



Predicting Diabetes with Decision Trees

PROJECT 2: MACHINE LEARNING USING WEKA, JAVA FX

DEPENDENCIES

Sondos Aabed | 1190652
Artificial Intelligence COMP438 | Section 1
Department Of Computer Science

This report was written for the Artificial intelligence coursework, specifically the Machine Learning Project. The dataset is used to build a decision tree model to predict diabetes.

Predicting Diabetes with Decision trees models

Using WEKA, JavaFX dependencies

Contents

Motive and Overview	3
Chapter 1: About the Dataset	4
1.1 Features Overview	4
1.2 Target Attribute	5
Chapter 2: Classification Model Training, testing, and Evaluation	6
2.1 About the Decision Tree Models	6
2.2 Two models, two splitting rules	6
Model 1 M1.....	7
Model 2 M2.....	7
2.3 Generated Trees	8
2.4 Models Comparison	9
Chapter 3: Technologies.....	10
3.1 Maven Project	10
3.2 Java Weka Dependency.....	11
3.3 JavaFX Dependencies	11
Chapter 4: Implementation	12
4.1 Curious about diabetes Statics?	13
4.2 Are you at risk of diabetes?	15
4.3 Error Handling	18
Conclusion	19
References.....	19

Predicting Diabetes with Decision trees models

Using WEKA, JavaFX dependencies

Motive and Overview

In Palestine, diabetes is referred to as an epidemic. It was found to be the fourth leading cause of death among Palestinians. One of the reasons is the poor Palestinian healthcare for diabetes and the delay in treating and coping with it. A (diabetes risk of having) prediction model would improve the healthcare of diabetes in Palestine a step forward.

With that in motive, this is a proposed maven project that uses WEKA to build a decision tree model and JavaFX to build Graphical User Interface (GUI). It predicts and classifies people at risk of diabetes based on a diabetes dataset. And shows different types of statics summaries.

The dataset used in this project is called the diabetes dataset prepared by the National Institute of Diabetes and Digestive and Kidney Diseases (NIDDK). It contains 768 instances. Nine numerical attributes include one binary, the target class (Diabetic or not). No missing data were detected. Please note that the database is not collected in Palestine.

This report documents the development process of this project, it contains 4 chapters. Chapter one is About the Dataset where the Attributes are overviewed and the target attribute is discussed. Chapter two is concerned with building the models, training, and testing them. The higher accuracy model was used in the project. Chapter three elaborates on the technologies used WEKA, Maven, and JavaFX. Chapter four shows the GUI of the project and a demo.

Predicting Diabetes with Decision trees models

Using WEKA, JavaFX dependencies

Chapter 1: About the Dataset

This chapter is concerned with the dataset itself and how it's handled. It contains three sections where the features are overviewed and the target feature is. Then the data cleansing process is performed.

1.1 Features Overview

This dataset contains all numeric-valued attributes. Each column name indicates a term used in healthcare. For example. NPG is the Number of times pregnant. PGL is the Plasma glucose concentration 2 hours in an oral glucose tolerance test. DIA is Diastolic blood pressure in (mm Hg). TSF is the Triceps skin fold thickness (mm). INS is 2-Hour serum insulin in (mu U/ml). BMI is Body mass index. DPF is a Diabetes pedigree function. AGE is Age (years).

In the following table, the main Summary statics of each of those is provided:

<i>ATTRIBUTE NAME</i>	<i>MEAN</i>	<i>MEDIAN</i>	<i>STANDARD DEVIATION</i>	<i>MIN</i>	<i>MAX</i>
<i>NPG</i>	3.8451	3.0	1.8356	0.0	17.0
<i>PGL</i>	120.8945	117.0	5.6544	0.0	199.0
<i>DIA</i>	69.1055	72.0	4.3995	0.0	122.0
<i>TSF</i>	20.5365	23.0	3.994	0.0	99.0
<i>INS</i>	79.7995	30.5	10.7352	0.0	846.0
<i>BMI</i>	31.9926	32.0	2.8079	0.0	67.1
<i>DBF</i>	0.4719	0.3725	0.5756	0.078	2.42
<i>AGE</i>	33.2409	29.0	3.4293	21.0	81.0
<i>DIABETIC</i>	0.349	0.0	0.6906	0.0	1.0

Table (1): Summary Statics

It is noticed that the dataset instances vary from minimum age starting at 21 to the maximum age of 81 years old. Although without the proper background, some of these numbers are not interpretable. The question is for these measurements is it possible to have min values of zero? It is also noticed that looking at the min and max of the attributes, the ranges vary. **Task (1)**

Predicting Diabetes with Decision trees models

Using WEKA, JavaFX dependencies

1.2 Target Attribute

The target attribute is whether the record has diabetes or not. It is (Diabetic) a numerical attribute. Its values are either 0: negative for diabetes or 1: positive for diabetes. Therefore the problem is a binary classification problem.

As for the distribution of each class of them, it was found to be the following: **Task (2)**

- 39.9% were negative not having diabetes.
- 65.10% were positive for having diabetes.

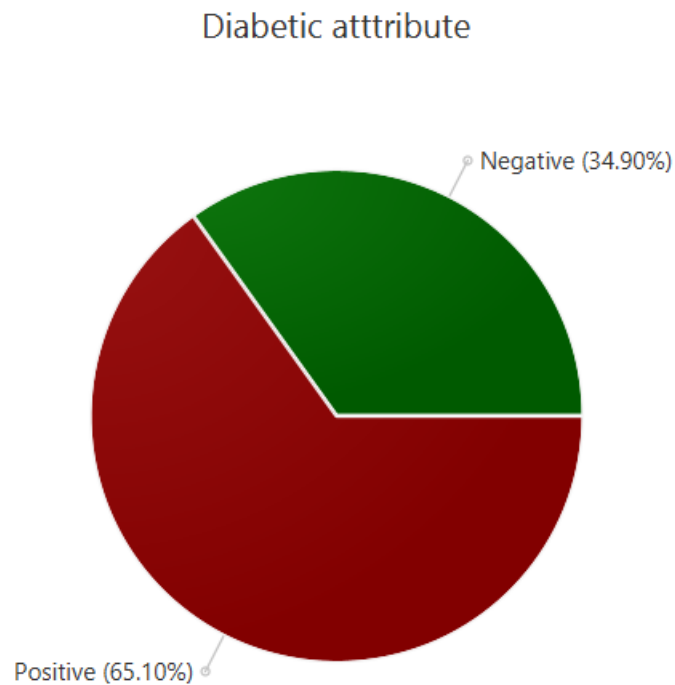


Image (1): Pie chart of Binary class target attribute

Predicting Diabetes with Decision trees models

Using WEKA, JavaFX dependencies

Chapter 2: Classification Model Training, testing, and Evaluation

In this chapter, two REPTree models are trained twice. One uses a 70:30 splitting rule and the other uses a 50:50 splitting rule. Trained and tested based on the splitting rule. The accuracy will be displayed and the generated tree will be shown. At the end of this chapter, a model will be selected based on the better evaluation.

2.1 About the Decision Tree Models

{Task (4)} In this project, the REPTree is used for many reasons. One of which that it is an open-source model that could be trained and tested. It is also a fast decision tree learner. Builds a decision/regression tree using information gain/variance and prunes it using reduced-error pruning. Only sorts values for numeric attributes once. Missing values are dealt with by splitting the corresponding instances into pieces as in C4.5. (WEKA, 2023)

Models training and testing

2.2 Two models, two splitting rules

As shown in the image below using the WEKA platform the percentage split is defined as 50% for M2 and once 30% for M1.

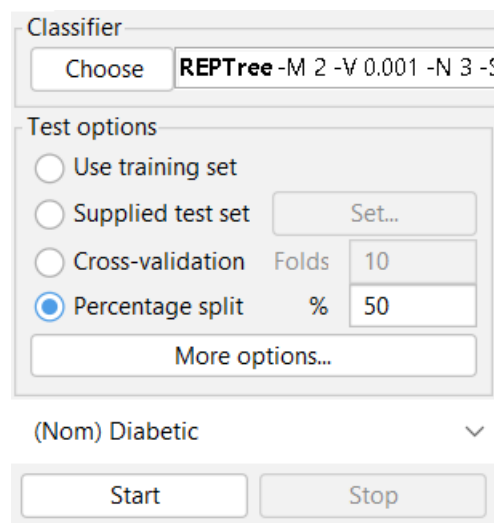


Image (2): define the splitting rule

Predicting Diabetes with Decision trees models

Using WEKA, JavaFX dependencies

Model 1 M₁

The first trained model is called M₁. The splitting rule is 70:30 training to testing rule. {Task (3)}. Before training this model the dataset is split into 70% of which are used for training and 30% of which are used for testing the accuracy of the generated model {Task (5)}. The dataset is also shuffled for better variation. This model has scored (73.4783%) incorrectly classified instances. It also incorrectly classified (26.5217%) instances. The following is the confusion matrix of M₁:

CLASSIFIED AS	A	B
A = 0	122	32
B = 1	29	47

Table (2): Confusion matrix of M₁

Model 2 M₂

The second trained model is called M₂. The splitting rule is 50:50 training to testing rule. {Task (7)}. Before training this model the dataset is split into 50% which is used for training and 50% which is used for testing the accuracy of the generated model {Task (7)}. The dataset is also shuffled for better variation. This model has scored (74.7396 %) correctly classified instances. It also incorrectly classified (25.2604%) instances.

CLASSIFIED AS	A	B
A = 0	206	53
B = 1	44	81

Table (3): Confusion matrix of m₂

2.3 Generated Trees

The following images show the Generated decision trees of models M₁ and model M₂ they have a few differences in the training model. For example, the tree generated by the first model has the size of the tree as 43. As for the second model, m₂ has the size of a tree as 23. But the final model has a tree-sized 33 and they both have the same tree {Task (9) and Task (8)}

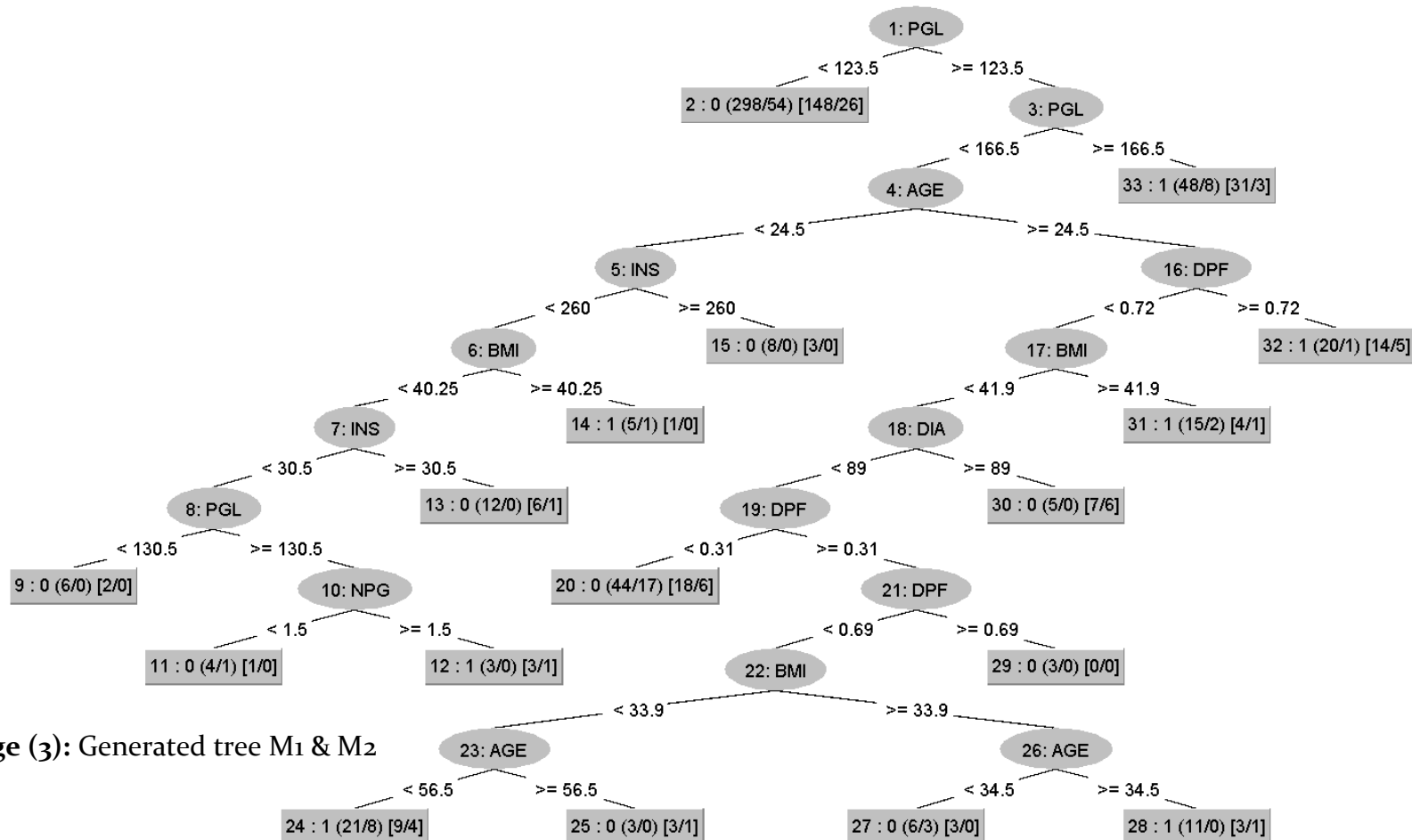


Image (3): Generated tree M₁ & M₂

2.4 Models Comparison

In this section, a comparison is made to return the better-evaluated model and use it in the Java project to predict a new instance. The training and testing are done outside of the project code using the WEKA platform because it is not a good practice to train and test the model each time the project is running instead it is trained and evaluated once and then used in the project to predict new instances. {Task (6)}

PERFORMANCE MEASURE	MODEL	
	MODEL 1 M1 70:30	MODEL 2 M2 50:50
CORRECTLY CLASSIFIED INSTANCES	169 (73.4783 %)	287 (74.7396 %)
INCORRECTLY CLASSIFIED INSTANCES	61 (26.5217 %)	97 (25.2604 %)
MEAN ABSOLUTE ERROR	0.3106	0.2989
TP RATE	0.735	0.747
FP RATE	0.324	0.304
PRECISION	0.738	0.753
RECALL	0.735	0.747
F-MEASURE	0.736	0.750
ROC AREA	0.743	0.777

Table (4): Performance Measurements

Based on this collected performance measurement is it noticed that Model 2 M2 has a better performance measurement but is very close to the first Model. So model 2 was selected to be used in the Java program prediction.

Predicting Diabetes with Decision trees models

Using WEKA, JavaFX dependencies

Chapter 3: Technologies

In this chapter, the technologies used to develop this project are explained in detail, those include Java Maven projects, Java Weka dependency, and JavaFX dependency.

3.1 Maven Project

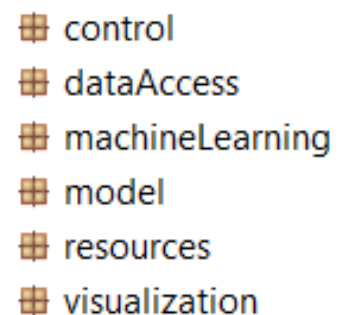
This project was created using a simple project (Archetype selection was ignored and the dependencies were added later manually). Using JAR packaging. This is the POM.xml where the project is created. JDK 20.

```
https://maven.apache.org/xsd/maven-4.0.0.xsd (xsi:schemaLocation)
1<project xmlns="http://maven.apache.org/POM/4.0.0"
2  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/
4  <modelVersion>4.0.0</modelVersion>
5  <groupId>birzeit.edu</groupId>
6  <artifactId>diabetis.predict</artifactId>
7  <version>0.0.1-SNAPSHOT</version>
8  <name>projecttwo</name>
9
10  <properties>
11    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
12    <maven.compiler.source>20</maven.compiler.source>
13    <maven.compiler.target>${maven.compiler.source}</maven.compiler.target>
14    <junit.jupiter.version>5.8.1</junit.jupiter.version>
15    <junit.platform.version>1.8.1</junit.platform.version>
16  </properties>
17
```

Image (4): POM.xml file snippet

These packages were created as shown below using JAR packaging. The control package contains controllers, and the DataAccess package loads the CSV file into a dataset and returns an Instances object. ML package created the models and trains them. Finally, the visualization package contains the scene classes.

Image (5): Project Architecture (packages)



About the resources package, it contains the diabetes.csv file. All the images contained also were retrieved from the flat Icon website the Glyph style by Luvdat. (Luvdat, 2023)

Predicting Diabetes with Decision trees models

Using WEKA, JavaFX dependencies

3.2 Java Weka Dependency

The Waikato Environment For Knowledge Analysis (WEKA) “The developer version” was added to the project POM.xml file as a dependency shown in the next image (3):

```
<!-- https://mvnrepository.com/artifact/nz.ac.waik  
<dependency>  
  <groupId>nz.ac.waikato.cms.weka</groupId>  
  <artifactId>weka-dev</artifactId>  
  <version>3.9.6</version>  
</dependency>
```

Image (6): WEKA dependency

3.3 JavaFX Dependencies

The JavaFX was used to develop the graphical user interface (GUI). The latest version is used 21-ea+23. The following image is a snippet of the POM.xml file that shows the JavaFX modules (controllers, base, and graphics) added.

```
26      <!-- https://mvnrepository.com/artifact/org.openjfx/javafx-controls -->  
27      <dependency>  
28          <groupId>org.openjfx</groupId>  
29          <artifactId>javafx-controls</artifactId>  
30          <version>21-ea+23</version>  
31      </dependency>  
32      <!-- https://mvnrepository.com/artifact/org.openjfx/javafx-base -->  
33      <dependency>  
34          <groupId>org.openjfx</groupId>  
35          <artifactId>javafx-base</artifactId>  
36          <version>21-ea+23</version>  
37      </dependency>  
38      <!-- https://mvnrepository.com/artifact/org.openjfx/javafx-graphics -->  
39      <dependency>  
40          <groupId>org.openjfx</groupId>  
41          <artifactId>javafx-graphics</artifactId>  
42          <version>21-ea+23</version>  
43      </dependency>
```

Image (7): JavaFX dependencies

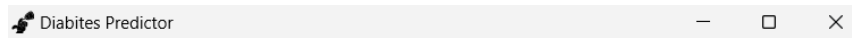
Predicting Diabetes with Decision trees models

Using WEKA, JavaFX dependencies

Chapter 4: Implementation

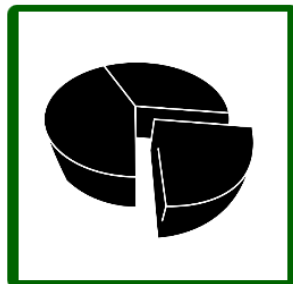
In this chapter, the implementation and showcase of the project are represented. The project's main scene will ask them to use a Call to action (CTA) whether they want to show statics or to check if they are at risk of diabetes.

Based on the user's choice, they will be redirected to another scene in which the user's needs are fulfilled. In the next two sections, these CTAs are shown.



Hello User!

Curious about Diabetes Statics?



Are you at risk of Diabetes?



Image (8): Home Scene

Predicting Diabetes with Decision trees models

Using WEKA, JavaFX dependencies

4.1 Curious about diabetes Statics?

Users are redirected to this scene if they were curious to know more about the diabetes statics. They are now able to decide whether they want to see statics in numerical summary and visualization. Another CTA is created here as shown in the following image:

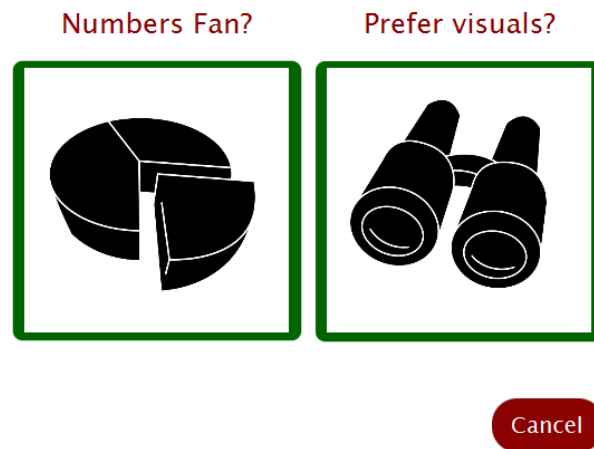


Image (9): Statics to be shown Type

If the user were a fan of numbers a table of numerical summary shows them:



Numeric Summary table

{Task (1)}

Image (10): Shows table

Name	Mean	Median	Std. Dev.	Min	Max	Count
NPG	3.8451	3.0	3.3696	0.0	17.0	768.0
PGL	120.8945	117.0	31.9726	0.0	199.0	768.0
DIA	69.1055	72.0	19.3558	0.0	122.0	768.0
TSF	20.5365	23.0	15.9522	0.0	99.0	768.0
INS	79.7995	30.5	115.244	0.0	846.0	768.0
BMI	31.9926	32.0	7.8842	0.0	67.1	768.0
DPF	0.4719	0.3725	0.3313	0.078	2.42	768.0
AGE	33.2409	29.0	11.7602	21.0	81.0	768.0
Diabetic	0.349	0.0	0.477	0.0	1.0	768.0

Cancel

Predicting Diabetes with Decision trees models Using WEKA, JavaFX dependencies

On the other hand, if the user prefers visuals a pie chart that shows the percentages of the target class will be shown to them as shown in image (5):

Task (2)

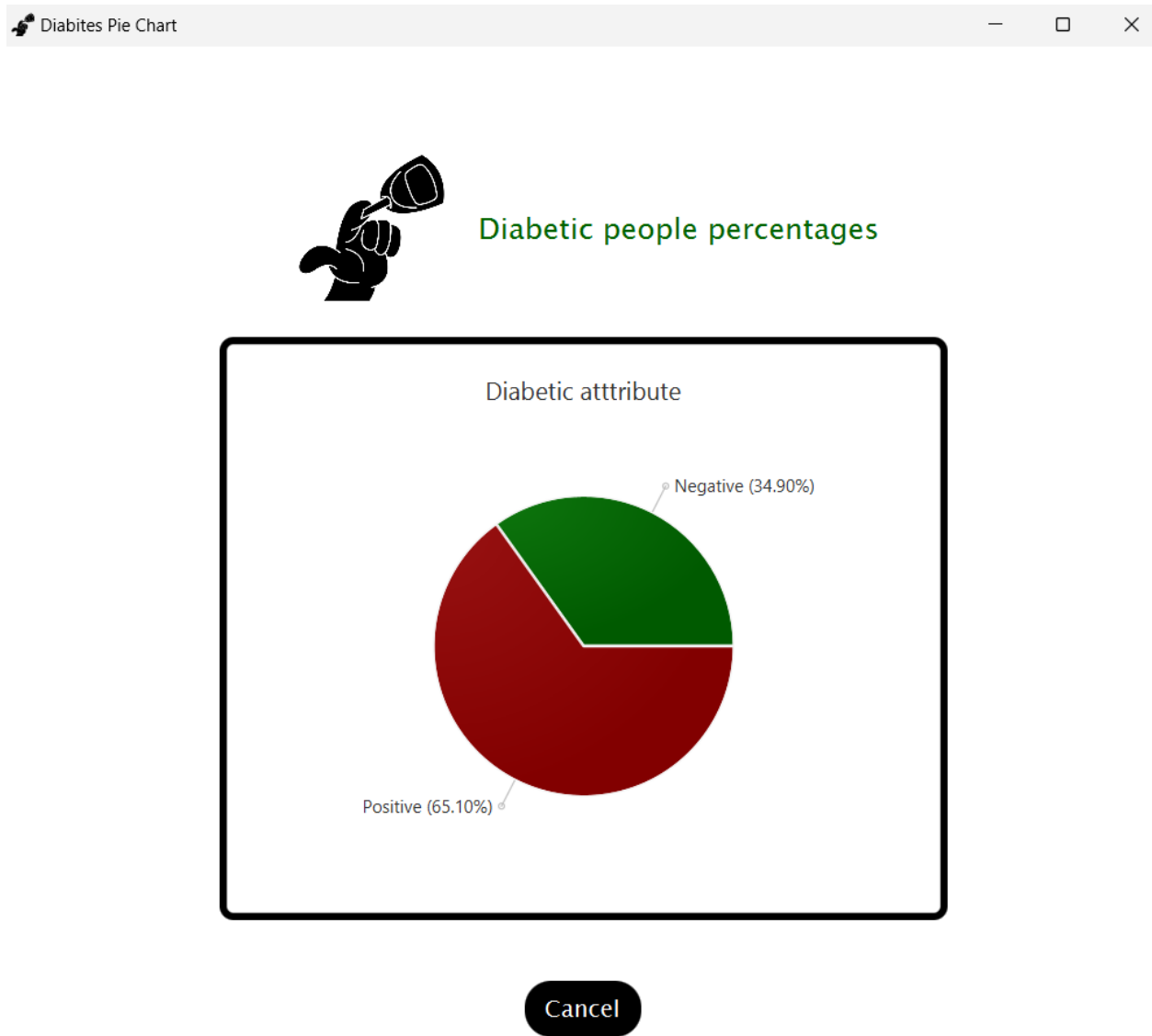



Image (11): Pie chart visualization


Predicting Diabetes with Decision trees models

Using WEKA, JavaFX dependencies

4.2 Are you at risk of diabetes?

If the user chooses the “Are you at risk of diabetes button to check if they are or not. They will be redirected to this form to create an instance and use the selected model in Chapter 2 to predict whether this user is at risk of diabetes.

 Diabetes Predictor — □ ×



Fill in the Form

NPG:	<input type="text" value="4"/>
PGL:	<input type="text" value="5"/>
DIA:	<input type="text" value="8"/>
TSF:	<input type="text" value="0"/>
INS:	<input type="text" value="0"/>
BMI:	<input type="text" value="35.6"/>
DPF:	<input type="text" value="0.134"/>
AGE:	<input type="text" value="39"/>

Image (12): Fill in the form

Predicting Diabetes with Decision trees models

Using WEKA, JavaFX dependencies

Clicking on the submit button will show a new stage that provides the user with whether they were at risk of getting diabetes or not. The resulting risk will be shown in image (7) and image (8).

If the predicted risk score is between 0.5 and 0.8 then the user gets this warning to be aware:



You're at 61.7 % risk of diabetes, Beaware!

Okay

Image (13): High risk

Else if the risk is less than 0.4 which is low this result risk will be shown to them:



Congratulations! your at low risk of diabetes 36.84%

Okay

Image (14): Low risk

Predicting Diabetes with Decision trees models Using WEKA, JavaFX dependencies

If the prediction score was close to 0.5 meaning is between 0.4 and 0.5 the user gets the following warning:



Warning! your at close risk of diabetes 45.6%

Okay

Image (15): close risk of diabetes

If the prediction score was higher than 0.8 that means that it's very likely they have diabetes image 16 will show them a warning to seek Dr:



You have a very high risk88.6%, Seek dr

Okay

Image (16): Very high risk

Predicting Diabetes with Decision trees models

Using WEKA, JavaFX dependencies

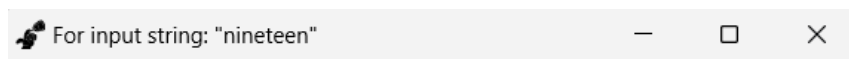
4.3 Error Handling

Whenever there is an exception caught on the program the Alert box Scene is shown to the user with the message of the cause of the error shown in code snippet line 100:

```
73  /**
74   * This is a method that handles the form submission
75   */
76  /**
77   * @param submit2
78   * @param data
79   */
80  private void handle_submit(Button submit2, Instances data) {
81      submit2.setOnAction(e -> {
82          RiskResultCtrl result;
83          try {
84              // Create a new instance based on the user's input on the form
85              Instance newInstance = new DenseInstance(data.numAttributes());
86              newInstance.setDataset(data);
87              newInstance.setValue(0, Integer.parseInt(npg.getText()));
88              newInstance.setValue(1, Integer.parseInt(pgl.getText()));
89              newInstance.setValue(2, Integer.parseInt(dia.getText()));
90              newInstance.setValue(3, Integer.parseInt(tsf.getText()));
91              newInstance.setValue(4, Integer.parseInt(ins.getText()));
92              newInstance.setValue(5, Double.parseDouble(bmi.getText()));
93              newInstance.setValue(6, Double.parseDouble(dpf.getText()));
94              newInstance.setValue(7, Integer.parseInt(age.getText()));
95              //create a new scene to show the result
96              result = new RiskResultCtrl(data, newInstance);
97              result.getWindow().show();
98          } catch (Exception e1) {
99              //If an exception is thrown show it in a new Alert box
100             AlertBoxCtrl alert = new AlertBoxCtrl(e1.getMessage(), e1.getLocalizedMessage());
101             alert.show();
102         }
103     });
104 }
105
```

Image (17): Low risk

An example of an error is if the user inputs string instead of numbers in the form



For input string: "nineteen"

Okay

Image (18): Input mismatch

Predicting Diabetes with Decision trees models

Using WEKA, JavaFX dependencies

Conclusion

In conclusion, a (diabetes risk of having) prediction model was built on this project. With the motive of assisting in the early diagnosis of diabetes in Palestinian healthcare. The project was built using maven project utilized WEKA in building the decision tree model and JavaFX in building GUI. To select the better performing model two splitting rules were performed and model 2 the one that uses the 50:50 splitting rule was found to be better and was selected to predict diabetes.

References

WEKA. (2023). Use WEKA in your Java code Retrieved from GitHub:

https://waikato.github.io/weka-wiki/use_weka_in_your_java_code/

Atilla Ozgur. (2012). How to decide which decision tree classifier to use? Retrieved from stack exchange:

<https://stats.stackexchange.com/questions/34940/how-to-decide-which-decision-tree-classifier-to-use>

WEKA. (2023). REPTree.html class Retrieved from WEKA documentation:

<https://weka.sourceforge.io/doc.dev/weka/classifiers/trees/REPTree.html>

Luvdat. (2023). glyph. Retrieved from FlatIcon.com:

<https://www.flaticon.com/authors/luvdat/glyph>