IDS - Assignment 2

Bjarke Kingo Iversen - SWD457

March 2018

Introduction

For all my code, please see the Jupyter notebook file ***Assignment 2.ipynb*** in the ***src.zip*** folder.

1 Exercise 1: Nearest neighbor classification

- Training accuracy for 1-NN: **1.0**
- Test accuracy for 1-NN: **0.9460** rounded to 4 decimal places

We see that the training accuracy actually has no errors in this case. This is due to the fact that we use 1 neighbor for classification and thus a consequence of this is that the bias is high, since our classification model is identical to the training data. Also the variance will be higher since the probability that we've modelled the noise in the training data is high. As a consequence the test accuracy may be worse than of a multiple neighbor classification.

2 Exercise 2: Cross-validation

For this part i made a method cross_validator, that takes a kNN-classifier and trains on the given k over all the 5 folds and finally returns the mean error rate of using this k for cross-validated classification.
I also made a best_k method that takes an array of k's and feeds it to the cross_validator and return the best k based on best average accuracy score.
I found that the k_{best} for this exercise was $k = 3$.

3 Exercise 3: Evaluation of classification performance

Given that i found my best hyperparameter $k = 3$ we feed it to the kNN-classifier to obtain the following results:

- Training accuracy for 3-NN: **0.971**
- Test accuracy for 3-NN: **0.9495** rounded to 4 decimal places

And thus we already see the improvement on testing accuracy by picking our hyperparameter $k = 3$.

4 Exercise 4: Data normalization

Which method is the right one for prepossessing and transformation?

```
XTrainN = scaler.transform(XTrain)
XTestN = scaler.transform(XTest)
```

The first method here is the right one, since we only use training data for preprocessing. The deviation and mean are only supposed to be calculated via. the training data.

```
XTrainN = scaler.transform(XTrain)
scaler = preprocessing.StandardScaler().fit(XTest)
XTestN = scaler.transform(XTest)
```

The second method is wrong since we use test data for the prepossessing part and transform the test data with respect to this normalization.

```
scaler = preprocessing.StandardScaler().fit(XTotal)
XTrainN = scaler.transform(XTrain)
XTestN = scaler.transform(XTest)
```

The third method is also wrong since we use both the train and test data for the normalization.

We normalize and transform the data to find that the variance is now 1, and the mean is now -6.5588560224e-18. However we regard this as 0 mean, since its numerically bounded by floating point precision given us a minimum of E-18 decimals places. The k_{best} using the cross-validation on the transformed data is $k = 3$ and the accuracy is thus:

- Training accuracy for 3-NN: **0.972**
- Test accuracy for 3-NN: **0.9599** rounded to 4 decimal places

We can thus conclude that we got higher testing accuracy by picking the right hyperparameter, and we got even higher testing accuracy by normalizing the data.

- Testing accuracy for 1-NN: **0.9460** rounded to 4 decimal places
- Testing accuracy for k_{best}-NN: **0.9495** rounded to 4 decimal places
- Testing accuracy for k_{best}-NN, normalized: **0.9599** rounded to 4 decimal places