IMT Atlantique
Technopôle de Brest-Iroise - CS 83818
29238 Brest Cedex 3
Téléphone : +33 (0)2 29 00 13 04
Télécopie : +33 (0)2 29 00 10 12
URL : **www.imt-atlantique.fr**

Development project in Machine Learning

BONNEM Emma
emma.bonnem@imt-atlantique.net

FERREIRA Kevin
kevin.ferreira@imt-atlantique.net

TRANCHANT Elias
elias.tranchant@imt-atlantique.net

Date d'édition : 4 décembre 2022
Version : 2.0

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Sommaire

1. Introduction

Machine learning is a computer programming technique that uses statistical probabilities to give computers the ability to learn by themselves without explicit programming. It is used in many fields, notably in health and finance.

In this project, we will focus on classification models that use labeled data to predict to which class an object belongs to. We will mainly talk about binary classification, where the goal is to distinguish whether an object belongs to a class or not.

We will be using the following classifications methods : *Linear SVC*, *Naive Bayes*, *SGD* classification, *KNeighbors* classification and *Random Forest* classification.

The aim of this project is to work in group by using a git repository, which is a version control software allowing each member to work individually on a single project at the same time.

2. The project and the data

This project aims to provide a global function for learning a model regardless of the dataset. For each dataset we implemented the following workflow :

Step 1 : Import the dataset by using pandas library.

Step 2 : A dataset may contain multiple missing values. In that situation, we have to clean the dataset. Clean the data and perform pre-processing by replacing the missing values by average for float64 or median values for objects. Then, we center and normalize the data.

Step 3 : Split the dataset between training set and test set using train_test_split from sklearn.

Step 4 : Train the model with different classifiers.

Step 5 : Compare and validate the models.

We used **feature selection** to have a faster and more correct prediction on our data. Before normalizing data, the columns with a variance under a certain trehold are deleted. The chosen threshold is 0.05 to not delete too many columns and distort the results.

To test our workflow, we used two different datasets.

First dataset : Banknote Authentication Dataset

This dataset provided a total of 1372 records (individuals) with 5 numeric variables (dimensions) of different authentic and counterfeit banknotes. The four first columns are data used to predict whether a note is genuine or not. In our dataset, these columns don't contain any labels. So before using this dataset, we have to label each column by their names which are variance, skew, curtosis, and entropy.

Second dataset : Chronic Kidney Disease

The objective of the dataset is to predict whether a patient have chronic kidney disease or not. The data was taken over a 2-month period in India with 400 individuals and 25 medical predictor variables (dimensions) such as : Blood pressure, specific gravity, albumin, sugar, red blood cells... The classification feature is called 'ckd' for chronic kidney disease. This data needs cleaning and pre-processing : NaNs has been replaced by average for float64 values and median values for objects. Also the numeric features has been forced into floats.

3. The estimation

3.1. Implementation

Among all the methods we know for this kind of problem, we chose five different ones as to test their respective efficiency. Here we will present those methods and explain how they work. We used *sklearn* to implement each method.

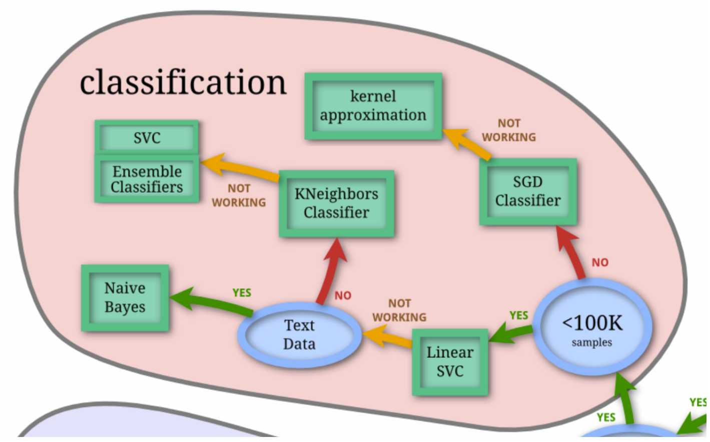
First of all, to effectively use the methods, we created in our python file a function named *train_validate_model*. For a given method, called *estimator*, we apply these steps :

```
# Model training
    estimator.fit(X_train, y_train)
    # Model accuracy based on the training set
    model_accuracy_score = estimator.score(X_train, y_train)
    # Model prediction
    y_pred = estimator.predict(X_test)
    # Model accuracy based on the prediction
    cm = confusion_matrix(y_test, y_pred)
```

Estimator is the chosen classifier, *model_accuracy_score* and *confusion_matrix* are values that define the accuracy of our model. We will get back to this later.

3.2. Methods

To select which methods to test, we used a flowchart available on the sklearn website. Here is a sample of this graph.



The first one is the **Linear SVC** (Linear Support Vector Classification) :

> Similar to SVC with parameter kernel='linear'. The SVC classifier method comes from the SVM (support vector machine) theory. SVMs' main purpose is to partition datasets into a large number of classes in order to discover a Maximum Marginal Hyperplane (MMH), which can be done in two steps : Support Vector Machines will initially iteratively build hyperplanes that best distinguish the classes. It will then select the hyperplane that best separates the classes. In our case we will apply it on a two class dataset.

Then we have the **Naive Bayes classification** method :

> Naive Bayes classifiers are a collection of classification algorithms based on **Bayes' Theorem**. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

The **SGD** (stochastic gradient descent) classifier :

> This classification function relies on the gradient descent algorithm which starts from a random point on a function and travels down its slope in steps until it reaches the lowest point of that function.

The **Kneighbors** classifier :

> The k-nearest neighbors algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. While it can be used for either regression or classification problems, it is typically used as a classification algorithm, working off the assumption that similar points can be found near one another.

The **Random forest** classifier :

> Assuming your dataset has “m” features, the random forest will randomly choose “k” features where k < m. Now, the algorithm will calculate the root node among the k features by picking a node that has the highest information gain. After that, the algorithm splits the node into child nodes and repeats this process “n” times. Now you have a forest with n trees. Finally, you’ll combine the results of all the decision trees present in your forest and thus obtain a classification.

3.3. Metrics

Three metrics are used to compare each estimator.

Model accuracy : This value, ranging from 0 to 1, represents the accuracy of our model based on our training data set. This metric allows us to check if the defined model is coherent with the data it was created from. The formula is :

$$\text{Model accuracy} = \frac{\text{Amount of correct predictions}}{\text{Total amount of training data}}$$

Confusion matrix : This matrix allows us to know the error rate of our prediction on the testing set. It represents the number of predicted and actual class for each value. With three classes $C1$, $C2$ and $C3$ we would have this :

$$\begin{array}{ccc} & & Predicted \\ & & \begin{array}{ccc} C1 & C2 & C3 \end{array} \\ Actual & \begin{array}{c} C1 \\ C2 \\ C3 \end{array} & \begin{bmatrix} 129 & 0 & 1 \\ 1 & 54 & 2 \\ 0 & 4 & 69 \end{bmatrix} \end{array}$$

Cross-validation score : This score is computed by divising our training set into 3 parts, before training the model on one of them and testing it on the two others. Three score are outputed, for each of the 3 divided parts of the set. For example, if the list [1.0 0.5 0.25] is outputed, it means that training the model on the first part allowed the program to correctly predict the two other parts, meanwhile training on the other two parts gave a bad prediction.

4. Results

4.1. Dataset *data_banknote_authentication*

The following output is obtained from the *data_banknote_authentication* dataset.

```
Linear SVC
Model accuracy : 0.9854227405247813
Confusion matrix :
[[384   2]
 [  2 298]]
Cross val score : [0.98689956 0.98253275 0.99122807]

Naive Bayes
Model accuracy : 0.8425655976676385
Confusion matrix :
[[333  53]
 [ 71 229]]
Cross val score : [0.88646288 0.81659389 0.82894737]

SGD Classifier
Model accuracy : 0.9752186588921283
Confusion matrix :
[[386   0]
 [ 23 277]]
Cross val score : [0.98253275 0.97816594 0.96929825]

KNeighbors Classifier
Model accuracy : 1.0
Confusion matrix :
[[385   1]
 [  0 300]]
Cross val score : [0.99126638 0.97816594 1.        ]

Random Forest
Model accuracy : 1.0
Confusion matrix :
[[381   5]
 [  6 294]]
Cross val score : [0.99126638 0.96943231 0.98684211]
```

The worst classifier is certainly the *Naive Bayes* one, with the lowest model accuracy, high values in the anti-diagonal of the confusion matrix, and low cross-values scores.

The other classifiers are relatively equivalent in term of efficiency. We can notice that the *KNeighbors* classifier has perfect model accuracy and near perfect confusion matrix, and that its cross-value scores are very high. It seems like it's the best algorithm for this dataset.

4.2. Dataset *kidney_disease*

```
Linear SVC
Model accuracy : 1.0
Confusion matrix :
[[116  13]
 [  2  69]]
Cross val score : [0.97014925 0.95522388 0.87878788]

Naive Bayes
Model accuracy : 0.95
Confusion matrix :
[[118  11]
 [  0  71]]
Cross val score : [0.92537313 0.98507463 0.93939394]

SGD Classifier
Model accuracy : 0.99
Confusion matrix :
[[125   4]
 [  1  70]]
Cross val score : [0.98507463 0.97014925 0.92424242]

KNeighbors Classifier
Model accuracy : 0.955
Confusion matrix :
[[110  19]
 [  1  70]]
Cross val score : [0.94029851 0.86567164 0.86363636]

Random Forest
Model accuracy : 1.0
Confusion matrix :
[[128   1]
 [  1  70]]
Cross val score : [0.98507463 0.97014925 0.92424242]
```

Unlike with the previous dataset, there is no clear loser here. All the classifiers have a model accuracy equal or very close to 1.0, an anti-diagonal of their confusion matrix close to 0 and cross-value scores close to 1.0.

Even if this dataset contains more dimensions than the previous one, it also has way less individuals. It seems like that, in this case, every methods are correct.

If we were to choose a perfect algorithm for this dataset, both the *Linear SVC* and *Random Forest* classifiers are returning near perfect results.

5. Good programming practices

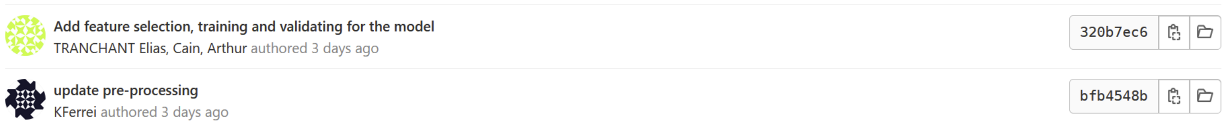
The main objective of this project was to learn to develop a python classification file using **Gitlab**. Even using Gitlab could be named a “good programming practice” for it is one of the most performant group work tools with Github. It enables the team to remotely develop files while retaining every version of the files.

Gitlab is a **version manager** for **Git** :

Git is a free and open source software for distributed version control : tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development. Git allows a wide variety of branching strategies and workflows.

A version manager such as Gitlab allows you to remember each modification of every file, why it happened and by whom it was modified. However to work effectively as a group even on Gitlab there are some practices that are useful :

- Developed **commentaries** on every commit :
  Every time a member of the group has to "commit" (update his progress on the file on Git and then Gitlab) he is allowed to leave a commentary. For the different commit to make sense, one must be careful to give enough detail in his commit about what modifications he made on the file. In the following screenshot you can see 3 different commit made by the members of our group on the "ML-projet" repository.



Add feature selection, training and validating for the model
TRANCHANT Elias, Cain, Arthur authored 3 days ago
320b7ec6

update pre-processing
KFerrei authored 3 days ago
bfb4548b

- One good programming practice is to not **push** and **commit** our work for every little modification. The "commit" stage is more when you have made some kind of progress or breakthrough in the project. Even more when the project gathers a lot of members, a lot of commits can make the overview of the project difficult to understand and untangle.
- We used unit testing to see if any of our function had problems. This way, we were able to identify bugs and malfunctions.
- Finally a good programming practice that our team has put into place is for each member of the group to work on different **branches**. A personnel branch will regroup all the commits of one developer and allow a different point of view on the progress he made in the project. It also allows us to see what kind of participation each member brought and in what quantity.

On the other hand, to develop a proper code by working in a group besides respecting the programming language, it is necessary that all members use the same standards. For example, the variables used must be named with an explicit name as well as the functions. Moreover, these functions must be commented on to explain their usefulness.

These rules allow a better reading of the code by all the members of the team and a better practicability because a more readable code, more commented, is much easier to modify in case of bugs or addition of new functions.

6. Conclusion

As we gave our thoughts on the results of our program in section 4, this conclusion will be focused on the group work in itself.

This project allowed us to improve our organisation of a group project based on git. We had to use the *commit* and *push* functions to each be part of this work. We learned a lot about git and its usage.

Apart from that, we used a large variety of algorithm through *sklearn*, and compared them with metrics seen in class. This is an important part of working on data : knowing which solution is the best by comparing each one on a part or the entirety of the set. This allows to find the best model and to have the most correct conclusions.

Thanks to Gitlab our team was able to set up an efficient programming environment and then implement the code for cleaning and testing the data. The use of the software Git will prove to be a great competence in the future for remote collaboration.

OUR WORLDWIDE PARTNERS UNIVERSITIES - DOUBLE DEGREE AGREEMENTS

3 CAMPUS, 1 SITE

IMT Atlantique Bretagne–Pays de la Loire – **http://www.imt-atlantique.fr/**

Campus de Brest
Technopôle Brest-Iroise
CS 83818
29238 Brest Cedex 3
France
T +33 (0)2 29 00 11 11
F +33 (0)2 29 00 10 00

Campus de Nantes
4, rue Alfred Kastler
CS 20722
44307 Nantes Cedex 3
France
T +33 (0)2 51 85 81 00
F +33 (0)2 99 12 70 08

Campus de Rennes
2, rue de la Châtaigneraie
CS 17607
35576 Cesson Sévigné Cedex
France
T +33 (0)2 99 12 70 00
F +33 (0)2 51 85 81 99

Site de Toulouse
10, avenue Édouard Belin
BP 44004
31028 Toulouse Cedex 04
France
T +33 (0)5 61 33 83 65

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

© IMT Atlantique, 2021
Imprimé à IMT Atlantique
Dépôt légal : Septembre 2017
ISSN : 2556-5060