

**Manuel Ruiz Botella**

**RetiRisk Calc: An online risk calculator for Diabetic Retinopathy.  
Comparison with a neural network approach.**

**FINAL DEGREE PROJECT**

**Advisors: Dr. Aïda Valls Mateu and Emran Saleh Ali**

**Degree in Computer Engineering**



**UNIVERSITAT ROVIRA I VIRGILI**

**Tarragona**

**2019**



## **Acknowledgements.**

This work is supported by the URV grants 2018PFR-URV-B2-61 and 2017PFRURV-B2-60, and the Spanish research projects PI18/00169 and PI15/01150 (Instituto de Salud Carlos III and Fondos FEDER).

I would like to thank Dr. Aida Valls for the opportunity she has given me to work on this project and for the knowledge and support she has provided throughout all the months of work. I would also like to thank Emran Saleh who has been an amazing partner for this project and has helped me anytime I had a difficulty understanding something on the project.

I must not forget about my friends who have been hearing about my college stories for 5 years and have been always there supporting and giving me advice.

I can't forget either of my double degree partners and friends who have struggled with me for all this time and have been there on the longest nights and harder tests sharing their work for the best of all of us.

Finally, I really want to say thank you to my family and especially my parents who have been supporting me emotionally and financially not only these years but all my life. I will never forget all your time you have sacrificed for me, and it feels incredible how you are always there for me. Any objective I get in my life is thanks to you both.

## **Resumen.**

La Retinopatía Diabética es una de las enfermedades derivadas de la Diabetes Mellitus que pueden acabar provocando ceguera a los pacientes. A partir de 20 años muchos de los pacientes tienen un alto riesgo de sufrir Retinopatía Diabética. Actualmente la situación para la detección del riesgo de Retinopatía diabética se lleva a cabo a través de un análisis de una foto del fondo del ojo y se realiza a todos los pacientes. Esto supone un excesivo gasto temporal y económico. Actualmente hay en desarrollo modelos predictivos del riesgo de Retinopatía diabética a partir del registro electrónico de los pacientes de Diabetes, este modelo se llama RETIPROGRAM y se basa en un Fuzzy Random Forest para realizar la predicción.

Este modelo existente solo lo pueden usar los médicos debido a la complejidad de los atributos utilizados para realizar esta predicción. En este trabajo hemos realizado un estudio sobre los atributos. Además, hemos simplificado y unido dos de ellos en un nuevo atributo llamado “Riñón” y creado una nueva versión disponible para el uso de los pacientes desde sus casas y las clínicas ópticas. El modelo nuevo (RetiRisk Calc) tiene un 77% de acierto y un 76% de sensibilidad, muy cerca del modelo actual de RETIPROGRAM.

También hemos implementado un servicio web RESTful y una base de datos en Java para poner online esta nueva aplicación simplificada. Además, permite consultar el historial de un paciente. Por último, hemos afrontado el problema de determinar si un paciente tiene o no Retinopatía Diabética mediante una red neuronal multicapa, y hemos obtenido resultados parecidos, que sugieren que las redes neuronales pueden ser útiles en este problema.

## **Resum.**

La Retinopatía Diabètica és una de les malalties derivades de la Diabetes Mellitus que poden acabar provocant ceguera als pacients. A partir de 20 anys, molts dels pacients tenen un alt risc de sofrir Retinopatía Diabètica. Actualment la situació per la detecció del risc de Retinopatía Diabètica es porta a fer mitjançant un anàlisi d'una fotografia del fons de l'ull i es realitza amb tots els pacients. Això, suposa un cost temporal i econòmic excessiu. Actualment hi ha en desenvolupament models predictius del risc de Retinopatía Diabètica mitjançant el registre electrònic dels pacients amb Diabetis. Aquest model es diu RETIPROGRAM i es basa en un Fuzzy Random Forest per a fer la predicció.

Aquest model només el poden utilitzar els metges degut a la complexitat dels atributs utilitzats per a realitzar aquesta predicció. En aquest treball hem fet un estudi sobre els atributs. A més, hem simplificat i unit dos dels atributs en un nou atribut anomenat “Ronyó” i creat una nova versió disponible per als pacients des de les seves cases i les clíniques òptiques. El nou model (RetiRisk Calc) té un 77% de encert y un 76% de sensibilitat, molt aprop del model actual de RETIPROGRAM.

També hem implementat un servei web RESTful i una base de dades en Java per posar online aquesta nova aplicació simplificada. A més, permet consultar l'historial de un pacient. Per últim, hem afrontat el problema de determinar si un pacient té o no Retinopatía Diabètica mitjançant una xarxa neuronal multicapa, i hem obtingut resultats pareguts, que suggereixen que les xarxes neuronals poden ser útils en aquest problema.

## **Abstract.**

Diabetic Retinopathy is one of the diseases derived from Diabetes Mellitus that can end generating blindness to the patient. Once a patient of Diabetes has been with the disease over 20 years, it has a high risk of developing Diabetic Retinopathy. The situation for detection of the risk of Diabetic Retinopathy is being taken by a screening of the fundus of the eye and it is done to every patient. This supposes an excessive temporal and economic cost. There are been developed some predictive models of the risk of Diabetic Retinopathy based on the electronic health record from the Diabetes patients, this model is called RETIPROGRAM and it is based on a Fuzzy Random Forest to make the prediction.

This model can only be used by doctors due to the complexity of the attributes used to make this prediction. In this work, we have realized a study of the attributes. Afterwards we have simplified and grouped two attributes into a new attribute called “Kidney”. We created a new version for the use of the patients from their homes or optic clinic. The new model is called RetiRisk Calc and has a 77% of accuracy and a 76% of sensitivity, very close to the results of RETIPROGRAM.

We have also implemented a RESTful web service and a database in Java. Our aim is to put online this new simplified application. It also allows consulting the history of a patient. Lastly, we have faced the problem of determining if a patient has or no Diabetic Retinopathy with a multilayer neural net. Our results are similar to the ones in RetiRisk Calc, which may suggest that neural net can be useful in this problem



# Index

|                                                                                                     |           |
|-----------------------------------------------------------------------------------------------------|-----------|
| <b>1. INTRODUCTION .....</b>                                                                        | <b>5</b>  |
| 1.1. OBJECTIVES .....                                                                               | 7         |
| 1.2. STRUCTURE OF THIS DOCUMENT .....                                                               | 7         |
| <b>2. METHODS.....</b>                                                                              | <b>9</b>  |
| 2.1. RETIPROGRAM .....                                                                              | 9         |
| 2.1.1. Linguistic variables .....                                                                   | 9         |
| 2.1.2. Fuzzy Decision Trees .....                                                                   | 11        |
| 2.1.3. Fuzzy Random Forests .....                                                                   | 14        |
| 2.2. WEB SERVICES.....                                                                              | 16        |
| 2.3. NEURAL NETWORKS.....                                                                           | 21        |
| <b>3. RETIRISK CALCULATOR.....</b>                                                                  | <b>26</b> |
| 3.1. STUDY OF THE REDUCTION ON THE NUMBER OF ATTRIBUTES .....                                       | 26        |
| 3.1.1. Data analysis for these 2 variables.....                                                     | 27        |
| 3.1.2. Binarization of the dataset.....                                                             | 37        |
| 3.1.3. Create a simpler attribute based on both MA and CKD-EPI .....                                | 38        |
| 3.1.4. Obtaining RetiRisk Calc offline version .....                                                | 39        |
| 3.2. DESIGN OF OUR WEB SERVICE FOR RETIRISK CALC .....                                              | 41        |
| 3.2.1. Class diagram. ....                                                                          | 43        |
| 3.2.2. Database design .....                                                                        | 44        |
| 3.2.3. Hosting .....                                                                                | 45        |
| 3.3. IMPLEMENTATION OF A FEED FORWARD NN FOR DR CLASSIFICATION.....                                 | 45        |
| <b>4. EVALUATION .....</b>                                                                          | <b>48</b> |
| 4.1. RETIRISK CALC RESULTS.....                                                                     | 48        |
| 4.1.1. Results of binarizing data .....                                                             | 49        |
| 4.1.2. Results of deciding combination of MA and CKD-EPI .....                                      | 51        |
| 4.1.3. Results of different models of RETIPROGRAM data binarized with different<br>parameters ..... | 52        |
| 4.1.4. Results of RetiRisk Calc .....                                                               | 53        |
| 4.2. TESTING THE WEB SERVICE FOR RETIRISK CALC. ....                                                | 54        |
| 4.3. RESULTS OF OUR NEURAL NETWORK. ....                                                            | 57        |
| <b>5. CONCLUSIONS. ....</b>                                                                         | <b>59</b> |
| <b>REFERENCES .....</b>                                                                             | <b>61</b> |
| <b>APPENDIX A. REGISTER EVIDENCE OF RETIRISK CALC. ....</b>                                         | <b>63</b> |

# Figures

|                                                                           |    |
|---------------------------------------------------------------------------|----|
| FIGURE 1. COMPARISON BETWEEN NORMAL EYE (UP) AND DR EYE (DOWN).           | 5  |
| FIGURE 2. COMPARISON BETWEEN NORMAL EYE AND DR EYE.                       | 6  |
| FIGURE 3. LINGUISTIC VARIABLE EXAMPLE.                                    | 10 |
| FIGURE 4. LINGUISTIC VARIABLE FILE EXAMPLE.                               | 11 |
| FIGURE 5. FDT EXAMPLE.                                                    | 13 |
| FIGURE 6. RETIPROGRAM ARCHITECTURE.                                       | 16 |
| FIGURE 7. CREATING DATABASE.                                              | 19 |
| FIGURE 8. DATABASE CONFIGURATION.                                         | 19 |
| FIGURE 9. CREATE TABLE.                                                   | 20 |
| FIGURE 10. CREATE ATTRIBUTE.                                              | 20 |
| FIGURE 11. BIOLOGICAL SIMIL WITH NEURAL NETWORKS.                         | 22 |
| FIGURE 12. EXAMPLE OF ONE NEURON. FROM HUGO LAROCHELLE.                   | 23 |
| FIGURE 13. EXAMPLE OF NEURAL NET. FROM HUGO LAROCHELLE.                   | 24 |
| FIGURE 14. ALL PATIENTS DISTRIBUTION.                                     | 28 |
| FIGURE 15. ALL PATIENTS DISTRIBUTION BY MA.                               | 28 |
| FIGURE 16. HIGH MA PATIENTS DISTRIBUTION.                                 | 29 |
| FIGURE 17. LOW MA PATIENTS DISTRIBUTION.                                  | 29 |
| FIGURE 18. ALL PATIENTS DISTRIBUTION BY CKD-EPI WITH VALUE 90.            | 30 |
| FIGURE 19. OVER 90 CKD-EPI DISTRIBUTION.                                  | 30 |
| FIGURE 20. UNDER 90 CKD-EPI DISTRIBUTION.                                 | 31 |
| FIGURE 21. ALL PATIENTS DISTRIBUTION BY CKD-EPI WITH VALUE 60.            | 32 |
| FIGURE 22. OVER 60 CKD-EPI DISTRIBUTION.                                  | 32 |
| FIGURE 23. UNDER 60 CKD-EPI DISTRIBUTION.                                 | 33 |
| FIGURE 24. DISTRIBUTION OF PATIENTS (BASED ON MA) WITH OVER 60 CKD-EPI.   | 34 |
| FIGURE 25. DISTRIBUTION OF PATIENTS WITH OVER 60 CKD-EPI AND CORRECT MA.  | 34 |
| FIGURE 26. DISTRIBUTION OF PATIENTS WITH OVER 60 CKD-EPI AND HIGH MA.     | 35 |
| FIGURE 27. DISTRIBUTION OF PATIENTS (BASED ON MA) WITH UNDER 60 CKD-EPI.  | 36 |
| FIGURE 28. DISTRIBUTION OF PATIENTS WITH UNDER 60 CKD-EPI AND CORRECT MA. | 36 |
| FIGURE 29. DISTRIBUTION OF PATIENTS WITH UNDER 60 CKD-EPI AND HIGH MA.    | 37 |
| FIGURE 30. LINGUISTIC VARIABLE CONVERSION OF KIDNEY.                      | 39 |
| FIGURE 31. CLASS DIAGRAM.                                                 | 44 |
| FIGURE 32. EXAMPLE OF ReLU FUNCTION.                                      | 46 |
| FIGURE 33. RESULTS OF THE PATIENT 0.                                      | 55 |
| FIGURE 34. RESULTS OF THE PATIENT 1.                                      | 55 |
| FIGURE 35. SECOND RESULTS OF PATIENT 0.                                   | 56 |
| FIGURE 36. PATIENTS DATABASE OF OUR PROJECT.                              | 56 |
| FIGURE 37. RESULTS DATABASE OF OUR PROJECT.                               | 56 |
| FIGURE 38. RESULTS OF THE HISTORY OF THE PATIENT 0.                       | 57 |

## Tables

|                                                                                                                   |    |
|-------------------------------------------------------------------------------------------------------------------|----|
| TABLE 1. DATA PATIENTS' EXAMPLE. ....                                                                             | 11 |
| TABLE 2. MIN MAX METHOD EXAMPLE. ....                                                                             | 15 |
| TABLE 3. CKD-EPI TABLE, FROM [15].....                                                                            | 27 |
| TABLE 4. EXAMPLE OF OUTPUT FILE OF RETIRISK CALC.....                                                             | 41 |
| TABLE 5. MODELS FOR DATA BINARIZATION.....                                                                        | 49 |
| TABLE 6. RESULTS OF DATA BINARIZATION.....                                                                        | 50 |
| TABLE 7. CONCRETE CASES OF MODEL OF DATA BINARIZATION.....                                                        | 50 |
| TABLE 8. RESULTS OF CONCRETE CASES.....                                                                           | 50 |
| TABLE 9. MODELS OF COMBINATION OF BOTH ATTRIBUTES.....                                                            | 51 |
| TABLE 10. RESULTS OF COMBINATION OF BOTH ATTRIBUTES.....                                                          | 51 |
| TABLE 11. MODELS OF RETIPROGRAM FOR BETTERS PARAMETERS.....                                                       | 52 |
| TABLE 12. RESULTS OF MODELS OF RETIPROGRAM LOOKING FOR BETTER PARAMETERS.....                                     | 53 |
| TABLE 13. COMPARISON OF NORMAL RETIPROGRAM, BINARIZED DATA RETIPROGRAM AND RETIRISK CALCULATOR... 54              |    |
| TABLE 14. RESULTS OF COMPARISON OF NORMAL RETIPROGRAM, BINARIZED DATA RETIPROGRAM AND RETIRISK<br>CALCULATOR..... | 54 |
| TABLE 15. RESULTS OF OUR DIFFERENTS NEURAL NETWORKS COMPARED WITH THE RETIRISK CALCULATOR RESULTS.....            | 57 |

## Code

|                                                      |    |
|------------------------------------------------------|----|
| CODE 1. CONFIGURATION CLASS OF THE WEB SERVICE. .... | 42 |
| CODE 2. CODE OF OUR NEURAL NETWORK.....              | 47 |

## 1. Introduction

Diabetes mellitus (DM) is a group of metabolic disorders characterized by high blood sugar levels over a prolonged period. Normally it is caused by a decrease in the production of insulin by the pancreas, or the cells of the body not responding properly to the use of the insulin, or due to an increase of the production of glucose. Symptoms of DM are unintended weight loss, polyuria, polydipsia and polyphagia. DM is associated with several complications in organic systems like damage to the eyes, chronic kidney disease, diabetic ketoacidosis, cardiovascular disease and even death. It is a chronic disease, so there is no way to cure the disease, but there are several methods for prevention and treatment of DM related with a healthy lifestyle like a healthy diet, physical exercise and avoiding use of tobacco. For treatment is also very important a control of blood pressure, foot care, oral medications and insulin.

At the end of 2017, an estimated of 435 million people had diabetes worldwide, with type 2 DM making up for the 90% of the cases, type 1 for the 6% of the cases and the rest gestational diabetes. This represents the 8.8% of the adult population, and trends suggest that rates will continue to rise, as in 2013 it was 382 million patients. In 2017, DM caused between 3.2 and 5.0 million deaths [1].

As mentioned, DM can cause damage to the eyes. The most important disease than can be developed in the eyes due to DM is Diabetic Retinopathy (DR), which is the result of damage to the small blood vessels and neurons of the retina. DR often has no early warning signs, and every person with DM either type 1 or 2 can suffer from this disease. The longer you have DM, the higher probabilities of developing this eye disease, after 20 years of diabetes, nearly all patients with type 1 diabetes and more than 60% of patients with type 2 have some degree of DR [2].

RD has several stages. The non-proliferative diabetic retinopathy has no symptoms in the patients, and they may not notice the signs of reduced vision. The only way to detect this stage is by fundus photography, in which microaneurysms can be seen. These microaneurysms are microscopic blood-filled bulges in the artery walls of the eyes and there is a line of study based on computer vision techniques for the study of these microaneurysms [3 and 4]. Ten percent of the patients in this stage will develop macular oedema, in which the blood vessels in the eye leak their content into the macular region, this produces blurred vision and darkened parts in the vision. At figure 1, we can see in the DR eye an example of the microaneurysm that develops in the back of the eye.

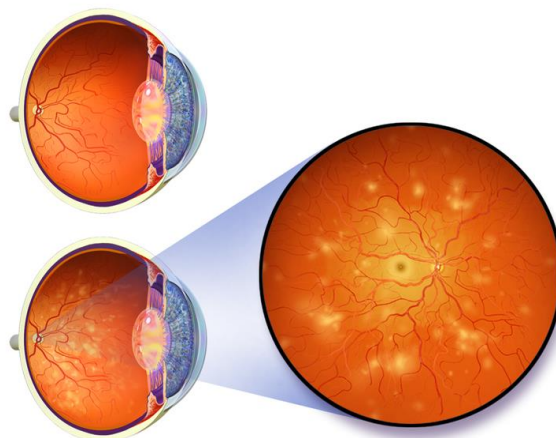


Figure 1. Comparison between normal eye (up) and DR eye (down).

At the second stage, abnormal blood vessels formed at the back of the eye will burst and bleed due to their fragility. These new vessels will produce bigger spots of blood floating in the patient visual field and more blur, that can lead to a patient being only able to differentiate between dark and light. As an example, we can see the comparison between a normal eye and an eye affected by DR in the figure 2.



**Figure 2. Comparison between normal eye and DR eye.**

Due to how severe is the damage done to the eye by the DR the early detection of this disease is very important. An early detection could be used for a better treatment for the patient and allows the patient to have a change in their lifestyle. If the DR is detected before the retina is severely damaged, there is a 95% chance of keeping the vision of the patient in a great state. The main treatments are laser surgery, injection of corticosteroids and vitrectomy.

For the detection of DR, doctors use screening of the fundus of eye. Sadly, even if doctors recommend that these screening should be done every year, it is very hard to achieve this due to the large number of diabetic people, the lack of enough materials, and the economic cost of the screening. In Spain with an established program, only 30% of patients can be screened each year, this leads to a patient being screened every 2.5 or 3 years, so a patient that did not have any sign in his last screening, could have developed a severe DR [5 and 6].

In the last years, the ITAKA research group (from URV) has worked with a medical team coordinated by Hospital Universitari Sant Joan de Reus in the design and implementation of a decision support system to assess DR risk by analysing the information contained in the electronic health record of a patient. This study line allows doctors to determine when a patient is at risk of developing DR or when this patient should be safe, and with this information, the next visit of the patient can be set earlier or later without a risk for the patient. This system is called RETIPROGRAM, and it uses a fuzzy random forest (FRF), which is an assemble of fuzzy decision trees, for the classification of patients that could have risk of developing DR [7 and 8].

Even with RETIPROGRAM being an important step in the good direction for determine whether a patient should be screened earlier or later, it relies on information of the electronic health record of a patient, and it can only be used and understood by doctors. This is the reason why some of the risk factors that lead to DR are difficult to be understood by non-medical people.

This Bachelor's Degree Final Project aims to study this risk factors in the RETIPROGRAM and develop a new simpler version (RetiRisk Calculator) that can be used by patients from their home with information that they know about their health. With this new version, the patients could have a rough estimation of the current risk of being

developing DR. It is important to achieve a constant self-monitoring of the disease, as well as to have an early detection of the first signs of DR in order that the person can ask for a new appointment with an ophthalmologist before it is too late. In addition, it could save even more time and money for doctors, who would see only patients that had used this version from home.

### **1.1. Objectives**

Herein I detail the objectives of this project:

- 1) Study and understand the current state of the RETIPROGRAM, how works a Fuzzy Random Forest, and how the classification algorithm and its parameters works.
- 2) Study the risk factors that are used in RETIPROGRAM for the classification of the patients and understand the factors that could not be known by the patients. Try changing some of the values of the risk factors in the dataset for see how the results of the RETIPROGRAM change.
- 3) Change the risk factors of the RETIPROGRAM, and redesign and implement a simpler version of this program trying to maintain similar accuracy results to the actual version.
- 4) Make a transformation of the training and testing datasets of the RETIPROGRAM to get this new simpler version usable by patients.
- 5) Design and implement a RESTful web service that hosts the new version of RETIPROGRAM. So that the patients can call this application with their browser in their homes giving their data to a formulary, and receive their possibility of DR.
- 6) Design a simple database that stores data of the patients and their results at each moment that the patient has used the application. Implement this database in the web service. The objective here is that the patient is able to understand if he is having an increment of its risk of DR and have this in account for next appointments with the doctor.
- 7) Develop the simpler version of RETIPROGRAM with a neural network classification model to compare them and see if this machine learning algorithm could lead to better results.

### **1.2. Structure of this document**

This document is divided in 5 sections.

In chapter 1, I have introduced DM and explain the risk factors of DR. I have also stated our objectives.

In chapter 2, I will show the reader the methods used in my project, from the RETIPROGRAM methodology and the depth knowledge and how it is built. Until how a Restful web service works and how neural networks are built and some interesting facts you should know to get a better neural network.

In chapter 3 I will explain my whole work in the project. First I will show you the data analysis I made for the simplification and how we built the new simplified version. After this I have built a web service so the RetiRisk Calc can be used on a web browser. Finally I have built a neural network so we can compare our application with a neural network approach.

In chapter 4, all the results obtained in the whole project of building the new application are shown. Also an example of how the web service would work and the final results of the neural network compared to the new application.

In chapter 5, I will talk about the conclusions we can obtain from this project and the future projects that can be developed from here.

## 2. Methods

In this chapter I'm going to expose all the information related to the overall project. The methodology used, and some studies made for justify this methodology. First, I will expose the theory of RETIPROGRAM, how RETIPROGRAM works and how it is built. Secondly, I will talk about the RetiRisk Calculator that is the main objective of this project. After this, I will make a brief introduction to Webservices to explain why we need them for this application, and a little explanation of databases, and how I've built a Restful WebService with a database. Finally, I will make an introduction to Neural Networks and why they are another possible solution for the classification of DM patients into yes or no DR.

### 2.1. RETIPROGRAM

RETIPROGRAM is a Clinical Decision Support System for Diabetic Retinopathy patients, which helps clinicians to estimate the risk that a DM patient has of developing DR. It is useful for personalize a patient's screening needs and to schedule follow up visits anytime in the following 3 years.

RETIPROGRAM is based on a Fuzzy Random Forest model (FRF), which is composed by a set of fuzzy decision trees (FDTs) that are used as classifiers. An FRF is an Artificial Intelligence model, specifically, it is an expert system based on fuzzy rules. Several rules are fired simultaneously with different levels of satisfaction, which lead to the assignment of different fuzzy scores to the set of possible conclusions. In this case, they make a binary classification between DR presence or not. Before explaining how FRF works, we will present the concept of linguistic fuzzy variables are and how FDTs use these variables.

#### 2.1.1. *Linguistic variables*

Fuzzy logic is the basis for definition of linguistic fuzzy variables. It is based on the generalization of classic Boolean logic by enabling a gradation in the membership function. It adapts better to the world in which we live because it is able to manage the uncertainty naturally present in many data evaluation [9 and 10].

A linguistic variable is a type of variable used in fuzzy logic, in which the variable takes values on a predefined set of linguistic terms (i.e. labels). Each of these terms has an associated fuzzy membership function on a reference numerical domain. In a way, we can translate the original precise numerical reference scale into a more general linguistic one. The main characteristics of these terms is that they have non-sharp boundaries. In fact, there is an intersection between the corresponding membership functions. Thus, depending on the membership function, several initial numerical values will belong to the same linguistic value for this term or to different linguistic values of this term. A variable can belong to two different linguistic values in a domain, but the membership value to each domain will be between 0 and 1, and both together must add 1.

For example, as a fuzzy membership function to convert the number of Km of distance to a linguist variable, we have the figure 3. There are 4 linguistic terms: {close, near, accessible or far}. Each one has a membership function for this domain of reference. In this case, the value of 5 km, would be converted to the linguistic value of "close", with

1.0 membership value. Also, the value 50 km, would be converted to the linguistic value of “near” in 0.5 membership value and to the linguistic variable of “accessible” in 0.5.

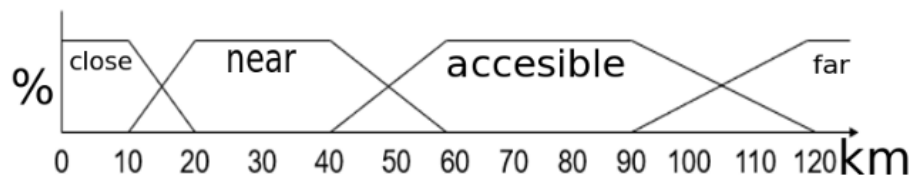


Figure 3. Linguistic variable example.

With this, if we made rules for deciding which transport should we use for travelling, we could have a rule of: IF “close” then WALK, then, the 5 km travel would be done by walking. We could have the rules: 1. IF “near” then BUS and 2. IF “accessible” then CAR. With these 2 rules, if we have the 50 km, both rules could be taken with the same score, and other attributes of the problem should be added for making the decision.

The data of the patients of the database, has been obtained by the doctors of Sant Joan University Hospital from Reus. This data is formed by many attributes of each patient. After a preliminary study, 9 attributes were selected as relevant ones [11].

The actual 9 attributes that are used in the FDTs of the RETIPROGRAM can be separated in:

- Numerical. They have a numerical scale that has been transformed into several fuzzy linguistic categories (2 or more) for the attribute, and depending on the numeric value, this attribute in the patient will belong to one of the linguistic values.
- Categorical. In which the value of the attribute is originally linguistic and there is not fuzziness. The set of possible categories is predefined in advance. For example, the gender of a patient can be only male or female. In this case, the membership to the labels is Boolean, for example for a woman, it would be 0 for male and 1 for female.

The data used in the current version of RETIPROGRAM has 3 categorical attributes and 6 numerical attributes. For training and testing with patients, the classifying attribute that indicates if the patient is ill or not, named RD, is categorical too. It can only be 0 if the patient does not have DR or 1 if the patient has DR. So, RETIPROGRAM works with the following risk factors:

- Current age. Numerical.
- Gender. Categorical.
- Body mass index. Numerical.
- Duration of DM. Numerical.
- Treatment of DM. Categorical.
- Control of arterial hypertension. Categorical.
- HbA1c percentage. Numerical.
- Estimated glomerular filtration rate, calculated from creatinine using chronic kidney disease epidemiology collaboration equation (CKD-EPI equation). Numerical.
- Microalbuminuria. Categorical.

For every numerical attribute, it has been studied by the doctors how to divide the original measurement scale into different linguistic categories, according to the different values and its incidence in the RD disease.

As an example of how the data of the patients is at first, before it enters the RETIPROGRAM, we have the table 1.

| ID | Age | Sex | EVOL | TTM | HbA1c | CDKEPI | MA   | BMI  | HTAR | RD |
|----|-----|-----|------|-----|-------|--------|------|------|------|----|
| 1  | 54  | 0   | 10   | 1   | 9.5   | 65     | 15.3 | 27.3 | 0    | 0  |
| 2  | 63  | 1   | 5    | 2   | 7.7   | 15     | 89   | 31   | 1    | 1  |
| 3  | 88  | 0   | 19   | 2   | 12    | 49     | 27   | 32.1 | 1    | 1  |

Table 1. Data patients' example.

And this data, is transformed to linguistic values, in the java code, with a key file, which is the LV file. An example of how numerical linguistic variables are classified for the FRF to work with the attributes, is in the figure 4.

```
#LV BMI Numerical 10 50
Underweight 18,19 RE
Normal-Low 18,19,22.5,23.5
Normal-High 22.5,23.5,24.5,25.5
Overweight-Low 24.5,25.5,27,28
Overweight-High 27,28,29.5,30.5
Obese-Low 29.5,30.5,34,36
Obese-High 34,36 LE
;
```

Figure 4. Linguistic variable file example.

### 2.1.2. Fuzzy Decision Trees

The whole FRF is composed by a number (which can be set as much bigger as we want) of FDTs. Each node of these trees corresponds to an attribute, each child corresponds to a possible linguistic value of this attribute, and each leaf node corresponds to one of the two possible outcomes of the tree: RD or no RD. Each decision tree in the forest makes an individual prediction. When every tree has given a prediction, the choice made by most of the trees is the final choice of the FRF.

Each tree is built in the same way, based on Yuan and Shaw work [8 and 10], which is similar to the nonfuzzy decision tree method ID3 Algorithm. The most important difference is that while the ID3 Algorithm uses the information entropy for choosing an attribute to separate the data, this fuzzy algorithm uses the classification ambiguity, a measure obtained for every attribute based on the different labels that exist for this attribute and the possibility of an object pertaining to this label. While it is near to 0, the ambiguity is better (there is no ambiguity).

Another important parameter in this fuzzy decision tree algorithm is the truth level, used for calculation of classification ambiguity, and is the relation between the conditions of a rule and the conclusion of a rule. For example, if we have a rule IF age is young AND bmi is normal THEN no RD. The truth level value is the possibility that if both conditions are true, the patient does not have DR.

The formula of the truth level measures the degree in which  $A$  is a subset of  $B$ , so if the  $A$  condition is true, how many objects belong to  $B$ .

$$S(A, B) = \frac{M(A \cap B)}{M(A)} = \frac{\sum_{u \in U} \min(\mu_A(u), \mu_B(u))}{\sum_{u \in U} \mu_A(u)}$$

Being  $\mu_A(u)$  the possibility of the object  $u$  of belonging to the label  $A$ .

Classification ambiguity, with fuzzy partitioning  $P = \{E_i, \dots, E_k\}$  on fuzzy evidence  $F$ , is the weighted average of classification ambiguity with each subset of the partition  $P$ . The equation of the classification ambiguity is:

$$G(P|F) = \sum_{i=1}^K w(E_i|F) * G(E_i \cap F)$$

, where  $w(E_i|F)$  is the weight which represents the relative size of a subset  $(E_i \cap F)$  in  $F$ ,

$$w(E_i|F) = \frac{M(E_i \cap F)}{\sum_{j=1}^K M(E_j \cap F)}$$

and with  $G(E_i \cap F)$  being the classification ambiguity with fuzzy evidence  $E_i \cap F$ . The classification ambiguity with fuzzy evidence  $E$  is defined as  $G(E) = g(\pi(C|E))$ . Given that  $\pi(C|E)$  is the possibility of classifying an object to class  $C$  if we have the evidence  $E$ ,

$$\pi(C_i|E) = \frac{S(E, C_i)}{\max(j) S(E, C_j)}$$

and  $g(\pi(C|E))$  is the ambiguity for this attribute  $E$  between the possible outcome classes.

$$g(\pi) = \sum_{i=1}^n (\pi t_i^* - \pi t_{i+1}^*) * \ln i$$

, where  $\pi t^* = \{\pi t_1^*, \pi t_2^*, \dots, \pi t_n^*\}$  is the permutation of the possibility distribution between all the objects of the dataset, sorted that  $\pi t_i^* \geq \pi t_{i+1}^*$  and  $\pi t_{n+1}^* = 0$ .

For getting better results and filtering nodes in the tree, we have two parameters:

- Significance level,  $\alpha$ . Used for eliminating the non-relevant evidence and objects that don't give valuable information to the calculation of classification ambiguity. If the membership of the objects to a fuzzy evidence (label of an attribute being used for the classification ambiguity) is lower than this  $\alpha$ , the membership of the objects will count as 0, being able to get a better classification ambiguity.
- Truth level threshold,  $\beta$ . Used for controlling the growth of a tree. Defines the minimum truth level for classifying all the objects within a branch into a certain class, in this case, a leaf node is created. A lower  $\beta$  can lead to smaller trees, with less accuracy, but a higher  $\beta$  can make that there are never leaf nodes in the tree.

The algorithm to build a FDT works with the next steps:

- The attribute with the smallest ambiguity is taken as the root node.
- A new branch is created for each of the values of the attribute selected, only those labels that have a possibility higher than the significance level defined.

- For each branch, the truth level of classification is calculated for both classes RD and no RD.
- If the truth level of classification is higher than  $\beta$ , this node converts to a leaf, which corresponds to the class with highest truth level (RD or no RD). With the nodes that do not accomplish this, a new attribute is chosen, depending on new calculations of the ambiguity with accumulated evidence. Now it repeats from step 2 until every end of branch is a leaf or until there is no way of continuing because every labels does not accomplish the significance level.

An example of a possible FDT that can be created with our dataset is shown in the Figure 5. The attributes used are selected randomly for the attributes mentioned before, but everything else in this example is made up for simplifying the process. We suppose that we have a subset of size 3 attributes for the tree. We set  $\beta$  as 0.7 and  $\alpha$  as 0.3. Every possibility of the node not mentioned is because it is higher than  $\alpha$ , so that node is created.

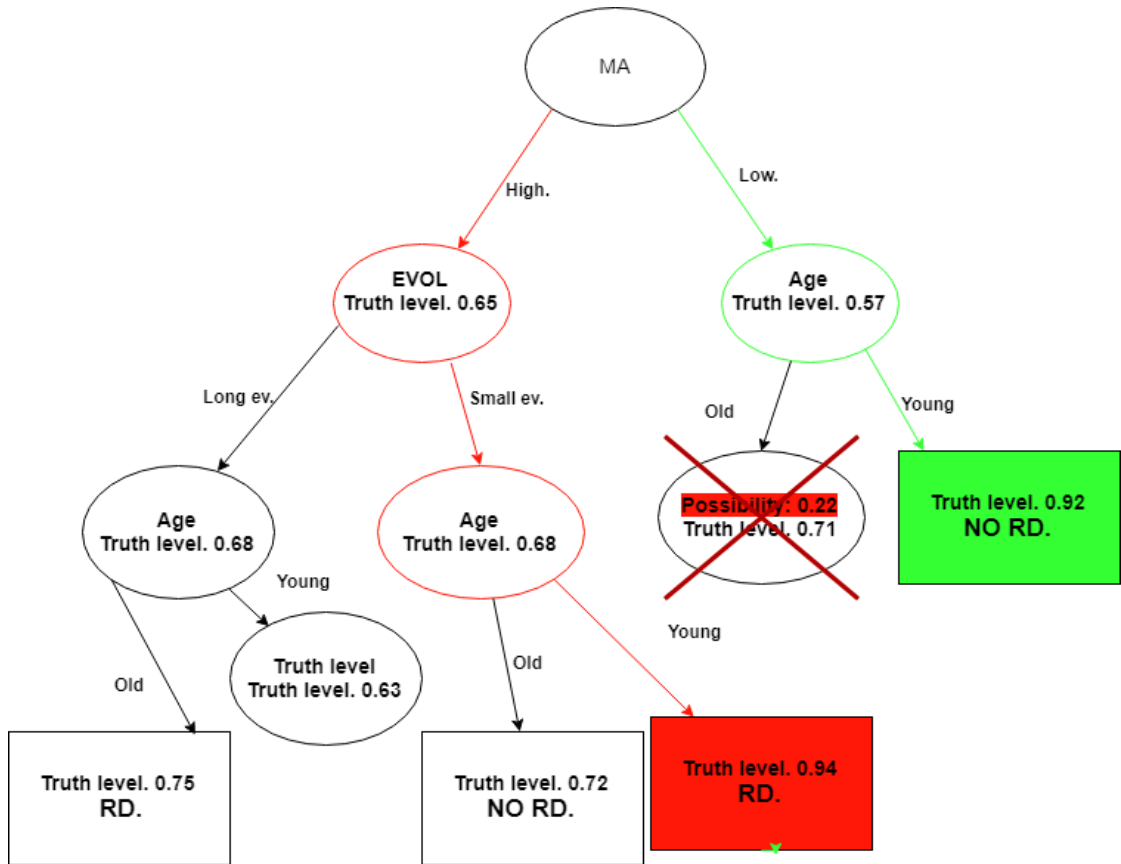


Figure 5. FDT example.

At first, the MA is the attribute with less classification ambiguity, so it is chosen for being the root. As none of the truth levels produced for the branches derived from this MA node is higher than 0.7, we continue evaluating the attributes checking the classification ambiguity.

For the “high” branch, the next attribute chosen is EVOL, and the branch that comes with “Long ev” has a truth level of 0.68 of RD, so the attribute Age is added to this branch. The “old” value has a truth level classification of 0.75 RD, so it becomes a leaf node of the branch, concluding class RD. The opposite happens with the “Young” value because the truth level is just 0.63, so it will not become a leaf node, and the algorithm will continue expanding this branch. For the “Small ev” branch, it has a truth level of 0.68

which is lower than 0.7, so it uses another attribute: Age. Every branch of Age has a truth level higher than 0.7 so they both produce leaf nodes. One for NO RD class and the other for RD class.

For the “Low” branch of MA (right side of the tree), we have that the attribute with lower ambiguity is Age, so we use that attribute for next classification. The “young” branch has a truth level of 0.92 for NO RD classification, so this node is a leaf. While the other branch “Old”, all the examples have a membership degree below 0.22 on this term, as it is lower than the significance level  $\alpha=0.3$ , this branch is stopped.

Once we have the tree, each path from the root to a leaf node is a rule, in this case we have 4 rules (a good rule would be the green in the example and a bad rule would be red in the example).

- 1 IF MA is High AND EVOL is Long ev AND Age is Old THEN RD
- 2 IF MA is High AND EVOL is small ev AND Age is Old THEN NO RD
- 3 IF MA is High AND EVOL is small ev AND Age is Young THEN RD
- 4 IF MA is Low AND AGE is Young THEN NO RD

Every of those rules has a final confidence value, equal to it final truth value.

For training and testing, RETIPROGRAM uses a dataset of 2323 patients, separated in a training group of 1212 patients and a test group of 1103 patients.

A usual problem with data in medical applications is that the low incidence of some diseases generates imbalanced datasets. In the training set, from the 2323 patients, 579 of them are patients with DR while 1744 patients are not affected by DR. This is big difference in favour of class 0, so it could exist an overfit in the learning process that cause problems to the learning algorithm. To avoid this problem, the training data set should be balanced so that the class distribution is 50 % of class 0 and 50% of class 1. This may be done by replicating in a random way some class 1 examples.

### **2.1.3. Fuzzy Random Forests**

A fuzzy random forest is composed by a set of fuzzy decision trees, each one used as a classifier [12]. As one FDT is built with the attributes, it is able to make a classification of the patients, but it is simply based on statistics and it is not really scalable unless new attributes are added to the classification model. A forest of a lot of trees, developed with the same attributes give us the opportunity of training different trees for the same problem with different attributes and see how the whole forest classifies the patients based on the result of each tree. With this method, some characteristics of the problem that only a tree would be unable to realize are recognized by more trees and can be useful for the classification.

The algorithm for a FRF that uses FDTs is the same that we explained before, but with some changes that allow us to have variety within the different trees. These changes are:

- Each tree gets two thirds of the whole dataset randomly to build each FDT. The selection is made in a way that the set obtained is balanced [13].
- Each time the FDT algorithm has to select an attribute (the root one or some other descendant), a subset of size  $\gamma$  of all the attributes is selected, and from these attributes the ambiguity classification is calculated. [12]

These two changes minimize the problem of overfitting as well as the class imbalance, which were the two major weak points of the FDT technique. In addition, the

redundancy brings some benefits in the classification accuracy. This has been experimentally proven in the RETIPROGRAM evaluation [8 and 11].

Once we have built every tree in the forest, we have to make a decision based on every vote of the trees. Many techniques can be used to establish the final decision of the random forest. The method used in the work to classify an example is based on Mamdani inference procedure but introduces two new thresholds.

The data of an example of the test set is fed into the  $n$  trees. Rules are activated using the Mamdani inference procedure, with t-norm minimum for calculating the satisfaction degree of the premises, and t-conorm maximum to aggregate the conclusions. Each activated branch (i.e. rule) of a tree gives a conclusion: the class label (RD or no RD) with a membership degree  $\mu$ , which is the truth level that the has leaf node used for this rule multiplied for results obtained by the Mamdani inference.

As an example, we have the rules obtained before for the tree, and we have a patient with the following membership values:

- 0.6 to High in MA and 0.4 to Low in MA.
- 0.85 to Long ev in EVOL and 0.15 to Small ev in EVOL.
- 1 to Old and 0 to Young.

The results are presented in the table 2.

| IF         | Premise | AND           | Premise | AND2         | Premise | MIN         | THEN  | Conclusion | Max         | Final       |
|------------|---------|---------------|---------|--------------|---------|-------------|-------|------------|-------------|-------------|
| MA is High | 0.6     | EVOL is Long  | 0.85    |              |         | <b>0.6</b>  | RD    | 0.75       | <b>0.6</b>  | <b>0.45</b> |
| MA is High | 0.6     | EVOL is Small | 0.15    | Age is Young | 0       | <b>0</b>    | RD    | 0.94       | <b>0.6</b>  | <b>0.56</b> |
| MA is High | 0.6     | EVOL is Small | 0.15    | Age is Old   | 1       | <b>0.15</b> | NO RD | 0.72       | <b>0.15</b> | <b>0.11</b> |
| MA is Low  | 0.4     |               |         | Age is Young | 0       | <b>0</b>    | NO RD | 0.92       | <b>0.15</b> | <b>0.14</b> |

Table 2. Min max method example.

Basically, this method gets the minimum of every premise in the rule. Then, it multiplies the maximum value obtained times the truth level of the rule (i.e. the rule confidence). After this, it gets the maximum of the minimums for every class, in this case, for RD, the maximum between 0.45 and 0.56 is 0.56 (Table 2). This will be membership degree  $\mu'$  of the tree for each class.

After executing all the classification trees for a certain patient, we get a result class with a  $\mu'$  degree on every tree and we have to aggregate all the results together. RETIPROGRAM admits different aggregation mechanisms. The ones that give best results are the following two:

#### 1. Minimax aggregation

First, for each tree, it compares the maximum  $\mu'$  of the two classes to see if the difference of the membership values is significant to decide a class. If the difference of the membership degree is smaller than given threshold  $\delta_1$ , no decision is made and the output class is set to *Unknown*.

Next, it counts the number of trees predicting each class. If the difference of the two classes is less than certain threshold  $\delta_2$ , the class is again *Unknown*. In a normal case, the conclusion will be RD or NO RD, depending on the trees.

#### 2. Choquet-based aggregation

Another classification method used in RETIPROGRAM is the Choquet model, which is based on the use as aggregation operator the Choquet integral. The Choquet integral equation is:

$$Choquet(f) = \sum_{i=1}^{\mu} [f(x_{s(i)}) - f(x_{s(i-1)})] * m(A_{s(i)})$$

This operator permits to aggregate a large set of values considering interactions between the sources. In our case, the interaction values (given in the  $m$  function) are obtained from the truth level of the rules, which are the sources of the class predictions. The process for calculating the  $m$  function is explained in [8].

A summary of the architecture of the decision support system RETIPROGRAM explained above is displayed in the Figure 6.

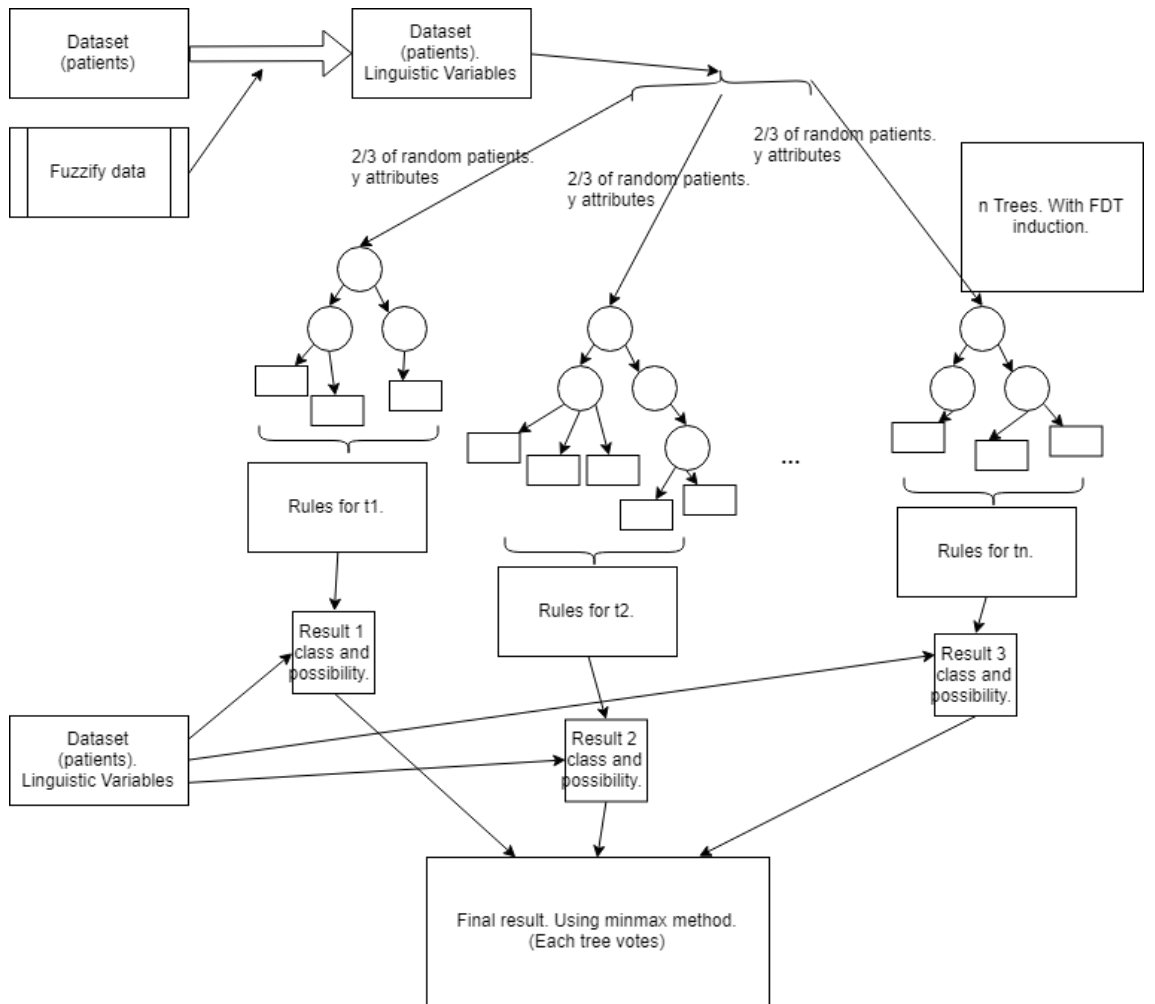


**Figure 6.** RETIPROGRAM architecture

## 2.2. Web services

A web service is a software system designed to support interoperable machine-to-machine interaction over a network. It is based in which other systems interact with the web service, this executes some operations based on the petition of the clients and then gives results to the client, usually via HTTP.

A web service architecture is composed by a service provider, which deploys the service and registers it, a service requestor (the client), who invokes the service and a service registry, that advertises the web service description for the client to be able to use the web service.

There are 2 popular methods of implementing a web service.

- Big web service, the SOAP web services. In which an interface in machine processable format is described so the clients can communicate with the web service via SOAP messages in XML or HTTP.
- RESTful web services. In which the main objective is to manipulate the representations of web resources using stateless operations.

While both types of web services are useful for our purpose, the RESTful type is simpler to use and simpler to learn, so we decided to design a Restful web service.

An example of how we could call to a method in a RESTful web service is:

GET <http://www.tfg.com/notas/UserName/ManuelRuiz>

And a response from the web service would be returned as a HTTP message, which could be seen in the web browser.

The most important characteristics of RESTful web services are that:

- Web services are stateless, so the connection between the web service and the client does not need to be open always, just the pass of messages by client and web service is needed for communication-
- Web services are made for code on demand, so there is no any problem if after the deployment of our application we want to add some new functionalities or modify the version of the application for one with better classification results, we simply can modify the project and restart the web service.
- It works with URI and it is a very simpler method to access to resources.
- It works with HTTP which is supported by almost every software in the world.
- RESTful web services are scalable.

Obviously, there are a lot of ways of developing a web service, we decided we wanted to develop one in the same IDE that the RETIPROGRAM, so we developed it in NetBeans in Java.

RESTful web services work mainly with resources, and every resource has a URI that the client can access through different interfaces similar to how SQL works with databases. The most important interfaces to work with resources are GET for reading information of the resource, POST for creating objects in the web service, UPDATE for creating or updating new objects and DELETE for delete resources from the web service. There are other interfaces that can be used like HEAD, SELECT, INSERT, UPDATE and (Systables).

An example of how these operators would be used in a database and in a web services, we have the next 2 points. Let's say we want to put the data of a customer with an ID of 567 in an application.

- SQL: PUT INTO CUSTOMERS (IF, ...) VALUES (567, ...)
- Rest: PUT [http://example.com /customers/567](http://example.com/customers/567)

In Java, we have the class `javax.ws.rs.Application` that allows us to create a Web Service. Basically, we need a configuration class, which would be the main of the

application and that is the only portable way of telling JAX-RS which classes are associated with our application. This main class has to extend the `javax.ws.rs.Application` and set the `ApplicationPath`, which normally would be simply a `"/"`, and override the `getClasses()` method of `Application`, on this `getClasses` we must add every class that we want to use in the web service. The application path of the web service is set with `"@ApplicationPath("/")"` and all the resources will be relative to this resource.

Once a configuration class has been created, we have to implement all the resources that the web service is going to use. For this, a normal class of java is created, but we need to add the labels that are used for Restful Web Services implementation.

On each class, we have to add before the definition of the class the label `@Path("$pathtothefile")`. Therefore, the class will be accessible via the URI of the web service and this extra path. Before each method, there are several labels that must be added for a correct configuration of the method in the web service. The most important one is the operator that would be used for accessing this method, if we want to get information, before the method we must add the label `@GET`. It is also very important to add what is the output of this method, as normally the message will be embedded in HTTP this is important for the browser and how to understand the body of the message. We can also define a path for this method relative to the class path, this is done with the label `@Path($path)` before the method.

For obtaining different parameters on our methods, we have 2 ways.

- Within the path defined, we can add a parameter between `{ }` and in the parameters of the method, the label used for getting this parameter has to be `@PathParam("")`. Then, when a client calls this method, the text where this parameter is will be used as a text. As an example: If the path is set as `@Path("/hola/{persona}")`, and the method has `@PathParam("persona") String persona`, the method will receive the parameter `persona`, and if the client uses the URI <http://localhost:8080/hola/carlos>, the method will receive "carlos" as the parameter `persona`.
- We can also define in the parameters of the method the label `@QueryParam("name") Type name`. For obtaining this parameter, the client must call the method with the path of the method and the webservice and after it add `"?name=value"`. Several attributes can be added with `&`. Relative to the example already stated with this implementation would be: <http://localhost:8080/hola?persona=carlos>.

With all this setup, we can create a working RESTful Web Service. As we are going to develop out Web Service in NetBeans we are also interested in the opportunity of designing a database in NetBeans. So, I will make a brief introduction to how databases are created in NetBeans and in the next points of the document I will explain the whole database.

Databases in NetBeans are very easy to create as we there is almost zero database knowledge and SQL knowledge for developing one. First, we have to access the Services tab in the menu of NetBeans and select Databases.

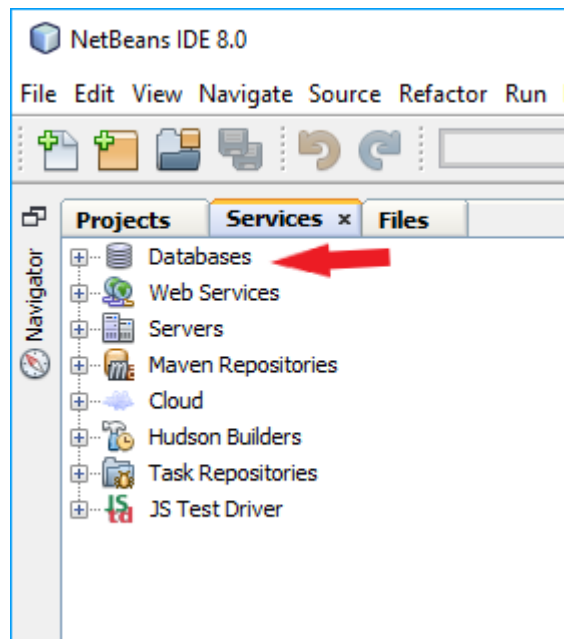


Figure 7. Creating database.

Inside Databases we click on Java DB and select “Create Database”.

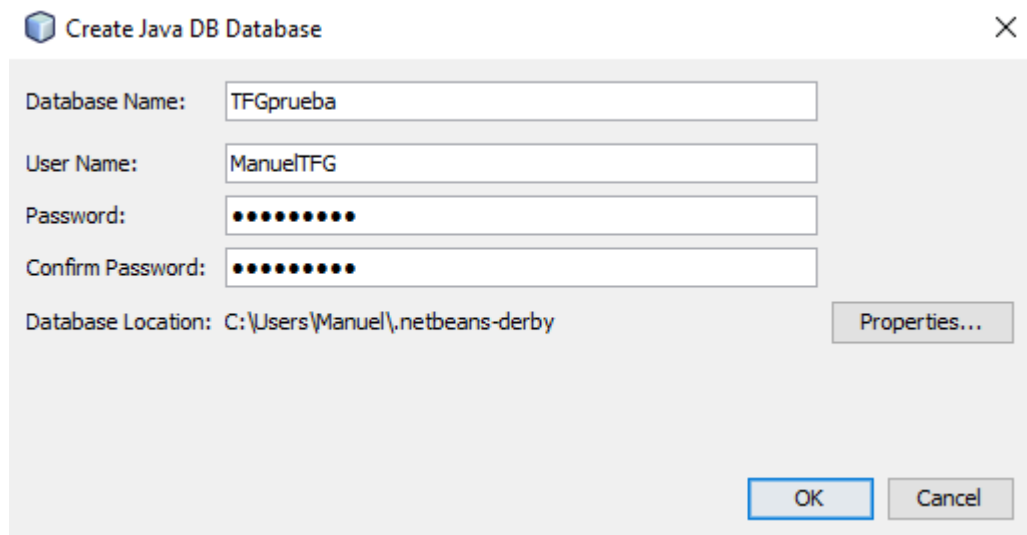


Figure 8. Database configuration.

It allows us to set a username and a password for the administration of the database and the use of the database. Once we click on OK the database is created. Now we have to click on it with the right click and click on “Connect”. Once we have this database connected we can create tables, views and procedures.

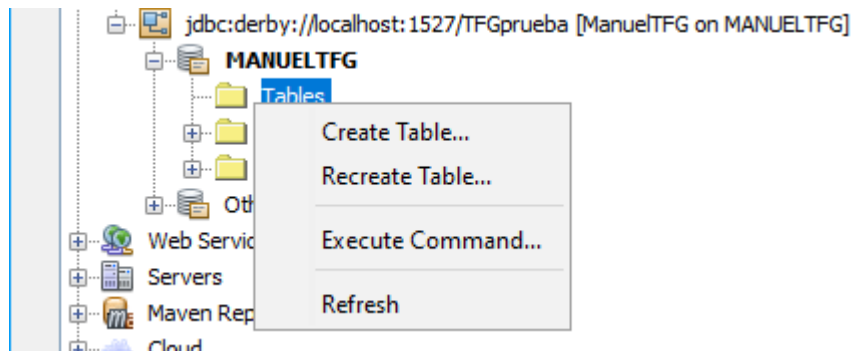


Figure 9. Create table.

We are going to create an example of a table to see what it offers. If we click on the “Add column” button, we are able to add a new attribute for this database. We can set the name, the type, the size and scale, a default value, the constraints that would be Primary key, Unique, Null and Index, at the end it also allows us to create check conditions to our attribute. An example of an Age attribute would be:

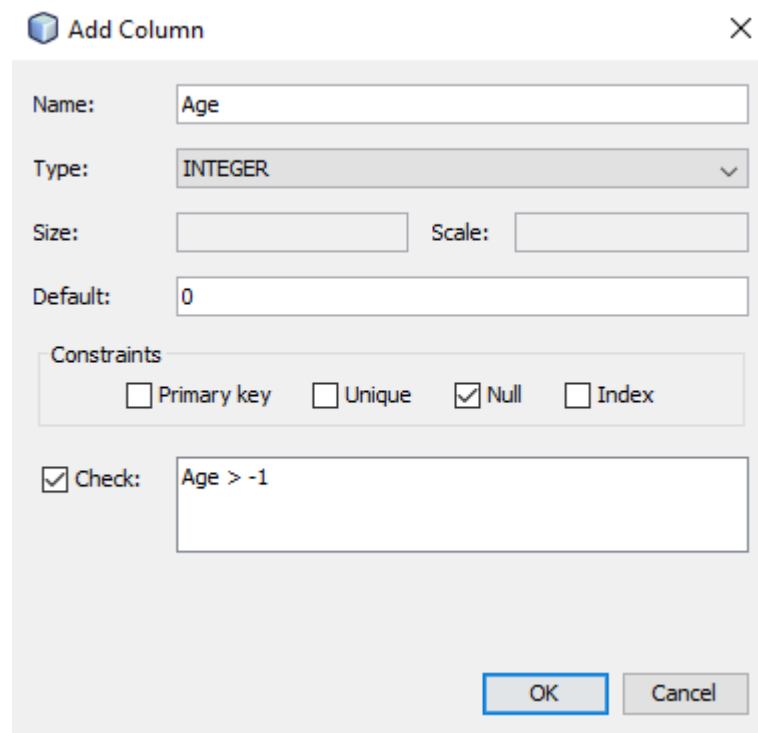


Figure 10. Create attribute.

After we created several attributes, we can click ok and end the creation of the database. An important feature of developing databases with NetBeans is that even if it usually works with java, whenever we select a database, we can see the database and its content in the bottom and in the top we can use SQL code for working with this database, so we can modify the database, filter the content and define new constraints.

Once we have a database developed, we must learn how to use from inside a project, in this case how to use it from the web service project. It is very important that we access to properties of the database and get the “Driver class” and the “Database URL”, as we need them for using the database in our code.

Once we have all set up and we want to access the database we must access the Driver class with “Class.forName(Driver class)” and create a connection to the database

with `DriverManager.getConnection("database URL", username, password)`. When this connection is established, for execute a call to the database we must follow:

1. Create an Statement as: `Statement st = connection.createStatement()`.
2. Set the query in sql language that we want to use in the database as a string.
3. Execute the query and obtain the results as `ResultSet rs = st.executeQuery(query)`.
4. Treat the results obtained.

It is recommended also to create java classes with the information of the database, then, if we get the results of the database, we can create the object of the class and treat the objects as we always do in java.

It is important to mention also in this point of the document, that the server used for the deployment of the application will be Glassfish 5. Once we have a web service created and built, Net Beans allows the user to test the webservice with their IDE, but as this is not a real-world environment, we wanted to use Glassfish 5 simulating that our computer is a server.

For the deployment of the webservice, we have to build the .war file that contains the web service and put it inside the directory of `"$Path/Glassfish5/glassfish/domains/domain1/autodeploy/"`. After this, we have to access via command line the directory `"$path/Glassfish5/bin/"` and once here, type the command `"asadmin start-domain"`. The web service will be deployed automatically, and it can be accessed via a client or via the web browser.

### **2.3. Neural networks**

Neural Networks are a type of Artificial Intelligence methods, inside the group of Deep Learning. These systems learn to perform tasks based on examples without any processing of the data and without any task-specific rule. They are like a black box that understands how to solve a task but is unable to give information about solving this task. For example, it could learn to distinguish between pictures of cats and dogs, so when you pass it a cat photo, it is able to detect the cat and classify the image as a cat but will not give a reason for this classification.

They are used for image classification, image detection, medical analysis, natural language processing, voice processing and identification and more.

An artificial neural network is a collection of connected units called artificial neurons, which are trying to imitate the neurons of our brain. Each connection between two neurons is the edge between the neurons, and it tries to imitate the synapsis between neurons. As we can see in the Figure 11, our brain has a "path" that follows the information we receive through the retina, and this information goes to a first group of neurons used for identify simple visual forms, edges and corners, the information is processed by the neurons and with the synapsis they pass the information to a second group of neurons that process again this information. Each neuron obviously has a lot of connections with other neurons. Once this information goes through several layers, it ends in a layer that identifies intermediate visual forms and group simple forms. This flow of the information through a lot of neurons in our brain ends being interpreted by another part of the brain and using the information to produce a response to what we saw in the retina.

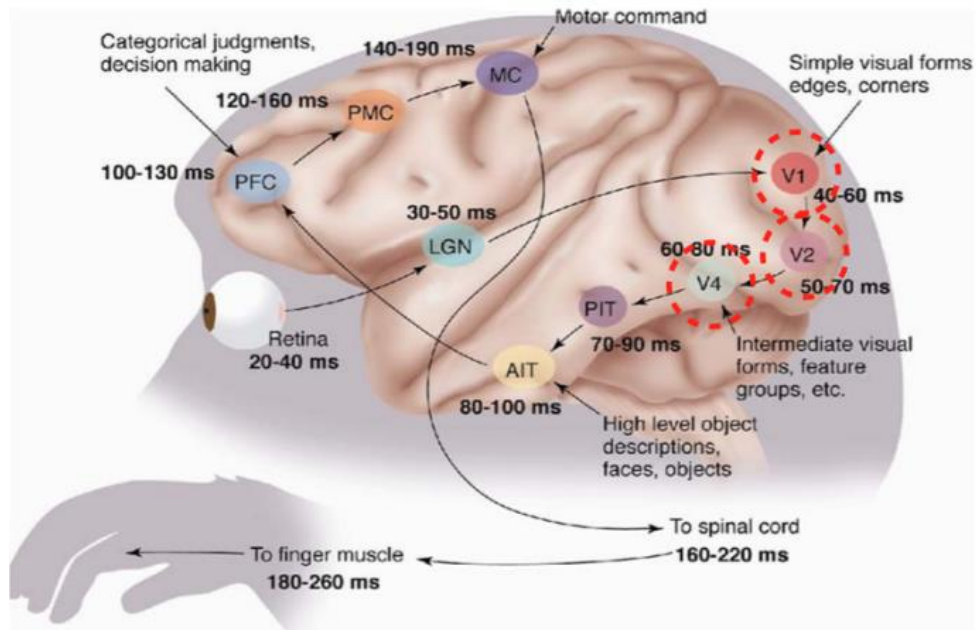


**Figure 11. Biological simil with neural networks**

A simple version of a neural network is composed by 3 layers.

- An input layer, which receives the data of the model inside the neurons, normally we will have as many neurons as input data.
- A hidden layer, which receives the output of the input layer, and produces an output.
- An output layer, which receives the output of the hidden layer and process it. It is the output of the model, usually, every output neuron will be a result, for example, if we want to classify between dogs and cat, we could have a neuron for the possibility of cat and one neuron for the possibility of dog.

Each neuron of the model has several connections with the neurons of the next layer, and a weight associated to this edge, that modifies the value that is passed from one neuron to the other. This allows the neural network to be able to distinguish between important connections between some neurons and other connections not that important. Also, each neuron has a bias value that it is always added to their value, so it adds some important value to the neurons that are crucial for solving a task. Finally, each neuron is also composed by an activation function which is a function applied over the values received from the neurons connected to it and the bias of the neuron. The typically used activation functions are sigmoid, tanh and ReLu. A neuron is represented in Figure 12.

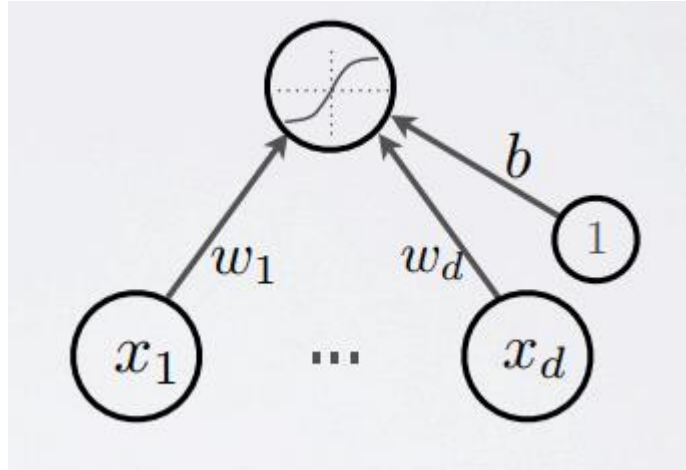


Figure 12. Example of one neuron. From Hugo Larochelle.

The preactivation of a neuron would be described as:

$$a(x) = b + \sum_i w_i x_i$$

Where the  $x$  is the neuron being activated,  $b$  the bias for this neuron and  $w_i x_i$  the weight of the connection with the neuron  $i$  times the output  $x_i$  of this neuron  $i$ .

And the activation function, which would be the output of a neuron is:

$$h(x) = g(a(x))$$

This activation function, in almost all neurons except for the output, is always the same and it is typically ReLu or Sigmoid or tanh. While the activation function used for output neurons in the output layer is typically the softmax function. This function is perfect for classification because it is strictly positive and sums to one between all the output neurons.

Obviously, a neural network can be composed by as much hidden layers as they are needed for solving the task assigned to this neural network. And there are a lot of types of neural networks, from the perceptron's used since the 50s composed by only one neuron, passing by the typical feedforward multilayer neural network to more complex neural networks like convolution neural networks.

An example of how a feed forward neural network looks is at figure 13.

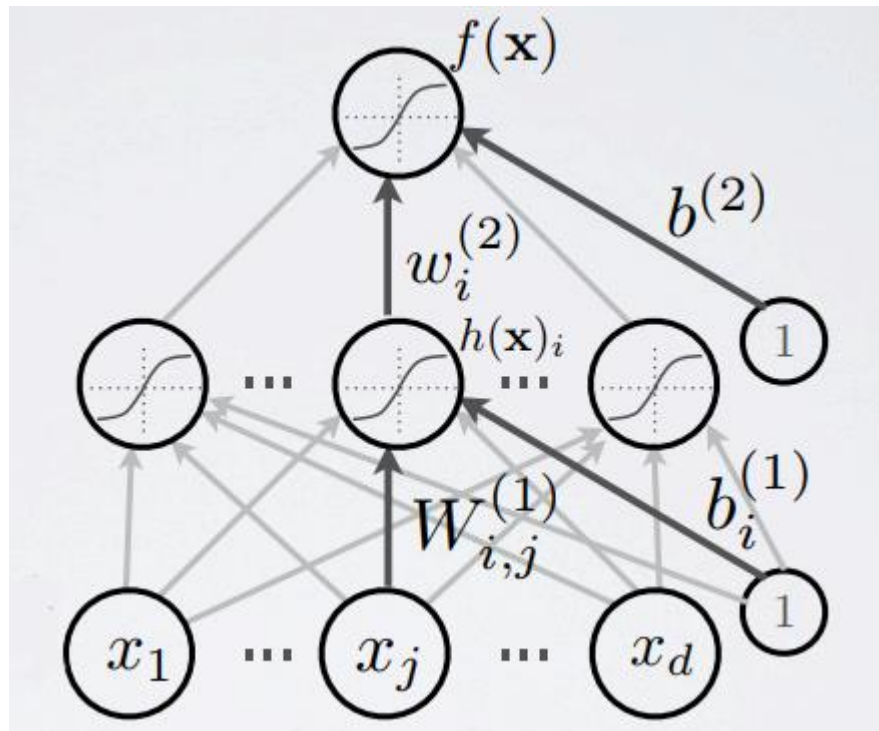


Figure 13. Example of neural net. From Hugo Larochelle.

Once we have a Neural Network created, we initialize the values of the weight and the biases and we have to train it, but we have to understand several parts of the training of a Neural Network.

Every time we feed an input into the neural network it will produce an output, at first steps this output will probably be totally wrong, so the neural network has to train over examples until it learns how to process the data. This learning is done modifying the values of the weights between the neurons and the bias of the neurons. An optimizer is an algorithm that performs updates after each example overall neural network. There are different optimizer algorithms like Ada or Stochastic Gradient Descent. To apply this algorithm to the neural network we need:

- A loss function. Which is calculated based on the difference between the expected output and the predicted output. There are several types like Mean Squared Error for linear regression or Negative loglikelihood.
- A procedure to compute the parameter gradients, based on the error of the model and the error of the previous layer.
- A regularizer for modifying the weights of the neurons. Allows the neural network to be simpler by reducing some weights of the neurons until 0, so they do nothing and the model is simpler and has less danger to overfit. Normally L1 or L2 regularization are used.
- An initialization method for initializing the values right.

Once we have set all these parameters and selected the optimizer, and when the neural network produces an output and compares it to the real output, the loss function is calculated, and the gradient from the output neurons to the hidden neurons is calculated too. The algorithm in charge of the propagation of this loss function, the error calculated and the gradient calculated for each neuron is the backpropagation algorithm. This algorithm propagates the error of the model in form of the gradient calculated to each neuron from the output layer, to the input layer, passing by all the hidden layers and

modifying the weights and biases with the objective that when the neural network processes another example, the results are a bit better until the neural network get to fully understand the problem and solve every example.

The initialization of the parameters is very important, while the biases are typically initialized to 0, the weights can not be initialize to 0 because all the gradients would be 0, can not be initialized to the same value because then all the hidden neurons in a layer will behave the same way and we need to break the symmetry. Thus, the weights are calculated in different forms and distributions always satisfying both constraints.

The final important part in the neural networks is the optimization. There are several ways of optimizing a neural network:

- Having a good separation between a train dataset, a validation dataset and a test dataset. The train dataset should be about 75 % percent of the dataset and be representative of the whole dataset. The validation dataset should be about 20 % of the dataset and it is used to select hyper-parameters. And the test set must only be used for estimate the performance of a neural network.
- Define a learning rate adequate to the problem it is being solved. This can only be done by trial and error, trying several learning rates and seeing which one performs better in the task. A big learning rate could lead to the model never finding the minimum of the loss function while a low learning rate could take forever to arrive to the minimum of the loss function or to get stuck in a local minimum.
- Early stopping of the model, once the training is working, the model can learn very good how to act in front of the training examples, but we have to control that it is still being useful and with similar results in the testing data set. This is called overfitting and can be eliminated by early stopping of the model. This is obviously related to the number of epochs trained.
- Define a good number of hidden layers and units of the layers. This can also be done trying and trying, it is recommended to start by an input and an output layer and start adding hidden layers and unit layers until the error does not improve.
- Dropout, it is another technique to avoid overfitting, in which a percentage between 20 and 50 % of the neurons of a layer deactivated, so the neural network has to develop a way of learning several ways of understanding the problem in case one of the important neurons cannot be used.

More knowledge about deep learning and neural network can be found on [18]. I learned almost everything about neural networks from this source.

We are interested in developing a neural network to compare the results with the existing rule-based model. We are going to build a neural network in java, using the library Deeplearning4j distributed by Skymind. As building a simple neural network is pretty straight forward with this library, I will not go into further explanations on how to build a neural network now. How we built this neural network is explained in point 3.3.

### 3. RetiRisk Calculator

As I have already stated in this project, the main goal is to get a simpler version of RETIPROGRAM. It has been called **RetiRisk Calculator**. This new version has the objective of being accessible to every diabetic person, with data that the user understands and knows about his health. The second objective is to make this calculator easily accessible, for this reason it will be deployed as a web service for this application, which can be called from home. In this chapter, I explain in detail the work done in this project.

#### 3.1. Study of the reduction on the number of attributes

Of all the 9 risk factors that RETIPROGRAM uses, there are several attributes that everyone knows, like Age or Sex. Other attributes like the type of treatment of DM (TTM), or the duration of DM (EVOL) known from most of the DM patients because they control them every day. But there are clearly 2 attributes of the risk-factors, that are probably not known because they are derived from blood tests analysis. These attributes, in which our study of simplification is going to be focused are the Microalbuminuria (MA) and the Estimated glomerular filtration rate, calculated from creatinine using chronic kidney disease epidemiology collaboration equation (CKD-EPI).

A renal problem can be shown with high presence of albumin or other proteins in pee. Usually it shows much more damage than only renal problems. Obviously renal problems are very related to the DM. This problem is caused by an alteration in the glomerular filtration barrier in the kidneys, and it is one of the two main interesting ways of definition and detection of chronic renal disease.

Normally is needed between 2 or 3 positive results of MA in 3 to 6 months for being sure that the patients have MA. The units are mg/24h, and a normal/healthy person has usually less than 30 mg/24h. Patients above 30 mg/24h are in danger of having chronic renal disease. This value is continue, so a patient that has 31mg/24h is way more safe than one that has 250 mg/24h. Once the value passes the barrier of 300mg/24h, the disease is called proteinuria. [15 and 16]

The definition of the fuzzy linguistic variable was done taking into account these ranges of MA. Using the value of 30 as main separation point, two labels were defined: “correct” and “high”. An intersection of the corresponding fuzzy membership functions was defined so that the transition from correct to high is gradual and not sharp.

The other main interesting way of detection of chronic renal disease is the glomerular filtration rate. [17] And a way of expressing this glomerular filtration rate is the CKD-EPI formula. The glomerular filtration is the out of liquid from glomerular capillary in the Bowman capsule, and it is a reflect of the filtration of the water and little solutes. In the renal glomeruli there are filtrated 180 liters each day between the 2 million of nephrons that each human has.

In CKD-EPI the result is expressed in  $\text{ml/min/1,73 m}^2$  and it is calculated from the creatinine of the patient. A young adult has a value between 120 or 130  $\text{ml/min/1,73 m}^2$  even if these values change in function of the age, sex, and corpulency [15 and 16].

The values used for fuzzify the CKD-EPI, are the one that are shown in table 3.

| Estadio | Descripción                              | FGe (mil/min/1,73 m <sup>2</sup> ) |
|---------|------------------------------------------|------------------------------------|
| Sin ERC | Pacientes con riesgo                     | > 90, con factor de riesgo         |
| 1       | Lesión renal* con FGe normal o aumentada | > 90                               |
| 2       | Ligera disminución de la FGe             | 60-89                              |
| 3       | Moderada disminución de la FGe           | 30-59                              |
| 4       | Importante disminución de la FGe         | 15-29                              |
| 5       | Insuficiencia renal terminal             | < 15 o diálisis                    |

Table 3. CKD-EPI table, from [15].

Based on the cut points established by the medical community (Table 3), the RETIPROGRAM has a variable with 5 fuzzy linguistic terms: Very Low, Low, Normal, High and Very High. A smooth transition between them is achieved by overlapping the corresponding membership functions.

As explained by medical experts, the values of these two variables will not be known by diabetic people, so we want to study the impact of recodifying them into a much simpler version with a single variable that can be linked to the question: “Do you suffer from renal problems”?

### 3.1.1. Data analysis for these 2 variables

In RETIPROGRAM both attributes are numerical, so they are treated as linguistic variables by means of fuzzy sets and depending on the numerical value they get a linguistic value during the fuzzification stage.

To avoid these fuzzy variables, a data study has been made on the testing dataset (with 1103 patients), to understand how these two variables influence the classification between DR and NO DR.

First, we observe the overall distribution of the patients in the two classes (Figure 14). From all these 1103 patients, there are 863 without DR and 240 with DR. This means that the 78.54 % of patients doesn't have DR while the other 21,76% does have DR. This is very important to determine the classification of a patient based on an attribute and see if this attribute is useful for our study or not.

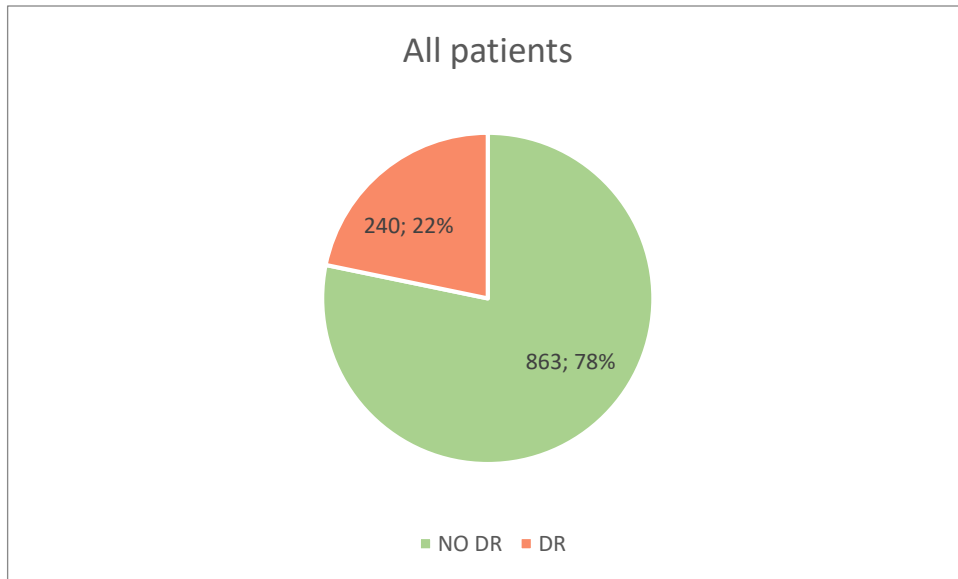


**Figure 14. All patients distribution.**

Second, we make a classification of the patients based on MA. There are 992 patients with a MA less than 30 mg/24h (“correct” category). All the other patients are in the High category (Figure 15). From all these patients that have a “correct” MA there are only 159 of them with DR, while the other 833 patients on “correct” MA do not have DR (Figure 17). From the 111 patients that have “High” MA, 81 of them have DR while the other 30 do not have the disease (Figure 16).

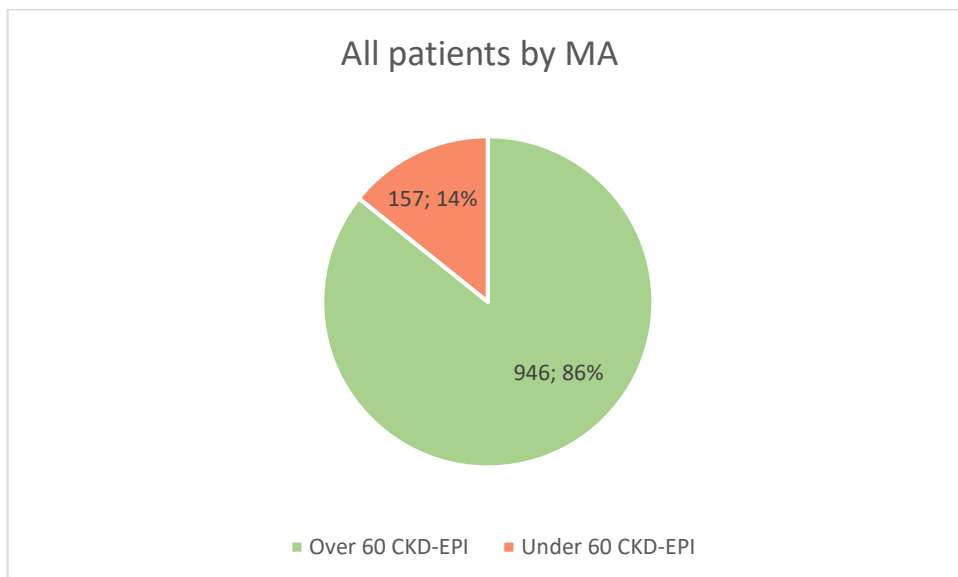


**Figure 15. All patients distribution by MA.**

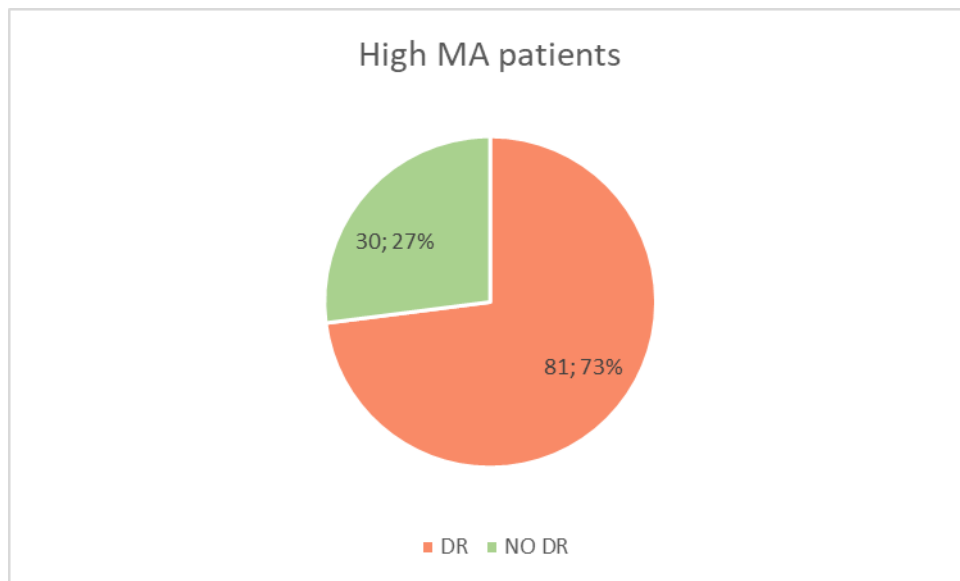


**Figure 16. High MA patients distribution.**

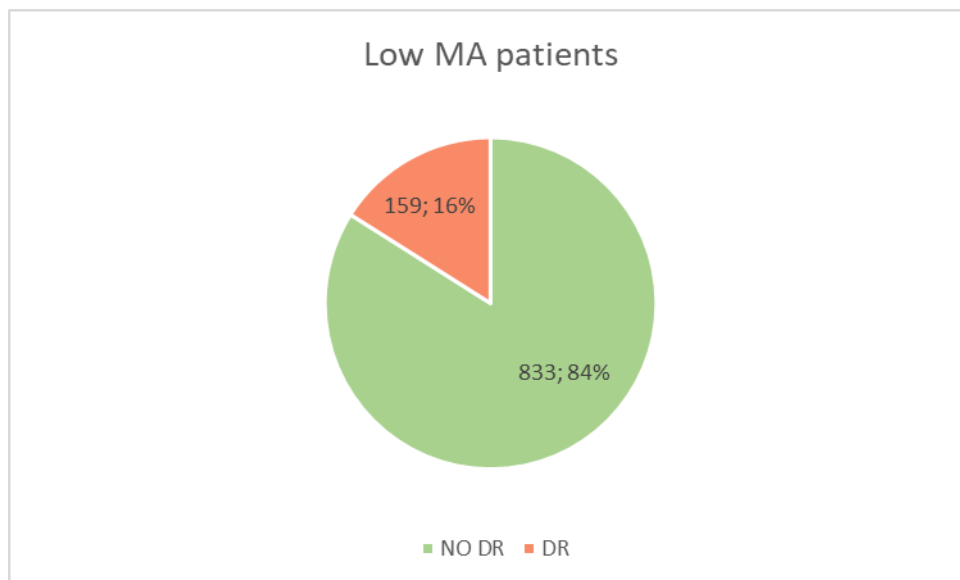


**Figure 17. Low MA patients distribution.**

Based on this data, we can extract the following conclusions.

1. If a patient has a High value of MA, it is classified as High, and then there is about a 73% of possibilities for developing DR. This 73% is very high considering that without this division for High MA a patient has a 27% of possibilities of developing DR.
2. If a patient has a correct MA, there is a big chance for him of no developing DR. The correct MA adds an extra 6% of security for a patient that it won't have DR.
3. From all patients, there is a lot that have correct MA, so when someone has a High value, it probably means some risk for DR or other diseases.

We consider that this attribute can be binarized in a simple way, if we set a barrier at 30 mg/24 h. In that way, we can create a categorical variable with two possible values: lower than 30mg/24h is assigned to label “correct” (i.e. value 0), whereas values higher than 30 mg/24h are assigned to label “high” (i.e. value 1). We hypothesize that using this

categorical variable instead of the fuzzy one will give us a similar accuracy in the DR classifier.

After this, I have tried to make a classification of patients based on their CKD-EPI, the most important in this binarization of the attributes is to find a threshold that separates patients with and without DR using this variable.

We have tried with 2 threshold values in CKD-EPI. As it has been stated in the previous section, a normal person has about 120 or 130 ml/min/1,73 m<sup>2</sup>. In addition, as it can be seen in the table 3, a person with over 90 is not considered to have chronic renal disease, while patients under this value already have kidney problems. Due to this, the first study we do for CKD-EPI it is with a threshold of 90.

There are 396 patients with a value of 90 or above, the ones that belong to VeryHigh category in the linguistic variable. From these 396 patients, 81 have DR and 315 do not have DR (Figure 19). There are 707 patients with a lower value of 90 for CKD-EPI, from these 707, only 159 have DR and 548 does not have DR (Figure 20).

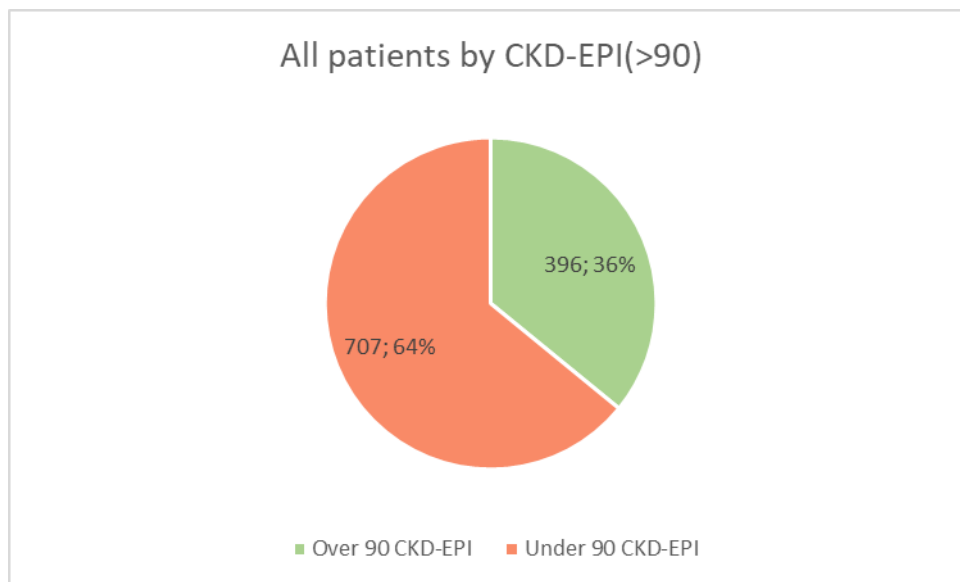


Figure 18. All patients distribution by CKD-EPI with value 90.

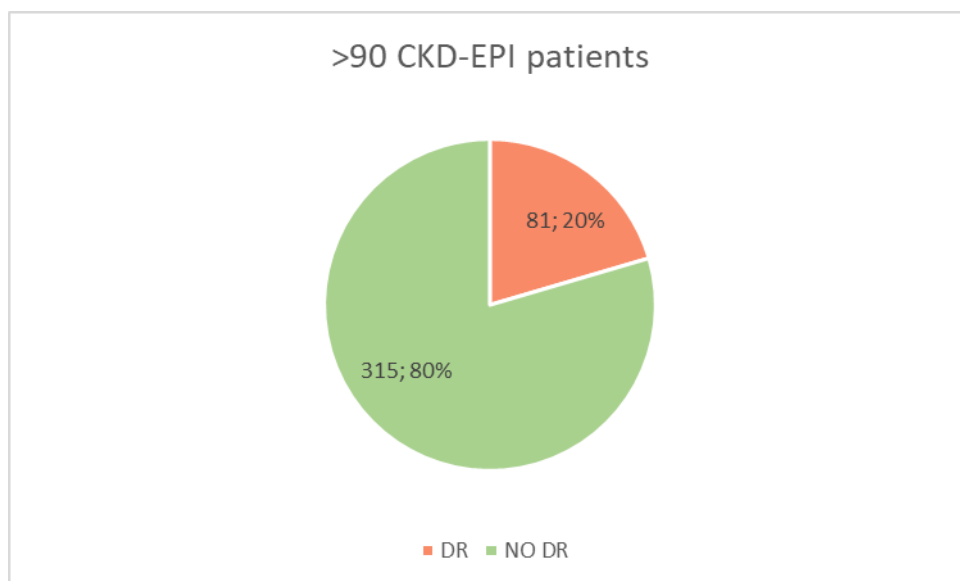


Figure 19. Over 90 CKD-EPI distribution.

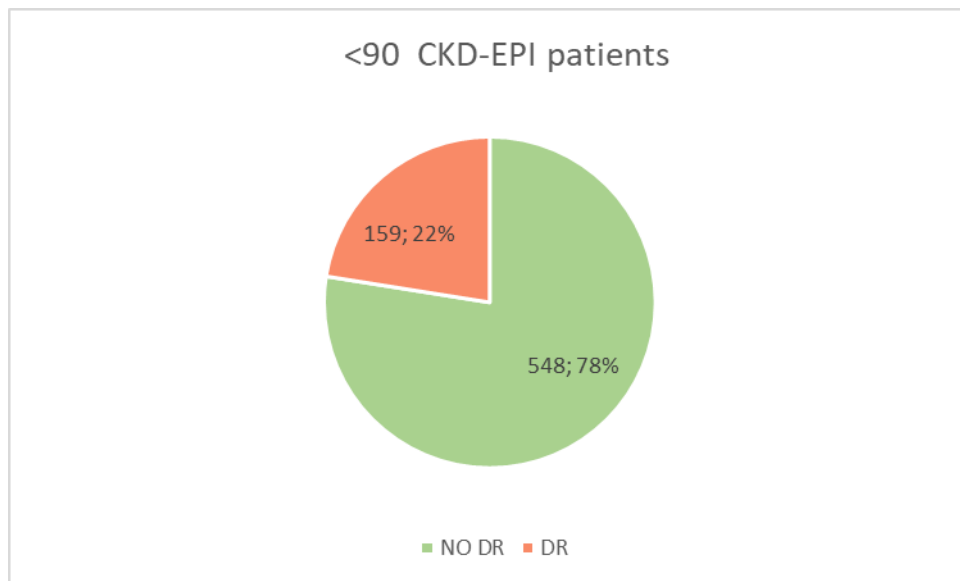


**Figure 20. Under 90 CKD-EPI distribution.**

Based on this analysis, we can get the next conclusions.

1. Most of the patients in our study have over >90 of CKD-EPI, exactly 64%, so there are 36 patients with low chances of having renal chronic disease.
2. On both classifications, for patients under and over 90 of CKD-EPI, the distribution is almost the same, which is very similar in percentages to the normal distribution of patients without taking in account any attribute.
3. This 90 threshold for CKD-EPI does not allow to make any good study.

Following the recommendations of the medical team, we studied a second threshold for CKD-EPI, because it is considered interesting for chronic renal disease. In that way, we want to confirm that it has some influence in DR patients' classification.

As can be seen in the table 3, patients with under 60 for CKD-EPI value have a moderated decrease in glomerular function, this may be more important to DM patients, as if their glomeruli do not work, their blood is filled with more proteins and the little vessels on the back of the eye could explode easily.

There are 946 patients that have a higher value than 60 CKD-EPI, while there are 157 patients with a lower value than 60, which means that their glomeruli aren't working fine (Figure 21). From the 946 patients, there are 177 with DR and 769 without DR (Figure 22). Inside the 157 patients, there are 63 patients with DR and 94 without DR (Figure 23).

With this study, we obtain the next conclusions:

1. A lot of patients have a CKD-EPI value over 60, which means that their glomeruli are still working fine.
2. From the patients that have over 60 of CKD-EPI, the distribution is a bit better than not using any attribute for the classification of patients. With a rate of 81%-19% with CKD-EPI and 78%-22% without attributes. It is also a bit better than using the patients with over >90 of CKD-EPI.
3. Patients that have a value under 60 of CKD-EPI, have a 40 % of possibilities of developing DR, this is a higher percentage than not using any attributes.

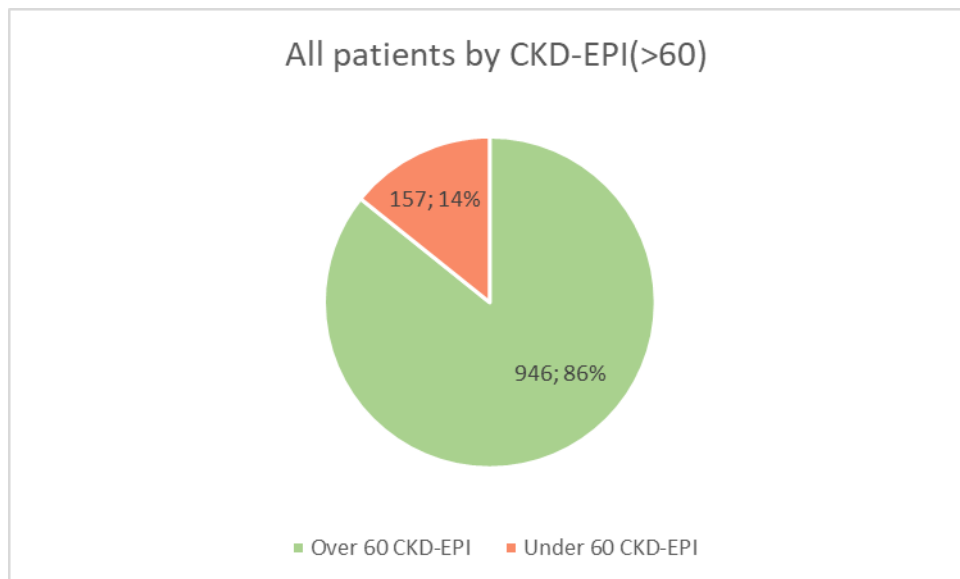


**Figure 21. All patients distribution by CKD-EPI with value 60.**

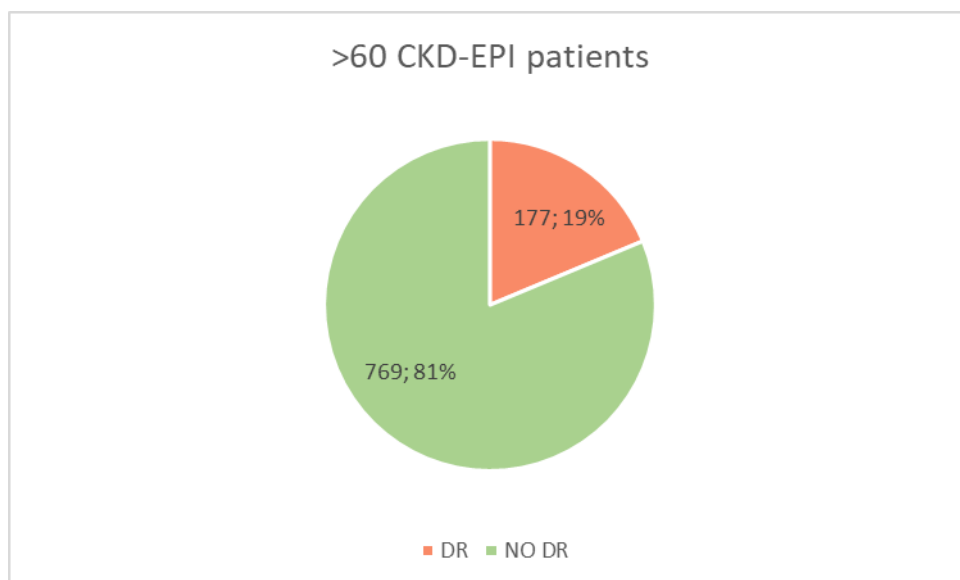


**Figure 22. Over 60 CKD-EPI distribution.**

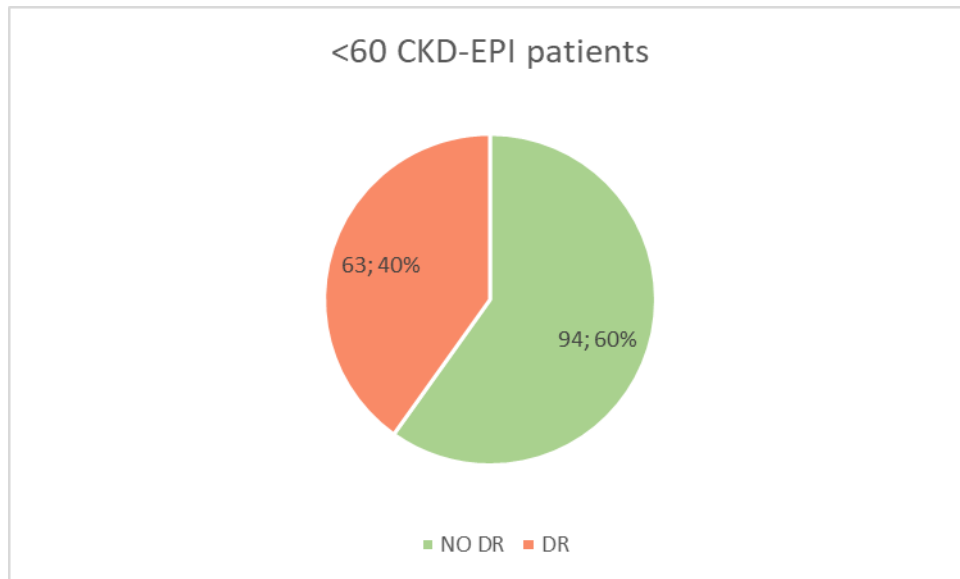


**Figure 23. Under 60 CKD-EPI distribution.**

After this study and its discussion with the medical experts, we chose the threshold level of 60 for CKD-EPI. Every patient with a value higher than this 60 will be set to a number of CKD-EPI, while the other patients with a value under 60 will be set to a number of CKD-EPI lower than 60.

After finding the appropriate threshold of these two variables, we have made a study to understand how both attributes are related to each other and what can we expect after we binarize them both. This could determine if these attributes can be grouped together in one attribute for the simpler version of RETIPROGRAM.

For the 946 patients with a value of CKD-EPI higher than 60, there are 892 patients with a correct MA (Figure 24). These are the patients that have both attributes in a good state, so they shouldn't develop DR, at least not based on their kidney problems. From these 892 patients, 745 do not have DR while the other 147 patients have DR (Figure 25).

Again, for the 946 patients with a value of CKD-EPI higher than 60, there are only 54 with High MA. From these 54 patients, there are 30 with DR and 24 without DR (Figure 26).

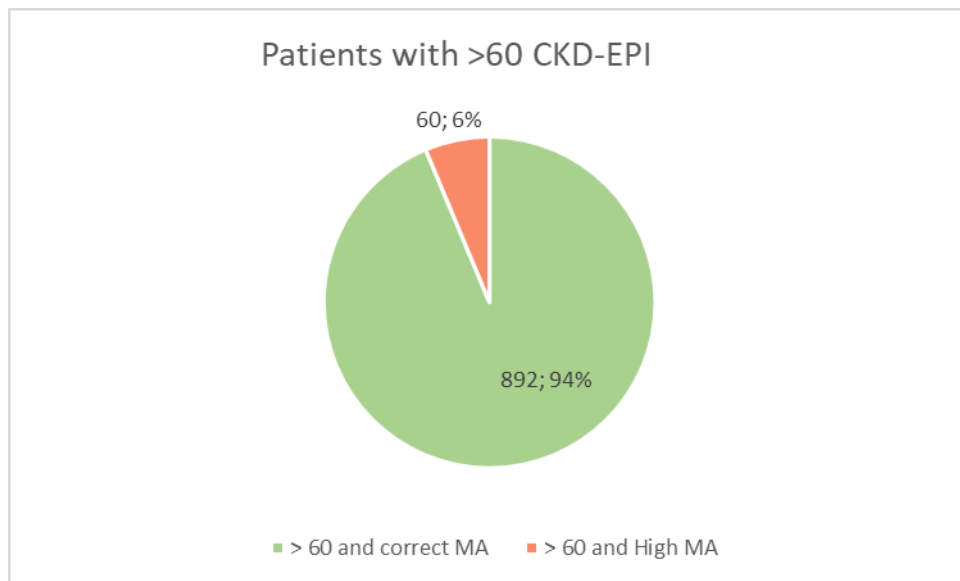


**Figure 24. Distribution of patients (based on MA) with over 60 CKD-EPI.**

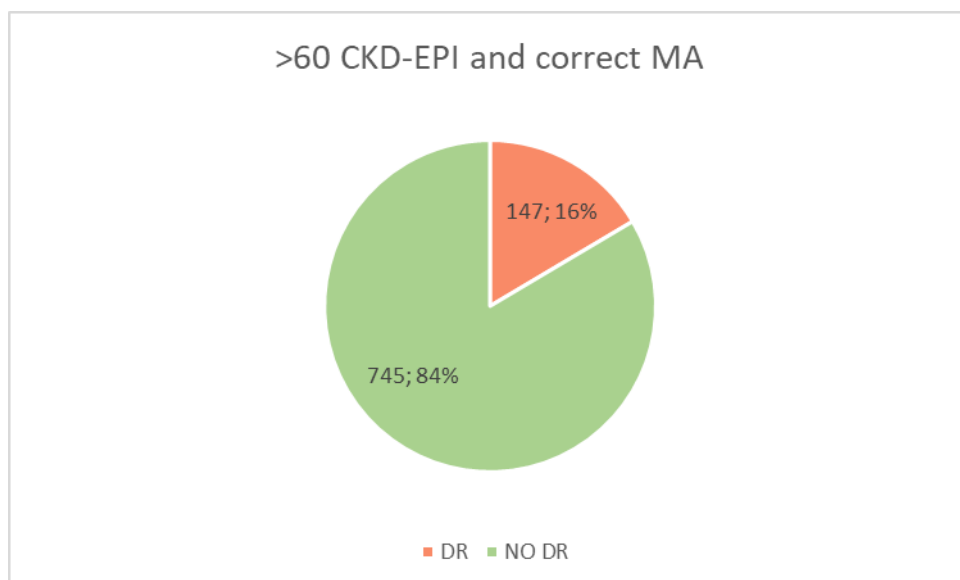


**Figure 25. Distribution of patients with over 60 CKD-EPI and correct MA.**

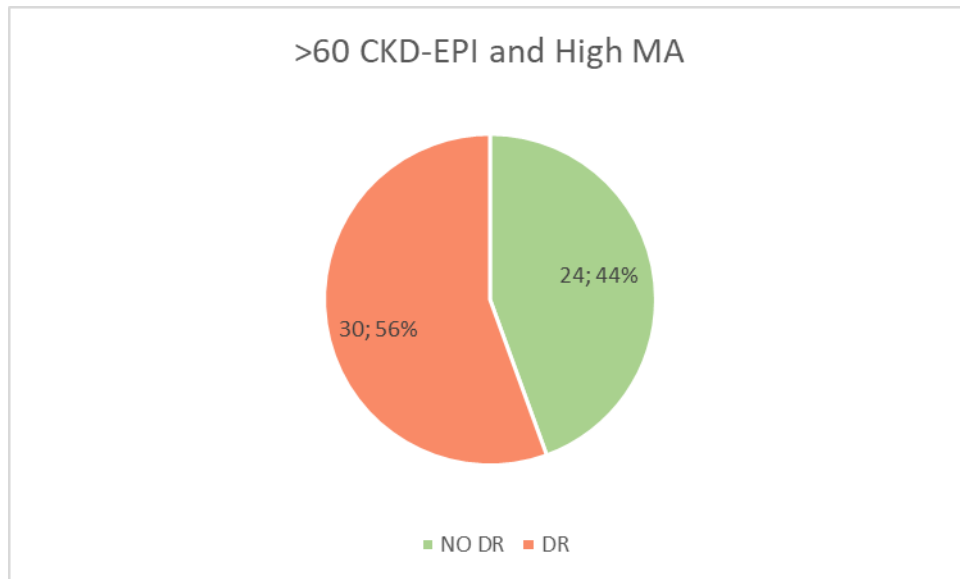


**Figure 26. Distribution of patients with over 60 CKD-EPI and high MA.**

We have obtained these conclusions:

1. Almost every patient that has CKD-EPI in a good range with at least 60 value or higher, has MA with a correct value, so that means some obvious correlation between both attributes.
2. A patient with both values in a good range (over 60 and correct MA) have the same distribution than MA alone has: 84% no DR vs 16 % DR. 6 points higher than not using any attribute. If a patient belongs to this group of patients, if he develops DR it won't probably be based on this risk factors.
3. In the patients that have a high value for MA, even with CKD-EPI being over 60, there is a good chance of 56% of developing DR. This determines that clearly, MA is way more important for DR than CKD-EPI.

When we look at the 157 patients with a value lower than 60 of CKD-EPI, there are 100 patients with a correct MA (Figure 27), from these, only 12 have DR and 88 do not have DR (Figure 28). Within the other 57 patients, with both values in a bad way (under 60 CKD-EPI and High MA), we see that there are 51 patients with DR and 6 lucky patients that do not have DR even if they have kidney problems (Figure 29). In this case, the conclusion for these people are:

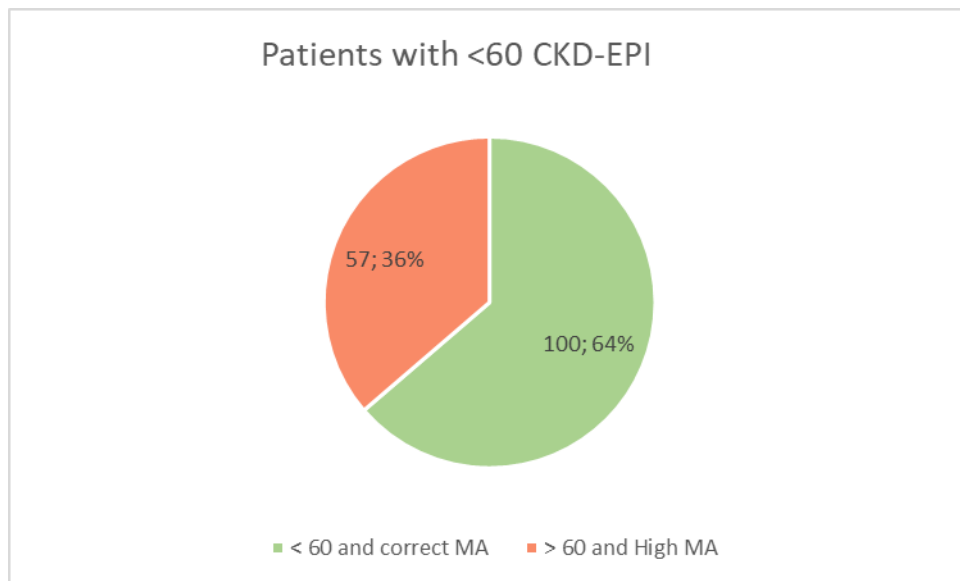


**Figure 27. Distribution of patients (based on MA) with under 60 CKD-EPI.**

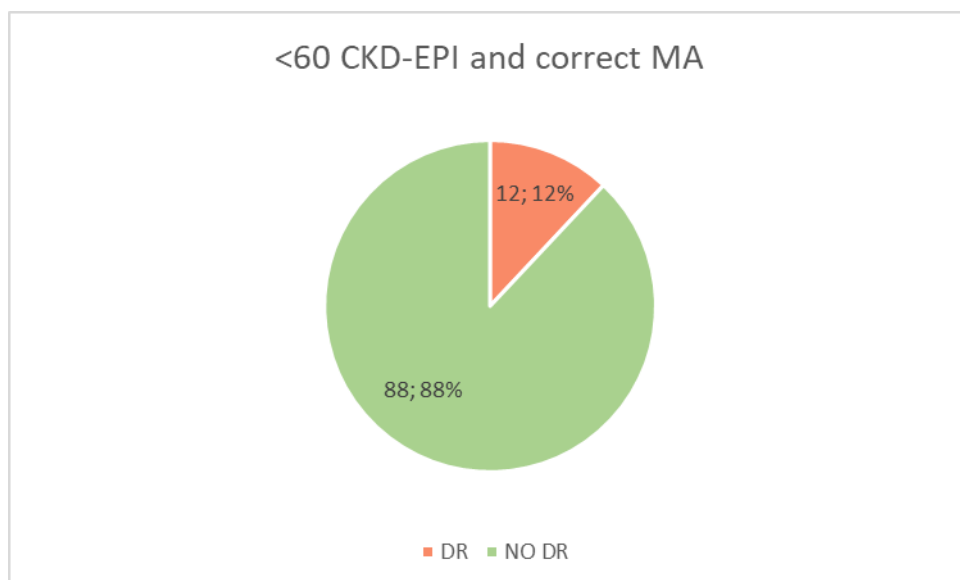


**Figure 28. Distribution of patients with under 60 CKD-EPI and correct MA.**

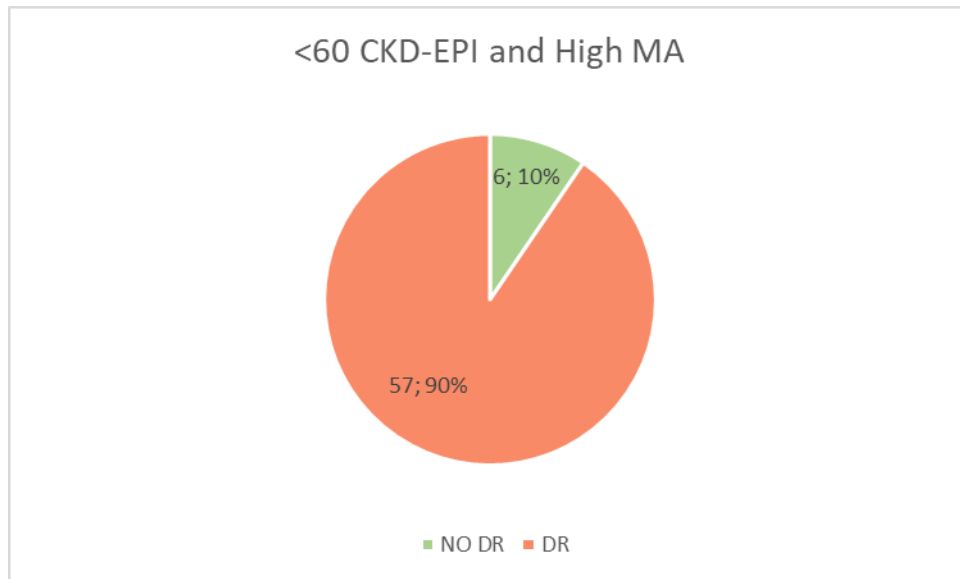


**Figure 29. Distribution of patients with under 60 CKD-EPI and high MA.**

Between patients that have a value lower than 60 of CKD-EPI, there is a 36% that also has High MA, while 64% only have CKD-EPI. This added to the previous results of over 60 CKD-EPI and MA, means that when a patient has correct MA, he probably does not have CKD-EPI problems, while if the patient has High MA there is a possibility that it has a bad value of CKD-EPI, and if it has a value under 60 of CKD-EPI it probably means that it already has a High MA. All of this seem to indicate that the first alert for DR related to kidney problems is MA, which comes followed by CKD-EPI.

1. Patients with correct MA and under 60 of CKD-EPI probably will not develop DR, which support the idea of MA being the strongest attribute of this two attributes and the idea that bad signs of MA appear before CKD-EPI.
2. Patients with both attributes in a bad value, are not designated to develop DR, there is still a chance of 10 %. This also means that based on these 2 attributes, a Fuzzy Decision Tree could develop a rule like:

IF CKD-EPI < NORMAL AND MA == High THEN DR (with a 90% of rule confidence).

### **3.1.2. Binarization of the dataset**

In the previous section, we have seen how these 2 attributes work separated and when we put together, and we have seen that both taken into account together as a single attribute could be fine for a simpler model that a normal patient of DM could use without electronic health records.

The actual problem is that even if we have seen a threshold for both attributes, we are not sure if after we make a model with an attribute that sums both attributes it will have good results. It is important to see a comparison between the actual RETIPROGRAM and the simpler version RetiRisk-Calc, for understanding how the simplification works.

First, we have seen how RETIPROGRAM works with the attributes binarized. The binarization of the dataset has been done using both thresholds defined:

- 30 mg/24h for MA.
- 60 ml/min/1,73 m<sup>2</sup>

For MA, we changed the original values of this variable by only two values: value 0 for patients under 30 and value 100 for people over 30. In that way, the fuzzification of these two values will lead to a membership of 1 on any of the two possible labels, not using the uncertainty range. So, the rules will activate with maximum level on this variable, as if it was categorical.

For CKD-EPI we tried 5 substitution configurations, so we can see several combinations of rules being activated and see which rules are more important for the correct classification of patients.

1. 89 for patients over 60 of CKD-EPI so the rules of High and VeryHigh are activated. 29 for patients under 60 for activating both rules Normal and Low.
2. 89 for patients over 60 of CKD-EPI so the rules of High and VeryHigh are activated. 15 for patients under 60 for activating both rules Low and VeryLow.
3. 100 for patients over 60 of CKD-EPI so the rules of VeryHigh are activated. 29 for patients under 60 for activating both rules Normal and Low.
4. 100 for patients over 60 of CKD-EPI so the rules of VeryHigh are activated. 15 for patients under 60 for activating both rules Low and VeryLow.
5. 100 for patients over 60 of CKD-EPI so the rules of VeryHigh are activated. 45 for patients under 60 for activating Normal rules.

The evaluation and results obtained using these new binarized versions of the variables will be given in Chapter 4.1.1. As main observation, the model with the best results is the one with a binarization on MA between 0 and 100 and a binarization on CKD-EPI with 15 and 100.

### ***3.1.3. Create a simpler attribute based on both MA and CKD-EPI***

Once we have these results, we know how RETIPROGRAM performs when the patients are divided between these 2 attributes, but we don't know what the best way is for grouping both attributes. As we can see there is much more influence in the MA than the CKD-EPI, then, for making a new attribute we must consider this.

For understanding the best combination of the attributes, I trained several models of RETIPROGRAM with the modified dataset.

We tried to train the model:

1. With normal non binarized data
2. Binarized data as it has been stated above with each attribute treated separated
3. Binarized data that whenever one of the attributes has the "bad" value for this attribute both attributes are assigned the "bad" values. So, if any patient has a value of under 60 in CKD-EPI or higher than 30 in MA, the values assigned are 15 for CKD-EPI and 100 for MA.
4. Binarized data that only whenever CKD-EPI is under 60 CKD-EPI is assigned as 15 and MA as 100. Ignoring MA data.
5. Binarized data that only whenever MA is higher than 30 CKD-EPI is assigned as 15 and MA as 100. Ignoring CKD-EPI data.

The results obtained are in the results part of this document in chapter 4.1.2, after consensus with doctors the binarize combination of both attributes for this work was the 5<sup>th</sup> one, which is the logical one when you look at the data analysis realized.

As we have seen, MA and CKD-EPI are both related to renal chronic disease and bad working of the kidney. Even if patients do not know the values of these attributes because they do not have the knowledge for understand them, they clearly know if they

have kidney problems, because as we have seen is one of the most important things to control for DM patients. Therefore, the simpler version RetiRisk Calc will use a new attribute called “Kidney”.

This attribute refers to having kidney problems or no, it is configured as a Categorical linguistic variable. A person using this new version should only indicate that it doesn’t have kidney problems, then the linguistic variable would be assigned to “Good”, or that it has kidney problems, then the linguistic variable would be assigned to “Bad”. To run the training and testing algorithms, the definition of the variables has been changed: MA and CKD-EPI are eliminated as we add the definition of the new one as seen in the figure 30.

```
#LV Kidney Categorical {0,1}
Good 0
Bad 1
```

Figure 30. Linguistic variable conversion of Kidney.

### 3.1.4. Obtaining RetiRisk Calc offline version

Now that we have made the whole study and established a way of simplifying RETIPROGRAM, we face the problem of developing the best classification model with the new variables, to be integrated in the RetiRisk Calc.

First, we have to train different models with binarized data based on MA values. While we change the values of the different parameters used in the FRF, so we can see which are the best parameters that fit our problem. This is done because we are really interested in having a model that classifies as much patients as possible, so that almost every patient is classified, even if some of the patients are not classified right. We need to classify everyone because the main public of this new application is to be used by people on their home or optic clinics, so it should give a result, even if it is not that right.

This will obviously penalize the results of our new model. Another important goal of this application is that it should minimize the false negatives. Otherwise, false positives are not so critical, not because we want to alarm people, but because we consider that a mistake that forces an unnecessary visit to the doctor is much less critical than making mistakes with people that really have DR and they are told by the application that they are fine.

Obviously in the regular RETIPROGRAM, as it is used by doctors, this is not needed, because only the doctor will see that there is not enough information for being sure about a diagnostic and he will be able to use his knowledge.

Once we generated all of these models, we tested them on a program that uses only the model generated and stored and the testing data. This program can’t create a model, so every fuzzy decision tree and every rule generated must be stored in the model file (which is a serializable Java file). These results are in the chapter 4.1.3.

It is important to separate in two programs the training and classification steps. RetiRisk Calc application will have only the classification step. It won’t be used for training. In the future, if we make new changes to the code for training, we will be able to generate a new file model that can be easily substituted in the classification application.

All the results obtained from the generated FRF and which version with different parameters is better is in the results and evaluation point of this document.

Finally, the program should be encapsulated in a JAR file. I had to perform two tasks:

- To group the Java source code with the resources inside of this JAR file. In particular, the FRF model and the variables definition file are inside the JAR.
- To change paths of the project to facilitate the input/output of the data. The test data set must be in the same folder as the jar and the results files will be stored in the same folder too.

It is very important to mention, that I had to make some changes to the program because some more information was needed to be stored in the output file. In particular, we implemented a way of obtaining both the result that each tree gives to each patient and the class membership obtained, while RETIPROGRAM only was obtaining the result of each tree. This required a modification of the Java code that enable to follow the next few steps:

1. Store in a matrix of  $k$  trees and  $r$  patients, the classification of the patient, DR, no DR or Unknow.
2. Store in another matrix of  $k$  trees and  $r$  patients, the membership value to the class decided by the tree, between 0 and 100.
3. Run over all patients, and for each patient, run over all trees. If the matrix in the position  $k$  tree and  $r$  patient has 0, the counter for no DR it is added by 1, and the counter for uncertainty of belonging to no DR it is added by the membership function stored in the second matrix for this patient and tree. If the matrix in the  $k$  tree and  $r$  patient has 1, the same but with DR is done, and if there is not a 0 or a 1, the same happens with the unknown counter.
4. Once all of the trees for a patient have been consulted, the code proves that the threshold  $\delta^2$  is less than the difference between the class chosen and the other class.
5. It calculates the percentage of the uncertainty of the class chosen and returns the results of the classification for this project.

Another modification we made for our final RetiRisk Calc that it is worth to mention, is that we changed the different files that are produced. RETIPROGRAM, as it is designed it is intended that it can process several patients and the same time in case a doctor wants to check information on a lot of patients, but this new simpler application is born with the aim of being used by people over the web, so it will only process one patient at the time, and the information of the results that we need to obtain has to be limited.

The first file produced by RETIPROGRAM are the results of all the patients in the dataset, so we can obtain how the model performs when we train it and how it varies with different parameters. All of the parameters that are used and all the scores that we use for considering if we have a good model or not are explained in the results part of this document.

Obviously, in our new application there is no need of using this file for results, so we deleted it from the code of the test project.

The second file produced is a file where each row is the class of each patient and some information of the parameters of the model, as well as the real class of the patient.

This file is more interesting for us and I modified it for just obtaining the predicted class and the uncertainty. Moreover, as our model will work with patients that are not already classified, we cannot print the real class of the patient.

An example of the output file with the results of RetiRisk Calc is:

| PredictedClass | Certainty |
|----------------|-----------|
| 1              | 65        |
| 0              | 75        |
| 1              | 83        |

**Table 4. Example of output file of RetiRisk Calc.**

In summary, the construction of the program RetiRisk Calc required the following actions:

1. We generated the new dataset from the original one with the new attribute Kidney, which is categorical, and every patient should know about the state of health of their kidneys. We generated this for both training and testing dataset.
2. We trained a model with the RETIPROGRAM code, but with the changes that we made for using now only 8 attributes and with the new fuzzification file. We also implemented a way of storing all of the parameters of the model inside the model, as before, everything in the test project had to be set again with the values used for training.
3. We obtained the results of this training model, which are in the results part of this document in chapter 4.1.4.
4. We introduced the model generated in the training inside the test project, and we generated the jar file containing the model and fuzzify file. Which produces the result file that we have indicated above.
5. We tested that the jar generated works by creating a new dataset of a patient with information made up and putting it in the same directory of the jar file. We then executed the jar file and we obtained a results file as the one showed.

### **3.2. Design of our web service for RetiRisk Calc**

After the implementation of a Java program RetiRisk Calc that works offline in a single computer, we approached the goal of deploying it as a web service in order to be called from a Web browser in any device that a diabetic person had at home or optic clinic. That's why we need to develop a web service which embeds the offline RetiRisk Calc.

Our web service is composed by the configuration class (Figure 31), which is on the path "webresources". It adds the resources of our project in the overridden getClasses method, the classes added are RD.PatientController and RD.WebRD.

```

package RD;

import java.util.Set;
import javax.ws.rs.core.Application;

/**
 *
 * @author manurubo
 */
@javax.ws.rs.ApplicationPath("webresources")
public class AppConfig extends Application {

    @Override
    public Set<Class<?>> getClasses() {
        Set<Class<?>> resources = new java.util.HashSet<>();
        addRestResourceClasses(resources);
        return resources;
    }

    /**
     * Do not modify addRestResourceClasses() method.
     * It is automatically populated with
     * all resources defined in the project.
     * If required, comment out calling this method in getClasses().
     */
    private void addRestResourceClasses(Set<Class<?>> resources) {
        resources.add(RD.PatientController.class);
        resources.add(RD.WebRD.class);
    }
}

```

**Code 1. Configuration class of the web service.**

Both classes added are the classes that have the resources of our web service. In the WebRd we have the main functionality of the web service, it performs the operation of adding a new patient to the database and execute the RetiRisk Calc on this patient, and it shows the id and the prediction of the patient. The other operation on the WebRd class is the same than the previous one, but for patients already in the system, so the patient passes its id to the web service along with his information for the application, the output it's the same.

Both classes, with the difference of the id, work with this algorithm:

1. Detection of the resources file. Create the testing file and introduce the data of the patient that are received via the @QueryParam label in the method.
2. Executes the jar file in which we have the RetiRisk Calc. The results are created in the same in resources file in the file "RulesResults.csv".
3. A connection to the database is created.
4. A query to the database of patients is realized to insert the patient in the database.
5. The "RulesResults.csv" is read and this information is stored in the database results database.
6. The results of this analysis are showed via HTTP to the client.

The paths used for accessing this classes are:

- <http://localhost:8080/WebRD/webresources/evaluation/newpatient?Age=x&Sex=x&EVOL=x&TTM=x&Kidney=x&HbA1c=x&HTAR=x&BMI=x>
- <http://localhost:8080/WebRD/webresources/evaluation/alreadypatient?id=x&Age=x&Sex=x&EVOL=x&TTM=x&Kidney=x&HbA1c=x&HTAR=x&BMI=x>

As both operations are used for getting an output, they are both labelled as @GET and they both are labelled as @Produces({MediaType.TEXT\_PLAIN+";charset=utf-8"}) for the output.

The other class is PatientController, this class has the functionality of retrieving the information of a patient from this id, in both databases: the patients database and the results database. If a patient has used only once the program, it will return only one patient data and one results data. In case the patient has used the web services several times, we have on the databases as many entries as uses of this patient. Then, this method will show every entry in both tables, with this a patient is able to see its development of information and predictions of the application along the time, and see if he is getting worse or is still safe from DR.

The algorithm for this method is:

1. Create a connection to the database of patients.
2. Execute a query to the database that selects all patients with the id passed by the patient in the path, this id is labelled as @QueryParam in the method.
3. It creates a patient for each iteration of the results obtained from the database and it stores the toString of the patient in a string variable.
4. Creates a connection to the results database.
5. Executes a query to the database that selects all patients with the id passed by the patient.
6. It creates a result for each iteration of the results obtained from the database and it stores the toString of the patient in a string variable.
7. Return the string variable, which prints via HTTP the information of the patient and the results obtained.

The path used for accessing this method is:

- [http://localhost:8080/WebRD/webresources/patients/get\\_patient\\_by\\_ID?id=x](http://localhost:8080/WebRD/webresources/patients/get_patient_by_ID?id=x)

As the operation is performed for obtaining the results in the database, it is labelled as a @GET resource and the result that produces is @Produces({MediaType.TEXT\_PLAIN+";charset=utf-8"}).

The classes used to work easier with the information received from the database are Paciente and Resultado, even if this classes are not resources they make the management of the web service easier, and in case new functionalities are added these classes would be very useful.

The class Paciente is a simple java class that stores the attributes of a patient in the database, these attributes are: Age, Sex, TTM, EVOL, HbA1c, Kidney, BMI, HTAR and fecha. It only has getters and setters and a toString() method.

The same happens with the Resultado class, it is just a class that stores of a result obtained from the results database, the attributes are RD, Certainty and fecha. It only has getters and setters and a toString() method.

### **3.2.1. Class diagram.**

The class diagram associated with our web service project is:

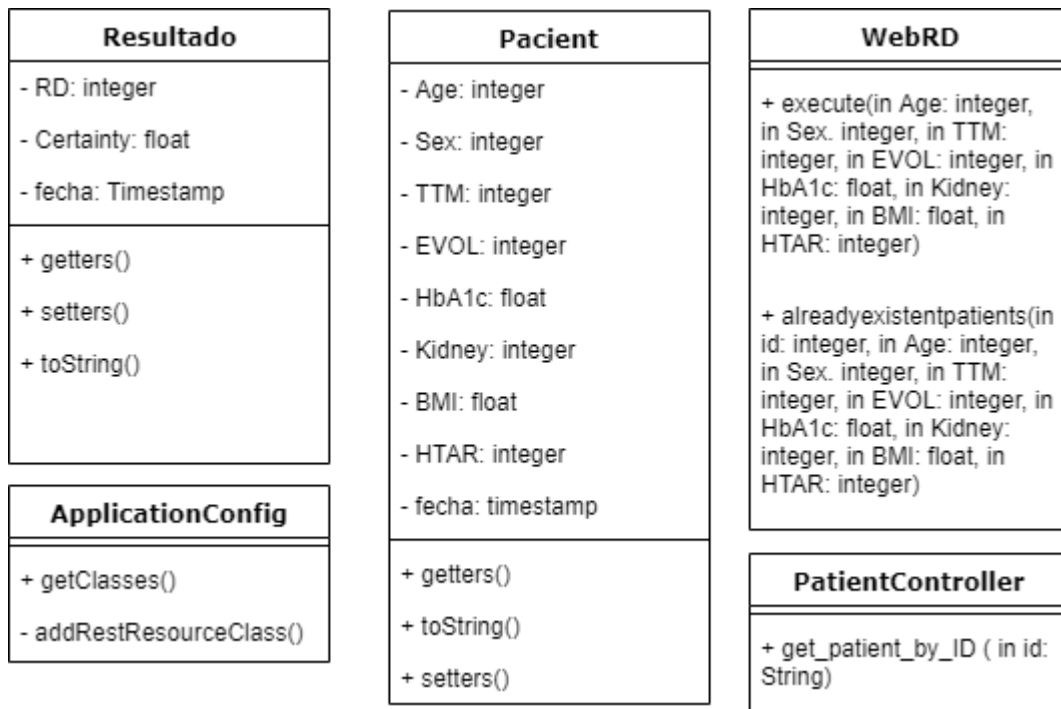


Figure 31. Class diagram.

### 3.2.2. Database design

Both of our tables have been created in NetBeans with the procedure we showed in the section 2.3, but as we want to represent them here, we are going to expose how they would be created in SQL code.

```

CREATE TABLE patients
(id INTEGER NOT NULL,
  age INTEGER NOT NULL CHECK (age >= 0),
  sex INTEGER NOT NULL CHECK (sex == 0 OR sex==1) ,
  evol INTEGER NOT NULL,
  TTM INTEGER NOT NULL CHECK (ttm == 0 OR ttm==1 OR ttm ==2),
  HbA1c FLOAT NOT NULL,
  Kidney INTEGER NOT NULL CHECK (kidney == 0 or kidney==1),
  BMI FLOAT NOT NULL,
  HTAR INTEGER NOT NULL CHECK (htar== 0 or htar==1),
  fecha TIMESTAMP NOT NULL,
  PRIMARY KEY (id, fecha),
)
CREATE TABLE results
(id INTEGER NOT NULL,
  rd INTEGER NOT NULL CHECK (rd == 0 OR rd==1) ,
  Certainty FLOAT NOT NULL,

```

```
    fecha TIMESTAMP NOT NULL,  
    PRIMARY KEY (id, fecha),  
)
```

### 3.2.3. *Hosting*

Once we have obtained our RESTful web service for the online RetiRisk Calc. We need to use a hosting service so the company that wants to use the web service, or the patients from home can access the web service. All of the test we have done are on local, with the test deployed on our computer. But at the end, we need a hosting, so everyone can access the web.

We already explained how the web service is deployed, and that we need glassfish to deploy it, the same has to be done if you deploy the web service on your computer for tests or on a hosting service, except if the hosting web services indicates the opposite.

The hostings we have been looking at are:

- DigitalOcean.
- Scaleway.
- Arubacloud.

### 3.3. **Implementation of a feed forward NN for DR classification**

Although the current version with Fuzzy Random Forest works quite well, we were interested in comparing the performance with a different approach using a different method of Machine Learning. We have developed a neural network in Java with the library DeepLearning4j. As said in Chapter 2, neural networks are widely used nowadays and they have been able to outperform other AI techniques.

Our NN is a feedforward neural network, pretty simple but effective, it is composed by 5 layers (one input layer, three hidden layers and one output layer). All of these layers have been initialized with the XAVIER distribution and the activation function for each layer except the output layer is the ReLu function (Figure 33).

Xavier distribution is based in that at each passing layer we want the variance of the weights of this layer to remain the same than the variance of the previous layer. The weights picked initially come from a gaussian distribution with zero mean and a variance of  $(1/N)$ , where N is average nodes between the nodes of the input and the nodes of the output [19].

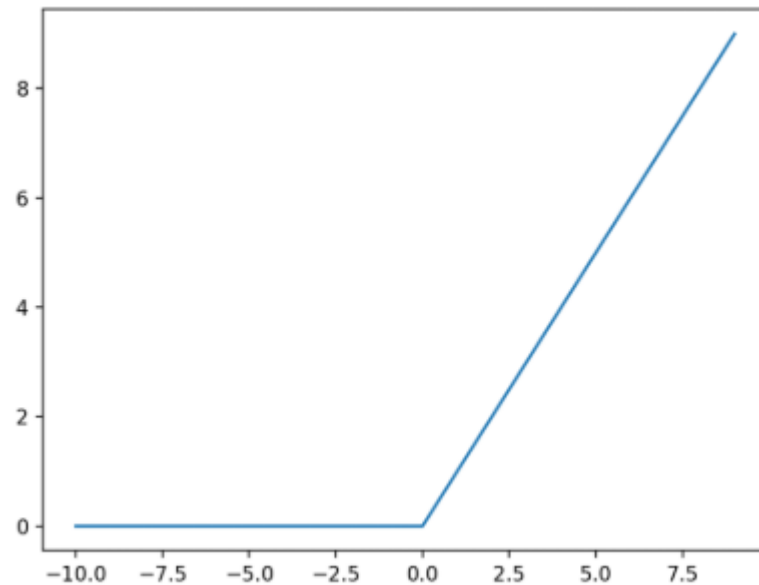


**Figure 32. Example of ReLu function.**

We have used Adam optimizer with a learning rate of 0.00147, we got this learning rate through trial and error, but we tried with 0.05, 0.0147, 0.01, 0.005, 0.00147, 0.001 and 0.005.

The loss function used for our algorithm was the squared loss, which is the most commonly used, and it is the sum of squared distances between the target value and the predicted value.

Our set up for the neural network is the following: an input layer of 8 neurons, a hidden layer of 20 neurons, a hidden layer of 10 neurons with a dropout of 20% of the neurons to stop overfitting, a hidden layer of 4 neurons and the output layer of 2 neurons.

Deeplearning4j allows to build simple neural networks in a simple way. It has a class called `NeuralNetConfiguration`, in which once you call its constructor and the method `builder()`, you can add new characteristics to the neural network by adding simple methods that modify this neural network, for example, each layer is added by calling the `layer` method, and there are several types of classes of layer that can be created. I will not go into further detail as everything can be seen into the `DeepLearning4j` web.

Our neural network configuration with this library is:

```

MultiLayerConfiguration conf = new NeuralNetConfiguration.Builder()
    .seed(seed)
    .weightInit(WeightInit.XAVIER)
    .updater(new Adam(0.00147))
    .list()
    .layer(new DenseLayer.Builder().nIn(labelIndex).nOut(20)
        .weightInit(WeightInit.XAVIER)
        .activation(Activation.RELU)
        .build())
    .layer(new DenseLayer.Builder().nIn(20).nOut(10)
        .weightInit(WeightInit.XAVIER)
        .activation(Activation.RELU)
        .build())
    .layer(new DenseLayer.Builder().nIn(10).nOut(4)
        .weightInit(WeightInit.XAVIER)
        .activation(Activation.RELU)
        .dropOut(20)
        .build())
    .layer(new OutputLayer.Builder(LossFunction.SQUARED_LOSS)
        .weightInit(WeightInit.XAVIER)
        .activation(Activation.SOFTMAX)
        .nIn(4).nOut(numClasses).build())
    .build();

```

**Code 2. Code of our Neural Network.**

And our algorithm for developing this neural network follows the next steps:

1. Read both dataset, training and testing.
2. Define number of epochs that will be iterated the dataset and the seed for initialize the neural network.
3. Define the configuration of the neural network, figure above.
4. Create a model of Multilayer network with the configuration.
5. Initialize the model.
6. Realize i iterations, in each iteration the model is trained with the training data over the number of epochs and an evaluation over the testing data set is done.

This simple feed forward neural network can understand some characteristics of the problem that we as humans cannot understand. This can be pretty good because it can be able to resolve a task that we cannot think how to solve it, but it will not show how the task is performed and will not give any clue in the decision made.

## 4. Evaluation

After explaining the design and implementation details of RetiRisk Calc, in this chapter, we are going to talk about the results in the different steps. After this, I will make an explanation of how our web service can be used and I will show a few examples. At the end of the chapter, I will show the results of our Neural Network and compare them with RETIPROGRAM and RetiRisk Calc results.

### 4.1. RetiRisk Calc results

In Chapter 3, a study of the new application has been done, decomposing the process in several steps. In this section, we will follow the same steps, showing the results separately to see the performance of the each of the parts of the whole process of developing this application. Before that, I will explain the evaluation measures that we are using to evaluate each model and decide which model is better at each step.

Taking into account that we want to classify patients as they are probably developing DR (class 1) or as they do not have risk of developing DR (class 0), we face a binary classification problem. With 2 classes, there are 4 possible outcomes.

1. True positive (TP). A patient that has DR and that the application identifies him as DR.
2. False positive (FP). A patient that does not have DR and that the application identifies him as DR.
3. True negative (TN). A patient that does not have DR and that the application identifies him as non DR.
4. False negative (FN). A patient that does have DR and that the application identifies him as non DR.

It is important to have present that even if we are aiming to have the best classification score, we are treating with diagnosis of a disease, which is a very sensible issue, since it is related with people's health. Consequently, we want a model that with as less FN as possible, while trying to get as much TP as possible. We are not so concerned about the number of false positives because, even if they suppose an extra unnecessary visit to the doctor, they are not as bad as false negatives because the patients will still be safe.

Our evaluation measures are:

- Total number of true positives.
- Total number of false positives.
- Total number of true negatives.
- Total number of false positives.
- Accuracy, which is the proportion of correct results divided by the total results, the unknown results are not taking account in this number.
- Specificity, corresponds to the people with no DR that are classified as such. So, the true negatives divided by the sum of true negatives and false positives.
- Sensitivity, corresponds to the people classified as DR that are classified as such. So, the true positives divided by the sum of true positives and false negatives.
- Precision, corresponds to the real number of DR within all the classified as DR. So, the true positives divided by the true positives and the false positives.

- Negative score, corresponds to the real number of no DR within all the classified as no DR. So, the true negatives divided between the true negatives and the false negatives.
- F1 score. Corresponds to the formula:  

$$2 * (Sensitivity * Precision) / (Sensitivity + Precision)$$
- All accuracy, which corresponds to the Accuracy but adding the unknown results.

In summary, we are interested in having high sensitivity, but without penalizing a lot the other classification scores.

#### 4.1.1. Results of binarizing data

First evaluation we did in this study was for deciding which binarization of the attributes was the best using the model of the RETIPROGRAM. We wanted to look if the attributes we had selected and the study for the binarization that we made were right, and even if we lose these 2 attributes we still have a good model and a good score.

In case we obtain a similar or a bit worse results for these models compared to a normal model, we can be sure that at least these attributes can be binarized individually between a good and a bad category.

We trained one model with the data non binarized and 5 models with binarized data as we explain in the point 3.1.2. Every model had the same parameters. These parameters were:

- $\delta_2$ . The difference that is needed between the number of one class and the other class after each tree has voted has to be higher than this number. It was set at 10.
- $\delta$ . The difference between the membership function of one class and the membership function of the other class in each tree to vote has to be higher than this number. It was set at 0.2.
- The  $\gamma$  attributes that each tree receives in every new decision was assigned to 1.
- The  $\alpha$ , which is the significance level of each tree, was assigned to 0.4.
- The  $\beta$ , which is the truth level threshold for each tree, was assigned to 0.8.

The different models we tested where:

| Name                   | Number |
|------------------------|--------|
| Normal_Data            | 1      |
| CKD(89-29) MA (0-100)  | 2      |
| CKD(89-15) MA (0-100)  | 3      |
| CKD(100-29) MA (0-100) | 4      |
| CKD(100-15) MA (0-100) | 5      |
| CKD(100-45) MA (0-100) | 6      |

Table 5. Models for data binarization.

The results obtained for these models were:

| Model | Acc   | Prec  | Sens  | Spec  | Negative | F1    | TP  | FN | FP  | TN  | U  | All accuracy |
|-------|-------|-------|-------|-------|----------|-------|-----|----|-----|-----|----|--------------|
| 1     | 81.21 | 53.80 | 76.23 | 82.54 | 92.87    | 63.08 | 170 | 53 | 146 | 690 | 44 | 77.97        |
| 2     | 80.89 | 53.02 | 75.91 | 82.21 | 92.81    | 62.43 | 167 | 53 | 148 | 684 | 51 | 77.15        |
| 3     | 80.93 | 53.31 | 76.13 | 82.21 | 92.81    | 62.71 | 169 | 53 | 148 | 684 | 49 | 77.33        |
| 4     | 80.85 | 53.11 | 76.34 | 82.05 | 92.87    | 62.64 | 171 | 53 | 151 | 690 | 38 | 78.06        |
| 5     | 80.88 | 53.40 | 76.55 | 82.05 | 92.87    | 62.91 | 173 | 53 | 151 | 690 | 36 | 78.24        |
| 6     | 80.85 | 53.11 | 76.34 | 82.05 | 92.87    | 62.64 | 171 | 53 | 151 | 690 | 38 | 78.06        |

Table 6. Results of data binarization.

As we can see, it is interesting to observe how when the patients are binarized and only some rules are activated some scores are better than in a normal model. This means that CKD-EPI probably could be separated in better intervals depending on the values. We can see than the model with better all accuracy, with better TP and with better sensitivity is the 5<sup>th</sup> model, which corresponds to the one that CKD-EPI is binarized between 100 and 15 and MA between 0 and 100.

The downside of this binarization is that even if there are 3 more patients in the TP, there are 5 more in FP, so doctors would have to work more. But at least, the number of FN is the same.

We also tested how this model performs on all the patients that have both attributes in the good value (MA under 30, now assigned to 0, and CKD-EPI over 60 now assigned to 100) and both attributes in the bad value (MA over 30, now assigned to 100, and CKD-EPI under 60 now assigned to 100).

| Name                   | Number |
|------------------------|--------|
| CKD(100-15) MA (0-100) | 1      |
| Good patients          | 2      |
| Bad patients           | 3      |

Table 7. Concrete cases of model of data binarization.

The results were:

| Model | Acc   | Prec  | Sens   | Spec  | Neg   | F1   | TP  | FN | FP  | TN  | U  | All accuracy |
|-------|-------|-------|--------|-------|-------|------|-----|----|-----|-----|----|--------------|
| 1     | 80.88 | 53.40 | 76.55  | 82.05 | 92.87 | 62.9 | 173 | 53 | 151 | 690 | 36 | 78.24        |
| 2     | 81.14 | 43.43 | 63.24  | 84.51 | 92.44 | 51.5 | 86  | 50 | 112 | 611 | 33 | 78.14        |
| 3     | 89.47 | 89.47 | 100.00 | 0.00  | NaN   | 94.4 | 51  | 0  | 6   | 0   | 0  | 89.47        |

Table 8. Results of concrete cases.

As we can see, the *good patients* model (number 2) has similar scores to the initial model even if the sensitivity decreased. This is due to the fact that even if these patients do not have kidney problems, they are not completely healthy. In the patients that have both attributes bad, the model classified them all to DR, and that means that both attributes are very important for classifying patients with DR. They are good results even if it failed on 6 patients of 57.

With these results, we are definitively sure that we can try our next step on the study with the objective of grouping both attributes into one.

#### 4.1.2. Results of deciding combination of MA and CKD-EPI

Now we are focused in trying which one is the best combination of grouping both attributes into only one. We cannot keep both attributes binarized, and we cannot do as we have done before. We have to take them both into account, not treated differently, they can't be one in the good value and one in the other value.

Now we are really expecting to have a downgrade in the model, but it is very important that this downgrade is not that big, if the model keeps about at least 70% accuracy it would be a good model for patients at home.

The combinations we tried were:

| Name                   | Number |
|------------------------|--------|
| CKD(100-15) MA (0-100) | 1      |
| Both bad               | 2      |
| MA bad                 | 3      |
| CKD-EPI bad            | 4      |

Table 9. Models of combination of both attributes.

These combinations mean:

- Both bad. When both attributes are the bad value (under 60 for CKD-EPI and higher that 30 for MA), they are both assigned the bad binarized value (15 for CKD-EPI and 100 for MA). Therefore, we are only considering a patient with bad kidney if it has problems on both MA and CKD-EPI. If a patient has problems only on one, the values assigned are 100 for CKD and 0 for MA.
- MA bad. We forget about CKD-EPI and only take into account MA, when a patient has under 30 of MA the values assigned are 0 for MA and 100 for CKD-EPI, even if the CKD-EPI was under 60. Moreover, when it has over 30 of MA the values assigned are 100 for MA and 15 for CKD-EPI, even if he had a good CKD-EPI value.
- CKD-EPI bad. We forget about MA and only take into account CKD-EPI, when a patient has under 60 of CKD-EPI the values assigned are 15 for CKD-EPI and 100 for MA. While if the patient has over 60 of CKD-EPI the values are assigned as 0 for MA and 100 for CKD-EPI.

In theory, as we saw in the study, the best results should be the MA bad and the both bad, because the MA has shown dominance over CKD-EPI. In addition, CKD-EPI values are good indicators of DR when they appear together with bad MA values.

The results were:

| Model | Acc   | Prec  | Sens  | Spec  | Neg   | F1   | TP  | FN | FP  | TN  | U  | All accuracy |
|-------|-------|-------|-------|-------|-------|------|-----|----|-----|-----|----|--------------|
| 1     | 80.88 | 53.40 | 76.55 | 82.05 | 92.87 | 62.9 | 173 | 53 | 151 | 690 | 36 | 78.24        |
| 2     | 78.84 | 50.86 | 78.07 | 79.05 | 92.85 | 61.6 | 178 | 50 | 172 | 649 | 54 | 74.98        |
| 3     | 80.77 | 53.11 | 76.34 | 81.96 | 92.83 | 62.6 | 171 | 53 | 151 | 686 | 42 | 77.70        |
| 4     | 79.14 | 51.18 | 76.65 | 79.83 | 92.54 | 61.4 | 174 | 53 | 166 | 657 | 53 | 75.34        |

Table 10. Results of combination of both attributes.

As predicted, the best results are obtained when we only have into account the MA. Even if the model of *both bad* (model 2) is better in sensitivity and TP, it loses a lot of accuracy compared to the *MA bad* model (num. 3). We decided to continue with the MA

model, that only losses 0.11 accuracy, 0.21 sensitivity and 2 TP against the model selected in the section before.

With these results, we see that we can clearly develop a new model with both attributes changed and one new attribute that englobes both of them, based on the MA values. This attribute as has been said is the Kidney attribute, which can be 0 for a patient with no problems in the kidney or 1 for a patient with problems in the kidney.

#### **4.1.3. Results of different models of RETIPROGRAM data binarized with different parameters**

Now, we are interested in finding the best parameters for our RetiRisk Calc, having into account that we do not want to be strict with both  $\delta$  and  $\delta_2$  values, because our aim is to have as low Unknown classified patients as possible.

For this evaluation, we used the data binarized based on MA, as we have chosen that model as the best for our work. First, we trained the models on the training project and obtained the FRF, then we put this FRF on the test project and we test it, the results obtained are from the test project.

We modified all parameters we have shown before as well as the aggregation in the classification algorithm to decide the vote of each tree. In particular, we tested:

- Minmax model and Choquet model for the classification algorithm to see if the new Choquet model works well in this new application too.
- $\delta_2$  and  $\delta$ : threshold values to see how they affect the number of unknowns.
- $\gamma$ : number of attributes used in each node on the tree, to see if when the node has more attributes to use, it has better results.
- $\alpha$  and  $\beta$ : to see if the truth threshold and significance level have influence in the accuracy and see if they have influence on the TP and TN metrics.

The models we tried were:

| Number | classification | delta2 | delta | gamma | Alpha | Beta |
|--------|----------------|--------|-------|-------|-------|------|
| 1      | minmax         | 0      | 0     | 3     | 0.3   | 0.7  |
| 2      | choquet        | 0      | 0     | 3     | 0.3   | 0.7  |
| 3      | choquet        | 0      | 0     | 3     | 0.4   | 0.8  |
| 4      | minmax         | 0      | 0     | 3     | 0.4   | 0.8  |
| 5      | minmax         | 10     | 0.2   | 1     | 0.3   | 0.7  |
| 6      | minmax         | 10     | 0.2   | 1     | 0.4   | 0.8  |
| 7      | minmax         | 0      | 0     | 1     | 0.3   | 0.7  |
| 8      | minmax         | 0      | 0     | 1     | 0.4   | 0.8  |

**Table 11. Models of RETIPROGRAM for better parameters.**

The results were:

| Model | Acc   | Prec  | Sens  | Spec  | Neg   | F1    | TP  | FN | FP  | TN  | U  | All accuracy |
|-------|-------|-------|-------|-------|-------|-------|-----|----|-----|-----|----|--------------|
| 1     | 77.50 | 48.94 | 76.67 | 77.73 | 92.29 | 59.74 | 184 | 56 | 192 | 670 | 1  | 77.43        |
| 2     | 77.93 | 49.6  | 77.5  | 78.05 | 92.56 | 60.49 | 186 | 54 | 189 | 672 | 2  | 77.79        |
| 3     | 80.27 | 53.58 | 71.67 | 82.67 | 91.27 | 61.32 | 172 | 68 | 149 | 711 | 3  | 80.05        |
| 4     | 80.16 | 53.29 | 71.13 | 82.67 | 91.15 | 60.93 | 170 | 69 | 149 | 711 | 4  | 79.87        |
| 5     | 80.39 | 52.81 | 77.17 | 81.27 | 92.91 | 62.71 | 169 | 50 | 151 | 655 | 78 | 74.71        |
| 6     | 82.13 | 55.75 | 74.07 | 84.28 | 92.4  | 63.62 | 160 | 56 | 127 | 681 | 79 | 76.25        |
| 7     | 78.09 | 49.86 | 76.25 | 78.60 | 92.22 | 60.30 | 183 | 57 | 184 | 676 | 3  | 77.88        |
| 8     | 79.75 | 52.65 | 70.42 | 82.35 | 90.90 | 60.25 | 169 | 71 | 152 | 709 | 2  | 79.60        |

**Table 12. Results of models of RETIPROGRAM looking for better parameters.**

The conclusions were:

- The Choquet model of classification is slightly better when compared to the minmax classification with the same parameters. Choquet has 2 more TP than the minmax in both models 2 and 3 compared to 1 and 4. Also all the other scores compared are better in 2 (with Choquet) than in the 1 (with min max) and in 3 (with Choquet) than in the 4 (with min max).
- $\delta_2$  and  $\delta$  are clearly affecting the unknown score. When  $\delta_2$  and  $\delta$  are higher, the unknown score is higher, and when they are lower the FRF classifies almost every patient and these patients are distributed between the other classes. Less unknowns in the model makes that some scores are a bit lower than in models 5 and 6: precision, specificity and F1. However, we can see that the number of TP is much higher with low values of these parameters, while keeping the same FN level.
- $\gamma$  almost has no effect, in the results, comparing models 1 and 4 with 7 and 9, we can see that both have similar results. While  $\gamma$  higher seems to have better numbers of TP and less U, a lower  $\gamma$  has more TN and less FP. The other scores are similar and derived from these values.
- $\alpha$  and  $\beta$  do not have influence in the level of unknowns but they suppose a big difference in the classification. Lower  $\alpha$  and  $\beta$  lead to a higher classification of patients into the TP and less FN, even if that means an increase in FP. We are interested in lower  $\alpha$  and  $\beta$  because we want to be able to get right most of the DR patients.

With all these conclusions, the selected model is the number 1. The models with largest number of unknowns are discarded because we do not want to build a tool that does not give an answer to the user. Even if Choquet is a bit better it takes much more time to develop and execute a Choquet model, so we do not see a big difference in performance to justify this a little bit better results.

#### **4.1.4. Results of RetiRisk Calc**

Until now, we have defined how we are going to create our new application. The binarization of the data is going to be based on MA. When MA in the patients is under 30, our new attribute Kidney will be 0, which means correct. Otherwise, when MA in the patients is over 30, our new attribute Kidney will be 1, which means bad.

In addition, we have defined the parameters that are going to be used with our new application, which are:

- $\alpha$  and  $\beta$ , 0.3 and 0.7 respectively.
- $\gamma$  equals to 3.

- $\delta_2$  and  $\delta$  equals to 0.
- minmax classification model.

After training the model with these parameter values and the new Kidney attribute, we obtain the RetiRisk calc model. Then, we tested the new model using the classification algorithm implemented in the testing project. The results obtained are shown in Table 13. They are compared to the results of RETIPROGRAM (normal data train) and with RETIPROGRAM using a binarized version of the attributes.

The models were:

| Name                    | Number |
|-------------------------|--------|
| RETIPROGRAM             | 1      |
| Binarized and parameter | 2      |
| RetiRisk Calc           | 3      |

**Table 13. Comparison of normal RETIPROGRAM, binarized data RETIPROGRAM and RetiRisk Calculator.**

And the results were:

| Model | Acc   | Prec  | Sens  | Spec  | Neg   | F1    | TP  | FN | FP  | TN  | U  | All accuracy |
|-------|-------|-------|-------|-------|-------|-------|-----|----|-----|-----|----|--------------|
| 1     | 81.21 | 53.80 | 76.23 | 82.54 | 92.87 | 63.08 | 170 | 53 | 146 | 690 | 44 | 77.97        |
| 2     | 77.5  | 48.94 | 76.67 | 77.73 | 92.29 | 59.74 | 184 | 56 | 192 | 670 | 1  | 77.43        |
| 3     | 77.77 | 49.19 | 76.15 | 78.22 | 92.21 | 59.77 | 182 | 57 | 188 | 675 | 1  | 77.70        |

**Table 14. Results of Comparison of normal RETIPROGRAM, binarized data RETIPROGRAM and RetiRisk Calculator.**

We can see that the new RetiRisk Calc, performs almost as good as the model 2. Differences are not significant. Compared to the original RETIPROGRAM, it has worse results, but still acceptable results, probably due to that we force the classification of every patient instead of allowing Unknowns.

After this study, doctors were satisfied with the new RetiRisk Calc and with the results obtained, so we developed the web service and we tested it, as is explained in the next section.

#### 4.2. Testing the web service for RetiRisk Calc.

In section 3.2, we have already explained how the web service is built, and how it works with the database and which resources are free to use for our patients. In 4.1, we have also seen that whenever a batch of patients is passed to the RetiRisk Calc model, it is able to classify the patients with a 77.70 accuracy. In this section, we are going to expose some examples of how our web service works, the results obtained and how the databases are updated with the new patients.

We will use invented patients as the data of the patients is confidential, but we are going to use possible examples based on the knowledge of biology and medical information.

Let us consider that we have our database free, and no one has even used our web service. We just started the web service and a first patient uses the web service. The data of this patient is:

- Age: 50.

- Sex: 0.
- EVOL: 5.
- TTM: 0.
- HbA1c: 7.2
- Kidney: 0
- BMI: 27.6
- HTAR:1

This patient would fill a form in a Web browser at home or in the optic clinic and the next URI would be generated:

<http://localhost:8080/WebRD/webresources/evaluation/newpatient?Age=50&Sex=0&EVOL=5&TTM=0&HbA1c=7.2&Kidney=0&BMI=27.6&HTAR=1>

As we can see, it uses the @QueryParam in the method, so the attributes are passed as name=value.

The results received by the patient in the browser in text plain are:

```
Su id es: 0 Predicción: NO tiene peligro de Retinopatía Diabética.
Porcentaje de seguridad: 54.0
```

**Figure 33. Results of the patient 0.**

We can see that this patient has no danger of developing DR, has an identifier of id=0 and the certainty of the prediction made by RetiRisk Calc is 54%.

After this, a new patient comes, and fills the form on the web. Her information is:

- Age: 67.
- Sex: 1.
- EVOL: 12.
- TTM: 1.
- HbA1c: 9
- Kidney: 1
- BMI: 26
- HTAR:0

Via the web browser, the client would call the URI:

<http://localhost:8080/WebRD/webresources/evaluation/newpatient?Age=67&Sex=1&EVOL=12&TTM=1&HbA1c=9&Kidney=1&BMI=26&HTAR=0>

The results obtained for this patient would be:

```
Su id es: 1 Predicción: SI tiene posibilades de Retinopatía Diabética.
Porcentaje de seguridad: 56.0
```

**Figure 34. Results of the patient 1.**

As we can see, the id of this patient is 1, and the prediction is that it has possibilities of DR with a certainty of 56%.

We have seen that our web service works fine with the RetiRisk Calc. It can predict patients who seem to be healthy, as the first one was, as well as pacientes with problems that could lead to DR, like the second one.

We have implemented the resource to be able to access as a patient that already exist and test again our health. Let us suppose that the first patient comes 3 years later, and fills the form again, now indicating his id number. The health of this patient has got a bit worse over these 3 years.

- Age: 53.
- Sex: 0.
- EVOL: 8.
- TTM: 2.
- HbA1c: 8.5
- Kidney: 0
- BMI: 30.2
- HTAR:1

Once he fills the form, the client used via HTTP calls the URI:

<http://localhost:8080/WebRD/webresources/evaluation/alreadypatient?id=0&Age=53&Sex=0&EVOL=8&TTM=2&Kidney=0&HbA1c=8.5&BMI=30.2&HTAR=1>

This patient has gone far worse than expected in these 3 years, and we can see it in his data. The results are:

Su id es: 0 Predicción: SI tiene posibilades de Retinopatía Diabética.  
Porcentaje de seguridad: 62.0

Figure 35. Second results of patient 0.

We can see how the RetiRisk Calc recognises that he is the same patient, so shows the id=0. Sadly, after 3 years the RetiRisk Calc verdict has changed and now this patient has possibilities of developing DR with a certainty of 62 %.

We can see all this information of the 3 patients stored in both of our databases. Notice that even if an entry has the same id than other entry, thanks that the primary key is ID + DATE, an entry of the same patient in different years will ever be stored while still keeping the other one.

| # | ID | AGE | SEX | EVOL | TTM | HBA1C | KIDNEY | BMI  | HTAR | FECHA                   |
|---|----|-----|-----|------|-----|-------|--------|------|------|-------------------------|
| 1 | 0  | 50  | 0   | 5    | 0   | 7.2   | 0      | 27.6 | 1    | 2019-05-31 03:06:37.862 |
| 2 | 1  | 67  | 1   | 12   | 1   | 9.0   | 1      | 26.0 | 0    | 2019-05-31 03:12:12.752 |
| 3 | 0  | 53  | 0   | 8    | 2   | 8.5   | 0      | 30.2 | 1    | 2019-05-31 03:18:26.027 |

Figure 36. Patients database of our project.

| # | ID | RD | CERTAINTY | DATE                    |
|---|----|----|-----------|-------------------------|
| 1 | 0  | 0  | 54.0      | 2019-05-31 03:06:37.862 |
| 2 | 1  | 1  | 56.0      | 2019-05-31 03:12:12.752 |
| 3 | 0  | 1  | 62.0      | 2019-05-31 03:18:26.027 |

Figure 37. Results database of our project.

Finally, we can use the last functionality, which is to retrieve all the information of a patient based on his id. We are going to try to see the history of the patient with id=0. We can see how the system has stored both results and this may be used to perform some data analysis.

When the patient id=0 wants to get his info to see what his results were 3 years ago, he marks this in the web browser and the web browser access the URI:

[http://localhost:8080/WebRD/webresources/patients/get\\_patient\\_by\\_ID?id=0](http://localhost:8080/WebRD/webresources/patients/get_patient_by_ID?id=0)

The results obtained by the patient, are the next ones. We can see both entries of the patient, with 3 years of difference, and we can see the different results he obtained. The patient would be interested in showing this result to a doctor.

```
Paciente{Age=50, Sex=0, TTM=0, EVOL=5, HbA1c=7.2, Kidney=0, BMI=27.6, HTAR=1, fecha=2019-05-31 03:06:37.862}
Paciente{Age=53, Sex=0, TTM=2, EVOL=8, HbA1c=8.5, Kidney=0, BMI=30.2, HTAR=1, fecha=2019-05-31 03:18:26.027}
Resultado{RD=0, Certainty=54.0, fecha=2019-05-31 03:06:37.862}
Resultado{RD=1, Certainty=62.0, fecha=2019-05-31 03:18:26.027}
```

Figure 38. Results of the history of the patient 0.

From these illustrative tests, we can conclude that we have developed a web service that works totally fine with the RetiRisk Calc program.

### 4.3. Results of our Neural Network.

We have already explained in section 3.3 the whole configuration of our neural network, which has 5 layers, an Adam optimizer with 0.00147 learning rate and a dropout in a hidden layer to try to avoid overfitting.

We have built models with many different configurations, but the better one has been presented in the section 3.3. We trained it with 50 iterations of 100 epochs, with 100 iterations of 100 epochs, with 200 iterations of 100 epochs and with 300 iterations of 100 epochs, because even if the accuracy is similar we can see differences.

| Model         | TP  | FN  | FP  | TN  | U | All accuracy |
|---------------|-----|-----|-----|-----|---|--------------|
| RetiRisk Calc | 182 | 57  | 188 | 675 | 1 | 77.7         |
| NN 100 iter   | 169 | 71  | 114 | 749 | 0 | 83.23        |
| NN 200 iter   | 121 | 119 | 58  | 805 | 0 | 83.95        |
| NN 300 iter   | 102 | 138 | 41  | 822 | 0 | 83.77        |
| NN 50 iter    | 203 | 37  | 227 | 636 | 0 | 76.07        |

Table 15. Results of our different neural networks compared with the RetiRisk Calculator results.

We can see how 2 different things are happening:

- When we train our neural network more than 100 iterations, it gets a higher accuracy in the problem, but it gets also less TP whenever it trains more iterations. The model of 100 iterations could be useful, It has 83% of accuracy, which is 5.5 points more than our RetiRisk Calc. In one hand, this neural network has 13 TP less. In the other hand, it has 74 FP less, so we would stop a lot of unnecessary visits to the doctor. Definitely, the 2 neural networks of 200 iterations and 300 iterations are useless for our objective even if they have higher accuracy.
- Our neural network with 50 iterations does not reach an 83 % accuracy of the other neural networks and it is 1.6 points below in accuracy of RetiRisk Calc. But it has 203 TP, 21 more than the RetiRisk Calc and 20 FN less than RetiRisk Calc. The downside of this Neural Network is that with it we would have 39 more FP, so we would not be stopping people going to the doctors as much as we would like to.

These results are fine and interesting and show that deep learning and neural network may have a word in classification of patients of DR. The big problem here is that even if they work with good results, our biggest problem is that deep learning does not give any feedback or reason when it classifies. When we are talking about medical treatments and diseases, it is important to have a reason on why you are classifying the health of a person into a class or another.

## 5. Conclusions.

Diabetic Retinopathy is one of the most important diseases derived from Diabetes Mellitus. Diabetes Mellitus patients have a risk of getting blind because of DR and the actual procedure for treating patients is high in time and economic costs. An early detection of DR is crucial and it allows fighting the blindness produced by DR. Sadly there is a high number of DM patients who have to be screened for an early detection of DR and this is costly and time consuming.

Several studies have been done to make the early detection of DR easier and reduce the work of the doctors, but these studies have been always made for doctors usage, while patients cannot use them. The aim of our study has been to simplify RETIPROGRAM, creating a new and simpler application called RetiRisk Calculator which can be used by patients from their home or optic clinic. We also aimed to create a web service for this new application and compare results with new Artificial Intelligence approaches, like neural networks.

For our new application we studied how the fuzzy logic works and how linguistic variables are obtained from numeric values, how a FDT is built and how a FRF is built and the different methods of grouping votes for all of the trees. Later we studied how MA and CKD-EPI are present in the dataset and the binarization we could do in the dataset. We also found how can we combine both these attributes into only one attribute and experimentally found the best parameters to train our FRF with the new datasets. At the end we built the RetiRisk Calculator offline version.

For the web service we studied what are web services, the two types that exist and why we decided to use a RESTful web service. Later we explained how to build this web service in java and how to create resources. Finally, we end up explaining how we have built our web service, how can we call it and an example of use, so we have the RetiRisk Calculator online version.

We studied how neural networks are similar to our brains, the mathematics behind neural networks and how can we optimize the training. Afterwards we explained how we built our neural network and, at the end, we explained the results obtained and compared them to RetiRisk Calculator offline.

We can conclude the next points from the project:

- CKD-EPI and MA can be simplified and merged to one attribute called Kidney, which is related to the knowledge of having renal problems or not.
- A new model of classification based on FRF can be obtained simplifying a completer and more complex one, with similar results in sensitivity and overall accuracy. The overall accuracy is 77% and 76% at sensitivity. RetiRisk Calculator is this model.
- RetiRisk Cals can be used by patients from their homes, preventing them for getting an appointment with the doctor if it is not necessary. This gives more safety to the patient and reduces the cost in the hospitals.
- An online version of RetiRisk Calculator has been developed as a RESTful web service.
- This online version can execute the RetiRisk Calculator offline version and has two ways of adding patients and a way to consult the history of the patient.

- A neural network can be built to solve the task of classification of DM patients into DR or no DR. The results obtained diverge in function of the training, but similar or slightly better performance is obtained.
- A big problem with a neural network is that even if it has better scores, it does not understand the problem in a human way, so we cannot extract information from a neural network. On the contrary, the FRF model is based on linguistic rules, so we can retrieve which are the rules used to classify a patient in order to give an explanation to the user about the reasons for the prediction made.

After a validation from the medical experts from University Hospital Sant Joan de Reus, the RetiRisk Calc tool has been registered for authorship in the international office for Intellectual Property at Benelux Office (see Appendix A).

This study can be continued in future projects in the next ways:

- Thanks to RESTful web services, new resources can be added easily to the online version and new modified models can be used,
- Thanks to storing different episodes (queries) in the RESTful web services, a time series analysis may be done in the future, in order to predict the risk of DR based on the evolution of the patient.
- A way of showing which rules were activated in the project has to be developed so we can see the most important rules that affect the risk of DR.
- Even if neural network alone should not be used for medical studies, both approaches could be combined in to one, so even if some rules of FDT are influenced by the result of a neural network, we would be taking into account both our knowledge and the one that a neural network has discovered.

From a personal experience, after finishing this work I have learned how a investigation project is done and I've gained a lot of experience in an investigation laboratory. I've also learnt a lot about Artificial Intelligence that was not taught in my college classes, and I've learnt about Web Services from 0. I consider this work has been a good experience that has showed me that hard work for 5 years pays itself with amazing opportunities.

From the double degree perspective, this project has demonstrated me how both worlds of biology science and computer engineering can be together and aim for objectives together. If I had 0 knowledge of any of the fields of the project I would have understood the whole project way less and would have probably took longer to develop our application.

## References

- [1] International Diabetes Federation. IDF Diabetes Atlas, 8th edn. Brussels, Belgium: International Diabetes Federation, 2017.
- [2] ["Causes and Risk Factors". Diabetic Retinopathy. United States National Library of Medicine.](#) 02/06/2019
- [3] Saleh MD, Eswaran C. An automated decision-support system for nonproliferative diabetic retinopathy disease based on MAs and HAs detection. *Comput Methods Progr Biomed*, 108:186–196, 2012.
- [4] De la Torre, J., Valls, A, Puig, D., A deep learning interpretable classifier for diabetic retinopathy disease grading, *Neurocomputing*, In press (available online). 2019.
- [5] López M, Cos FX, Álvarez-Guisasola F, Fuster E Prevalence of diabetic retinopathy and its relationship with glomerular filtration rate and other risk factors in patients with type 2 diabetes mellitus in Spain. DM2 hope study. *J Clin Transl Endocrinol*, 9:61–65, 2017.
- [6] Romero, P., de la Riva, S., Valls, A., Sagarra, R., Moreno, A., Soler, N., Changes observed in diabetic retinopathy. Eight year follow up of a Spanish population. *British journal of ophthalmology*, 100, 1366-1371, 2016.
- [7] Saleh, E., Moreno, A., Valls, A., Romero-Aroca, P., & de la Riva-Fernandez, S. (2016). A Fuzzy Random Forest Approach for the Detection of Diabetic Retinopathy on Electronic Health Record Data. In book: Artificial intelligence research and development, *Frontiers in artificial intelligence and applications*, vol. 288, pp. 169-172, IOS Press 2016.
- [8] Saleh, E., Valls, A., Moreno, A., Romero-Aroca, P, Torra, V., Bustince, H., Learning fuzzy measures for aggregation in fuzzy rule-based models, *Modeling Decisions for Artificial Intelligence*, Lecture notes in Artificial Intelligence, vol. 11144, pp. 114-127, Springer, 2018.
- [9] Zadeh, L.A. Fuzzy sets. *Information and Control*. 8 (3): 338–353. 1965.
- [10] Yuan Y, Shaw MJ. Induction of fuzzy decision trees. *Fuzzy Sets Syst*, 69:125–135, 1995.
- [11] Romero-Aroca, P., Valls, A., Moreno, A., Sagarra-Alamo, R, Basora-Gallisa, J., Saleh, E., Baget-Bernaldiz, M., Puig, D., A Clinical Decision Support System for diabetic retinopathy screening: Creating a clinical support application, *Telemedicine and E-Health*, 25(1), 31-40, 2019.
- [12] L. Breiman. Random forests, *Machine Learning* 45: 5-32, 2001.
- [13] L. Rokach. Decision forest: twenty years of research, *Information Fusion*, 27: 111-125, 2016.
- [14] [http://www.dmi.usherb.ca/~larocheh/neural\\_networks/content.html](http://www.dmi.usherb.ca/~larocheh/neural_networks/content.html). The whole neural net course. 02/06/2019
- [15] Martínez-Castelao, A., Górriz, J. L., Bover, J., Segura-de la Morena, J., Cebollada, J., Escalada, J., ... & Hernández-Moreno, J. Documento de consenso para la detección y manejo de la enfermedad renal crónica. *Endocrinología y Nutrición*, 61(9), e25-e43, 2014.
- [16] Teruel, J. L. G., & Catalán, S. B. Valoración de afección renal, disfunción renal aguda e hiperpotasemia por fármacos usados en cardiología y nefrotoxicidad por contrastes. *Revista Española de Cardiología*, 64(12), 1182-1192, 2011.
- [17] Wu, J., Geng, J., Liu, L., Teng, W., Liu, L., & Chen, L. (2015). The relationship between estimated glomerular filtration rate and diabetic retinopathy. *Journal of ophthalmology*, 2015.
- [18] [http://www.dmi.usherb.ca/~larocheh/neural\\_networks/content.html](http://www.dmi.usherb.ca/~larocheh/neural_networks/content.html). The whole neural net course. 02/06/2019

[19]<https://prateekvjoshi.com/2016/03/29/understanding-xavier-initialization-in-deep-neural-networks/> Xavier initialization. 02/06/2019.

## Appendix A. Intellectual Property register of RetiRisk Calc.

B  
O  
I  
P



Benelux Office  
for Intellectual Property



i-DEPOT evidence

Number

116715

Date

01-04-2019

Reference

IISPV\_URV\_2/19

Title

RetiRisk-Calc: An online risk calculator for Diabetic Retinopath

In the name of

Fundació Institut d'Investigació Sanitària Pere Virgili (IISPV)  
c/ Esconador s/n  
43003 Tarragona  
Spain

This electronic file constitutes proof that all the included data was submitted to the Benelux Office for Intellectual Property (BOIP) on the date mentioned and has not been altered subsequently.

Edmond Simon  
Director General BOIP



1/5

BENELUX OFFICE FOR INTELLECTUAL PROPERTY | WWW.BOIP.INT