MACHINE LEARNING

Software Project 2, Component 4

Jose Luis Galiano Gómez

FINAL IMPLEMENTATION DECISIONS

Before analyzing the performance results obtained, there were some decisions outlined in the previous component that were not fully decided then, and need to be specified now that the project is complete.

Firstly, it has been decided to keep outliers in the dataset, given the very low amount of them with respect to the full dataset. At worst, they will not impact the performance or ability to generalize due to their low volume, and at best they will improve robustness and represent better a real-world application.

Both the activation function and model architecture remain the same as in the previous component; performance is more than good enough, so there is no need to add hidden layers or change the activation function. The same can be said about the learning rate chosen, which was the standard value of 0.1, and the loss function (binary cross-entropy).

In the end, the learning method chosen was standard Gradient Descent, rather than Stochastic Gradient Descent. SGD is great for improving computational efficiency, especially when training large models, but given the simplicity of the dataset and network architecture GD suffices.

PERFORMANCE EVALUATION METHOD

The method chosen will be 10-fold cross-validation. The specifics of it were already discussed in the previous component, but in a few words, the dataset is divided into 10 subsets (folds), and the model is trained and evaluated 10 times. In each iteration, one fold is used for testing, while the remaining 9 folds are used for training. This process is repeated, with a different fold serving as the test set in each iteration. The final performance metric is the average of the metrics obtained in all 10 iterations. Cross-validation helps assess a model's generalization performance and reduces the impact of data partitioning on evaluation.

EVALUATION MEASURES

Each of the following evaluation measures was computed in each iteration of the cross-validation and put into a list, and then the mean of the lists of each measure was calculated, those being the values presented here. The model has been trained and tested from scratch several times and the values for each measure have been monitored for every iteration, and it has been observed that the values are very similar across all iterations.

Accuracy (proportion of correctly classifed instances out of the total instances)**:** **0.9873851294903927**

Precision (proportion of true positive predictions out of all positive predictions made)**:** **0.9802697727480705**

Recall (proportion of true positive predictions out of all actual positive instances)**:** **0.9974999999999999**

F_Score (harmonic mean of the precision and recall)**:**
0.9887047332301278

AUC (area under the Receiver Operating Curve, which plots true positive rate against false positive rate at various thresholds)**:**
0.9864355597473118

AUPRC (area under the Precision-Recall curve)**:**
0.9914229334846855

OVER-FITTING CONCERNS

It can be easily observed that these performance values are ridiculously good. This immediately raises concerns of possible over-fitting. However, this is precisely the reason why cross-validation was used, as well as the use of a model as simple as possible, which are less prone to over-fitting. Looking at the dataset instead of the model, the classes are well-balanced (54.96% one class and 45.05% the other), there is little to no noise, and the dimensionality is low. It is unlikely that the problem lies with the dataset. Due to a lack of time further over-fitting measures were not implemented, but the most sensible options would be to add regularization techniques (L1 and L2) and dropout (randomly deactivating some neurons to prevent the network from relying too much on specific neurons).

CONFIDENCE INTERVALS – BOOTSTRAPPING

Confidence intervals provide a range of values within which we can be reasonably confident the true metric lies. They quantify the uncertainty associated with estimates, such as means, proportions, or other statistics. They also help convey the precision of an estimate and provide a more informative perspective than a single point estimate. They are valuable in statistical analysis for understanding the range of likely values for a parameter, making decisions, and comparing results.

Calculating confidence intervals for a specific performance measure involves using statistical methods to estimate the range within which the true value of the metric is likely to fall, considering the variability in the data. Bootstrapping is a resampling technique where multiple samples are drawn with replacement from the original dataset. This process is repeated many times, and the variability in the computed metric across these samples is used to estimate the confidence interval.

Upon obtaining the list of values for the different performance metrics across the 10 steps of the cross-validation, the mean is calculated and displayed for each metric, as well as the confidence values, using the Sci-Py stats library.