



TÉLÉCOM PARIS

SD-TSIA210

MACHINE LEARNING

---

## Project Report

---

*Author:*  
Geoffroy MATEU

*Supervisor:*  
Florence D'ALCHÉ-BUC

# Contents

<b>I</b>	<b>Introduction</b>	<b>5</b>
<b>II</b>	<b>Binary Classification - Seismic-bumps Dataset</b>	<b>7</b>
<b>1</b>	<b>Informations on the dataset</b>	<b>8</b>
<b>2</b>	<b>Observation of the dataset</b>	<b>9</b>
<b>3</b>	<b>Preprocessing the data</b>	<b>10</b>
3.1	Clear the data . . . . .	10
3.2	One-hot encoding . . . . .	10
3.3	Scaling . . . . .	10
<b>4</b>	<b>Classification of the data</b>	<b>11</b>
4.1	SVC . . . . .	11
4.2	KNN, NN, Random Forest, XGBoost . . . . .	11
<b>5</b>	<b>Rebalancing the data</b>	<b>12</b>
5.1	Up-sampling the label '1' . . . . .	12
5.2	Random Forest . . . . .	12
5.3	Neural Networks . . . . .	13
<b>6</b>	<b>Conclusion</b>	<b>14</b>
<b>III</b>	<b>Multi-class Classification - Handwritten numerals</b>	<b>15</b>
<b>7</b>	<b>Informations on the dataset</b>	<b>16</b>
<b>8</b>	<b>Observation of the dataset</b>	<b>17</b>
<b>9</b>	<b>Preprocessing the data</b>	<b>18</b>
<b>10</b>	<b>Classification of the data</b>	<b>19</b>
10.1	Size of the hidden layer . . . . .	19
10.2	Activation function . . . . .	20
10.3	Learning rate tuning . . . . .	21
10.4	Optimizer tuning . . . . .	22
10.5	Batch size tuning . . . . .	23
<b>11</b>	<b>Results</b>	<b>24</b>
<b>IV</b>	<b>Regression : Communities and Crime</b>	<b>26</b>
<b>12</b>	<b>Informations on the dataset</b>	<b>27</b>
<b>13</b>	<b>Observation of the dataset</b>	<b>28</b>

<b>14 Preprocessing the data and regressors used</b>	<b>29</b>
<b>15 Results</b>	<b>30</b>



# Part I

## Introduction

The goal of this project is to study three datasets by doing on each of them a Binary Classification, a Multi-Class Classification and a Regression respectively on :

- Seismic-bumps Dataset : <https://archive.ics.uci.edu/ml/datasets/seismic-bumps>
- Multiple Features Dataset : <https://archive.ics.uci.edu/ml/datasets/Multiple+Features>
- Communities and Crime Dataset : <https://archive.ics.uci.edu/ml/datasets/Communities+and+Crime>

## Part II

# Binary Classification - Seismic-bumps Dataset

# Informations on the dataset

The data describe the problem of high energy (higher than  $10^4$  J) seismic bumps forecasting in a coal mine. Data come from two of longwalls located in a Polish coal mine.

There are 2485 instances of 19 attributes with no missing attribute values.

The goal is to predict the last attribute which displays whether a high energy seismic bump occurred in the next shift ("hazardous-state = "1") or no energy seismic bump occurred in the next shift ("non-hazardous state" = "0").

Here is the description of the attributes found in the dataset :

1. seismic: result of shift seismic hazard assessment in the mine working obtained by the seismic method (a - lack of hazard, b - low hazard, c - high hazard, d - danger state);
2. seismoacoustic: result of shift seismic hazard assessment in the mine working obtained by the seismoacoustic method;
3. shift: information about type of a shift (W - coal-getting, N -preparation shift);
4. genergy: seismic energy recorded within previous shift by the most active geophone (GMax) out of geophones monitoring the longwall;
5. gpuls: a number of pulses recorded within previous shift by GMax;
6. gdenergy: a deviation of energy recorded within previous shift by GMax from average energy recorded during eight previous shifts;
7. gdgpuls: a deviation of a number of pulses recorded within previous shift by GMax from average number of pulses recorded during eight previous shifts;
8. ghazard: result of shift seismic hazard assessment in the mine working obtained by the seismoacoustic method based on registration coming form GMax only;
9. nbumps: the number of seismic bumps recorded within previous shift;
10. nbumps2: the number of seismic bumps (in energy range  $[10^2, 10^3)$ ) registered within previous shift;
11. nbumps3: the number of seismic bumps (in energy range  $[10^3, 10^4)$ ) registered within previous shift;
12. nbumps4: the number of seismic bumps (in energy range  $[10^4, 10^5)$ ) registered within previous shift;
13. nbumps5: the number of seismic bumps (in energy range  $[10^5, 10^6)$ ) registered within the last shift;
14. nbumps6: the number of seismic bumps (in energy range  $[10^6, 10^7)$ ) registered within previous shift;
15. nbumps7: the number of seismic bumps (in energy range  $[10^7, 10^8)$ ) registered within previous shift;
16. nbumps89: the number of seismic bumps (in energy range  $[10^8, 10^{10})$ ) registered within previous shift;
17. energy: total energy of seismic bumps registered within previous shift;
18. maxenergy: the maximum energy of the seismic bumps registered within previous shift;
19. class: the decision attribute - "1" means that high energy seismic bump occurred in the next shift ("hazardous state"), "0" means that no high energy seismic bumps occurred in the next shift ("non-hazardous state").

# Observation of the dataset

Here are the first inputs of the dataset :

	seismic	seismoacoustic	shift	energy	gpuls	gdenergy	gdipuls	ghazard	nbumps	nbumps2	nbumps3	nbumps4	nbumps5	nbumps6	nbumps7	nbumps89	energy	maxenergy	class
0	b'a'	b'a'	b'N'	15180.0	48.0	-72.0	b'a'	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	b'0'	
1	b'a'	b'a'	b'N'	14720.0	33.0	-70.0	-79.0	b'a'	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	2000.0	2000.0	b'0'
2	b'a'	b'a'	b'N'	8050.0	30.0	-81.0	-78.0	b'a'	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	b'0'
3	b'a'	b'a'	b'N'	28820.0	171.0	-23.0	40.0	b'a'	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	3000.0	3000.0	b'0'
4	b'a'	b'a'	b'N'	12640.0	57.0	-63.0	-52.0	b'a'	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	b'0'

We can see several things. First there is data on byte form which will need to be cleared. Especially the *class* attribute.

We can see that the number of label 0 and label 1 is very unbalanced which may present an issue because the algorithms will learn how to predict 0 label much better way than label 1.

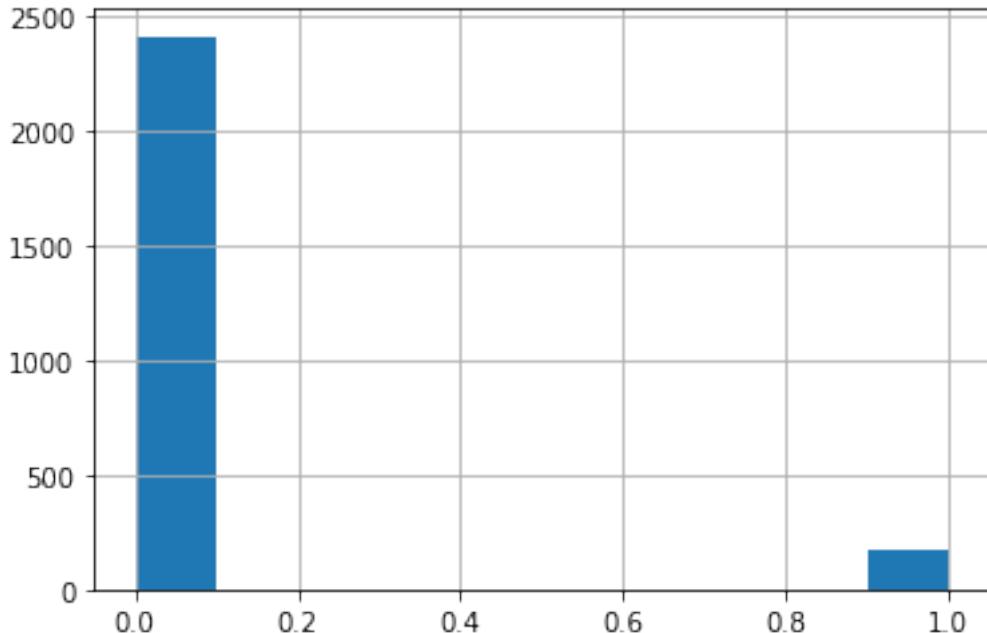


Figure 2.1: An unbalanced set of labels

Around 7% of the data is labelled as 1 and 93% as 0.

So the difficulty here will be to predict correctly label 1 because for example, the algorithm could predict that the label will always be 0 and still have a good accuracy because label 0 happens pretty much all the time.

# Preprocessing the data

The goal of this step is to pre-process the data before using it.

## 3.1 Clear the data

The first step is to clear the data of all byte inputs. This is done by going through each input and convert it to strings or integers for the next step.

## 3.2 One-hot encoding

The goal of this step is to convert categorical data to one-hot encoding labels. This is done because machine learning algorithms need to have integers to run. By the end of this step we have only integers as inputs.

## 3.3 Scaling

By applying this scaling to the dataset, we prevent some features to be more important than others. Since we do not have any infos of how important some features may be regarding others, we normalize everything to make their average equal to 0 with a unit variance.

# Classification of the data

I will show in this part all the algorithms used for a first approach. For each algorithms, a GridSearchCV have been used to find good parameters.

## 4.1 SVC

Let's compute the confusion matrix of this method.

		Prediction	
		0	1
True Label	0	0,74	0,26
	1	0,32	0,68

Results are all right. SVC is good at trying to predict with a low amount of values of label 1.

## 4.2 KNN, NN, Random Forest, XGBoost

All these techniques are not efficient. They all predict that the test inputs will be labelled as 0. So they have good accuracy because 0 label is present in much larger amount in the test dataset but it is hiding a poor prediction of label 1. We can see it with a confusion matrix of this kind :

		Prediction	
		0	1
True Label	0	~ 1	~ 0
	1	~ 1	~ 0

# Rebalancing the data

## 5.1 Up-sampling the label '1'

We use the `resample` function of `sklearn` module. This allows us to have a greater amount of label '1' to feed our machine learning algorithms and correct the major default of this dataset.

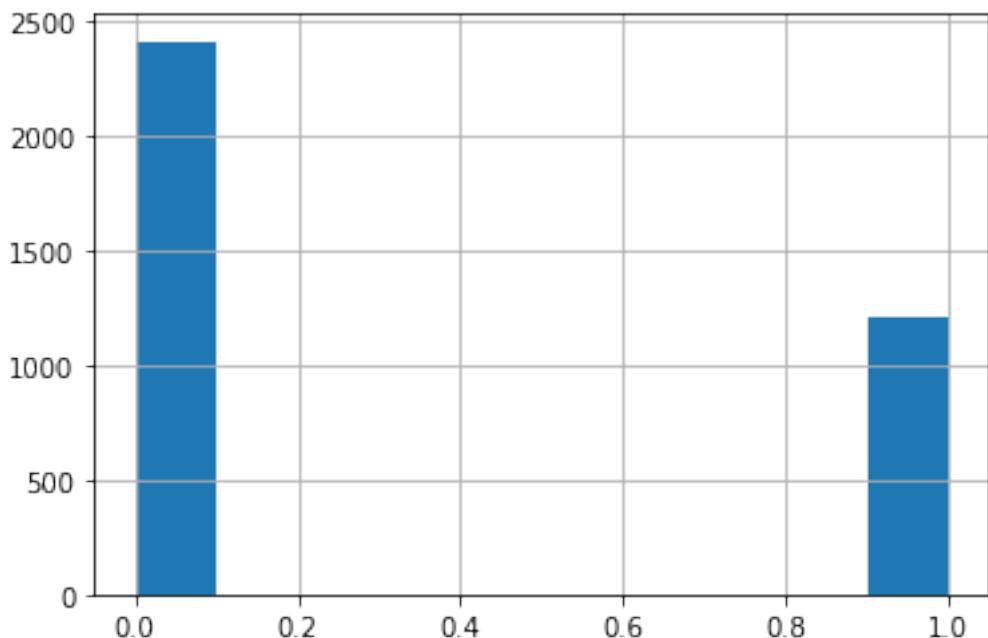


Figure 5.1: A more balanced dataset

## 5.2 Random Forest

We use first a Random Forest model with a `GridSearchCV` which gives a quite good result. We can see an improvement of the method : this is a confirmation that the unbalanced output was the issue there.

Parameters :

- criterion: entropy,
- maximum depth: 4,
- maximum of features: auto,
- number of estimators: 100

*Accuracy : 0,83*

		Prediction	
		0	1
True Label	0	0,84	0,16
	1	0,34	0,66

### 5.3 Neural Networks

The best result is given by the Neural Network. Here are the parameters :

- hidden layer size : 100,
- learning rate : 0.001,
- maximum iterations : 300

Results are really good with an accuracy of 0,91 along with a good prediction on label 1.

		Prediction	
		0	1
True Label	0	0,92	0,08
	1	0,17	0,83

**Important Note :** The test dataset is sampled in the **original** sample and not the up-sampled one. This is important because we want to test our model on real values and not on values that were generated because the generation could include predictive patterns.

# Conclusion

An unbalanced dataset is very hard to process. Indeed, with a metric like the accuracy, we are easily fooled because a great accuracy does not mean great results. The model can predict only label 0 and this will give great accuracy. This is why a tool like the confusion matrix is great to make sure that label 1 are correctly predicted as well and we achieve quite good results with the Neural Network by up-sampling label 1.

## Part III

# Multi-class Classification - Handwritten numerals

# Informations on the dataset

This dataset consists of features of handwritten numerals ('0'-'9') extracted from a collection of Dutch utility maps. 200 patterns per class (for a total of 2,000 patterns) have been digitized in binary images. These digits are represented in terms of the following six feature sets (files):

1. mfeat-fou: 76 Fourier coefficients of the character shapes;
2. mfeat-fac: 216 profile correlations;
3. mfeat-kar: 64 Karhunen-Loëve coefficients;
4. mfeat-pix: 240 pixel averages in 2 x 3 windows;
5. mfeat-zer: 47 Zernike moments;
6. mfeat-mor: 6 morphological features.

# Observation of the dataset

I will use the dataset mfeat-pix to classify the data as this is the one I understand the most which is important for tuning parameters. I tested the process on other files and this is exactly the same process and it gives very good result on the other files.

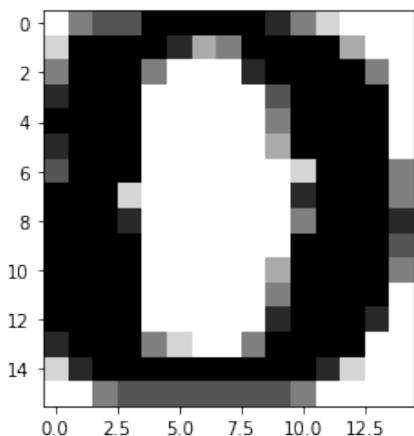


Figure 8.1: Example of label 0

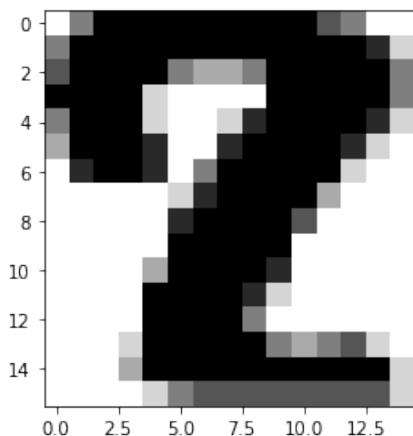


Figure 8.2: Example of label 2

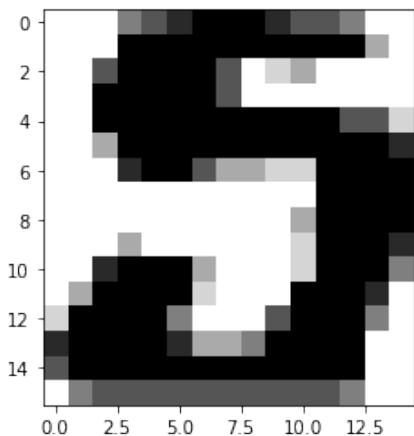


Figure 8.3: Example of label 5

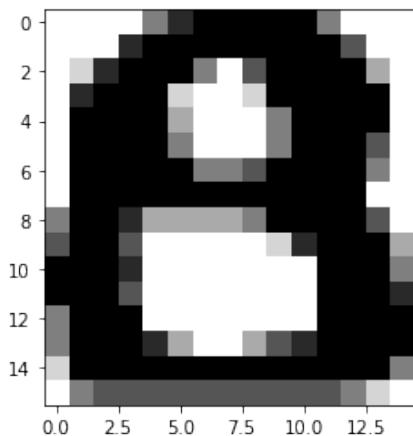


Figure 8.4: Interpolation for Data 8

# Preprocessing the data

They are 10 labels from '0' to '9' which are ordered this way : 200\*'0' then 200\*'1' until 200\*'9'.

Then I shuffle the data to make it more realistic because ordered labels can possibly give poor results with machine learning algorithms.

Then the dataset is separated and then scaled like in the first part.

We then use the function `to_categorical` to make sure that the true class is represented as a one-hot encoded vector, and the closer the model's outputs are to that vector, the lower the loss so it is quite useful to use this function to evaluate this crossentropy to make sure that, for example, the label "9" has the same weight as the label "1".

# Classification of the data

## 10.1 Size of the hidden layer

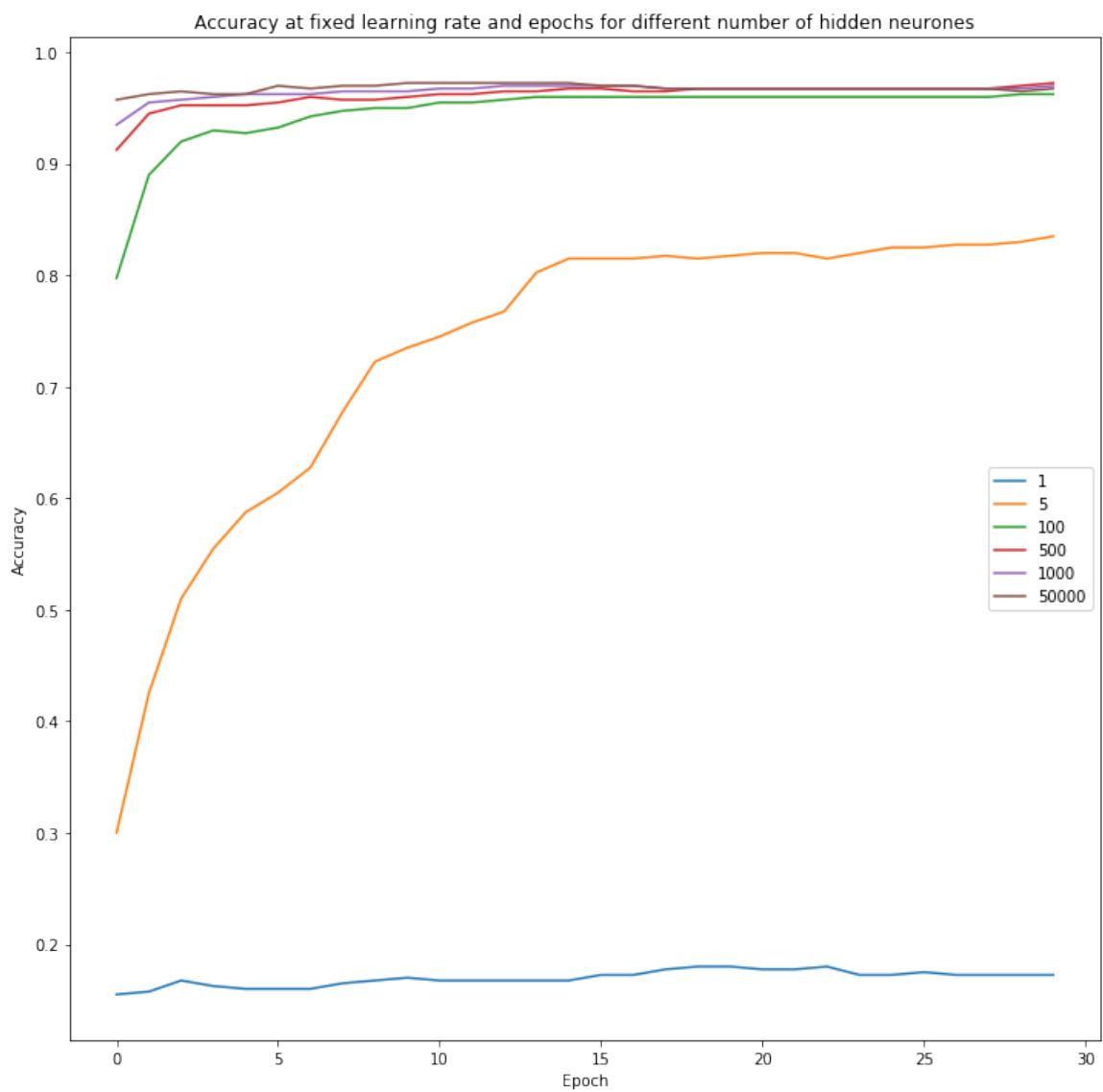


Figure 10.1: Optimization process accuracy for various hidden layer sizes

500 is chosen because it gives already great results close to the highest possible while still being quite low.

## 10.2 Activation function

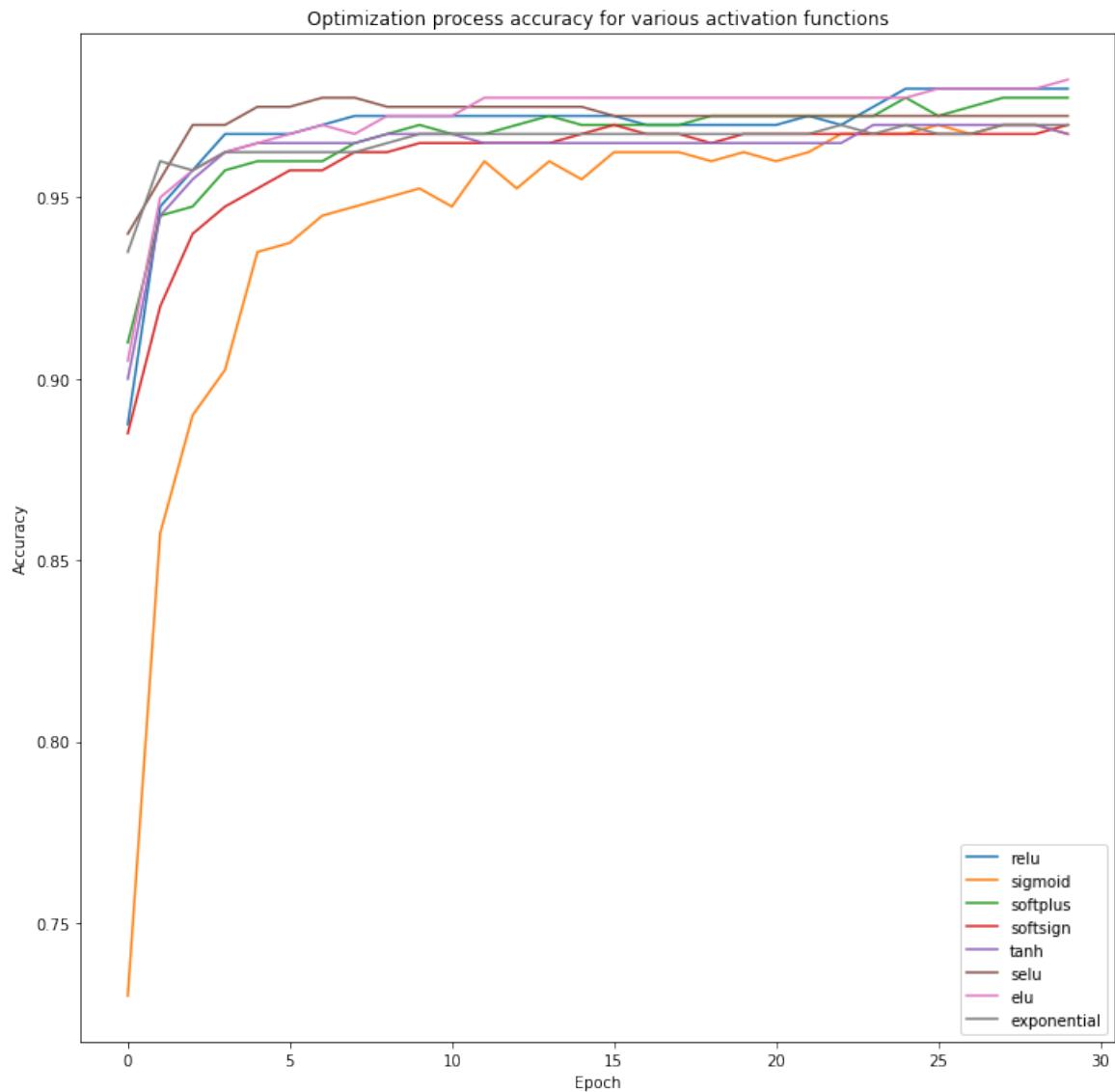


Figure 10.2: Optimization process accuracy for various activation functions

`relu` and `selu` gives great results and are therefore chosen.

### 10.3 Learning rate tuning

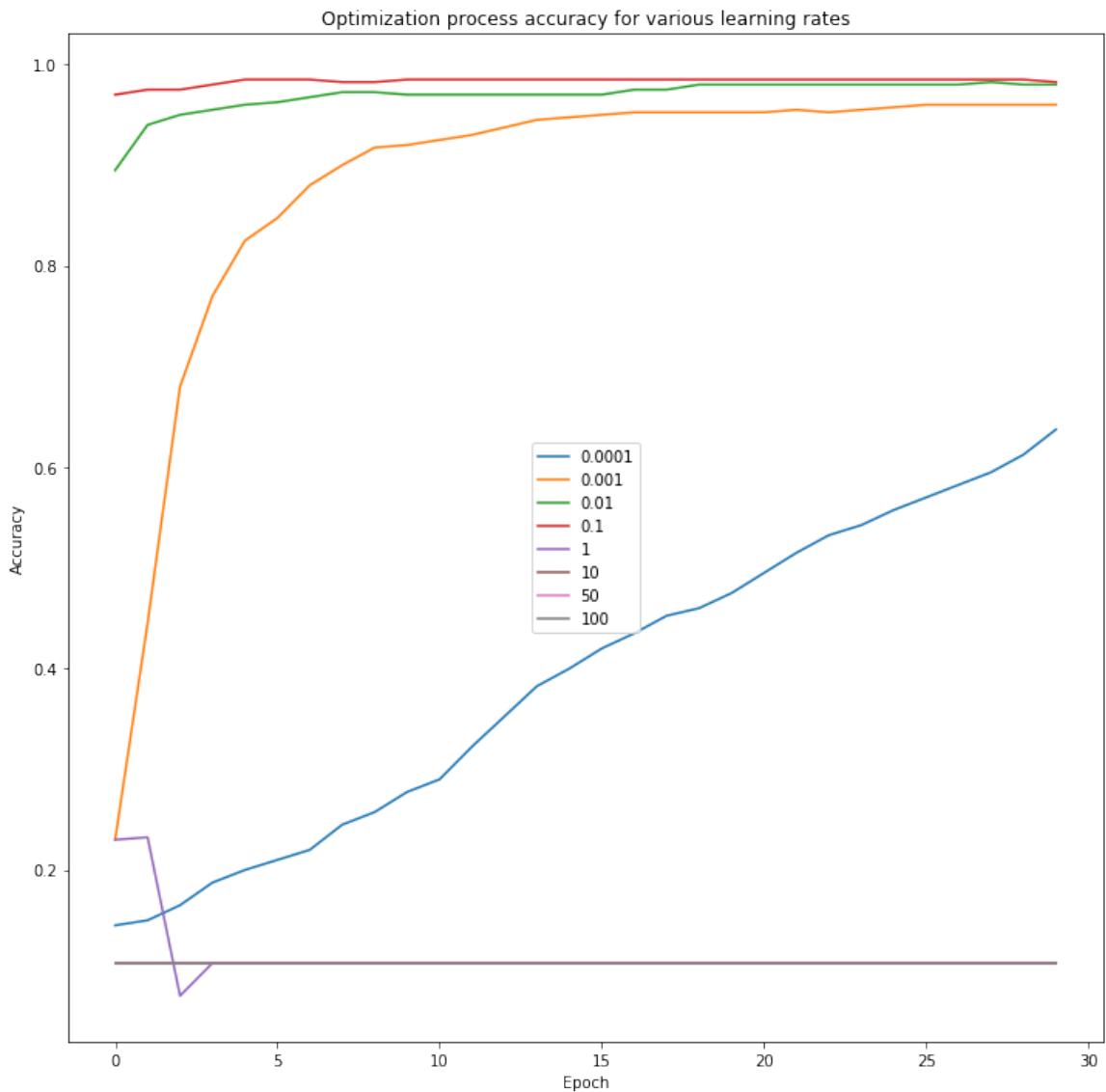


Figure 10.3: Optimization process accuracy for various learning rates

A learning rate of 0.1 is chosen for its highest performance and low oscillation.

## 10.4 Optimizer tuning

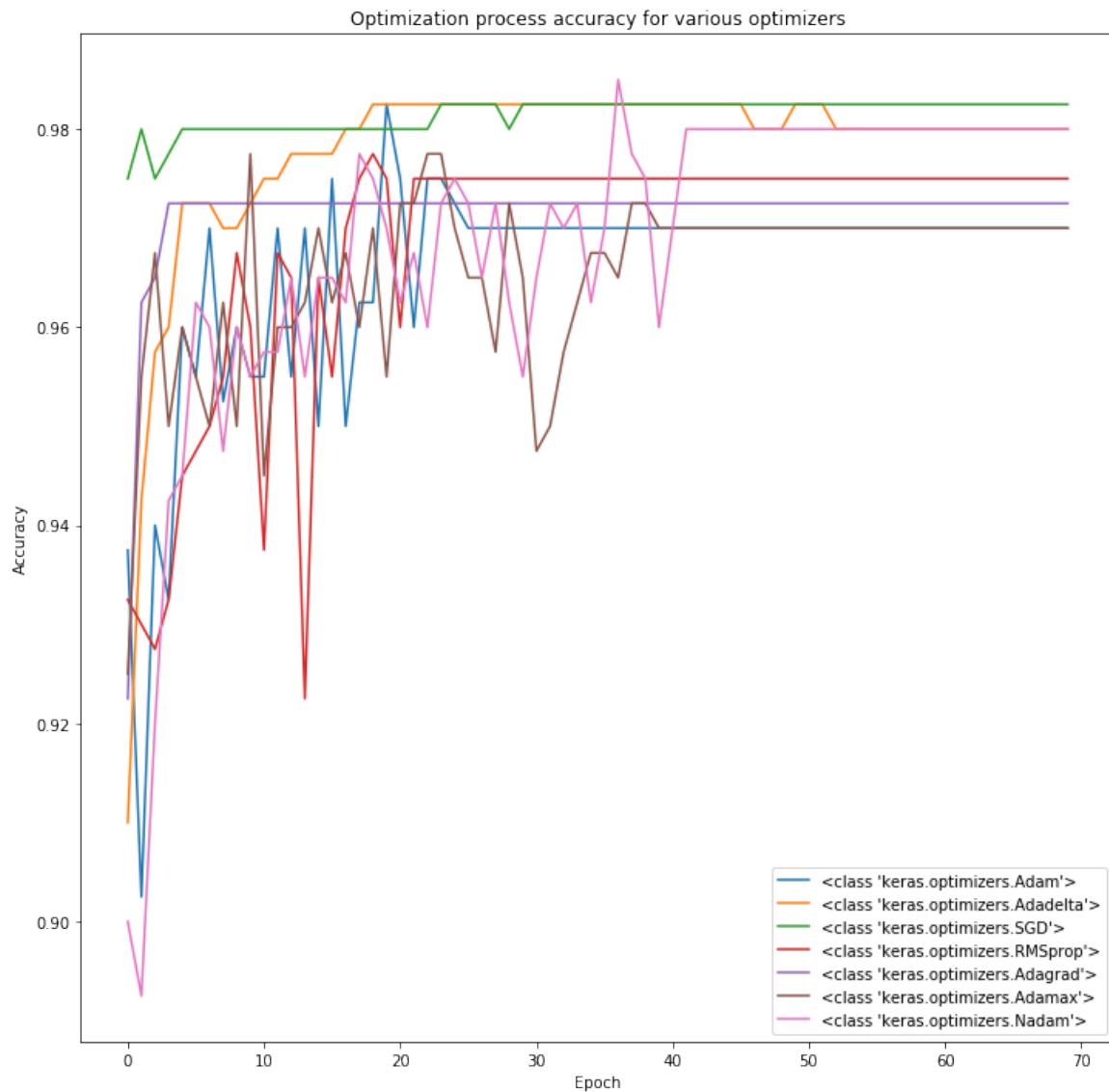


Figure 10.4: Optimization process accuracy for various optimizers

SGD optimizers from Keras module will be chosen for its highest performance and the fact that it is less subject to oscillation than other optimizers.

## 10.5 Batch size tuning

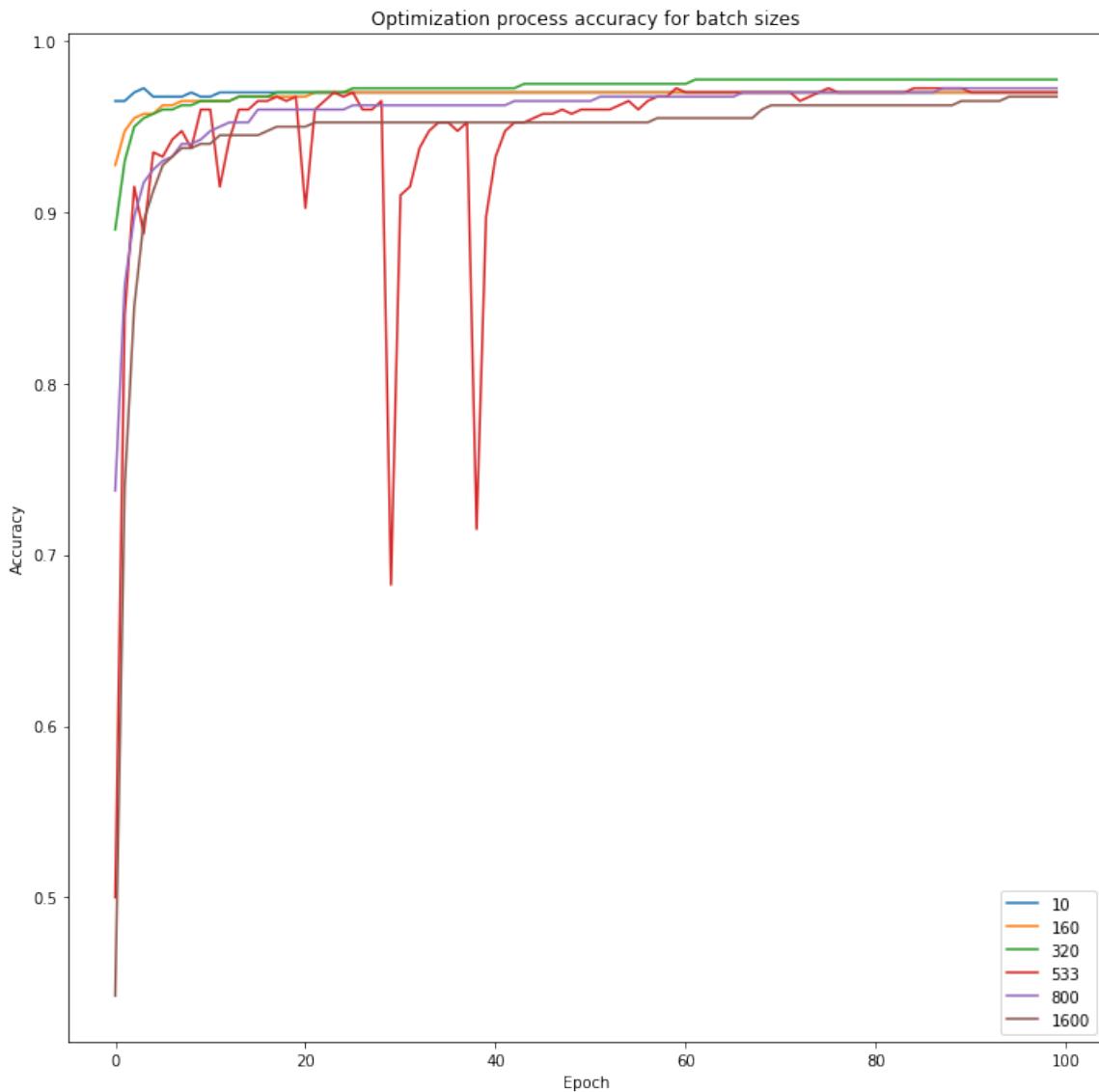


Figure 10.5: Optimization process accuracy for various batch sizes

We will chose 320 for batch size for an overall good performance while not being to high and avoiding oscillations.

# Results

Lets resume all the parameters and conclude on the efficiency of this method.

Parameters:

- number of epochs : 500
- size of the hidden layer : 500
- activation function : relu
- learning rate : 0.1
- optimizer : SGD
- batch size : 320
- metric : categorical accuracy
- loss : categorical crossentropy

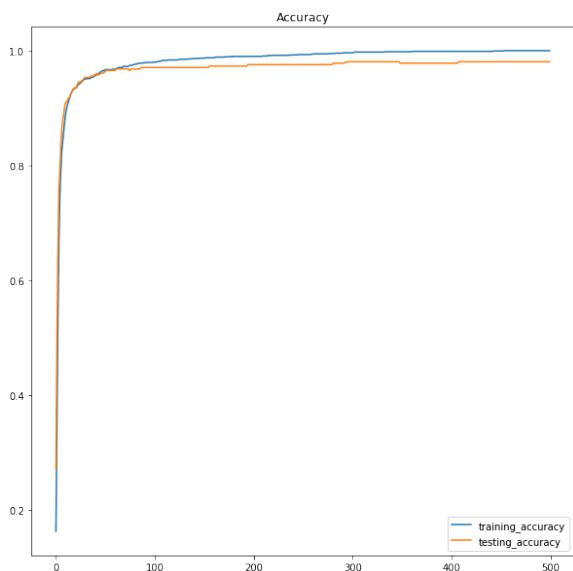


Figure 11.1: Training accuracy vs. testing accuracy

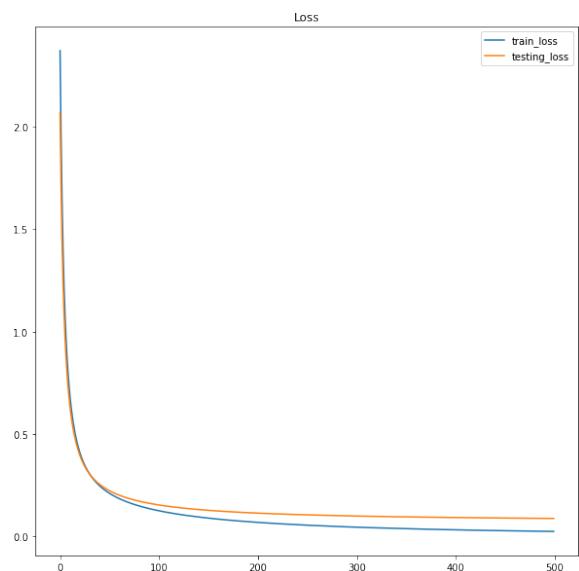


Figure 11.2: Train loss vs. testing loss

The model is quite promising. Let's analyse the confusion matrix to make sure our model is correct.

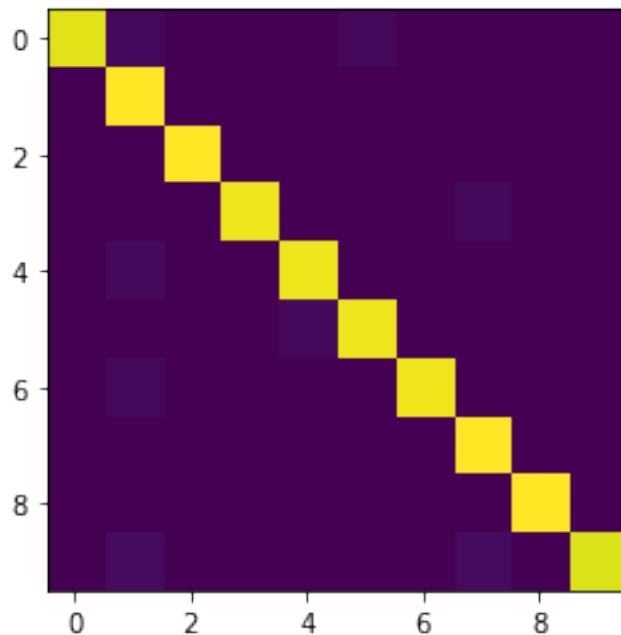


Figure 11.3: Confusion Matrix

*Accuracy : 0,98  
F1 macro score : 0,97*

The accuracy is very good and the labels are very well predicted.

## Part IV

# Regression : Communities and Crime

# Informations on the dataset

Communities within the United States. The data combines socio-economic data from the 1990 US Census, law enforcement data from the 1990 US LEMAS survey, and crime data from the 1995 FBI UCR.

Attribute Information (122 predictive, 5 non-predictive, 1 goal):

- state: US state (by number) - not counted as predictive above, but if considered, should be considered nominal (nominal)
  - county: numeric code for county - not predictive, and many missing values (numeric)
  - community: numeric code for community - not predictive and many missing values (numeric)
  - communityname: community name - not predictive - for information only (string)
  - fold: fold number for non-random 10 fold cross validation, potentially useful for debugging, paired tests - not predictive (numeric)
  - population: population for community: (numeric - decimal)
  - householdsize: mean people per household (numeric - decimal)
  - racepctblack: percentage of population that is african american (numeric - decimal)
  - racePctWhite: percentage of population that is caucasian (numeric - decimal)
- .....
- **ViolentCrimesPerPop:** total number of violent crimes per 100K population (numeric - decimal) **GOAL** attribute (to be predicted)

# Observation of the dataset

Here are few inputs of the dataset.

	state	county	community	communityname	ViolentCrimesPerPop
0	8	?	?	Lakewoodcity	0.20
1	53	?	?	Tukwilacity	0.67
2	24	?	?	Aberdeentown	0.43
3	34	5	81440	Willingborotownship	0.12
4	42	95	6096	Bethlehemtownship	0.03

We can see that there are missing values which is going to be an issue for machine learning algorithms.  
Here is a distribution of the labels :

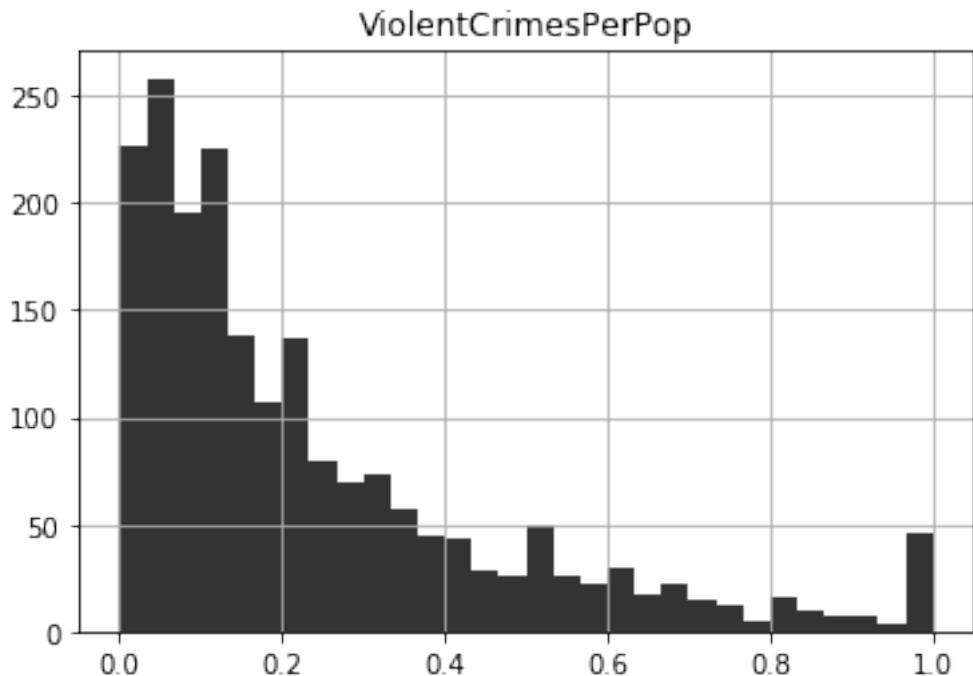


Figure 13.1: Distribution of ViolentCrimesPerPop label

# Preprocessing the data and regressors used

From 122 predictive features, 25 contains missing values. So we can drop the features that have missing values.

Next, we have to deal with the missing values issue.

The scaler and the algorithms can only deal with numeric values. Furthermore, there is a set of non-predictive features that need to be eliminated. These are : **state**, **county**, **community**, **communityname**, **fold**. Some of those have already been eliminated because they contained NaN values (= missing values).

The regressors that were used for this dataset are :

- DecisionTreeRegressor
- RandomForestRegressor
- LinearRegression
- KNeighborsRegressor
- SVR

# Results

For each regressor used, the Mean Absolute Error, the Mean Squared Error, and the Root Mean Squared Error are showed.

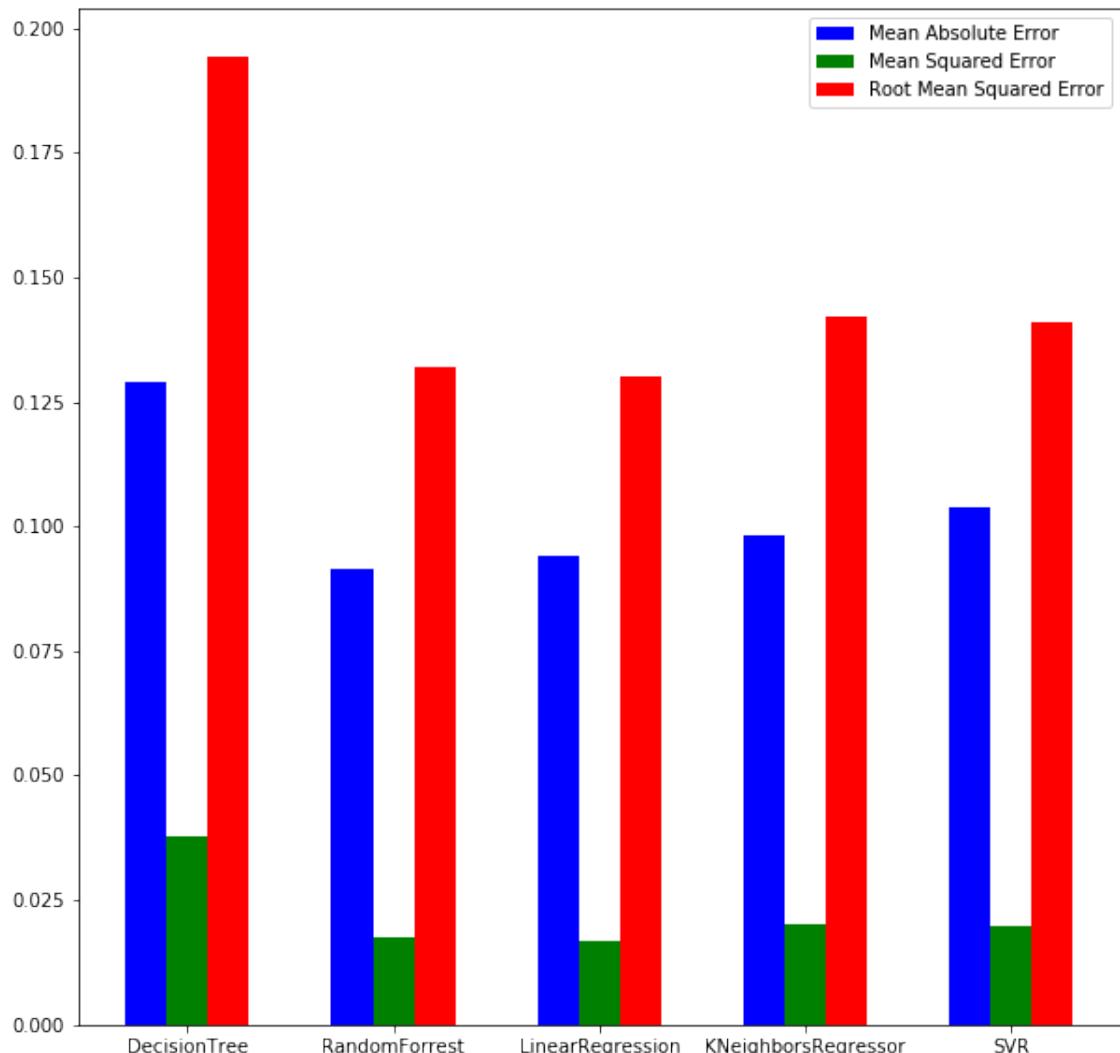


Figure 15.1: Errors of different regressors

We can see that the Random Forest Regressor is the most efficient model out of the 5. We can evaluate finally its efficiency with a KFold cross validation.

The results are good and prove than RandomForest Regressor is the most efficient on this dataset.

- Cross Validation Score mean is 0.644
- Cross Validation standard deviation is 0.038