MACHINE LEARNING

SHEET 1

25.11.2021

10:15 A.M. UNTIL 11:45 A.M. VIA ZOOM

Please prepare the exercises in order to present them in the meeting. Fill-in the exercises you solved in the questionnaire in StudIP until 17.11.2021 4:00 p.m. Note that the exercise meeting is on 25.11.2021!
Join the meeting via this Zoom link:
https://uni-trier.zoom.us/j/88057925689?pwd=bVBIS3NqZ3JQSjZSTWd1alZFSlZ6UT09

TASK 1: DATA ANALYSIS AND VISUALIZATION

For this exercise, we will use the wine dataset that is provided by scikit-learn (https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_wine.html). It is a simple multi-label classification dataset. Load this dataset with the method described under the link above and perform some tasks:

a) Analyze the data by using Python code and answer the following questions by printing to the console:
   - How many examples are contained in this dataset? How many attributes are contained? What are the attribute names? What datatypes do the individual attributes have?
   - Pick out the values of the attribute “ash”: What is the maximum, minimum, median, and mean value for this attribute?
   - Look at the label attribute: How many classes does this attribute have? How many examples are associated to each of the classes?

b) Use matplotlib to visualize the data:
   - Again, take the attribute “ash” and visualize the values as a discrete histogram. Assume a number of 10 bins for the visualization. Display the value range of each bin on the x-axis of the plot.
   - Additionally, visualize the data of the attribute “ash” in a boxplot. What does this say about the data range (short comment in the program code)?

TASK 2: DATA PREPARATION

For this exercise, we will also use the wine dataset (https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_wine.html). Perform the following tasks:

a) Create a new attribute:
   - Create a new attribute “substracted_phenols” by subtracting the values of “nonflavanoid_phenols” from the values of “total_phenols”.
   - Increase all values of the attribute “alcohol” by 1.0.

b) Data augmentation:
   - Shuffle the data.

DEPARTMENT OF BUSINESS INFORMATION SYSTEMS II
PROF. DR. RALPH BERGMANN
UNIVERSITY OF TRIER

- Augment the dataset by creating 20 more examples that are derived from real examples. Therefore, select 20 random examples from the original dataset and apply Gaussian noise (mean = 0 and std. dev. = 1) to those examples. Finally, add the examples to the dataset.

c) Training preparation:
   - Scale the data of the attribute "ash" to the range [0,1] with one of the scalers of scikit-learn. Which scaler did you choose and why?
   - Split the data into a training set (85 % of the examples) and a test set (15 % of the examples).

TASK 3: CLASSIFICATION

In this exercise, we will have a look at classification problems. We will reuse the dataset that was also used in the first exercise.

a) Take the original wine data from the first example:
   - Transform the problem to a binary classification task by combining labels of class 1 and 2. Name the resulting labels *class-0* and *not-class-0*.
   - Split the original data in a training set and a test set with a training ratio of 0.85.
   - After that, train a classifier of your choice with the training data.
   - Finally, compute the accuracy of this classifier with the examples from the test set and compute the confusion matrix.
   - Give a short remark on the performance of this classifier by interpreting the metrics.

b) Take the modified wine data that was created in Exercise 2:
   - Repeat the steps from Exercise 3 a)
   - Compare the performance of both trained models in some short sentences. Was the data transformation in Exercise 2 helpful? Does a comparison make sense?

TASK 4: REGRESSION

Besides classification problems, regression problems are a major use case of machine learning. For this exercise, we use the example dataset of California house prices (https://scikit-learn.org/stable/datasets/real_world.html#california-housing-dataset).

a) First, get familiar with the dataset and prepare it for modeling (by using Python code for the following tasks):
   - Check for missing values.
   - Compute the correlations between the attributes and analyze possible dependencies of the data by visualizing the correlations.

b) Next, suppose you are a housing expert that has additional information to add to the dataset.

- All blocks were houses are on average older than 25 years (*HouseAge > 25*) …
- … and the average number of bedrooms exceeds 3 (*AveBedrms > 3*)
- … are most likely inhabited by Hollywood actors.
- Add a categorical attribute *HOL* to the dataset that expresses this information.
- Encode the new attribute in a suitable way, such that it can be used for regression.

c) Split the data into a training set and a test set (ratio of 0.8 for training cases).
d) Train a linear regressor with the training data.
e) Analyze the performance of the trained regressor on the training set and the test set. Compute the values of Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE) for both sets. How can the MAE values be interpreted? Is the regressor overfitting or underfitting?

TASK 5: LASSO REGRESSION

A common use case of linear regression models is feature selection. Therefore, the feature weights of the trained model are inspected.

a) Load the California housing dataset and split the data into a training and a test set (test ratio of 0.2).
b) Train a Lasso regression model on the training set. Use $\alpha = 1$ as the regularization parameter.
c) Measure the performance of the model on the test set in terms of MAE.
d) Inspect the coefficients of the trained model. Are some equal to zero or close to zero? What can you do with this information?
e) Remove all features with a coefficient close to zero. Retrain the model and evaluate its performance now. Did feature removal yield models with better performance?

HINTS

Make sure that all actions that involve some kind of randomness, e.g., shuffling and random initialization, depend on the same random state for these exercises and for all exercises from upcoming sheets. Otherwise, results are not reproducible! This can be achieved by using the same random seeds in all runs. For instance, scikit-learn has a flag to many functions called *random_state* that is used for this purpose.