Practical session 4: Implementation of the one-year mortality model for palliative care needs assessment

Biomedical Data Science

Ascensión Doñate, Vicent Blanes, Pablo Ferri & Juan Miguel García Gómez

Biomedical Data Science Lab (BDSLab)

ITACA, UPV

Contents

1. Objective

Clinical Decision Support Systems (CDSSs) type IV are systems based on machine learning technology. At the core of these systems there is usually a statistical model that has been trained with data from past episodes. Those models can make predictions over similar new cases. Machine learning models have demonstrated great predictive power, but they also present some downsides such as the need for data preprocessing and the difficulty understanding the reasoning behind the prediction.

During the practical session, we will implement a simplified section of the common pipeline in data science projects that has been developed during the InAdvance project. Including the data loading, basic analysis of the dataset, the data preprocessing (categorical & missing values), the development of the predictive models and a basic evaluation. For the proposed experimentation we will be using the python programming language and some of the common libraries used in data science within the Jupyter environment.

2. Material

- Dataset (inadvance_synth.csv).

3. Evaluation

A Jupyter notebook file (.ipynb) containing the source code and markdown cells containing the responses to the questions.

4. Tasks

4.1. Block I. Data loading and basic description

4.1.1. Objective

Create the Jupyter Notebook and load the data as a pandas DataFrame. Study some of the patients' features.

- Use the inadvance_synth.csv' file.
- You can load the dataset using the pandas' *load_csv* function, remember to specify semicolon (;) as the separator.
- Once the data is loaded as a Dataframe, we will examine the data.

4.1.2. Notes

- Panda's documentation is available online at https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html
- A great alternative to code a data analysis is to use the pandas *profiling library* (https://pandas-profiling.github.io/pandas-profiling/docs/master/index.html). Though its use is not mandatory in this practical session.

- The labels for this problem are included in the columns 'labels' inside the dataset. The positive label (1) means that the patient died within the year since the admission and data gathering. The negative label (0) means that patients did not pass within the 365 days period.

4.1.3. *Questions & exercises*

1. What is the size of the dataframe?
2. What is the mean age?
3. What is the age standard deviation (std)?
4. Which is the variable with the most amount of missing values? Can you list the name of the variables, sorting them by number of missing values?
5. Name which are the categorical variables.
6. Extract the 'label' column to another variable. How many positive cases there are? And negatives?

4.2. Block II. Data preprocessing

4.2.1. *Objective*

Most of the machine learning algorithms implementation are unable to deal with non-numerical data. The most common problems present in datasets are categorical variables, e.g. the ICD codes, and missing values.

- Split the dataset in two: train (80%) and test (20%). Use a seed to allow replication.
- Implement a method to deal with categorical variables.
- Implement a method to deal with missing values using simple techniques such as the imputation by the mean or the median.
- The transformation applied on the training set of the data should be the same applied to the test set.

4.2.2. *Notes*

- The scikit-learn (sklearn) library offers an object with methods to perform simple imputation: https://scikit-learn.org/stable/modules/generated/sklearn.impute.SimpleImputer.html#sklearn.impute.SimpleImputer
- The sklearn's SimpleImputer object could be 'fit' using the train data and the apply it both to train and test using the 'transform' method. Using this approximation, we will ensure that that same transformation is applied to both sets of data.
- One way to deal with categorical variables is to assign a number to each category. This could be done using a simple python dict. Sklearn offers tools for it too. In addition, you could find other libraries/methods to perform the same operation (e.g. https://pypi.org/project/categorical-encoder-pipeline/0.0.9/)

4.2.3. Questions & exercises

1. How many samples have each set after the split?
2. Implement the method to deal with categorical variables. Briefly explain the chosen alternative. What would happen if a variable in the test set contains a category that doesn't exist on the train set? How would you deal with this situation?
3. Implement the method to deal with the missing values. Briefly explain the chosen alternative. What would happen if a variable without missing on the train set appears to have been missing in the test set? How would you deal with them?
4. What if we used the whole dataset to fit the imputer and the categorical variables' method so we don't have to worry about unconsidered values? Explain briefly why this is a terrible idea.

4.3. Block III. Modelling & Evaluation

4.3.1. Objective

With the data split and ready for the training we will select a machine learning method and some metrics to evaluate the predictive power.

- Create the predictor object
- Fit the predictor with the train set
- Predict on the test set
- Quantify the predictive power using standard metrics

4.3.2. Notes

- Any ML method for classification can be used to solve this problem, a nice starting point is the Random Forest, which is fast to train and works very well for tabular data out of the box (https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html?highlight=randomforest#sklearn.ensemble.RandomForestClassifier)
- The area under the ROC curve (AUC ROC) is a very widely used metric to be used in binary classification, especially in medicine. The AUC ROC is calculated a little bit different from other metrics such as the accuracy because it uses the predicted probabilities instead of the predicted labels: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html?highlight=roc%20auc#sklearn.metrics.roc_auc_score
- Most of the sklearn classifiers implement the *predict_proba* method, which returns the probabilities for each class instead of the labels.

4.3.3. Questions & exercises

1. Import the RandomForestClassifier and create the model with the default arguments

2. Fit the model using the train set and the train labels
3. Get the probabilities for the positive class from the test set
4. Use the test labels and the positive class probabilities to calculate the AUC ROC
5. In addition to the AUC ROC, calculate the accuracy, the sensitivity and the specificity of the model
6. (Optional) Plot the ROC curve
7. (Optional) Try different hyperparameters configurations and create a table with the results obtained. You can also use another classifier: Gradient Boosting Machines, Support Vector Machines, Neural networks, etc…