Aprendizagem Automática

Assignment 1 - Classifiers

Andrea Mascaretti N52222 Daniel Pimenta N45404

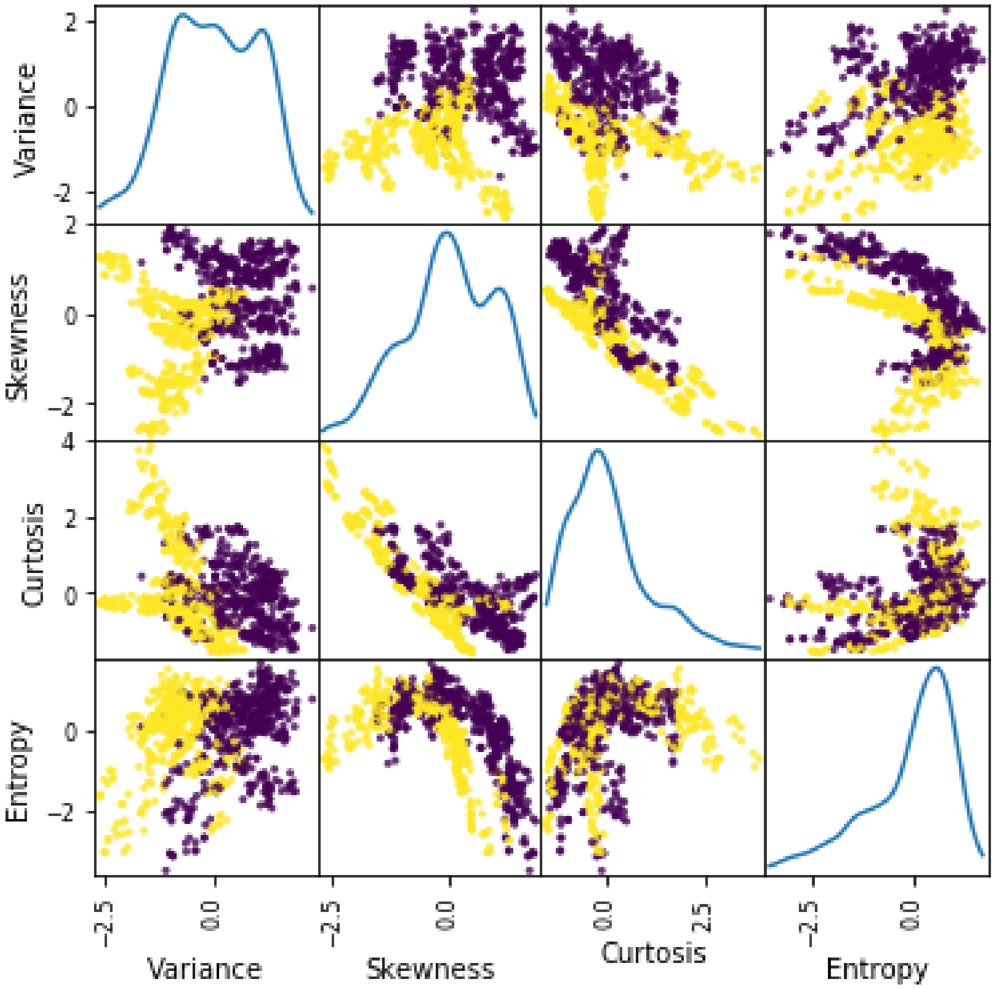
October 21, 2017

Abstract

The main goal of this assignment was the parameterization, fitting and comparation of Logistic Regression, K-nearest Neighbours and Naive Bayes classifiers.
The data set used was the banknote authentication which was obtained from the UCI machine learning repository.
To achieve our goals we used the Spyder IDE with programming language Python 3.6 and some of its modules such as *Panda* to load and manage our data, *NumPy* which is the fundamental package for scientific computing with Python and was used for various mathematical calculations and managing multidimensional arrays, *sklearn (scikit-learn)* which has efficient tools for data mining and data analysis used in our assignment to preprocess data (shuffle, standardize, split), fold our data into k stratified folds, calculate cross validation score and fit our data on Logistic Regression and K-nearest Neighbours classifiers. Another imported module was Matplotlib.Pyplot which provides a MATLAB-like plotting framework used to build the error plots.

1 Introduction

Using the Spyder IDE and programming language Python 3.6 and with a csv file which held data about bank notes authenticity we were asked to parameterize, fit and compare three different classifiers: Logistic Regression, K-nearest Neighbours and Naive Bayes. For the first two classifiers, we were allowed to import them from a module which was provided by scikit-learn. The Naive Bayes classifier was fully implemented for our assignment to fit our data using the best bandwidth we could find and to then later predict the real values for a test set.
The initial data was split into a training set and a test set. The training set was then folded into 5 folds to be used by Cross Validation. For the Logistic Regression classifier we started with a C value of 1 and incremented to its double for 20 iterations. For the K value of the K-Nearest Neighbours classifier, we tested k values from 1 to 39 using odd values only. In the end, we hope to compare our three classifiers performance using McNemar's test with 95% confidence interval by comparing their predictions with the real values and pick the best classifier approached in this assignment.
Before proceeding with the analysis, we standardised our data. To do so, we centred with respect to the mean and normalised taking into account the standard deviation. It is important to notice that we considered the mean and the standard deviation computed from the training set. Those same transformations were applied afterwards to the test set in order to obtain an estimate of the accuracy of our classifiers. To get an idea of the distribution of our data, we also decided to visualise them by means of a series of scatterplots and density graphs.



2 Classifiers

2.1 Logistic Regression

The logistic regression is a generalised linear model. Before considering a more classic machine learning framework, we provide a brief introduction to the main idea behind the model.

The logistic regression is composed, as every linear model, of three part:

- A sequence of *response variables* $Y_1, \ldots, Y_n$. Such response variables are called random components. The main assumption we shall make regarding those variables is that they are all independent random variables. Moreover, each one of them will have a distribution from the exponential family. Bear in mind that we are not imposing that the various Y_is are identically distributed.

- The *systematic component* is our model. It is a function of some predictors (also known as regressors or covariates), linear in the parameter and related to the mean of Y_i.

- The *link function* $g(\mu_i)$ will allow us to link the two components, allowing us to say that

$$g(\mu_i) = \beta_0 + \sum_{i=1}^{r} \beta_i x_i \tag{1}$$

where $\mu_i = \mathbb{E}[Y_i]$. In the machine learning framework, it is usually more common to consider the inverse of the link function.

In our model we will assume the following:

$$Y_i \overset{ind}{\sim} \mathsf{Bernoulli}\left(\pi_i\right) \tag{2}$$

and therefore we will have that

$$\mathbb{E}\left[Y_i\right] = \pi_i. \tag{3}$$

Now, in this model we will assume a relationship between the logarithm of the odds of success for Y_i (the log odds or *logit*) and the predictor $\mathbf{x} = (1,\, x_1,\, \dots,\, x_r)$. Mathematically, this entails

$$\ln\left(\frac{\pi}{1-\pi}\right) = \beta'\mathbf{x} \tag{4}$$

and $\beta = (\beta_0,\, \dots,\, \beta_r)$.

Rewriting in order to obtain the exponential family form we have

$$\pi^y\left(1-\pi\right)^{1-y} = \left(1-\pi\right)\exp\left\{y\,\ln\left(\frac{\pi}{1-\pi}\right)\right\}, \tag{5}$$

where the term $\ln\left(\frac{\pi}{1-\pi}\right)$ is the natural parameter of the exponential family and shows why we will implement the link function

$$g\left(\pi\right) = \ln\left(\frac{\pi}{1-\pi}\right). \tag{6}$$

Rewriting 4, we obtain

$$\pi\left(x\right) = \frac{\exp\left\{\beta'\mathbf{x}\right\}}{1+\exp\left\{\beta'\mathbf{x}\right\}}. \tag{7}$$

What we have written is equivalent to saying that

$$\mathsf{p}\left(\mathcal{C}_1 \middle| \mathbf{x}\right) = \pi\left(\mathbf{x}\right) = \sigma\left(\beta'\mathbf{x}\right) \tag{8}$$

where $\sigma\left(\cdot\right)$ is the logistic sigmoid and $\mathcal{C}_1$ is to identify class 1. Notice that it holds $\mathsf{p}\left(\mathcal{C}_2 \middle| \mathbf{x}\right) = 1 - \mathsf{p}\left(\mathcal{C}_1 \middle| \mathbf{x}\right) = 1 - \sigma\left(\beta'\mathbf{x}\right)$.

Suppose we are given a dataset $\{\mathbf{x}_i,\, y_i\}$ where now $y_i \in \{-1,\, 1\}$ and $\mathbf{x}_i \in \mathbb{R}^r$ for $i = 1,\, \dots,\, n$. Given our hypotheses, we can write the negative log-likelihood function as

$$\sum_{i=1}^{n} \ln\left(1 + \exp\left\{-y_i\, \left(\beta'\mathbf{x}_i\right)\right\}\right). \tag{9}$$

To this term we added a $\mathsf{L2}$ penalization term to obtain a more regularized result, preferring this to its $\mathsf{L1}$ version as the latter usually provides sparser solutions and we were working in an environment with only four covariates. This lead to the following problem:

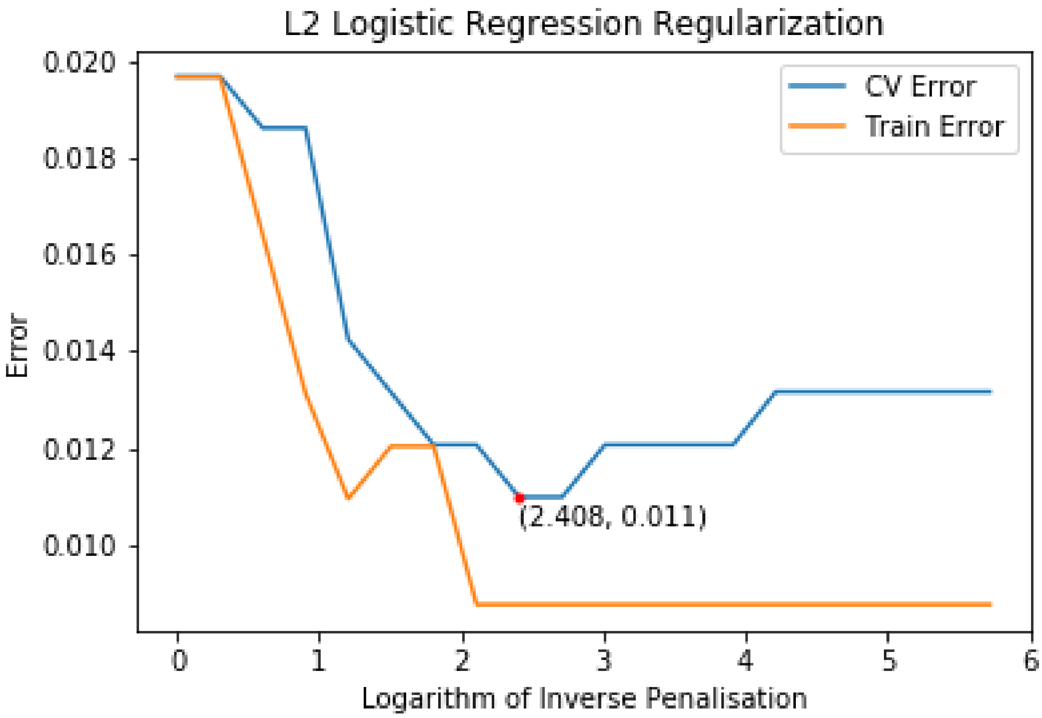
$$\min_{\beta} f\left(\beta\right) = \frac{1}{2}\beta'\beta + C\sum_{i=1}^{n} \ln\left(1 + \exp\left\{-y_i\, \left(\beta'\mathbf{x}_i\right)\right\}\right), \tag{10}$$

where $C > 0$ is a parameter that can be assigned in order to balance the two terms. To tune it, we relied on a stratified 5-fold cross-validation on the training set, picking the value that yielded the lowest value. From a numerical point of view, the trust region Newton method built in the ScikitLearn class for L2-logistic regression was used.

2.1.1 Optimal C

Optimal C value found: 256
At Cross Validation error value: 0.011
«««< HEAD

======= »»»> 1f465b1161c6e2e69022aa434928637e690d039d To tune the parameter, we relied on a stratified 5-fold cross-validation on the training set, picking the value that yielded the lowest value. From a numerical point of view, the trust region Newton method built in the ScikitLearn class for L2-logistic regression was used. The C parameter acts as a regularising term: the bigger it is, the lesser the regularising impact on the coefficients. From a bias-variance trade-off point of view, a bigger C term increases the variance and reduces the bias.



2.2 k-Nearest Neighbours

Suppose that we have some classes $\mathcal{C}_k$ such that each class contains N_k points and let $N = \sum_k N_k$ be the total number of points. Let us suppose that we are also given a distance function $\rho : \mathbb{R}^r \to \mathbb{R}^+$. To classify a new data point $\mathbf{x}$, we draw a sphere (according to ρ) such that exactly k points, regardless of their class, fall into it. If we let V be the volume of such sphere and K_k be the number of points of class k contained in it, we see that

$$\mathsf{p}\left(\mathbf{x}|\,\mathcal{C}_k\right) = \frac{K_k}{N_k V}, \tag{11}$$

the marginal probability is given by

$$\mathsf{p}\left(\mathbf{x}\right) = \frac{K}{NV} \tag{12}$$

and the prior distributions are

$$\mathsf{p}\left(\mathcal{C}_k\right) = \frac{N_k}{N}. \tag{13}$$

Therefore, applying Bayes' theorem we have

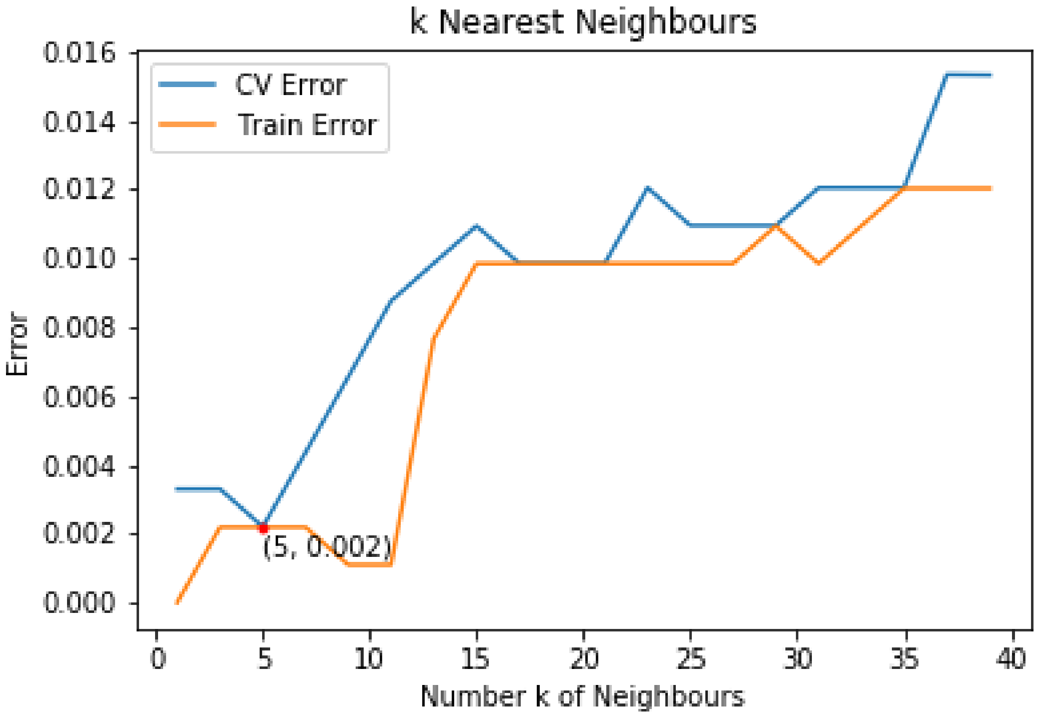
$$\mathsf{p}\left(\mathcal{C}_k|\mathbf{x}\right) = \frac{\mathsf{p}\left(\mathbf{x}|\,\mathcal{C}_k\right)\mathsf{p}\left(\mathcal{C}_k\right)}{\mathsf{p}\left(\mathbf{x}\right)} = \frac{K_k}{K}. \tag{14}$$

Comparing the postierior probabilities, we classified each data picking the maximum *a posteriori*. We consider the usual Euclidean distance for our distance function and trained only for odds value of k (between 1 and 39), picking the one that yielded the best result on a Stratified 5-fold cross validation on our data set.

2.2.1 Optimal K

Optimal K value found: 5
At Cross Validation error value: 0.002

Comparing the posterior probabilities, we classified each data picking the maximum *a posteriori*. We consider the usual Euclidean distance for our distance function and trained only for odds value of k (between 1 and 39), picking the one that yielded the «««< HEAD best result on a Stratified 5-fold cross validation on our data set. We only picked odd numbers to avoid having ties.
Alternatively, a rule for breaking ties could have been implemented. ======= best result on a Stratified 5-fold cross validation on our data set. We only picked odd numbers to avoid having ties.
Alternatively, a rule for breaking ties could have been implemented. »»»> 1f465b1161c6e2e69022aa434928637e690d039c



2.3 Kernel Naive Bayes Classifier

Let $\mathcal{C}_k$ be a class and let $\mathbf{x}$ our data vector. As done before, we can apply Bayes' theorem to provide the *a posteriori* probability of belonging to such class by stating 14. Naive Bayes' classifier is based on the hypothesis that the attributes of our data are conditionally independent, allowing us to rewrite the conditional likelihood as

$$\mathsf{p}\left(\mathbf{x}|\,\mathcal{C}_k\right) = \prod_{i=1}^{r} \mathsf{p}\left(x_i|\,\mathcal{C}_k\right). \tag{15}$$

This assumption, that is in fact *naive*, does not usually hurt too much as we are interested in picking the maximum probability a *posteriori*.

Naive Bayes is usually implemented by considering a probability density function, that is fitted to the feature being taken into account. In this particular instance, however, the conditional likelihood was modelled applying kernel density estimation techniques.

If we let Ω be a nonempty set, a symmetric function (*i.e.* a function that attains the same value for any permutation of its arguments) $K : \Omega \times \Omega \to \mathbb{R}$ is called a positive definite kernel on Ω if

$$\sum\sum c_i c_j K\left(x_i,\, x_j\right) \geq 0 \tag{16}$$

for any $n \in \mathbb{N}$, $x_1, \ldots, x_n \in \Omega$, $c_1, \ldots, c_n \in \mathbb{R}$.

In this specific setting, we considered the RBF Kernel (also known as Gaussian kernel), of the form

$$K_h(\mathbf{x}, \mathbf{x}) = \exp\left\{-\frac{\|\mathbf{x} - \mathbf{x}'\|}{h}\right\}. \tag{17}$$

Once the kernel form is fixed, the density for a point y can be computed by considering

$$\rho_K(y) = \sum K((y - x_i)/h).$$

Notice that the bandwidth here acts as a bias-variance trade-off parameter.

To classify the data we considered the logarithm of the posterior probability, obtaining

$$\ln(\mathsf{p}(\mathcal{C}_k|\mathbf{x})) \propto \ln(\mathsf{p}(\mathcal{C}_k)) + \sum \ln(\mathsf{p}(\mathbf{x}|\mathcal{C}_k)). \tag{18}$$

The prior was fixed simply by taking into accounting the elements per class and the likelihood by means of the KernelDensity function in the ScikitLearn library.

To classify our data into the two categories, we considered

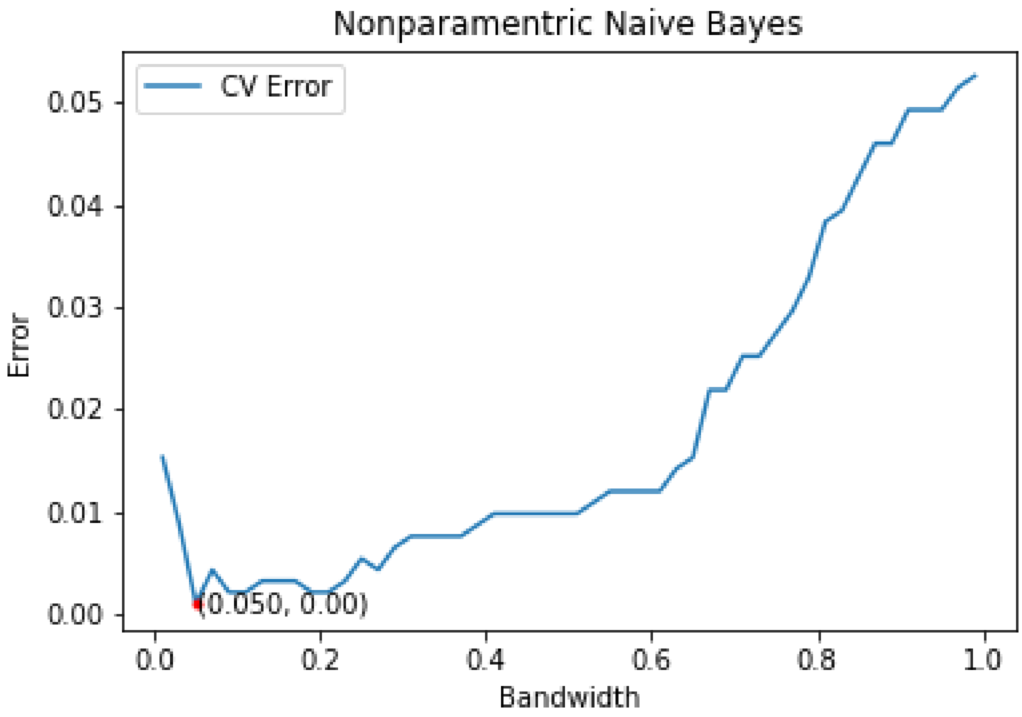
$$\arg\max_{k\in\{0,1\}} \ln(\mathsf{p}(\mathcal{C}_k)) + \sum \ln(\mathsf{p}(\mathbf{x}|\mathcal{C}_k)). \tag{19}$$

To set the parameter h, once again a 5-fold stratified cross validation was implemented.

2.3.1 Optimal Bandwidth found

Optimal Bandwidth found: 0.05
At Cross Validation error value: 0.00

«««< HEAD To set the parameter h, once again a 5-fold stratified cross validation was implemented.
Notice that in this case a smaller bandwidth allows for lesser bias and greater variance. =======
To set the parameter h, once again a 5-fold stratified cross validation was implemented. Notice that in this case, allowing for smaller values of h implies reducing the bias and therefore increasing the variance. »»»> 1f465b1161c6e2e69022aa434928637e690d039d



3 Comparing Classifiers

After parameterizing and fitting our train data into our classifiers we used our test data to calculate the true error of all three classifiers. It was important to split our initial data into training and testing data so we could have an unbiased set to now test our predictions.

3.1 True Error

The following results were obtained with a random split seed value of 10182017 and a cross validation random seed value of 52222.

3.1.1 Logisitic Regression

True Negative: 253
False Positive: 1
False Negative: 2
True Positive: 201
Test Error: 0.0066

3.1.2 K-nearest Neighbours

True Negative: 254
False Positive: 0
False Negative: 0
True Positive: 203
Test Error: 0.0000

3.1.3 Naive Bayes

True Negative: 254
False Positive: 0
False Negative: 0
True Positive: 203
Test Error: 0.0000

For comparing the three classifiers performances, we used McNemar's test with a 95% confidence interval.

3.2 McNemar's test (with continuity correction)

Let e01 be the number of examples the first classifier misclassifies but the second classifies correctly, and e10 be the number of examples the second classifier classifies incorrectly but the first classifier classifies correctly. The difference divided by the total follows approximately a chi-squared distribution with one degree of freedom:

$$\frac{(|e_{01} - e_{10}| - 1)^2}{e_{01} + e_{10}} \approx \chi^2_1 \tag{20}$$

From the predicted classification values of two classifiers we were able to calculate the e01 and e10 values by doing a simple loop and comparing both classifications with the real classification value:

Algorithm 1: McNemar's test

```
Data: first predictions, second predictions, real values
Result: McNemar's test
initialize e01 and e10;
for every value in real values do
    get current value of first predictions;
    get current value of second predictions;
    if $|currentFirstPrediction - value|$ equals 1 and second prediction - value equals 0 then
        increment e01;
    if first prediction - value equals 0 and $|currentSecondPrediction - value|$ equals 1 then
        increment e10;
return $\frac{(|e_{01}-e_{10}|-1)^2}{e_{01}+e_{10}}$
```

After running the McNemar's test to compare all three classifiers we got the following result:

Logistic Regression VS K-nearest neighbours: 1.333
Logistic Regression VS Naive Bayes: 1.333
K-nearest neighbours VS Naive Bayes: Not applicable

Which indicates that Logistic Regression and K-nearest neighbours classifiers have similar performance and also that Logistic Regression and Naive Bayes have a similar performance. The comparation of K-nearest neighbours and Naive Bayes was not possible with McNemar's test because they both achieved a 0% test error. But that could indicate both classifiers have similar performance.

4 Discussion and Conclusion

This assignment aimed to help us understand the fundamental principles of three classifiers: Logistic Regression, K-nearest Neighbours and Naive Bayes. It also allowed us to understand and apply some data management techniques such as preprocessing data (Load, Shuffle, Split) and the process of Cross Validation. The first step on this assignment was to load and preprocessing the data from a file which held data from the UCI machine learning repository about bank notes authenticity. The goal of the three classifiers was to predict either a bank note was real of fake based on four features (Variance, Skewness, Curtosis, Entropy). A test of performance of those classifiers was done to compare and eventually choose the best and to discuss if any is significantly better than the others.

From the McNemar's tests we noticed that Logistic Regression and K-nearest Neighbours classifiers performances were similar, the latter being just a bit better, but not enough to differentiate their performances. The same can be said when comparing Logistic Regression and Naive Bayes classifiers. Their performances were similar but Naive Bayes was slightly better, but then again, not enough to differentiate the two. For the test set we had, the McNemar's test with K-nearest neighbours and Naive Bayes was not possible because both classifiers had a success rate of 100% when predicting the classifications.

5 Bibliography

- Lecture 5, Lecture 5 Notes, Lecture 6, Lecture 6 Notes, Lecture 7, Lecture 7 Notes

- Tutorial Notes

- Scikit-learn Preprocessing docs

- Scikit-learn Supervised Learning docs

- Scikit-learn Model Selection docs

- Christopher M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006
- George Casella, Roger L. Berger, Statistical Inference, 2nd Ed., Duxbury Press, 2011