# EFC2 - Exercise 2

Rafael Claro Ito (R.A.: 118430

September 2019

## 1 Source files

**All code cited and all figures showed here can be found at the following GitHub repository:**
`https://github.com/ito-rafael/IA006C-MachineLearning/tree/master/efc2`
**In this repository, one can found the following files:**

- Jupyter Notebook
  - efc2_pre-ex1.ipynb
  - efc2_ex1_binary_classification.ipynb
  - efc2_ex2_multiclass_classification.ipynb
  - efc2_ex2_knn.ipynb
- LaTeX
  - efc2.tex

**The notebook "efc2_pre-ex1" plots the histograms for the exercise 1 and it is used for data visualization. It shows the input features histograms for the raw data and after a data standardization. Also, it shows the correlation between these data.**

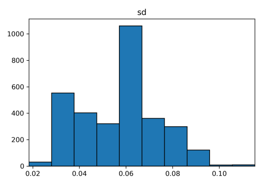
**The notebook "efc2_ex1_binary_classification" effectively implements the logistic regression used to perform a binary classification proposed in exercise 1.**

**The notebooks "efc2_ex2_multiclass_classification" and "efc2_ex2_knn" implements the algorithms to perform a multiclass classification proposed in exercise 2. The former one uses the softmax approach while the latter one implements the K-Nearest Neighbors (KNN) algorithm.**

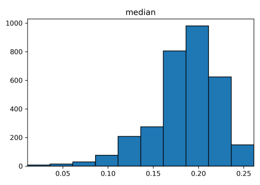
## 2 Part 1 - Binary Classification

### 2.1 a) Input features characteristics analysis considering the histograms and correlation measures between them.

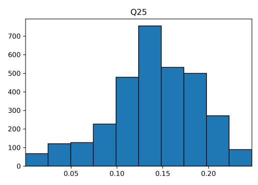
**The first thing we did for this item, was to plot the histograms of the data before and after standardization. The histograms for the raw data can be seen in Figure 2 and the histograms for the standardized data can be seen in Figure 3. To perform the standardization, the StandardScaler class provided by the scikit-learn library was used.**



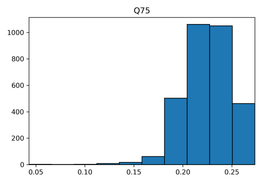
(a) sd



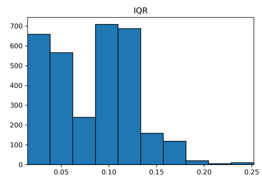
(b) median



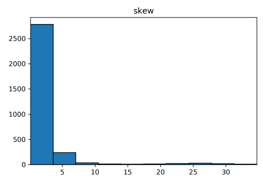
(c) Q25



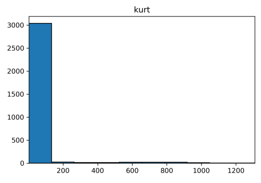
(d) Q75



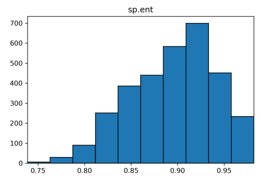
(e) IQR



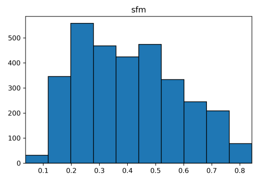
(f) skew



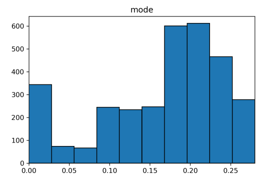
(g) kurt



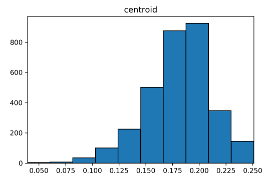
(h) sp ent



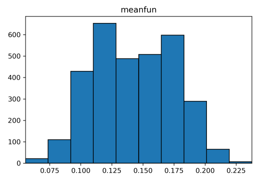
(i) smf



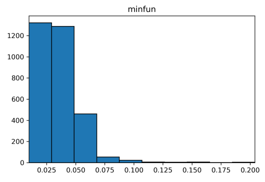
(j) mode



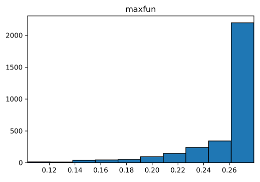
(k) centroid



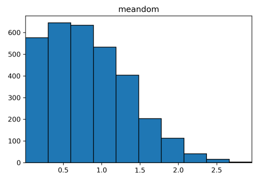
(l) meanfun



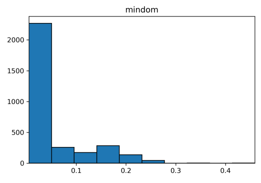
(m) minfun



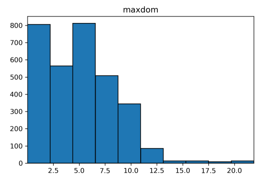
(n) maxfun



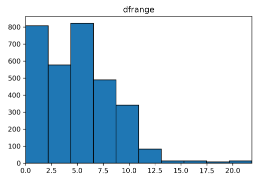
(o) meandom



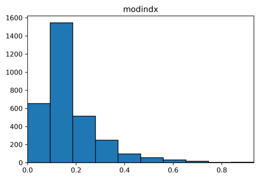
(p) mindom



(q) maxdom

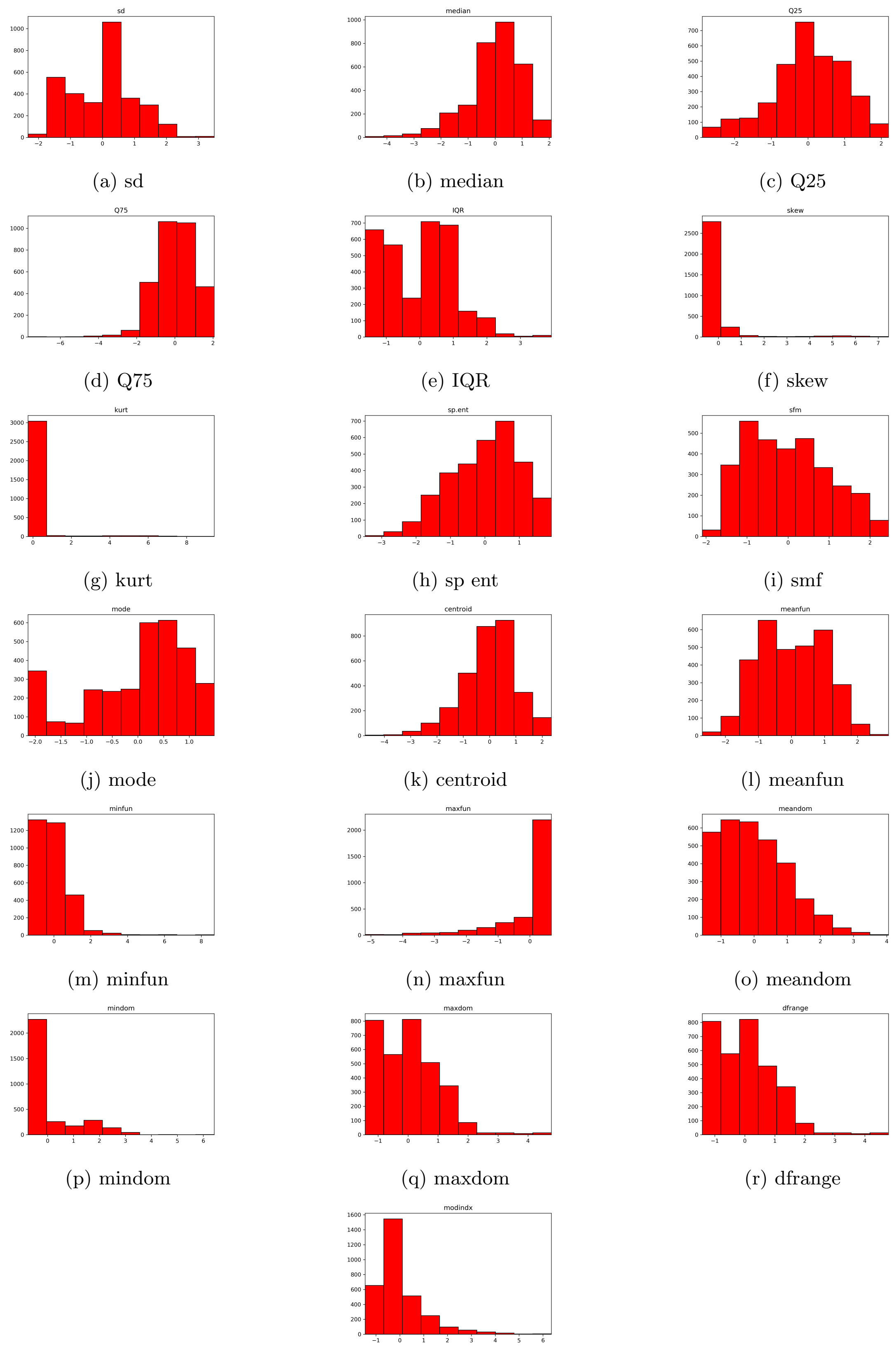


(r) dfrange



(s) modindx

Figure 2: Histograms of raw input features



(a) sd (b) median (c) Q25

(d) Q75 (e) IQR (f) skew

(g) kurt (h) sp ent (i) smf

(j) mode (k) centroid (l) meanfun

(m) minfun (n) maxfun (o) meandom

(p) mindom (q) maxdom (r) dfrange

(s) modindx

Figure 3: Histograms of input features after data standardization

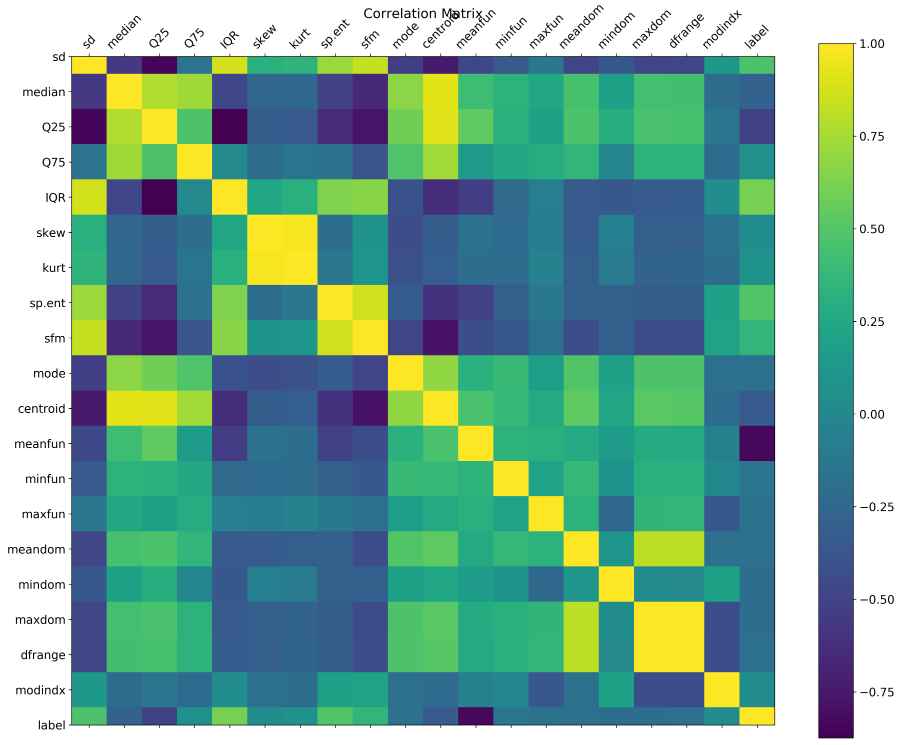


Figure 4: Correlation between input features

**In the Figure 4 we can see the correlation matrix, with values between 0 and 1, mapped in colors.**

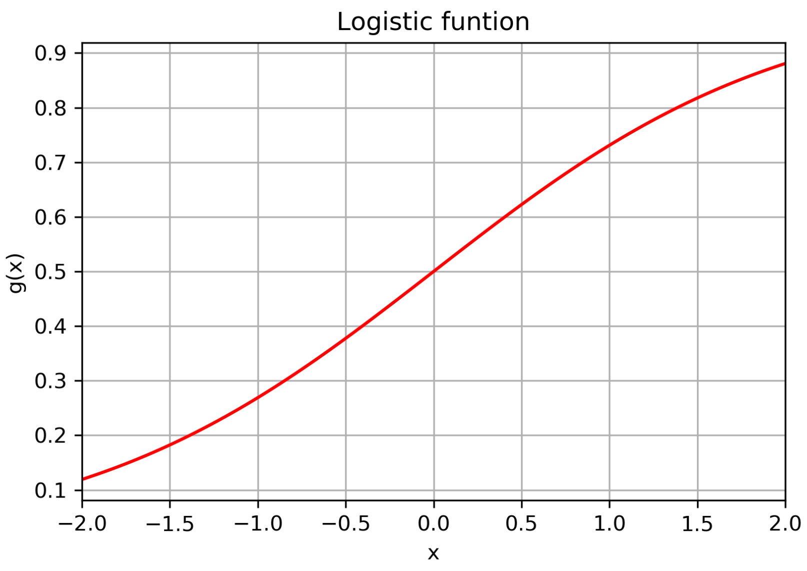


Figure 5: Logistic function

**In the first notebook we also plotted the logistic function, used in the model proposed for binary classification. The Figure 5 shows the logistic function in an interval of -2 and 2.**

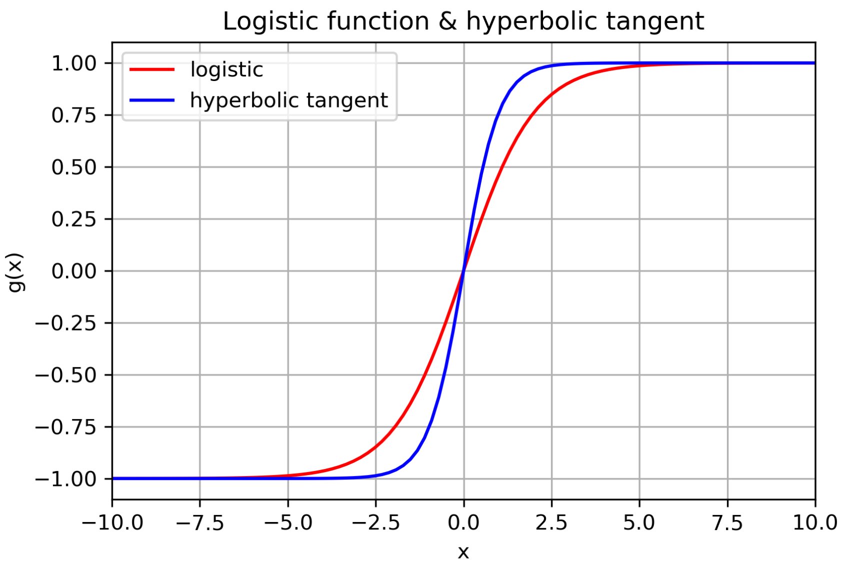


Figure 6: Comparasion between hyperbolic tangent and logistic function(scaled and with offset)

**Before actually starting the binary classification proposed in exercise 1, one fact can be noted when comparing the logistic function (used in this exercise) and the non-linear function (hyperbolic tangent) used to transform the data in the exercise 2 of the previous list of exercises EFC1. The shape of these functions are very similar. Just to visualize the shape of the functions, the two functions are plotted in the same Figure 6. Here, the logistic function was multiplied by two, and subtracted by 1 in order to show the similar shape with the hyperbolic tangent (but with a different scale).**

## 2.2 b) Logistic regression, ROC and F1-score curves.

**In this subsection the binary classifier is actually built. Here we are using the shuffle function provided by the scikit-learn library. First, we separate the data according to the label. We do this to ensure that we have the same rate of both classes in the training set and also in the test set. The data in both groups are then shuffled.**

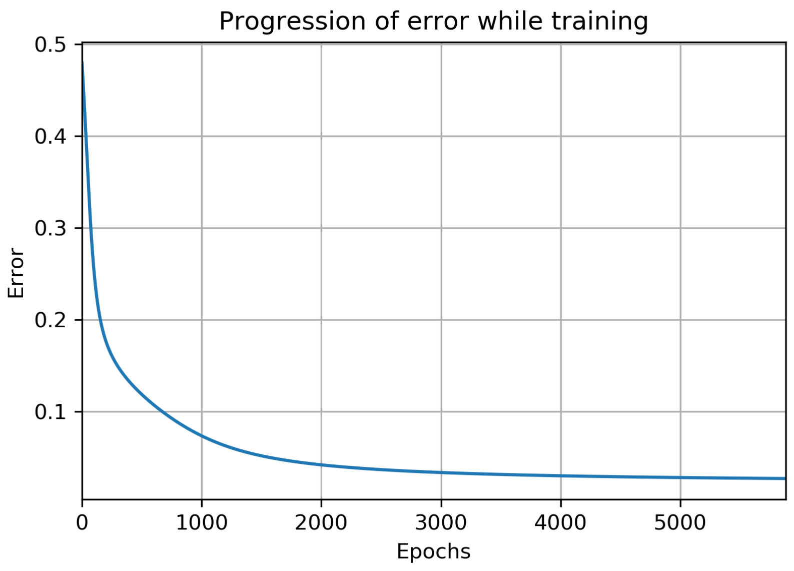


Figure 7: Error progression in training

**The training step is performed until the difference between successive iteration errors is under a provided tolerance. We are using a tolerance of $1.10^{-6}$, which means that when the error between the real and the predicted class for a iteration is less than the error in the previous iteration minus this tolerance, we should stop the training process. The error progression with the number of epochs is shown in Figure 7. We are considering the whole batch of samples for each step during the training.**

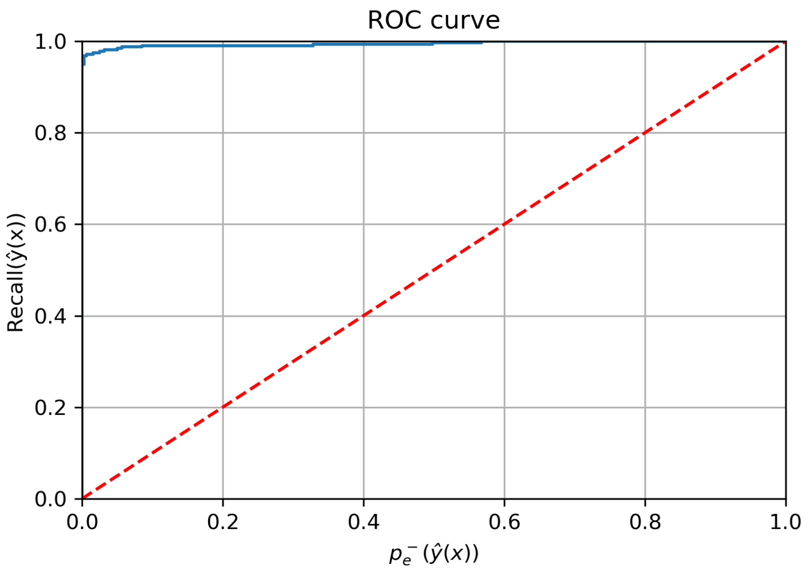


Figure 8: ROC curve

**Now we are going to plot the receiver operating characteristic (ROC) curve and the F1-score for the classifier. For this, we are using the metrics provided by the scikit-learn library. The ROC curve is plotted with the recall (true positive rate - tpr) being the y axis and the false positive rate (fpr) being the x axis. The ROC curve can be seen in the Figure 8. We can see that the ROC curve is located at a high part of the graph as well it is to the left, indicating a good performance.**

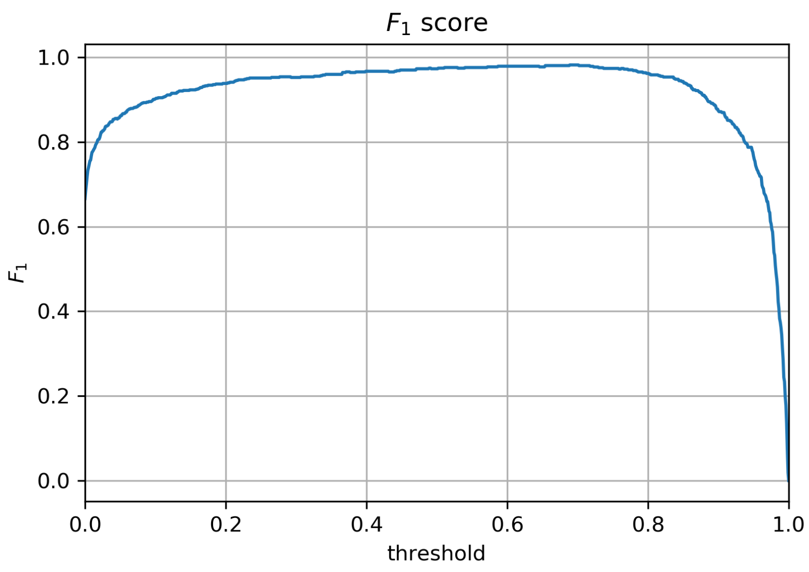


Figure 9: F1-score curve

**We now change the threshold used to classify a sample with the positive or negative class. We use values of threshold from 0 to 1 with step of 0.001, i.e., we are using 1000 thresholds values. For each value of threshold we calculate the F1-score. Figure 9 shows this plot.**

## 2.3 c) Threshold, confusion matrix and accuracy.

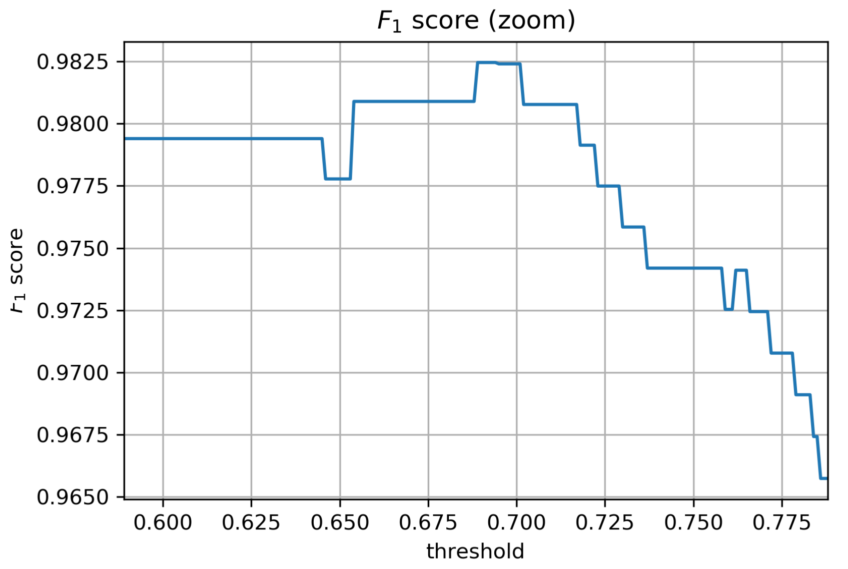


Figure 10: Zoom in F1-score curve

**If we give a zoom in Figure 9 highlighting the points near the maximum of the curve, as shown in Figure 10, we can choose the most appropriate threshold value. Knowing that the F1-score can be interpreted as a weighted average of the precision and recall (with 1 being the best value and 0 being the worst), we then choose for the threshold which maximizes the F1-score. This gives us a threshold of 0.689.**

## 3 Part 2 - Multiclass Classification

### 3.1 a) Logistic regression (using softmax approach)

**We start this exercise by** **output** ¡br¿ 1 – caminhada ¡br¿ 2 – subindo escadas ¡br¿ 3 – descendo escadas ¡br¿ 4 – sentado ¡br¿ 5 – em pé ¡br¿ 6 – deitado ¡br¿ ¡br¿ **one-hot enconding** ¡br¿ $[1\ 0\ 0\ 0\ 0\ 0]^T$: walking ¡br¿ $[0\ 1\ 0\ 0\ 0\ 0]^T$: climbing stairs ¡br¿ $[0\ 0\ 1\ 0\ 0\ 0]^T$: going down stairs ¡br¿ $[0\ 0\ 0\ 1\ 0\ 0]^T$: seated ¡br¿ $[0\ 0\ 0\ 0\ 1\ 0]^T$: standing ¡br¿ $[0\ 0\ 0\ 0\ 0\ 1]^T$: lying ¡br¿

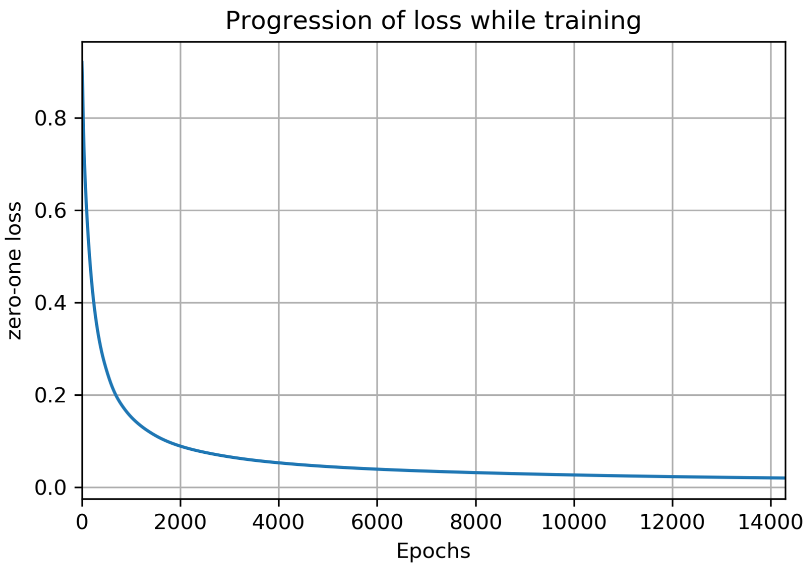


Figure 11: Zero-one loss curve

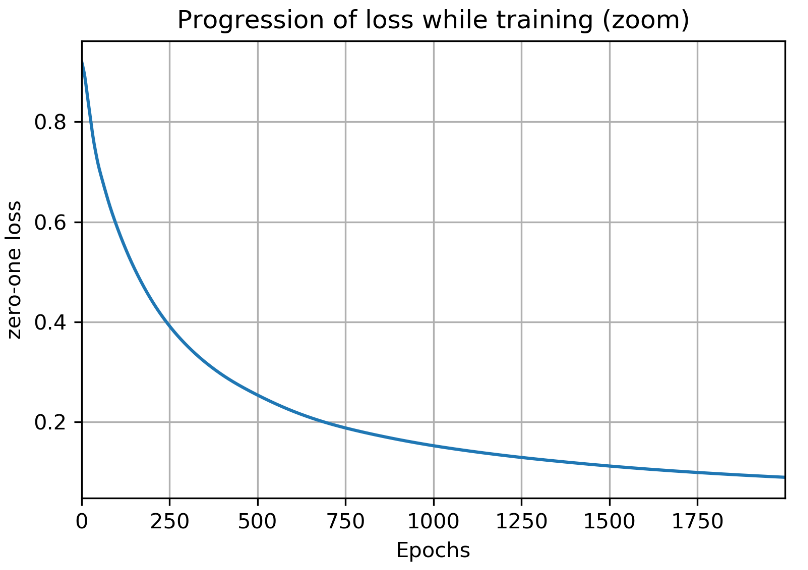


Figure 12: Zoom in zero-one loss curve

## 3.2 b) K-Nearest Neighbors

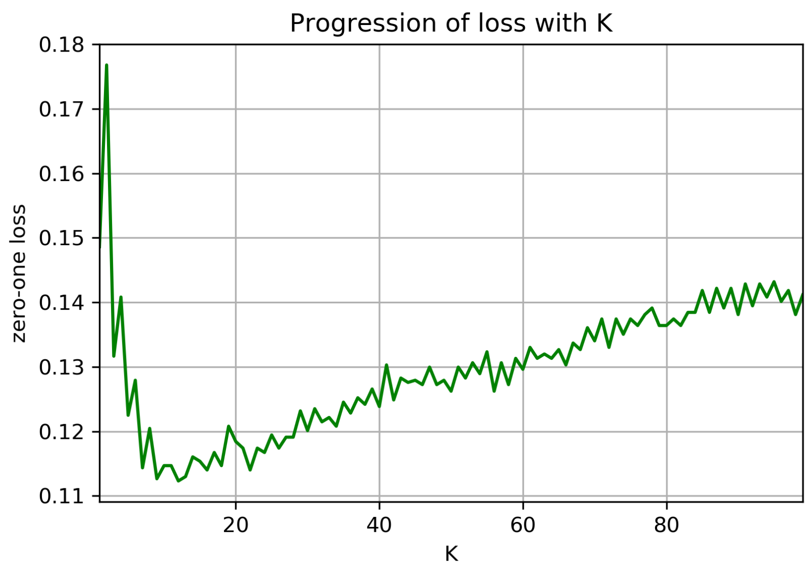


Figure 13: Zero-one loss curve (uniform weights

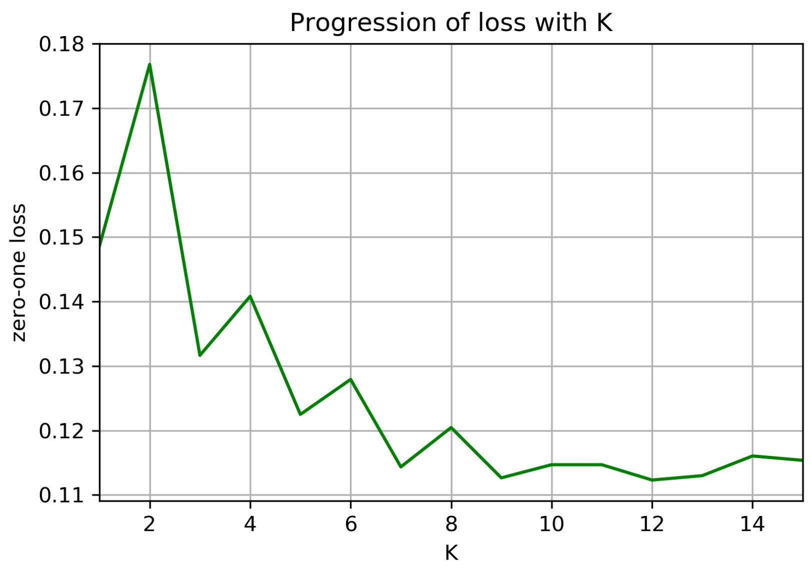


Figure 14: Zero-one loss curve (uniform weights

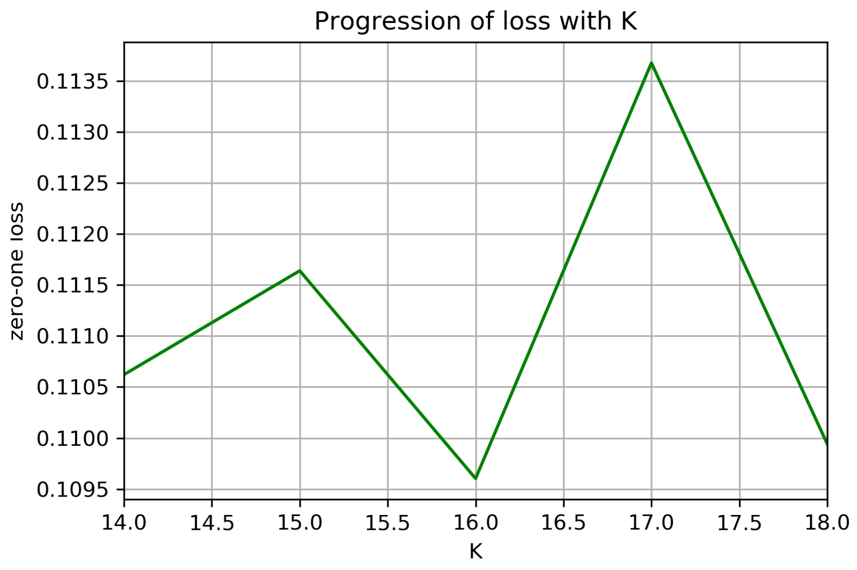


Figure 15: Zero-one loss curve (uniform weights

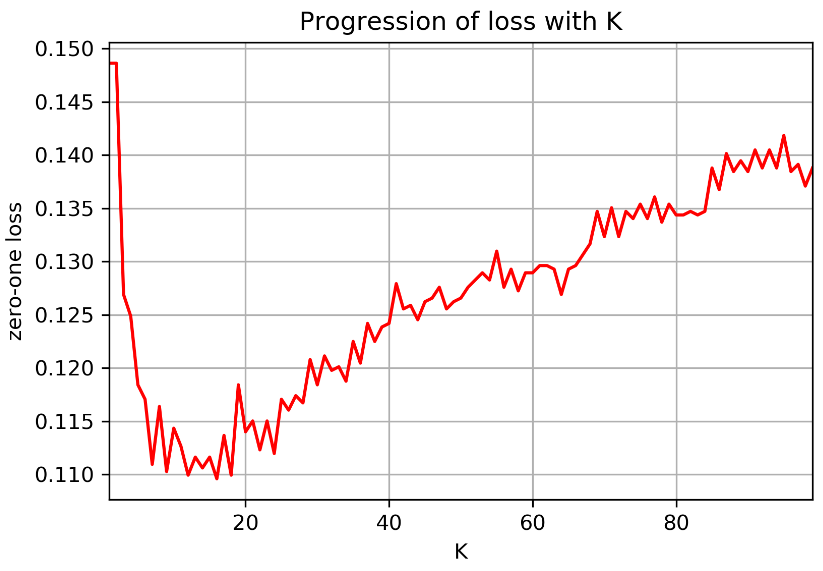


Figure 16: Zero-one loss curve (inversely proportional to distance weights

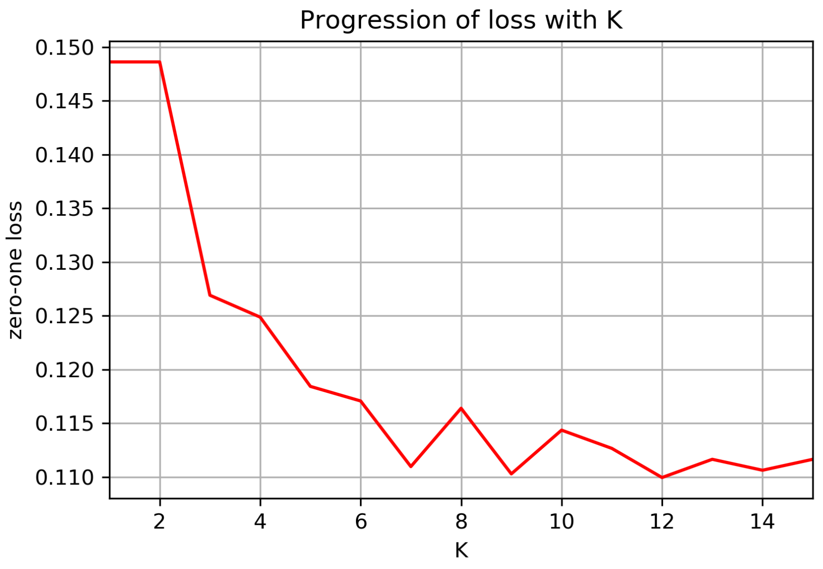


Figure 17: Zero-one loss curve (inversely proportional to distance weights

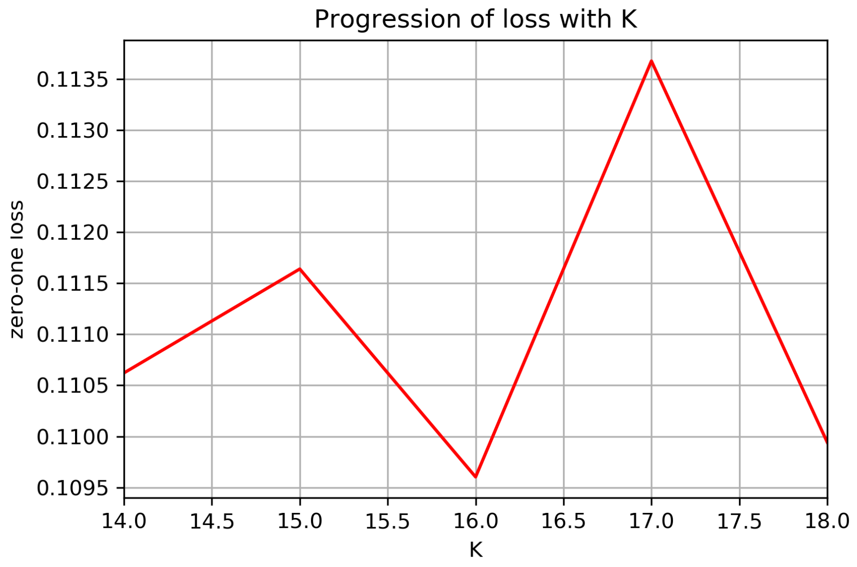


Figure 18: Zero-one loss curve (inversely proportional to distance weights