Cardiovascular disease detection

Rodrigo Ibarra Gaitan, Jaime Yussef León Contreras, Iván Alejandro Ávila González and Hiram Ponce

Universidad Panamericana, Facultad de Ingeniería,
Augusto Rodin 498, Ciudad de México, 03920, México
0252692@up.edu.mx, 0251991@up.edu.mx, 0252002@up.edu.mx

Abstract. In this study we find an interpretable way to detect Cardiovascular diseases. Using CDC's surveys data set consisting of over 320K records and 279 columns originally but reduced to 18. We explore performance of Random Forest and Logistic Regression models on this data set, hyper parameter tuning and interpretability. Though some improvement can be done by finding more features or using more advanced feature engineering techniques we found Random Forest performed better than Logistic Regression and best predictors for CVDs are Age, Body mass index (BMI), Sleep Time, Physical Activity and Mental Health.

Keywords: Machine Learning · Classification · Heart Disease

1 Introduction

Detecting Cardiovascular diseases (CVDs) prematurely is of great interest for the Healthcare Industry. According to the World Health Organization (WHO), almost 18M deaths, representing 32% of global deaths in 2019 were caused by Heart Diseases, also referred to as CVDs. Furthermore, 75% of these cases took place in low or middle income countries, not to mention that 38% of premature deaths (under the age of 70) were caused by CVDs.

On top of that, Centers for Disease Control and Prevention (CDC) claim that about half of USA's population has at least one of three risk factors for CVD, high blood pressure, high cholesterol and smoking. With that in mind, CDC surveyed American citizens all over the country which led to a data set containing almost 400k observations, whether the person in question has ever been diagnosed either with Coronary Heard Disease or Myocardial Infraction.

Even though several approaches on how to detect these kind of disease using Machine Learning (ML) have been developed through out recent times, in this study we aim to find a robust yet "interpretable" way of finding out which risk factors are more strongly correlated with CVDs, particularly Coronary Heart Disease and Myocardial Infarction.

2 Related Work

Various efforts have been made towards finding out which is the best, most accurate model for CVD detection. Here we aim not only to ding an accurate predictor but to fins which risk factors are more closely related to CVD detection, leading to prevention or better and early management.

In 2018 [4] showed how Decision Trees might not be a great candidate by themselves as it's quite hard not to over fit, while Support Vector Machines can easily outperform Decision Trees and Random Forest or Ensemble Methods can perform well too. Complementing these results, [2] have shown show how different models performed better while coupled with different over sampling techniques. Particularly, it was found that Synthetic Minority Oversampling was the best match for Random Forest, SVM and Random Over Sampling work great together, too. And, Adaptive Synthetic Sampling helped, once again, Random Forest get the best results.

The latter is very relevant as it turns out some other work has been done around finding out which Tree based models work better on CVD detection. Using a data set from UCI Machine learning repository, [3] found that J48 is the best technique for CVD detection amongst some other well known Tree based methods.

3 Dataset Description

The dataset used for this study, was downloaded from kaggle repository. Originally, it comes from CDC's (Centers for Disease Control and Prevention) Behavioral Risk Factor Surveillance System which conducts telephone surveys about the health status of USA's citizens and contained around 300 features. It has been pre processed and cleaned so that this study will be conducted using a smaller data set with 18 features.

The data set is divided into 13 categorical variables, 4 numerical variables and a categorical feature, consisting of 319795 records without null values but it is heavily unbalanced. The following table represents the frequency of the most comon category on each categorical feature.

Column	Unique	Top	Frequency
HeartDisease	2	No	292422
Smoking	2	No	187887
AlcoholDrinking	2	No	298018
Stroke	2	No	307726
DiffWalking	2	No	275385
Sex	2	Female	167805
AgeCategory	13	65-69	34151
Race	6	White	245212
Diabetic	2	No	269653
PhysicalActivity	2	Yes	247957
GenHealth	5	Very Good	113858
Asthma	2	No	276923
KidneyDisease	2	No	308016
SkinCancer	88	788	289976

The following image describe the distribution of some unbalanced categorical variables:

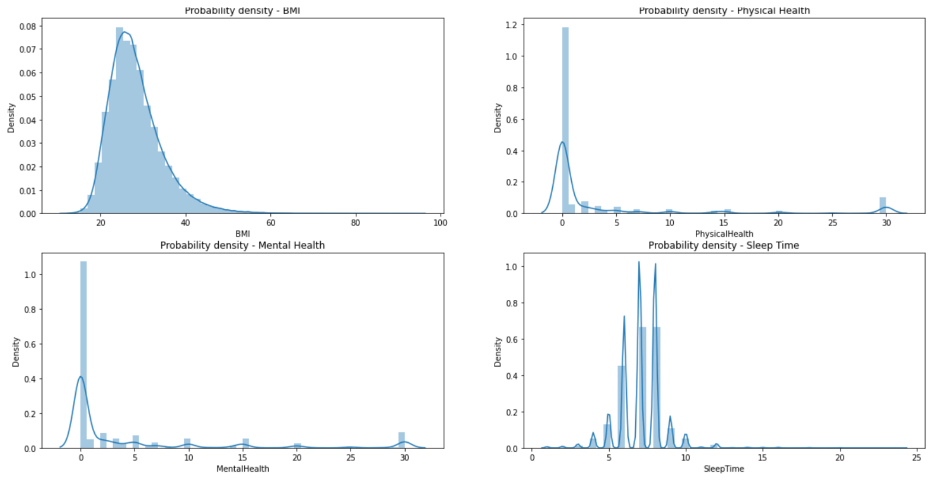

Fig. 1. Probability density for input variables.

We know that the number of observations we have is 319,795, of those 292,422 belong to the negative class. We need to balance the data.

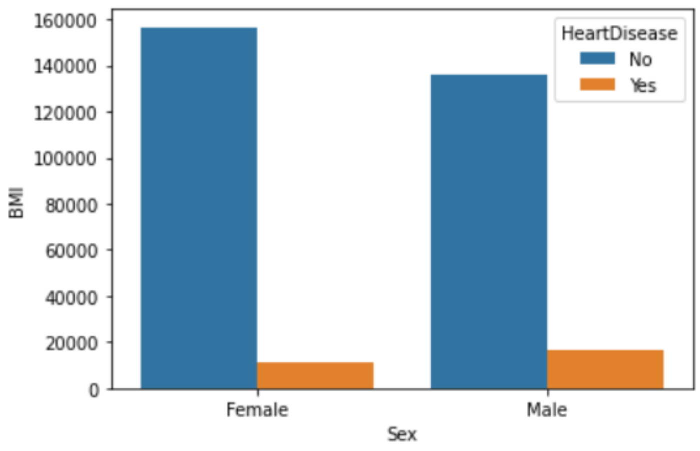

Fig. 2. Biased target column.

Now we can also do the visual analysis that relates each entry to the target.

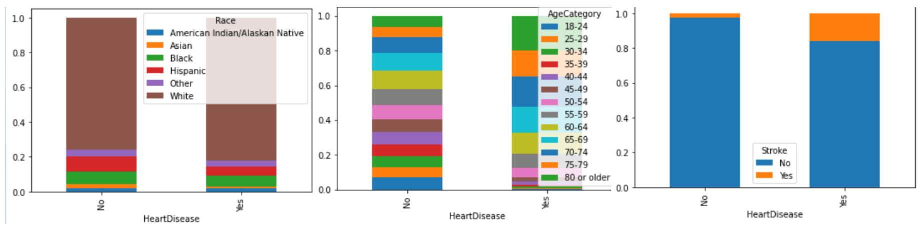

Fig. 3. Percentage of categories in relation to the target.

4 Description of the Proposal

As we are dealing with both categorical and numerical variables on top of an imbalanced dataset, we must have at least three pre porcessing tasks to do:

– Perform One Hot Encoding on categorical variables
– Perform standard scaling to numerical variables
– We use over sampling and under sampling to balance classes

With that in mind, the overall flow for training is:

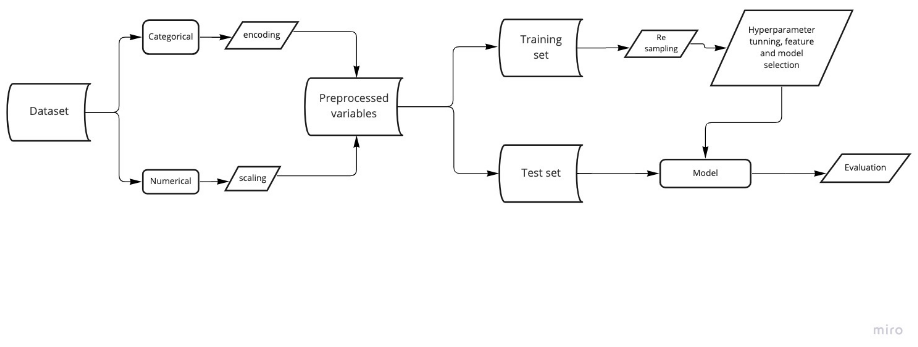

Fig. 4. ML Flow - An Overview

For feature prep we decided to use sklearn preprocessing module, OneHotEncoder and StandardScaler to perform the first part of the process. Then, for re sampling, we have used RandomOverSampler and RandomUnderSampler from imblearn.over_sampling. And, finally, sklearn.model_selection's train_test_split to separate the training and holdout sets. With that we are ready to move into hyperparameter tuning.

Instead of using the GridSearchCV library for hyperparameter tuning for the three models, we decided to use optuna [1] , which is an open source hyperparameter optimization framework to autmoate hyperparameter search more efficiently, one of the key reasons we decided to use this library is because it's able to do automated search for optimal hyperparameters using Python conditionals, loops, and syntax. It also uses State-of-the-art algorithms which help to efficiently search large spaces and prune unpromising trials for faster results, we can also parallelize hyperparameter searches over multiple threads or processes without modifying code. Bayesian optimization builds a probability model of the objective function and uses it to select hyperparameters to evaluate in the true objective function.

We decided to test three different algorithms with different hyperparameters, the first classifier is a SGD(Stochastic Gradient Descent) Classifier, the second classifier we used is a Logistic Regression and last a Random Forest Classifier.

– First we wrap model training with an **objective** function and return accuracy
– Suggest hyperparameters using a **trial** object
– Create a **study** object and execute the optimization

For the SGDClassifier we used four different hyperparameters such as alpha, penalty, loss and max_iter. For the Logistic Regression model the hyperparameters used were: penalty, c value, solver and fit_intercept and last for the Random Forest Classifier, it used four different hyperparameters such as max_depth, n_estimators, criterion and max_features.

This is how we define the objective function in Optuna. The objective function takes the trial object and returns the objective value. In our example, the objective function is called 20 times, for each model it does a cross validation with value of 3 and the best value, which is the mean of the three cross validations, obtained out of 20 trials and the parameters are printed.

At the end of the trial, optuna prints the best model with the best hyperparameters used for the model. In our example is trial 15, which is the RandomForest with the following hyper parameters. Additionally from being the best model, the Random Forest Classifier is a great option, since it is robust to outliers, works well with non-linear data, it has a lower risk of overfitting, runs efficiently on large datasets such as ours and has a better accuracy than the two models previously discussed.

Hyper-parameter	value
max_depth	38
n_estimators	299
criterion	gini
max_features	auto

The following graphs shows the objective value for each trial ran by the optuna objective function, in this case, it only shows the top 10 trials. We can see that it starts with an objective value around 0.76 and after the fourth trial, it shows the best objective value with a value around 0.915

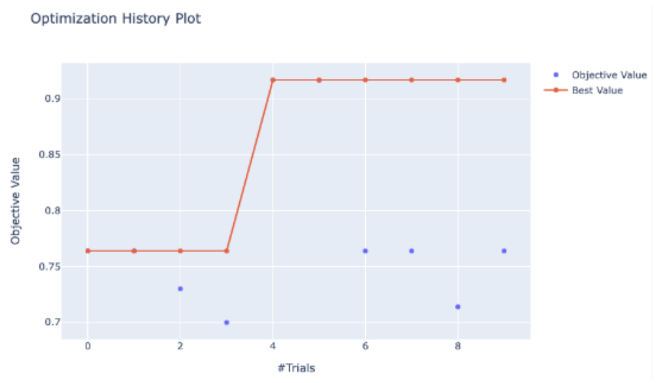

Fig. 5. Optimization History Plot for the objective value

Optuna has different visualizations once the objective function is run to have a better understanding of how the hyperparameters were obtained, the following picture shows how the model determined the hyperparameters only for the Random Forest Classifier.

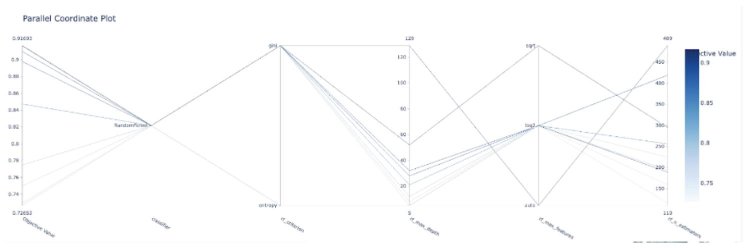

Fig. 6. Parallel Coordinate Plot

After understanding how the hyperparameters were obtain, it is also important knowing which hyperparameters has higher importance for the objective value, the following bar chart shows the importance of the hyperparameters for the Random Forest Classifier, were we can see that the max_depth has the higher importance with a value of 0.63, followed by the n_estimators.

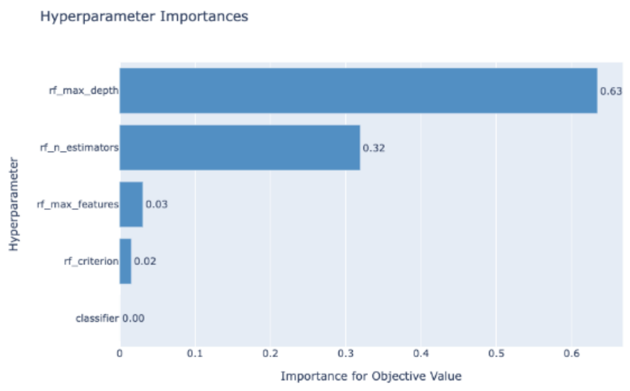

Fig. 7. Hyperparameter Importances

5 Experimentation

After the EDA we know the data types of the variables, and the next step was to proprocess them, for this, we created a pipeline, the first step was to normalize all numeric features using an standard scaler, for categorical values we used a One Hot encoder and last we used a label enconder for binary columns, including the target. Inputing of null values was not necessary because we did not have any empty values.

Next, we separated the dataset in the train/test split in this occasion we used 80 of the dataset as the training set, it is important to mention that we used the stratify parameter because we wanted to assure all target values were distributed in both train/test sets.

The third phase consisted on balancing the target feature because originally we had a ratio of 91 vs 9 of the negative and positive classes respectively. We did a random oversample to get a relationship of 70-30 and the a random undersample to get a 50-50 ratio. Important to say that we only did the balance of the train set, because it is incorrect to balance the whole dataset.

In order to find the best algorithm we ran a portion of Optuna code, where we tried several combinations of hyper-parameters for 2 main algorithms: logistic regression and random forest classifier. A priori the second one seemed like the best approach, and after the trials we found the indeed, the best algorithm was a random forest classifier, we found an accuracy of 94, while the logistic regression barely reached the 80.

With the best classifier chosen, we train our model using a pipline containing the prepossessing and the model sections.

6 Results and Discussion

After the training we proceeded to evaluate our model in the test set achieved an overall accuracy score of 0.85, but we had 2 very deficient metrics, the precision and recall were 0.28 and 0.46 for the positive class, while the f1 score was about 0.35, very far from the results for the negative class of 0.95, 0.89 and 0.92 respectively.

We also did a feature importance graph where we could find which variables were the most important ones for the model. Out of 41 different features, the following graph only shows the fourteen most important features.

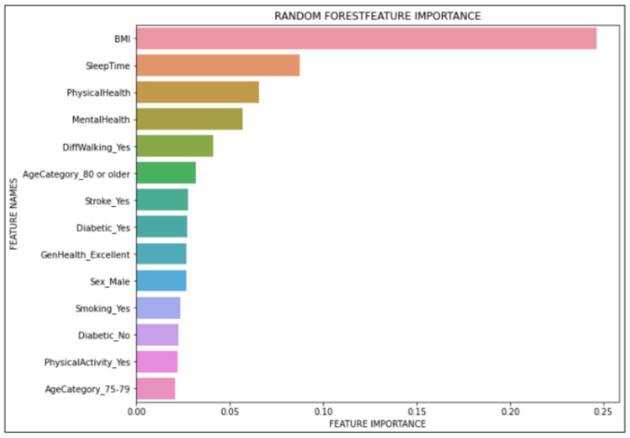

Fig. 8. Feature importance

Finally we did a confusion matrix in order to visually understand our results and a ROC curve to complete our model evaluation analysis. We can see that the percentage number of true negative values is 81.36 percent, which is around 104,068 negatives classified correctly, for the true positives, which are people with a heart disease the model classified correctly 5023 out of the 10,949 people with a heart disease. We can see that the number of false negative values are less than the true positive, which is a good thing, since we don't want to classify people with a heart disease negative more than people without a heart disease.

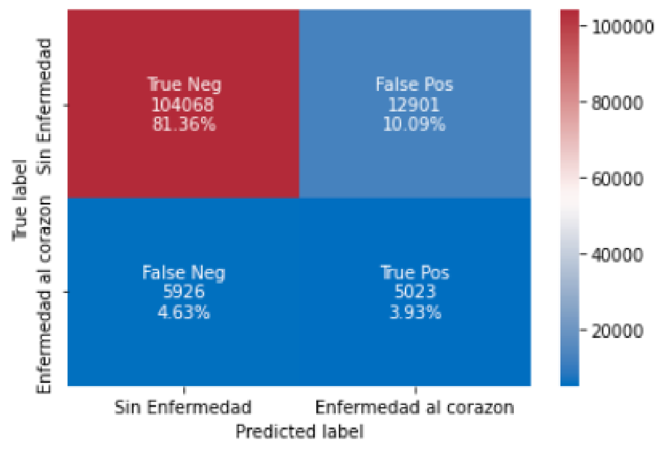

Fig. 9. Confusion matrix

The following graph shows a ROC(Receiver Operator Characteristic) curve, which helps us understand the trade-off between sensitivity (or TPR) and specificity (1-FPR). Classifiers that give curves closer to the top-left corner indicate a better performance. Where the higher value you can obtain is equals to 1, our models has an AUC (Area Unders Curve) of 0.81, which is equivalent to the probability that a randomly chosen positive instance is ranked higher than a randomly chosen negative instance. Note that the ROC does not depend on the class distribution. This makes it useful for evaluating classifiers predicting rare events such as ours.

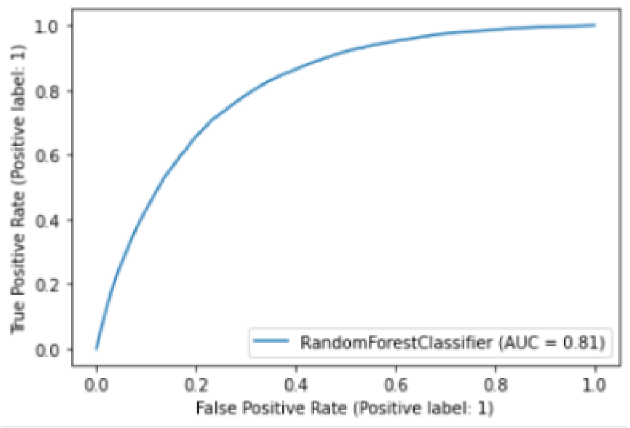

Fig. 10. ROC Curve

In order to create a functional App to show our model work we created an App on gradio, where the user insert the required parameters using a friendly interface and those inputs are processed in order to obtain a quick result of the heart problems of the person.

7 Conclusions

Random Forest out performed Logistic Regression which is consistent with EDA where we found features are not linearly separable. For Random Forest, more important features are what one would expect: weight, physical activity are best predictors for CVDs. Optuna turned out to be better than grid search as it allowed to test a bigger range of hyper parameters without exhaustive testing.

That being said, it is clear that imbalance is a big problem for this data set on top on non-separability. Other techniques for imbalanced learning can be applied to this problem like class weights, using gradient boosting for a more robust sequential tree ensemble would help better differentiate these two classes. Using these techniques should result in better classification performance that could help lots of business cases. For instance, a Hospital could survey patients and find out whether they are candidates for CVD prevention treatment or feature importance could help pharmaceutical companies target specific drugs on their campaigns or simply.

Even though our model has a good performance, since we're talking about the health industry, the model needs to improve, we can use techniques such as changing the threshold to classify false negatives even better and adding more features to our model to have a better complexity and understanding of the patient. Features such as: How many times a week the patient eats fast food, number of steps walked a day/week, grams of sugar consumed by the week, and screen time.

References

1. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: A next-generation hyperparameter optimization framework. In: Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2019)
2. Lakshmanarao, A., Swathi, Y., Sundareswar, P.S.S.: Machine learning techniques for heart disease prediction. Forest **95**(99), 97 (2019)
3. Patel, J., Tejalupadhyay, S., Patel, S.: Heart disease prediction using machine learning and data mining technique (03 2016). https://doi.org/10.090592/IJCSC.2016.018
4. Ramalingam, V.V., Dandapath, A., Raja, M.: Heart disease prediction using machine learning techniques: A survey. International Journal of Engineering Technology **7**, 684 (03 2018). https://doi.org/10.14419/ijet.v7i2.8.10557