Laboratory assignment

Component 1-5

Authors: Alexandru Profir, Tudor-Octavian Mihăiţă
Group: ICA 246-2

January 14, 2026

1 Overview

This software project aims to develop two fully functional Machine Learning solutions addressing distinct **Supervised Learning** tasks within the healthcare domain, focusing on the diagnosis and risk assessment of diabetes. The subsequent sections present a common overview of of the input data and its specification, followed by the the individual definitions and formalizations of the two learning problems.

1.1 Data Specification

The input dataset consists of $100,000$ patient records characterized by features based on demographics, lifestyle habits, family history, and clinical measurements that are well-established indicators of diabetes risk. The data was generated using statistical distributions from real-world medical research, ensuring *privacy preservation* while reflecting realistic health patterns. The dataset is publicly available on Kaggle.

Table 1 presents an overview of all input features used in both learning tasks. Each feature is described by its name, data type, a brief explanation, and a corresponding domain (*value range*), therefore also specifying the **preconditions** of the utilized data. As confirmed by the creators of the dataset, all values fall within realistic medical ranges, matching realistic population health patterns, being a viable option when exploring lifestyle and clinical health patterns.

1.2 Data Analysis

We conduct a series of data analysis experiments on the *Diabetes Health Indicators* dataset [Tha24] in order to understand the statistical properties of the features, their relationships to each other, and their relevance for predicting the binary outcome variable `diagnosed_diabetes`. The analysis includes descriptive statistics, distributional examination, correlation assessment, statistical independence testing, and feature importance estimation based on filter methods.

1.2.1 Dataset Description and Descriptive Statistics

The dataset comprises $N = 100{,}000$ instances and $d = 31$ attributes. These attributes cover:

- demographic factors (e.g. age, gender, ethnicity, education level, income level, employment status),

Feature	Type	Description	Domain
patient_id	Integer	Unique patient identifier	$\mathbb{N} \cap [1, 100000]$
age	Integer	Age of patient in years	$\mathbb{N} \cap [18, 90]$
gender	Categorical	Patient gender	{Male, Female, Other}
ethnicity	Categorical	Ethnic background	{White, Hispanic, Black, Asian, Other}
education_level	Categorical	Highest completed education	{No formal, Highschool, Graduate, Postgraduate}
income_level	Categorical	Income category	{Low, Medium, High}
employment_status	Categorical	Employment type	{Employed, Unemployed, Retired, Student}
smoking_status	Categorical	Smoking behavior	{Never, Former, Current}
alcohol_consumption_per_week	Float	Drinks consumed per week	$\mathbb{R}_+ \cap [0, 30]$
physical_activity_minutes_per_week	Integer	Physical activity (weekly minutes)	$\mathbb{N} \cap [1, 100000]$
diet_score	Integer	Diet quality (higher = healthier)	$\mathbb{N} \cap [1, 10]$
sleep_hours_per_day	Float	Average daily sleep hours	$\mathbb{R}_+ \cap [3, 12]$
screen_time_hours_per_day	Float	Average daily screen time hours	$\mathbb{R}_+ \cap [0, 12]$
family_history_diabetes	Integer	Family history of diabetes	$\mathbb{N} \cap \{0, 1\}$
hypertension_history	Integer	Hypertension history	$\mathbb{N} \cap \{0, 1\}$
cardiovascular_history	Integer	Cardiovascular history	$\mathbb{N} \cap \{0, 1\}$
bmi	Float	Body Mass Index (kg/m^2)	$\mathbb{R}_+ \cap [15, 45]$
waist_to_hip_ratio	Float	Waist-to-hip ratio	$\mathbb{R}_+ \cap [0.7, 1.2]$
systolic_bp	Integer	Systolic blood pressure ($mmHg$)	$\mathbb{N} \cap [90, 180]$
diastolic_bp	Integer	Diastolic blood pressure ($mmHg$)	$\mathbb{N} \cap [60, 120]$
heart_rate	Integer	Resting heart rate (bpm)	$\mathbb{N} \cap [50, 120]$
cholesterol_total	Float	Total cholesterol (mg/dL)	$\mathbb{R}_+ \cap [120, 30]$
hdl_cholesterol	Float	HDL cholesterol (mg/dL)	$\mathbb{R}_+ \cap [20, 100]$
ldl_cholesterol	Float	LDL cholesterol (mg/dL)	$\mathbb{R}_+ \cap [50, 200]$
triglycerides	Float	Triglycerides (mg/dL)	$\mathbb{R}_+ \cap [50, 500]$
glucose_fasting	Float	Fasting glucose (mg/dL)	$\mathbb{R}_+ \cap [70, 250]$
glucose_postprandial	Float	Post-meal glucose (mg/dL)	$\mathbb{R}_+ \cap [90, 350]$
insulin_level	Float	Blood insulin level ($\mu U/mL$)	$\mathbb{R}_+ \cap [2, 50]$
hba1c	Float	HbA1c (%)	$\mathbb{R}_+ \cap [4, 14]$

Table 1: Specification and preconditions of the input features

- lifestyle indicators (e.g. smoking status, alcohol consumption per week, physical activity minutes per week, diet score, sleep hours per day, screen time hours per day),
- medical history (family history of diabetes, hypertension history, cardiovascular history),
- physiological and biochemical measurements (e.g. BMI, waist-to-hip ratio, blood pressure values, heart rate, lipid profile, fasting and postprandial glucose, insulin level, HbA1c),
- a composite *diabetes risk score*, the *diabetes stage*, and the binary target `diagnosed_diabetes`.

For each numerical variable, standard descriptive statistics (mean, variance, minimum, maximum, quartiles) were computed. Given a numerical feature $X = (x_1, \ldots, x_N)$, the empirical mean and variance are given by:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i, \qquad \mathrm{Var}(X) = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2$$

These statistics confirm that:

- Most clinical variables (e.g. glucose values, lipid parameters, blood pressure) span realistic clinical ranges with substantial dispersion;
- Several lifestyle variables (e.g. physical activity) present highly heterogeneous values, with a large mass at relatively low activity levels;
- Categorical variables (such as gender, ethnicity, education level, employment status, smoking status, and diabetes stage) are appropriately encoded in a finite number of categories and later transformed into numerical codes for subsequent analysis.

The descriptive analysis was carried out both on the raw dataset and on a preprocessed version in which missing values were handled, categorical features were numerically encoded, and data types were downcast to reduce memory usage, without altering the underlying information.

1.2.2 Distributional Analysis and Visualisation

To better understand the empirical distributions of the features and to identify potential outliers and deviations from normality, several visualizations were employed on the numerical attributes of the dataset. Due to the high dimensionality of the feature space, we chose to visualize only a subset of characteristics in order to keep the visualizations relevant.

Boxplots. Boxplots were used to summarize the distribution of each continuous variable through its median, interquartile range (**IQR**) and potential outliers. For a variable X, the boxplot highlights:

- the lower (Q_1 / 25th percentile), the median (Q_2 / 50th percentile), and the upper (Q_3 / 75th percentile) quartiles,
- observations falling outside the typical range, often defined as points beyond $Q_1 - 1.5 \cdot IQR$ or $Q_3 + 1.5 \cdot IQR$.

As shown in Figure 1, the resulting boxplots indicate:

- noticeable right-skewed distributions for variables such as triglycerides, LDL cholesterol and postprandial glucose, with several high-value outliers;

- a moderate spread for BMI and waist-to-hip ratio, with some extreme values but largely plausible ranges;
- a binary or near-binary pattern for some history variables (e.g., hypertension history, cardiovascular history), reflecting the presence or absence of specific conditions.

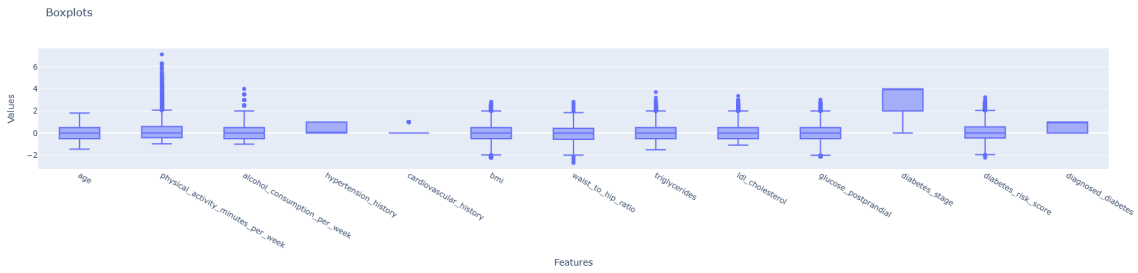

Figure 1: Boxplots - selected subset of features

Histograms and Density Plots. Histograms and smoothed distribution plots (e.g., kernel density estimates) were employed to inspect the shape of the distributions and the presence of skewness or multimodality. Supported by Figure 2 and Figure 3, the analysis shows that:

- Many physiological and biochemical measurements deviate from a Gaussian distribution, often exhibiting skewness and heavy tails;
- Lifestyle attributes, such as physical activity minutes per week and alcohol consumption, are highly skewed towards lower values;
- The composite diabetes risk score displays a unimodal but non-symmetric distribution, consistent with being an aggregate measure of several heterogeneous risk factors.

These distributional properties support the use of learning methods that do not rely on strong parametric assumptions (e.g., normality), such as tree-based models.

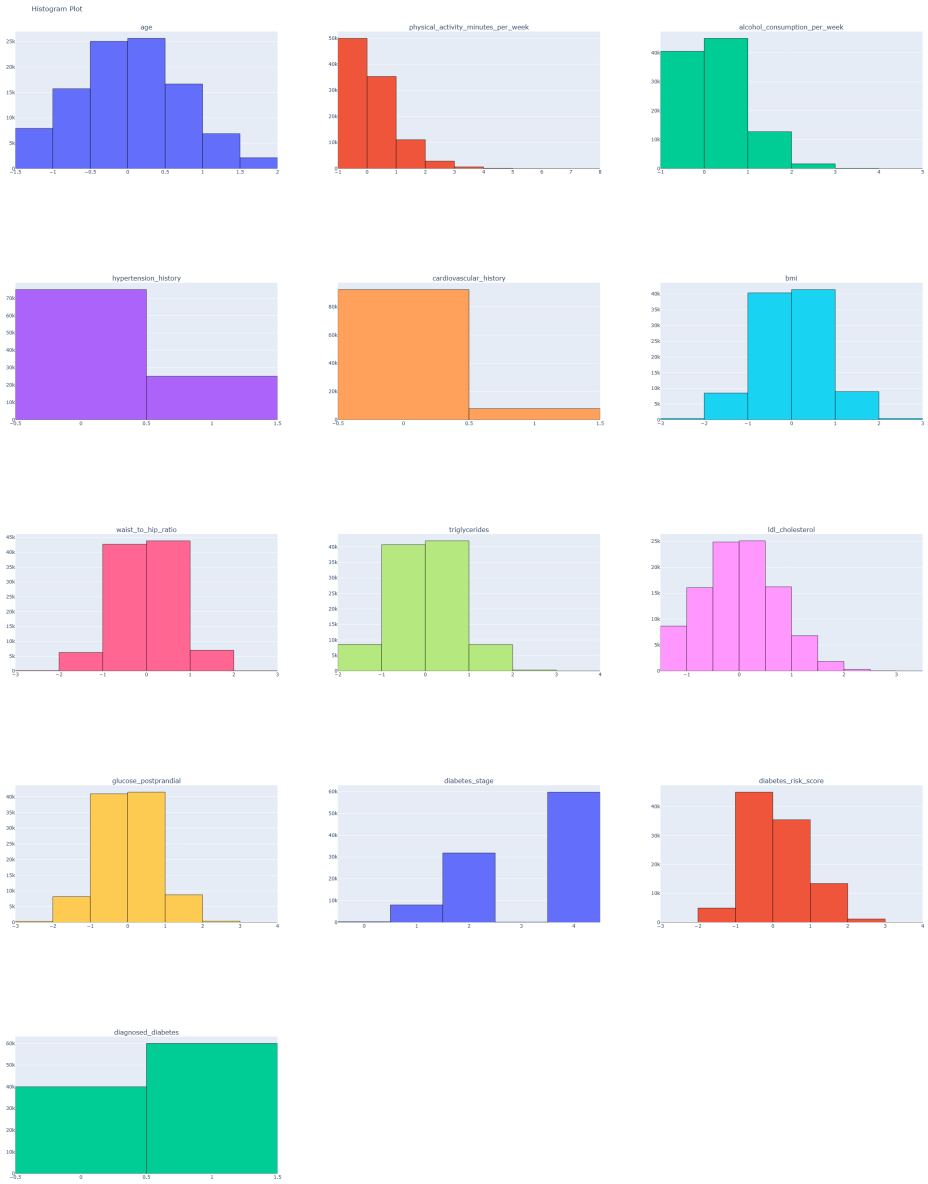

Figure 2: Histograms - Selected subset of features

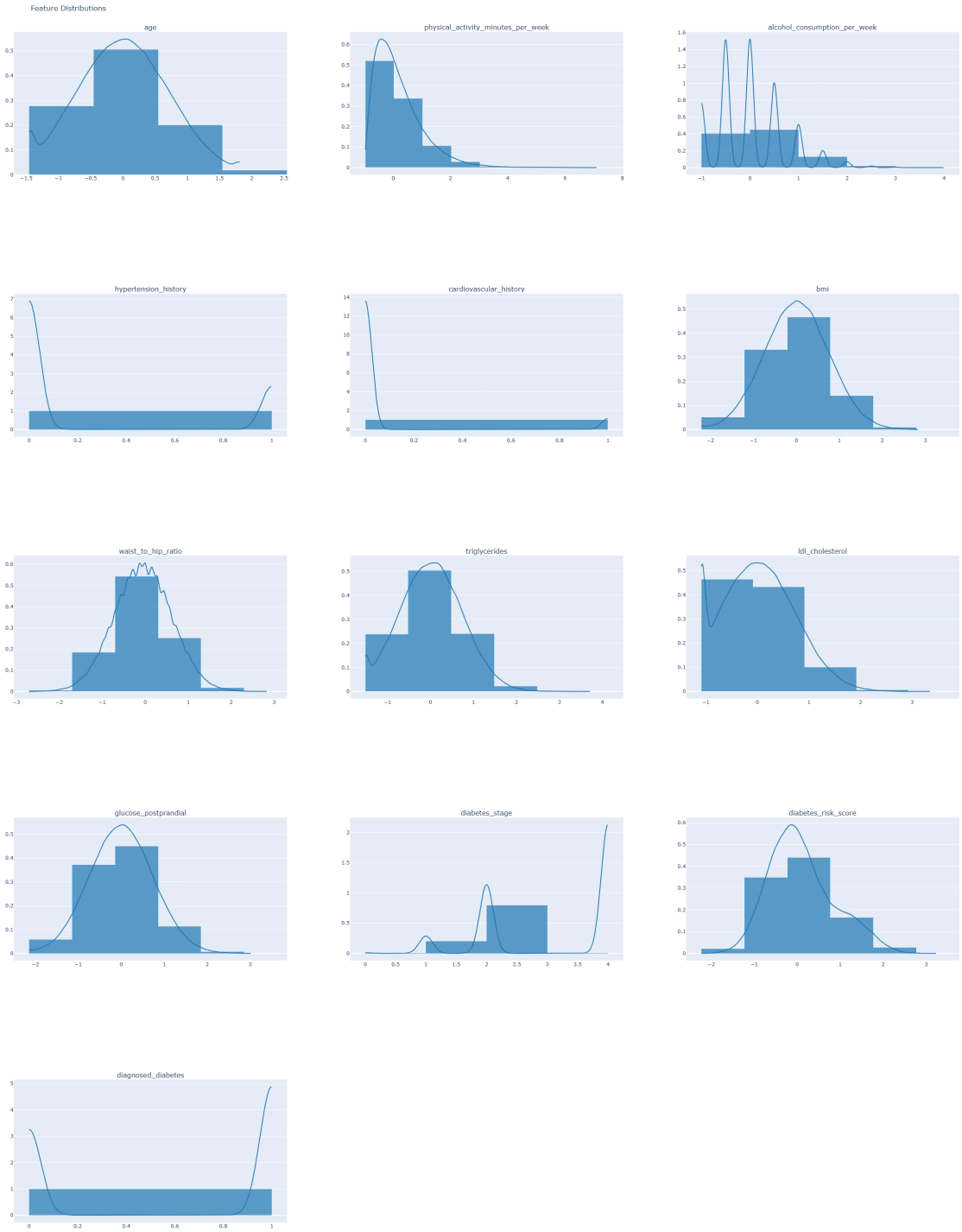

Figure 3: Distplots - Selected subset of features

1.2.3 Correlation Analysis

To quantify linear relationships between pairs of numerical variables and to detect redundancy among features, the Pearson correlation matrix was computed on the numerically encoded dataset.

Given two numerical features, X and Y, the Pearson correlation coefficient is defined as:

$$\rho_{X,Y} = \frac{\mathrm{Cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2}\sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(y_i - \bar{y})^2}}$$

The resulting correlation matrix shown in Figure 4 reveals:

- strong positive correlations within groups of related clinical measurements, such as total cholesterol, LDL cholesterol and triglycerides;
- pronounced associations between glycaemic biomarkers (fasting and postprandial glucose, HbA1c) and the diabetes risk score;
- moderate correlations between anthropometric indices (BMI, waist-to-hip ratio) and diabetes-related variables;
- generally weaker correlations between demographic or lifestyle variables and the more direct clinical indicators.

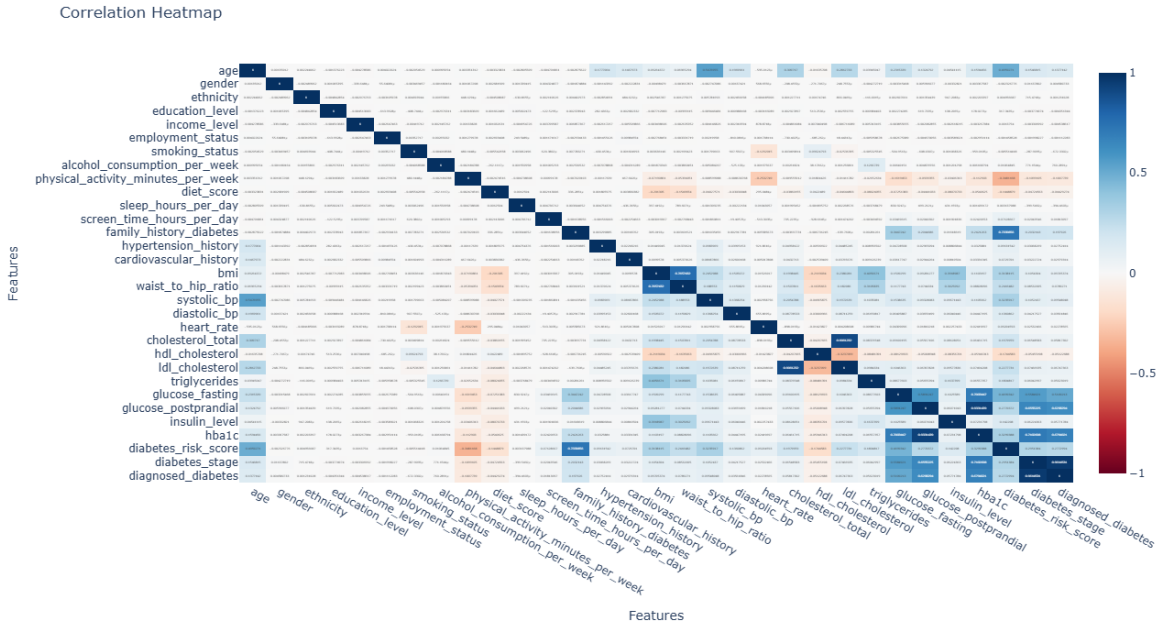

Figure 4: Correlation Matrix

This analysis highlights clusters of highly correlated variables, which may later be considered for dimensionality reduction or regularization, as they provide overlapping information.

1.2.4 Statistical Independence Testing

To assess the degree of association between each feature and the binary target `diagnosed_diabetes`, a chi-squared test of independence was performed. For each feature-target pair, a contingency table was constructed (using appropriate discretization where required), and the Pearson chi-squared statistic was computed.

Let the contingency table entries be O_{ij} (observed frequencies) and E_{ij} (expected frequencies under independence). The test statistic is:

$$\chi^2 = \sum_i \sum_j \frac{(O_{ij} - E_{ij})^2}{E_{ij}},$$

with an associated p-value obtained from the chi-squared distribution with the appropriate degrees of freedom.

The chi-squared analysis shown in Figure 5 indicates that:

- `family_history_diabetes`, `glucose_postprandial`, and `hba1c` display exceptionally larger χ^2 values, compared to all other features;
- This suggests a strong correlation between the target variable, `diagnosed_diabetes`, and strong indicators regarding diabetes diagnosis.
- A small subset of variables yields high p-values, suggesting limited predictive relevance with respect to the target and marking them as potential candidates for removal in more constrained models.

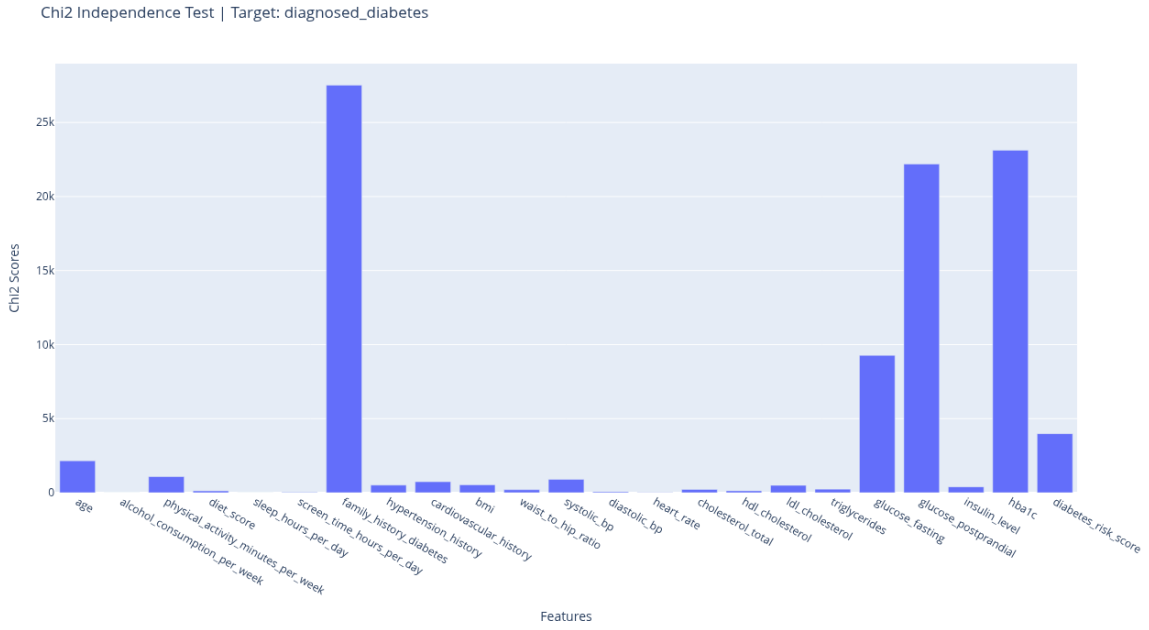

Figure 5: χ^2 Independence Test - Classification Target Variable

This statistical test complements the correlation analysis by directly targeting dependence with the outcome variable.

1.2.5 Feature Importance and Selection

To identify the most informative features and reduce dimensionality while preserving predictive power, two filter-based feature selection approaches were applied: variance thresholding and univariate scoring. A relief-based method was also considered but not ultimately used due to its high memory requirements for the given dataset size.

Variance Thresholding. Variance thresholding removes features whose variance is below a specified threshold θ, under the rationale that features with very low variability carry limited discriminatory information. Formally, a feature X is discarded if

$$\mathrm{Var}(X) < \theta$$

In this analysis, a threshold of $\theta = 0.2$ was employed on the numerically encoded dataset. Only a small number of features failed to satisfy this criterion, confirming that most variables show sufficient variation across instances.

Univariate Feature Selection. To estimate the individual discriminative power of each feature, a univariate F-test was applied. The test measures how well a feature separates the two classes by comparing the variance of its values *between* classes to the variance *within* classes. For a feature X, the F-statistic is defined as:

$$F = \frac{\text{variance between classes}}{\text{variance within classes}}$$

Features with a higher F-statistic (or equivalently, with a smaller associated p-value) provide stronger separation between diabetic and non-diabetic instances and are therefore considered more informative.

The F-test was accompanied by the Mutual Information test, described by the following formula:

$$MI(X,Y) = \sum_{x \in X} \sum_{y \in Y} P(X = x, Y = y) \log\left(\frac{P(X = x, Y = y)}{P(X = x) \cdot P(Y = y)}\right)$$

Mutual information tests how two variables may influence each other by comparing how the joint occurrence of both X and Y may occur by taking the events X and Y indepedently. The logarithm measures the information gained about X knowing Y and vice-versa, while the double summation sums up the information gain across the event horizon of X and Y

The resulting ranking present in Figure 6 is consistent with the medical intuition and with the chi-squared analysis:

- Glucose-related measures, HbA1c, and the composite diabetes risk score are among the top-ranked features.
- Several physiological variables (such as blood pressure and lipid parameters) also receive considerable scores;
- Certain demographic and lifestyle attributes obtain much lower scores, indicating a weaker direct contribution to class discrimination.
- The differences between the F-test and the Mutual Information test come from the assumption of the F-test that the dependencies are linear, which does not always happen in this case; Mutual Information is able to capture any kind of dependency between variables, thus resulting in slight differences in both classification and regression scenarios.

Figure 6: Feature Scores Comparison - Classification & Regression Target Variable

Following `Occam's razor` principle, Table 2 and Table 3 highlight the features selected through the feature selection process, using a threshold equal to 0.2 and a majority voting strategy. This narrows the feature space enough so that the model does not struggle to learn the representation of the sample space, but it also introduces the underfitting risk given the drastic reduction of the number of features.

Feature	**F-Score**	**Mutual-Info Score**	**χ^2 Value**	**Correlation**	**Variance**	**Mean Score**
hba1c	$1.000000e+00$	1.000000	$8.402536e-01$	0.679404	$9.297901e-05$	0.703950
glucose_postprandial	$7.669454e-01$	0.649412	$8.062800e-01$	0.629832	$1.343164e-01$	0.597357
glucose_fasting	$4.120398e-01$	0.375798	$3.369023e-01$	0.510919	$2.594257e-02$	0.332320

Table 2: Classification Task - Selected Features - Majority Voting

Feature	F-Score	Mutual-Info Score	Correlation	Variance	Mean Score
family_history_diabetes	1.000000	1.000000	0.733082	$2.403806e-05$	0.683277
physical_activity_minutes_per_week	0.118700	0.293177	0.348120	$1.000000e+00$	0.439999
age	0.280752	0.773999	0.495927	$3.417598e-02$	0.396214
glucose_fasting	0.243972	0.361410	0.469935	$2.594257e-02$	0.275315

Table 3: Regression Task - Selected Features - Majority Voting

1.2.6 Data Visualization through Component Reduction

To explore the intrinsic structure of the dataset in a non-linear, low-dimensional space and to visually assess potential separability between classes, the *Uniform Manifold Approximation and Projection* (UMAP) algorithm was applied to the numerically preprocessed feature matrix.

UMAP is a manifold-learning technique that constructs a weighted graph representing local relationships in the high-dimensional space and then optimizes a low-dimensional embedding that preserves these neighborhood structures as faithfully as possible.

UMAP was selected over alternative dimensionality reduction techniques such as Principal Component Analysis (PCA) and t-SNE due to its ability to capture complex, non-linear relationships that are characteristic of clinical and behavioral health data. PCA, being linear, cannot reveal complex relationships that may exist between variables (as is the case of metabolic markers, lifestyle factors, and physiological measures from the dataset), while t-SNE, although effective at preserving local neighborhoods, distorts global structure and scales poorly to large datasets. UMAP provides a more balanced preservation of local and global topology, offers faster computation on high-dimensional data of this size, and yields stable embeddings that align well with both the diabetes diagnosis and the continuous variation in the risk score. This makes it particularly suitable for visualizing potential class separability and population substructure in the Diabetes Health Indicators dataset.

Given a dataset $X = \{x_1, \ldots, x_N\}$, UMAP proceeds by:

1. Computing pairwise distances and building a fuzzy topological representation of the data manifold using k-nearest neighbors;

2. Constructing a low-dimensional graph whose edge weights are optimized to match the high-dimensional graph using stochastic gradient descent;

3. Producing a two- or three-dimensional embedding in which similar observations are placed close together, while dissimilar ones are pushed apart.

The resulting UMAP projections shown in Figures 7 and 2refdata-analysis:umap-regr3d reveal the following key patterns:

- **UMAP embedding colored by diagnosed diabetes (2D plot):** the projection cleanly separates the dataset into two distinct regions corresponding to the positive and negative classes. This indicates that the full multivariate feature space contains sufficiently strong signals, primarily driven by glycaemic measurements and metabolic indicators, to linearly or non-linearly distinguish diabetic individuals from non-diabetic ones.

- **UMAP embedding colored by diabetes risk score (3D plot):** the three-dimensional embedding displays a smooth, continuous gradient of color across each cluster, reflecting how the risk score varies gradually within the manifold. Higher-risk observations tend to group closer together, while lower values form separate, well-defined subregions, suggesting that the risk score aligns with the underlying geometry of the clinical and behavioral features.

- **Cluster substructure and manifold shape:** both embeddings exhibit characteristic "lobed" subclusters, a typical outcome of UMAP when high-dimensional neighborhoods compress into lower dimensions. These structures represent finer stratifications of the population, such as combinations of metabolic markers, lifestyle factors, or demographic attributes, while still maintaining the clear global separation driven by the diabetic status.

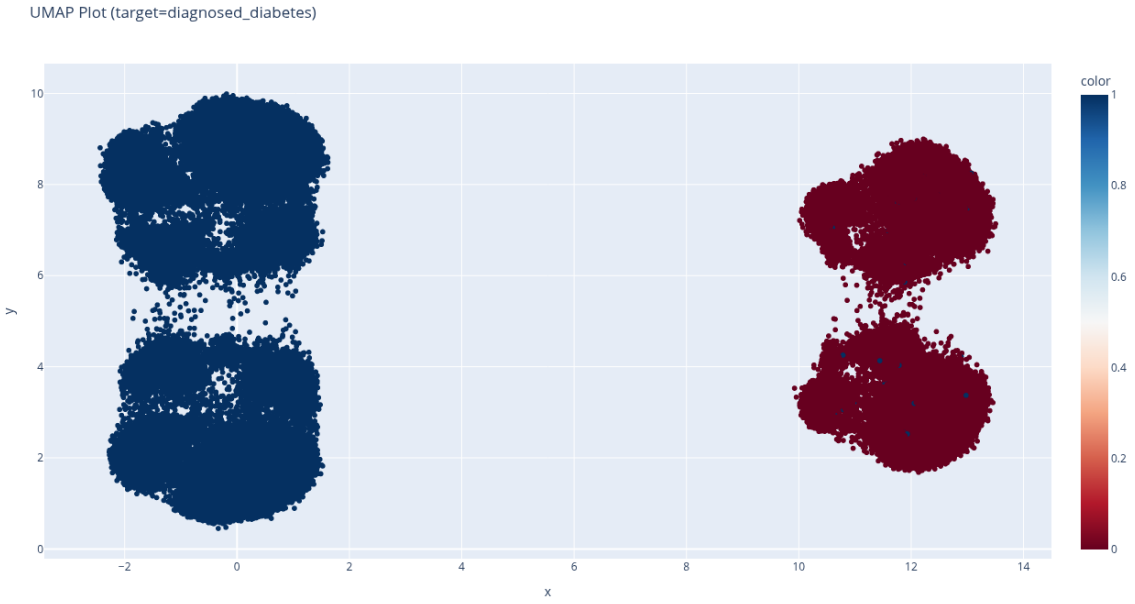

Figure 7: UMAP Visualization - Classification Setting

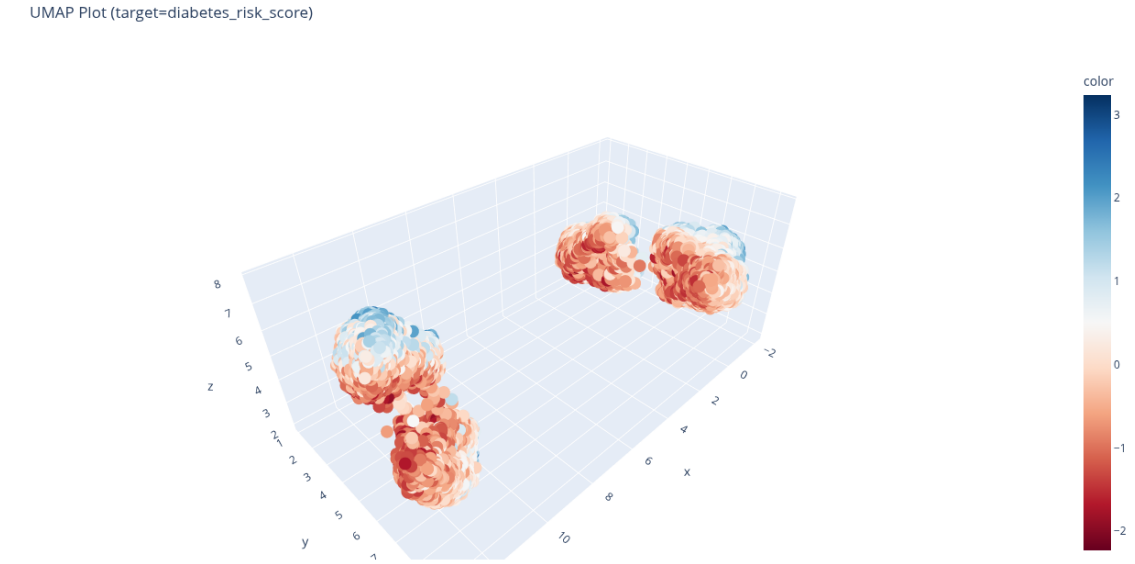

Figure 8: UMAP Visualization - Regression Setting

1.2.7 Summary of Findings

In summary, the data analysis experiment conducted comprises:

- **Descriptive statistics and dataset characterisation**, confirming realistic ranges and substantial variation for most clinical and lifestyle features.
- **Distributional analysis and visualization** via boxplots, histograms, and density plots, revealing skewness, heavy tails, and outliers in several key variables.
- **Correlation analysis** through a Pearson correlation matrix, highlighting strongly related groups of clinical measurements and suggesting potential redundancy.
- **Statistical independence testing** using chi-squared tests between each feature and the diabetes diagnosis, quantifying the strength of association with the target.
- **Feature importance and selection** using variance thresholding and univariate filter methods, supported by (but ultimately not extended with) a Relief-based approach.
- **UMAP visualizations** provide insightful component reduction results, suggesting patient profiles according to the reduced features.

Collectively, these analyses provide a comprehensive understanding of the structure and informativeness of the dataset. They confirm the central role of glycemic control indicators and related metabolic measures in predicting diabetes, while also clarifying the complementary and sometimes limited contributions of demographic and lifestyle variables. This knowledge directly informs the subsequent design and evaluation of the supervised learning models.

2 Supervised Regression

The purpose of this particular approach is to develop a framework capable of predicting a continuous *diabetes risk score* based on a set of individual health indicators described in Section 1. This predictive capability can assist in preventive healthcare and early detection, helping individuals and practitioners make informed medical decisions.

2.1 Problem specification

This problem proposes to solve the issue of predicting the diabetes risk score, based on social & health indicators. The *diabetes risk score* is a computed integer value based on the proposed social & health indicators, ranging between $[0, 100]$. Based on the proposed features, the solution will provide a *risk score* within the same range, indicating the likelihood of a patient having the medical condition.

2.1.1 Learning Task

This section presents the formalization of the learning task regarding the presented problem, accounting the features proposed to solve the problem and the goal provided by the solution. The specification of the learning task is presented below:

- **Task (T)**: Predict the diabetes risk score.
- **Performance (P)**: Quality of the resulted prediction, represented by how accurate is the prediction with respect to the real value.

- **Experience (E)**: Prediction of the diabetes risk score on a large set of labeled training instances.

The above described task will be solved through an empirical analysis, driven by data patterns rather than explicit programming.

2.2 Support Vector Regression

Unlike linear regression, Support Vector Regression (**SVR**) attempts to fit a set of data points by finding the optimal set of hyperplanes that maximizes the separation of classes with respect to a fixed threshold margin ε. Unlike the classical Support Vector Machine (**SVM**) framework, SVR defines a `"tube"` of width/radius $2 \cdot \varepsilon$ around the regression function. The errors within the tube are ignored, while errors outside are penalized in a linear manner; thus, the value ε highly influences the performance of the trained model.

2.2.1 Target function

In the context of supervised learning, algorithms assume that there exists an unknown and non-unique function f that accurately maps the input space X to the continuous output domain Y, where $Y \subseteq R^n$. Formally, such a function is expressed as:

$$f : X \to Y$$

where X represents the input space and Y represents the continuous output domain. In our regression setting, a sample $x \in X$ is represented by a subset of features selected during the data analysis phase, while $y \in Y$ is represented by a 1 by 1 vector encompassing the `diabetes risk score`.

2.2.2 Learning hypothesis

A hypothesis $h(x)$ is an approximation of the target function selected from a hypothesis space H. The hypothesis chosen as the backbone of the algorithm is the one that best approximates the target function: $h(x) \approx f(x)$.

For example, the hypothesis for linear-kernel-based SVR is defined as:

$$h(x) = \langle w, x \rangle + b$$

where w represents the weight vector (a not necessarily optimized normal to the hyperplane), $x \in X$ represents a single sample from the input space, b represents the bias term, and $\langle \cdot, \cdot \rangle$ represents the dot product in the input space, or feature space if the `kernel trick` is employed.

2.2.3 Representation of the learned function

The algorithms employed by SVR when trying to fit the data address two separate problems:

1. **Intuitive/Primal way**: finding the weights for each feature in order to predict the output; this method is hard to implement when the data to be fit is in a non-linear shape, as in kernel usage over the feature space; kernels are used to project the initial feature space into a non-linear shape by creating non-linear combinations from the initial features, followed by a back projection to the initial shape.

2. **Dual way**: instead of weighting features, this method implies the weighting of data points; this helps us to measure how similar a data point is compared to the ones used in the training phase; thus, the algorithm does not care about the features directly.

As for the representation of the learning function, we can express it as such:

$$h(x^*) = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) \cdot \langle x_i, x^* \rangle + b$$

where

- $h(x^*)$ represents the predicted value for a new input sample x;
- n is the total number of training samples;
- x_i represents a data point seen in training, while x^* represents a new data point, unseen in the training phase;
- $\langle \cdot, \cdot \rangle$ represents the `kernel function`, which measures the similarity between two data points; if the data points are very similar, the result will be large, compared to a small result when the data points are very dissimilar;
- α_i and α_i^* represents the `dual coefficients`, also named `Lagrange coefficients` or `importance weights`:
  - α_i: the regression line is pushed `up` because the error of the data point x_i was `below` the error tube;
  - α_i^*: the regression line is pushed `down` because the error of the data point x_i was `above` the error tube;
  - Following the principle of `Occam's razor`, data points that are considered a "good fit" because they are inside the error tube will have both coefficients equal to 0; these data points are irrelevant to the model.
  - Only those data points that are outside the error tube are relevant in the learning phase; thus, all data points $\alpha_i - \alpha_i^* \neq 0$ are the `Support Vectors`.

Therefore, the internal representation of the learning function will be stored as:

- a subset $SV \subset X$ of data points which are not inside the error tube;
- a vector of coefficients/weights β, where $\beta_i = (\alpha_i - \alpha_i^*)$; a positive β_i suggests that the data point pulls the regression line `up`, while a negative β_i suggests that the data point pulls the regression line `down`.

2.2.4 Learning algorithm

The learning algorithm is represented as an optimization procedure of a regularized risk function under the `dual formulation` setting.

1. **Goal**: finding a learning hypothesis that is as `"flat"` as possible, resulting in minimal model complexity, while keeping the prediction error in the training phase within an error threshold ε.
   - The loss function used in this process is an ε`-insensitive loss function`, which discards errors smaller than ε, which are those inside the error tube.

$$L_\varepsilon(y, h(x)) = \begin{cases} 0, if |y - h(x)| \leq \varepsilon \\ |y - h(x)| - \varepsilon, otherwise \end{cases}$$

- The minimization goal lies in the minimization of the regularized risk function:

$$F(w) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} L_\varepsilon(y_i - h(x_i))$$

where

  - $\frac{1}{2}\|w\|^2$ represents the regularization term; minimizing the norm of the weights keeps the error tube as "flat" as possible, while also preventing overfitting;
  - C represents the regularization parameter, controlling the trade-off between flatness and the degree to which deviations larger than ε influence the result;
  - $\sum L_\varepsilon$ represents the sum of all errors that fall outside the error tube.

2. **Optimization Algorithm**: since the intuitive objective function involves absolute values, it is difficult to compute due to non-differentiability at 0. By introducing `Lagrange multipliers`, we convert the initial problem into a `quadratic programming` problem known as the `Dual Problem`. Instead of minimizing the weights of each feature, the algorithm maximizes the following function with respect to the dual coefficients α^+, α^-, in order to maximize the margin between data points:

$$W(\alpha, \alpha^*) = -\frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\langle x_i, x_j\rangle - \varepsilon\sum_{i=1}^{n}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{n} y_i(\alpha_i - \alpha_i^*)$$

The implementation is performed according to the Sequential Minimal Optimization (**SMO**) algorithm:

   (a) Breaking the QP problem into the smallest possible sub-problems by selecting a pair of dual coefficients to optimize at each step;

   (b) Analytical optimization is possible because only two variables are variable at a given step; therefore, no matrix inversion is needed;

   (c) Convergence is guaranteed due to the algorithm iterating through dual coefficient pairs until the Karush-Kuhn-Tucker (**KKT**) conditions are satisfied, within a specified tolerance; the solution is considered globally optimal when the QP problem is as high as possible.

2.3 Related Work

Support Vector Machines have been extensively employed and validated in the medical domain, particularly in tasks regarding diabetes management. While early applications focused on binary classification, for example, various diagnosis applications, researchers have demonstrated the efficacy of this particular machine learning algorithm for predicting continuous physiological parameters, establishing a strong foundation for approximating glucose values or risk scores.

2.3.1 Mathematical Foundation of SVR in Medicine

SVR is suitable, from a theoretical perspective, for medical regression tasks due to its ability to handle non-linear attribute relationships while minimizing overfitting. As detailed by [ZO20], SVR differs from traditional regression by introducing an ε-insensitive loss function, as described in 2.2. The authors described how this loss function allows the model to ignore errors within a certain threshold (ε) while minimizing the weight vector $||w||^2$ in order to flatten the resulting error tube and to provide a smooth solution. Due to the critical nature of the approached problem, that being the sensitivity of health indicators, this handles the balance between model complexity and training error where noise and individual variability are common.

2.3.2 Continuous Glucose Prediction Application

Glucose forecasting for type I and type II diabetes patients is the most direct precedent for using SVR in medical prediction tasks. [HADC+18] successfully applied this technique in order to predict continuous blood glucose levels using **Continous Glucose Monitoring** (**CGM**) data. Their study highlighted that SVR, particularly when optimized with algorithms such as Differential Evolution (**DE**), could accurately model the non-linear fluctuations of glucose levels, especially by continuously improving a candidate solution with respect to a measure of quality (`glucose levels`), achieving this by employing various **metaheuristics**. The researchers achieved a Root Mean Square Error (**RMSE**) as low as 9.44 mg/dL for 15-minute prediction horizons.

Similarly, [GPA+13] utilized SVR to predict subcutaneous glucose concentrations to anticipate hypoglycemic events. Their research is especially relevant due to its ability to demonstrate how this algorithm can integrate multivariate inputs, such as insulin intake, meal-related data, and physical activity, to regress a future physiological state. They concluded that SVR outperformed other machine learning techniques, such as MLP neural networks, in specific diurnal prediction tasks, further validating the algorithm's robustness in handling multi-dimensional clinical attributes.

2.3.3 Relevance of Feature Sets

While the aforementioned papers focus on modeling regression tasks, the validity of using standard health indicators, such as age, BMI, or glucose, for diabetes modeling raises questions regarding their impact on the target variables. This is supported by classification literature, as [VHBCP20] applied SVMs to diagnosed diabetes in a Colombian dataset, a country that has the third highest number of adults (20-79 years) with diabetes in the South America region. While achieving 95.36% accuracy, they crucially identified that a compact feature set including age, BMI, and blood glucose was sufficient for high-performance modeling. This supports the feature selection typically found in diabetes risk datasets, including the dataset used in this study, and justifies the use of these specific variables for regression tasks.

2.3.4 Conclusion

Collectively, these studies demonstrate the potential that SVR poses for regressing continuous diabetes-related metrics. The transition from predicting blood glucose levels to predicting a diabetes risk score constitutes a change in the target variable but relies on the same underlying capacity of SVR to model non-linear interactions between demographic and physiological health indicators.

2.4 Experimental Evaluation

Following the specification of the supervised regression task, the theoretical presentation of the SVR algorithm, and the mentions of related work in this field, this section details the architecture of the model implemented from scratch, the experimental results obtained through cross-validation and hyperparameter optimization, and a discussion that analyzes the obtained performance, the explainability of the model, a comparison to related work, and lastly the implementation details of both **sklearn** and scratch SVR implementations.

2.4.1 Experimental Setup

Table 4 presents the configurable parameters for the SVR algorithm implemented from scratch, following closely the comprehensive description of the ML technique present in Section 2.2.

Parameter	Description	Domain
kernel	Kernel function used by SVR	{linear, poly, rbf, sigmoid}
C	Regularization parameter controlling the trade-off between flatness and training errors outside the ε-tube	$\mathbb{R}_+$
epsilon	ε-insensitive tube radius; errors within the tube do not contribute to the loss	$\mathbb{R}_+ \cup \{\texttt{auto}\}$
gamma	Kernel coefficient for `poly`, `rbf`, and `sigmoid`	$\mathbb{R}_+ \cup \{\texttt{scale}\}$
degree	Polynomial degree for `poly` kernel	$\mathbb{N}^*$
coef0	Independent term for `poly` and `sigmoid`	$\mathbb{R}$
tol	KKT violation tolerance used for stopping and for defining "meaningful" progress	$\mathbb{R}_+$
max_iter	Maximum number of SMO iterations	$\mathbb{N}^*$
max_passes	Maximum number of consecutive failed passes without progress before terminating	$\mathbb{N}^*$
cache_kernel	Cache the full kernel matrix $K(X, X)$ (memory intensive but faster updates)	{True, False}
kernel_batch_size	Batch size for chunked kernel computations when caching is disabled	$\mathbb{N}^*$
shrinking	Enables working-set shrinking during SMO	{True, False}
shrinking_interval	Interval (in iterations) between shrinking steps; also used for periodic unshrinking checks	$\mathbb{N}^*$
random_state	Random seed for tie-breaking and randomized selections	$\mathbb{N}$
verbose	Controls optimization logging verbosity	{True, False}

Table 4: SVR Scratch - Parameter Grid

The optimizer is based on Sequential Minimal Optimization (SMO) for ε-SVR. At each iteration, SMO selects a pair of dual variables and updates them while respecting feasibility: (i) box constraints for each variable and (ii) the equality constraint. In our implementation, the first index is chosen as the variable with the largest Karush-Kuhn-Tucker (**KKT**) violation computed from the dual gradient; the second index is chosen among variables with opposite sign (to preserve the equality constraint) and with a gradient direction that yields a productive step.

Kernel functions and hyperparameters. The implementation supports the standard SVR kernels: `linear`, `poly`, `rbf`, and `sigmoid`. The kernel parameters (γ, `degree`, `coef0`) follow the same interpretation as in common libraries. In addition, the model supports:

- **automatic** ε selection (`epsilon=auto`), to adapt the tube width to the scale of the target variable;
- kernel matrix caching (`cache_kernel=True`) when memory permits;
- kernel batching for memory-safe computations when caching is disabled;
- shrinking (working set reduction) to temporarily remove variables unlikely to violate KKT conditions and accelerate convergence.

In addition, the scratch SVR implementation stores the dual variables (α, α^*) in a contiguous vector: $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_1^*, \alpha_2^*, \ldots), idx(\alpha^*) = n + idx(\alpha)$, while also maintaining a sign vector $sign = \{\pm 1\}^{2n}$, such that $\alpha_i : sign_i = +1$ and $\alpha_i^* : sign_{i+n} = -1$. This allows writing the prediction in a compact form while still performing coordinate updates on α during optimization.

Convergence and support vectors. The SMO loop stops when either (i) no more optimizable KKT violations are found, (ii) the relative range of the most recent k maximum KKT violations falls below a minimum improvement threshold, (iii) the algorithm stalls and does not improve following consecutive un-shrinking of the support vector active set, or (iv) the number of maximum iterations or the maximum number of consecutive failure SMO steps is achieved. Following the optimization process, the fitting process ends in the following manner:

1. We extract the support vectors for which their $beta = \alpha - \alpha^*$ is non-negligible information; this ensures that we exclude any support vectors that do not offer any meaningful information.

2. For resource efficiency purposes, the model stores only the support vectors and their coefficients, as well as any non-vectorial variables, with prediction being done only on the shrunk vector set; this ensures that the model stays rather compact and helps with the speed of the inference process.

Algorithm 1 summarizes the main SMO routine used in the from-scratch SVR. Any theoretical steps were removed to keep the explanations compact enough.

Algorithm 1 SMO Optimization for scratch ε-SVR

1. Initialize $a \leftarrow 0_{2n}$, $sign = \{+1_1, +1_2, \ldots, +1_n, -1_{n+1}, -1_{n+2}, \ldots, -1_{2n}\}$ $b \leftarrow 0$, error cache $e_i \leftarrow -y_i$.

2. Optionally cache $K(X, X)$; otherwise compute kernels on demand (resource constrained).

3. Resolve γ according to strategy or mathematical value, resolve ε according to strategy or mathematical value.

4. Repeat until convergence, iteration limits, or failure step limits:

   (a) Compute dual gradients (KKT signals) from the current error cache and ε.

   (b) Select index i with maximal KKT violation.

   (c) Select index j with opposite sign (to preserve $\sum(\alpha - \alpha^*) = 0$) and compatible gradient direction.

   (d) Compute feasible bounds for the pair and update (a_i, a_j).

   (e) Update bias b and incrementally update error cache e.

   (f) If shrinking is enabled: periodically shrink the active set; unshrink and re-check if necessary.

5. Extract support vectors: $\beta_i = \alpha_i - \alpha_i^*$, keep indices with $|\beta_i| > \tau$.

The necessary KKT conditions in our case are defined by:

1. $\alpha \approx 0, gradient \leq 0$: the coefficient is close to 0 and it wants to decrease, so no optimization can be applied;

2. $\alpha \approx C, gradient \geq 0$: the coefficient is close to the box bounds defined by the C regularization parameter, and it wants to increase, so no optimization can be applied.

These conditions ensure that we select the most appropriate coefficient to be optimized, ensuring also that we do not select any coefficients that cannot benefit from further optimization.

The shrinking applies to those support vectors that are near bounds and do not show any optimization potential, because they would get out of bounds. This ensures that the algorithm does not enter an infinite loop and prevents noisy vectors from blocking the optimization step.

Regarding early stopping and performance optimizations:

- We have limited the data type for array-specific operations to single-precision floating-point numbers in order to prevent out-of-memory crashes.
- Early stopping occurs with respect to the objective function, allowing the model to converge faster if improvements of the KKT violations are not significant; a violation window of at most 10 violations and an improvement threshold of 5% were used, ensuring an acceptable range of maximum violations and an acceptable improvement ratio to improve on.

The general cross-validation and hyperparameter optimization pipeline is detailed in Section 2.4.2, which includes a detailed discussion on the obtained results, the grid search cross-validation scratch implementation, and how parameters of this model are selected to provide the best results possible.

The above-presented model architecture respects the "from scratch" implementation constraints and also respects the mathematical foundations presented in [DBK+96]. Purely implemented in Python, this architecture represents a venerable alternative to **sklearn**'s SVR implementation, which will be detailed later in this section.

2.4.2 Experimental Results

We evaluate the from-scratch SVR implementation under a supervised regression protocol using a 80%/20% train/test split. All hyperparameter tuning is performed *only* on the training portion of the dataset using a k-fold cross-validation, and the final reported test metrics are computed on the held-out test split.

Hyperparameter optimization is implemented via an exhaustive grid search cross validation. Each candidate configuration is trained on each fold and evaluated using appropriate regression metrics. The best model, selected by a primary refit metric (RMSE in this case), is refit on the full training split using the best cross-validation configuration. This methodology is applied identically to our from-scratch SVR and to **sklearn**'s SVR to ensure a fair comparison.

Cross-validation grid. For both **sklearn** and scratch implementations of SVR, we have provided the parameter grid described in Table 5. The SMO tolerance and epsilon radius were left with default values, specifically: $tol = 1e-3$, $\epsilon = auto$. The ϵ parameter scales according to the standard deviation of the target variable instead of keeping it fixed so that the model tries to fit the insensitive error tube according to the domain of the target variable. A `null` *gamma* resume is a *scale* gamma resolution approach. Each instance of the from-scratch SVR model was trained with 50 maximum failure passes, 50000 maximum iterations, applied shrinking at 50 iterations, and a kernel computation batching size of 5000. The kernel was not cached due to limited computational resources.

Due to the fact that our data did not show any sigmoidal shape, we chose to exclude it $kernel = sigmoid$ from the parameter grid, as this kernel has very specific use cases, like the one mentioned before. This type of kernel tries to fit the data within a complex boundary and might not generalize well on unseen instances, making it less popular than other kernels.

Parameter	Value Space
kernel	{linear, poly, rbf}
C	[0.5, 0.75, 1, 1.5]
gamma	[scale]
degree	[3, 4, 5]
coef0	[0.0, 0.5, 1.0]

Table 5: SVR - Cross-Validation Parameter Grid

We used 5 folds to perform the cross-validation fitting, along with 44 candidates resulting from the parameter grid. The actual number of candidate, 44, is lower than the total number of configurations 81, due to grid pruning in our configurations, which removed combinations that are not relevant. For example, a linear kernel does not use *gamma*, *degree*, or $coef0$, while an RBF kernel uses just *gamma* and $coef0$.

Result interpretation. Table 6 reflects the results of the best from-scratch SVR model that resulted from the cross-validation pipeline: $SVR(kernel = rbf, C = 1.0, gamma = scale, epsilon = auto)$.

Metric	Value
Mean Absolute Error (**MAE**)	2.400320
Mean Absolute Percentange Error (**MAPE**)	9.253411%
Mean Squared Error (**MSE**)	8.877404
Median Absolute Error	2.050436
Root Mean Squared Error (**RMSE**)	2.979497
Normalized Root Mean Squared Error (**NRMSE**)	0.098380
Coefficient of Determination (R^2)	0.892843
Mean Squared Log Error (**MSLE**)	0.012509
Root Mean Squared Log Error (**RMSLE**)	0.111847

Table 6: SVR Scratch - Best Estimator Performance Results

Statistical analysis of cross-validation results. Along with the classical evaluation metrics for regression, Table 7 contains a collection of statistical values of cross-validation fold scores. We used the Shapiro-Wilk test in order to test whether the fold scores follow a normal distribution or not and the Student's t-distribution to compute a confidence interval for the mean of the fold scores using a 95% confidence level.

Key	Value
Refit Metric	RMSE
Best Model Parameters	$\{kernel = rbf, C = 1.0, gamma = scale, epsilon = auto\}$
Fold Scores	$(5.035965, 4.370107, 4.460083, 4.646090, 2.912629)$
Mean	4.284975
Standard Deviation	0.808646
Standard Deviation Error	0.361637
Median	4.460083
Min	2.912629
Max	5.035965
IQR	0.275983
Q1	4.370107
Q3	4.646090
Coefficient of Variation	18.871666
Number of Samples	5
Confidence Interval - Lower Bound	3.280908
Confidence Interval - Upper Bound	5.289042
Confidence Interval - Confidence Level	0.95
Confidence Interval - Point to Estimate (Mean)	4.284975
Confidence Interval - Margin of Error	1.004066
Normality Test - Statistic	0.835010
Normality Test - p-Value	0.151595
Normality Test - Is Normal?	*True*
Normality Test - α	0.05

Table 7: SVR Scratch - Statistical Analysis of Cross-Validation Folds

2.4.3 Discussion

The previous section presented the cross-validation parameter grid, the performance results of the best estimator resulted from exhaustive grid search cross-validation, and a statistical analysis of the cross-validation results with respect to the fold scores of this resulting best estimator.

Discussion on achieved results. We consider the results presented in Table 6 impressive, considering the from-scratch implementation of SVR and the collection of problems that can be encountered in the development of such a solution. The value range of the test target split is $[4, 62]$, encompassing 58 units which should account for most typical clinical cases.

- The `median absolute error` suggests that half of the predictions have an error of ≈ 2.05 points or less; this can be considered as the baseline performance of the model, resulting in a highly accurate model for the vast majority of patients.
- A `MAE` of ≈ 2.40, which is greater than the baseline error of ≈ 2.05 by $\approx 14.58\%$, confirms that the error distribution is possibly right-skewed; in other words: the model is generally precise, but a small subset of this dataset contains some samples for which the SVR struggles.
- A `MSE` of ≈ 8.87 and a `RMSE` of ≈ 2.98 provide a general idea on how some errors affected the model's performance and how it generally behaves on inference; a `MSE` value of ≈ 9, combined with the other metrics, suggests that generally the errors are small, but those few errors around ≈ 11 heavily impacted the model's performance; on the other hand, `RMSE` essentially represents the standard deviation of the computed errors, and an error band of $\approx \pm 3$ suggests that the model is stable.
- A `NRMSE` of ≈ 0.098 measures the prediction quality of the model acceptably well; it indicates that the `RMSE` represents only $\approx 9.8\%$ of the data's central tendency, resulting in actually lower errors in practice.

- Given the low minimum value of our target test split (≈ 4.2), a `MAPE` of $\approx 9\%$ suggests that the model does not fail catastrophically with low-risk patients; otherwise, this percentage would be much greater.
- A `MSLE` of ≈ 0.012 and `RMSLE` of ≈ 0.112 capture the order of magnitude of the risk score predicted especially well; these two metrics dampen the effect of outliers and focus on relative error, indicating that the resulting model is unbiased.
- As for the "goodness of fit", a R^2 score of $\approx 89.3\%$ suggests that the model successfully explains $\approx 89.3\%$ of the variance in the risk score range; this demonstrates that the from-scratch implementation of the SVR model has a high predictive fidelity.

As for the cross-validation results presented in Table 7, we conducted an interpretation in the following manner:

- The `mean RMSE` of ≈ 4.28 represents the expected generalization error of the model when trained on partial data. Notably, this is higher than the final test RMSE (≈ 2.98), indicating that the model benefits significantly from the increased training size during the final refit phase.
- A `standard deviation` of ≈ 0.81 and a `coefficient of variation` of $\approx 18.87\%$ highlight a moderate degree of variance in model performance. This sensitivity suggests that the distribution of "hard-to-predict" outliers (discussed in the test results) is not uniform across all folds.
- The gap between the `min RMSE` (≈ 2.91) and `max RMSE` (≈ 5.04) is significant. The best-performing fold achieved an error nearly identical to the final test set (≈ 2.98), confirming that under favorable data distributions, the scratch implementation achieves optimal convergence.
- The `95% confidence interval` ranges from $[3.28, 5.29]$. We can assert with 95% confidence that the true average error of the model falls within this band. The fact that our final test error (2.98) lies slightly below this interval is a strong indicator of successful model tuning and refitting.
- Finally, the `normality test` yields a pp-value of ≈ 0.15 ($p > 0.05 = \alpha$), confirming that the distribution of error scores across folds is Gaussian. This validates the reliability of the mean and standard deviation metrics used above.

Model interpretability. Regarding model interpretability, we employed a method for visualizing the support vectors, training data, error tube, and the overall predictions of some SVR model. In order to provide a meaningful representation of the support vectors, we reduced the dimensionality of the input test split to 2 components using Principal Component Analysis (**PCA**). This enabled us to visualize how the support vectors behave and how they are spread out on the decision plane. Figure 9 contains this visualization, where we can observe that the model splits the data acceptably well, where the red dashed line represents the boundary of the epsilon tube. Besides that, we used SHapley Additive exPlanations (**SHAP**) to resolve around feature importance and overall principal driving factors of the prediction. Figure 10, Figure 11, Figure 12, and Figure 13 showcase the following SHAP results:

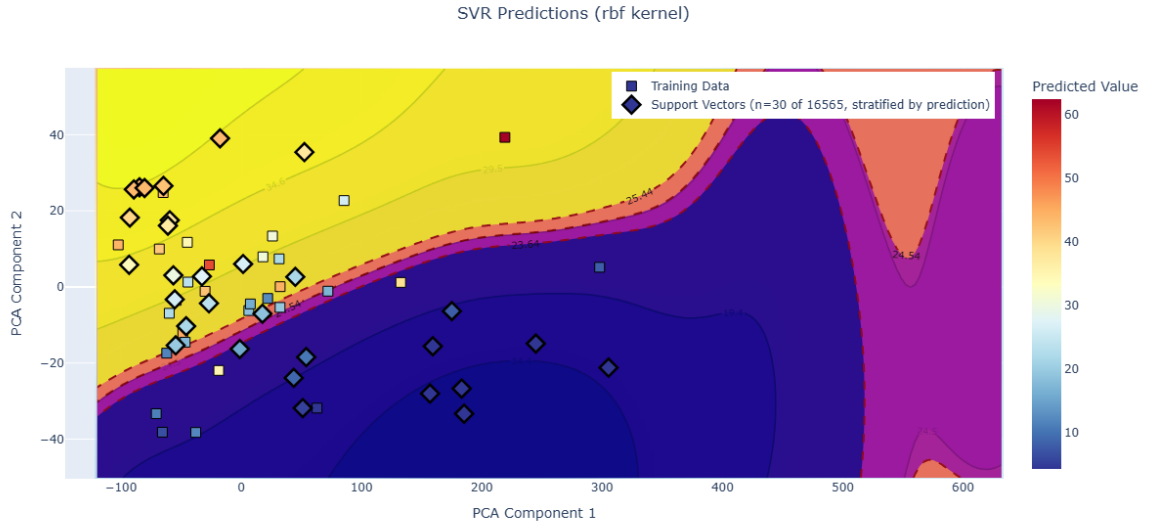

Figure 9: SVR Scratch - Support Vectors Visualization

- Figure 10 - Beeswarm Plot: highlights how each feature value affects the final results of the model; we can see that as age increases, so does the risk of diabetes, which is expected because there is rarely diabetes risk in young people.
- Figure 11 - Bar Plot: contains the overall influence of each feature used in the prediction process; the history of diabetes in one individual's family has great influence, as do age and the physical activity minutes per week; however, the glucose fasting does not have any clear influence over the model's decision-making process, typically because the amount of sugar in blood increases as we age.
- Figure 12 - Waterfall Plot: provides the SHAP values for a single prediction, enabling a local analysis on specific samples that prediction was done on; for this specific risk score of $f(x) \approx 34.494$ and the value that the model expects $E[f(x)] \approx 30.922$, we see that the physical activity minutes per week increase the risk as a history of diabetes in the family decreases it, but age here is the primary risk factor.
- Figure 13 - Dependence Plot: provides an idea of how features influence each other in the final prediction process; for this plot we've chosen to visualize the dependence of glucose fasting on age in order to justify that the amount of sugar increases as people get older.

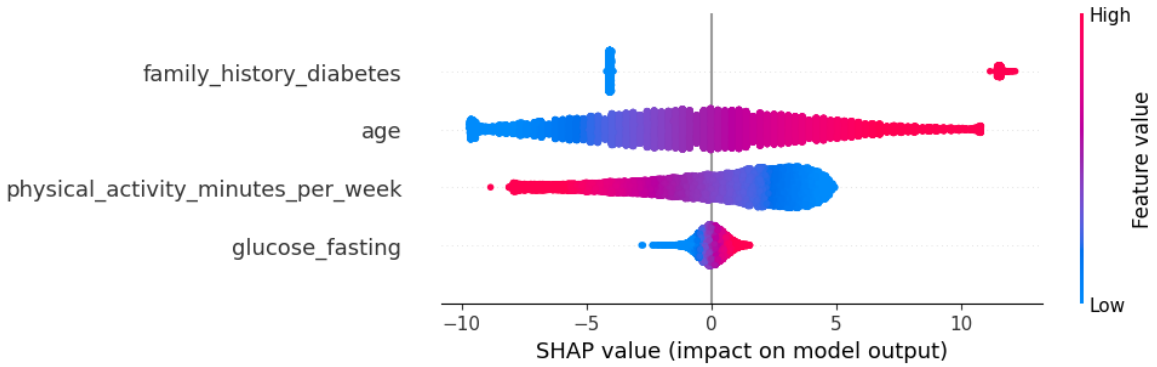

Figure 10: SVR Scratch - SHAP Beeswarm

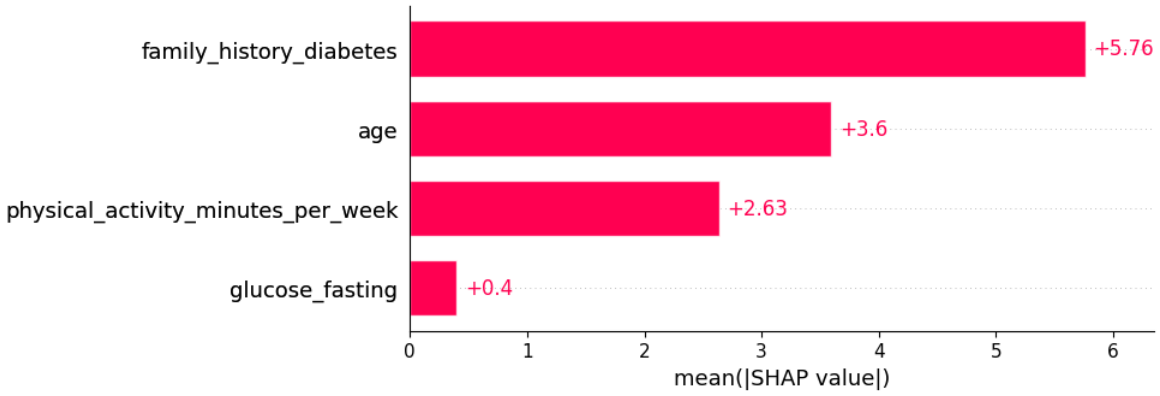

Figure 11: SVR Scratch - SHAP Bar

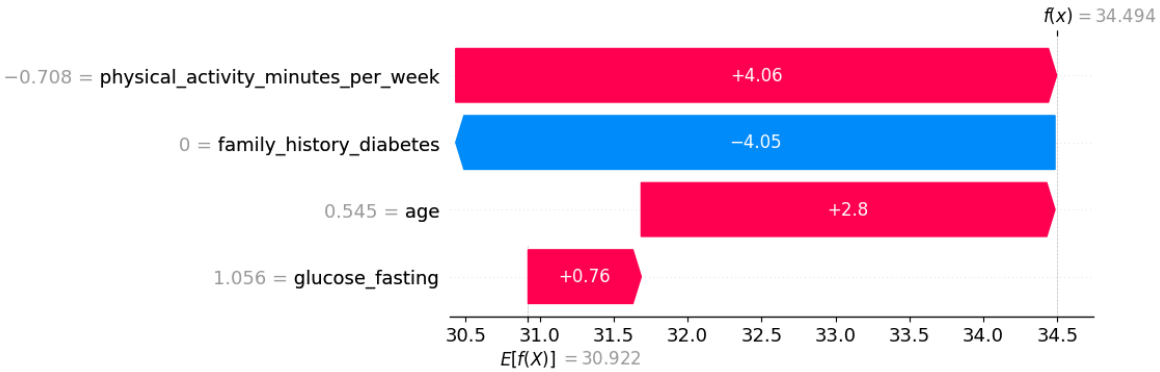

Figure 12: SVR Scratch - SHAP Waterfall

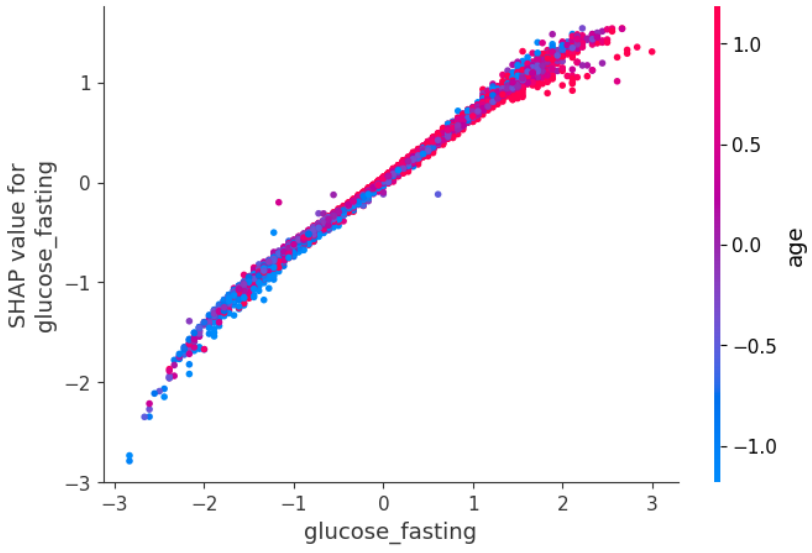

Figure 13: SVR Scratch - SHAP Dependence - Glucose Fasting vs. Age

Comparison to related work. Unfortunately, the studied data collection is mostly a synthetically generated one; therefore, we did not find any relevant articles that employed SVR for a regression task specifically for predicting the `diabetes_risk_score` target variable. Thus, Table 8 contains a comparison between performance evaluations of the **sklearn**

implementation of SVR, the scratch implementation of SVR, and relevant literature, with all results having the task to be learned specified. We have also referred to classification tasks that used SVMs for classification purposes to understand the ability of this algorithm to perform on multiple domains.

Model	Learning Task	MAE	MAPE	MSE	RMSE	R^2	MSLE	RMSLE	Accuracy	Sensitivity	Specificity	PPV	NPV
sklearn	Section 2.1	2.152873	7.862774%	7.303286	2.702459	0.911876	0.009594	0.097953	-	-	-	-	-
scratch SVR	Section 2.1	2.400320	9.253411%	8.877404	2.979497	0.892843	0.012509	0.111847	-	-	-	-	-
[HADC$^+$18]	Blood Glucose Prediction	-	3.7416%	-	9.4483	0.9715	-	-	-	-	-	-	-
[GPA$^+$13]	Hypoglycemic Event Prediction	-	-	-	-	-	-	-	-	$0.82 \pm 0.32, 0.78 \pm 0.37$	-	-	-
[VHBCP20]	Diabetes Diagnosis	-	-	-	-	-	-	-	0.9536	0.9436	0.9532	0.9436	0.9389

Table 8: SVR - Comparison to Related Work

sklearn's SVR performance is actually close to our from-scratch SVR, which is impressive. The library baseline model is slightly more performant than our implementation, but not by far, suggesting that we correctly implemented the algorithm that **sklearn** also uses. [HADC$^+$18] proposed a blood glucose prediction learning task, where they reported `MAPE`, `RMSE`, and R^2, having larger values, especially for `RMSE`. This suggests that they have a higher error, all while having a good performance reflected by high R^2 score. The authors in [HADC$^+$18] referred to articles written by researchers in [GPA$^+$13], the latter having better results than they achieved. Having mentioned [GPA$^+$13], they proposed a hypoglycemic event prediction task, where they achieved a sensitivity of about 82%, an acceptable metric regarding event forecasting. Lastly, a classification task in [VHBCP20] achieved surprising results using SVM, posing as an appropriate solution for diabetes diagnosis.

Scikit-learn Implementation Details. To validate the performance of our solution, we benchmarked it against the standard `sklearn.svm.SVR` estimator. This estimator is, in essence, a Python wrapper around the highly optimized `libsvm` C++ library developed by [CL11]. Specifically, it implements the Epsilon-Support Vector Regression (ϵ-SVR) algorithm, addressing the quadratic programming (QP) optimization problem via Sequential Minimal Optimization (SMO). It solves the dual formulation to identify support vectors within the ϵ-tube, explicitly leveraging the kernel trick to map input vectors into high-dimensional feature space.

It is worth noting the computational complexity of this industry-standard implementation. Since the solver relies on a kernel matrix, the fit time complexity scales between $O(n^2_{samples} \cdot n_{features})$ and $O(n^3_{samples} \cdot n_{features})$, depending on cache efficiency and dataset sparsity. Due to its expensiveness on large datasets, such as those with more than $100,000$ samples, we completed the training process of this estimator in approximately 50 minutes.

We employed the exact same experimental methodology for this estimator as for the from-scratch implementation. The same parameter grid (Table 5), number of folds ($k = 5$), and refit strategy were used to ensure a direct and fair comparison. Following the grid search, the best **sklearn** estimator was identified with the following hyperparameters: $SVR(kernel = poly, C = 1.0, degree = 4, coef0 = 1.0, gamma = scale, epsilon = 0.1)$. The results of this library baseline estimator are reported in Table 8.

2.4.4 Conclusion and Future Directions

Concluding the study on Support Vector Regression, we proposed a base algorithm that works generally well on regression tasks, at least when predicting risk scores in clinical settings. We achieved a ± 3 margin of error between predictions and actual values, resulting in a capable model that fits data accurately. We used **sklearn** as a baseline, and we achieved results comparable to an optimized version that was fine-tuned for years, confirming that our algorithm respects the modern technical requirements and expectations.

However, there are potential future directions that we would like to work on next:

1. Optimizations of the from-scratch SVR algorithm: porting to other programming languages that are faster than Python, optimizations of the mathematical operations.

2. Further improving the automation of feature selection: we suspect that our model underperformed, although we have used a 0.2 threshold to select features through a majority strategy.

3. Experiment with from-scratch SVR in other settings: extend the SVR application domain to other fields, not just the clinical one.

3 Supervised Classification

The goal of this task is to build a model capable of determining whether an individual is diabetic or non-diabetic, based on a set of health-related indicators described in Section 1, that have been demonstrated to have an impact in the development of this specific medical condition.

3.1 Problem specification

The target variable of this task is categorical, representing two possible outcomes, in the form of discrete values: *diabetic* patient (positive class), or *non-diabetic* patient (negative class). The model is therefore expected to predict a binary value $\{0, 1\}$, corresponding to the two output classes, alongside an associated probability score expressing the model's confidence.

3.2 Learning Task

- **Task (T)**: Predict if a patient is diabetic or non-diabetic.
- **Performance (P)**: Model's capability of correctly predicting the health status of a patient. Can be expressed as accuracy, precision, recall, F1-score, or AUC-ROC.
- **Experience (E)**: A set of labeled training examples, representing pairs of health indicators and the correct diabetes diagnosis. The model learns from these examples to generalize predictions on unseen data.

3.3 Random Forest Classification

Random Forests [Bre01] are ensemble learning methods that combine multiple **Decision Trees** (DTs) weak learners into a single predictive model to achieve higher accuracy and improved robustness compared to individual trees. They are based on the principles of *Bagging* (Bootstrap Aggregating) [Bre96] and *random feature subspacing*, both of which introduce diversity among the trees in the ensemble and therefore reduce variance.

Decision Trees themselves are eager inductive learners that recursively partition the feature space based on splitting criteria such as *Information Gain* (Entropy) or *Gini impurity*. However, individual trees tend to overfit and are sensitive to noise. Random Forests overcome these limitations by combining many decorrelated trees, producing a single strong learner with significantly improved generalization performance.

3.3.1 Target function

As with all supervised classification tasks, we assume the existence of an unknown target function:

$$f : X \to Y,$$

where $X \subseteq \mathbb{R}^n$ represents the feature space and $Y = \{0, 1\}$ corresponds to the diabetes label (non-diabetic or diabetic). Each training example is a pair (x_i, y_i) drawn from an underlying joint distribution $P(X, Y)$.

The goal of Random Forest learning is to approximate f with a hypothesis h that achieves high predictive performance and generalizes to unseen samples.

3.3.2 Learning hypothesis

In Random Forest classification, the hypothesis space H consists of *ensembles of decision trees*, where each individual hypothesis $h_t \in H_{\text{tree}}$ corresponds to a single classification tree trained on a bootstrap sample of the dataset and using random subsets of features at each split. Since Decision Trees are high-variance learners, each h_t provides only a rough approximation of the target function:

$$h_t(x) \approx f(x),$$

but their errors tend to be uncorrelated when bootstrap sampling and feature subspacing are used. The Random Forest leverages this property by combining many such weak hypotheses into a single, more stable ensemble hypothesis.

Formally, a Random Forest hypothesis h is defined through an aggregation function Φ that combines the outputs of T decision trees:

$$h(x) = \Phi\big(h_1(x), h_2(x), \ldots, h_T(x)\big).$$

In the classical Random Forest formulation used for binary classification, Φ corresponds to the **majority voting** rule. Let $Y = \{0, 1\}$ be the label set. For each class $c \in Y$, define the vote count:

$$V_c(x) = \sum_{t=1}^{T} \mathbf{1}[h_t(x) = c]\,,$$

where $\mathbf{1}[\cdot]$ is the indicator function. The ensemble prediction selects the class receiving the most votes:

$$h(x) = \arg\max_{c \in \{0,1\}} V_c(x).$$

In the particular case of binary classification, where each tree outputs a hard binary label $h_t(x) \in \{0, 1\}$, the argmax formulation reduces to the equivalent familiar majority-threshold rule:

$$h(x) = \begin{cases} 1 & \text{if } \displaystyle\sum_{t=1}^{T} h_t(x) \geq \frac{T}{2}, \\ 0 & \text{otherwise.} \end{cases}$$

This follows because $V_1(x) = \sum_{t=1}^{T} h_t(x)$ and $V_0(x) = T - V_1(x)$, so selecting the class with the maximal vote count reduces to checking whether class 1 receives at least half of the votes.

Therefore, the Random Forest hypothesis represents a consensus classifier that aggregates multiple tree hypotheses to obtain a more robust approximation of the underlying target function f.

3.3.3 Representation of the learned function

The learned Random Forest model is represented as an ensemble of T independently trained decision trees. Each decision tree h_t is stored as a hierarchical structure consisting of internal decision nodes and leaf nodes. An internal node contains:

- the selected splitting feature,
- the split threshold (for continuous attributes) or branch conditions (for categorical attributes),
- pointers to its child nodes.

Each leaf node stores the predicted class label or, depending on the implementation, the empirical class distribution of the training samples that reach that leaf. Due to bootstrap sampling and random feature subspacing, the trees in the ensemble differ in both structure and selected features, which ensures model diversity.

Therefore, the final learned function is represented internally as the collection

$$\mathcal{T} = \{h_1, h_2, \ldots, h_T\},$$

where each h_t is a full decision tree with its own feature tests, thresholds, and leaf predictions. The Random Forest stores no additional parameters: all predictive information is encoded in the structures of the individual trees.

3.3.4 Learning algorithm

The Random Forest learning algorithm follows the Bagging paradigm described in the ensemble learning framework and constructs an ensemble of decision trees using bootstrap sampling and random feature subspacing. The training process consists of three main stages:

1. **Bootstrap Sampling**: For each tree $t = 1, \ldots, T$:
   - draw a bootstrap dataset D_t by sampling n training instances with replacement;
   - instances not included in D_t become out-of-bag samples for internal validation.
2. **Tree Induction (Weak Learner Construction)**: Each tree h_t is trained greedily:
   - at each node, select m random features from the full set of n;
   - choose the best split using an impurity measure such as Gini index or entropy;
   - recursively grow the tree until a stopping criterion is met (pure leaves, maximum depth, or minimum samples).
3. **Aggregation (Ensemble Output)**: After all trees are trained, the forest prediction of the model represented as $\mathcal{T} = \{h_1, h_2, \ldots, h_T\}$ is obtained by majority voting over individual tree predictions, as described in Section 3.3.2.

A formal description of the training procedure is provided in Algorithm 2.

Algorithm 2 Random Forest Training Algorithm

Input: Training set D, number of trees T, feature subset size m
Output: Ensemble of decision trees $\mathcal{T}$

$\mathcal{T} \leftarrow \emptyset$
for $t = 1$ to T **do**
    Draw bootstrap sample D_t from D
    Initialize empty tree h_t
    while stopping criterion not met **do**
        Select random subset of m features
        Compute the best split using Gini or entropy
        Split node and create child nodes
    end while
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{h_t\}$
end for
    return $\mathcal{T}$

3.4 Related Work

Random Forest (RF) classifiers have emerged as a cornerstone of ensemble learning in medical diagnostics. While other algorithms focus on single-model optimization, Random Forests leverage a *wisdom of the crowd* approach, combining multiple decision trees to achieve superior predictive accuracy. This ensemble strategy has proven particularly effective in classifying chronic conditions like diabetes, where the interaction between lifestyle and physiological markers is multifaceted and non-linear.

3.4.1 Mathematical Foundation of Random Forests

Theoretically, the Random Forest algorithm is an extension of the Bootstrap Aggregation (bagging) that addresses the inherent limitations of individual Decision Trees weak learners, specifically their tendency to overfit noisy data. Therefore, it introduces feature randomness to reduce correlation before individual learners.

As detailed by Breiman [Bre01], the algorithm creates a forest of k uncorrelated trees. Each tree is trained on a bootstrap sample of the data, and at each node split, only a random subset of features is considered. For a binary classification task such as diabetes prediction, the final output class Y is determined by a majority vote across all trees, represented as:

$$H(x) = \arg\max_y \sum_{i=1}^{k} I(h_i(x) = y)$$

where $H(x)$ is the ensemble classifier, $h_i(x)$ represents an individual decision tree, and I is the indicator function. This stochastic approach reduces the overall variance of the model without significantly increasing bias, making it particularly suitable for clinical datasets where minor variations in metrics like HbA1c or Glucose levels can drastically shift diagnostic outcomes.

3.4.2 Random Forest in Clinical Diabetes Prediction

In clinical settings, the use of Random Forest has been validated for its robustness against missing values and its ability to handle both categorical and continuous variables simultaneously. Peerbasha et al. [PIS+23] compared several machine learning algorithms on diabetes datasets and concluded that Random Forest consistently outperformed Support Vector Machines (SVM) and Logistic Regression, achieving accuracies as high as 99% when the dataset includes key biomarkers like insulin and glucose concentrations.

Furthermore, the work of Sarwar et al. [SKHS18] emphasized that the *Variable Importance* feature of Random Forest provides critical interpretability for healthcare providers. By ranking features such as HbA1c and Blood Sugar as the most influential predictors, the algorithm aligns with established medical guidelines, thereby bridging the gap between "black-box" machine learning and clinical decision support.

3.4.3 Application to Large-Scale Health Indicators

The emergence of large-scale synthetic datasets, such as the *Diabetes Health Indicators Dataset* [Tha24], has allowed researchers to simulate population-level screening. This dataset, comprising 100,000 records, includes a diverse set of 35+ features spanning clinical measurements and lifestyle habits. Early benchmarks on this specific dataset have shown that tree-based ensemble methods like Random Forest are highly effective at handling the non-linear interactions between age, smoking status, and clinical markers like BMI.

Comparative studies using similar high-volume health data, such as the investigation by Abousaber et al. [AAEG25], indicate that Random Forest models optimized with hyperparameter tuning (e.g., Grid Search) achieve an Area Under the Curve (AUC) between 0.81 and 0.86. These results demonstrate that even with the inclusion of potentially noisy lifestyle indicators, the ensemble nature of Random Forest maintains high sensitivity and specificity, which are vital for early-stage diabetes detection.

3.4.4 Conclusion

These studies collectively justify the selection of Random Forest as a primary classifier for the current task. By transitioning from traditional survey-based indicators to clinical-grade health metrics, the Random Forest algorithm leverages its ensemble structure to provide a robust, interpretable, and high-performance model for predicting diabetes risk in a large, diverse population.

3.5 Experimental Evaluation

This section details the conducted experiments for the Supervised Classification task using the Random Forest model, providing architectural insight in the from-scratch implementation of a Random Forest classifier, alongside the adopted experimental protocol and the results obtained, compared with the built-in **sklearn** implementation.

3.5.1 Experimental Setup

We implemented a Random Forest–style ensemble classifier composed of decision tree base estimators trained on bootstrap samples. Randomization is introduced through instance-level bootstrapping and per-tree feature subsampling, where each tree is trained on a fixed random subset of input features. During tree construction, all features in the selected subset are considered at each split, and nodes are split to maximize impurity reduction using either the *Gini impurity* or *Entropy* criterion.

Decision tree as base estimator. Each tree is trained with a set of stopping conditions, including a maximum depth, a minimum number of samples required to split an internal node, and a minimum number of samples required to be at a leaf node. During tree construction, a candidate split is evaluated by computing the **impurity decrease**:

$$\Delta I = I(\text{parent}) - \frac{n_L}{n} I(\text{left}) - \frac{n_R}{n} I(\text{right}),$$

where $I(\cdot)$ denotes a node impurity measure, n is the number of samples in the parent node, and n_L, n_R are the sample counts of the left and right child nodes, respectively. A split is selected greedily by maximizing ΔI.

In our implementation, node impurity $I(\cdot)$ can be computed using either **Gini impurity** or **Entropy**. Given class proportions p_k for K classes at a node, these metrics are defined as:

$$I_{\text{gini}} = 1 - \sum_{k=1}^{K} p_k^2, \qquad I_{\text{entropy}} = -\sum_{k=1}^{K} p_k \log_2 p_k.$$

Gini impurity measures the probability of misclassification when randomly labeling a sample according to the node's class distribution, while entropy quantifies the uncertainty of the class labels. Both metrics favor splits that produce purer child nodes, though entropy typically penalizes class mixing more strongly.

Ensemble aggregation and prediction. For the final prediction, the forest aggregates outputs from individual trees using a voting strategy:

- **Hard voting:** the predicted class is the majority vote among all trees.
- **Soft voting:** the predicted class is obtained by averaging per-tree class probabilities,

$$\hat{p}(y = 1 \mid x) = \frac{1}{T} \sum_{t=1}^{T} p_t(y = 1 \mid x),$$

  followed by a decision threshold τ such that $\hat{y} = 1$ if $\hat{p}(y = 1 \mid x) \geq \tau$.

In the from-scratch Random Forest implementation, both voting strategies are supported. The decision threshold $\tau \in (0, 1)$ is configurable and allows explicit control over the precision–recall tradeoff, which is particularly relevant in medical diagnosis tasks.

Scratch implementation parameters. Table 9 lists the main configurable parameters of the from-scratch Random Forest implementation.

Parameter	**Description**	**Domain**
n_trees	Number of decision trees in the forest	$\mathbb{N}^*$
max_depth	Maximum depth of each decision tree	$\mathbb{N}^* \cup \{\texttt{None}\}$
min_samples_split	Minimum samples required to split an internal node	$\mathbb{N}^*$
min_samples_leaf	Minimum samples required to create a leaf node	$\mathbb{N}^*$
impurity_metric	Impurity function used at each split	{`gini`, `entropy`}
vote	Ensemble aggregation strategy	{`hard`, `soft`}
threshold	Positive-class decision threshold for soft voting	$(0, 1)$

Table 9: Random Forest Scratch Hyperparameters

3.5.2 Experimental Results

We evaluate the Random Forest models in a Supervised Classification context, using a 80%/20% train-test split. Hyperparameter tuning is performed exclusively on the training split via stratified k-fold cross-validation. The best configuration, selected by a primary scoring metric, is refit on the full training split and evaluated once on the held-out test split.

Feature set and preprocessing. Based on the statistical methods implemented during the exploratory analysis of the employed dataset (Section 1.2), an automated statistical feature selection mechanism was integrated, in order to filter the most relevant features for predicting the target column of diagnosed diabetes diagnosis. The process is applied to the training split only of the data, to ensure no data distribution leakage occurs.

Since Random Forest is an Ensemble Learning method composed of decision trees weak learners, which are especially good using their splitting mechanisms to select the relevant features and detect the ones that have the same contribution to predicting the target feature, this feature selection optimization step is not as important as it would be in a Linear Regression algorithm, therefore the strategy employed for selecting the features uses *Union* voting and takes into consideration all relevant columns identified by each statistical method employed, selecting both glucose-related characteristics, and lifestyle-oriented ones (Table 10, leaving it up to the weak learners to determine their relevance further.

Category	**Selected Features**
Glucose-related	`glucose_fasting`, `glucose_postprandial`, `hba1c`
Lipid profile	`cholesterol_total`, `ldl_cholesterol`, `triglycerides`
Demographic / Lifestyle	`age`, `family_history_diabetes`, `physical_activity_minutes_per_week`

Table 10: Feature set selected via statistical union-based feature selection

All models were trained and evaluated further on the same feature set to ensure a fair comparison.

Cross-validation protocol and refit metric. We used **Stratified K-fold Cross-Validation** with $k = 5$ folds and shuffling enabled. Since the target distribution can be imbalanced in diagnosis prediction, we selected **F1-score** as the primary refit metric, due to its ability to balance precision and recall.

Hyperparameter optimization grid. We performed exhaustive **Grid Search Cross-Validation** across a series of parameter values considered as general solid choices for configuring a Random Forest model. Table 11 describes the explored hyperparameter space.

Parameter	**Value Space**
n_trees	[50, 100]
max_depth	[10, 15]
min_samples_split	[5, 10]
min_samples_leaf	[2, 5]
criterion / impurity_metric	{`gini`, `entropy`}

Table 11: Random Forest Cross-Validation Parameter Grid

Best estimator and performance results. Table 12 reports the held-out test set performance of the best Random Forest estimators obtained via grid search cross-validation. The **sklearn** implementation serves as an optimized baseline, while the from-scratch implementation validates the correctness and robustness of the custom training pipeline. All metrics are computed on the same held-out test split.

Metric	Random Forest (Scratch)	Random Forest (sklearn)
Accuracy	0.909	0.920
Precision	0.985	0.999
Recall	0.850	0.866
F1-score	0.917	0.928
ROC-AUC	0.938	0.943
Average Precision (PR-AUC)	0.969	0.971
Log Loss	0.358	0.215

Table 12: Random Forest Classification Performance on Held-Out Test Set

All reported metrics are computed on the held-out test split using standard binary classification definitions. Accuracy measures the proportion of correctly classified samples. Precision and recall quantify false positive and false negative behavior, respectively, while the F1-score represents their harmonic mean and is used as the primary refit metric during cross-validation. The ROC-AUC and Average Precision scores evaluate ranking quality based on predicted probabilities, while the Log Loss penalizes miscalibrated probabilistic predictions.

Statistical analysis of cross-validation fold scores. Following hyperparameter optimization, the best-performing Random Forest configuration identified by grid search was refit on the full training split. The selected model used $n\_estimators = 100$, a maximum tree depth of 15, a minimum split size of 5, a minimum leaf size of 2, and the `gini` impurity criterion.

In addition to the held-out test metrics, we performed a statistical analysis of the cross-validation fold scores obtained during hyperparameter optimization (Table 13). Descriptive statistics and a 95% confidence interval were computed for the F1-score across folds. To justify the use of a Student's t-distribution for confidence interval estimation, a Shapiro-Wilk normality test was applied to the fold scores.

Statistic	Value
Refit Metric	F1-score
Fold Scores	(0.9319, 0.9274, 0.9306, 0.9289, 0.9310)
Mean	0.92997
Standard Deviation	0.00181
Standard Error of the Mean	0.00081
Minimum / Maximum	0.92738 / 0.93192
95% Confidence Interval	[0.92772, 0.93222]
Shapiro–Wilk Test (p-value)	0.7169

Table 13: Random Forest Statistical Analysis of Cross-Validation F1 Scores

The Shapiro-Wilk normality test returned a p-value of 0.7169, which is well above the conventional significance level of $\alpha = 0.05$. This indicates that the cross-validation fold scores do not significantly deviate from a normal distribution. Consequently, the use of a Student's t-distribution to compute the 95% confidence interval for the mean F1-score is statistically justified.

3.5.3 Discussion

Relevance of Random Forest for diabetes diagnosis. Random Forest is a strong candidate for diabetes diagnosis due to the following properties:

- **Non-linear modeling capability:** It naturally captures threshold effects and complex interactions between heterogeneous predictors such as glucose measurements, age, and lifestyle indicators, without assuming linear or monotonic relationships.
- **Robustness to correlated features:** Physiologically related variables (e.g., fasting glucose, postprandial glucose, HbA1c) do not degrade performance, as feature subsampling and ensemble averaging reduce sensitivity to redundancy.
- **Interpretability support:** While the ensemble itself is complex, Random Forest integrates well with post-hoc explainability techniques (e.g., SHAP), which is critical for clinical decision-support systems.

Interpretation of achieved performance. Both implementations achieved strong and stable classification performance on the held-out test split.

- **Predictive balance:** The F1-scores of 0.917 (scratch) and 0.928 (**sklearn**) indicate an effective balance between precision and recall, appropriate for medical screening tasks.
- **Precision-dominant behavior:** Precision values above 0.98 for both models indicate very few false positives, reducing unnecessary clinical follow-ups.
- **Recall behavior:** Recall values of 0.850 (scratch) and 0.866 (**sklearn**) show that most diabetic cases are detected, with the remaining errors attributable to the default decision threshold.
- **Ranking quality:** High ROC-AUC (> 0.93) and Average Precision (> 0.96) confirm strong discriminative ability across thresholds.

Cross-validation results further support model reliability. The F1-score standard deviation across folds is 0.00181, and the narrow 95% confidence interval [0.92772, 0.93222] indicates low sensitivity to data partitioning. The test F1-score closely matches the cross-validation mean, suggesting good generalization and no overfitting.

Scratch vs. sklearn Random Forest. A direct comparison between implementations reveals the following:

- **Performance:** The **sklearn** model achieves slightly higher performance (approximately +1.1% F1-score), which is expected given its optimized split search and mature implementation.
- **Calibration:** Lower Log Loss for the **sklearn** model (0.215 vs. 0.358) indicates better probability calibration, impacting threshold-independent metrics.
- **Transparency and control:** The from-scratch implementation offers full visibility into tree construction, voting strategies, and prediction logic, which is valuable for educational purposes and algorithmic inspection.

Overall, the from-scratch Random Forest achieves competitive performance while validating the correctness of the implemented learning pipeline. The modest gap to the library baseline highlights the impact of low-level optimizations rather than conceptual or algorithmic shortcomings.

Model interpretability. To analyze the factors driving the Random Forest predictions, we employed **SHapley Additive exPlanations (SHAP)**, which decompose each model output into additive feature-level contributions. This enables both **global interpretability**, by identifying which variables are most influential across the dataset, and **local interpretability**, by explaining individual predictions in a case-specific manner.

SHAP values were computed using the unified `shap.Explainer` interface. Since explanation cost depends on repeated model evaluations, SHAP computation for the from-scratch Random Forest is slower than for optimized library implementations. To maintain tractable runtime while preserving representativeness, we therefore relied on a **subsampled background set** and a **subsampled explanation set**.

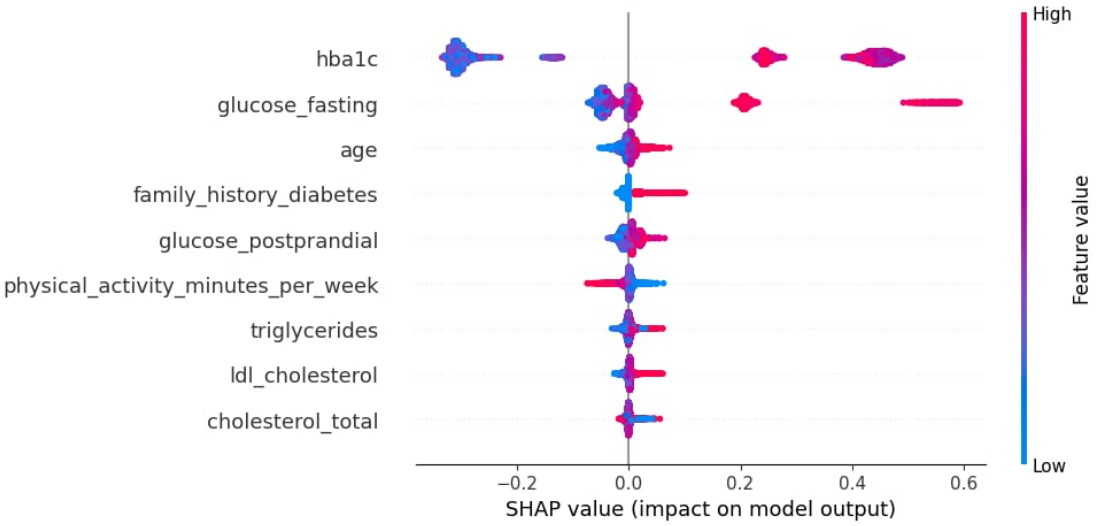

Figure 14: SHAP beeswarm plot for the Random Forest model. Each point represents one sample; the x-axis shows the SHAP value (impact on the model output), while color encodes the feature value (low to high). Features are ordered by global importance.

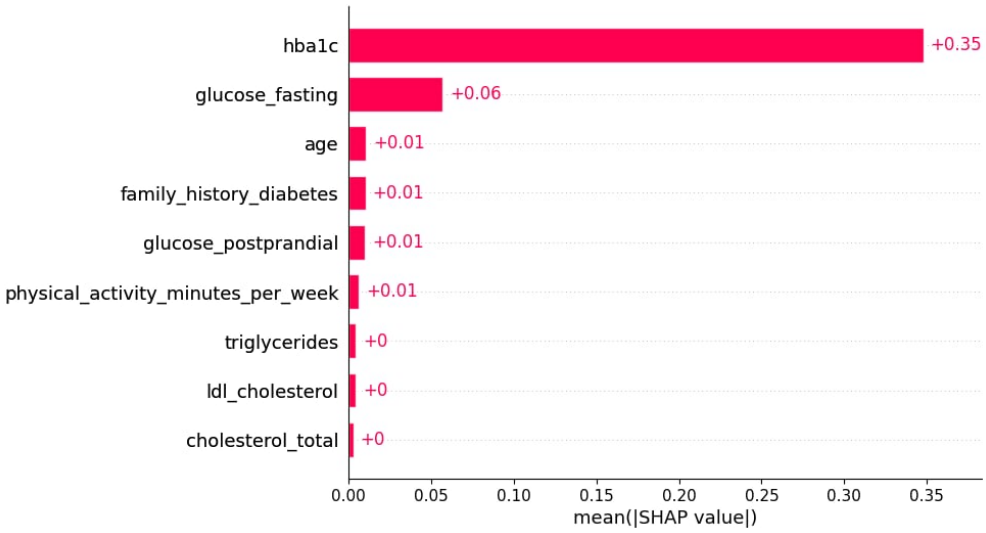

Figure 15: Global SHAP feature importance (mean absolute SHAP value). Higher bars indicate features with stronger average influence on predictions, irrespective of direction.

Figures 14 and 15 provide complementary global explanations. Both plots consistently indicate that `hba1c` is the dominant predictor, exhibiting the largest magnitude and variability

of SHAP values. High `hba1c` values strongly increase the predicted probability of diabetes, while low values exert a pronounced protective effect. `glucose_fasting` represents the second most influential feature, contributing additional risk stratification. In contrast, demographic, lifestyle, and lipid-related variables (e.g., `age`, `physical_activity_minutes_per_week`, and cholesterol measures) exhibit comparatively smaller contributions, acting primarily as secondary refinements rather than primary decision drivers.

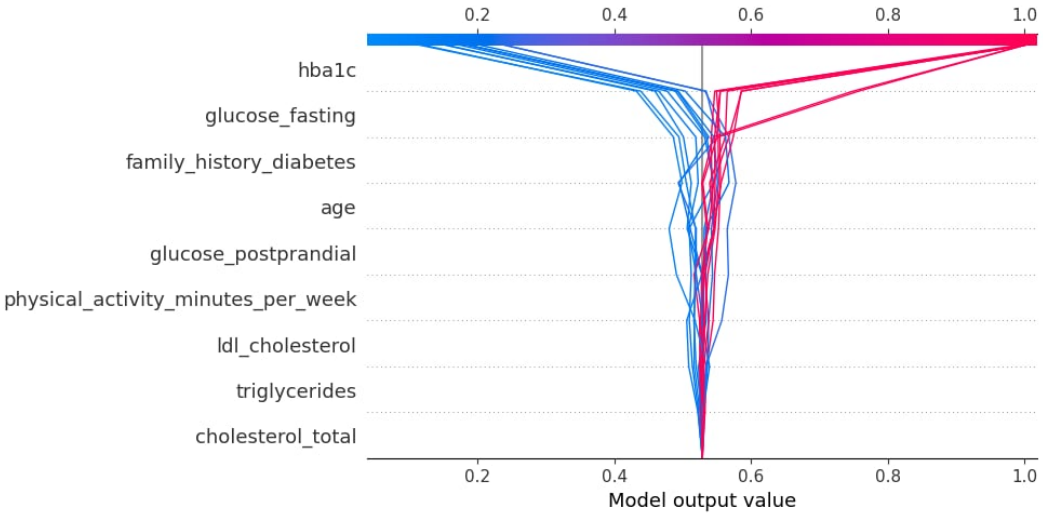

Figure 16: Local SHAP waterfall plot for a representative test instance. Starting from the expected model output, features push the prediction toward the positive (diabetes) or negative class by additive contributions.

Local explanations further illustrate how predictions are formed for individual samples. The waterfall plot in Figure 16 shows how the model output deviates from the baseline expectation through successive feature contributions. In the illustrated example, low `hba1c` and `glucose_fasting` values produce large negative SHAP contributions, outweighing minor positive effects from other variables and resulting in a low predicted diabetes risk. Such explanations are particularly valuable in clinical decision-support settings, as they explicitly link predictions to clinically interpretable factors.

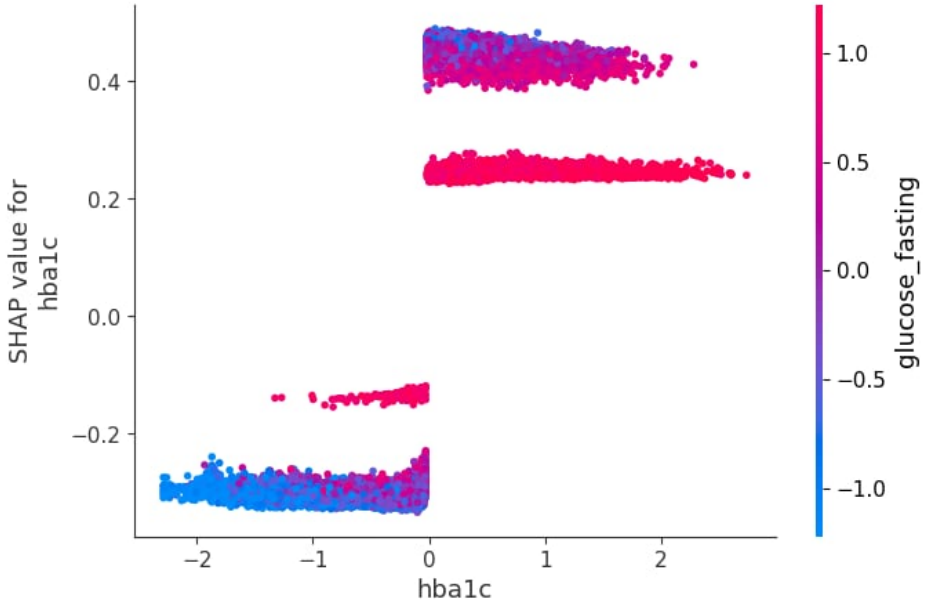

Figure 17: SHAP dependence plot for `hba1c`. The x-axis shows the feature value, while the y-axis shows its SHAP contribution. Color indicates a potentially interacting feature (e.g., `glucose_postprandial`), highlighting interaction patterns captured by the ensemble.

Finally, the dependence plot in Figure 17 highlights non-linear feature effects captured by the ensemble. The contribution of `hba1c` exhibits threshold-like behavior, with sharp increases in SHAP value beyond specific ranges. Coloring by glucose-related features further reveals interaction effects, indicating that the influence of `hba1c` is modulated by concurrent glucose measurements. This behavior aligns with known physiological relationships and demonstrates the ability of the Random Forest to model non-linear risk patterns beyond the capacity of linear classifiers.

Comparison to related work. Table 14 compares our Random Forest classification results with prior studies on diabetes diagnosis. Reported performance in the literature often exceeds 94% accuracy, particularly when evaluated on smaller benchmark datasets such as the Pima Indians Diabetes Dataset [SKHS18, PIS+23], or when advanced imbalance-handling techniques such as SMOTE-Tomek are employed [AAEG25].

These results are not directly comparable, as the cited studies use different datasets, feature sets, and class distributions. In contrast, our experiments are conducted on the *Diabetes Health Indicators* dataset comprising 100,000 samples [Tha24], which reflects a more heterogeneous and realistic clinical population. Within this setting, the obtained performance remains competitive without relying on synthetic resampling, indicating strong generalization and robustness of the proposed Random Forest model.

Study	Accuracy	Precision	Recall	F1-score	Notes
Peerbasha et al. [PIS+23]	$\sim 94\%$	0.94	0.94	0.94	Best performer among SVM and KNN classifiers.
Sarwar et al. [SKHS18]	94.1%	0.93	0.92	0.92	Evaluated on Pima Indians Dataset; RF outperformed Logistic Regression.
Abousaber et al. [AAEG25]	$\sim 98.2\%$	0.98	0.98	0.98	AUC = 0.99; results boosted by SMOTE–Tomek imbalance handling.
scratch RF	90.9%	0.985	0.850	0.917	Diabetes Health Indicators dataset (100k samples); no synthetic resampling.
sklearn RF	92.0%	0.999	0.866	0.928	Optimized library baseline on the same dataset and feature set.

Table 14: Comparison of Random Forest-based diabetes diagnosis performance across studies

Scikit-learn Implementation Details. To validate the performance of the proposed Random Forest classifier, we benchmarked it against the standard `RandomForestClassifier` implementation. This estimator is a mature and highly optimized library baseline that relies on efficient C-optimized routines for decision tree construction, impurity-based split search, and ensemble aggregation, making it a widely adopted reference in applied machine learning.

The **sklearn** implementation constructs each decision tree using greedy impurity minimization (Gini or entropy) and employs bootstrap sampling and feature subsampling to decorrelate trees within the ensemble. These design choices improve generalization and reduce variance compared to single-tree classifiers, particularly in high-dimensional or noisy settings.

We employed the *same experimental protocol* as for the from-scratch Random Forest to ensure a fair comparison. Hyperparameter tuning was performed using 5-fold stratified cross-validation on the training split only, with **F1-score** used as the refit metric. The explored parameter grid matches the one described in Table 11, and the selected best estimator was refit on the full training split before evaluation.

The best **sklearn** configuration was obtained with $n\_estimators = 100$, `max_depth` $= 15$, `min_samples_split` $= 5$, `min_samples_leaf` $= 2$, and the `gini` impurity criterion. Final performance metrics are reported on the held-out test set and directly compared to the from-scratch implementation in Table 12.

3.5.4 Conclusion and Future Directions

In conclusion, the Random Forest classifier provides robust predictive performance for diabetes diagnosis, achieving high discriminative ability while remaining interpretable through feature attribution methods.

Future work directions include:

1. Improving the efficiency of the from-scratch implementation (vectorized inference, optimized split search, parallelism).

2. Better threshold calibration to control the clinical precision–recall tradeoff (e.g., maximizing recall under a minimum precision constraint).

3. Extending evaluation with calibration metrics and more detailed error analysis (confusion matrix stratified by risk factors).

References

[AAEG25] Inam Abousaber, Haitham F Abdallah, and Hany El-Ghaish. Robust predictive framework for diabetes classification using optimized machine learning on imbalanced datasets . *Frontiers in Artificial Intelligence* , 7:1499530, 2025.

[Bre96] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[Bre01] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[CL11] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):1–27, 2011.

[DBK+96] Harris Drucker, Christopher J Burges, Linda Kaufman, Alex Smola, and Vladimir Vapnik. Support vector regression machines. *Advances in neural information processing systems*, 9, 1996.

[GPA+13] Eleni I Georga, Vasilios C Protopappas, Diego Ardigo, Demosthenes Polyzos, and Dimitrios I Fotiadis. A glucose model based on support vector regression for the prediction of hypoglycemic events under free-living conditions. *Diabetes technology & therapeutics*, 15(8):634–643, 2013.

[HADC+18] Takoua Hamdi, Jaouher Ben Ali, Véronique Di Costanzo, Farhat Fnaiech, Eric Moreau, and Jean-Marc Ginoux. Accurate prediction of continuous blood glucose based on support vector regression and differential evolution algorithm. *Biocybernetics and Biomedical Engineering*, 38(2):362–372, 2018.

[PIS+23] S Peerbasha, Y Mohammed Iqbal, M Mohamed Surputheen, A Saleem Raja, et al. Diabetes prediction using decision tree, random forest, support vector machine, k-nearest neighbors, logistic regression classifiers . *Journal of Advanced Applied Scientific Research* , 5(4):42–54, 2023.

[SKHS18] Muhammad Azeem Sarwar, Nasir Kamal, Wajeeha Hamid, and Munam Ali Shah. Prediction of diabetes using machine learning algorithms in healthcare . In *2018 24th international conference on automation and computing (ICAC)* , pages 1–6. IEEE, 2018.

[Tha24] Mohan Krishna Thalla. Diabetes Health Indicators Dataset: A Comprehensive Dataset of 100,000 Patient Records . `https://www.kaggle.com/datasets/mohankrishnathalla/diabetes-health-indicators-dataset`, 2024.

[VHBCP20] Amelec Viloria, Yaneth Herazo-Beltran, Danelys Cabrera, and Omar Bonerge Pineda. Diabetes diagnostic prediction using vector support machines. *Procedia Computer Science*, 170:376–381, 2020.

[ZO20] Fan Zhang and Lauren J O'Donnell. Support vector regression. In *Machine learning*, pages 123–140. Elsevier, 2020.