AUTOMATIC MODEL SELECTION REPORT

Opioid abuse in the United States of America

Ivanna SAVONIK

Florine PRITZY

Laura THONNEAU

January 5th, 2021

Contents

1. Introduction

Since 2010, the United States has been facing what we call an "opioid epidemic" and having strong consequences. Indeed, opioids are substances that interact with receptors on nerve cells in the brain and body. Opioid pain relievers are medically used and prescribed by doctors for pain relief or even anaesthesia. They are generally harmless when the person takes it for a short time, but because of the euphoria that it produces in addition to pain relief, it can be misuse. Some can take it in higher quantity, during a longer period or even without any medical prescription.

These situations can lead to addiction, overdoses or even death. In 2018, 46,802 people died from an opioids overdose in the United States. And the opioids (mainly synthetics), are the principal cause of drug overdose deaths, with 69.5% of overdose deaths in 2018. And the sources can be medical as well as illegal.

However, we saw that opioids abuse can be treated with effective medications, like methadone, buprenorphine and naltrexone that already recover many people from their opioid addiction. An overdose can also be reversed by taking a certain drug quickly after.

We believe that predicting opioid abuse could allow insurance companies to identify addicts, so that they receive treatment. It could also allow targeted prevention of opioid-related risks and thus spare a lot of lives. Especially since these abuses keep victims, who are often precarious, away from employment, which is probably intensified by the impact of the COVID pandemic on both social safety and health care delivery systems.

And indeed we were able to read some papers on this subject, which we will talk about right after. Thus in this project, we'll try to use several Econometrics and Machine learning models, in order to get a performant model that predicts opioid abuse.

Main results : Overall, our models have a good performance. But the best ones, in terms of AUC score and minimum number of false negatives, are the Artificial Neural Network, the SVM, the Logistic regression with l1-penalty and the Random Forest.

This report is organized as follows: *Section 2*, presents some previous studies and techniques that have been used in opioid use disorder. *Section 3*, will present a description of our dataset as well as the theory of the methods used. Then, *section 4*, presents the results of our models. Finally, *section 5*, presents our comments and conclusions.

2. Related literature

A large growing literature exists on the subject of opioid use disorder, in our project we describe those, concentrated on the use of Machine Learning techniques in order to predict opioid use abuse. Hasan et al. (2019) use multiple Machine learning techniques for prediction of opioid use disorder and compare their performance and find that tree-based (Decision Tree, Random Forest, Gradient Boosting) methods outperform Logistic regression, achieving the best recall of 0.97 using Random Forest Classifier after Recursive feature elimination. Hunker (2019) applied Logistic Regression. Decision tree, ANN and SVM, selecting the features using the Gini index and L1-normalized Penalized algorithms and obtains the accuracy of about 99% using the SVM Classifier. Lo-Ciganic et al. (2020) had the objective to improve prediction of incident opioid use disorder diagnosis among Medicare beneficiaries with ≥1 opioid prescriptions. In their study they used : elastic net (EN), random forests (RF), gradient boosting machine (GBM), and deep neural network (DNN) and found that all of them had similar predictive performance (the performance was evaluated based on C-statistic). However, the elastic net needed less predictors to achieve the same results. The C-statistic was higher than 0.87 for all of the models.

Unlike previous studies, we use different dimensionality reduction and feature selection techniques, such as Lasso and PCA, we also use some other classifiers including LDA and KNN. We as well evaluate the performance of the unsupervised machine learning algorithms on our data. As it was done in previous studies we apply SVM, Logistic Regression, Neural Networks, Artificial Neural Network and Random Forests.

3. Materials and Methods

3.1. Data presentation

We will work, in this project, on a United States insurance database, dating from 2020, which includes 92 features and more than 160,000 observations. We found this database from a data scientist and psychologist working in Chicago. As we know, the opioid crisis in the USA has reached increasingly tragic proportions, accounting for two thirds of the 72,000 overdose deaths of 2017. We investigated whether US health insurance claims data could perform to identify people with evidence for opioid abuse. The data describes a set of characteristics about patients who have had an accident leading to an insurance claim. We have two continuous variables, the age of the claimant and its weekly wage. The rest of the variables are categoricals. We have for instance, the marital status of the claimant, the state in which the claimant resides, the employment status flag, psychiatry or neurosurgery payment flag and the target variable “Opioids used”. After removing the missing data, our database contains 120,661 observations. For more details about variables, see **Appendix A.**

3.2. Data pre-processing

It is important to know the pattern of missing data because if the MCAR or MAR hypothesis is met, there is no need to model the behavior directing the missing data for a variable because it is random. But if the data are MNAR, the cause of the missing data must be determined and therefore the missing data behavior must be modeled to better handle it. We, therefore, decided to analyze our dataset carefully and to know the

variables as well as possible in order to determine which approaches to use to deal with these missing data. We are now going to describe all the steps performed for the pre-processing of our database.

a) **Harmonization of missing information :** As a first step, we harmonized all missing data. Indeed, some variables were not coded in NaN but in " " or " X ". We then transformed these values.

b) **Variable deletion :** Next, we looked at the percentage of missing values for all variables. We made the choice to delete variables with a percentage of missing values greater than 50%. These are intractable because they are incomplete. This concerns the variables *"Accident Source Code"* and *"Max Medical Improvement DateID"*. We also removed variables that did not provide any information: the ID variables are unique to each insured, they do not introduce any additional information. We have therefore deleted *"Claim ID"*.

c) **Correlation between numerical variables :** We now investigate whether numerical variables are correlated with each other. The objective will be to measure the strength of the link between the variables (this link may be more or less complex). The correlation matrix is then used to evaluate the dependence between several variables at the same time.

	Accident DateID	Claim Setup DateID	Report To DateID	Employer Notification DateID	Industry ID
Accident DateID	1.0	0.99	0.99	1.0	0.0042
Claim Setup DateID	0.99	1.0	1.0	1.0	0.0071
Report To DateID	0.99	1.0	1.0	1.0	0.0069
Employer Notification DateID	1.0	1.0	1.0	1.0	0.0061
Industry ID	0.0042	0.0071	0.0069	0.0061	1.0

The result is a table, containing the correlation coefficients between each variable and the others.The rule is that variables are highly correlated when the correlation coefficient is greater than 90% (in the red part).

Four highly correlated variables are found: *"Accident Date ID"*, *"Claim Setup DateID"*, *"Report to DateID"* and *"Employer Notification DateID"*. These variables are all dates, created by the author of the database. We decide to delete the last three variables because they have a strong collinearity.

Between qualitative variables : Three variables reflect the same information: "*Claimant State*", "*Accident State*" and "*Benefit State*" correspond to the state where the insured lives, where he has had his accident and where he benefits from his compensation. We keep the variable that has no missing information: *"Benefit State"* and delete the others.

d) **Handling missing data :** There are two possible techniques for dealing with missing data. Either the missing lines are deleted or replaced. Each of these techniques has its disadvantages and advantages. Removing missing values is the simplest and most common technique. But one may have to delete too many observations, which may be important for the learning phase of Machine Learning algorithms. It is also possible to replace missing data in the database with artificial values.
After many queries and tests, we decide to delete the lines with missing values. Now, we have a database containing 120,661 rows and 81 variables.

e) **Extreme values :** We can observe extreme value for continuous variable but we cannot modify them because we don't have the data knowledge to support to handle them

f) **Categorical variable encoding :** For models we need to modify our categorical variables in order to have binary variables. Some variables have too many categories, so we have to modify them. For example, we have modified the *"SIC Group"* variable which corresponds to the professional social categories of policyholders who have had an accident. We went from a variable of 36 categories to a variable of 10 categories: (*Agriculture,Forestry,Fishing/ Mining /Construction/ Manufacturing/ Transportation/ Wholesale,Trade/ Retail Trade / Finance, Insurance/ Services/ Public Administration).*
We also create 4 variables : age and weekly wage squared and cubed.
Afterwards, we use one hot encoding in order to create dummies for each category of each categorical variable.

g) **Unbalanced database :** In medical data classification, we often face the imbalanced number of data samples where at least one of the classes constitutes only a very small minority of the data. It's our case, we have a minority of individuals who have abused opioids (11.39 % in the database). We decide to rebalance our data (only the train set) with the NearMiss method, because we want to see the predictive capabilities of our models in real situations. If we had modified our original database, we would not have taken into account the fact that we have an under-represented class (opioid consumers). This class is the one we are interested in. By using the **NearMiss method**, we apply an algorithm that can help in balancing an imbalanced data. The algorithm does this by looking at the class distribution and randomly eliminating samples from the larger class. When two individuals belonging to different classes are very close to each other in the distribution, this algorithm eliminates the individual of the larger class thereby trying to balance the distribution. We remove individuals who do not consume opioids, but who most closely resemble each other, so as not to lose important information. The train set therefore has 10,995 individuals in each category.

h) **Features standardization:** Taking into account the algorithms we will apply later, we need to standardize our variables. We want to put different variables on the same scale, for more convenience and for some methods. It's also useful when the data follows a Gaussian Distribution. The procedure is as follow :

$z = \frac{X-\mu}{\sigma}$ - *Step 1*: Subtract the mean, μ, from the value you want to convert, X.
- *Step 2* : Divide the result from Step 1 by the standard deviation, σ.

3.3. Logistic Regression

The logistic regression is a generalized linear model using a logistic function as a link function. The aim is to predict the probability that the individual will consume opioids (value of 1) or not (value of 0) based on the optimization of regression coefficients. When the predicted value is greater than a threshold, the event is likely to occur, whereas when this value is less than the same threshold, it is not. The objective of logistic regression is to find a function h such that we can calculate :

$y = \{1 \text{ if } hX \geq \text{seuil}, \ 0 \text{ if } hX < \text{seuil}\}$

We expect our function h to be a probability between 0 and 1, to be optimized, and that the threshold

we define corresponds to our classification criterion, generally it is taken as 0.5 The sigmoid function is used. The formula is $\sigma(x) = \frac{1}{1+e^{-x}}$

We can rewrite the function h as : $h(X) = \sigma(\Theta X) = \frac{1}{1+e^{-\Theta X}}$

The problem of classification by logistic regression then appears as a simple optimization problem where, from data, we try to obtain the best set of parameters Θ allowing our sigmoid curve to stick as well as possible to the data.

3.4. Ridge regularization with the latent regression

First of all, in our project we apply the ridge regression, which shrinks the coefficients towards zero. This estimator is often used to cope with multicollinearity and can be shown to be a good strategy for the estimation of non-linear and non-parametric models (Pohlmeier 2019). The coefficients estimated by ridge regression are extracted from the following formula:

$$\hat{\beta}_{\lambda}^{RIDGE} = (X'X + \lambda I_N)^{-1}X'y$$

To apply the ridge regression, we combine it with the latent regression, which will help us to determine the best threshold for the predicted values of our target variable. So here we assume that our ridge regression is a latent regression with ridge penalization. As taught in the Machine learning class, the latent variable is represented like this:

Let us suppose there is an unobserved (or latent) variable y* ranging from $-\infty$ to $+\infty$ that generates the observed values of y:

• the large values of y*are observed as 1

• the small values of y*are observed as 0.

Here, to determine τ we take the minimum and the maximum of values predicted by the ridge regression.

The latent y* is assumed to be linearly related to the observed predictors:

$$y_i^\star = \mathbf{x}_i\boldsymbol{\beta} + \varepsilon_i.$$

The observed variable y writes:

$$y_i = \begin{cases} 1 & \text{if } y_i^\star > \tau \\ 0 & \text{if } y_i^\star \leq \tau \end{cases}$$

Where τ is called de threshold.

3.5. Lasso

L1- norm penalty in the Logistic Regression

In our project we also perform the Logistic regression with the Least Absolute Shrinkage and Selection Operator (Lasso) regularization on our data. In contrast to Ridge regularization, lasso penalty is a ‘l1’ penalty,

which forces some of the coefficients to be 0 and therefore leads to models which are more easy to interpret. The optimization problem therefore looks like this:

$$\max_{\beta_0,\beta} \left\{ \sum_{i=1}^{N} \left[y_i(\beta_0 + \beta^T x_i) - \log(1 + e^{\beta_0+\beta^T x_i}) \right] - \lambda \sum_{j=1}^{p} |\beta_j| \right\}.$$

(Hastie et al. (2008))

On the contrary to the L2-norm penalty used in the ridge estimator, the Lasso penalty is more likely to exclude the variables from the model and simplify the interpretability in this way.

As for the application of this model, we use the in-built Logistic Regression function from scikit-learn. We set the inputs to this values:

- **C= grid of the values we defined** (C should be considered as **1/λ** (the inverse of the regularization strength))
- **penalty = 'l1'**, which corresponds to the lasso penalty
- **solver= 'liblinear'**
- **max_iter= 100,** which corresponds to the maximum number of iterations for the solver to converge
- **tol=1e-4** , which is the tolerance for the stopping criteria (tells to the optimization algorithms when to stop).

LIBLINEAR is an open source library for large-scale linear classification.Given a set of instance-label pairs $(\boldsymbol{x}_i, y_i)$, $i = 1, \ldots, l$, $\boldsymbol{x}_i \in R^n$, $y_i \in \{-1, +1\}$, this method solve the following unconstrained optimization problem with different loss functions $\xi(\boldsymbol{w}; \boldsymbol{x}_i, y_i)$:

$$\min_{\boldsymbol{w}} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum\nolimits_{i=1}^{l} \xi(\boldsymbol{w}; \boldsymbol{x}_i, y_i)$$

where $C > 0$ is a penalty parameter. For the Logistic regression, the loss function is $\log(1+e^{-y_i w^T x_i})$, which is derived from a probabilistic model. LIBLINEAR handles this term by augmenting the vector $\boldsymbol{w}$ and each instance $\boldsymbol{x}_i$ with an additional dimension: $\boldsymbol{w}^T \leftarrow [\boldsymbol{w}^T, b]$, $\boldsymbol{x}_i^T \leftarrow [\boldsymbol{x}_i^T, B]$, where B is a constant specified by the user. For the Logistic Regression, it uses the Newton method. The information is from Fan et al. (2008), for more details concerning this classifier use this reference.

3.6. Elastic Net

Elastic Net with the Logistic Regression

As Lasso sometimes does not perform very well with highly correlated variables the Elastic Net is often used instead, it uses the combination of squared L1 and L2 norm penalty. This penalized method deals better with such correlated groups, and tends to select the correlated features (or not) together. In other applications, features may be structurally grouped.(Tibshirani et al. (2015)).

Thus, the optimization function is represented in this form:

$$\max_{\beta_0,\beta} \left\{ \sum_{i=1}^{N} \left[y_i(\beta_0 + \beta^T x_i) - \log(1 + e^{\beta_0+\beta^T x_i}) \right] - \lambda \left[\tfrac{1}{2}(1-\alpha)\|\beta\|_2^2 + \alpha\|\beta\|_1 \right] \right\}$$

where a is the L1-ratio.

To implement this model in our project, we use the python in-built Logistic regression function from scikit-learn. We set the parameters to the following values:

- **C= 0.0937** (C should be considered as **1/λ** (the inverse of the regularization strength)
- **penalty = 'elasticnet'**, which corresponds to the elastic net penalty

- **l1_ratio=0.5**, what means the we will give half importance to the l1-norm penalty and half to the l2-norm penalty
- **solver= 'saga'**
- **max_iter= 100**, which corresponds to the maximum number of iterations for the solver to converge
- **tol=1e-4** , which is the tolerance for the stopping criteria (tells to the optimization algorithms when to stop).

The SAGA optimization algorithm proceeds as follows:
We start with some known initial vector $x^0 \in R^d$ and known derivatives $f_i'(\varphi^0) \in R^d$ with $\varphi_i^0 = x^0$ for each i. For the Logistic regression, derivative f_i' is just a simple weighting of the ith data vector.In such cases, instead of storing the full derivative f_i' for each i, we need only to store the weighting constants. This information is extracted from Defazio et al. (2014).

SAGA Algorithm: Given the value of x^k and of each $f_i'(\phi_i^k)$ at the end of iteration k, the updates for iteration $k+1$ is as follows:

1. Pick a j uniformly at random.
2. Take $\phi_j^{k+1} = x^k$, and store $f_j'(\phi_j^{k+1})$ in the table. All other entries in the table remain unchanged. The quantity ϕ_j^{k+1} is not explicitly stored.
3. Update x using $f_j'(\phi_j^{k+1})$, $f_j'(\phi_j^k)$ and the table average:

$$w^{k+1} = x^k - \gamma \left[f_j'(\phi_j^{k+1}) - f_j'(\phi_j^k) + \frac{1}{n}\sum_{i=1}^{n} f_i'(\phi_i^k) \right], \quad (1)$$

$$x^{k+1} = \text{prox}_\gamma^h \left(w^{k+1} \right). \quad (2)$$

The proximal operator we use above is defined as

$$\text{prox}_\gamma^h (y) := \underset{x \in \mathbb{R}^d}{\text{argmin}} \left\{ h(x) + \frac{1}{2\gamma}\|x - y\|^2 \right\}. \quad (3)$$

3.7. Sequential Forward selection

Sequential Forward selection is an algorithm used to reduce the number of features from k to specified number of features p. It first selects the best single feature, based on the accuracy (or another evaluation metric) score of the model. Then, it will form the pairs of features based on the remaining set of features and the best feature selected in the previous step and select the second best feature. And so on. In our case we perform this algorithm with the Random Forest Classifier. Thus, it would mean that at the first step it will construct 211 Random Forest classifiers for each feature and choose the best of them. Then 210 Random Forest Classifiers with the first best feature and each of the remaining ones and select the best of them. Just from this description we can guess that this feature selection method is very slow, especially in such large datasets. The choice of the best feature is done using the accuracy score computed with 5-fold cross-validation.

More detailed description of the algorithm is shown below:

Initialization : *$X_0 = \varnothing$, $k=0$*
→ We initialize the algorithm with an empty set $\varnothing$ ("null set") so that k=0 (where k is the size of the subset).
Step 1 (Inclusion) :
$x^+ = arg\ max\ J(X_k + x)$, where $x \in Y - X_k$
$X_{k+1} = X_k + x^+$
$k = k+1$
Go to Step 1 :
→ In this step, we add an additional feature, x^+, to our feature subset X_k.
→ x^+ is the feature that maximizes our criterion function, that is, the feature that is associated with the best classifier performance if it is added to X_k.
→ We repeat this procedure until the termination criterion is satisfied.
Termination : k=p
→ We add features from the feature subset X_k until the feature subset of size kk contains the number of desired features pp that we specified a priori.

3.8. Dimensionality Reduction with PCA

Another method that we use in our project is Principal Component Analysis. It is a dimensionality reduction technique, which rewrites a complex system of correlations thanks to linear combinations of the original explanatory variables. In this way, it drops the least important features but at the same time keeps the most 'valuable' information from all the variables. The combinations of these variables will result in 'new' variables, called principal components, which are independent of each other. PCA tries to put the maximum information in the first components.The 'core' of the PCA are the eigenvectors and the eigenvalues of the covariance matrix and the eigenvector (principal component) with the largest eigenvalue represents the direction with the highest variation.
This technique is very useful when one has a lot of variables in the data and is not sure of which variable to remove. In this way it allows to reduce the dimensionality without losing a lot of information.

3.9. Dimensionality Reduction with LDA

LDA which tends for Linear Discriminant Analysis is, as PCA, is a linear transformation technique that helps to reduce the dimensionality, but unlike PCA, LDA is a supervised technique. LDA will try to identify the attributes that take into account the most of the variance between classes. We apply LDA on our standardized Data and with only one component since the number of components can take at most the number of classes minus 1 which is in our case 1. We also tested 2 possible solvers that we can use in our case, which are the Singular value decomposition('svd') and the Eigenvalue decomposition ('eigen'). It gives both the same results and we will keep the default solver which is 'svd'.

3.10. KNN

KNN, which stands for K-nearest Neighbours, is a supervised Machine learning algorithm that is usually used to do both classification and regression. In our case, we will use KNN as a classification method. KNN is a good and convenient non-parametric algorithm, since it does not make any assumption about the distribution of the data and it is quite easy to implement. The principal of KNN is to only calculate the distance between a new point and all other data points is the training set. And there are only two required parameters to implement to the algorithm: the value of K i.e. the K neighbours of the new point that our algo will look at, and the king of metric that will compute the distance between those points (the Euclidean metric, the Canberra, the Manhattan etc.). The algorithm works, for each point in the test set, as follow :

Step 1: Compute the distance between the new point and each row of the training set.

Step 2: Sort the distances in the ascending order and keep the top K distances of the array.

Step 3: Assign to the test point the class of the most frequent class in the top K rows

We first did a KNN model with all the default parameters, the AUC score and the confusion matrix were not bad but we wanted to improve it, therefore we tuned our hyperparameters. To do so, we did a loop to test several combinations of parameters. We tested n_neighbours from 1 to 20, we tried four different metrics for the distance ('euclidean', 'manhattan', 'hamming' and 'canberra') and we also wanted to test 3 values of the

leaf size that can affect the speed. It turns out that the best combination is where K = 11, with the 'canberra ' metric, which is a weighted version of Manhattan distance , and with a leaf size of 30 (the default one).

We illustrated the AUC score evolution of our model according to the number of neighbours in the following graph :

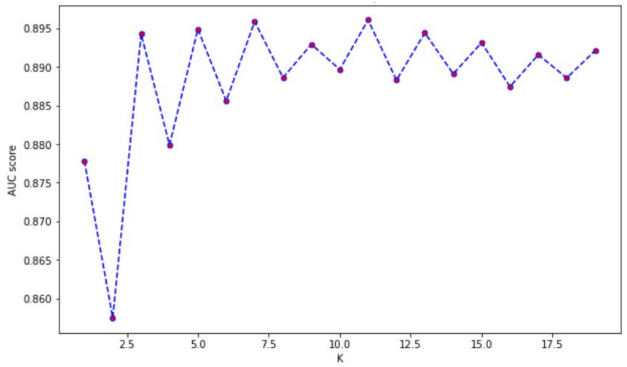


Figure: variation of the AUC score according to the K neighbours

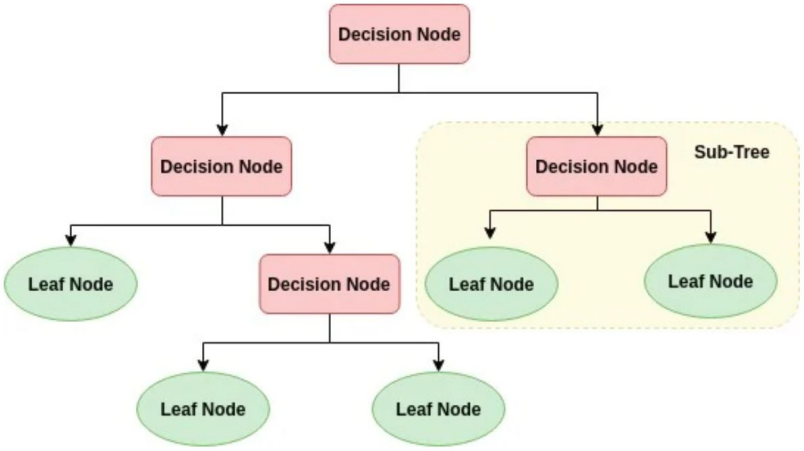
Indeed, we can see that the highest score is for K = 11.

3.11. Decision Tree

The decision tree is an easy and interpretable machine learning algorithm that belongs to the supervised algorithms. With decision tree algorithms, we can do regressions, classification and even clustering. Here, as you must have guessed, we will use it to predict our 2 categories and we did it with the Sklearn's DecisionTreeClassifier. It uses an optimal version of the CART algorithm, which is one of the most popular methods of decision tree algorithms.

The principal of the algorithm is to first, start at the root node with all the training observations and select, among all the features, which split is the best according to the splitting criteria. We have different types of criteria, the most used are The Gini index and the cross-entropy. They are qualified as "purity indices" since these criteria indicate if a subset of data is pure i.e. if all the observations belong to the same class. Thus, the goal is to maximize this "purity". On the other hand, there exists a less used criteria which is the classification error rate, and which gives the proportion of observations that do not belong to the majority class and we need to minimize this rate.

Then, we continue to partition instances with the same criteria, we split the data into several branches and the algorithm keeps going into each branch and gets smaller groups. The algorithm stops when the groups are pure which gives terminal nodes. However, we can determine some stopping criteria, such as the depth of the tree or the minimal number of instances required to split in a node to reduce the tree, it is what we call pruning.



https://www.datacamp.com/community/tutorials/decision-tree-classification-python

Indeed, one of the main problems with Classification trees is that we can overfit our data i.e. the algorithm learns training data so accurately that it fails to generalize a satisfactory result to new data. And to deal with this we can try to tune the hyperparameters to have the best criteria and that is we tried to do now. In a next part, we will also use a random forest method, which results from decision trees and can perform better and resolve the problem of instability that brings the randomness of the decision tree.

Before parameters tuning, we did a classification tree with the default hyperparameters: the Gini criterion as the quality split measure, no maximum depth of the tree, 2 as the minimum number of samples required to split, and one as the minimum number of samples needed to be a leaf node. With these hyperparameters, the AUC score was already good but we wanted to improve it and have a better confusion matrix. Thus, we tested multiple combinations of parameters, both Gini and entropy criteria, a tree depth and a minimum of sample for a leaf node in range (1,60,3), a minimum number of sample to split in range (2,100,3) and we also wanted to test different values for the parameter that displaces the number of features to consider for the best split. Finally we obtained the best predictions with the model having as parameters:

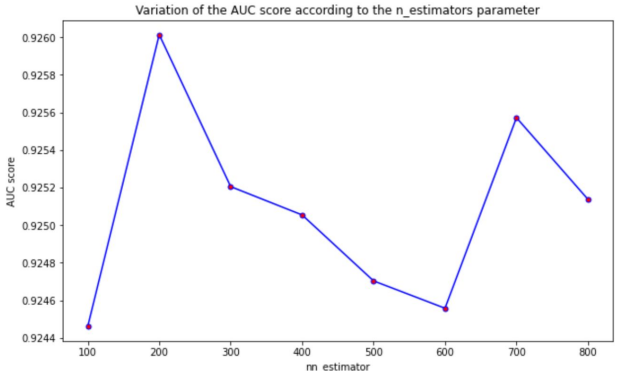
- criterion= 'entropy'
- max_depth= 16
- min_samples_split= 26
- min_samples_leaf= 10
- max_features = 'auto'

3.12. Random Forest

As said before this method comes from the decision tree algorithm and thus, has a lot in common with it. As the decision tree, random forest is an easy and often used machine learning algorithm and has almost the same hyperparameters. However, random forest can resolve the problem of overfitting and instability with decision trees and performs better even without hyperparameter tuning. Indeed, this model builds several decision trees and combines them to be more accurate and have more stable predictions. The random forest method is adding, as indicated by its name, randomness to the model and this, while the trees grow. Namely, instead of taking the best feature when splitting , the algorithm is randomly taking a number of features and takes the best one among those ones.

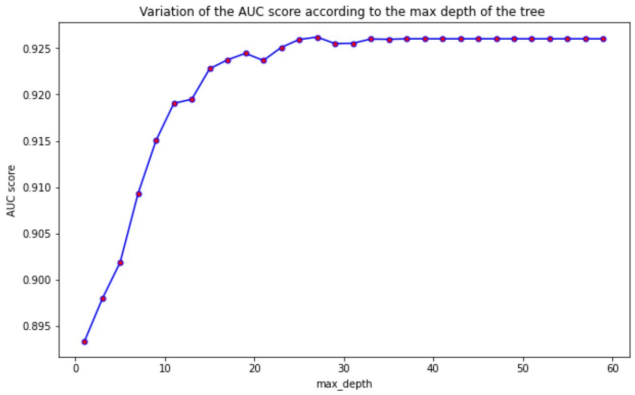
Even if the model performs good without hyperparameters tuning, we will try to find better predictions with à step-by-step tuning hyperparameters. First of all , we have analysed one of the parameters which differs from

those of the decision tree, which is the number of trees in our forest. By default it is equal to 100, and we have looked at the evolution of the auc score when the number of trees goes from 100 to 800.

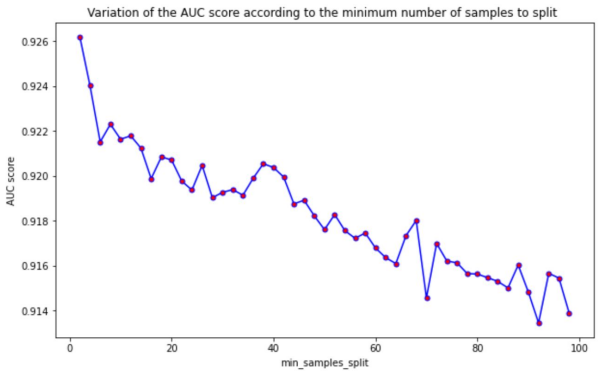


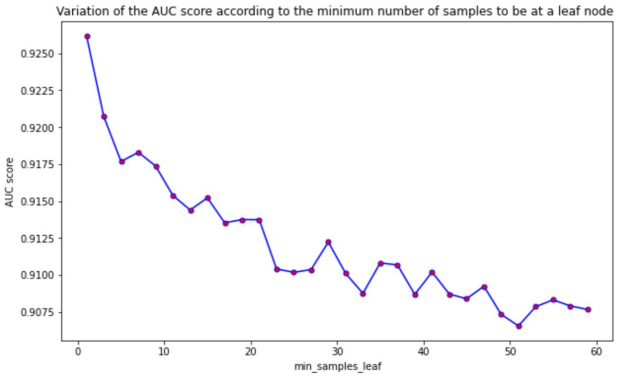
Increasing the number of trees has a small impact on the AUC score, the difference between each point is minimal the higher AUC is 0.9260 and the smaller one is approximately 0.9244 which doesn't make a big difference. However we choose 200 as n_estimator. Then, we tested the criterion, as with the number of trees the difference in terms of AUC is minimal, but the Gini criterion seems to be a little better.

The next parameter we tuned was the maximum depth allowed in trees. We choose a range between 1 and 60.



Here the difference in score is greater and it seems that a higher max depth increases the auc score until a stagnant point. This threshold is for a maximum depth of 27, which we will choose. Then, we tuned the minimum number of samples to split and required to have a leaf node. For min_sample_split we chose a range between 2 and 100, and for min_sample_leaf an range between 1 and 60.





It appears that when we increase both of the parameters, the auc score decreases and a slightly more for the minimum of samples required to a leaf node. We thus decided to keep the default values for these 2 hyperparameters. We also wanted to check what number of features to look during a split and the first 2 (auto

and sqrt) had similar results. Thus we should probably take the 'sqrt' one, since it takes into account less features and thus should decrease the computational time.

Our tuned parameters are the following :
n_estimators= 200

- criterion= 'gini'
- max_depth= 27
- min_samples_split= 2
- min_samples_leaf= 1
- max_features = 'sqrt'

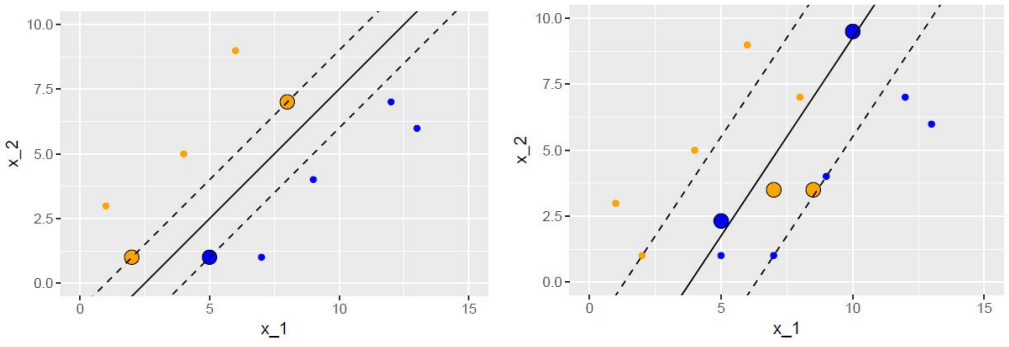
3.13. Support Vector Machine

A Support Vector Machine (SVM) is a supervised machine learning algorithm that can be used for both classification and regression. It is more common to use SVMs in classification problems and that's what we did in this report. But first let's try to understand how this method works. Firstly, SVM can be seen as a generalized version of the maximal margin classifier, the difference is that MMC requires a linear boundary to separate the classes of the response variable whereas SVM allows non-linear ones.

To better understand SVM we will take a look at the MMC. First let's consider a p dimension where our response variables are separated perfectly in two by an hyperplane. In this case we can have a lot of hyperplanes that cuts perfectly this hyperplane, that's why we need to find the optimal one, i.e. the one for which the margin, i.e. the smallest distance from each observation to a given separating hyperplane, is the farthest from the observation and we call it the maximal margin Hyperplane. Now, we can introduce the support vectors that are the closest and equidistance observations to the margin hyperplane, if they move the maximum margin hyperplane moves too. After finding the maximum margin hyperplane the classification can follow, one side of the hyperplane is a class and the other side in the other class.

However this is in the ideal world, and in reality we can't have a hyperplane that linearly separates perfectly our observations and thus have a maximum margin classifier. That is why, we introduce the support vector classifier and a non-linear kernel that will finally give us the Support vector machines. In support vector classifiers we allow slack variables i.e. we allow observations to lie on the wrong side of the hyperplane. In the optimization pb we introduce a new parameter in addition to the previous one (the width of the margin), which is C, the parameter to allow observations to be on the wrong side, if C=0 we don't allow any.

Figure A: Maximal margin classifier *Figure B: Support vector Classifier*



Ewen Galic, Machine learning and statistical learning, 2020

Now we just have to include non-linear decision boundaries. And to do so the model adds dimensions to the space with functions on the predictors and thus uses a kennel instead of a set of predictors. We finally call the Support vector classifier with a non-linear kernel a **support vector machine which is :**

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i K(x, x_i)$$

In our study we used SVM, after normalizing our data and we first looked at the SVM with the defaults hyperparameters. There are 3 main parameters : first the kernel type which is **Gaussian** by default, then we have C, the regularization parameter which is 1 by default and finally we have gamma, the kernel coefficient. We tuned the parameters to have better results. We did a loop with different parameters combinaisons, we chose 4 values for C (0.1, 1, 10, 100) and also 4 values for gamma (1, 0.1, 0.01, 0.001) and we tested 3 kernels (sigmoid, polynomial and gaussian). It appears that the best combinaison in terms of auc score and confusion matrix is the following:

- C= 100
- kernel = 'rbf'
- gamma = 0.001

3.14. Logistic Regression with Gradient Descent Algorithm

Standard Gradient Descent Algorithm is an iterative optimization algorithm to find the minimum of a function. To answer the optimization problem of a Logistic Regression and estimate our parameters, we can use this algorithm. We know that the partial derivative of a function at its minimum value is equal to 0. The gradient descent uses this concept to estimate the parameters or weights of our model by minimizing the loss function. We are going to describe all the steps of this algorithm that we have coded ourselves.

Step 1 : We initialize W=0, b=0 and lr= learning rate. W is the weight, b is the biais and the learning rate controls how much W and b change with each step. If the value of learning rate is too high then gradient descent won't reach the minimum point and we will just keep oscillating. If the learning rate is too small, it will take a lot of iterations to get to the accurate value We initialize lr = 0.001
Step 2 : We approximate y with linear combination of weights x and bias : f(w,b)= Wx + b. We apply the sigmoid function to get probability
Step 3 : We have to compute gradients so calculate the derivative of W and b

Update Rules

$$w = w - \alpha \cdot dw$$
$$b = b - \alpha \cdot db$$

$$J'(\theta) = \begin{bmatrix} \frac{dJ}{dw} \\ \frac{dJ}{db} \end{bmatrix} = [\dots] = \begin{bmatrix} \frac{1}{N} \sum 2x_i(\hat{y} - y_i) \\ \frac{1}{N} \sum 2(\hat{y} - y_i) \end{bmatrix}$$

Step 4 : We update the current value of W and b and we repeat the process until the loss function = 0

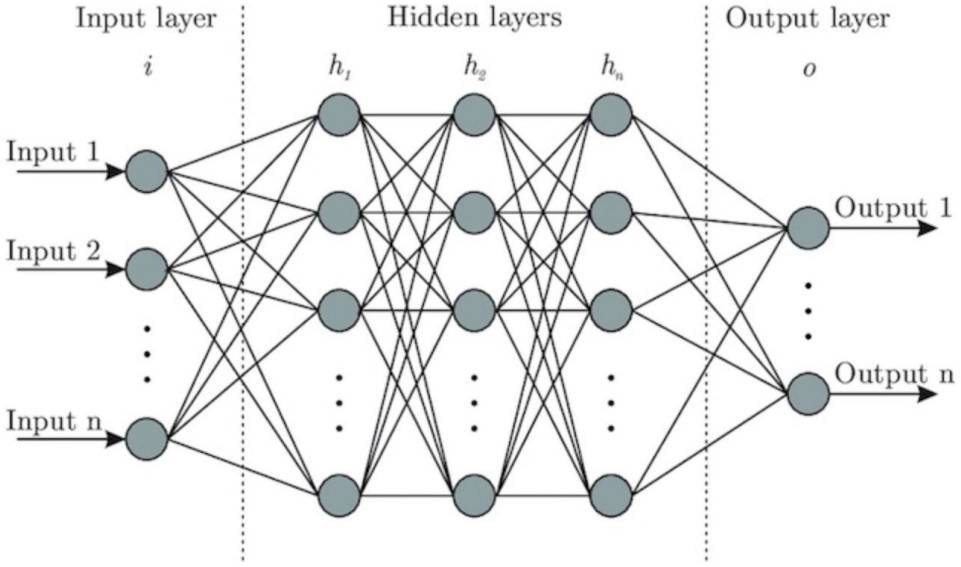
3.15. Neural Network

Neural networks are a set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns. It is a Deep Learning method. All variables need to be transformed into a numeric form because neural networks interpret sensory data through a kind of machine perception, labeling or clustering raw input and recognize only numerical information. Neural networks help us to group and classify. They help group untagged data based on the similarities between the entries in the example, and they classify data when they have a set of labeled data to train on. The neural network uses the examples to automatically infer rules for recognizing the class. Furthermore, by increasing the number of training examples, the network can learn more about data, and so improve its accuracy.

In our case, we only use supervised learning methods so we will explain the Artificial Neural Network (ANN). We want to explain step by step this method. In ANN, we consider a non linear prediction function $h_{W,b}(x)$ where W and b are parameters

- **W is the weight**. This is a connection between neurons that carries a value. The higher the value, the larger the weight, and the more importance we attach to neuron on the input side of the weight
- **b is the biais**, is also a weight. Every neuron that is not on the input layer has a bias attached to it, and this bias, like weight, has a value.
- **Activation** : a neuron is supposed to make a tiny decision on that output and return another output. This process is called an activation. We represent it as $f(z)$ where z is the aggregation of all the input. They are 2 categories of activation : linear and nonlinear. Few popular are ReLU activation equal to $f(z)= \max(0,z)$ and sigmoid equal to $f(z)= \frac{1}{1+e^{(-z)}}$

In **Figure 1**, we can see an example of a neural network with several neurons.



A neuron is a computational unit that takes inputs and returns outputs. The output of the neuron is $f(\sum_{i=1}^{n} W_i x_i + b)$ where x is input to the neuron and n the number of inputs from the incoming layer.

About the activation function, the choice of it depends on the nature of the problem. We use, in the first time, sigmoid function, because it exists between (0 to 1). Therefore, it is especially used for models where we have to predict the probability as an output. Since probability of anything exists only between the range of 0 and 1, sigmoid is the right choice. The function is differentiable.That means, we can find the slope of the sigmoid curve at any two points. The function is monotonic but the function's derivative is not. Other functions:

- **The logistic sigmoid** function can cause a neural network to get stuck at the training time.

- **The softmax function** is a more generalized logistic activation function which is used for multiclass classification. Generally, we use softmax activation instead of sigmoid with the cross-entropy loss because softmax activation distributes the probability throughout each output node. But, since it is a binary classification, using sigmoid is the same as softmax. However, with softmax function the performance of our model is better, so we decided to keep going with softmax function.

The basic structure of an ANN consists of artificial neurons (similar to biological neurons in the human brain) that are grouped into layers. The most common ANN structure consists of an input layer, one or more hidden layers and an output layer but many other architectures can be chosen and the neural network will be very complex. We talk about Deep Learning when the number of hidden layers is higher than 3. We can also choose the number of hidden layers, the connections between layers (fully or densely) or the activation of layers.

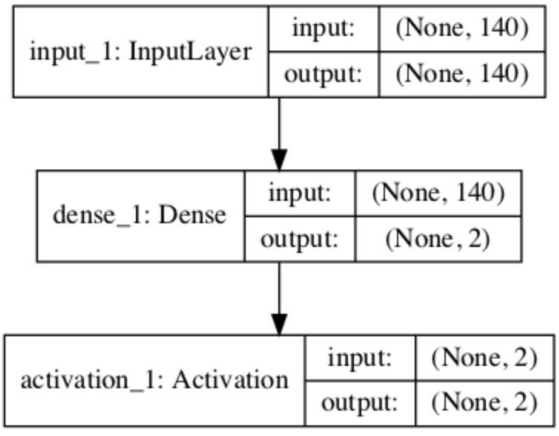
After the configuration of the neural network, we want to train it. The most common use is the Batch Gradient Descent Algorithm. We explain the principal in the Logistic Regression but we can just rewrite the cost function, we want to minimize with this algorithm :

$$J(W,b) = \left[\frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}||h_{W,b}(x^{(i)}) - y^{(i)}||^2\right)\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}(W_{ji}^{(l)})^2$$

We can see that the first term is the classical sum of squares error and the second term is the regularization parameter. It's used to control the magnitude of the parameters. ƛ controls the importance of the second and the first parameter.

After studying the method carefully, we decided to set up our Neural Network (**Figure**) with :

- ➔ one hidden layer
- ➔ dense : fully connected layer
- ➔ activation function : Softmax
- ➔ loss function : Adam



We use only one hidden layer because we do not want to complexify the model that answers a binary classification problem. Finding the optimal number of hidden layers seems to be complicated and in Deep Learning, there are no strict rules to define the number of layers, the number of hidden units per layer and even the type of connections between layers.

After several tests, we decided to use Adam, instead of Batch Gradient Descent Procedure. Adam is an optimization algorithm that can be used to update network weights iteratively based on training data. The algorithm leverages the power of adaptive learning rates methods to find individual learning rates for each

parameter. Adam can be looked at as a combination of RMSprop (RMSprop is a gradient based optimization technique used in training neural networks) and Stochastic Gradient Descent with momentum. It uses the squared gradients to scale the learning rate like RMSprop and it takes advantage of momentum by using moving average of the gradient instead of gradient itself.

3.16. K-means : Unsupervised Method

We wanted to practise an unsupervised method to our supervised classification problem. We decided to use the K-means algorithm, which we applied only on our numerical standardized variables because we cannot execute it on mixte variables. Another possible method is K-modes, to be used on the qualitative variables. The K-means algorithm allows us to group the individuals in our database into distinct K clusters. This K number must be specified, but there are approaches to determine its optimal number.
This algorithm is based on minimizing the sum of the squared Euclidean distances between each point and the centroid of its cluster. It is a non-hierarchical clustering approach because within a cluster individuals are not ordered according to their resemblance, whereas hierarchical clustering just tells us what two things are most similar.
The steps of the algorithm is :

Step 1 : Assign a cluster to each individual, randomly.
Step 2 : Calculate the centroid of each cluster
Step 3 : For each individual, calculate his Euclidean distance with the centroids of each cluster.
Step 4 : Assign the closest cluster to the individual
Step 5 : Calculate the sum of the intra-cluster variability
We have to do *steps 2 to 5,* until reaching an equilibrium. After that, we talk about convergence: no more change of clusters, or stabilization of the sum of the intra-cluster variability.
Our database is used unbalanced for this algorithm (many more individuals in class 0 than 1) because we are in a case of unsupervised predictions. But K-means, produces a cluster of uniform size even when the input data have different sizes. Biased results are therefore expected.

Now we can wonder, how do we choose the right number of Cluster K ? Many methods exist, but we decided to apply the Elbow and the Silhouette methods.

- **The Elbow method** : based on the minimization of the sum of squares of deviations within clusters. We calculate this score for many K and choose the K for which the curve first starts to diminish. If the line chart looks like an arm, then the "elbow" on the arm is the value of *k* that is the best : a huge reduction in variation. **Figure** shows it.

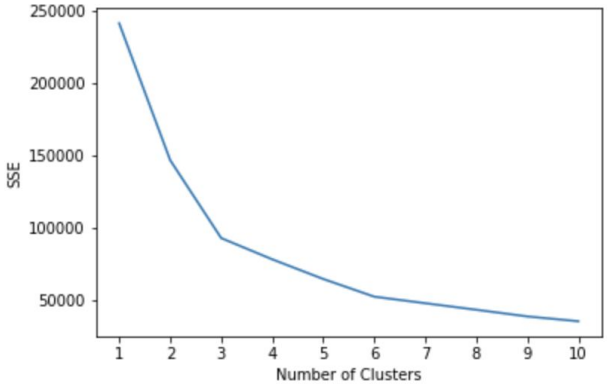


Figure : Elbow Curve to find the best K for Kmeans

Unfortunately, we do not always have such clearly clustered data : it's our case. This means that the elbow may not be clear and sharp. We can see 2 "Elbow" K=2 or K=3. We need another method.

- **The Silhouette Method :** measures how similar a point is to its own cluster (cohesion) compared to other clusters (separation). The range of the Silhouette value is between +1 and -1. A high value of 1 indicates that the individual is placed in the right group. On the other hand, if many individuals have a negative Silhouette value, it may indicate that we have created too many or conversely too few groups, it is a trade-off. **Figure** shows it

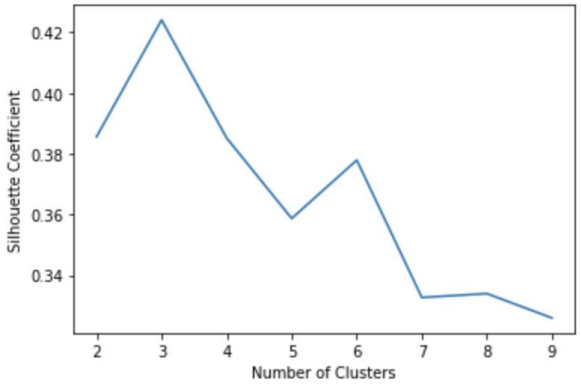


Figure : Silhouette Coefficient Curve to find the best K for Kmeans

Now we can confirm that K=3, because it's the higher point in the curve, when we use only 2 numerical variables : the age and the weekly wage. It is not the same number of clusters as our database but the information has been reduced to only 2 variables and makes the specificity of these 2 variables more important in clustering. In our case, in the medical field, meaningful clusters expand domain knowledge. The clustering results identified groups of patients who have consumed or not opioids and have different drug reactions or identified groups of patients who respond differently to medical treatments (with opioids used).

4. Results

4.1 Descriptive statistics

Our final database contains 120,661 rows and 81 variables. Before transforming the database, we compute some descriptive statistics to better understand our data. If we look at the target variable "*Opioids used*", we have 11.39% of true and 88.61% of false. The class we are interested in is under-represented in our sample. Through these descriptive statistics, we seek to find out if there is a category of the population that is more affected and more vulnerable to this crisis. We also want to compare our results with existing publications on the subject. From Ivana Obradovic, sociologist, this epidemic initially affected both men and women from the middle classes (between $20,000 and $50,000 per year), having exceeded 40 years of age. This crisis also affects another "unusual" audience such as mothers and newborns, victims of a withdrawal syndrome at birth. We want to compare these results with our database. **Table** shows some descriptive

statistics about the continuous variable : "*Claimant Age*" and "*Weekly Wage*".

	Claimant Age								Weekly Wage							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
Opiods Used																
0	106917.0	42.486200	13.413243	0.0	31.0	43.0	53.0	113.0	106917.0	725.238486	488.866826	0.0	399.9500	605.600	970.9000	9999.99
1	13744.0	44.111685	11.103587	13.0	36.0	45.0	52.0	82.0	13744.0	676.128058	426.766100	0.0	383.7125	567.785	880.5275	9465.38

The average age of the individuals is 43 years. The majority of opioid consumers are between 36 and 51 years and their average salary per week is $676 (around $32,000 per year). It's the middle class. We can see that the average weekly wage is lower for those who use opioids than other individuals. About the wage, we can see aberrante value in the **Figure**, but we cannot correct them, we don't have enough information about the income of the claim.

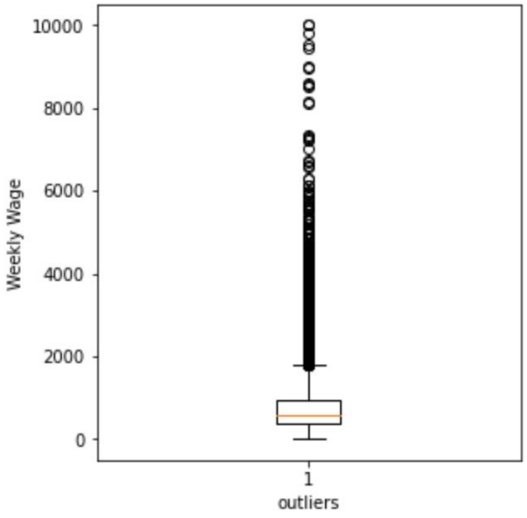


Figure : Boxplot about the Weekly Wage

There is a large disparity in wages between the 25% with the highest wages and the rest of the distribution. The majority of individuals earn less than $2,000 per week. The richest 25% of the distribution have wages ranging from $2,000 to $8,000 per week (wages of $10,000 per week seems to be the lowest).

About categorical variables, there is no significant difference between men and women on opioid use. Both sexes are equally vulnerable to this crisis. Individuals, on average, who have used opioids have had an accident that resulted in medical care, compared to only 3% of individuals who have not used opioids but have had an accident. Accidents are more serious when opioid use is observed on average. Using opioids increases the average chance of being hospitalized or needing medical attention. This is also the case on average for laboratory tests (13% vs. 4%), needing an X-ray (4% vs. 0%) and needing surgery (5% vs. 2%). Regarding the taking of medication, we note that among individuals who have taken opioids, an average of 5% have taken Benzo (an Anxiolytic) compared to 0% for other individuals. About the situation of the individual, it can be seen from **Figure** that married individuals, on average, have consumed the most opioids in our database. However, the differences with the other categories (single, etc.) are not very high.

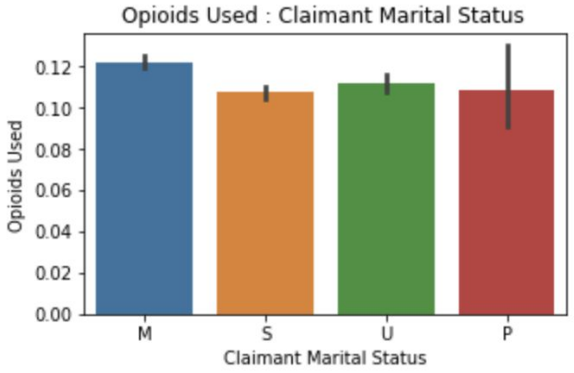


Figure : Histogram of Opioids Used and Claimant Marital Status

In addition, some economists have suggested that the epidemic has had a collateral effect on unemployment and is a factor driving people away from employment. The "NLF" category, which includes 23 million adults aged 25 to 54, the age group most affected by the epidemic, would be inactive without looking for work. We have in our database a majority of full time workers. But when we look at each category, we can see in **Figure** that the unemployed, volunteers and seasonal workers are those who, on average, consume the most opioids.

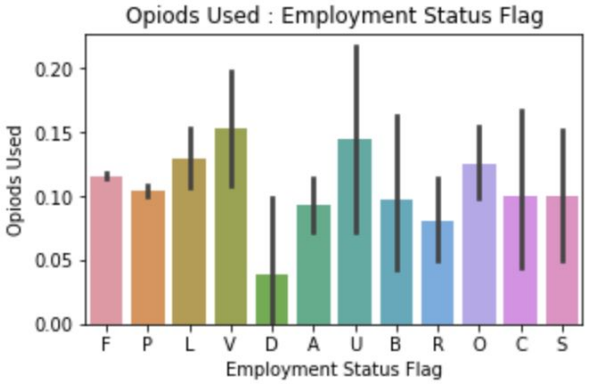


Figure : Histogram of Opioids used and Employment Status Flag

To finish this part, we decided to focus on the states most affected by this crisis. We made two different maps. The first one, **Figure,** shows the states where there are individuals who have used opioids or not. The second map shows the states of opioid consumption intensity in the states of America : North Dakota and Wyoming there is no consumption of any opioids.

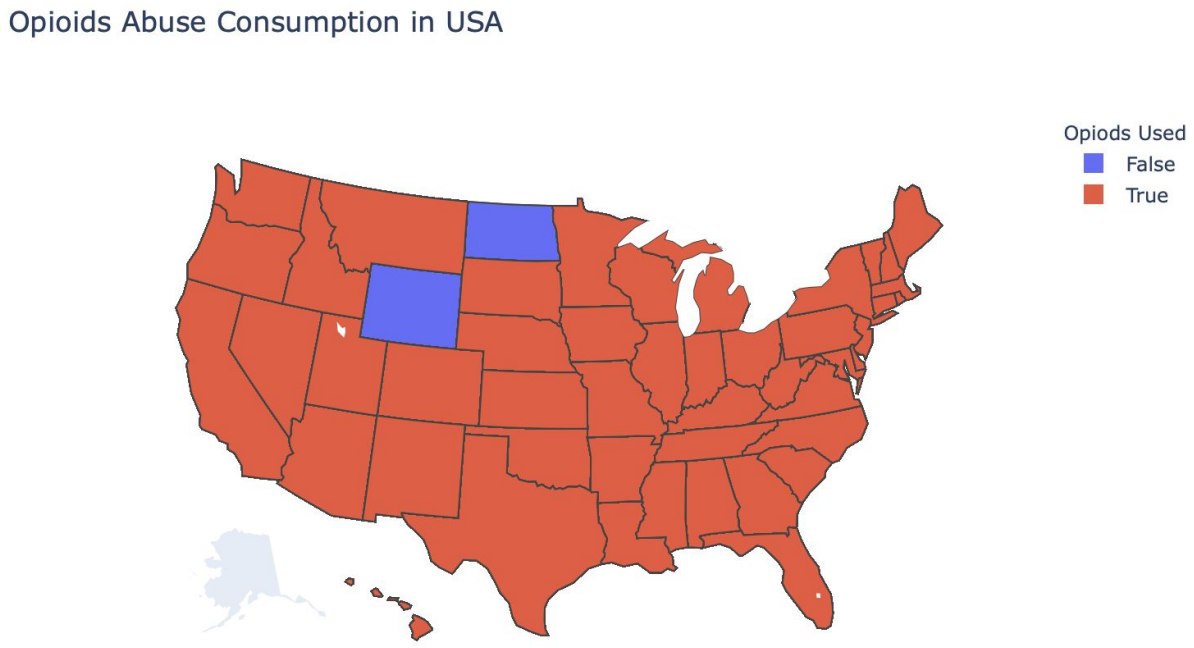


Figure : Map of States in America with Opioids used or not

In our database, we can see that in 2 states, On this second map, **Figure**, we can see that 3 states in our database are most affected by opioid addiction: the lighter colored states have the highest consumption intensity. It's Pennsylvania, Georgia and Mississippi states. For example in 2018, Georgia providers wrote 63.2 opioid prescriptions for every 100 persons, compared to the average U.S. rate of 51.4 prescriptions. This result is given by the National Institute on Drug Abuse.

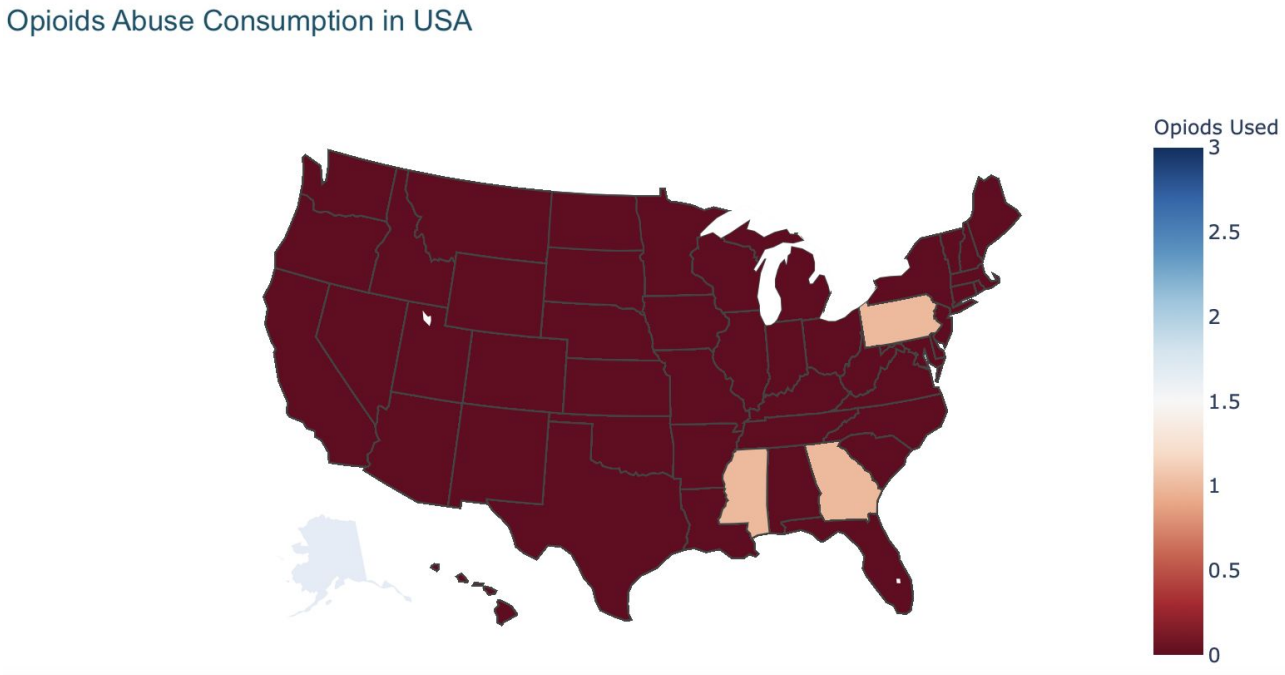


Figure : Map of States in America with Opioids used intensity

For Ivana Obradovic, the most affected states are characterized by three aspects; significant prescribing practices by physicians, the specific characteristics of opioid users, and a local socio-sanitary environment lacking appropriate care and treatment services that are unconditionally accessible.

4.2 Evaluating performance estimator

Evaluation of the performance of the models

In order to evaluate the performance of our models on the unseen data, we use train/ test split and k-fold cross-validation.

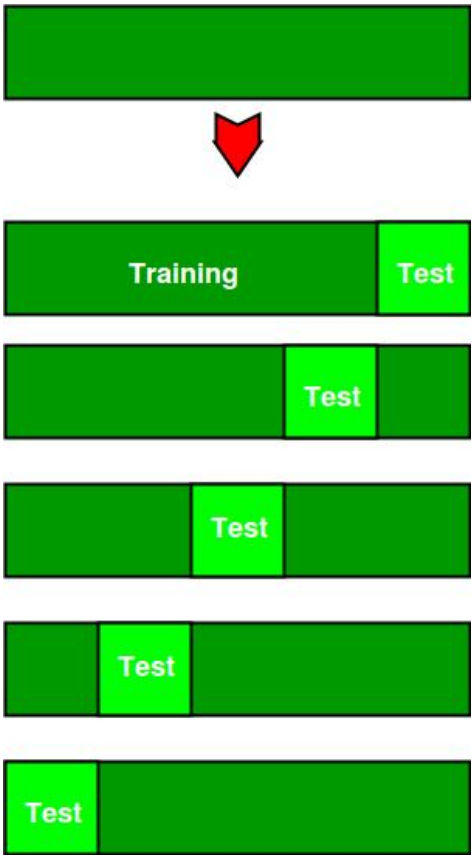
Train/test split

The quality of the algorithm is usually measured not on the whole data set of observations, but rather on a subsample. This is why we put 80% of our data into the train set and 20% in the test set (to make out of sample predictions). Using the test data we can see how well the model will perform in the future.

K-fold cross-validation

In some of our models we use the k-fold cross-validation method in order to tune over the hyperparameters and choose the best one. This method works by splitting the dataset in k folds. In each step we fit the model to k-1 folds and evaluate the performance with the help of the chosen metric on the k-th fold. Once all of the k folds were used in the evaluation set, the average of the metric is computed. The use of this method will lead to the better out-of-sample performance of the model.

The example of the schema of the 5-fold cross-validation is shown above:



Evaluation metrics

As we deal with the imbalanced dataset and we are dealing with insurance data, the accuracy is not representative for the performance of the models, as it can be over-optimistic. Thus, we need to define some other evaluation metrics.

First of all let's display the confusion matrix:

		Predicted class		
		−	+	Model error
True class	−	TN	FP	$FPR = FP/(TN+FP)$
	+	FN	TP	$FNR = FN/(FN+TP)$
	Use error	$FOR = FN/(TN+FN)$	$FDR = FP/(FP+TP)$	Overall error $ACC = \frac{(FP+FN)}{(TN+FP+FN+TP)}$

Table 6: A confusion table.

The first evaluation metric, which can be used in the imbalanced datasets is AUC (Area under the ROC curve).Its formula is equal to AUC=(1+TPR-FPR)/2. Where TPR=TP/(TP+TN). AUC provides a single measure of a classifier's performance for the evaluation of which model is better on average (Lopez et al. (2013)). This measure shows how good the model distinguishes between 2 classes.

We also will assess the performance of our models with the help of the Matthews correlation coefficient (MCC). This is a good performance measure, as it takes into account all 4 confusion matrix categories.

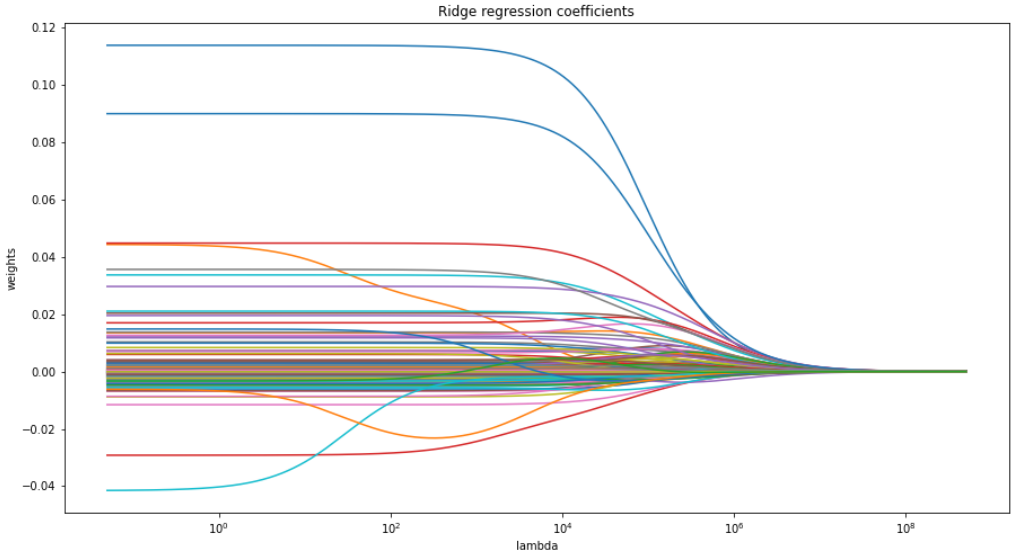
The formula of the MCC is:

MCC = (TP * TN—FP * FN) / ((TP + FP)(TP + FN)(TN + FP)(TN + FN))^{½}

Especially on imbalanced datasets, MCC is correctly able to inform you if your prediction evaluation is going well or not, while accuracy or F1 score would not.(Chicco (2017)). MCC close to 0 would mean that the performance of the classifier is close to the random guessing.

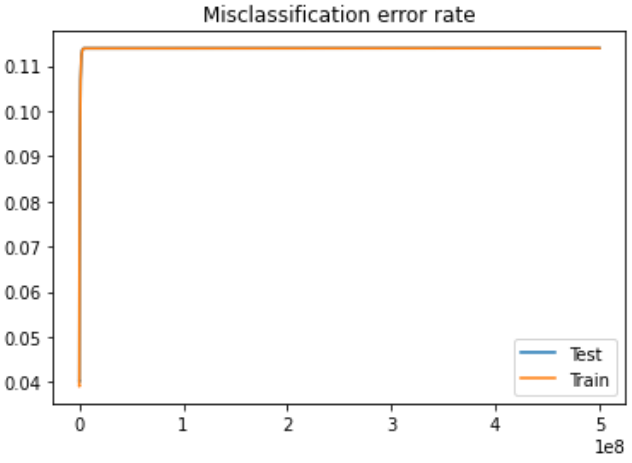
4.3. Ridge regression

Before applying the ridge regression, we create the values of lambda going from 0.5^{-1} to 0.5^{9}. After, we compute the coefficients and predict the target variable using the features from the test sample. The result for each lambda is stocked in the variable ***pred*** . We also compute the predicted values for the training set and stock the results in the variable ***pred_train***. We also plot the graph in order to display the evolution of the coefficients as long as lambda increases. This evolution is shown in the graph below (here we do not apply any undersampling techniques):

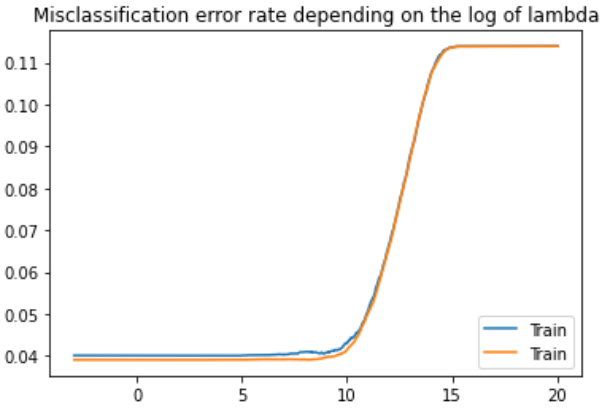


As we can see from the graph above, the coefficients are shrunk to 0, when lambda goes to infinity. We can also see that even for low values of lambda, most of the coefficients are already close to 0.

As our target variable is binary, MSE would not be representative for the performance of the model. This is why we need to compute the misclassification error rate. But to do so we first need to assign the class to the predicted values stocked in the variable ***pred*** and ***pred_train***. Here we have an imbalanced dataset, which makes this threshold value disproportional. We compute the threshold using lambda equal to 0.05 and obtain τ=0.3575. After this step we are ready to assign the classes to predicted values as it is shown in the equations for the latent regression. Once the classes are assigned, we can compute the misclassification error rate. The graphs representing these rates for train and test sample are displayed below:

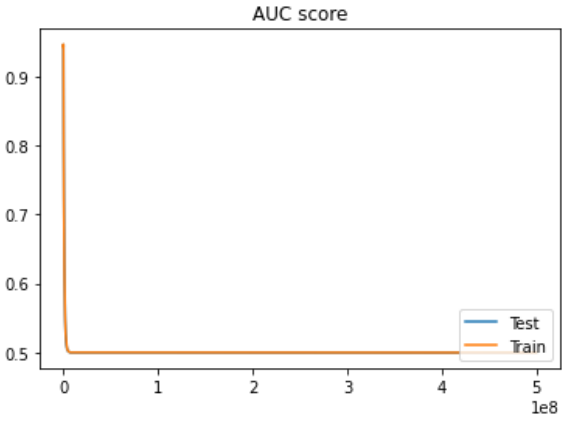


Although here we cannot see the difference between the 2 of misclassification error rates, if we zoom this graph we could see that the misclassification error rate is higher on the test data. Which means that there is a slight overfitting. But, this overfitting becomes lower as soon as lambda becomes bigger. We also plot the misclassification error rate together with the log of lambda, which results in the graph below:

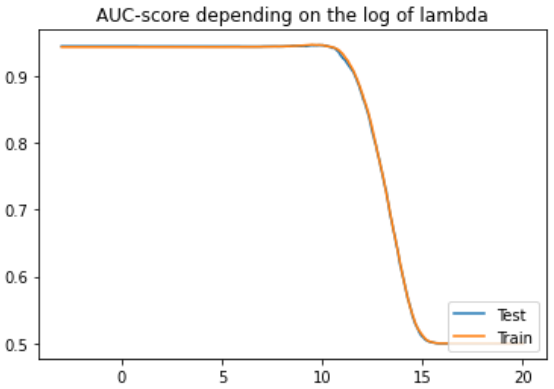


One must know that our dataset is imbalanced, thus the misclassification error rate is not always a good representation. This is why we also compute the AUC-score, which would be more representative for the performance of our model.

This is why based on the AUC-score we compute the new threshold which is now equal to 0.2300. And we compute the evolution of AUC-scores on the training and testing sample. We obtain the graphs below:



From this graph we can observe a good performance for low values of lambda, but poor for higher values.

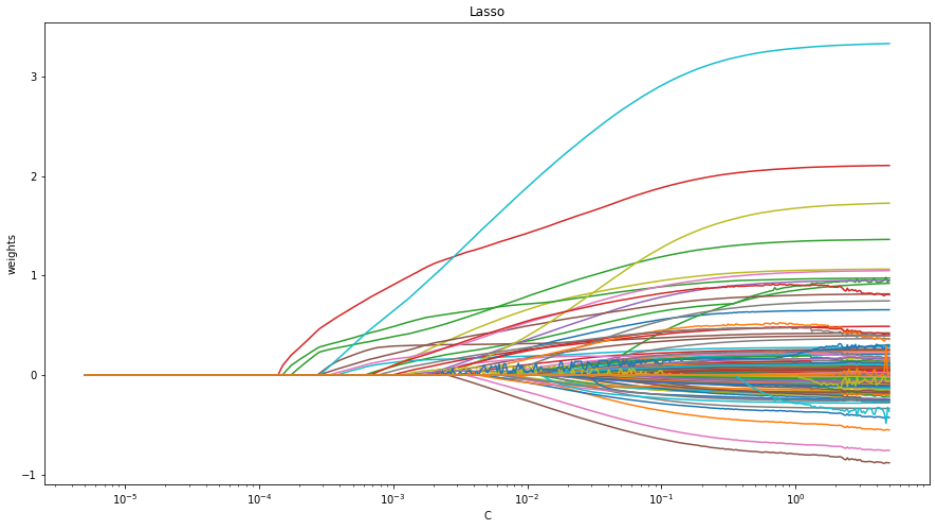


As before, we zoom the graph and see that train AUC-score is higher than the test one, which means that the performance on the train set is better than the one on the test and thus there is a slight overfitting. We can also observe that after using AUC-score as a performance measure, the performance becomes lower.

4.4. Logistic regression with the Lasso penalty

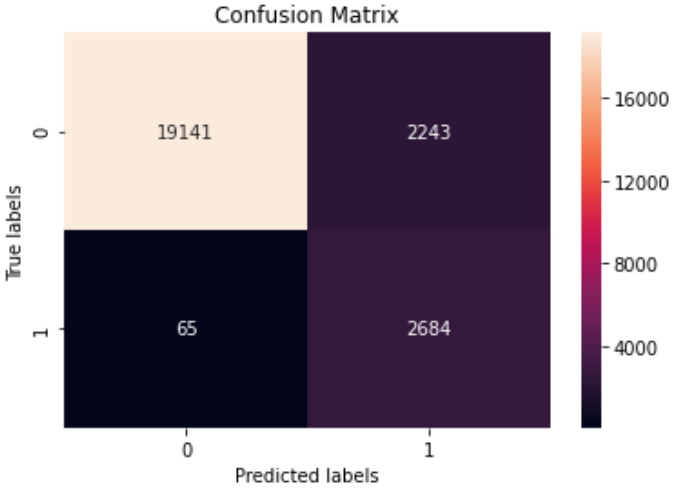
In order to apply the Lasso penalty to the logistic regression we will use the python function for Logistic regression and assign a 'penalty' equal to 'l1'. Here we define parameter C between $0{,}5^{-5}$ to $0{,}5^{1}$. And one must know that this parameter C should be considered as **1/λ** (the inverse of the regularization strength)**.**

Before applying the Logistic regression, we balance our dataset using NearMiss-3 undersampling. We do so because our dataset is large and contains a lot of features. Thus, applying any model would take a lot of time. After balancing the dataset we obtain the following result for Lasso coefficients:



As we can see for very large values of the penalty parameter (or very low values of parameter C) all parameters are shrunk to 0, we risk to have a very simple model with low performance. While, when lambda is too low (C is too large), the model will include all the parameters and we risk to have a very complex model and overfitting. Thus, we need to find a balance between 2.

To do so, we select 100 values of C and apply a GridSearchCV in order to determine the best value of C. We use 10-fold cross-validation in order to tune C (here 1/lambda). We take the values of C between $0{,}5*10^{-4}$ and $0{,}5*10^{0}$. The results show that the best value of C is equal to 0.0936 (which means lambda equal to approximately 10,68) and the cross-validation accuracy on the training set is equal to 0.9537. We then apply the model on the test data and obtain the following confusion matrix:

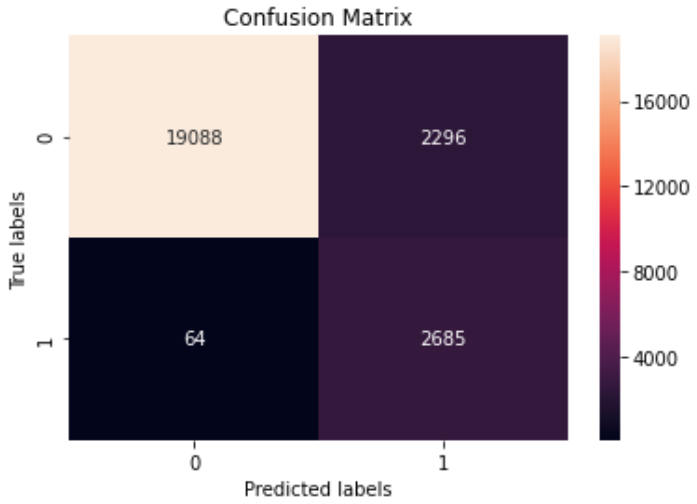


The accuracy on the testing set is 0.9044. The AUC is 0.9357 and the MCC is 0.6869. As for the training set, the accuracy and the AUC is of 0.9731 and the MCC is of 0.9462.

Afterwards, we want to see how many coefficients were removed from the model. We observe that there are 72 features removed (see **Appendix B** for the list)

4.5. Elastic net with Logistic Regression

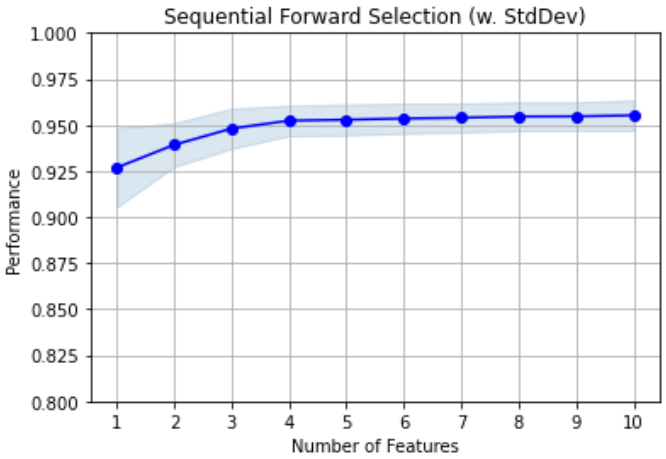
For the Elastic Net penalty we define the 'fair' proportion for each of 2 penalties equal to 0.5 and we set the C parameter to the value found for the Logistic regression with Lasso penalty above. We find that the testing accuracy is 0.9022, the AUC is 0.9347 and the MCC is 0.6824. On the training set the accuracy and AUC are 0.9733, MCC is 0.9467. To see the selected features and the features set to) with the elastic net see Appendix C.
The confusion matrix is shown above.



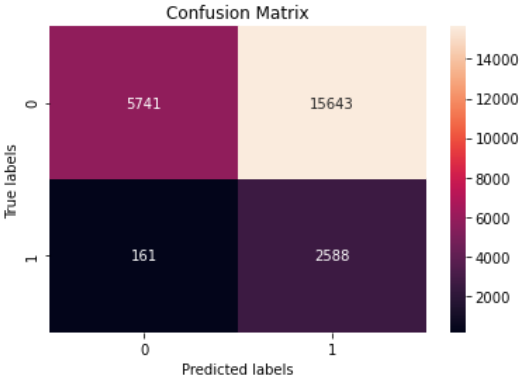
If we compare this confusion matrix with the one of the Logistic regression with the L1-norm penalty, we can see that the number of False negatives is lower of only 1 observation for the Elastic Net, while the number of False Positives is increased with 53.Thus, it is better to use the Lasso penalty in order to remove non-important features (except if we tune over the regularization strength and l1-ratio in order to find the best model and compare then the results). The elastic net selects 49 features and removes 46 of them (see **appendix C** for the list).

4.6. Feature selection with Sequential Feature Selection method

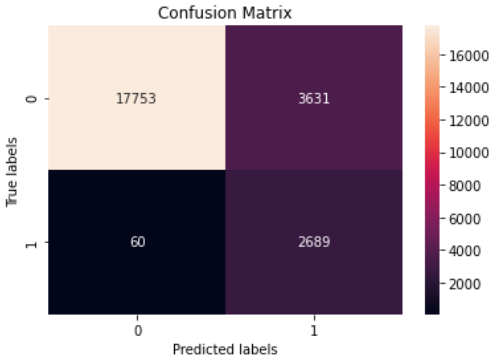
After applying the Sequential Feature Selection method to the Random forest Classifier, we find 10 best features with the following ranking:
1.'Weekly Wage squared'
2.'NDC Class - Muscle Relaxants'
3.'CPT Category - Anesthesia'
4.'HCPCS M Codes'
5.'Weekly Wage cubic'
6.'NDC Class - Stimulants'
7.'NDC Class - Misc (Zolpidem)'
8.'Benefits State_WA'
9.'HCPCS B Codes'
10.'NDC Class - Benzo'
The accuracy on the training set after adding every feature is shown in the graph below:



The accuracy is surprisingly high for such a small number of features. But when we check the accuracy on the testing set we observe that it is only 0.345, AUC is 0.606 and MCC is 0.155 (which means that it is close to the random guessing). If we look at the same metric using the whole features set, we observe the accuracy of 0.847, AUC of 0.904 and MCC of 0.584.



1.With the whole features set



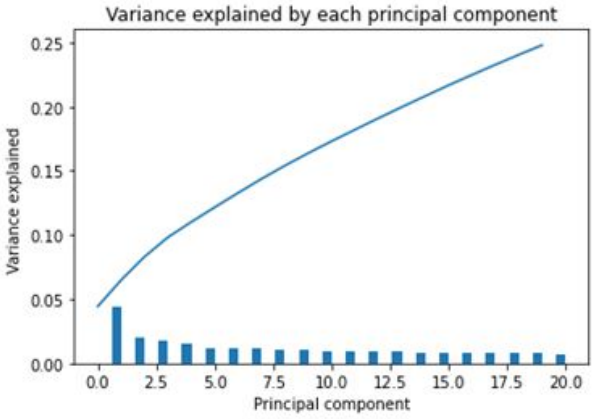
2. With selected features

This technique is not very reliable and leads to a strong overfitting. In addition, this technique takes a lot of time (almost 2 hours to select 2 features).

As we can see the fact that there are many explanatory variables and a large dataset leads to the problem of model selection. It is practically infeasible to evaluate each possible model with each possible combination of features in order to pick the best subset of features based on some goodness of fit criterion.

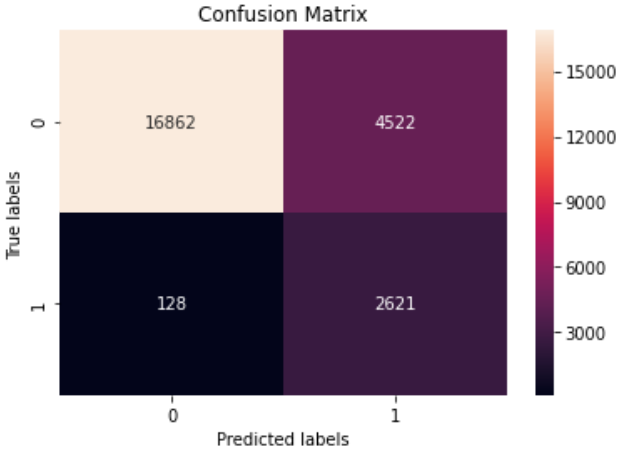
4.7. PCA

After applying PCA to our features and extracting from it the explained variance ratio, we display the explained variance and the cumulative explained variance in the graph below:

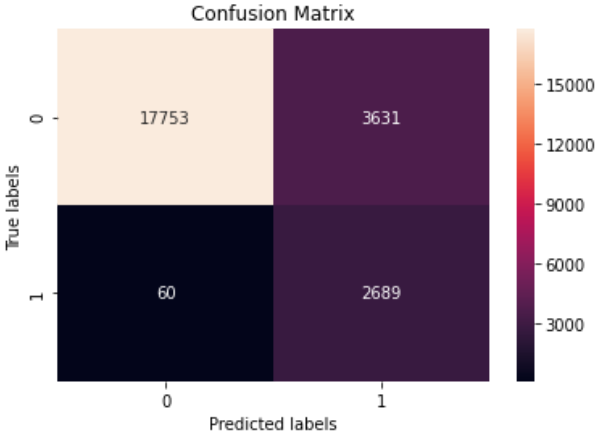


The bars correspond here to the percentage of variance in the data explained by each principal component. The curve is the cumulative variance explained. Plotting a curve we observe that the first Principal Component explains approximately 4% of variability in data. Another key observation is that 25% of the variability can be explained by the first 20 Principal components.

After this, we compare the accuracy of 2 models, the one with the use of PCA and the other without. In this case we apply the Random Forest Classifier. We observe that after applying PCA the accuracy on the test set is about 81%, while without it is 85%. The AUC is 0,87 while without PCA it is 0,90, and the MCC is 0,52 while without PCA it is 0.58. If we also compare the confusion matrices, we could see that the number of False Negatives becomes 2 times higher, which is quite important in our project.



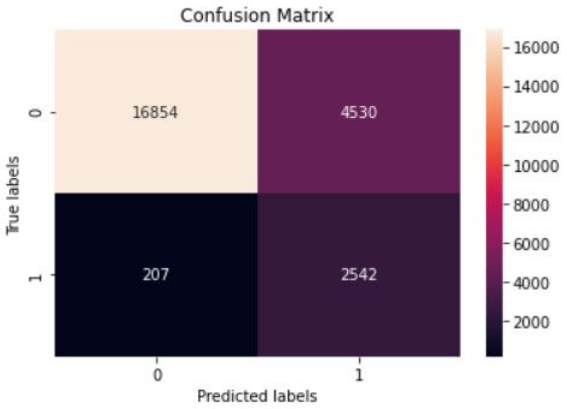
1. With PCA



2. No PCA

4.8. LDA

The LDA method gives us a AUC score of 0.856 , an MCC score of 0.498 and this confusion matrix:

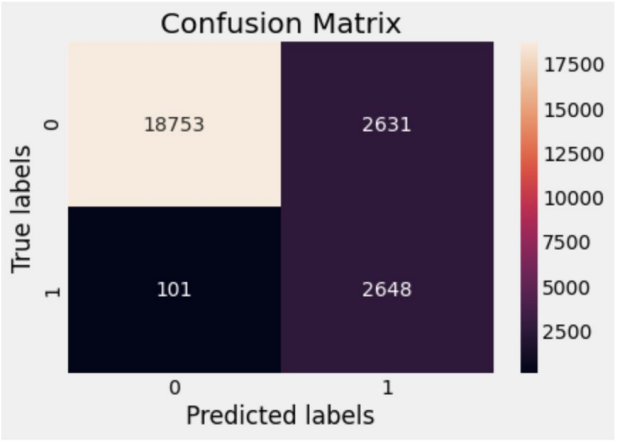


The AUC score compared to the PCA method is similar but the confusion matrix is less good in terms of False Negative and True positives. Moreover, LDA doesn't allow us to have two components and to plot anything since we have à binary classification.

4.9. Logistic Regression with Gradient Descent Algorithm

After applying Standard Gradient Descent Algorithm to Logistic Regression, we initialize the value of the learning rate to 0.003 and the number of iterations from n to 15000. We split our data into a train and a test set and we train the model on the train set. On the training set the accuracy and AUC are 0.9678, MCC is 0.9356. We find that the testing accuracy is 0.887, the AUC is 0.92 and the MCC is 0.646.

The Confusion matrix is below (**Figure)** and the ROC Curve is in **Appendix C.**



If we compare this confusion matrix with the previous result of Logistic Regression, we can see that the number of False negatives is higher (101 against 64 for ElasticNet) and the number of False Positives is increased by approximately 4,000. This prediction is not the most optimal and accurate possible. Other methods should be preferred.

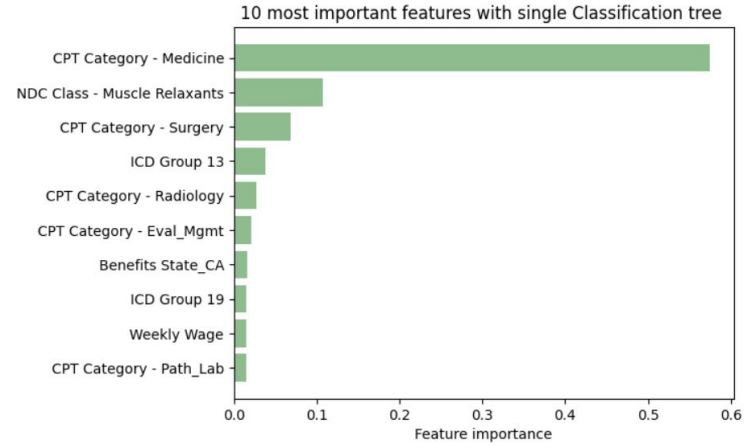
4.10. Decision tree and Random Forest

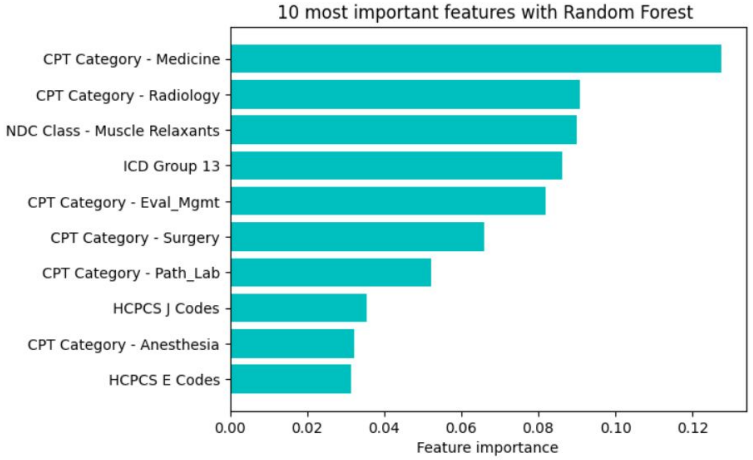
As we said in the method part, we tuned our hyperparameters in both the Classification Tree and Random forest method. Concerning the Classification tree parameters tuning gives a slightly less good model, with 0.901 AUC score for the final model against 0.906 for the default model. However, if we compare the

confusion matrix for each model (**Appendix E),** we can see that the tuned parameters model has less false negatives and more true positives than the default model, at the expense of false positives. The question here is, what is the best for the insurance. We guess that it's better to have less False positives, as we said prevention is better than cure, that's why we kept the tuned parameter model. Nevertheless, we know that Classification trees can sometimes overfit the data and are instables. That's why we did the Random forest method.

Concerning the results of our Random Forest model where we set the parameters, the performance is only slightly better with an AUC score of **0.926** against 0,924 for the default model, with a little more true positives 0 (see Appendix E). This illustrates the fact that the Random Forest model works really well without hyperparameters tuning.

Now we'll look at the 10 most important features in the Classification Tree and in the Random Forest:





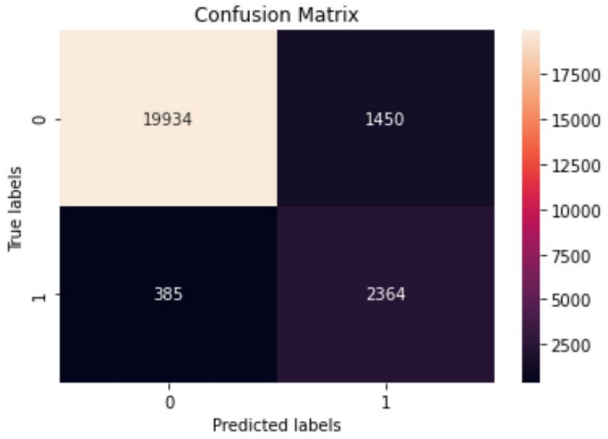
We can see redundant features in both techniques like the first one, but we can also see that in the classification the Medicine category is way more important, whereas in the Random Forest the features importance is more homogeneous. Finally, if we compare the 10 most important features we got with the Random forest method and those we got from the forward method, we only have 2 in common.

4.11. K-Nearest-Neighbours

For this KNN model we did, as said previously, an hyperparameters tuning to get a better performance of the model. And this setting of the parameters proved to be very useful since we gained almost 7 AUC score points as we can see in this table:

KNN with default parameters	**KNN with parameters tuning**
AUC score : 0.832	AUC score : 0.896
MCC score : 0.532	MCC score : 0.690
Accuracy score : 0.869	Accuracy score: 0.924

And if we look at our confusion matrix for the tuned parameter model we have not too bad results, even if other techniques give us better results:

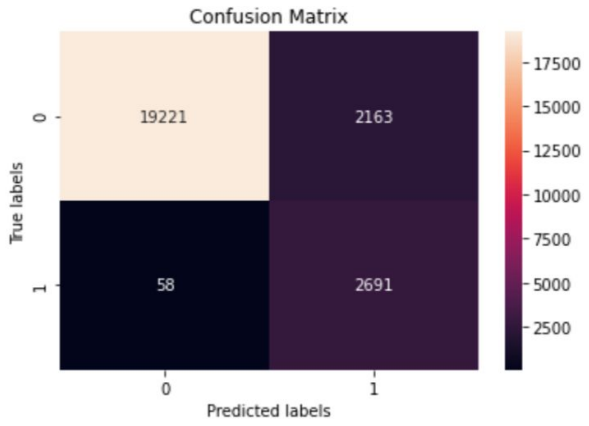


4.12. Support Vector Machine

As the previous models, for the Support Vector Machine we try to find the best parameters in order to have better performance. Unlike the tuning of the KNN parameters, with the combinaison that we found with our test, we only get a really small improvement. It is true that the model with the default settings already had a very high performance. Moreover, we wanted to test the 'linear' kernel which only requires the 'C' parameter. We did a loop with several values for C and got a performance almost as good as the tuned parameter model as you can see in this table :

SVM with default parameters	**Tuned SVM with Gaussian kernel**	**Tuned SVM with Linear kernel**
AUC score: 0.922 MCC score: 0.634	AUC score: 0.939 MCC score: 0.696	AUC score: 0.935 MCC score: 0.681

Our SVM model gives a very good prediction and is probably one of our best models. The confusion matrix associated with our tuned SVM model with the Gaussian kernel is also good with just a few false negatives:



4.13. Artificial Neural Network

We want to compare all Machine Learning methods (KNN, SVM or ANN) to predict that it is the most powerful method. We don't want to make a too complex ANN, with a binary classification problem, because otherwise it could overfit our data. So we use only one hidden layer. Concerning the choice of the batch size, the number of epochs and the number of iterations, we can see in the litterature a trade-off. One Epoch is when the dataset passes forward and backward through the neural network only once. Concerning the batch size, it has been observed in practice that when using a larger batch there is a significant degradation in the quality of the model, as measured by its ability to generalize. So for the size of the batch, we test 32, 64 and 128. The best is the smallest : 32. We initialize the number of epochs = 10. As we can see, there is no consensus about the choice of these parameters. We therefore tested different alternatives and kept the best one. We also use a validation set (20% of the train set) which provides an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters.

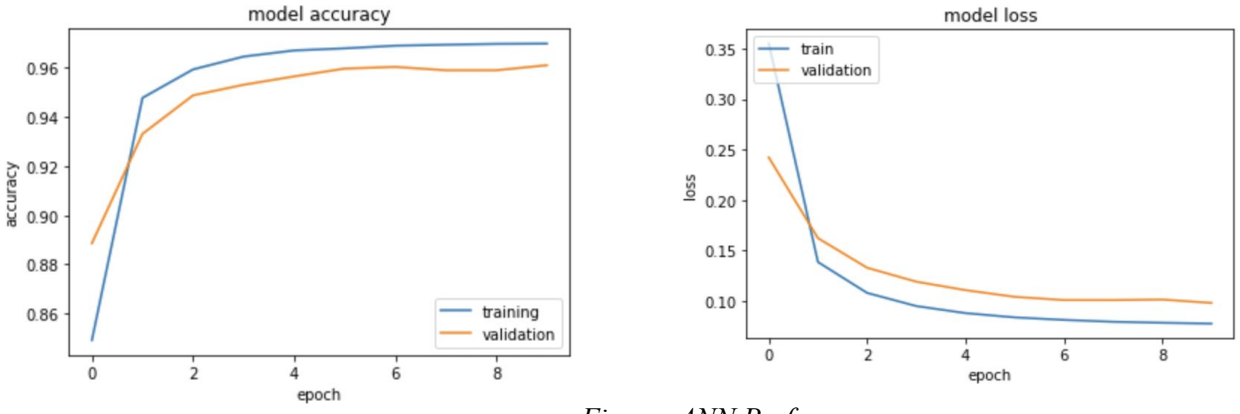
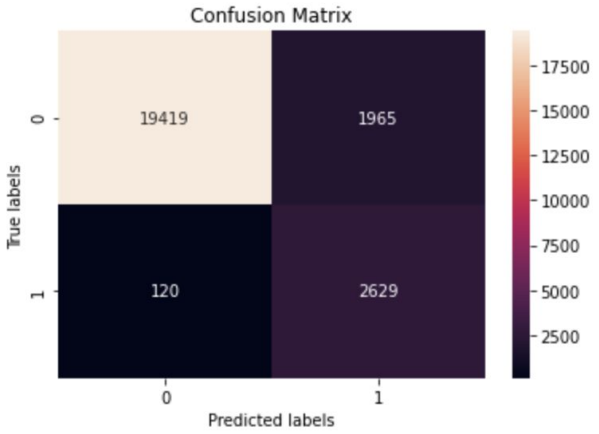


Figure: ANN Performance

Also, on the testing set the accuracy is 0.9136 and AUC is 0.9811. MCC is equal to 0.669. When we look at the confusion matrix (**Figure)**, the number of False negatives is equal to 120 and the number of False positives is equal to 1,965, which is pretty good comparing to the other method



4.14. Unsupervised Learning : K Means

We want to analyze our data, from a different point of view: unsupervised. As we have seen in the method part, we have realized the K-means algorithm with the number of clusters corresponding to our labels so K=2, but only on numerical variables. Here is the result with the application of Kmeans and with our standard features and labels (**Figure 1** and **Figure 2**)

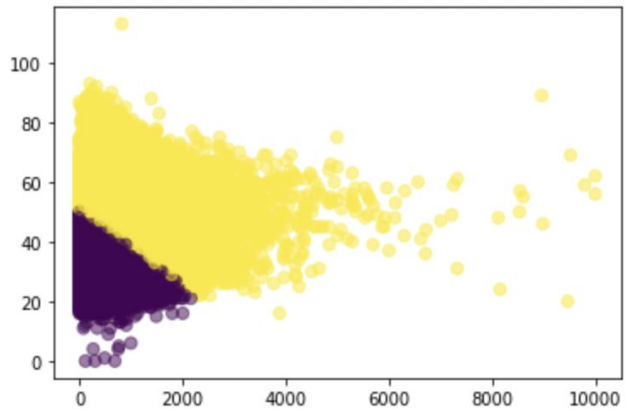


Figure 1 : K means on Age and Weekly Wage

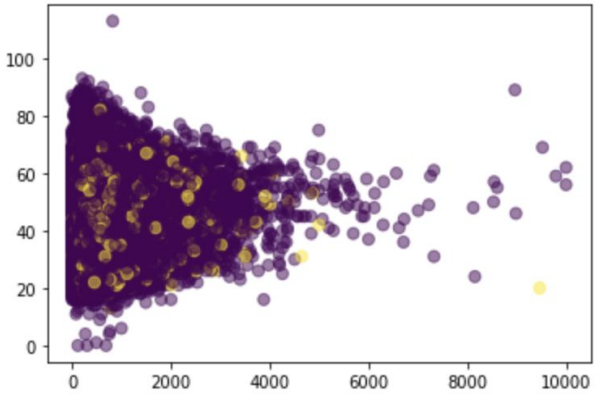


Figure 2 :Age and Weekly Wage

We also performed K-means on the optimal number of clusters suggested by the Silhouette Coefficient, K=3. (**Figure 3**)

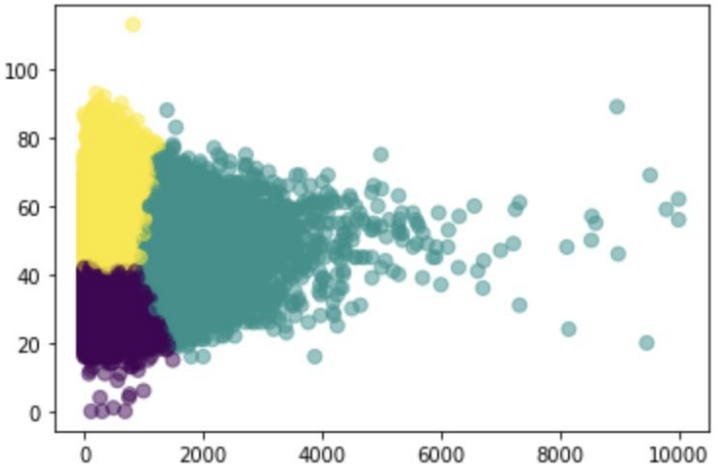


Figure 3 : K means on Age and Weekly Wage with K=3

When K= 2, we can clearly see 2 different clusters. The first cluster brings together the youngest individuals (between 20 and 40 years old), who earn the lowest salary per week. The second cluster regroups the largest part of the population of the distribution, older, 50 years old and over, who earns the highest salary.

When K=3, we can clearly see 3 different clusters. One cluster brings together the richest individuals in our distribution, regardless of age, and the other two clusters cut the population into the youngest and oldest individuals. With only these two numerical variables, it is almost impossible to find our 2 groups of "opioid consumers or not". It is a good approach for data mining purposes. We also decided to use PCA (with more than 2 variables) to test this algorithm. See the result in (**Appendix B**). We use PCA, n=2 components, so we can also visualize the data in the context of the true labels and predicted labels. But we know in the previous results that PCA with n=2 explained only 0.06971022 of the variance. So the prediction of cluster will be very poor in the graph

Table of performance of the models:

Method	**Training set**			**Testing set**		
Score	***Accuracy***	***AUC***	***MCC***	***Accuracy***	***AUC***	***MCC***
Logistic regression with l1-penalty	0.9731	0.9731	0.9462	0.9044	0.9357	0.6869
Logistic regression with elastic net	0.9733	0.9733	0.9467	0.9022	0.9347	0.6824
KNN	0.9222	0.9222	0.8513	0.9240	0.8961	0.6900
Decision Tree	0.9438	0.9438	0.8878	0.8646	0.9012	0.5990
Random Forest	1.	1.	1.	0.8816	0.9262	0.6448
SVM	0.9900	0.9900	0.9801	0.9080	**0.9389**	0.6957
Logistic Regression with Gradient Descent	0.9678	0.9678	0.9356	0.887	0.92	0.646
Artificial Neural Network	0.9699	-	-	0.9136	0.9811	0.669

5. Comments and conclusion

As we can see, the majority of models that we have used perform rather well to predict opioid abuse. The best one is the Artificial Neural Network with a AUC score of 0.9811. The second best is the Support Vector Machine with a AUC score of 0.9389, closely followed by the logistic regression with l1-penalty with a score of 0.9357. The logistic regression with elastic net and the Random Forest have also good performances. In comparison, the models that perform less well are KNN, Decision Tree and the Logistic Regression with Gradient Descent.

However, something that matters probably the most for this type of study is the number of False negatives, it is probably better to minimize the number of False negatives. And if we take a look at the confusion matrix for each model, the models that give the minimum number of false negatives are Random Forest with 44 FN, and SVM with 58 FN. Whereas the Artificial Neural Networks method gives 120 FN.

As for the feature selection, we observe that the 'usual' methods such as sequential feature selection (also often called forward feature selection) become practically infeasible when the dataset is large and the number of features is huge. In this case, the regularization methods, such as the Lasso penalty, are handy and provide sparse solutions with still showing good predictive performance.

In the future studies it can be useful to bias the cost function towards the False negatives. As we are working with a severe medical condition, it would be crucial if the opioid use abuse is not identified early. Another alternative would be to apply some other methods, such as for example Generalized Additive model, which could in order to capture better the non-linearities for some variables and simplify the interpretation of some of the variables, but it should be done on a smaller set of features as it could be time consuming.

In any case, the most performant models that we found as well as the most important features that we found in several methods, could be used to help, in real life specialists or even assurances, identify potential opioid addicts and do prevention.

References

Literature:

1. Chicco, D. Ten quick tips for machine learning in computational biology. *BioData Mining* 10, 35 (2017).

2. Defazio, Aaron & Bach, Francis & Lacoste-Julien, Simon. (2014). SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives. Advances in Neural Information Processing Systems. 2.

3. Fan, Rong-En & Chang, Kai-Wei & Hsieh, Cho-Jui & Wang, Xiang-Rui & Lin, Chih-Jen. (2008). LIBLINEAR: a library for large linear classification. Journal of Machine Learning Research. p.18

4. Hastie, Trevor & Tibshirani, Robert & Friedman, Jerome. (2008). The Elements Of Statistical Learning. Aug, Springer. 1. 10.1007/978-0-387-21606-5_7. p.125

5. Hastie, Trevor & Tibshirani, Robert & Wainwright, Martin. (2015). Statistical Learning with Sparsity: The Lasso and Generalizations. 10.1201/b18401. p.55

6. Jacob Huinker (2019) 'Using machine learning to predict prescription opioid misuse in patients' p. 30-34

7. Lo-Ciganic, Weihsuan & Huang, James & Zhang, Hao & Weiss, Jeremy & Kwoh, C. & Donohue, Julie & Gordon, Adam & Cochran, Gerald & Malone, Daniel & Kuza, Courtney & Gellad, Walid. (2020). Using machine learning to predict risk of incident opioid use disorder among fee-for-service Medicare beneficiaries: A prognostic study. PLOS ONE. p.9-10

8. López V., Fernández A., García S. , Palade V., Herrera F.. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics.Information sciences 250, 113-141

9. Md Mahmudul Hasan, Md. Noor-E-Alam, Mehul Rakeshkumar Patel, Alicia Sasser Modestino, Leon D. Sanchez, Gary Young (2019) 'A Big Data Analytics Framework to Predict the Risk of Opioid Use Disorder' p. 13, 15-16

10. Obradovic Ivana, "La crise des opioïdes aux Etats-Unis, d'un abus de prescriptions à une épidémie aiguë", Potomac Papers, n°35, Ifri, décembre 2018

11. Pohlmeier Winfried. Advanced Econometrics Lecture Notes (2019) p.244

12. Tibshirani Robert The Lasso method for variable selection in the cox model . Statistics in medicine. Vol. 16, 385—395 (1997)

Online resources:

1. https://towardsdatascience.com/feature-selection-using-regularisation-a3678b71e499
2. https://www.kaggle.com/enespolat/grid-search-with-logistic-regression?fbclid=IwAR3jHGoerY2cDoPidNs-G4VU5xkBvdw_xBPdMeINjHMBT1P_qy6e14YC37k

3. https://towardsdatascience.com/how-to-handle-missing-data-8646b18db0d4
4. https://towardsdatascience.com/feature-selection-using-regularisation-a3678b71e499
5. http://rasbt.github.io/mlxtend/user_guide/feature_selection/SequentialFeatureSelector/#:~:text=Sequential%20feature%20selection%20algorithms%20are,feature%20subspace%20where%20k%20%3C%20d.
6. https://towardsdatascience.com/gradient-descent-demystified-bc30b26e432a
7. https://www.knowledgehut.com/tutorials/machine-learning/hyperparameter-tuning-machine-learning
8. https://towardsdatascience.com/nothing-but-numpy-understanding-creating-binary-classification-neural-networks-with-e746423c8d5c
9. https://realpython.com/k-means-clustering-python/
10. https://towardsdatascience.com/random-forest-in-python-24d0893d51c0
11. https://medium.com/datadriveninvestor/k-nearest-neighbors-in-python-hyperparameters-tuning-716734bc557f

Instructions for script code and database :

You will find all the instructions to read the codes on Github by following this link :

https://github.com/flo-pritzy/Advanced-methods-in-big-data-THONNEAU-SAVONIK-PRITZY

Database :

1. The original database is : *opioid_use_data.csv*
2. The database after data pre-processing is : *cleaned_final.csv*

Scripts :

1. Data pre_pocessing.ipynb
2. Selection de variables -2.ipynb
3. modele.ipynb
4. Model_LDA.ipynb
5. Model_RF.ipynb
6. Models_KNN.ipynb
7. Models_SVM.ipynb
8. Forward_final.ipynb

APPENDIX

Appendix A :

- **Variable dictionary :**

- **ClaimID Unique**: Identifier for a claim
- **Accident DateID**: Number of days since the accident occurred from an arbitrary date
- **Claim Setup DateID**: Number of days since the Resolution Manager sets up the claim from an arbitrary date
- **Report To DateID:** Number of days since the employer notifies insurance of a claim from an arbitrary date
- **Employer Notification DateID**: Number of days since the claimant notifies employer of an injury from an arbitrary date
- **Benefits State**: The jurisdiction whose benefits are applied to a claim
- **Accident State**: State in which the accident occurred
- **Industry ID:** Broad industry classification categories
- **Claimant Age**: Age of the injured worker
- **Claimant Sex**: Sex of the injured worker
- **Claimant State**: State in which the claimant resides
- **Claimant Marital Status:** Marital status of the injured worker
- **Number Dependents:** Number of dependents the claimant has
- **Weekly Wage:** An average of the claimant's weekly wages as of the injury date.
- **Employment Status Flag:** *F — Regular full-time employee /P — Part-time employee /U — Unemployed/ S — On strike/ D — Disabled/ R — Retired/ O — Other/ L — Seasonal worker/ V — Volunteer worker/ A — Apprenticeship full-time/B — Apprenticeship part-time/ C — Piece worker*
- **RTW Restriction Flag:** A Y/N flag, used to indicate whether the employees responsibilities upon returning to work were limited as a result of his/her illness or injury.
- **Max Medical Improvement DateID:** DateID of Maximum Medical Improvement, after which further recovery from or lasting improvements to an injury or disease can no longer be anticipated based on reasonable medical probability.
- **Percent Impairment:** Indicates the percentage of anatomic or functional abnormality or loss, for the body as a whole, which resulted from the injury and exists after the date of maximum medical improvement
- **Post Injury Weekly Wage:** The weekly wage of the claimant after returning to work, post-injury, and/or the claim is closed.
- **NCCI Job Code:** A code that is established to identify and categorize jobs for workers' compensation.
- **Surgery Flag:** Indicates if the claimant's injury will require or did require surgery
- **Disability Status:***— Temporary Total Disability (TTD)/ — Temporary Partial Disability (TPD)/ — Permanent Partial Disability (PPD)/ — Permanent Total Disability (PTD)*
- **SIC Group:** Standard Industry Classification group for the client
- **NCCI BINatureOfLossDescription**: Description of the end result of the bodily injury (BI) loss occurrence
- **Accident Source Code:** A code identifying the object or source which inflicted the injury or damage.
- **Accident Type Group:** A code identifying the general action which occurred resulting in the loss
- **Neurology Payment Flag:** Indicates if there were any payments made for diagnosis and treatment of disorders of the nervous system without surgical intervention
- **Neurosurgery Payment Flag:** Indicates if there were any payments made for services by physicians specializing in the diagnosis and treatment of disorders of the nervous system, including surgical intervention if needed

- **Dentist Payment Flag:** Indicates if there were any payments made for prevention, diagnosis, and treatment of diseases of the teeth and gums
- **Orthopedic Surgery Payment Flag:** Indicates if there were any payments made for surgery dealing with the skeletal system and preservation and restoration of its articulations and structures.
- **Psychiatry Payment Flag:** Indicates if there were any payments made for treatment of mental, emotional, or behavioral disorders.
- **Hand Surgery Payment Flag:** Indicates if there were any payments made for surgery only addressing one or both hands.
- **Optometrist Payment Flag:** Indicates if there were any payments made to specialists who examine the eye for defects and faults of refraction and prescribe correctional lenses or exercises but not drugs or surgery
- **Podiatry Payment Flag:** Indicates if there were any payments made for services from a specialist concerned with the care of the foot, including its anatomy, medical and surgical treatment, and its diseases.
- **HCPCS A Codes — HCPCS Z Codes:** Count of the number of HCPCS codes that appear on the claim within each respective code group
- **ICD Group 1 — ICD Group 21:** Count of the number of ICD codes that appear on the claim w/in each respective code group
- **Count of the number of codes on the claim :** *— CPT Category — Anesthesia/ — CPT Category — Eval_Mgmt/ — CPT Category — Medicine/ — CPT Category — Path_Lab/ — CPT Category — Radiology/ — CPT Category — Surgery*
- **Count of the number of NDC codes on the claim within each respective code class —** *NDC Class — Benzo/ — NDC Class — Misc (Zolpidem)/ — NDC Class — Muscle Relaxants/ — NDC Class — Stimulants*
- **Opioids Used:** A True (1) or False (0) indicator for whether or not the claimant abused an opioid

Appendix B :

- **Removed Features :**

The list of removed features after applying the l1-norm penalty to the Logistic regression:

['Claimant Age', 'Post Injury Weekly Wage', 'Neurology Payment Flag',
'HCPCS M Codes', 'HCPCS Q Codes', 'HCPCS Y Codes', 'ICD Group 1',
'ICD Group 3', 'ICD Group 4', 'ICD Group 7', 'ICD Group 8',
'ICD Group 9', 'ICD Group 14', 'ICD Group 20', 'ICD Group 21',
'Benefits State_TX', 'Benefits State_PA', 'Benefits State_MD',
'Benefits State_IN', 'Benefits State_VA', 'Benefits State_AL',
'Benefits State_NV', 'Benefits State_RI', 'Benefits State_DC',
'Benefits State_KS', 'Benefits State_UT', 'Benefits State_NE',
'Benefits State_WV', 'Benefits State_ME', 'Benefits State_MT',
'Benefits State_LH', 'Benefits State_VT', 'Benefits State_WY',
'Benefits State_JA', 'Benefits State_ND', 'SIC Group_services',
'SIC Group_retail_trade', 'SIC Group_public_admi',
'Disability Status_TTD', 'Disability Status_PTD', 'Claimant Sex_M',
'Claimant Sex_F', 'Claimant Marital Status_U',
'Employment Status Flag_L', 'Employment Status Flag_A',
'Employment Status Flag_R', 'Employment Status Flag_B',
'Employment Status Flag_C', 'Employment Status Flag_D',
'NCCI BINatureOfLossDescription_Strain',
'NCCI BINatureOfLossDescription_Sprain',
'NCCI BINatureOfLossDescription_Skin',
'NCCI BINatureOfLossDescription_Foreign Body',
'Accident Type Group num_1', 'Accident Type Group num_4',
'Accident Type Group num_11', 'Accident Type Group num_12',
'Accident Type Group num_15', 'Accident Type Group num_14',
'Accident Type Group num_13', 'Accident Type Group num_16',
'Industry ID_14', 'Industry ID_13', 'Industry ID_16', 'Industry ID_12',
'Industry ID_18', 'Industry ID_9', 'Industry ID_0', 'Industry ID_1',
'Claimant Age squared', 'Weekly Wage squared', 'Weekly Wage cubic']

Appendix C :

Selected features

'Number Dependents', 'Weekly Wage', 'Percent Impairment',

'Surgery Flag', 'Orthopedic Surgery Payment Flag',

'Optometrist Payment Flag', 'HCPCS D Codes', 'HCPCS E Codes',

'HCPCS G Codes', 'HCPCS I Codes', 'HCPCS J Codes', 'HCPCS L Codes',

'HCPCS R Codes', 'HCPCS W Codes', 'ICD Group 6', 'ICD Group 11',

'ICD Group 12', 'ICD Group 13', 'ICD Group 18',

'CPT Category - Eval_Mgmt', 'CPT Category - Medicine',

'CPT Category - Path_Lab', 'CPT Category - Radiology',

'CPT Category - Surgery', 'NDC Class - Benzo',

'NDC Class - Misc (Zolpidem)', 'NDC Class - Muscle Relaxants',

'Benefits State_CA', 'Benefits State_FL', 'Benefits State_NY',

'Benefits State_CT', 'Benefits State_NJ', 'Benefits State_WI',

'Benefits State_MN', 'Benefits State_MA', 'Benefits State_GA',

'Benefits State_NC', 'Benefits State_CO', 'Benefits State_OR',

'Benefits State_WA', 'NCCI BINatureOfLossDescription_Other',

'NCCI BINatureOfLossDescription_Contusions',

'NCCI BINatureOfLossDescription_Lacerations/Scarring',

'NCCI BINatureOfLossDescription_Inflammation',

'Accident Type Group num_8', 'Accident Type Group num_9',

'Accident Type Group num_10', 'Industry ID_6', 'Claimant Age cubic'

Removed features

'Claimant Age', 'HCPCS Q Codes', 'HCPCS X Codes', 'HCPCS Y Codes',

'ICD Group 1', 'ICD Group 3', 'ICD Group 7', 'ICD Group 14',

'ICD Group 20', 'Benefits State_TX', 'Benefits State_PA',

'Benefits State_KS', 'Benefits State_ME', 'Benefits State_MT',

'Benefits State_LH', 'Benefits State_WY', 'Benefits State_JA',

'Benefits State_ND', 'SIC Group_retail_trade', 'SIC Group_public_admi',

'Disability Status_TTD', 'Disability Status_PTD', 'Claimant Sex_F',

'Claimant Marital Status_U', 'Employment Status Flag_L',

'Employment Status Flag_A', 'Employment Status Flag_R',

'Employment Status Flag_B', 'Employment Status Flag_C',

'Employment Status Flag_D', 'Accident Type Group num_1',

'Accident Type Group num_4', 'Accident Type Group num_12',

'Accident Type Group num_15', 'Accident Type Group num_14',

'Accident Type Group num_13', 'Accident Type Group num_16',

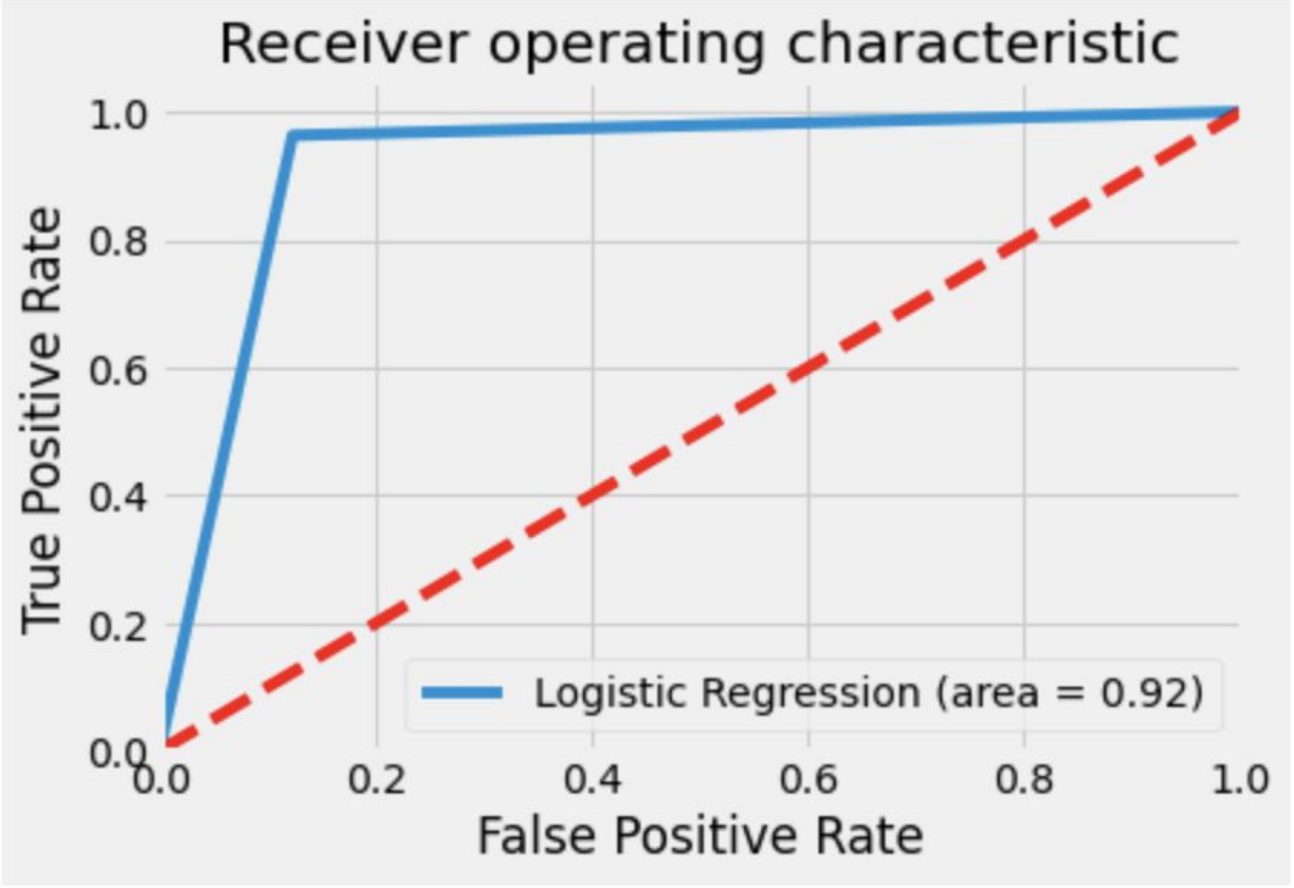
'Industry ID_14', 'Industry ID_13', 'Industry ID_16', 'Industry ID_12',

'Industry ID_18', 'Industry ID_0', 'Industry ID_1',

'Weekly Wage squared', 'Weekly Wage cubic'

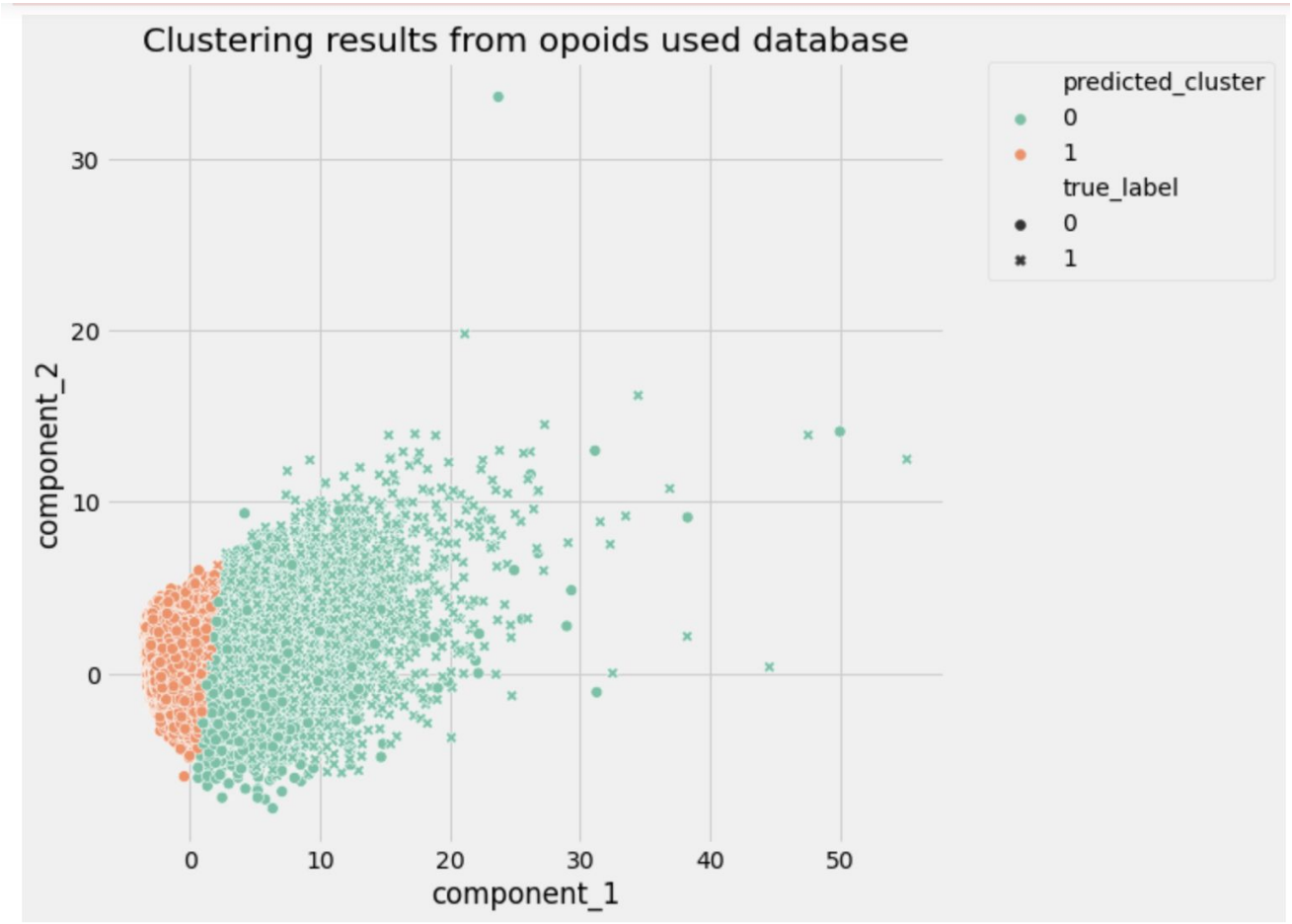
Appendix C :

- **ROC Curve of the Logistic Regression with Gradient Descent Algorithm :**

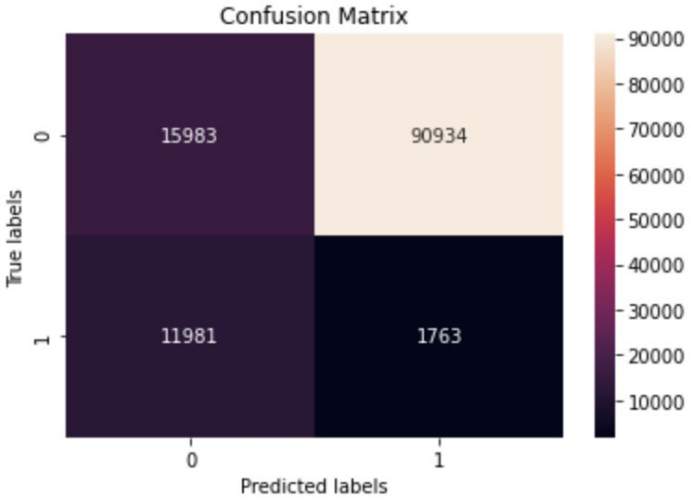


Appendix D :

- **K means with PCA, n=2 components**

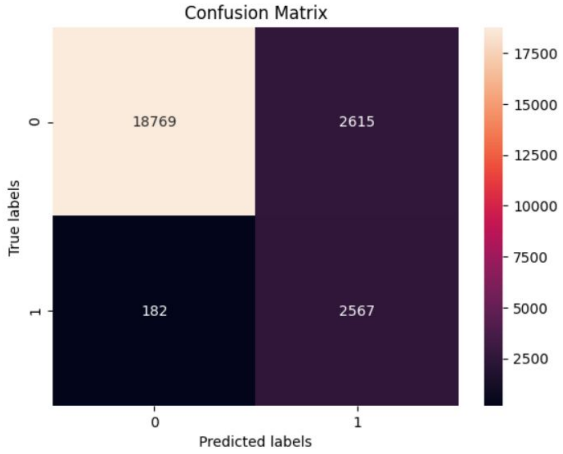


- **Confusion matrix :**

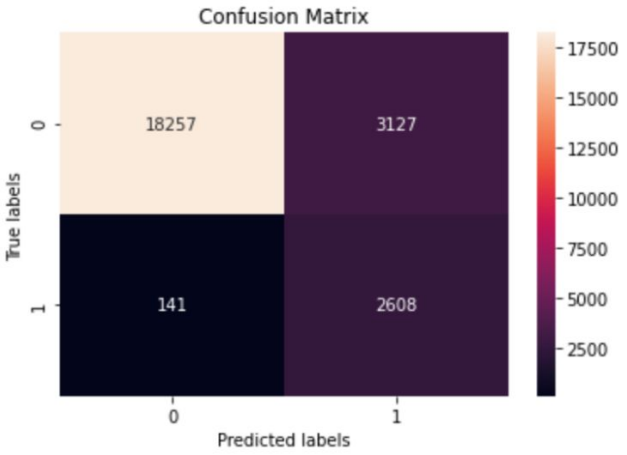


Appendix E:

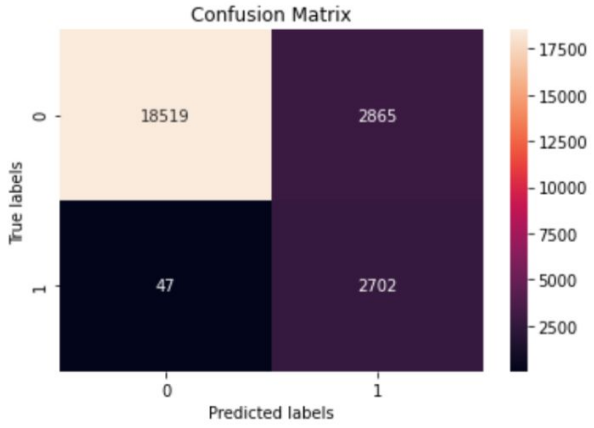
Default model(tree):



Tuned parameters model :



Default model(random forest):



Tuned parameters model :