ESILV - Python for data analysis - project 2022

Presented by:

HACHEM Mohamad
BENYEMNA Hamza

Outline:

Topic :

Diabetes is a chronic (long-lasting) health condition that affects how your body turns food into energy. To put things into perspective, 34.2 million US adults have diabetes, and 1 in 5 of them don't know they have it. Diabetes is the seventh leading cause of death in the United States.

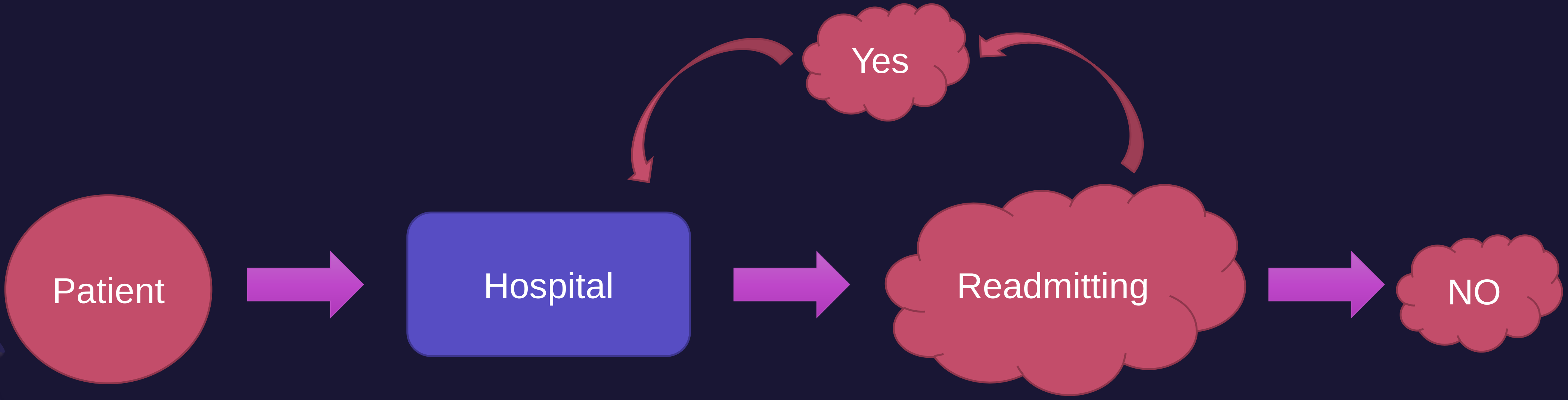
Is it possible to bring these numbers down by using data and Machine Learning ?

General context:

In our problem we will study the trait of being readmitted after treatment of his/her diabetes.

Why is such point so important ?

- It let us see if the treatment was valid for the patient.
- It will make us better at predicting who are the people who are more likely to be readmitted; therefore, be better prepared.



The goal of the project:

This work need skills in:

- Data cleaning.
- Data visualization.
- Machine Learning.

We must get a dataset as clean as possible in order to provide the best model. Furthermore, we must make many tests with different models. After this, we must compare our results to find out the best one.

Now that we know what are we trying to study exactly, so what is our goal ?

It is to predict as accurately as possible how likely is it for a patient to be readmitted. In our problem there are 3 states for being readmitted.

NOT readmitted.	Readmitted before 30 days	Readmitted after 30 days

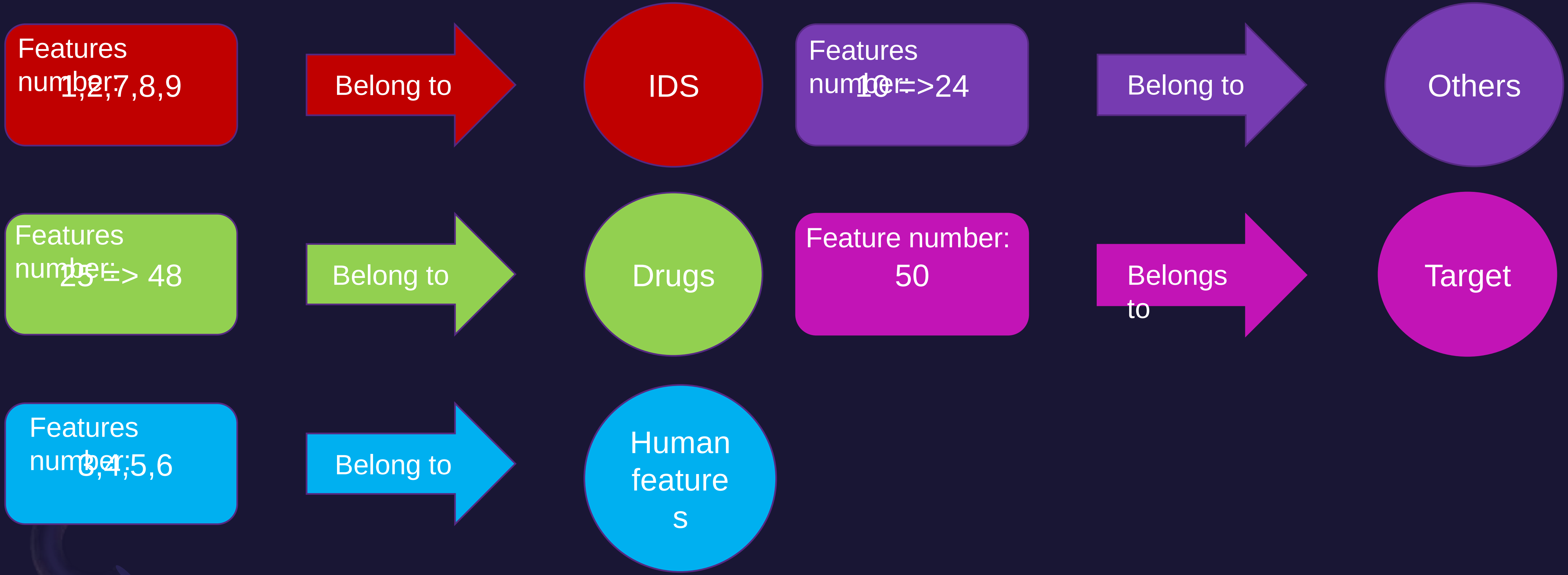
Presenting the dataset (1/2):

Our dataset contains originally 50 features sited as follow with color coded.

Feature_ID	Feature name
1	encounter_id
2	patient_nbr
3	race
4	gender
5	age
6	weight
7	admission_type_id
8	discharge_disposition_id
9	admission_source_id
10	time_in_hospital
11	payer_code
12	medical_specialty
13	num_lab_procedures
14	num_procedures
15	num_medications
16	number_outpatient
17	number_emergency
18	number_inpatient
19	diag_1
20	diag_2
21	diag_3
22	number_diagnoses
23	max_glu_serum
24	A1Cresult
25	metformin
26	repaglinide
27	nateglinide
28	chlorpropamide
29	glimepiride
30	acetohexamide
31	glipizide
32	glyburide
33	tolbutamide
34	pioglitazone
35	rosiglitazone
36	acarbose
37	miglitol
38	troglitazone
39	tolazamide
40	examide
41	citoglipton
42	insulin
43	glyburide-metformin
44	glipizide-metformin
45	glimepiride-pioglitazone
46	metformin-rosiglitazone
47	metformin-pioglitazone
48	change
49	diabetesMed
50	readmitted'

Introducing the Dataset (2/2):

Now that we seeing our features, we will classify them by IDS according to their functionality and they are color coded as the previous dataset.



Our solution:

Now that we understood more about the topic, and we saw the dataset, we are working with let’s see how did we proceed to solve this issue.

The method used has 2 main parts:

1. First part includes data cleaning and visualizations.
2. Second part includes Machine Learning models.

Data cleaning:

- To be able to work with ease with our dataset, we had to start by data cleaning and so we did the following:

1. Removed the duplicated entries from our dataset (from 101766 to 71518 rows which represents **30%** of the dataset).

2. Removed any useless feature. Some columns such as the weight were full of ‘?’ with only 2853 rows with no missing data (**3%** of the dataset). Moreover, we removed all the drugs that has less than **5%** impact.

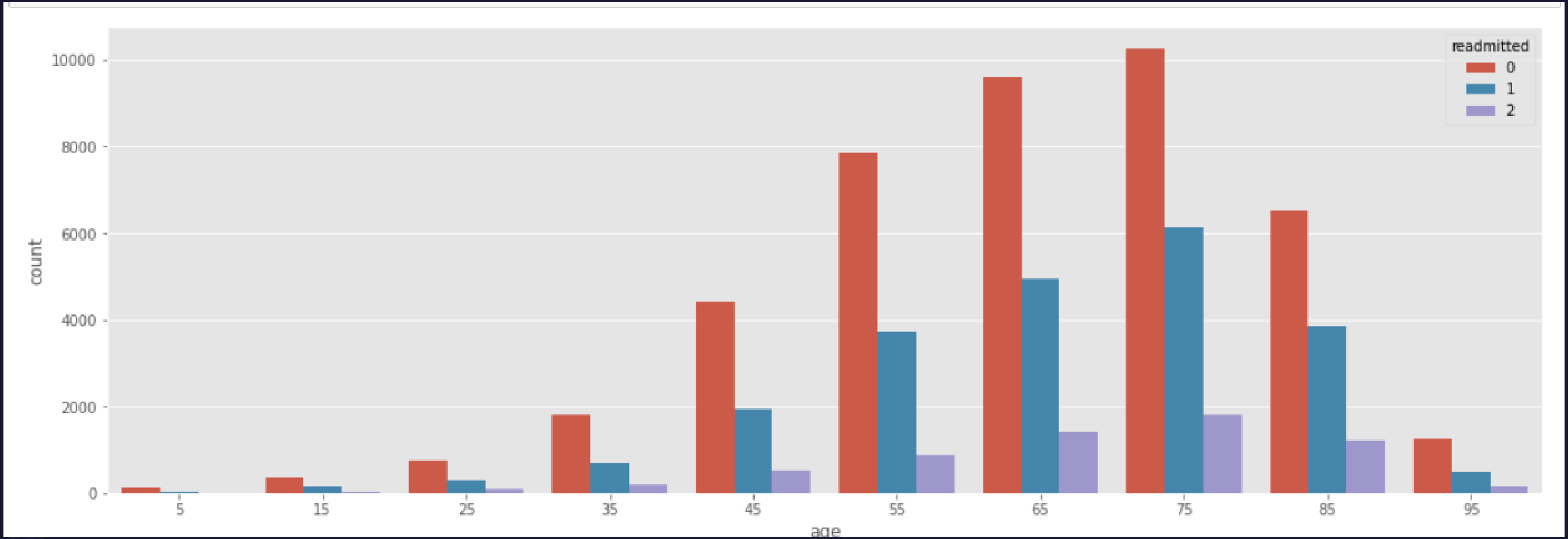
3. Transforming the dataset into numeric values. This helps us to make correlations and studies that require to compare the different features.

This way, having a lighter and homogeneous dataset with no missing data simplifies the work and let us focus on the essential aspects of the predicting work.

Data visualization (1/3):

This work of visualization helps us to point at important features. The aim is to keep essential features as predictors.

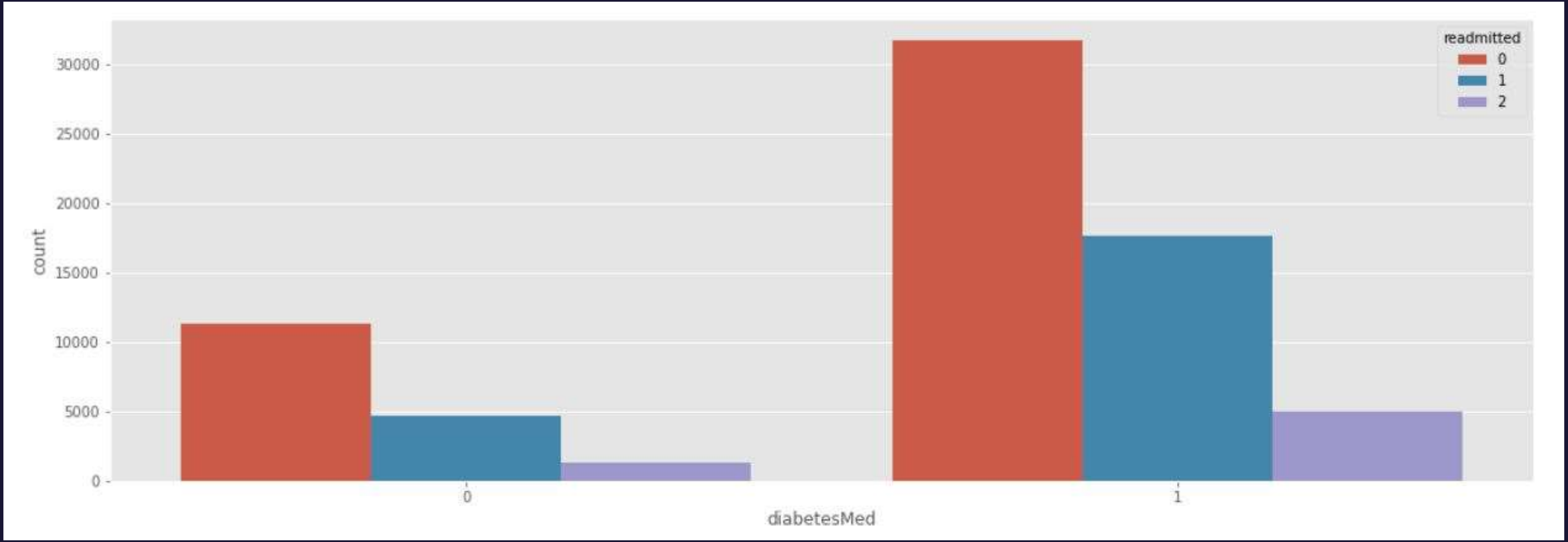
Plot of age:



As we can see, patients around 75 years old are more concerned by readmission.

Data visualization (2/3):

Plot of diabetes medications:



We can say that readmitted has the same distribution for the people with or without diabetes medications in terms of proportions between no readmission, < 30 days and > 30 days. But it is interesting to add that the patients that have medications are more concerned by readmission than the ones with no medications (**3x** more).

Data visualization (3/3):

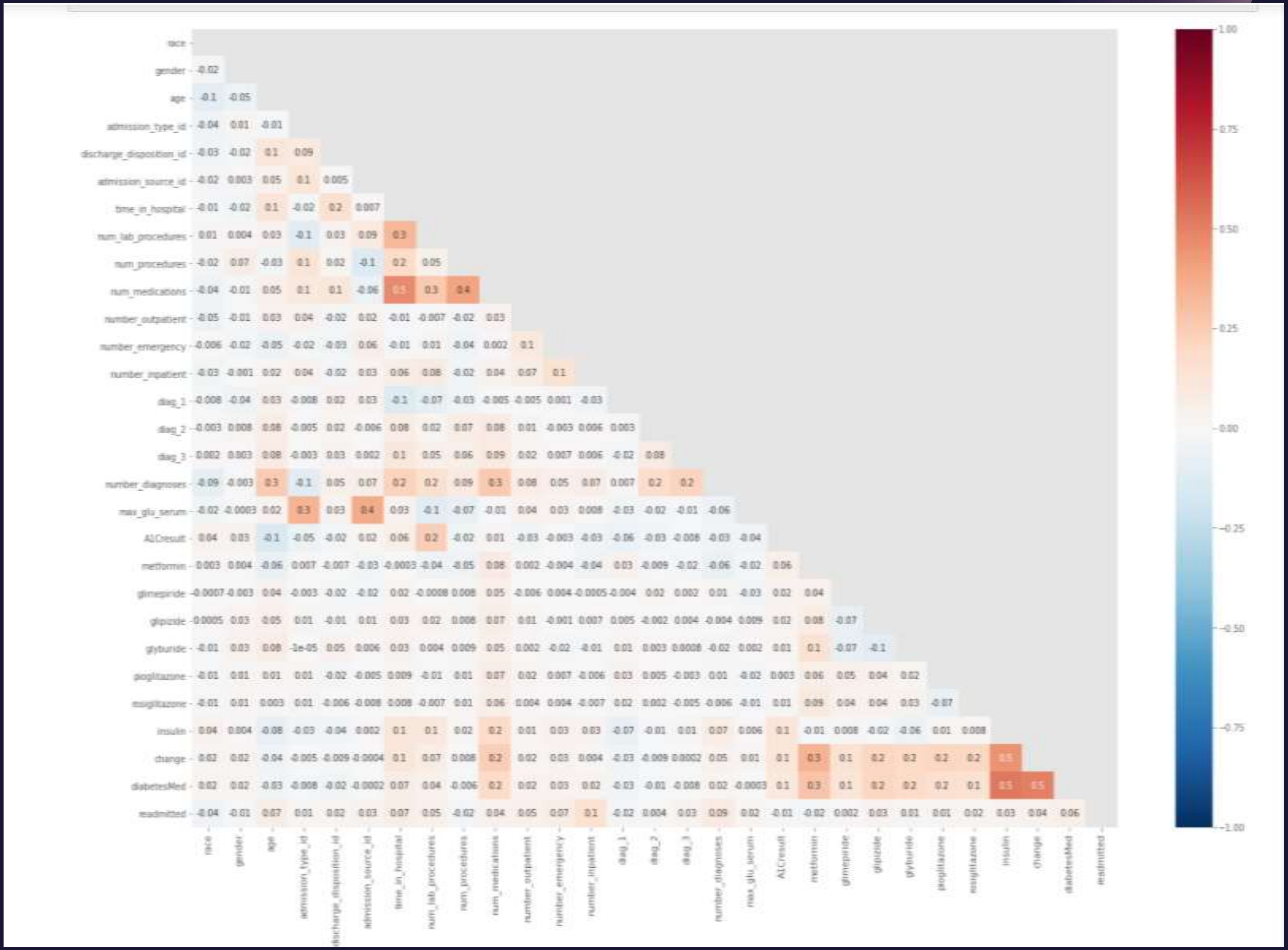
Plot of correlation:

In the following plot we can see the correlation between all the features. And the reason to do so is to understand which features should be removed while we are training our dataset.

The most interesting ones are :

- **number_inpatient** (0.1).
- **number_diagnoses** (0.09).
- **number_emergency** (0.07).
- **age** (0.07).
- **diabetesMed** (0.06).

We notice that there isn't any strong correlation with the target (no one has more than **0,1**).



Machine Leaning:

Now that we cleaned our dataset, we are ready to creating some machine learning models. In order to do that, we need to:

1. Identify the model we want to work on.
2. Split the data into training and testing set.
3. Scale the data.
4. Run the machine learning algorithm.
5. Proceed to validation test.
6. Try to improve it.

Explanation of Machine Learning:

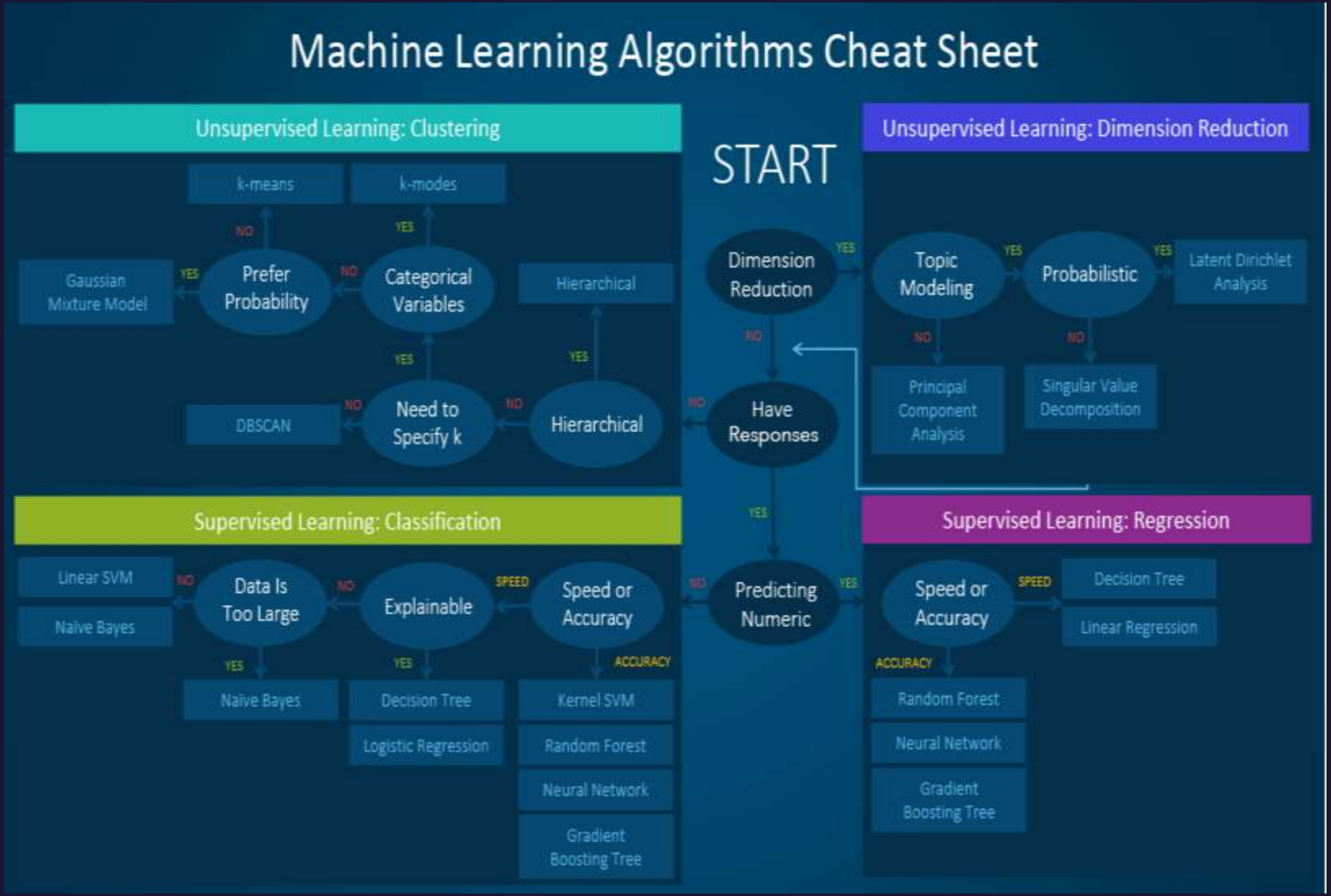
Schema of different possibilities:

Since our problems deals in knowing whether our patient will be readmitted or not. Then we are dealing with a 3-class problem with a labelled dataset. Indeed, it is a **supervised** model of **classification**.

So, according to this schema, we picked out 3 methods:

- **Naïve Bayes**.
- **Random Forest**.
- **Gradient Boosting Tree**.

We chose **Naïve Bayes** model in order to test the speed of this method and the other two because we aim for accuracy.



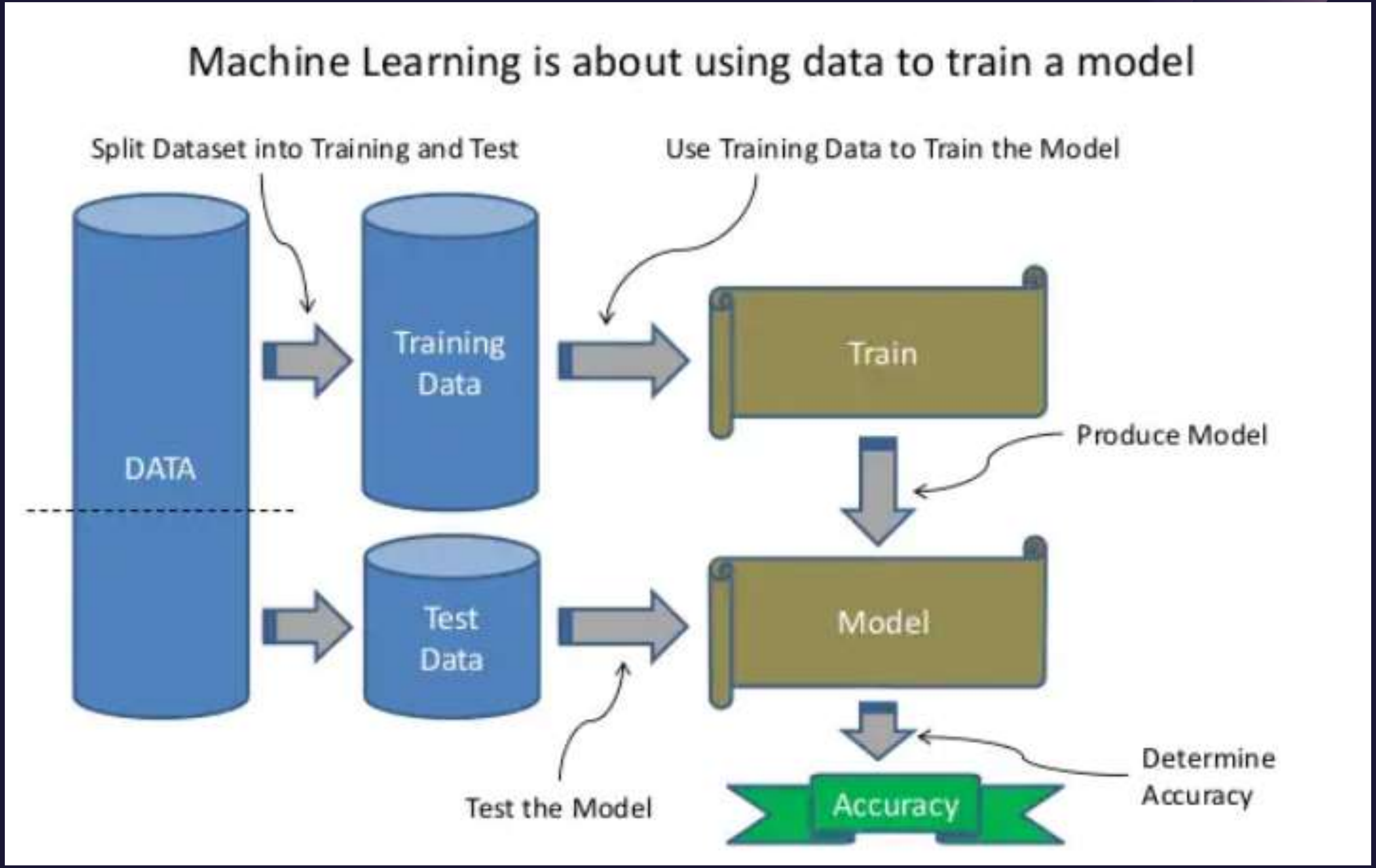
Splitting the data:

To be able to start our model, we need to first split our data into a training and a testing set to work on it. There is different ways of doing so. Usually, we split this ways:

- **80%** for training / **20%** for testing.
- **~70%** for training / **~30%** for testing.

We decided to use **1/3** of the dataset to test and the other **2/3** to train.

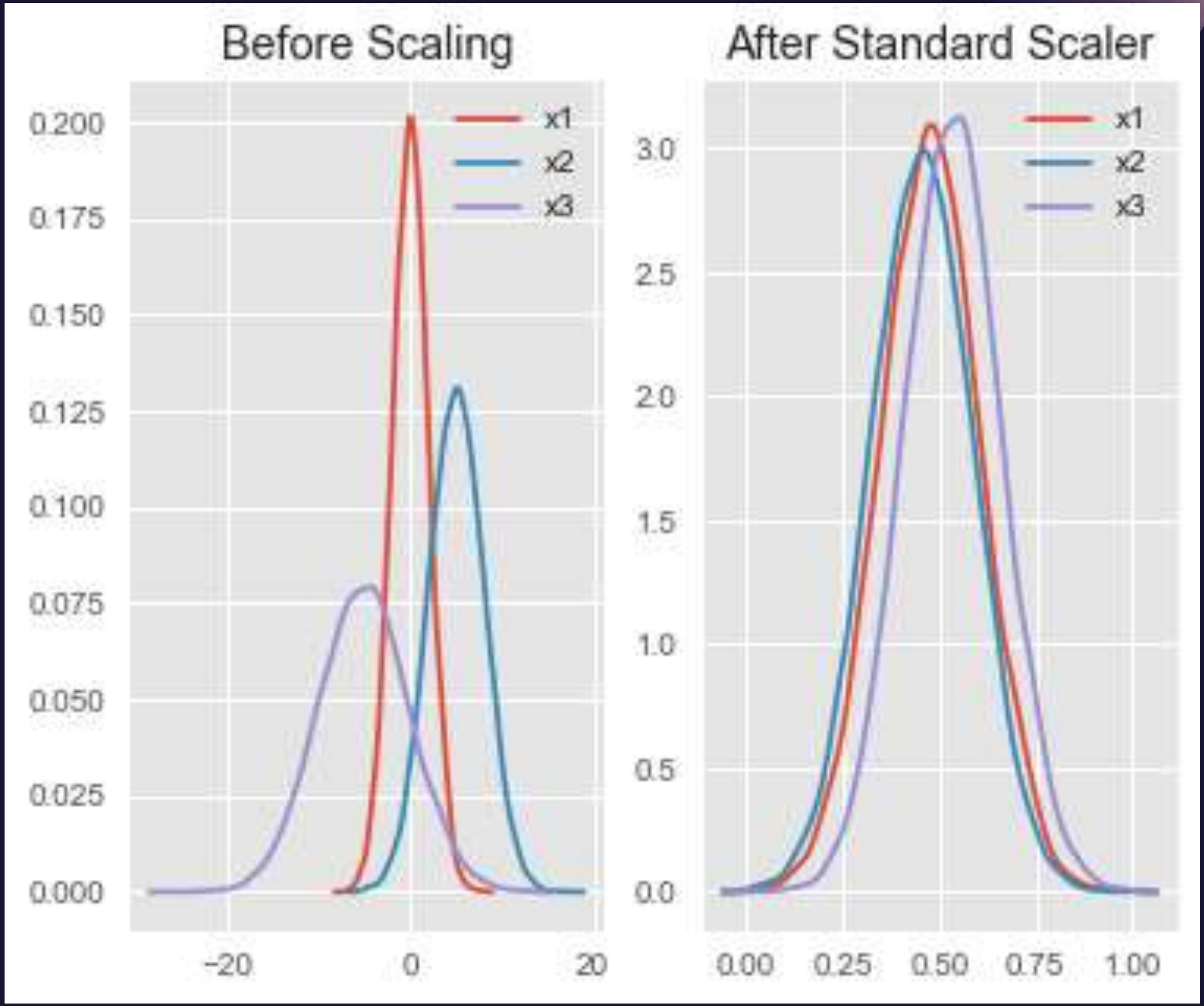
After that, we split the testing set into 2 parts (**50/50**) for the validation test.


Machine Learning is about using data to train a model
Split Dataset into Training and Test
Use Training Data to Train the Model
DATA
Training Data
Test Data
Train
Produce Model
Model
Test the Model
Determine Accuracy
Accuracy

Scaling the data:

Scaling the data makes building a machine learning algorithm way easier and more accurate so it is very important to do that.

As seen in this plot, the data is heterogeneous before the scaling. The **Standard Scale** method that we used bring all the data to the same range. This way, the variance of the data will not affect our work on the accuracy through many attempts.



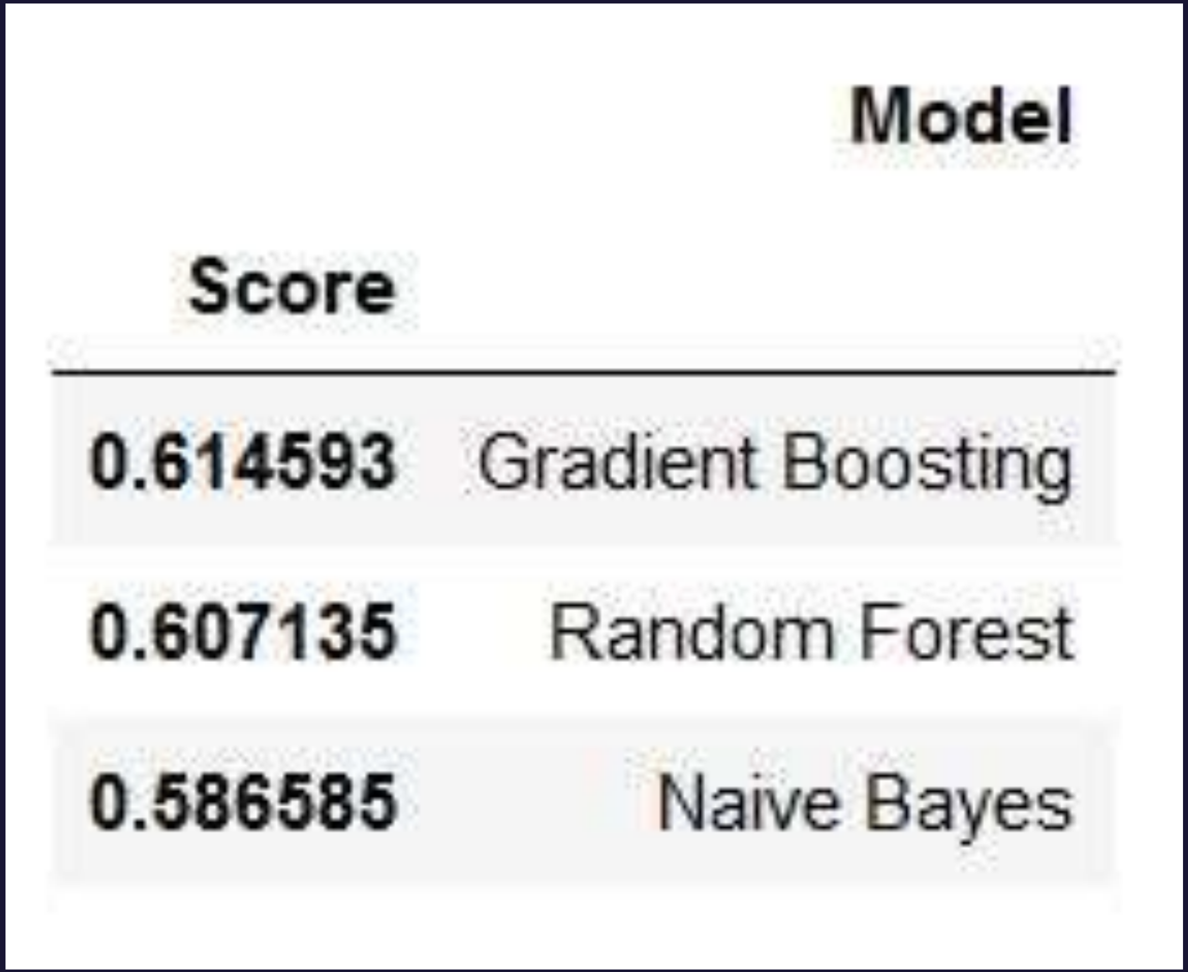
Different Machine Learning models:

We can see the accuracy rate for each model that we applied on the dataset. The accuracy score obtained is around **60,5-62,5%**.

There are 3 important outcomes to talk about:

- **Naive Bayes** is the fastest and the one.
- **Gradient Boosting** is the best model.

Based on that, we decided to keep working on the **Gradient Boosting** model (since it provided us with the highest accuracy) but before that, we tried to improve these results in order to be sure of our model choice.

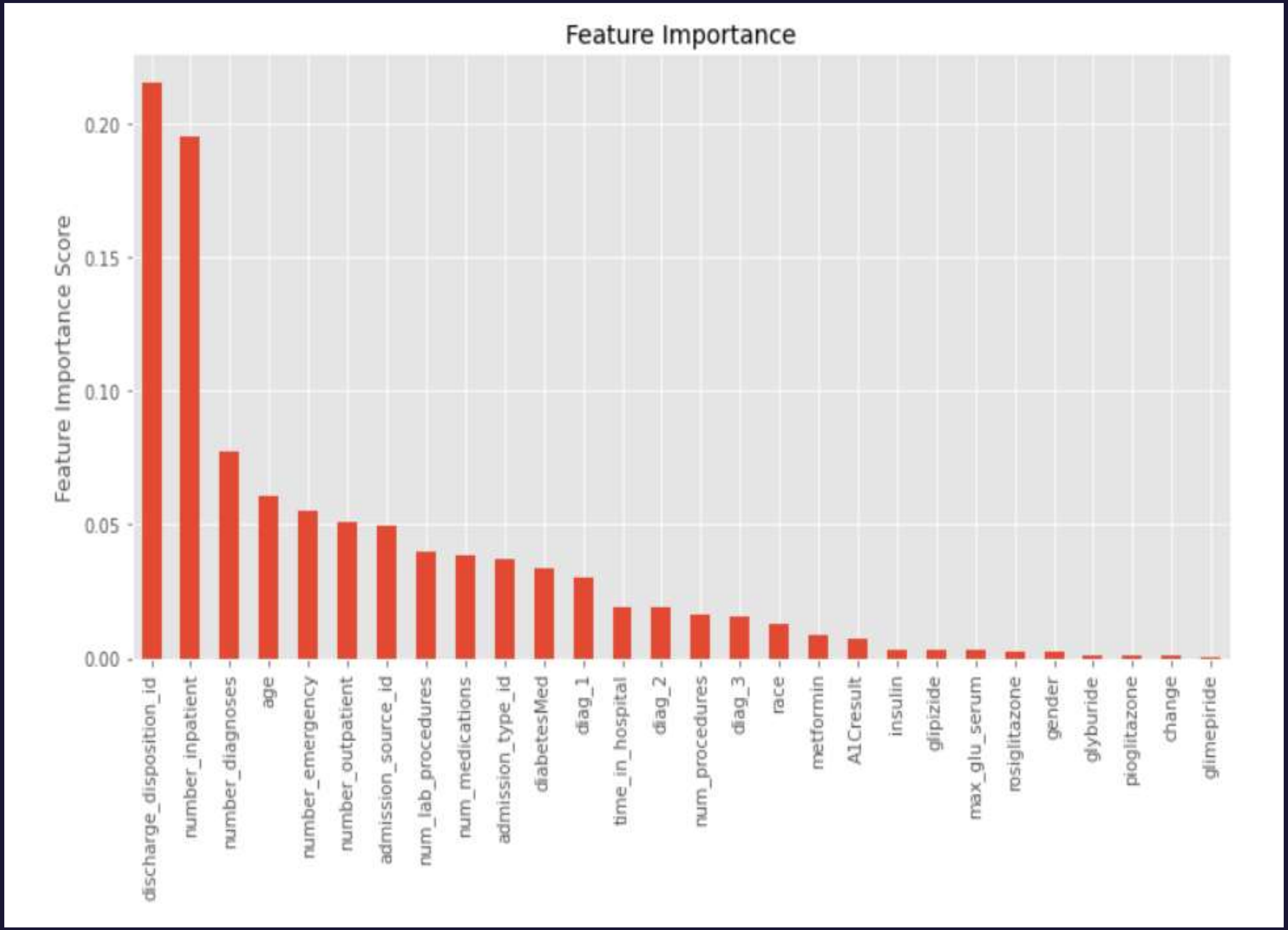


	Model
Score	
0.614593	Gradient Boosting
0.607135	Random Forest
0.586585	Naive Bayes

Variable importance:

This bar plot shows us the importance of the features on the model's accuracy. We can say that **discharge_disposition_id** and **number_inpatient** are the most important features with a score about **0,20**.

We noticed that the features with less than **0,02** importance score do not affect the accuracy that much.



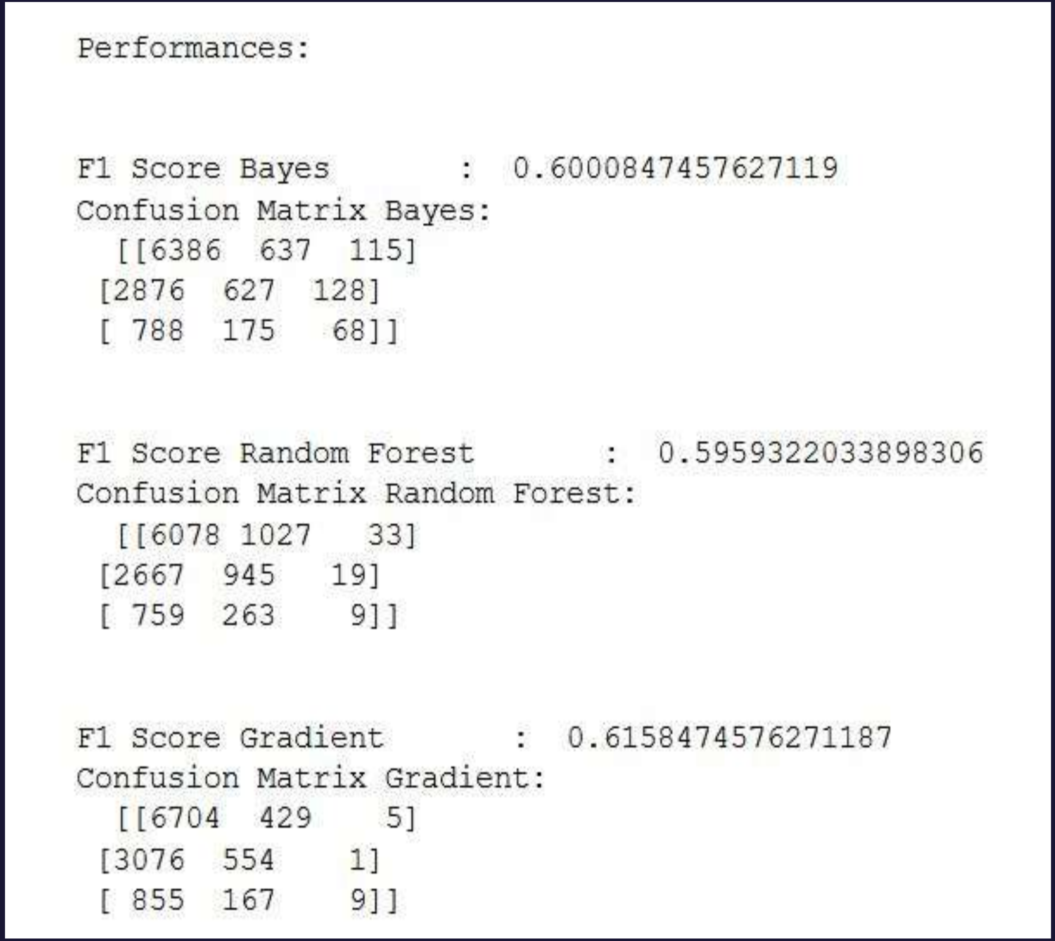
Improvement of the model:

Indeed, we first tried our models on the dataset with different combinations of features according to the **importance score.**

We started choosing from 1 (with the highest importance) all the way to 27 predictors (total list of features). We noticed after many attempts that:

- The less we have features(less than 5), the more **Naïve Bayes** model is the best. **Random Forest** and **Gradient Boosting** models shows some limits with less data.
- The more we have features, the more **Gradient Boosting** has the best accuracy.
- Running time and accuracy increase deeply with the number of predictors selected.

Besides this, we proceeded to validation test that shows confusing matrix and accuracy score. We still in an accuracy rate around **60%-62,5%**. Therefore, the improvement is not obvious, that is why we tried the **Gradient Boosting Model Tuning**.



```
Performances:


F1 Score Bayes         :  0.6000847457627119
Confusion Matrix Bayes:
  [[6386  637  115]
 [2876  627  128]
 [ 788  175   68]]


F1 Score Random Forest        :  0.5959322033898306
Confusion Matrix Random Forest:
  [[6078 1027   33]
 [2667  945   19]
 [ 759  263    9]]


F1 Score Gradient       :  0.6158474576271187
Confusion Matrix Gradient:
  [[6704  429    5]
 [3076  554    1]
 [ 855  167    9]]
```

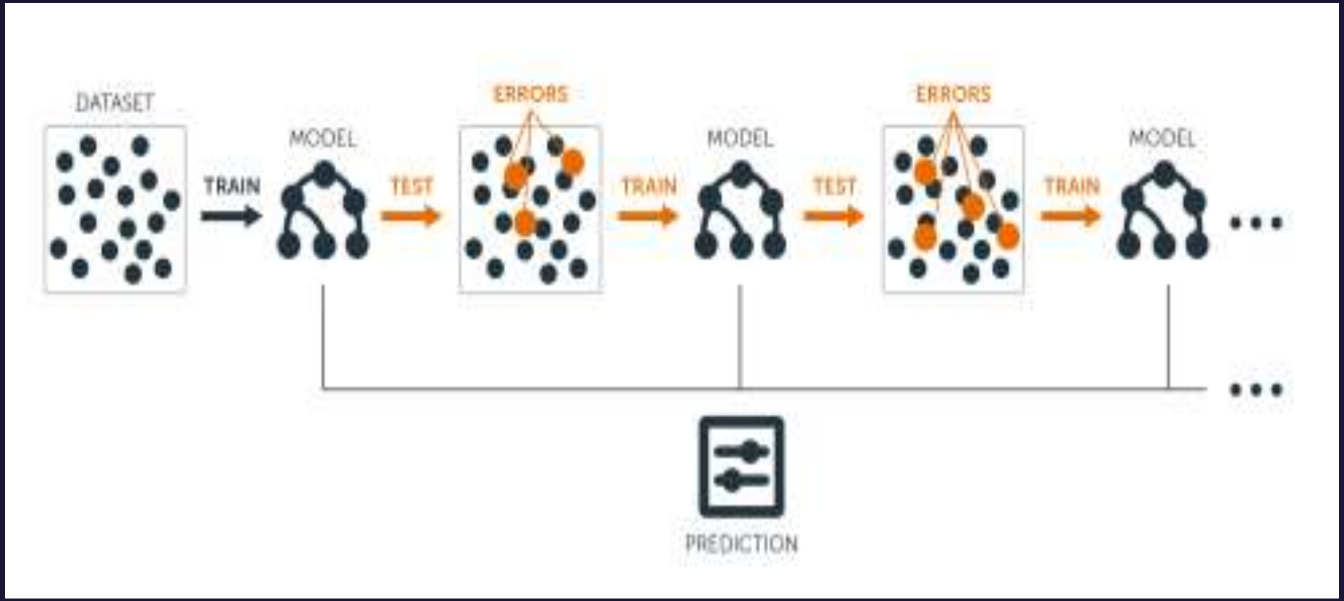
Gradient Boosting Model Tuning:

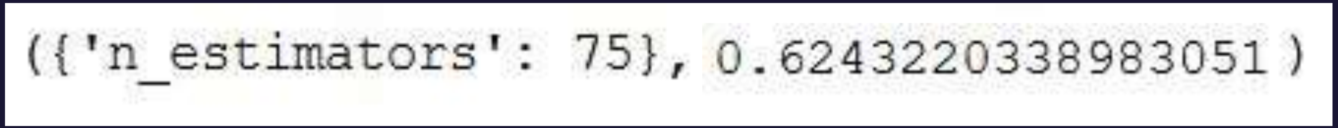
Based on the **Gradient Boosting** model that we chose; we applied a **hyper parameters tuning** method. As explained in the schema, this method is based on the crossing of different results in order to create a more powerful model.

So, we first initialize the parameters according to basic criteria to be able to use a **grid search** method.

This method is based on intuition and testing many times in order to find the best **n_estimators** parameter by fumbling. We noticed that the more the parameters are high, the more the **grid search** take time to return a result.

This way, we could obtain the best score.





```
({'n_estimators': 75}, 0.6243220338983051)
```

Conclusion:

By simply taking data from diabetes patients and treating them, we can help them out by predicting whether a patient will need to be treated again any time soon

Through our Machine Learning model based on **Gradient Boosting** which is boosted by the **Model Tuning** thanks to a grid search, we were able to have an accuracy that reached **62,5%.**