Simple document

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(visdat)
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(AppliedPredictiveModeling)
library(pROC)
```

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

##
## Attaching package: 'pROC'

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

```
library(rpart.plot)
```

```
## Loading required package: rpart
```

```
library(vip)
```

```
##
## Attaching package: 'vip'

## The following object is masked from 'package:utils':
##
##     vi
```

```
library(ranger)
library(tidytext)
library(pdp)
```

```
##
## Attaching package: 'pdp'

## The following object is masked from 'package:purrr':
##
##     partial
```

```
library(lime)
```

```
##
## Attaching package: 'lime'

## The following object is masked from 'package:dplyr':
##
##     explain
```

```
ctrl <- trainControl(method = "cv",
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE)

knitr::opts_chunk$set(
  fig.width = 6,
  fig.asp = .6,
  out.width = "90%"
)
```

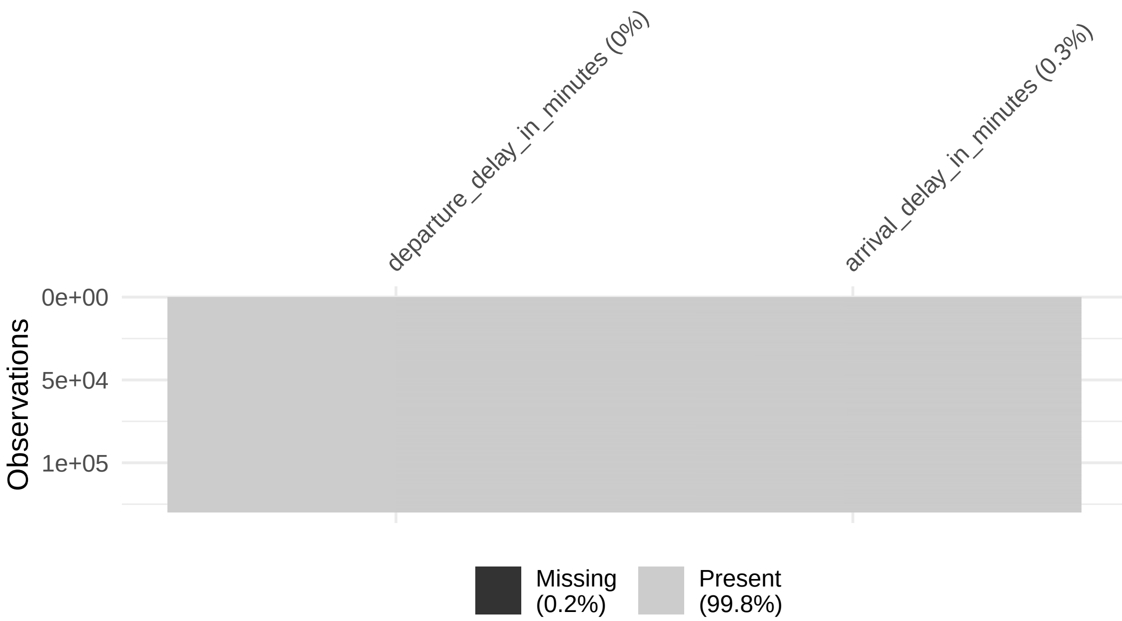
Data pre-process

```
# Import data
dat_raw <- read.csv("./airline.csv")

# Check missing value
sapply(dat_raw, function(x) sum(is.na(x)))
```

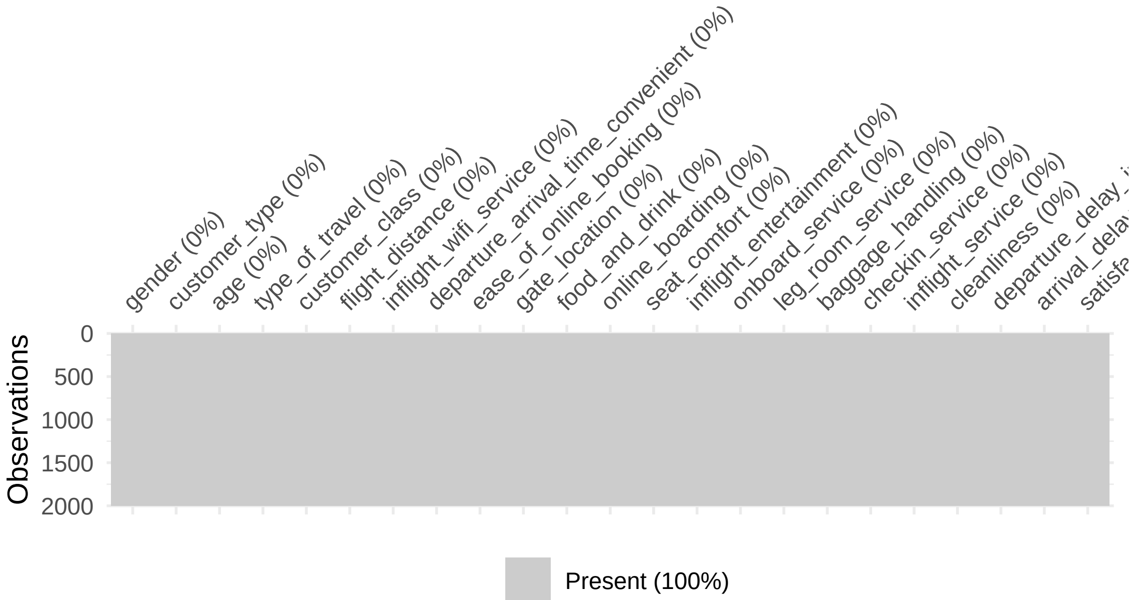
```
##                                  X                             Gender
##                                  0                                  0
##                      customer_type                                age
##                                  0                                  0
##                     type_of_travel                     customer_class
##                                  0                                  0
##                    flight_distance              inflight_wifi_service
##                                  0                                  0
## departure_arrival_time_convenient             ease_of_online_booking
##                                  0                                  0
##                      gate_location                     food_and_drink
##                                  0                                  0
##                    online_boarding                       seat_comfort
##                                  0                                  0
##             inflight_entertainment                    onboard_service
##                                  0                                  0
##                   leg_room_service                   baggage_handling
##                                  0                                  0
```

```
##                   checkin_service                  inflight_service
##                                 0                                 0
##                       cleanliness        departure_delay_in_minutes
##                                 0                                 0
##          arrival_delay_in_minutes                      satisfaction
##                               393                                 0
```

```
# data clean
dat <- dat_raw %>%
  janitor::clean_names() %>%
  select(-1) %>%
  mutate(satisfaction = recode(satisfaction,
                          "satisfied" = "yes",
                          "neutral or dissatisfied" = "no"))


# deal with missing values
deal_mis <- dat[, 21:22]
bagImp = preProcess(deal_mis, method = "bagImpute")
dat = predict(bagImp, dat)
vis_miss(deal_mis)
```




```
# sample data
set.seed(1234)
dat <- dat[sample(1:nrow(dat), 2000, replace = FALSE), ]
vis_miss(dat) ## check
```

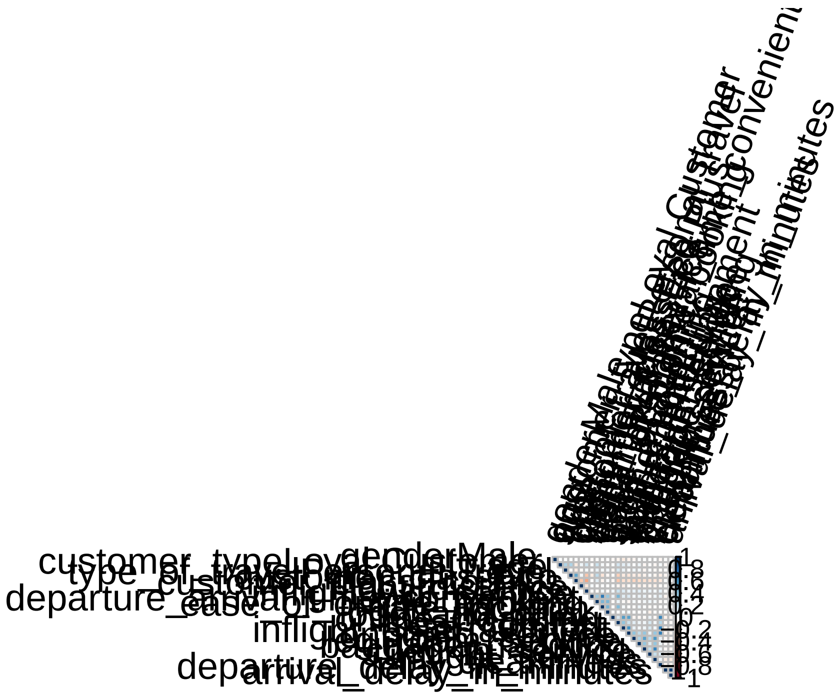



```r
# --- Split data ---
set.seed(1234)
trRow <- createDataPartition(dat$satisfaction, p = 0.8, list = F)

# Train data
train <- dat[trRow, ]
x_train <- model.matrix(satisfaction ~., train)[,-1]
y_train <- train$satisfaction

# Test data
test <- dat[-trRow, ]
x_test <- model.matrix(satisfaction ~., test)[,-1]
y_test <- test$satisfaction
```

EDA

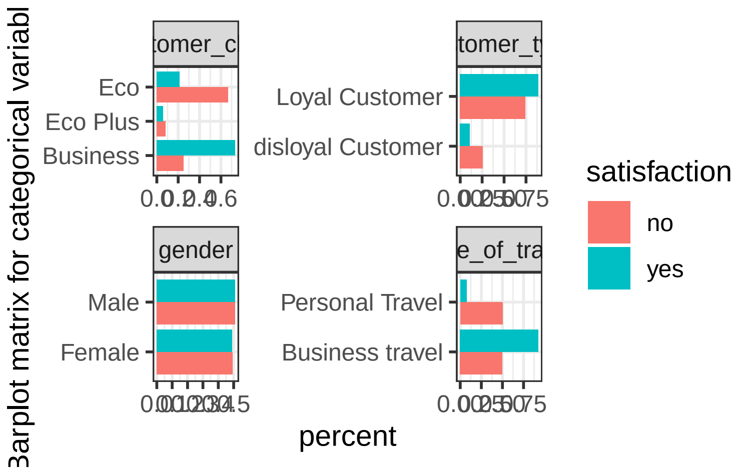
```r
# Correlation plot
corrplot(cor(x_train),
         method = "circle",
         type = "upper",
         tl.col = "black",
         tl.cex = 1.2,
         tl.srt = 70)
```

```
## Warning in corrplot(cor(x_train), method = "circle", type = "upper", tl.col =
## "black", : Not been able to calculate text margin, please try again with a clean
## new empty window using {plot.new(); dev.off()} or reduce tl.cex
```

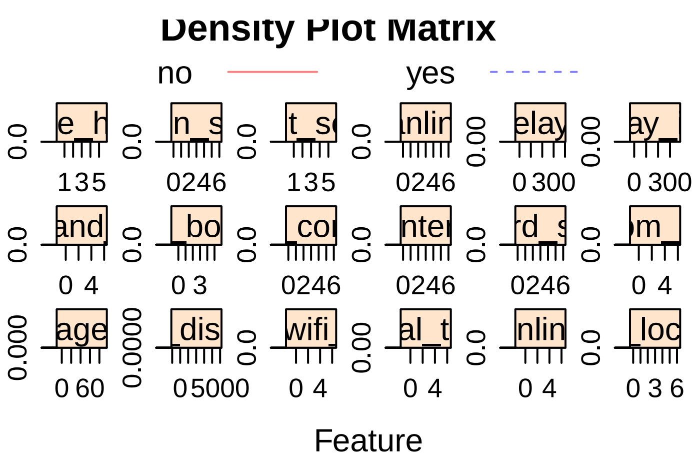


```r
# Barplot matrix for categorical variables
train %>%
  select(1:2, 4:5, 23) %>%
  pivot_longer(-5,
               names_to = "variable",
               values_to = "value") %>%
  group_by(variable, value, satisfaction) %>%
  summarize(num = n()) %>%
  ungroup() %>%
  group_by(variable, satisfaction) %>%
  mutate(percent = num / sum(num),
         indicator = case_when(value == "Eco" ~ 3,
                               value == "Eco Plus" ~ 2,
                               value == "Business" ~ 1,
                               TRUE ~ 0)) %>%
  ggplot(aes(x = reorder_within(value, indicator, variable),
             y = percent, fill = satisfaction)) +
  geom_col(position = "dodge") +
  xlab("Barplot matrix for categorical variables") +
  coord_flip() +
  scale_x_reordered() +
  facet_wrap(~ variable, scales = "free") + theme_bw()
```

```
## `summarise()` has grouped output by 'variable', 'value'. You can override using the `.groups` argumen
```




```
# Density plot matrix
theme1 <- transparentTheme(trans = .5)
trellis.par.set(theme1)

plt_feature <-
  featurePlot(x = x_train[ , c(-1, -2, -4, -5, -6)],
              y = as.factor(y_train),
              plot = "density",
              scales = list(x = list(relation = "free"),
                            y = list(relation = "free")),
              pch = "|", auto.key = list(columns = 2))
update(plt_feature, main = "Density Plot Matrix")
```



Model fitting

Logistic regression

```
set.seed(1234)
model.glm <- train(x = x_train,
                   y = y_train,
                   method = "glm",
                   metric = "ROC",
                   trControl = ctrl)
```

```r
# Test AUC and Misclassification error rate
pred_glm_auc <- predict(model.glm, newdata = x_test, type = "prob")[,2]
roc(y_test, pred_glm_auc)$auc[1]
```

```
## Setting levels: control = no, case = yes
## Setting direction: controls < cases
## [1] 0.8923937
```

```r
# Confusion matrix
pred_glm <- predict(model.glm, newdata = x_test)
confusionMatrix(data = as.factor(pred_glm),
                reference = as.factor(y_test))
```

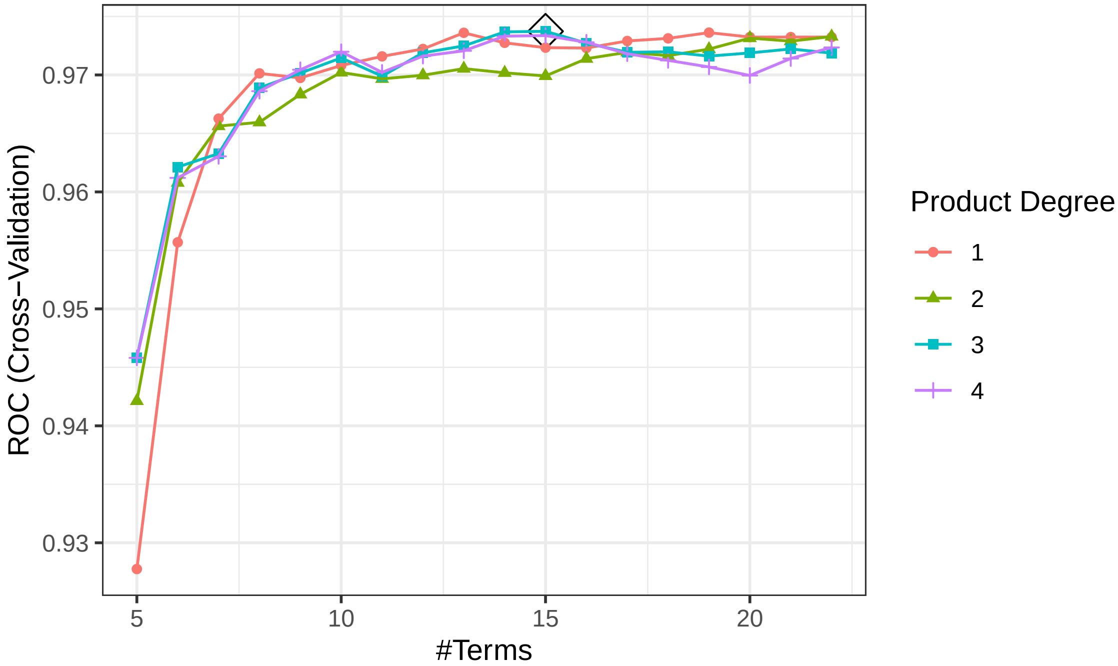
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  no yes
##        no  201  36
##        yes  24 139
##
##                Accuracy : 0.85
##                  95% CI : (0.8112, 0.8835)
##     No Information Rate : 0.5625
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6929
##
##  Mcnemar's Test P-Value : 0.1556
##
##             Sensitivity : 0.8933
##             Specificity : 0.7943
##          Pos Pred Value : 0.8481
##          Neg Pred Value : 0.8528
##              Prevalence : 0.5625
##          Detection Rate : 0.5025
##    Detection Prevalence : 0.5925
##       Balanced Accuracy : 0.8438
##
##        'Positive' Class : no
##
```

MARS

```r
set.seed(1234)
model.mars <- train(x = x_train,
                    y = y_train,
                    method = "earth",
                    tuneGrid = expand.grid(degree = 1:4,
                                           nprune = 5:22),
                    metric = "ROC",
                    trControl = ctrl)

model.mars$bestTune
```

```
##    nprune degree
## 47     15      3
```

```r
ggplot(model.mars, highlight = T) +
  theme_bw()
```




```r
## test auc and misclassification error rate
pred_mars_auc <- predict(model.mars, newdata = x_test, type = "prob")[,2]
roc(y_test, pred_mars_auc)$auc[1]
```

```
## [1] 0.9773587
```

```r
pred_mars <- predict(model.mars, newdata = x_test)
pred.miserror_mars <- 1 - mean(pred_mars == y_test)
pred.miserror_mars
```

```
## [1] 0.0775
```

```r
confusionMatrix(data = as.factor(pred_mars),
                reference = as.factor(y_test))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  no yes
##        no  214  20
##        yes  11 155
##
##                Accuracy : 0.9225
##                  95% CI : (0.8918, 0.9467)
##     No Information Rate : 0.5625
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8416
##
```

```
##  Mcnemar's Test P-Value : 0.1508
##
##             Sensitivity : 0.9511
##             Specificity : 0.8857
##          Pos Pred Value : 0.9145
##          Neg Pred Value : 0.9337
##              Prevalence : 0.5625
##          Detection Rate : 0.5350
##    Detection Prevalence : 0.5850
##       Balanced Accuracy : 0.9184
##
##        'Positive' Class : no
##
```

LDA

```
set.seed(1234)
model.lda <- train(x = x_train,
                   y = y_train,
                   method = "lda",
                   metric = "ROC",
                   trControl = ctrl)

## test auc and misclassification error rate
pred_lda_auc <- predict(model.lda, newdata = x_test, type = "prob")[,2]
roc(y_test, pred_lda_auc)$auc[1]
```

```
## Setting levels: control = no, case = yes

## Setting direction: controls < cases

## [1] 0.8954413
```

```
pred_lda <- predict(model.lda, newdata = x_test)
pred.miserror_lda <- 1 - mean(pred_lda == y_test)
pred.miserror_lda
```

```
## [1] 0.1525
```

```
confusionMatrix(data = as.factor(pred_lda),
                reference = as.factor(y_test))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  no yes
##        no  201  37
##        yes  24 138
##
##                Accuracy : 0.8475
##                  95% CI : (0.8085, 0.8813)
##     No Information Rate : 0.5625
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6876
##
```

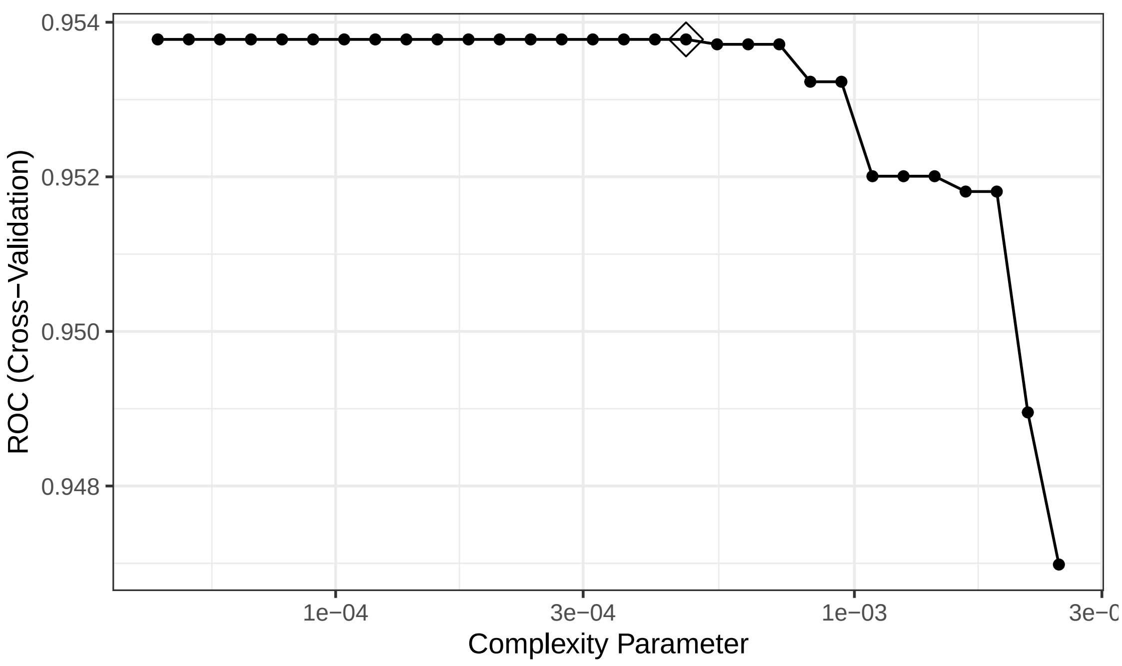
```
##   Mcnemar's Test P-Value : 0.1244
##
##              Sensitivity : 0.8933
##              Specificity : 0.7886
##           Pos Pred Value : 0.8445
##           Neg Pred Value : 0.8519
##               Prevalence : 0.5625
##           Detection Rate : 0.5025
##     Detection Prevalence : 0.5950
##        Balanced Accuracy : 0.8410
##
##         'Positive' Class : no
##
```

Classification Tree

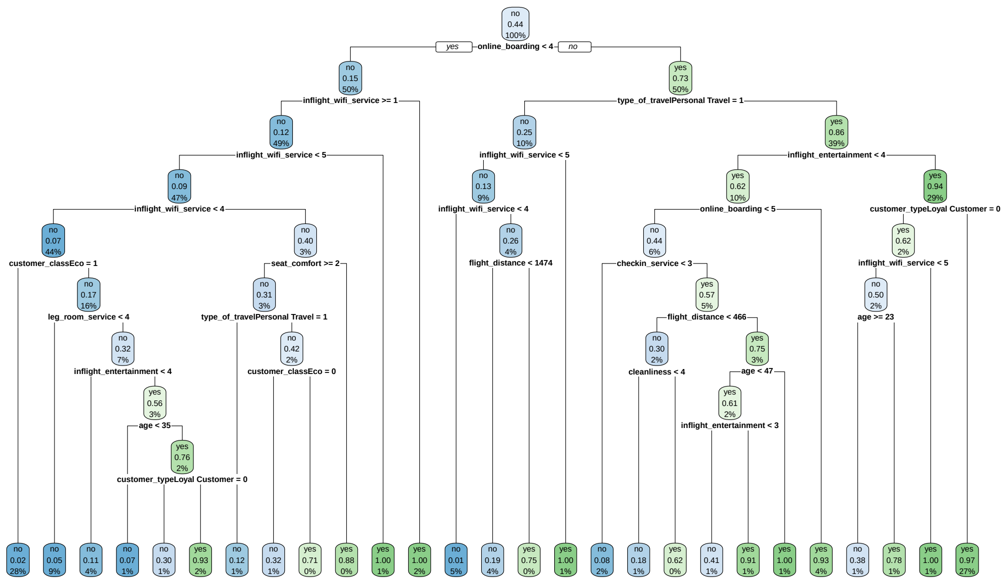
```
set.seed(1234)
model.tree <- train(x_train,
                    y_train,
                    method = "rpart",
                    tuneGrid = data.frame(cp = exp(seq(-10, -6, length = 30))),
                    trControl = ctrl,
                    metric = "ROC")

model.tree$bestTune
```

```
##               cp
## 18 0.0004735882
```

```
ggplot(model.tree, highlight = TRUE) +
  scale_x_continuous(trans = scales::log_trans(),
                     breaks = scales::log_breaks()) +
  theme_bw()
```



```r
rpart.plot(model.tree$finalModel)
```



```r
## test auc and misclassification error rate
pred_tree_auc <- predict(model.tree, newdata = x_test, type = "prob")[,2]
roc(y_test, pred_tree_auc)$auc[1]
```

```
## [1] 0.9453333
```

```r
pred_tree <- predict(model.tree, newdata = x_test)
pred.miserror_tree <- 1 - mean(pred_tree == y_test)
pred.miserror_tree
```

```
## [1] 0.1025
```

```r
confusionMatrix(data = as.factor(pred_tree),
                reference = as.factor(y_test))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  no yes
##        no  201  17
##        yes  24 158
##
##                Accuracy : 0.8975
##                  95% CI : (0.8635, 0.9254)
##     No Information Rate : 0.5625
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7927
##
##  Mcnemar's Test P-Value : 0.3487
##
##             Sensitivity : 0.8933
##             Specificity : 0.9029
##          Pos Pred Value : 0.9220
##          Neg Pred Value : 0.8681
```

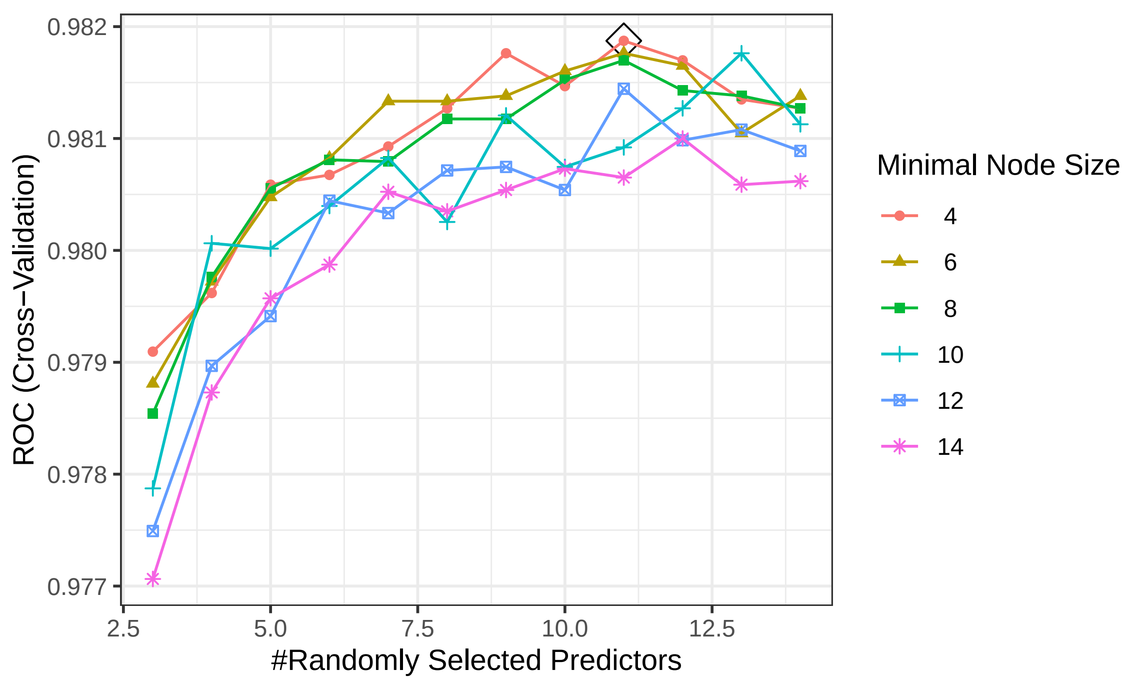
```
##              Prevalence : 0.5625
##          Detection Rate : 0.5025
##    Detection Prevalence : 0.5450
##       Balanced Accuracy : 0.8981
##
##        'Positive' Class : no
##
```

Ramdom forests

```
set.seed(1234)
model.rf = train(x_train,
                 y_train,
                 method = "ranger",
                 tuneGrid = expand.grid(mtry = 3:14,
                                        splitrule = "gini",
                                        min.node.size = seq(4, 14, by = 2)),
                 metric = "ROC",
                 trControl = ctrl)

model.rf$bestTune
```

```
##    mtry splitrule min.node.size
## 49   11      gini             4
```

```
ggplot(model.rf, highlight = T) +
  theme_bw()
```




```
## test auc and misclassification error rate
pred_rf_auc <- predict(model.rf, newdata = x_test, type = "prob")[,2]
roc(y_test, pred_rf_auc)$auc[1]
```

```
## Setting levels: control = no, case = yes
```

```
## Setting direction: controls < cases
```

```
## [1] 0.9715175
```

```
pred_rf <- predict(model.rf, newdata = x_test)
pred.miserror_rf <- 1 - mean(pred_rf == y_test)
pred.miserror_rf
```

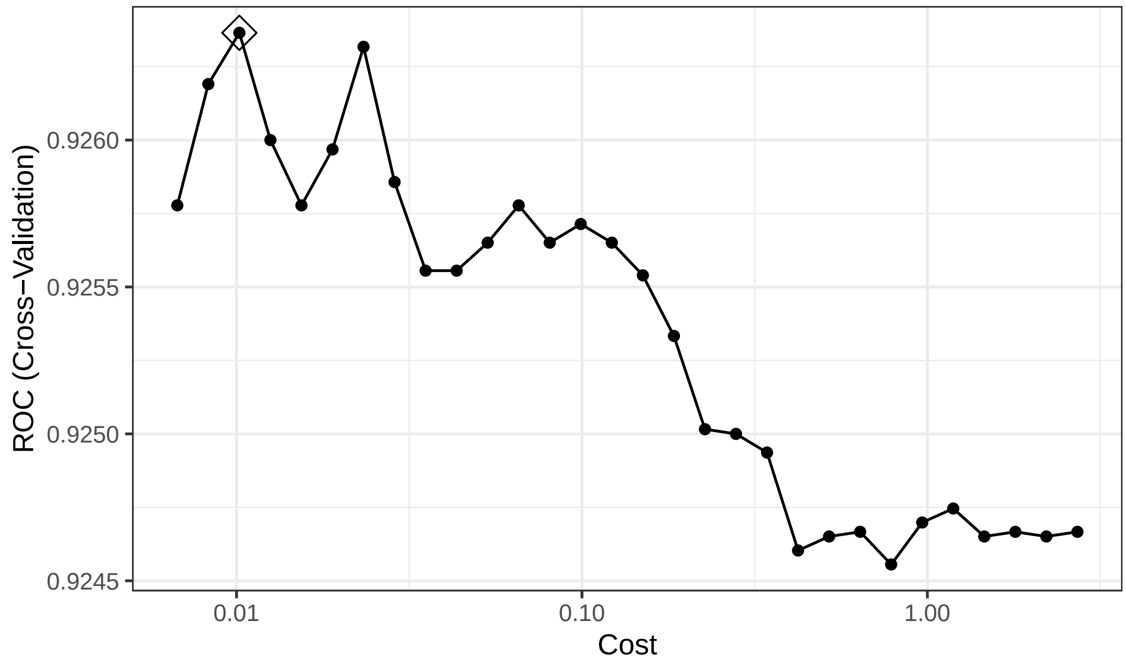
```
## [1] 0.075
```

```
confusionMatrix(data = as.factor(pred_rf),
                reference = as.factor(y_test))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  no yes
##        no  213  18
##        yes  12 157
##
##                Accuracy : 0.925
##                  95% CI : (0.8947, 0.9488)
##     No Information Rate : 0.5625
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.847
##
##  Mcnemar's Test P-Value : 0.3613
##
##             Sensitivity : 0.9467
##             Specificity : 0.8971
##          Pos Pred Value : 0.9221
##          Neg Pred Value : 0.9290
##              Prevalence : 0.5625
##          Detection Rate : 0.5325
##    Detection Prevalence : 0.5775
##       Balanced Accuracy : 0.9219
##
##        'Positive' Class : no
##
```

Fit a support vector classifier (linear kernel)

```
set.seed(1234)
model.svml = train(x_train,
                   y_train,
                   method = "svmLinear",
                   metric = "ROC",
                   tuneGrid = data.frame(C = exp(seq(-5, 1, length = 30))),
                   trControl = ctrl)

ggplot(model.svml, highlight = TRUE) +
  scale_x_continuous(trans = scales::log_trans(),
                     breaks = scales::log_breaks()) +
  theme_bw()
```




```
## test auc and misclassification error rate
pred_svml_auc <- predict(model.svml, newdata = x_test, type = "prob")[,2]
roc(y_test, pred_svml_auc)$auc[1]
```

```
## [1] 0.8928508
```

```
pred_svml <- predict(model.svml, newdata = x_test)
pred.miserror_svml <- 1 - mean(pred_svml == y_test)
pred.miserror_svml
```

```
## [1] 0.1475
```

```
confusionMatrix(data = as.factor(pred_svml),
                reference = as.factor(y_test))
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   no yes
##        no   201  35
##        yes   24 140
##
##                Accuracy : 0.8525
##                  95% CI : (0.8139, 0.8858)
##     No Information Rate : 0.5625
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6982
##
##  Mcnemar's Test P-Value : 0.193
##
##             Sensitivity : 0.8933
##             Specificity : 0.8000
##          Pos Pred Value : 0.8517
##          Neg Pred Value : 0.8537
```

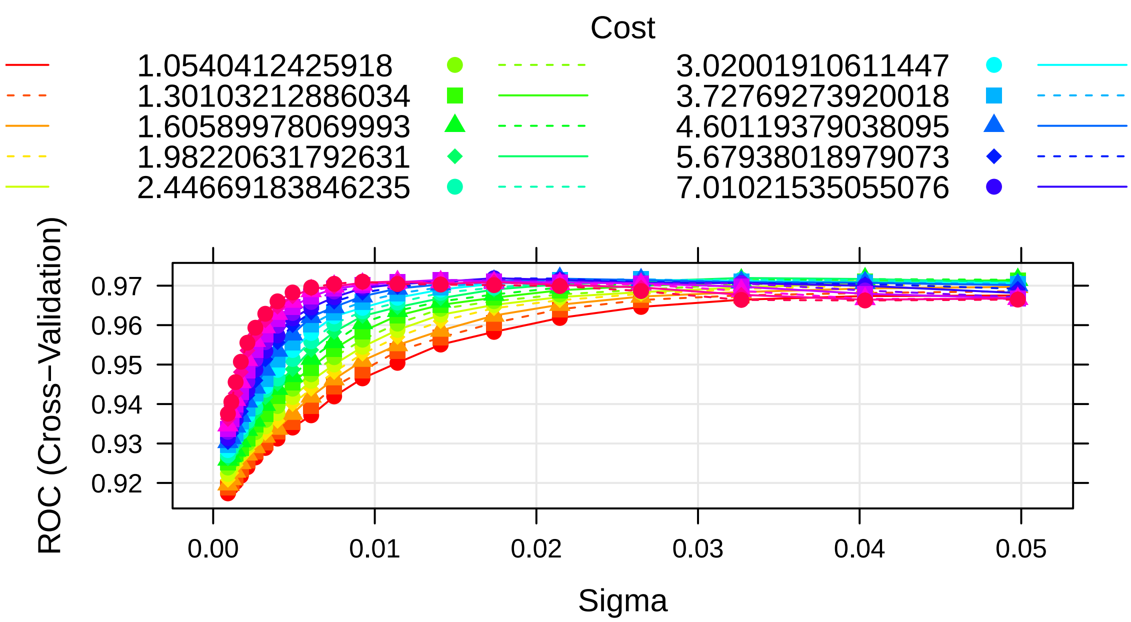
```
##                 Prevalence : 0.5625
##             Detection Rate : 0.5025
##       Detection Prevalence : 0.5900
##          Balanced Accuracy : 0.8467
##
##           'Positive' Class : no
##
```

Fit a support vector machine with a radial kernel

```
set.seed(1234)
model.svmr = train(x_train,
                 y_train,
                 method = "svmRadialSigma",
                 metric = "ROC",
                 tuneGrid = expand.grid(C = exp(seq(-1, 3, length = 20)),
                                        sigma = exp(seq(-7, -3, length = 20))),
                 trControl = ctrl)

model.svmr$bestTune
```

```
##          sigma        C
## 178 0.03267802 1.982206
```

```
myCol<- rainbow(20)
myPar <- list(superpose.symbol = list(col = myCol),
superpose.line = list(col = myCol))
plot(model.svmr, highlight = TRUE, par.settings = myPar)
```




```
## test auc and misclassification error rate
pred_svmr_auc <- predict(model.svmr, newdata = x_test, type = "prob")[,2]
roc(y_test, pred_svmr_auc)$auc[1]
```

```
## Setting levels: control = no, case = yes

## Setting direction: controls < cases
```

```
## [1] 0.9647492
```

```r
pred_svmr <- predict(model.svmr, newdata = x_test)
pred.miserror_svmr <- 1 - mean(pred_svmr == y_test)
pred.miserror_svmr
```

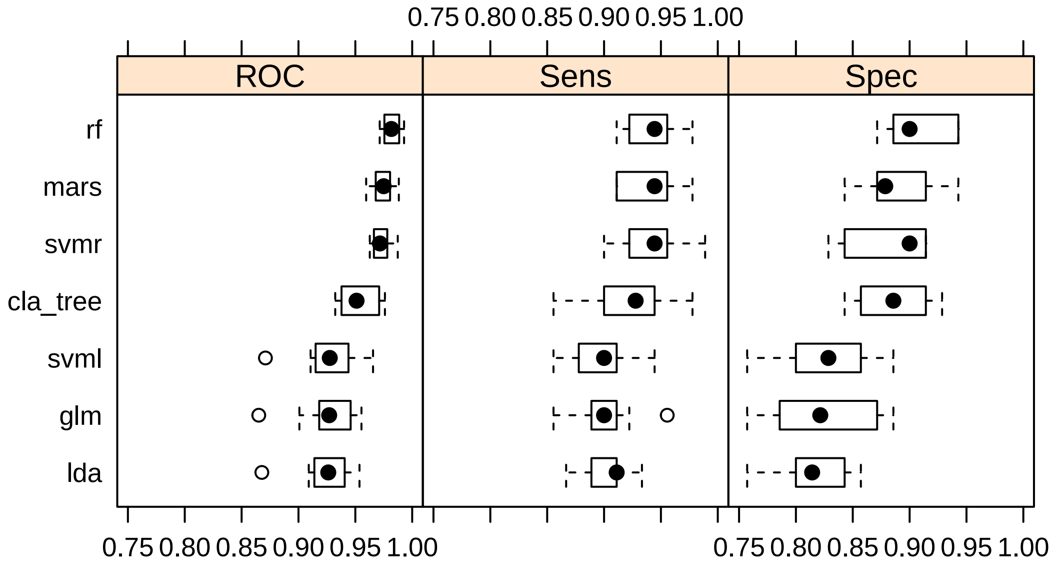
```
## [1] 0.1
```

```r
confusionMatrix(data = as.factor(pred_svmr),
                reference = as.factor(y_test))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  no yes
##        no  208  23
##        yes  17 152
##
##                Accuracy : 0.9
##                  95% CI : (0.8663, 0.9276)
##     No Information Rate : 0.5625
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.796
##
##  Mcnemar's Test P-Value : 0.4292
##
##             Sensitivity : 0.9244
##             Specificity : 0.8686
##          Pos Pred Value : 0.9004
##          Neg Pred Value : 0.8994
##              Prevalence : 0.5625
##          Detection Rate : 0.5200
##    Detection Prevalence : 0.5775
##       Balanced Accuracy : 0.8965
##
##        'Positive' Class : no
##
```

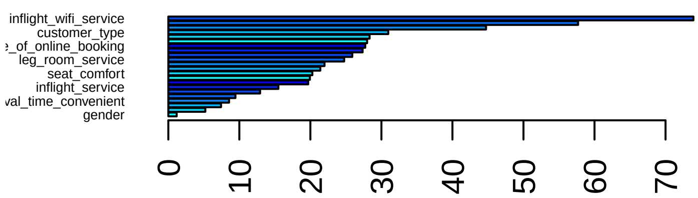
Resample

```r
resamp <- resamples(list(glm = model.glm, mars = model.mars,
                         lda = model.lda, cla_tree = model.tree,
                         rf = model.rf, svml = model.svml,
                         svmr = model.svmr))

bwplot(resamp)
```



Select the rf model and interpret

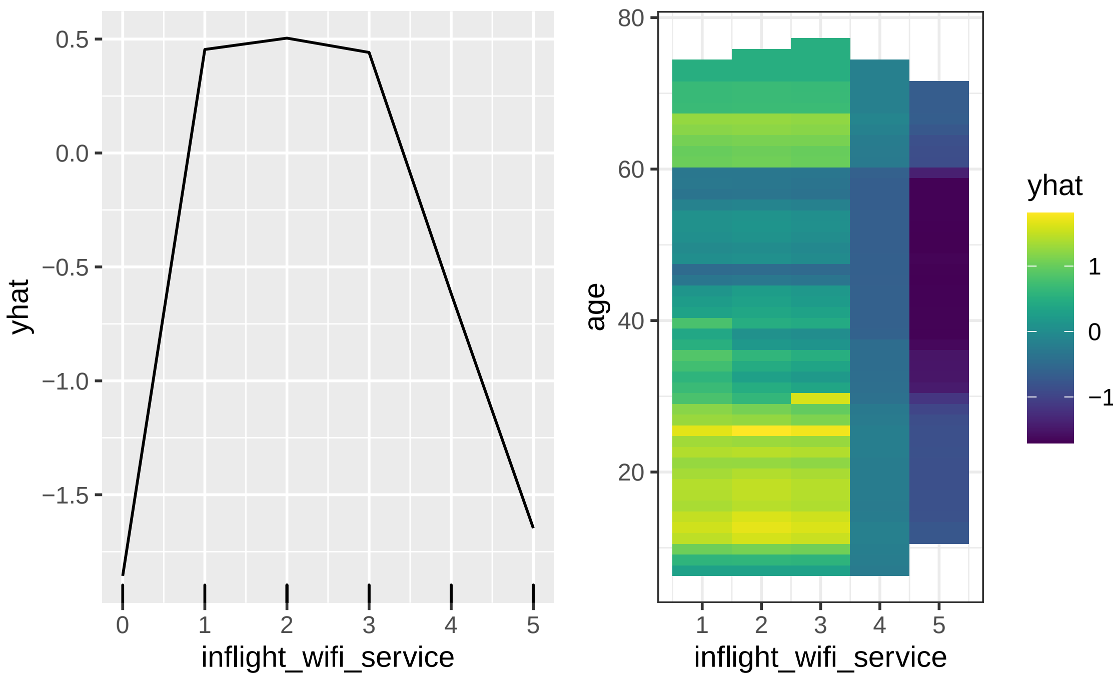
```
## importance variable
set.seed(1234)
rf2.final.per <- ranger(factor(satisfaction) ~ .,
                        data = train,
                        mtry = model.rf$bestTune[[1]],
                        min.node.size = model.rf$bestTune[[3]],
                        splitrule = "gini",
                        importance = "permutation",
                        scale.permutation.importance = TRUE)

barplot(sort(ranger::importance(rf2.final.per), decreasing = FALSE),
        las = 2,
        horiz = TRUE,
        cex.names = 0.4,
        col = colorRampPalette(colors = c("cyan", "blue"))(8))
```




```
pdp1.rf <- model.rf %>%
  partial(pred.var = c("inflight_wifi_service")) %>%
  autoplot(train = train, rug = TRUE)

pdp2.rf <- model.rf %>%
  partial(pred.var = c("inflight_wifi_service", "age"), chull = TRUE) %>%
  autoplot(train = train, rug = TRUE)

grid.arrange(pdp1.rf, pdp2.rf, nrow = 1)
```

```
## Warning: Use of `object[[1L]]` is discouraged. Use `.data[[1L]]` instead.
```

```
## Warning: Use of `object[["yhat"]]` is discouraged. Use `.data[["yhat"]]`
## instead.

## Warning: Use of `x.rug[[1L]]` is discouraged. Use `.data[[1L]]` instead.

## Warning: Use of `object[[1L]]` is discouraged. Use `.data[[1L]]` instead.

## Warning: Use of `object[[2L]]` is discouraged. Use `.data[[2L]]` instead.

## Warning: Use of `object[["yhat"]]` is discouraged. Use `.data[["yhat"]]`
## instead.

## Warning: Use of `object[["yhat"]]` is discouraged. Use `.data[["yhat"]]`
## instead.
```

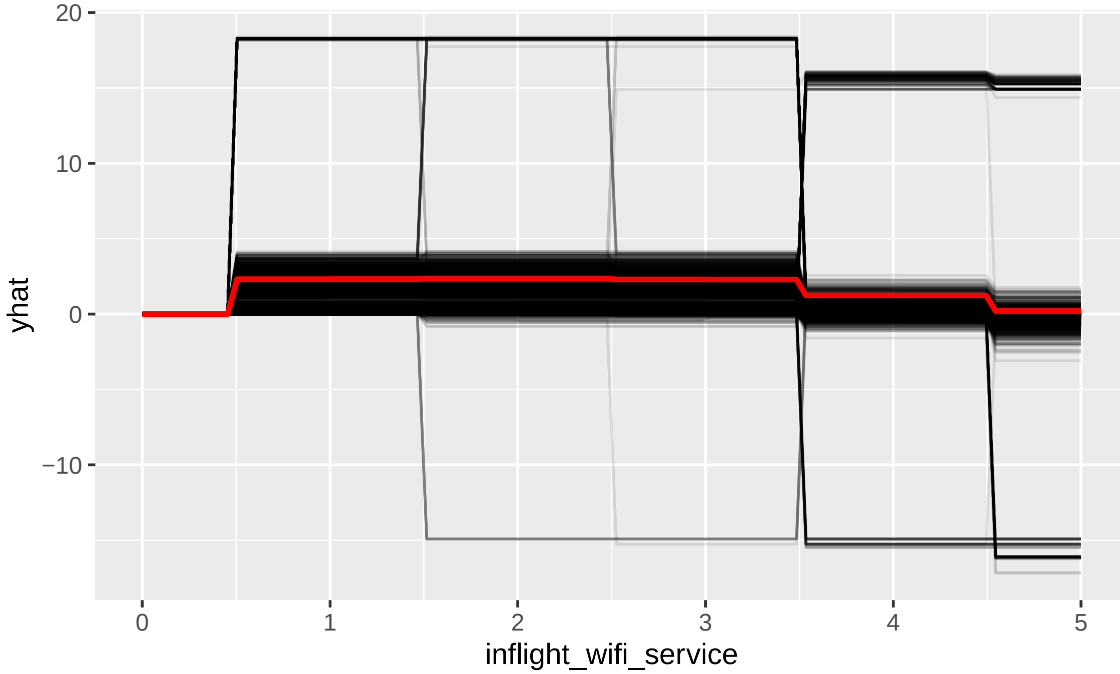



```r
ice.rf <- model.rf %>%
  partial(pred.var = "inflight_wifi_service",
          grid.resolution = 100,
          ice = TRUE) %>%
  autoplot(train = dat, alpha = .1,
          center = TRUE)
```

```
## Warning: `fun.y` is deprecated. Use `fun` instead.
```

```r
ice.rf
```

```
## Warning: Use of `object[["yhat.id"]]` is discouraged. Use `.data[["yhat.id"]]`
## instead.

## Warning: Use of `object[[1L]]` is discouraged. Use `.data[[1L]]` instead.

## Warning: Use of `object[["yhat"]]` is discouraged. Use `.data[["yhat"]]`
## instead.

## Warning: Use of `object[[1L]]` is discouraged. Use `.data[[1L]]` instead.

## Warning: Use of `object[["yhat"]]` is discouraged. Use `.data[["yhat"]]`
## instead.
```




```
##lime
#explain.rf <- lime(data.frame(x_train), model.rf)

#new_obs = x_train[1:10,]
#explana.obs = explain(data.frame(new_obs),
#                       explain.rf,
#                       n_features = 10)

#plot_features(explana.obs)
```