A STUDY OF VARIOUS MODELS FOR CARDIOVASCULAR DISEASES PREDICTION

Halvard Bariller, Valentin Gölz
Institut de Mathématiques d'Orsay
Université Paris-Saclay
{halvard.bariller,valentin.golz}@universite-paris-saclay.fr

ABSTRACT

This report presents the results of a model comparison analysis conducted on a dataset [1] related to heart disease diagnosis. The analysis included several feature engineering techniques to enhance the performance of the models. The objective was to predict potential heart failure for patients using classification models. The maximum accuracy achieved was 90%. We conclude that the approach taken in this study can be an effective method to assist medical corpse in heart disease prediction, building on a feasible check-up procedure. A wider collection of data could be a factor for further enhancement of the models.

***K*eywords** Machine learning · Classification · Cardiovascular disease

1 Introduction

Cardiovascular diseases (CVDs) are a leading cause of mortality worldwide, accounting for 31% of all deaths globally. Early detection and management of CVDs is crucial for people at high cardiovascular risk or those already diagnosed with CVDs. In this context, machine learning models can be of significant help. This dataset contains 11 features that can predict the possible occurrence of heart diseases, including age, sex, chest pain type, blood pressure, cholesterol level, fasting blood sugar, electrocardiogram results, maximum heart rate achieved, exercise-induced angina, ST segment slope, and heart disease status. This dataset is created by combining five medical datasets, making it the largest heart diseases dataset available for research purpose. With 918 observations, this dataset can be used for further analysis and research to develop more effective ways of detecting and managing cardiovascular diseases through medical check-up procedure.

2 Understanding and visualizing the dataset

2.1 Data exploration

We start our research by an exploratory analysis, going through the dataset in order to understand the variables and their respective effects.

```
RangeIndex: 918 entries, 0 to 917
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Age             918 non-null    int64
 1   Sex             918 non-null    object
 2   ChestPainType   918 non-null    object
 3   RestingBP       918 non-null    int64
 4   Cholesterol     918 non-null    int64
 5   FastingBS       918 non-null    int64
 6   RestingECG      918 non-null    object
 7   MaxHR           918 non-null    int64
 8   ExerciseAngina  918 non-null    object
 9   Oldpeak         918 non-null    float64
 10  ST_Slope        918 non-null    object
 11  HeartDisease    918 non-null    int64
```

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease
count	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000
mean	53.510893	132.396514	198.799564	0.233115	136.809368	0.887364	0.553377
std	9.432617	18.514154	109.384145	0.423046	25.460334	1.066570	0.497414
min	28.000000	0.000000	0.000000	0.000000	60.000000	-2.600000	0.000000
25%	47.000000	120.000000	173.250000	0.000000	120.000000	0.000000	0.000000
50%	54.000000	130.000000	223.000000	0.000000	138.000000	0.600000	1.000000
75%	60.000000	140.000000	267.000000	0.000000	156.000000	1.500000	1.000000
max	77.000000	200.000000	603.000000	1.000000	202.000000	6.200000	1.000000

Table 1: Data types and summary of continuous variables

First of all, we notice that our dataset is composed of a mix of categorical and continuous variables. This consideration will be essential when preprocessing the data.
In this dataset, we notice that there are no missing values nor duplicates.

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40	M	ATA	140	289	0	Normal	172	N	0.0	Up	0
1	49	F	NAP	160	180	0	Normal	156	N	1.0	Flat	1
2	37	M	ATA	130	283	0	ST	98	N	0.0	Up	0
3	48	F	ASY	138	214	0	Normal	108	Y	1.5	Flat	1
4	54	M	NAP	150	195	0	Normal	122	N	0.0	Up	0

Figure 1: Extract of the dataset

Going further in our variable analysis, we consider the following correlation plot. We can see that the variables *Oldpeak*, *MaxHR* and *Age* seems rather correlated with the variable of interest, *HeartDisease*.

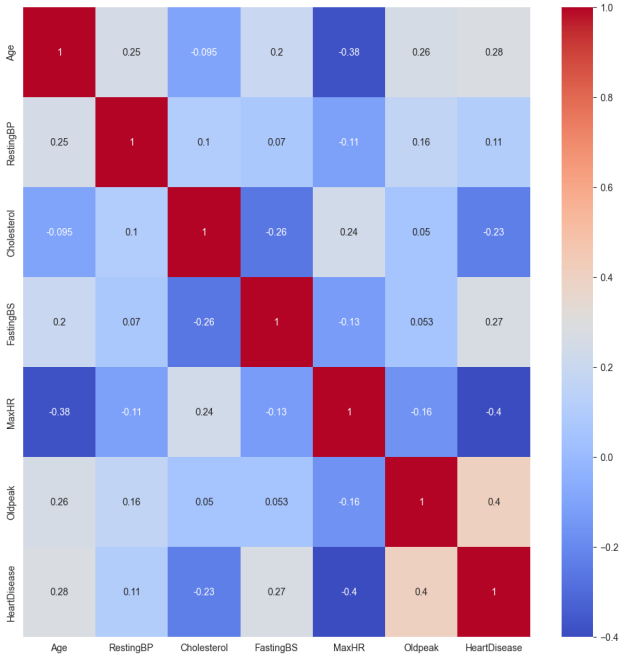


Figure 2: Features correlation matrix

2.2 Categorical variables

As mentioned before, several of our variables (including the target) are categorical features. We thus pursue our exploration by analyzing the repartition of the classes of these variables.
Firstly, we are interested in the balancing of heart diseases classes in our dataset. As we can see in figure 3, patients with a heart disease are slightly more represented than patients with no heart disease. We can consider the dataset as roughly well-balanced, since we have enough samples of each category to create meaningful models for prediction. We won't be needing data augmentation techniques such as *Synthetic Minority Over-sampling* (SMOTE).

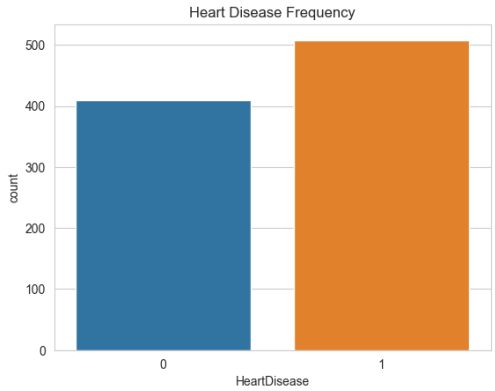


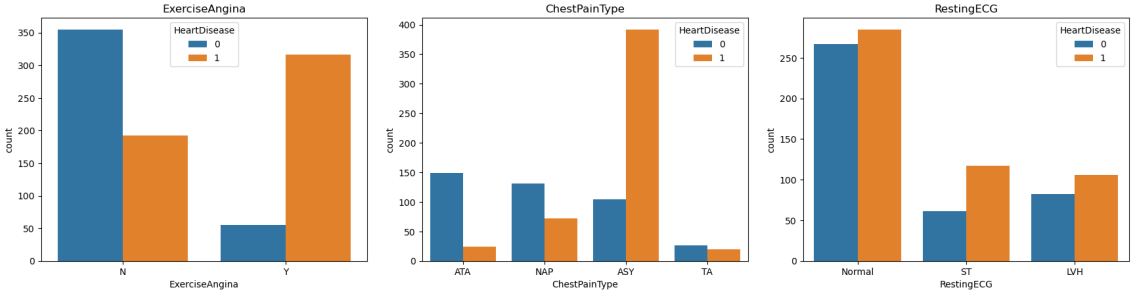
Figure 3: Frequency of each target class in the dataset

The output being well-balanced, we now consider the categorical features. We have the following repartition:

Table 2: Categorical features

Feature	Classes
Sex	2
ChestPainType	4
RestingECG	3
ExerciseAngina	2
ST_Slope	3

We plot the repartition of target classes by modality of each discrete feature.



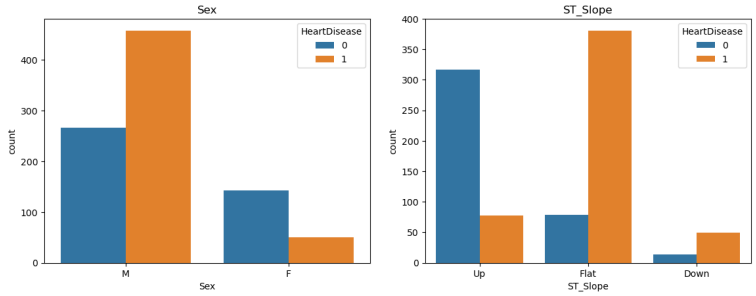


Figure 4: HeartDisease by categorical variable modality

While *HeartDisease* classes seem balanced between certain modalities (such as for the *RestingECG* variable), other modalities are more represented in one or another HeartDisease class - that is the case for the *ST_Slope* variable. However, we cannot conclude just yet on the importance of such modalities in the decision-making process - a flat *ST_Slope* does not necessarily imply a *HeartDisease*, this could also be the consequence of underrepresented categories.

2.3 Continuous variables

After delving into the categorical variables specificities, we briefly represent the target as a function of eah continuous features.

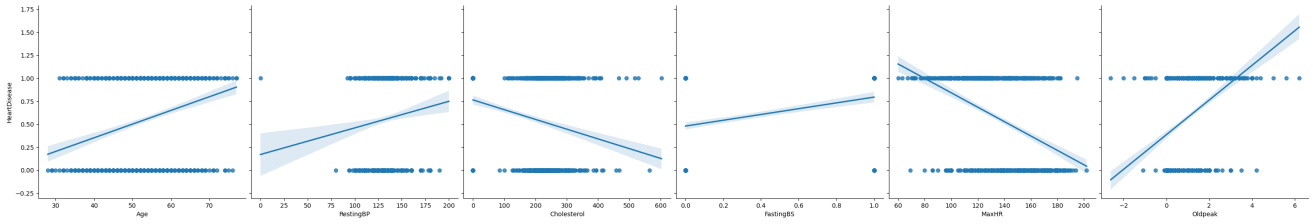


Figure 5: Naive regression on continuous variables

As one could expect, these variables alone won't sufficiently capture the model variance. We can however observe the subtle negative correlation of *MaxHR* and the positive correlation of *Oldpeak* aforementioned.

3 Feature engineering

After getting acquainted with the dataset, we now apply relevant feature engineering techniques in order to enhance the performance of the classifiers that will be built.

3.1 Outliers detection with PCA

The first manipulation we will proceed to is the detection and removal of outliers. To do so, we perform a principal component analysis on the dataset, and perform two different tests : the Hotelling's T-Squared test and an evaluation of the Squared Prediction Error (or Distance to Model). We use the Python package *pca* [2].

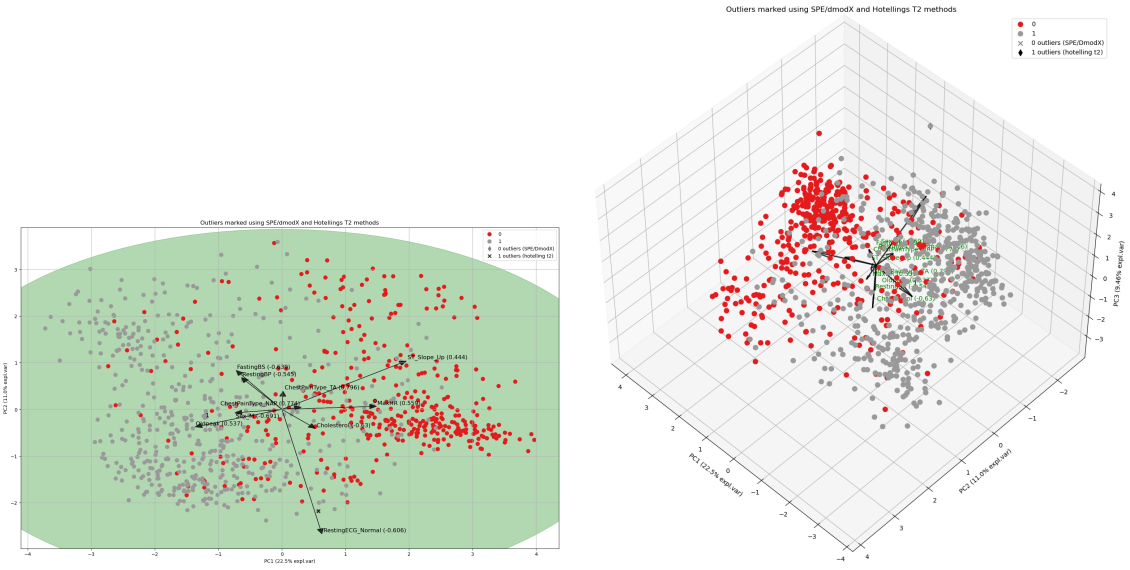


Figure 6: Outliers detection through Hotelling's T2 and SPE/dmodX methods

These methods have detected a single outlier in the dataset, represented by a cross on the 2D and 3D plots on principal components above. This outlier is then removed.

3.2 Feature creation and transformation

To better capture the interactions between features and the information carried by each of them, we proceed to several transformations.
Firstly, we augment our explanatory variables set by creating cubic and squared versions of each of the continuous features so as to smooth our regressors and capture potential polynomial curvature. The degree has been tuned by cross-validation on performing models. Moreover, here we don't perform a stepwise model selection based on dimensionality penalisation (such as the Akaike or Bayesian information criterion), because we will consider the penalty function as an hyperparameter to tune and will evaluate LASSO performance and the relevancy of variable selection when training our models.
Secondly, we re-encode our categorical variables as one-hot numeric arrays.
At this point, we need to mention a specificity of the train-test split method we applied : in order to avoid unbalanced class between the train and the test set (which amounted for 20% of the observations), we applied a stratification parameter on the output variable allowing us to have identical representation of each modality in both train and test (cf. 7).

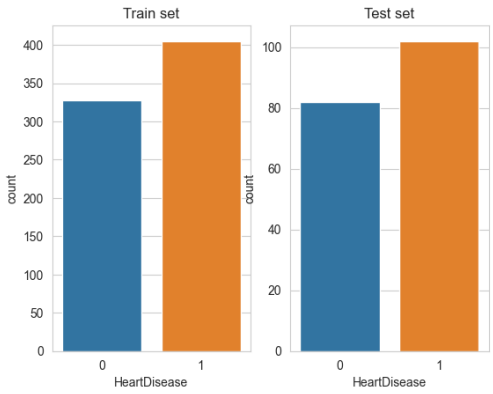


Figure 7: Output variable representation in train and test sets

Once the data separated in two subsets, we proceeded to a Min-Max scaling of our data : this has revealed to be at least as efficient as Standard scaling when performing grid-search for hyperparameter tuning. We thus removed it from the pipeline, fitted it on the training data and applied the same transformation to the test set.
(We note that there is no data leakage either when performing cross-validation on gridsearch).

After applying these transformations, we thus obtain datasets composed of 917 observations in total (outlier removed) and 27 features (9 One-Hot encoded categorical features and 18 continuous features).

4 Benchmarks and model selection

The data having been cautiously prepared for our analysis, we then performed a benchmark of various classification methods on our problem.
We have tried to diversify our models as much as possible, testing basic regression methods as well as well ensemble methods, SVMs, Decision Trees...
For each model, we have performed several GridSearches over refined sets of parameters (including loss functions, arrays of values for specific parameters...), using a 5-folds Leave-One-Out cross-validation method to select our model. We used RandomizedSearches for trees methods.

In order to relevantly take into account false predictions, we used the F1-score as the model selection's metric.

The obtained results are summarized in the below table.

Table 3: Comparison of diverse classifications methods for heart disease prediction

	Score			
Model	Accuracy	Balanced Accuracy	ROC AUC	F1-Score
RandomForestClassifier	0.89	0.89	0.93	0.90
NuSVC	0.86	0.87	0.94	0.89
KNeighborsClassifier	0.89	0.89	0.89	0.89
SVC	0.89	0.88	0.88	0.89
LinearSVC	0.88	0.88	0.95	0.89
GradientBoostingClassifier	0.86	0.86	0.93	0.88
BernoulliNB	0.87	0.87	0.87	0.87
XGBClassifier	0.86	0.87	0.95	0.87
LogisticRegression	0.86	0.86	0.95	0.87
CatBoost	0.86	0.86	0.86	0.87
GaussianNB	0.85	0.86	0.86	0.85
BaggingClassifier	0.85	0.85	0.85	0.85
AdaBoostClassifier	0.84	0.84	0.84	0.84
LGBMClassifier	0.83	0.83	0.83	0.83
DecisionTreeClassifier	0.77	0.77	0.77	0.79

5 Findings and discussion

To elaborate on the obtained model performances, we first notice that apart from a couple of models that seemed to under-perform the benchmark, the scores are quite close from one another. Overall, the model yielding the best performance on the considered metric is the *RandomForestClassifier* : it is the only one with which we reached the 90% performance threshold.
This result was unexpected, since usually other model architectures such as SVMs or boosting methods perform better on this kind of problem. To understand better these results, we compared the training error with the XGBoost model : we achieved a F1-score of 96% on training data with XGBoost, whereas we only have 87% with the Random Forest. Even if boosting methods tend to perform better in general, their main problem is overfitting : it was obvious in this case, we thus concluded that we probably didn't succeed in tuning efficiently the boosting parameters and that it could be an axe for further exploration.

To add details to the scores, we can consider the ROC curve as well as the confusion matrix. The latter allows us to compute two additional metrics : the model obtained a $recall$ of 0.912 and a $precision$ of 0.894.

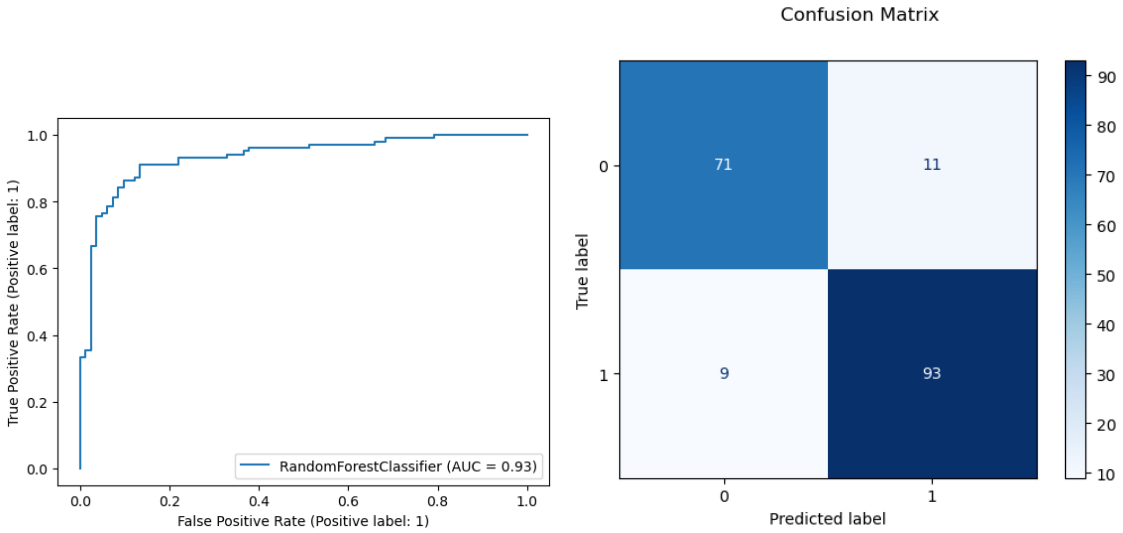


Figure 8: ROC Curve and Confusion Matrix of Random Forest model

It is actually convenient that such a model performed so well since bootstrap methods provide us information on the data through variable importance, helping us to interpret the model performance.
We can thus consider the following figure for explainability with a relatively high degree of confidence.
This has been obtained through our best performing model, a Random Forest composed of 500 trees with a maximum depth of 2 and the Gini index as purity criterion.

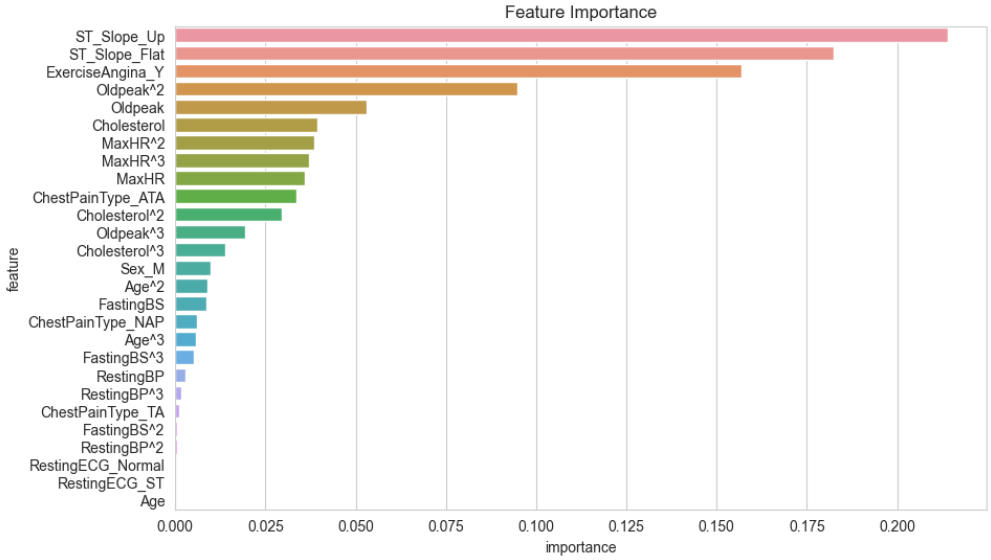


Figure 9: Variable importance retrieved from Random Forest model

The plot shows that *ST_Slope* and *ExerciseAngina* are the two most important features.

We note that as compared to the observation made during the exploratory analysis of categorical variables, we now have more tangible arguments to draw a conclusion on the relevancy of features such as the *ST_Slope* when diagnosing cardiovascular disease.

Another consideration worth being made is that we note the presence of several features created through polynomial transformations amongst the most relevant features : the feature engineering techniques seem to have been fruitful.

6 Conclusion

In this report we presented our results of a model comparison for cardiovascular disease detection.

We first conducted a detailed analysis of the chosen dataset : we detected several types of variables, we were very careful with potential imbalances that could distort the results, and achieved a better understanding of the data. Leveraging on this knowledge, we conducted feature engineering to enhance the models ability to learn on the given task, from outliers detection through PCA to feature creation. We then trained several different models to predict the probability of a heart disease for a given patient. We used exhaustive and randomized grid searches combined with cross-validation for hyperparameters optimization and model selection. After comparing the performance of the models, it turned out that the *RandomForestClassifier* model was the one yielding the best results. However, it is important to note that most of the models we tested were within a 5% difference in prediction performance: this means that different techniques were able to generalize well on the given data, yet each of these models seemed to have difficulty to break the 90% threshold in F1-score.

We would like to put an emphasis on several important points that helped us in building effective machine learning models: check for feature imbalances, categorical variable encoding, scaling and normalizing data to ensure consistency and comparability across features, as well as features creation were essential in this process. Additionally, we note that performing a benchmark was the best way to proceed given the fact that the obtained results were not as we expected : practical experience suggests that it's often best to try various models and select the one that performs the best on the observed data.

To conclude, through variable importance and model prediction, we are satisfied with the findings this research has provided us with : it is clear that gathering the information we used when building a predictor makes sense when performing a diagnosis of such diseases, and machine learning models could help practitioners in their tasks, automating the first analysis on a what could be a standard check-up procedure.

Regarding the models, gathering data at a wider scale could surely help us break the 90% threshold we struggled with, since we only had less than a thousand observations.

References

[1] Fedesoriano. Heart failure prediction dataset, 2021. Retrieved 08.03.2023 from https://www.kaggle.com/fedesoriano/heart-failure-prediction.

[2] Erdogan Taskesen. pca: A Python Package for Principal Component Analysis., 10 2020.