WWW.IJRPR.COM

**International Journal of Research Publication and Reviews**

Journal homepage: www.ijrpr.com ISSN 2582-7421

# Pre-Processing Techniques for Preparing Clean and High-Quality Data for Diabetes Prediction

***Dr. A. Antony Prakash*[1]**

*[1] Department of Information Technology, St. Joseph's college, Trichy, Tamilnadu, India
aantonyprakash@gmail.com[1]
DOI: https://doi.org/10.55248/gengpi.5.0224.0412

**ABSTRACT**

With the rapid development of data science and the increasing availability of operational data for building construction, there are many opportunities for data-driven solutions to intelligent management of building operations. Data pre-processing is an important step in the analysis of building operational data, considering the complexity of building operations and data quality constraints. Data pre-processing includes a variety of methods to improve the quality of raw data, such as removing outliers and imputing missing values. In this article, we will take a deep dive into data pre-processing techniques used in the analysis of large-scale building operational data. The following techniques are discussed: Missing Value Imputation Outlier Detection Data Reduction Data Scaling Data Transformation Data Partitioning Each of these techniques serves a specific purpose during the pre-processing stage. The research results predict whether or not a patient has diabetes based on the features and determine which method is appropriate for analysing the data.

**KEYWORDS:** Data Preprocessing; Data Cleaning; Data Transformation; Data Integration; Data Optimization;

## I. INTRODUCTION

The heart of this learning process is "Data Preprocessing" as it acts as an integral part of learning (machine learning). Data pre-processing is the most important phase of any machine learning model because the ability of any model to learn efficiently and perform effectively highly and directly depends upon the quality of data [1].

The diversification and rapid increase of machine learning applications across various technology fields have led to their widespread use in solving real-life problems. The effectiveness of machine learning models is heavily reliant on the quality, quantity, and diversity of data used. To ensure the reliability of the algorithm, it is crucial to carefully select the target data from the original dataset. This data can be obtained in the form of symbolic and numeric attributes, ranging from human input to sensor data, with varying degrees of complexity and trustworthiness [2].

The success of a Machine Learning Algorithm is influenced by a wide range of factors, with the representation and quality of the dataset being key considerations. In particular, when the dataset includes redundancy, noise, or unreliable data, it poses challenges for the algorithm to effectively uncover information and deliver enhanced performance [3].

An extensive volume of data is accessible within the realm of medical science. The data acquired may not be in an appropriate format for data analysis; therefore, meticulous preprocessing of raw data is imperative to ensure accurate disease diagnosis. Data preprocessing constitutes a crucial stage in data mining, encompassing data transformation, imputation, outlier removal, normalization, feature selection, and dimensionality reduction [4].

It is crucial to carefully choose the target data, which should be extracted from the original data set to ensure reliability. Once the target data is generated, data preprocessing is necessary to prepare it for analysis. The prepared data can then be used for data analysis and knowledge generation through the application of mining techniques. The five activities involved in data preprocessing are Data Cleaning, Data Optimization, Data Transformation, Data Integration, and Data Conversion [5].

The quality of operational data in buildings is generally low, necessitating data preprocessing to ensure the dependability of data analysis through diverse techniques. This task is widely recognized as a challenging aspect of data analysis and can consume up to 80% of the overall data mining effort [6].

Data pre-processing is a crucial stage within the data mining process, encompassing the essential tasks of data cleaning, transforming, and integrating. Its purpose is to prepare the data for analysis by enhancing its quality and ensuring its suitability for the particular data mining task at hand [7].

In order to address the challenges in building management tasks, advanced methods are often required in addition to conventional data preprocessing techniques. For instance, when dealing with the fault detection and diagnosis task, it is crucial to have a sufficiently large and properly labelled training data set to ensure optimal model performance. However, in practice, individual buildings may encounter data shortage issues, as they may not have

enough data due to limited data accumulation time and the absence of automated data collection systems. Moreover, the availability of labelled data can be scarce, considering the time and costs associated with manual labelling. In such cases, it may not be feasible to utilize advanced classification algorithms due to the problems of overfitting and non-convergence [8].

Data pre-processing in Machine Learning is an essential process that plays a vital role in improving the data's quality, thereby facilitating the extraction of valuable insights. It involves the preparation of raw data, including cleaning and organizing, to ensure its suitability for constructing and training Machine Learning models. In essence, data pre-processing in Machine Learning can be described as a data mining technique that converts raw data into a comprehensible and coherent format [9].

Data normalization involves converting variables with varying ranges into a uniform range. This process is carried out to bring the data points closer to each other. There are several techniques available for data normalization, including Min-max, Z-score, and decimal scaling [10].

Imbalanced data pertains to classification issues wherein the output variables or classes are not evenly distributed. One class significantly outnumbers the other, particularly in binary classification. This occurrence poses a challenge to the classifier as it may lead to overfitting. Therefore, it is crucial to balance the datasets before utilizing them [11].

This paper presents a comprehensive evaluation of data preprocessing techniques utilized in the development of operational data analysis. Its objective is to offer a comprehensive overview of data preprocessing methods for data-driven building energy management. The subsequent sections of the paper are structured as follows. The General Framework for Building Operational Data Preprocessing introduces a general framework for data preprocessing in the context of building operational data analysis. Subsequently, Data Cleaning Methods for Building Operational Data Analysis, Data Reduction, Data Scaling, Data Transformation, and Data Partitioning elucidate representative techniques for various data preprocessing tasks.

## II. RELATED WORK

Data preprocessing in Machine Learning is an essential process that plays a vital role in improving the data's quality, thereby facilitating the extraction of valuable insights. It involves the preparation of raw data, including cleaning and organizing, to ensure its suitability for constructing and training Machine Learning models. In essence, data preprocessing in Machine Learning can be described as a data mining technique that converts raw data into a comprehensible and coherent format.

**Data Cleaning:** Data cleaning is the process of identifying and rectifying errors or inconsistencies in the data, including missing values, outliers, and duplicates. Several techniques, such as imputation, removal, and transformation, can be employed to accomplish this task.

**Data Integration:** The integration of data is a crucial task in data preprocessing for machine learning. This involves merging information from various sources to create a unified dataset. The complexity of handling data in different forms, formats, and semantics poses a significant challenge for many developers in the field of ML.

**Data Transformation:** ML programmers need to be meticulous in their approach towards data preprocessing, particularly when it comes to data transformation. This involves formatting the data in a way that facilitates analysis. Common data transformation techniques include normalization, standardization, and discretization. Standardization adjusts the data to have a mean of zero and a variance of one, while normalization scales the data to a uniform range. Discretization is used to categorize continuous data into discrete groups.

**Data Reduction:** Data reduction involves reducing the size of a dataset while preserving important information. This can be achieved through the utilization of feature selection and feature extraction algorithms. Feature extraction involves transforming the data into a lower-dimensional space while retaining the essential information, whereas feature selection involves selecting a subset of relevant characteristics from the dataset.

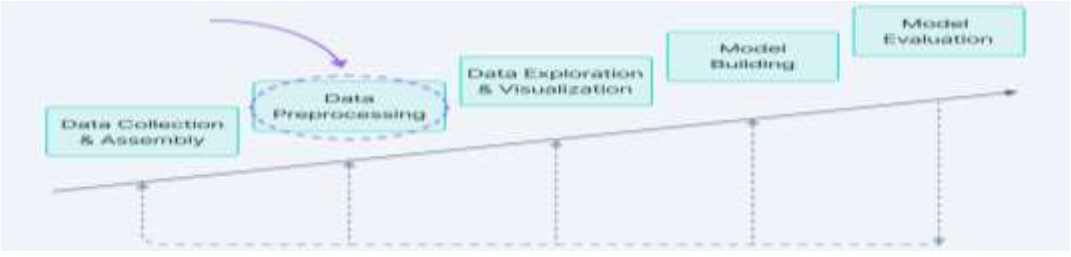



**Figure: 1 Data pre-processing**

## III. IMPORT ALL THE CRUCIAL LIBRARIES

Python is widely used and highly favoured by Data Scientists worldwide. In this tutorial, we will demonstrate the process of importing Python libraries for data preprocessing in Machine Learning. For further information on Python libraries for Data Science, please refer to the provided link. These predefined Python libraries are capable of executing various data preprocessing tasks. Importing the essential libraries constitutes the second step in the data preprocessing phase of machine learning. The three fundamental Python libraries employed for this purpose are:

**NumPy –** NumPy serves as the essential package for conducting scientific computations within the Python programming language. Consequently, it facilitates the inclusion of diverse mathematical operations within the code. Moreover, NumPy enables the incorporation of extensive multidimensional arrays and matrices into your code.

**Pandas** – Pandas, a remarkable Python library for data manipulation and analysis, is widely utilized for importing and managing datasets. This open-source library encompasses efficient data structures and user-friendly tools for data analysis in Python.

**Matplotlib** – Matplotlib is a versatile Python library for creating 2D plots of various types in Python. It is capable of producing high-quality figures in a variety of hard copy formats and interactive environments on multiple platforms, including IPython shells, Jupyter notebooks, and web application servers.

## IV. IMPORT THE DATASET

To import the dataset/s for the machine learning project at hand, it is crucial to follow this step. Data preprocessing in machine learning involves importing the dataset, which is an essential process. However, before proceeding with the dataset import, it is necessary to designate the current directory as the working directory. This can be easily accomplished in the Spyder IDE by following three straightforward steps.

1. Save your Python file in the directory containing the dataset.
2. Go to File Explorer option in Spyder IDE and choose the required directory.
3. Now, click on the F5 button or Run option to execute the file.

## V. DATA PRE-PROCESSING

Data preprocessing refers to the procedure of cleaning and transforming data, regardless of its format, whether it is structured, unstructured, or semi-structured. This data is obtained or derived from various sources, such as historical data, stream data, and application data. Pre-processing involves applying transformations to the data prior to inputting it into the algorithm. This technique, known as data preprocessing, is utilized to convert raw data into a refined data set. Essentially, when data is collected from various sources, it is obtained in a raw format that is unsuitable for analysis.


**Need of Data Preprocessing**

To optimize the performance of Machine Learning models, it is crucial to ensure that the data is properly formatted. Certain models require specific data formats, such as the Random Forest algorithm which cannot handle null values. Therefore, it is necessary to manage null values in the original raw data set before executing the algorithm. Additionally, the data set should be formatted in a manner that allows for the execution of multiple Machine Learning and Deep Learning algorithms, with the best outcome being selected.

**Step 1: Import the necessary libraries**

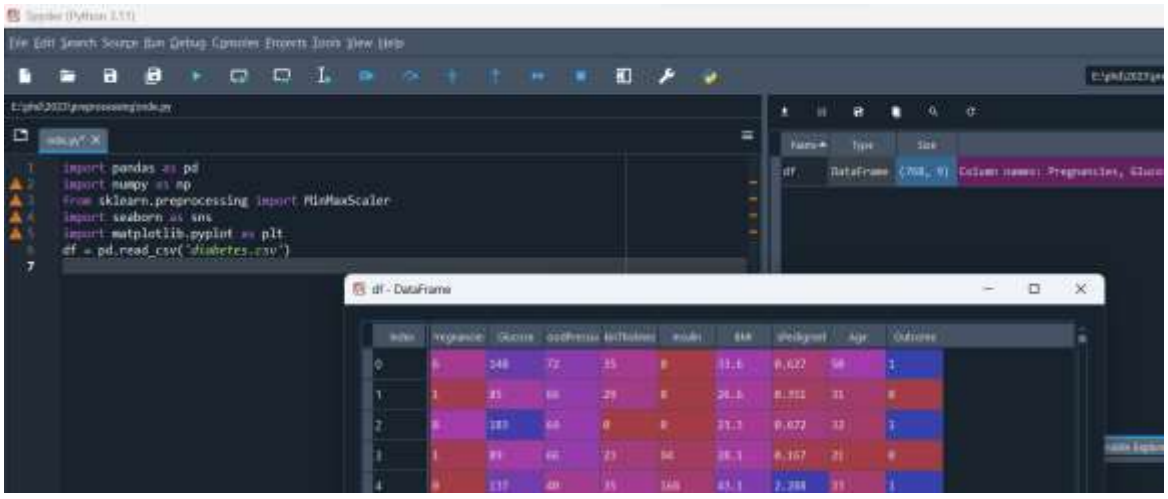



**Figure 2 Import pima-indian-diabetes Dataset from kaggle**

As we can see from the above info that our dataset has 9 columns and each columns has 768 values. There is no Null values in the dataset. We can also check the null values using df.isnull()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

**Figure 3 Notnull value**

**Step 3: Statistical Analysis**

In statistical analysis, first, we use the df.describe() which will give a descriptive overview of the dataset.

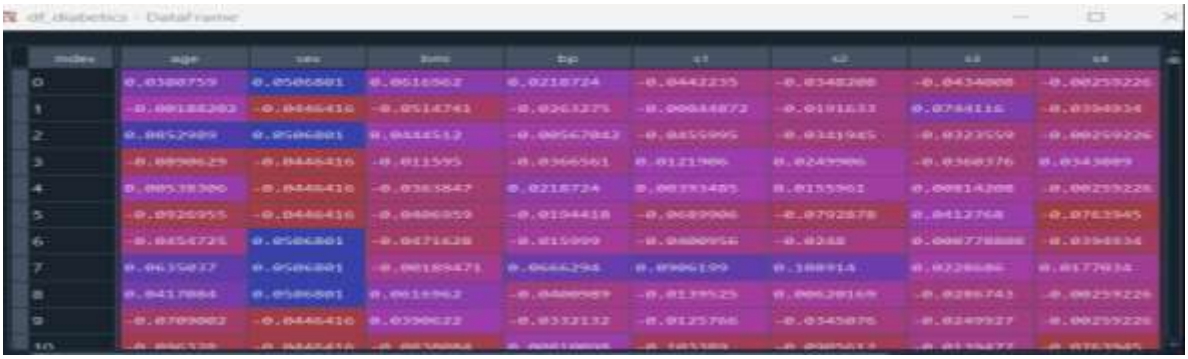
| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
|---|---|---|---|---|---|---|---|---|---|
| count | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 |
| mean | 3.845052 | 120.894531 | 69.105469 | 20.536458 | 79.799479 | 31.992578 | 0.471876 | 33.240885 | 0.348958 |
| std | 3.369578 | 31.972618 | 19.355807 | 15.952218 | 115.244002 | 7.884160 | 0.331329 | 11.760232 | 0.476951 |
| min | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.078000 | 21.000000 | 0.000000 |
| 25% | 1.000000 | 99.000000 | 62.000000 | 0.000000 | 0.000000 | 27.300000 | 0.243750 | 24.000000 | 0.000000 |
| 50% | 3.000000 | 117.000000 | 72.000000 | 23.000000 | 30.500000 | 32.000000 | 0.372500 | 29.000000 | 0.000000 |
| 75% | 6.000000 | 140.250000 | 80.000000 | 32.000000 | 127.250000 | 36.600000 | 0.626250 | 41.000000 | 1.000000 |
| max | 17.000000 | 199.000000 | 122.000000 | 99.000000 | 846.000000 | 67.100000 | 2.420000 | 81.000000 | 1.000000 |

The above table shows the count, mean, standard deviation, min, 25%, 50%, 75%, and max values for each column. When we carefully observe the table we will find that. Insulin, Pregnancies, BMI, Blood Pressure columns has outliers.

**Step 4: Detect and Remove the Outliers**

An Outlier is a data-item/object that deviates significantly from the rest of the objects considered normal. These deviations can be caused by measurement or execution errors. The process of identifying outliers is known as outlier mining. Various methods exist for detecting outliers, and the removal process is similar to removing a data item from a panda's data frame.

In real-world projects, the use of pandas data frame provides a more realistic approach to detecting outliers during the data analysis phase. However, the same approach can also be applied to lists and series-type objects.



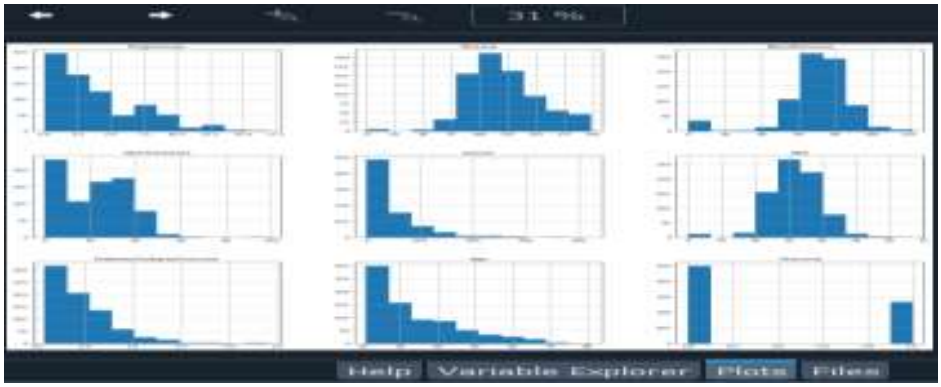
**Step 5: Outliers Visualization**

The data is effectively and efficiently summarized through the utilization of a simple box and whiskers, encapsulating the key information. The boxplot provides a concise summary of the sample data by representing the 25th, 50th, and 75th percentiles. By solely examining the boxplot, one can easily gain insights into the dataset, including quartiles, median, and outliers.


**Step 6**: **Data Visualization**

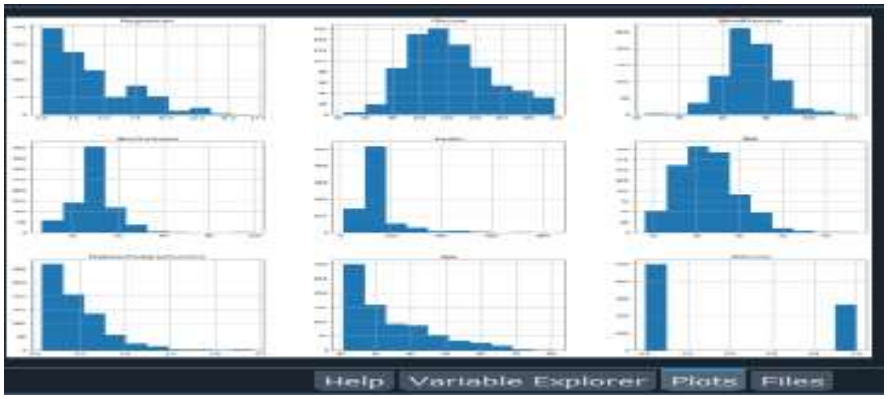
Plotting the data distribution plots before removing null values

p = df.hist(figsize = (20,20))




The distribution of each feature, whether it is dependent or independent data, has been observed. It is worth considering why it is necessary to examine the distribution of data. The answer is straightforward: it is the optimal approach to initiate dataset analysis. By visualizing the occurrence of various values in a graphical format, it provides insights into the data range.

**Plotting the distributions after removing the NAN values.** p = df_copy.hist(figsize = (20,20))




Once again, the hist plot is utilized to observe the dataset's distribution. However, this time, the focus is on examining the alterations that occur after eliminating the null values from the dataset. Notably, in the age column, a noticeable spike within the range of 50 to 100 becomes apparent after the removal of the null values, which aligns with logical expectations.

**Step 7: check that how well our outcome column is balanced**

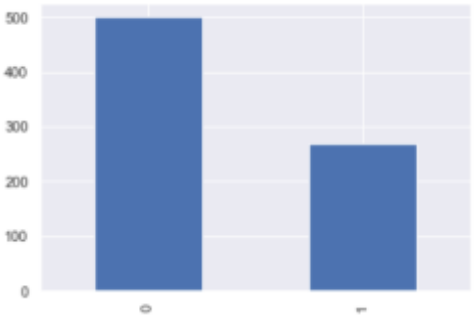
color_wheel = {1: "#0392cf", 2: "#7bc043"}

colors = diabetes_df["Outcome"].map(lambda x: color_wheel.get(x + 1))

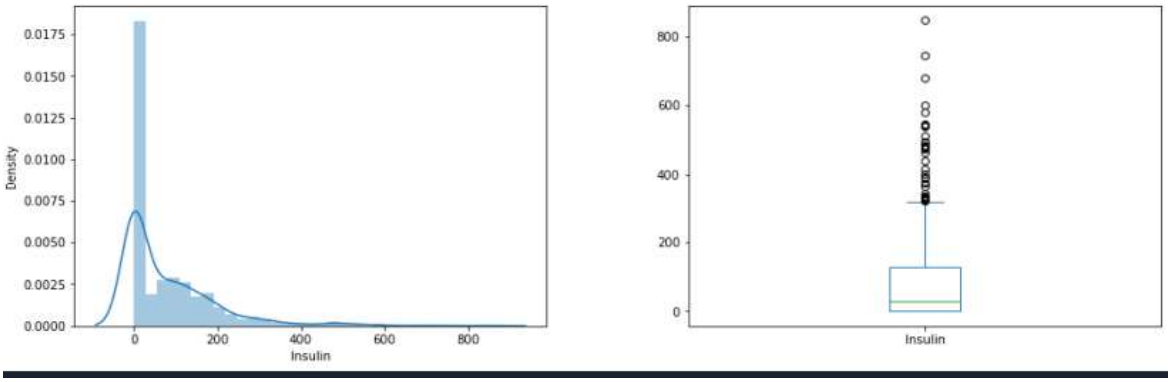
print(diabetes_df.Outcome.value_counts())

p=diabetes_df.Outcome.value_counts().plot(kind="bar")

```
 0   Pregnancies                768 non-null    int64
 1   Glucose                    768 non-null    int64
 2   BloodPressure              768 non-null    int64
 3   SkinThickness              768 non-null    int64
 4   Insulin                    768 non-null    int64
 5   BMI                        768 non-null    float64
 6   DiabetesPedigreeFunction   768 non-null    float64
 7   Age                        768 non-null    int64
 8   Outcome                    768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
Outcome
0    500
1    268
Name: count, dtype: int64
```




Here from the above visualization it is clearly visible that our dataset is completely imbalanced in fact the number of patients who are diabetic is half of the patients who are non-diabetic.




Distplot is a valuable tool for visualizing data distribution. Additionally, by utilizing boxplot, one can easily identify outliers within a specific column and extract other relevant information from the box and whiskers plot.

## VI. RESULT AND FINDINGS

Analysing the dataset it will split the data into training and testing data using the train_test_split function then building the model using RandomForest. After building the model let's check the accuracy of the model on the training dataset. **Accuracy_Score = 1.0**

**Random Forest Model**

So here we can see that on the training dataset our model is over fitted. Getting the accuracy score for **Random Forest Accuracy_Score = 0.7559055**

**Classification report and confusion matrix of random forest model**

```
[[127  35]
 [ 41  51]]
              precision    recall  f1-score   support

           0       0.76      0.78      0.77       162
           1       0.59      0.55      0.57        92

    accuracy                           0.70       254
   macro avg       0.67      0.67      0.67       254
weighted avg       0.70      0.70      0.70       254

Accuracy_Score = 0.7755905511811023
```

**Decision Tree Model**

Now we will be making the predictions on the testing data directly as it is of more importance. Getting the accuracy score for Decision Tree Accuracy Score = 0.71259842

**Classification report and confusion matrix of the decision tree model**

```
[[125  37]
 [ 36  56]]
              precision    recall  f1-score   support

           0       0.78      0.77      0.77       162
           1       0.60      0.61      0.61        92

    accuracy                           0.71       254
   macro avg       0.69      0.69      0.69       254
weighted avg       0.71      0.71      0.71       254
```

**Support Vector Machine (SVM)**

Now we will be making the predictions on the testing data directly as it is of more importance. Getting the accuracy score for Support Vector Machine (SVM) Accuracy Score = 0.748031496

**Classification report and confusion matrix of the SVM classifier**

```
[[124  38]
 [ 38  54]]
              precision    recall  f1-score   support

           0       0.77      0.77      0.77       162
           1       0.59      0.59      0.59        92

    accuracy                           0.70       254
   macro avg       0.68      0.68      0.68       254
weighted avg       0.70      0.70      0.70       254

Accuracy Score = 0.7480314960629921
```

***FINDINGS***

| Model | Random Forest Model | Decision Tree Model | Support Vector Machine (SVM) |
|---|---|---|---|
| **Accuracy level** | 0.7559055 | 0.71259842 | 0.748031496 |

## VII. CONCLUSION

In this paper carried out seven sequence steps of data preprocessing. The processed data after complete analysis shows that the data is noise free, null value or missing values are replaced by any one of strategy like mean, median and mode. They have also handled the categorical variables, separation of independent variable and dependent variable. After utilizing all the patient records, we have successfully constructed a machine learning model, specifically a random forest model, which effectively predicts the presence or absence of diabetes among the patients in the dataset. Additionally, through data analysis and visualization, we have gained valuable insights from the data.

**REFERENCES**

1) W.M.S. Famili , “Data preprocessing and intelligent data analysis Intell”. Data Anal., 1 (1997), pp. 3-23

2) Anbarasi MS. “Outlier detection for multi-dimensional data”. IJCSIT. 2011; 2(1):512–51.

3) R.K. Dash, T.N. Nguyen, K. Cengiz, A. Sharm, “Fine-tuned support vector regression model for stock predictions” Neural Comput. Appl (2021), pp. 1-15.

4) Pachgade SD, Dhande SS. "Outlier detection over dataset using cluster-based and distance-based approach". International Journal of Advanced Research in Computer Science and Software Engineering. 2012; 2(6):1–5

5) Chae, Y. T., Horesh, R., Hwang, Y., and Lee, Y. M. (2016). "Artificial neural network model for forecasting sub-hourly electricity usage in commercial buildings". *Energy Build.* 111, 184–194. doi:10.1016/j.enbuild.2015.11.045

6) Dey, M., Rana, S. P., and Dudley, S. (2018). "Semi-supervised learning techniques for automated fault detection and diagnosis of HVAC systems," in IEEE 30th international conference on tools with artificial intelligence (ICTAI, Volos, Greece, November 5–7, 2018 (New York, NY: IEEE), 872–877.

7) A.I. Kadhim, "An evaluation of preprocessing techniques for text classification", Int. J. Comput. Sci. Inf. Secur. 16 (6) (2018).

8) Goodfellow, I., Bengio, Y., and Courville, "A. Deep learning". 1st Edn. Cambridge, MA: MIT Press. (2016).

9) Agarwal V. Research on Data Preprocessing and Categorization Technique for Smartphone Review Analysis. International Journal of Computer Applications. 2015 Dec; 975:8887.

10) Humphries M. Missing data and how to deal. "An overview of missing data. Population", Research Center; 2017. p. 1–45.

11) Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP. "SMOTE: Synthetic Minority Over-sampling Technique", Journal of Artificial Intelligence Research. 2002; 16:1–37.