LELEC2870

-

MACHINE LEARNING : REGRESSION, DEEP NETWORKS AND DIMENSIONALITY REDUCTION

Heart failure on the rise in the Smurf society

Report

Members	*Group*
	4
Baptiste PIERLOT - 28082000	
Cyril BOUSMAR - 63401800	

Academic year 2024-2025

December 6, 2024

Contents

1 Introduction

This report details the steps and methods implemented to help Smurfs identify the causes of heart failures using machine learning techniques. The process began with the implementation of a simple linear model (Section 2), followed by an exploration of non-linear models without incorporating heart images (Section 3). Finally, experiments were conducted with models capable of incorporating data from heart images (Section 4). The report also presents hypotheses on the potential causes of these heart failures (Section 5).

2 Linear model (Part 1)

2.1 Previous pipeline analysis

Arbitrarily choosing the number of features leads here to under-performing system. But *Pearson* correlation performs very well in this case and is a good choice. Then, the lack of expressiveness from the profession feature impacts negatively the precision, and `LabelEncoder` transformer should only be use on target and not input data as prescribed by the library. Finally, there is a risk of over-fitting the model as the data is not splitted into training, validation, and test sets. Moreover, not trying to use another model despite poor results is dangerous as patients predictions will be off.

2.2 Methods and results

Python library `scikit-learn` is used extensively for implementation. Previous LR model from (Brainy Smurf et al., 2024) and its *Root Mean Squared Error (RMSE)* metric at 0.08 are used as benchmark for all improvement methods. *KFold* and *RepeatedKFold* (10 repeats) cross-validators (5 folds := 80-20% training/validation sets in our case) are used in feature selection & model selection for robust results, to prevent over-fitting, and to allow hyper-parameters fine tuning.

Pre-processing

Features can be separated into 3 categories: normally distributed, ordinal, and professions. Professions contains qualitative information and thus is transformed into multiple binary quantitative features through *One-Hot Encoding*. *Ordinal Encoding* is applied to ordinal features to express them quantitatively. Finally, `StandardScaler` is applied to standardize all the data. Other scalers and norms were considered but had no impact difference.
The *Z-score* method is used to measure the impact of outliers (the lower, the better). Cholesterol, relatively impacting feature (cf. Figure 1a) had a big outlier (index 992). Switching its value to the median had a very bad result but removing it improved the RMSE from 0.0777 to 0.0776 and cleaned the distribution. *Z-score* dropped from 3.0% to 2.0%, and benchmark RMSE, dropped to 0.0776.

Feature selection

Multiple methods where considered as possible candidate to provide the best set of features: *Correlation Filtering (CF)*, *Mutual Information*, *Maximum Relevance and Minimum Redundancy*, *Forward Wrapper*, *Backward Wrapper*, n most important features from *Decision Tree*, and *Recursive Feature Elimination*. Alongside RMSE, *Mean Absolute Error* ensures the model is not subject to RMSE bias due to outliers, and R^2 regression score function evaluates model's performance. Note that some will only make sense in part 2 as they are non-linear selection processes.
Measurements in Table 2 show that CF is the best selection method. It drops RMSE to 0.0774, and improves the model expressiveness. Selecting the 11^{th} most correlated features has the best result.

Model selection

Candidate linear models are those provided by `sklearn`, for convenience: *LR*, *Lasso*, *Ridge*, *ElasticNet* and *Poisson Regressor*. The best identified model's performance is thereafter validated on the test set. `GridSearchCV` is used to optimize the hyper-parameters of regularized models (*Ridge*, *Lasso*, and *ElasticNet*). This approach enables the testing of different values of the regularization parameters (alpha, l1_ratio, etc.), guaranteeing an optimal compromise between bias and variance.

The *Ridge* model with its default hyper-parameters performed best and achieved an RMSE of 0.0770.

2.3 Discussion

While this part of the study achieved some improvement in heart failure prediction by addressing previous shortcomings, the modest gain of 3.75% in accuracy (RMSE reduction from 0.08 to 0.0770), and the presence of negative predictive values show that it remains suboptimal.
Investigating nonlinear feature transformations like *Polynomial Feature Expansion*, or advanced models like *Random Forest* or *Neural Networks* is worth trying to achieve more robust and actionable predictions.

3 Non-linear model without integration of images (Part 2)

3.1 Previous pipeline analysis

Data pre-processing
The observations made in the previous section remain valid, particularly with regard to the use of *Label Encoder* for the occupation variable. The adoption of an *Ordinal Encoder* for the other three categorical variables is judicious, as it preserves the order of importance of these categories.

Feature selection
Pearson correlation gives good results in this case. However, exploring complementary methods such as *Recursive Feature Elimination* or *Mutual Information* could improve performance. As such, incorporating the number of variables as a hyper-parameter is a relevant approach.

Model selection
Setting hyper-parameters is a key step in optimizing model performance and is therefore good practice. However, limiting ourselves to `SVR` is restrictive. Testing other models, such as *Random Forest* or *Decision Trees*, could offer potentially better alternatives, more favorable to our case.

3.2 Methods

For the *Fisher score-based* method, the `fisher_score` function from the `skfeature` library was used. The model provided by the smurfs serves as a reference to measure the improvement of the tested model and its RMSE value on the test set, which was 0.073. For all the models tested, a *Grid Search* with a *Cross-Validation* of 5 folds was applied for robust results, to avoid over-fitting, and to allow fine-tuning of the hyper-parameters.

3.2.1 Data pre-processing

Section 2.2 stays valid since no other technique has been used to increase the quality of the model.

3.2.2 Feature selection

First of all, the correlation between the various features was examined in order to identify and remove any redundancies. From the analysis, no feature was found to correlate strongly enough with another to warrant its removal.

Correlation with target
A method was implemented to identify the k features with the highest correlation with the target variable, where k is a hyper-parameter. From the graph below (Figure 1a), the interval $k = [9, 12]$ was chosen. This decision is based on the observation that the correlation begins to stabilize from the ninth feature. Up to 12 features were included to assess whether the additional 3, although more weakly correlated, could still enhance model performance.

SelectKBest method (Figure 1b)

The `scikit-learn SelectKBest` method was used, which selects the k most important features based on a score function. The `mutual_info_regression` function was chosen, as it calculates the reduction in entropy resulting from the transformation of a dataset.

Fisher's score (Figure 1c)

A method was implemented that calls `scikit-learn`'s `fisher_score` function and selects the k features with the highest scores. The *Fisher score* assesses a feature's ability to distinguish between different classes or levels of a target. It does so by measuring the ratio between inter-class variance (differences between class means) and intra-class variance (dispersion within classes). A feature with a high Fisher score is therefore a variable that effectively discriminates between target classes.

Recursive feature elimination

The RFECV (Recursive Feature Elimination with Cross-Validation) method was applied using a RandomForestRegressor as estimator. The goal of RFE is to select features by recursively considering smaller and smaller sets of features. The estimator is trained on the initial set of features and give a score of importance for each feature. Then, the least important features are pruned from the current set of features and so on.

Random Forest Importance (Figure 1d)

Trees naturally rank features according to their contribution to reducing node impurity (measured via *Gini impurity*). Taking advantage of this intrinsic ability of *Random Forests* to evaluate the importance of features, the most important features, located in the top nodes of the trees, were selected by pruning the trees below a certain threshold.

3.2.3 Model selection

Having implemented various feature selection techniques, it was essential to develop a model that would outperform Smurfs. As mentioned in the discussion of their pipeline, it is crucial to explore several models to identify the one best suited to our task. Following models are considered for testing: `SVR`, *Decision Tree Regressor*, *Random Forest Regressor* and finally *Multi-Layer Perceptron (MLP) Regressor*. For each of these models, a *Grid Search* is carried to optimize their respective hyper-parameters, and *Cross-Validation* with 5 folds is performed. In parallel, all the feature selection methods described above are evaluated to determine which offers the best results, with the aim of achieving optimal performance.

SVR model

First, smurfs' `SVR` model is used to see if data pre-processing and the use of another feature selection technique could improve performance. The range of hyper-parameters was also extended, as shown in Table 4.

The best parameters are {'C': 0.1, 'epsilon': 0.05, 'kernel': 'rbf'} with correlation with the target as feature selection method. The number of selected features are 12. The mean RMSE on the test set for this model is 0.07372.

Decision Tree Regressor model

Next, the *Decision Tree Regressor* is chosen as it is a simple yet powerful model that can capture non-linear relationships in the data and provides a solid baseline for regression tasks. To have the best performance, *Grid Search* is used to systematically explore different values of the hyper-parameters (Table 5).

The best parameters are {'criterion': 'poisson', 'max_depth': 5, 'max_features': 'sqrt', 'min_samples_leaf': 8, 'min_samples_split': 20, 'splitter': 'best'} with *Random Forest Importance* as feature selection method. The number of selected features are 10. The mean RMSE on the test set for this model is 0.08425.

Random Forest Regressor

As the results of the models were not very conclusive, *Random Forest Regressor* model was tested, as it combines the predictions of multiple decision trees to improve accuracy, reduce over-fitting, and capture complex relationships within the data, making it a robust choice for regression tasks. Considered hyper-parameters are in Table 6.

The best parameters are {'criterion': 'friedman_mse', 'max_depth': 10, 'min_samples_leaf': 4, 'min_samples_split': 2, 'n_estimators': 100} with correlation with the target as feature selection method. The number of selected features are 12. The mean RMSE on the test set for this model is 0.07268.

Multi-Layer Perceptron (MLP) Regressor

Finally *MLP Regressor* is tested, which is a powerful tool for modeling complex relationships between features and the target variable. Its strength is in his ability to learn patterns through a series of connected layers, thus this model can handle non-linearity. *Grid Search* is used (Table 7) to fine-tuned hyper-parameters.

The best parameters are {'activation': 'relu', 'alpha': 0.1, 'hidden_layer_sizes': (100, 100, 100), 'learning_rate': 'constant', 'solver': 'adam'} with correlation with the target as feature selection method. The number of selected features are 9. The mean RMSE on the test set for this model is 0.06913.

3.3 Results and discussions

The best model at present is therefore a *MLP Regressor* using feature selection based on the correlation between tabular features and target, taking the 9 most correlated and with the following hyper-parameters: {'activation': 'relu', 'alpha': 0.1, 'hidden_layer_sizes': (100, 100, 100), 'learning_rate': 'constant', 'solver': 'adam'}. It gives a RMSE on the test set of 0.06913 (5.56% of improvement). Interestingly, the feature selection method that gave the best results in the majority of cases was the simple selection of the k features most correlated with the target.

4 Non-linear model with integration of images (Part 3)

4.1 Comments on technical report nº3

The approach in the technical report is very interesting, and few process steps can be highlighted to further explore possibilities. As the images are small in size, using pixels directly performs quite well, with a 27.78% improvement (from 0.0619 to 0.0499 RMSE) compared to the best model from section 3. Nonetheless, the following questions come to mind. What is the impact of cropping images? Should feature selection be made globally or separately on tabular and image features? Can an other model better extract features from images? What is the impact of data augmentation on this small dataset? Would another model be better to predict heart failures based on the complete set of features?

4.2 Method

Third technical report prediction model, `SVR`, and its space search are used as a benchmark. Each method is tested one after the other, in report order, and knowledge gained from the previous step is used to test the next one. Therefore, the benchmark model is the best performing model from previous task. The RMSE given by the prediction model is the comparison criteria.
To train and evaluate, *Grid Search* is always used on the prediction model, and *Cross Validation* (5 folds := slitting 80-20% training/validation sets) is used both for prediction and feature extraction model. Epochs are carefully selected to prevent Neural Networks over-fitting (10 proved best everytime).

4.2.1 Cropping

Flattening image extraction method is used to compare accuracy and the running time of prediction model for both pre-processing methods: with and without cropping.

4.2.2 Global and separate feature selection

Prediciton model RMSE for both, *global selection* and *separate selection* is compared, using *Flattening* image extractor method. The chosen selection mechanism is CF, the best found in section 3. The model considers a broad range of number of features: {2,4,8,16,32,64,128,256}.

4.2.3 Images features extractor models

Taking the small size of images and dataset into account, lightweight following images features extractor, from three different categories, are considered: image descriptor models (*Harris corner*, *Canny Edge Detection (CED)*, *Oriented FAST and Rotated BRIEF (ORB)*, *Histogram of Oriented Gradients (HOB)*, *Scale-Invariant Feature Transform (SIFT)*), pre-trained and fine-tuned on our dataset CNN models (*ResNet18*, *SqueezeNet*, *small MobileNetV3*), and custom made CNN models (with simple convolution layers, `CNN` (Figure 2), atrous convolution layers, `CNNATrous` (Figure 3), and double convolution layers, `CNNDoubleConv` (Figure 4)).

4.2.4 Data augmentation

Because the dataset is very limited, data augmentation could potentially improve the predictions. Randomized considered transformations are: horizontal and vertical flips, rotations (up to 15 degrees), and adjustments in brightness, contrast, and saturation.

4.2.5 Prediction model and fine-tuning

With cropping, data augmentation, bests feature selection, and best extractor model, selecting a prediction model and fine-tuning its hyper-parameters is the last step. Both `SVR` and `MLPRegressor` are selected, as the first one was used as benchmark and the second one is the best performing of Section 3.
`SVR` model's hyper-parameters are: the kernel (k), the number of image features selected (p), the regularization parameter (C), and ϵ (Tables 8, 9 for space search).
`MLPRegressor` model's hyper-parameters are: hidden layers configuration (hlc), activation function (A), α, type of learning rate (tlr), number of image features selected (p) (Tables 10, 11 for space search).

4.3 Results and discussion

4.3.1 Cropping

For both methods, $RMSE = 0.0499$ and running time is around 1m32s. In other words, there is no incentive in keeping full scale images. This is probably due to the fact that image size is very small to begin with (28x28 px, reduced to 18x18 px), and most importantly, that the cropped area possess very few usefull data. This is also comforted by the fact that most important pixels are not part of the cropped area and are always taken into account in the feature selection.

4.3.2 Global and separate feature selection

As showed in Table 3, separating the feature selection process for tabular and image data has much better result. Not only it has a better RMSE (0.598% improvement), but the model also considers much fewer features to make its decision. This means that, even though many image features have a higher score than tabular features (whatever the feature selection process), tabular features still hold very important information while only but a part of higher scoring image features are needed for good image representation. In fact, it will be seen in Section 4.3.5 that only 21 image extracted features are needed for the best prediction model.

4.3.3 Images features extractor models

From all extractor models, custom made `CNNDoubleConv` performs best, with 9.62% RMSE improvement over the benchmark model.

Descriptors	Flattening	Harris	CED	HOG	OBF	SIFT
RMSE	0.0499	0.0730	0.0585	0.0515	0.0730	0.728
CNNs	`ResNet18`	`SqueezeNet`	`MobileNetv3`	`CNN`	`CNNATrous`	`CNNDoubleConv`
RMSE	0.0580	0.0541	0.0612	0.0458	0.0481	**0.0451**

Table 1: RMSE of the benchmark prediction model `SVR`, based on each extractor model. Flattening is the benchmark extractor from the technical report n^{o}3.

4.3.4 Data augmentation

With data augmentation, `CNNDoubleConv` improves its RSME by 3.1% (from 0.0451 to **0.0437**) which proves that it has consequent effect over this small dataset.

4.3.5 Prediction model and fine-tuning

The `SVR` model RMSE improved by 2.52% (from 0.0437 to **0.0426**) with fine-tuned hyper-parameters: $\{k$=rbf, p=21, C=0.2, ϵ=0.012$\}$.
Surprisingly, the `MLPRegressor` model under-performs and only scores (0.0436) as much as the `SVR` model without fine-tuning from Section 4.3.4. Its best hyper-parameters are: $\{hlc$=(150,150,150), A=relu, α=0.11, tlr=constant, p=21$\}$.

5 Hypotheses on the causes of heart failure (Part 4)

Figure 5 highlights the features that appear to have the strongest impact on the risk of heart disease among Smurfs. From these plots, it is evident that Smurfs with higher blood pressure and elevated cholesterol levels are at significantly greater risk. Additionally, high consumption of smurfin donuts is associated with an increased risk of heart problems, whereas low consumption of sarsaparilla leaves also correlates with a heightened risk. This is confirmed by the *t-SNE* visualization conducted over the data in Figure 7, and the *KMeans* clusters representatives in Figure 8.

On Figure 6, the first 3 hearts are healthy and the last 3 are at-risk of failure. We can see that healthy hearts have a larger area with high entropy than sick hearts. At-risk smurfs therefore seem to have a heart with more centralized high-entropy zones. Most important anatomical part seams to be: aorta, aortical valve, septum, and contour membrane.

6 Conclusion

This report introduces a model 46.75% more efficient (from 0.08 to 0.0426 RMSE) in predicting heart failures for the Smurf population (Figure 9), compared to the first presented by (Brainy Smurf et al.). Its pipeline process (Figure 10) is carefully considered, whose key elements are: data augmentation, custom made CNN for image feature extraction, correct tabular features transformation, separate selection for tabular and image features, and fine-tuning of a *SVM* predictor model.

Although `CNNDoubleConv` custom made model (Figure 4) is proven best for features extraction and many more models were considered, unexplored possibility is the use of specific medical imaging pre-trained models that should at least perform better than the other pre-trained considered models. Up-scaling images had no real effect (if not worse), so having higher resolution dataset seams essential in this case. Image size is too small for descriptor models.

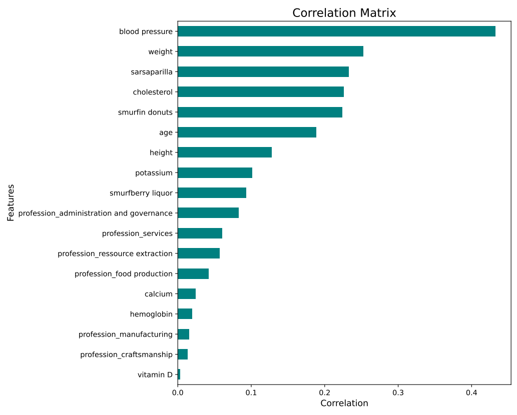
7 Appendix

Selection	RMSE	MAE	R^2
Correlation Filtering	**0.077385**	**0.056731**	**0.315622**
Mutual Information	0.077626	0.056826	0.311351
Maximum Relevance and Minimum Redundancy	0.077655	0.056826	0.310824
Forward Wrapper	0.077596	0.057068	0.311880
Backward Wrapper	0.077596	0.057068	0.311880
Decision Tree	0.077627	0.056825	0.311337
Recursive Feature Elimination	0.077558	0.056857	0.312549

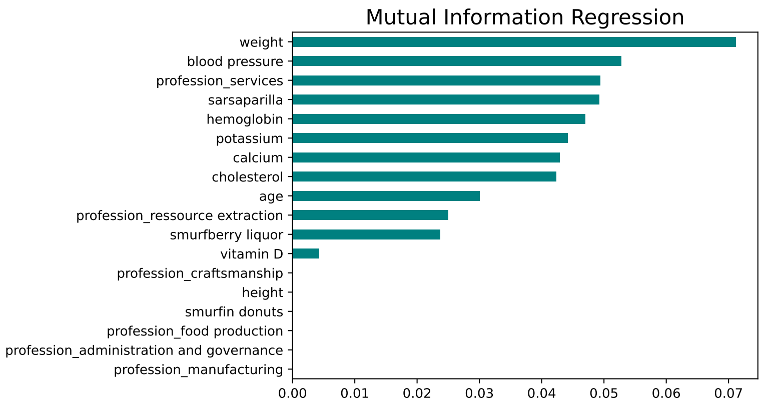
Table 2: Feature selection metrics results

Feature selection	**RMSE**	**Tabular features**	**Image features**	**Total features**
Global	0.0502	5	59	64
Separate	**0.0499**	9	16	25

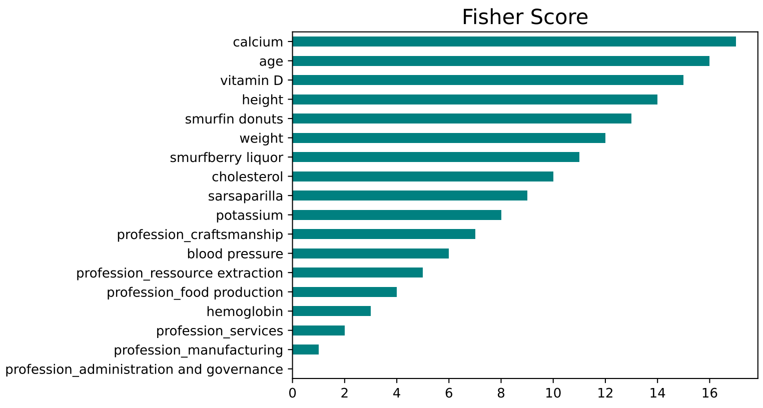
Table 3: Metrics comparison between global and separate feature selection.



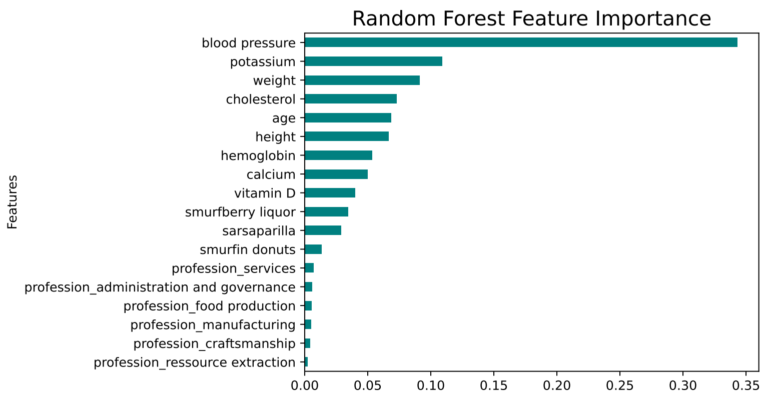
(a) Correlation between features and target



(b) Mutual Information Regression



(c) Fisher score



(d) Random Forest Importance

Figure 1: Features importance after transformation and standardization.

7.1 Search Space for model (Part 2)

You can find here the search space for the the GridSearch of differents models tested:

SVR model :

Hyper-parameter	**Allowed values**
kernel	`poly`, `rbf`, `sigmoid`
C	0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10
ϵ	0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10

Table 4: Search space for the grid search of SVR model.

Decision Tree Regressor model :

Hyper-parameter	**Allowed values**
criterion	`friedman_mse`, `poisson`, `absolute_error`, `squared_error`
splitter	`best`, `random`
max_depth	2, 5, 10, 20, 50, 100
$min\_samples\_split$	2, 5, 10, 20, 50, 100
$min\_samples\_leaf$	1, 2, 4, 8, 16, 32, 64
max_features	`sqrt`, `log2`

Table 5: Search space for the grid search of DecisionTreeRegressor model.

Random Forest Regressor :

Hyper-parameter	**Allowed values**
$n\_estimators$	100, 150, 200, 250, 300
criterion	`friedman_mse`, `squared_error`
max_depth	5, 10, 15, 20, 30, 50
$min\_samples\_split$	2, 5
$min\_samples\_leaf$	1, 2, 4

Table 6: Search space for the grid search of RandomForestRegressor model.

Multi-Layer Perceptron (MLP) Regressor :

Hyper-parameter	**Allowed values**
$hidden\_layer\_sizes$	(100), (100, 100), (100, 100, 100)
activation	`identity`, `logisitc`, `tanh`, `relu`
solver	`sgd`, `adam`
$alpha$	0.001, 0.01, 0.1
learning_rate	`constant`, `invscaling`, `adaptive`

Table 7: Search space for the grid search of MLPRegressor model.

7.2 Search space for the grid search (Part 3)

SVR model :

Hyper-parameter	**Allowed values**
k	`poly`, `rbf`, `sigmoid`
C	0.01, 0.05, 0.1, 0.5, 1
ϵ	0.01, 0.05, 0.1, 0.5, 1
p	2,4,8,16,32,64,128, 256

Table 8: First search space for the grid search of `SVR` prediction model. Those parameters are used to give a general idea of the best fit.

Hyper-parameter	**Allowed values**
k	`rbf`
C	0.18, 0.9, 0.2
ϵ	0.0115, 0.012, 0.0125
p	19,20,21,22,23

Table 9: Second search space for the grid search of `SVR` prediction model. Using best recorded parameters from Tab. 8, to specialize the parameters a find the bests to feed the model.

Multi-Layer Perceptron (MLP) Regressor :

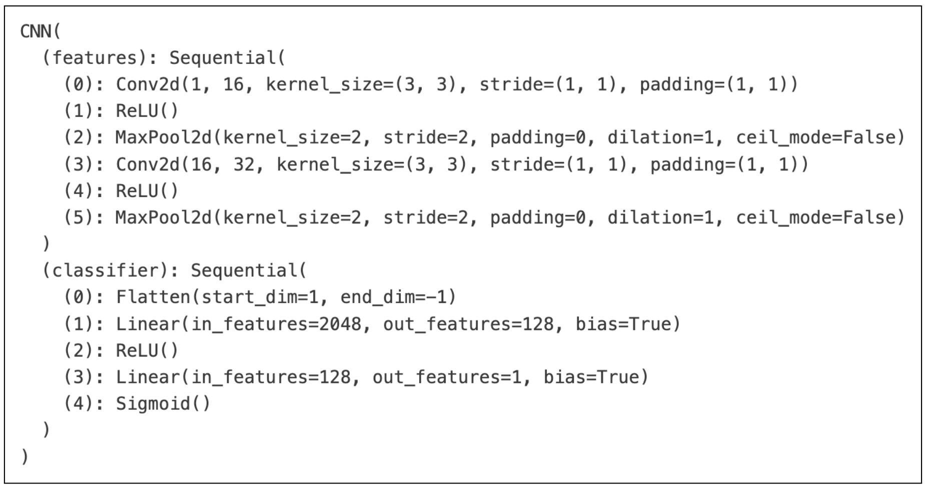
Hyper-parameter	**Allowed values**
hlc	(100, 100, 100), (150, 150, 150), (200, 200, 200)
A	`identity`, `logisitc`, `tanh`, `relu`
α	0.001, 0.01, 0.1
tlr	`constant`, `invscaling`, `adaptive`

Table 10: First search space for the grid search of `MLPRegressor` prediction model. Those parameters are used to give a general idea of the best fit.

Hyper-parameter	**Allowed values**
hlc	(150, 150, 150)
A	`relu`
alpha	0.11, 0.15, 0.2
tlr	`constant`

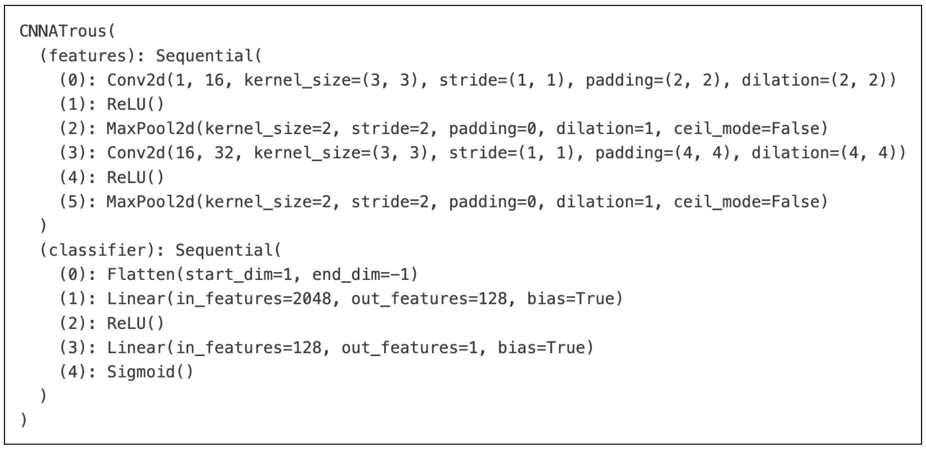
Table 11: Second search space for the grid search of `MLPRegressor` prediction model. Using best recorded parameters from Tab. 10, to specialize the parameters a find the bests to feed the model.

7.3 Custom made CNNs architectures (Part 3)



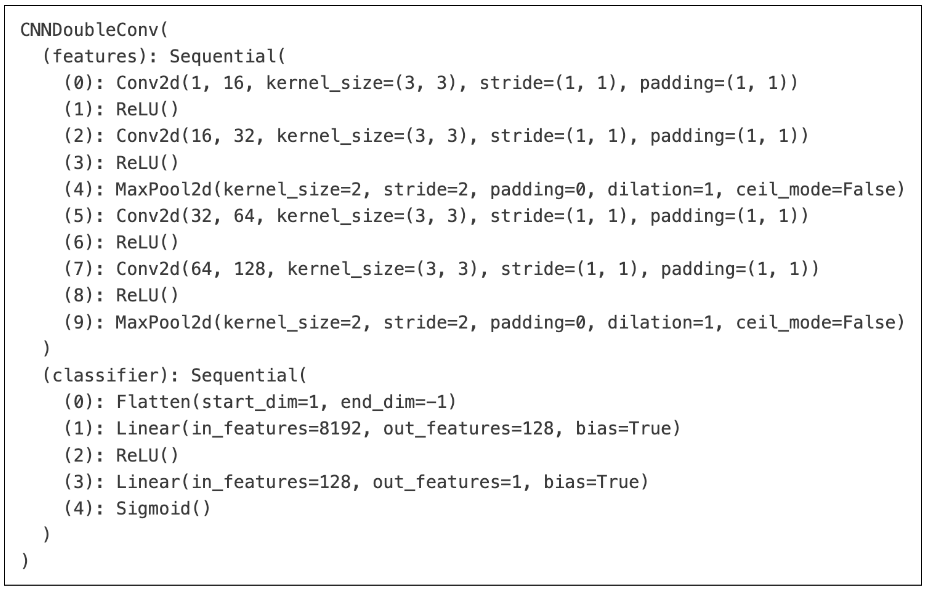
```
CNN(
  (features): Sequential(
    (0): Conv2d(1, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (4): ReLU()
    (5): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (classifier): Sequential(
    (0): Flatten(start_dim=1, end_dim=-1)
    (1): Linear(in_features=2048, out_features=128, bias=True)
    (2): ReLU()
    (3): Linear(in_features=128, out_features=1, bias=True)
    (4): Sigmoid()
  )
)
```

Figure 2: Architecture of simple `CNN`



```
CNNATrous(
  (features): Sequential(
    (0): Conv2d(1, 16, kernel_size=(3, 3), stride=(1, 1), padding=(2, 2), dilation=(2, 2))
    (1): ReLU()
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(4, 4), dilation=(4, 4))
    (4): ReLU()
    (5): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (classifier): Sequential(
    (0): Flatten(start_dim=1, end_dim=-1)
    (1): Linear(in_features=2048, out_features=128, bias=True)
    (2): ReLU()
    (3): Linear(in_features=128, out_features=1, bias=True)
    (4): Sigmoid()
  )
)
```

Figure 3: Architecture of `CNNATrous`



```
CNNDoubleConv(
  (features): Sequential(
    (0): Conv2d(1, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU()
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (5): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (6): ReLU()
    (7): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (8): ReLU()
    (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (classifier): Sequential(
    (0): Flatten(start_dim=1, end_dim=-1)
    (1): Linear(in_features=8192, out_features=128, bias=True)
    (2): ReLU()
    (3): Linear(in_features=128, out_features=1, bias=True)
    (4): Sigmoid()
  )
)
```

Figure 4: Architecture of `CNNDoubleConv`

7.4 Results figures

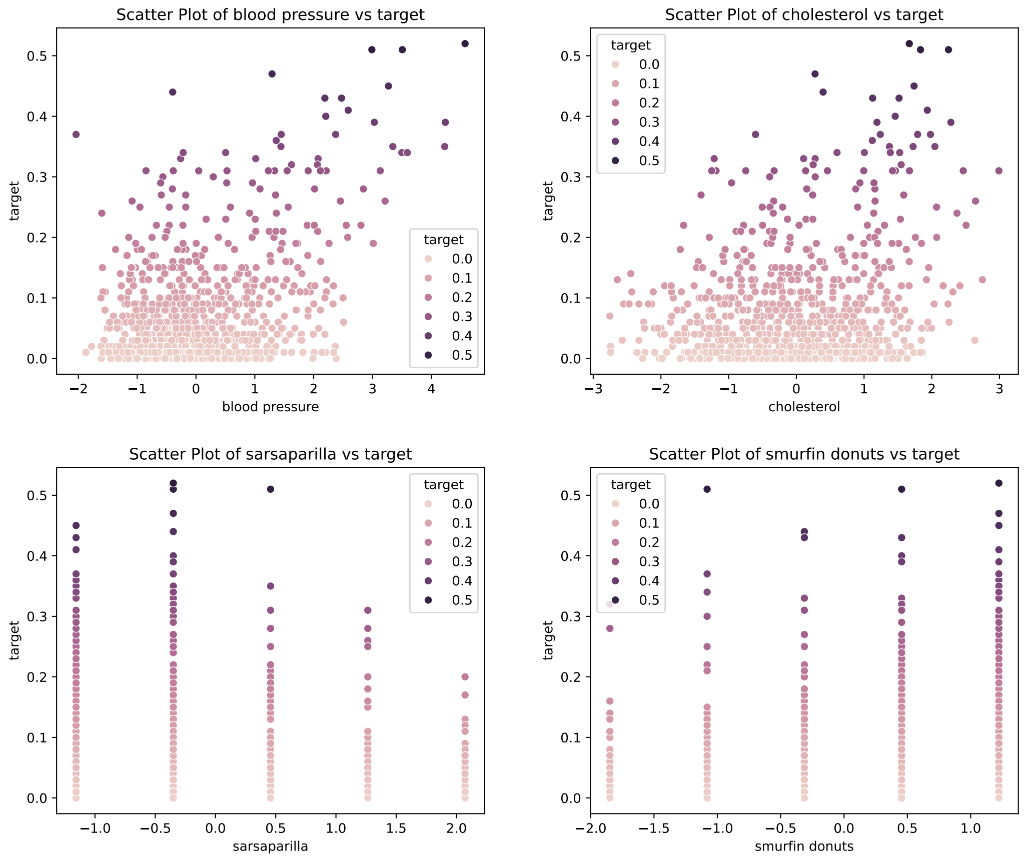


Figure 5: Analysis of the impact of several features on the risk of heart failure.

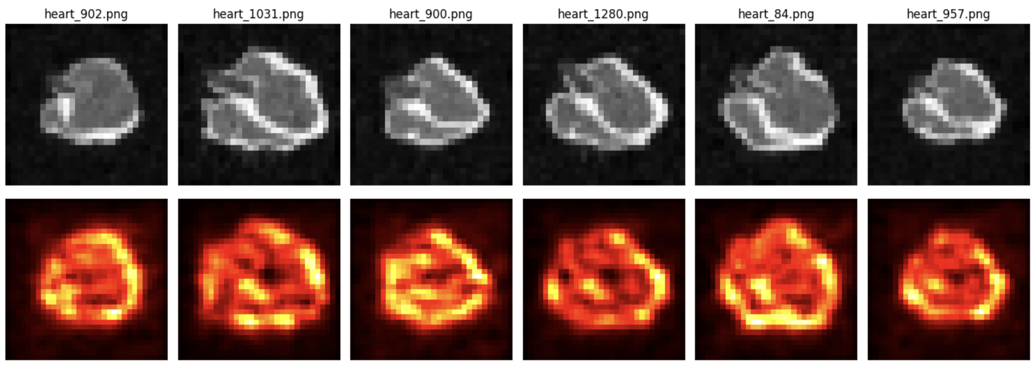


Figure 6: Occlusion sensitivity heat-map of smurfs' hearts showing pixels entropy for heart failure prediction compared to the original image.

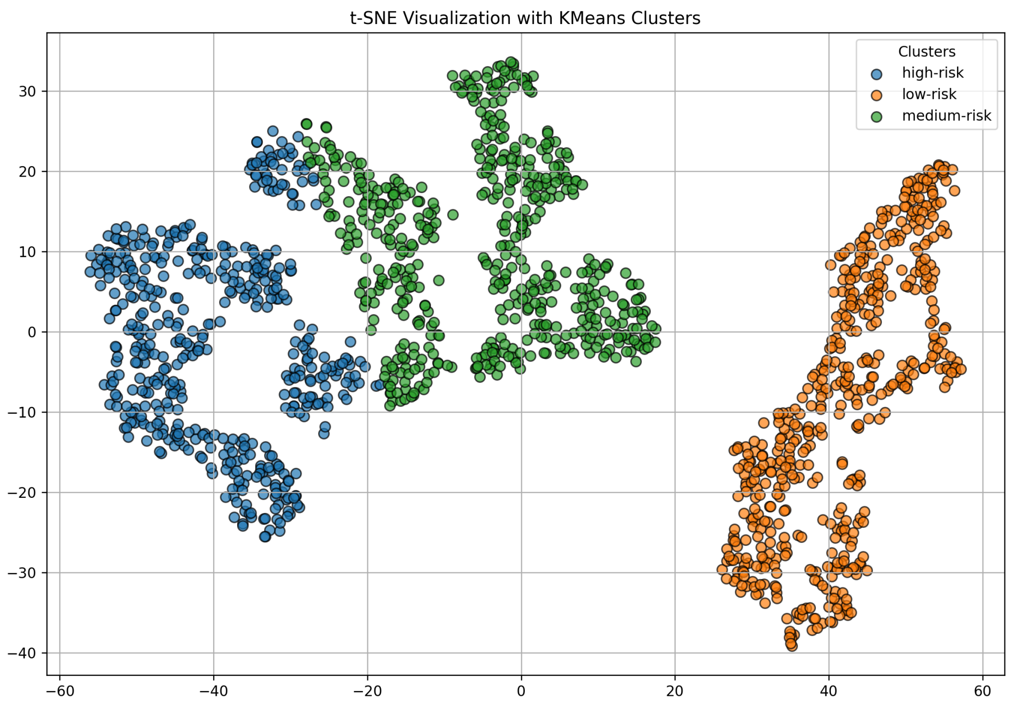
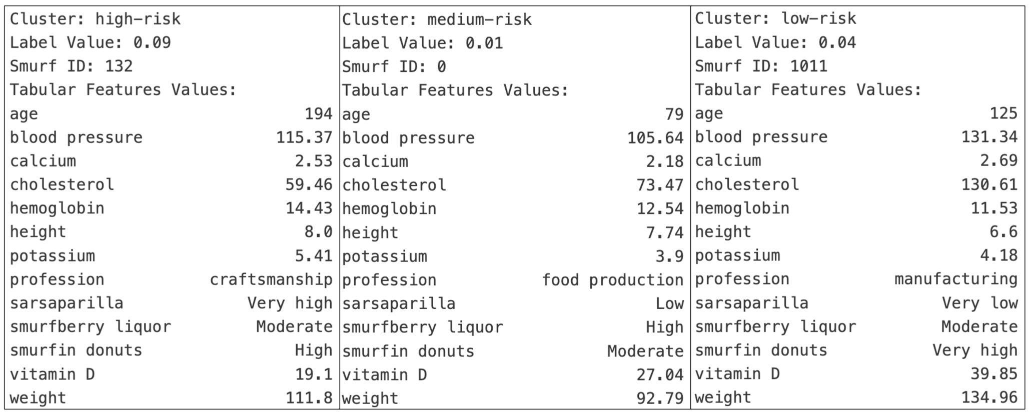


Figure 7: t-SNE visualization with KMeans clusterization to analyse smurfs population in the perspective of heart failures. Three groups are identified, corresponding to the risk level of heart failures.



Cluster: high-risk		Cluster: medium-risk		Cluster: low-risk	
Label Value: 0.09		Label Value: 0.01		Label Value: 0.04	
Smurf ID: 132		Smurf ID: 0		Smurf ID: 1011	
Tabular Features Values:		Tabular Features Values:		Tabular Features Values:	
age	194	age	79	age	125
blood pressure	115.37	blood pressure	105.64	blood pressure	131.34
calcium	2.53	calcium	2.18	calcium	2.69
cholesterol	59.46	cholesterol	73.47	cholesterol	130.61
hemoglobin	14.43	hemoglobin	12.54	hemoglobin	11.53
height	8.0	height	7.74	height	6.6
potassium	5.41	potassium	3.9	potassium	4.18
profession	craftsmanship	profession	food production	profession	manufacturing
sarsaparilla	Very high	sarsaparilla	Low	sarsaparilla	Very low
smurfberry liquor	Moderate	smurfberry liquor	High	smurfberry liquor	Moderate
smurfin donuts	High	smurfin donuts	Moderate	smurfin donuts	Very high
vitamin D	19.1	vitamin D	27.04	vitamin D	39.85
weight	111.8	weight	92.79	weight	134.96

Figure 8: Representing profiles from the three identified clusters in Fig. 7 with tabular data values.

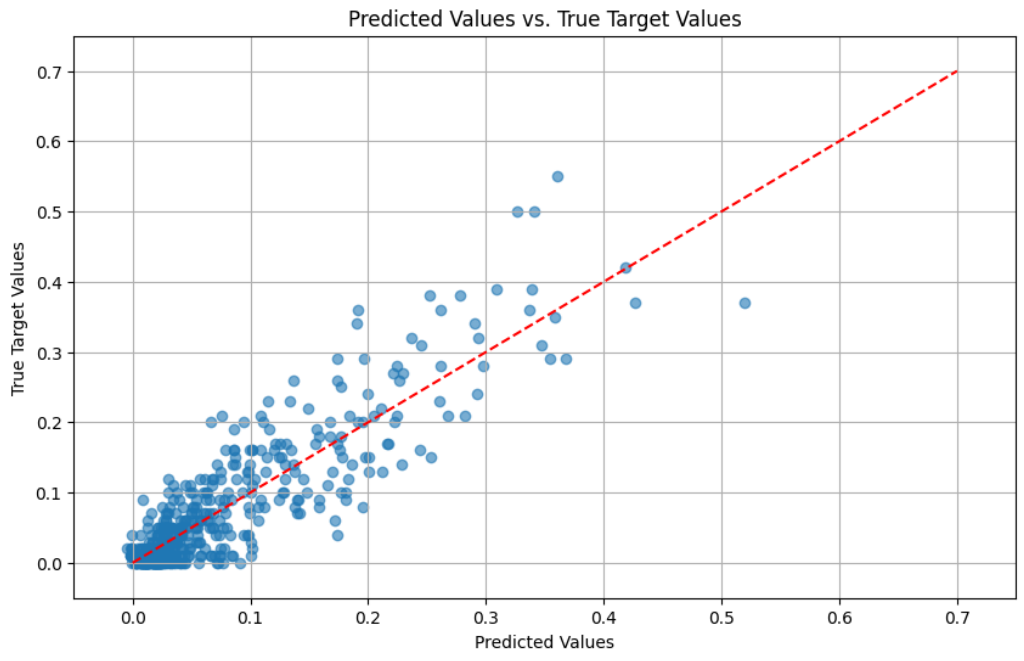


Figure 9: Comparisons of the new model predictions and to the ground truth values using the test set.

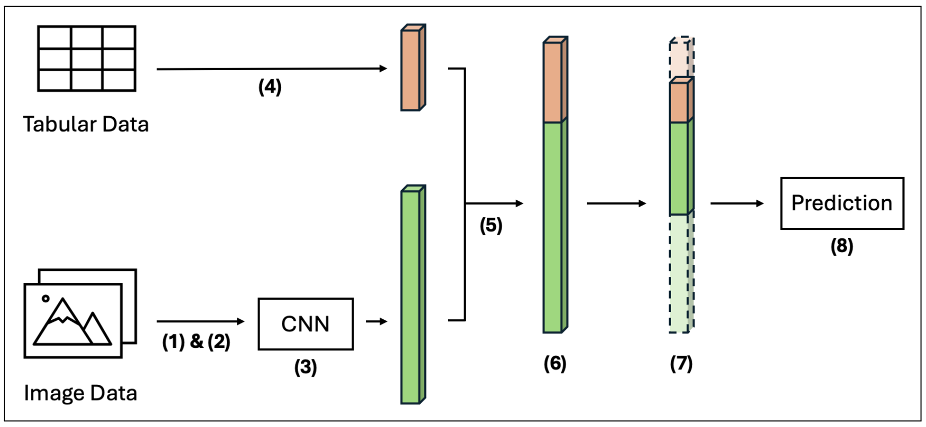


Figure 10: Complete process pipeline. `1)` cropping, `2)` data augmentation, `3)` feature extraction (w/ *Double Conv CNN*), `4)` tabular transformation, `5)` tabular and image data merger, `6)` standardization, `7)` feature selection, and `8)` predictions (w/ `SVR`).