Universität Augsburg
Fakultät für Angewandte Informatik

MISIT IT-Infrastrukturen für die Translationale Medizinische Forschung

Prof. Dr. Frank Kramer
Dominik Müller
eMail: dominik.mueller@informatik.uni-augsburg.de

Wintersemester 2019/20

Grundlagen des Programmierens in der biomedizinischen Informatik

Medical Informatics A: Machine Learning in Medicine

Exercise – *Autism Spectrum Disorder (ASD) Classification:*

We are going to analyze a clinical trial about autism spectrum disorder diagnosis by a new screening method. This method consists of 5 screening questions, 2 tests and the inclusion of some personal data (age, gender, ethnicity, …). The aim of this study was to evaluate the accuracy of an autism prediction based on the collected data. After the data acquisition, our goal is to use a machine learning algorithm to evaluate the predictive power of the new screening method. The task is to train an algorithm on the screening data and try to classify patients as ASD or non-ASD from an untouched validation data set. Afterwards, we will compare our predicted classifications with the (ground-truth) autism diagnosis by physicians.

Data:

Filename: Autism_Screening.Data.csv Format: comma-separated-values (CSV)

Submission Server:

Program name:	autism_classification.py	
Arguments:	-i/--input <data_set>	Path to the data set (Autism_Scrrening.Data.csv)
	-a/--algorithm <algorithm>	Selection of the machine learning algorithm. Possible algorithm: [explore, lr, nb, knn, svm, rf, compare]
Example call:	python autism_classification.py -i Autism_Screening.Data.csv -a lr	

Tasks:

(a) **Download Data and Program Base:**
Download the clinical trial data and program base structure from Digicampus.

In the following tasks, we will extend, step by step, the programs base structure with several functions. At the end, the autism classification program will be able to run a preprocessing, training, prediction and evaluation pipeline with several machine learning algorithm.

In order to get familiar with the data set, have a look on the data set description on the slides, again, and also briefly inspect the data set file with a plain text editor.

(b) **Data Processing - File loading:**
Write the function read_dataset() in the data_processing.py file. The function should be able to load the autism data set given a file path and return the data in an appropriate data structure.
It is recommended to use data frames from the Pandas package.

(c) **Data Processing - Data Exploration:**
In a properly performed data analysis, it is required to get an overview of the data. Therefore, we are going to inspect some exemplary variables. This task will not be evaluated in the submission server (does not have a strict output), so feel free to explore the data set by plotting, data peeking, summaries…
Implement the function exploration() in the data_processing.py file. The function should be able to give insights on the following questions given the autism screening data set.

- Print out the head (first 5 lines including column names) of the autism screening data set
- How many patients are diagnosed with ASD and how many are not?
- Identify all possible categories of the column “Entry_Person”
- Identify the age ranges and the number of patients categorized in each age range
- Inspect the column “Age”. Can you find any abnormalities in the data? (ignoring incomplete data)
- How many patients have incomplete data?

The data exploration should be only performed when running the explore mode as algorithm argument.

(d) **Data Processing - Preprocessing:**
Data Preprocessing is a technique that is used to convert the raw data into a clean data set before feeding it to the algorithm. It is an integral step in Machine Learning as the quality of data and the useful information that can be derived from it directly affects the ability of our model to learn.

Implement the preprocessing() function in the data_processing.py file which performs the following methods:
- Remove all patients with incomplete data
- Transform categorical variables by one-hot-encoding
- Transform binary variables (yes/no or female/male) to 0/1.
- Normalize numeric (continuous & discrete) variables by MinMax Scaler (scikit-learn module)

The preprocessing function should return a modified and cleaned data set which is ready for machine learning.

(e) **Data Processing - Split Data into Training and Test Set:**
In order to properly validate predictions of an algorithm, it is required to test the algorithm on data which it has not seen before (not been trained on). Therefore, we are going to split our preprocessed data set into a training and testing set. There are a variety of splitting techniques (e.g. cross-validation), but we will implement a simple 80%-training 20%-testing data set split in this exercise. Additionally, for easier access, we are going to divide the data set into our data (x), which we will use for prediction, and into our classification/result variable (y).

The split_dataset() function in the data_processing.py file should be able to return the following variables:
- train_x: Training set – Data (all variables except the class)
- train_y: Training set – Class
- test_x: Testing set – Data (all variables except the class)
- test_y: Testing set – Class

(f) **Machine Learning Model - Implement Logistic Regression:**
After preprocessing and splitting the data set into machine-learning-ready data sets, we are going to implement the machine learning algorithm. We will start with the Logistic Regression algorithm by using the scikit-learn module.

Implement the Logistic Regression algorithm as a class into the algorithm.py file. The class should consists of the following functions:
- __init__(): Initialization of the Logistic Regression model
- train(): The model will be trained on a given training data and class set
- predict(): A trained model will predict and return the classes for a given testing data set

(g) **Evaluation - Confusion Matrix:**
Implement the confusion_matrix() function in the evaluation.py file which should return the confusion matrix (as four variables) of a given predicted and ground-truth list of classes.
Calculate the confusion matrix and scores by yourself without using any metric functions from e.g. scikit-learn.

(h) **Evaluation - Score Calculation:**
Implement the compute_scores() function in the evaluation.py file. The function should calculate the following scores given the four variables TP, TN, FP, FN:
- Accuracy
- F1
- Precision
- Sensitivity
- Specificity

(i) **Machine Learning Model - Algorithm Library:**
Extend the algorithm library with the following machine learning algorithm by using the scikit-learn module:
- Naive Bayes
- k-Nearest Neighbors
- Support Vector Machine
- Random Forest

Each algorithm should be implemented as a class with an __init__, train and predict function identical to the Logistic Regression class.

(j) **Evaluation - Comparison:**
Implement the comparison mode in which all ML algorithm will be compared with each other based on the same training and testing data set. Print out the accuracy for each algorithm.

Note: This time, it is not necessary to have an identical output as the test cases on the submission server.