# Machine Learning - Report of the first practical work

João Albuquerque (45040) & Miguel Almeida (45526)

October 21, 2017

**Abstract**

Our work consists in parametrizing fitting and comparing Logistic Regression, K-Nearest Neighbours and Naive Bayes classifiers using a data-set given to us by the teacher. We implemented and trained our set with each of the classifiers and then found which value of each parameter would be best for each classifier. Then we trained our classifiers according to those parameters and we used McNemar's test with 95 per cent confidence interval to compare the results obtained by the classifier.

## 1 Introduction

Machine learning is the science of getting computers to act without being explicitly programmed. Machine learning is so pervasive today that we use it dozens of times a day without knowing it.

In this work we intend to see and study the results produced in a data-set trained by different classifiers, comparing the results to document the behaviour of each classifier on this data-set.

First of all we processed the data randomizing the order of the data points and standardizing the values. Then we splitted the data using stratified sampling making the use of cross-validation and testing a possibility.

We then found the best regularization parameters for the all the classifiers that we used plotting each one of them to study its behaviour, and checking its accuracy.

To determine those parameters we used cross-validation on two thirds of the data, leaving one third out for testing. We used 5 folds for cross-validation.

After we determined those parameters we used these parameters to train our test data so that we could achieve the best results with minimal error. And then we predicted the classes using those classifiers. Finally we compared the classifiers to see which fared better in classifying correctly the features in the data-set.

# 2 System Implementation

## 2.1 Implementation of the Naïve Bayes classifier

To implement our Naive Bayes classifier we divided our implementation into three steps or functions Fit, Score, Calc_FoldNB.

First of all in our fit function we divide all the data that is received by class, so that we can later for each of the features predict its class. Then we calculate the prior probabilities for each class, meaning that we calculate the percentage of each class in all the data. After that we calculate the Kernel Density of each feature for the two classes using the Gaussian Kernel and a given value of Bandwidth. Calculating the Kernel Density ensures us that we have the probability of, at any given sample or point in the sample place, the relative likelihood of the random feature being of each class. The function fit returns the Kernel Densities for each feature as well as the prior probabilities of each class.

Our score function takes those return values and for each feature it calculates the probability of it belonging to each class by doing the sum of all the probabilities density functions calculated for that feature in the function fit. Finally it compares the values of the two probabilities and based on those values it makes the prediction that the feature belong in the class with the higher value of probability.

Finally we use the Calc_FoldNB to use the score and fit and to test the accuracy of the classifier using the specific bandwidth.

## 2.2 The Three Parameters

To explain what are the three parameters that were optimized and their effects on their respective classifiers we need to know how each classifier works.

Lets start with the Logistic Regression, as the name points out the Logistic Regression is a regression model, but it can be used as a classifier, with the purpose of obtaining a decision hyperplane that separates different classes, which is the main objective of this practical work.

If the classes are not linearly separable, it is impossible to find a straight line that can separate the two classes. This happens because the Logistic Regression uses the probabilities to choose if something belong to class 0 or 1, written in another way, it uses the probability to know if an input belongs to a certain class. Without going deep in math this probabilities are calculated using the values of our data, and with this different values the classification model can start to learn that, if some feature always has, lets say, a value of more than 4, then almost every time that this happens the class this input represents is class 0, sub-sequentially if in the future it looks at something with a value of 4 or higher, then it has more probability to be classified as class 0 than it has to be classified as class 1.

Logistic Regression classifier can take a regularization parameter C, without regularization coefficients can be as large as necessary and the logistic function can be

very sharply sloped, which results in over-fitting the data, because the different sharply curves are trying to fit with the data, even though some of the points are likely to be an outliers and not representatives of the data in general. If we regularize the Logistic Regression classifier, the regularization will force the coefficients to be smaller and thus decrease the slope of the logistic function, so that's the job of parameter C, making the logistic function less steep.

The next classifier if the K-Nearest Neighbours classifier. The way the kNN classifier works is, given the data to train, the kNN algorithm identifies the k nearest neighbours of each point, regardless of labels and, using a distance function it chooses the class for that specific point, if the k nearest neighbours are from class 0, the probability of the point belonging to class 0 is higher as well.

For a 1-NN classifier, each training point defines a region in space, and the new points are labelled with the label of the closest point in the training set, resulting in a Voronoy tessellation. For classifiers with more neighbours, the labels are determined by the majority label of the k nearest neighbours and the tessellation becomes way more complex.

The parameter that can be used in the kNN classifier if the parameter K, with defines how many neighbours we want to take in consideration to classify a certain point in our data.

Last but not least comes the Naive Bayes classifier, the Naive Bayes uses a conditional probability model. For this classifier we use something more to classify certain points of our data, for example, if we know that our data has a lot more objects of class 1 than class 0 then we know that if a new element comes in, its more likely to be a class 1 than a class 0. In the Bayesian analysis, this belief is known as the prior probability. Prior probabilities are based on previous experience.

To classify a new point, the Naive Bayes classifier uses two things, prior probabilities and likelihood, the likelihood is obtained from how close to general features of some class the new point is, if we compute this point in a 2D model, the likelihood can be seen by how close the point are from each other.

In this classifier the parameter that can be optimised is the bandwidth, that is adjusted on the training set to minimise the overall classification error. Several tests that we performed showed that kernel density estimation can achieve statistically significantly better classification performance results, but only when appropriate bandwidth selection is applied, because of this, is our job to find the best possible value for the bandwidth.

## 2.3 Finding Optimal Values

As explain in the third topic of the report, each classifier has associated with himself a certain parameter. With the intention of reducing the error of each classifier, we used a technique called Cross-Validation, in order to find the best result for each of the parameters.

To do cross-validation, we partitioned our data into a number of disjoint folds, and we used a 5-fold cross-validation method, witch means that we grouped up our data in

5 different groups, all with similar size. Then we trained our model with all folds but one, validating on the fold that was left out, and repeat this for all the folds.

In all iterations, the validation error was saved with the intention to verify if, in each new iteration we obtained a new minimum value of validation error, if we did w would save that value in a variable that we would later use to test our test set. This way after every iteration we would have found the value of the parameter, being C,K or Bw, which would minimize the error of prediction in the respective classifier.

To get the optimal values for each parameter, in the Logistic Regression model we used a C variable that started at the value of 1 and for each iteration we doubled it for 20 sequential iterations, in order to see which value would minimize the error in the training set. For the kNN classifier we used a K variable starting at the value of 1 and use every odd value between 1 and 39 in subsequent iterations, finally, for the Naive Bayes implementation we used a bandwidth with values from 0.01 to 1 with a step of 0.02.

In all runs we verified that our values of training error and test error would behave differently.

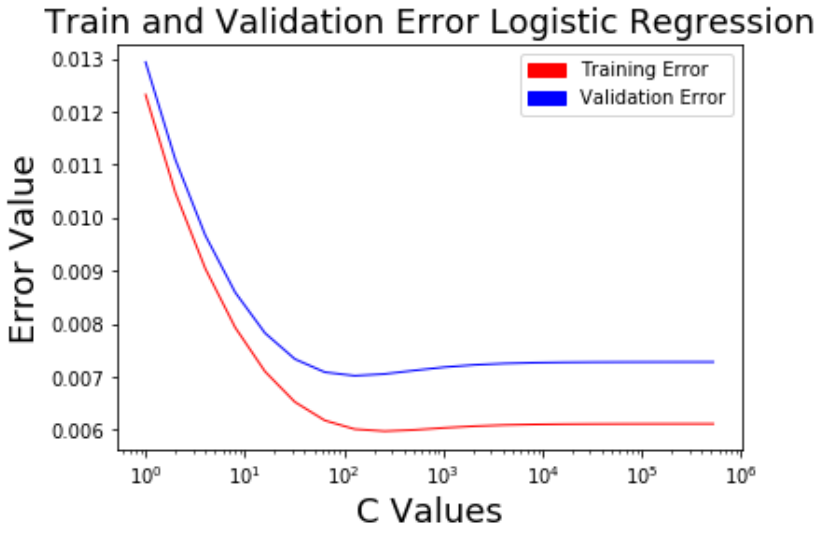


Figure 1: Graph of the errors for the Logistic Regression classifier.

In the Logistic Regression model both errors would start with higher value but both would have the same downwards curve to lower values as the value of C is increased, we also noted that the training error is always smaller than the validation error.


Figure 2: Graph of the errors for the KNN classifier.

In the KNN model they both behave very linearly, both increasing their value of error as the we increased the value of K. The training error is always slightly below the validation error.

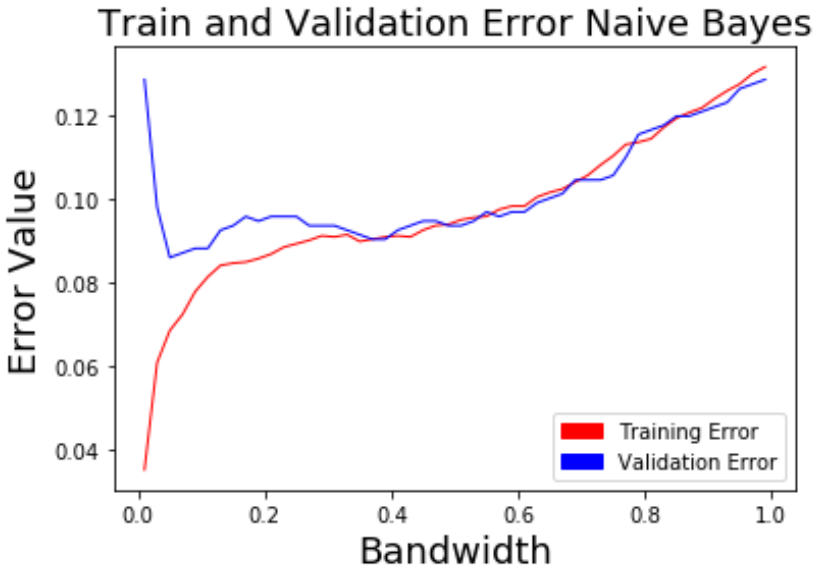


Figure 3: Graph of the errors for the Naive Bayes classifier.

In the Naive Bayes model, the validation error starts at very high values in contrast with the training error which starts at very low levels. They rapidly converge to a

similar value as we increase the bandwidth, in which afterwards they both behave very similarly, growing linearly.

It is also important to note that we always leave a third of the data-set out, so that after we find the optimal parameter for each classifier we would train a different part of the data-set than those in which the optimal parameter was found so that we minimize the risk of having overfitted results.

### 2.4 McNemar's test and Evaluation

McNemar's test is a statistical test used on paired nominal data. It is applied to 2 * 2 contingency tables with a dichotomous trait, with matched pairs of subjects, to determine whether the row and column marginal frequencies are equal.

We used this test to compare the performance of our classifiers in the sense that we pair up all the combinations (LogReg x KNN; KNN x NB; NB x LogReg) of our three classifiers and see for each value of feature which classifier got it right and which classifier got it wrong.

We used a 95 per cent of confidence in this test which means that if the result of this test is lower than the value of 3.84, the performance of both of the classifiers are very similar and we can't conclude which is better for classifying this data-set. However if the result is higher or equal to the value of 3.84 one of the classifier out-performs the other.

If the last case happens, then we need to see the true error of each of the classifiers, and the one with the lower percentage of error is the best classifier.

After all the tests were made and after running the McNemar's test we came to the conclusion that the Logistic regression and the KNN classifiers had a very similar performance and we can't conclude which one out-performs the other. However with the KNN and Naive Bayes classifiers the McNemar's test proved that one of them had a significant higher performance than the other, after calculating the true error we observed that KNN had the lower value of true error and so it had the best performance. Finally after analysing the result from the McNemar's test between the Naive Bayes and the Logistic regression classifiers, we saw that there was an higher performance form the Logistic Regression proving that the Naive Bayes model clearly has the poorest performance out of the three.

## 3 Conclusion

In this project we presented a way to parametrize, fit and compare different classifiers using the same data-set, to see which had the better performance. We used stratified sampling to split our data into tests and training sets. We learned how to implement the Logistic Regression model and how to find the best value of its regularization parameter. We learned how to implement the K-Nearest Neighbours model and how to find how many Neighbours would need to influence the decision of the classifier to minimize its

error. We learned how to implement the Naive Bayes model and how to find the best value of its Bandwidth parameter. We learn how to use the cross validation to decide the best value of the three parameters, by training the sets of data and seeing which one had the minimal value of error. We learned how to use the McNemar's test to compare the performance of each of the three parameters.

# 4 Bibliography

# References


[1] Bin Liu, Ying Yang, Geoffrey I. Webb and Janice Boughton, A Comparative Study of Bandwidth Choice in Kernel Density Estimation for Naive Bayesian Classification (2009)

[2] Christopher Bishop, Pattern Recognition and Machine Learning (2016);