project_ML

Yaojie Wang, Jianan Zhu

3/14/2021

Steps

(1) Import data
(2) Check missing values
(3) Change diagnosis results from "M" and "B" to 1 and 0
(4) Check correlation among variables
(5) Conduct pca to reduce variables and correlation
(6) Get data with only reduced variables
(7) Divide data into training and test data

import data

```
bc <- read.csv("data.csv")
```

```
bc <- bc %>%
  select(-c(X, id)) # delete a blank column and id in the original data
```

data cleaning

```
#transforming the target variable(diagnosis) as M==1 and B==0
bc$diagnosis <- ifelse(bc$diagnosis == 'M', 1, 0)
bc$diagnosis <- as.factor(bc$diagnosis)
```

```
head(bc)
```

```
##   diagnosis radius_mean texture_mean perimeter_mean area_mean smoothness_mean
## 1         1       17.99        10.38         122.80    1001.0         0.11840
## 2         1       20.57        17.77         132.90    1326.0         0.08474
## 3         1       19.69        21.25         130.00    1203.0         0.10960
## 4         1       11.42        20.38          77.58     386.1         0.14250
## 5         1       20.29        14.34         135.10    1297.0         0.10030
## 6         1       12.45        15.70          82.57     477.1         0.12780
##   compactness_mean concavity_mean concave.points_mean symmetry_mean
## 1          0.27760         0.3001             0.14710        0.2419
## 2          0.07864         0.0869             0.07017        0.1812
```

```
## 3          0.15990         0.1974             0.12790        0.2069
## 4          0.28390         0.2414             0.10520        0.2597
## 5          0.13280         0.1980             0.10430        0.1809
## 6          0.17000         0.1578             0.08089        0.2087
##   fractal_dimension_mean radius_se texture_se perimeter_se area_se
## 1                0.07871    1.0950     0.9053        8.589  153.40
## 2                0.05667    0.5435     0.7339        3.398   74.08
## 3                0.05999    0.7456     0.7869        4.585   94.03
## 4                0.09744    0.4956     1.1560        3.445   27.23
## 5                0.05883    0.7572     0.7813        5.438   94.44
## 6                0.07613    0.3345     0.8902        2.217   27.19
##   smoothness_se compactness_se concavity_se concave.points_se symmetry_se
## 1      0.006399        0.04904      0.05373           0.01587     0.03003
## 2      0.005225        0.01308      0.01860           0.01340     0.01389
## 3      0.006150        0.04006      0.03832           0.02058     0.02250
## 4      0.009110        0.07458      0.05661           0.01867     0.05963
## 5      0.011490        0.02461      0.05688           0.01885     0.01756
## 6      0.007510        0.03345      0.03672           0.01137     0.02165
##   fractal_dimension_se radius_worst texture_worst perimeter_worst area_worst
## 1             0.006193        25.38         17.33          184.60     2019.0
## 2             0.003532        24.99         23.41          158.80     1956.0
## 3             0.004571        23.57         25.53          152.50     1709.0
## 4             0.009208        14.91         26.50           98.87      567.7
## 5             0.005115        22.54         16.67          152.20     1575.0
## 6             0.005082        15.47         23.75          103.40      741.6
##   smoothness_worst compactness_worst concavity_worst concave.points_worst
## 1           0.1622            0.6656          0.7119               0.2654
## 2           0.1238            0.1866          0.2416               0.1860
## 3           0.1444            0.4245          0.4504               0.2430
## 4           0.2098            0.8663          0.6869               0.2575
## 5           0.1374            0.2050          0.4000               0.1625
## 6           0.1791            0.5249          0.5355               0.1741
##   symmetry_worst fractal_dimension_worst
## 1         0.4601                 0.11890
## 2         0.2750                 0.08902
## 3         0.3613                 0.08758
## 4         0.6638                 0.17300
## 5         0.2364                 0.07678
## 6         0.3985                 0.12440
```

```
dim(bc)
```

```
## [1] 569  31
```

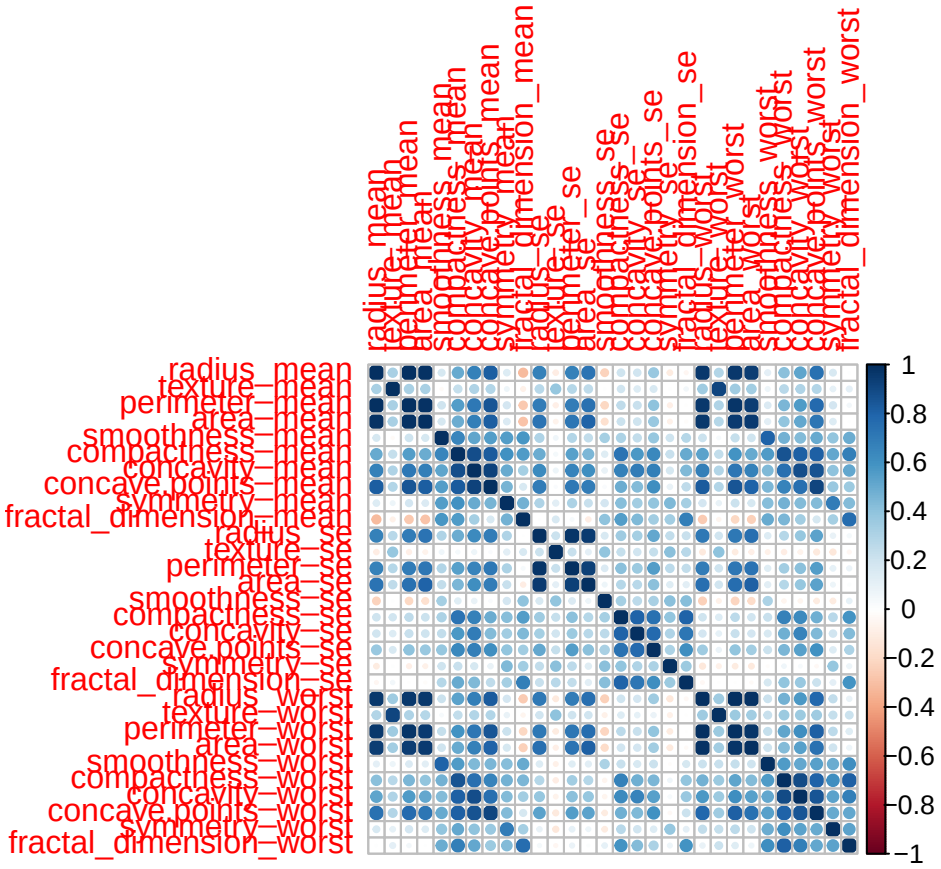
```
sum(is.na(bc)) # check if there are missing values
```

```
## [1] 0
```

There is no missing value in this data.

correlation

```
X=bc[,-c(1)] ## except "diagnosis"
corrplot(cor(X))
```



The plot shows high correlation among variables, and this can lead to skewed or misleading results for Logistic Regression, so we conduct pca to reduce variable and correlation

feature selection: pca

```
pca <- prcomp(bc[,c(2:31)], scale = T)
summary(pca)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     3.6444 2.3857 1.67867 1.40735 1.28403 1.09880 0.82172
## Proportion of Variance 0.4427 0.1897 0.09393 0.06602 0.05496 0.04025 0.02251
## Cumulative Proportion  0.4427 0.6324 0.72636 0.79239 0.84734 0.88759 0.91010
##                           PC8    PC9    PC10   PC11    PC12    PC13    PC14
## Standard deviation     0.69037 0.6457 0.59219 0.5421 0.51104 0.49128 0.39624
## Proportion of Variance 0.01589 0.0139 0.01169 0.0098 0.00871 0.00805 0.00523
## Cumulative Proportion  0.92598 0.9399 0.95157 0.9614 0.97007 0.97812 0.98335
##                           PC15    PC16    PC17    PC18    PC19    PC20   PC21
```

```
## Standard deviation     0.30681 0.28260 0.24372 0.22939 0.22244 0.17652 0.1731
## Proportion of Variance 0.00314 0.00266 0.00198 0.00175 0.00165 0.00104 0.0010
## Cumulative Proportion  0.98649 0.98915 0.99113 0.99288 0.99453 0.99557 0.9966
##                            PC22    PC23   PC24    PC25    PC26    PC27    PC28
## Standard deviation     0.16565 0.15602 0.1344 0.12442 0.09043 0.08307 0.03987
## Proportion of Variance 0.00091 0.00081 0.0006 0.00052 0.00027 0.00023 0.00005
## Cumulative Proportion  0.99749 0.99830 0.9989 0.99942 0.99969 0.99992 0.99997
##                            PC29    PC30
## Standard deviation     0.02736 0.01153
## Proportion of Variance 0.00002 0.00000
## Cumulative Proportion  1.00000 1.00000
```

we can see that pc10 can explan 95% of the results

```
## get the name of the top 10 predictors that contribute
## most to pc1.
loading_scores <- pca$rotation[,1]
bc_scores <- abs(loading_scores) ## get the magnitudes
bc_score_ranked <- sort(bc_scores, decreasing=TRUE)
top_10_bc <- names(bc_score_ranked[1:10])

top_10_bc ## show the names of the top 10 variables
```

```
##  [1] "concave.points_mean"  "concavity_mean"       "concave.points_worst"
##  [4] "compactness_mean"     "perimeter_worst"      "concavity_worst"
##  [7] "radius_worst"         "perimeter_mean"       "area_worst"
## [10] "area_mean"
```

get data with only reduced variables

```
bc1 <- bc # for random forests
```

```
bc <- bc %>%
  select( diagnosis,concave.points_mean, concavity_mean, concave.points_worst,compactness_mean, perimet
radius_worst, perimeter_mean, area_worst, area_mean)
```

```
head(bc)
```

```
##   diagnosis concave.points_mean concavity_mean concave.points_worst
## 1         1             0.14710         0.3001               0.2654
## 2         1             0.07017         0.0869               0.1860
## 3         1             0.12790         0.1974               0.2430
## 4         1             0.10520         0.2414               0.2575
## 5         1             0.10430         0.1980               0.1625
## 6         1             0.08089         0.1578               0.1741
##   compactness_mean perimeter_worst concavity_worst radius_worst perimeter_mean
## 1          0.27760          184.60          0.7119        25.38         122.80
## 2          0.07864          158.80          0.2416        24.99         132.90
## 3          0.15990          152.50          0.4504        23.57         130.00
## 4          0.28390           98.87          0.6869        14.91          77.58
```

```
## 5          0.13280          152.20          0.4000        22.54         135.10
## 6          0.17000          103.40          0.5355        15.47          82.57
##   area_worst area_mean
## 1     2019.0    1001.0
## 2     1956.0    1326.0
## 3     1709.0    1203.0
## 4      567.7     386.1
## 5     1575.0    1297.0
## 6      741.6     477.1
```

set training and test dataset

```
set.seed(2)
nrow(bc)
```

```
## [1] 569
```

```
train <- sample(1:569, 300)
test <- bc[-train,]
```

logistic regression

```
#fit a full model
mod <- glm(diagnosis ~ ., data = bc, subset = train, family = "binomial")
```

```
step(mod)
```

```
## Start:  AIC=97.3
## diagnosis ~ concave.points_mean + concavity_mean + concave.points_worst +
##     compactness_mean + perimeter_worst + concavity_worst + radius_worst +
##     perimeter_mean + area_worst + area_mean
##
##                        Df Deviance     AIC
## - perimeter_mean        1   75.308  95.308
## - concave.points_worst  1   75.432  95.432
## - perimeter_worst       1   75.444  95.444
## - area_mean             1   75.650  95.650
## - radius_worst          1   76.278  96.278
## - concavity_mean        1   76.776  96.776
## - compactness_mean      1   76.818  96.818
## <none>                      75.298  97.298
## - concavity_worst       1   77.961  97.961
## - concave.points_mean   1   80.557 100.557
## - area_worst            1   82.121 102.121
##
## Step:  AIC=95.31
## diagnosis ~ concave.points_mean + concavity_mean + concave.points_worst +
##     compactness_mean + perimeter_worst + concavity_worst + radius_worst +
```

```
##     area_worst + area_mean
##
##                         Df Deviance     AIC
## - concave.points_worst   1   75.465  93.465
## - perimeter_worst        1   75.546  93.546
## - radius_worst           1   76.309  94.309
## - concavity_mean         1   77.026  95.026
## <none>                       75.308  95.308
## - compactness_mean       1   77.927  95.927
## - concavity_worst        1   77.964  95.964
## - concave.points_mean    1   81.112  99.112
## - area_worst             1   83.170 101.170
## - area_mean              1   84.876 102.876
##
## Step:  AIC=93.46
## diagnosis ~ concave.points_mean + concavity_mean + compactness_mean +
##     perimeter_worst + concavity_worst + radius_worst + area_worst +
##     area_mean
##
##                         Df Deviance     AIC
## - perimeter_worst        1   75.735  91.735
## - radius_worst           1   76.401  92.401
## <none>                       75.465  93.465
## - concavity_mean         1   78.440  94.440
## - compactness_mean       1   78.453  94.453
## - concavity_worst        1   82.221  98.221
## - area_worst             1   83.643  99.643
## - area_mean              1   87.745 103.745
## - concave.points_mean    1   90.951 106.951
##
## Step:  AIC=91.73
## diagnosis ~ concave.points_mean + concavity_mean + compactness_mean +
##     concavity_worst + radius_worst + area_worst + area_mean
##
##                         Df Deviance     AIC
## - radius_worst           1   76.920  90.920
## <none>                       75.735  91.735
## - concavity_mean         1   78.441  92.441
## - compactness_mean       1   78.968  92.968
## - concavity_worst        1   82.814  96.814
## - area_worst             1   83.643  97.643
## - area_mean              1   89.242 103.242
## - concave.points_mean    1   90.954 104.954
##
## Step:  AIC=90.92
## diagnosis ~ concave.points_mean + concavity_mean + compactness_mean +
##     concavity_worst + area_worst + area_mean
##
##                         Df Deviance     AIC
## - concavity_mean         1   78.487  90.487
## <none>                       76.920  90.920
## - compactness_mean       1   80.493  92.493
## - concavity_worst        1   82.939  94.939
## - area_mean              1   90.748 102.748
```

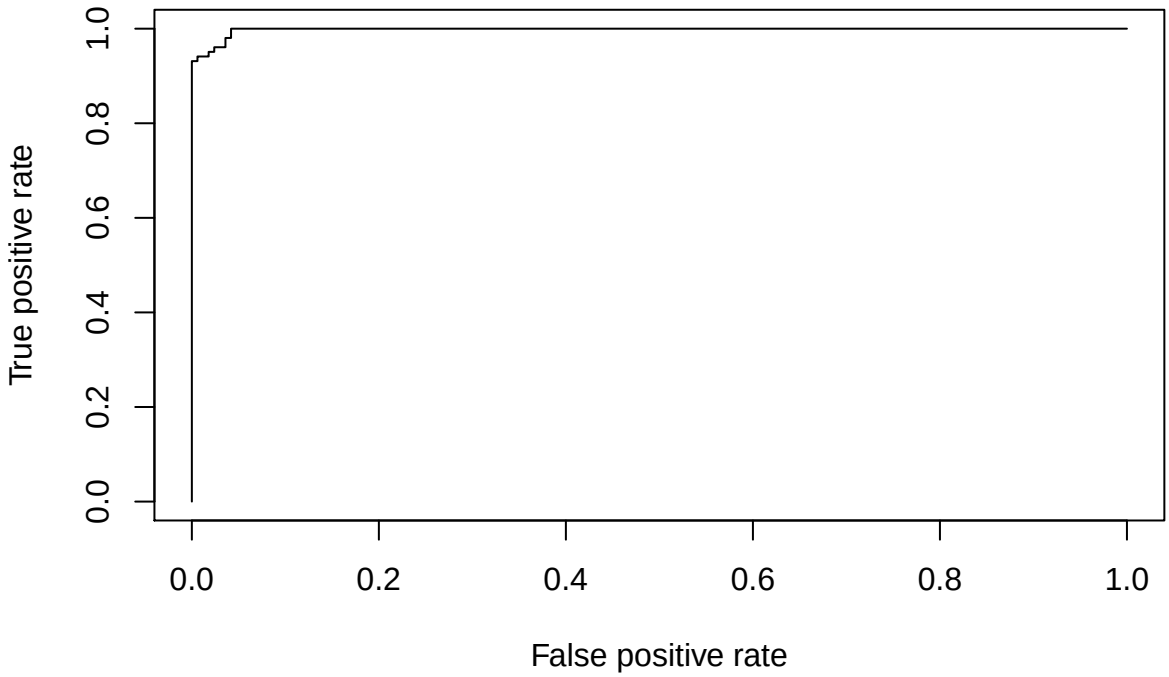
```
## - concave.points_mean  1   91.135 103.135
## - area_worst           1  107.600 119.600
##
## Step:  AIC=90.49
## diagnosis ~ concave.points_mean + compactness_mean + concavity_worst +
##     area_worst + area_mean
##
##                       Df Deviance     AIC
## <none>                     78.487  90.487
## - compactness_mean     1   81.363  91.363
## - concavity_worst      1   83.191  93.191
## - area_mean            1   92.795 102.795
## - concave.points_mean  1   93.044 103.044
## - area_worst           1  114.282 124.282


##
## Call:  glm(formula = diagnosis ~ concave.points_mean + compactness_mean +
##     concavity_worst + area_worst + area_mean, family = "binomial",
##     data = bc, subset = train)
##
## Coefficients:
##         (Intercept)  concave.points_mean     compactness_mean
##            -9.67833             95.84676            -23.15127
##     concavity_worst           area_worst            area_mean
##             5.25036              0.02530             -0.02329
##
## Degrees of Freedom: 299 Total (i.e. Null);  294 Residual
## Null Deviance:       394.3
## Residual Deviance: 78.49     AIC: 90.49
```

```
library("ROCR")
# fit the final model after aic
glm.fit <- glm(diagnosis ~ concave.points_mean + compactness_mean +
    concavity_worst + area_worst + area_mean, family = "binomial",
    data = bc, subset = train)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
glm.prob <- predict(object = glm.fit, test, type = 'response')
pred <- prediction(glm.prob, test$diagnosis)
perf <- performance(pred,"tpr","fpr")
plot(perf)
```




```
auc.perf <- performance(pred, measure = "auc")
print(auc.perf@y.values)
```

```
## [[1]]
## [1] 0.998004
```

```
glm.pred = ifelse(glm.prob> 0.5, '1', '0')
mean(glm.pred == test$diagnosis)
```

```
## [1] 0.9628253
```

```
table(glm.pred, test$diagnosis)
```

```
##
## glm.pred   0   1
##        0 161   4
##        1   6  98
```

random forest

```
# Random Forest
set.seed(2)
rf.bc <- randomForest(diagnosis ~ ., data = bc, subset = train, ntree = 500)
rf.bc
```

```
##
## Call:
```

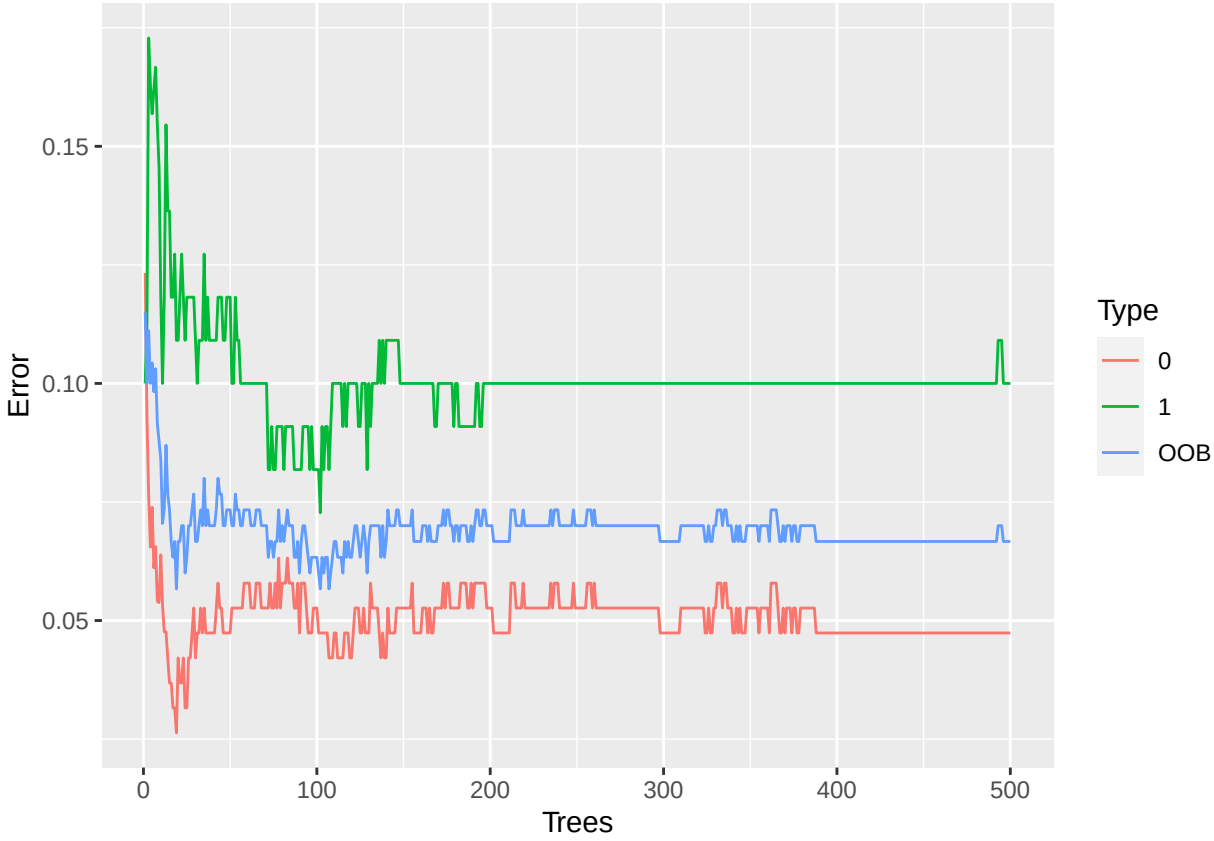
```
##  randomForest(formula = diagnosis ~ ., data = bc, ntree = 500,      subset = train)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 6.67%
## Confusion matrix:
##     0  1 class.error
## 0 181  9  0.04736842
## 1  11 99  0.10000000
```

dataframe format the error rate

```
oob.error.data <- data.frame(
  Trees = rep(1:nrow(rf.bc$err.rate), times = 3),
  Type = rep(c("OOB","0","1"), each = nrow(rf.bc$err.rate)),
  Error = c(rf.bc$err.rate[,"OOB"],
            rf.bc$err.rate[,"0"],
            rf.bc$err.rate[,"1"])
)
```

error rate visualization

```
ggplot(data = oob.error.data, aes(x = Trees, y = Error)) +
  geom_line(aes(color = Type))
```



number of variables

Then we create a loop that tests different numbers of variables at each step

```
set.seed(2)
oob.values <- vector(length = 10)
for(i in 1:10){
  temp.rf <- randomForest(diagnosis ~ ., data = bc, subset = train, mtry = i, ntree = 1000 )
  oob.values[i] <- temp.rf$err.rate[nrow(temp.rf$err.rate),1]
}
```

```
oob.values
```

```
##  [1] 0.06666667 0.06333333 0.07000000 0.06333333 0.07000000 0.06666667
##  [7] 0.07000000 0.07333333 0.06666667 0.06666667
```

```
# refit model with best argument mtry = 4
set.seed(2)
rf.bc <- randomForest(diagnosis ~ ., data = bc, subset = train, ntree = 500, mtry = 4, proximity = TRUE)
rf.bc
```

```
##
## Call:
##  randomForest(formula = diagnosis ~ ., data = bc, ntree = 500,      mtry = 4, proximity = TRUE, subs
##                Type of random forest: classification
```

```
##                       Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 6.67%
## Confusion matrix:
##     0  1 class.error
## 0 181  9  0.04736842
## 1  11 99  0.10000000
```

prediction

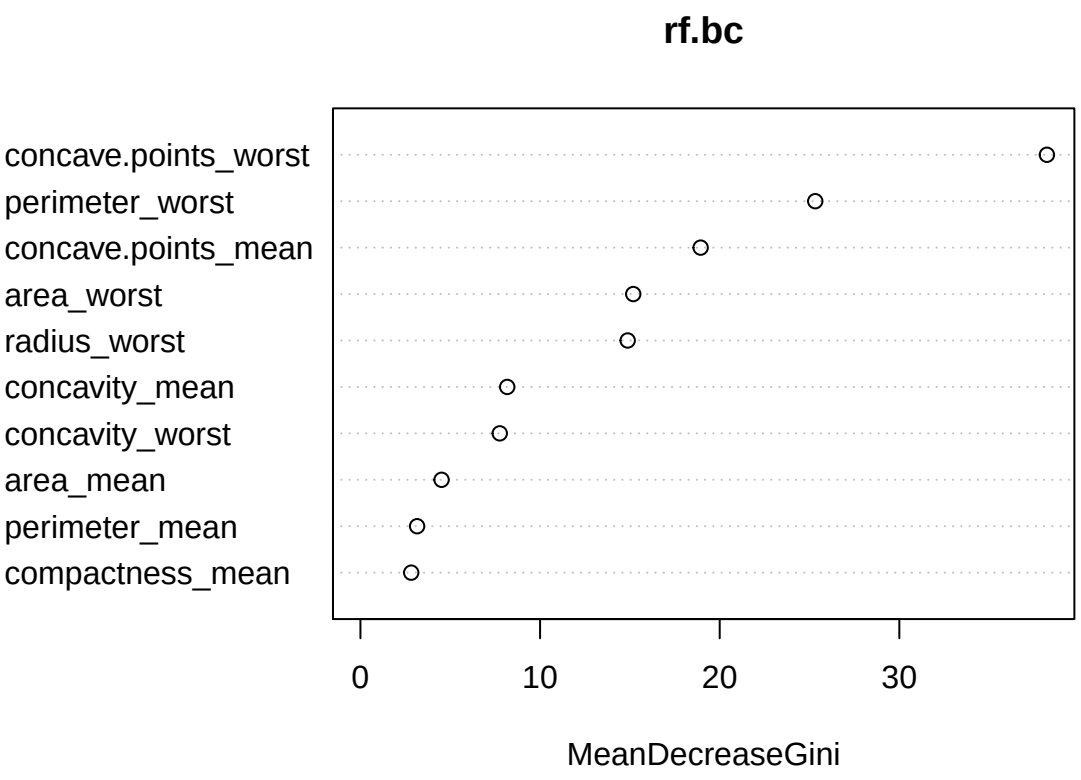
```
yhat.rf <- predict(rf.bc, newdata = test)
mean(yhat.rf==test$diagnosis)
```

```
## [1] 0.9702602
```

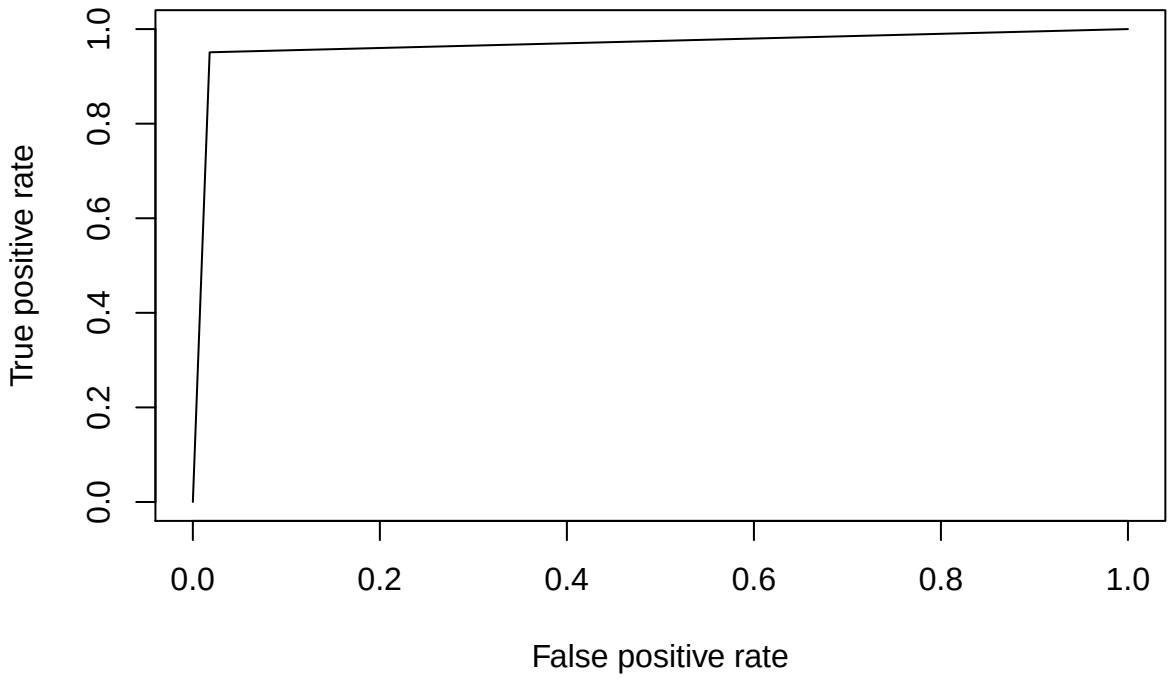
```
table(yhat.rf, test$diagnosis)
```

```
##
## yhat.rf   0   1
##       0 164   5
##       1   3  97
```

```
varImpPlot(rf.bc)
```



```
glm.prob <- as.numeric(predict(object = rf.bc, test, type = 'response'))
pred <- prediction(glm.prob, test$diagnosis)
perf <- performance(pred,"tpr","fpr")
plot(perf)
```




```
auc.perf <- performance(pred, measure = "auc")
print(auc.perf@y.values)
```

```
## [[1]]
## [1] 0.9665082
```

reference

https://github.com/StatQuest/pca_demo/blob/master/pca_demo.R

(https://www.kaggle.com/jaehoonmoon/pca-logistic-regression-auc-99)