Data Science session: Classification

In this data science session, we are going to focus on supervised machine learning (ML), a major field of AI algorithms, and, more precisely, on classification. Contrary to unsupervised ML, in supervised ML you have one or multiple associated label(s)/class(es) for your data. In classification, the aim is to predict the class of unseen observations (eg. healthy or diseased?).

The goal of the session is to familiarize yourself with model development and validation for classification tasks, including how to prepare your dataset. We are going to work with random forests, but the general pipeline applies to other classifiers too. We will use functions available in the scikit-learn library (<https://scikit-learn.org/>). If you wish, you can use Google Colab. To facilitate your coding, we advise you to use pandas, but this is not essential.

At the end of this session, you should be able to build your own models from scratch for performing different classification tasks using random forests and beyond!

**Basics of classification & Random Forests**

Classification: In classification tasks, you have a set of samples with their corresponding features/variables, also called predictors (model’s input), and you want to find to which class the samples belong to (model output). To this end, we use data whose class we already know (known as labeled data) to train your model so that when the model is faced with previously unseen data, it can predict to which class the unseen data belongs. We can roughly divide the process of development and validation of an AI model for classification into the following steps:

1. Define the problem
2. Prepare the data
   1. Split into training and testing
   2. Exploratory analysis to get an understanding of the data
   3. Pre-process data
      1. Handle missing data
      2. Handle categorical data
      3. Feature scaling for numerical data
         1. Normalization
         2. Standardization
      4. Dimensionality reduction / Feature Selection / Remove correlated features
3. Train – Fit a model
   1. Choose an evaluation metric
   2. Fine-Tune the model’s hyperparameters
4. Test – Make predictions
5. Evaluate the model
   1. Performance metrics beyond accuracy
   2. Interpret your model’s results e.g. by looking at feature importance

During this Data Science Session, due to time limitations, we are going to focus only on the highlighted steps ignoring the ones appearing in gray above, but feel free to explore with the rest of the steps and see how they improve your algorithm after the class. For those interested in further experimentation, some additional questions can be found at the end of the document.

Random forests: Random forests consist of several individual decision trees. Each individual tree makes a prediction, and the final decision of the classifier is given by majority voting, ie. the class that most of the individual trees have voted than the observation belongs to. The key success component of random forests is that ensemble decisions are more accurate than individual ones.

**The problem**

Despite continuous advances in diagnosis and treatment, cardiovascular diseases (CVDs) remain the main cause of mortality worldwide, accounting for about a third of annual deaths. Developing tools to predict CVD complications, such as heart failure, using artificial intelligence could greatly enhance patient outcome and management, but also enable acquiring preventive measures.

In this session, we aim at predicting whether a patient is at risk of heart disease related events given information regarding the patient demographics, measurements, electrocardiogram, and other exams. More precisely, you are provided a hd\_data.csv that contains at each row a potential patient and at each column a feature. The features that you are given are the following:

1. Age: age of the patient [years]
2. Sex: sex of the patient [M: Male, F: Female]
3. ChestPainType: chest pain type [TA: Typical Angina, ATA: Atypical Angina, NAP: Non-Anginal Pain, ASY: Asymptomatic]
4. RestingBP: resting blood pressure [mm Hg]
5. Cholesterol: serum cholesterol [mm/dl]
6. FastingBS: fasting blood sugar [1: if FastingBS > 120 mg/dl, 0: otherwise]
7. RestingECG: resting electrocardiogram results – measures the electrical activity of the heart [Normal: Normal, ST: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV), LVH: showing probable or definite left ventricular hypertrophy by Estes' criteria]
8. MaxHR: maximum heart rate achieved [Numeric value between 60 and 202]
9. ExerciseAngina: exercise-induced angina [Y: Yes, N: No]
10. Oldpeak: oldpeak = ST [Numeric value measured in depression]
11. ST\_Slope: the slope of the peak exercise ST segment [Up: upsloping, Flat: flat, Down: downsloping]
12. **HeartDisease: output class [1: heart disease, 0: Normal]**

In a nutshell, your goal is to predict the output class using the 11 aforementioned features. To this end, you will first need to train your model. Once the model is trained you can apply it to predict the class of the patients and validate its performance.

The dataset was adopted from: <https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction>

**Your task**

You are provided with *predict\_hd.py* and *predict\_hd.ipynb* (in case you want to use Google Colab) with comments and tips to guide you through the steps you have to perform. These comments and tips are there to help you, but you are not obligated to follow them. In brief, you are asked to implement the following pipeline:

1. Read the data and acquire a basic understanding of their properties.
2. Split your dataset into a training and testing set using a balanced number of patients of each class. To this end, you can downsample the majority class.
3. Preprocess the data: encode categorical data, scale numerical features.
4. Train a Random Forest model using the training data.
5. Apply the model to the test set and evaluate its accuracy, AUC, precision and recall.
6. Create a plot with the most important features.
7. Modify the parameters of the Random Forest (eg maximum depth of the trees and number of trees you use). What do you observe?

At the end of the session, you will be also given with a potential solution: *predict\_hd\_solution.py* and *predict\_hd\_solution.ipynb*

**Did you enjoy this session and want to explore more? Here are some additional questions you could try to answer**

* Try to repeat the above steps varying the percentage of data used for training and testing. What changes do you observe? Why?
* What happens if you do not use a balanced number of patients for each category?
* K-fold cross-validation is a popular alternative way for splitting training-validation set and evaluating your algorithms to avoid overfitting. Give it a try!
* You probably noticed that our model has several parameters and their impact in the performance of the model. Did you know that you do not need to manually fine-tune these parameters, but you could estimate optimal parameters automatically? Have a look at grid search algorithm ([sklearn.model\_selection.GridSearchCV](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html))
* Can you repeat the above steps using another classifier and compare the results using a plot?

**Appendix – useful classes & functions:**

[sklearn.model\_selection.train\_test\_split](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html)

[sklearn.preprocessing.OrdinalEncoder](http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OrdinalEncoder.html)

[sklearn.preprocessing.MinMaxScaler](https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html) ([fit](https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html#sklearn.preprocessing.MinMaxScaler.fit), [fit\_transform](https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html#sklearn.preprocessing.MinMaxScaler.fit_transform), [transfrom](https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html#sklearn.preprocessing.MinMaxScaler.transform))

[sklearn.ensemble.RandomForestClassifier](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier)

[pandas.DataFrame.sample](https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.sample.html)

[sklearn.compose.ColumnTransformer](https://scikit-learn.org/stable/modules/generated/sklearn.compose.ColumnTransformer.html) & ([Column transformer with mixed types](https://scikit-learn.org/stable/auto_examples/compose/plot_column_transformer_mixed_types.html))

[sklearn.pipeline.Pipeline](https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html)