

VIETNAM NATIONAL UNIVERSITY - HO CHI MINH CITY
UNIVERSITY OF INFORMATION TECHNOLOGY
FACULTY OF INFORMATION SYSTEMS



PROJECT REPORT
DATA MINING

TOPIC
DIABETES PREDICTION

Lecturer: PhD. Cao Thi Nhan

Instructor: MSc. Vu Minh Sang

Class: IS252.O22.HTCL

Member: Team 5

21520596 – Tran Thi Kim Anh

21521049 – Ho Quang Lam

21521586 – Le Thi Le Truc

21521882 – Le Minh Chanh

Ho Chi Minh City, June 2024

TABLE OF CONTENTS

LIST OF FIGURES	4
LECTURER'S COMMENT	5
WORK DISTRIBUTION	6
Chapter 1. INTRODUCTION	7
1.1. Problem	7
1.2. List of algorithms	7
1.3. Intention	8
1.4. Developer tools & technology	8
Chapter 2. ALGORITHMS	9
2.1. K – Nearest Neighbor (KNN)	9
2.2. Decision Tree	10
2.3. Random Forest	12
2.4. Support Vector Machine (SVM).....	14
2.5. Extreme Gradient Boost (XGBOOST).....	16
2.6. Evaluate your monitor's performance	18
2.6.1. Accuracy.....	18
2.6.2. F1 – Score	19
2.6.2.1. Precision	19
2.6.2.2. Recall.....	19
2.6.2.3. F1 – Score.....	19
2.7. Pipeline	20
Chapter 3. DATA PREPROCESSING	21
3.1. Description of original data.....	21

3.2.	Data preprocessing	26
3.2.1.	Drop column HvyAlcoholConsump and CholCheck	26
3.2.2.	Check correlation of other columns with Diabetes column	27
3.2.3.	Scaling data for features selection using MinMaxScaler method. 29	
3.2.4.	Feature importance	30
3.2.5.	Diabetes distribution	30
Chapter 4. PREDICTIVE SOFTWARE		32
4.1.	Technology	32
4.2.	Key features	32
4.3.	About the web application – Diabetes Prediction	33
4.4.	Results	34
Chapter 5. CONCLUSION		37
5.1.	The advantage.....	37
5.2.	The disadvantage.....	37
5.3.	The development	37
REFERENCE		39

LIST OF FIGURES

<i>Table 1: F1-Score and Accuracy Score of Models</i>	34
<i>Figure 2.1: Structure of the decision tree</i>	11
<i>Figure 2.2: Random Forest Operations</i>	13
<i>Figure 2.3: Architecture and working of SVM</i>	15
<i>Figure 2.4: Architecture and working of SVM</i>	17
<i>Figure 2.5: Project pipeline</i>	20
<i>Figure 3.1: Amount of data in the dataset</i>	22
<i>Figure 3.2: Number of unique values in each column</i>	23
<i>Figure 3.3: Descriptive statistics of properties in the dataset</i>	24
<i>Figure 3.4: Bar charts of health variables</i>	25
<i>Figure 3.5: Correlation of other columns with Diabetes column</i>	27
<i>Figure 3.6: Correlation matrix between columns in the dataset</i>	28
<i>Figure 3.7: The remaining columns can be used</i>	28
<i>Figure 3.8: Data of columns</i>	29
<i>Figure 3.9: Data after using MinMaxScaler</i>	29
<i>Figure 3.10: Check the importance of features</i>	30
<i>Figure 3.11: Diabetes Distribution</i>	30
<i>Figure 3.12: Diabetes Distribution in Test Data</i>	31
<i>Figure 3.13: Diabetes Distribution in Train Data</i>	31
<i>Figure 4.1: Steamlit's logo</i>	32
<i>Figure 4.2: User interface - Select parameters</i>	33
<i>Figure 4.3: Pick a model to predict</i>	33
<i>Figure 4.4: Result: Not at high risk of diabetes</i>	34
<i>Figure 4.5: Result: High risk of diabetes</i>	34
<i>Figure 4.6: The chart shows F1-Score</i>	35
<i>Figure 4.7: The chart shows Accuracy Score</i>	35

LECTURER'S COMMENT

This image shows a full page of primary-ruled paper. It features approximately 20 horizontal dotted lines spaced evenly down the page, providing a guide for handwriting practice. The paper is otherwise blank, with no margins, text, or other markings.

WORK DISTRIBUTION

Task	Tran Thi Kim Anh	Ho Quang Lam	Le Thi Le Truc	Le Minh Chanh
Find data and state the problem				✓
Data preprocessing	✓	✓	✓	
SVM Model	✓			
KNN Model			✓	
Random Forest Model		✓		
Decision Tree				✓
XGBoost Model				✓
Presentation of results and evaluation			✓	
Web Demo		✓		✓
Reports	✓	✓	✓	✓
Completion (%)	100%	100%	100%	100%

Chapter 1. INTRODUCTION

1.1. Problem

In the context of heightened awareness and concern for community health, the prediction and analysis of the risk of diabetes have become a crucial area of research in healthcare. This report focuses on data mining for diabetes prediction, utilizing attributes such as age, body mass index, family history, diet, physical activity, and other factors that may be associated with the risk of this condition.

The objective of this report is to develop an effective forecasting model capable of accurately predicting the risk of diabetes for individuals based on medical information and relevant risk factors. Consequently, diabetes prediction can play a vital role in early diagnosis, preventive measures, and efficient management of this disease.

In this report, we will present the process of data mining, analysis, and attribute handling alongside the construction of the prediction model. Additionally, we will explore popular techniques and approaches in diabetes prediction, including machine learning models and data mining techniques.

We hope that this report will provide a comprehensive and insightful perspective on the application of data and prediction techniques to enhance the ability to predict and manage diabetes. The results from this study can significantly contribute to improving early diagnosis, prevention, and effective treatment for diabetes patients, thereby enhancing their quality of life and reducing mortality rates associated with this disease..

1.2. List of algorithms

- K – Nearest Neighbor (KNN)
- Decision Tree
- Random Forest
- Support Vector Machine
- Extreme Gradient Boost Model (XGBoost)

1.3. Intention

The objective of this study is to develop an accurate predictive model for the risk of diabetes. Relevant data such as age, gender, cholesterol, BMI, smoking, history of cardiovascular disease, physical activity, blood pressure, and dietary habits will be collected and cleaned. This process will include handling missing values, outliers, and normalizing the data, as well as conducting exploratory data analysis to gain a better understanding of the features.

Advanced machine learning techniques such as Decision Tree, Random Forest, KNN, SVM, and XGBoost will be applied. The results from these models will help improve early diagnosis, prevention, and more effective management of diabetes. The models will be built and then their performance will be compared using metrics such as accuracy, sensitivity, and specificity. The best-performing model will be selected based on the comparative results.

The goal is to leverage data mining and predictive modeling to enhance the ability to predict and manage diabetes. This can significantly contribute to improving early diagnosis, prevention, and effective treatment for diabetes patients, thereby enhancing their quality of life and reducing mortality rates associated with this disease.

1.4. Developer tools & technology

In the process of implementation, the group used a number of software for researching and developing the topic:

- Information collection and analysis using the python library and programming language
- Data sources: [Diabetes Prediction | Kaggle.com](#)

Chapter 2. ALGORITHMS

2.1. K – Nearest Neighbor (KNN)

K-NN is a versatile and widely used machine learning algorithm, mainly used for its simplicity and ease of implementation. It does not require any assumptions about the distributed database. It can also handle both data numbers and data types, successfully creating a flexible selection tool for a variety of data types in various types of scale classification and recovery tasks. This is a method that gives expected parameters based on the similarity of the data in a given data set. K-NN is less sensitive to outliers than other algorithms.

The K-NN algorithm works by finding the K closest neighbors to a well defined database according to a distance measure.

The KNN algorithm assumes that similar data will exist close to each other in a space, so our job is to find the k points closest to the data to be tested. There are many formulas that can be used to find the distance between two points. Depending on the situation, we can choose accordingly. Here are 3 basic ways to calculate the distance of 2 data points x and y with k attributes:

- Euclidean distance:

This is nothing more than the Cartesian distance between two points located in the 2D/super-construction. Euclidean distance can also be shown as the length of a straight line connecting the two points under test. This metric helps us perform transition calculations between two states of an object.

$$\text{distance}(x, X_i) = \sqrt{\sum_{j=1}^d (x_j - X_{i_j})^2}$$

- Manhattan distance:

The Manhattan Distance metric is often used when we are interested in the total distance an object has traveled rather than in transitions. This metric is calculated by summing the absolute difference between the heights of the points in the integer dimension.

$$d(x, y) = \sum_{i=1}^n |x_i - y_i|$$

- Minkowski Distance

We can say that the Euclidean distance as well as the Manhattan distance are special cases of the Minkowski distance.

$$d(x, y) = \left(\sum_{i=1}^n (x_i - y_i)^p \right)^{\frac{1}{p}}$$

From the above formula, we can say that when $p = 2$ it is the same as the formula for calculating the Euclidean distance, and when $p = 1$ then we obtain the formula for calculating the Manhattan distance.

2.2. Decision Tree

Decision trees are a popular and powerful tool used in many different fields such as machine learning, data mining, and statistics. They provide a clear and intuitive way to make data-driven decisions by modeling relationships between different variables.

A decision tree is a diagram-like structure used to make decisions or predictions. It consists of nodes that represent decisions or tests of attributes, branches that represent the results of these decisions, and leaf nodes that represent the final results or predictions. Each internal node corresponds to a test on an attribute, each branch corresponds to the result of the test, and each leaf node corresponds to a class label or a continuous value.

The structure of the decision tree:

- **Root node:** Represents the entire data set and the initial decision made.
- **Internal node:** Represents decisions or tests about properties. Each internal node has one or more branches.

- **Branch:** Represents the result of a decision or experiment, leading to another node.
- **Leaf Node:** Represents the final decision or prediction. No further division occurs at these nodes.

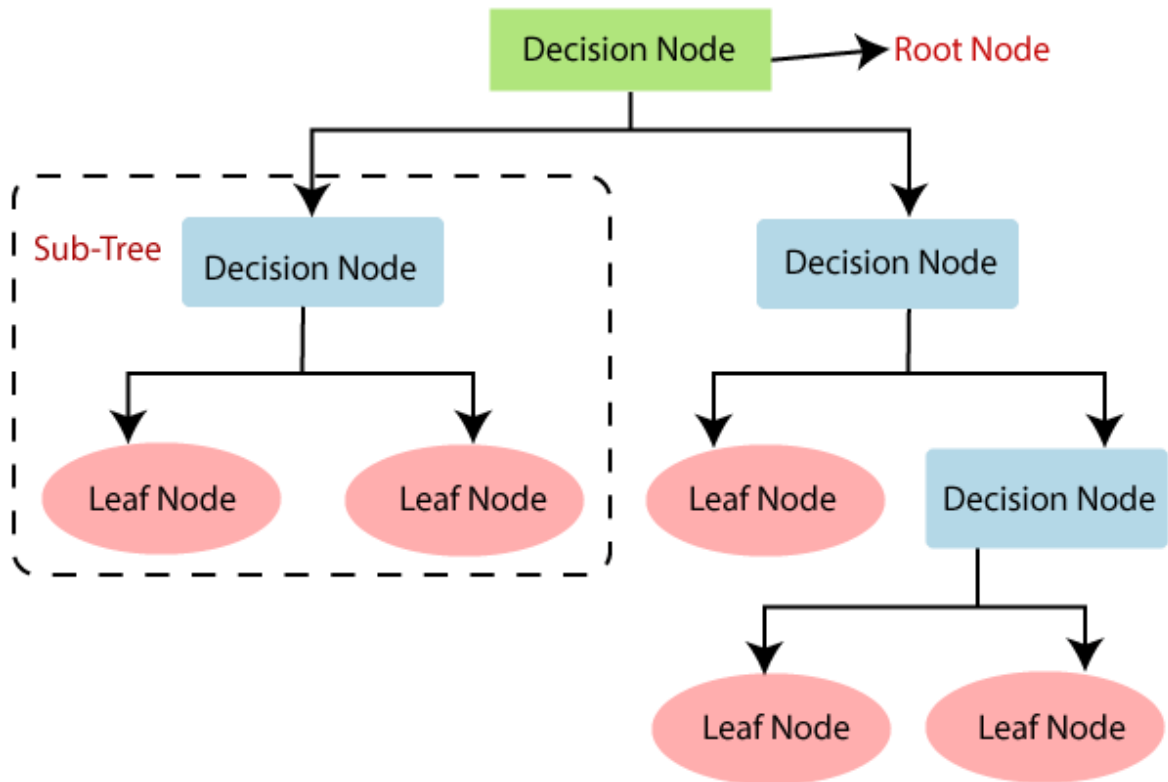


Figure 2.1: Structure of the decision tree

The process of creating a decision tree includes:

- Choose the best attribute: Using metrics such as Gini impurity, entropy or information gain, the best attribute to divide the data will be selected.
- Split dataset: The dataset is divided into subsets based on the selected attribute.
- Repeat the process: The process is repeated recursively for each subset, creating a new internal or leaf node until the stopping criterion is met (e.g. all instances in a node belong to the same a layer or reach a predetermined depth).

Metrics for analysis:

- Gini impurity: Measures the likelihood of a new instance being classified incorrectly if it were randomly classified according to the distribution of classes in the dataset.

$$Gini = 1 - \sum_{i=1}^C (p_i)^2$$

- Entropy: Measures the amount of uncertainty or impurity in a data set

$$E(S) = \sum_{i=1}^c -p_i \log_2 p_i$$

2.3. Random Forest

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

The below diagram explains the working of the Random Forest algorithm:

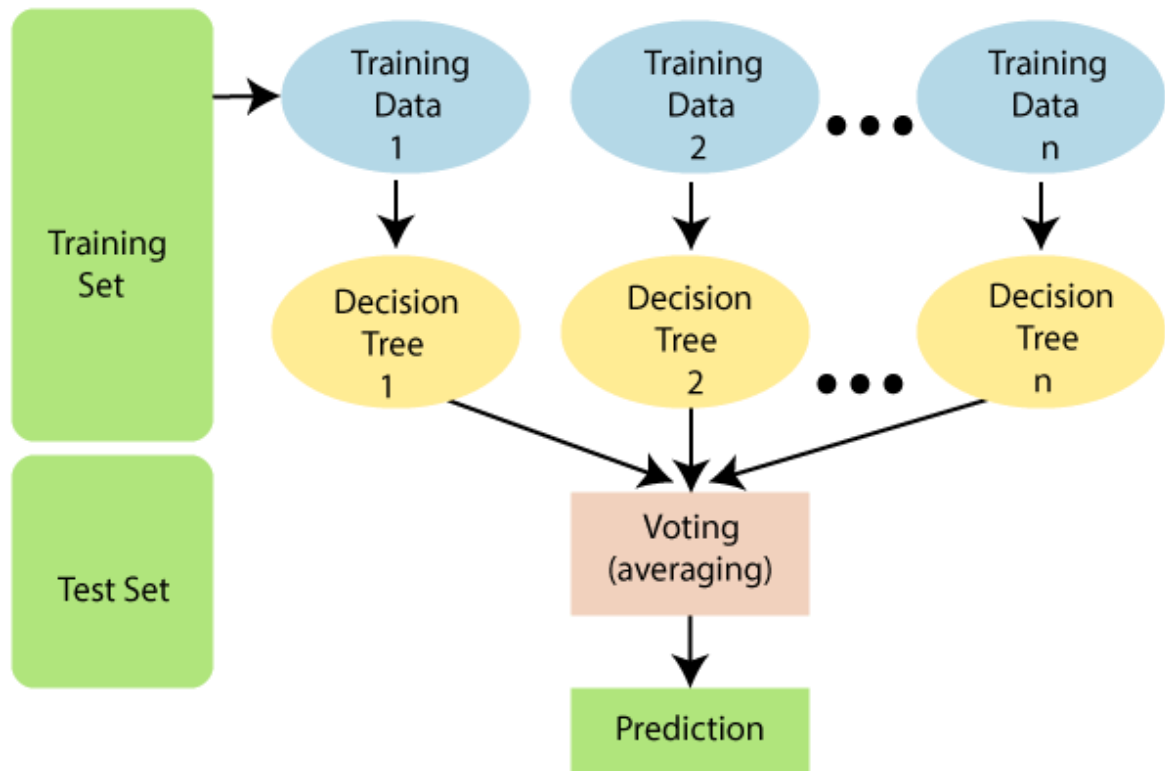


Figure 2.2: Random Forest Operations

Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random Forest classifier:

- There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
- The predictions from each tree must have very low correlations.

Random Forest's activities: Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

- **Step-1:** Select random K data points from the training set.
- **Step-2:** Build the decision trees associated with the selected data points (Subsets).
- **Step-3:** Choose the number N for decision trees that you want to build.
- **Step-4:** Repeat Step 1 & 2.

- **Step-5:** For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

2.4. Support Vector Machine (SVM)

A Support Vector Machine (SVM) is one of the most popular Supervised learning algorithms, which is used for Classification as well as Regression problems. However, it is primarily used for Classification problems in Machine Learning.

The advantages of support vector machines are:

- Effective in high dimensional spaces.
- Remains effective even when the number of dimensions exceeds the number of samples.
- Utilizes a subset of training points in the decision function (known as support vectors), making it memory efficient.
- Versatile: different kernel functions can be specified for the decision function.

While common kernels are provided, custom kernels can also be specified.

The disadvantages of support vector machines include:

- When the number of features significantly exceeds the number of samples, it is crucial to avoid over-fitting by carefully choosing the kernel functions and regularization term.
- SVMs do not directly provide probability estimates; these are calculated using an expensive five-fold cross-validation (see Scores and probabilities below).

The goal of the SVM algorithm is to create the optimal line or decision boundary that can divide an n-dimensional space into classes, allowing us to easily categorize new data points correctly in the future. This optimal decision boundary is referred to as a hyperplane. The hyperplane aims to maximize the margin between the closest points of different classes. The dimension of the hyperplane depends on the number of features. If there are two input features, the hyperplane is a line. If there are three input features, the hyperplane becomes a 2-D plane.

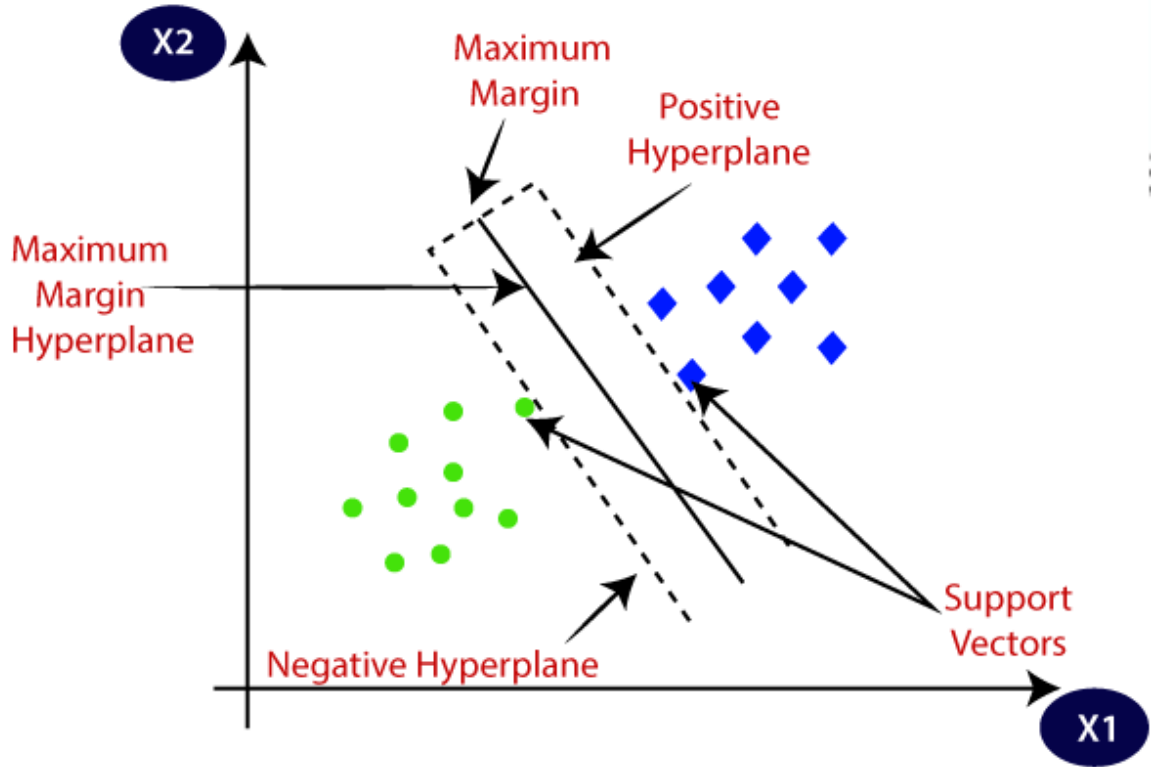


Figure 2.3: Architecture and working of SVM

In a 2-dimensional space, we know that the distance from a point with coordinates (x_0, y_0) to the line with the equation $(w_1x + w_2y + b = 0)$ is given by:

$$\frac{|w_1x_0 + w_2y_0 + b|}{\sqrt{w_1^2 + w_2^2}}$$

In a three-dimensional space, the distance from a point with coordinates (x_0, y_0, z_0) to a plane with the equation $w_1x + w_2y + w_3z + b = 0$ is given by:

$$\frac{|w_1x_0 + w_2y_0 + w_3z_0 + b|}{\sqrt{w_1^2 + w_2^2 + w_3^2}}$$

Moreover, if we remove the absolute value in the numerator, we can determine whether the point lies on which side of the line or plane being considered. Points for which the expression inside the absolute value is positive lie on one side (which I'll temporarily refer to as the positive side) of the line or plane, while points for which the expression inside the absolute value is negative lie on the other side (which I'll call the negative side). Points lying exactly on the line/plane make the numerator zero, hence the distance is zero.

This can be generalized to higher-dimensional spaces: The distance from a point (vector) with coordinates \mathbf{x}_0 to a hyperplane with the equation $\mathbf{w}^T \mathbf{x} + b = 0$ is determined by:

$$\frac{|\mathbf{w}^T \mathbf{x}_0 + b|}{\|\mathbf{w}\|_2}$$

Here, $\|\mathbf{w}\|_2 = \sqrt{\sum_{i=1}^d w_i^2}$ is the number of dimensions in the space.

2.5. Extreme Gradient Boost (XGBOOST)

Extreme Gradient Boosting (XGBoost) is an ensemble learning algorithm widely used for supervised learning tasks like regression and classification. XGBoost constructs a predictive model by sequentially combining the predictions of multiple base models, typically decision trees.

The optimization method (gradient) reduces a cost function by continuously adjusting the model's parameters based on the gradients of the errors. The algorithm introduces “gradient boosting with decision trees”, where the objective function is minimized by determining the importance of each decision tree added to the ensemble sequentially. XGBoost enhances accuracy and efficiency by incorporating a regularization term and employing a more advanced optimization algorithm.

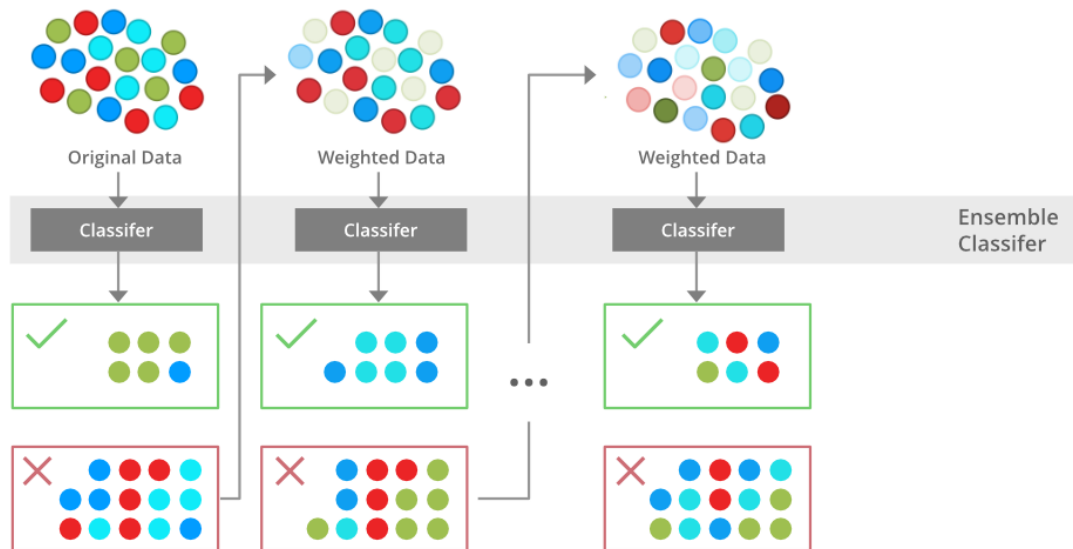


Figure 2.4: Architecture and working of SVM

XGBoost is a popular implementation of gradient boosting. Here are some unique features of the XGBoost model that make it particularly interesting:

Regularization: XGBoost includes options for both L1 and L2 regularization, allowing it to penalize complex models. This helps prevent overfitting.

Handling Sparse Data: XGBoost incorporates a sparsity-aware split finding algorithm to manage different types of sparsity patterns in the data, such as missing values or one-hot encoding that make data sparse.

Weighted Quantile Sketch: While most existing tree-based algorithms can find split points with data points of equal weights using the quantile sketch algorithm, they struggle with weighted data. XGBoost has a distributed weighted quantile sketch algorithm to effectively handle weighted data.

Block Structure for Parallel Learning: For faster computation, XGBoost can utilize multiple CPU cores. Its block structure design allows data to be sorted and stored in in-memory units called blocks, enabling reuse of data layout in subsequent iterations instead of recomputing it. This feature is particularly useful for split finding and column sub-sampling.

Cache Awareness: XGBoost is designed to optimize hardware usage. It minimizes non-continuous memory access required to obtain gradient statistics by

row index by allocating internal buffers in each thread to store these statistics. This design improves efficiency and performance, with parallel tree construction enhanced by languages like Julia and Java.

Out-of-Core Computing: This feature maximizes the usage of available disk space, optimizing handling of large datasets that do not fit into memory.

2.6. Evaluate your monitor's performance

When building a Machine Learning model, we need an evaluation metric to see how effective the model is and to compare the capabilities of different models.

The performance of a model is typically evaluated based on a test dataset. Specifically, let's assume the output of the model when the input is the test dataset is described by the vector y_{pred} - this is the vector of predicted classes for each data point in the test set. We need to compare this predicted vector y_{pred} with the true class vector of the data, denoted by y_{true} .

There are many ways to evaluate a classification model. Depending on the specific problem, we use different evaluation methods. Some commonly used metrics are: accuracy score, confusion matrix, ROC curve, Area Under the Curve, Precision and Recall, F1 score, Top R error, etc.

In your project, you mentioned that your team used the Accuracy score and F1 score to evaluate the algorithms.

2.6.1. Accuracy

The accuracy score is the simplest metric for evaluating classification problems, calculated by dividing the number of correct predictions by the total number of predictions.

The limitation of this evaluation method is that it only tells us the percentage of data that was correctly classified, but does not provide specific details on how each class was classified. It does not reveal which classes were classified most accurately or which data was most frequently misclassified as other classes.

$$accuracy = \frac{true\ positives + true\ negatives}{true\ positives + true\ negatives + false\ negatives + false\ positives}$$

2.6.2. F1 – Score

2.6.2.1. Precision

Precision is a metric that gives you the proportion of true positives to the amount of total positives that the model predicts.

$$precision = \frac{true\ positives}{true\ positives + false\ positives}$$

2.6.2.2. Recall

Recall focuses on how good the model is at finding all the positives.

$$recall = \frac{true\ positives}{true\ positives + false\ negatives}$$

2.6.2.3. F1 – Score

F1 Score is a measure that combines recall and precision. As we have seen there is a trade-off between precision and recall, F1 can therefore be used to measure how effectively our models make that trade-off.

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

One important feature of the F1 score is that the result is zero if any of the components (precision or recall) fall to zero. Thereby it penalizes extreme negative values of either component.

2.7. Pipeline

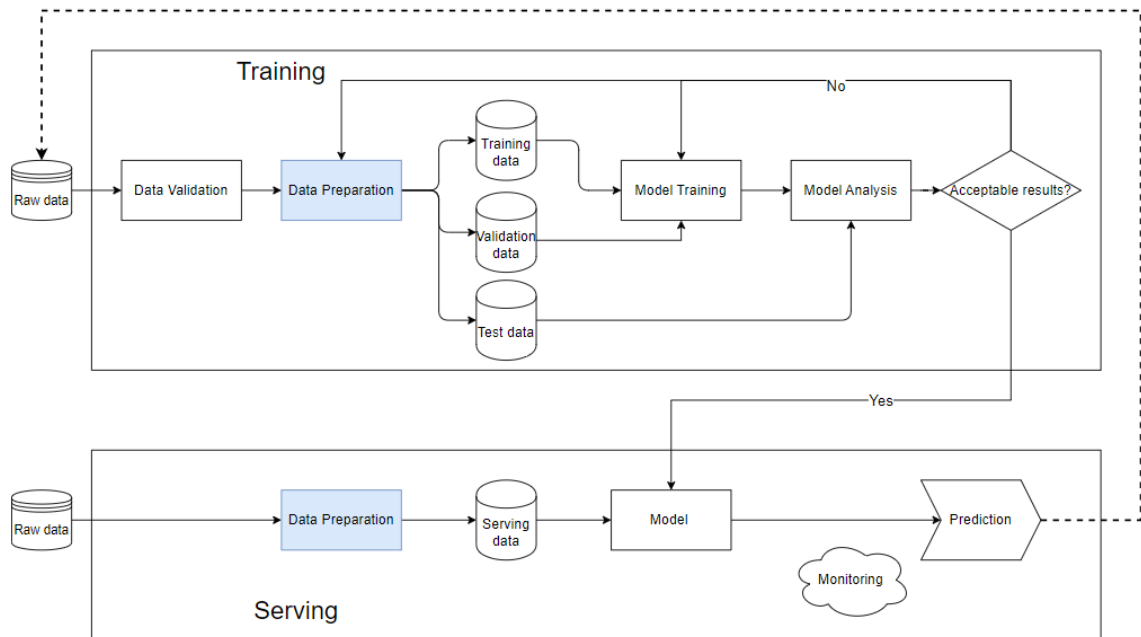


Figure 2.5: Project pipeline

Chapter 3. DATA PREPROCESSING

3.1. Description of original data

Link dataset: [Diabetes Prediction \(kaggle.com\)](https://www.kaggle.com/datasets/uciml/diabetes-prediction)

Dataset name: Diabetes Prediction

The dataset comprises 70,692 survey responses from the Behavioral Risk Factor Surveillance System (BRFSS) in the United States in 2015, which have been cleaned and balanced. The dataset used contains 18 columns and 70,692 rows of data, fully standardized into numerical values.

- **Age:** 13-level age category arranged in 5 years steps: 1 = 18 - 24; 2 = 25 -29; ...; 9 = 60 – 64; ...; 13 = 80 or older.
- **Sex:** patient's gender 1 = male; 0 = female.
- **HighChol:** 0 = no high cholesterol; 1 = high cholesterol.
- **CholCheck:** 0 = no cholesterol check in 5 years; 1 = yes cholesterol check in 5 years.
- **BMI:** Body Mass Index
- **Smoker:** Have you smoked at least 100 cigarettes in your entire life? (Note: 5 packs = 100 cigarettes) 0 = no; 1 = yes.
- **HeartDiseaseorAttack:** coronary heart disease (CHD) or myocardial infarction (MI) 0 = no; 1 = yes.
- **PhysActivity:** physical activity in past 30 days - not including job 0 = no; 1 = yes.
- **Fruits:** Consume Fruit 1 or more times per day 0 = no; 1 = yes.
- **Veggies:** Consume Vegetables 1 or more times per day 0 = no; 1 = yes.
- **HvyAlcoholConsump:** (adult men ≥ 14 drinks per week and adult women ≥ 7 drinks per week) 0 = no; 1 = yes.
- **GenHlth:** Would you say that in general your health is: scale 1-5 1 = excellent; 2 = very good; 3 = good; 4 = fair; 5 = poor.
- **MentHlth:** days of poor mental health scale 1-30 days.
- **PhysHlth:** physical illness or injury days in past 30 days scale 1-30.

- **DiffWalk:** Do you have serious difficulty walking or climbing stairs? 0 = no; 1 = yes.
- **Stroke:** you ever had a stroke 0 = no; 1 = yes.
- **HighBP:** 0 = no high; BP 1 = high BP.
- **Diabetes:** 0 = no diabetes; 1 = diabetes.

The number of values:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70692 entries, 0 to 70691
Data columns (total 18 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Age                                   70692 non-null  int32
1   Sex                                   70692 non-null  int32
2   HighChol                             70692 non-null  int32
3   CholCheck                            70692 non-null  int32
4   BMI                                   70692 non-null  int32
5   Smoker                               70692 non-null  int32
6   HeartDiseaseorAttack                 70692 non-null  int32
7   PhysActivity                         70692 non-null  int32
8   Fruits                               70692 non-null  int32
9   Veggies                              70692 non-null  int32
10  HvyAlcoholConsump                    70692 non-null  int32
11  GenHlth                              70692 non-null  int32
12  MentHlth                             70692 non-null  int32
13  PhysHlth                             70692 non-null  int32
14  Diffwalk                             70692 non-null  int32
15  Stroke                               70692 non-null  int32
16  HighBP                               70692 non-null  int32
17  Diabetes                             70692 non-null  int32
dtypes: int32(18)
memory usage: 4.9 MB
```

Figure 3.1: Amount of data in the dataset

⇒ We have no NaN value but we will have missing values.

The number of unique values by columns:

```
THE NUMBER OF UNIQUE VALUES BY COLUMNS:

Age          13
Sex           2
HighChol     2
CholCheck    2
BMI          80
Smoker        2
HeartDiseaseorAttack  2
PhysActivity  2
Fruits        2
Veggies       2
HvyAlcoholConsump  2
GenHlth       5
MentHlth     31
PhysHlth     31
DiffWalk      2
Stroke        2
HighBP        2
Diabetes       2
dtype: int64
```

Figure 3.2: Number of unique values in each column

	count	mean	std	min	25%	50%	75%	max
Age	70692.0	8.584055	2.852153	1.0	7.0	9.0	11.0	13.0
Sex	70692.0	0.456997	0.498151	0.0	0.0	0.0	1.0	1.0
HighChol	70692.0	0.525703	0.499342	0.0	0.0	1.0	1.0	1.0
CholCheck	70692.0	0.975259	0.155336	0.0	1.0	1.0	1.0	1.0
BMI	70692.0	29.856985	7.113954	12.0	25.0	29.0	33.0	98.0
Smoker	70692.0	0.475273	0.499392	0.0	0.0	0.0	1.0	1.0
HeartDiseaseorAttack	70692.0	0.147810	0.354914	0.0	0.0	0.0	0.0	1.0
PhysActivity	70692.0	0.703036	0.456924	0.0	0.0	1.0	1.0	1.0
Fruits	70692.0	0.611795	0.487345	0.0	0.0	1.0	1.0	1.0
Veggies	70692.0	0.788774	0.408181	0.0	1.0	1.0	1.0	1.0
HvyAlcoholConsump	70692.0	0.042721	0.202228	0.0	0.0	0.0	0.0	1.0
GenHlth	70692.0	2.837082	1.113565	1.0	2.0	3.0	4.0	5.0
MentHlth	70692.0	3.752037	8.155627	0.0	0.0	0.0	2.0	30.0
PhysHlth	70692.0	5.810417	10.062261	0.0	0.0	0.0	6.0	30.0
DiffWalk	70692.0	0.252730	0.434581	0.0	0.0	0.0	1.0	1.0
Stroke	70692.0	0.062171	0.241468	0.0	0.0	0.0	0.0	1.0
HighBP	70692.0	0.563458	0.495960	0.0	0.0	1.0	1.0	1.0
Diabetes	70692.0	0.500000	0.500004	0.0	0.0	0.5	1.0	1.0

Figure 3.3: Descriptive statistics of properties in the dataset

⇒ Most columns are in the form of values of 0 and 1, except for the columns Age, BMI, MentHlth(0 - 30), PhysHlth (0 - 30), GenHlth (1 - 5).

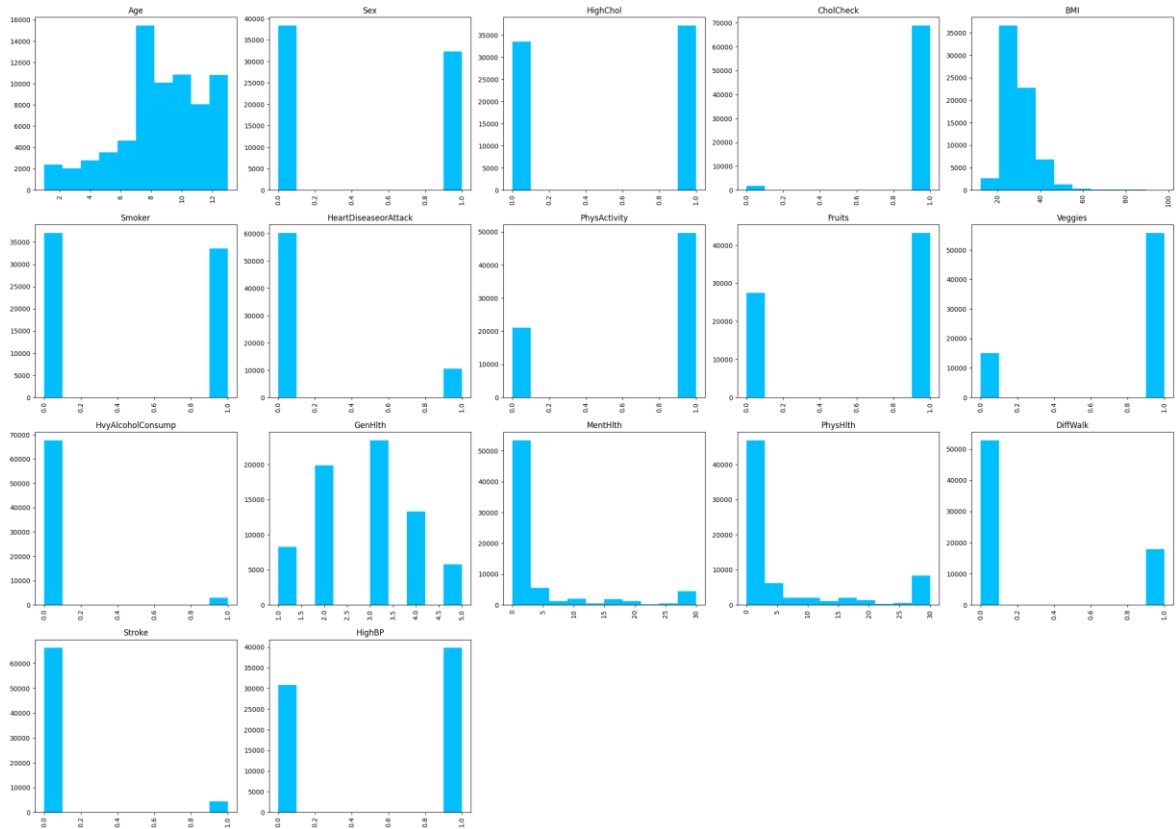


Figure 3.4: Bar charts of health variables

The bar plots provide visual summaries of different health-related variables from a dataset. Here's an explanation of each plot:

- **Age:** The age distribution is categorized into 13 groups, each representing a 5-year age range. The highest frequency is in the group aged 70-74, followed by 65-69 and 75-79.
- **Sex:** The dataset contains more females (0) than males (1).
- **HighChol (High Cholesterol):** A significant portion of the sample has high cholesterol (1).
- **CholCheck (Cholesterol Check):** Almost everyone in the dataset has had their cholesterol checked in the past 5 years (1).
- **BMI (Body Mass Index):** The BMI values are spread across a wide range, with a significant concentration around the lower end of the scale.
- **Smoker:** A notable portion of the sample has smoked at least 100 cigarettes in their lifetime (1).

- **HeartDiseaseorAttack (Heart Disease or Attack):** Most individuals in the dataset do not have a history of coronary heart disease or myocardial infarction (0).
- **PhysActivity (Physical Activity):** The majority of the sample has engaged in physical activity in the past 30 days (1).
- **Fruits:** Many individuals consume fruits one or more times per day (1).
- **Veggies:** Similarly, a significant portion of the sample consumes vegetables one or more times per day (1).
- **HvyAlcoholConsump (Heavy Alcohol Consumption):** Very few individuals consume alcohol heavily (1).
- **GenHlth (General Health):** The general health is distributed across the scale, with the highest frequencies in the "very good" (2) and "good" (3) categories.
- **MentHlth (Mental Health):** Most individuals report fewer days of poor mental health, with a concentration at the lower end of the scale (1-5 days).
- **PhysHlth (Physical Health):** Similar to mental health, most report fewer days of physical illness or injury, with a concentration at the lower end of the scale (1-5 days).
- **DiffWalk (Difficulty Walking):** Most individuals do not have difficulty walking or climbing stairs (0).
- **Stroke:** The vast majority have never had a stroke (0).
- **HighBP (High Blood Pressure):** There is a significant portion of the sample with high blood pressure (1).

3.2. Data preprocessing

3.2.1. Drop column HvyAlcoholConsump and CholCheck

Follow the chart above, some attributes with very skewed frequencies, such as **HvyAlcoholConsump** and **CholCheck**, may provide less information for the model. This is because the imbalance in their distributions can lead to biased predictions and

limited generalization. Therefore, it is advisable to drop these columns to improve the model's performance and accuracy.

3.2.2. Check correlation of other columns with Diabetes column

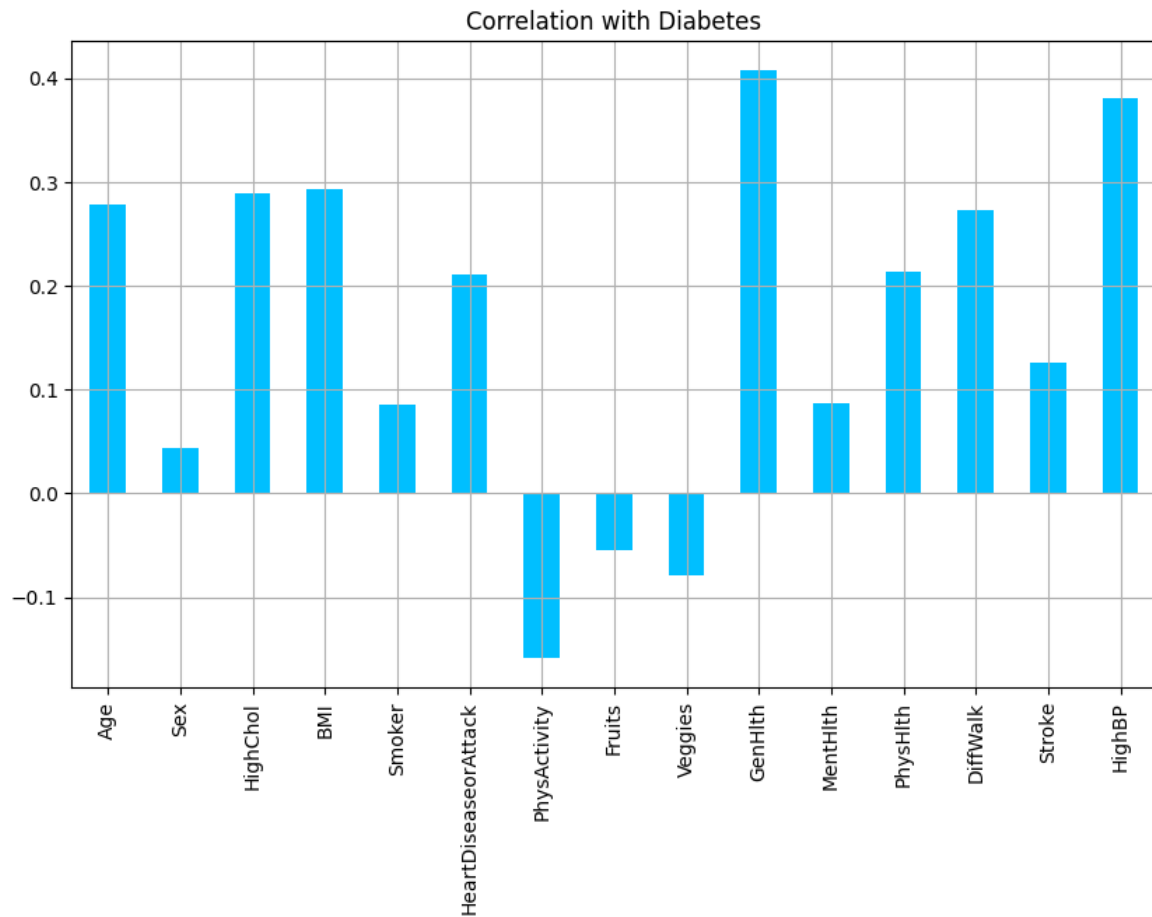


Figure 3.5: Correlation of other columns with Diabetes column

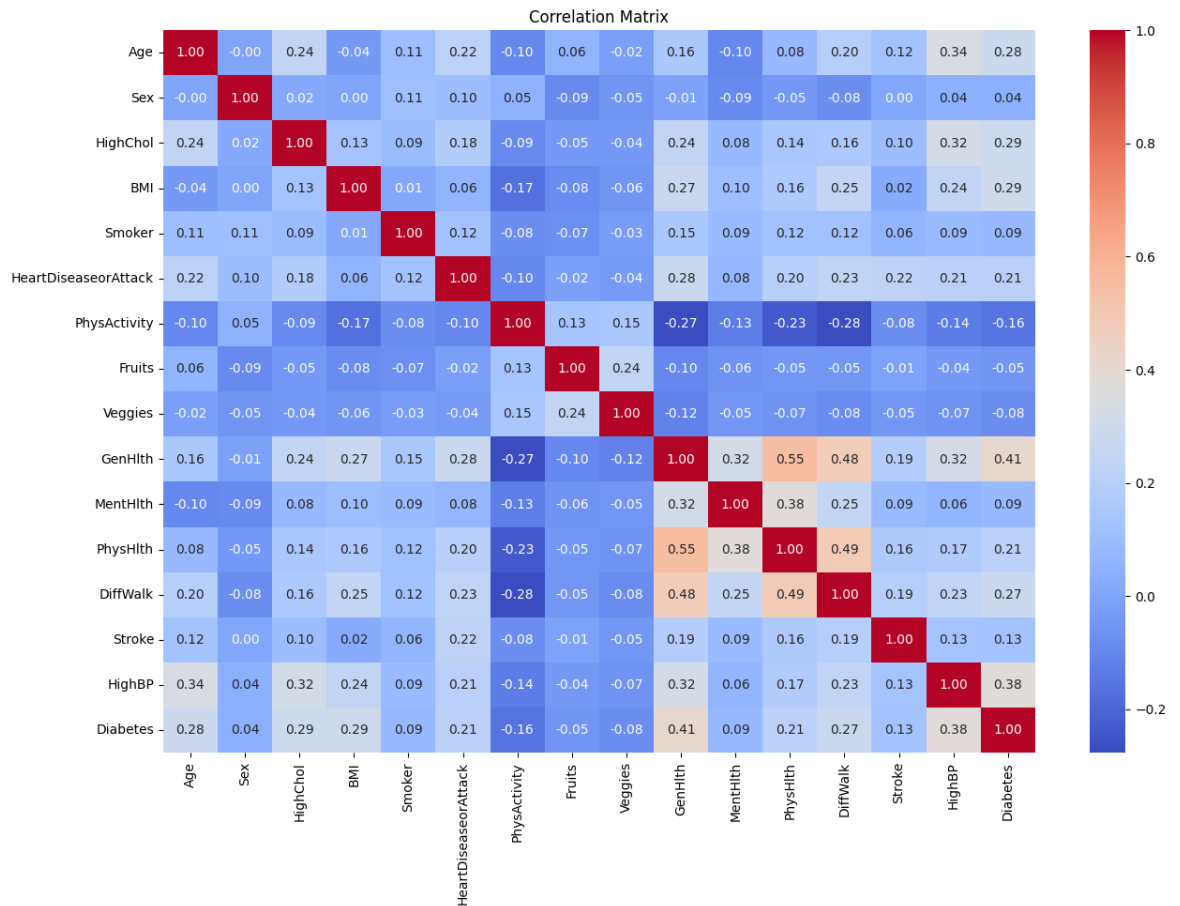


Figure 3.6: Correlation matrix between columns in the dataset

According to the two charts above, the correlation between the Diabetes column and other columns shows that Sex, Smoker, Fruits, Veggies, and MentHlth have low correlation with Diabetes. Therefore, the group has decided to remove these columns to reduce the bias in the models.

So the remaining columns after deletion (11 columns).

```
df.columns
[45]
... Index(['Age', 'HighChol', 'BMI', 'HeartDiseaseorAttack', 'PhysActivity',
        'GenHlth', 'PhysHlth', 'DiffWalk', 'Stroke', 'HighBP', 'Diabetes'],
        dtype='object')
```

Figure 3.7: The remaining columns can be used

	Age	HighChol	BMI	HeartDiseaseorAttack	PhysActivity	GenHlth	PhysHlth	DiffWalk	Stroke	HighBP	Diabetes
0	4	0	26	0	1	3	30	0	0	1	0
1	12	1	26	0	0	3	0	0	1	1	0
2	13	0	26	0	1	1	10	0	0	0	0
3	11	1	28	0	1	3	3	0	0	1	0
4	8	0	29	0	1	2	0	0	0	0	0
...
70687	6	1	37	0	0	4	0	0	0	0	1
70688	10	1	29	1	0	2	0	1	0	0	1
70689	13	1	25	1	0	5	0	1	0	1	1
70690	11	1	18	0	0	4	0	1	0	1	1
70691	9	1	25	1	1	2	0	0	0	1	1

70692 rows × 11 columns

Figure 3.8: Data of columns

3.2.3. Scaling data for features selection using MinMaxScaler method.

MinMaxScaler from scikit-learn to standardize numeric columns ['BMI', 'Age', 'PhysHlth', 'GenHlth'] within the DataFrame df. This scaling is essential for ensuring uniform data ranges, which is beneficial for machine learning algorithms that rely on consistent input scales. The serialized scaler object (minmax_scaler.pkl) enables efficient reuse of scaling parameters for future data transformations, maintaining data integrity and consistency across analyses.

	Age	HighChol	BMI	HeartDiseaseorAttack	PhysActivity	GenHlth	PhysHlth	DiffWalk	Stroke	HighBP	Diabetes
0	0.250000	0	0.162791	0	1	0.50	1.000000	0	0	1	0
1	0.916667	1	0.162791	0	0	0.50	0.000000	0	1	1	0
2	1.000000	0	0.162791	0	1	0.00	0.333333	0	0	0	0
3	0.833333	1	0.186047	0	1	0.50	0.100000	0	0	1	0
4	0.583333	0	0.197674	0	1	0.25	0.000000	0	0	0	0
...
70687	0.416667	1	0.290698	0	0	0.75	0.000000	0	0	0	1
70688	0.750000	1	0.197674	1	0	0.25	0.000000	1	0	0	1
70689	1.000000	1	0.151163	1	0	1.00	0.000000	1	0	1	1
70690	0.833333	1	0.069767	0	0	0.75	0.000000	1	0	1	1
70691	0.666667	1	0.151163	1	1	0.25	0.000000	0	0	1	1

70692 rows × 11 columns

Figure 3.9: Data after using MinMaxScaler

3.2.4. Feature importance

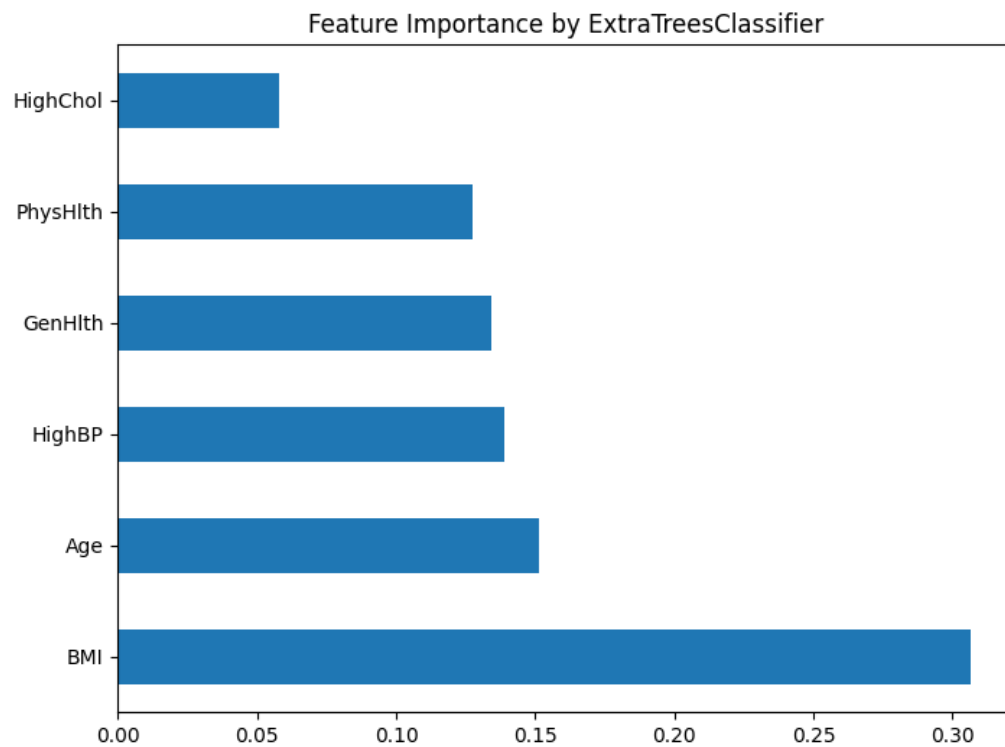


Figure 3.10: Check the importance of features

3.2.5. Diabetes distribution

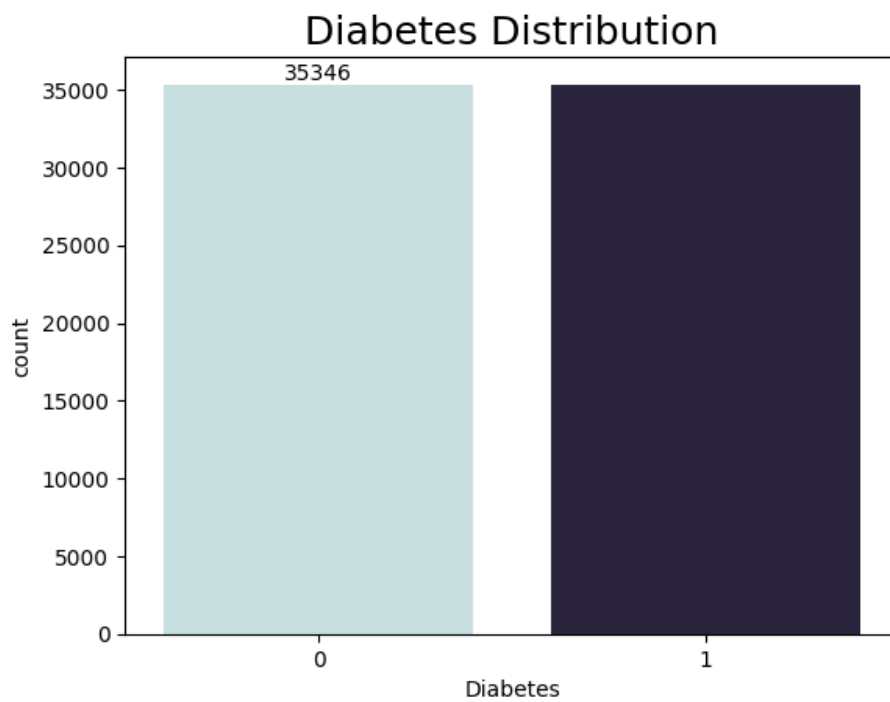


Figure 3.11: Diabetes Distribution

Because the data has already been balanced by the provider, the Diabetes column has been adjusted to ensure no further processing is needed. This contributes to higher accuracy in the data.

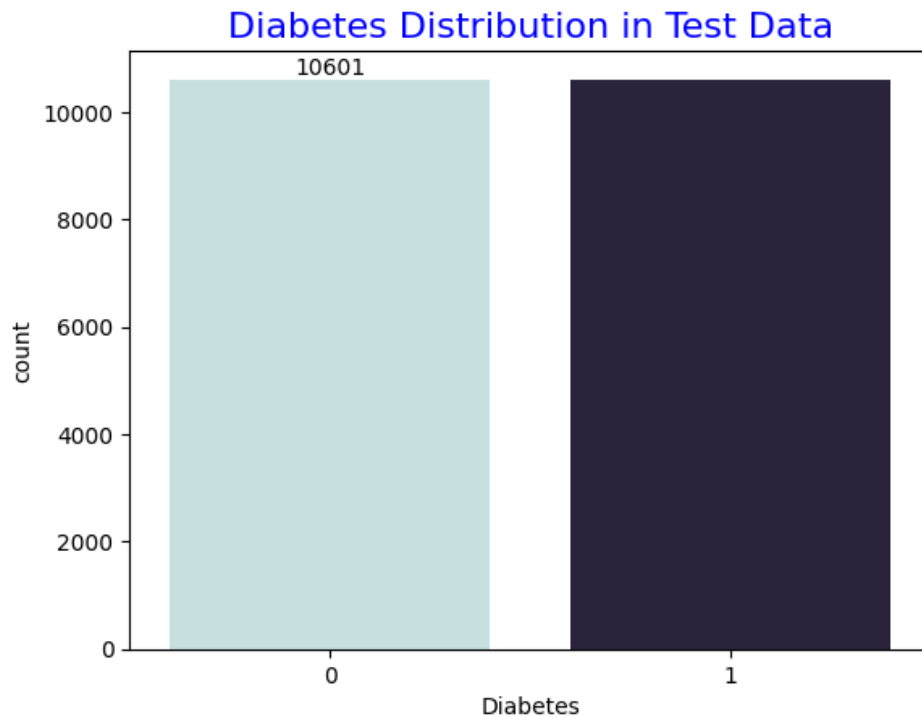


Figure 3.12: Diabetes Distribution in Test Data

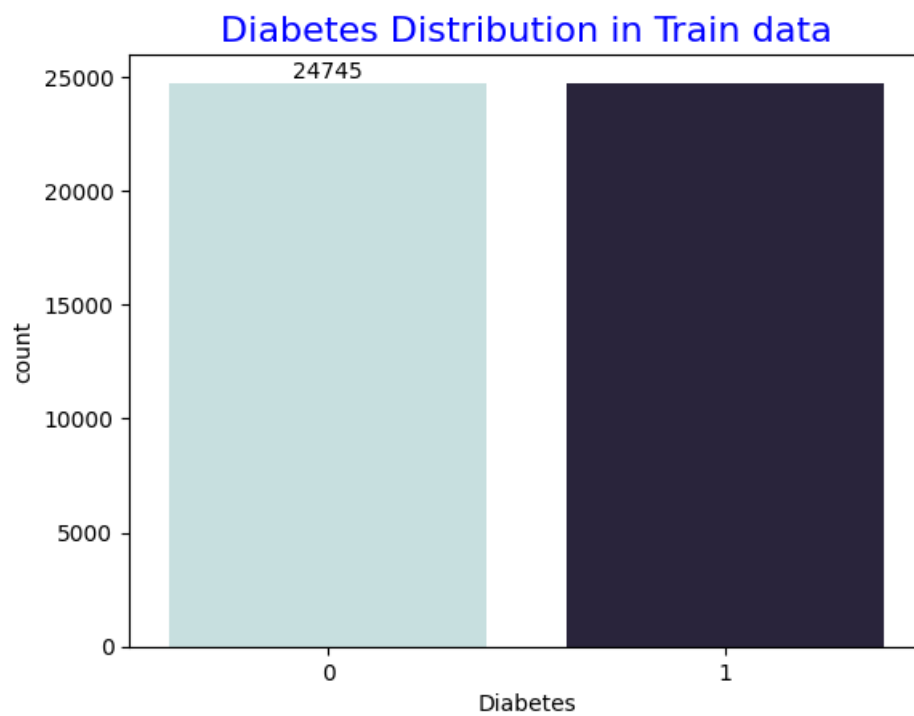


Figure 3.13: Diabetes Distribution in Train Data

Chapter 4. PREDICTIVE SOFTWARE

4.1. Technology



Figure 4.1: Streamlit's logo

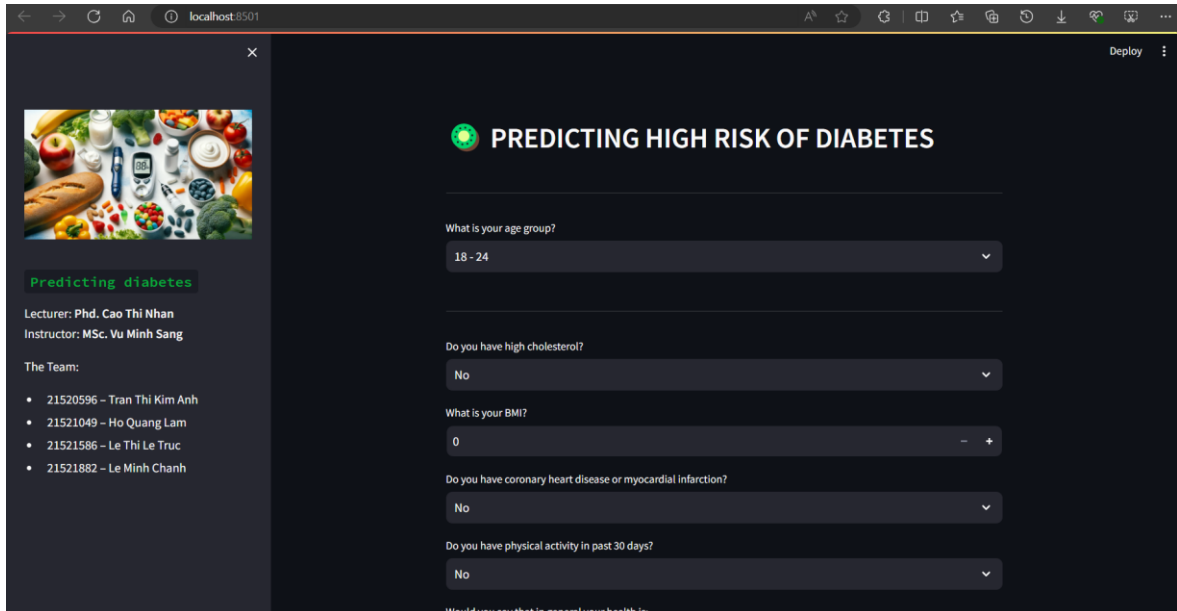
Streamlit is a Python library that simplifies the creation of web applications for exploring data and prototyping machine learning models. It enables developers to build interactive and customizable web apps using straightforward Python scripts. Streamlit streamlines the development process by allowing users to focus on coding to visualize data, generate charts, plots, tables, and other interactive elements for their applications.

4.2. Key features

- **Ease of Use:** Streamlit emphasizes simplicity, enabling users to create web apps with minimal effort using Python syntax.
- **Fast Prototyping:** Developers can quickly prototype and share their data analysis or machine learning models by leveraging Streamlit's fast iteration capabilities.
- **Interactive Components:** It offers various interactive widgets (such as sliders, buttons, text inputs, and more) that enable users to control and interact with their data visualizations dynamically.
- **Integration:** Streamlit easily integrates with popular data science libraries like Pandas, Matplotlib, Plotly, and scikit-learn, allowing seamless incorporation of these tools into web applications.
- **Customization:** Users can customize the appearance and layout of their web apps using Streamlit's flexible design options.

4.3. About the web application – Diabetes Prediction

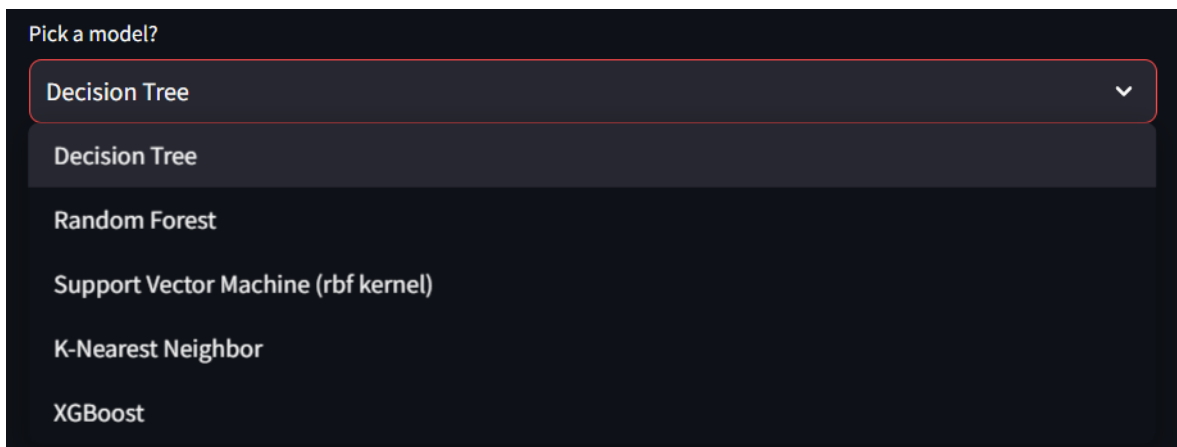
After trained 5 models (Support Vector Machine, Decision Tree, Random Forest, XGBoost and KNN) used in the project. We saved them as a pkl file and use them to applying to our web application for user to experience the Diabetes prediction.



The screenshot shows a web application titled "PREDICTING HIGH RISK OF DIABETES". On the left sidebar, there is a header "Predicting diabetes" with a green checkmark, followed by "Lecturer: Phd. Cao Thi Nhan" and "Instructor: MSc. Vu Minh Sang". Below this, "The Team:" is listed with four members: 21520596 – Tran Thi Kim Anh, 21521049 – Ho Quang Lam, 21521586 – Le Thi Le Truc, and 21521882 – Le Minh Chanh. The main content area contains a form with the following fields: "What is your age group?" (dropdown menu showing "18 - 24"), "Do you have high cholesterol?" (dropdown menu showing "No"), "What is your BMI?" (input field showing "0" with minus and plus buttons), "Do you have coronary heart disease or myocardial infarction?" (dropdown menu showing "No"), "Do you have physical activity in past 30 days?" (dropdown menu showing "No"), and a partially visible "Would you say that in general your health is:" field.

Figure 4.2: User interface - Select parameters

The web application allows users to choose one of several algorithms to predict results after they have entered suitable information.



The screenshot shows a dropdown menu titled "Pick a model?". The selected option is "Decision Tree". The dropdown list is open, showing the following options: "Decision Tree", "Random Forest", "Support Vector Machine (rbf kernel)", "K-Nearest Neighbor", and "XGBoost".

Figure 4.3: Pick a model to predict

The result will be whether the patient is having a high risk of diabetes or not.

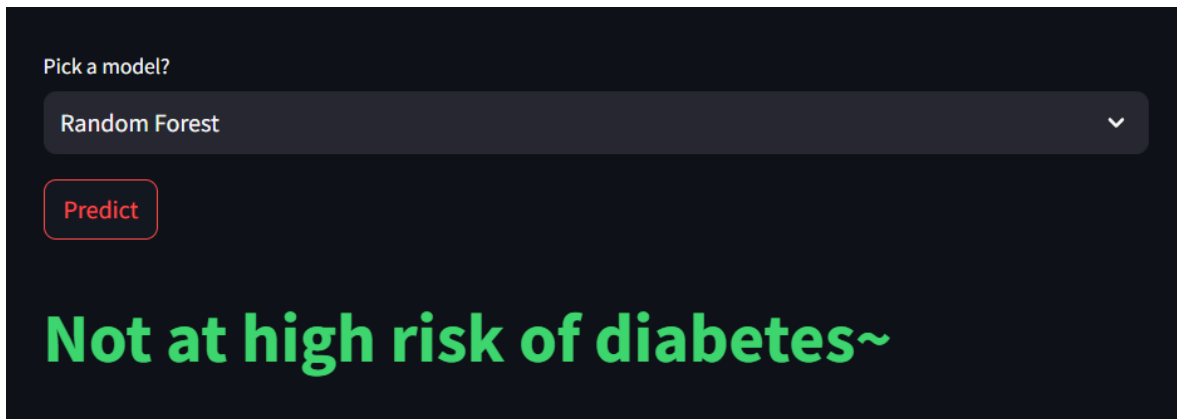


Figure 4.4: Result: Not at high risk of diabetes

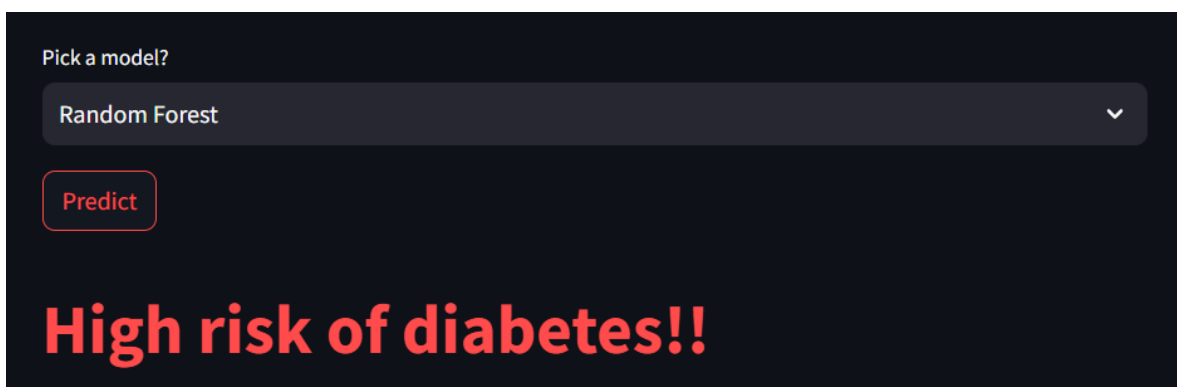


Figure 4.5: Result: High risk of diabetes

4.4. Results

Table 1: F1-Score and Accuracy Score of Models

Algorithms	F1 – Score (%)	Accuracy Score (%)
Decision Tree (DT)	74.7	71.9
Random Forest (RF)	74.3	73.5
K-Nearest Neighbors (KNN)	71.7	70.9
Support Vector Machine kernel rbf (SVM)	71.7	70.9
Extreme Gradient Boost (XGBoost)	71.7	70.9

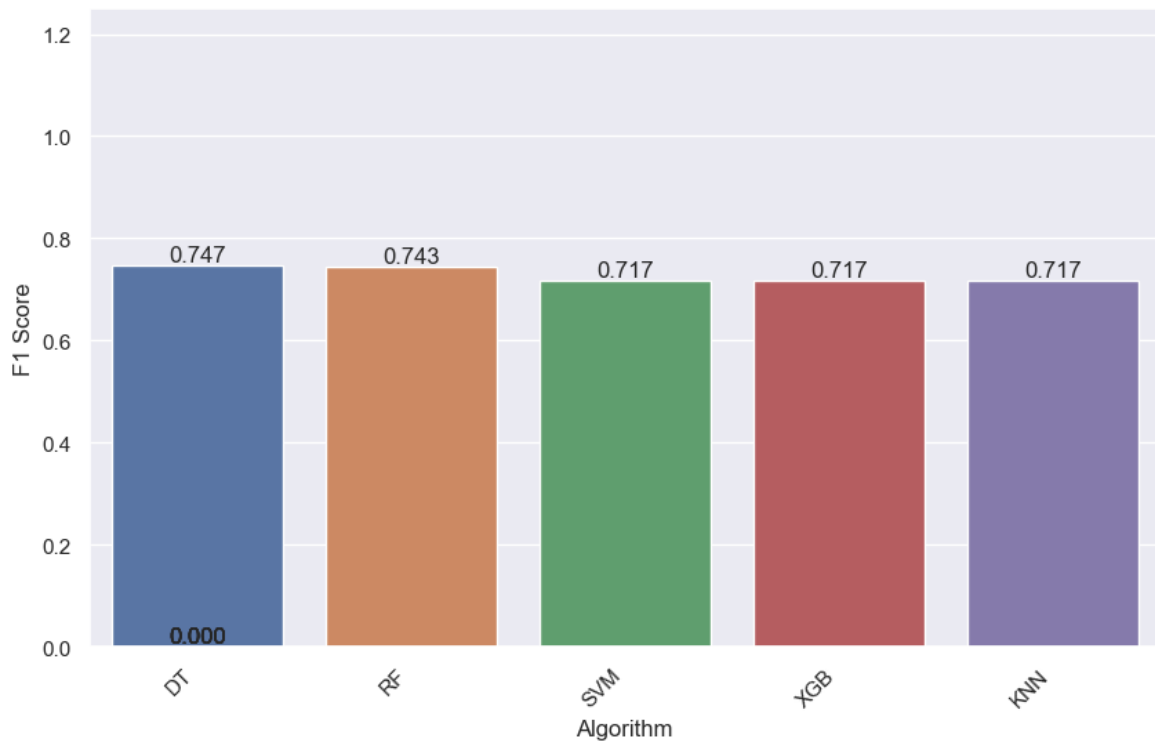


Figure 4.6: The chart shows F1-Score

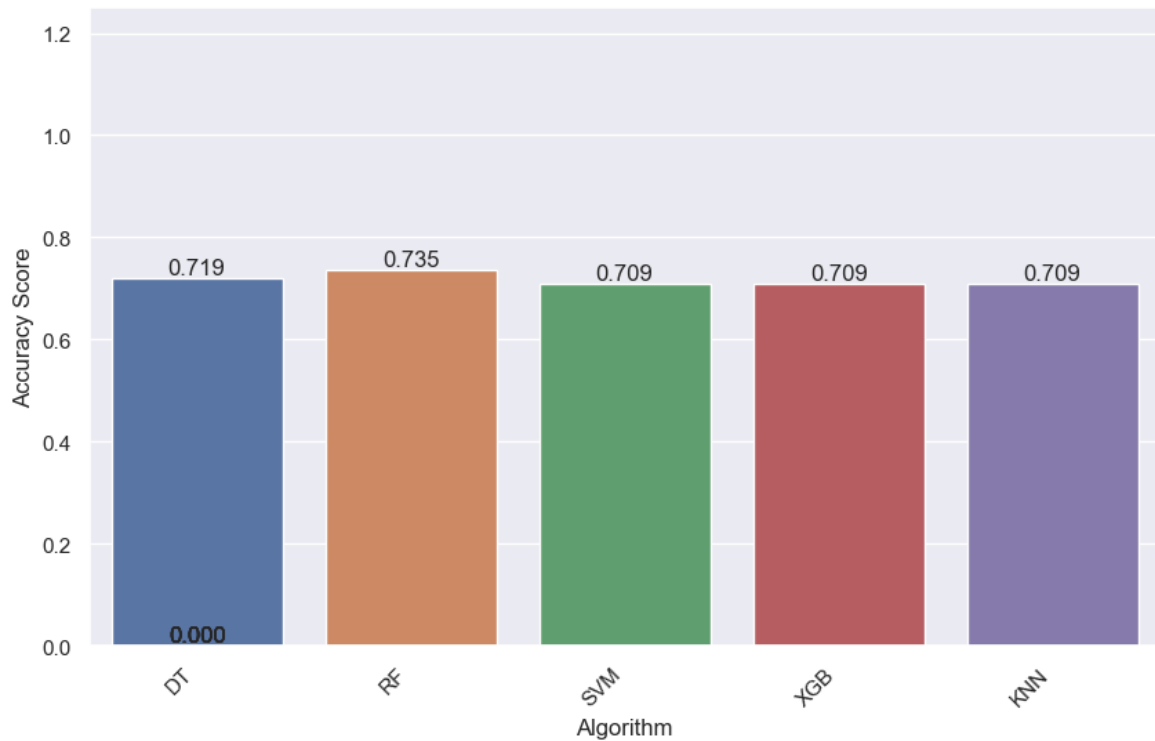


Figure 4.7: The chart shows Accuracy Score

The results presented in the table display the F1-scores and accuracy scores for various algorithms, showcasing their performance in a comparative manner. The algorithms evaluated include Decision Tree (DT), Random Forest (RF), K-Nearest Neighbors (KNN), Support Vector Machine with an RBF kernel (SVM), and Extreme Gradient Boost (XGBoost). Among these, the Decision Tree algorithm achieved the highest F1-score of 74.7% and an accuracy score of 71.9%, highlighting its effectiveness in classifying the data accurately. Random Forest follows closely with an F1-score of 74.3% and the highest accuracy score of 73.5%, demonstrating robust performance and slightly better accuracy compared to the Decision Tree.

K-Nearest Neighbors, Support Vector Machine with an RBF kernel, and Extreme Gradient Boost all achieved identical F1-scores of 71.7%, indicating comparable effectiveness in terms of the balance between precision and recall. However, their accuracy scores are also the same at 70.9%, reflecting their consistency in correctly predicting the classes. The uniformity in the performance of these three algorithms suggests that they might be similarly affected by the characteristics of the dataset or the parameter settings used during evaluation.

These results provide a comprehensive overview of each algorithm's performance, with Decision Tree and Random Forest demonstrating particularly strong results, especially in terms of F1-score and accuracy. The Decision Tree's high F1-score and Random Forest's top accuracy score underscore the reliability and predictive power of tree-based models. In contrast, the consistent performance of K-Nearest Neighbors, Support Vector Machine, and Extreme Gradient Boost illustrates their stable and reliable nature across different evaluation metrics. Overall, this comparative analysis highlights the strengths and weaknesses of each algorithm, offering valuable insights for selecting the appropriate model based on specific needs and performance criteria.

Chapter 5. CONCLUSION

5.1. The advantage

Diabetes is one of the most dangerous health problems globally, causing death and serious illness in patients.

Machine learning models are capable of analyzing large amounts of medical data, from risk factors to clinical symptoms, helping to produce more accurate predictions than conventional systems based on process-based methods. occlusion. This allows doctors to intervene early, minimizing the risk of diabetes. Furthermore, these models can personalize treatments, increasing efficiency and saving costs.

5.2. The disadvantage

Besides the advantages of accuracy and efficiency, applying machine learning in diabetes prediction also poses some challenges.

The issue of privacy and security of medical data needs attention, an medical teams also need to be trained to be able to understand and effectively use machine learning tools.

5.3. The development

With the potentials and challenges presented, here are some development directions that can be considered to improve the effectiveness of applying machine learning in stroke prediction and prevention:

Improve the quality and diversification of medical data: Access to multiple sources of high-quality medical data, including clinical data, images and data from monitoring devices will help improve health outcomes. accuracy of machine learning models.

Develop more adaptive and transparent machine learning models: Models that are able to explain their decision-making logic will help increase trust and acceptance among doctors and patients.

Integrating machine learning solutions into the clinical workflow: This requires close collaboration between medical professionals and computer engineers, to ensure the tool is designed to suit the needs and workflow doctor.

REFERENCE

- [1] “Lời nói đầu — Machine Learning cho dữ liệu dạng bảng,” *Machinelearningcoban.com*. [Online]. Available: https://machinelearningcoban.com/tabml_book/intro.html. [Accessed: 24-Jun-2024].
- [2] S. Ray, “Learn how to use Support Vector Machines (SVM) for data science,” *Analytics Vidhya*, 12-Sep-2017. [Online]. Available: <https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/>. [Accessed: 24-Jun-2024].
- [3] “1.4. Support vector machines,” *scikit-learn*. [Online]. Available: <https://scikit-learn.org/stable/modules/svm.html>[Accessed: 24-Jun-2024].
- [4] aswathisasidharan Follow Improve, “Support vector machine (SVM) algorithm,” *GeeksforGeeks*, 20-Jan-2021. [Online]. Available: <https://www.geeksforgeeks.org/support-vector-machine-algorithm/>. [Accessed: 24-Jun-2024].
- [5] T. Vu, “Bài 19: Support Vector Machine,” *Tiep Vu’s blog*, 09-Apr-2017. [Online]. Available: <https://machinelearningcoban.com/2017/04/09/smv/>. [Accessed: 24-Jun-2024].
- [6] “ML,” *GeeksforGeeks*, 19-Aug-2019. [Online]. Available: <https://www.geeksforgeeks.org/ml-xgboost-extreme-gradient-boosting/>. [Accessed: 24-Jun-2024].
- [7] guest_blog, “Introduction to XGBoost algorithm in machine learning,” *Analytics Vidhya*, 06-Sep-2018. [Online]. Available: <https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/>. [Accessed: 24-Jun-2024].
- [8] “Decision Tree algorithm — Machine Learning cho dữ liệu dạng bảng,” *Machinelearningcoban.com*. [Online]. Available: https://machinelearningcoban.com/tabml_book/ch_model/decision_tree.html. [Accessed: 24-Jun-2024].
- [9] “Random forest algorithm,” *www.javatpoint.com*. [Online]. Available: <https://www.javatpoint.com/machine-learning-random-forest-algorithm>. [Accessed: 24-Jun-2024].
- [10] susmit_sekhar_bhakta Follow Improve, “Random forest algorithm in machine learning,” *GeeksforGeeks*, 22-Feb-2024. [Online]. Available: <https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/>. [Accessed: 24-Jun-2024].
- [11] “Thuật Toán K-Nearest Neighbors (KNN) Siêu Cơ Bản,” *Codelearn.io*. [Online]. Available: <https://codelearn.io/sharing/thuat-toan-k-nearest-neighbors-knn>. [Accessed: 24-Jun-2024].