Machine Learning - Report 1

António Ferreira & Ricardo Esteves

October 21, 2017

Abstract

For this assignment, we should implement three classifiers: Logistic Regression, K-Nearest Neighbors and Naïve Bayes, lectured on Machine Learning classes. To do this we will import our data from a .csv file. Each point in the data file has four features and one class. Based on this, we will analyse how our three classifiers perform while classifing this data.

1 Introduction

As stated above, we were going to implemnt three classifiers: Logistic Regression, K-Nearest Neighbors and Naïve Bayes. The most challenging part of this assignment is how we will implement Naïve Bayes Classifier, because we will implement our own version of Naïve Bayes with the help of the lectures notes. Lastly we will use McNemar's Test with a 95% confidence interval to compare our classifiers and consequently choose the one/ones that best fits our data.

2 Naïve Bayes Classifier Implementation

For the implementation of the Naïve Bayes classifier, we considered the following parameters: two training sets for the training features and their respective class values, one validation set for the validation features and a bandwidth value which will be used later on while creating the Kernel Density Estimators.

In our implementation, we start by splitting the features training set by class type into two arrays, one containing all the features of class type '0' and the other one of class type '1'. After this step, for each training features array previously split by class type, we calculate the logarithmic probability densities for each different feature in the array and add them all together to obtain the final probabilities array which will give us the probability of each data point belonging to a specific class (0 or 1).

To do this, after we have split the training features set by class types, we create a Kernel Density Estimator Object with the corresponding kernel (In this project, we use a Gaussian kernel because we are working with continuous data) and the bandwidth value obtained from the parameter. The next step is to fit (to train) all the previous split training data points with same features using the previously created Kernel Density Estimator. Then, we obtain the logarithmic probability densities by scoring the validation data points with the same feature with the score_samples method from the Kernel Density Estimator.

Lastly, the logarithmic probability density obtained will be added to the current probabilities in a matrix that contains the sum of all the logarithmic probability densities calculated for each feature and separated by class type. The final output will be the matrix described above, which contains the probabilities of each data point belonging to a specific class.

The matrix lines correspond to the number of classes (0 or 1) and the columns represent each data point in the validation set. For instance, given the position [0][15] of the matrix, its value will be the probability of that data point number 15 belonging to class 0.

Lastly to obtain the predicted class values for the given validation data set, we use the argmax function in the numpy library. The input for argmax method is the matrix obtained above. With this input argmax is going to compare for each data point its probabilities of belonging to class '0' or class '1', where the predicted class is the one that has the highest probability value.

3 Optimize Parameters and there Effects on the Classifiers

Logistic Regression classifier

Logistic Regression estimates the probability of an event occurring having been given some previous data. In our case we want to predict whether or not a given data point belongs a certain class given a known set of points which is the training set. This will be done by creating a Logistic Regression Model given a C value and fit all the data from the training set. Then, we predict the probabilities of each point, in the test set, belonging to each class and build a prediction class set based on these probabilities, which we can use to compare with the original class set (class test set) and obtain an accuracy measure for our model.

The "C" value is a regularization parameter meant to help reduce overfitting. When we train our logistic regression model we are finding the best C value that tells which is the best model to fit the data while reducing overfitting.If we use an optimal "C" value, we will find a balance between under-fitting and over-fitting whilst finding the model that best fits the data.

K-Nearest Neighbors classifier

The k-nearest neighbour's classifier is an example of a lazy learning method. The algorithm compares points in the test set with points in the training set. Given a k value, which represents the amount of neighbour's around a given point, the k-nearest neighbour's classifier will label (define its class value) a given point by the labels of the majority of the k points in the training set that are closer to this new point. For example, given a point X which has 3 neighbours (k = 3) and 2 of them belong to the class 'A' while the other one belongs to the class 'B', the KNN classifier will classify this new X point with the label A, because there was a bigger amount of class A points around X compared to B.

We should care when defining K values because if we have a small value of k we are only looking to its closest neighbour to define its label which could lead to a poorly classification as we are under an example of under-fitting. High values of K could also affect the classification because we are increasing the boundaries between neighbours which could lead to a possible example of over-fitting. The only way to obtain an optimal K value is to do a cross-validation method in order to determine with k value fits best our data.

One last important thing to notice is that we should only define odd values of k to avoid ties, for instance, if we have a point we wish to classify and we look at its two neighbours which have different labels, let's say labels 'A' and 'B', we can't decide whether or not the new point should be labelled 'A' or 'B' because there is no majority of labelled points.

Naïve Bayes classifier

The Naïve Bayes classifier works under the assumption that the features are conditionally independent of each other given their class. With that being said, we need to determine the conditional probability distribution of each feature given each class. Because our features have continuous values, we will use a Kernel Density Estimator with a Gaussian kernel, to find the distribution of the features from each class of our data set. The KDE's gives us an estimate of the joint probability distribution of the features given each class, under the naïve assumption that the features are conditionally independent given the class.

The KDE we used has one parameter called bandwidth, which determines the width of the kernel function, and different values of bandwidth's lead to different classifiers. In our solution, we found the best bandwidth values using cross-validation and noticed as the Figure 4 suggests that smaller values of bandwidth have a higher validation error and so does higher bandwidths' values, which could lead to under-fitting and over-fitting problems, respectively.

4 Optimal Values Found

In this section we will explain how we find the best parameter for each classifier.

Logistic Regression classifier

In the Logistic Regression classifier, we optimize the "C" parameter, in other words we found the "C" that optimize our problem by using cross-validation method. We use a range of 20 values and in each iteration, we duplicate the value of "C", and we choose the "C" that minimize our validation error.

We test multiple times and we obtain the best c value of 256.

K-Nearest Neighbors classifier

We found the best "K" on K-Nearest Neighbors classifier by cross-validation method again, so we use multiple values of "K" in a 40 range values (only odd values to avoid ties), and we choose the "K" that minimize the validation error.

We test multiple times and we obtain the best k value of 1.

Naïve Bayes classifier

In the Naïve Bayes Neighbors classifier, we optimize the “bw” (bandwidth) parameter on cross-validation method one more time. So, we use multiple values of “bw” in a 50 range of values starting with 0.01 and with a 0.02 step, and we choose the “bw” that minimize the validation error.

We tested multiple times and we obtain the best bw value of 0.13.

If we use the test set in an early evaluation, then the test error becomes a biased estimator and in the final our result is biased to, so to avoid that situation we use the test set only in the final evaluation.

5 MacNemar’s Test

MacNemar’s is a statistic test based on a prediction that allow us to say what is the best classifier for this application only if the MacNemar’s test result is above 3.84, because in this test we will use chi-square with a 95% confidence interval. To implement this, we had to calculate the prediction for all the three classifiers and compare then use the follow formula.

$$\frac{(|e_{01} - e_{10}| - 1)^2}{e_{01} + e_{10}} \geq 3.84$$

Figure 1: Formula of MacNemar’s Test using a 95% confidence interval where e01 and e10: count examples wrong in one and correct in the other.

MacNemar’s test between the three classifiers results:

- Logistic Regression Vs K-Nearest Neighbors: 0.8000
- Naïve Bayes Vs Logistic Regression: 19.3600
- K-Nearest Neighbors Vs Naive Bayes: 22.3214

So, analyzing these results we only can conclude about the Naïve Bayes vs Logistic Regression and K-Nearest Neighbors Vs Naive Bayes test, because these values are above 3.84 and in that case, we can reject the null hypothesis (that the two classifiers perform identically) with 95% confidence. When the MacNemar’s test gets a value above 3.84 we will analyze the accuracy of the two classifiers and compare them.

For the same test, we obtain the follow accuracy:

- Logistic Regression Accuracy: 99.1170%
- K-Nearest Neighbors Accuracy: 99.7792%
- Naïve Bayes Accuracy: 94.0397%

Comparing Naïve Bayes with Logistic Regression we can easily see the accuracy of Logistic Regression is slightly better than Naive Bayes and the same occurs between KNN and Naive Bayes accuracy. Leading us to conclude that Logistic Regression and K-Nearest Neighbors perform better vs. Naïve Bayes. Between Logistic Regression and K-Nearest Neighbors nothing can be concluded because we obtained a value bellow 3.84 in the MacNemar's test, concluding this way that these two classifiers perform identically.

6 Other important aspects

Why use Brier Score on Logistic Regression and Accuracy Score on KNN and Bayes?

Brier Score is used to verify the accuracy of a probability prevision, that we were doing on the logistic regression classifier.

The accuracy score is used to measure the number of times that a prediction is right and is used to get the accuracy of the train set vs. the test set. So, we use this on KNN and Bayes, because we need to predict the probabilities to use in the MacNemar's tests.

7 Plot Errors

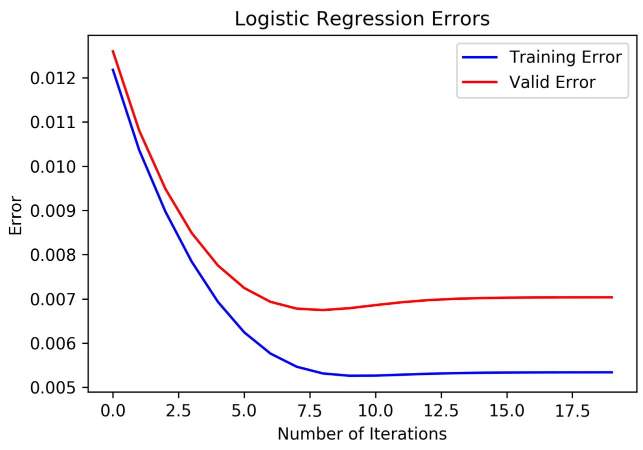

Figure 2: Shows the training error and the validation error of Logistic Regression in function of the number of iterations made in Cross Validation method.

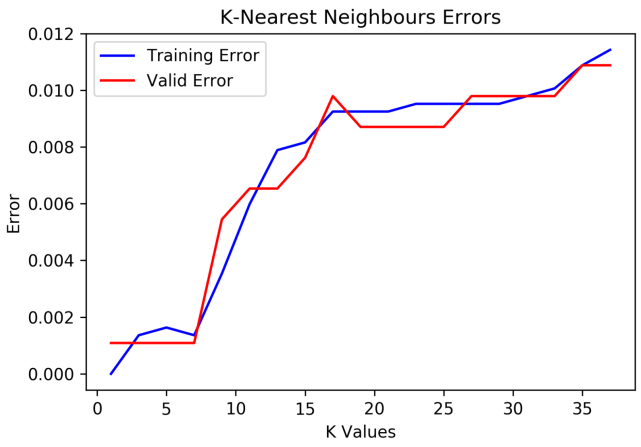

Figure 3: Shows the training error and the validation error of K-Nearest Neighbours in function of the "K" parameter.

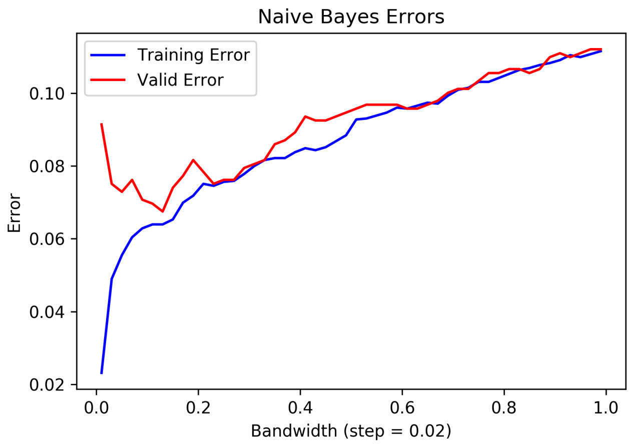

Figure 4: Shows the training error and the validation error of Naïve Bayes in function of bandwidth parameter.

8 Conclusions

Due to the nature of the assignment, we achieve the main goal of this project and that is implementing the three classifiers correctly. We approached the problem with realistic expectations, and so we think that reaching a Naïve Bayes implementation with considerably higher accuracy is our main goal and another goal is keeping our code simple and clean.

By developing this assignment, we have little bit of struggle to get used to python language, because is our first contact with the language, but in the end, we overcome that.

As stated in the Plot errors section, we did multiple accuracy tests and plotted the validation and trainning erros. We also compared our grafics with the ones in the lectures notes, leading us to consider that our implementation is correct.

6 Bibliography

[1] Logistic Regression Score function,
`http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html#sklearn.linear_model.LogisticRegression.score`

[2] K-Neighbors Classifier Score function,
`http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier.score`

[3] Naïve Bayes accuracy Score function,
`http://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html`

[4] Video Lectures,
`https://www.youtube.com/watch?v=EQ3KvuKnOrs&list=PL2TJmrWpyg7Su-Td5pojMSfjJCmkLbqwv`

[5] Lectures Notes,
`http://aa.ssdi.di.fct.unl.pt/files/LectureNotes.pdf`