

# Binary Prediction of Smoker Status using Bio-Signals

Mikel Albisu

Andreu Artigues

Lea Brunner

Juan Sebastian Chombo

Yazeed Yabroudi

February 20, 2024

## **Abstract**

This project delves into a machine learning exercise aimed at predicting smoking status based on a series of medical features. The report comprehensively covers all stages involved in deploying the machine learning model. Through meticulous exploration, experimentation, and optimization, we aim to provide insights into the process of deploying machine learning models in healthcare contexts. This includes data collection, preprocessing, model selection, evaluation, and deployment considerations.

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Exploratory Data Analysis</b>	<b>3</b>
<b>3</b>	<b>Feature Engineering</b>	<b>5</b>
<b>4</b>	<b>Model Configuration</b>	<b>6</b>
4.1	Data Scaling and Sampling . . . . .	6
4.2	Hyper-parameter tuning . . . . .	7
<b>5</b>	<b>Model Selection</b>	<b>8</b>
5.1	First impressions . . . . .	9
5.2	Training with hyper-parameter tuning . . . . .	9
<b>6</b>	<b>Results</b>	<b>10</b>
<b>7</b>	<b>Model Evaluation</b>	<b>11</b>
7.1	Feature Importance . . . . .	15
<b>8</b>	<b>Conclusion</b>	<b>15</b>
<b>A</b>	<b>Evolution of the Accuracy and Precision-Recall Trade-off as a function of the threshold using the original dataset</b>	<b>17</b>
<b>B</b>	<b>Evolution of the Accuracy and Precision-Recall Trade-off as a function of the threshold using both versions of the transformed dataset</b>	<b>19</b>
<b>C</b>	<b>Kaggle submissions</b>	<b>20</b>

# 1 Introduction

Biological signals related to smoking behavior serve as invaluable indicators in healthcare research, offering insights into disease risks, treatment efficacy, and public health interventions. Our project aimed to harness the power of machine learning and algorithmic approaches to analyze these bio-signals, predicting smoking status, identifying at-risk populations and informing targeted interventions.

Despite the prevalence of smoking-related conditions, there has been a lack of specific diagnostic codes and treatment options. Many patients may receive other diagnoses before their smoking behavior is addressed, hindering both clinical care and research efforts. Our project sought to address these gaps by leveraging machine learning techniques to identify and characterize individuals based on their smoking bio-signals.

By leveraging administrative data sources, we aimed to improve the accuracy and scalability of smoking status identification, ultimately contributing to more targeted research and interventions in smoking cessation and related health outcomes. Moreover, our primary objective with this project was to build confidence and expertise in manipulating machine learning models. We aimed to understand the various challenges encountered when working with multiple machine learning models simultaneously for the first time and derive meaningful insights. This exploration extended beyond classification within the realm of health factors and laid the groundwork for future machine learning tasks.

## 2 Exploratory Data Analysis

Our initial examination of the data revealed significant variability among the features, particularly in terms of the quantity of unique values for each one. This underscores the importance of understanding each variable. Therefore, we considered scaling measures or reductions in complexity were necessary for optimal model performance.

As anticipated for a dataset sourced from Kaggle, the dataset is clean and free of major inconsistencies, eliminating the need for extensive cleaning procedures. However, to maintain data integrity, we conducted individual analyses of each variable to identify any potential null values and computed statistical measures such as mean, standard deviation, quartiles, among others...

These and other statistical metrics offered insights into the central tendency, dispersion, and overall distribution of the feature's values within our dataset. Additionally, two graphical representations of the data were generated, histogram and box-plot for each variable, to uncover any

hidden patterns that could impact our models.

In our analysis, we focused on evaluating the individual features to gain a comprehensive understanding of their characteristics and relevance to our objectives. Specifically, we assessed the correlation between each feature and our target variable, 'smoking'. This enabled us to determine which features were most pertinent to our goals and required further consideration for feature engineering.

We divided the process into six steps, including:

- Medical Description: Understanding the medical context of the data.
- Reasonable Values: Ensuring the validity and reasonableness of data points.
- Data Type: Verifying the appropriate data types for each variable.
- Values Collected: Examining the completeness and accuracy of data collection.
- Relationship with smoking: Investigating the associations between features and smoking behavior.
- Observations for feature engineering: Identifying insights and considerations for optimizing feature engineering strategies.

Firstly, we've identified strong correlations among various variables, suggesting opportunities to enhance our models through tailored ratios for the feature engineering process. However, a notable discrepancy existed in the age distribution within our dataset, particularly with a concentration of samples between 38 and 43 years. Rectifying this imbalance is imperative to ensure the representativeness of our analyses.

Furthermore, discrepancies in certain variables between smokers and non-smokers indicated the need for alignment to ensure consistency in our model predictions. Also, erroneous values were detected within variables such as eyesight, necessitating data validation and cleansing to maintain a robust foundation.

We found extreme values within certain features present a challenge to model stability and interpretation, requiring careful consideration and potential mitigation. Additionally, redundancies were identified in our dataset, including columns capturing duplicate metrics or derived features obtained through summation such as hearing and eyesight.

By addressing these findings, we elevate the accuracy and impact of our predictive analytics initiatives. Our exploratory data analysis revealed pivotal insights that demand attention in our

upcoming feature engineering and data preprocessing efforts.

### 3 Feature Engineering

First we remove the feature 'id', as it is irrelevant for the purpose of the project. After, we adopted an approach to handle the imbalance in our dataset, specifically regarding the age feature. This undersampling technique strategically balances the distribution of age groups within our data.

Second, we focus on a balanced mix of both age groups, rather than letting the majority class, 40 years old, overshadow the minority classes and misrepresent the dataset. This is done by selectively sampling the larger group, ensuring that both age groups are represented equally. We aim for an accurate model that reflects the age diversity of our population.

Next, we implemented a series of strategic steps to refine the features on our dataset. Firstly, we introduced a 'blood\_pressure\_ratio' to capture the relationship between systolic and relaxation blood pressures, streamlining the dataset by removing the original columns. Then, we further distilled the data by generating 'weight\_height\_ratio' and 'weight\_waist\_ratio', dropping the base measurements to focus on these more telling ratios.

Additionally, less predictive columns like 'hearing', which was giving contrary information due to low cases with hearing problems, and 'Cholesterol', as it is calculated from other metrics, were removed for a leaner dataset. Another step was the creation of 'eyesight\_max', aggregating eyesight data into a single feature that registers the worst metric between the left and the right eyesight. To address missing values covered as outliers on the eyesight feature, we imputed them based on the mean of the feature's data, ensuring a robust dataset.

Lastly, because some of the data points in these features were extreme, we employed clipping to limit the range of values, ensuring our variables stay within realistic bounds<sup>1</sup> as seen in [Table 1](#):

---

<sup>1</sup>All variables present a minimum value equal to 0.

Feature	Max Value
Blood_pressure_ratio	150
Fasting blood sugar	200
HDL	110
LDL	200
Serum creatinine	4
Urine protein	4
AST	100
ALT	100
Gtp	300

Table 1: Range of values to make sure we are not dealing with extreme values so that they stay within realistic bounds.

This preprocessing lays a foundation for deploying machine learning models to further understand the interplay between bio-signals and smoking status.

## 4 Model Configuration

### 4.1 Data Scaling and Sampling

After performing the different changes to the original dataset, one step prior to using the dataframe was the scaling of the data. Some algorithms, like gradient descent use the Euclidean distance between two data points in their computations. If one feature has a broad range of values, the distance will be dominated by this feature. Therefore, the range of all features should be normalized so that each feature contributes approximately proportionately to the final distance. We used `MinMaxScaler()` to do so and not others like `StandardScaler()` because of the non-Gaussian distribution of our data, or `RobustScaler()` because our transformed dataset does not present outliers.

On top of that, we quickly realised that aiming to train a model given approximately 160 thousand records would be a difficult task for our computers. This is why we opted for taking a random sample out of the whole training dataframe and train the models with 30% of the original number of records. This helped us reduce waiting time and achieve a smoother code execution. With the dataset ready, it was time to use it for the different machine learning algorithms selected for the project.

Nevertheless, this idea turned out to completely knock down our models completely.

<b>submission_dt_original_df.csv</b> Complete (after deadline) · 2d ago	<b>0.5</b>	<b>0.5</b>
<b>submission_clf_original_df.csv</b> Complete (after deadline) · 2d ago	<b>0.5</b>	<b>0.5</b>

Figure 1: Kaggle scores after training the Logistic Regression and Decision Trees using sampled records.

What in the beginning seemed like a good and reasonable idea, had to be immediately discarded. This situation forced us to work and deploy our based based on the training of the whole dataset from now on.

## 4.2 Hyper-parameter tuning

Although the models could be trained by simply calling them, we optimised their performance by tuning the hyper-parameters to best suit the characteristics of our dataset. This approach aimed to enhance the predictive power of our models and ensure their optimal performance in capturing the underlying patterns present in the data.

The hyper-parameters between models are different because each machine learning algorithm has its own set of characteristics, complexities, and underlying assumptions. Different machine learning algorithms have varying levels of complexity. For example, decision trees have hyper-parameters related to the tree's depth, number of leaves, and splitting criteria, while logistic regression has regularisation hyper-parameters like the penalty term. XGBoost may require tuning of parameters related to boosting rounds and tree depth, while SVM may require tuning of kernel parameters and regularisation strength.

As it can be seen, not only each model requires personalised tuning, but the different combinations must be tested. This is because when trying different combination of hyper-parameters, we can obtain the model with the best performance. In order to achieve this, through the assignment process we first made use of the `GridSearch()` function, which exhaustively tries every combination of the provided hyper-parameter values to find the best model. While this can be very effective, it can also be very time-consuming, especially if the number of parameters and the range of values are large.

In the end, we soon realised that taking advantage of the `RandomizedSearchCV()` function was a better approach. This method samples a given number of candidates from a parameter space with a specified distribution. This approach can be more efficient than `GridSearch()` because it

doesn't try every combination, but rather a random subset of combinations. This can save a lot of time and still yield a good result, especially when dealing with a large number of parameters and/or large datasets.

Independently of the function chosen, in both cases the function performed  $K$ -Fold cross-validation for evaluating the model. This helps in selecting the most suitable model for your data by comparing the performance on different folds and choosing the model with the hyper-parameters that performs best on average. Additionally,  $K$ -Fold helps to balance bias and variance. A single train/test split can have high variance due to the randomness of the split.  $K$ -fold cross-validation reduces the variance by averaging over  $K$  different partitions, so the performance estimate is less sensitive to the partitioning of the data. It can be said at this moment, that even prior to using the latter mentioned functions, only `cross_val_score()` was used with the models due to the lack of experience in model tuning. Nevertheless, this helped us obtain a first insight into the models running time as well as its score results.

When using `RandomizedSearchCV()`, the number of folds used were  $K = 5$  and we then decided to use as scoring metric the ROC AUC (Receiver Operating Characteristic Area Under the Curve). ROC AUC is a widely used metric for evaluating the performance of binary classifiers, and it provides a comprehensive measure of a model's ability to distinguish between positive and negative classes across various threshold settings. However, while ROC AUC scores are indeed comparable between different models, it's important to note that they are not always directly interpretative in terms of model performance. For example, a model with a higher ROC AUC score may not necessarily perform better in terms of other evaluation metrics like accuracy, precision, or recall<sup>2</sup>.

## 5 Model Selection

The reason why more than one model was selected for this assignment, was to find out the different features that characterise each one of them, such as its scoring results, processing time or sensibility to over-fitting among others. We highlight that, the machine learning models chose for this project have been taught in class and according to the assignment rules, models such as Neural Networks or any other supervised or unsupervised model not seen in class have not been taken into account for this project. We can enumerate the different characteristic of each one that we thought to use.

Firstly, Logistic Regression creates a very easily understandable model that is less inclined to

---

<sup>2</sup>Further explanation in [section 7](#)



over-fitting and perform with good accuracy. Unfortunately this is not our scenario. Moreover, feature engineering is required since it is quite prone to noise. Obviously, this model assumes linearity, something that we cannot guarantee at the moment. However, since this was the first model we learned, we decided to try it out.

Then we moved to the non-probabilistic models. Decision Trees are straightforward and easy to understand, making them suitable for explaining predictions. They can capture complex relationships between features and the target variable but are prone to over-fitting and high variance. Random Forests, on the other hand, are an ensemble method that combines multiple decision trees to improve predictive performance and reduce over-fitting. By using bagging techniques, random forests reduce variance and provide more stable predictions compared to individual decision trees.

Furthermore, XGBoost is a gradient boosting algorithm that iterative builds an ensemble of weak learners, typically decision trees, to optimize predictive performance. It incorporates regularization techniques to prevent over-fitting and uses a gradient descent approach to minimize loss during training. Finally, SVMs are robust against over-fitting, especially in high-dimensional spaces, although they are computationally intensive.

## 5.1 First impressions

When calling each default models in our simplest scenario, where the default models were used with no hyper-parameter tuning nor feature engineering (only scaling), Logistic Regression and XGBoost seemed to be the models that behaved more realistically with ROC AUC mean value over 5 folds of 0.8338 and 0.8536 respectively. Once the model was fitted to the same sampled dataset the scores obtained were of 0.83 and 0.94, respectively. On the other hand, models like Decision Trees and Random Forest when putted to predict values showed a ROC AUC of 1. At first, this seemed suspicious to us, but due to the nature of the dataframe and the simplicity of the technique used we kept the models for further inspection. However, SVM were very time-consuming algorithm. Added to the fact that this model is not efficient with large number of observations we decided at this point to not continue with this model due to the ineffectiveness for model exploration. We can also notice that all the previous models except for Random Forest were easily computable, whereas this last one took several minutes to run.

## 5.2 Training with hyper-parameter tuning

At this point we developed the necessary code to train each model that we wanted to work with given a series of hyper-parameter values with the help of `RandomizedSearchCV()`. Throughout

the experimentation with the assignment, we highlight 3 situation in which depending on the dataset used how would the performance result. This 3 dataset were:

1. Original dataframe: No feature engineering/selection process was done to the given Kaggle dataset.
2. Transformed dataframe: Feature engineering process was made adequately to the meaning of the features.
3. Transformed dataframe, with ages under-sampled: Added to the above point and given a great unbalance in ages, an under-sampling technique was implemented here.

However, it was after the first scenario, that we stopped using all models to just focus one of them; the one that overall performed the best as a combination of running time, predicted accuracy score on the training data and the final Kaggle score. Through this process, we are formulating a plan to optimize our strategy for identifying the best possible model efficiently. This involves considering that we intend to experiment with a variety of potential models.

## 6 Results

In the present section we present the results obtained in Kaggle during the different interventions explained above.

	Log. Reg.	Decision Tree	Random Forest	XGBoost
Public Score	0.7673	0.75894	0.78071	0.78673
Private Score	0.76207	0.75714	0.77894	0.78687

Table 2: Kaggle scores for the different models trained on the whole original dataset.

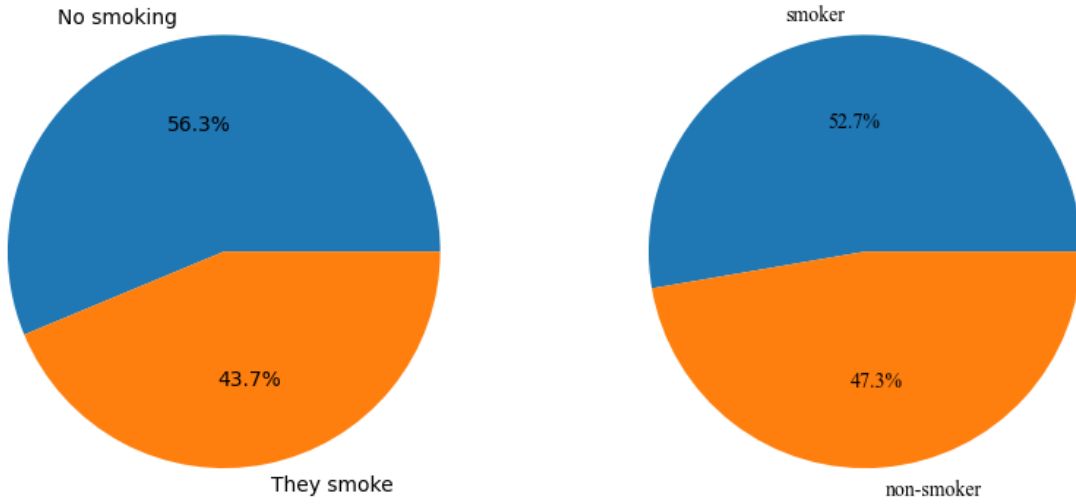
The first thing to notice here is that all results comprehend a close range of score, with a minimum score of 0.75714 and a maximum one of 0.78687. To this, it can be seen that Decision Tree was the model that performed the worst, followed by Logistic Regression and Random Tree. XGBoost, as expected, was the one to get the highest accuracy.

In terms of running time, Random Forest was the most time-consuming one. As a matter of fact, in Andreu’s computer is wasn’t able to run in 8’3 h but in Yazeed took 15 min. So it’s a model highly sensible to the CPU usage.

	Private Score	Public Score
Original	0.78687	0.78673
Transformed	0.7484	0.74877
Transformed & ages undersampled	0.75247	0.75457

Table 3: Kaggle scores for XGBoost as our final model trained different settings of dataframes.

After the different transformations on the dataset, we got no doubt that the model that got us a higher score used the original Kaggle dataset, the one with no feature engineering/selection at all. To this model there is one thing interesting to notice in the image below.



(a) Smokers distribution in the original training dataset.

(b) Smokers distribution for the predictions in the test dataset.

With the final model, its even interesting to notice the distribution of the smoking status, both in the training set and predicted test set. It stands out that, while in the original dataset 56.3% of records corresponded to non-smoker, the predictions gave more representation to smokers than to those who aren't.

## 7 Model Evaluation

As it was previously explained, one can not only use the ROC AUC for testing a model's performance. This is why we elaborated the self-named `threshold()` for the probabilistic models (Logistic Regression and XGBoost). When evaluating the model using the ROC AUC, the threshold for setting the binary classification is set at default at 50%. However, evaluating a model in a more understandable way we opted to study the accuracy metric. We wanted to find the optimal

threshold that contributed to the most accuracy. Let's present the different metrics for the optimal threshold for the different datasets used with the help of the `classification_report()`<sup>3</sup> function.

	precision	recall	f-1 score
non-smoker	0.83	0.70	0.76
smoker	0.68	0.82	0.75
accuracy			0.75

Table 4: Scoring metrics for Logistic Regression model trained on the original dataframe

	precision	recall	f-1 score
non-smoker	0.82	0.71	0.76
smoker	0.68	0.82	0.74
accuracy			0.75

Table 5: Scoring metrics for Decision Tree model trained on the original dataframe

	precision	recall	f-1 score
non-smoker	0.91	0.95	0.93
smoker	0.93	0.88	0.91
accuracy			0.92

Table 6: Scoring metrics for Random Forest model trained on the original dataframe

	precision	recall	f-1 score
non-smoker	0.86	0.78	0.81
smoker	0.74	0.83	0.79
accuracy			0.80

Table 7: Scoring metrics for XGBoost model trained on the original dataframe.

In order to get a sense of the importance of the threshold optimization, let's look at one example in [Figure 3](#).

In the upper picture, we can see this belly shape reaching a maximum that then starts smoothly falling down. Meanwhile, the bottom image highlights another concept: the precision-recall trade-off, which holds significant importance depending on the context and the specific business

---

<sup>3</sup>for non-probabilistic models, the function used is called `is_tree()`

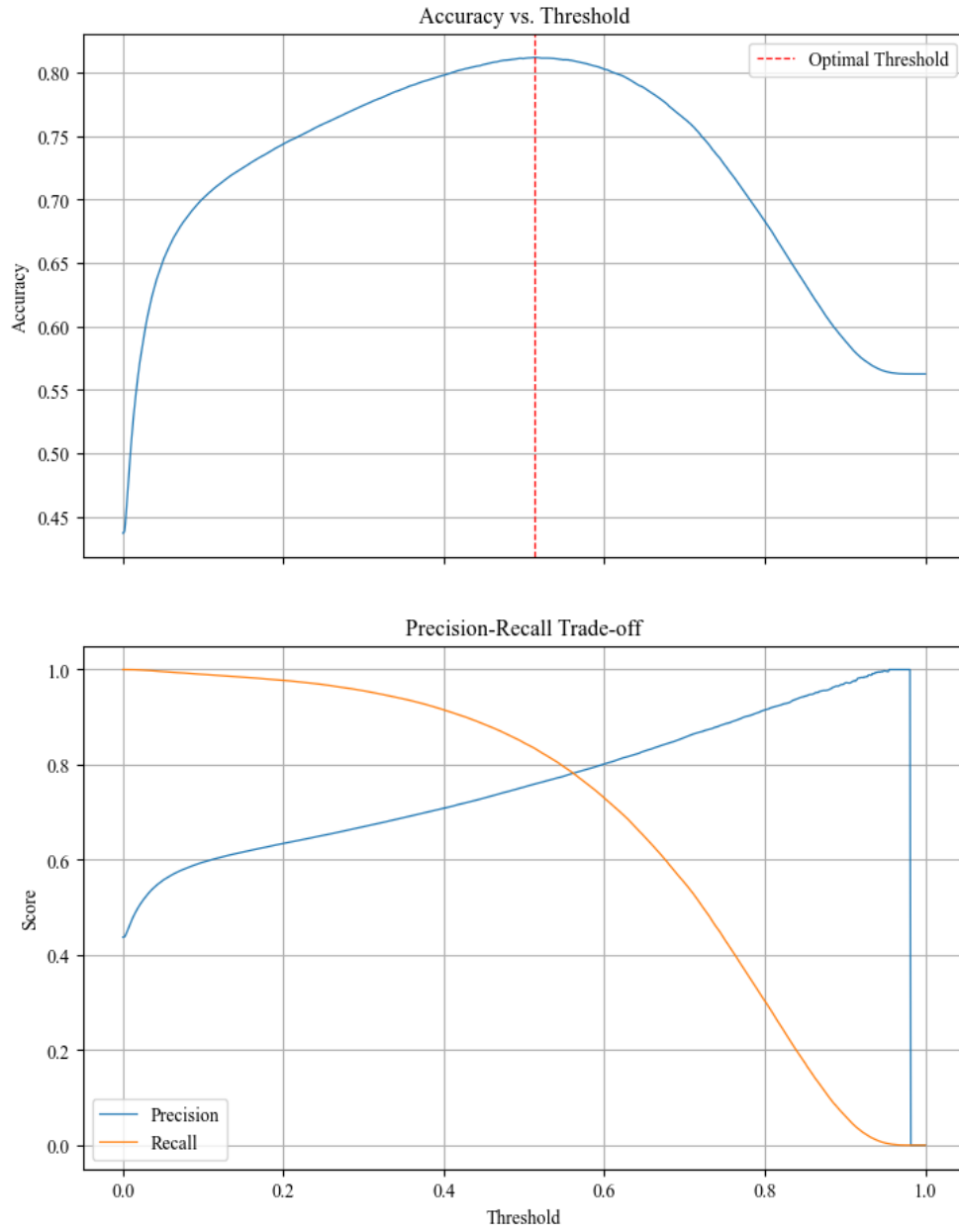


Figure 3: Evolution of the Accuracy and Precision-Recall Trade-off in terms of different class 1 thresholds when using a scaled version of the Original training dataframe using XGBoost as our model.

need at hand. The precision-recall trade-off refers to the inherent relationship between precision and recall in classification tasks. Increasing precision typically results in a decrease in recall, and vice versa, as the model’s threshold for classifying positive instances is adjusted. Achieving higher precision often means sacrificing recall, and vice versa, leading to a balancing act to optimize both metrics based on the specific needs of the application. For more visual information, please refer to [Appendix A](#).

As already explained, after performing the different machine learning models, XGBoost was the model we selected to perform further studies. When we aimed to delve deeper into the feature engineering part aiming for improving the Kaggle score, turns out it happened the opposite. As previously seen in [Table 3](#), both transformation decreased the score minimum in a 4%, something not foreseen in the classification report as the values did not seem to indicate lower performance.

	precision	recall	f1-score
non-smoker	0.86	0.81	0.84
smoker	0.78	0.83	0.80
accuracy			0.82

Table 8: Scoring metrics for XGBoost model trained on the transformed dataframe.

	precision	recall	f1-score
non-smoker	0.85	0.80	0.82
smoker	0.76	0.81	0.78
accuracy			0.80

Table 9: Scoring metrics for XGBoost model trained on the transformed and age undersampled dataframe.

This was a very unexpected result to us. The only possible explanation to why this is happening could be that around 25% of the data is real actual data, whereas the rest is artificially generated. Perhaps, this newly artificial generated data could not be truly capturing the essence of the original dataset.

## 7.1 Feature Importance

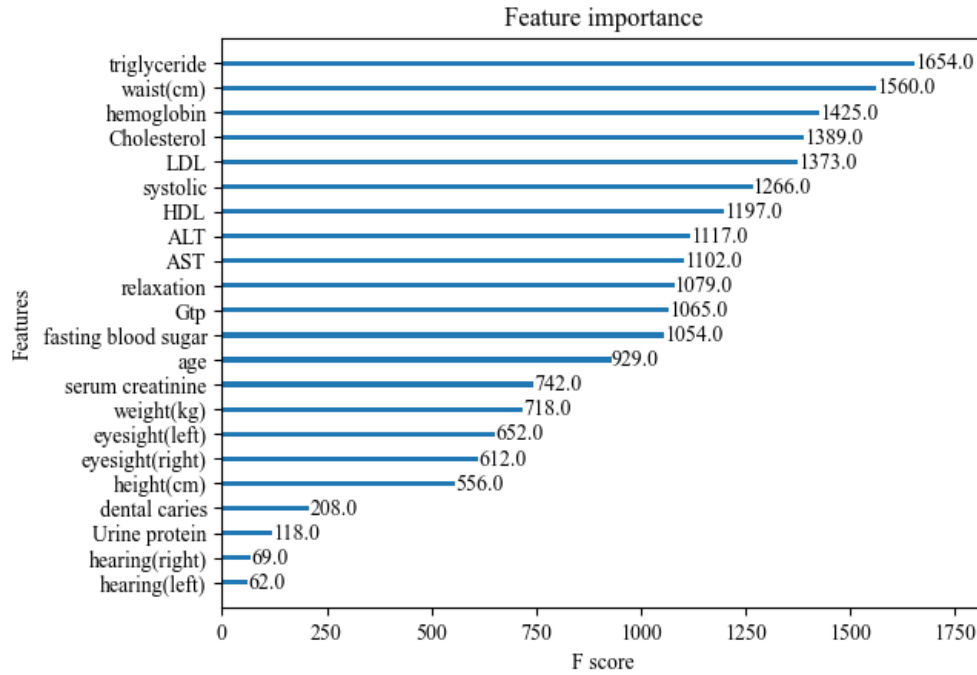


Figure 4: Distribution of smokers status in the test dataset.

Lastly, we wanted to go on step further and get to know what are the features in our model that played a major role. Which ones are most relevant. The analysis of the chart reveals notable variability in the significance of each feature within the model. The triglyceride level emerges as the most influential factor, serving as a robust indicator of dietary habits. This finding suggests a strong correlation with smoking behavior, a conclusion supported by the observation that waist circumference—the second most impactful factor and a metric closely associated with obesity—also plays a significant role. Additional variables of importance include hemoglobin, which is linked to lung health, and various cholesterol-related metrics. Conversely, features deemed least critical to the model encompass auditory variables, aligning with expectations set by our Exploratory Data Analysis (EDA). Also categorized as less influential are factors pertaining to kidney function, such as urine protein and serum creatinine levels, along with dental caries.

## 8 Conclusion

This assignment has served us to gain our first real involvement in a Machine Learning practice from all perspectives, from Exploratory Data Analysis to Model Evaluation, with all its intermediate trials and errors. In the end, we were left with a model with a Kaggle ROC AUC score of 78.687% using XGBoost trained on the Kaggle provisioned dataset. We can highlight that

our model is very sensitive to over-fitting. In all cases, the difference between our score and the Kaggle one was noticeably different, indicating that our model focused so much on the training data that perhaps on new unseen data it did not perform adequately. One possible solution to over-fitting for further improvements would be to use techniques of dimensionality reduction such as PCA. We can also say that we have been very surprised by the fact that despite engineering the features to more meaningful ones due to the fact that we knew what each one of them knew, actually led to performance degradation. Also, getting to the root of why sampling didn't even predict with sense would be a priority to get out of doubts.

During this assignment, we also recognized the vast number of possible combinations that can be explored, both in the feature engineering stage and during model deployment. The permutations between these elements are extensive, but the challenge here relies is being able to identify and condense the possibilities to a few. One undeniable believe we all commonly shared is that, as seen in many Kaggle notebooks for this assignment, most of them rely on focusing on a single model and using brute force to train as best a possible, which turned out to have great score. Although only models discussed in class have been utilized, it is acknowledged that there exist alternative models that may yield superior performance. Some of these alternative models include LightGBM, CatBoost, as well as hyper-parameter tuning techniques such as Optuna or Ray Tune.



## A Evolution of the Accuracy and Precision-Recall Trade-off as a function of the threshold using the original dataset

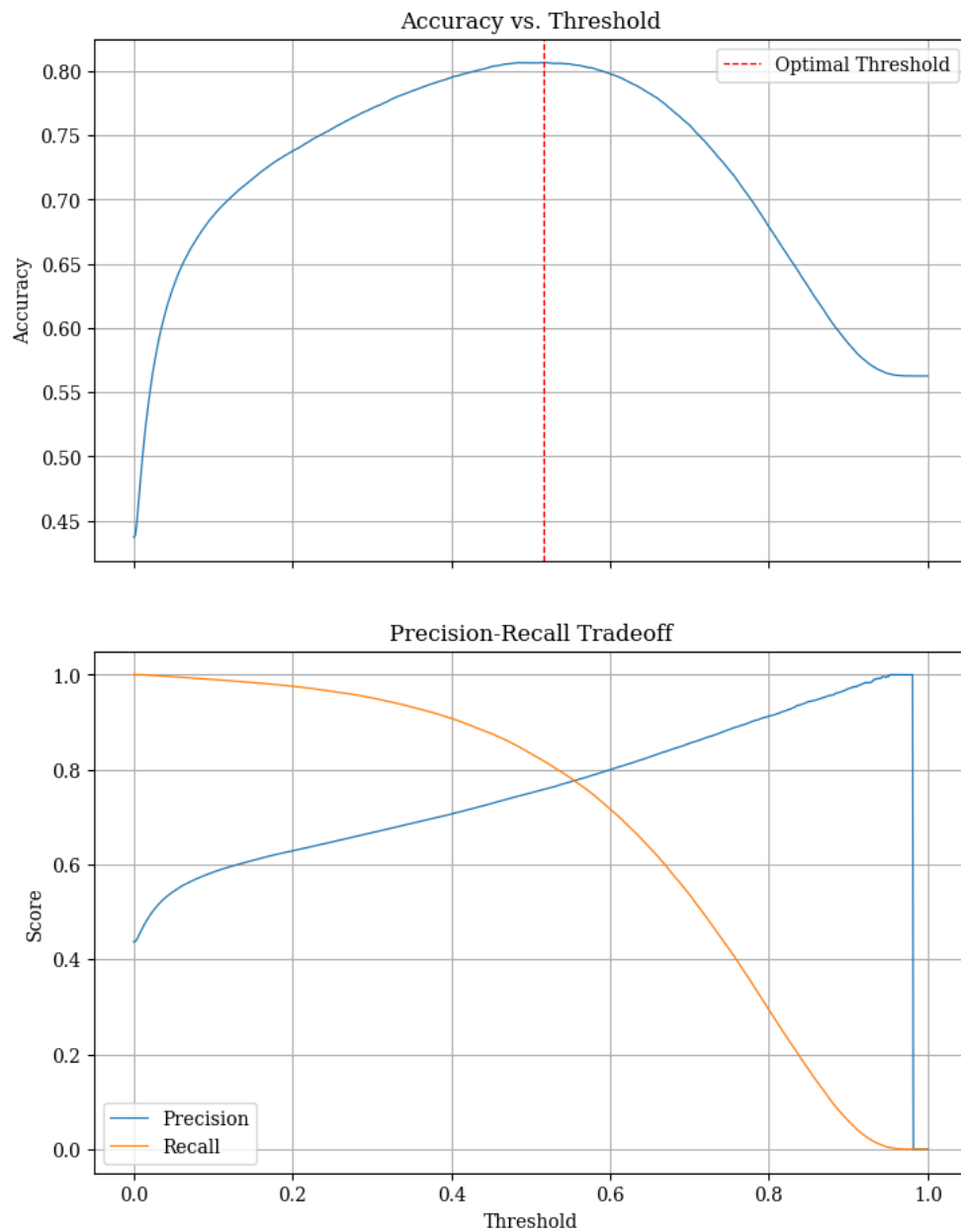


Figure 5: Evolution of the Accuracy and Precision-Recall Trade-off in terms of different class 1 thresholds when using a scaled version of the Original training dataframe using Logistic Regression as our model.

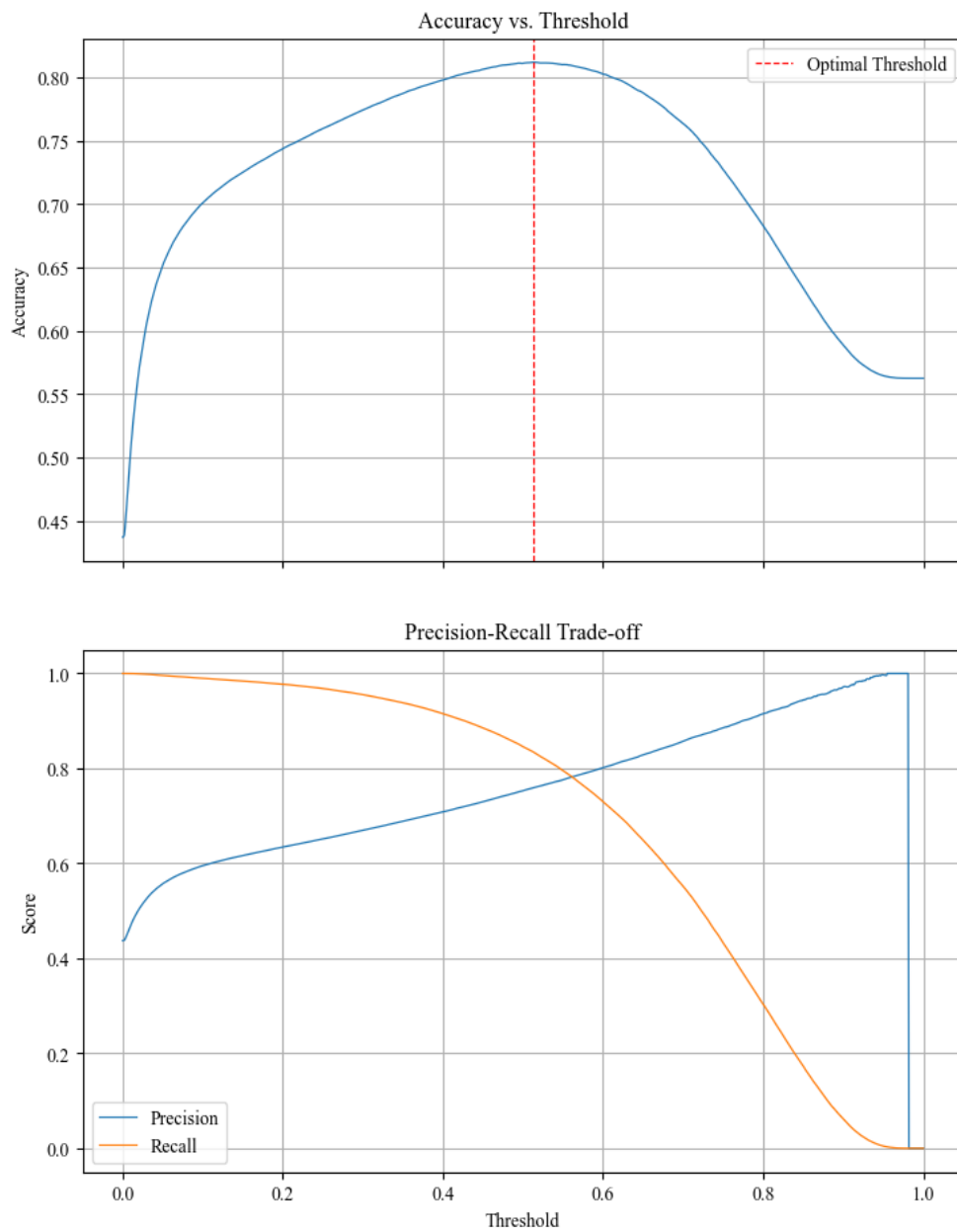


Figure 6: Evolution of the Accuracy and Precision-Recall Trade-off in terms of different class 1 thresholds when using a scaled version of the Original training dataframe using XGBoost as our model.

**B Evolution of the Accuracy and Precision-Recall Trade-off as a function of the threshold using both versions of the transformed dataset**

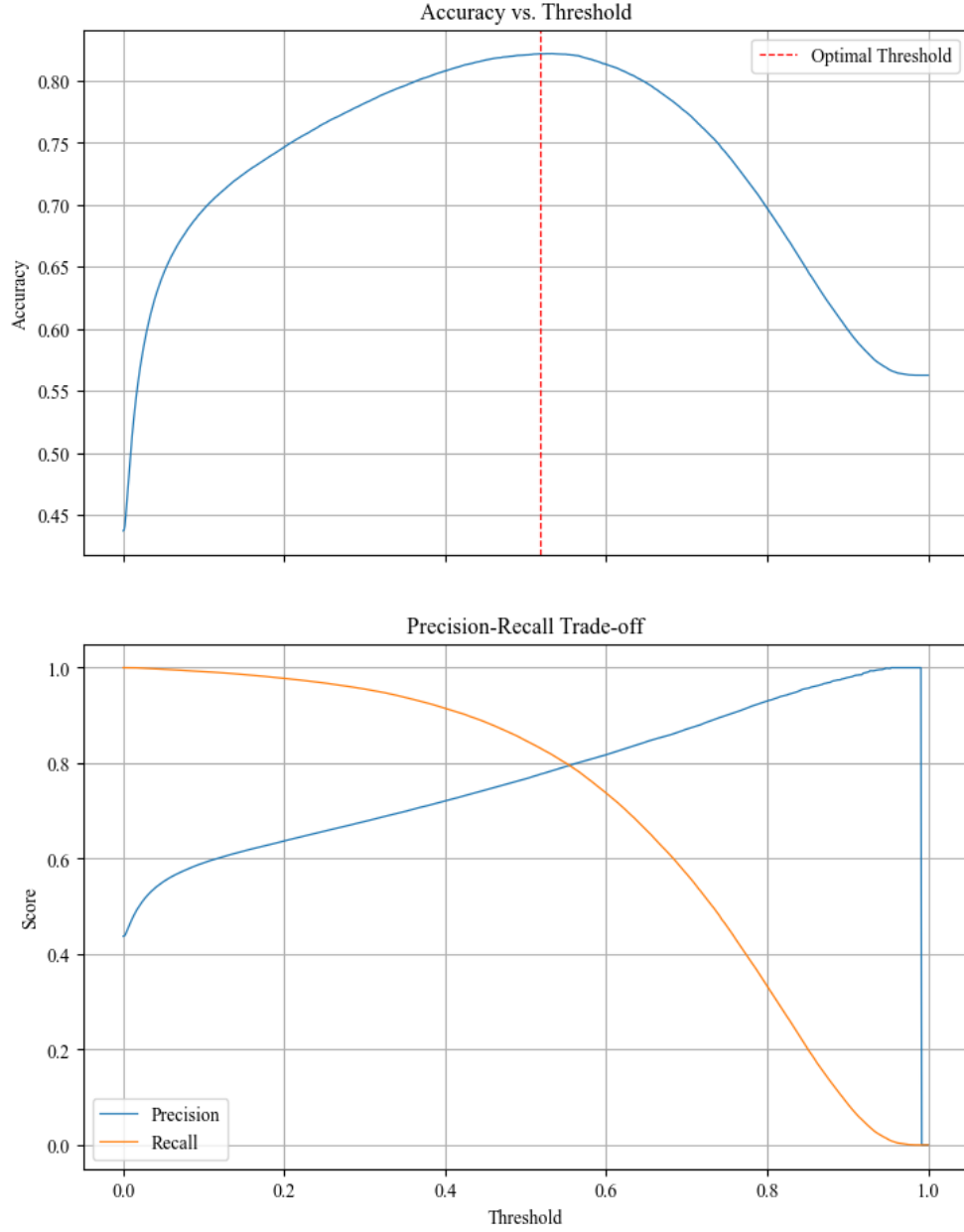


Figure 7: Evolution of the Accuracy and Precision-Recall Trade-off in terms of different class 1 thresholds when using a scaled version of the transformed dataset using XGBoost as our model.

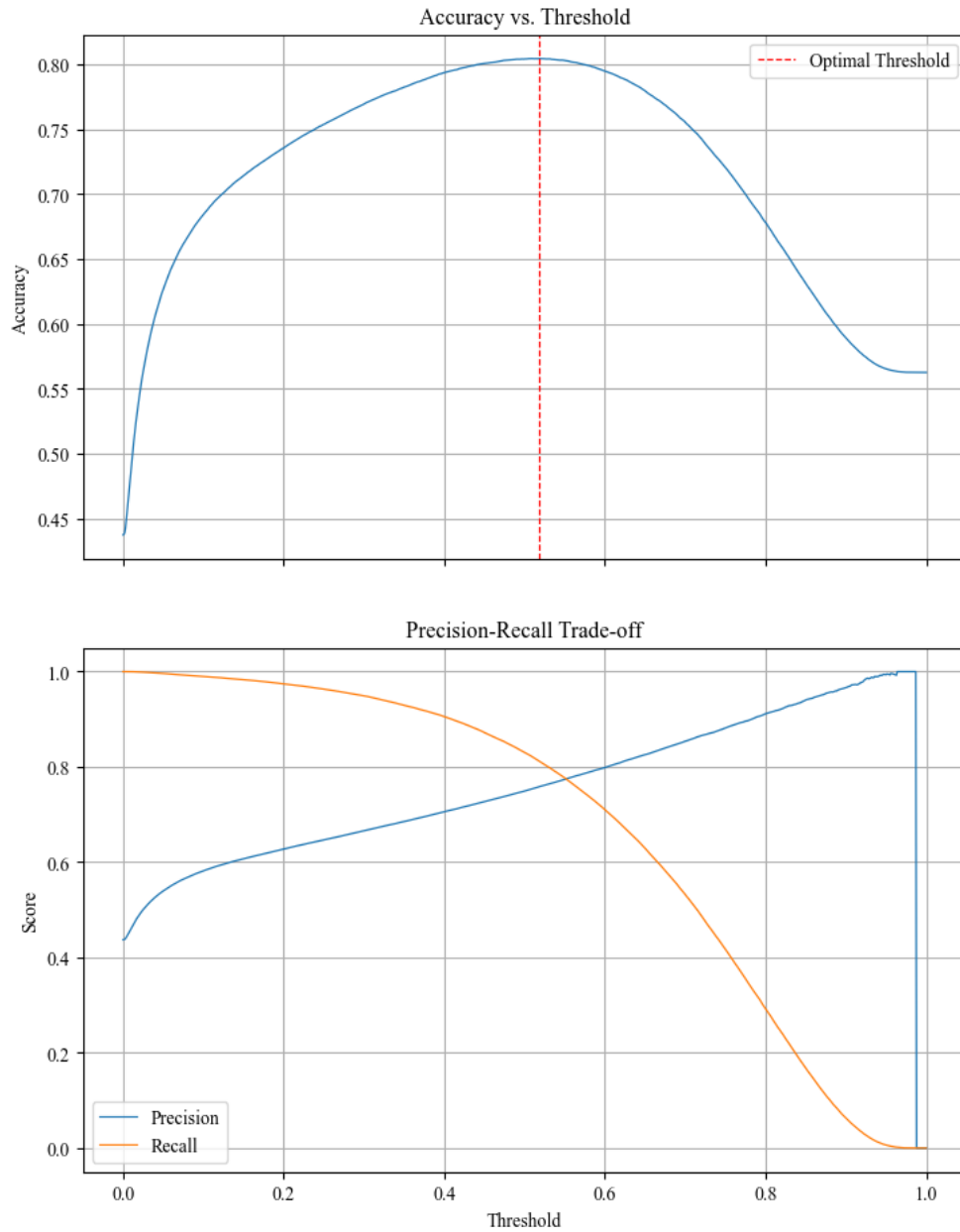


Figure 8: Evolution of the Accuracy and Precision-Recall Trade-off in terms of different class 1 thresholds when using a scaled version of the transformed and age under-sampled dataset using XGBoost as our model.













## C Kaggle submissions

These screenshots make up for the whole submission this group has done to Kaggle.

## Binary Prediction of Smoker Status using Bio-Signals

Late Submission

Overview Data Code Models Discussion Leaderboard Rules Team

	<b>submission_xg_transformed_aged_df.csv</b> Complete (after deadline) · 10s ago	<b>0.75247</b>	<b>0.75457</b>
	<b>submission_xg_transformed_df.csv</b> Complete (after deadline) · 21h ago	<b>0.7484</b>	<b>0.74877</b>
	<b>submission_xg_original_df.csv</b> Complete (after deadline) · 1d ago	<b>0.78687</b>	<b>0.78673</b>
	<b>submission_rf_original_df.csv</b> Complete (after deadline) · 1d ago	<b>0.77894</b>	<b>0.78071</b>
	<b>submission_dt_original_df.csv</b> Complete (after deadline) · 1d ago	<b>0.75714</b>	<b>0.75894</b>
	<b>submission_clf_original_df.csv</b> Complete (after deadline) · 2d ago	<b>0.76209</b>	<b>0.7673</b>
	<b>submission_dt_original_df.csv</b> Complete (after deadline) · 2d ago	<b>0.5</b>	<b>0.5</b>
	<b>submission_clf_original_df.csv</b> Complete (after deadline) · 2d ago	<b>0.5</b>	<b>0.5</b>
	<b>submission.csv</b> Complete (after deadline) · 4d ago	<b>0.76215</b>	<b>0.7621</b>
	<b>submission.csv</b> Complete (after deadline) · 5d ago	<b>0.78566</b>	<b>0.78741</b>
	<b>submission.csv</b> Complete (after deadline) · 5d ago	<b>0.5</b>	<b>0.5</b>
	<b>submission.csv</b> Complete (after deadline) · 9d ago	<b>0.76262</b>	<b>0.76037</b>