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## PMCMR is superseded by PMCMRplus and will be no longer maintained.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_out_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

Get the list of plasma predictors with Pearson correlation coefficient > 0.85

```
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: 19 x 3
## # Groups:   value [19]
##    var1  var2  value
##    <chr> <chr> <dbl>
##  1 G28   G26   0.941
##  2 G15   G9    0.929
##  3 G31   G25   0.917
##  4 G26   G23   0.917
##  5 G33   G31   0.911
##  6 G20   G15   0.904
```

```
##   7 G6    G5    0.892
##   8 G17   G12   0.882
##   9 G36   G35   0.879
##  10 G36   G34   0.876
##  11 G15   G13   0.871
##  12 G11   G5    0.868
##  13 G14   G6    0.868
##  14 G28   G23   0.867
##  15 G16   G11   0.866
##  16 G33   G25   0.862
##  17 G35   G28   0.861
##  18 G20   G19   0.856
##  19 G7    G3    0.852
```

Plotting correlation matrix

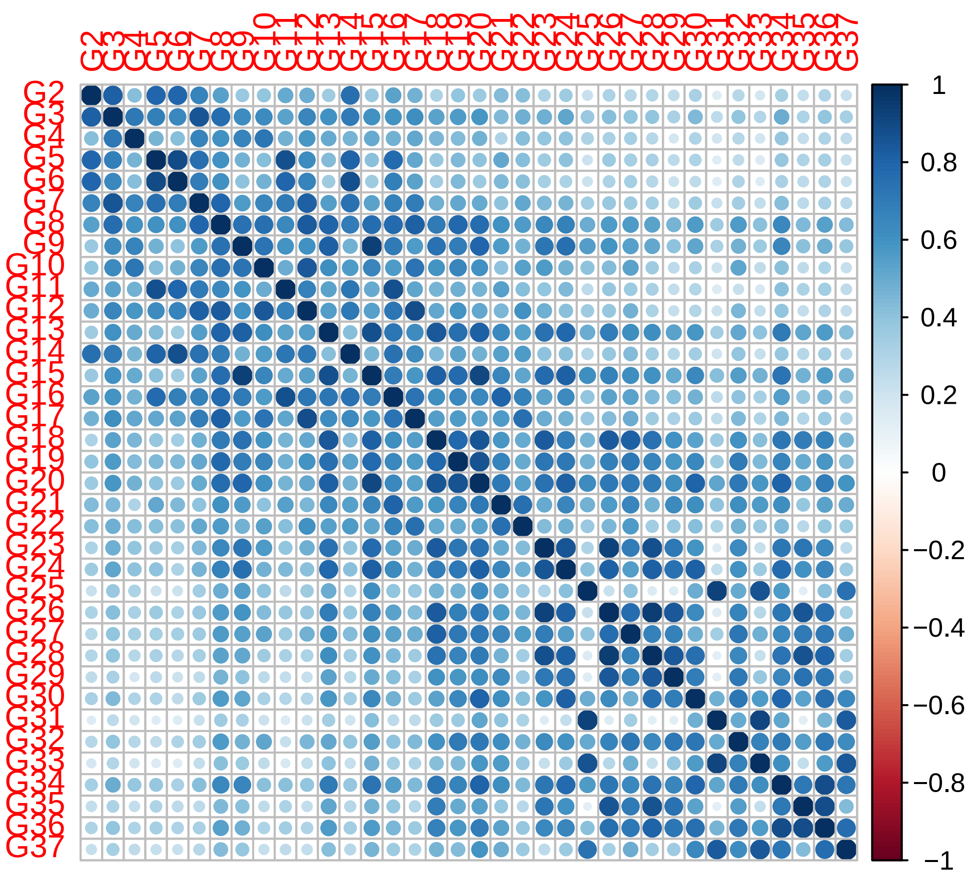
```r
corr_p <- round(cor(pred),2)
corr_p
```

```
##       G2   G3   G4   G5   G6   G7   G8   G9  G10  G11  G12  G13  G14  G15  G16
## G2  1.00 0.82 0.42 0.79 0.79 0.66 0.54 0.37 0.39 0.50 0.49 0.35 0.75 0.37 0.53
## G3  0.82 1.00 0.71 0.68 0.64 0.85 0.76 0.62 0.63 0.53 0.65 0.60 0.70 0.60 0.59
## G4  0.42 0.71 1.00 0.46 0.42 0.66 0.60 0.66 0.72 0.47 0.58 0.50 0.46 0.50 0.47
## G5  0.79 0.68 0.46 1.00 0.89 0.75 0.60 0.47 0.42 0.87 0.62 0.43 0.80 0.41 0.77
## G6  0.79 0.64 0.42 0.89 1.00 0.69 0.60 0.40 0.47 0.79 0.66 0.35 0.87 0.35 0.68
## G7  0.66 0.85 0.66 0.75 0.69 1.00 0.79 0.57 0.65 0.70 0.82 0.55 0.74 0.53 0.67
## G8  0.54 0.76 0.60 0.60 0.60 0.79 1.00 0.74 0.75 0.64 0.83 0.80 0.69 0.76 0.77
## G9  0.37 0.62 0.66 0.47 0.40 0.57 0.74 1.00 0.73 0.59 0.60 0.82 0.47 0.93 0.70
## G10 0.39 0.63 0.72 0.42 0.47 0.65 0.75 0.73 1.00 0.49 0.84 0.61 0.57 0.65 0.57
## G11 0.50 0.53 0.47 0.87 0.79 0.70 0.64 0.59 0.49 1.00 0.67 0.53 0.72 0.50 0.87
## G12 0.49 0.65 0.58 0.62 0.66 0.82 0.83 0.60 0.84 0.67 1.00 0.55 0.71 0.54 0.72
## G13 0.35 0.60 0.50 0.43 0.35 0.55 0.80 0.82 0.61 0.53 0.55 1.00 0.42 0.87 0.72
## G14 0.75 0.70 0.46 0.80 0.87 0.74 0.69 0.47 0.57 0.72 0.71 0.42 1.00 0.46 0.74
## G15 0.37 0.60 0.50 0.41 0.35 0.53 0.76 0.93 0.65 0.50 0.54 0.87 0.46 1.00 0.69
## G16 0.53 0.59 0.47 0.77 0.68 0.67 0.77 0.70 0.57 0.87 0.72 0.72 0.74 0.69 1.00
## G17 0.47 0.61 0.51 0.51 0.53 0.69 0.82 0.56 0.73 0.52 0.88 0.62 0.63 0.58 0.73
## G18 0.31 0.53 0.45 0.38 0.34 0.48 0.70 0.74 0.59 0.46 0.51 0.84 0.44 0.82 0.61
## G19 0.39 0.56 0.43 0.44 0.44 0.51 0.78 0.69 0.65 0.48 0.59 0.75 0.53 0.77 0.64
## G20 0.36 0.58 0.47 0.40 0.36 0.50 0.76 0.79 0.60 0.46 0.51 0.82 0.47 0.90 0.64
## G21 0.43 0.44 0.30 0.51 0.44 0.40 0.60 0.56 0.40 0.54 0.45 0.64 0.54 0.65 0.80
## G22 0.42 0.48 0.42 0.42 0.41 0.51 0.57 0.47 0.54 0.42 0.60 0.54 0.56 0.52 0.67
## G23 0.31 0.48 0.39 0.35 0.33 0.44 0.64 0.72 0.57 0.39 0.48 0.74 0.40 0.77 0.53
## G24 0.35 0.52 0.40 0.40 0.31 0.46 0.67 0.75 0.47 0.44 0.41 0.78 0.41 0.82 0.63
## G25 0.22 0.37 0.31 0.21 0.21 0.34 0.49 0.55 0.40 0.26 0.35 0.49 0.29 0.61 0.39
## G26 0.31 0.42 0.32 0.36 0.31 0.37 0.57 0.59 0.43 0.38 0.38 0.69 0.38 0.67 0.53
## G27 0.28 0.39 0.33 0.33 0.33 0.35 0.56 0.54 0.53 0.36 0.47 0.61 0.43 0.61 0.53
## G28 0.29 0.39 0.28 0.32 0.28 0.33 0.53 0.51 0.35 0.32 0.31 0.61 0.34 0.60 0.44
## G29 0.25 0.32 0.18 0.26 0.21 0.25 0.46 0.40 0.26 0.24 0.23 0.53 0.28 0.50 0.42
## G30 0.32 0.44 0.30 0.30 0.25 0.35 0.56 0.52 0.32 0.29 0.29 0.58 0.34 0.64 0.47
## G31 0.15 0.26 0.19 0.14 0.15 0.22 0.35 0.32 0.21 0.16 0.21 0.34 0.20 0.42 0.26
## G32 0.28 0.39 0.28 0.25 0.30 0.34 0.57 0.47 0.52 0.23 0.45 0.51 0.39 0.55 0.40
## G33 0.18 0.29 0.19 0.16 0.15 0.24 0.41 0.36 0.25 0.17 0.24 0.40 0.23 0.47 0.33
## G34 0.33 0.49 0.38 0.38 0.32 0.42 0.64 0.65 0.41 0.41 0.39 0.70 0.38 0.73 0.55
```

```
## G35 0.24 0.31 0.24 0.30 0.26 0.26 0.44 0.41 0.25 0.31 0.24 0.52 0.28 0.47 0.38
## G36 0.30 0.39 0.30 0.33 0.30 0.32 0.54 0.48 0.30 0.34 0.30 0.57 0.34 0.56 0.45
## G37 0.23 0.33 0.25 0.23 0.22 0.27 0.43 0.38 0.23 0.25 0.24 0.42 0.27 0.46 0.35
##      G17  G18  G19  G20  G21  G22  G23  G24  G25  G26  G27  G28  G29  G30  G31
## G2  0.47 0.31 0.39 0.36 0.43 0.42 0.31 0.35 0.22 0.31 0.28 0.29 0.25 0.32 0.15
## G3  0.61 0.53 0.56 0.58 0.44 0.48 0.48 0.52 0.37 0.42 0.39 0.39 0.32 0.44 0.26
## G4  0.51 0.45 0.43 0.47 0.30 0.42 0.39 0.40 0.31 0.32 0.33 0.28 0.18 0.30 0.19
## G5  0.51 0.38 0.44 0.40 0.51 0.42 0.35 0.40 0.21 0.36 0.33 0.32 0.26 0.30 0.14
## G6  0.53 0.34 0.44 0.36 0.44 0.41 0.33 0.31 0.21 0.31 0.33 0.28 0.21 0.25 0.15
## G7  0.69 0.48 0.51 0.50 0.40 0.51 0.44 0.46 0.34 0.37 0.35 0.33 0.25 0.35 0.22
## G8  0.82 0.70 0.78 0.76 0.60 0.57 0.64 0.67 0.49 0.57 0.56 0.53 0.46 0.56 0.35
## G9  0.56 0.74 0.69 0.79 0.56 0.47 0.72 0.75 0.55 0.59 0.54 0.51 0.40 0.52 0.32
## G10 0.73 0.59 0.65 0.60 0.40 0.54 0.57 0.47 0.40 0.43 0.53 0.35 0.26 0.32 0.21
## G11 0.52 0.46 0.48 0.46 0.54 0.42 0.39 0.44 0.26 0.38 0.36 0.32 0.24 0.29 0.16
## G12 0.88 0.51 0.59 0.51 0.45 0.60 0.48 0.41 0.35 0.38 0.47 0.31 0.23 0.29 0.21
## G13 0.62 0.84 0.75 0.82 0.64 0.54 0.74 0.78 0.49 0.69 0.61 0.61 0.53 0.58 0.34
## G14 0.63 0.44 0.53 0.47 0.54 0.56 0.40 0.41 0.29 0.38 0.43 0.34 0.28 0.34 0.20
## G15 0.58 0.82 0.77 0.90 0.65 0.52 0.77 0.82 0.61 0.67 0.61 0.60 0.50 0.64 0.42
## G16 0.73 0.61 0.64 0.64 0.80 0.67 0.53 0.63 0.39 0.53 0.53 0.44 0.42 0.47 0.26
## G17 1.00 0.55 0.57 0.54 0.57 0.75 0.49 0.47 0.37 0.43 0.49 0.35 0.32 0.36 0.25
## G18 0.55 1.00 0.78 0.85 0.58 0.50 0.83 0.69 0.46 0.83 0.81 0.74 0.60 0.53 0.36
## G19 0.57 0.78 1.00 0.86 0.66 0.50 0.72 0.71 0.47 0.68 0.71 0.66 0.58 0.64 0.36
## G20 0.54 0.85 0.86 1.00 0.71 0.54 0.74 0.81 0.62 0.71 0.71 0.70 0.61 0.80 0.52
## G21 0.57 0.58 0.66 0.71 1.00 0.75 0.50 0.65 0.47 0.57 0.65 0.47 0.63 0.61 0.40
## G22 0.75 0.50 0.50 0.54 0.75 1.00 0.43 0.48 0.37 0.45 0.56 0.34 0.37 0.42 0.31
## G23 0.49 0.83 0.72 0.74 0.50 0.43 1.00 0.85 0.30 0.92 0.69 0.87 0.71 0.59 0.14
## G24 0.47 0.69 0.71 0.81 0.65 0.48 0.85 1.00 0.41 0.81 0.55 0.82 0.74 0.82 0.24
## G25 0.37 0.46 0.47 0.62 0.47 0.37 0.30 0.41 1.00 0.22 0.40 0.16 0.15 0.49 0.92
## G26 0.43 0.83 0.68 0.71 0.57 0.45 0.92 0.81 0.22 1.00 0.76 0.94 0.84 0.63 0.15
## G27 0.49 0.81 0.71 0.71 0.65 0.56 0.69 0.55 0.40 0.76 1.00 0.67 0.67 0.48 0.34
## G28 0.35 0.74 0.66 0.70 0.47 0.34 0.87 0.82 0.16 0.94 0.67 1.00 0.84 0.75 0.13
## G29 0.32 0.60 0.58 0.61 0.63 0.37 0.71 0.74 0.15 0.84 0.67 0.84 1.00 0.69 0.13
## G30 0.36 0.53 0.64 0.80 0.61 0.42 0.59 0.82 0.49 0.63 0.48 0.75 0.69 1.00 0.48
## G31 0.25 0.36 0.36 0.52 0.40 0.31 0.14 0.24 0.92 0.15 0.34 0.13 0.13 0.48 1.00
## G32 0.44 0.60 0.70 0.71 0.61 0.47 0.63 0.60 0.50 0.66 0.72 0.64 0.71 0.72 0.50
## G33 0.30 0.42 0.44 0.58 0.57 0.37 0.23 0.36 0.86 0.28 0.48 0.23 0.38 0.57 0.91
## G34 0.44 0.72 0.66 0.80 0.61 0.44 0.72 0.77 0.56 0.72 0.64 0.73 0.65 0.79 0.52
## G35 0.29 0.69 0.50 0.54 0.38 0.27 0.72 0.60 0.12 0.85 0.70 0.86 0.74 0.53 0.13
## G36 0.36 0.67 0.58 0.69 0.53 0.38 0.64 0.65 0.41 0.75 0.71 0.80 0.72 0.75 0.46
## G37 0.30 0.46 0.43 0.59 0.49 0.36 0.26 0.36 0.74 0.33 0.49 0.34 0.35 0.64 0.83
##      G32  G33  G34  G35  G36  G37
## G2  0.28 0.18 0.33 0.24 0.30 0.23
## G3  0.39 0.29 0.49 0.31 0.39 0.33
## G4  0.28 0.19 0.38 0.24 0.30 0.25
## G5  0.25 0.16 0.38 0.30 0.33 0.23
## G6  0.30 0.15 0.32 0.26 0.30 0.22
## G7  0.34 0.24 0.42 0.26 0.32 0.27
## G8  0.57 0.41 0.64 0.44 0.54 0.43
## G9  0.47 0.36 0.65 0.41 0.48 0.38
## G10 0.52 0.25 0.41 0.25 0.30 0.23
## G11 0.23 0.17 0.41 0.31 0.34 0.25
## G12 0.45 0.24 0.39 0.24 0.30 0.24
## G13 0.51 0.40 0.70 0.52 0.57 0.42
## G14 0.39 0.23 0.38 0.28 0.34 0.27
```

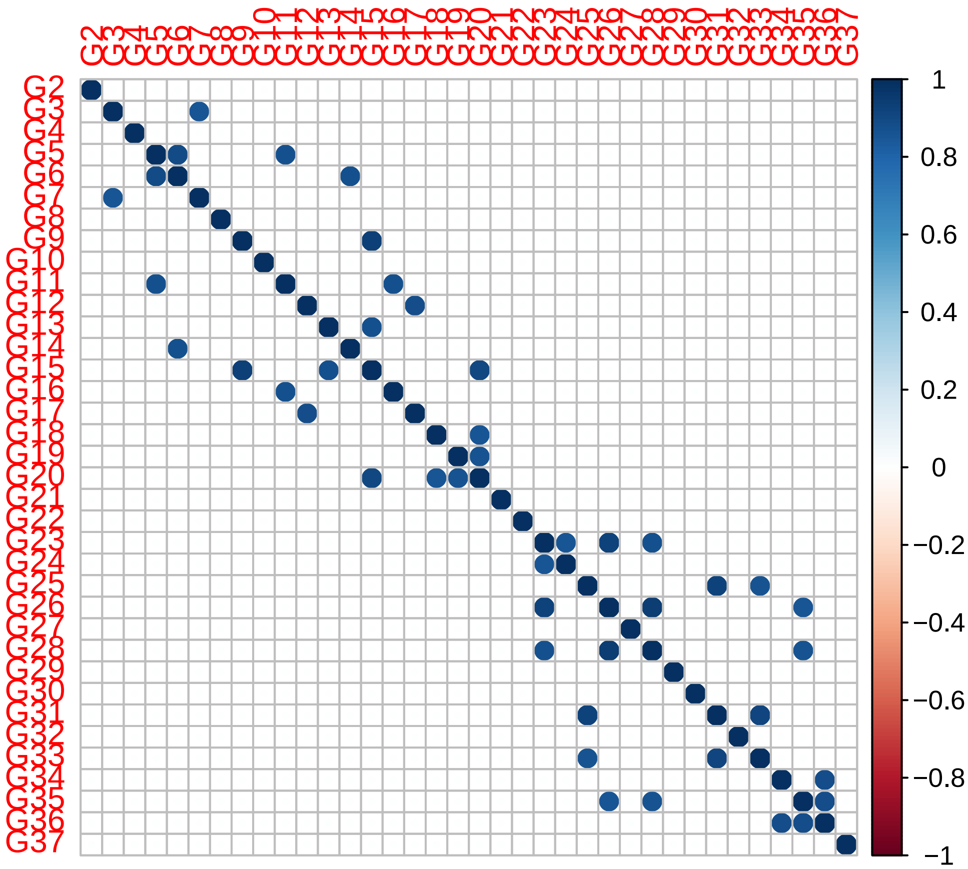
```
## G15 0.55 0.47 0.73 0.47 0.56 0.46
## G16 0.40 0.33 0.55 0.38 0.45 0.35
## G17 0.44 0.30 0.44 0.29 0.36 0.30
## G18 0.60 0.42 0.72 0.69 0.67 0.46
## G19 0.70 0.44 0.66 0.50 0.58 0.43
## G20 0.71 0.58 0.80 0.54 0.69 0.59
## G21 0.61 0.57 0.61 0.38 0.53 0.49
## G22 0.47 0.37 0.44 0.27 0.38 0.36
## G23 0.63 0.23 0.72 0.72 0.64 0.26
## G24 0.60 0.36 0.77 0.60 0.65 0.36
## G25 0.50 0.86 0.56 0.12 0.41 0.74
## G26 0.66 0.28 0.72 0.85 0.75 0.33
## G27 0.72 0.48 0.64 0.70 0.71 0.49
## G28 0.64 0.23 0.73 0.86 0.80 0.34
## G29 0.71 0.38 0.65 0.74 0.72 0.35
## G30 0.72 0.57 0.79 0.53 0.75 0.64
## G31 0.50 0.91 0.52 0.13 0.46 0.83
## G32 1.00 0.67 0.70 0.55 0.71 0.62
## G33 0.67 1.00 0.61 0.24 0.56 0.84
## G34 0.70 0.61 1.00 0.71 0.88 0.72
## G35 0.55 0.24 0.71 1.00 0.88 0.43
## G36 0.71 0.56 0.88 0.88 1.00 0.76
## G37 0.62 0.84 0.72 0.43 0.76 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

```
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

```
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
## -2.0523  -0.5921  -0.2428   0.4597   3.0415
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)                 -3.6403     0.6916  -5.263 1.41e-07 ***
## AgeAge_70_75                 0.8341     0.4261   1.957 0.050302 .
## Sexmale                      1.2204     0.5635   2.166 0.030345 *
## EthnicityEthnic_group_2      1.2882     0.4800   2.684 0.007278 **
## EthnicityEthnic_group_3      3.0728     0.8326   3.691 0.000224 ***
## SmokingCurrent               1.4671     0.7125   2.059 0.039476 *
## G2                          -1.4462     0.7632  -1.895 0.058106 .
## G3                           3.6021     1.0399   3.464 0.000532 ***
## G5                           2.1838     1.0551   2.070 0.038486 *
## G6                          -1.5131     0.6744  -2.244 0.024861 *
## G7                          -2.4077     1.0624  -2.266 0.023430 *
## G9                          -3.0768     0.6713  -4.583 4.57e-06 ***
## G12                          5.7915     1.2900   4.489 7.14e-06 ***
## G13                          1.1905     0.6619   1.799 0.072075 .
## G16                         -3.9060     1.1201  -3.487 0.000488 ***
## G17                         -3.4022     0.8062  -4.220 2.44e-05 ***
## G19                         -3.1586     0.6891  -4.583 4.58e-06 ***
## G20                          1.2562     0.6828   1.840 0.065782 .
## G21                          2.9113     0.8275   3.518 0.000434 ***
## G24                          3.0571     0.7515   4.068 4.74e-05 ***
## G29                         -1.2232     0.4924  -2.484 0.012986 *
## G33                          1.2201     0.5928   2.058 0.039563 *
## G34                          1.0649     0.6536   1.629 0.103234
## G37                         -2.2352     0.7099  -3.149 0.001640 **
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 307.27  on 247  degrees of freedom
## Residual deviance: 187.82  on 224  degrees of freedom
## AIC: 235.82
##
## Number of Fisher Scoring iterations: 6
```

So the features selected by stepAIC are: AgeAge_70_75,Sexmale ,EthnicityEthnic_group_2 , EthnicityEthnic_group_3 ,SmokingCurrent, G2, G3 ,G5 ,G6, G7, G9 , G12, G13 , G16 , G17 , G19 , G20, G21 , G24, G29, G33, G34, G37. Let's create a formula for them

```r
var_stepAIC = c("AgeAge_70_75","Sexmale"      ,"EthnicityEthnic_group_2" , "EthnicityEthnic_group_3" ,"S
f_stepAIC = as.formula(
  paste("outcome", paste(var_stepAIC, collapse = " + "), sep = " ~ ")
)
```

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

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

Get NonRejected features selected by boruta

```r
getNonRejectedFormula(boruta)
```

```
## outcome ~ EthnicityEthnic_group_2 + G9 + G13 + G15 + G16 + G18 +
##     G19 + G23 + G24 + G26 + G28 + G29 + G30 + G32
## <environment: 0x0000000018cf4828>
```

Make formula for boruta

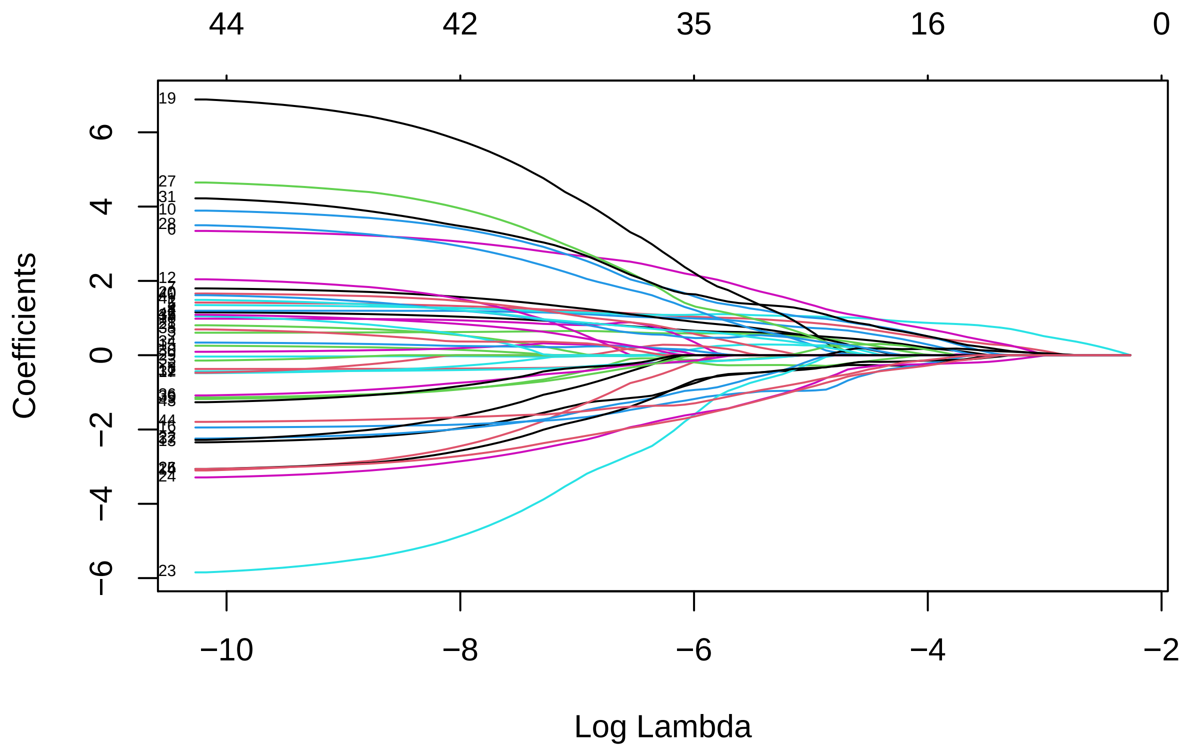
```r
var_boruta = c("EthnicityEthnic_group_2", "G9","G13","G15","G18","G19","G20","G23","G24","G26","G29","G3
f_boruta = as.formula(
  paste("outcome", paste(var_boruta, collapse = " + "), sep = " ~ ")
)
```

Feature selection using lasso

```r
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")
```

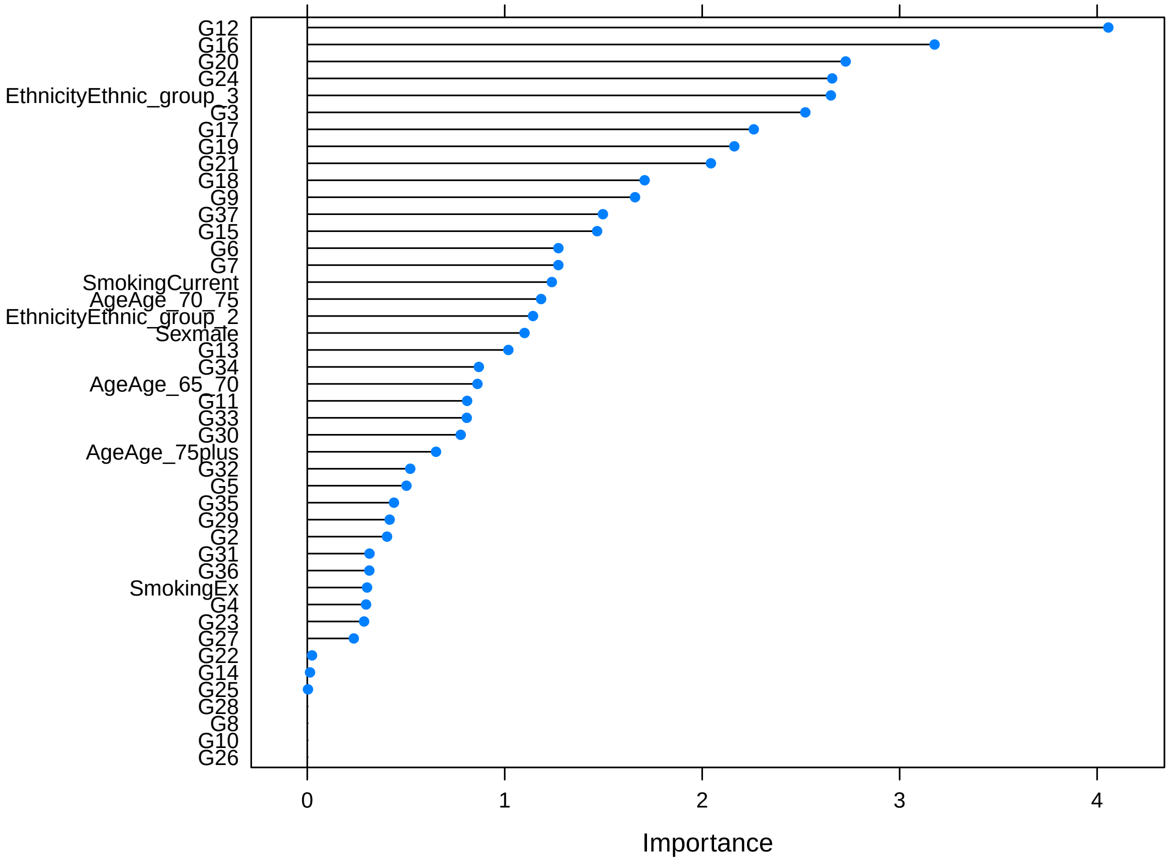
This plot shows which predictors remain while increasing lambda

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



plot Variable importance

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




```
varImp(lasso)
```

```
## glmnet variable importance
##
##   only 20 most important variables shown (out of 44)
##
##                         Overall
## G12                      100.00
## G16                       78.30
## G20                       67.21
## G24                       65.52
## EthnicityEthnic_group_3   65.36
## G3                        62.17
## G17                       55.74
## G19                       53.31
## G21                       50.39
## G18                       42.11
## G9                        40.92
## G37                       36.91
## G15                       36.18
## G6                        31.35
## G7                        31.34
## SmokingCurrent            30.53
```

```
## AgeAge_70_75               29.19
## EthnicityEthnic_group_2   28.18
## Sexmale                   27.12
## G13                       25.10
```

lasso significantly decreased the number of predictors: G12, G16, G20,G24,EthnicityEthnic_group_3,G3,G17,G19,G21,G18,G
So here are the formula for lasso selected predictors

```
var_lasso = c("G12", "G16", "G20","G24","EthnicityEthnic_group_3","G3","G17","G19","G21","G18","G9","G3
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

```r
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

```r
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
```

Save the results

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

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

```
##                        Model Accuracy_train Accuracy_test        F1       AUC
## Accuracy5  GLM_step_stepAIC      0.7487091     0.7316384 0.8204159 0.6355812
## Accuracy25      SVM_stepAIC      0.7469880     0.7316384 0.8210923 0.6330849
## Accuracy13    ridge_stepAIC      0.7435456     0.7259887 0.8193669 0.6190015
## Accuracy3         GLM_lasso      0.7710843     0.7175141 0.8062016 0.6353204
## Accuracy11         EN_lasso      0.7710843     0.7146893 0.8053950 0.6282787
## Accuracy17    lasso_stepAIC      0.7487091     0.7146893 0.8076190 0.6207899
## Accuracy9        EN_stepAIC      0.7487091     0.7118644 0.8053435 0.6187407
## Accuracy12        ridge_all      0.7418244     0.7118644 0.8210526 0.5613264
## Accuracy            GLM_all      0.7865749     0.7090395 0.8030593 0.6166915
## Accuracy8            EN_all      0.7831325     0.7062147 0.8015267 0.6121461
```

```
## Accuracy1       GLM_stepAIC      0.7521515     0.7033898 0.7969052 0.6175857
## Accuracy20           RF_all      1.0000000     0.7033898 0.8188153 0.5451937
## Accuracy4      GLM_step_all      0.7590361     0.6977401 0.7946257 0.6059985
## Accuracy21       RF_stepAIC      1.0000000     0.6977401 0.8158348 0.5311103
## Accuracy24          SVM_all      0.7607573     0.6977401 0.7946257 0.6059985
## Accuracy26       SVM_boruta      0.7383821     0.6949153 0.8091873 0.5465350
## Accuracy27        SVM_lasso      0.7607573     0.6949153 0.7954545 0.5939642
## Accuracy16        lasso_all      0.7332186     0.6920904 0.8084359 0.5394933
## Accuracy7    GLM_step_lasso      0.7555938     0.6864407 0.7869482 0.5928092
## Accuracy23         RF_lasso      1.0000000     0.6836158 0.7932961 0.5707899
## Accuracy18     lasso_boruta      0.7228916     0.6807910 0.7985740 0.5362891
## Accuracy6   GLM_step_boruta      0.7366609     0.6779661 0.7896679 0.5567064
## Accuracy19      lasso_lasso      0.7452668     0.6779661 0.7927273 0.5467213
## Accuracy10        EN_boruta      0.7349398     0.6751412 0.7897623 0.5471684
## Accuracy14     ridge_boruta      0.7246127     0.6723164 0.7906137 0.5351341
## Accuracy15      ridge_lasso      0.7246127     0.6723164 0.7906137 0.5351341
## Accuracy2        GLM_boruta      0.7349398     0.6694915 0.7821229 0.5530551
## Accuracy22        RF_boruta      1.0000000     0.6638418 0.7783985 0.5464605
```

```
results
```

```
##                       Model Accuracy_train Accuracy_test        F1       AUC
## Accuracy            GLM_all      0.7865749     0.7090395 0.8030593 0.6166915
## Accuracy1       GLM_stepAIC      0.7521515     0.7033898 0.7969052 0.6175857
## Accuracy2        GLM_boruta      0.7349398     0.6694915 0.7821229 0.5530551
## Accuracy3         GLM_lasso      0.7710843     0.7175141 0.8062016 0.6353204
## Accuracy4      GLM_step_all      0.7590361     0.6977401 0.7946257 0.6059985
## Accuracy5  GLM_step_stepAIC      0.7487091     0.7316384 0.8204159 0.6355812
## Accuracy6   GLM_step_boruta      0.7366609     0.6779661 0.7896679 0.5567064
## Accuracy7    GLM_step_lasso      0.7555938     0.6864407 0.7869482 0.5928092
## Accuracy8            EN_all      0.7831325     0.7062147 0.8015267 0.6121461
## Accuracy9        EN_stepAIC      0.7487091     0.7118644 0.8053435 0.6187407
## Accuracy10        EN_boruta      0.7349398     0.6751412 0.7897623 0.5471684
## Accuracy11         EN_lasso      0.7710843     0.7146893 0.8053950 0.6282787
## Accuracy12        ridge_all      0.7418244     0.7118644 0.8210526 0.5613264
## Accuracy13    ridge_stepAIC      0.7435456     0.7259887 0.8193669 0.6190015
## Accuracy14     ridge_boruta      0.7246127     0.6723164 0.7906137 0.5351341
## Accuracy15      ridge_lasso      0.7246127     0.6723164 0.7906137 0.5351341
## Accuracy16        lasso_all      0.7332186     0.6920904 0.8084359 0.5394933
## Accuracy17    lasso_stepAIC      0.7487091     0.7146893 0.8076190 0.6207899
## Accuracy18     lasso_boruta      0.7228916     0.6807910 0.7985740 0.5362891
## Accuracy19      lasso_lasso      0.7452668     0.6779661 0.7927273 0.5467213
## Accuracy20           RF_all      1.0000000     0.7033898 0.8188153 0.5451937
## Accuracy21       RF_stepAIC      1.0000000     0.6977401 0.8158348 0.5311103
## Accuracy22        RF_boruta      1.0000000     0.6638418 0.7783985 0.5464605
## Accuracy23         RF_lasso      1.0000000     0.6836158 0.7932961 0.5707899
## Accuracy24          SVM_all      0.7607573     0.6977401 0.7946257 0.6059985
## Accuracy25      SVM_stepAIC      0.7469880     0.7316384 0.8210923 0.6330849
## Accuracy26       SVM_boruta      0.7383821     0.6949153 0.8091873 0.5465350
## Accuracy27        SVM_lasso      0.7607573     0.6949153 0.7954545 0.5939642
```

We see that overfitting for most linear models and SVM is relatively low, while it is high for Random forest. Interestingely, for the dataset with removed outliers, the best performance is for models which which tool predictors selected by StepAIC. However, the best achieved accuracy for the test is 0.731, 2% lower than the best model for the dataset with all the data.