DIABETES PREDICTION MODEL REPORT

S. A. L. M. Mihiliya Jayasiri

23113610

Module : **Principles of Data Science (4FTC2110)**

Lecturer : **Ms. Tilani Gunawardena**

Due date : **22nd September 2024**

Table of Contents

1 Introduction

Diabetes is an incurable condition that affects how a body processes blood sugar (glucose). There are three diabetes types: type 1 diabetes, type 2 diabetes and gestational diabetes. In the dataset used for creating the model, attributes are concerned with type 2 diabetes and gestational diabetes. Type 2 diabetes is caused when the body either does not produce enough insulin or when cells become resistant to insulin, and gestational diabetes is caused during pregnancy when the body cannot produce enough insulin to meet the extra needs of the mother and the fetus. Early detection and management of diabetes is crucial for preventing complications. The development of machine learning techniques has become promising in predicting diabetes by analyzing various health related attributes.

Data mining is the process of finding patterns, connections, and valuable information from large datasets using different techniques from machine learning, statistics, and database systems. Techniques like classification, regression, clustering, anomaly detection, etc… is used in the process of data mining. In this process benefits like discovering insights, efficiency, and powerful predictions can be gained. Also, challenges with quality of the data, scalability, privacy concerns can be faced. Data mining can be done with the help of tools like WEKA.

WEKA, also known as Waikato Environment for Knowledge Analysis, is an open-source software suite developed by University of Waikato New Zealand. It offers a variety of machine learning algorithms and data preprocessing tools that are designed for data mining tasks.

This report presents a diabetes prediction model developed using the WEKA tool, an open-source software for data mining and machine learning. The model aims to predict the likelihood of diabetes in individuals based on a set of medical and demographic attributes. The dataset used for this study is the Pima Indians Diabetes dataset, originally from the National Institute of Diabetes and Digestive and Kidney Diseases. It includes medical data from 768 female patients of Pima Indian Heritage who are at least 21 years old.

2 Data Preprocessing

2.1 Loading the Dataset

Method:

Open Weka ⟶ Explorer ⟶ Preprocess ⟶ Open file ⟶ Look In: Windows (C:)

⟶ Program Files ⟶ Weka ⟶ data ⟶ diabetes ⟶ Open

Result:

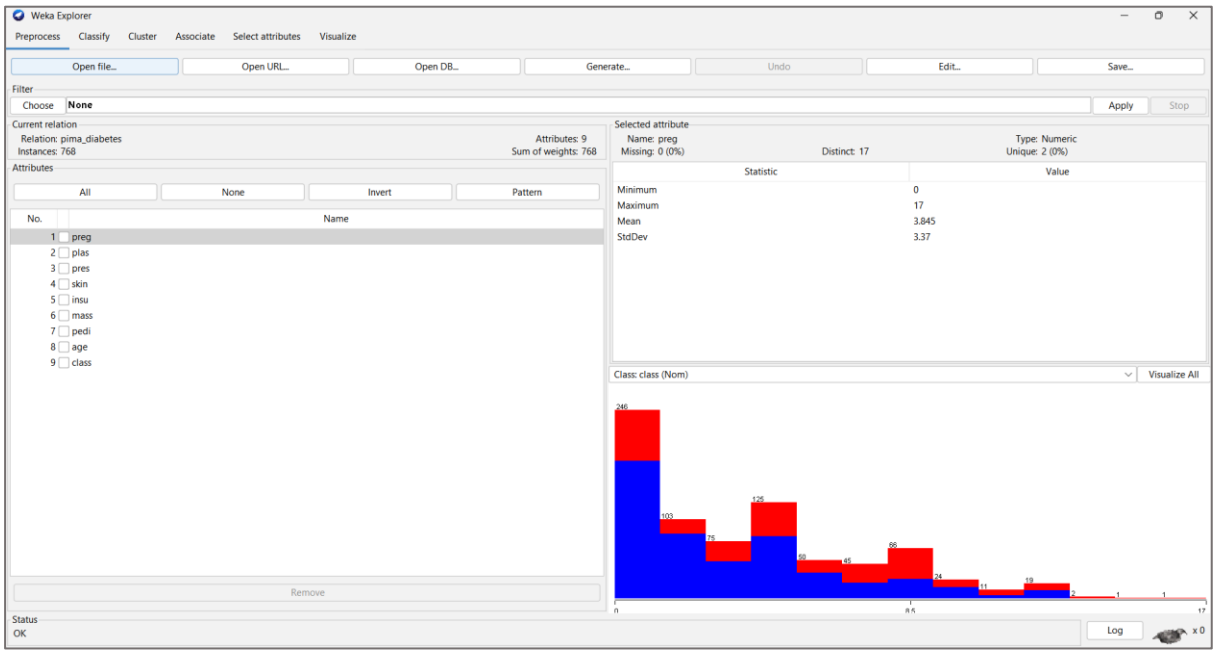

Figure 1: Loaded Diabetes Dataset

Here,

- All the attributes are numeric-valued
- All the attributes are variables
- 1 – 8 attributes are independent variables
- 9th attribute is the dependent variable

2.2 Significance of Attributes in Predicting Diabetes

Attributes in the dataset;

1: preg – Number of times pregnant

2: plas –2-hour Plasma glucose concentration in an oral glucose tolerance test

3: pres – Diastolic blood pressure (mm Hg)

4: skin – Triceps skin fold thickness (mm)

5: insu – 2-hour serum insulin (mu U/ml)

6: mass – Body mass index (weight in kg/ (height in m) ^2)

7: pedi – Diabetes pedigree function

8: age – age (years)

9: class – Class variable (0 or 1)

Significance of each independent attribute;

1: The number of pregnancies is mainly helpful in predicting the risk of a diabetes type known as gestational diabetes mellitus (GDM) – Increased risk of GDM and type 2 diabetes with multiple pregnancies.

2: Plasma glucose measured 2 hours after an oral glucose tolerance test (OGTT) involves measuring blood glucose level after fasting and then again after consuming a glucose-rich drink – The 2-hour plasma level;

- Normal range: below 7.8 mmol/L,
- A pre-diabetic state: between 7.8 mmol/L and 11.1 mmol/L,
- Diabetes: 11.1 mmol/L and above,

It is a strong predictor of future diabetes.

3: Diastolic blood pressure (DBP) – Indicates cardiovascular health, which is a common complication in people with diabetes. High DBP can damage blood vessels, which exacerbates diabetes complications, regular monitoring and management of DBP are essential for diabetes patients to prevent complications.

4: Triceps skin fold thickness – Indicates body fat where higher body fat percentages have an increased risk of developing type 2 diabetes.

5: The 2-hour serum insulin level – Indicates insulin resistance, which is a precursor to type 2 diabetes.

6: BMI – Higher BMI values, which means obesity is linked to an increased risk of type 2 diabetes.

7: Diabetes pedigree function (DPF) – Indicates genetic risks of getting diabetes.

8: Age – The risk of developing diabetes increases with age.

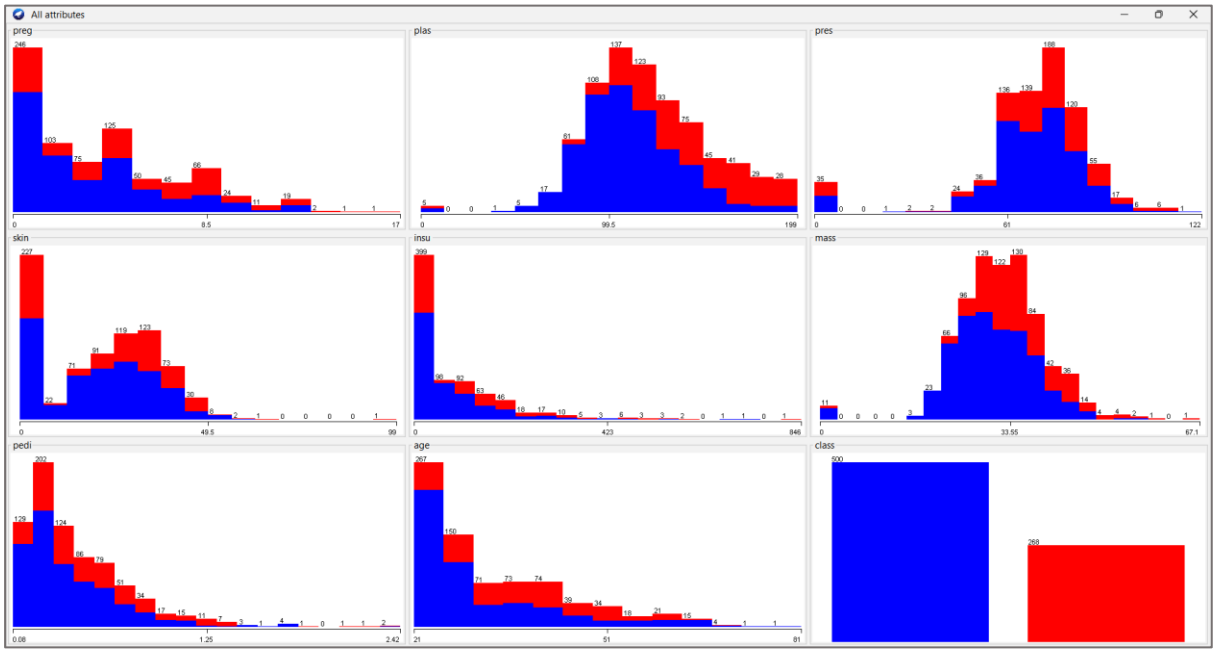

Figure 2: Visualization of the 9 Attributes

2.3 Preparing the Data

In this dataset, each independent variable has different minimum, maximum, mean and standard deviation values, and also all the attributes are numeric which becomes problematic when building a classification model. Therefore, it is necessary to perform data scaling or data categorizing, which are data pre-processing techniques.

Data scaling makes every independent variable to be on a common scale. It improves the performance of machine learning algorithms, which helps in reducing the impact of outliers and also in improving the accuracy of the model.

Data scaling techniques: Normalize, Standardize.

Data categorizing converts continuous numeric attributes into discrete categories or bins. It makes the dataset easier to analyze, especially when dealing with algorithms that perform better with categorical data.

Data categorizing technique: Discretize.

Here, for the datasets, standardization and discretization techniques were used.

2.3.1 Standardization

Standardization transforms the features of a dataset to have a mean of 0 and a standard deviation of 1. This process is also known as Z-score normalization.

Method:

In the Filter section: Choose ⟶ filters ⟶ unsupervised ⟶ attribute ⟶ Standardize ⟶ Apply

Result:

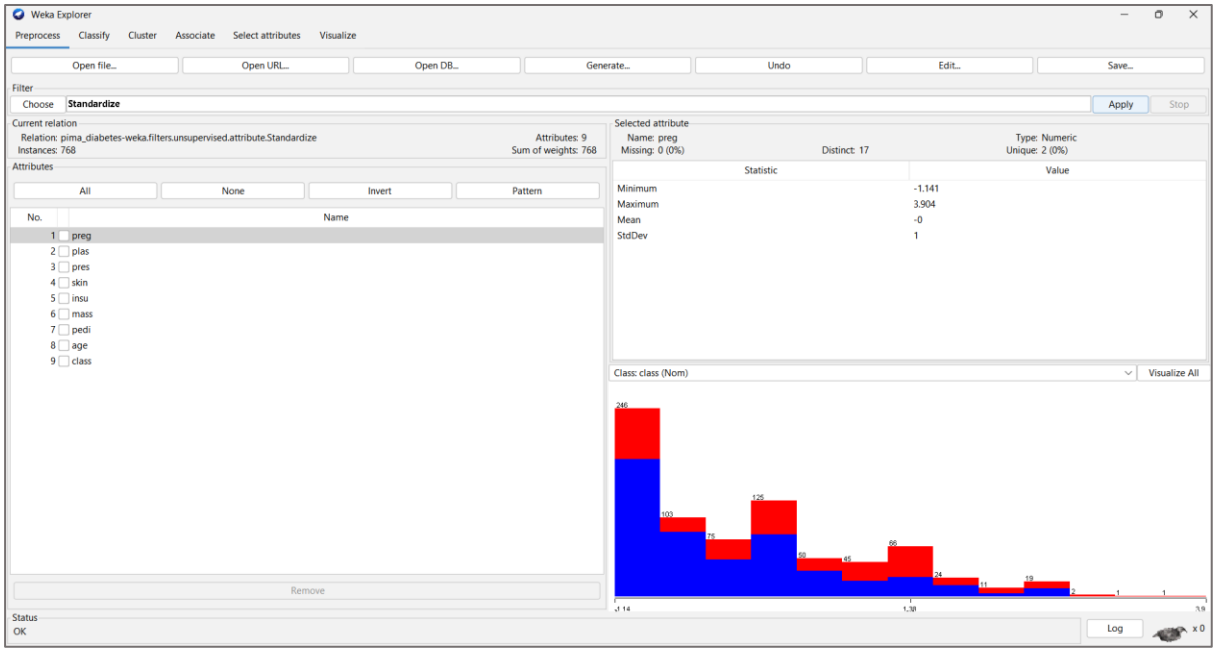

Figure 3: Standardized Dataset

Algorithms trained on the standardized dataset: SMO, LMT

Sequential Minimal Optimization (SMO) – SMO for Support Vector Machines (SVM), where SVMs are sensitive to the scale of input features, can be standardized, which helps the optimization process converge more efficiently.

Logistic Model Tree (LMT) – LMT combines decision trees with logistic regression. It benefits from standardized data because it assumes that the input features are on a similar scale at the leaves.

2.3.2 Discretization

Discretization converts continuous numeric attributes into discrete nominal attributes or bins. This process is also known as binning.

Method:

In the Filter section: Choose ⟶ filters ⟶ unsupervised ⟶ attribute ⟶ Discretize ⟶ Apply

Result:

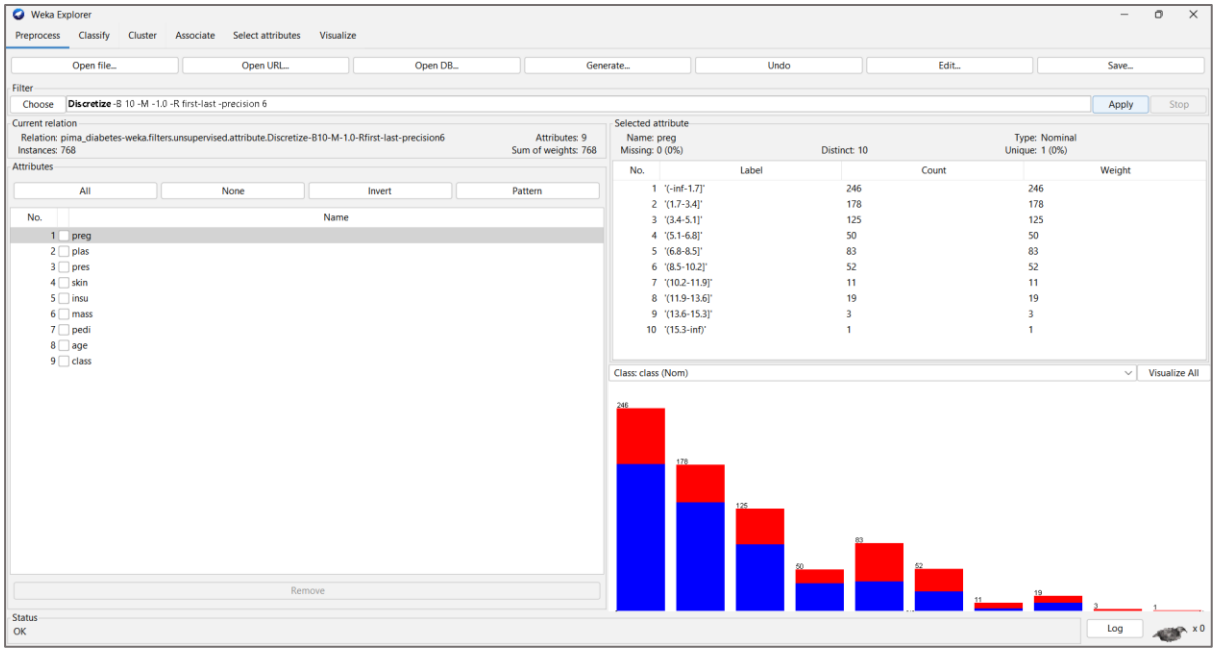

Figure 4: Discretized Dataset

Algorithms trained on the discretized dataset: Naïve Bayes, Random Forest, J48

Naïve Bayes – Naïve Bayes assumes that the features are conditionally independent of the class attribute. When dealing with continuous attributes, estimating the probability density function is complex. Discretizing makes it easier to estimate probabilities.

Random Forest, J48 – Divides features into distinct categories, making it simpler to understand the splits and rules of the derived from the trees. Discretizing data makes the resulting decision tree easier to interpret.

2.4 Splitting the Data

Normally, a dataset is getting split into two parts as training dataset and testing dataset where training dataset is used to create the model and testing dataset is used to test the accuracy and performance of the created model.

Here, since a prediction model is being created it is necessary to split the data.

- Training dataset – 80% of the full dataset
- Testing dataset – remaining 20% of the full dataset

2.4.1 Randomization

Before splitting the data, it is better to perform randomization so, that all the data will be randomly shuffled. Randomizing the data helps in reducing bias, improving generalization, and also it ensures fair evaluation.

Method:

In Filter section: Choose ⟶ filters ⟶ unsupervised ⟶ instance ⟶ Randomize ⟶ Apply

Result:

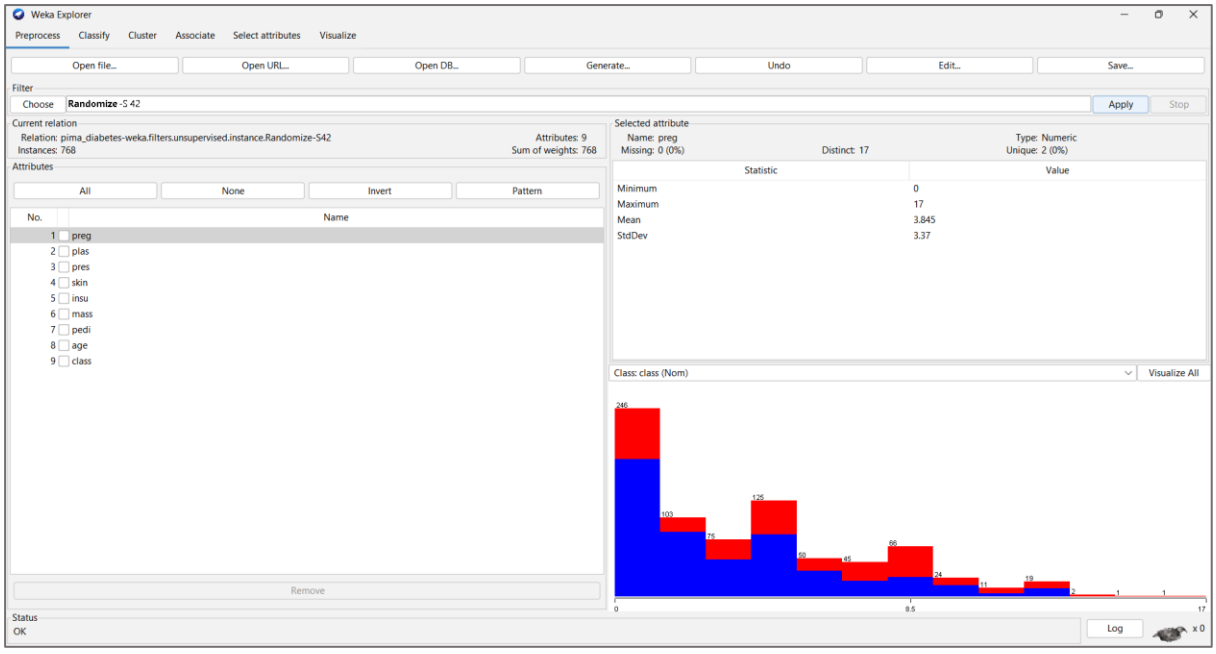

Figure 5: Randomized Dataset

2.4.2 Training Dataset

Method:

In Filter section: Choose ⟶ filters ⟶ unsupervised ⟶ instance ⟶

RemovePercentage ⟶ Click on (RemovePercentage -P 50.0) ⟶

Change the percentage to 20.0 ⟶ OK ⟶ Apply ⟶ Save ⟶

File Name: diabetes_training.arff ⟶ Save

Result:

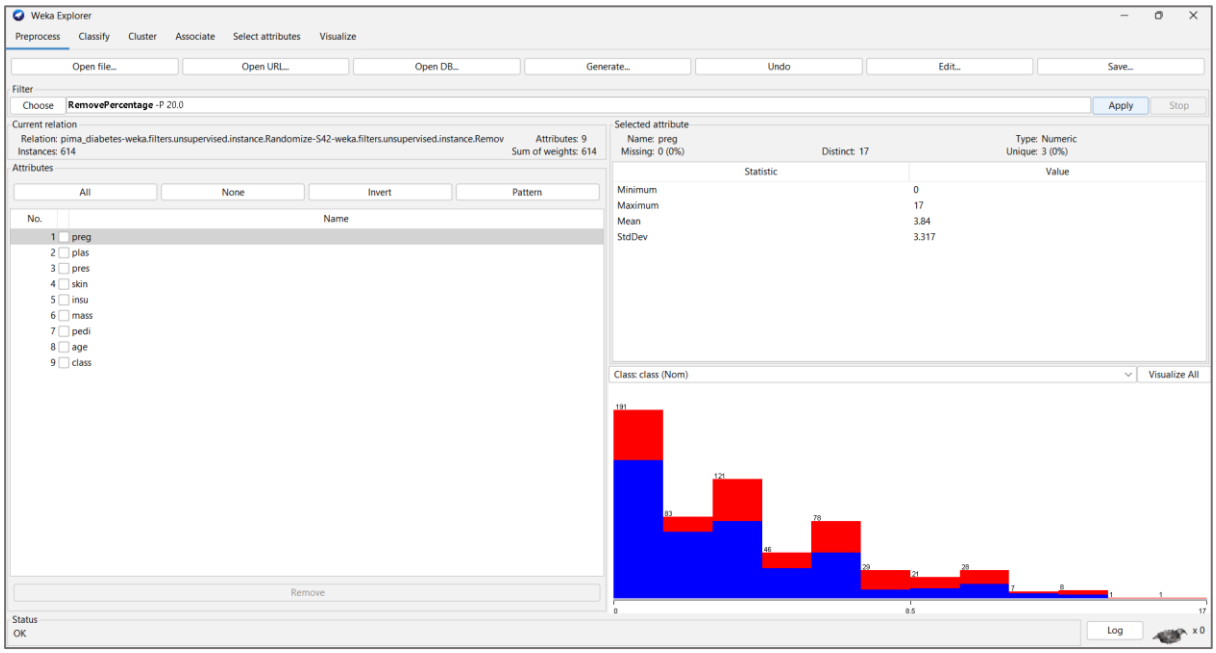

Figure 6: Training Dataset

2.4.3 Testing Dataset

Method:

After creating the training dataset and saving, click the **Undo** button to restore the full dataset. Then,

Click on (RemovePercentage -P 20.0) ⟶ invertSelection - True ⟶ OK ⟶ Apply Save ⟶ File Name: diabetes_testing.arff ⟶ Save

Result:

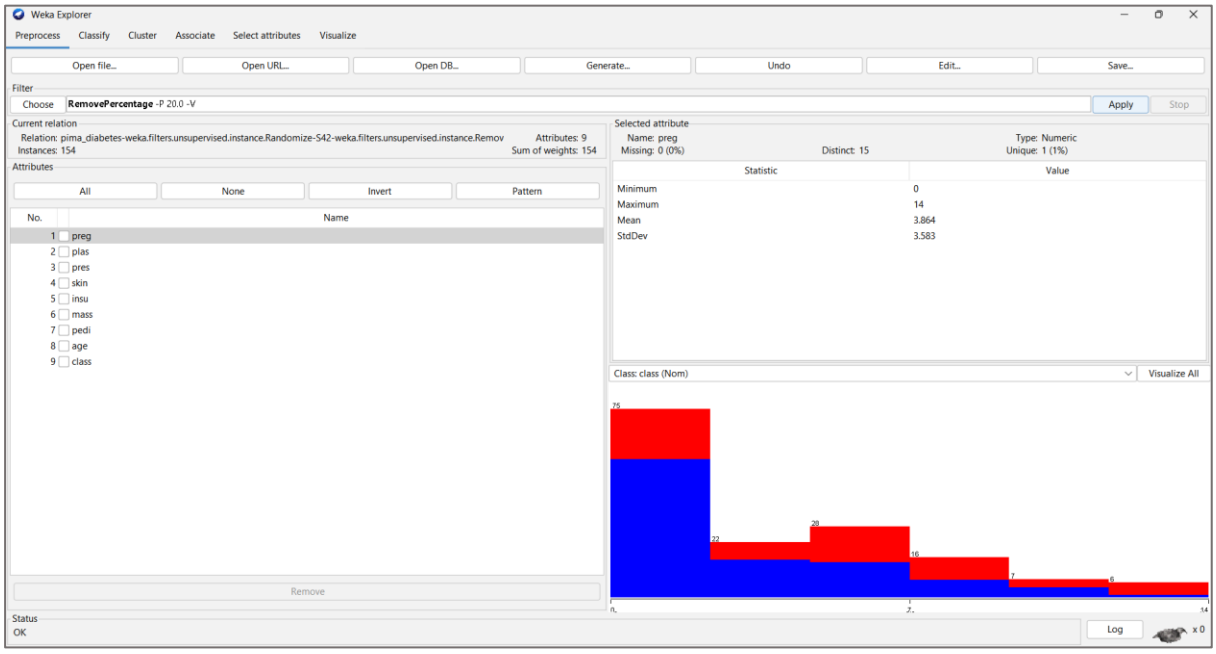

Figure 7: Testing Dataset

3 Types of Data Mining Algorithms Used

There are many algorithms in Weka but out of them below shown five algorithms was chosen as suitable algorithms to decide the best performing model for this dataset.

1) Naïve Bayes

   Naïve Bayes is a probabilistic classifier which is based on Bayes' theorem. This algorithm assumes that there is independence between predictors and the assumed independence assumptions may not have an impact on reality. This is why they are considered as naive. This model is also useful for very large datasets [1].

2) SMO

   Implements John Platt's sequential minimal optimization algorithm for training a support vector classifier. This implementation globally replaces all missing values and transforms nominal attributes into binary ones. It also normalizes all attributes by default. (In that case the coefficients in the output are based on the normalized data, not the original data --- this is important for interpreting the classifier). Multi-class problems are solved using pairwise classification (aka 1-vs-1). To obtain proper probability estimates, use the option that fits calibration models to the outputs of the support vector machine. In the multi-class case, the predicted probabilities are coupled using Hastie and Tibshirani's pairwise coupling method. [2][3][4].

3) Random Forest

   The random forest classifier is defined as combination of tree classifiers. Aim of random forests algorithm is increasing the classification value. The algorithm achieves this aim by generating more than one decision tree while performing the classification process. Decision trees which are created individually come together to form a decision forest. Each classifier is generated with a random vector sampled independently from input vector [5]. In this algorithm, newly created training sets are built with replacement from the original ones. The tree is created by using a random attribute selection and a new subset. The best split on the random attributes selected is used to split the node [6].

4) LMT

Random Tree is a supervised classifier and fast decision tree learner. The algorithm uses bagging idea to create a random set of data to build a decision tree. The algorithm prunes the tree by using reduced-error pruning with back fitting. It can handle both binary and multi-class target variables [7].

5) Decision Tree (J48)

This statistical classification algorithm is re-implemented version of C4.5 with JAVA for WEKA. J48 chooses the attribute of the data at every node of the tree, which is most effectively splits its arrangement of tests into subsets improved in one class or the other. J48 have some features such as, continuous attribute value ranges, accounting for missing values, decision trees pruning, derivation of rules, etc. This algorithm works for both categorical and continuous dependent variables [8].

4 Building and Evaluating the Models

Models are built and tested in Weka Explorer ⟶ Classify

4.1 Cross-Validation

4.1.1 What is Cross-Validation?

Cross-validation is a technique used to test how well a model will perform on new, unseen data. Also, it evaluates the performance of a model and ensure its ability to generalize to an independent dataset.

4.1.2 Reason to Use Cross-Validation Technique

Since the diabetes dataset is of 786 instances, it is considered as a small dataset. So, cross-validation is the best option for this dataset, because it helps to maximize the use of available data by ensuring that every data point is used for both training and testing. It's beneficial as it reduces overfitting and efficiently uses the data.

4.2 Procedure of Using Cross-Validation Technique

Here, 10-fold cross-validation is used, the dataset is divided into 10 subsets. The model is trained on 9 subsets and tested on the remaining one. This process is repeated 10 times, and the results are averaged to provide a powerful estimate of model performance.

A. Creating the Model

Open the diabetes_training.arff in Preprocess ⟶ Proceed to Classify

Choose the algorithm ⟶ Change the Test Options ⟶ Cross-validation: 10 Folds ⟶

Click on Start and evaluate

B. Testing the Created Model

Change the Test Options ⟶ Supplied test set ⟶ Set ⟶ Open file ⟶

File Name: diabetes_testing.arff ⟶ Open ⟶ Close ⟶ Click on the created model ⟶

Click on Start and evaluate the test set

4.2.1 SMO

A.

Choose the algorithm:

In Classifier section: Choose ⟶ classifier ⟶ functions ⟶ SMO

Result:

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         476               77.5244 %
Incorrectly Classified Instances       138               22.4756 %
Kappa statistic                          0.4693
Mean absolute error                      0.2248
Root mean squared error                  0.4741
Relative absolute error                 49.4745 %
Root relative squared error             99.4903 %
Total Number of Instances              614

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.905    0.467    0.784      0.905    0.840      0.483    0.719     0.771     tested_negative
                 0.533    0.095    0.750      0.533    0.623      0.483    0.719     0.562     tested_positive
Weighted Avg.    0.775    0.338    0.772      0.775    0.764      0.483    0.719     0.698

=== Confusion Matrix ===

   a   b   <-- classified as
 362  38 |   a = tested_negative
 100 114 |   b = tested_positive
```

Figure 8: Training Set Evaluation in SMO Algorithm

B.

Result:

```
=== Evaluation on test set ===

Time taken to test model on supplied test set: 0 seconds

=== Summary ===

Correctly Classified Instances         118               76.6234 %
Incorrectly Classified Instances        36               23.3766 %
Kappa statistic                          0.4491
Mean absolute error                      0.2338
Root mean squared error                  0.4835
Relative absolute error                 51.3881 %
Root relative squared error            101.324  %
Total Number of Instances              154

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.900    0.481    0.776      0.900    0.833      0.463    0.709     0.763     tested_negative
                 0.519    0.100    0.737      0.519    0.609      0.463    0.709     0.551     tested_positive
Weighted Avg.    0.766    0.348    0.762      0.766    0.755      0.463    0.709     0.689

=== Confusion Matrix ===

  a  b   <-- classified as
 90 10 |  a = tested_negative
 26 28 |  b = tested_positive
```

Figure 9: Testing Set Evaluation in SMO Model

4.2.2 LMT

A.

Choose the algorithm:

In Classifier section: Choose ⟶ classifier ⟶ trees ⟶ LMT

Result:

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         476               77.5244 %
Incorrectly Classified Instances       138               22.4756 %
Kappa statistic                          0.4754
Mean absolute error                      0.3095
Root mean squared error                  0.3929
Relative absolute error                 68.1282 %
Root relative squared error             82.4459 %
Total Number of Instances              614

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.893    0.444    0.790      0.893    0.838      0.485    0.836     0.885     tested_negative
                 0.556    0.108    0.735      0.556    0.633      0.485    0.836     0.715     tested_positive
Weighted Avg.    0.775    0.327    0.771      0.775    0.767      0.485    0.836     0.825

=== Confusion Matrix ===

   a   b   <-- classified as
 357  43 |   a = tested_negative
  95 119 |   b = tested_positive
```

Figure 10: Training Set Evaluation in LMT Algorithm

B.

Result:

```
=== Evaluation on test set ===

Time taken to test model on supplied test set: 0 seconds

=== Summary ===

Correctly Classified Instances         119               77.2727 %
Incorrectly Classified Instances        35               22.7273 %
Kappa statistic                          0.481
Mean absolute error                      0.314
Root mean squared error                  0.4027
Relative absolute error                 69.02   %
Root relative squared error             84.391  %
Total Number of Instances              154

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.870    0.407    0.798      0.870    0.833      0.485    0.812     0.881     tested_negative
                 0.593    0.130    0.711      0.593    0.646      0.485    0.812     0.731     tested_positive
Weighted Avg.    0.773    0.310    0.768      0.773    0.767      0.485    0.812     0.829

=== Confusion Matrix ===

  a  b   <-- classified as
 87 13 |  a = tested_negative
 22 32 |  b = tested_positive
```

Figure 11: Testing Set Evaluation in LMT Model

4.2.3 Naïve Bayes

A.

Choose the algorithm:

In Classifier section: Choose ⟶ classifier ⟶ bayes ⟶ Naïve Bayes

Result:

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         465               75.7329 %
Incorrectly Classified Instances       149               24.2671 %
Kappa statistic                          0.4615
Mean absolute error                      0.2837
Root mean squared error                  0.4119
Relative absolute error                 62.4482 %
Root relative squared error             86.4497 %
Total Number of Instances              614

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.823    0.364    0.808      0.823    0.815      0.462    0.829     0.906     tested_negative
                 0.636    0.178    0.657      0.636    0.646      0.462    0.829     0.677     tested_positive
Weighted Avg.    0.757    0.299    0.756      0.757    0.756      0.462    0.829     0.826

=== Confusion Matrix ===

   a   b   <-- classified as
 329  71 |   a = tested_negative
  78 136 |   b = tested_positive
```

Figure 12: Training Set Evaluation in Naïve Bayes Algorithm

B.

Result:

```
=== Evaluation on test set ===

Time taken to test model on supplied test set: 0 seconds

=== Summary ===

Correctly Classified Instances         117               75.974  %
Incorrectly Classified Instances        37               24.026  %
Kappa statistic                          0.4746
Mean absolute error                      0.2905
Root mean squared error                  0.4119
Relative absolute error                 63.8604 %
Root relative squared error             86.3141 %
Total Number of Instances              154

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.810    0.333    0.818      0.810    0.814      0.475    0.821     0.894     tested_negative
                 0.667    0.190    0.655      0.667    0.661      0.475    0.821     0.707     tested_positive
Weighted Avg.    0.760    0.283    0.761      0.760    0.760      0.475    0.821     0.828

=== Confusion Matrix ===

  a  b   <-- classified as
 81 19 |  a = tested_negative
 18 36 |  b = tested_positive
```

Figure 13: Testing Set Evaluation in Naïve Bayes Model

4.2.4 Random Forest

A.

Choose the algorithm:

In Classifier section: Choose ⟶ classifier ⟶ trees ⟶ Random Forest

Result:

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         446               72.6384 %
Incorrectly Classified Instances       168               27.3616 %
Kappa statistic                          0.3643
Mean absolute error                      0.3317
Root mean squared error                  0.4182
Relative absolute error                 73.0127 %
Root relative squared error             87.7608 %
Total Number of Instances              614

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.850    0.505    0.759      0.850    0.802      0.370    0.795     0.882     tested_negative
                 0.495    0.150    0.639      0.495    0.558      0.370    0.795     0.637     tested_positive
Weighted Avg.    0.726    0.381    0.717      0.726    0.717      0.370    0.795     0.797

=== Confusion Matrix ===

   a   b   <-- classified as
 340  60 |   a = tested_negative
 108 106 |   b = tested_positive
```

Figure 14: Training Set Evaluation in Random Forest Algorithm

B.

Result:

```
=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.01 seconds

=== Summary ===

Correctly Classified Instances         119               77.2727 %
Incorrectly Classified Instances        35               22.7273 %
Kappa statistic                          0.481
Mean absolute error                      0.3164
Root mean squared error                  0.4082
Relative absolute error                 69.5476 %
Root relative squared error             85.5412 %
Total Number of Instances              154

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.870    0.407    0.798      0.870    0.833      0.485    0.816     0.894     tested_negative
                 0.593    0.130    0.711      0.593    0.646      0.485    0.816     0.676     tested_positive
Weighted Avg.    0.773    0.310    0.768      0.773    0.767      0.485    0.816     0.817

=== Confusion Matrix ===

  a  b   <-- classified as
 87 13 |  a = tested_negative
 22 32 |  b = tested_positive
```

Figure 15: Testing Set Evaluation in Random Forest Model

4.2.5 Decision Tree (J48)

A.

Choose the algorithm:

In Classifier section: Choose ⟶ classifier ⟶ trees ⟶ J48

Result:

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         461               75.0814 %
Incorrectly Classified Instances       153               24.9186 %
Kappa statistic                          0.4123
Mean absolute error                      0.3403
Root mean squared error                  0.4201
Relative absolute error                 74.9006 %
Root relative squared error             88.1549 %
Total Number of Instances              614

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.885    0.500    0.768      0.885    0.822      0.424    0.764     0.839     tested_negative
                 0.500    0.115    0.699      0.500    0.583      0.424    0.764     0.609     tested_positive
Weighted Avg.    0.751    0.366    0.744      0.751    0.739      0.424    0.764     0.758

=== Confusion Matrix ===

   a   b   <-- classified as
 354  46 |   a = tested_negative
 107 107 |   b = tested_positive
```

Figure 16: Training Set Evaluation in J48 Algorithm

B.

Result:

```
=== Evaluation on test set ===

Time taken to test model on supplied test set: 0 seconds

=== Summary ===

Correctly Classified Instances         108               70.1299 %
Incorrectly Classified Instances        46               29.8701 %
Kappa statistic                          0.2961
Mean absolute error                      0.3508
Root mean squared error                  0.4332
Relative absolute error                 77.1112 %
Root relative squared error             90.7858 %
Total Number of Instances              154

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.850    0.574    0.733      0.850    0.787      0.305    0.742     0.809     tested_negative
                 0.426    0.150    0.605      0.426    0.500      0.305    0.742     0.614     tested_positive
Weighted Avg.    0.701    0.425    0.688      0.701    0.686      0.305    0.742     0.740

=== Confusion Matrix ===

  a  b   <-- classified as
 85 15 |  a = tested_negative
 31 23 |  b = tested_positive
```

Figure 17: Testing Set Evaluation in J48 Model

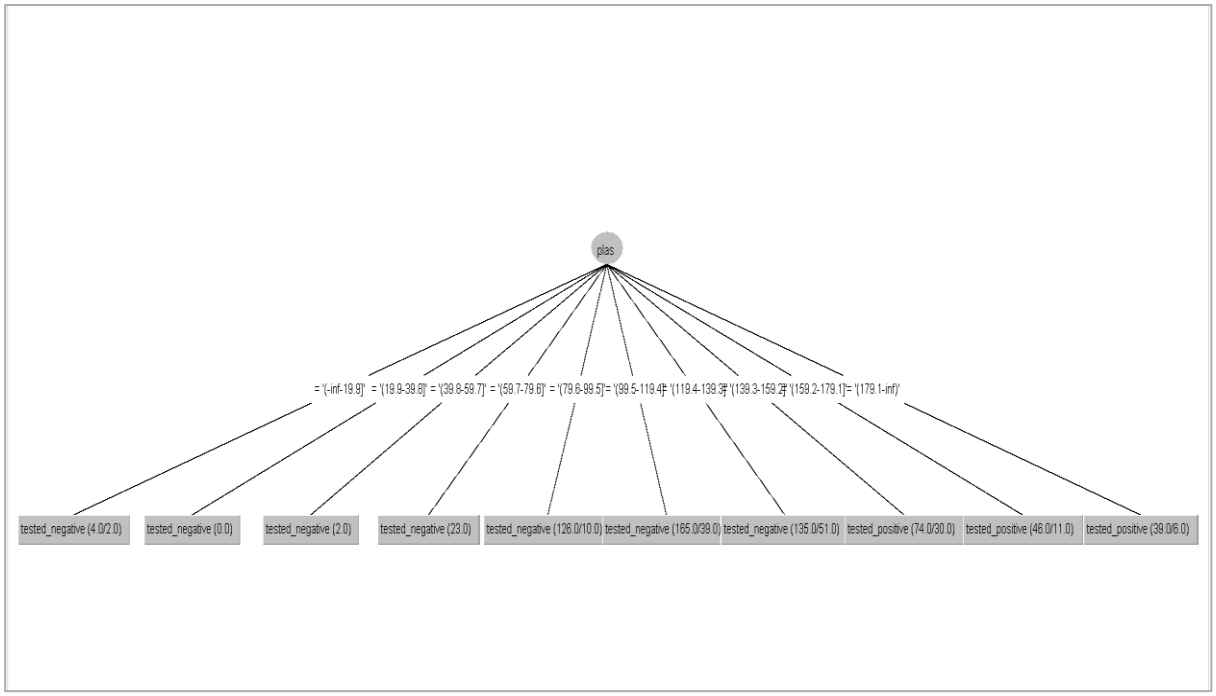

Figure 18: Visualized Decision Tree of J48 Model

5 Results and Findings

5.1 Result of Accurate Data Preprocessing

The performance of the models differs according to the technique that is used to preprocess the data.

Algorithm	Evaluated test dataset of 154 instances (Correctly Classified Instances)							
	No technique		Discretize		Normalize		Standardize	
SMO	115	74.68%	111	72.08%	115	74.68%	118	76.62%
LMT	114	74.03%	114	74.03%	114	74.03%	119	77.27%
Naïve Bayes	114	74.03%	117	75.97%	114	74.03%	116	75.32%
Random Forest	112	72.73%	119	77.27%	113	73.38%	116	75.32%
J48	106	68.83%	108	70.13%	106	68.83%	102	66.23%

Table 1: Data Preprocessing Using Different Techniques

It can be seen that using data preprocessing techniques have a positive effect on the accuracy of the model.

Each model is created and evaluated after preprocessing the data using the technique where the accuracy of the model is highlighted.

Discretization – Naïve Bayes, Random Forest, J48

Standardization – SMO, LMT

5.2 Results of All the Algorithms Modelled Using Cross-Validation Technique

Algorithm	Correctly Classified Instances		Incorrectly Classified Instances		Tot. Instances
SMO	476	77.52%	138	22.48%	614
LMT	476	77.52%	138	22.48%	614
Naïve Bayes	465	75.73%	149	24.27%	614
Random Forest	446	72.34%	168	27.36%	614
J48	461	75.08%	153	24.92%	614

Table 2: Training Model Evaluation Results

Algorithm	Correctly Classified Instances		Incorrectly Classified Instances		Tot. Instances
SMO	118	76.62%	36	23.38%	154
LMT	119	77.27%	35	22.73%	154
Naïve Bayes	117	75.97%	37	24.03%	154
Random Forest	119	77.27%	35	22.73%	154
J48	108	70.13%	46	29.87%	154

Table 3: Testing Model Evaluation Results

5.3 Best Performing Classification Algorithm

Algorithm	Correctly Classified	Incorrectly Classified
SMO	76.62%	23.38%
LMT	77.27%	22.73%
Naïve Bayes	75.97%	24.03%
Random Forest	77.27%	22.73%
J48	70.13%	29.87%

Table 4: Results of Classification Algorithms (Testing Model)

In model evaluation, LMT and Random Forest algorithms performed the highest accuracy with 77.27% while J48 performed the lowest accuracy of 70.13%.

In order to choose the best performing algorithm, error rates are considered.

Algorithm	Mean Absolute Error (MAE)	Root Mean Squared Error (RMSE)	Relative Absolute Error (RAM)	Root Relative Squared Error (RRSE)
Random Forest	0.3164	0.4082	69.55%	85.54%
LMT	0.314	0.4027	69.02%	84.39%

Table 5: Error Rates

Lower MAE and RMSE values suggest that the model's predictions are more accurate.

Lower RAE and RRSE percentages indicate better performance.

Therefore, the chosen algorithm is the **LMT algorithm**.

5.4 Confusion Matrix

Algorithm	True Positive (TP)	False Negative (FN)	False Positive (FP)	True Negative (TN)
SMO	362	38	100	114
LMT	357	43	95	119
Naïve Bayes	329	71	78	136
Random Forest	340	60	108	106
J48	354	46	107	107

Table 6: Confusion Matrix Values for Each Algorithm Training Model

Algorithm	True Positive (TP)	False Negative (FN)	False Positive (FP)	True Negative (TN)
SMO	90	10	26	28
LMT	87	13	22	32
Naïve Bayes	81	19	18	36
Random Forest	87	13	22	32
J48	85	15	31	23

Table 7: Confusion Matrix Values for Each Algorithm Testing Model

Here,

- True Positive (TP) – Instances correctly predicted as the positive class (Tested negative, it's true)
- True Negative (TN) – Instances correctly predicted as the negative class (Tested positive, it's true)
- False Positive (FP) – Instances incorrectly predicted as the positive class (Tested negative, it's false)
- False Negative (FN) – Instances incorrectly predicted as the negative class (Tested positive, it's false)

5.5 Performance Metrics

Algorithm	Positive Class			Negative Class			Accuracy	Error Rate
	Precision	Recall	F1 Score	Precision	Recall	F1 Score		
SMO	0.784	0.905	0.840	0.750	0.533	0.623	0.775	0.225
LMT	0.790	0.893	0.838	0.735	0.556	0.633	0.775	0.225
Naïve Bayes	0.808	0.823	0.815	0.657	0.636	0.646	0.757	0.243
Random Forest	0.759	0.850	0.802	0.639	0.495	0.558	0.726	0.274
J48	0.768	0.885	0.822	0.699	0.500	0.583	0.751	0.249

Table 8: Performance Metrics of Training Model

Algorithm	Positive Class			Negative Class			Accuracy	Error Rate
	Precision	Recall	F1 Score	Precision	Recall	F1 Score		
SMO	0.776	0.900	0.833	0.737	0.519	0.609	0.766	0.234
LMT	0.798	0.870	0.833	0.711	0.593	0.646	0.773	0.227
Naïve Bayes	0.818	0.810	0.814	0.655	0.667	0.661	0.760	0.240
Random Forest	0.798	0.870	0.833	0.711	0.593	0.646	0.773	0.227
J48	0.733	0.850	0.787	0.605	0.426	0.500	0.701	0.299

Table 9: Performance Metrics of Testing Model

6 Discussion

When comparing the training set results and testing test results, two algorithms have the highest accuracy. In the training set results, they are SMO and LMT with an accuracy of 77.52%. In the testing set results, they are LMT and Random Forest with an accuracy of 77.27%. Therefore, LMT algorithm can be chosen as the most accurate model.

Generalization of the models,

Algorithm	Training Dataset Accuracy	Testing Dataset Accuracy	Difference between the Accuracies	Generalization
SMO	77.5%	76.6%	0.9%	quite well
LMT	77.5%	77.3%	0.2%	very good
Naïve Bayes	75.7%	76%	0.3%	very good
Random Forest	72.6%	77.3%	4.7%	not so well
J48	75.1%	70.1%	5%	not so well

Table 10: Level of Generalization of All the Models

Very good, quite well – The model's performance on the testing dataset is identical to its performance on the training dataset, indicates that it generalizes well to new data, which means the model is likely to make accurate predictions on similar unseen datasets.

Not so well – The model's performance drop from the training set to the testing set indicates that it does not generalize well and have to adjust the model to improve its generalization ability.

Confusion Matrix of LMT model, correctly identifies 87 true positives and 32 true negatives, it indicates that the model performs well in both detecting positive cases and correctly identifying negative cases. The 22 false positives indicates that the model has incorrectly predicted positive outcomes and the 13 false negatives indicates that the model has missed positive instances which is critical for diabetes, because positive cases can have serious consequences.

Attributes included in models,

Algorithm	preg	plas	pres	skin	insu	mass	pedi	age
SMO	✓	✓	✓	✓	✓	✓	✓	✓
LMT	✓	✓	✓		✓	✓	✓	✓
Naïve Bayes	✓	✓	✓	✓	✓	✓	✓	✓
Random Forest	Not included in the output							
J48		✓						

Table 11: Attributes Included in the Models

Here, 2-hour Plasma glucose concentration in an oral glucose tolerance test seems to be a significant predictor of diabetes.

Intuitiveness of the models,

Intuitive - Models created by algorithms like LMT, Naïve Bayes and J48 can be recognized as highly intuitive models, because they are particularly user-friendly due to their straightforward nature.

Not Intuitive - Models created by algorithms like SMO and Random Forest require an understanding about their ensemble methods and complex algorithms but still can be made intuitive with right tools and explanations.

7 Conclusion

In this study, five diabetes prediction models were created using the WEKA tool using five different algorithms. The primary objective was to identify the most effective algorithm for accurate and reliable diabetes prediction, which can aid in early diagnosis and intervention of diabetes.

The analysis revealed that the LMT algorithm demonstrated superior performance compared to other algorithms used such as SMO, Naïve Bayes, Random Forest, and J48. The model achieved an accuracy of 77.3%, indicating its robustness in predicting diabetes.

Some key insights of this study include significant attributes that predict diabetes such as 2-hour Plasma glucose concentration in an oral glucose tolerance test, and the performance of the models created such as the performance metrics, which provided a comprehensive understanding of each algorithm's strengths and weaknesses.

In conclusion, the diabetes prediction model developed using the WEKA tool demonstrates the potential of machine learning in healthcare.

8 Reference

[1] P. A. Flach and N. Lachiche, “Naive Bayesian classification of structured data,” *Machine Learning*, vol. 57, no. 3, pp. 233-269, 2004.

[2] J. Platt: Fast Training of Support Vector Machines using Sequential Minimal Optimization. In B. Schoelkopf and C. Burges and A. Smola, editors, Advances in Kernel Methods - Support Vector Learning, 1998.

[3] S.S. Keerthi, S.K. Shevade, C. Bhattacharyya, K.R.K. Murthy (2001). Improvements to Platt's SMO Algorithm for SVM Classifier Design. Neural Computation. 13(3):637-649.

[4] Trevor Hastie, Robert Tibshirani: Classification by Pairwise Coupling. In: Advances in Neural Information Processing Systems, 1998.

[5] K. Sridharan and G. Komarasamy, “Sentiment classification using harmony random forest and harmony gradient boosting machine,” *Soft Computing*, vol. 24, no. 10, pp. 7451-7458, 2020.

[6] Z. Wang, F. Chegdani, N. Yalamarti, B. Takabi, B. Tai, M. El Mansori, and S. Bukkapatnam, “Acoustic emission characterization of natural fiber reinforced plastic composite machining using a random forest machine learning model,” *Journal of Manufacturing Science and Engineering*, vol. 142, no. 3, 2020.

[7] N. Landwehr, M. Hall, and E. Frank, “Logistic Model Trees,” *Machine Learning*, vol. 95, no. 1-2, pp. 161-205, 2005.

[8] N. Bhargava, G. Sharma, R. Bhargava, and M. Mathuria, “Decision tree analysis on J48 algorithm for data mining,” *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, no. 6, 2013.