Université Jean Monnet de Saint-Etienne

Machine learning

Pr. EYRAUD Remi
Pr. SERRANO Richard
Pr. KALIDINDI SHANMUGHA Sri

AMIROUCHE Azedine BAYAZID Hany CARVAJAL Alejandro

March 21, 2024

1 Workload

Contributers	Contribution
AMIROUCHE Azedine	30%
BAYAZID Hany	30%
CARVAJAL Alejandro	30%
Generative AI (ChatGPT 3.5)	10%

2 Introduction

During the past decade, artificial intelligence **(AI)** has witnessed an exponential growth, emerging as a focal point of innovation and developement across numerous industries.
Giants in the technology sector, like Google, Mircosoft and even NVIDIA, have intensified their focus and increased their budget on AI research and implementation.
The domain of artificial intelligence is vast and encompasses numerous fields. However, discussions about artificial intelligence often overlook its diverse composition.

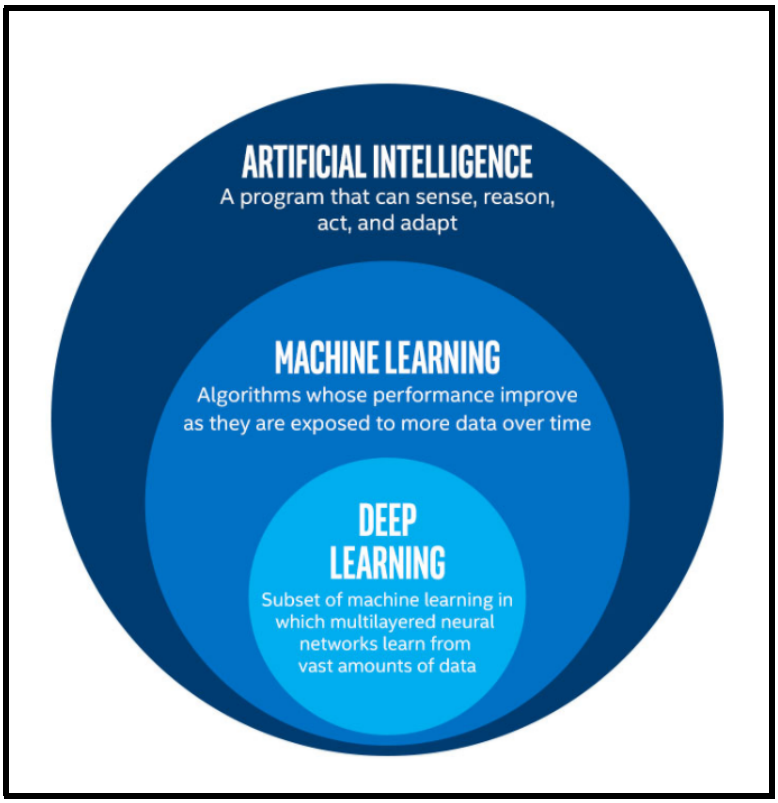


Figure 1: The diverse fields of artificial intelligence

The image above illustrates the various fields that constitute this expansive domain.
Each of these domains has its own field of practice and uses.
Some of these domains are :

- **Machine learning :**
  This domain of artificial intelligence involves the development of algorithms and statistical models that enable computers to perform specific tasks without explicit instructions.
  This domain of artificial intelligence involves the development of algorithms and statistical models that enable computers to perform specific tasks without explicit instructions. ML techniques are commonly utilized in tasks such as classification, where data is categorized into distinct groups based on specific features, and clustering, which involves grouping similar data points together based on inherent patterns or similarities.

- **Deep learning :**
  Deep learning is a subfield of machine learning that focuses on training artificial neural networks

with multiple layers to learn hierarchical representations of data.
DL algorithms have demonstrated remarkable success in various tasks such as image recognition, natural language processing (NLP), and speech recognition.

This accentuates why understanding these various domains is crucial in order to choose the right methods for the specific tasks at hand.

3 Support Vector Machine Algorithm (SVM)

In this section, we will delve into a brief explanation of how Support Vector Machines (SVMs) work.

Support Vector Machines (SVMs) are supervised learning models used for classification and regression analysis. They are particularly useful in scenarios where data is separable into distinct classes or groups. The fundamental principle behind SVMs is to find the optimal hyperplane that best separates data points belonging to different classes in a high-dimensional feature space.

- **Linear Separability :**
  The primary objective of SVM is to find the hyperplane that maximally separates the data points belonging to different classes.
- **Margin Maximization :**
  In addition to the hyperplane, SVMs use a margin, which is the distance between the hyperplane and the nearest data points from each class, also known as support vectors. SVMs aim to maximize this margin, as it leads to better generalization and improved performance on unseen data.
- **Feature space :**
  SVM operates in a high-dimensional feature space where each data point is represented by a vector.
  The vector itself is a vector containing multiple elements about an object. These features could be any measureable characteristics of the data that is being used for whatever task at hand. Some examples of characteristics could be pixels of an image, or attributes of a dataset, or numerical data.

In cases where the data is not linearly separable in the original feature space, SVMs employ the use Kernels (for example polynomial or sigmoid) that allow to implicitly map the input data into a higher-dimensional space where it becomes linearly separable.

4 Metrics

To accurately assess the performance of our classifier, we have chosen specific metrics that provide insight into its effectiveness. These metrics allow us to gain a deeper understanding of the results produced by our model :

- **Accuracy :**
  Accuracy informs us about the proportion of correct predictions compared to the total number of evaluated samples. We used it to observe the difference between the results of the validation set and the training set.
- **F1-score :**
  The F1-score informs us about the performance of our model in a balanced way since it takes into consideration both false positives and false negatives, providing a more comprehensive measure of a classifier's effectiveness, unlike accuracy, which may be misleading in the presence of imbalanced data.

5 Part I : Random points dataset

In this section, an artificial balanced dataset containing 1000 samples was generated with two distinct classes characterized by two-dimensional features. Each class has a normally distributed data with unique mean and standard deviation parameters.

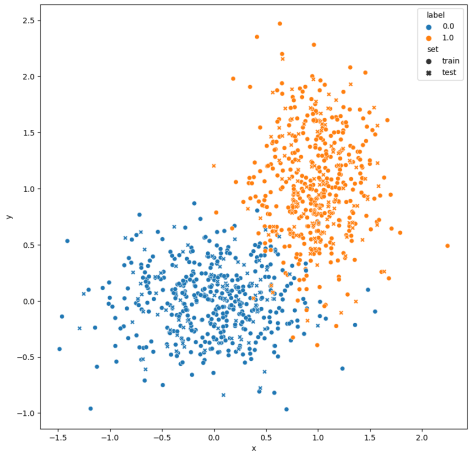


Figure 2: Dataset with class distinction.

5.1 Finding best parameters

The hyperparameters tuned were: the regularization parameter, denoted as C, with candidate values of 1, 10, 50, and 100. The kernel parameter, with possible values such as linear, radial basis function (rbf), and polynomial (poly). Additionally, the gamma parameter, with options of 'scale' and 'auto', was explored.

Employing the *GridSearchCV* option from *sklearn*, the best combination of hyperparameters among the possible options specified was found. This search procedure was complemented by cross-validation. The optimal hyperparameters ascertained through this process were determined to be: $C = 1$, kernel = *linear*, and gamma = *scale*

5.2 Results

Utilizing the optimal hyperparameters derived from the cross-validation process, a model was trained. After classifying the samples on the test set, a 96% accuracy was found. The following graph shows the decision surface and the support vectors, clarifying the model's discriminative efficacy

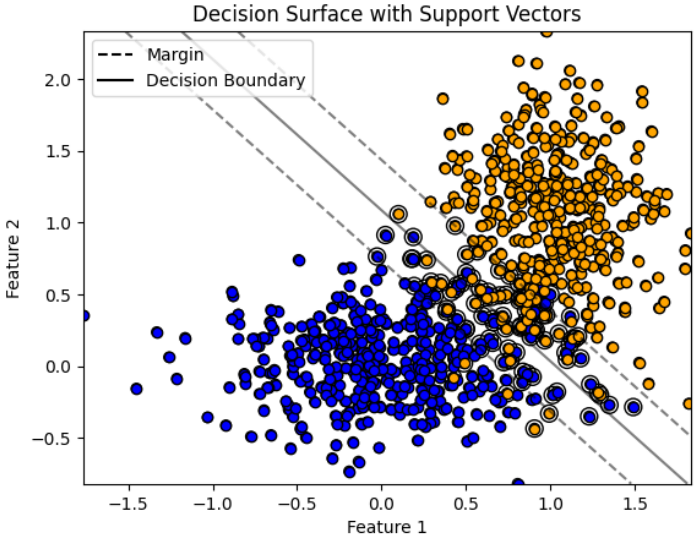


Figure 3: Decision boundary display of the SVM model

6 Part II : Pulsar stars dataset

The pulsar dataset describes a sample of pulsar candidates.
The dataset is split in two dataframes, **pulsar_train_df** and **pulsar_test_df**, each one respectively containing 12528 candidates and 5370 candidates.

6.1 Data preprocessing

Before working with the pulsar dataset, we had to verify the data's integrity and make sure that the data is actually in a good condition to work with.
This step ensured that we could work with reliable data, enhancing the accuracy of our classification tasks.

6.1.1 Handling missing values

Before doing anything, we started by checking if our dataset contains missing values or not, and if it does, how to correctly fill the missing values.
We found that each dataframe contained a certain number of missing values detailed in the table below : We calculated the percentage of missing values for each feature in both dataframes. Given the

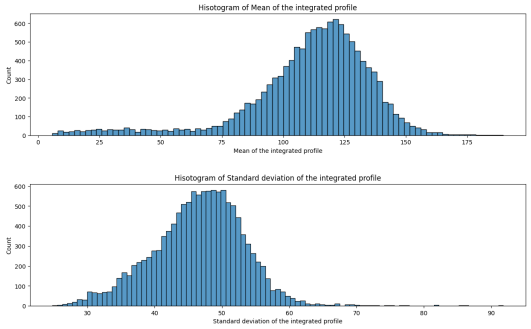
Feature	**Training Set**	**Ratio (Training)**	**Test Set**	**Ratio (Test)**
Mean of the integrated profile	0	-	0	-
Standard deviation of the integrated profile	0	-	0	-
Excess kurtosis of the integrated profile	1735	13.85%	767	14.28%
Skewness of the integrated profile	0	-	0	-
Mean of the DM-SNR curve	0	-	0	-
Standard deviation of the DM-SNR curve	1178	9.40%	524	9.76%
Excess kurtosis of the DM-SNR curve	0	-	0	-
Skewness of the DM-SNR curve	625	4.99%	244	4.54%
target_class	0	-	5370	-

Table 1: Number of missing values in each dataframe

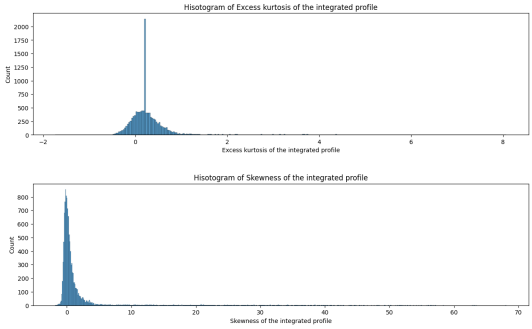
presence of missing data, it was necessary to check the distribution of our dataset before proceeding with any imputation

6.1.2 Data distribution

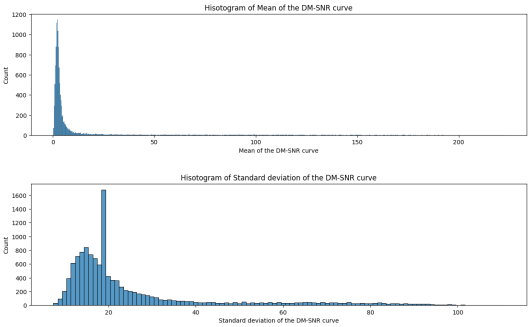
To find whether or not our data is normally distributed, we used the Shapiro-Wilk test to compute the p-value and see whether or not we reject the hypothesis of whether our data fits a normal distribution. We've done this test for both our training and test dataframes, nd it indicated that the data is uniformly distributed in both cases. However, since it's just a simple hypothesis, we can't assume that the data fits a normal distribution without a visual verification.



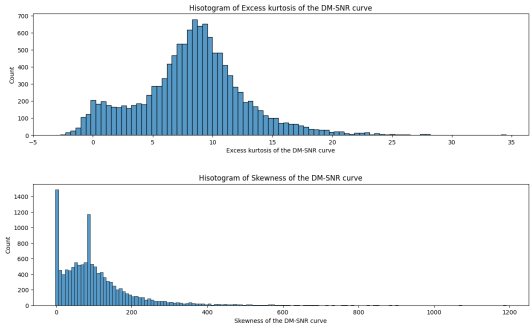
(a) Mean of the integrated profile and Std of the integrated profile



(b) Excess kurtosis of the integrated profile and Skewness of the integrated profile



(c) Mean of the DM-SNR curve and Std of the DM-SNR curve



(d) Excess kurtosis of the DM-SNR curve and Skewness of the DM-SNR curve

Figure 4: Histogram plots of each of the features in the pulsar training dataset

6.1.3 Skewness

After plotting our data in our training set, it became clear to us that the data was skewed. This information needs to be taken into account when imputing missing values in order to make sure that the data is correctly filled and doesn't falsify the performance of our model.
To be completly sure about skewnsess, we've also used the **skew()** function in order to see if our features are right-skewed or left-skewed.

Feature	**Training**	**Testing**
Mean of the integrated profile	-1.386845	-1.348135
Standard deviation of the integrated profile	0.083716	0.219772
Excess kurtosis of the integrated profile	3.658438	3.552928
Skewness of the integrated profile	5.235255	5.045145
Mean of the DM-SNR curve	3.650861	3.762464
Standard deviation of the DM-SNR curve	1.896039	1.883122
Excess kurtosis of the DM-SNR curve	0.443657	0.434087
Skewness of the DM-SNR curve	2.708265	2.808774

Table 2: Skewness of each feature for the training and testing dataframes

The above table showcases how skewed each feature is, and we'll use this data to impute our missing values accordingly.

6.1.4 Imputing

After we've made sure that our data is skewed and in which direction it is skewed, we decided to impute our missing values using the SimpleImputer from Scikit-learn.

- For the training set, we used the median as a strategy for most of the features except for the **Standard deviation of the integrated profile** feature, which has a skewness that is the closest to zero.
- For the testing set, we used the median as a strategy for all the features, since no feature is really close to zero.

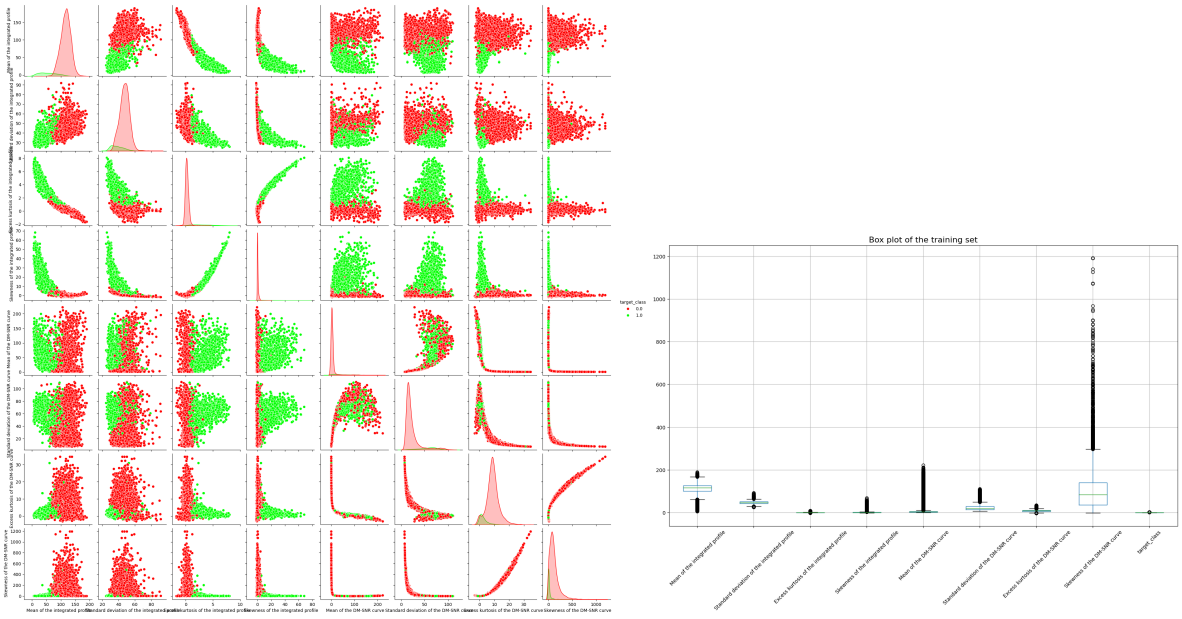
The reason behind using the median as an imputing strategy is because this strategy is robust to outliers (which we will talk about later) and offers more accurate values when the data is skewed.

6.1.5 Scaling

Now that the missing values have been filled in our dataset, we proceeded to scaling our data in order to ensure optimal performance during model training and evaluation.
We have hesitated a lot in choosing our scaling algorithim, since each and every single one of them have different use cases, which could lead our model to underperform if the wrong choice is made.
We decided to scale using **Min-Max Scaling** since it preserves the shape of the original distribution, making it suitable for non-Gaussian data, which is the case we're in.

6.1.6 Handling outliers

To ensure consistency and detect outliers within our dataset, we opted to employ box plots and pair plots.



(a) Pair plot of the training set

(b) Box plot of the training set

Figure 5

As we can observe from the figures above, the presence of outliers is undeniable in our dataset, and they must be handled in some way.
However, When it came to handling outliers, we were pretty hesitant on whether or not should we get rid of them, since this dataset applies to a natural phenomenon, and removing outliers might actually affect the performance of our model, but their presence can also lead to inaccuracy in the model's performance.
That being said, we decided to evaluate our model with and without treating the outliers, to see if their persence would really affect our model's performance.
The way in which we decided to deal with outliers is by capping them, which consists of setting a predetermined maximum value and replacing any data points exceeding this maximum with the value of the maximum itself. This method reduces the effect of extreme values on our analysis while keeping the integrity of the dataset.

The maximum value has been computed using the interquartile range **(IQR)** which is the difference between the 75th and 25th percentile of the data.
Any value that is below the lower bound by $1.5 \times IQR$ is replaced by the lower range, and any value that exceeds the upper bound by the same formula is replaced by that upper bound.

6.2 Finding best parameters

Now that our data has been completely set up to work with, we decided to train and evaluate our model based off a parameter grid that we defined as follows :

Kernel	**C**
Linear	1–100
RBF	1–100
Poly	1–100
Sigmoid	1–100

Table 3: Parameter Grid

We are trying to find the best combination of C values ranging from 1 to 100 and finding the best kernel from the list.

6.3 Results

In this section, we will detail the obtained results with our model.

Metric	**With Outliers**	**Without Outliers**
Best Parameters	{C: 100, kernel: 'rbf'}	{C: 100, kernel: 'rbf'}
Accuracy	0.9789269969326014	0.9776499340879017
F1 Score	0.877216107199688	0.8730093180451227

Table 4: Comparison of results with and without Outliers

As we can see, the values obtained after evaluating our model with and without outliers didn't affect the performance of our model.

7 Part III : Personal dataset : SA-Heart disease

7.1 Data Analysis

7.1.1 Data description

The data includes ten relevant characteristics for assessing cardiovascular health. These features include systolic blood pressure (sbp) (min: 101, max: 218, mean: 138.33), cumulative tobacco consumption (tobacco) (min: 0, max: 31.20, mean: 3.64), low density lipoprotein cholesterol (ldl) (min: 0.98, max: 15.33, mean: 4.74), adiposity (min: 6.74, max: 42.49, mean: 25.41), family history of heart disease (famhist) (yes or no), type-A behavior (typea) (min: 13, max: 78, mean: 53.10), obesity (min: 14.7, max: 46.58, mean: 26.04), current alcohol consumption (alcohol) (min: 0, max: 147.19, mean: 17.04), age at onset of study (age) (min: 15, max: 64, mean: 42.81), and the presence of coronary heart disease (chd) (yes or no). These variables provide a comprehensive overview of risk factors and factors influencing cardiac health in the dataset.

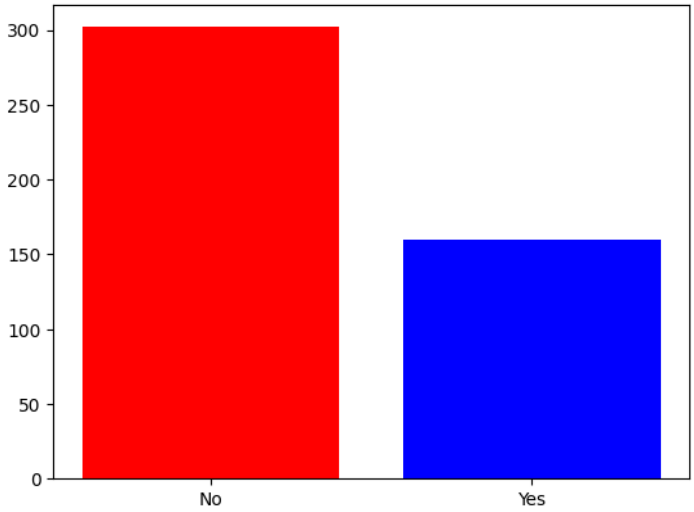


Figure 6: Distribution of the data

7.1.2 Data distribution

The data is imbalanced: Out of a total of 462 individuals, 302 do not have coronary heart disease (65.37 %) compared to 160 individuals who do (34.63 %). We can easily conclude that this sample does not represent a normal population. Indeed, according to the British Heart Foundation, in 2024, there are 620 million people worldwide with heart problems, which represents slightly less than 10 % of the global population. We doubt that this figure, although significant, reaches 30 % in South Africa.

We also observe that there is no strong correlation between the features, with the strongest correlation being between adiposity and obesity (correlation coefficient: 0.72). The pairplot also shows us that the data is not linearly separable, which could justify the use of a model like Support Vector Machine.(Figure 7)

7.2 Data preprocessing

For preprocessing, we decided to use a SMOTE algorithm to balance the data by adding points to the minority class, which consists of positive cases (individuals with coronary disease). After processing, the distribution can be seen in Figure 8.

Fortunately for us, this dataset has neither duplicate values nor missing values. However, we performed a transformation on the "famhist" column by converting "present" and "absent" into 0 or 1. 0 when the individual has no family history and 1 when they do. To achieve this, we used the Pandas DataFrame function get_dummies().

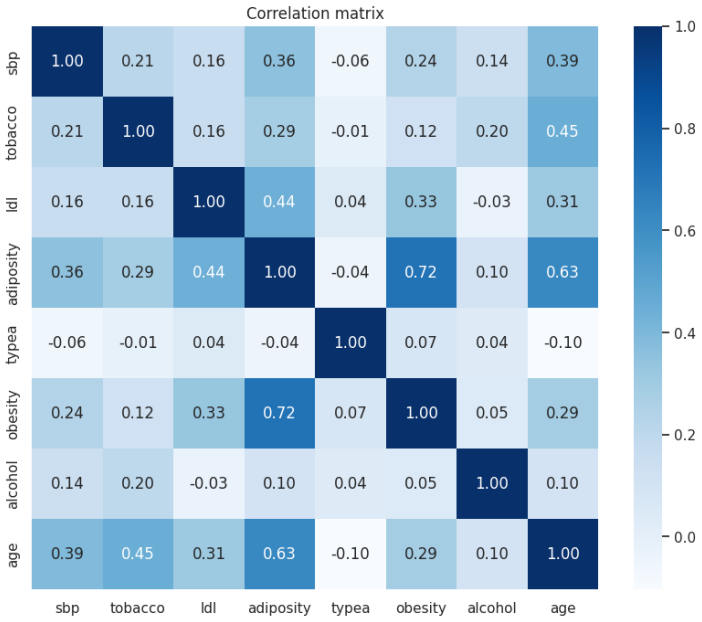


Figure 7: Correlation Matrix of the SA-heart dataset

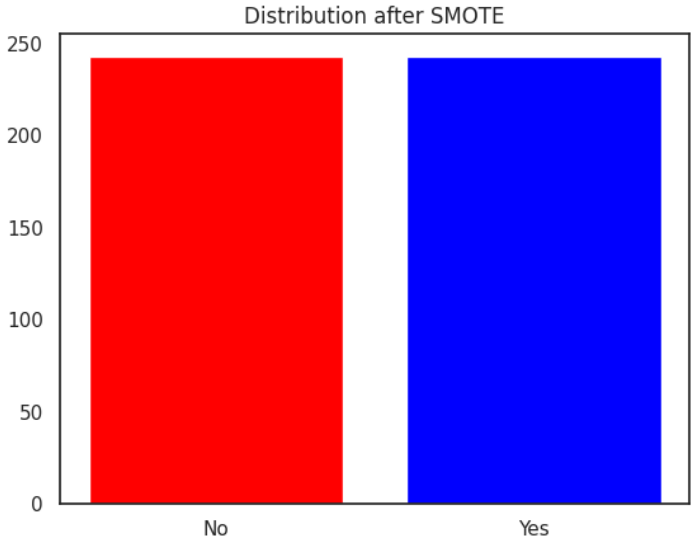


Figure 8: Using SMOTE on imbalanced dataset(SA-heart)

7.3 Finding best parameters

To find the best parameters, we used a GridSearch Cross Validation. Initially, we used a dictionary of all parameters to be tested, but the computation time was extremely long. Not knowing where the issue lay, we decided to investigate. Perhaps because there were too many parameter combinations to analyze. Therefore, we separated the analysis of the kernels from the analysis of the parameters specific to those kernels.

First, we compared the kernels separately. Then, we focused on the polynomial kernel by testing various degrees (from 2 to 25). Finally, for our tests, we examined the parameters C and gamma in a third stage.

7.4 Results

Metric	**C: 10, kernel: 'linear', gamma : 0.001**	**degree: 21, 'poly'**
Accuracy	0.7634408602150538	0.7

Table 5: Best accuracy for 2 tuned models on SA heart dataset

7.4.1 Best kernel : linear

Upon observing the results, we note that the winning kernel with an accuracy of 0.75 is the linear kernel. It achieved a precision of 0.73, recall of 0.73, and F1-score of 0.73 each.

The RBF and polynomial kernels come in second place with slightly less than 0.70 accuracy, while the sigmoid kernel trails far behind with approximately 0.40 of accuracy. (Figure 9)

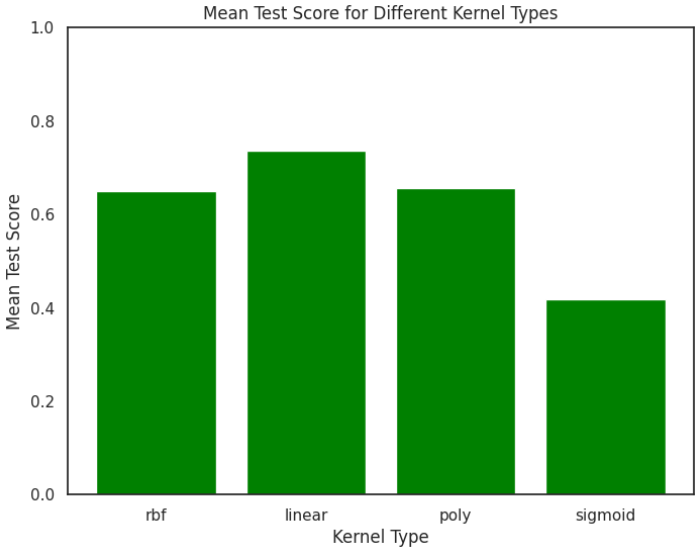


Figure 9: Finding the best kernel

7.4.2 Finding the best degree for polynomial kernel

To ensure that we do not achieve better results with the polynomial kernel, we test various polynomial degrees for our model. We obtain the highest score of 0.7 for degree 21. Beyond 25, the computation time becomes too long, and to avoid overfitting, we conclude that the best kernel is the linear kernel.

7.4.3 others parameters...

Finally, by tuning the parameters C and gamma, we find that the best compromise is C: 10 and gamma: 0.001. We achieve a mean test score of 0.68.

In conclusion, when we combine all our parameters, that is, C: 10, kernel: linear, gamma: 0.001, we achieve an accuracy of 0.76.

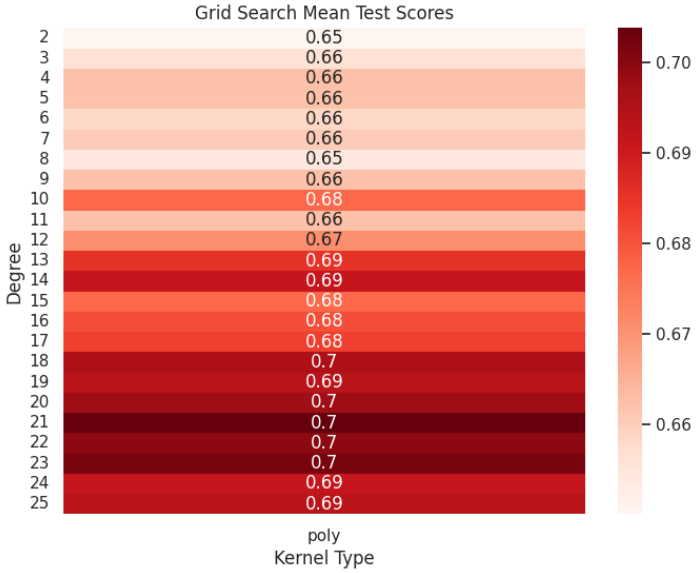


Figure 10: Heatmap to find the best degree for polynomial kernel

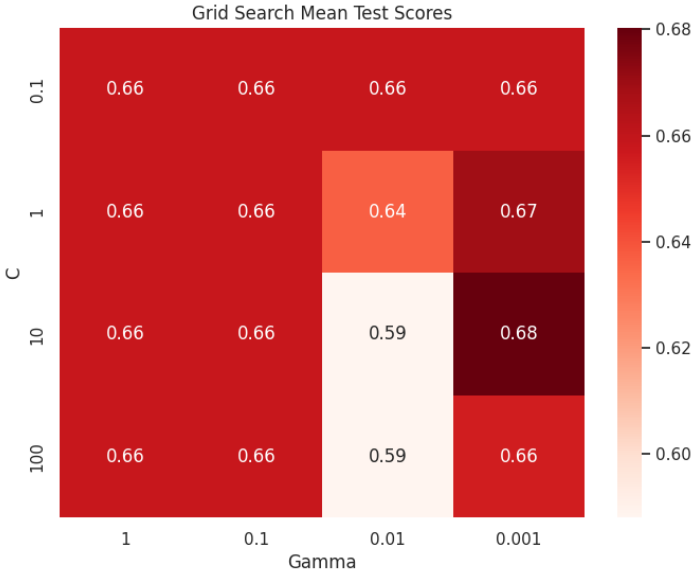


Figure 11: Tuning C and gamma parameters

8 Conclusion

In conclusion, this project provided us with the opportunity to put into practice the knowledge acquired during the semester in the Machine learning class.
The project was comprehensive enough to allow us to consolidate our knowledge and to gain a deeper understanding of what machine learning is and how it works in detail.

It also taught us that we should deeply analyze and treat our data before doing any work, no matter what the final objective is.
This includes ensuring data quality by detecting and handling missing values, outliers, and inconsistencies. By addressing these issues, we ensure that the data used for analysis and modeling is accurate and reliable, which directly impacts model performance by improving accuracy and reducing overfitting.
Our project reinforced the importance of investing time and effort in data preparation to ensure the reliability and effectiveness of the results obtained by any model, regardless of the task.