Elios BUZO – Laurenti CRISTUREAN – Anthony GIUSTI
Ludovico LENTINI – Michele SPINA

SAPIENZA
UNIVERSITÀ DI ROMA

FDS PROJECT - HEART DISEASE PREDICTION

Report

Professor: Fabio Galasso

Year: 2021-2022

What

Introduction and related work

The goal of this project is to build a *binary* classifier to determine if a patient is at risk for getting a heart disease or not, based on their clinical data. To do this we used a dataset[1] found on Kaggle and we analysed the work-flow and the suggestions present on other similar projects[2]. The dataset used is relatively small, consisting of 918 rows and 12 attributes, both categorical and numeric, such as *chest pain type* and *resting blood pressure*.

Why

Motivation

The application of machine learning techniques in the medical context is very useful if combined with the analysis and diagnosis of doctors. In this way it is possible to help doctors to identify the patients who most need more in-depth clinical examinations, reducing the margins of error and increasing the speed of analysis.

How

Proposed method explained

To achieve our goal we decided to try to apply several different models:

- Logistic regression
- Logistic regression with stochastic gradient ascent
- Gaussian discriminant analysis
- Gaussian Naive Bayes

We explored also this models:

- Logistic regression with gradient ascent
- Bernoulli Naive Bayes

To compare the performance of the various models we used accuracy and area under the ROC curve as metrics.

In order to evaluate the performances of each model we considered the fact of splitting the dataset in two subset. The first, 70% of the total, was used to train the models, then the remaining 30% was used to test them.

Despite the results that we obtained with this approach seemed to be very good we decided to change strategy. Considering the narrow dimension of the dataset, splitting it will result in a loss of precious data. At the same time the randomness of the splitting operation could affect the accuracy of the prediction. So for those reasons we used an approach based on cross-validation, in particular we chose the *k-fold cross-validation*, so in that way we can use all the data without hoping for a lucky splitting.

Where

Exploratory data analysis and pre-processing

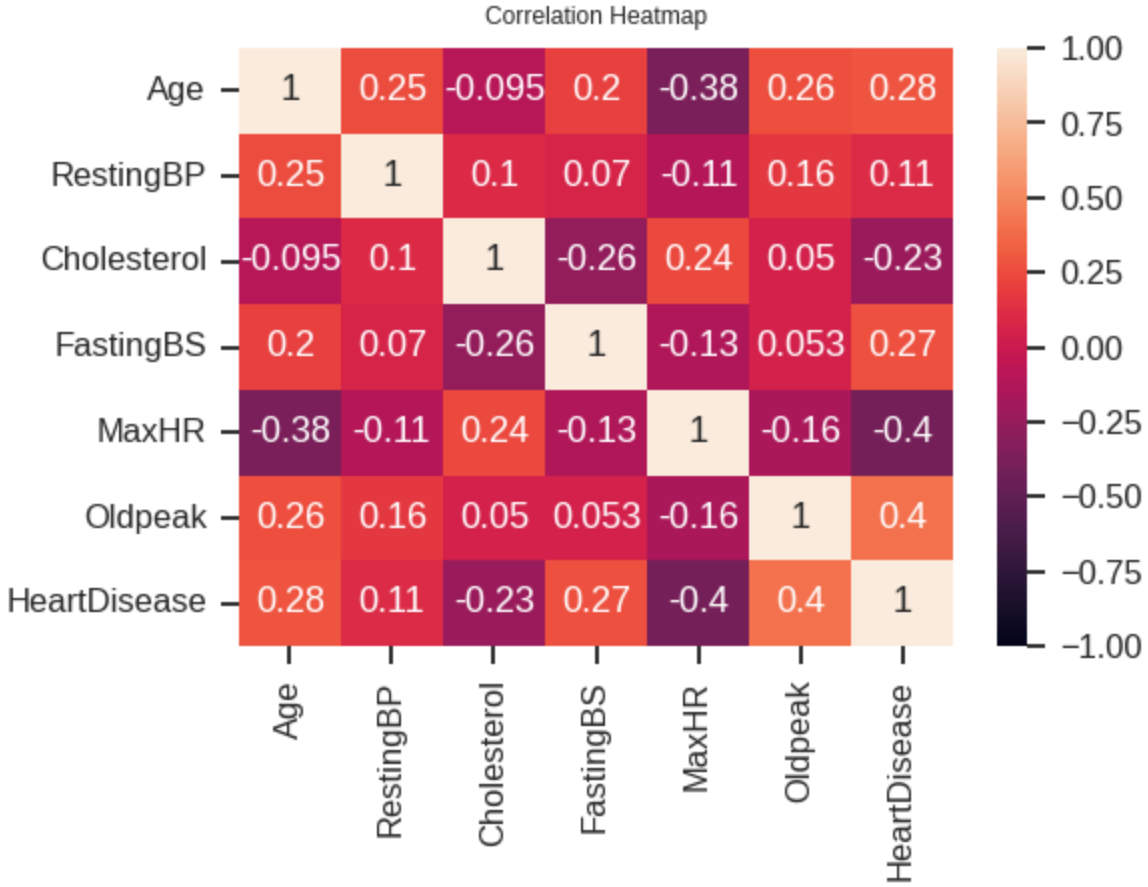
The dataset we are using is taken from kaggle and it's composed of 918 rows each one with 12 attributes. We observed that there is one inconsistent value in the column "RestingBP" and we decided to delete that row because it's only 0.1% of the entire dataset. On the other hand there are 172 inconsistent values in the "Cholesterol" column and we tried several methods to handle this values:

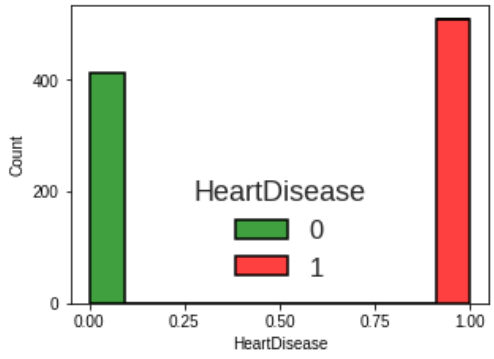
1. Deleting the columns with the inconsistent values;
2. Deleting the rows with the inconsistent values;
3. Filling the inconsistent values with the mean;
4. Filling the inconsistent values with the mean and adding a new column that tells us if the cholesterol value was added or not;
5. Using a linear regression to predict the inconsistent values.

In the end the highest precision method turned out to be the fourth, in fact we observed that the score we calculated for accuracy, AUROC and AP through the logistic regression were always the best, as visible in the pre-processing results section.

Analysing our dataset deeper we also observed that the numerical attributes are normally distributed and there is no strong correlation between our variables which

means there is no strong redundancy and we don't have to delete any attributes.





We also observed that the distribution of the target value, *HeartDisease*, is sufficiently balanced. So we have decided not to apply any oversampling or undersampling on this feature.
It is also fair to say that we have some categorical variables which we manage using one-hot encoding.
In conclusion, we normalized all the values in a range between 0 and 1.

Experimental results

Pre-processing results

In the next table it is possible to observe an example of the result of the different ways to manage the inconstancy. This table, which reflects the common trend of these different methods, shows how the fourth is the most effective.

	Accuracy	AUC	AP
Deleting cols	0.84	0.92	0.92
Deleting rows	0.84	0.92	0.92
Filling w/ mean	0.85	0.92	0.93
Filling w/ mean and adding a col	0.86	0.93	0.95
Linear regression	0.60	0.52	0.59

Splitting

As we said in the previous section, for the first time we splitted the dataset and trained various models. As we can see from the table below the Gaussian Discriminant Analysis produced the better results, even if we achieved good similar results also with the other models.

Model	Accuracy	AUC	AP
Logistic regression	0.869	0.92	0.92
LR with Gradient	0.86	0.92	0.92
GDA	0.86	0.93	0.93
GaussianNB	0.87	0.92	0.92
BernoulliNB	0.86	0.91	0.91

Cross validation and tuning hyperparameters

In this table it is possible to observe the **best results** achieved appling *k-fold cross-validation* and tuning the hyperparameters.

Model	Tuning hyperparameters	Mean accuracy	Mean Roc_Auc
Logistic regression	Max_iter = 1000, C=0.1, Degree = 1	0.872	0.928
Logistic regression with SGA	Max_iter = 500, α = 0.003, Degree = 2	0.874	0.931
GDA	solver = svd	0.870	0.928
GaussianNB	var_smooth = 0.8111	0.870	0.926

Conclusion

From the EDA we expected that the models that are favoured by a *gaussian distribution* of the data have better performance, on the contrary no particularly more effective models have emerged observing final results, even it's fair to say that the logistic regression model is the best one based on the metrics adopted.

The hyperparameter tuning, on the other hand, reflected our initial expectations. We observed that the Logistic regression models are not particularly influenced by max_iter, the convergence is fast. Also, applying the gradient ascent has improved performance for a low enough alpha value.

We observed that for a high value of degree (≥3) performance of Logistic Regression models begins to suffer, this is due to the size of the dataset. In the end, we learned that in this context the logistic regression has better results with a value of the hyperparameter *C* between 0 and 1.

Future works

Drop features

Study in deep

References

1. https://www.kaggle.com/fedesoriano/heart-failure-prediction
2. https://www.kaggle.com/kaanboke/beginner-friendly-catboost-with-optuna
3. https://machinelearningmastery.com/linear-discriminant-analysis-with-python/
4. https://pbpython.com/categorical-encoding.html
5. https://www.analyticsvidhya.com/blog/2021/05/dealing-with-missing-values-in-python-a-complete-guide