Machine Learning
-
Supervised Learning

Brandon Alves

September 16, 2022

Contents

List of Figures

1 Introduction

In this article, I will present the results of various Supervised Learning methods applied on two classification problems. Those methods are:

- Decision Trees with some form of pruning;
- Neural Networks;
- Boosting;
- Support Vector Machines;
- K-Nearest Neighbors.

For each of them, I will discuss the results obtained, the pros and cons of using them, and the parameters that were used. I will also discuss the results when applying the methods on the two different datasets. At the end I will also compare the methods between them and highlight the best one for each dataset.

I will present many plots on this report to show the evolution of the accuracy depending on some parameters. Those plots will be presented in the following way : the hard line represents the median precision value for a given parameter value while the tint area goes from the minimum to the maximum precision value for the given parameter value.

The experiments on this report were performed on a computer with the following specifications:

- CPU : Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz, 6 cores, 12 threads
- RAM : 16GB DDR4
- GPU : NVIDIA Quadro P620

2 Datasets

In this article, I will learn over two datasets:

- Credit Card Fraud;
- Starcraft II.

Credit Card Fraud

This dataset is a credit card fraud detection dataset. The dataset contains transactions made by credit cards in September 2013 by European cardholders. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions. It contains only numerical input variables which are the result of a PCA transformation. The only features which have not been transformed with PCA are *Time* and *Amount*. Feature *Time* contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature *Amount* is the transaction amount. Feature *Class* is the response variable and it takes value 1 in case of fraud and 0 otherwise.

The training dataset contains between 10 000 and 210 000 transactions to evaluate how the training size can impact the results of the leaning process. The test dataset contains 236 transactions from the remaining samples that are not in the testing set. Half of the test dataset is composed of frauds and half of it is composed of non-fraud transactions. I have chosen that ratio in order to compare the ressults of the two datasets.

Starcraft II

This dataset is composed of recorded competition games from the Starcraft II game. For each games, the dataset provides the player name and the player actions at a given time for a set of 30 different actions. The purpose is to find the player names having only the actions recorded.

To allow all the medels to use that dataset, a number of feature is precomputed on it, such as the first time of each action or the average action per minute rate. This sum up into 72 different features for 200 class splits on 1950 games on the training set and 884 on the testing one. The training and testing set are constructed so that the same proportion are applied to each class to ensure that every class are represented on both datasets. The train dataset is then randomly flushed to allow to crop it for training and still have the possibility to have every class.

The difficulty of this dataset is in hight number of different classes and the few numbers of example. There is 24 classes of the data set that have only 1 example to train from with an average of less than 10 per class. In addition, the dataset is noisy, some players appear under more than one pseudo and so some classes are very similar.

Scaling

Some methods require the data to be scaled between 0 and 1. Both the training and the testing datasets are scaled between 0 and 1. The scaling is done by dividing each value by the maximum value of the dataset. The scaling is done on the training dataset and then applied on the testing dataset.

3 Decision Trees with some form of pruning

In that section I will present the results obtained when using Decision Trees with some form of pruning. We will here discuss the influence of different parameters :

- The maximum depth of the tree;
- The minimum number of samples required to split an internal node;
- The size of the training set.

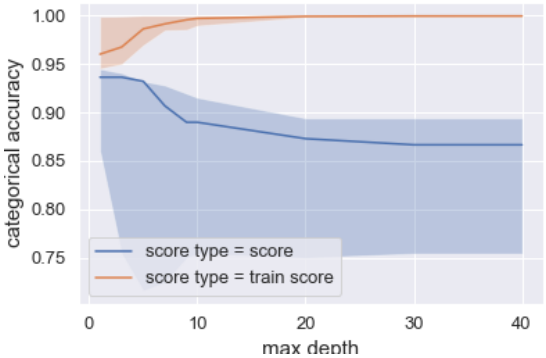
We begin by finding the limit of our classifier. That limit can be of two natures : overfiting or a flat level on both training and test accuracy. Figure 1 represents those limits.

On figure 1a, we can see that the Decision Tree method quickly overfits. Indeed the training accuracy quickly rise above 99% while the test accuracy decreases to reach around 0.87%. We can notice that the best score is obtained for a maximum depth value of 1. Also the score considerably decreases for a maximum depth value of bigger than 3. therefore we can conclude that only 1 out of the 28 dimensions of the input space is relevant for the classification output.

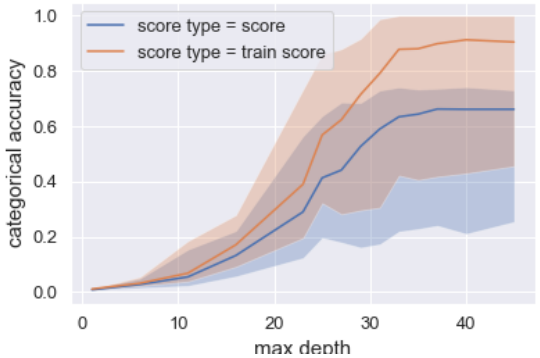
On figure 1b, we can notice that the classifier does not overfit. Although the training accuracy sometimes succeeds to reach 100%, the test accuracy stays on a flat level around 0.75%, meaning that increasing the maximum depth further does not impact the score of the classifier.

Let's discuss now the influence of the size of the training set on the accuracy of the classifier. Figure 2 represents the accuracy of the classifier according to the size of the training set for the credit-card dataset. As expected the accuracy of the classifier decreases when the size of the training set decreases. The accuracy of the classifier is also impacted by the maximum depth of the tree. Indeed, the score for all the different sizes of the training set is almost the same when the maximum depth is around 1, whereas the score differs a lot when the maximum depth of the tree increases. We can also notice that the training set with 160 000 samples gives better results than the trainingset with 210 000 samples. This is probably due to the fact that the training set with 160 000 samples is more balanced than the training set with 210 000 samples, meaning that when adding those 50 000 samples, we probably introduce some noise in the training set.

On the Starcraft sataset, we can see on figure 2b that the accuracy of the classifier is also impacted by the size of the training set. Once again on that dataset there is no overfitting in the learning process. We can notice that the evolution of the gap between each step seems to evolve exponentially. But that comportment is mostly due to the classes not represented on the training set when dealing with few randomly chosen data for training. The standard deviation of accuracy is also reduced by the augmentation of the training set size. Indeed, the more data

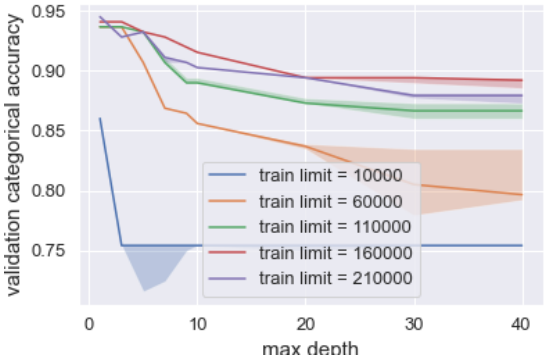


(a) Accuracy for train and test credit-card datasets

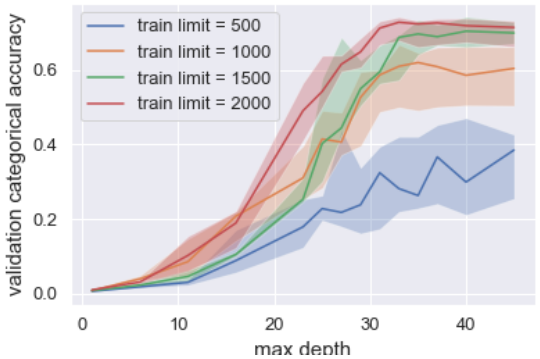


(b) Accuracy for train and test Starcraft datasets

Figure 1: Evolution of the decision tree classifier accuracy according to the maximum depth of the tree

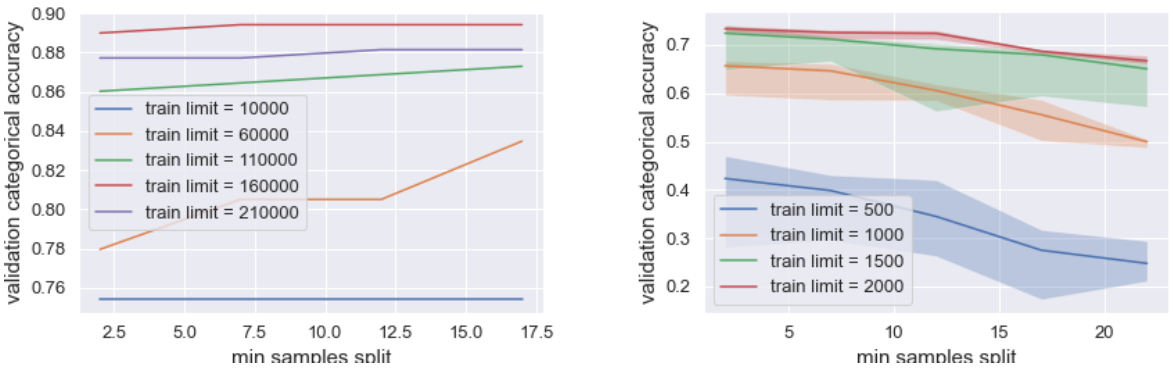


(a) Accuracy for train and test credit-card datasets



(b) Accuracy for train and test Starcraft datasets

Figure 2: Evolution of the decision tree classifier accuracy according to the size of the training set



(a) Accuracy for train and test credit-card datasets

(b) Accuracy for train and test Starcraft datasets

Figure 3: Evolution of the decision tree classifier accuracy according to minimum number of samples required to split an internal node

we consider, the less important fine-tuning the meta parameters is : the classifier will learn the best way to use that parameters and will so end up with similar results than another classifier with slightly different parameters.

The last parameter I will discuss is the minimum number of samples required to split an internal node. That variation is displayed on figure 3. In the case of the credit-card dataset, we can see that the accuracy of the classifier is only impacted by the minimum number of samples required to split an internal node when overfiting. In Starcraft dataset, we see that this paramet has a little influence even if smaller than the over parameters. In that particular case, this sort of pruning only led to decrease the accuracy of the classifier on the test set. This is probably caused by sometimes the very few data available per class, and at the same time do not prevent overfitting as the only overfitting possible here is due to the few data available.

Decision Tree is the faster method compared with all the other ones. To get those results and generate the data associated with, it took 28 seconds. The full run coun 208 leanr and test cycles, which gives an average of 279 milliseconds per cycle. That is two time faster than KNN.

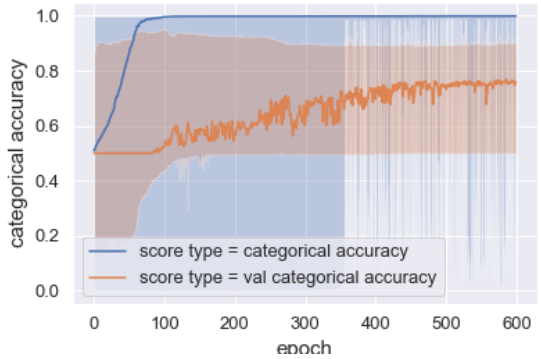
The decision tree classifier performs pretty well as seen above. But that method does not perform very well on highly dimensional datasets. To resolve that problem, Random Forrest can be used. Random Forrest is a method that uses multiple decision trees to classify the data. The method is based on the idea that a large number of relatively uncorrelated classifiers (trees) operating as a committee will outperform any of the individual classifiers. The method is also known as bagging.

4 Neural Networks

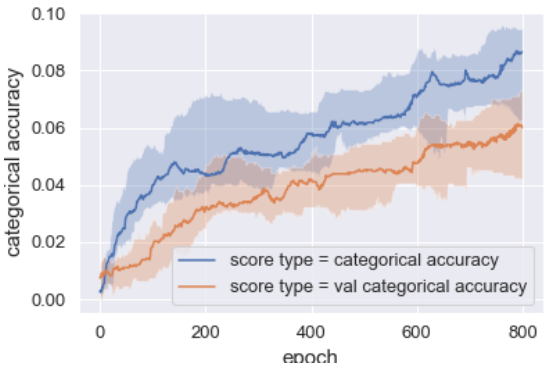
Neural Network model is the most complicated in terms of parameters in comparison with the other models. We will discuss the influence of the followings:

- The number of epochs;
- The use of batch normalization;
- The number of layers;
- The activation function;
- The size of the training set.

The first thing to say is that our Neural Network classifier does not perform as well as espected with all the parameters explored. For example in the second dataset, in figure 4b, our classifier difficulty scores above 0.5. As expected the accuracy goes up with the number of epochs.



(a) Accuracy for the train and test dataset on credit-card



(b) Accuracy for the train and test dataset on Starcraft

Figure 4: Evolution of the neural network classifier accuracy according to thenumber of epochs

The parameter that affects the most the accuracy of the calssifier is the use of a batch normalization as we can see on figure 5a and 5b. We know that the use of a batch normalization permits to decrease the number of training epochs required to train a neural network. We can see on figures 5 it decreases drastically the number of epochs.

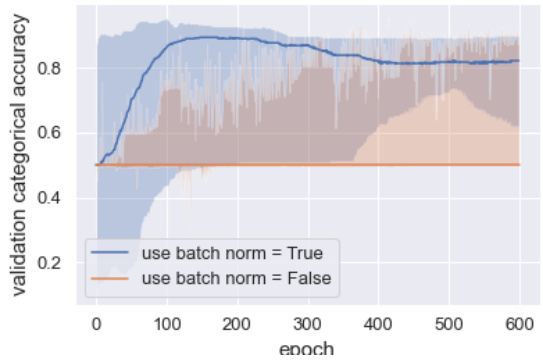
The impact of the activation function is not as important as the batch normalization. We can see on figure 6b and 6e that the accuracy is not very different between the different activation functions expected for the sigmoid function on figure 6b.

Another parameter that affected the score of the classifier was the number of layers. We can see on figures 6a and 6d that for both datasets, the accuracy depends on the layers. One interesting thing to notice is that for Starcraft dataset, the best fit is obtained for no intermediate layers (output neurones directly connected to input) while for credit-card dataset, that configuration gives very poor results.

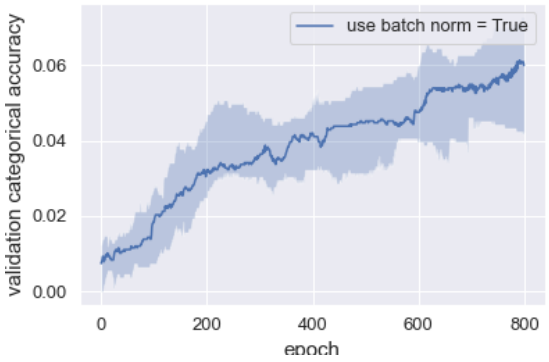
One interesting thing in that case is that the training size does not seem to have a big impact on the classification score. We can see on figure 6c and 6f that the accuracy is not very different between the different training sizes. For example we have pretty much the same accuracy for a 1000 and a 160000 training size.

In terms of computation time, our Neural Network classifier is the slower one. In addition, the memory used in the learning process was also very high. It took 2h30m to train the neural network on both datasets.

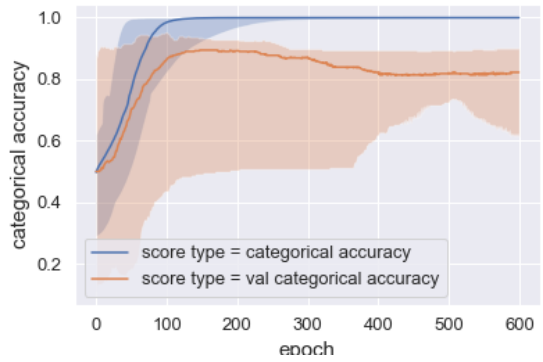
I saw that it was not easy to find a good configuration for the neural network for all these parameters to get good results.



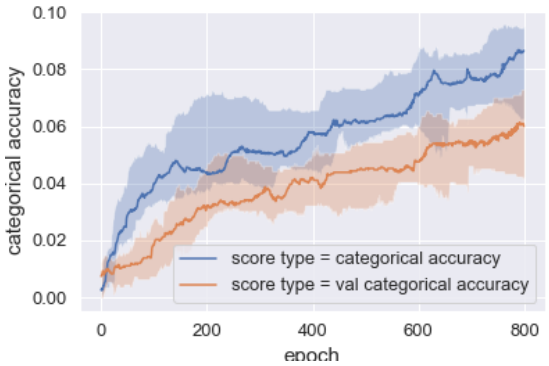
(a) Accuracy with and without batch normalization for credit-card



(b) Accuracy with and without batch normalization for Starcraft

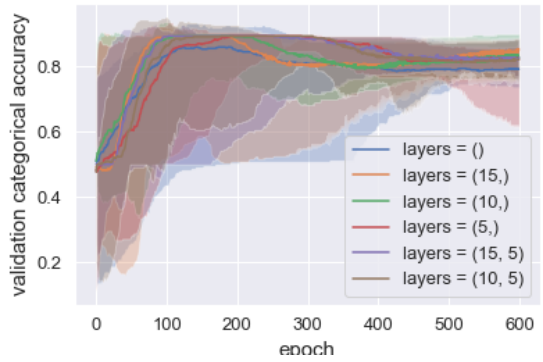


(c) Accuracy for the train and test dataset on credit-card, with batch normalization

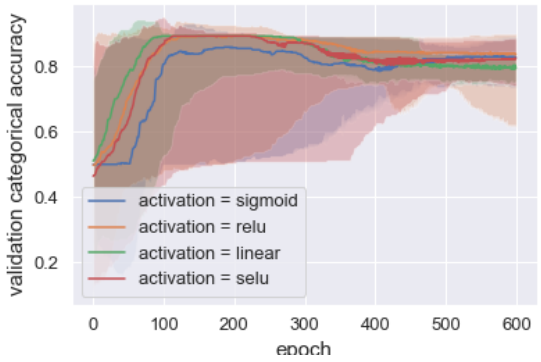


(d) Accuracy for the train and test dataset on Starcraft, with batch normalization

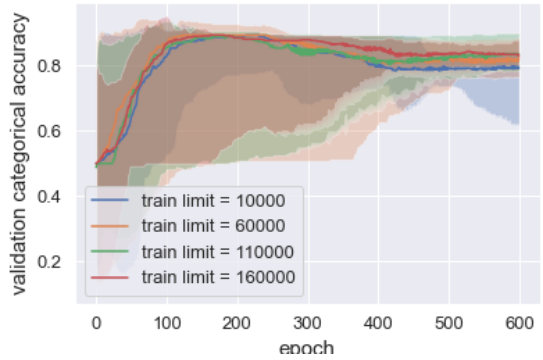
Figure 5: Evolution of the neural network classifier accuracy according to the use of batch normalization



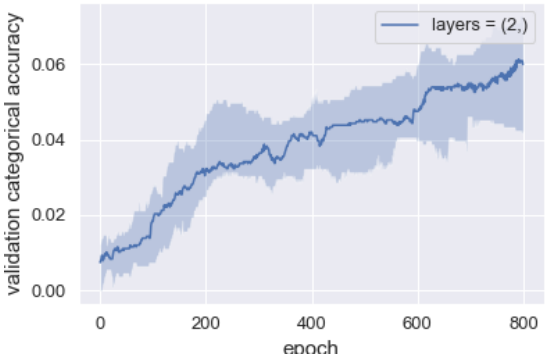
(a) Accuracy for different layers on credit-card



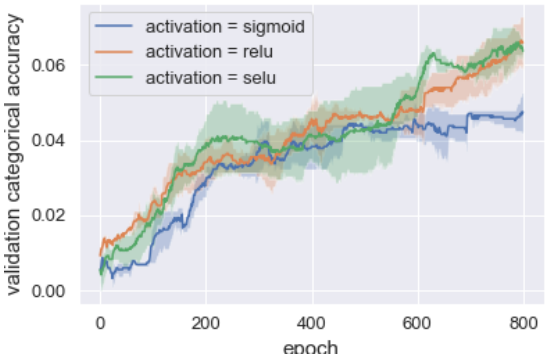
(b) Accuracy for different activations on credit-card



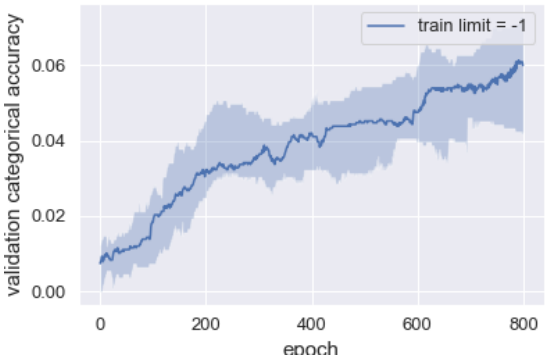
(c) Accuracy for different training dataset size on credit-card



(d) Accuracy for different layers on Starcraft



(e) Accuracy for different activations on Starcraft



(f) Accuracy for different training dataset size on Starcraft

Figure 6: Evolution of the neural network classifier accuracy for different parameters

5 Boosting

Let's now discuss the influence of the followings parameters on the accuracy of the Boosting classifier:

- The number of boosting stages to perform;
- The maximum depth of an individual regression estimator;
- The minimum number of sample needed to split a node of an estimator;
- The size of the training set.

6 Support Vector Machines

In this section we will discuss about the Support Vector Machines classifier. We will see the influence of the followings parameters:

- The penalty of the error term;
- The kernel function;
- The training set size.

7 K-Nearest Neighbors

Here I will discuss the results obtained using a K-Nearest Neighbors classifier. We will see the influence on the accuracy of the classifier of :

- The number of neighbors;
- The distance metric (p-norm);
- The size of the training set.

By design, the accuracy on the training set is always 100% for KNN, so I will not display it on the plots.

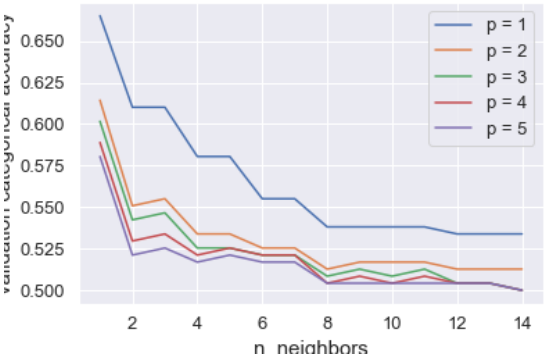
On figure 7, I present the effect of those parameters on the classification score. First, we can notice that the norm used for the distance metric has a consequent influence on the classification accuracy. The accuracy is better when using the Manhattan norm ($p = 1$). This is not a surprise as higher norm are generally inadequate to measure distances on highly dimensional spaces.

On figures 7a and 7b, we can see that the accuracy of the classifier is better for a small number of neighbors. This is not a surprise as both datasets have very few examples in each classes. KNN is a non-parametric classifier. It is not able to generalize the data, so it is better to use a small number of neighbors. We usually increase the research perimeter to be more resilient to noise but here the data do not allow us such a liberty. This is particularly true in the credit card dataset where the fraud are really close to legal transactions.

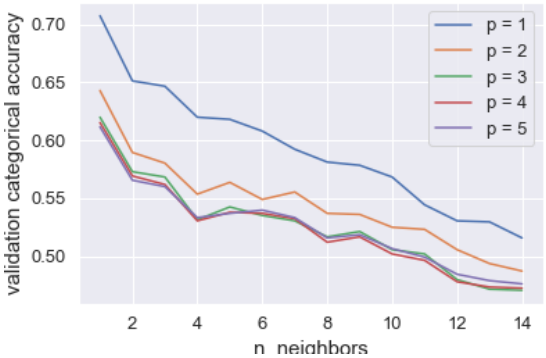
On figures 7c and 7d, we can see that the accuracy of the classifier is better for an important number of training examples. Also the learning process don't overfit. However, on figure 7c, for the credit-card datatset, we notice that there is a flat zone where the accuracy is pretty much constant. This is due to the fact that the number of frauds is very low compared to the number of legal transactions. I imagine that the samples added to the training set during that flat are mostly similar to previous ones. This is why the accuracy is not really increasing.

Concerning time efficiency, the KNN classifier achieve a correct performance. On the Starcraft dataset, it takes 2m31s to run 280 times the learn and test process with the parameters explored above. KNN uses multiple thread to perform the classification task.

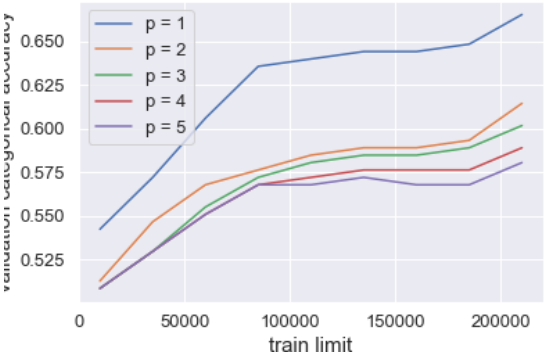
In comparison with the other methods, KNN does not perform very well. Having a discussion with an expert of the dataset would be a godd idea for designing a distance metric that is better suited to the problem. Indeed, the distance metric is the key to the success of KNN. So it is important to choose the most appropriate one.



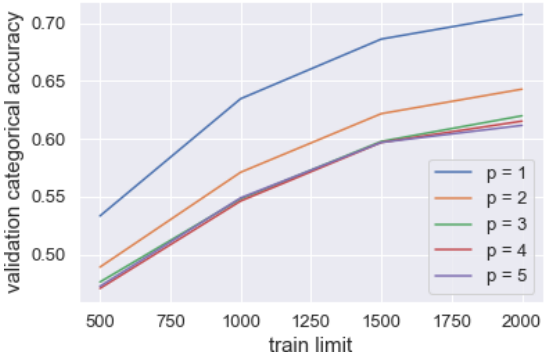
(a) Evolution of accuracy according to the number of neighbors used for the classification on the credit-card fraud dataset, with $train\_limit = 90000$



(b) Evolution of accuracy according to the number of neighbors used for the classification on the Starcraft dataset, with $train\_limit = 2000$



(c) Evolution of accuracy according to the number of training examples on the credit-card fraud dataset, with $n\_neighbors = 1$



(d) Evolution of accuracy according to the number of training examples on the Starcraft dataset, with $n\_neighbors = 1$

Figure 7: Evolution of the KNN classifier on the test set with different parameters

8 Conclusion