Supervised Learning methods in regression & multiclass classification.

Lucas BIECHY & Angel REYERO LOBO
Teacher: Olivier COUDRAY

Institut Mathématique d'Orsay(IMO)
Paris-Saclay University

November 13, 2023

Abstract

Systematic review of various supervised methods for tackling two real-world problems.

In the first problem, centered on regression, we initially employed standard regression techniques including penalized linear regression, Generalized Additive Models, SVR, Random Forest, and Boosting. Subsequently, we incorporated an additional learning layer to combine insights from these diverse approaches.

In the second problem, involving multiclass classification, we similarly applied standard classification methods. Additionally, we applied another aggregation of base models. In parallel, we introduced a greedy, semi-supervised method inspired by the principles of the Self-training method.

Keywords: Regression, Classification, Supervised Learning, Semi-supervised Learning, Experts Aggregation (Bagging, Boosting, Stacking), Two-Sample testing.

Notations. Let $\mathcal{X}$ be the covariate space and $\mathcal{Y}$ the label space. We denote $\mathcal{F} := \{f | f : \mathcal{X} \to \mathcal{Y}\}$ the set of predictors. We denote $\mathcal{D}_n$ the training set and $\mathcal{U}_m$ the test set. Whenever we perform a train-test split on the training set $\mathcal{D}_n$, we use the notation $\mathcal{D}_{\text{train}}$ for the training subset and $\mathcal{D}_{\text{test}}$ for the testing subset. For a learning rule $\widehat{f} : \bigcup_{i=1}^{\infty} \to \mathcal{F}$ we usually omit the dependence on the training set and simply denote $\widehat{f}_n := \widehat{f}\,(\mathcal{D}_n)$ unless there are specific cases where we intend to emphasize this relationship. For the classification problem, we denote K the number of classes, which will be the set $\{1, \ldots, K\}$. Given a set $\mathcal{A}$ we denote by $\text{card}(\mathcal{A})$ its cardinality.

Contents

1 Introduction

The objective of this document is to provide an explanation of the various methods that have been employed within the framework of two data challenges introduced by Olivier Coudray as part of the Advanced Supervised Learning and Data Challenge course in the second year of the Master's program in *Mathematics and Artificial Intelligence* at the *Institut Mathématique d'Orsay* (IMO) at Paris-Saclay University.

The first problem goal is to predict the quality of wine on a scale of 1 to 10. Because the labels have an inherent order, this problem is approached as a regression task, as discussed in Section 2.

The second problem involves classifying celestial objects into one of three classes (0, 1, or 2). This problem is treated as a multiclass classification task, which is elaborated upon in Section 3.

General aspects of the methodology: In both scenarios, we commonly employ techniques that draw heavily from Scikit-Learn's methods (Pedregosa et al. (2011)). However, we have also leveraged other Python libraries like CatBoost(Prokhorenkova et al. (2018)), XGBoost(Chen and Guestrin (2016)) or pyGAM(daniel servén et al. (2018)).

To establish consistent hyperparameter settings for the algorithms, we often use cross-validation, thereby ensuring the robustness and independence of our model configurations.

2 Regression problem: Wine quality prediction.

In this section, we outline our approach to the regression problem of predicting the quality of wine, rated from 1 to 10, based on 12 covariates that explain its chemical properties. Although the label is discrete, given its inherent order, we can treat it as a regression problem.

We commence by detailing the data and the preprocessing techniques applied. Then, we explain the learning techniques used to train the models.

This project is part of a Kaggle data challenge for our MSc class.

All code is publicly available.

We observe that the metric used to compute the accuracy of a predictor $\widehat{h}$ in this problem is the coefficient of determination, commonly denoted as R^2, an defined by

$$R^2 := 1 - \frac{\sum_{i=1}^{n}(Y_i - \widehat{h}(X_i))^2}{\sum_{i=1}^{n}(Y_i - \overline{Y})^2}, \tag{1}$$

where $\overline{Y} := \frac{\sum_{i=1}^{n} Y_i}{n}$.

2.1 Preprocessing & data exploration

We have a training set with 4547 observations with 12 covariates each. As a result, we can approach this problem from a *standard* perspective without the complexities of high-dimensionality. The covariates include *fixed acidity, volatile acidity, citric acidity, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol* and *wine type*. All covariates are continuous except for *wine type*, which is denoted as 0 for red wine and 1 for white wine. Additionally, we have ensured our dataset is clean, free from repeated wine IDs or missing values.

We begin by representing the data thanks to the boxplot depending on the wine type as shown in Figure 1. We notice a great shift in multiple covariates that indicates us that they potentially do not follow the same distribution.

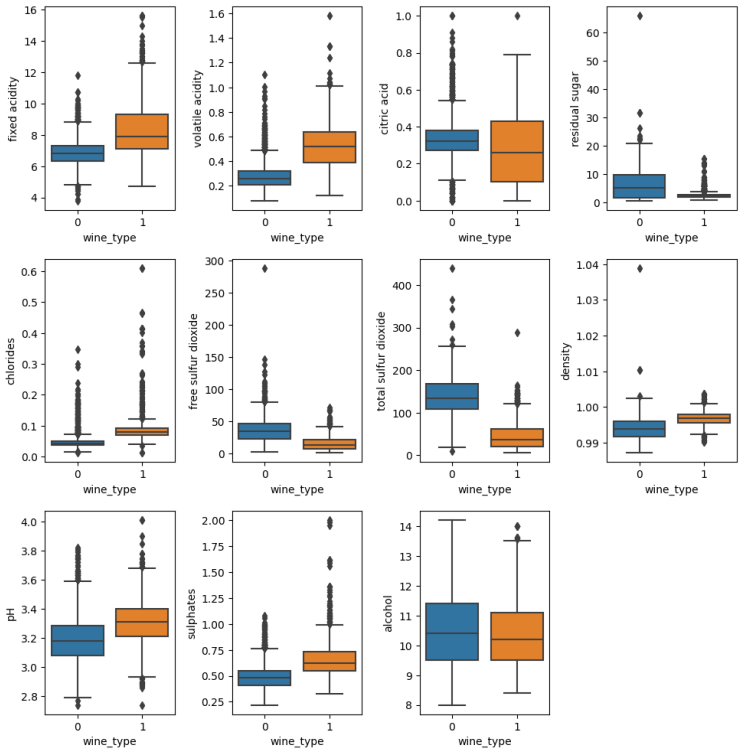

Figure 1: Boxplot of the covariates based on wine type, illustrating substantial shifts in several cases, indicating potential differences in their distributions.

Additionally, distinct patterns based on wine type can be observed, as shown, for example, in Figure 22. In such cases, applying the same predictor for both types might not be optimal. Specialized predictors for each type could be more suitable. Simple methods like linear regressions might not perform well in such scenarios due to their limited expressiveness, as discussed in the chapter on linear regression in Hastie et al. (2001).

Hence, the aim is to assess whether both samples follow the same distribution or not, which is the exact objective of the Two-Sample testing problem. Numerous approaches have been suggested to address this challenge. For instance, methods such as the one proposed in Gretton et al. (2012a), which are kernel-based tests leveraging the Maximum Mean Discrepancy (MMD) as a measure of distance between distributions, are well-suited for this context. However, their effectiveness crucially depends on the choice of the kernel. One method to address this is the hold-out technique proposed in Gretton et al. (2012b), where half of the data is used to train the kernel and the other half for the actual test.

Yet, alternative techniques such as MMD-FUSE, proposed in Felix Biggs and Gretton (2023), can avoid the need for data-splitting by employing an unsupervised learning approach while maintaining non-asymptotic test-level. In our analysis, we have used the latter test to confirm the anticipated difference in distributions between the two samples, employing the available code from the referenced article

Finally, we have confirmed the feasibility of this separation—ensuring we have enough data for each wine type to train a specific model. Specifically, we have 3411 observations for red wine,

constituting 75% of the dataset, and 1136 observations for white wine, representing the remaining 25%.

We observe from Figure 2 that certain labels are underrepresented in both datasets, whereas predominantly, there is a notable overrepresentation of mean-quality wine. This prevalence might lead to potentially unreliable information.

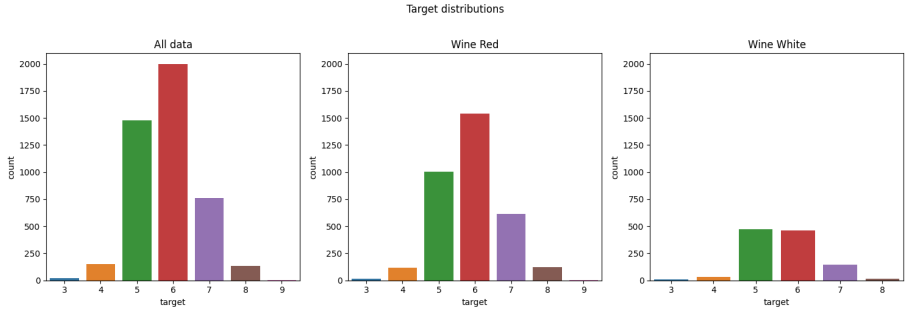

Figure 2: Count plot for the labels in the entire training set and then after the data split.

Therefore, we need a distinct data exploration for each wine type.

2.1.1 Red wine data exploration

We start by examining the correlation among the covariates. As shown in Figure 3, there is not a prevailing high correlation across the covariates. However, we do notice a strong correlation between *density* and *residual sugar*, and a significant negative correlation between *density* and *alcohol*. Even though, we choose not to transform the covariates, as done for instance in Section 3.

Next, we seek to gain an intuitive understanding of wine quality concerning a specific covariate. For instance, we would like a covariate c to see if higher values correspond to better wine, indicating a linear correlation. To explore this, we employ box plots. From Figure 4, we derive several insights. Unfortunately, we do not observe the wanted linear relationships with a covariate c.

However, we do notice, for instance, an interesting effect with *alcohol*. Wines rated as *mean*(with a score of 5) tend to improve as alcohol content increases. Contrarily, on the other tail, as alcohol content increases, the quality diminishes. A similar pattern is observed with pH, and the inverse relationship with density. Consequently, for predictive purposes, combining covariates in various ways will be essential to enhance accuracy.

It is crucial to note the underrepresented classes, such as quality 3 or 9, will exhibit more variability, potentially leading to less reliable information.

Further insight into the relationships between covariates can be observed through a pairplot, as shown in Figure 23 in the Appendix.

2.1.2 White wine data exploration

Similar to the case of red wine, we proceed by creating a correlation map. Illustrated in Figure 5, we observe that most cases are lightly shaded, indicating a lack of strong correlation, except for *fixed acidity* and *citric acid*, also between *fixed acidity* and *density* and high negative correlation between *fixed acidity* and *pH*.

Similar to the approach taken with the red wine, we use boxplots to examine the relationship between the covariates and the label. In this case, we notice more apparent linear relationships, such

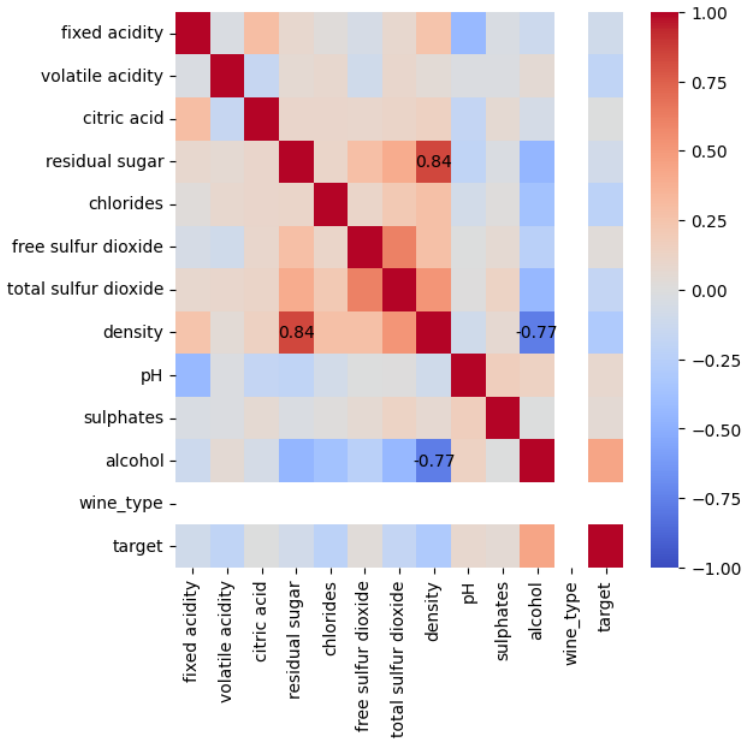

Figure 3: Red wine correlation map. We observe a strong positive correlation between *residual sugar* and *density*, as well as a notable high negative correlation between *density* and *alcohol*.

as those observed with *volatile acidity* or *citric acid*. However, it is notable that these relationships are not entirely linear at the extremities. Hence, incorporating more complex relationships will be necessary to improve the accuracy.

2.2 Learning

In this section, we explain our approach to the learning process applied in this regression problem. We initiated the process with standard regression methods, such as linear regression. Subsequently, we aimed to augment the expressiveness of our linear regression by establishing connections between covariates, filtering only valuable ones via model penalization. Following this, we aimed to capturing complex interactions by utilizing a Generalized Additive Model (GAM). To expand the model's expressiveness, we introduced various kernels for the Support Vector Machine adapted for regression task (SVR).

Continuing our exploration, we used expert aggregation methods through bagging and boosting, and then experimented with Neural Networks.

After implementing these methods, we considered *a posteriori* techniques that we applied to the predictions generated by the previous algorithms. The first method involves a meta-aggregator similar to the stacking method, while the second method introduces a residual correction filter

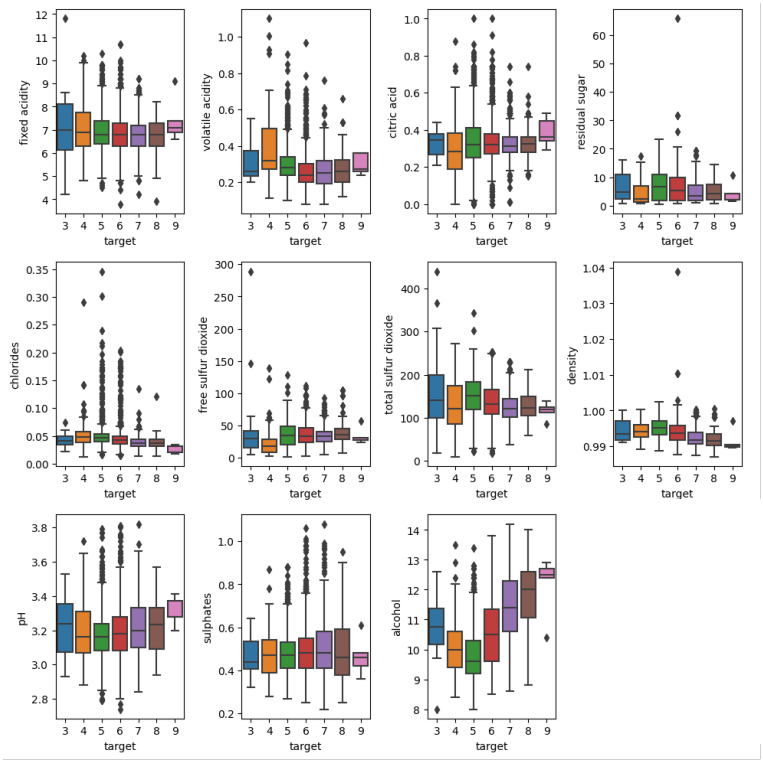

Figure 4: Red wine boxplot.

2.2.1 Linear Regression

Even though we have already pointed out the necessity to include more relationships than the linear ones, we are going to start from this ordinary least squares Linear Regression. It is based on the following minimization problem:

$$\widehat{\beta} \in \underset{\beta \in \mathbb{R}^{p+1}}{argmin} \|Y - X'\beta\|^2,$$

where X' represents the augmented covariate space to accommodate the intercept, and $p+1$ stands for the number of covariates plus one for the intercept. In our case, $p = 11$ as we have 11 covariates accounting for the displayed wine type.

In the sequel we are going to omit this distinction with the intercept to lighten the notation, but it will always be included.

We may observe higher accuracy in the white wine than in the red wine, even if we have a lot more observations for the red wine than for the white wine to train the specific model. This can be explained by the more linear relationships outlined in the boxplots in Section 2.1.2, making linear regression more suited for the white wine task than for the red wine task.

2.2.2 Linear Regression with penalizing terms

Expanding the covariates space: As outlined in the data exploration section, achieving a model with greater proximity to real data demands more flexibility than a linear approach. A preliminary

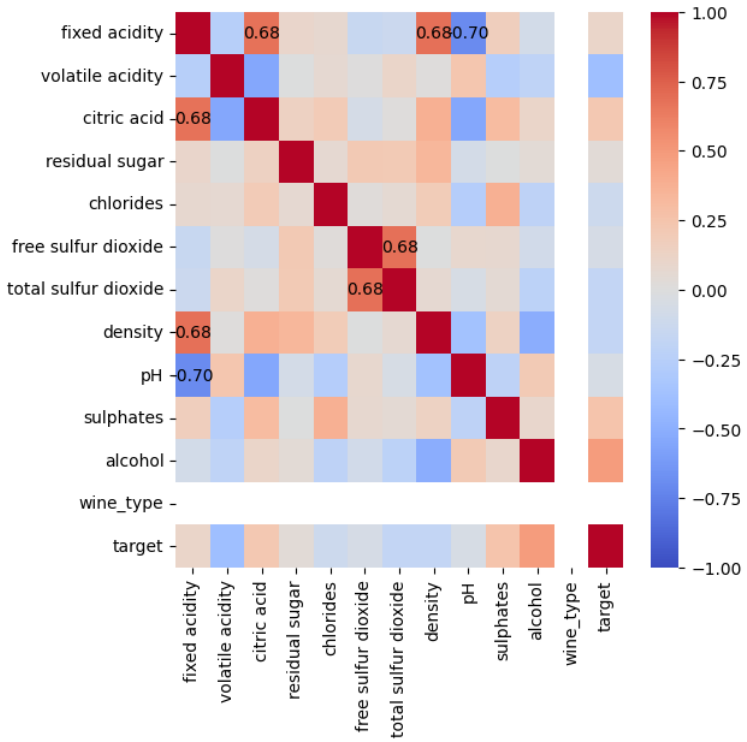

Figure 5: White wine correlation map. We observe a strong positive correlation between *fixed acidity* and *citric acid*, also between *fixed acidity* and *density* as well as a notable high negative correlation between *fixed acidity* and *pH*.

strategy involves expanding the covariate space to allow for second-order interactions among the covariates. This expansion involves multiplying each column of the dataset with every other column. Consequently, this process yields an expanded covariate space of size p (representing the initial space) plus $\sum_{i=1}^{p} i$ (the sum of interactions among the covariates). In our case, this expansion results in $11 + \frac{11*12}{2} = 77$ covariates.

This expansion was conducted without an initial data-driven justification for each added variable, potentially introducing noise into the model. To address this, we will reduce the model complexity using penalization criteria to avoid the overfitting.

Lasso penalization: The first penalizing method we are introducing is the standard Lasso regularization, which applies an l^1 penalization specifically designed for variable selection. It is associated with soft-thresholding and takes the following explicit form for linear regression:

$$\widehat{\beta} \in \underset{\beta \in \mathbb{R}^p}{argmin} \|Y - X\beta\|^2 + \lambda|\beta|_1.$$

We notice that as the coefficient λ increases, the penalty on the variables intensifies. Therefore, tuning the λ parameter becomes crucial to control the regularization effect.

Multiple approaches have been developed in order to compute this minimization. In this project, we opted for the Least Angle Regression (see Efron et al. (2004)) because it is computationally just

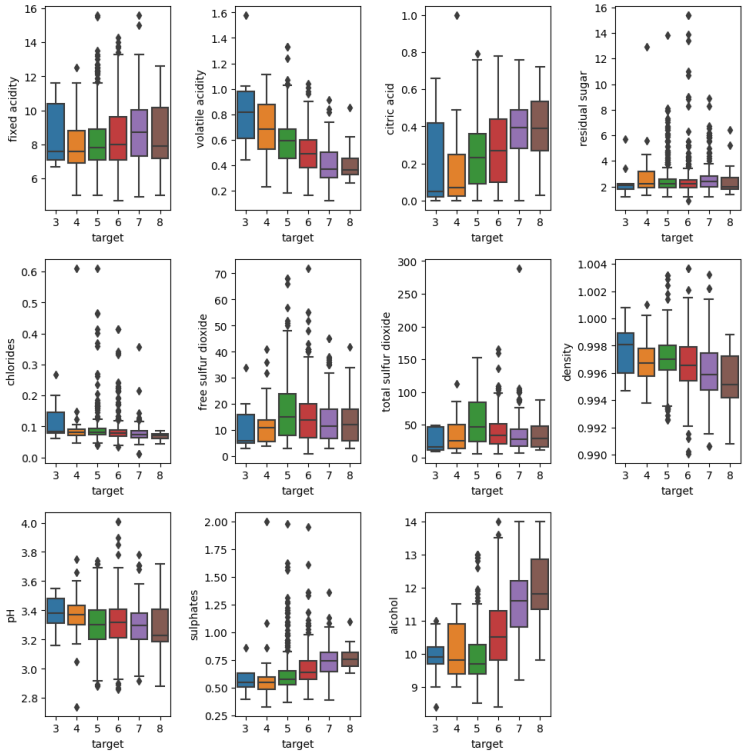

Figure 6: White wine boxplot.

as fast as forward selection and it produces a full piecewise linear solution path, which is useful in cross-validation to tune the model.

A significant limitation of the LASSO approach is the bias introduced by the penalization term. With its soft-thresholding, it drives the estimates toward zero, diminishing the estimates even for crucial covariates. One proposed solution is to use LASSO solely for selecting important covariates and then follow up with a standard linear regression using those selected variables. Alternative methods, such as the Adaptive Lasso or Gauss Lasso(see Giraud (2021)), could also have been considered.

Akaike Information Criterion(AIC) and Bayesian Information Criterion(BIC): Another standard criterion for model selection involves AIC and BIC. AIC is derived from frequentist probability, while BIC is grounded in Bayesian probability. The penalty term incorporated by AIC is smaller than that of BIC, rendering it more conservative and more open to accepting complex models.

However, we do not notice a substantial difference compared to the previous model without the covariate expansions. This lack of significant change is due to the fact, as previously stated, that the covariate space has not been expanded based on a data-driven criterion focusing on specific interactions to introduce into the model. To address this, we introduce the Generalized Additive Model (GAM), which offers enhanced expressiveness for capturing interactions.

2.2.3 General Additive Models(GAM)

GAMs are smooth semi-parametric models, similar to Generalized Linear Models(GLM), incorporating a link function g and additivity. However, they are more versatile as they permit non-linear functions of features.

As our aim is regression, the link function g we will use is the identity.

In practice, we will apply the already implemented methods in the python library pyGAM(daniel servén et al. (2018)).

Thanks to the model's additivity, we can assess the impact of each interaction by either adding or removing it. Hence, we will undertake an iterative manual procedure, beginning with a basic model and gradually incorporating feature functions. This process commences with the classic linear model:

$$\text{gam} = \text{LinearGAM}(l(1) + ... + l(11)),$$

where we recall that we have 11 covariates.

Then, we try to add smooth functions. To do this, we refer to the boxplots of each variable. Whenever we see that the means of the boxplots follow a polynomial function, we try to add a spline function on the variable. For example, as shown in Figure 7, we thought that adding a spline function on the alcohol variable would give us more information because it seems to follow a polynomial function, draw in red. Following a data-driven methodology, we will include or remove splines or linear covariates based on their significance in the model.

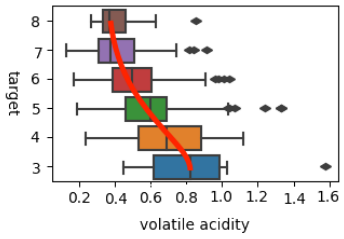

Figure 7: Visual representation of identified data patterns for incorporating splines.

We also aim to identify smooth interactions in the data by referencing pair plots. When observing a potential polynomial trend among data points, we introduce a tensor spline function to capture these interactions.

For instance, we note a visible pattern highlighted in red between the *pH* and *fixed_acidity* variables in Figure 8, which makes us consider incorporating a tensor spline between them to extract additional information.

Following a similar data-driven methodology, we will include or remove tensor splines or linear covariates or splines based on their significance in the model.

2.2.4 SVM for Regression

We note that this method is an extension of the classification technique SVM, which is detailed in Section 3.2.4 for celestial object classification. In essence, we adopt a similar principle of considering only a subset of the training data to form predictions. Just as the standard SVM focuses on points that are misclassified or near the margin to construct its margin, in this regression scenario, we consider only the subset of points where predictions deviate significantly from the true labels.

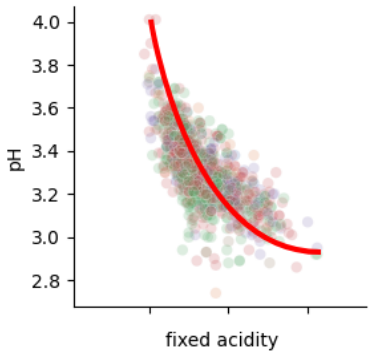

Figure 8: Visual representation of identified data patterns for incorporating smooth interactions.

Like the standard SVM, kernels can be introduced to extend the feature space through mapping to a higher-dimensional space, enabling the construction of more expressive hyperplanes.

Additionally, similarly to the standard SVM, it incorporates the regularizing term C, where the regularization strength is inversely proportional to the parameter.

2.2.5 Random Forest

Decision tree: This method attempts to partition the space by posing binary questions, each on a single variable. We end up separating the covariate space $\mathcal{X}$ in m regions given by $(R_1, ..., R_m)$. Then, the regressor is given by

$$f(x) = \sum_{j=1}^{m} v_j \mathbb{1}_{x \in R_j}.$$

Finding the optimal partition of the space is infeasible, thus multiple heuristic algorithms are proposed. Additionally, we must estimate the values v_j. To achieve this, we minimize the empirical risk within each region, maintaining a constant value for each, which is given by $\widehat{v}_j = \underset{v \in \mathcal{Y}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} c(v, Y_i) 1_{X_i \in R_j}$, where c is the cost function. In our case, derived from the R^2 criterion, we aim to minimize the quadratic loss, leading to the typical estimate obtained by the empirical mean within each region. (We could have directly observed that theoretically, $v_j = \mathbb{E}[Y|X \in R_j]$ and we estimate it with the empirical mean).

It is highly flexible but faces the inconvenience of potentially interpolating the entire dataset. To manage this, we tested multiple parameters, like the minimum number of samples required to split a leaf or the maximum allowed depth.

One drawback is that, as demonstrated in Section A.2.1, one covariate tends to concentrate the importance, resulting in underuse of other covariates. To address this, we aim to circumvent the issue by selecting only a subset of possible covariates at each split, enabling the extraction of information from all features.

Another drawback of its flexibility is the significant stochastic error it induced. To address this, the bagging method aims to average out this error across multiple predictors constructed to be as uncorrelated as possible, introducing the Random Forest.

Random Forest: It is a bagging predictor introduced by Breiman (2001) that aim to aggregate multiple decision trees as uncorrelated as possible. To do so,it leverages the bootstrap technique,

involving resampling the data with replacement to simulate new datasets. A decision tree is trained for each bootstrap sample. The final predictor is determined by averaging the individual predictors:

$$\widehat{f}_{\text{bag}}(\mathcal{D}_n)(x) = \frac{1}{n}\sum_{i=1}^{b} \widehat{f}_i(x).$$

The primary advantage of this method is its reduction of variance while sustaining an equal level of bias. Another benefit is the diversity introduced not only by the data bootstrap but also by randomly sampling a subset of features for conducting splits. This method is parameterized by *max_features* to determine the size of the allowed covariates subset. In our scenario, the exclusion of certain features could potentially enable exploration of new space separations, potentially counterbalancing the drawbacks observed in our previous decision trees regarding bias and variance.

Therefore, we also seek to enhance the method by fine-tuning multiple parameters, such as the number of estimators or the hyperparameters inherited by the decision tree, like the maximum depth.

As detailed in Section A.2.2, the prior issue of a single dominant covariate has been addressed, and the weight is more evenly distributed, although the *alcohol* covariate still remains the most significant.

2.2.6 Gradient Boosting

It is an aggregation method, different from bagging, where the base predictors are not built in parallel and independently. Instead, they are constructed iteratively based on the residuals from previous iterations. The initial problem is given by

$$\widehat{f} \in \underset{(h_m,\alpha_m)_{1\leq m\leq M}}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} c\left(\sum_{m=1}^{M} \alpha_m h_m(X_i), Y_i\right).$$

However, this approach is not feasible. Therefore, a general approach to solve this problem is an approximation given by:

1. Initialize with $\widehat{f}_0 = 0$.
2. For each m from 1 to M, find

$$(\widehat{h}_m, \widehat{\alpha}_m) \in \underset{\alpha\in\mathbb{R}, h\in\mathcal{F}}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} c(\widehat{f}_{m-1}(X_i) + \alpha h(X_i), Y_i)$$

   with $\widehat{f}_{m-1}(X_i) = \sum_{k=1}^{m-1} \widehat{\alpha}_k \widehat{h}_k$.
3. Use the final predictor : $\widehat{f} = \widehat{f}_M = \sum_{m=1}^{M} \widehat{h}_m \widehat{\alpha}_m$

Even with this simplification, the minimization step remains too costly. Therefore, for the regression problem, a gradient step is suggested. Specifically, in quadratic regression, the method relies on the residuals of previous predictions. We update the predictor using a fixed learning rate, similar to the gradient descent. However, similarly to the gradient descent, multiple approaches can be proposed to adaptively set this learning rate.

In practice, we will use the algorithms already optimized and implemented in the Python libraries XGBoost and CatBoost (see Chen and Guestrin (2016); Prokhorenkova et al. (2018)).

2.2.7 Neural Networks

A neural network consists of a series of interconnected layers, each formed by multiple adapted weight combinations from the preceding layer. These combinations are activated by a function to introduce non-linearity, a key aspect that allows neural networks to act as universal approximators (see Kratsios and Papon (2022)).

Despite the opacity of these models, their consistent accuracy across numerous cases makes us consider them in our models.

We acknowledge the potential for optimizing various parameters and exploring diverse architectures to suit different objectives. However, in this project, we focus on using the methods already implemented in the *MLPRegressor* model of Scikit-learn. This includes optimizing essential parameters like the number and size of layers, choice of optimizers, activation functions, the number of iterations, and strategies for learning rate optimization.

2.2.8 Meta-experts aggregation

Through the application of bagging methods, we have learned how to leverage the power of multiple predictors, even when their individual accuracy was not particularly high. By aggregating these predictors, we successfully mitigated the *stochastic/variance* error caused by the flexibility of the predictor class. In this section, we detail various techniques we have employed to imitate this strategy to our particular problem. A similar approach will be employed in the classification problem.

We denote $\mathcal{H}$ the set of learning rules

$$\mathcal{H} := \left\{ \widehat{h} : \bigcup_{i=1}^{\infty} (\mathcal{X}_i, \mathcal{Y}_i) \to \mathcal{F} \right\}.$$

Specifically, we use the notation $\widehat{\mathcal{H}}$ when the set of learning rules is trained on the entire training set $\mathcal{D}_n$, and $\widehat{\mathcal{H}}_{\text{train}}$ when it is exclusively trained on the training subset $\mathcal{D}_{\text{train}}$:

$$\widehat{\mathcal{H}} := \left\{ \widehat{h}\left(\mathcal{D}_n\right) \in \mathcal{F} \middle| \widehat{h} \in \mathcal{H} \right\} \qquad \widehat{\mathcal{H}}_{\text{train}} := \left\{ \widehat{h}\left(\mathcal{D}_{\text{train}}\right) \in \mathcal{F} \middle| \widehat{h} \in \mathcal{H} \right\}.$$

Mean approach: In this naive first approach we only take the mean over the whole predictions giving the same power to all the regressors. Therefore, for a given entry X it predicts

$$\widehat{f}_{\text{mean}}(X) := \frac{\sum_{\widehat{h} \in \widehat{\mathcal{H}}} \widehat{h}(X)}{\text{card}(\widehat{\mathcal{H}})},$$

where card() indicates the cardinal.

Accuracy weighted mean approach: Similarly as before but multiplying each prediction by a weight in order to give more power to the more accurate regressors. We have only tested using raw accuracy measures, yet exploring alternative methodologies, such as exponentiation to amplify the distinctions in regressor weights, could be a consideration for future work.

We notice that both approaches aggregate linearly the predictors. With the next method we are going to aggregate them in a more complex and flexible way.

Residuals correction approach: In this approach, we draw inspiration from boosting methods to correct the residuals. We propose a two-level algorithm. The first step involves training the initial list of regressors $\mathcal{H}$ on the training subset $\mathcal{D}_{\text{train}}$. Subsequently, we use these regressors to make predictions on the test subset. Finally, we train another regressor on the last predictions. We refer to this algorithm as a two-level approach because it involves training two types of predictors: a first-level based on the base algorithms $\mathcal{H}$ and a second-level meta-aggregator used to combine the first-level predictions in a more complex manner. This second-level meta-aggregator, denoted by $\widehat{m}$, can be, for example, a Random Forest or a neural network.

With this approach, our objective was to understand how to effectively combine the first-level predictions to enhance our results. This concept is closely related to the correction of residuals. Our aim was to correct these residuals so that the final predictions on the test set involved applying this correction to the first-level predictions. It is worth noting that achieving independence between the first-level predictors $\widehat{\mathcal{H}}_{\text{train}}$ and the training set

$$\left\{\widehat{m} := \widehat{h}\left(\mathcal{D}_{\text{train}}\right)\left(\mathcal{D}_{\text{test}}\right) \middle| \widehat{h} \in \widehat{\mathcal{H}}\right\} \in \mathcal{F}$$

for the second-level algorithm is required. Otherwise, we would introduce a bias by using the predictions over the same set from which we have trained the predictors.

We were able to apply this method because we had a sufficient amount of data. Additionally, we observe that this two-level algorithm can be extended to create more layers while following the same principles.

Furthermore, drawing on memory techniques applied in Recurrent Neural Networks (as seen in, for example, Hochreiter and Schmidhuber (1997)), we intend to provide the meta-aggregator with information from the previous level. Specifically, we will train it using the raw data before it undergoes processing by the first-level predictors. Once all models are trained, to predict a new observation X, we will initially apply the base models, providing $\widehat{h}(X)$ for $\widehat{h} \in \widehat{\mathcal{H}}_{\text{train}}$. Subsequently, we will predict with

$$\widehat{m}\left(X, \left(\widehat{h}(X)\right)_{\widehat{h}\in\widehat{\mathcal{H}}_{\text{train}}}\right).$$

Upon implementing this method, we noticed parallels with the ensemble aggregation technique *Stacking* introduced by Wolpert (1992). Furthermore, we follow a similar procedure to ensure independence between the base-level predictors and the meta-aggregator. However, we have incorporated memory to extract information from the original data that our base predictors could not obtain.

It is important to note that in this approach, we have opted for a holdout method instead of utilizing cross-validation principles to reduce computational load. Typically, for a more robust and thorough analysis, employing the latter technique is recommended.

2.2.9 Student's filter

From the outset, we have approached this problem as a regression problem, even though we have noticed that the set of true labels $\mathcal{Y}$ is discrete. As a result, it is worth exploring the extent to which we compromise the integrity of the labels by treating them as continuous.

Subsequently, we will introduce an *a posteriori* technique aimed at correcting the residuals after the prediction of the model. Well-trained models tend to concentrate predictions around the neighborhood of the true labels, resulting in centered residuals. It is as if the values of the predictions, conditioned on the true label, follow a Gaussian distribution. To test this hypothesis, we can conduct a test where the null hypothesis assumes that the distribution of predictions conditioned on the label

follows a Gaussian distribution $\widehat{h}(X)|Y = k \sim \mathcal{N}(\mu_k, \sigma_k^2)$, while the alternative hypothesis suggests the contrary. One example of this type of test is the Shapiro-Wilk test.

Once this assumption is verified, we would like to exploit it in order to collapse the values for which we have a high probability of belonging to the class, thereby avoiding any cumulative bias resulting from the continuization of values. To do this, we will leverage the fact that the sample is Gaussian to construct a statistical interval of belonging to the class. We begin by estimating the mean and the variance for each class with

$$\widehat{\mu}_k = \frac{\sum_{i=1}^n X_i \mathbb{1}_{Y_i=k}}{\sum_{i=1}^n \mathbb{1}_{Y_i=k}} \qquad \widehat{\sigma}_k^2 = \frac{\sum_{i=1}^n (X_i - \widehat{\mu}_k)^2 \mathbb{1}_{Y_i=k}}{\sum_{i=1}^n \mathbb{1}_{Y_i=k} - 1}.$$

In the sequel we denote by $n_k = \sum_{i=1}^n \mathbb{1}_{Y_i=k}$ to lighten the notation. Then, we observe that the statistic

$$Z_k := \sqrt{n_k} \frac{\widehat{\mu}_k - \mu_k}{\widehat{\sigma}_k^2}$$

follows a Student's t-distribution with $n_k - 1$ freedom degree. We denote by $q_n^t(\alpha)$ the α quantile of a Student's t-distribution with n degrees of freedom. Finally, using all this information, we build up the confidence interval $[l_k; r_k]$ for the k-th class as

$$l_k := \widehat{\mu}_k - q_{n_k}^t(\alpha) \frac{\widehat{\sigma}_k^2}{\sqrt{n_k}} \qquad r_k := \widehat{\mu}_k + q_{n_k}^t(\alpha) \frac{\widehat{\sigma}_k^2}{\sqrt{n_k}}.$$

We observe that, thanks to this statistic, we will be able to adapt to each specific class, as the variance will be specific to each class, and the radius of the interval will decrease with a square root speed relative to the number of samples in the observed class.

Practically, once the bounds of the interval are computed, we can just assign to all the predicted values that are inside the mean of the class.

We notice that, up to this point, we have not explained how to determine the α hyperparameter. Moreover, one could be misled by the concept of confidence intervals into thinking that a higher α value would increase our certainty that the observation belongs to the class. In reality, it is the opposite. This is because as α increases, so does the diameter of the intervals.

Choosing the α: In fact, the choice of this parameter is not straightforward, as it involves a trade-off. On one hand, we aim to make the intervals as large as possible to minimize bias. On the other hand, enlarging the intervals increases the chances of assigning the label of the wrong class. Moreover, we observe that the gain obtained from a correct assignment is lower than the loss incurred by a wrong assignment. This observation arises from the quadratic dependence of the residuals on the determination coefficient (see equation (1)). As a result, a small residual becomes even smaller, but a larger one counts even more. Hence, from a practical standpoint, we decided to empirically compute this confidence level to improve predictions by a usual data-driven hyperparamete estimate.

We also note that for complete consistency, the mean and variance of the residuals should be computed in an independent set from the training set used for predictors, as training on the same set may lead to overfitting. Additionally, the set used to choose α should be independent not only from the training set but also from the set used to estimate the mean and variance. In our specific case, considering the need to divide by the total number of classes and the previous division for the meta-aggregator, we will omit this subsetting division as we lack sufficient data for accurate estimation of all quantities.

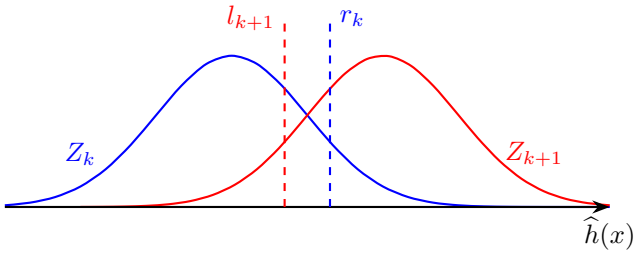

Figure 9: Overlap of intervals creating undecisive intervals between class k and $k+1$.

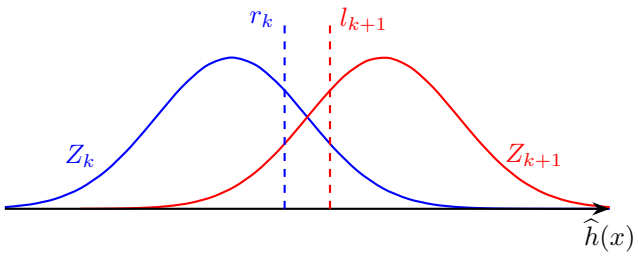

Figure 10: Interval reduction to tackle undecisive intervals.

Dealing with overlaps: As shown in Figure 9, there may be cases in which the *confidence* intervals overlap, making regimes where both labels could be assigned. This situation arises when the right boundary r_k of the k-th class exceeds the left boundary l_{k+1} of the $k+1$-th class.

Multiple approaches can be proposed to assign a class in these ambiguous regions. For example, a probabilistic approach could determine the class by assessing where the density is higher at a given point. However, as previously discussed, the benefits of accurate labeling are outweighed by the consequences of misclassification. Therefore, a conservative strategy is adopted, leaving these uncertain regions with their initial labels. This involves reducing the left boundary of the $k+1$-th class to meet the right boundary of the k-th class, and vice versa, as illustrated in Figure 10.

Underrepresentation of certain classes: In our specific dataset, as seen in Figure 2, it is evident that certain classes, such as Class 3, lack sufficient representation with a small number of observations. Consequently, accurate estimation of distributions for such classes becomes unfeasible, leading us to omit their consideration.

2.3 Discussion

This section primarily focuses on discussing the limitations of the choices made since the beginning of this data challenge, as well as exploring potential alternatives for future work.

Wine type split-up: In our project, we have consistently employed the concept of having a specific model for each type of wine. However, this data split raises concerns about whether we have sufficient data for each type to develop robust models, as shown in Figure 2. Hence, an alternative

approach could involve a two-level algorithm mixing both concepts. Initially, we could employ a global model, like a Random Forest trained across both wine types. Subsequently, for classes well represented in both types (such as the mean labels), employing a specialized model could enhance accuracy. However, for extreme labels like 3 or 9 where data might be insufficient to construct a robust model, retaining the initial prediction might be the pragmatic choice.

Furthermore, in the latest aggregation technique outlined in Section 2.2.8, a train-test split was necessary. This split was used to train the base-learners on one hand and the meta-aggregators on the other. However, due to these consecutive split-ups and the limited availability of data for extreme values in the initial dataset, the final subsets become considerably sparse, posing challenges in learning all the necessary information.

Regression: Although we initially approached this problem as a regression due to the inherent ordered nature of wine quality, our observations revealed a significant *mean effect*, as discussed in Section 2.1.1 through boxplot analysis. Notably, with as some covariates increase, the wine could be better or worse.

This trend persisted in our models, reflecting a bias toward concentrating values around the mean to mitigate method penalization, evident in the residuals showcased in Figure 11. These residuals, exhibiting a linear decrease, indicated a propensity to favor *normal* values over extremes.

To address this, a potential approach involves eliminating this pattern by disregarding the ordinal relationship of the labels. This could be done by training a classifier instead of a regressor.

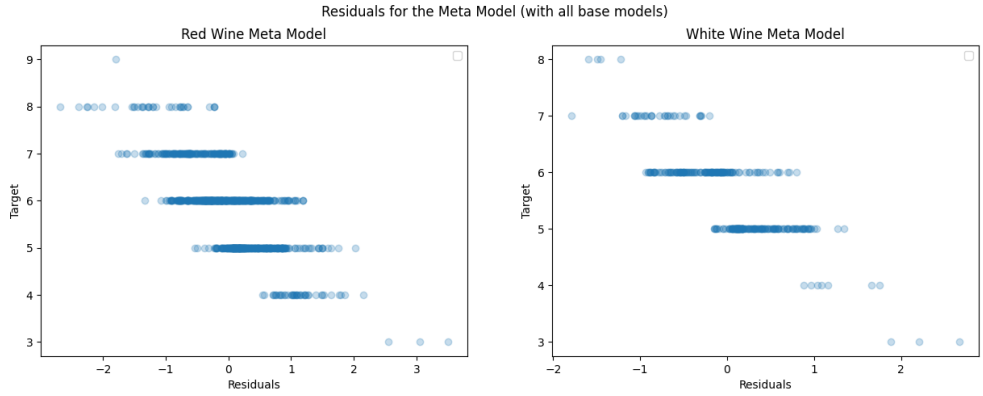

Figure 11: Residuals plot of the last meta-model trained over all the base-models. On the horizontal axis we have the residuals given by the difference of the true label Y with the predicted label $\widehat{Y}$. On the vertical axis we represent the true labels Y.

Filtering: Reviewing the final scores of our filtered models, it becomes evident that the accuracy did not improve; in fact, it often performed worse than the previous models. To investigate this, we initially examine the shift shown in Figure 11. This observation reveals that the predictions do not align with the centered assumption. Instead, they are shifted due to the *mean* effect, as previously discussed. This discrepancy is particularly visible in Figure 12, where the residuals of the Random Forest model are plotted. It is apparent that the predicted labels tend to cluster around the global mean value. Moreover, extreme values are notably scarce. As a result, the filtering method might not be effective as a post-modeling technique.

In future work, addressing this issue might involve artificially assigning greater weight to the underrepresented extreme values. For instance, a knockoff technique could generate synthetic observations resembling these underrepresented labels, or directly assigning more weight to these values in the learning methods. By correcting this imbalance, the residuals should align more closely with the mean, potentially enhancing the effectiveness of this *a posteriori* approach.

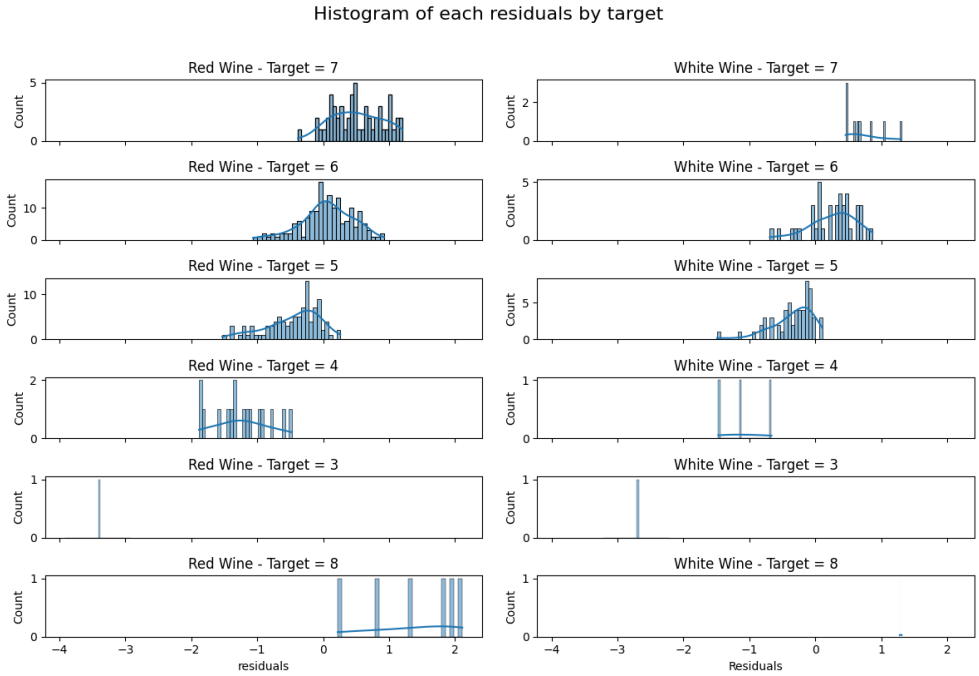

Figure 12: Distributions of the residuals of the Random Forest. It plots the difference of the true label Y with the predicted label $\widehat{Y}$ by label: $Y - \widehat{Y}$.

Variable importance: It is essential not just for the interpretability of the model, facilitating further studies—for instance, enhancing wine quality by specifically improving the most influential predictive variables—but also for refining the current model. This involves directing the model to extract more targeted information from these important variables, reducing noise from other covariates. For more information, we refer to Appendix A.2.

3 Multiclass classification problem: Celestial objects prediction.

In this section, we describe our approach for the multiclass classification problem of predicting celestial object types (galaxy, star, or quasar) using physics observation data sourced from the Sloan Digital Sky Survey.

This project is part of a Kaggle data challenge for our MSc class.

All code is publicly available.

The selected metric is the weighted F1-score, which equally values precision and recall. This is evident in the formula, as it is given by

$$F_1 := 2\frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}.$$

Additionally, this metric is weighted, factoring in class frequencies when computing the mean.

3.1 Preprocessing & data exploration

We observe that our training set consists of 52295 observations of 8 variables. Then, we do not have to deal with high-dimensional issues and we have enough observations to apply standard machine learning algorithms. The covariates consist of $\alpha, \delta, u, g, r, i, z$, and $redshift$. The labels correspond to values of 0, 1, or 2. It is worth noting, as illustrated in Figure 13, that while class 0 is more prevalent than the other classes, we still have an adequate number of samples for each class. When performing the train-test split, it is important to maintain the same proportion per class in each subset.

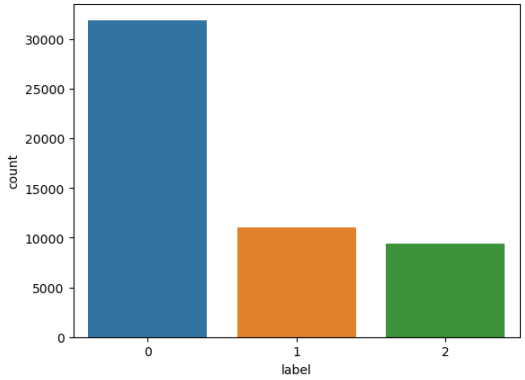

Figure 13: Number of observations per label in the training set.

All covariates are continuous and complete, lacking any missing values. Beginning our data representation with boxplots for these variables, we gain an initial understanding of their *discriminatory* potential. When each label is plotted separately, significant differences between the boxes of a variable become apparent. For instance, this is evident in the *redshift* variable, as illustrated in Figure 28 in the Appendix. However, this visual analysis does not account for inter-variable relationships.

In fact, as depicted in Figure 29 in Appendix, numerous patterns emerge from the data. Notably, combinations involving the *redshift* covariate are distinctive and valuable for our discrimination task. Additionally, the high correlation between variables like u, g, r, and i is evident, approaching linearity, as confirmed in Figure 14 where the correlation values are close to 1.Usually, machine learning algorithms work better with uncorrelated data. This is because there can be technical issues such as the inability to invert the covariance matrix and because it can create models that are no longer identifiable. To address this issue, we will try to employ dimensional reduction methods to ensure that each new variable contributes unique information after having scaled the data.

t-SNE: Our data reduction process begins with t-SNE(see Van der Maaten and Hinton (2008)), a method adept at representing high-dimensional data while maintaining point relationships. This technique relies on a perplexity parameter, determining the number of nearest neighbors considered

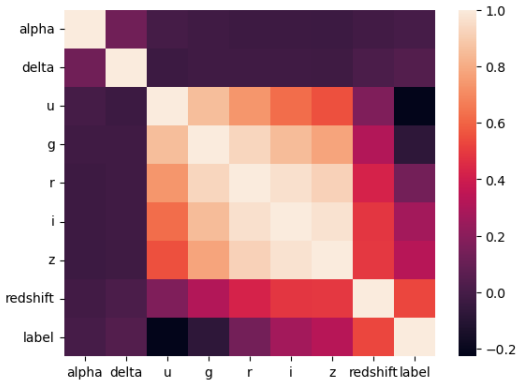

Figure 14: This plot illustrates the correlation between covariates. Notably, there is a high correlation between u, g, r, i, and z, as observed in the central light-shaded square.

in embedding construction. Experimenting with various perplexity parameters, as shown in Figure 15, revealed that the labels are intermixed in a manner that makes classification through this technique seem unfeasible. Thus, we aim to explore alternative dimension reduction methods with stronger discriminative potential.

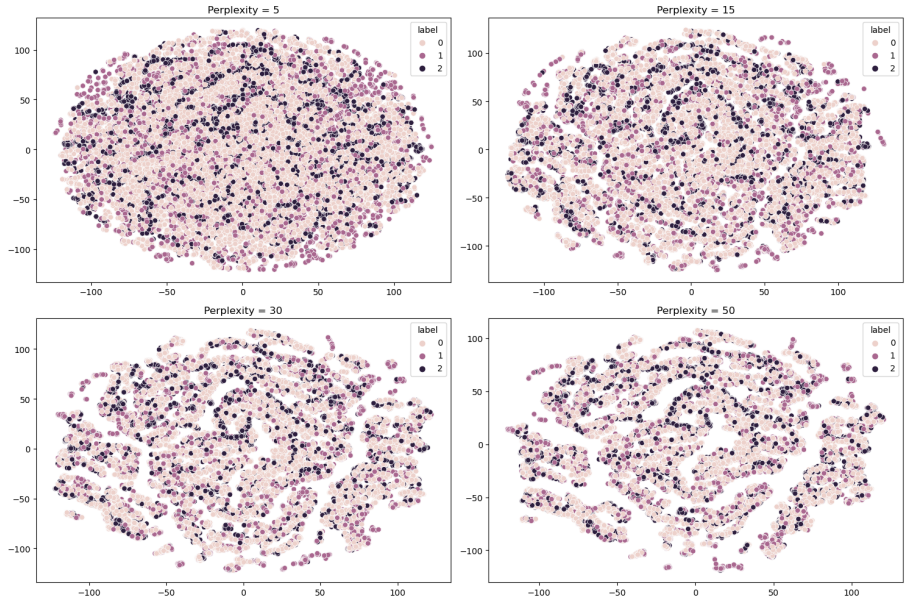

Figure 15: 2-D t-SNE with multiple perplexity parameters with different color for each label.

PCA: With this method, our aim is to identify the directions that maximize the projected inertia. To assess the number of principal axes, we refer to Figure 16 where the *elbow rule* indicates selecting only the first axis.

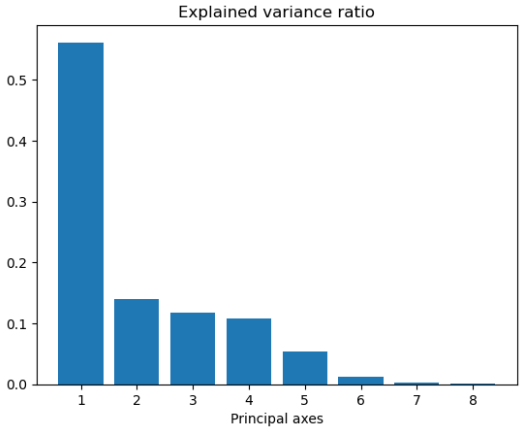

Figure 16: Barplot representing the explained variance of each principal axis.

However, since our objective is classification rather than data representation, we will retain more axes. Specifically, aiming to preserve at least 95.00% of the explained variance, we need to retain the first 5 principal axes, ensuring a retention of 98.27% of the variance. Additionally, as shown in 17, we observe improved separability within these axes, even if the data is being plotted only in different 2-D principal plane.

3.2 Learning

This section outlines the general methodology adopted for constructing the classifiers intended for this problem. The progression begins with basic algorithms like linear classifiers and culminates with more complex techniques such as bagging, boosting, or neural networks.

As previously outlined in the introductory section, the selection of hyperparameters is computed through a process of cross-validation.

3.2.1 Naive Bayes

Based on the Bayes rule, this classifier estimates the maximum a posteriori by presuming conditional independence among each pair of features, given the class variable's value. We conducted tests using both Bernoulli and Gaussian models, but they were not accurate for our data.

3.2.2 Linear methods

Linear Discriminant Analysis: Thanks to Naive Bayes, we observed that the model was more accurate in the Gaussian case than in the Bernoulli one. Following this approach, we continued with a model precisely based on the Gaussian assumption for the covariates conditionally on the label. Initially, we assumed that the variance was the same for the three classes, resulting in LDA. Later, we removed this assumption, leading to QDA, which resulted in a quadratic decision boundary instead of a linear one. This method proved to be more accurate than the previous ones.

Logistic Regression: This model assumes an underlying probability structure of belonging to a class based on the available data. It can also be viewed as a method that optimizes the logistic

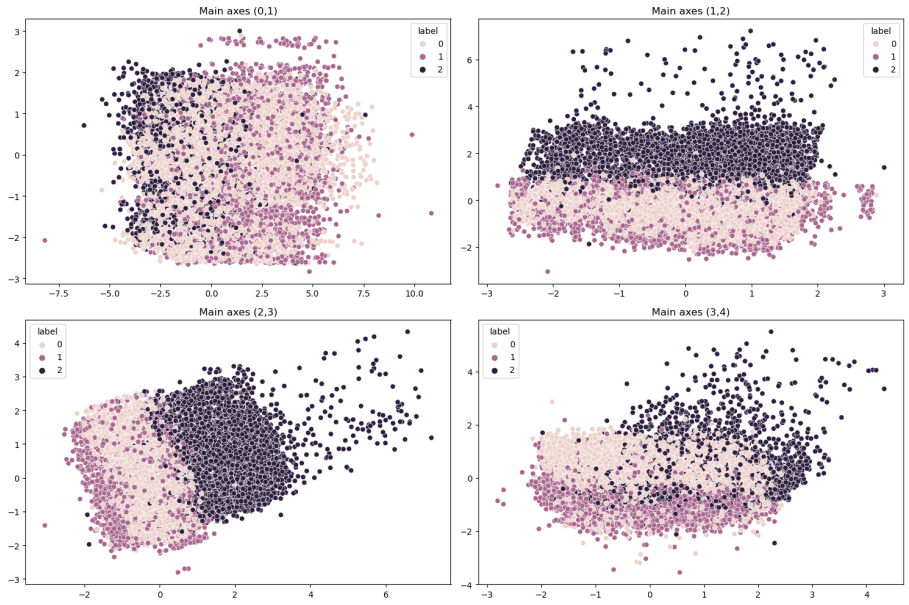

Figure 17: Data representation in different principal planes.

risk. Despite its simplicity, we observe that it yields accurate results. Additionally, we noted that a penalization term can be included to regulate the method. However, after testing multiple parameters for this regularization, we found that this term was unnecessary. This observation may stem from the implicit regularization applied through the initial PCA process.

Perceptron: This iterative geometric method does not assume a specific underlying distribution in the data but does require the assumption of linear separability. However, since this assumption is not realistic, the method stops after a certain number of iterations.

Passive-aggressive: The idea of this family of classifiers it to respond *passive* for the correct answers and *aggressive* for the incorrect ones.

3.2.3 No-linear standard methods

Nearest neighbours: This simple method relies on a continuity assumption, considering only the nearest neighbors to determine the classification of a given data point. Once the dimension has been reduced using PCA, we notice that the concept of distance becomes well-defined, mitigating the impact of the curse of dimensionality.

QDA: As previously discussed in the context of LDA, this method assumes a Gaussian distribution of the covariates conditioned on the label, this time without assuming a shared covariance across the classes.

3.2.4 SVM

In this section, we study the accuracy of the Support Vector Machine (SVM), a method that aims to construct a discriminating hyperplane that maximizes the margins between classes. Once this it is computed, the predictions will be done by taking the sign of the signed distance to the hyperplane. This hyperplane is computed by minimizing the hinge loss, which is given by

$$\Phi(X) := (1 - X)_+,$$

where $(x)_+ := \max(0, x)$. Furthermore, there is also a term that can be interpreted as a Ridge regularisation. Therefore, the SVM can be seen as the following minimisation problem:

$$\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \left(1 - Y_i(w^\top X_i + b)\right)_+,$$

so that we are penalizing the signed distance to the hyperplane defined by w and b. The regularization parameter C, works inversely to the regularization strength. A higher value of C leads to increased penalization of misclassified observations or those near the margin. Consequently, this results in a more complex construction of the margin which is less *regularized*. After some rewriting, the dual problem can be expressed as maximising

$$\max \sum_{i=1}^{n} \mu_i - \frac{1}{2}\sum_{i,j} Y_i Y_j \mu_i \mu_j X_i^\top X_j \text{ under the constraints } 0 \leq \mu_i \leq C \ \& \ \sum_{i=1}^{n} \mu_i Y_i = 0. \tag{2}$$

Moreover, we observe that the prediction function can be given by the sign of

$$\widehat{g}(X) := \sum_{i=1}^{n} \widehat{\mu}_i Y_i X_i^\top x + \widehat{b},$$

where the estimates are the result of the optimization problem. This method gives the Linear SVM. However, we observe the dependence of the prediction function and in the optimization problem on the inner product. We can significantly increase the expressiveness of the separating hyperplane by mapping our data into a higher-dimensional space using a feature mapping denoted as ϕ. This mapping can be extraordinarily complex, potentially infinite-dimensional. Surprisingly, we do not need to explicitly compute this feature mapping; our requirement is uniquely knowing how to compute the scalar product $\langle \phi(X), \phi(X') \rangle$ for any $X, X' \in \mathcal{X}$. Introducing K as the kernel defined by $K(X, X') := \langle \phi(X), \phi(X') \rangle$, the existence of the Reproducing Kernel Hilbert Space (RKHS) is assured if K is symmetric and semi-definite positive.

Thus, we will not explicitly provide the feature mapping used, for instance, in the polynomial kernel, as it is not necessary to compute it explicitly. However, what is crucial is understanding that the polynomial kernel is computed as

$$K_{\text{poly}}(x, y) := (\gamma \langle x, y \rangle + c)^d,$$

where d represents the degree and c and γ are coefficients controlling the significance of higher-order versus lower-order polynomials. In the Scikit-learn library, the value of the coefficient c can be adjusted using the parameter *coef0*, the value of d can be modified using the parameter *degree*, and the value of γ by *gamma*.

In our dataset, we have mostly tried multiple degrees for our polynomial SVM.

Moreover, we have also tried other different kernels such as the Gaussian one, which is usually referred as *Radial Basis Function*(RBF) and which is computed by

$$K_{\text{RBF}}(x, y) := \exp\left(-\gamma \|x - y\|^2\right),$$

where the γ can be seen as the inverse of the $2\sigma^2$ of a Gaussian distribution, so as the γ increases, the variance decreases, causing the kernel to yield smaller values. This parameter can be tuned by the parameter *gamma* in Scikit-learn.

Finally, another kernel denoted as Sigmoid is given by

$$K_{\text{sigm}}(x, y) := \tanh\left(\langle x, y \rangle + r\right),$$

where α represents the parameter *alpha* and r the parameter *coef0*.

3.2.5 Random Forest

Decision tree: This method is essentially similar to the decision tree detailed in 2.2.5, but adapted to the classification problem instead of regression. In contrast to making predictions based on the mean of observations in the leaf, it takes the most frequently occurring value. Apart from this distinction, it maintains the same parameters for controlling method flexibility, such as setting the minimum samples required to split a leaf or the maximum allowable depth.

Additionally, it shares the same significant variance error, which can be mitigated by employing a bagging approach using Random Forests.

Random Forest: Exactly as with the Random Forest in the regression context, we construct multiple decision trees, aiming to make them as uncorrelated as possible. This is achieved, for instance, by diversifying the training set using bootstrapping or by selecting a random subset of covariates allowed at each tree split. However, in our specific scenario, the latter method of introducing diversity might not be as effective, given that the number of covariates is only 5 due to dimension reduction.

Therefore, we also seek to enhance the method by fine-tuning multiple parameters, such as the number of estimators or the hyperparameters inherited by the decision tree, like the maximum depth.

3.2.6 Neural Networks

In this section, we elaborate on the various parameters we have tested for the neural network constructed using the Multiple Layer Perceptron method, already implemented in Pedregosa et al. (2011). We have mainly tested the layer size and the number of layers, the activation functions, the optimizer, and the learning rate.

3.2.7 Boosting

This technique is similar to the one detailed in Section 2.2.6, but adapting the loss functions to this classification context. Using this aggregation technique, rather than aggregating predictions at the end made by predictors trained simultaneously and independently from the rest as done in the bagging, we construct base-learners iteratively. These learners progressively learn from the errors of the preceding steps. As a result, the number of iterations corresponds to the number of base-learners to be trained.

Specifically, employing the exponential loss leads to the standard Adaboost. However, we have observed higher accuracy using the log loss.

We have not uniquely used the boosting techniques provided in Pedregosa et al. (2011). Instead, we have also experimented with other efficient libraries: XGBoost and CatBoost (see Chen and Guestrin (2016);Prokhorenkova et al. (2018)).

3.2.8 Meta-experts aggregation

The concept behind this method is the same of the one that was previously detailed in Section 2.2.8, adapted for the multiclass classification context. Specifically, the mean approach simplifies to a mode prediction, while the accuracy-weighted mean approach also assigns varying influence to predictions based on classifier accuracy.

Furthermore, the residuals correction approach incorporates a stacking mechanism involving memory integration. Looking for the diversity, we introduced raw data as memory. This addition predates the preprocessed data obtained through PCA and projection onto the first 5 principal axes.

3.3 Semi-supervised: Self-training

In semi-supervised learning, instead of working with a single dataset $\mathcal{D}_n := \{(X_i, Y_i)_{i\in\{1,\dots n\}}\}$ containing labeled data, we also exploit the underlying structure of an unlabeled set $\mathcal{U}_m := \{(X_{i+n})_{i\in\{1,\dots m\}}\}$.

In this context, two primary perspectives emerge: the *inductive* perspective, which aims to leverage both $\mathcal{D}_n$ and $\mathcal{U}_m$ to enhance predictive capabilities for new, independent observations X. In contrast, the *transductive* perspective seeks to improve predictions specifically for the unlabeled dataset $\mathcal{U}_m$.

In our project, we adopt a transductive perspective, treating the training set as our labeled data $\mathcal{D}_n$ and the test set as the unlabeled data $\mathcal{U}_m$. This approach assumes that the entire set we aim to predict is available from the outset. It is worth noting that this assumption might not hold in other scenarios, such as temporal series.

Self-training: This method iteratively trains a base-learner on the labeled data extended by the unlabeled data which was predicted with at least a high confidence level τ in the previous step. We denote respectively $\mathcal{D}^t$, $\mathcal{U}^t$ and $\widehat{h}^t := \widehat{h}\left(\mathcal{D}^t\right)$ the training set, the unlabeled set and the base-learner at the t-step. We denote by $\mathbb{P}_h\left(Y_i = k\right)$ the probability of the i-th observation belonging to the class k according to the classifier h. We also denote by $\mathcal{S}^t \subset \{n, \dots, n+m\}$ the set of the indices of the initial unlabeled data that has been predicted with high probability with the $t-1$ classifier.

This greedy algorithm start by training the initial predictor $\widehat{h}^1$ over the training set $\mathcal{D}_n$. Therefore, we initialise $\mathcal{D}^1 = \mathcal{D}_n$, $\mathcal{U}^1 = \mathcal{U}_m$ and $\mathcal{S}^1 = \emptyset$. Iteratively, at the t-step, it predicts the probabilities of the unlabeled data $\mathcal{U}^{t-1}$ using the classifier $\widehat{h}^{t-1}$. Then, it computes the set $\mathcal{S}^t = \left\{i \in \mathcal{U}^{t-1} \middle| \mathbb{P}_{\widehat{h}^{t-1}}\left(Y_i = k\right) > \tau\right\}$ of unlabeled indices such that the probability of belonging to a class k for any $k \in \{1 \dots, K\}$ is higher than a established threshold τ. The new training set is constructed by adding to the previous training set this new set $\mathcal{D}^t = \mathcal{D}^{t-1} \cup \mathcal{S}^t$.

We notice that to achieve a strong sense of membership confidence, it is necessary to ensure a high probability of being assigned to a class when the classifier makes its prediction. Moreover, the classifier must be capable of calculating membership probabilities. This intuitive idea is formalized by the *calibration*.

Calibration: Well-calibrated learners are those in which the output of the class belonging probability can be directly interpreted as a confidence level.

On binary classification, calibration curves compare how well the probabilistic predictions are calibrated. It is done by plotting the probability of belonging to the class 1 conditioned on the

predicted probability. This method could be generalized for our 3-class classification by using a One-vs-Rest strategy. Other methods based on cross-validation are also available.

For instance, Logistic Regression returns well-calibrated predictions thanks to its *balance property* as referred to in Wuthrich and Merz (2022). This is due to its loss function.

On the other hand, as shown by Niculescu-Mizil and Caruana (2005), methods such as SVM or boosted trees tend to push probabilities away from 0 and 1. Furthermore, they also show that methods such as neural networks or bagged trees do not exhibit this bias and predict the true posterior probability accurately.

Therefore, considering these observations, we will establish varying confidence level thresholds based on the calibration of the base-learner. For instance, we may require a higher probability threshold for Logistic Regression compared to other methods.

Other calibration methods, like *Platt calibration* or *Isotonic Regression*, are presented in Niculescu-Mizil and Caruana (2005). However, we have chosen to adopt this simple approach. The authors explain that these methods are designed for binary classification and do not easily extend to our multiclass case.

Moreover, following the same perspective as in section 3.2.8, we aim to enhance the robustness of our self-training approach by leveraging the consensus of multiple classifiers. This helps to avoid the typical cascade effect associated with the method. We will introduce our version of aggregated self-training.

***Robust* self-training:** To accommodate multiple base-learners, we will modify the standard self-training algorithm. Only unlabeled observations whose class membership has been verified with high confidence by all the classifiers will be labeled.

More formally, keeping the same meaning as before for $\mathcal{D}^t$ and $\mathcal{U}^t$, we introduce a set

$$\mathcal{H} := \left\{ \widehat{h} : \bigcup_{i=1}^{\infty} (\mathcal{X}_i, \mathcal{Y}_i) \to \mathcal{F} \right\}$$

of learning rules. For each step t we will train all the learning rules $\mathcal{H}$ on the t-training set $\mathcal{D}^t$, giving a set of predictors denoted as

$$\widehat{\mathcal{H}}^t := \left\{ \widehat{h}\left(\mathcal{D}^t\right) \in \mathcal{F} \middle| \widehat{h} \in \mathcal{H} \right\} = \left\{ \widehat{h}^t \in \mathcal{F} \middle| \widehat{h} \in \mathcal{H} \right\}.$$

Based on the previously discussed calibration properties, we will use different confidence thresholds $\tau_{\widehat{h}}$ for each associated learning rule $\widehat{h}$. Finally, in order to adapt the notation of the unlabeled indices that are assigned with high probability to a class $k \in \{1, \ldots, K\}$ by a learning rule $\widehat{h} \in \mathcal{H}$ at the t-step, we introduce:

$$\mathcal{S}^t_{k,\widehat{h}} := \left\{ i \in \mathcal{U}^{t-1} \middle| \mathbb{P}_{\widehat{h}^{t-1}}\left(Y_i = k\right) > \tau_{\widehat{h}} \right\}.$$

Thereby, the indices in which we are highly confident will be the ones for which all the predictors agree on the same class with high probability, which is the same as

$$\mathcal{S}^t := \bigcup_{k \in [K]} \bigcap_{\widehat{h} \in \mathcal{H}} \mathcal{S}^t_{k,\widehat{h}}.$$

Hence, at the following step, we update $\mathcal{D}^t = \mathcal{D}^{t-1} \cup S^t$.

Stopping criterion: As usual, we stop the algorithm after a limited number of iterations or when we can no longer label more data in the unlabeled dataset.

Final predictor: When the stopping criterion is met, the base predictors are trained using the confident set of observations. This set comprises the extended initial values along with observations predicted with high probability. To classify the observations that were not *confident*, a methodology similar to the aggregations discussed in Section 2.2.8 is employed. Specifically, it involves summing the predicted probabilities for each class and selecting the maximum probability as the solution. We notice that other approaches more elaborated could have been selected, as the one of a weighted mean or even a meta-aggregator.

3.4 Discussion

Similarly to Section 2.3, this section primarily focuses on discussing the limitations of the choices made since the beginning of this data challenge, as well as exploring potential alternatives for future work.

Robust Self-Training: We observe that the main goal of this approach was to enhance confidence levels through an aggregation method, thereby making the probability of membership more reliable. However, as shown in the contingency table in Figure 18, the results do not align with our expectations. In fact, we can see that the first class is disproportionately represented in the predictions of the other classes. This is attributed to a cascade effect, where errors from previous layers propagate into subsequent predictions. As demonstrated in Figure 13, class 0 has a significantly larger count than the others, leading to a more pronounced influence in the propagation process.

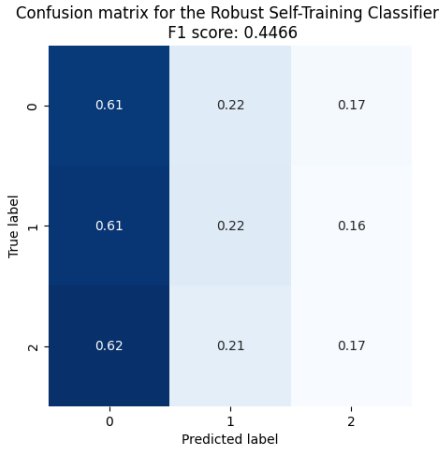

Figure 18: Contingency table of a robust self-training using an aggregation of Random Forest and Logistic Regression. We observe a cascade effect.

To understand the origin of these errors, our initial step was to identify which of the models introduced bias in the aggregation. Figure 19 demonstrates that the Self-Training version of the Logistic Regression produced poorer results compared to the initial one, which obtained over 95% F1-score.

Nevertheless, when comparing the outcomes of a basic Random Forest model (refer to Figure 20) with those obtained from a self-training variant of this model (see Figure 21), the benefits are obvious. Hence, future work could focus on exploring how to leverage both techniques: **Model Aggregation** and **Self-Training**.

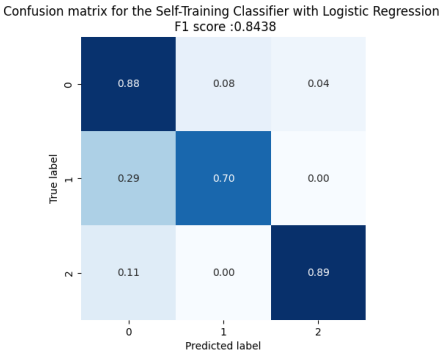

Figure 19: Contingency table of a robust self-training using a Logistic Regression.

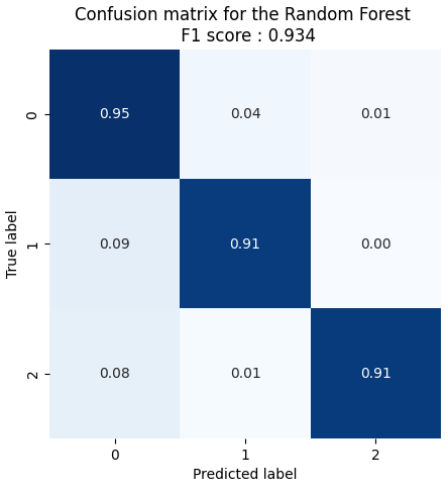

Figure 20: Contingency table of a simple Random Forest model.

Variable importance: In this scenario, direct computation of variable importance, through methods like Decision Trees or Random Forest as previously demonstrated in Appendix A.2, is not feasible as they are applied on transformed covariates and not on the *pure* ones. Nevertheless, alternative model-agnostic approaches, as discussed at the end of Appendix A.2 could be considered.

References

Breiman, L. (2001). Random forests. *Machine Learning*, 45:5–32.

Chamma, A., Engemann, D. A., and Thirion, B. (2023). Statistically valid variable importance assessment through conditional permutations.

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. *CoRR*, abs/1603.02754.

daniel servén, Brummitt, C., Abedi, H., and hlink (2018). dswah/pygam: v0.8.0.

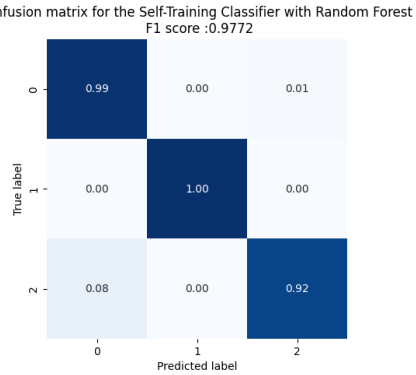

Figure 21: Contingency table of a robust self-training using a Random Forest.

Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. (2004). Least angle regression. *The Annals of Statistics*, 32(2):407 – 499.

Felix Biggs, A. S. and Gretton, A. (2023). MMD-FUSE: Learning and combining kernels for two-sample testing without data splitting.

Giraud, C. (2021). *Introduction to high-dimensional statistics*. CRC Press.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. (2012a). A kernel two-sample test. *J. Mach. Learn. Res.*, 13(1):723–773.

Gretton, A., Sriperumbudur, B., Sejdinovic, D., Strathmann, H., Balakrishnan, S., Pontil, M., and Kenji, F. (2012b). Optimal kernel choice for large-scale two-sample tests. volume 25.

Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9:1735–80.

Kratsios, A. and Papon, L. (2022). Universal approximation theorems for differentiable geometric deep learning. *Journal of Machine Learning Research*, 23(196):1–73.

Niculescu-Mizil, A. and Caruana, R. (2005). Predicting good probabilities with supervised learning. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, page 625–632, New York, NY, USA. Association for Computing Machinery.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., and Gulin, A. (2018). Catboost: Unbiased boosting with categorical features. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 6639–6649, Red Hook, NY, USA. Curran Associates Inc.

Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, 5(2):241–259.

Wuthrich, M. V. and Merz, M. (2022). *Statistical Foundations of Actuarial Learning and its Applications*. Springer Actuarial, Open Access. Available at SSRN: https://ssrn.com/abstract=3822407 or http://dx.doi.org/10.2139/ssrn.3822407.

A Wine quality prediction

A.1 Data exploration

Figure 22 represents the relationship between covariates. It allows us to distinguish different patterns for each wine type. For instance, when comparing citric acid against fixed acidity, the orange points representing white wine (type 1) demonstrate a slope closer to the identity line, while the blue points, indicating red wine (type 0), align parallel to the citric acid axis. Additionally, when observing pH against fixed acidity, the blue points appear to form a linear pattern, whereas the orange points exhibit a more quadratic relationship.

Moreover, our observation of the last row, where we plot the covariates against the target, highlights the significant shifts between both types. This observation could have an impact on our predictive task.

A.1.1 Red wine data exploration

In Figure 23, We observe numerous distinct patterns in the relationships among the covariates, color-shaded by the labeling. These observations enable us to identify interactions between the covariates, such as the linear decrease of *alcohol* with *density*. Such insights could be particularly valuable, for instance, in the application of the GAM model(see Section 2.2.3).

A.1.2 White wine data exploration

Similarly to the previous case, we can observe multiple distinct patterns among the covariates in Figure 24. These observations enable us to identify interactions between the covariates, such as the linear increase of *fixed acidity* with *density* or the quadratic relation of *pH* with *fixed acidity*.

A.2 Variable importance of wine for quality prediction

The aim of this section is to explore the significance of covariates for the predictive objective, using multiple methods to assess if they have used all the available information. Some models, like Decision Trees, offer high transparency and directly return variable importance. However, several models, such as neural networks, lack direct interpretability. To address this, various model-agnostic methods for variable importance have been proposed, such as permutation important. This method involves evaluating the change in prediction accuracy after disabling a variable through a simple permutation. However, this method might not provide reliable information, particularly in cases with correlated covariates. Addressing this limitation, Chamma et al. (2023) introduced conditional importance permutation, which specifically studies the relationship between other covariates while shuffling only the added information, thereby providing more nuanced insights.

A.2.1 Decision tree variable importance

Figure 25 reveals that a substantial portion of importance is concentrated on a specific covariate, while several other covariates are underused. This suggests that in order to enhance performance and use all available information, it is essential to reduce the influence of this dominant variable at certain splits, allowing other covariates to contribute. This adjustment can be accomplished using Random Forest, as it employs only a random subset of covariates at each split, potentially excluding the *alcohol* covariate.

A.2.2 RF variable importance

From Figures 26 and 27, the prior issue of a single dominant covariate has been addressed, and the weight is more evenly distributed, although the *alcohol* covariate still remains the most significant. We note that the variable importance is computed using Mean Decrease Impurity (MDI), an impurity-based feature importance method that might show bias with categorical covariates possessing a large cardinality. However, in our case, this issue does not arise.

A.3 Results

Table 1: Results for wine regression

Model	Wine	Score	Filtered Score	Comment
Linear	red	0.273	0.269	
	white	0.339	0.313	
Lasso LARS	red	0.305	0.343	
	white	0.347	0.259	
Lasso AIC	red	0.312	0.343	
	white	0.348	0.226	
Lasso BIC	red	0.306	0.328	
	white	0.350	0.345	
GAM	red	0.422	0.344	
	white	0.501	0.283	
SVR	red	0.389	0.374	
	white	0.379	0.347	
Decision Tree	red	0.299	0.278	
	white	0.302	0.253	
Random Forest	red	0.503	0.438	
	white	0.464	0.366	
XGBoost	red	0.463	0.416	
	white	0.399	0.373	
CatBoost	red	0.498	0.459	
	white	0.445	0.412	
MPL	red	0.389	0.291	
	white	0.331	0.294	
Weighted Average	red	0.464/0.473	0.418/0.421	all models/RF+XGB+Cat
	white	0.427/0.429	0.367/0.382	
Meta Model	red	0.449/0.441	0.403/0.402	Models : All/RF+XGB+Cat — MetaModel : RF
	white	0.393/0.385	0.324/0.331	

B Celestial objects prediction

B.1 Data exploration

We have started with the representation of the boxplot as shown in Figure 28, from which we can assess an initial *discriminatory* power for each variable. Then we have also represented pairwise

combination of covariates in Figure 29.

From Figure 28, we can observe for example, the *redshift* variable appears to offer substantial information, given the distinctly varied distributions for the three labels. Label 1 shows concentrated values, while Label 2 spans a wide range of possible values. However, this one-dimensional view does not consider relationships between the variables. To do so, Figure 29 allows us to observe correlations, such as the nearly linear relationships between variables like u, g, r, and i. Additionally, we can discern the distinctiveness of labels based on combinations involving the redshift variable.

B.2 Results

Table 2: Results for celestial object classification

Model	F1-Score	Comment
BernoulliNB	0.682	
GaussianNB	0.817	
LinearDiscriminantAnalysis	0.816	
LogisticRegression	0.951	
LogisticRegression	0.957	No Regularized
Perceptron	0.917	
PassiveAggressive	0.901	
KNeighbors	0.923	
QuadraticDiscriminantAnalysis	0.946	
LinearSVC	0.958	
SVC	0.958	Polynomial Kernel
SVC	0.966	RBF Kernel
DecisionTree	0.905	
RandomForest	0.934	
MLP	0.966	
GradientBoosting	0.942	
XGB	0.941	
CatBoost	0.953	
WeigthAggregation	0.951	
MLPAggregation	0.953	
RFAggregation	0.939	
RobustSelfTrainingClassifior	0.435	Logistic + RF + Cat

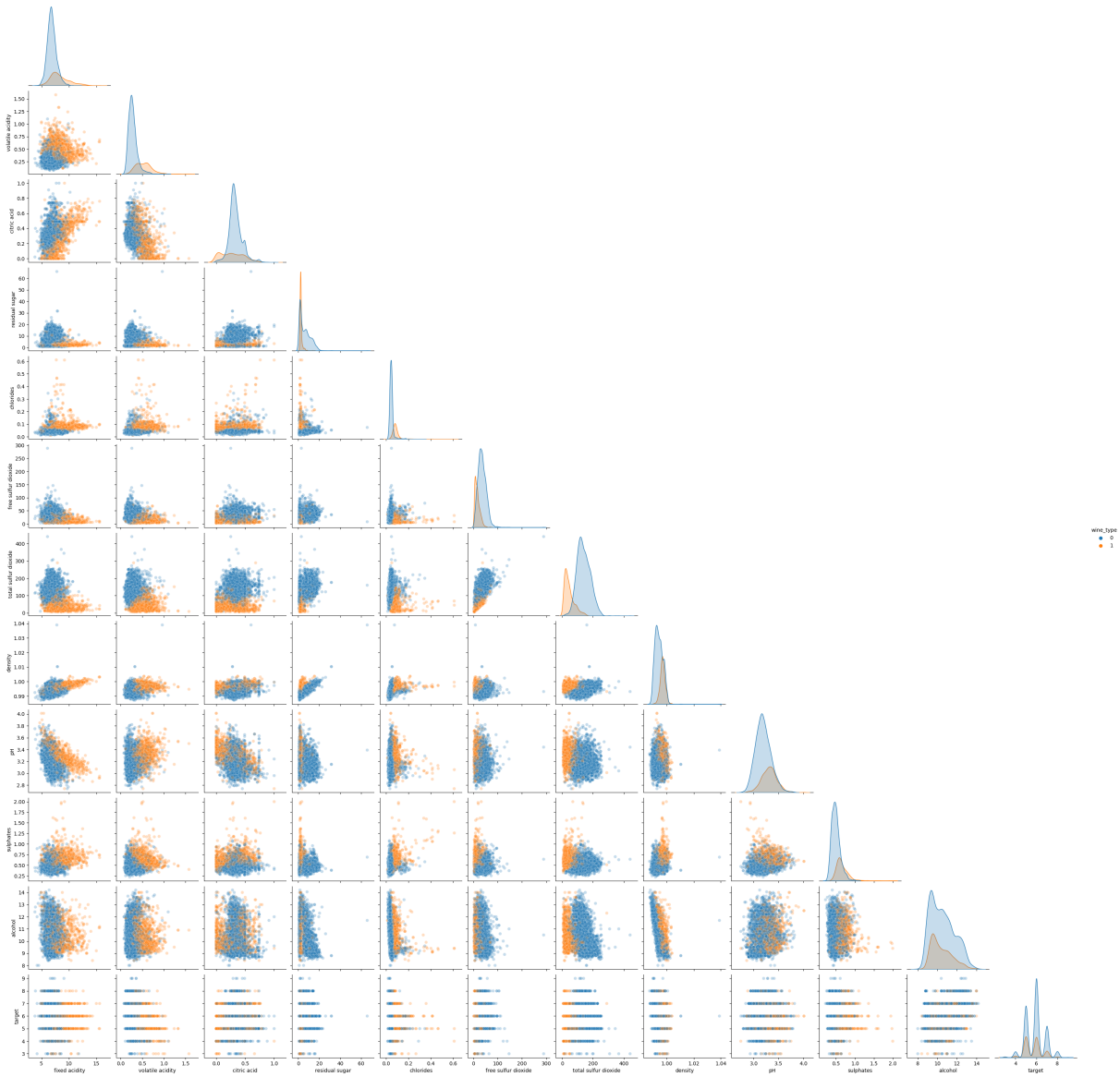

Figure 22: Plotting the relationship between covariates, color-coded by wine type, reveals various distinct patterns based on the type of wine.

Figure 23: Plotting the relationship between covariates for red wine, color-coded by wine quality, reveals diverse patterns depending on the specific covariates.

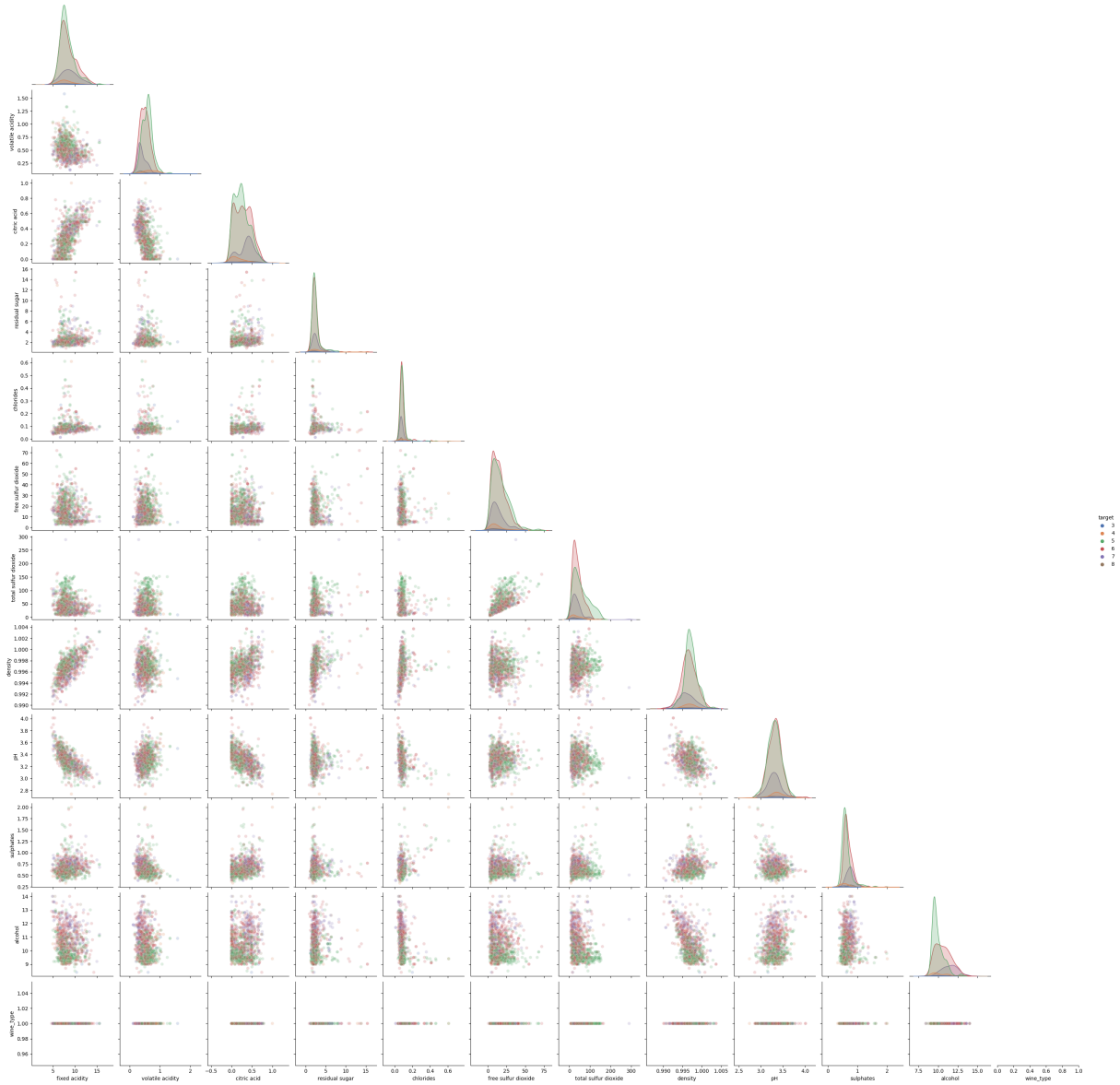

Figure 24: Plotting the relationship between covariates for white wine, color-coded by wine quality, reveals diverse patterns depending on the specific covariates.

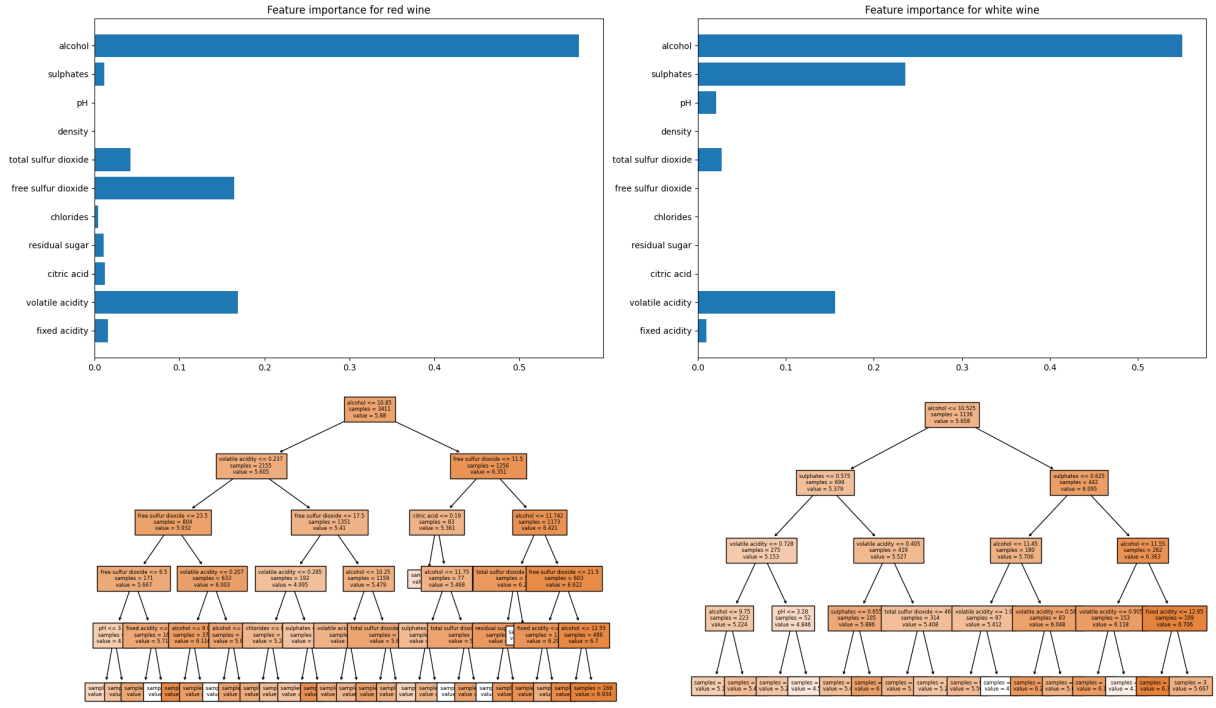

Figure 25: Variable importance and represention of the Decision Trees for each wine type.

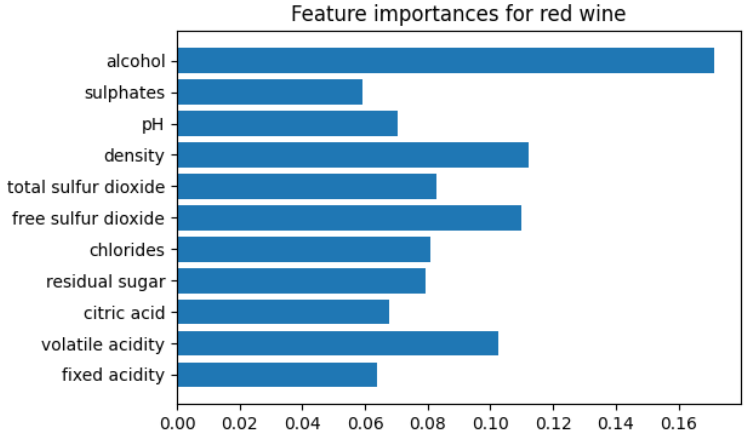

Figure 26: Variable importance of the RF for the red wine.

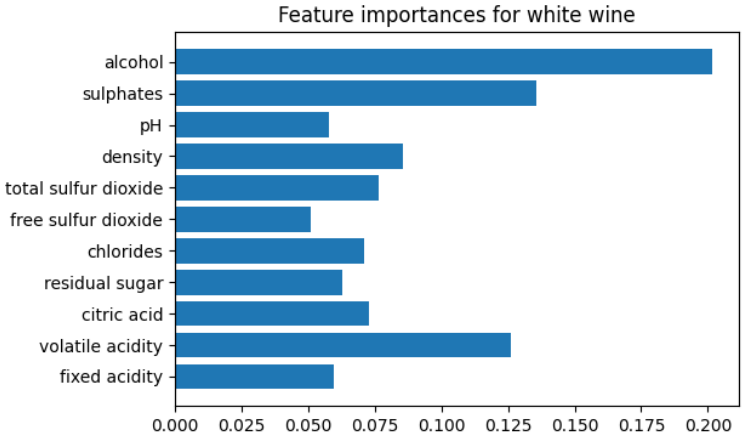

Figure 27: Variable importance of the RF for the white wine.

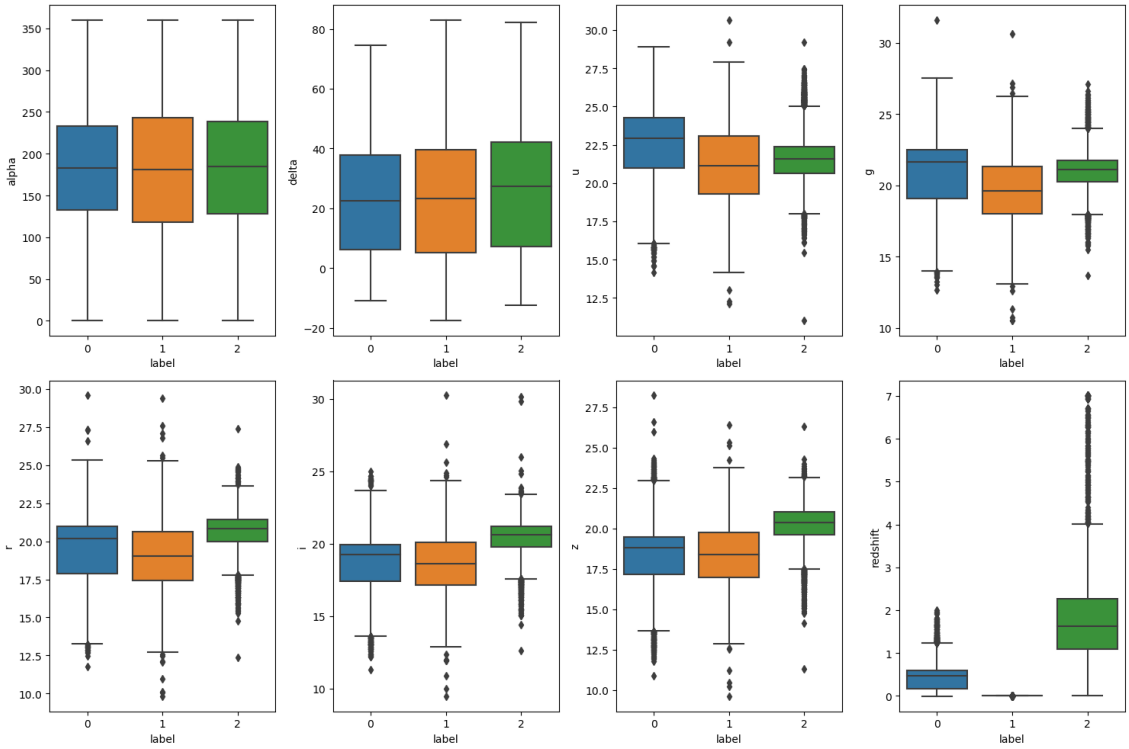

Figure 28: Boxplot of the initial variables based on the labels provides an initial assessment of each variable's *discriminatory* power.

Figure 29: Pairwise combinations of covariates are displayed with different colors for each observation label.