HarvardX Capstone Project: Choose Your Own

Diabetes Risk Classification Using Machine Learning Algorithms: The Behavioural Risk Factor Surveillance System

Francis Dzakpasu

2025-11-07

Summary

Diabetes is a major chronic disease with a high economic burden. Early identification of risk factors and diagnosis is important for the prevention and management of diabetes. Machine Learning techniques have emerged as an important and reliable classification tool for chronic diseases, including diabetes. This project used supervised Machine Learning methods to train algorithms for diabetes risk classification. A large dataset, the Behavioural Risk Factor Surveillance System (BRFSS), collected by the Centres for Disease Control and Prevention (CDC), was used. Specifically, a subset of the dataset, comprising the diabetes_binary and diabetes_split, was used. The diabetes_binary dataset was partitioned into training and test sets, and the diabetes_split dataset was used as a validation set. The Machine Learning approaches utilised to build the algorithms included *Logistic Regression, K-Nearest Neighbour (KNN), Decision Tree, and Random Forest*. The variable importance of the trained models was similar across the Machine Learning algorithms, comprising mainly features that are common risk factors of diabetes, including age, high blood pressure, high cholesterol, BMI, and heavy alcohol consumption. The performance of the trained algorithms on the testing dataset was higher (Random Forest, highest: 0.8625) but lower on the validation dataset (Decision Tree, highest: 0.5741). There was little difference in the observed accuracy of the trained algorithms on the test dataset, and it is consistent with those reported in the literature. There are other Machine Learning algorithms, e.g. Support Vector Machine (SVM), which can perform better in classification tasks but are computationally intensive and were not considered.

Introduction

The global prevalence of diabetes is high, and it contributes substantially to the burden of chronic diseases[1]. Over 830 million people in the world live with diabetes, which is a major risk factor for cardiovascular disease and the associated healthcare costs[2]. In the United States of America, the economic burden of diabetes affects millions of people each year. According to the Centres for Disease Control and Prevention (CDC) data, about 34.4 million Americans live with diabetes, and 97.6 million with prediabetes in 2021, and an estimated 1 in 5 people with diabetes, whereas 8 in 10 with prediabetes are unaware of their condition. As a result, there are massive burdens on the economy, with diagnosed diabetes costing about 327 billion dollars and undiagnosed diabetes and prediabetes approximately 400 billion dollars annually[3].

[1] Chen, X., Zhang, L., & Chen, W. (2025): Global, regional, and national burdens of type 1 and type 2 diabetes mellitus in adolescents from 1990 to 2021, with forecasts to 2030: a systematic analysis of the global burden of disease study 2021. BMC medicine, 23(1), 48. https://doi.org/10.1186/s12916-025-03890-w

[2] World Health Organisation, Diabetes. Available at: https://www.who.int/news-room/fact-sheets/detail/diabetes

[3] CDC: Diabetes, National Diabetes Statistics Report. https://www.cdc.gov/diabetes/php/data-research/index.html

Diabetes is a debilitating chronic disease characterised by the inability of the body to effectively regulate blood glucose levels, which can lead to increased risk of cardiovascular diseases, chronic kidney disease, and complications that can reduce quality of life and life expectancy. There are different types of diabetes; however, type 2 diabetes is the most common, accounting for about 97% of all cases. The prevalence is influenced by sociodemographic and other social determinants of health, as well as lifestyle and health behaviour factors, with the highest burden among those of lower socioeconomic status[4]. The risk increases significantly with age and a high level of adiposity[5].

Early identification of risk factors and diagnosis of diabetes are crucial for the prevention and effective management of the condition. Machine Learning techniques have emerged as an effective tool for predicting and classifying chronic diseases, such as cardiovascular disease, hypertension, and diabetes[6]. Several Machine Learning techniques have been used to develop algorithms capable of accurately classifying some health conditions. For example, clinicians use algorithms to predict 5-year cardiovascular disease risk, cancer prognosis, etc.[7]. These Machine Learning algorithms can support intelligent clinical and public health decisions and improve management and prevention strategies for chronic diseases[8].

This HarvardX Professional Certificate in Data Science Capstone Project – Choose Your Own – aimed to build Machine Learning algorithms to classify diabetes based on demographic, clinical, lifestyle and health behaviours features using a large dataset from the Behavioural Risk Factor Surveillance System (BRFSS) available at the CDC. Supervised Machine Learning approaches were applied to train the algorithms, examining the contributions of variables (features) to diabetes risk classification and the algorithms' accuracy.

Methods

Overview of the datasets

The Behavioural Risk Factor Surveillance System (BRFSS) is an annual health-related telephone survey collected by the CDC. The BRFSS conducts state-based random–digit–dialled telephone surveys in the 50 states, District of Columbia (DC), Puerto Rico and other U.S. territories. BRFSS respondents were classified as having diagnosed diabetes if they answered YES to the question – *Has a doctor ever told you that you have diabetes?* – Women with only gestational diabetes during pregnancy were classified as not having diabetes. BRFSS data were weighted to estimate the number of noninstitutionalized persons aged 18 years with diagnosed diabetes in each state, DC, and Puerto Rico[9]. Since 1984, responses from over 400,000 Americans on health-related risk behaviours, chronic health conditions, and the use of preventative services have been collected annually. The original dataset contains 441,455 participants, with 330 features. The features are either survey questions or calculated variables derived from survey question responses. A subset of the BRFSS dataset available on Kaggle from 2015, the Diabetes Health Indicators Dataset, which has been cleaned and contains 21 features, was utilised in this project[10]. The Diabetes Health Indicators Dataset contains 3 files:

[4]Zhou, Bin et al. (2024): Worldwide trends in diabetes prevalence and treatment from 1990 to 2022: a pooled analysis of 1108 population-representative studies with 141 million participants. The Lancet, 404(10467):2077 – 2093

[5]Hossain MJ, Al-Mamun M, Islam MR. (2024): Diabetes mellitus, the fastest growing global public health concern: Early detection should be focused. Health Sci Rep. 7(3):e2004. doi: 10.1002/hsr2.2004.

[6]Abnoosian, K., Farnoosh, R. & Behzadi, M.H. (2023): Prediction of diabetes disease using an ensemble of machine learning multi-classifier models. BMC Bioinformatics 24: 337. https://doi.org/10.1186/s12859-023-05465-z

[7]Zhang, B., Shi, H., & Wang, H. (2023). Machine Learning and AI in Cancer Prognosis, Prediction, and Treatment Selection: A Critical Approach. Journal of multidisciplinary healthcare, 16, 1779–1791. https://doi.org/10.2147/JMDH.S410301

[8]Alanazi R. (2022): Identification and Prediction of Chronic Diseases Using Machine Learning Approach. J Healthc Eng. 2022:2826127. doi: 10.1155/2022/2826127.

[9]Burrows NR, Hora I, Geiss LS, Gregg EW, Albright A. (2017): Incidence of End-Stage Renal Disease Attributed to Diabetes Among Persons with Diagnosed Diabetes — United States and Puerto Rico, 2000–2014. MMWR Morb Mortal Wkly Rep, 66:1165–1170. DOI: http://dx.doi.org/10.15585/mmwr.mm6643a2

[10]Alex Teboul (2021): Diabetes Health Indicators Dataset; Kaggle. Available at https://www.kaggle.com/datasets/alexteboul/diabetes-health-indicators-dataset

1. *diabetes_012_health_indicators_BRFSS2015.csv:* is a clean dataset of 253,680 survey responses to the CDC's BRFSS2015. The target variable Diabetes_012 has 3 classes. 0 is for no diabetes or only during pregnancy, 1 is for prediabetes, and 2 is for diabetes. There is a class imbalance in this dataset. This dataset has 21 feature variables

2. *diabetes_binary_5050split_health_indicators_BRFSS2015.csv:* is a clean dataset of 70,692 survey responses to the CDC's BRFSS2015. It has an equal 50-50 split of respondents with no diabetes and with either prediabetes or diabetes. The target variable Diabetes_binary has 2 classes. 0 is for no diabetes, and 1 is for prediabetes or diabetes. This dataset has 21 feature variables and is balanced.

3. *diabetes_binary_health_indicators_BRFSS2015.csv:* is a clean dataset of 253,680 survey responses to the CDC's BRFSS2015. The target variable Diabetes_binary has 2 classes. 0 is for no diabetes, and 1 is for prediabetes or diabetes. This dataset has 21 feature variables and is not balanced.

The second and third files were used for the Machine Learning project. The dataset in the third file (*diabetes _ binary _ health _ indicators _ BRFSS2015.csv*) has a binary diabetes variable, and the prevalence of diabetes risk (diabetes and prediabetes) is realistic to the expected prevalence in the general population. Therefore, this dataset was used for building the algorithms. The second file (*diabetes _ binary _ 5050split _ health _ indicators _ BRFSS2015.csv:*), which also has a binary diabetes variable, but an equal proportion of diabetes risk (pre-diabetes and diabetes) and 'no diabetes', illustrates an unrealistic population prevalence of diabetes risk and was used as a validation dataset to verify the performance of the trained classification algorithms in a non-ideal population.

Data downloading and cleaning

The Diabetes Health Indicators Dataset, a subset of the BRFSS dataset from CDC, available on Kaggle, cannot be automatically downloaded from the website through the R code. Hence, the dataset was initially downloaded and uploaded to GitHub to enable automatic download through the R code. The *diabetes _ binary _ health _ indicators _ BRFSS2015.csv* (named: diabetes_binary) and the *diabetes _ binary _ 5050split _ health _ indicators _ BRFSS2015.csv* (named: diabetes_split) were further cleaned to remove some features to minimise the complexity and the Machine Learning computation time. The cleaned datasets used have the same number of features – nine features – and were named diabetes_df and diabetes_val for diabetes_binary and diabetes_split, respectively. The diabetes_df dataset was partitioned into training and testing sets. The Machine Learning algorithms were built using the training set and were tested on the testing set. The features (variables) in the datasets used for the Machine Learning algorithms are:

- *Diabetes_binary:* coded 0 = no diabetes; and 1 = diabetes (prediabetes or diabetes)
- *HighBP:* coded 0 = no high blood pressure (BP); and 1 = high BP
- *HighChol:* coded 0 = no high cholesterol; and 1 = high cholesterol
- *BMI:* body mass index (a continuous variable)
- *Smoker:* Have you smoked at least 100 cigarettes in your entire life? [Note: 5 packs = 100 cigarettes], which was coded 0 = no; and 1 = yes
- *PhysActivity:* physical activity in past 30 days, not including job, was coded 0 = no; and 1 = yes
- *HvyAlcoholConsump:* (adult men >=14 drinks per week and adult women>=7 drinks per week) was coded 0 = no and; 1 = yes
- *Sex:* coded 0 = female; and 1 = male
- *Age:* 13-level age category 1 = 18-24years; 2 = 25-29; 3 = 30-34; 4 = 35-39; 5 = 40-44; 6 = 45-49; 7 = 50-54; 8 = 55-59; 9 = 60-64; 10 =65-69; 11 = 70-74; 12 = 75-79; and 13 = 80 years or older

- *Education:* Education level scale 1-6: coded 1 = Never attended school or only kindergarten; 2 = Grades 1 through 8 (Elementary); 3 = Grades 9 through 11 (Some high school); 4 = Grade 12 or GED (High school graduate); 5 = College 1 year to 3 years (Some college or technical school); and 6 = College 4 years or more (College graduate)

First of foremost, the required packages were installed and the libraries loaded.

```
## Installing required packages

if(!require(rlang)) install.packages("rlang",
                                     repos = "http://cran.us.r-project.org")
if(!require(tidyverse)) install.packages("tidyverse",
                                         repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret",
                                     repos = "http://cran.us.r-project.org")
if(!require(Metrics)) install.packages("Metrics",
                                       repos = "http://cran.us.r-project.org")
if(!require(kableExtra)) install.packages("kableExtra",
                                         repos = "http://cran.us.r-project.org")
if(!require(recosystem)) install.packages("recosystem",
                                         repos = "http://cran.us.r-project.org")
if(!require(ggthemes)) install.packages("ggthemes",
                                       repos = "http://cran.us.r-project.org")
if(!require(R.utils)) install.packages("R.utils",
                                       repos = "http://cran.us.r-project.org")
if(!require(tictoc)) install.packages("tictoc",
                                     repos = "http://cran.us.r-project.org")
if(!require(tidymodels)) install.packages("tidymodels",
                                         repos = "http://cran.us.r-project.org")
if(!require(probably)) install.packages("probably",
                                       repos = "http://cran.us.r-project.org")
if(!require(rpart.plot)) install.packages("rpart.plot",
                                         repos = "http://cran.us.r-project.org")
if(!require(ranger)) install.packages("ranger",
                                     repos = "http://cran.us.r-project.org")
if(!require(vip)) install.packages("vip",
                                   repos = "http://cran.us.r-project.org")
if(!require(cluster)) install.packages("cluster",
                                       repos = "http://cran.us.r-project.org")
if(!require(purrr)) install.packages("purrr",
                                     repos = "http://cran.us.r-project.org")
if(!require(dplyr)) install.packages("dplyr",
                                     repos = "http://cran.us.r-project.org")
if(!require(readr)) install.packages("readr",
                                     repos = "http://cran.us.r-project.org")
if(!require(glmnet)) install.packages("glmnet",
                                      repos = "http://cran.us.r-project.org")
if(!require(kknn)) install.packages("kknn",
                                    repos = "http://cran.us.r-project.org")
if(!require(MASS)) install.packages("MASS",
                                    repos = "http://cran.us.r-project.org")
```

Note that some of the base packages required may need updating for the models to run.

```r
## Load required libraries
#update.packages()

library(tidyverse)
library(caret)
library(broom)
library(lubridate)
library(gam)
library(randomForest)
library(Rborist)
library(matrixStats)
library(rafalib)
library(naivebayes)

library(ggplot2)
library(ggthemes)
library(scales)
library(dslabs)
ds_theme_set()
library(knitr)
library(kableExtra)
library(recosystem)
library(R.utils)
library(tictoc)

library(dplyr)
library(readr)
library(tidymodels)
library(probably)
library(rpart.plot)
library(ranger)
library(vip)
library(cluster)
library(purrr)
tidymodels_prefer()

library(glmnet)
library(kknn)
library(MASS)
```

Note that the datasets cannot be downloaded automatically from the source into R. Therefore, the dataset has been downloaded and uploaded onto GitHub to enable automatic download with the R code. The codes for downloading and cleaning the datasets are shown below:

```r
##Downloading the CDC Diabetes Health Indicators Dataset
#https://archive.ics.uci.edu/dataset/891/cdc+diabetes+health+indicators
#https://www.kaggle.com/datasets/alexteboul/diabetes-health-indicators-dataset?resource=download

##Download and unzip the dataset
db <- "archive_diabetes health indicators dataset.zip"

download.file(
"https://github.com/fdzakpasu/dfedx_capstone-project-choose-your-own/raw/refs/heads/main/archive_diabet
```

```
  db)

db <- unzip(db)

db_binary <- read.csv(
"./archive_diabetes health indicators dataset/diabetes_binary_health_indicators_BRFSS2015.csv")

db_split <- read.csv(
"./archive_diabetes health indicators dataset/diabetes_binary_5050split_health_indicators_BRFSS2015.csv

rm(db)

#Renaming the datasets
diabetes_binary <- as.data.frame(db_binary)
glimpse(diabetes_binary)
```

```
## Rows: 253,680
## Columns: 22
## $ Diabetes_binary      <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0~
## $ HighBP               <dbl> 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1~
## $ HighChol             <dbl> 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1~
## $ CholCheck            <dbl> 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
## $ BMI                  <dbl> 40, 25, 28, 27, 24, 25, 30, 25, 30, 24, 25, 34, 2~
## $ Smoker               <dbl> 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0~
## $ Stroke               <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0~
## $ HeartDiseaseorAttack <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0~
## $ PhysActivity         <dbl> 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1~
## $ Fruits               <dbl> 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1~
## $ Veggies              <dbl> 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1~
## $ HvyAlcoholConsump    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ AnyHealthcare        <dbl> 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
## $ NoDocbcCost          <dbl> 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0~
## $ GenHlth              <dbl> 5, 3, 5, 2, 2, 2, 3, 3, 5, 2, 3, 3, 3, 4, 4, 2, 3~
## $ MentHlth             <dbl> 18, 0, 30, 0, 3, 0, 0, 0, 30, 0, 0, 0, 0, 0, 30, ~
## $ PhysHlth             <dbl> 15, 0, 30, 0, 0, 2, 14, 0, 30, 0, 0, 30, 15, 0, 2~
## $ DiffWalk             <dbl> 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0~
## $ Sex                  <dbl> 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0~
## $ Age                  <dbl> 9, 7, 9, 11, 11, 10, 9, 11, 9, 8, 13, 10, 7, 11, ~
## $ Education            <dbl> 4, 6, 4, 3, 5, 6, 6, 4, 5, 4, 6, 5, 5, 4, 6, 6, 4~
## $ Income               <dbl> 3, 1, 8, 6, 4, 8, 7, 4, 1, 3, 8, 1, 7, 6, 2, 8, 3~
```

```
diabetes_split <- as.data.frame(db_split)
glimpse(diabetes_split)
```

```
## Rows: 70,692
## Columns: 22
## $ Diabetes_binary      <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ HighBP               <dbl> 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0~
## $ HighChol             <dbl> 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0~
## $ CholCheck            <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
## $ BMI                  <dbl> 26, 26, 26, 28, 29, 18, 26, 31, 32, 27, 24, 21, 2~
## $ Smoker               <dbl> 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0~
```

```
## $ Stroke               <dbl> 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ HeartDiseaseorAttack <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0~
## $ PhysActivity         <dbl> 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1~
## $ Fruits               <dbl> 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0~
## $ Veggies              <dbl> 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1~
## $ HvyAlcoholConsump    <dbl> 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ AnyHealthcare        <dbl> 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
## $ NoDocbcCost          <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ GenHlth              <dbl> 3, 3, 1, 3, 2, 2, 1, 4, 3, 3, 3, 1, 2, 3, 1, 3, 2~
## $ MentHlth             <dbl> 5, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0~
## $ PhysHlth             <dbl> 30, 0, 10, 3, 0, 0, 0, 0, 0, 6, 4, 0, 0, 3, 0, 0,~
## $ DiffWalk             <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0~
## $ Sex                  <dbl> 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0~
## $ Age                  <dbl> 4, 12, 13, 11, 8, 1, 13, 6, 3, 6, 12, 4, 7, 10, 1~
## $ Education            <dbl> 6, 6, 6, 6, 5, 4, 5, 4, 6, 4, 4, 6, 6, 4, 5, 4, 5~
## $ Income               <dbl> 8, 8, 8, 8, 8, 7, 6, 3, 8, 4, 6, 8, 8, 6, 1, 6, 7~
```

To make the modelling less complex and minimise the computation time, the diabetes_binary and diabetes_split datasets were further cleaned to select a few variables (features).

```
## Cleaning the datasets and setting categorical variables as factors and their levels

# diabetes_binary dataset will be further split into training and testing sets
diabetes_df <- diabetes_binary %>% select("Diabetes_binary", "HighBP", "HighChol", "BMI",
                                          "PhysActivity", "Sex", "Age", "Smoker",
                                          "HvyAlcoholConsump", "Education") %>%
  mutate(Diabetes_binary = relevel(factor(Diabetes_binary), ref="0",),
         HighBP = relevel(factor(HighBP), ref="0"),
         HighChol = relevel(factor(HighChol), ref="0"),
         PhysActivity = relevel(factor(PhysActivity), ref="0"),
         Sex = relevel(factor(Sex), ref="0"),
         Age = relevel(factor(Age), ref="1"),
         Smoker = relevel(factor(Smoker), ref="0"),
         HvyAlcoholConsump = relevel(factor(HvyAlcoholConsump), ref="0"),
         Education = relevel(factor(Education), ref="1"))
str(diabetes_df)
```

```
## 'data.frame':    253680 obs. of  10 variables:
##  $ Diabetes_binary  : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 2 1 ...
##  $ HighBP           : Factor w/ 2 levels "0","1": 2 1 2 2 2 2 2 2 2 1 ...
##  $ HighChol         : Factor w/ 2 levels "0","1": 2 1 2 1 2 2 1 2 2 1 ...
##  $ BMI              : num  40 25 28 27 24 25 30 25 30 24 ...
##  $ PhysActivity     : Factor w/ 2 levels "0","1": 1 2 1 2 2 2 1 2 1 1 ...
##  $ Sex              : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 1 1 1 2 ...
##  $ Age              : Factor w/ 13 levels "1","2","3","4",..: 9 7 9 11 11 10 9 11 9 8 ...
##  $ Smoker           : Factor w/ 2 levels "0","1": 2 2 1 1 1 2 2 2 2 1 ...
##  $ HvyAlcoholConsump: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Education        : Factor w/ 6 levels "1","2","3","4",..: 4 6 4 3 5 6 6 4 5 4 ...
```

```
### diabetes_split cleaning (Validation dataset)
diabetes_val <- diabetes_split %>%select("Diabetes_binary", "HighBP", "HighChol", "BMI",
                                         "PhysActivity", "Sex", "Age","Smoker",
                                         "HvyAlcoholConsump", "Education") %>%
```

```
  mutate(Diabetes_binary = relevel(factor(Diabetes_binary), ref="0",),
         HighBP = relevel(factor(HighBP), ref="0"),
         HighChol = relevel(factor(HighChol), ref="0"),
         PhysActivity = relevel(factor(PhysActivity), ref="0"),
         Sex = relevel(factor(Sex), ref="0"),
         Age = relevel(factor(Age), ref="1"),
         Smoker = relevel(factor(Smoker), ref="0"),
         HvyAlcoholConsump = relevel(factor(HvyAlcoholConsump), ref="0"),
         Education = relevel(factor(Education), ref="1"))
str(diabetes_val)
```

```
## 'data.frame':    70692 obs. of  10 variables:
##  $ Diabetes_binary  : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ HighBP           : Factor w/ 2 levels "0","1": 2 2 1 2 1 1 1 1 1 1 ...
##  $ HighChol         : Factor w/ 2 levels "0","1": 1 2 1 2 1 1 2 1 1 1 ...
##  $ BMI              : num  26 26 26 28 29 18 26 31 32 27 ...
##  $ PhysActivity     : Factor w/ 2 levels "0","1": 2 1 2 2 2 2 2 1 2 1 ...
##  $ Sex              : Factor w/ 2 levels "0","1": 2 2 2 2 1 1 2 2 1 2 ...
##  $ Age              : Factor w/ 13 levels "1","2","3","4",..: 4 12 13 11 8 1 13 6 3 6 ...
##  $ Smoker           : Factor w/ 2 levels "0","1": 1 2 1 2 2 1 2 2 1 2 ...
##  $ HvyAlcoholConsump: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 2 1 1 1 ...
##  $ Education        : Factor w/ 6 levels "1","2","3","4",..: 6 6 6 6 5 4 5 4 6 4 ...
```

Data exploration

The diabetes_df dataset was explored to understand its structure and the distribution of the variables (features) prior to partitioning into two sets.

```
# The class of the dataset
class(diabetes_df)
```

```
## [1] "data.frame"
```

```
# The structure
str(diabetes_df, vec.len = 2)
```

```
## 'data.frame':    253680 obs. of  10 variables:
##  $ Diabetes_binary  : Factor w/ 2 levels "0","1": 1 1 1 1 1 ...
##  $ HighBP           : Factor w/ 2 levels "0","1": 2 1 2 2 2 ...
##  $ HighChol         : Factor w/ 2 levels "0","1": 2 1 2 1 2 ...
##  $ BMI              : num  40 25 28 27 24 ...
##  $ PhysActivity     : Factor w/ 2 levels "0","1": 1 2 1 2 2 ...
##  $ Sex              : Factor w/ 2 levels "0","1": 1 1 1 1 1 ...
##  $ Age              : Factor w/ 13 levels "1","2","3","4",..: 9 7 9 11 11 ...
##  $ Smoker           : Factor w/ 2 levels "0","1": 2 2 1 1 1 ...
##  $ HvyAlcoholConsump: Factor w/ 2 levels "0","1": 1 1 1 1 1 ...
##  $ Education        : Factor w/ 6 levels "1","2","3","4",..: 4 6 4 3 5 ...
```

```
# Examining the first few rows
head(diabetes_df)
```

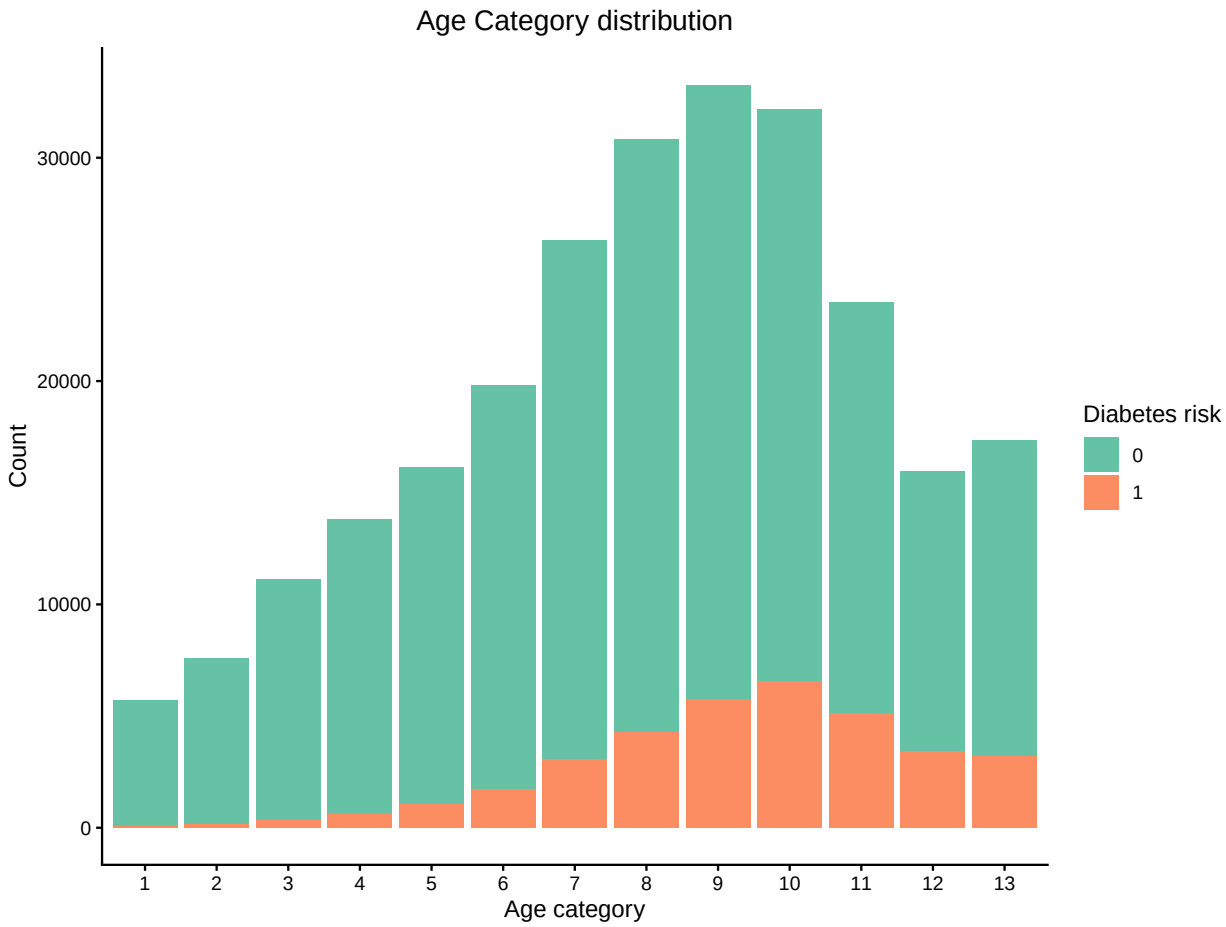
```
##   Diabetes_binary HighBP HighChol BMI PhysActivity Sex Age Smoker
## 1               0      1        1  40            0   0   9      1
## 2               0      0        0  25            1   0   7      1
## 3               0      1        1  28            0   0   9      0
## 4               0      1        0  27            1   0  11      0
## 5               0      1        1  24            1   0  11      0
## 6               0      1        1  25            1   1  10      1
##   HvyAlcoholConsump Education
## 1                 0         4
## 2                 0         6
## 3                 0         4
## 4                 0         3
## 5                 0         5
## 6                 0         6
```

```
# The summary of the edx set
summary(diabetes_df)
```

```
##  Diabetes_binary HighBP     HighChol        BMI        PhysActivity Sex
##  0:218334        0:144851   0:146089   Min.   :12.00   0: 61760     0:141974
##  1: 35346        1:108829   1:107591   1st Qu.:24.00   1:191920     1:111706
##                                        Median :27.00
##                                        Mean   :28.38
##                                        3rd Qu.:31.00
##                                        Max.   :98.00
##
##       Age        Smoker     HvyAlcoholConsump Education
##  9      :33244   0:141257   0:239424          1:   174
##  10     :32194   1:112423   1: 14256          2:  4043
##  8      :30832                               3:  9478
##  7      :26314                               4: 62750
##  11     :23533                               5: 69910
##  6      :19819                               6:107325
##  (Other):87744
```

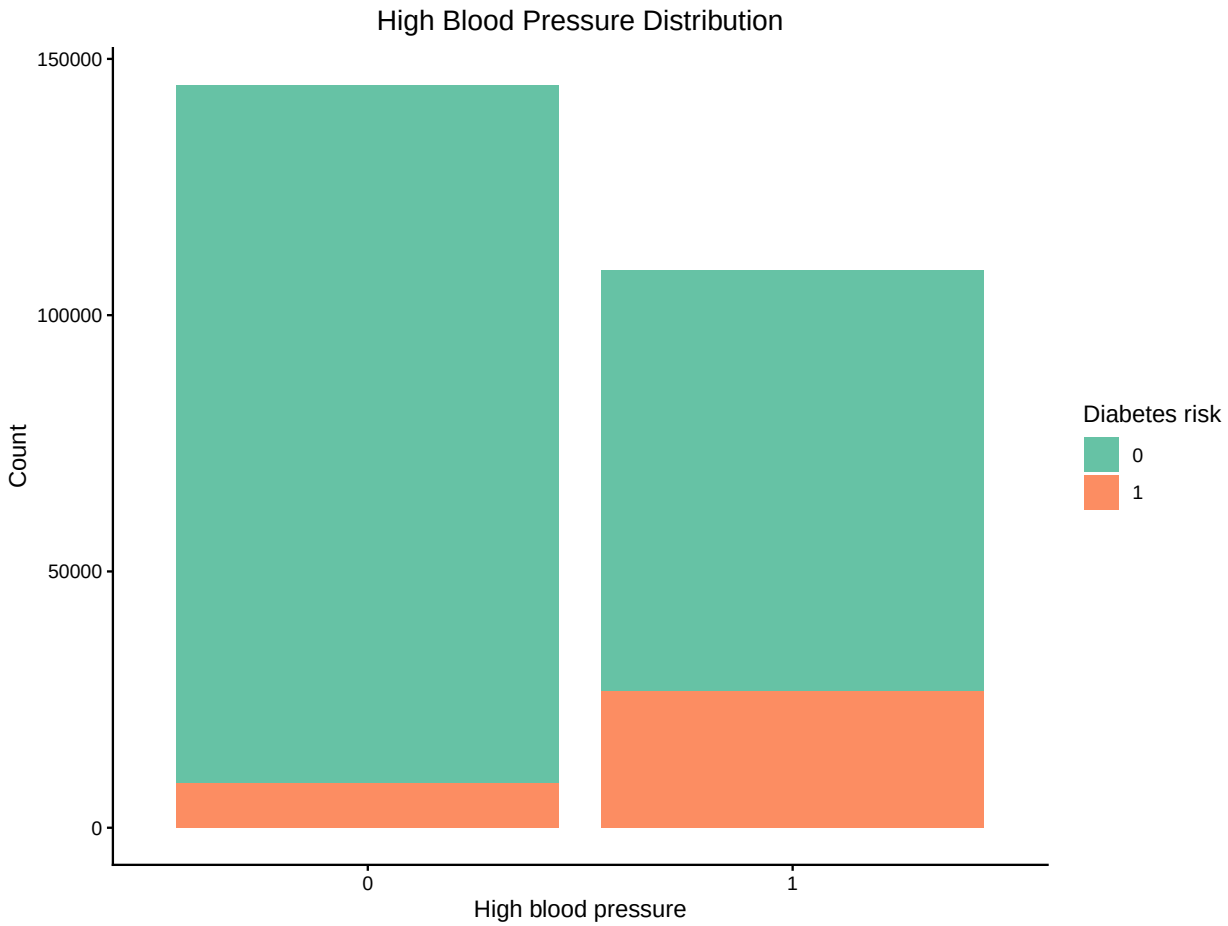
The distribution of the *Age* categories by the diabetes risk.

```
## Bar graph of Age
ggplot(diabetes_df, aes(x = Age, fill=Diabetes_binary)) +
  geom_bar() +
  scale_fill_brewer(palette = "Set2") +
  labs(y = "Count",
       x = "Age category",
       fill = "Diabetes risk",
       title = "Age Category distribution") +
  theme_classic()  +
  theme(plot.title = element_text(hjust = 0.5))
```



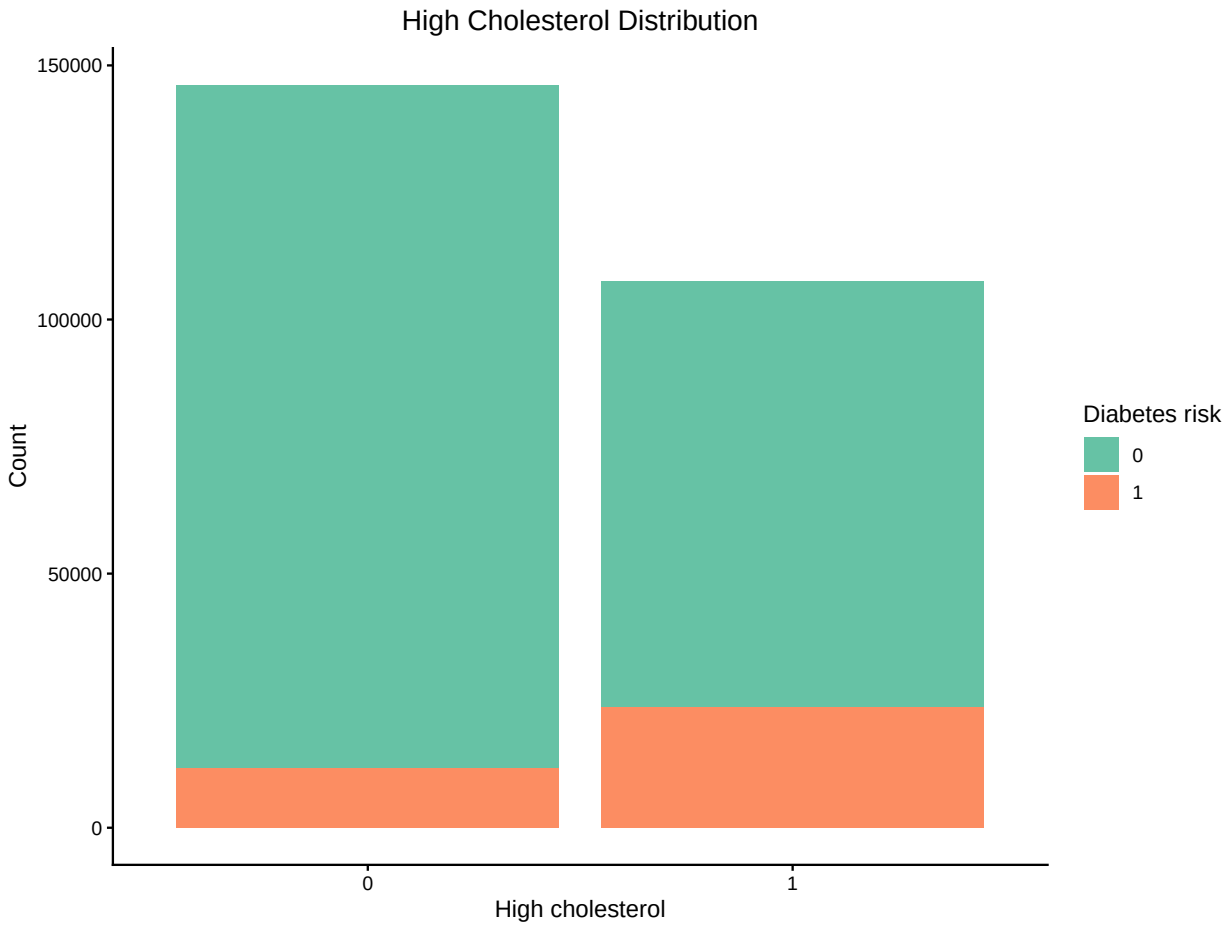
High blood pressure distribution according to diabetes risk

```
## Bar graph of High Blood Pressure
ggplot(diabetes_df) +
  geom_bar(aes(x = HighBP, fill = Diabetes_binary)) +
  scale_fill_brewer(palette = "Set2") +
  labs(y = "Count",
       x = "High blood pressure",
       fill = "Diabetes risk",
       title = "High Blood Pressure Distribution") +
  theme_classic()  +
  theme(plot.title = element_text(hjust = 0.5))
```



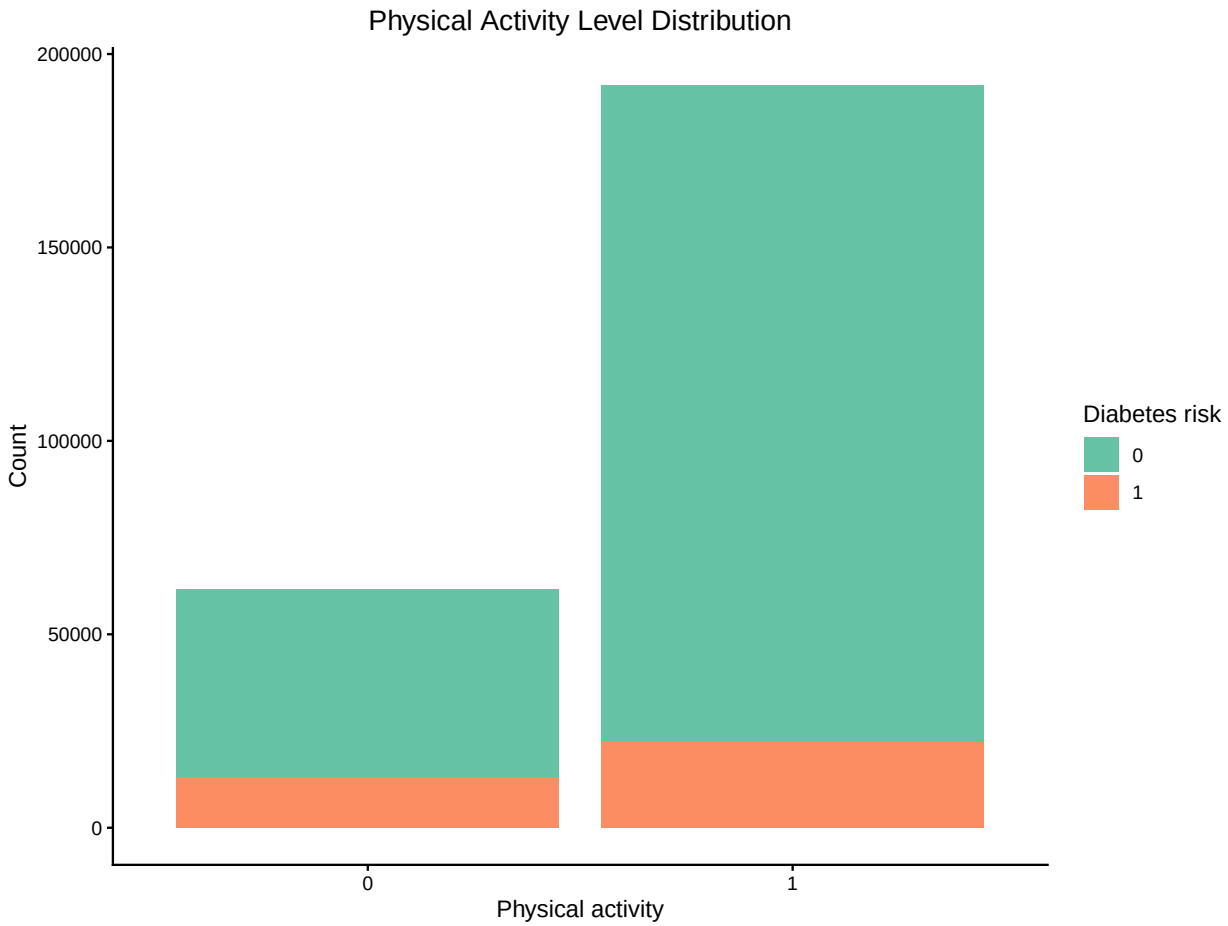
The distribution of *HighCholesterol* by diabetes risk is shown in a bar chart.

```
## Bar graph of High Cholesterol
ggplot(diabetes_df) +
  geom_bar(aes(x = HighChol, fill = Diabetes_binary)) +
  scale_fill_brewer(palette = "Set2") +
  labs(y = "Count",
       x = "High cholesterol",
       fill = "Diabetes risk",
       title = "High Cholesterol Distribution") +
  theme_classic()  +
  theme(plot.title = element_text(hjust = 0.5))
```



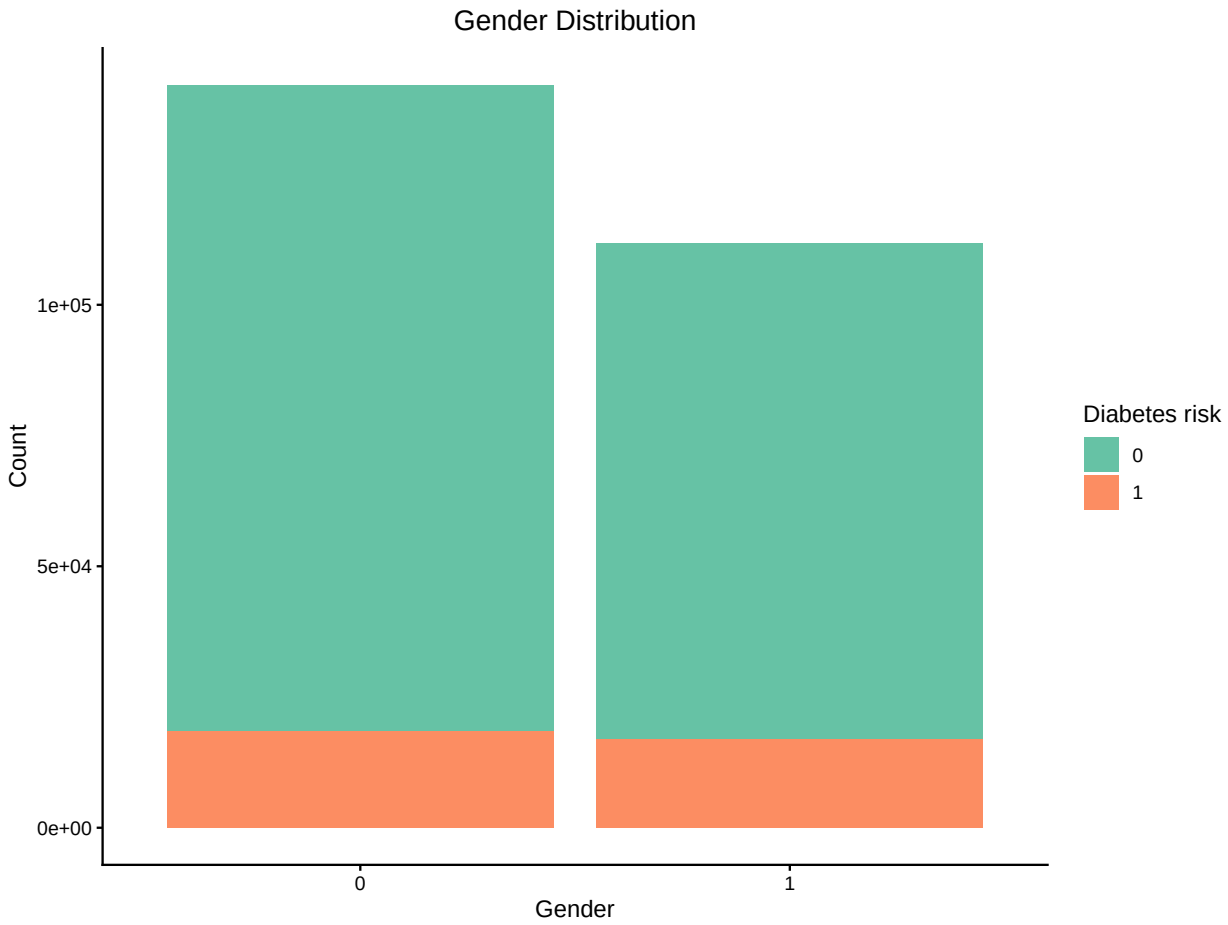
The distribution of *PhysicalActivity* level according the diabetes risk is shown below.

```
## Bar graph of Physical Activity
ggplot(diabetes_df) +
  geom_bar(aes(x = PhysActivity, fill = Diabetes_binary)) +
  scale_fill_brewer(palette = "Set2") +
  labs(y = "Count",
       x = "Physical activity",
       fill = "Diabetes risk",
       title = "Physical Activity Level Distribution") +
  theme_classic()  +
  theme(plot.title = element_text(hjust = 0.5))
```



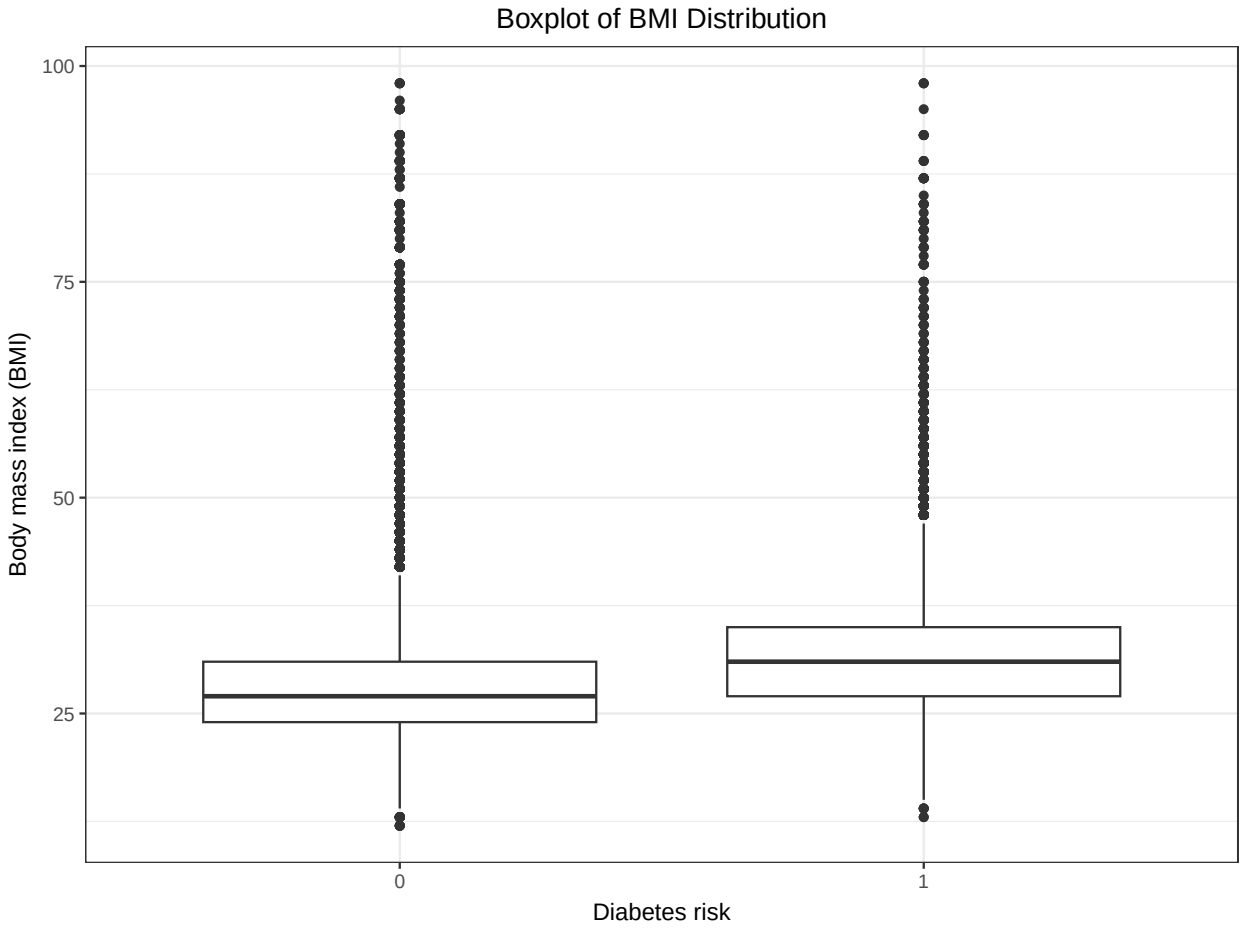
The *Sex* (gender) distribution of the diabetes risk is illustrated with a bar plot below.

```
## Bar graph of Sex
ggplot(diabetes_df) +
  geom_bar(aes(x = Sex, fill = Diabetes_binary)) +
  scale_fill_brewer(palette = "Set2") +
  labs(y = "Count",
       x = "Gender",
       fill = "Diabetes risk",
       title = "Gender Distribution") +
  theme_classic()  +
  theme(plot.title = element_text(hjust = 0.5))
```



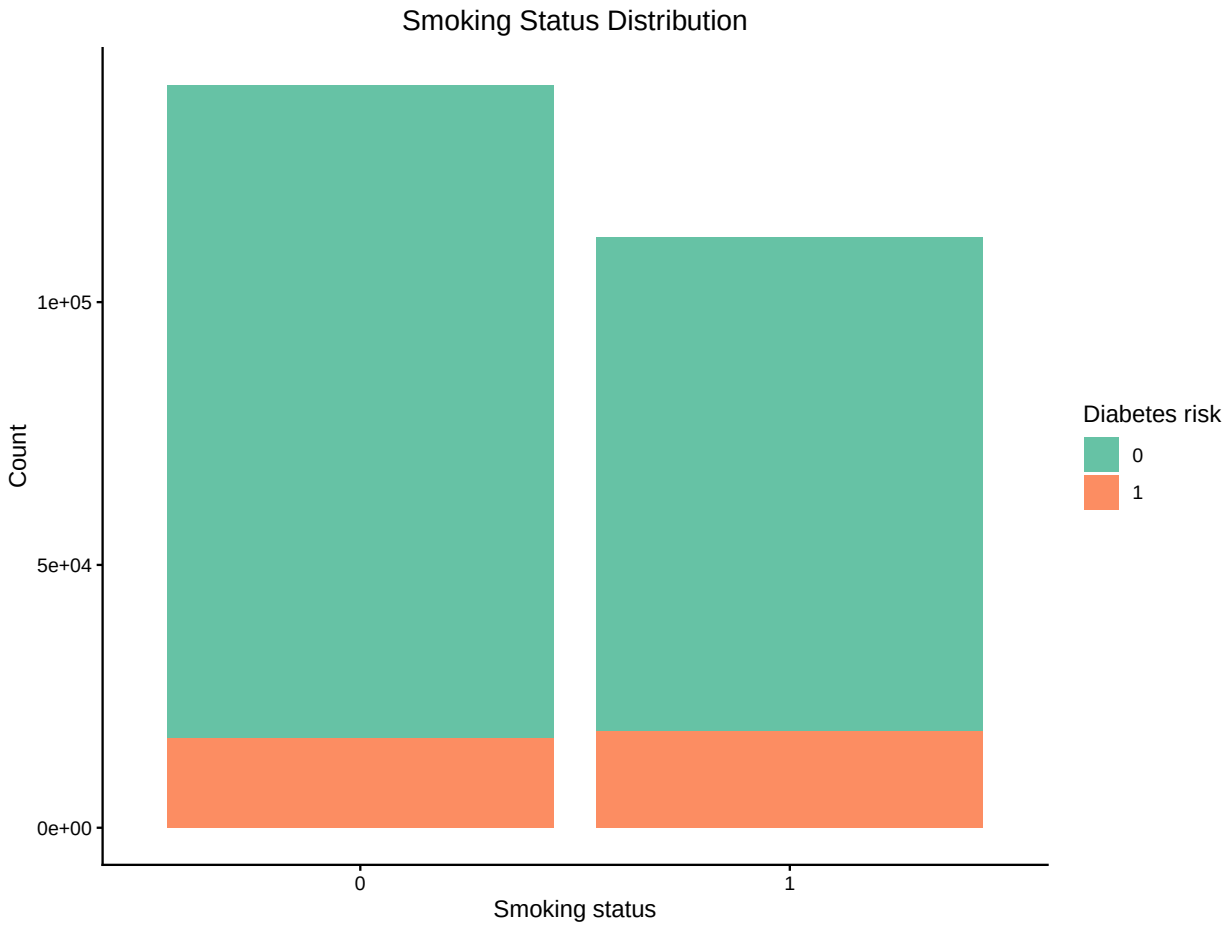
A boxplot of the BMI by diabetes risk is also plotted as shown below.

```
## Box plot for body mass index (BMI)
ggplot(diabetes_df) +
  geom_boxplot(aes(x = Diabetes_binary, y = BMI)) +
  ggtitle("Boxplot of BMI Distribution") +
  xlab("Diabetes risk") +
  ylab("Body mass index (BMI)") +
  theme_bw() +
  theme(axis.title.x = element_text(vjust = 0),
        axis.title.y = element_text(vjust = 2),
        plot.title = element_text(hjust=0.5))
```



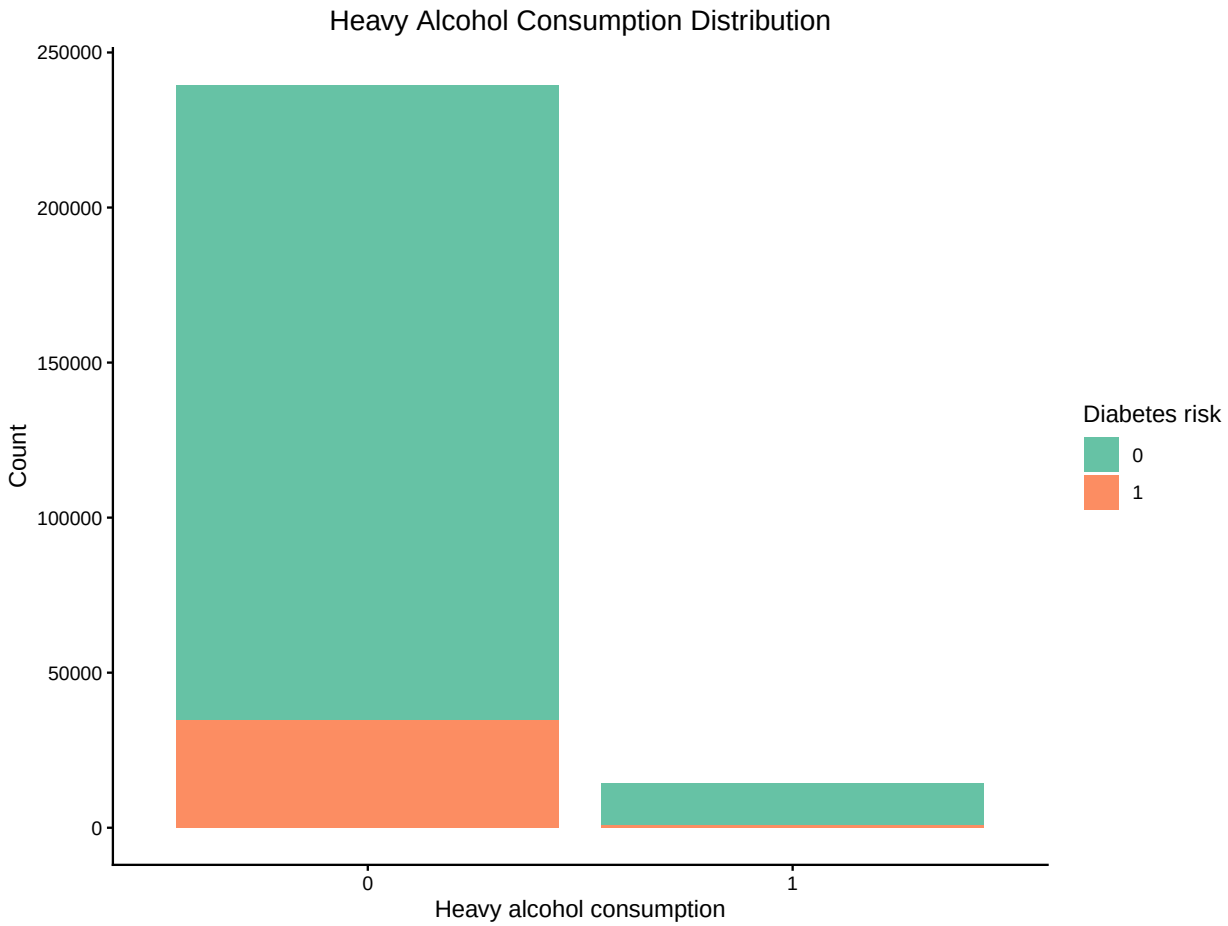
A bar chart visualisation of the distribution of *Smoker* (smoking status) by diabetes risk.

```
## Bar graph of Smoker
ggplot(diabetes_df) +
  geom_bar(aes(x = Smoker, fill = Diabetes_binary)) +
  scale_fill_brewer(palette = "Set2") +
  labs(y = "Count",
       x = "Smoking status",
       fill = "Diabetes risk",
       title = "Smoking Status Distribution") +
  theme_classic()  +
  theme(plot.title = element_text(hjust = 0.5))
```



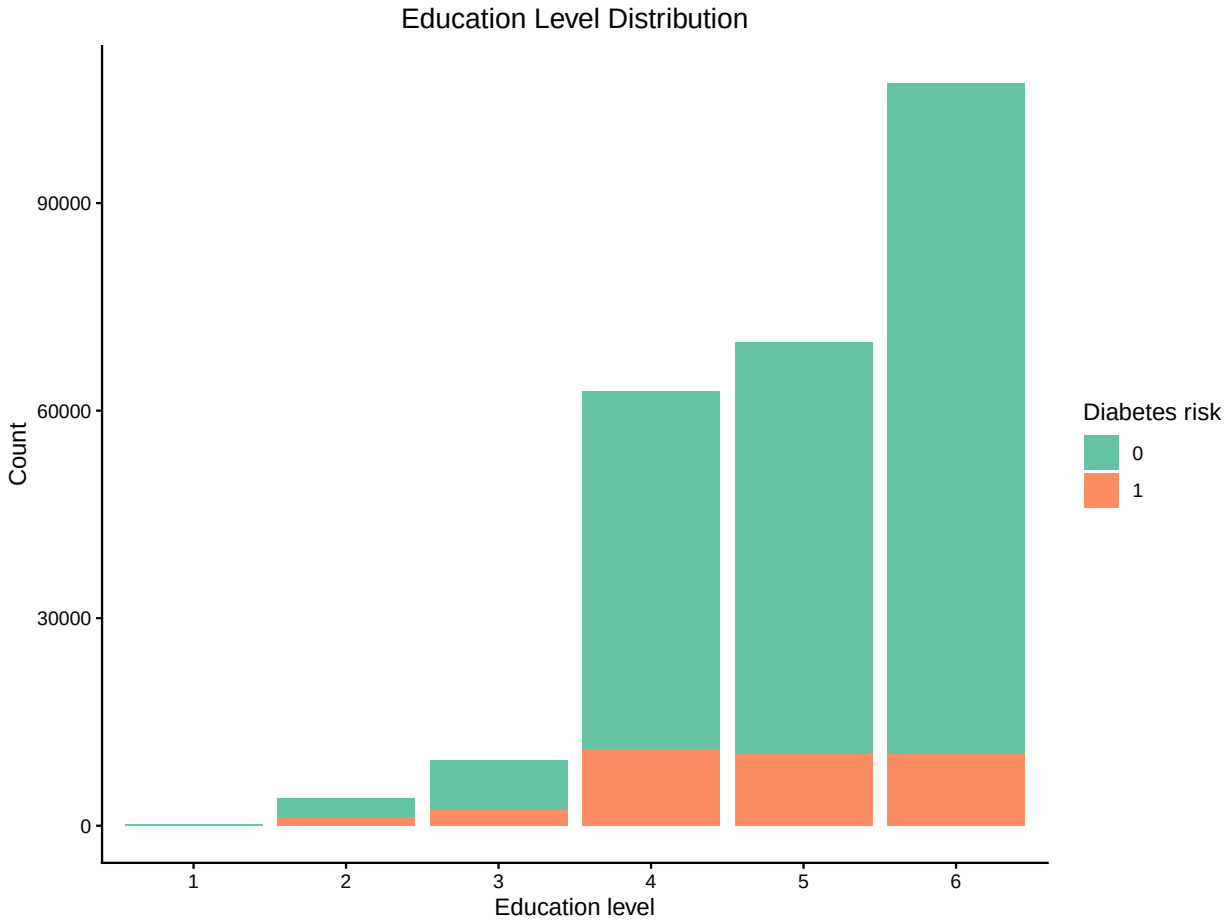
Also, a bar chart of $HeavyAlcoholConsumption$ distribution according to diabetes risk is illustrated below.

```
## Bar graph of Heavy Alcohol Consumption
ggplot(diabetes_df) +
  geom_bar(aes(x = HvyAlcoholConsump, fill = Diabetes_binary)) +
  scale_fill_brewer(palette = "Set2") +
  labs(y = "Count",
       x = "Heavy alcohol consumption",
       fill = "Diabetes risk",
       title = "Heavy Alcohol Consumption Distribution") +
  theme_classic()  +
  theme(plot.title = element_text(hjust = 0.5))
```



Finally, the distribution of *Education* level by the diabetes risk is also plotted in a bar graph.

```
## Bar graph of Education
ggplot(diabetes_df) +
  geom_bar(aes(x = Education, fill = Diabetes_binary)) +
  scale_fill_brewer(palette = "Set2") +
  labs(y = "Count",
       x = "Education level",
       fill = "Diabetes risk",
       title = "Education Level Distribution") +
  theme_classic()  +
  theme(plot.title = element_text(hjust = 0.5))
```



The Machine Learning Process

Dataset partitioning

The Diabetes_df dataset was partitioned into training (85%) and test (15%) sets, allowing the trained models to learn effectively and avoid overfitting. Also, the test set will be a representative sample for an unbiased evaluation of the models' generalisability. This balance will ensure that the built Machine Learning algorithms are robust and accurate[11]. The splitting process is shown in the code below.

```
# Set the seed to 1
set.seed(1)

partition_index <- createDataPartition(y = diabetes_df$Diabetes_binary,
                                       times = 1, p = 0.15, list = FALSE)
diabetes_train <- diabetes_df[-partition_index,]
diabetes_test <- diabetes_df[partition_index,]


## Check the balance of the partitioned sets
```

[11]Irizarry R. A: Introduction to Data Science. Statistics and Prediction Algorithms Through Case Studies. Accessed on 10 October 2025. url: https://rafalab.dfci.harvard.edu/dsbook-part-2/ml/ml-in-practice.html

```r
# Outcome variable - Diabetes_binary
diabetes_train %>% dplyr::count(Diabetes_binary)
```

```
##   Diabetes_binary      n
## 1               0 185583
## 2               1  30044
```

```r
diabetes_test %>% dplyr::count(Diabetes_binary)
```

```
##   Diabetes_binary     n
## 1               0 32751
## 2               1  5302
```

The Machine Learning Techniques

This project explores four Machine Learning techniques to build algorithms to classify diabetes based on carefully selected features (nine predictor variables). The Machine Learning methods included Logistic Regression, K-Nearest Neighbours, Decision Tree, and Random Forest[12]. For each approach, the performance of the trained algorithm was checked on the testing and validation datasets.

1 **Logistic regression**

This is a classic and effective technique for Machine Learning classification tasks; however, its use can be limited by potential overfitting and multicollinearity. Regularisation of logistic regression using LASSO (Least Absolute Shrinkage and Selection Operator) or Ridge can partly mitigate these limitations[13]. The LASSO and Ridge logistic regression have unique characteristics; therefore, both were applied to train a classification algorithm. The `glmnet` package was used together with the `tidymodels` package to prepare a hyperparameter tuning model, specifically optimising the penalty value.

a. **LASSO Logistic regression**

The LASSO hyperparameter tuning to controls how much shrinkage happens to the coefficients is illustrated by the equation:

$$\text{Loss} \;=\; \text{- LogLikelihood } (\beta) + \lambda \sum_{j=1}^{p} |\beta_j|$$

Where, $-LogLikelihood(\beta)$: The loss function for the logistic model, which depends on the model's coefficients, β.

λ(lambda): A hyperparameter that controls the strength of the regularization. A higher λ leads to stronger shrinkage.

$|\beta_j|$: The absolute value of the $j-th$ coefficient.

$\sum_{j=1}^{p} |\beta_j|$: The sum of the absolute values of all coefficients (excluding the intercept), known as the L1 norm.

The model building is shown in the codes below:

[12]Branch N. (n.d): Machine Learning Final Project: Predicting Diabetes. Available at https://rpubs.com/nbranch/preddiabetes

[13]Herawati N., Wijayanti A., Sutrisno A., & Misgiyati N. (2024). The Performance of Ridge Regression, LASSO, and Elastic-Net in Controlling Multicollinearity: A Simulation and Application. Journal of Modern Applied Statistical Methods, 23(2). https://doi.org/10.56801/Jmasm.V23.i2.4

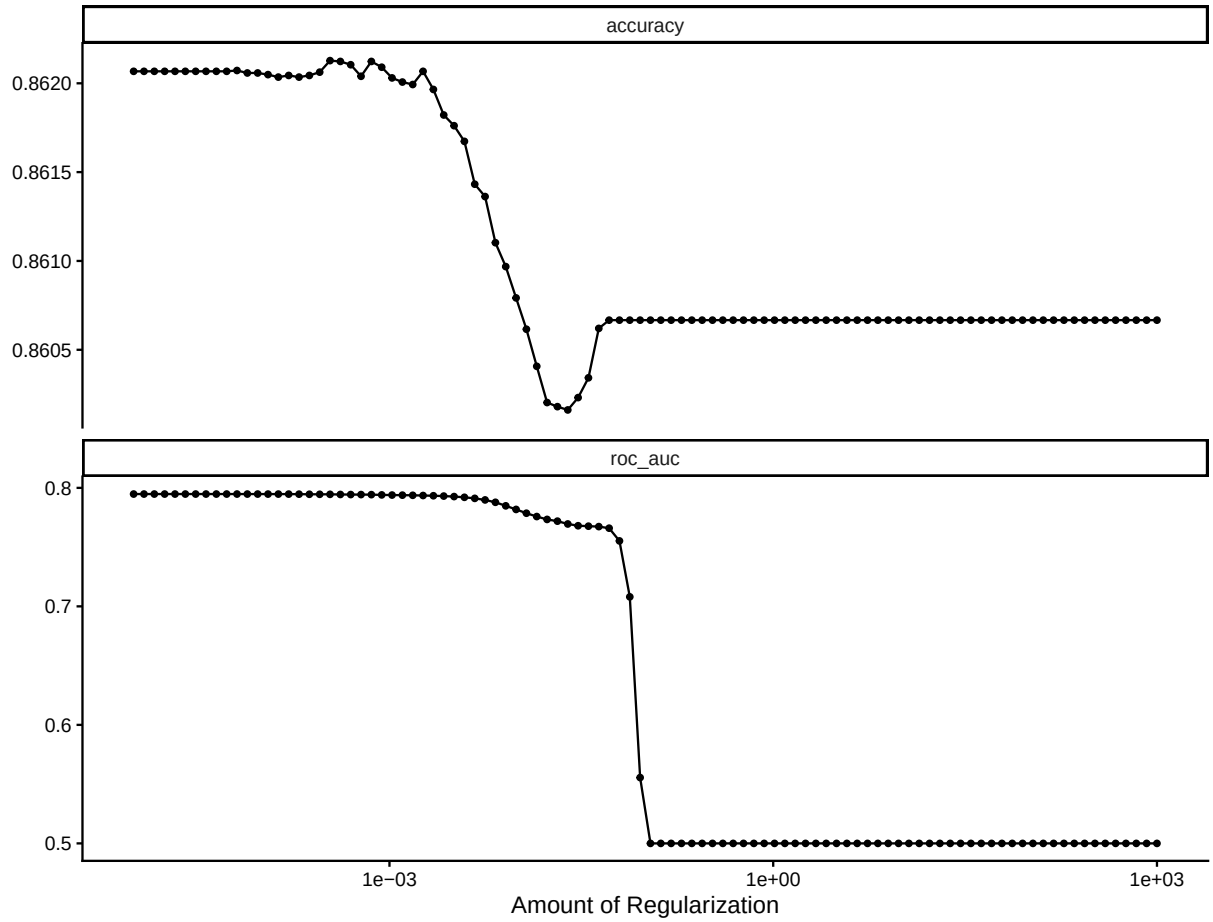
```r
#Set seed to ensure the results are reproducible.
set.seed(123)

# Set up cross-validation (CV) folds
lasso_cv <- vfold_cv(diabetes_train, v = 10) #small folds to minimise computation time

# model specification
lasso_spec <- logistic_reg() %>%
  set_engine("glmnet") %>%
  set_args(mixture = 1, penalty = tune()) %>% #pure LASSO model uses only the L1 penalty.
  set_mode("classification")

# Recipe
lasso_rec <- recipe(Diabetes_binary ~ .,
                    data = diabetes_train) %>%
  step_normalize(all_numeric_predictors()) %>%
  step_dummy(all_nominal_predictors())

# Workflow (Recipe + Model)
lasso_wf <- workflow() %>%
  add_model(lasso_spec) %>%
  add_recipe(lasso_rec)

# Tune model: specify grid of parameters and tune
lasso_penalty_grid <- grid_regular(penalty(range = c(-5,3)),
                           levels = 100
                           )

conflicted::conflicts_prefer(yardstick::accuracy)
```

```
## [conflicted] Will prefer yardstick::accuracy over any other package.
```

```r
lasso_tune_output <- tune_grid(lasso_wf,
                         resamples = lasso_cv,
                         metrics = metric_set(roc_auc, accuracy),
                         grid = lasso_penalty_grid
                         ) # This process will take some time to run
```

The best fitting model can be visualise as shown below

```r
### Selecting the best fitting LASSO model

autoplot(lasso_tune_output) + theme_classic()
```



The best penalty term is then selected

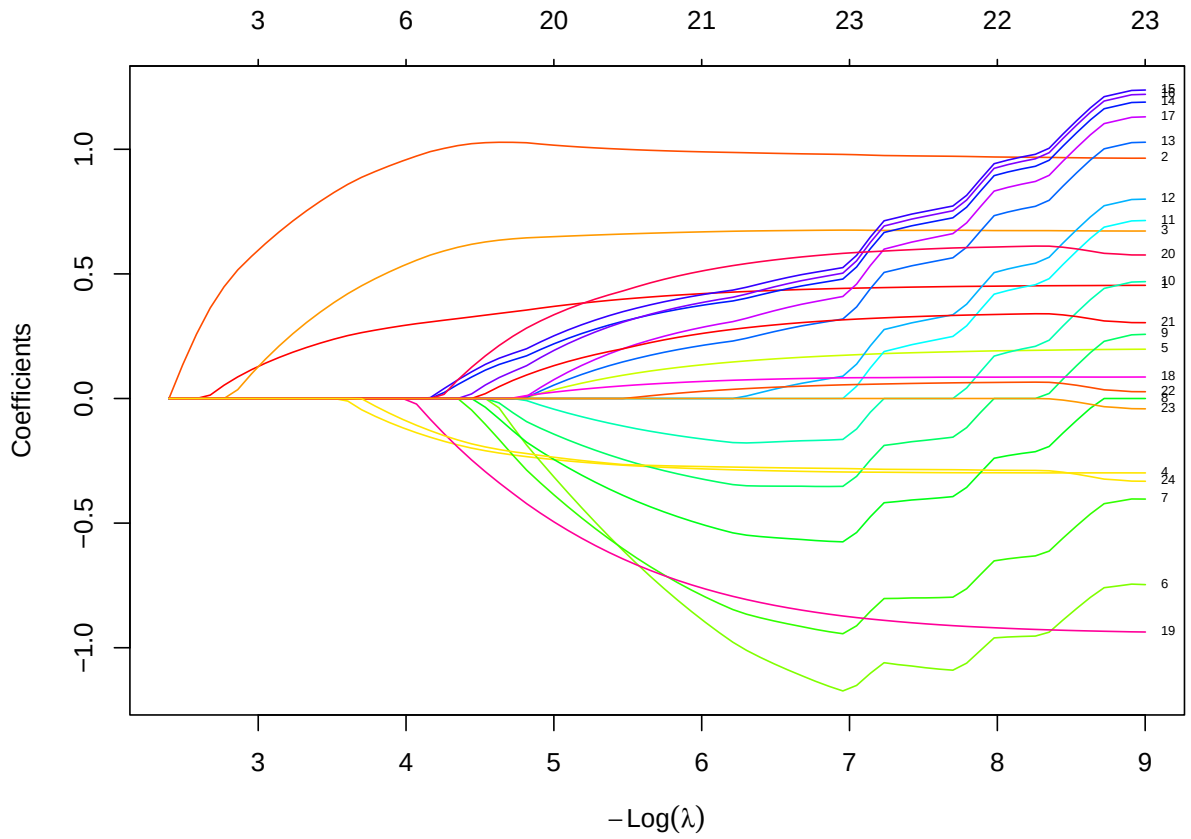
```
# Select "best" penalty by one standard error
lasso_best_penalty <- select_by_one_std_err(lasso_tune_output,
                                            metric = "roc_auc", desc(penalty)
                                            )

# Define workflow with "best" penalty value
lasso_fit_wf <- finalize_workflow(lasso_wf, lasso_best_penalty)

# Use final_wf to fit final model with "best" penalty value
lasso_fit_bestpenalty <- fit(lasso_fit_wf, data = diabetes_train)
```

Variable Importance

The LASSO variable importance in is determined by the magnitude of the non-zero absolute coefficients, where larger coefficients indicate the more important variables. The LASSO model performs automatic feature selection by shrinking the coefficients of less important variables to exactly zero. Variables with zero coefficients are considered insignificant and are effectively removed from the model, while those with non-zero coefficients are deemed important.

The code for determining the variable importance is given below.

```
# get the original glmnet output
glmnet_output <- lasso_fit_bestpenalty %>%
  extract_fit_parsnip() %>%
  pluck("fit")

plot(glmnet_output, xvar = "lambda",
     label = TRUE,
     col = rainbow(20)
     )
```




```
lasso_vip_p <- lasso_fit_bestpenalty %>%
  extract_fit_engine() %>%
  vip(num_features = 30) + theme_classic()
#lasso_vip_p
```

The numerical values of the variable importance were extracted.

```
# Extract the numerical values of the variable importance
lasso_var_imp <- as.data.frame(lasso_fit_bestpenalty %>%
  extract_fit_engine() %>%
  vip::vi())

#knitr::kable(lasso_var_imp)
```

The model evaluation accuracy and ROC_AUC metrics are indicated below.

```
 # Accuracy and ROC AUC measures
lasso_tune_output %>%
  collect_metrics() %>%
  filter(penalty == lasso_best_penalty %>%
           pull(penalty))     #get the mean with standard error
```

```
## # A tibble: 2 x 7
##   penalty .metric  .estimator  mean     n  std_err .config
##     <dbl> <chr>    <chr>      <dbl> <int>    <dbl> <chr>
## 1 0.00221 accuracy binary     0.862    10 0.000991 pre0_mod030_post0
## 2 0.00221 roc_auc  binary     0.793    10 0.00145  pre0_mod030_post0
```

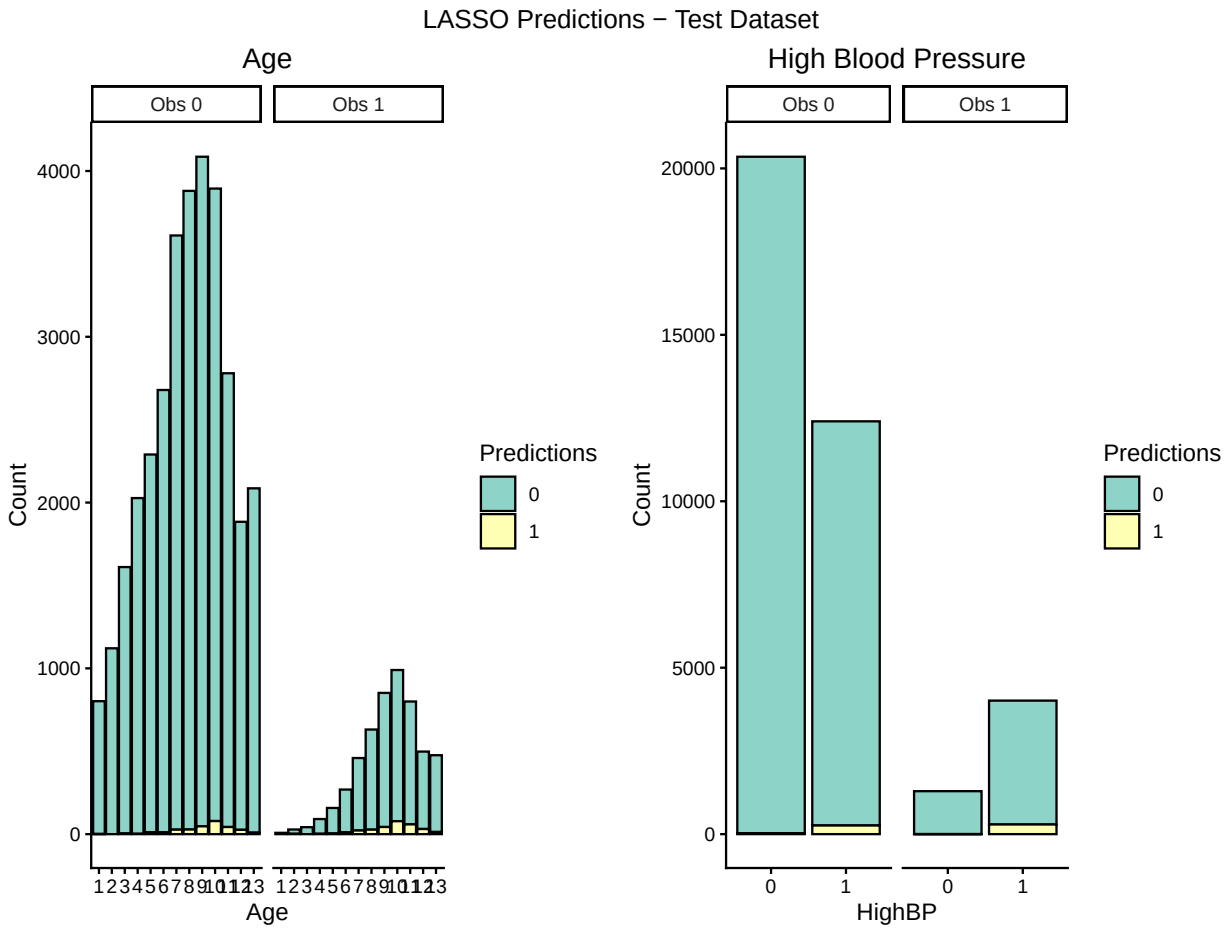
Model Performance The model prediction (classification) performance was evaluated using the testing and validation datasets.

i. *Testing dataset*

The predicted diabetes risk based on the trained LASSO algorithm using the testing dataset, and a visualisation plot using *HighBloodPressure* and *Age* features to illustrate the model performance, and the evaluation of the prediction are shown below.

```
### Predictions on the test dataset and plots
lasso_pred <- predict(lasso_fit_bestpenalty,
                      new_data = diabetes_test) %>%
  bind_cols(diabetes_test)

lasso_pred_labels <- c("0" = "Obs 0", "1"  = "Obs 1") #Obs - observed

# Plot for Age
ptest_lasso <- ggplot(lasso_pred,
                      aes(x = Age, fill = .pred_class)) +
  geom_bar(color = "black") +
  scale_fill_brewer(palette = "Set3") +
  facet_wrap(~Diabetes_binary,
             labeller = labeller(Diabetes_binary = lasso_pred_labels)) +
  labs(y = "Count",
       fill = "Predictions",
       title = "Age") +
  theme_classic() +
  theme(plot.title = element_text(hjust = 0.5))

# A plot for High blood pressure
ptest1_lasso <- ggplot(lasso_pred,
       aes(x = HighBP, fill = .pred_class)) +
  geom_bar(color = "black") +
  scale_fill_brewer(palette = "Set3") +
  facet_wrap(~Diabetes_binary,
             labeller = labeller(Diabetes_binary = lasso_pred_labels)) +
  labs(y = "Count",
       fill = "Predictions",
       title = "High Blood Pressure") +
```

```
  theme_classic() +
  theme(plot.title = element_text(hjust = 0.5))

# Combining the plots
gridExtra::grid.arrange(ptest_lasso, ptest1_lasso, ncol = 2,
                        top = "LASSO Predictions - Test Dataset")
```



Evaluation of the LASSO predictions on the test dataset

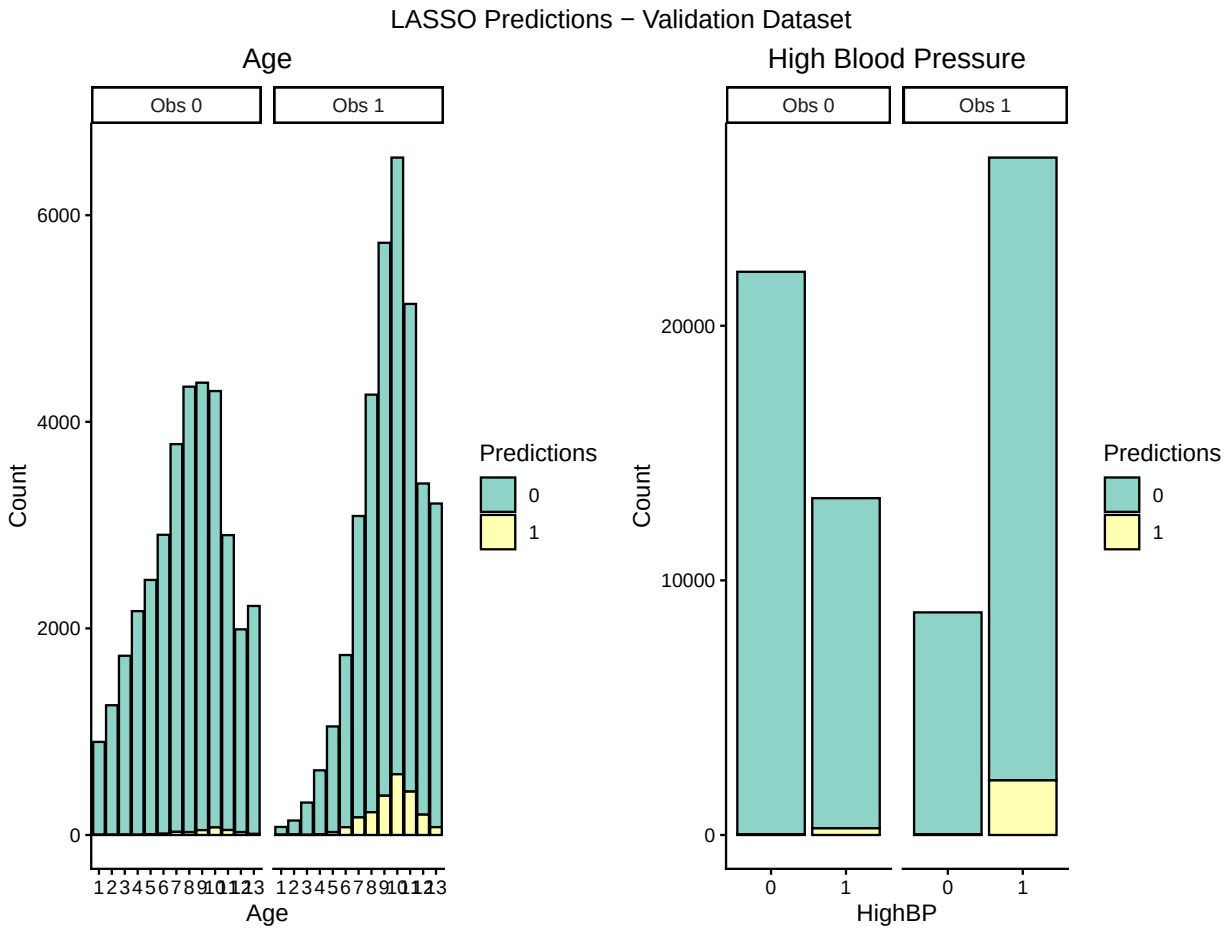
```
## Confusion matrix
confmat_lasso <- confusionMatrix(lasso_pred$.pred_class,
                                 diabetes_test$Diabetes_binary)

accuracy_lasso <- confmat_lasso$overall["Accuracy"]
sensitivity_lasso <- confmat_lasso$byClass["Sensitivity"]
specificity_lasso <- confmat_lasso$byClass["Specificity"]
det_rate_lasso <- confmat_lasso$byClass["Detection Rate"]



models_evaluations <- tibble(Method = "Logistic regression (LASSO): test",
                             Accuracy = accuracy_lasso,
                             Sensitivity = sensitivity_lasso,
```

```
                              Specificity = specificity_lasso,
                              Detection_Rate = det_rate_lasso)

models_evaluations
```

ii. *Validation dataset*

The predicted diabetes risk based on the trained LASSO algorithm, visualisation plots using $HighBloodPressure$ and Age features to illustrate the model performance, and the evaluation of the prediction on the validation dataset are shown below.

```
### Predictions on the validation dataset - 50:50 split of diabetes
lasso_pred_val <- predict(lasso_fit_bestpenalty, new_data = diabetes_val) %>%
  bind_cols(diabetes_val)

lasso_pred_val_labels <- c("0" = "Obs 0", "1"  = "Obs 1") #Obs - observed

# Plot for Age
pval_lasso <- ggplot(lasso_pred_val,
                     aes(x = Age, fill = .pred_class)) +
  geom_bar(color = "black") +
  scale_fill_brewer(palette = "Set3") +
  facet_wrap(~Diabetes_binary,
             labeller = labeller(Diabetes_binary = lasso_pred_val_labels)) +
  labs(y = "Count",
       fill = "Predictions",
       title = "Age") +
  theme_classic() +
  theme(plot.title = element_text(hjust = 0.5))

# A plot for High blood pressure
pval1_lasso <- ggplot(lasso_pred_val,
                      aes(x = HighBP, fill = .pred_class)) +
  geom_bar(color = "black") +
  scale_fill_brewer(palette = "Set3") +
  facet_wrap(~Diabetes_binary,
             labeller = labeller(Diabetes_binary = lasso_pred_val_labels)) +
  labs(y = "Count",
       fill = "Predictions",
       title = "High Blood Pressure") +
  theme_classic() +
  theme(plot.title = element_text(hjust = 0.5))

# Combining the plots
gridExtra::grid.arrange(pval_lasso, pval1_lasso, ncol = 2,
                        top = "LASSO Predictions - Validation Dataset")
```



Evaluation of the LASSO predictions on the validation dataset

```
## Confusion matrix
confmat_lasso_val <- confusionMatrix(lasso_pred_val$.pred_class,
                                     diabetes_val$Diabetes_binary)

accuracy_lasso_val <- confmat_lasso_val$overall["Accuracy"]
sensitivity_lasso_val <- confmat_lasso_val$byClass["Sensitivity"]
specificity_lasso_val <- confmat_lasso_val$byClass["Specificity"]
det_rate_lasso_val <- confmat_lasso_val$byClass["Detection Rate"]



models_evaluations <- bind_rows(models_evaluations,
                                tibble(Method = "Logistic regression (LASSO): validation",
                                       Accuracy = accuracy_lasso_val,
                                       Sensitivity = sensitivity_lasso_val,
                                       Specificity = specificity_lasso_val,
                                       Detection_Rate = det_rate_lasso_val)
                                )

knitr::kable(models_evaluations)
```

b. **Ridge Logistic regression**

Ridge logistic regression applies 'ridge' (L2) regularisation to add penalty terms to the loss function to prevent overfitting. The penalty term shrinks the coefficients toward zero, making the model less sensitive to individual data points and improving its ability to generalise to new data. The model minimises the sum of the negative log-likelihood and the ridge penalty:

$$\text{Minimise } (\text{ - LogLikelihood } + \lambda \sum_{j=1}^{K} \beta_j^2)$$

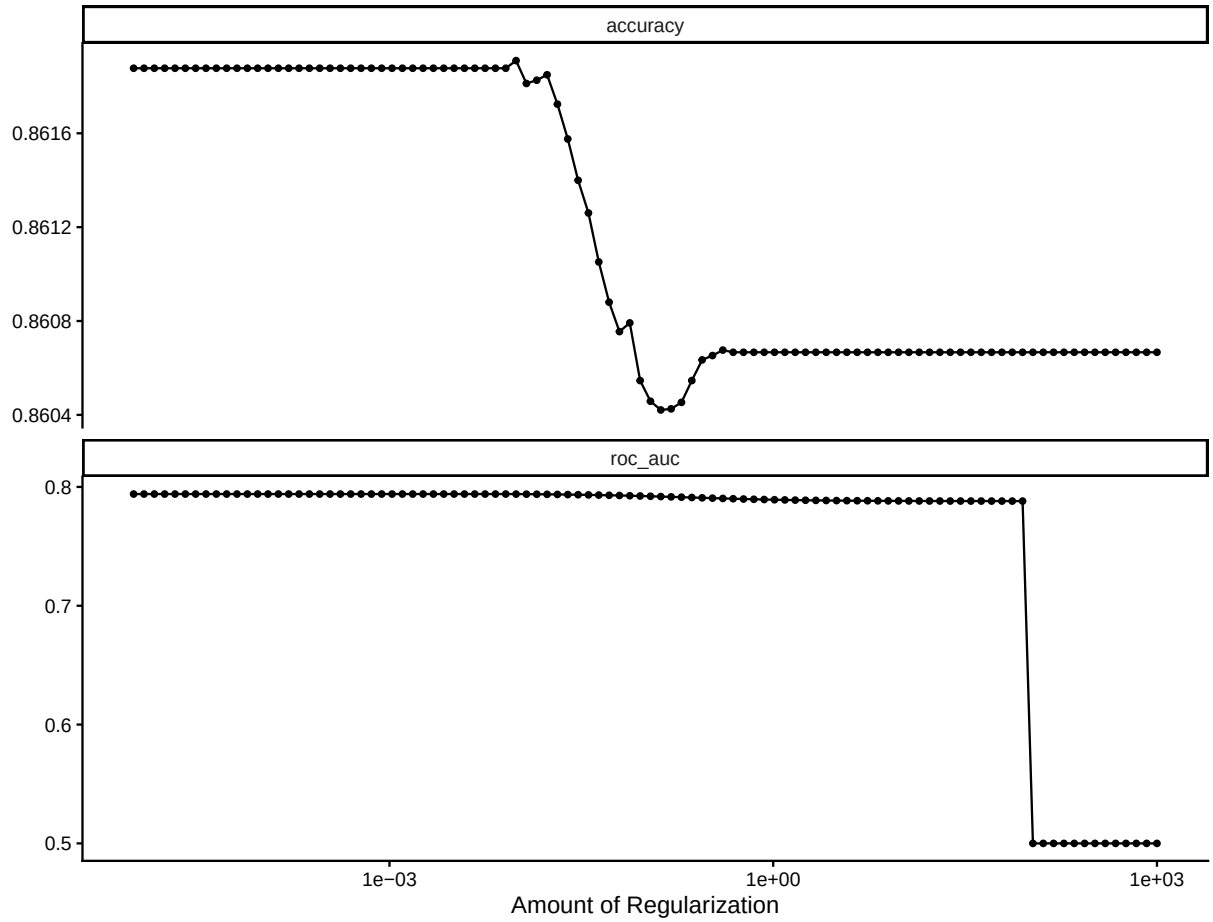
Where, λ is the regularization parameter, and β_j the regression coefficients, with the penalty term applied to all coefficients except the intercept.

The ridge model fitting is below

```
#Set seed
set.seed(123)

# Set up CV folds
ridge_cv <- vfold_cv(diabetes_train, v = 10)

# Ridge logistic regression model specification
ridge_spec <- logistic_reg() %>%
  set_engine("glmnet") %>%
  set_args(mixture = 0, penalty = tune()) %>% #the argument for Ridge L2 penalty,
  set_mode("classification")

# Recipe
ridge_rec <- recipe(Diabetes_binary ~ .,
                    data = diabetes_train) %>%
  step_normalize(all_numeric_predictors()) %>%
  step_dummy(all_nominal_predictors())

# Workflow (Recipe + Model)
ridge_wf <- workflow() %>%
  add_model(ridge_spec) %>%
  add_recipe(ridge_rec)

# Tune model: specify grid of parameters and tune
ridge_penalty_grid <- grid_regular(penalty(range = c(-5,3)),
                                   levels = 100
                                   )

ridge_tune_output <- tune_grid(ridge_wf,
                               resamples = ridge_cv,
                               metrics = metric_set(roc_auc, accuracy),
                               grid = ridge_penalty_grid
                               )
```

The best fitting Ridge model can be seen in this plot.

```
### Selecting the best fitting Ridge model

autoplot(ridge_tune_output) + theme_classic()
```



Best penalty term selection to fit the final model.

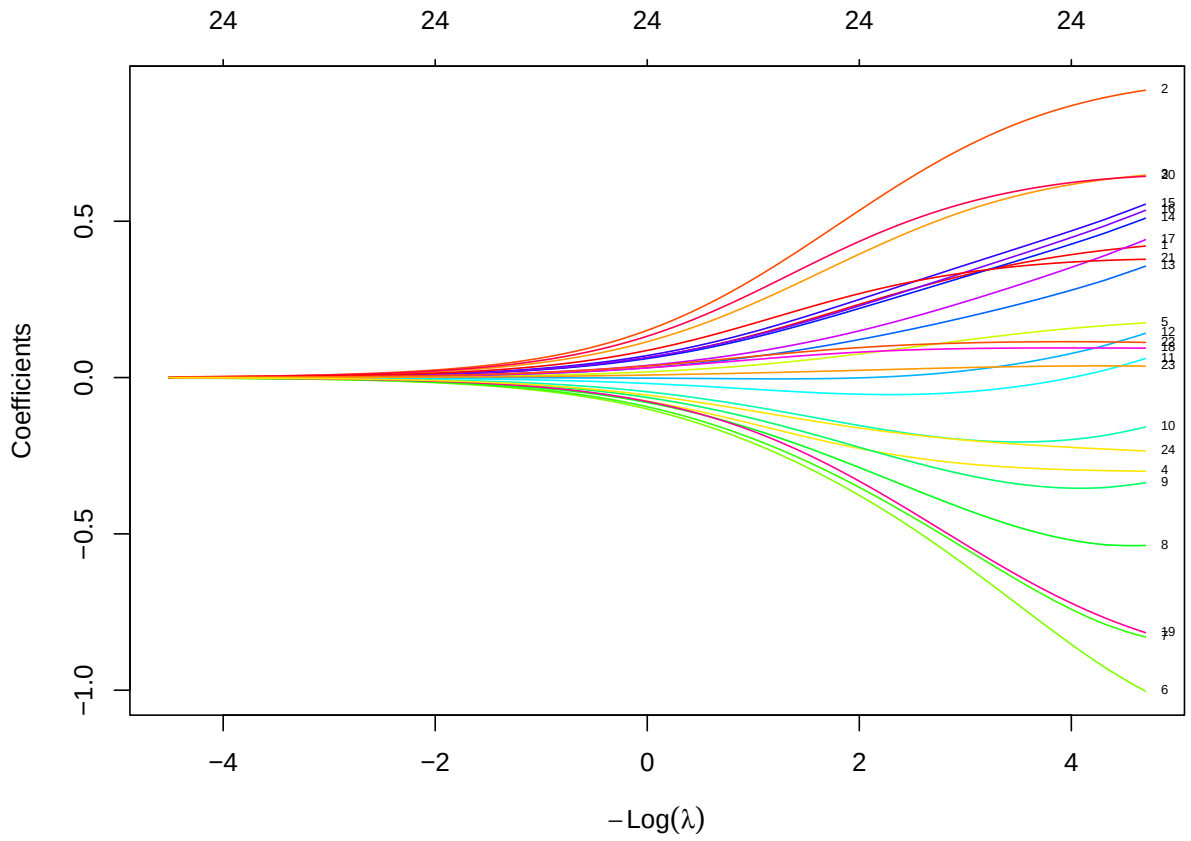
```
# Select "best" penalty by one standard error
ridge_best_penalty <- select_by_one_std_err(ridge_tune_output,
                                            metric = "roc_auc", desc(penalty)
                                            )

# Define workflow with "best" penalty value
ridge_fit_wf <- finalize_workflow(ridge_wf, ridge_best_penalty)

# Use final_wf to fit final model with "best" penalty value
ridge_fit_bestpenalty <- fit(ridge_fit_wf, data = diabetes_train)
```

Variable Importance

The magnitude of the coefficients primarily assesses variable importance in Ridge logistic regression; however, they must first be standardised as the Ridge penalty shrinks coefficients toward zero without setting them to exactly zero. Since Ridge regression does not perform automatic feature selection like Lasso, the absolute values of the standardised coefficients indicate importance. Features with larger absolute coefficient values are more influential.

```
# get the original glmnet output
glmnet_output_ridge <- ridge_fit_bestpenalty %>%
  extract_fit_parsnip() %>%
```

```
  pluck("fit")

plot(glmnet_output_ridge, xvar = "lambda",
     label = TRUE,
     col = rainbow(20)
     )
```




```
ridge_vip_p <- ridge_fit_bestpenalty %>%
  extract_fit_engine() %>%
  vip(num_features = 30) + theme_classic()
#ridge_vip_p
```

The numerical values of the variable importance in the model are shown below.

```
# Extract the numerical values of the variable importance
ridge_var_imp <- as.data.frame(ridge_fit_bestpenalty %>%
  extract_fit_engine() %>%
  vip::vi())

#knitr::kable(ridge_var_imp)
```

The trained Ridge model evaluation metrics are below.

```
 # Accuracy and ROC AUC measures

ridge_tune_output %>%
  collect_metrics() %>%
  filter(penalty == ridge_best_penalty %>%
           pull(penalty))     # get the mean with standard error
```

```
## # A tibble: 2 x 7
##   penalty .metric  .estimator  mean     n std_err .config
##     <dbl> <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1  0.0756 accuracy binary     0.861    10 0.00104 pre0_mod049_post0
## 2  0.0756 roc_auc  binary     0.793    10 0.00142 pre0_mod049_post0
```

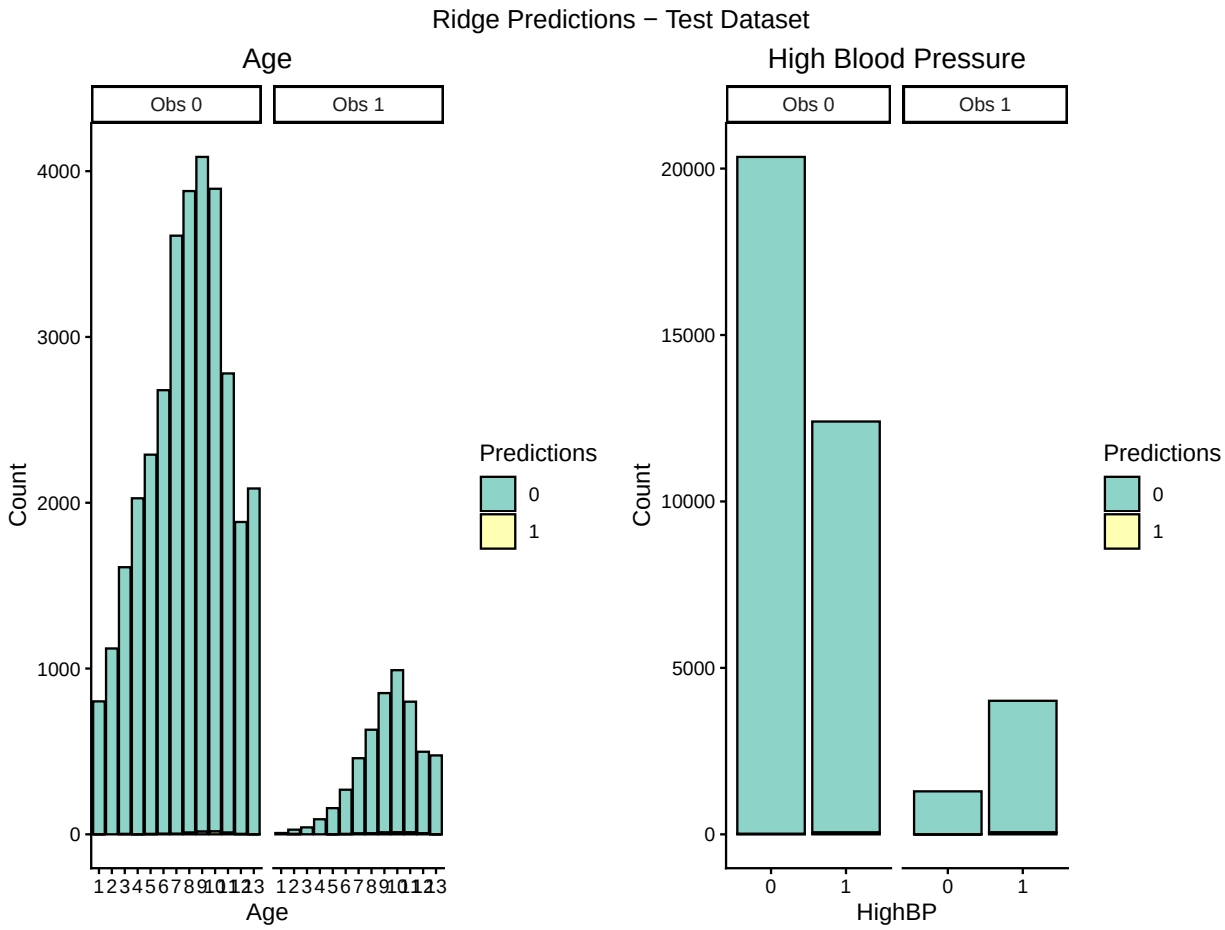
Model Performance The model prediction (classification) performance was evaluated using the testing and validation datasets.

i. *Testing dataset*

The trained Ridge algorithm was applied to the testing dataset to predict diabetes risk, and the performance was evaluated. The performance is illustrated in graphs using *HighBloodPressure* and *Age* variables in the model.

```
### Predictions on the test dataset and plots
ridge_pred <- predict(ridge_fit_bestpenalty,
                      new_data = diabetes_test) %>%
  bind_cols(diabetes_test)

ridge_pred_labels <- c("0" = "Obs 0", "1"  = "Obs 1") #Obs - observed

# Plot for Age
ptest_ridge <- ggplot(ridge_pred,
                      aes(x = Age, fill = .pred_class)) +
  geom_bar(color = "black") +
  scale_fill_brewer(palette = "Set3") +
  facet_wrap(~Diabetes_binary,
             labeller = labeller(Diabetes_binary = ridge_pred_labels)) +
  labs(y = "Count",
       fill = "Predictions",
       title = "Age") +
  theme_classic() +
  theme(plot.title = element_text(hjust = 0.5))

# A plot for High blood pressure
ptest1_ridge <- ggplot(ridge_pred,
                       aes(x = HighBP, fill = .pred_class)) +
  geom_bar(color = "black") +
  scale_fill_brewer(palette = "Set3") +
  facet_wrap(~Diabetes_binary,
             labeller = labeller(Diabetes_binary = ridge_pred_labels)) +
  labs(y = "Count",
       fill = "Predictions",
       title = "High Blood Pressure") +
```

```
  theme_classic() +
  theme(plot.title = element_text(hjust = 0.5))

# Combining the plots
gridExtra::grid.arrange(ptest_ridge, ptest1_ridge, ncol = 2,
                        top = "Ridge Predictions - Test Dataset")
```



Evaluation of the Ridge predictions on the test dataset

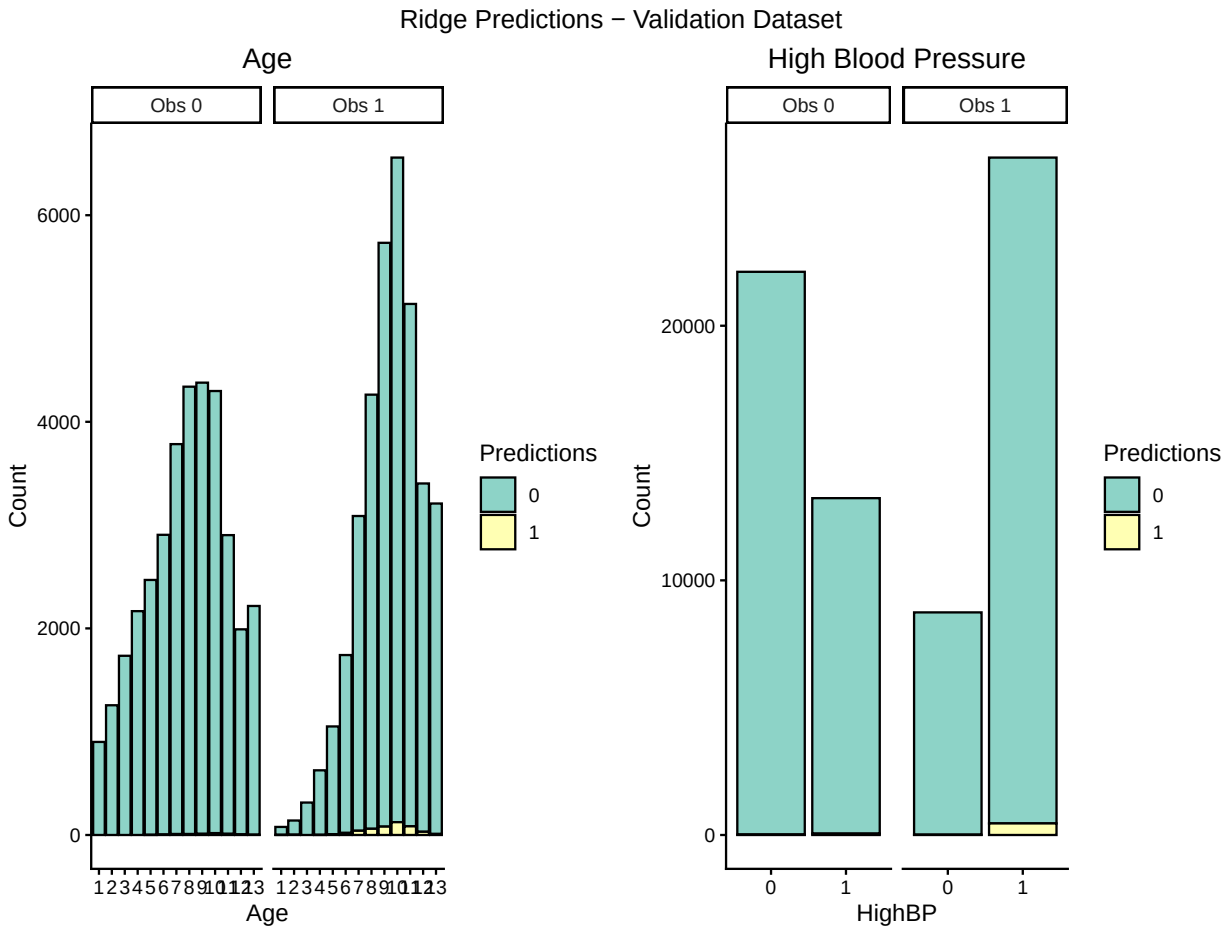
```
## Confusion matrix
confmat_ridge <- confusionMatrix(ridge_pred$.pred_class,
                                 diabetes_test$Diabetes_binary)

accuracy_ridge <- confmat_ridge$overall["Accuracy"]
sensitivity_ridge <- confmat_ridge$byClass["Sensitivity"]
specificity_ridge <- confmat_ridge$byClass["Specificity"]
det_rate_ridge <- confmat_ridge$byClass["Detection Rate"]



models_evaluations <- bind_rows(models_evaluations,
                                tibble(Method = "Logistic regression (Ridge): test",
                                       Accuracy = accuracy_lasso,
```

```
                                        Sensitivity = sensitivity_ridge,
                                        Specificity = specificity_ridge,
                                        Detection_Rate = det_rate_ridge))

knitr::kable(models_evaluations)
```

ii. *Validation dataset*

Similarly, the trained algorithm's performance on the validation dataset was checked.

```
### Predictions on the validation dataset - 50:50 split of diabetes
ridge_pred_val <- predict(ridge_fit_bestpenalty, new_data = diabetes_val) %>%
  bind_cols(diabetes_val)

ridge_pred_val_labels <- c("0" = "Obs 0", "1"  = "Obs 1") #Obs - observed

# Plot for Age
pval_ridge <- ggplot(ridge_pred_val,
                     aes(x = Age, fill = .pred_class)) +
  geom_bar(color = "black") +
  scale_fill_brewer(palette = "Set3") +
  facet_wrap(~Diabetes_binary,
             labeller = labeller(Diabetes_binary = ridge_pred_val_labels)) +
  labs(y = "Count",
       fill = "Predictions",
       title = "Age") +
  theme_classic() +
  theme(plot.title = element_text(hjust = 0.5))


# A plot for High blood pressure
pval1_ridge <- ggplot(ridge_pred_val,
                      aes(x = HighBP, fill = .pred_class)) +
  geom_bar(color = "black") +
  scale_fill_brewer(palette = "Set3") +
  facet_wrap(~Diabetes_binary,
             labeller = labeller(Diabetes_binary = ridge_pred_val_labels)) +
  labs(y = "Count",
       fill = "Predictions",
       title = "High Blood Pressure") +
  theme_classic() +
  theme(plot.title = element_text(hjust = 0.5))

# Combining the plots
gridExtra::grid.arrange(pval_ridge, pval1_ridge, ncol = 2,
                        top = "Ridge Predictions - Validation Dataset")
```



Evaluation of the Ridge predictions on the validation dataset

```
## Confusion matrix
confmat_ridge_val <- confusionMatrix(ridge_pred_val$.pred_class,
                                     diabetes_val$Diabetes_binary)

accuracy_ridge_val <- confmat_ridge_val$overall["Accuracy"]
sensitivity_ridge_val <- confmat_ridge_val$byClass["Sensitivity"]
specificity_ridge_val <- confmat_ridge_val$byClass["Specificity"]
det_rate_ridge_val <- confmat_ridge_val$byClass["Detection Rate"]



models_evaluations <- bind_rows(models_evaluations,
                                tibble(Method = "Logistic regression (Ridge): validation",
                                       Accuracy = accuracy_ridge_val,
                                       Sensitivity = sensitivity_ridge_val,
                                       Specificity = specificity_ridge_val,
                                       Detection_Rate = det_rate_ridge_val)
                                )

knitr::kable(models_evaluations)
```

2. **K-nearest neighbour (KNN)**

This is a relatively simple, non-parametric supervised Machine Learning algorithm that can be applied in classification and regression models. The principle that underpins K-Nearest Neighbour (KNN) is the classification of a new data point based on the majority class of its 'k' nearest neighbours in the feature space. Thus, the KNN classifier algorithm involves finding {k} predefined training samples that are closest to a new point in terms of distance and predicting a label for the new point using these samples[14]. A k-fold cross-validation in KNN is a technique used to evaluate the performance of a KNN model and to help select the optimal value for 'k' (the number of neighbours). KNN with k-fold cross-validation provides a more robust estimate of model performance than a single train-test split[15].

The `caret` package was used for the KNN algorithm with k-fold cross-validation. The algorithm training processes are shown in the following code.

```
# Set seed
set.seed(123)

## Fit KNN Model
knn_spec <- nearest_neighbor() %>%           # General model type
  set_args(neighbors = tune()) %>%           # Model tuning parameter
  set_engine(engine = "kknn") %>%            # Engine name
  set_mode("classification")

knn_cv <- vfold_cv(diabetes_train, v = 5) #Small folds to minimise the computation time

knn_rec <- recipe(Diabetes_binary ~ HighBP + BMI + HighChol + Age +
                    Sex + PhysActivity + HvyAlcoholConsump +
                    Smoker + Education, data = diabetes_train) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_normalize(all_numeric_predictors())

knn_wf <- workflow() %>%
  add_model(knn_spec) %>%         # Model specification object
  add_recipe(knn_rec)             # Data preprocessing recipe object

knn_param_tuning_grid <- grid_regular(neighbors(range = c(1, 100)),#Neighbors values range
                                levels = 5
                                )
```

Note: there is a conflict between the recipe and stringr packages that interferes with the code running. The stringr package is detached to allow the code to run smoothly and later uploaded.

```
### Conflictted: recipe and stringr packages, detach stringr and uploaded later

detach("package:stringr", unload = TRUE)
```

```
## Warning: 'stringr' namespace cannot be unloaded:
##   namespace 'stringr' is imported by 'ggthemes', 'kableExtra', 'reshape2' so cannot be unloaded
```

KNN model tuning is computationally intensive, and it took several hours to compute.

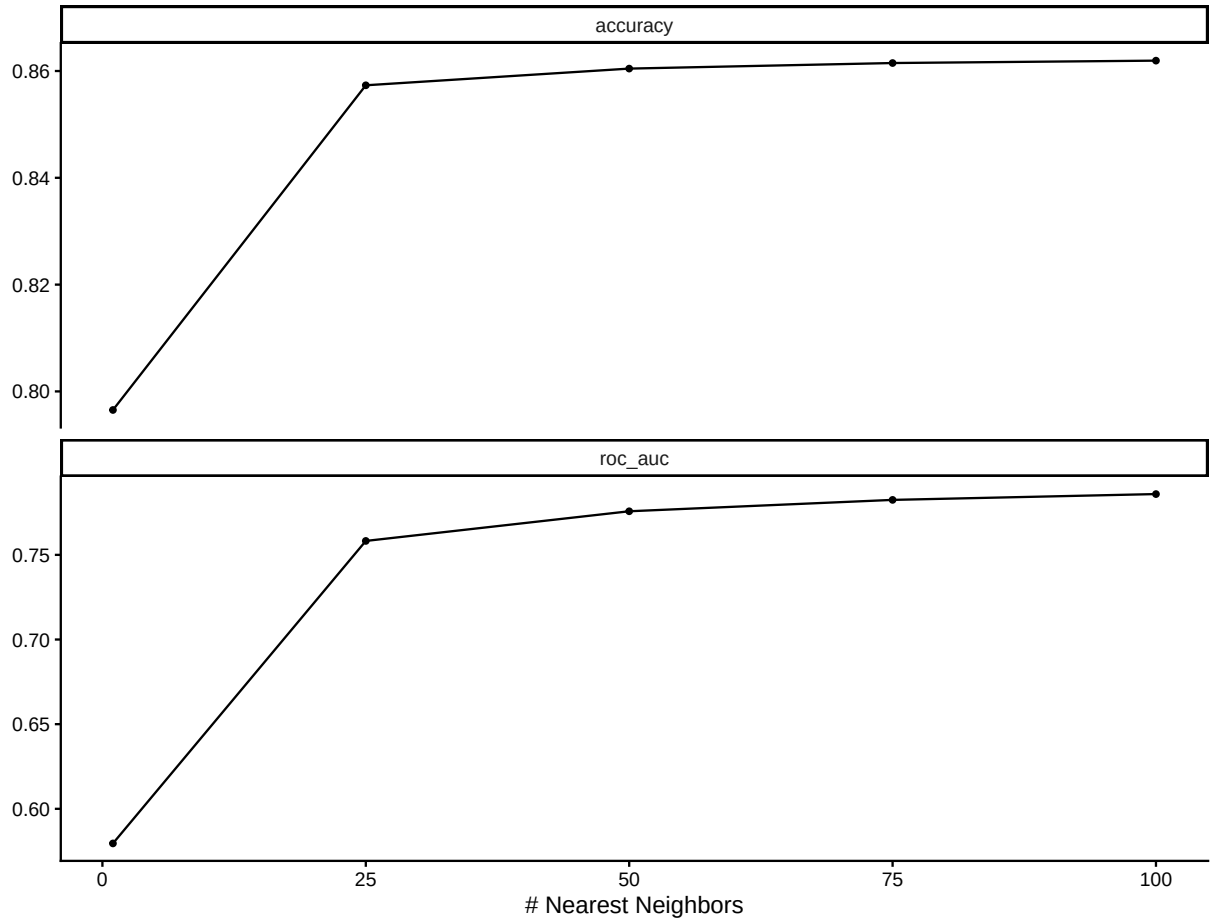
[14] Halder, R.K., Uddin, M.N., Uddin, M.A. et al. (2024): Enhancing K-nearest neighbor algorithm: a comprehensive review and performance analysis of modifications. J Big Data 11:113. https://doi.org/10.1186/s40537-024-00973-y

[15] Açıkkar, M. & Tokgöz, S. (2025): Improving multi-class classification: scaled extensions of harmonic mean-based adaptive k-nearest neighbors. Appl Intell 55:168. https://doi.org/10.1007/s10489-024-06109-2

```
knn_tune_output <- tune_grid(knn_wf,
                             resamples = knn_cv,
                             metrics = metric_set(roc_auc, accuracy),
                             grid = knn_param_tuning_grid
                             )  # NOTE: THIS STAGE WILL TAKE SEVERAL HOURS TO COMPLETE
```

The best tuning number of neighbours.

```
###Select best fitting number of neighbors
autoplot(knn_tune_output) + theme_classic()
```



The neighbour value with the highest number of neighbours was selected to fit the final best KNN model.

```
## Choose neighbors value that leads to the highest neighbors within 1 standard error.
knn_tune_output %>%
  select_by_one_std_err(metric = "roc_auc",
                        desc(neighbors)
                        ) #The desc(neighbors) sorts the data (most simple to most complex)
knn_tune_output %>%
  select_by_one_std_err(metric = "accuracy",
                        desc(neighbors)
                        )
best_neighbors <- select_by_one_std_err(knn_tune_output,
```

```
                                        metric = "roc_auc",
                                        desc(neighbors)
                                        )
fit_wf_knn <- finalize_workflow(knn_wf,
                                best_neighbors)
fit_bestneighbors <- fit(fit_wf_knn,
                         data = diabetes_train)
```

Variable importance

K-Nearest Neighbours (KNN) does not inherently provide a direct measure of variable importance as some other Machine Learning algorithms, such as decision trees or regression models. This is because KNN is a non-parametric, instance-based learning algorithm that relies on distances between data points in a multi-dimensional space, rather than learning explicit feature weights or coefficients.

The model fitting evaluation metrics were examined as follow:

```
# Show evaluation metrics for different values of neighbors, ordered
knn_tune_output %>% show_best(metric = "roc_auc")
```

```
## # A tibble: 5 x 7
##   neighbors .metric .estimator  mean     n std_err .config
##       <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1       100 roc_auc binary     0.786     5 0.00150 pre0_mod5_post0
## 2        75 roc_auc binary     0.782     5 0.00157 pre0_mod4_post0
## 3        50 roc_auc binary     0.776     5 0.00168 pre0_mod3_post0
## 4        25 roc_auc binary     0.758     5 0.00199 pre0_mod2_post0
## 5         1 roc_auc binary     0.580     5 0.00155 pre0_mod1_post0
```

```
knn_tune_output %>% show_best(metric = "accuracy")
```

```
## # A tibble: 5 x 7
##   neighbors .metric  .estimator  mean     n std_err .config
##       <int> <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1       100 accuracy binary     0.862     5 0.00164 pre0_mod5_post0
## 2        75 accuracy binary     0.861     5 0.00159 pre0_mod4_post0
## 3        50 accuracy binary     0.860     5 0.00142 pre0_mod3_post0
## 4        25 accuracy binary     0.857     5 0.00117 pre0_mod2_post0
## 5         1 accuracy binary     0.797     5 0.00118 pre0_mod1_post0
```

```
library("stringr")
```

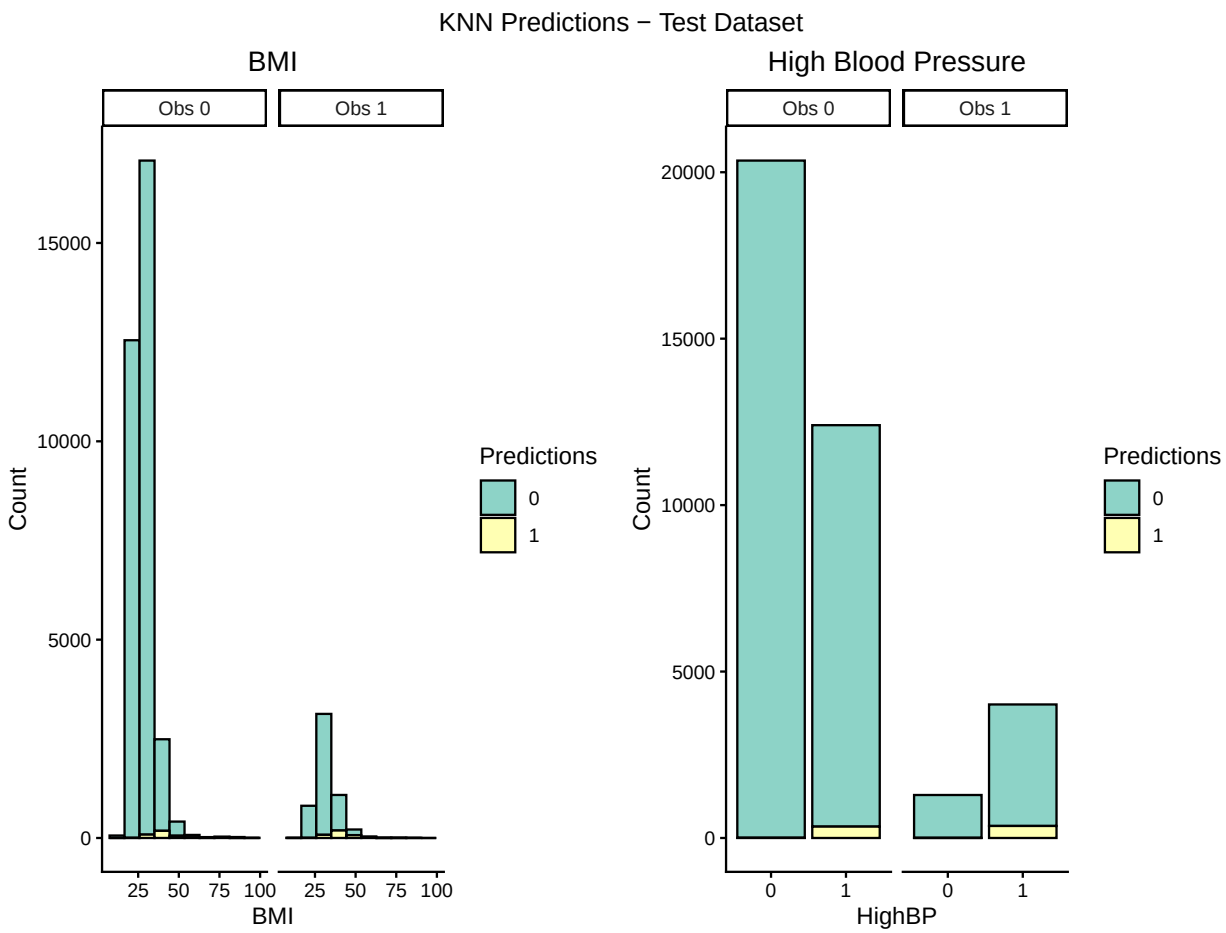
The KNN Model Performance

The built KNN algorithm performance was checked by predicting (classifying) diabetes risk using the testing and validation datasets.

i. *Testing dataset*

The diabetes risk classification and the performance evaluation, with visual illustrations using $HighBloodPressure$ and BMI variables, are demonstrated below.

```r
# Applying the best model to make predictions on testing dataset
knn_pred <- predict(fit_bestneighbors,
                    new_data = diabetes_test) %>%
  bind_cols(diabetes_test)

knn_pred_labels <- c("0" = "Obs 0", "1"  = "Obs 1") #Obs - observed

# A plot to illustrate the model performance using BMI as example
ptest_knn <- ggplot(knn_pred,
                    aes(x = BMI, fill = .pred_class)) +
  geom_histogram(bins = 10, color = "black") +
  scale_fill_brewer(palette = "Set3") +
  facet_wrap(~Diabetes_binary,
             labeller = labeller(Diabetes_binary = knn_pred_labels)) +
  labs(y = "Count",
       fill = "Predictions",
       title = "BMI") +
  theme_classic() +
  theme(plot.title = element_text(hjust = 0.5))

# A plot for High blood pressure
ptest1_knn <- ggplot(knn_pred,
                     aes(x = HighBP, fill = .pred_class)) +
  geom_bar(color = "black") +
  scale_fill_brewer(palette = "Set3") +
  facet_wrap(~Diabetes_binary,
             labeller = labeller(Diabetes_binary = knn_pred_labels)) +
  labs(y = "Count",
       fill = "Predictions",
       title = "High Blood Pressure") +
  theme_classic() +
  theme(plot.title = element_text(hjust = 0.5))

# Combining the plots
gridExtra::grid.arrange(ptest_knn, ptest1_knn, ncol = 2,
                        top = "KNN Predictions - Test Dataset")
```



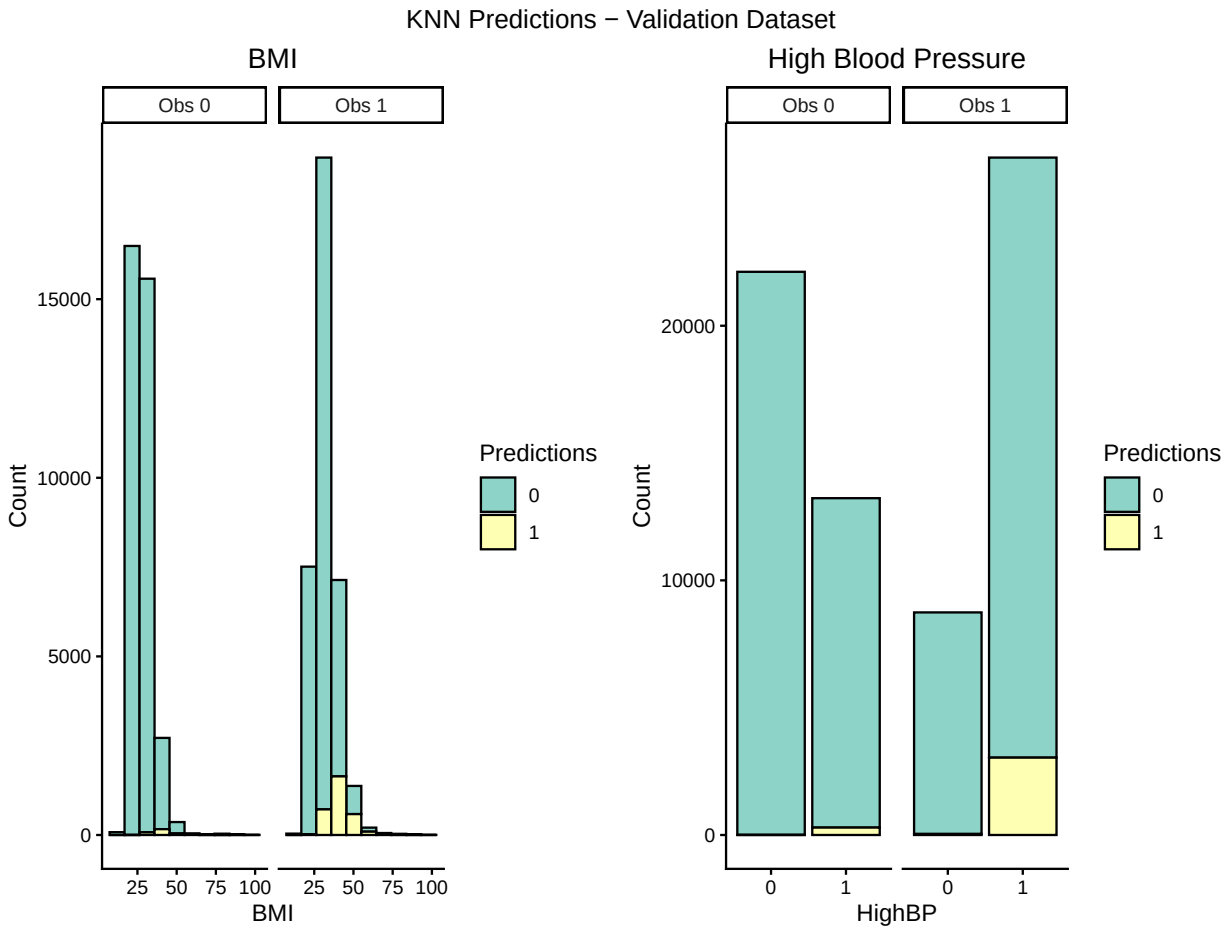
Evaluation of the KNN predictions on the test dataset

```
## Confusion matrix
confmat_knn <- confusionMatrix(knn_pred$.pred_class,
                               diabetes_test$Diabetes_binary)

accuracy_knn <- confmat_knn$overall["Accuracy"]
sensitivity_knn <- confmat_knn$byClass["Sensitivity"]
specificity_knn <- confmat_knn$byClass["Specificity"]
det_rate_knn <- confmat_knn$byClass["Detection Rate"]



models_evaluations <- bind_rows(models_evaluations,
                                tibble(Method = "K-Nearest Neighbors (KNN): test",
                                       Accuracy = accuracy_knn,
                                       Sensitivity = sensitivity_knn,
                                       Specificity = specificity_knn,
                                       Detection_Rate = det_rate_knn)
                                )

#knitr::kable(models_evaluations)
```

ii. *Validation dataset*

Likewise, the best-trained KNN algorithm was applied to the validation dataset to check the performance.

```
# Predictions on validation dataset: 50:50 split of diabetes

knn_pred_val <- predict(fit_bestneighbors,
                        new_data = diabetes_val) %>%
  bind_cols(diabetes_val)

knn_pred_val_labels <- c("0" = "Obs 0", "1"  = "Obs 1") #Obs - observed

# A plot to illustrate the model performance using BMI as example
pval_knn <- ggplot(knn_pred_val,
                   aes(x = BMI, fill = .pred_class)) +
  geom_histogram(bins = 10, color = "black") +
  scale_fill_brewer(palette = "Set3") +
  facet_wrap(~Diabetes_binary,
             labeller = labeller(Diabetes_binary = knn_pred_val_labels)) +
  labs(y = "Count",
       fill = "Predictions",
       title = "BMI") +
  theme_classic() +
  theme(plot.title = element_text(hjust = 0.5))


# A plot for High blood pressure
pval1_knn <- ggplot(knn_pred_val,
                    aes(x = HighBP, fill = .pred_class)) +
  geom_bar(color = "black") +
  scale_fill_brewer(palette = "Set3") +
  facet_wrap(~Diabetes_binary,
             labeller = labeller(Diabetes_binary = knn_pred_val_labels)) +
  labs(y = "Count",
       fill = "Predictions",
       title = "High Blood Pressure") +
  theme_classic() +
  theme(plot.title = element_text(hjust = 0.5))

# Combining the plots
gridExtra::grid.arrange(pval_knn, pval1_knn, ncol = 2,
                        top = "KNN Predictions - Validation Dataset")
```



Evaluation of the KNN predictions on the validation dataset

```
## Confusion matrix
confmat_knn_val <- confusionMatrix(knn_pred_val$.pred_class,
                                   diabetes_val$Diabetes_binary)

accuracy_knn_val <- confmat_knn_val$overall["Accuracy"]
sensitivity_knn_val <- confmat_knn_val$byClass["Sensitivity"]
specificity_knn_val <- confmat_knn_val$byClass["Specificity"]
det_rate_knn_val <- confmat_knn_val$byClass["Detection Rate"]


models_evaluations <- bind_rows(models_evaluations,
                                tibble(Method = "K-Nearest Neighbors (KNN): validation",
                                       Accuracy = accuracy_knn_val,
                                       Sensitivity = sensitivity_knn_val,
                                       Specificity = specificity_knn_val,
                                       Detection_Rate = det_rate_knn_val)
                                )

knitr::kable(models_evaluations)
```

3. **Decision Tree**

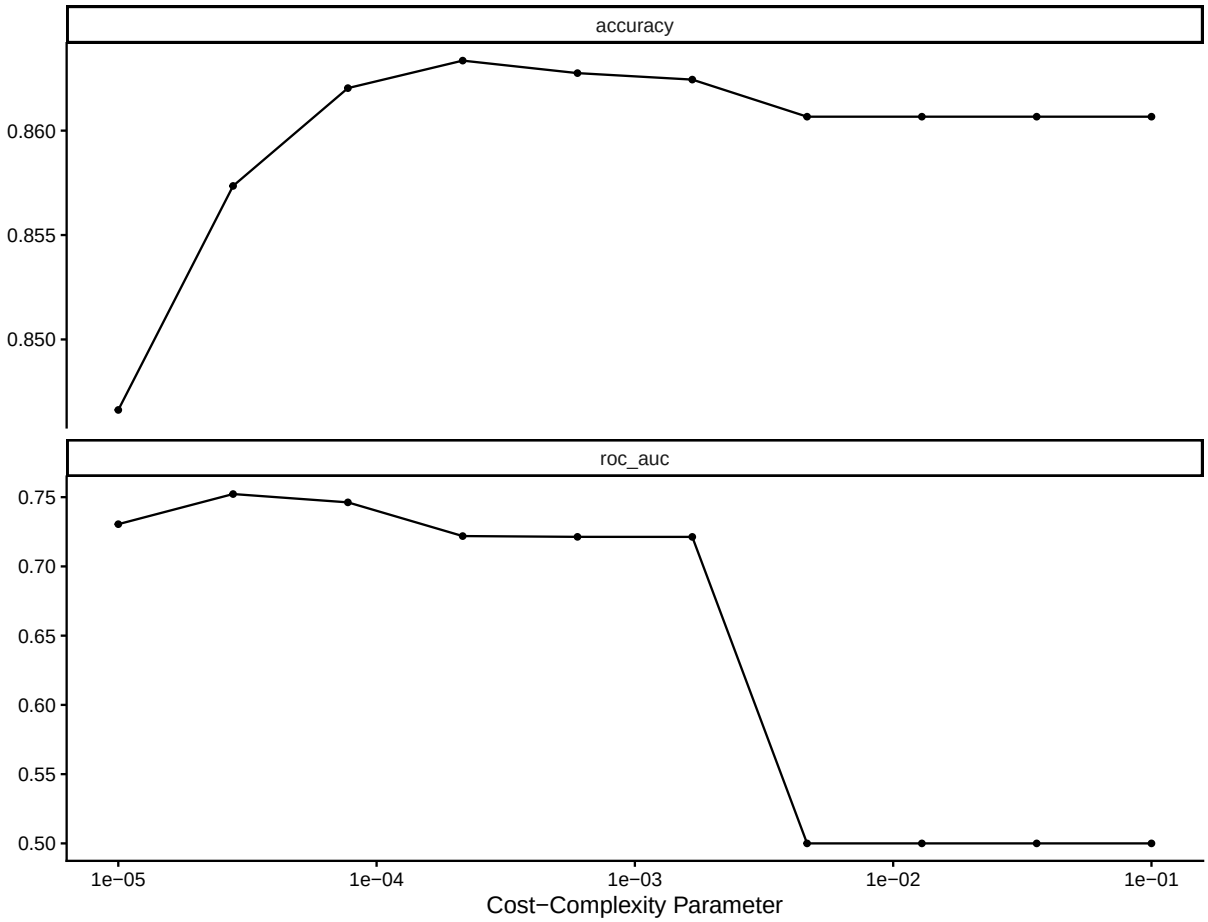
Decision tree classification is a non-parametric method that recursively partitions data into more "pure" nodes, based on splitting rules. This involves building a model that predicts a categorical outcome variable based on a set of predictor variables. Cross-validation is often used in decision trees to determine the optimal tree complexity and the best complexity parameter (cp) for pruning. This approach potentially prevents overfitting and improves the generalisation of the trained algorithm to external datasets[16]. The `rpart` and `caret` packages were utilised to build the tree and cross-validation, respectively. The codes henceforth show the decision tree training and the predictions on the testing and validation datasets.

```
# Set seed
set.seed(123)

## Fit Decision Tree
dtree_spec <- decision_tree() %>%
  set_engine(engine = "rpart") %>%
  set_args(cost_complexity = tune(),
           min_n = 2,
           tree_depth = NULL) %>%
  set_mode("classification")

dtree_cv <- vfold_cv(diabetes_train, v = 6) #Small folds to minimise computation time

dtree_rec <- recipe(Diabetes_binary ~ .,
                    data = diabetes_train)

#the workflow
dtree_wf <- workflow() %>%
  add_model(dtree_spec) %>%
  add_recipe(dtree_rec)

#tuning parameter grid
dtree_param_grid <- grid_regular(cost_complexity(range = c(-5, -1)),
                            levels = 10)
#Model tuning
dtree_tune_res <- tune_grid(dtree_wf,
                        resamples = dtree_cv,
                        grid = dtree_param_grid,
                        metrics = metric_set(accuracy, roc_auc)
                        )
```

Visualise and select the best tuning parameter for the tree.

```
## Select and Fit Best Tree
autoplot(dtree_tune_res) + theme_classic()
```

[16]Rokach, L. & Maimon, O. (2005): Decision trees. Data Mining and Knowledge Discovery Handbook, Springer, 165-192



Cost-complexity selection to balance model performance and complexity to prevent overfitting of the decision tree algorithm.

```
#the cost-complexity selection
best_complexity <- select_by_one_std_err(dtree_tune_res,
                                         metric = 'roc_auc',
                                         desc(cost_complexity))

#the final workflow
dtree_fit_wf <- finalize_workflow(dtree_wf, best_complexity)

#the final best fit decision tree model
fit_dtree_best <- fit(dtree_fit_wf, data = diabetes_train)
```

```
## Visualise the Tree
dtree_plpot <- fit_dtree_best %>%
  extract_fit_engine()
```

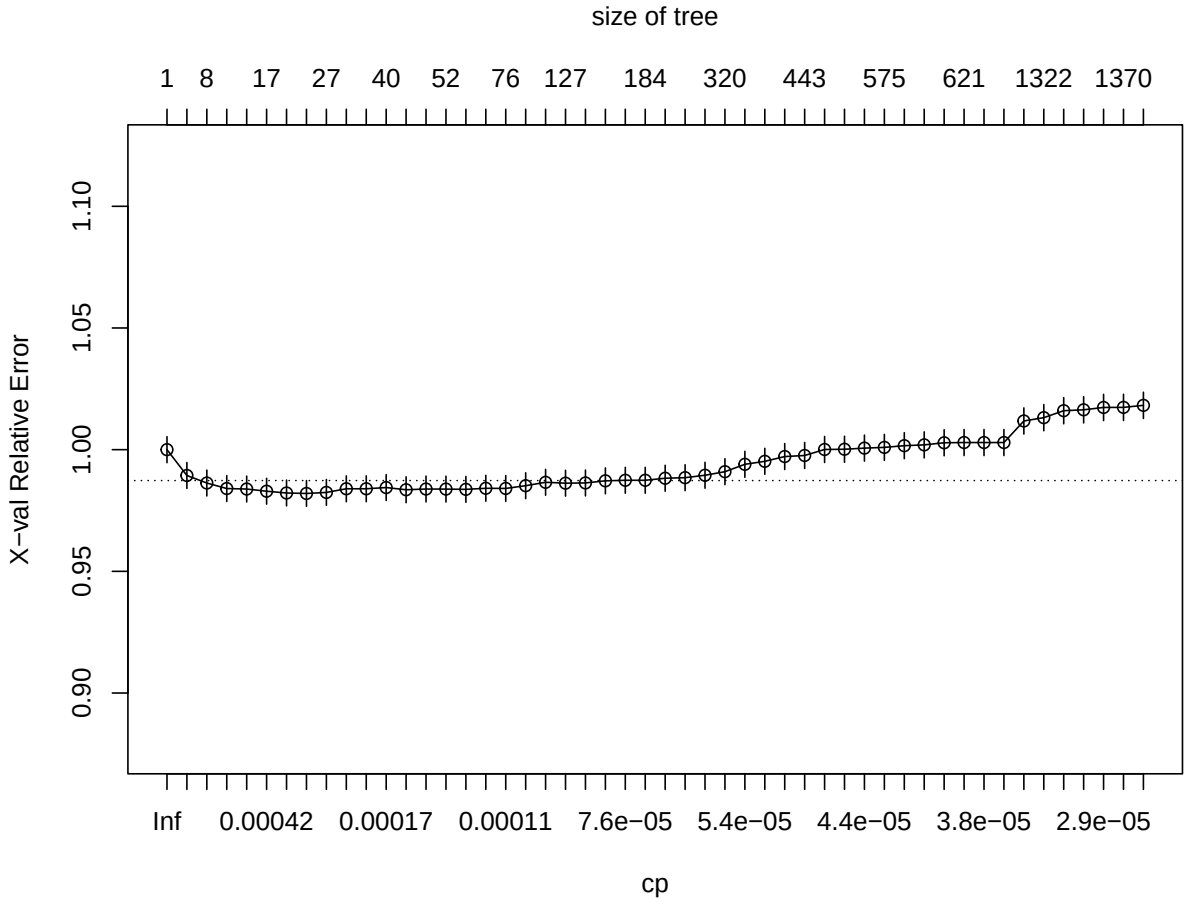
Tree visualisation The tree was pruned to reduce its size and improve the visual.

```r
##Prune the tree
#Determine the optimal complexity parameter (cp)
printcp(dtree_plpot)
```

```
##
## Classification tree:
## rpart::rpart(formula = ..y ~ ., data = data, cp = ~2.78255940220713e-05,
##     minsplit = min_rows(2, data))
##
## Variables actually used in tree construction:
## [1] Age               BMI               Education         HighBP
## [5] HighChol          HvyAlcoholConsump PhysActivity      Sex
## [9] Smoker
##
## Root node error: 30044/215627 = 0.13933
##
## n= 215627
##
##            CP nsplit rel error  xerror      xstd
## 1  2.5230e-03      0   1.00000 1.00000 0.0053523
## 2  1.8306e-03      5   0.98739 0.98938 0.0053284
## 3  6.3241e-04      7   0.98372 0.98629 0.0053213
## 4  5.4919e-04      8   0.98309 0.98402 0.0053162
## 5  5.0592e-04     10   0.98199 0.98382 0.0053158
## 6  3.5282e-04     16   0.97873 0.98293 0.0053137
## 7  3.1620e-04     21   0.97697 0.98219 0.0053120
## 8  2.9956e-04     25   0.97570 0.98199 0.0053116
## 9  2.2467e-04     26   0.97540 0.98243 0.0053126
## 10 1.8861e-04     30   0.97450 0.98392 0.0053160
## 11 1.8306e-04     33   0.97394 0.98392 0.0053160
## 12 1.6642e-04     39   0.97277 0.98442 0.0053171
## 13 1.5533e-04     41   0.97244 0.98352 0.0053151
## 14 1.4978e-04     44   0.97197 0.98382 0.0053158
## 15 1.4423e-04     51   0.97078 0.98379 0.0053157
## 16 1.3314e-04     57   0.96991 0.98369 0.0053155
## 17 1.1650e-04     73   0.96758 0.98412 0.0053164
## 18 1.1095e-04     75   0.96735 0.98406 0.0053163
## 19 9.9854e-05     97   0.96478 0.98519 0.0053189
## 20 9.1532e-05    117   0.96279 0.98659 0.0053220
## 21 8.7372e-05    126   0.96186 0.98622 0.0053212
## 22 8.3211e-05    144   0.95999 0.98632 0.0053214
## 23 7.7664e-05    168   0.95790 0.98719 0.0053234
## 24 7.4890e-05    174   0.95743 0.98742 0.0053239
## 25 7.3226e-05    183   0.95673 0.98742 0.0053239
## 26 7.0267e-05    202   0.95533 0.98825 0.0053258
## 27 6.6569e-05    215   0.95397 0.98848 0.0053263
## 28 5.8248e-05    311   0.94738 0.98948 0.0053286
## 29 5.5474e-05    319   0.94661 0.99095 0.0053319
## 30 5.3255e-05    340   0.94538 0.99398 0.0053387
## 31 4.9927e-05    345   0.94511 0.99521 0.0053415
## 32 4.7549e-05    435   0.94035 0.99717 0.0053459
## 33 4.6598e-05    442   0.94002 0.99760 0.0053469
## 34 4.5766e-05    460   0.93896 1.00007 0.0053524
```

```
## 35 4.4379e-05    468   0.93859 1.00017 0.0053527
## 36 4.3270e-05    556   0.93390 1.00063 0.0053537
## 37 4.2794e-05    574   0.93306 1.00093 0.0053544
## 38 4.1606e-05    581   0.93277 1.00163 0.0053559
## 39 3.9941e-05    597   0.93210 1.00196 0.0053567
## 40 3.9336e-05    602   0.93190 1.00283 0.0053586
## 41 3.8832e-05    620   0.93117 1.00293 0.0053588
## 42 3.8039e-05    665   0.92897 1.00293 0.0053588
## 43 3.7445e-05    682   0.92827 1.00293 0.0053588
## 44 3.3285e-05    710   0.92717 1.01178 0.0053786
## 45 3.0259e-05   1321   0.90607 1.01315 0.0053816
## 46 2.9956e-05   1337   0.90554 1.01601 0.0053880
## 47 2.9586e-05   1351   0.90511 1.01638 0.0053888
## 48 2.9124e-05   1360   0.90484 1.01734 0.0053909
## 49 2.8530e-05   1369   0.90457 1.01737 0.0053910
## 50 2.7826e-05   1424   0.90281 1.01821 0.0053928
```

```
plotcp(dtree_plpot)
```

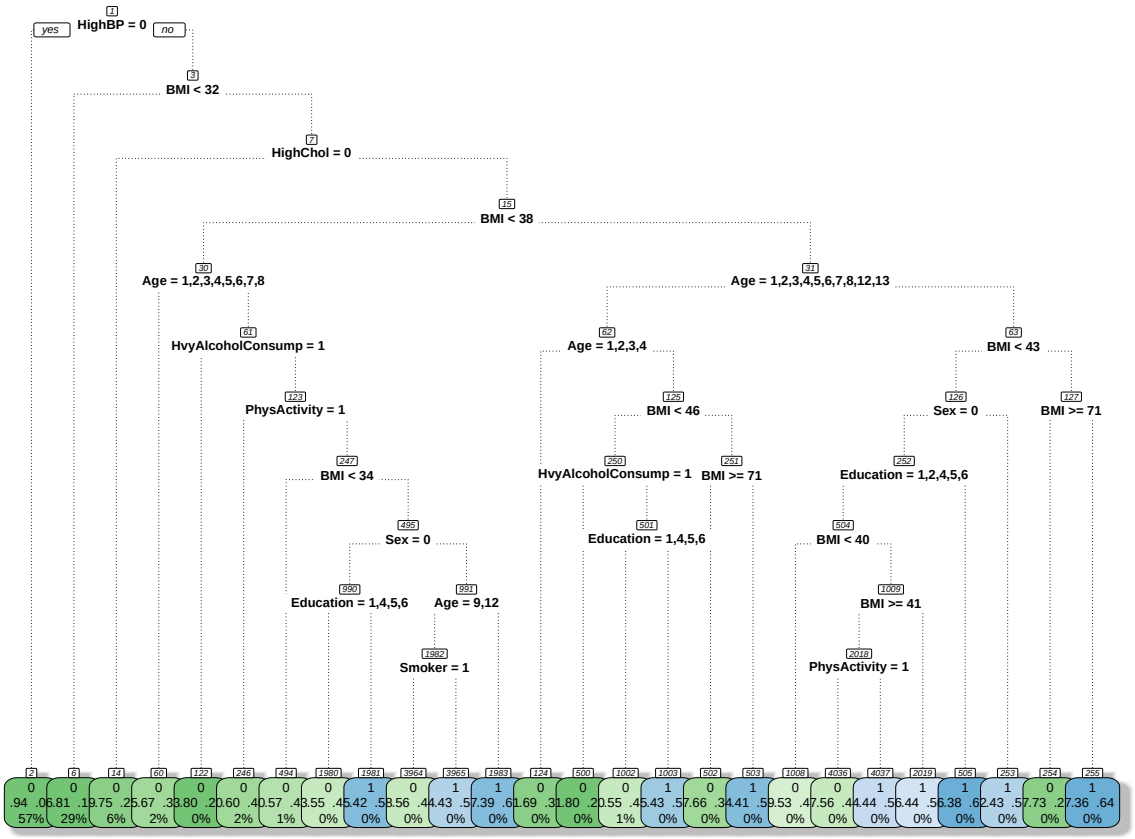



```
# Find the best cp value
best_cp <- dtree_plpot$cptable[which.min(dtree_plpot$cptable[, "xerror"]), "CP"]
best_cp
```

```
## [1] 0.0002995606
```

```
# Prune the tree
dtree_pruned <- rpart::prune(dtree_plpot, cp = best_cp)
```

```
#Plot
rpart.plot(dtree_pruned, type = 0, extra = 104,
           fallen.leaves = TRUE, cex = 0.5,
           box.palette="GnBu",
           branch.lty=3, shadow.col="gray", nn=TRUE,
           roundint=FALSE)
```



Variable Importance

Variable importance in a decision tree quantifies the relative contribution of each input feature to the model's predictive power. It helps understand which features are most influential in determining the outcome.

```
# Variable importance metrics
vip_dtree <- fit_dtree_best %>%
  extract_fit_engine() %>%
  pluck('variable.importance')

vip_dtree <- as.data.frame(vip_dtree)

names(vip_dtree) <- "Values"
```

```
#knitr::kable(vip_dtree)
```

The trained decision tree model evaluation metrics are as follows:

```
## Model Evaluation
dtree_tune_res %>%
  select_by_one_std_err(metric = "accuracy",
                        desc(cost_complexity))
```

```
## # A tibble: 1 x 2
##   cost_complexity .config
##             <dbl> <chr>
## 1        0.000599 pre0_mod05_post0
```

```
dtree_tune_res %>%
  select_by_one_std_err(metric = "roc_auc",
                        desc(cost_complexity))
```

```
## # A tibble: 1 x 2
##   cost_complexity .config
##             <dbl> <chr>
## 1       0.0000278 pre0_mod02_post0
```

The Decision tree algorithm performance

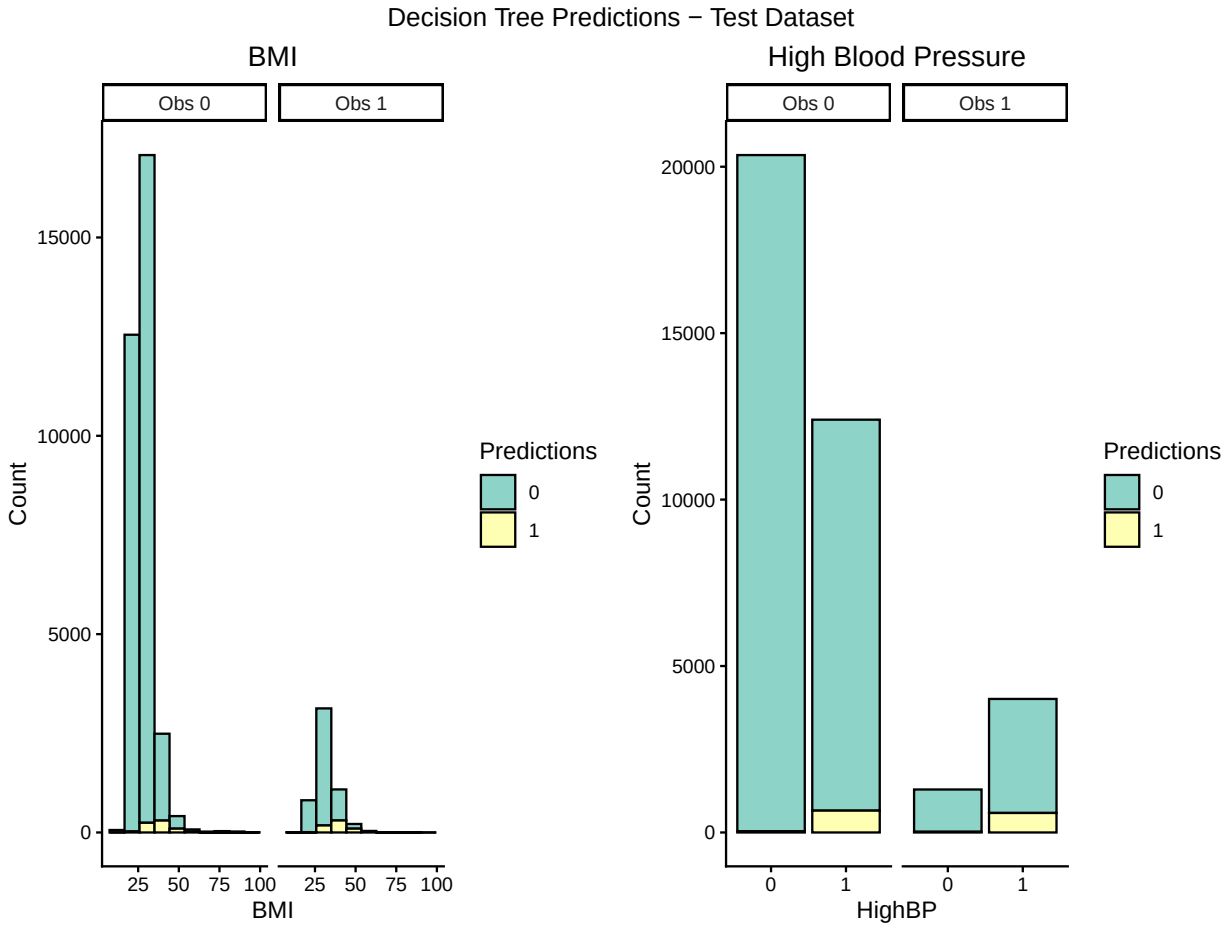
The trained decision tree algorithm was checked by predicting (classifying) diabetes risk on the testing and validation datasets.

i. *Testing dataset*

The diabetes risk classification and the performance evaluation, with visual illustrations using $HighBloodPressure$ and BMI variables, are demonstrated below.

```
# Applying the best model to make predictions on testing dataset
dtree_pred <- predict(fit_dtree_best,
                      new_data = diabetes_test) %>%
  bind_cols(diabetes_test)

dtree_pred_labels <- c("0" = "Obs 0", "1"  = "Obs 1") #Obs - observed


### A plot to illustrate the model performance using BMI as an example

ptest_dtree <- ggplot(dtree_pred,
                aes(x = BMI, fill = .pred_class)) +
  geom_histogram(bins = 10, color = "black") +
  scale_fill_brewer(palette = "Set3") +
  facet_wrap(~Diabetes_binary,
             labeller = labeller(Diabetes_binary = dtree_pred_labels)) +
  labs(y = "Count",
       fill = "Predictions",
```

```
       title = "BMI") +
  theme_classic() +
  theme(plot.title = element_text(hjust = 0.5))

# A plot for High blood pressure
ptest1_dtree <- ggplot(dtree_pred,
                       aes(x = HighBP, fill = .pred_class)) +
  geom_bar(color = "black") +
  scale_fill_brewer(palette = "Set3") +
  facet_wrap(~Diabetes_binary,
             labeller = labeller(Diabetes_binary = dtree_pred_labels)) +
  labs(y = "Count",
       fill = "Predictions",
       title = "High Blood Pressure") +
  theme_classic() +
  theme(plot.title = element_text(hjust = 0.5))

# Combining the plots
gridExtra::grid.arrange(ptest_dtree, ptest1_dtree, ncol = 2,
                        top = "Decision Tree Predictions - Test Dataset")
```



Evaluation of the Decision Tree predictions on the test dataset

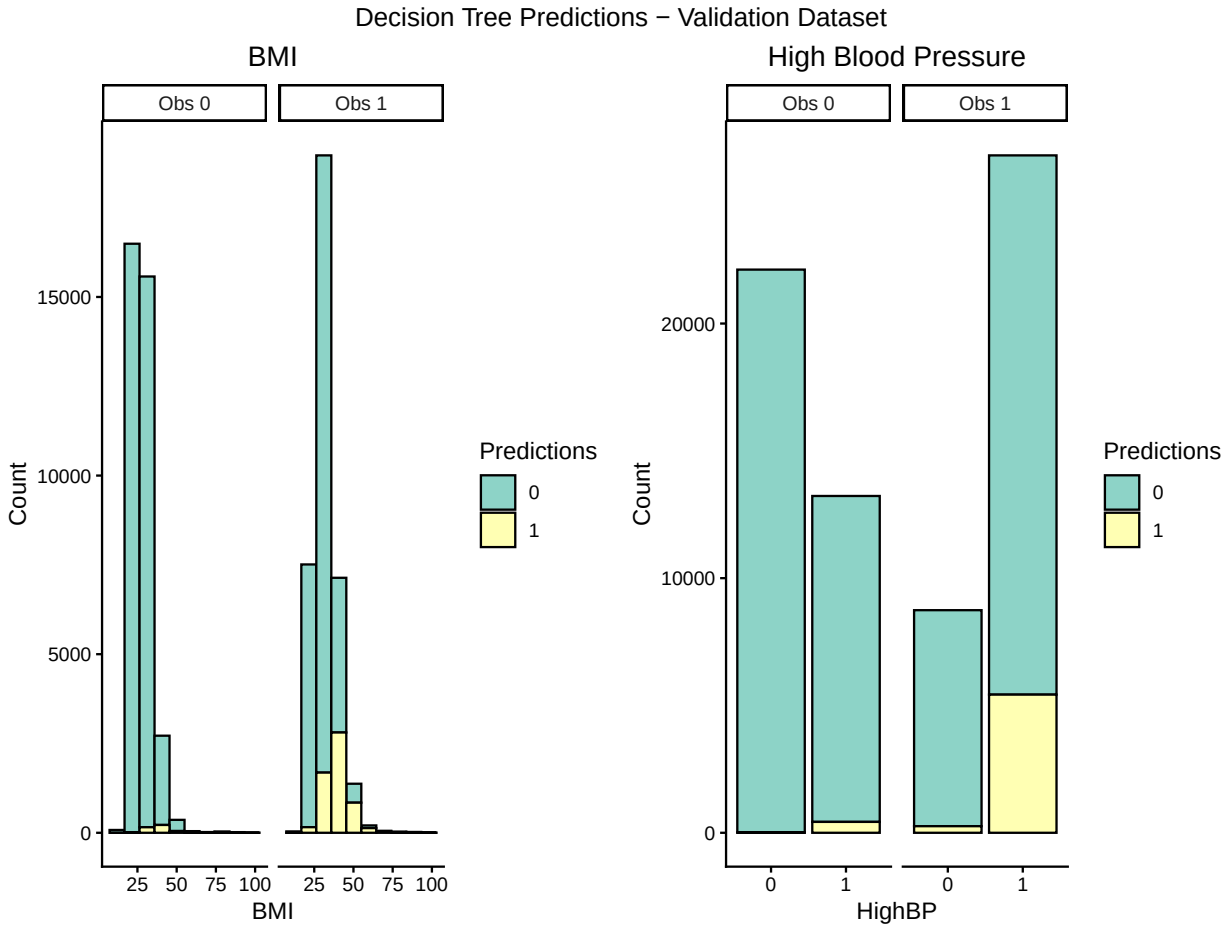
```
## Confusion matrix
confmat_dtree <- confusionMatrix(dtree_pred$.pred_class,
                                 diabetes_test$Diabetes_binary)

accuracy_dtree <- confmat_dtree$overall["Accuracy"]
sensitivity_dtree <- confmat_dtree$byClass["Sensitivity"]
specificity_dtree <- confmat_dtree$byClass["Specificity"]
det_rate_dtree <- confmat_dtree$byClass["Detection Rate"]


models_evaluations <- bind_rows(models_evaluations,
                                tibble(Method = "Decision Tree: test",
                                       Accuracy = accuracy_dtree,
                                       Sensitivity = sensitivity_dtree,
                                       Specificity = specificity_dtree,
                                       Detection_Rate = det_rate_dtree)
                                )

knitr::kable(models_evaluations)
```

ii. *Validation dataset*

Likewise, the built decision tree was applied to the validation dataset to check the performance.

```
# Predictions on validation dataset: 50:50 split of diabetes

dtree_pred_val <- predict(fit_dtree_best,
                          new_data = diabetes_val) %>%
  bind_cols(diabetes_val)

dtree_pred_val_labels <- c("0" = "Obs 0", "1"  = "Obs 1") #Obs - observed

### A plot to illustrate the model performance using BMI as an example
pval_dtree <- ggplot(dtree_pred_val,
                     aes(x = BMI, fill = .pred_class)) +
  geom_histogram(bins = 10, color = "black") +
  scale_fill_brewer(palette = "Set3") +
  facet_wrap(~Diabetes_binary,
             labeller = labeller(Diabetes_binary = dtree_pred_val_labels)) +
  labs(y = "Count",
       fill = "Predictions",
       title = "BMI") +
  theme_classic() +
  theme(plot.title = element_text(hjust = 0.5))


# A plot for High blood pressure
pval1_dtree <- ggplot(dtree_pred_val,
                      aes(x = HighBP, fill = .pred_class)) +
  geom_bar(color = "black") +
  scale_fill_brewer(palette = "Set3") +
  facet_wrap(~Diabetes_binary,
             labeller = labeller(Diabetes_binary = dtree_pred_val_labels)) +
```

```
  labs(y = "Count",
       fill = "Predictions",
       title = "High Blood Pressure") +
  theme_classic() +
  theme(plot.title = element_text(hjust = 0.5))

# Combining the plots
gridExtra::grid.arrange(pval_dtree, pval1_dtree, ncol = 2,
                        top = "Decision Tree Predictions - Validation Dataset")
```



Evaluation of the Decision Tree predictions on the validation dataset

```
## Confusion matrix
confmat_dtree_val <- confusionMatrix(dtree_pred_val$.pred_class,
                                     diabetes_val$Diabetes_binary)

accuracy_dtree_val <- confmat_dtree_val$overall["Accuracy"]
sensitivity_dtree_val <- confmat_dtree_val$byClass["Sensitivity"]
specificity_dtree_val <- confmat_dtree_val$byClass["Specificity"]
det_rate_dtree_val <- confmat_dtree_val$byClass["Detection Rate"]


models_evaluations <- bind_rows(models_evaluations,
```

```
                                tibble(Method = "Decision Tree: validation",
                                       Accuracy = accuracy_dtree_val,
                                       Sensitivity = sensitivity_dtree_val,
                                       Specificity = specificity_dtree_val,
                                       Detection_Rate = det_rate_dtree_val)
                                )

knitr::kable(models_evaluations)
```

4. **Random Forest**

A Random Forest is an ensemble Machine Learning algorithm that is used for both classification and regression tasks. The principle of Random Forest is to construct a "forest" of many individual decision trees and then aggregate their predictions[17]. The `randomForest` package was used to build the algorithm. The training and testing of the Random Forest classification algorithm are shown below.

```
# Set seed
set.seed(123)

# Model Specification
rf_spec <- rand_forest() %>%
  set_engine(engine = "ranger") %>%
  set_args(
    mtry = NULL, # size of random subset of variables
    trees = 1000, # Number of trees
    min_n = 2,
    probability = FALSE, # FALSE: get hard predictions
    importance = "impurity"
  ) %>%
  set_mode("classification")

# Recipe
rf_rec <- recipe(Diabetes_binary ~ ., data = diabetes_train)

# Workflows
rf_wf <- workflow() %>%
  add_model(rf_spec) %>%
  add_recipe(rf_rec)

# No tune_grid() or vfold_cv()
rf_fit <- fit(rf_wf, data = diabetes_train)
```

Variable importance

Variable importance in a random forest quantifies the magnitude to which each feature contributes to the model's predictive accuracy. It is calculated in two main ways: mean decrease impurity (MDI), which measures how much a feature decreases impurity (like Gini impurity) across all splits in the trees, and permutation importance, which assesses how much the model's performance drops after randomly shuffling a feature's values. Variables with higher importance scores are more influential in the model's predictions[18].

[17]Mushagalusa, C.A., Fandohan, A.B. & Glèlè Kakaï, R. (2024): Random Forest and spatial cross-validation performance in predicting species abundance distributions. Environ Syst Res 13:23. https://doi.org/10.1186/s40068-024-00352-9

[18]Irizarry R. A: Introduction to Data Science. Statistics and Prediction Algorithms Through Case Studies. Accessed on 10 October 2025. url: https://rafalab.dfci.harvard.edu/dsbook-part-2/ml/ml-in-practice.html

```r
## Variable Importance
# Plot of the variable importance information
rf_vip_p <- rf_fit %>%
  extract_fit_engine() %>%
  vip(num_features = 30) + theme_classic()

#rf_vip_p
```

The numerical values of the variable importance are:

```r
# Extract the numerical information on variable importance
rf_var_imp <- rf_fit %>%
  extract_fit_engine() %>%
  vip::vi()

#knitr::kable(as.data.frame(rf_var_imp))
```

The evaluation metrics of the trained Random Forest model are shown below.

```r
## Model Evaluation
rf_fit
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: rand_forest()
##
## -- Preprocessor ----------------------------------------------------------------
## 0 Recipe Steps
##
## -- Model -----------------------------------------------------------------------
## Ranger result
##
## Call:
##  ranger::ranger(x = maybe_data_frame(x), y = y, num.trees = ~1000,      min.node.size = min_rows(~2,
##
## Type:                             Classification
## Number of trees:                  1000
## Sample size:                      215627
## Number of independent variables:  9
## Mtry:                             3
## Target node size:                 2
## Variable importance mode:         impurity
## Splitrule:                        gini
## OOB prediction error:             13.67 %
```

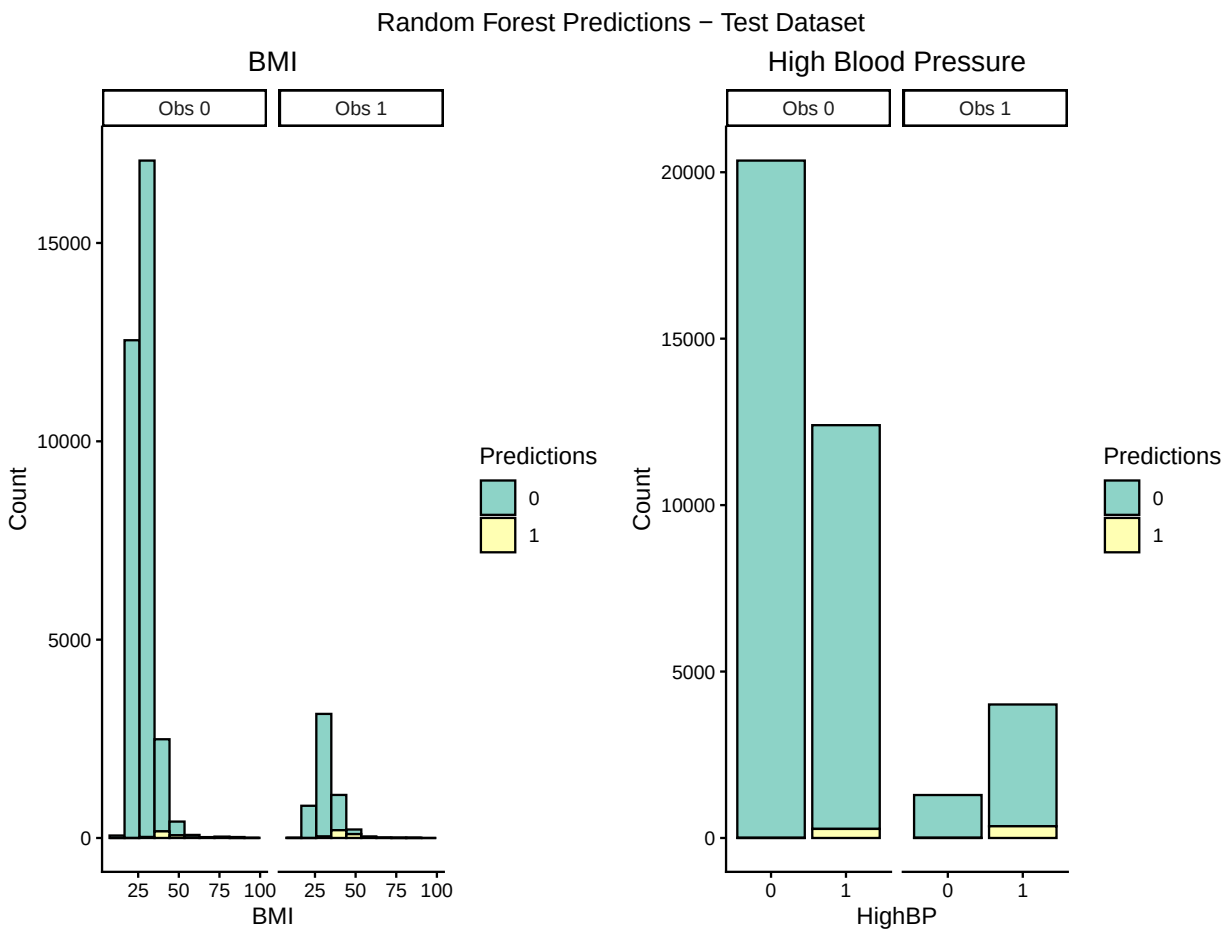
The Random forest performance

The trained random forest model performance was checked by predicting (classifying) diabetes risk on the testing and validation datasets.

i. *Testing dataset*

The diabetes risk classification and the performance evaluation, with visual illustrations using $HighBloodPressure$ and BMI variables, are demonstrated below.

```
# Applying the best model to make predictions on testing dataset
rf_pred <- predict(rf_fit,
                   new_data = diabetes_test) %>%
  bind_cols(diabetes_test)

rf_pred_labels <- c("0" = "Obs 0", "1"  = "Obs 1") #Obs - observed

# A plot to illustrate the model performance using BMI as example
ptest_rf <- ggplot(rf_pred,
                   aes(x = BMI, fill = .pred_class)) +
  geom_histogram(bins = 10, color = "black") +
  scale_fill_brewer(palette = "Set3") +
  facet_wrap(~Diabetes_binary,
             labeller = labeller(Diabetes_binary = rf_pred_labels)) +
  labs(y = "Count",
       fill = "Predictions",
       title = "BMI") +
  theme_classic() +
  theme(plot.title = element_text(hjust = 0.5))

# A plot for High blood pressure
ptest1_rf <- ggplot(rf_pred,
                    aes(x = HighBP, fill = .pred_class)) +
  geom_bar(color = "black") +
  scale_fill_brewer(palette = "Set3") +
  facet_wrap(~Diabetes_binary,
             labeller = labeller(Diabetes_binary = rf_pred_labels)) +
  labs(y = "Count",
       fill = "Predictions",
       title = "High Blood Pressure") +
  theme_classic() +
  theme(plot.title = element_text(hjust = 0.5))

# Combining the plots
gridExtra::grid.arrange(ptest_rf, ptest1_rf, ncol = 2,
                        top = "Random Forest Predictions - Test Dataset")
```



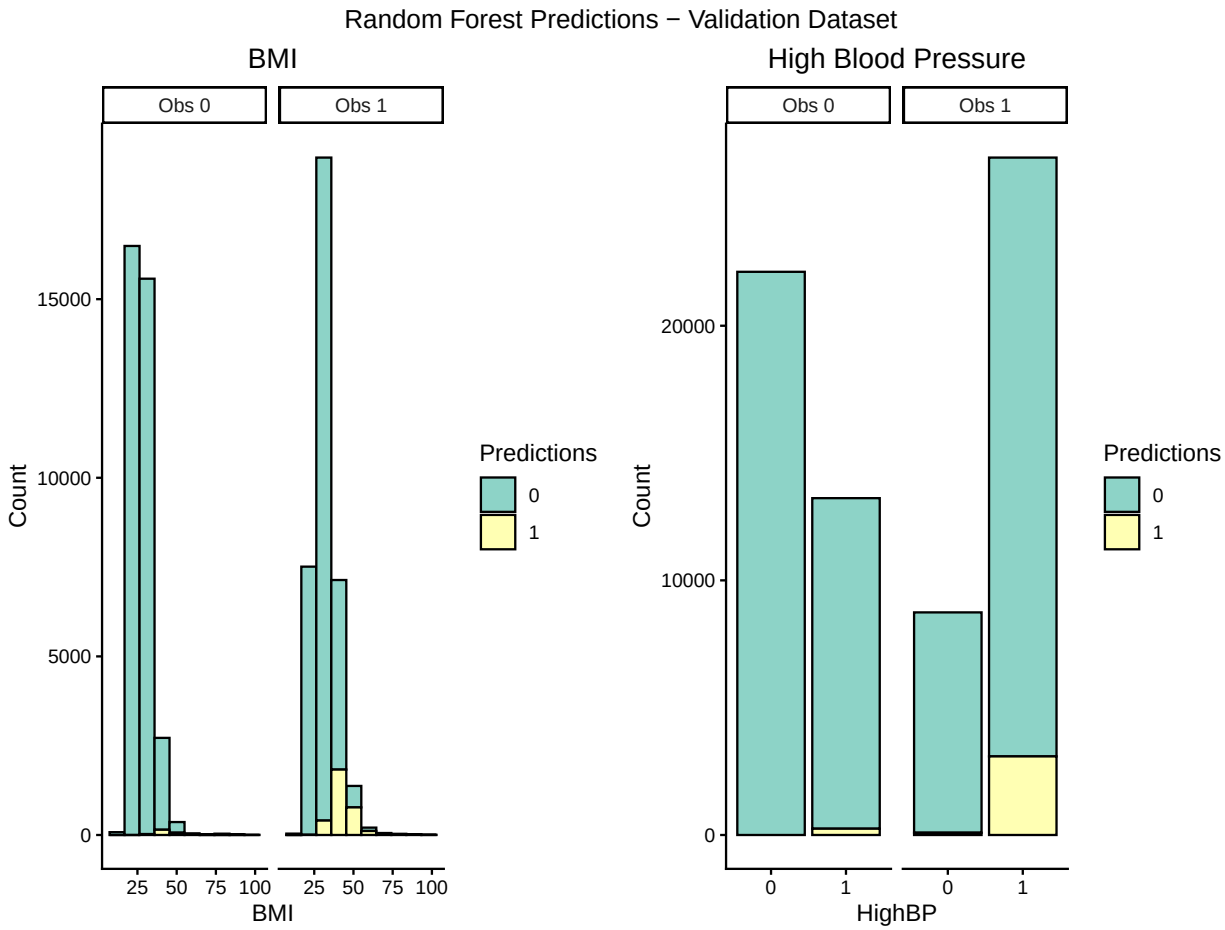
Evaluation of the Random Forest predictions on the test dataset

```
## Confusion matrix
confmat_rf <- confusionMatrix(rf_pred$.pred_class,
                              diabetes_test$Diabetes_binary)

accuracy_rf <- confmat_rf$overall["Accuracy"]
sensitivity_rf <- confmat_rf$byClass["Sensitivity"]
specificity_rf <- confmat_rf$byClass["Specificity"]
det_rate_rf <- confmat_rf$byClass["Detection Rate"]



models_evaluations <- bind_rows(models_evaluations,
                                tibble(Method = "Random Forest: test",
                                       Accuracy = accuracy_rf,
                                       Sensitivity = sensitivity_rf,
                                       Specificity = specificity_rf,
                                       Detection_Rate = det_rate_rf)
                                )

knitr::kable(models_evaluations)
```

ii. *Validation dataset*

Likewise, the trained random forest algorithm was applied on the validation dataset to check the performance.

```
# Predictions on validation dataset: 50:50 split of diabetes
rf_pred_val <- predict(rf_fit,
                       new_data = diabetes_val) %>%
  bind_cols(diabetes_val)

rf_pred_val_labels <- c("0" = "Obs 0", "1"  = "Obs 1") #Obs - observed

# A plot to illustrate the model performance using BMI as example
pval_rf <- ggplot(rf_pred_val,
                  aes(x = BMI, fill = .pred_class)) +
  geom_histogram(bins = 10, color = "black") +
  scale_fill_brewer(palette = "Set3") +
  facet_wrap(~Diabetes_binary,
             labeller = labeller(Diabetes_binary = rf_pred_val_labels)) +
  labs(y = "Count",
       fill = "Predictions",
       title = "BMI") +
  theme_classic() +
  theme(plot.title = element_text(hjust = 0.5))



# A plot for High blood pressure
pval1_rf <- ggplot(rf_pred_val,
                   aes(x = HighBP, fill = .pred_class)) +
  geom_bar(color = "black") +
  scale_fill_brewer(palette = "Set3") +
  facet_wrap(~Diabetes_binary,
             labeller = labeller(Diabetes_binary = rf_pred_val_labels)) +
  labs(y = "Count",
       fill = "Predictions",
       title = "High Blood Pressure") +
  theme_classic()  +
  theme(plot.title = element_text(hjust = 0.5))

# Combining the plots
gridExtra::grid.arrange(pval_rf, pval1_rf, ncol = 2,
                        top = "Random Forest Predictions - Validation Dataset")
```



Evaluation of the Random Forest predictions on the validation dataset

```
## Confusion matrix
confmat_rf_val <- confusionMatrix(rf_pred_val$.pred_class,
                                  diabetes_val$Diabetes_binary)

accuracy_rf_val <- confmat_rf_val$overall["Accuracy"]
sensitivity_rf_val <- confmat_rf_val$byClass["Sensitivity"]
specificity_rf_val <- confmat_rf_val$byClass["Specificity"]
det_rate_rf_val <- confmat_rf_val$byClass["Detection Rate"]


models_evaluations <- bind_rows(models_evaluations,
                                tibble(Method = "Random Forest: validation",
                                       Accuracy = accuracy_rf_val,
                                       Sensitivity = sensitivity_rf_val,
                                       Specificity = specificity_rf_val,
                                       Detection_Rate = det_rate_rf_val)
                                )

knitr::kable(models_evaluations)
```

Results

The Models' Variable Importance

The variable importance of the features in classifying diabetes risk across the various Machine Learning techniques is presented below, except for the KNN model, which does not inherently provide a direct measure of variable importance.

Table 1: LASSO Variable Importance

Variable	Importance	Sign
Age_X11	1.2376882	POS
Age_X12	1.2205201	POS
Age_X10	1.1891730	POS
Age_X13	1.1301548	POS
Age_X9	1.0282893	POS
HighBP_X1	0.9641727	POS
HvyAlcoholConsump_X1	0.9369267	NEG
Age_X8	0.8000033	POS
Age_X2	0.7468015	NEG
Age_X7	0.7142137	POS
HighChol_X1	0.6720536	POS
Education_X2	0.5759093	POS
Age_X6	0.4689436	POS
BMI	0.4540882	POS
Age_X3	0.4035638	NEG
Education_X6	0.3319873	NEG
Education_X3	0.3043158	POS
PhysActivity_X1	0.2985550	NEG
Age_X5	0.2575672	POS
Sex_X1	0.1977871	POS
Smoker_X1	0.0862021	POS
Education_X5	0.0414175	NEG
Education_X4	0.0272503	POS
Age_X4	0.0000000	NEG

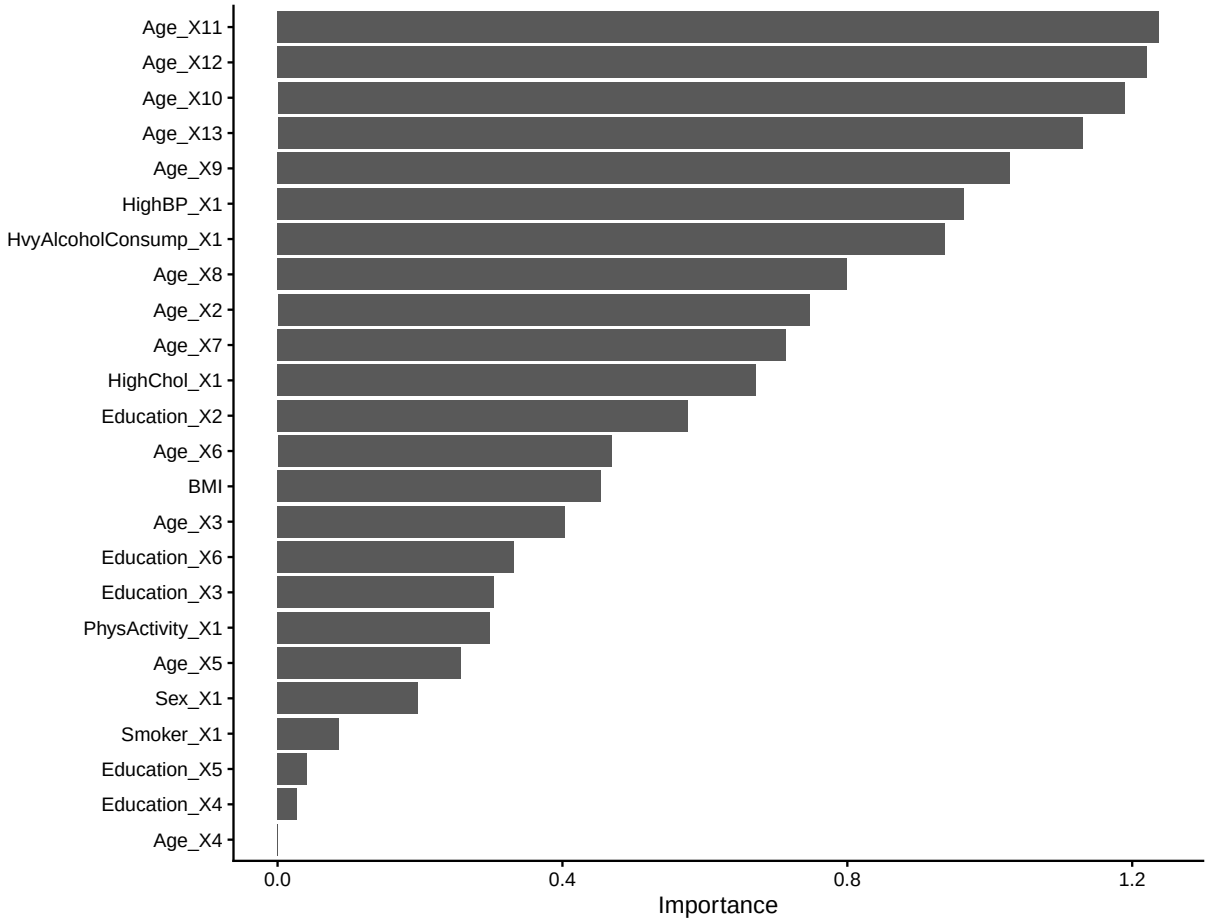


Figure R1: LASSO Variable Importance

Table 2: Ridge Variable Importance

Variable	Importance	Sign
Age_X2	1.0029177	NEG
HighBP_X1	0.9190531	POS
Age_X3	0.8298627	NEG
HvyAlcoholConsump_X1	0.8158064	NEG
HighChol_X1	0.6475050	POS
Education_X2	0.6435637	POS
Age_X11	0.5544242	POS
Age_X4	0.5370037	NEG
Age_X12	0.5348468	POS
Age_X10	0.5095605	POS
Age_X13	0.4407227	POS
BMI	0.4205254	POS
Education_X3	0.3782890	POS
Age_X9	0.3560637	POS
Age_X5	0.3364185	NEG
PhysActivity_X1	0.2995594	NEG
Education_X6	0.2348949	NEG

Variable	Importance	Sign
Sex_X1	0.1747079	POS
Age_X6	0.1581862	NEG
Age_X8	0.1414296	POS
Education_X4	0.1123685	POS
Smoker_X1	0.0940102	POS
Age_X7	0.0606175	POS
Education_X5	0.0366052	POS

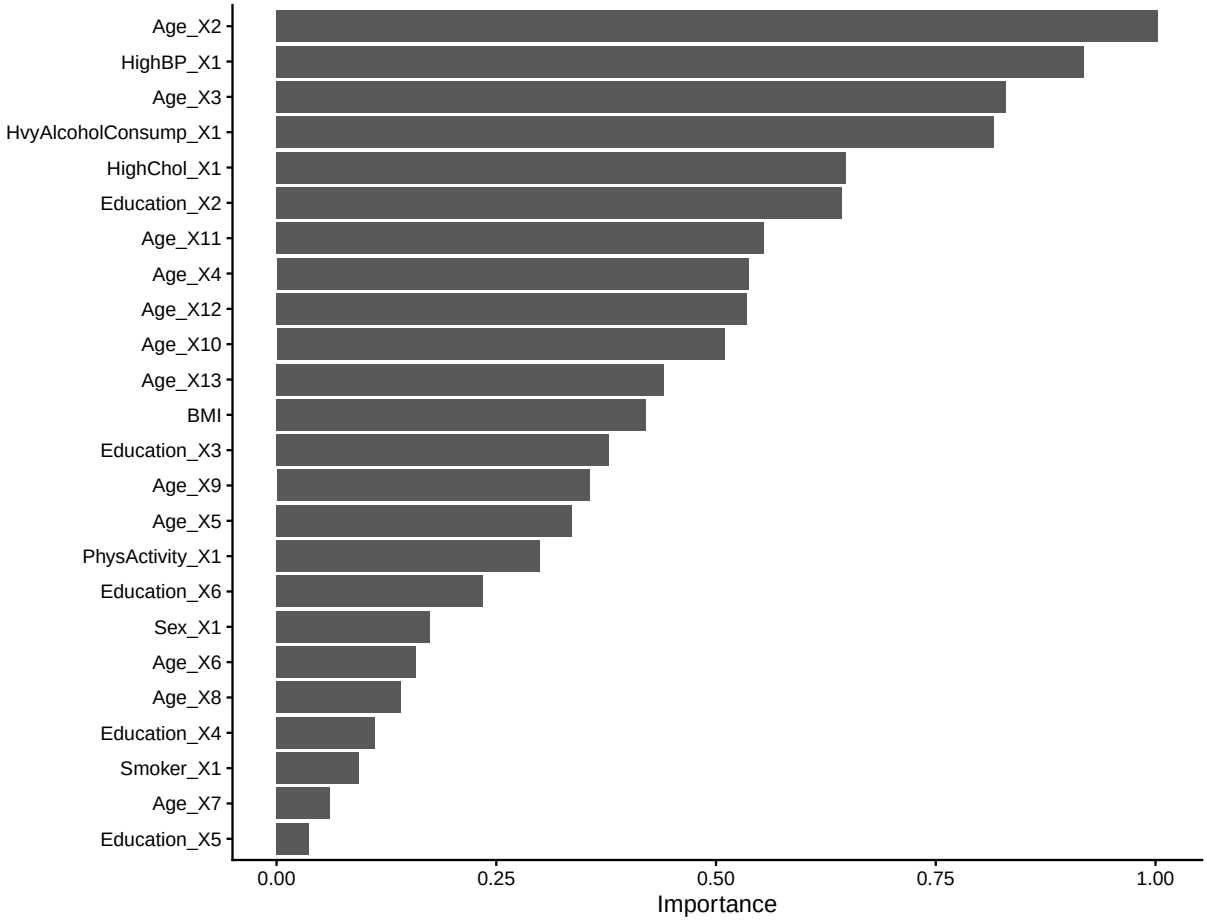


Figure R2: Ridge Variable Importance

Table 1 and Figure R1 show variable importance in the LASSO logistic regression, and Table 2 and Figure R2 show variable importance in the Ridge logistic regression model. In both the LASSO and Ridge logistic regression models, age, high blood pressure, heavy alcohol consumption, and high cholesterol were the features that contributed most to the diabetes risk classification. The main difference is the age category: older age contributes most to the LASSO model, whereas younger age is the most important contributing feature in the Ridge model.

Table 3: Decision Tree Variable Importance

	Values
HighBP	3575.4615
BMI	2836.1856
Age	1552.6984
HighChol	1547.5617
Education	666.5414
PhysActivity	346.9649
Smoker	207.7422
Sex	169.9359
HvyAlcoholConsump	121.7374

Table 4: Random Forest Variable Importance

Variable	Importance
BMI	3460.3367
HighBP	2578.7762
Age	1770.0524
HighChol	1293.4348
Education	903.1202
PhysActivity	414.3477
Sex	318.4288
Smoker	284.0024
HvyAlcoholConsump	249.4985

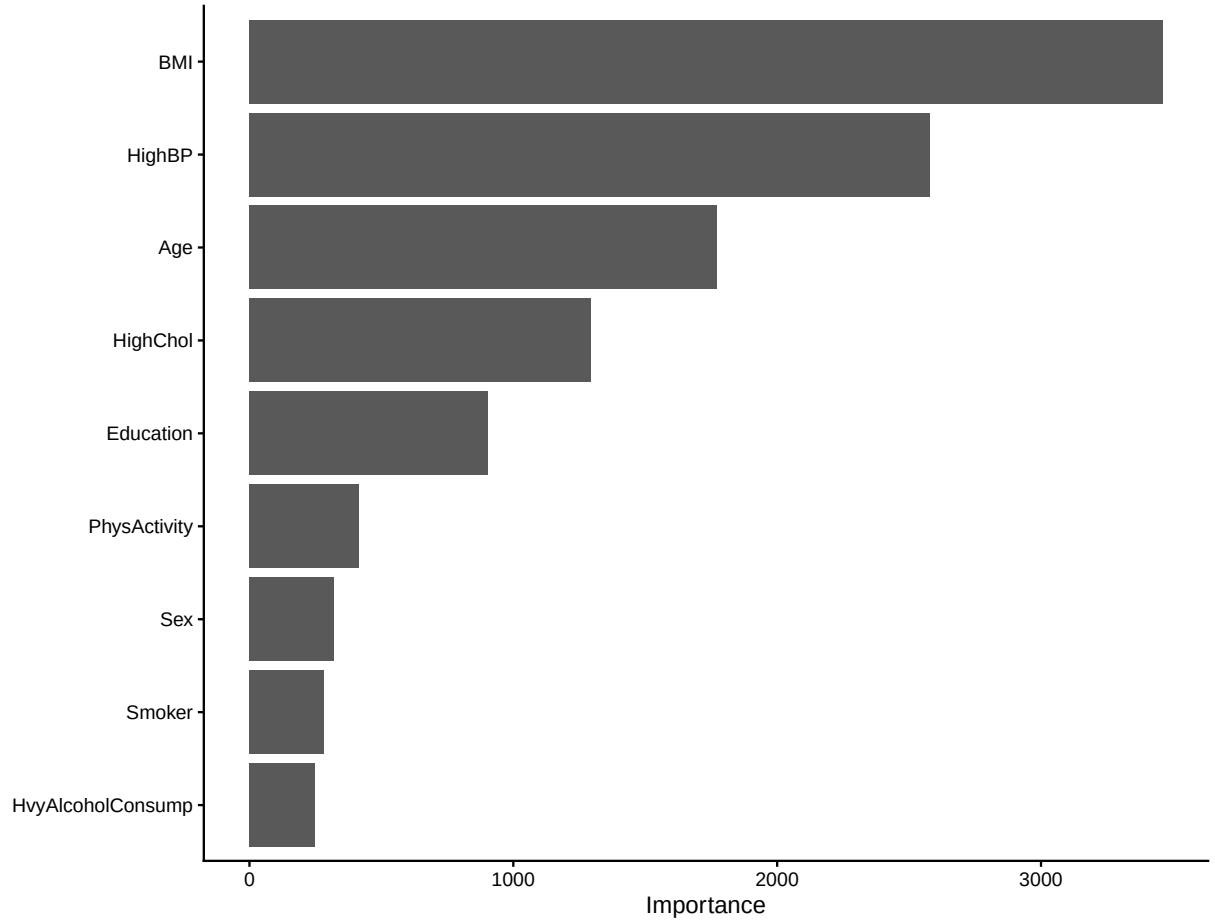


Figure R3: Random Forest Variable Importance

Furthermore, Table 3 shows variable importance in the Decision Tree model. High blood pressure, BMI, age and high cholesterol contributed most to the classification tree. The variable importance of the Random Forest model, presented in Table 4 and Figure R3, indicates that BMI, high blood pressure, age, and high cholesterol are the most important features. In both the Decision Tree and the Random Forest models, heavy alcohol consumption contributed the least to the diabetes risk classification, in contrast to the contribution in the LASSO and Ridge logistic regression. While the contribution of BMI to the LASSO and Ridge models is equally substantial, the importance is not as significant as in the Decision Tree and Random Forest models. The contributions of the other features – education level, physical activity, smoking status, and gender (sex) – are relatively low and of similar importance in all the models.

Performance of the models' predictions

The models' prediction evaluations are shown in Table 5. The models' accuracy in predicting diabetes risk on the testing dataset is higher than on the validation dataset.

Table 5: Algorithms Performance Evaluation

Method	Accuracy	Sensitivity	Specificity	Detection_Rate
Logistic regression (LASSO): test	0.8607469	0.9911758	0.0550736	0.8530733
Logistic regression (LASSO): validation	0.5265518	0.9917388	0.0613648	0.4958694
Logistic regression (Ridge): test	0.8607469	0.9979237	0.0113165	0.8588810
Logistic regression (Ridge): validation	0.5055593	0.9980196	0.0130991	0.4990098
K-Nearest Neighbors (KNN): test	0.8611148	0.9894049	0.0686533	0.8515492
K-Nearest Neighbors (KNN): validation	0.5393821	0.9916822	0.0870820	0.4958411
Decision Tree: test	0.8584080	0.9788709	0.1142965	0.8424829
Decision Tree: validation	0.5740961	0.9873536	0.1608386	0.4936768
Random Forest: test	0.8627703	0.9915728	0.0671445	0.8534150
Random Forest: validation	0.5414191	0.9930119	0.0898263	0.4965060

The average accuracy across the algorithms is about 0.86 on the testing dataset and 0.54 on the validation dataset. Among the Machine Learning algorithms' performance on the testing dataset, the Random Forest performed best, with an accuracy of 0.8625. This was followed by the KNN algorithm, with an accuracy of 0.8611. The accuracy of the LASSO and Ridge logistic regression model is the same on the testing dataset (0.8607). The Decision Tree performed worst on the test dataset, with an accuracy of 0.8584; however, it performed better on the validation dataset, achieving an accuracy of 0.5741. The LASSO and Ridge logistic regression models' performance on the validation dataset was the worst, with the Ridge model having the lowest accuracy of 0.5056. The performance accuracy on the validation dataset of the Random Forest and KNN models was 0.5411 and 0.5394, respectively.

Conclusion

This final capstone project, Choose Your Own, utilised a supervised Machine Learning technique to train algorithms – *Logistic Regression, K-Nearest Neighbour (KNN), Decision Tree, and Random Forest* – for diabetes risk classification using the Behavioural Risk Factor Surveillance System (BRFSS) dataset available at the CDC, and examined the variable importance and the accuracy of the trained models.

Among the models trained, with the exception of the KNN algorithm, which lacks inherent variable importance in the modelling, similar features were observed to contribute to the classification of diabetes risk; however, the magnitude of the contribution varied with models. Some features, such as age, high blood pressure, high cholesterol, and BMI, were important and contributed substantially to all the models. Also, some features were observed to be important in some models but not in others. For example, heavy alcohol consumption was an important variable in the LASSO and Ridge logistic regression but not in the decision Tree and Random Forest models. In general, the features (variables) that contributed most to the models are important risk factors for diabetes. This is consistent with evidence on diabetes risk factors in the general population[19].

The trained models' performance in predicting diabetes risk on the testing dataset was high, with an average accuracy of 0.86. The Random Forest algorithm (0.8625) had the highest accuracy, and the Decision Tree (0.8584) had the lowest. The observed accuracy of these Machine Learning algorithms in this project is consistent with previous findings in the literature using the same Machine Learning techniques. Studies have reported classification accuracies ranging from 0.69 to 0.82, including those using KNN and Decision Tree algorithms[20]. However, the observed algorithms' accuracy in this project is slightly higher than that

[19]Hossain MJ, Al-Mamun M, Islam MR. (2024): Diabetes mellitus, the fastest growing global public health concern: Early detection should be focused. Health Sci Rep. 7(3):e2004. doi: 10.1002/hsr2.2004.

[20]Tasin I, Nabil TU, Islam S, Khan R. (2022): Diabetes prediction using machine learning and explainable AI techniques. Healthc Technol Lett. 10(1-2):1-10. doi: 10.1049/htl2.12039.

reported in previous studies. The difference could partly be due to differences in the population, the size of the training datasets and the features in the datasets.

Furthermore, the models' accuracy on the validation dataset was very low, ranging from 0.50 (Ridge logistic regression – 0.5056) to 0.57 (Decision Tree – 0.5741). The poor performance of the models on the validation dataset might be due to differences in the characteristics of the training and the validation datasets. The prevalence of diabetes risk in the training dataset is lower than in the validation dataset. The validation dataset is a 50:50 split between diabetes risk and no diabetes; thus, the diabetes risk prevalence is significantly higher compared to the training dataset. Therefore, the trained algorithms performed as though they were overfitted when applied to the validation dataset. It is important to note that the validation dataset is an unrealistic population sample, as the general population prevalence of diabetes risk about 11.6% in the USA[21].

Although several Machine Learning algorithms were utilised in this project to determine the model with the best performance, other algorithms, such as Support Vector Machine (SVM), are superior for classification tasks[22]. For a large dataset, as in this project, SVM can be computationally intensive and time-consuming; therefore, it was not considered.

In conclusion, Machine Learning techniques were used to train algorithms for diabetes risk classification, using features which are common risk factors of diabetes. The accuracy of the trained algorithms on the testing dataset, which has the same characteristics as the training dataset, was high. However, it was low on a validation dataset, which is an unrealistic population sample with a 50% prevalence of diabetes risk.

Acknowledgment

There were several useful online materials, and the course textbook, which serves as a guide in completing this project, is cited in the report and has been helpful throughout the course.

Bibliography

- Abnoosian, K., Farnoosh, R. & Behzadi, M.H. (2023): Prediction of diabetes disease using an ensemble of machine learning multi-classifier models. BMC Bioinformatics 24: 337. https://doi.org/10.1186/s12859-023-05465-z
- Açıkkar, M. & Tokgöz, S. (2025): Improving multi-class classification: scaled extensions of harmonic mean-based adaptive k-nearest neighbors. Appl Intell 55:168. https://doi.org/10.1007/s10489-024-06109-2
- Alanazi R. (2022): Identification and Prediction of Chronic Diseases Using Machine Learning Approach. J Healthc Eng. 2022:2826127. doi: 10.1155/2022/2826127.
- Alex Teboul (2021): Diabetes Health Indicators Dataset; Kaggle. Available at https://www.kaggle.com/datasets/alexteboul/diabetes-health-indicators-dataset
- Branch N. (n.d): Machine Learning Final Project: Predicting Diabetes. Available at https://rpubs.com/nbranch/preddiabetes

[21]CDC: Diabetes, National Diabetes Statistics Report. https://www.cdc.gov/diabetes/php/data-research/index.html

[22]Rodríguez-Pérez R, Bajorath J. (2022): Evolution of Support Vector Machine and Regression Modeling in Chemoinformatics and Drug Discovery. J Comput Aided Mol Des. 36(5):355-362. doi: 10.1007/s10822-022-00442-9.

- Burrows NR, Hora I, Geiss LS, Gregg EW, Albright A. (2017): Incidence of End-Stage Renal Disease Attributed to Diabetes Among Persons with Diagnosed Diabetes — United States and Puerto Rico, 2000–2014. MMWR Morb Mortal Wkly Rep, 66:1165–1170. DOI: http://dx.doi.org/10.15585/mmwr.mm6643a2
- CDC: Diabetes, National Diabetes Statistics Report. https://www.cdc.gov/diabetes/php/data-research/index.html
- Chen, X., Zhang, L., & Chen, W. (2025): Global, regional, and national burdens of type 1 and type 2 diabetes mellitus in adolescents from 1990 to 2021, with forecasts to 2030: a systematic analysis of the global burden of disease study 2021. BMC medicine, 23(1), 48. https://doi.org/10.1186/s12916-025-03890-w
- Halder, R.K., Uddin, M.N., Uddin, M.A. et al. (2024): Enhancing K-nearest neighbor algorithm: a comprehensive review and performance analysis of modifications. J Big Data 11:113. https://doi.org/10.1186/s40537-024-00973-y
- Herawati N., Wijayanti A., Sutrisno A., & Misgiyati N. (2024). The Performance of Ridge Regression, LASSO, and Elastic-Net in Controlling Multicollinearity: A Simulation and Application. Journal of Modern Applied Statistical Methods, 23(2). https://doi.org/10.56801/Jmasm.V23.i2.4
- Hossain MJ, Al-Mamun M, Islam MR. (2024): Diabetes mellitus, the fastest growing global public health concern: Early detection should be focused. Health Sci Rep. 7(3):e2004. doi: 10.1002/hsr2.2004.
- Irizarry R. A: Introduction to Data Science. Statistics and Prediction Algorithms Through Case Studies. Accessed on 10 October 2025. url: https://rafalab.dfci.harvard.edu/dsbook-part-2/ml/ml-in-practice.html
- Mushagalusa, C.A., Fandohan, A.B. & Glèlè Kakaï, R. (2024): Random Forest and spatial cross-validation performance in predicting species abundance distributions. Environ Syst Res 13:23. https://doi.org/10.1186/s40068-024-00352-9
- Rodríguez-Pérez R, Bajorath J. (2022): Evolution of Support Vector Machine and Regression Modeling in Chemoinformatics and Drug Discovery. J Comput Aided Mol Des. 36(5):355-362. doi: 10.1007/s10822-022-00442-9.
- Rokach, L. & Maimon, O. (2005): Decision trees. Data Mining and Knowledge Discovery Handbook, Springer, 165-192
- Tasin I, Nabil TU, Islam S, Khan R. (2022): Diabetes prediction using machine learning and explainable AI techniques. Healthc Technol Lett. 10(1-2):1-10. doi: 10.1049/htl2.12039.
- World Health Organisation, Diabetes. Available at: https://www.who.int/news-room/fact-sheets/detail/diabetes
- Zhang, B., Shi, H., & Wang, H. (2023). Machine Learning and AI in Cancer Prognosis, Prediction, and Treatment Selection: A Critical Approach. Journal of multidisciplinary healthcare, 16, 1779–1791. https://doi.org/10.2147/JMDH.S410301
- Zhou, Bin et al. (2024): Worldwide trends in diabetes prevalence and treatment from 1990 to 2022: a pooled analysis of 1108 population-representative studies with 141 million participants. The Lancet, 404(10467):2077 – 2093