NOVA INFORMATION
MANAGEMENT SCHOOL

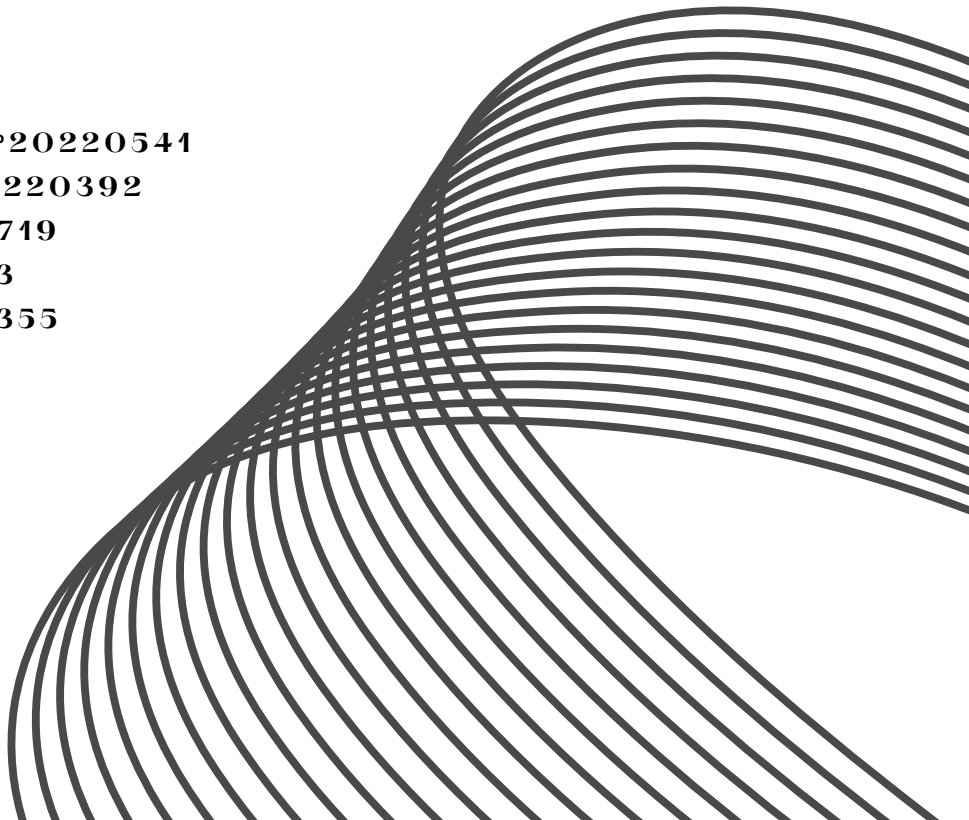
2022/2023

Machine Learning- Project Report

The Smith Parasite Project

GROUP MEMBERS:

ANA CAROLINA OTTAVI, Nº20220541
CAROLINA BEZERRA, Nº20220392
DANIEL FRANCO, Nº20210719
RAFAEL DINIS, Nº20221643
TOMAS VICENTE, Nº20221355



Abstract

Globally, the complexity of underlying symptoms presents a lot of challenges in understanding the disease we are dealing with, and consequently developing an effective treatment. Machine Learning allows researchers to identify early on patterns or designs that would be impossible to detect through the human eye.

For El Naqa and Murphy (2015), Machine Learning is defined as the evolving branch of computational algorithms that are designed to emulate human intelligence by learning from the surrounding environment. Machine Learning algorithms have the ability to learn from the existing data and predict unseen tasks, leading to a better understanding and improvements.

In this study, we review how to employ machine learning approaches to predict if a patient will suffer, or not, from the Smith Disease. The main purpose of this machine learning project was to maximize the prediction accuracy, with two possible targets.

The CRISP-DM framework was the architecture chosen to outline the planning of this project. For this purpose, we used patient information created to investigate Smith's disease, which is accessible in a Train and Test dataset. At this stage we started by performing Data Preparation, where we prepare our data to use further on in our machine learning training. Data Preparation includes tasks such as outlier and anomalies identification, missing values imputation, feature selection, feature engineering. Several models were tested, separated according to the type of algorithm, including decision trees, random forests, logistic regressions, K-Nearest Neighbors, AdaBoost, Gradient Boosting, XGBoost and Neural Networks. In this study, the Decision Tree model was the most successful machine-learning classifier in detecting diseased and non-diseased patients (with a base f1-score of 1.000000, a 0.974359 on the tuned dataset and 0.981000 on the validation subset).

Keywords

Machine Learning; Predictive Analysis; Supervised Learning; Binary Classification; Disease breakout prediction.

Introduction

This report addresses the final project in the course of Machine Learning of the Master’s Degree in Data Science and Advanced Analytics at Nova IMS. In this project, the purpose is to build a predictive model that answers the following questions: “Who are the people more likely to suffer from the Smith Parasite?”, “Can a model predict if a Patient will suffer, or not, from the Smith Disease with high precision?” and “What are the main drivers for an individual to have the disease?”.

Recently, a new disease has been discovered by Dr. Smith, which has already affected more than 5000 people. The effects this virus has on some people are quite dramatic and, in some cases, associated with conditions such as loss of speech, confusion, chest pain and shortness of breath. While there are no correlations that indicate a patient will suffer from it, there are groups of people more associated with this parasite than others.

Throughout the development of the project, the team impersonated a team of data scientists, which in order to answer the core goal of this research, we were provided a fictional dataset, containing patient information, including sociodemographic, health, and behavioral data.

The training dataset consists of 18 variables and 800 observations. The target variable “Disease” is the disease status for the patient, whereas it is equal to 1 if the patient has the disease and it is equal to 0 if the patient doesn’t have the disease. Since the target can only possibly fall into two categories (Diseased or Not Diseased), it can be termed as a classification problem. At last, a final model will be performed on the test dataset, consisting of 225 unseen observations and 17 variables (doesn't include the target).

Data Description

The datasets consist of sociodemographic, health and habits related features. The available demographic data contains basic information about the patients, such as Birth Year, Name, Region and Education level. The datasets on health and behavioral related data comprehend specific information of each patient, including height, weight, checkup, diabetes, high cholesterol, blood pressure, mental and physical health, smoking habit, drinking habit, exercise, fruit habit and water habit. Of the full training dataset, we can take away that out of the eighteen features, eight are quantitative variables and 10 qualitative. Table x, in the appendix, summarizes all the initial variables, as well as their data types and a brief description of what each one of them represents.

The dataset used for this project is balanced, from which of the total 800 observations, Disease= 0 represents around 48,63% of the dataset and the remaining occur when Disease=1. Given due consideration how balanced the dataset is, there was no need to apply under sample or oversample techniques.

Methodology

i. Handling duplicates

Regarding duplicates, initially we verified that ‘Name’ had one value in common among two of the patients. Afterwards, we checked for duplicates’ existence on both the testing and training data, and concluded that there were no duplicates.

ii. Handling Outliers

Finally, the outlier removal process for this dataset required multiple techniques, due to the complexity of our variables. For the first method, we used z-score, which reveals how many standard deviations away a given observation is from the mean, to detect outliers. For this analysis, values that lie within three and a half standard deviations above and below the mean will be considered outliers. We reported 3 observations for the feature ‘High_Cholesterol’ with a z-score higher or equal to 3.5; another 6 observations for ‘Blood_Pressure’ and, finally, 12 observations for ‘Birth_Year’. However, at last, we did not exclude the observations identified as outliers by this method. Lastly, after outlier removal we verified that 97.50% remained from the initial training dataset.

The other method used for outlier detection was the IQR method, considering prior boxplot analysis (Figure 1) for visualization of the features. In this method, initially we set up a fence outside of the first and third quartile. To build this fence we take the values that lie under the first quartile by more than 1.5 times the size of the interquartile range or lie above the third quartile by more than 1.5 times the size of the interquartile range. All the values outside the fence will be considered as outliers. After outlier removal with the IQR method, we determined the percentage of remaining data, which provided a value of 98.125%, when criterion is 3.

In the end, we were able to confirm the existence of extreme values both from the Z-score a

nd IQR approach, however we decided to remove the most extreme values we encountered through a manual approach. With manual outlier removal, several outliers can be excluded with precision, since it is the user who defines manually the conditions to determine which values are outliers. These conditions were defined based on the extreme values observed in the summary statistics and distribution plots, and other Outlier Detection Methods). After excluding the outliers manually, the dataset will remain with 98.25 % of its original observations.

The Manual Method is the one who conserves more data. IQR Method considers the maximum Physical Health (30) to be an outlier, but that's impossible since it's within the predefined range of the feature [0,30], therefore this will not be the method used. The Z-Score test detected the almost the same outliers as the Manual, but also considered high values of Blood Pressure to be outliers, but since they're at rest value it's possible that Patients can present values as high as those. Therefore, the Manual Method will be the one used.

iii. Data Split

The train-test split is a technique for evaluating the performance of a machine learning algorithm. On this step, the split will leave 20% in the hold out sample, meaning 80% of the Train Data will be used to train models, also the data will be randomly shuffled before being split and stratified, meaning it'll retain the same proportion of classes on train and test sets that are found in the original dataset.

Test data is data from the future, meaning it's unseen data to the models, therefore, all data imputation involving techniques (like mean, median, mode, z-score standardization etc...) have to be done after train-test split, otherwise might lead into a Data Leakage issue (when the training data contains information about the target, but similar data will not be available when the model is used for prediction. This led to high performance on the training set (and possibly even the test data), but the model will perform poorly in production). This is why we have to split the data now before finishing Data Preparation.

iv. Missing values

First, we checked for strange characters such as ('!', '$','%','?','*','+','_','@','€',' ','{'), which can be the result of mistakes or failure to record the data, and replace those with missing values. Afterwards we proceeded with a function that returns missing values considering a threshold of 5%, which concluded only the variable ‘Education’ reported missing values, which comprised around 1% of all the features for the training subset. In order to preserve the data, for the categorical feature ‘Education’ we selected to fill in the missing values with the most common value for this feature.

v. Feature Engineering

In order to extract more valuable insights from the dataset, we decided to create new variables. ‘Age’ was created with the sole purpose of increasing readability, ‘BMI’ was created to capture the same amount of information while minimizing the number of variables. Additionally, ‘Diabetes_Contact’ was created, which assumes the value of 1 whenever patients had direct or indirect contact with Diabetes, and assumes the value of 0 otherwise. This feature was later disconsidered from our models, since it didn’t improve our f1-score.

For numerical features, there will be redundancy among the newly created variables and the ones that generated them, so in order to select the features that'll proceed to the modeling stage, it was performed the Spearman correlation test on the train data . ‘Mental_Health’ revealed the highest correlation coefficient with the target, ‘Disease’. ‘BMI’ showed slightly more correlation with the target

than the feature ‘Weight’. In spite of that, since ‘Weight’ is used to calculate BMI, and consequently they share a strong relationship, so we decided to drop only the feature ‘Weight’. ‘Age’ and ‘Birth_Year’ are perfectly collinear (their correlation coefficient is +/- 1.0) and both have the same relationship strength with the target, just different directions. For better readability of values ‘Age’ will stay and ‘Birth_Year’ will be dropped.

vi. Normalization Study

Normalization is an important technique for Machine Learning often applied to a dataset.
The main purpose of normalization is to change the values to a common scale in such a manner that it's possible to run any algorithm without distortion. Because variance is dependent on scale, it's a good practice to normalize features before feature selection.

For this dataset, we compared four different approaches, including the raw dataset, MinMax scaler [0,1], MinMax scaler [-1,1] and Robust scaler. From the different scaling techniques, the MinMax[-1,1] Scaler had the better score by a slight margin. It'll be the one used moving forward. Since we obtained very similar results using MinMax Scaling (which preserves the most the shape of the original distributions) and Robust Scaling (which is better to control extreme values), we can conclude that, at this stage, we don't contain outliers in our dataset.

vii. Feature Selection

Feature selection means selecting and retaining only the most important and relevant features in the model. It is used to find the best set of features that allows one to build more useful models, while promoting a decrease in overfitting, an improvement in accuracy and a reduction of the training time of the machine learning algorithm. Feature selection techniques can be divided into two types - supervised and unsupervised. Regardless, our approach for feature selection focuses on supervised techniques, which is considered more accurate in contrast with unsupervised techniques. Supervised techniques handle labeled data where the output data patterns are known beforehand.

Categorical Features

Feature selection methods that are independent of any machine learning algorithm - the so called filter methods - make use of statistical techniques for selecting the features. Chi square statistical test, which is applied to categorical features, compares two variables in a contingency table to check their correlation using their frequency distribution. With this method, already on normalized data, it was assessed that variables such as ‘Name’, ‘Region’, ‘Education’, ‘Water_Habit’ and ‘Smoking_Habit’ were not relevant predictors, so they were immediately discarded. At this stage, we applied One-hot encoding on the remaining categorical variables, considering the use of pandas.get_dummies, whose purpose is turning categorical values into binary values of either 0 or 1. However, pandas.get_dummies creates a column for each unique value in the categorical variable. After categorical encoding, we built a spearman correlation heatmap as means to check columns that cause multicollinearity and have lower correlation strength with target. For these results, see Figure 2 on Attachments.

Numerical Features

To determine which features were relevant to the model, we used Filter, Wrapper, and Embedded methods for Feature Selection. As for numerical features, eight different methods were used. First of them all, we used the spearman correlation on the training data test to check which variables are highly correlated with our target variable. Afterwards, some other methods included: Step Forward and Backward Selection (both with a Random Forest Model), Recursive Feature Elimination (using a Logistic Regression Model), Kendall’s, Anova. We also applied LASSO regularization (L1) for feature

importance and Entropy Criteria for feature importance with a Random Forest, which revealed the most relevant features in our dataset. The Random Forest considers all current numerical features important, while LASSO Regression discards all variables except Diabetes Contact, Physical and Mental Health.

viii. Predictive Modeling

Globally, we used 16 predictive models: Decision Tree Classifier, Random Forest Classifier, AdaBoost Classifier, Gradient Boost Classifier, XGBoost Classifier, Extra Trees Classifier, Logistic Regression, K-Nearest Neighbours Classifier, Bagging Classifier, Neural Network, Hist Gradient Boosting Classifier, Stacking Classifier, Support Vector Classification, Gaussian Naive Bayes, Stochastic Gradient Descent, Voting Classifier.

Note: The random_seed of all models was defined as 0

Results

i. Cross Validation Study

“When evaluating different settings (“hyperparameters”) for estimators, there is still a risk of overfitting on the test set because the parameters can be tweaked on the model and evaluation metrics no longer report on generalization performance” (Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011). Cross Validation also allows the evaluation of the performance of a machine learning model, while repeatedly splitting the dataset and getting the performance of each one of the groups. This test was performed, so that later we could benchmark the models. We employed multiple cross-validation techniques, such as the k-fold, Repeated K-Fold, Shuffle Split, Stratified K-Fold and Stratified Shuffle Split methods. Repeated K-Fold and Shuffle Split seem to be the methods that most penalize the models' average accuracy, therefore any of these two can be used for model benchmarking.

ii. Baseline Models Performance

The first result worth mentioning is from our first official trial. For this we used our baseline attempt, which was quite successful as its sixteen models generally resulted in high accuracy scores. Models using a Decision Tree Classifier, accumulating a f1-score of 1.00, and using a Random Forest and an Extra Trees Classifier, both with a f1 score of 0.980769, were the best performers for the baseline attempt. Tree based and ensemble boosting models have better performance with raw data, whereas the remaining models (using Linear Regression, KNN, Bagging, SVC, Gaussian Naïve Bayes, Neural Networks and Stochastic Gradient Descent Classifiers) perform better with Normalized Data. The results of all the models are summarized in Figure 3 (Attachments).

iii. Hyperparameter Tuning

Model hyperparameter tuning, the process of determining the right combination of hyperparameters that maximizes the model performance, should be performed in order to ensure good results from our machine learning model. There are three different approaches for hyperparameter tuning methods - Grid Search, Random Search, and Bayesian Optimization. While Grid Search is computationally expensive, it is also more effective rather than Random. For models with a high number of parameter possibilities we have chosen to implement Random Search (and Grid Search otherwise).

Grid search is a technique for hyperparameter optimization that defines hyperparameter values, training a model for each combination, evaluating each model performance and selecting the best

performing model. Random Search, in contrast to Grid Search, not all parameter values are tried out, but rather a fixed number of parameter settings is sampled from the specified distributions. Random Search can be less reliable than Grid Search because it relies on chance to sample combinations of hyperparameters. For models with a high number of parameter possibilities it'll be implemented Random Search and Grid Search otherwise.

iv. Tuned Models Performance

The hyperparameter tuning method chosen included both Grid and Random Search, whereas cross-validation for Grid Search was performed with 5 splits and cross-validation for Random Search was performed with 10 different splits.

As it was expected, the results for tuned models were slightly better in comparison with baseline models. After assessing tuned scores, we concluded that KNN Classifier, SVC and Ada Boost Classifier performed significantly better. In contrast, the Logistic Regression Classifier got slightly worse after hyperparameter tuning. Lastly, it was visible, as it was for baseline models, that Tree-based & Ensemble algorithms have better performances than the remaining algorithms.The results for algorithms using a tuned approach are represented in Figure 4 (Attachments).

v. Manual Tuning

Random Search and Grid Search are computationally expensive, therefore testing parameters manually after some input from these tuning methods can bring better results. Different parameters were tested on some of what we considered to be the best models. In order to check parameter performance, Confusion Matrices were built on models that were revealed the best f1-score performance.

Random Forest

Another result worth stating was the final model using a Gradient Boosting Classifier, which out of all the Non Diseased Patients, 48.73% were predicted correctly, while out of the Diseased, 50.63% were predicted correctly. For this algorithm, we observed a f1-score of 0.981 for our baseline model, followed by 0.974 for the tuned model and, finally, 0.994 for the model manually tuned. Furthermore, this model had one false positive observation, which explains the reasoning beyond a worse recall value compared to the models performed with either a Histogram-based Gradient Boosting Classifier or a Gradient Boosting Classifier. The results for the Confusion Matrix and Metrics are represented in Figure 6 and 7.

Gradient Boosting Classifier

The results obtained proved to be exactly the same to the model performed using a Histogram-based Gradient Boosting Classifier. For this algorithm, we observed a f1-score of 0.955 for our baseline model, followed by 0.994 for the tuned model and, finally, 1.000 for the model manually tuned. Both this and Histogram-based Gradient Boosting Classifier achieved an f1-score of 1.000 which makes sense considering none accounted for any false positives or false negatives, predicting correctly their extent of observations. On Figures 8 and 9 we are able to see the confusion matrix for these model and its metrics in this approach.

Histogram-based Gradient Boosting Classifier

Out of all the Non Diseased Patients, 48.73% were predicted correctly, while out of the Diseased, 51.27% were predicted correctly. For this algorithm, we observed a f1-score of 0.987 for our baseline model, followed by 0.994 for the tuned model and, finally, 1.000 for the model manually tuned. The results are in Figure 10 and 11 (Attachments).

v. Model Final Selection

At this stage, the four best performing models were tested considering Cross-Validation and ROC AUC. As tested previously, the Cross-Validation methods that penalize the models the most are the Repeated K-Fold and Shuffle Split, and consequently these cross-validation techniques will be used going forward. Repeated K-Fold cross validation repeats K-Fold n times with different randomization in each repetition. Shuffle Split splits the dataset randomly into training and validation subsets. Contrary to other cross-validation strategies, random splits do not guarantee that all folds will be different, although this is still very likely for sizable datasets. This cross-validation object is a variation of KFold that returns stratified folds. The folds are made by preserving the percentage of samples for each class.

Manually Tuned Histogram-Based Gradient Boosting, Manually Tuned Gradient Boosting, Base Extreme Gradient Boosting and Manually tuned Random Forest seem to be the best performing models with F1-Scores higher than 0.99 and therefore, will be the ones tested for model selection. The best results of both cross-validation methods belong to the Histogram-Based Gradient Boosting model that averaged high scores even on 10-Fold splits, despite being trained with less and not stratified data. Using this algorithm, we obtained a F1-Score of 0.973 +/- 0.02 for Repeated K-Fold Test, and obtained a F1-Score of 0.963 +/- 0.02 for Shuffle Split.

ROC AUC, which stands for Receiver Operating Characteristic and area under ROC curve, is an evaluation method for checking binary classification problems performance. This metric doesn’t depend on the classification threshold or the scale of probabilities. According to the AUC - ROC curve, Histogram-Based Gradient Boosting Classifier and Gradient Boosting Classifier shared an ideal measure of separability, meaning both of them are perfectly able to distinguish between positive and negative classes. These results are proven in figure 12 (Attachments).

Histogram-Based Gradient Boosting Classifier manually tuned showed better overall performance during the model selection process, therefore it'll be the model used to predict whether a Patient will suffer, or not, from the Smith Disease.

vi. Predictions Data Analysis

SHAP values method, which stands for Shapley Additive Explanations, and PDP, short for Partial Dependence Plots, were used to explore the results. SHAP values is a method based on cooperative game theory and used to increase transparency and interpretability of machine learning models.

Based on SHAP values, the conclusion is patients that haven't got a check-up in the last 3 years, that consume less than 1 fruit everyday, that have contact with Diabetes (directly or indirectly), who declared low physical health, don't exercise more than 30 minutes 3 times per week or who usually consumes alcohol everyday, have high positive contribution on the prediction and vice-versa. The features with higher impact on the predictions are Check Up, Fruit Habit, Diabetes and Physical Health.

Scikit-learn websites, in their documentation on Partial Dependence Plots, notes:

> Partial dependence plots (PDP) show the dependence between the target response and a set of input features of interest, marginalizing over the values of all other input features (the ‘complement’ features). Intuitively, we can interpret the partial dependence as the expected target response as a function of the input features of interest.

As for the interpretation of Partial dependence plots, having a check up done more than 3 years ago increased the odds of having Smith's Disease. People who reported higher Physical Health had better odds of having the disease. We reported both results from SHAP values and Partial dependence plots in Figures 14, 15 and 16 (see in Attachments).

Conclusion

The main objective of this project was to build a predictive model, aimed at tackling a classification problem, that would answer the question "Who are the groups of individuals more likely to suffer from the Smith Parasite?". In this context, the following steps were conducted: Data Info and Summary Statistics, Data Quality Verification and Exploratory Data Analysis to explore and understand the dataset, Feature Engineering and Selection, build a predictive model using machine learning techniques and deployment.

Outliers were manually excluded, only being considered 98.25 % of its original observations, and, as for missing values, they were filled in, through an imputation with the mode after the Data Split. Regarding Feature Engineering, 2 new variables were created, in order to increase readability or reduce the number of variables. Going forward, for data normalization, the MinMax [-1,1] had the best score out of the four variations compared, presenting no signs of overfitting. As for Feature Selection, multiple techniques were performed to identify which variables to keep and which to discard.

Regarding the algorithms used, we can conclude that the best performing predictive models were the manually tuned Histogram-Based Gradient Boosting, followed by manually tuned Gradient Boosting, baseline Extreme Gradient Boosting and manually tuned Random Forest. Histogram-Based Gradient Boosting Classifier manually tuned showed better overall performance during the model selection process, therefore it'll be the model used to predict whether a Patient will suffer, or not, from the Smith Disease. The baseline model, without normalization, provided a F1-Score of 0.987, while with Random Search Tuning, it provided a F1-Score of 0.993. On the manually tuned model however, it provided a F1-Score of 1.000, after Random Search tuning. Finally, it is only left to mention that 51.10% of the unchecked 225 Patients are predicted to suffer from Smith's Disease.

Moreover, considering SHAP value, the horizontal position of the data point in the plot indicates if the feature has a high impact in the prediction. We could conclude that having done a checkup more than 3 years ago, not eating fruit everyday and not having contact with diabetes were the most impactful features in order to make a prediction on whether a Patient had Smith's Disease or not. As for having done a checkup more than 3 years ago, as this value increases, the model's prediction moves to higher values. As for Physical Health, as this value increases, the model's predictions move to lower values. However, the trend drops slightly until 7, while after suffers from an accentuated decrease until 10.

References

Brownlee, J. (2020). accessed 19 December 2022, https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

Brownlee, J. (2020). accessed 19 December 2022, https://machinelearningmastery.com/hyperparameter-optimization-with-random-search-and-grid-search/

Hyperparameter Optimization in Machine Learning Models. accesses 23 December 2022, https://www.datacamp.com/tutorial/parameter-optimization-machine-learning-models

Wirth, R., & Hipp, J. (2000). CRISP-DM: Towards a Standard Process Model for Data Mining

Scikit-learn: Machine Learning in Python, Pedregosa, et al., JMLR 12, pp. 2825-2830, 2011

Feature Selection using Filter Methods (2021). accessed 16 December 2022, https://www.linkedin.com/pulse/feature-selection-using-filter-methods-runa-veigas/

Pedro Alceo, R. Henriques (2019). Sports Analytics: Maximizing Precision in Predicting MLB Base Hits

M. Luckert, Moritz Schaefer-Kehnert (2016). Using Machine Learning Methods for Evaluating the Quality of Technical Documents

Gajawada, Sampath K. (2019). Chi-Square Test for Feature Selection in Machine learning. accessed 20 December, https://towardsdatascience.com/chi-square-test-for-feature-selection-inmachine-learning-206b1f0b8223

The Fundamental Differences Of Pearson Correlation, Spearman Rank, Kendall Tau, And Chi-Square (2022). accessed at 21 December 2022, https://kandadata.com/the-fundamental-differences-of-pearson-correlation-spearman-rank-kendall-tau-and-chi-square/

Attachments

1. Figures

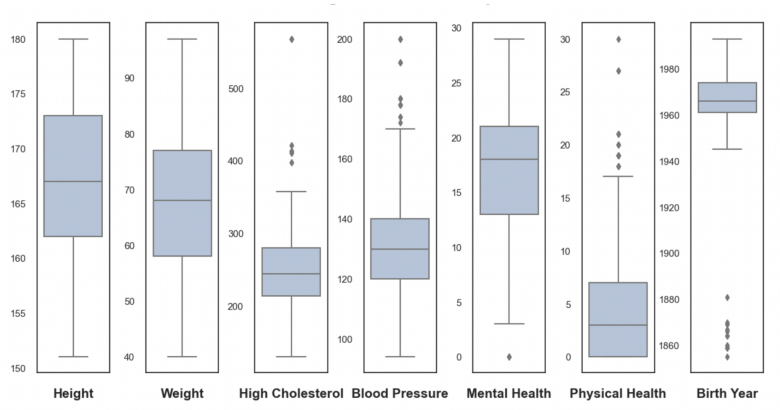


Figure 1: Checking Outliers with Boxplots

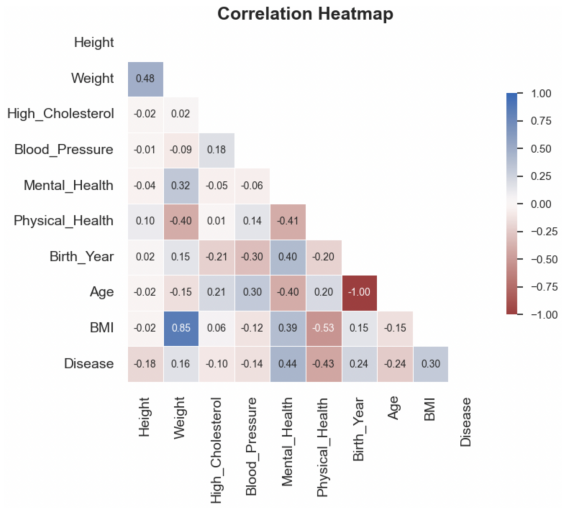


Figure 2: Redundancy Correlation Heatmap

Model	F1-Score	Accuracy	Precision	Recall	AUC
X G B Classifier	0.993671	0.993671	0.993671	0.993671	0.993827
Decision Tree Classifier	0.987342	0.987342	0.987342	0.987342	0.987654
Hist Gradient Boosting Classifier	0.987342	0.987342	0.987342	0.987342	0.987334
Random Forest Classifier	0.981013	0.981013	0.981013	0.981013	0.981161
Extra Trees Classifier	0.981013	0.981013	0.981013	0.981013	0.980840
Gradient Boosting Classifier	0.955696	0.955696	0.955696	0.955696	0.955187
Bagging Classifier	0.949367	0.949367	0.949367	0.949367	0.950617
Stacking Classifier	0.949367	0.949367	0.949367	0.949367	0.949335
Voting Classifier	0.905063	0.905063	0.905063	0.905063	0.906125
Ada Boost Classifier	0.898734	0.898734	0.898734	0.898734	0.899311
M L P Classifier	0.892405	0.892405	0.892405	0.892405	0.893458
Logistic Regression	0.873418	0.873418	0.873418	0.873418	0.873657
K Neighbors Classifier	0.873418	0.873418	0.873418	0.873418	0.874940
S V C	0.867089	0.867089	0.867089	0.867089	0.867484
S G D Classifier	0.867089	0.867089	0.867089	0.867089	0.867484
Gaussian N B	0.803797	0.803797	0.803797	0.803797	0.803832

Figure 3: Baseline Models Performance

Model	F1-Score	Accuracy	Precision	Recall	AUC
Hist Gradient Boosting Classifier	0.993671	0.993671	0.993671	0.993671	0.993827
X G B Classifier	0.993671	0.993671	0.993671	0.993671	0.993827
Decision Tree Classifier	0.987342	0.987342	0.987342	0.987342	0.987334
Extra Trees Classifier	0.981013	0.981013	0.981013	0.981013	0.980840
Ada Boost Classifier	0.981013	0.981013	0.981013	0.981013	0.980840
Random Forest Classifier	0.974684	0.974684	0.974684	0.974684	0.974347
Gradient Boosting Classifier	0.974684	0.974684	0.974684	0.974684	0.974347
K Neighbors Classifier	0.968354	0.968354	0.968354	0.968354	0.967853
S V C	0.968354	0.968354	0.968354	0.968354	0.968815
Stacking Classifier	0.968354	0.968354	0.968354	0.968354	0.968174
M L P Classifier	0.962025	0.962025	0.962025	0.962025	0.962322
Bagging Classifier	0.955696	0.955696	0.955696	0.955696	0.955507
Voting Classifier	0.905063	0.905063	0.905063	0.905063	0.906125
Logistic Regression	0.860759	0.860759	0.860759	0.860759	0.861312
Gaussian N B	0.860759	0.860759	0.860759	0.860759	0.860350
S G D Classifier	0.860759	0.860759	0.860759	0.860759	0.860991

Figure 4: Random and Grid Search Tuning Performance

Model	Best Version	F1-Score
Hist Gradient Boosting Classifier	Tuned	1.000000
Gradient Boosting Classifier	Tuned	1.000000
X G B Classifier	Base	0.993671
Random Forest Classifier	Tuned	0.993671
Decision Tree Classifier	Base	0.987342
Extra Trees Classifier	Base	0.981013
Ada Boost Classifier	Tuned	0.981013
Stacking Classifier	Tuned	0.968354
K Neighbors Classifier	Tuned	0.968354
S V C	Tuned	0.968354
M L P Classifier	Tuned	0.962025
Bagging Classifier	Tuned	0.955696
Voting Classifier	Base	0.905063
Logistic Regression	Base	0.873418
S G D Classifier	Base	0.867089
Gaussian N B	Tuned	0.860759

Figure 5: Models best performances and comparison

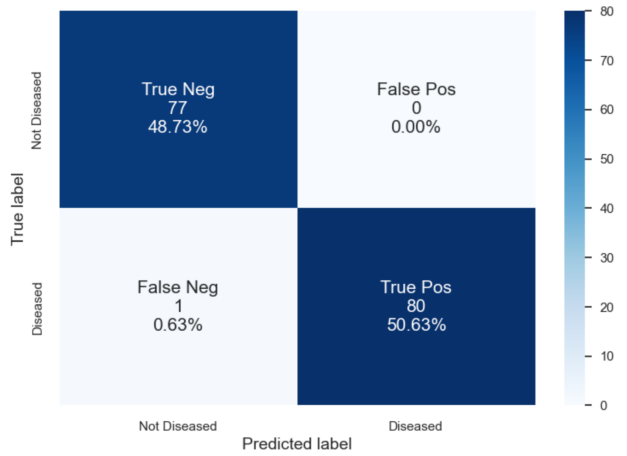


Figure 6: Manually tuned Random Forest Classifier Confusion Matrix

	precision	recall	f1-score	support
Not Diseased	0.99	1.00	0.99	77
Diseased	1.00	0.99	0.99	81
accuracy			0.99	158
macro avg	0.99	0.99	0.99	158
weighted avg	0.99	0.99	0.99	158

Figure 7: Manually tuned Random Forest Classifier metrics

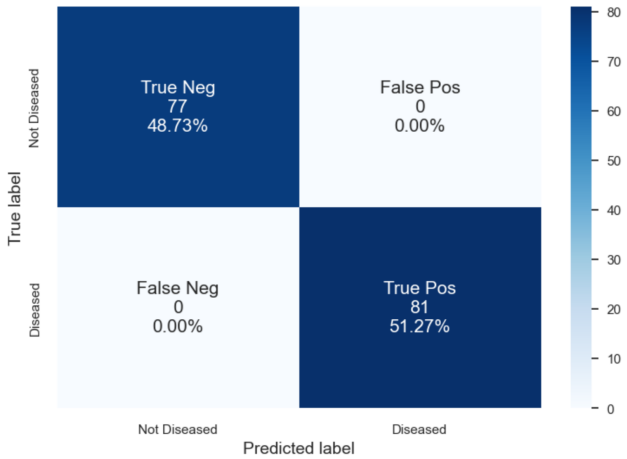


Figure 8: Manually tuned Gradient Boosting Classifier Confusion Matrix

	precision	recall	f1-score	support
Not Diseased	1.00	1.00	1.00	77
Diseased	1.00	1.00	1.00	81
accuracy			1.00	158
macro avg	1.00	1.00	1.00	158
weighted avg	1.00	1.00	1.00	158

Figure 9: Manually tuned Gradient Boosting Classifier Metrics

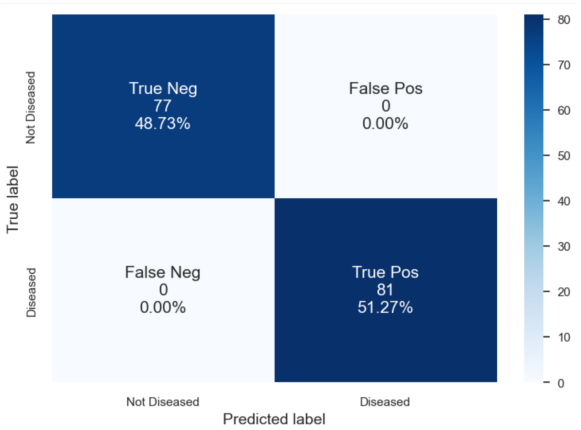


Figure 10: Manually tuned Histogram-based Gradient Boosting Classifier Confusion Matrix

	precision	recall	f1-score	support
Not Diseased	1.00	1.00	1.00	77
Diseased	1.00	1.00	1.00	81
accuracy			1.00	158
macro avg	1.00	1.00	1.00	158
weighted avg	1.00	1.00	1.00	158

Figure 11: Manually tuned Histogram-based Gradient Boosting Classifier Metrics

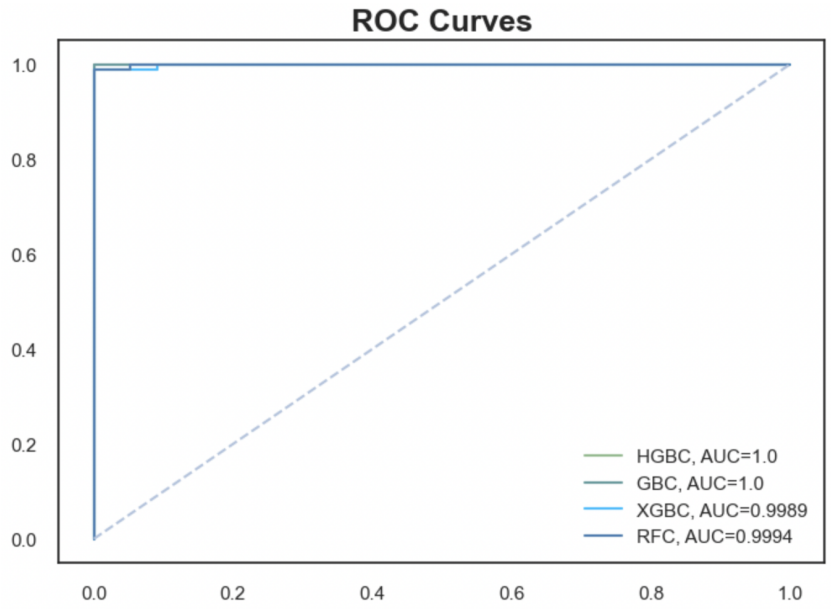


Figure 12: AUC-ROC Curve

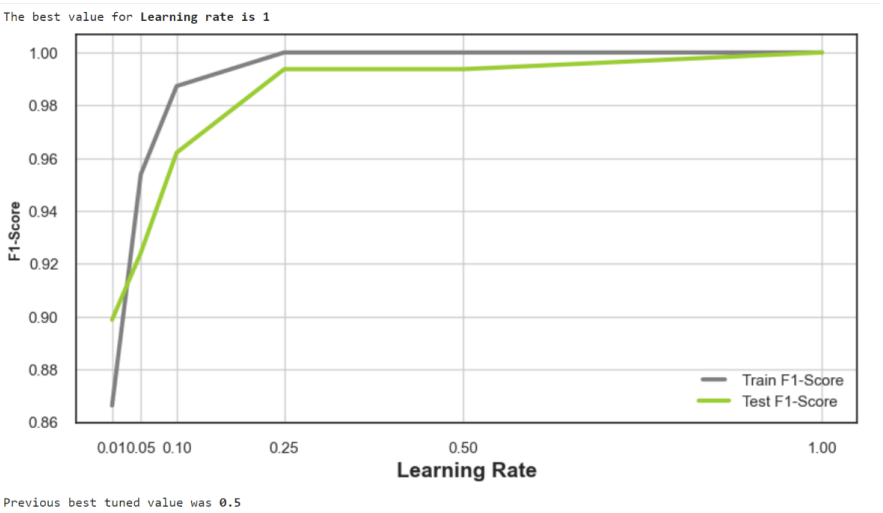


Figure 13: Manually tuned learning rate for Gradient Boosting Classifier

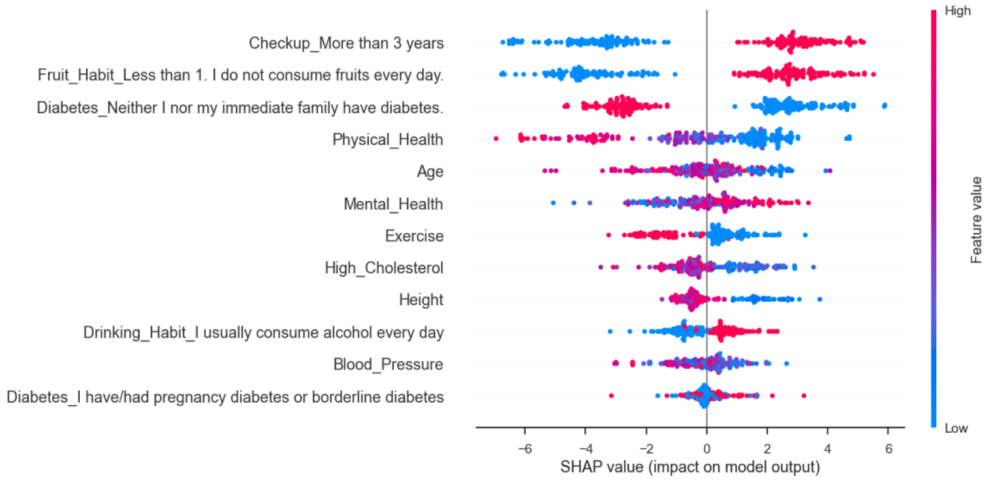


Figure 14: Plot for SHAP value

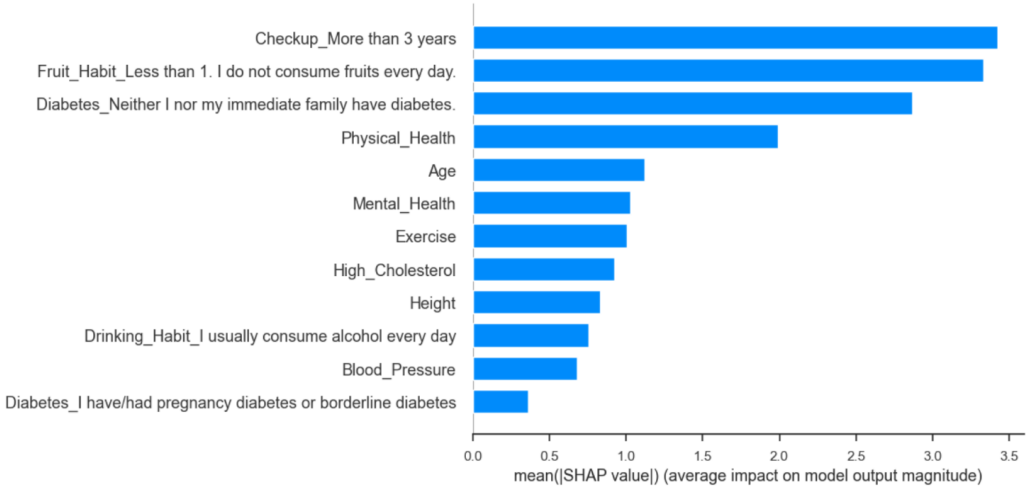


Figure 15: Plot for mean importance SHAP values

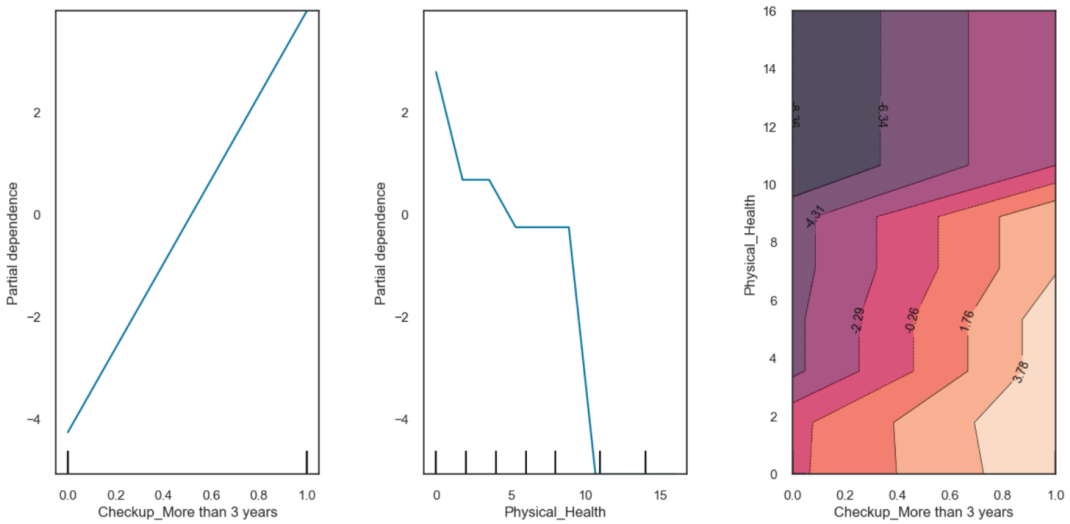


Figure 16: Partial Dependence Plot

2. Explanation of top 4 models and chosen hyperparameters

Random Forest

A random forest is an ensemble machine learning algorithm. Random Forest uses uses multiple decision trees together applied to different subsets, and then combines their predictions.

The Random Forest algorithm is applied using the RandomForestClassifier class from the sklearn. The code uses Random search with cross-validation to tune the hyperparameters of the random forest in order to improve its performance.

The hyperparameters arrived at with the grid search are:

- n_estimators:. The model uses 600 decision trees.
- 'criterion': 'entropy' - This specifies the criteria used to split the data by entropy.
- 'max_depth': 50 - This specifies the maximum depth of each decision tree in the random forest, which determines how many layers of nodes the trees can have.
- 'bootstrap': False - This specifies that the bootstrapping is not used to sample the data for each tree in the forest.
- 'oob_score': False - This specifies that the out-of-bag samples are not used to estimate the generalization error of the model.
- 'warm_start': False - This specifies that the previous state of the model is not employed in order to speed up training.
- 'max_features': 5 - This specifies that 5 are the number of features to consider when looking for the best split at each node of each tree in the forest. *
- 'min_samples_split': 5 - This specifies that 5 is the minimum number of samples required to split a node in each tree of the forest.
- 'min_samples_leaf': 1- This specifies that 1 is the minimum number of samples required to be at a leaf node in each tree of the forest.
- 'class_weight': 'balanced' - The "balanced" mode in Random Forests adjusts the weights of each class inversely proportional to their frequencies in the input data, with the expression: n_samples / (n_classes * np.bincount(y))

* Best values obtained through manual tunning

Gradient Boosting

The Gradient boosting classifier is an ensemble machine learning algorithm that uses multiple weak learners to make predictions. It works by training a weak learner on the residual errors of the previous weak learner, and then combining the predictions of all of the weak learners using a weighted average.

The tuning is done by Random search to arrive at the adequate hyperparameters. A confusion matrix and a classification report are also created.

The hyperparameters chosen are:

- n_estimators:. The model uses 400 decision trees.
- max_features: 1 – The number of features considered when looking for the adequate split at each node in the Decision trees.
- learning_rate: 1 – Rate of learning from each decision tree. *
- max_depth: 50 – Defines the maximum depth of the decision trees in the model.
- min_samples_split: 0.1 - Determines the minimum number of samples required to split an internal node in the decision trees. *
- min_samples_leaf: 0.01 - determines the minimum number of samples required to be at a leaf node in the decision trees. *

* Best values obtained through manual tunning

HGBmodel

A HGBmodel is a type of machine learning model called a Histogram-based Gradient Boosting Classifier. It is a type of ensemble model, which means that it combines the predictions of multiple weaker models.

The hyperparameters for the model are:

- min_samples_leaf: 0.01 - determines the minimum number of samples required to be at a leaf node in the decision trees.
- max_leaf_nodes: 35 - Maximum number of leaf nodes using a best-first search
- max_iter: 200 - The maximum number of binary trees.
- max_depth: 45 - The maximum number of levels in the tree.

XGBOOST

The XGBoost (eXtreme Gradient Boosting) Classifier is a decision tree-based ensemble machine learning algorithm that uses boosting techniques for performance, specially used in datasets that are imbalanced or with many missing values.

The default hyperparameters were chosen for the XGBoost model, these are:

- booster: 'gbtree' - The type of booster to use, 'gbtree' is used.
- tree_method: ‘auto’- The method to use for building the trees.
- max_depth: 6 - The maximum depth of each decision tree in the ensemble.
- eta: 0.3 The step size shrinkage used in each boosting step.