Final Project Code

Homayoon Fotros

Libraries & Data Pre-processing

```r
library(rio) ## importing data
library(tidyverse) ## data wrangling
library(corrplot) ## correlation plot
library(boot) ## bootstrap
library(gridExtra) ## grid visualization
library(glmnet) ## glm models
library(caret) ##  various ML tools & models
library(ROCR) ## ROC-AUC score
library(leaps) ## model selection
library(bestglm) ## extended model selection
#library(GGally) ## optional: full features visualization

dt <- import('breastR2.xlsx') ## Make sure the data set file is in the working directory
head(dt[,1:9]) ## excluding the response variable
```

```
##   Age      BMI Glucose Insulin      HOMA  Leptin Adiponectin Resistin   MCP.1
## 1  48 23.50000      70   2.707 0.4674087  8.8071    9.702400  7.99585 417.114
## 2  83 20.69049      92   3.115 0.7068973  8.8438    5.429285  4.06405 468.786
## 3  82 23.12467      91   4.498 1.0096511 17.9393   22.432040  9.27715 554.697
## 4  68 21.36752      77   3.226 0.6127249  9.8827    7.169560 12.76600 928.220
## 5  86 21.11111      92   3.549 0.8053864  6.6994    4.819240 10.57635 773.920
## 6  49 22.85446      92   3.226 0.7320869  6.8317   13.679750 10.31760 530.410
```

```r
## The positive response (diagnosed with BC) is coded as 1
dt$Classification <- ifelse(dt$Classification==1, 0,1)
table(dt$Classification)
```

```
##
##  0  1
## 52 64
```

```r
str(dt) ## All the features are continuous numerical variables
```

```
## 'data.frame':    116 obs. of  10 variables:
##  $ Age           : num  48 83 82 68 86 49 89 76 73 75 ...
##  $ BMI           : num  23.5 20.7 23.1 21.4 21.1 ...
##  $ Glucose       : num  70 92 91 77 92 92 77 118 97 83 ...
##  $ Insulin       : num  2.71 3.12 4.5 3.23 3.55 ...
##  $ HOMA          : num  0.467 0.707 1.01 0.613 0.805 ...
##  $ Leptin        : num  8.81 8.84 17.94 9.88 6.7 ...
##  $ Adiponectin   : num  9.7 5.43 22.43 7.17 4.82 ...
##  $ Resistin      : num  8 4.06 9.28 12.77 10.58 ...
##  $ MCP.1         : num  417 469 555 928 774 ...
```

```
##  $ Classification: num  0 0 0 0 0 0 0 0 0 0 ...
```

```r
summary(dt)
```

```
##       Age             BMI            Glucose          Insulin      
##  Min.   :24.0   Min.   :18.37   Min.   : 60.00   Min.   : 2.432  
##  1st Qu.:45.0   1st Qu.:22.97   1st Qu.: 85.75   1st Qu.: 4.359  
##  Median :56.0   Median :27.66   Median : 92.00   Median : 5.925  
##  Mean   :57.3   Mean   :27.58   Mean   : 97.79   Mean   :10.012  
##  3rd Qu.:71.0   3rd Qu.:31.24   3rd Qu.:102.00   3rd Qu.:11.189  
##  Max.   :89.0   Max.   :38.58   Max.   :201.00   Max.   :58.460  
##       HOMA              Leptin        Adiponectin        Resistin     
##  Min.   : 0.4674   Min.   : 4.311   Min.   : 1.656   Min.   : 3.210  
##  1st Qu.: 0.9180   1st Qu.:12.314   1st Qu.: 5.474   1st Qu.: 6.882  
##  Median : 1.3809   Median :20.271   Median : 8.353   Median :10.828  
##  Mean   : 2.6950   Mean   :26.615   Mean   :10.181   Mean   :14.726  
##  3rd Qu.: 2.8578   3rd Qu.:37.378   3rd Qu.:11.816   3rd Qu.:17.755  
##  Max.   :25.0503   Max.   :90.280   Max.   :38.040   Max.   :82.100  
##      MCP.1         Classification  
##  Min.   :  45.84   Min.   :0.0000  
##  1st Qu.: 269.98   1st Qu.:0.0000  
##  Median : 471.32   Median :1.0000  
##  Mean   : 534.65   Mean   :0.5517  
##  3rd Qu.: 700.09   3rd Qu.:1.0000  
##  Max.   :1698.44   Max.   :1.0000  
```

```r
set.seed(123) ## for replication
## Setting aside the validation holdout set (15% of the data)
train_test <- createDataPartition(dt$Classification, times = 1, p = 0.15, list = F)
holdout_set <- dt[train_test,]
dt <- dt[-train_test,]

## Number of instances in the training set:
nrow(dt)
```

```
## [1] 98
```

```r
## The training set is fairly split between the two classes of response
table(dt$Classification)
```

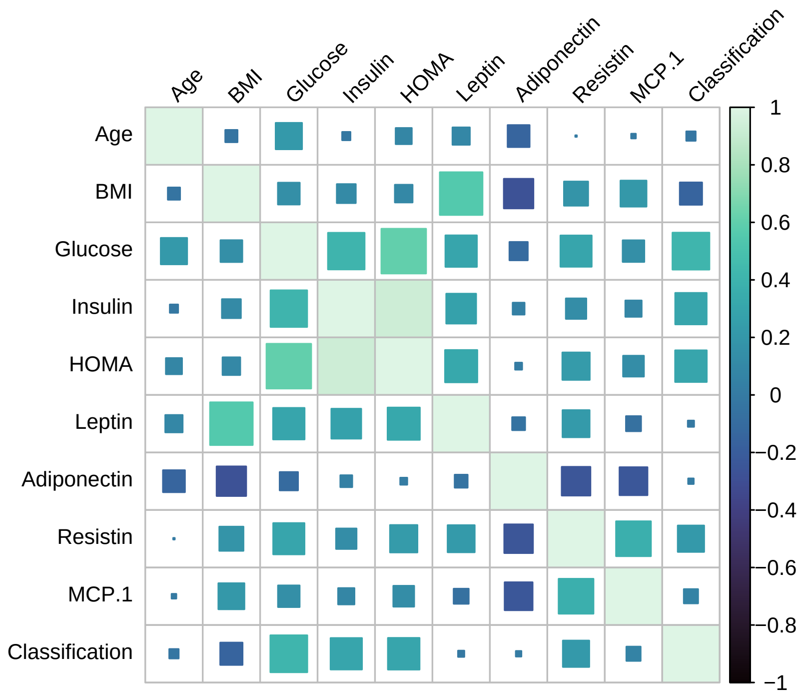
```
## 
##  0  1 
## 44 54
```

```r
table(holdout_set$Classification)
```

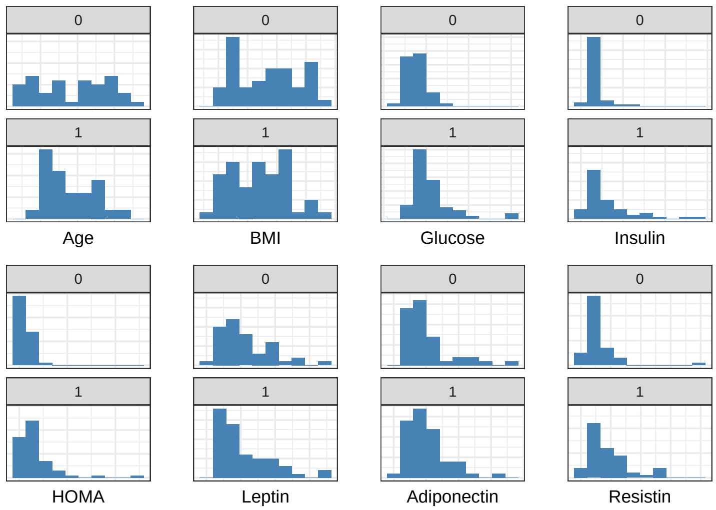
```
## 
##  0  1 
##  8 10
```

Exploratory Data Analysis (EDA)

Correlation between features and the response

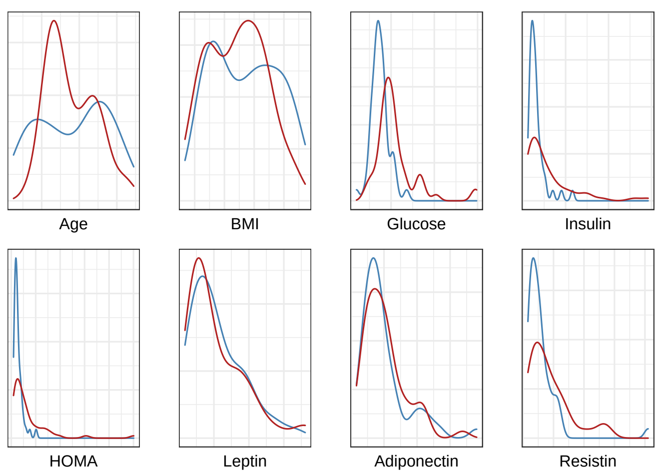

Age
BMI
Glucose
Insulin
HOMA
Leptin
Adiponectin
Resistin
MCP.1
Classification
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1

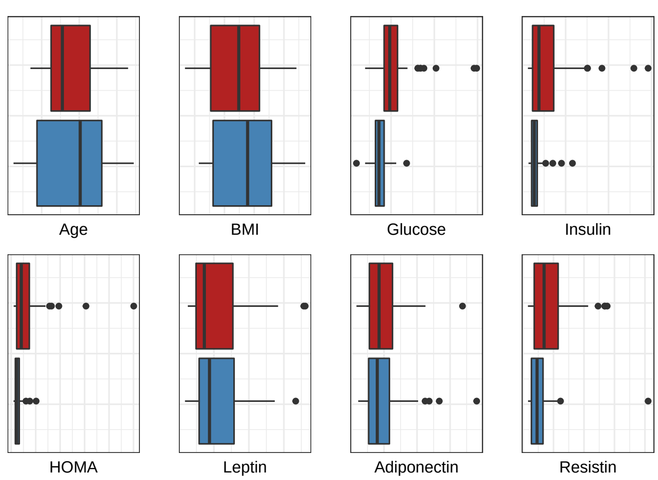
Distribution of Instances for each Feature by Class

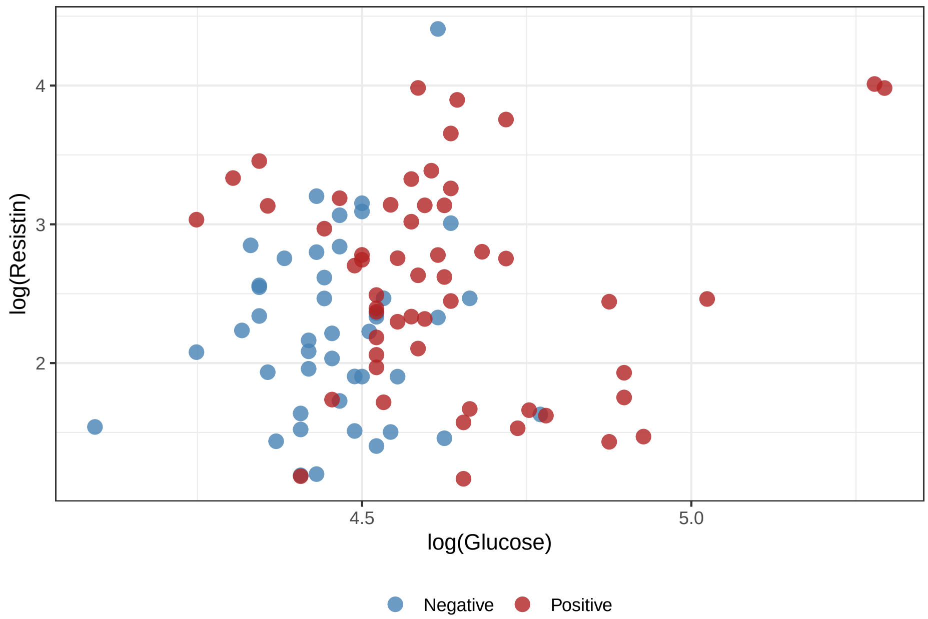


Frequency & Range of Instances for each Feature by Class

Red: Positive (w/ BC) | Blue: Negative (wo/ BC)







Data Preparation

Log & Scale Transformation

```
dt_old <- dt ## backup of train data set before transformations
#dt <- dt_old

log_trans <- function (x) {sapply(x, log)}
scale_trans <- function(x){sapply(x, scale)}

dt[,3:9] <- log_trans(dt[,3:9])
dt[,1:9] <- data.frame(scale_trans(dt[,1:9]))

holdout_old <- holdout_set ## backup of test data set before transformations
#holdout_set<- holdout_old

holdout_set[,3:9] <- log_trans(holdout_set[,3:9])
holdout_set[,1:9] <- data.frame(scale_trans(holdout_set[,1:9]))
```

Setting Up Feature Matrix & Dataframe

```
# Train data matrix and data frame
mdl.mat <- model.matrix(Classification ~ . + I(Age^2) + I(BMI^2), data = dt)
train_dat <- data.frame(mdl.mat, Classification = factor(dt$Classification),
                        check.names = F)[,-1]

# Validation data matrix and data frame
mdl.tst <- model.matrix(Classification ~ . + I(Age^2) + I(BMI^2), data = holdout_set)
test_dat <- data.frame(mdl.tst, Classification = factor(holdout_set$Classification),
                       check.names = F)[,-1]
```

Classification Models

Linear Logistic Models

Logistic Model w/ & Wo/ Bootstrap

```
## Simple Logistic Model
mdl.log.simp <- glm(Classification ~ ., family = 'binomial', data = train_dat)
coef.simp <- coef(mdl.log.simp)
#summary(mdl.log.simp)
exp(coef(mdl.log.simp))
```

```
## (Intercept)         Age         BMI     Glucose     Insulin        HOMA
##   3.6636467   0.5248574   0.1995530   0.0000000   0.0000000         Inf
##      Leptin Adiponectin    Resistin       MCP.1  `I(Age^2)`  `I(BMI^2)`
##   1.5915139   1.7569791   2.2827981   1.2534737   0.2128077   0.6976718
```

```
## Testing for models' fit (reduced (without Age^2 & BMI^2) vs. full)
mdl.log.red <- glm(Classification ~ ., family = 'binomial', data = train_dat[,c(-10,-11)])
#summary(mdl.log.red)
anova(mdl.log.red, mdl.log.simp, test = 'F')
```

```
## Analysis of Deviance Table
##
## Model 1: Classification ~ Age + BMI + Glucose + Insulin + HOMA + Leptin +
##     Adiponectin + Resistin + MCP.1
## Model 2: Classification ~ Age + BMI + Glucose + Insulin + HOMA + Leptin +
##     Adiponectin + Resistin + MCP.1 + `I(Age^2)` + `I(BMI^2)`
##   Resid. Df Resid. Dev Df Deviance      F   Pr(>F)
## 1        88     81.000
## 2        86     62.696  2   18.303 9.1517 0.000106 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Pseudo R-Squared values
DescTools::PseudoR2(mdl.log.simp,which = c('CoxSnell', 'McFadden'))
```

```
##  CoxSnell  McFadden
## 0.5210244 0.5350134
```

```
## Bootstrap Logistic Model
## Setting up Bootstrap function
set.seed(123)
fn_est <- function(data, idx){
  mdl.boot <- glm(Classification ~ ., family = 'binomial', data = data[idx,])
  coefs <- mdl.boot$coefficients
  return(coefs)
}

L <- 1000 ## bootstrap samples = 1000
bts_est <- boot(data = train_dat, statistic = fn_est, R=L)

## storing bootstrap results in a matrix
boot.mat <- matrix(0, nrow = L, ncol=length(colnames(mdl.mat)))
for (i in 1:L) boot.mat[i,] <- bts_est$t[i,]

boot.mat <- apply(boot.mat,MARGIN = 2,median,na.rm = T)  ## Median of bootstraped coefficients
```

```
names(boot.mat) <- colnames(mdl.mat)

## setting up a function to calculate prob. score out of logit score
logit.pred <- function(data, coefs, thresh=0.5){
  factor(ifelse(1/(1+exp(-(data %*% coefs)))>=thresh, 1,0))
}

y_prd_simp <- logit.pred(mdl.mat, coef.simp) ## prediction of simple logistic model
y_prd_boot <- logit.pred(mdl.mat, boot.mat) ## prediction of logit bootstrap model

## setting up a function to calculate classification metrics
metrics <- function(y_true, y_hat){
  tr_accu <- mean(y_true == y_hat)
  tr_sens <- sensitivity(y_true, y_hat, positive = '1')
  tr_spc <- specificity(y_true, y_hat, negative = '0')
  tr_f1 <- F_meas(y_true, y_hat)
  prd <- prediction(as.numeric(y_hat),labels = y_true)
  tr_auc <- as.numeric(performance(prd,measure="auc")@y.values)
  data.frame(Accuracy = tr_accu, Recall=tr_sens, Specificity=tr_spc, F1=tr_f1,AUC=tr_auc)
}

## this data frame store our models' prediction metrics on the training set
model.perf <- data.frame(Model="Logistic",
                         round(metrics(train_dat$Classification, y_prd_simp),2))
model.perf <- rbind(model.perf, data.frame(Model="Logistic (Boot)",
                         round(metrics(train_dat$Classification, y_prd_boot),2)))
```

Lasso CV Models

```
## Lasso LOOV
set.seed(123)
mdl.loigtLOOV <- cv.glmnet(x = mdl.mat[,-1], y = train_dat$Classification,
                           family=binomial, nfolds = nrow(train_dat))

## Coefficients (LOOV Lasso)
Lasso_LV_Coefs <- predict(mdl.loigtLOOV, newx = mdl.mat[,-1], type='coefficient')
coef_lass_LV <- data.frame(Var = c('Intercept',
                                   Lasso_LV_Coefs@Dimnames[[1]][-1])[c(Lasso_LV_Coefs@i+1)],
           `Lasso LOOV` = round(Lasso_LV_Coefs@x, 3))

coef_lass_LV_ac <- rbind(coef_lass_LV,
                         data.frame(Var = colnames(mdl.mat)[-1][c(!((colnames(mdl.mat)[-1])%in% coef_la

## Performance on the training set
y_prd_lgLOOV <- factor(ifelse(predict(mdl.loigtLOOV, newx = mdl.mat[,-1],
                                      type='response')>0.5, 1,0))

## Lasso K-fold(10)
set.seed(123)
mdl.loigtKfold <- cv.glmnet(x = mdl.mat[,-1], y = train_dat$Classification,
                            family=binomial, nfolds = 10)

#Coefficients (k-fold Lasso) [uncomment to run]
```

```r
Lasso_K10_Coefs <- predict(mdl.loigtKfold, newx = mdl.mat[,-1],type='coefficient')

coef_lass_k10 <- data.frame(Var = c('Intercept',
                                    Lasso_K10_Coefs@Dimnames[[1]][-1])[c(Lasso_K10_Coefs@i+1)],
           `Lasso K-fold` = round(Lasso_K10_Coefs@x, 3))

coef_lass_k10_ac <- rbind(coef_lass_k10,
                          data.frame(Var = colnames(mdl.mat)[-1][c(!((colnames(mdl.mat)[-1])%in% coef_la

## Performance on the training set
y_prd_lgKfold <- factor(ifelse(predict(mdl.loigtKfold, newx = mdl.mat[,-1],
                                       type='response')>0.5, 1,0))

## coefficients of the Lasso models:
coefs_lasso_final <- merge(coef_lass_k10_ac, coef_lass_LV_ac,  by='Var')

## storing the models' performance in the metrics data frame
model.perf <- rbind(model.perf, data.frame(
            Model='Logistic (LOOV)',round(metrics(train_dat$Classification,
                                                  y_prd_lgLOOV),2)))
model.perf <- rbind(model.perf, data.frame(
            Model='Logistic (Kfold=10)', round(metrics(train_dat$Classification,
                                                       y_prd_lgKfold),2)))
```

Logit Subset Models

```r
## Subset Criteria (BestGLM)
set.seed(123)
lgs_subs <- function(data, ctn='AIC', meth='exhaustive'){
  mdl.bglm <- bestglm(data, family = binomial, IC=ctn, method = meth)
  tmp_names <- names(mdl.bglm$BestModel$coefficients)
  tmp_names <- case_when(
              tmp_names == "`I(Age^2)`" ~ "I(Age^2)",
              tmp_names == "`I(BMI^2)`" ~ "I(BMI^2)",
              TRUE ~ as.character(tmp_names)
              )
  coefs <- mdl.bglm$BestModel$coefficients
  names(coefs) <- tmp_names
  return(coefs)
}
coefs_ex_AIC <- lgs_subs(data=train_dat)
coefs_ex_BIC <- lgs_subs(data=train_dat,ctn='BIC')
coefs_fw_AIC <- lgs_subs(data=train_dat,meth = 'forward')
coefs_fw_BIC <- lgs_subs(data=train_dat,ctn = 'BIC', meth = 'forward')
coefs_bw_AIC <- lgs_subs(data=train_dat,meth = 'backward')
coefs_bw_BIC <- lgs_subs(data=train_dat,ctn = 'BIC', meth = 'backward')

y_prd_ex_AIC <- logit.pred(mdl.mat[,names(coefs_ex_AIC)],
                            coefs_ex_AIC)
y_prd_ex_BIC <- logit.pred(mdl.mat[,names(coefs_ex_BIC)],
                           coefs_ex_BIC)
y_prd_fw_AIC <- logit.pred(mdl.mat[,names(coefs_fw_AIC)],
                           coefs_fw_AIC)
```

```r
y_prd_fw_BIC <- logit.pred(mdl.mat[,names(coefs_fw_BIC)],
                           coefs_fw_BIC)
y_prd_bw_AIC <- logit.pred(mdl.mat[,names(coefs_bw_AIC)],
                           coefs_bw_AIC)
y_prd_bw_BIC <- logit.pred(mdl.mat[,names(coefs_bw_BIC)],
                           coefs_bw_BIC)
## Metrics of All Subset Models
cbind(Model=c('Best subset - AIC', 'Best subset - BIC', 'Forward - AIC',
              'Forward - BIC','Backward - AIC', 'Backward - BIC' ),
      round(rbind(metrics(train_dat$Classification, y_prd_ex_AIC ),
            metrics(train_dat$Classification, y_prd_ex_BIC ),
            metrics(train_dat$Classification, y_prd_fw_AIC ),
            metrics(train_dat$Classification, y_prd_fw_BIC ),
            metrics(train_dat$Classification, y_prd_bw_AIC ),
            metrics(train_dat$Classification, y_prd_bw_BIC )),2))
```

```
##               Model Accuracy Recall Specificity   F1  AUC
## 1 Best subset - AIC     0.87   0.87        0.86 0.85 0.86
## 2 Best subset - BIC     0.84   0.84        0.83 0.81 0.83
## 3     Forward - AIC     0.87   0.87        0.86 0.85 0.86
## 4     Forward - BIC     0.84   0.84        0.83 0.81 0.83
## 5    Backward - AIC     0.87   0.87        0.86 0.85 0.86
## 6    Backward - BIC     0.84   0.84        0.83 0.81 0.83
```

```r
## Coefficients of all selected logistic models
coefs_all_logit <- merge(data.frame(Var = c('Intercept',colnames(mdl.mat)[-1]),
                                    Simple.Logit = round(exp(coef.simp),3),
                                    row.names = NULL, check.names = F),
      merge (coefs_lasso_final,
             data.frame(Var = c('Intercept',names(coefs_fw_AIC)[-1]),
                        Subset = round(exp(coefs_fw_AIC),3), row.names = NULL),
             by='Var', all.x = T, all.y = T, ),
      all.x = T, all.y = T, by='Var')

coefs_all_logit[is.na(coefs_all_logit)] <-0
coefs_all_logit[rank(c('Intercept',colnames(mdl.mat)[-1])),] %>%
  data.frame(row.names = NULL)
```

```
##            Var Simple.Logit Lasso.K.fold Lasso.LOOV Subset
## 1    Intercept        3.664        0.734      0.786  2.059
## 2          Age        0.525        0.000      0.000  0.000
## 3          BMI        0.200       -0.245     -0.293  0.267
## 4      Glucose        0.000        0.739      0.783  0.000
## 5      Insulin        0.000        0.000      0.000  0.000
## 6         HOMA          Inf        0.149      0.176    Inf
## 7       Leptin        1.592        0.000      0.000  0.000
## 8  Adiponectin        1.757        0.000      0.000  1.810
## 9     Resistin        2.283        0.218      0.256  2.670
## 10       MCP.1        1.253        0.000      0.000  0.000
## 11    I(Age^2)        0.213       -0.490     -0.526  0.236
## 12    I(BMI^2)        0.698        0.000     -0.010  0.000
```

```r
model.perf <- rbind(model.perf, data.frame(
  Model='Logistic (Fwd Sel.)', round(metrics(train_dat$Classification, y_prd_fw_AIC ),2)))
```

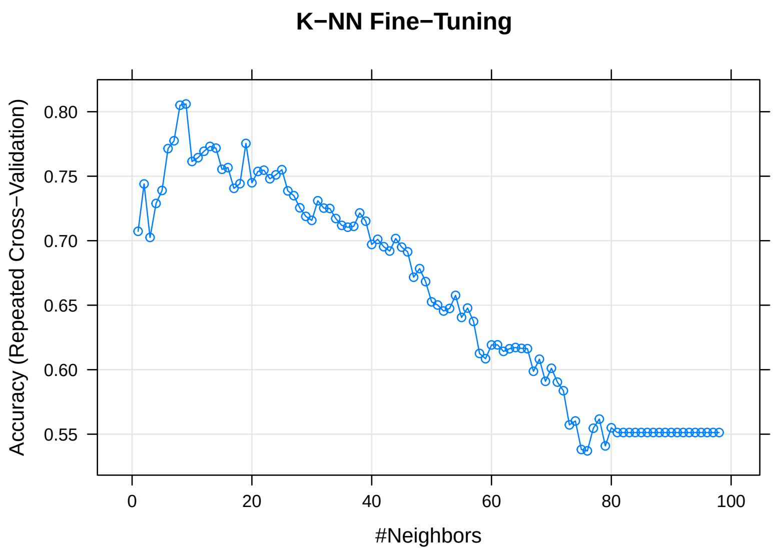
Non-Parametric Models

k-NN

```
set.seed(123)
## set CV (10 fold, 3 repeats)
tr <- trainControl(method = 'repeatedcv', number = 10, repeats = 3)

k_grid <- data.frame(k=seq(1,nrow(dt),1)) # tune for number of neighbors: 1 to n
clf_knn <- train(Classification ~. , data = train_dat,
                 trControl = tr, method = 'knn', tuneGrid = k_grid)
# Best k (number of neighbors) based on cross-validated accuracy
clf_knn$bestTune
```

```
##   k
## 9 9
```

```
plot(clf_knn, main = 'K-NN Fine-Tuning')
```




```
y_prd_knn <- predict(clf_knn, newdata = train_dat, type='raw')
model.perf <- rbind(model.perf, data.frame(
  Model='K-NN', round(metrics(train_dat$Classification, y_prd_knn),2)))
```

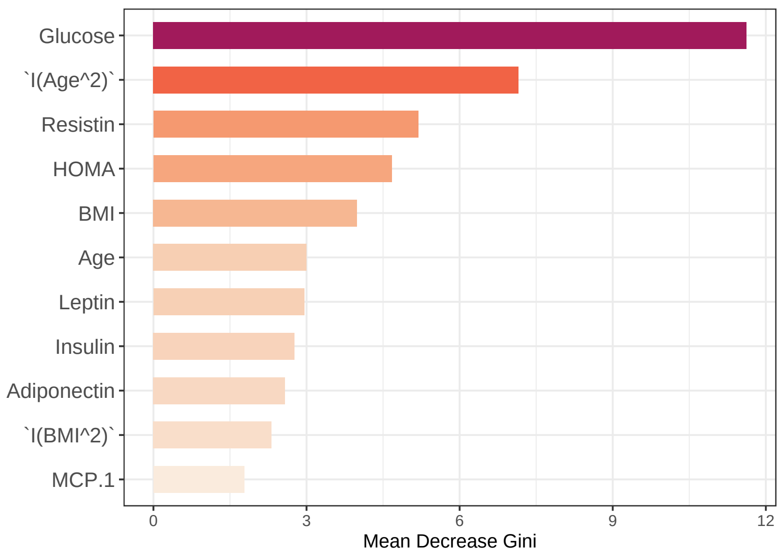
Random Forest

```
rf_res <- data.frame(N_Trees = NULL, mtry = NULL, Accuracy =NULL)

for (nt in seq(500, 3000, by = 500)){

  set.seed(123)
  tg <- expand.grid(.mtry = seq(1,11,by=1))
  clf_rf <- train(Classification ~. , data = train_dat,
                  trControl = tr, method = 'rf', tuneGrid = tg, ntree=nt)


  rf_res <- rbind(rf_res, data.frame(N_Trees = clf_rf$finalModel$ntree,
                                     mtry = clf_rf$finalModel$mtry,
```

```
                                    Accuracy = max(clf_rf$results$Accuracy)))

}
## result of grid search for RF models
rf_res
```

```
##   N_Trees mtry  Accuracy
## 1     500   10 0.7394949
## 2    1000    6 0.7398653
## 3    1500    6 0.7361616
## 4    2000    6 0.7324579
## 5    2500    6 0.7394949
## 6    3000    6 0.7321549
```

```
## so we use mtry = 6 and ntree=1000

clf_rf <- train(Classification ~. , data = train_dat, method = 'rf',
                tuneGrid = expand.grid(.mtry = 6), ntree=1000)
rf_feats <- clf_rf$finalModel$importance[,1]
ggplot() +
  geom_col(aes(y=reorder(names(rf_feats), rf_feats), x= rf_feats, fill=rf_feats), width = 0.6) +
  scale_fill_viridis(option = 'rocket', direction = -1, begin=0.4, end=1) +
  labs(x='Mean Decrease Gini', y='') +
  theme(legend.position = 'none', axis.text.y = element_text(size=12))
```




```
y_pred_rf <- predict(clf_rf, newdata = train_dat, type = 'raw')

model.perf <- rbind(model.perf, data.frame(
  Model='RF', round(metrics(train_dat$Classification, clf_rf$finalModel$predicted),2)))
```

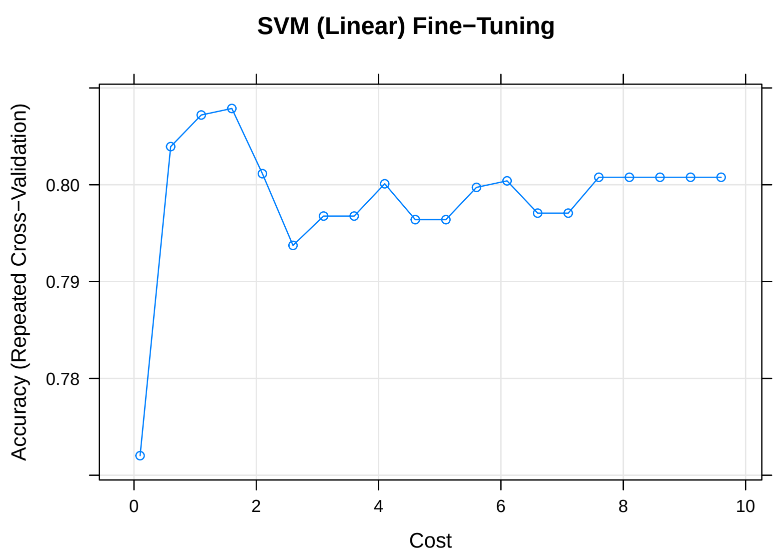
SVM (Linear & Radial Kernel)

```
### SVM Linear
set.seed(123)

clf_svm_l <- train(Classification~., data = train_dat,
```

```r
      trControl = tr, method = "svmLinear", preProcess = c("center","scale"),
      tuneGrid = expand.grid(C=seq(0.1,10, by=0.5)))

y_pred_svm_l <- predict(clf_svm_l, newdata = train_dat, type = 'raw')
clf_svm_l$bestTune
```

```
##     C
## 4 1.6
```

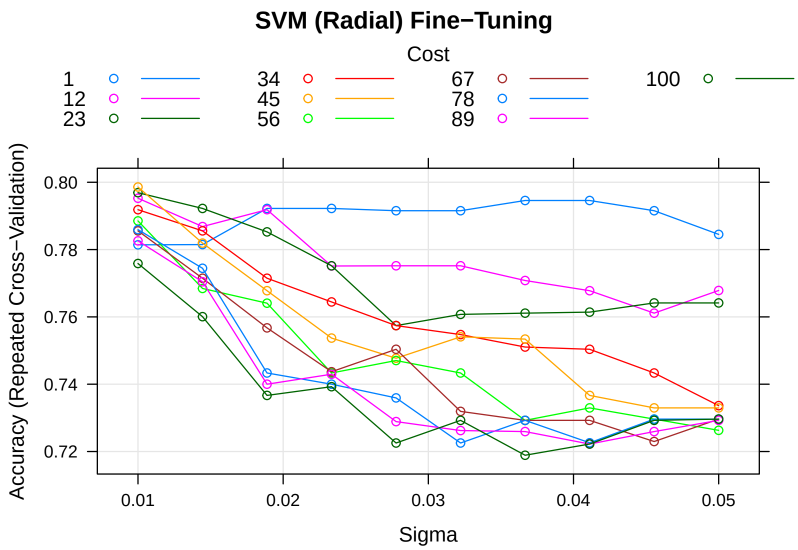
```r
plot(clf_svm_l, main='SVM (Linear) Fine-Tuning')
```




```r
model.perf <- rbind(model.perf, data.frame(
  Model='SVM (Linear)', round(metrics(train_dat$Classification, y_pred_svm_l),2)))


### SVM Radial
set.seed(123)

clf_svm_r <- train(Classification~., data = train_dat,
                 trControl = tr, method = "svmRadial",
                 preProcess = c("center","scale"),
                 tuneGrid = expand.grid(C=seq(1,100, length.out = 10),
                                        sigma= seq(0.01, 0.05, length.out = 10)))
clf_svm_r$bestTune
```

```
##    sigma  C
## 41  0.01 45
```

```r
#clf_svm_r$finalModel
plot(clf_svm_r, main='SVM (Radial) Fine-Tuning')
```




```
y_pred_svm_r <- predict(clf_svm_r, newdata = train_dat, type = 'raw')

model.perf <- rbind(model.perf, data.frame(
  Model='SVM (Radial)', round(metrics(train_dat$Classification, y_pred_svm_r),2)))
```

```
model.perf
```

```
##                   Model Accuracy Recall Specificity   F1  AUC
## 1              Logistic     0.87   0.87        0.86 0.85 0.86
## 2      Logistic (Boot)      0.64   0.73        0.58 0.65 0.65
## 3      Logistic (LOOV)      0.83   0.80        0.86 0.79 0.82
## 4 Logistic (Kfold=10)      0.82   0.79        0.86 0.78 0.81
## 5 Logistic (Fwd Sel.)      0.87   0.87        0.86 0.85 0.86
## 6                 K-NN     0.86   0.87        0.84 0.84 0.86
## 7                   RF     0.76   0.75        0.76 0.71 0.75
## 8         SVM (Linear)     0.88   0.86        0.90 0.86 0.87
## 9         SVM (Radial)     0.94   0.93        0.95 0.93 0.94
```

Applying on the Validation/Holdout Set

```
## Logistic Simple Model
test_lm <- logit.pred(mdl.tst,coef.simp)
tst.perf <- data.frame(Model="Logistic (Base)",round(metrics(test_dat$Classification, test_lm),2))


## Logistic Bootstrap
test_boot <- logit.pred(mdl.tst,boot.mat)
tst.perf <- rbind(tst.perf, data.frame(Model="Logistic (Boot)",
                                       round(metrics(test_dat$Classification, test_boot),2)))

## Lasso
test_lgLOOV <- factor(ifelse(predict(mdl.loigtLOOV, newx = mdl.tst[,-1],
                                     type='response')>0.5, 1,0))

test_lgKfold <- factor(ifelse(predict(mdl.loigtKfold, newx = mdl.tst[,-1],
```

```
                                      type='response')>0.5, 1,0))

tst.perf <- rbind(tst.perf, data.frame(
  Model='Lasso (LOOV)',round(metrics(test_dat$Classification, test_lgLOOV),2)))
tst.perf <- rbind(tst.perf, data.frame(
  Model='Lasso (K-fold)', round(metrics(test_dat$Classification, test_lgKfold),2)))

## FW. Sel.
test_fw_AIC <- logit.pred(mdl.tst[,names(coefs_fw_AIC)],
                           coefs_fw_AIC)
tst.perf <- rbind(tst.perf, data.frame(
  Model='Logistic (Fwd Sel.)', round(metrics(test_dat$Classification, test_fw_AIC ),2)))

## K-NN
test_knn <- predict(clf_knn, newdata = test_dat, type='raw')
tst.perf <- rbind(tst.perf, data.frame(
  Model='K-NN', round(metrics(test_dat$Classification, test_knn),2)))

## RF
test_rf <- predict(clf_rf, newdata = test_dat, type = 'raw')
tst.perf <- rbind(tst.perf, data.frame(
  Model='RF', round(metrics(test_dat$Classification, test_rf),2)))


## SVM Linear
test_svm_l <- predict(clf_svm_l, newdata = test_dat, type = 'raw')
tst.perf <- rbind(tst.perf, data.frame(
  Model='SVM (Linear)', round(metrics(test_dat$Classification, test_svm_l),2)))

## SVM Radial
test_svm_r <- predict(clf_svm_r, newdata = test_dat, type = 'raw')
tst.perf <- rbind(tst.perf, data.frame(
  Model='SVM (Radial)', round(metrics(test_dat$Classification, test_svm_r),2)))

tst.perf
```

```
##                 Model Accuracy Recall Specificity   F1  AUC
## 1     Logistic (Base)     0.50   0.55        0.43 0.40 0.49
## 2     Logistic (Boot)     0.44   0.50        0.33 0.29 0.42
## 3        Lasso (LOOV)     0.67   0.67        0.67 0.57 0.65
## 4      Lasso (K-fold)     0.67   0.67        0.67 0.57 0.65
## 5 Logistic (Fwd Sel.)     0.50   0.55        0.43 0.40 0.49
## 6                K-NN     0.83   0.82        0.86 0.80 0.82
## 7                  RF     0.72   0.86        0.64 0.74 0.74
## 8        SVM (Linear)     0.72   0.73        0.71 0.67 0.71
## 9        SVM (Radial)     0.89   1.00        0.80 0.89 0.90
```