R_Data_Science_Predictive_Modeling

Kim Wong

04/01/2021

Material taken from

Graham J. Williams. 2017. The Essentials of Data Science: Knowledge Discovery Using R (1st ed.). Chapman & Hall/CRC.

Load required libraries

```
library(magrittr)     # Pipe operator %>% %<>% %T>% equals().
library(lubridate)    # Dates and time.
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(rattle)       # normVarNames().
```

```
## 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.
```

```
library(ROCR)         # Use prediction() for evaluation.
library(rpart)        # Model: decision tree.
library(scales)       # Include commas in numbers.
library(stringi)      # String concat operator %s+%.
library(tidyverse)    # ggplot2, tibble, tidyr, readr, purr, dplyr, stringr
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.3     v dplyr   1.0.5
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1
## v purrr   0.3.4
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x lubridate::as.difftime() masks base::as.difftime()
## x readr::col_factor()      masks scales::col_factor()
```

```
## x lubridate::date()        masks base::date()
## x purrr::discard()         masks scales::discard()
## x tidyr::extract()         masks magrittr::extract()
## x dplyr::filter()          masks stats::filter()
## x lubridate::intersect()   masks base::intersect()
## x dplyr::lag()             masks stats::lag()
## x purrr::set_names()       masks magrittr::set_names()
## x lubridate::setdiff()     masks base::setdiff()
## x lubridate::union()       masks base::union()
```

Load dataset

specify folder name

```
fpath <- getwd() %>% print()
```

```
## [1] "/Users/kimwong/OneDrive - University of Pittsburgh/Documents/Kim F. Wong/CRC_Workshop/2021/Advar
```

specify timestamp

```
dsdate <- "_20201104"
```

specify filename for dataset

```
dsname="cleaned_weatherAUS"
dsrdata <-
  file.path(fpath, dsname %s+% dsdate %s+% ".RData") %T>%
  print()
```

```
## [1] "/Users/kimwong/OneDrive - University of Pittsburgh/Documents/Kim F. Wong/CRC_Workshop/2021/Advar
```

load dataset

```
load(dsrdata) %>% print()
```

```
##  [1] "ds"        "dsname"    "dspath"    "dsdate"    "nobs"      "vars"
##  [7] "target"    "risk"      "id"        "ignore"    "omit"      "inputi"
## [13] "inputs"    "numi"      "numc"      "cati"      "catc"      "form"
## [19] "seed"      "train"     "validate"  "test"      "tr_target" "tr_risk"
## [25] "va_target" "va_risk"   "te_target" "te_risk"
```

inspect metadata

```
dsname
```

```
## [1] "cleaned_weatherAUS"
```

```
dspath
```

```
## [1] "https://rattle.togaware.com/weatherAUS.csv"
```

```
dsdate
```

```
## [1] "_20201104"
```

```
nobs %>% comcat()
```

```
## 173,890
```

```
vars
```

```
##  [1] "rain_tomorrow"   "min_temp"        "max_temp"        "rainfall"
##  [5] "evaporation"     "sunshine"        "wind_gust_dir"   "wind_gust_speed"
##  [9] "wind_dir_9am"    "wind_dir_3pm"    "wind_speed_9am"  "wind_speed_3pm"
## [13] "humidity_9am"    "humidity_3pm"    "pressure_9am"    "cloud_9am"
## [17] "cloud_3pm"       "rain_today"      "season"          "cluster"
```

```
target
```

```
## [1] "rain_tomorrow"
```

```
risk
```

```
## [1] "risk_mm"
```

```
id
```

```
## [1] "date"     "location" "year"
```

```
ignore
```

```
## [1] "date"         "location"     "risk_mm"      "temp_3pm"     "pressure_3pm"
## [6] "temp_9am"
```

```
omit
```

```
## NULL
```

```
train %>% length() %>% comcat()
```

```
## 121,722
```

```
validate %>% length() %>% comcat()
```

```
## 26,083
```

```
test %>% length() %>% comcat()
```

```
## 26,085
```

Building a decision tree model

train a decision tree model

```
m_rp <- rpart(form, ds[train, vars])
```

recast model in terms of generic variables

```
model <- m_rp
mtype <- "rpart"
mdesc <- "Decision Tree"
```

basic model structure

```
model
```

```
## n= 121722
##
## node), split, n, loss, yval, (yprob)
```

```
##       * denotes terminal node
##
##  1) root 121722 26184 no (0.7848869 0.2151131)
##    2) humidity_3pm< 71.5 102450 14431 no (0.8591410 0.1408590) *
##    3) humidity_3pm>=71.5 19272  7519 yes (0.3901515 0.6098485)
##      6) humidity_3pm< 82.5 10665  4983 no (0.5327707 0.4672293)
##       12) rainfall< 1.1 6116  2220 no (0.6370177 0.3629823) *
##       13) rainfall>=1.1 4549  1786 yes (0.3926138 0.6073862) *
##      7) humidity_3pm>=82.5 8607  1837 yes (0.2134309 0.7865691) *
```

summary of basic model structure

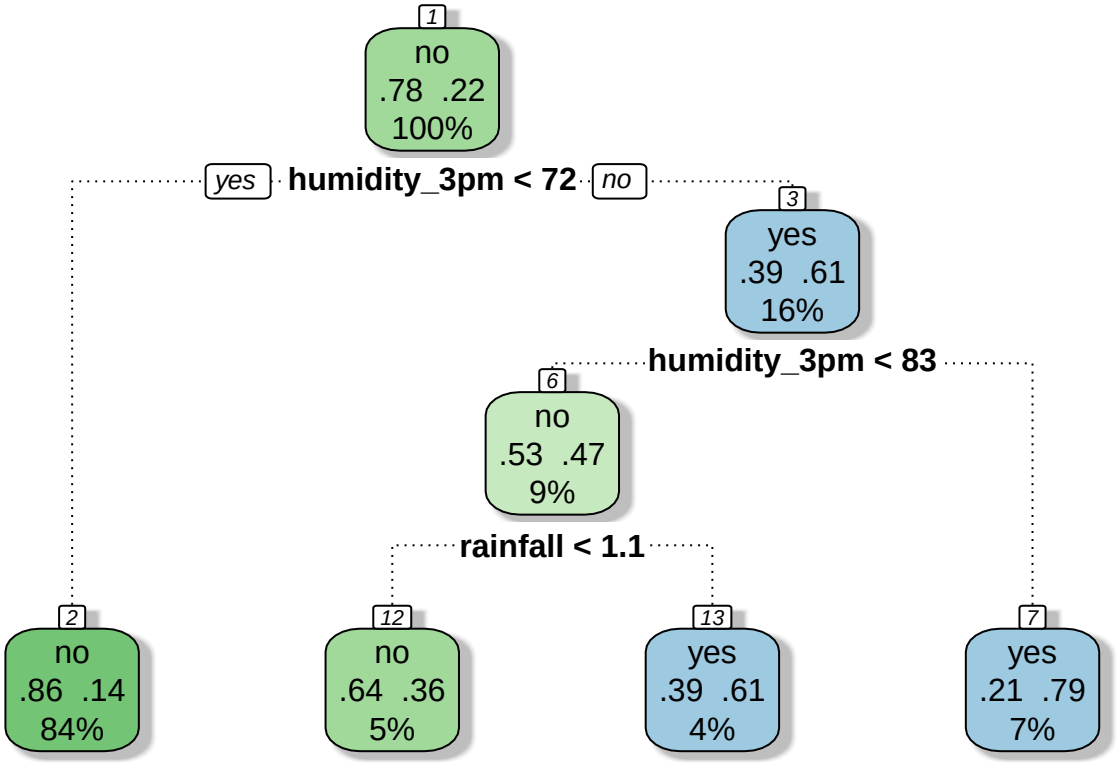
```
summary(model)
```

```
## Call:
## rpart(formula = form, data = ds[train, vars])
##   n= 121722
##
##           CP nsplit rel error    xerror        xstd
## 1 0.16170180      0 1.0000000 1.0000000 0.005475018
## 2 0.03200428      1 0.8382982 0.8382600 0.005122634
## 3 0.01000000      3 0.7742896 0.7758555 0.004968461
##
## Variable importance
## humidity_3pm     rainfall     max_temp humidity_9am      cluster
##           90            4            3            2            1
##
## Node number 1: 121722 observations,    complexity param=0.1617018
##   predicted class=no   expected loss=0.2151131  P(node) =1
##     class counts: 95538 26184
##    probabilities: 0.785 0.215
##   left son=2 (102450 obs) right son=3 (19272 obs)
##   Primary splits:
##       humidity_3pm < 71.5    to the left,  improve=7206.462, (3607 missing)
##       rainfall     < 0.65    to the left,  improve=4146.530, (1194 missing)
##       rain_today   splits as  LR,          improve=4113.371, (1194 missing)
##       sunshine     < 6.95    to the right, improve=3233.815, (64159 missing)
##       cloud_3pm    < 6.5     to the left,  improve=2832.268, (53089 missing)
##   Surrogate splits:
##       max_temp < 10.65   to the right, agree=0.843, adj=0.03, (3355 split)
##
## Node number 2: 102450 observations
##   predicted class=no   expected loss=0.140859  P(node) =0.841672
##     class counts: 88019 14431
##    probabilities: 0.859 0.141
##
## Node number 3: 19272 observations,    complexity param=0.03200428
##   predicted class=yes  expected loss=0.3901515  P(node) =0.158328
##     class counts:  7519 11753
##    probabilities: 0.390 0.610
##   left son=6 (10665 obs) right son=7 (8607 obs)
##   Primary splits:
##       humidity_3pm    < 82.5    to the left,  improve=984.5670, (104 missing)
##       rainfall        < 2.15    to the left,  improve=610.6582, (219 missing)
```

```
##       rain_today      splits as  LR,          improve=609.6742, (219 missing)
##       wind_gust_speed < 42      to the left,  improve=418.2756, (1349 missing)
##       pressure_9am    < 1014.05 to the right, improve=403.6948, (2375 missing)
##   Surrogate splits:
##       humidity_9am < 92.5    to the left,  agree=0.609, adj=0.120, (35 split)
##       cluster      splits as  LLRRL,       agree=0.587, adj=0.072, (69 split)
##       max_temp     < 12.45   to the right, agree=0.587, adj=0.071, (0 split)
##       rainfall     < 3.15    to the left,  agree=0.575, adj=0.044, (0 split)
##       rain_today   splits as  LR,          agree=0.566, adj=0.022, (0 split)
##
## Node number 6: 10665 observations,    complexity param=0.03200428
##   predicted class=no   expected loss=0.4672293  P(node) =0.08761769
##     class counts:  5682  4983
##    probabilities: 0.533 0.467
##   left son=12 (6116 obs) right son=13 (4549 obs)
##   Primary splits:
##       rainfall        < 1.1     to the left,  improve=314.86430, (120 missing)
##       rain_today      splits as  LR,          improve=314.86430, (120 missing)
##       wind_gust_speed < 42      to the left,  improve=296.24870, (813 missing)
##       pressure_9am    < 1014.05 to the right, improve=242.62100, (903 missing)
##       cluster         splits as  RLLLL,       improve= 97.07454, (0 missing)
##   Surrogate splits:
##       humidity_9am   < 86.5    to the left,  agree=0.623, adj=0.120, (105 split)
##       cluster        splits as  RLRLL,       agree=0.583, adj=0.026, (15 split)
##       min_temp       < 22.85   to the left,  agree=0.577, adj=0.012, (0 split)
##       wind_speed_9am < 29      to the left,  agree=0.574, adj=0.006, (0 split)
##
## Node number 7: 8607 observations
##   predicted class=yes  expected loss=0.2134309  P(node) =0.07071031
##     class counts:  1837  6770
##    probabilities: 0.213 0.787
##
## Node number 12: 6116 observations
##   predicted class=no   expected loss=0.3629823  P(node) =0.05024564
##     class counts:  3896  2220
##    probabilities: 0.637 0.363
##
## Node number 13: 4549 observations
##   predicted class=yes  expected loss=0.3926138  P(node) =0.03737204
##     class counts:  1786  2763
##    probabilities: 0.393 0.607
```

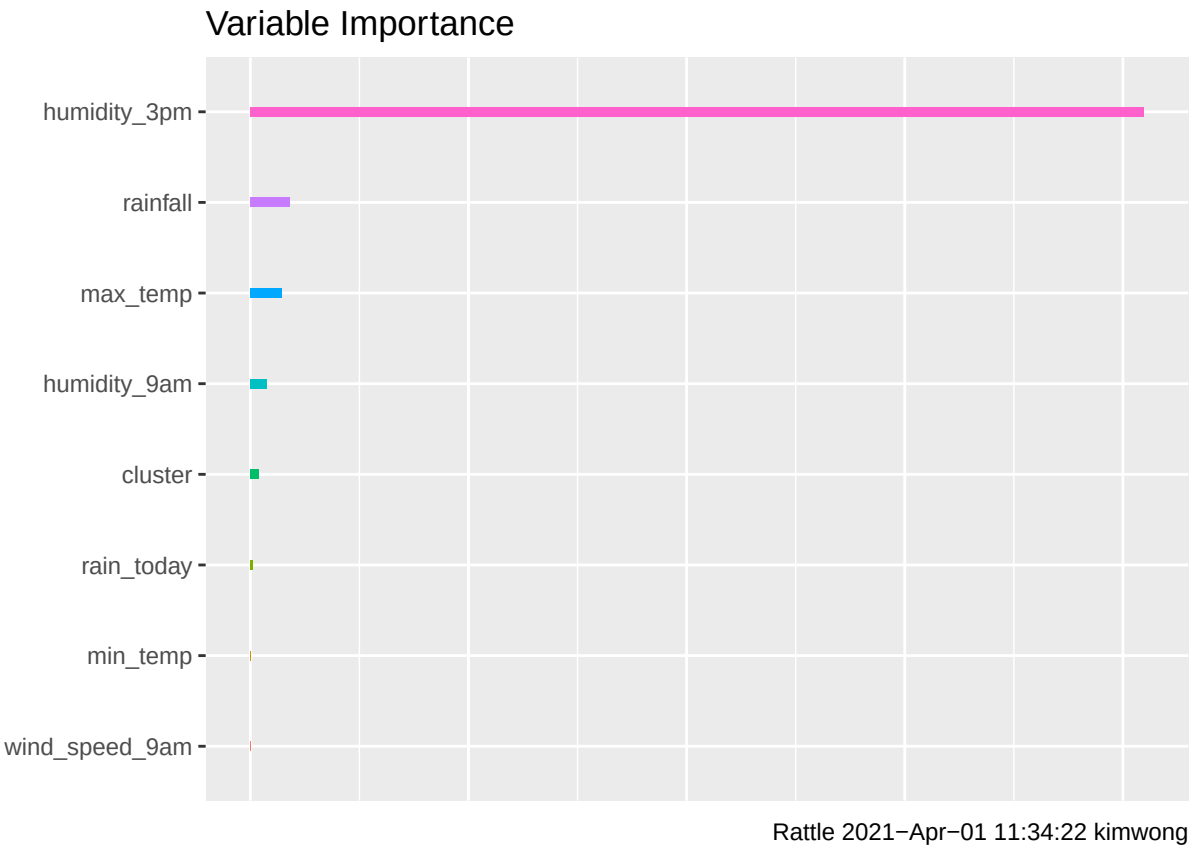
visualize the tree

```
fancyRpartPlot(model)
```



visualize the variable importance

```
ggVarImp(model)
```



Evaluate model performance

using decision tree model, make prediction on the training set

```
model %>%
  predict(newdata=ds[train, vars], type="class") %>%
  set_names(NULL) %T>%
  {head(., 20) %>% print()} ->
tr_class
```

```
##  [1] no  no  no  no  no  no  yes no  no  no  no  no  yes no  no  no  no  no  no
## [20] no
## Levels: no yes
```

compare above prediction against observations in the training dataset

```
head(tr_target, 20)
```

```
##  [1] no  no  no  no  no  yes yes no  no  no  no  no  yes no  no  no  no  no  no
## [20] no
## Levels: no yes
```

obtain matches between prediction and observations

```
head(tr_class) == head(tr_target)
```

```
## [1]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
```

```
sum(head(tr_class) == head(tr_target))
```

```
## [1] 5
```

```
sum(tr_class == tr_target)
```

```
## [1] 101448
```

```
length(train)
```

```
## [1] 121722
```

using decision tree model, predict the probability of raining tomorrow on the training set

```
model %>%
  predict(newdata=ds[train, vars], type="prob") %>%
  .[,2] %>%
  set_names(NULL) %>%
  round(2) %T>%
  {head(., 20) %>% print()} ->
tr_prob
```

```
##  [1] 0.14 0.14 0.14 0.14 0.14 0.36 0.79 0.14 0.14 0.14 0.14 0.14 0.79 0.14 0.14
## [16] 0.14 0.14 0.14 0.14 0.14
```

compute overall accuracy

```
sum(tr_class == tr_target) %>%
  divide_by(length(tr_target)) %T>%
  {
    percent(.) %>%
      sprintf("Overall accuracy = %s\n", .) %>%
      cat()
  } ->
tr_acc
```

```
## Overall accuracy = 83%
```

compute overall error

```
sum(tr_class != tr_target) %>%
  divide_by(length(tr_target)) %T>%
  {
    percent(.) %>%
      sprintf("Overall error = %s\n", .) %>%
      cat()
  } ->
tr_err
```

```
## Overall error = 17%
```

comparison of prediction and observation as a confusion matrix (counts)

```
table(tr_target, tr_class, dnn=c("Actual", "Predicted"))
```

```
##       Predicted
## Actual    no   yes
##    no  91915  3623
##    yes 16651  9533
```

comparison of prediction and observation as a confusion matrix (percentage)

```
table(tr_target, tr_class, dnn=c("Actual", "Predicted")) %>%
  divide_by(length(tr_target)) %>%
  multiply_by(100) %>% round(1)
```

```
##       Predicted
## Actual   no  yes
##    no  75.5  3.0
##    yes 13.7  7.8
```

rattle::errorMatrix provides confusion matrix and class errors

```
errorMatrix(tr_target, tr_class, count=TRUE)
```

```
##       Predicted
## Actual    no  yes Error
##    no  91915 3623   3.8
##    yes 16651 9533  63.6
```

```
errorMatrix(tr_target, tr_class) %T>%
  print() ->
tr_matrix
```

```
##       Predicted
## Actual   no yes Error
##    no  75.5 3.0   3.8
##    yes 13.7 7.8  63.6
```

compute recall, precision, and F-score. The recall is the proportion of true positives that are identified by the model. The precision is the proportion of true positives that are among the positives predicted by the model. The F-score is the harmonic mean of these two measures.

```
tr_rec <- (tr_matrix[2,2]/(tr_matrix[2,2]+tr_matrix[2,1])) %T>%
  {percent(.) %>% sprintf("Recall = %s\n", .) %>% cat()}
```

```
## Recall = 36%
```

```
tr_pre <- (tr_matrix[2,2]/(tr_matrix[2,2]+tr_matrix[1,2])) %T>%
  {percent(.) %>% sprintf("Precision = %s\n", .) %>% cat()}
```

```
## Precision = 72%
```

```
tr_fsc <- ((2 * tr_pre * tr_rec)/(tr_rec + tr_pre))  %T>%
  {sprintf("F-Score = %.3f\n", .) %>% cat()}
```

```
## F-Score = 0.483
```

Random Forest

load additional library

```
library(randomForest)           # Model: randomForest() na.roughfix()
```

```
## randomForest 4.6-14
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:dplyr':
##
##     combine
## The following object is masked from 'package:ggplot2':
##
##     margin
## The following object is masked from 'package:rattle':
##
##     importance
```

reload data

```
load(dsrdata) %>% print()
```

```
##  [1] "ds"        "dsname"    "dspath"    "dsdate"    "nobs"      "vars"
##  [7] "target"    "risk"      "id"        "ignore"    "omit"      "inputi"
## [13] "inputs"    "numi"      "numc"      "cati"      "catc"      "form"
## [19] "seed"      "train"     "validate"  "test"      "tr_target" "tr_risk"
## [25] "va_target" "va_risk"   "te_target" "te_risk"
```

train a random forest model

```
m_rf <- randomForest(form, data=ds[train, vars], ntree=10, na.action=na.roughfix, importance=TRUE)
```

recast model in terms of generic variables

```
model <- m_rf
mtype <- "randomForest"
mdesc <- "Random Forest"
```

basic model structure

```
model
```

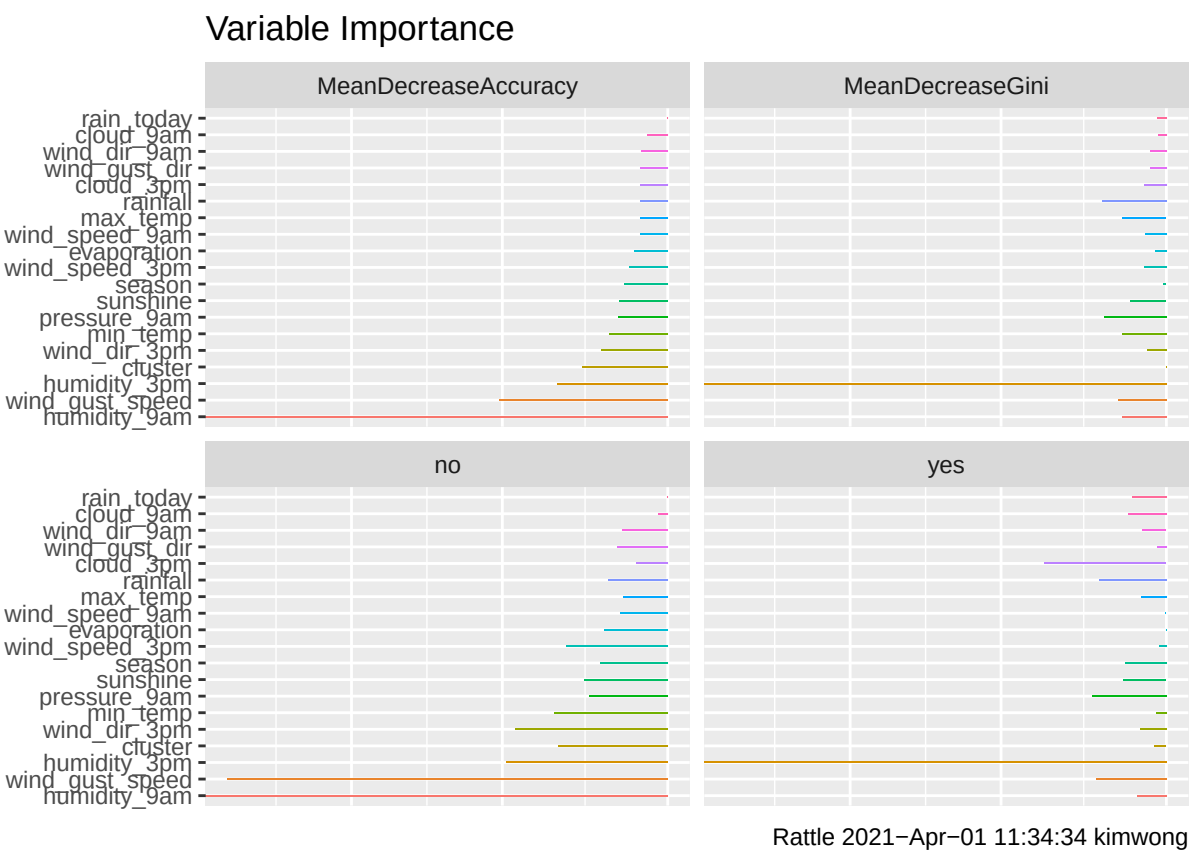
```
##
## Call:
##  randomForest(formula = form, data = ds[train, vars], ntree = 10,      importance = TRUE, na.action =
##                Type of random forest: classification
##                      Number of trees: 10
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 17.86%
```

```
## Confusion matrix:
##        no   yes class.error
## no  86234  8337  0.08815599
## yes 13192 12751  0.50849940
```

visualize the variable importance

```
ggVarImp(model, log=TRUE)
```

```
## Warning: Transformation introduced infinite values in continuous y-axis
```



make prediction on the validation dataset

```
model %>%
  predict(newdata=ds[validate, vars], type="prob") %>%
  .[,2] %>%
  set_names(NULL) %>%
  round(2) %T>%
  {head(., 20) %>% print()} ->
va_prob
```

```
##  [1]  NA  NA  NA 0.0  NA 0.0  NA  NA 0.0  NA  NA  NA  NA  NA 0.0  NA  NA  NA 1.0
## [20] 0.2
```

```
model %>%
  predict(newdata=ds[validate, vars], type="response") %>%
  set_names(NULL) %T>%
```

```
  {head(., 20) %>% print()} ->
va_class
```

```
##  [1] <NA> <NA> <NA> no   <NA> no   <NA> <NA> no   <NA> <NA> <NA> <NA> <NA> no
## [16] <NA> <NA> <NA> yes  no
## Levels: no yes
```

compute overall accuracy (note the na.rm=TRUE for checking and discarding prediction/observations with missing values).

```
sum(va_class == va_target, na.rm=TRUE) %>%
  divide_by(va_class %>% is.na() %>% not() %>% sum()) %T>%
  {
    percent(.) %>%
      sprintf("Overall accuracy = %s\n", .) %>%
      cat()
  } ->
va_acc
```

```
## Overall accuracy = 86%
```

compute overall error (note the na.rm=TRUE for checking and discarding prediction/observations with missing values).

```
sum(va_class != va_target, na.rm=TRUE) %>%
  divide_by(va_class %>% is.na() %>% not() %>% sum()) %T>%
    {
    percent(.) %>%
      sprintf("Overall error = %s\n", .) %>%
      cat()
  } ->
va_err
```

```
## Overall error = 14%
```

rattle::errorMatrix provides confusion matrix and class errors

```
errorMatrix(va_target, va_class, count=TRUE)
```

```
##       Predicted
## Actual   no  yes Error
##    no  6985  409   5.5
##    yes  902 1053  46.1
```

```
errorMatrix(va_target, va_class) %T>%
  print() ->
va_matrix
```

```
##       Predicted
## Actual   no  yes Error
##    no  74.7  4.4   5.5
##    yes  9.6 11.3  46.1
```

```
va_matrix %>%
  diag() %>%
```

```
  sum(na.rm=TRUE) %>%
  subtract(100, .) %>%
  sprintf("Overall error percentage = %s%%\n", .) %>%
  cat()
```

```
## Overall error percentage = 14%
```

```
va_matrix[,"Error"] %>%
  mean(na.rm=TRUE) %>%
  sprintf("Averaged class error percentage = %s%%\n", .) %>%
  cat()
```

```
## Averaged class error percentage = 25.8%
```

compute recall, precision, and F-score. The recall is the proportion of true positives that are identified by the model. The precision is the proportion of true positives that are among the positives predicted by the model. The F-score is the harmonic mean of these two measures.

```
va_rec <- (va_matrix[2,2]/(va_matrix[2,2]+va_matrix[2,1])) %T>%
  {percent(.) %>% sprintf("Recall = %s\n", .) %>% cat()}
```

```
## Recall = 54%
```

```
va_pre <- (va_matrix[2,2]/(va_matrix[2,2]+va_matrix[1,2])) %T>%
  {percent(.) %>% sprintf("Precision = %s\n", .) %>% cat()}
```

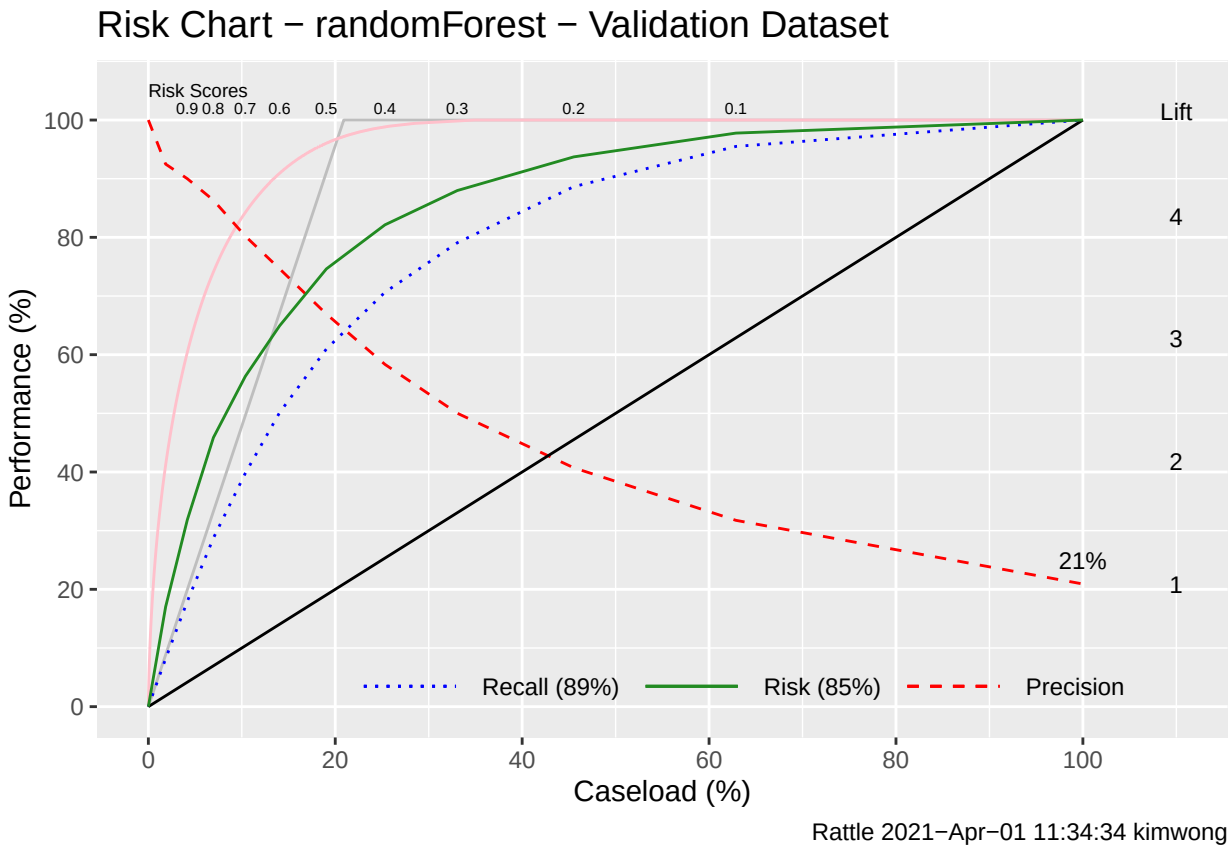
```
## Precision = 72%
```

```
va_fsc <- ((2 * va_pre * va_rec)/(va_rec + va_pre))  %T>%
  {sprintf("F-Score = %.3f\n", .) %>% cat()}
```

```
## F-Score = 0.617
```

plot risk chart

```
riskchart(va_prob, va_target, va_risk) +
  labs(title="Risk Chart - " %s+% mtype %s+% " - Validation Dataset") +
  theme(plot.title=element_text(size=14))
```



Extreme gradient boosting

```
library(Matrix)                 # Data wrangling: sparse.model.matrix()
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
library(xgboost)                # Models: extreme gradient boosting
```

```
##
## Attaching package: 'xgboost'
```

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

```
## The following object is masked from 'package:rattle':
##
##     xgboost
```

convert categoric variables into numeric

```
formula(target %s+% "~ .-1") %>%
  sparse.model.matrix(data=ds[vars] %>% na.roughfix()) %T>%
```

```
  {dim(.) %>% print()} %T>%
  {head(.) %>% print()} ->
sds
```

```
## [1] 173890     67
## 6 x 67 sparse Matrix of class "dgCMatrix"
```

```
##    [[ suppressing 67 column names 'min_temp', 'max_temp', 'rainfall' ... ]]
```

```
##
## 1 13.4 22.9 0.6 4.8 8.5 . . . . . . . . . . . . 1 . . . 44  0.24404677
## 2  7.4 25.1 .   4.8 8.5 . . . . . . . . . . . . . 1 . . 44  0.40674461
## 3 12.9 25.7 .   4.8 8.5 . . . . . . . . . . . . 1 . . . . 46  0.24404677
## 4  9.2 28.0 .   4.8 8.5 . . 1 . . . . . . . . . . . . . . 24 -0.08134892
## 5 17.5 32.3 1.0 4.8 8.5 . . . . . . . . . . . . . 1 . . . 41 -0.24404677
## 6 14.6 29.7 0.2 4.8 8.5 . . . . . . . . . . . . . . 1 . . 56  0.24404677
##
## 1 -0.0132314 -0.2659700 -0.2930988 -0.07350559  0.21190692  0.3402856
## 2  0.4630991  0.4532448  0.3980895  0.31852424  0.23345677  0.1568692
## 3 -0.0132314 -0.2659700 -0.2930988 -0.07350559  0.21190692  0.3402856
## 4 -0.2513967  0.1783095  0.1881082 -0.25615585 -0.08979107  0.3016716
## 5 -0.0132314  0.2659700 -0.2930988  0.07350559  0.21190692 -0.3402856
## 6 -0.0132314 -0.2659700 -0.2930988 -0.07350559  0.21190692  0.3402856
##
## 1  0.22101863 -0.05630793 -0.31520718 -0.4272698 -0.379967215 -0.247869131
## 2  0.09641752  0.05393707  0.02724013  0.0122678  0.004834188  0.001615835
## 3  0.22101863 -0.05630793 -0.31520718 -0.4272698 -0.379967215 -0.247869131
## 4 -0.03708366 -0.29932108  0.18289799  0.2274800 -0.329691654 -0.046212889
## 5  0.22101863  0.05630793 -0.31520718  0.4272698 -0.379967215  0.247869131
## 6  0.22101863 -0.05630793 -0.31520718 -0.4272698 -0.379967215 -0.247869131
##
## 1 -0.118040549 -0.0365326327  0.2982794  0.1190826 -0.14244837 -0.3222629
## 2  0.000432383  0.0000802915  0.1898142 -0.1190826 -0.29983887 -0.1472785
## 3 -0.118040549 -0.0365326327  0.1898142 -0.1190826 -0.29983887 -0.1472785
## 4  0.432815345 -0.4018589595 -0.1898142 -0.1190826  0.29983887 -0.1472785
## 5 -0.118040549  0.0365326327  0.3525120  0.2778595  0.09064896 -0.1326965
## 6 -0.118040549 -0.0365326327  0.2440468 -0.0132314 -0.26597003 -0.2930988
##
## 1 -0.31852424 -0.1400741  0.1150374  0.3278196  0.42379124  0.3982247
## 2  0.17151305  0.3124729  0.1262998 -0.1972851 -0.34081114 -0.1725208
## 3  0.17151305  0.3124729  0.1262998 -0.1972851 -0.34081114 -0.1725208
## 4 -0.17151305  0.3124729 -0.1262998 -0.1972851  0.34081114 -0.1725208
## 5 -0.31852424 -0.4202222 -0.4287759 -0.3663866 -0.26968533 -0.1725208
## 6 -0.07350559  0.2119069  0.3402856  0.2210186 -0.05630793 -0.3152072
##
## 1  0.2975817  0.18079865  0.08844005  0.033293488  0.008430608 20 24 71 22
## 2  0.1587803  0.39930397  0.41871678  0.275427947  0.109597898  4 22 44 25
## 3  0.1587803  0.39930397  0.41871678  0.275427947  0.109597898 19 26 38 30
## 4 -0.1587803  0.39930397 -0.41871678  0.275427947 -0.109597898 11  9 45 16
## 5 -0.0956888 -0.04544137 -0.01798963 -0.005620979 -0.001204373  7 20 82 33
## 6 -0.4272698 -0.37996721 -0.24786913 -0.118040549 -0.036532633 19 24 55 23
##
## 1 1007.7 8 5 . . 1 . . . . 1
## 2 1010.6 5 5 . . 1 . . . . 1
## 3 1007.6 5 2 . . 1 . . . . 1
```

```
## 4 1017.6 5 5 . . 1 . . . . 1
## 5 1010.8 7 8 . . 1 . . . . 1
## 6 1009.2 5 5 . . 1 . . . . 1
```

generate a vector to populate the values of the target variable

```
ds[target] %>%
  unlist(use.names=FALSE) %>%
  equals("yes") %T>%
  {head(., 20) %>% print()} ->
label
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE  TRUE
## [13]  TRUE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE
```

train an extreme gradient boosting model

```
m_xg <- xgboost(data=sds[train,],
                label=label[train],
                nrounds=100,
                print_every_n=15,
                objective="binary:logistic")
```

```
## [11:34:40] WARNING: amalgamation/../src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evalu
## [1]  train-logloss:0.554269
## [16] train-logloss:0.326039
## [31] train-logloss:0.309459
## [46] train-logloss:0.299390
## [61] train-logloss:0.291464
## [76] train-logloss:0.282682
## [91] train-logloss:0.275746
## [100]    train-logloss:0.271274
```

recast model in terms of generic variables

```
model <- m_xg
mtype <- "xgboost"
mdesc <- "Extreme Gradient Boosting"
```

basic model structure

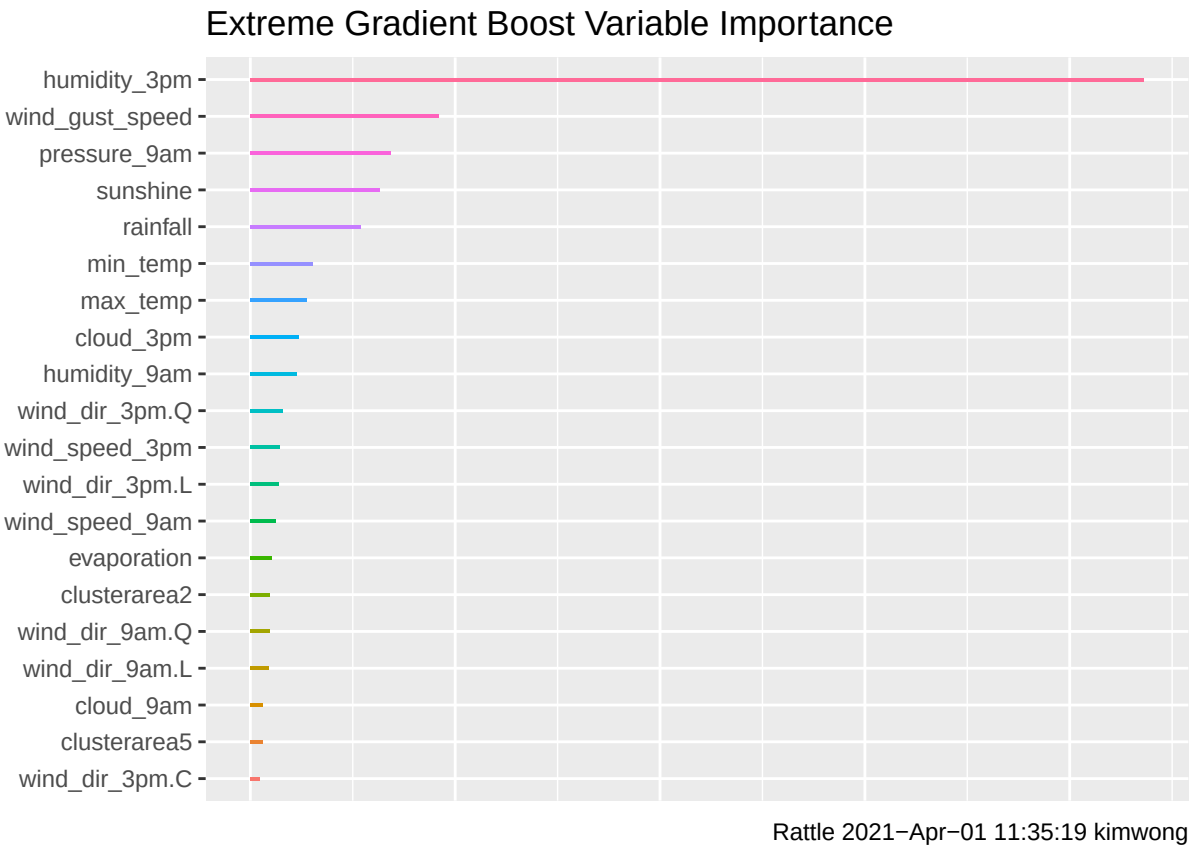
```
model
```

```
## ##### xgb.Booster
## raw: 648.2 Kb
## call:
##   xgb.train(params = params, data = dtrain, nrounds = nrounds,
##     watchlist = watchlist, verbose = verbose, print_every_n = print_every_n,
##     early_stopping_rounds = early_stopping_rounds, maximize = maximize,
##     save_period = save_period, save_name = save_name, xgb_model = xgb_model,
##     callbacks = callbacks, objective = "binary:logistic")
## params (as set within xgb.train):
##   objective = "binary:logistic", validate_parameters = "TRUE"
## xgb.attributes:
```

```
##   niter
## callbacks:
##   cb.print.evaluation(period = print_every_n)
##   cb.evaluation.log()
## # of features: 67
## niter: 100
## nfeatures : 67
## evaluation_log:
##     iter train_logloss
##        1      0.554269
##        2      0.478052
## ---
##       99      0.272162
##      100      0.271274
```

visualize the variable importance

```
ggVarImp(model, feature_names=colnames(sds), n=20)
```



make prediction on the validation dataset

```
model %>%
  predict(newdata=sds[validate,], type="prob") %>%
  set_names(NULL) %>%
  round(2) %T>%
  {head(., 20) %>% print()} ->
```

```
va_prob
```

```
##  [1] 0.02 0.15 0.02 0.03 0.90 0.01 0.08 0.73 0.03 0.92 0.73 0.56 0.16 0.06 0.06
## [16] 0.95 0.02 0.14 0.97 0.17
```

```
va_prob %>%
  is_greater_than(0.5) %>%
  ifelse("yes", "no") %T>%
  {head(., 20) %>% print()} ->
va_class
```

```
##  [1] "no"  "no"  "no"  "no"  "yes" "no"  "no"  "yes" "no"  "yes" "yes" "yes"
## [13] "no"  "no"  "no"  "yes" "no"  "no"  "yes" "no"
```

compute overall accuracy (note the na.rm=TRUE for checking and discarding prediction/observations with missing values).

```
sum(va_class == va_target, na.rm=TRUE) %>%
  divide_by(va_class %>% is.na() %>% not() %>% sum()) %T>%
  {
    percent(.) %>%
      sprintf("Overall accuracy = %s\n", .) %>%
      cat()
  } ->
va_acc
```

```
## Overall accuracy = 86%
```

compute overall error (note the na.rm=TRUE for checking and discarding prediction/observations with missing values).

```
sum(va_class != va_target, na.rm=TRUE) %>%
  divide_by(va_class %>% is.na() %>% not() %>% sum()) %T>%
    {
    percent(.) %>%
      sprintf("Overall error = %s\n", .) %>%
      cat()
  } ->
va_err
```

```
## Overall error = 14%
```

rattle::errorMatrix provides confusion matrix and class errors

```
errorMatrix(va_target, va_class, count=TRUE)
```

```
##       Predicted
## Actual    no  yes Error
##    no  19470 1041   5.1
##    yes  2585 2987  46.4
```

```
errorMatrix(va_target, va_class) %T>%
  print() ->
va_matrix
```

```
##       Predicted
```

```
## Actual   no  yes Error
##    no  74.6  4.0   5.1
##    yes  9.9 11.5  46.4
```

```
va_matrix %>%
  diag() %>%
  sum(na.rm=TRUE) %>%
  subtract(100, .) %>%
  sprintf("Overall error percentage = %s%%\n", .) %>%
  cat()
```

```
## Overall error percentage = 13.9%
```

```
va_matrix[,"Error"] %>%
  mean(na.rm=TRUE) %>%
  sprintf("Averaged class error percentage = %s%%\n", .) %>%
  cat()
```

```
## Averaged class error percentage = 25.75%
```