of a specific size, these subsets will again be divided into a development and evaluation sets, to apply to them the grid search and cross-validation. We use cross-validation and grid search to select the hyper-parameters of the models and to quantify the error of the algorithms.

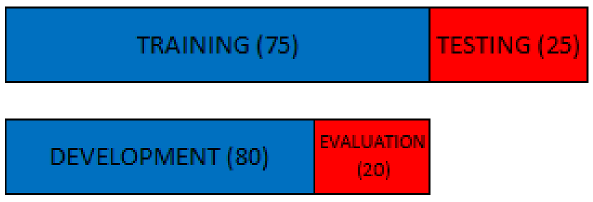


Figure 2.4: Divition of the dataset in train and test. The train set is divided in development-evaluation subsets.

2.4.1 Training

For training, we use grid search and cross validation to select the hyper-parameters of the models. Grid search executes an exhaustive search over specified parameters values for an estimator. The parameters are specified on a grid as follow:

```
param_grid = [
{'C': [1, 10, 100, 1000], 'kernel': ['linear']},
{'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001], 'kernel': ['rbf']},
]
```

this specifies that two grids should be explored: one with a linear kernel and C values in [1, 10, 100, 1000], and the second one with an RBF kernel, and the cross-product of C values ranging in [1, 10, 100, 1000] and gamma values in [0.001, 0.0001].

Learning the parameters of a prediction function and testing it on the same data is a methodological mistake: a model that would just repeat the labels of the samples that it has just seen would have a perfect score but would fail to predict anything useful on yet-unseen data. To avoid this, the grid search method is used alongside with **cross validation** (CV). In the *k-fold* CV approach, the training set is divided into k smaller sets and the following procedure is followed for each of the k 'folds':

- A model is trained using $k - 1$ of the folds as training data (development, see Figure 2.4).
- the resulting model is validated on the remaining part of the data (validation, see Figure 2.4)

The performance measure reported by k-fold cross-validation is then the average of the values computed in the loop. This approach can be computationally expensive.

2.4.2 Testing

Once we have trained the models, they will be tested on the unseen 25% of the data to check its generalization ability. This means that we need to see whether the model truly 'learns' to predict redshifts or just memorized the training dataset. To select the best performing algorithm on the test dataset, the criterion will be the r^2 measure and the simplicity of the model.

2.4.3 Model Performance

The performance metric used to evaluate a model is the coefficient of determination, r^2 score. It provides a measure of how well future samples are likely to be predicted by the model, or the proportion of the variance in the dependent variable that is predictable from the independent variables. The best possible score is 1.0 and it can be negative (because the model can be arbitrarily worse). A constant model that always predicts the expected value of y, disregarding the input features, would get an r^2 score of 0.0 [8].

If $\hat{y_i}$ is the predicted value of the i-th sample and y_i is the corresponding true value, then the score r^2 estimated over $n_{samples}$ is defined as

$$r^2(y,\hat{y}) = 1 - \frac{\sum_{i=0}^{n_{samples}-1}(y_i - \hat{y_i})^2}{\sum_{i=0}^{n_{samples}-1}(y_i - \bar{y_i})^2}, \tag{2.1}$$

where $\bar{y} = \frac{1}{n_{samples}}\sum_{i=0}^{n_{samples}-1} y_i$.

In general, the methodology for this project consists of analyzing the complete dataset and selecting the valuable features that can be used to predict (or recover) a correct redshift, finding the scalability of the training time with respect to the size of the dataset and the cluster resources, and finally, training an evaluating the machine learning models.

Chapter 3

Data Analysis and Pre-processing

3.1 Data descripcion

The dataset consists of two FITS files corresponding to the simulated observations of the DESI instrument and the ”truth” data from cosmological simulations. Each data file has a key column called TARGETID, this variables is the same across all simulations and identifies a particular object in the sky, thus it is needed to relate both datasets.

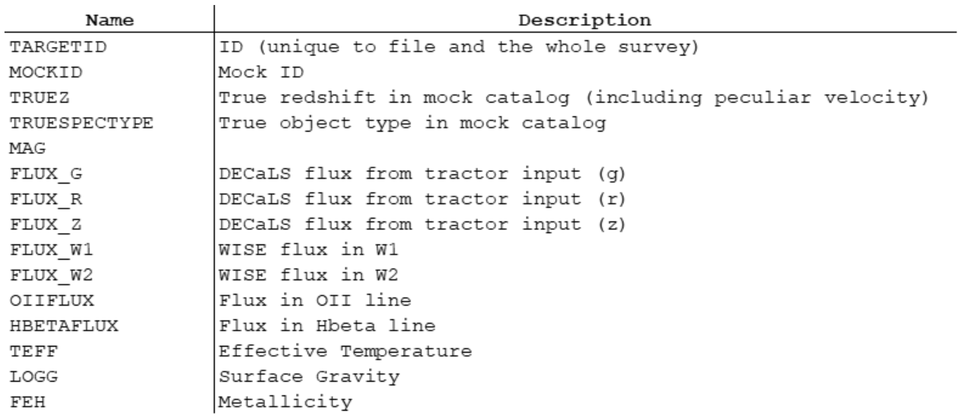
3.1.1 Simulated expected results - truth file

This file contains the data from the cosmological simulation of the target objects. Thus, this file contains the expected redshift that the instrument *should* measure. The complete list of columns in the file is shown in table 3.1[1]. Since we aim to correct the measurements of redshifts given by the instrument, the variable TRUEZ will be our target variable or ”*output*” to the machine learning models. The rest of the information will be used mostly for understanding the dataset but no as part of the ML model since this information would not be available to the instrument in reality. This dataset contains the redshift of 24.851.543 objects.

3.1.2 Simulated Observations - target file

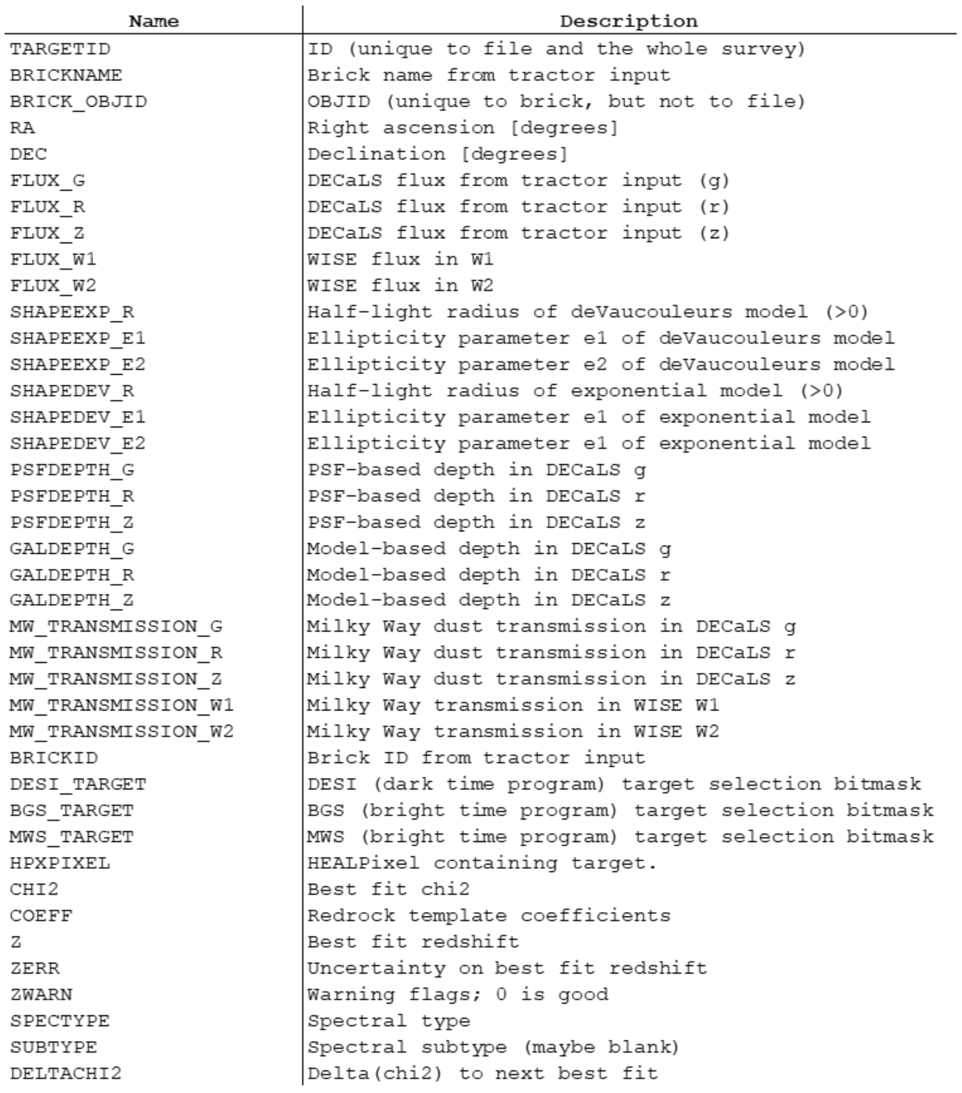
This file contains the redshift values of the targets as measured by the instrument in simulation. Apart from this, it also contains the columns shown in table 3.2. The different characteristics listed in Table 3.2 are the ones that we will use as input in our machine learning models, since is the data available from the instrument. This dataset contains the redshift of 2.131.896 objects. Since the files don´t have the

[1]Additional information on `https://desidatamodel.readthedocs.io/en/stable/`



Name	Description
TARGETID	ID (unique to file and the whole survey)
MOCKID	Mock ID
TRUEZ	True redshift in mock catalog (including peculiar velocity)
TRUESPECTYPE	True object type in mock catalog
MAG	
FLUX_G	DECaLS flux from tractor input (g)
FLUX_R	DECaLS flux from tractor input (r)
FLUX_Z	DECaLS flux from tractor input (z)
FLUX_W1	WISE flux in W1
FLUX_W2	WISE flux in W2
OIIFLUX	Flux in OII line
HBETAFLUX	Flux in Hbeta line
TEFF	Effective Temperature
LOGG	Surface Gravity
FEH	Metallicity

Table 3.1: Columns in the cosmological simulation data file



Name	Description
TARGETID	ID (unique to file and the whole survey)
BRICKNAME	Brick name from tractor input
BRICK_OBJID	OBJID (unique to brick, but not to file)
RA	Right ascension [degrees]
DEC	Declination [degrees]
FLUX_G	DECaLS flux from tractor input (g)
FLUX_R	DECaLS flux from tractor input (r)
FLUX_Z	DECaLS flux from tractor input (z)
FLUX_W1	WISE flux in W1
FLUX_W2	WISE flux in W2
SHAPEEXP_R	Half-light radius of deVaucouleurs model (>0)
SHAPEEXP_E1	Ellipticity parameter e1 of deVaucouleurs model
SHAPEEXP_E2	Ellipticity parameter e2 of deVaucouleurs model
SHAPEDEV_R	Half-light radius of exponential model (>0)
SHAPEDEV_E1	Ellipticity parameter e1 of exponential model
SHAPEDEV_E2	Ellipticity parameter e1 of exponential model
PSFDEPTH_G	PSF-based depth in DECaLS g
PSFDEPTH_R	PSF-based depth in DECaLS r
PSFDEPTH_Z	PSF-based depth in DECaLS z
GALDEPTH_G	Model-based depth in DECaLS g
GALDEPTH_R	Model-based depth in DECaLS r
GALDEPTH_Z	Model-based depth in DECaLS z
MW_TRANSMISSION_G	Milky Way dust transmission in DECaLS g
MW_TRANSMISSION_R	Milky Way dust transmission in DECaLS r
MW_TRANSMISSION_Z	Milky Way dust transmission in DECaLS z
MW_TRANSMISSION_W1	Milky Way transmission in WISE W1
MW_TRANSMISSION_W2	Milky Way transmission in WISE W2
BRICKID	Brick ID from tractor input
DESI_TARGET	DESI (dark time program) target selection bitmask
BGS_TARGET	BGS (bright time program) target selection bitmask
MWS_TARGET	MWS (bright time program) target selection bitmask
HPXPIXEL	HEALPixel containing target.
CHI2	Best fit chi2
COEFF	Redrock template coefficients
Z	Best fit redshift
ZERR	Uncertainty on best fit redshift
ZWARN	Warning flags; 0 is good
SPECTYPE	Spectral type
SUBTYPE	Spectral subtype (maybe blank)
DELTACHI2	Delta(chi2) to next best fit

Table 3.2: Columns in the Simulated Observations data file

same amount of points, they were cut to the one with the least (the target file) by linking the rows by its TARGETID.

3.2 Overview of the dataset

3.2.1 Redshift relations

To understand the dataset, the first thing we need to do is see how the Z and TRUEZ variables behave. In Figure 3.1 we see three distinct regions: a 45-degree line that corresponds to the redshifts measurements of DESI that are very close to the expected 'real' value, a square region where the data seems to scatter randomly except for some line groupings, and a third region of horizontal line near True Z = 0. The task is, therefore, to dissolve the square region and have all the points along the diagonal line.

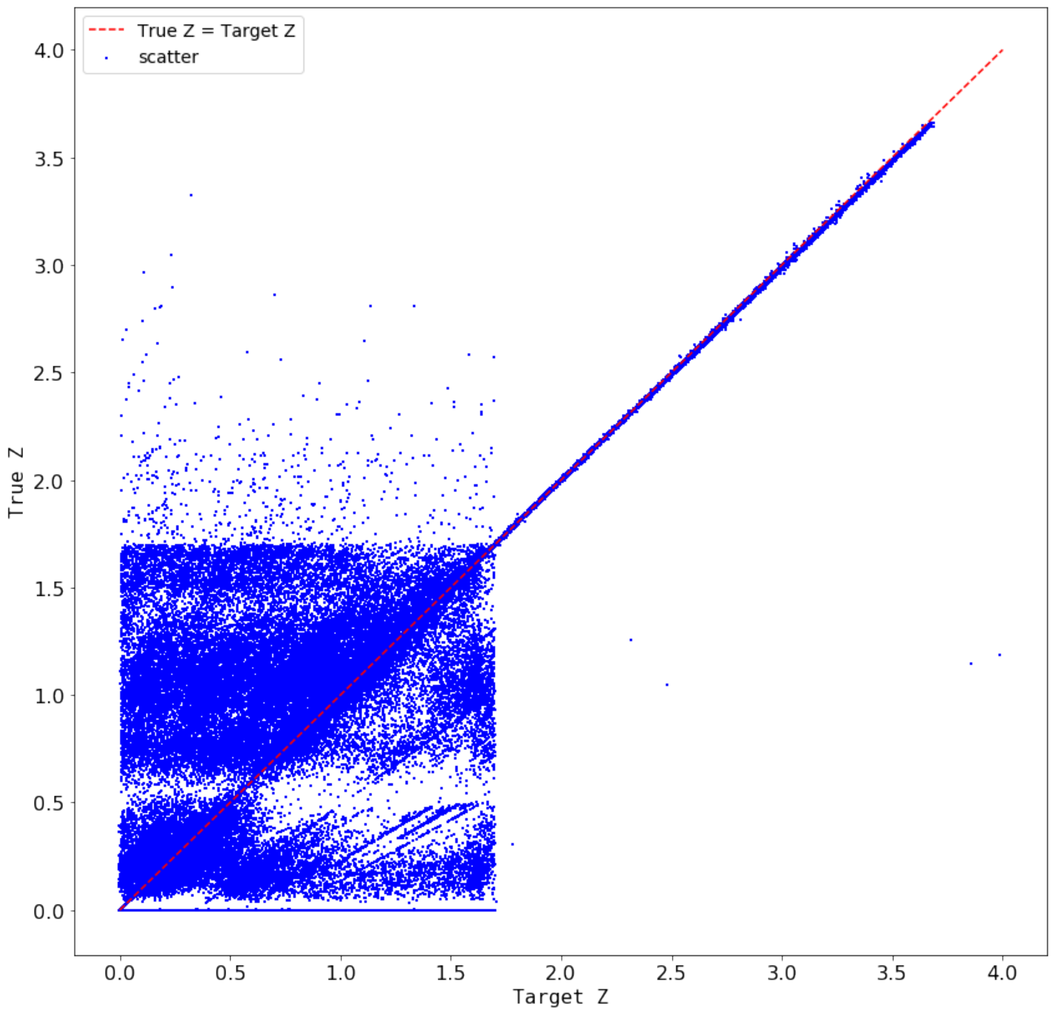
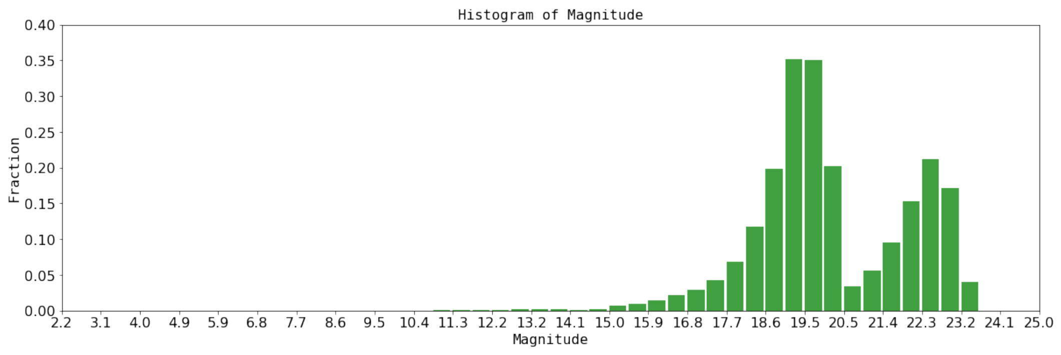
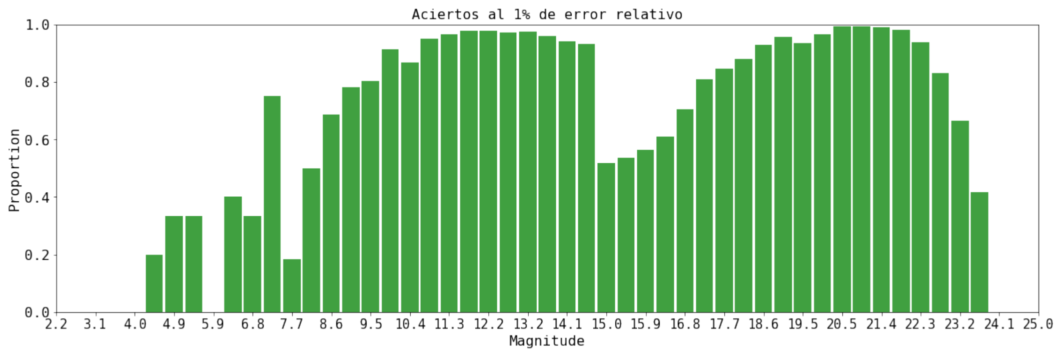


Figure 3.1: Relation between the 'observed' readshift - target Z, and the 'generated' redshift - True Z.

Now, cutting along the MAG (magnitude) variable, it is possible to see the distribution of magnitudes of the gathered data an see if there is any relation with



(a)



(b)

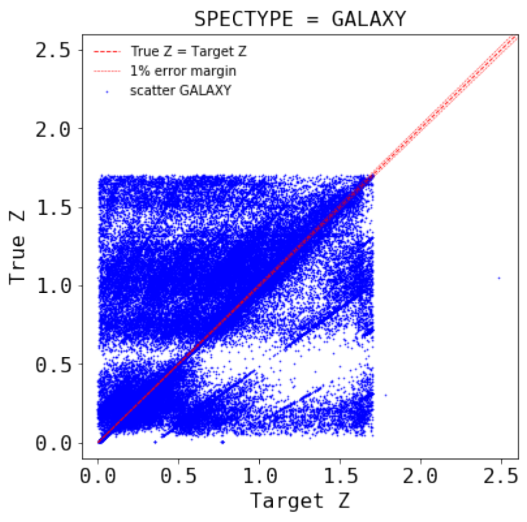
Figure 3.2: (a) Distribution of magnitudes in the dataset. (b) Proportion of the number of measured redshifts that are within the 1% error of their corresponding TRUEZ

the square region. Figure 3.2a shows this distribution and Figure 3.2b shows the fraction redshifts of each bin that is within a 1% error of the true value. We can see that the majority of the redshift is near its true value, however, some valleys indicate that some regions (for example between 15 and 17 in magnitude) are not that good. For simplicity, from now on we will refer to the **simulated expected redshift** as TRUEZ (the output to the ML models) and the **simulated observations of the targets** as Z (the input to the models) or TARZ.

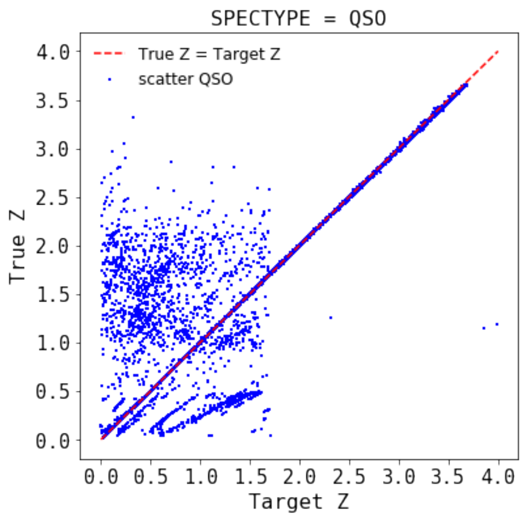
3.2.2 Spectral types

From figure 3.2 it is possible to infer that the different magnitudes of the targets can be related to the different regions on figure 3.1, and this difference in magnitude is also related to the SPECTYPE of each target. The spectral type of the data in the tar-file is distributed as shown in Table 3.3. Near 85% of the dataset is composed of Galaxy-type objects, for this reason, this is the most interesting class.

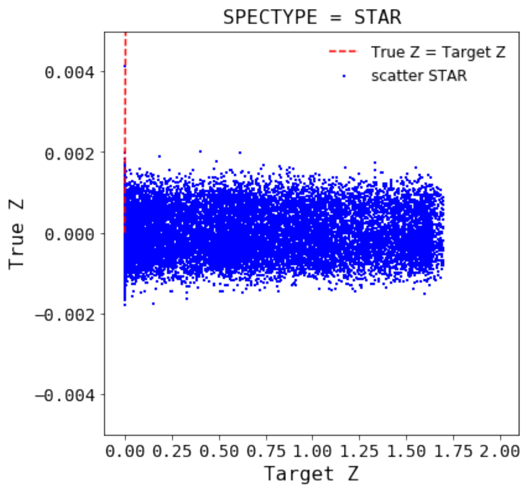
The redshift relations similar to Figure 3.1 discriminated by SPECTYPE are shown in Figure 3.3, where the three regions in Figure 3.1 seems to be related to each spectral type. The GALAXY objects are distributed along a square and there are lines formed at angles different than 45, which means that there is a relation between the two variable but 'out of calibration'. It is worth mentioning that Figure



(a)



(b)



(c)

Figure 3.3: Redshift relation for (a) Galaxy-type objects, (b) QSO-type objects, and (c) Star-type objects

SPECTYPE	N samples	% of dataset
Galaxy	1796213	84.25
QSO	194319	9.11
Star	141364	6.63
Total	2131896	100

Table 3.3: Spectral type distribution of the tar file.

3.3a can be deceiving because 96.34% of the Galaxy-type data is within the 1% margin line in the figure, which means that the square is formed only by the 3.66 % of the galaxy data, corresponding to approximately 65.742 measurements, still a lot.

The QSO-type objects in Figure 3.3b correspond to the 45-degree line in Figure 3.1 since the majority of points are along this line, although the same line pattern and dispersion of the galaxy-type objects are present, in the least quantity. However, the Star-type in Figure 3.3c is randomly scattered over the TARZ range and correspond to the horizontal line in Figure 3.1. Once again, the most interesting SPECTYPE is GALAXY, because it has the majority of error in TARZ, and also presents different patterns, QSOs are already fine and are not a priority while STAR is completely random a represent a small fraction of the whole dataset. From now on we will focus on Galaxy-type objects only.

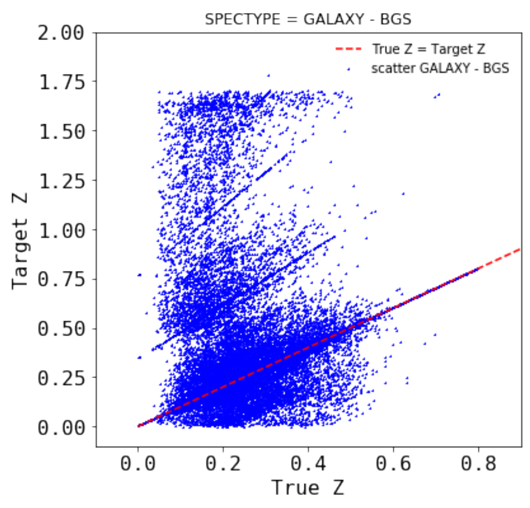
3.2.3 Galaxy-type objects

Galaxy-type objects are classified as Bright Galaxy Survey (BGS), Emission Line Galaxies (ELG) and Luminous Red Galaxies (LRG) distributing according to Table 3.4. In this case, the sub-types are more evenly distributed. The redshift relations of each sub-type is shown in Figure 3.4. BGS and ELG follow a similar pattern, however, ELG is more disperse while BGS is clustered along the TARZ = TRUEZ line, whereas LRG sub-type is already perfect.

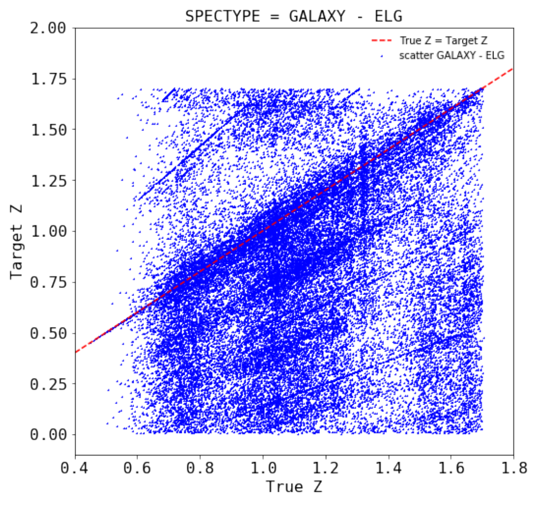
Galaxy subtype	N samples	% of dataset
BGS	889336	49.51
ELG	601847	33.51
LRG	305030	16.98
Total	1796213	100

Table 3.4: Distribution of Galaxy sub-types

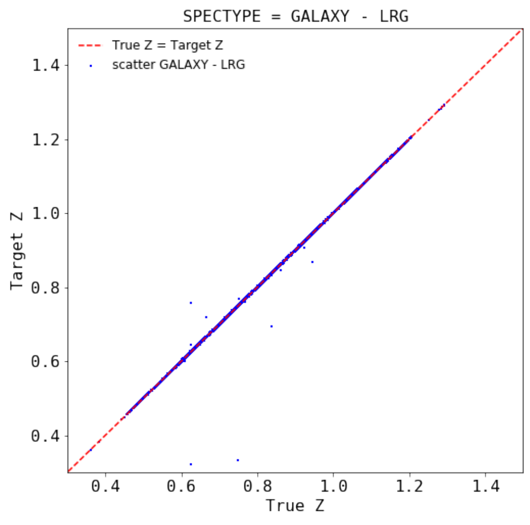
Note that the line structures are more visible in Figure 3.4a than in Figure 3.4b, therefore, to extract the relevant features in the tar dataset that may be related to this structure, we will use the BGS subset and then see if the feature extracted are also useful for the ELG subset. Therefore, from now on we will use only the BGS data subset and proceed to find the relevant features (in Table 3.2)



(a)



(b)



(c)

Figure 3.4: Redshift relation for Galaxy (a) BGS-type objects, (b) ELG-type objects, and (c) LRG-type objects

3.3 Relevant Features

The ability of the instrument for measuring the correct redshift will probably depend on the quality of the fluxes that the fiber optics receive. The fluxes of the objects are related to its magnitude, as we saw in Figure 3.2, the magnitude is related to the correct prediction of the instrument's Z, but since the information of magnitude is known to the instrument in form of fluxes, this variables will be explored next. The six fluxes variables are named in Table 3.2. The distributions of the fluxes in the BGS dataset are shown in Figure 3.5.

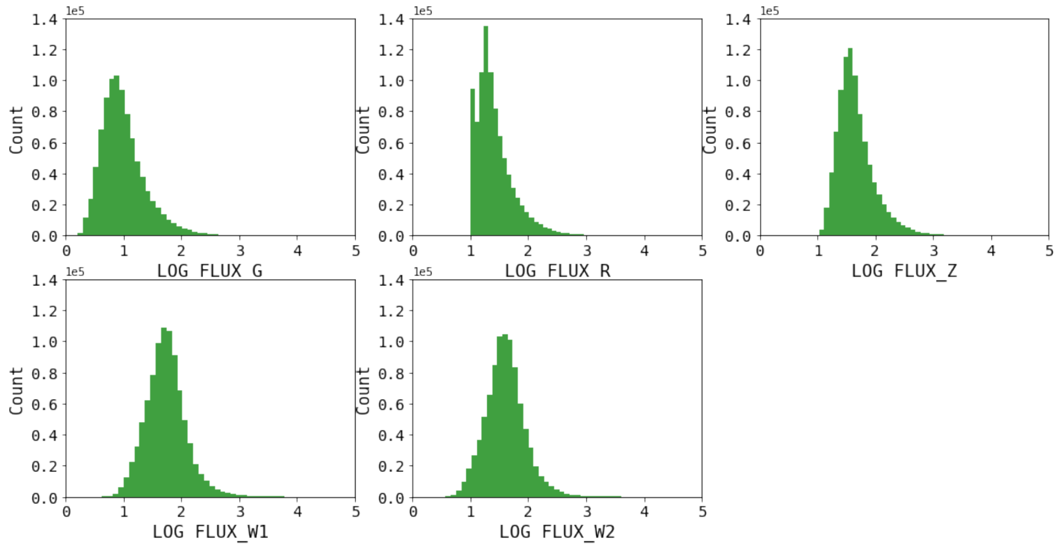


Figure 3.5: Distribution of the flux variable in the BGS subset.

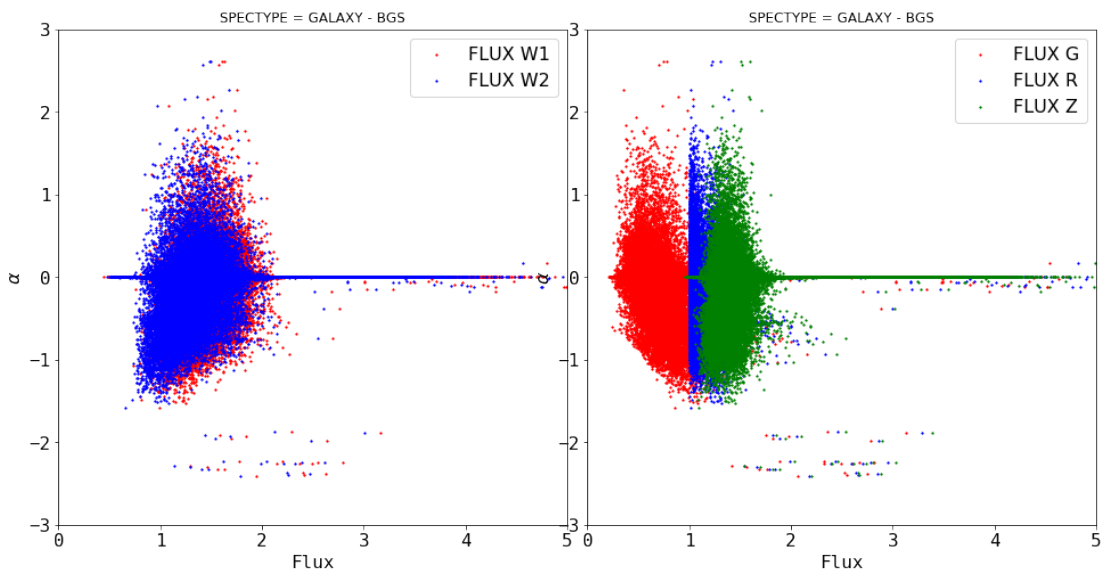
In order to keep track of the relation between TARZ y TRUEZ and see the behavior of the flux variable with respect to the two, we define the following variable

$$\alpha = \frac{TRUEZ}{TARZ}, \tag{3.1}$$

therefore, alpha have a value near 1 when the redshifts are along de 45-degree line.

We see in Figure 3.6 that each flux has a similar behavior, however, they are dispersed in different values and present different minimum values. As the flux magnitude increases, α tends to equal 1, meaning that as the fluxes received by the instrument increases, the redshift measurements are closer to the expected real values. Therefore, the dispersion regions may contain the information necessary for the ML models to learn no predict better redshifts. To see the influence of the dispersion at low fluxes over α, the cuts at different fluxes values in Figure 3.7 show the distribution of α at particular values of the fluxes.

The plots in Figure 3.7 are constructed by taking thin slices at fixed values of the fluxes in Figure 3.6 and constructing the histogram of the points that lie within the slice. The first approximation is to fit the distribution to a Gaussian distribution and estimate its mean and variance. The relation between mean and fluxes is shown in Figure 3.8, were the green points indicate the mean of α for the slice taken at the



(a) (b)

Figure 3.6: Relation between fluxes, TRUEZ and TARZ. (a) α as a function of W1 and W2 fluxes. (b)α as a function of G, R and Z fluxes. $\log_{10}$ values presented.

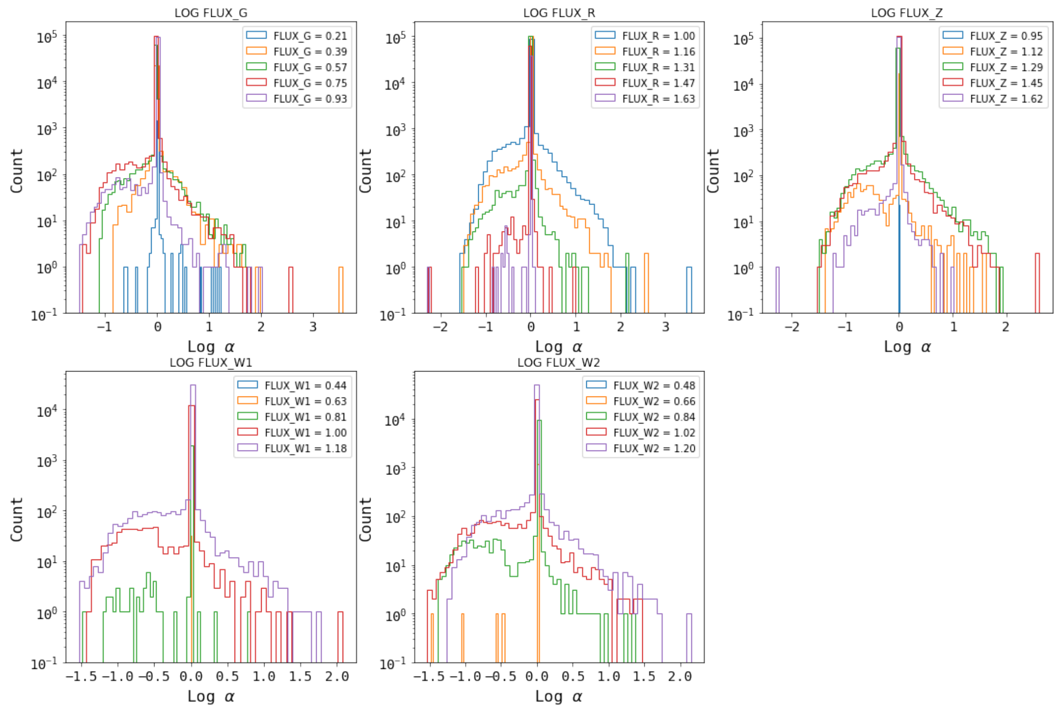


Figure 3.7: Distribution of α at different cuts in the flux variable from figure 3.6

given flux. The blue line in Figure 3.8 indicates when $TRUEZ = TARZ$, therefore, we see in that for each flux, there are ranges within $\mu(\alpha)$ is not 1, even when the typical error is small. These ranges are around 1 (in log scale) for all the fluxes. Also, note that at very high fluxes, the error in the mean increases, related to the fewer data available at those values and its high dispersity.

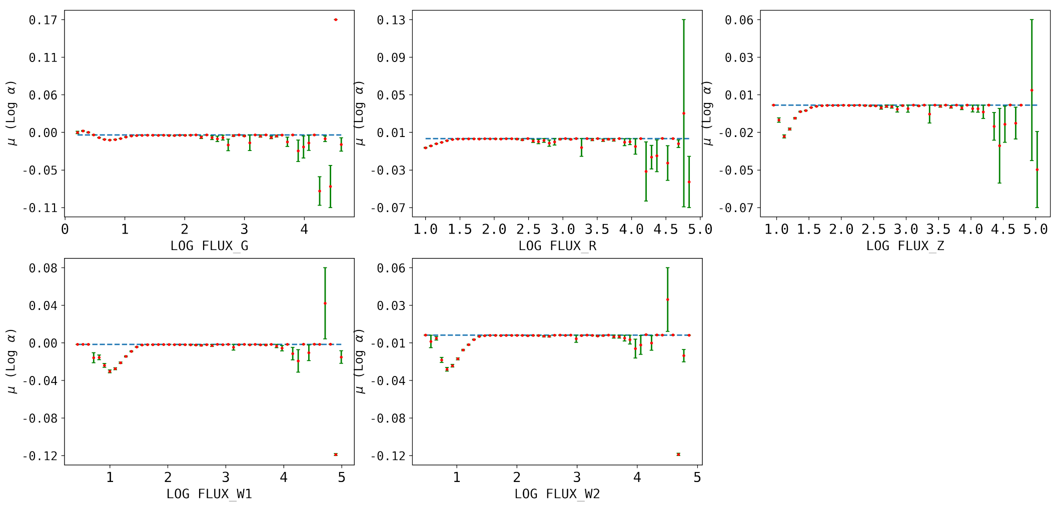


Figure 3.8: Mean (red) and standard error (green) of $\log_{10}(\alpha)$ for every Flux interval (slice).

3.4 Conclusions

The dataset consists of two sets of simulated observation, the truth-file, with the simulated expected results of redshifts (TRUEZ) from the survey targets, and the target-file with the simulated observations by the instrument of the targets redshifts (TARZ). The targets are classified as Galaxies, QSOs and Stars, being Galaxies the most representative type with 84.25% of the dataset. From this, the most interesting subtype of galaxies is the Bright Galaxy Survey (BGS) because of its patterns in redshift and the amount of data (49.51% of the Galaxy type). The results in the following chapter are focused only in the BGS subtype, using as input features for the machine learning algorithms the fluxes and the target redshift, and the output the true redshift.

Chapter 4

Results

4.1 Scalability of the training time of the ML models

Since the BGS dataset contains about 800000 data points, the training of algorithms, model selection, and validation can be too demanding, in processing and memory. Therefore, it is necessary to evaluate how the training time scales with the data when running in the High-Performance Computing (HPC) cluster of the University, taking account several variables as the number of processors to run in parallel the algorithm and the resources requested to the cluster. In particular, we will focus on the following variables:

$$\begin{aligned} m &= \text{Requested memory to solve the job [Gb]}, \\ n_{jobs} &= \text{Number of jobs to run in parallel}, \\ ppn &= \text{Requested processors per node}, \\ n &= \text{size of the dataset used}, \\ M &= \frac{n_{jobs} \times n}{ppn \times m}. \end{aligned}$$

The variable M measures the relationship between the requested resources and the resources used. Figure 4.1 shows how time increases as a function of variable M. Note that M^{-1} can be seen as a 'unitary memory', that is, the memory used per data per processor,

$$M^{-1} = \frac{m}{p_u \times n},$$

where $p_u = n_{jobs}/ppn$ is the fraction of the requested processors that is actually being used by the parallel code. Therefore, when M is big, it means that there is too much data per memory per processor, and the time to train the model is too high. It is worth mentioning that the time was measured for the kernel ridge regression model particularly, using grid search and 3-fold cross-validation. Others model may be faster or slower to train, but the behavior should be the same. The

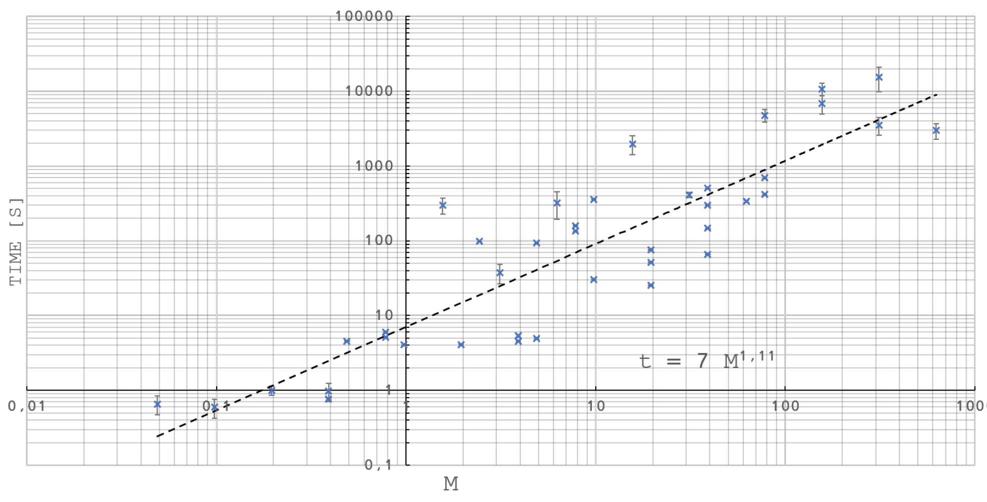


Figure 4.1: Time as a function of requested memory, number of data points and jobs used to train the model.

time of training for a given M can be approximated by the relation

$$t = 7M^{1.11}[s]. \tag{4.1}$$

This results have to be constrained to the waiting time in queue of the cluster, in general, the smaller M, the best time, but that means, using small dataset or increasing the resources, but for the second option, the more resources you request, the longer the queue and waiting time to execute the code. It was found that a configuration with $n_{jobs} = 16$**,** $ppn = 32$**,** $m = 128Gb$ **and** $n = 100000$, gives good results. n is critical since it determines how well the model will learn, 100000 data points work very well as it will be shown in the next sections.

4.2 Model results

The three model tested consist of a KRR, an SVR, and an ensemble model. The original BGS dataset was divided into a 75 - 25 train-test set, from the 75 %, each model was trained on a subset of size 100.000, as mentioned in the previous section. To account for the bias, we trained the same model in tree 100.000 dataset and the results were averaged. The selection of the model hyperparameters was made by using grid search a 3-fold cross-validation, using an 80-20 development-evaluation scheme.

4.2.1 Support Vector Regression (SVR) Model

The grid-search and cross-validation on the train-development set use the r^2 measure to select the best parameters.

```
Model 1:
Best parameters set found on development set:

{'gamma': 0.1, 'C': 1, 'kernel': 'rbf'}

Grid scores on development set:

0.877 (+/-0.001) for {'gamma': 0.001, 'C': 1, 'kernel': 'rbf'}
0.927 (+/-0.011) for {'gamma': 0.1, 'C': 1, 'kernel': 'rbf'}
0.848 (+/-0.017) for {'gamma': 1, 'C': 1, 'kernel': 'rbf'}
0.896 (+/-0.001) for {'gamma': 0.001, 'C': 10, 'kernel': 'rbf'}
0.924 (+/-0.010) for {'gamma': 0.1, 'C': 10, 'kernel': 'rbf'}
0.870 (+/-0.013) for {'gamma': 1, 'C': 10, 'kernel': 'rbf'}
0.898 (+/-0.006) for {'gamma': 0.001, 'C': 100, 'kernel': 'rbf'}
0.922 (+/-0.007) for {'gamma': 0.1, 'C': 100, 'kernel': 'rbf'}
0.868 (+/-0.020) for {'gamma': 1, 'C': 100, 'kernel': 'rbf'}

r2 score computed on the full evaluation set:

0.928

---------------------------------------------------------------------------
Model 2:

Best parameters set found on development set:

{'gamma': 0.1, 'C': 1, 'kernel': 'rbf'}

Grid scores on development set:

0.878 (+/-0.002) for {'gamma': 0.001, 'C': 1, 'kernel': 'rbf'}
0.928 (+/-0.007) for {'gamma': 0.1, 'C': 1, 'kernel': 'rbf'}
0.859 (+/-0.020) for {'gamma': 1, 'C': 1, 'kernel': 'rbf'}
0.892 (+/-0.002) for {'gamma': 0.001, 'C': 10, 'kernel': 'rbf'}
0.919 (+/-0.019) for {'gamma': 0.1, 'C': 10, 'kernel': 'rbf'}
0.859 (+/-0.010) for {'gamma': 1, 'C': 10, 'kernel': 'rbf'}
0.902 (+/-0.003) for {'gamma': 0.001, 'C': 100, 'kernel': 'rbf'}
0.909 (+/-0.006) for {'gamma': 0.1, 'C': 100, 'kernel': 'rbf'}
0.861 (+/-0.020) for {'gamma': 1, 'C': 100, 'kernel': 'rbf'}

r2 score computed on the full evaluation set:

0.92
```

```
----------------------------------------------------------------------------
Model 3:

Best parameters set found on the development set:

{'gamma': 0.1, 'C': 1, 'kernel': 'rbf'}

Grid scores on development set:

0.879 (+/-0.003) for {'gamma': 0.001, 'C': 1, 'kernel': 'rbf'}
0.926 (+/-0.019) for {'gamma': 0.1, 'C': 1, 'kernel': 'rbf'}
0.847 (+/-0.008) for {'gamma': 1, 'C': 1, 'kernel': 'rbf'}
0.898 (+/-0.005) for {'gamma': 0.001, 'C': 10, 'kernel': 'rbf'}
0.921 (+/-0.015) for {'gamma': 0.1, 'C': 10, 'kernel': 'rbf'}
0.868 (+/-0.009) for {'gamma': 1, 'C': 10, 'kernel': 'rbf'}
0.901 (+/-0.005) for {'gamma': 0.001, 'C': 100, 'kernel': 'rbf'}
0.891 (+/-0.018) for {'gamma': 0.1, 'C': 100, 'kernel': 'rbf'}
0.853 (+/-0.012) for {'gamma': 1, 'C': 100, 'kernel': 'rbf'}

r2 score computed on the full evaluation set:

0.93
```

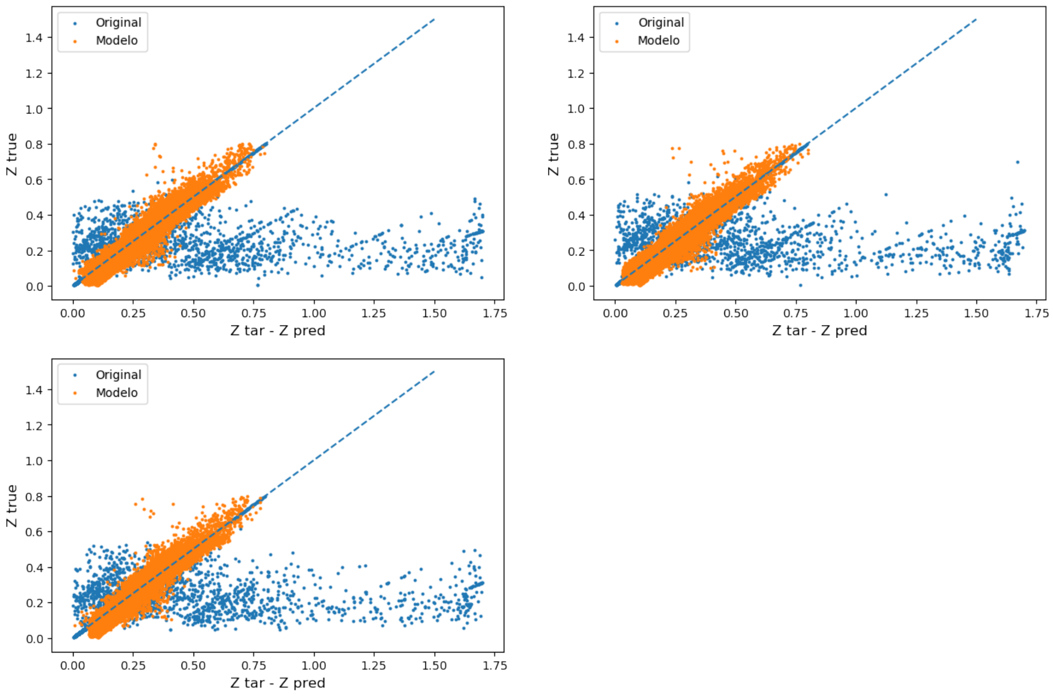


Figure 4.2: Results on the training set after training the SVR model.

The results on the training set of the best models are shown in Figure 4.2. On training, the model shows good results by gathering the blue dots (original data)over the orange dots (the model results after applying it to the training data). The blue scatter is substantially diminished and the data is clustered along de TRUEZ = TARZ line, However, the data is not entirely over the line, but in a wide range. Figure 4.3 shows the result on the evaluation set for each model. Figure 4.3 shows

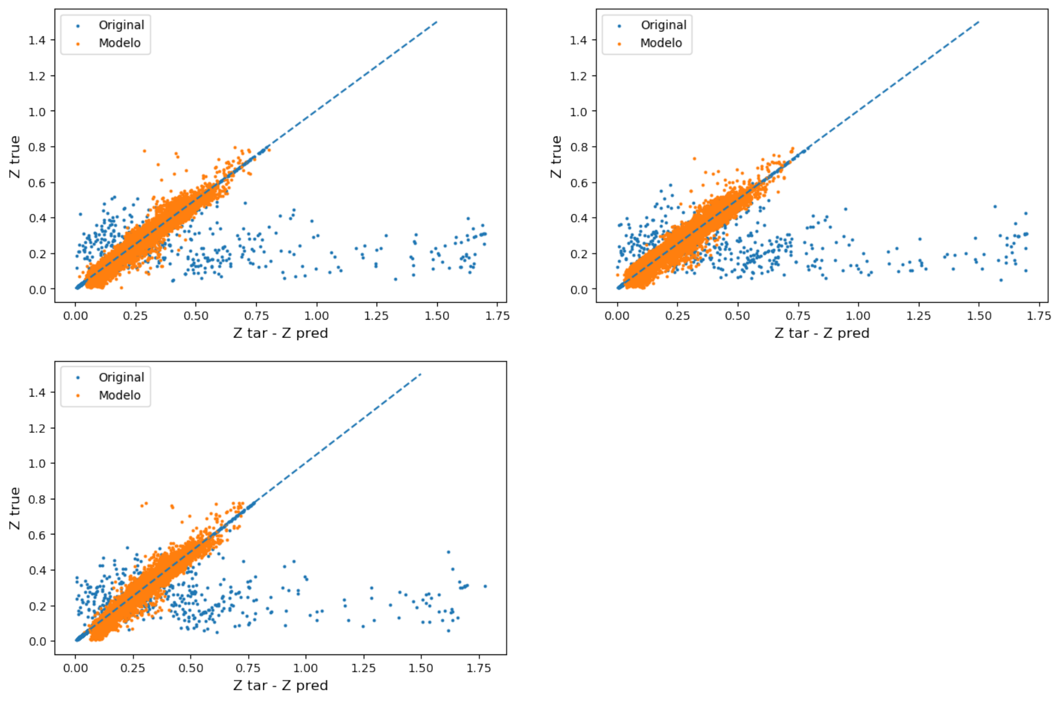


Figure 4.3: Results of each model on the development set.

how the model generalize to unseen data. The gathering capacity of the model to move the blue dots towards the TRUEZ = TARZ line present on the training set is also visible on the evaluation set, were the r^2 scores reaches 93% only.

4.2.2 Kernel Ridge Regression (KRR) Model

The grid-search and cross-validation on the train-development set use the r^2 measure to select the best parameters.

```
--------------------------------------------------------------------------
Model 1:
Best parameters set found on development set:

{'alpha': 0.0001, 'gamma': 0.2, 'kernel': 'rbf'}

Grid scores on development set for r2:

0.986 (+/-0.002) for {'alpha': 0.001, 'gamma': 0.1, 'kernel': 'rbf'}
0.988 (+/-0.002) for {'alpha': 0.001, 'gamma': 0.2, 'kernel': 'rbf'}
0.987 (+/-0.002) for {'alpha': 0.0001, 'gamma': 0.1, 'kernel': 'rbf'}
```

```
0.989 (+/-0.002) for {'alpha': 0.0001, 'gamma': 0.2, 'kernel': 'rbf'}

r2 score computed on the full evaluation set:

0.990
--------------------------------------------------------------------------
Model 2:
Best parameters set found on the development set:

{'alpha': 0.001, 'gamma': 0.2, 'kernel': 'rbf'}

Grid scores on development set for r2:

0.986 (+/-0.001) for {'alpha': 0.001, 'gamma': 0.1, 'kernel': 'rbf'}
0.988 (+/-0.002) for {'alpha': 0.001, 'gamma': 0.2, 'kernel': 'rbf'}
0.987 (+/-0.002) for {'alpha': 0.0001, 'gamma': 0.1, 'kernel': 'rbf'}
0.987 (+/-0.004) for {'alpha': 0.0001, 'gamma': 0.2, 'kernel': 'rbf'}

r2 score computed on the full evaluation set:

0.988
--------------------------------------------------------------------------
Model 3:
Best parameters set found on the development set:

{'alpha': 0.001, 'gamma': 0.2, 'kernel': 'rbf'}

Grid scores on development set r2:

0.987 (+/-0.001) for {'alpha': 0.001, 'gamma': 0.1, 'kernel': 'rbf'}
0.988 (+/-0.002) for {'alpha': 0.001, 'gamma': 0.2, 'kernel': 'rbf'}
0.987 (+/-0.003) for {'alpha': 0.0001, 'gamma': 0.1, 'kernel': 'rbf'}
0.987 (+/-0.003) for {'alpha': 0.0001, 'gamma': 0.2, 'kernel': 'rbf'}

r2 score computed on the full evaluation set:

0.9899
```

The results on the training set of the best models are shown in Figure 4.4. On training, the model shows better results than SVR by gathering the blue dots (original data)over the orange dots (the model results after applying it to the training data) on a tighter region. The blue scatter is substantially diminished and the data is

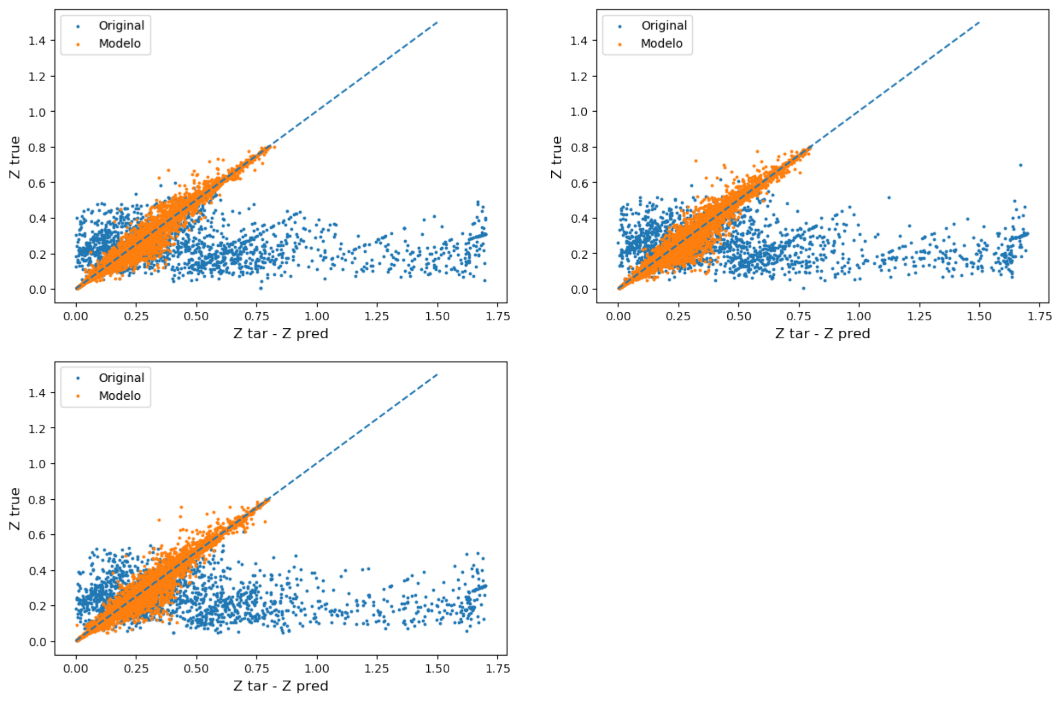


Figure 4.4: Results on the training set after training the KRR model.

clustered along de TRUEZ = TARZ line. Figure 4.5 shows the result on the evaluation set for each model. Figure 4.5 shows how the model generalize to unseen data.

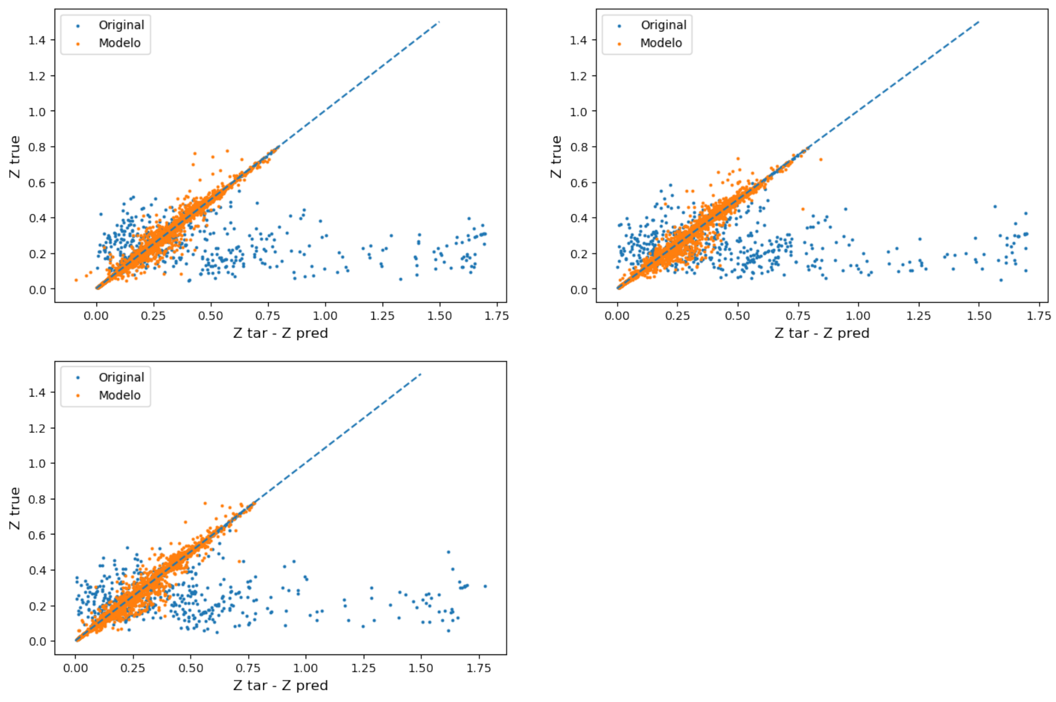


Figure 4.5: Results of each model on the development set.

The gathering capacity of the model to move the blue dots towards the TRUEZ = TARZ line present on the training set is also visible on the evaluation set, were the r^2 scores reaches 99%, a more better result than that obtaining using SVR.

4.2.3 Model Ensemble

The results from the KRR are much better than those of the SVR, also noting that the KRR model takes about half the time to train. Given that we have three different models to predict on the same dataset, we have to make this model to predict a single output. For this, we tried taking the maximum, the average or a weighted average of the model.

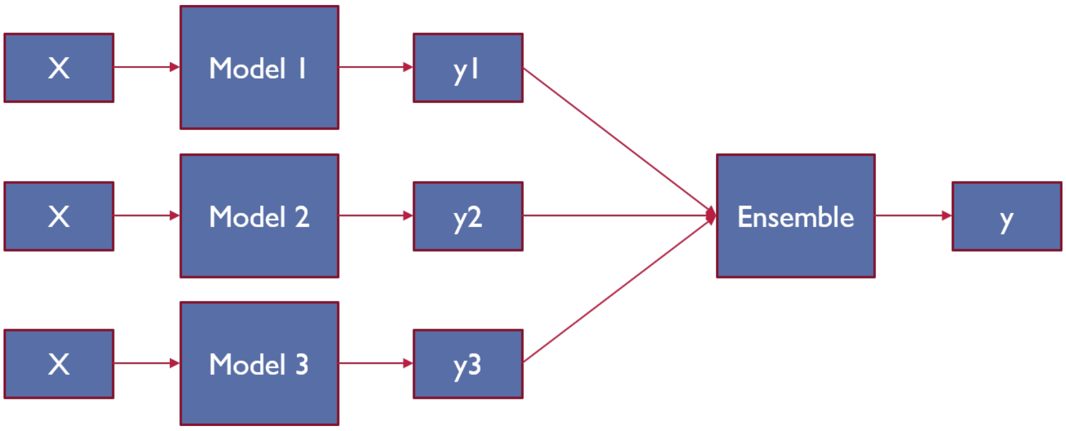


Figure 4.6: Graphic description of the ensemble model. It is formed based on the trained KRR models 1, 2 and 3.

Figure 4.6 shows this in a diagram, it is important to note that the ensemble is made over the **trained** models, that means that the parameters of the models are not re-learned in the ensemble. The only ensemble model that involves training, is the weighted average, where the weights are select by training a linear model. The results of the models on the **test** subset (unseen 25%) are shown in Table 4.1.

Model	r^2
Model 1	0.958
Model 2	0.956
Model 3	0.962
Model avg	0.962
Model max	0.949
Model w	0.983

Table 4.1: Results on the **test** set

Table 4.1 shows that the weighted average of the three models is the best model in terms of accuracy, taking into account that these are the predictions of the models on the totally-unseen test data corresponding to the 25% of the original BGS dataset.

4.3 Selección de modelo

The class of models evaluated is that of kernels models, the complexity of both, the SVR and the KRR is very similar, however, the KRR gave better results. This could be attributed to the lack of tuning of the ϵ parameter in SVR, due to the time of computation of SVR training. Smaller values of ϵ may result in a tighter region over the TRUEZ = TARZ line, therefore increasing the r^2. The weighted average model has at least 3 times more parameters that any model individually, its results are much better. Therefore, the best model that keeps simplicity and very good results is the weighted average model.

4.4 Conclusions

We evaluated two types of kernels method, the support vector regression (SVR) and the kernel ridge regression (KRR). The dataset was split in a training part (75%) and a test part (25%). Each model was trained three times on subset of the training part of size 100.000, this subset was also subdivided in an 80-20 development - evaluation datasets for the application of grid search and cross-validation. The KRR gave better results than the SVR, also, the three KRR models were ensemble as a weighted average, where the weights were found using a linear regressor. The ensemble average KRR model reached an r^2 of 0.98 on the test set.

Chapter 5

Conclusions

The object of this monograph was to apply machine learning (ML) methods to data coming from DESI end-to-end simulations to correct (or recover) its redshift measurements using observational variables as input.

We found that the true redshift of the Bright Galaxy Sample (BGS) can be recovered using the variables FLUX_G, FLUX_R, FLUX_Z, FLUX_W1 and FLUX_W2 as input to kernel methods. Using an ensemble model consisting of three KRRs (Kernel Ridge Regression) trained on subsamples of size 100.000, we were capable of approximate the DESI instrument redshift measurements to its true value up to a coefficient of determination $r^2 = 0.98$. This model shows great potential to further enhancement, i.e. the fine-tuning of the model hyper-parameters, different preprocessing methods, or even different classes of machine learning algorithms.

Although the trained models did not work for other classes of galaxies, probably because of the different distributions of fluxes. However, we infer that the same methodology could be used starting by training the models in the corresponding datasets. We expect that by doing so, the results of this work can be replicated and applied to all the classes in DESI.

Bibliography

[1] B. D. Nord *et al.*, “SPOKES: an End-to-End Simulation Facility for Spectroscopic Cosmological Surveys,” *Astron. Comput.*, vol. 15, pp. 1–15, 2016.

[2] A. Aghamousa *et al.*, “The DESI Experiment Part I: Science,Targeting, and Survey Design,” 2016.

[3] A. Aghamousa *et al.*, “The DESI Experiment Part II: Instrument Design,” 2016.

[4] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning.* Springer Series in Statistics, New York, NY: Springer New York, 2009.

[5] V. N. Vapnik, “The nature of statistical learning theory,” 2000.

[6] T. Hofmann, B. Schölkopf, and A. J. Smola, “Kernel methods in machine learning,” *Annals of Statistics*, vol. 36, no. 3, pp. 1171–1220, 2008.

[7] A. J. SMOLA and B. SCHOLKOPF, “A tutorial on support vector regression,” *Statistics and Computing*, pp. 199–222, 2004.

[8] S. L. P. Library, “Scikit-learn 0.19.2 documentation.”