Application of ML techniques in the medical field, prediction of heart diseases

E Ancarani

1. Introduction

In the medical field Artificial Intelligence (AI) employs the use of Machine Learning (ML) models to search through medical data and uncover insights to help improve health outcomes and patient experiences [1]. One application of ML is also the generation of prediction models for early detection and treatment of diseases. In our case we will focus on heart diseases which are one of the most prevalent diseases as well as one of the one of the main causes of death in the United States. The majority of individuals only learn they have a heart disease following severe symptoms such as chest pain, a heart attack, or sudden cardiac arrest. This fact highlights the importance of preventative measures and tests that can accurately predict heart disease in the population prior to negative outcomes like myocardial infarction (heart attacks) taking place. [2]. The goal of this project is to investigate on different methods for predicting heart diseases by analysing a heart disease database provided on Kaggle [2]. In order to do that, we divide our problem into three tasks; first we carry out an EDA for to have a clear understanding of the data at hand, then we build the model and we apply machine learning algorithms to perform a binary classification of heart disease. For our study, we analyse different classification models such as k-nearest neighbors (KNN), decisional tree (DT), random forest and XGBoost (XGB). Finally, we are going to compare their results in terms of accuracy and F1-score.

2. Operating Environment

This project project is developed entirely on Jupyter Notebook. The version of IPython used is the 8.5.0.

3. Dataset and Features

The Behavioral Risk Factor Surveillance System (BRFSS) is a health-related telephone survey that is collected annually by the CDC. This original dataset collected in 2015 contains responses from 441,455 individuals and has 330 features. These features are either questions directly asked of participants, or calculated variables based on individual participant responses. The Kaggle dataset under study is a cleaner version of the BRFSS collected in 2015 which includes 253,680 survey responses and 22 features stored in a .csv file to be used primarily for the binary classification of heart disease. [2] In the annotation .cvs file we have the columns: High Blood Pressure, High Cholesterol , Cholesterol Check, BMI, Smoker, Stroke, Diabates, Physical Activity, Fruits, Veggies, Heavy Alchool Consumption, Health Care Insurance, No Doctor because of the cost, Genetic Health. Physical Health, Difficulty in Walking, Gender, Age, Education, Income. The target variable is Heart Disease or Attack, i.e the feature under study. Columns are better explained in the notebook, but a clear explanation of all the 330 features, comprising of those under study, is provided by the CDC official website [3]. The challenge of this dataset is that is heavily imbalanced. The distribution of the target variable is uneven. In fact, out of 229781 patients recorded, only 23717 have heart disease, which is the 10.3%. Meaning that the result will be biased towards that majority class. In this project are explored different approaches for handling this problem.

4. Data Analysis

Before applying the model, the following step to prepare and understand the data are carried out:

4.1. Data Preprocessing

All features, except for BMI, are integer values but initially stored as float64. In order to save resources, all features but BMI are converted as int32. In this dataset there's no presence of null values. Instead, there were `23899` duplicates values which have been removed in order to allow the model to better generalise the full dataset. Diabetes feature values have been transformed into binary as, initially, there was a value which represented a severe risk of diabetes. As a value with very few samples, I decided to merge those borderline samples with the diabates ones. Outliers are only present in the feature of BMI. However, the distribution is skewed (figure 4.1), i.e the peak to the left and then descend more slowly, this means it considers people with very high BMI (outliers). Performing statistical analysis it has been observed that people with a high body mass index (BMI) are more likely to have diabetes and relevant percentages also show how these people also had cases of heart disease. Then, removing outlier could lead to the risk of cutting important data for training the model.

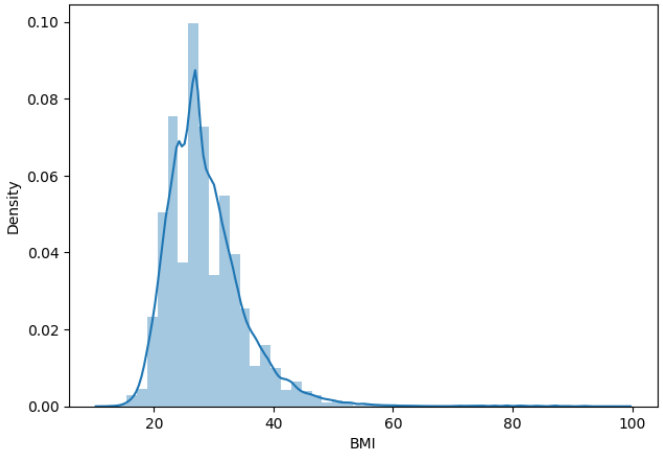

Figure 4.1: Variable correlation

4.2. Exploratory Data Analysis

Through EDA is possible to discover pattern in the data, investigate on data characteristics through statistics and graphical representations. In the notebook are performed some statistics through graphical representation and crosstab in order to deepen the knowledge of the data. In figure 4.2 it is shown a map of the correlation of the features.

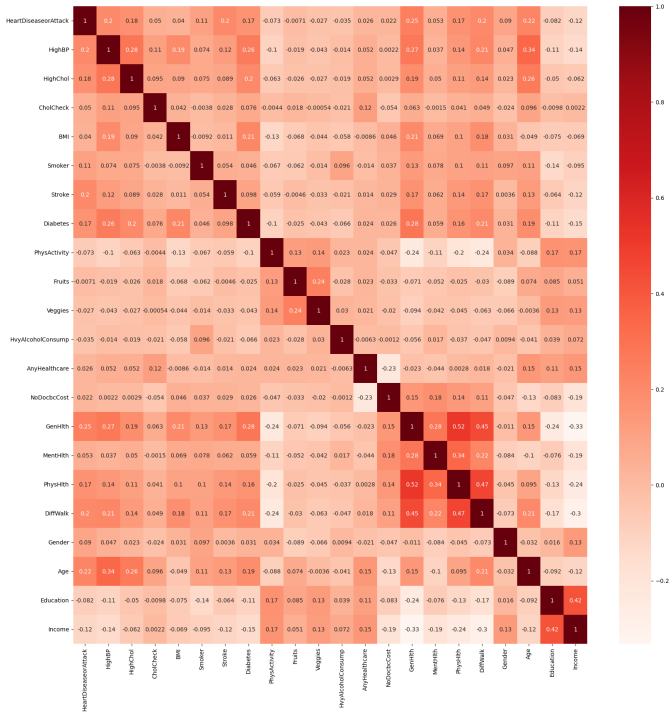

Figure 4.2: Variable correlation

Correlation value is between -1 (negatively correlated features) and 1 (positively correlated features). Default method used to calculate the strength of linear relationship between two features X and Y is Person Correlation coefficient which is calculated as follows:

$$\frac{covariance(X,Y)}{std(X) * std(Y)} \tag{1}$$

In this case, features do not present a particular high correlation value between each other. For this reason, I decided to keep all the features to train the models.

5. Sampling Methods

To handle the imbalance problem different sampling techniques have been implemented. In this study are presented different oversampling and undersampling techniques and hybrid methods that are combinations of the two. After that, those methods were tested on the models to see which sampling technique leads to the best results. In the section 7, are shown the results of the models using the following techniques used to handle imbalance problem.

5.1. Random Oversampling

Naive oversampling technique, it randomly duplicates samples from the minority class in the training set. This can lead to overfitting.

5.2. Random Undersampling

Naive undersampling technique, it delete sampeles from the majority class in the training set. This can lead to a loss of important information. In our case, a great deal of information from the majority class is lost as it deletes important amounts of data. Random Undersamping could be more effecting when the dataset is not as heavily imbalanced as in our case.

5.3. SMOTE

SMOTE is an oversampling algorithm and its advantage is that rather than generating duplicates, it creates synthetic data points which are slightly different from the original data points. SMOTE algorithms firstly draw a random sample from the minority class, from its observation, it identifies the k-nearest neighbors (default value is 5), then take one of those neighbors and identify the vector between the current data point and the selected neighbor. Then, it multiplies the vector by a random number between 0 and 1. To obtain the synthetic data point, it adds the result to the current data point. This way, you ensure that your synthetic data point is not an exact copy of an existing data point and that it doesn't differs too much from the known observations in your minority class. [4] Hybrid SMOTE techniques are also studied. The simplest one is SMOTE oversampling combined with random undersampling. Another hybrid techniques using SMOTE is SMOTETomek which combines the SMOTE ability to generate synthetic data for minority class and Tomek Links ability to remove the data that are identified as Tomek links from the majority class [5]. Tomek Links are pairs of opposite-class instances that are very close together. SMOOTEEN hybrid technique is also presented.

It performs firstly over-sample of the minority class using SMOTE and then uses the Edited Nearest Neighbor method to under-sample the data by removing samples close to the decision boundary. [6]

5.4. ADASYN

ADASYN is an oversampling method proposed by He et al. [7]. The main idea of ADASYN is to create data according to the density of existing ones. ADASYN uses a weighted distribution for different samples of the minority class according to their level of difficulty in learning, where more synthetic data is generated for samples of the minority class that are harder to learn compared to those minority samples that are easier to learn. As a result, the ADASYN approach improves learning with respect to the data distributions by reducing the bias introduced by the class imbalance, and by adaptively shifting the classification decision boundary toward the difficult examples. [7].

5.5. NearMiss

NearMiss is an under-sampling technique. It has some rules additional to RandomUnderSampler. NearMiss has two important hyperparameter: number of neighbor and version. The former refers to the neighborhood size to consider to compute the average distance to the minority samples. The latter refers to the fact that NearMiss has 3 versions. In this study we refer to version 3 which is a 2-step algorithm: for each minority sample, their m nearest-neighbors will be kept; then, the majority samples selected are the ones for which the average distance to the k nearest neighbors is the largest [8].

6. Methods

Dataset is split into training data with the 0.30% of test data. In order to predict whether a patient is affected by heart disease or not, in this study are used several classifier models.

6.1. K Nearest Neighbour

First classifier method used to test the dataset is k-nearest neighbors algorithm. It is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. This classifier, in fact, generates all the distance between the different data in the graphs using Euclidean distance (typically), to retrieve which are the k (settled during design time) closest neighbors starting from unseen datata. [9]

6.2. XGBC

XGBoost stands for eXtreme Gradient Boosting and it is an implementation of Gradient Boosted decision trees. In gradient boosting algorithms, each predictor corrects its predecessor's error. In XGBoost algorithm, decision trees are created in sequential form. Weights play an important role as they are assigned to all the independent variables which are then fed into the decision tree which predicts results. The weight of variables predicted wrong by the tree is increased and these variables are then fed to the second decision tree. These individual classifiers/predictors then ensemble to give a strong and more precise model. [10] The 'xgboost' is an open-source library that provides machine learning algorithms under the gradient boosting methods. The xgboost.XGBClassifier is a scikit-learn API compatible class for classification. In this study, we use XGBClassifier together with cross validation as XGBoost supports k-fold cross validation. [11]

6.3. Decision Tree Algorithm

The decision tree algorithm is a non-parametric supervised learning method that is used to predict a value based on decision rules gathered from a data structure. The decisional tree aims to minimize the impurity function, which measures the label purity of the children of a given leaf's tree. In this study we will use the library *sklearn*, more specifically, we will use the class *DecisionTreeClassifier*. The impurity function used is the *entropy* one.

$$Entropy = \sum_{j} p_j * log_2 * p_j \tag{2}$$

6.4. Random Forest Algorithm

Last model I would like to test the dataset with is the random forest. This classifier should increase the accuracy of the prediction since it is composed of several decisional trees. Theoretically, the more trees we have in the forest, the more robust it is. On the other hand, it is not always needed to use huge tree numbers, it would be good trade-off based on the accuracy we get and the complexity of the model. It is possible to sum up the classification process of the random forest for an unseen data in the following way:

Algorithm 1 Random forest prediction pseudocode

```
for each unseen data of the test set do
    for each tree in the random forest do
        - get prediction of x based on tree
        - pick next tree in the random forest
    end
    - make majority voting
    - assign new value to label
    - pick next unseen data
end
```

6.5. Hyperparameter Tuning

By dividing the data into train and test, overfitting may not be avoided. Cross validation helps to mitigate the problem of overfitting by splitting the training data into k subset, 5 in this case. With cross validation the

model is trained using k-1 of the folds as training data and the resulting model is validated on the remaining part of the data [12]. This methods is computationally expensive but helps in assessing the effectiveness of the model. All methods presented, except KNN, were trained using cross validation technique.

7. Results

As the dataset is heavily imbalanced, high accuracy does not mean necessarily good results. The models used to perform binary classification are KNN, DecisionTree, RandomForest, XGBoost. Table 1 contains all the results obtained by combining the different sampling methods and the chosen models. The chosen strategies are respectively: Random Over Sampling (ROS), Adasyn, SMOTE, Random Under Sampling (RUS), NearMiss. Hybrid resampling strategies tried: combination of random under sampling and random over sampling (ROS & RUS), SMOTE & Random Under Sampling (SM & RUS), SMOOTEN with Edited Nearest Neighbours (ENN) has has been tried by handling the parameter `enn` in two different ways: resample all classes (SMOOTEN in Table 1) and resample only the majority class (SM.EN-E in Table 1).SMOTE with Tomek Links combines over- and under-sampling using SMOTE and Tomek Links [13] and it has been tried by handling the parameter `tomek` in two different ways: resample all classes and resample only the majority class (relatively STK and STKTL in Table 1).

Table 1: Results

Model	Strategy	TN	TP	Acc	F1
	OS				
KNN	ROS	0.95	0.15	0.87	(0) 0.93 (1) 0.18
DT	ROS	0.91	0.24	0.84	(0) 0.91 (1) 0.24
RF	ROS	0.88	0.49	0.84	(0) 0.91 (1) 0.24
XGB	ROS	0.76	0.49	0.84	(0) 0.91 (1) 0.38
KNN	Adasyn	0.88	0.23	0.87	(0) 0.93 (1) 0.18
DT	Adasyn	0.82	0.35	0.77	(0) 0.86 (1) 0.24
RF	Adasyn	0.77	0.60	0.75	(0) 0.86 (1) 0.24
XGB	Adasyn	0.78	0.50	0.75	(0) 0.85 (1) 0.29
KNN	SMOTE	0.89	0.23	0.82	(0) 0.90 (1) 0.21
DT	SMOTE	0.82	0.36	0.77	(0) 0.87 (1) 0.24
RF	SMOTE	0.70	0.74	0.70	(0) 0.81 (1) 0.34
XGB	SMOTE	0.69	0.73	0.70	(0) 0.80 (1) 0.33
	US				
KNN	RUS	0.87	0.35	0.82	(0) 0.90 (1) 0.29
DT	RUS	0.72	0.78	0.72	(0) 0.82 (1) 0.37
RF	RUS	0.70	0.79	0.71	(0) 0.81 (1) 0.36
XGB	RUS	0.71	0.76	0.71	(0) 0.82 (1) 0.35
KNN	NearMiss	0.61	0.42	0.59	(0) 0.72 (1) 0.17
DT	NearMiss	0.50	0.60	0.51	(0) 0.65 (1) 0.20
RF	NearMiss	0.39	0.66	0.41	(0) 0.54 (1) 0.19
XGB	NearMiss	0.49	0.66	0.51	(0) 0.64 (1) 0.22
	Hybrid				
KNN	RUS&ROS	0.95	0.13	0.87	(0) 0.93 (1) 0.17
DT	RUS&ROS	0.91	0.26	0.84	(0) 0.91 (1) 0.25
RF	RUS&ROS	0.75	0.76	0.75	(0) 0.84 (1) 0.39
XGB	RUS&ROS	0.74	0.76	0.74	(0) 0.84 (1) 0.38
KNN	SM.&RUS	0.89	0.21	0.82	(0) 0.90 (1) 0.20
DT	SM.&RUS	0.82	0.34	0.77	(0) 0.87 (1) 0.24
RF	SM.&RUS	0.70	0.73	0.71	(0) 0.81 (1) 0.34
XGB	SM.&RUS	0.69	0.73	0.70	(0) 0.86 (1) 0.30
KNN	SMOTEEN	0.79	0.50	0.76	(0) 0.85 (1) 0.30
DT	SMOTEEN	0.71	0.62	0.70	(0) 0.81 (1) 0.30
RF	SMOTEEN	0.58	0.88	0.61	(0) 0.73 (1) 0.32
XGB	SMOTEEN	0.59	0.86	0.62	(0) 0.73 (1) 0.32
KNN	STK	0.89	0.23	0.82	(0) 0.90 (1) 0.21
DT	STK	0.82	0.36	0.77	(0) 0.87 (1) 0.24
RF	STK	0.70	0.73	0.70	(0) 0.81 (1) 0.34
XGB	STK	0.69	0.73	0.70	(0) 0.80 (1) 0.33
KNN	STKTL	0.89	0.23	0.82	(0) 0.90 (1) 0.21
DT	STKTL	0.82	0.35	0.77	(0) 0.87 (1) 0.24
RF	STKTL	0.70	0.73	0.70	(0) 0.80 (1) 0.33
XGB	STKTL	0.69	0.73	0.70	(0) 0.80 (1) 0.33
KNN	SM.EN-E	0.78	0.51	0.75	(0) 0.85 (1) 0.29
DT	SM.EN-E	0.71	0.64	0.70	(0) 0.81 (1) 0.30
RF	SM.EN-E	0.57	0.88	0.61	(0) 0.72 (1) 0.31
XGB	SM.EN-E	0.58	0.87	0.61	(0) 0.73 (1) 0.31

Best Results is obtained by Random Forest using SMOTE as resampling method.

Table 2: SMOTE - Model Parameters

Model	Max Depth	Accuracy	Precision
RF	12	70	(0) 0.96 (1) 0.22
XGBoost	10	70	(0) 0.96 (1) 0.21

Nearly the same results are obtained using both versions of SMOTE with Tomek Links. As in the medical field it is usually preferred to detect true positives values, this are the models that are slightly more precise and that gives a good trade off between accuracy value and the F1 score. Precision for these models is the same `0.96` for the detection healthy patients and `0.22` for the detection of sick ones. This result can be explained by the number of false positives which is much higher compared to the number of true positives.

References

[1] IBM: *How is artificial intelligence used in medicine?* https://www.ibm.com/topics/artificial-intelligence-medicine.

[2] CDC: *Heart disease health indicator*, 2015. https://www.kaggle.com/datasets/alexteboul/heart-disease-health-indicators-dataset.

[3] CDC: *Column explanation.* https://www.cdc.gov/brfss/annual_data/2013/pdf/codebook13_llcp.pdf.

[4] https://towardsdatascience.com/smote-fdce2f605729.

[5] https://towardsdatascience.com/imbalanced-classification-in-python-smote-tomek-links-method-6e48dfe69bbc.

[6]

[7] He, Haibo, Yang Bai, Edwardo Garcia, and Shutao Li: *Adasyn: Adaptive synthetic sampling approach for imbalanced learning*, July 2008.

[8] https://hersanyagci.medium.com/under-sampling-methods-for-imbalanced-data-clustercentroids-randomundersampler-nearmiss-eae0eadcc145.

[9] https://www.ibm.com/topics/knn.

[10] https://www.geeksforgeeks.org/xgboost/.

[11] https://www.datatechnotes.com/2019/07/classification-example-with.html.

[12] https://scikit-learn.org/stable/modules/cross_validation.html.

[13] https://imbalanced-learn.org/stable/references/generated/imblearn.combine.SMOTETomek.html.