ENSIIE

Machine Learning Project

Handwritten Digit Classifier

Mame Diarra Toure
Sandrine Siga Sene

Enseignant :
Mrs. Mougeot

23/10/2020

3 novembre 2020

Table des matières

1 Introduction

The aim of this project is to study the ability of several machine learning algorithms for building a handwritten digit classifier. The dataset contains a sample of handwritten digits. Each digit is described by an array of 8x8 pixels in normalized grayscale. Each line of the dataset is composed 64 explanatory variables each of them representing a gray level and the target variable being the digit. We are going to start by having a first look of our dataset, then we're gonna try out multiple ML algorithms and evaluate their performance in order to choose the most suitable one for our problem and finally we're going to proceed with a variable selection to find out which pixels are more important for the digit recognition and see if selecting them can improve our learning algorithms

2 The dataset

The whole dataset is composed of 1797 observations each oberservation corresponding to a digit from 0 to nine

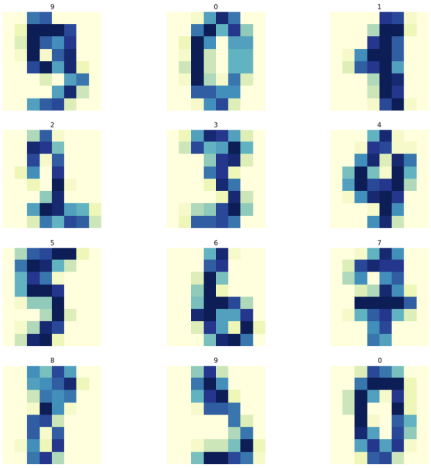


FIGURE 1 – Representation of the digits in our dataset

```
1 X
```

	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixe
0	0.0	0.0	7.0	15.0	13.0	1.0	0.0	0.0	0.0	8.0	...	
1	0.0	0.0	11.0	12.0	0.0	0.0	0.0	0.0	0.0	2.0	...	
2	0.0	2.0	9.0	15.0	14.0	9.0	3.0	0.0	0.0	4.0	...	
3	0.0	0.0	6.0	14.0	4.0	0.0	0.0	0.0	0.0	0.0	...	1
4	0.0	1.0	8.0	12.0	15.0	14.0	4.0	0.0	0.0	3.0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
358	0.0	2.0	10.0	12.0	16.0	8.0	0.0	0.0	0.0	4.0	...	
359	0.0	0.0	5.0	14.0	11.0	8.0	0.0	0.0	0.0	4.0	...	
360	0.0	0.0	4.0	14.0	15.0	6.0	0.0	0.0	0.0	5.0	...	
361	0.0	0.0	4.0	10.0	13.0	6.0	0.0	0.0	0.0	1.0	...	
362	0.0	0.0	2.0	10.0	7.0	0.0	0.0	0.0	0.0	0.0	...	

363 rows × 64 columns

FIGURE 2 – X=The array of pixels

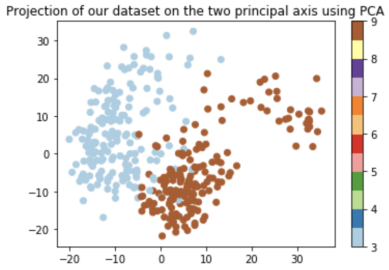


FIGURE 3 – Projection of our dataset on the two principal axis using PCA

Since we are dealing with binary classification, **We decided to work with the digit 9 and 3 because they are quite similar and that will enable us to see if our algorithms are truly efficients** When we subset the row corresponding to our two chosen digits we obtain a dataset composed of 361 observations that we are going to divide into a training and testing set but first lets have a first look into our data We have **180 observations for the the digit 3 and 183 for the digit 9. We see that the classes are balanced and that's an important information because it helps us decide which metrics to use to evaluate our models (since the classes are balancd we can at first look at the accuracy here).**

3 The Machine Learning algorithms

3.1 Gaussian Naive Bayes Classifier

Naive Bayes is a simple technique for constructing classifiers : models that assign class labels to problem instances, represented as vectors of feature. Naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the color, roundness, and diameter features.When dealing with continuous data, a typical assumption is that the continuous values associated with each class are distributed according to a normal (or Gaussian) distribution. hence the name gaussian naive bayes. **We applied the GNB classifier to our dataset and we obtained,in average, over a 20 splits K-Fold an accuracy of 91.81 percent**

```
Accuracy of Naive bayes classifier: 91.81
%
```

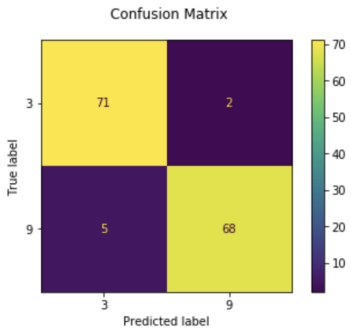


FIGURE 4 – Accuracy and confusion matrix of GNB classifier

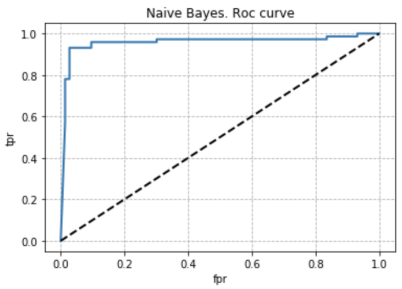


FIGURE 5 – ROC curve of the Naive bayes

3.2 Linear Discriminant Analysis classifier

The idea behind LDA is simple. It was proposed by a statistician called Ronald Fisher.The Fisher's propose is basically to maximize the distance between the mean of each class and minimize the spreading within the class itself. Thus, we come up with two measures : the within-class and the between-class. However, this formulation is only possible if we assume that the dataset has a Normal distribution. LDA basically projects the data in a new linear feature space, obviously the classifier will reach high accuracy if the data are linear separable which is quite the case for our data if you look at the figure 3 in section 2. **We applied the LDA classifier to our dataset and we obtained,in average, over a 20 splits K-Fold an accuracy of 98.07 percent**

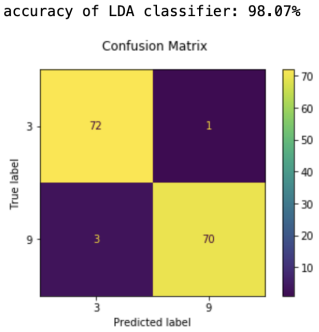


FIGURE 6 – Accuracy and confusion matrix of LDA classifier

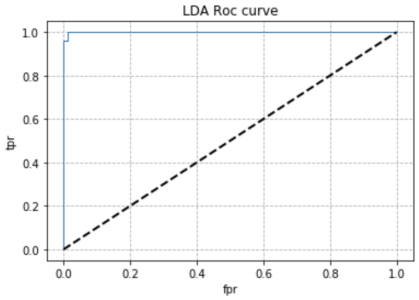


FIGURE 7 – ROC curve of the LDA classifier

3.3 Logistic Regression classifier

Logistic regression is a classification algorithm, used when the value of the target variable is categorical in nature. Logistic regression is most commonly used when the data in question has binary output, so when it belongs to one class or another. Logistic regression transforms its output using the logistic sigmoid function to return a probability value.We expect our classifier to give us a set of outputs or classes based on probability when we pass the inputs through a prediction function and returns a probability score between 0 and 1 **We applied the Logistic regression classifier to our dataset and we obtained,in average, over a 20 splits K-Fold an accuracy of 98.63 percent**

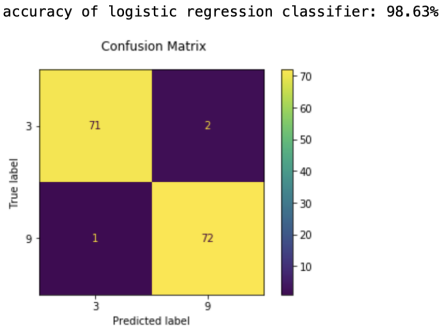


FIGURE 8 – Accuracy and confusion matrix of LR classifier

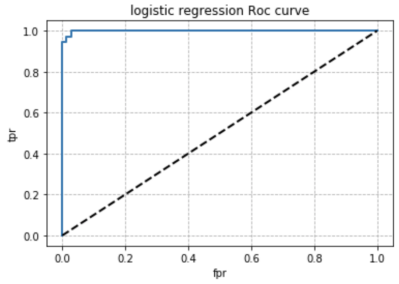


FIGURE 9 – ROC curve of the LR classifier

3.4 K-Nearest Neighbours classifier

he k-nearest neighbors classifier(k-NN) is a non-parametric method proposed by Thomas Cover used for classification the input consists of the k closest training examples in the feature space. the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.**Before proceeding wih classification here we had to tune in the hyperparameter k corresponding to the numbe of neighbors we are going to consider. The greatest accuracy on the testing set was obtained with k=7 We applied the KNN classifier to our dataset and we obtained,in average, over a 20 splits K-Fold an accuracy of 99.17 percent**

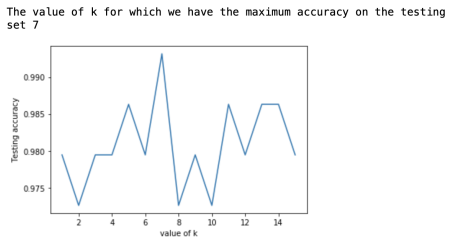


FIGURE 10 – choice of the number of neighbors k

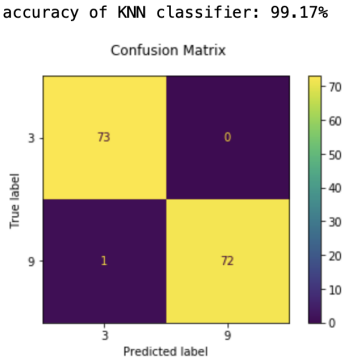


FIGURE 11 – Accuracy and confusion matrix of KNN classifier

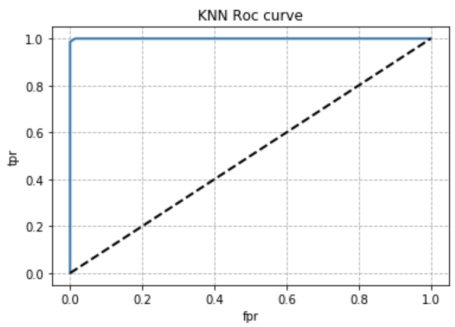


FIGURE 12 – ROC curve of the KNN classifier

3.5 Decision tree classifier

A Decision Tree is a simple representation for classifying examples. It is a Supervised Machine Learning algorithm where the data is continuously split according to a certain parameter.Using the decision algorithm, we start at the tree root and split the data on the feature that results in the largest information gain (IG) (reduction in uncertainty towards the final decision). In an iterative process, we can then repeat this splitting procedure at each child node until the leaves are pure. This means that the samples at each leaf node all belong to the same class. Before implementing the decision tree classifiier we are going to tune the hyperparameters. Hyperparameters are important because they directly control the behaviour of the training algorithm and have a significant impact on the performance of the model is being trained. First we look if the hyperparameter is relevant to our model or not by seeing how it affects the training set AUC and the testing set AUC. If it is, indeeed, releavnt we then pick the value that maximises our AUC score for our model

— min sample leaf :specifies the minimum number of samples required to split an internal node

— max depth :The maximum depth of the tree.The theoretical maximum depth a decision tree can achieve is one less than the number of training samples, for obvious reasons we must not reach that point, one big reason being overfitting. Note here that it is the number of training samples and not the number of features because the data can be split on the same feature multiple times.

— min sample split :specifies the minimum number of samples required to be at a leaf node.

— max feature : the number of features to consider each time to make the split decision. Let us say the the dimension of your data is 50 and the maxfeature is 10, each time you need to find the split, you randomly select 10 features and use them to decide which one of the 10 is the best feature to use. When you go to the next node you will select randomly another 10 and so on.

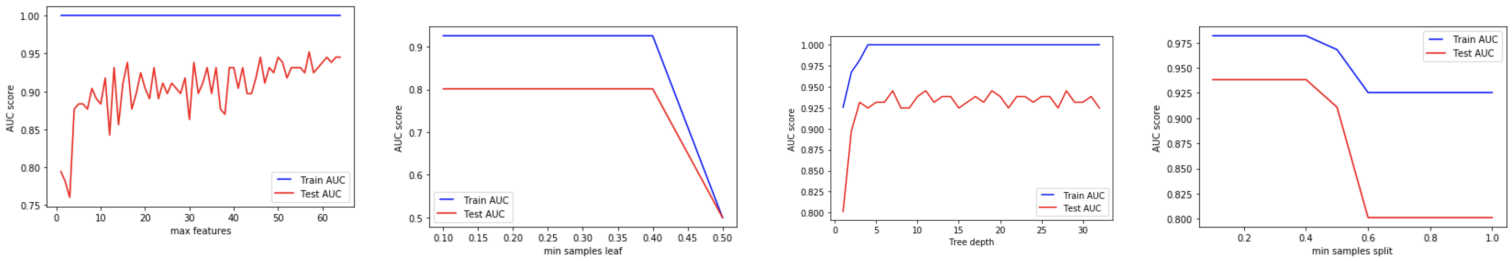
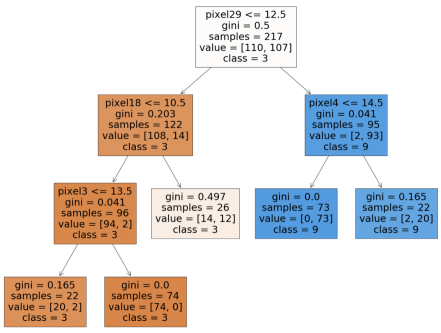


FIGURE 13 – Effects of the hyperparameters on the training and testing set

We applied the decision tree classifier to our dataset and we obtained,in average, over a 20 splits K-Fold an accuracy of 88.71 percent



our decision tree

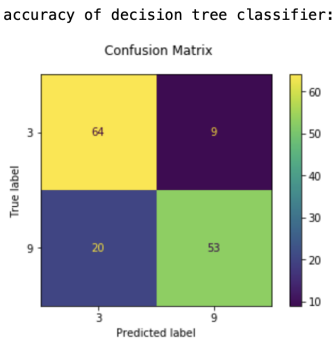


FIGURE 14 – Accuracy and confusion matrix decision classifier

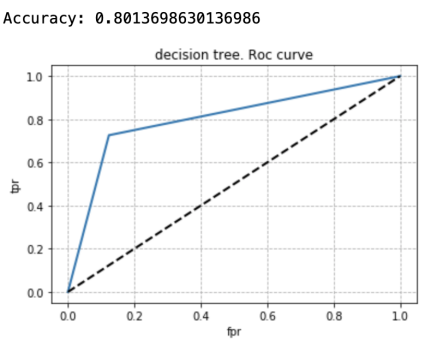


FIGURE 15 – ROC curve of the decision tree classifier

3.6 Bagging classifier

Bagging refers to bootstrap aggregation (repeated sampling with replacement and perform aggregation of results to be precise), which is a general purpose methodology to reduce the variance of models, in this case, they are decision trees.A Bagging classifier is an **ensemble** meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction. As We did for decision trees, we also tuned in the hyperparameter of the classifier : max features, max samples and n estimators. You can find more about that in the attached jupyter notebook. **We applied the bagging classifier to our dataset and we obtained,in average, over a 20 splits K-Fold an accuracy of 97.27 percent**

```
accuracy of bagging classifier: 97.27%
```

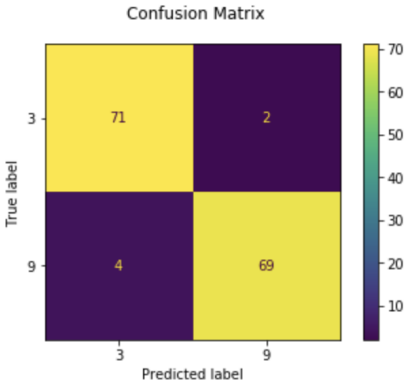


FIGURE 16 – Accuracy and confusion matrix of bagging classifier

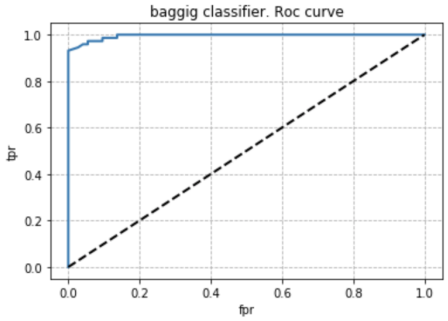


FIGURE 17 – ROC curve of the bagging classifier

3.7 Random Forest

It is an ensemble tree-based learning algorithm. The Random Forest Classifier is a set of decision trees from randomly selected subset of training set. It aggregates the votes from different decision trees to decide the final class of the test object. The RandomForestClassifier is trained using bootstrap aggregation, where each new tree is fit from a bootstrap sample of the training observations $z_i = (x_i, y_i)$. The out-of-bag (OOB) error is the average error for each z_i calculated using predictions from the trees that do not contain z_i in their respective bootstrap sample. This allows the RandomForestClassifier to be fit and validated whilst being trained

As We did for decision trees, we also tuned in the hyperparameter of the classifier : max features, max depth and n estimators. You can find more about that in the attached jupyter notebook. **We applied the Random forest algorihtm to our dataset and we obtained,in average, over a 20 splits K-Fold an accuracy of 98.18 percent**

```
accuracy of random forest classifier: 98.18%
```

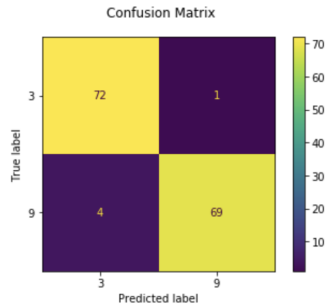


FIGURE 18 – Accuracy and confusion matrix of RF classifier

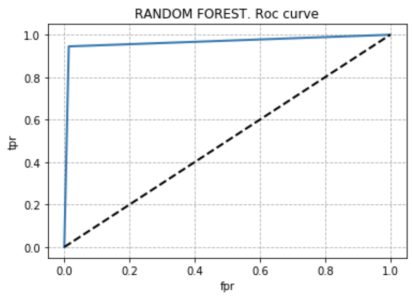


FIGURE 19 – ROC curve of the RF classifier

3.8 Adaboost

AdaBoost, short for Adaptive Boosting, is a machine learning meta-algorithm It can be used in conjunction with many other types of learning algorithms to improve performance. The output of the other learning algorithms ('weak learners') is combined into a weighted sum that represents the final output of the boosted classifier. AdaBoost is adaptive in the sense that subsequent weak learners are tweaked in favor of those instances misclassified by previous classifiers. AdaBoost (with decision trees as the weak learners) is often referred to as the best out-of-the-box classifier

As We did for decision trees, we also tuned in the hyperparameter of the classifier : learning rate and n estimators. You can find more about that in the attached jupyter notebook. **We applied the Random forest algorihtm to our dataset and we obtained,in average, over a 20 splits K-Fold an accuracy of 98.64 percent**

```
accuracy of adaboost classifier: 79.70%
```

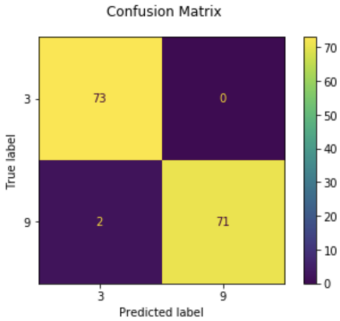


FIGURE 20 – Accuracy and confusion matrix of Adaboost classifier

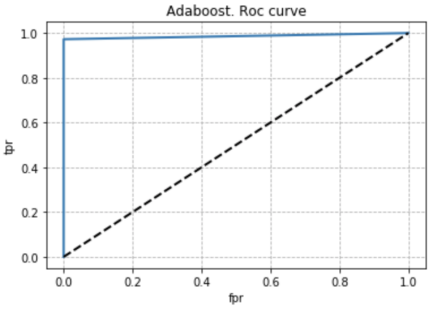


FIGURE 21 – ROC curve of the Adaboost classifier

4 comparison of the different models and our final choice

Now that we've performed multiple machine learning algorithms on our dataset we want to pick the best one, meaning the one that optimize our evaluation metrics. We decided to work with two metric the accuracy and the AUC. For each model, we've performed a Kfold cross validation each time storing the value of our evaluation metrics in an array. We then ploted the corresponding Boxplots

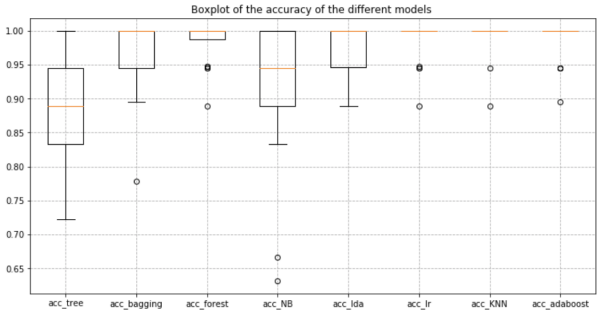


FIGURE 22 – Comparing the models using the accuracy

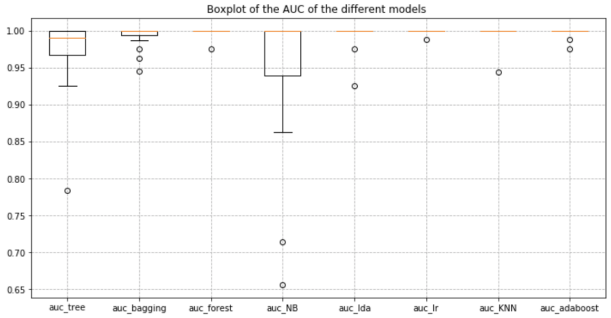


FIGURE 23 – comparing the models using the AUC

According to these boxplots, for both metrics 5 out of our 8 models perform almost perfectly. That is quite understandable. Indeed as you can see on the figure 3 section 2 the classes are neatly separated meaning that it is not hard to distinguish in what class an observation belongs hence it is quite easy for our models. We can thus pick any of these 5 (Random Forest,LDA, Logistice Regression, KNN, Adaboost) to be our classifier. However due to parameter tuning the random forest annd the adaboost are a little bit more time consuming Hence we decided to use KNN as our classifier for this digit recognition since it gives us the highest accuracy (99.17 percent) among the 3 classifier left

5 Going Further :Variables seletion

Even though our models perform generally very well, we can try to see if they can be improved with variable selection. As we can see in the picture in the introduction, it seems like not all pixels are relevant to determine the written digit. Hence we are going to proceed with feature selection and see if we can improve the results of our learning algorithms. We chose to use univariate feature selection Univariate feature selection works by selecting the best features based on univariate statistical tests. It can be seen as a preprocessing step to an estimator. Scikit-learn exposes feature selection routines as objects that implement the transform method : Select K Best removes all but the k highest scoring features. When we performed feature seletion selectiong the 10 best features we saw that in general all classifiers performances are worse except for the K-Nearest Neighbors Classifier. So in this case we saw that the feature selection didnt influence much our choice for a classifier. However it showed us that are assumption that all pixels are not relevant maybe false. To go further we could try to see the performance of our learning algorithms by increasingly selecting more features.

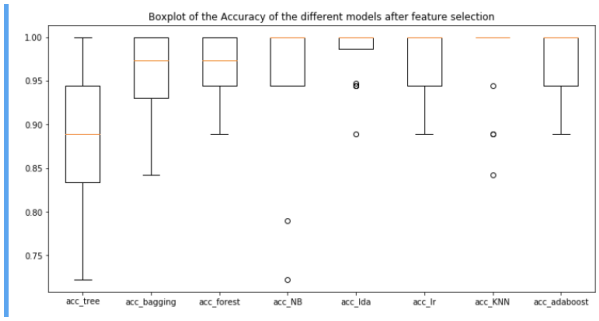


FIGURE 24 – Caption