



Faculty of Engineering and Technology

Electrical and Computer Engineering Department

Intelligent Systems Laboratory – ENCS5141

Case Study #1—*Data Cleaning, Feature Engineering, and
Classification for the Diabetes Dataset*

Prepared By: Rivan Jaradat

Section#: 3

Instructor: Dr. Mohammad Jubran

Assistant teacher: Eng. Hanan Awawdeh

Date of Submission: 30. Nov.2024

BIRZEIT

NOV – 2024

1. Abstract:

This study aims to analyze and prepare the diabetes dataset for training. In terms of data cleaning and preprocessing, which includes handling missing values, encoding categorical variables, and expanding numerical features. The impact of these preprocessing steps is evaluated by comparing the performance of a random forest model trained on both raw and preprocessed data, focusing on key metrics such as accuracy, precision, recall, and training time. The performance of several different models will also be compared. Three models (RF, SVM, and MLP) will be trained on the preprocessed dataset, and their performance will be compared in terms of accuracy, precision, recall, F1 score, and computational efficiency. through the analysis I found , if we need high accuracy and do not mind memory and time consumption, the MLP model is the best choice. If we are looking for a balance between speed and accuracy with less memory consumption, the RFM model will be the ideal choice. If we are looking for stable performance but do not mind long time, the SVM model may be suitable, but in cases where higher speed is required

Table of Contents

1. Abstract:	I
Table of Tables	III
2. Introduction:	1
2.1. Motivation:	1
2.2. Background	1
2.3 Objective:	2
3. Procedure and Discussion:	3
3.1. Data Cleaning and Feature Engineering for the <i>Diabetes Dataset</i>	3
3.1.1. Load the dataset:	3
3.1.2. Data Exploration:	3
3.1.4. Data cleaning:	8
3.1.5. Analyze the relevance of each feature:	10
3.1.6. Encode categorical variables:	11
3.1.7. Scale or normalize numerical features for consistency:	12
3.1.8. Apply suitable dimensionality reduction techniques:	12
3.1.8. Model Evaluation:	13
3.2. Comparative Analysis of Classification Techniques	14
3.2.1. SVM:	14
3.2.2. MLP:	15
3.2.3. Random Forest model	15
3.2.4. how model parameters affect performance:	19
4. Conclusion:	21

Table of Tables

TABLE. 1. THE PERFORMANCE OF THE MODEL TRAINED ON PREPROCESSED DATA VS. RAW DATA	14
TABLE. 2 COMPARE THE PERFORMANCE OF RF, SVM, AND MLP	16
TABLE. 3.SVM PARAMETERS AFFECT PERFORMANCE	19
TABLE. 4.RFM PARAMETERS AFFECT PERFORMANCE	19
TABLE. 5.SVM PARAMETERS AFFECT PERFORMANCE	20

2. Introduction:

2.1. Motivation:

Diabetes is one of the chronic diseases that are among the biggest health challenges in the world, affecting millions of people and causing a huge burden on the health system and the economy. Hence, the motivation behind this study, which is to understand data preprocessing and its role in improving machine learning models, by understanding and analyzing the "Diabetes" dataset, to analyze the factors that affect the incidence of the disease, and apply machine learning techniques to classify patients based on those factors. With this study, we aim to improve early diabetes diagnosis through machine learning

2.2. Background

Data Exploration:

Data exploration involves analyzing descriptive statistics of variables in order to understand the overall structure of a data set. Data characteristics such as the number of samples, type and number of characteristics, and value limits are outlined. As a result of this step, early decisions can be made about how to improve the quality of the data and utilize it effectively for predictive modeling by identifying patterns or flaws in it, such as deviations or outliers.

Data visualization:

In data visualization, charts, graphs, maps, and other visual elements are used to enhance comprehension and insight of data and information. An efficient tool for transforming raw data into a more useful format and identifying patterns, trends, and relationships, it aids in the identification of patterns, trends, and relationships. Above all, data visualization simplifies complex information, facilitates decision-making, and enables effective communication of findings.

Data cleaning:

In the data cleaning process, missing values are treated using appropriate compensation methods such as mean or median or using more complex algorithms such as KNN-Imputation. Outliers that may affect the performance of the model are also identified using statistical methods such as calculating the standard deviation or box plots and then a decision is made to treat or exclude them.

Feature engineering:

is a vital step in improving the performance of predictive models. As part of the machine learning process, we begin by analyzing the importance of each feature based on methods such as correlation analysis or tree-based methods like feature importance in random forests. To make categorical features easier to handle, they are converted to numerical forms using methods such as one-hot encoding to ensure that they are understood by the algorithms. Numerical features are normalized or scaled to maintain consistency and ensure that all features are treated with equal importance during training. High-dimensional data can be used to make models more efficient and

reduce training times through dimensionality reduction techniques such as principal component analysis (PCA).

Random Forest:

Random Forests is an ensemble-based algorithm used for classification and prediction. It is essential to generate many decision trees and combine their results in order to obtain a robust classification. Due to its resistance to outliers and reduced overfitting risk, it can effectively process nonlinear data.

SVM:

SVMs separate data using a "dividing line" or decision level that achieves the largest margin between classes. They are very effective when dealing with small or high-dimensional data, but they can require careful parameterization (such as kernels) and take longer when dealing with large datasets.

MLP:

MLP is a type of artificial neural network based on multiple layers of connected neurons. Since weights need to be adjusted in the learning process, a non-linear model can represent complex patterns in data. However, a non-linear model requires well-prepared data and longer training.

2.3 Objective:

The aim of this study is to evaluate and compare the performance of three classification models - Random Forest (RF), Support Vector Machine (SVM), and Multilayer Perceptron (MLP) - on a preprocessed diabetes dataset. The study aims to answer the following research questions:

What is the difference in the performance of each model (RF, SVM, and MLP) in terms of accuracy, precision, recall, and F1 score?

What is the impact of data preprocessing techniques, such as handling missing values and gradient features, on the performance of the model?

How do computational efficiency and training time affect a model, and which model achieves the best balance between prediction accuracy and computational cost?

3.Procedure and Discussion:

3.1. Data Cleaning and Feature Engineering for the *Diabetes Dataset*

3.1. 1.Load the dataset:

In this code, diabetes dataset is loaded, using pandas library.

```
#load dataset
import pandas as pd
# read the dataset using pandas
data = pd.read_csv('diabetes dataset.csv')
|
```

figure 3. 1.Load the dataset

To display the first 5 rows of the dataset I used .head() from pandas library, this step helps to get a quick overview of the data.

```
data.head()
```

3.1.2.Data Exploration:

To quickly get a comprehensive understanding of the data structure and identify the steps required for the data cleaning and processing phase, .info() is used.

```
data.info()
```

the following information is displayed:

- Data Count: Displays the number of rows and columns in the dataset.
- Column Names: Specifies the names of the features available in the data.
- Data Type: Specifies whether the features are integers, floats, or text.
- Missing Values: Displays the number of non-missing values in each column to identify the columns that need cleaning.
- Memory Consumption: Shows the amount of memory the data consumes.

As shown in Figure 3.2, the data frame consists of 70,000 entries and 33 columns. 21 of them are of the Object data type and the rest are of the int type. This means that there is numeric and categorical data, and the data frame uses about 18.2 MB of memory. We also notice that all the columns have non-missing values of 70,000, which means that there is no missing data.

```

RangeIndex: 70000 entries, 0 to 69999
Data columns (total 34 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Target                                70000 non-null  object
1   Genetic Markers                       70000 non-null  object
2   Autoantibodies                       70000 non-null  object
3   Family History                       70000 non-null  object
4   Environmental Factors                 70000 non-null  object
5   Insulin Levels                       70000 non-null  int64
6   Age                                  70000 non-null  int64
7   BMI                                  70000 non-null  int64
8   Physical Activity                     70000 non-null  object
9   Dietary Habits                       70000 non-null  object
10  Blood Pressure                        70000 non-null  int64
11  Cholesterol Levels                   70000 non-null  int64
12  Waist Circumference                 70000 non-null  int64
13  Blood Glucose Levels                70000 non-null  int64
14  Ethnicity                           70000 non-null  object
15  Socioeconomic Factors               70000 non-null  object
16  Smoking Status                      70000 non-null  object
17  Alcohol Consumption                 70000 non-null  object
18  Glucose Tolerance Test               70000 non-null  object
19  History of PCOS                     70000 non-null  object
20  Previous Gestational Diabetes       70000 non-null  object
21  Pregnancy History                   70000 non-null  object
22  Weight Gain During Pregnancy         70000 non-null  int64
23  Pancreatic Health                   70000 non-null  int64
24  Pulmonary Function                  70000 non-null  int64
25  Cystic Fibrosis Diagnosis            70000 non-null  object
26  Steroid Use History                 70000 non-null  object
27  Genetic Testing                     70000 non-null  object
28  Neurological Assessments            70000 non-null  int64
29  Liver Function Tests                70000 non-null  object
30  Digestive Enzyme Levels              70000 non-null  int64
30  Digestive Enzyme Levels              70000 non-null  int64
31  Urine Test                          70000 non-null  object
32  Birth Weight                        70000 non-null  int64
33  Early Onset Symptoms                70000 non-null  object
dtypes: int64(13), object(21)
memory usage: 18.2+ MB

```

figure 3. 2.info() output

To get a statistical summary of the dataset, we used `.describe()`, where we calculate the count, mean, standard deviation, minimum, 25th percentile, 50th percentile, 75th percentile, and maximum values for each numeric column. These are the results we got for the numeric values in our dataset.

	count	mean	std	min	\
Insulin Levels	70000.0	21.607443	10.785852	5.0	
Age	70000.0	32.020700	21.043173	0.0	
BMI	70000.0	24.782943	6.014236	12.0	
Blood Pressure	70000.0	111.339543	19.945000	60.0	
Cholesterol Levels	70000.0	194.867200	44.532466	100.0	
Waist Circumference	70000.0	35.051657	6.803461	20.0	
Blood Glucose Levels	70000.0	160.701657	48.165547	80.0	
Weight Gain During Pregnancy	70000.0	15.496414	9.633096	0.0	
Pancreatic Health	70000.0	47.564243	19.984683	10.0	
Pulmonary Function	70000.0	70.264671	11.965600	30.0	
Neurological Assessments	70000.0	1.804157	0.680154	1.0	
Digestive Enzyme Levels	70000.0	46.420529	19.391089	10.0	
Birth Weight	70000.0	3097.061071	713.837300	1500.0	

	25%	50%	75%	max
Insulin Levels	13.0	19.0	28.00	49.0
Age	14.0	31.0	49.00	79.0
BMI	20.0	25.0	29.00	39.0
Blood Pressure	99.0	113.0	125.00	149.0
Cholesterol Levels	163.0	191.0	225.00	299.0
Waist Circumference	30.0	34.0	39.00	54.0
Blood Glucose Levels	121.0	152.0	194.00	299.0
Weight Gain During Pregnancy	7.0	16.0	22.00	39.0
Pancreatic Health	32.0	46.0	64.00	99.0
Pulmonary Function	63.0	72.0	79.00	89.0
Neurological Assessments	1.0	2.0	2.00	3.0
Digestive Enzyme Levels	31.0	48.0	61.00	99.0
Birth Weight	2629.0	3103.0	3656.25	4499.0

figure 3. 3.describe()

3.1.3.Data Visualization:

box plots:

They were used to identify outliers and the overall spread of the data.

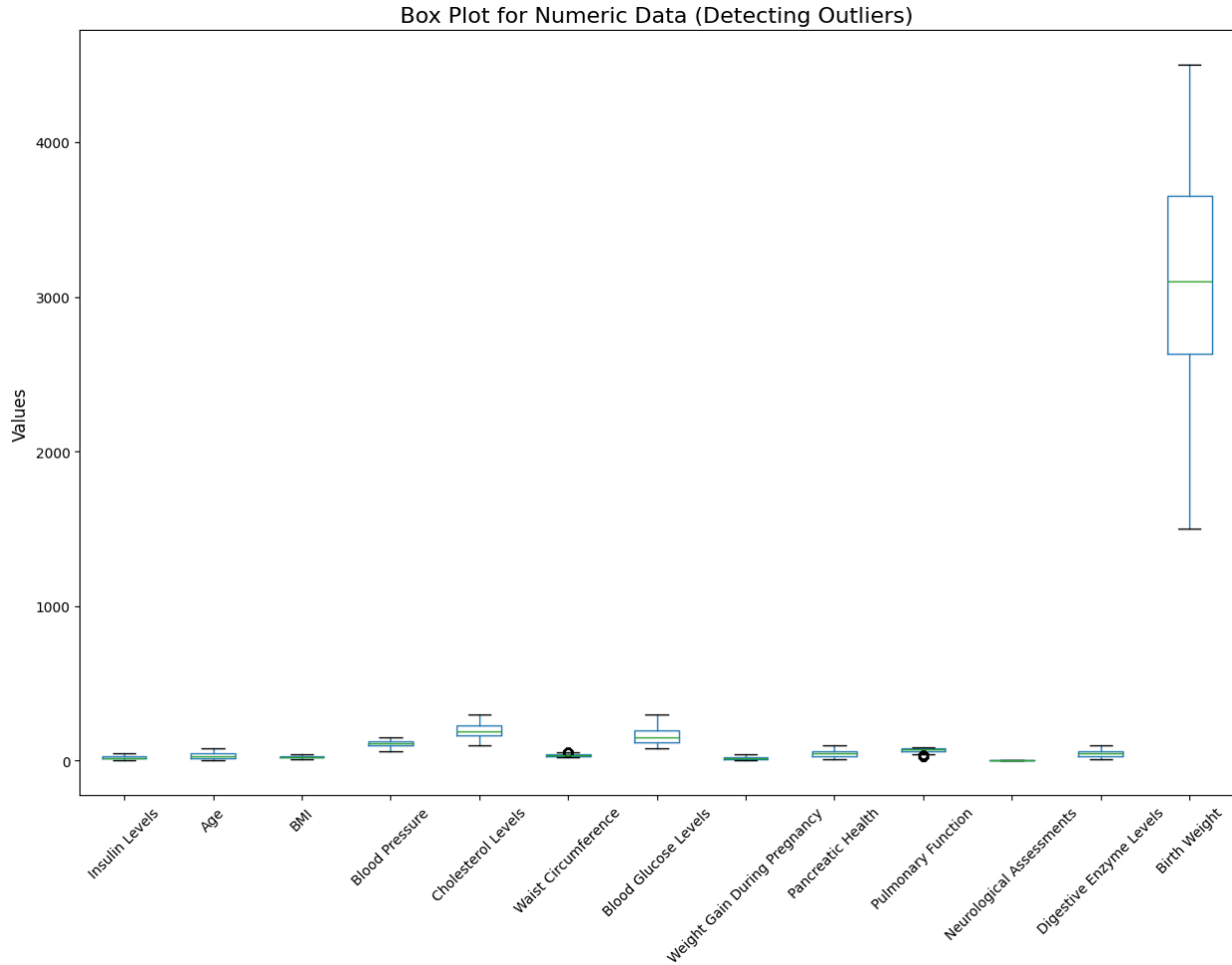


figure 3. 4.Box plot

From this Figure we find that there are several points outside the boxes ("Pulmonary Function" and "Waist Circumference"), indicating the presence of outliers that need to be addressed during data cleaning. We also notice that there is a large difference in values between the different variables, as we find that "Birth Weight" extends its range to high values compared to the other variables.

scatter plots:

The purpose of using them is to discover trends and linear or non-linear relationships between variables.

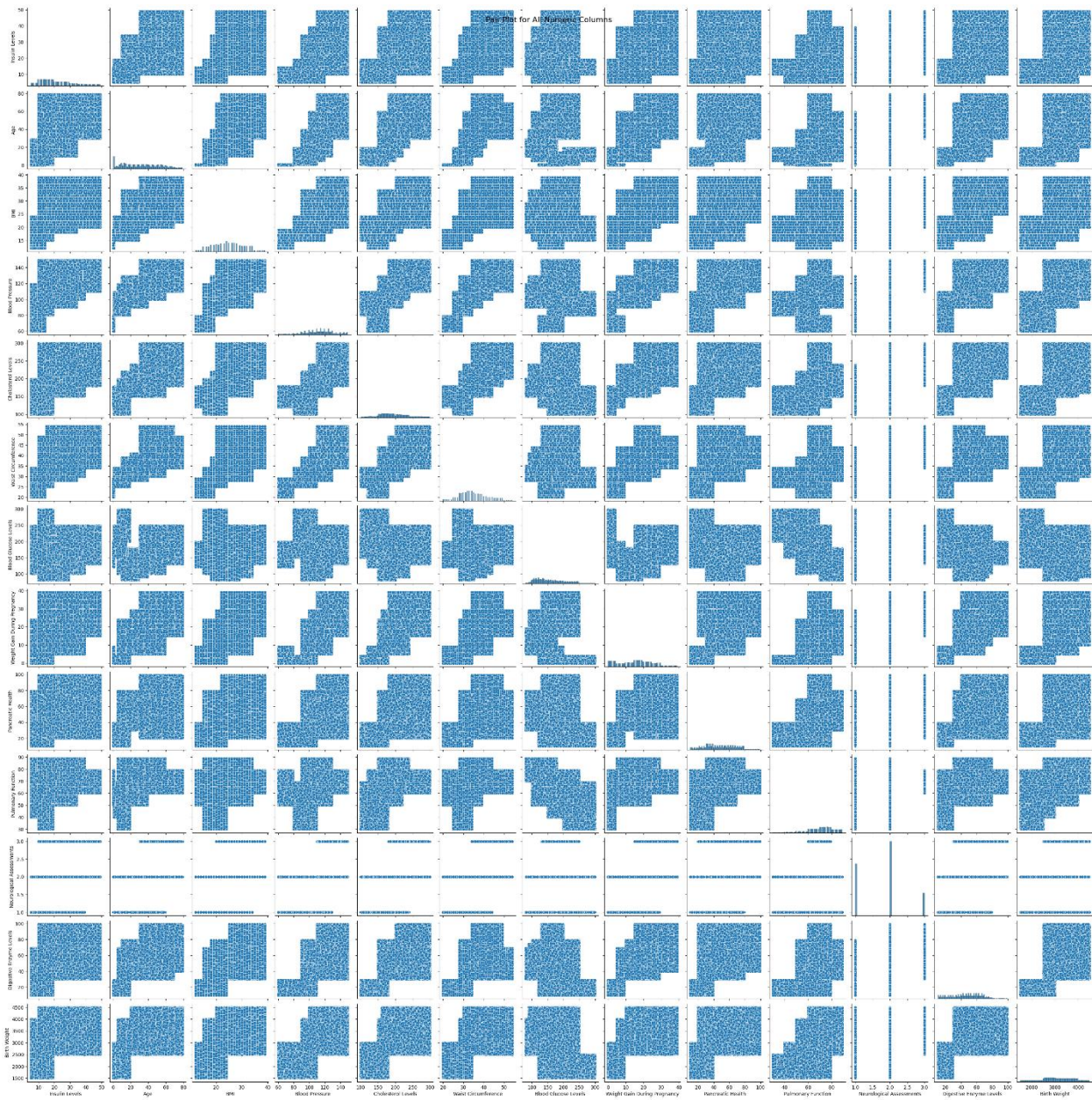


figure 3. 5.scatter plots

histograms:

The purpose of using it is to analyze the distribution and identify the features.

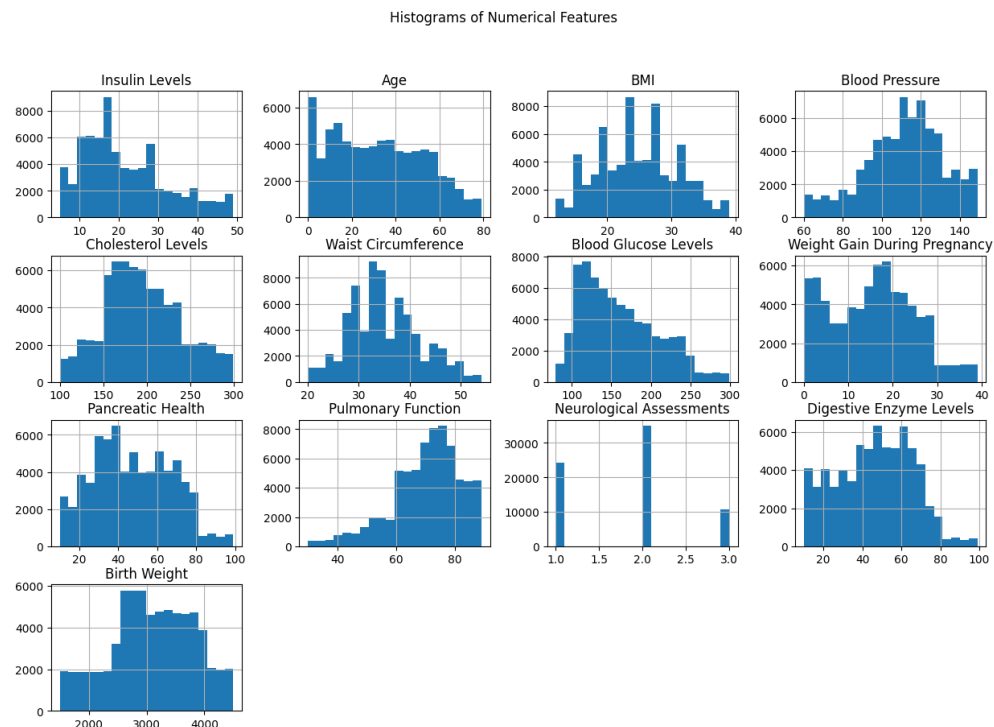


figure 3. 6.histograms

Through this image we reached these analyses

Insulin Levels: Values are highly concentrated at values below 20. There is an asymmetric distribution.

Age: Data are more concentrated at younger ages, with a marked decrease in frequency with age.

BMI (Body Mass Index): Multimodal Distribution indicates different groups. Data are often clustered around values from 20 to 35.

Blood Pressure: Distribution tends to be intermediate (between 80 and 140).

Cholesterol Levels: Values are approximately normally distributed with a concentration at 150 to 250.

Waist Circumference: Distribution centered around 30-50, which may reflect the presence of populations with specific lifestyles.

Blood Glucose Levels: Asymmetric Distribution tends to be lower (Skewed Right). Indicates a large number of individuals with low to moderate blood sugar levels.

Weight Gain During Pregnancy: Distribution is variable, showing multiple peaks.

Pancreatic Health: Data are clustered at different values, reflecting a variety of clinical measurements.

Pulmonary Function: The distribution is skewed toward high values.

Neurological Assessments: There are a few discrete values that stand out clearly.

Digestive Enzyme Levels: Distribution is fairly evenly distributed.

Birth Weight: Data show an approximately normal distribution.

3.1.4. Data cleaning:

Check missing data:

I used `.isnull().sum()` to check if the data have a missing value in each column and I get the result in figure below, there is no missing data

		Waist Circumference	0
		Blood Glucose Levels	0
	0	Ethnicity	0
Target	0	Socioeconomic Factors	0
Genetic Markers	0	Smoking Status	0
Autoantibodies	0	Alcohol Consumption	0
Family History	0	Glucose Tolerance Test	0
Environmental Factors	0	History of PCOS	0
Insulin Levels	0	Previous Gestational Diabetes	0
Age	0	Pregnancy History	0
BMI	0	Weight Gain During Pregnancy	0
Physical Activity	0	Pancreatic Health	0
Dietary Habits	0	Pulmonary Function	0
Blood Pressure	0	Cystic Fibrosis Diagnosis	0
Cholesterol Levels	0	Steroid Use History	0
Waist Circumference	0	Genetic Testing	0
		Neurological Assessments	0
		Liver Function Tests	0
		Digestive Enzyme Levels	0
		Liver Function Tests	0
		Digestive Enzyme Levels	0
		Urine Test	0
		Birth Weight	0
		Early Onset Symptoms	0

`dtype: int64`

figure 3. 7. Check missing values

also I check if the dataset have a duplicated row, from the figure we notice there no duplicated row

Number of duplicate rows before removal: 0

Check outliers:

Outliers that may affect data analysis and machine learning models will be identified. I used the "Interquartile Range (IQR)" method to identify outliers. I calculated the following values on each numeric column: the first quartile (Q1) and the third quartile (Q3), the interquartile range (IQR) as the difference between Q3 and Q1. Also, the lower and upper limits were determined using the formula:

Minimum = $Q1 - (1.5 \times IQR)$

Upper = $Q3 + (1.5 \times IQR)$

Then, the number of values that fall outside these limits for each column is found, and they are considered outliers.

```
IQR:
Insulin Levels: 0 outliers
Age: 0 outliers
BMI: 0 outliers
Blood Pressure: 0 outliers
Cholesterol Levels: 0 outliers
Waist Circumference: 522 outliers
Blood Glucose Levels: 0 outliers
Weight Gain During Pregnancy: 0 outliers
Pancreatic Health: 0 outliers
Pulmonary Function: 1206 outliers
Neurological Assessments: 0 outliers
Digestive Enzyme Levels: 0 outliers
Birth Weight: 0 outliers
```

figure 3. 8.outliet output

As the results show, most of the columns such as insulin levels, BMI, and blood pressure do not contain outliers. However, there are some columns that contain a wide range of outliers:

Waist circumference: contains 522 outliers.

Lung function: contains 1206 outliers.

The presence of a large number of outliers in these columns may negatively affect the performance of machine learning models. Therefore, it is necessary to process these values in a way that ensures maintaining the quality and accuracy of the data. Since the number of outliers is large, dropping these values may result in the loss of a large amount of important data, which affects the representativeness of the studied sample. Therefore, it is better to replace the outliers with more appropriate methods such as using the mean, as these methods can reduce the impact of outliers while maintaining the overall data size.

```
features_to_impute = ["Waist Circumference", "Pulmonary Function"]
for column in features_to_impute:
    median_value = numeric_data[column].median()
    numeric_data[column] = numeric_data[column].where(~outliers[column], median_value)
```

All outliers handled successfully as shown below


```

Number of outliers in the selected columns:
Waist Circumference    0
Pulmonary Function    0
dtype: int64

```

figure 3. 9.after remove outlier output

Also we can notice the result by box plot

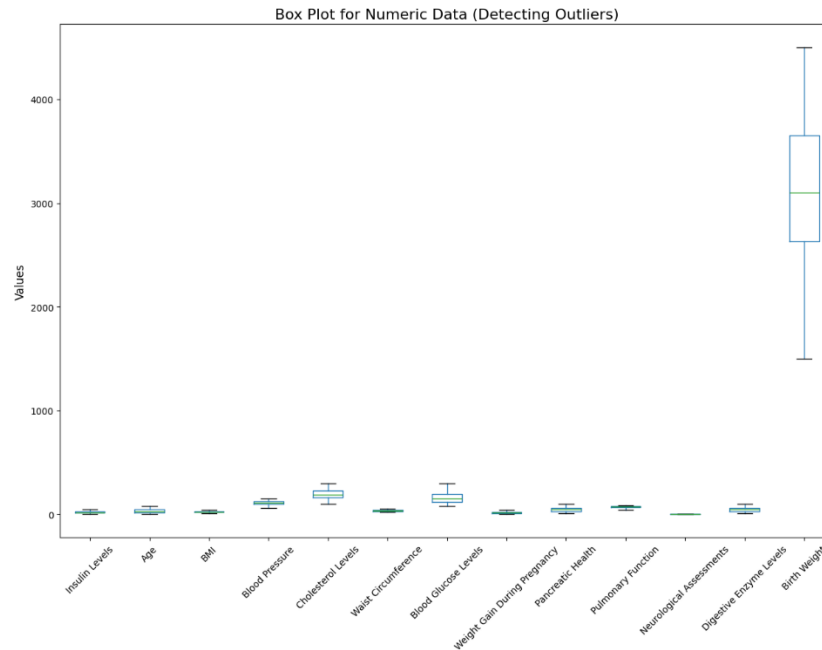


figure 3. 10.box plot after remove outlier

3.1.5. Analyze the relevance of each feature:

I used variance to assess how important each feature is in the dataset. Variance shows how much the values of a feature differ from each other, helping to understand how unique or distinct that feature is. Features with high variance are usually more important because they carry more useful information for machine learning models. On the other hand, features with very low variance may not contribute much and could even negatively affect the model, so they might need to be removed.

Insulin Levels	116.334604
Age	442.815140
BMI	36.171031
Blood Pressure	397.803022
Cholesterol Levels	1983.140552
Waist Circumference	43.735368
Blood Glucose Levels	2319.919962
Weight Gain During Pregnancy	92.796541
Pancreatic Health	399.387550
Pulmonary Function	102.695390
Neurological Assessments	0.462609
Digestive Enzyme Levels	376.014342
Birth Weight	509563.691452
dtype:	float64

```

variance=numeric_data.var()
print(variance)

```

figure 3. 11.variance output

From the results we notice a variety of variance values where there are very high values such as Birth Weight which has the highest variance among all features, indicating the importance of this feature, and there are features with medium variance: such as Age These features have moderate variance, which makes them useful, but their effect depends on the relationship with the output. There are features with low variance which indicates that the values are almost constant or have little change, and may be irrelevant to the task such as Neurological Assessments where they will be removed

The feature after removes Neurological Assessments

```

Index(['Target', 'Genetic Markers', 'Autoantibodies', 'Family History',
      'Environmental Factors', 'Insulin Levels', 'Age', 'BMI',
      'Physical Activity', 'Dietary Habits', 'Blood Pressure',
      'Cholesterol Levels', 'Waist Circumference', 'Blood Glucose Levels',
      'Ethnicity', 'Socioeconomic Factors', 'Smoking Status',
      'Alcohol Consumption', 'Glucose Tolerance Test', 'History of PCOS',
      'Previous Gestational Diabetes', 'Pregnancy History',
      'Weight Gain During Pregnancy', 'Pancreatic Health',
      'Pulmonary Function', 'Cystic Fibrosis Diagnosis',
      'Steroid Use History', 'Genetic Testing', 'Liver Function Tests',
      'Digestive Enzyme Levels', 'Urine Test', 'Birth Weight',
      'Early Onset Symptoms'],
      dtype='object')

```

figure 3. 12.The feature after removes Neurological Assessments

3.1.6.Encode categorical variables:

I converted the categorical variables in the dataset to numeric formats so that they can be used in machine learning models. Two types of encoding were applied based on the nature of the variables:

One-Hot Encoding: I used this type of encoding for most of the text features. This encoding converts each unique text value to a binary column (0 or 1).

This approach is useful when dealing with features that have a limited number of unique values, because it does not impose any ordinal relationship between the classes.

Example: The Autoantibodies feature contains the values Negative, Positive; it is converted to columns: Autoantibodies_Positive, Autoantibodies_Negative.

Label Encoding for the Target feature: I used Label Encoding for the Target feature instead of One-Hot Encoding. This type of encoding converts each categorical value to a unique number (e.g. Class A → 0, Class B → 1, and so on).

The reason for using this type of encoding is that the target feature contains a large number of unique values, and using One-Hot encoding will greatly increase the number of columns, making data processing inefficient.

Target	Genetic Testing_Negative	Genetic Testing_Positive \
7	0.0	1.0
4	1.0	0.0
5	1.0	0.0
8	0.0	1.0
12	0.0	1.0

figure 3. 13.Encode categorical variables output

3.1.7.Scale or normalize numerical features for consistency:


Data was standardized using the StandardScaler in the sklearn.preprocessing library. Standardizing data ensures that all numeric features have the same scale, which improves the performance of machine learning models. StandardScaler standardizes the data by removing the mean and making the standard deviation of each feature equal to 1, meaning that the values will be centered around zero. Standardization makes the data more balanced, which helps algorithms handle it better.

	Target	Insulin Levels	Age	BMI	Blood Pressure \
0	0.270290	1.705261	0.569277	2.197644	0.634773
1	-0.532337	-0.798037	-1.474156	-1.294096	-1.922277
2	-0.264794	0.499969	0.189103	-0.130183	0.484358
3	0.537833	-1.261610	-1.189026	-1.460369	-0.568545
4	1.608002	-0.427178	-1.046461	-1.294096	-0.418130

figure 3. 14.Standardizing output

3.1.8.Apply suitable dimensionality reduction techniques:

This section represents the final step in the preprocessing of the dataset. After the feature identification and expansion are complete, the next step is to reduce the dimensionality of the dataset, which involves reducing the number of features. To achieve this, Principal Component Analysis (PCA) is used, with the key parameter being n_components, which specifies the number of components to retain. Set to 0.95, this means that PCA retains components that explain 95% of the variance in the data.



```

0 2.584041 1.759651 0.375818 -0.358858 -0.525832 1.376637 -2.574143
1 -4.436892 -0.432942 2.198066 -1.538823 -1.185547 1.411212 -0.524321
2 0.701662 -1.635956 -0.613970 -0.562351 1.615157 0.563979 0.572206
3 -1.309889 -1.809701 -2.803346 -0.131026 -0.598807 -1.725267 -0.867185
4 -4.149032 3.456822 0.249244 0.583826 -0.343102 0.054735 -1.649507

7 8 9 ... 21 22 23 24
0 0.177514 -0.038432 -2.070886 ... 2.844597 -0.021998 -0.704805 -0.588252
1 -0.699839 2.779882 -1.200257 ... 0.723055 -0.074157 -0.390817 -0.305256
2 -1.692047 2.238478 0.156108 ... 2.871844 -0.121989 -0.708577 -0.612997
3 2.001291 0.879904 0.316779 ... -0.850348 -1.371572 1.376553 -0.498913
4 1.673832 -1.834199 -2.122501 ... -0.729075 -0.292336 -1.890915 1.914666

25 26 27 28 29 30
0 -0.749407 1.784818 1.212915 0.670275 0.476472 -1.090515
1 -1.263823 -1.491381 0.601004 -0.014584 0.435369 -0.583214
2 -0.639169 1.730831 0.347014 -0.732078 -0.405034 0.699337
3 -0.805272 1.819162 -1.913802 0.929373 -0.633746 0.752751
4 0.571713 0.154055 0.767565 -1.055986 -0.697523 0.000880


```

figure 3. 15.PCA output

3.1.8.Model Evaluation:

Split the dataset into training and testing subsets.:

In this step, we split the data into two parts: one for training and one for testing using the `train_test_split` function from the `sklearn.model_selection` library. The goal of this process is to use part of the data to train the model and the other part for testing it. It was determined that 20% of the data will be used to test the model while 80% will be used for training. Using this ratio, we obtained 56,000 training samples and 14,000 testing samples, with 31 features per sample in both parts.



```

Data train size: (56000, 31)
Data Test size (14000, 31)

```

figure 3. 16.Split the dataset

Train a Random Forest model

In this step, we trained a Random Forest model using the pre-processed data (on which PCA and standardization were applied) to compare the model's performance when using the pre-processed data with its performance when trained on the raw data (before feature selection or standardization was applied). The `RandomForestClassifier` from the `sklearn.ensemble` library was used to build the model, where we set the number of trees in the forest to 100. Next, we trained the model using the training data `X_train` and `y_train`, and then predicted the results using `X_test`.

Analyze:

Table. 1. the performance of the model trained on preprocessed data vs. raw data

Model	accuracy	precision	recall	Training Time:	Cross-validated Accuracy (Consistency):
model trained on preprocessed data	73.09%	72.41%	72.89%	70.76 seconds	72.86%
model trained on. raw data	89.51%	89.94%	89.55%	21.95 seconds	89.73%

We notice that the model trained on raw data shows much higher accuracy compared to the model trained on preprocessed data. The accuracy reaches 89.51% while the accuracy on preprocessed data is 73.09%. In this case, it indicates that the raw data contains more detail, which helps the model perform better. Precision and recall follow the same trend. On raw data, the model shows better performance, reaching 89.94% precision and 89.55% recall, compared to lower percentages on preprocessed data.

Effect of pre-processing:

Preprocessing techniques such as consolidation (e.g. PCA) improve the performance of the model, but the results here show that the raw data allowed the model to learn patterns better, which suggests that it may be more information-rich. This is because preprocessing may lose some of the finer details in the data that are valuable to the model. This leads to lower accuracy in the case of the processed data compared to the raw data, as we saw in the previous results.

Preprocessing also made the model take longer to train than on the raw data, which makes sense since preprocessing requires additional time, such as data scaling.

As for consistency between the data, we notice that consistency decreased after data processing

3.2. Comparative Analysis of Classification Techniques

3.2.1.SVM:

I created an SVM model for prediction, where the model was implemented using the RBF kernel type and the C parameter value = 1, with the gamma parameter set to 0.1. The C value controls the model's balance between accuracy and flexibility; high values make the model more accurate but may overlearn, while low values make it more able to generalize. On the other hand, the gamma parameter controls the extent of the influence of individual data points; large values focus on the nearby details, while small values lead to a broader influence but are less accurate.

I created an SVM classifier using SVC(). I then calculated performance metrics, including accuracy, precision, and recall, to evaluate the model's performance.

The following results were obtained

```
Kernel: rbf
C: 1
Accuracy: 64.58%
Precision: 64.44%
Recall: 64.43%
Training Time: 1412.8675 seconds
Prediction Time: 88.1042 seconds
Memory Usage: 1574.20 MB
```

figure3. 17.SVM result

We can improve the performance of the model by increasing the values of c and gamma.

3.2.2.MLP:

I used a multi-layer neural network (MLPClassifier) to classify and analyze the data. The network has one hidden layer with 100 neurons, using the ReLU activation function and the Adam optimizer for training. To ensure proper learning, I set the maximum number of training iterations to 200. During the process, I measured the time required for training and prediction. To evaluate the model's performance, I looked at metrics like accuracy, the classification report, and the confusion matrix to understand the results better. Additionally, I monitored memory usage to assess how efficiently the model used system resources.and i get the result below:

```
Accuracy: 73.42%
Recall: 73.25%
Precision: 73.00%
F1 Score: 73.03%
Training Time: 126.31 seconds
Prediction Time: 868.8954 seconds
Memory Usage: 1838.05 MB
```

figure3. 18.MLP result

3.2.3.Random Forest model

In this step, we trained a Random Forest model. The RandomForestClassifier from the sklearn.ensemble library was used to build the model, where we set the number of trees in the forest to 100 and random_state=123. Next, we trained the model using the training data X_train and y_train, and then predicted the results using X_test.

Accuracy: 73.21%
Recall: 73.02%
Precision: 72.56%
Memory Usage: 1827.93 MB
Time: 73.15 seconds
F1 Score: 72.66%

figure3. 191.Random Forest result

Table. 2 Compare the performance of RF, SVM, and MLP

Algorithm	Accuracy	Precision	Recall	F1 Score	Time	Memory Usage
SVM	64.58	64.43%	64.43%	64.43%	23.5 min	1574.20 MB
MLP	73.42%	73.00%	73.25%	73.03%	2.6 min	1838.05 MB
RFM	73.21%	72.56%	73.02%	72.66%	1.13 min	1827.93 MB

Accuracy: MLP has a slightly higher accuracy of 73.42%, followed by RFM with a slight difference of 73.21%, while SVM was the least accurate at 64.58%.

Precision: MLP has an Precision of 73.00%, RFM has an Precision of 72.56%, and SVM has an Precision of 64.43%.

Recall: Both MLP and RGM have a recall that is very close to their recall, with MLP having of 73.25% and RFM having an recall of 73.02%. In SVM the recall is around 64.43%.

F1-score: SVM has an F1-score of 64.43%%, while MLP has an F1-score of 73.03%% and the RFM has an F1-score of 72.66%

Model Comparison: Classification Performance and Computational Efficiency:

Time:

SVM took about 23.5 minutes to train and is considered the most time-consuming, which is quite a long time compared to other models.

MLP comes in second in terms of time, taking 2.6 minutes. Although it is faster than SVM, it still takes longer than the Random Forest model.

Random Forest (RFM) is the least time-consuming model, taking only 1.13 minutes to train, making it the fastest model among the three.

Memory Consumption:

SVM The model used the least amount of memory, consuming 1574.20 MB, which is the best in terms of memory consumption.

MLP is the worst in terms of memory, it used the most amount of memory, consuming 1838.05 MB, due to the complexity of the neural network.

Random Forests (RFM) is the second most efficient, consuming 1827.93 MB, which is slightly less than MLP but still higher than SVM.

If we want to choose the best model among the three in terms of time and memory consumption together, we find that Random Forests (RFM) was more efficient in terms of time and memory, while SVM, despite being the least memory-consuming model, takes a long time, while MLP performed well but took longer and consumed more memory compared to the other two models.

summary of model performances, discussing the strengths and weaknesses of each model:

SVM (Support Vector Machines) model:

Strengths:

The accuracy and reliability are quite good with an accuracy of 64.58%.

It takes longer to train than other models (23.5 minutes), but its performance is generally stable.

Weaknesses:

It takes longer to train, which makes it not ideal if you need fast results.

The memory consumption is relatively high (1574.20 MB).

MLP (Multilayer Neural Networks) model:

Strengths:

This model achieved the best performance in terms of accuracy (73.42%), which means it provides more accurate prediction results than other models.

Although the training time is less than SVM (2.6 minutes), it requires more memory (1838.05 MB), making it a good choice if you are looking for better accuracy.

Weaknesses:

The memory consumption is much higher than other models, which may cause problems if you are running on a machine with limited resources.

RFM (Random Forest) Model:

Strengths:

Good performance with accuracy very close to MLP (73.21%), but much faster than the other two models (1.13 minutes).

Less memory consumption (1827.93 MB), making it the best choice if you need a balance between performance and resources.

Weaknesses:

Despite its speed and memory efficiency, its accuracy is not the best among the three models, which may hinder it in some cases that require very high accuracy.

So we conclude that if we need high accuracy and do not mind memory and time consumption, the MLP model is the best choice. If we are looking for a balance between speed and accuracy with less memory consumption, the RFM model will be the ideal choice. If we are looking for stable performance but do not mind long time, the SVM model may be suitable, but in cases where higher speed is required

3.2.4.how model parameters affect performance:

To discover how model parameters affect performance. I trained each model on a set of variables to observe the effect of the variable on the performance and efficiency of the model.

MLP:

In this model, I studied the effect of the variable `hidden_layer_sizes` on the performance and efficiency of the model. I tried 3 values: 50, 100, and 150, and obtained the following results:

Table. 3.SVM parameters affect performance

hidden_layer_sizes	Accuracy	Precision	Recall	F1 Score	Time	Memory Usage
50	74.16%	73.58%	73.95%	73.44%	84.22 seconds	1837.62 MB
100	73.42%	73.00%	73.25%	73.03%	128.34 seconds	1838.05 MB
150	72.58%	71.95%	72.40%	72.02%	121.86 seconds	1840.16 MB

We notice that the values of Accuracy, Precision, Recall, and F1 Score decreased by small percentages with the increase in the value of `hidden_layer_sizes`. While it took longer with each increase in `hidden_layer_sizes`. As for Memory Usage, it increased slightly

RFM:

In this model, I studied the effect of the variable `n_estimators` on the performance and efficiency of the model. I tried 3 values: 50, 100, and 150, and obtained the following results:

Table. 4.RFM parameters affect performance

hidden_layer_sizes	Accuracy	Precision	Recall	F1 Score	Time	Memory Usage
50	72.62%	72.03%	72.43%	72.13%	36.7893 seconds	1635.64 MB
100	73.21%	72.56%	73.02%	72.66%	69.5929 second	1827.82 MB
150	73.52%	72.84%	73.33%	72.94%	138.5221 seconds	2213.21 MB

From this table we notice that all values were affected by the increase in the number of trees, as all values increased with each increase.

SVM:

In this model, I studied the effect of the variable C with different values (0.1, 1, and 10). As for the value of gamma, the best value was determined by using grid search. These are the results I obtained:

Table. 5.SVM parameters affect performance

C	gamma	Accuracy	Precision	Recall	F1 Score	Time	Memory Usage
0.1	0.01	72.54%	71.92%	72.33%	71.99%	4780.2148 seconds	1840.67 MB
1	0.01	73.19%	72.65%	73.01%	72.75%	4794.8450 seconds	1841.70 MB
10	0.01	71.65%	71.29%	71.48%	71.36%	4568.4192	1842.22 MB

Through these values, we notice the extent of the effect of the C coefficient on the efficiency and performance of the model, as all values increased with the increase in the C value, but it takes a long time, as it took about 4 hours to obtain all the values in the table.

4. Conclusion:

Our study compared three classification models: Random Forest (RF), Support Vector Machine (SVM), and Multilayer Perceptron (MLP) using the diabetes dataset. The results showed that:

As MLP is able to capture complex patterns in the data, MLP achieved higher accuracy (73.42%) compared to RF (73.21%) and SVM (64.58%), which is in line with expectations. Due to its sensitivity to data expansion and kernel approach, SVM performed poorly in terms of precision, accuracy, and recall.

Random Forest showed similar performance to MLP, but was faster and more memory efficient. Also By cleaning and expanding the data, the models became more accurate. The data pre-preparation process significantly improved the performance of the models. In terms of computational efficiency, MLP took longer (2.6 minutes) and consumed more memory (1838.05 MB), while RF and SVM were faster and less memory intensive.

Based on these results, MLP is the most accurate model, while RF provides a good balance between performance and efficiency. However, SVM performed less than expected, suggesting that it should be improved by tuning the parameters or choosing a different kernel.