Machine Learning models for risk of diabetes prediction

Alexandru DAMIAN, Erwan DEHILLERIN, Kenza MAKHLOUF, Apolline PIGNATO, Jean-Luc WOLKEN

Abstract

The project aims to build a model for diabetes prediction. The data set is first explored, the section 3 then gathers information found in the literature about different diabetes data sets. Section 4 describes the proposed method and the workflow used to answer the research question. Section 5 provides an overview of the experiments performed on the selected models. Finally, section 6 discusses the results from the experiments.

Previous papers have tackled this problem. The existing literature exploring diabetes and machine learning agrees to say that lifestyle plays a key role in diabetes, as also socioeconomic factors, but of lesser importance. Our goal is to look into this dataset to find out the most relevant factors and enhance diabetes prediction. Several methods were applied for the features selection, namely F-test, tree-based methods, RFE and Mutual Information, which had similar results in feature importance ranking. The following subset of features was considered as being the most relevant: general health, blood pressure, BMI and cholesterol.

To tackle the classification task, the following models have been used: SVM, eXtreme Gradient Boosting (XGBoost), and a neural network (NN) architecture. Furthermore, multiple values for hyperparameters and probability thresholds have been explored. XGBoost and NN were able to achieve the best results on this data set with an accuracy of 75.4% and 75.3%, respectively. For the neural network, LIME was used in order to increase explainability of the results.

1. Introduction

According to the World Health Organization, "Diabetes is a chronic disease that occurs either when the pancreas does not produce enough insulin (a hormone that regulates blood sugar) or when the body cannot effectively use the insulin it produces". This disease can be either a birth disease (type 1) or caught for 95% of the patients (type 2), mainly caused by a sedentary lifestyle and unhealthy diet. In 2014, 8.5% of adults older than 18 had diabetes. In 2019, diabetes was the direct cause of 1.5 million deaths, thus making this disease the ninth leading cause of death. [1]

The International Diabetes Organization states that in 2021, 537 million people worldwide from 20 to 79 lived with diabetes and this number increases quickly (+100 million cases per decade) [2]. It causes trouble to the heart, blood vessels, eyes, kidneys, and nerves. Then diabetes has raised major concerns since the last decades. Not only is this chronic disease a major public health problem but it also comes with a high cost[1]. That's why it is even more important to detect this disease at an early stage.

Email addresses: aldam21@student.sdu.dk (Alexandru DAMIAN), erdeh22@student.sdu.dk (Erwan DEHILLERIN), kemak22@student.sdu.dk (Kenza MAKHLOUF), appig22@student.sdu.dk (Apolline PIGNATO), jewol22@student.sdu.dk (Jean-Luc WOLKEN)

[1] World Health Organization, https://www.who.int/news-room/fact-sheets/detail/diabetes

[2] International Diabetes Organization, https://www.idf.org/aboutdiabetes/what-is-diabetes/facts-figures.html

In this context, a study was carried out on the Diabetes Health indicator dataset[3] to answer the following research questions: What are the most relevant risk factors to diabetes and how to predict diabetes with the best accuracy? This paper investigates the results of three machine learning models: Support Vector Machine (SVM), eXtreme Gradient Boosting (XGBoost) and Neural Network model (NN).

2. Description of the data set

The Diabetes Health Indicators data set used in this study was created after a survey from the Center for Disease Control and Prevention in 2015 and is available on Kaggle. It aims at predicting whether a person is diabetic or not considering different risk factors which can be regrouped into three categories, as shown in Table 2.

Medical factors	Lifestyle factors	Socio-economic factors
High blood pressure	Smoker	Sex
High cholesterol	Physical activity	Education
Cholesterol check, BMI, Stroke	Fruits, Veggies	Age
Heart diseasor attack	Heavy alcohol consumption	Income
Mental health, General health		Any health care
Physical health, Difficulty walking		No Doctor because of cost

Table 1: Categories of the features.

The data set originally contained 300 features from the questions asked during the survey, 21 features were selected based on a research on diabetes type 2 [2] analysing the questions from this survey in 2014. The data set was then undersampled and the final data set is balanced and clean with 21 features and 70692 observations.

The heat map plotted in Figure 1 highlights the correlation between the features. The conclusion that results from this plot is the following :

- Diabetes_binary is highly positively correlated with general health, high BP, high cholesterol, BMI and age.
- Diabetes_binary is highly negatively correlated with income, education and physical activity.
- There is a high correlation between GenHlth, PhysHlth and MentHlth.

[3]Diabetes Health indicator dataset `https://www.kaggle.com/datasets/alexteboul/diabetes-health-indicators-dataset?select=diabetes_012_health_indicators_BRFSS2015.csv`

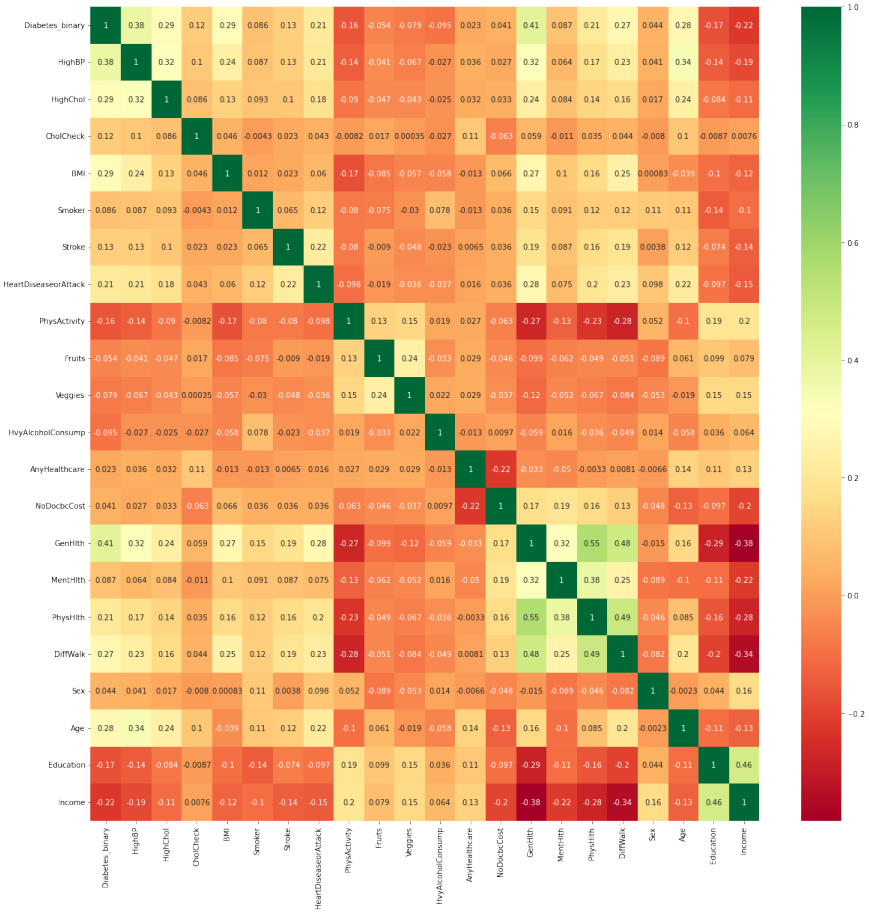


Figure 1: Correlation matrix of the dataset

To get a better overview of the distribution of the data, a distribution plot was made in Figure 2. It is already possible to predict that some features won't be correlated with the target feature because their distribution is very imbalanced contrary to the label distribution.

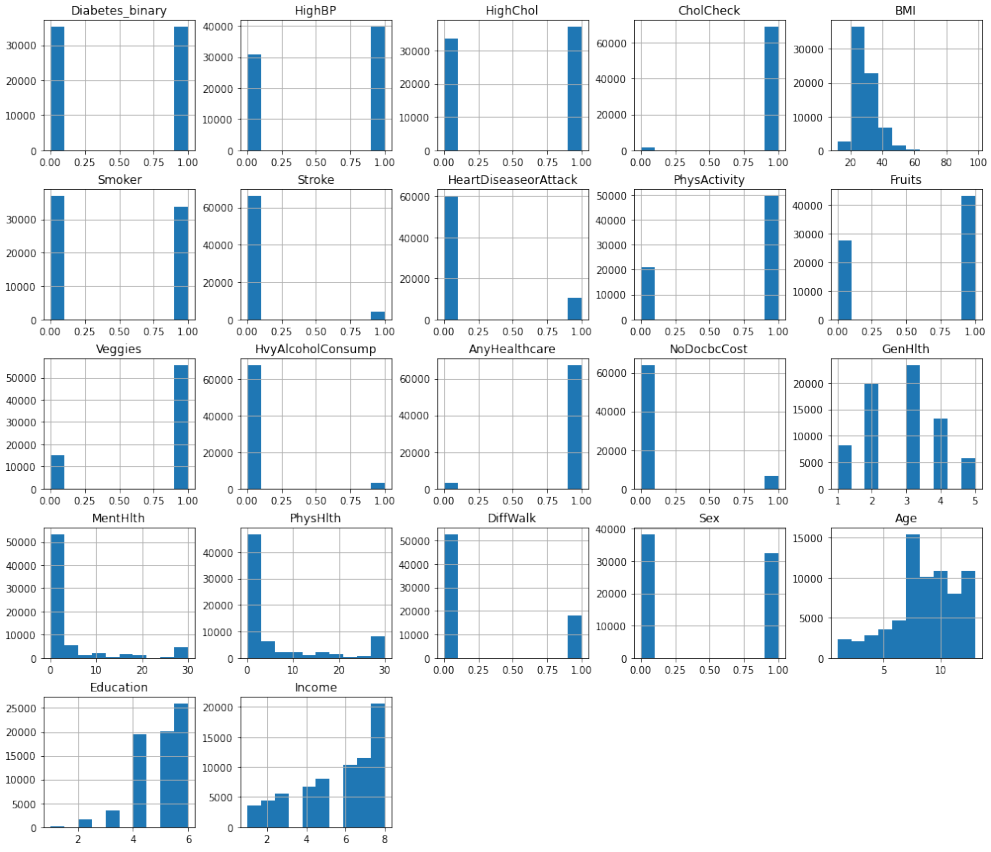


Figure 2: Distribution of the features

3. Related Work

3.1. Feature selection

A literature review was performed to determine which risk factors are relevant in other notebooks dealing with the prediction of diabetes.

Considering that there are different types of features in the data set (socio-economic, medical and lifestyle features), one of the purposes of this review was to find out if it is necessary to include all the categories. An analysis conducted on 90 papers concluded that " some data types produce better models, such as lifestyle, socioeconomic and diagnostic data"[3], which suggests that all three categories should be included in our analysis.

A notebook on the same data set studied the feature importance using the XGBoost model and showed that the most significant ones were Age, General Health, High Blood pressure and BMI [4].

Simran Gill and Prathmesh Pathwar worked on the Vanderbilt dataset and tried different feature selection methods to find the most important features among the 18 features of the data set. They concluded that Cholesterol, Glucose, High-density lipoprotein cholesterol (Chol/HDL), Systolic BP, and Weight are the most significant ones[5].

3.2. Classification task

Multiple studies have attempted to implement Machine Learning models for the type 2 diabetes classification task. Few of the widely used methods published were support vector machine (SVM), K-Nearest Neighbor (KNN), Random Forest (RF), Neural Networks (DNN), Naïve Bayes (NB) and Decision Trees

(DT) [6, 7, 8, 9]. Ensemble-based Machine Learning techniques were also implemented. For instance, Taz et al. [10] explored four individual models: Light Gradient Boosting Machine (LightGBM), XGBoost, Adaptive Boosting (AdaBoost) and Random Forest and implemented a soft voting ensemble classifier that enhanced the individual performance metrics.

Most of the papers published used the Pima Indian data [11] for their experiments, the data in this paper was used previously for diabetes classification mission as well, with some of the aforementioned techniques and the works are available as notebooks on Kaggle ML Repository. The proposed methods also implemented various preprocessing techniques. A Neural Network approach with all the features scaled with MinMax scaler [12] achieved an accuracy of 74.6%, and a second Neural Network architecture with only 6 selected features [13] achieved 74.8%. A voting ensemble technique based on KNN classifiers with 23 neighbours and a standard scaler [14] for preprocessing had a performance of 74.1%. Finally, the highest score was obtained with XGBoost [15] with an accuracy of 75.4%.

A comparison table between these approaches and our proposed models is presented in section 5.

4. Proposed Method

The process of this project followed the workflow presented in the figure 3. After converting the data set into categorical features, it was split into three parts: training, validation and test. The training and testing ratio was 75:25, the validation set was taken from the training set before any experiment, which gave a total ratio of 67.5:7.5:25. The training set was used for the experiments with the Grid search cross-validation to get the best parameters for the models whereas the validation was used to adjust the threshold to this problem. The purpose of using a validation set for this task was to avoid any over-fitting and to keep the test set independent from the experiments.

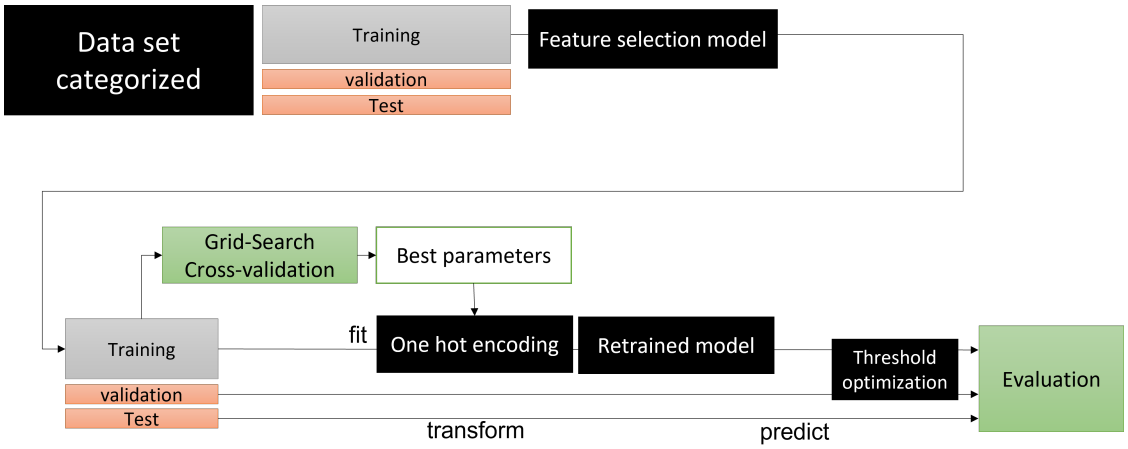


Figure 3: Project workflow

4.1. Preprocessing

The dataset didn't require much preprocessing, given that it's balanced and there are no missing values. For preprocessing, the boxplot diagram was also considered (figure 4). The values within the box are between the upper quantile (75% of the values) and the lower quantile (25% of the values). The line to the outer bars (whiskers) is 1.5 times the length of the box. Any values that fall outside the whiskers are called outliers and can cause experimental errors. In addition, these can cause weaker model performance. For the numerical feature BMI, a total of 79 different values were recorded and a large portion of them was identified as outliers. To address this, BMI was converted to a categorical feature by dividing the values into seven different groups. The now purely categorical data set was then one hot encoded so that all values were binary.

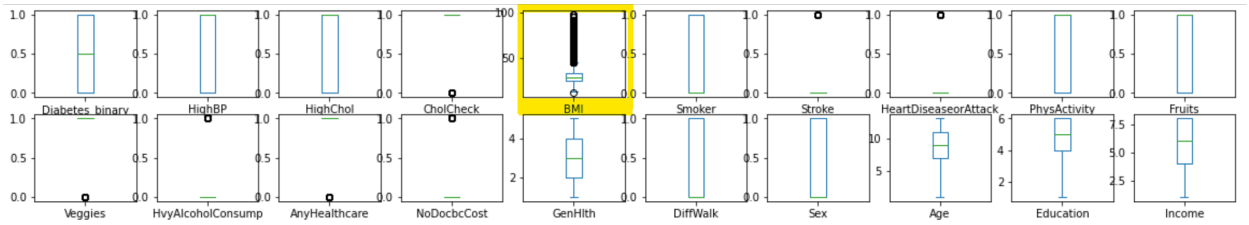


Figure 4: Boxplot of the features

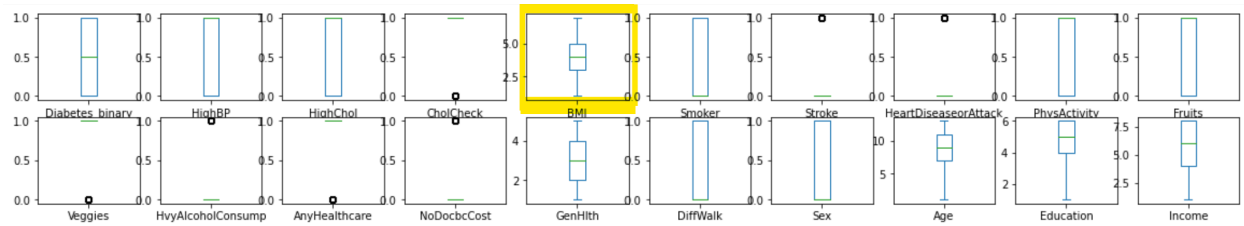


Figure 5: Boxplot with categorized BMI

4.2. Feature selection models

Regarding the feature selection, two common methods were first used: F-tests and trees (more precisely decision tree, extra tree and random forest). Those tests were subjected to raw and preprocessed data in order to compare the results. Then we also wanted to understand the weight of the 6 socioeconomic features (no healthcare, no doctor because of costs, sex, age, income, education) to make sure that they didn't add bias in the tests as it may be the case in Machine Learning works, therefore those 6 features were deleted and a comparison of all the results was performed.

Two conclusions can be drawn from this comparison. First, the categorization of the feature 'BMI' only modified the weight of the first 3 features (about cholesterol and blood pressure), whose selection or not can be decided with an average weight or thanks to the literature. The importance of the BMI is squashed by this transformation but BMI is relevant so we decided to keep it. Second, the deletion of the 6 socioeconomic features doesn't impact the relative weight of the other features. That's why the 4 variants of the test will only be shown once to make this paper more readable : those conclusions are illustrated with the F-test in Appendix .2.

So we already have a subset of relevant features, namely :

- General health
- High blood pressure
- BMI
- High cholesterol
- Age
- Difficulty to walk

The second proposed method, based on trees, doesn't require any transformation. Decision tree is the basic tree-based method used for ExtraTree and RandomForest by building only one tree hence a high variance. All of those methods use random subsets of features and samples. While RandomForest uses

bootstrapping and split nodes at the best split, ExtraTree decreases variance by splitting nodes at random splits and also sampling without replacing them in the bootstrapping process. Some randomness is moved from bootstrapping to node splitting. ([16])

It can be noticed in figure .15 that those 3 variants of trees give very close results to each other and are consistent with the F-test method. Then it yields a new subset of selected features, broadened from the previous one with the following features:

- Mental health
- Physical health
- Education
- Income

RFE (Recursive Feature Selection) and Mutual Information were used and both of these methods confirm the relevancy of our subset. The RFE recursively removes features and then builds a model on the remained features. It can be inferred from those models the features which provide the best predictions. It's a wrapper method, suitable for both classification and regression problems. The output of RFE is only a list of the selected features; to sort the list and thus get a ranking, the algorithm was applied to select successively 1 to 11 features (11 was arbitrarily chosen to have a wide enough overview of the ranking). Mutual Information is a filter method which measures the dependence between two features and gives the amount of information we can get from the one given to the other one. The value is between 0 and 1; the larger the value, the greater the relationship between the selected feature and the target. It works in a similar way to the trees. ([17])

The feature "cholesterol check" had a good score but is highly correlated to "high cholesterol" so it was dropped. Since the features "general health", "mental health", and "physical health" are highly correlated with each other according to the matrix, we only chose the most relevant of them which is "general health". Besides, some of the 6 socioeconomic features can be dropped since they are not correlated enough with diabetes.

Finally, the most relevant features are **"high blood pressure", "high cholesterol", "BMI", and "general health"** and **"difficulty to walk", "heart disease", "age"** in a lesser importance.

This step was relevant to put forward some criteria but the methods which have been applied afterwards didn't require feature selection. Furthermore, some results didn't match with well-known diabetes factors, maybe because of the data set itself.

4.3. Classification models

Given the presented goals of this analysis and the nature of the data set, the following machine learning models have been proposed, their configurations were explored during the experiments:

- **Support Vector Machine**

  SVM is a supervised learning method that looks at data and sorts it into one of two categories. Regarding classification, SVM's goal is to find an optimal hyperplane which acts as a decision boundary that differentiates between the classes. Since the data set used in this analysis consists of 21 features, SVM should work well for classification because the algorithm works well in a high dimensional space. SVM is robust to noisy features compared to other algorithms, but having uninformative features will impact the classifier's performance.

The major drawback of using SVM is the limited use cases because the algorithm's training complexity is highly dependent on the size of the data set. Having 70692 observations in the data set means that it is fairly large and to achieve the goals of the analysis, additional methods will be explored.
In order to implement SVM, the sklearn.svm.SVC module from sci-kit learn library will be used. Scikit-learn is largely written in Python and uses NumPy extensively for high-performance linear algebra and array operations. Furthermore, some core algorithms are written in Cython to improve performance. Support vector machines are implemented by a Cython wrapper around LIBSVM. The fit time scales at least quadratically with the number of samples and may be impractical beyond tens of thousands of samples. [4]

- **XGBoost**

XGBoost is an algorithm that focuses on computational speed and model performance. XGBoost dominates structured or tabular data sets on classification and regression predictive modelling problems, and it is the go-to algorithm for competition winners on the Kaggle competitive data science platform and won many challenges. [5]
XGBoost can easily handle large data sets and doesn't require much if any feature engineering. The results of the algorithm are easy to interpret since visualizing the trees offers good explainability. The challenge regarding this approach is tuning the hyper-parameters to avoid over-fitting.
XGBoost will be implemented in python using the library with the same name. XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solves many data science problems in a fast and accurate way.
Boosting is an ensemble method, meaning it's a way of combining predictions from several models into one. It does that by taking each predictor sequentially and modelling it based on its predecessor's error (e.g giving more weight to predictors that perform better). [6] This model was implemented as a XGBClassifier() module with xgb library in python, with an early_stopping_rounds of 10, and an evaluation metric (eval_metric = 'aucpr'), the hyper-parameters were explored during the experiments.

- **Neural Network**

An ANN is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. ANNs can learn and model non-linear and complex relationships, which is important because in real-life many of the relationships between inputs and outputs are non-linear as well as complex.
After learning from the initial inputs and their relationships, a Neural Network can infer complex relationships on unseen data as well, thus making the model generalize and predict on unseen data.
Neural Networks require vast amounts of data and computing resources. Their black-box nature makes it hard to explain the results, but the performance of this model should be superior to other approaches.

Keras library will be used for implementing the NN, with a sequential model and only dense layers, the optimizer was set as Stochastic gradient descent (SGD) since it performed better than Adam on our data set. The metric was set as accuracy and the loss function as categorical crossentropy since the target array was converted to categorical arrays and the final layer output two probabilities with

[4] Sklearn documentation for SVM,
https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
[5] XGBoost winning solutions,
https://github.com/dmlc/xgboost/tree/master/demo#machine-learning-challenge-winning-solutions
[6] XGBoost documentation,
https://xgboost.readthedocs.io/en/stable/#

a sum of 1, in order to implement LIME for interpretation afterwards.

Keras runs on top of open-source machine libraries like TensorFlow, Theano or Cognitive Toolkit (CNTK). Theano is a python library used for fast numerical computation tasks. TensorFlow is the most famous symbolic math library used for creating neural networks and deep learning models. TensorFlow is very flexible and the primary benefit is distributed computing. CNTK is a deep learning framework developed by Microsoft. It uses libraries such as Python, C#, C++ or standalone machine learning toolkits. Theano and TensorFlow are very powerful libraries but difficult to understand for creating neural networks. Keras is based on a minimal structure that provides a clean and easy way to create deep learning models based on TensorFlow or Theano. Keras is designed to quickly define deep learning models. [7]

4.4. Evaluation methods

After the preprocessing, and classification models implementation, the following metrics and curves were implemented to map out the models' performances and compare with the previous works on this data set:

- Accuracy: getting an accuracy of 70% would mean that 70% of the data False and True combined, were well classified, the worst value for a binary classification would be 50% as it's a performance of a random and non-skill model.
- F1-score[8]: the higher the F1-score, the better performance on the positive class as the precision shows how much the model should be trusted when it classified the instance as True, and recall shows how much it classifies instances as True amongst the True examples.
- Confusion Matrix and Matthew Correlation coefficient (MCC)[9]: the MCC is known to be more robust than other metrics in regards to the fluctuation of results at each execution as it takes into account the four parts of the confusion matrix. It ranges from -1 to 1, and the goal is to have a value greater than 0.7 that indicates a strong correlation between the predictions and the ground truth [18].
- Receiver Operating Characteristics (ROC) curve: plots the evolution of True Positive Rate (TPR) versus False Positive Rate (FPR) for a decreasing value of threshold. It gives a good perspective of the performances on the positive class.
- MCC-F1 curve: plots the evolution of MCC and F1 scores for a decreasing value of threshold. This plot put together the two metrics F1-score and MCC in order to get a better overview of the global results [19].
- Probability Distribution: this plot shows how decisive the model is and how the probability threshold could affect the performance.

4.5. Lime for understanding the classification

Local Interpretable Model-agnostic Explanations (LIME) is a technique that allows an approximate interpretability to any black box model, by explaining the prediction for a certain instance with an interpretable model. For this project, our model would be of a better help to doctors if intelligible explanations to the predictions are provided.

The idea behind the algorithm as explained in [20] is to perturb the data in multiple points (i.e features) and try to find an interpretable model such as decision trees or linear models that minimizes a fidelity loss while taking into consideration its complexity. The loss compares the predictions between the model in hand

[7] Keras documentation, `https://keras.io/about/`

[8] the geometric average of the precision and recall

[9] the pearson correlation coefficient on the confusion matrix

and that of the interpretable model on the perturbed instance. The authors further use the similarity of the examples produced from the perturbation to the original sample as weights when calculating the fidelity loss.
The chosen model would then be fitted on the new data set, and the data point can be explained by the newly trained model.
This algorithm will not only help understand the decision, but also figure out why certain patients were misclassified.

5. Experiments

Various experiments were applied to find out which subset of the data set should be used and to fine-tune the models' hyperparameters. The final experiment performed aimed to find the best threshold for the classification task.

5.1. SVM tuning

The first model applied was the classification model support vector machine (SVM). The advantage of the SVM is that the model is easy to implement and does not require many optimizations. Experiments were performed with three different datasets: the first one contains all features (dataset wfs), the second one contains the seven best rated features from the feature selection (dataset 7) and the third one contains the four best features (dataset 4). Based on the data, a dividing line is determined that splits the datasets according to the decision to be made. This dividing line can take different forms and which in turn is determined by the kernel functions. The following kernel functions are available:

- Linear
- Radial basis function of the Gaussian kernel
- Sigmoid kernel
- Polynomial kernel

The dataset without feature selection (wfs) was applied to all kernel functions after the train-test-val-split without further adjustments. The result is recorded in figure 7 and table 6. A 2D representation of the SVM is shown in figure 2, where the first two features (BMI, Physical Health) are compared for each kernel.

Kernel	Linear	RBF	Poly	Sigmoid
Accuracy	0.7509	0.7504	0.7517	0.658
Sensitivity	0.73	0.72	0.73	0.657
Precision	0.73	0.72	0.76	0.66
Recall	0.75	0.75	0.75	0.66
F1 Score	0.75	0.75	0.75	0.66
MCC	0.5039	0.505	0.507	0.317

Figure 6: Tables of results with four different kernels on the whole data set

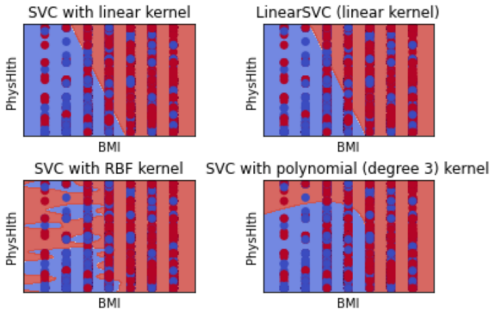


Figure 7: SVM boundaries of different kernels on two features : BMI and Physical Health

The sigmoid kernel function has the worst performance in all categories, so it can be excluded directly. The SVC with polynomial kernel performs best with an accuracy of 75.2%, the sensitivity however is quite similar between the three kernels. The next step was to apply the other two datasets to the polynomial kernel to check the influence of feature selection.

5.2. XGBoost tuning

XGBoost is the algorithm that was proven to be the top performer when it comes to tabular data sets on classification problems. The main challenge of this approach is tuning the hyper-parameters, in order to avoid overfitting. After performing the optimization, an accuracy of 83.18% was obtained on the training data set and 75.4% on the test data set.

- **Maximum depth of a tree:** It is a positive integer value, and is responsible for how deep each tree will grow during any boosting round. After trying out values ranging from 4 to 10, the optimal one was found to be 5.
- **Learning rate:** It affects how quickly the model fits the residual error using additional base learners.The values which have been used were [0.01, 0.1, 0.15, 0.2, 0.3]. The optimal value was found to be 0.1.
- **Gamma:** It is the minimum loss reduction to create new tree-split. To make the algorithm more conservative, the high value of gamma is preferred. The values which have been used were [1, 1.5, 2, 3, 4]. The optimal value was found to be 1.
- **reg_alpha and reg_lambda:** These parameters are L1 and L2 regularisation terms, respectively. The greater these numbers, the more conservative the model becomes. For this parameter values from 1 to 5 have been used with increments of 0.5. The best value for accuracy was found to be 3 for reg_alpha and 2 for reg_lambda.
- **Subsample:** It ranges from 0 to 1 and is the fraction of total training set that can be used for any given boosting round. The low value of this parameter may lead to unfitting problems and high values may lead to overfitting. A good compromise for this parameter was using 85% of the dataset.
- **colsample_bytree:** This parameter also ranges from 0 to 1. It is the fraction of features that can be selected during any given boosting rounds. The selected value was 0.5, meaning each tree uses only 50% of the features.

5.3. Neural Network tuning

The dense neural network underwent multiple experiments, the structure was created with only one hidden layer with 50 neurons, another layer of 30 neurons was added as it improved the performance, the addition of new layers did not further improve the accuracy. This model was therefore set for further experiments.
Various other factors could influence the model's performance, we performed a grid search cross validation with 5 folds on the training set, by putting the sequential model in a keras wrapper classifier, the experiments included the following hyperparameters:

- **Number of epochs and Batch size:** This tuning will look for a number of epochs that would maximize the performance whilst looking out for any overfitting. the ranges chosen for the number of epochs and batch sizes were respectively [10, 25, 50] and [256, 512, 1024, 2048]. We were able to choose high values for batch size our large training dataset.
- **Learning rate and momentum:** The optimizer used is SGD, the learning rate is usually higher than that of Adam optimizer so the values we used were [0.01, 0.1, 0.2, 0.3]. The momentum parameter was added to speed up the convergence of the gradient to the right direction, we explored the following values [0.0, 0.2, 0.4, 0.6, 0.8, 0.9].
- **kernel initializer:** The kernel initializer plays a big part on how the model will converge, various initializers were tried in this experiments: ['uniform', 'lecun_uniform', 'normal', 'zero', 'glorot_normal', 'glorot_uniform', 'he_normal', 'he_uniform'].

- **activation function:** The most commonly used activation function is Rectified Linear Unit (ReLU). However, as it disregards the negative values it can easily lead to a vanishing gradient. In this experiments the following activation function were applied for the hidden layers : ['softmax', 'softplus', 'softsign', 'relu', 'tanh', 'sigmoid', 'hard_sigmoid', 'linear'].

- **Dropout rate:** The dropout rate will decide the percentage of neurons to randomly leave out after the second hidden layer. This value ranged from 0.0 to 0.9 with a step of 0.1.

These experiments were performed step by step. At each step, the model's structure was updated with regards to the results of the previous experiment.

5.4. Threshold optimization

The threshold decides the range of probability for an instance to be considered True. The intuitive and most commonly used value is 0.5. However, if the model has more difficulty deciding on a class than another, it would be more practical to consider a different value of threshold.

The confusion matrix and the probability distribution plot allow to have an idea of which class is more difficult to predict than the other. It does not however provide an alternative value to 0.5. By leveraging the ROC curve and the MCC-F1 curve, as they plot the performance for a decreasing threshold, we can find a more suitable value for this problem.

The best threshold corresponds to a maximum value in a metric from the curves. For the ROC curve, we look to maximize the g-mean metric, an average of TPR and FPR. For the MCC-F1 curve, we're maximising the sum. We also tried to find the best threshold corresponding to the maximum accuracy as our dataset is balanced. This experiment was performed on the validation set, and the results were analysed on the test set.

5.5. Hardware

The computation power of Kaggle and google Colaboratory were leveraged during this project. Kaggle provides a Tesla P100 16gb VRAM as GPU, with 13gb RAM + 2-core of Intel Xeon as CPU and No-GPU option gives a 4-cores + 16gb RAM.
Google colab offers a Nvidia K80 / T4 16GB VRAM as GPU with 12gb RAM + 2-core of Intel Xeon as CPU. The only CPU execution mode provides a 2-cores + 12gb RAM [21].
The Neural network was run with the GPU using google Colaboratory, and the XGBoost on Kaggle. Since SVM doesn't support GPU, it was run with a CPU-only VM in Google Colaboratory. However, one training on the whole dataset can take at least 15 minutes.

6. Results and Discussion

When applying the SVM, the dataset without any omission gave the best results with an accuracy of 75.2% and higher values in all other metrics (figure 2) and the same observation was made for the neural network and the XGBoost models. Therefore, feature selection is not necessary in this case since the rest of the features, whilst not as important as the best 4 features, they still give information to the model.

We were able to achieve an accuracy of 75.1% with a linear SVC, one of the best performances compared

	Dataset without FS	Dataset with 7 features	Dataset with 4 features
Accuracy	0.7516	0.72	0.717
Sensitivity	0.73	0.69	0.68
Precision	0.76	0.73	0.72
Recall	0.75	0.72	0.72
F1 Score	0.75	0.72	0.72
MCC	0.507	0.445	0.44

Table 2: Tables of results with polynomial kernel on three different subsets

to the previous works on the data set. These results suggest that this problem is not highly non-linear, and can be processed by simple models (e.g few layers in the Neural Network). This also means the ability to leverage LIME for interpretations, as it only fails on highly non-linear problems [20]. This model was not kept for further analysis given the feasibility constraints in regards to the training duration and performance on our large data set.

The implementation of XGBoost improved the results to reach 75%. Based on the grid search, it used a value of gamma of 1.5, a learning rate of 0.2, a maximum depth of 5 and a regression lambda of 4.

As for the neural network, the structure from the tuning experiments contained two hidden layers of 100 and 30 neurons respectively. the model was trained on 90% of the training set, and validated on 10% of it, then tested on the rest of the data set. The training was performed with batches of 1024 samples and for 25 epochs. The kernel intializer function was set at "normal", and the activation function at "tanh".

The learning rate and momentum for SGD optimizer were set at 0.3 and 0.6 respectively and the dropout rate at 0.4.

The pipeline containing a one hot encoder, and the classification model was retrained on the entirety of the training set, and the performance curves on the validation set for the neural network and XGBoost are shown in figure 8 and the distributions in figures 10 and 9.

From the probability distribution plots, both models are more likely to predict a True class. However the neural Network would outperform the XGBoost on the False class as the former is able to output more probabilities around 0. The figures 11 and 12 show the confusion matrix of each model with a threshold of 0.5.

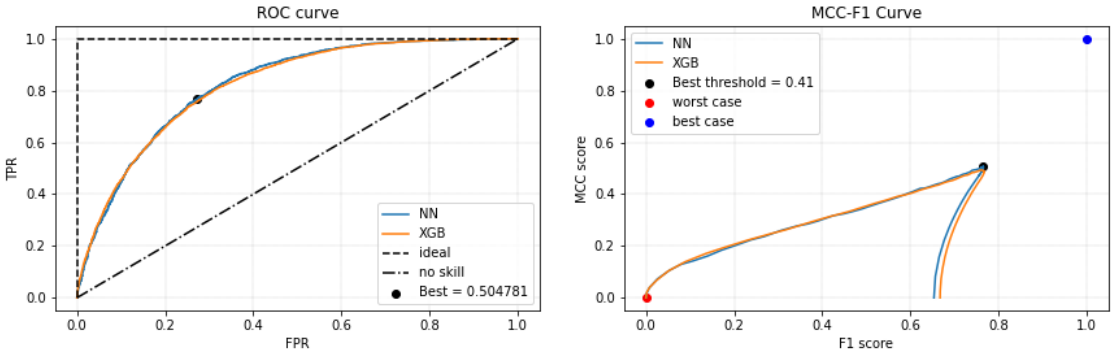


Figure 8: Performance curves with fine-tuned Neural Network and XGBoost models

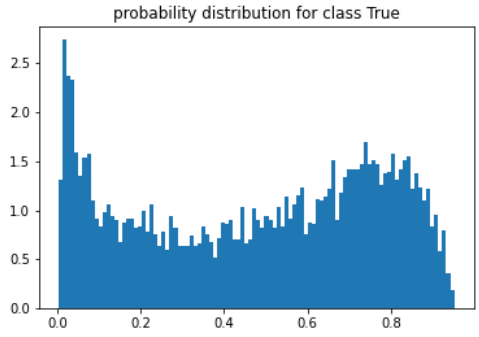


Figure 9: Probability distribution for the neural network

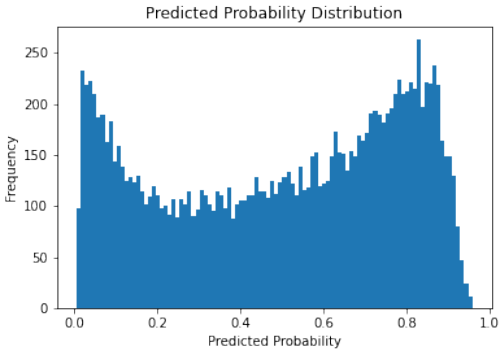


Figure 10: Probability distribution for XGBoost

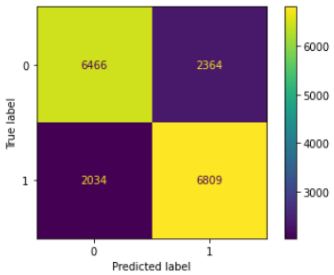


Figure 11: Confusion matrix on the test set from the neural network

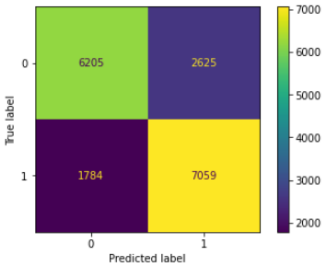


Figure 12: Confusion matrix on the test set from the XGboost

The classification results on the test set are presented in the tables 3 and, 4 with an average value of threshold between the one maximizing the g-mean , f1+mcc and accuracy values.

Threshold	Accuracy	Precision	Recall	F1-score	MCC	Confusion Matrix
0.5	0.751	0.742	0.770	0.756	0.503	$\begin{bmatrix} 6466 & 2364 \\ 2034 & 6809 \end{bmatrix}$
0.466	0.753	0.732	0.8	0.764	0.509	$\begin{bmatrix} 6237 & 2593 \\ 1769 & 7074 \end{bmatrix}$

Table 3: Results with Neural Network after threshold adjustment

Threshold	Accuracy	Precision	Recall	F1-score	MCC	Confusion Matrix
0.5	0.753	0.732	0.798	0.764	0.508	$\begin{bmatrix} 6252 & 2578 \\ 1786 & 7057 \end{bmatrix}$
0.48	0.754	0.725	0.817	0.768	0.511	$\begin{bmatrix} 6094 & 2736 \\ 1620 & 7227 \end{bmatrix}$

Table 4: Results with XGBoost after threshold adjustment

The table 5 presents a comparison of our results on the final three models with the previous works:

Model	Accuracy
NN [12]	74.6%
NN with 6 features [13]	74.8%
KNN voting ensemble [14]	74.1%
XGBoost [15]	**75.3**%
Our SVM	75.1%
Our XGBoost	**75.4**%
Our NN	**75.3**%

Table 5: Overview of the models accuracy, a comparison of our results with previous works

The new threshold from this experiment was lower than 0.5, which meant that more instances would be considered positive. This change was reflected in the value of recall and F1-score, the precision decreased as the number of False Positives increased more than True Positives. The correlation of the prediction and the truth however increased slightly by this adjustment.
The probability distribution shows that the model is still indecisive around the false class (healthy patients), changing the threshold can't have therefore a big impact on the results. It is worth noting however, that even if we predicted that model has more difficulty classifying the False instances, the best threshold still favored the performance on the True class. Perhaps other metrics should be taken into account, for instance, the F1-score in the MCC-F1 curve should an average of the score on both classes.

To understand more about this confusion, we plot the results from LIME on a False Positive as shown in the figure 13 with the 7 features contributing the most to the classification.

The high blood pressure and cholesterol level are important features for the classification task as per the aforementioned feature selection experiment. However, they also play a big role when mis-classifying patients. Having a high blood pressure and cholesterol level will most likely result in a classification as diabetic, the absence of heart diseases and alcohol consumption on the other hand contribute in classifying a patient as healthy but they are not as important as the former two features, based on the feature selection work.

When dealing with feature importance, it was surprising that "smoking" never appeared since it's known as a major factor. Then we dived into how the dataset was built and noticed that someone is considered a

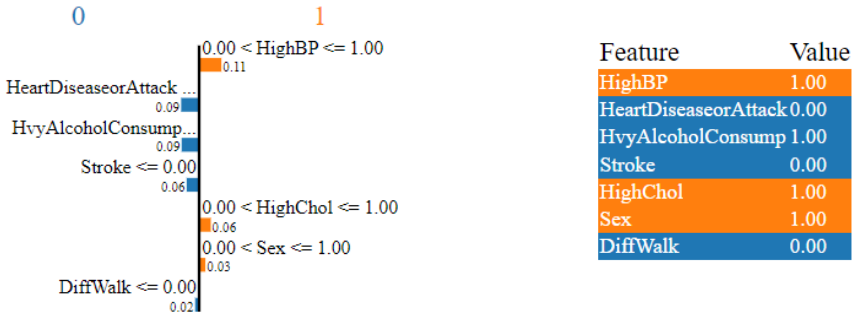


Figure 13: LIME output to a False Positive example with the Neural Network
Right: the values of the features in the dataset
Left: How important is each feature for the classification

smoker if he has already smoked a minimum of 100 cigarettes in his entire life. This threshold is very low and could explain why smoking wasn't considered as being important : it's easy to be assessed as a smoker even if you are actually not.

7. Conclusions

The project went through the Diabetes health indicators data set, from BFRSS survey in 2015, from understanding the questions asked during the survey, the relationships between the features, and those who play a big role for the classification task. It was preprocessed and the classification models were fine-tuned to get the best out the data available. Further experiments were performed to adapt the classification parameters to the problem and to get an overview of the models' decision making process. High blood pressure, high cholesterol, BMI, health, age and difficulty to walk have proven to have a high impact on the classification. They are usually also the factors behind the misclassification of patients, since the symptoms are similar. Their importance was highlighted when the performances from training with this subset of features were comparable to those with all the features. Perhaps the reason why none of the works have achieved an accuracy greater than 75.3% is that these questions were not enough to predict whether a patient has diabetes, more information could be used such as the features from the pima indian dataset. However, this information is not usually available when conducting surveys, as not all the participants visited the doctors for a diabetes check-up. The binary class value stems from a question "were you ever told that you have diabetes?", it doesn't necessarily mean that patients who were classified as healthy were not in reality diabetic.

Many actions could be taken at this point. For instance, not all the features from the survey were explored in this project, we could go back to the original data from BFRSS and implement our proposed models to see how the performance would vary, and to confirm the results on the research paper about important features for diabetes type 2 classification [2], new questions could also be proposed for future surveys. Another approach would be to implement a voting model with XGBoost and our Neural Network architecture as this approach was able to improve the performance compared to the individual results [10].

8. Acknowledgements

The authors would like to thank Abdolrahman Peimankar, professor at University of Southern Denmark, for his inputs during this project, on both the implementations and analysis, and Alex Teboul for his work on the original BFRSS survey dataset.

9. Contributions

The authors Alexandru Damian, Erwan Dehillerin, Kenza Makhlouf, Apolline Pignato and Jean-Luc Wolken all partook and contributed equally to this project. The tasks were assigned so as each step is tackled thoroughly, from data exploration, preprocessing, feature selection, and classification models.

References

[1] M. P. Petersen, Economic costs of diabetes in the u.s. in 2017 41 (2018) 917–928. doi:10.2337/dci18-0007.

[2] Z. Xie, O. Nikolayeva, J. Luo, D. Li, Building risk prediction models for type 2 diabetes using machine learning techniques, Preventing Chronic Disease 16 (09 2019). doi:10.5888/pcd16.190109.

[3] L. Fregoso-Aparicio, J. Noguez, L. Montesinos, J. García-García, Machine learning and deep learning predictive models for type 2 diabetes: a systematic review 22 (2021) 13–148. doi:10.1186/s13098-021-00767-9.

[4] S. Gkouzias, Diabetes prediction and risk factors evaluation (jan 2022).
URL https://www.kaggle.com/code/encode0/diabetes-prediction-and-risk-factors-evaluation

[5] S. Gill, P. Pathwar, Prediction of diabetes using various feature selection and machine learning paradigms, EasyChair Preprint 13 (2021).
URL https://easychair.org/publications/preprint/6TSn

[6] E. Longato, G. Acciaroli, A. Facchinetti, A. Maran, G. Sparacino, Simple linear support vector machine classifier can distinguish impaired glucose tolerance versus type 2 diabetes using a reduced set of cgm-based glycemic variability indices, Journal of Diabetes Science and Technology 14 (2) (2020) 297–302, pMID: 30931604. doi:10.1177/1932296819838856.

[7] C. Devadass, K. Velswamy, R. Velswamy, Classification of diabetes dataset using knn classifier and attribute selection through bees algorithm, Test Engineering and Management 83 (2020) 8195–8199.

[8] A. Iyer, J. S, R. Sumbaly, Diagnosis of diabetes using classification mining techniques, International Journal of Data Mining &amp Knowledge Management Process 5 (1) (2015) 01–14. doi:10.5121/ijdkp.2015.5101.
URL https://doi.org/10.5121%2Fijdkp.2015.5101

[9] H. Zhou, R. Myrzashova, R. Zheng, Diabetes prediction model based on an enhanced deep neural network, Eurasip Journal on Wireless Communications and Networking 2020 (12 2020). doi:10.1186/S13638-020-01765-7.

[10] N. H. Taz, A. Islam, I. Mahmud, A comparative analysis of ensemble based machine learning techniques for diabetes identification, in: 2021 2nd International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST), 2021, pp. 1–6. doi:10.1109/ICREST51555.2021.9331036.

[11] V. Sigillito, Uci repository of machine learning databases: Pima indians diabetes.
URL https://data.world/uci/pima-indians-diabetes

[12] L. Jaemin, Diabetes prediction with dnn (2021).
URL https://www.kaggle.com/code/ohguri/diabetes-prediction-with-dnn-pytorch

[13] W. L, Diabetes (keras dnn) (oct 2021).
URL https://www.kaggle.com/code/dogfish87/diabetes-keras-dnn

[14] Sayantan, Knn diabetes ensemble (2021).
URL https://www.kaggle.com/code/sayantan99/knn-diabetes-ensemble-191020450

[15] A. Agleev, Xgboost with parameter tuning (jan 2022).
URL https://www.kaggle.com/code/agleev/xgboost-with-parameter-tuning

[16] N. Bhandari, Extratreesclassifier (oct 2018).
URL https://medium.com/@namanbhandari/extratreesclassifier-8e7fc0502c7

[17] A. Zhu, Select features for machine learning model with mutual information (jun 2021).
URL https://towardsdatascience.com/select-features-for-machine-learning-model-with-mutual-information-534fe387d5c8

[18] P. Schober, C. Boer, L. Schwarte, Correlation coefficients: Appropriate use and interpretation, Anesthesia & Analgesia 126 (2018) 1. doi:10.1213/ANE.0000000000002864.

[19] C. Cao, D. Chicco, M. M. Hoffman, The mcc-f1 curve: a performance evaluation technique for binary classification (2020). arXiv:2006.11278.

[20] M. T. Ribeiro, S. Singh, C. Guestrin, "why should i trust you?": Explaining the predictions of any classifier (2016). doi:10.48550/ARXIV.1602.04938.
URL https://arxiv.org/abs/1602.04938

[21] A. Kazemnejad, How to do deep learning research with absolutely no gpus - part 2 (aug 2019).
URL https://kazemnejad.com/blog/how_to_do_deep_learning_research_with_absolutely_no_gpus_part_2/

Appendix .1. Feature selection models

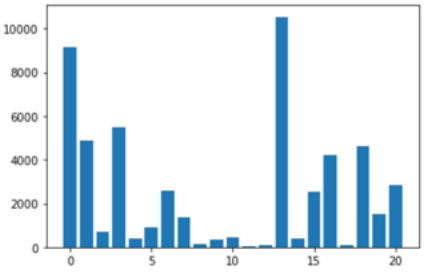
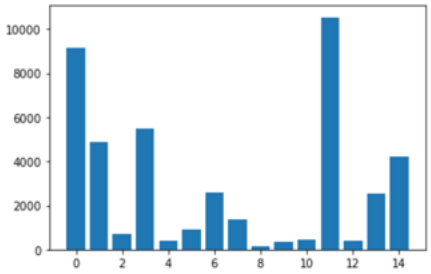
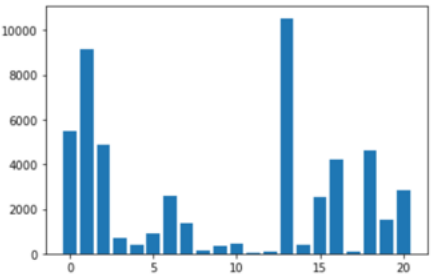
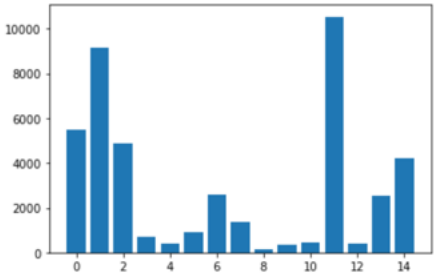
	All the features	Without socioeconomic features
Not scaled	 Specs Score 13 GenHlth 10538.865874 0 HighBP 9138.556329 3 BMI 5490.976831 1 HighChol 4866.193190 18 Age 4623.810764 16 DiffWalk 4229.145051 20 Income 2842.280735 6 HeartDiseaseorAttack 2583.641497 15 PhysHlth 2552.742212 19 Education 1537.536244	 Specs Score 11 GenHlth 10538.865874 0 HighBP 9138.556329 3 BMI 5490.976831 1 HighChol 4866.193190 14 DiffWalk 4229.145051 6 HeartDiseaseorAttack 2583.641497 13 PhysHlth 2552.742212 7 PhysActivity 1354.624265 5 Stroke 904.071576 2 CholCheck 723.997307
Scaled	 Specs Score 13 GenHlth 10538.865874 1 HighChol 9138.556329 0 HighBP 5490.976830 2 CholCheck 4866.193190 18 Age 4623.810764 16 DiffWalk 4229.145051 20 Income 2842.280735 6 HeartDiseaseorAttack 2583.641497 15 PhysHlth 2552.742212 19 Education 1537.536244	 Specs Score 11 GenHlth 10538.865874 1 HighChol 9138.556329 0 HighBP 5490.976830 2 CholCheck 4866.193190 14 DiffWalk 4229.145051 6 HeartDiseaseorAttack 2583.641497 13 PhysHlth 2552.742212 7 PhysActivity 1354.624265 5 Stroke 904.071576 3 BMI 723.997307

Figure .14: Results with the F-test

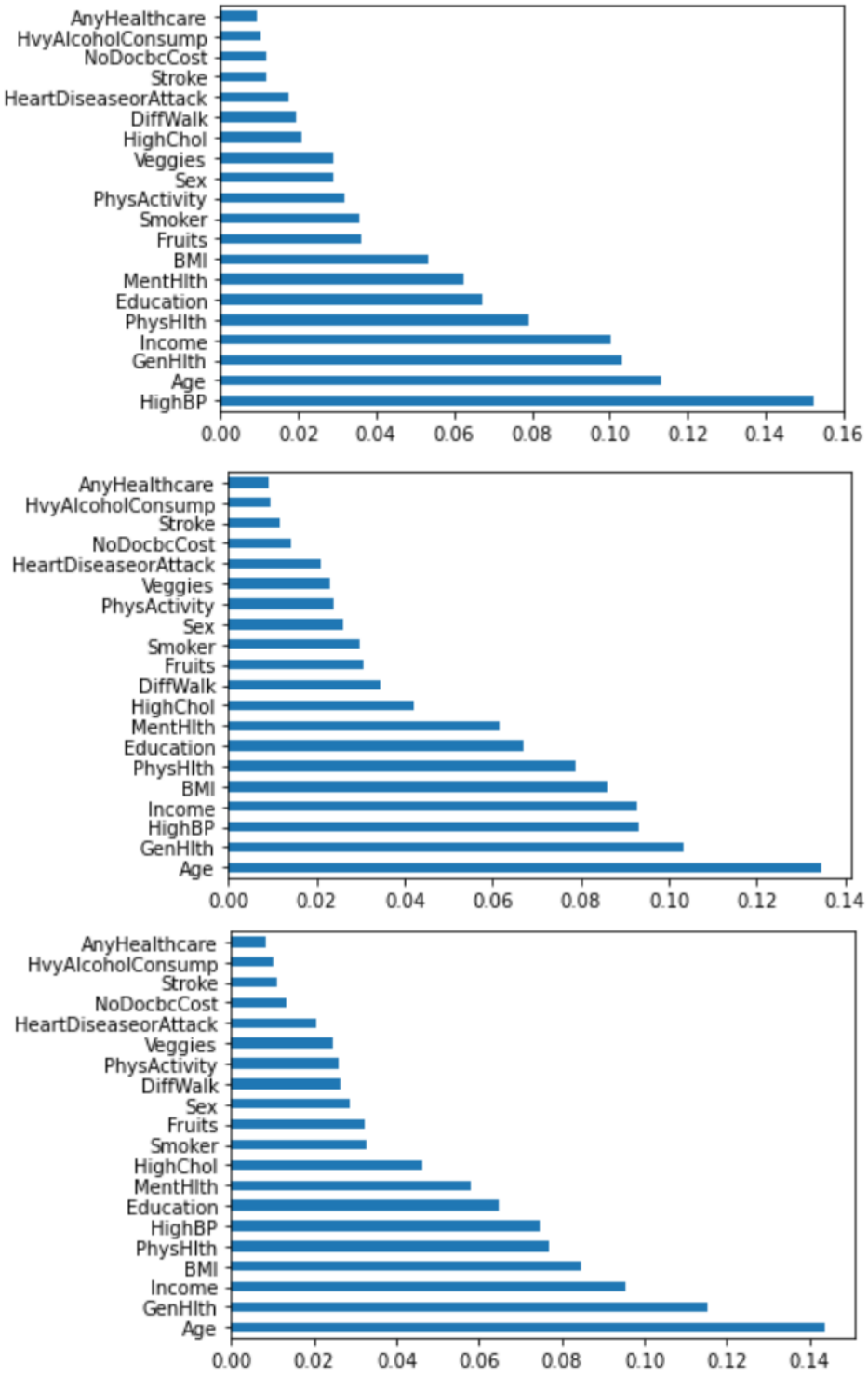


Figure .15: From the top to the bottom : Decision tree, Extra tree, Random Forest

Appendix .2. XGBoost feature importance

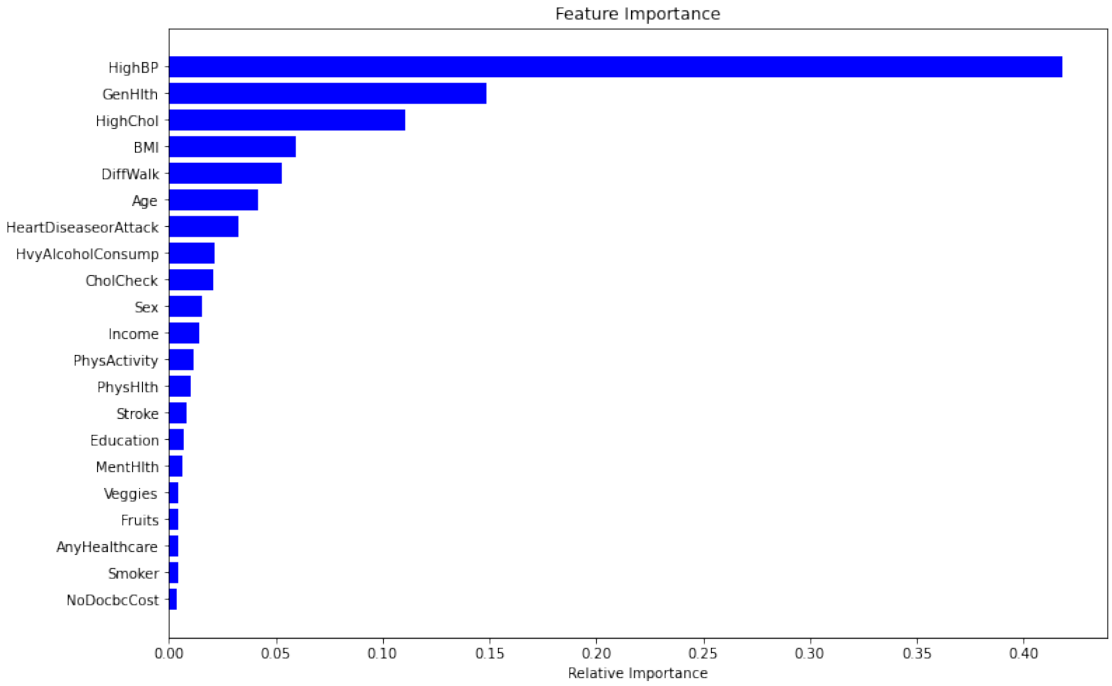


Figure .16: Relative feature importance for XGBoost model