University of Mons

Faculty of Sciences

Machine Learning Proyect

Authors:

Carlos María del Pino

Lamlih Houssam

May 2023

1 Exploratory Data Analysis

Exploratory Data Analysis (EDA) plays a pivotal role in any data research or analysis project, as it allows us to gain initial insights into the structure and characteristics of the data we are working with. The primary objective of our study was to examine the distribution of each variable within our Data Frame. In this context, we employed various statistical techniques and visualizations to delve into the distribution of variables and gain a comprehensive understanding of their behavior. Histograms, scatter plots, box plots, among others, were utilized to visualize and comprehend the distribution patterns present in each variable.

In order to gain an initial understanding of the dataset, we commenced by visualizing the distribution of data across different categories. By examining the count or frequency of observations in each category, we were able to identify any potential data imbalances or biases that might influence subsequent analyses.

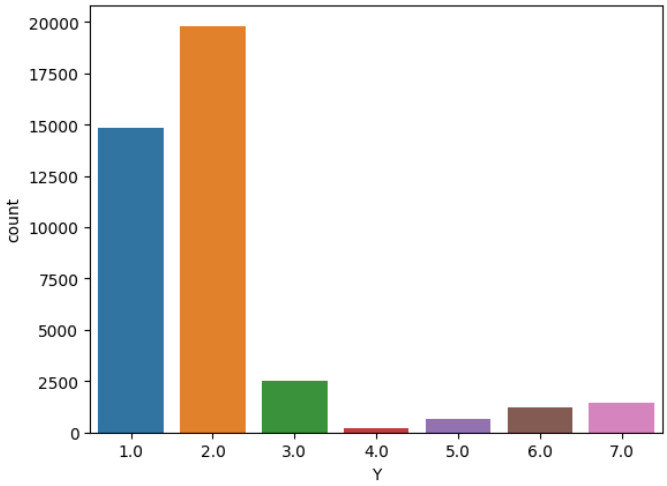

Figure 1: Data

It was observed (Fig. 1) that the vast majority of the data fell into categories $Y = 1$ and $Y = 2$. This finding holds significant importance and necessitates careful consideration in our analysis.

As we can observe in Figure 2, the bloxpot represents the numerical features and how they are distributed, with their anomalous data points. We can conclude from it that features such as X1, X3, X6 or X10 are going to be suitable for using for predicting the value of Y. On the contrary, feaatures like X5 give us no relevant information about which value of Y could take an input. Moreover, features like X2 or X9, which have a wide range of values, were not adequate to predict the label.

After deriving insights from the box plot analysis, we proceeded to investigate the potential relationships between select variables using scatter plots. This step aimed to explore any associations or patterns that could exist among the variables of interest.

As shown in Figure 3, we conclude there was a positive correlation between variables X1 and X6, but it does not seem to be extremely strong. Additionally, X1 and X3 have no relation amog them. We also study another feautres, ending into a positive relation between, for instance, among X2 and X9.

Here there ir graph depicting the correlation coefficients among all the numerical variables. While it may not be the most intuitive approach, this graph provides a clear visualization of the relationships between the variables.

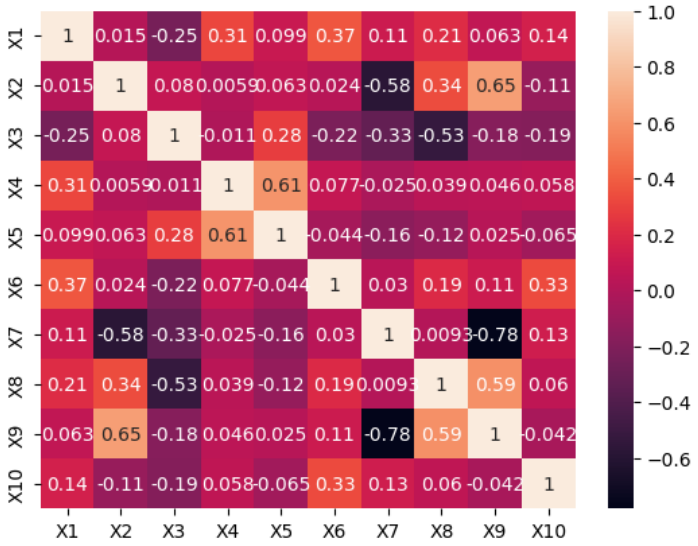

Figure 2: Data

2 Methodology

In this section, you can describe the methodology used in your study.

K-Nearest Neighbors (KNN)

First, let's study the KNN learning model. Initially, we encountered a challenge as we lacked test data to evaluate the prediction performance. To overcome this, we made the decision to split the training set into two separate sets, allowing us to simulate a test set, as we have learned during the course. Nevertheless, this conclusion was derived from a comprehensive analysis of the optimization results on the entire training dataset. It was realized through multiple iterations of visualizing the outcomes, as we observed the metrics' performance on the same set—a methodology that lacked logical coherence, as we were overfitting the model. In reltion to the model, we examined the impact of choosing the appropriate value of 'k', the number of nearest neighbors considered during the prediction process, which was taken into account over the other parametres.

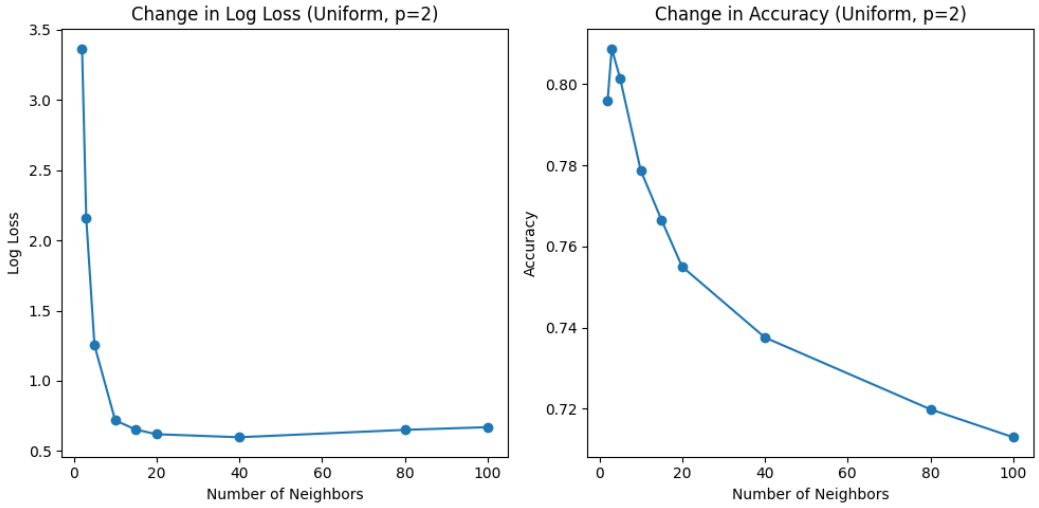

Figure 3: Log Loss and Accuracy in predictions

Upon careful examination of the data presented in Figure 3, we have arrived at the determination to establish our algorithm with a value of k=20. The observed decrease in accuracy was marginal in comparison to the considerable improvement achieved for the LogLoss metric. Additionally, as we progressively increased the number of neighbors in the algorithm, we observed through the confusion matrix a confirmation of our initial assumption derived from the data analysis: the insufficient number of instances classified as class $y = 4$ and $y = 5$ indicated a lack of representation for these particular classes in our predictions.

However, although this LogLoss (≈ 0.65) score was not unfavorable, we deemed it necessary to explore alternative algorithms to further improve our results.

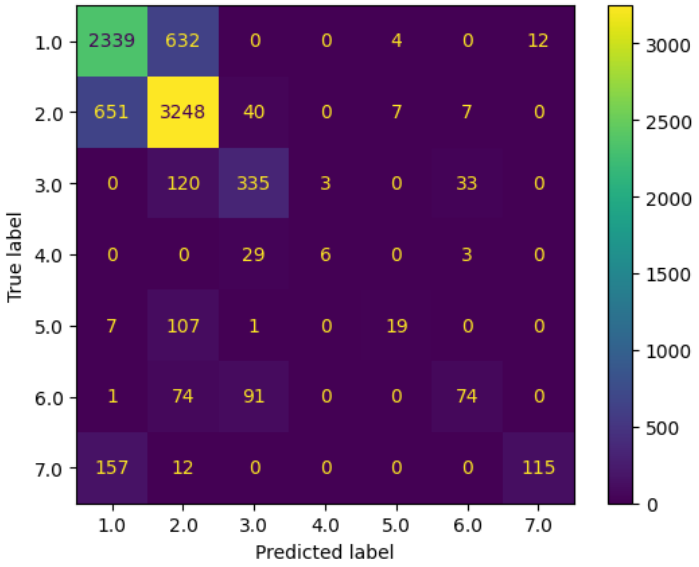

Figure 4: Confusion Matrix pour $n = 20$

Logistic Regression (LR)

In the implementation of the LogisticRegression algorithm, we employed the RandomizedSearchCV class to iterate over hyperparameters and obtain the most optimal results. This technique allowed us to systematically explore different combinations of hyperparameters within predefined ranges. By leveraging random sampling, RandomizedSearchCV efficiently traverses the hyperparameter space, evaluating each configuration's performance using a specified evaluation metric. This iterative process enabled us to identify the hyperparameter values that yielded the best results in terms of our evaluation metric, ultimately enhancing the performance and accuracy of the LogisticRegression algorithm. After its execution was completed, we obtained the best parameters for the method, which are as follows:

Best hyperparameters: 'penalty': 'l2', 'fit_intercept': True, 'C': 0.01

Despite having the best hyperparameters, we found that the LogLoss coefficient did not meet our expectations. Even after introducing new variables into the Logistic Regression model, the improvement in performance was not significant. Faced with this situation, we decided to iterate over the parameter 'C' of Logistic Regression to observe how certain metrics evolved with different values.

However, after a certain threshold, further increases in 'C' led to a gradual increase in the LogLoss coefficient, suggesting overfitting and a decline in predictive power.

Despite making adjustments to the 'C' parameter, we observed that the metrics did not improve significantly. Despite our efforts to fine-tune this parameter, it became evident that manipulating it further would not lead to the desired outcome.

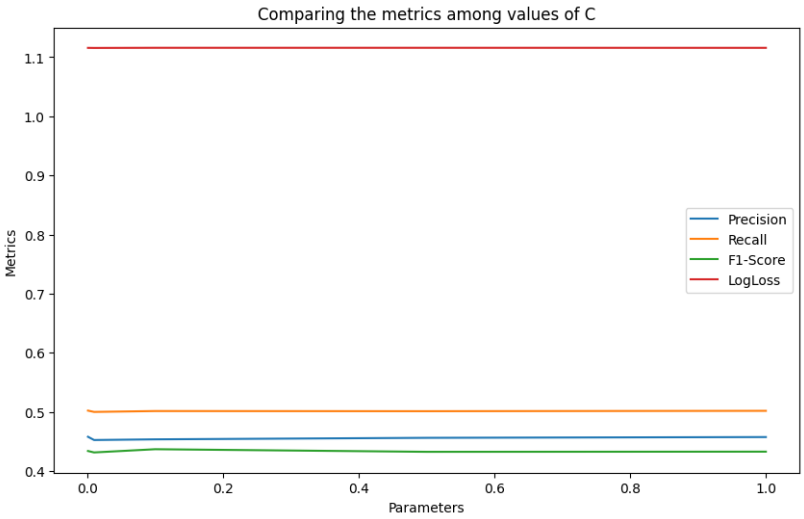

Figure 5:

Upon analyzing the confusion matrices, we observed that the model's predictions for classes $y = 3, y = 4, y = 5, and y = 6$ were consistently low. This finding is of significant relevance as it indicates that the model struggles to accurately predict instances belonging to these particular classes.

Gradient Boosting

In this section of our project, we delve into the analysis of the Gradient Boosting algorithm and explore the impact of the *n_estimators* parameter on its performance. Gradient Boosting is a powerful machine learning technique known for its ability to create strong predictive models by combining multiple weak learners.

Through an iterative process, we will systematically adjust the *n_estimators* parameter, training and evaluating the Gradient Boosting model for different values. This will enable us to observe how the model's performance changes as we increase or decrease the number of estimators.

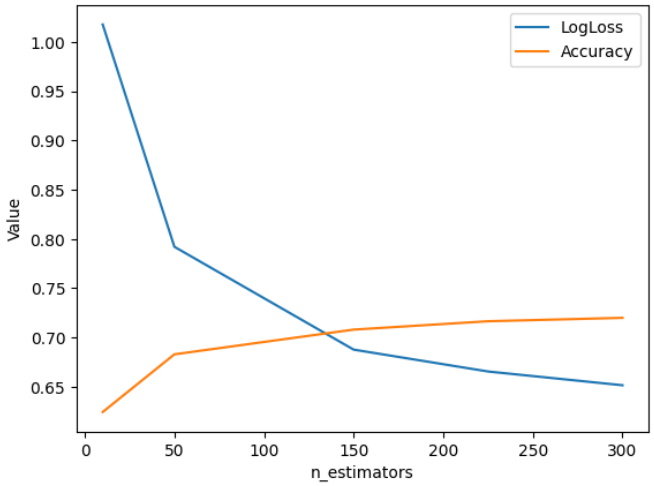

Figure 6: Effect of *n_stimators* on metrics

As we increase the number of estimators in the Gradient Boosting algorithm, we observe a notable improvement in both the LogLoss and Accuracy metrics. This positive trend indicates that as the model iteratively learns and refines its predictions, it becomes more accurate and better at minimizing the logarithmic loss.

After comparing the number of estimators in our model, we proceeded to perform a similar analysis on the learning rate to observe its impact on the evolution of the log loss and accuracy.

The learning rate plays a crucial role in gradient boosting algorithms as it determines the contribution of each tree to the final prediction. A lower learning rate leads to slower convergence but can potentially improve the model's generalization capability. Conversely, a higher learning rate allows for faster convergence but may result in overfitting.

Upon analyzing the impact of different learning rates on our model, we observed that a learning rate between 0.1 and 0.2 yielded optimal results, considering that we had set the number of estimators to 200. Notably, as the learning rate approached this range, the log loss gradually decreased, indicating improved model performance. It was particularly encouraging to see the log loss approaching 0.5, as this suggested a higher level of confidence in our predictions.

After conducting all the steps mentioned above, we proceeded to compute the outcomes obtained

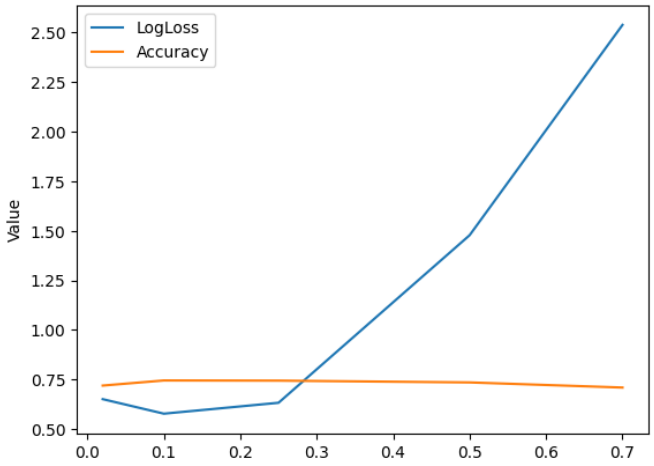

Figure 7: Effect of the *learning rate* parameter on metrics

with the optimal parameters. This approach was undertaken prior to employing the RandomizedSearchCV function, as its execution time was considerably lengthy owing to the algorithm's complexity. Moreover, adopting this method enabled us to gain a clearer understanding of the results. Ultimately, we draw the following conclusions from our analysis, along with the associated confusion matrix: unlike other models in which certain classes were underrepresented due to a minority of examples, this model successfully addressed this issue.

```
Test log-loss : 0.5780918644696508
Accuracy: 0.7455395594930478
```

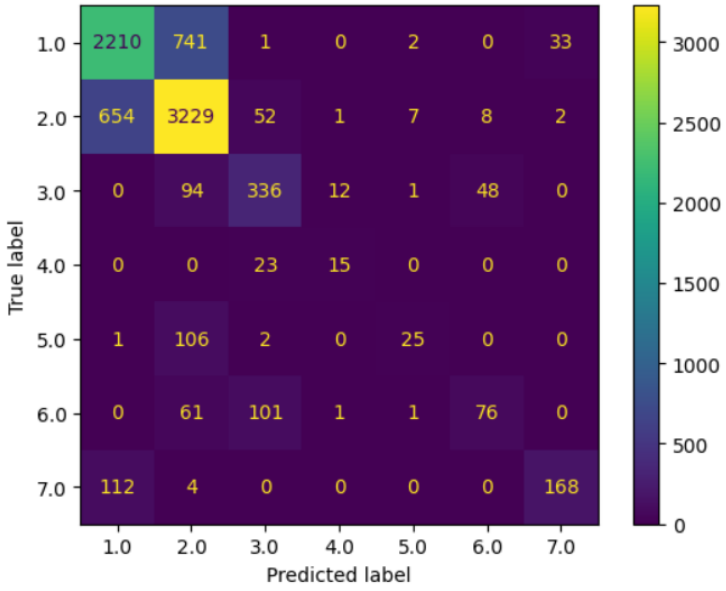

Figure 8: Effect of the *learning rate* parameter on metrics

[Texto relacionado con Decision Trees]

Naive Bayes

In this section, we will delve into the study of the Naive Bayes algorithm. Naive Bayes is a popular classification algorithm based on Bayes' theorem, which assumes that the features are conditionally independent given the class label. Despite this simplification, Naive Bayes has shown strong performance in various applications and is particularly useful when dealing with high-dimensional datasets. We will explore the underlying principles, assumptions, and implementation of Naive Bayes, and analyze its strengths and limitations in different contexts.

Considering the algorithm's efficient processing speed, we extended the number of variables used to assess its performance across varying quantities. This approach aimed to address the challenges encountered with previous algorithms, primarily stemming from their prolonged execution times, which hindered comprehensive exploration of variable dependencies. As for Naive Bayes, we attempted parameter optimization using the RandomizedSearchCV class. However, the results yielded suboptimal metrics, necessitating further adjustments to improve performance. Here, we present the outcomes obtained after meticulous manipulation of the variables to enhance the algorithm's effectiveness.

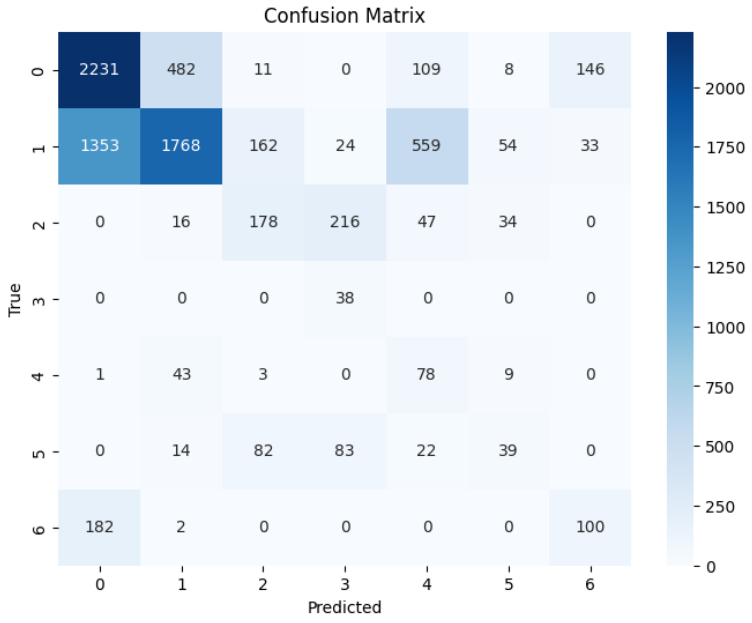

Figure 9: Confusion Matrix

As shown in the confusion matrix, it becomes evident that a considerable number of examples in the test set were misclassified. This persistent misclassification issue resulted in an excessively high log loss, rendering further investigation with this particular model futile.

[Texto relacionado con Ensemble Methods]

Random Forest

In this section, we will explore the Random Forest algorithm in detail. Random Forest is a versatile and powerful ensemble learning method that combines the concepts of decision trees and randomness to create a robust and accurate model. It operates by constructing multiple decision trees on different subsets of the training data and then aggregating their predictions through voting or averaging.

Similar to our approach with other models, we employed a training set splitting strategy to optimize a subset of the data and evaluate its performance on another subset. By dividing the training set into distinct portions, we could effectively tune the model's parameters and assess its performance on unseen data.

Initially, we sought to examine the relationship between the LogLoss metric and the number of estimators in order to understand how it evolved. As we have done in previous instances, we visualized the trend by plotting a graph to identify potential instances of underfitting and overfitting in relation to the new dataset.

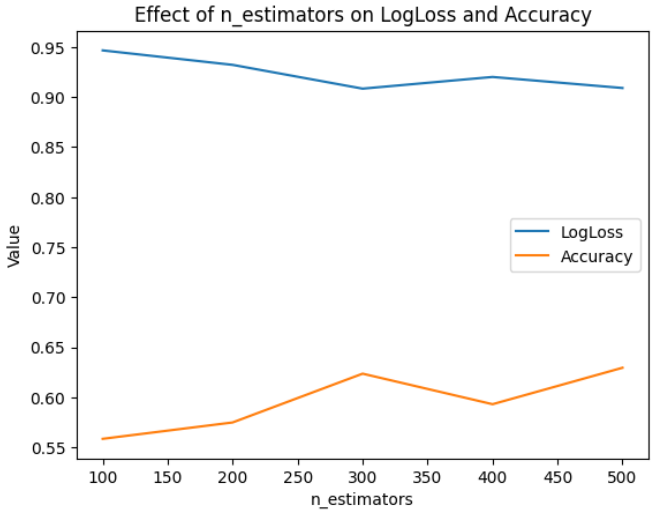

Figure 10: LogLoss Accuracy

Secondly, our investigation involved determining the optimal set of variables to include in the Random Forest algorithm, once we have fixed the *n_estimators*. To achieve this, we employed the SelectKBest function, which allowed us to identify the most influential features for model performance. Subsequently, given the extensive range of parameters available for Random Forest, we employed the RandomizedSearchCV class to systematically explore various parameter combinations and identify the configuration that would optimize algorithm performance. It was crucial to exercise caution during this process to avoid overfitting the model to the training data.

Best hyperparameters: n_estimators: 500, min_samples_split: 5, min_samples_leaf': 2, max_depth: None, criterion: gini

$$Log\ loss \approx 0.41\ \ Accuracy \approx 0.84$$

Having observed an improvement in our metrics, we decided to explore variable manipulation by experimenting with different combinations to assess the potential outcomes. Our criterion for selecting the variables to mix was based on the correlation between them. To our advantage, we succeeded in further enhancing the log loss, reaffirming the effectiveness of the algorithm. Consequently, we opted to proceed with this modified approach, leveraging the positive impact on our performance metrics.

The results obtained after these experiments were as follows:

$$Log\ loss \approx 0.38\ \ Accuracy \approx 0.86$$

Despite the presence of unclassified categories evident in the confusion matrix, the evaluation of the model on the test set revealed a remarkably low LogLoss metric, which proved to be the best among all previously obtained results. This discrepancy suggests that although certain categories may not be adequately classified, the overall predictive performance of the model remains considerably strong and promising.

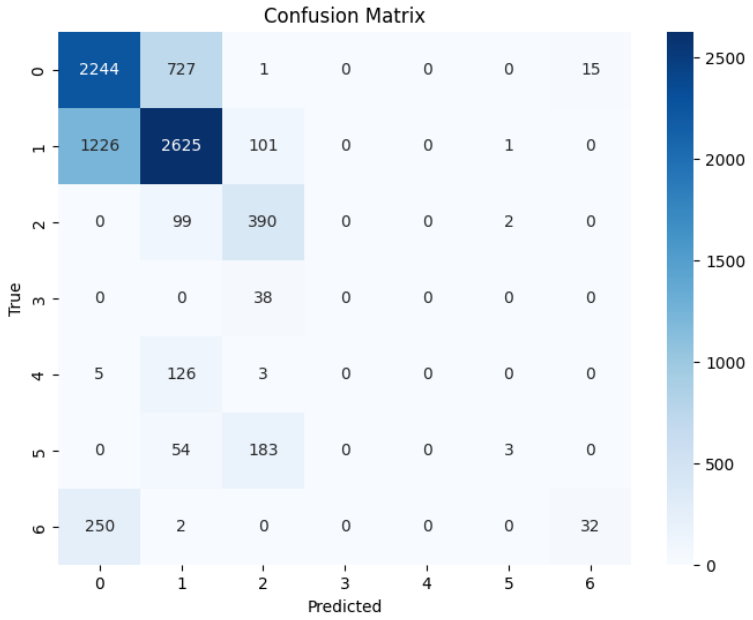

Figure 11: Confusion Matrix (RF)

3 Results and Discussion

First and foremost, the obtained result on the test set in the competition yielded a LogLoss score of 0.38, which aligned closely with the predictions made by the aforementioned algorithm.

From the comprehensive study conducted, several notable conclusions can be drawn. The initial observation relates to the issue of overfitting. Prior to implementing a proper separation of the training set, intensive efforts to enhance algorithms on the same training set inadvertently led to overfitting. Subsequently, when evaluating the performance on the test set and visualizing the LogLoss score, a significant increase was observed, which was anticipated given the evaluation on the identical training set.

With the implementation of a properly divided training set, noteworthy improvements were witnessed across all models utilized, albeit to varying extents. This led us to focus specifically on two models, namely Gradient Boosting and Random Forest.

A second evident conclusion is that the limited availability of data for certain classifications occasionally necessitated omitting any examples to achieve a more accurate approximation. Although peculiar, this approach was essential in obtaining better results.

Ultimately, it can be inferred that the combination of variables and the diligent study of parameters based on their respective values proved to be fundamental steps in achieving favorable outcomes. Furthermore, it is believed that by continuing along the current trajectory, even better scores could have been attained.

Appendix

In this section, we provide some refrence to diagrams or plot we have been made during the projet in order to understand better the data, how the method was developing,...

Box Plots

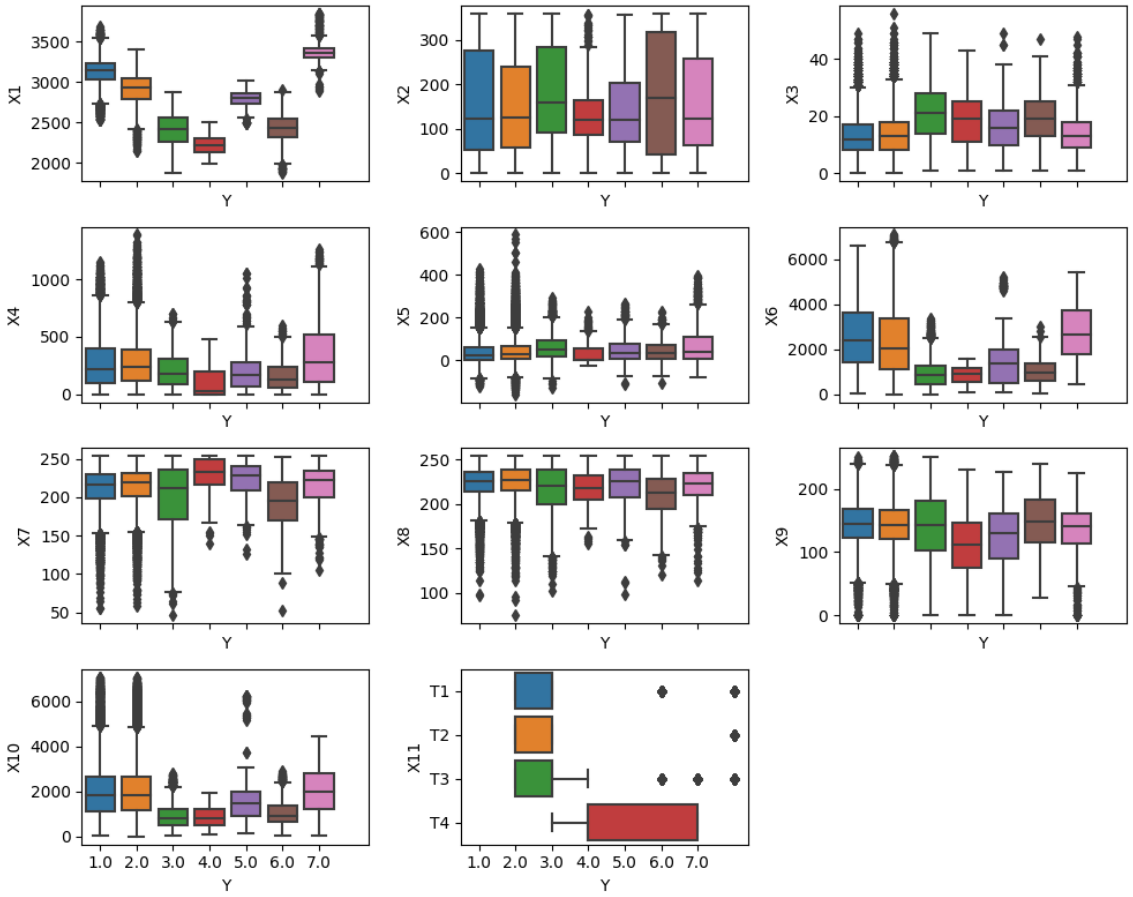

Figure 12: Box plot of numerical features.

Scatter Plots

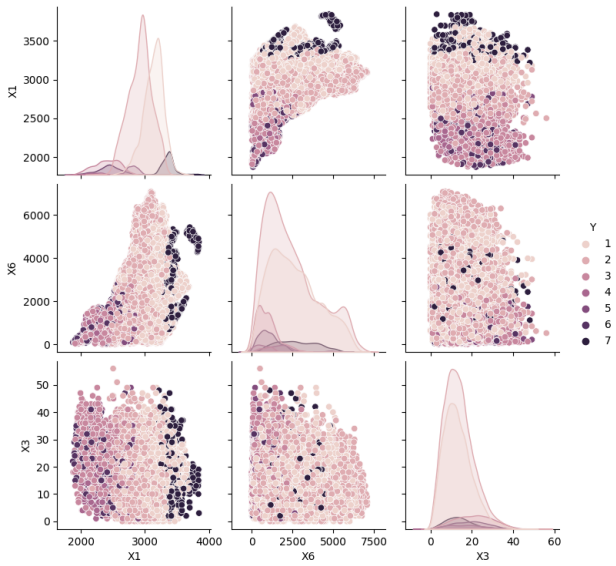

Figure 13: Scatter plot of X1, X3, X6