MACHINE LEARNING PROJECT REPORT

Assignment 1

Abstract

In this project I will discuss my implementations of two machine learning models, the comparison between them but also conducting an EDA, Data Quality Report for which I chose to go with a diabetes dataset. In the study I learned all about the key points needed to be able to implement a machine learning model.

K00273028: Kamil Woloszyn

K00273028@student.tus.ie

Table of Contents

Contents

Introduction

Project Objective

This project is all about Machine Learning, we will be discussing two different models that I have selected, the first model is decision trees. I decided to select this model as it was simple to implement and was easy to understand, next machine learning model I have chosen for my assignment is neural networks, now I decided to use this model as I found it very interesting and versatile. The data set I have chosen for this project is a diabetes dataset, this data set is available online for free to anyone through Kaggle. The features in the dataset of which has 8 different features, there is also the target feature, which is necessary for training a machine learning model, in this dataset that target feature is whether a patient has diabetes. I decided to choose this set as I know some people who have diabetes, so this dataset felt very close to home.

Dataset Overview

To get a better understanding of the dataset, the features include Gender, Age, Hypertension, Heart disease, smoking history, BMI, HbA1c levels, blood glucose levels. Out of all of these features the there were two that I did not understand these were Hypertension and HbA1c levels, Hypertension is a fancy way to say high blood pressure while HbA1c levels refer to the measure of haemoglobin that has bound itself to glucose in red blood cells. Now that we all know what the features are one thing to keep in mind is the size of the dataset, now the one I have selected has around 100,000 rows which means that I should have plenty of data to run my machine learning models on while ensuring a consistent result.

Exploratory Data Analysis

Dataset Structure

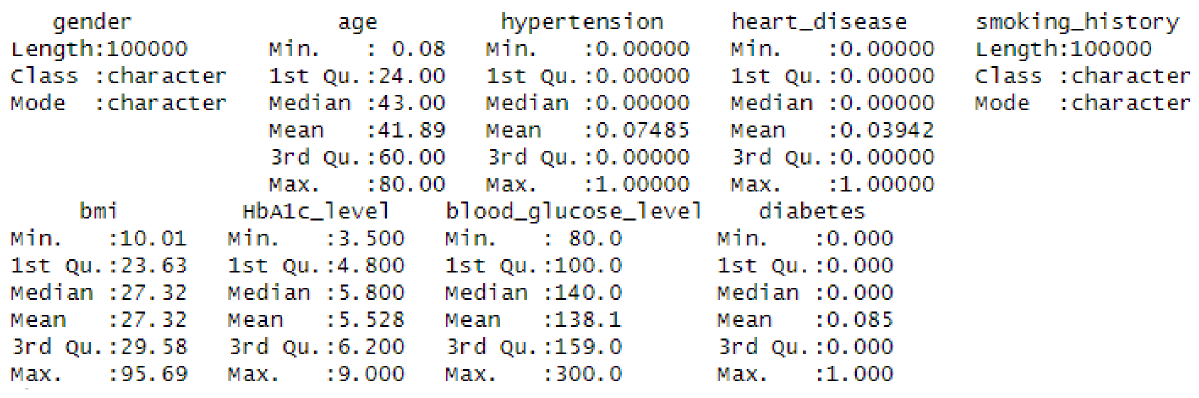
Gender	Age	Hypertension	Heart Disease	Smoking History	BMI	HbA1c Levels	Blood Glucose Levels	Diabetes
Male	23	1	0	Never	29.12	6.6	140	0
Female	42	0	0	Former	35.23	6.5	200	1
Male	54	1	1	No Info	21.24	5.6	85	0

Figure 1.1 Dataset example

Just from looking at the data in figure 1.1 I can tell you that I would have to find a way to change the data from a string into preferably an int that way it can be normalized, age, BMI, HbA1c and Blood Glucose all seem like they have an unknown range which means that I could potentially benefit from normalizing them for the models.

Feature Analysis

I was able to verify that all other values except gender and smoking history were input as a number or an int when it came to columns that were true or false like diabetes, hypertension and heart disease, for these I decided to leave them the way they are. Next up I wanted to make sure that there were no nulls hidden inside the data, so I performed some data analysis be checking total nulls in the dataset and verifying it with another function which checked individual columns for nulls. I then wanted to see the different statistics of each column like what the max and min values for each column that is the mean etc, to do this I used the summary R function which returned me with the data seen below in figure 1.2.



```
    gender               age         hypertension      heart_disease      smoking_history   
 Length:100000      Min.   : 0.08   Min.   :0.00000   Min.   :0.00000   Length:100000     
 Class :character   1st Qu.:24.00   1st Qu.:0.00000   1st Qu.:0.00000   Class :character  
 Mode  :character   Median :43.00   Median :0.00000   Median :0.00000   Mode  :character  
                    Mean   :41.89   Mean   :0.07485   Mean   :0.03942                     
                    3rd Qu.:60.00   3rd Qu.:0.00000   3rd Qu.:0.00000                     
                    Max.   :80.00   Max.   :1.00000   Max.   :1.00000                     
      bmi         HbA1c_level    blood_glucose_level    diabetes    
 Min.   :10.01   Min.   :3.500   Min.   : 80.0       Min.   :0.000  
 1st Qu.:23.63   1st Qu.:4.800   1st Qu.:100.0       1st Qu.:0.000  
 Median :27.32   Median :5.800   Median :140.0       Median :0.000  
 Mean   :27.32   Mean   :5.528   Mean   :138.1       Mean   :0.085  
 3rd Qu.:29.58   3rd Qu.:6.200   3rd Qu.:159.0       3rd Qu.:0.000  
 Max.   :95.69   Max.   :9.000   Max.   :300.0       Max.   :1.000  
```

Figure 1.2 Data Analysis on each data column

My initial thoughts on the statistics of each column of the data set is that there seems to be quite an unusual age range, you don't typically see ages below 1, also the statistics for gender and smoking history are unavailable, this was to be expected as they are characters. Later, I will be normalizing the data which means that statistics can be done after that time or by simply making the fields factors would help solve this issue.

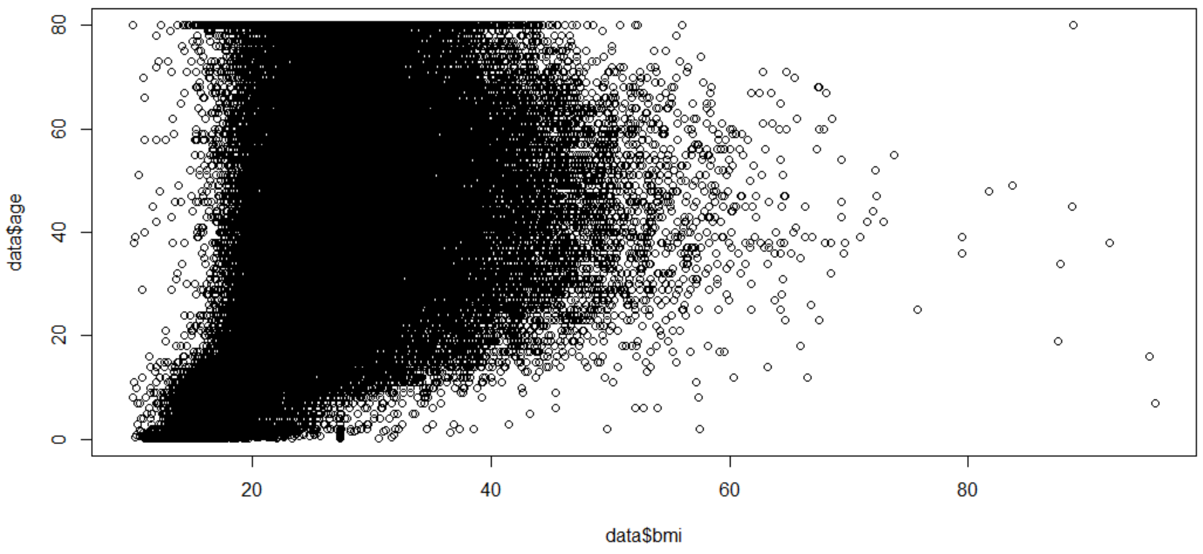
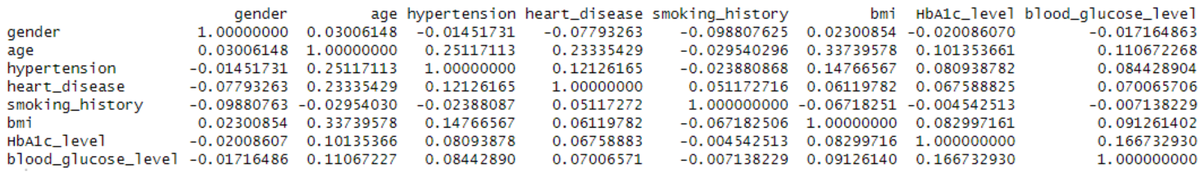


Figure 1.3 Diagram representing BMI vs Age plotted

(Additional images related to graphs in this section have been added to appendices as I did not have space here).

The final bit of exploratory analysis I did was displaying all data in frequency histogram graphs and also plotted some data against each other as seen in figure 1.3 this gave me a lot more insight into how the data is proportioned and how that is going to affect my final outcome it also told me there is a few outliers in the data that I may have to address later on in the preprocessing stage, the result of the analysis concluded with me realizing that the data is most likely real world data or something similar, which means that for the Data Quality report I would have a choice of changing the biases of the real world or not when it came to preprocessing the data.



```
                       gender         age hypertension heart_disease smoking_history         bmi  HbA1c_level blood_glucose_level
gender             1.00000000  0.03006148  -0.01451731   -0.07793263    -0.098807625  0.02300854 -0.020086070        -0.017164863
age                0.03006148  1.00000000   0.25117113    0.23335429    -0.029540296  0.33739578  0.101353661         0.110672268
hypertension      -0.01451731  0.25117113   1.00000000    0.12126165    -0.023880868  0.14766567  0.080938782         0.084428904
heart_disease     -0.07793263  0.23335429   0.12126165    1.00000000     0.051172716  0.06119782  0.067588825         0.070065706
smoking_history   -0.09880763 -0.02954030  -0.02388087    0.05117272     1.000000000 -0.06718251 -0.004542513        -0.007138229
bmi                0.02300854  0.33739578   0.14766567    0.06119782    -0.067182506  1.00000000  0.082997161         0.091261402
HbA1c_level       -0.02008607  0.10135366   0.08093878    0.06758883    -0.004542513  0.08299716  1.000000000         0.166732930
blood_glucose_level -0.01716486  0.11067227   0.08442890    0.07006571    -0.007138229  0.09126140  0.166732930         1.000000000
```

Figure 1.4 Correlation Matrix

In the above figure 1.4 we have the correlation matrix which shows the relationship between different variables where 0 is no relation and 1 is absolute relation, from the above matrix we can surmise that BMI and age have a slight positive relationship which means that they could potentially work really well together in data visualization, as well as age relationship to hypertension and heart disease seems slightly positive too.

Data Quality Report

Data Quality Issues

I decided to check the data for any mission values early on; this was to prevent any issues with me being able to create plots and histograms of the different columns inside the dataset. I check the total null values which initially I found a few so I quickly decided to run the code again and all of the null values disappeared, I realized later after 30 minutes of trying to figure out where the null values went to and I just had accidentally ran a piece of code that was not finished and the dataset I had open was corrupted so re importing it actually fixed the issue this meant that I wasted about 30 minutes on finding a bug that did not exist. After that I wanted to check the data for any outliers or what could be considered an outlier, I used boxplot stats method to extract outliers which the values captured by the function all seemed relatively close enough to the median so I decided to keep them, this did not stop me from figuring out the different interquartile ranges that would allow me to clamp down on some data for example BMI as previously noticed in figure 1.3 there is quite a few people with a BMI of over 80.

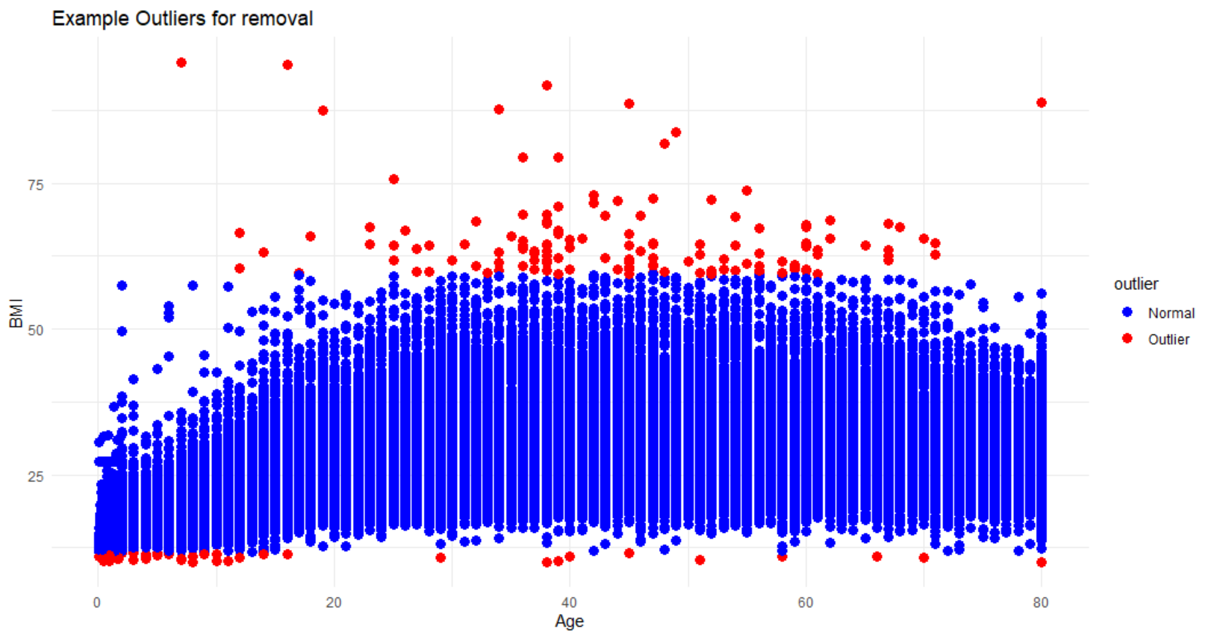
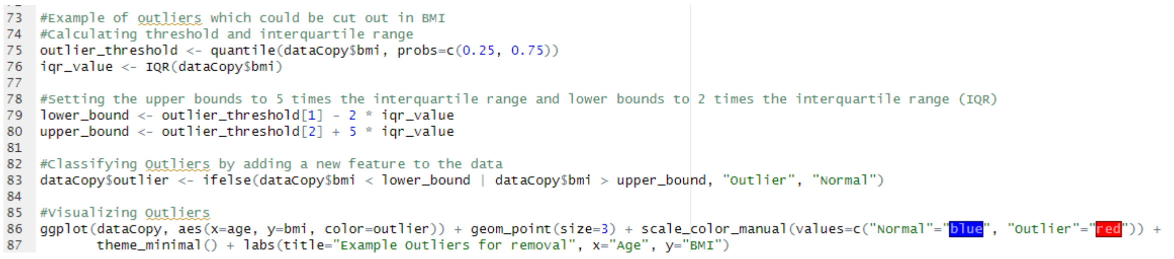


Figure 1.5 Outlier clamping example visualized

In the above figure 1.5 I have prepared a ggplot graph that represents all the different outliers that could be removed to improve the kappa measurement when it comes to performance of the two models implemented later, the reason I chose to only visualize this and not actually perform the clamping is because the data is currently a real world example of the spread of patients with actually confirmed cases of diabetes but also because if I were to clamp it I believe yes the kappa measurement would go up but the way the models currently work is to prefer ensuring that people who do not have diabetes will not be given a false positive but a false negative is much more likely, this could be a



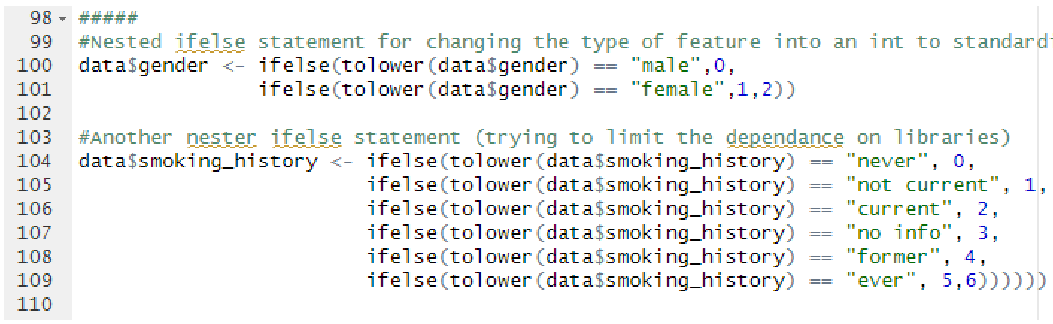
```r
#Example of outliers which could be cut out in BMI
#Calculating threshold and interquartile range
outlier_threshold <- quantile(dataCopy$bmi, probs=c(0.25, 0.75))
iqr_value <- IQR(dataCopy$bmi)

#Setting the upper bounds to 5 times the interquartile range and lower bounds to 2 times the interquartile range (IQR)
lower_bound <- outlier_threshold[1] - 2 * iqr_value
upper_bound <- outlier_threshold[2] + 5 * iqr_value

#Classifying Outliers by adding a new feature to the data
dataCopy$outlier <- ifelse(dataCopy$bmi < lower_bound | dataCopy$bmi > upper_bound, "Outlier", "Normal")

#Visualizing Outliers
ggplot(dataCopy, aes(x=age, y=bmi, color=outlier)) + geom_point(size=3) + scale_color_manual(values=c("Normal"="blue", "Outlier"="red")) +
      theme_minimal() + labs(title="Example Outliers for removal", x="Age", y="BMI")
```

Figure 1.6 Code for clamping down outliers in BMI column

crucial piece of information if processing data for the goal of administering medication and the medication has severe adverse effects on those who do not have diabetes. Additionally in figure 1.6 we can observe that due to the nature of the way I decided to clamp down on outliers cuts out some most likely average and healthy newborn and at the same time it does not cut out the children with a BMI level of over 50.

Data Preprocessing Strategy

When preprocessing my data before it was passed into the machine learning models, I had to make sure that all the data passed in was not going to cause any weird behaviour when it came to training the models. To this end I decided to turn the previously mentioned character columns into integer values by for example turning the word “Male” in the gender column to 0, “Female” to 1 and anything else to a 2, I achieved this by implementing a nested ifelse statement as seen below in figure 1.6.



```r
#####
#Nested ifelse statement for changing the type of feature into an int to standardize it
data$gender <- ifelse(tolower(data$gender) == "male",0,
                      ifelse(tolower(data$gender) == "female",1,2))

#Another nester ifelse statement (trying to limit the dependance on libraries)
data$smoking_history <- ifelse(tolower(data$smoking_history) == "never", 0,
                               ifelse(tolower(data$smoking_history) == "not current", 1,
                               ifelse(tolower(data$smoking_history) == "current", 2,
                               ifelse(tolower(data$smoking_history) == "no info", 3,
                               ifelse(tolower(data$smoking_history) == "former", 4,
                               ifelse(tolower(data$smoking_history) == "ever", 5,6))))))

```

Figure 1.6 Code for preprocessing character data

I decided to use the nested ifelse statement just to reduce the need to different libraries through out my code as I believe I am already using enough libraries. After this step I normalized the data by applying the functions seen underneath at figure 1.7.

$$\boldsymbol{result} = \frac{(\boldsymbol{x} - \boldsymbol{MIN})}{(\boldsymbol{MAX} - \boldsymbol{MIN})}$$

What this function does is it brings all the data into the 0 to 1 range, in the case of my data there is not much if any data loss when doing this as my initial ranges are small. Now after applying the final step, we are ready for implementation of the machine learning models, as there were no null values I did not have to remove any, but if there was, I included a command to get rid of all of them inside of the comments.

Machine Learning Implementation

Model Selection Rationale

When selecting my machine learning models I really wanted to make sure I picked something that would not cause me too much trouble to begin with, this is why I selected Decision Trees as it was easy enough to understand the concept of it, the next choice I was making I wanted to make sure that my second model was something I tried implementing in the past with a weak foundational knowledge of machine learning this time around I was able to implement it correctly which was a big deal to me as it showed me the growth I experienced as a software developer in the past year. Given the current data and the knowledge, I gained from the models and everything I believe that the decision tree model will be faster to execute put provide worse accuracy than the neural network model, but that is yet to be seen.

Model 1: Decision Trees

The way that this model works is by when it starts it starts the root node which is essentially the first entry into our dataset, the goal of the whole process is to find the best feature to split the dataset on but also the threshold of the given feature. For example, the threshold of HbA1c could be 0.58 meaning anything above this value is a no or a yes and anything under this value is the opposite of the value. There are quite a few split metrics you may choose from when going with the implementation of this model, the one I chose to go with is the Gini Impurity split metric, this metric is for classification and is the default metric used in the decision tree model library I decided to use for this project: rpart.

I have implemented this machine learning model with the use of the rpart library and the caret library which allowed me much more hyper parameter control than the original rpart library. The main reason I chose caret instead of just using rpart is because it is a unified framework that has a consistent interface for over 200 different machine learning models, this means that by implementing it this way I just made it easy for anyone to come along and quickly change the model used in the code. The library offers a lot of different features which can be used by people who are implementing the machine learning models for example caret allows for parallel processing for models significantly speeding up training times.

Model Evaluation

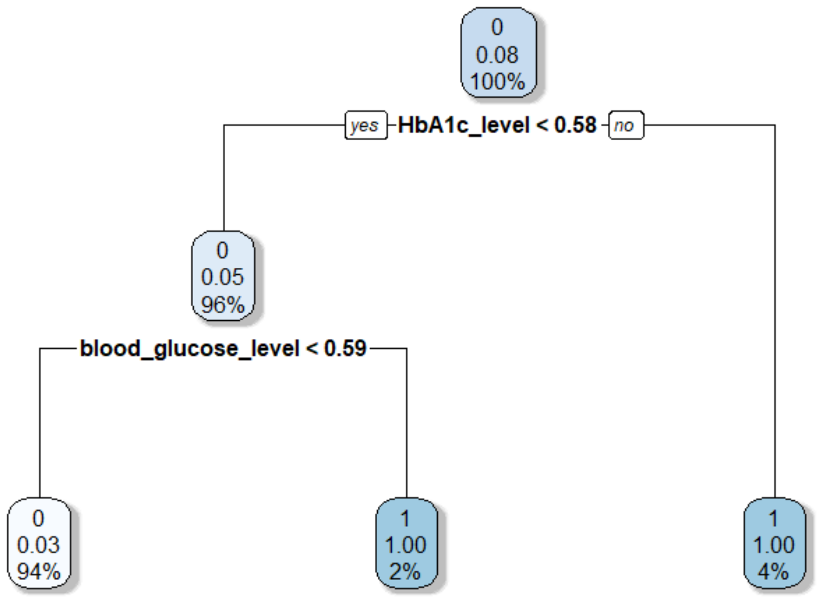
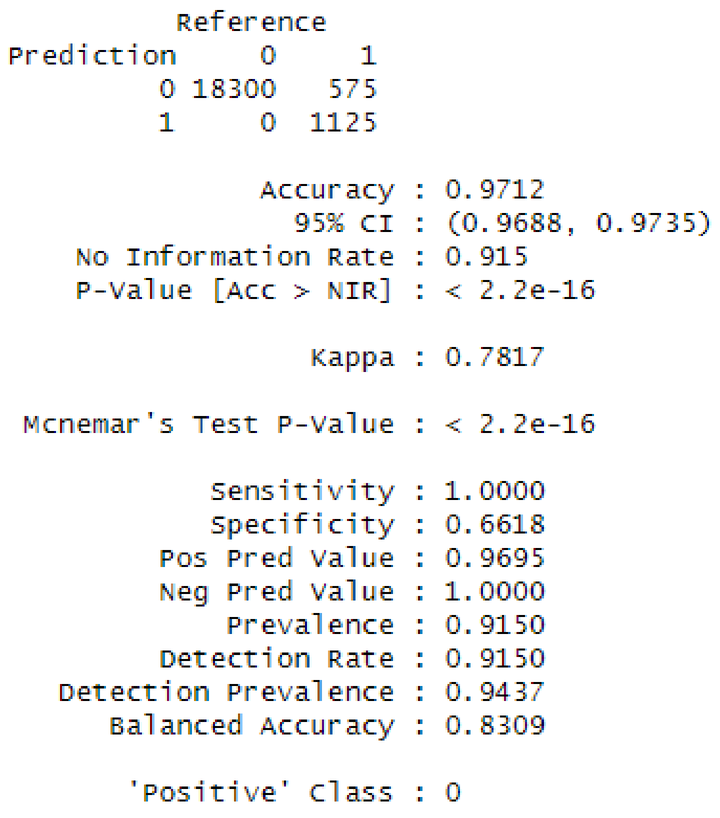


Figure 2.1 Decision Tree final tree visualization

It took me quite some time to get my code to a stage where the first machine learning model was running and working, the first problem I ran into was that the model was actually just not running any of the data, this issue was fixed by me making the diabetes field into a factor, the second problem I ran into while implementing this machine learning model was to do with actually predicting the data, I kept getting null predictions until I realized that I was normalizing incorrectly and by mistake turning a column into just nulls. There were other issues, but they were small and will be talked about later. Above in figure 2.1 you can see the decision tree created by the model during training; it is only 2 features deep which is great for a machine learning model. After performing some performance checks the kappa value of this model came out to around 0.7817 keeping within 0.05 through different runs which I would consider a success, though kappa is great for checking accuracy of precision for imbalanced data, the fact that my data is imbalanced could cause issues in the long run. I will address this in the comparison chapter of this report.



```
          Reference
Prediction     0     1
         0 18300   575
         1     0  1125

               Accuracy : 0.9712
                 95% CI : (0.9688, 0.9735)
    No Information Rate : 0.915
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7817

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 1.0000
            Specificity : 0.6618
         Pos Pred Value : 0.9695
         Neg Pred Value : 1.0000
             Prevalence : 0.9150
         Detection Rate : 0.9150
   Detection Prevalence : 0.9437
      Balanced Accuracy : 0.8309

       'Positive' Class : 0
```

Figure 2.2 Confusion Matrix & Statistics for Decision Tree Model

In the above figure 2.2 is the confusion matrix for the decision tree model, as can be seen by it the model is amazing at predicting negative cases to 100% accuracy which is not ideal if you are trying to make a flexible model but for medical cases I think it is perfect, at the same time the model is also pretty good at predicting positive patients with over 60 percent accuracy, this is a really nice score while keeping flexibility, though it misclassified positive patients as negative too which means that this model is mainly suited towards negative patients. This is since the dataset used was imbalanced and this imbalance can be over corrected by under sampling negative patients.

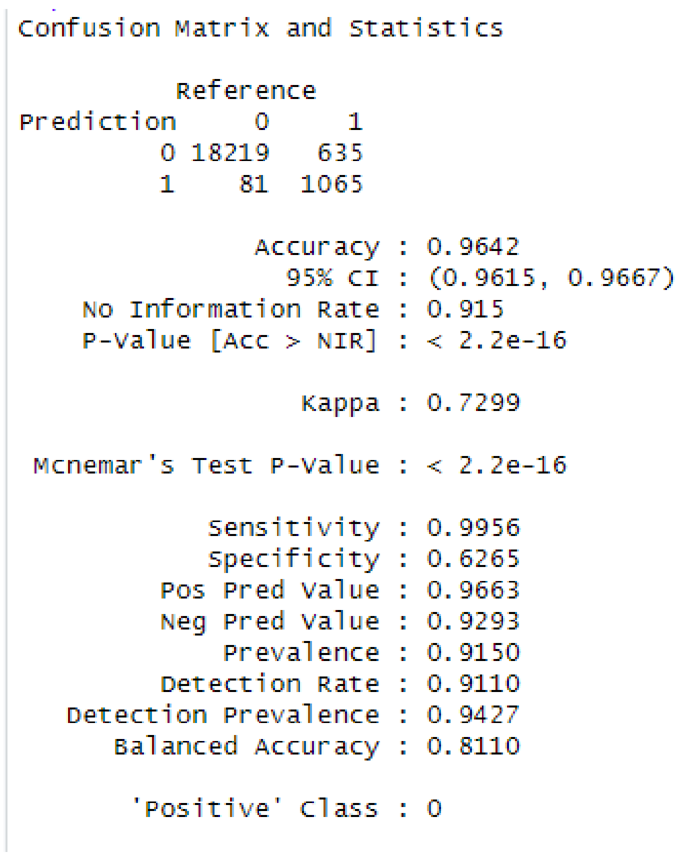
Model 2: Neural Network

The way this model works is by mimicking the human brain, we are able to learn by building paths by sending signals through our neurons, then activating is what supposedly gives us our intelligence, but in reality we do not really know exactly how it happens because when it comes to the brain we still don't know anything concrete and more research is being done every day. We mimic this behaviour in the model by having neurons(nodes) set up with connections to other nodes nearby, a row of nodes is called a layer and neural networks have a secret hidden layer which holds around 5 to 20 neurons, between nodes there is a weight this weight can go up or down based on an algorithm applied later. The way that the model's data flows is that each neuron receives inputs, using these inputs it calculates a weight, then it squashed the output into a 0 to 1 probability. The way that learning happens in this model is by a calculation that happens towards end of a training cycle. What essentially happens at this spot is that the model makes a prediction, then an error amount is calculated based on prediction, and then we propagate backwards updating all the weights. This entire process takes a while to complete as it needs to complete these steps quite a few times. In my case 100 iterations take around 30 seconds to complete which is not ideal when trying to speed through something like testing.

The way I have implemented this model is like the previous model as I used the caret library for this model as well to simplify the codebase and not going through different libraries that would end up doing the exact same thing. The only difference in the implementation was the library selection and the hyper parameter tuning and figuring out exactly what the different ranges are for the hyperparameters.

Model Evaluation

In comparison to the previous model this models implementation was so much easier as I already had fixed issues with my data, the only this that remained was to test the model and actually alter the hyper parameters until I am happy with the results, it is a bit of an educated guess when it comes to trying to find the best hyper parameters for the model, one thing that helped in this process is the ability to parallel process the models which meant that they finished training and predicting much faster. The kappa value for this model was 0.7299 which really surprised me as I thought that the neural network model would do much better.



```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 18219   635
         1    81  1065

               Accuracy : 0.9642
                 95% CI : (0.9615, 0.9667)
    No Information Rate : 0.915
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7299

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9956
            Specificity : 0.6265
         Pos Pred Value : 0.9663
         Neg Pred Value : 0.9293
             Prevalence : 0.9150
         Detection Rate : 0.9110
   Detection Prevalence : 0.9427
      Balanced Accuracy : 0.8110

       'Positive' Class : 0
```

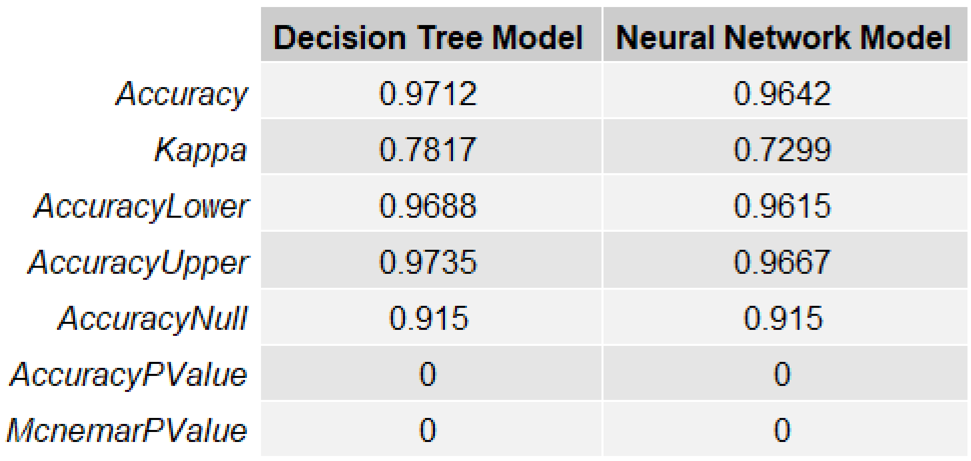
Figure 2.3 Confusion Matrix & Statistics for Neural Network Model

In the above figure 2.3 I have included the confusion matrix for my neural network model implementation, this model makes a total of 81 false negative and 635 false positives which could be great depending how you look at it but generally speaking a more flexible approach would be preferred where the target accuracy of the model was around 70-80% instead of my current 96%. This again is due to the imbalanced dataset which had a lot more negative patients than it did positive patients causing this real-world imperfection to occur.

Model Comparison and Discussion

Performance Comparison

Comparing the speeds of the two models when it came to the actual time spent training the model and predicting results using the model, I did not think that the neural network model would perform worse than the decision tree model. It may be possible to get the neural network model to run much faster but I believe when it comes to the time taken to train and predict with the decision tree model was far superior to neural networks to the point I did not even have to use parallel processing and the result was still instantaneous, when it came to the neural network model I would often leave the room while it completed its training and prediction.



	Decision Tree Model	Neural Network Model
Accuracy	0.9712	0.9642
Kappa	0.7817	0.7299
AccuracyLower	0.9688	0.9615
AccuracyUpper	0.9735	0.9667
AccuracyNull	0.915	0.915
AccuracyPValue	0	0
McnemarPValue	0	0

Figure 2.4 Comparing Statistics of the two machine learning models

When it comes to the actual performance of the models in terms of correctness, I have provided a quick comparison table I figure 2.4 to help me showcase the differences. As seen in the diagram the accuracy of the decision tree is only 0.7% from each other and yet they are much more different than that. Accuracy does not provide an accurate measure of the distribution of data, a variable that does that is the kappa, kappa considers the possibility of imbalanced data. Kappa is a measure of agreement between the predicted result and the results used for testing. A score of 0 shows no agreement in data and it would be worse than just doing random guessing, a 1 is perfect agreement between predicted and actual classes. In my case a score of around 0.75 is considered a strong agreement. This means that the data is imbalanced, but the accuracy of the model is good enough to account for it.

The variables AccuracyLower and AccuracyUpper variables refer to the 95% confidence interval, which in my case I believe a score of over 0.9 is pretty good, also AccuracyNull refer to the baseline accuracy if the model guessed the answers instead. For example, if my data contains 80% of patients who are not diabetic, a score lower than 80% when it comes to negative patients could be something to worry about. The next statistic is the AccuracyPValue, what this variable represents is the comparison of the AccuracyNull variable mentioned previously to my models accuracy and tools you whether or not your model is better than just guessing the correct answer, the same concept is applied to McnemarPValue variable as this variable analyses the discordant cases, though in this variables case the closer to 1 this value is the better. This value represents the a potential systematic error, in my case I believe this value is 0 due to the imbalanced nature of the dataset as if I limit the dataset to 50% negative diabetes cases this value rises to being close to 1, so for the purposes of this project and my sanity I ignored this value as I could not find anything wrong and all other measured seemed ok.

Implementation Challenges

When it came to implementing the two machine learning models I did not really know where to start the project, it seemed like such a massive undertaking at the time. To learn and get myself familiar with the topic I had to do a lot of looking at documentations and YouTube tutorials on implementing machine learning. All that work combined and by the time I knew it I was writing this report.

I think the most frustrating part of the whole project is the amount of time I spent waiting for training or prediction to finish before I could work some more, due to this I cannot recommend trying to implement machine learning on a tight schedule.

Insights

While developing this project I learned a lot of context when it comes to machine learning like knowing what a kappa is or knowing what is meant by the McnemarPValue I think that now I would be able to advise someone during a standup whether or not it would be a good idea to implement a machine learning model or that there is an easier solution. Which was my main goal that motivated me through the countless hours of waiting for my training model to finish training.

Conclusion

Summary of Findings

In the EDA chapter we found out that the data have around 100,000 rows and has 8 features and 1 target feature this was great as it was exactly something I was looking for. I also went in depth on the statistics of each feature and discussed the correlation matrix of the data. Then in the data quality report section I discussed how to deal with outliers, how to deal with missing values and we also talked about how to I went about normalizing the data before putting it into the machine learning models. After that I discussed the implementation of my chosen machine learning models and the different performance metrics I used to classify the model and then those same metrics to compare the two models to each other to find a best model which in my opinion must be the decision tree model using rpart library as it offered a less imbalanced result.

Limitations

The main limitation of neural networks is that it cannot be ran on small datasets as it is prone to overfitting this is the reason why I decided to settle upon my 100,000 entries dataset.

A limitation when it comes to decision trees is that it gives very poor performance when it comes to unorganized data, but apart from that it can be used very flexibly with majority of other tasks like being placed in real time systems like games etc.

Future Work

In this project I implemented a few things that should make it easier for future developers to refractor the code or use it in a different situation, the first this I added were comments, I decided to comment almost everything in the code, I used highlighted comments for different sections of the code and I also used folding comments to close certain section of code so you don't accidentally run it over and over, this really helped me out during programming. Though these implementations are made assuming the user uses RStudio.

The last thing I added for future proofing my code is the addition of seeding at the start of the code for reproducibility of results.

Appendices

References

https://www.statology.org/exploratory-data-analysis-in-r/

https://www.geeksforgeeks.org/exploratory-data-analysis-in-r-programming/

https://r4ds.had.co.nz/exploratory-data-analysis.html

https://mljourney.com/exploratory-data-analysis-in-r/

https://www.geeksforgeeks.org/building-a-simple-neural-network-in-r-programming/

https://www.geeksforgeeks.org/decision-tree-in-r-programming/

https://www.geeksforgeeks.org/r-tutorial/

https://www.geeksforgeeks.org/r-programming-for-data-science/

https://www.geeksforgeeks.org/multi-layered-neural-networks-in-r-programming/

https://www.w3schools.com/r/default.asp

https://bradleyboehmke.github.io/HOML/DT.html#final-thoughts-5

https://cran.r-project.org/web/packages/caret/vignettes/caret.html

https://www.rdocumentation.org/packages/nnet/versions/7.3-20/topics/nnet

https://ggplot2.tidyverse.org/reference/

https://www.rdocumentation.org/packages/rpart/versions/4.1.24/topics/rpart

https://dplyr.tidyverse.org/

https://www.rdocumentation.org/packages/gridExtra/versions/2.3

https://cran.r-universe.dev/doParallel/doc/manual.html

Additional Images

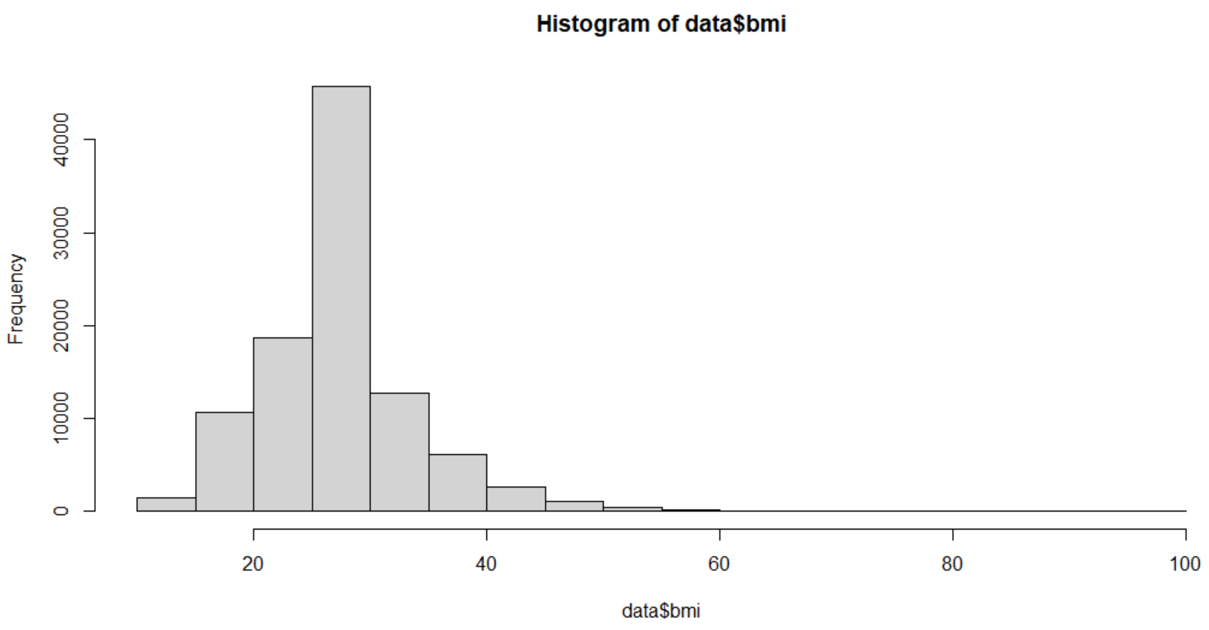


Figure A Diagram created during EDA representing BMI spread

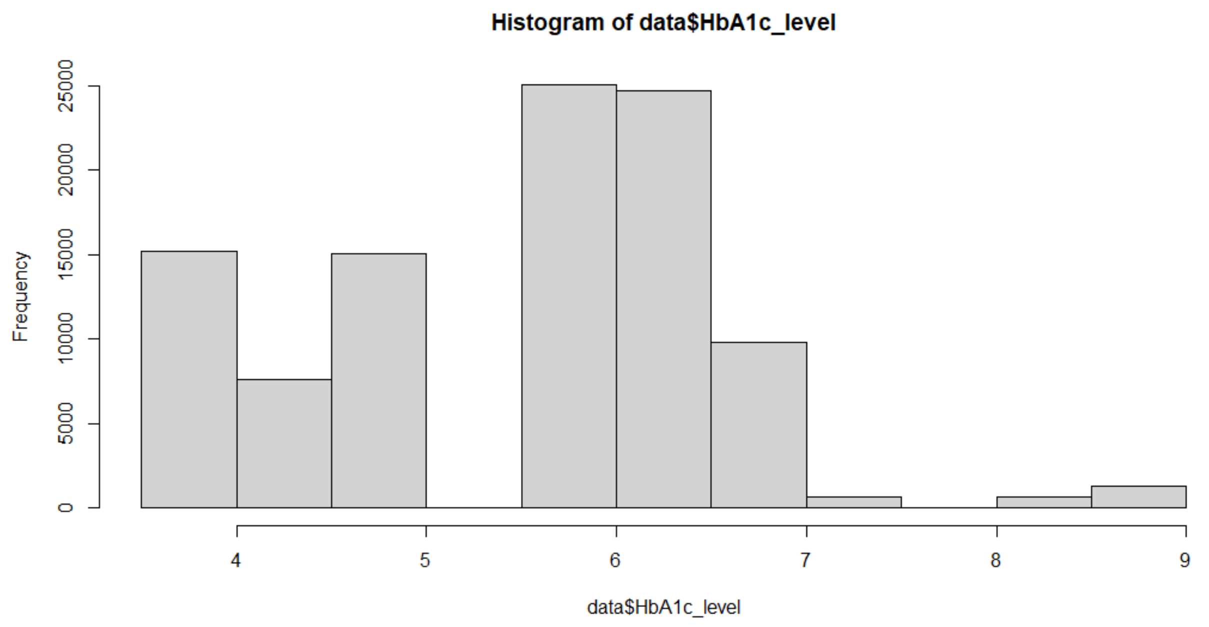


Figure B Diagram created during EDA representing HbA1c Levels spread

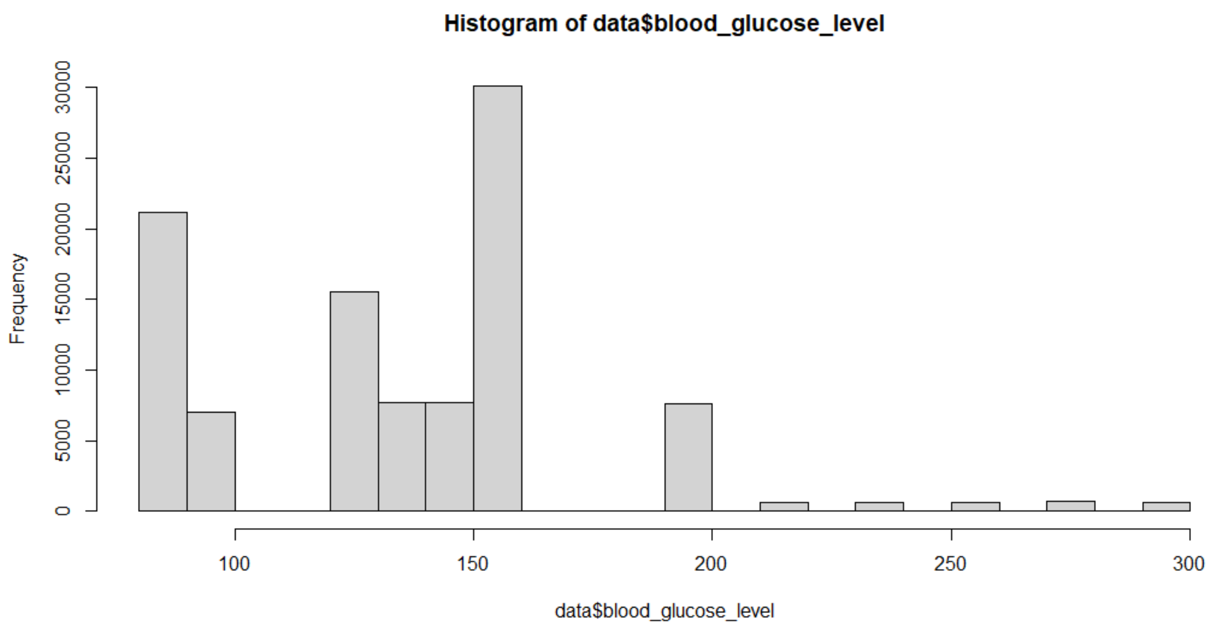


Figure C Diagram created during EDA representing blood glucose levels spread

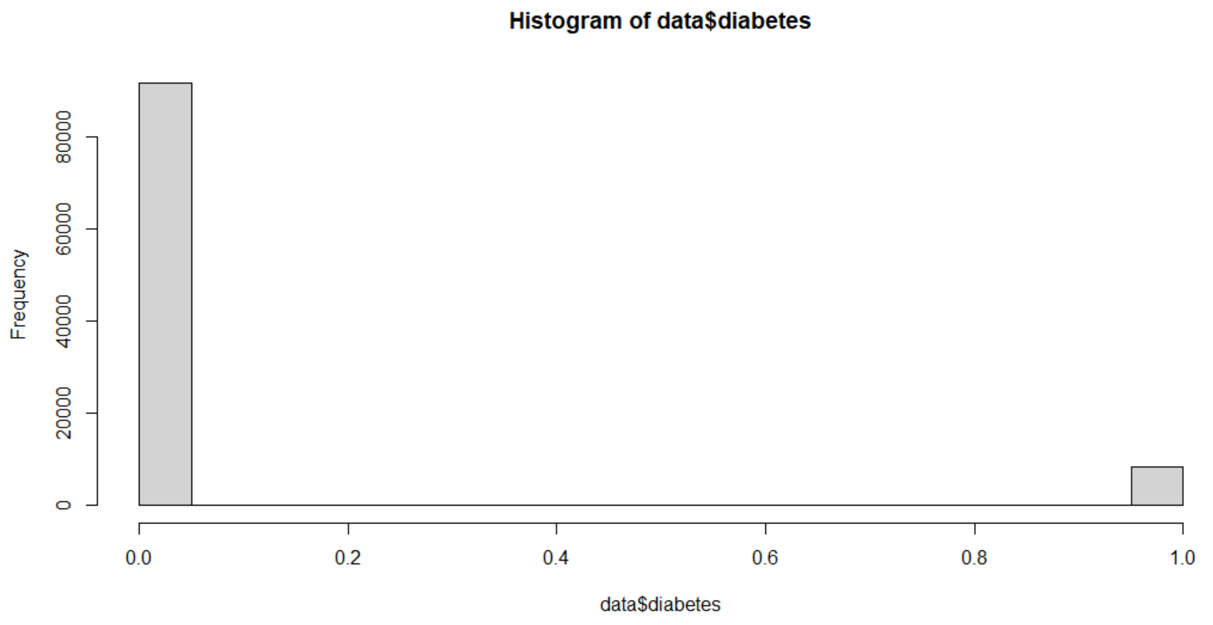


Figure D Diagram created during EDA representing diabetes spread

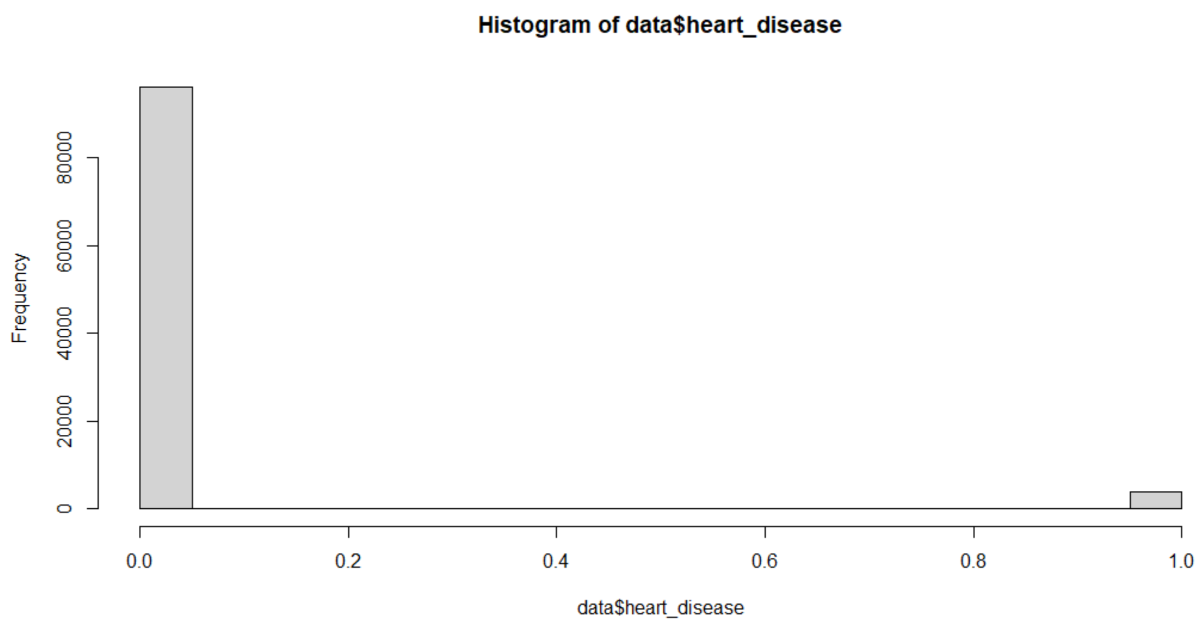


Figure E Diagram created during EDA representing heart disease spread

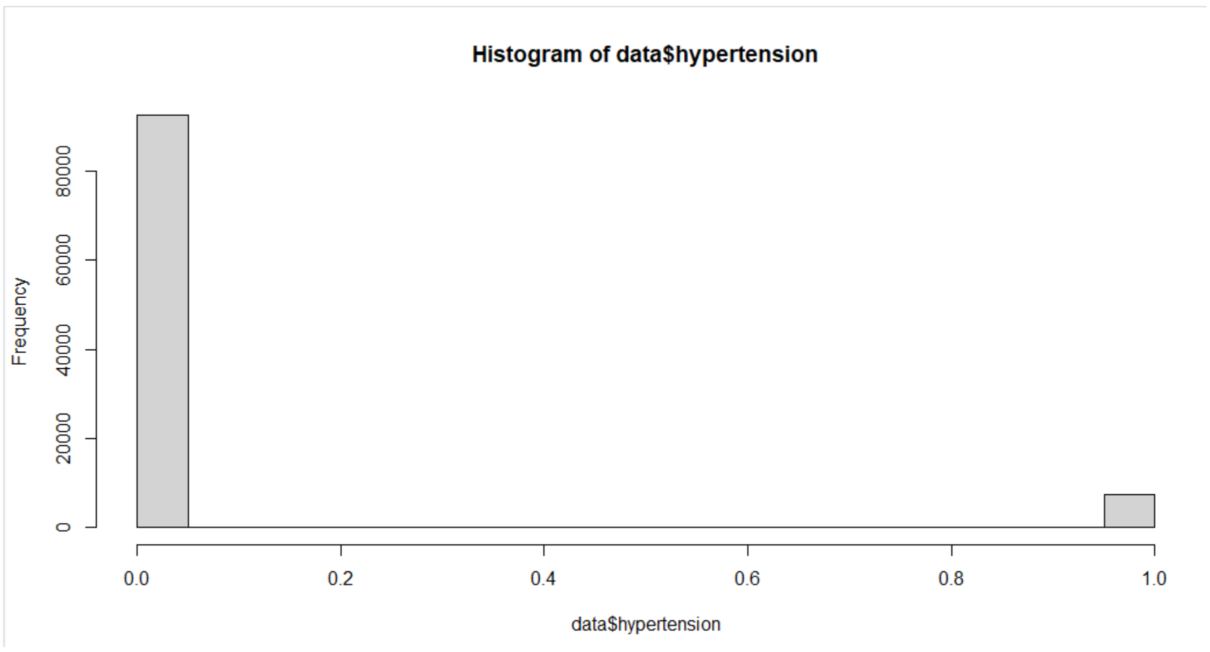


Figure F Diagram created during EDA representing hypertension

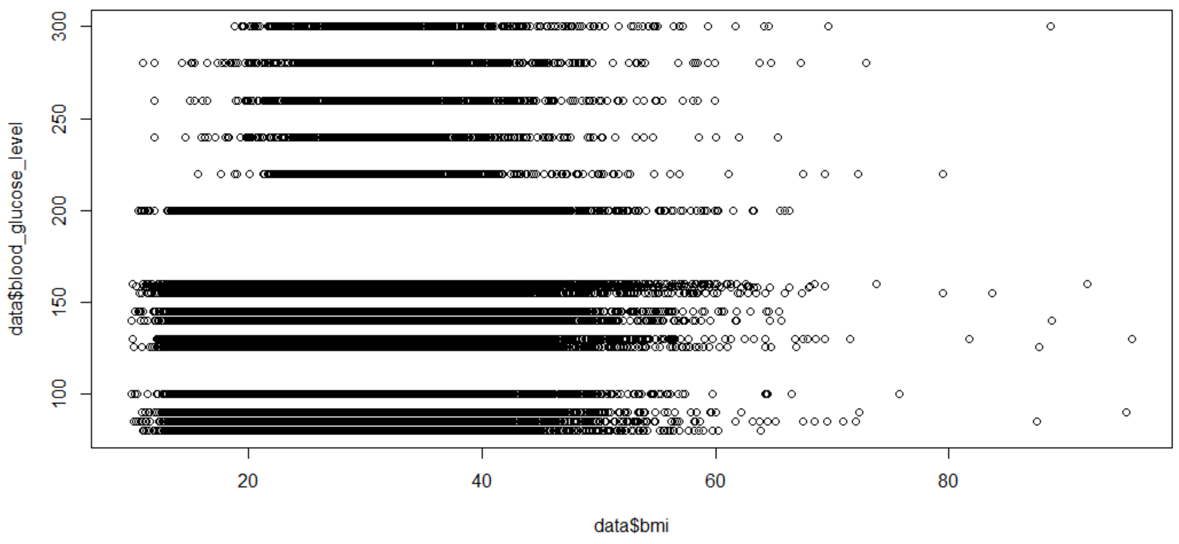


Figure G Diagram created during EDA representing bmi vs blood glucose levels plotted

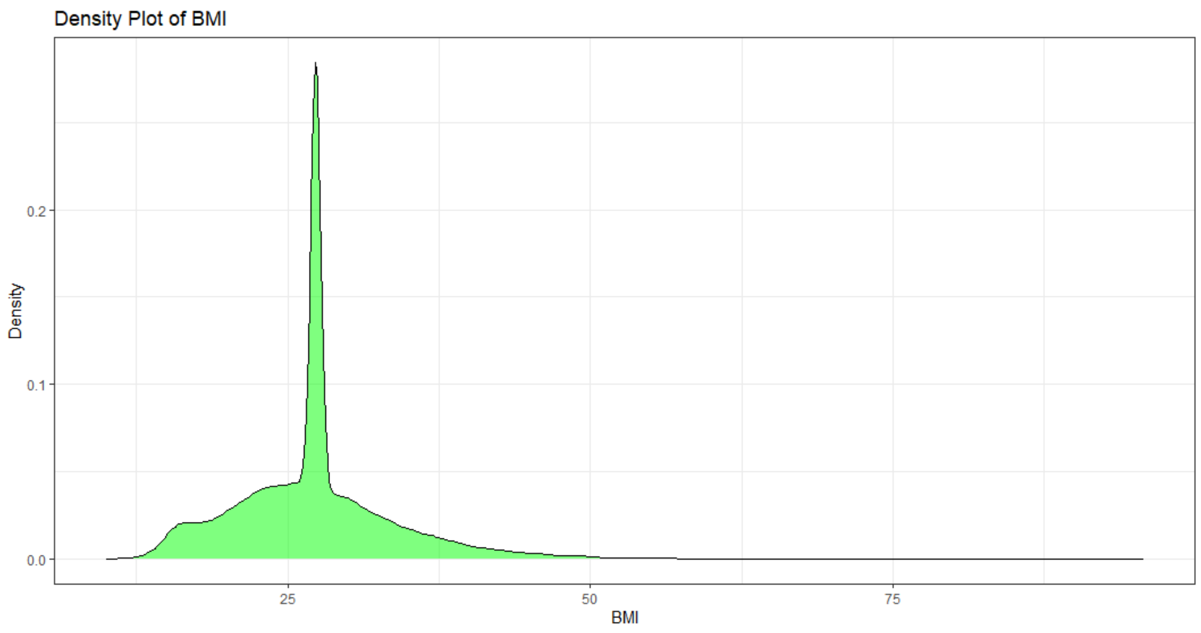


Figure H Diagram created during Data Quality Report representing BMI density