HarvardX : CHOOSE YOUR OWN

AHMED EL HAMOUNI

Health Diabetes

Contents

https://github.com/ahmedelhamouni/EL-HAMOUNI-AHMED-Your-Own-Project-edx-

INTRODUCTION

The view

This document pertains to the HarvardX PH125.9x Data Science: Capstone certification and focuses on the selected topic of Diabetes Binary Classification. For this project, we will utilize a dataset available at https://github.com/ahmedelhamouni/EL-HAMOUNI-AHMED-Your-Own-Project-edx-/blob/main/diabetes.csv.

The objective of this project is to develop a machine learning model that can predict whether a woman has diabetes based on physiological variables. Given the binary nature of the classification problem, we will evaluate the performance of the models using two metrics: accuracy and f1-score. These metrics will help us assess the model's ability to correctly classify instances and its overall predictive power.

The data :

The dataset utilized in this project is obtained from the National Institute of Diabetes and Digestive and Kidney Diseases. It encompasses physiological attributes collected from women aged 21 years and above belonging to a specific community. The dataset comprises several variables that will be employed in our analysis, enabling us to explore and derive insights from the collected physiological data.

Pregnancies : times a woman has been pregnant.

Glucose : glucose level in blood.

BloodPressure : blood pressure measurement.

SkinThickness : thickness of their skin.

Insulin : insulin level in blood.

BMI : body mass index.

DiabetesPedigreeFunction : diabetes percentage.

Age : age of the woman

Outcome : 1 if the woman has diabetes, 0 otherwise.

ANALYSING :

After downloading the dataset, we proceed to import it into our analysis environment. Next, we perform computations to extract and display the first five lines of the dataset. This step allows us to gain an initial understanding of the data and its structure by observing a subset of the information. Analyzing the first few lines provides insights into the variables, their

formats, and the overall data organization, helping us prepare for further exploration and analysis.

DATA SET DIMENSIONS

Rows	Columns
768	9

```
> diabetes <- data.frame(read_csv("diabetes.csv"))
Rows: 768 Columns: 9
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
dbl (9): Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFun
ction, Age, Outcome

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

VARIABLES STATISTICS

```
# statistical description

summary(diabetes[1:4]) %>% knitr::kable()

summary(diabetes[5:8]) %>% knitr::kable()

str(diabetes)
```

```
summary(diabetes[1:4]) %>% knitr::kable()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness
	Min. : 0.000	Min. : 0.0	Min. : 0.00	Min. : 0.00
	1st Qu.: 1.000	1st Qu.: 99.0	1st Qu.: 62.00	1st Qu.: 0.00
	Median : 3.000	Median :117.0	Median : 72.00	Median :23.00
	Mean : 3.845	Mean :120.9	Mean : 69.11	Mean :20.54
	3rd Qu.: 6.000	3rd Qu.:140.2	3rd Qu.: 80.00	3rd Qu.:32.00
	Max. :17.000	Max. :199.0	Max. :122.00	Max. :99.00

```
summary(diabetes[5:8]) %>% knitr::kable()
```

	Insulin	BMI	DiabetesPedigreeFunction	Age
	Min. : 0.0	Min. : 0.00	Min. :0.0780	Min. :21.00
	1st Qu.: 0.0	1st Qu.:27.30	1st Qu.:0.2437	1st Qu.:24.00
	Median : 30.5	Median :32.00	Median :0.3725	Median :29.00
	Mean : 79.8	Mean :31.99	Mean :0.4719	Mean :33.24
	3rd Qu.:127.2	3rd Qu.:36.60	3rd Qu.:0.6262	3rd Qu.:41.00
	Max. :846.0	Max. :67.10	Max. :2.4200	Max. :81.00

STRUCTURE OF THE DATA SET

```
str(diabetes)

'data.frame':  768 obs. of  9 variables:
 $ Pregnancies             : num  6 1 8 1 0 5 3 10 2 8 ...
 $ Glucose                 : num  148 85 183 89 137 116 78 115 197 125 ...
 $ BloodPressure           : num  72 66 64 66 40 74 50 0 70 96 ...
 $ SkinThickness           : num  35 29 0 23 35 0 32 0 45 0 ...
 $ Insulin                 : num  0 0 0 94 168 0 88 0 543 0 ...
 $ BMI                     : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
 $ DiabetesPedigreeFunction: num  0.627 0.351 0.672 0.167 2.288 ...
 $ Age                     : num  50 31 32 21 33 30 26 29 53 54 ...
 $ Outcome                 : num  1 0 1 0 1 0 1 0 1 1 ...
```

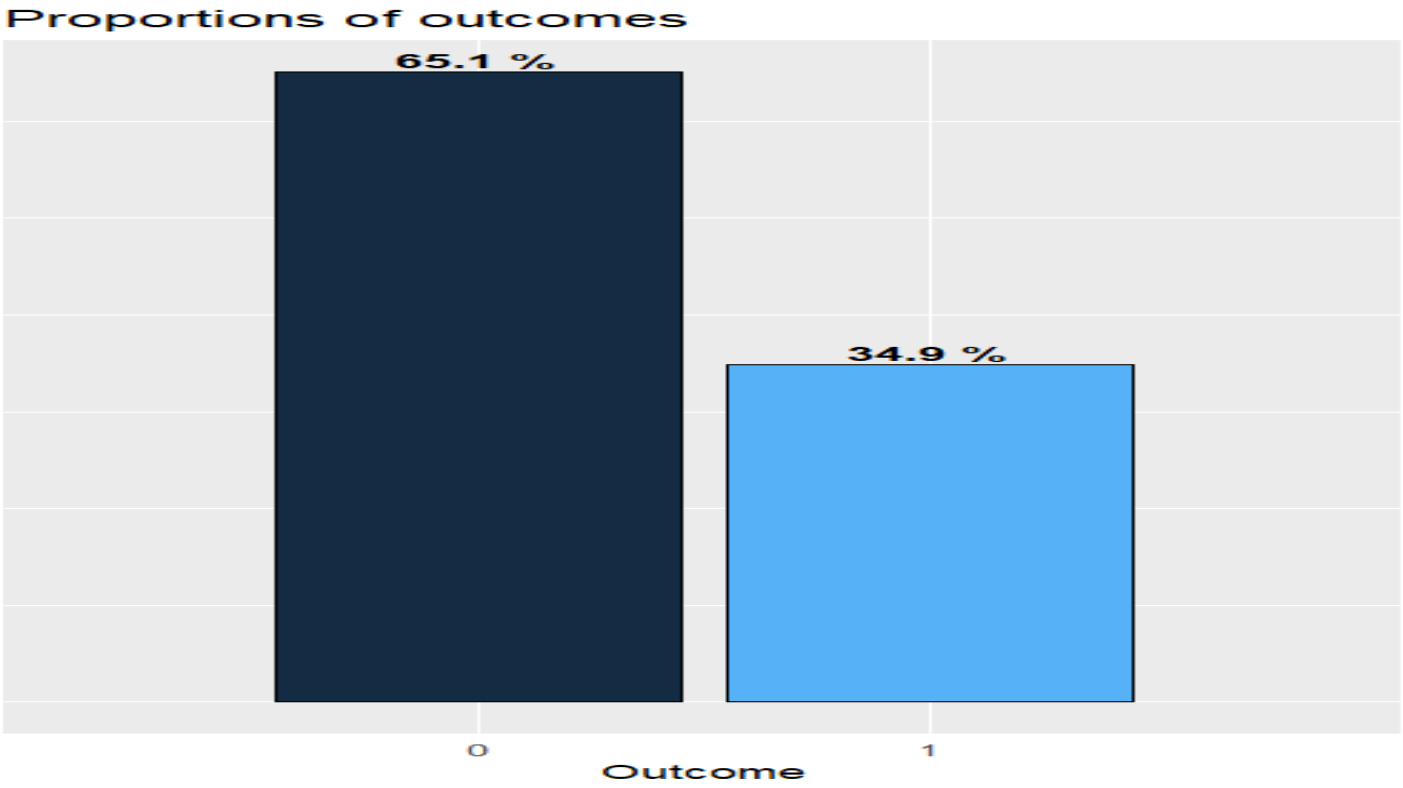
PROPORTION OF OUTCOME VALUES

To further analyze the dataset, we calculate the proportion of values for the "Outcome" variable. The "Outcome" variable represents a specific attribute or condition of interest in the dataset. By computing the proportion of Outcome values, we determine the relative frequency of different outcomes within the dataset. This information helps us understand the distribution and prevalence of specific outcomes, providing insights into the overall characteristics and trends related to the variable. Analyzing the proportion of Outcome values allows us to gain a deeper understanding of the dataset and its potential implications for our analysis.

outcome display

```
diabetes %>% group_by(Outcome) %>%

 summarise(N=n()*100/dim(diabetes)[1]) %>%

 ggplot(aes(x=Outcome,y=N,fill=Outcome))+

 geom_text(aes(label=paste(round(N,1),"%"),vjust=-0.25,fontface='bold'))+

 geom_bar(stat = 'identity',color='black') +

 theme(axis.title.y = element_blank(),

       axis.text.y = element_blank(),

       axis.ticks.y = element_blank(),

       axis.ticks.x = element_blank(),

       legend.position = "") +
```

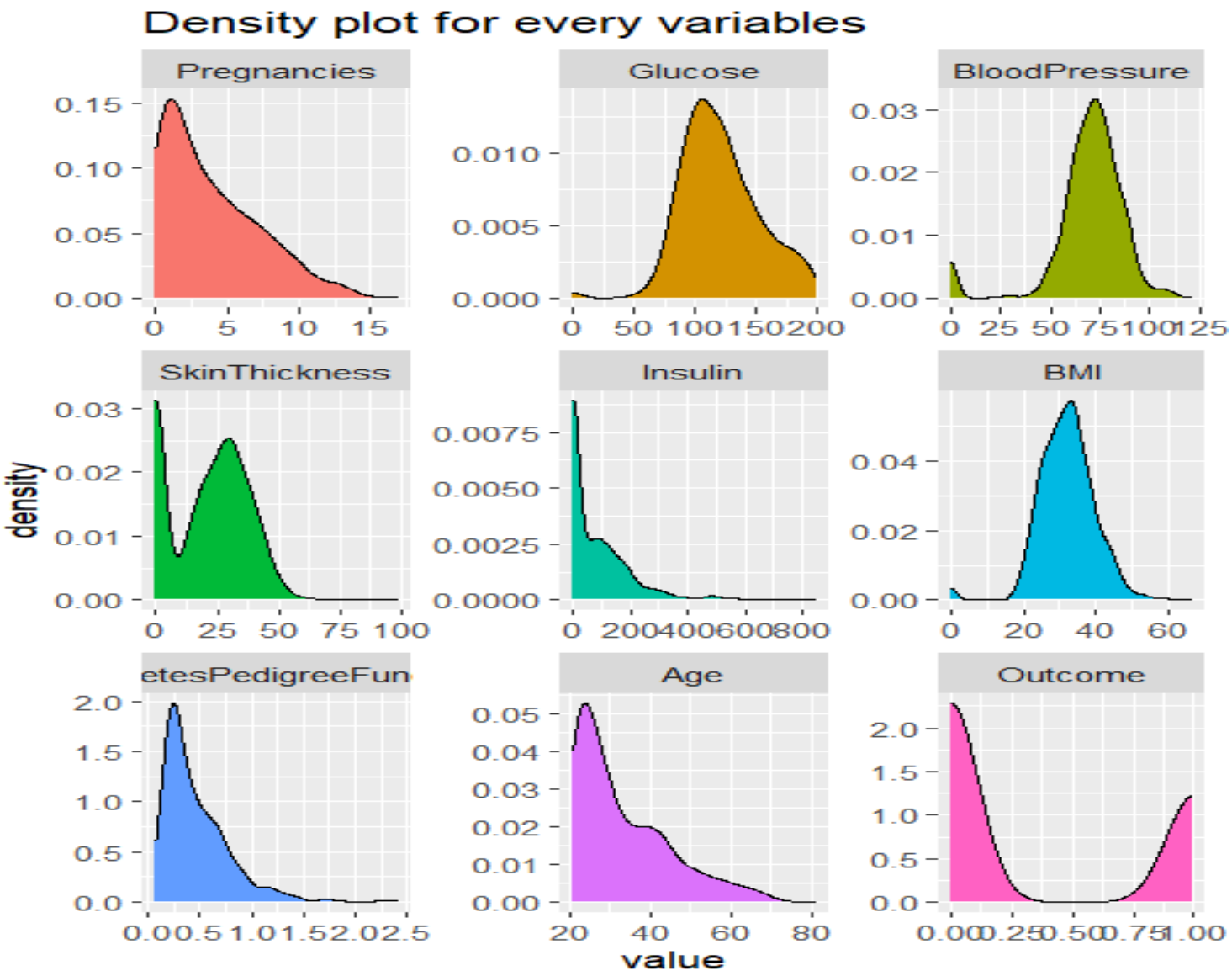
```
scale_x_discrete(limits=c(0,1)) +
ggtitle("Proportions of outcomes")
```



distribution of each variable

To gain a visual understanding of the dataset, we generate density plots for the variables. A density plot represents the distribution of values for each variable in a smooth and continuous manner. By creating density plots for the variables, we can observe the shape, spread, and concentration of the data. This visualization helps us identify patterns, potential outliers, and the overall distribution characteristics of each variable. Analyzing the variables' density plots provides valuable insights into the data's underlying structure and assists in making informed decisions during the analysis process.

```
# distribution of each variable
ggplot(melt(diabetes),aes(x=value,fill=variable)) +
 geom_density() +
 facet_wrap(~variable,scale="free") +
 theme(legend.position = "") +
 ggtitle("Density plot for every variables")
```

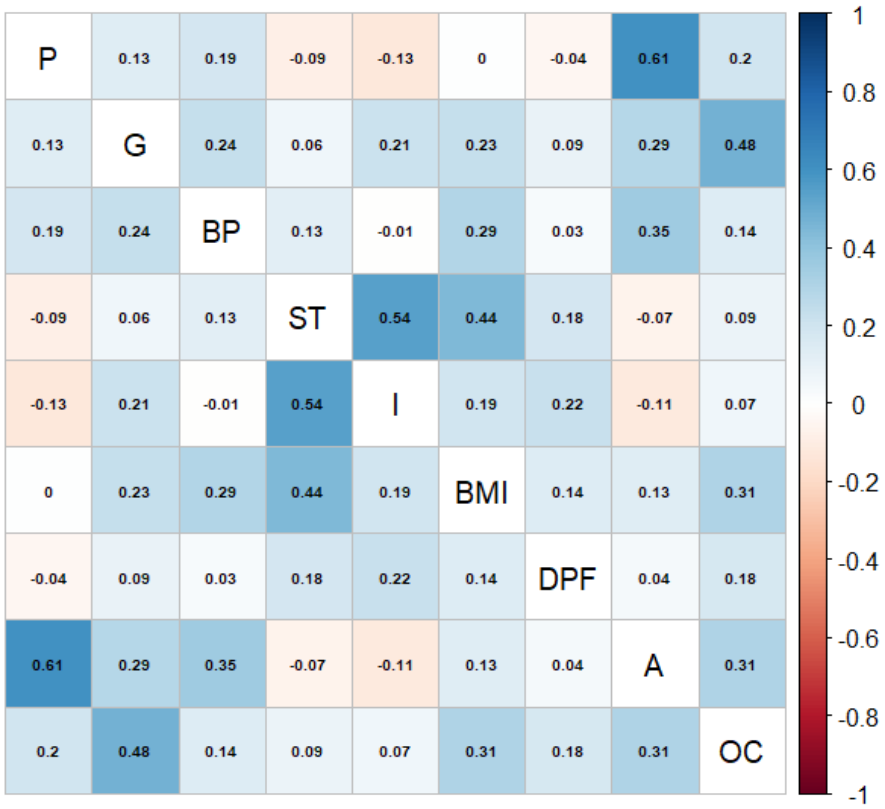


CORRELATION BETWEEN VARIABLES

To examine the relationships within the dataset, we calculate the correlation between variables. Correlation measures the strength and direction of the linear relationship between two variables. By computing the correlation between variables, we can determine the degree to which they are associated with each other. A positive correlation indicates that the variables tend to move in the same direction, while a negative correlation indicates they move in opposite directions. Analyzing the correlation between variables helps us identify potential dependencies or patterns, providing insights into the interconnectedness and potential predictive power of the variables in our analysis.

```
# correlation plot

diabetes_abb <- diabetes

diabetes_abb_names <- c("P","G","BP","ST","I","BMI","DPF","A","OC")

colnames(diabetes_abb) <- diabetes_abb
```

```
cor_diabetes <- cor(diabetes_abb,method=c("spearman"))
corrplot(cor_diabetes,
        method = 'square',
        diag = FALSE,
        tl.pos = "d",
        main="Correlation matrix")
```



any NaN in the set ?

```
sum(is.na(diabetes))
```

Based on these findings, we can extract the following information from the dataset:

- All variables in the dataset are numerical. The only issue we need to address is that the "Outcome" variable is currently numerical instead of a factor or categorical variable.
- The dataset is clean, with no missing values (NaN) present.
- Approximately one-third of the patients in the dataset are classified as sick.
- There doesn't seem to be any aberration or anomalies in the distribution of variables.
- The variables "Glucose," "BloodPressure," and "BMI" exhibit distributions that are close to normal, although extreme values in "Glucose" may contradict this observation.
- There are no negative correlations between variables.
- The correlation between variables and the "Outcome" variable ranges from 0.07 (indicating no significant correlation) to 0.48 (indicating moderate correlation).
- In conclusion, the dataset is clean with no data problems, and the absence of strong correlations between variables makes the model-building process interesting and challenging.

MODEL BUILDING

In order to proceed with our construction, it is necessary to divide our dataset into three separate parts:

- The train set: This portion of the data will be used to train our models. It is the primary dataset on which the models will learn the underlying patterns and relationships.
- The test set: This portion of the data will be utilized to evaluate the performance of our models. By applying the trained models to the test set, we can assess how well they generalize to unseen data and measure their predictive capabilities.
- The validation set: This subset of the data will be reserved for the final evaluation of our models. It helps us gauge the performance of the models after fine-tuning and hyperparameter optimization. The validation set aids in making informed decisions regarding the selection of the final model.

To achieve this division, we have implemented the following code:

```
# train/test sets creation

X <- diabetes %>% select(-Outcome)

y <- as.factor(diabetes$Outcome)

## Validation set

test_validation <- createDataPartition(y, times = 1, p = 0.9, list = FALSE)

validation <- diabetes[-test_validation, ]

test_index <- createDataPartition(y, times = 1, p = 0.8, list = FALSE)

## Train & Test set

train <- diabetes[test_index, ]

test <- diabetes[-test_index, ]


# Outcome as factor

train$Outcome <- as.factor(train$Outcome)

test$Outcome <- as.factor(test$Outcome)

validation$Outcome <- as.factor(validation$Outcome)
```

set length
train 615
test 153
validation 76

TRY OF A FIRST BASIC MODEL

```
# fitting
fit <- train(Outcome~.,
             data=train,
             method="glm")


# predict
pred <- predict(fit,test)


# confusion matrix
cm <- confusionMatrix(pred,test$Outcome,mode="everything",positive="1")


# accuracy
cm[3]$overall[1]
```

```
Accuracy
0.7712418
```

```
# f1-score
cm[4]$byClass[7]
```

```
0.6236559
```

For an initial model, without any optimization, the results are quite promising. With the methodology in place for model fitting and prediction, we can now proceed to create a function that generates a tibble containing performance metrics for the models on both the train and test sets.

This function will allow us to efficiently evaluate and compare the performance of different models. By obtaining metrics such as accuracy, precision, recall, and F1-score for each model on both the train and test sets, we can gain insights into their effectiveness in capturing patterns and making accurate predictions.

Creating this function streamlines the process of evaluating multiple models and provides a convenient summary of their respective performances on different datasets.

```
# Model creation
model_result <- function(method){

 # cross-validation control
 ctrl <- trainControl(method = "repeatedcv",
                      number = 10,
                      repeats = 3)

 # fitting
 fit <- train(Outcome~.,
              data=train,
              trControl = ctrl,
              method=method)

 # --- TEST ---
 # predict
 pred_test <- predict(fit,test)

 # confusion matrix
 cm_test <- confusionMatrix(pred_test,test$Outcome,mode="everything",positive="1")

 # accuracy
 accuracy_test <- round(cm_test[3]$overall[1],3)
 # f1-score
 f1_score_test <- round(cm_test[4]$byClass[7],3)

 # --- TRAIN ---
 # predict
 pred_train <- predict(fit,train)

 # confusion matrix
```

```
cm_train <- confusionMatrix(pred_train,train$Outcome,mode="everything",positive="1")


# accuracy

accuracy_train <- round(cm_train[3]$overall[1],3)

# f1-score

f1_score_train <- round(cm_train[4]$byClass[7],3)




# tibble with results

result <- tibble("model"=method,

                 "train_acc"=accuracy_train,

                 "test_acc"=accuracy_test,

                 "acc_diff"=round(abs(accuracy_train-accuracy_test),5),

                 "train_f1"=f1_score_train,

                 "test_f1"=f1_score_test,

                 "f1_diff"=round(abs(f1_score_train-f1_score_test),5)

                 )

return(result)
```

In our quest to find the best model, we will experiment with various models from a predefined list. We have selected five popular binary classification models from the caret package, namely:

1. svmLinear: Support Vector Machines with a Linear Kernel.
2. rf: Random Forest.
3. glm: Generalized Linear Model (initial attempt).
4. glmboost: Boosted Generalized Linear Model (tunable glm).
5. lda: Linear Discriminant Analysis.

We will now proceed to analyze and evaluate the performance metrics returned by these models. Please note that this step may take a few minutes to complete as we carefully assess the capabilities and suitability of each model for our specific task.

```
## model train_acc test_acc acc_diff train_f1 test_f1 f1_diff ratio

## 3 glm 0.771 0.771 0.000 0.628 0.660 0.032 3.86

## 5 lda 0.771 0.778 0.007 0.630 0.667 0.037 3.76
```

4 glmboost 0.769 0.765 0.004 0.628 0.640 0.012 3.75

1 svmLinear 0.774 0.784 0.010 0.627 0.673 0.046 3.73

2 rf 1.000 0.752 0.248 1.000 0.612 0.388 1.68

Based on the information you provided, it seems that you have a results table with columns labeled "diff" and "ratio" in addition to train and test values of a metric. Here's an explanation of what these columns represent:

1. "Diff" Columns: These columns represent the differences between the train and test values of the metric. The purpose of including these columns is to help identify cases of overfitting. Overfitting occurs when a model performs well on the training data but fails to generalize to unseen data (test data). By calculating the difference between the train and test values, you can get an idea of how much the model's performance is affected by overfitting. If the differences are significant, it suggests that the model may not be able to generalize well.
2. "Ratio" Columns: These columns are a homemade variable that aims to determine the best model based on the accuracy rate and the ratio. While you haven't provided specific details about how this ratio is calculated, it appears to be a measure of model performance that takes into account the accuracy rate and another factor, possibly the difference between train and test values. The purpose of this homemade variable is to provide a holistic view of model performance and assist in determining which model is the best among the alternatives.

In summary, the "diff" columns help identify potential overfitting cases by comparing the performance of the model on training and test data, while the "ratio" columns are a custom metric that combines accuracy rate and another factor to assess the relative performance of different models.

In summary, with the exception of "rf" (Random Forest), all the models show results within a similar range. Considering that "glm" (Generalized Linear Model) and "lda" (Linear Discriminant Analysis) do not have tunable values, we need to decide between "glmboost" and "svmLinear" models. To make an informed decision, we will proceed with tuning both models to determine which one performs better.

TUNING GLMBOOST

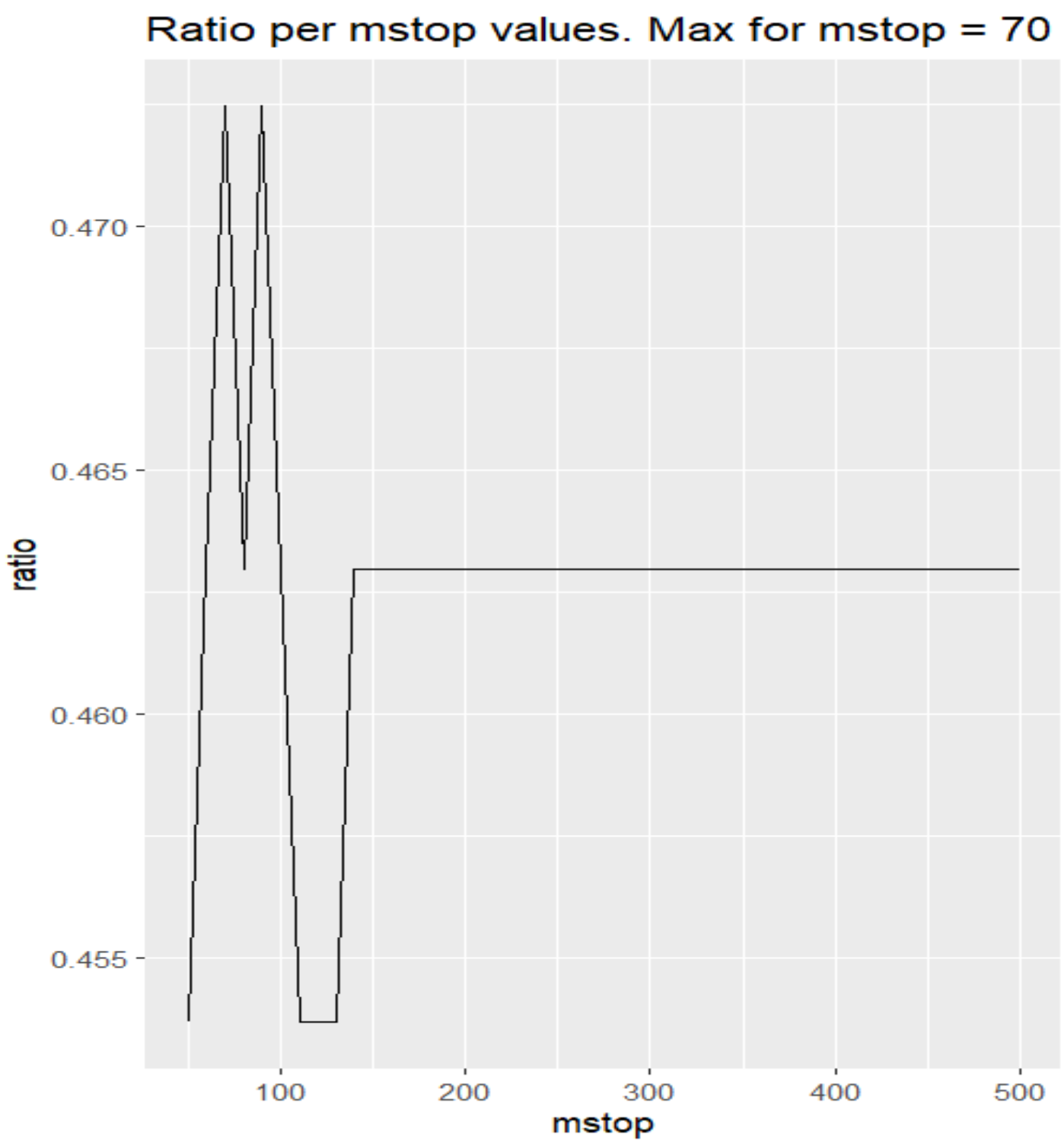
To optimize the glmBoost method, we need to tune two parameters: mstop and prune. The prune parameter is a binary value that, when set to "yes," automatically selects an appropriate mstop value. However, since we want to manually tune this parameter, we will set it as "no."

For the mstop parameter, we will explore a range from 50 to 500 with increments of 10. To assess the performance of different mstop values, we will introduce a new metric called "ratio," which is calculated as the product of accuracy and f1-score. This metric allows us to identify the best trade-off between accuracy and f1-score by plotting the ratio against the mstop values. The higher the ratio, the better the model performance.

By examining the ratio values across different mstop values, we can determine the optimal mstop value that maximizes the ratio and achieves the desired balance between accuracy and f1-score.

```
# tuning plot

mstops_tuning <- data.frame(tibble("mstop"=mstops,"accuracy"=accuracies,"f1_score"=f1_scores))

mstops_tuning <- as.data.frame(lapply(mstops_tuning, unlist))

mstops_tuning <- mstops_tuning %>% mutate(ratio = (accuracy*f1_score))
```

```
ggplot(mstops_tuning,aes(x=mstop,y=ratio)) +
 geom_line() +
 ggtitle(paste("Ratio per mstop values. Max for mstop =",
              mstops_tuning$mstop[which.max(mstops_tuning$ratio)]))
```



We have the mstop value that maximize the ratio, we can now plot the result for an optimized glmboost model :

```
# Final model testing
grid_opt <- expand.grid(mstop = mstops_tuning$mstop[which.max(mstops_tuning$ratio)],
                        prune = 'no')

fit_opt <- train(Outcome~.,
                 data=train,
                 trControl=ctrl,
                 method="glmboost",
                 tuneGrid=grid_opt)

pred_opt <- predict(fit_opt,test)

cm_opt <- confusionMatrix(pred_opt,
                          test$Outcome,
                          mode="everything",
                          positive="1")

glm_opt_result <- data.frame(tibble("model"    = "glmBoost",
                                    "accuracy" = round(cm_opt[3]$overall[1],3),
                                    "f1_score" = round(cm_opt[4]$byClass[7],3)))
print(glm_opt_result)
```

```
     model accuracy f1_score
1 glmBoost    0.784    0.602
```

The accuracy of the glmBoost model is moderate, and the f1-score is not significantly higher compared to the other models we have tested. To explore the potential for improved performance, we will proceed with tuning the svmLinear model and evaluate if it yields better results.

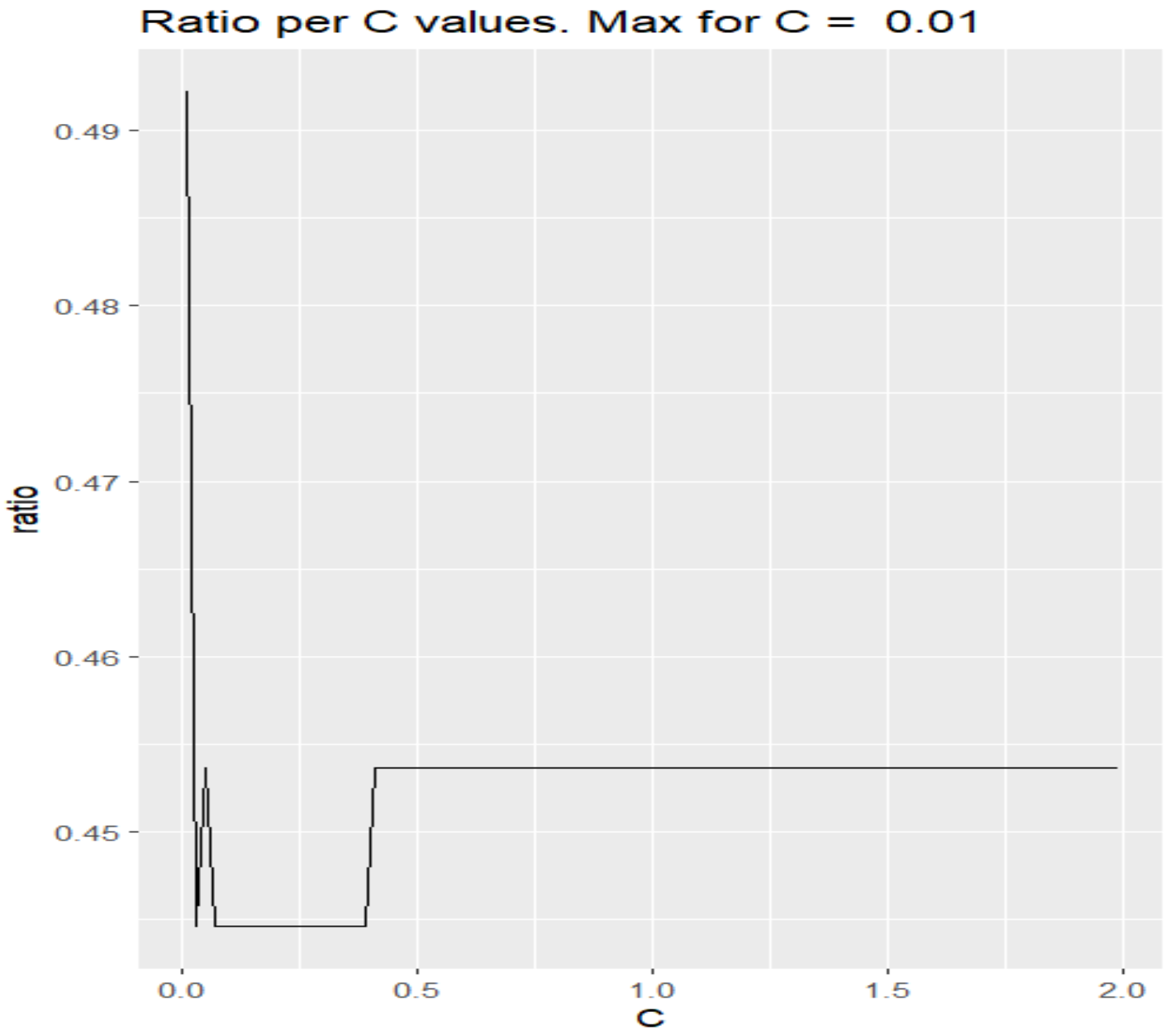
TUNING SVM

The tuning process for svmLinear will follow a similar approach to glmBoost. However, there is only one parameter to tune in svmLinear, which is C. We will explore a range of values from 0.01 to 2 with a step size of 0.02 to determine the optimal C value.

Similar to before, we will plot the ratio metric, which is calculated as the product of accuracy and f1-score, against the different C values. This will allow us to assess the model's performance for each C value and identify the C value that maximizes the ratio.

By examining the ratio values across the range of C values, we can determine the most suitable C value that leads to the best compromise between accuracy and f1-score in the svmLinear model.

model	accuracy	f1_score
1 svmLinear	0.797	0.617



```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 45 10
         1  5 16

               Accuracy : 0.8026
                 95% CI : (0.6954, 0.8851)
    No Information Rate : 0.6579
    P-Value [Acc > NIR] : 0.004209

                  Kappa : 0.5403

 Mcnemar's Test P-Value : 0.301700

            Sensitivity : 0.6154
            Specificity : 0.9000
         Pos Pred Value : 0.7619
         Neg Pred Value : 0.8182
              Precision : 0.7619
                 Recall : 0.6154
                     F1 : 0.6809
             Prevalence : 0.3421
         Detection Rate : 0.2105
   Detection Prevalence : 0.2763
      Balanced Accuracy : 0.7577

       'Positive' Class : 1
```

FINAL RESULTS

```
# final results

final_results <- bind_rows(glm_opt_result,svm_opt_result) %>% knitr::kable()

print(final_results)
```

```
|model     | accuracy| f1_score|
|:---------|--------:|--------:|
|glmBoost  |    0.784|    0.602|
|svmLinear |    0.797|    0.617|
```

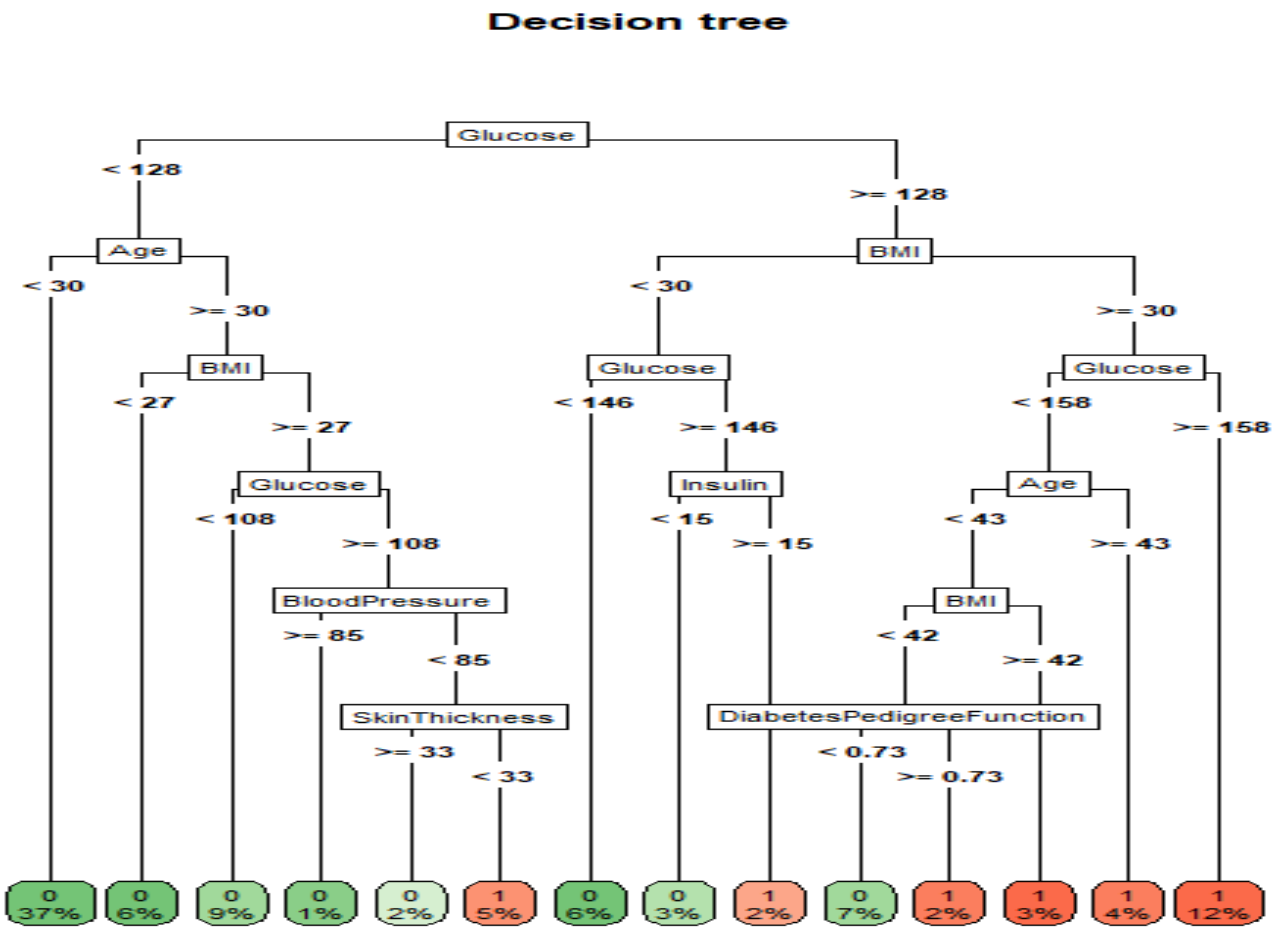
Based on our experimentation with various models, svmLinear appears to be the most promising. However, it is worth noting that while glmBoost demonstrated improvements in its metrics through tuning, svmLinear's performance remained consistent.

To gain a comprehensive understanding of the svmLinear model's performance, we will recreate the model from scratch. By doing so, we can gather all the available information and insights regarding the model's performance. This will allow us to assess its performance in its default configuration and explore any additional details that can be extracted from the model.

DECISION TREE

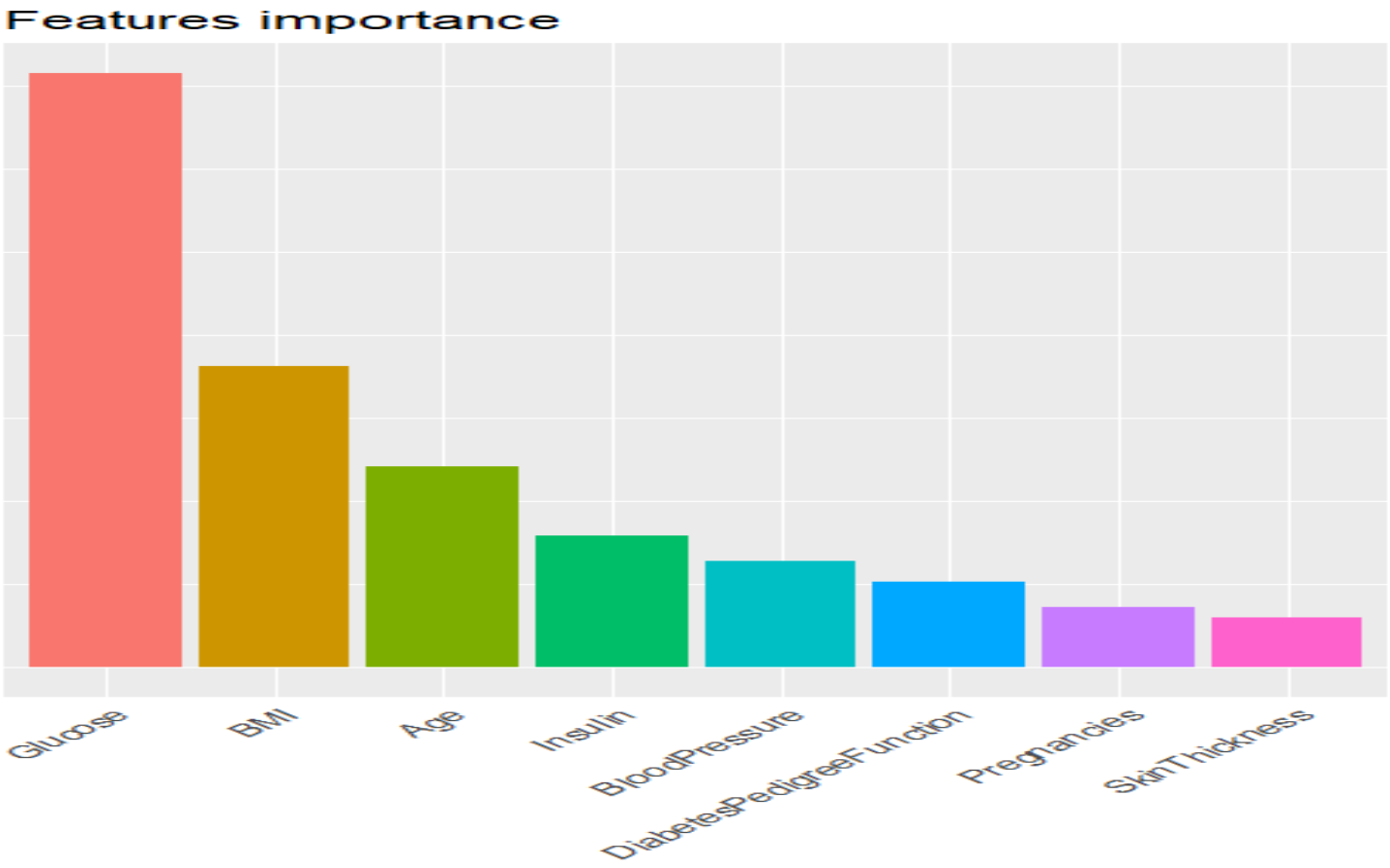
Bonus : decision tree

```
tree <- rpart(Outcome~.,data=train)

rpart.plot(tree,
           type = 5,
           extra = 100,
           box.palette = "GnRd",
           main="Decision tree")



importances <- data.frame(tree$variable.importance)

feature_importance <- data.frame(tibble("feature"=rownames(importances),
                                        "importance"=importances$tree.variable.importance))


feature_importance <- feature_importance[order(desc(feature_importance$importance)),]
```



FEATURES IMPORTANCE

```
# feature importance plot
feature_importance %>%
 arrange(desc(importance)) %>%
 mutate(feature=factor(feature,levels=feature)) %>%
 ggplot(aes(y=importance,x=feature,fill=feature)) +
 geom_col() +
 theme(axis.ticks.y = element_blank(),
       axis.text.y = element_blank(),
       axis.ticks.x = element_blank(),
       axis.title.x = element_blank(),
       axis.title.y = element_blank(),
       legend.position = "none",
       axis.text.x = element_text(angle=45
                                  ,hjust=1))+
 ggtitle("Features importance")
```



CONCLUSION :

In summary, from a machine learning perspective, the final model we have developed shows promising metrics, indicating its usefulness. However, in the context of a medical case, there are certain limitations that prevent its practical application. Here are a few key points to explain this:

1. Limited Data: Although the dataset is usable, for a highly accurate and precise project, a larger dataset would be required. The current dataset is limited in size and scope, which may affect the model's ability to generalize well to new cases.
2. Niche Target Audience: The dataset specifically focuses on women from a specific area, making it applicable to a relatively narrow population. This restricts the model's usability in broader medical contexts.
3. Weak Variable Correlation: The variables in the dataset do not exhibit strong correlations with the outcome, which poses a challenge for achieving higher predictive accuracy. This lack of strong relationships between the variables and the outcome hinders the model's performance.

Furthermore, to improve the prediction model, employing Deep Learning and Neural Networks could be considered. However, this would require advanced knowledge and expertise beyond traditional machine learning techniques. Deep Learning and Neural Networks operate at a different level of data science and extend beyond the scope of conventional machine learning approaches.