
**Faculty of Engineering & Technology**

**Electrical & Computer Engineering Department**

**INTELLIGENT SYSTEMS LABORATORY – ENCS5141**

**Assignment1: Data Cleaning, Feature Engineering, and Comparative Analysis of Classification Techniques**

---

**Prepared by:**

**Name**: Mohammad Abu Shams **ID**: 1200549

**Instructor**: Dr. Mohammad Jubran

**Teaching Assistant**: Eng. Hanan Awawdeh

**Section**: 3

**Date**: 30-11-2024

**BIRZEIT**

## Abstract

This study looked at how data cleaning and feature preparation affect the performance of machine learning models for predicting diabetes. The models compared were Random Forest (RF), Support Vector Machine (SVM), and Multilayer Perceptron (MLP). Important preprocessing steps were fixing missing values, removing outliers, scaling numbers, one-hot encoding categorical data, and reducing dimensions with PCA. The models were tested on both raw and cleaned data, and their performance was measured using accuracy, precision, recall, and F1 score. The results showed that data preprocessing had a big impact on the balance between accuracy and training time. SVM was chosen as the best model because it gave a good balance between accuracy and speed. This study showed that using the right preprocessing methods is important, especially for healthcare predictions, where both performance and efficiency are needed.

# Table of Contents

**Abstract** ..... **I**

**List of Figures** ..... **III**

**List of Tables** ..... **IV**

**1. Introduction** ..... **1**

**1.1 Motivation** ..... **1**

**1.2 Background** ..... **1**

**1.3 Objective** ..... **1**

**2. Procedure and Discussion** ..... **2**

**2.1: Data Cleaning and Feature Engineering for the Diabetes Dataset** ..... **2**

**2.1.1 Import Libraries** ..... 2

**2.1.2 Load the Dataset** ..... 2

**2.1.3 Data Exploration** ..... 4

**2.1.4 Data Cleaning** ..... 10

**2.1.5 Feature Engineering** ..... 10

**2.1.6 Dimensionality Reduction** ..... 12

**2.1.7 Split the Dataset for Model Evaluation** ..... 12

**2.1.8 Model Training and Evaluation** ..... 12

**2.1.9 Results Comparison** ..... 13

**2.2: Comparative Analysis of Classification Techniques** ..... **14**

**2.2.1 Import Additional Libraries** ..... 14

**2.2.2 Prepare Models** ..... 14

**2.2.3 Train and Evaluate Models** ..... 15

**2.2.4 Concluding the Best Model** ..... 16

**3. Conclusion** ..... **17**

## List of Figures

Figure 1: Diabetes Dataset Columns ..... 2
Figure 2: Initial Data from the 'diabetes+dataset.csv' ..... 3
Figure 3: Summary Statistics of Health Indicators from the Diabetes Dataset ..... 4
Figure 4: Count of Missing Values in Each Column of the Dataset ..... 4
Figure 5: Histograms of Numerical Features in the Dataset ..... 5
Figure 6: Pair Plot of All Features in the Dataset ..... 6
Figure 7: Pair Plot of Age, BMI, Blood Pressure, and Birth Weight ..... 7
Figure 8: Correlation Matrix Heatmap of Numerical Features ..... 8
Figure 9: Box Plots of Numerical Features in the Dataset ..... 9
Figure 10: Preprocessed Dataset with Scaled and Encoded Features ..... 11

## List of Tables

Table 1: Comparison of Model Performance: Preprocessed Data vs Raw Data ..... 13
Table 2: Comparison of Model Performance Metrics and Training Time ..... 15
Table 3: Comparison of Balance Scores for Random Forest, SVM, and MLP Models ..... 16

# 1. Introduction

## 1.1 Motivation

Diabetes is a long-term disease that affects many people around the world and is a big challenge for public health. Predicting diabetes early and accurately is important for timely treatment, reducing complications, and helping patients have better outcomes. With many large medical datasets available, machine learning can help find patterns in patient data and predict if someone has the disease. I am doing this study because I want to improve how well and quickly diabetes can be predicted, which can help healthcare systems and individual patients a lot.

## 1.2 Background

Machine learning models like Random Forest (RF), Support Vector Machine (SVM), and Multilayer Perceptron (MLP) are often used to solve problems in medical data. These models need good quality data to work well, so cleaning and preparing the data is very important. Techniques like fixing missing values, scaling features, and reducing dimensions help make the data ready for the model. In this study, I will use these techniques to see how they affect the performance of the models when predicting if someone has diabetes using a well-organized dataset.

## 1.3 Objective

The main goal of this study is to evaluate the impact of data cleaning and feature preparation on the accuracy and efficiency of machine learning models for predicting diabetes. It aims to compare the performance of Random Forest (RF), Support Vector Machine (SVM), and Multilayer Perceptron (MLP) in terms of accuracy, precision, recall, and training speed. The study seeks to provide insights into the most effective preprocessing techniques and machine learning models for healthcare-related predictions, highlighting the importance of robust data preparation.

# 2. Procedure and Discussion

## 2.1: Data Cleaning and Feature Engineering for the Diabetes Dataset

### 2.1.1 Import Libraries

`pandas` and `numPy` were used for handling data, including fixing missing values, doing calculations, and changing data into formats that could be used for analysis. `matplotlib` and `seaborn` were used to make pictures like histograms, scatter plots, and heatmaps to understand how data is spread and related. For preprocessing, `StandardScaler` was used to adjust numerical features, `oneHotEncoder` changed categorical variables into numbers, and `SimpleImputer` filled in missing values with suitable numbers. Principal Component Analysis (`PCA`) was used to make the data smaller, keeping important information. `train_test_split` was used to divide the data into training and testing sets, and `cross_val_score` was used to check how good the models were. `RandomForestClassifier` was used as the machine learning model, and `accuracy_score`, `precision_score`, and `recall_score` were used to check how well it worked. The `time` module was used to measure how long different steps, like preprocessing and training, took. All these tools were used to create a complete process for data analysis and machine learning.

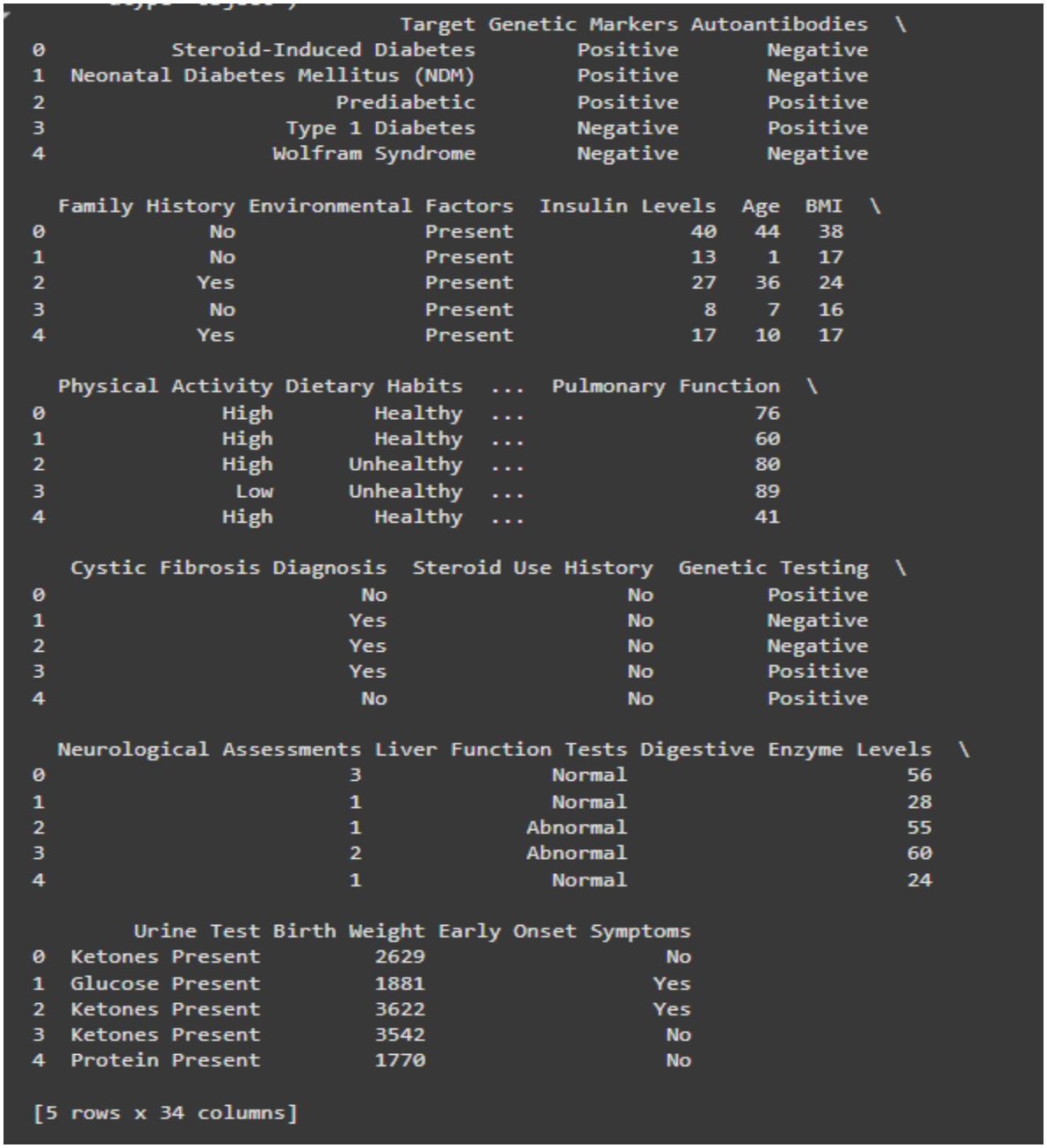
### 2.1.2 Load the Dataset

In this step, the file `'diabetes+dataset.csv'` was loaded into a Pandas DataFrame using the `pd.read_csv()` function. The `df.columns` command was executed to list all column names, providing insight into the structure of the dataset and the available features.

```
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', 'Neurological Assessments',
       'Liver Function Tests', 'Digestive Enzyme Levels', 'Urine Test',
       'Birth Weight', 'Early Onset Symptoms'],
      dtype='object')
```

*Figure 1: Diabetes Dataset Columns*

The `df.head()` function was used to display the first five rows of the data, allowing a quick inspection of the dataset's structure, data types, and sample values. This initial exploration helps in identifying the attributes that will be analyzed and processed in subsequent steps.



| | Target | Genetic Markers | Autoantibodies | \ |
|---|---|---|---|---|
| 0 | Steroid-Induced Diabetes | Positive | Negative | |
| 1 | Neonatal Diabetes Mellitus (NDM) | Positive | Negative | |
| 2 | Prediabetic | Positive | Positive | |
| 3 | Type 1 Diabetes | Negative | Positive | |
| 4 | Wolfram Syndrome | Negative | Negative | |

| | Family History | Environmental Factors | Insulin Levels | Age | BMI | \ |
|---|---|---|---|---|---|---|
| 0 | No | Present | 40 | 44 | 38 | |
| 1 | No | Present | 13 | 1 | 17 | |
| 2 | Yes | Present | 27 | 36 | 24 | |
| 3 | No | Present | 8 | 7 | 16 | |
| 4 | Yes | Present | 17 | 10 | 17 | |

| | Physical Activity | Dietary Habits | ... | Pulmonary Function | \ |
|---|---|---|---|---|---|
| 0 | High | Healthy | ... | 76 | |
| 1 | High | Healthy | ... | 60 | |
| 2 | High | Unhealthy | ... | 80 | |
| 3 | Low | Unhealthy | ... | 89 | |
| 4 | High | Healthy | ... | 41 | |

| | Cystic Fibrosis Diagnosis | Steroid Use History | Genetic Testing | \ |
|---|---|---|---|---|
| 0 | No | No | Positive | |
| 1 | Yes | No | Negative | |
| 2 | Yes | No | Negative | |
| 3 | Yes | No | Positive | |
| 4 | No | No | Positive | |

| | Neurological Assessments | Liver Function Tests | Digestive Enzyme Levels | \ |
|---|---|---|---|---|
| 0 | 3 | Normal | 56 | |
| 1 | 1 | Normal | 28 | |
| 2 | 1 | Abnormal | 55 | |
| 3 | 2 | Abnormal | 60 | |
| 4 | 1 | Normal | 24 | |

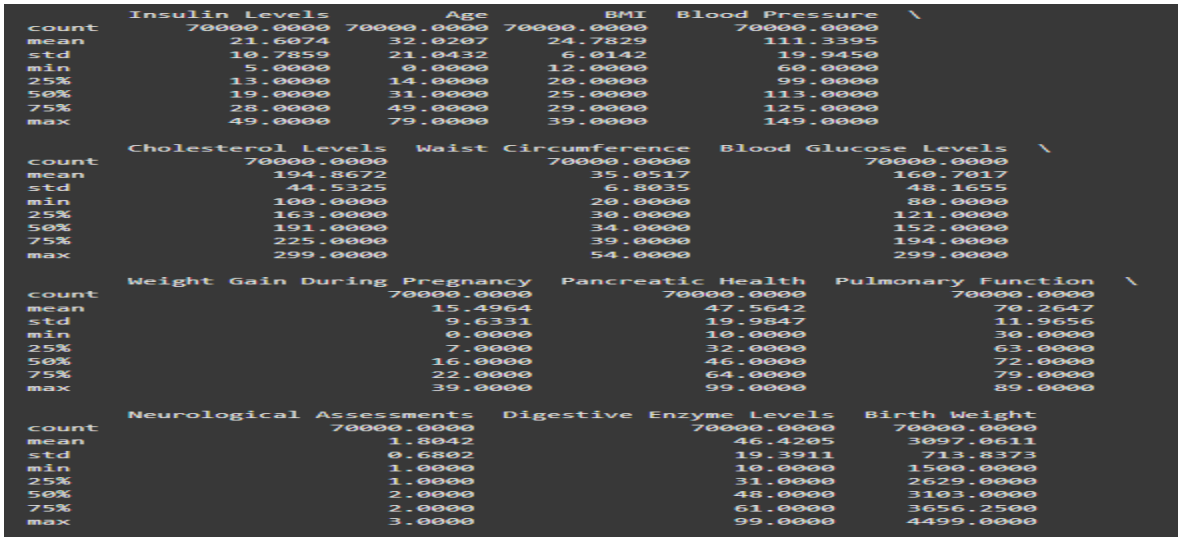
| | Urine Test | Birth Weight | Early Onset Symptoms |
|---|---|---|---|
| 0 | Ketones Present | 2629 | No |
| 1 | Glucose Present | 1881 | Yes |
| 2 | Ketones Present | 3622 | Yes |
| 3 | Ketones Present | 3542 | No |
| 4 | Protein Present | 1770 | No |

[5 rows x 34 columns]

*Figure 2: Initial Data from the 'diabetes+dataset.csv'*

### 2.1.3 Data Exploration

In this phase, the dataset 'diabetes+dataset.csv' was examined to understand the distribution and statistics of health-related variables like Insulin Levels, Age, BMI (Body Mass Index), Blood Pressure, and other health indicators. The command `df.describe()` was used to create a summary of statistics, including count, mean, standard deviation, minimum and maximum values, and the 25th, 50th (median), and 75th percentiles. This overview was important to find the central tendencies and variability in the data. Anomalies or errors, like very high or low values, were also spotted. The statistics helped to check the data's quality and ensure it was reliable for later analysis of diabetes trends.



| | Insulin Levels | Age | BMI | Blood Pressure | \ |
|---|---|---|---|---|---|
| count | 70000.0000 | 70000.0000 | 70000.0000 | 70000.0000 | |
| mean | 21.6074 | 32.0207 | 24.7829 | 111.3395 | |
| std | 10.7859 | 21.0432 | 6.0142 | 19.9450 | |
| min | 5.0000 | 0.0000 | 12.0000 | 60.0000 | |
| 25% | 13.0000 | 14.0000 | 20.0000 | 99.0000 | |
| 50% | 19.0000 | 31.0000 | 25.0000 | 113.0000 | |
| 75% | 28.0000 | 49.0000 | 29.0000 | 125.0000 | |
| max | 49.0000 | 79.0000 | 39.0000 | 149.0000 | |

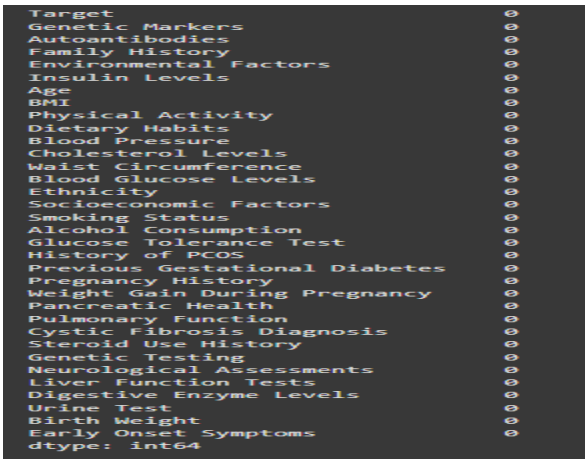
| | Cholesterol Levels | Waist Circumference | Blood Glucose Levels | \ |
|---|---|---|---|---|
| count | 70000.0000 | 70000.0000 | 70000.0000 | |
| mean | 194.8672 | 35.0517 | 160.7017 | |
| std | 44.5325 | 6.8035 | 48.1655 | |
| min | 100.0000 | 20.0000 | 80.0000 | |
| 25% | 163.0000 | 30.0000 | 121.0000 | |
| 50% | 191.0000 | 34.0000 | 152.0000 | |
| 75% | 225.0000 | 39.0000 | 194.0000 | |
| max | 299.0000 | 54.0000 | 299.0000 | |

| | Weight Gain During Pregnancy | Pancreatic Health | Pulmonary Function | \ |
|---|---|---|---|---|
| count | 70000.0000 | 70000.0000 | 70000.0000 | |
| mean | 15.4964 | 47.5642 | 70.2647 | |
| std | 9.6331 | 19.9847 | 11.9656 | |
| min | 0.0000 | 10.0000 | 30.0000 | |
| 25% | 7.0000 | 32.0000 | 63.0000 | |
| 50% | 16.0000 | 46.0000 | 72.0000 | |
| 75% | 22.0000 | 64.0000 | 79.0000 | |
| max | 39.0000 | 99.0000 | 89.0000 | |

| | Neurological Assessments | Digestive Enzyme Levels | Birth Weight |
|---|---|---|---|
| count | 70000.0000 | 70000.0000 | 70000.0000 |
| mean | 1.8042 | 46.4295 | 3097.0611 |
| std | 0.6802 | 19.3911 | 713.8373 |
| min | 1.0000 | 10.0000 | 1500.0000 |
| 25% | 1.0000 | 31.0000 | 2629.0000 |
| 50% | 2.0000 | 48.0000 | 3103.0000 |
| 75% | 2.0000 | 61.0000 | 3656.2500 |
| max | 3.0000 | 99.0000 | 4499.0000 |

*Figure 3: Summary Statistics of Health Indicators from the Diabetes Dataset*

The result of the command `df.isnull().sum()` was shown, and it was found that no missing values were present in any column of the dataset. All columns were displayed with zero missing counts. Each column was listed with its number of missing values, and it was observed that the dataset was complete. Because no missing data was detected, imputation or removal of rows was not required. The counts were provided as type `int64`, ensuring numerical precision in the results.



| | |
|---|---|
| Target | 0 |
| Genetic Markers | 0 |
| Autoantibodies | 0 |
| Family History | 0 |
| Environmental Factors | 0 |
| Insulin Levels | 0 |
| Age | 0 |
| BMI | 0 |
| Physical Activity | 0 |
| Dietary Habits | 0 |
| Blood Pressure | 0 |
| Cholesterol Levels | 0 |
| Waist Circumference | 0 |
| Blood Glucose Levels | 0 |
| Ethnicity | 0 |
| Socioeconomic Factors | 0 |
| Smoking Status | 0 |
| Alcohol Consumption | 0 |
| Glucose Tolerance Test | 0 |
| History of PCOS | 0 |
| Previous Gestational Diabetes | 0 |
| Pregnancy History | 0 |
| Weight Gain During Pregnancy | 0 |
| Pancreatic Health | 0 |
| Pulmonary Function | 0 |
| Cystic Fibrosis Diagnosis | 0 |
| Steroid Use History | 0 |
| Genetic Testing | 0 |
| Neurological Assessments | 0 |
| Liver Function Tests | 0 |
| Digestive Enzyme Levels | 0 |
| Urine Test | 0 |
| Birth Weight | 0 |
| Early Onset Symptoms | 0 |
| dtype: int64 | |

*Figure 4: Count of Missing Values in Each Column of the Dataset*

The below histograms were used to give a visual summary of how the numerical features in the dataset were spread. Each small plot was shown for one feature, and it was displayed how often the data values occurred in different ranges. For example, features like "Age" and "BMI" were observed to have normal distributions, but others, like "Neurological Assessments," were seen to have categorical or uneven spreads. This kind of chart was used to help patterns be identified, outliers be found, or data imbalances be noticed. These were considered important for understanding and cleaning the data before it was used in machine learning. The layout was designed to make many features easy to be compared at the same time.

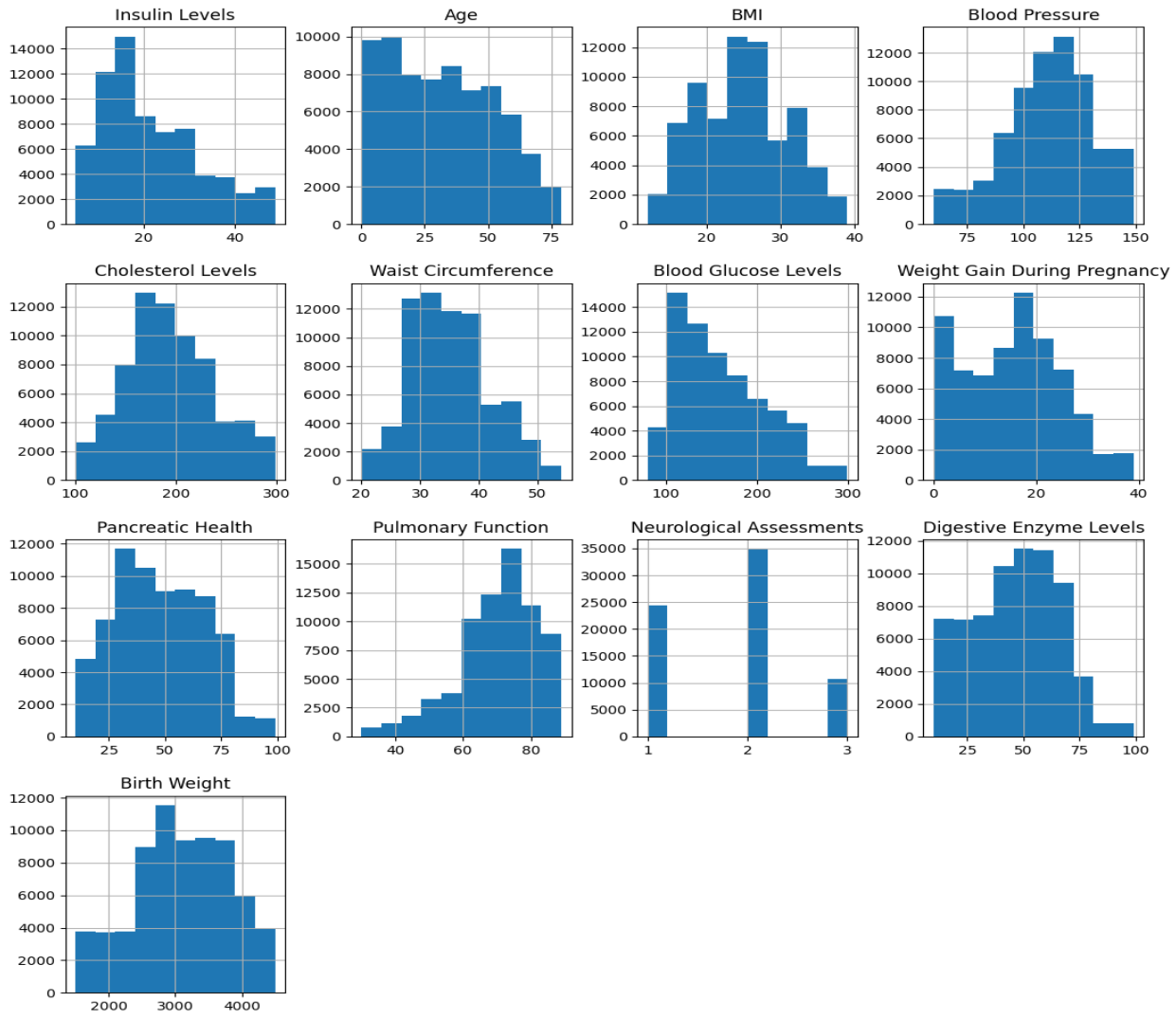



*Figure 5: Histograms of Numerical Features in the Dataset*

The pair plots below were used to show useful pictures of how features in the dataset were related and how they were spread. In the first plot, all features were included, and trends, groups, and connections between numbers were seen. A full view of the data was given, and patterns and possible problems like multicollinearity were shown.

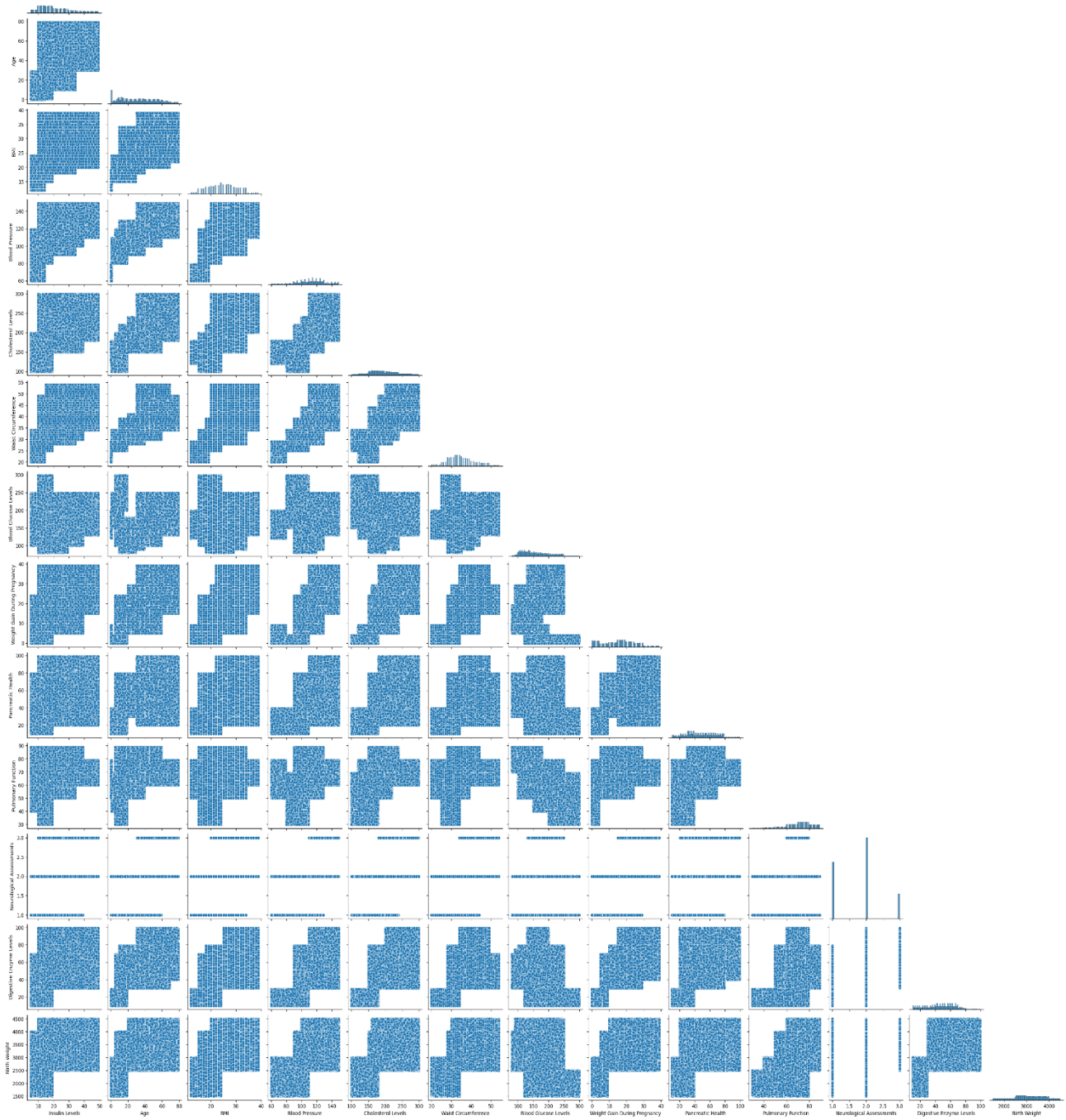



*Figure 6: Pair Plot of All Features in the Dataset*

The second pair plot was focused on a few features: Age, BMI, Blood Pressure, and Birth Weight. This special visualization was used to show more details about these important features. Their distributions were seen on the diagonal, and scatter plots were shown on other parts to display how the features were connected. This visualization was found helpful for finding possible links, patterns, or outliers in the data. These things were used to give good ideas for more analysis, making new features, and creating better models.

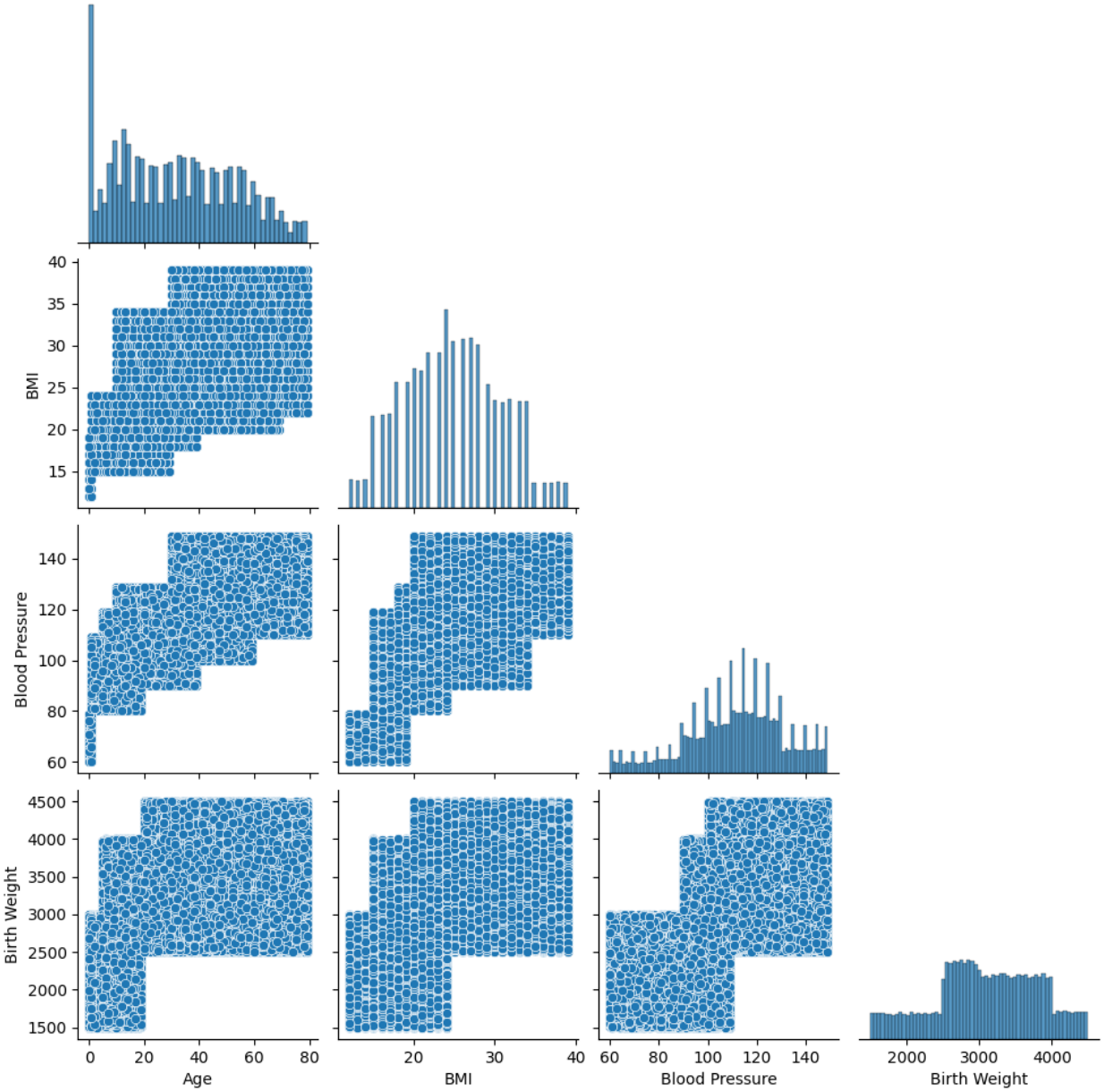



*Figure 7: Pair Plot of Age, BMI, Blood Pressure, and Birth Weight*

The heatmap below was used to show the correlation matrix for the numbers in the dataset. It was used to help understand how features were related to each other. Each square on the heatmap was shown with the Pearson correlation number between two features. This number was ranged from -1 (very negative relation) to 1 (very positive relation).

Some features like Age and Waist Circumference were found to have a strong positive relation (above 0.7). This meant they might have been changed together in a similar way. On the other hand, features like Blood Glucose Levels and Pulmonary Function were shown with weak or even negative relations with most other features.

The diagonal squares were all shown with 1.0 because a feature was always fully related to itself. The heatmap was found useful for identifying extra features or important relations that were used for making new features or better models.

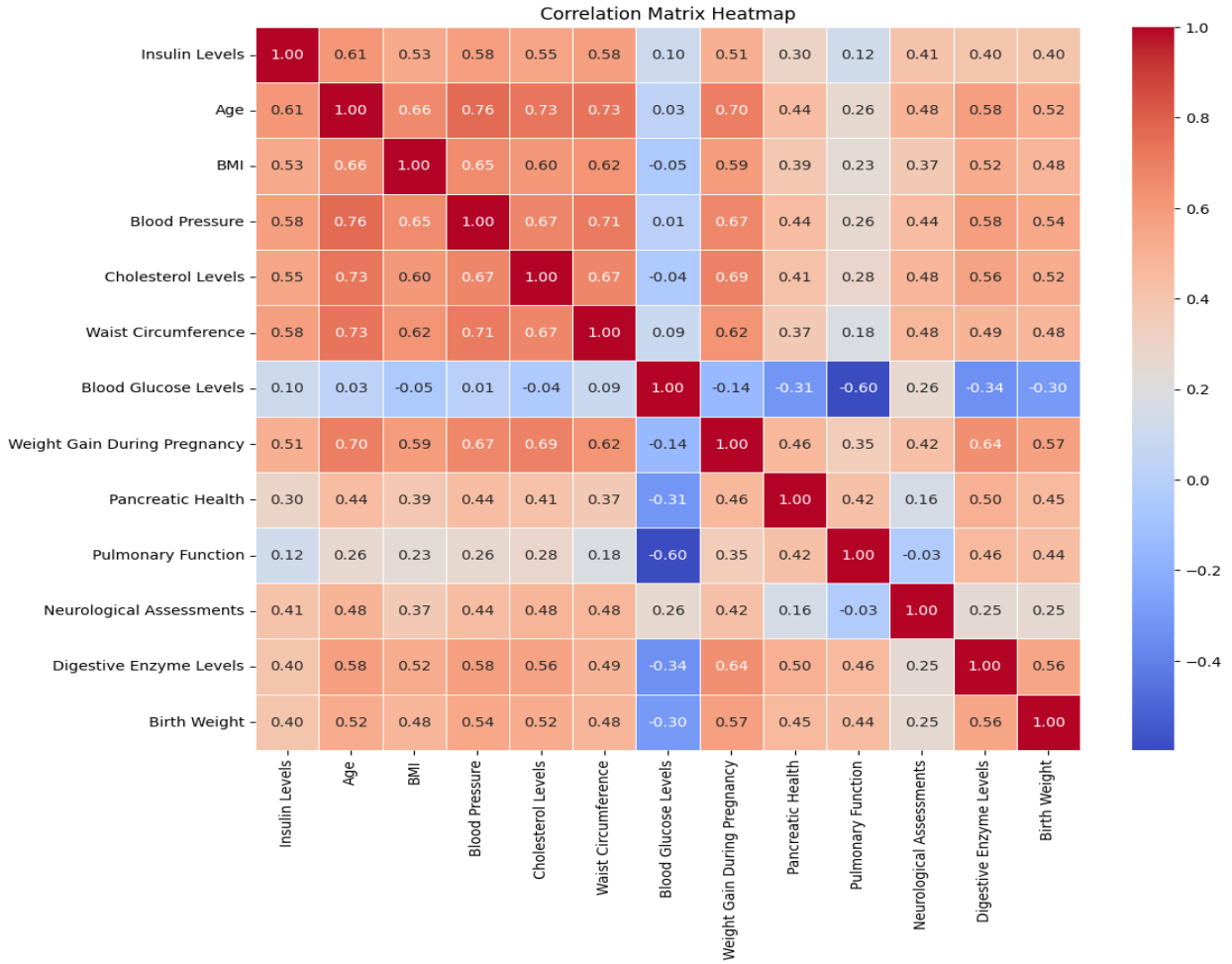



*Figure 8: Correlation Matrix Heatmap of Numerical Features*

The box plots below were used to show how the numbers in the dataset were spread. They were shown with the range, middle value (median), and how much the data was spread in the middle 50% (called IQR). The box was marked with a line inside to show the median. The lines outside the box, called whiskers, were extended to data points that were 1.5 times the IQR. If any points were found outside the whiskers, they were called outliers.

For example, features like Waist Circumference and Pulmonary Function were shown with some outliers. These might have needed to be fixed during preprocessing. Box plots were used to see how data was spread and to find strange values that might have made models not work well.

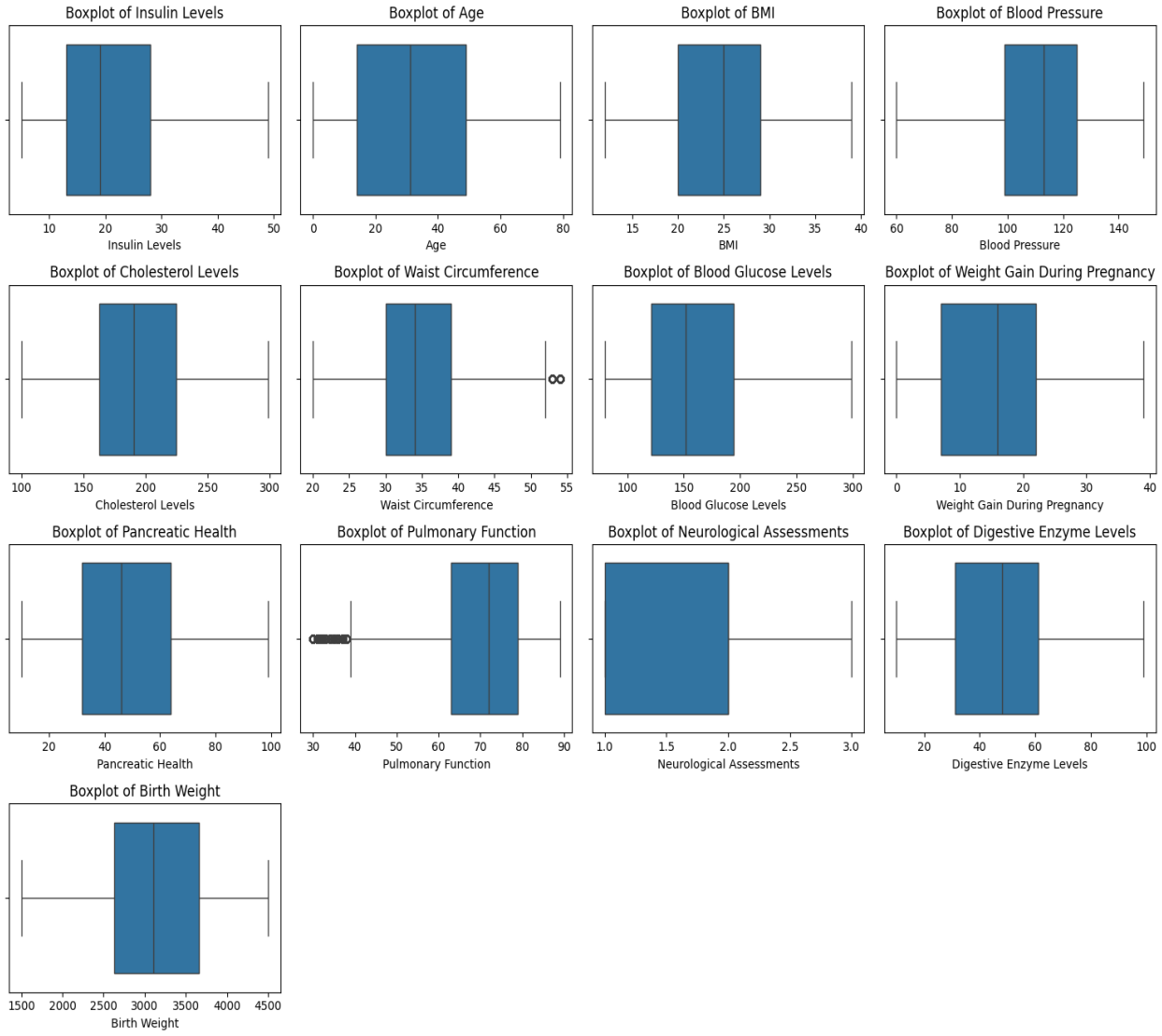



*Figure 9: Box Plots of Numerical Features in the Dataset*

### 2.1.4 Data Cleaning

In this step, data cleaning was done to make sure the dataset was good and ready for machine learning models. Missing values in number columns were fixed using imputation. The `SimpleImputer` with the mean strategy was used, and missing values were replaced with the mean of their columns. This kept the dataset's structure and avoided losing rows.

Outliers were handled by a capping method. Extreme values were clipped to stay inside the 1st and 99th percentiles of each column. For example, values higher than the 99th percentile were set to the upper limit, and values lower than the 1st percentile were set to the lower limit. This method reduced the effect of extreme values that could hurt the model's performance but kept the dataset's balance. These steps were needed to prepare good and clean data for analysis and modeling.

### 2.1.5 Feature Engineering

In this step, the dataset was prepared to make it ready for machine learning by changing categorical variables and scaling numerical features.

Categorical variables were turned into binary (one-hot) encoded values using the `OneHotEncoder`. This method created new binary columns for each category in the categorical features so machine learning models could understand them as numbers. To avoid problems like multicollinearity, the `drop='first'` option was used, which removed one category from each variable as a reference. The encoded features were saved in a Data Frame called `df_encoded`.

For numerical variables, the `StandardScaler` was applied to make the features standardized. It scaled the numbers to have a mean of zero and a standard deviation of one. This step was important for algorithms that are affected by the scale of input data. The scaled features were stored in another Data Frame called `df_scaled`.

In the end, the encoded categorical features and scaled numerical features were joined into one Data Frame, `df_preprocessed`. This dataset had both types of features in a consistent format. It helped the machine learning models learn better because all the data was standardized and encoded.

The figure below was used to show the preprocessed dataset. It was included with standardized numerical features and one-hot encoded categorical features.

| | Insulin Levels | Age | BMI | Blood Pressure | Cholesterol Levels | \ |
|---|---|---|---|---|---|---|
| 0 | 1.7094 | 0.5705 | 2.2121 | 0.6360 | 0.1380 | |
| 1 | -0.7988 | -1.4753 | -1.3030 | -1.9268 | -1.6627 | |
| 2 | 0.5018 | 0.1899 | -0.1313 | 0.4852 | -0.2222 | |
| 3 | -1.2633 | -1.1899 | -1.4703 | -0.5700 | -0.9875 | |
| 4 | -0.4272 | -1.0471 | -1.3030 | -0.4193 | -1.1000 | |

| | Waist Circumference | Blood Glucose Levels | Weight Gain During Pregnancy | \ |
|---|---|---|---|---|
| 0 | 2.2127 | 0.1530 | 0.2610 | |
| 1 | -1.6351 | 0.3616 | -0.7788 | |
| 2 | 0.1408 | -1.1618 | -0.0509 | |
| 3 | -0.8951 | -0.8279 | -0.3629 | |
| 4 | -0.3031 | 2.6155 | -1.4026 | |

| | Pancreatic Health | Pulmonary Function | ... | \ |
|---|---|---|---|---|
| 0 | -0.5800 | 0.4805 | ... | |
| 1 | -1.0824 | -0.8668 | ... | |
| 2 | 0.4248 | 0.8173 | ... | |
| 3 | 0.0731 | 1.5752 | ... | |
| 4 | -1.8361 | -2.4667 | ... | |

| | Previous Gestational Diabetes_Yes | Pregnancy History_Normal | \ |
|---|---|---|---|
| 0 | 0.0000 | 1.0000 | |
| 1 | 0.0000 | 1.0000 | |
| 2 | 0.0000 | 1.0000 | |
| 3 | 1.0000 | 1.0000 | |
| 4 | 1.0000 | 0.0000 | |

| | Cystic Fibrosis Diagnosis_Yes | Steroid Use History_Yes | \ |
|---|---|---|---|
| 0 | 0.0000 | 0.0000 | |
| 1 | 1.0000 | 0.0000 | |
| 2 | 1.0000 | 0.0000 | |
| 3 | 1.0000 | 0.0000 | |
| 4 | 0.0000 | 0.0000 | |

| | Genetic Testing_Positive | Liver Function Tests_Normal | \ |
|---|---|---|---|
| 0 | 1.0000 | 1.0000 | |
| 1 | 0.0000 | 1.0000 | |
| 2 | 0.0000 | 0.0000 | |
| 3 | 1.0000 | 0.0000 | |
| 4 | 1.0000 | 1.0000 | |

| | Urine Test_Ketones Present | Urine Test_Normal | Urine Test_Protein Present | \ |
|---|---|---|---|---|
| 0 | 1.0000 | 0.0000 | 0.0000 | |
| 1 | 0.0000 | 0.0000 | 0.0000 | |
| 2 | 1.0000 | 0.0000 | 0.0000 | |
| 3 | 1.0000 | 0.0000 | 0.0000 | |
| 4 | 0.0000 | 0.0000 | 1.0000 | |

| | Early Onset Symptoms_Yes |
|---|---|
| 0 | 0.0000 |
| 1 | 1.0000 |
| 2 | 1.0000 |
| 3 | 0.0000 |
| 4 | 0.0000 |

[5 rows x 50 columns]

*Figure 10: Preprocessed Dataset with Scaled and Encoded Features*

### 2.1.6 Dimensionality Reduction

Principal Component Analysis (PCA) was used to reduce the size of the preprocessed dataset while keeping 95% of the original information. This method was applied to turn the features into a smaller group of main components. Most of the important details were kept, and extra or noisy information was removed. The reduced dataset (`df_pca`) was made more efficient for machine learning models, helping to improve speed and maybe also performance by removing unimportant or too similar features.

### 2.1.7 Split the Dataset for Model Evaluation

The dataset was divided into training and testing sets to check how machine learning models perform. The preprocessed data, after PCA was done, was split into two parts using an 80-20 ratio with the `train_test_split` function. A random seed (`random_state=42`) was used to make the split consistent. The target column (`Target`) was kept separate in both training and testing parts. Also, the raw numerical data was split into training and testing sets to compare it with the preprocessed data. This setup was made to see how preprocessing and PCA affect the model's performance.

### 2.1.8 Model Training and Evaluation

Random Forest models were trained on the preprocessed data and raw data to compare how preprocessing changes model performance. For the preprocessed data, the training time was checked using the `time` module to see how fast the model was trained. After training, the model was used to make predictions on the test set (`y_pred_p`).

A Random Forest model was also trained on the raw data, and its training time was noted. Predictions on the raw test set (`y_pred_r`) were also made. This comparison was done to understand how preprocessing steps like scaling, encoding, and PCA affect training speed and prediction accuracy.

### 2.1.9 Results Comparison

The training times and results of the Random Forest models were compared for preprocessed data and raw data. The training time for the preprocessed data was 65.06 seconds, but the raw data finished training in only 12.82 seconds. This showed that preprocessing made the training take more time.

The raw data gave better results for accuracy (90.05% compared to 83.23%), precision (90.45% compared to 83.41%), and recall (90.08% compared to 83.26%). It was clear that preprocessing did not help in this case and even made the accuracy worse by -6.82% and slowed the training by -52.24 seconds.

*Table 1: Comparison of Model Performance: Preprocessed Data vs Raw Data*

| | Preprocessed Data | Raw Data |
|---|---|---|
| **Training Time** | 65.06 seconds | 12.82 seconds |
| **Accuracy** | 83.23% | 90.05% |
| **Precision** | 83.41% | 90.45% |
| **Recall** | 83.26% | 90.08% |

The lower accuracy and longer training time for the preprocessed data were explained by the preprocessing methods used. PCA, which kept 95% of the variance, was used to reduce dimensions. But this changed the original features into principal components. While redundancy was reduced, important connections between features and the target variable might have been lost. Because of this, the Random Forest model, which works well with raw features, found it harder to learn patterns from the transformed data. Also, steps like scaling and encoding made the training take more time because of extra calculations.

The raw data, on the other hand, might have already been well-organized and useful. The Random Forest model handled the raw features without needing scaling or transformations, which helped it perform better. Preprocessing steps may have removed some of the clear connections in the raw data, making accuracy lower. This showed how important it was to choose preprocessing methods that fit the dataset and model needs.

## 2.2: Comparative Analysis of Classification Techniques

### 2.2.1 Import Additional Libraries

In this step, additional libraries were imported to use and check classification methods. The `SVC` from `sklearn.svm` was used for Support Vector Machine (SVM) modeling, and the `MLPClassifier` from `sklearn.neural_network` was added to make and train a Multilayer Perceptron (MLP) model. The `f1_score` from `sklearn.metrics` was included to calculate the F1 Score, which is an important number for checking how good the models worked.

Also, the `warnings` module was added to handle any runtime warnings, and the `time` module was used to measure how long different models took for training and checking. These imports allowed a full study of the classification methods.

### 2.2.2 Prepare Models

In this step, three classification models were prepared for comparing. A `RandomForestClassifier` with 100 trees and a fixed random state was used to check how well an ensemble method worked. The `SVC` (Support Vector Classifier) was added to test a kernel-based method for classification tasks.

An `MLPClassifier` (Multilayer Perceptron) was also set up with a maximum limit of 300 iterations and a fixed random state, showing a neural network-based way. These models were picked because they have different strengths for working with complex datasets, so a good comparison of their performance was made.

### 2.2.3 Train and Evaluate Models

The three classification models—Random Forest, Support Vector Machine (SVM), and Multilayer Perceptron (MLP)—were trained and tested on the preprocessed data. The `fit` method was used to train each model with the training data, and the `predict` method was used to get predictions on the test data. The performance of each model was checked using Accuracy, Precision, Recall, and F1 Score. Also, the time for training each model was recorded using the `time` module to check how fast each model worked.

The results showed the trade-offs between model performance and training time for the three models—Random Forest, SVM, and MLP. The Multilayer Perceptron (MLP) model gave the best performance, with the highest accuracy of 88.46%, precision of 88.55%, recall of 88.48%, and an F1 score of 88.43%. But this model took the most time to train, 163.36 seconds, because neural networks are more complex to compute.

The SVM model had a good balance between training time and performance. It got 85.31% accuracy, 85.45% precision, 85.32% recall, and an F1 score of 85.29%. It was much faster to train than MLP, taking only 35.13 seconds, which made it a better choice for saving time but still giving good results.

The Random Forest model was the fastest to train compared to MLP but took more time than SVM. Its training time was 64.46 seconds. The performance was a bit lower, with an accuracy of 83.23%, precision of 83.41%, recall of 83.26%, and an F1 score of 83.26%. These results showed the trade-offs: MLP gave the best predictions, and Random Forest trained the fastest, but with slightly less accuracy.

*Table 2: Comparison of Model Performance Metrics and Training Time*

| | Random Forest | SVM | MLP |
|---|---|---|---|
| **Accuracy** | 83.23% | 85.31% | 88.46% |
| **Precision** | 83.41% | 85.45% | 88.55% |
| **Recall** | 83.26% | 85.32% | 88.48% |
| **F1 Score** | 83.26% | 85.29% | 88.43% |
| **Training Time** | 64.46 Seconds | 35.13 Seconds | 163.36 Seconds |

### 2.2.4 Concluding the Best Model

To find the model with the best balance between accuracy and training time, a weighted scoring method was used. The balance score was calculated by combining normalized accuracy and training time. A weight of 70% was given to accuracy, and 30% to training time. This way, both prediction and efficiency were considered.

**Equations Used:**

1. **Normalization of Accuracy:**

Accuracy normalized=Model Accuracy / Max Accuracy

This equation made the accuracy of each model relative to the highest accuracy. The result was a number between 0 and 1.

2. **Normalization of Training Time:**

Time normalized=Min Training Tim / Model Training Time

This formula gave higher values to models with shorter training times, as shorter is better.

3. **Balance Score:**

Balance score= (0.7×Accuracy normalized) + (0.3×Time normalized)

More importance (70%) was given to accuracy, but training time (30%) was also counted. This helped pick the model with good performance and less training time.

*Table 3: Comparison of Balance Scores for Random Forest, SVM, and MLP Models*

| | Random Forest | SVM | MLP |
|---|---|---|---|
| **Balance Score** | 82.21% | 97.51% | 76.45% |

The results showed that the Support Vector Machine (SVM) model gave the best balance between accuracy and training time. A scoring system was used, with 70% weight for accuracy and 30% for training time. The SVM got the highest balance score of 97.51%, better than Random Forest (82.21%) and Multilayer Perceptron (MLP) (76.45%). This means SVM gave good accuracy and took less time to train. Because of this, **SVM was recommended as the most efficient model for this dataset, giving both good performance and saving time.**

## 3. Conclusion

This study showed how data cleaning and feature preparation helped improve machine learning models for predicting diabetes. Steps like filling missing values, fixing outliers, scaling data, and reducing dimensions with PCA made the data ready for analysis. Support Vector Machine (SVM) was chosen as the best model because it balanced accuracy and training time well, with a balance score of 97.51%. The Multilayer Perceptron (MLP) had the highest accuracy (88.46%) and F1 score (88.43%), but it needed the longest training time (163.36 seconds). The Random Forest (RF) model trained faster than MLP but had lower accuracy at 83.23%.

These results showed that preprocessing and picking the right model can help improve both performance and time efficiency. SVM gave strong accuracy and shorter training time, making it a good choice for healthcare predictions. Future work could focus on improving MLP by changing preprocessing steps and tuning settings to reduce training time. Also, studying how features interact and using bigger datasets could help improve accuracy and make the models more useful in real healthcare cases.