

Exercise 1

Your Name

November 10, 2024

Abstract

This is the abstract of the document. It provides a brief summary of the content.

Contents

1	Introduction	2
2	Methodology	2
2.1	Subsection Example	2
3	Preprocessing	2
4	Hyperparameters	3
5	Conclusion	3

1 Introduction

This is the introduction section. Here you can introduce the topic of your document.

2 Methodology

This section describes the methodology used in your work.

2.1 Subsection Example

This is an example of a subsection.

3 Preprocessing

We will do four different measures to properly preprocess the data. These are: data imputation, encoding of categorical variables, scaling of numerical values and dimensionality reduction. In this section we will look at each dataset and describe which tasks can be used on the dataset and how a specific task is performed. Then we will compare the different settings for all the models and determine their effectiveness using measures described in the previous section. In order to make this a meaningful comparison, we will use the default hyperparameters for all classifiers and we will split the data equally in the cross-validation.

Data leakage occurs when data is preprocessed before it was split into the training and test set. This way information from the test set is used to preprocess the training set. For example the MinMaxScaler uses the minimum and maximum values to scale these values properly. The values should only be computed on the training set and then be used on both training and test set. This can get rather difficult when using cross-validation. In order to avoid this, we will use the pipelines. Pipelines are a feature of scikit-learn that allow for the chaining of multiple transformers and estimators and also avoid data leakage. We will define a different pipeline for each preprocessing task and variant and chain these pipelines to compare the different settings.

Congressional Voting: As this dataset contains missing values we will need a strategy to impute these values. It is important to note that the missing values are all labeled as *'unknown'*. As all the values are categorical there are two possible strategies for data imputation. We can either use the most frequent value of the respective attribute in the training dataset or we can treat the missing values as a separate category. We will use one-hot-encoding for the categorical values, except for the binary ones. Binary values will be encoded as 0 and 1. By using one-hot-encoding the two different treatments of the missing values will yield a different encoding. Still all the values remain binary, so no further scaling is needed. As the dataset is rather small, we will not use dimensionality reduction. This means that we will only compare the two different strategies for data imputation. In table 1 the results are visualized for the different classifiers. Here the right number in each cell is the average accuracy and the left number is the average F1 score.

For every classifier the most frequent value imputation strategy yields better results for both accuracy and F1 score. In this test we used 5 folds and 10 repetitions for the cross-validation.

Data Imputation	Random Forest	LinearSVC	Ridge
most frequent	0.957 /0.949	0.962/0.954	0.963 /0.957
nan is category	0.955 /0.947	0.955 /0.947	0.960/0.953

Table 1: Comparison of data imputation for different classifiers for *Congressional Voting* dataset

Amazon Commerce Reviews: This dataset contains no missing values, therefore we don't need to impute any values. The target attribute has 50 different values, which will be encoded as $0, 1, 2, \dots, 49$. All the other values are numerical, therefore we will try different scaling strategies. We will try the most common scalers in the scikit-learn library, namely the MinMaxScaler, the StandardScaler, the RobustScaler, the power

Census Income: This dataset has missing values, but only for categorical values. Those values are labeled with '?'. We will try the same two strategies as for the *Congressional Voting* dataset.

4 Hyperparameters

Now we want to take a closer look at the hyperparameters of our models. In a first step we want to optimize the hyperparameters for the different datasets. In a second step we will analyze how sensitive the models are to changes in the hyperparameters. Unfortunately we will not be able to find the optimum for all hyperparameters, as scikit-learn offers a wide range, so we are going to focus on the most important ones.

For tuning we want to use Bayesian Optimization from the scikit-optimize package. Bayesian Optimization is a probabilistic approach to finding the minimum or maximum. This means that it builds a probabilistic model of the function and uses this model to decide on where to evaluate in the next step. In our case the function is given by the performance measure.

Bayesian Optimization does not only take a function as input but a search space as well, which is why we need to decide on an appropriate range of values for the parameters as well. Moreover, because we chose two different performance measures for each dataset, we will have to do the optimization twice and then compare results to get the overall best.

5 Conclusion

This is the conclusion section. Summarize your findings and suggest future work.

References

- [1] Author, *Title of the Book*, Publisher, Year.
- [2] Author, *Title of the Article*, Journal, Volume, Page numbers, Year.