Medical insurance charges: Random forest model

Jeewoen Shin

02/16/2022

Load input data and check missing data

```
insurance.data.full = readr::read_csv("data/insurance.csv",
                              col_types = cols( age = col_integer(),
                              sex = col_factor(),
                              bmi = col_double(),
                              children = col_integer(),
                              smoker = col_factor(),
                              region = col_factor(),
                              charges = col_double()
));

# check missing data
sum(is.na(insurance.data.full))
# view raw data
insurance.data.full %>% tbl_summary() # reporting the median and IQR
```

```
## Table printed with `knitr::kable()`, not {gt}. Learn why at
## http://www.danieldsjoberg.com/gtsummary/articles/rmarkdown.html
## To suppress this message, include `message = FALSE` in code chunk header.
```

```
insurance.data.full %>% tbl_summary(statistic = list(all_continuous() ~ "{mean} ({sd})"))
```

```
## Table printed with `knitr::kable()`, not {gt}. Learn why at
## http://www.danieldsjoberg.com/gtsummary/articles/rmarkdown.html
## To suppress this message, include `message = FALSE` in code chunk header.
```

```
# reporting mean and SD

insurance.data <- insurance.data.full %>% mutate(id = row_number()) # person id
```

Split into training and test set

```r
set.seed(0)

# split training & test set
training_set = insurance.data %>% sample_frac(0.75)
test_set = anti_join(insurance.data, training_set, by='id')

training_set = training_set %>% select(-"id")
test_set = test_set %>% select(-"id")
```

Build random forest model (ranger)

```r
set.seed(1)
# importance = permutation:
rf.model.permutation <- ranger(charges ~ ., data = training_set,
                               importance = "permutation", mtry=3)

if(FALSE){ #importance = impurity: variance of the responses for regression
  rf.model.impurity <- ranger(cases ~ ., data = training_set,
                              importance = "impurity", mtry=3)
}
```

Note: mtry = the number of variables randomly sampled as candidates at each split. Default: floor(# variables/3) which is 2. Increase mtry to 3 instead of 2.

Print R squared and MSE

```r
print(rf.model.permutation)
```

```
## Ranger result
##
## Call:
##  ranger(charges ~ ., data = training_set, importance = "permutation",      mtry = 3)
##
## Type:                             Regression
## Number of trees:                  500
## Sample size:                      1004
## Number of independent variables:  6
## Mtry:                             3
## Target node size:                 5
## Variable importance mode:         permutation
## Splitrule:                        variance
## OOB prediction error (MSE):       19401629
## R squared (OOB):                  0.8669511
```

```r
#print(rf.model.impurity)
```

Print variable importance

```
rf.model.permutation$variable.importance
```

```
##          age          sex          bmi     children       smoker       region
##  30071825.44     96992.74  37822186.57   2877513.85 208794323.94    804440.97
```

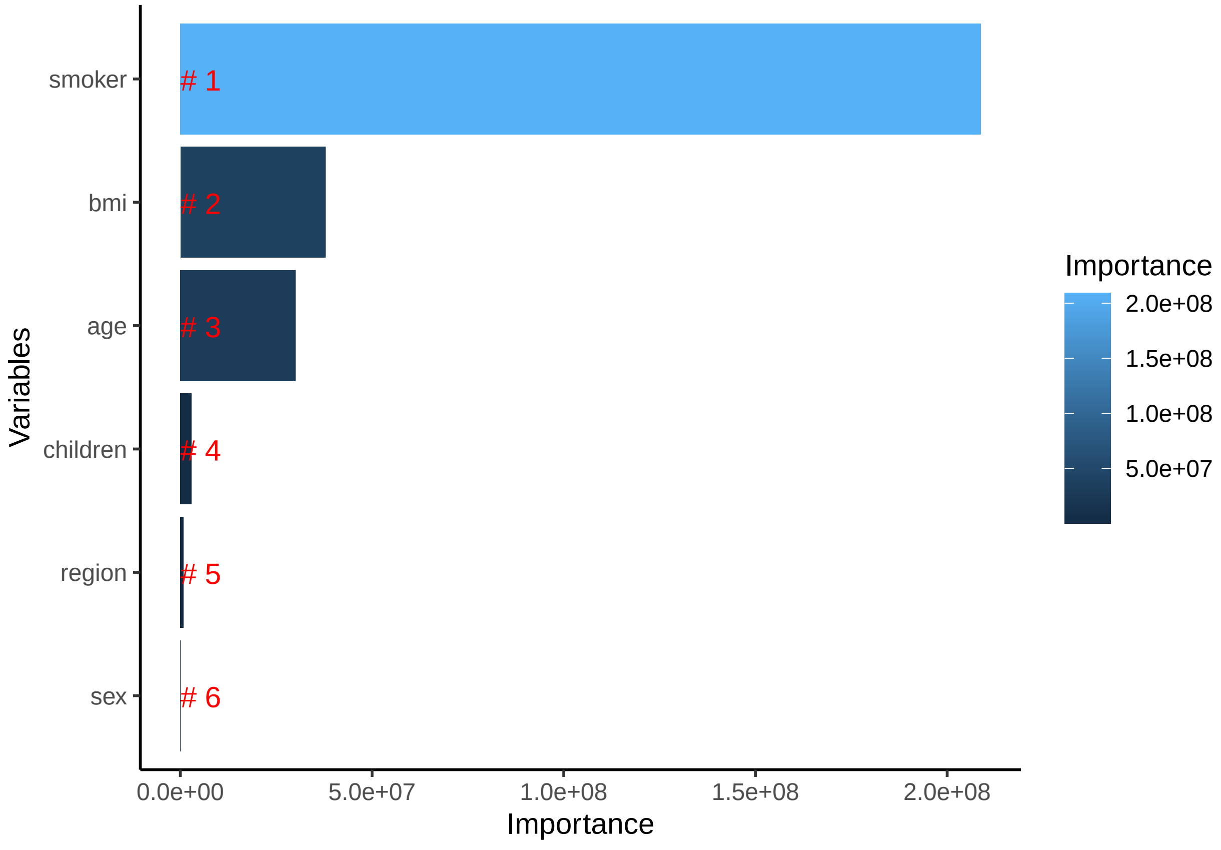
```
#rf.model.impurity$variable.importance
```

Plot variable importance

```
varImportance.permutation = data.frame(Variables = names(rf.model.permutation$variable.importance),
                                       Importance =round(rf.model.permutation$variable.importance,2))

rankImportance.permutation=varImportance.permutation%>%mutate(Rank=paste('#',dense_rank(desc(Importance)

ggplot(rankImportance.permutation,aes(x=reorder(Variables,Importance),
                                      y=Importance,fill=Importance))+
  geom_bar(stat='identity') +
  geom_text(aes(x = Variables, y = 0.5, label = Rank),
            hjust=0, vjust=0.55, size = 4, colour = 'red') +
  labs(x = 'Variables') +
  coord_flip() +
  theme_classic()
```



Model accuracy, Goodness-of-fit

```
training_set$predicted.rf.permutation <- predict(rf.model.permutation, dat = training_set)$predictions

test_set$predicted.rf.permutation <- predict(rf.model.permutation, dat = test_set)$predictions

# calculate root-mean-square deviation
rmse <- function(actual, estimate) {
  rmse = sqrt(sum((actual - estimate)^2) / length(actual))
  return(rmse)
}

print(rmse(training_set$charges, training_set$predicted.rf.permutation)) # training set
```

```
## [1] 2415.884
```

```
print(rmse(test_set$charges, test_set$predicted.rf.permutation)) # test set
```

```
## [1] 5252.589
```

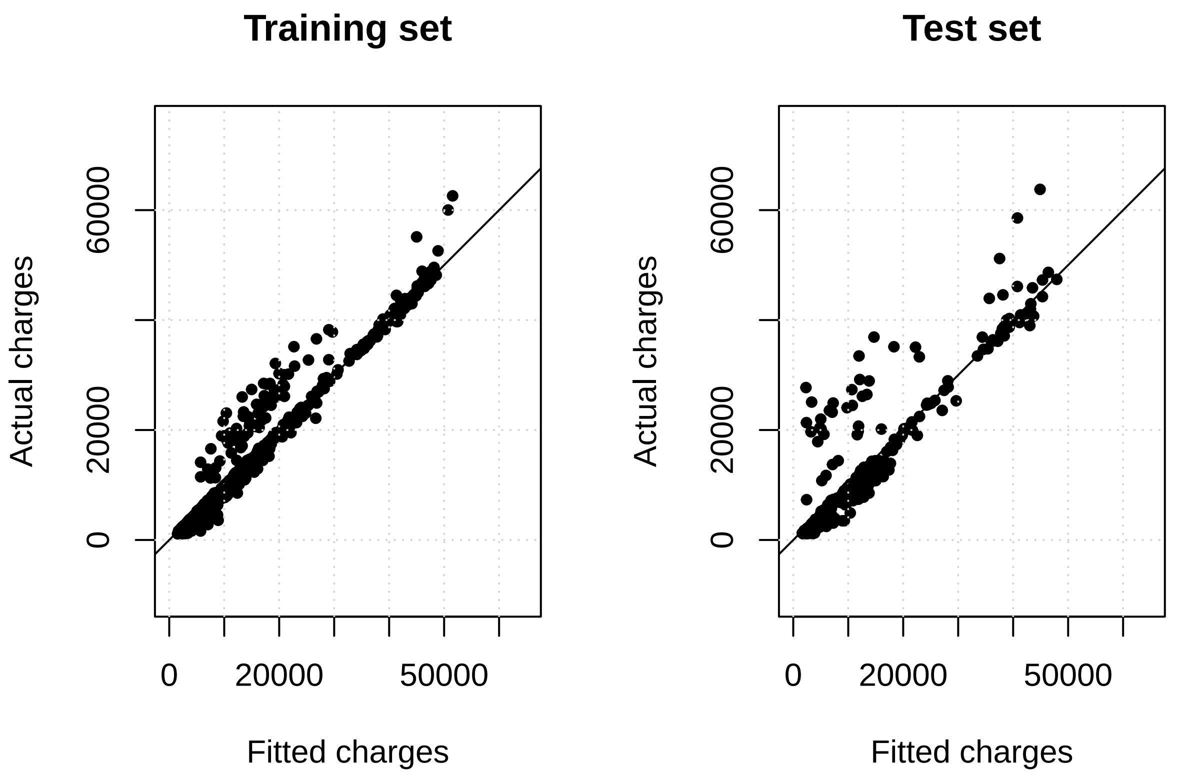
```
if(FALSE){ # impurity
  training_set$predicted.rf.impurity <- predict(rf.model.impurity, dat = training_set)$predictions
  sum((training_set$charges - training_set$predicted.rf.impurity)^2) / nrow(training_set)

  test_set$predicted.rf.impurity <- predict(rf.model.impurity, dat = test_set)$predictions
  sum((test_set$charges - test_set$predicted.rf.impurity)^2) / nrow(test_set)
}
```

Plot charges and predicted charges

```
par(mfrow=c(1,2))
plot(training_set$charges ~ training_set$predicted.rf.permutation, asp=1, pch=20, xlab="Fitted charges"
grid()
abline(0,1)

plot(test_set$charges ~ test_set$predicted.rf.permutation, asp=1, pch=20, xlab="Fitted charges", ylab="
grid()
abline(0,1)
```


Training set
Test set
Actual charges
Fitted charges
0
20000
50000
60000