Model comparison and analyses of effects

Loasding libraries and data

```r
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```r
library(rattle)
```

```
## Loading required package: tibble

## Loading required package: bitops

## Rattle: A free graphical interface for data science with R.
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```r
library(glmnet)
```

```
## Loading required package: Matrix

## Loaded glmnet 4.0-2
```

```r
library(ROCR)
library(MASS)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following object is masked from 'package:MASS':
##
##     select

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(gbm)
```

```
## Loaded gbm 2.1.8
```

```r
library(e1071)
library(cvAUC)
```

```
## Loading required package: data.table

##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

##

## cvAUC version: 1.1.0

## Notice to cvAUC users: Major speed improvements in version 1.1.0

##
```

```r
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```r
library(RColorBrewer)
library(corrplot)
library(ggpubr)
library(PMCMR)
```

```
## PMCMR is superseded by PMCMRplus and will be no longer maintained. You may wish to install PMCMRplus
```

```r
library(devtools)
```

```
## Loading required package: usethis
```

```r
library(PMCMR)
library(PMCMRplus)
```

```
## Registered S3 methods overwritten by 'PMCMRplus':
##   method        from
##   print.PMCMR   PMCMR
##   summary.PMCMR PMCMR
```

```r
library(tidyverse)
```

```
## Found more than one class "atomicVector" in cache; using the first, from namespace 'Matrix'
```

```
## Also defined by 'Rmpfr'
```

```
## Found more than one class "atomicVector" in cache; using the first, from namespace 'Matrix'
```

```
## Also defined by 'Rmpfr'
```

```
## Found more than one class "atomicVector" in cache; using the first, from namespace 'Matrix'
```

```
## Also defined by 'Rmpfr'
```

```
## Found more than one class "atomicVector" in cache; using the first, from namespace 'Matrix'
```

```
## Also defined by 'Rmpfr'
```

```
## Found more than one class "atomicVector" in cache; using the first, from namespace 'Matrix'
```

```
## Also defined by 'Rmpfr'
```

```
## Found more than one class "atomicVector" in cache; using the first, from namespace 'Matrix'
```

```
## Also defined by 'Rmpfr'
```

```
## Found more than one class "atomicVector" in cache; using the first, from namespace 'Matrix'
```

```
## Also defined by 'Rmpfr'
```

```
## Found more than one class "atomicVector" in cache; using the first, from namespace 'Matrix'
```

```
## Also defined by 'Rmpfr'
```

```
## Found more than one class "atomicVector" in cache; using the first, from namespace 'Matrix'
```

```
## Also defined by 'Rmpfr'

## Found more than one class "atomicVector" in cache; using the first, from namespace 'Matrix'

## Also defined by 'Rmpfr'

## Found more than one class "atomicVector" in cache; using the first, from namespace 'Matrix'

## Also defined by 'Rmpfr'

## Found more than one class "atomicVector" in cache; using the first, from namespace 'Matrix'

## Also defined by 'Rmpfr'

## -- Attaching packages ----------------------------------------------- tidyverse 1.3.0 --

## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
## v purrr   0.3.4

## -- Conflicts -------------------------------------------------- tidyverse_conflicts() --
## x data.table::between() masks dplyr::between()
## x tidyr::expand()       masks Matrix::expand()
## x dplyr::filter()       masks stats::filter()
## x data.table::first()   masks dplyr::first()
## x dplyr::lag()          masks stats::lag()
## x data.table::last()    masks dplyr::last()
## x purrr::lift()         masks caret::lift()
## x tidyr::pack()         masks Matrix::pack()
## x dplyr::select()       masks MASS::select()
## x purrr::transpose()    masks data.table::transpose()
## x tidyr::unpack()       masks Matrix::unpack()
```

```r
library(rstatix)
```

```
##
## Attaching package: 'rstatix'

## The following object is masked from 'package:MASS':
##
##     select

## The following object is masked from 'package:stats':
##
##     filter
```

```r
library(Boruta)
library(rmarkdown)
```

```
data = read.csv("R_patients_scaled_age_4groups.csv", header=TRUE)
```

Select relevant columns with predictors, age, ethnicity, smoking

```
x_col = c('Diabetes','Age','Sex','Ethnicity','Smoking',
          'G2','G3','G4','G5','G6','G7','G8','G9','G10','G11','G12','G13','G14','G15','G16','G17','G18',
data2 <- data[x_col]
```

resetting levels for Smoking to have Never as baseline

```
data2 = subset(data2, Smoking!="")
data2$outcome = as.factor(data2$Diabetes)
data2<-data2[,-c(1)]
data2$Smoking <- factor( data2$Smoking , ordered = FALSE )
data2$Smoking <- relevel(data2$Smoking, ref = "Never")
data2$Age <- factor( data2$Age , ordered = FALSE )
data2$Age <- relevel(data2$Age, ref = "Age_65below")
```

resetting levels for Smoking to have Never as baseline and removing empty cells

Correlation between predictors

```
library(corrr)
library(dplyr)
library(tidyr)
pred = data[c('G2','G3','G4','G5','G6','G7','G8','G9','G10','G11','G12','G13','G14','G15','G16','G17','

corr_list_high = cor(pred) %>%
  as.data.frame() %>%
  mutate(var1 = rownames(.)) %>%
  gather(var2, value, -var1) %>%
  arrange(desc(value)) %>%
  group_by(value) %>%
  filter(row_number()==1)
corr_list_high = corr_list_high[-1,]
corr_list_high = corr_list_high[corr_list_high$value > 0.85,]
corr_list_high
```

```
## # A tibble: 35 x 3
## # Groups:   value [35]
##    var1  var2  value
##    <chr> <chr> <dbl>
##  1 G28   G26   0.949
##  2 G15   G9    0.943
##  3 G26   G23   0.926
##  4 G20   G15   0.922
##  5 G33   G31   0.921
##  6 G31   G25   0.908
##  7 G6    G5    0.905
```

```
## 8 G36   G34   0.902
## 9 G17   G12   0.902
## 10 G36   G35   0.897
## # ... with 25 more rows
```

Plotting correlation matrix

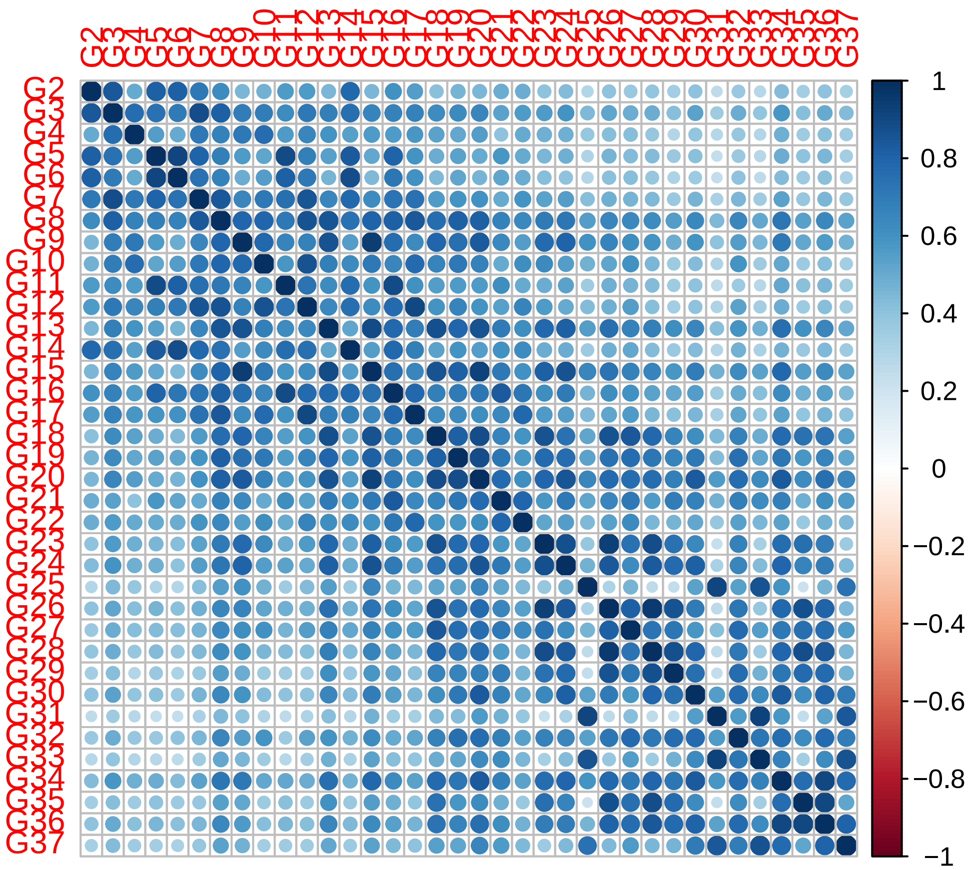
```r
corr_p <- round(cor(pred),2)
corr_p
```

```
##       G2   G3   G4   G5   G6   G7   G8   G9  G10  G11  G12  G13  G14  G15  G16
## G2  1.00 0.84 0.50 0.82 0.82 0.71 0.62 0.45 0.47 0.57 0.56 0.45 0.78 0.45 0.60
## G3  0.84 1.00 0.76 0.74 0.70 0.88 0.81 0.69 0.69 0.62 0.71 0.68 0.75 0.66 0.67
## G4  0.50 0.76 1.00 0.55 0.50 0.71 0.67 0.71 0.76 0.56 0.65 0.59 0.54 0.57 0.56
## G5  0.82 0.74 0.55 1.00 0.91 0.80 0.68 0.56 0.52 0.89 0.68 0.54 0.83 0.51 0.80
## G6  0.82 0.70 0.50 0.91 1.00 0.74 0.67 0.49 0.55 0.82 0.71 0.46 0.88 0.44 0.72
## G7  0.71 0.88 0.71 0.80 0.74 1.00 0.84 0.65 0.71 0.75 0.85 0.65 0.78 0.62 0.73
## G8  0.62 0.81 0.67 0.68 0.67 0.84 1.00 0.79 0.79 0.71 0.86 0.85 0.74 0.80 0.82
## G9  0.45 0.69 0.71 0.56 0.49 0.65 0.79 1.00 0.78 0.66 0.67 0.86 0.55 0.94 0.76
## G10 0.47 0.69 0.76 0.52 0.55 0.71 0.79 0.78 1.00 0.58 0.86 0.68 0.63 0.71 0.65
## G11 0.57 0.62 0.56 0.89 0.82 0.75 0.71 0.66 0.58 1.00 0.73 0.62 0.76 0.59 0.89
## G12 0.56 0.71 0.65 0.68 0.71 0.85 0.86 0.67 0.86 0.73 1.00 0.64 0.75 0.62 0.77
## G13 0.45 0.68 0.59 0.54 0.46 0.65 0.85 0.86 0.68 0.62 0.64 1.00 0.52 0.89 0.78
## G14 0.78 0.75 0.54 0.83 0.88 0.78 0.74 0.55 0.63 0.76 0.75 0.52 1.00 0.55 0.78
## G15 0.45 0.66 0.57 0.51 0.44 0.62 0.80 0.94 0.71 0.59 0.62 0.89 0.55 1.00 0.75
## G16 0.60 0.67 0.56 0.80 0.72 0.73 0.82 0.76 0.65 0.89 0.77 0.78 0.78 0.75 1.00
## G17 0.55 0.67 0.58 0.59 0.60 0.74 0.84 0.63 0.77 0.60 0.90 0.69 0.68 0.65 0.78
## G18 0.41 0.62 0.53 0.49 0.44 0.57 0.76 0.79 0.66 0.55 0.59 0.87 0.53 0.86 0.68
## G19 0.46 0.63 0.51 0.53 0.52 0.59 0.81 0.75 0.71 0.56 0.66 0.79 0.59 0.81 0.70
## G20 0.45 0.65 0.55 0.50 0.46 0.60 0.81 0.83 0.67 0.56 0.59 0.86 0.55 0.92 0.72
## G21 0.49 0.53 0.40 0.58 0.52 0.50 0.67 0.64 0.50 0.61 0.54 0.70 0.60 0.71 0.83
## G22 0.49 0.56 0.50 0.50 0.49 0.59 0.64 0.55 0.61 0.50 0.66 0.61 0.62 0.60 0.72
## G23 0.40 0.56 0.48 0.45 0.42 0.53 0.70 0.77 0.63 0.49 0.56 0.78 0.49 0.81 0.61
## G24 0.43 0.59 0.48 0.48 0.40 0.55 0.72 0.80 0.55 0.53 0.50 0.81 0.49 0.86 0.70
## G25 0.29 0.44 0.38 0.30 0.29 0.42 0.55 0.60 0.47 0.35 0.43 0.55 0.36 0.65 0.46
## G26 0.40 0.52 0.42 0.46 0.41 0.48 0.65 0.66 0.52 0.48 0.48 0.75 0.48 0.73 0.61
## G27 0.37 0.49 0.42 0.43 0.42 0.46 0.64 0.61 0.60 0.46 0.55 0.67 0.51 0.67 0.60
## G28 0.39 0.49 0.38 0.43 0.38 0.44 0.62 0.59 0.45 0.43 0.42 0.68 0.43 0.66 0.53
## G29 0.34 0.42 0.29 0.37 0.31 0.37 0.55 0.49 0.36 0.35 0.34 0.61 0.37 0.58 0.51
## G30 0.40 0.53 0.39 0.41 0.36 0.46 0.64 0.59 0.43 0.40 0.40 0.65 0.43 0.69 0.55
## G31 0.25 0.35 0.28 0.24 0.24 0.32 0.44 0.40 0.30 0.26 0.30 0.42 0.29 0.47 0.35
## G32 0.37 0.49 0.38 0.37 0.40 0.45 0.65 0.55 0.59 0.36 0.53 0.59 0.47 0.62 0.50
## G33 0.28 0.39 0.29 0.28 0.26 0.35 0.51 0.45 0.35 0.28 0.34 0.48 0.32 0.53 0.42
## G34 0.43 0.58 0.48 0.49 0.43 0.53 0.72 0.71 0.51 0.51 0.49 0.75 0.47 0.78 0.63
## G35 0.34 0.42 0.35 0.40 0.36 0.38 0.54 0.51 0.36 0.41 0.36 0.60 0.37 0.55 0.48
## G36 0.40 0.50 0.41 0.45 0.41 0.45 0.64 0.57 0.42 0.45 0.42 0.65 0.44 0.63 0.54
## G37 0.33 0.43 0.35 0.34 0.32 0.38 0.53 0.47 0.34 0.35 0.35 0.51 0.36 0.53 0.44
##      G17  G18  G19  G20  G21  G22  G23  G24  G25  G26  G27  G28  G29  G30  G31
## G2  0.55 0.41 0.46 0.45 0.49 0.49 0.40 0.43 0.29 0.40 0.37 0.39 0.34 0.40 0.25
## G3  0.67 0.62 0.63 0.65 0.53 0.56 0.56 0.59 0.44 0.52 0.49 0.49 0.42 0.53 0.35
## G4  0.58 0.53 0.51 0.55 0.40 0.50 0.48 0.48 0.38 0.42 0.42 0.38 0.29 0.39 0.28
## G5  0.59 0.49 0.53 0.50 0.58 0.50 0.45 0.48 0.30 0.46 0.43 0.43 0.37 0.41 0.24
## G6  0.60 0.44 0.52 0.46 0.52 0.49 0.42 0.40 0.29 0.41 0.42 0.38 0.31 0.36 0.24
```

```
## G7  0.74 0.57 0.59 0.60 0.50 0.59 0.53 0.55 0.42 0.48 0.46 0.44 0.37 0.46 0.32
## G8  0.84 0.76 0.81 0.81 0.67 0.64 0.70 0.72 0.55 0.65 0.64 0.62 0.55 0.64 0.44
## G9  0.63 0.79 0.75 0.83 0.64 0.55 0.77 0.80 0.60 0.66 0.61 0.59 0.49 0.59 0.40
## G10 0.77 0.66 0.71 0.67 0.50 0.61 0.63 0.55 0.47 0.52 0.60 0.45 0.36 0.43 0.30
## G11 0.60 0.55 0.56 0.56 0.61 0.50 0.49 0.53 0.35 0.48 0.46 0.43 0.35 0.40 0.26
## G12 0.90 0.59 0.66 0.59 0.54 0.66 0.56 0.50 0.43 0.48 0.55 0.42 0.34 0.40 0.30
## G13 0.69 0.87 0.79 0.86 0.70 0.61 0.78 0.81 0.55 0.75 0.67 0.68 0.61 0.65 0.42
## G14 0.68 0.53 0.59 0.55 0.60 0.62 0.49 0.49 0.36 0.48 0.51 0.43 0.37 0.43 0.29
## G15 0.65 0.86 0.81 0.92 0.71 0.60 0.81 0.86 0.65 0.73 0.67 0.66 0.58 0.69 0.47
## G16 0.78 0.68 0.70 0.72 0.83 0.72 0.61 0.70 0.46 0.61 0.60 0.53 0.51 0.55 0.35
## G17 1.00 0.62 0.63 0.61 0.64 0.78 0.56 0.55 0.44 0.52 0.57 0.45 0.41 0.45 0.33
## G18 0.62 1.00 0.82 0.88 0.66 0.59 0.86 0.74 0.52 0.86 0.84 0.78 0.66 0.61 0.44
## G19 0.63 0.82 1.00 0.88 0.72 0.58 0.77 0.76 0.53 0.74 0.76 0.72 0.66 0.71 0.43
## G20 0.61 0.88 0.88 1.00 0.77 0.61 0.79 0.85 0.65 0.77 0.76 0.76 0.68 0.83 0.57
## G21 0.64 0.66 0.72 0.77 1.00 0.79 0.58 0.71 0.52 0.65 0.71 0.56 0.69 0.67 0.47
## G22 0.78 0.59 0.58 0.61 0.79 1.00 0.52 0.55 0.44 0.54 0.63 0.45 0.46 0.51 0.38
## G23 0.56 0.86 0.77 0.79 0.58 0.52 1.00 0.87 0.38 0.93 0.74 0.88 0.74 0.64 0.23
## G24 0.55 0.74 0.76 0.85 0.71 0.55 0.87 1.00 0.47 0.84 0.62 0.83 0.77 0.82 0.32
## G25 0.44 0.52 0.53 0.65 0.52 0.44 0.38 0.47 1.00 0.31 0.46 0.25 0.24 0.53 0.91
## G26 0.52 0.86 0.74 0.77 0.65 0.54 0.93 0.84 0.31 1.00 0.81 0.95 0.86 0.70 0.26
## G27 0.57 0.84 0.76 0.76 0.71 0.63 0.74 0.62 0.46 0.81 1.00 0.73 0.72 0.58 0.42
## G28 0.45 0.78 0.72 0.76 0.56 0.45 0.88 0.83 0.25 0.95 0.73 1.00 0.87 0.79 0.25
## G29 0.41 0.66 0.66 0.68 0.69 0.46 0.74 0.77 0.24 0.86 0.72 0.87 1.00 0.75 0.24
## G30 0.45 0.61 0.71 0.83 0.67 0.51 0.64 0.82 0.53 0.70 0.58 0.79 0.75 1.00 0.55
## G31 0.33 0.44 0.43 0.57 0.47 0.38 0.23 0.32 0.91 0.26 0.42 0.25 0.24 0.55 1.00
## G32 0.52 0.67 0.75 0.76 0.68 0.54 0.67 0.65 0.54 0.72 0.77 0.71 0.76 0.77 0.56
## G33 0.39 0.49 0.52 0.63 0.62 0.45 0.33 0.43 0.86 0.38 0.55 0.35 0.47 0.63 0.92
## G34 0.53 0.77 0.72 0.83 0.68 0.53 0.76 0.79 0.59 0.78 0.71 0.79 0.72 0.83 0.58
## G35 0.39 0.74 0.58 0.62 0.48 0.37 0.75 0.66 0.21 0.87 0.75 0.88 0.77 0.62 0.24
## G36 0.46 0.73 0.66 0.75 0.61 0.47 0.69 0.69 0.46 0.80 0.76 0.84 0.77 0.80 0.53
## G37 0.40 0.54 0.52 0.65 0.56 0.44 0.36 0.44 0.74 0.44 0.57 0.45 0.46 0.70 0.84
##      G32  G33  G34  G35  G36  G37
## G2  0.37 0.28 0.43 0.34 0.40 0.33
## G3  0.49 0.39 0.58 0.42 0.50 0.43
## G4  0.38 0.29 0.48 0.35 0.41 0.35
## G5  0.37 0.28 0.49 0.40 0.45 0.34
## G6  0.40 0.26 0.43 0.36 0.41 0.32
## G7  0.45 0.35 0.53 0.38 0.45 0.38
## G8  0.65 0.51 0.72 0.54 0.64 0.53
## G9  0.55 0.45 0.71 0.51 0.57 0.47
## G10 0.59 0.35 0.51 0.36 0.42 0.34
## G11 0.36 0.28 0.51 0.41 0.45 0.35
## G12 0.53 0.34 0.49 0.36 0.42 0.35
## G13 0.59 0.48 0.75 0.60 0.65 0.51
## G14 0.47 0.32 0.47 0.37 0.44 0.36
## G15 0.62 0.53 0.78 0.55 0.63 0.53
## G16 0.50 0.42 0.63 0.48 0.54 0.44
## G17 0.52 0.39 0.53 0.39 0.46 0.40
## G18 0.67 0.49 0.77 0.74 0.73 0.54
## G19 0.75 0.52 0.72 0.58 0.66 0.52
## G20 0.76 0.63 0.83 0.62 0.75 0.65
## G21 0.68 0.62 0.68 0.48 0.61 0.56
## G22 0.54 0.45 0.53 0.37 0.47 0.44
## G23 0.67 0.33 0.76 0.75 0.69 0.36
```

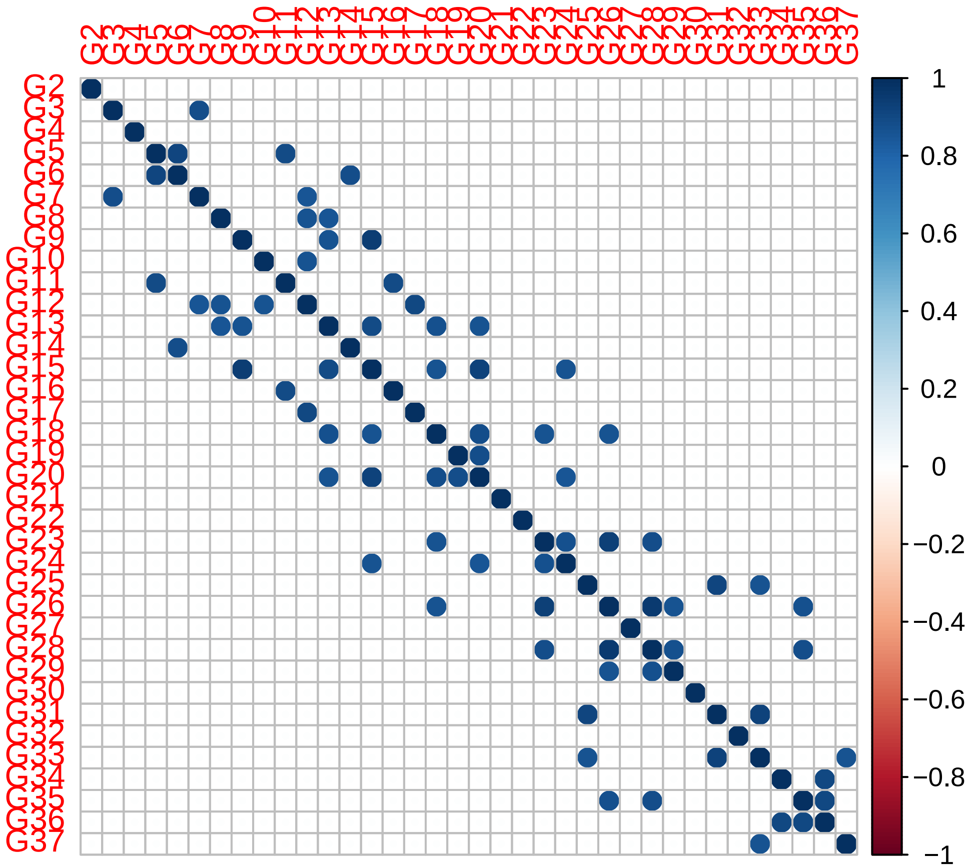
```
## G24 0.65 0.43 0.79 0.66 0.69 0.44
## G25 0.54 0.86 0.59 0.21 0.46 0.74
## G26 0.72 0.38 0.78 0.87 0.80 0.44
## G27 0.77 0.55 0.71 0.75 0.76 0.57
## G28 0.71 0.35 0.79 0.88 0.84 0.45
## G29 0.76 0.47 0.72 0.77 0.77 0.46
## G30 0.77 0.63 0.83 0.62 0.80 0.70
## G31 0.56 0.92 0.58 0.24 0.53 0.84
## G32 1.00 0.72 0.76 0.62 0.77 0.69
## G33 0.72 1.00 0.67 0.34 0.62 0.86
## G34 0.76 0.67 1.00 0.76 0.90 0.77
## G35 0.62 0.34 0.76 1.00 0.90 0.52
## G36 0.77 0.62 0.90 0.90 1.00 0.80
## G37 0.69 0.86 0.77 0.52 0.80 1.00
```

```
corrplot(corr_p)
```



Plot plasma predictors with Pearson correlation coefficient > 0.85

```
f <- function(x) if_else(x < 0.85, 0, x)
corr_p2 = apply(corr_p, c(1,2), f)
corrplot(corr_p2)
```



FEATURE SELECTION converting the data to matrix

```
data3 <- model.matrix(outcome ~ ., data=data2)
data3<-data3[,-c(1)]
diab_y <- as.vector(data2['outcome'])
data5 = cbind(data3, diab_y)
write.csv(data5, file = "diabetes_dp.csv",row.names=FALSE)
```

Splitting the results into feature selection, training and test sets

```
set.seed(123)
partition1 <- createDataPartition(data5$outcome, p = 0.7, list = FALSE)
train <- data5[partition1, ]
test_set <- data5[-partition1, ]
partition2 <- createDataPartition(train$outcome, p = 0.7, list = FALSE)
train_set <- train[partition2, ]
feature_selection_set <- train[-partition2, ]
```

saving datasets

```
write.csv(feature_selection_set, file = "1_feature_selection_set.csv",row.names=FALSE)
write.csv(train_set, file = "1_train_set.csv",row.names=FALSE)
write.csv(test_set, file = "1_test_set.csv",row.names=FALSE)
```

set crossvalidation

```r
tr = trainControl(method="cv", number=5, allowParallel = TRUE)
```

Feature selection -StepAIC To make the model simpler but still good and find important predictors I performed Stepwise feature selection

```r
f_all = as.formula(
  paste("outcome", paste('.'), sep = " ~ ")
)
GLM_stepAIC = train(f_all, data = feature_selection_set, method = "glmStepAIC", trControl = tr, family =
summary(GLM_stepAIC)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8813  -0.7731  -0.4443   0.6698   2.7362
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)               -2.2556     0.4389  -5.140 2.75e-07 ***
## Sexmale                    0.8036     0.4211   1.908 0.056336 .
## EthnicityEthnic_group_2    0.8588     0.3637   2.361 0.018209 *
## EthnicityEthnic_group_3    1.6118     0.5337   3.020 0.002525 **
## G3                         2.0763     0.4399   4.720 2.35e-06 ***
## G5                        -2.1014     0.5884  -3.571 0.000355 ***
## G9                        -2.0858     0.6831  -3.053 0.002264 **
## G11                        1.4246     0.5429   2.624 0.008683 **
## G14                       -0.6956     0.3735  -1.862 0.062575 .
## G15                        2.4791     0.7783   3.185 0.001446 **
## G18                       -2.0511     0.4321  -4.747 2.07e-06 ***
## G19                       -0.7071     0.3670  -1.927 0.054001 .
## G21                        0.6355     0.2771   2.293 0.021828 *
## G28                        1.4947     0.3311   4.515 6.33e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 323.38  on 258  degrees of freedom
## Residual deviance: 247.27  on 245  degrees of freedom
## AIC: 275.27
##
## Number of Fisher Scoring iterations: 5
```

So the features selected by stepAIC are: Sexmale + EthnicityEthnic_group_2 + EthnicityEthnic_group_3 + G3 + G5 + G9 + G11 + G14 + G15 + G18 + G19 + G21 + G28. Let's create a formula for them

```r
var_stepAIC = c("Sexmale" , "EthnicityEthnic_group_2" , "EthnicityEthnic_group_3" ,"G3" , "G5" , "G9" ,
f_stepAIC = as.formula(
  paste("outcome", paste(var_stepAIC, collapse = " + "), sep = " ~ ")
)
```

Another method for feature selected based on random forest is called boruta

```
boruta  <- Boruta(f_all,
                  data = feature_selection_set, doTrace=0, maxRuns = 500)
```

Get NonRejected features selected by boruta

```
getNonRejectedFormula(boruta)
```

```
## outcome ~ EthnicityEthnic_group_2 + G2 + G3 + G9 + G11 + G18 +
##     G23 + G24 + G28 + G29 + G30 + G34 + G35 + G36
## <environment: 0x0000000023650448>
```

Make formula for boruta

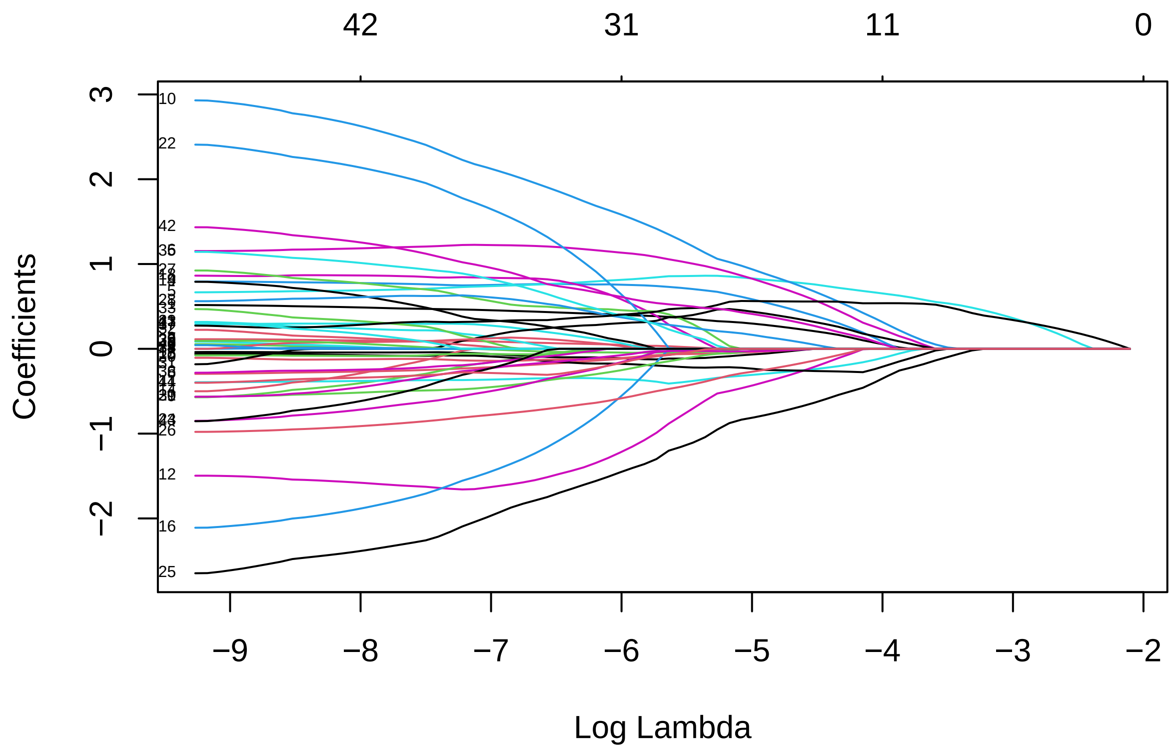
```
var_boruta = c("EthnicityEthnic_group_2","G2","G3","G9","G11","G18",
    "G23","G24",'G28',"G29","G30","G34","G35","G36")
f_boruta = as.formula(
  paste("outcome", paste(var_boruta, collapse = " + "), sep = " ~ ")
)
```

Feature selection using lasso

```
lasso = train(f_all, data = feature_selection_set,
              method = "glmnet",
              tuneGrid = expand.grid(alpha = 1,
                                     lambda = seq(0.001, 0.1, length = 10)),
              trControl = tr, family = "binomial")
```

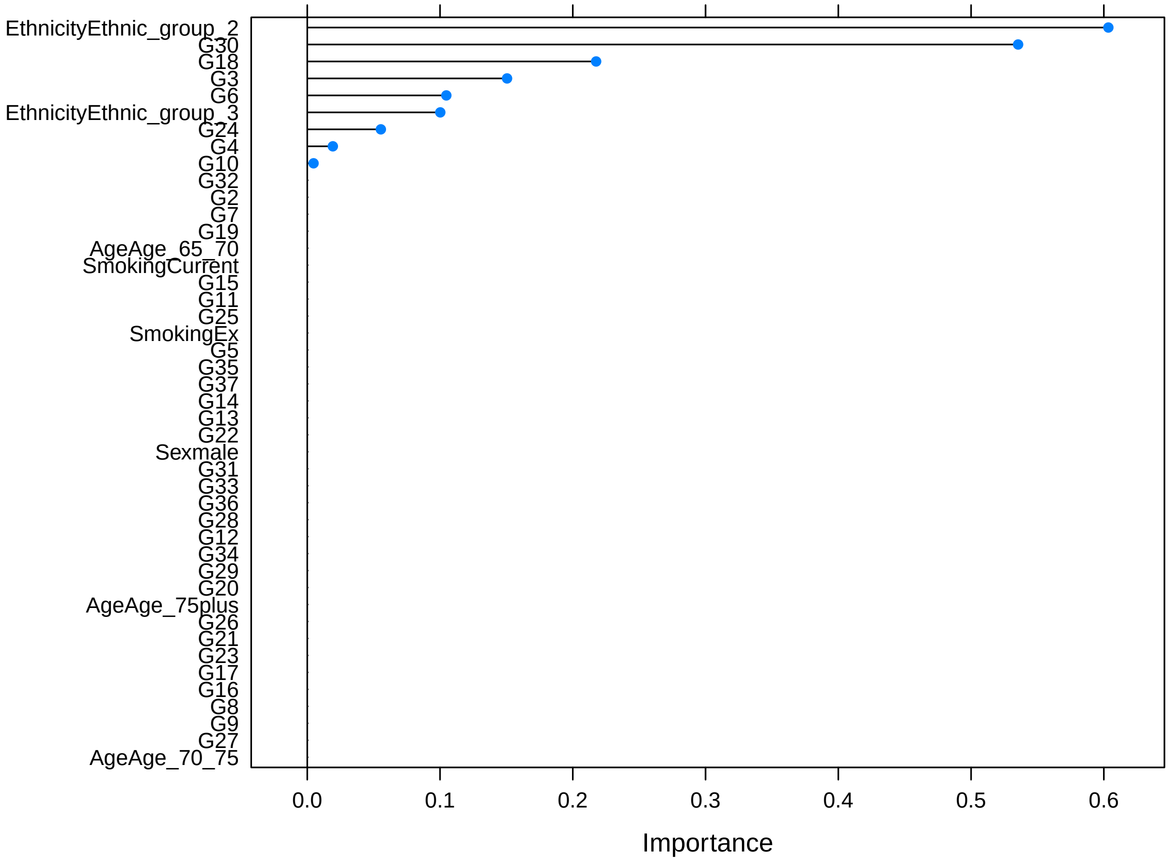
This plot shows which predictors remain while increasing lambda

```
plot(lasso$finalModel, xvar = "lambda", label = T)
```



plot Variable importance

```
plot(varImp(lasso, scale = F))
```



lasso significantly decreased the number of predictors: GP2, GP18, GP31. So here are the formula for lasso selected predictors

```
var_lasso = c("EthnicityEthnic_group_2","G30","G18","G3","G6","EthnicityEthnic_group_3","G24",'G4',"G10
f_lasso = as.formula(paste("outcome", paste(var_lasso, collapse = " + "), sep = " ~ "))
```

Training and testing various models on selected features

First part of the model name is the method of ML (GLM, StepAIC, EN, ridge, lasso, RF, SVM) and second part is features set selected previously (all, stepAIC, boruta, lasso)

```
GLM_all = train(f_all, data = train_set, method = "glm", trControl = tr, family = "binomial")
GLM_stepAIC = train(f_stepAIC, data = train_set, method = "glm", trControl = tr, family = "binomial")
GLM_boruta = train(f_boruta, data = train_set, method = "glm", trControl = tr, family = "binomial")
GLM_lasso = train(f_lasso, data = train_set, method = "glm", trControl = tr, family = "binomial")

GLM_step_all = train(f_all, data = train_set, method = "glmStepAIC", trControl = tr, family = "binomial
GLM_step_stepAIC = train(f_stepAIC, data = train_set, method = "glmStepAIC", trControl = tr, family = "
GLM_step_boruta = train(f_boruta, data = train_set, method = "glmStepAIC", trControl = tr, family = "bi
GLM_step_lasso = train(f_lasso, data = train_set, method = "glmStepAIC", trControl = tr, family = "bino
```

```
EN_all = train(f_all, data = train_set, method = "glmnet",
           tuneGrid = expand.grid(alpha = seq(0,1,length = 10),
                                  lambda = seq(0.0001, 0.001, length = 10)),
           trControl = tr, family = "binomial")
EN_stepAIC = train(f_stepAIC, data = train_set, method = "glmnet",
               tuneGrid = expand.grid(alpha = seq(0,1,length = 10),
                                      lambda = seq(0.0001, 0.001, length = 10)),
               trControl = tr, family = "binomial")
EN_boruta = train(f_boruta, data = train_set, method = "glmnet",
               tuneGrid = expand.grid(alpha = seq(0,1,length = 10),
                                      lambda = seq(0.0001, 0.001, length = 10)),
               trControl = tr, family = "binomial")
EN_lasso = train(f_lasso, data = train_set, method = "glmnet",
               tuneGrid = expand.grid(alpha = seq(0,1,length = 10),
                                      lambda = seq(0.0001, 0.001, length = 10)),
               trControl = tr, family = "binomial")

ridge_all = train(f_all, data = train_set, method = "glmnet",
              tuneGrid = expand.grid(alpha = 0, lambda = seq(0.001, 0.1, length = 10)),
              trControl = tr, family = "binomial")
ridge_stepAIC = train(f_stepAIC, data = train_set, method = "glmnet",
                  tuneGrid = expand.grid(alpha = 0, lambda = seq(0.001, 0.1, length = 10)),
                  trControl = tr, family = "binomial")
ridge_boruta = train(f_boruta, data = train_set, method = "glmnet",
                  tuneGrid = expand.grid(alpha = 0, lambda = seq(0.001, 0.1, length = 10)),
                  trControl = tr, family = "binomial")
ridge_lasso = train(f_lasso, data = train_set, method = "glmnet",
                  tuneGrid = expand.grid(alpha = 0, lambda = seq(0.001, 0.1, length = 10)),
                  trControl = tr, family = "binomial")

lasso_all = train(f_all, data = train_set, method = "glmnet",
                  tuneGrid = expand.grid(alpha = 1, lambda = seq(0.001, 0.1, length = 10)),
                  trControl = tr, family = "binomial")
lasso_stepAIC = train(f_stepAIC, data = train_set, method = "glmnet",
                      tuneGrid = expand.grid(alpha = 1, lambda = seq(0.001, 0.1, length = 10)),
                      trControl = tr, family = "binomial")
lasso_boruta = train(f_boruta, data = train_set, method = "glmnet",
                     tuneGrid = expand.grid(alpha = 1, lambda = seq(0.001, 0.1, length = 10)),
                     trControl = tr, family = "binomial")
lasso_lasso = train(f_lasso, data = train_set, method = "glmnet",
                    tuneGrid = expand.grid(alpha = 1, lambda = seq(0.001, 0.1, length = 10)),
                    trControl = tr, family = "binomial")

RF_all <- train(f_all, data=train_set, method="rf", trControl=tr, verbose=FALSE)
RF_stepAIC <- train(f_stepAIC, data=train_set, method="rf", trControl=tr, verbose=FALSE)
RF_boruta <- train(f_boruta, data=train_set, method="rf", trControl=tr, verbose=FALSE)
RF_lasso <- train(f_lasso, data=train_set, method="rf", trControl=tr, verbose=FALSE)


SVM_all = svm(f_all, data = train_set, kernel = "linear", cost = 10, scale = FALSE)
SVM_stepAIC = svm(f_stepAIC, data = train_set, kernel = "linear", cost = 10, scale = FALSE)
SVM_boruta = svm(f_boruta, data = train_set, kernel = "linear", cost = 10, scale = FALSE)
SVM_lasso = svm(f_lasso, data = train_set, kernel = "linear", cost = 10, scale = FALSE)
```

Create a list of model names - models

```
f_list = c("_all","_stepAIC", "_boruta", "_lasso")
models1 <- paste0("GLM", f_list)
models2 <- paste0("GLM_step", f_list)
models3 <- paste0("EN", f_list)
models4 <- paste0("ridge", f_list)
models5 <- paste0("lasso", f_list)
models6 <- paste0("RF", f_list)
models7 <- paste0("SVM", f_list)

models = c(models1, models2, models3, models4, models5, models6, models7)
```

Create a list of models

```
modellist = list(GLM_all, GLM_stepAIC,GLM_boruta,GLM_lasso,
                 GLM_step_all,GLM_step_stepAIC,GLM_step_boruta,GLM_step_lasso,
                 EN_all,EN_stepAIC,EN_boruta,EN_lasso,
                 ridge_all,ridge_stepAIC,ridge_boruta,ridge_boruta,
                 lasso_all,lasso_stepAIC,lasso_boruta,lasso_lasso,
                 RF_all,RF_stepAIC,RF_boruta,RF_lasso,
                 SVM_all,SVM_stepAIC,SVM_boruta,SVM_lasso)
```

Create a dataframe with metrics: “Accuracy_train”,“Accuracy_test”, “F1”, “AUC” for all the models trained above

```
results = data.frame(matrix(NA, nrow = 0, ncol = 5))
colnames(results) <- c("Model","Accuracy_train","Accuracy_test", "F1", "AUC")
```

Add metrics for the models and features sets to the results table

```
for(i in 1:28){
  add_model<-data.frame(models[i],
                       confusionMatrix(data = predict(modellist[[i]], newdata=train_set), reference = t
                       confusionMatrix(data = predict(modellist[[i]], newdata=test_set), reference = te
                       confusionMatrix(data = predict(modellist[[i]], newdata=test_set), reference = te
                       auc(roc(test_set$outcome, factor(predict(modellist[[i]], newdata=test_set), orde
  )
  names(add_model)<-c("Model","Accuracy_train","Accuracy_test", "F1", "AUC")
  results <- rbind(results, add_model)
}
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

Save the results

```r
write.csv(results, file = "all_models_results.csv",row.names=FALSE)
```

```r
#results[,order(accuracy_test)]
results = as.data.frame(results)
results[order(-results$Accuracy_test),]
```

```
##                        Model Accuracy_train Accuracy_test        F1       AUC
## Accuracy             GLM_all      0.7651888     0.7513514 0.8315018 0.6665653
## Accuracy8             EN_all      0.7668309     0.7513514 0.8321168 0.6642681
## Accuracy24           SVM_all      0.7701149     0.7486486 0.8312160 0.6576974
## Accuracy4       GLM_step_all      0.7602627     0.7405405 0.8235294 0.6563630
## Accuracy13     ridge_stepAIC      0.7471264     0.7270270 0.8224956 0.6120232
## Accuracy25       SVM_stepAIC      0.7290640     0.7189189 0.8181818 0.5992027
## Accuracy26        SVM_boruta      0.7142857     0.7189189 0.8278146 0.5624472
## Accuracy12         ridge_all      0.7323481     0.7162162 0.8235294 0.5696598
## Accuracy1        GLM_stepAIC      0.7504105     0.7108108 0.8085868 0.6047600
## Accuracy17     lasso_stepAIC      0.7520525     0.7108108 0.8092692 0.6024628
## Accuracy9         EN_stepAIC      0.7520525     0.7081081 0.8071429 0.6004865
## Accuracy2         GLM_boruta      0.7274220     0.7054054 0.8104348 0.5801324
## Accuracy14      ridge_boruta      0.7241379     0.7054054 0.8180301 0.5525658
## Accuracy15       ridge_lasso      0.7241379     0.7054054 0.8180301 0.5525658
## Accuracy18      lasso_boruta      0.7224959     0.7054054 0.8174204 0.5548630
## Accuracy10         EN_boruta      0.7290640     0.7027027 0.8083624 0.5781561
## Accuracy19       lasso_lasso      0.7241379     0.7027027 0.8070175 0.5827506
## Accuracy11          EN_lasso      0.7241379     0.7000000 0.8076256 0.5715854
## Accuracy27         SVM_lasso      0.7389163     0.7000000 0.8115450 0.5578021
## Accuracy5   GLM_step_stepAIC      0.7520525     0.6972973 0.7992832 0.5902841
## Accuracy7     GLM_step_lasso      0.7241379     0.6972973 0.8028169 0.5787980
## Accuracy3          GLM_lasso      0.7257800     0.6945946 0.8000000 0.5791189
## Accuracy6    GLM_step_boruta      0.7290640     0.6945946 0.8021016 0.5722273
## Accuracy20            RF_all      1.0000000     0.6945946 0.8068376 0.5561468
## Accuracy16         lasso_all      0.7405583     0.6918919 0.8027682 0.5610621
## Accuracy22         RF_boruta      1.0000000     0.6891892 0.8013817 0.5567886
## Accuracy21        RF_stepAIC      1.0000000     0.6864865 0.7958115 0.5574305
## Accuracy23          RF_lasso      1.0000000     0.6702703 0.7925170 0.5288504
```

```r
results
```

```
##                        Model Accuracy_train Accuracy_test        F1       AUC
## Accuracy             GLM_all      0.7651888     0.7513514 0.8315018 0.6665653
## Accuracy1        GLM_stepAIC      0.7504105     0.7108108 0.8085868 0.6047600
## Accuracy2         GLM_boruta      0.7274220     0.7054054 0.8104348 0.5801324
## Accuracy3          GLM_lasso      0.7257800     0.6945946 0.8000000 0.5791189
## Accuracy4       GLM_step_all      0.7602627     0.7405405 0.8235294 0.6563630
## Accuracy5   GLM_step_stepAIC      0.7520525     0.6972973 0.7992832 0.5902841
## Accuracy6    GLM_step_boruta      0.7290640     0.6945946 0.8021016 0.5722273
## Accuracy7     GLM_step_lasso      0.7241379     0.6972973 0.8028169 0.5787980
## Accuracy8             EN_all      0.7668309     0.7513514 0.8321168 0.6642681
## Accuracy9         EN_stepAIC      0.7520525     0.7081081 0.8071429 0.6004865
## Accuracy10         EN_boruta      0.7290640     0.7027027 0.8083624 0.5781561
## Accuracy11          EN_lasso      0.7241379     0.7000000 0.8076256 0.5715854
## Accuracy12         ridge_all      0.7323481     0.7162162 0.8235294 0.5696598
```

```
## Accuracy13    ridge_stepAIC      0.7471264     0.7270270 0.8224956 0.6120232
## Accuracy14     ridge_boruta      0.7241379     0.7054054 0.8180301 0.5525658
## Accuracy15      ridge_lasso      0.7241379     0.7054054 0.8180301 0.5525658
## Accuracy16        lasso_all      0.7405583     0.6918919 0.8027682 0.5610621
## Accuracy17    lasso_stepAIC      0.7520525     0.7108108 0.8092692 0.6024628
## Accuracy18     lasso_boruta      0.7224959     0.7054054 0.8174204 0.5548630
## Accuracy19      lasso_lasso      0.7241379     0.7027027 0.8070175 0.5827506
## Accuracy20           RF_all      1.0000000     0.6945946 0.8068376 0.5561468
## Accuracy21       RF_stepAIC      1.0000000     0.6864865 0.7958115 0.5574305
## Accuracy22        RF_boruta      1.0000000     0.6891892 0.8013817 0.5567886
## Accuracy23         RF_lasso      1.0000000     0.6702703 0.7925170 0.5288504
## Accuracy24          SVM_all      0.7701149     0.7486486 0.8312160 0.6576974
## Accuracy25      SVM_stepAIC      0.7290640     0.7189189 0.8181818 0.5992027
## Accuracy26       SVM_boruta      0.7142857     0.7189189 0.8278146 0.5624472
## Accuracy27        SVM_lasso      0.7389163     0.7000000 0.8115450 0.5578021
```

We see that overfitting for most linear models and SVM is relatively low, while it is high for Random forest. The best performance is for models which include all the data, however StepAIC selected features which are twice smaller in number show only slightly lower permormance metrics

I also trained various deep neural networks (see the the model in 2_Diabetes_prediction/all_samples/dnn and results of different training on 2_Diabetes_prediction/all_samples/dnn/1_results_FS_train_test_split). I varied the amount of dense layers between 4 and 6, number of units between 2 and 1000 (1_dnn_layers_var.sh) and the best model showed accuracy on test - 0.732. Next, I tried adding various dropout layers, however it did not improve accuracy. Finally, I tried l1 and l2 regularisation, and l2 = 0.001 added to the penultimate layer improved test accuracy to 0.735. Not bad but it is only 5th place after GLM and SVM models used above.

Further analyses

```
data = subset(data, Smoking!="")
data$Diab_status  =""
data$Diab_status <- ifelse(data$Diabetes==1, "sick", "healthy")
```

Let's see the effects of predictors on the whole dataset

```
GLM = train(outcome ~ ., data = data5, method = "glm", trControl = tr, family = "binomial", trace=0)
summary(GLM)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1539  -0.7921  -0.4813   0.8946   2.7125
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)             -2.188118   0.258129  -8.477  < 2e-16 ***
## AgeAge_65_70             0.148999   0.196607   0.758  0.44854
```

```
## AgeAge_70_75              0.566843   0.205428   2.759  0.00579 **
## AgeAge_75plus             0.313220   0.225472   1.389  0.16478
## Sexmale                   0.246342   0.212609   1.159  0.24660
## EthnicityEthnic_group_2  1.062764   0.196044   5.421 5.92e-08 ***
## EthnicityEthnic_group_3  1.764305   0.272551   6.473 9.59e-11 ***
## SmokingCurrent            0.228823   0.319122   0.717  0.47335
## SmokingEx                 0.174677   0.170310   1.026  0.30506
## G2                       -0.160546   0.351378  -0.457  0.64774
## G3                        1.155973   0.366835   3.151  0.00163 **
## G4                       -0.089596   0.176281  -0.508  0.61127
## G5                       -0.759579   0.390803  -1.944  0.05194 .
## G6                       -0.045489   0.319912  -0.142  0.88693
## G7                       -0.636592   0.349579  -1.821  0.06860 .
## G8                        0.153080   0.318597   0.480  0.63088
## G9                       -0.626844   0.389213  -1.611  0.10728
## G10                      -0.037844   0.260536  -0.145  0.88451
## G11                       0.425975   0.418545   1.018  0.30880
## G12                       1.259827   0.464712   2.711  0.00671 **
## G13                       0.460433   0.262646   1.753  0.07959 .
## G14                       0.266309   0.243596   1.093  0.27429
## G15                       0.078739   0.481046   0.164  0.86998
## G16                      -1.200384   0.535985  -2.240  0.02512 *
## G17                      -0.837974   0.331330  -2.529  0.01143 *
## G18                      -0.945293   0.471489  -2.005  0.04497 *
## G19                      -0.920774   0.212620  -4.331 1.49e-05 ***
## G20                       1.114730   0.516808   2.157  0.03101 *
## G21                       0.966464   0.380912   2.537  0.01117 *
## G22                       0.003265   0.207379   0.016  0.98744
## G23                      -0.690332   0.412031  -1.675  0.09385 .
## G24                       1.285208   0.483211   2.660  0.00782 **
## G25                      -0.405227   0.361529  -1.121  0.26234
## G26                      -0.897445   0.520738  -1.723  0.08481 .
## G27                       0.215563   0.226797   0.950  0.34187
## G28                       1.140917   0.603384   1.891  0.05864 .
## G29                       0.010247   0.263928   0.039  0.96903
## G30                      -0.496116   0.463903  -1.069  0.28487
## G31                      -0.016533   0.411273  -0.040  0.96793
## G32                       0.035130   0.271509   0.129  0.89705
## G33                       0.260233   0.379704   0.685  0.49312
## G34                       0.416341   0.266449   1.563  0.11816
## G35                       0.292663   0.383963   0.762  0.44593
## G36                      -0.781653   0.573152  -1.364  0.17264
## G37                      -0.055971   0.294650  -0.190  0.84934
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1545.8  on 1237  degrees of freedom
## Residual deviance: 1250.7  on 1193  degrees of freedom
## AIC: 1340.7
##
## Number of Fisher Scoring iterations: 5
```

Summary shows effect size (estimate), their standard errors and p-values For example, G24 has the larges coefficient - 1.29, which means that odds ratio of having diabetes increases e^1.29 = 3.6 times for each unit of G24, standard error for G24 equals 0.48, and the effect is significant with p-value - 0.008. Other plasma predictors with the strongest effects are: G3 1.155973 G12 1.259827 G16 -1.200384 G17 -0.837974 G18 -0.945293
G19 -0.920774 G20 1.114730 G21 0.966464
Given that some predictors are highly correlated (i.e. G12 and G17 or G18 and G20), it is difficult to interpret their individual effects

Effect of age

Of 4 age groups, all 3 older groups have increased chances of diabetes, though only the group 70-75 effect is significant - belonging to this group incerases chances of diabetes e^0.56 = 1.75 times.

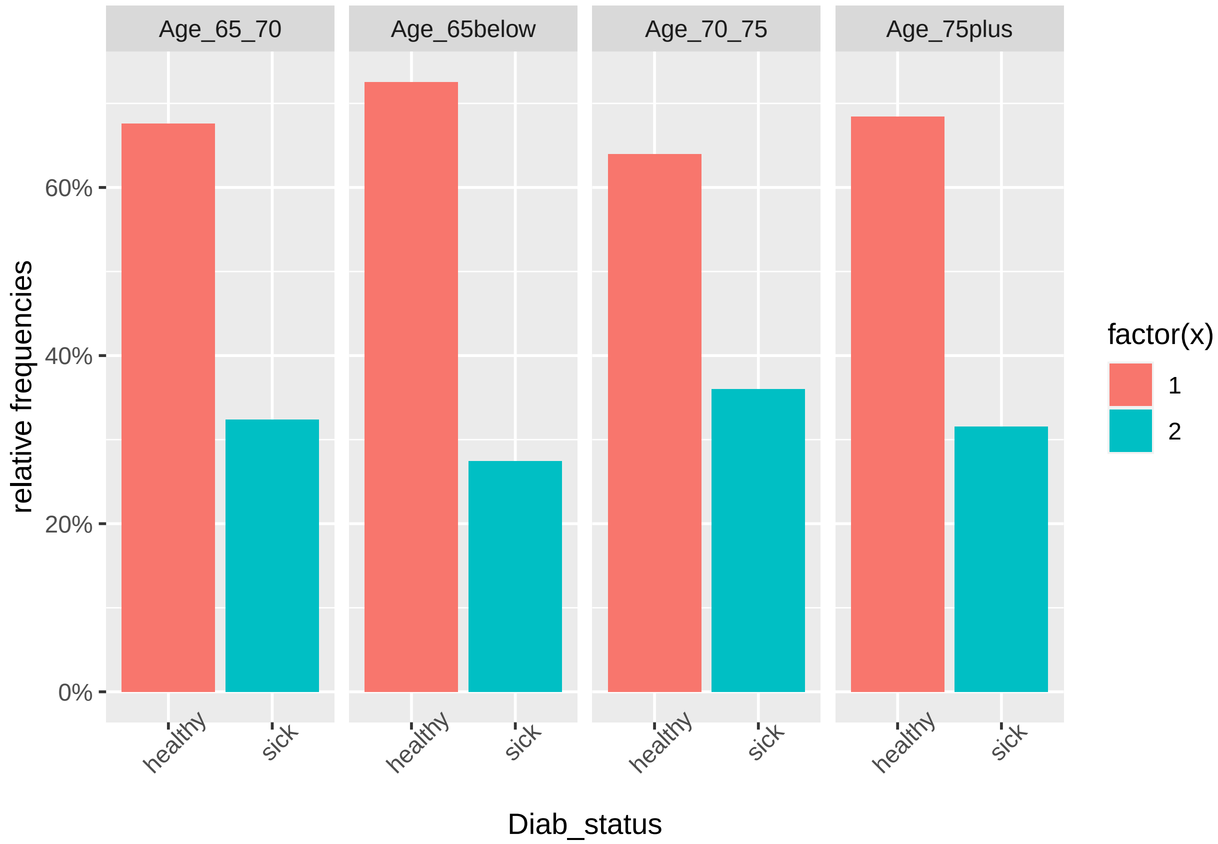
Nevetheless, the sample is relatively small and chi-square test does not show significant enough dependence between between age and diabetes status

```
chisq.test(table(data$Age, data$Diabetes))
```

```
##
##  Pearson's Chi-squared test
##
## data:  table(data$Age, data$Diabetes)
## X-squared = 5.6227, df = 3, p-value = 0.1315
```

The percentage of diabetes is the highest in 70-75 group

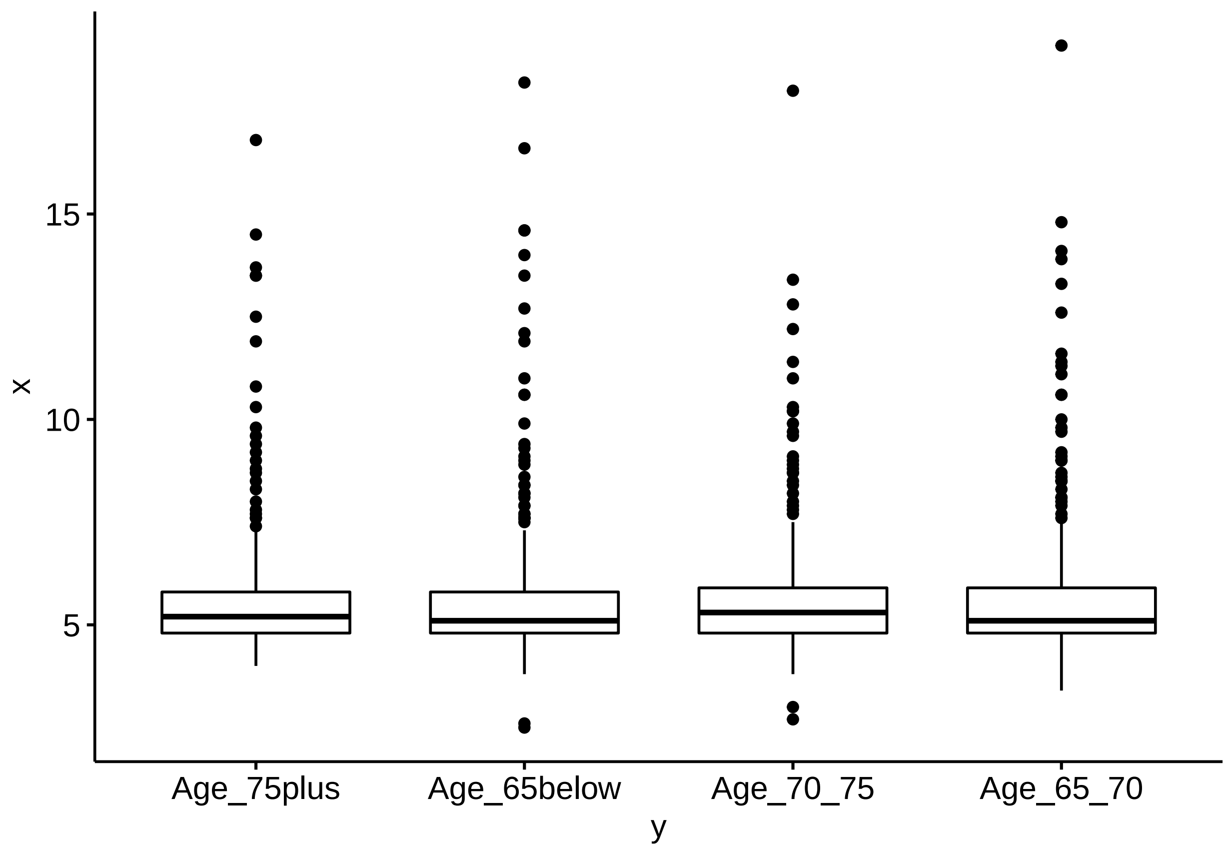
```
ggplot(data, aes(Diab_status, group = Age)) +
          geom_bar(aes(y = ..prop.., fill = factor(..x..)), stat="count") +
          scale_y_continuous(labels=scales::percent) +
          ylab("relative frequencies") +
          facet_grid(~Age) +
          theme(axis.text.x = element_text(angle = 45))
```



Glucose level difference however is not significant between age groups.

```
ggboxplot(data, x = "Age", y = "Glucose1",
          ylab = "x", xlab = "y")
```

```
## Warning: Removed 2 rows containing non-finite values (stat_boxplot).
```




```r
kruskal.test(Glucose1 ~ Age,data = data)
```

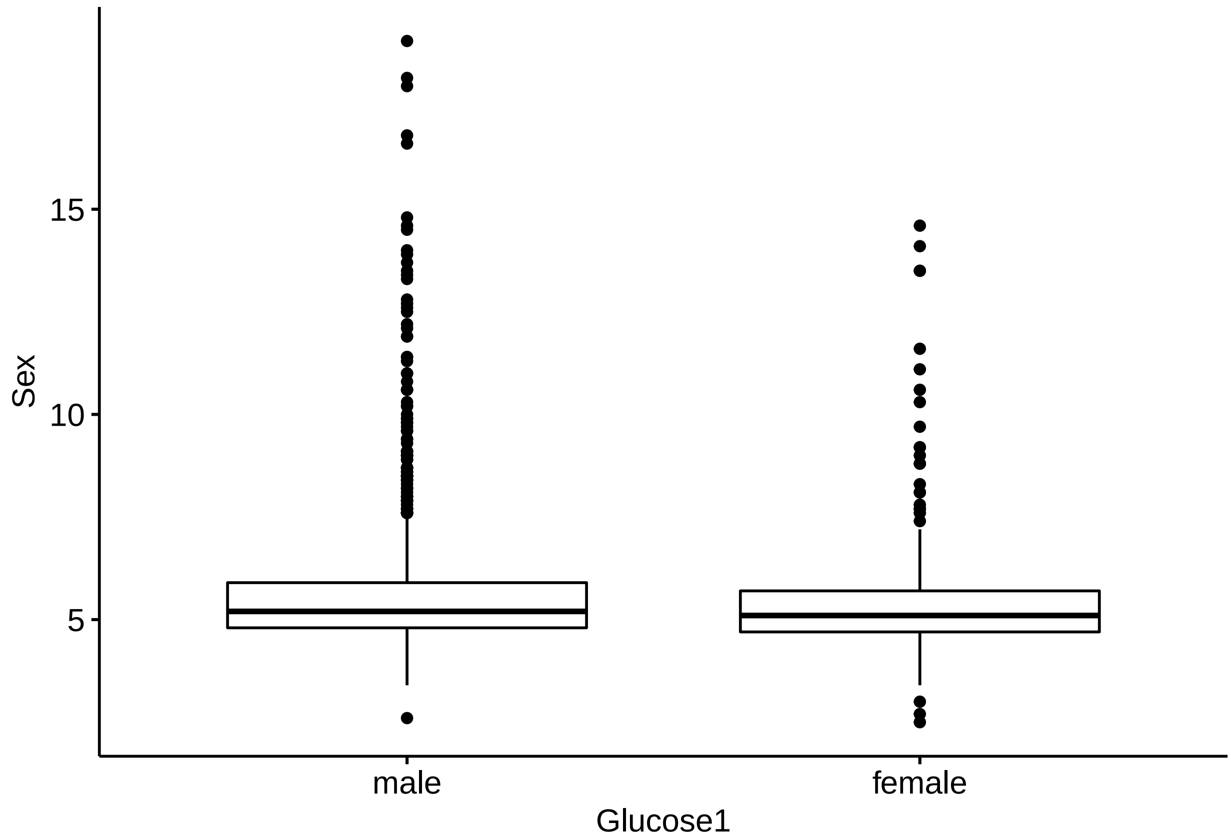
```
##
##  Kruskal-Wallis rank sum test
##
## data:  Glucose1 by Age
## Kruskal-Wallis chi-squared = 1.998, df = 3, p-value = 0.5728
```

Sex and diabetes

male sex increases the chances of diabetes in e^0.25 = 1.28 times, however it was not significant. This is probably due to the small sample since glucose level in males is significantly higher

```r
ggboxplot(data, x = "Sex", y = "Glucose1",
          ylab = "Sex", xlab = "Glucose1")
```

```
## Warning: Removed 2 rows containing non-finite values (stat_boxplot).
```




```r
t.test(Glucose1 ~ Sex,alternative = c("two.sided"), conf.level = 0.95, data = data)
```

```
##
##  Welch Two Sample t-test
##
## data:  Glucose1 by Sex
## t = -2.5286, df = 535.39, p-value = 0.01174
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.49149075 -0.06171947
## sample estimates:
## mean in group female   mean in group male
##             5.469338             5.745943
```

The effect of smoking

Current and Ex status of smoking slightly increase the chances of diabetes though the effect was not significant due to small sample size Glucose level difference is also not significant in smoking groups

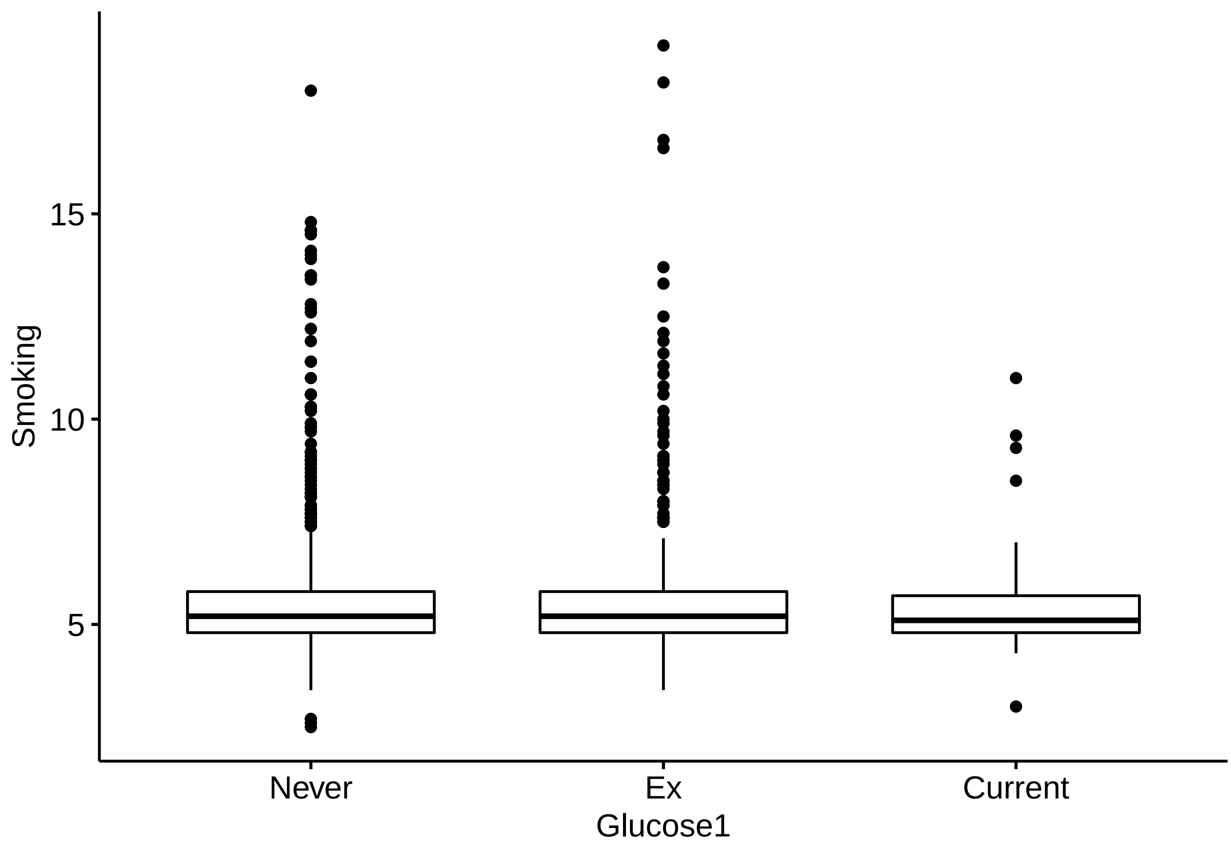
```r
chisq.test(table(data$Smoking, data$Diabetes))
```

```
##
##  Pearson's Chi-squared test
```

```
##
## data:  table(data$Smoking, data$Diabetes)
## X-squared = 1.9297, df = 2, p-value = 0.381
```

```
ggboxplot(data, x = "Smoking", y = "Glucose1",
          ylab = "Smoking", xlab = "Glucose1")
```

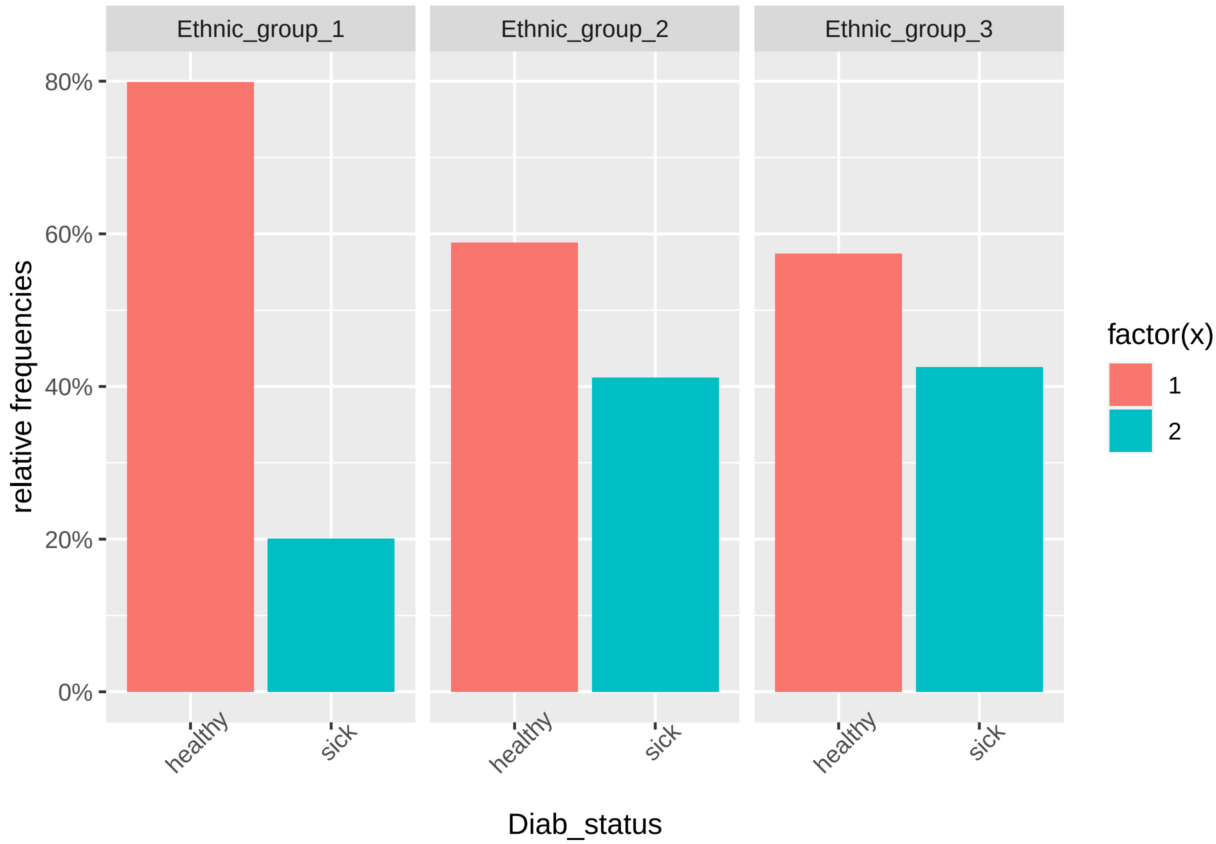
```
## Warning: Removed 2 rows containing non-finite values (stat_boxplot).
```



Ethnicity and diabetes Belonging to Ethnic group 2 and Ethnic group 2 increases chances of diabetes in 2.9 and 5.8 times accordingly.

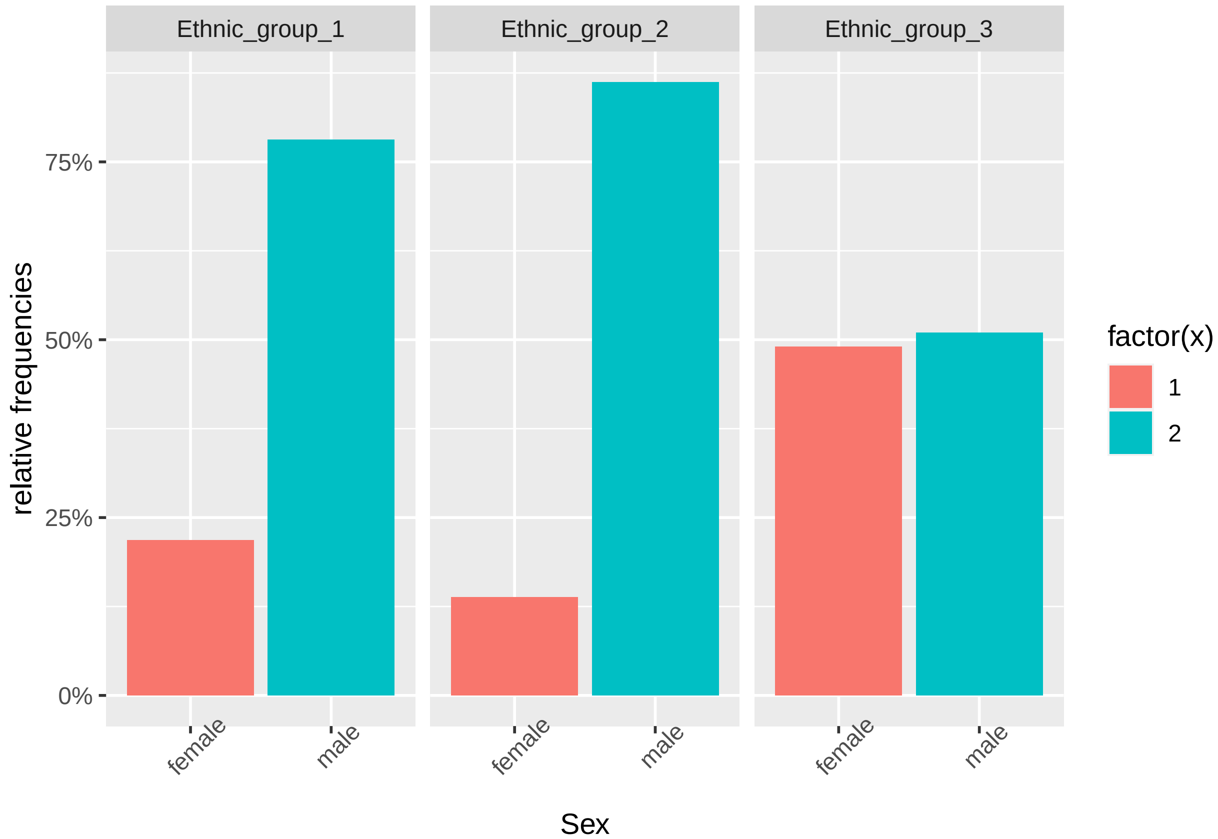
the proportion of diabetes is much higher in these two groups indeed

```
ggplot(data, aes(Diab_status, group = Ethnicity)) +
          geom_bar(aes(y = ..prop.., fill = factor(..x..)), stat="count") +
          scale_y_continuous(labels=scales::percent) +
          ylab("relative frequencies") +
          facet_grid(~Ethnicity) +
          theme(axis.text.x = element_text(angle = 45))
```



Despite the proportion of men is lower in the Ethnic group 3

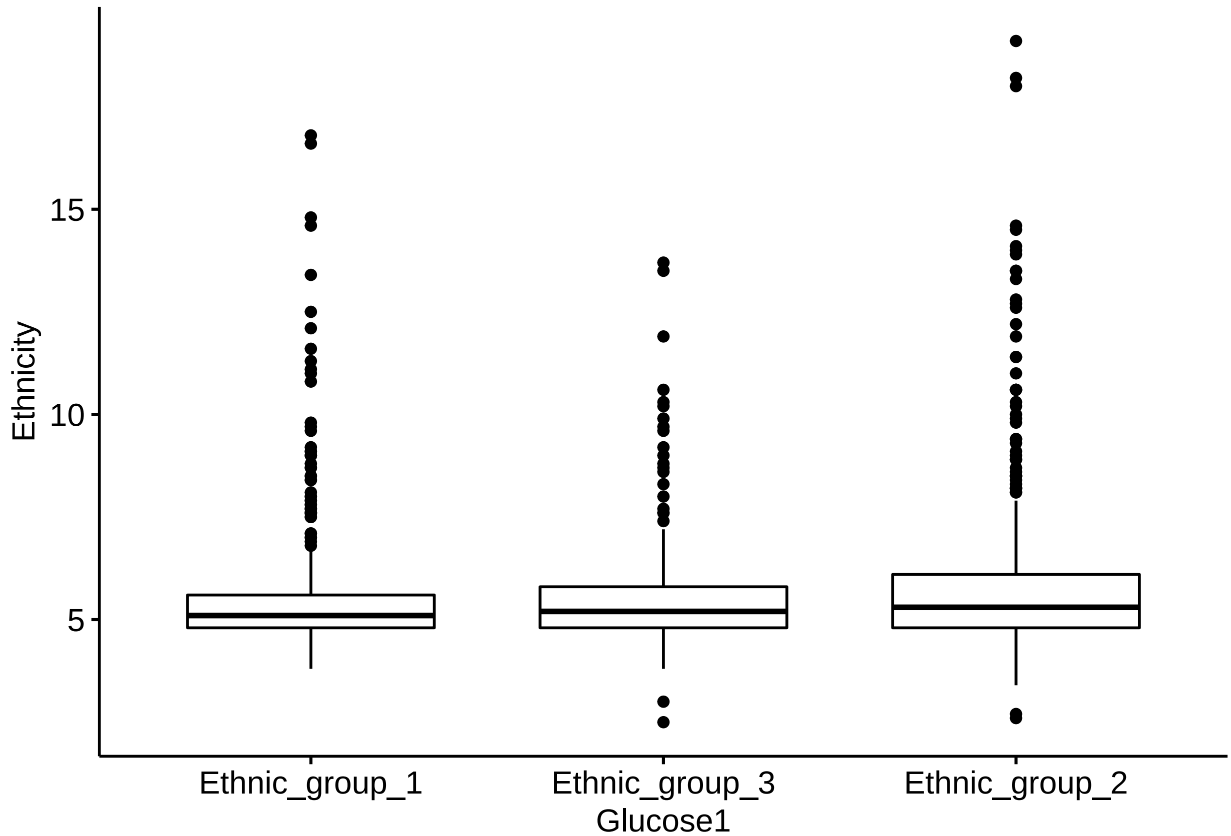
```
ggplot(data, aes(Sex, group = Ethnicity)) +
          geom_bar(aes(y = ..prop.., fill = factor(..x..)), stat="count") +
          scale_y_continuous(labels=scales::percent) +
          ylab("relative frequencies") +
          facet_grid(~Ethnicity) +
          theme(axis.text.x = element_text(angle = 45))
```



Glucose level is also higher in Ethnic group 2

```
ggboxplot(data, x = "Ethnicity", y = "Glucose1",
          ylab = "Ethnicity", xlab = "Glucose1")
```

```
## Warning: Removed 2 rows containing non-finite values (stat_boxplot).
```



kruskal.test says at least one of the groups belong to a different distribution.

```
kruskal.test(Glucose1 ~ Ethnicity, data = data)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  Glucose1 by Ethnicity
## Kruskal-Wallis chi-squared = 8.6618, df = 2, p-value = 0.01316
```

posthoc.kruskal.nemenyi.test says Ethnic group 2 is significantly different from Ethnic group 1 in regards to the glucose level Even if we apply bonferoni correction p-val = 0.0279

```
data$Ethnicity <- as.factor(data$Ethnicity)
posthoc.kruskal.nemenyi.test(Glucose1~Ethnicity, dist = c("Tukey", "Chisquare"),data = data)
```

```
## Warning in posthoc.kruskal.nemenyi.test.default(c(5, 6.300000191, 5.900000095, :
## Ties are present, p-values are not corrected.
```

```
##
##  Pairwise comparisons using Tukey and Kramer (Nemenyi) test
##                    with Tukey-Dist approximation for independent samples
```

```
## data: Glucose1 by Ethnicity
```

```
##                 Ethnic_group_1 Ethnic_group_2
## Ethnic_group_2 0.0093         -
## Ethnic_group_3 0.4822         0.5424
##
## P value adjustment method: none
```

Conclusion

The best model GLM shows that several plasma predictors have strong effects on diabetes incidence: G3, G12, G16, G17, G18, G19, G20, G21. G3 and G18 were also selected by 3 feature selection approaches (stepAIC, boruta and lasso). However given that some predictors are highly correlated (i.e. G12 and G17 or G18 and G20), it is difficult to interpret their individual effects

We see that older age increases chances of diabetes incidence (which is expected as diabetes is age-related disease). Sex difference in diabetes is not significant but in males glucose is significantly higher. Smoking slightly increase the chances of diabetes, but insignificantly for this sample. Ethnic group 2 and 3 increases chances of diabetes, and Ethnic group 2 also represents higher glucose levels.