BRFSS - Heart attack prediction

A machine learning approach

Written by Alice Anna Maria Brunazzi[a], Alessandro Della Beffa[b], Daniele Lepre[c]

[a]Bachelor's degree in Banking and Finance, University of Milano-Bicocca
[b]Bachelor's degree in Mathematics, University of Milano
[c]Bachelor's degree in Banking and Finance, University of Milano-Bicocca

Introduction

Every 40 seconds, someone in the United States has a heart attack. Each year, about 805,000 people in the United States have a heart attack. Of these, 605,000 are first-time attacks and 200,000 are in people who have already had an attack. In addition, about 1 in 5 heart attacks are silent: the damage occurs but the person is unaware of it.

Given the high incidence and serious consequences of heart attacks, it is essential to develop effective predictive tools to identify their risk and thus improve preventive interventions. In this context, machine learning can provide powerful and efficient solutions.

For our study, we used data from the 2011 Behavioral Risk Factor Surveillance System (BRFSS [1]). The BRFSS is the leading telephone survey system in the United States that collects state data on health risk behaviours, chronic health conditions, and use of preventive services. Established in 1984 with 15 states, BRFSS now collects data in all 50 states, the District of Columbia, and three U.S. territories. It conducts more than 400,000 interviews with adults annually, making it the largest continuous health survey system in the world.

The project was carried out using the KNIME software, including official extensions such as H2O and Python. This document and the accompanying workflow describe the common steps in a typical machine learning procedure [2]: reading data, data wrangling, exploration, transformation and analysis (learning, prediction and scoring). Given the health-medical nature of the data set, special attention has been paid to the issue of unbalanced data.

The data, which is publicly available in ASCII format [3], has been converted into a CSV file using Python and consist of 506,467 records and 450 variables. These variables, which are described in detail in the codebook [4], all take numeric, possibly symbolic, values and are divided into record identification variables, questionnaire variables, and computed variables [5]. The questionnaire variables include the binary target variable `CVDINFR4` (ever diagnosed with heart attack).

Using this data, our goal is to develop a machine learning model that can predict the risk of heart attack, helping to improve the prevention and management of this critical condition.

1. Reading data

The import process in the workflow began by accessing the `LLCP2011.csv` data set and making an initial selection of variables. This first step removed variables that were constructionally dependent on others (i.e. `PREGNANT` and `SEX`). In addition, variables conventionally coded as numbers (`MRACE`) were discarded in favour of the corresponding more meaningful calculated variables (`_RACE_G`). The vast majority of calculated variables can be identified by the presence of an underscore at the beginning of their name.

Fifty-nine variables have been retained, including the target `CVDINFR4` (ever diagnosed with heart attack), have been retained. These variables are mostly the same as those available in the BRFSS Prevalence and Trends Tool [6] for 2011.

During the import phase, some variables underwent a conversion process to ensure the use of the most appropriate data type: number (double) for continuous variables, number (integer) for ordinal variables, string for nominal variables.

2. Data wrangling

Uncertainty is taken into account for the majority of variables by defining the values '*Don't know/Not sure*' and '*Refused*' as arbitrary values, typically 7 and 9; these values were converted into missing values.

The variables CHECKUP1 (length of time since last routine checkup), EDUCA (education level) and INCOME2 (income level), originally categorical, were recoded as numbers in order to preserve the level of information provided. Specifically, EDUCA was transformed into an integer number, corresponding to the average number of years of study in the US education system [7]. Other transformations took place via the node (Column Expressions), such as the conversion of the unit of measurement (e.g. from Imperial to International), using the KNIME Expressions extension.

The final step of the import phase was to filter out the 2465 records where the value of the target variable CVDINFR4 was missing. The resulting dataset was then divided into a training (80%) and a test (20%) set using stratified sampling with respect to CVDINFR4. To ensure reproducibility, a random seed value of 1618 was chosen.

3. Exploration

The Data Explorer node, from the KNIME JavaScript Views (Labs) extension provides an accurate univariate analysis of the training set, distinguishing between numeric and categorical variables. From this point on, the ordinal variables (GENHLTH, USENOW3 and SEATBELT) were treated as numeric.

The generated histograms showed unbalanced distributions with asymmetric long tails: figure 1 shows the distribution of the variable PHYSHLTH for illustrative purposes, as most numerical variables show a similar pattern.

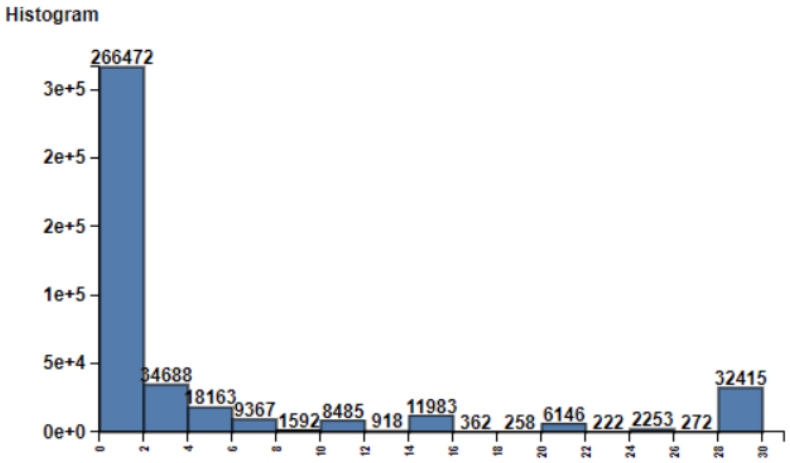

Figure 1: physical health distribution

The only visible exceptions to this trend were the anthropometric measures such as AGE, HTIN4 (computed height in inches) and WTKG3 (computed weight in kilograms).

Bar charts showed categorical variables, mostly binary and clinical in nature, that were moderately imbalanced. According to some classifications [8], the target variable CVDINFR4 also falls into the moderate imbalance case, where the minority class is 1%–20% of the data set.

All variables (except SEX and, of course, CVDINFR4) had missing values. The number of missing values varied widely between variables, ranging from 331 for USENOW3 to 129,069 for PADUR1_ (minutes of first activity).

4. Transformation

Based on the knowledge of the shape and quality of the data obtained so far, the process proceeded with the evaluation and application of common preprocessing operations: outlier treatment, missing value treatment, normalization of numerical variables, encoding of nominal variables, and feature selection. The treatment of outliers was prioritized over all other operations in order to prevent outliers from distorting subsequent analyses [9], including the analysis of missing values.

4.1. Treatment of numerical outliers

Outliers, especially in medical dataset, require attentive and cautious analysis, due to their highly informative content.

The box plot generated during the data visualization phase confirmed highly asymmetric distributions, so the approach taken to eliminate the outliers was the isolation forest model, provided by the KNIME H2O Machine Learning Integration extension. The standard parameters of the model were left unchanged.

The algorithm assigns to each record the average depth of the decision trees constructed to isolate it from other records: the lower the average depth, the greater the degree of anomaly associated with the record. For more details, see the article How to Detect Outliers). Records with an average depth lesser than q_α are considered outliers, where q_α is the quantile representing the threshold of the assumed proportion α of outliers in the data. Given the difficulty of detecting this proportion in the box plots, the number of true outliers was roughly estimated by the total number of records that appeared isolated in the histograms. The estimated proportion, rounded to the first significant digit, was $\alpha = 0.0001$, and corresponding $q_\alpha = 5.17$. Filtering the average depth below the threshold q_α removed 45 records from the training set, and 12 records from the test set.

4.2. Missing values

As noted above, some of the variables had a number of missing values in the order of hundreds or thousands, which was less than 2.5% of the training set. For these variables, record deletion or any of the basic univariate techniques for replacing missing values would be appropriate and sufficient, partly because of to the large size of the data set. However, for variables with tens or hundreds of thousands of missing values (up to 32%), either strategy would artificially alter the statistics and variability of the original data. For replacing missing values in numerical variables, more sophisticated alternative solutions are suitable, such as k-nearest neighbours, which replaces the missing values in each record with the average of the corresponding values in the nearest k records (usually according to Euclidean distance).

Unfortunately, k-nearest neighbours—at least in the KNIME implementation—is not suitable for more than a few tens of thousands of instances. However, thanks to the KNIME Python Integration extension, it is possible to access a Python environment and its library *scikit-learn*, which implements several multivariate missing value replacement strategies. The `IterativeImputer` model was chosen, which alternately models each variable with missing values as a function of the others. As it is generally a good idea to normalize numerical variables before training a model, we used `MinMaxScaler` from *scikit-learn* to rescale the variables to the interval $[0, 1]$ before `IterativeImputer`.

For nominal variables, missing values were treated as true categories, with the absence of information also being considered as information. For this purpose, the necessary encoding of nominal variables was anticipated. One-hot encoding, despite the drawbacks of increasing dimensionality and collinearity, remains the standard for encoding nominal variables (e.g. it is the option adopted in the AutoML component of KNIME), especially when they have few categories. By default, *scikit-learn*'s `OneHotEncoder` transformer treats missing values as separate categories, which is why it was preferred to KNIME's native One to Many node.

A single nominal variable, `CTYCODE1` (county code), with 256 categories, was excluded from the encoding process. When the number of categories seems too large to justify one-hot encoding, a viable alternative is target encoding [10], which transforms the variable from nominal to numeric by exploiting the conditional mean, with respect to the target variable. `CTYCODE1` was encoded separately using `TargetEncoder` from *scikit-learn*. he details of the transformation depend on the type of the target variable (binary in this case) and are discussed in the `TargetEncoder` documentation.

All transformations in this subsection were performed on the training set and then applied to the test set using the Python Script node of the KNIME extension mentioned above.

4.3. Normalization

As mentioned above, some machine learning models benefit from normalization to make numerical variables comparable. Rescaling to the interval $[0, 1]$ and standardizing to mean 0 and variance 1 (Z-score normalization) are common and alternative operations, both of which are sensitive to outliers [11]. Rescaling is more robust than standardization when dealing with small values of variance, and may therefore be preferable when the distribution of the variable is not known a priori. By this time, the numerical variables had already been sufficiently rescaled. However, neither the previously missing and now replaced values nor the variable `CTYCODE1`—previously nominal and now numeric—were guaranteed to always take values in $[0, 1]$. To ensure consistency, rescaling to $[0, 1]$ was applied again to the training set and then to the test set.

4.4. Feature selection

After one-hot coding, the number of columns in the training set and test sets increased from 59 to 142. According to the usual rules of thumb [12], this increase is still not detrimental to dimensionality (the data set retains a ratio of more than 3500 records per column). However, the spatial dimension of the data set almost tripled, mainly due to columns that that are likely to have almost perfect collinearity, resulting in redundancy. The negative impact of collinearity and correlation on statistical models (especially in regressions, on the significance of the coefficients) is well known [13], so to simplify the models, a feature selection was performed.

Apart from an initial filter to remove all columns with zero variance, the actual feature selection was multivariate. Recursive feature selection and elimination, implemented natively in KNIME, proved too computationally expensive. To measure the importance of features, we used the impurity indices computed naturally by tree-based estimators [14], in particular by random forests [15].

The random forest model from the H2O AutoML Learner node, which automatically implements parameter optimization, was applied. Given the imbalance in the data, the area under the precision-recall curve was set as the metric for selecting the best model, while leaving the other default settings unchanged. From the resulting model specification (nodes 31 and 32), the feature importance values were extracted, rescaled to sum to 1, and sorted as shown in figure 2.

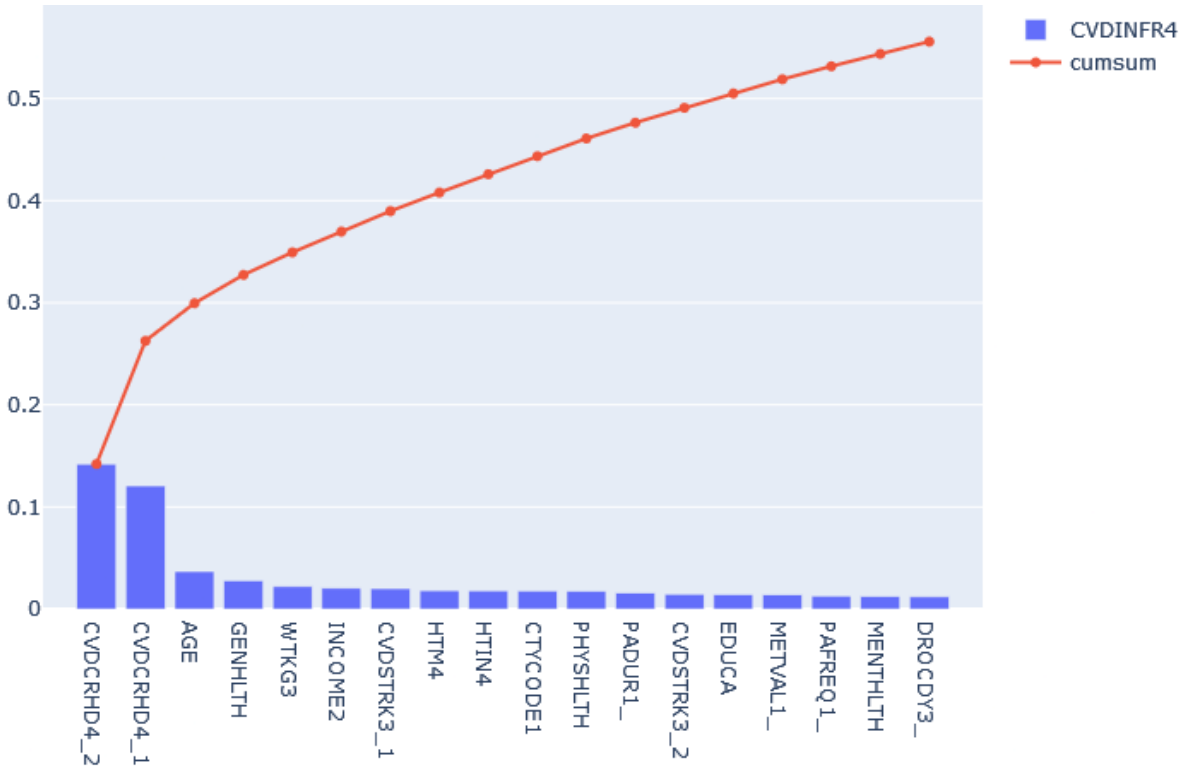

Figure 2: importance of the first 18 variables and corresponding cumulative function

The cumulative curve in figure 2 shows that `CVDCRHD4_2` and `CVDCRHD4_1` (ever diagnosed with angina [16] or coronary heart disease [17]) were by far more important than all other variables. These conditions are known risk factors for heart attack, so their importance is understandable and expected. Other variables contribute to depict the general health status of the subjects and may increase the risk of heart disease in the presence of comorbid conditions (such as stroke history, `CVDSTRK3_1`, and physical activity levels, `PAFREQ_1`).

There was no obvious change in the slope and therefore no reasonable cut-off between the first most important variables and the rest. A heuristic criterion for the number of factors in a scree plot [18], valid not only for the eigenvalues of a PCA, but in general for any ordered sum-1 n-tuple, suggested setting the cut-off at the minimum point of the curve in figure 3. This minimum was found—excluding the first three variables—in `CHCKIDNY_2`(ever told you have kidney disease?), and represented the neutral compromise between the number of variables and their importance.

Finally, from the selected variables, those with correlation indices greater than 0.75 were filtered out, thus

removing the variables most at risk of high collinearity (mostly binarized variables from the one-hot encoding).

Overall, the changes made to the training set by feature selection and then applied to the test set, reduced the number of independent variables to 56.

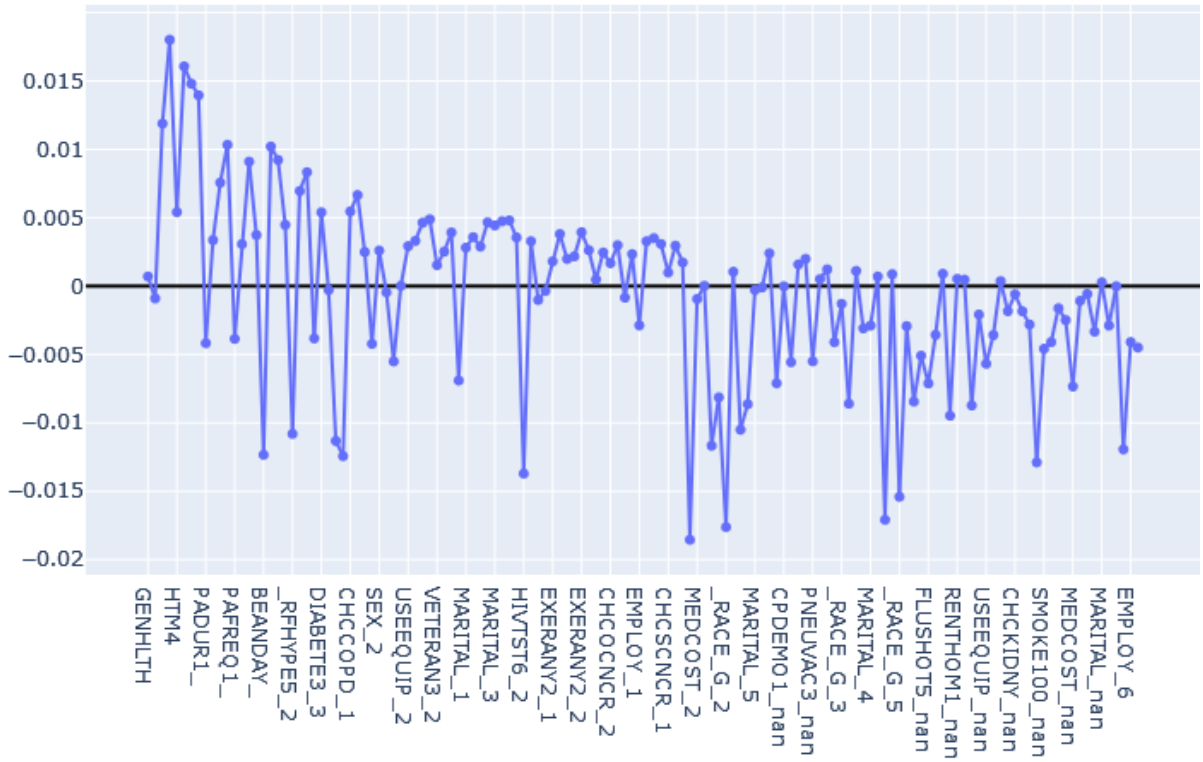

Figure 3: measure of the compromise between the importance of variables and their progressive position

5. Analysis

Classification models were trained using the H2O AutoML Learner node, which had already been used for feature selection. The H2O AutoML Learner is optimized to train not only random forests (RFs), but also generalized linear models (GLMs), gradient boosting machines (GBMs), feedforward artificial neural networks, and stacked ensembles. After training the chosen models with cross-validation, H2O AutoML Learner selects the best scoring model according to a predefined sorting metric. To address the challenges posed by data imbalance, four experiments were conducted with different adjustments to the training set distribution, following guidelines from specialized sources such as Google [8] and TensorFlow [19]. In these experiments, training was enabled for all five algorithms (the GLM was obviously a logistic regression, given the binary classification task), with the area under the precision-recall curve set as the sorting metric.

In each experiment, the selected models were evaluated both numerically (confusion matrices and appropriate accuracy statistics) and graphically with ROC and precision-recall curves.

5.1. Learning and prediction

- In the first experiment, the distribution of the training set remained unchanged, with 24,034 records (6.0% of the training set) from the minority class and 379,122 records (94%) from the majority class.

- In the second experiment, the majority class was undersampled to 96,136 records in order to increase the representativeness of the minority class to 20% of the training set and fall within a case of mild imbalance [8]. The records of the undersampled class were then reweighted in proportion to the old distribution, with a factor of $\frac{379{,}122}{96{,}136} \approx 3.9$. The option to weight the records was available in the advanced settings of H2O AutoML Learner.

- In the third experiment, the distribution of the training set was kept unchanged, but the records were weighted to achieve a virtual balance between the classes: the minority class was weighted with a normalized factor of $\frac{201{,}578}{24{,}034} \approx 8.4$ and the majority class was weighted with $\frac{201{,}578}{379{,}122} \approx 0.5$.
- In the fourth experiment, the minority class was oversampled to the same number of records as the majority class (379,122), resulting in a total of 758,244 records in the training set. The option to change the proportions of the two classes was available in the advanced settings of H2O AutoML Learner.

The models that proved to be most effective were the stacked ensembles, specifically in the form of a GLM trained on various minor models (first, second and fourth experiments), and a gradient boosting machine (third experiment).

5.2. Scoring

Table 1 summarizes the goodness-of-fit scores of the four selected models in the order in which they were presented, including confusion matrix scores (TP, TN, FP and FN) and metrics such as accuracy (CA) and area under the ROC curve (AUC) [20].

Experiment	TP	TN	FP	FN	CA	Precision	Recall	F_1	AUC
1	3,286	91,159	3,621	2,723	0.937	0.476	0.547	0.509	0.90
2	3,272	91,209	3,571	2,737	0.937	0.478	0.545	0.509	0.90
3	3,178	91,381	3,399	2,831	0.938	0.483	0.529	0.505	0.90
4	3,213	91,334	3,446	2,796	0.938	0.483	0.535	0.507	0.90

Table 1: accuracy statistics obtained

Figure 4 also shows (for the first model, with similar results in the other cases) the ROC and precision-recall curves, obtained using the Python View node of the Python extension.

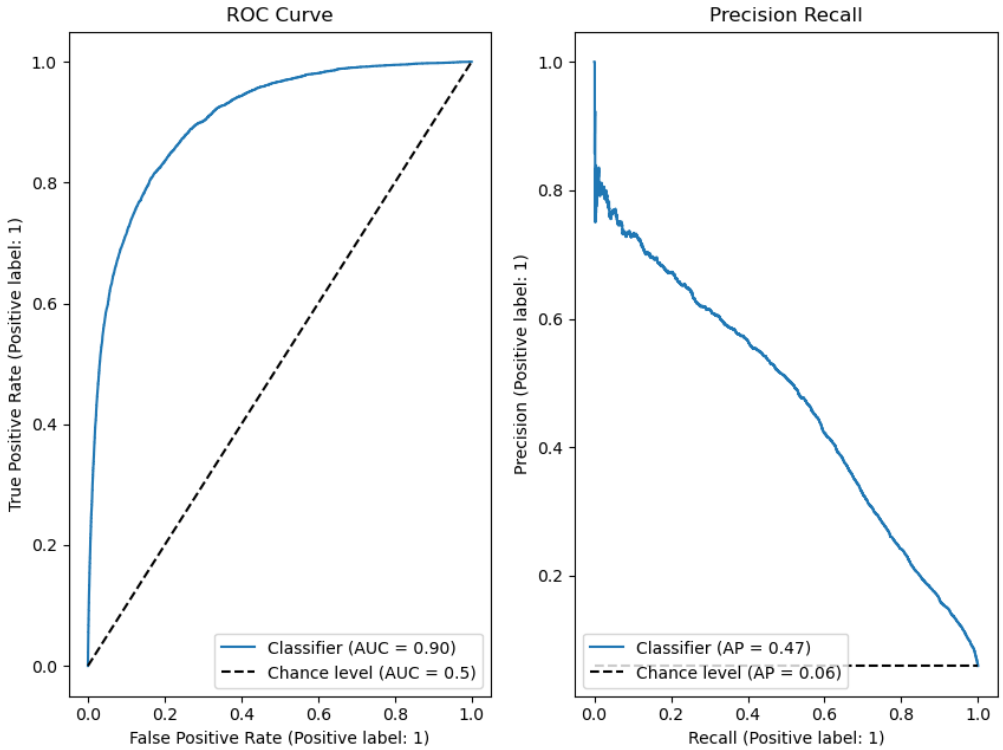

Figure 4: curves of the first model (in blue) and the random model (dashed)

6. Conclusion

Due to the skewness of the data, the accuracy remains consistently high and unchanged, making it an unsatisfactory metric. The AUC, which is a measure of the trade-off between recall (the proportion of true positives) and the proportion of false positives, is known to be more appropriate for similar cases; it also remains high and constant (0.9) across models. The F_1, which is the harmonic mean of recall and precision, is a comparable but perhaps even more appropriate compromise measure for the purposes of these models, as it depends directly on the classification errors of the positive class. It is important to emphasize that the primary interest of this work is to identify true positives (those truly at risk of heart attack) and secondarily to reduce false positives (those considered at risk when they are not). The cost of missing a true positive (i.e. a high-risk patient) is much higher than the cost of a false positive. Therefore, a higher recall is preferred, even if it means a decrease in precision, leading to a preference for the first model over the others.

Realistically, however, it must be acknowledged that the results, at least those considered, were approximately invariant across the four models. This observation, given the variety of strategies used to address the issue of data imbalance, led to the belief that data imbalance did not significantly affect the training of the models.

It was concluded that, in general, the constructed models failed to recognize almost half of the true positives. It should be noted, however, that this consideration, which is rather severe in itself, is only informative when combined with the value of the area under the precision-recall curve of the trained models (figure 4). This value is almost 8 times that of the area under the same curve in the absence of a model. A moderate change in slope, approximately in the region $[0.4, 0.6] \times [0.4, 0.6]$ of the precision-recall curve, also indicated the threshold possibly most suitable for the trade-off between the two metrics. Thus, if one assumes an acceptable proportion of true positives of 0.6, for example, one obtains a detection of positives that is almost 7 times more accurate than would otherwise be the case.

References

[1] Centers for Disease Control and Prevention. *Behavioral Risk Factor Surveillance System (BRFSS)*. Accessed: 2024-06-14. 2024. URL: https://www.cdc.gov/brfss/index.html.

[2] *KNIME software*. Accessed: 2024-06-14. 2024. URL: https://www.knime.com/blog/how-to-automate-machine-learning.

[3] BRFSS. *CDC – BRFSS 2011 Survey Data and Documentation*. Accessed: 2024-06-14. 2024. URL: https://www.cdc.gov/brfss/annual_data/annual_2011.htm.

[4] *Codebook*. Accessed: 2024-06-14. 2024. URL: https://www.cdc.gov/brfss/annual_data/2011/pdf/CODEBOOK11_LLCP.pdf.

[5] BRFSS. *Calculated Variables*. Accessed: 2024-06-14. 2011. URL: https://www.cdc.gov/brfss/annual_data/2011/pdf/2011_Calculated_Variables_Version29.pdf.

[6] BRFSS. *Prevelance and trends tool*. Accessed: 2024-06-14. 2024. URL: https://www.cdc.gov/brfss/brfssprevalence/index.html.

[7] Wikipedia. *Education in the Unitaed States*. Accessed: 2024-06-14. 2024. URL: https://en.wikipedia.org/wiki/Education_in_the_United_States.

[8] Google. *Unbalance dataset*. Accessed: 2024-06-14. 2024. URL: https://developers.google.com/machine-learning/data-prep/construct/sampling-splitting/imbalanced-data?hl=it.

[9] Sang Kyu Kwak-Jong Hae Kim. *Statistical data preparation: management of missing values and outliers*. Accessed: 2024-06-14. 2017. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5548942/.

[10] Skikit-learn. *Preprocessing data*. Accessed: 2024-06-14. 2024. URL: https://scikit-learn.org/stable/modules/preprocessing.html.

[11] Skikit-learn. *Compare the effect of different scalers on data with outliers*. Accessed: 2024-06-14. 2024. URL: https://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html.

[12] Wikipedia. *Curse of dimensionality*. Accessed: 2024-06-14. 2024. URL: https://en.wikipedia.org/wiki/Curse_of_dimensionality.

[13] Felicity Boyd Enders. *Collinearity*. Accessed: 2024-06-14. 2013. URL: https://www.britannica.com/topic/collinearity-statistics.

[14] scikit-learn. *Feature selection*. Accessed: 2024-06-14. 2024. URL: https://scikit-learn.org/stable/modules/feature_selection.html#tree-based-feature-selection.

[15] scikit-learn. *Feature importances with a forest of trees*. Accessed: 2024-06-14. 2024. URL: https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html.

[16] MS Ranya N. Sweis MD et al. *Angina*. Accessed: 2024-06-14. 2024. URL: https://www.merckmanuals.com/home/heart-and-blood-vessel-disorders/coronary-artery-disease/angina.

[17] NHS. *Coronary heart disease*. Accessed: 2024-06-14. 2024. URL: https://www.nhs.uk/conditions/coronary-heart-disease/.

[18] A.H.J.d. Reijer et al. *An heuristic scree plot criterion for the number of factors*. Accessed: 2024-06-14. 2024. URL: https://link.springer.com/article/10.1007/s00362-023-01517-x#citeas.

[19] Tensorflow. *Classificazione su dati sbilanciati*. Accessed: 2024-06-14. 2024. URL: https://www.tensorflow.org/tutorials/structured_data/imbalanced_data?hl=it.

[20] Wikipedia. *AUC:area under the curve*. Accessed: 2024-06-14. 2024. URL: https://it.wikipedia.org/wiki/Area_Under_the_Curve.