

**RAPPORT CODING-WEEK :**

**Bone-Marrow**

**Prediction**

**PROJECT 4: MEDICAL DECISION SUPPORT APPLICATION PREDICTING SUCCESS OF PEDIATRIC BONE MARROW TRANSPLANTS WITH EXPLAINABLE ML (SHAP)**

**GROUPE 28**

**Summary**

**1. Abstracts**

**2. Project Goals**

**3. Task Distribution and Management**

**4. Preliminary Data Analysis (Critical Questions)**

**5. ML Models**

**6. Optimizing Memory Usage**

**7. Interface Development**

1. **Abstract**

We have been tasked with developing a decision-support application to assist physicians in predicting the success rate of bone marrow transplants in pediatric patients. The motivation of the study is to prevent failure in transplantation procedure . For this purpose, we worked with a data set that describes pediatric patients with several hematologic diseases: malignant disorders (i.a. acute lymphoblastic leukemia, acute myelogenous leukemia, chronic myelogenous leukemia, myelodysplastic syndrome) and nonmalignant cases (i.a. severe aplastic anemia, Fanconi anemia, with X-linked adrenoleukodystrophy). All patients were subject to the unmanipulated allogeneic unrelated donor hematopoietic stem cell transplantation. [Bone marrow transplant: children - UCI Machine Learning Repository](https://archive.ics.uci.edu/dataset/565/bone+marrow+transplant+children)

1. **Project Goals**

* Develop a robust, explainable machine learning model.
* Ensure transparency of model predictions using SHAP explainability
* Create an intuitive user interface (Streamlit or Flask)
* Follow professional software development practices (GitHub, automated CI/CD)
* Demonstrate prompt engineering by clearly documenting AI-generated prompts used in our workflow

1. **Task distribution and management**

To better organize our work, we used Trello, an online project management tool .

1. **Preliminary Data Analysis (Critical Questions)**

* **Missing Values**

Before all we decided to verify the missing values proposition in the set of each column.

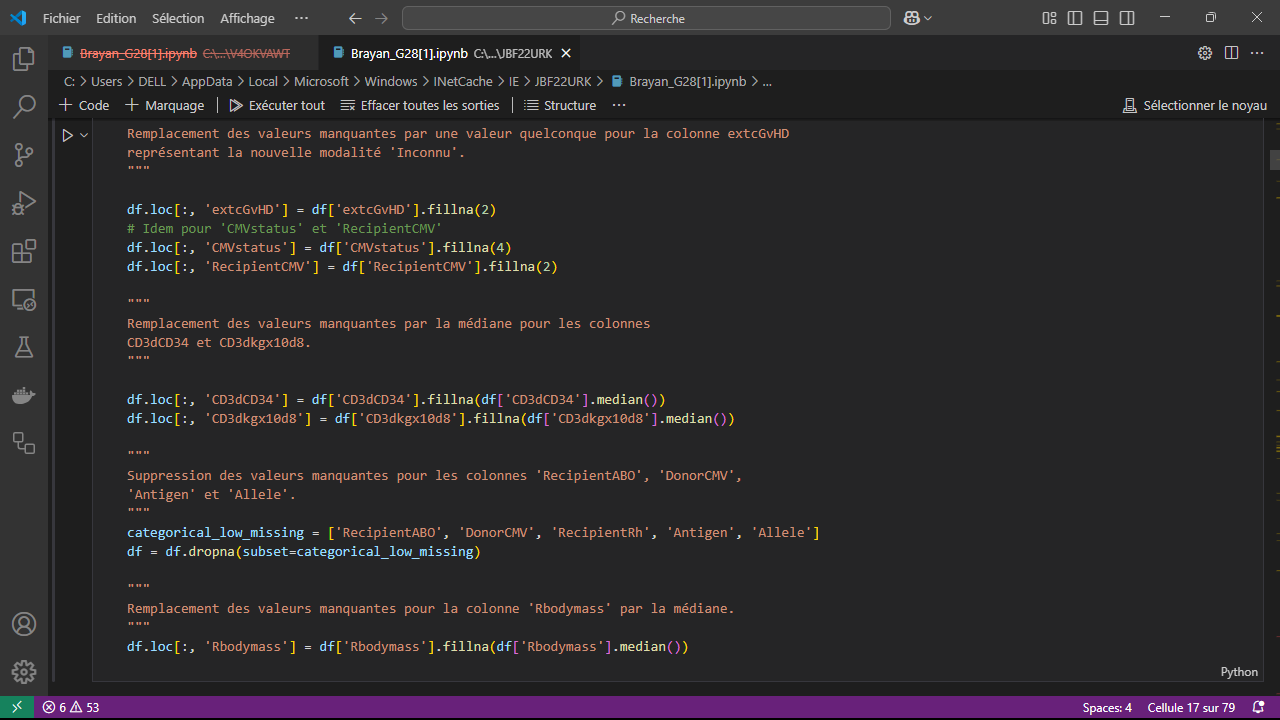
Here we notice that some values are missing.

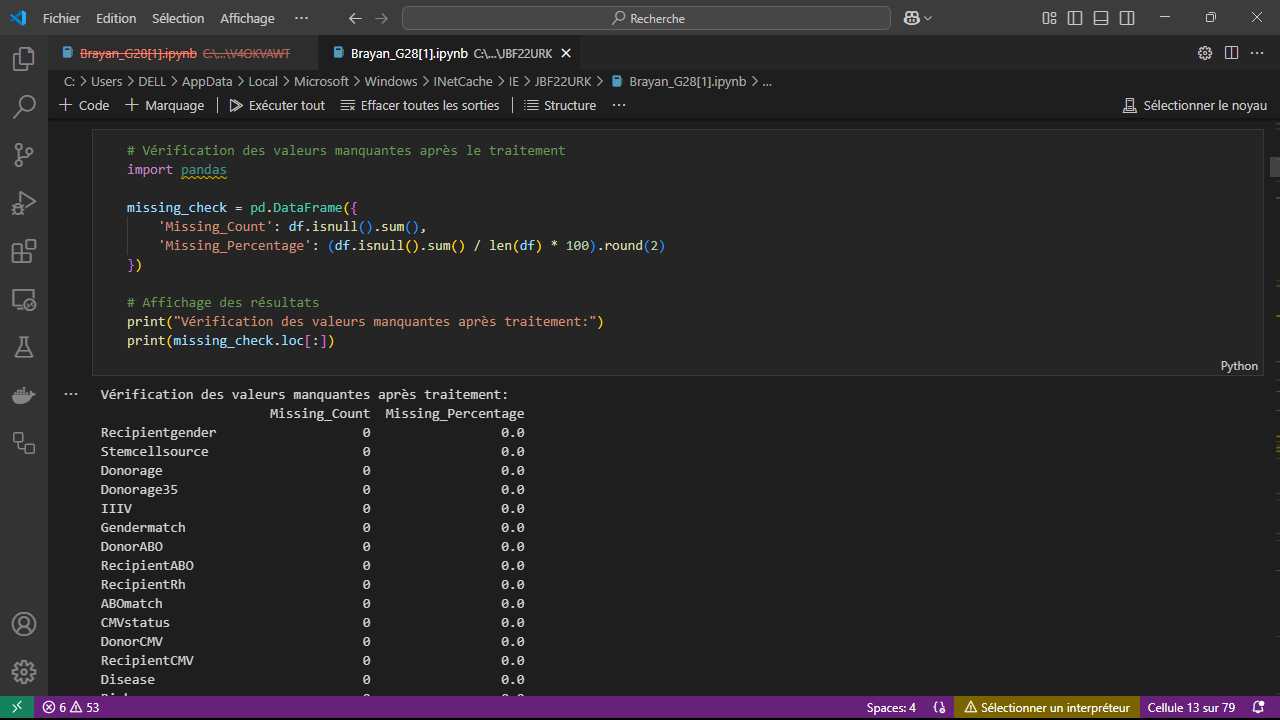
Thus we propose the following recommendation.

**Removing rows** with missing values for categorical variables with high percentage of missing values, while missing values in numerical variables will be replaced by the mediane.

**Creating a new category** for categorical variables with a high percentage of missing values. This way, we can replace missing values wuth a new category named “**Unknowng**”.

**Inputing missing values** for numerical variables with high percentage of missing values using the **median**.





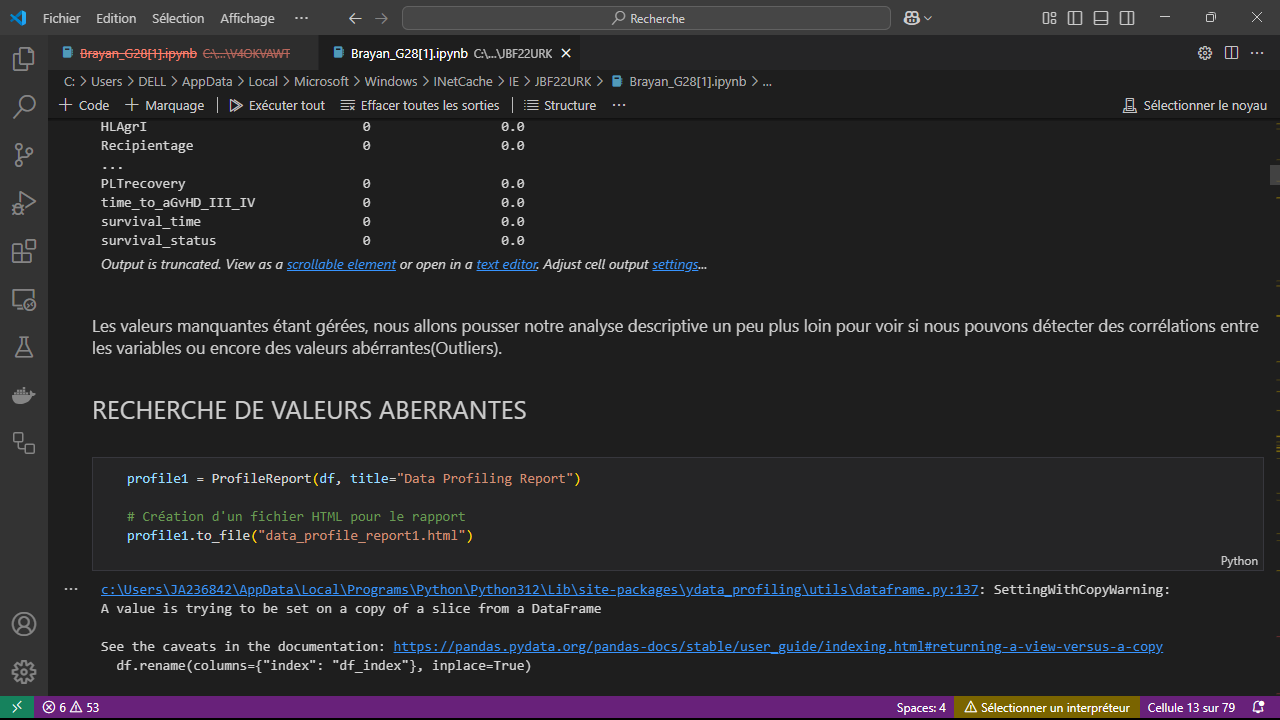
* **Outliers**

Outliers in a dataset are observations that differ significantly from the majority of the data, as in our case. These extreme values may be due to measurements errors, anomalies in data colllecton process or unusual but real phenomena.

We need to detect them in order to improve the data quality, optimize the model and identify interesting phenomena.

We noticed that some of those values where written in encoded form, so we replaced them by binary variables.

Other outliers where medically possible so we kept them.



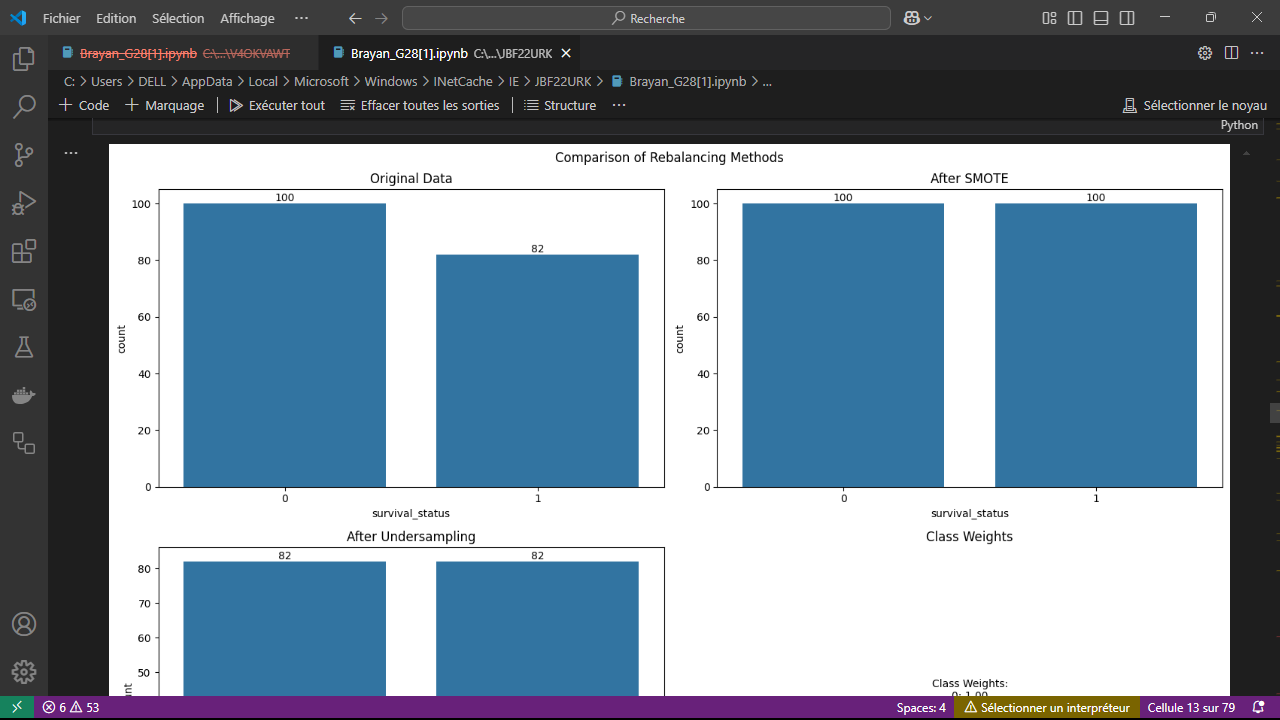
* **Class imbalance**

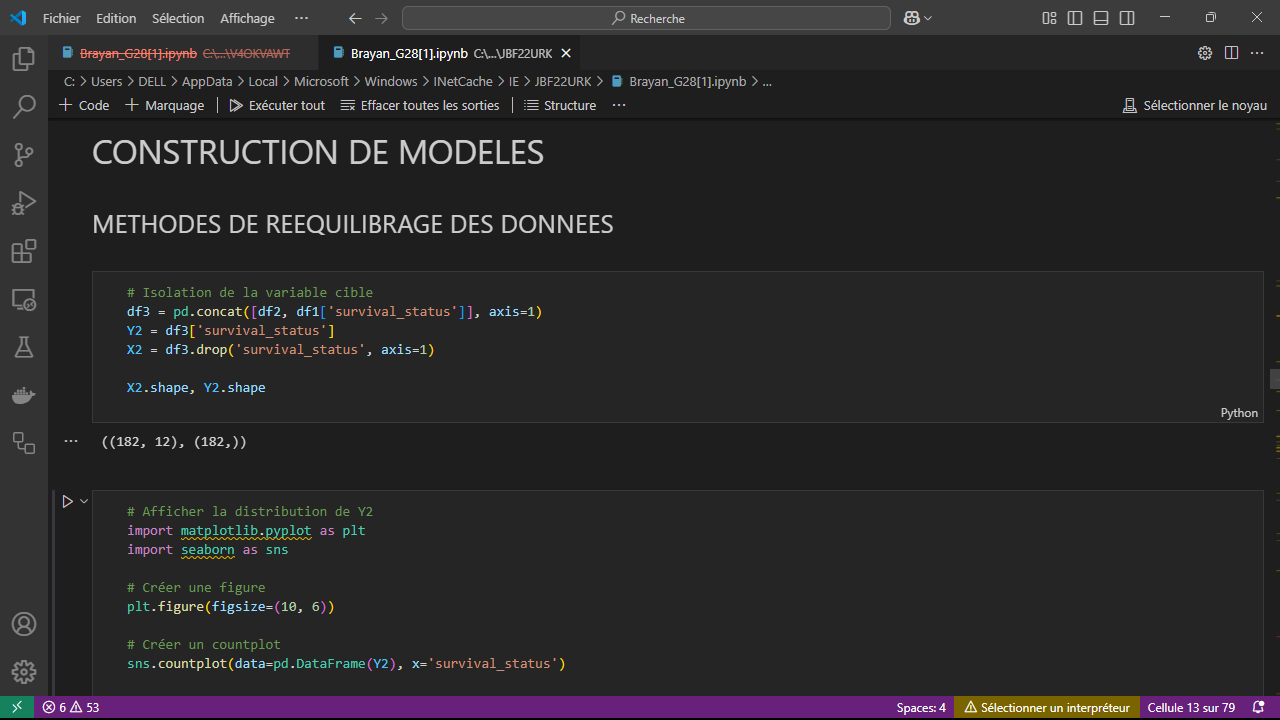
**Class distribution**

We noticed that the dataset was moderately imbalanced (54,95 percent survived, 45,05 percent not survived). To improve that situation, we will use rebalancing methods.

**Oversampling:** This technique consists of increase instances from the minoritarian class to balance the dataset. We used SMOTE that consist of generating synthetic observations instead of simply duplicating existing ones

**Undersampling:** The idea is to reduce the size of the majority class by randomly removing observations to achieve a better balance with the minority class. This prevents the model from being biased in favor of the dominant class.



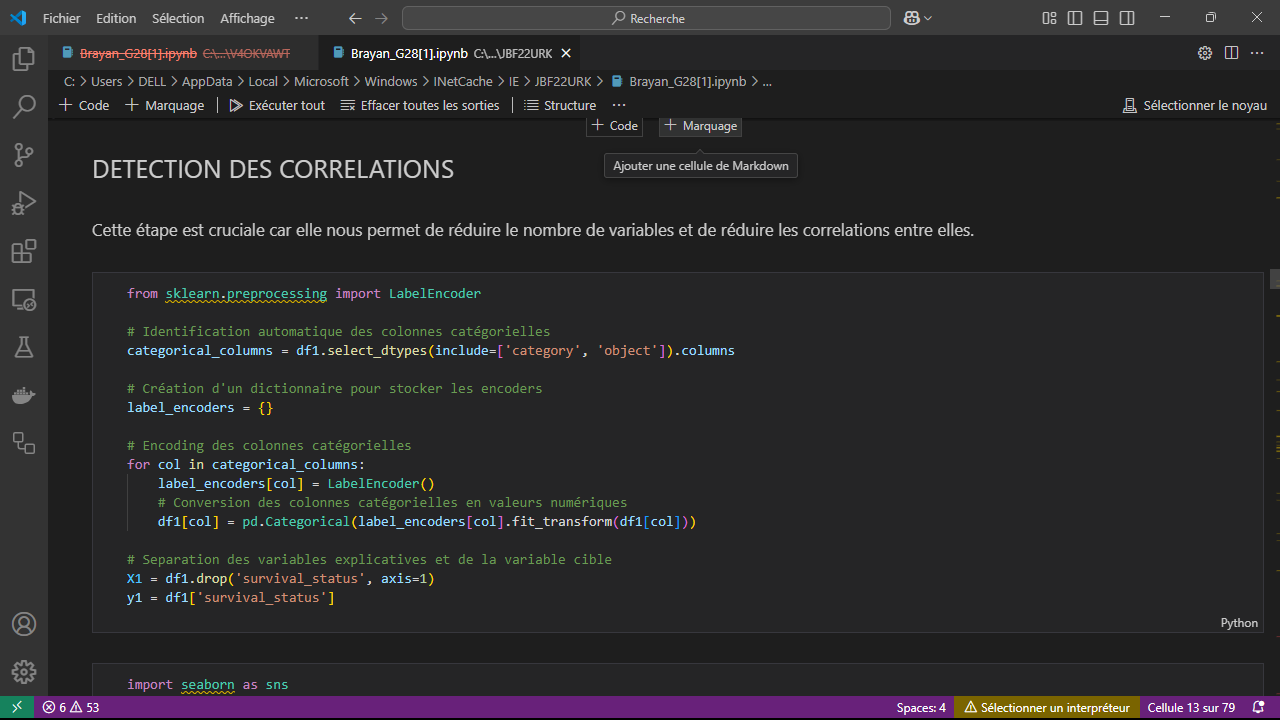


**Correlations detection**

This step is crucial because it allows us to reduce the number of variables and reduce the correlations between them.

We noticed that a huge part of variables are related each other. That is normal because those variables are linked. We just need to reduce the number of variables and keep most important of them.

15 principal components represent approximately 80% of the variance. We therefore used these 15 components for the rest of the project. To do this, We retained the most representative variable of each of the 15 principal components to build our new database.



1. **ML Models**
   * + - 1. **SVC (Support Vector Classifier)**

**Principle:** Based on Support Vector Machines, this model aims to separate classes by finding an optimal hyperplane that maximizes the margin between points of different classes. It can use kernel tricks to handle nonlinear data.

**Advantages :** Very effective for small , well-separated datasets. Can also model complex relationships using kernels.

**Disadvantages:** Can be slow for large Datasets. Is sensitive to hyperparameter choices and kernel selection.

* + - * 1. **XGBoost (Extreme Gradient Boosting**

**Principle:** It is an advanced version of Gradient Boosting, where trees are trained sequentially to correct the errors of previous trees Each new tree focuses on the residual errors of the previous one.

**Advantages:** Very efficient on complex and large dataset; supports parallel optimization, making it faster than other boosting methods.

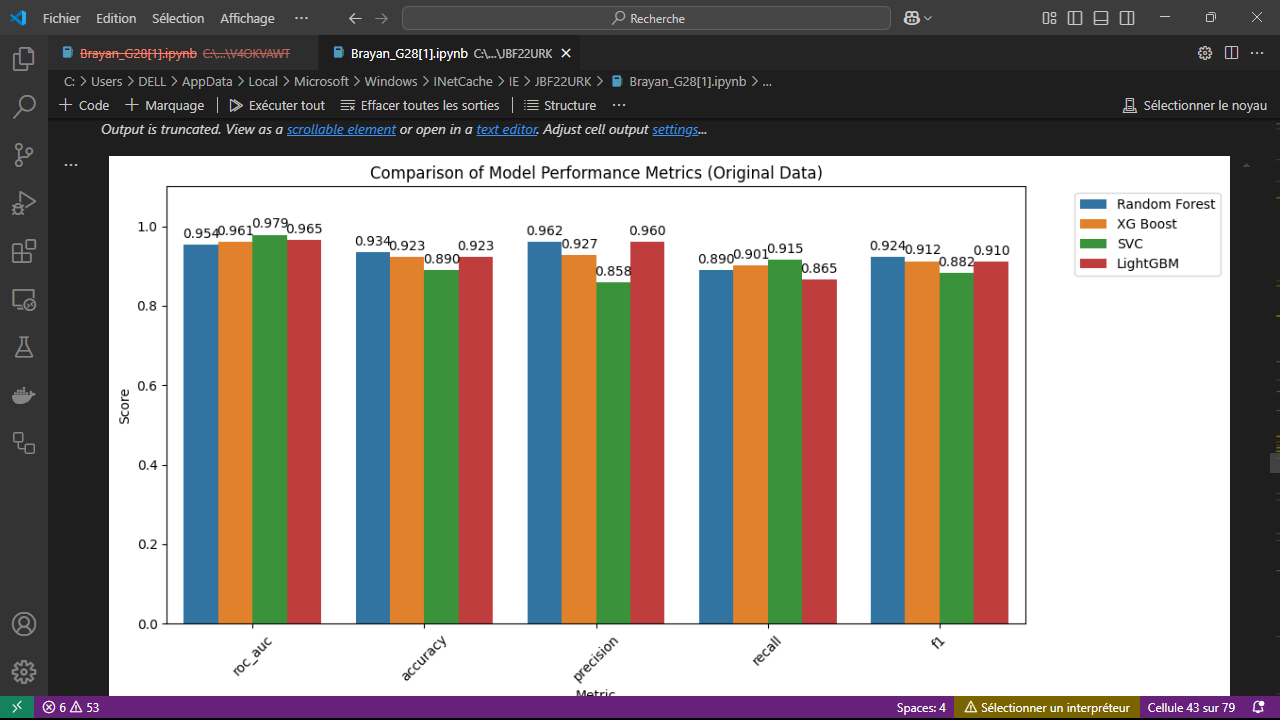
**Desadvantages:** Can overfit if not properly tuned; more complex to configure than random forest

* + - * 1. **Random forest**

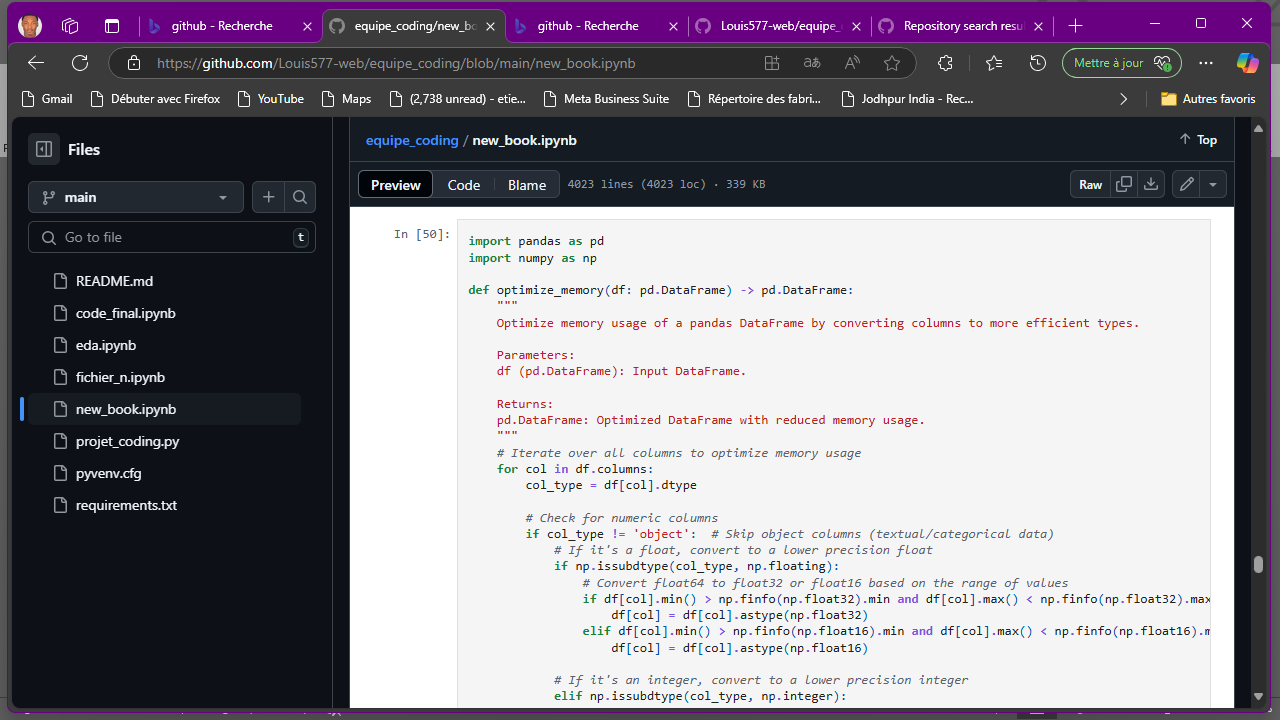
**Principle:** It is a decision tree set (a forest of randomized trees). Each tree is trained on a different subset of the data and the final decision is obtained by majority voting or averaging.

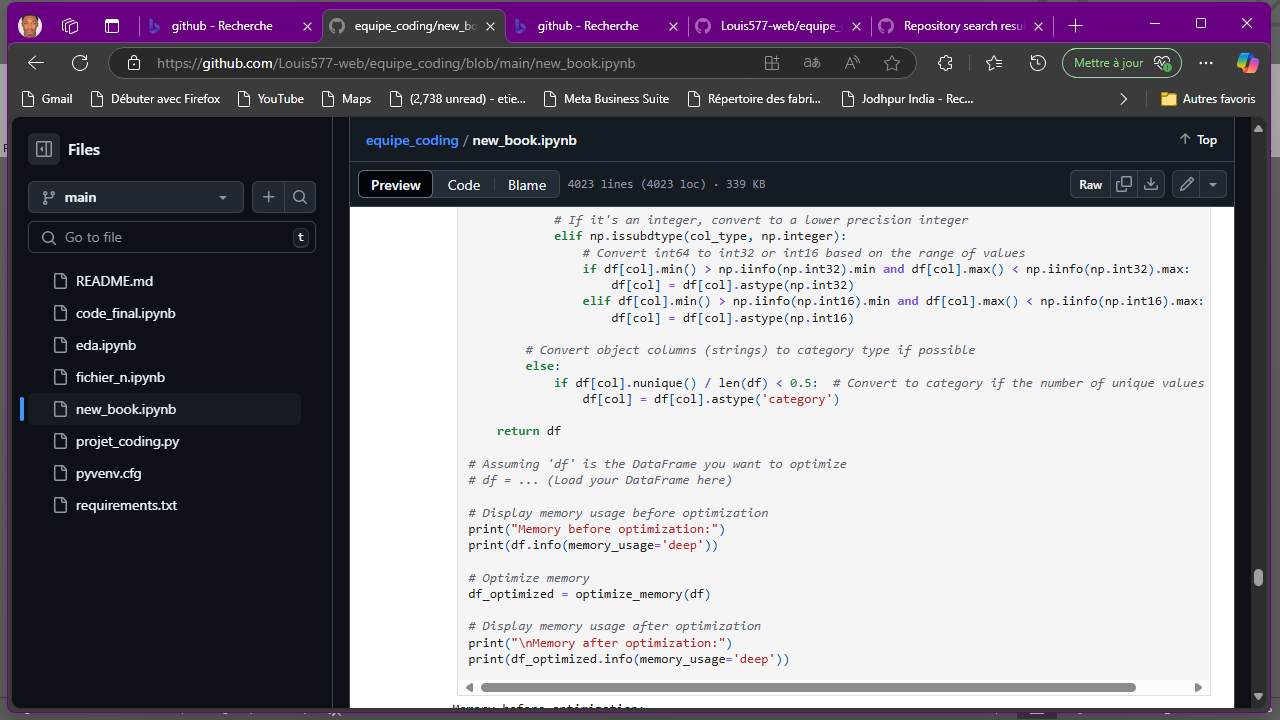
**Advantage:** Reduce the risk of overfitting; handle noisy data and missing values well;works well on medium to large datasets

**Disadvantages:** Slower than simple models; less interpretable than individual decision trees

****

1. **Optimizing memory usage**

****

****

1. **Interface development**

**Importing Libraries**

To set up our plateform, we used different libraries, namely:

* **Streamlit**: To create a user interface
* **Pandas** and **Numpy**: To manage data
* **Joblib**: To load pre-trained machine learning models

**Configuring the Streamlit Page**

For this purpose, we set the **page title** and **icon**. We also used a wide **layout** for better column display.

**Loading the models**

* We load a **data transformation pipeline** and a machine learning model.s
* **@st.cache\_resource**: We caught the model to avoid reloading it every time the user interact

**User interface**

The application is structured into three columns: The recipient information, the donor information and the clinical parameters.

**Prediction using model**

When the user clicks the prediction button, the model predicts whether the patient will survive:

Transforms data using pipeline.transform(input\_df).

Predicts the outcome using model.predict(input\_scaled)[0].

Displays results:

"The patient will not survive" if prediction == 1.

"The patient will survive" otherwise.

**Error handling**

except Exception as e:

st.error("Error during prediction. Please check input data.")

st.error(f"Error details: {str(e)}")

Captures and displays any errors that occur during prediction.

**Running the Script**

if \_name\_ == "\_main\_":

main()

Executes the main() function to start the application.