Reproducibility in Diabetes Research Articles Using Machine Learning Classifiers

Faezeh Aliverdi

Supervised by: Krzysztof Podgórski

STAN40: One Year Master Thesis in Statistics
15 ECTS

DATA SCIENCE COURSE PACKAGE

Lund University
School of Economics and Management
Department of Statistics
Spring 2024

Abstract

Diabetes is a chronic disease that affects lots of people all around the world. It greatly impacts the health of those with it and places many demands on healthcare services. Early detection of diabetes can significantly improve treatment outcomes, lower the chances of future health issues, and help patients to have a better healthier life.

In this study, I evaluated the reproducibility of the performance of various machine learning (ML) classifiers from the four articles in predicting diabetes using two datasets: the updated Pima Indians Diabetes Database and the Diabetes Health Indicator from the Behavioral Risk Factor Surveillance System. Reproducing results in medical ML research is crucial for validating accuracy, ensuring generalizability, and building trust. The medical community can integrate ML into disease prediction and treatment protocols more safely and effectively by testing and reproducing results.

Using thorough data preprocessing methods, I prepared the datasets for analysis. Then, I utilized a range of ML algorithms, including support vector machine, decision trees, random forest, naive Bayes, logistic regression, and k-nearest neighbors to predict the diabetes cases on these datasets. I compared the performance of ML algorithms in terms of accuracy, precision, recall, F1-score, specificity, and area under the curve. The experimental results demonstrated that the random forest method is the most effective across different algorithms and datasets. In addition, the results show that I could achieve approximately similar results to previous articles on these datasets, supported by the outcomes of different ML algorithms. The slight differences between my findings and those in previous articles may be due to updates in the Pima Indians Diabetes Database and the specific methods I used for data preparation.

Keywords: Diabetes diagnosis, reproduction, ML algorithms, performance metrics.

Contents

List of Figures

List of Tables

List of Acronyms

ML:	Machine Learning
SVM:	Support Vector Machine
BMI:	Body Mass Index
IQR:	Interquartile Range
SMOTE:	Synthetic Minority Over-sampling Technique
RBF:	Radial Basis Function
RF:	Random Forest
LR:	Logistic regression
NB:	Naive Bayes
AUC:	Area Under Curve
ROC:	Receiver Operating Curve
DT:	Decision Tree
KNN:	K-Nearest Neighbours
TP:	True Positives
TN:	True Negatives
FP:	False Positives
FN:	False Negatives

1. Introduction

Diabetes is a major health problem around the world. It occurs when high levels of glucose remain in the blood for a prolonged period because the body cannot use it properly. This condition arises when the pancreas does not produce sufficient insulin or when the body cannot effectively utilize the insulin it has.

Early detection of diabetes is crucial for preventing the progression of disease and minimizing the risk of severe complications such as heart disease and kidney damage. By identifying diabetes at an early stage, individuals can manage their condition more effectively through lifestyle changes. This approach reduces healthcare costs by preventing the need for stronger treatments and dealing with serious health problems that come with later stages of diabetes [1].

Machine learning (ML) techniques are employed to identify patterns that aid in the early diagnosis of diabetes. Data scientists still face a big challenge: ensuring research results can be reproduced. It is essential to consistently reproduce results across different studies in the domain of using ML approaches to predict diabetes. It helps prove that the models are reliable and suitable for practical use in healthcare environments.

In this study, I concentrate on evaluating the reproducibility of four selected articles by reconstructing their ML models and methodologies to compare the original results with those obtained through replication. This study significantly contributes to medical informatics by assessing the reproducibility of ML models for diabetes prediction. Through thorough testing and validation of results, the medical community can integrate ML approaches into disease prediction and treatment protocols more safely and effectively. In addition, it provides practical insights for healthcare professionals while also highlighting the importance of research ethics and transparency. The selected articles have utilized a variety of ML algorithms, such as support vector machine (SVM), naive Bayes (NB), decision trees (DT), logistic regression (LR), random forest (RF), and k-nearest neighbor (KNN). The primary datasets discussed across these studies include the Pima Indians Diabetes Database and the Diabetes Health Indicators2015 Dataset, which feature comprehensive health data crucial for diabetes prediction.

Early and accurate diabetes prediction is crucial, as early detection can greatly improve both the management and treatment results [1]. There is a wide range of research on using ML to predict diabetes, showing a strong need for new ways to manage this disease. In this study, I reviewed several articles that applied various ML techniques for predicting diabetes. In particular, I chose four relevant articles that used notable ML algorithms and whose databases were publicly accessible. The four discussed articles are:

1. Kumari, V.A. and Chitra, R., 2013. Classification Of Diabetes Using Support Vector Machine. International Journal of Engineering Research and Applications, 3(2), pp.1797-1801. [2]
2. Hasan, M.K. et al., 2020. Diabetes prediction using ensembling of different machine learning classifiers. IEEE Access, 8, pp.76516-76531. [3]
3. Sisodia, D. and Sisodia, 2018. Prediction of Diabetes Using Classification Algorithms. Procedia Computer Science, 132, pp.1578-1585. [4]
4. Chang, V. et al., 2022. An Assessment of Machine Learning Models and Algorithms for Early Prediction and Diagnosis of Diabetes Using Health Indicators. Healthcare Analytics, 2, p.100118. [5]

In the first article [2], Kumari and Chitra utilized a SVM algorithm and 10-fold cross-validation for predicting diabetes using the Pima Indians Diabetes Database [6]. Their research demonstrated that SVM is effective in diagnosing diabetes. In the second article [3], Hasan et al. (2020) explored the performance of different ML and ensemble ML classifiers in predicting diabetes using the Pima Indians Diabetes Database, showing enhancements in both accuracy and reliability compared to single-model techniques. However, in this study, I will focus only on the ML classifiers. In the third article [4], Sisodia and Sisodia employed a variety of ML techniques to assess their effectiveness in diagnosing diabetes using the Pima Indians Diabetes Database. In the fourth article [5], Chang et al. (2022) introduced a comprehensive evaluation of various ML models and algorithms for the early prediction and diagnosis of diabetes using the Diabetes Health Indicators BRFSS2015 dataset [7].

The primary objective behind selecting these specific articles was to evaluate the reproducibility of their respective ML algorithms. By focusing on reproducibility, this study aims to contribute significantly to validating ML as a reliable tool in the predictive diagnosis of diabetes.

The rest of this study is organized as follows. Section 2 will introduce the datasets used, provide a summary of the key variables, and describe the design of the predictive models. Sections 3 and 4 contain the analysis of the results' reproducibility and the models' performance evaluation. Section 5 includes the study's conclusions.

2. Data Description

In this section, I discuss the datasets I use in my study. I describe how the datasets are prepared and define the important variables. For preparing the datasets, I followed the methods mentioned in the articles, which included normalizing or standardizing the data, removing duplicate rows, and fixing missing values. However, for feature engineering, I use a different method.

2.1 Source of Data

As previously mentioned, the datasets utilized in this research are sourced from the *Kaggle.com* website. The first three articles used the Pima Indians Diabetes Database and the fourth article Diabetes Health Indicator BRFSS2015 dataset provided by the National Institute of Diabetes and Digestive and Kidney Diseases.

Pima Indians Diabetes Database is provided by the National Institute of Diabetes and Digestive and Kidney Diseases. This dataset includes various medical diagnostic measurements critical for predicting diabetes. It includes health data crucial for predicting diabetes in Pima Indian women. This dataset has 768 records, each representing a different patient, and is available from the Kaggle.com website. It contains several important health measures like glucose level, blood pressure, Body Mass Index (BMI), and insulin level. With 8 variables related to diabetes symptoms and one outcome variable showing whether diabetes is present, this dataset helps analyze what factors contribute to diabetes, supporting detailed predictive studies.

In the fourth article analyzed, the Diabetes Health Indicator BRFSS2015 dataset from the centers for disease control and prevention (CDC) is utilized. The dataset “BRFSS2015.csv” is a clean set of 253,680 survey answers collected by the CDC in 2015. It has a main variable called “Diabetes_binary”, which marks people as '0' if they do not have diabetes and '1' if they have prediabetes or diabetes. 21 other variables in the dataset include various health and personal details. It is important to note that the Diabetes_binary is not balanced, which may require specific analytical approaches to address potential biases in model training and evaluation.

2. 2 Data Preprocessing

The data preprocessing steps are carefully designed to make sure the datasets are ready for further analysis and model training. These steps are essential for getting reliable and accurate results from the models.

Data Cleaning

For the Pima Indians Diabetes Database and the Diabetes Health Indicator BRFSS2015 dataset, the following data cleaning procedures are applied:

Handling missing values: In the Pima Indians Diabetes Database, several important features like glucose levels, blood pressure, skin thickness, insulin, and BMI contain zero values, which do not make sense medically. Specifically, the 'Glucose' feature has 5 instances of zero values, 'Blood Pressure' has 35, 'Skin Thickness' accounts for 227, 'Insulin' shows 347, and 'BMI' has 11 zero entries. These zeros are treated as missing data. To fix this, for the second article, I replaced the zeros with statistical measures (mean) from the respective feature values, keeping the data reliable for analysis. For the first article, I also considered removing rows containing these zero values from the dataset entirely. However, there are no zero values in the features of the Diabetes Health Indicator BRFSS2015 dataset.

Removal of duplicates: I removed any duplicate entries from the datasets to make sure they do not negatively affect the models' learning. This was especially important for the Diabetes Health Indicator dataset, where I found and removed 24,206 duplicate entries, unlike the fourth article, which did not address these duplicates.

Outlier rejection: Outlier rejection is a crucial step in data preprocessing, especially in tasks involving ML. Outliers can significantly distort the results of data analysis and statistical modeling. It is important to identify and possibly remove the outliers to improve the accuracy and performance of data analysis. Interquartile Range (IQR) is a commonly used statistical method to detect outliers. It measures the statistical spread of the middle 50% of the data. IQR is calculated by subtracting the first quartile (25th percentile) from the third quartile (75th percentile). Any data points outside these bounds are typically considered outliers. For the second article, I used outlier rejection.

Data Transformation

Data transformation methods were used to normalize and standardize the data, ensuring that all data points were on a similar scale and preventing any single feature from having too much influence on the predictions. This is important because it helps ML models learn and make predictions more effectively. Normalization, often done using min-max scaling, brings all numerical values into a range between 0 and 1. Standardization was applied using the standard scaler to bring all features to a mean of zero and a standard deviation of one. I applied the standard scaler for the first three articles, while for the fourth article, I used min-max scaling.

Feature Engineering

Correlation analysis is used to find out how strongly each feature is related to diabetes prediction. This step is important for picking out the most helpful features, allowing us to concentrate on variables that have a big impact on diabetes risk. The features with strong correlations are selected because their values are closely linked to whether or not someone has diabetes, making them effective for predicting diabetes in ML models.

In the second article, for ML algorithms like the KNN, RF, and NB, six features were manually selected: 'Pregnancies', 'Glucose', 'Blood Pressure', 'Skin Thickness', 'Insulin', and 'Diabetes Pedigree Function'. Additionally, for the DT algorithm, four features were manually selected: 'Glucose', 'Insulin', 'Skin Thickness', and 'BMI'.

In the fourth article, for ML algorithms such as the KNN, RF, LR, and NB, eight features were manually selected: BMI, Age, High Blood Pressure, High Cholesterol, General Health, Physical Health, Difficulty Walking, and Income.

For the first article and the third article, I did not use the feature engineering method.

Data Augmentation

Synthetic Minority Over-sampling Technique (SMOTE) is a method used to balance class distribution in datasets by artificially generating new examples in the minority class. This method creates new synthetic examples to balance the number of instances between classes. It works by taking samples from the minority class and creating similar, but slightly altered, new samples. This helps ensure that the training dataset has an equal number of examples from each class, making it better for training the model. I used the SMOTE technique to balance the classes in the Diabetes Health Indicator dataset, as suggested in the fourth article. Figure 1 illustrates the distribution of each class of the Diabetes Health Indicator dataset after using the SMOTE technique.

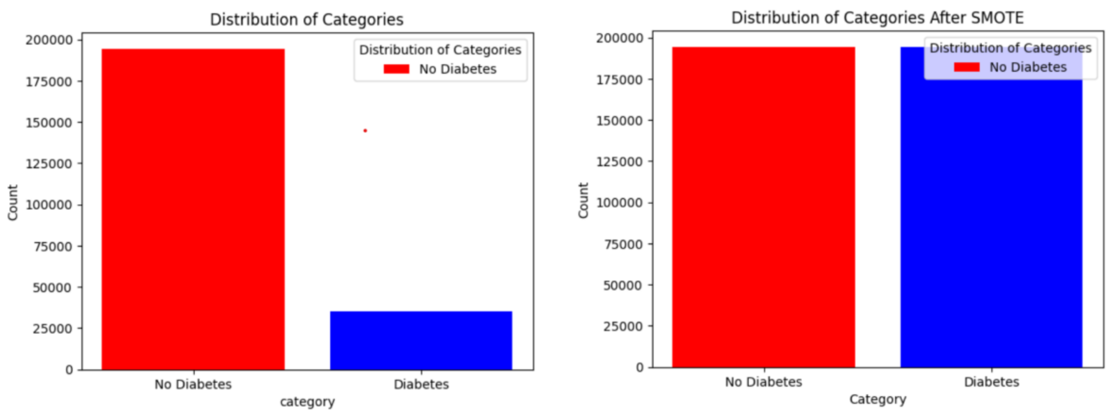


Figure 1. The distribution of each class before (left) and after (right) data augmentation in the Diabetes Health Indicator dataset.

2.2.1 Summary of Preprocessing for each dataset

In this section, I provide the summary of the preprocessing process applied to each dataset, presented in Table 1.

The first article used the Pima Indian Diabetes Database. The preprocessing steps involved removing all missing values and performing data transformation using the standard scaler. In the second article, which is also based on the Pima Indian Diabetes Database, missing values were replaced with the mean values from the respective features. Additionally, outlier rejection was performed on the dataset. Standard scaling was used to make the dataset uniform. Feature selection was executed, with either 4 or 6 features being selected based on their correlation. The third article also utilized the Pima Indian Diabetes Database. Like the second article, it replaced missing values with mean and did not address duplicate rows or outlier rejection. The standard scaler was used for data transformation without any feature selection or data augmentation. The fourth article employed the Diabetes Health Indicator dataset. This study removed all duplicated rows. Data transformation was conducted using min-max scaling, and eight features were selected based on correlation. Additionally, the article used SMOTE for data augmentation to address the class imbalance of the response.

Table 1. Overview of preprocessing steps applied to each dataset.

Article	**Dataset**	**Missing values**	**Duplicated rows**	**Outlier rejection**	**Data transformation**	**Features selection**	**Data argumentation**
First article	Pima Indian Diabetes Databas e	Remov e all missing values	_	_	Standard scaler	_	_
Second article	Pima Indian Diabetes Databas e	Replac e the zero values with the mean	_	Outlier rejection by interqua rtile range	Standard scaler	4 and 6 features were selected with correlatio n	_
Third article	Pima Indian Diabetes Databas e	Replac e the zero values with the mean	_	_	Standard scaler	_	_
Fourth article	Diabetes health indicator	_	Remove all duplicated rows	_	Min-max scaling	8 features were selected with correlatio n	SMOTE

3. Methodology

In this section, I discuss the methods used in four different articles that focus on predicting diabetes with various ML algorithms. Each study uses a clear and organized approach to developing models, which involves training the models with different ML algorithms and evaluating their performance.

3.1 Support Vector Machine (SVM)

SVM is one of the popular nonlinear supervised ML models. The objective of the SVM algorithm is to find a hyperplane in an N-dimensional space (, where N is the number of the features) that distinctly classifies the data points. In this study, SVM aims to find the best hyperplane that separates the dataset into two classes (for binary classification like both Diabetes datasets). In one-dimensional space, the hyperplane is a point, in two-dimensional space, the hyperplane is a line and in three-dimensional space, the hyperplane is a surface that divides space into two parts. Each class lies on either side. SVM can help us choose the best hyperplane, which maximizes the margin. Maximizing this distance can minimize the risk of misclassifying examples from the test dataset [8].

Another concept in SVM is called support vectors. Support vectors are the data points that lie closest to the hyperplane. They are the data points most difficult to classify and determine which hyperplane we should choose. For SVM, only the difficult points, which are support vectors, impact the decision boundary. Moving a support vector moves the decision boundary [1].

As illustrated in Figure 2, the hyperplane can be described mathematically by the equation $w \cdot x+b=0$, where w is a weight vector that points perpendicular to the hyperplane, and b is a bias term that adjusts the position of the hyperplane along the direction of w. The weight vector helps determine the orientation of the hyperplane, while the bias shifts it closer to or farther from the origin. The space or *margin* between the dividing line (hyperplane) and the nearest data points from each class is calculated as $2/\|w\|$. In simple terms, this margin is the safe zone that helps prevent misclassification. To make this margin as wide as possible, SVM tries to make the value of $\|w\|$ (the length or size of the weight vector w) as small as possible. However, while doing this, SVM also ensures that all the training data points are correctly classified. This means each data point x_i must meet the requirement $y_i(w \cdot x_i + b) \geq 1$, where y_i is the class label. This condition helps to make sure the data points are not only classified correctly but also stay as far away from the hyperplane as possible, within their correct sides [1].

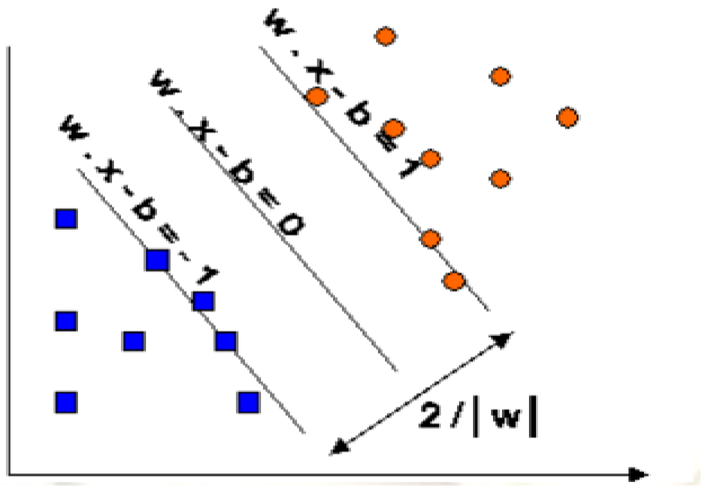


Figure 2. Classification of data by SVM, reproduced from [1].

Radial Basis Function (RBF) Kernel in SVM

The RBF is a kernel of SVM that measures the distance between a data point and a fixed support vector and maps this distance onto a higher-dimensional space. It is defined as $K(x_i,x_j) = \exp(-\gamma\|x_i - x_j\|^2)$, where γ is the kernel parameter, and x_i and x_j are the support vector and a test data point. A larger γ value results in a smoother decision boundary. This kernel allows each support vector to influence only nearby points (determined by γ), making the classifier adaptable to more complex data patterns [9].

The first and third articles both used SVM. In the first article, the SVM model uses the RBF kernel, which is a popular choice for classification problems involving non-linear data. It transforms the feature space into a higher dimension where the separation hyperplane can easily classify the data points into different categories. In the first article, the authors split the data into the 46% train set and 56% test set. Then, the 10-fold cross-validation was used during the training phase to validate the performance of SVM. This method splits the whole dataset into ten equal sections. The model is trained on nine of these sections and tested on the remaining one. This process is repeated ten times, each time a different section is used for testing. This ensures that all parts of the data are used for both training and testing, helping to make a thorough and fair evaluation of the model.

The third article does not mention any specific kernel, which means it likely uses the basic linear kernel. This method works well when the data can be clearly separated by a straight line, or when a simple straight boundary is enough to classify the data accurately. Instead of splitting the data into separate training and testing sets, the third article uses 10-fold cross-validation directly to assess how well the SVM model works.

3.2 Random Forest (RF)

RF [10] is a supervised ML algorithm that can be used for both classification and regression tasks. The algorithm is very adaptable and can be applied to various datasets because of its straightforward and flexible nature. RF constructs an ensemble of DTs, each trained on different bootstrapped samples. Each DT is created using a random selection of features at every division point. This approach helps protect the trees against errors and inaccurate forecasts. Typically, about two-thirds of the data is utilized in these randomly selected bootstrapped datasets. Once the trees are built, the algorithm tests each observation through all the trees and averages their outcomes. This process enhances the accuracy of predictions and helps prevent overfitting. Through these steps, the algorithm ensures more reliable results.

Generally, the primary benefit of utilizing the RF algorithm in this study is its adaptability and high efficiency across various data types. It has a minimal risk of overfitting as there are sufficient trees in the model, and it can quickly generate predictions as it only uses a subset of features.

In the second article, the RF algorithm was used as part of a wider study of ML techniques. Instead of splitting the data into separate training and testing sets, the study used the 5-fold cross-validation to assess the RF model. This involves dividing the whole dataset into five equal sections, training the model in four sections, and testing it in the remaining one. This process is repeated five times, each time a different section is used for testing. This technique ensures that every data point is used in both training and testing, improving the overall reliability of the model's performance measures. The RF model was configured with 100 decision trees, using the "gini" criterion for splits, and a fixed random state for reproducibility "random_state =100".

In the fourth article, the RF model was employed, and the dataset was explicitly split into training and testing sets with 80% of the data used for training and 20% for testing. This setup is a common practice for evaluating the model's performance on unseen data, ensuring that the model is not only fit to the training data but also capable of generalizing well to new unseen instances. The RF employed 10 trees and focused on the *entropy* criterion to measure the quality of splits and a fixed random state for reproducibility "random_state =42". These setup were chosen to enhance the reliability and consistency of the model's predictions.

3.3 Logistic Regression (LR)

LR [11] is a type of regression analysis used in predictive modeling, particularly suited for binary classification tasks. This means it predicts outcomes that have two possible states, such as yes or no, true or false, 0 or 1, or high or low. If the probability is below 0.5, it is rounded down to 0, and if it is above 0.5, it is rounded up to 1. The algorithm uses an S-shaped sigmoid function, presented in equation (1) below, which takes any real-valued number (y) and maps it onto a range between 0 and 1:

$$sigmoid(y) = \frac{1}{1 + e^{-y}} \tag{1}$$

In the fourth article, the LR was employed, and the dataset was split into training and testing sets with 80% of the data used for training and 20% for testing. The model used was a standard LR from the scikit-learn library, configured with L2 regularization to prevent overfitting and 'liblinear' as the solver, which is well-suited for small to medium-sized datasets and binary classification problems. The main hyperparameters discussed include the penalty type and solver.

3.4 Naive Bayes (NB)

NB is one of the simplest yet efficient ML algorithms. It is based on the principles of probability, utilizing past data to predict outcomes for new, unseen data. NB calculates the likelihood of different outcomes (classes) for a new observation by considering each attribute's frequencies observed in the training data's classes [12]. For example, NB could be used to predict whether an individual has diabetes by analyzing various health indicators. The model treats each feature independently, assuming no correlation between them. This assumption simplifies the calculations but ignores any potential interactions between features, which might affect the accuracy in cases where feature dependencies exist. The algorithm operates based on Bayes' Theorem, which in a simplified form is:

$$P(A|B) = P(B|A) \times P(A) \,/\, P(B) \tag{2}$$

where:

- P(A|B) is the probability of hypothesis *A* given the data *B*.
- P(B|A) is the probability of the data *B* given that the hypothesis *A* is true.
- P(A) is the probability of hypothesis *A* being true, irrespective of the data.
- P(B) is the probability of the data, irrespective of the hypothesis.

The third article implemented 10-fold cross-validation to assess the NB model while the NB classifier was implemented without specific mention of parameter adjustments or tuning parameters. The primary focus was on the basic implementation and evaluation across different metrics such as accuracy, precision, recall, and F1 score.

Also, in the second article, 5-fold cross-validation and the "var_smoothing" parameter, set to 0.01, are used in the NB classifier. This parameter helps in smoothing the probabilities of the features to avoid zero probabilities, thus making the model more robust against data variations and improving classification performance.

Moreover, unlike the other articles that used cross-validation, article four utilized an 80-20 train-test split to evaluate the NB model, although specific tuning parameters are not detailed.

3.5 Decision Trees (DT)

DT is a popular and powerful ML technique used for both classification and regression tasks. They are simple to understand and use, and can work with data that includes both numbers and categories, making them useful for many kinds of tasks. A DT looks like a flowchart where each point where a decision branches off (called a node) represents a decision based on one piece of information or attribute. Each branch shows the possible outcome of that decision, and each endpoint (called a leaf) represents the prediction. The whole path from the start to the end of the tree shows the steps taken to reach that prediction [12].

In the third article, the DT model was applied without mentioning the parameter tuning setup, focusing only on model evaluation through a 10-fold cross-validation process.

In the second article, 5-fold cross-validation was employed to assess the performance of the DT model, and several parameters were specifically tuned to optimize its performance, such as:

- "min_sample_split = 0.1", which means that a node must have at least 10% of the training data to consider a split, preventing overly complex branches and helping in generalization.
- criterion = "gini", which refers to the Gini impurity measure to decide where to split the data, trying to create groups that are as similar as possible within each group.
- Min_sample_leaf = 1 and splitter = "best", which ensures that each leaf node has at least one sample, and the splitter chooses the best split at each node to optimize purity.

The fourth article employed an 80-20 split between training and testing data to assess the performance of the DT model. Also, the DT in the fourth article was tuned with specific parameters such as max_depth and criterion. Max_depth limits the depth of the tree to 10 levels, which controls overfitting by preventing the tree from growing too deep. Also, the criterion chooses splits based on entropy, which maximizes information gain per split, effectively reducing uncertainty with each decision, and a fixed random state for reproducibility "random_state =42".

3.6 K-Nearest Neighbours (KNN)

KNN is a type of classification algorithm used to predict which group a piece of data belongs to, based on certain rules already set in the system. This method is often described as a "lazy" approach because it does not create a model from the training data until it needs to make a prediction [12]. It is also called a "non-parametric" method, meaning it does not assume anything specific about the underlying pattern of the data.

In the second and fourth articles, the KNN algorithm was used to classify data, focusing on adjusting important settings to improve how well it works. The second article utilized a 5-fold cross-validation and article 4 employed an 80-20 split.

In the second article, the KNN model was specifically configured with 27 neighbors, using a brute-force search approach to compute the distances between all pairs of points. This approach ensured precise identification of the nearest neighbors, utilizing the Manhattan distance (p = 1) for its calculations and leaf_size=30.

The fourth article applied a more detailed parameter configuration to the KNN model, setting *n_neighbours* to 3, using the *kd_tree* algorithm, which is efficient for large datasets, and choosing the Euclidian metric for distance calculation.

3.7 Performance Metrics

In ML, particularly in classification tasks such as predicting diabetes, various performance metrics are used to evaluate and compare the effectiveness of models. True Positives (TP) and True Negatives (TN) are cases where the model correctly predicts the positive and negative classes, respectively. Also, False Positives (FP) and False Negatives (FN) are cases where the model incorrectly predicts the positive and negative classes, respectively.

Accuracy is a fundamental metric that measures the ratio of correctly predicted observations, both TP and TN, to the total number of observations. The formula for accuracy is expressed as:

$$\frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

Also, training accuracy refers to how well the model predicts the outcome of the data it was trained on, while test accuracy indicates the model's performance on new, unseen data, providing a measure of its generalizability.

Precision focuses on the positive predictions made by the model. It calculates the ratio of TP to the total predicted positives, which include both TP and FP. The precision formula is:

$$\frac{TP}{TP + FP} \tag{4}$$

Recall, also known as sensitivity, checks how well the model can correctly find all the positive cases. It looks at the number of TP and compares it to the total number of actual positive cases, which include the ones it missed. The formula for the recall is:

$$\frac{TP}{TP + FN} \tag{5}$$

F-Measure provides a single score that balances the trade-offs between precision and recall, making it particularly useful when comparing two or more models. It is the weighted average of precision and recall, calculated as:

$$\frac{2TP}{2TP \; + \; FP \; + \; FN} \tag{6}$$

Specificity is a statistical measure used to evaluate the performance of a binary classification test. It assesses the ability of the test to correctly identify negative cases. In other words, specificity measures the proportion of actual negatives that are correctly identified as such by the test. This is particularly important in medical testing, where it is crucial to identify those individuals who do not have a disease. The formula for specificity is expressed as:

$$\frac{TN}{TN + \; FP} \tag{7}$$

The area under the curve (AUC) is an important metric in evaluating the performance of diagnostic tests. The AUC represents the probability that a randomly chosen positive instance is ranked higher than a randomly chosen negative one. This metric is useful because it provides a single number summarizing the model's performance across all threshold settings.

Summary of all tunning parameters of models

Table 2 provides a summary of ML approaches and validation methods used in each of the mentioned articles.

Table 2. ML approaches and validation methods were used in the four selected articles.

Article	**Model**	**Methodology/Tunning**	**Validation method**
1	SVM	Uses RBF kernel	10-fold cross-validation
3	SVM	Likely uses a basic linear kernel.	10-fold cross-validation
2	RF	Configured with 100 decision trees, "gini" criterion, random state=100.	5-fold cross-validation
4	RF	10 trees, "entropy" criterion, random state set for reproducibility.	80-20 split
4	LR	Standard Logistic Regression with L2 regularization, 'liblinear' solver.	80-20 split
3	NB	Basic implementation, possibly with default settings.	5-fold cross-validation

2	NB	"var_smoothing" set to 0.01.	10-fold cross-validation
4	NB	Basic implementation, possibly with default settings.	80-20 split
2	DT	Tuned parameters: "min_sample_split = 0.1", "criterion = ‘gini’", "min_sample_leaf = 1", "splitter = ‘best’".	5-fold cross-validation
3	DT	with default settings.	10-fold cross-validation
4	DT	Configured with "max_depth=10", "criterion='entropy'", random state=42.	80-20 split
2	KNN	Configured with 27 neighbors, "algorithm='brute'", "metric='Manhattan'", "leaf_size=30'".	10-fold cross-validation
4	KNN	Configured with 3 neighbors, "algorithm='kd_tree'", "metric='euclidean'".	80-20 split

4. Results and Discussion

In this section, I compare the results of this study and the articles that contributed to predicting diabetes using ML approaches. The models were compared based on several key measures, including accuracy, precision, recall, F1-score, and AUC, to help us understand how well each model performs. For the implementation, I utilized common data science tools and libraries in Python using Google Colab, including Pandas for data manipulation, Scikit-learn for ML model implementation, and Matplotlib for data visualization.

4.1 First article

The first article employed only one ML algorithm, which is SVM. In this study, I applied the SVM algorithm to predict diabetes using the Pima Indians Diabetes Database. After training the model, it was used to make predictions on the test set. I split the dataset into training and testing sets, using 56% of the data for testing to see how well the model works. I trained the SVM model, which uses an RBF kernel and predicts probabilities, on the remaining 44% of the data. To make sure the model was reliable and worked well on different types of data, I used the 10-fold cross-validation on the training data. This method helped confirm that the model was performing consistently by testing it on several smaller sections of the training data, which also helped avoid biases that could come from using just one split of the data. I evaluated the effectiveness of the SVM model using a range of metrics, as presented in Table 4. Table 3 presents a pseudocode for reproducing the approach carried out in the first article for the early prediction of diabetes.

Table 3. Pseudocode for reproducing the approach presented in the first article, which uses SVM for early prediction of diabetes.

```
1. Import the Pima Indians Diabetes Database
2. Remove zero values in important features, such as glucose levels, blood pressure, skin
   thickness, insulin, and body mass index (BMI).
3. StandardScaler
4. Model = SVC (kernel = 'rbf', probability=True, random_state=42)
5. Model. fit()
6. Model. predict()
7. Print (Training_accuracy ( ), Test_accuracy ( ), Sensitivity ( ), Specificity () )
```

Table 4 presents the results of this study and the first article. I achieved 0.74 for the mean accuracy, while the article reported an accuracy of 0.78. Achieving higher training accuracy and specificity in this study, along with lower test accuracy and sensitivity, suggests that the model might be overfitting the training data. These differences might also be due to changes in the dataset used in the first article. The first article used an older version of the dataset, which is no longer available in Kaggle. While I used a newer updated version in this study. According to the results, I could acquire better training accuracy and specificity than the first article.

Table 4. Comparison of the results of this study and the first article.

Study	**Model**	**Training accuracy**	**Test accuracy**	**Sensitivity**	**Specificity**	**AUC**
This thesis	SVM	0.89	0.74	0.49	0.87	0.80
First article	SVM	0.65	0.78	0.80	0.76	_

4.2 Second article

In the second article, four ML algorithms were used: DT, RF, KNN, and NB. I trained these algorithms separately on the Pima Indians Diabetes Database; however, this time, I chose four specific features for DT and six specific features for RF, KNN, and NB based on their correlation to improve the model's accuracy. I utilized 5-fold cross-validation and made several adjustments to the settings, as mentioned in Table 2. Then, I evaluated the model's performance on the testing data using a range of metrics, as presented in Table 6. Table 5 represents a pseudocode for reproducing the approach carried out in the second article for the early prediction of diabetes.

Table 5. Pseudocode for reproducing the approach presented in the second article for the early prediction of diabetes using ML classifiers.

```
1. Import the Pima Indians Diabetes Database
2. Replace zero values with the mean of the respective feature values, such as glucose levels,
   blood pressure, skin thickness, insulin, and body mass index (BMI).
3. Outlier rejection
4. Feature engineering
5. StandardScaler
6. MLC = [DecisionTreeClassifier(min_samples_split=0.1, criterion='gini',
                                   min_samples_leaf=1, splitter='best'),
   RandomForestClassifier(n_estimators=100, criterion='gini',random_state=100),
   KNeighborsClassifier (n_neighbors=27, algorithm='brute', p=1, leaf_size=30),
   NB (var_smoothing= 0.01)]

7. for (i=0;i<4;i++) do
       model = MLC[i]
       model.fit ( );
       model.predict ( );
       print(classification_report ( ), accuracy_score ( ), precision_score ( ),
             recall_score ( ), F1_score ( ), AUC_score( ))
   end
```

Table 6 presents the results of this study and the second article concerning the AUC. The results indicate that the classifiers from both this study and the second article had almost similar performances with only slight differences. The second article tends to show a trend towards slightly higher AUC values with lower variance, suggesting potentially more robust or consistent models. Comparing the performance of different classifiers in this study and the article reveals that the RF classifier performs the best. The results of the second article also show that RF with the AUC of 0.938±0.01outperforms other algorithms.

Table 6. Comparison of the results of this study and the second article.

Study	**Model**	**Mean accuracy**	**Precision**	**Recall**	**F1**	**AUC**	**Winner**
This thesis	DT	0.85	0.78	0.72	0.75	0.91±0.03	Both in this study and the second article, the RF model is the winner.
Second article	DT	_	_	_	_	0.91±0.01	
This thesis	RF	0.86±0.02	0.80	0.73	0.76	0.927±0.01	
Second article	RF	_	_	_	_	0.938±0.01	
This thesis	KNN	0.85	0.76	0.79	0.77	0.924±0.02	
Second article	KNN	_	_	_	_	0.922±0.02	
This thesis	NB	0.808 ±0.039	0.723	0.621	0.66	0.872±0.04	
Second article	NB	_	_	_	_	0.879±0.02	

4.3 Third article

The third article implemented three ML algorithms: DT, NB, and SVM. I used 10-fold cross-validation to assess the performance of these models without specific mention of parameter adjustments or tuning practices. I assessed the performance of each model using various metrics, as shown in Table 8. Table 7 presents a pseudocode for reproducing the approach carried out in the third article for the early prediction of diabetes.

Table 7. Pseudocode for reproducing the approach presented in the third article for the early prediction of diabetes using ML classifiers.

```
1. Import the Pima Indians Diabetes Database
2. Replace zero values with the mean of the respective feature values, such as glucose levels,
   blood pressure, skin thickness, insulin, and body mass index (BMI).
3. StandardScaler
4. MLC=[DecisionTreeClassifier(), SVC(), NB (var_smoothing= 0.01)]

5. for (i=0;i<3;i++) do
       model = MLC[i]
       model. fit ( );
       model. predict ( );
       print(classification_report ( ), accuracy_score ( ), precision_score ( ),
      recall_score ( ), F1_score ( ), AUC _score( ))
   end
```

Table 8 presents the results of this study and the third article. The comparison indicates that the DT and NB models implemented in the third article consistently show superior performance across all metrics (Mean accuracy, Precision, Recall, F1 Score, and AUC) compared to those in this study. The article's findings favored the NB, achieving an accuracy of 0.76, followed by the DT with 0.74. However, the SVM used in this study does much better than the SVM from the third article with higher scores in all metrics except recall. My results show the SVM performs the best among other classifiers with an accuracy of 0.76. The difference between my results and the article is probably due to the difference in data preprocessing, as mentioned in Section 2.

Table 8. Comparison of the results of this study and the third article.

Study	**Model**	**Mean accuracy**	**Precision**	**Recall**	**F1**	**AUC**	**Winner**
This thesis	DT	0.70	0.57	0.56	0.57	0.67	In my study, the SVM is the winner, while in the third article, the NB is the winner.
Third article	DT	0.73	0.73	0.73	0.73	0.75	
This thesis	NB	0.75	0.65	0.60	0.62	0.81	
Third article	NB	0.76	0.75	0.76	0.76	0.81	
This thesis	SVM	0.76	0.69	0.52	0.59	0.82	
Third article	SVM	0.65	0.42	0.65	0.51	0.50	

4.4 Fourth article

In the fourth article, five ML algorithms were used: NB, DT, LR, RF, and KNN. I trained these algorithms on the Diabetes Health Indicator dataset and employed an 80-20 split between training and testing data to assess the performance of these models. I also adjusted the specific parameters of these models, as mentioned in Table 2. Then, I evaluated the model's performance on the testing data using a range of metrics, as presented in Table 10. Table 9 presents a pseudocode for reproducing the approach carried out in the fourth article for the early prediction of diabetes.

Table 9. Pseudocode for reproducing the approach presented in the fourth article for the early prediction of diabetes using ML classifiers.

```
1. Import the Diabetes Health Indicator dataset
2. Remove duplicated rows
3. Feature engineering (select 8 features)
4. Class Imbalance: SMOTE
5. Min-max scaling
6. MLC=[DecisionTreeClassifier( max_depth=10, criterion='entropy', random_state=42),
            RandomForestClassifier(n_estimators=10,criterion='entropy',
                                        max_depth=None,random_state=42),
           KNeighborsClassifier (n_neighbors=3, algorithm='kd_tree',metric='euclidean'),
           NB (),
           LogisticRegression(random_state=0, solver='liblinear',  and binary classification
                               penalty='l2')]

7. for (i=0;i<5;i++) do
       model = MLC[i]
       model. fit ( );
       model. predict ( );
       print(classification_report ( ), accuracy_score ( ), precision_score ( ),
             recall_score ( ), F1_score ( ), AUC_score( ))
   end
```

Table 10 presents the results of this study and the fourth article. In comparing the results between this study and the fourth article, the RF classifier emerges as the top classifier in both sets of results. In my study, RF achieved an accuracy of 0.86. Similarly, in the fourth article, RF also outperformed the other classifiers with an accuracy of 0.82.

DT in this study shows slightly lower accuracy, precision, and F1 with a higher recall compared to the fourth article. KNN in this study outperforms the fourth article in mean accuracy, recall, and F1 metrics, while the fourth article demonstrates higher precision. For the LR model, this study achieves better results than the fourth article in recall and F1 metrics, while mean accuracy is equal in this study and the fourth article. However, the fourth article achieves a higher precision. For the NB model, this study slightly outperforms the fourth article in mean accuracy, while precision and F1 scores are equal in this study and the fourth article.

Table 10. Comparison of the results of this study and the fourth article.

Study	**Model**	**Mean accuracy**	**Precision**	**Recall**	**F1**	**AUC**	**Winner**
This thesis	DT	0.79	0.80	0.78	0.79	0.88	Both in this study and the fourth article, RF is the
Fourth article	DT	0.81	0.83	0.77	0.81	_	
This thesis	RF	0.86	0.88	0.85	0.86	0.93	
Fourth article	RF	0.82	0.83	0.80	0.82	_	

This thesis	KNN	0.81	0.78	0.84	0.81	0.86	winner.
Fourth article	KNN	0.80	0.81	0.79	0.80	_	
This thesis	LR	0.73	0.70	0.76	0.74	0.81	
Fourth article	LR	0.73	0.72	0.74	0.73	_	
This thesis	NB	0.71	0.72	0.60	0.70	0.78	
Fourth article	NB	0.70	0.72	0.67	0.70	_	

5. Conclusions

This thesis focused on reproducing results from four articles on predicting diabetes using ML algorithms. These articles used publicly available datasets and employed algorithms including SVM, DT, RF, NB, LR, and KNN. After analyzing these algorithms, I discovered several important insights. In this thesis, the second article, and fourth article, the RF algorithm consistently performed better than other ML classifiers because it can handle large and complex data without overfitting, which is crucial in medical diagnosis. My evaluation focusing on the reproducibility of these articles confirmed these findings. The RF algorithm not only outperformed other models in the studies reviewed but also proved to be the most effective in this thesis's experiments. For the fourth article, I achieved higher accuracy than the accuracy reported in the article. This consistent outcome confirms that RF is a reliable and effective tool for predicting diabetes, making it very useful for both research and practical healthcare applications. However, models like SVM in the first and third articles achieved more inconsistent results. This inconsistency may be due to changes in the data over time and differences in how the data was prepared, compared to the methods described in the articles. For the first article, the accuracy I achieved with the SVM model was 0.74, lower than the 0.78 reported in the article. Conversely, for the third article, using the same SVM model, I achieved a higher accuracy of 0.76 compared to the 0.65 reported in the article. For the other classifiers, I achieved approximately similar accuracy to those reported in the articles, especially for the second and fourth articles. The differences between my results and those in previous articles may be due to updates in the Pima Indians Diabetes Database, especially for the first article, and the specific methods I used for data preparation for the third article.

References

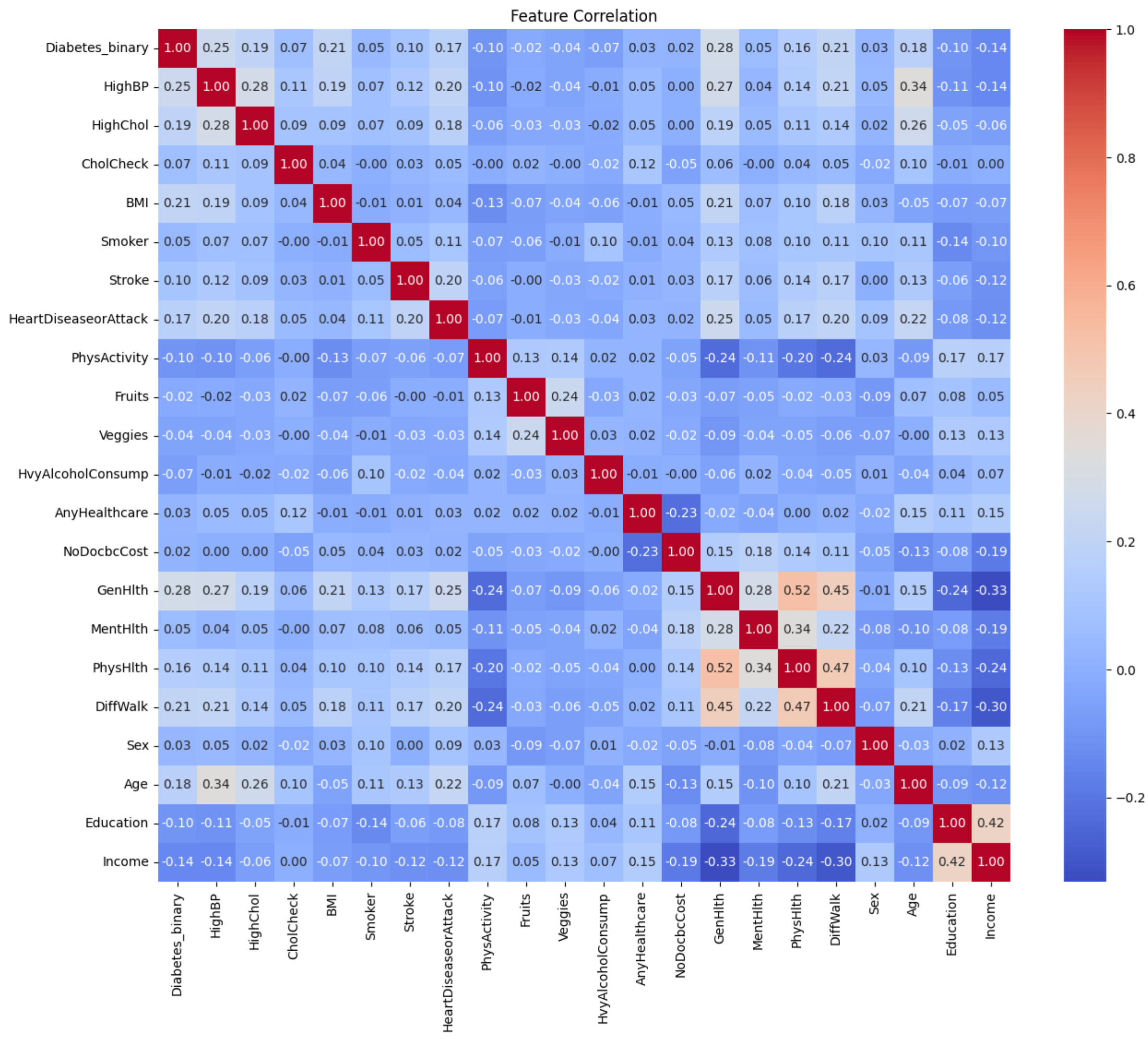
1. Jaiswal, V., Negi, A. and Pal, T., 2021. A review on current advances in machine learning based diabetes prediction. *Primary Care Diabetes*, *15*(3), pp.435-443.
2. Kumari, V.A. and Chitra, R., 2013. Classification of diabetes disease using support vector machine. *International Journal of Engineering Research and Applications*, *3*(2), pp.1797-1801.
3. Hasan, M.K., Alam, M.A., Das, D., Hossain, E. and Hasan, M., 2020. Diabetes prediction using ensembling of different machine learning classifiers. *IEEE Access*, *8*, pp.76516-76531.
4. Sisodia, D. and Sisodia, D.S., 2018. Prediction of diabetes using classification algorithms. *Procedia computer science*, *132*, pp.1578-1585.
5. Chang, V., Ganatra, M.A., Hall, K., Golightly, L. and Xu, Q.A., 2022. An assessment of machine learning models and algorithms for early prediction and diagnosis of diabetes using health indicators. *Healthcare Analytics*, *2*, p.100118.
6. UCI Machine Learning (latest update: 2016). *Pima Indians Diabetes Database* [Data set]. Kaggle. Available at: https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database/code (Accessed: April 1, 2024).
7. Teboul, A. (latest update: 2022). *Diabetes Health Indicators Dataset* [Data set]. Kaggle. Available at: https://www.kaggle.com/datasets/alexteboul/diabetes-health-indicators-dataset (Accessed: April 1, 2024).
8. Burges, C.J., 1998. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, *2*(2), pp.121-167.
9. Park, J. and Sandberg, I.W., 1991. Universal approximation using radial-basis-function networks. *Neural computation*, *3*(2), pp.246-257.
10. Hastie, T., Tibshirani, R., Friedman, J.H. and Friedman, J.H., 2009. *The elements of statistical learning: data mining, inference, and prediction* (Vol. 2, pp. 1-758). New York: springer.
11. Hilbe, J.M., 2009. *Logistic regression models*. Chapman and hall/CRC.
12. Russell, S.J. and Norvig, P., 2016. *Artificial intelligence: a modern approach*. Pearson.

Appendix

Table 11. Description of the Diabetes Health Indicator dataset [5].

Number	Features	Description
1	Diabetes binary	0 = no diabetes, 1 = diabetes
2	HighBP	0 = no high BP 1 = high BP
3	HighChol	0 = no high cholesterol 1 = high cholesterol
4	CholCheck	0 = no 1 = yes
5	BMI	Body Mass Index
6	Smoker	Have you smoked at least 100 cigarettes in your entire life? 0 = no 1 = yes
7	Stroke	You had a stroke. 0 = no 1 = yes
8	HeartDiseaseorAttack	Coronary Heart Disease (CHD) or myocardial infarction (ML) 0 = no 1 =yes
9	PhysActivity	physical activity in the past 30 days - not including job 0 = no 1 = yes
10	Fruits	Consume Fruits 1 or more times per day 0 = no 1 = yes
11	Veggies	Consume Vegetables 1 or more times per day 0 = no 1 = yes
12	HvyAlcoholConsump	Heavy drinkers (adult men having more than 14 drinks per week and adult women having more than 7 drinks per week) 0 = no 1 = yes
13	AnyHealthcare	Have any kind of health care coverage, including health insurance, prepaid plans such as HMO, etc. 0 = no 1 = yes
14	NoDocbcCost	Was there a time in the past 12 months when you needed to see a doctor but could not because of cost? 0 = no 1= yes
15	GenHlth	Would you say that in general, your health is? Scale 1–5 1= excellent 2 = very good 3 = good 4 = fair 5 = poor
16	MentHlth	Now thinking about your mental health, which includes stress, depression, and problems with emotions, for how many days during the past 30 days was your mental health not good? scale 1–30 days.
17	PhysHlth	Now thinking about your physical health, which includes physical illness and injury, for how many days during the past 30 days was your physical health not good? scale 1–30 days
18	DiffWalk	Do you have serious difficulty walking or climbing stairs? 0 = no 1 = yes
19	Sex	0 = female 1 = male
20	Age	13-level age category (_AGEG5YR see codebook) 1 = 18–29 9 = 60–64 13=80 or older
21	Education	Education level (EDUCA see codebook) scale 1–6 1 = Never attended school or only kindergarten 2 = Grades through 8 (Elementary) 3 = Grades 9 through 11 (some high school) 4 = Grade 12 or GED (High school graduate) 5 = College 1 year to 3

		years (Some college or technical school) 6 = College 4 years or more (College graduate)
22	Income	Income scale (INCOME2 see codebook) scale 1–8 1 = less than $10,000 5 = less than $35,000 8 = $75,000 or more



	Diabetes_binary	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	HeartDiseaseorAttack	PhysActivity	Fruits	Veggies	HvyAlcoholConsump	AnyHealthcare	NoDocbcCost	GenHlth	MentHlth	PhysHlth	DiffWalk	Sex	Age	Education	Income
Diabetes_binary	1.00	0.25	0.19	0.07	0.21	0.05	0.10	0.17	-0.10	-0.02	-0.04	-0.07	0.03	0.02	0.28	0.05	0.16	0.21	0.03	0.18	-0.10	-0.14
HighBP	0.25	1.00	0.28	0.11	0.19	0.07	0.12	0.20	-0.10	-0.02	-0.04	-0.01	0.05	0.00	0.27	0.04	0.14	0.21	0.05	0.34	-0.11	-0.14
HighChol	0.19	0.28	1.00	0.09	0.09	0.07	0.09	0.18	-0.06	-0.03	-0.03	-0.02	0.05	0.00	0.19	0.05	0.11	0.14	0.02	0.26	-0.05	-0.06
CholCheck	0.07	0.11	0.09	1.00	0.04	-0.00	0.03	0.05	-0.00	0.02	-0.00	-0.02	0.12	-0.05	0.06	-0.00	0.04	0.05	-0.02	0.10	-0.01	0.00
BMI	0.21	0.19	0.09	0.04	1.00	-0.01	0.01	0.04	-0.13	-0.07	-0.04	-0.06	-0.01	0.05	0.21	0.07	0.10	0.18	0.03	-0.05	-0.07	-0.07
Smoker	0.05	0.07	0.07	-0.00	-0.01	1.00	0.05	0.11	-0.07	-0.06	-0.01	0.10	-0.01	0.04	0.13	0.08	0.10	0.11	0.10	0.11	-0.14	-0.10
Stroke	0.10	0.12	0.09	0.03	0.01	0.05	1.00	0.20	-0.06	-0.00	-0.03	-0.02	0.01	0.03	0.17	0.06	0.14	0.17	0.00	0.13	-0.06	-0.12
HeartDiseaseorAttack	0.17	0.20	0.18	0.05	0.04	0.11	0.20	1.00	-0.07	-0.01	-0.03	-0.04	0.03	0.02	0.25	0.05	0.17	0.20	0.09	0.22	-0.08	-0.12
PhysActivity	-0.10	-0.10	-0.06	-0.00	-0.13	-0.07	-0.06	-0.07	1.00	0.13	0.14	0.02	0.02	-0.05	-0.24	-0.11	-0.20	-0.24	0.03	-0.09	0.17	0.17
Fruits	-0.02	-0.02	-0.03	0.02	-0.07	-0.06	-0.00	-0.01	0.13	1.00	0.24	-0.03	0.02	-0.03	-0.07	-0.05	-0.02	-0.03	-0.09	0.07	0.08	0.05
Veggies	-0.04	-0.04	-0.03	-0.00	-0.04	-0.01	-0.03	-0.03	0.14	0.24	1.00	0.03	0.02	-0.02	-0.09	-0.04	-0.05	-0.06	-0.07	-0.00	0.13	0.13
HvyAlcoholConsump	-0.07	-0.01	-0.02	-0.02	-0.06	0.10	-0.02	-0.04	0.02	-0.03	0.03	1.00	-0.01	-0.00	-0.06	0.02	-0.04	-0.05	0.01	-0.04	0.04	0.07
AnyHealthcare	0.03	0.05	0.05	0.12	-0.01	-0.01	0.01	0.03	0.02	0.02	0.02	-0.01	1.00	-0.23	-0.02	-0.04	0.00	0.02	-0.02	0.15	0.11	0.15
NoDocbcCost	0.02	0.00	0.00	-0.05	0.05	0.04	0.03	0.02	-0.05	-0.03	-0.02	-0.00	-0.23	1.00	0.15	0.18	0.14	0.11	-0.05	-0.13	-0.08	-0.19
GenHlth	0.28	0.27	0.19	0.06	0.21	0.13	0.17	0.25	-0.24	-0.07	-0.09	-0.06	-0.02	0.15	1.00	0.28	0.52	0.45	-0.01	0.15	-0.24	-0.33
MentHlth	0.05	0.04	0.05	-0.00	0.07	0.08	0.06	0.05	-0.11	-0.05	-0.04	0.02	-0.04	0.18	0.28	1.00	0.34	0.22	-0.08	-0.10	-0.08	-0.19
PhysHlth	0.16	0.14	0.11	0.04	0.10	0.10	0.14	0.17	-0.20	-0.02	-0.05	-0.04	0.00	0.14	0.52	0.34	1.00	0.47	-0.04	0.10	-0.13	-0.24
DiffWalk	0.21	0.21	0.14	0.05	0.18	0.11	0.17	0.20	-0.24	-0.03	-0.06	-0.05	0.02	0.11	0.45	0.22	0.47	1.00	-0.07	0.21	-0.17	-0.30
Sex	0.03	0.05	0.02	-0.02	0.03	0.10	0.00	0.09	0.03	-0.09	-0.07	0.01	-0.02	-0.05	-0.01	-0.08	-0.04	-0.07	1.00	-0.03	0.02	0.13
Age	0.18	0.34	0.26	0.10	-0.05	0.11	0.13	0.22	-0.09	0.07	-0.00	-0.04	0.15	-0.13	0.15	-0.10	0.10	0.21	-0.03	1.00	-0.09	-0.12
Education	-0.10	-0.11	-0.05	-0.01	-0.07	-0.14	-0.06	-0.08	0.17	0.08	0.13	0.04	0.11	-0.08	-0.24	-0.08	-0.13	-0.17	0.02	-0.09	1.00	0.42
Income	-0.14	-0.14	-0.06	0.00	-0.07	-0.10	-0.12	-0.12	0.17	0.05	0.13	0.07	0.15	-0.19	-0.33	-0.19	-0.24	-0.30	0.13	-0.12	0.42	1.00

Figure 3. Representation of the correlation matrix for the Diabetes Health Indicator dataset used in the fourth article. Eight features were selected: BMI, Age, High Blood Pressure, High Cholesterol, General Health, Physical Health, Difficulty Walking, and Income [5].

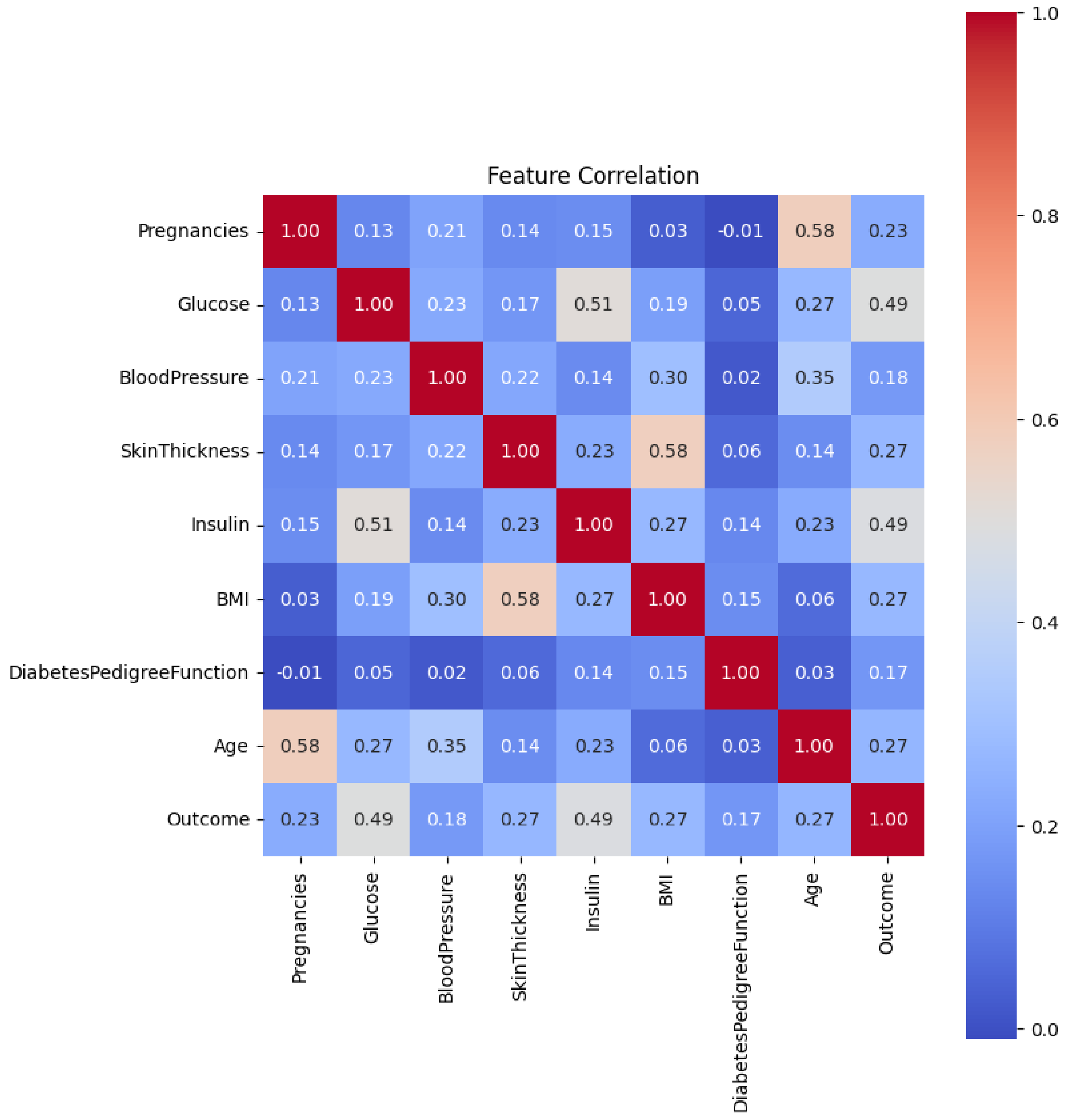


Figure 4. Representation of the correlation matrix for the Pima Indians Diabetes Database dataset used in the second article. In the second article, for ML algorithms like K-Nearest Neighbours, Random Forest, and Naive Bayes, six features were manually selected: 'Pregnancies', 'Glucose', 'Blood Pressure', 'Skin Thickness', 'Insulin', and 'Diabetes Pedigree Function'. Additionally, for the Decision Tree algorithm, four features were manually selected: 'Glucose', 'Insulin', 'Skin Thickness', and 'BMI' [3].