Final Project

Shiloh Bradley

7/9/2020

Real Estate Data

```
df <- read.csv("Real estate valuation data set.csv") %>%
      mutate(N.Cstores = as.numeric(N.Cstores),
             Y = normalize(Y)) %>%
      select(-No)

# df <- apply(df, 2, normalize)
# summary(df.Z)
```

```
n.NA <- colSums(is.na(df))
n.NA ## there are none
```

```
##       Date        Age   Dist.MRT  N.Cstores   latitude  longitude       Dist
##          0          0          0          0          0          0          0
##          Y
##          0
```

```
set.seed(1197317)

M <- trunc(.25 * nrow(df))

holdout <- sample(1:nrow(df), M, replace = F)
df.train <- df[-holdout, ]
df.test <- df[holdout, ]
dim(df.train) ## 311  8
```

```
## [1] 311   8
```

```
dim(df.test)  ## 103  8
```

```
## [1] 103   8
```

(a) Fit MLR model to Y as a function of the potential predictors Age, Dist.MRT, N.Cstores,

Dist. Verify assumptions, using the training set. Compute RMSE and R^2 for both training and test sets.

```
MLR1 <- glm(Y ~ Age + Dist.MRT + N.Cstores + Dist,
           family = binomial("logit"),
           data = df.train)

summary(MLR1)
```

```
##
## Call:
## glm(formula = Y ~ Age + Dist.MRT + N.Cstores + Dist, family = binomial("logit"),
##     data = df.train)
##
## Deviance Residuals:
##       Min        1Q    Median        3Q       Max
```

```
## -0.85961  -0.10659  -0.02240   0.07164   1.52430
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.1248860  2.1803463   0.516    0.606
## Age         -0.0117595  0.0111896  -1.051    0.293
## Dist.MRT    -0.0002907  0.0002143  -1.357    0.175
## N.Cstores    0.0361438  0.0536109   0.674    0.500
## Dist        -0.2014883  0.2503063  -0.805    0.421
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 25.961  on 310  degrees of freedom
## Residual deviance: 10.625  on 306  degrees of freedom
## AIC: 230.52
##
## Number of Fisher Scoring iterations: 5
```

```r
vif(MLR1)
```

```
##       Age  Dist.MRT N.Cstores      Dist
##  1.013822  2.072922  1.470591  1.802713
```

(b) Fit SVM model to Y as a function of the potential predictors Age, Dist.MRT, N.Cstores, Dist. Verify assumptions, using the training set. Compute RMSE and R^2 for both training and test sets.

```r
svm1 <- svm(formula = Y ~ Age + Dist.MRT + N.Cstores + Dist,
            data = df.train,
            kernel = "radial",
            probability = TRUE)
print(svm1)
```

```
##
## Call:
## svm(formula = Y ~ Age + Dist.MRT + N.Cstores + Dist, data = df.train,
##     kernel = "radial", probability = TRUE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.25
##     epsilon:  0.1
##
## Sigma:  0.3749404
##
##
## Number of Support Vectors:  250
```

```r
names(svm1)
```

```
##  [1] "call"             "type"             "kernel"
##  [4] "cost"             "degree"           "gamma"
##  [7] "coef0"            "nu"               "epsilon"
## [10] "sparse"           "scaled"           "x.scale"
```

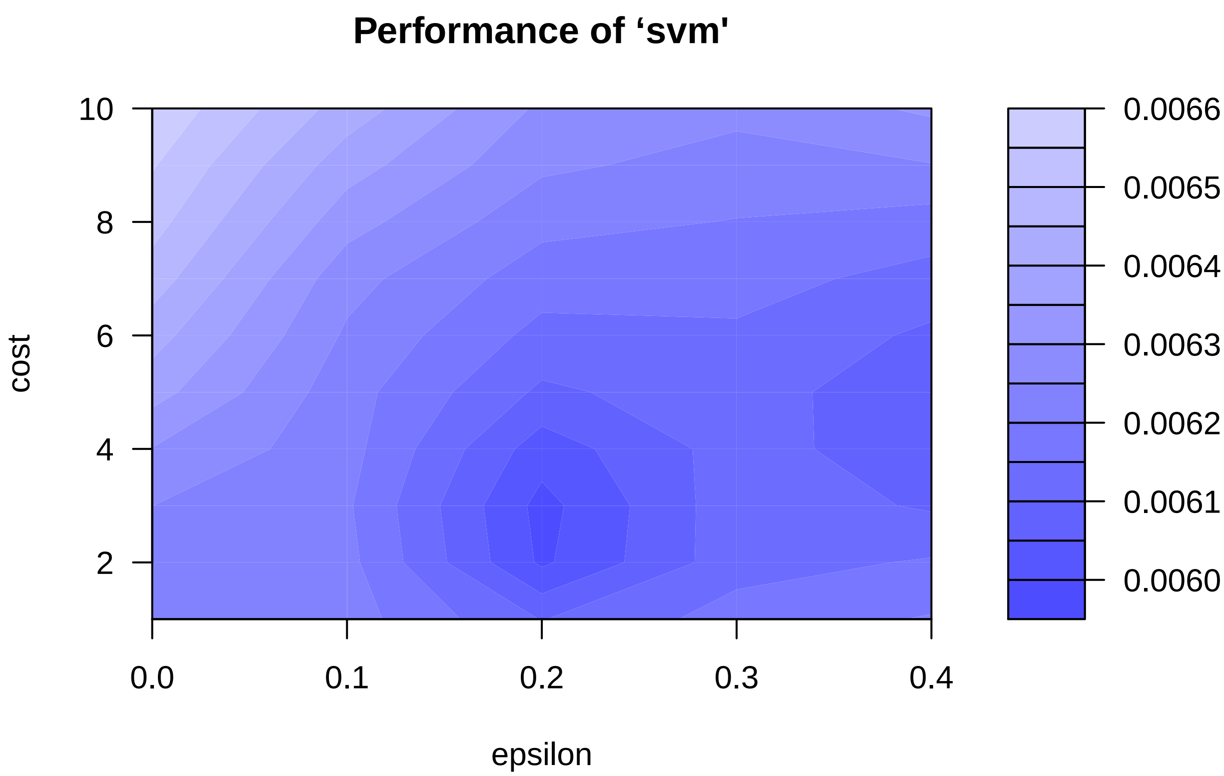
```
## [13] "y.scale"         "nclasses"        "levels"
## [16] "tot.nSV"         "nSV"             "labels"
## [19] "SV"              "index"           "rho"
## [22] "compprob"        "probA"           "probB"
## [25] "sigma"           "coefs"           "na.action"
## [28] "fitted"          "decision.values" "residuals"
## [31] "terms"
```

```r
# SVM performance can be improved further by tuning the SVM
# perform a grid search to tune(optimize) SVM HYPERPARAMETERS
tune.svm <- tune(svm,
                 Y ~ Age + Dist.MRT + N.Cstores + Dist,
                 kernel = "radial",
                 data = df.train,
                 ranges = list(epsilon = seq(0, 0.4, 0.1),
                               cost = c(1:10))
                 )

print(tune.svm)
```

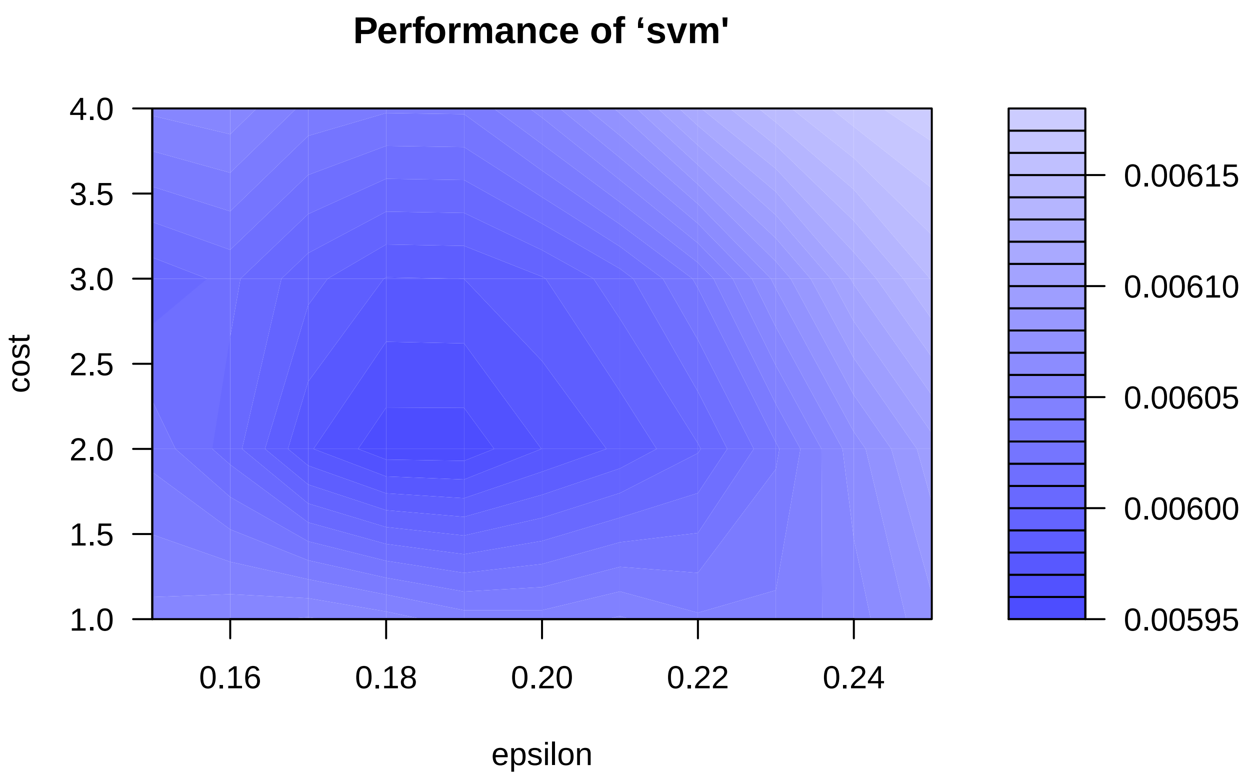
```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  epsilon cost
##      0.2    3
##
## - best performance: 0.005983129
```

```r
# Draw the tuning graph
plot(tune.svm)
```




```r
tune.svm2 <- tune(svm,
                  Y ~ Age + Dist.MRT + N.Cstores + Dist,
                  kernel = "radial",
                  data = df.train,
                  ranges = list(epsilon = seq(0.15, 0.25, 0.01),
                                cost = c(1:4))
                  )

plot(tune.svm2)
```




```
tune.svm2
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  epsilon cost
##     0.19    2
##
## - best performance: 0.005953638
```

```
svmF <- svm(formula = Y ~ Age + Dist.MRT + N.Cstores + Dist,
            kernel = "radial",
            data = df.train,
            ranges = list(epsilon = seq(0.15, 0.25, 0.01),
                          cost = c(1:4))
            )

svmF
```

```
##
## Call:
## svm(formula = Y ~ Age + Dist.MRT + N.Cstores + Dist, data = df.train,
##     kernel = "radial", ranges = list(epsilon = seq(0.15, 0.25,
##         0.01), cost = c(1:4)))
##
##
## Parameters:
##    SVM-Type:  eps-regression
```

```
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.25
##     epsilon:  0.1
##
##
## Number of Support Vectors:  250
```

```r
Y.train <- df.train$Y
Y.test <- df.test$Y
Yhat.train_svm <- svmF$fitted
Yhat.test_svm <- predict(svmF, df.test)
RMSE.train_svm <- RMSE(Y.train, Yhat.train_svm)
RMSE.test_svm <- RMSE(Y.test, Yhat.test_svm)
df.RMSE_svm <- rbind.data.frame(RMSE.train_svm, RMSE.test_svm)
colnames(df.RMSE_svm) <- c("svm.R_Square", "svm.RMSE")
rownames(df.RMSE_svm) <- c("train", "test")
df.RMSE_svm
```

```
##       svm.R_Square   svm.RMSE
## train    0.6657473 0.07343889
## test     0.7908167 0.05456337
```

(c) Compare the results for MLR and SVM for both training and test sets.

Bank data

```r
df <- read.csv("bank.1.csv")
```

```r
n.NA <- colSums(is.na(df))
n.NA ## there are none
```

```
##       age       job   marital education   default   balance   housing
##         0         0         0         0         0         0         0
##      loan   contact       day     month  duration  campaign     pdays
##         0         0         0         0         0         0         0
##  previous  poutcome         y
##         0         0         0
```

```r
set.seed(1197317)

M <- trunc(.25 * nrow(df))

holdout <- sample(1:nrow(df), M, replace = F)
df.train <- df[-holdout, ]
df.test <- df[holdout, ]
dim(df.train) ## 3391   17
```

```
## [1] 3391   17
```

```r
dim(df.test)  ## 1130   17
```

```
## [1] 1130   17
```

(a) Fit a logistic regression model to the response y=yes, and compute its PRF1 values.

(Note: The LR model must have all VIF < 5, and all predictors must be significant at test size 0.05).

```
LR1 <- glm(y ~ .,
          family = binomial("logit"),
          data = df.train)

# summary(LR1)

vif(LR1)
```

```
##               GVIF Df GVIF^(1/(2*Df))
## age       2.359744  1        1.536146
## job       5.515737 11        1.080710
## marital   1.452938  2        1.097897
## education 2.482835  3        1.163656
## default   1.034320  1        1.017015
## balance   1.080682  1        1.039559
## housing   1.485906  1        1.218978
## loan      1.064807  1        1.031895
## contact   2.029508  2        1.193570
## day       1.349727  1        1.161778
## month     4.960028 11        1.075506
## duration  1.166616  1        1.080100
## campaign  1.154314  1        1.074390
## pdays     3.908706  1        1.977045
## previous  1.936280  1        1.391503
## poutcome  5.676161  3        1.335598
```

```
LR2 <- glm(y ~ . - poutcome,
          family = binomial("logit"),
          data = df.train)

# summary(LR2)

vif(LR2)
```

```
##               GVIF Df GVIF^(1/(2*Df))
## age       2.346173  1        1.531722
## job       5.339297 11        1.079114
## marital   1.457750  2        1.098805
## education 2.448726  3        1.160976
## default   1.029865  1        1.014823
## balance   1.077221  1        1.037893
## housing   1.475705  1        1.214786
## loan      1.065843  1        1.032397
## contact   1.976901  2        1.185758
## day       1.347353  1        1.160755
## month     4.665641 11        1.072519
## duration  1.169750  1        1.081550
## campaign  1.146167  1        1.070592
## pdays     1.663859  1        1.289907
## previous  1.485622  1        1.218861
```

```r
LR3 <- glm(y ~ . - poutcome - job,
           family = binomial("logit"),
           data = df.train)

# summary(LR3)

vif(LR3)
```

```
##               GVIF Df GVIF^(1/(2*Df))
## age       1.549286  1        1.244703
## marital   1.376402  2        1.083144
## education 1.213404  3        1.032764
## default   1.017930  1        1.008925
## balance   1.068325  1        1.033598
## housing   1.431652  1        1.196517
## loan      1.065901  1        1.032425
## contact   1.968105  2        1.184437
## day       1.350619  1        1.162162
## month     4.176512 11        1.067134
## duration  1.147543  1        1.071234
## campaign  1.144511  1        1.069818
## pdays     1.625474  1        1.274941
## previous  1.459758  1        1.208205
```

```r
LRF <- glm(y ~ . - poutcome - job - education - day - age - default - balance - pdays,
           family = binomial("logit"),
           data = df.train)

summary(LRF)
```

```
##
## Call:
## glm(formula = y ~ . - poutcome - job - education - day - age -
##     default - balance - pdays, family = binomial("logit"), data = df.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.3270  -0.4022  -0.2663  -0.1629   3.1069
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -2.1254744  0.2951954  -7.200 6.01e-13 ***
## maritalmarried   -0.4789204  0.1983023  -2.415 0.015731 *
## maritalsingle    -0.2231886  0.2134808  -1.045 0.295804
## housingyes       -0.5203199  0.1497519  -3.475 0.000512 ***
## loanyes          -0.6667422  0.2242161  -2.974 0.002943 **
## contacttelephone -0.0566919  0.2554795  -0.222 0.824389
## contactunknown   -1.6102497  0.2492985  -6.459 1.05e-10 ***
## monthaug         -0.5448020  0.2690187  -2.025 0.042852 *
## monthdec          0.6157915  0.6398552   0.962 0.335853
## monthfeb         -0.2208852  0.3114562  -0.709 0.478199
## monthjan         -1.0549573  0.4472436  -2.359 0.018334 *
## monthjul         -1.0055766  0.2730814  -3.682 0.000231 ***
## monthjun          0.2152349  0.3271735   0.658 0.510627
```

```
## monthmar           1.7110878  0.3959691   4.321 1.55e-05 ***
## monthmay          -0.4497763  0.2478364  -1.815 0.069553 .
## monthnov          -1.0179949  0.3039343  -3.349 0.000810 ***
## monthoct           1.3968167  0.3627696   3.850 0.000118 ***
## monthsep           1.2127110  0.4249210   2.854 0.004318 **
## duration           0.0043509  0.0002364  18.402  < 2e-16 ***
## campaign          -0.0703655  0.0302223  -2.328 0.019898 *
## previous           0.0976797  0.0283497   3.446 0.000570 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2408.0  on 3390  degrees of freedom
## Residual deviance: 1699.7  on 3370  degrees of freedom
## AIC: 1741.7
##
## Number of Fisher Scoring iterations: 6
```

(b) Fit the default random forest model to the response, and compare the PRF1 values of the LR and the default random forest model.

```r
features <- setdiff(names(df.train), "y")
rf1 <- randomForest(y ~ ., data = df.train)
rf1
```

```
##
## Call:
##  randomForest(formula = y ~ ., data = df.train)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 9.67%
## Confusion matrix:
##       no yes class.error
## no  2906  98  0.03262317
## yes  230 157  0.59431525
```

```r
CM.rf_train <- rf1$confusion
CM.rf_train
```

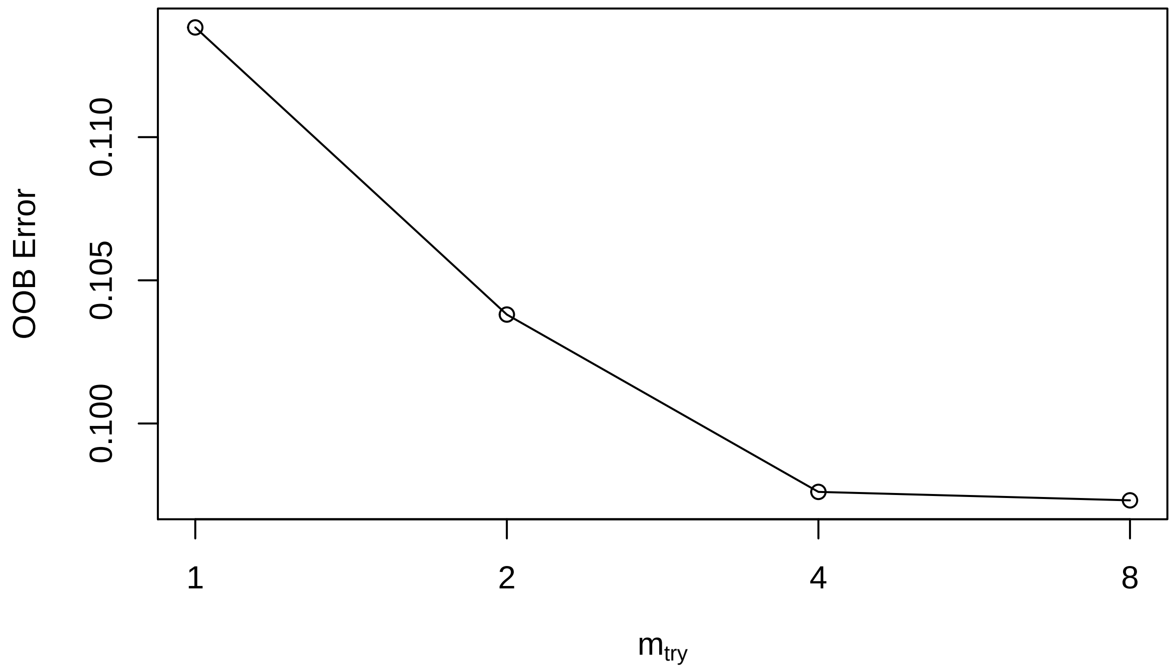
```
##       no yes class.error
## no  2906  98  0.03262317
## yes  230 157  0.59431525
```

```r
OA.rf_train <- sum(diag(CM.rf_train))/sum(CM.rf_train)
```

```r
set.seed(7231)
rf2 <- tuneRF(
  x          = df.train[features],
  y          = factor(df.train$y),
  ntreeTry   = 500,
  mtryStart  = 2,
  stepFactor = 2,
  improve    = 0.01,
```

```
  trace       = FALSE      # to not show real-time progress
)
```

```
## -0.09659091 0.01
## 0.05965909 0.01
## 0.003021148 0.01
```




```
set.seed(11713)
rf2 <- randomForest(y ~ ., mtry = 4, ntree = 500, importance = TRUE, data = df.train)
rf2
```

```
##
## Call:
##  randomForest(formula = y ~ ., data = df.train, mtry = 4, ntree = 500,      importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 10%
## Confusion matrix:
##       no yes class.error
## no  2894 110  0.03661784
## yes  229 158  0.59173127
```

```
CM.rf_train <- rf2$confusion
CM.rf_train
```

```
##       no yes class.error
## no  2894 110  0.03661784
## yes  229 158  0.59173127
```

```
OA.rf_train <- sum(diag(CM.rf_train))/sum(CM.rf_train)
```

```
VI.FL <- as.data.frame(rf2$importance)
names(VI.FL)
```

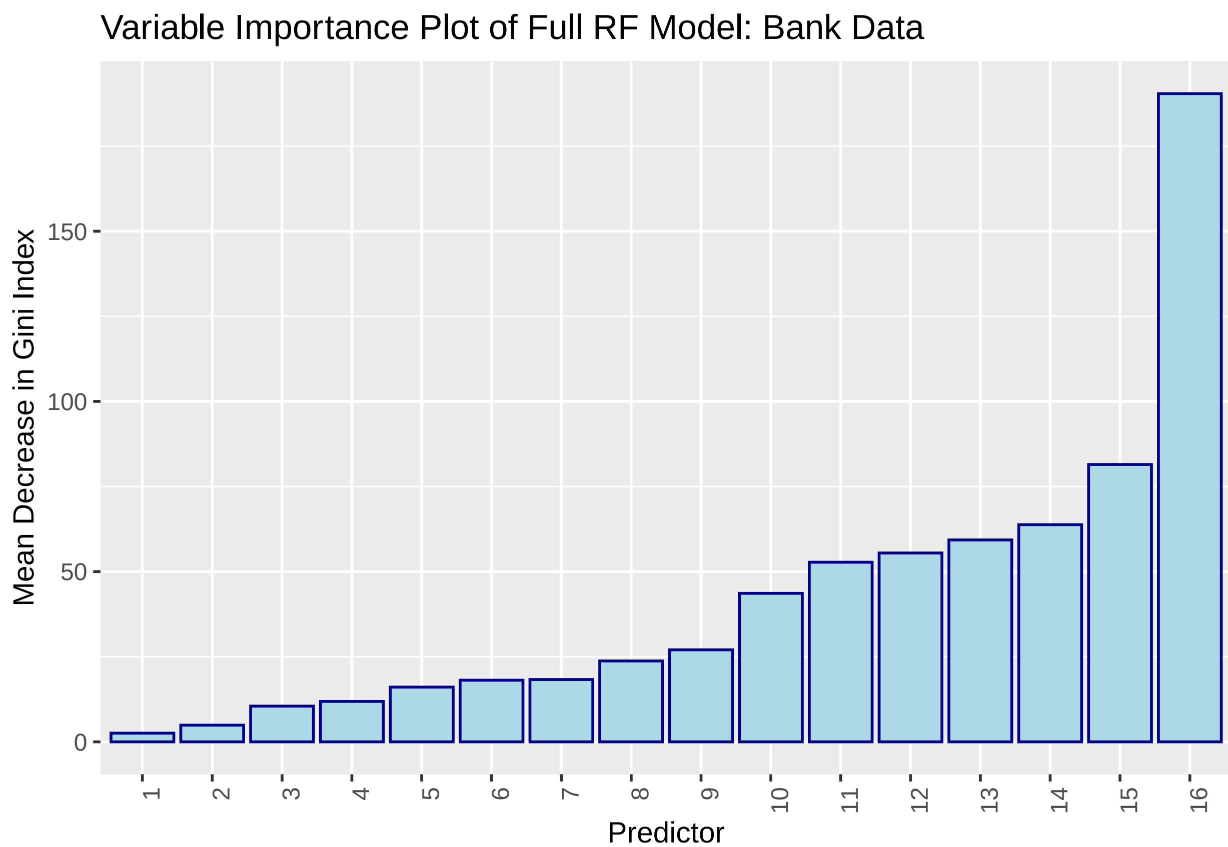
```
## [1] "no"                    "yes"                   "MeanDecreaseAccuracy"
## [4] "MeanDecreaseGini"
```

```
VIFL.sort <- VI.FL %>% arrange(MeanDecreaseGini)

VIFL.sort$X <- rownames(VIFL.sort)
VIFL.sort$X <- factor(VIFL.sort$X, levels = VIFL.sort$X)

p.FL <- ggplot(VIFL.sort, aes(x = X, y = MeanDecreaseGini)) +
  geom_bar(stat = "identity", position = "dodge", fill = "lightblue", color = "darkblue") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ylab("Mean Decrease in Gini Index") +
  xlab("Predictor") +
  ggtitle("Variable Importance Plot of Full RF Model: Bank Data")

p.FL
```




```
set.seed(11713)
rf3 <- randomForest(y ~ . , mtry = 4, ntree = 500, data = df.train)
rf3
```

```
##
## Call:
##  randomForest(formula = y ~ ., data = df.train, mtry = 4, ntree = 500)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
```

```
##         OOB estimate of  error rate: 9.73%
## Confusion matrix:
##       no yes class.error
## no  2904 100  0.03328895
## yes  230 157  0.59431525
```

```
CM.rf_train <- rf3$confusion
CM.rf_train
```

```
##       no yes class.error
## no  2904 100  0.03328895
## yes  230 157  0.59431525
```

```
OA.rf_train <- sum(diag(CM.rf_train))/sum(CM.rf_train)
OA.rf_train
```

```
## [1] 0.9025165
```

```
VI.FL <- as.data.frame(rf3$importance)
names(VI.FL)
```

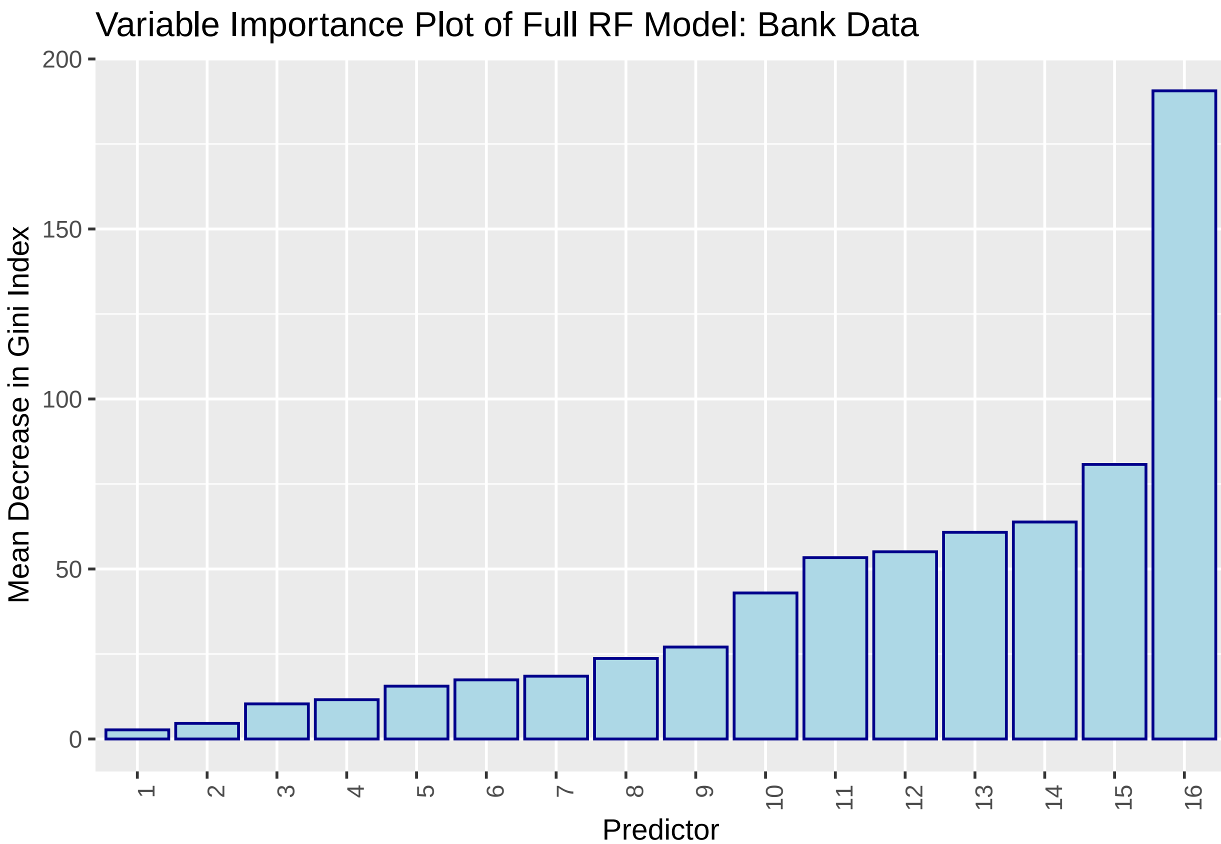
```
## [1] "MeanDecreaseGini"
```

```
VIFL.sort <- VI.FL %>% arrange(MeanDecreaseGini)

VIFL.sort$X <- rownames(VIFL.sort)
VIFL.sort$X <- factor(VIFL.sort$X, levels = VIFL.sort$X)

p.FL <- ggplot(VIFL.sort, aes(x = X, y = MeanDecreaseGini)) +
  geom_bar(stat = "identity", position = "dodge", fill = "lightblue", color = "darkblue") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ylab("Mean Decrease in Gini Index") +
  xlab("Predictor") +
  ggtitle("Variable Importance Plot of Full RF Model: Bank Data")

p.FL
```




```
prf1_train <- PRF1(CM.rf_train)
prf1_train
```

```
## Precision_1    Recall_1        F1_1 Precision_0    Recall_0        F1_0
##        0.93        0.97        0.95        0.93        0.97        0.95
```

```
pred.test <- predict(rf3, df.test)
CM.test <- table(df.test$y, pred.test)
CM.test
```

```
##      pred.test
##        no yes
##   no  954  42
##   yes  82  52
```

```
prf1_test <- PRF1(CM.test)
prf1_test
```

```
## Precision_1    Recall_1        F1_1 Precision_0    Recall_0        F1_0
##        0.92        0.96        0.94        0.92        0.96        0.94
```

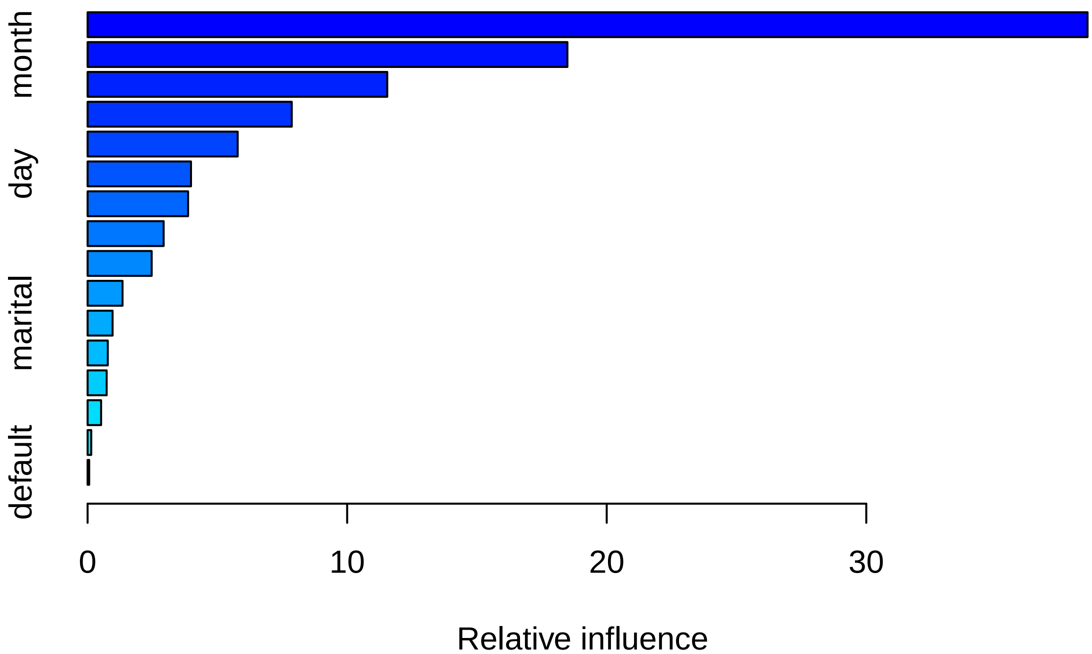
(c) Extra Credit: Fit the default xgboost model to the response, and compare the PRF1 values of the three fitted models.

```
gbm1 <- gbm(
  y ~ .,
  data = df.train,
  distribution = "gaussian",
  n.trees = 10000,
```

```r
  shrinkage = 0.001,
  interaction.depth = 4,
  n.cores = NULL, # will use all cores by default
  verbose = FALSE
  )
# print results
print(gbm1)
```

```
## gbm(formula = y ~ ., distribution = "gaussian", data = df.train,
##     n.trees = 10000, interaction.depth = 4, shrinkage = 0.001,
##     verbose = FALSE, n.cores = NULL)
## A gradient boosted model with gaussian loss function.
## 10000 iterations were performed.
## There were 16 predictors of which 16 had non-zero influence.
```

```r
smreGB1 <- summary(gbm1)
```




```r
str(smreGB1)
```

```
## 'data.frame':    16 obs. of  2 variables:
##  $ var    : Factor w/ 16 levels "age","balance",..: 7 13 15 10 1 5 2 14 16 4 ...
##  $ rel.inf: num  38.53 18.49 11.54 7.87 5.78 ...
```

```r
names(smreGB1)
```

```
## [1] "var"     "rel.inf"
```