IPSA
ÉCOLE D'INGÉNIEURS
DE L'AIR, DE L'ESPACE ET DE LA MOBILITÉ DURABLE

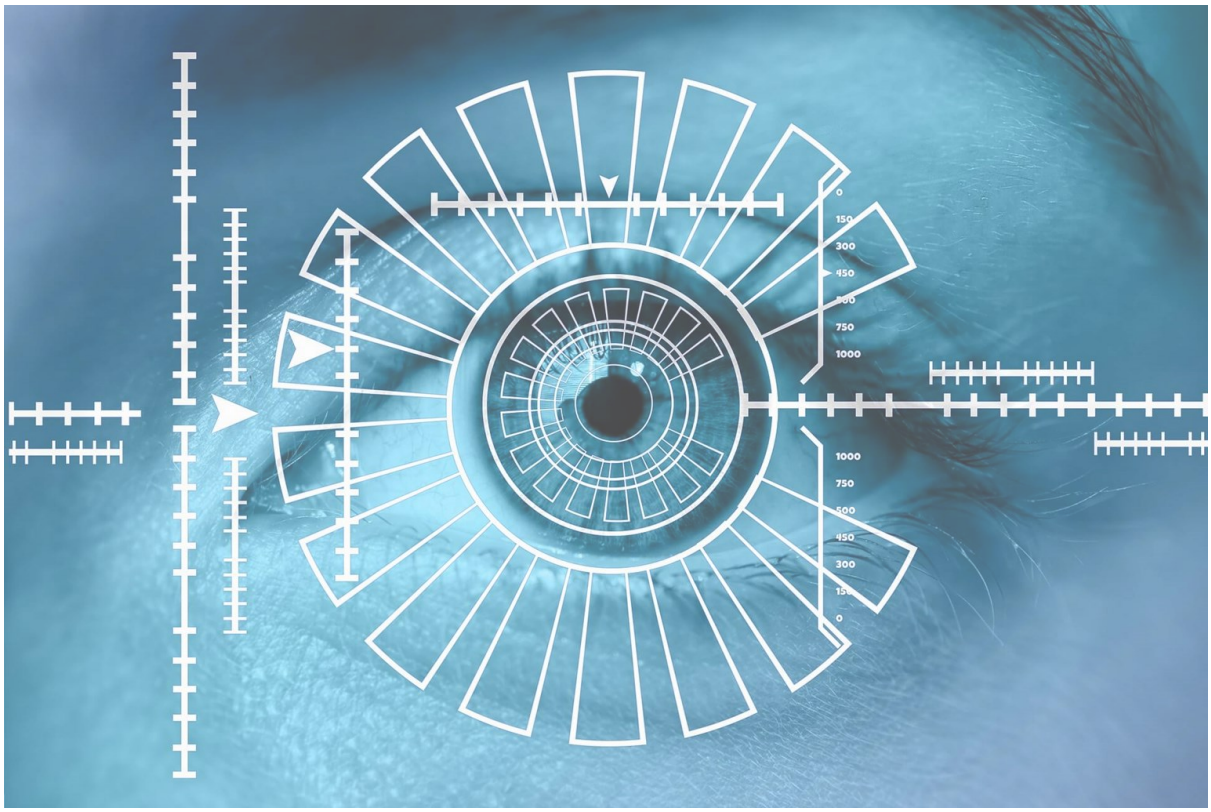
PROJECT MA513

Securing User Authentication

Côme Decaens / Elies Hatoum / Alexandre Pipet / Antony Priem / Paul Remacle

Subject 1

I. Table des matières



I. Introduction

In this project we will be talking about security user authentication with Keystroke dynamics.
Keystroke dynamics is the analyze of the behavior and pattern of individuals.
This analysis is done from the moment the user starts typing.

Here is the process:
When the user creates his password, a behavior analysis is created to understand how the user types his own password as well as his login and, if the security is high, other common words.
The password is asked to be repeated at the creation of an account, for example, as well as common words, which can also be asked later on when the user tries to connect again. This is done so that we can have a more precise analysis of the user's behavior and we can recognize him as he types his password multiple times. This will also allow us to identify a user even if his password changes because his keystroke dynamic will stay relatively the same.
This analysis allows a heightened security in the sense that if the user type his password and we detect a different keystroke dynamic, then the authentication will fail.

This double authentication is safe as it is very hard to recreate a habit even if we have the user's password.

This method of user recognition has been developed to address several issues related to traditional authentication methods. Here are some problems that have led to the development of keystroke dynamics:

Password vulnerabilities: Traditional password-based authentication systems are susceptible to various security threats, such as password guessing, brute-force attacks, and password sharing. Keystroke dynamics offer an additional layer of security by considering the unique way individuals' type, making it more challenging for unauthorized users to mimic the legitimate user's typing patterns.

User convenience and authentication speed: While passwords are a common form of authentication, they can be inconvenient for users to remember and enter, especially when strong and complex passwords are required. Keystroke dynamics provide a non-intrusive and transparent authentication method, as users can be identified based on their typing patterns without the need for explicit actions.

Continuous authentication: Traditional authentication methods, such as passwords or PINs, usually provide access after a single successful verification. Keystroke dynamics enable continuous authentication by continuously monitoring a user's typing behavior throughout a session. This helps to enhance security by detecting anomalies or unauthorized users even after the initial login.

Impersonation and fraud prevention: As cyber threats evolve, methods such as phishing and social engineering become more sophisticated. Keystroke dynamics add an extra layer of protection by focusing on a behavioral biometric that is difficult for attackers to replicate, making it harder for them to impersonate legitimate users.

Multifactor authentication enhancement: Keystroke dynamics can be integrated into multifactor authentication systems, providing an additional factor for user verification. Combining something a user knows (e.g., password) with something they are (e.g., typing behavior) and possibly something they have (e.g., a smart card) strengthens overall authentication security.

Remote authentication challenges: In an era where remote access and telecommuting are prevalent, traditional authentication methods may face challenges. Keystroke dynamics can be applied remotely without the need for specialized hardware, offering a solution for secure authentication in various remote access scenarios.

Biometric spoofing concerns: Some traditional biometric methods, such as fingerprint or facial recognition, may be vulnerable to spoofing. Keystroke dynamics are less susceptible to physical spoofing attempts since they rely on the unique way individuals' type, which is harder to replicate without advanced knowledge of the user's specific typing patterns.

User recognition in shared environments: In shared computing environments or situations where multiple users access the same device, distinguishing between users becomes crucial. Keystroke dynamics can help in accurately identifying individuals based on their typing patterns, even in shared or public computing environments.

Now we are going to explain our problem, present multiple solutions and the many strengths and weaknesses of the solutions that we are going to use.

II. Explanation of the problem

Nowadays, users tend to have weaker passwords since we have more and more accounts. We need another way to check the identity of the users.

We can have multi-factor authentications with a phone to some account, but we need an easier and more reliable way to authenticate.

To answer this problem of low strength password, multiple solutions are possible.

- Create multiple models per user when they login on different websites or applications.
- Add a second authentication on the phone or by mail.

And what interest us in this report:

- Have the user write multiple common words to add more precision to the way he types out words and analyze his keystroke dynamic more easily.

However, this solution has weaknesses as well as strengths that we are going to explain:

A. Weaknesses

It depends on the user's mood (the way he types) or the user's fatigue, the keyboard disposition and the user's knowledge on the equipment.
It has no way to differentiate between potential threats and an eventual sharing of an account.
You are monitored from the moment you are trying to type your password.

If there is a sophisticated Troyan horse, the attacker can eventually try to get the keystroke dynamic of the user to bypass this security. The research document also points out the capability for the "Wolves" to spoof the keystroke of a person. The difficulty for the attacker is dependent on the characteristics of the victim. Some are extremely hard to imitate while others are much easier.

B. Strengths

Even with the right password, we can detect if someone else is trying to log in.

Moreover, this method is not intrusive, it does not require something to link a fingerprint, a scanner but can still identify a specific user.

One more strength is the fact that analyzing Keystroke dynamics provides continuous authentication and can always monitor users by checking anomalies.

It is a very low-cost method since it only needs input from keyboard.

It adapts to the users over time.

We can see that the strengths outweigh the weaknesses, and we have realistic solutions that minimize the weaknesses presented earlier while maximizing the strengths.

Indeed, by trying out different algorithms we can determine which one is the most precise algorithm. By having more precise algorithms, we can secure the user's account while increasing the strength of the keystroke dynamics.

III. Possible solutions

We first have to find a good machine learning model. In our case we will use the Classifier model because it will predict the appurtenance of the input data to a certain class. It is not the case with Regressor that search to give a numerical value. For us, we need to find a user and not a value, so it is a class and so we use a Classifier.

Numerous classification models exist. We are going to try out multiple algorithms to determine which one is the most precise one.
So, our possible solutions are the following:

Decision Tree Classifier, Random Forest Classifier, Support Vector Machine, K-Nearest Neighbors, Naïve Bayes Classifier, Gradient Boosting Classifier, Multi-Layer Perception.

Other methods exist such as the LightGBM, CatBoost, Quadratic Discriminant Analysis and much more but because we have to make a choice, we only test the most common and used ones.

IV. Explanation of the algorithm

From our data, the first thing we will do is split it in training set and testing set :

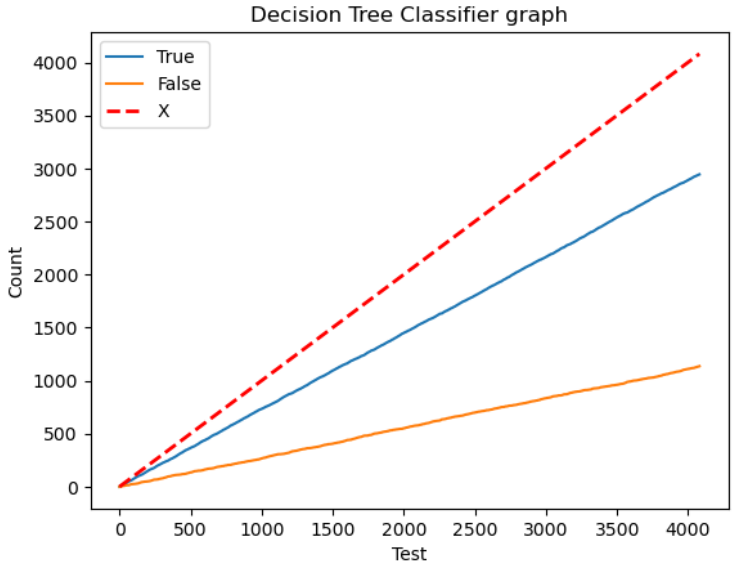
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split( samples, targets, test_size=0.1, random_state=42)
```

We will then try a first Algorithm:

Decision Tree Classifier: It makes decisions by going through the branches of the tree, ultimately assigning a class label or predicting a target variable for the input.

```
decision_tree_classifier = tree.DecisionTreeClassifier()
decision_tree_classifier.fit(X_train, y_train)
```

We make the predictions and obtain this graph:



There are 3 elements in this graph.

The X line is the perfect case with 100% accuracy.

The Blue curve represents the true answers. Each time a new input is tested, the output of the prediction is analyzed. If the output is equal to the true value, we do a +1, if not we do a +0. This incremental approach allows us to see the progression of the system and detect possible anomalies such as a steep increase of false answers.

The orange curve represents the false answers. It is the opposite of the blue curve; we do a +1 when the result is false. It could be argued that it is redundant to have 2 curves and not just one. The idea behind it is to find ways to obtain better results. Without this orange curve we would see the blue one bellow the red line and think the results are not so bad but by showing the failed part it shows the default of the system. It is a good help for the assessment of the quality of the system.

All of this makes the understanding of the performances of the system much easier by giving us a goal, the X line. We can know by a quick look the accuracy of the system. The closer the blue curve is to the red line the better it is.

Random Forest Classifier: It is a model that uses a lot of decision tree put together. Each tree vote for a decision and the final decision is based on the majority.

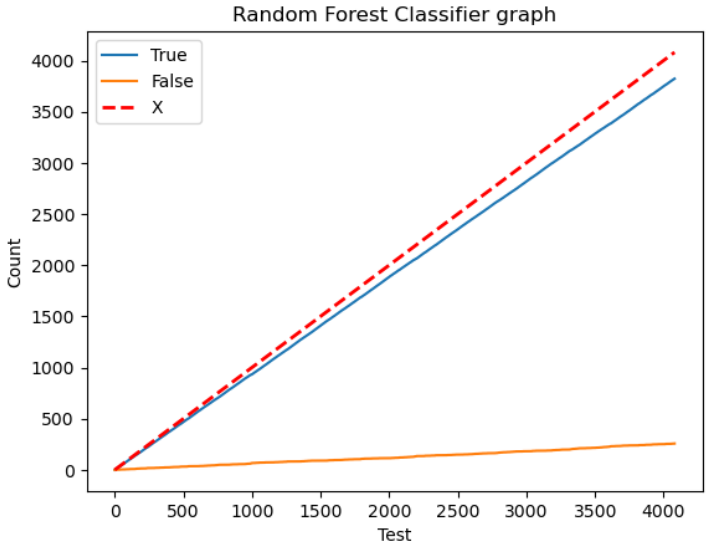
```
from sklearn.ensemble import RandomForestClassifier
randomforestC = RandomForestClassifier()
randomforestC.fit(X_train, y_train)
acc_RFC = randomforestC.score(X_test, y_test)
acc_RFC = acc_RFC*100

print("Random Forest Classifier")
print("accuracy: ",acc_RFC)
```

```
Random Forest Classifier
accuracy:  94.16666666666667
```

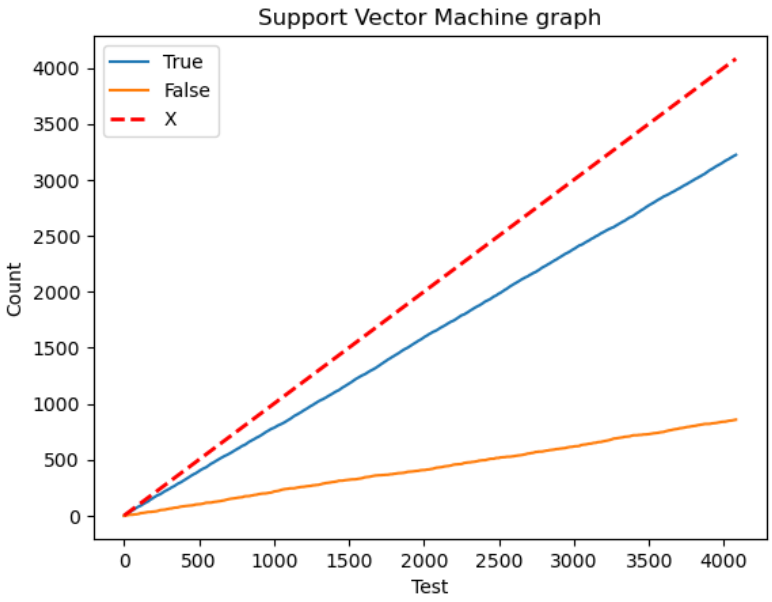
We obtain with it an accuracy of 94.167 %

And this graph with a very good true line:



Support Vector Machine (SVM): This supervised model is used for regression and classification tasks. It finds the best “line” to separate the data points in different classes. What we call Support Vector are the points closest to the line.

We find with SVM a success rate of 88.0. SVM is more used in complex and multidimensional system, so our data may be not fit for this model.

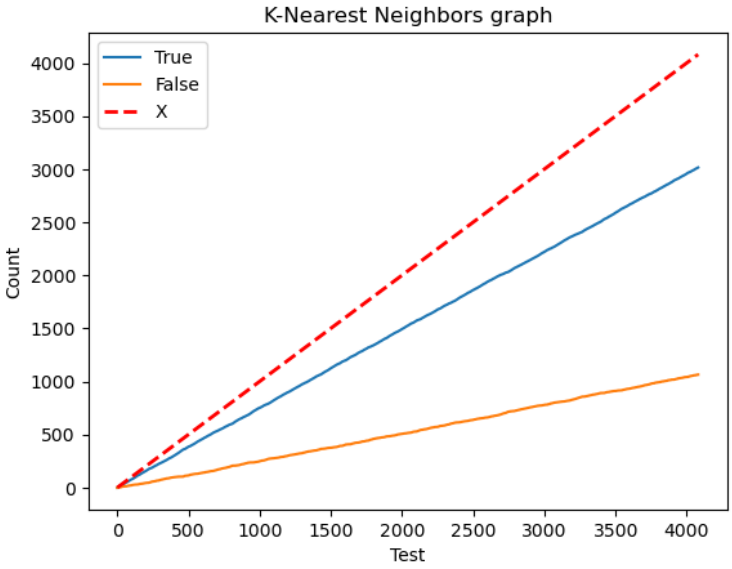


The number of errors is higher than with Random Forest

K-Nearest Neighbors (KNN) : This model as SVM can do tasks related to regression and classification. It makes predictions by identifying the k nearest data points in a space and what are their influence on the other points. It measures the distance between point to classify which is the similarity between points.

```
K nearest neighbors
accuracy:  75.63725490196079
```

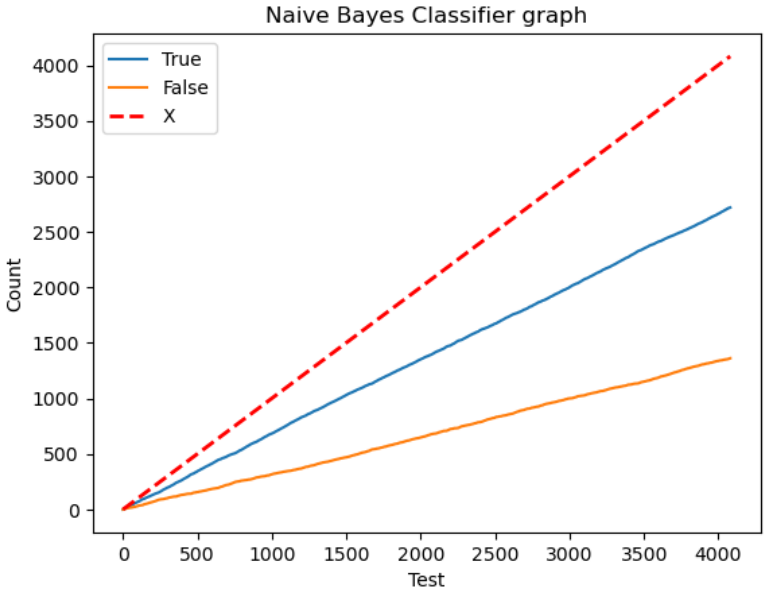
We obtain an accuracy of 75.64 % and this graph:



The results of the graph are similar to the KNN.

Naives Bayes Classifier : This Classifier works by assuming that all features are independent from each other. Then it calculated the probability of a data point belonging to a certain class by combining the probabilities of each of its features individually.

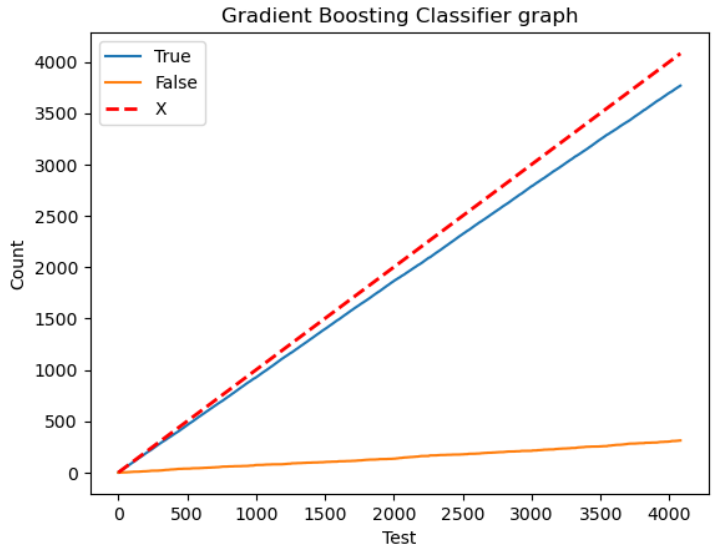
```
Naive Bayes Classifier
accuracy:  67.1078431372549
```



The number of True and False are closer than ever, it is not a good method for us.

Gradient Boost Classifier (GBC) is a ensemble of machine learning techniques. It builds a predictive model by combining the output of weak predictor (usually decision tree). It is iterative so each new tree will correct error made by the whole ensemble so far.

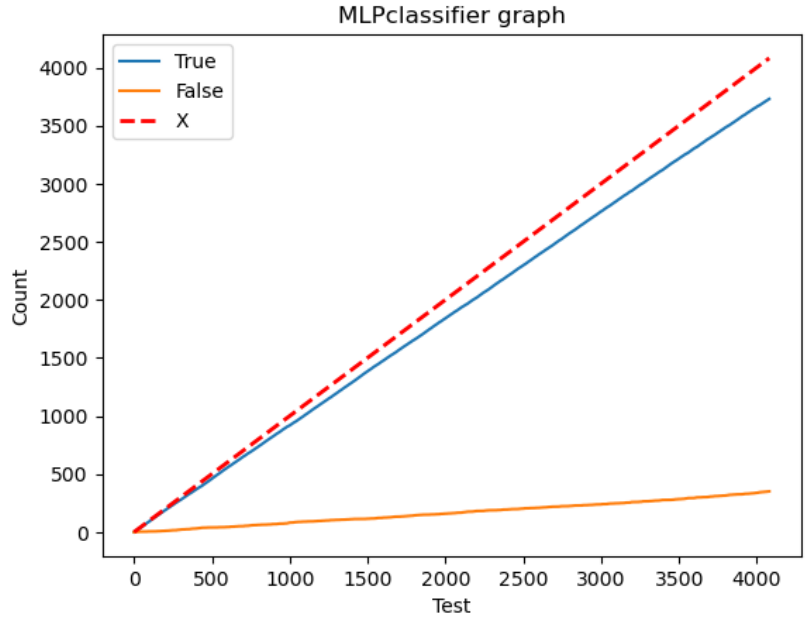
```
Cross-Validation Scores: [0.91339869 0.91230937 0.92075163 0.91285403 0.91448802]
Mean Accuracy: 0.9147603485838779
```



The results are very good especially in the beginning of the tests.

Multi-Layer Perceptron (MLP) is a neural network. It contains interconnected nodes which are associated with weights. The model learns by adjusting the weights during the training.

```
Multi-Layer Perceptron
accuracy:  92.1078431372549
```

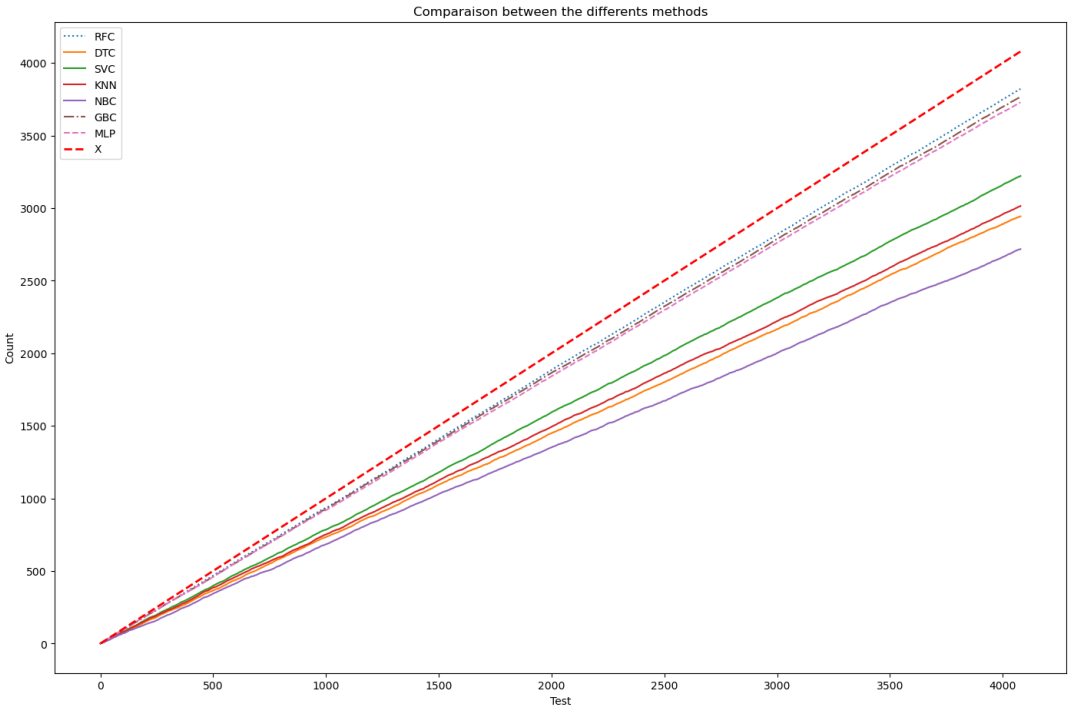


Overall, very good results and only 8% of error.

V. Conclusion

When comparing all our algorithms, we observe that 3 model are above the others :

- Random Forest
- Gradient boosting
- Multi-Layer Perceptron



We can note that both Random Forest and Gradient Boosting are using several dozens of decision trees. MLP is the only one using the network of models which has permitted him to capture complex pattern.

All of these showcases how important it is to choose the appropriate model to the appropriate data.