IDS2018 - Assignment 2

Isabela Blucher

March 3, 2018

Running the program

The implemenration for this assignment was done in such a way that with command line input arguments, the user can choose if they want to run Exercises 2 and 3 with normalized or non-normalized data. For non-normalized data, type:

```
$ python assignment2.py 0
```

And for normalized data (Exercise 4):

```
$ python assignment2.py 1
```

Exercise 1 (Nearest neighbor classification)

The 1-NN classifier from scikit-learn was used to create our model. The given training data was fitted on the classifier and then it became possible to query it about the accuracy score of the prediction of the test data. The accuracy score for the test data was 0.945993031359, which means that our classifier predicted the targets for the test data with great exactness. If we run the same test for our training data the accuracy is 1.0, because our classifier was fitted with the training dataset.

Exercise 2 (Cross-validation)

For this part of the assignment, a loop was implemented so that for every iteration a new value of k was tested. The k-nearest neighbors classifier was initialized with different values of $k \in 1, 3, 5, 7, 9, 11$ and then its performance was estimated with a 5-fold cross-validation. In the cross-validation process our training data was split so that we could train and test our classifier with different inputs and targets. After the fitting and scoring of the data, the 0-1 loss was calculated with the `zero_one_loss` function from `sklearn.metrics` for each one of the folds and then, an average was computed. With the average it is possible to see which value of k is the best.
The results were

Table 1: Average 0-1 loss for different k values

1	3	5	7	9	11
0.046	0.037	0.044	0.05	0.055	0.056

With the results documented on Table 1, we can set our $k_{best} = 3$.

Exercise 3 (Evaluation of classification perfomance)

Building a 3-NN classifier using the `IDSWeedpCropTrain.csv` and then evaluating it based on the `IDSWeedpCropTest.csv` we have an accuracy score of 0.949477351916, which is better than the score obtained with the 1-NN on Exercise 1. Also, if we evaluate the performance on our training data, the accuracy score is 0.971, which indicates a very good classification exactness.

Exercise 4 (Data normalization)

Out of the three given versions of data normalization from `scikit-learn`, the first one is correct, and was used on the implementation of this exercise. It is correct because our objective is to compute the mean and standard deviation of the training set to standardize the features by removing their mean and scaling them to unit variance. We want to apply that same transformation to our test set. That is why it makes no sense initializing a `StandardScaler` for the test set, which makes the second version of the preprocessing wrong. As for the third version, it is wrong because we want the first transformation (the fitting) to only be applied to the training set and not to both training and test datasets together, it would not give us accurate scaling when applying the transformations to both datasets later on.

Repeating the process from exercises 2 and 3 to find the best value of k using 5-fold cross-validation we get the following 0-1 loss values

Table 2: Average 0-1 loss for different k values with normalized input data

1	3	5	7	9	11
0.041	0.036	0.044	0.047	0.048	0.051

From Table 2, we can set our $k_{best} = 3$ yet again.

Building a 3-NN classifier from the normalized training dataset and then evaluating it based on the test dataset we have an accuracy score of 0.959930313589. Evaluating it based on the training set the accuracy score is 0.972.

With data normalization the average 0-1 loss values when computing the 5-fold cross-validation are smaller and the accuracy score of our 3-NN classifier is greater. With that we can conclude that centering and normalizing the data is a good preprocessing practice when dealing with data analysis and interpretation as it can better the performance of the tool being used (in this case, the KNN classifier).